



Ultimate

Quarkus

Last modified: 07 September 2023

Required plugin: **Quarkus** (bundled)

[Quarkus](#) [↗] is a Kubernetes-native Java framework mainly aimed at building microservices. IntelliJ IDEA provides the following:

- Coding assistance specific to Quarkus.
- Integration with the [Bean Validation](#), [CDI](#), and [Endpoints](#) tool windows.
- A dedicated project creation wizard based on code.quarkus.io [↗].
- A dedicated run configuration for Quarkus applications.


Create a new Quarkus project

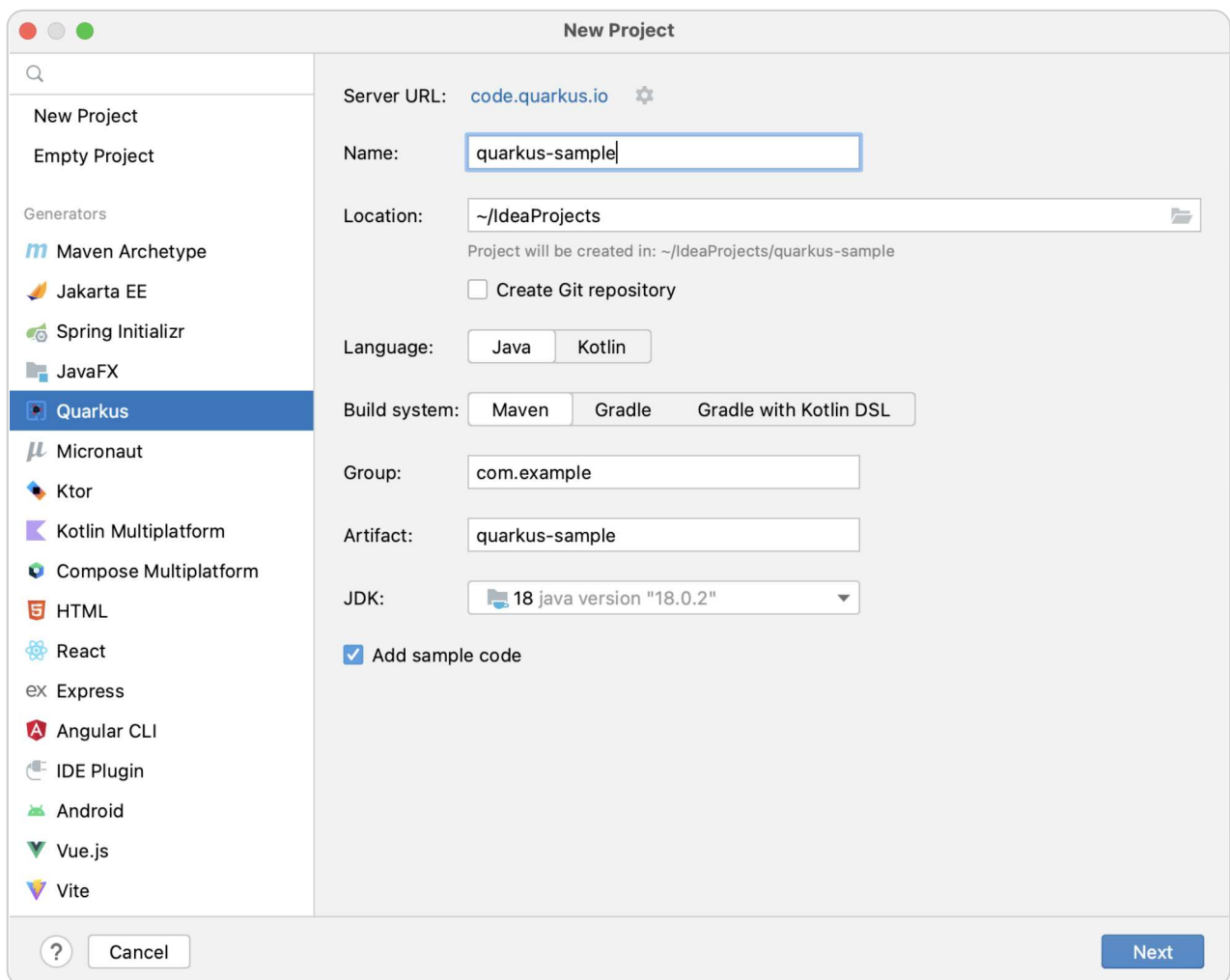
1. Launch IntelliJ IDEA.

If the Welcome screen opens, click **New Project**.

Otherwise, go to **File | New | Project**.

2. Select **Quarkus** from the left pane.

- Click  to enter the URL of the service that you want to use, or leave the default one.
- Specify a name and location for your project and configure project metadata: select a language, a build tool, and specify an artifact ID.
- From the **JDK** list, select the JDK that you want to use in your project.
If the JDK is installed on your computer, but not defined in the IDE, select **Add JDK** and specify the path to the JDK home directory.
If you don't have the necessary JDK on your computer, select **Download JDK**.
- Select the **Add sample code** option to create a REST endpoint named `ExampleResource` together with the project.



Click **Next**.

3. On the next step of the wizard, from the **Extensions** list, select the necessary options and click **Create**.

If you have selected the **Add sample code** option in the **New Project** wizard, the generated project will contain a REST endpoint named `ExampleResource` with the following code:

```
package com.example;

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

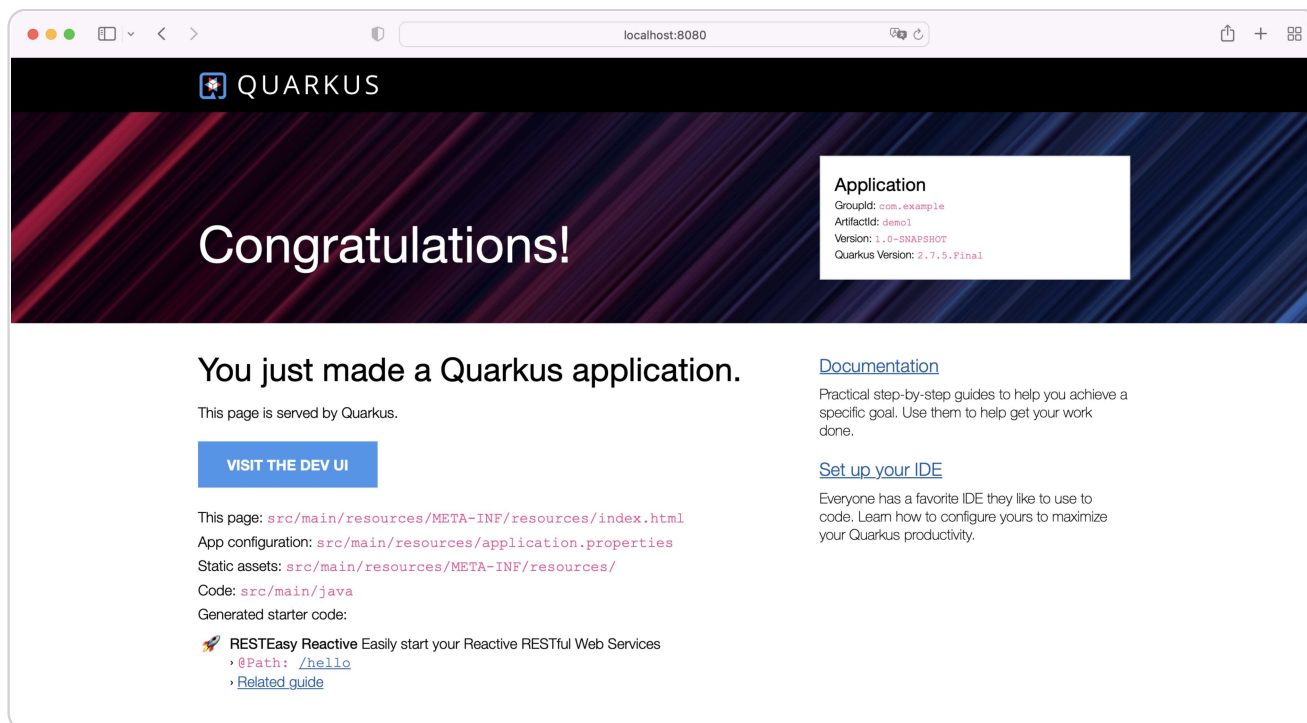
@Path("/hello")
public class ExampleResource {

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

You can open the Endpoints tool window (**View | Tool Windows | Endpoints**) and see this endpoint:

| Endpoints | | Documentation | HTTP Client | OpenAPI |
|-----------------------------------|-----------|---------------------------------|-------------|---------|
| Module: All | Type: All | Framework: All | | |
| hello-quarkus | | com.example.ExampleResource | | |
| /hello [GET] | | @GET | | |
| HTTP Server JEE RESTful WS Server | | @Produces(MediaType.TEXT_PLAIN) | | |
| | | public String hello() | | |

By default, the application starts on <http://localhost:8080> [↗]. Open this address in a web browser to see the Quarkus landing page:



If you open the `http://localhost:8080/hello` [↗] endpoint, you will see the string `Hello` from RESTEasy Reactive .

The default configuration runs your Quarkus application in development mode, which enables background compilation. For example, you can change the string returned by the `hello()` method in the `ExampleResource` class to `Hello` from a modified Quarkus endpoint and you will see this new string after you refresh `http://localhost:8080/hello` [↗] without restarting your application.

Debug the Quarkus application

To debug a running Quarkus application, attach the debugger to it.


1. Set a breakpoint in your code.


For example, you can set it on the line with the `return` statement in the `hello()` method.

2. Go to **Run | Attach to Process**.

3. From the list of Java processes, select the process of your Quarkus application.

If successful, IntelliJ IDEA will open the **Debug** tool window with the established debugger connection.

4. Now open <http://localhost:8080/hello> ↗ to call the `hello()` method. The debugger should stop at the breakpoint just before returning the greeting string.
5. In the **Debug** tool window, click  `F9` to continue the execution and return the string to the web browser.

To detach the debugger, click  `Ctrl F2`. This does not stop the actual application process, only detaches the debugger from it.