Red Hat Developer Studio 12.9

Release Notes and Known Issues

Highlighted features in 12.9

Last Updated: 2018-10-04

Red Hat Developer Studio 12.9 Release Notes and Known Issues

Highlighted features in 12.9

Supriya Takkhi
sbharadw@redhat.com

Legal Notice

Abstract

This document lists and briefly describes new and improved features of Red Hat Developer Studio 12.9.

Table of Contents

CHAPTER 1. INTRODUCTION TO RED HAT DEVELOPER STUDIO

1.1. ABOUT RED HAT DEVELOPER STUDIO

Developer Studio is a set of Eclipse-based development tools. It contains plug-ins that integrate with Eclipse to extend the existing functionality of the integrated development environment (IDE).

Developer Studio is designed to increase your productivity when developing applications. You can focus on building, testing, and deploying your applications because JBoss application development tools are integrated in one IDE. Developer Studio can also assist your application development with its unique features in the following ways:

- Develop new applications using the wizards and project examples of Red Hat Central
- Add powerful functionality to applications with minimal effort using Forge Tools
- Build web interfaces with ease using the visual editing and drag-and-drop utilities of Visual Web Tools and Mobile Web Tools
- Incorporate Hibernate, CDI, JAX-RS, JSF, and other popular APIs into applications with simplicity using the tool-driven interface
- Deploy applications to JBoss runtime servers and the cloud using JBoss Server Tools and OpenShift Tools

Developer Studio is built around Eclipse and packaged with all the necessary dependencies and third-party plug-ins for simplified installing. For developers already running Eclipse, Developer Studio can also be installed through Eclipse Marketplace. Installing Developer Studio in an existing Eclipse installation is referred to as BYOE (Bring Your Own Eclipse).

1.2. USE CASES OF DEVELOPER STUDIO

Developer Studio assists Java EE developers by integrating JBoss technology and APIs in a single development environment. Here are a few ways that Developer Studio helps make development easier:

1.2.1. Web Applications

Red Hat Central provides wizards that generate skeletons and sample projects, enabling you to focus on developing the functionality of your applications. The wizards create web applications based on different APIs and technologies, showing the usage and advantages of each. Developer Studio also offers project file templates in a range of popular programming languages, including HTML, XHTML, and JSF.

Palettes in Developer Studio give access to the core elements of the JSF and RichFaces APIs, for use in developing the user interfaces of your applications. Elements of these APIs can be dragged and dropped directly into your project so that you can create richer user interfaces quickly. Visual Web Tools offers graphical and source viewing of files and defaults to dedicated editors for different file types.
Developer Studio supports the Java EE specification and provides tools for JAX-RS, Hibernate, and CDI APIs so you can develop the server-side components of your applications effortlessly.

LiveReload Tools automatically refreshes browsers of local or deployed applications as you modify project resources to avoid needing to manually refresh. You can experience automatic refreshing when viewing applications in browsers on external and mobile devices, with application web addresses easy to navigate to with QR codes.

1.2.2. Web Applications Optimized for Mobile Devices

Mobile Web Tools provides support for HTML5 and jQuery Mobile to enable you to create web applications optimized across desktop and mobile clients. The HTML5 Project wizard in Red Hat Central generates a sample application using HTML5 and jQuery Mobile technologies and, together with HTML5 and jQuery Mobile project file templates, helps you to get up and running with these APIs and technologies quickly. HTML5 and jQuery Mobile widgets can be dragged from the jQuery Mobile palette into your project files and, in conjunction with the widget wizards, enable you to effortlessly develop customized user interfaces for your mobile web applications.

1.2.3. Applications for Cloud Deployment

OpenShift Tools deploys your applications directly to the cloud on the Red Hat OpenShift platform. You can create and manage your OpenShift account and manage the deployment of applications to OpenShift within the IDE. In addition to using the OpenShift Application wizard to create and deploy new OpenShift applications, OpenShift Tools can import applications already deployed on OpenShift so that you can further develop them and manage their deployment from the comfort of the IDE.

1.3. ABOUT THIS RELEASE

Red Hat Developer Studio 12.9 is an update of Red Hat Developer Studio 12.0 and it has the following features:

- It includes Eclipse SimRel 2018-09 GA.
- It requires a minimum of Java 8 to run.
- It introduces new features, which are outlined in the Release Notes section.
- It contains new features for the existing tools.
- It resolves issues identified in earlier versions of Developer Studio.

For more information about operating systems, chip architectures and Java developer kits supported by this release, see Supported Configurations and Components page on the Red Hat Customer Portal.

1.4. ECLIPSE AUTOMATED REPORTING INTERFACE (AERI)

To contribute to JBoss Tools, we recommend you to enable the Eclipse Automated Reporting Interface (AERI) in JBoss Tools. To read about configuring error reporting in JBoss Tools, see: http://tools.jboss.org/usage/#error-reporting.

CHAPTER 2. RELEASE NOTES

2.1. FORGE TOOLS

2.1.1. Forge Runtime updated to 3.9.1.Final

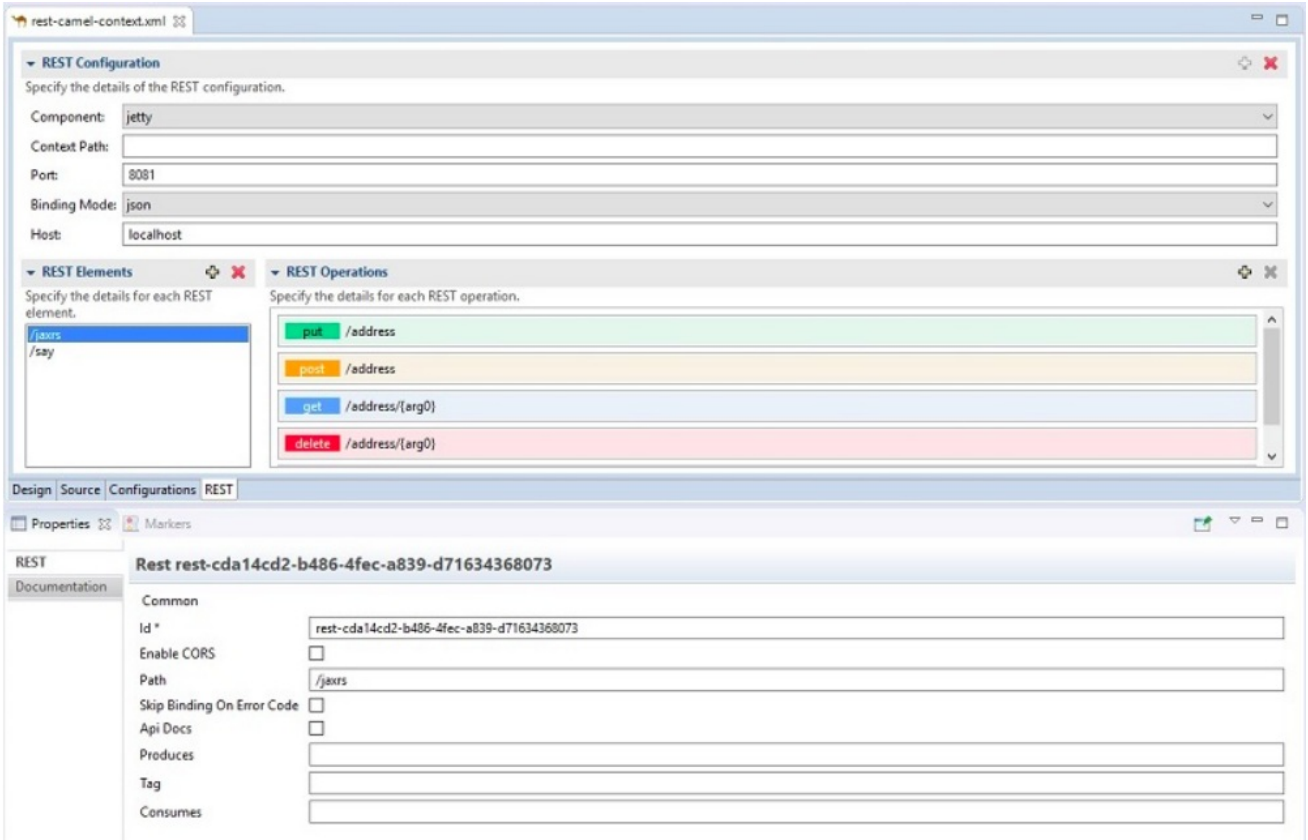
The included Forge runtime is now 3.9.1.Final. Read the official announcement here.

2.2. FUSE TOOLING

2.2.1. REST Viewer becoming an editor

Previously, there was a REST editor that was read-only. It is useful to have a great overview of the already defined Camel REST DSL definitions. Now the editor and its related properties tab also provide editing capabilities allowing you to develop faster.

Figure 2.1. REST Viewer Editor



You can now:

- Create and delete REST Configurations
- Create and delete new REST Elements
- Create and delete new REST Operations
- Edit properties for a selected REST Element in the Properties view
- Edit properties for a selected REST Operation in the Properties view

The scrolling capabilities of the REST Element and REST Operations lists are improved.

2.3. HIBERNATE TOOLS

2.3.1. Hibernate Runtime Provider Updates

A number of additions and updates have been performed on the available Hibernate runtime providers.

2.3.2. New Hibernate 5.3 Runtime Provider

The Hibernate 5.3 runtime provider now incorporates Hibernate Core version 5.3.3.Final and Hibernate Tools version 5.3.3.Final.

2.3.3. Other Runtime Provider Updates

The Hibernate 5.1 runtime provider now incorporates Hibernate Core version 5.1.15.Final and Hibernate Tools version 5.1.9.Final.

The Hibernate 5.2 runtime provider now incorporates Hibernate Core version 5.2.17.Final and Hibernate Tools version 5.2.11.Final.

2.4. OPENSHIFT

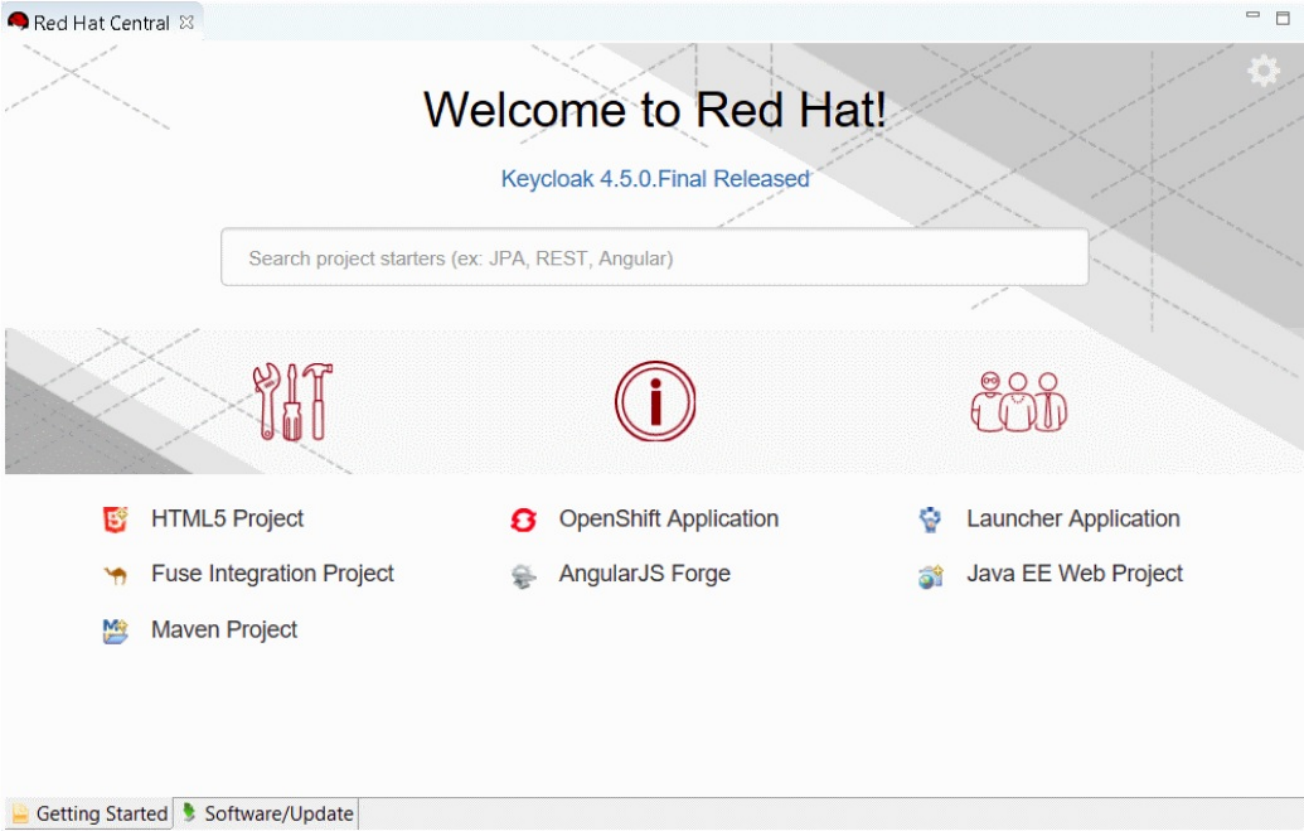
2.4.1. Inner loop for Spring Boot applications

Although Spring Boot applications were already supported by the OpenShift server adapter, the global developer experience has been enhanced. Following is the complete workflow.

2.4.1.1. Bootstrapping your Spring Boot applications

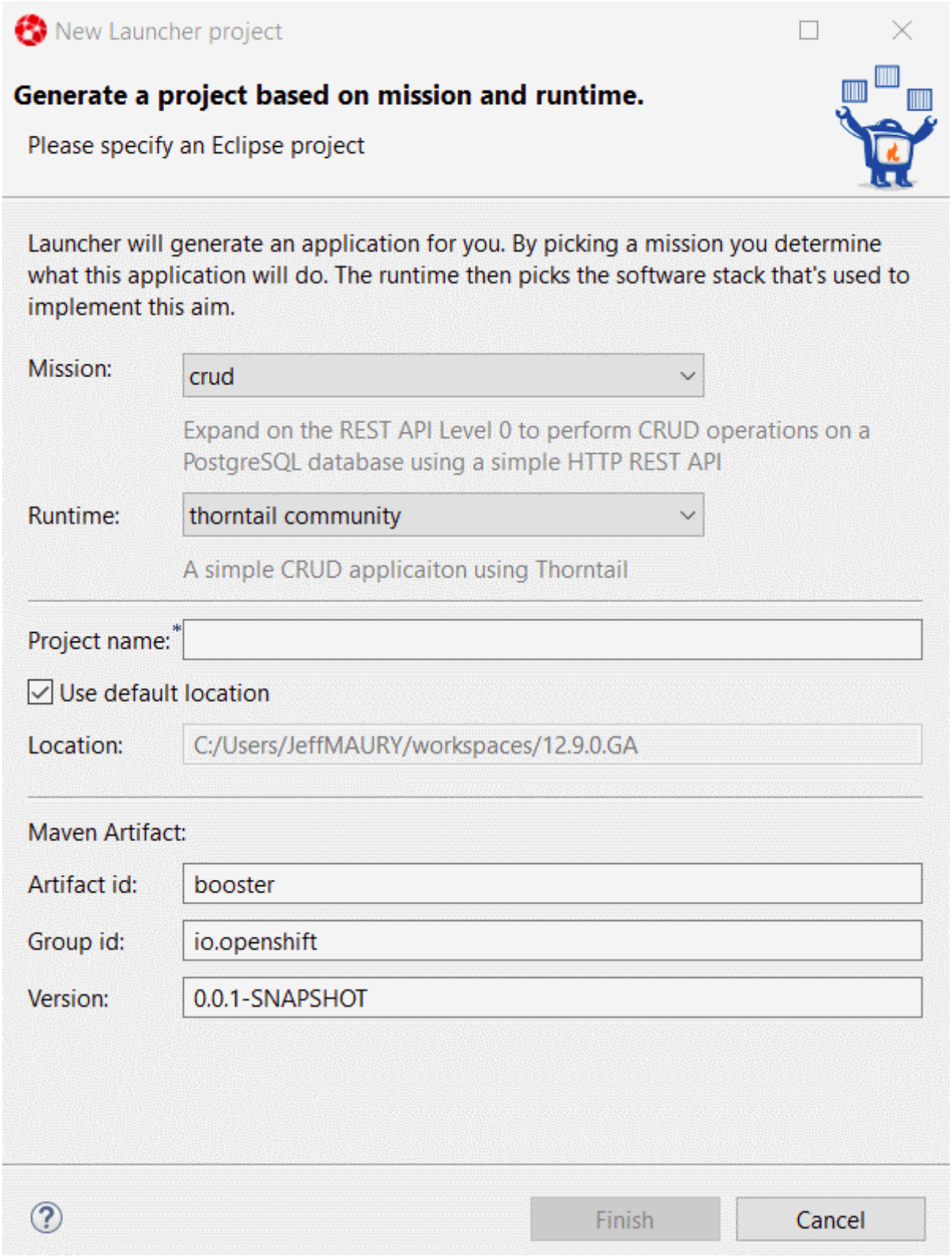
A new generator (wizard) **Launcher Application** has been added to JBoss Tools. It is based on the fabric8-launcher project. When you launch JBoss Tools, you should see the following in Red Hat Central.

Figure 2.2. Launcher Application Option



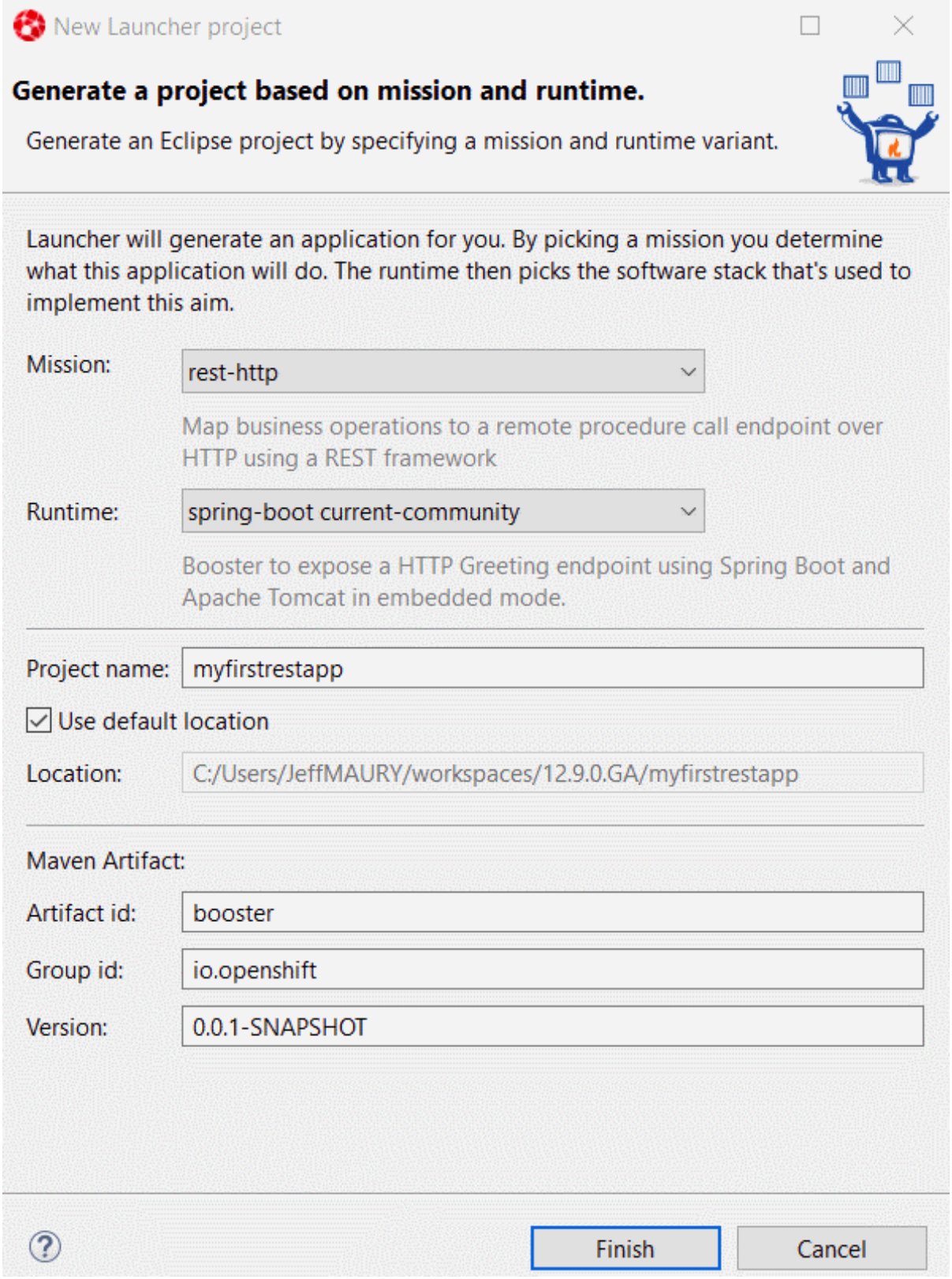
1. Click the **Launcher Application** link to display the **New Launcher project** wizard.

Figure 2.3. New Launcher project Wizard



2. In the **Mission** dropdown list, click **rest-http** to generate a simple REST application.
3. In the **Runtime** dropdown list, click **spring-boot current-community** to generate a Spring Boot based application.
4. In the **Project name** field, type *myfirstrestapp* and let the other fields be as-is.

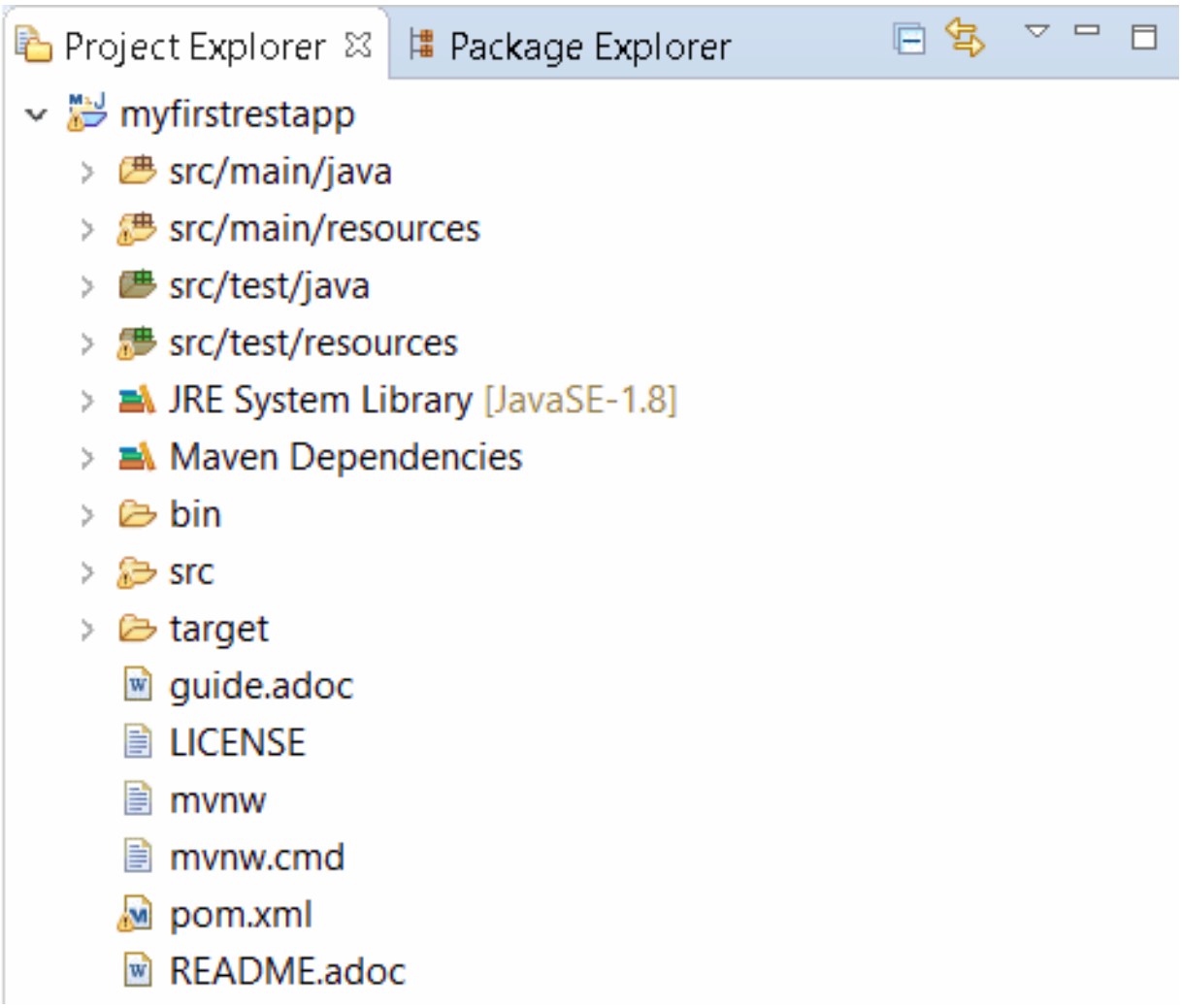
Figure 2.4. Setting the fields in the New Launcher project Wizard



5. Click **Finish**. A new project is added to your local workspace. This may take some time because Maven resolves all the Spring Boot dependencies by downloading them from the Internet.

When the project is built and you expand **myfirstrestapp** in the **Project Explorer** view, the following window must display.

Figure 2.5. Expanded myfirstrestapp application



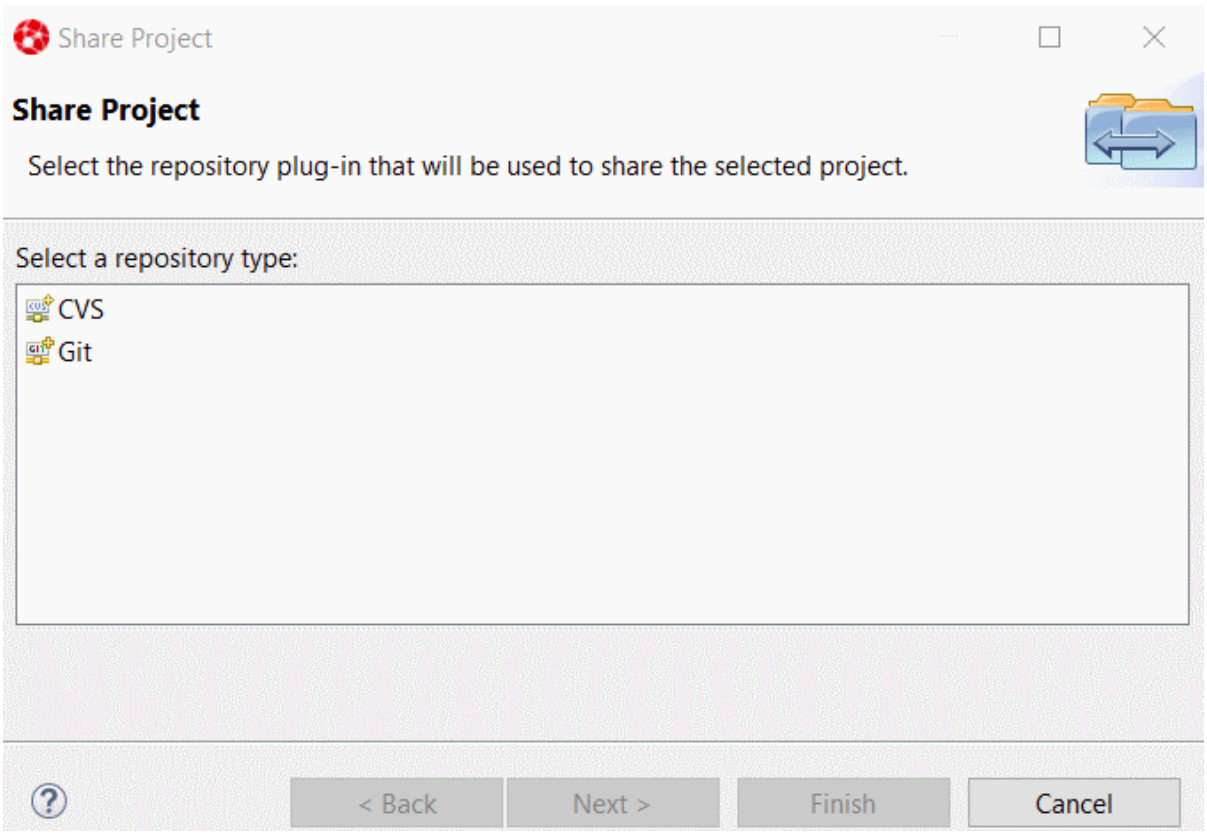
2.4.1.2. Storing your source code on GitHub

OpenShift builder image retrieves code from a Git repository. For this you must first push the code of the application that you generated to GitHub. The following section assumes that you created a repository called **myfirstrestapp** in your GitHub account.

You will first create a local Git repository for the application code then push it to GitHub.

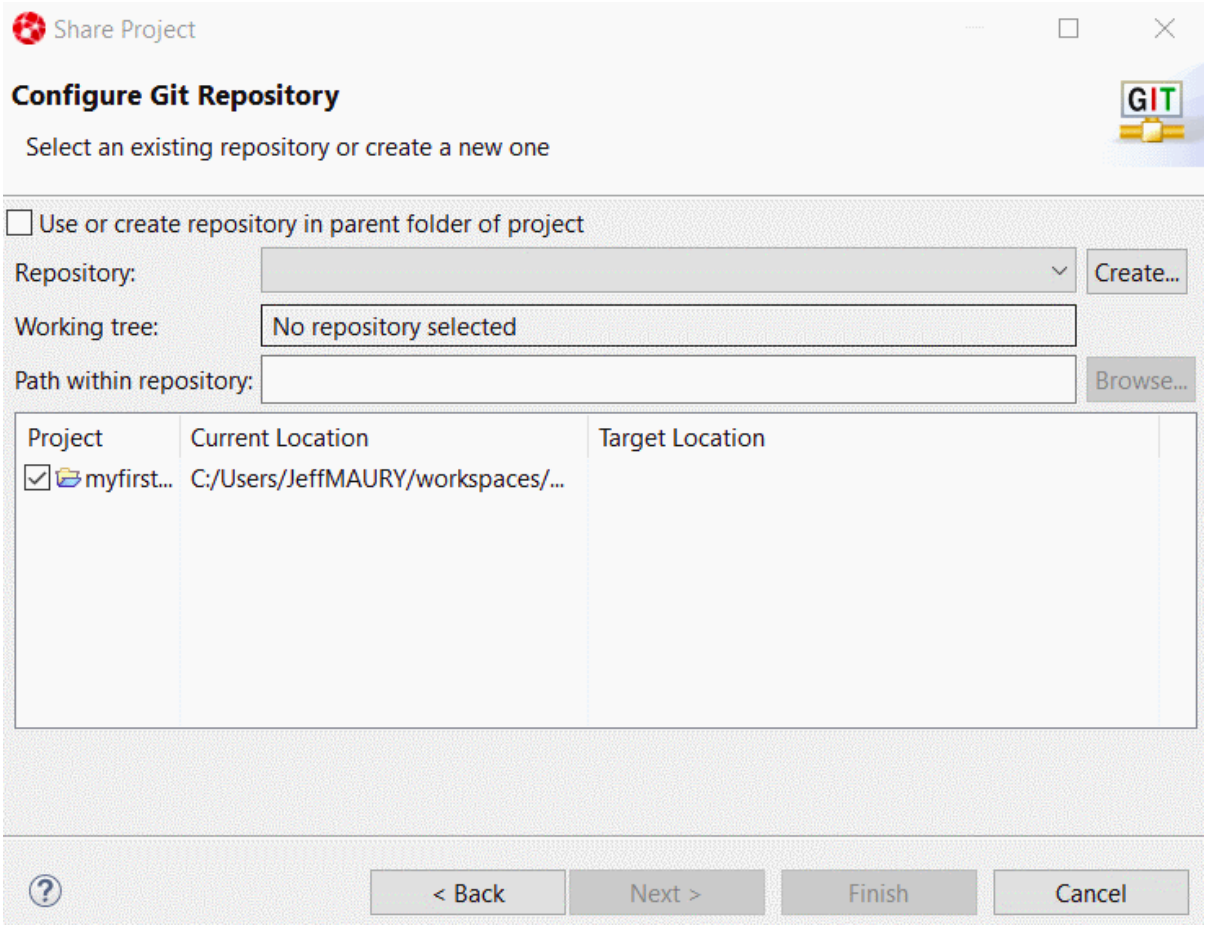
1. Click the **myfirstrestapp** project and right click **Team** → **Share project**.

Figure 2.6. Share Project



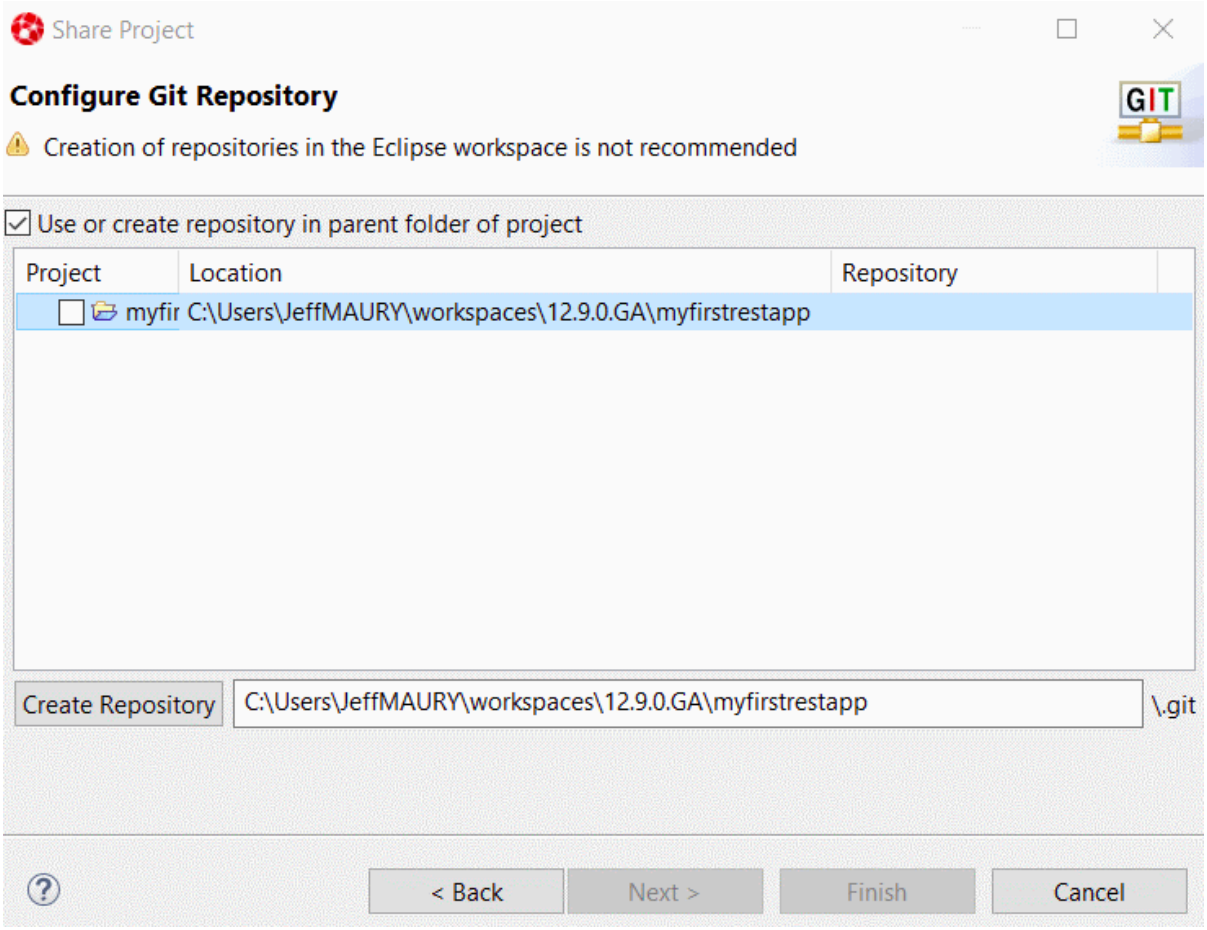
2. Select the Git repository type and click **Next**.

Figure 2.7. Configuring the Git Repository



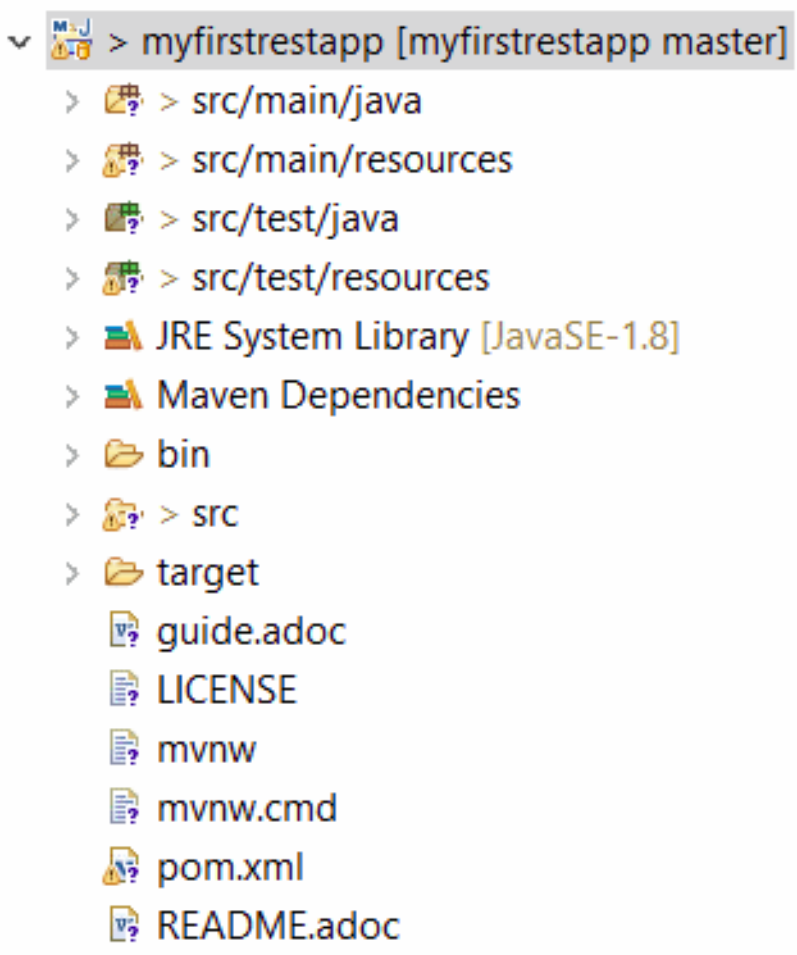
3. Click the **Use or create repository in parent folder of project** checkbox then select the **myfirstrestapp** project.

Figure 2.8. Additional Git Configuration



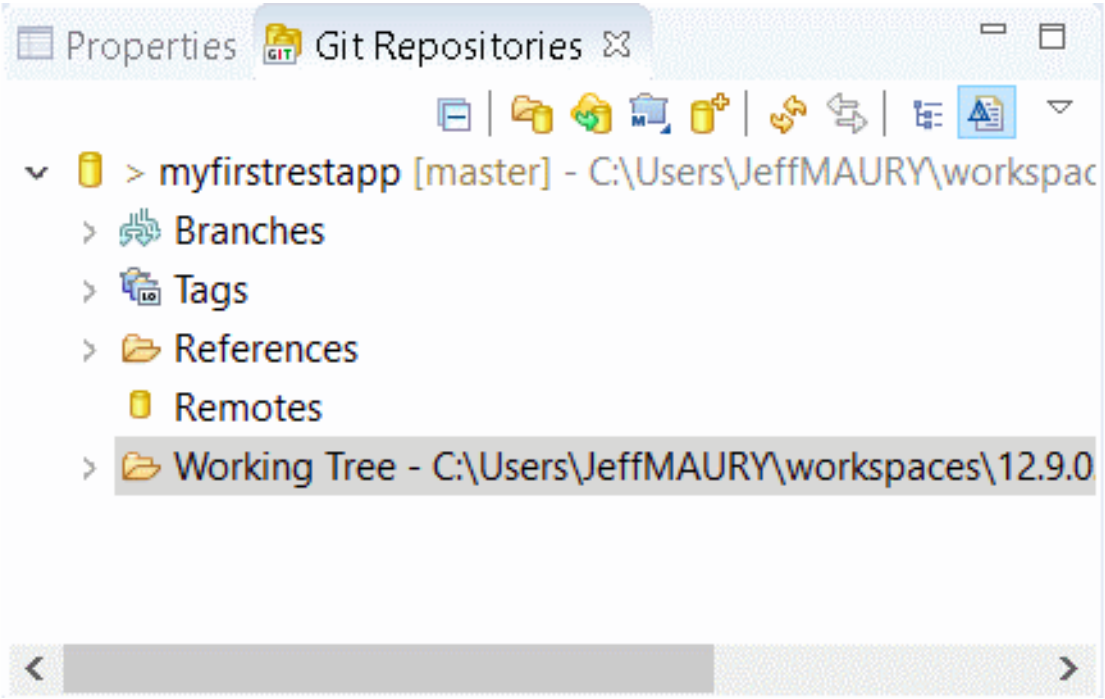
4. Click **Create Repository** and then click **Finish**. The **Project Explorer** view is updated.

Figure 2.9. Updated Project Explorer View



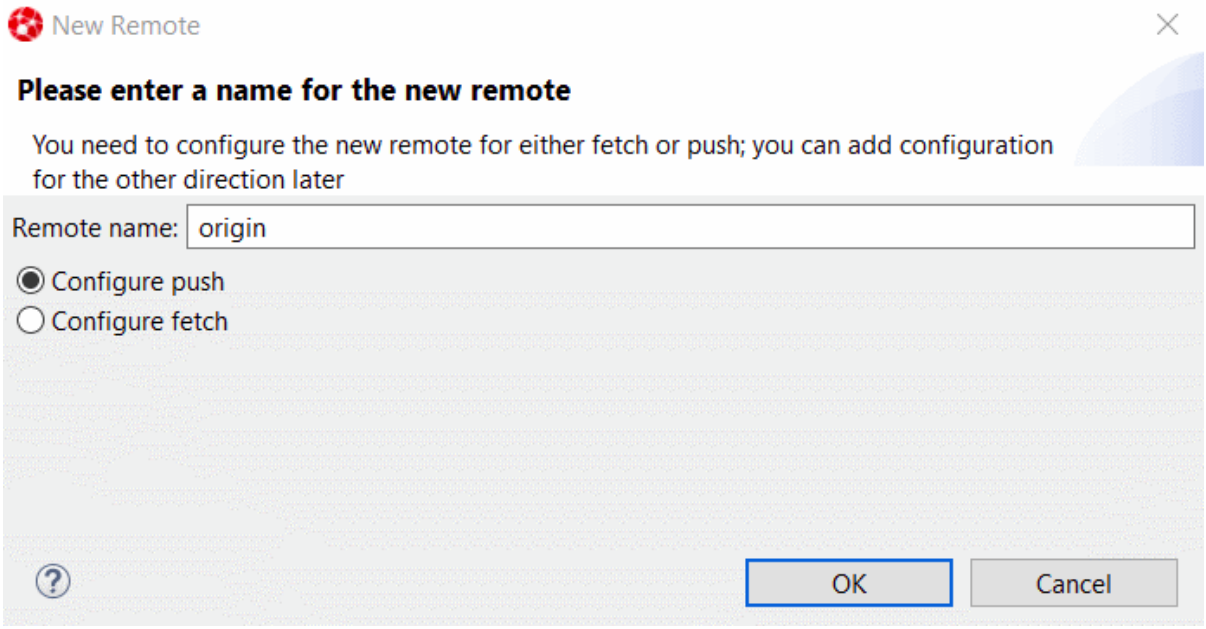
5. Click the **myfirstrestapp** project and right-click **Team** → **Show in Repositories View**. The new **Git Repositories** view is added to the perspective.

Figure 2.10. A new Git Repositories view added



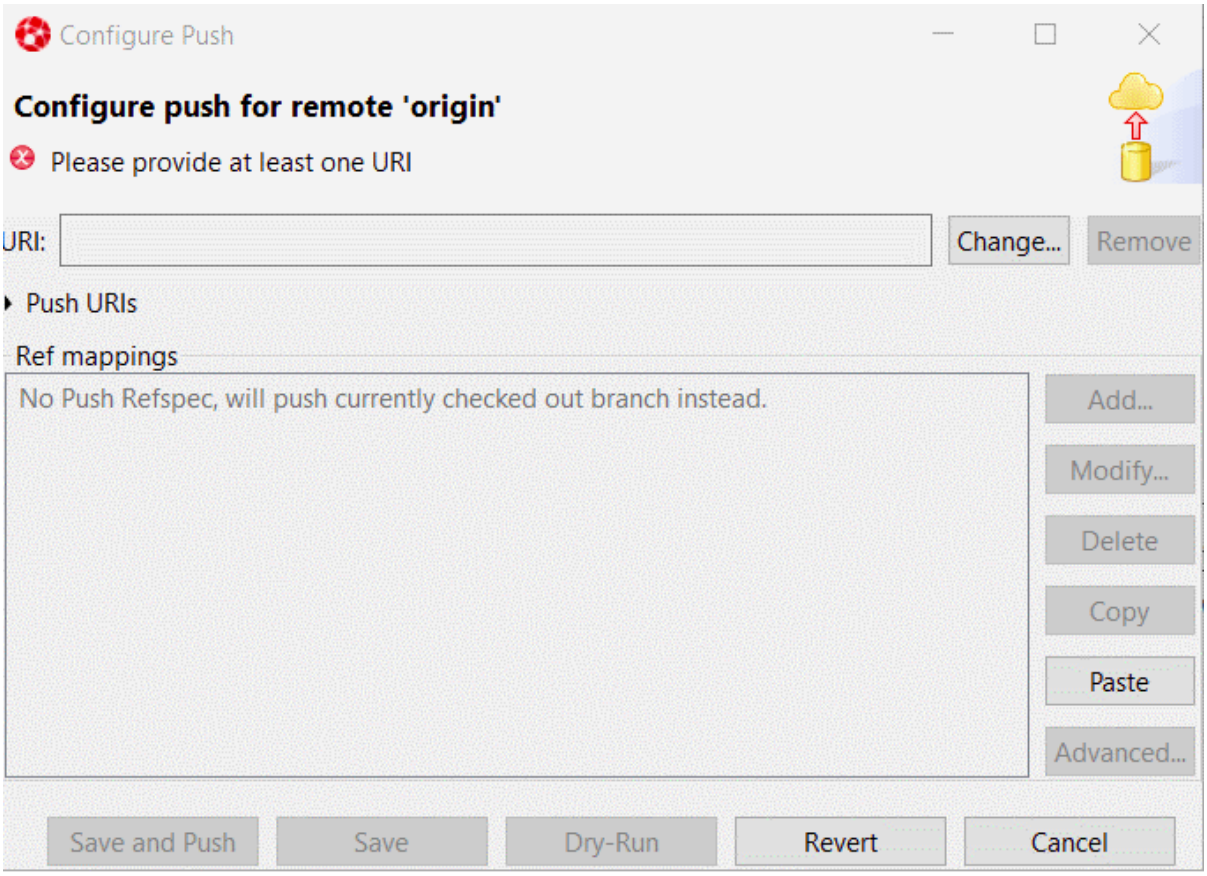
6. In the **Git Repositories** view, select the **Remotes** node and right-click **Create Remote**. The **New Remote** dialog box displays.

Figure 2.11. New Remote Dialog Box



7. Click **OK**, the **Configure Push** window displays.

Figure 2.12. The Configure Push window



8. Click **Change**. In the **URI** field, type **git@github.com:{*GITHUB-USER*}/myfirstrestapp** (replace GITHUB_USER with your GitHub user name).

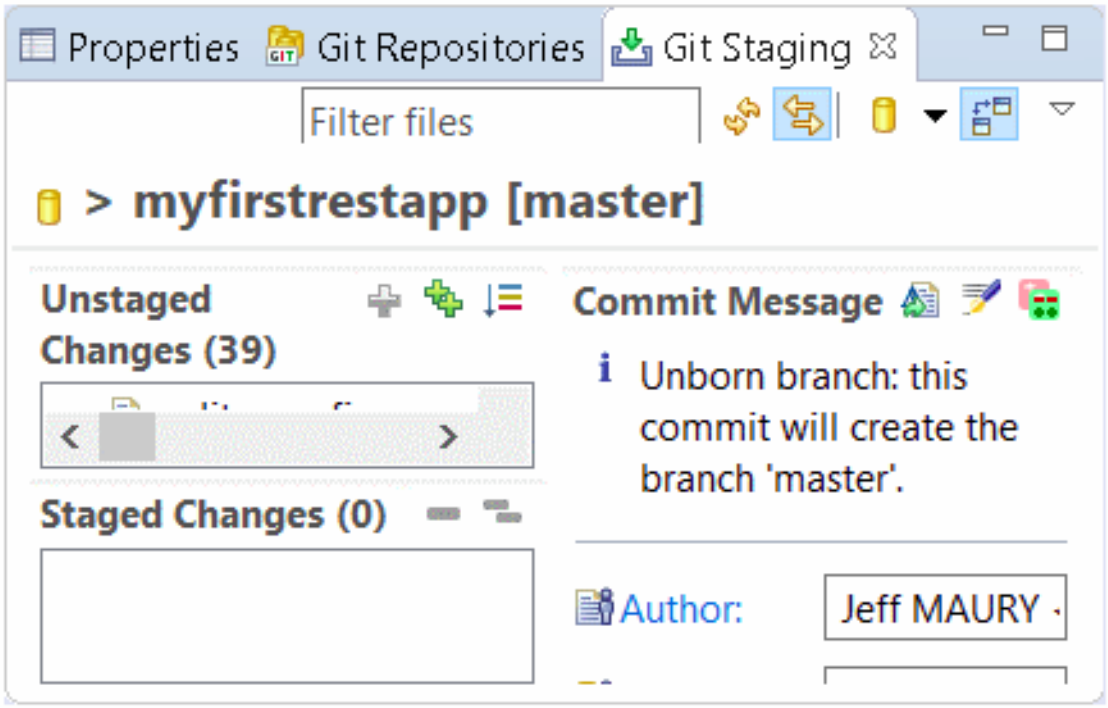
9. Click **Finish** and then click **Save**.

2.4.1.2.1. Pushing code to GitHub

You are now ready to push the application code to GitHub.

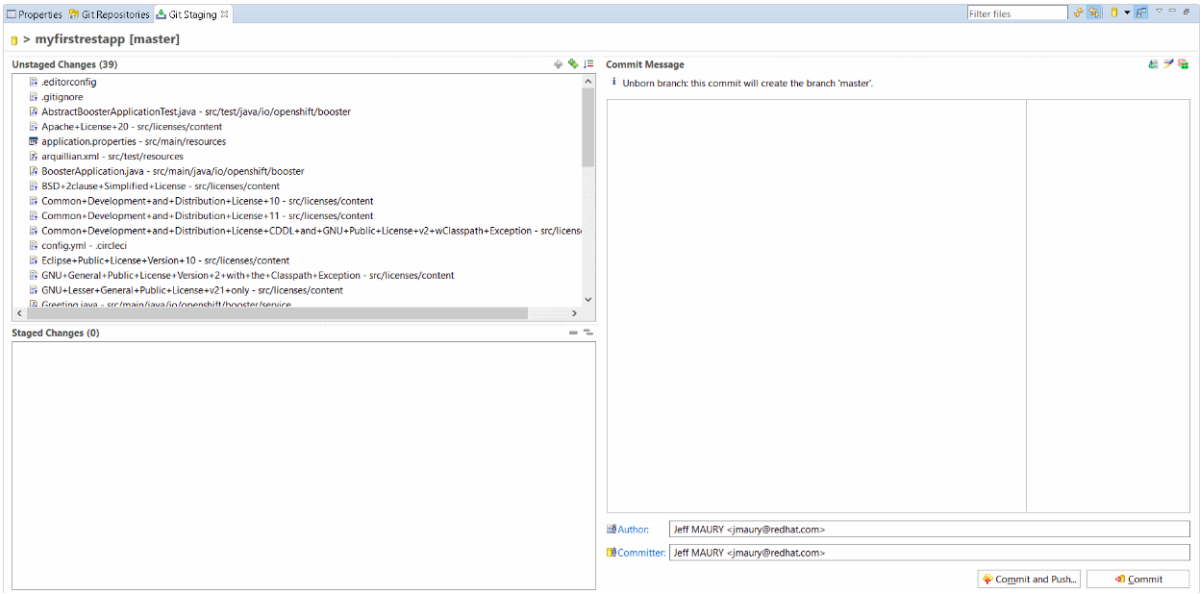
1. In the **Project Explorer** view, click the **myfirstrestapp** project. Right-click **Team** → **Commit**. A new **Git Staging** view opens.

Figure 2.13. New Git Staging View



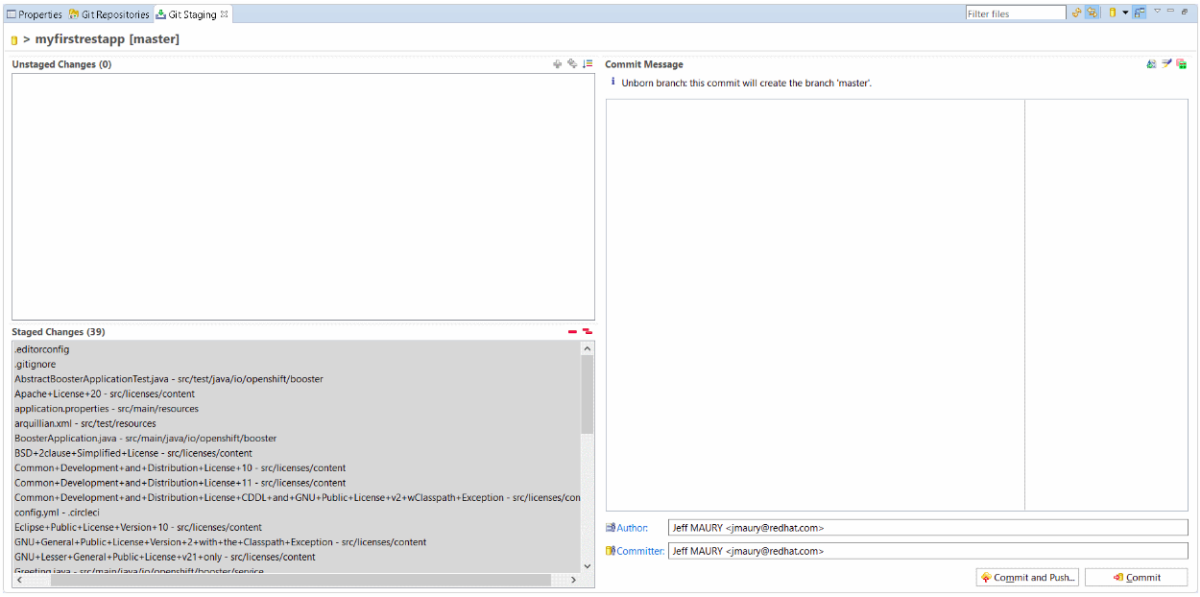
2. Double-click the view title to maximize it.

Figure 2.14. Expanded Git Staging view



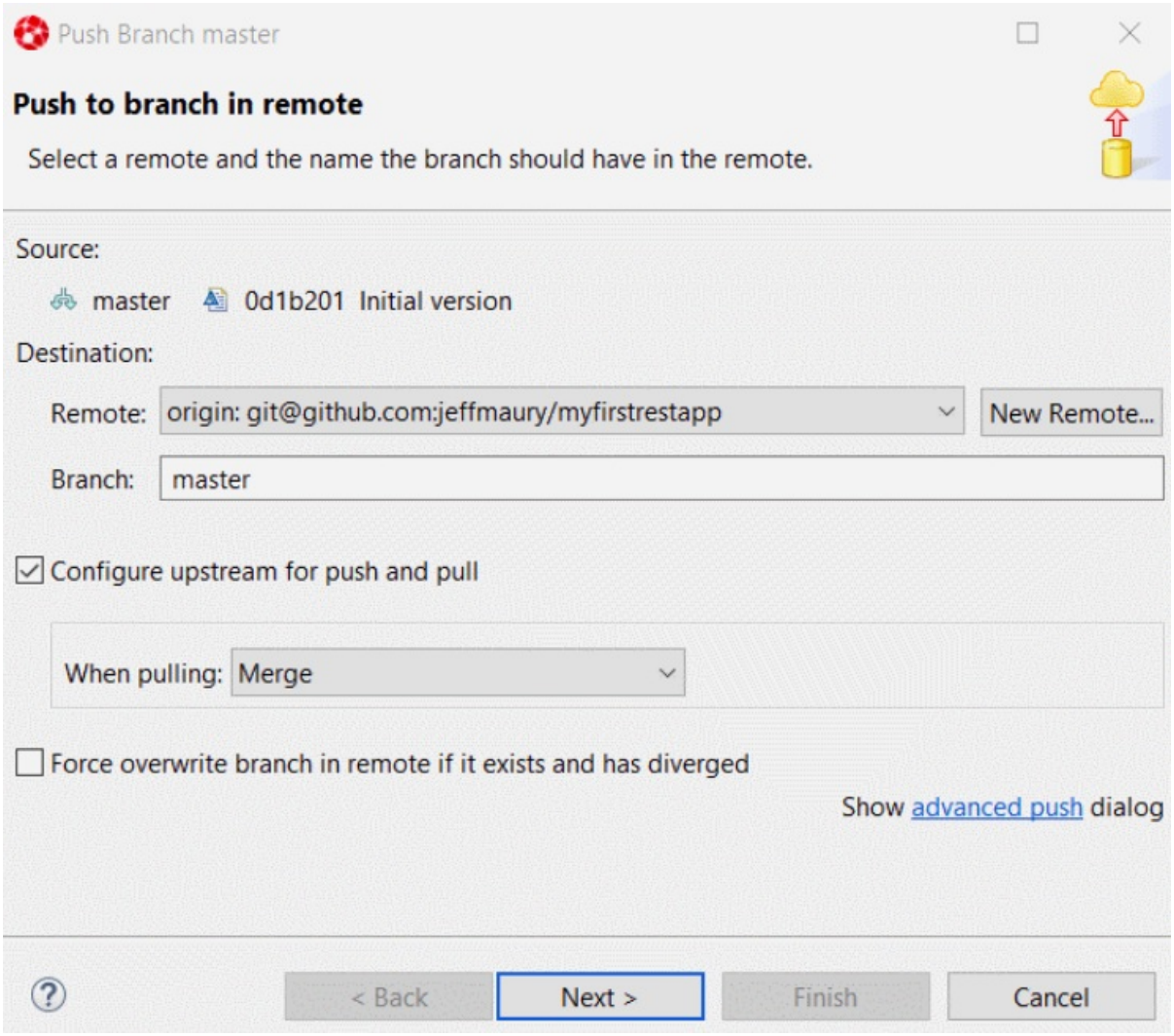
3. Select all the files listed in the **Unstaged Changes** list and click the **+** icon. The files move to the **Staged Changes** list.

Figure 2.15. Selecting files from the Unstaged Changes list



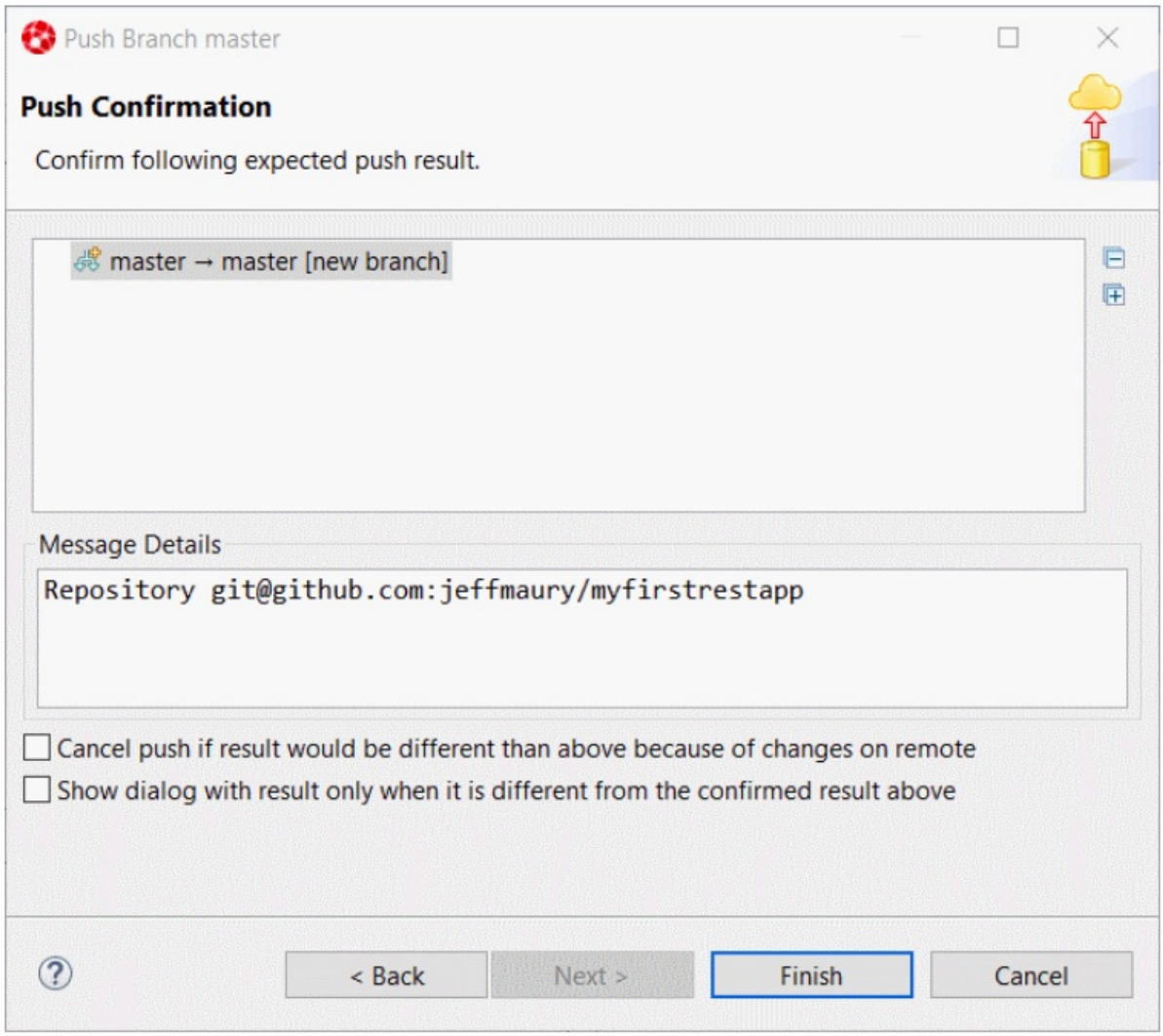
4. Enter a commit message (example: *Initial version*) and click **Commit and Push**. The **Push to branch in remote** window displays.

Figure 2.16. Push to branch in remote window



5. Click **Next**.

Figure 2.17. Push Confirmation window



6. Click **Finish** to start the push operation. A dialog with the result of the push operation displays. Click **OK** to dismiss it.

2.4.1.2.2. Add Spring Boot Devtools to the packaged application

To support live update on an OpenShift cluster, you must add Spring Boot DevTools to the Spring Boot application.

1. Open the **`pom.xml`** file in the **myfirstrestapp** project. Locate the **`spring-boot-maven-plugin`** and add the following section.

```
<configuration>
 <excludeDevtools>false</excludeDevtools>
</configuration>
```

The **`spring-boot-maven-plugin`** section must look like the following.

```
<plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <executions>
          <execution>
            <goals>
```

```
              <goal>repackage</goal>
            </goals>
            <configuration>
              <excludeDevtools>false</excludeDevtools>
            </configuration>
          </execution>
        </executions>
      </plugin>
```

2. Save and close the **pom.xml** file.

3. To push the change to GitHub, click **Team → Commit** with a new commit message (example: *With DevTools*).

2.4.1.3. Deploy the application on OpenShift

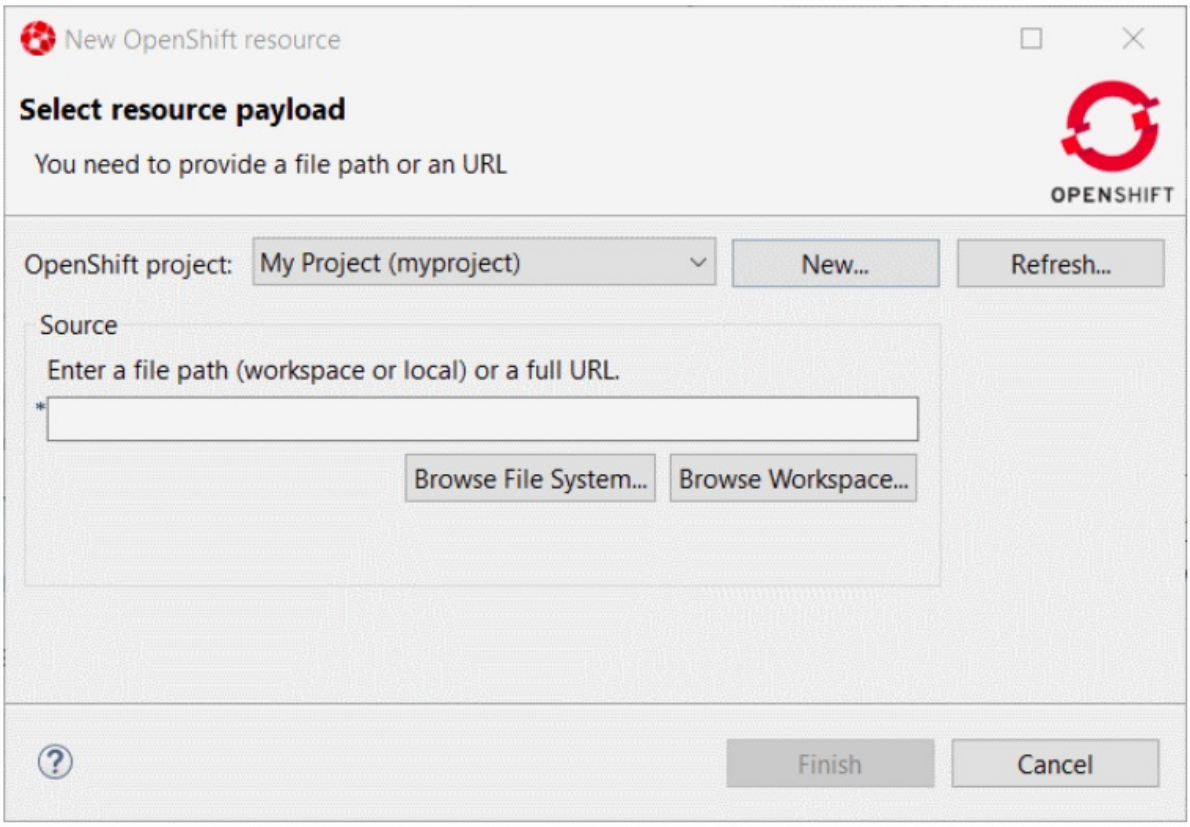
Before you can deploy the application on OpenShift, you must create an ImageStream on the OpenShift cluster. The reason is that the Spring Boot support relies on S2I builds that will explode the Spring Boot uber JAR when Spring Boot DevTools is present. As this is not supported by all Java based S2I images, you will use **fabric8/s2i-java:2.2** that supports it.

1. In the **myfirstrestapp** project, create a new JSON file called **springboot.json** and set the content of this file to the following.

```
{
    "apiVersion": "image.openshift.io/v1",
    "kind": "ImageStream",
 "metadata": {
  "name": "springboot"
 },
    "spec": {
        "lookupPolicy": {
            "local": false
        },
        "tags": [
            {
                "annotations": {
     "tags": "builder,java"
    },
                "from": {
                    "kind": "DockerImage",
                    "name": "registry.access.redhat.com/fuse7/fuse-
java-openshift:1.1"
                },
                "importPolicy": {},
                "name": "1.1",
                "referencePolicy": {
                    "type": "Source"
                }
            }
        ]
    }
}
```

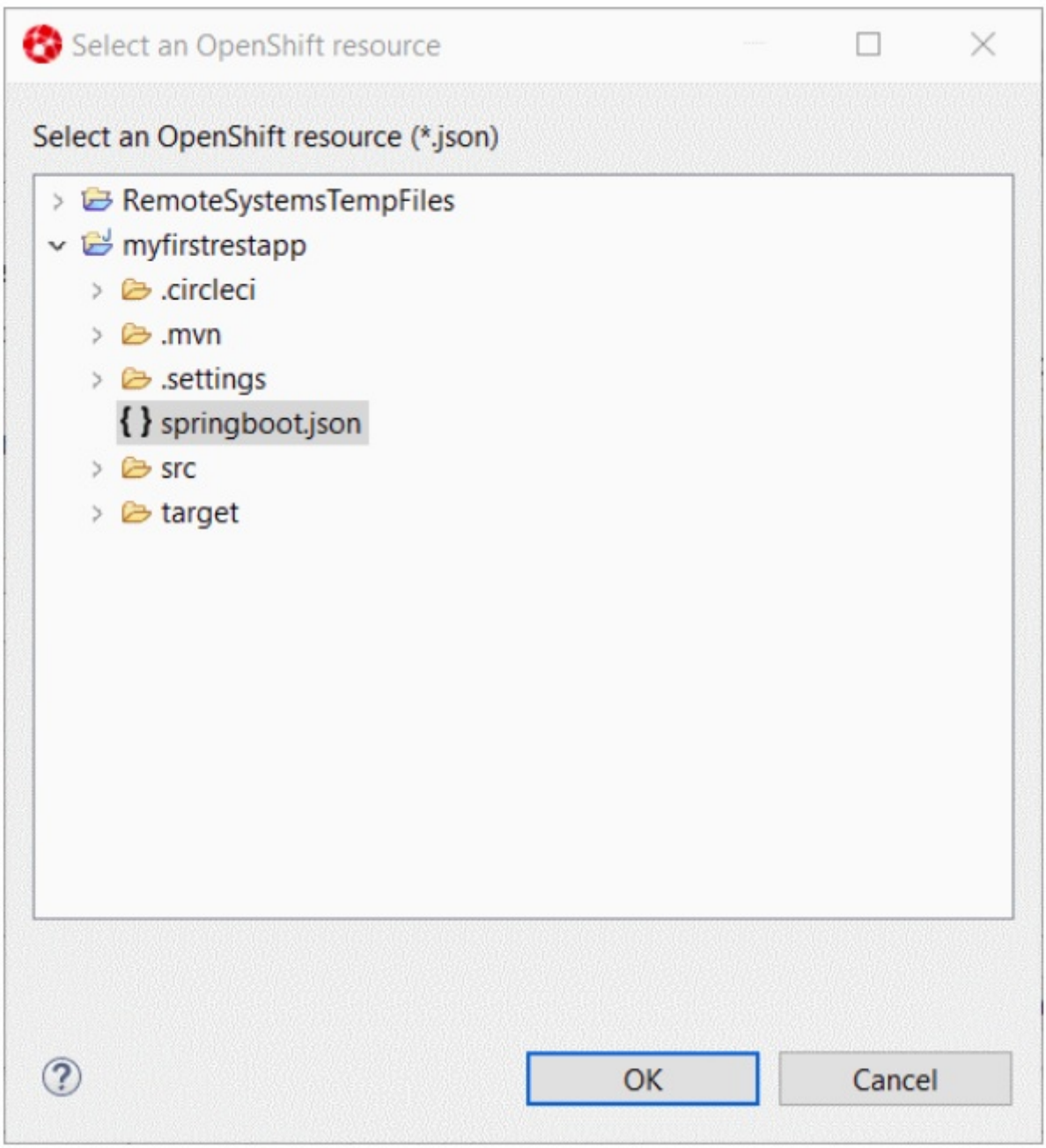
2. In the **OpenShift Explorer** view, select the OpenShift connection for your cluster (if you don't have one already defined, you must define it). Right-click **New** → **Resource**. The **Select resource payload** window displays.

Figure 2.18. Select resource payload window



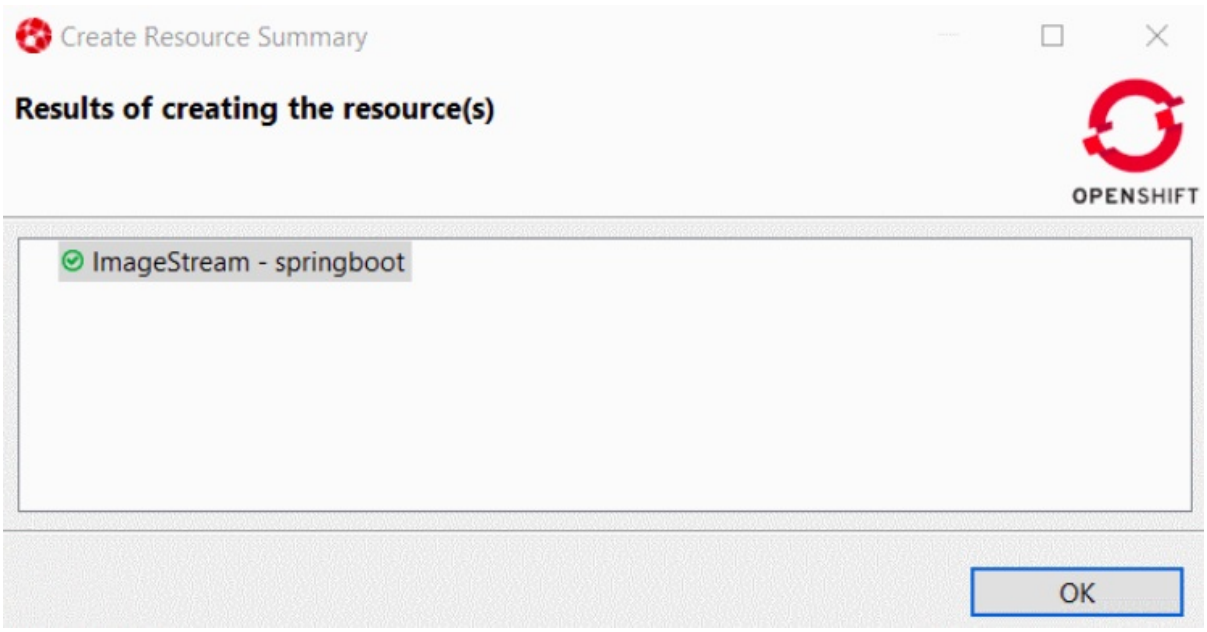
3. Select the OpenShift project that you want to work with and then click **Browse Workspace**. Select the **`springboot.json`** file in the **myfirstrestapp** project.

Figure 2.19. Select an OpenShift resource window



4. Click **OK** and then click **Finish**. The new ImageStream is created and a status dialog displays.

Figure 2.20. Results of creating the resource window

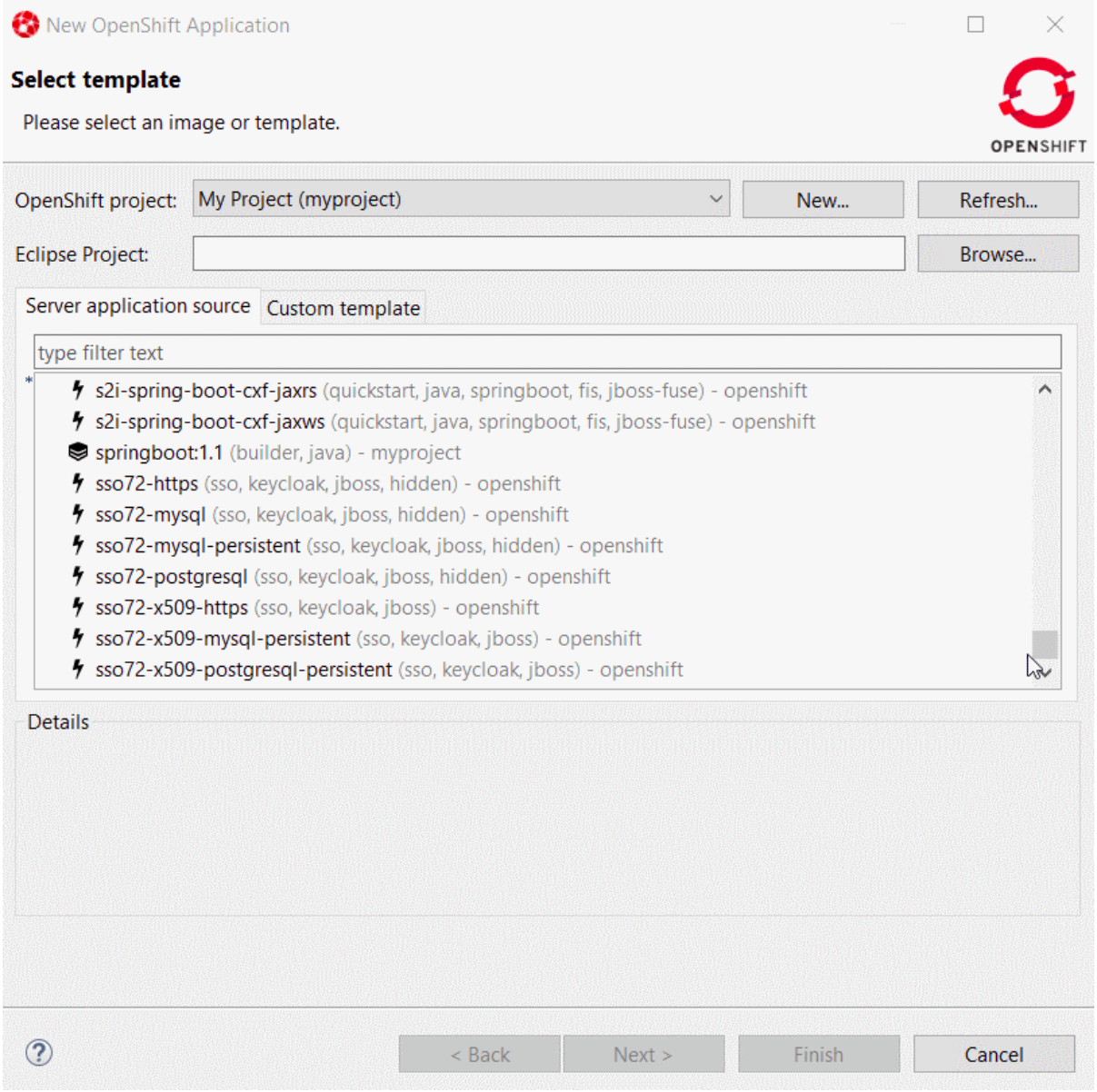


2.4.1.3.1. Create the application on OpenShift

You will now create the application on the OpenShift cluster.

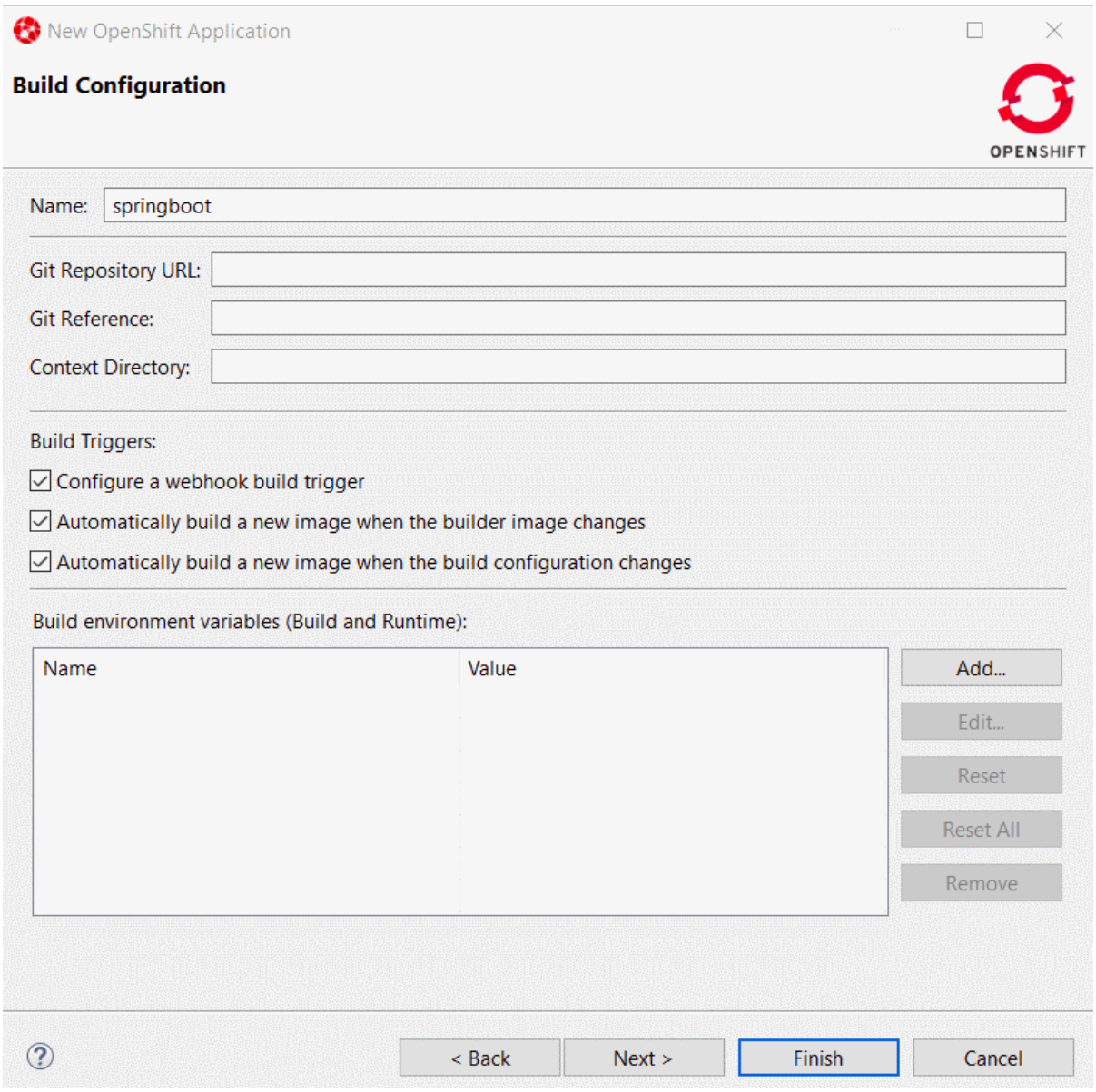
1. Select the OpenShift connection, right-click **New** → **Application**. Scroll down the list to see the springboot ImageStream that you just created.

Figure 2.21. Select template window



2. Select the ImageStream and click **Next**.

Figure 2.22. Build Configuration window



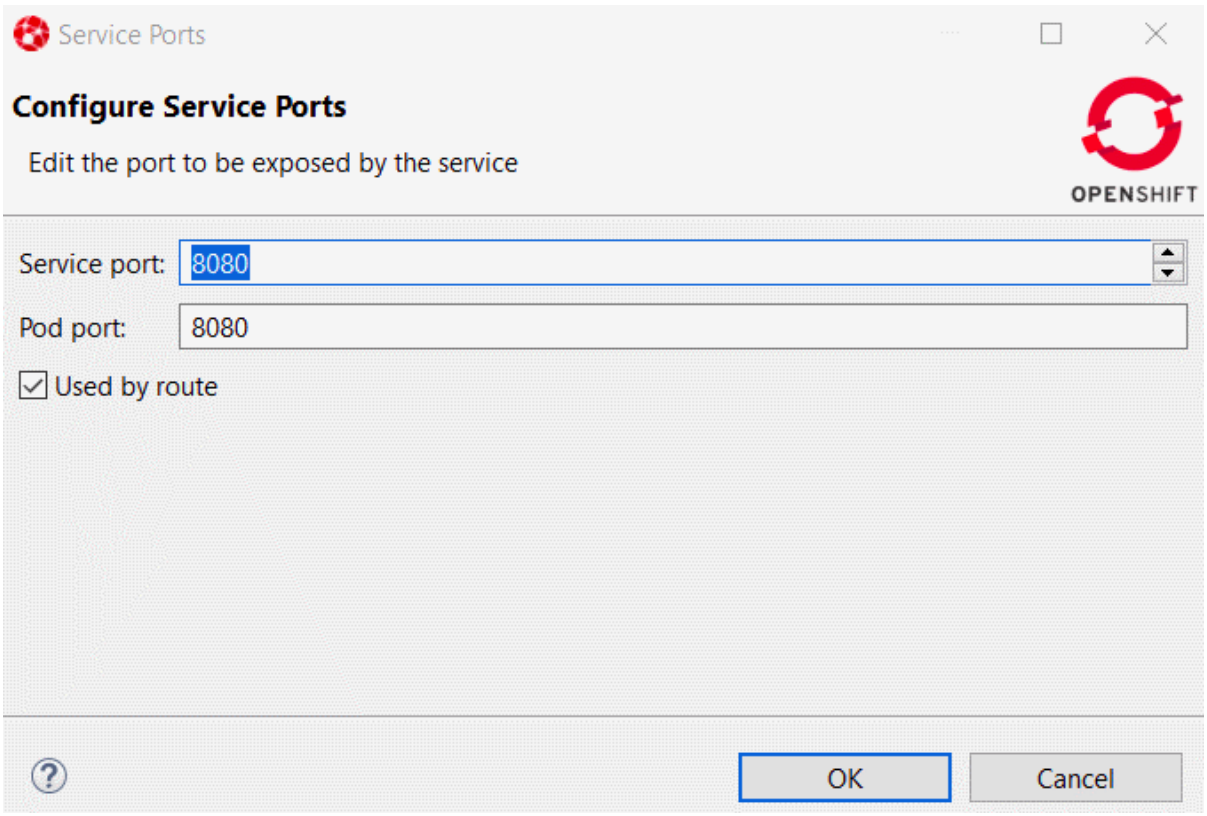
3. In the **Name** field, enter **myfirstrestapp**.

4. In the **Git Repository URL** field, type **https://github.com/{*GITHUB_USER*}/myfirstrestapp** replacing GITHUB_USER with your GitHub user name and click **Next**.

5. In the **Deployment Configuration & Scalability** window, click **Next**.

6. In the **Service & Routing Settings** window, select the **8778-tcp** port and click **Edit**. Change the 8787 value to *8080*.

Figure 2.23. Configure Service Ports window



7. Click **OK** and then click **Finish**. The list of OpenShift resources created displays.

Figure 2.24. Create Application Summary window

8. Click **OK** to dismiss it. When asked to import the application code, click **Cancel** because you already have the source code.

After the build is run (this may take several minutes as the Maven build will download the dependencies), you will see a running pod.

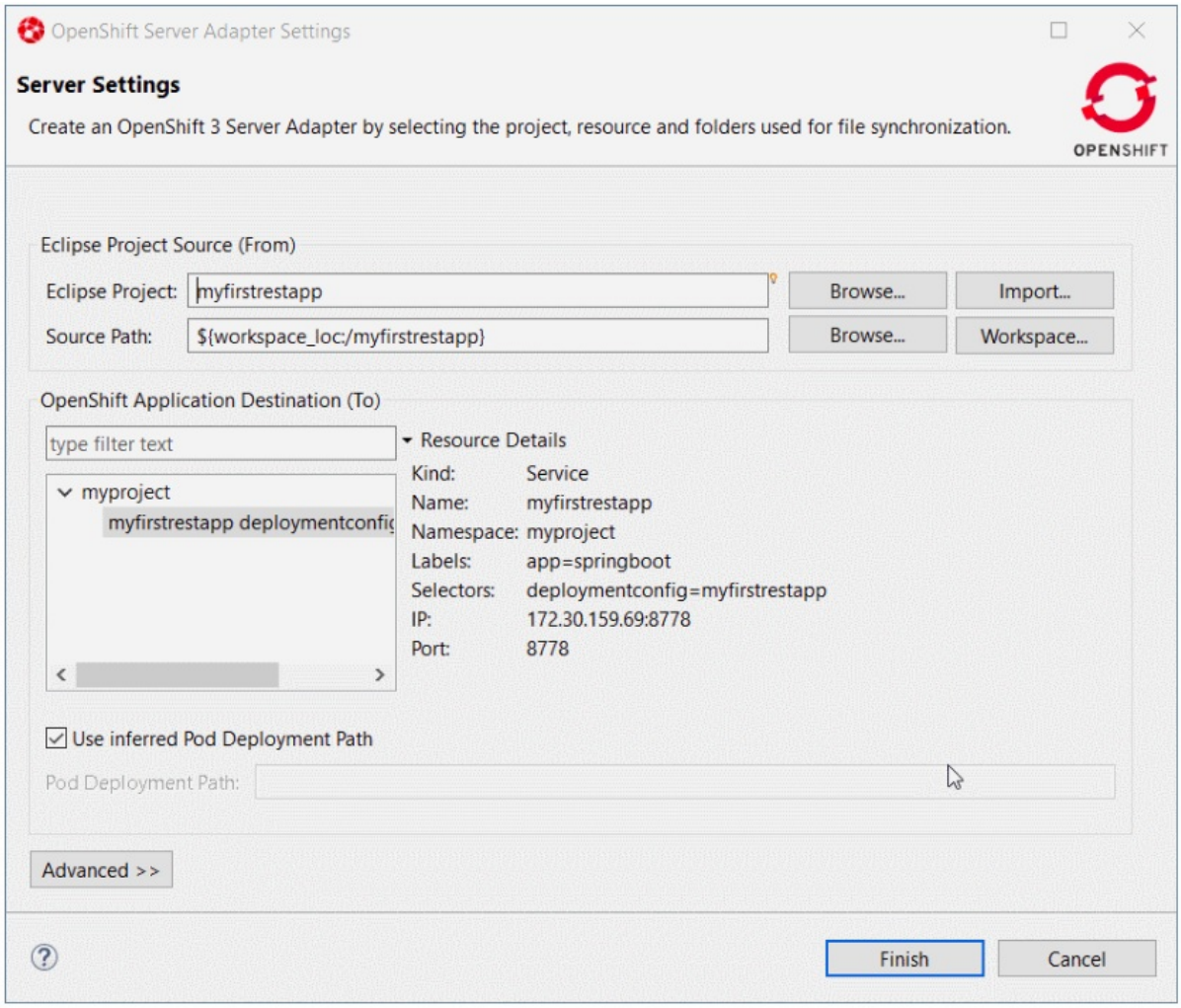
Figure 2.25. Running Pod

2.4.1.4. Inner loop setup

You will now synchronize the local Eclipse project with the remote OpenShift pod. Every time a file is modified locally, the pod is updated accordingly.

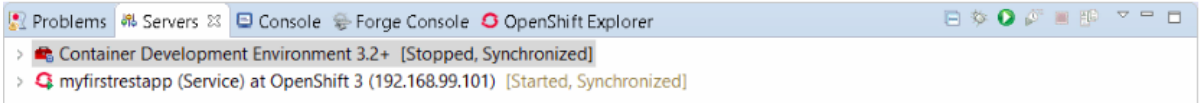
1. Select the running pod in the **OpenShift Explorer** view, right-click the **Server Adapter**. The **Server Settings** window displays.

   Figure 2.26. Server Settings window



2. Click **OK**; the initial synchronization begins and the **Servers** view displays.

   Figure 2.27. Servers view



So far, you have not set up the synchronization between the local Eclipse project and the remote OpenShift project. Each modification done locally will be reported on the remote OpenShift cluster.

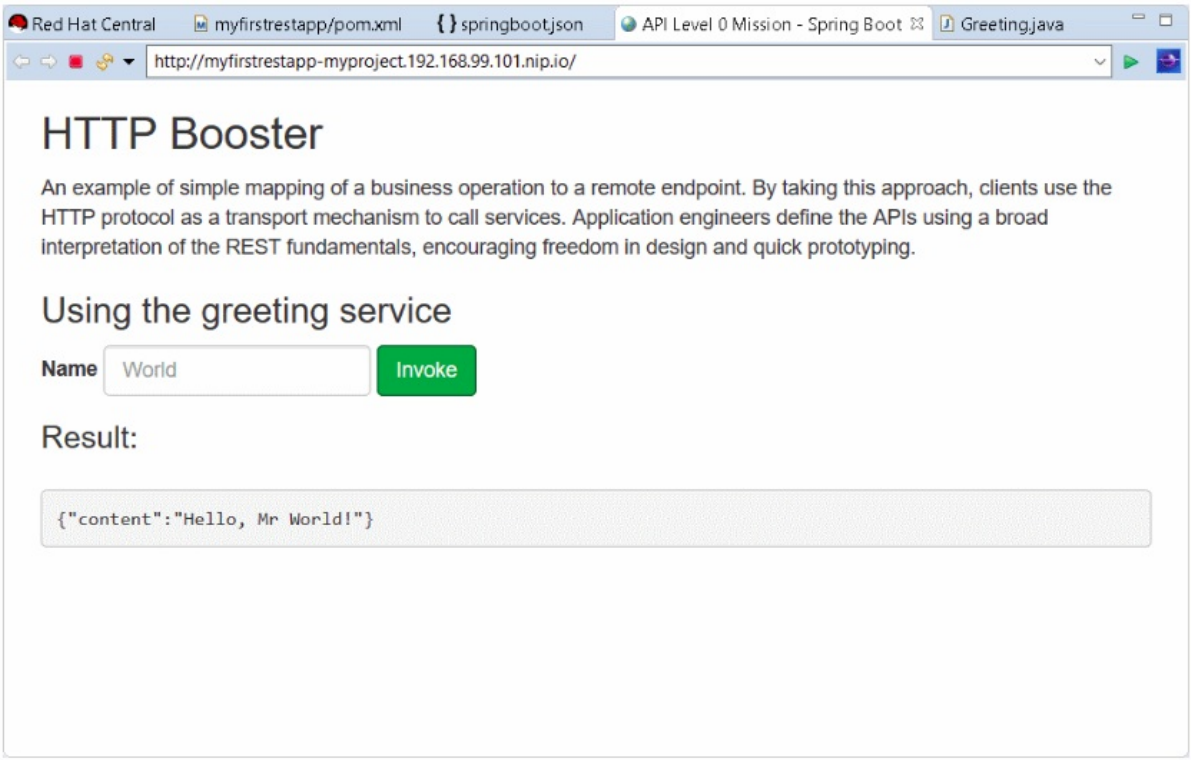
To modify the local application code and see the changes applied almost instantly:

1. Edit the **`src/main/java/io/openshift/booster/service/Greeting.java`** file in the **myfirstrestapp** project and change the **`FORMAT`** string value from **Hello, %s!** to **Hello, Mr %s!** and save the file.

```
/*
 * Copyright 2016-2017 Red Hat, Inc, and individual contributors.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */
package io.openshift.booster.service;

// tag::snippet-greeting[]
public class Greeting {

    public static final String FORMAT = "Hello, Mr %s!";

    private final String content;

    public Greeting() {
        this.content = null;
    }

    public Greeting(String content) {
        this.content = content;
    }

    public String getContent() {
        return content;
    }
}
// end::snippet-greeting[]
```

2. In the **ÒpenShift Explorer** view, select the myfirstrestapp deployment and select **Show In → Web Browser**. After the web browser displays, click **Invoke**.

Figure 2.28. myfirstrestapp in the web browser



The preceding sections show the inner loop on your Spring Boot application. Any change done locally is reported and can be tested almost immedialtely on your OpenShift cluster.

You can restart the deployment in debug mode and you can remote debug your Spring Boot application.

Related JIRA: JBIDE-26162

2.5. SERVER TOOLS

2.5.1. Wildfly 14 Server Adapter

A server adapter is added to work with Wildfly 14. It adds support for Java EE 8.

Related JIRA: JBIDE-26335

CHAPTER 3. ISSUES

3.1. RESOLVED ISSUES FOR RED HAT DEVELOPER STUDIO

To view information about resolved issues in this release of Developer Studio, see the Resolved Issues.

3.2. KNOWN ISSUES FOR RED HAT DEVELOPER STUDIO

To view information about known issues in this release of Developer Studio, see the Known Issues.

The following known issues are highlighted:

- JBIDE-20983: cannot use oracle service name in datasource creation
- JBIDE-19633: Not able to create 'non-bare' repository in JBDS 8.1.
- JBIDE-17176: Unable to browse and select PortletBridge runtime libraries in JPP 6
- JBIDE-12957: Xhtml files appear garbled when it’s reopened in the JBDS editor
- JBDS-3645: Installation of Developer Studio to a network drive fails
- JBDS-3470: Toolbars + Icons unusable on UHD screens
- JBDS-3069: Ungraceful shutdown results in multiple errors on startup
- JBDS-4442: Central page does not work on Fedora 26 if package webkitgtk3 is not installed
- JBIDE-25146: Eclipse annotation processing not enabled by default can result in errors

3.3. KNOWN ISSUES FOR RED HAT FUSE

Following is a list of issues for the DevStudio and Fuse integration:

- FUSETOOLS-2859: For a project that contains a data transformation file, if you change the Camel version from a version earlier than 2.20 to version 2.20 or later, you must open the data transformation file in the Data Transformation editor to complete the Camel version change. Otherwise, the project creates an error at runtime.

CHAPTER 4. ADDITIONAL RESOURCES

Developer Studio 12.9 is available from a number of sources:

- To install Developer Studio 12.9, use the universal installer available from the Red Hat Customer Portal or from the Red Hat Developer Program.
- To install Developer Studio BYOE 12.9 in Eclipse Oxygen, use Eclipse Marketplace, the Developer Studio update site or the update *.zip* file available from the Red Hat Customer Portal.

In all cases, for more information, see the Red Hat Developer Studio Installation Guide at the Developer Studio Documentation page.