

Table of Contents

Lab Setup

1. Install Required Software on Your Machine
 2. Install the Exercises
 3. Configure Your Local Maven Environment
 4. Compile the Lab Assets
 5. JBoss Developer Studio
 6. Import the Project in JBoss Developer Studio
 7. Create a JBoss Fuse Application in OpenShift Enterprise
 8. Access the JBoss Fuse Management Console
 - 8.1. Access Application Details
 - 8.2. Create a New JBoss Fuse Instance
 - 8.3. Deploy the Camel Twitter Application
 9. SSH into Your Online Lab Account
 10. Tail Your Application Log Files
-

Lab Setup

Goals:

- Verify prerequisites
- Configure your local Maven environment
- Compile the lab assets project
- Download and install Red Hat® JBoss® Developer Studio and the Integration Stack plugin
- Import the project in JBoss Developer Studio
- Create your Red Hat® JBoss® Fuse app using the OpenShift Enterprise by Red Hat® gear
- Access the JBoss Fuse Management Console
- Use SSH to connect to your online lab account
- Tail your application log files

Prerequisites:

- Access to the Internet
- Access to your course confirmation email
- Experience with Java, Apache Maven, and SSH

1. Install Required Software on Your Machine

Software	Version & URL	Notes	
Java SE	1.7 or higher	Required	
Apache Maven	3.0.5 or higher	Required	
JBoss Developer Studio	8.1.0.GA	Required. Account on the jboss.org web site is needed.	
Integration Stack	8.0.2	Required. Account on the jboss.org web site is needed.	
SoapUI	latest	Optional	

2. Install the Exercises

The lab exercises and their solutions are available in the following zip archives:

- <https://github.com/gpe-mw-training/camel-labs/archive/v0.3-exercise.zip>.
- <https://github.com/gpe-mw-training/camel-labs/archive/v0.3-solution.zip>.

If you have not already done so, install the exercises by unzipping this archive to a location on your file system. You will then find all the Camel exercises stored in the

camel-labs-VERSION-NUMBER-exercise and
camel-labs-VERSION-NUMBER-solution folders.



On Windows machines, it is recommended that you designate either **C:** or **C:\dev** as the destination folder when unzipping the files to provide easy file access.

3. Configure Your Local Maven Environment

All the labs for this course involve using Apache Maven modules with dependencies on Maven libraries supported in JBoss Fuse. Red Hat provides both online and offline Maven repositories for JBoss Fuse. Follow these steps to configure these online repositories on your machine:

1. To get started, edit the Apache Maven `settings.xml` file located under the directory `home_directory/.m2` or `~/.m2`.



If this file does not exist under this location, you can also edit the same file which is available under the `conf` subdirectory of the Apache Maven installation directory.

2. Replace the contents of the `settings.xml` with the contents of `camel-labs-VERSION-NUMBER-exercise/feature-exercises/settings.xml`
3. Within the `<profile>` tags in the `settings.xml` are defined the **JBoss Fuse** repositories that you need to build the code during execution of `mvn compile` or `mvn install` commands.
4. Within the `<activeProfile>` tags in the `settings.xml` is the name of the default profile used when Apache Maven is running.

Final settings.xml file

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <profiles>
    <!-- Profile with online repositories required by Fuse -->
    <profile>
      <id>fuse-online-repos</id>
      <repositories>
        <repository>
          <id>jboss-ga-repository</id>
          <url>http://maven.repository.redhat.com/techpreview/all</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
        <repository>
          <id>jboss-public-repository</id>
          <url>http://repository.jboss.org/nexus/content/repositories/public</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
```

```

<pluginRepository>
  <id>jboss-ga-plugin-repository</id>
  <url>http://maven.repository.redhat.com/techpreview/all</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</pluginRepository>
<pluginRepository>
  <id>jboss-public-plugin-repository</id>
  <url>http://repository.jboss.org/nexus/content/repositories/public/</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</pluginRepository>
</pluginRepositories>
</profile>
</profiles>

<activeProfiles>
  <!-- Activation of the Fuse profile -->
  <activeProfile>fuse-online-repos</activeProfile>
</activeProfiles>

</settings>

```

4. Compile the Lab Assets

To verify that Apache Maven is working correctly on your machine, you are going to compile the code for the Apache Camel labs.

1. Open a Linux/Unix/MS-DOS terminal and change to **camel-labs-VERSION-NUMBER-solution** directory.
2. Execute the **mvn compile** command to compile the code.
3. If the compilation process succeeds, you will see the following *BUILD SUCCESS* message (at the end of a long list of trace results) on the console. If you see an error message instead, respond to the error reported.

```

[INFO] Copying 1 resource
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] RedHat GPE Training :: Camel Project Exercise ..... SUCCESS [0.002s]

```

```
[INFO] RedHat GPE Training :: Camel :: Exercise :: Standalone SUCCESS [0.948s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Spring Standalone SUCCESS
[0.123s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Expressions SUCCESS [0.121s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Unit test SUCCESS [0.016s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: DataFormat SUCCESS [0.103s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Exceptions SUCCESS [0.083s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: CBR ... SUCCESS [2.523s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Web ... SUCCESS [0.126s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: WebService SUCCESS [17.661s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Aggregator SUCCESS [0.107s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Splitter SUCCESS [0.096s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: JMS Transactional Route SUCCESS
[1.202s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: JDBC - JMS Transaction SUCCESS
[0.214s]
[INFO] RedHat GPE Training :: Camel :: Exercise :: Features file SUCCESS [0.404s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 24.551s
[INFO] Finished at: Thu Oct 09 15:06:50 CEST 2014
[INFO] Final Memory: 55M/442M
[INFO] -----
```

5. JBoss Developer Studio

JBoss Developer Studio is an integrated development environment (IDE) that combines both tooling and runtime components including Eclipse plug-ins, best-of-breed open source tools, and the Red Hat® JBoss® Enterprise Application Platform (JBoss EAP).

For all the lab exercises, you must have JBoss Developer Studio installed in your local development environment. You will use JBoss Developer Studio to design Apache Camel Routes and to manage your remote lab applications which are enabled by OpenShift Enterprise (via SSH and the OpenShift Explorer view).

You can download JBoss Developer Studio from the jboss.org web site and the install guide is available from the [RedHat Customer Portal](#).

JBoss Developer Studio includes a variety of Eclipse plug-ins. The following is a list of the JBoss Developer Studio plug-ins that you need to complete the labs in the middleware courses sponsored by Red Hat OPEN:

Integration Stack Suite

The Integration Stack Suite of plug-ins is of particular importance for use with JBoss Fuse and Red Hat® JBoss® A-MQ middleware. The Integration Stack is not included out-of-the-box with JBoss Developer Studio. You must manually install the suite by following the instructions

in the [IntegrationStack documentation](#).

OpenShift Enterprise Plug-in

JBoss Developer Studio includes an out-of-the-box plug-in for management of OpenShift Enterprise environments. No additional installation is required to use this plug-in. With this OpenShift Enterprise plug-in, you can fully manage your remote OpenShift Enterprise environment (ie; upload SSH keys, manage domains and apps, etc.).

Remote System Explorer Plug-in

JBoss Developer Studio includes an out-of-the-box plug-in for creating SSH / SCP connections to remote SSH-enabled servers. No additional installation is required to use this plug-in.

Eclipse eGit Plug-in

JBoss Developer Studio includes the eGit plug-in, which provides Git project support. No additional installation is required to use this plug-in. Git is an open source version control system, providing developers with fast, versatile access to the entire revision history of the application code that they are working on.

Eclipse m2e Plug-in

JBoss Developer Studio includes the m2e plug-in, which provides support for Apache Maven projects. No additional installation is required to use this plug-in. The m2e plug-in enables you to edit a Maven project's `pom.xml` file and to run a Maven build from the IDE.

6. Import the Project in JBoss Developer Studio

Now that JBoss Developer Studio is installed, you are ready to import the `camel-labs-VERSION-NUMBER-exercise` Apache Maven project as follows:

1. Open **JBoss Developer Studio**.
2. Import the `camel-labs-VERSION-NUMBER-exercise` Apache Maven project into a new workspace:
 - a. Select **File** → **Import** from the menu.
 - b. Select **Maven** → **Existing Maven Projects**.

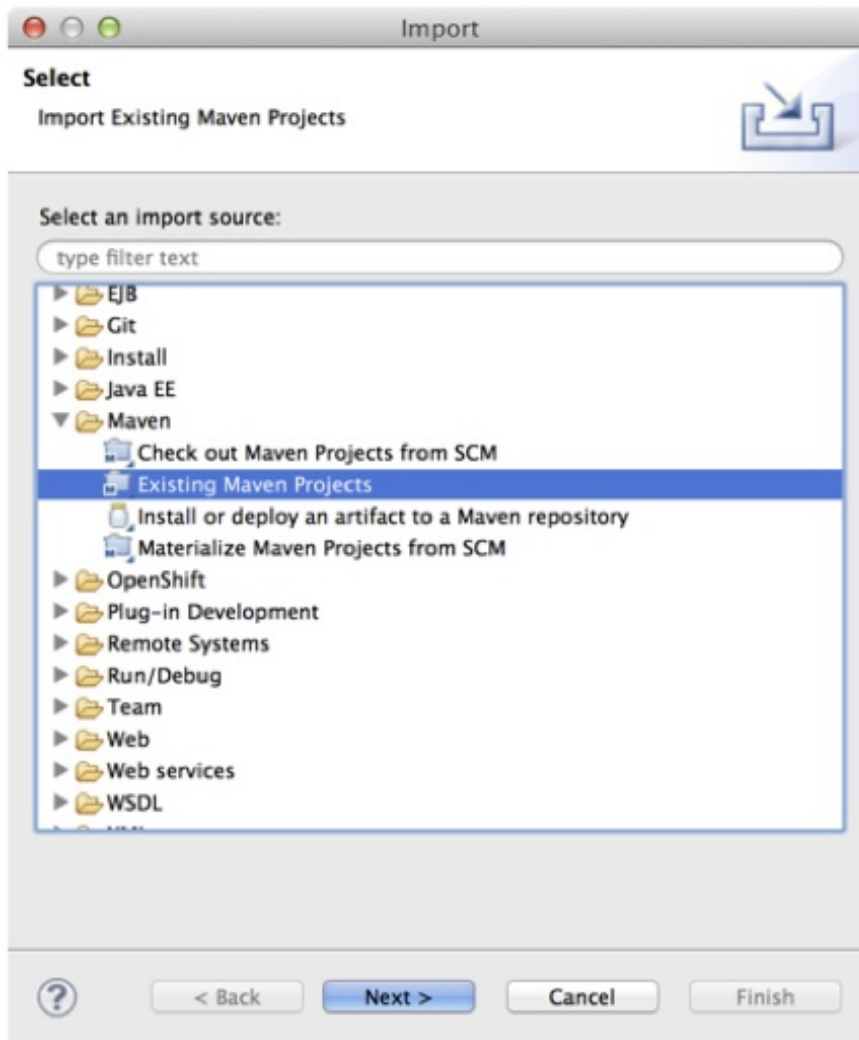


Figure 1. Import lab - part 1

- c. Click **Next**.
- d. Click **Browse** and enter `$FUSE_LAB_HOME/lab2`.
- e. Make sure the **pom.xml** box is checked for: `com.redhat.gpe.training.camel`.

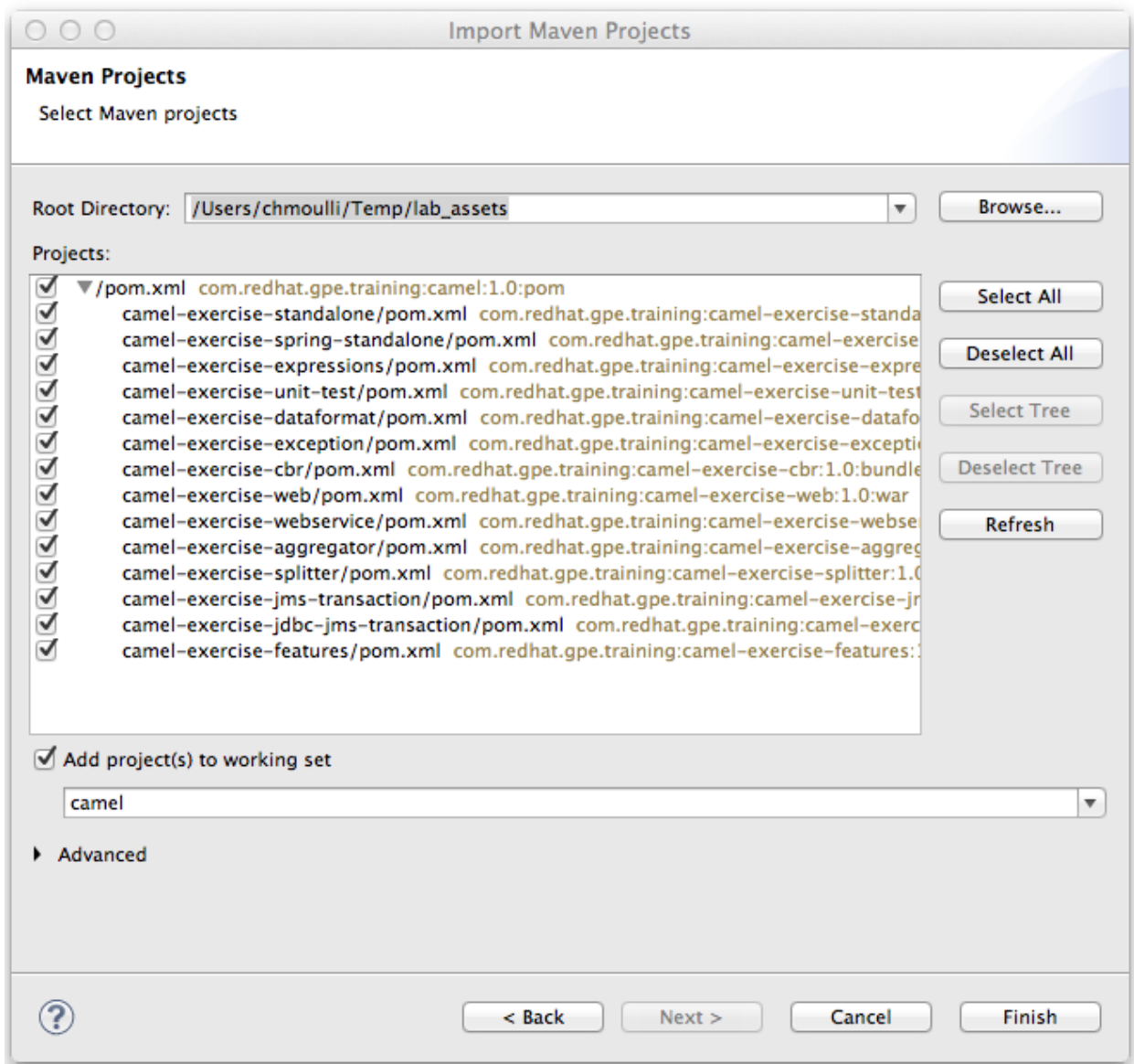


Figure 2. Import lab - part 2

- f. Click **Finish**.
3. Confirm that the project imported correctly and that you can compile it using Maven by selecting **Run as** → **maven install** from the **pom.xml** file of the parent Camel project.

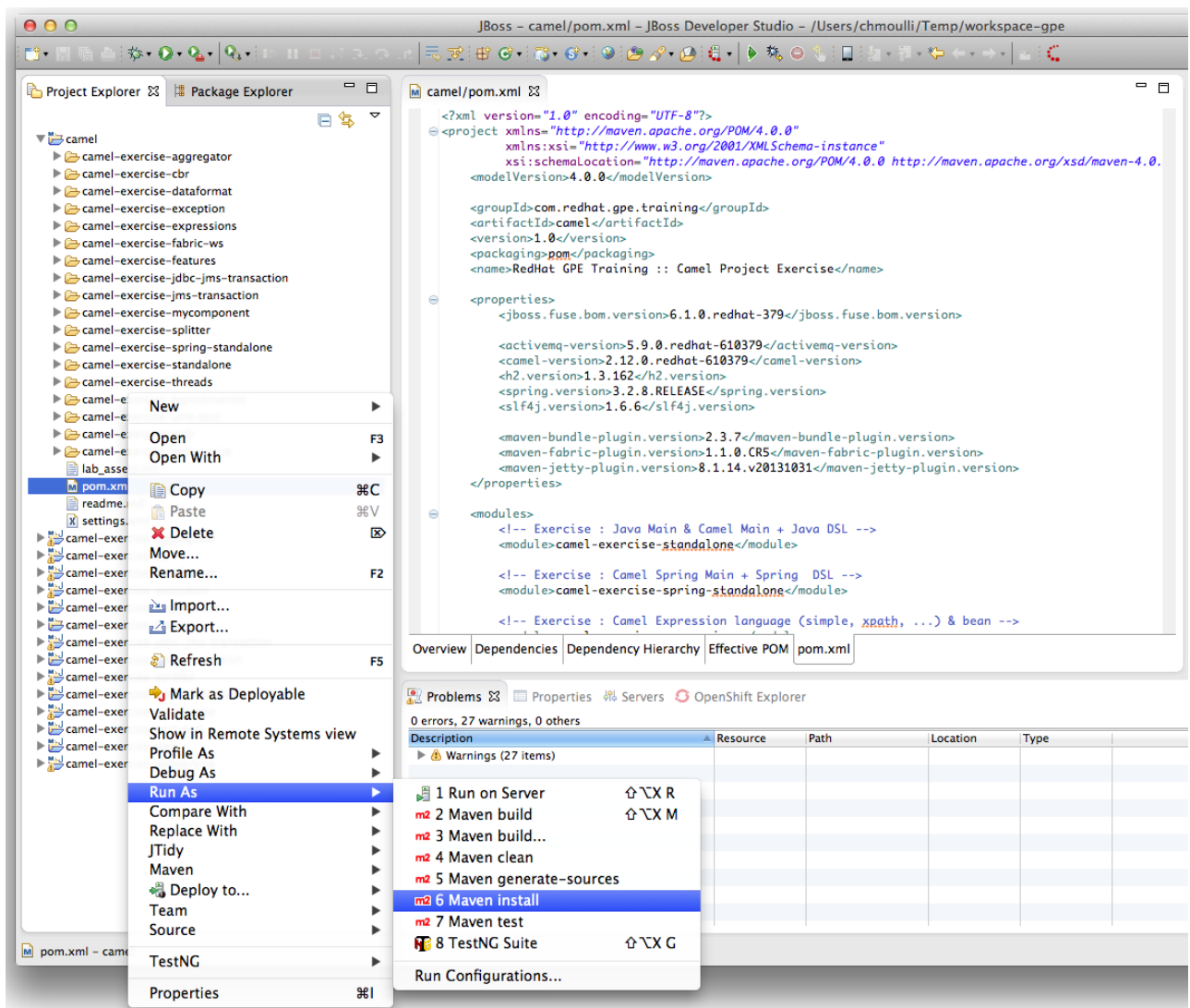


Figure 3. Run As

4. If the process succeeds, you will see the *BUILD SUCCESS* message in the console.

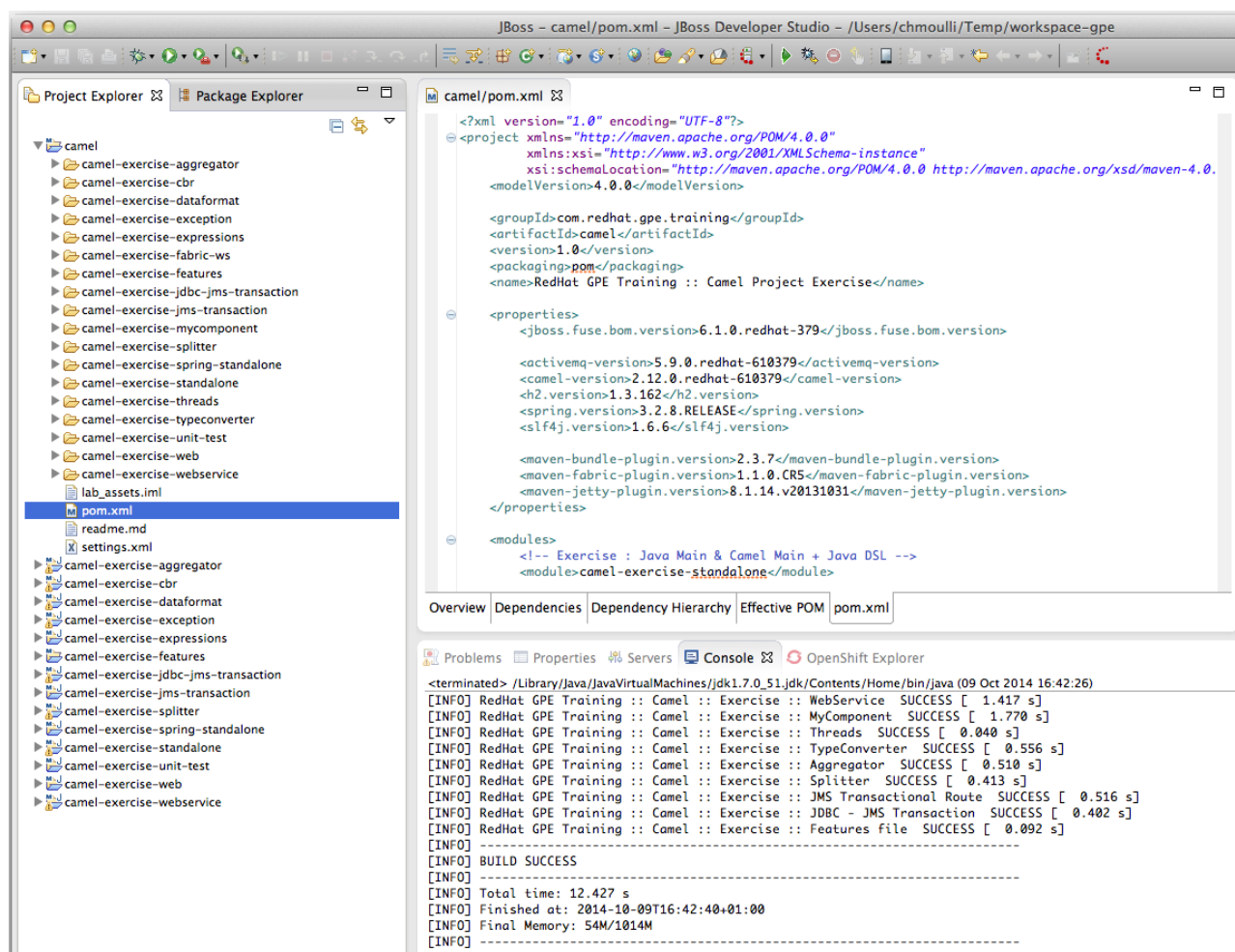


Figure 4. Compilation result

7. Create a JBoss Fuse Application in OpenShift Enterprise

OpenShift Enterprise is a private Platform-as-a-Service (PaaS) that provides developers and IT organizations with an auto-scaling cloud application platform for deploying new applications on secure scalable resources with minimal configuration and management overhead. It supports a wide selection of programming languages and frameworks, such as Java, Ruby, and PHP. JBoss Developer Studio completes the application life cycle by providing tooling support.

You will use OpenShift Enterprise as a platform for hosting your JBoss Fuse Application during this training. The procedure is described in the GPE Common labs, and you will use OpenShift Explorer in JBoss Developer Studio to reach this goal.

1. Follow the steps defined [here](#) until you reach the **Cartridges and Gear Type** selection step.
2. Instead of selecting the JBoss EAP 6 cartridge, select the **JBoss Fuse 6.1.0 EA** cartridge as shown here:

New OpenShift Application

New or existing OpenShift Application

Create a new OpenShift Application.

Domain: fuse [Manage Domains](#)

Existing Application

☐ Use existing application:

New application

Name: demo

Type: JBoss Fuse 6.1.0 EA (fuse-1.0.0)

Gear profile: fuse_medium ☐ Enable scaling

Embeddable Cartridges

<input type="checkbox"/>	Cron 1.4 (cron-1.4)
<input type="checkbox"/>	Jenkins Client (jenkins-client-1)
<input type="checkbox"/>	MongoDB 2.4 (mongodb-2.4)
<input type="checkbox"/>	MySQL 5.1 (mysql-5.1)
<input type="checkbox"/>	MySQL 5.5 (mysql-5.5)
<input type="checkbox"/>	PostgreSQL 8.4 (postgresql-8.4)
<input type="checkbox"/>	PostgreSQL 9.2 (postgresql-9.2)
<input type="checkbox"/>	Web Load Balancer (haproxy-1.4)

Figure 5. JBoss Fuse cartridge

3. Save the passwords generated for the JBoss Fuse application; you will need this password to access the web-based JBoss Fuse Management Console.



This password appears only when the gear is created and you cannot retrieve it unless you connect to OpenShift Enterprise using SSH protocol.



Figure 6. Passwords generated

You should now have a **demo** application that was provisioned using a JBoss Fuse cartridge on OpenShift Enterprise.

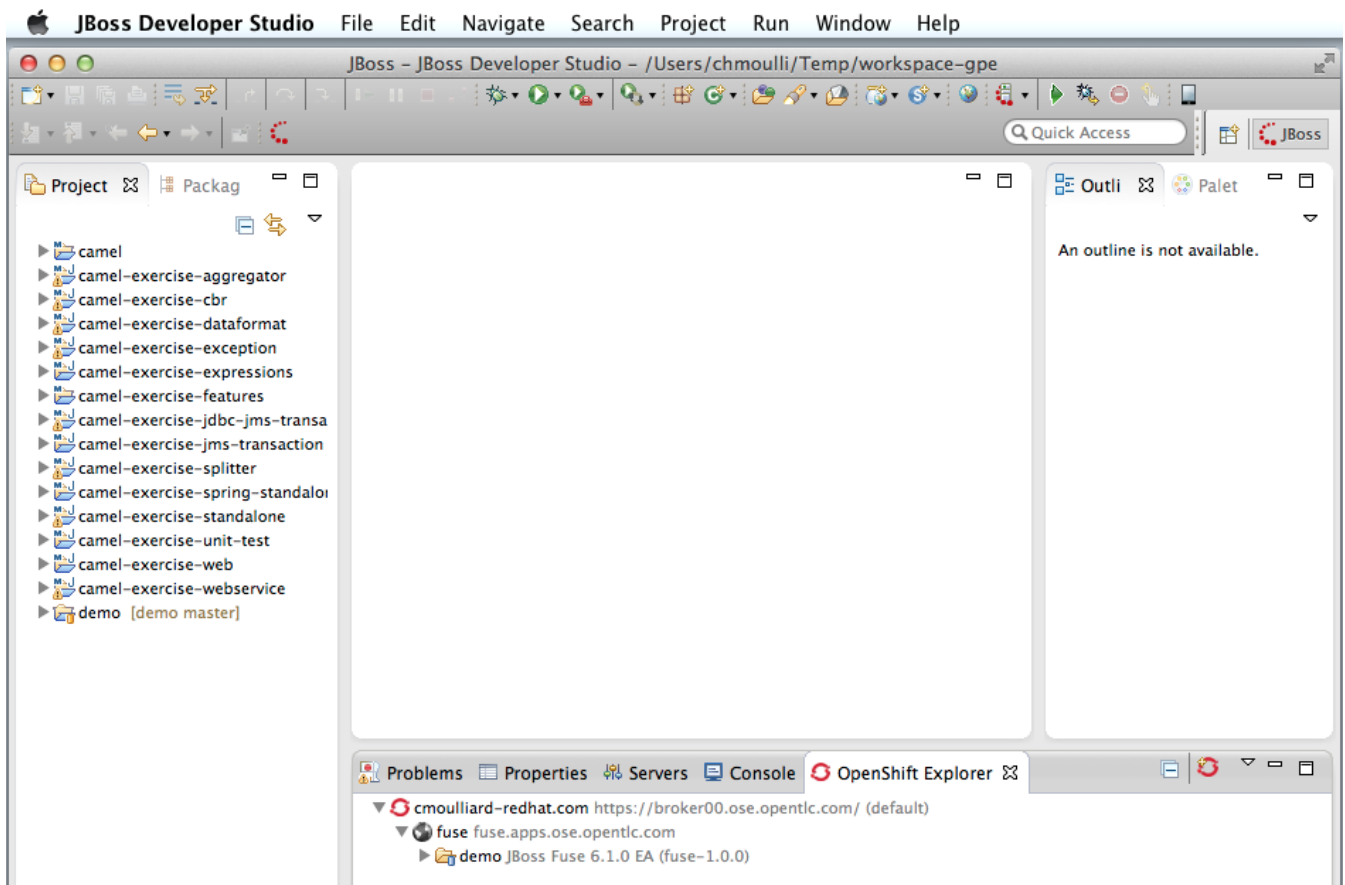


Figure 7. OpenShift Explorer

8. Access the JBoss Fuse Management Console

Next you will use a browser to access the JBoss Fuse Management Console, create a new container, and then install a Camel Twitter project in your new container.

8.1. Access Application Details

1. Expand the **OpenShift connection** (<https://broker00.ose.opentlc.com>) to display the domain (**fuse.apps.ose.opentlc.com**) and the **demo** application you created.
2. Right-click the **demo** application icon and select **Details**.

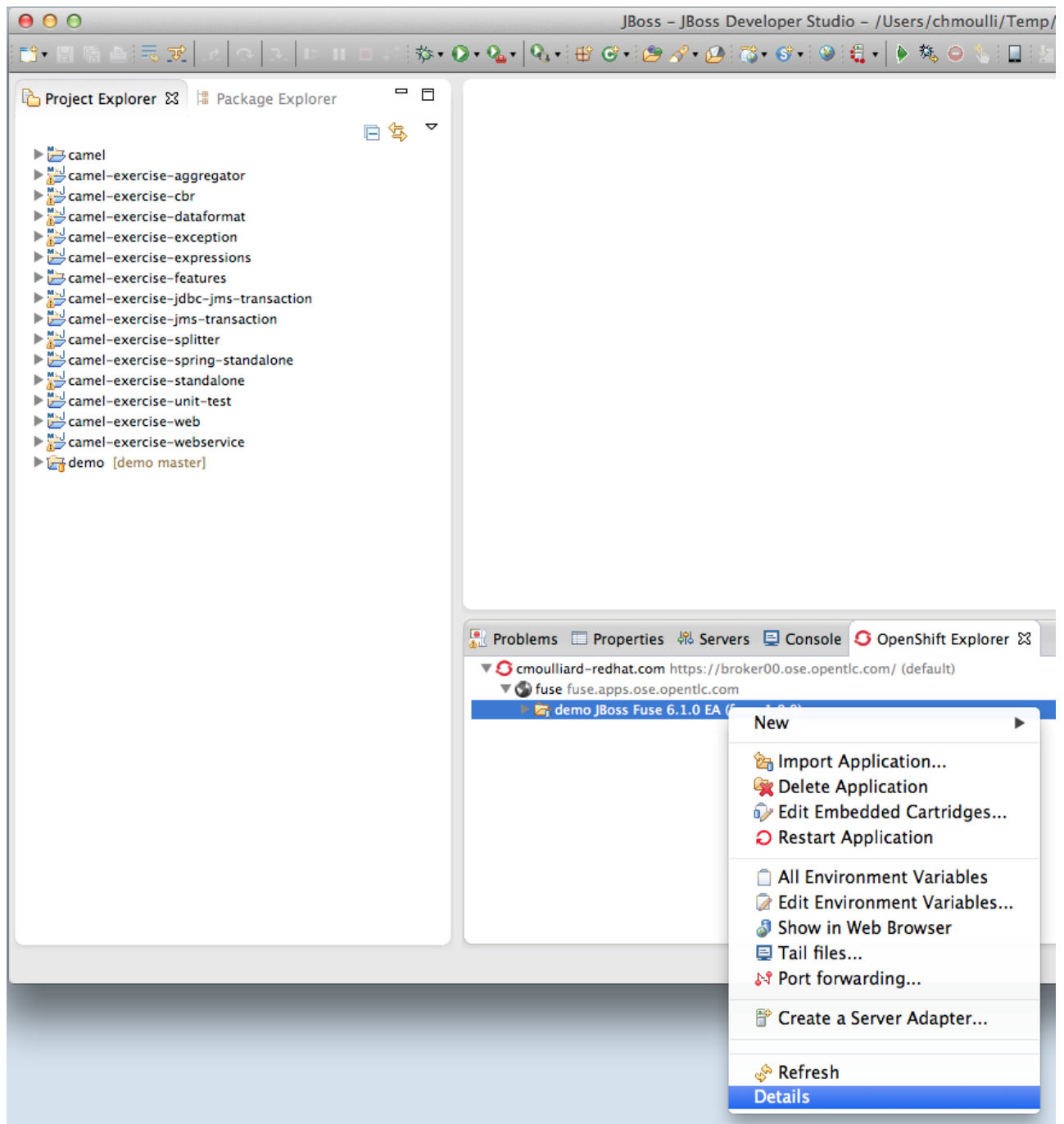


Figure 8. OpenShift application name

3. Copy the console URL and paste it in your browser.

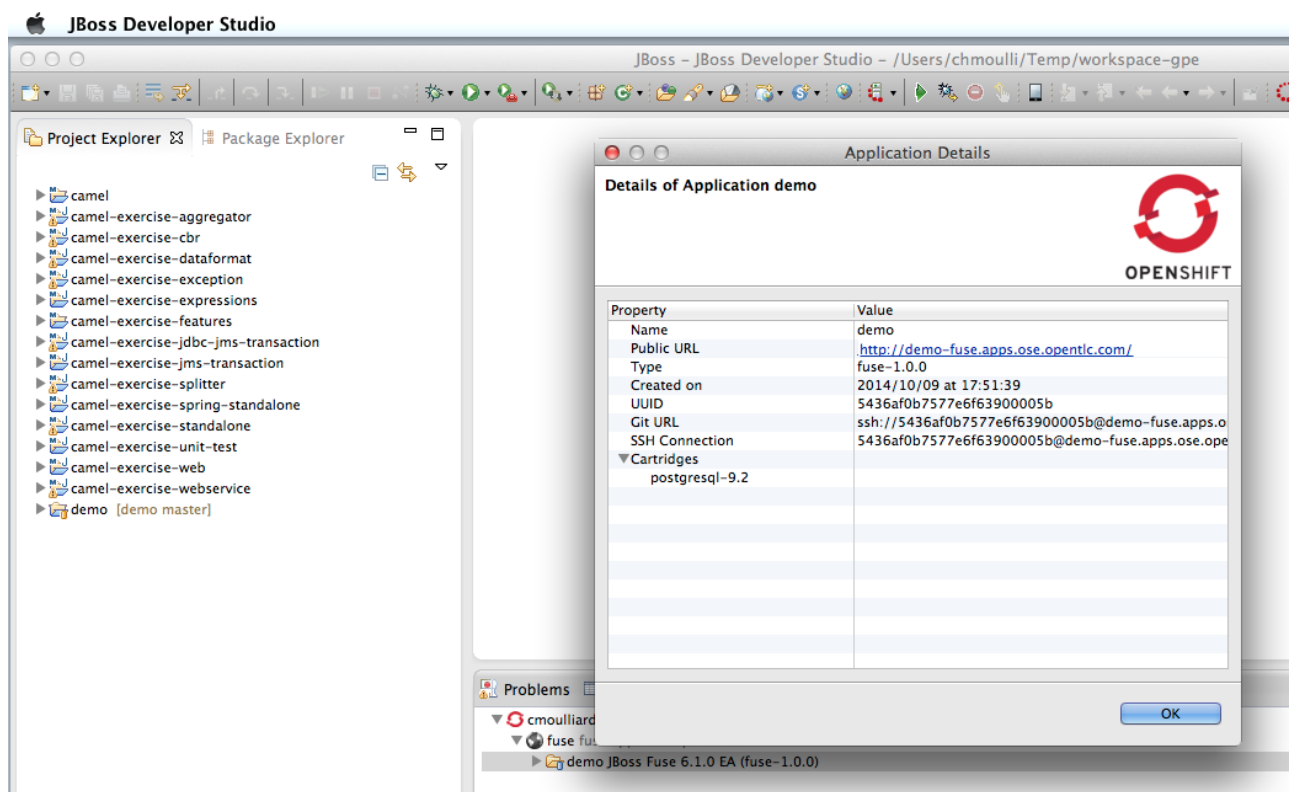


Figure 9. Name of the JBoss Fuse Management Console

4. Provide your **username** (admin) and **password** that you captured while creating the OpenShift application (see Figure 6. Passwords generated)

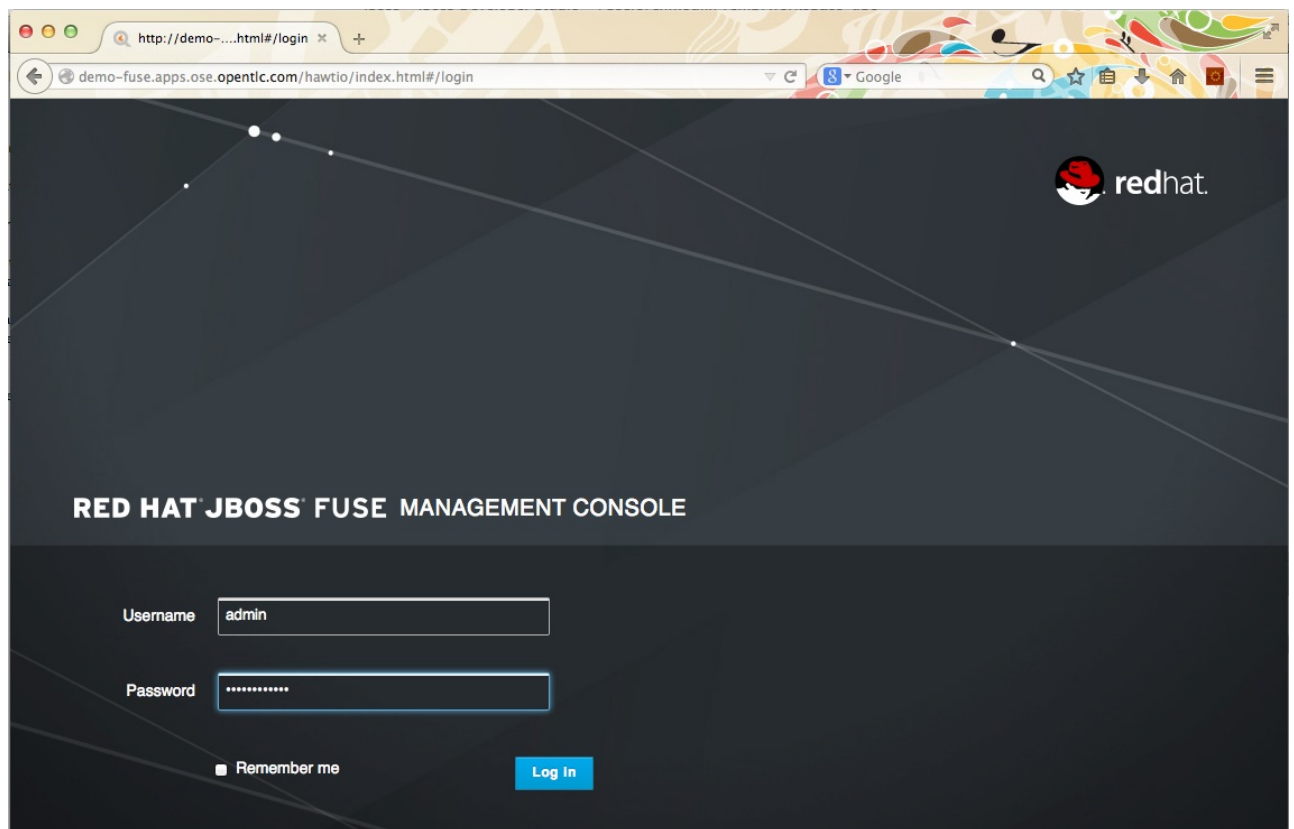


Figure 10. Login screen

After you connect to the **JBoss Fuse Management Console**, you are redirected to the menu, which contains options for navigating between the containers created, the application performance dashboard, the Fabric8 health monitor, and the Fabric8 wiki.

The console includes documentation about the various menu options. To access the documentation, click the **Help** icon () in the main navigation bar.

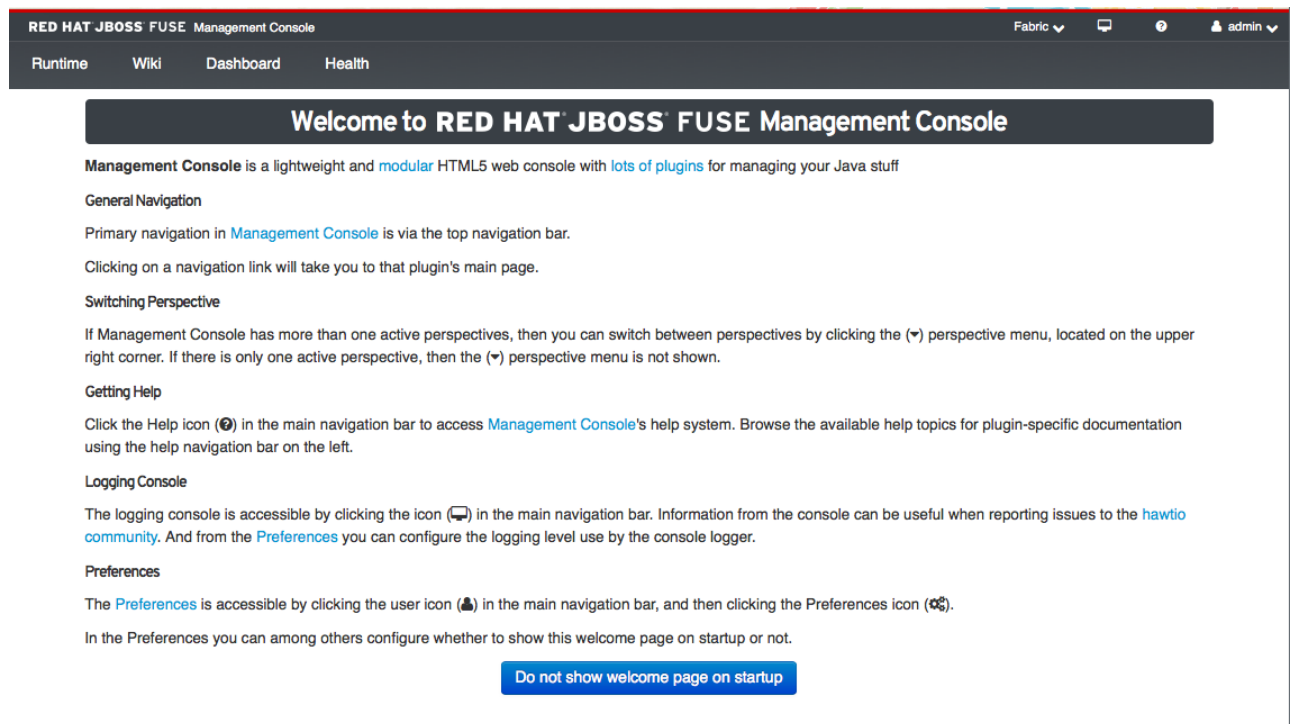


Figure 11. JBoss Fuse Management Console Menu

Next, to familiarize you with **JBoss Fuse Middleware** technology, you will now do the following:

- Create a new JBoss Fuse instance (which is a Java OSGi Container)
- Deploy a simple Camel Twitter project into this newly created container

8.2. Create a New JBoss Fuse Instance

1. From the JBoss Fuse Management Console, click the **Runtime** menu.
2. Click **Create** (on the right side of the screen).

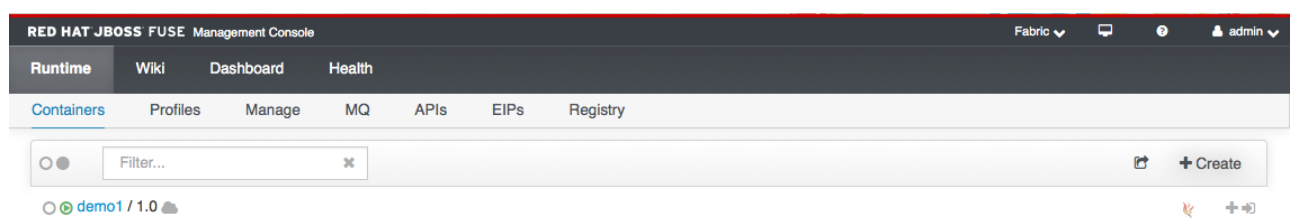


Figure 12. Create a container

Before you create the new container, you must authenticate yourself with the OpenShift Enterprise Server by providing the OpenShift Login ID at the **Create New Container** screen:

3. Ensure that the **Container Type** is set to **openshift**; this creates the contain in OpenShift Enterprise instead of on your local machine.

Figure 13. OpenShift Login

4. Enter your **OpenShift Password** and click **Login to OpenShift**. After login succeeds, you the OpenShift Domain field is populated with a domain name. You can now create a new container.

Figure 14. OpenShift Password

5. Select the gear profile **fuse_medium** from the pop-up list and enter **demo1** in the **Container Name** field.
6. Click **Create and start container**.

The screenshot shows the 'Create New Container' page in the Red Hat JBoss Fuse Management Console. The page has a top navigation bar with 'Runtime', 'Wiki', 'Dashboard', and 'Health'. Below this is a sub-navigation bar with 'Containers', 'Profiles', 'Manage', 'MQ', 'APIs', 'EIPs', and 'Registry'. The main heading is 'Create New Container'. A text block explains that clicking the button will configure and start the container. Below this is a green button labeled 'Create and start container'. A section titled 'Selected Profiles' shows 'No profiles selected'. On the right, a form for 'Container type: openshift' is displayed. The form has two tabs: 'Common' and 'Advanced'. The 'Common' tab contains fields for 'Container Name' (demo1), 'OpenShift Broker' (broker00.ose.opentlc.com), 'OpenShift Login' (cmoulliard-redhat.com), 'OpenShift Password' (masked), 'Authenticate' (Login to OpenShift button), 'OpenShift Domain' (fuse), 'Gear profile' (fuse_medium), and 'Number of containers' (1).

Figure 15. Container Demo1

After a few moments, a new container appears in the JBoss Fuse Management Console under the **Runtime/Containers** submenu. This newly created container is a JBoss Fuse OSGi container, which is managed in the JBoss Fuse cartridge in OpenShift Enterprise. This mechanism is covered in detail in the *Fabric* module of this course.

Now you are ready to deploy the **Camel Twitter** application into your new container.

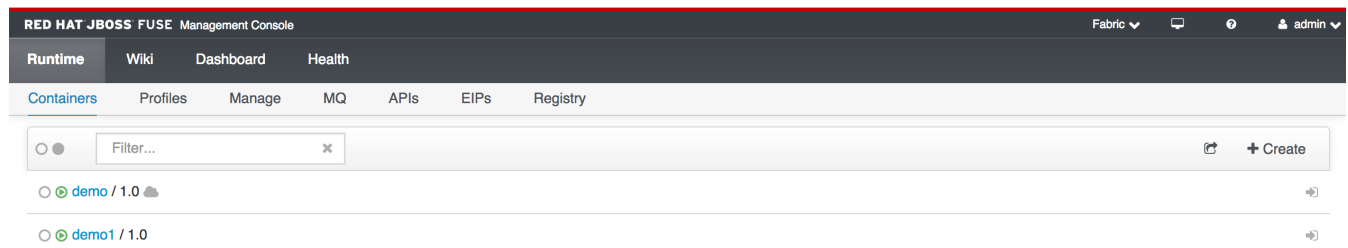


Figure 16. Container added

8.3. Deploy the Camel Twitter Application

1. Click the **demo1** container to select it.

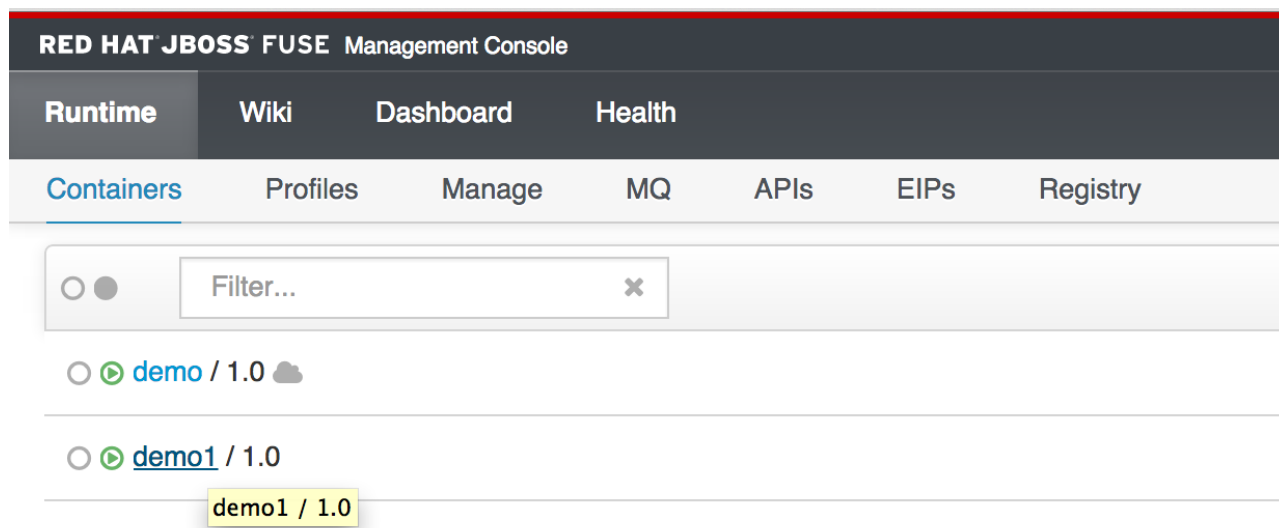


Figure 17. Select Container

2. On the **Status** tab, confirm the following information about your container:
 - **Server Status** = Running
 - **Provision Status** = success
3. On the **Settings** tab, note the following values: **Local IP**, Local Hostname* and **Public Hostname**.

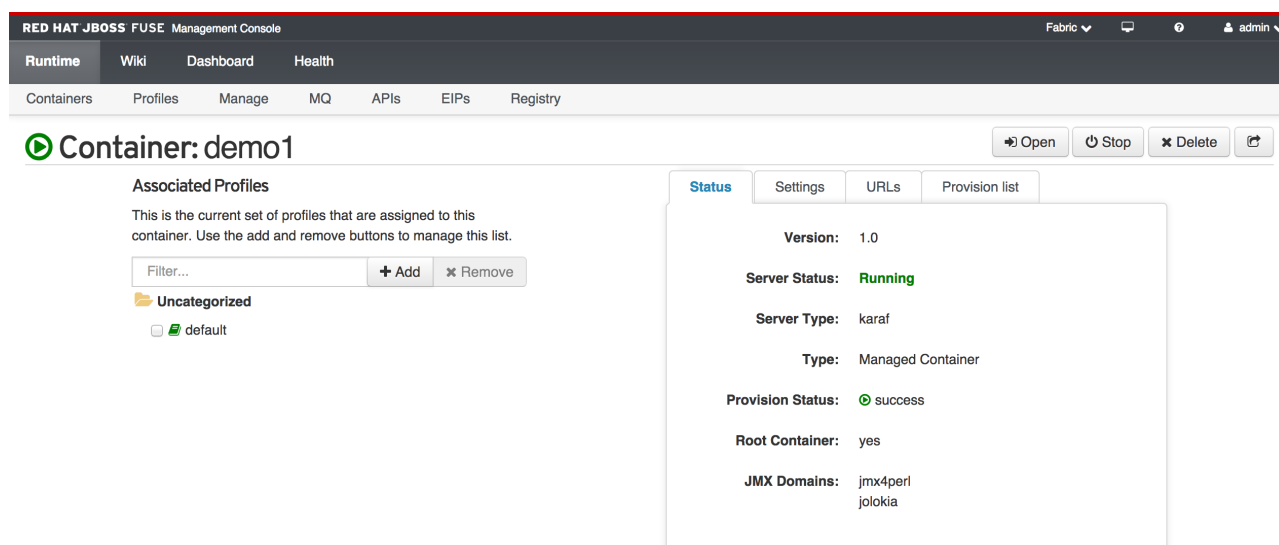


Figure 18. Container status

4. Under **Associated Profiles**, click **Add** to deploy the **Camel Twitter Application** to your container. A directory tree appears in a pop-up window, where you can filter the applications that you want to deploy (referred to as Fabric profiles).

Add profiles to container: demo1

Select one or more profiles to add to this container:

- Uncategorized
- Cloud
- Example
- Example / Camel
- Example / Camel / Loanbroker
- Example / Dosgi
- Example / Mq
- Example / Quickstarts
- Feature
- Feature / Camel
- Feature / Fabric

Add


Cancel

Figure 19. Profile list

5. Enter **twitter** in the **Filter** field and observe that the tree reduces to one branch: **Example / Camel** → **Twitter**
6. Check the **twitter** box and then click **Add**.

Add profiles to container: demo1

Select one or more profiles to add to this container:

- Example / Camel
 - ☒  twitter

Add

Cancel

Figure 20. Profile Camel Twitter

When you add a profile or an application within a container, JBoss Fuse Fabric provisions and deploys some predefined Java libraries (e.g. camel, camel-twitter) as well as the application code.

Because the **twitter** service is an Apache Camel application, a



Camel icon appears next to `camel.xml` in the container contents list on the left side of the JBoss Fuse Management Console.

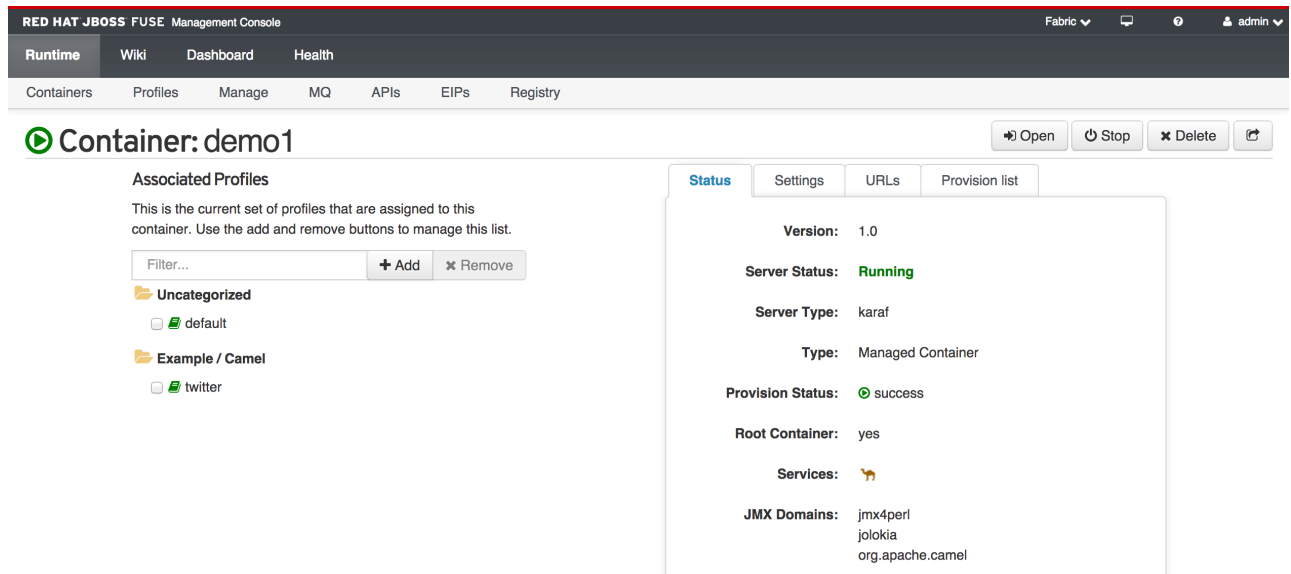


Figure 21. Container Camel Service

7. Confirm that your **Camel Twitter Application** is working correctly:
 - a. Open the `demo1` container by clicking **Open** in upper-right corner of the screen.
image::images/instruction/container_buttons.png[].
 - b. A new browser window opens. If an **Authentication Required** message appears, enter **admin** as the **User Name** and the password provided by OpenShift earlier (when you created the `demo` container) and then click **Log In**.

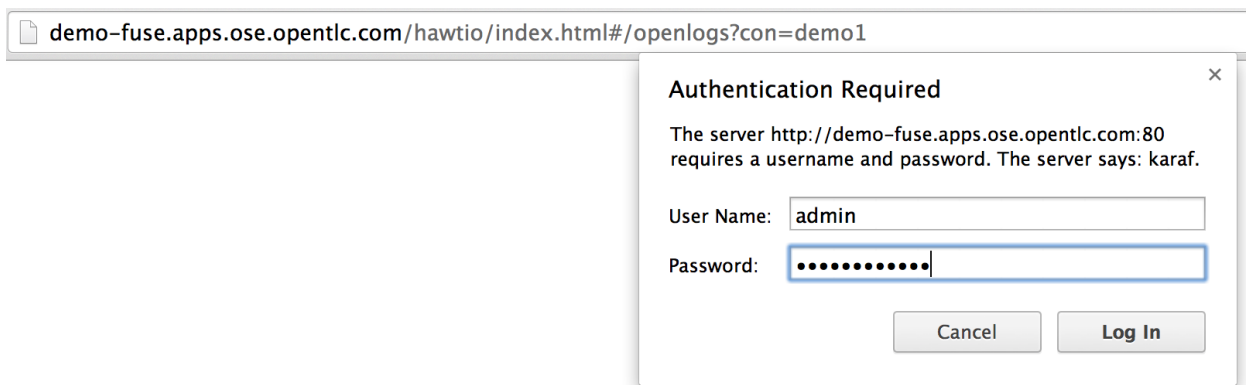
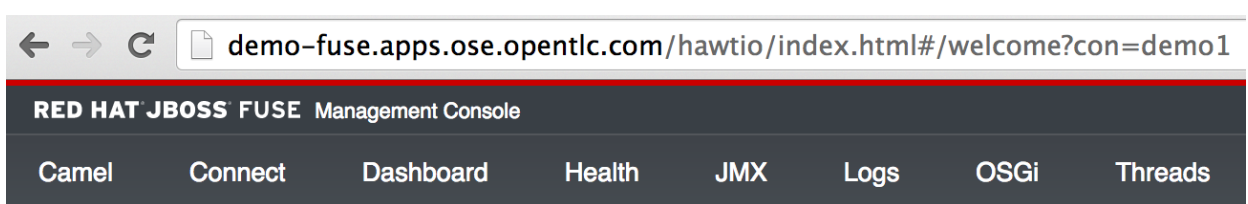


Figure 22. Screen login

The JBoss Fuse Management Console of the managed container `demo1` is displayed:



8. Discover what the Camel Twitter application is currently doing by using the **Camel** and

Logs menu options:

- a. From the **Camel** menu, select the **twitter-demo** route (**Camel Contexts** → **camel-example-twitter** → **Routes**), and then click **Diagram**.

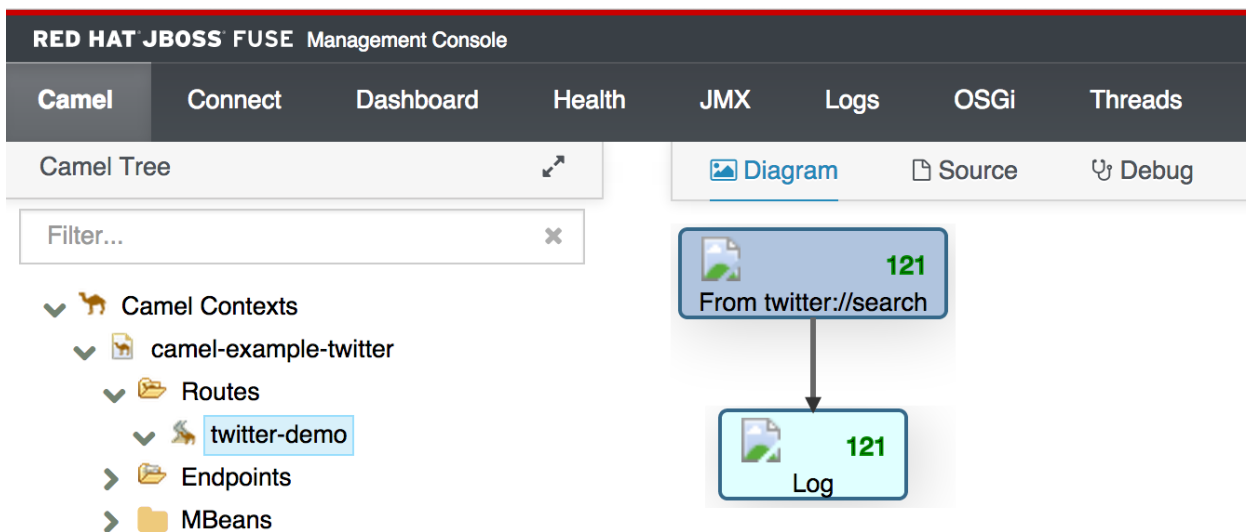


Figure 23. Camel Route

The **Apache Camel Route** connects to the twitter.com server every 10ms to search for tweets containing the word *Camel*. Every time a result is found, it is sent to the log of the **demo1** container.

- b. View the tweets received by clicking the **Logs** tab of the menu bar.
- c. Filter the results by entering 'twitter-demo' in the **Filter logs...** field.
- d. Wait for the screen to refresh. You should see the tweets received.

The screenshot shows the 'Logs' tab in the management console. The filter 'twitter-demo' is entered in the search field. The log entries are as follows:

Timestamp	Level	Source	Message
2014-10-13 13:02:11	INFO	twitter-demo	>>> priyapatell131 tweeted: RT @FreakyHumor: 12 UNBELIEVABLE GIRLS IN PANTS WITH CAMEL TOE http://t.co/xSWz...
2014-10-13 13:02:11	INFO	twitter-demo	>>> Tej_MW tweeted: @Original_Camel La laverie, du con. La laverie. Le lavoir c'est près de la mairie. (:
2014-10-13 13:02:21	INFO	twitter-demo	>>> j0sh_25 tweeted: RT @LearnSomething: This is what the Google Streetview Camel looks like. http://t.co/...
2014-10-13 13:02:31	INFO	twitter-demo	>>> F_Camel tweeted: #kakamigahara @kakamigahara_PR: 台風19号の接近にともない、自主避難所を開設しています。 http://...
2014-10-13 13:02:41	INFO	twitter-demo	>>> newna_camel tweeted: RT @chiyu_nuna: アイスの食べ方クソ萌えるwwwヨジャかお前はwww#wexo #luh...
2014-10-13 13:02:41	INFO	twitter-demo	>>> Original_Camel tweeted: @Tej_MW autant pour moi mdrrrrr
2014-10-13 13:02:52	INFO	twitter-demo	>>> A_Hiccup tweeted: who takes selfies with camel? *shakes head*
2014-10-13 13:03:02	INFO	twitter-demo	>>> Rawd_Aljanah tweeted: He could not even bear to see a camel in pain! Imagine how much Prophet Muhammad...
2014-10-13 13:03:12	INFO	twitter-demo	>>> kryzaldy_webber tweeted: RT @TweakTown: .@Google rents a camel for taking Arabian desert's #streetview...

Figure 24. Tweets received

- e. Close this browser window when you are done viewing the tweets.
9. Delete the **demo1** container; you will not use it in the next modules.
- a. Select the **demo1** container under **Runtime** → **Containers**.
 - b. Click **Stop**.
 - c. Click **Delete**.

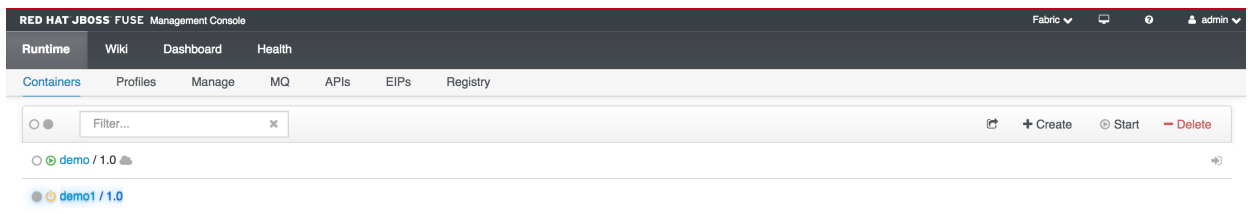


Figure 25. Delete Container

- d. When prompted to confirm deletion, click **Yes, delete**, and then close the window within your browser.

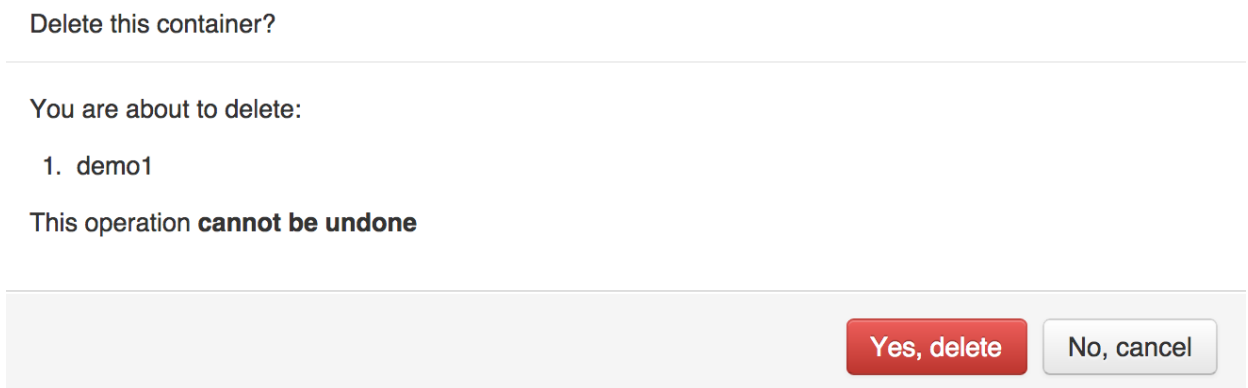


Figure 26. Deletion confirmation



This lab serves as a technology overview, and does not aim to cover all key concepts regarding JBoss Fuse. Much of what you learned in this introductory lab is directly applicable for tasks in the *Modularity* and *Fabric* labs. The OpenShift Enterprise Git project (typically included as part of an OpenShift application) is never used in this lab.

9. SSH into Your Online Lab Account

Your lab instructions include steps that require you to access your OpenShift gear via SSH. If you want to connect to the remote gear directly, the instructions are described [here](#).

10. Tail Your Application Log Files

While completing the labs in this training, you will be asked to **tail** one or more application log files using the OpenShift plug-in of JBoss Developer Studio.

1. In the OpenShift Explorer panel of JBoss Developer Studio, right-click your remote application.
2. From the drop-down list, select **Tail files**. In the dialog box that appears, enter `-f -n 100 /logs/` in the **Tail options** text box.
3. Click **Finish**.

A new Console panel appears in JBoss Developer Studio that shows the various log files of your remote OpenShift application.

Last updated 2015-11-12 12:04:12 EST