Red Hat JBoss Developer Studio 10.2
Getting Started with Container and Cloud-based Development

Starting Development of Container and Cloud-based Applications Using Red Hat JBoss Developer Studio

Misha Husnain Ali   Supriya Bharadwaj
Red Hat Developer Group Documentation Team

Red Hat JBoss Developer Studio 10.2 Getting Started with Container and Cloud-based Development

Starting Development of Container and Cloud-based Applications Using Red Hat JBoss Developer Studio

Misha Husnain Ali
mhusnain@redhat.com

Supriya Bharadwaj
sbharadw@redhat.com

Legal Notice

Abstract

This compilation of topics contains information on how to start developing containerized applications and applications for cloud deployment.

Table of Contents

CHAPTER 1. DEVELOPING USING CONTAINERS AND THE CLOUD

1.1. USING CONTAINER DEVELOPMENT KIT TOOLING IN JBOSS DEVELOPER STUDIO 10.X

1.1.1. About Using Red Hat Container Development Kit with the IDE

Red Hat Container Development Kit is a pre-built container development environment based on Red Hat Enterprise Linux. Red Hat Container Development Kit helps you get started with developing container-based applications quickly. You can easily set up Red Hat Container Development Kit and then use toolings, such as, OpenShift Container Platform and Docker, through JBoss Developer Studio, without spending any additional time in setting up and configuring the supplementary tooling.

There are two ways to install Red Hat Container Development Kit with JBoss Developer Studio:

1. Section 1.1.2, "Installing Red Hat Container Development Kit and JBoss Developer Studio Using the Red Hat Development Suite Installer"

2. Section 1.1.3, "Installing Red Hat Container Development Kit and JBoss Developer Studio as Separate Entities"

Once installed, you can use the installed components with the Docker Tooling.

1.1.2. Installing Red Hat Container Development Kit and JBoss Developer Studio Using the Red Hat Development Suite Installer

Use the Red Hat Development Suite Installer to install Red Hat Container Development Kit, JBoss Developer Studio, and other relevant components. The Installer automatically configures these components for use together. This option is currently available for Windows and macOS.

For instructions about using the Red Hat Development Suite Installer, see Red Hat Development Suite Istallation Guide.

After using the Red Hat Development Suite Installer to install Red Hat Container Development Kit and JBoss Developer Studio, manually run the Container Development Environment tooling, which also creates an OpenShift and a Docker connection.

1.1.3. Installing Red Hat Container Development Kit and JBoss Developer Studio as Separate Entities

You can download and install Red Hat Container Development Kit and the JBoss Developer Studio separately. This option requires some additional configuration steps before the two can be used together.

1.1.3.1. Prerequisites

Ensure that the following are installed on your system:

- A virtualization system such as VirtualBox, VMWare, or Linux KVM/libvirt

- Vagrant, which is an open source tool to create and distribute portable development environments

- Red Hat Container Development Kit 2.3

- JBoss Developer Studio 10.x

For details about installing these prerequisites, see the Red Hat Container Development Kit Installation Guide.

1.1.3.2. Set Up Red Hat Container Development Kit in the IDE

To set up Red Hat Container Development Kit in the IDE:

1. Start the IDE.

2. Press Ctrl+3 and in the **Quick Access** bar, type **CDK**.

3. From the results, click **Launch Container Development Environment using Red Hat CDK**.

4. If asked, enter your user credentials.

5. In the **New Server** dialog box:

   a. Ensure that **Red Hat Container Development** is selected by default.

   b. In the **Server's host name** field, type the desired server host name.

   c. In the **Server Name** field, type the desired server name.

   d. Click **Next** to continue.
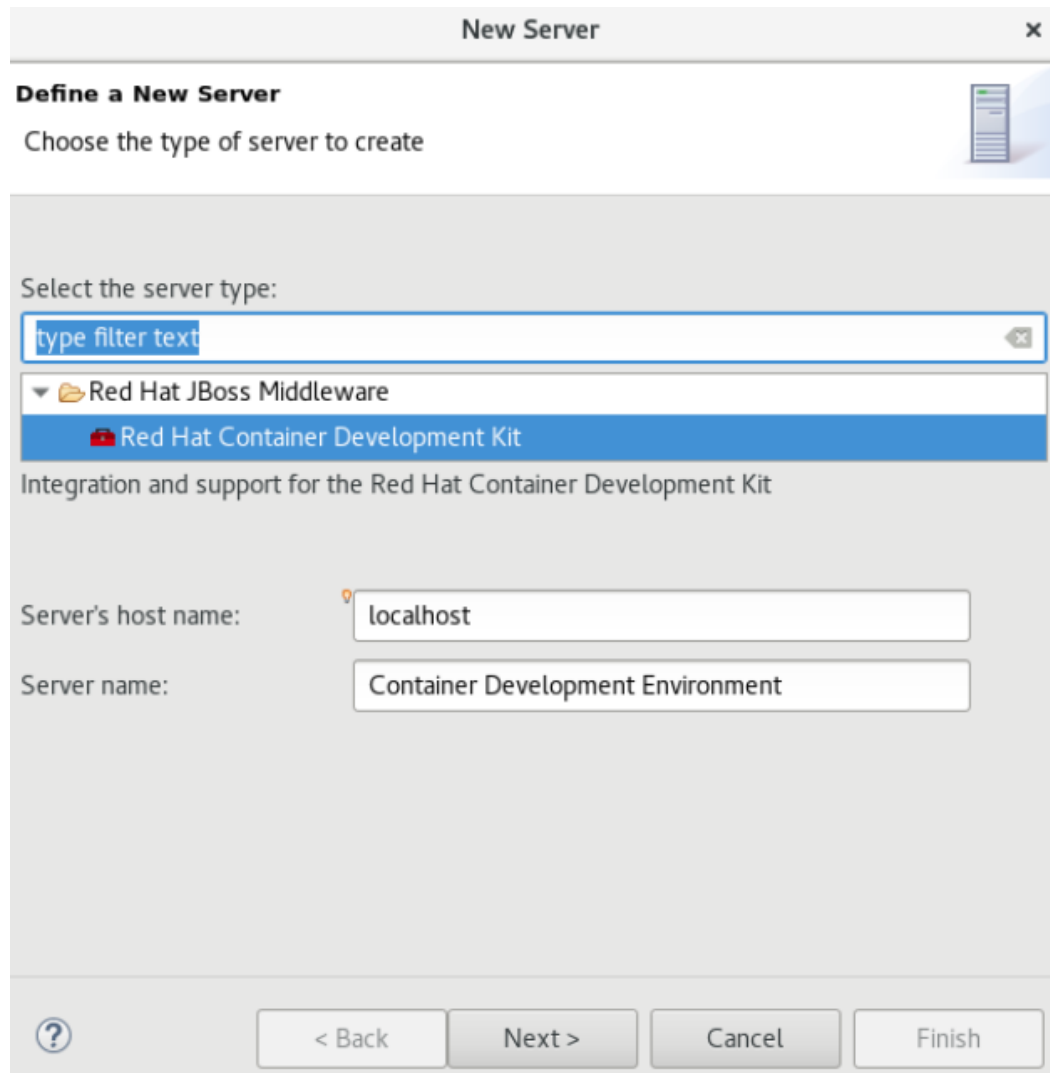
6. In the **New Server Red Hat Container Development Environment** window, add the security information and your access.redhat.com credentials:

   a. In the **Username** field, click **Add** to add a new username for the Red Hat Customer Portal or select an existing user from the drop-down menu.

   b. In the **Folder** field, navigate to the directory that contains your local **Vagrantfile**. You can also use the **Vagrantfile** supplied in the **cdk.zip** file to initialize the Red Hat Container Development Kit box.

7. Click **Finish** to conclude setting up the Red Hat Container Development Kit server adapter.
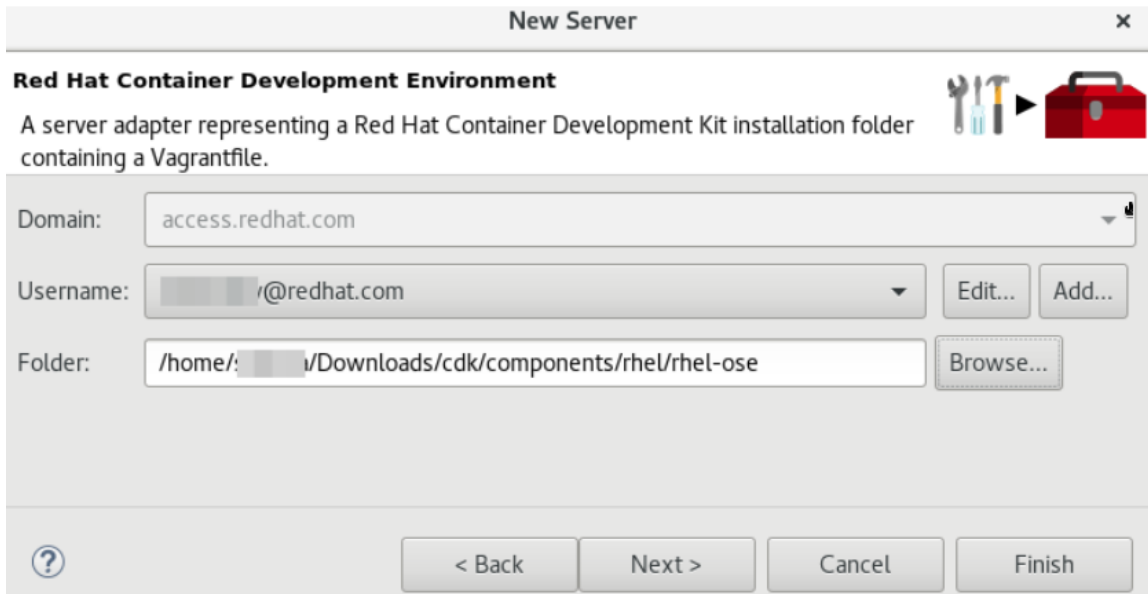
8. Open the **Servers** view, right-click **Container Development Environment** and click **Start**.

9. If asked, enter the password.

10. In the **Untrusted SSL Certificate** dialog box, click **Yes**.
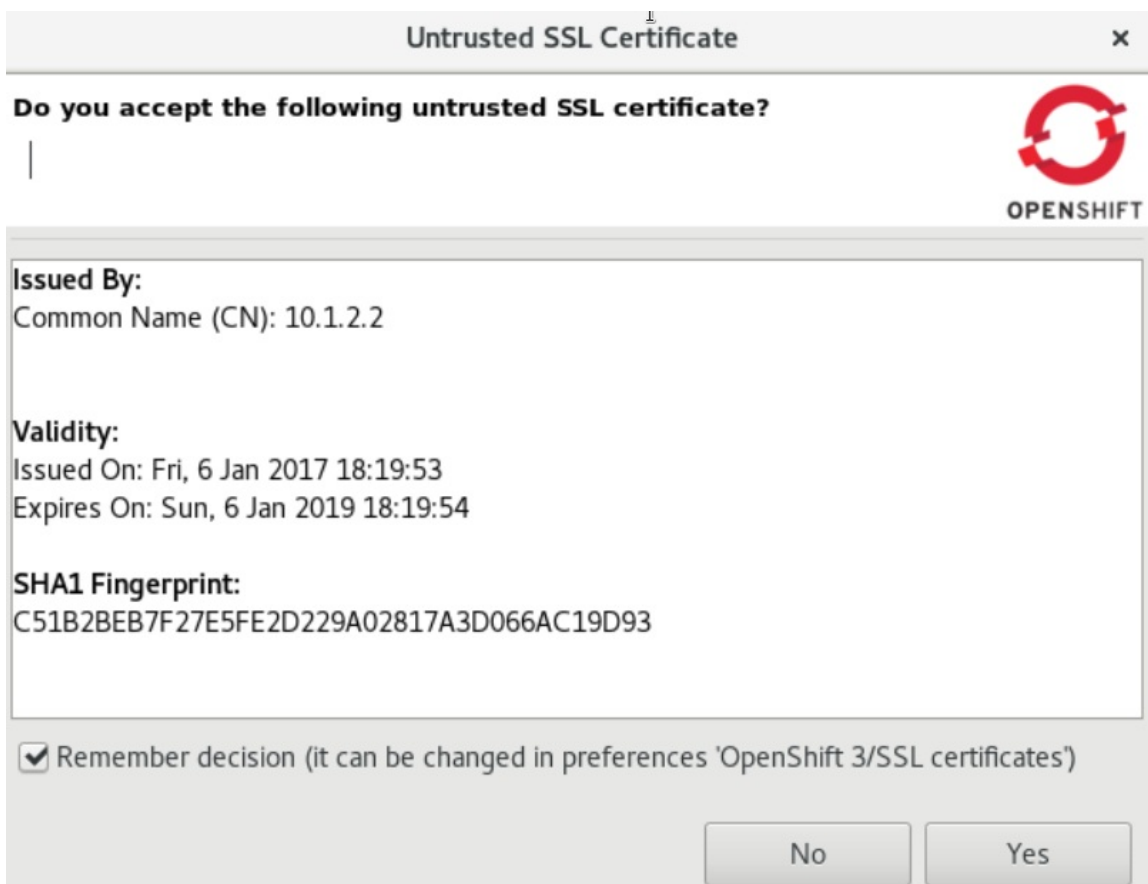
11. The **Servers** view shows the server adapter started. To monitor the status of the server adapter, read the log in the built-in terminal.
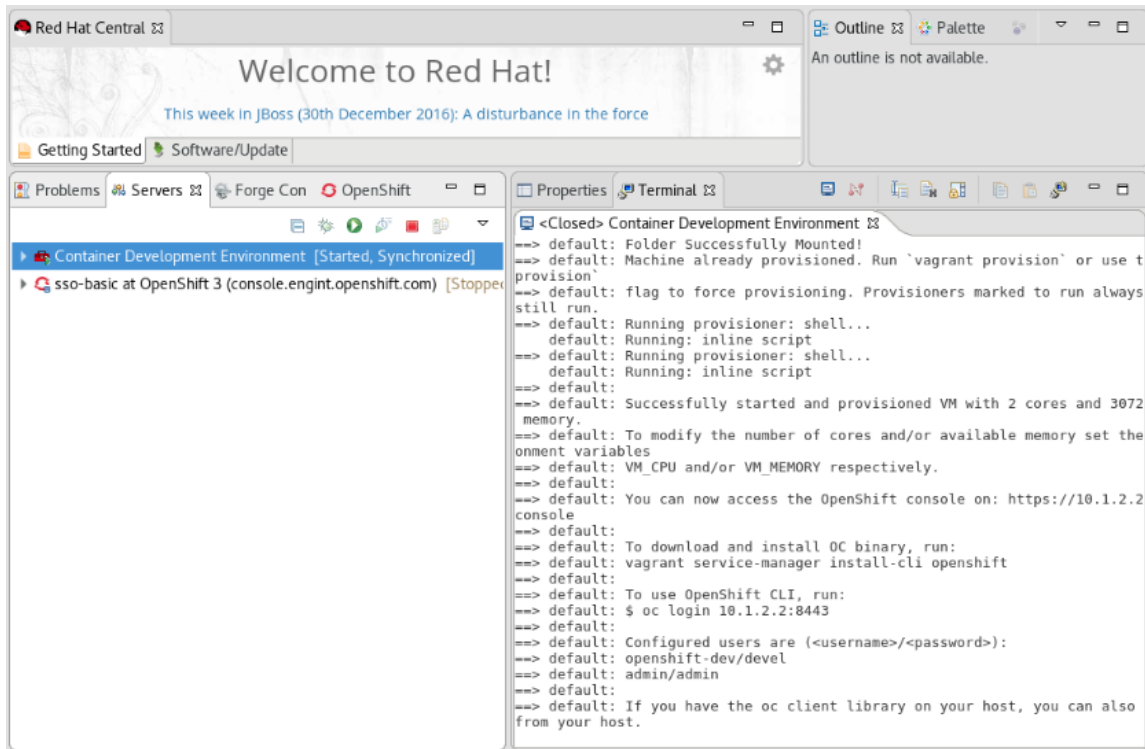
You can open the **OpenShift Explorer** view to see the IP address and the port of the OpenShift Conatiner Platform that you have connected to (example: **openshift-dev https://10.1.2.2:8443**). Expand the connection to see the sample projects. You can also open the **Docker Explorer** view to view the Container Development Environment connection and expand the connection to see the Containers and Images.

Choose to continue working with OpenShift Container Platform within JBoss Developer Studio or view instructions for Container-Based Development with JBoss Developer Studio.

1.1.4. Using the Docker Tooling

After starting the Red Hat Container Development Kit server in the IDE, you can follow one of the two container development workflows:

» Section 1.1.4.1, "Use Docker for Container-based Development"

» Section 1.1.4.2, "Build Docker Using the Container Development Environment"

1.1.4.1. Use Docker for Container-based Development

Use Docker for Container-based Development as follows:

1. Create a new project with your **Dockerfile**.

    a. Click **File** > **New** > **Project**.

    b. Type **java** in the search field and from the results, select **Java Project** and click **Next** to continue.

    c. Add a name for the new project and click **Finish**. The **Project Explorer** view shows the project that you just created.

    d. Click **File** > **New** > **File**.

     e. In the **New File** window:

        i. In the **Enter or select the parent folder** field, click the project name that you created.

        ii. In the **File name** field, type **Dockerfile** and click **Finish**.

        iii. Edit the **Dockerfile** as desired and then save (for example, by creating a new Docker image to customize a given version of JBoss/WildFly by adding a datasource definition and its associated driver). The **Dockerfile** may also package your application as a **war** file via a Maven command, and copy it into the container in the WildFly deployments directory. See https://docs.docker.com/engine/reference/builder for more information about the **Dockerfile** instructions.

1.1.4.2. Build Docker Using the Container Development Environment

Do a Docker build using the Container Development Environment as follows:

1. In the **Project Explorer** view, right-click the **Dockerfile** and select **Run As** > **Docker Image Build**.

2. In the dialog box:

    a. In the **Connection** field, select your Container Development Environment server adapter.

    b. In the **Repository Name** field, enter the desired name for the docker image and click **OK**. After the build is done, a new image with the given name will be listed in the **Docker Explorer** view and in the **Docker Images** view.

3. Do a Docker run using the Container Development Environment:

    a. Open the **Docker Explorer** view by typing Ctrl+3 in the quick access menu.

    b. Navigate to the **Images** node under the Docker connection.

    c. Right-click your image and click **Run**.

    d. Fill in the necessary details and click **Finish** to run your image. It is optional to give the container a name, but it is recommended to name it. This name helps locate the specific container in a list of containers in the future.

4. In the **Docker Explorer** view, select the container and expand its node and select the 8080 port and click **Show In** > **Web Browser** to access the application deployed in the Docker container.

1.1.4.2.1. Next Steps for the Docker Tooling

For further information about the basics of Docker Tooling, see Configure Docker Tooling (Basic).

1.1.5. Using OpenShift Container Platform Tooling

Use OpenShift for Container-based Development as follows:

1. Create a new OpenShift Container Platform project. These projects are like namespaces for OpenShift applications. They are different from how Eclipse projects relate to Eclipse applications. Additionally, Eclipse projects can be mapped to OpenShift applications.

   a. In the **OpenShift Explorer** view, right-click the name of the connection and select **New** > **Project** to create a new OpenShift Container Platform project.

   > **Note**
   >
   > The CDK server adapter creates the OpenShift Container Platform connection when you start the CDK server adapter in the preceding sections.

   b. Add the name and any other relevant details for the new project and click **Finish**.

2. Create an application in your OpenShift Container Platform project using the templates:

   a. Right-click your new project name and click **New** > **Application**.

   b. In the **Select Template** dialog box, search box, type the application type required. For example, for a node.js application, type **nodejs** and from the displayed list, select the relevant nodejs template and click **Finish**.

   c. Click **OK** to accept the results of the application creation process.

   d. When prompted, enter a new git location or click **Finish** in the dialog box to use the listed default git location for your application.

3. Create a new OpenShift Container Platform server adapter for your project and application:

   a. In the **OpenShift Explorer** view, expand the connection and then the project.

   b. Right-click the service and click **Server Adapter**.

   c. In the **OpenShift Adapter Settings** window:

      i. Ensure that the **Eclipse Project** field, shows the relevant project or click **Browse** to locate the project.

      ii. In the **Services** field, select the relevant service.

      iii. Click **Finish**.

4. Debug the application, if required.

   a. In the **Servers** view, right-click the server adapter and click **Restart in Debug**.

1.1.5.1. Next Steps for the OpenShift Tooling

See the Developing for the Cloud with OpenShift 3 section in the index for additional tasks using the OpenShift Container Platform tooling.

1.1.6. Known Issues

- When the **Docker Explorer** is first started, attempting to extend the Containers or Images causes the explorer to fail and throw an exception. To work around this issue, restart Eclipse/JBoss Developer Studio. Details are in JBIDE-21983.

CHAPTER 2. DEVELOPING FOR THE CLOUD WITH OPENSHIFT 3

2.1. CREATING AN OPENSHIFT CONTAINER PLATFORM APPLICATION IN RED HAT JBOSS DEVELOPER STUDIO

Using the OpenShift Container Platform Tooling you can create, import, and modify OpenShift Container Platform applications by:

1. Creating a New OpenShift Container Platform Connection

2. Creating a New OpenShift Container Platform Project

3. Creating a New OpenShift Container Platform Application

4. Importing an Existing OpenShift Container Platform Application into the IDE

5. Deploying an Application Using the Server Adapter

6. Viewing an Existing Application in a Web Browser

7. Deleting an OpenShift Container Platform Project

2.1.1. Creating a New OpenShift Container Platform Connection

To use the OpenShift Container Platform tooling in the IDE, you must first create an OpenShift Container Platform connection. To create a new connection:

1. In the **OpenShift Explorer** view, click **New Connection Wizard**. If the **OpenShift Explorer** view is not available, click **Window → Show View → Other** and then in the **Show View** window search for **OpenShift Explorer** and after you find it, click **OK**.

2. In the **New OpenShift Connection** wizard:

   a. In the **Connection** list, click **<New Connection>**.

   b. In the **Server type** list, click **OpenShift 3**.

   c. In the **Server** field, type the URL for an OpenShift Container Platform server.

   d. In the **Authentication** section, in the **Protocol** list, click **OAuth** to authenticate using the token or click **Basic** to authenticate using login credentials.

3. Click **Finish**.

Figure 2.1. Set up a New OpenShift Container Platform Connection

Result: The connection is listed in the **OpenShift Explorer** view.

2.1.2. Creating a New OpenShift Container Platform Project

To create a new OpenShift Container Platform project:

1. In the **OpenShift Explorer** view, right-click the connection and click **New** → **Project**.

2. In the **Create OpenShift Project** window:

a. In the **Project Name** field, type a name for the project. Project names must be alphanumeric and can contain the character "-" but must not begin or end with this character.

b. In the **Display Name** field, type a display name for the project. This name is used as the display name for your project in the **OpenShift Explorer** view and on the OpenShift Container Platform web console after the project is created.

c. In the **Description** field, type a description of the project.

3. Click **Finish**.

Result: The project is listed in the **OpenShift Explorer** view, under the relevant connection.

2.1.3. Creating a New OpenShift Container Platform Application

Use the **New OpenShift Application** wizard to create OpenShift Container Platform applications from default or custom templates. Using a template to create an application is useful because the same template can be used to create multiple similar applications with different or identical configurations for each of them.

> **Note**
>
> To learn more about using and creating templates with OpenShift Container Platform, see Templates.

1. In the **OpenShift Explorer** view, right-click the connection and click **New → Application**.

2. If required, in the **New OpenShift Application** wizard, sign into your OpenShift Container Platform server using the **Basic** protocol (username and password) or the **OAuth** protocol (token) and click **Next**.

3. In the **Select Template** window, click the **Server application source** tab.

> **Note**
>
> To create an application from the a local template, click the **Local template** tab and then click **Browse File System** or **Browse Workspace** to locate the template that you want to base the project on.

4. From the list, click the template that you want to base your project on. You can also use the **type filter text** field to search for specific templates.

5. Click **Next**.

Figure 2.2. Select a Template for Project Creation
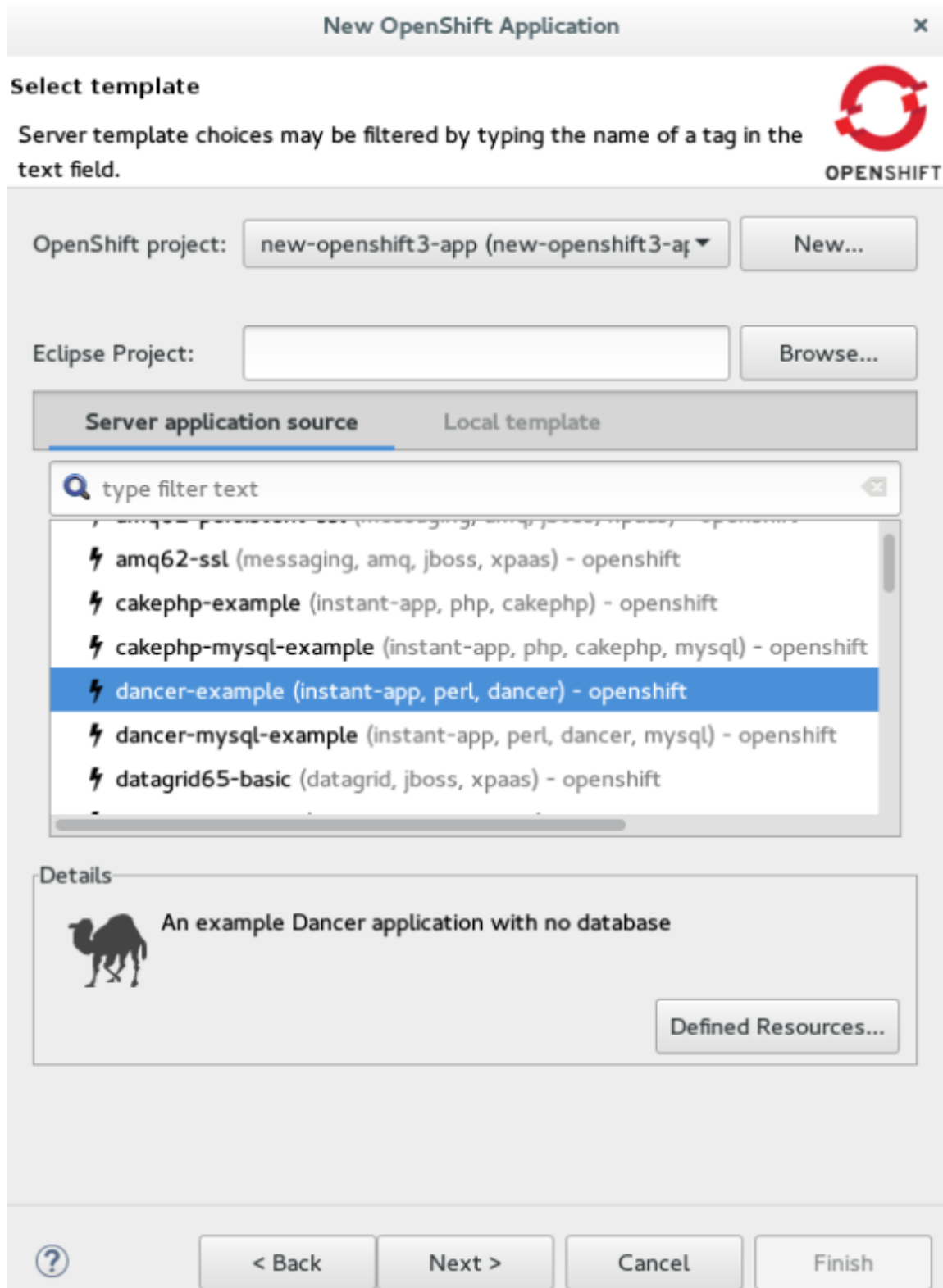
6. In the **Template Parameters** window, confirm the parameter values and click **Next**.

7. In the **Resource Labels** window, confirm the labels that you want to add to each resource. You can also click **Add** or **Edit** to add labels or edit the existing ones.

8. Click **Finish**.

9. In the **Results of creating the resources from the [template_name]** window, review the details and click **OK**.

10. In the **Import Application** window, click **Use default clone destination** to clone the application at the default location or in the **Git Clone Location** field, type or

browse for the location where you want to clone the application and click **Finish**.

Figure 2.3. Select a Git Clone Location
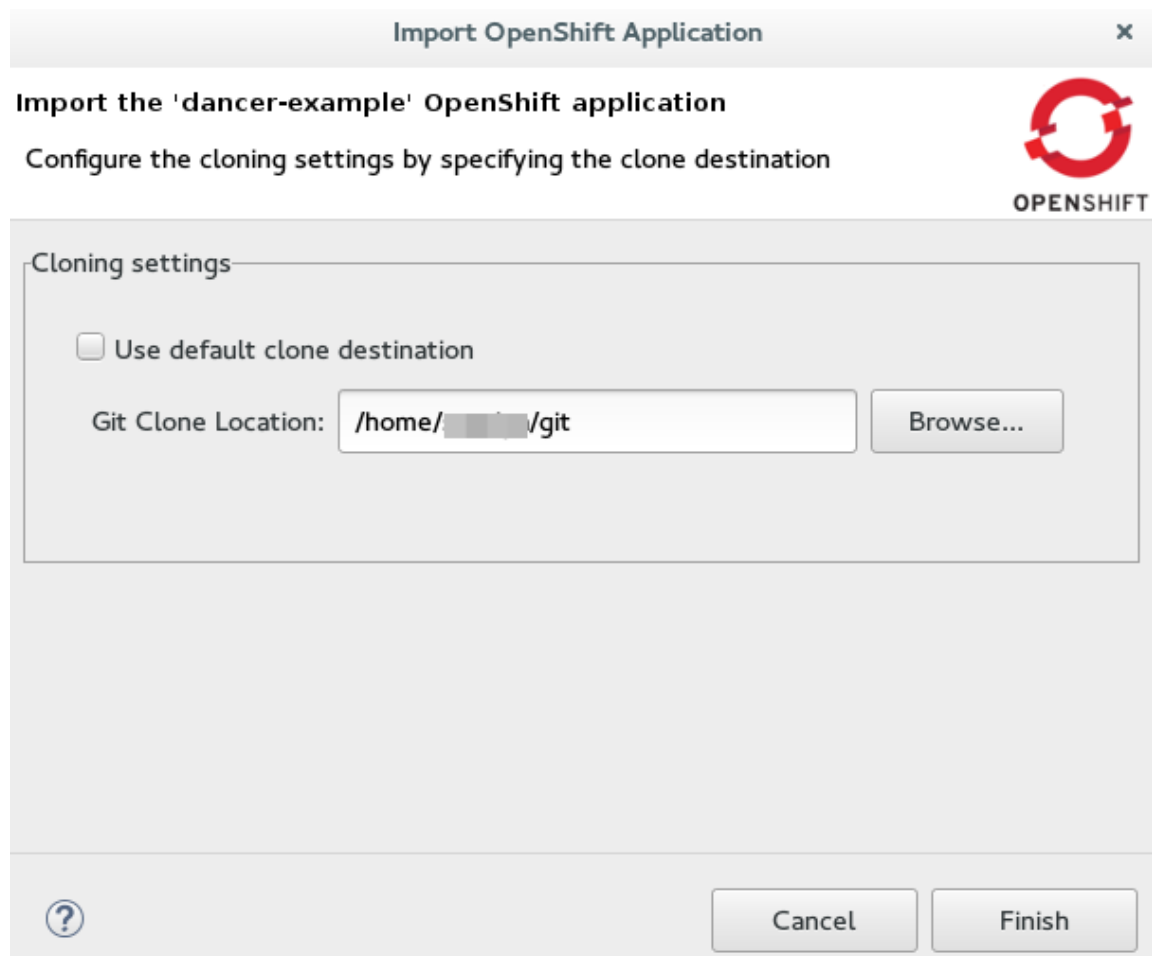
> **Note**
>
> If the Git location chosen to clone the application already contains a folder with the application name that you are trying to import, you must select a new location for the Git clone. If you do not select a new location, the existing repository will be reused with the changes you made being retained but not reflected on the OpenShift Container Platform console.

Figure 2.4. Git Clone Location Reuse
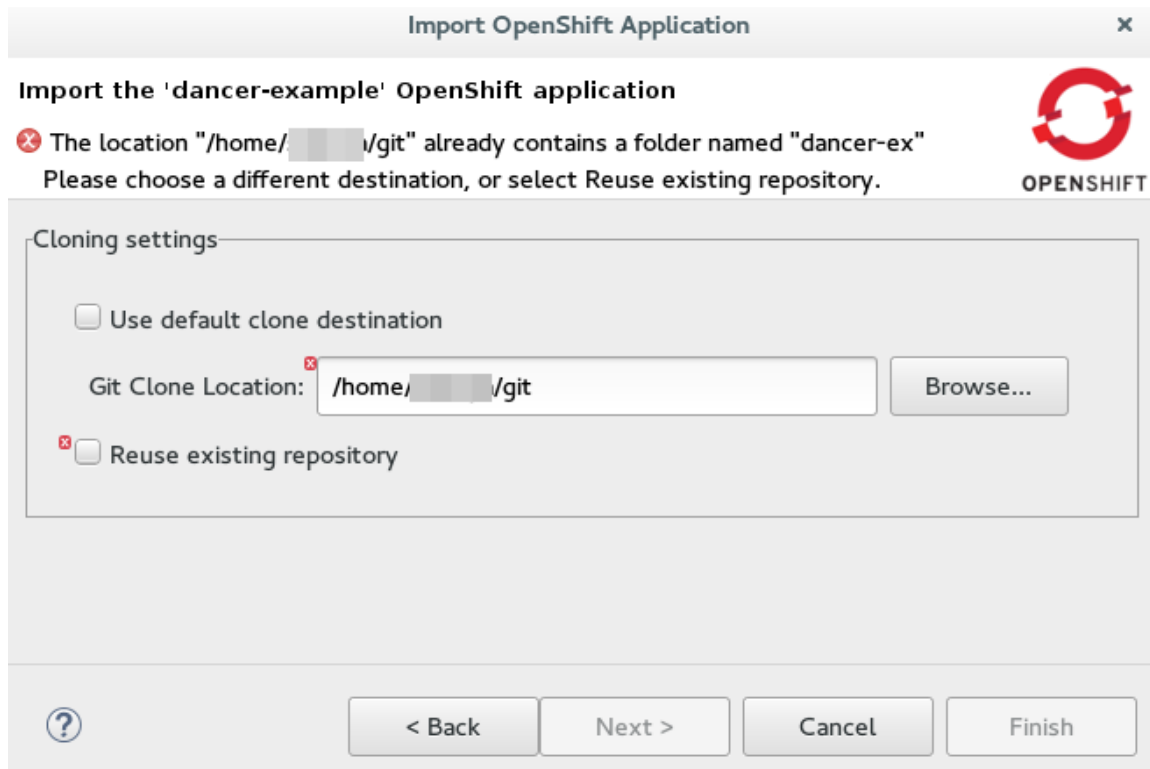
Result: The application appears in the **Project Explorer** view.

2.1.4. Importing an Existing OpenShift Container Platform Application into the IDE

> **Note**
>
> Only an application that has its source specified in the **build config** file can be imported in the workspace.

Applications associated with your OpenShift Container Platform account(s) are listed in the **OpenShift Explorer** view. The source code for these applications can be individually imported into the IDE using the **Import OpenShift Application** wizard. Once imported, the user can easily modify the application source code, as required, build the application and view it in a web browser.

To import an existing OpenShift Container Platform application as a new project in the existing IDE workspace:

1. If required, sign into your OpenShift Container Platform server using the **Basic** protocol or the **OAuth** protocol.

2. In the **OpenShift Explorer** view, expand the connection to locate the application to import.

3. Right-click *{project name}* and click **Import Application**.

Note

To import a particular application from a service, right-click the service and then click **Import Application**. If you right-click a project and click **Import Application**, and if there are more than one build configs with source code under a project, you will be prompted to select the desired application for import because of existence of several applications under one project.

4. In the **Import OpenShift Application** wizard, **Existing Build Configs** list, click the application that you want to import and click **Next**.

5. Ensure the location in the **Git Clone Destination** field corresponds to where you want to make a local copy of the application Git repository and click **Finish**.

Result: The application is listed in the **Project Explorer** view.

2.1.5. Deploying an Application Using the Server Adapter

The server adapter allows incremental deployment of applications directly into the deployed pods on OpenShift Container Platform.

To deploy an application:

1. In the **OpenShift Explorer** view, expand the connection, the project, and then the application.

2. Right-click the *and click* **Server Adapter**. *In the* **Server Settings** *window,* **Services** *section, select the service.*

Note

A workspace project will be selected automatically, if the OpenShift service has a Build Config with a git URL matching the git remote URL of one of the workspace projects.

3. *Click* **Finish**.

Result: The **Servers** *view is the view in focus with the server showing* **[Started, Publishing…]** *followed by the* **Console** *view showing the progress of application publishing.*

Figure 2.5. Console View Showing Application Publication Progress
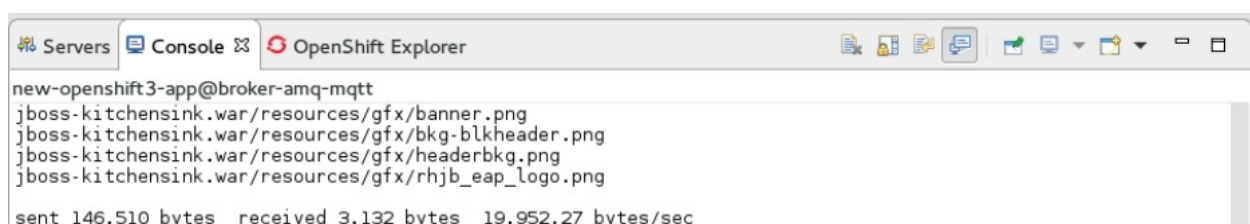
2.1.6. Viewing an Existing Application in a Web Browser

To view an application in the internal web browser, after it has been successfully deployed, in the **OpenShift Explorer** view, right-click the application, and click **Show In → Web browser**.

Result: The application displays in the built-in web browser.

2.1.7. Deleting an OpenShift Container Platform Project

You may choose to delete a project from the workspace to make a fresh start in project development or after you have concluded development in a project. All resources associated with a project get deleted when the project is deleted.

To delete a project:

1. In the **OpenShift Explorer** view, expand the connection and then the project to locate the application you want to delete.

2. Right-click {project name} and click **Delete Project**.

3. In the **OpenShift resource deletion** window, click **OK**.

> **Note**
>
> To delete more than one project (and the containing applications), in the **OpenShift Explorer** view, click the project to select it and while holding the Control key select another project that you want to delete and then press **Delete**.

2.1.8. Did You Know

» Scale the project deployment, using the context menu for the service (the first node below the project). You can also scale the deployment from the **Properties** tab of a deployment (replication controller) and deploymentconfig.

» View the rsync output in the **Console** view. You can also see the progress of the file transfer after you publish local changes to OpenShift Container Platform.

2.2. SETTING UP AND REMOTELY MONITORING AN OPENSHIFT CONTAINER PLATFORM APPLICATION

In some scenarios, the user already has a remote instance of OpenShift Container Platform running with various applications on it and may want to monitor it. The IDE allows users to set up a connection to a remote instance of OpenShift Container Platform and then use logs (application logs and build logs) to troubleshoot and monitor running applications. Connect to and work with a remote OpenShift Container Platform instance by:

1. *Setting up OpenShift Client Binaries*

2. *Setting up Port Forwarding*

3. *Streaming Pod Logs*

4. *Streaming Build Logs*

2.2.1. Setting up OpenShift Client Binaries

Before setting up port forwarding or streaming application and build logs, it is mandatory to set up OpenShift Client Binaries as follows:

1. *In the IDE, click* **Windows** > **Preferences** > **JBoss Tools** > **OpenShift v3**.

2. *Click the* **here** *link.*

3. *In the* **Download from GitHub** *section, click the* **Release page** *link.*

4. *Scroll to the* **Downloads** *section and click the appropriate link to begin the client tools download for the binary for your operating system.*

5. *After the download is complete, extract the contents of the file.*

6. *Click* **Windows** > **Preferences** > **JBoss Tools** > **OpenShift v3**.

7. *Click* **Browse** *and select the location of the OpenShift Client executable file.*

8. *Click* **Apply** *and then click* **OK**.

Result: *OpenShift Client Binaries are now set up for your IDE.*

2.2.2. Setting up Port Forwarding

Using the **Application Port Forwarding** *window, you can connect the local ports to their remote counterparts to access data or debug the application. Port forwarding automatically stops due to any one of the following reasons:*

» *The OpenShift Container Platform connection terminates*

» *The IDE shuts down*

» *The workspace is changed*

Port forwarding must be enabled each time to connect to OpenShift Container Platform from the IDE.

Prerequisite: *Ensure that the OpenShift Client Binaries are set up (see Setting up OpenShift Client Binaries for instructions).*

Set up port forwarding as follows:

1. *In the* **OpenShift Explorer** *view, expand the connection and then expand the project, the services, and then the Pods.*

2. *Right-click the relevant pod and then click* **Port Forwarding**.

   Figure 2.6. Set up Port Forwarding
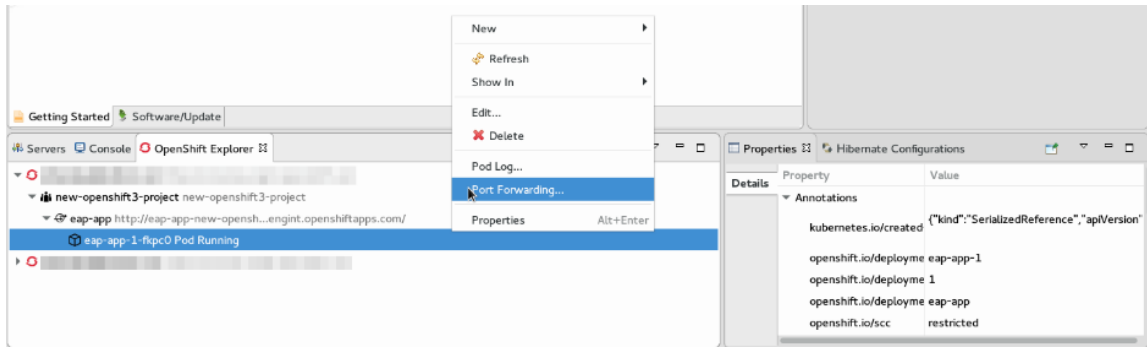
3. In the **Application Port Forwarding** window, click the **Find free local ports for remote ports** check box.

4. Click **Start All**.

Result: The **Status** column shows **Started**, indicating that port forwarding is now active. Additionally, the **Console** view shows the status of port forwarding for the particular service.

Figure 2.7. Start Port Forwarding
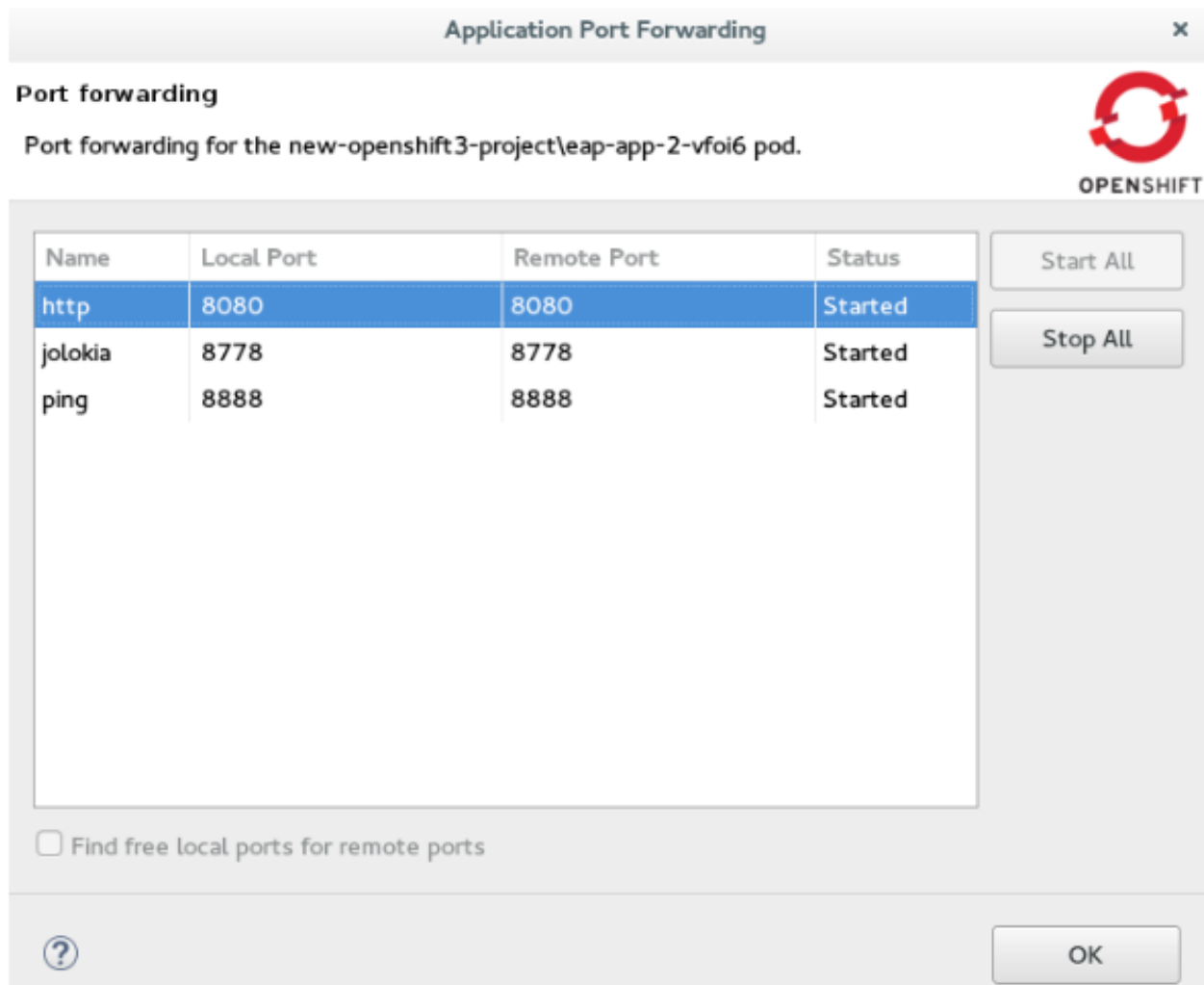
2.2.3. Streaming Pod Logs

Pod logs are general logs for an application running on a remote OpenShift Container Platform instance. The streaming application logs feature in the IDE is used to monitor applications and use the previous pod log to troubleshoot if the application fails or returns errors.

Prerequisite: *Ensure that the OpenShift Client Binaries are set up (see Setting up OpenShift Client Binaries for instructions).*

To stream the application logs:

1. *In the **OpenShift Explorer** view, expand the project, the services, and then the Pods.*

2. *Right-click the relevant Pod and then click **Pod Log**.*
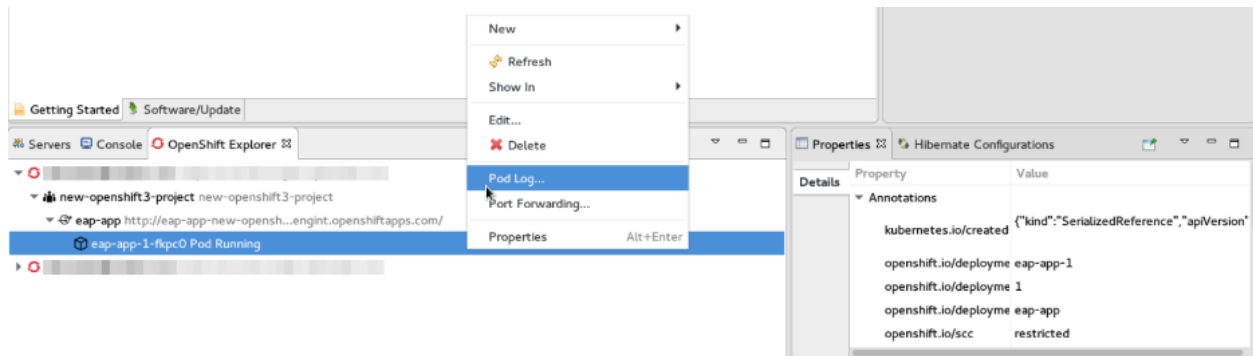
Figure 2.8. Stream Pod Log

Result: *The **Console** view displays the Pod log.*

2.2.4. Streaming Build Logs

Build logs are logs documenting changes to applications running on a remote OpenShift Container Platform instance. The streaming build logs feature in the IDE is used to view the progress of the application build process and to debug the application.

Prerequisite: *Ensure that the OpenShift Client Binaries are set up (see Setting up OpenShift Client Binaries for instructions).*

To stream build logs:

1. *In the **OpenShift Explorer** view, expand the project, the services, and then the build.*

2. *Right-click the relevant build instance and click **Build Log**.*

Figure 2.9. Stream Build Log
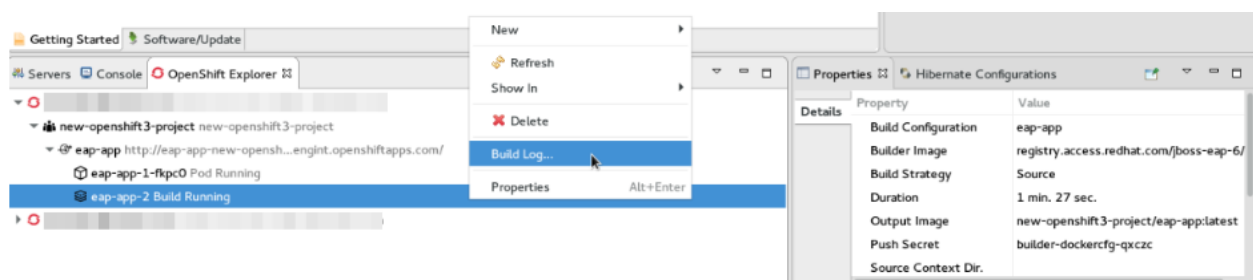
Result: *The **Console** view is now the view in focus showing the build log.*

2.3. BUILDING AND DEPLOYING DOCKER-FORMATTED CONTAINER IMAGE TO CONTAINER DEVELOPMENT KIT

CONTAINER IMAGE TO CONTAINER DEVELOPMENT KIT OPENSHIFT REGISTRY

In this article we deploy the Docker based microservices, **frontend** *and* **bonjour***, into an OpenShift Container Platform instance running on Red Hat Container Development Kit, in JBoss Developer Studio 10. We use the Helloworld-MSA tutorial available in GitHub at: https://github.com/redhat-helloworld-msa/helloworld-msa.*

The article shows how you can easily build a local Docker image, not present on Docker Hub, to Container Development Environment and then deploy that image to an OpenShift Container Platform instance, using JBoss Developer Studio. **frontend** *and* **bonjour** *microservices, used here, are examples of such private images that are not present in Docker Hub.*

You can build and deploy a Docker-formatted Container Image to Container Development Kit OpenShift Registry by:

1. *Section 2.3.2, "Installing the javascript Modules"*

2. *Section 2.3.3, "Building the frontend Microservice"*

    a. *Section 2.3.3.1, "Deploying the frontend Microservice"*

3. *Section 2.3.4, "Connecting the frontend and bonjour Microservices"*

    a. *Section 2.3.4.1, "Deploying the bonjour Microservice"*

    b. *Section 2.3.4.2, "Scalling the Pod"*

4. *Section 2.3.5, "Editing the bonjour Microservice"*

    a. *Section 2.3.5.1, "Viewing the Edited bonjour Microservice on the frontend Microservice"*

2.3.1. Prerequisites

1. *Install nmp: Before running JBoss Developer Studio, install npm on your system. See the npm documentation for instructions for various platforms: https://docs.npmjs.com/getting-started/what-is-npm.*

2. *Download and install JDK 8.*

3. *Install JBoss Developer Studio and Red Hat Container Development Kit.*

    a. *On a Windows system: Install Red Hat Development Suite to automatically install both: JBoss Developer Studio and Red Hat Container Development Kit (for installation instructions, see https://access.redhat.com/documentation/en/red-hat-development-suite/1.1/paged/installation-guide/).*

    b. *On other operating systems: Install JBoss Developer Studio (for installation instructions, see: https://access.redhat.com/documentation/en/red-hat-jboss-developer-studio/10.1/paged/installation-guide/) and install Red Hat Container Development Kit (for installation instructions, see https://access.redhat.com/documentation/en/red-hat-container-development-kit/2.2/paged/installation-guide/).*

4. *Clone the following projects and then import them into JBoss Developer Studio using the* **Import** *wizard (from* **File** *>* **Open Projects from File System***).*

   a. **bonjour** *project from:* https://github.com/redhat-helloworld-msa/bonjour

   b. **frontend** *project from:* https://github.com/redhat-helloworld-msa/frontend

5. *Set up the oc client binaries in the IDE from* **Window** *>* **Preferences***, expand* **JBoss Tools***, and then click* **OpenShift 3***.*

2.3.2. Installing the javascript Modules

To download and install all the required javascript modules:

1. *In the* **Project Explorer** *view, expand* **frontend** *and right-click* `package.json`.

2. *Click* **Run As** *>* **npm Install** *to download and install the required javascript modules in the project.*

Result: *After the build is complete, a new* `node_modules` *folder is listed under the project in the* **Project Explorer** *view.*

2.3.3. Building the frontend Microservice

In this section we build the **frontend** *microservice which is the landing page for the application being built. The* **frontend** *microservice calls other microservices (***bonjour***, in this case) and displays the results from these calls.*

To build the Docker-formatted Container image:

1. *In the* **Project Explorer** *view, expand* **frontend** *and right-click* **Dockerfile** *and then click* **Run As** *>* **Docker Image Build***.*

2. *In the* **Docker Image Build Configuration** *window:*

   a. *In the* **Connection** *list, select* **Container Development Environment***.*

   b. *In the* **Repository Name** *field, type* **demo/frontend***.*

3. *Click* **OK***.*

Result: *The Docker-formatted Container image starts building against the Docker Daemon running in the Container Development Environment.*

2.3.3.1. Deploying the frontend Microservice

After the build is complete, the Docker-formatted Container image **demo/frontend** *is available in the* **Docker Explorer** *under* **Container Development Environment***.*

To deploy the frontend microservice:

1. *In the* **Docker Explorer** *view,* **Container Development Environment** *>* **Images***, right-click* **demo/frontend** *and click* **Deploy to OpenShift***.*

2. In the **Deploy an Image** window, click **New**.

3. In the **Create OpenShift Project** window:

   a. In the **Project Name** field, type the name of the new project, **demo**.

   b. Optionally, in the **Display Name** and **Description** fields, enter the required details.

   c. Click **OK**.

4. In the **Deploy an Image** window, click the **Push Image to Registry** check box and click **Next**.

5. In the **Deployment Configuration & Scalability** window, change the following environment variables:

   a. Click **OS_PROJECT** to open the **Environment Variable** window and in the **Value** field, type **demo** (from step 5) and click **OK**.

6. In the **Deployment Configuration & Scalability** window, click **Next** and then click **Finish**. After the Docker-formatted Container image is pushed to the Docker Registry on OpenShift Container Platform, the Eclipse plugin generates all the required OpenShift Container Platform resources for the application to run.

7. In the **Deploy Image to OpenShift** window, review the details of deploying the image and click **OK**.

8. In the **OpenShift Explorer** view, expand the connection > **> Service > Pod to see the Pod running. Right-click the Pod and click Pod Log. The Console view shows the frontend service running. In the OpenShift Explorer view, expand the application and right-click the service and click Show In > Web Browser.**

Result: The frontend microservice, in the Bonjour Service shows: Error getting value from service <microservice> meaning the bonjour microservice must be connected.

2.3.4. Connecting the frontend and bonjour Microservices

In this section we build the bonjour microservice and then view it on the frontend microservice. The bonjour microservice is a simple node.js application that returns the string bonjour-de-<pod_ID>.

To connect the Microservices:

1. **In the Project Explorer view, expand bonjour and right-click package.json.**

2. **Click Run As > npm Install.**

3. **In the Project Explorer view, expand bonjour and right-click Dockerfile.**

4. **Click Run As > Docker Image Build.**

5. **In the Docker Image Build Configuration window:**

   a. **In the Connectio*n list, select *Container Development Environment.**

   b. **In the Repository Name field, type demo/bonjour.**

6. *Click OK.*

2.3.4.1. Deploying the bonjour Microservice

You can either deploy the Docker-formatted Container image from the Docker Explorer (as done in step 3 of the Building a Docker-formatted Container Image section above), or in the following way from the OpenShift Explorer view:

1. *In the OpenShift Explorer view, right-click the project (demo), and click Deploy Docker Image.*

2. *In the Deploy an Image window:*

   a. *In the Docker Connection list, click the Docker connection.*

   b. *In the Image Name field, type* `demo/bonjour`.

   c. *Click the Push Image to Registry check box.*

3. *Click Next.*

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

5. *In the Services and Routing Settings window, click Finish.*

6. *In the Deploy Image to OpenShift window, click OK.*

2.3.4.2. Scalling the Pod

To see the bonjour service with the Pod running:

1. *In the OpenShift Explorer view, expand the application name (demo).*

2. *Right-click the pod and click Pod Log to check if the pod is running.*

3. *Navigate to the browser where you have the application running and click Refresh Results. You will see a greeting from the bonjour service with a hostname that matches the Pod name in the OpenShift Explorer view.*

4. *In the OpenShift Explorer view, right-click the service and click Scale > Up. You now have two Pods running on OpenShift Container Platform.*

Result: Navigate to the browser and click Refresh Results to see the service balancing between the two Pods.

2.3.5. Editing the bonjour Microservice

In this section we edit the bonjour microservice and then view the results on the frontend microservice.

To edit the bonjour microservice:

1. *In the Project Explorer view, expand bonjour, and double-click bonjour.js to open it in the default editor.*

2. **Find**

```
function say_bonjour(){
    Return "Bonjour de " + os.hostname();
```

3. **Change it to:**

```
    function say_bonjour(){
    Return "Salut de " + os.hostname();
```

4. **Save the file.**

2.3.5.1. Viewing the Edited bonjour Microservice on the frontend Microservice

After you have edited the bonjour microservice:

1. **In the Project Explorer view, expand bonjour, and right-click Dockerfile.**

2. **Click Run As > Docker Image Build.**

   Note

   Here, the Docker run configuration, the connection, and the repository name used earlier are being reused. To edit the configuration, open the Run Configuration window.

   After the Console view shows that the Docker-formatted Container image has been successfully pushed to the Docker Daemon:

3. **In the Docker Explorer view, expand Container Development Environment > Images.**

4. **Right-click the image and click Deploy to OpenShift.**

5. **In the Deploy an Image window, click Push Image to Registry and then click Next.**

6. **In the Deployment Configuration & Scalability window, click Finish. The OpenShift Explorer view, under bonjour shows the Pods being added and then running. Navigate to the browser and click Refresh Results.**

Result: The new greeting appears.

2.3.6. Troubleshooting

2.3.6.1. No Docker Connection Available

Error message: No Docker Connection available to build the image.

Issue: You have installed JBoss Developer Studio through Red Hat Development Suite and you must start Red Hat Container Development Kit for it to be available. Resolution:

1. **In the Servers view, right-click Container Development Environment and click Start.**

2. *Enter your credentials in the box provided.*

If, after doing this the Container Development Environment does not start and you get the following error: Error message: Server Container Development Environment failed to start.

On the command prompt, cd to `cdk/components/rhel/rhel-ose` and run the vagrant destroy command. After it is destroyed, run the vagrant up command. In the IDE, in the Servers view, right-click Container Development Environment and click Start once again.

CHAPTER 3. DEVELOPING FOR THE CLOUD WITH OPENSHIFT 2

3.1. CREATING YOUR FIRST OPENSHIFT ONLINE APPLICATION

OpenShift Tools provides an all-in-one `New OpenShift Application` wizard for creating new OpenShift applications from templates and existing projects. This wizard is the starting point for creating all new OpenShift Online applications from the IDE and also for importing OpenShift Online applications to your workspace.

The `New OpenShift Application` wizard is an ideal starting point for new users to OpenShift and OpenShift Tools as it guides you through all the steps necessary to set up the IDE to use your OpenShift Online account and configure your account ready for applications.

The instructions here demonstrate how to use this wizard and a default OpenShift application template to create a basic OpenShift Online application. This includes one-time steps, such as signing up for an OpenShift Online account, creating an OpenShift Online domain and uploading SSH keys. If you have previously used OpenShift Online or OpenShift Tools, you can omit unnecessary one-time steps as appropriate.
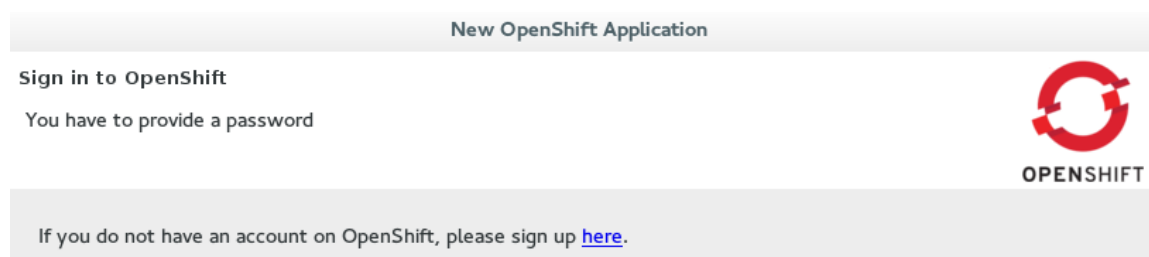
3.1.1. Starting the New OpenShift Application Wizard

1. In `JBoss Central`, under `Start from scratch`, click `OpenShift Application`.

3.1.2. Signing up for an OpenShift Online Account

If you do not have an OpenShift Online account, you should complete the following steps:

1. Click the link provided to sign up for an OpenShift account and follow the instructions on the OpenShift website.

   Figure 3.1. Link to Sign up for a New OpenShift Online User Account

2. When you have completed the sign-up process, restart the `OpenShift Application` wizard from JBoss Central.

3.1.3. Connecting to OpenShift Online

1. Complete the fields about your OpenShift Online account as follows:

   a. From the `Connection` list, select `New Connection`.

b. *Ensure the disabled `Server` field states `https://openshift.redhat.com`.*

c. *In the `Username` and `Password` fields, type your account credentials.*

Figure 3.2. Connection Information Provided for OpenShift Online Account

2. *Click `Next`.*

3.1.4. Enabling Communication between the IDE and OpenShift

If your OpenShift Online account has no SSH public keys stored, you are prompted with the Add SSH Keys wizard and you should complete the following steps:

1. *Click `New`.*

2. *Complete the fields about the SSH Keys to be created as follows:*

   a. *In the `Name` field, type a name for the SSH key pair.*

   b. *From the `Key Type` list, ensure SSH_RSA is selected.*

   c. *In the `SSH2 Home` field, ensure your .ssh directory path is correct.*

   d. *In the `Private Key File Name` field, type a name for the private key file name. The `Public Key File Name` field populates automatically with the name of the private key file name with .pub appended.*

   Figure 3.3. New SSH Key Information for OpenShift Online Provided

3. **Click Finish.**

4. **Click Finish to close the Add SSH Keys window.**

3.1.5. Creating a Domain

If your OpenShift Online account has no domains, you are prompted with the Create Domain wizard and you should complete the following step:

1. **In the Domain Name field, type an alphanumeric name for your new OpenShift Online domain and click Finish. The provided domain name must be unique across all domains on OpenShift Online; if it is not unique, you are directed back to the Create Domain window to provide a unique domain name.**

   Figure 3.4. New OpenShift Domain Name Provided

3.1.6. Providing Essential New Application Details

1. *Complete the fields about the type of OpenShift application you want to create as follows:*

   a. *Ensure `Create a new OpenShift application` is selected.*

   b. *Expand `Basic Cartridges` and select `JBoss Application Server 7`.*

      Figure 3.5. Basic Cartridge Selected for the New OpenShift Online Application

2. *Click Next.*

3. *Complete the fields about your OpenShift application as follows:*

a. *Ensure the `Domain` field displays the OpenShift Online domain with which you want to host your application.*

b. *In the `Name` field, type an alphanumeric name for your application.*

c. *From the `Gear profile` list, select `small`.*

d. *Select the `Enable scaling` check box.*

Figure 3.6. New OpenShift Application Information Provided

4. *Click `Next`.*

3.1.7. Configuring the Corresponding Workspace Project for the New Application

1. *Complete the fields about the corresponding workspace project as follows:*

   a. *Ensure the `Create a new project` check box is selected.*

   b. *Ensure the `Create and set up a server for easy publishing` check box is selected. This automatically creates an OpenShift server adapter for the application, enabling you to easily publish project changes to the OpenShift server.*

2. *Click `Next`.*

3. *Ensure the location in the `Git Clone Destination` field corresponds to where you want to make a local git repository for the project source code.*

Figure 3.7. Git Clone Destination Specified

3.1.8. Creating the OpenShift Online Application

18. **Click `Finish` for the wizard to start generating the new OpenShift application. This process may take some time to complete.**

19. **If you are prompted that the authenticity of the host cannot be established and asked whether you want to continue connecting, ensure that the host name matches that of your application and domain and click `Yes`.**

3.1.9. Viewing the OpenShift Online Application

20. **In the `OpenShift Explorer` view, expand the connection and domain.**

21. **Right-click `{application name}` and click Show In → Web Browser.**

Your OpenShift Online application is displayed in the IDE default web browser.

3.1.10. Some OpenShift Terminology

» **Gear: A server container with a set of resources that allow you to run your application**

» **Cartridge: Plug-ins that house the framework or components that can be used to create and run your application**

  » **Standalone cartridge: Languages and application servers that serve your application**

  » **Embedded cartridge: Functionality to enhance your application**

» **Scaling: Enables your application to react to changes in traffic and automatically allocate the necessary resources to handle the current demand**

3.1.11. Did You Know?

❯ **You can also start the `New OpenShift Application` wizard from the `OpenShift Explorer` view by right-clicking a `connection`, domain or existing application and clicking New → Application or from the IDE main menus by clicking File → New → OpenShift Application.**

❯ **To save time when logging in to OpenShift Online in future, you can click the `Save Password` check box in the `Sign in to OpenShift` window. The password is retained in secure storage provided by the IDE and automatically populates the `Password` field for the associated connection.**

❯ **Using the `New OpenShift Application` wizard, you can also create a new OpenShift application from an existing workspace project or a Git source.**

❯ **Each time you start the IDE or switch workspaces the IDE is initially disconnected from OpenShift. When you attempt to complete an action that requires an active OpenShift connection, you are automatically prompted to reconnect.**

3.2. DEVELOPING AN EXISTING OPENSHIFT APPLICATION

OpenShift Tools enables you to import existing OpenShift Online applications into the IDE so that you can take advantage of the IDE features in further developing your applications. As illustrated in this article, during the import process you can configure the IDE for easy republishing to OpenShift Online.

Develop an existing OpenShift application by:

1. **Importing an Existing OpenShift Online Application**

2. **Modifying the Application Source Code**

3. **Republishing the Modified Application**

4. **Viewing the Modified Application**

3.2.1. Importing an Existing OpenShift Online Application

Applications associated with your OpenShift Online account(s) are listed in the `OpenShift Explorer` view. The source code for these applications can be individually imported into the IDE using the OpenShift Tools `Import OpenShift Application` wizard.

To import an existing OpenShift Online application as a new project in the existing IDE workspace, complete the following steps:

1. **In the `OpenShift Explorer` view, ensure your OpenShift Online connection is listed or click the `Connection` icon** **and complete your OpenShift Online account details to create a new connection.**

2. **In the `OpenShift Explorer` view, expand the connection and domain to locate the application you want to import.**

3. **Right-click `{application name}` and click `Import Application`.**

4. *Complete the fields about the application to be imported as follows:*

   a. *Ensure that `Use my existing OpenShift application` is selected.*

   b. *Ensure that the name of the application you want to import is listed. If this is not the case, type the name of the application or click `Browse` to select the application.*

Figure 3.8. Existing OpenShift Online Application Information Provided

5. *Click `Next`.*

6. *Complete the fields about the corresponding new workspace project as follows:*

   a. *Ensure the `Create a new project` check box is selected.*

   b. *Ensure the `Create and set up a server adapter for easy publishing` check box is selected.*

7. *Click `Next`.*

8. *Ensure the location in the `Git Clone Destination` field corresponds to where you want to make a local copy of the OpenShift Online application Git repository.*

9. *Ensure a public SSH key is uploaded to OpenShift Online and private key location is specified in the IDE preferences by clicking `SSH Keys Wizard` and reviewing the information.*

10. *Click `Finish`.*

If you are prompted that the authenticity of the host cannot be established and asked whether you want to continue connecting, ensure that the host name matches that of your application and domain and click Yes.

When the import process is complete, the project is listed in the `Project Explorer` view and a server adapter is listed for the application in the `Servers` view.

OpenShift Tools makes a number of small changes to the application source code on import. It adds several IDE-specific files to the project and modifies the `.gitignore` file so that you are not prompted about these files each time you commit project changes to the Git repository.

3.2.2. Modifying the Application Source Code

The files contained in your project depend on the type of project that you have imported. Here a common OpenShift application file, `index.html`, is changed as an example of modifying the project source code. You can opt to change this file or another file of your project.

To modify the `index.html` file, complete the following steps:

1. *In the `Project Explorer` view, expand `{project name}` → `src` → `main` → `webapp`.*

2. *Double-click `index.html` to open it in the JBoss Tools HTML Editor.*

3. *After the opening <body> tag add the following line:*

   ```
   <h1>This is a change made to my OpenShift Online app from the
   IDE.</h1>
   ```

4. *Save the `index.html` file by pressing `Ctrl+S` (or `Cmd+S`).*

Note that in the `Project Explorer` view, `index.html` has > prepended to show that the source code has changed since the last Git commit.

3.2.3. Republishing the Modified Application

You must commit and push any changes you have made to the project source code and then republish the application before you can see changes reflected in the OpenShift Online application. As an example of committing and pushing source code changes to the OpenShift Online application repository and republishing the application, here the `index.html` file changed earlier is used. You can opt to use the project file that you changed earlier.

To commit and push the `index.html` changes and republish the application to OpenShift Online, complete the following steps:

1. *In the `Servers` view, right-click `{application name} at OpenShift` and click `Publish`. Or you can drag the from the `Project Explorer` view and drop it on the server adapter in the `Servers` view.*

2. *Complete the fields about the project changes to commit and push to the OpenShift Online application Git repository as follows:*

a. *In the `Commit message` field, type the following message*

> *Added new heading in body of index.html*

b. *In the table of files, ensure the `index.html` file is selected.*

Figure 3.9. Commit Message Provided and Changed Project File Selected
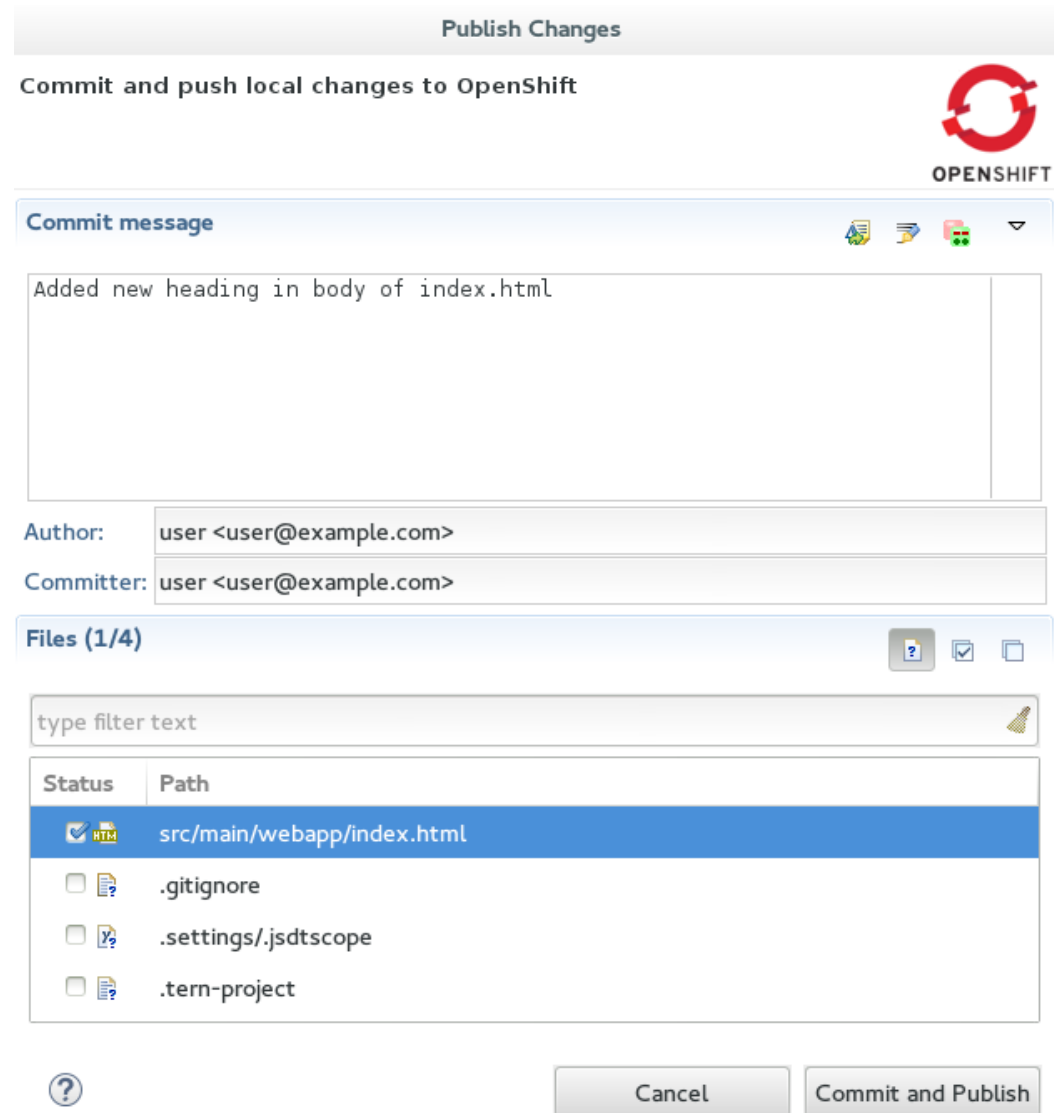
3. *Click `Commit and Publish`.*

The republishing process can take some time. When the `Console` view shows `Deployment completed with status: success`, the republishing process is complete.

3.2.4. Viewing the Modified Application

After the modified application is republished, you can view the live updated version at the OpenShift Online application URL.

To view the modified OpenShift Online application in the IDE default web browser, complete the following steps:

1. *In the `OpenShift Explorer` view, expand the connection and the domain.*

2. *Right-click {application name} and click Show In → Web Browser.*

Your modified and republished OpenShift Online application is displayed in the IDE default web browser.

3.2.5. Did You Know?

» *On the first occasion that you republish your application to OpenShift Online, you may be prompted to provide a username and email address for use by Git. The IDE looks for a default Git configuration file on your system from which to obtain this information and if the IDE cannot find the file it prompts you to provide the values. You can specify the location of your system Git configuration file in the IDE Preferences, under Team → Git → Configuration.*

» *You can review the progress of republishing applications to OpenShift Online in the `Console` view.*

» *You can also open the OpenShift application in a web browser from the `Servers` view by right-clicking {application name} at OpenShift and clicking Show In → Web Browser.*

» *You can change the IDE default web browser to be either the IDE internal web browser, BrowserSim (when installed) or an external web browser. Click Window → Web Browser and select from the available web browser options or click Window → Preferences → General → Web Browser to extend the list of available external web browsers.*

3.3. CUSTOM PUBLISHING YOUR OPENSHIFT APPLICATION

Through the IDE you can manage your OpenShift Online application beyond simply its source code. OpenShift Online allows customization of the build and deployment process with markers and action hooks, each of which can be added to the application configuration files using OpenShift Tools and existing IDE functionality. Further, the Git commit and push processes for your changed application source code and configuration files can be customized through the IDE preferences to meet your needs.

Custom publish your OpenShift application by:

1. *Adding a Marker to the Application*

2. *Adding an Action Hook to the Application*

3. *Extending the Git Remote Connection Timeout*

4. *Republishing the Application*

3.3.1. Adding a Marker to the Application

OpenShift Tools provides the `Configure Markers` wizard for adding markers to and removing them from your OpenShift application. The wizard lists markers that are already applied to your application and those that are available to add given the cartridge configuration of your application. The wizard manages the adding and removing of markers from your application, creating or deleting marker files and adding or removing them from the Git index respectively.

As an example, here the hot deploy marker is added to the application, which triggers OpenShift to publish application changes without first restarting the application cartridges and hence making the republishing faster.

To add the hot deploy marker to the application, complete the following steps:

1. *In the `Project Explorer` view, right-click `{project name}` and click OpenShift → Configure Markers.*

2. *From the `Marker` table, select `Hot Deploy` and click `OK`.*

Figure 3.10. Hot Deploy Marker Selected
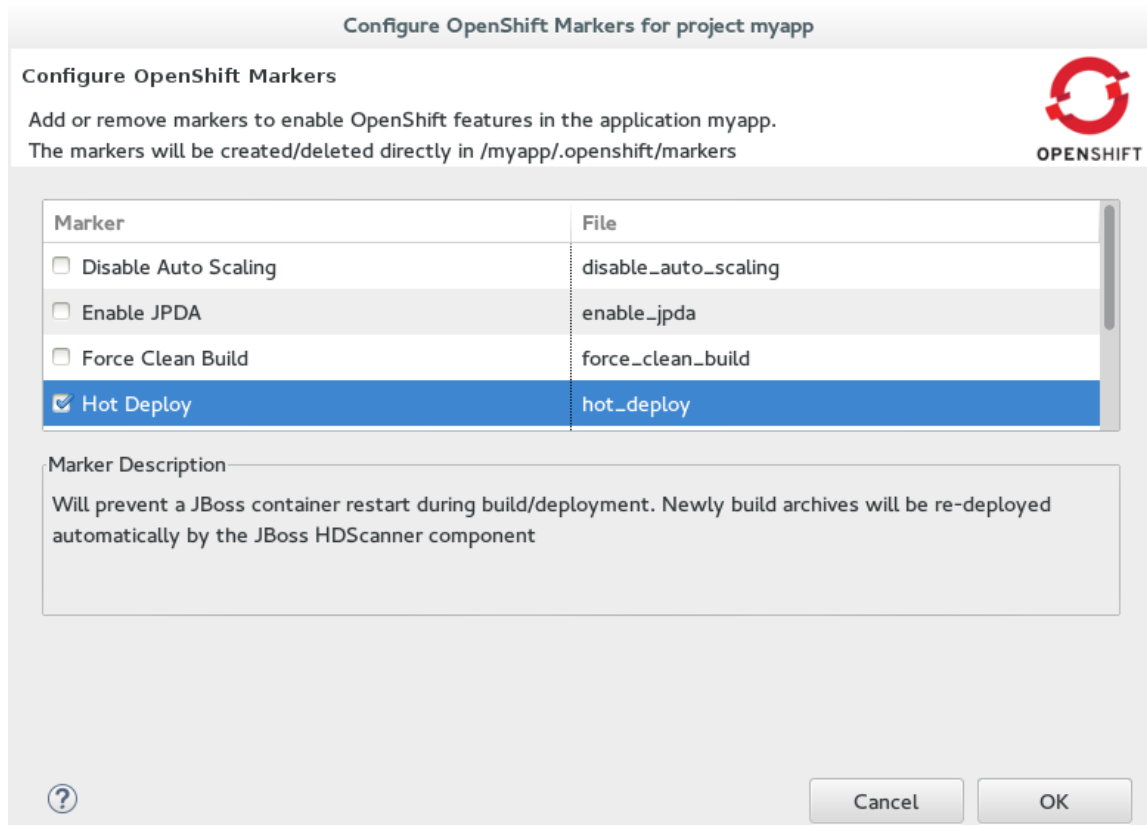
An empty `.openshift/markers/hot_deploy` file is added to the application. OpenShift Tools automatically completes the `git add` action so that this new file is added to the Git index and can be committed and pushed to the OpenShift application repository when ready.

3.3.2. Adding an Action Hook to the Application

Using IDE features, you can quickly add action hooks to your OpenShift application. Adding an action hook requires creating a script file named according to the phase in which it is to run, locating it in the application `.openshift/action_hooks` directory, adding the file to the Git index and ensuring the file is executable by all.

As an example, here a post deploy action hook is added to the application, which triggers a simple bash script to execute on the application main gear after the application is deployed.

To add a post deploy action hook to the application, complete the following steps:

Create the post deploy action hook

1. *In the `Navigator` view, expand `{project name}` → `.openshift`.*

2. *Right-click `action_hooks` and click New → File.*

3. *In the `File` name field, type `post_deploy` and click `Finish`.*

4. *In the file editor, add the following lines to the post_deploy file:*

    ```
    #!/bin/bash
    echo "This is my post-deploy bash script"
    ```

5. *Save the file by pressing `Ctrl+S` (or `Cmd+S`).*

Make the post deploy action hook executable

6. *In the `Navigator` view, right-click the `post_deploy` file and click `Properties`.*

7. *In the `Permissions` table, select the `Execute` check boxes for all user types.*

Figure 3.11. Execute Permissions Check Boxes Selected for All Users
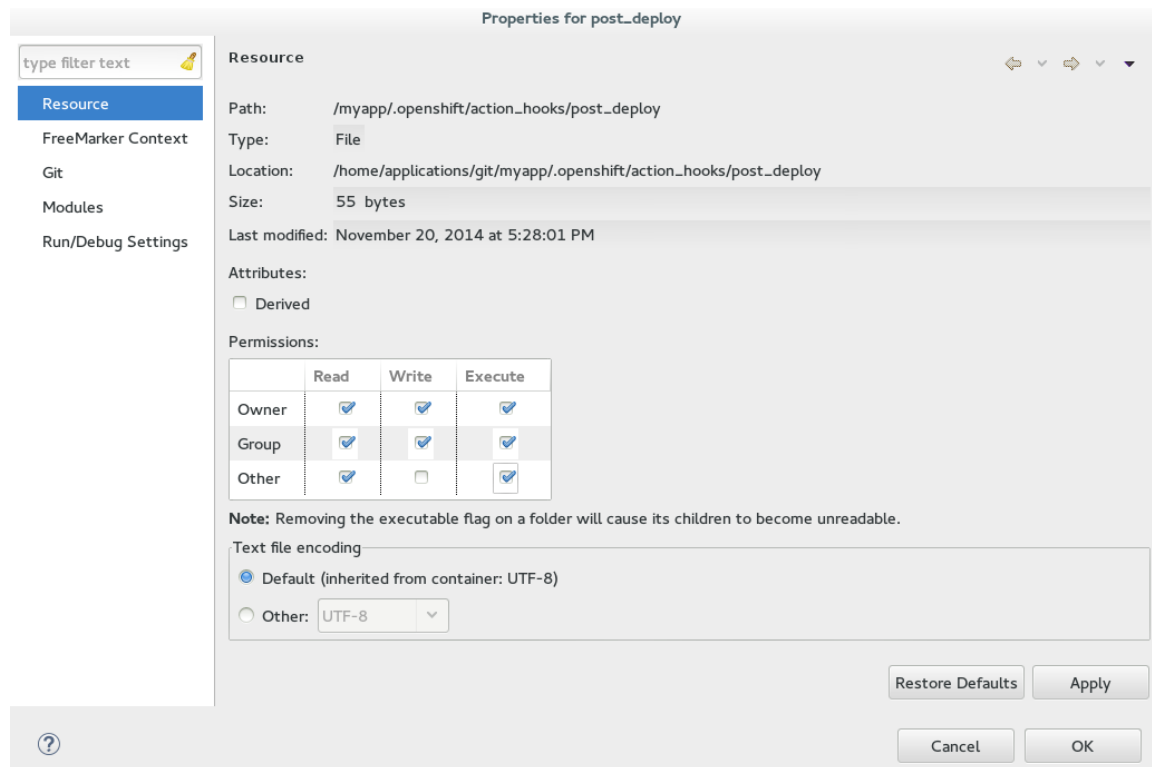
8. *Click `Apply` and click `OK`.*

3.3.3. Extending the Git Remote Connection Timeout

As your application source code grows in size, the remote connection from the IDE to the OpenShift Online application Git repository may not remain open long enough for the push process to complete. The default Git remote connection timeout is set to 30 seconds after which the connection is closed. But you can extend the Git timeout through the IDE preferences to ensure that the push process is provided with sufficient time to complete.

To extend the Git remote connection timeout, complete the following steps:

1. *Click Window → Preferences → Team → Git.*

2. *In the `Remote connection timeout (seconds)` field, type a value in seconds.*

Figure 3.12. Git Remote Connection Timeout Set
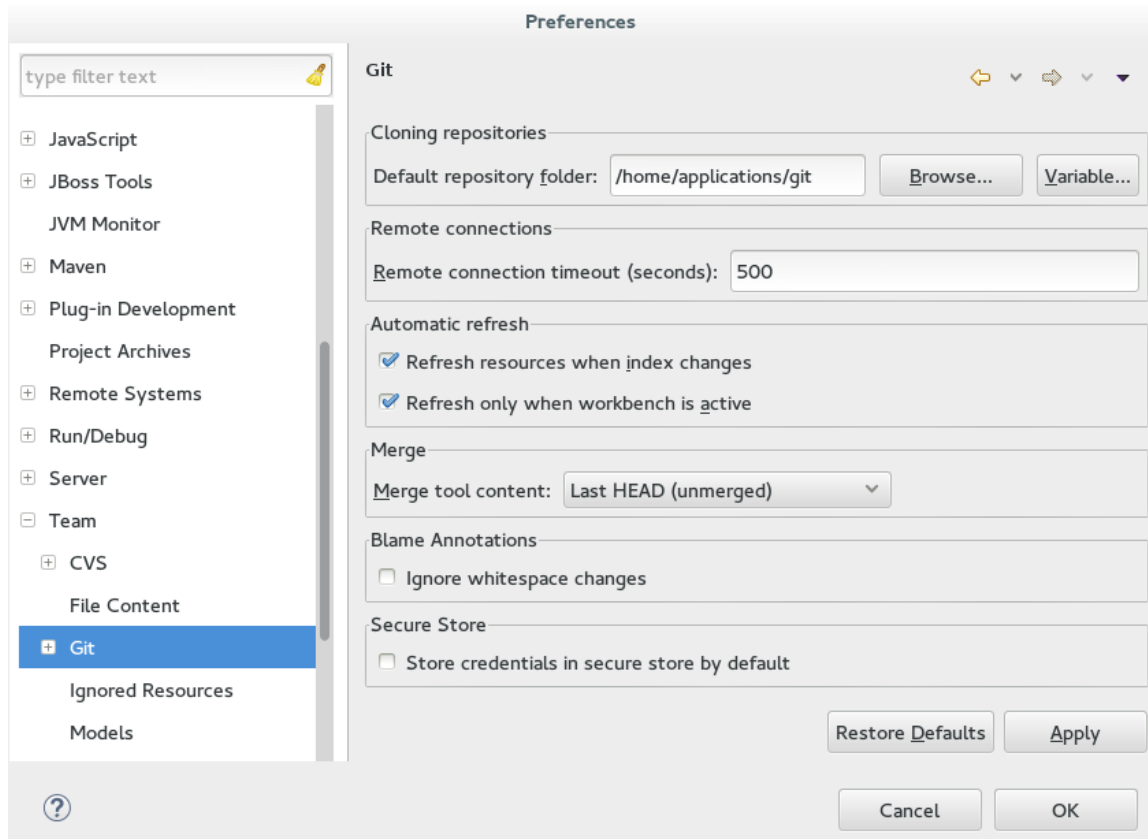
3. *Click `Apply` and click `OK` to close the `Preferences` window.*

3.3.4. Republishing the Application

Before markers and action hooks take effect on your application building and deployment process you must commit the files and push changes to the OpenShift application repository. OpenShift Tools assists you to perform these Git actions as part of the republishing process.

To republish the application, complete the following steps:

1. *In the `Servers` view, right-click `{application name} at OpenShift` and click `Publish`.*

2. *Complete the fields about the application changes to commit and push to the OpenShift Online application repository as follows:*

   a. *In the `Commit message` field, type the following message:*

   > *Added hot_deploy marker and post_deploy bash script*

b. *In the `Files` table, ensure the `hot_deploy` file is selected and select the `post_deploy` file.*

Figure 3.13. Commit Message Supplied and Files Selected for Committing and Publishing
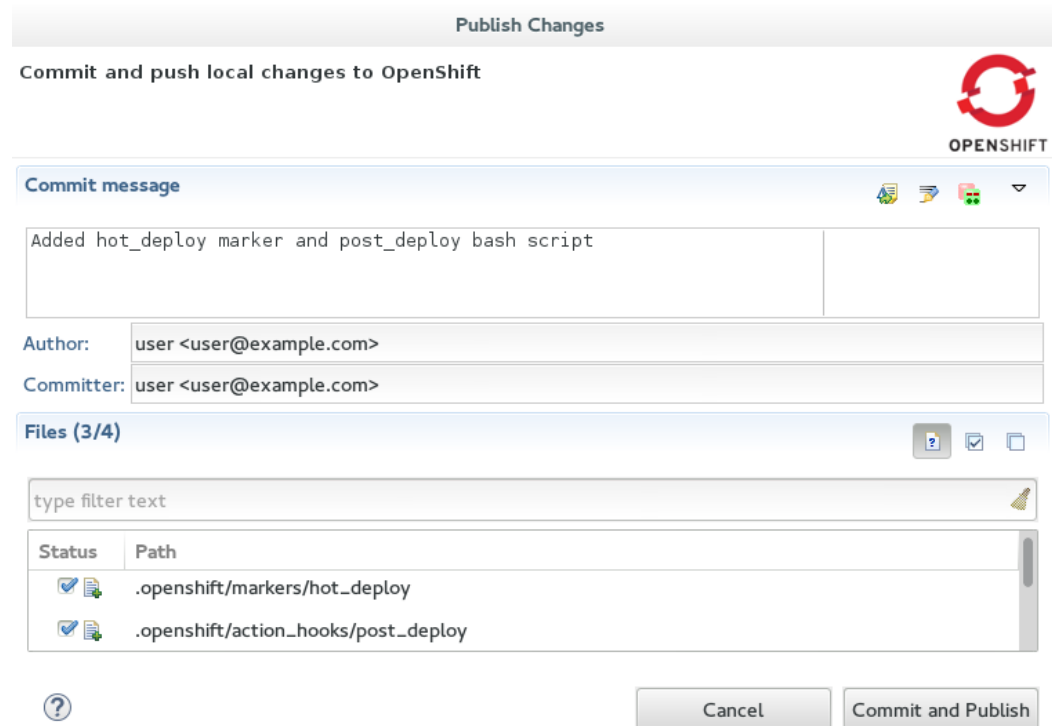
3. *Click `Commit and Publish`.*

The `Console` view becomes the view in focus showing the application publication progress. A snippet of the output demonstrating the effect of the hot deploy marker and post deploy action hook is shown here for the sample application:

```
Not stopping cartridge xyz because hot deploy is enabled
Building git ref 'master', commit abc123

Activating deployment
Deploying xyz cartridge
Not starting cartridge xyz because hot deploy is enabled
This is my post-deploy bash script
-----------------------
Git Post-Receive Result: success
Activation status: success
Deployment completed with status: success
```

Note that the server has not been stopped and restarted because hot deploy is enabled and the bash script is run post application deployment as required.

3.3.5. Terminology

❯ *Git: The revision control system used by OpenShift.*

❧ *Marker: A set-named empty file added in the OpenShift application in the `.openshift/markers` directory; markers are used to specify configuration to the OpenShift server.*

❧ *Action hook: A user-specified script that is added to the OpenShift application, in the `.openshift/action_hooks` directory; scripts are run by OpenShift at specified stages of the application build and deploy process as denoted by the file name.*

3.3.6. Did You Know?

❧ *You can also access the `Configure Markers` wizard from the `Server` view by right-clicking `{application name}` at OpenShift and clicking OpenShift → Configure Markers.*

❧ *You can add files to the Git index at any time by right-clicking the file in, for example, the `Navigator` view and clicking Team → Add to Index.*

❧ *You can see more information relating to the application Git repository by opening the `Git` perspective or individual Git views. All of these can be assessed from the `Window` menu.*

3.4. DEBUGGING AN OPENSHIFT JAVA APPLICATION

OpenShift Tools enables you to debug your deployed OpenShift applications within the IDE, enabling you to take advantage of the IDE debugging tools. This article specifically details the steps needed to set up an OpenShift Online Java application for debugging. A number of configuration tasks are required both locally and remotely to enable the IDE debugger to connect to the OpenShift server and OpenShift Tools for achieving this. Some tasks only need to be completed once for each OpenShift Online application but others must be completed every time you reconnect to OpenShift Online from the IDE.

Debug an OpenShift Java application by:

1. *Configuring the OpenShift Application for Debugging*

2. *Enabling Port Forwarding for the Local and Remote Debug Ports*

3. *Configuring and Connecting the IDE Debugger*

This article guides you through each of these configuration requirements and must be completed in the given order.

> *Note*
>
> *Your application must be deployed on OpenShift before attempting to configure the OpenShift application for debugging and enabling port forwarding.*

### 3.4.1. Configuring the OpenShift Application for Debugging

You must first configure your OpenShift application for debugging, which requires setting the `Enable JPDA` (Java Platform Debugger Architecture) marker in your application source code and republishing the application. Marker information is retained with the application so you only need to complete this task once for each OpenShift Online application:

To configure the OpenShift application for debugging, complete the following steps:

1. *In the `Project Explorer` view, right-click `{project name}` and click OpenShift → Configure Markers.*

2. *In the `Configure OpenShift Markers` window, select the `Enable JPDA` check box and click `OK`.*

   Figure 3.14. OpenShift Enable JPDA Marker Selected
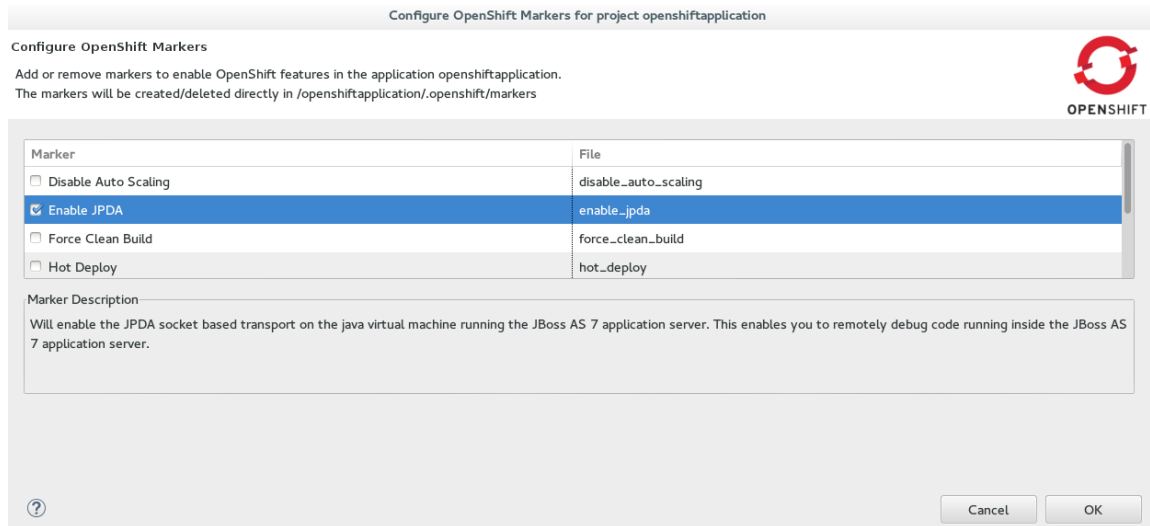
3. *In the `Servers` view, right-click `{application name} at OpenShift` and click `Publish`.*

4. *In the `Publish Changes` window, in the `Commit message` field type a message for the commit.*

5. *From the `Files` list, ensure the `.openshift/markers/enable_jpda` check box is selected and click `Commit and Publish`.*

The project changes are pushed to the remote Git repository and the application is automatically updated on the OpenShift server.

> *Note*
>
> *When debugging is enabled on the OpenShift application a debug port is assigned; for default Java applications the debug port is 8787. To perform the remaining tasks, it is important to know which port is the debug port. To identify the debug port for other applications, see the cartridge documentation.*

3.4.2. Enabling Port Forwarding for the Local and Remote Debug Ports

After the OpenShift application is configured for debugging, you must enable port forwarding for the local (IDE) and remote (OpenShift server) debug ports. You can achieve this with the OpenShift Tools `Application port forwarding` wizard, which connects all local ports to their remote counterpart ports, including the local and remote debug ports.

Port forwarding is automatically stopped when your OpenShift Online connection closes; this includes closing the IDE or changing workspaces. You must enable port forwarding
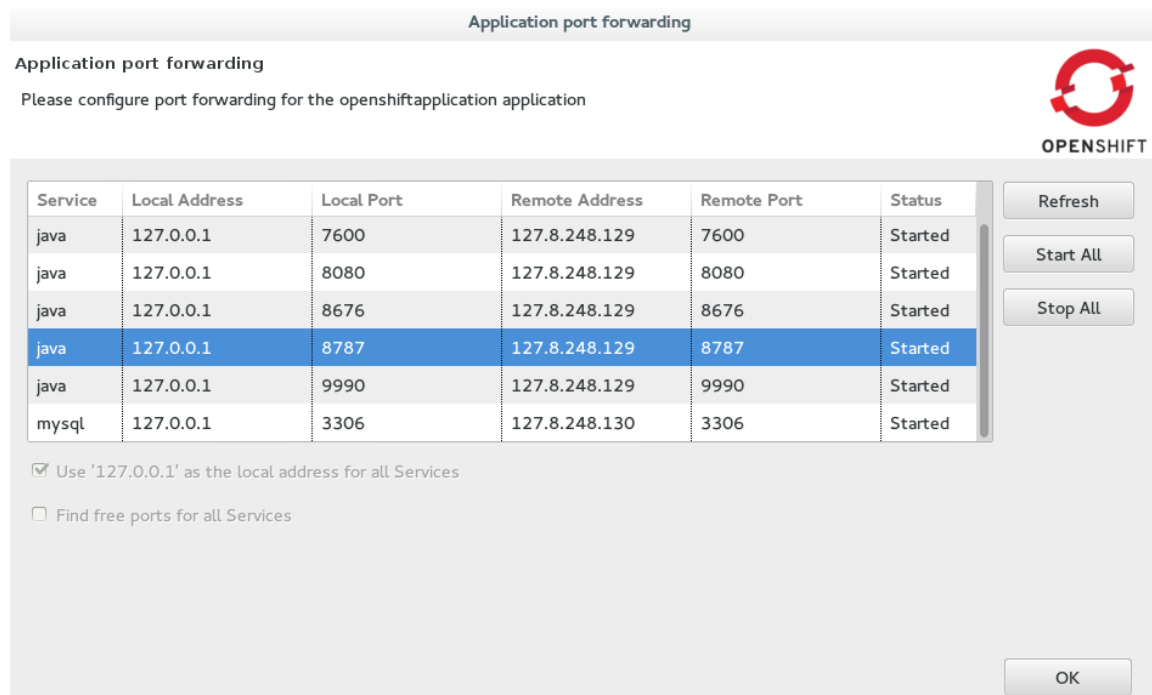
every time you reconnect to OpenShift Online from the IDE.

To enable port forwarding, complete the following steps:

1. *In the `OpenShift Explorer` view, right-click `{application name}` and click `Port Forwarding`.*

2. *In the `Application port forwarding` window, click `Start All`. Ensure the `Status` value shows `Started` for the debug ports and click `OK` to close the `Application port forwarding` window.*

Figure 3.15. Port Forwarding Started for All Ports

3.4.3. Configuring and Connecting the IDE Debugger

With port forwarding configured for the debug ports, you must create a debug configuration for the OpenShift server and connect the IDE debugger. You can then review debug output in the `Debug` and `Console` views. The debug configuration is retained with the workspace so you only need to create a new debug configuration once for each OpenShift Online application. But you must restart each debug configuration every time you reconnect to OpenShift Online from the IDE.

To configure and connect the IDE Debugger, complete the following steps:

1. *In the global toolbar of the JBoss perspective, click the drop-down list for the Debug icon and select `Debug Configurations`.*

2. *From the debug configurations list, select `Remote Java Application` and click the New launch configuration icon*

3. *In the `Connect` tab, complete the following fields:*

a. *In the `Project` field, type the name of the workspace project associated with the OpenShift application or click `Browse` to locate the project.*

b. *In the `Port` field, type the value of the debug port.*

c. *Click `Apply` and then click `Debug`.*

Figure 3.16. Debug Configuration Ready for Your OpenShift Application
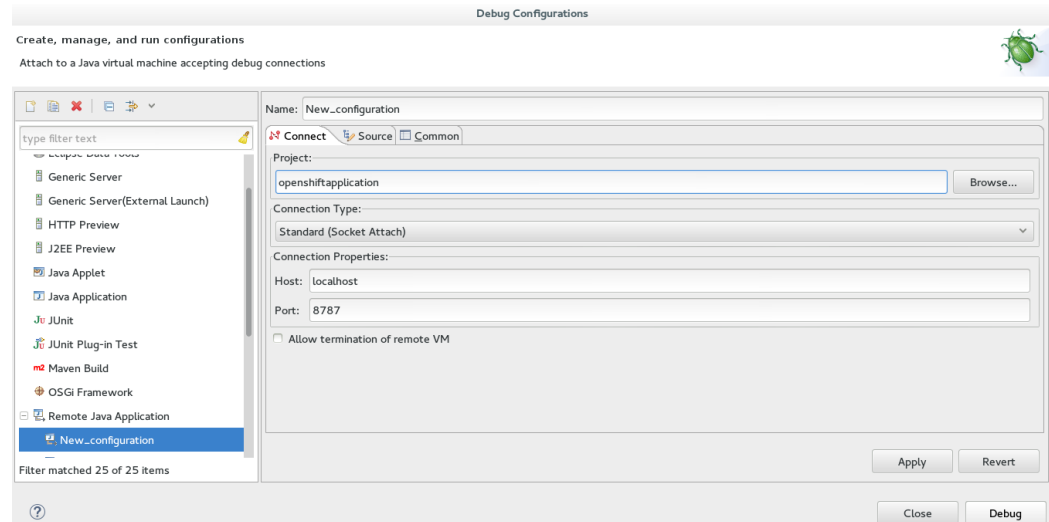
Connecting to the OpenShift server may take some time to complete and you can monitor the progress in the `Progress` bar or `Progress` view.

3.4.4. Did You Know?

» *Setting the Enable JPDA marker adds an `.openshift/markers/enable_jpda` file to your project. To locate the hidden `.openshift` directory and access the file, open the `Navigator` view.*

» *You can also access the `Port Forwarding` wizard by right-clicking `{application name} at OpenShift` in the `Servers` view, and clicking OpenShift → Port forwarding.*

» *The Debug perspective automatically arranges useful views for debugging. To open the Debug perspective, click Window → Open Perspective → Debug.*

» *You can set the debugger to look up source code for classes it encounters that are not contained in your project, for example classes used by application servers defined in the IDE. When inputing values for the launch configuration, in the `Source` tab click Add. From the list of source containers, select `JBoss Maven Source Container` and click OK. The JBoss Maven Source Container uses indexes available from Maven repositories to locate the source code. From the list, select a runtime server, the libraries of which will be indexed in the JBoss Maven Source Container, and click OK.*

3.5. DEPLOYING A WORKSPACE PROJECT TO OPENSHIFT ONLINE

OpenShift Tools enables you to deploy existing workspace projects to OpenShift Online using the `OpenShift Application` wizard. The wizard creates a templated OpenShift application based on your cartridge choices, modifies the project as necessary for use with OpenShift Online and commits the project code to the local application Git repository ready

for publishing. The wizard can also generate a server adapter for easy publishing of the application both for first deployment and for future changes.

Deploy a workspace project to OpenShift Online by:

1. *Preparing a Workspace Project for OpenShift Online Deployment*

2. *Deploying the OpenShift Online Application*

3. *Viewing the Deployed OpenShift Online Application*

3.5.1. Preparing a Workspace Project for OpenShift Online Deployment

In preparing workspace projects for OpenShift Online, the `OpenShift Application` wizard merges the existing project content with the key metadata files from a new OpenShift application and connects the project to the OpenShift application Git repository.

To prepare an existing workspace project for OpenShift Online, complete the following steps:

1. *In the `Project Explorer`, right-click `{project name}` and click Configure → New/Import OpenShift Application.*

2. *Complete the fields about your OpenShift Online account as follows:*

   a. *In the `Connection` list, select the connection.*

   b. *In the `Password` field, type your OpenShift Online account password.*

3. *Click Next.*

4. *Complete the fields about the type of OpenShift application you want to create as follows:*

   a. *Ensure `Create a new OpenShift application` is selected.*

   b. *Expand `Basic Cartridges` and select the core cartridge to match the application type; for example, for applications with Java-based server-side functionality select `jbossas-7`, `jboss-wildfly-8`, or `jbosseap-6`.*

5. *Click Next.*

6. *Complete the fields about your OpenShift application as follows:*

   a. *Ensure that the 'Domain' list displays the domain in which you want to host your application.*

   b. *Ensure the `Name` field contains a valid alphanumeric name for your application.*

   c. *In the `Gear profile` field, select the gear size that you want for your application or leave blank to use the default gear profile.*

   d. *Select the `Enable Scaling` check box to make your application scalable.*

7. **Click Next.**

8. **Complete the fields about the corresponding workspace project as follows:**

   a. **Ensure the `Create a new project` check box is clear.**

   b. **Ensure the `Use existing project` field lists the workspace project that you want to deploy on OpenShift.**

   c. **Ensure the `Create and set up a server adapter for easy publishing` check box is selected.**

   Figure 3.17. Existing Workspace Project Selected for the New OpenShift Application

9. **Click Next.**

10. **Ensure the location in the `Git Clone Destination` field exists and corresponds to where you want to make a local git repository for the project source code.**

11. **Ensure a public SSH key is uploaded to OpenShift Online and private key location specified in the IDE preferences by clicking SSH Keys wizard and reviewing the information.**

12. **Click Finish.**

13. **At the prompt informing that changes will be made to the workspace project, click OK.**

If you are prompted that the authenticity of the host cannot be established and asked whether you want to continue connecting, ensure that the host name matches that of your application and domain and click Yes.

3.5.2. Deploying the OpenShift Online Application

When the workspace project is prepared, you must republish the OpenShift Online application to push the project source code to your waiting OpenShift Online application and

trigger a build and redeployment of the application. Republishing can be achieved using the server adapter created by the `OpenShift Application` wizard and completes the deployment of the workspace project to OpenShift Online.

To deploy the OpenShift Online application, complete the following steps:

1. In the `Servers` view, right-click `{application name} at OpenShift` and click `Full Publish`.

2. When prompted if you want to publish the committed project changes to OpenShift, click Yes.

The `Console` view becomes the view in focus showing the application publication progress. After the workspace project is published at OpenShift Online, the following message displays in the `Console` view:

```
Deployment completed with status: success
```

3.5.3. Viewing the Deployed OpenShift Online Application

After the application is published, you can view the live version at the OpenShift Online application URL.

To view the OpenShift Online application in a web browser, complete the following steps:

1. In the `OpenShift Explorer` view, expand the connection and the domain.

2. Right-click `{application name}` and click Show In → Web Browser.

Your deployed OpenShift Online application is displayed in the IDE default web browser.

3.5.4. Did You Know?

» You can find the URL of your OpenShift Online application from the `OpenShift Explorer` view by right-clicking `{application name}` and clicking `Details` or from the `Servers` view by right-clicking `{application name}` on `OpenShift` and clicking OpenShift → Details.

» You can set the preference for which type of web browser the IDE opens by default. Click Window → Web Browser and click the type of web browser.

» If your project is large, it may require more time than the default set by the IDE to push changes to the application OpenShift Online Git repository. You can increase the Git timeout by clicking Windows → Preferences → JBoss Tools → OpenShift and changing the `OpenShift request timeout (in seconds)` value.

» You can also access the same wizard for deploying an existing workspace project to OpenShift Online from JBoss Central by click `OpenShift Application` under `Start from scratch`.

3.6. CONFIGURING SSH KEYS FOR OPENSHIFT

OpenShift Online uses SSH authentication for development tasks that interact directly with gears such as pushing code changes to application OpenShift Online Git repositories, reading application log files, listing and setting environment variables, and port forwarding.

You must have an SSH key pair configured for your OpenShift Online account before using OpenShift Tools features to create, edit and manage OpenShift Online applications. This requires a private-public key pair, with the private key stored on your local system and its location specified in the IDE and the associated public key stored on the OpenShift Online server and associated with your account.

OpenShift Tools enables you to quickly complete these requirements with the assistance of a multipurpose SSH wizard. If you are new to OpenShift Online, the wizards can help you configure a new or existing SSH key pair. If you are already using OpenShift Online through the web interface or RHC client tools, you only need to specify the private key location in the IDE preferences to complete the SSH key configuration.

Configure the SSH keys for OpenShift by:

  A. ***Creating and Using a New SSH Key Pair***

  B. ***Using an Existing SSH Key Pair***

  C. ***Specifying an Existing Private Key File***

3.6.1. Option A: Creating and Using a New SSH Key Pair

You can use the OpenShift Tools `Manage SSH Keys` wizard to create a new SSH key pair. In addition to creating an SSH key pair, this wizard completes the OpenShift Online SSH configuration process by automatically uploading the public key to the OpenShift Online server and setting the location of the corresponding private key in the IDE preferences. This option may be applicable if you have not used your system with OpenShift Online before.

To create and use a new SSH key pair, complete the following steps:

  1. *In the `OpenShift Explorer` view, right-click the connection and click `Manage SSH Keys`.*

  2. *In the `Manage SSH Keys` wizard, click `New`.*

  3. *Complete the fields about the new SSH keys as follows:*

      » *In the `Name` field, type a name to distinguish the key pair from any other keys associated with the connection.*

      » *From the `Key Type` list, select `SSH_RSA`.*

      » *Ensure that the `SSH2 Home` field contains the location where you want to create the files associated with the key pair. To change the location, clear the `Default` check box and type the location in the `SSH2 Home` field or click `Browse` to navigate to the desired location.*

      » *In the `Private Key File Name` field, type a name for the private key file. The `Public Key File Name` field is automatically completed and displays the private key file name appended with `.pub`.*

      Figure 3.18. Completed Fields for the New SSH Key Pair

4. **Click `Finish` and click `OK` to close the `Manage SSH Keys` wizard.**

3.6.2. Option B: Using an Existing SSH Key Pair

You can use the OpenShift Tools `Manage SSH Keys` wizard to use an already generated SSH key pair with OpenShift Online. This wizard uploads the public key to the OpenShift Online server and sets the location of the corresponding private key in the IDE preferences. This option may be applicable if you have removed a SSH key pair from your OpenShift account, still have the generated key files available on your system and want to reassociate them with your OpenShift account.

To use an existing SSH key pair, complete the following steps:

1. *In the `OpenShift Explorer` view, right-click the connection and click `Manage SSH Keys`.*

2. *Click `Add Existing`.*

3. *Complete the fields about the existing public SSH key as follows:*

   ≫ *In the `Name` field, type a name for the SSH key.*

> In the `Public Key` field, type the path of the public key file or click`Browse` to locate the public key file.

Figure 3.19. Completed Fields for Adding an Existing SSH Public Key

4. *Click the `SSH2 Preferences` link.*

5. *In the `Private keys` field, click `Add Private Key` and locate the private key file.*

6. *Click `Apply` and click `OK` to close the`Preferences` window.*

7. *Click `Finish` and click `OK`.*

3.6.3. Option C: Specifying an Existing Private Key File

When using a public key that is already uploaded to OpenShift Online, you must ensure the location of the corresponding private key is set in the IDE preferences. This option may be applicable if you are an existing OpenShift Online user and you have not yet accessed OpenShift Online from your IDE workspace or you are using multiple IDE workspaces.

To specify an existing private key file, complete the following steps:

1. *Click Window → Preferences → General → Network Connections → SSH2.*

2. *In the `General` tab, click `Add Private Key` and locate the private key file.*

3. *Click `Apply` and click `OK` to close the`Preferences` window.*

3.6.4. Did You Know?

» *To quickly open the `OpenShift Explorer` view, in the `IDE Quick Access` field type `OpenShift Explorer` and choose `OpenShift Explorer` view from the list of items.*

» *For added security, you can add a passphrase to the private SSH keys associated with your OpenShift account when you create them. You must enter the passphrase for the specific key for each OpenShift Tools action that uses the SSH key for authentication with OpenShift Online.*

» *IDE preferences are workspace specific and therefore you must specify private key location under `Preferences` for each workspace.*

» *You can remove an existing public SSH key from your OpenShift Online account with the `Manage SSH Keys` wizard. Note that this only disassociates keys with your OpenShift account. The files associated with a removed SSH key pair still exist in the local location where they were generated and the private key is still set in the IDE preferences.*

» *If you are using the OpenShift Online web interface or RHC client tools to manage SSH keys simultaneously with OpenShift Tools, the information specified by OpenShift Tools about keys may be out of sync. You can resync the information in the `Manage SSH Keys` wizard with the `Refresh` button.*

» *You can change the default location used by the IDE to store newly created SSH keys. Click Window → Preferences → General → Network Connections → SSH2. In the `General` tab, in the `SSH2` home field type the new location.*

3.6.5. Troubleshoot SSH Key Issues

3.6.5.1. "You have no SSH public keys in your OpenShift account"

Error Message

You have no SSH public keys in your OpenShift account `user@example.com` yet, please add your key(s) or create new one(s).

Figure 3.20. No SSH Keys Window Displaying the Error Message

Issue

There are no SSH public keys assigned to your OpenShift account. After you have connected to OpenShift Online from the IDE, the `No SSH Keys` window appears informing you that there are no SSH public keys associated with your account.

Resolution

In the `No SSH Keys` window, click `Add Existing` or `New` and add an SSH public key to your OpenShift Online account as detailed in Option A, Create and Use a New SSH Key Pair and Option B, Use an Existing SSH Key Pair above.

3.6.5.2. "Please make sure that you added your private key to the ssh preferences."

Error Message

```
Could not clone the repository. Authentication failed. Please make sure
that you added your private key to the ssh preferences.
```

Figure 3.21. Problem Occured Window Displaying the Error Message
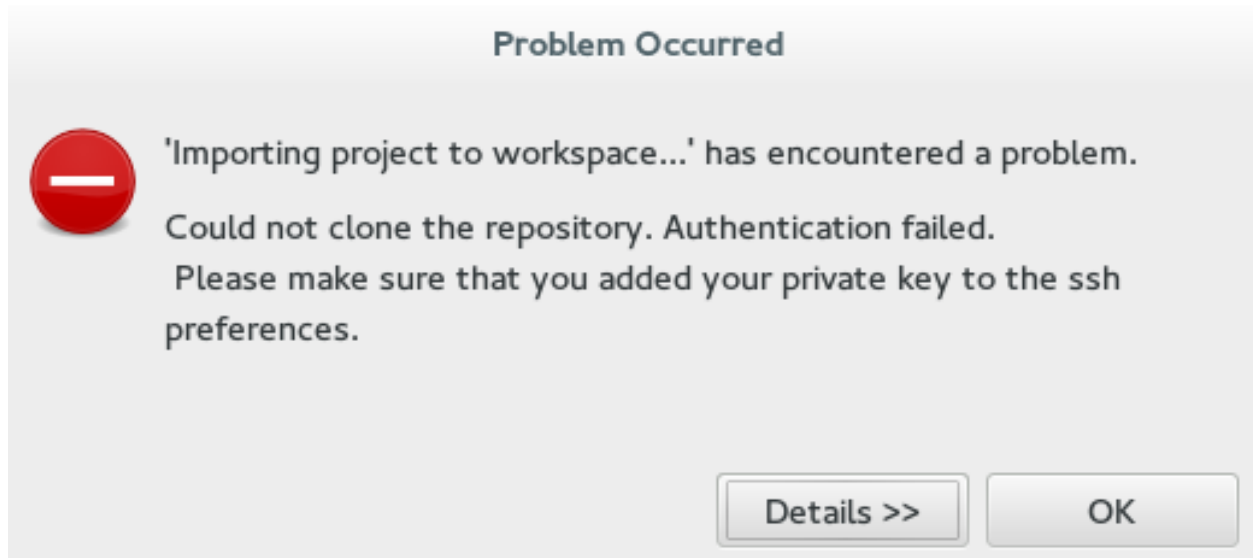
Issue

An SSH public key is associated with your OpenShift Online account but the IDE cannot locate the companion private key. Consequently, the IDE cannot complete actions for the OpenShift applications associated with your account. This error may be encountered when creating new OpenShift applications or importing existing OpenShift applications into the IDE.

Resolution

Click `OK` to close the `Problem Occurred` window. The `OpenShift Application` wizard opens. Click the `SSH2 Preferences` link to add the private key as detailed in Option C, *Specify an Existing Private Key Location* **above.**

CHAPTER 4. DEVELOPING WITH DOCKER

4.1. CONFIGURING DOCKER TOOLING BASICS

Docker offers images that can be managed in one of two ways: build/create your own image using a script file, or pull down an existing image file from public or private registries online. This procedure contains instructions for pulling down an image from the JBoss registry. Such registries are useful to share images between developers or between environments to ensure a standardized software stack for development or testing requirements. Once the Docker Container is created and running, users can manage the container.

JBoss Tools 4.3.0 Beta1 includes the Docker tooling out of the box. This article introduces the basic usage for the Docker tooling, such as:

1. *Prerequisites for Docker Tooling*

2. *Pulling Docker Images from a Docker Registry*

3. *Managing Docker Containers*

4.1.1. Prerequisites for Docker Tooling

Ensure that the following prerequisites are met when using the Docker tooling with the IDE:

1. *Install and set up JBoss Tools 4.3.0 Beta1 or higher.*

2. *Install and set up Docker.*

3. *Establish a connection to a Docker daemon within JBoss Tools as follows:*

   a. *Click Window → Show View → Other….*

   b. *In the `View` window, type docker in the filter text box to view Docker-related options in the list.*

   c. *Expand the `Docker` item in the results and select `Docker Explorer`.*

   d. *In the `Docker Explorer` view, if no connection is configured, a message appears stating that "No connection to a Docker daemon is available. Click this link to create a new connection…". Click the link to start configuring a new Docker daemon connection.*

   Figure 4.1. No Connection to a Docker Daemon is Available
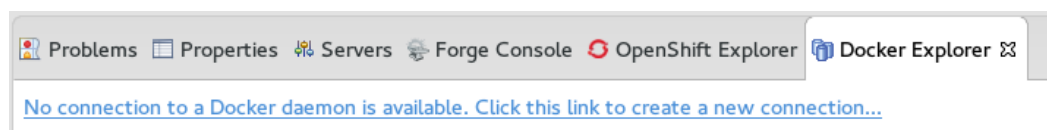
   e. *In the `Connect to a Docker daemon` wizard:*

      i. *Use the default value or set a desired connection name in the `Connection Name` field.*

ii. *Check the `Use custom connection settings` field.*

iii. *Add the relevant socket information for your host. If you are unsure about this step, use the default Unix socket Location value.*

iv. *Click `Test Connection` to test the connection.*

v. *When a connection is successfully established, the following message appears:*

Figure 4.2. Ping Success Message
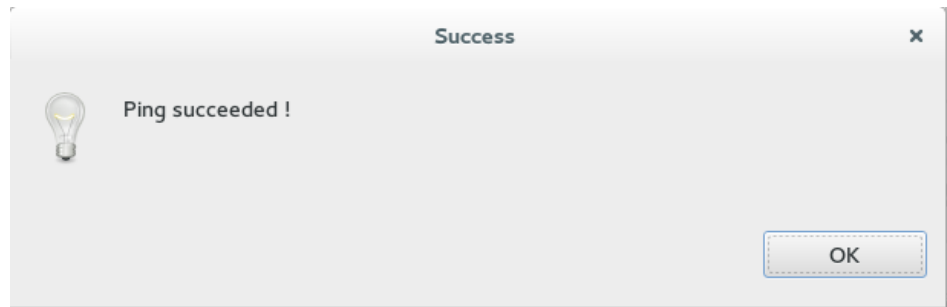
vi. *This message indicates that the Docker daemon connection is successfully established.*

4.1.2. Pulling Docker Images from a Docker Registry

The following instructions detail how to pull a new WildFly Docker Image from the JBoss registry.

1. *Click Window → Show View → Other….*

2. *Type Docker in the filter text field to view Docker-related options in the list.*

3. *Expand the `Docker` entry and double click `Docker Images`.*

4. *In the listed entries for the `Docker Images` view, locate the entry that ends with wildfly:latest. If this entry is not listed, pull the Wildfly image as follows:*

   a. *Click the Pull Image icon at the top bar of the `Docker Images` view.*

   b. *In the resulting wizard, type jboss/wildfly and click finish.*

   Figure 4.3. Pull WildFly Image from JBoss Registry
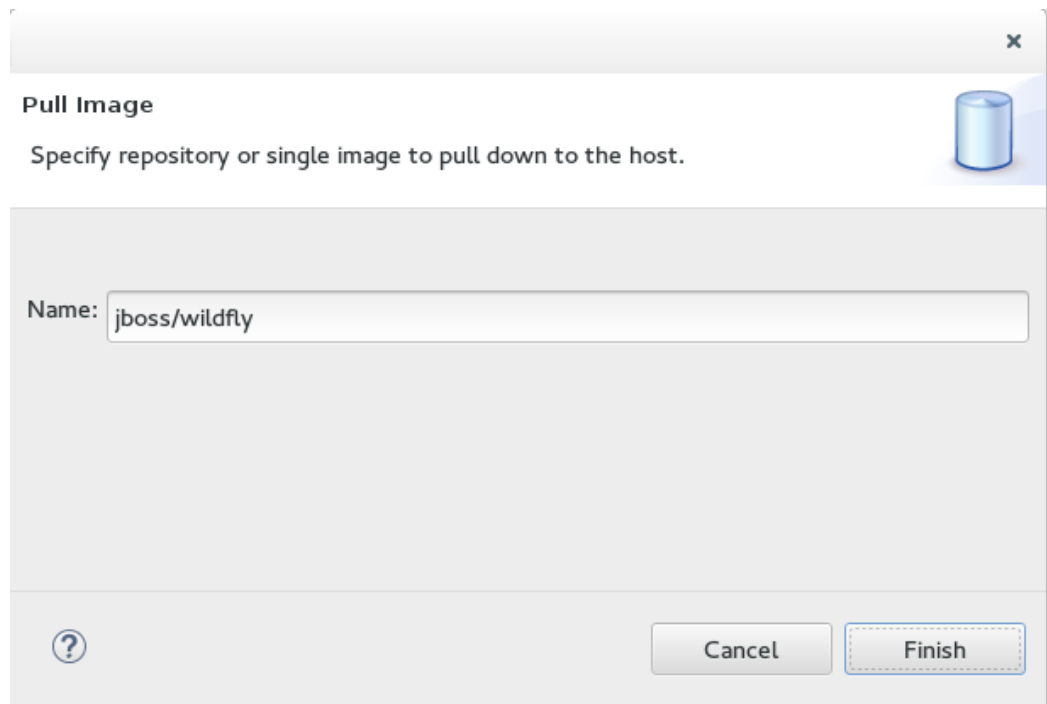
c. **The bottom right corner displays the progress as the image is pulled down from the registry. When the pulling process completes, the appropriate entry appears on the Docker Images list as follows:**

Figure 4.4. The Docker WildFly Image

Result: You have now pulled a new image for Wildfly from the JBoss registry and run the Docker Image.

4.1.3. Managing Docker Containers

Docker containers are isolated processes that are based on Docker Images. Once created, users can stop, start, pause, unpause, kill, or remove the containers, or read their logs.

1. **Click Window → Show View → Other….**

2. **Type Docker in the filter text field to view Docker-related options in the list.**

3. **In the search results, expand the Docker entry and double-click `Docker Containers`.**

4. **The `Docker Containers` view appears, displaying a list of all containers running on the Docker host.**

5. **Click the desired container to select it. You can now manage your containers using the following operations:**

    a. *Click the pause button from the icons on the top right corner of the view to pause the container.*

    b. *Click the green triangle icon in the row to unpause the container.*

    c. *To view the container logs, right click the container name and click `Display Log`.*

    d. *To view a list of all containers, click on the right-most icon in the list of icons in the view, which displays a drop-down option to view all containers. Click this option to view all available containers.*

Result: You have performed various management operations on your Docker container.

4.1.4. Troubleshooting

Attempting to connect to a running local Docker instance as a non-root user results in errors being logged, but not displayed in the User Interface, which results in the error being non-obvious. The following workarounds are available for this problem:

- *Connect to the Docker instance manually. Define a custom configuration file and specify the TCP URL displayed by the systemctl status docker service. As an example, you can use a TCP URL such as `tcp://0.0.0.0:2375` to connect to the running Docker instance instead of the default `unix:///var/run/docker.sock` configuration file.*

Figure 4.5. Connect to a Docker Daemon
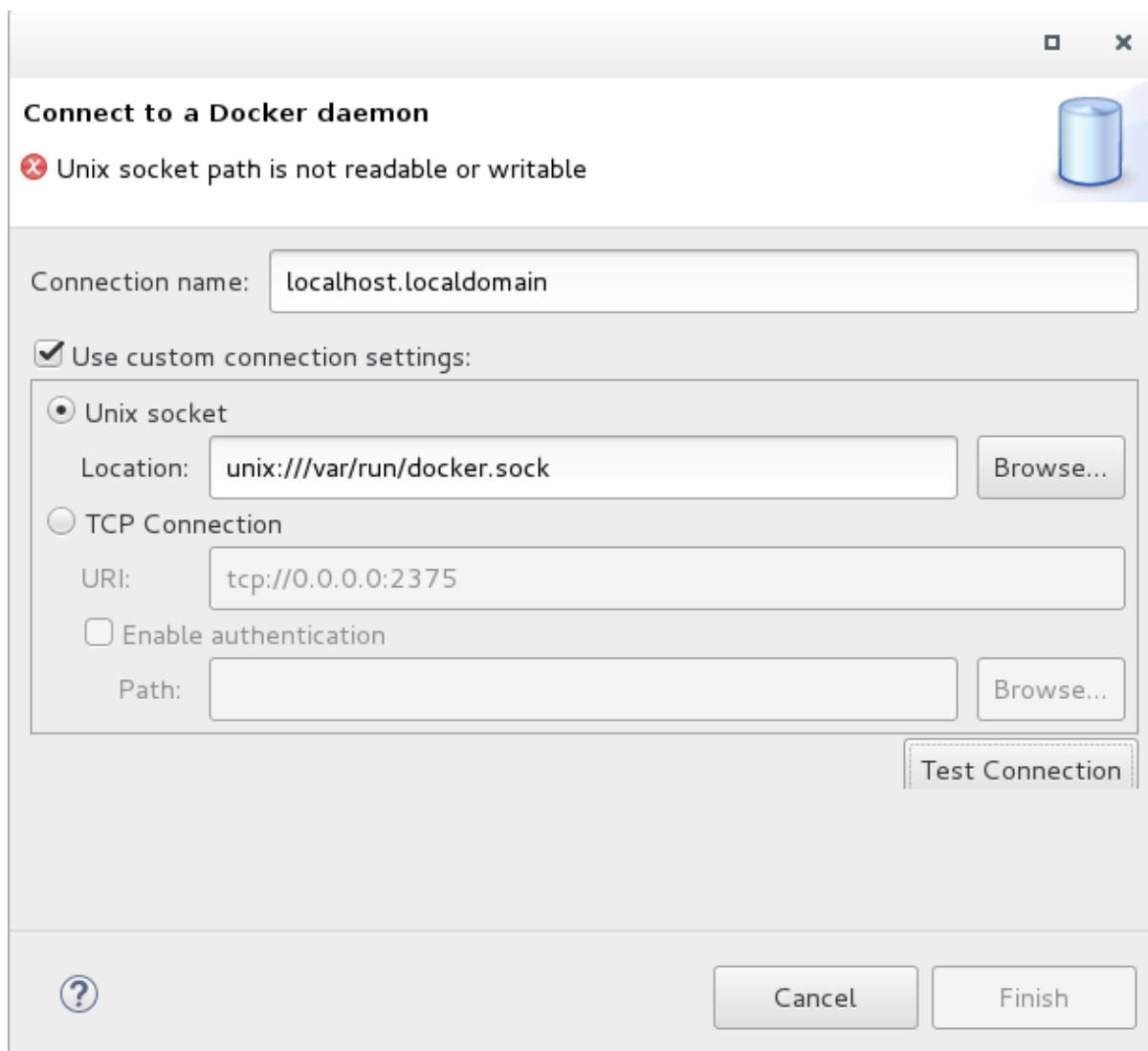
⟫ ***Run Eclipse as root. This solution avoids the problem but is not the recommended solution.***