



# Red Hat CodeReady Studio 12.21

## Getting Started with CodeReady Studio Tools

Introduction to using Red Hat CodeReady Studio tools



# Red Hat CodeReady Studio 12.21 Getting Started with CodeReady Studio Tools

---

Introduction to using Red Hat CodeReady Studio tools

Eva-Lotte Gebhardt  
egebhard@redhat.com

Levi Valeeva  
levi@redhat.com

Yana Hontyk  
yhontyk@redhat.com

Supriya Takkhi

## 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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> 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<sup>®</sup> 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 compilation of topics contains information on how to start using Red Hat CodeReady Studio Tools for efficient development.

## Table of Contents

<b>MAKING OPEN SOURCE MORE INCLUSIVE</b> .....	<b>4</b>
<b>CHAPTER 1. GIT BASICS IN CODEREADY STUDIO</b> .....	<b>5</b>
1.1. SETTING UP GIT PERSPECTIVE	5
1.2. MANAGING REPOSITORIES IN GIT PERSPECTIVE	7
1.2.1. Creating a new Git repository	7
1.2.2. Adding an existing local Git repository	8
1.2.3. Cloning an existing Git repository	9
1.2.4. Adding a remote for the repository	11
1.3. MANAGING BRANCHES IN GIT PERSPECTIVE	16
1.3.1. Creating a new branch	16
1.3.2. Working in the branch	19
1.3.3. Updating your local repository	20
1.4. COMMITTING AND PUSHING CHANGES	21
<b>CHAPTER 2. MAVEN BASICS IN CODEREADY STUDIO</b> .....	<b>23</b>
2.1. CREATING A NEW MAVEN PROJECT	23
2.2. IMPORTING EXISTING MAVEN PROJECTS	27
2.2.1. Importing an existing locally stored Maven project	28
2.2.2. Importing an existing remotely stored Maven project	30
2.3. CREATING A NEW MAVEN MODULE	32
2.4. ADDING A MAVEN DEPENDENCY TO A MAVEN PROJECT	35
2.5. ADDING MAVEN SUPPORT TO AN EXISTING NON-MAVEN PROJECT	37
2.6. ADDITIONAL RESOURCES	39
<b>CHAPTER 3. APPLICATION DEPLOYMENT IN CODEREADY STUDIO</b> .....	<b>40</b>
3.1. CONFIGURING A LOCAL SERVER	40
3.2. CONFIGURING A REMOTE SERVER	43
3.3. DEPLOYING AN APPLICATION	47
<b>CHAPTER 4. JBOSS EAP AND JBOSS WFK BASICS IN CODEREADY STUDIO</b> .....	<b>50</b>
4.1. CONFIGURING MAVEN REPOSITORIES	50
4.2. SETTING UP JBOSS EAP	53
<b>CHAPTER 5. OPENSIFT BASICS IN CODEREADY STUDIO</b> .....	<b>61</b>
5.1. SETTING UP THE OPENSIFT APPLICATION EXPLORER VIEW	61
5.2. CONNECTING TO AN OPENSIFT CLUSTER USING OPENSIFT APPLICATION EXPLORER	63
5.3. CONNECTING TO AN OPENSIFT CLUSTER USING BROWSER-BASED TOKEN RETRIEVAL	65
5.3.1. Pasting your login command	65
5.3.2. Retrieving your token	66
5.4. SETTING UP A DEVELOPER SANDBOX USING OPENSIFT TOOLS	67
5.5. BUILDING AN APPLICATION BASED ON DEVFILES	69
5.5.1. Managing your devfile registries	74
5.5.1.1. Adding a devfile registry	74
5.5.1.2. Deleting a devfile registry	75
5.5.1.3. Editing a devfile registry	76
5.5.1.4. Creating a component from your devfile registry	77
5.6. BUILDING AN APPLICATION BASED ON S2I FILES	79
5.7. DEPLOYING A COMPONENT ON A CLUSTER USING OPENSIFT APPLICATION EXPLORER	84
5.8. DEFINING AN EXTERNAL ACCESS URL USING OPENSIFT APPLICATION EXPLORER	84
5.9. DEBUGGING AN APPLICATION ON A CLUSTER USING OPENSIFT APPLICATION EXPLORER	86
5.10. CREATING APPLICATION SERVICES USING OPENSIFT APPLICATION EXPLORER	87

<b>CHAPTER 6. QUARKUS TOOLS BASICS IN CODEREADY STUDIO</b> .....	<b>90</b>
6.1. CREATING A NEW QUARKUS PROJECT	90
6.2. RUNNING A QUARKUS APPLICATION	93
6.3. DEBUGGING A QUARKUS APPLICATION	95
6.4. USING LANGUAGE SUPPORT IN CODEREADY STUDIO	97
6.4.1. Using Quarkus content assist	97
6.4.2. Enabling language support for MicroProfile REST Client properties	99
<b>CHAPTER 7. HIBERNATE TOOLS BASICS IN CODEREADY STUDIO</b> .....	<b>103</b>
7.1. CREATING A NEW JAKARTA PERSISTENCE PROJECT	103
7.2. ADDING LIBRARIES	111
7.3. GENERATING TABLES FROM ENTITIES	114
7.4. CREATING A HIBERNATE MAPPING FILE	116
7.5. CREATING A HIBERNATE CONFIGURATION FILE	119
7.6. CREATING A HIBERNATE CONSOLE CONFIGURATION FILE	122
7.7. EDITING HIBERNATE PROJECT CONFIGURATIONS	128
7.8. SETTING PREFERENCES FOR HIBERNATE RUNTIME IMPLEMENTATIONS	133
<b>CHAPTER 8. MOBILE WEB TOOLS BASICS IN CODEREADY STUDIO</b> .....	<b>134</b>
8.1. CREATING AN HTML5 PROJECT	134
8.2. ADDING A NEW HTML5 JQUERY MOBILE FILE	136
8.3. ADDING A NEW MOBILE PAGE	140



## 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. GIT BASICS IN CODEREADY STUDIO

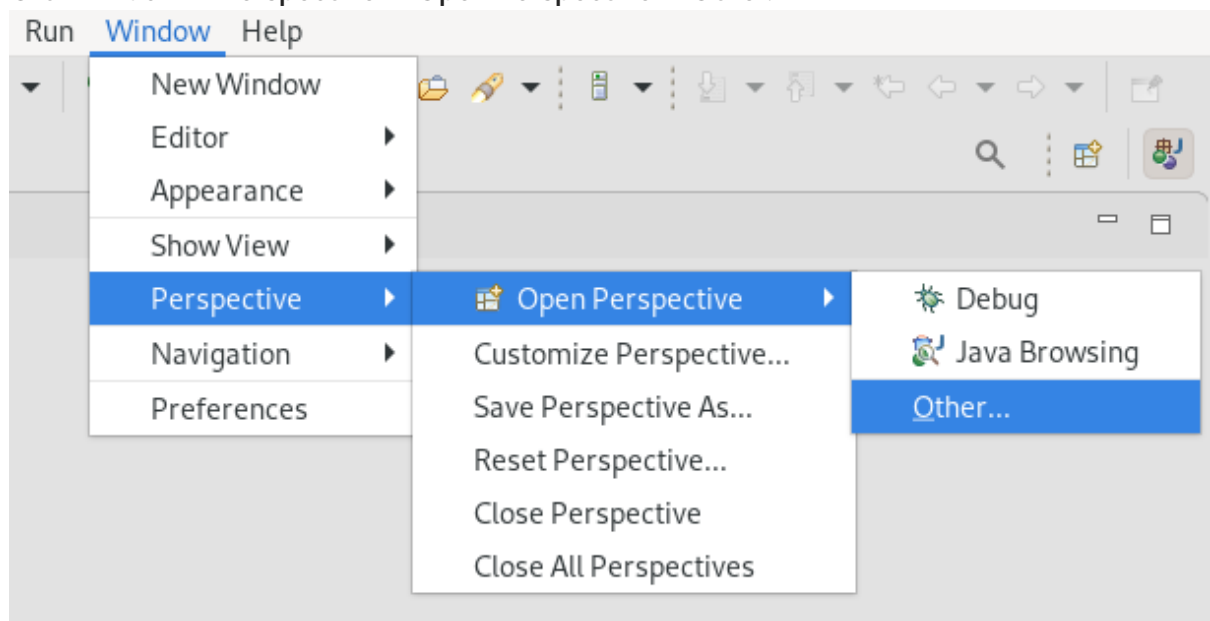
CodeReady Studio includes Git Perspective, which allows developers to manage their Git repositories from a graphical interface. The following section outlines the basic workflow of a Git project in Git Perspective and describes how to accomplish the most common Git-related tasks.

## 1.1. SETTING UP GIT PERSPECTIVE

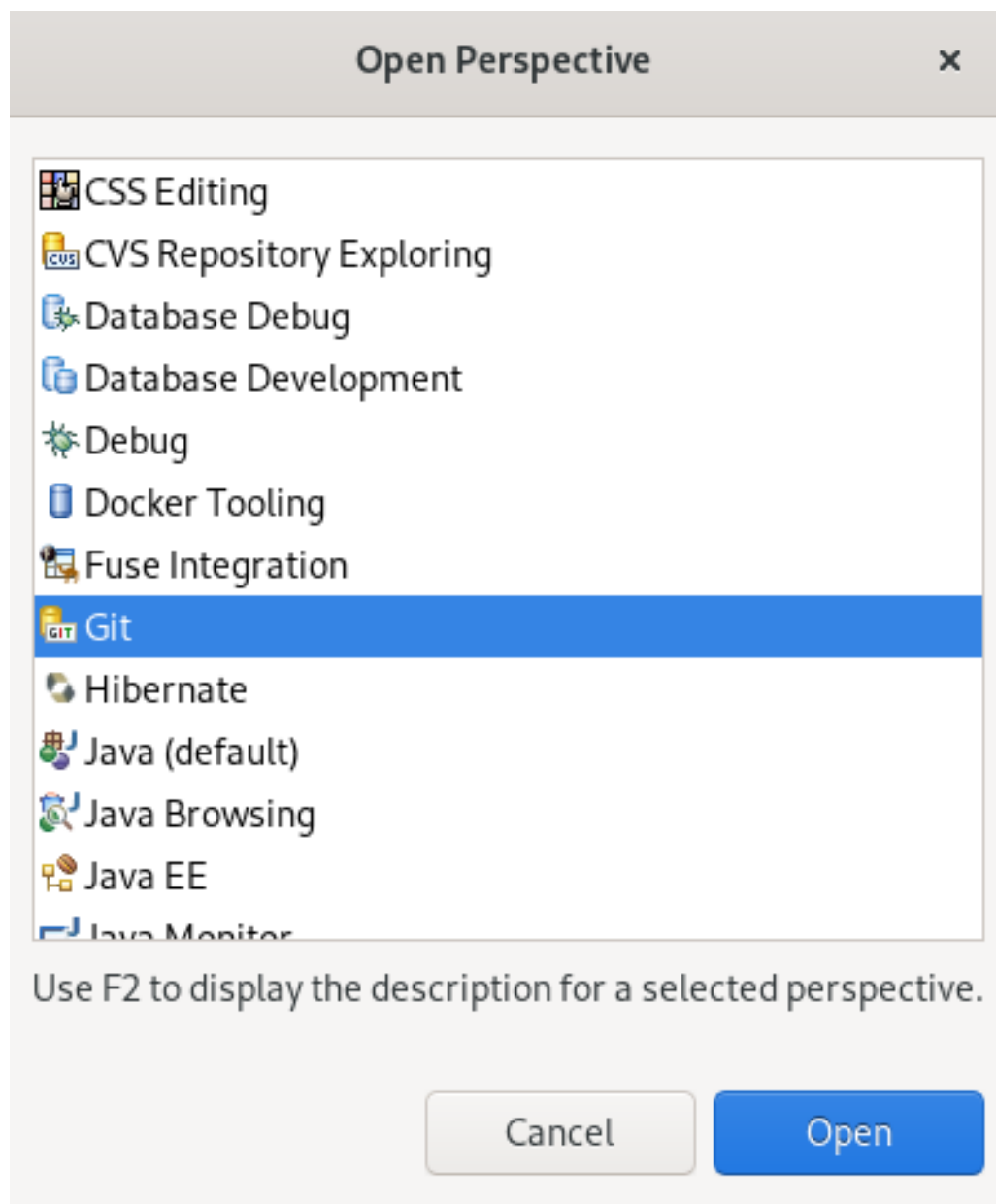
The following section describes how to open Git Perspective in CodeReady Studio.

### Procedure

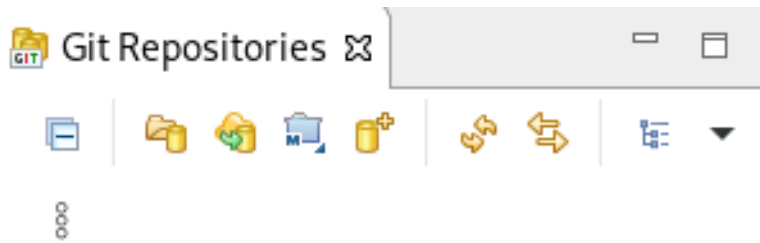
1. Start CodeReady Studio.
2. Click **Window** → **Perspective** → **Open Perspective** → **Other**.



The **Open Perspective** window appears.



3. Select **Git**.
4. Click **Open**.  
The **Git Repositories** view appears.



Select one of the following to add a repository to this view:

-  [Add an existing local Git repository](#)
-  [Clone a Git repository](#)
-  [Create a new local Git repository](#)

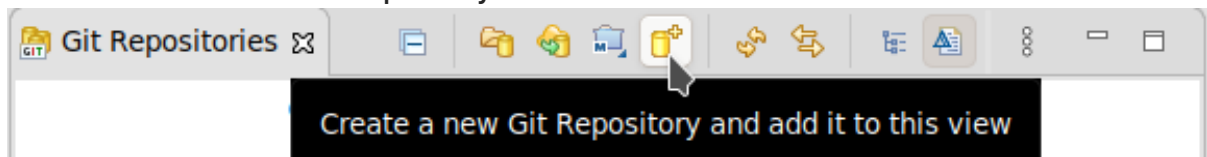
## 1.2. MANAGING REPOSITORIES IN GIT PERSPECTIVE

### 1.2.1. Creating a new Git repository

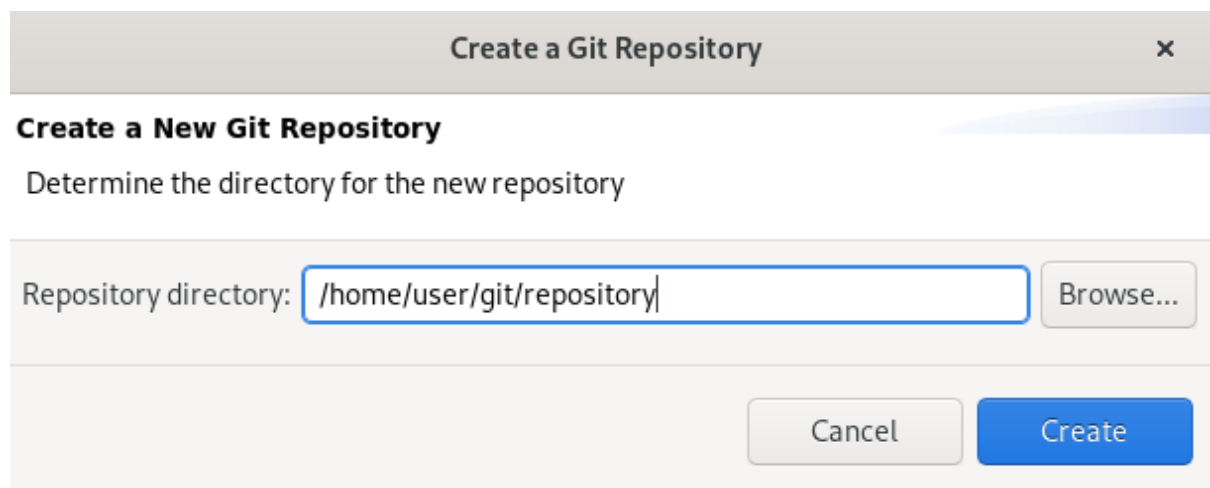
The following section describes how to use Git Perspective to create a new Git repository.

#### Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Click the **Create a new Git Repository and add it to this view** icon.

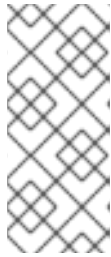


The **Create a Git Repository** window appears.



The path to the default **Repository directory** is generated automatically. Choose the path where you want your repository to be stored at and continue with the repository creation.

Optionally, you can select the **Create as bare repository** check box.



#### NOTE

Bare repositories are recommended for central repositories, not for development environments. They do not contain a working or checked out copy of any source file. This prevents editing files and committing changes. Additionally, they store the Git revision history for your repository in the **root** folder instead of a **.git** subfolder.

4. Click **Create**.

A new Git repository is created on your local machine and is now listed in the **Git Repositories** view.

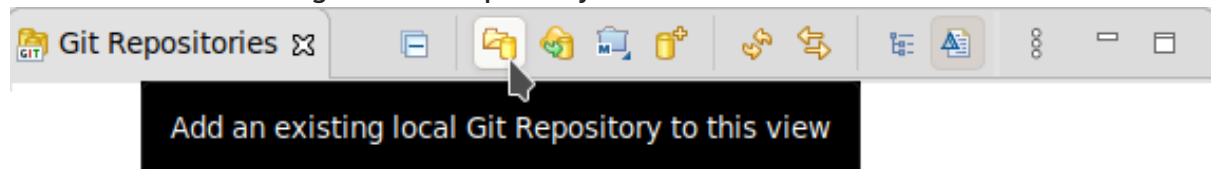


### 1.2.2. Adding an existing local Git repository

The following section describes how to use Git Perspective to add a local Git repository to CodeReady Studio.

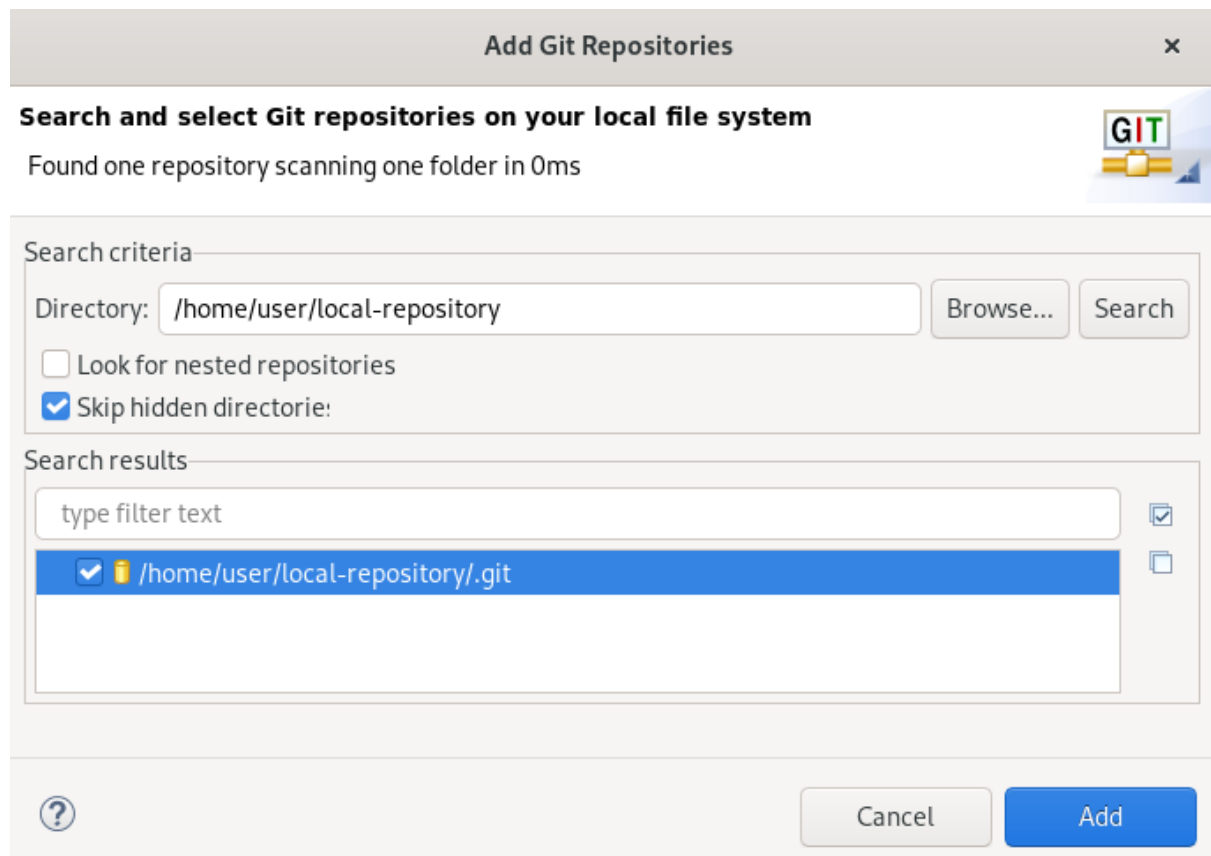
#### Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Click the **Add an existing local Git Repository to this view** icon.



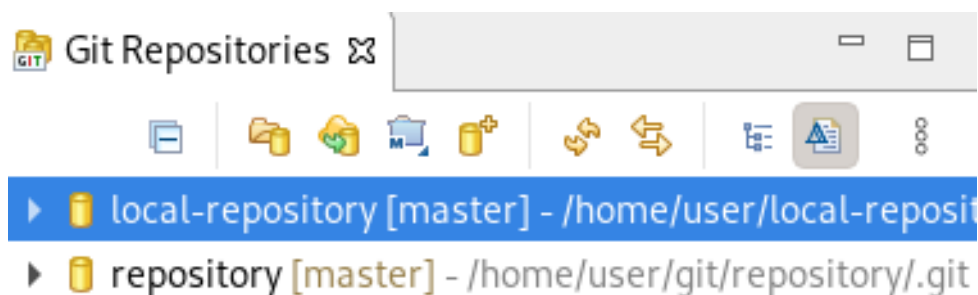
The **Add Git Repositories** window appears.

4. Click **Browse** to locate your local Git repository.



5. In the **Search results** field, select the checkbox displaying the path to the **.git** file.
6. Click **Add**.

Your local repository is now listed in the **Git Repositories** view.

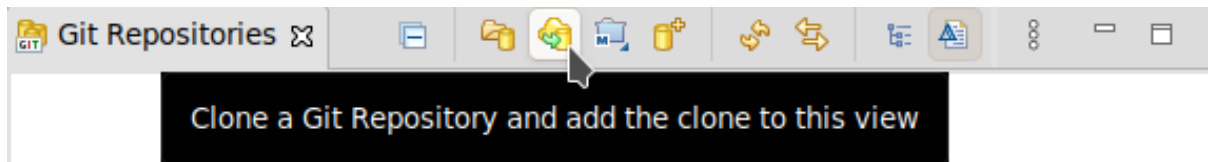


### 1.2.3. Cloning an existing Git repository

The following section describes how to use Git Perspective to create a local clone of a repository that already exists online (GitHub, GitLab).

#### Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Click the **Clone a Git Repository and add the clone to this view** icon.



The **Clone Git Repository** window appears.

Clone Git Repository ✕

**Source Git Repository**

Enter the location of the source repository.

**Location**

URI:  Local File...

Host:

Repository path:

**Connection**

Protocol: https ▾

Port:

**Authentication**

User:


Password:

Store in Secure Store

?
< Back
Next >
Cancel
Finish

4. Add the address for the source repository to the **URI** field.  
The **Host** and **Repository path** fields are populated automatically.
5. Click **Next**.
6. Select the branches you want to clone.
7. Click **Next**.
8. Ensure that the **Directory** path and **Initial branch** are set correctly.

Clone Git Repository ✕

**Local Destination** 

Configure the local storage location for cloned-repository.

Destination

Directory:  Browse

Initial branch: master ▼

Clone submodules

Configuration

Remote name:

Projects

Import all existing Eclipse projects after clone finishes

Working sets

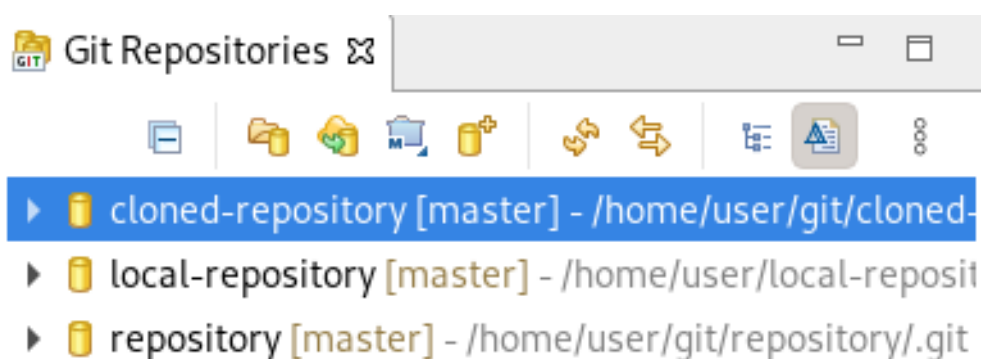
Add project to working sets New...

Working sets: ▼ Select...

? < Back
Next >
Cancel
Finish

9. Click **Finish**.

Your cloned repository is now listed in the **Git Repositories** view of CodeReady Studio.



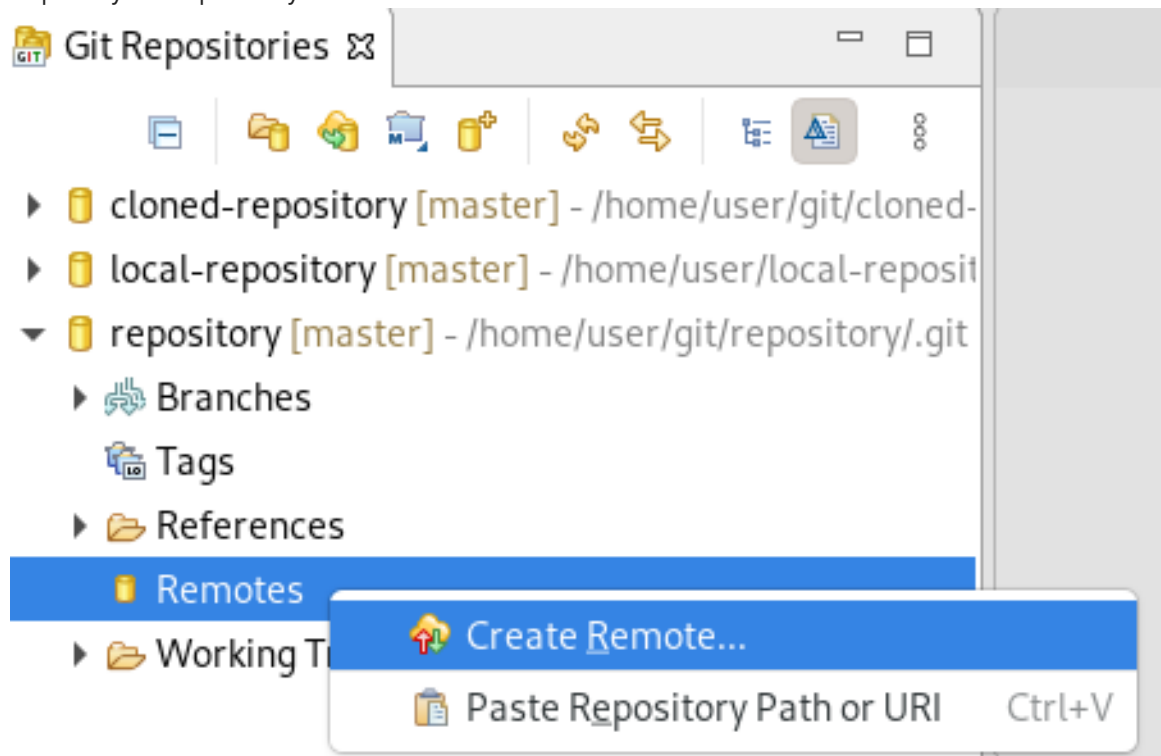
#### 1.2.4. Adding a remote for the repository

After setting up your repository in Git Perspective for the first time, add a remote for the repository. This is a one-time set up step for newly created or added repositories.

The following section describes how to use Git Perspective to set up the remote for your repository.

### Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Expand your repository.



4. Right-click **Remotes** → **Create Remote**.  
The **New Remote** window appears.



**New Remote** x

**Please enter a name for the new remote**

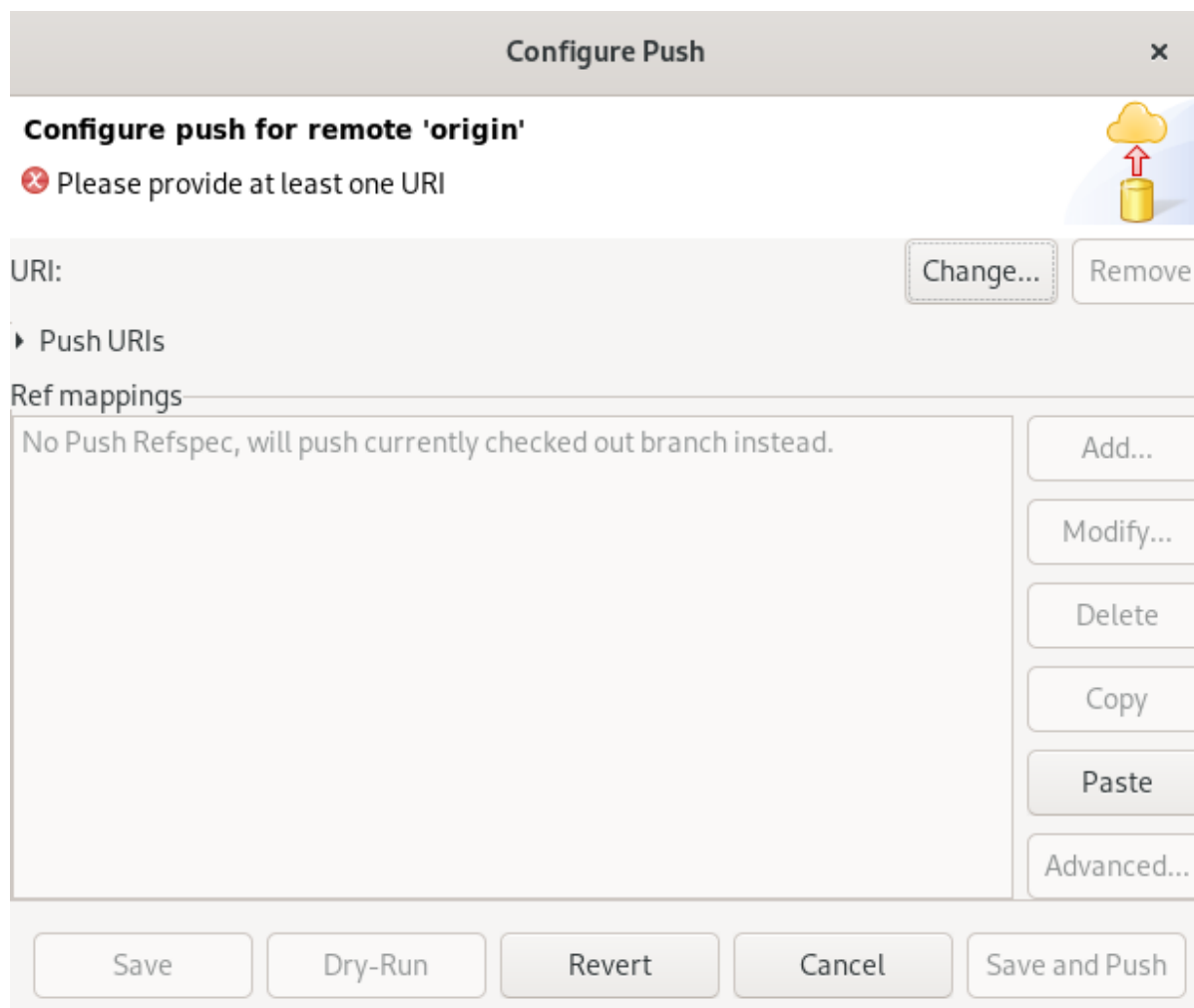
You need to configure the new remote for either fetch or push; you can add configuration for the other direction later

Remote name:

Configure push  
 Configure fetch

? Cancel Create

5. Name your remote.
6. Ensure that **Configure push** is selected.
7. Click **Create**.  
The **Configure Push** window appears.



8. Click **Change**.  
The **Select a URI** window appears.

Select a URI

**Source Git Repository**

Enter the location of the source repository.

Location

URI:  Local File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

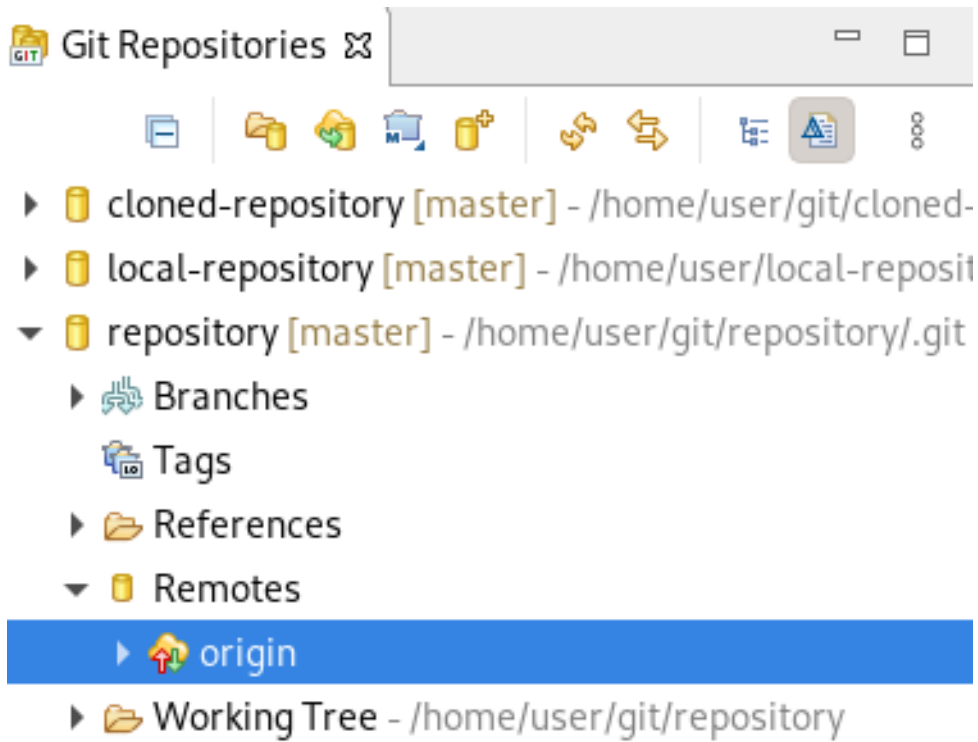
Store in Secure Store

?

Cancel Finish

9. Add the URI, username and password for the source repository.  
The **Host** and **Repository** path fields are populated automatically.
10. Click **Finish**.
11. Click **Save**.

Your newly added remote is now listed in the **Git Repositories** view in CodeReady Studio.



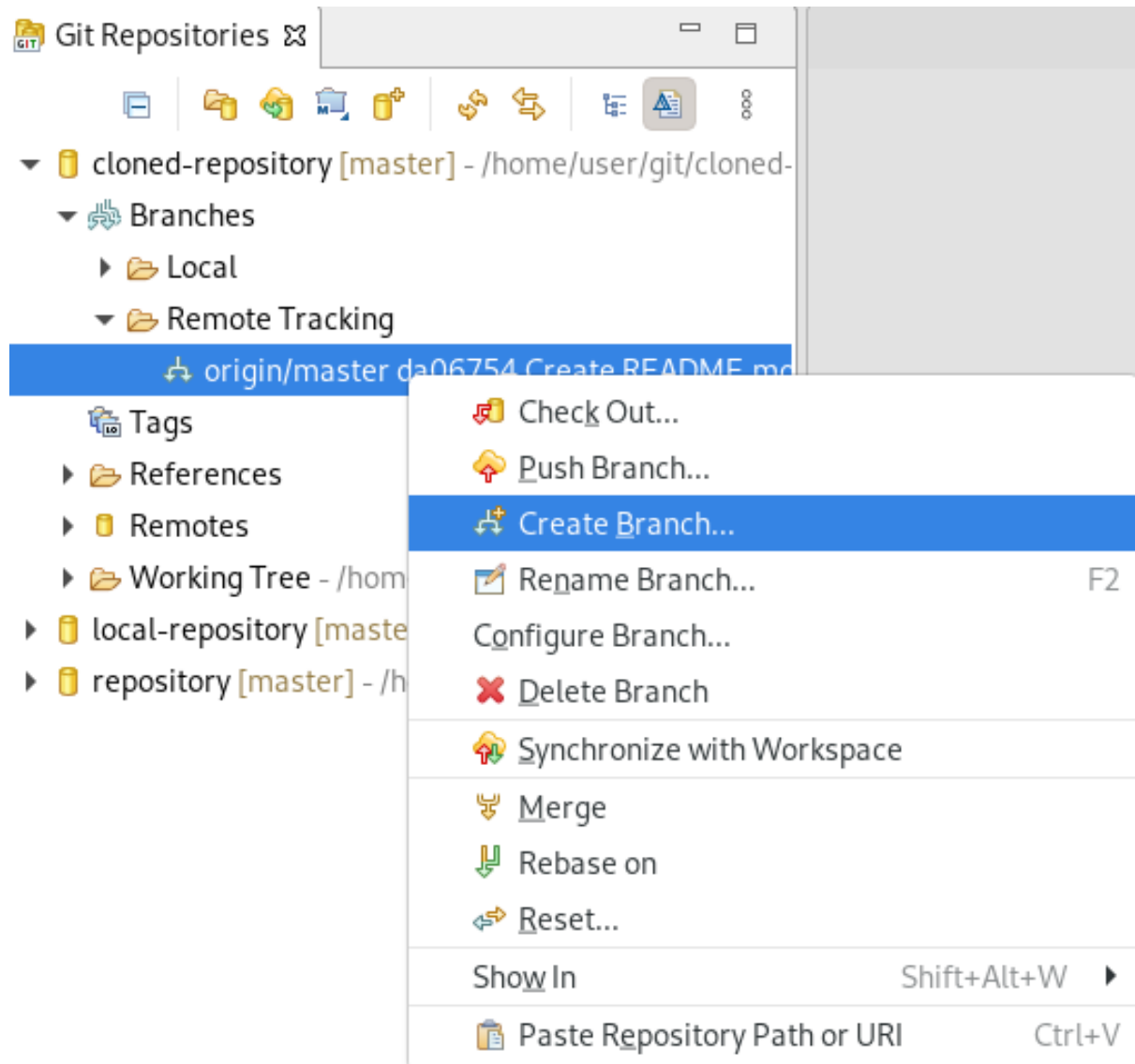
## 1.3. MANAGING BRANCHES IN GIT PERSPECTIVE

### 1.3.1. Creating a new branch

The following section describes how to use Git Perspective to create a new branch.

#### Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Expand your repository.
4. Under **branches** → **Remote Tracking**, right-click **master** → **Create Branch**.



The **Create Branch** window appears.

**Create Branch** ×

**Create a new branch in repository cloned-repository**

**i** Local branch as upstream is not recommended, use remote branch

Source: master Select...

Branch name:

Configure upstream for push and pull

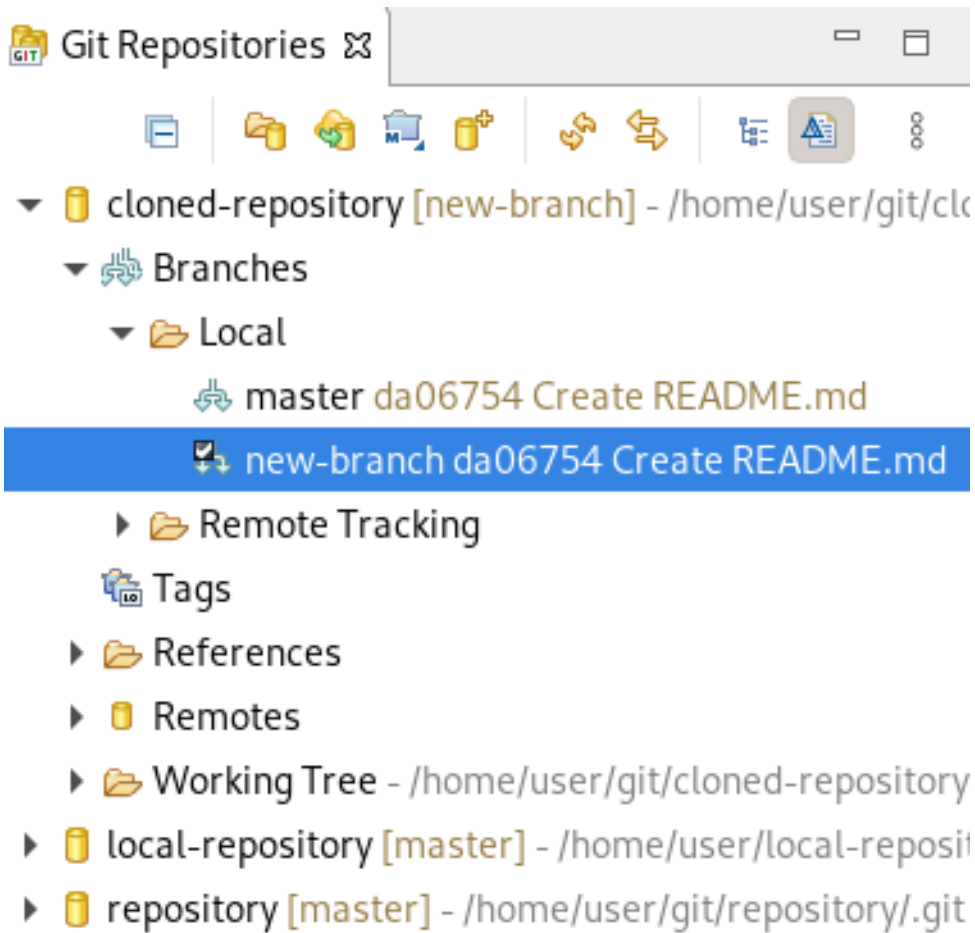
When pulling:  ▼

Check out new branch

**?** Cancel Finish

5. Click **Select** to pick the source of the new branch.
6. Name your branch.
7. Select the **Configure upstream for push and pull** and **Checkout new branch** check boxes.
8. Select an option in the **When pulling** field.
9. Click **Finish**.

Your newly added branch is now listed in the **Git Repositories** view under **branches** → **Local** in CodeReady Studio.

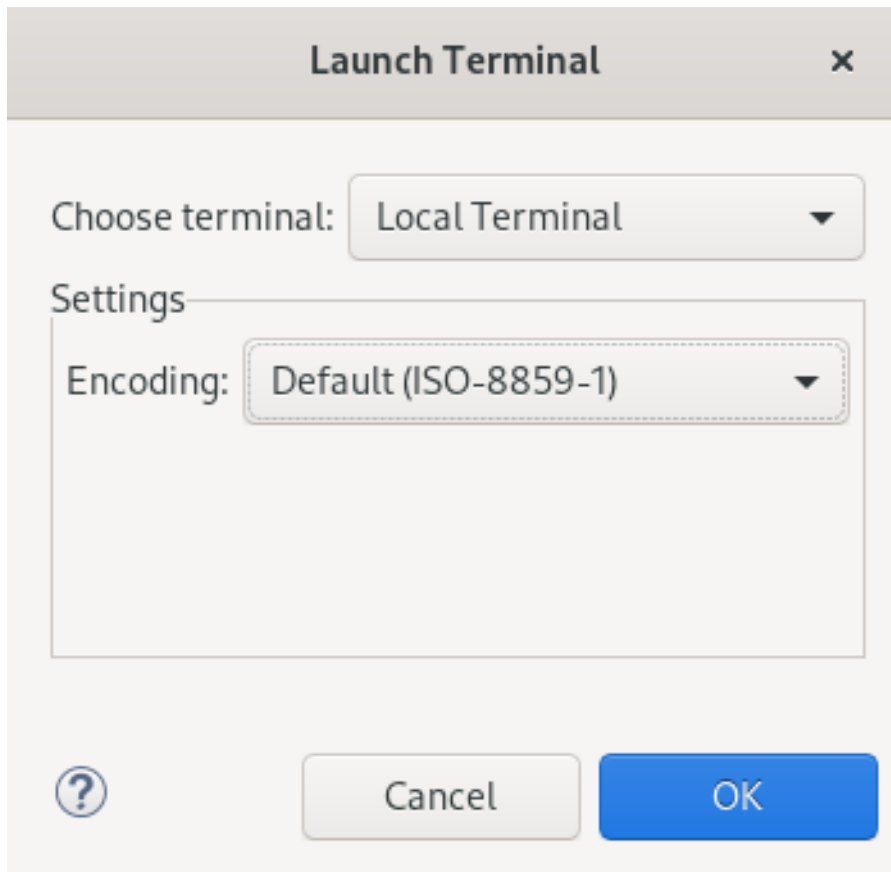


### 1.3.2. Working in the branch

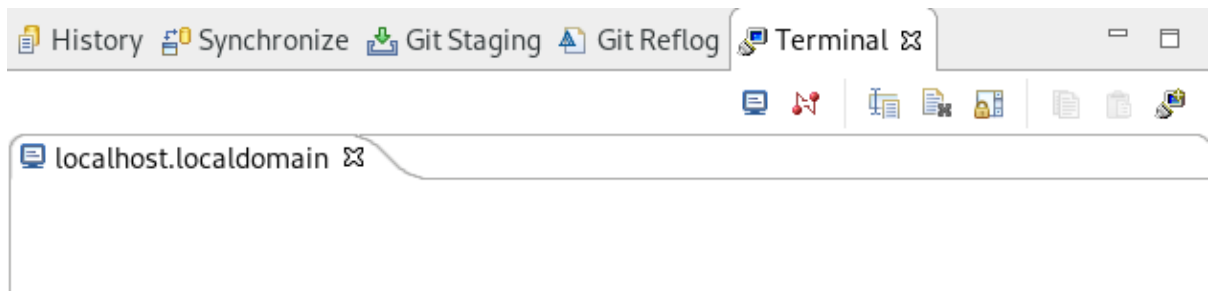
The following section describes how to open a built-in terminal in Git Perspective so you can work on the created branch.

#### Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Press **Shift+Ctrl+Alt+T**.  
The **Launch Terminal** window appears.



4. Choose **Local Terminal**.
5. Set **Encoding** to **Default (ISO-8859-1)**.
6. Click **OK**.  
The **Terminal** window now displays the command-line terminal.



Note that by default the current working directory is the home directory of your current user.

### 1.3.3. Updating your local repository

To avoid merge conflicts, update your local repository before merging your changes, especially when working in a shared repository.

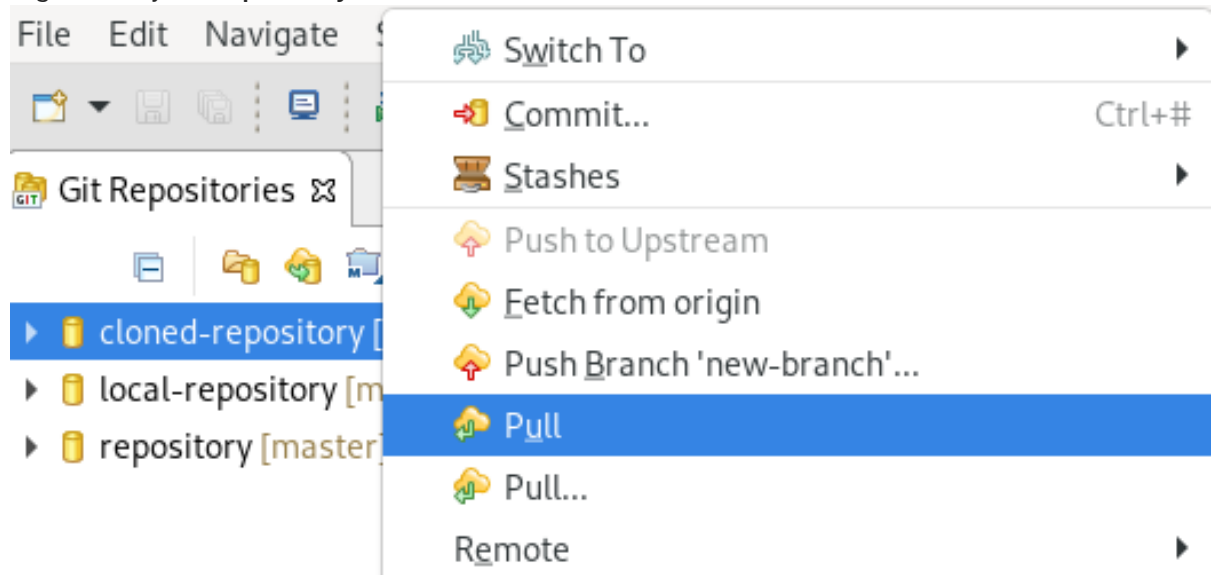
To update your local repository, pull changes from the remote repository and merge them into your local repository.

#### Procedure

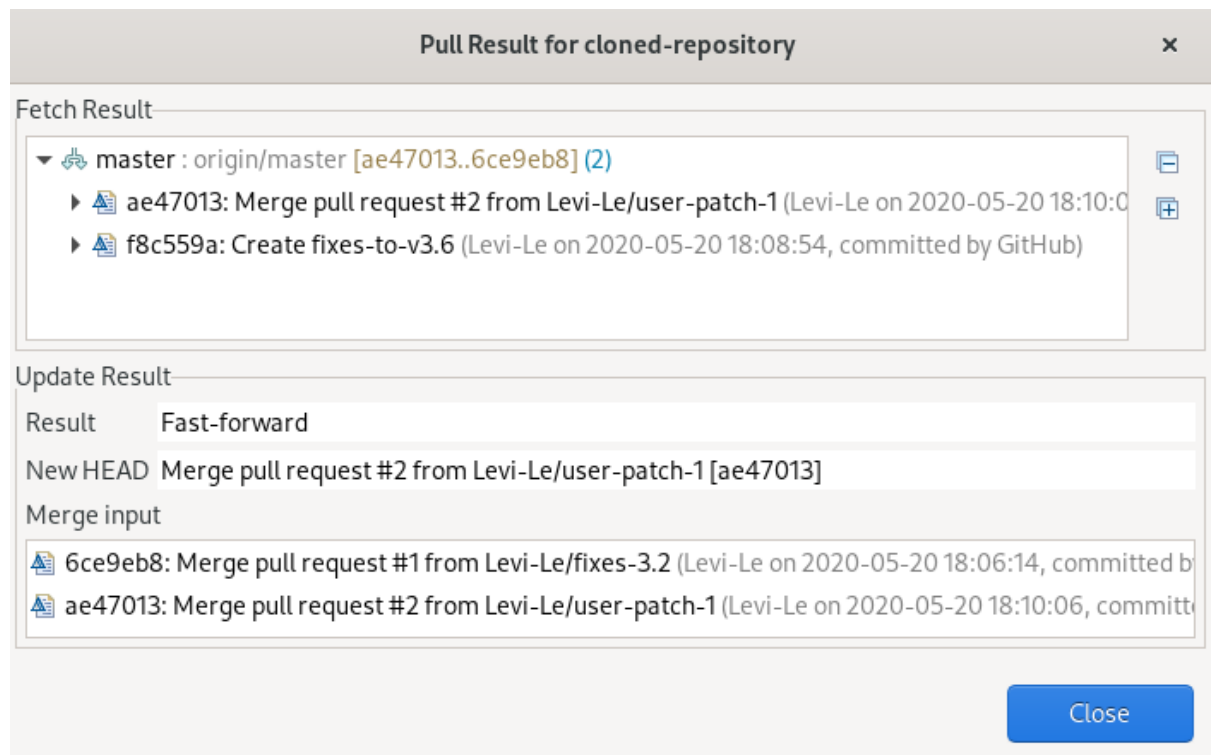
1. Start CodeReady Studio.
2. Open **Git Perspective**.



3. Right-click your **repository** → **Pull**.



The **Pull Results** window appears.



4. Click **Close**.

Now the changes from the remote repository are merged into your local repository.

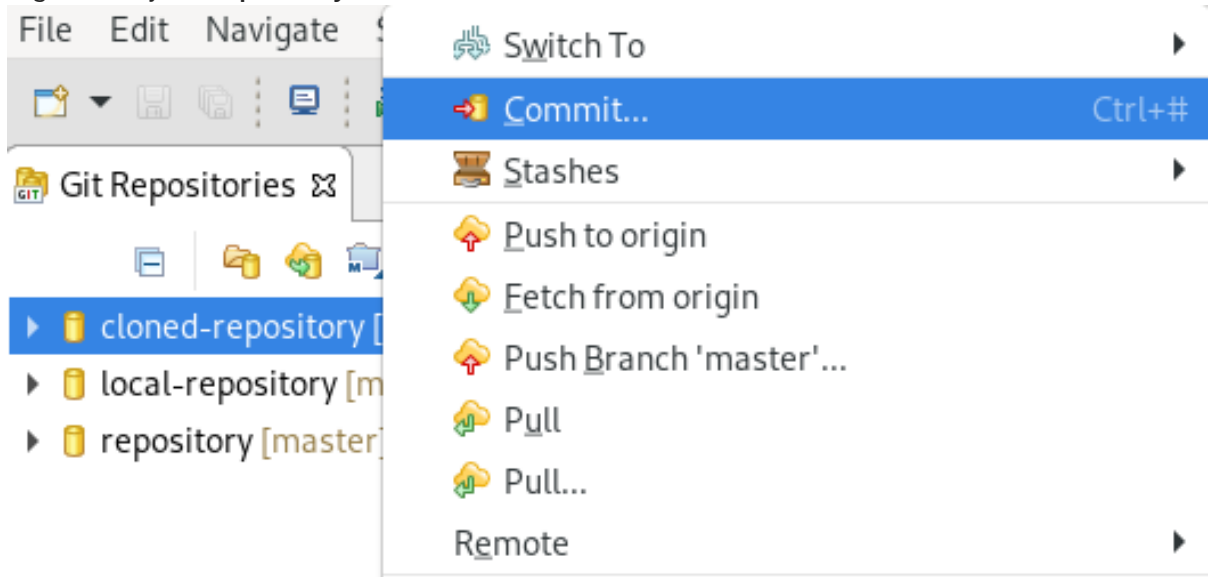
## 1.4. COMMITTING AND PUSHING CHANGES

The following section describes how to commit and push changes in CodeReady Studio.

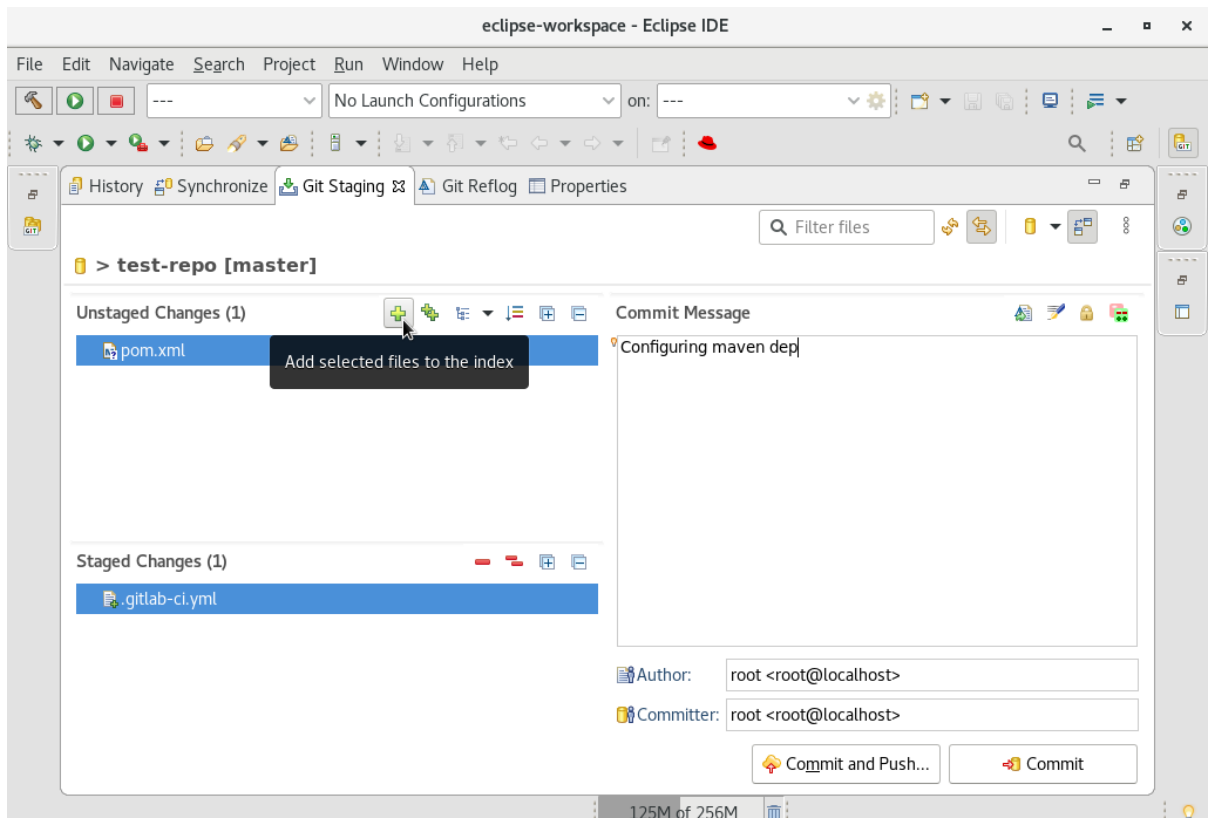
### Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.

3. Right-click your repository → Commit.



The **Git Staging** view appears.



4. Select the changes you want to stage.
5. Click the **Add selected files to the index** icon to stage the changes.
6. Add a commit message to the **Commit Message** field.  
**Author** and **Committer** fields are populated automatically.
7. Click **Commit** to commit your changes, or **Commit and Push** to commit your changes and push them to the remote repository.

Note that when selecting the **Commit and Push** option you are prompted to enter the repository address, your access username, and password for the repository.

## CHAPTER 2. MAVEN BASICS IN CODEREADY STUDIO

Maven provides a standardized build system for application development, and facilitates fetching dependencies from one or more repositories.

Root Maven projects can serve as aggregators for multiple Maven modules (sub-projects). For each module that is part of a maven project, a `<module>` entry is added to the project's **pom.xml** file. A **pom.xml** contains `<module>` entries and is often referred to as an **aggregator pom**.

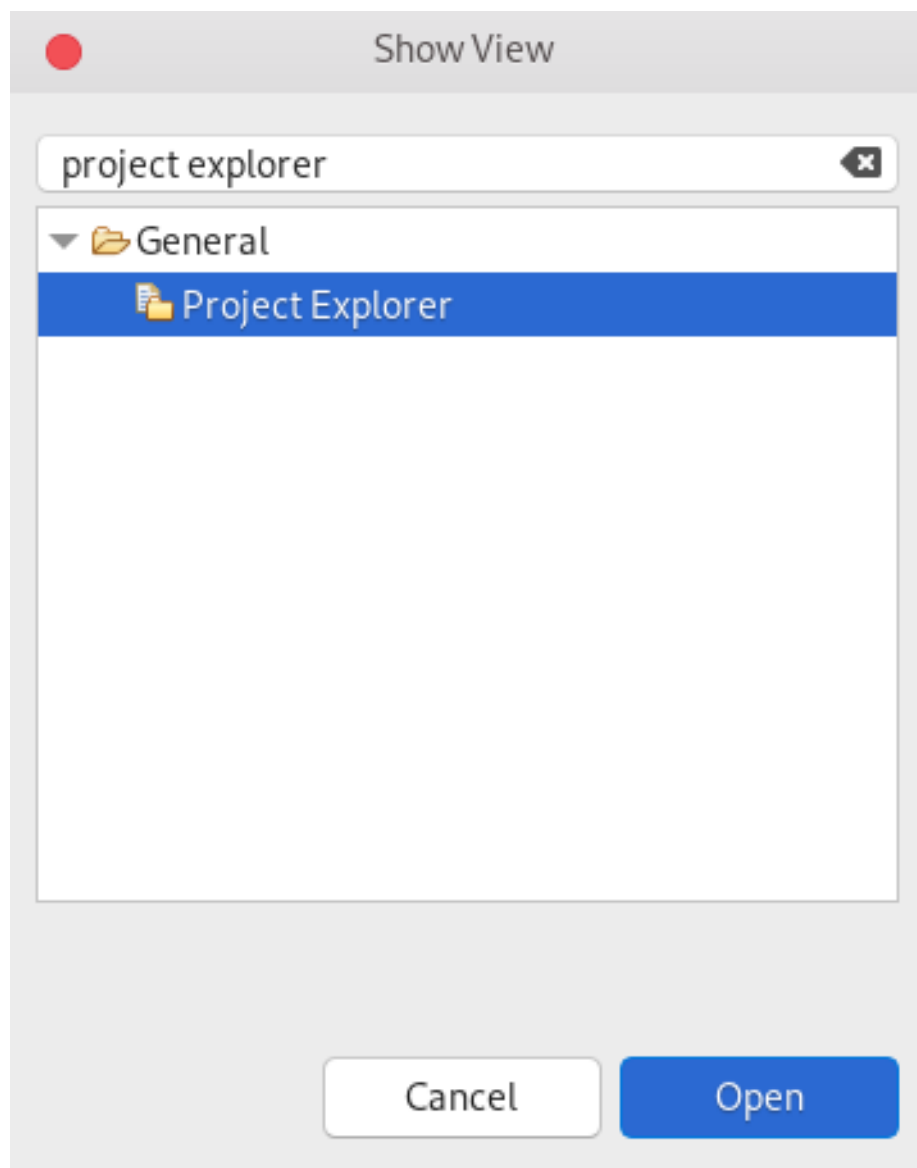
When modules are included into a project it is possible to execute Maven goals across all modules by a single command issued from the parent project directory.

### 2.1. CREATING A NEW MAVEN PROJECT

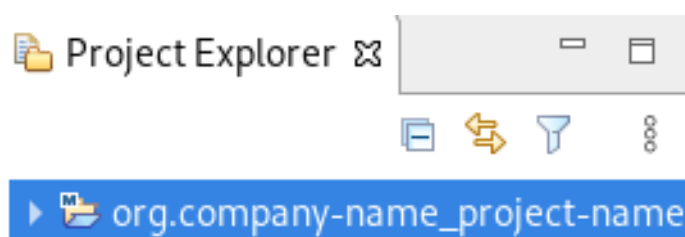
The following section describes how to create a new Maven project in CodeReady Studio.

#### Procedure

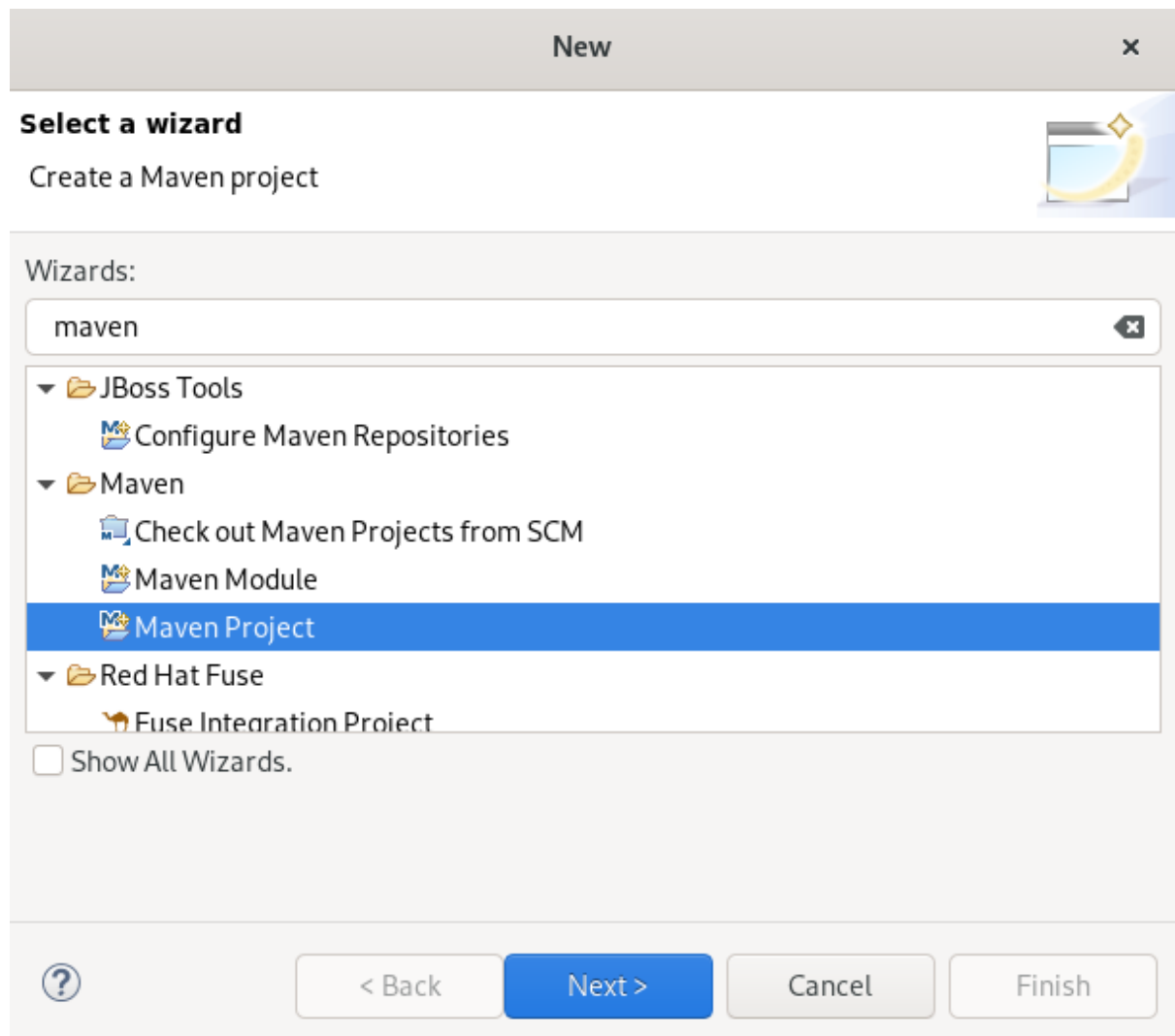
1. Start CodeReady Studio.
2. Click **Window** → **Show View** → **Other**.  
The **Show View** window appears.



3. Enter Project Explorer in the search field.
4. Select Project Explorer.
5. Click **Open**.  
The Project Explorer view appears.



6. Press **Ctrl+N**.  
The **Select a wizard** window appears.



7. Enter **Maven** in the **Wizards** field.
8. Select **Maven Project**.
9. Click **Next**.  
The **New Maven Project** window appears.

10. Select the **Create a simple project** check box.



#### NOTE

By selecting the **Create a simple project** check box you are skipping the archetype selection and the project type is automatically set to Project Object Model (POM), which is a requirement for multi-module Maven projects.

To create a standalone Maven project instead, clear the **Create a simple project** check box and follow the onscreen instructions to set the packaging option to **jar** or **war**.

11. Click **Browse** to select the workspace location.
12. Click **Next**.

13. Enter the group ID and the artifact ID.



#### NOTE

The values for the IDs cannot include spaces or special characters. The only special characters allowed are periods (.), underscores (\_), and dashes (-). An example of a typical group ID or artifact ID is **org.company-name\_project-name**.

Optionally, you can name your project and add a description.

14. Set **Packaging** to **pom**, **jar** or **war**.
15. Click **Finish**.

Your newly created Maven project is now listed in the Project Explorer view.

## 2.2. IMPORTING EXISTING MAVEN PROJECTS

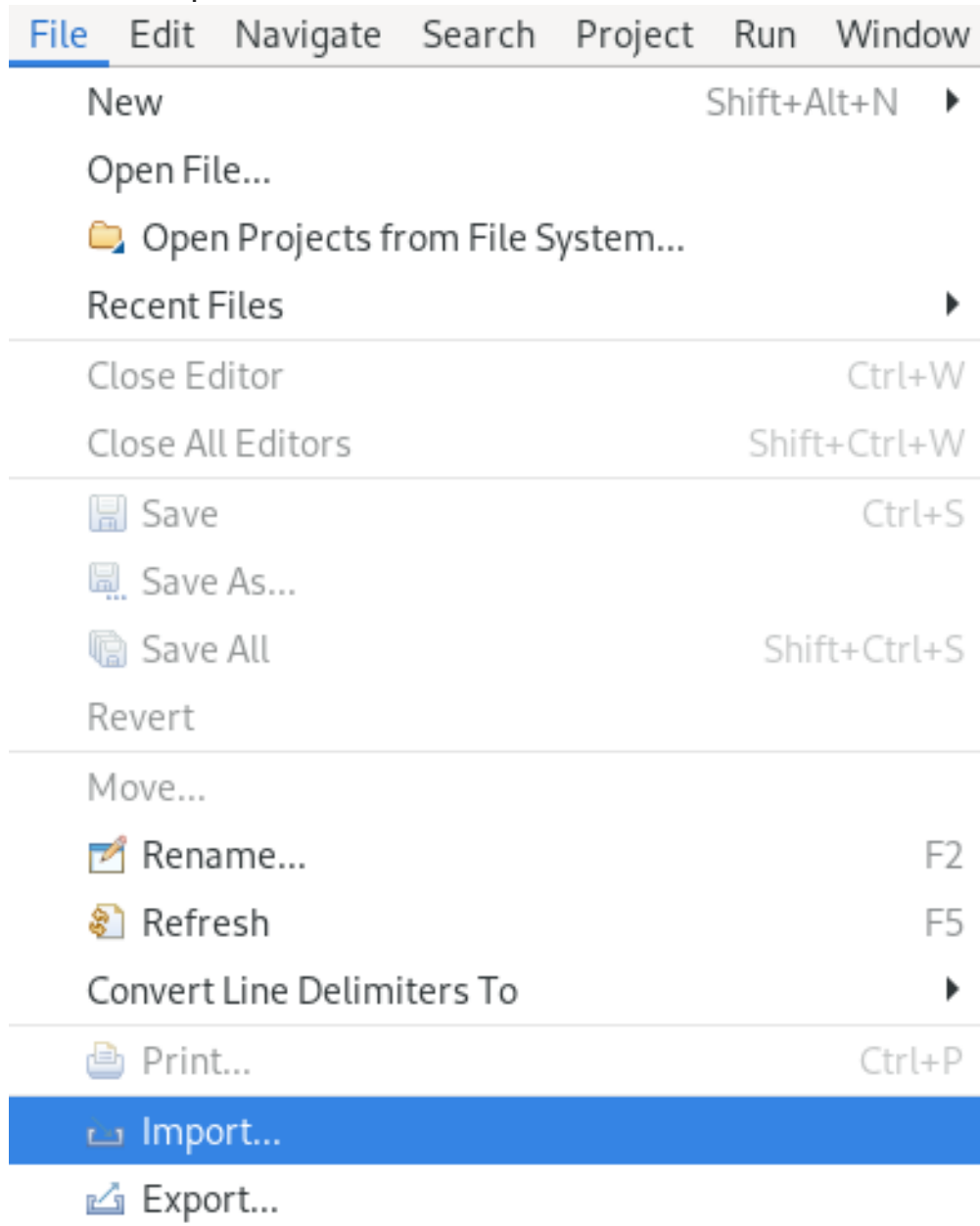
The following section describes how to import existing Maven projects into CodeReady Studio.

## 2.2.1. Importing an existing locally stored Maven project

The following section describes how to import an existing locally stored Maven project into CodeReady Studio.

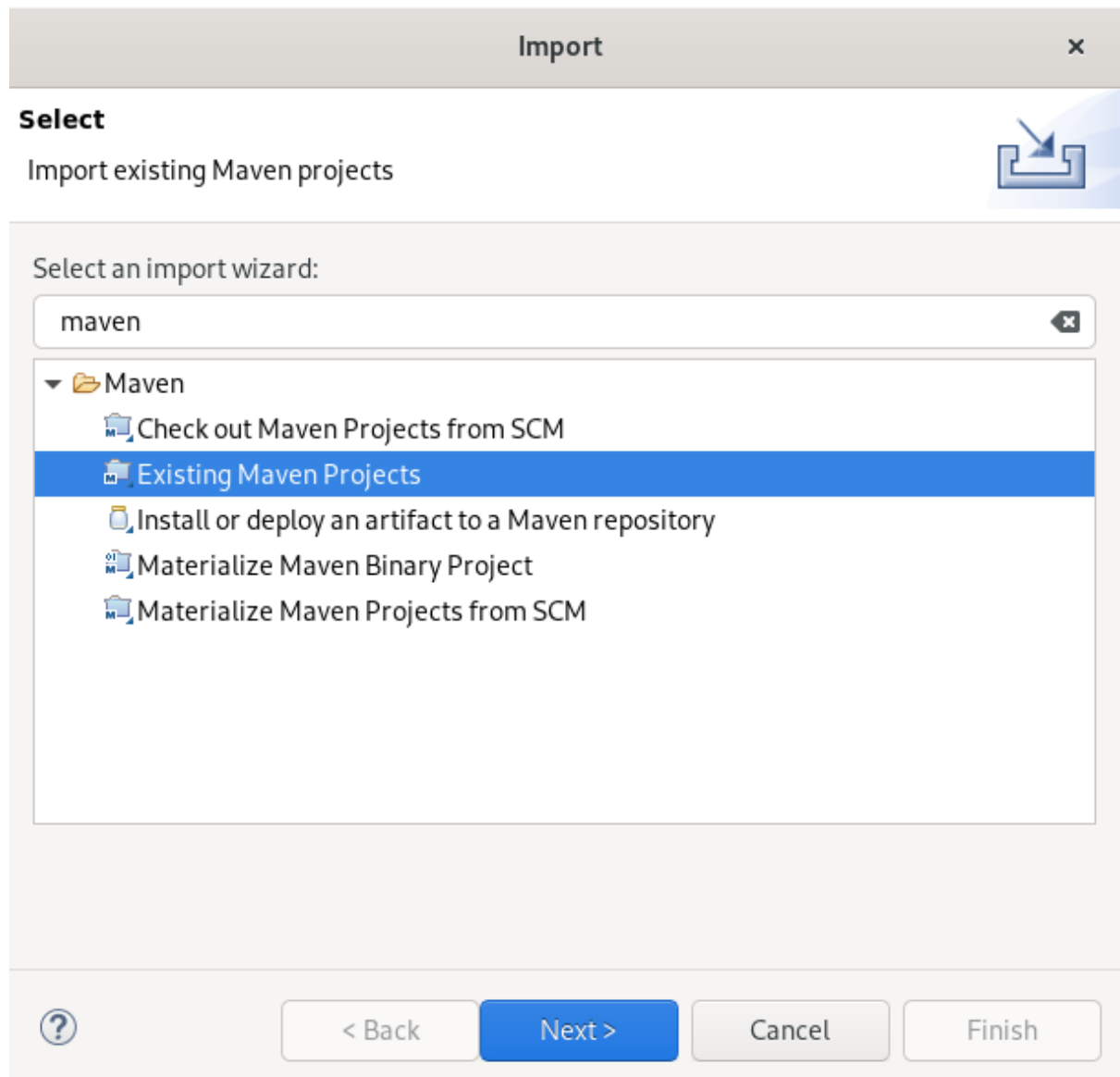
### Procedure

1. Start CodeReady Studio.
2. Click **File** → **Import**.

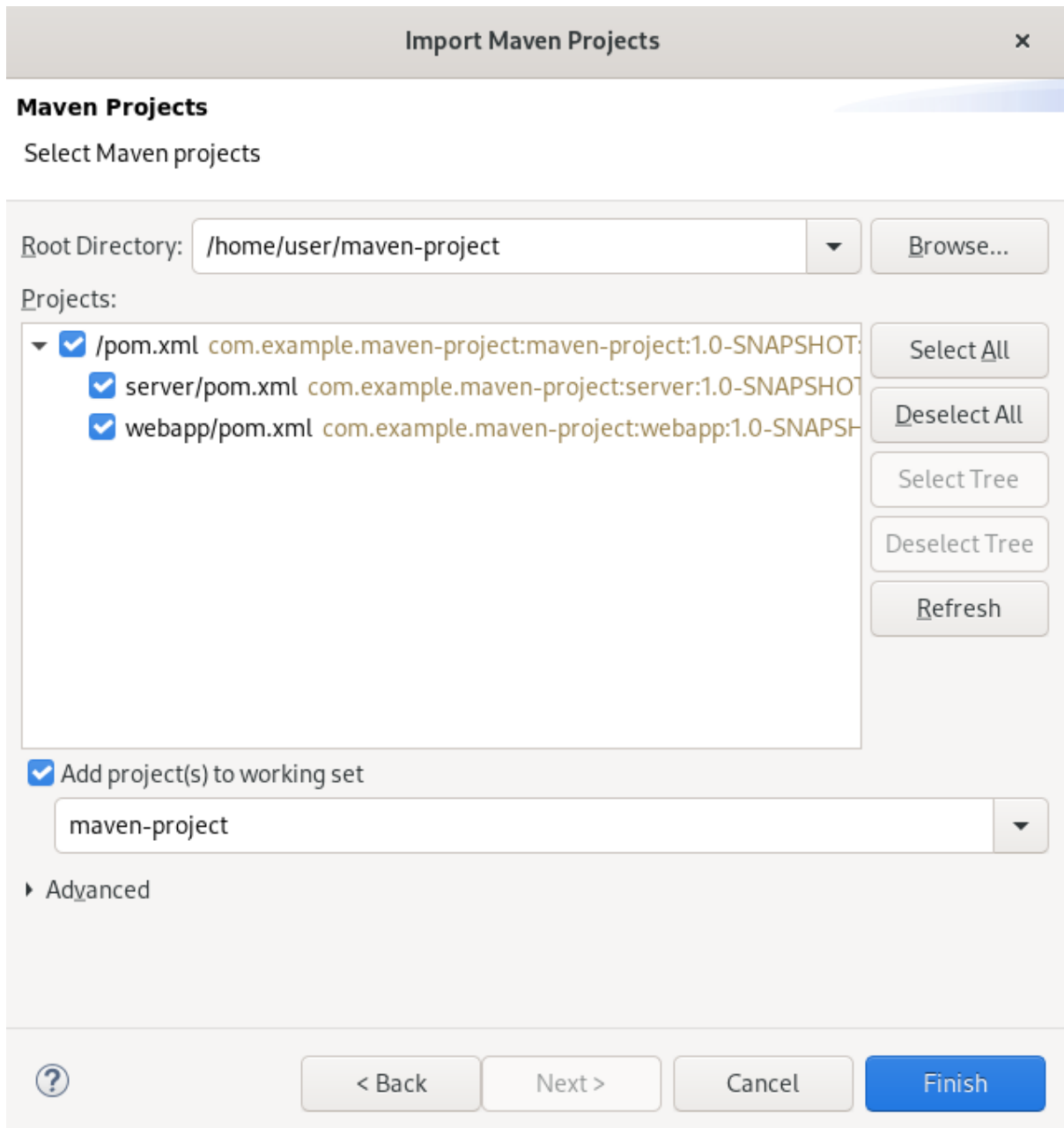


The **Import** window appears.





3. Enter **Maven** in the **Select an import wizard** field.
4. Select **Existing Maven Projects**
5. Click **Next**.  
The **Import Maven Project** window appears.



6. Click **Browse** to locate your Maven project.
7. Select the **Add project(s) to working set** check box.
8. Click **Finish**.

Your local Maven project is now listed in the Project Explorer view.

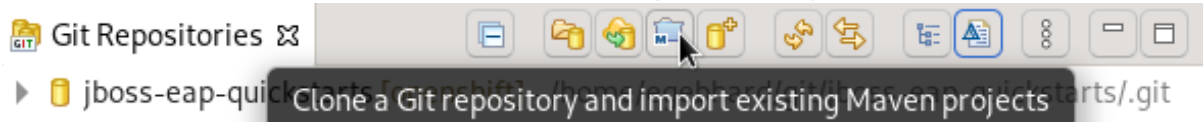
### 2.2.2. Importing an existing remotely stored Maven project

The following section describes how to import an existing remotely stored Maven project into CodeReady Studio.

#### Procedure

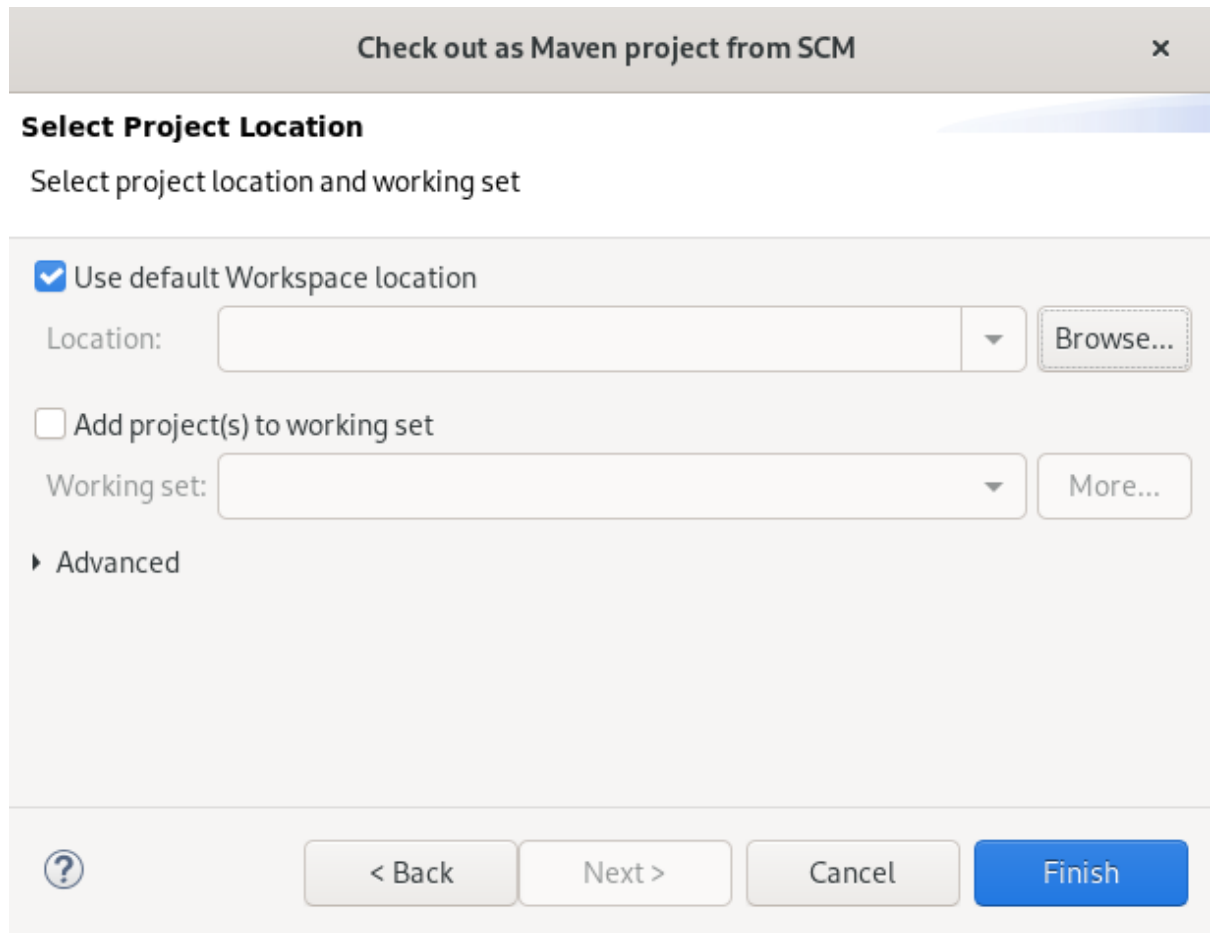
1. Start CodeReady Studio.
2. Open Git Perspective.

3. Click the **Clone a Git repository and import existing Maven projects** icon.



The **Check out as Maven project from SCM** window appears.

4. Add the address for the source repository to the **SCM URL** field.
5. Click **Next**.  
The **Select Project Location** window appears.



6. Click **Browse** to select the workspace location.
7. Click **Finish**.

Your remote Maven project is now listed in the **Git Repositories** view.

## 2.3. CREATING A NEW MAVEN MODULE

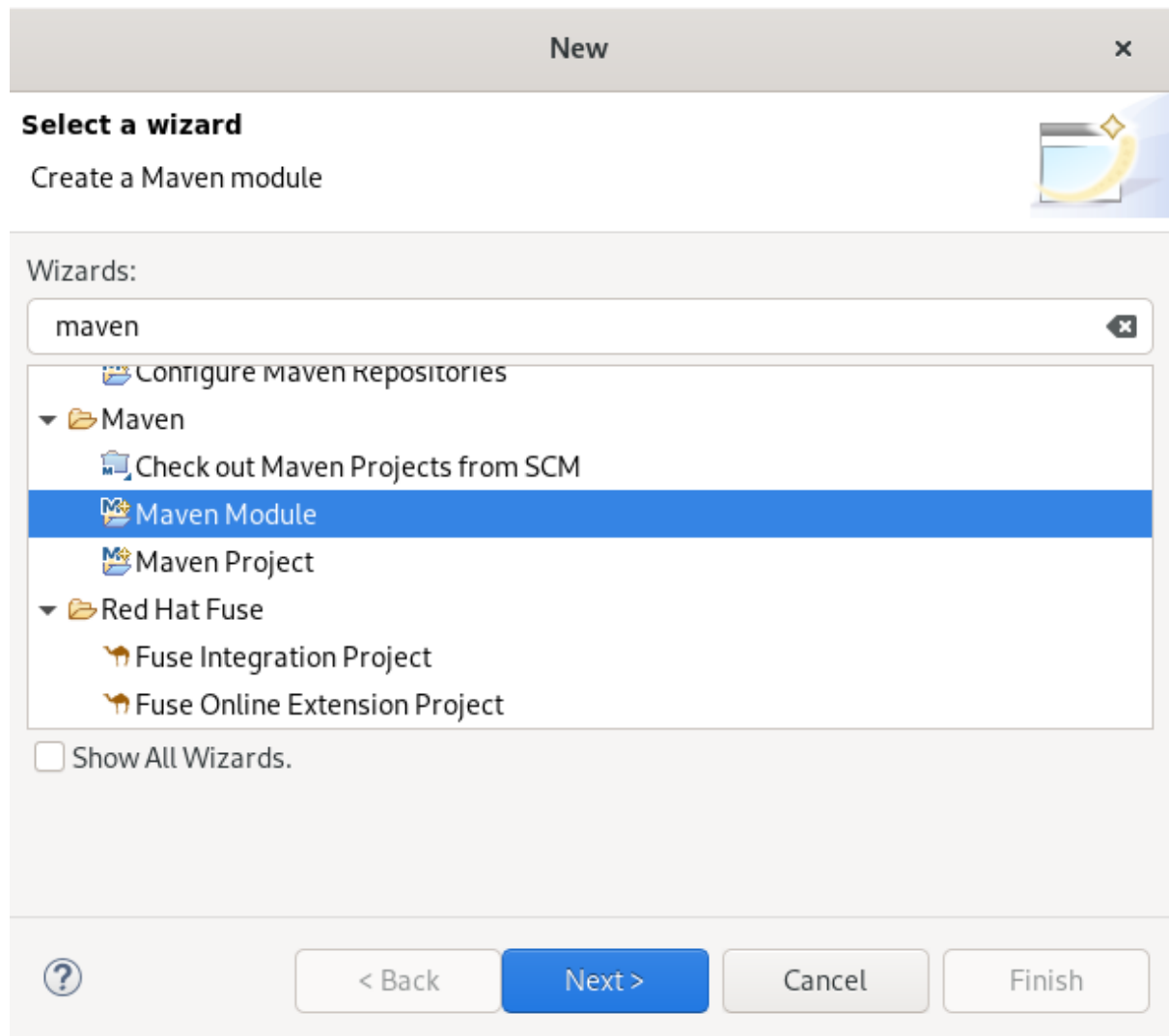
The following section describes how to create a new Maven module.

### Prerequisites

- An existing Maven project.  
For more information on how to create a Maven project, see [Section 2.1, "Creating a new Maven project"](#).

### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



3. Enter **Maven** in the **Wizards** field.
4. Select **Maven Module**.
5. Click **Next**.  
The **New Maven Module** window appears.

6. Select the **Create a simple project** check box.




#### NOTE

By selecting the **Create a simple project** check box you are skipping the archetype selection and the project type is automatically set to Project Object Model (POM), which is a requirement for multi-module Maven projects.

To create a standalone Maven project instead, clear the **Create a simple project** check box and follow the onscreen instructions to set the packaging option to **jar** or **war**.

7. Name your module.
8. Click **Browse** to select the parent project.
9. Click **Next**.  
The **Configure Project** window appears.

New Maven Module ✕

**New Maven Module** 

Configure project

**Artifact**

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

**Parent Project**

Group Id:

Artifact Id:

Version:

▶ **Advanced**

?
< Back
Next >
Cancel
Finish

10. Set **Packaging** to **pom**, **jar** or **war**.  
Optionally, you can name your module and add a description.
11. Click **Finish**.

Your newly created Maven module is now listed below your Maven project.

## 2.4. ADDING A MAVEN DEPENDENCY TO A MAVEN PROJECT

The following section describes how to add a Maven dependency to a Maven project in CodeReady Studio.

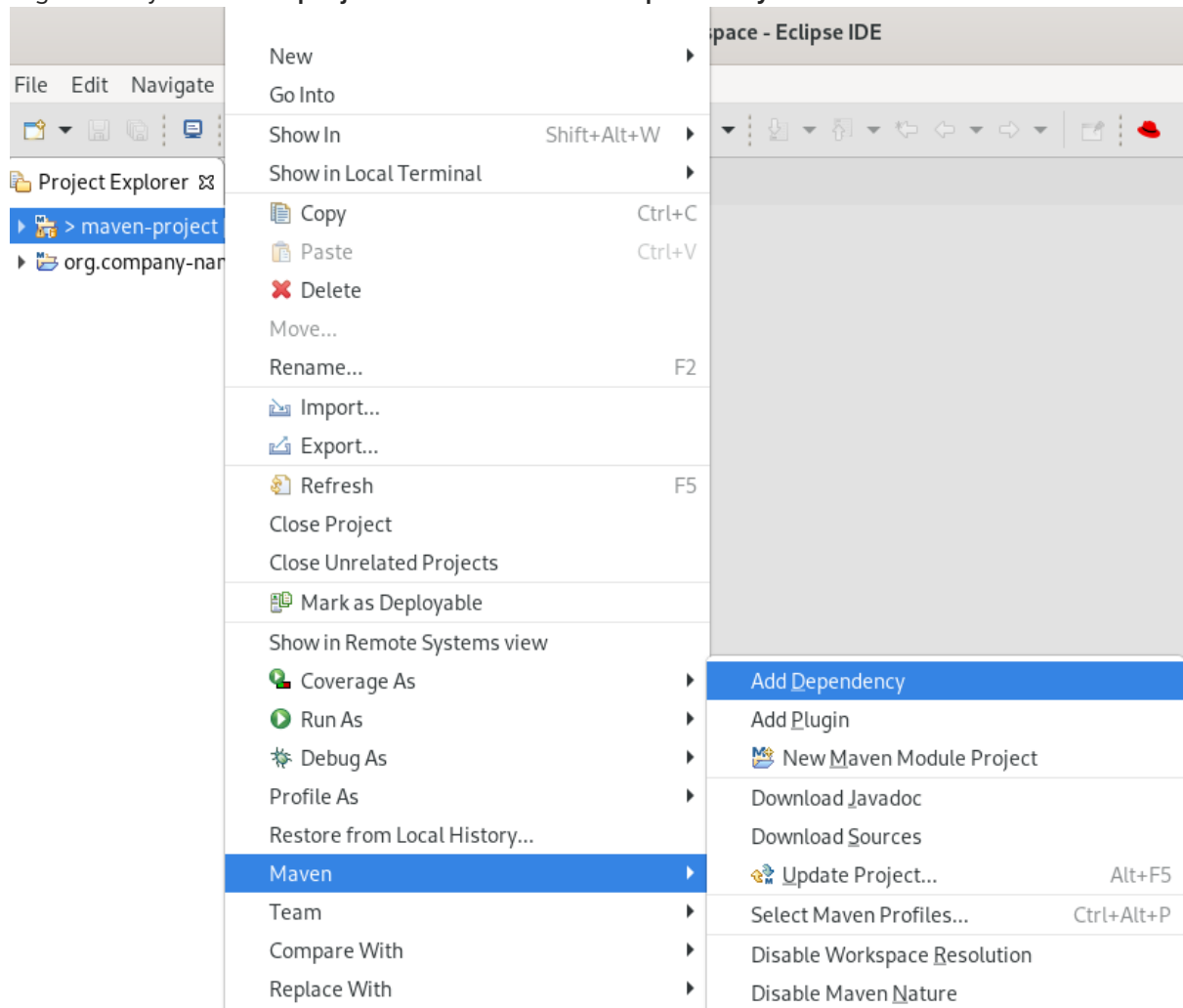
### Prerequisites

- An existing Maven project.  
For more information on how to create a Maven project, see [Section 2.1, "Creating a new Maven project"](#).

### Procedure

1. Start CodeReady Studio.

2. Open Project Explorer.
3. Right-click your Maven project → Maven → Add Dependency.



The **Add Dependency** window appears.



**Add Dependency** ✕

Group Id: \*

Artifact Id: \*

Version:  Scope:  ▼

---

Enter groupId, artifactId or sha1 prefix or pattern (\*):

⚠ Index downloads are disabled, search results may be incomplete.

Search Results:

▶ 📦 org.company-name\_project-name org.company-name\_project

Cancel
OK

4. Enter the group ID or the artifact ID in the **Enter groupId, artifactId or sha1 prefix or pattern** field.  
The fields above are populated automatically.
5. Click **OK**.

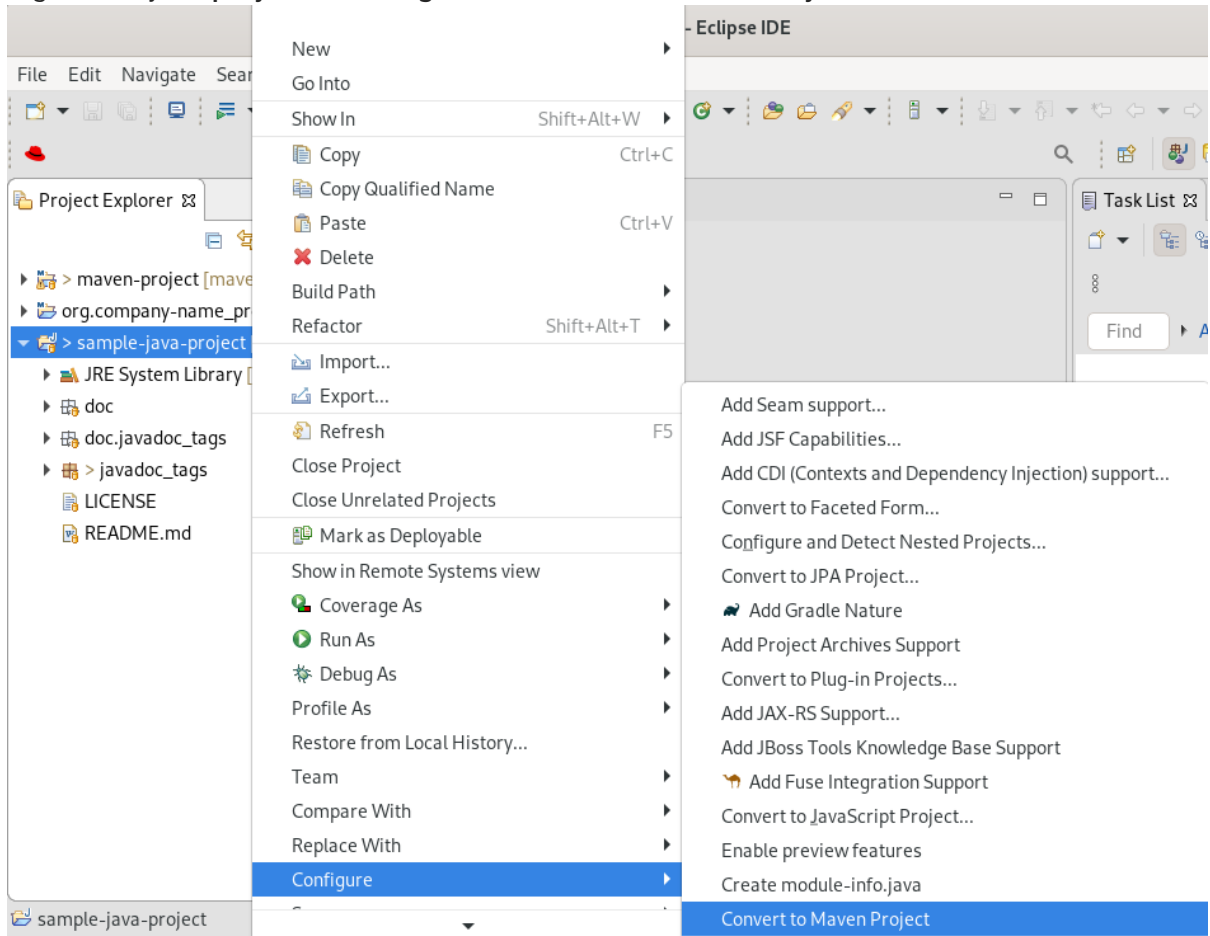
The dependency is now added to the **pom.xml** file of your project.

## 2.5. ADDING MAVEN SUPPORT TO AN EXISTING NON-MAVEN PROJECT

The following section describes how to add Maven support to an application created without Maven support.

1. Start CodeReady Studio.

- Open Project Explorer.
- Right-click your project → **Configure** → **Convert to Maven Project**



The **Create a new POM** window appears.

Create new POM ✕

**Maven POM**

This wizard creates a new POM (pom.xml) descriptor for Maven.

Project:

Artifact

Group Id:  ▼

Artifact Id:  ▼

Version:  ▼

Packaging:  ▼

Name:  ▼

Description:

?
Cancel
Finish

All fields are populated automatically. If you want to change the group ID or the artifact ID, note that the values cannot include spaces or special characters. The only special characters allowed are periods (.), underscores (\_), and dashes (-). An example of a typical group ID or artifact ID is `org.company-name_project-name`.

4. Click **Finish**.

Your newly generated **pom.xml** file appears under your Java project.

## 2.6. ADDITIONAL RESOURCES

- For more information on how to use the Maven software project management and comprehension tool, see the [JBoss Community Archive](#).

## CHAPTER 3. APPLICATION DEPLOYMENT IN CODEREADY STUDIO

In order to deploy applications to a server from within CodeReady Studio you must configure the IDE with information about the server. For a local server this information includes the following:

- A server runtime environment with details about the server location, runtime JRE, and configuration files
- A server adapter with management settings for the server runtime environment, including access parameters, launch arguments, and publishing options

JBoss Server Tools enables you to efficiently configure a local server ready for use with CodeReady Studio using Runtime Detection. This feature is useful for quickly configuring a server for deploying and testing an application.

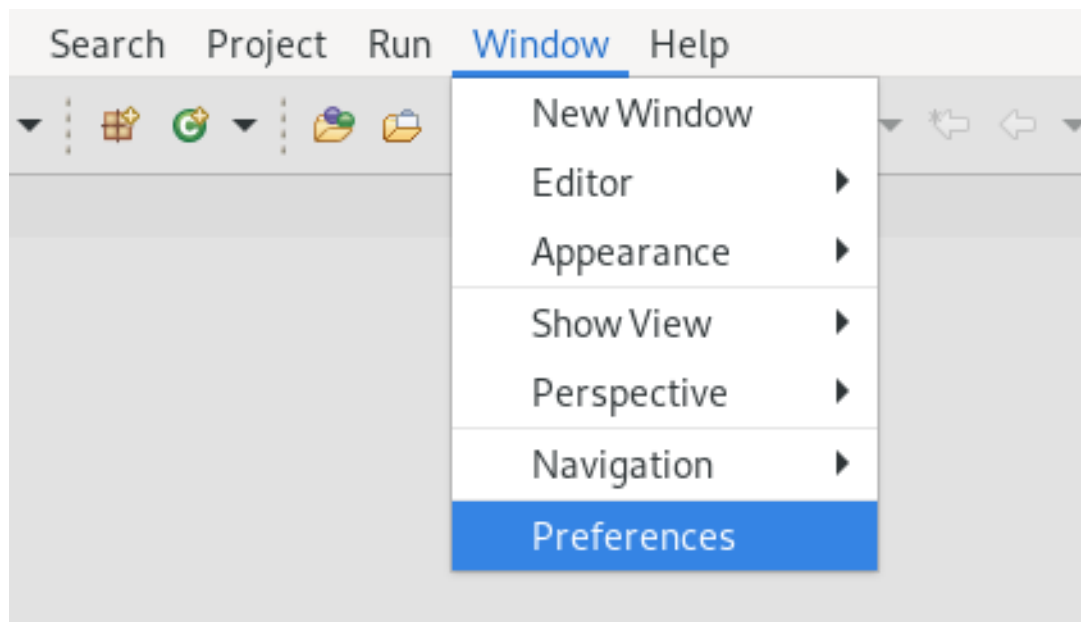
### 3.1. CONFIGURING A LOCAL SERVER

Runtime Detection searches a given local system path to locate certain types of runtime servers. For any servers found, Runtime Detection automatically generates both a default server runtime environment and a default server adapter. These items can be used for immediate application deployment as is or they can be customized to meet your requirements.

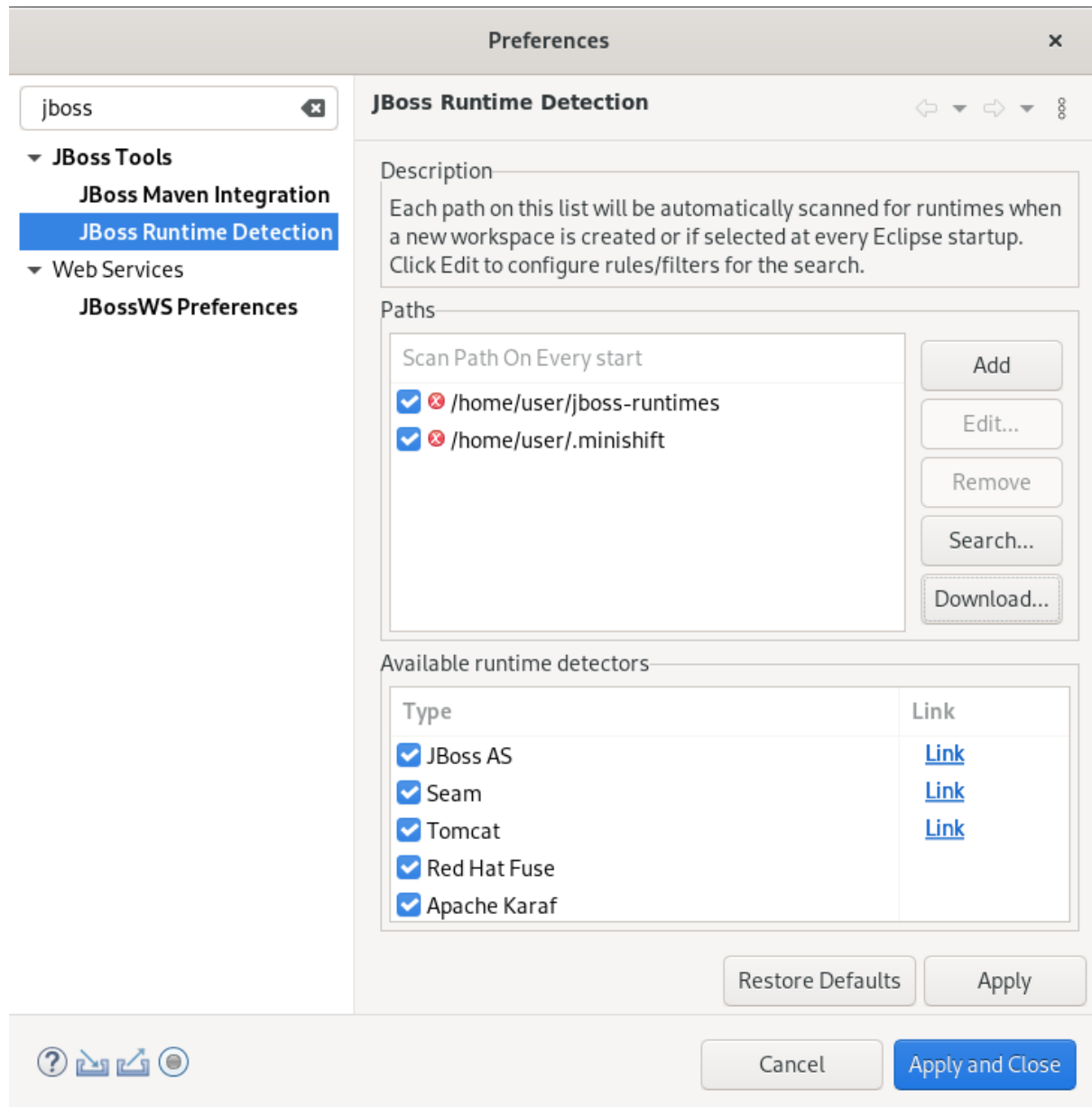
The following section describes how to configure a local server in CodeReady Studio.

#### Procedure

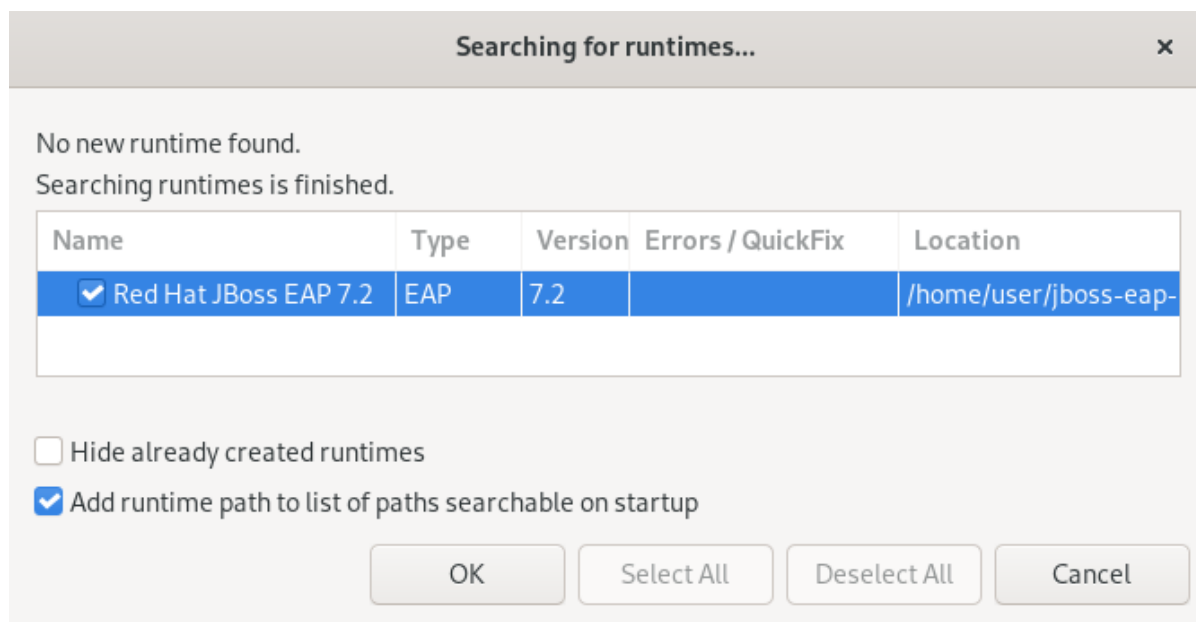
1. Start CodeReady Studio.
2. Click **Window** → **Preferences**.



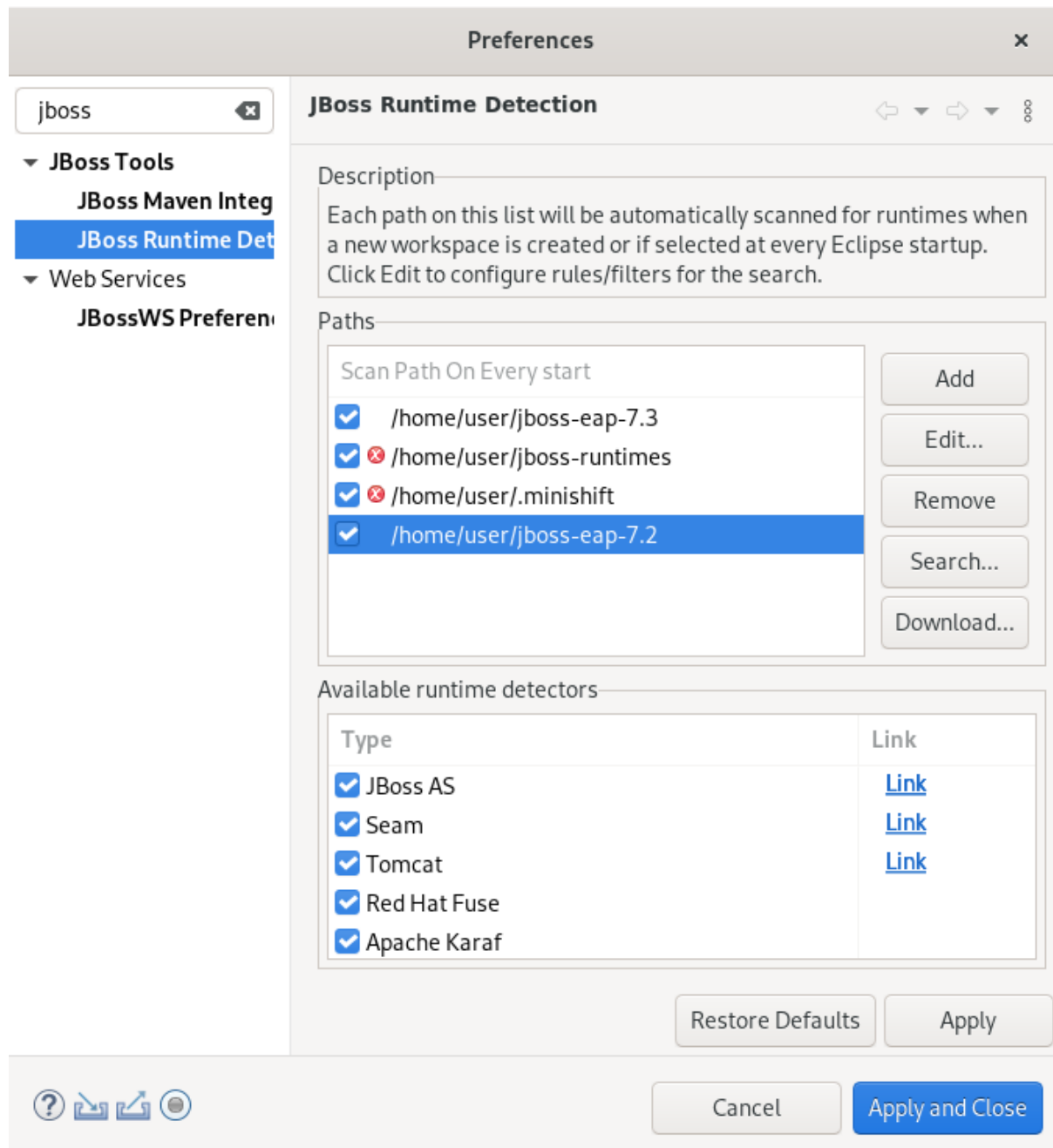
The **Preferences** window appears.



3. Enter **JBoss** in the search field.
4. Select **JBoss Runtime Detection**.
5. Click **Add**.
6. Locate the directory containing the runtime server.
7. Click **Open**.  
The **Searching for runtimes** window appears.



8. Click **OK**.
9. Select the path to the runtime server directory.



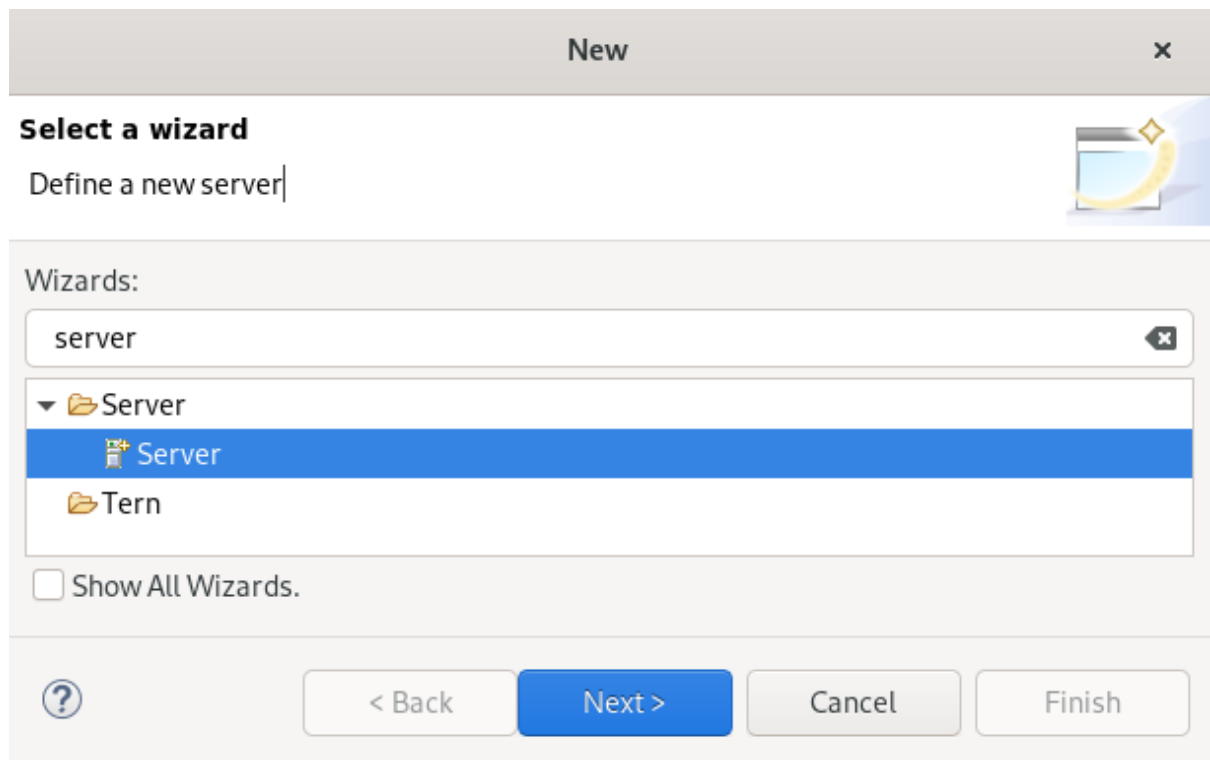
10. Click **Apply and Close**.

## 3.2. CONFIGURING A REMOTE SERVER

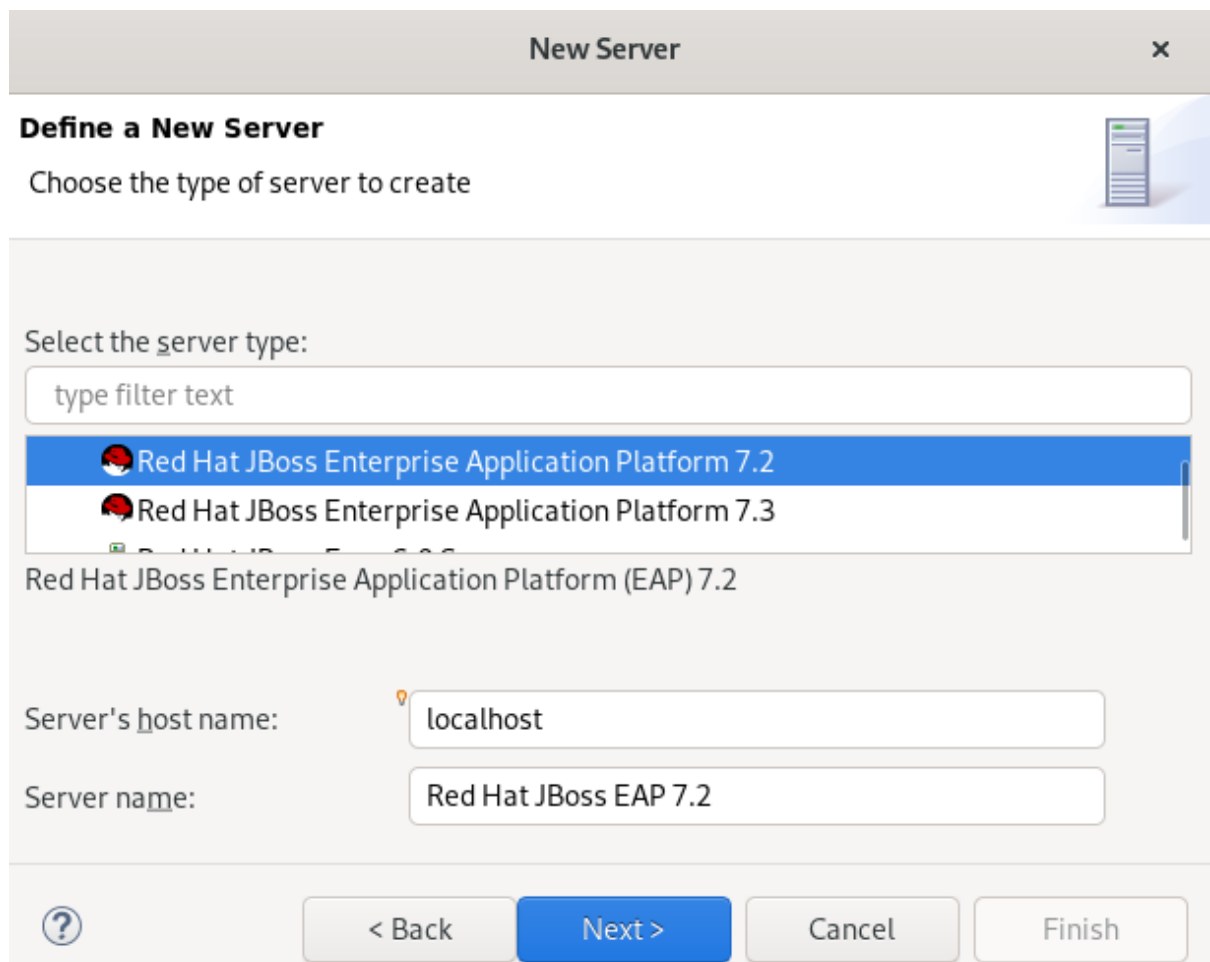
The following section describes how to configure a remote server in CodeReady Studio.

### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



3. Enter **Server** in the search field.
4. Select **Server**.
5. Click **Next**.  
The **Define a New Server** window appears.





6. Select a server type.
7. Click **Next**.  
The **Create a new Server Adapter** window appears.

**New Server** ×

**Create a new Server Adapter** **RED HAT JBOSS MIDDLEWARE**

Red Hat JBoss Enterprise Application Platform (EAP) 7.2

A Server Adapter manages starting and stopping instances of your server. It manages command line arguments and keeps track of which modules have been deployed.

The server is:  Local  
 Remote

Controlled by:  Filesystem and shell operations  
 Management Operations

Server lifecycle is externally managed.

The selected profile does not require a runtime, though some features (ex: JMX) may not be available without one.

Assign a runtime to this server

Create new runtime (next page) ▼

?
< Back
Next >
Cancel
Finish

8. Select the **Remote** check box.
9. Select a **Controlled by** option.
10. Select the **Server lifecycle externally managed** check box.
11. Select the **Assign a runtime to the server** check box.
12. Click **Next**.  
The **JBoss Runtime** window appears.

**New Server** [Close]

**JBoss Runtime** **RED HAT JBOSS MIDDLEWARE**

Red Hat JBoss Enterprise Application Platform (EAP) 7.2

A JBoss Server runtime references a JBoss installation directory. It can be used to set up classpaths for projects which depend on this runtime, as well as by a "server" which will be able to start and stop instances of JBoss.

Name:

Home Directory:  [Download and install runtime...](#)

Runtime JRE:

Execution Environment:

Alternate JRE:

Server base directory:

Configuration file:

[?]

13. Click **Browse** in the **Home Directory** field to locate the runtime server.

14. Click **Next**.  
The **Remote System Integration** window appears.

**New Server** [Close]

**Remote System Integration** **JBoss by Red Hat**

Please set the properties required for connecting to a remote system.

Host:   [Open Remote System Explorer View...](#)

Remote Runtime Details:

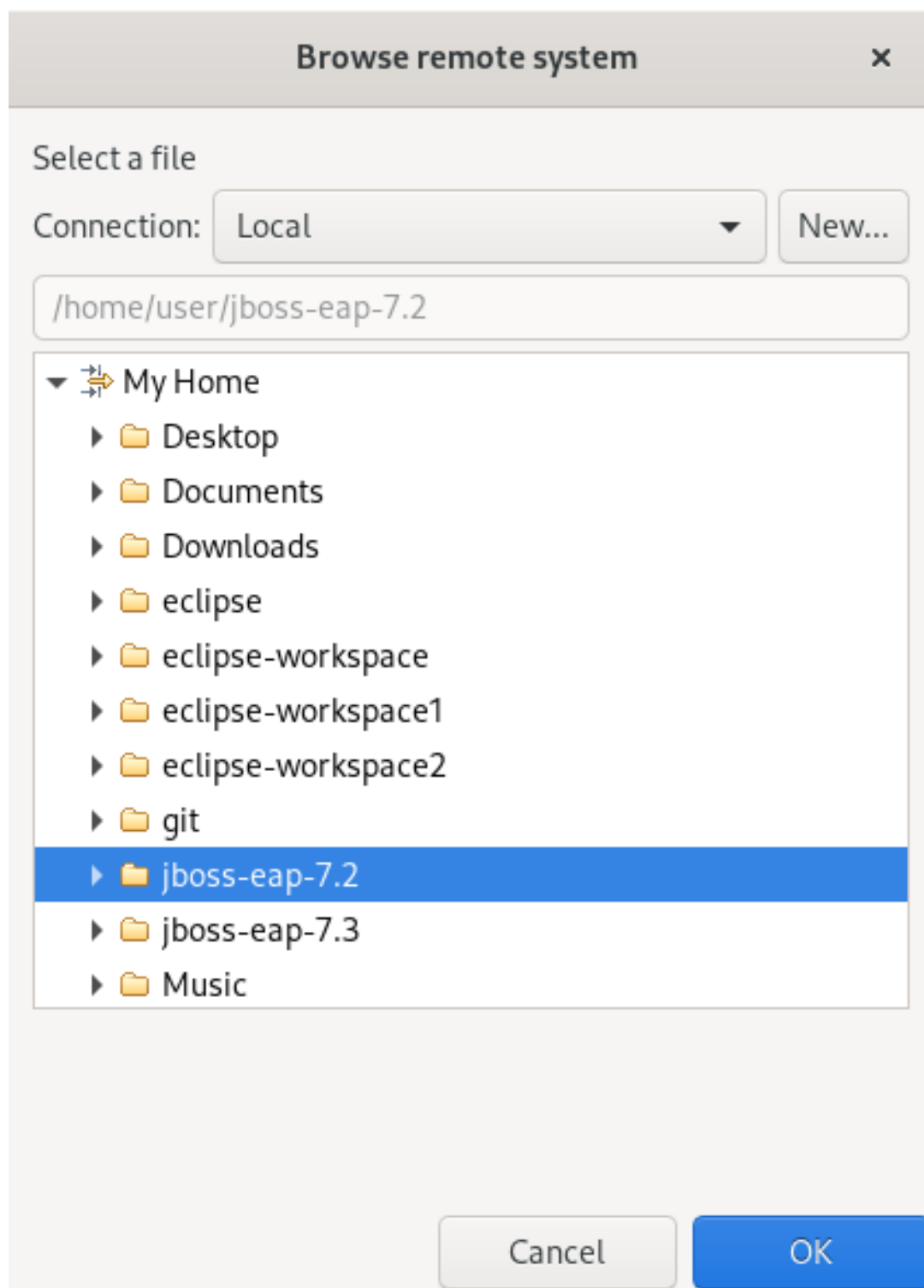
Remote Server Home:

Remote Server Base Directory:

Remote Server Configuration File:

[?]

15. Click **Browse** in the **Remote Server Home** field.  
The **Browse remote system** window appears.



16. Specify the path to the directory that contains the remote server.

17. Click **Finish**.

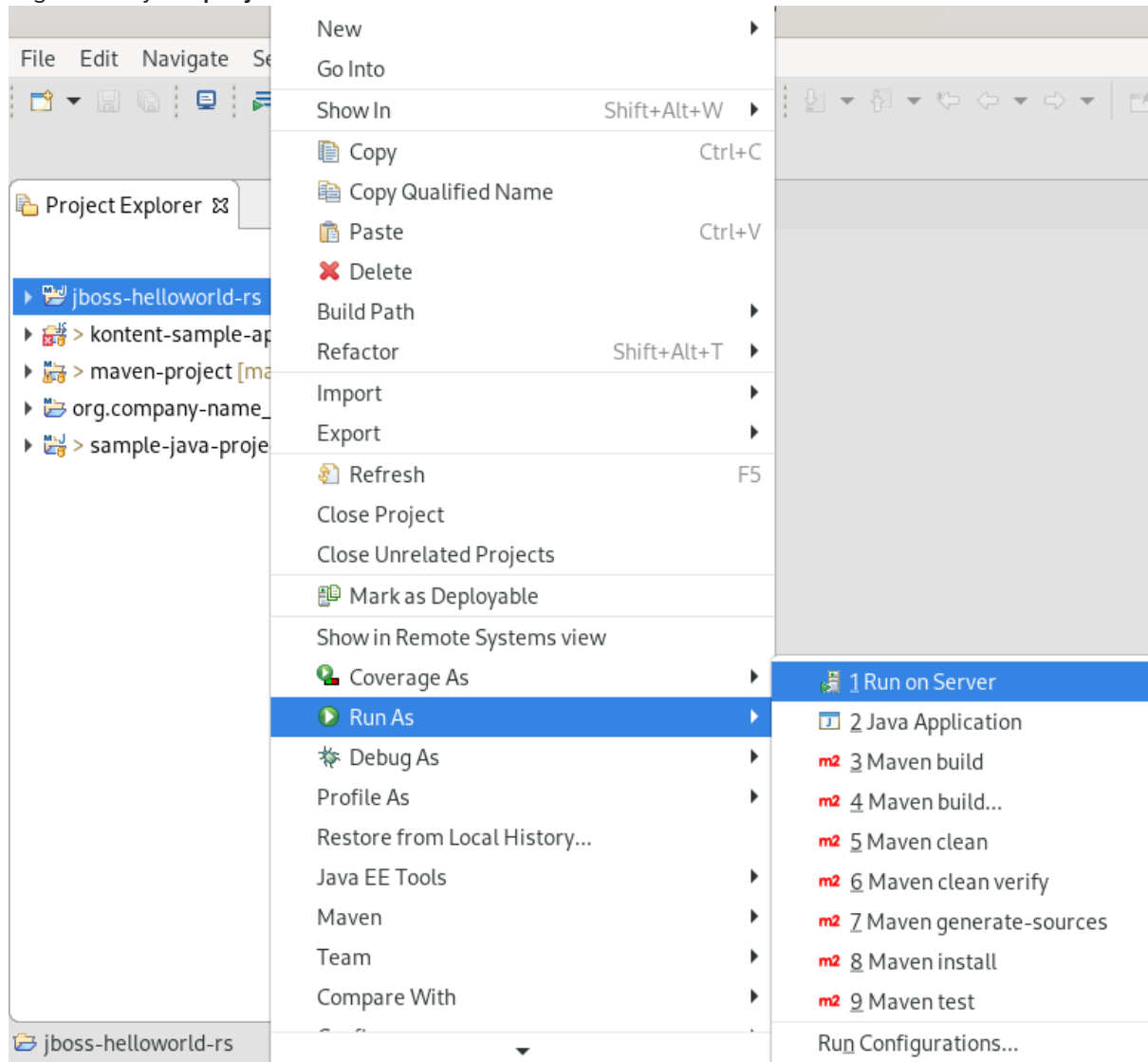
### 3.3. DEPLOYING AN APPLICATION

After configuring the local server, you can deploy applications to the server from CodeReady Studio using the server adapter. The server adapter enables runtime communication between the server and CodeReady Studio for easy deployment of applications and server management.

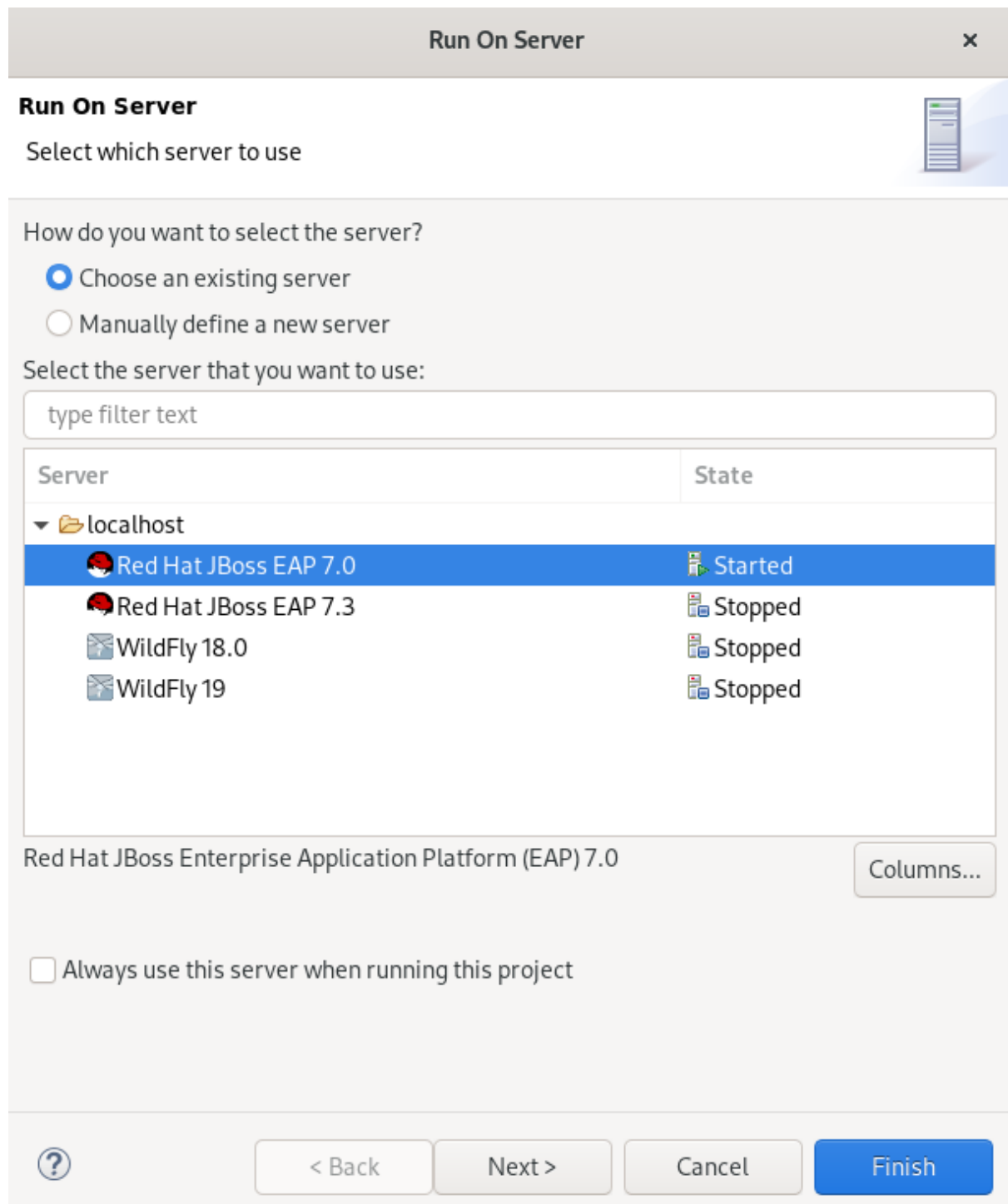
The following section describes how to deploy an application to the server in CodeReady Studio.

## Procedure

1. Start CodeReady Studio.
2. Right-click your **project** → **Run as** → **Run on Server**.



The **Run on Server** window appears.



3. Select the **Choose an existing server** check box.
4. Select the server you want to deploy.
5. Click **Finish**.

Your application opens in the internal CodeReady Studio web browser.

## CHAPTER 4. JBOSS EAP AND JBOSS WFK BASICS IN CODEREADY STUDIO

The Eclipse IDE supports application development and deployment with Red Hat JBoss Enterprise Application Platform (JBoss EAP) and Red Hat JBoss Web Framework Kit (JBoss WFK).

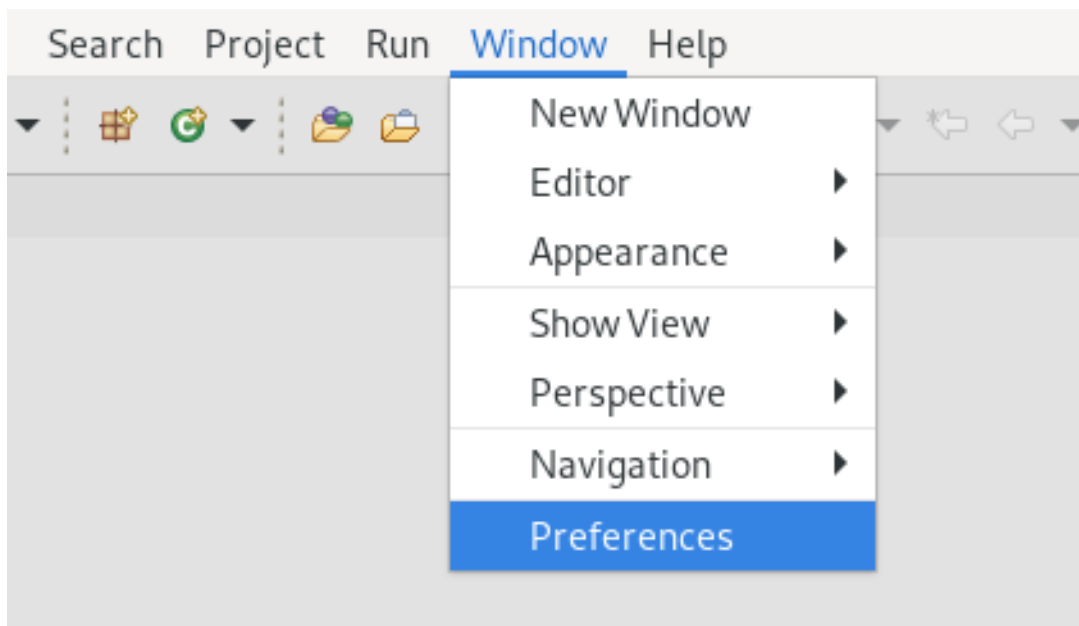
However, you need to configure Maven repositories first. This configuration is essential for using the enterprise versions of the example Maven projects provided in Red Hat Central. These projects are intended for deployment to JBoss EAP and require IDE access to JBoss EAP and JBoss WFK repositories.

### 4.1. CONFIGURING MAVEN REPOSITORIES

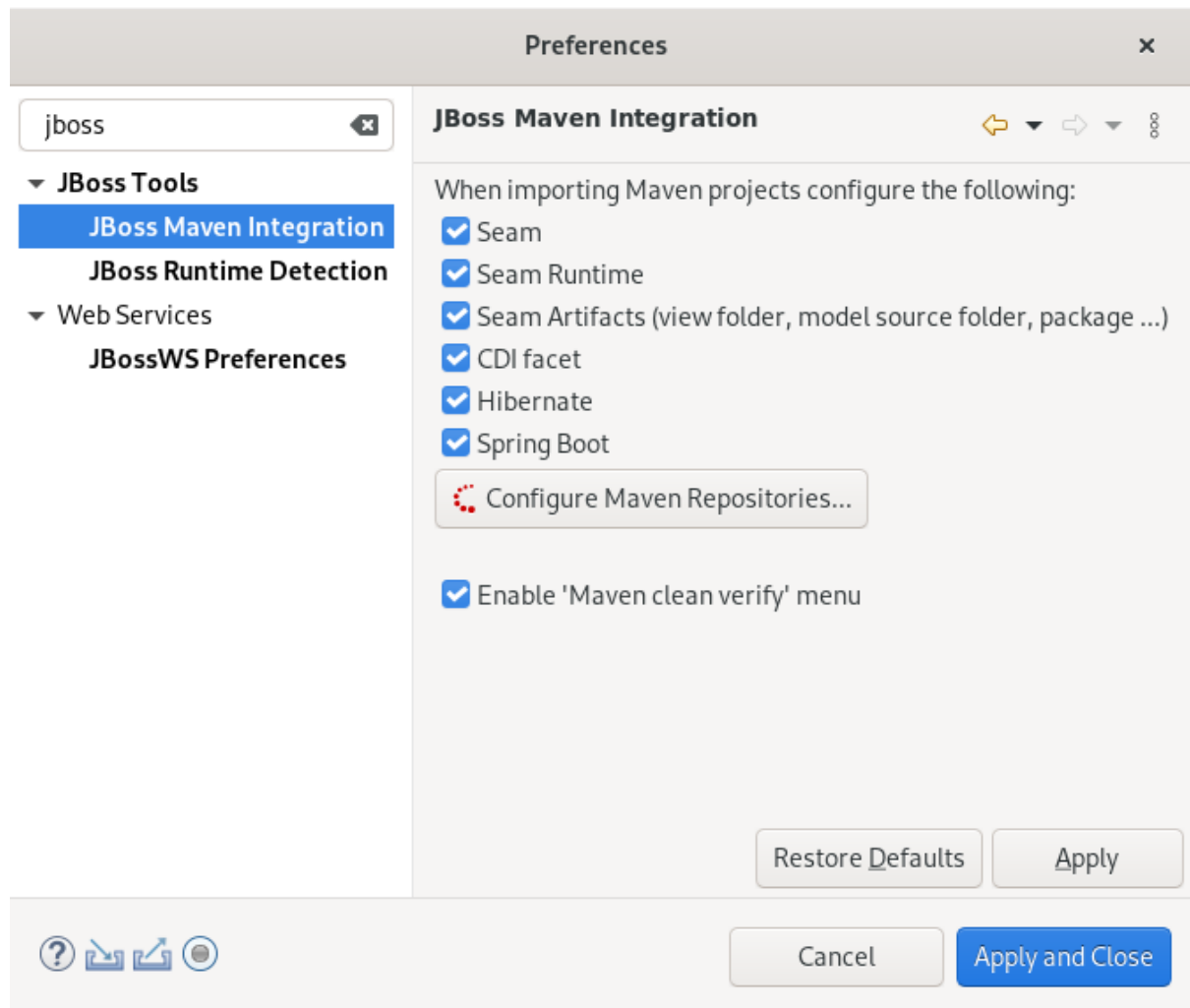
The following section describes how to configure Maven repositories.

#### Procedure

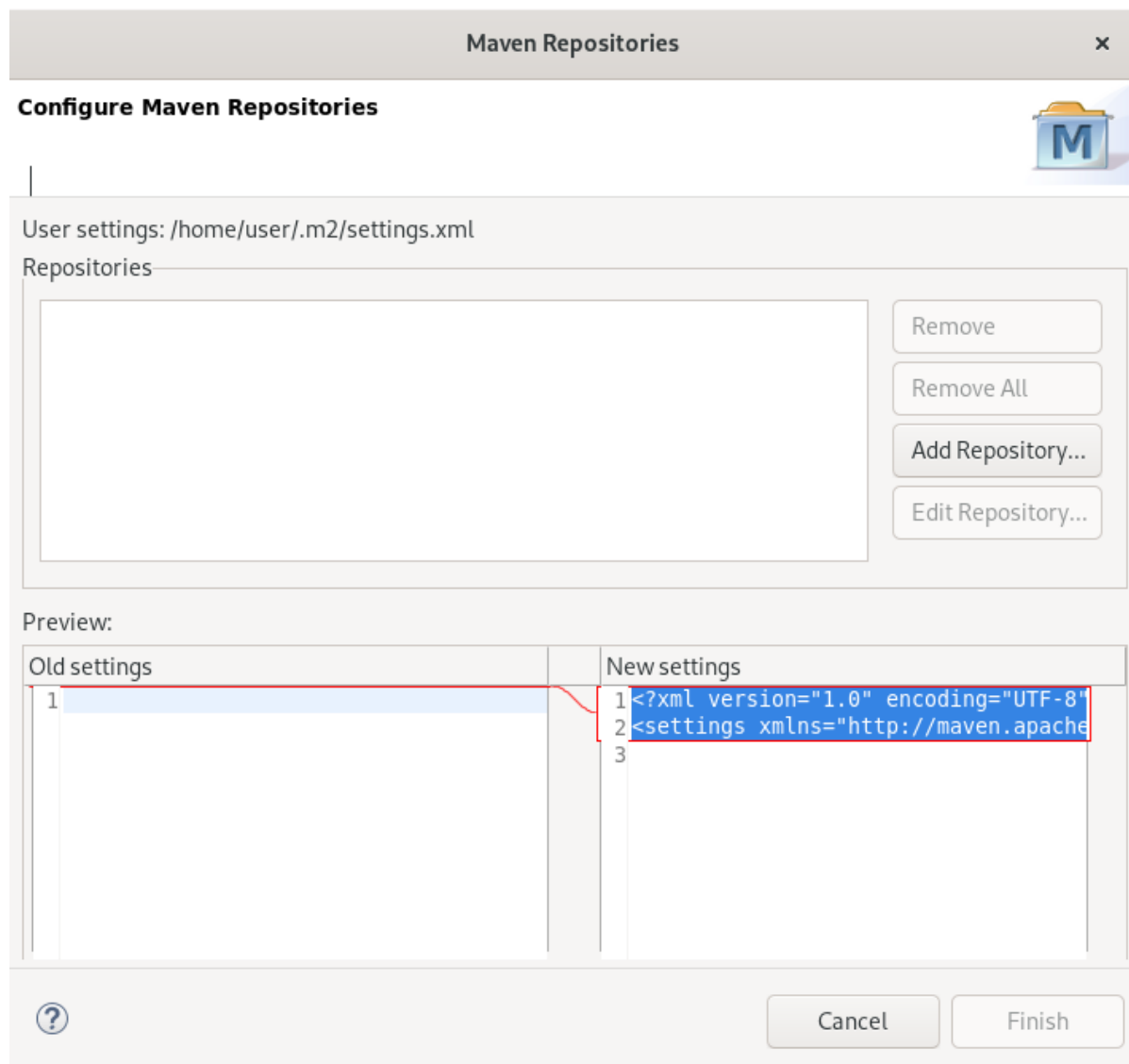
1. Start CodeReady Studio.
2. Click **Window** → **Preferences**.



The **Preferences** window appears.



3. Enter **JBoss** in the search field.
4. Select **JBoss Maven Integration**.
5. Click **Configure Maven Repositories**.  
The **Configure Maven Repositories** window appears.



6. Click **Add Repository**.  
The **Add Maven Repository** window appears.



7. Click the down-arrow in the **Profile ID** field.
8. Select the **redhat-ga-repository**.  
Other fields are populated automatically.
9. Click **OK**.
10. Click **Finish**.  
The **Confirm File Update** window appears.
11. Click **Yes**.
12. Click **Apply and Close**.

#### Additional resources

- For more information on Maven repositories, see [Maven: Getting Started - Developers](#).

## 4.2. SETTING UP JBOSS EAP

To set up JBoss EAP in Eclipse, you must direct the IDE to the local or remote runtime servers. This establishes a communication channel between the IDE and the JBoss EAP server for efficient deployment and server management workflows.

The following section describes how to install JBoss EAP in CodeReady Studio.

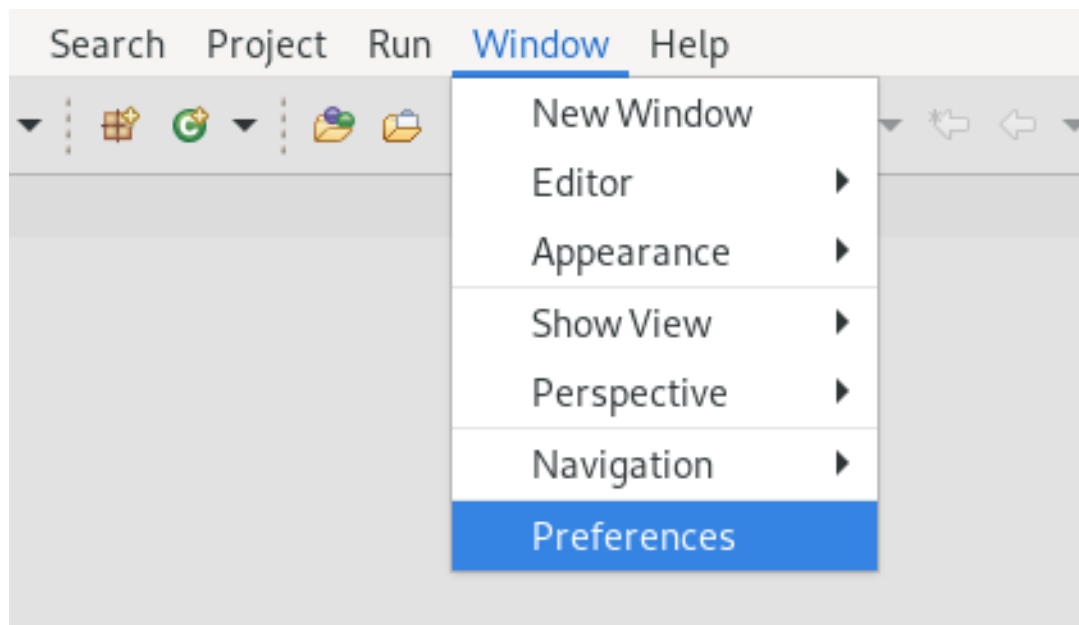
#### Prerequisites

- Configured Maven repositories.

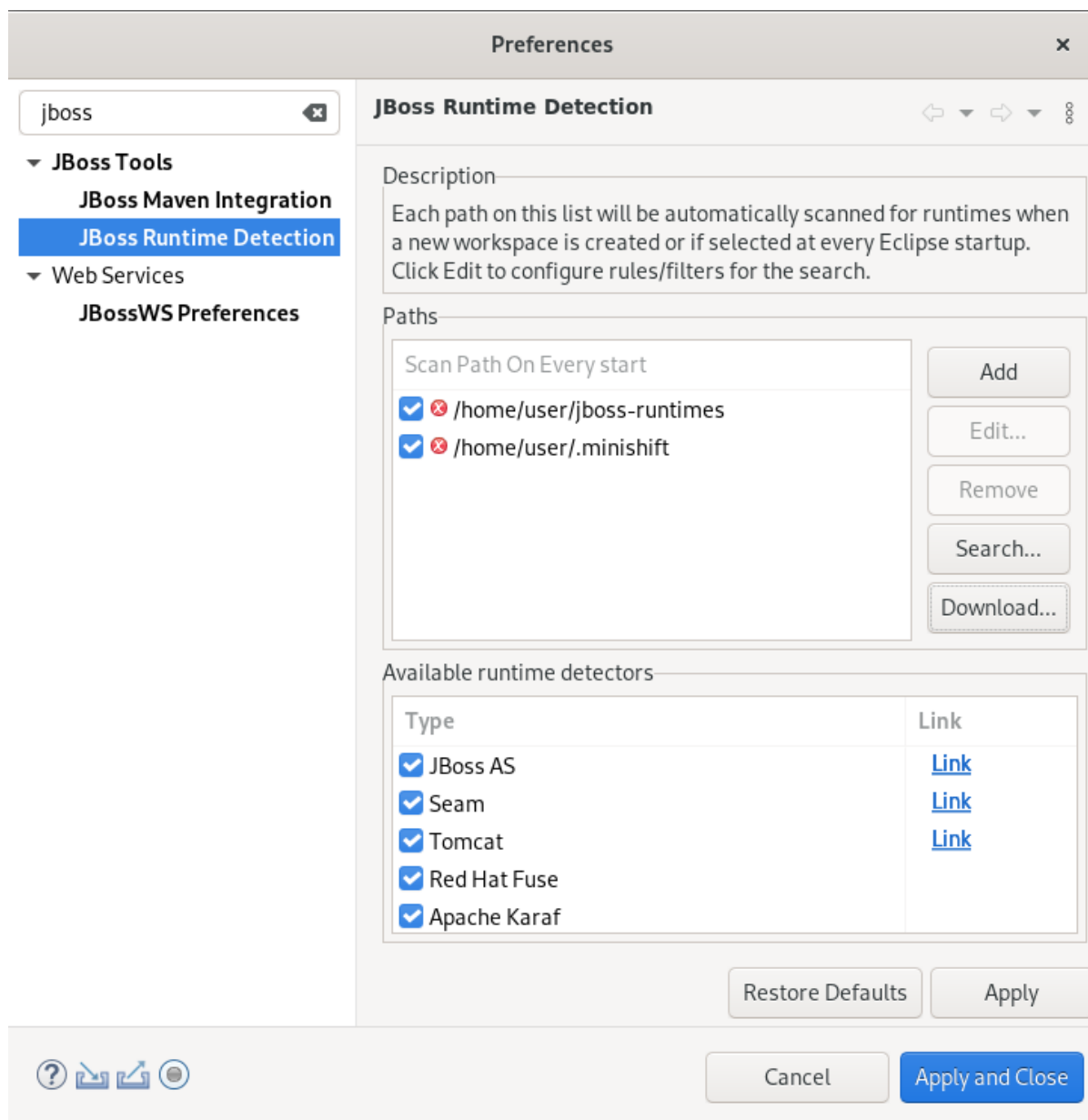
For more information on how to configure Maven repositories, see [Section 4.1, “Configuring Maven repositories”](#).

## Procedure

1. Start CodeReady Studio.
2. Click **Window** → **Preferences**.



The **Preferences** window appears.



3. Enter **JBoss** in the search field.
4. Select **JBoss Runtime Detection**.
5. Click **Download**.  
The **Download Runtimes** window appears.

✕
Download Runtimes

**Download Runtimes**

Please select a runtime to download and install.

Name	Version
Red Hat JBoss EAP 6.0.0	6.0.0
Red Hat JBoss EAP 6.0.1	6.0.1
Red Hat JBoss EAP 6.1.0	6.1.0
Red Hat JBoss EAP 6.2.0	6.2.0
Red Hat JBoss EAP 6.3.0	6.3.0
Red Hat JBoss EAP 6.4.0	6.4.0
Red Hat JBoss EAP 7.0.0	7.0.0
Red Hat JBoss EAP 7.1.0	7.1.0
Red Hat JBoss EAP 7.2.0	7.2.0
Red Hat JBoss EAP 7.3.0	7.3.0

Selected Runtime Details

Project URL: <https://developers.redhat.com/products/eap>

Download URL: <https://www.jboss.org/download-manager/jdf/file/jboss-eap-7.3.0.zip>

Registration required. Downloads require accepting the terms and conditions of the JBoss Developer Program which provides \$0 subscription for development use only.

?

< Back
Next >
Cancel
Finish

6. Select the JBoss EAP version you need.



#### NOTE

If you select the JBoss EAP version 6.0.x or earlier, follow the on-screen instructions. If you select a later version, follow the instructions below.

7. Click **Next**.  
The **Credentials** window appears.

8. Click **Add**.
9. Enter your **access.redhat.com** username and password.
10. Click **OK**.
11. Click **Next**.  
Review the license agreement, if satisfied, accept the license and click **Next** to continue with the installation.

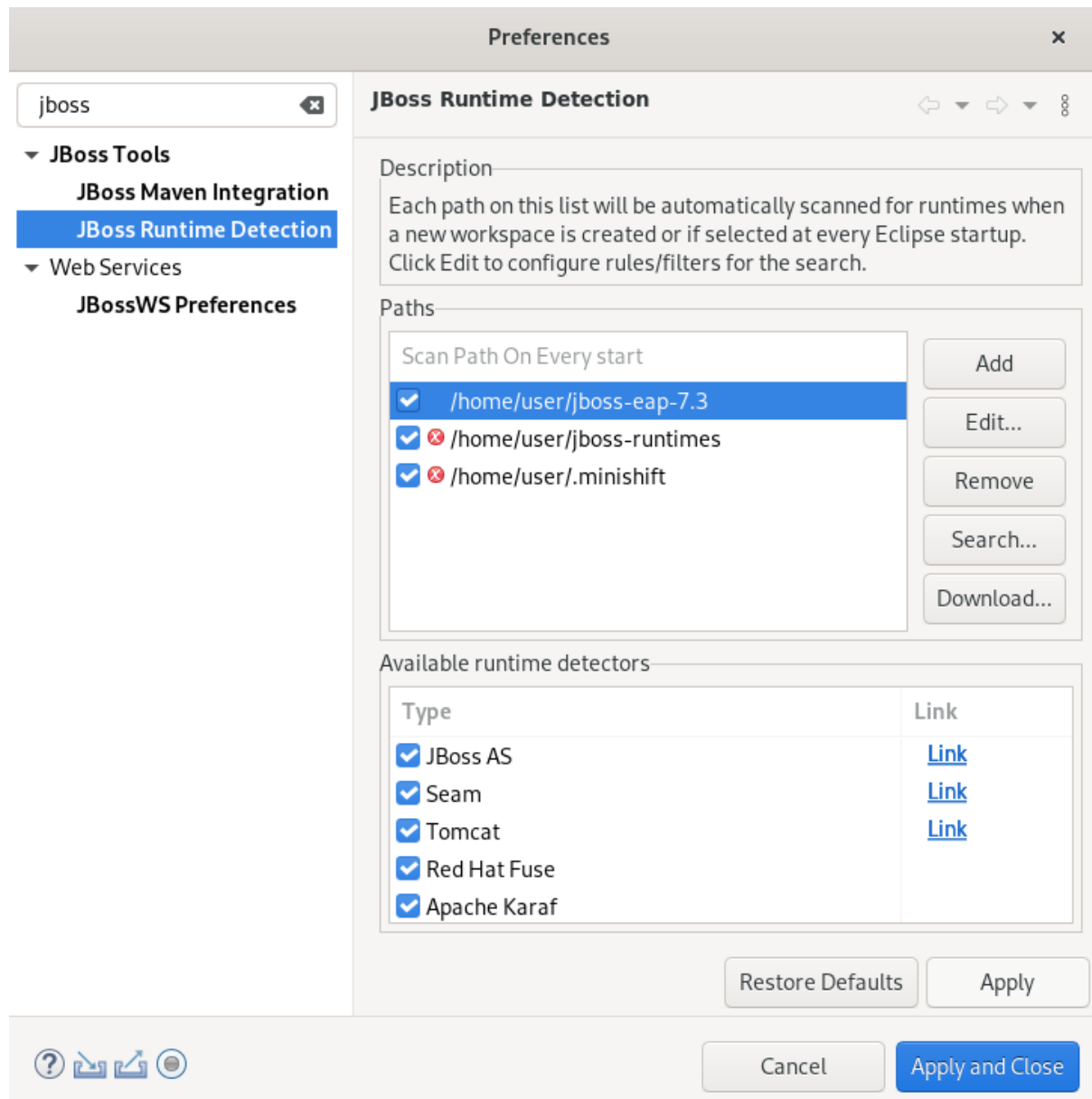
The **Download Runtimes** window appears.

12. Click **Browse** to select the **Install folder**.
13. Click **Browse** to select the **Download folder**.

- Click **Finish**.

Note that downloading and installing the Runtime might take a while.

The **JBoss Runtime Detection** window appears.

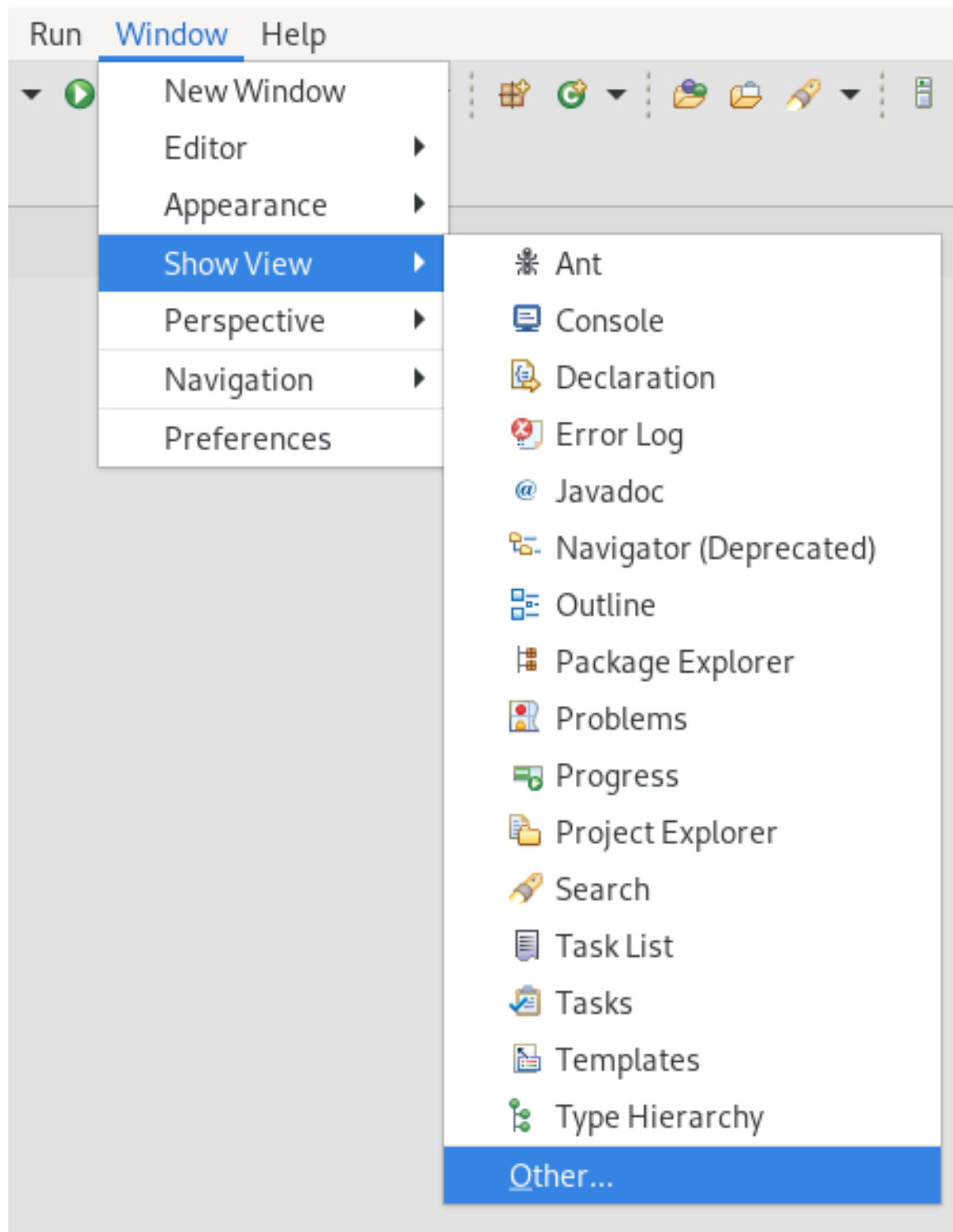


- Select the path to the JBoss EAP installation file check box.

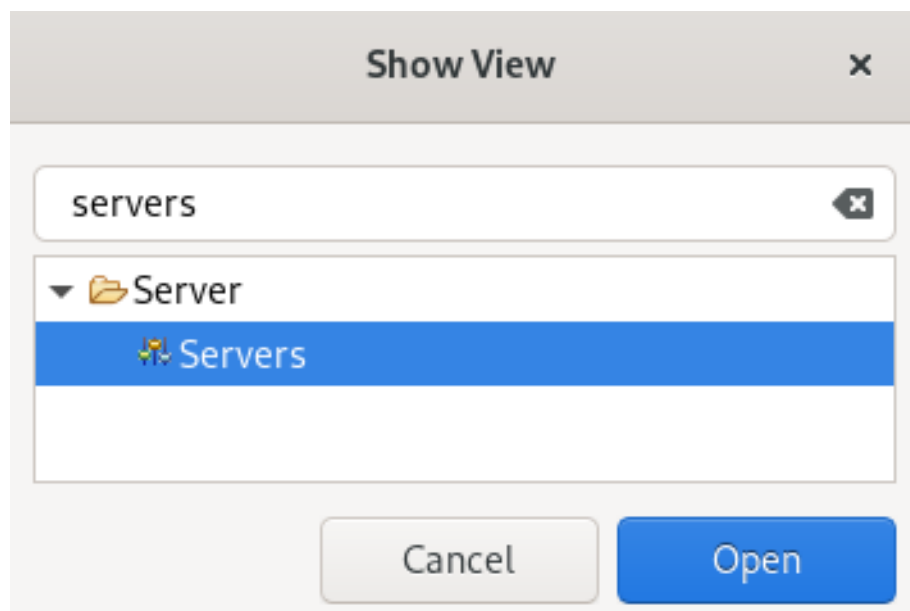
- Click **Apply and Close**.

### Verification steps

- Click **Window** → **Show View** → **Other**.

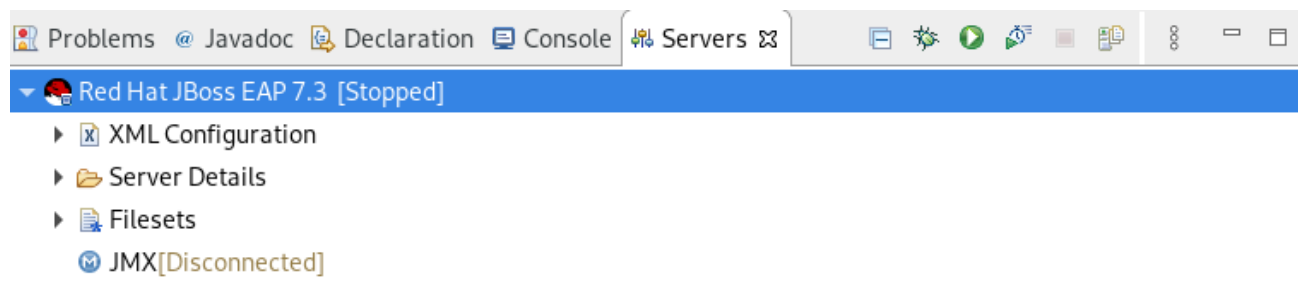


The **Show View** window appears.



2. Type **Servers** in the search field.
3. Select **Servers**.
4. Click **Open**.  
The **Servers** view appears.

Your newly added JBoss EAP is now listed in the **Servers** view.





## CHAPTER 5. OPENSIFT BASICS IN CODEREADY STUDIO

CodeReady Studio includes the OpenShift Application Explorer view, which provides a simplified user experience allowing easy and rapid feedback through the inner loop as well as debugging.

The OpenShift Application Explorer is set in CodeReady Studio as the default view. In case you need to open it manually, follow the instructions in [Setting Up Openshift App Explorer View](#).

### Prerequisites

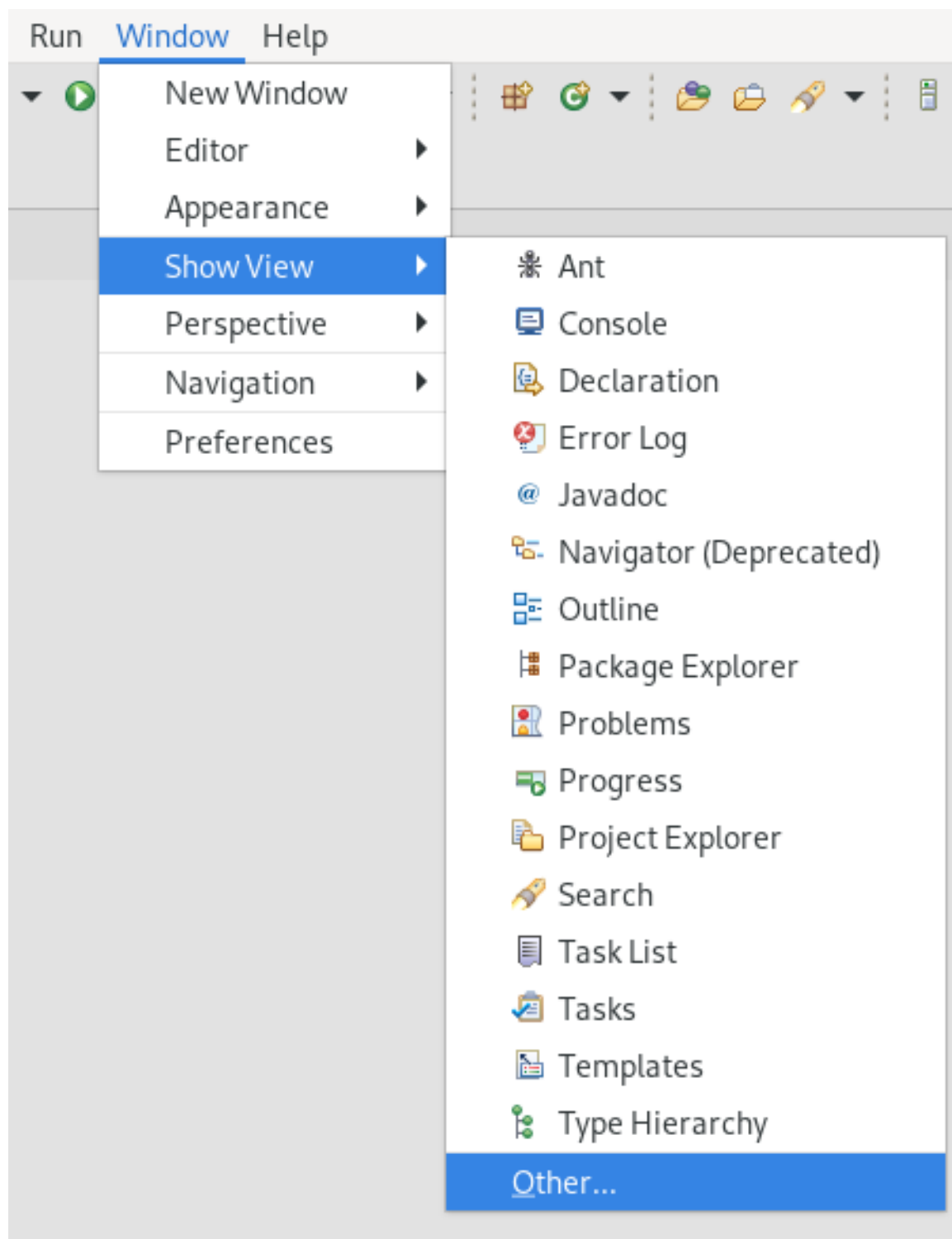
- A running OpenShift cluster.

### 5.1. SETTING UP THE OPENSIFT APPLICATION EXPLORER VIEW

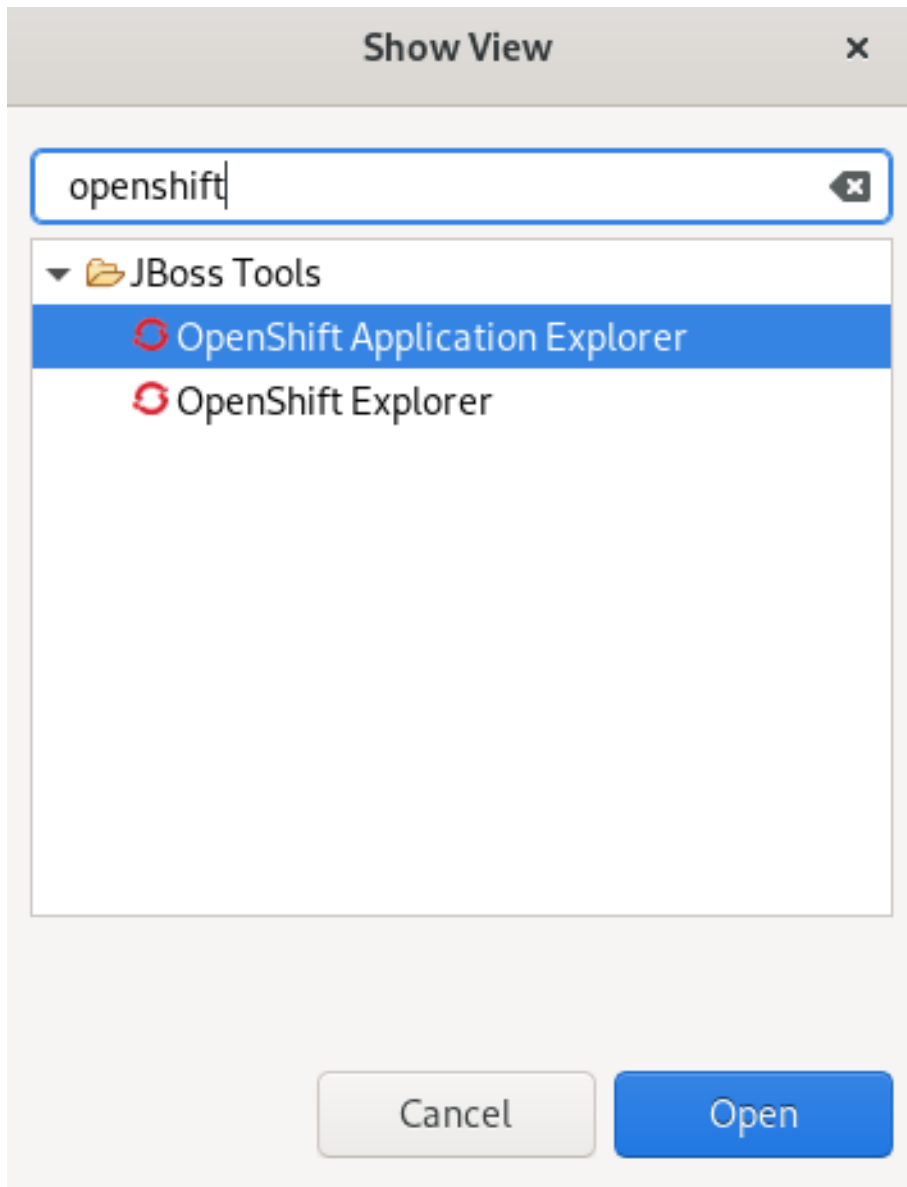
The following section describes how to open OpenShift Application Explorer in CodeReady Studio.

#### Procedure

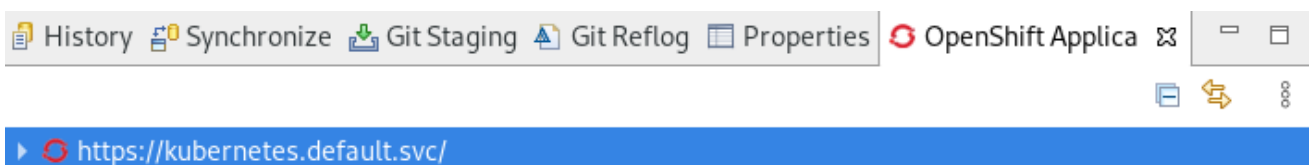
1. Start CodeReady Studio.
2. Click **Window** → **Show View** → **Other**.



The **Show View** window appears.



3. Enter **OpenShift** in the search field.
4. Select **OpenShift Application Explorer**.
5. Click **Open**.  
The **OpenShift Application Explorer** view appears.

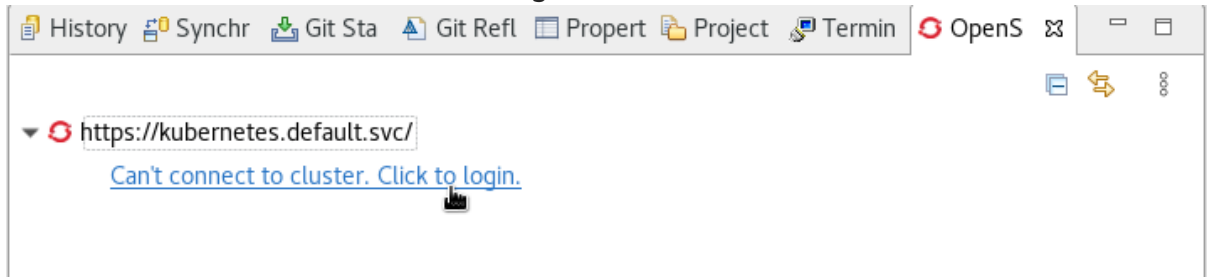


## 5.2. CONNECTING TO AN OPENSIFT CLUSTER USING OPENSIFT APPLICATION EXPLORER

The following section describes how to login to an OpenShift cluster in CodeReady Studio using OpenShift Application Explorer.

### Procedure

1. Start CodeReady Studio.
2. Open **OpenShift Application Explorer**.
3. Click **Can't connect to cluster. Click to login**



The **Login** window appears.

 A screenshot of the 'Login' dialog box. The title bar says 'Login'. Below the title bar, it says 'Sign in to OpenShift' with the OpenShift logo. A red error icon and text state 'User and password or token must be provided'. The dialog contains four input fields: 'URL:' with the value 'https://api.crc.testing:6443/' and a 'Paste login command' button; 'Username:'; 'Password:'; and 'Token:' with a 'Retrieve token' button. At the bottom, there is a help icon (question mark in a circle), a 'Cancel' button, and a 'Finish' button.

4. Paste your OpenShift API URL into the **URL** field.  
For more information on accessing your cluster through OpenShift API URL, visit [Red Hat OpenShift - Accessing your Services](#).
5. Enter your username and password or token.
6. Click **Finish**.

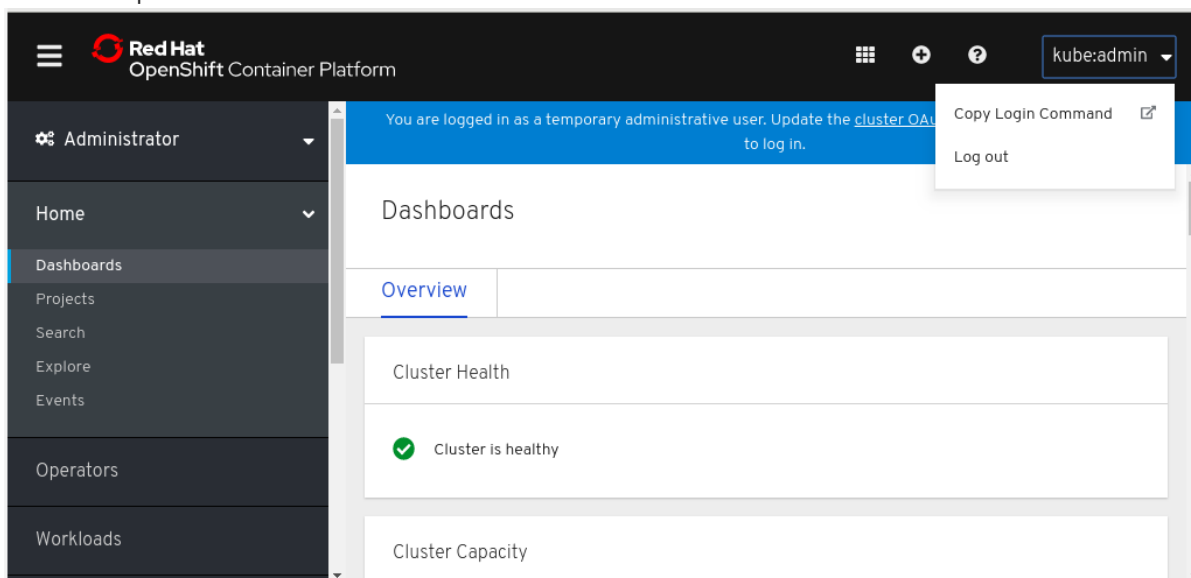
## 5.3. CONNECTING TO AN OPENSIFT CLUSTER USING BROWSER-BASED TOKEN RETRIEVAL

Alternatively to providing your username and password or token to the OpenShift Application Explorer, you can use browser based token retrieval to log in to your OpenShift cluster. There are two login options, pasting your login command or retrieving your token.

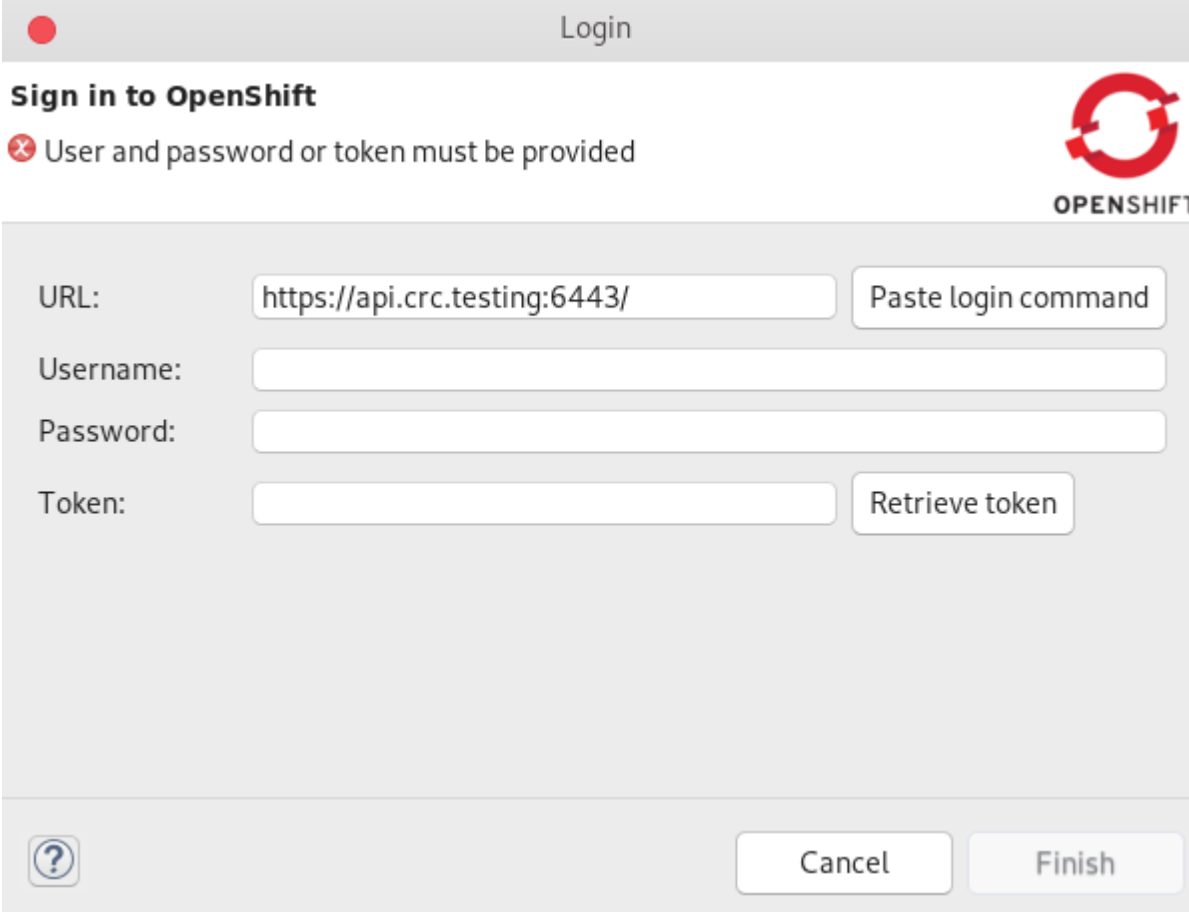
### 5.3.1. Pasting your login command

#### Procedure

1. Paste your OpenShift API URL into the **URL** field.  
For more information on accessing your cluster through OpenShift API URL, visit [Red Hat OpenShift - Accessing your Services](#).
2. Visit the OpenShift Container Platform web UI.



3. Click the drop-down menu in the top right corner.
4. Click **Copy Login Command**.
5. Click **Display Token**.
6. Copy the login command.
7. In the **Sign in to OpenShift** window, click **Paste login command**.



**Sign in to OpenShift**

✖ User and password or token must be provided

URL:

Username:

Password:

Token:

8. Click **Finish**.



#### NOTE

For OpenShift 3, the login command is copied into your clipboard automatically.

### 5.3.2. Retrieving your token

#### Procedure

1. Paste your OpenShift API URL into the **URL** field.  
For more information on accessing your cluster through OpenShift API URL, visit [Red Hat OpenShift - Accessing your Services](#).
2. Click **Retrieve token**.

Login

**Sign in to OpenShift**

✖ User and password or token must be provided

OPENSIFT

URL:

Username:

Password:

Token:

3. Enter your username and password.
4. Click **Log in**.
5. Click **Display Token**.
6. Click **Finish**.

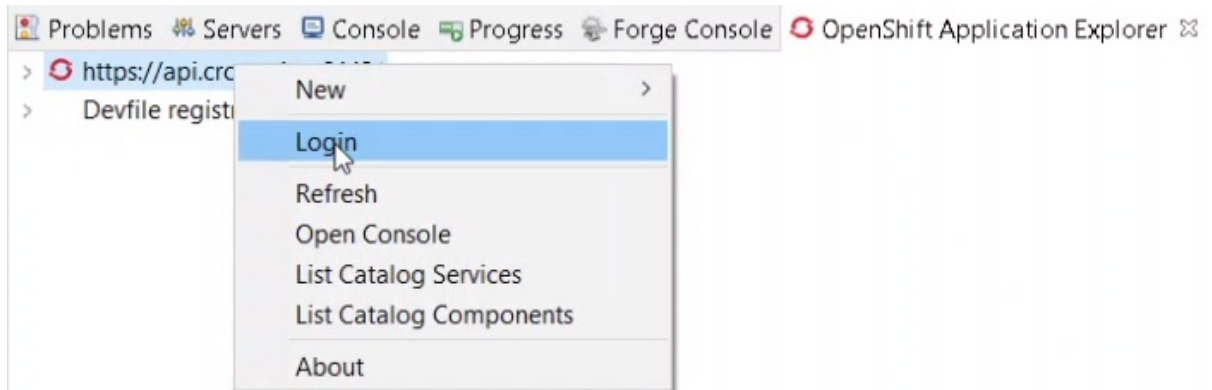
Your projects now appear in the **OpenShift Application Explorer** view.

## 5.4. SETTING UP A DEVELOPER SANDBOX USING OPENSIFT TOOLS

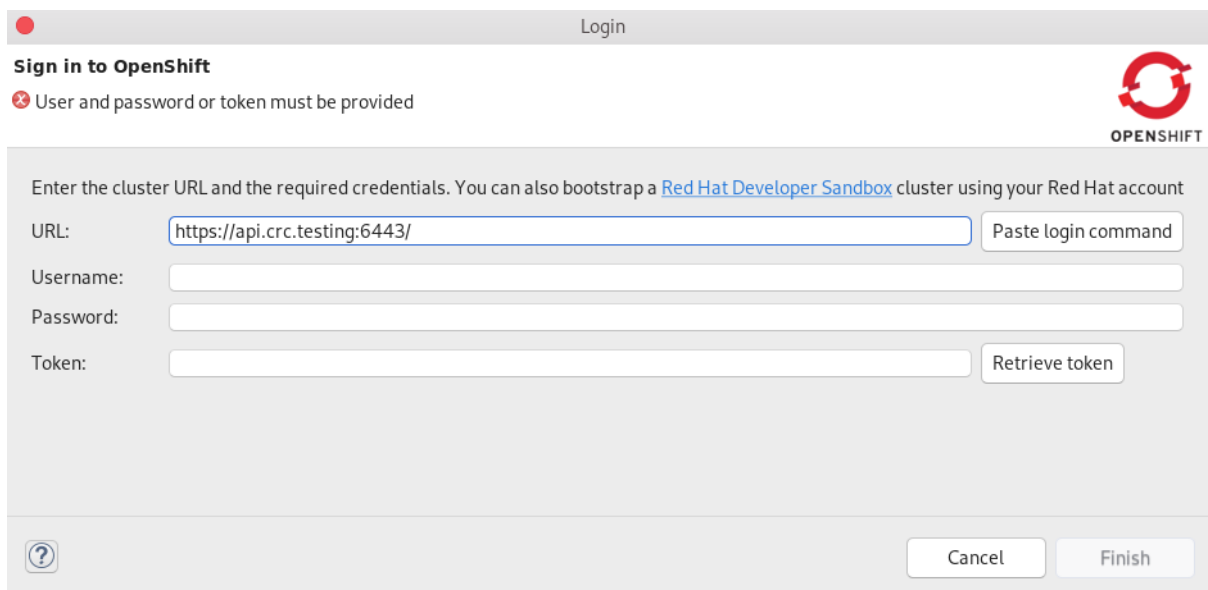
The following section describes how to bootstrap and login to a Developer Sandbox in CodeReady Studio.

### Procedure

1. Start CodeReady Studio.
2. Open **OpenShift Application Explorer**.

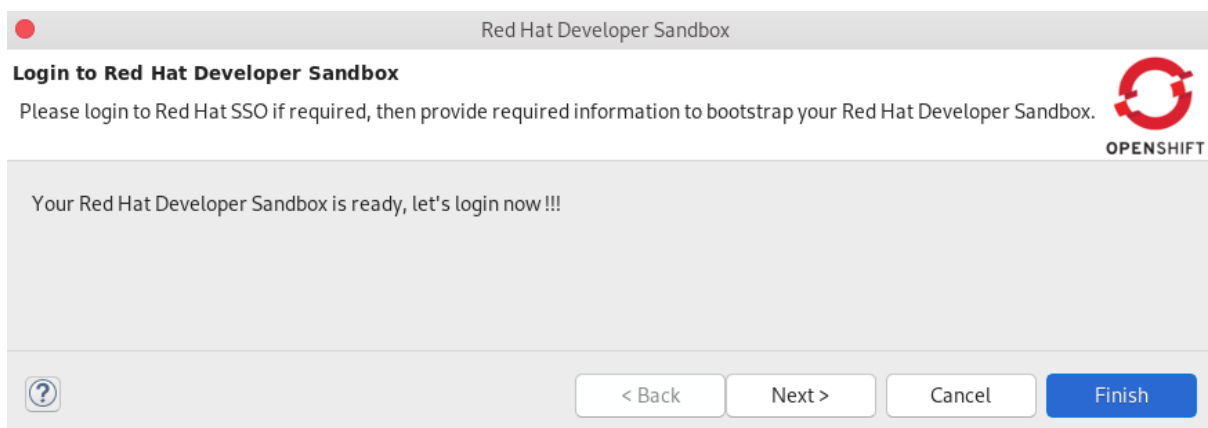


3. Right-click your OpenShift connection.
4. Click **Login**.  
The **Sign in to OpenShift** window appears.



5. Click **Red Hat Developer Sandbox**
6. Provide the credentials of your Red Hat account and click **Log in**.  
Your Developer Sandbox has been bootstrapped.

The **Login to Red Hat Developer Sandbox** window appears.



7. Click **Next**.



8. Click **DevSandbox**.
9. Provide the credentials of your Red Hat account again and click **Log in**.
10. Click **Display Token**.
11. Click **Finish**.  
Your Token is displayed in the **Sign in to OpenShift** window.

12. Click **Finish**.  
You are now logged in to your Developer Sandbox.

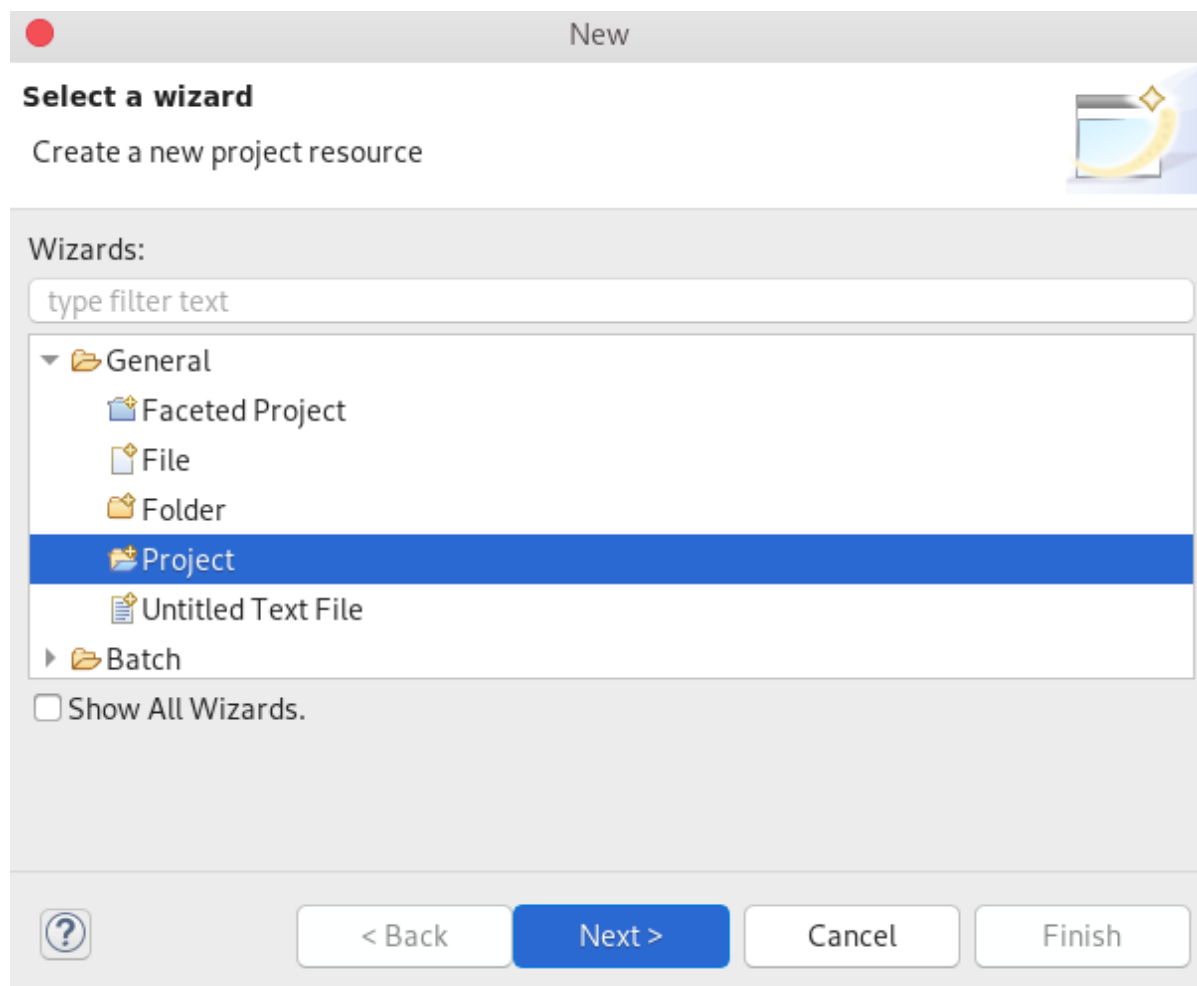
Your Developer Sandbox shows in the OpenShift Application Explorer view.

## 5.5. BUILDING AN APPLICATION BASED ON DEVFILES

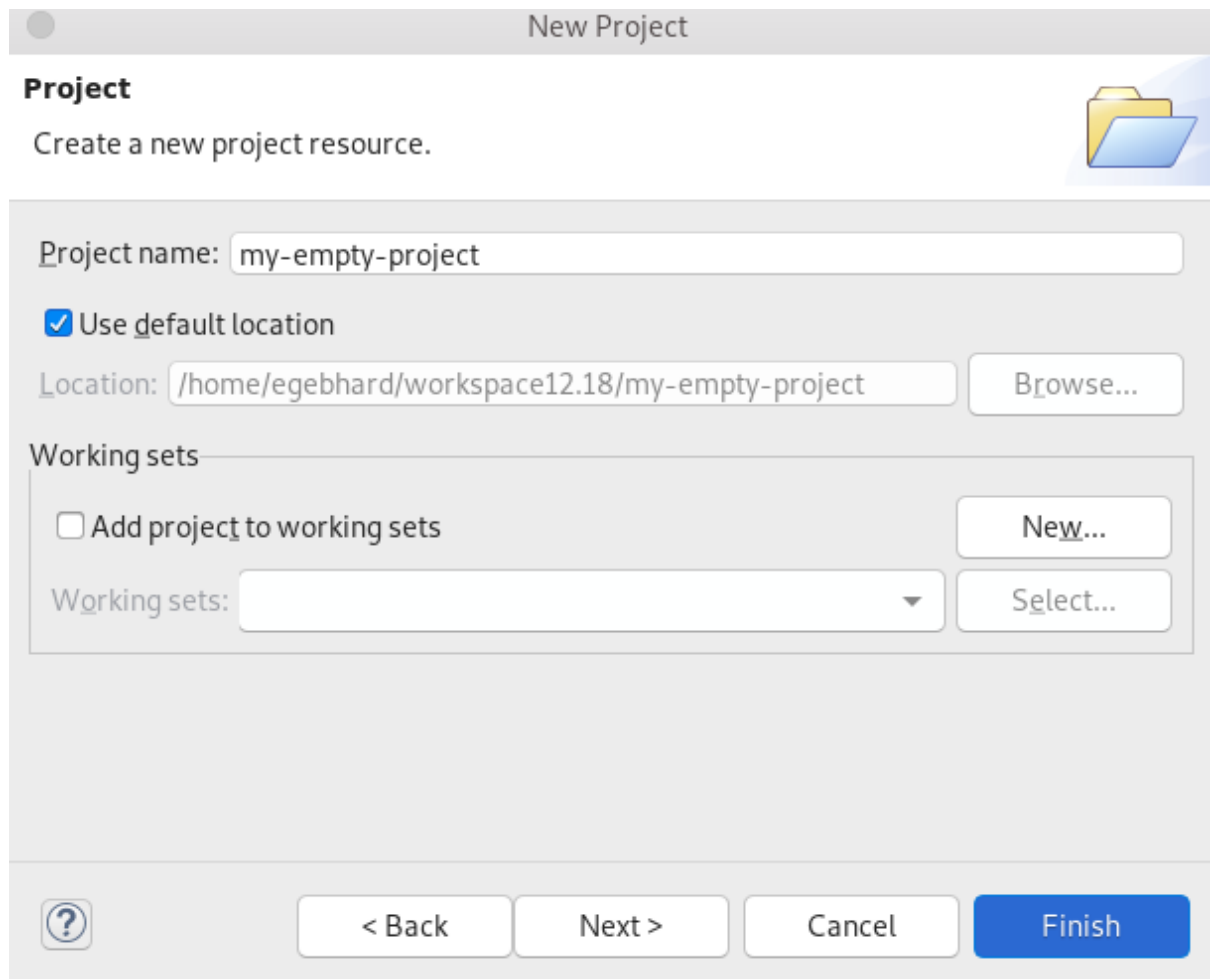
To deploy applications based on devfiles, you need an empty project in your local workspace as well as an empty project in OpenShift, for which you need to create a devfile component. After the component is established, your project will be updated and local and remote artifacts created in OpenShift.

### Procedure

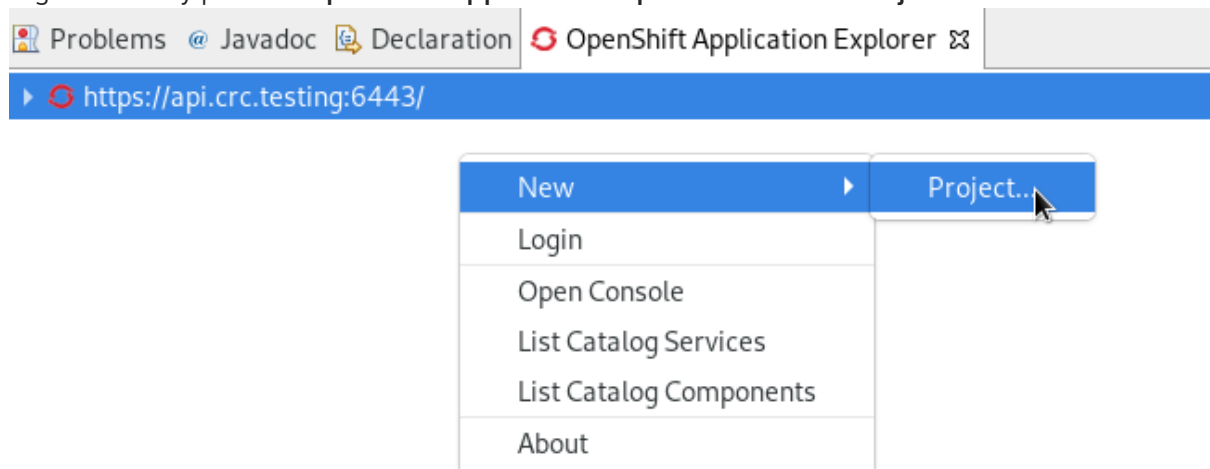
1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



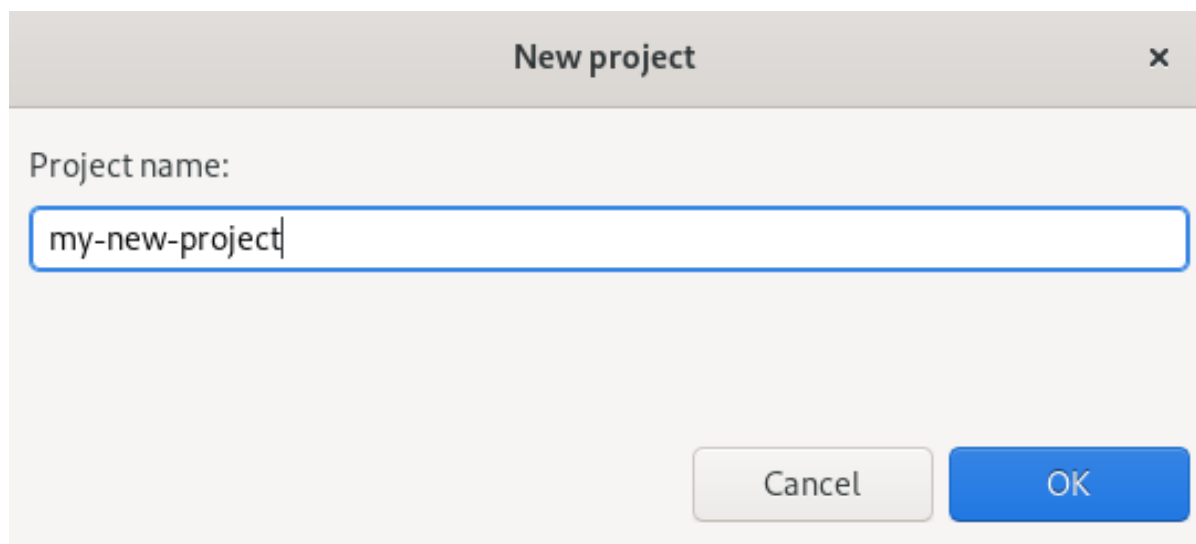
3. Select **General** → **Project**.
4. Click **Next**.  
The **New Project** window appears.



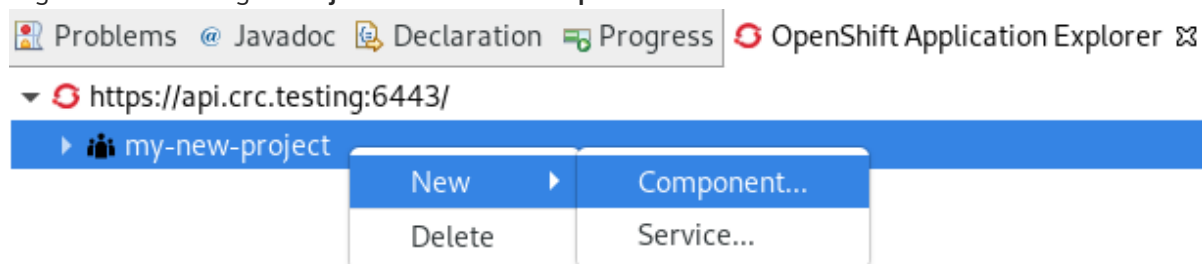
5. Name your project.
6. Select the location for your project.
7. Click **Finish**.  
Your newly created empty project is now listed in the **Package Explorer** view.
8. Start **OpenShift Application Explorer**.
9. Right-click any place in **OpenShift Application Explorer** → **New** → **Project**.



The **New project** window appears.



10. Name your project.
11. Click **OK**.  
Your newly created project is now listed in the **OpenShift Application Explorer** view.
12. Right-click the target **Project** → **New** → **Component**.




The **Create component** window appears.

● Create component

### Create component

Specify component parameters.



**OPENSIFT**

Name:

Eclipse Project:

i Your project is empty, you can initialize it from starters (templates)

Component type: 

- java-maven
- java-openliberty
- java-quarkus
- java-springboot
- java-vertx
- nodejs
- python
- python-django
- ▶ S2I

Component version:

Project starter:

Application:

Push after create:

?

13. Name your project.
14. Click **Browse** to select your **Eclipse Project**.
15. Set your **Component type** to **java-vertx**.
16. Set the **Project starter** to **java-vertx**.
17. Name your application.
18. Clear the **Push after create** check box.
19. Click **Finish**.

The **Console** view appears, displaying the validation process.

Your newly created component is now listed in the **OpenShift Application Explorer** view under your project.

Your application based on devfiles is built.

## 5.5.1. Managing your devfile registries

The following section describes how to create, delete, and edit devfile registries using OpenShift Application Explorer in CodeReady Studio.

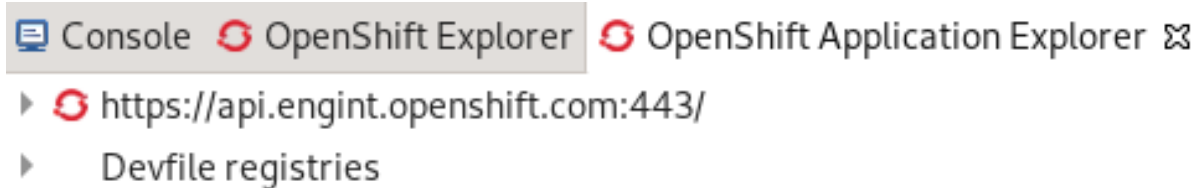
### 5.5.1.1. Adding a devfile registry

#### Prerequisites

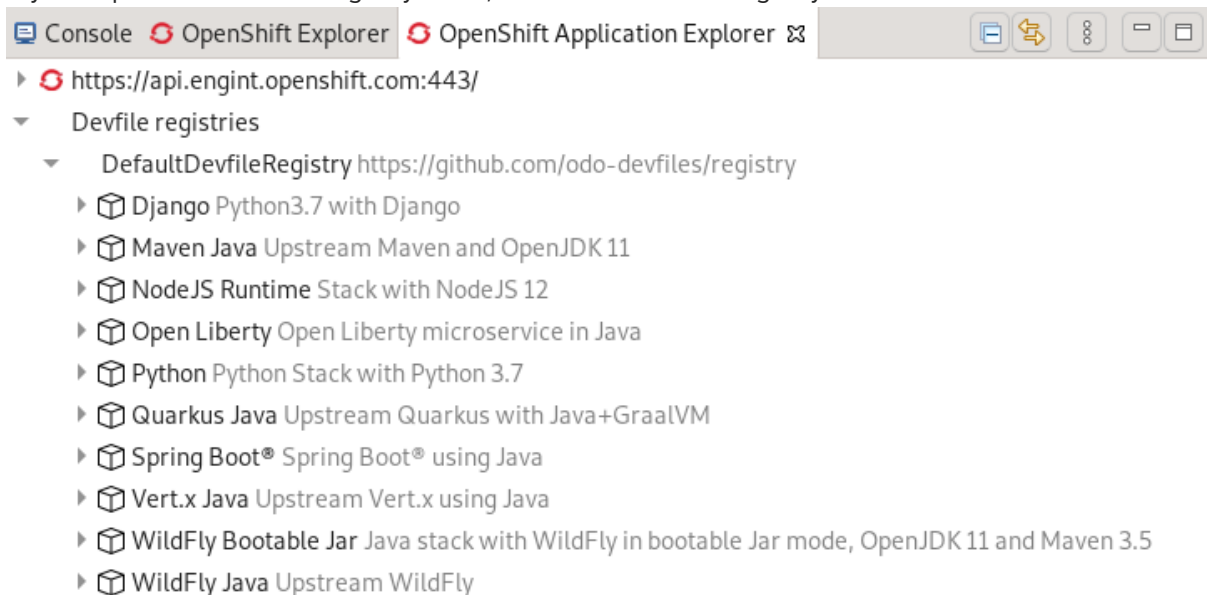
- A running OpenShift cluster.

#### Procedure

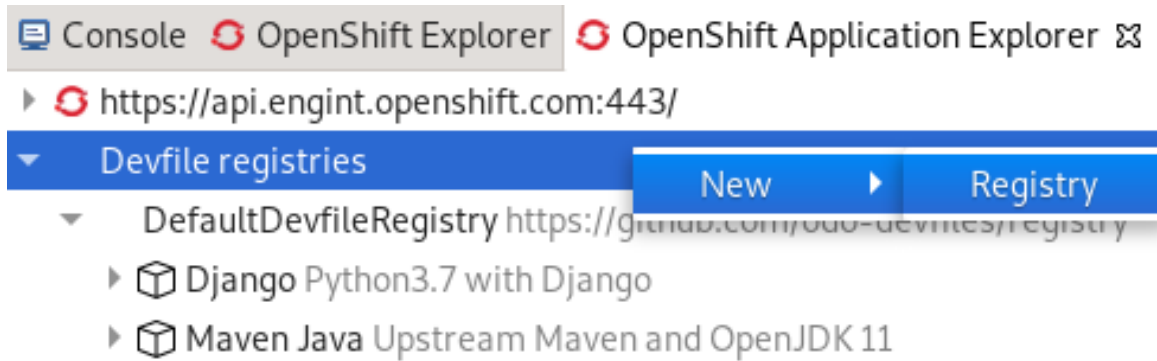
1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Devfile registries are displayed under the **Devfile registries** node.



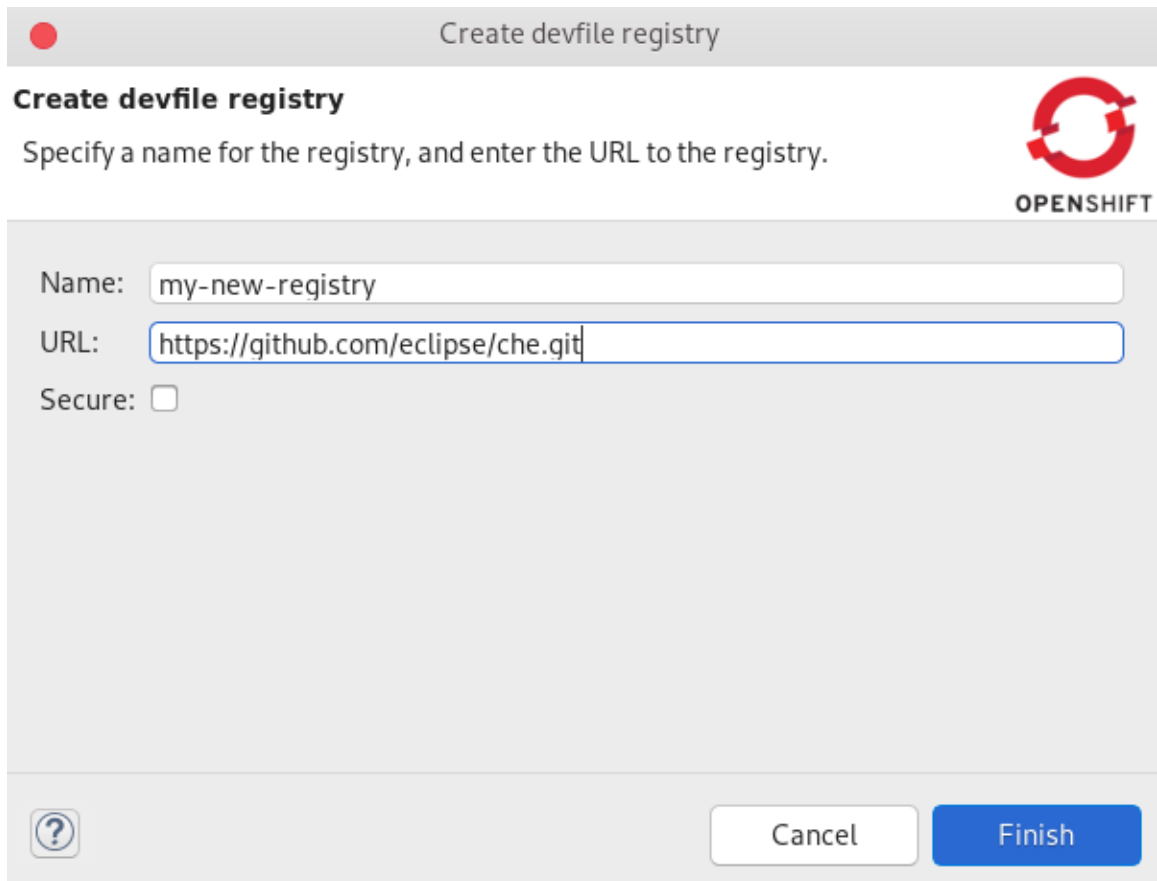
4. If you expand the devfile registry node, all devfiles of that registry are shown.



- To add a new devfile registry, right-click **Devfile registries** and click **new**.



The **Create devfile registry** window appears.



5. Name your devfile registry.
6. Paste your devfile URL.
7. Click **Finish**.

Your newly created devfile registry is now listed in the **OpenShift Application Explorer** view under **Devfile registries**.

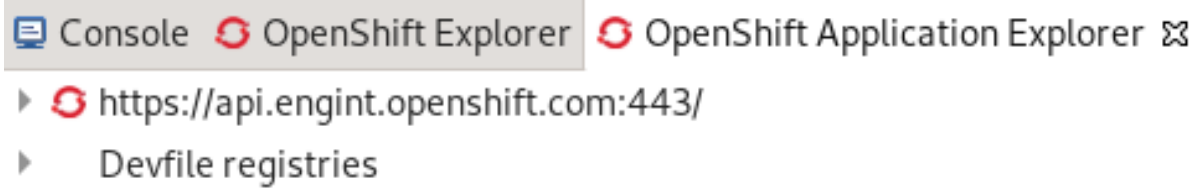
### 5.5.1.2. Deleting a devfile registry

#### Prerequisites

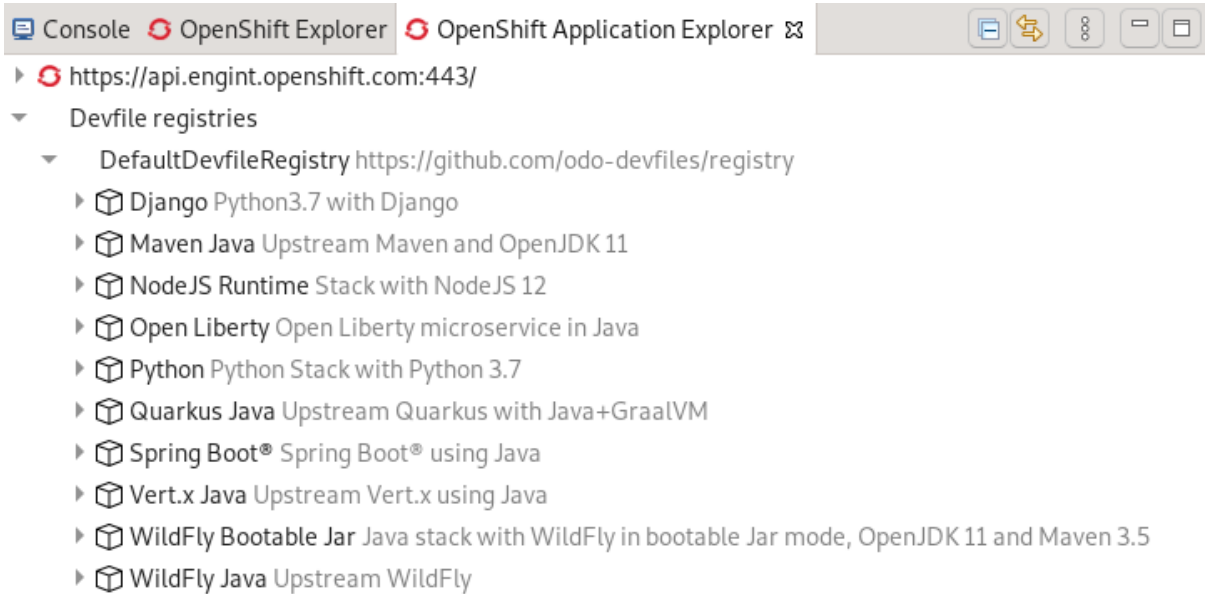
- A running OpenShift cluster.

#### Procedure

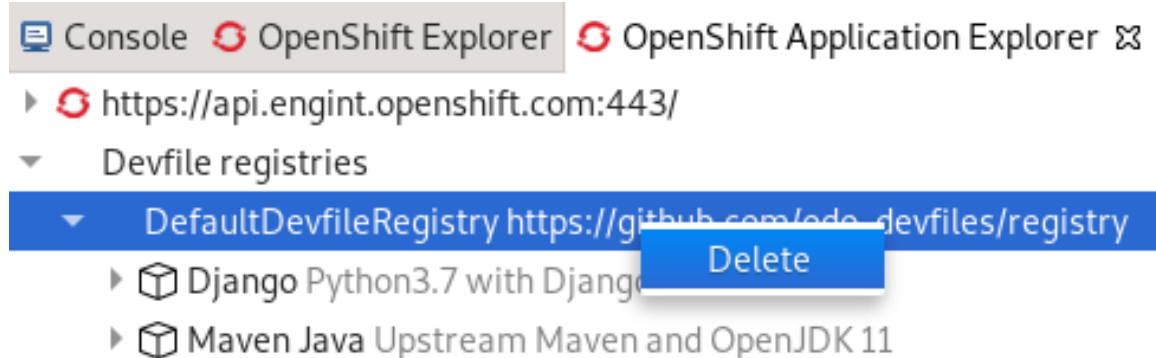
1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Devfile registries are displayed under the **Devfile registries** node.



4. An expanded devfile registry node shows all devfiles of that registry.



- To delete a devfile registry, right click the node of a devfile registry and click **delete**.



Your devfile registry is now deleted.

### 5.5.1.3. Editing a devfile registry

#### Prerequisites

- A running OpenShift cluster.

#### Procedure

- To edit a devfile registry, use the YAML editor. The YAML editor provides syntax validation and content assist.



### 5.5.1.4. Creating a component from your devfile registry

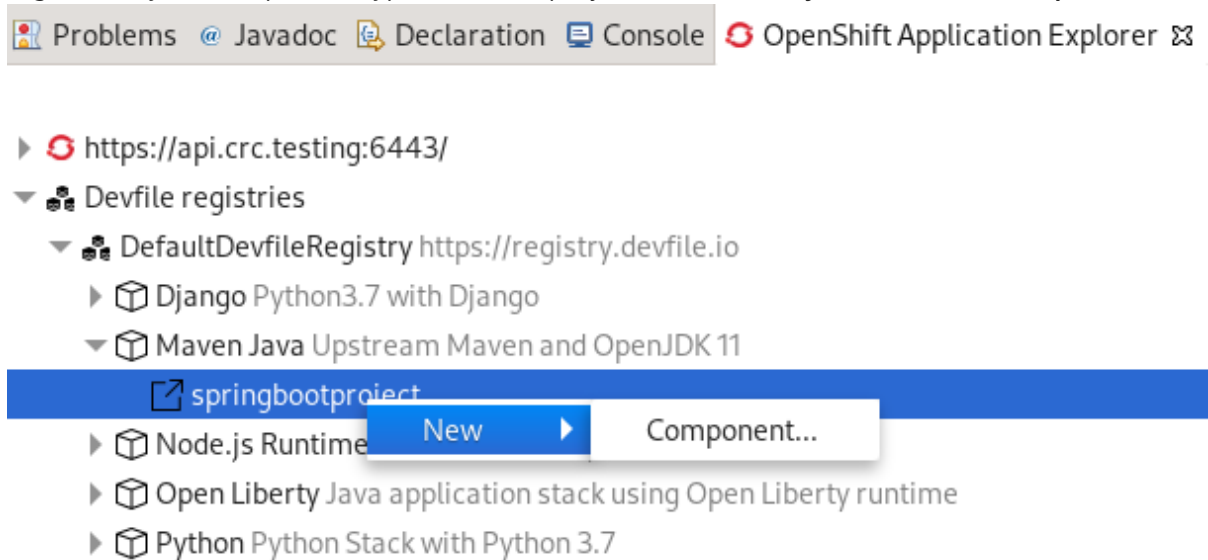
You can create a new component from your devfile registry from either a component type or a starter project.

#### Prerequisites

- An existing Eclipse project.
- An existing OpenShift project.
- You are connected to an OpenShift cluster.

#### Procedure

1. Expand your devfile registry under the **Devfile registry** node.
2. Right-click your component type or starter project and click **Project → New → Component**.




The **Create component** window appears.

● Create component

### Create component

Specify component parameters.



Name:

Eclipse Project:

i Your project is empty, you can initialize it from starters (templates)

Component type: 

java-wildfly (from DefaultDevfileRegistry)  
 java-wildfly-bootable-jar (from DefaultDevfileRegistry)  
 nodejs (from DefaultDevfileRegistry)  
 python (from DefaultDevfileRegistry)  
python-django (from DefaultDevfileRegistry)  
 ▶ S2I

Component version:

Project starter:

Application:

Push after create:

?

3. Name your component.
4. Click **Browse** to select your Eclipse project.
5. Choose your Component type.
6. Choose your project starter.
7. Name your application.
8. Click **Finish**.

The new component is created from your devfile registry.

### Additional resources

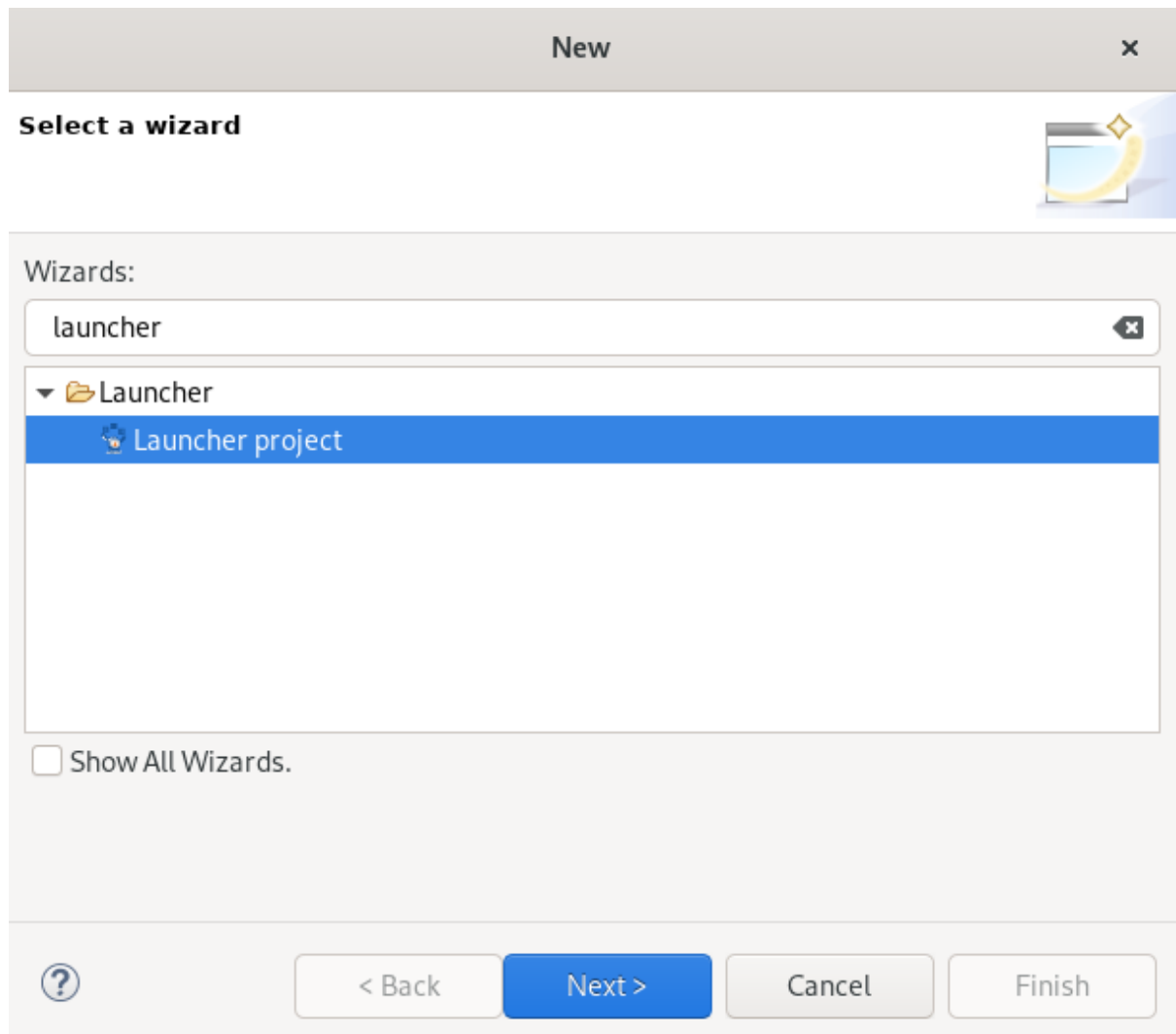
For further information on devfiles, visit [Introduction to Devfile](#).

## 5.6. BUILDING AN APPLICATION BASED ON S2I FILES

To deploy applications based on S2I files, you need a launcher project in your local workspace as well as an empty project in OpenShift, for which you need to create a component. After the component is established, your project will be updated and local and remote artifacts created in OpenShift.

### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.




3. Enter **Launcher** in the search field.
4. Select **Launcher project**
5. Click **Next**.  
The **New Launcher project** window appears.

New Launcher project ✕

**Generate a project based on mission and runtime.**

Generate an Eclipse project by specifying a mission and runtime variant.



Launcher will generate an application for you. By picking a mission you determine what this application will do. The runtime then picks the software stack that's used to implement this aim.

Mission:

Map business operations to a remote procedure call endpoint over HTTP using a REST framework

Runtime:

Runs a Node.js HTTP application

---

Project name:

Use default location

Location:

---

Maven Artifact:

Artifact id:

Group id:

Version:

?

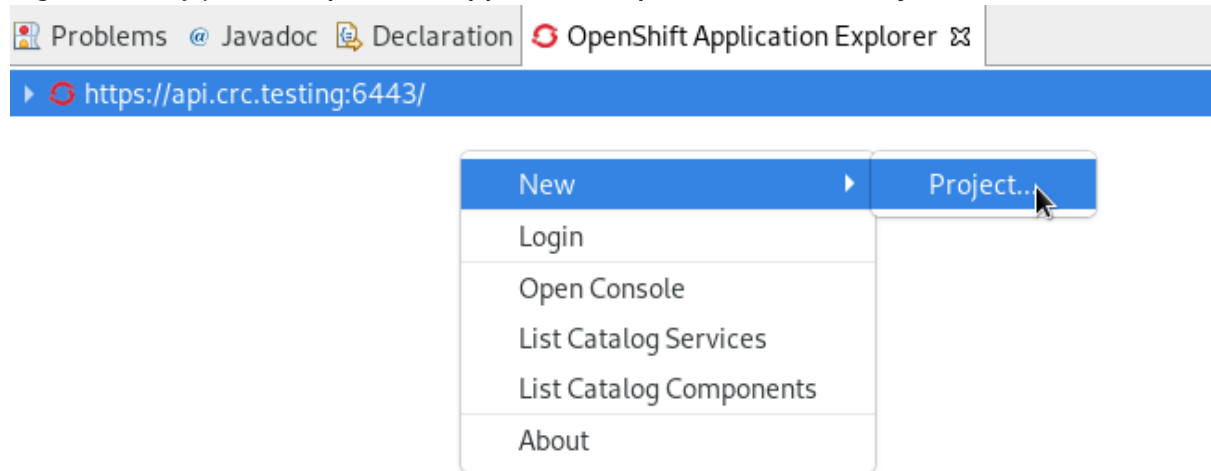
< Back
Next >
Cancel
Finish

6. Set **Mission** to **rest-http**.
7. Set **Runtime** to **vert.x community**.
8. Name your project.
9. Select the location for your project.
10. Click **Finish**.

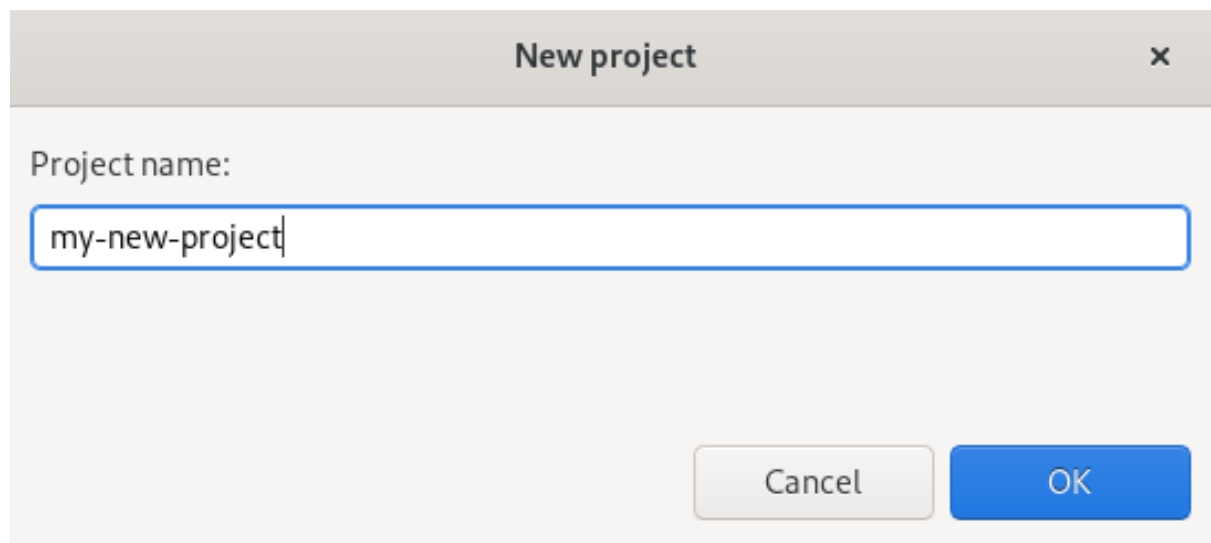
Note that the process of resolving dependencies might take some time to complete.

Your newly created launcher project is now listed in the **Project Explorer** view.

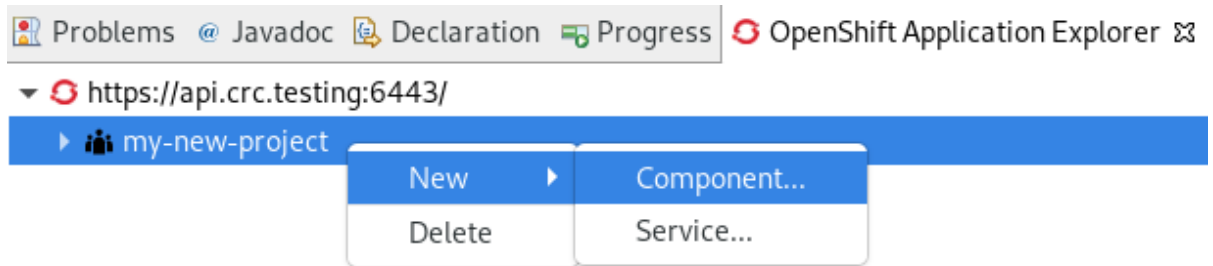
11. Start **OpenShift Application Explorer**.
12. Right-click any place in **OpenShift Application Explorer** → **New** → **Project**.



The **New project** window appears.



13. Name your project.
14. Click **OK**.  
Your newly created project is now listed in the **OpenShift Application Explorer** view.
15. Right-click the target **Project** → **New** → **Component**.




The **Create component** window appears.

● Create component

### Create component

Specify component parameters.



**OPENSIFT**

Name:

Eclipse Project:

Component type:

- nodejs
- python
- python-django
- ▼ S2I
  - dotnet
  - golang
  - httpd
  - java
  - nginx
  - nodejs
  - perl

Component version:  ▼

Project starter:  ▼

Application:

Push after create:

?

16. Name your project.
17. Click **Browse** to select your **Eclipse Project**.
18. Click on the arrow next to **S2I** and set your **Component type** to **java**.
19. Set the **Component version** to **latest**.
20. Name your application.
21. Clear the **Push after create** check box.

22. Click **Finish**.

The **Console** view appears, displaying the validation process.

Your newly created component is now listed in the **OpenShift Application Explorer** view under your project.

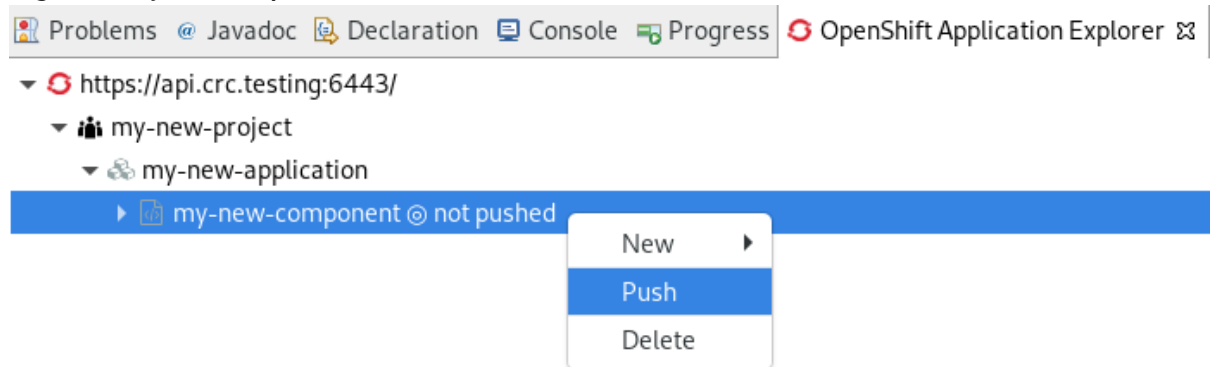
Your application based on S2I files is built.

## 5.7. DEPLOYING A COMPONENT ON A CLUSTER USING OPENSIFT APPLICATION EXPLORER

The following section describes how to deploy a component on a cluster using OpenShift Application Explorer in CodeReady Studio.

### Procedure

1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Expand your project.
4. Expand your application.
5. Right-click your **component** → **Push**.



The **Console** view appears, displaying the process of file synchronization.

## 5.8. DEFINING AN EXTERNAL ACCESS URL USING OPENSIFT APPLICATION EXPLORER

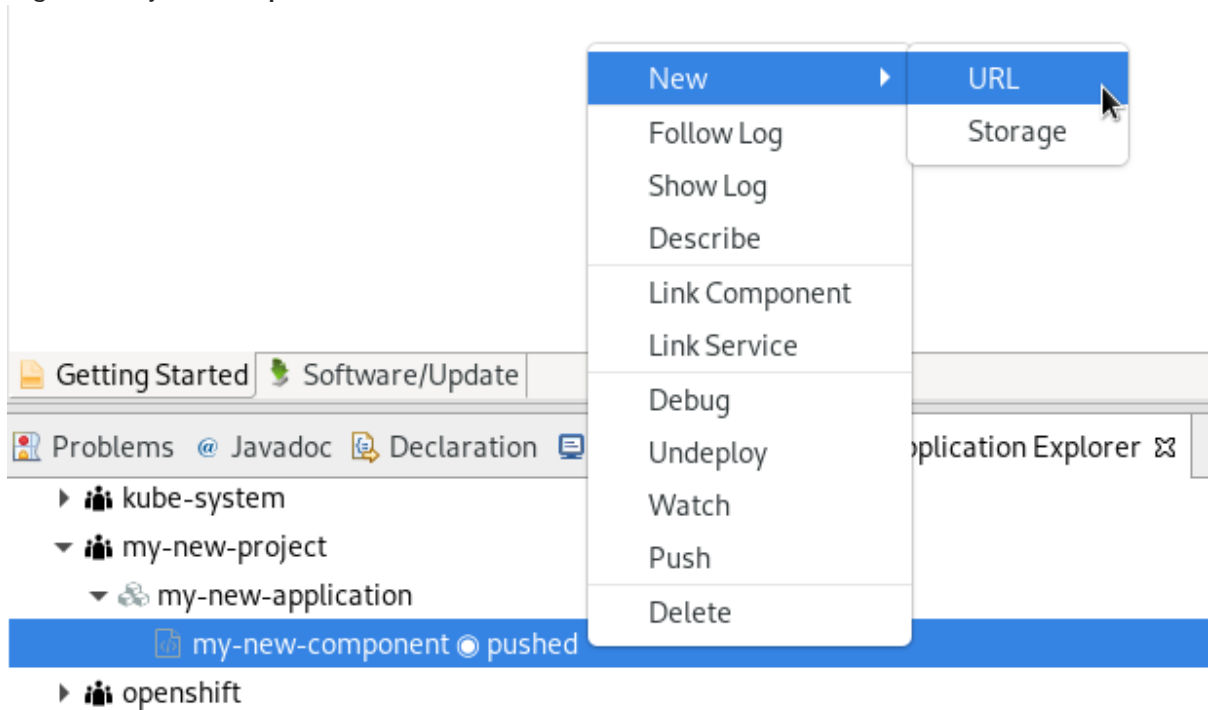
The following section describes how to define an external access URL using OpenShift Application Explorer in CodeReady Studio.

### Procedure

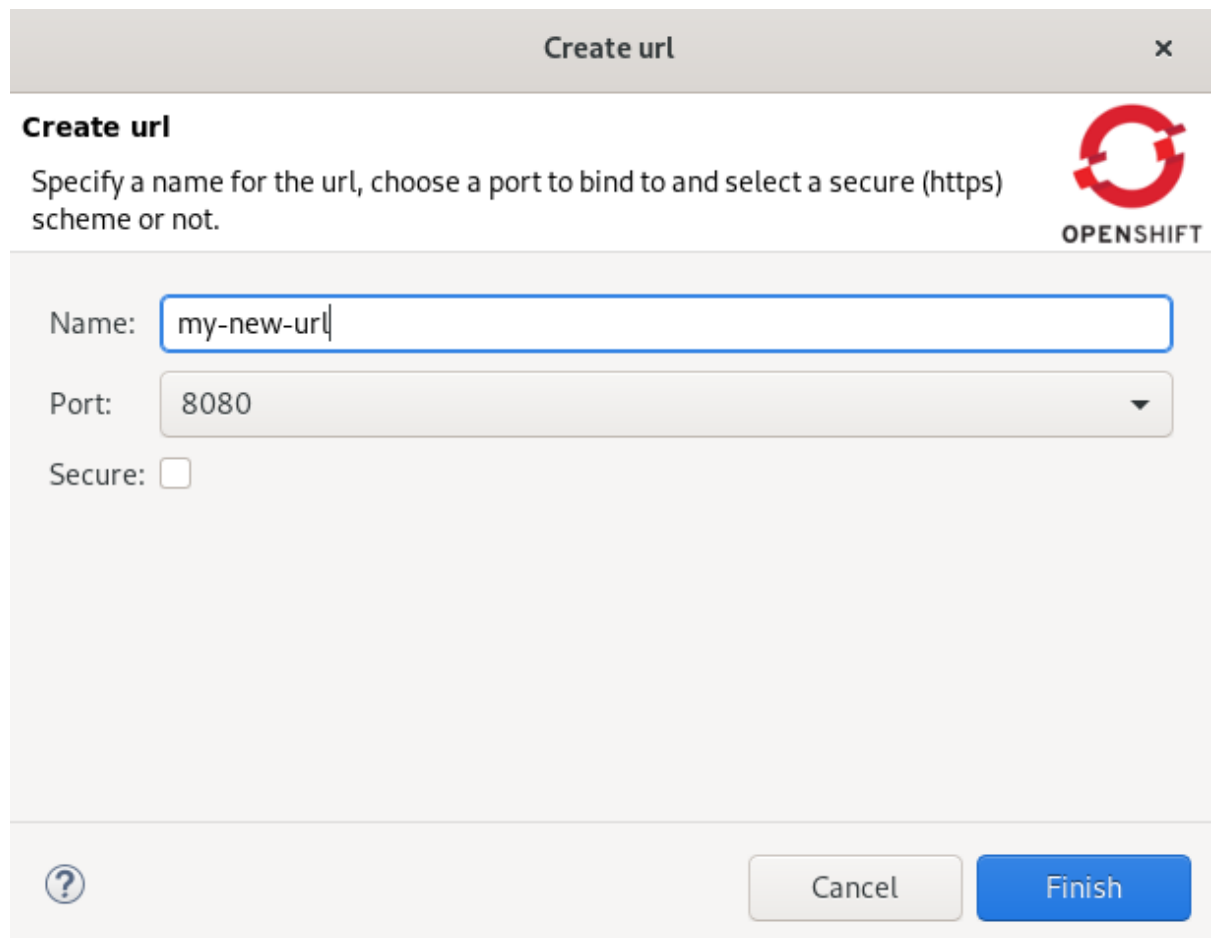
1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Expand your project.
4. Expand your application.



5. Right-click your **component** → **New** → **URL**.



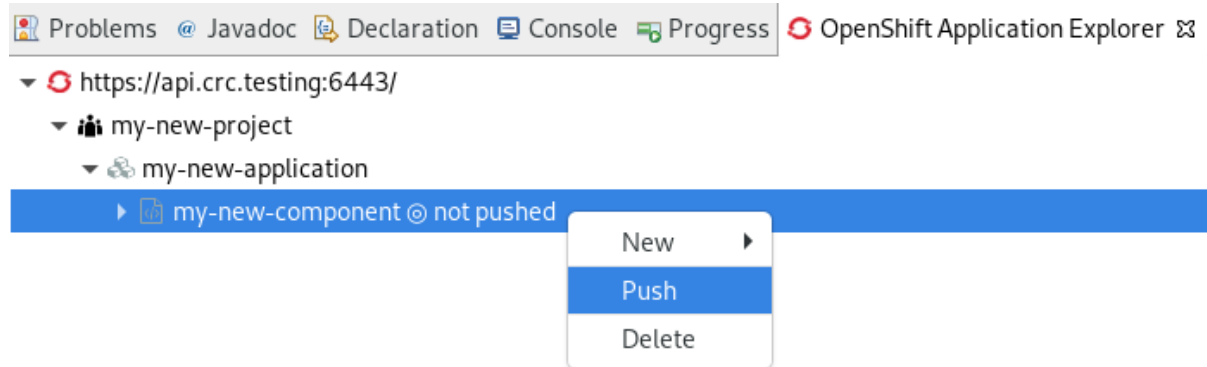
The **Create URL** window appears.



6. Name your URL.
7. Set the **Port** value to **8080**.
8. Click **Finish**.

The **Console** view appears, displaying the process of URL creation.

- In **OpenShift Application Explorer**, right-click your **component** → **Push**.



The **Console** view appears, displaying the process of file synchronization.

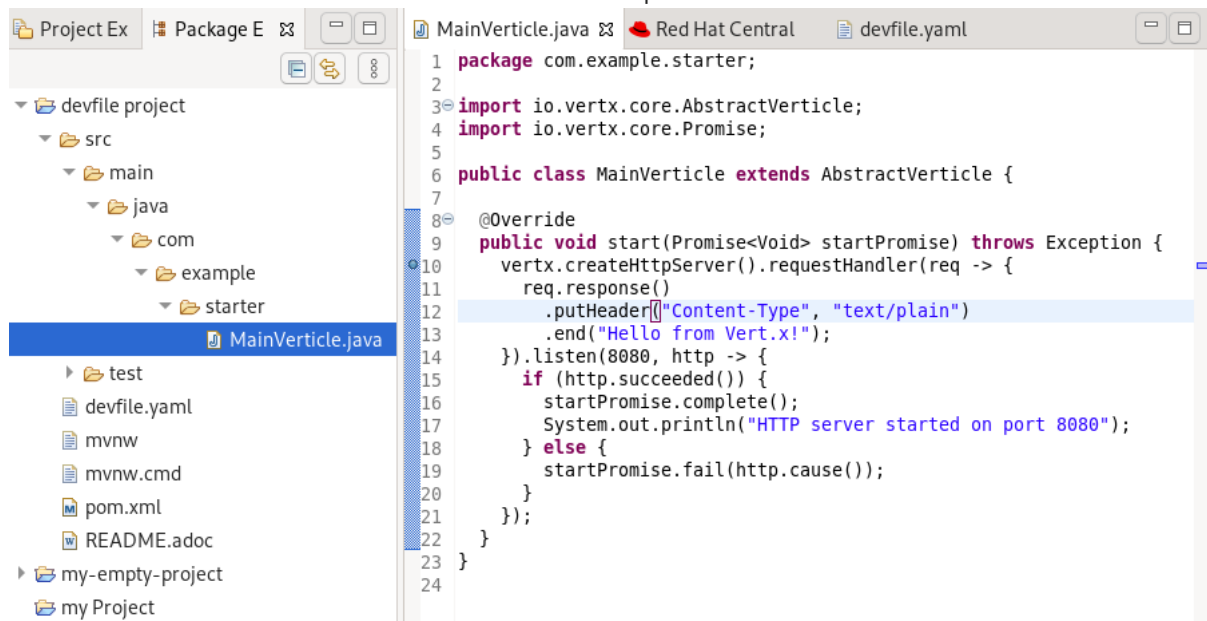
Your newly created URL is now listed in the **OpenShift Application Explorer** view under your component.

## 5.9. DEBUGGING AN APPLICATION ON A CLUSTER USING OPENSIFT APPLICATION EXPLORER

The following section describes how to debug an application on a cluster using OpenShift Application Explorer in CodeReady Studio.

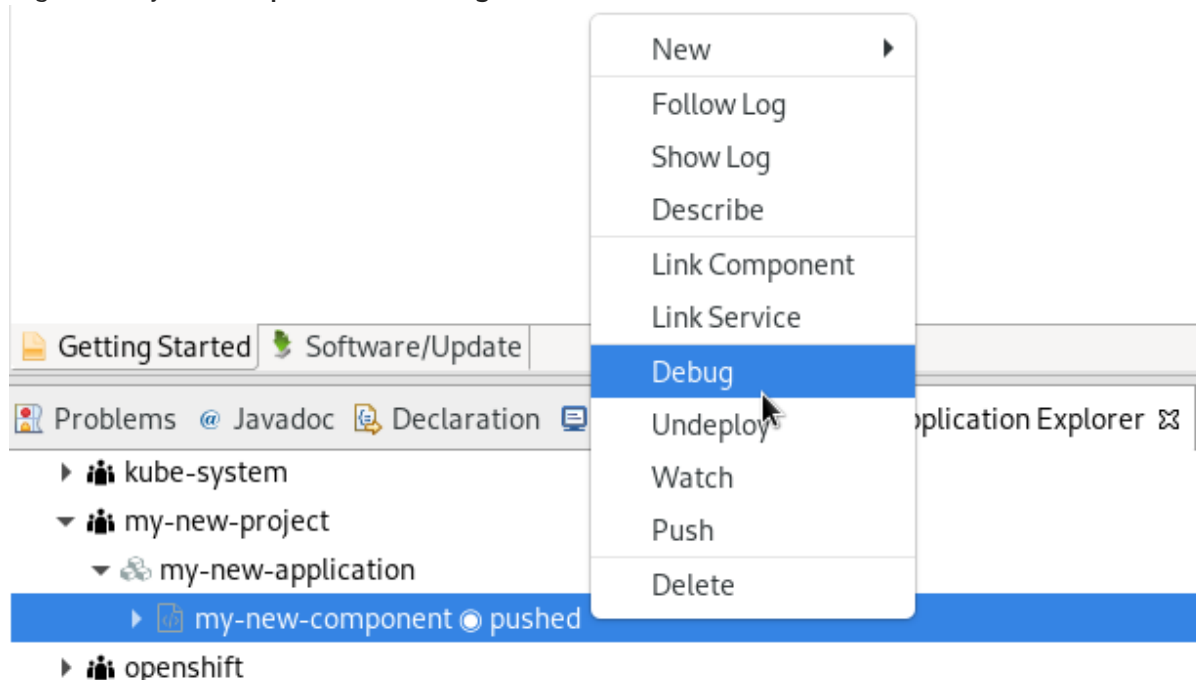
### Procedure

- Start CodeReady Studio.
- In the Project Explorer view, locate the **MainVerticle.java**(devfiles) or **HttpApplication.java**(S2I) file and double-click to open it.
- Double-click on the left ruler column to set a breakpoint.



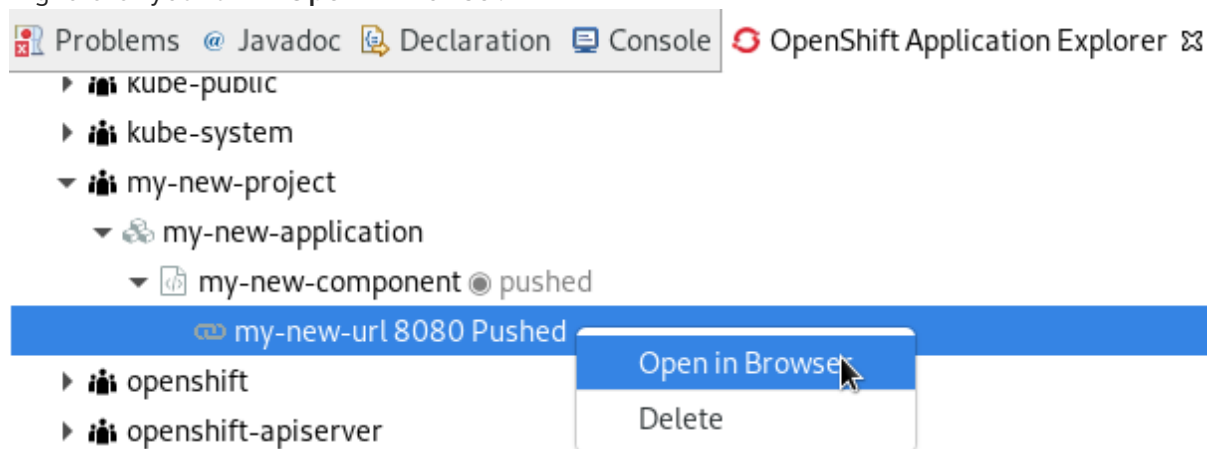
- Open **OpenShift Application Explorer**.
- Expand your project.

6. Expand your application.
7. Right-click your **component** → **Debug**.



The **Console** view appears.

8. In **OpenShift Application Explorer**, expand your component.
9. Right-click your **url** → **Open in Browser**.



The **Confirm Perspective Switch** window appears.

10. Click **Switch**.  
The **Debug Perspective** window appears displaying the debugging process.

## 5.10. CREATING APPLICATION SERVICES USING OPENSIFT APPLICATION EXPLORER

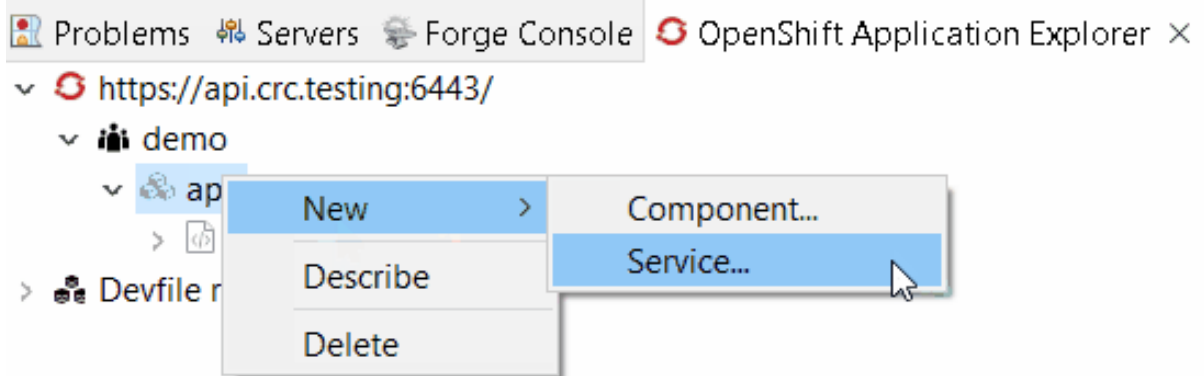
Use operators to create services when developing cloud native applications on OpenShift.

### Prerequisites

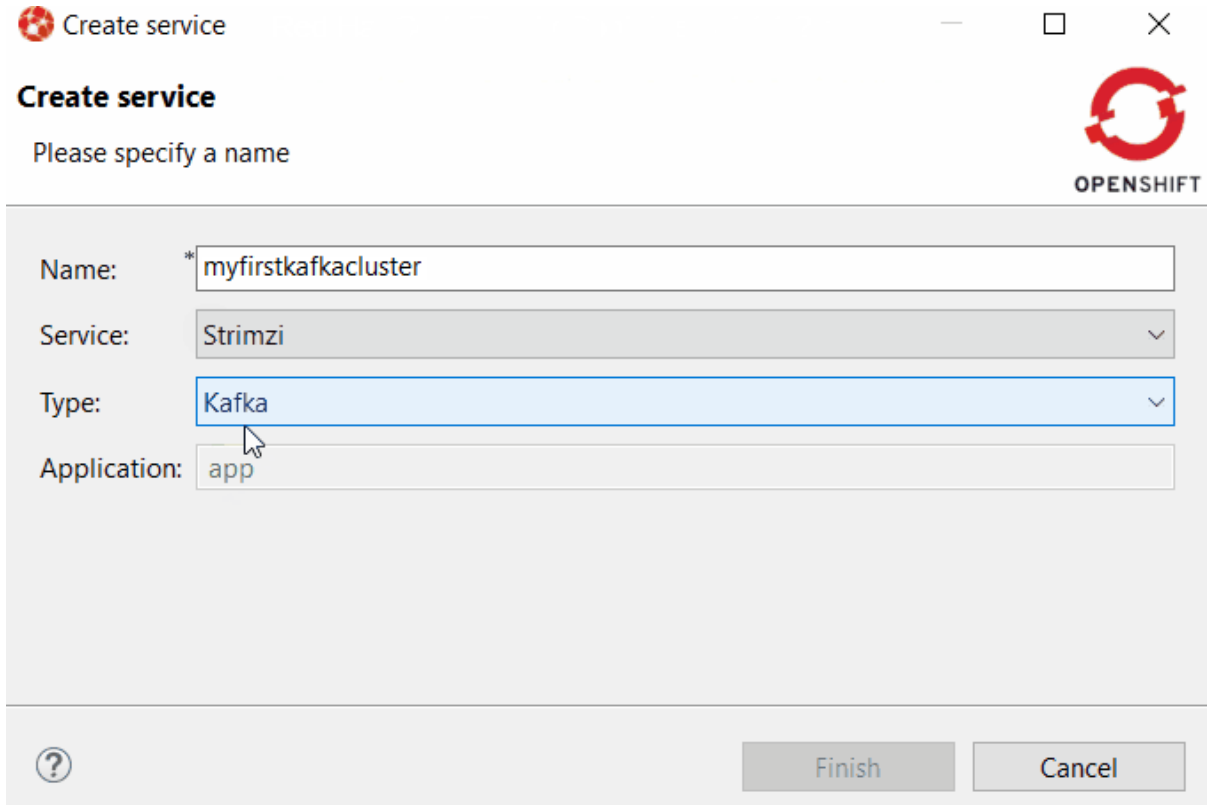
- An installed operator on your cluster.  
For more information on installing operators, see [Adding Operators to a cluster](#).

## Procedure

1. In the OpenShift Application Explorer view, right-click your application.

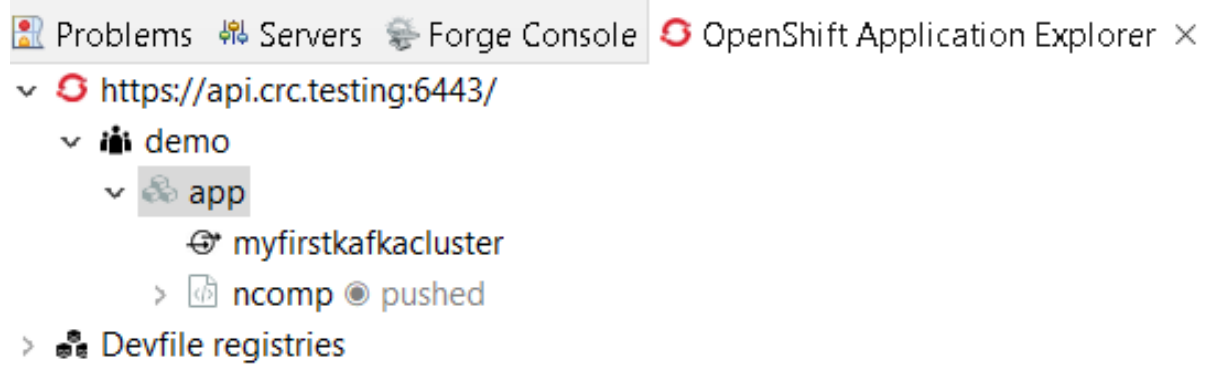


2. Select **New** → **Service**.



The **Create service** view appears.

3. Enter a name for your service.
4. From the **Service** drop-down menu, select a service from the installed operators on your cluster.
5. From the **Type** drop-down menu, select a type of deployment.
6. Click **Finish**.



Your newly created service is displayed in the OpenShift Application Explorer view.

## CHAPTER 6. QUARKUS TOOLS BASICS IN CODEREADY STUDIO

Quarkus is a Kubernetes-Native full-stack Java framework aimed to optimize work with Java virtual machines. Quarkus provides tools for Quarkus application developers, helping to reduce the size of Java applications and container image footprints, as well as the amount of memory required.

### Prerequisites

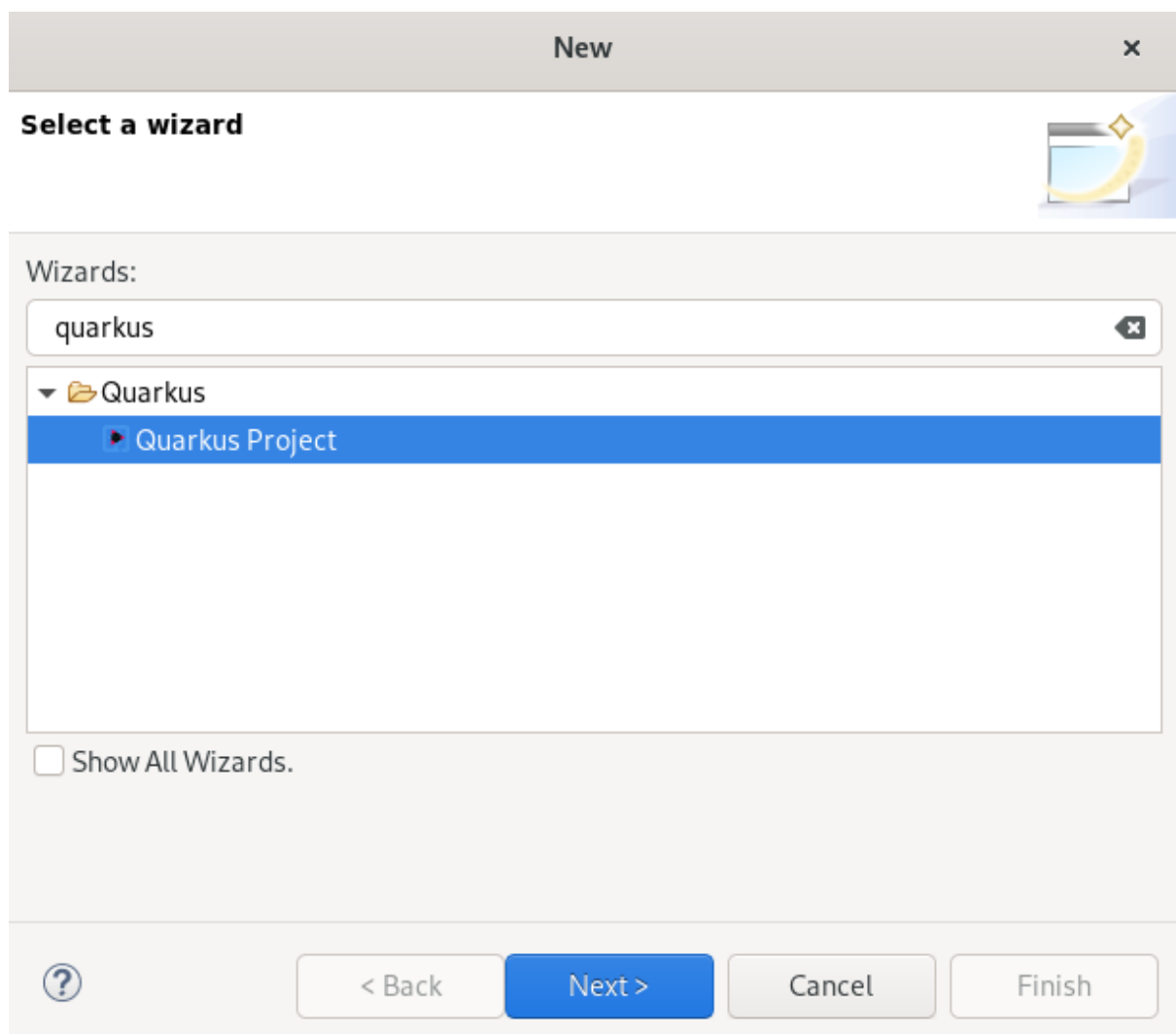
- The latest version of JBoss Tools is installed. For more information, see [JBoss Tools Downloads](#).

### 6.1. CREATING A NEW QUARKUS PROJECT

The following section describes how to create a new Quarkus project in CodeReady Studio.


#### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



3. Enter **Quarkus** in the search field.
4. Select **Quarkus Project**.
5. Click **Next**.  
The **New Quarkus project** window appears.

**New Quarkus project** x

**Project type** 

Select the code.quarkus.io endpoint and project type


code.quarkus.io will generate an application for you. Select the project type according to your favorite build tool. Then select the Quarkus dependencies you plan to use in your application.

Project type:

Project name:

Use default location

Location:



6. Select the needed project type.
7. Name your project.
8. Select the location for your project.
9. Click **Next**.  
The **Project type** window appears.

New Quarkus project x

### Project type

Select the code.quarkus.io endpoint and project type

Maven Artifact:

Artifact id:

Group id:

Version:

REST:

Class name:

Path:

?
< Back
Next >
Cancel
Finish

10. Ensure that the default values are correct.

11. Click **Next**.

The **Quarkus extensions** window appears.

New Quarkus project x

### Quarkus extensions

Select the Quarkus extensions for your project

Clicking on a category will display the extensions in the middle column. Double clicking on an extension will add/remove the extension from the selected extensions list. The current selected extensions are displayed in the third column.

Categories	Extensions	Selected
Web	RESTEasy JAX-RS (Included)	RESTEasy JAX-RS (Included)
Data	RESTEasy JSON-B	RESTEasy Qute (Experimental)
Messaging	RESTEasy Jackson	
Core	Hibernate Validator	
Reactive	REST Client	
Cloud	REST Client JAXB	
Observability	REST Client JSON-B	
Security	REST Client Jackson	
Integration	REST resources for Hibernate ORM with Panache (Experimental)	
Business Automation	RESTEasy JAXB	
Serialization	RESTEasy Mutiny (Preview)	
Miscellaneous	RESTEasy Qute (Experimental)	
Compatibility	Reactive Routes	
Alternative languages	SmallRye GraphQL (Preview)	
	SmallRye JWT	
	SmallRye OpenAPI	
	Undertow Servlet	
	Undertow WebSockets	
	gRPC (Experimental)	

Quute Templating integration for RESTEasy. [Click to open guide](#)

?
< Back
Next >
Cancel
Finish



12. Select the needed **Categories** for your projects.  
The available extensions of the selected category are displayed in the **Extensions** column.
13. Select the needed **Extensions** for your projects.  
Double-click on the extension to select or deselect it. The selected extensions appear in the **Selected** column.
14. Click **Finish**.

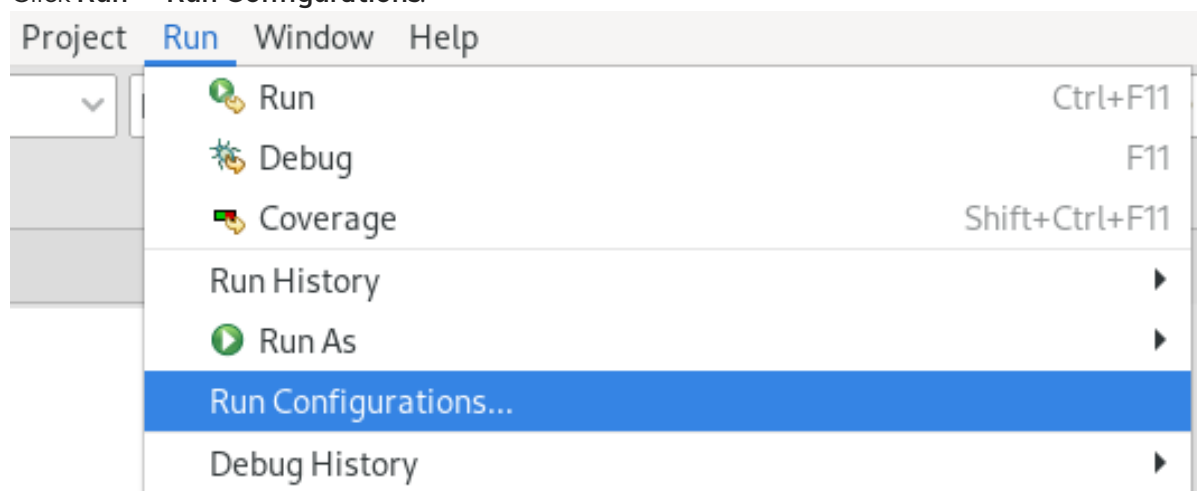
Your newly created Quarkus project is now listed in the **Project Explorer** view.

## 6.2. RUNNING A QUARKUS APPLICATION

The following section describes how to run a Quarkus application in CodeReady Studio.

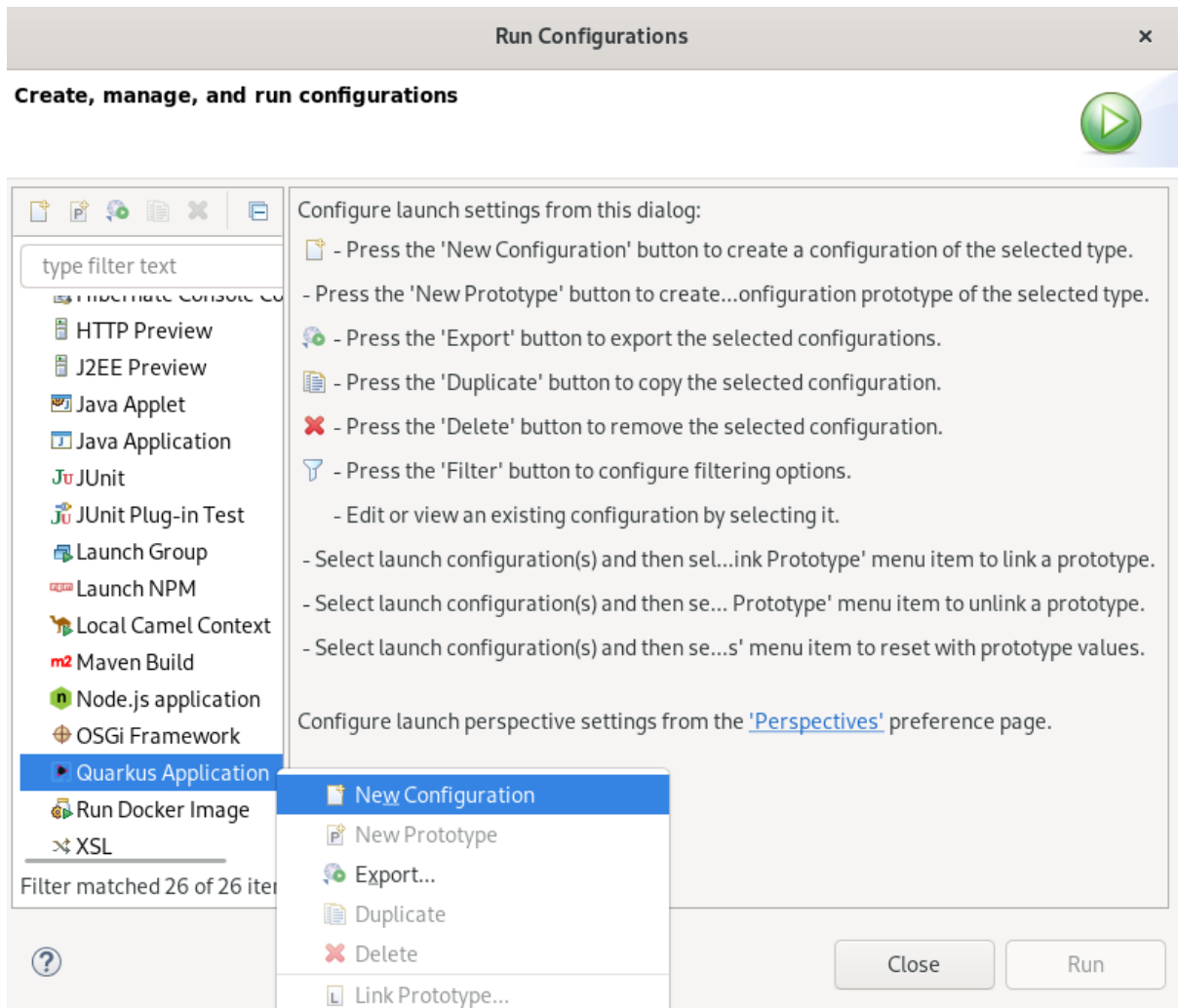
### Procedure

1. Start CodeReady Studio.
2. Click **Run** → **Run Configurations**.

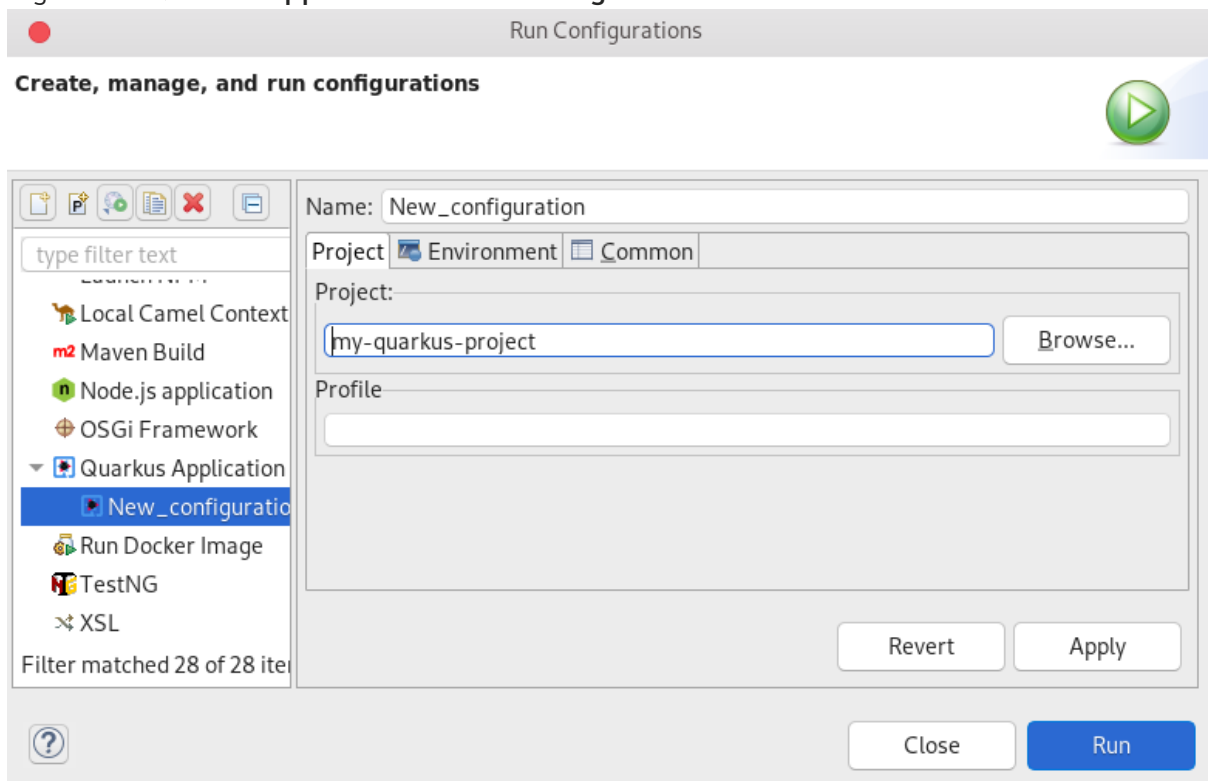


The **Run Configurations** window appears.

3. Scroll down to **Quarkus Application**.



4. Right-click **Quarkus Application** → **New Configuration**.



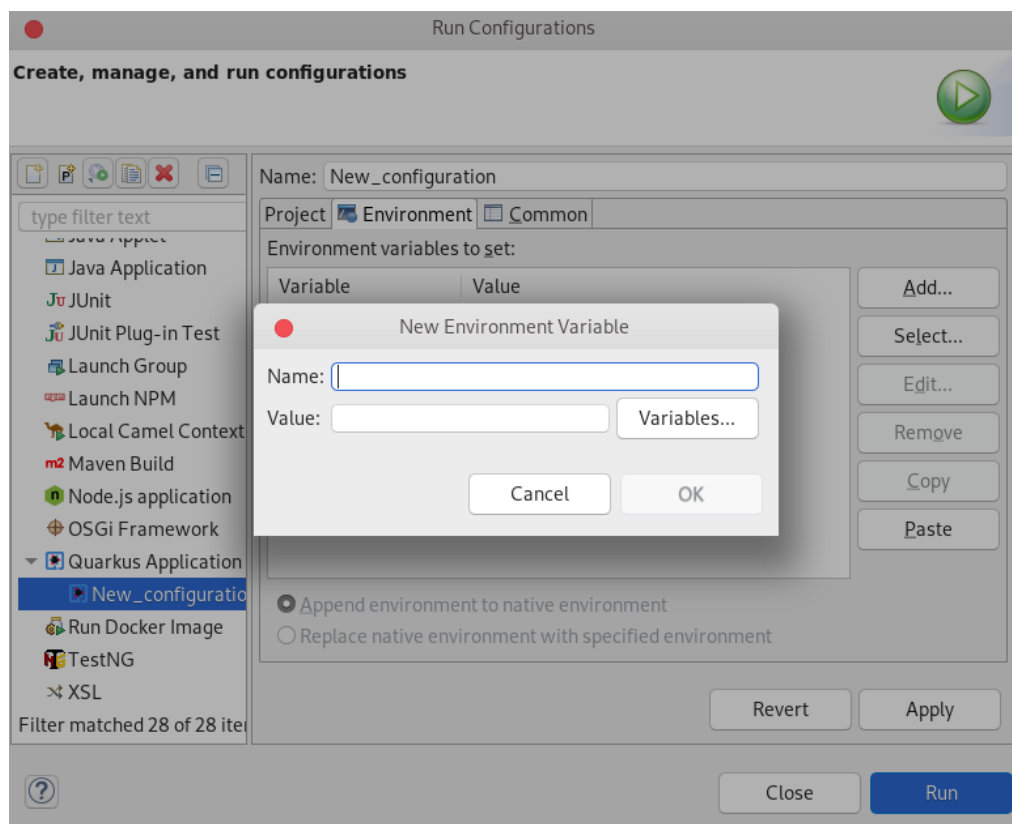
5. Name your configuration.

- Click **Browse** to locate your project.



## NOTE

It is possible to add environment variables to your Quarkus project. To add a new environment variable, click **Environment** → **add** and select a name and value.



- Click **Apply**.
- Click **Run**.  
The **Console** view appears.

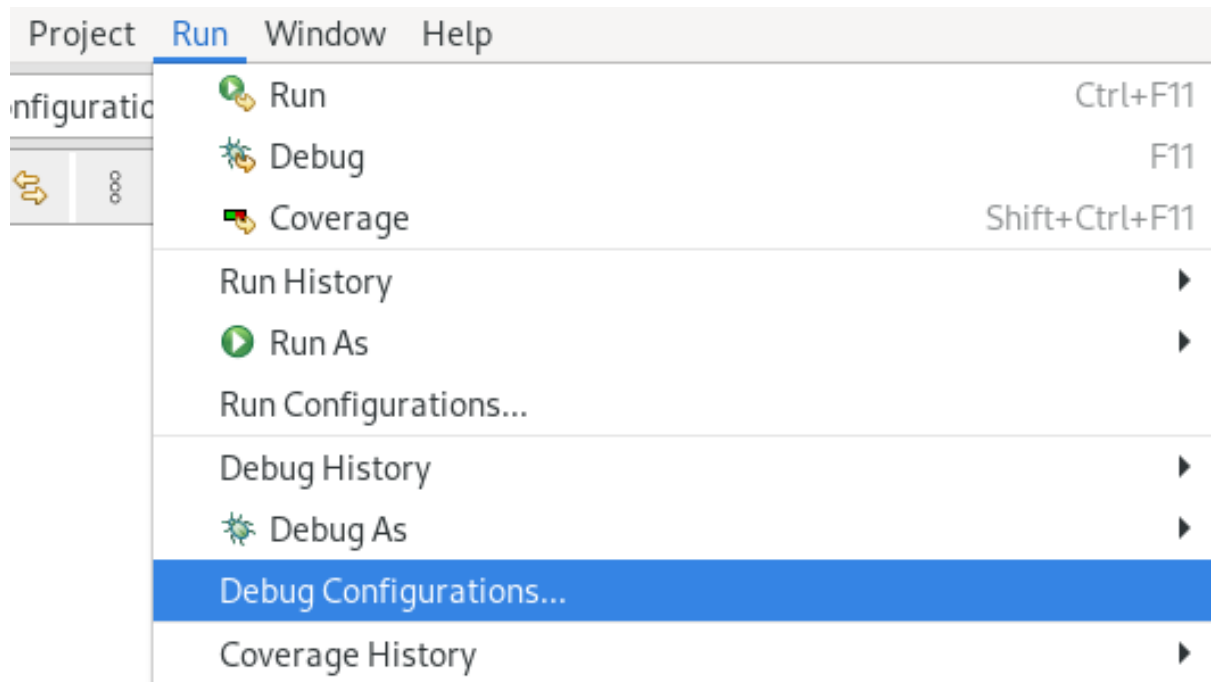
Your application will start after the built process.

## 6.3. DEBUGGING A QUARKUS APPLICATION

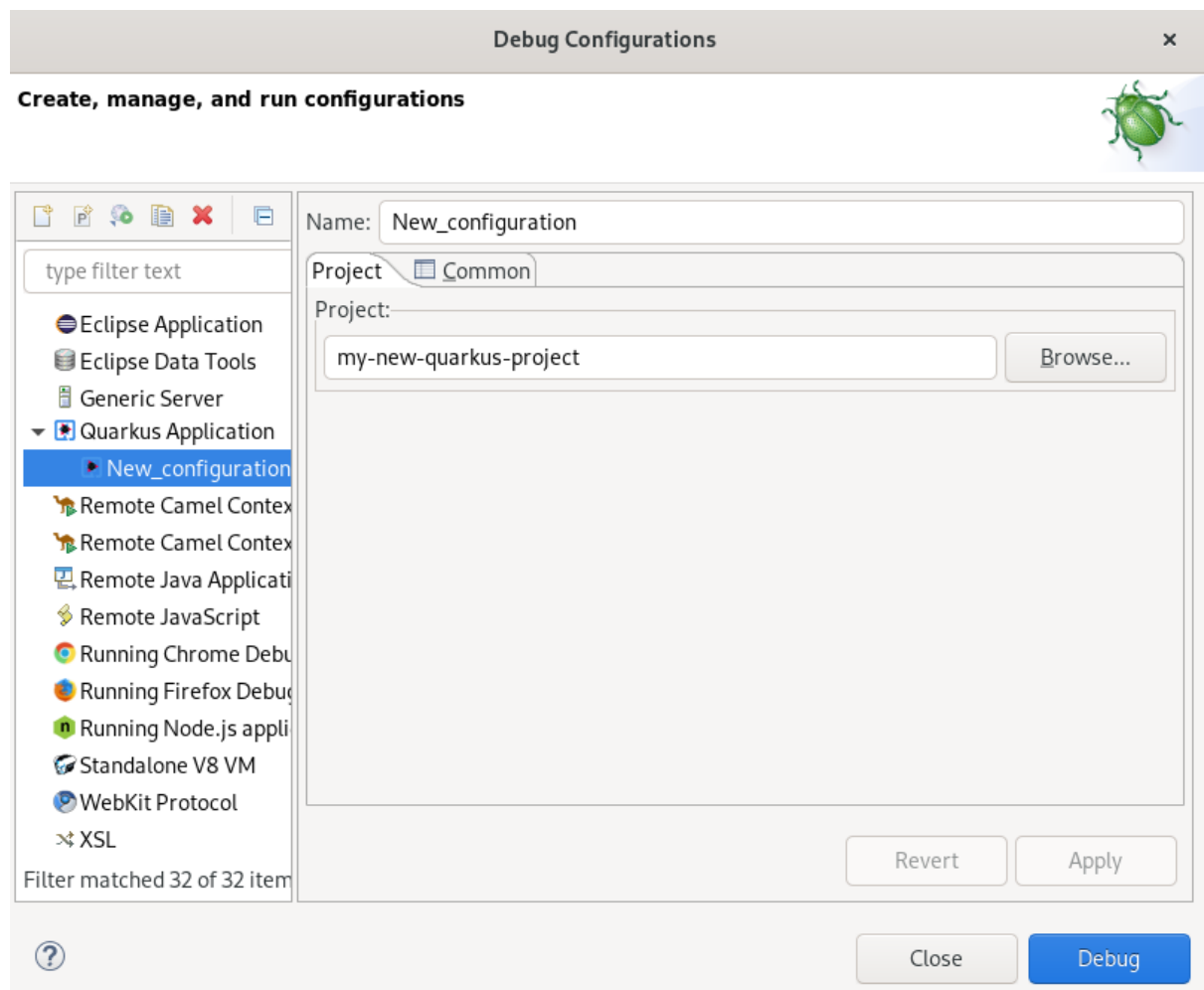
The following section describes how to debug a Quarkus application in CodeReady Studio.

### Procedure

- Start CodeReady Studio.
- Click **Run** → **Debug Configurations**.



The **Debug Configurations** window appears.



3. Expand **Quarkus Application**.
4. Select your configuration.
5. Click **Debug**.

The **Console** view appears.

Your Quarkus application starts and connects to a remote JVM debug configuration. If you set breakpoints in your application source files, the execution automatically stops after reaching the breakpoint.

## 6.4. USING LANGUAGE SUPPORT IN CODEREADY STUDIO

Every Quarkus application is configured through an **application.properties** configuration file. The content of this configuration file is dependent on the set of Quarkus extensions that your application is using.

Quarkus Tools includes content assist, which provides code completion, validation, and documentation. Code completion allows you to quickly complete statements in your code. Multiple choices are available to you via popups. This language support is now available for Kubernetes, OpenShift, S2i, Docker properties, MicroProfile REST Client properties, and MicroProfile Health artifacts. Note that language support for MicroProfile REST Client properties needs to be enabled separately. For more information, see [Section 6.4.2, "Enabling language support for MicroProfile REST Client properties"](#).

### 6.4.1. Using Quarkus content assist

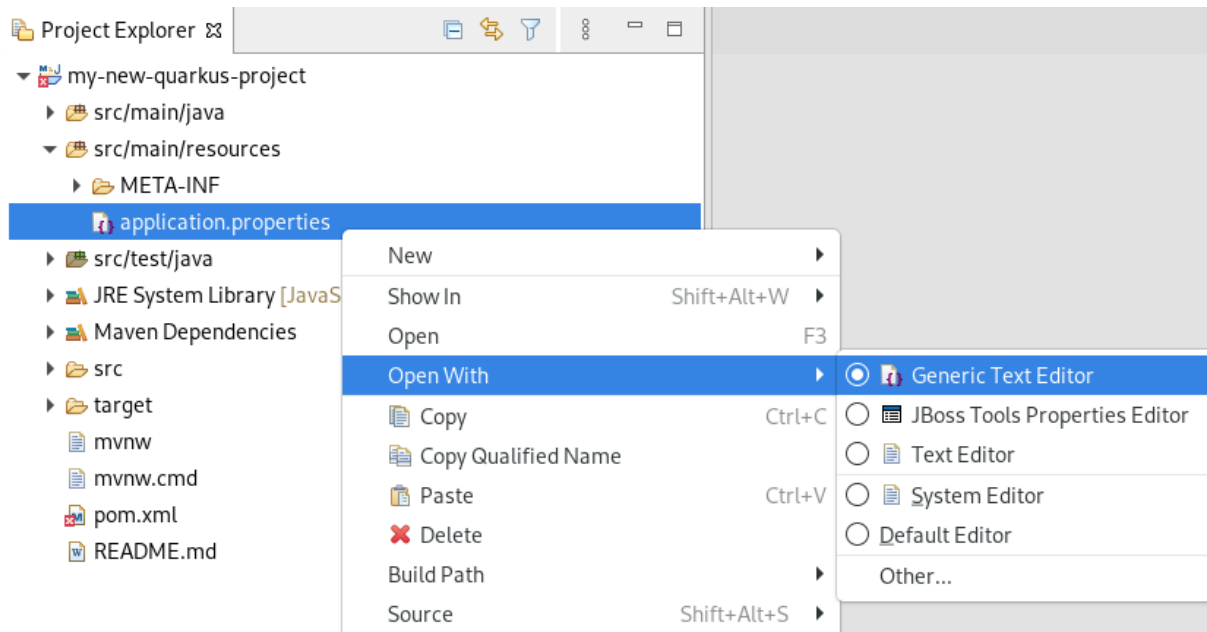
The following section describes how to use Quarkus **application.properties** content assist in CodeReady Studio.

#### Prerequisites

- An existing Quarkus project.  
For more information on how to create a Quarkus project, see [Section 6.1, "Creating a new Quarkus project"](#).

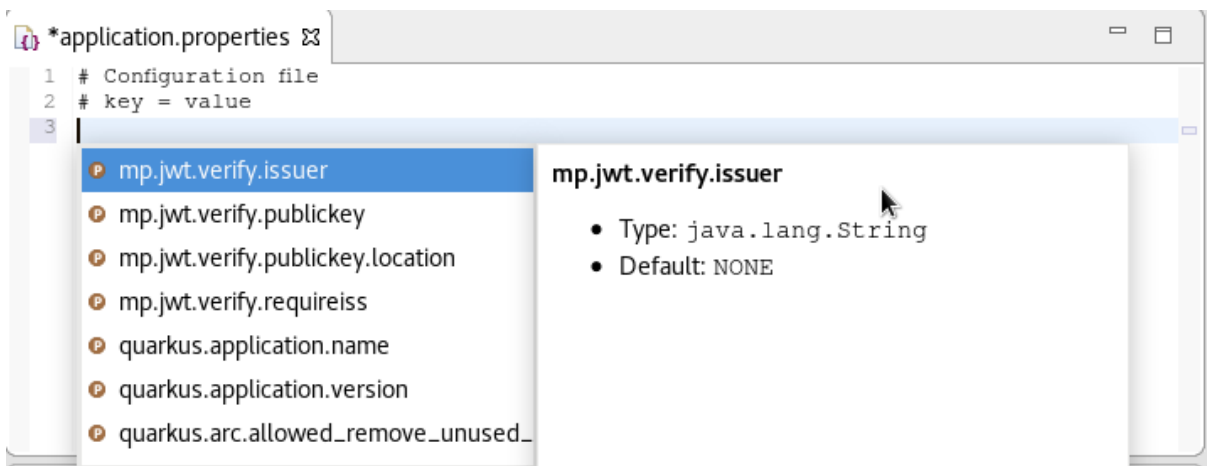
#### Procedure

1. Start CodeReady Studio.
2. Start **Project Explorer**.
3. Expand your **Quarkus project** → **src/main/resources**.
4. Right-click **application.properties** → **Open With** → **Generic Text Editor**.

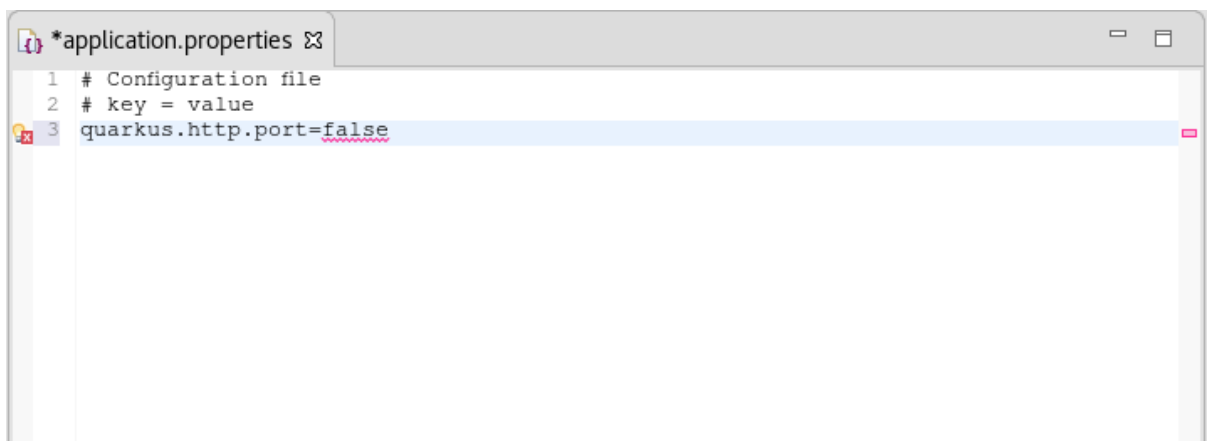


The **Generic Text Editor** window appears.

5. Navigate to an empty line.
6. Press **Ctrl+Space** to invoke code completion. The code completion suggestions appear. Hover the mouse over the suggestions to display documentation.



If you enter a wrong value, the editor underlines the error with a red wavy line.



### Additional resources

- Language support for MicroProfile REST Client properties needs to be enabled separately. For more information, see [Section 6.4.2, “Enabling language support for MicroProfile REST Client properties”](#).

## 6.4.2. Enabling language support for MicroProfile REST Client properties

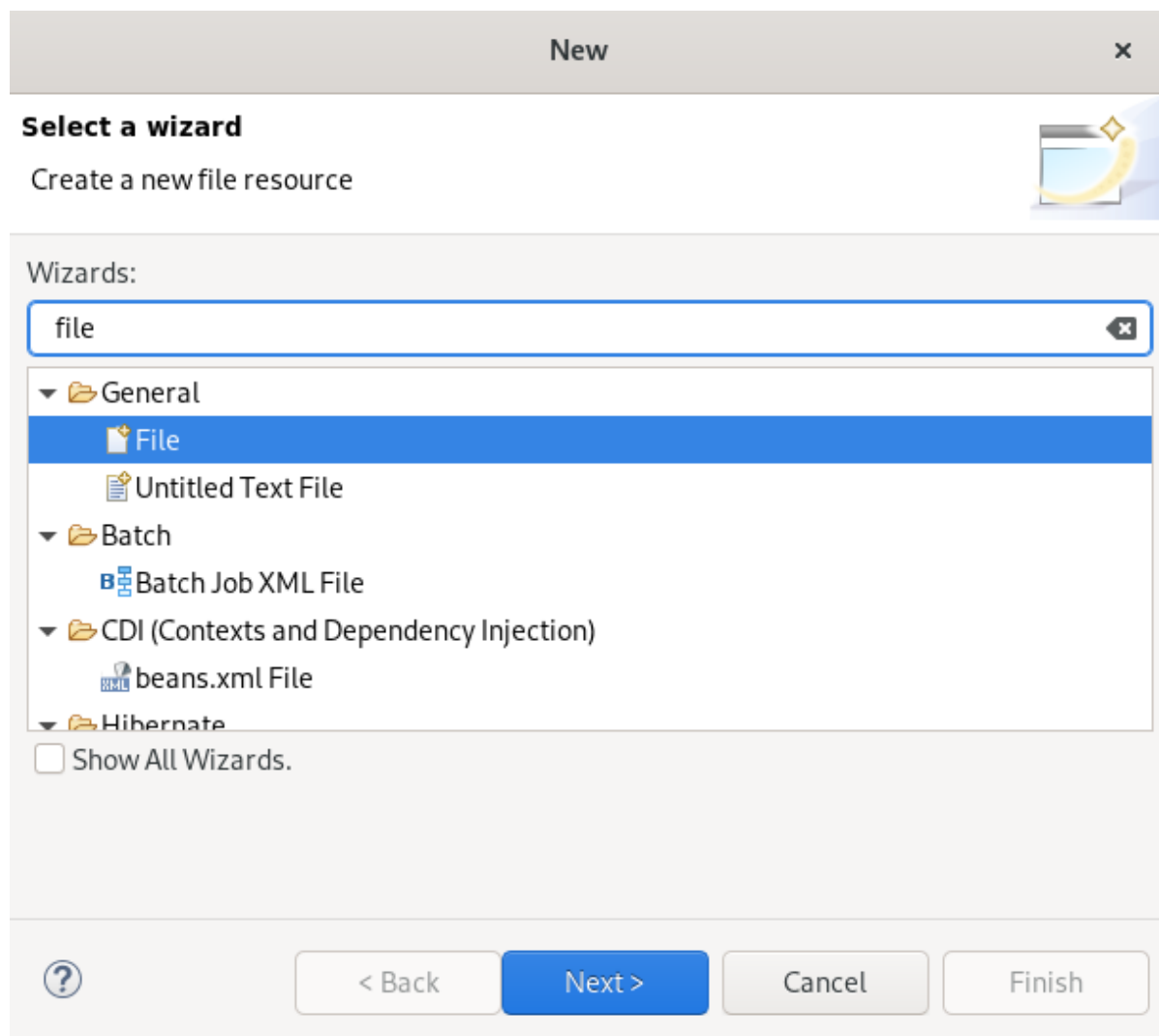
The following section describes how to enable language support for MicroProfile REST Client properties.

### Prerequisites

- An existing Quarkus project.  
For more information on how to create a Quarkus project, see [Section 6.1, “Creating a new Quarkus project”](#).

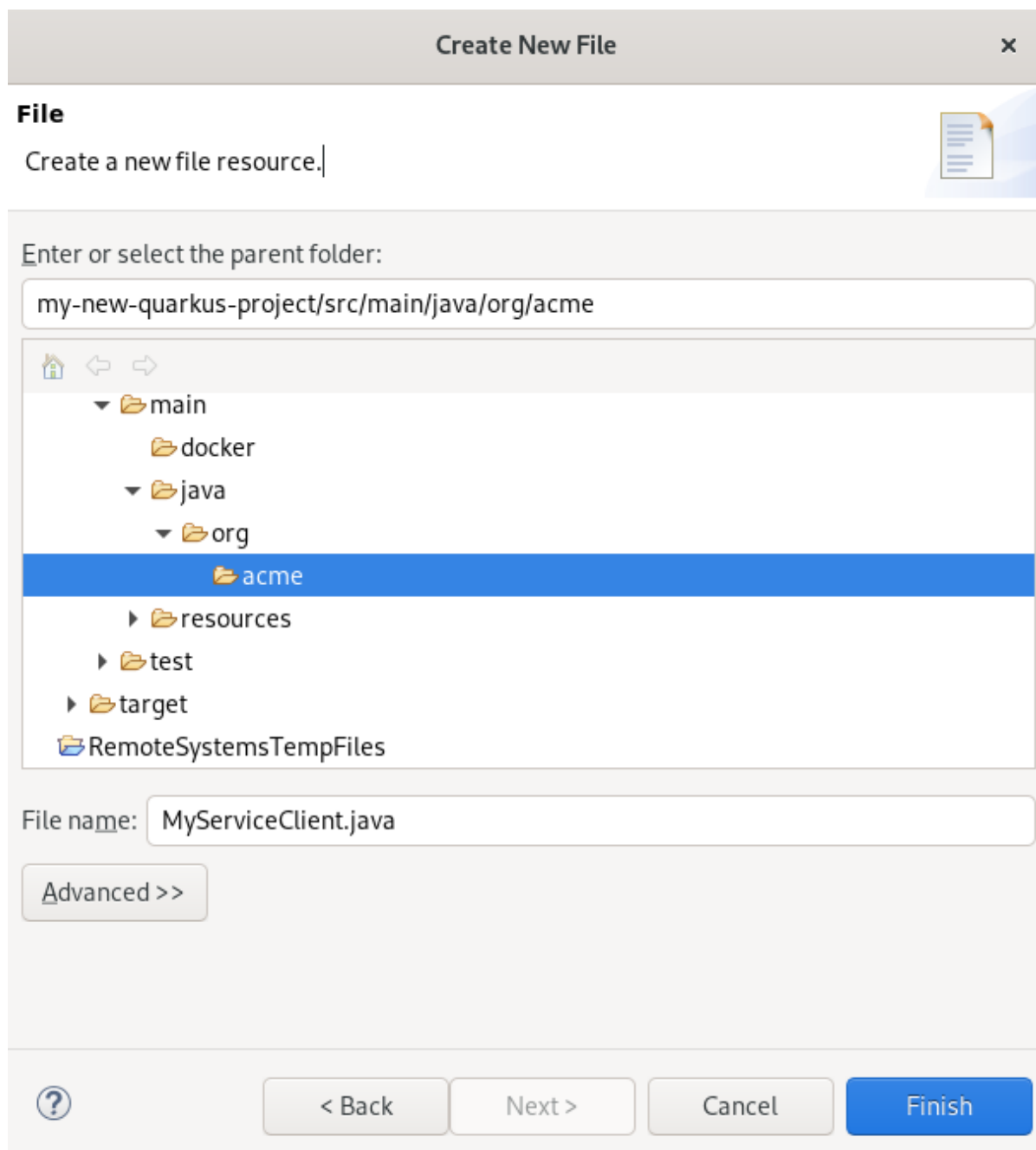
### Procedure

1. Start CodeReady Studio.
2. Start **Project Explorer**.
3. Expand your **Quarkus project** → **src/main/java**.
4. Right-click **org.acme** → **New** → **Other**.  
The **Select wizard** window appears.



5. Enter **file** in the search field.
6. Select **File**.
7. Click **Next**.  
The **Create a new file resource** window appears.





8. Name your new file.
9. Click **Finish**.
10. Paste the following content into your newly created file:

```

package org.acme;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;

@RegisterRestClient
public interface MyServiceClient {
    @GET

```

```
    ("/greet")  
    Response greet();  
}
```

11. Press **Ctrl+S** to save the changes.

### Additional resources

- For more information on how to use language support, see [Section 6.4.1, "Using Quarkus content assist"](#).
- For information on adjusting language support, see [Quarkus - Using the REST client, Create the interface](#).

## CHAPTER 7. HIBERNATE TOOLS BASICS IN CODEREADY STUDIO

Hibernate Tools is a collection of tools for projects related to Hibernate version 5 and earlier. The tools provide Eclipse plugins for reverse engineering, code generation, visualization, and interaction with Hibernate.

### 7.1. CREATING A NEW JAKARTA PERSISTENCE PROJECT

The following section describes how to create a new Jakarta Persistence project in CodeReady Studio.

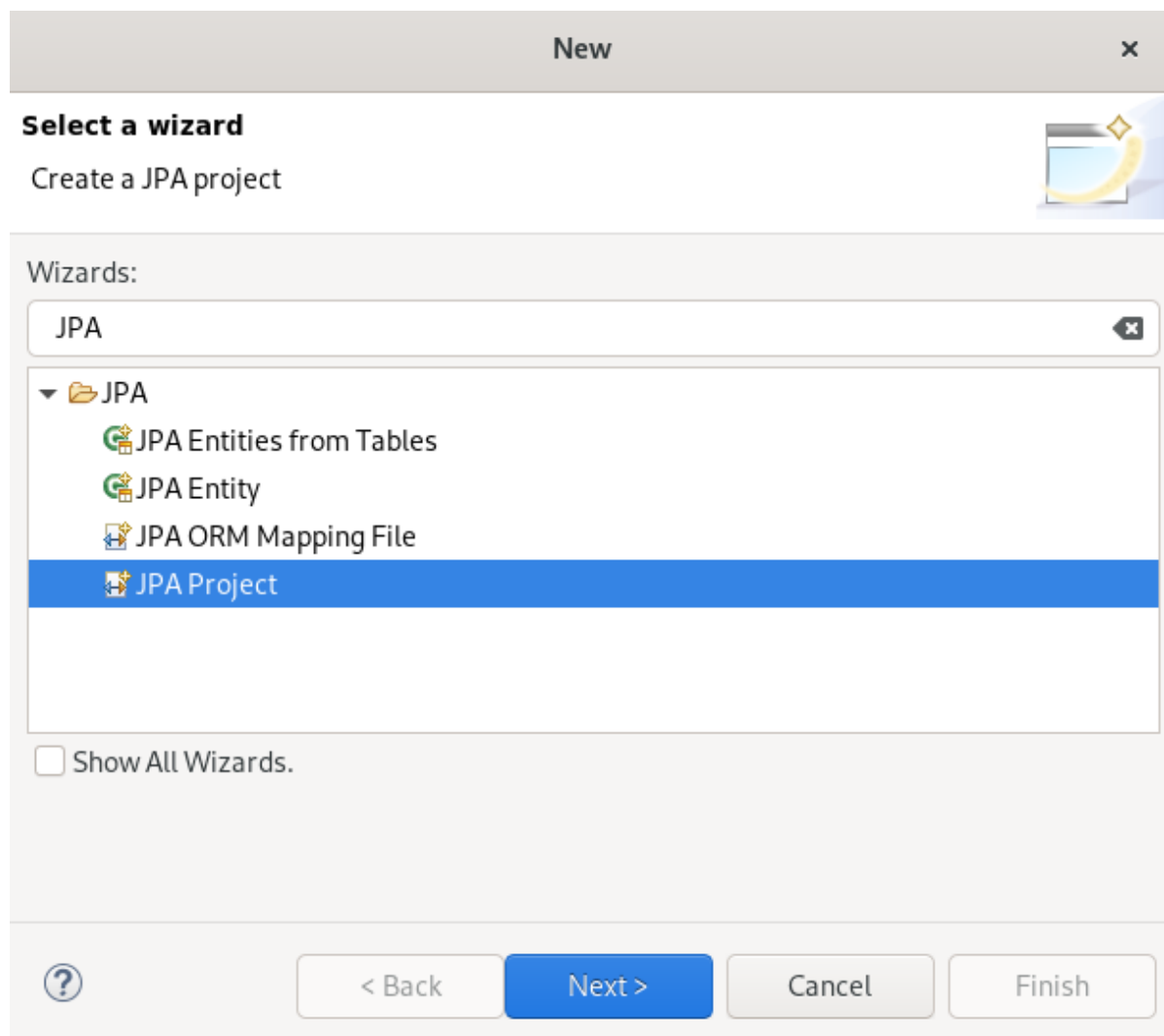
#### Prerequisites

- The Sakila database server is started.  
To start the Sakila database:
  - a. Download the [h2 version of the Sakila database](#).
  - b. Navigate to the directory that contains the **runh2.sh** file.
  - c. Execute the **runh2.sh** file:

```
┆ $ ./runh2.sh
```

#### Procedure


1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a Wizard** window appears.



3. Enter **JPA** in the search field.
4. Select **JPA Project**.
5. Click **Next**.  
The **New JPA Project** window appears.

New JPA Project ✕

### JPA Project

Configure JPA project settings. 

Project name:

Project location

Use default location

Location:  Browse...

Target runtime

New Runtime...

JPA version

Configuration

Modify...

Hint: Get started quickly by selecting one of the pre-defined project configurations.

EAR membership

Add project to an EAR

EAR project name:  New Project ...

Working sets

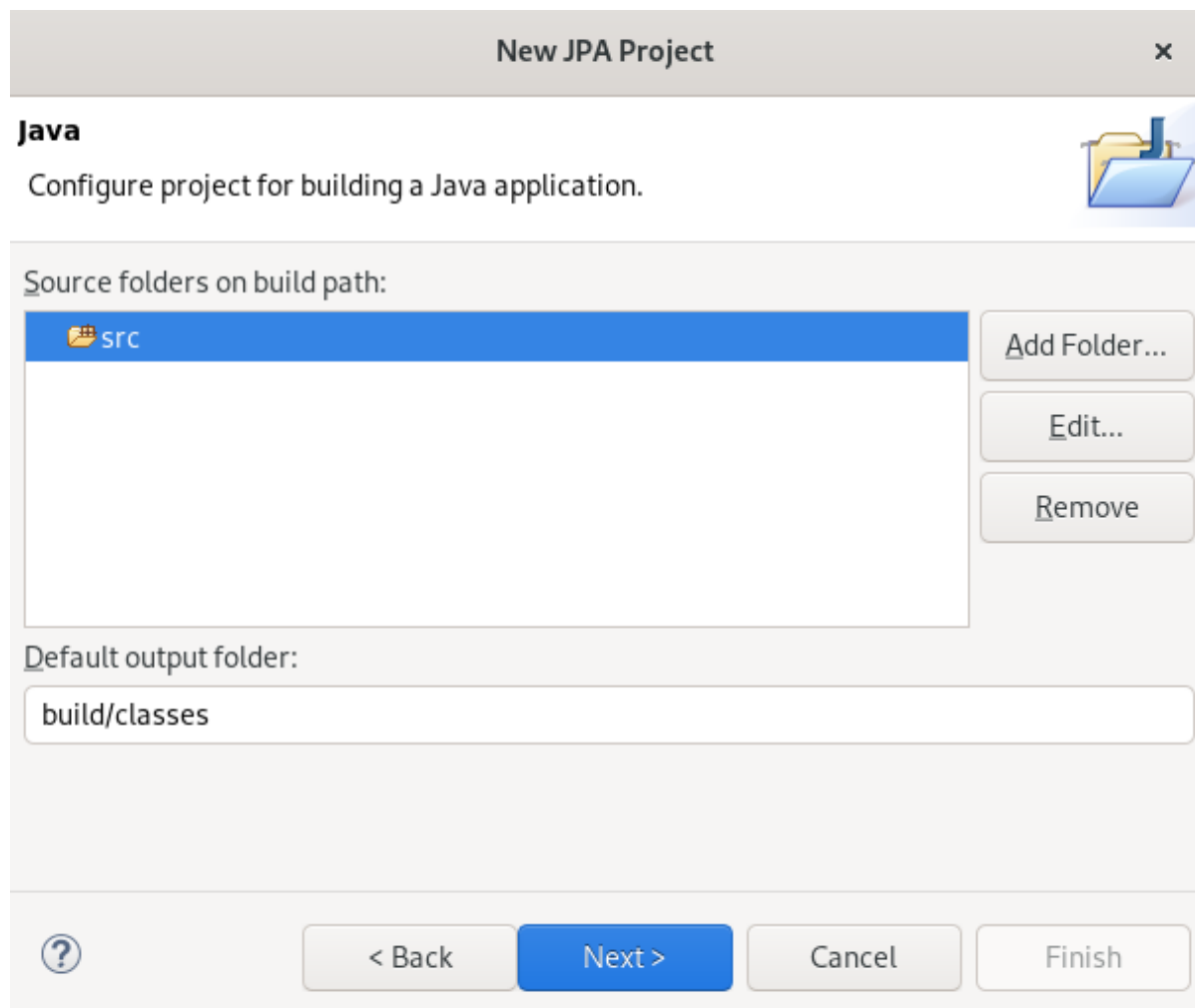
Add project to working sets New...

Working sets:  Select...

? Help
< Back
Next >
Cancel
Finish

6. Name your project.
7. Select the location for your project.
8. Click the down-arrow in the **Target runtime** field to select the runtime server.

- Set the **JPA version** to 2.1.
- Click **Next**.  
The **Java** window appears.



- Select the source folder.
- Click **Next**.  
The **JPA Facet** window appears.

**New JPA Project**

**JPA Facet**

⚠ Library configuration is disabled. The user may need to configure further classpath changes later.

Platform

Hibernate (JPA 2.1)

JPA implementation

Type: Disable Library Configuration

The JPA facet requires a JPA implementation library to be present on the project classpath. By disabling library configuration, the user takes on the responsibility of ensuring that the classpath is configured appropriately via alternate means.

Connection

<None>

[Add connection...](#)

Connect

13. Click the down-arrow in the **Platform** field and select **Hibernate (JPA 2.1)**.
14. Add user libraries or set the **JPA Implementation Type** to **Disable Library Configuration**. For more information on how to set up user libraries, see [Section 7.2, "Adding libraries"](#).
15. Click **Add connection**.  
The **Connection Profile** window appears.

**New Connection Profile**

**Connection Profile**

Create a Generic JDBC connection profile.

Connection Profile Types:

generic

Generic JDBC

Name:

sakila

Description (optional):

?

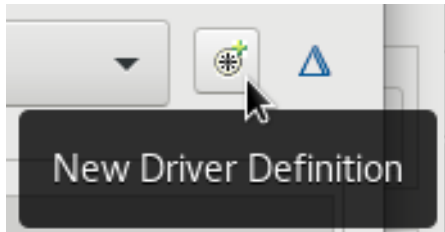
< Back

Next >

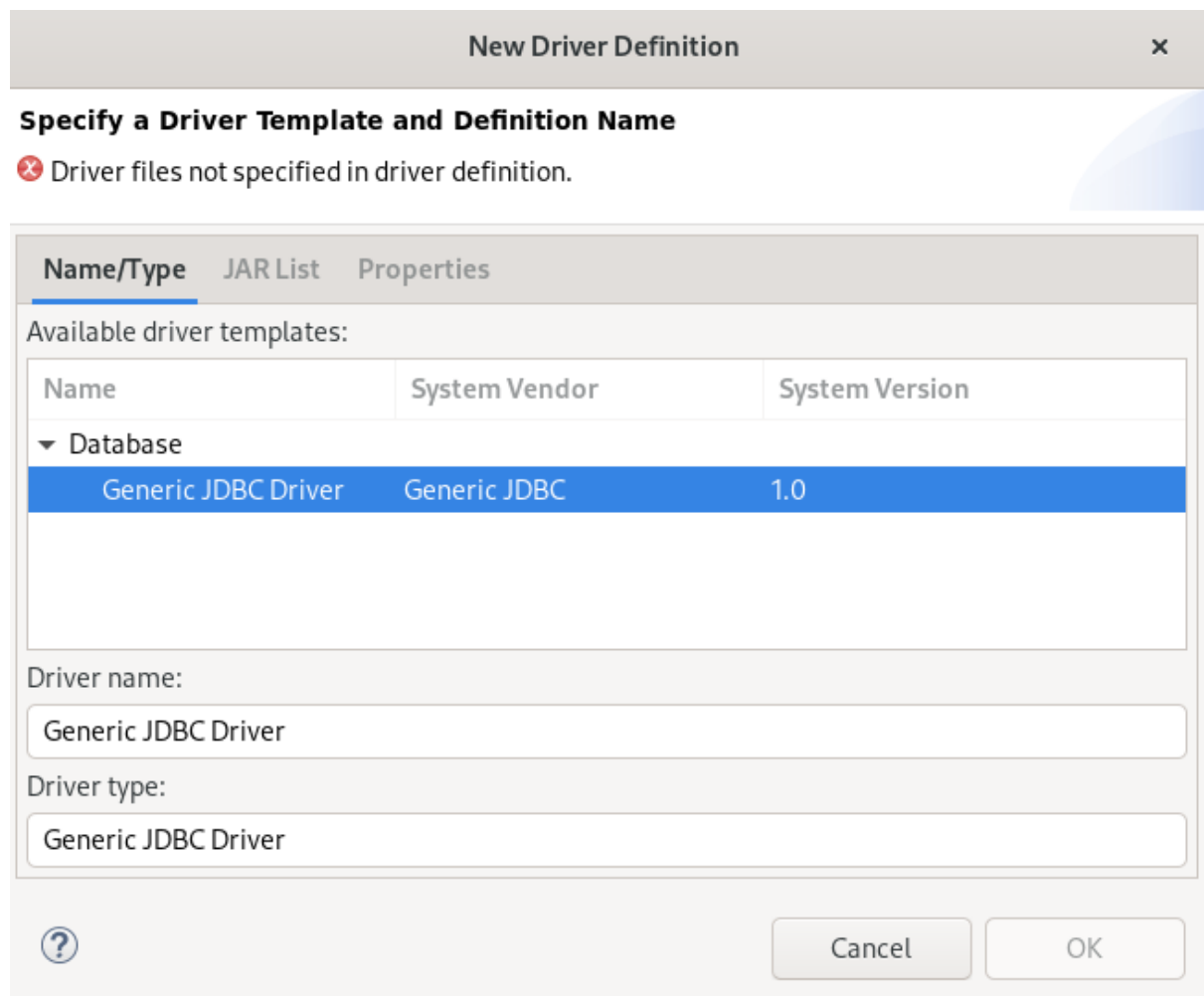
Cancel

Finish

16. Enter **Generic** in the search field.
17. Select **Generic JDBC**.
18. Enter **Sakila** in the **Name** field.
19. Click **Next**.  
The **Specify a Driver and Connection Details** window appears.
20. Click the **New Driver Definition** icon.

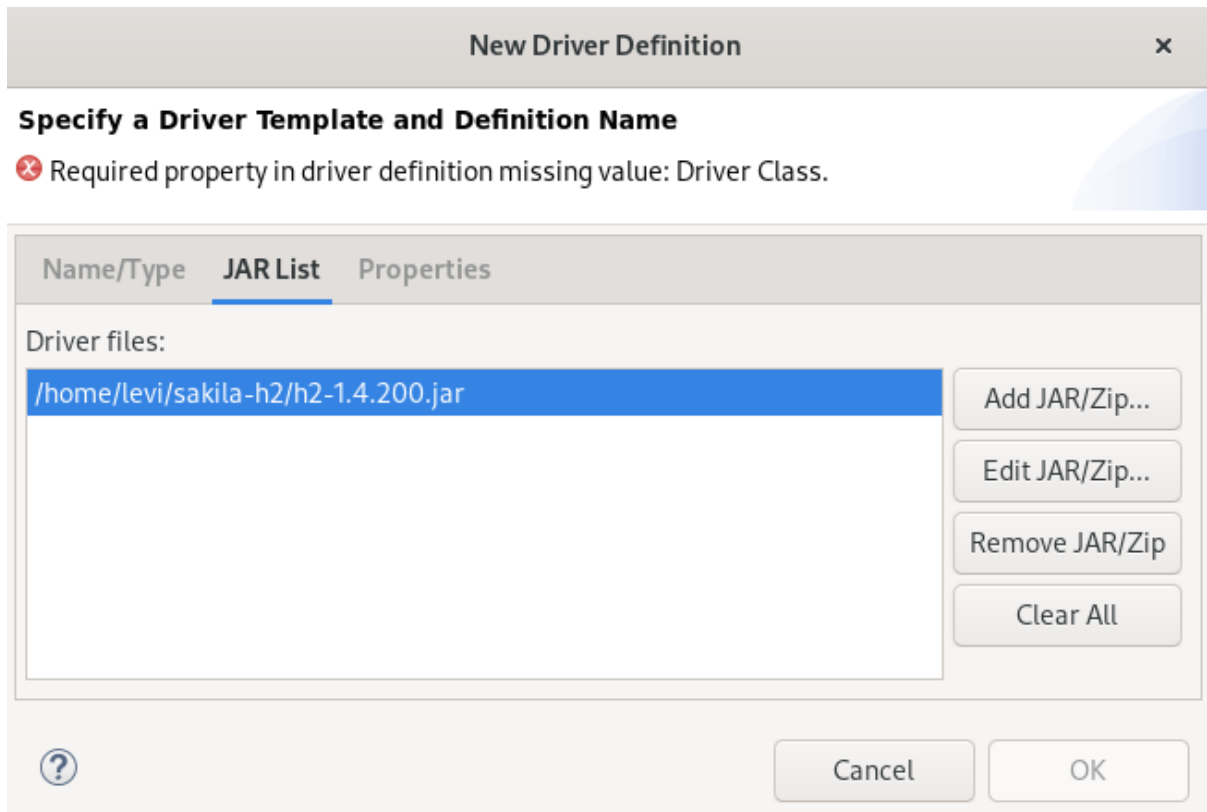


The **New Driver Definition** window appears.

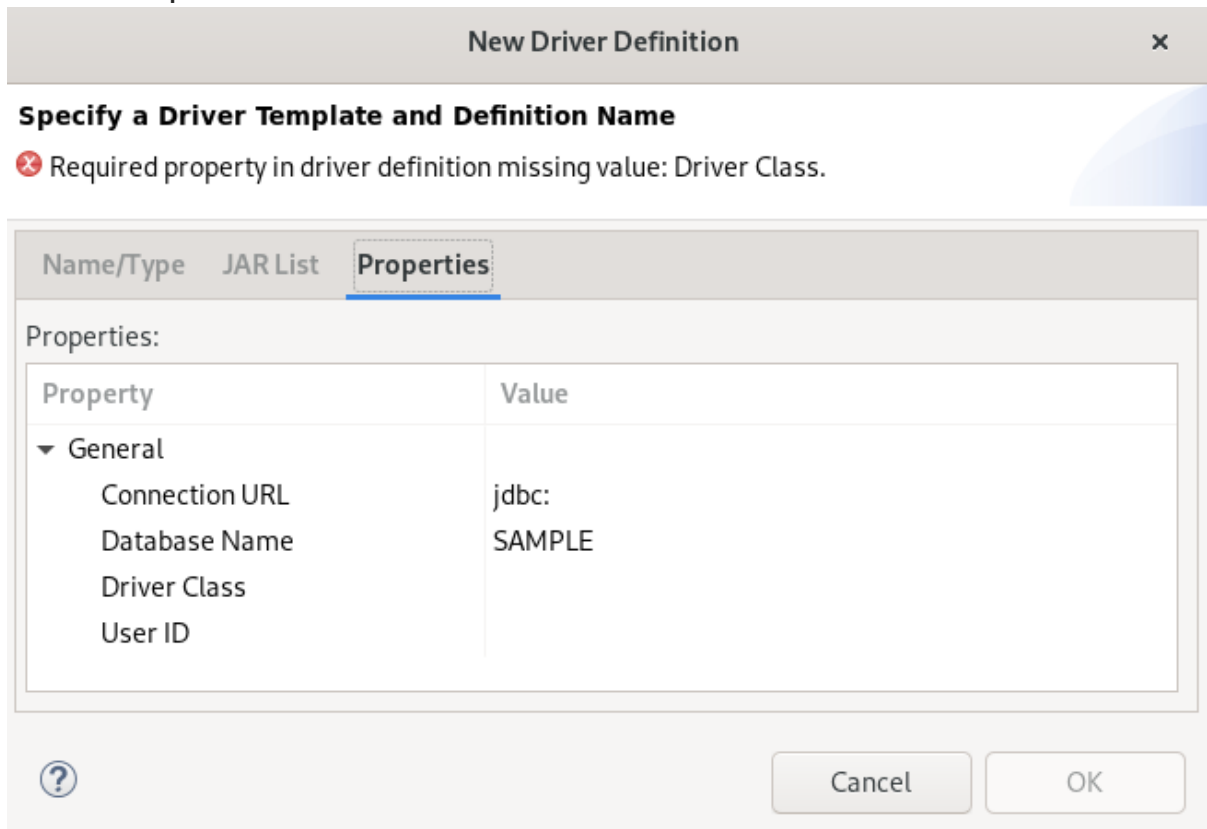


21. Select the **Generic JDBC Driver**.
22. Click the **JAR List** tab.



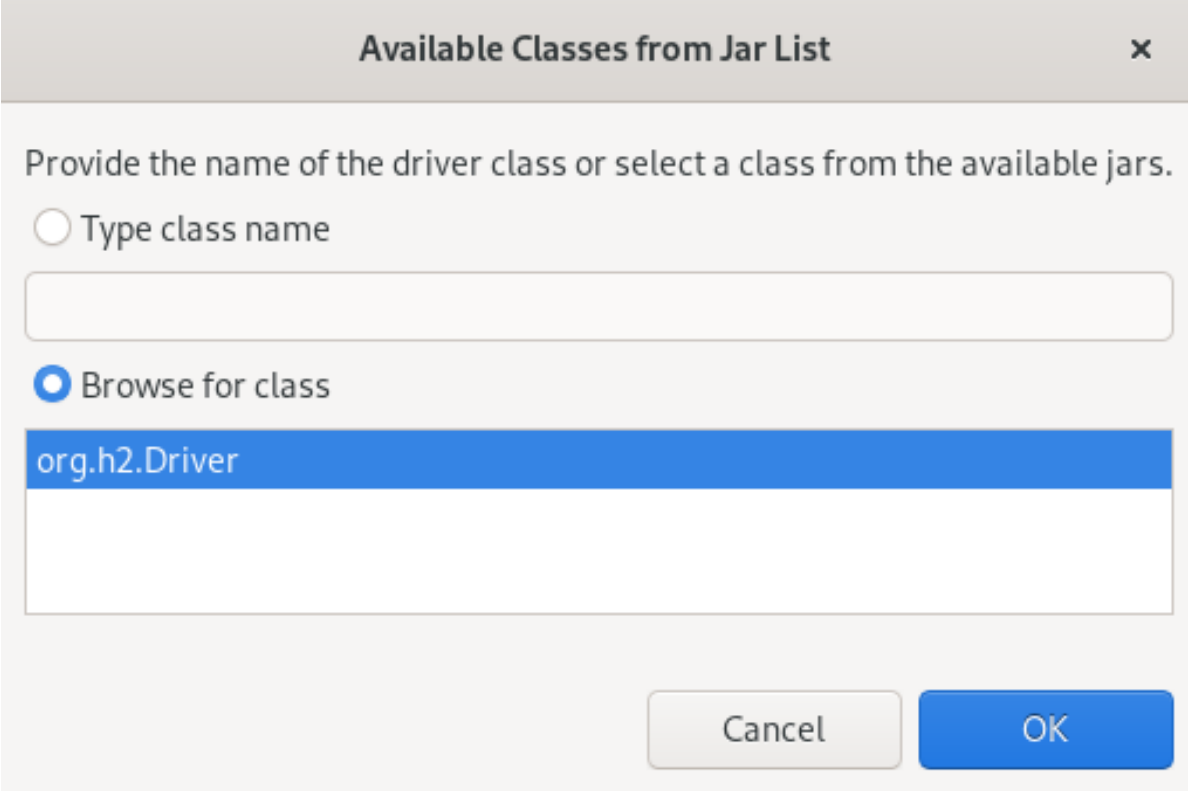


23. Click the **Add JAR/Zip** button.
24. Select the **.jar** file for the Sakila database.
25. Click the **Properties** tab.



26. Add **jdbc:h2:tcp://localhost/./sakila** to the **Connection URL** field.
27. Click the **Driver Class** field.

- Click the three dots icon at the end of the **Driver Class** field.  
The **Available Classes from Jar List** window appears.



**Available Classes from Jar List** ×

Provide the name of the driver class or select a class from the available jars.

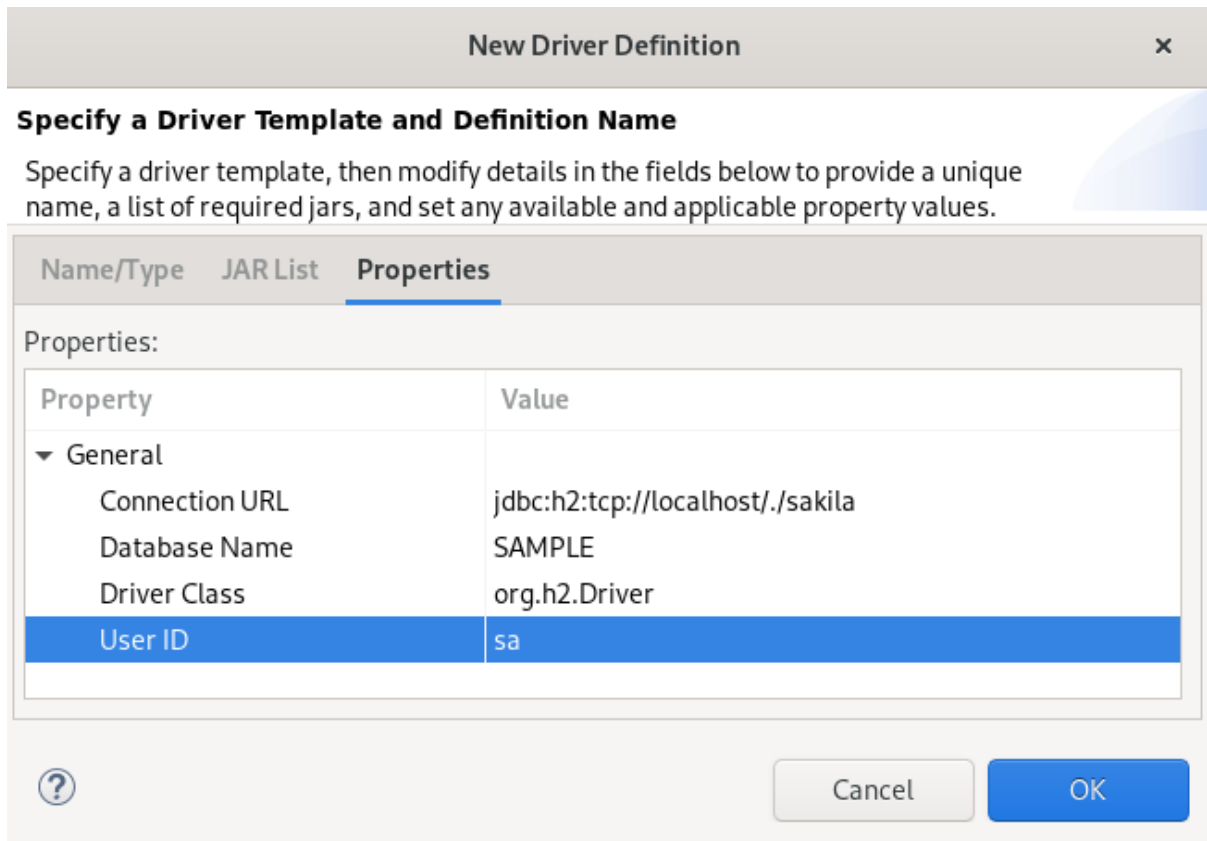
Type class name

Browse for class

org.h2.Driver

Cancel OK

- Select the **Browse for Class** option.
- Select **org.h2.Driver**.
- Click **OK**.
- Enter **sa** in the **User ID** field.



33. Click **OK** → **Finish** → **Finish**.

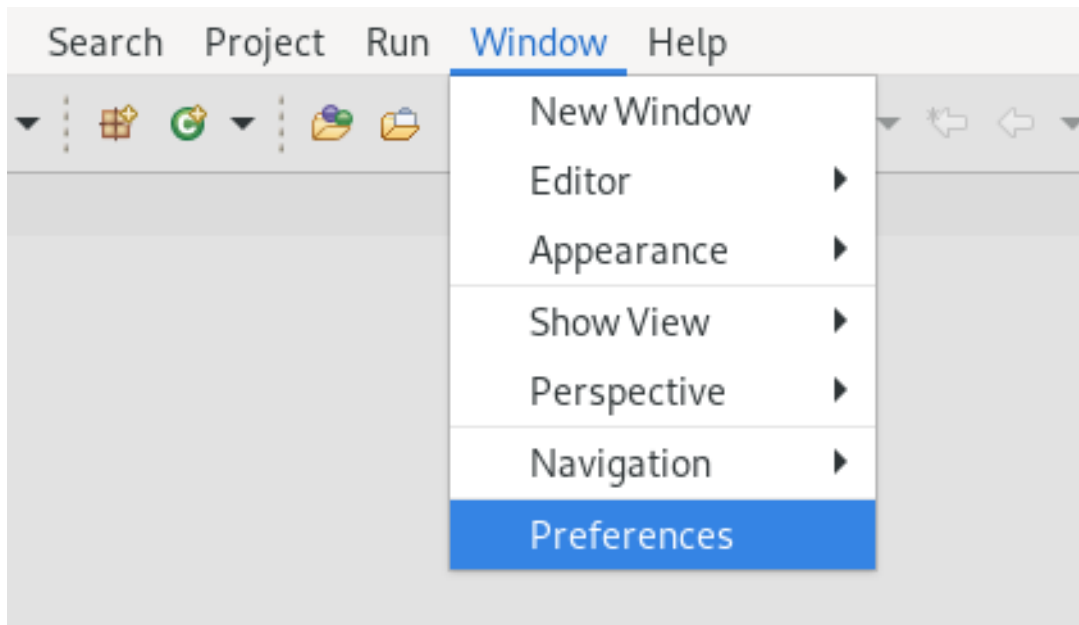
Your newly created Jakarta Persistence project is now listed in the **Project Explorer** view.

## 7.2. ADDING LIBRARIES

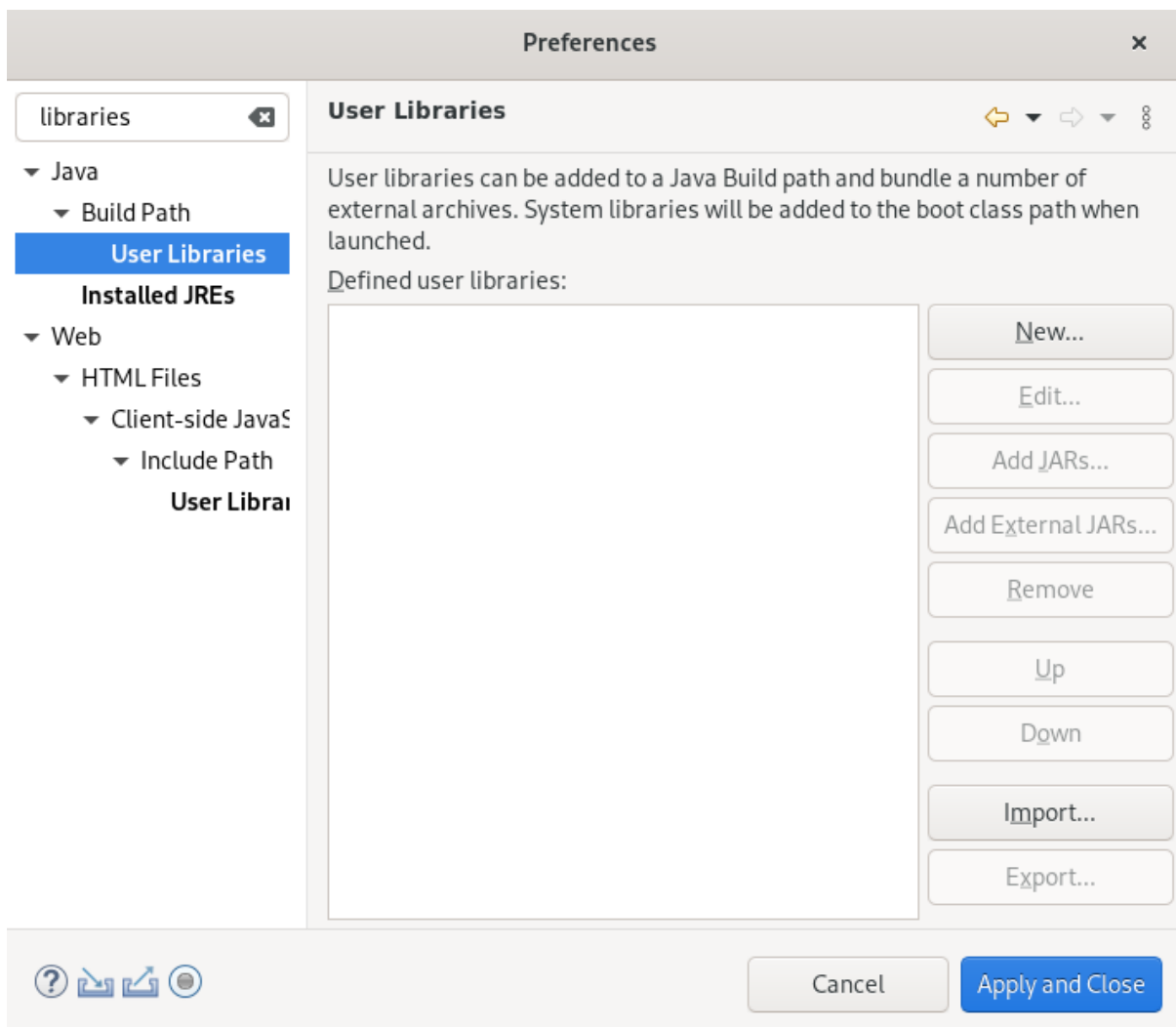
The following section describes how to add libraries to your Hibernate project in CodeReady Studio.

### Procedure

1. Download [Hibernate ORM](#).
2. Extract the files.
3. Start CodeReady Studio.
4. Click **Window** → **Preferences**.



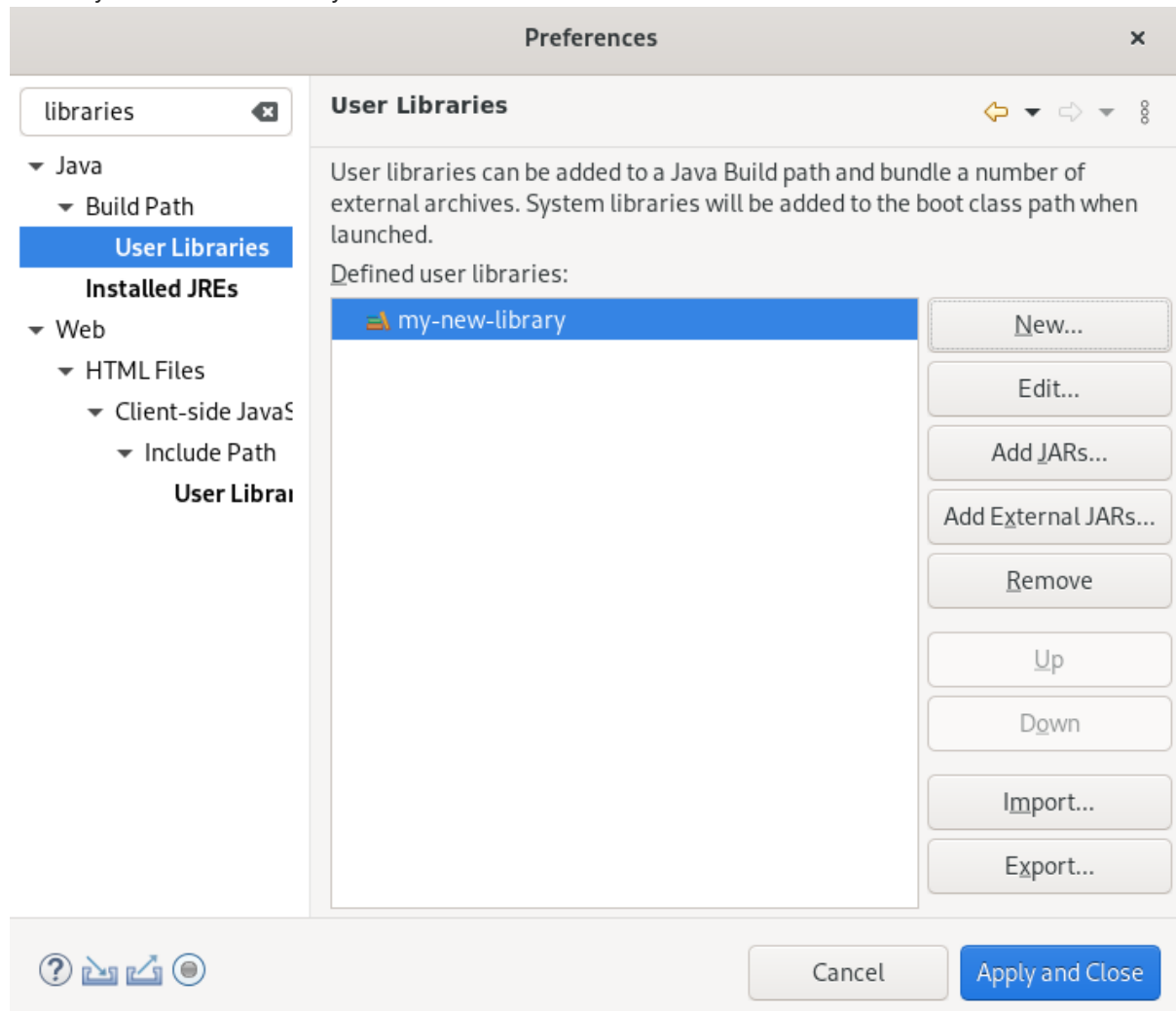
The **Preferences** window appears.



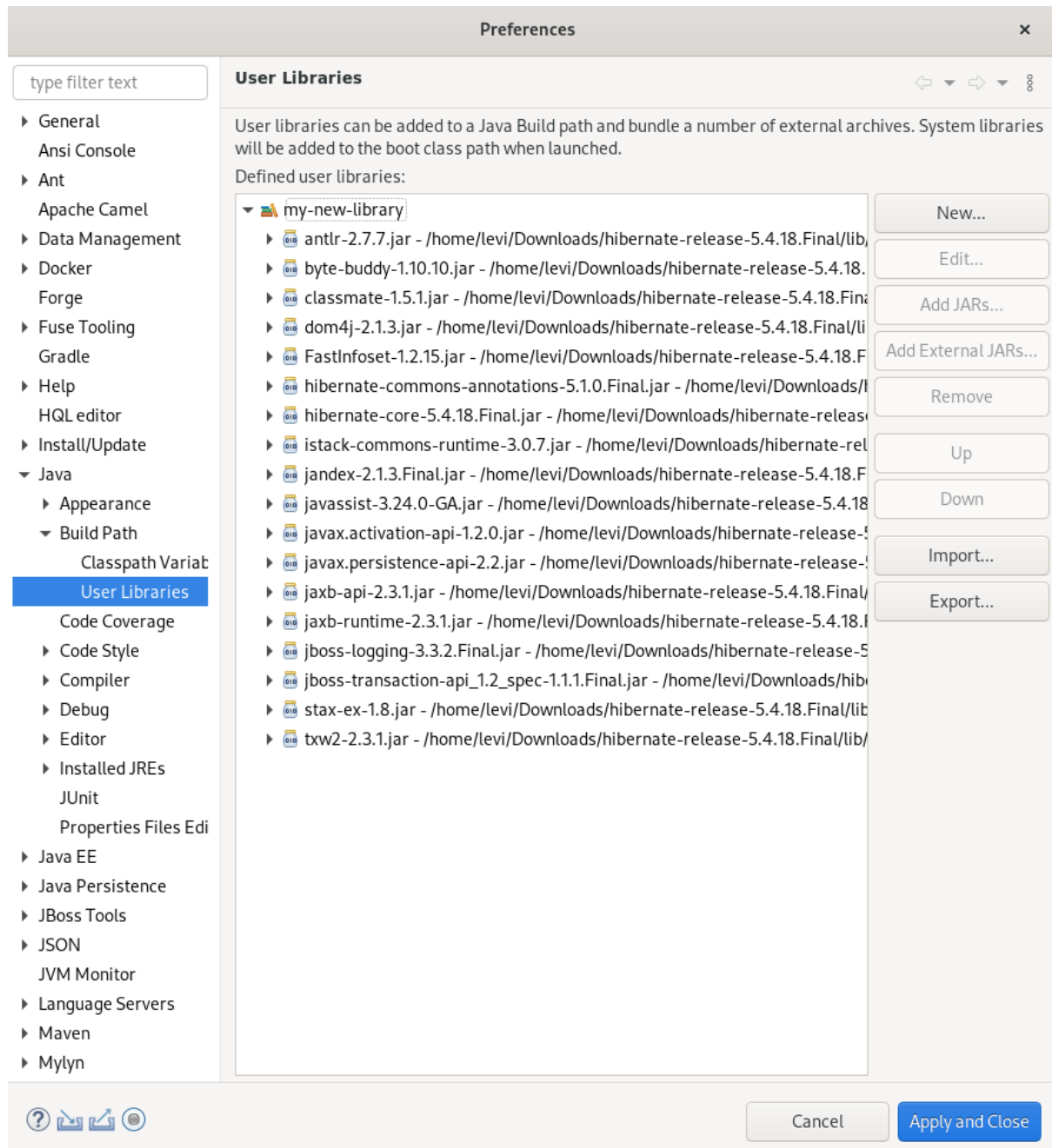
5. Enter **Libraries** in the search field.
6. Select **User Libraries** under **Java**.
7. Click the **New** button.

The **New User Library** window appears.

8. Name your user library.
9. Click **OK**.
10. Select your new user library.



11. Click the **Add External JARs** button.
12. Select the directory you extracted the **Hibernate ORM** files into.
13. Navigate to the **/lib/required/** directory.
14. Select a **.jar** file.
15. Click **Open**.  
Your selected **.jar** file appears under your user library.



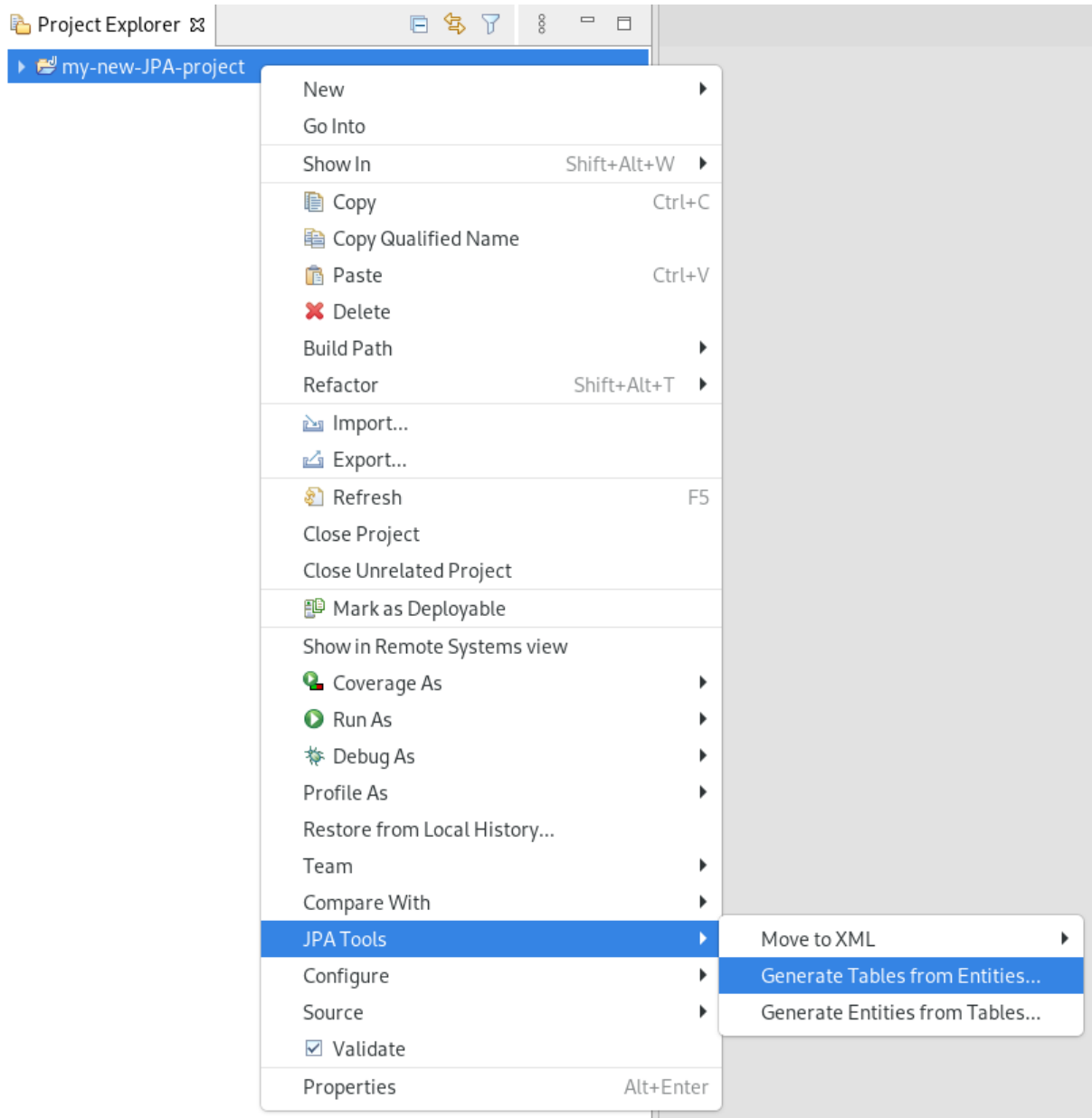
16. Click **Apply and Close**.

## 7.3. GENERATING TABLES FROM ENTITIES

The following section describes how to generate tables from entities for your Hibernate project in CodeReady Studio.

### Procedure

1. Start CodeReady Studio.
2. Open **Project Explorer**.
3. Right-click your **JPA project** → **JPA Tools** → **Generate Tables from Entities**.



The **Generate Tables from Entities** window appears.

**Generate Tables from Entities** x

**Use existing console configuration or connection profile for database connection**

Output directory: /my-new-JPA-project/src Browse...

File name: schema.ddl

Export to Database

Use Console Configuration

Console configuration: my-new-JPA-project

Hibernate Version: 3.5

**Database Settings**

Database Connection: sakila

Database dialect: [Autodetect]

? Cancel Finish

4. Select the **Use Console Configuration** check box.
5. Click **Finish**.

## 7.4. CREATING A HIBERNATE MAPPING FILE

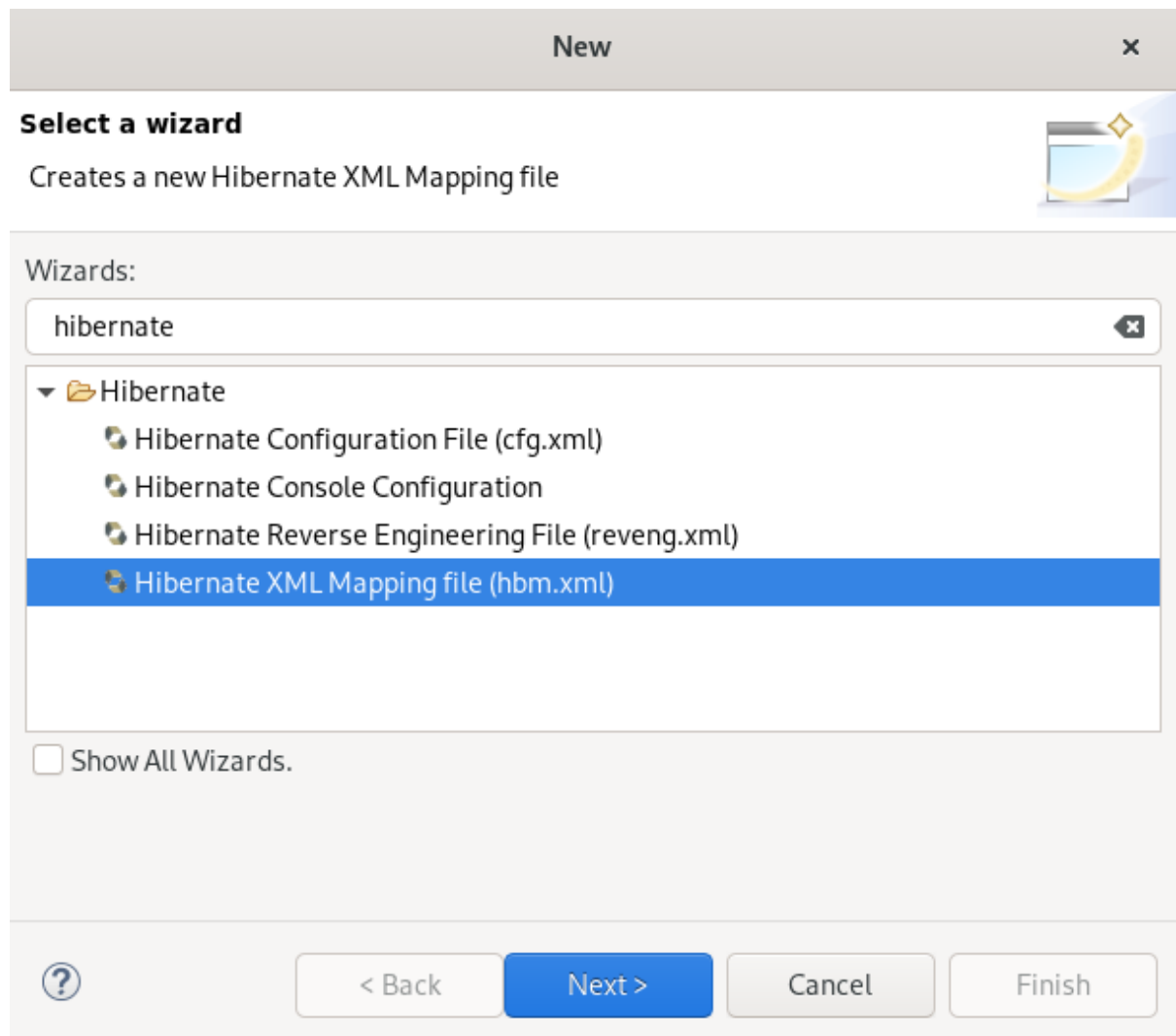
Hibernate mapping files specify how your objects relate to the database tables.

The following section describes how to create a Hibernate mapping file in CodeReady Studio.

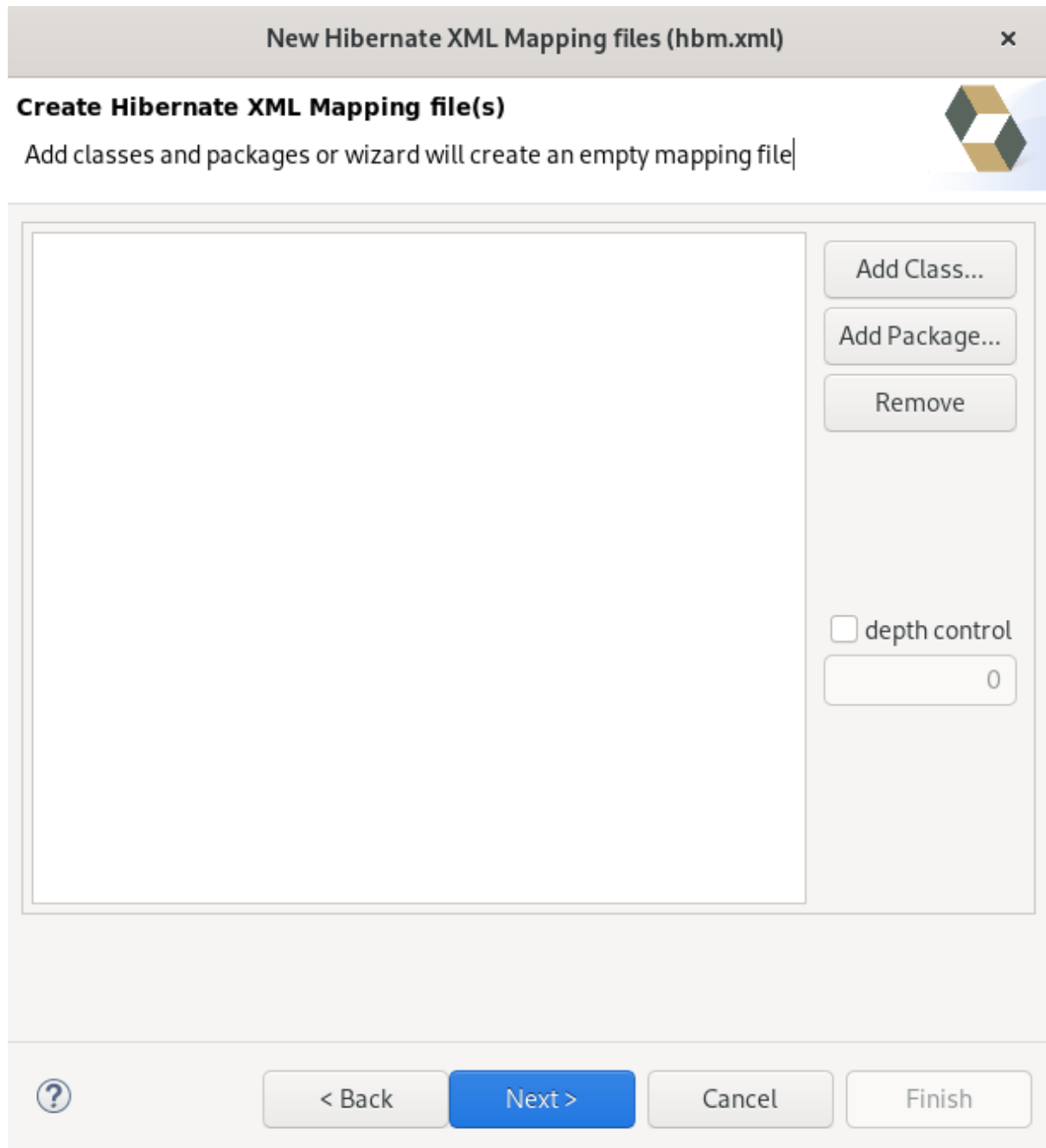
### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.

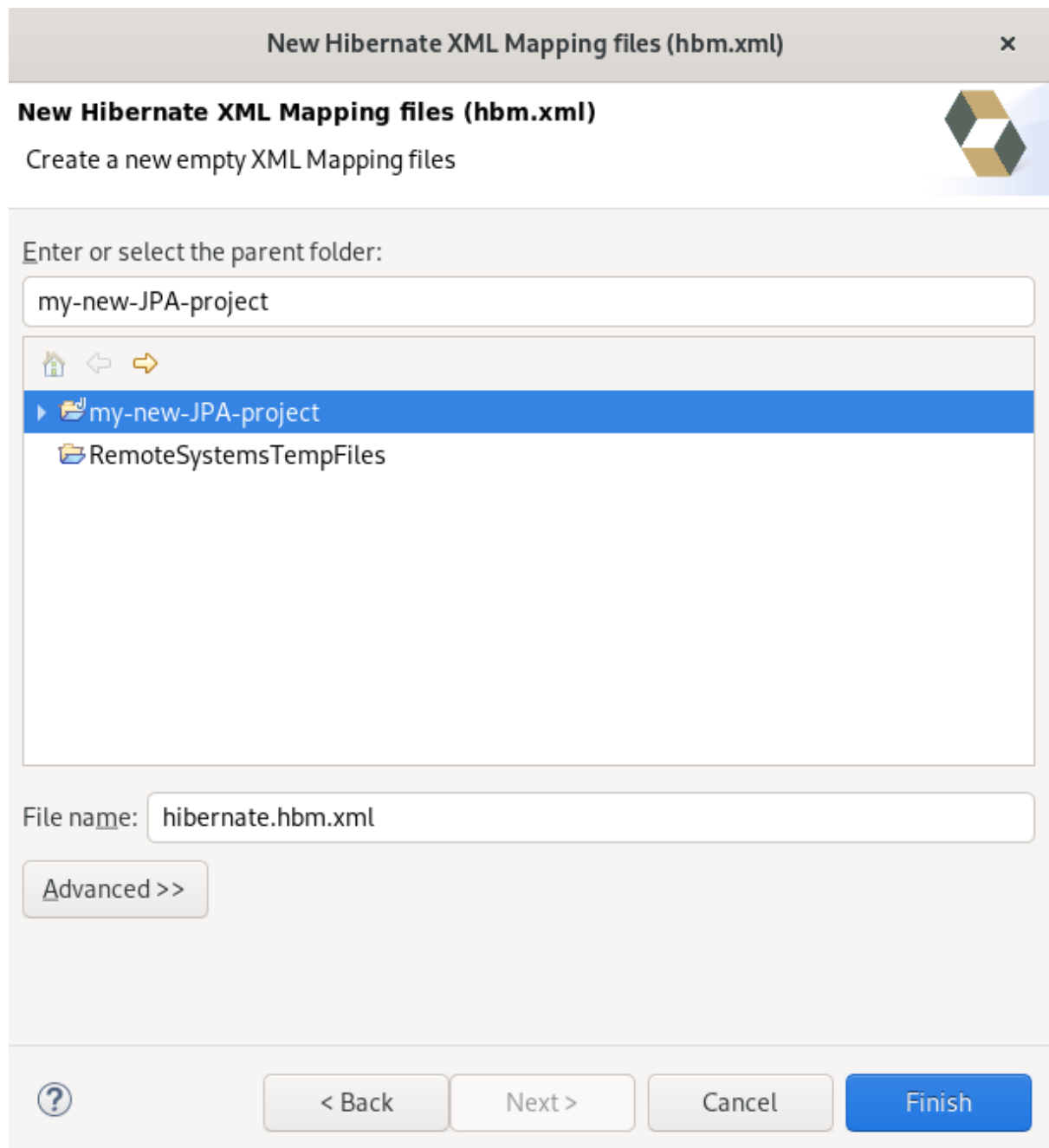




3. Enter **Hibernate** in the search field.
4. Select **Hibernate XML Mapping file (hbm.xml)**
5. Click **Next**.  
The **Create Hibernate XML Mapping file** window appears.



6. Click the **Add Class** button to add classes.
7. Click the **Add Package** button to add packages.  
Alternatively, you can create an empty **.hbm.xml** file by not selecting any packages or classes.
8. Select the **depth control** check box to define the dependency depth used when choosing classes.
9. Click **Next**.  
The **New Hibernate XML Mapping files** window appears.



10. Select the parent directory.
11. Name your **.hbm.xml** file .
12. Click **Finish**.

## 7.5. CREATING A HIBERNATE CONFIGURATION FILE

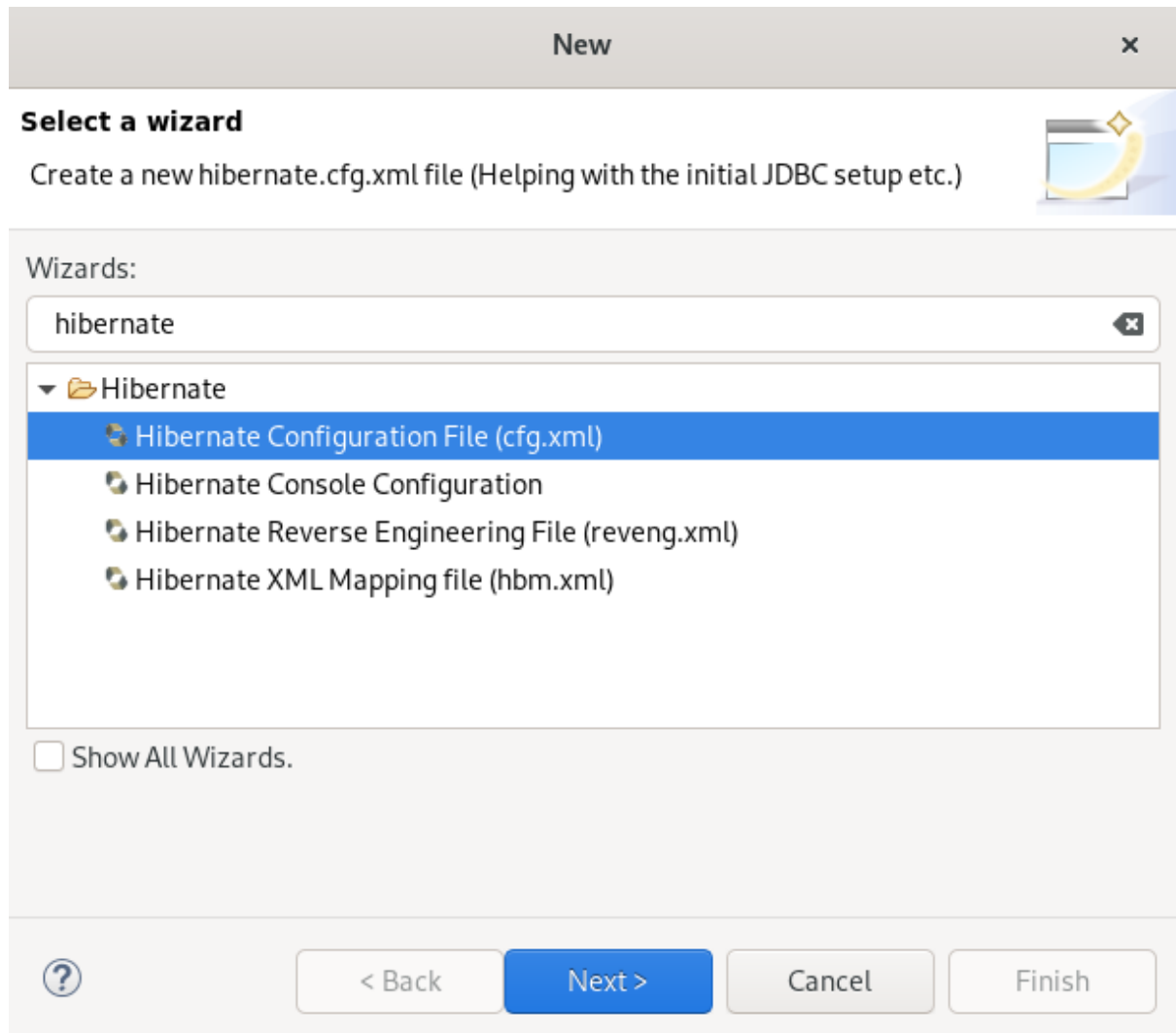
For reverse engineering, prototype queries, or Hibernate Core usage, a **hibernate.properties** or a **hibernate.cfg.xml** file is required. CodeReady Studio provides a wizard to generate the configuration file **hibernate.cfg.xml**.

The following section describes how to create a Hibernate configuration file in CodeReady Studio.

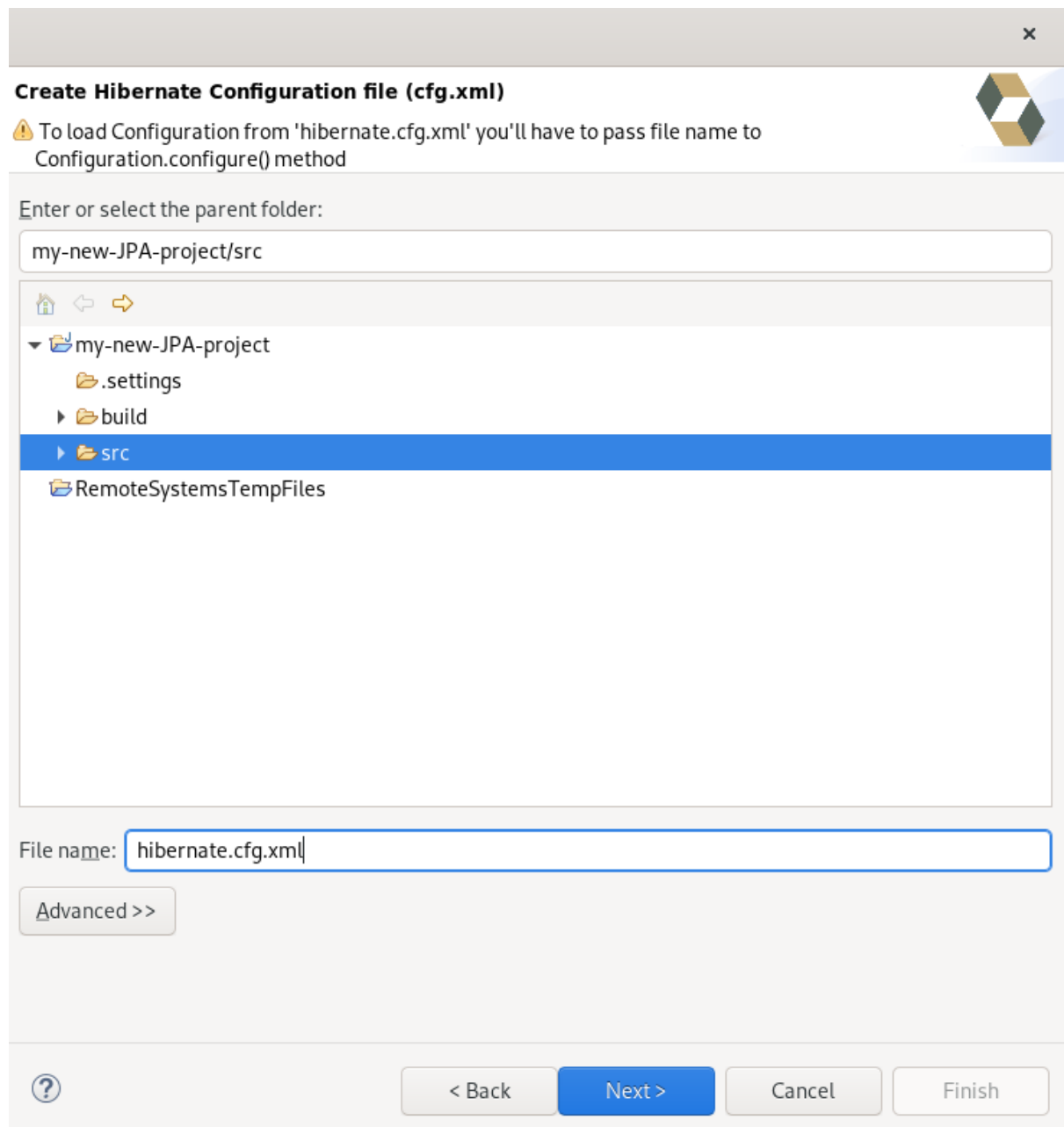
### Procedure

1. Start CodeReady Studio.

2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



3. Enter **Hibernate** in the search field.
4. Select **Hibernate Configuration file (cfg.xml)**.
5. Click **Next**.  
The **Create Hibernate Configuration file (cfg.xml)** window appears.



6. Select the parent directory.
7. Click **Next**.  
The **Hibernate Configuration File (cfg.xml)** window appears.

**Hibernate Configuration File (cfg.xml)**

This wizard creates a new configuration file to use with Hibernate.

Container: /my-new-JPA-project/src

File name: hibernate.cfg.xml

Hibernate version: 5.4

Session factory name:

[Get values from Connection](#)

Database dialect: MySQL

Driver class: org.gjt.mm.mysql.Driver

Connection URL: jdbc:mysql://<hostname>/<database>

Default Schema:

Default Catalog:

Username:

Password:

Create a console configuration

? < Back Next > Cancel Finish

8. Click the down-arrow in the **Database dialect** field to select the database.
9. Click the down-arrow in the **Driver class** field to select the driver.
10. Click the down-arrow in the **Connection URL** field to select the URL.
11. Click **Finish**.

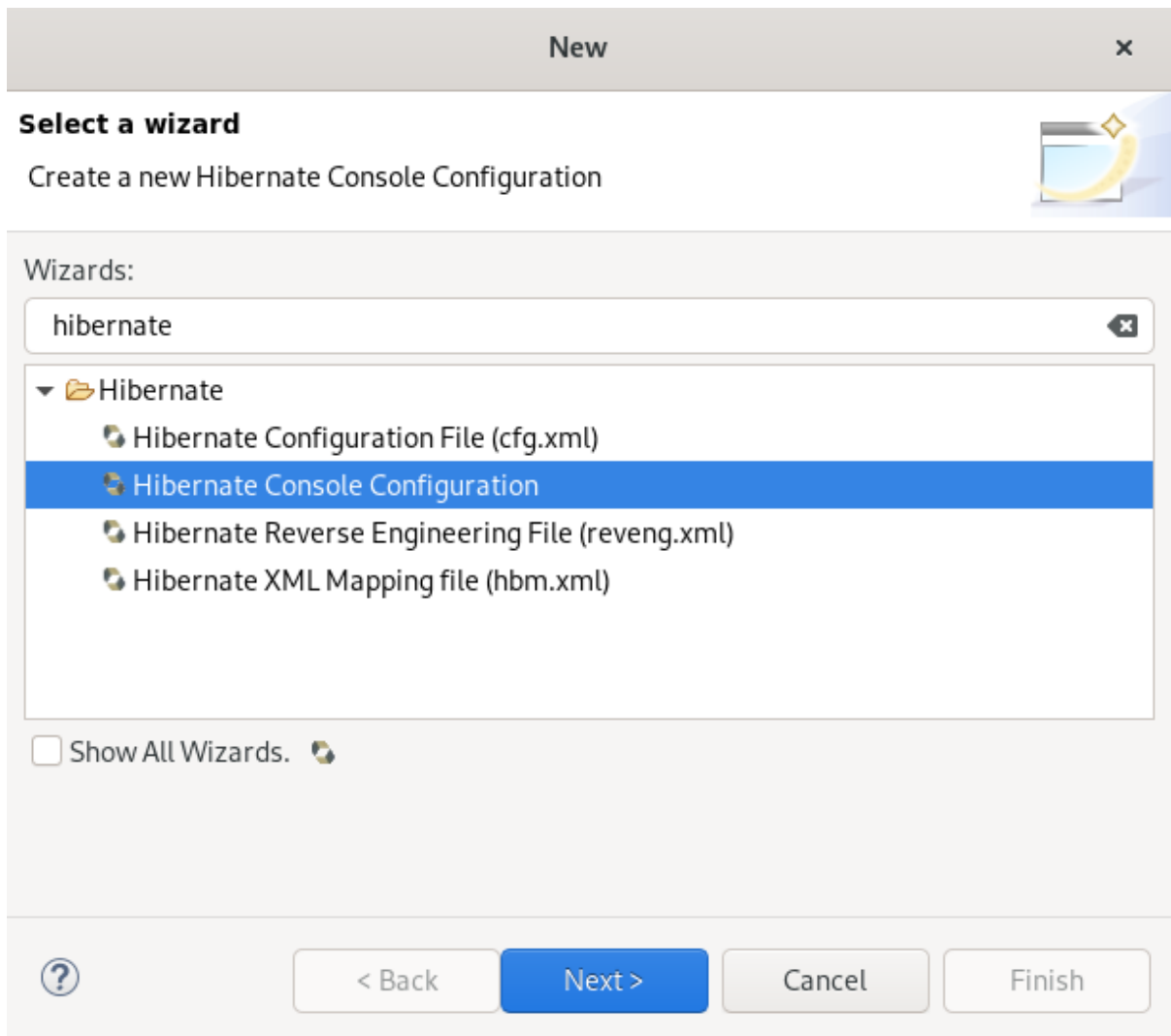
## 7.6. CREATING A HIBERNATE CONSOLE CONFIGURATION FILE

A console configuration file describes how the Hibernate plugin configures Hibernate. It also describes the configuration files and classpaths needed to load the POJOs, JDBC drivers, and so on. It is required to make use of query prototyping, reverse engineering and code generation. You can have multiple console configurations per project, however, one configuration is sufficient.

The following section describes how to create a Hibernate console configuration file in CodeReady Studio.

### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



3. Enter **hibernate** in the search field.
4. Select **Hibernate Console Configuration**.
5. Click **Next**.  
The **Create Hibernate Console Configuration** window appears.

**Create Hibernate Console Configuration**

Unable to create requested service [org.hibernate.engine.jdbc.env.spi.JdbcEnvironment]

Name: my-new-console-config

Main Options Classpath Mappings Common

Type:  
 Core  Annotations (jdk 1.5+)  JPA (jdk 1.5+)

Hibernate Version: 5.4

Project:  
 my-new-JPA-project Browse...

Database connection:  
 [Hibernate configured connection] New... Edit...

Property file:  
 Setup...

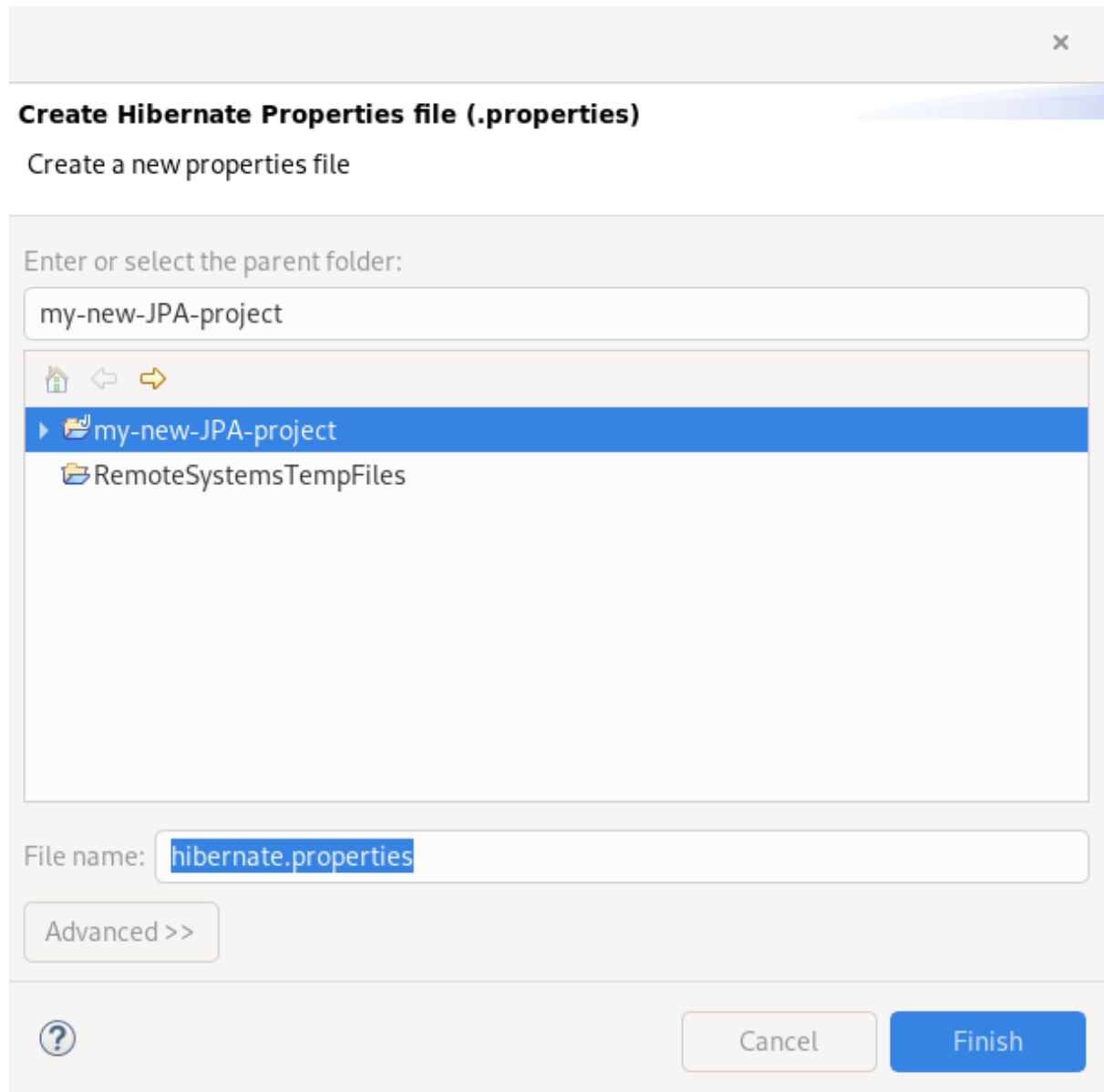
Configuration file:  
 Setup...

Persistence unit:  
 Browse...

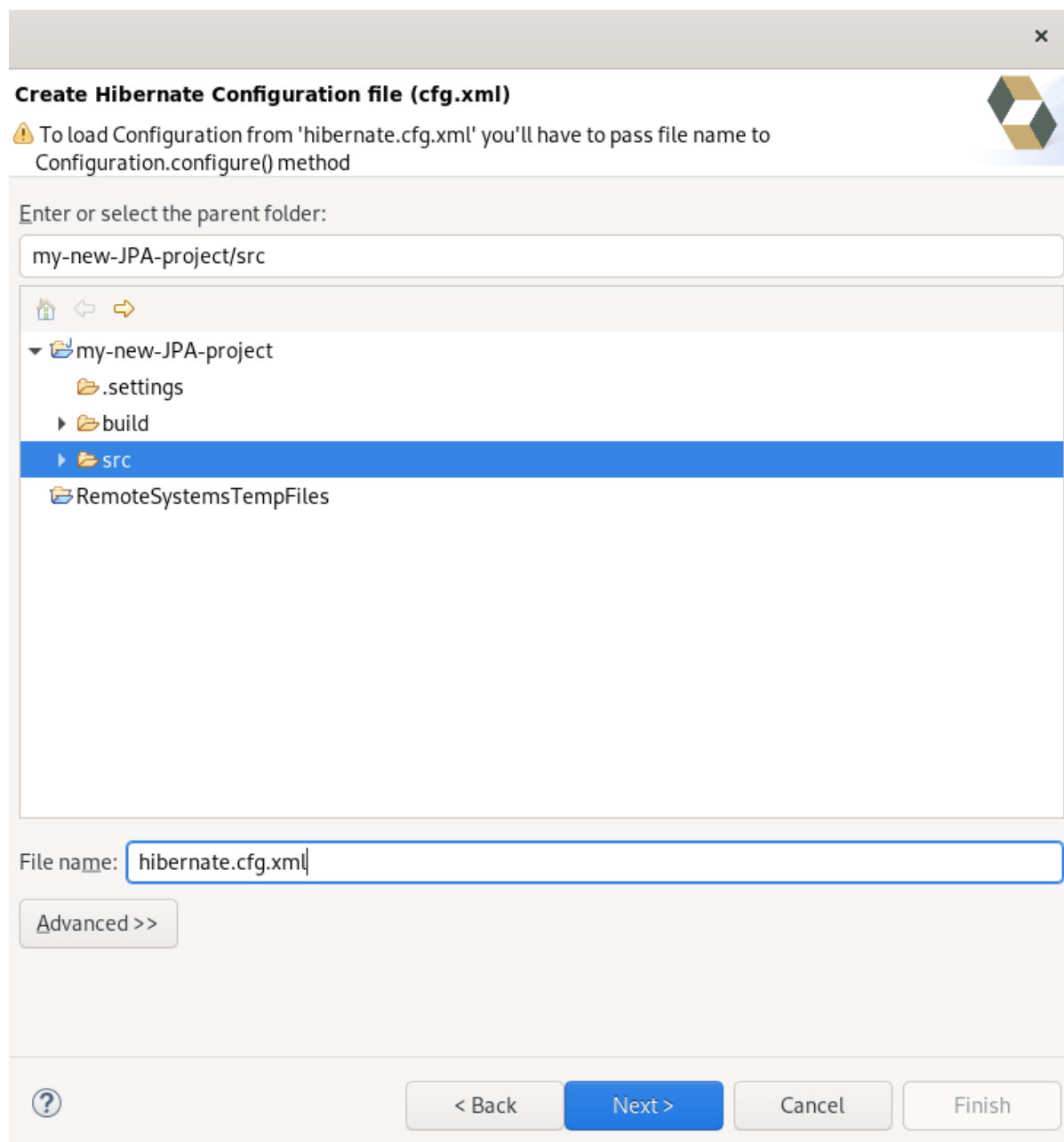
? < Back Next > Cancel Finish

6. Name your configuration file .
7. Ensure that the **Type** is set to **Core**.
8. Select the correct **Hibernate version**.
9. Click **Browse** to locate your project.
10. Click **New** to configure a new **Database connection**.  
The **New Connection Profile** window appears.
11. Select the **Database Connection** or create a new one.
12. Click **Setup** to set up the **Property file**.  
The **Setup property file** window appears.
13. Click **Create new**.  
The **Create Hibernate Properties file (.properties)** window appears.





14. Select the parent directory.
15. Name your **.properties** file.
16. Click **Finish**.
17. Click **Setup** to set up the **Configuration file**.
18. Select the path to the target **.cfg.xml** file.  
The **Setup configuration file** window appears.
19. Click **Create new**.  
The **Create Hibernate Configuration file (cfg.xml)** window appears.



20. Select the parent directory.

21. Click **Next**.

The **Hibernate Configuration File (cfg.xml)** window appears.

**Hibernate Configuration File (cfg.xml)**

This wizard creates a new configuration file to use with Hibernate.

Container: /my-new-JPA-project/src

File name: hibernate.cfg.xml

Hibernate version: 5.4

Session factory name:

[Get values from Connection](#)

Database dialect: MySQL

Driver class: org.gjt.mm.mysql.Driver

Connection URL: jdbc:mysql://<hostname>/<database>

Default Schema:

Default Catalog:

Username:

Password:

Create a console configuration

? < Back Next > Cancel Finish

22. Click the down-arrow in the **Database dialect** field to select the database.
23. Click the down-arrow in the **Driver class** field to select the driver.
24. Click the down-arrow in the **Connection URL** field to select the URL.
25. Click **Finish**.  
The **Create Hibernate Console Configuration** window appears.

**Create Hibernate Console Configuration**

This wizard allows you to create a configuration for Hibernate Console.

Name:

Main
  Options
  Classpath
  Mappings
  Common

Type:

Core
  Annotations (jdk 1.5+)
  JPA (jdk 1.5+)

Hibernate Version:

Project:

Database connection:

Property file:

Configuration file:

Persistence unit:

26. Set the database connection to **sakila**.

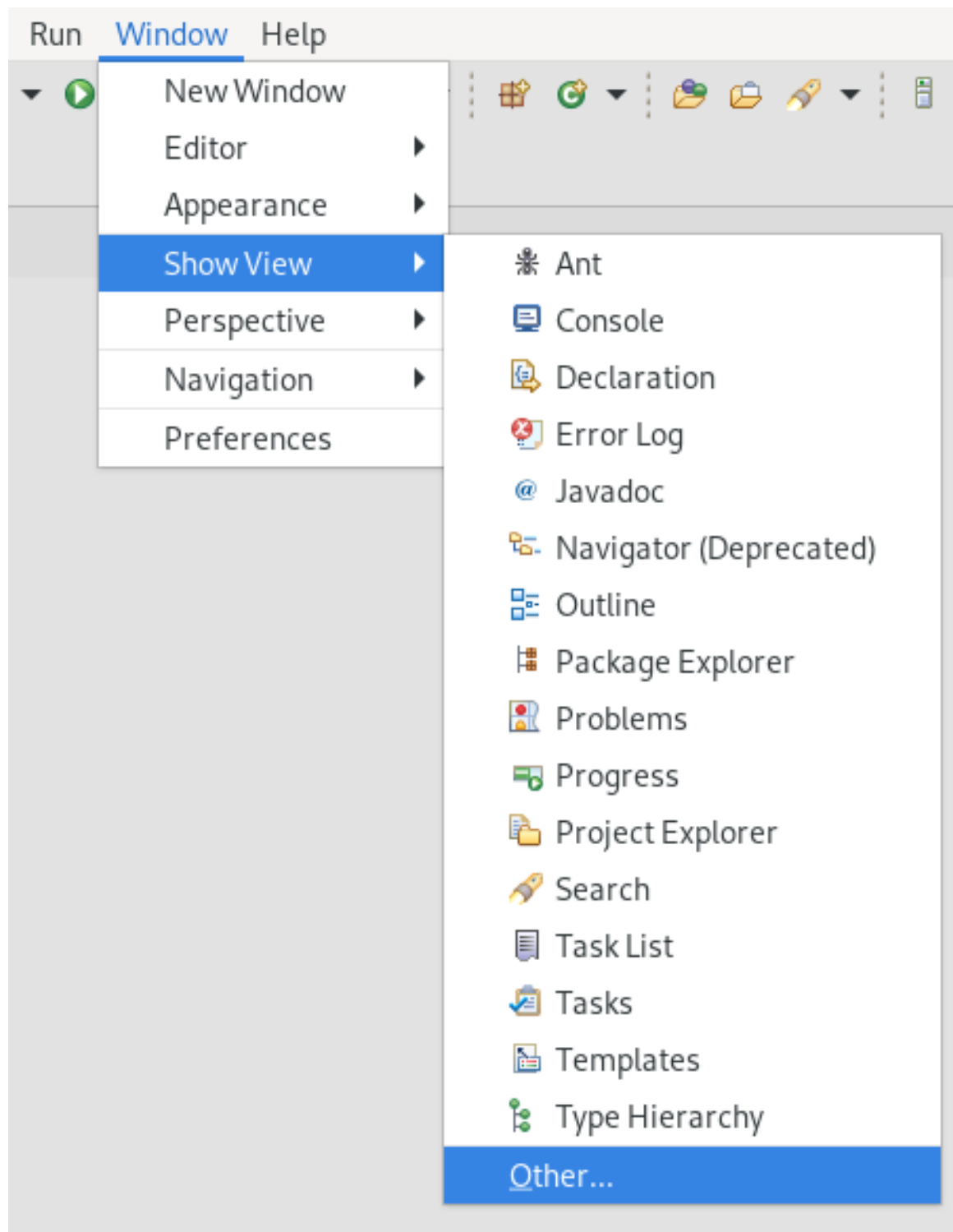
27. Click **Finish**.

## 7.7. EDITING HIBERNATE PROJECT CONFIGURATIONS

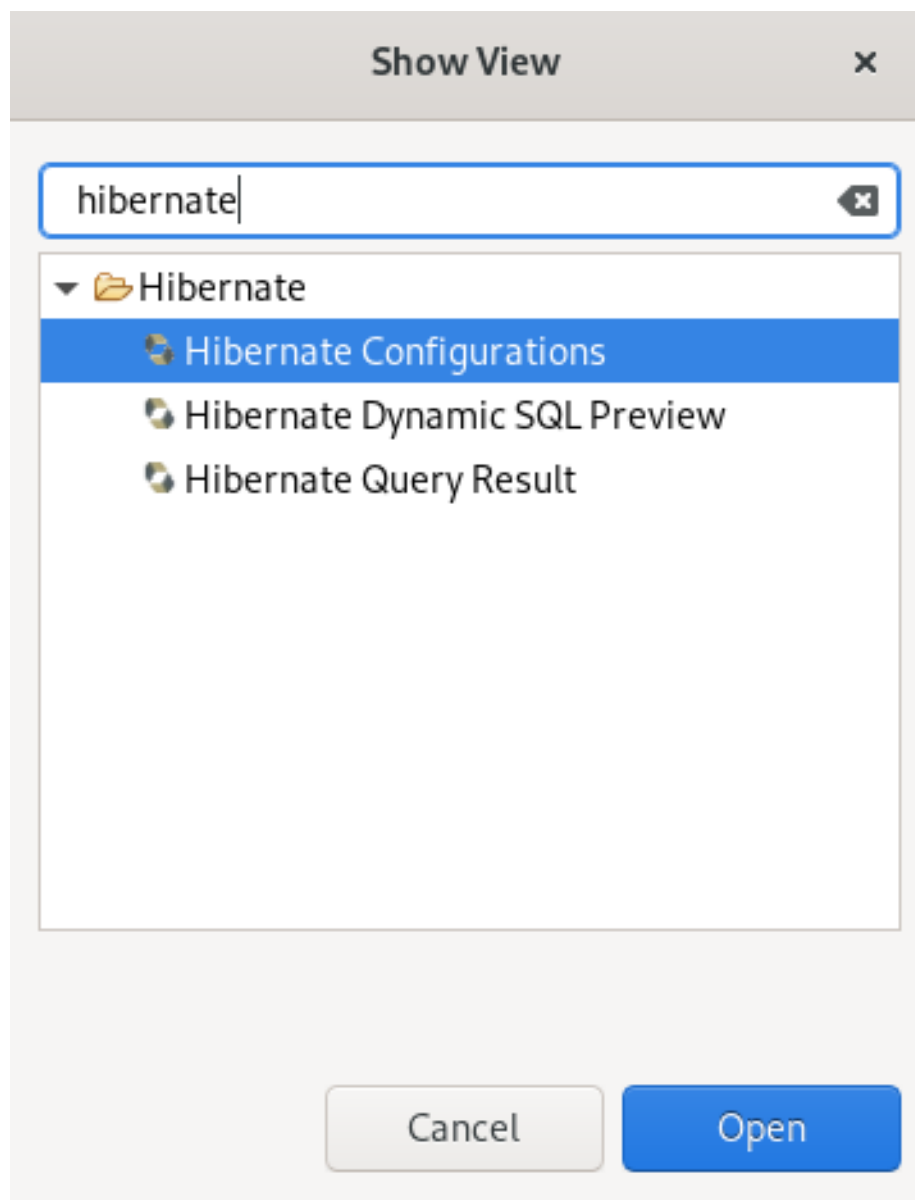
The following section describes how to edit configurations for your Hibernate project in CodeReady Studio.

### Procedure

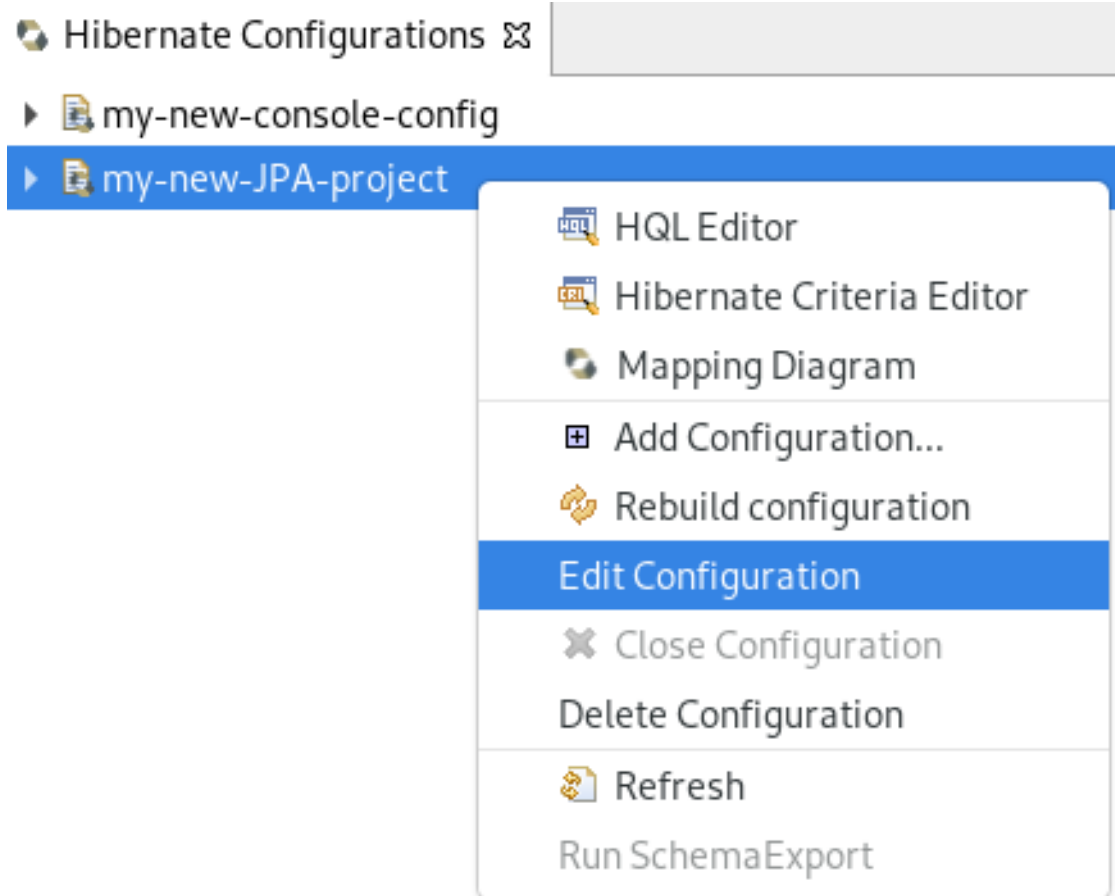
1. Start CodeReady Studio.
2. Click **Window** → **Show View** → **Other**.



The **Show View** window appears.



3. Enter **Hibernate** in the search field.
4. Select **Hibernate Configurations**.
5. Click **Open**.  
The **Hibernate Configurations** view appears.



6. Right-click your **project** → **Edit Configuration**.  
The **Edit Configuration** window appears.

Edit Configuration ✕

### Edit launch configuration properties

Select or configure a Console Configuration ▶

Name:

Main Options Classpath Mappings Common

Type:

Core  Annotations (jdk 1.5+)  JPA (jdk 1.5+)

Hibernate Version:  ▼

Project:

Browse...

Database connection:

▼ New... Edit...

Property file:

Setup...

Configuration file:

Setup...

Persistence unit:

Browse...

Revert Apply

Cancel OK

7. Edit your configurations.
8. Click **Apply**.

132



9. Click **OK**.

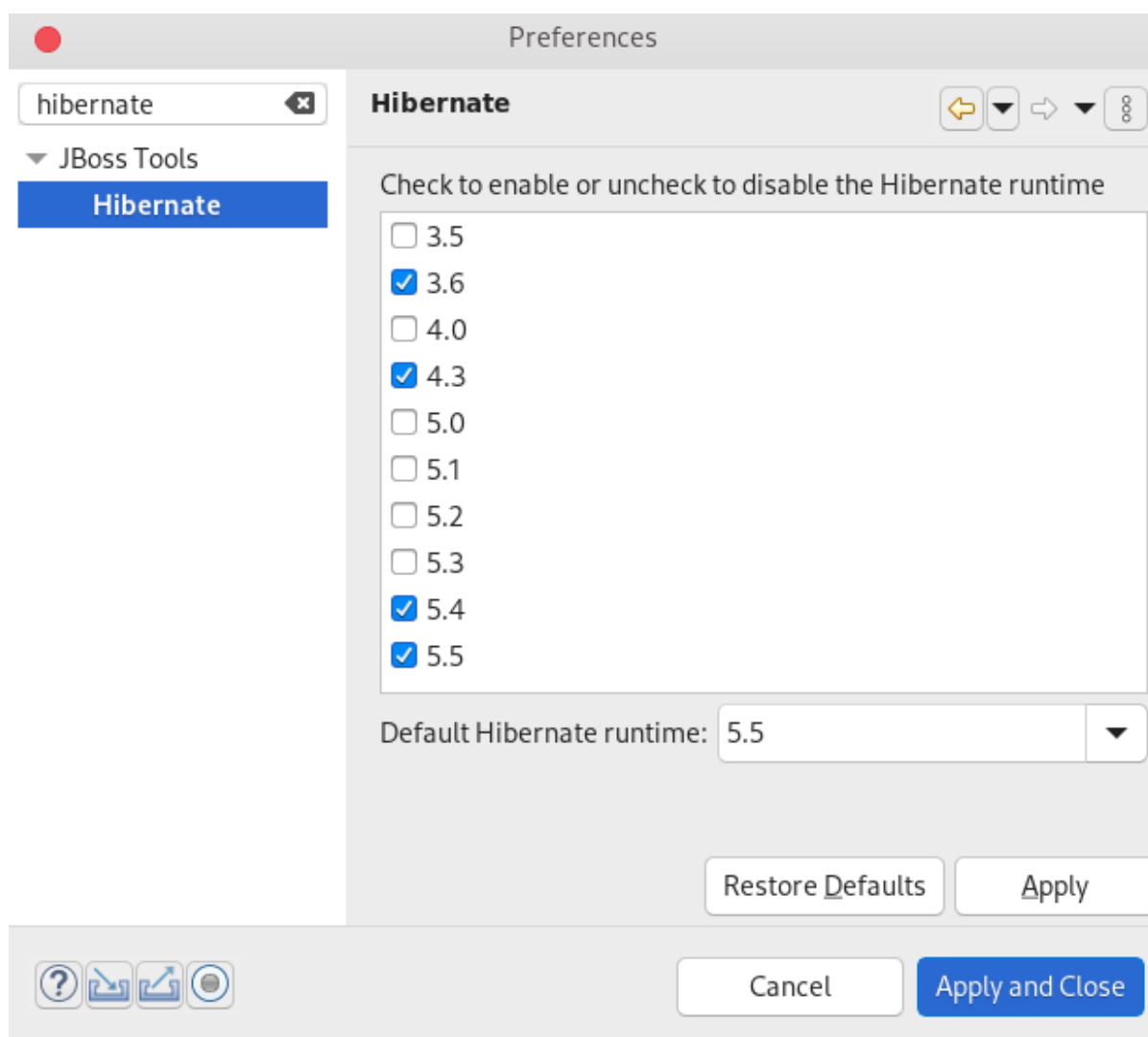
## 7.8. SETTING PREFERENCES FOR HIBERNATE RUNTIME IMPLEMENTATIONS

You can enable or disable runtime implementations in the Hibernate preference page.

The following section describes how to configure your runtime preferences.

### Procedure

1. Click **Window** → **Preferences**.  
The preference page opens.



2. Enter **Hibernate** in the search field.
3. Select **Hibernate**.

Check to enable or uncheck to disable the runtimes for your Hibernate project.

## CHAPTER 8. MOBILE WEB TOOLS BASICS IN CODEREADY STUDIO

Mobile Web Tools provide an **HTML5 Project** wizard that enables you to create web applications optimized for mobile devices. The **HTML5 Project** wizard is a useful starting point for creating all new HTML5 web applications in CodeReady Studio. The wizard generates a sample ready-to-deploy HTML5 mobile application with REST resources from a Maven archetype.

You can customize the application using the built-in editor, and deploy and view the application with the built-in browser.

CodeReady Studio provides the **Mobile Web** palette that allows the user to make interactive web applications. This palette offers a wide range of features including drag-and-drop widgets for adding common web interface framework features such as HTML5, jQuery Mobile, and Ionic tags to html files. It also contains widgets like **Panels**, **Pages**, **Lists**, and **Buttons** to make your applications more user friendly and efficient.

### 8.1. CREATING AN HTML5 PROJECT

The **HTML5 Project** wizard generates a sample project based on a Maven archetype and the project and application identifiers provided by you. The Maven archetype version is indicated in the **Description** field on the first page of the wizard. You can change the version, and therefore the project look and dependencies, by selecting either an enterprise or non-enterprise target runtime within the wizard.

The following section describes how to create an HTML5 project in CodeReady Studio.

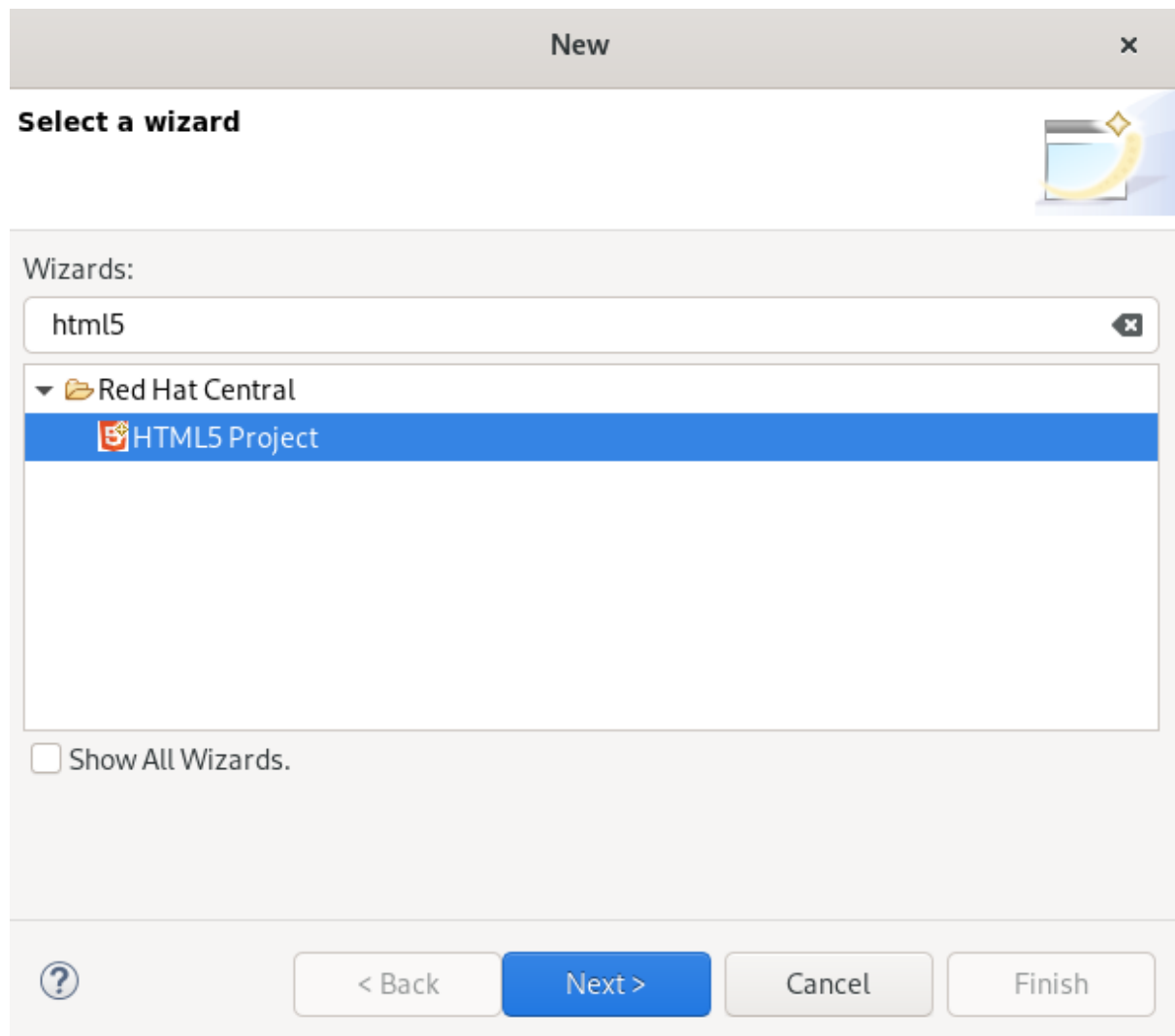
#### Prerequisites

- A configured local server.  
For information on configuring a local runtime server and deploying applications to it, see [Section 3.1, “Configuring a local server”](#).

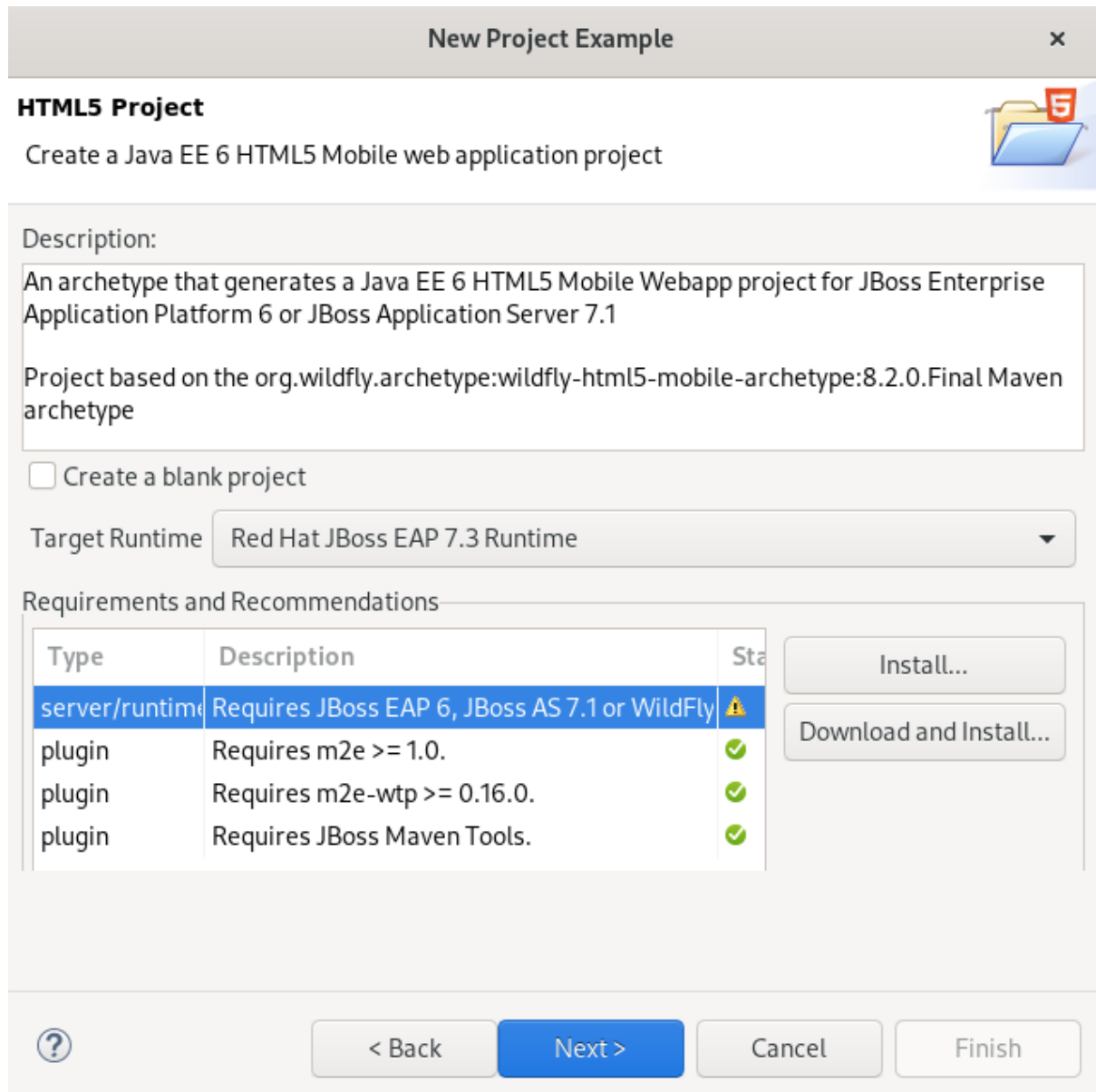
CodeReady Studio must be configured for any servers to which you want to deploy your application, including the location and type of the application server and any custom configuration or management settings.

#### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



3. Enter **HTML5** in the search field.
4. Select **HTML5 Project**.
5. Click **Next**.  
The **New Project Example** window appears.



6. Click the down-arrow in the **Target Runtime** field.
7. Select your server.
8. Click **Next**.
9. Name your project and your package.
10. Select the location for your project.
11. Click **Finish**.  
Note that it might take some time for the project to generate.

The **New Project Example** window appears.

12. Click **Finish**.

Your newly created project is now listed in the **Project Explorer** view.

## 8.2. ADDING A NEW HTML5 JQUERY MOBILE FILE

The HTML5 **jQuery Mobile** file template consists of JavaScript and CSS library references that are inserted in the file's HTML header. The template also inserts a skeleton of the **jQuery Mobile** page and **listview** widgets in the file's HTML body.

The following section describes how to add a new HTML5 jQuery Mobile file to an existing project.

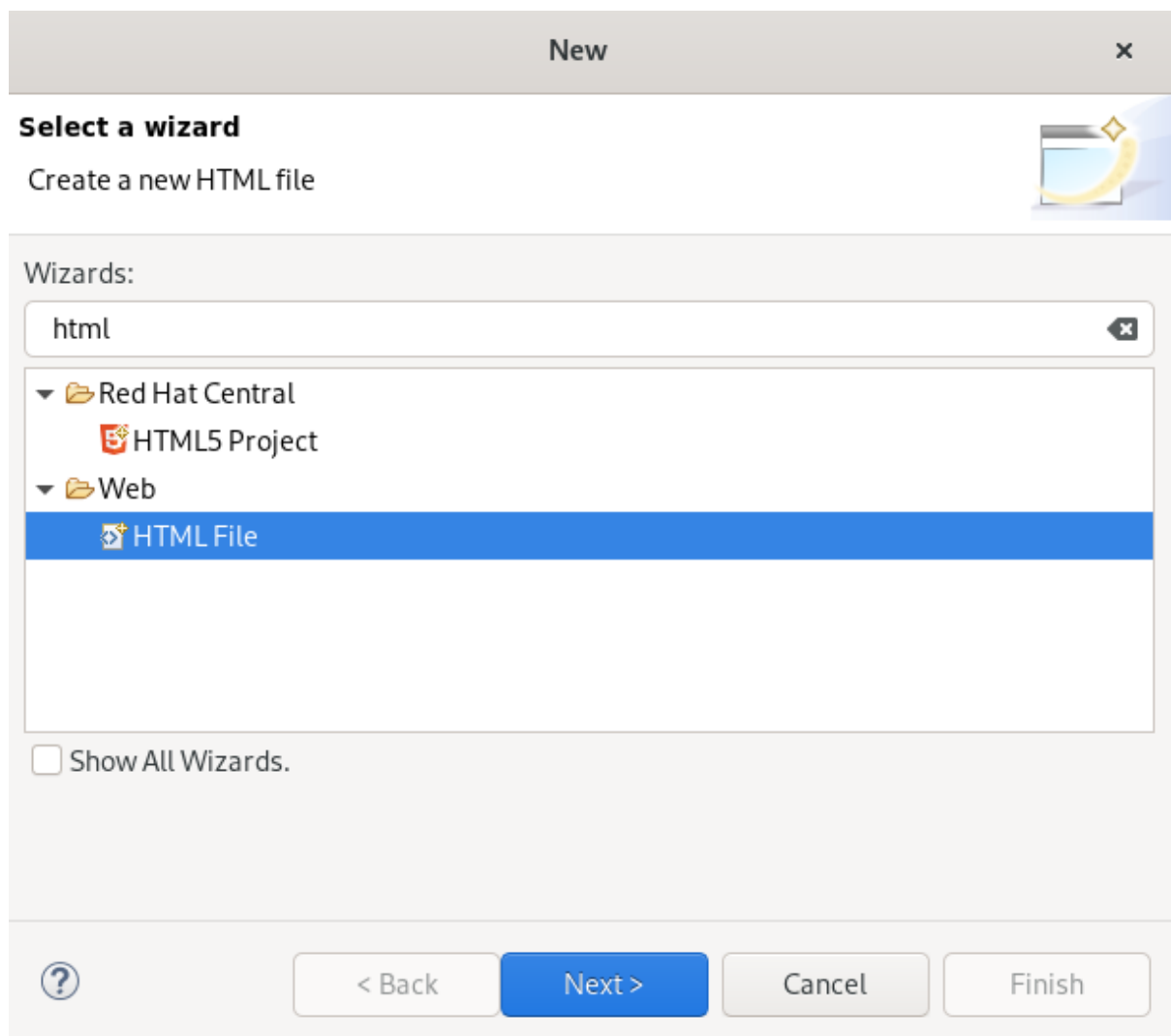
### Prerequisites

- A configured server.  
For information on configuring a local runtime server and deploying applications to it, see [Section 3.1, "Configuring a local server"](#).

CodeReady Studio must be configured for any servers to which you want to deploy your application, including the location and type of the application server and any custom configuration or management settings.

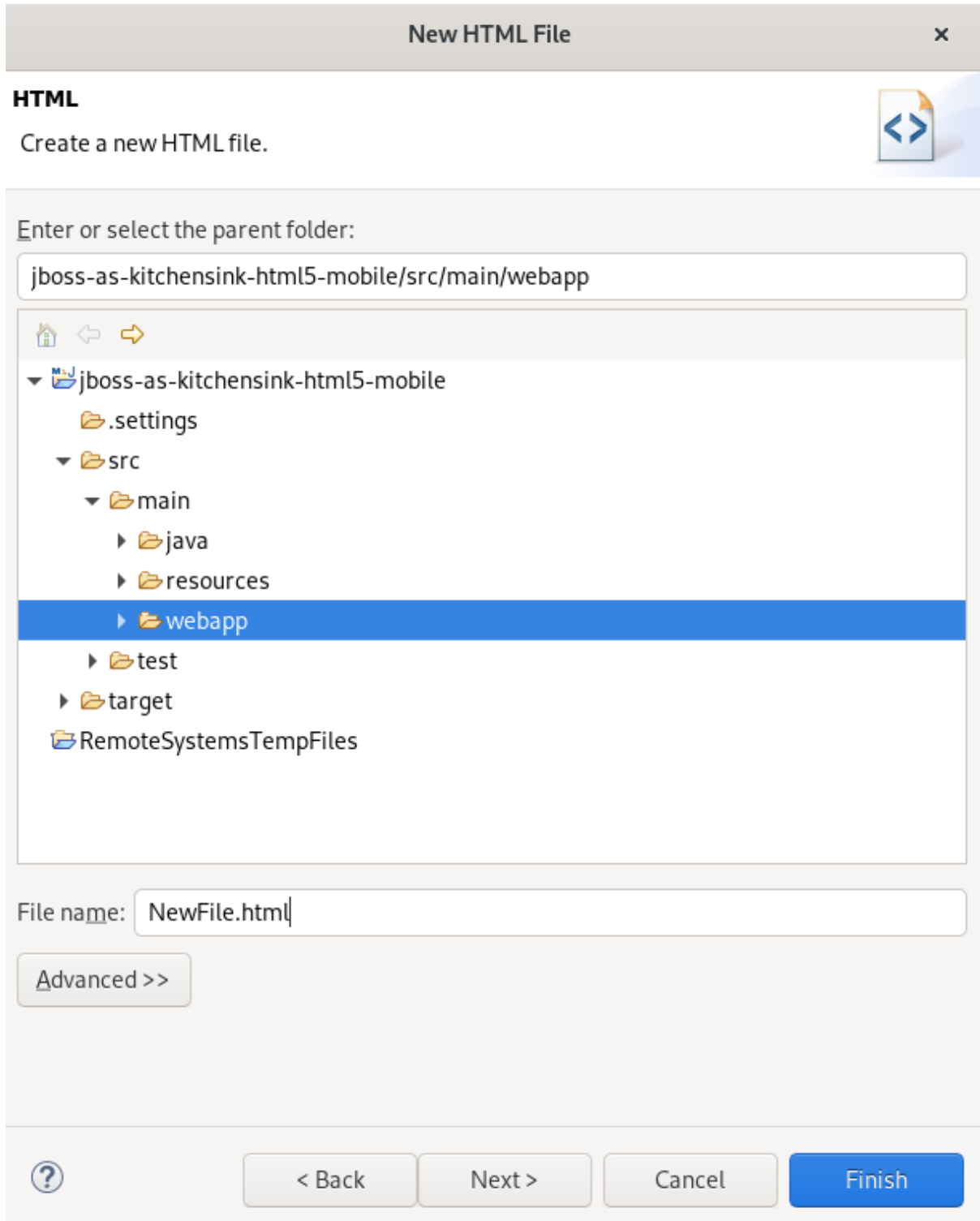
### Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.  
The **Select a wizard** window appears.



3. Enter **HTML** in the search field.

4. Select **HTML File**.
5. Click **Next**.  
The **New HTML File** window appears.



6. Select the location for your file.
7. Name your file.
8. Click **Next**.  
The **Select HTML Template** window appears.

New HTML File ✕

### Select HTML Template

Select a template as initial content in the HTML page.

Use HTML Template

Templates:

Name	Description
Facelets XHTML Page	Facelets XHTML Page Template
HTML5 jQuery Mobile Page (1.3)	HTML5 jQuery Mobile 1.3 Template
HTML5 jQuery Mobile Page (1.4)	HTML5 jQuery Mobile 1.4 Template
New Facelet Composition Page	Creates a new Facelet page for use with a tem
New Facelet Footer	Creates a footer for use with the Facelet templ
New Facelet Header	Creates a header for use with the Facelet temp
New Facelet Template	Creates a basic header/content/footer Facelet
New HTML File (4.01 frameset)	html 4.01 frameset
New HTML File (4.01 strict)	html 4.01 strict
New HTML File (4.01 transitional)	html 4.01 transitional
New HTML File (5)	html 5
New XHTML File (1.0 frameset)	xhtml 1.0 frameset
New XHTML File (1.0 strict)	xhtml 1.0 strict
New XHTML File (1.0 transitional)	xhtml 1.0 transitional

Preview:

```

<!DOCTYPE html>
<html>
<head>
  <title>jQuery Mobile Template</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="viewport"
    content="width=device-width, initial-scale=1" />

```

Templates are 'New HTML' templates found in the [HTML Templates](#) preference page.

?
< Back
Next >
Cancel
Finish

9. Select a template.
10. Click **Finish**.

The newly created HTML file is now displayed in the CodeReady Studio editor.

## 8.3. ADDING A NEW MOBILE PAGE

The following section describes how to add a new jQuery Mobile Page to an existing web application.

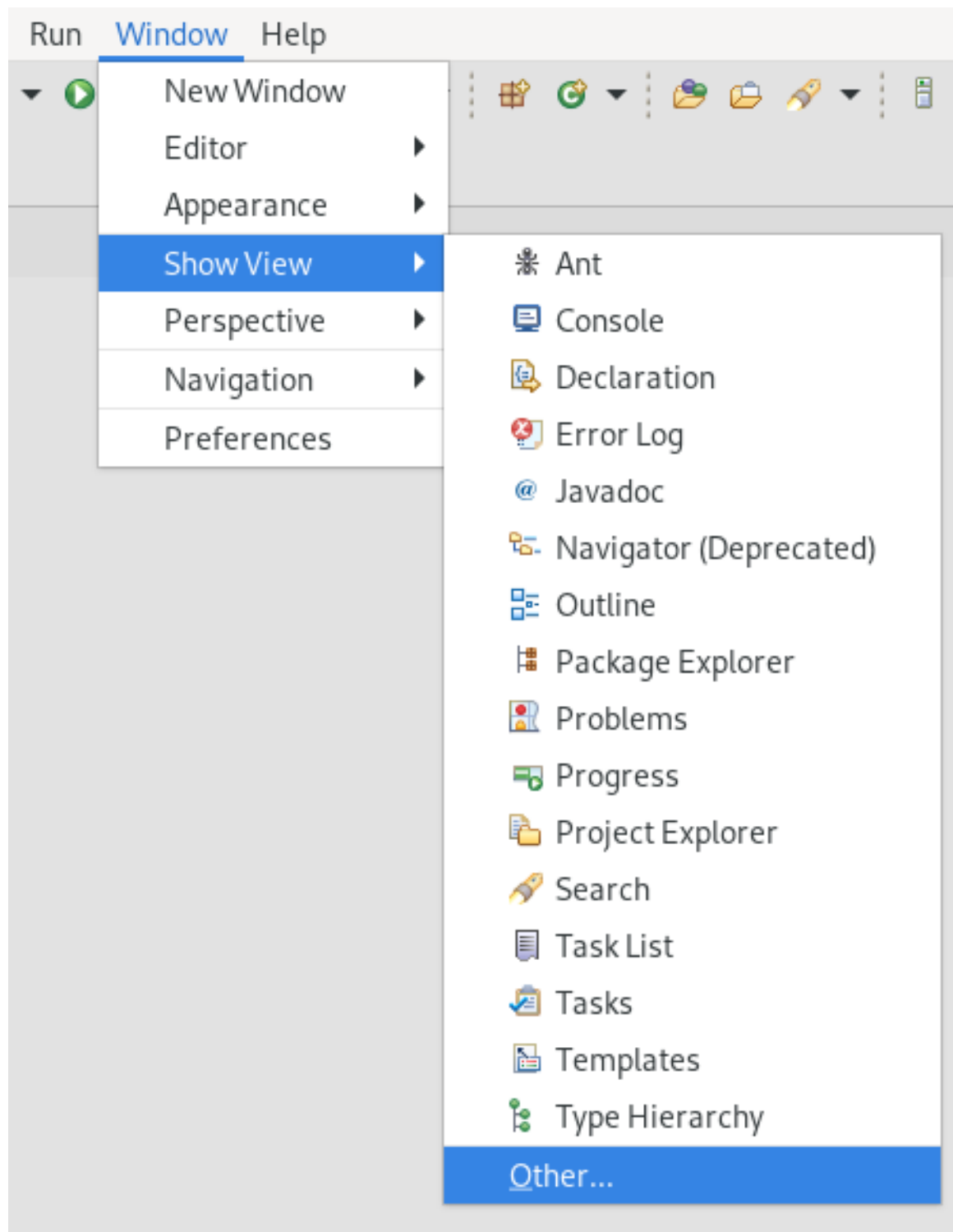
### Prerequisites

- An HTML5 project.  
For more information on how to create an HTML5 project, see [Section 8.1, “Creating an HTML5 Project”](#).

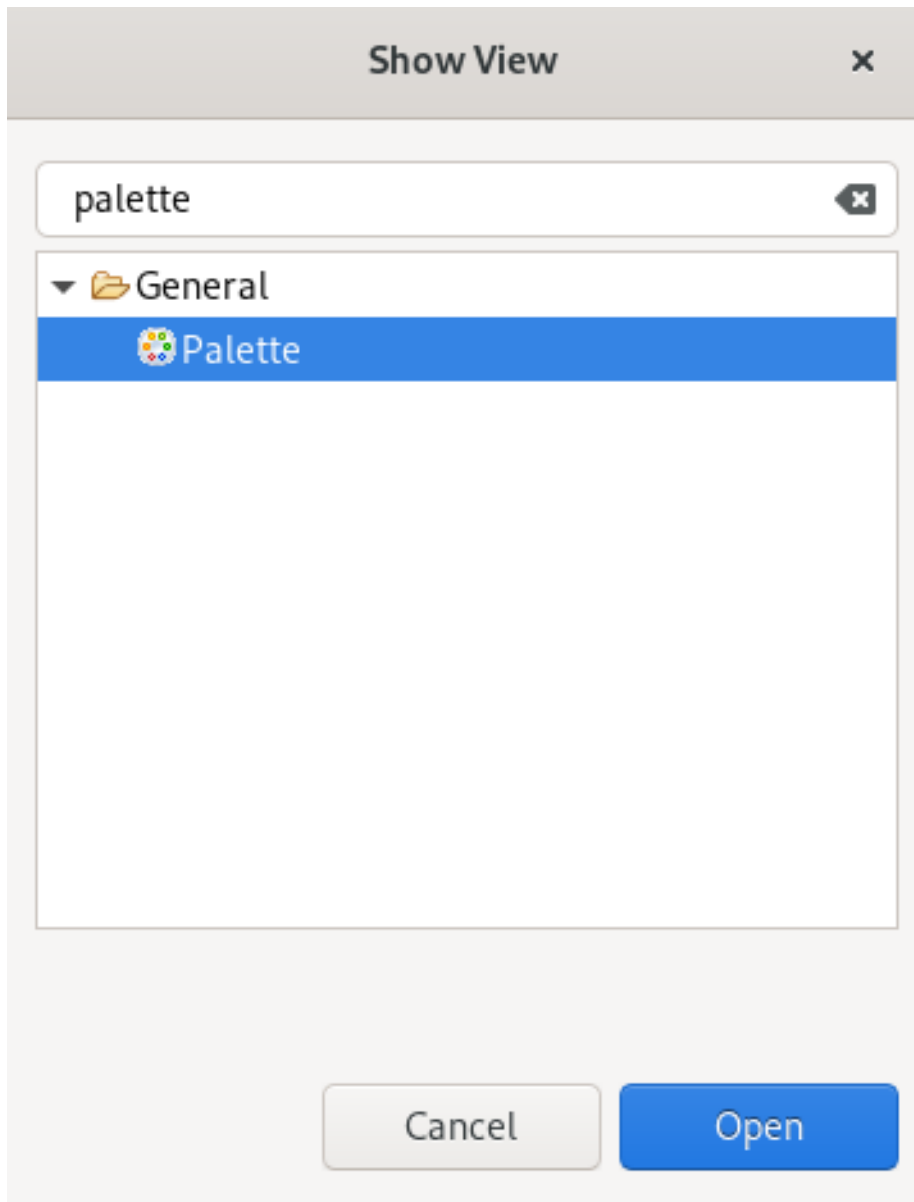
### Procedure

1. Start CodeReady Studio.
2. Click **Window** → **Show view** → **Other**.

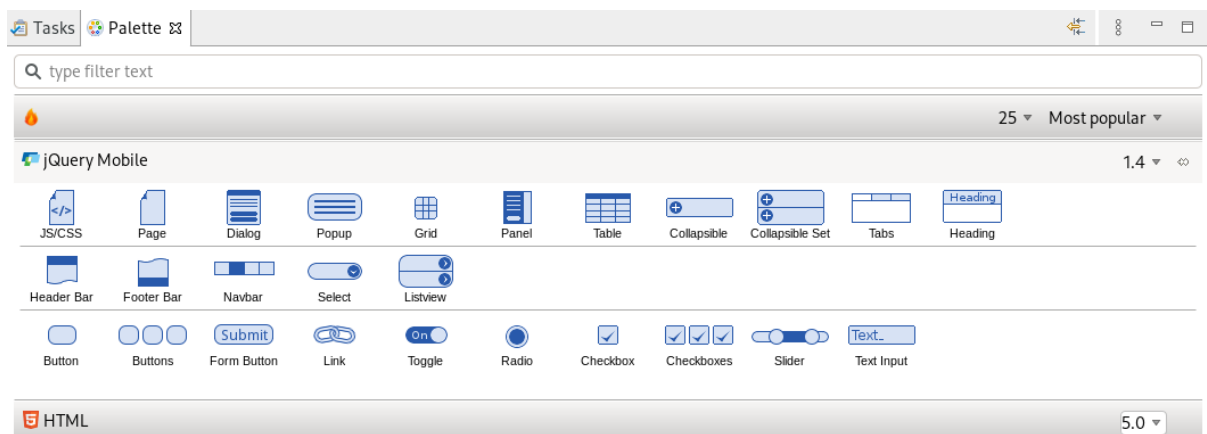




The **Show View** window appears.



3. Enter **Palette** in the search field.
4. Select **Palette**.
5. Click **Open**.  
The **Palette** view appears.



6. Click the **Page** icon.

The **Insert Tag** window appears.

7. Name the **Header**.
8. Name the **Footer**.
9. Click **Finish**.

Your newly added page is now displayed in the CodeReady Studio editor.



## NOTE

You can use the same workflow to customize the pages of your web application by selecting widgets from the **Palette** view.