



Red Hat build of Quarkus 1.11

Getting started with Quarkus

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This guide describes how to create a simple Quarkus application with Apache Maven.

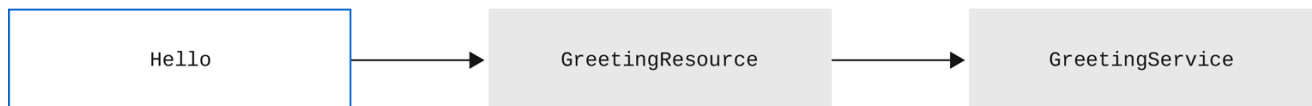
Table of Contents

PREFACE	3
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	4
MAKING OPEN SOURCE MORE INCLUSIVE	5
CHAPTER 1. RED HAT BUILD OF QUARKUS	6
CHAPTER 2. APACHE MAVEN AND QUARKUS	7
2.1. CONFIGURING THE MAVEN SETTINGS.XML FILE FOR THE ONLINE REPOSITORY	7
2.2. DOWNLOADING AND CONFIGURING THE QUARKUS MAVEN REPOSITORY	8
CHAPTER 3. CREATING THE GETTING STARTED PROJECT	11
CHAPTER 4. COMPILING AND STARTING THE QUARKUS GETTING STARTED PROJECT	14
CHAPTER 5. USING QUARKUS DEPENDENCY INJECTION	16
CHAPTER 6. TESTING YOUR QUARKUS APPLICATION WITH JUNIT	18
CHAPTER 7. PACKAGING AND RUNNING THE QUARKUS GETTING STARTED APPLICATION	21
CHAPTER 8. CREATING A QUARKUS MAVEN PROJECT USING CODE.QUARKUS.REDHAT.COM	22
CHAPTER 9. ADDITIONAL RESOURCES	26

PREFACE

As an application developer, you can use Red Hat build of Quarkus to create microservices-based applications written in Java that run in OpenShift and serverless environments. Applications compiled to native executables have small memory footprints and fast startup times.

This guide shows you how to use Apache Maven to create, test, package, and run a simple Quarkus project that exposes a **hello** HTTP endpoint. To demonstrate dependency injection, this endpoint uses a **greeting** bean.



78_OpenShift_0420

Prerequisites

- Have OpenJDK (JDK) 11 installed and the **JAVA_HOME** environment variable specifies the location of the Java SDK.
 - Log in to the Red Hat Customer Portal to download Red Hat build of Open JDK from the [Software Downloads](#) page.
- Have Apache Maven 3.6.2 or later installed. Maven is available from the [Apache Maven Project](#) website.



NOTE

For a completed example of the getting started exercise, download the [Quarkus quickstart archive](#) or clone the **Quarkus Quickstarts** Git repository. The Getting Started example is in the **getting-started** directory.

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. 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 startup times.

CHAPTER 2. APACHE MAVEN AND QUARKUS

Apache Maven is a distributed build automation tool used in Java application development to create, manage, and build software projects. Maven uses standard configuration files called Project Object Model (POM) files to define projects and manage the build process. POM files describe the module and component dependencies, build order, and targets for the resulting project packaging and output using an XML file. This ensures that the project is built in a correct and uniform manner.

Maven repositories

A Maven repository stores Java libraries, plug-ins, and other build artifacts. The default public repository is the Maven 2 Central Repository, but repositories can be private and internal within a company to share common artifacts among development teams. Repositories are also available from third-parties.

You can use the online Maven repository with your Quarkus projects or you can download the Red Hat build of Quarkus Maven repository.

Maven plug-ins

Maven plug-ins are defined parts of a POM file that achieve one or more goals. Quarkus applications use the following Maven plug-ins:

- Quarkus Maven plug-in (**quarkus-maven-plugin**): Enables Maven to create Quarkus projects, supports the generation of uber-JAR files, and provides a development mode.
- Maven Surefire plug-in (**maven-surefire-plugin**): Used during the test phase of the build life cycle to execute unit tests on your application. The plug-in generates text and XML files that contain the test reports.

2.1. CONFIGURING THE MAVEN `SETTINGS.XML` FILE FOR THE ONLINE REPOSITORY

You can use the online Quarkus repository with your Quarkus Maven project by configuring your user **settings.xml** file. This is the recommended approach. Maven settings used with a repository manager or repository on a shared server provide better control and manageability of projects.



NOTE

When you configure the repository by modifying the Maven **settings.xml** file, the changes apply to all of your Maven projects.

Procedure

1. Open the Maven `~/.m2/settings.xml` file in a text editor or integrated development environment (IDE).



NOTE

If there is not a **settings.xml** file in the `~/.m2/` directory, copy the **settings.xml** file from the `$MAVEN_HOME/.m2/conf/` directory into the `~/.m2/` directory.

2. Add the following lines to the `<profiles>` element of the **settings.xml** file:

```

<!-- Configure the Quarkus Maven repository -->
<profile>
  <id>red-hat-enterprise-maven-repository</id>
  <repositories>
    <repository>
      <id>red-hat-enterprise-maven-repository</id>
      <url>https://maven.repository.redhat.com/ga/</url>
      <releases>
        <enabled>>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>red-hat-enterprise-maven-repository</id>
      <url>https://maven.repository.redhat.com/ga/</url>
      <releases>
        <enabled>>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </pluginRepository>
  </pluginRepositories>
</profile>

```

3. Add the following lines to the **<activeProfiles>** element of the **settings.xml** file and save the file.

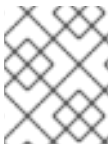
```

<activeProfile>red-hat-enterprise-maven-repository</activeProfile>

```

2.2. DOWNLOADING AND CONFIGURING THE QUARKUS MAVEN REPOSITORY

If you do not want to use the online Maven repository, you can download and configure the Quarkus Maven repository to create a Quarkus application with Maven. The Quarkus Maven repository contains many of the requirements that Java developers typically use to build their applications. This procedure describes how to edit the **settings.xml** file to configure the Quarkus Maven repository.



NOTE

When you configure the repository by modifying the Maven **settings.xml** file, the changes apply to all of your Maven projects.

Procedure

1. Download the Quarkus Maven repository ZIP file from the [Software Downloads](#) page of the Red Hat Customer Portal (login required).
2. Expand the downloaded archive.

3. Change directory to the `~/.m2/` directory and open the Maven `settings.xml` file in a text editor or integrated development environment (IDE).
4. Add the path of the Quarkus Maven repository that you downloaded to the `<profiles>` element of the `settings.xml` file. The format of the path of the Quarkus Maven repository must be `file://$PATH`, for example `file:///home/userX/rh-quarkus-1.11.7.GA-maven-repository/maven-repository`.

```

<!-- Configure the Quarkus Maven repository -->
<profile>
  <id>red-hat-enterprise-maven-repository</id>
  <repositories>
    <repository>
      <id>red-hat-enterprise-maven-repository</id>
      <url>file:///path/to/Quarkus/Maven/repository/</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>red-hat-enterprise-maven-repository</id>
      <url>file:///path/to/Quarkus/Maven/repository/</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </pluginRepository>
  </pluginRepositories>
</profile>

```

5. Add the following lines to the `<activeProfiles>` element of the `settings.xml` file and save the file.

```

<activeProfile>red-hat-enterprise-maven-repository</activeProfile>

```



IMPORTANT

If your Maven repository contains outdated artifacts, you might encounter one of the following Maven error messages when you build or deploy your project, where **<artifact_name>** is the name of a missing artifact and **<project_name>** is the name of the project you are trying to build:

- **Missing artifact <project_name>**
- **[ERROR] Failed to execute goal on project <artifact_name>; Could not resolve dependencies for <project_name>**

To resolve the issue, delete the cached version of your local repository located in the **~/.m2/repository** directory to force a download of the latest Maven artifacts.

CHAPTER 3. CREATING THE GETTING STARTED PROJECT

The **getting-started** project lets you get up and running with a simple Quarkus application using Apache Maven and the Quarkus Maven plug-in.

Procedure

1. In a command terminal, enter the following command to verify that Maven is using JDK 11 and that the Maven version is 3.6.2 or higher:

```
mvn --version
```

2. If the preceding command does not return JDK 11, add the path to JDK 11 to the PATH environment variable and enter the preceding command again.
3. To generate the project, enter one of the following commands:



NOTE

Apple macOS and Microsoft Windows are not supported production environments.

- If you are using Linux or Apple macOS, enter the following command:

```
mvn io.quarkus:quarkus-maven-plugin:1.11.7.Final-redhat-00009:create \
  -DprojectId=org.acme \
  -DprojectId=getting-started \
  -DplatformGroupId=com.redhat.quarkus \
  -DplatformVersion=1.11.7.Final-redhat-00009 \
  -DclassName="org.acme.quickstart.GreetingResource" \
  -Dpath="/hello"
cd getting-started
```

- If you are using the Microsoft Windows command line, enter the following command:

```
mvn io.quarkus:quarkus-maven-plugin:1.11.7.Final-redhat-00009:create -
  DprojectId=org.acme -DprojectId=getting-started -
  DplatformGroupId=com.redhat.quarkus -DplatformVersion=1.11.7.Final-redhat-00009 -
  DclassName="org.acme.quickstart.GreetingResource" -Dpath="/hello"
```

- If you are using the Microsoft Windows Powershell, enter the following command:

```
mvn io.quarkus:quarkus-maven-plugin:1.11.7.Final-redhat-00009:create "-
  DprojectId=org.acme" "-DprojectId=getting-started" "-
  DplatformVersion=1.11.7.Final-redhat-00009" "-DplatformGroupId=com.redhat.quarkus"
  "-DclassName=org.acme.quickstart.GreetingResource" "-Dpath=/hello"
```

These commands create the following elements in the **./getting-started** directory:

- The Maven project directory structure
- An **org.acme.quickstart.GreetingResource** resource exposed on **/hello**

- Associated unit tests for testing your application in native mode and JVM mode
 - A landing page that is accessible on **http://localhost:8080** after you start the application
 - Example **Dockerfile.jvm**, **Dockerfile.native**, and **Dockerfile.fast-jar** files in the **src/main/docker** directory
 - The application configuration file
4. After the directory structure is created, open the **pom.xml** file in a text editor and examine the contents of the file:

```

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.redhat.quarkus</groupId>
      <artifactId>quarkus-universe-bom</artifactId>
      <version>${quarkus.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<build>
  <plugins>
    <plugin>
      <groupId>io.quarkus</groupId>
      <artifactId>quarkus-maven-plugin</artifactId>
      <version>${quarkus-plugin.version}</version>
      <executions>
        <execution>
          <goals>
            <goal>build</goal>
            <goal>generate-code</goal>
            <goal>generate-code-tests</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>

```

The Quarkus BOM is included in the **<dependencyManagement>** section of the **pom.xml** file. Therefore, you do not need to list the versions of individual Quarkus dependencies in the **pom.xml** file. In addition, you can see the **quarkus-maven-plugin** plug-in that is responsible for packaging the application and providing the development mode.

5. Review the **quarkus-resteasy** dependency in the **pom.xml** file. This dependency enables you to develop REST applications:

```

<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-resteasy</artifactId>
</dependency>

```


6. Review the `src/main/java/org/acme/quickstart/GreetingResource.java` file:

```
package org.acme.quickstart;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/hello")
public class GreetingResource {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String hello() {
        return "hello";
    }
}
```

This file contains a simple REST endpoint that returns **hello** as a response to a request that you send to the **/hello** endpoint.



NOTE

With Quarkus, the **Application** class for JAX-RS is supported but not required. In addition, only one instance of the **GreetingResource** class is created and not one per request. You can configure this by using different ***Scoped** annotations, for example **ApplicationScoped**, **RequestScoped**, and so forth.

You can create a Quarkus Maven project using the code.quarkus.redhat.com project generator. See [Creating a Quarkus Maven project using code.quarkus.redhat.com](#) for details.

CHAPTER 4. COMPILING AND STARTING THE QUARKUS GETTING STARTED PROJECT

After you have created the Quarkus Getting Started project, you can compile the Hello application and verify that the **hello** endpoint returns **hello**.

This example uses the Quarkus built-in development mode. In development mode, you can update the application sources and configurations while your application is running. Your changes will appear in the running application.

Prerequisites

- You have created the Quarkus Getting Started project.

Procedure

1. To compile the Quarkus Hello application in development mode, enter the following command from the project directory:

```
./mvnw compile quarkus:dev
```

The following example shows the output of this command:

```
[INFO] -----< org.acme:getting-started >-----
[INFO] Building getting-started 1.0.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ getting-started ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/starksm/Dev/JBoss/Quarkus/starksm64-quarkus-quickstarts/getting-started/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ getting-started ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to /Users/starksm/Dev/JBoss/Quarkus/starksm64-quarkus-quickstarts/getting-started/target/classes
[INFO]
[INFO] --- quarkus-maven-plugin:<version>:dev (default-cli) @ getting-started ---
Listening for transport dt_socket at address: 5005
2019-02-28 17:05:22,347 INFO [io.qua.dep.QuarkusAugmentor] (main) Beginning quarkus augmentation
2019-02-28 17:05:22,635 INFO [io.qua.dep.QuarkusAugmentor] (main) Quarkus augmentation completed in 288ms
2019-02-28 17:05:22,770 INFO [io.quarkus] (main) Quarkus started in 0.668s. Listening on: http://localhost:8080
2019-02-28 17:05:22,771 INFO [io.quarkus] (main) Installed features: [cdi, resteasy]
```

2. Enter the following command in a new terminal window to send a request to the endpoint provided by the application:

```
curl -w "\n" http://localhost:8080/hello
hello
```

**NOTE**

This example uses the "`\n`" attribute to automatically add a new line before the output of the command. This prevents your terminal from printing a '%' character or putting both the result and the next command prompt on the same line.

3. Press **CTRL+C** to stop the application.

CHAPTER 5. USING QUARKUS DEPENDENCY INJECTION

Dependency injection enables a service to be used in a way that is completely independent of any client consumption. It separates the creation of client dependencies from the client's behavior, which enables program designs to be loosely coupled.

Dependency injection in Red Hat build of Quarkus is based on Quarkus ArC which is a CDI-based build-time oriented dependency injection solution tailored for Quarkus architecture. Because ArC is a transitive dependency of **quarkus-resteasy**, and **quarkus-resteasy** is a dependency of your project, ArC will already be downloaded.

Prerequisites

- You have created the Quarkus Getting Started project.

Procedure

1. To modify the application and add a companion bean, create the **src/main/java/org/acme/quickstart/GreetingService.java** file with the following content:

```
package org.acme.quickstart;

import javax.enterprise.context.ApplicationScoped;

@ApplicationScoped
public class GreetingService {

    public String greeting(String name) {
        return "hello " + name;
    }

}
```

2. Edit the **src/main/java/org/acme/quickstart/GreetingResource.java** to inject the **GreetingService** and create a new endpoint using it:

```
package org.acme.quickstart;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import org.jboss.resteasy.annotations.jaxrs.PathParam;

@Path("/hello")
public class GreetingResource {

    @Inject
    GreetingService service;

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @Path("/greeting/{name}")
```

```
public String greeting(@PathParam String name) {  
    return service.greeting(name);  
}  
  
@GET  
@Produces(MediaType.TEXT_PLAIN)  
public String hello() {  
    return "hello";  
}  
}
```

3. If you stopped the application, enter the following command to restart it:

```
./mvnw compile quarkus:dev
```

4. To verify that the endpoint returns **hello quarkus**, enter the following command in a new terminal window:

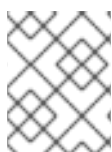
```
curl -w "\n" http://localhost:8080/hello/greeting/quarkus  
hello quarkus
```

CHAPTER 6. TESTING YOUR QUARKUS APPLICATION WITH JUNIT

After you compile your Quarkus Getting Started project, test your application with the JUnit 5 framework to ensure that it runs as expected. There are two test dependencies in the Quarkus project generated **pom.xml** file:

```
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-junit5</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>rest-assured</artifactId>
  <scope>test</scope>
</dependency>
```

The **quarkus-junit5** dependency is required for testing because it provides the **@QuarkusTest** annotation that controls the JUnit 5 testing framework. The **rest-assured** dependency is not required but because it provides a convenient way to test HTTP endpoints, it is integrated as well. It automatically sets the correct URL so no configuration is required.



NOTE

These tests use the REST-assured framework, but you can use a different library if you prefer.

Prerequisites

- You have compiled the Quarkus Getting Started project.

Procedure

1. Open the generated **pom.xml** file and review the contents:

```
<plugin>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>${surefire-plugin.version}</version>
  <configuration>
    <systemPropertyVariables>

    <java.util.logging.manager>org.jboss.logmanager.LogManager</java.util.logging.manager>
      <maven.home>${maven.home}</maven.home>
    </systemPropertyVariable>
  </configuration>
</plugin>
```

Note, that:

- the **java.util.logging.manager** system property is set to ensure that you application uses the correct log manager for the test.

- the **maven.home** property points to the location of the **settings.xml** file in which you can store custom Maven configuration that you want to apply to your project.
2. Edit the **src/test/java/org/acme/quickstart/GreetingResourceTest.java** file to match the following content:

```

package org.acme.quickstart;

import io.quarkus.test.junit.QuarkusTest;
import org.junit.jupiter.api.Test;

import java.util.UUID;

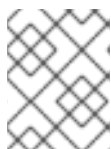
import static io.restassured.RestAssured.given;
import static org.hamcrest.CoreMatchers.is;

@QuarkusTest
public class GreetingResourceTest {

    @Test
    public void testHelloEndpoint() {
        given()
            .when().get("/hello")
            .then()
                .statusCode(200)
                .body(is("hello"));
    }

    @Test
    public void testGreetingEndpoint() {
        String uuid = UUID.randomUUID().toString();
        given()
            .pathParam("name", uuid)
            .when().get("/hello/greeting/{name}")
            .then()
                .statusCode(200)
                .body(is("hello " + uuid));
    }
}

```

**NOTE**

By using the **QuarkusTest** runner, you instruct JUnit to start the application before starting the tests.

3. To run these tests from Maven, enter the following command:

```
./mvnw test
```

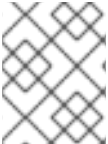
**NOTE**

You can also run the test from your IDE. If you do this, make sure to stop the application first.

By default, tests run on port **8081** so they do not conflict with the running application. In Quarkus, the **RestAssured** dependency is configured to use this port. If you want to use a different client, use the **@TestHTTPResource** annotation to directly inject the URL of the tested application into a field on the **Test** class. This field can be of the type **String**, **URL** or **URI**. You can also provide the test path in this annotation. For example, to test a servlet mapped to **/myservlet**, add the following lines to your test:

```
@TestHTTPResource("/myservlet")  
URL testUrl;
```

4. If necessary, specify the test port in the **quarkus.http.test-port** configuration property.



NOTE

Quarkus also creates a system property called **test.url** that is set to the base test URL for situations where you cannot use injection.

CHAPTER 7. PACKAGING AND RUNNING THE QUARKUS GETTING STARTED APPLICATION

After you compile your Quarkus Getting Started project, you can package it in a JAR file and run it from the command line.

Prerequisites

- You have compiled the Quarkus Getting Started project.

Procedure

1. To package your Quarkus Getting Started project, enter the following command in the **root** directory:

```
./mvnw package
```

This command produces the following JAR files in the **/target** directory:

- **getting-started-1.0-0-SNAPSHOT.jar**: Contains the classes and resources of the projects. This is the regular artifact produced by the Maven build.
- **getting-started-1.0-0-SNAPSHOT-runner.jar**: Is an executable JAR file. Be aware that this file is not an uber-JAR file because the dependencies are copied into the **target/lib** directory.



WARNING

When your application is running in development mode, you must press **CTRL+C** to stop your application. You will encounter a port conflict when you try to package your application when development mode is enabled.

2. Enter the following command to start your application:

```
java -jar target/getting-started-1.0-0-SNAPSHOT-runner.jar
```



NOTE

The **Class-Path** entry of the **MANIFEST.MF** file from the **runner** JAR file explicitly lists the JAR files from the **lib** directory. If you want to deploy your application from another location, you must copy the **runner** JAR file as well as the **lib** directory.

CHAPTER 8. CREATING A QUARKUS MAVEN PROJECT USING CODE.QUARKUS.REDHAT.COM

As an application developer, you can use `code.quarkus.redhat.com` to generate a Quarkus Maven project and automatically add and configure the extensions that you want to use in your application. In addition, `code.quarkus.redhat.com` automatically manages the configuration parameters required to compile your project into a native executable.

This section walks you through the process of generating a Quarkus Maven project including:

- Specifying basic details about your application.
- Choosing the extensions that you want to include in your project.
- Generating a downloadable archive with your project files.
- Using the custom commands for compiling and starting your application.

Prerequisites

- Have a web browser.

Procedure

1. Navigate to <https://code.quarkus.redhat.com> using a web browser.
2. Specify basic details about your project:
 - a. Enter a group name for your project. The format of the name follows the Java package naming convention, for example, **org.acme**.
 - b. Enter a name that you want to use for Maven artifacts generated from your project, for example **code-with-quarkus**.
 - c. Select the build tool that you want to use to compile and start your application. The build tool that you choose determines:
 - the directory structure of your generated project.
 - the format of configuration files used in your generated project.
 - the custom build script and command for compiling and starting your application that `code.quarkus.redhat.com` displays for you after you generate your project.




NOTE

Red Hat provides support for using `code.quarkus.redhat.com` to create Quarkus Maven projects only. Generating Gradle projects is not supported by Red Hat.

Configure your application details

Group	org.acme	
Artifact	code-with-quarkus	
Build Tool	Maven	<input type="checkbox"/> CONFIGURE MORE OPTIONS

3. Specify additional details about your application project:
 - a. Select *Configure more options* to display the fields that contain additional application details.
 - b. Enter a version that is used in artifacts generated from your project. The default value of this field is **1.0.0-SNAPSHOT**. Using [semantic versioning](#) is recommended, but you can use a different type of versioning, if you prefer.
 - c. Select whether you want code.quarkus.redhat.com to add example code to your project.

When you add extensions that are marked with the  icon to your project from the list of extensions, you can enable this option to automatically create example class files and resource files for those extensions when you generate your project. When you do not add any extensions that provide example code, this option does not affect your generated project.


Configure your application details

Group	org.acme	Version	1.0.0-SNAPSHOT
Artifact	code-with-quarkus	<input checked="" type="checkbox"/> Example Code Yes, Please	<input type="checkbox"/> CLOSE
Build Tool	Maven		



NOTE

code.quarkus.redhat.com automatically uses the latest release of Red Hat build of Quarkus. You can manually change the BOM version in the **pom.xml** file after you generate your project.

4. Select the extensions that you want to use in your application from the list of extensions. The selected extensions are included as dependencies of your Quarkus application with their versions being managed by the Quarkus platform to ensure their compatibility. You can enable the option to automatically generate example code for extensions that are marked with the  icon.



Pick your extensions

Selected Extensions

REST Client SUPPORTED 


YAML Configuration SUPPORTED 

Web

- | | | | |
|-------------------------------------|---|---|---|
| <input type="checkbox"/> | RESTEasy JAX-RS SUPPORTED  | REST endpoint framework implementing JAX-RS and more | ⋮ |
| <input type="checkbox"/> | RESTEasy Jackson SUPPORTED  | Jackson serialization support for RESTEasy | ⋮ |
| <input type="checkbox"/> | RESTEasy JSON-B SUPPORTED | JSON-B serialization support for RESTEasy | ⋮ |
| <input type="checkbox"/> | Eclipse Vert.x GraphQL TECH-PREVIEW | Query the API using GraphQL | ⋮ |
| <input type="checkbox"/> | Hibernate Validator SUPPORTED | Validate object properties (field, getter) and method parameters for y... | ⋮ |
| <input type="checkbox"/> | Mutiny support for REST Client TECH-PREVIEW | Enable Mutiny for the REST client | ⋮ |
| <input checked="" type="checkbox"/> | REST Client SUPPORTED | Call REST services | ⋮ |
| <input type="checkbox"/> | REST Client JAXB SUPPORTED | Enable XML serialization for the REST Client | ⋮ |

















Note, that Red Hat provides [different levels](#) of support for individual extensions on the list, which are indicated by labels next to the name of each extension:






- *SUPPORTED* extensions are fully supported by Red Hat for use in enterprise application in production environments.
- *TECH-PREVIEW* extensions are subject to limited support by Red Hat in production environments under the [Technology Preview Features Support Scope](#).
- *DEV-SUPPORT* extensions are not supported by Red Hat for use in production environments, but the core functionalities that they provide are supported by Red Hat developers for use in developing new applications.
- Unlabeled extensions are not supported by Red Hat for use in production environments.
- *DEPRECATED* extension are planned to be replaced by a newer technology or implementation that provides the same functionality.

You can expand the overflow menu () next to each of the extensions to access additional options that you can use to:

- add the extension to an existing project using the Quarkus maven plugin on the command line.
- copy an XML snippet to add the the extension to the **pom.xml** file of a project.
- obtain the **groupid**, **artifactId** and **version** of each extension.
- open the extension guide.

Web

<input checked="" type="checkbox"/>	RESTEasy JAX-RS	SUPPORTED		REST endpoint framework implementing JAX-RS and more	
<input type="checkbox"/>	RESTEasy Jackson	SUPPORTED		Jackson serialization support for RESTEasy	
<input type="checkbox"/>	RESTEasy JSON-B	SUPPORTED		JSON-B serialization support for RESTEasy	
<input type="checkbox"/>	Eclipse Vert.x GraphQL	TECH-PREVIEW		Query the API using GraphQL	
<input type="checkbox"/>	Hibernate Validator	SUPPORTED		Validate object properties (field, getter) and method parameters for y...	
<input type="checkbox"/>	Mutiny support for REST Client	TECH-PREVIEW		Enable Mutiny for the REST client	
<input checked="" type="checkbox"/>	REST Client	SUPPORTED		Call REST services	
<input type="checkbox"/>	REST Client JAXB	SUPPORTED		Enable XML se	
<input checked="" type="checkbox"/>	REST Client JSON-B	SUPPORTED		Enable JSON-f	
<input type="checkbox"/>	REST Client Jackson	SUPPORTED		Enable Jackson	
<input type="checkbox"/>	REST resources for Hibernate ORM with P...	TECH-PREVIEW		Generate JAX-	
<input type="checkbox"/>	REST resources for MongoDB with Panache			Generate JAX-	
<input type="checkbox"/>	RESTEasy JAXB	SUPPORTED		XML serializati	
<input type="checkbox"/>	RESTEasy Multipart	SUPPORTED		Multipart support for RESTEasy	

-  Copy the command to add it with Maven
-  Copy the command to add it with Gradle
-  Copy the extension pom.xml snippet
-  Copy the extension GAV
-  Open Extension Guide

5. Select *Generate your application* to confirm your choices and display the overlay screen with the download link for the archive that contains your generated project. The overlay screen also shows the custom command that you can use to compile and start your application.
6. Select *Download the ZIP* to save the archive with the generated project files to your machine.
7. Extract the contents of the archive.
8. Navigate to the directory that contains your extracted project files:

```
cd <directory_name>
```

9. Compile and start your application in development mode:

```
./mvnw compile quarkus:dev
```

CHAPTER 9. ADDITIONAL RESOURCES

- [Developing and compiling your Quarkus applications with Apache Maven](#)
- [Using Quarkus development mode](#)
- [Configuring your Quarkus applications](#)
- [Compiling your Quarkus applications to native executables](#)
- [Deploying your Quarkus applications to OpenShift](#)
- [Testing your Quarkus applications](#)
- [Apache Maven Project](#)
- [Guide to naming conventions on groupId, artifactId, and version](#)
- [JUnit 5 website](#)
- [REST-assured website](#)

Revised on 2021-06-22 13:03:35 UTC