Table of Contents

Introduction to Messaging Lab

Goals:

- Verify prerequisites
- Configure local Maven environment
- Compile the lab assets project
- Download and install Red Hat® JBoss® Developer Studio and the Integration Stack plug-in
- Import the project into JBoss Developer Studio
- Set up a local Red Hat® JBoss® A-MQ server
- Create a 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
- Access a simple queue
- Publish to/read from a simple topic

Prerequisites:

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

1. Install Required Software

Software	Version	URL	Notes
Java SE	1.7 or higher	http://www.oracle.com/technetwork/java/javase/downloads/index.html	Required.
Apache Maven	3.0.5 or higher	http://maven.apache.org	Required.
JBoss Developer Studio	8.1.0.GA	http://www.jboss.org/products/devstudio/overview/	Required. Need account on jboss.org web site.
Integration Stack	8.0.2	https://devstudio.jboss.com/updates/8.0/integration-stack/	Required. Need account on jboss.org web site.
JBoss Fuse AMQ	6.2.0.GA	https://www.jboss.org/download-manager/file/jboss-amq-6.2.0.GA.zip[	Required to run exercises locally. Need account on jboss.org web site.

2. Install the Lab Exercises

The lab exercises are available in the following zip archive:

- GitHub GPE MW Training : Messaging Labs Repository

If you have not already done so, install the exercises by unzipping this archive to a location on your file system. The JBoss Fuse Messaging exercises are stored in the `messaging-labs-0.1` folder.

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

3. Configure the Local Maven Environment

The labs for this course use Apache Maven modules with dependencies on Maven libraries supported in JBoss A-MQ. Red Hat provides both online and offline Maven repositories for JBoss A-MQ.

To configure these online repositories on your machine:

1. Edit the Apache Maven `settings.xml` file located in the `home_directory/.m2 or ~/.m2` directory.

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

2. Add a new `<profile>` to `settings.xml` that contains the `JBoss Fuse` repositories that you need to build code by executing the `mvn compile` or `mvn install` commands.
3. Add the `<activeProfile>` element to `settings.xml` to set the default profile used when Apache Maven is running.

A sample `settings.xml` file is provided in the labs root folder. You can copy it to your `/.m2` folder.

```
<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 Lab Assets

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

1. Open a Linux/UNIX/MS-DOS terminal and change to the **`messaging-labs-0.1`** directory.
2. To compile the code, run the **`mvn compile`** command.
   - If the process succeeds, you see a **`BUILD SUCCESS`** message on the Console (at the end of a long list of trace results, as shown below).
   - If you see an error message instead, respond to the error reported.

```
[INFO] Reactor Summary:
[INFO]
[INFO] RedHat GPE Training :: Fuse Messaging Project ..... SUCCESS [0.256s]
[INFO] RedHat GPE Training :: Messaging :: Lab 1 :: Project  SUCCESS [0.014s]
[INFO] RedHat GPE Training :: Messaging :: Lab 1 :: Simple Queue  SUCCESS [0.278s]
[INFO] RedHat GPE Training :: Messaging :: Lab 1 :: Simple Topic  SUCCESS [0.025s]
[INFO] RedHat GPE Training :: Messaging :: Lab 2 :: Project  SUCCESS [0.002s]
[INFO] RedHat GPE Training :: Messaging :: Lab 2 :: Durable Subscription  SUCCESS [0.025s]
[INFO] RedHat GPE Training :: Messaging :: Lab 2 :: Multiple Consumer Handling  SUCCESS [0.026s]
[INFO] RedHat GPE Training :: Messaging :: Lab 2 :: Request Reply  SUCCESS [0.029s]
[INFO] RedHat GPE Training :: Messaging :: Lab 2 ::  Transacted Message Handling  SUCCESS
[0.028s]
[INFO] RedHat GPE Training :: Messaging :: Lab 3 :: Project  SUCCESS [0.002s]
[INFO] RedHat GPE Training :: Messaging :: Lab 3 :: Exclusive Consumer Handling  SUCCESS
[0.024s]
[INFO] RedHat GPE Training :: Messaging :: Lab 3 :: Hierarchical Chat  SUCCESS [0.022s]
```

```
[INFO] RedHat GPE Training :: Messaging :: Lab 3 :: Message Groups Handling  SUCCESS [0.022s]
[INFO] RedHat GPE Training :: Messaging :: Lab 3 :: Prioritized Exclusive Consumer Handling
SUCCESS [0.019s]
[INFO] RedHat GPE Training :: Messaging :: Lab 3 :: Virtual Destinations Handling  SUCCESS
[0.017s]
[INFO] RedHat GPE Training :: Messaging :: Lab 4 :: Project  SUCCESS [0.014s]
[INFO] RedHat GPE Training :: Messaging :: Lab 4 :: Failover  SUCCESS [0.015s]
[INFO] RedHat GPE Training :: Messaging :: Lab 4 :: Failover Dynamic  SUCCESS [0.016s]
[INFO] RedHat GPE Training :: Messaging :: Lab 4 :: Network of Brokers  SUCCESS [0.009s]
[INFO] RedHat GPE Training :: Messaging :: Lab 4 :: Persistence  SUCCESS [0.011s]
[INFO] RedHat GPE Training :: Messaging :: Lab 5 :: Project  SUCCESS [0.002s]
[INFO] RedHat GPE Training :: Messaging :: Lab 5 :: Security  SUCCESS [0.011s]
[INFO] RedHat GPE Training :: Messaging :: Lab 5 :: JMX .. SUCCESS [0.009s]
[INFO] RedHat GPE Training :: Messaging :: Lab 5 :: Logging Audit  SUCCESS [0.009s]
[INFO] RedHat GPE Training :: Messaging :: Lab 6 :: Project  SUCCESS [0.017s]
[INFO] RedHat GPE Training :: Messaging :: Lab 6 :: A-MQ & Camel  SUCCESS [0.017s]
[INFO] RedHat GPE Training :: Messaging :: Lab 6 :: A-MQ & Camel Producer  SUCCESS [0.451s]
[INFO] RedHat GPE Training :: Messaging :: Lab 6 :: A-MQ & Camel Consumer  SUCCESS [0.030s]
[INFO] RedHat GPE Training :: Messaging :: Lab 6 :: A-MQ & Camel Service  SUCCESS [0.034s]
[INFO] RedHat GPE Training :: Messaging :: Lab 6 :: A-MQ & Camel - Features  SUCCESS [0.118s]
[INFO] RedHat GPE Training :: Broker ..................... SUCCESS [0.005s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

5. Install 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).

You can download JBoss Developer Studio from jboss.org. The install guide is available from https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Developer_Studio/7.1/html/Getting_Started_Guide/Install_JBoss_Developer_Studio.html.

JBoss Developer Studio includes a variety of Eclipse plug-ins. The following table lists the JBoss Developer Studio plug-ins that you need to complete the labs in this course:

Plug-in	Description
Integration Stack Suite	This suite plug-ins is important for use with JBoss Fuse and Red Hat® JBoss® A-MQ middleware but is not included with JBoss Developer Studio. You must manually install the suite by following the instructions at https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Developer_Studio/7.1/html-single/Integration_Stack_Guide/index.html#chap-Introduction_to_Red_Hat_JBoss_Developer_Studio_Integration_Stack.
OpenShift Enterprise Plug-in	JBoss Developer Studio includes an out-of-the-box plug-in for managing OpenShift Enterprise environments. No additional installation is required to use this plug-in. With this plug-in you can fully manage your remote OpenShift Enterprise environment (for example, upload SSH keys and manage domains and apps).

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 for Git project support. No additional installation is required to use this plug-in. Git is an open source version control system that provides 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 to support 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 Into JBoss Developer Studio

Now that JBoss Developer Studio is installed, import the `messaging-labs-0.1` Apache Maven project.

1. Open **JBoss Developer Studio**.
2. Import the `messaging-labs-0.1` Apache Maven project into a new workspace:
   a. Select **File** → **Import** from the menu.
   b. Select **Maven** → **Existing Maven Projects**.

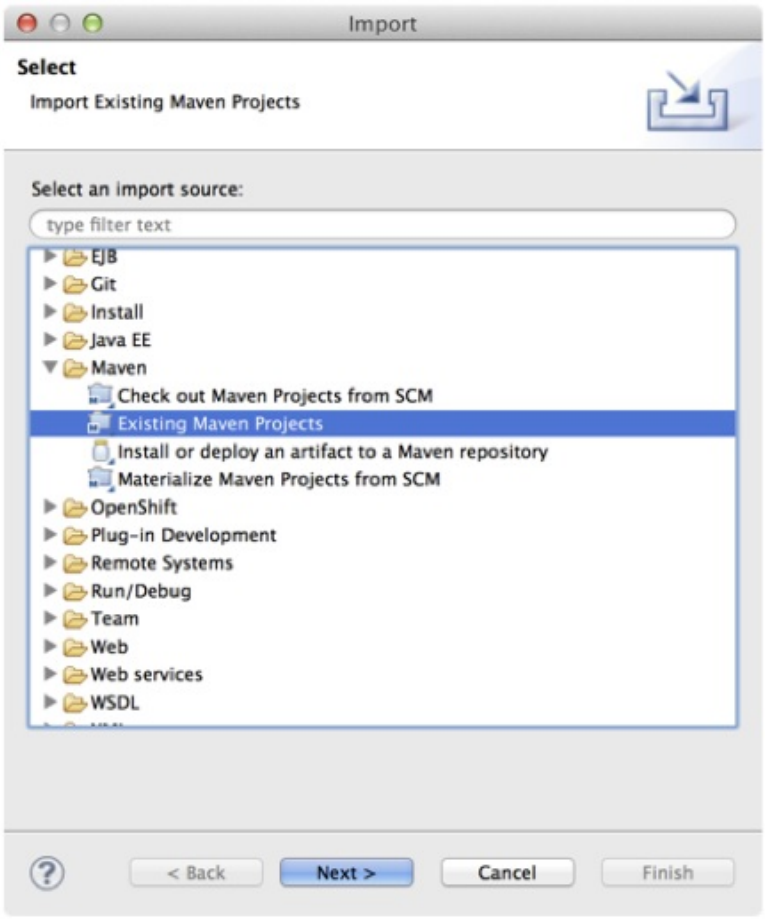

Figure 1. Import lab project (part 1)

c. Click **Next**.

d. Click **Browse** and navigate to the `messaging-labs-0.1` folder.

e. Select `/pom.xml com.redhat.gpe.training:fuse-messaging`.

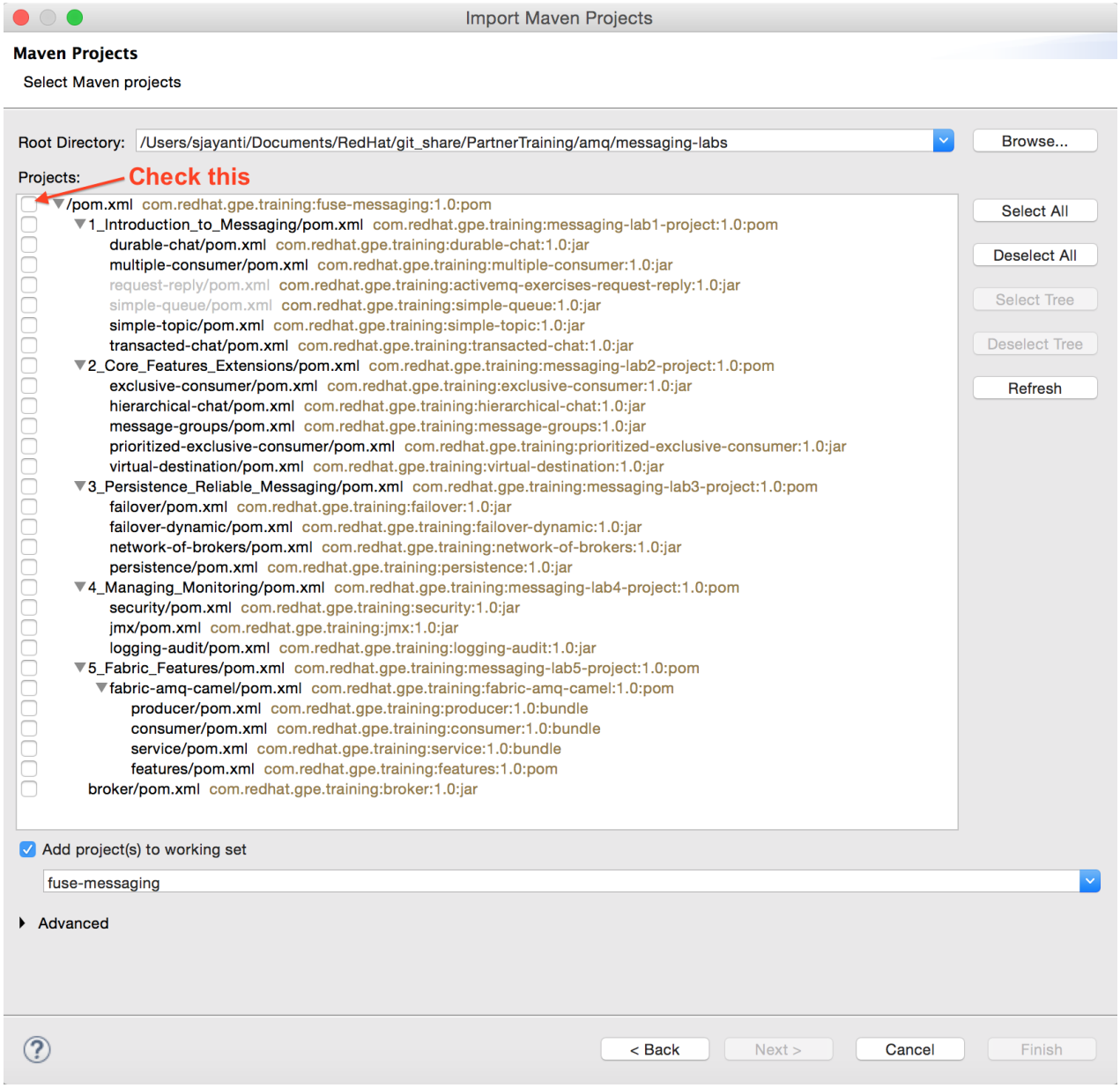

Figure 2. Import lab (part 2)

f. Click **Finish**.

3. Confirm that the project imported correctly and that you can compile it using Maven:

a. From the `pom.xml` file of the **fuse-messaging** parent project, select **Run as** → **Maven install**.

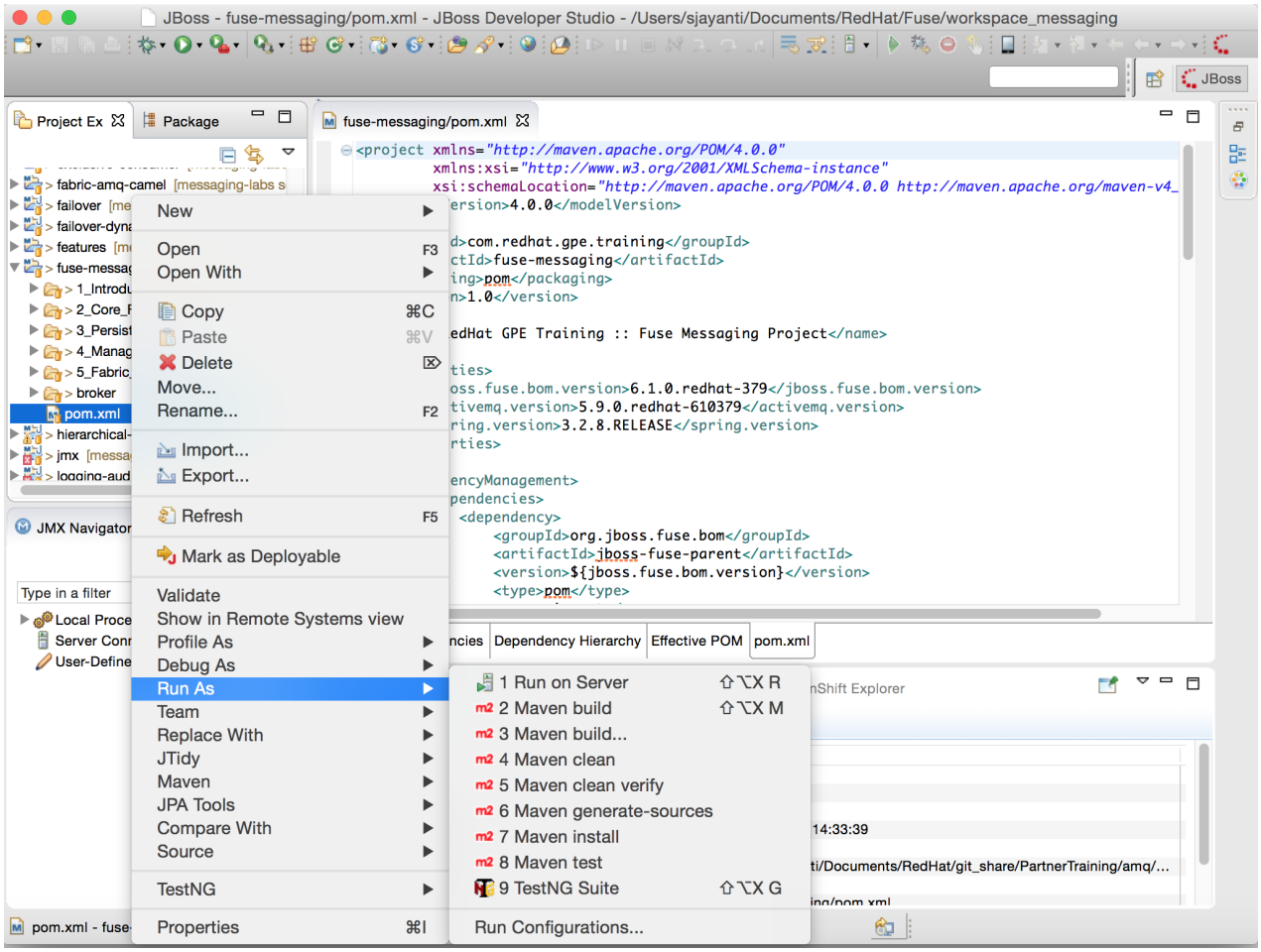

Figure 3. Verifying project import

b. Check the Console output.

- If the process was successful, you see the **`BUILD SUCCESS`** message.

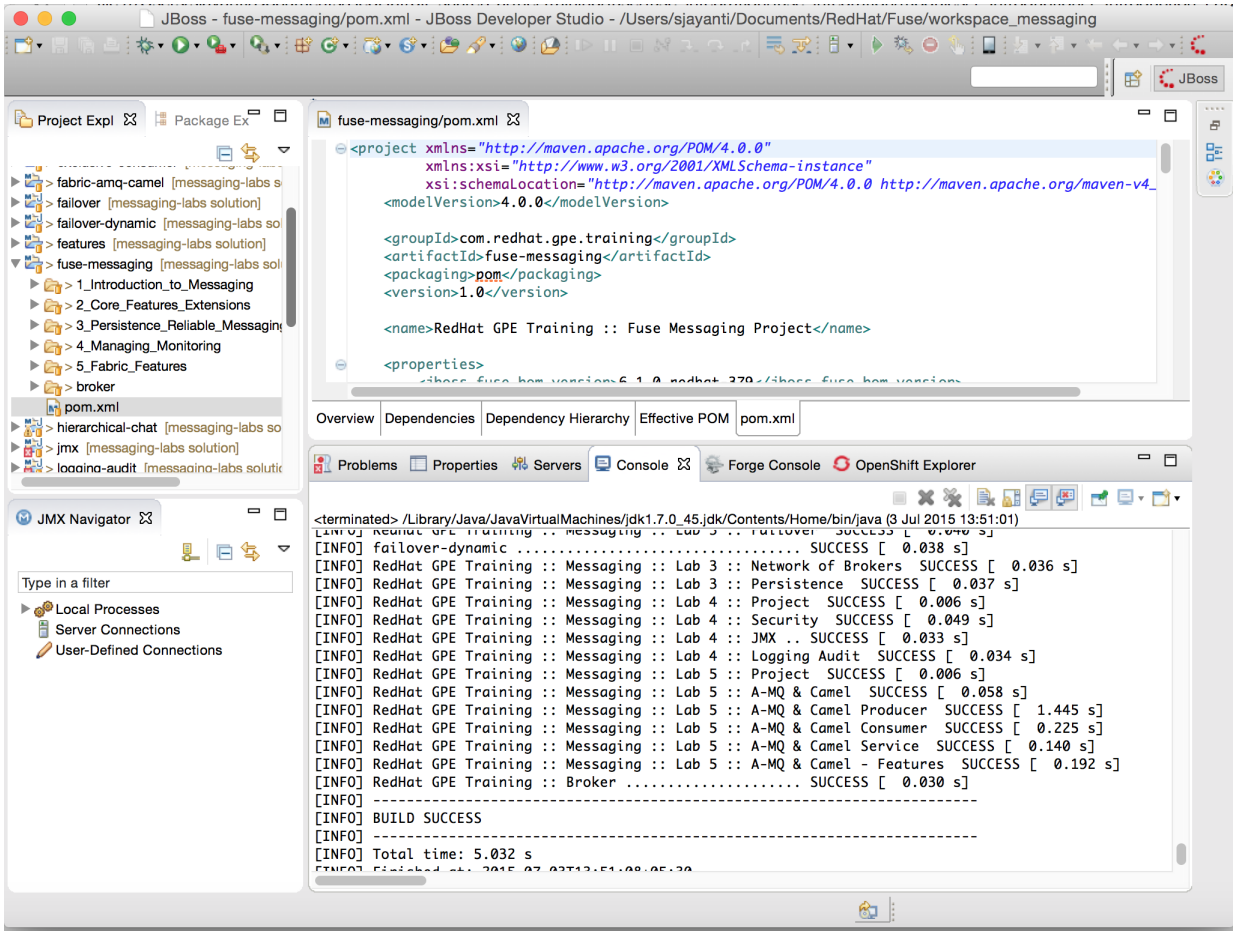

Figure 4. Build success result

7. Install JBoss Fuse AM-Q as a Local Server

Do this procedure only if you are running your JBoss A-MQ broker on a local machine. If you are not, go to the next section.

For the labs to work on a Windows system, you need to edit the `hosts` file. Open `C:\Windows\System32\drivers\etc\hosts`, add `0.0.0.0 localhost` to the end of the file, and then save and close the file.

1. Download JBoss A-MQ from http://www.jboss.org/products/amq/download/.
   - The install guide is available from https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_A-MQ/6.2/html/Installation_Guide/index.html.
2. Navigate to the `$AMQ_INSTALL_DIR/bin` folder and run `./amq`.
   - If the A-MQ server was installed correctly, you see the Karaf console:

```
$ ./amq
Please wait, JBoss A-MQ is initializing...
100% [========================================================================]

       _ ____                                __  __  ____
      | |  _ \                    /\        |  \/  |/ __ \
      | | |_) | ___  ___ ___     /  \ ______| \  / | |  | |
  _   | |  _ < / _ \/ __/ __|   / /\ \______| |\/| | |  | |
 | |__| | |_) | (_) \__ \__ \  / ____ \     | |  | | |__| |
  \____/|____/ \___/|___/___/ /_/    \_\    |_|  |_|\___\_\

  JBoss A-MQ (6.2.0.redhat-133)
  http://www.redhat.com/products/jbossenterprisemiddleware/amq/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.

Open a browser to http://localhost:8181 to access the management console

Hit '<ctrl-d>' or 'osgi:shutdown' to shutdown JBoss A-MQ.

No user found in etc/users.properties. Please use the 'amq:create-admin-user'
command to create one.

JBossA-MQ:karaf@root>
```

3. Create the `admin` user to start accessing the Management Console and connect via the MQ broker:
   a. Open the `$AMQ_HOME/etc/user.properties` file and uncomment the following line:

```
admin=admin,admin
```

   b. Save the file and restart the JBoss A-MQ server.
4. Test the login:

a. In your browser, access the management console by opening http://localhost:8181.

b. Enter `admin` and `admin` in the **Userid** and **Password** fields.

   - You should see the welcome screen.

8. Create a JBoss Fuse Application in OpenShift Enterprise

You need to do this procedure only if you are using the OpenShift gear to run your JBoss A-MQ broker. If you are not using the OpenShift gear, go to the next section.

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 lifecycle by providing tooling support.

You use OpenShift Enterprise as a platform to host your JBoss A-MQ applications during this training. The procedure is described in the *GPE Common* labs. You use OpenShift Explorer in JBoss Developer Studio to reach this goal.

1. Follow the steps defined at http://people.redhat.com/jbride/labsCommon/setup.html#_create_connection_to_openshift_from_jbds 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 (fuse-1.0.0)** cartridge:

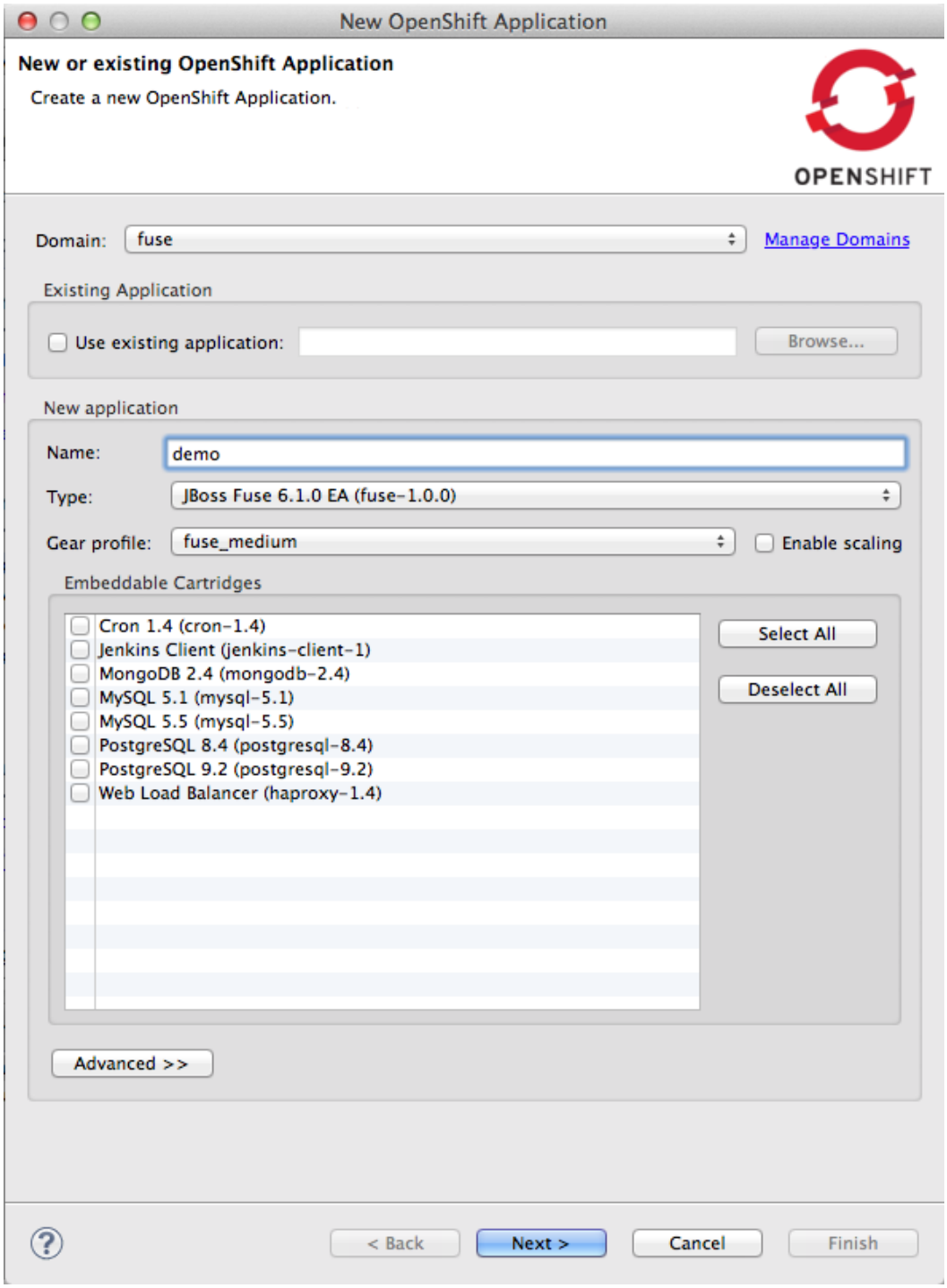

Figure 5. JBoss Fuse cartridge

3. Save the passwords generated for the JBoss Fuse application. You need them to access the web-based JBoss Fuse Management Console.

The passwords appear only when the gear is created. You cannot retrieve them unless you connect with SSH to OpenShift Enterprise.

Figure 6. Generated passwords

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

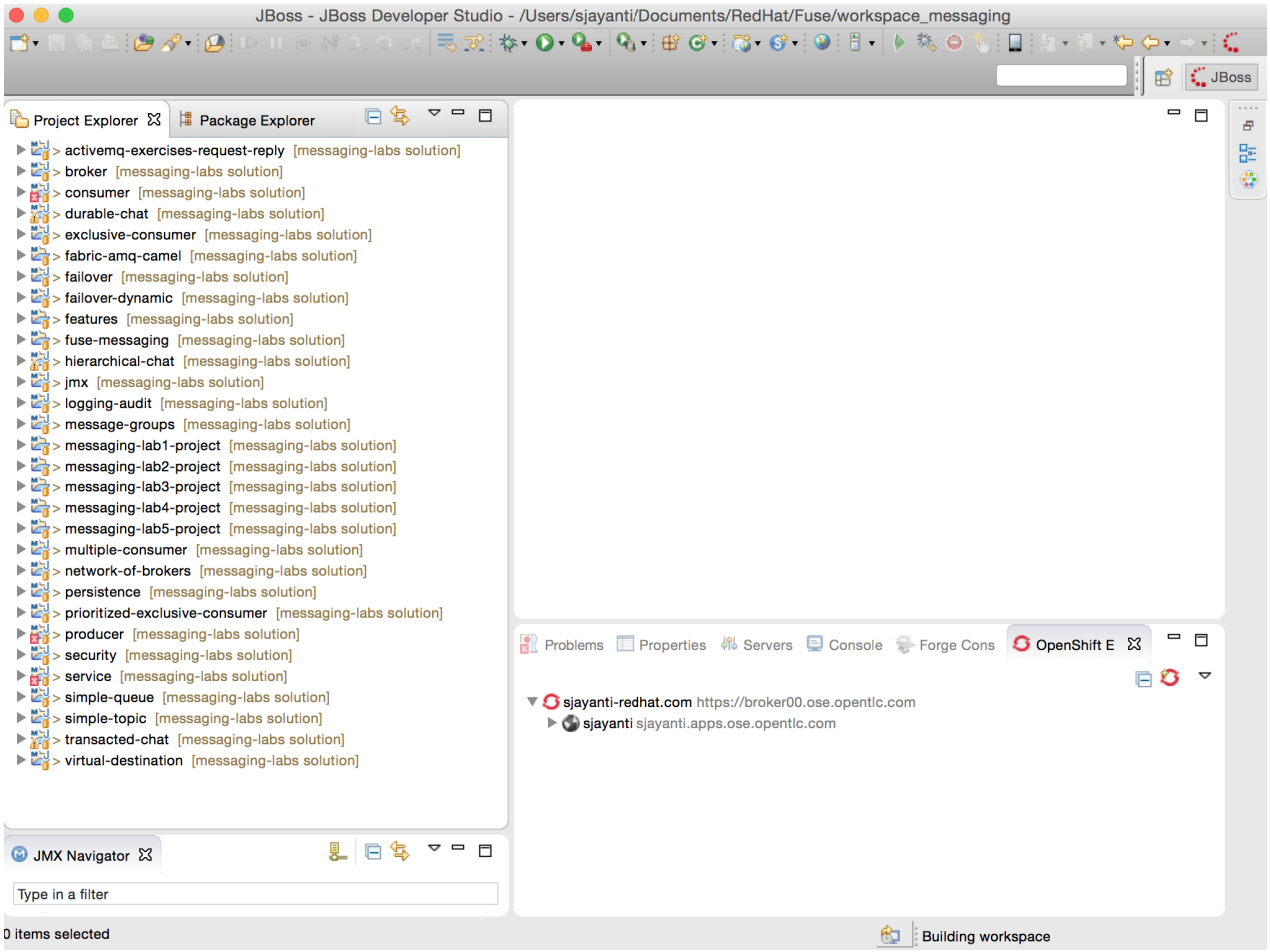

Figure 7. OpenShift Explorer

9. Access the JBoss Fuse Management Console

The procedures in this section are for using the OpenShift JBoss Fuse server. If you are using the localhost broker, you can skip this section because the local broker is available by default. To access the Management Console, go to http://localhost:8181 and use the default userid/password: **admin/admin**.

Next, you use a browser to access the JBoss Fuse Management Console, create a new broker, and assign the broker to a new container.

9.1. Access Application Details

1. Expand the **OpenShift connection** (http://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**.

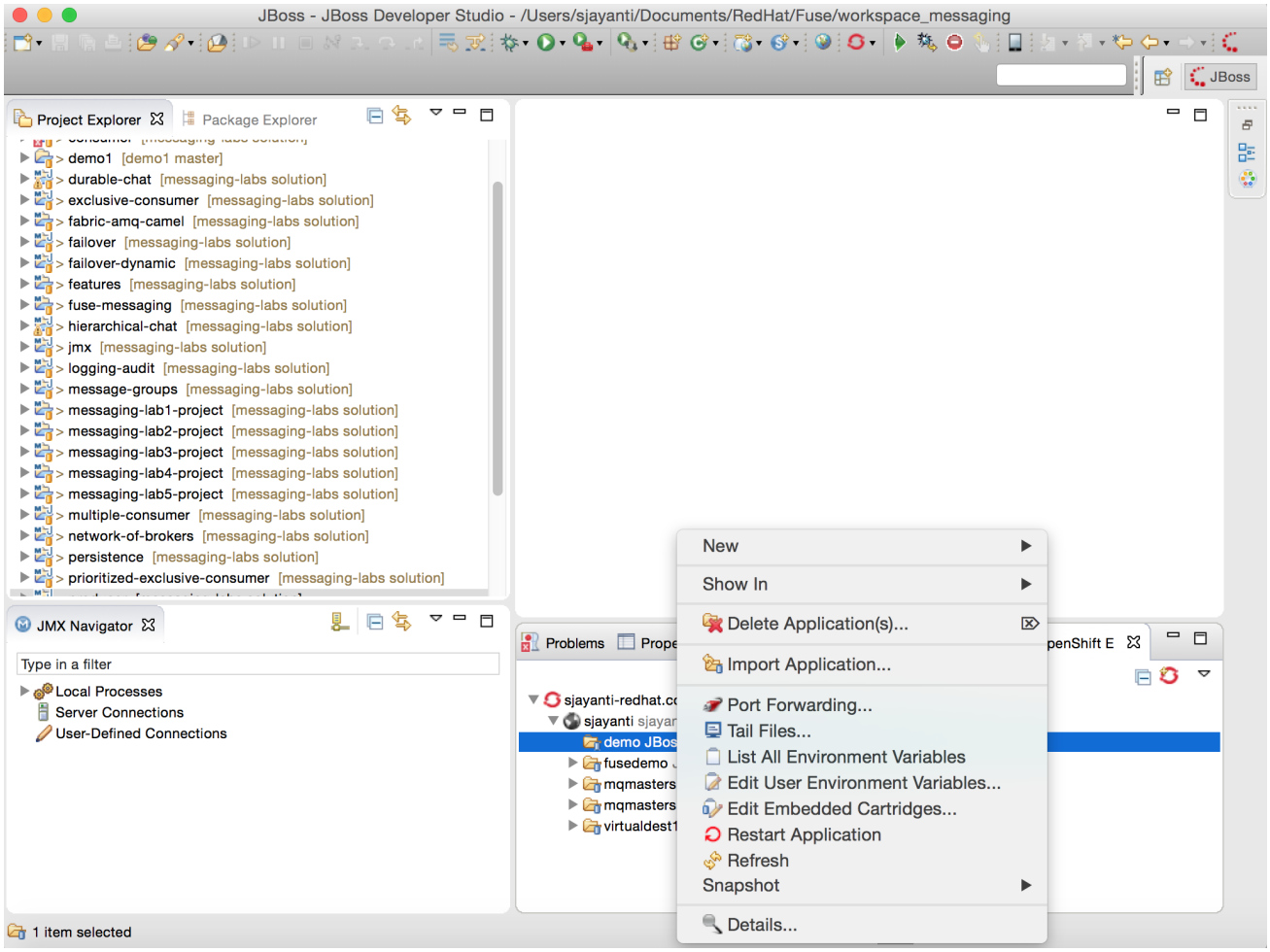

Figure 8. OpenShift application name

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

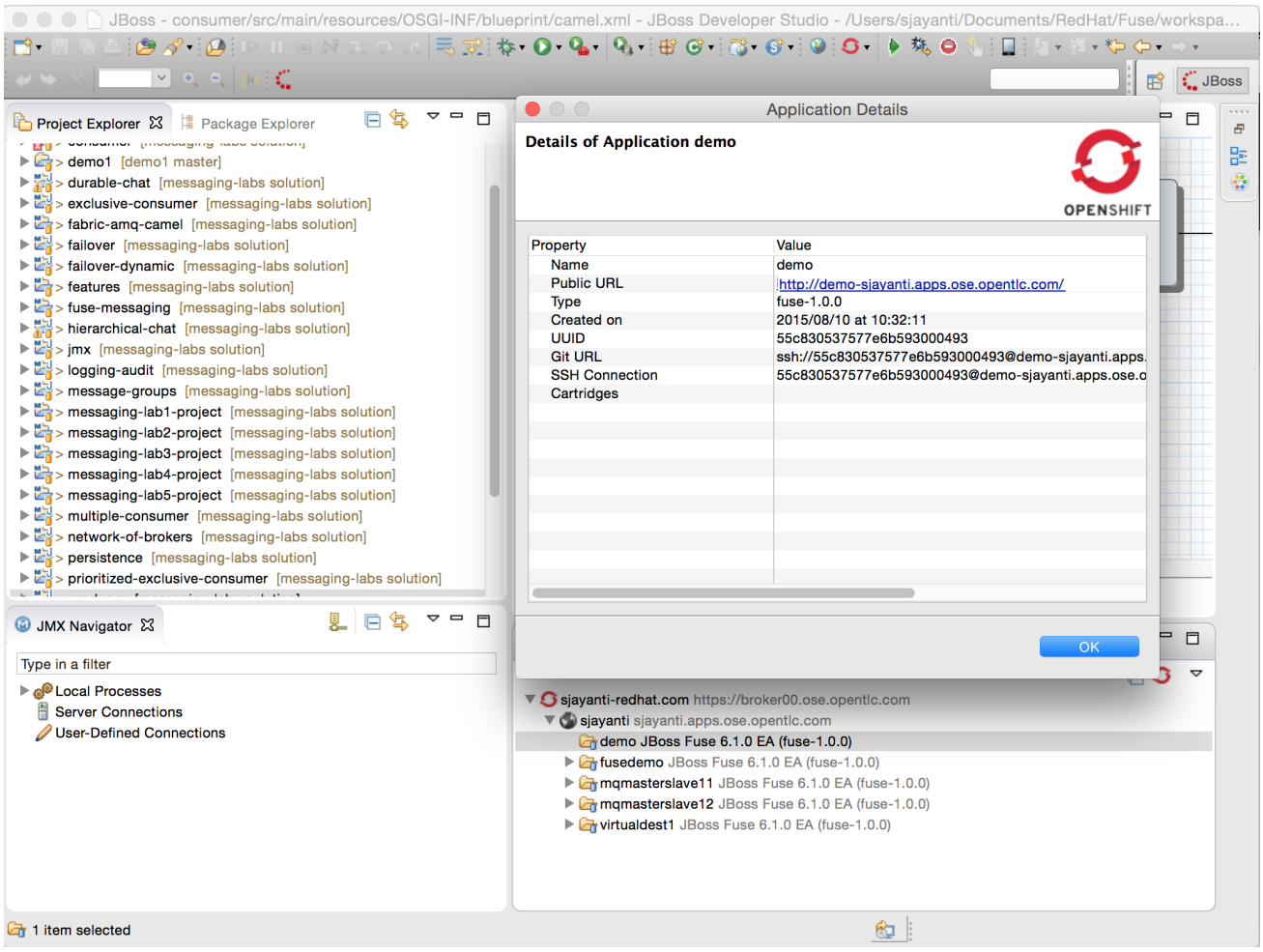

Figure 9. Application details - demo

4. Provide your username (`admin`) and the password captured when you created the OpenShift application.

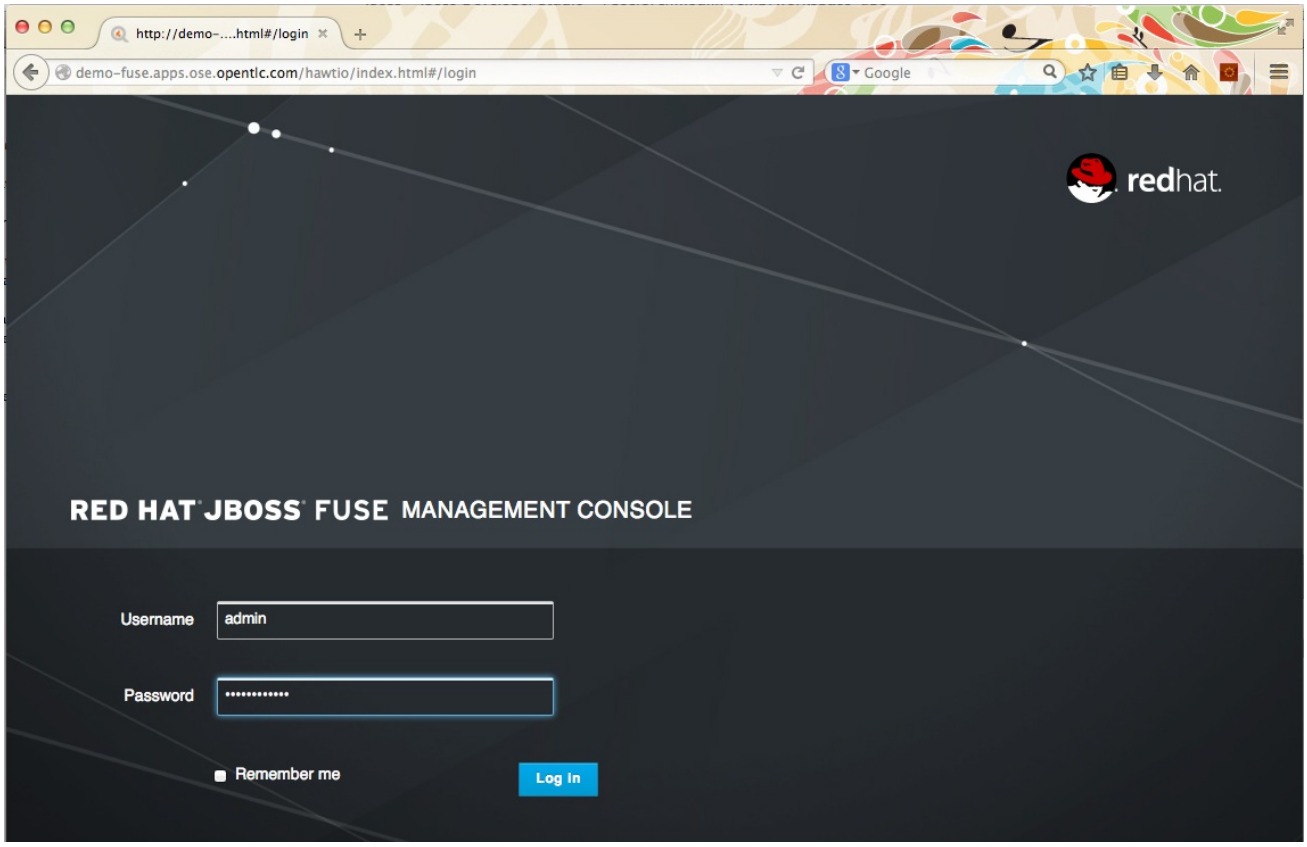

Figure 10. JBoss Management Console Login screen

- 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 Management Console includes documentation about the various menu options. To access the documentation, click **Help** () in the main navigation bar.

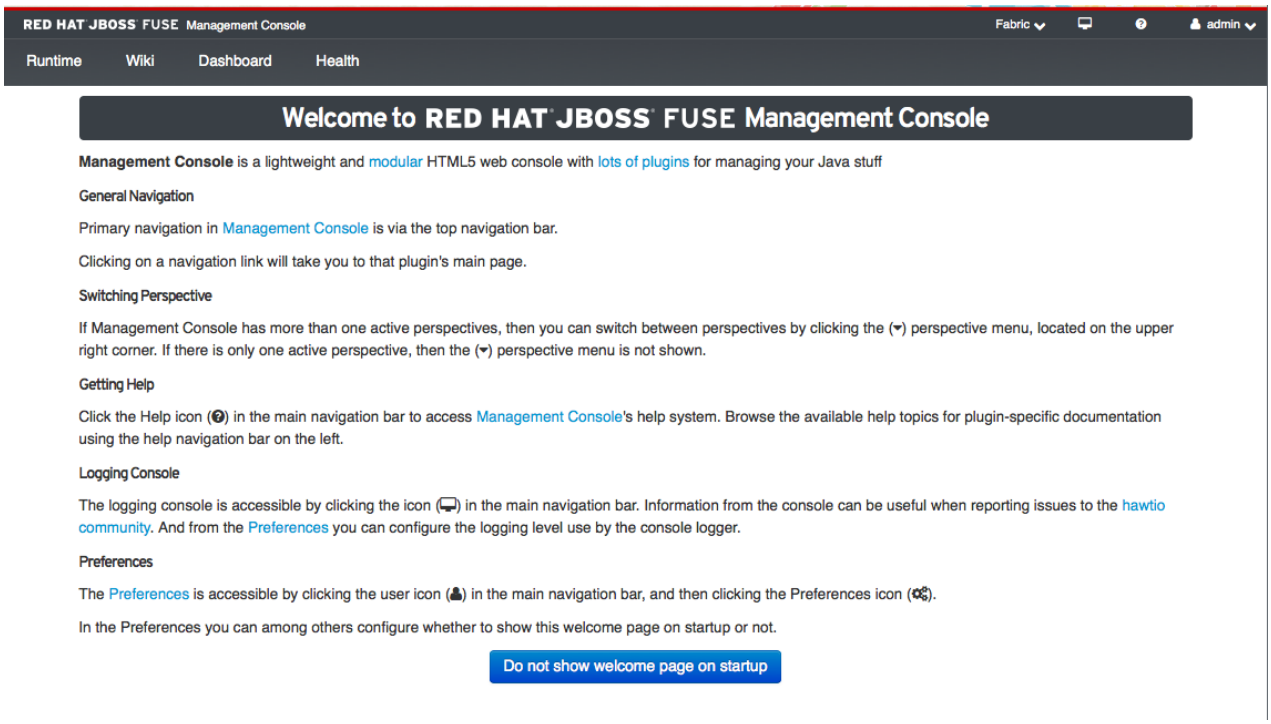

Figure 11. JBoss Fuse Management Console menu

Next, to familiarize you with JBoss Fuse Messaging technology, you will:

- Create a new JBoss Fuse instance (which is a Java OSGi container)
- Create a new MQ broker and associate it with a new container

9.2. 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).

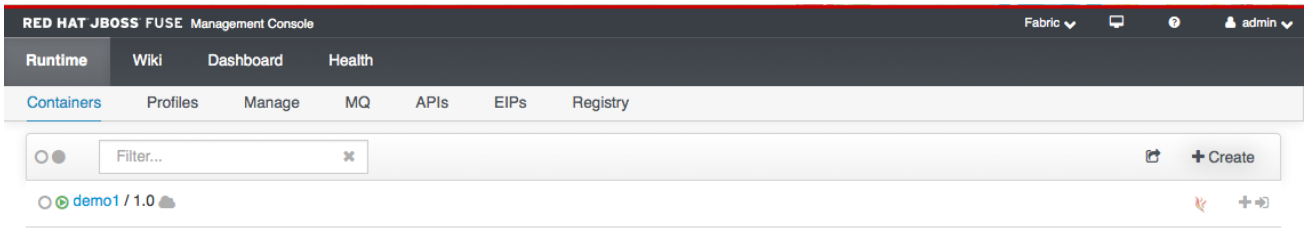

Figure 12. Create a container

3. Authenticate yourself with the OpenShift Enterprise server by providing the OpenShift login ID at the **Create New Container** screen.
4. Verify that **Container type** is set to **openshift**.
   - This creates the container in OpenShift Enterprise instead of on your local machine.

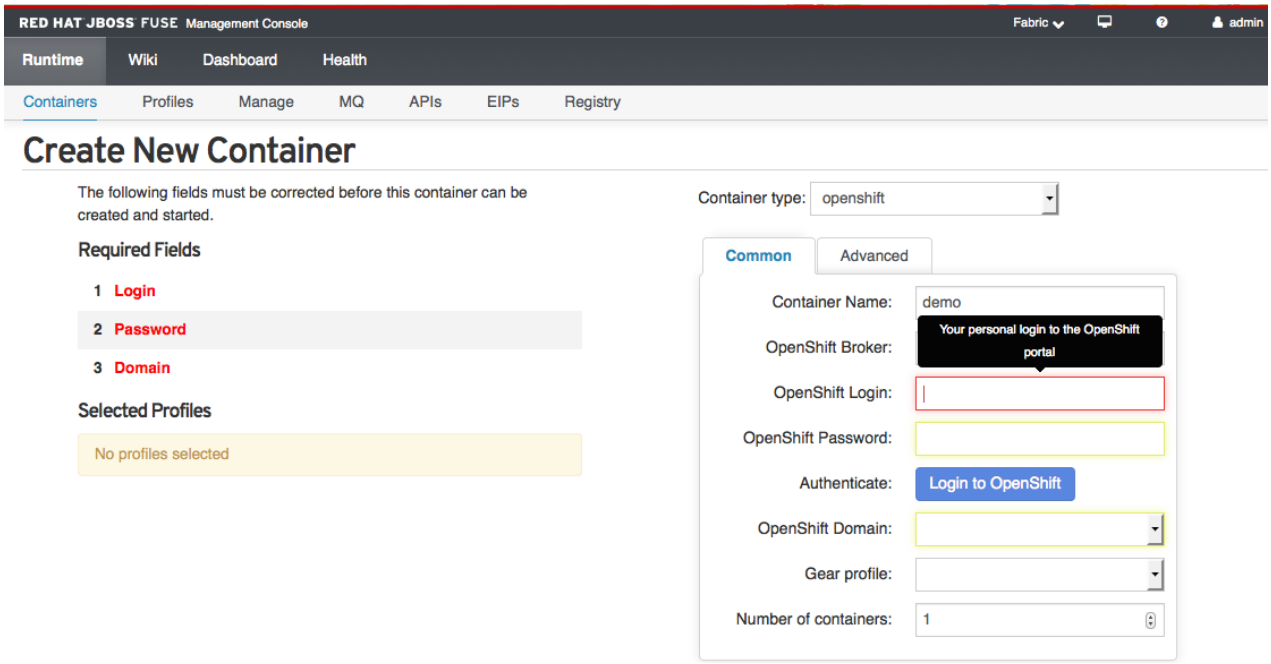

Figure 13. OpenShift login

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

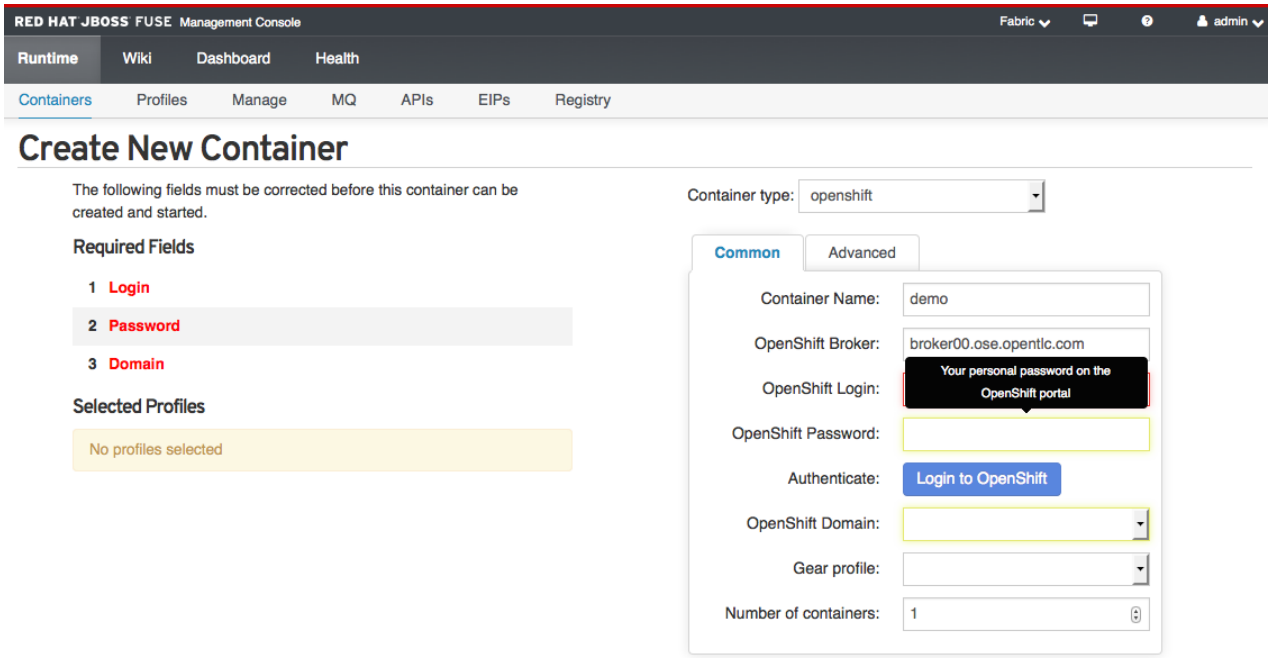

Figure 14. OpenShift portal password

6. Select the gear profile **fuse_medium** from the list and enter `demo1` in the **Container Name** field.
7. Click **Create and start container**.

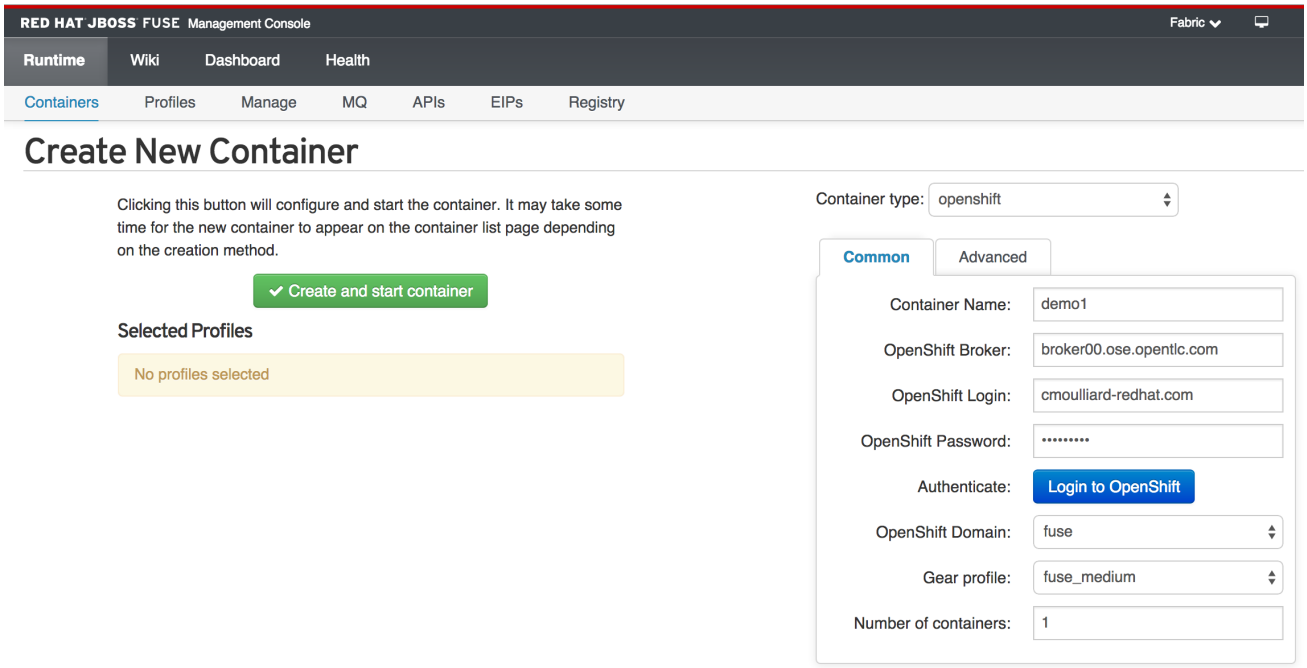

Figure 15. Create and start container

- After a few moments, a new container appears in the JBoss Fuse Management Console in the **Runtime/Containers** submenu. This JBoss Fuse OSGi container is managed in the JBoss Fuse cartridge in OpenShift Enterprise.

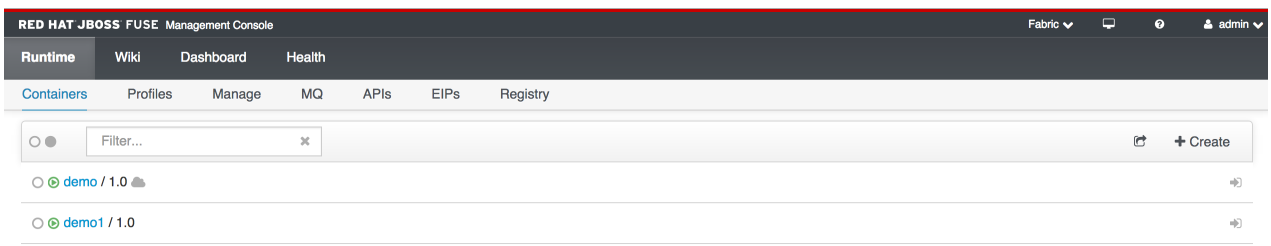

Figure 16. Container added

9.3. Create and Configure a New MQ Broker

1. Navigate to the **MQ** menu. No brokers are configured.
2. Create a new broker:
   a. Click **Create broker configuration**.

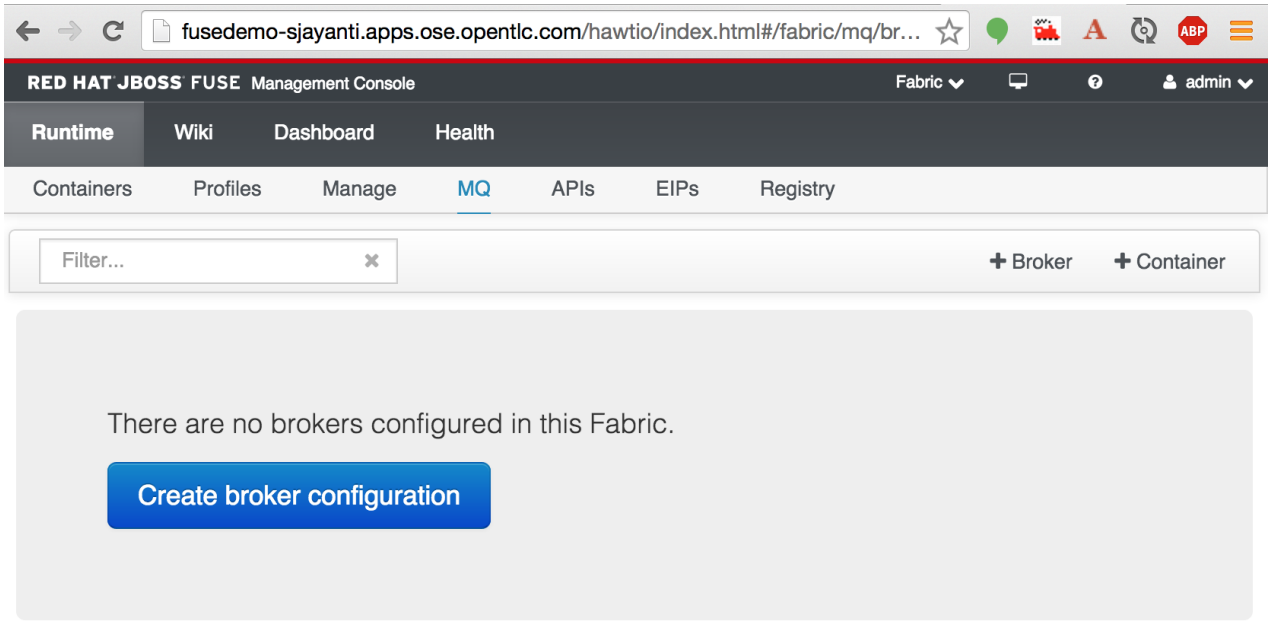

Figure 17. MQ menu

   b. In the **Create Broker** form, enter the following information:
      - **Broker name:** `demoBroker`
      - **Kind:** `StandAlone`
      - Leave the defaults in other fields
   c. Click **Create broker**.

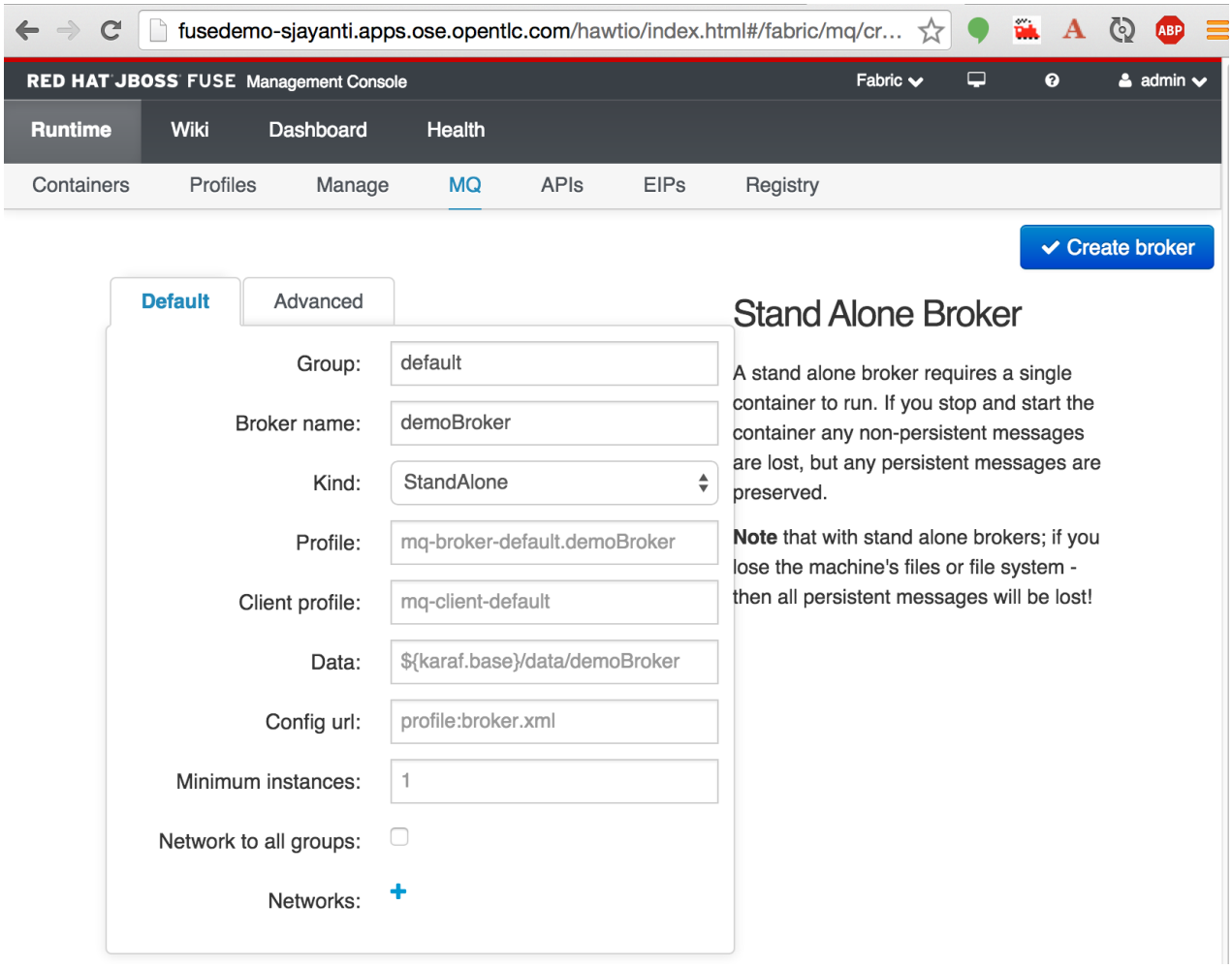

Figure 18. Create standalone broker

3. After the broker is created, view it under the **MQ** tab:

A warning icon displays because the broker is not yet assigned to any containers. For a broker to be started, it needs to run within a container.

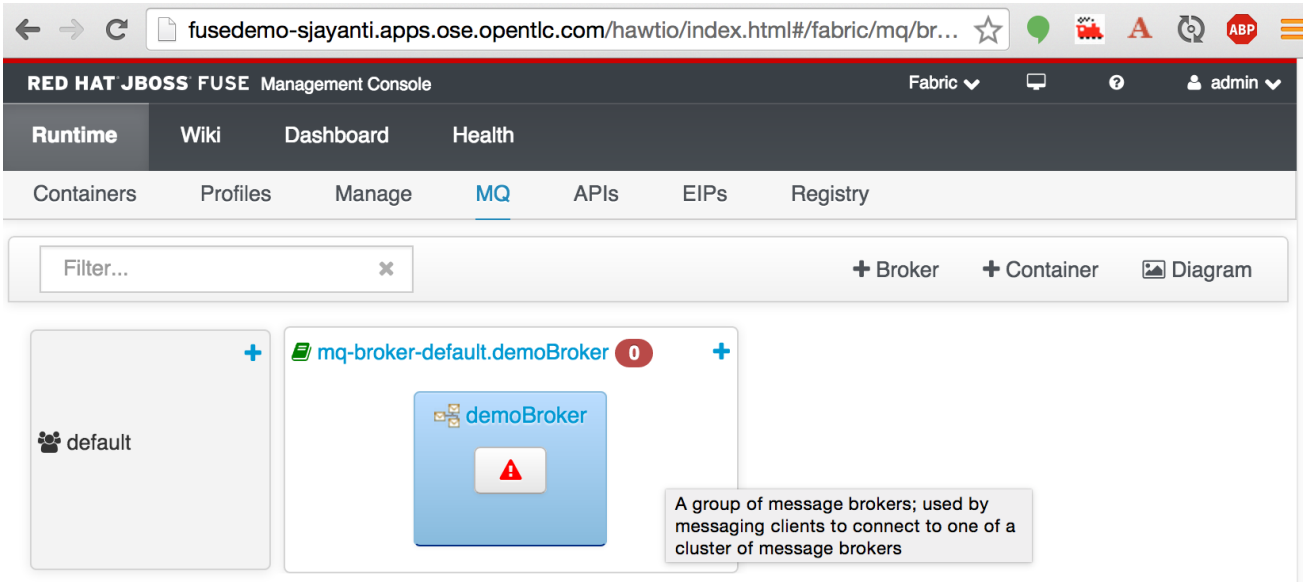

Figure 19. New demoBroker

4. Create a new container:
   a. Click the red warning triangle and select **Create a New Container**.
   b. Enter the following details:

- **Container type: openshift**
- **Container Name: `mqdemo`**
- Leave the defaults in other fields

c. Enter your OpenShift login and password.

d. Click **Login to OpenShift**.

e. Click **Create and start container**.

- Execution takes a little while.

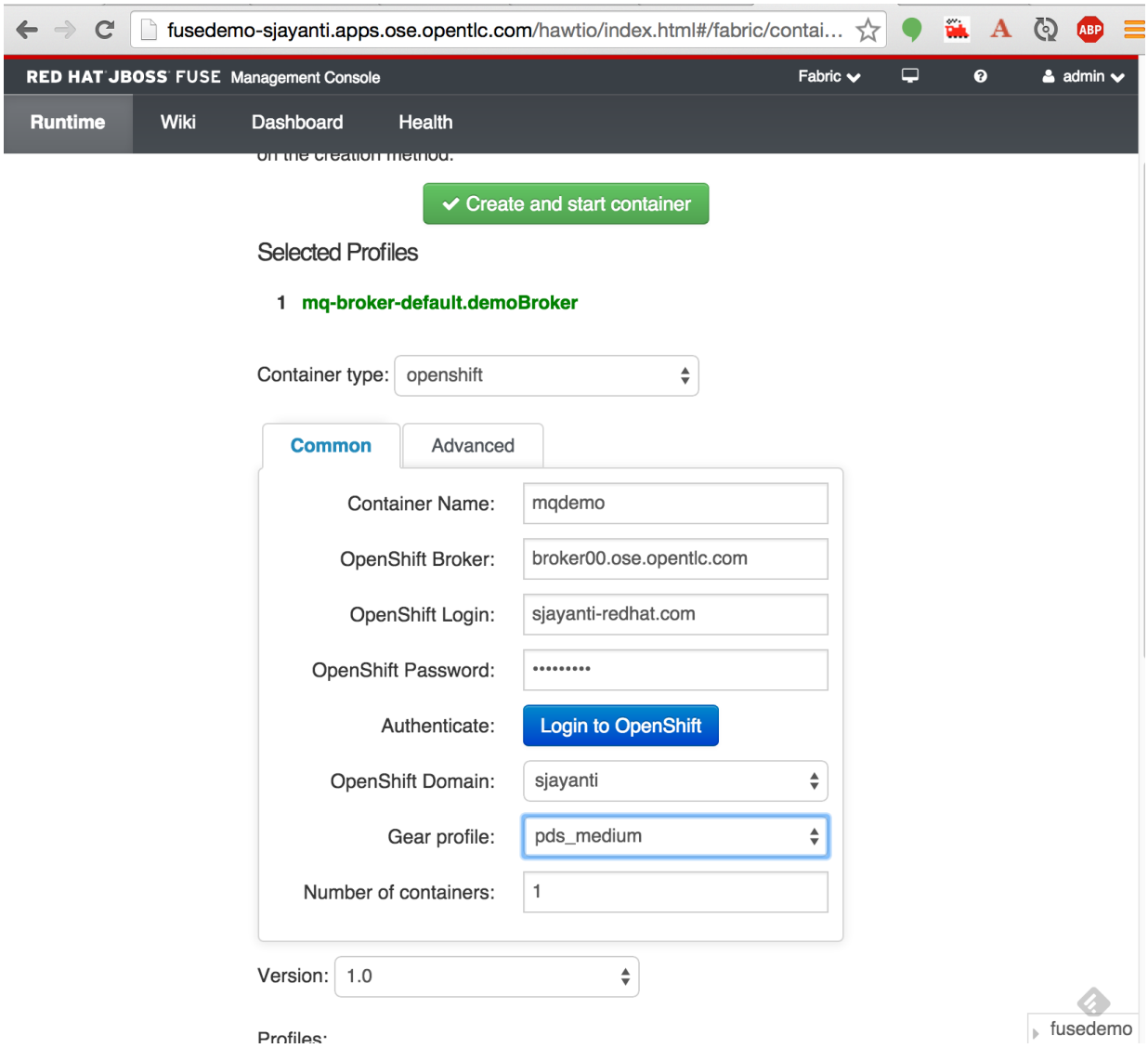

Figure 20. Create and start container

5. Click **Containers** to see two containers, with the `mqdemo` container created and started successfully.

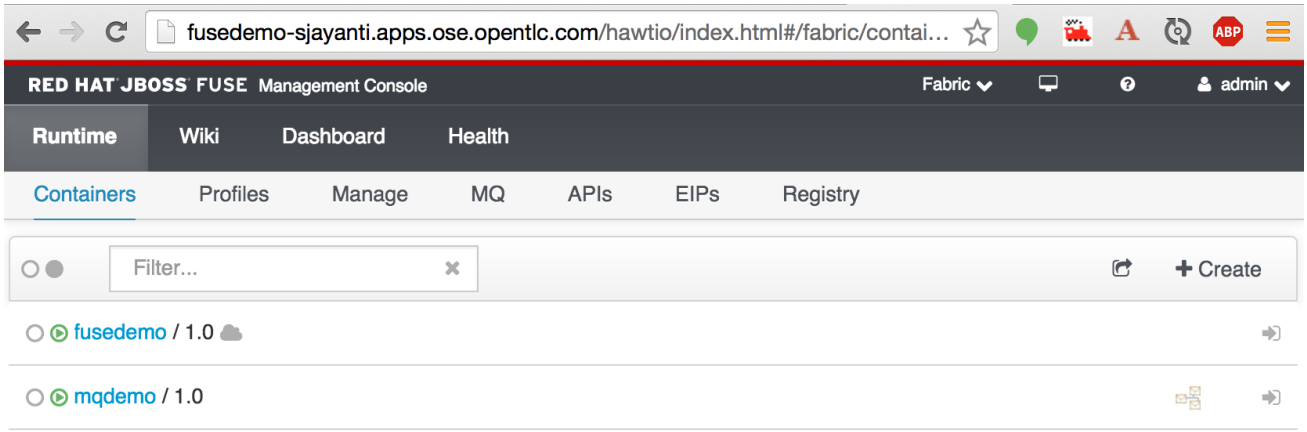

Figure 21. Two new containers: `fusedemo` and `mqdemo`

6. Click the `mqdemo` container to see the status and other details associated with the container.

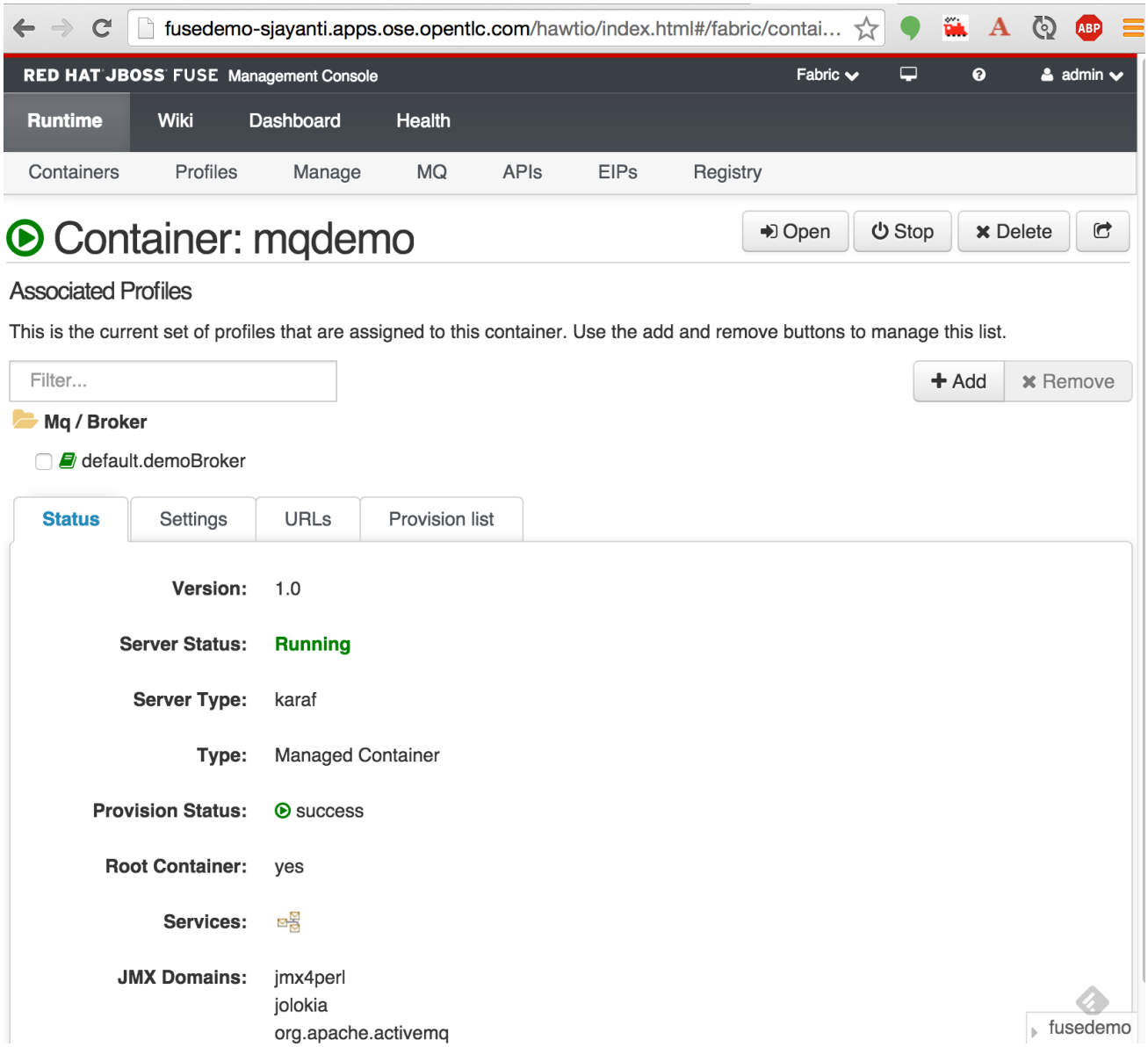

Figure 22. mqdemo container

7. To get the URL of the broker for connections, click **Registry** and follow the links to **fusemq**, **default**, and the service number **0000002**.
   - Now **Services** shows the TCP URL that external clients can use to connect to the message broker.

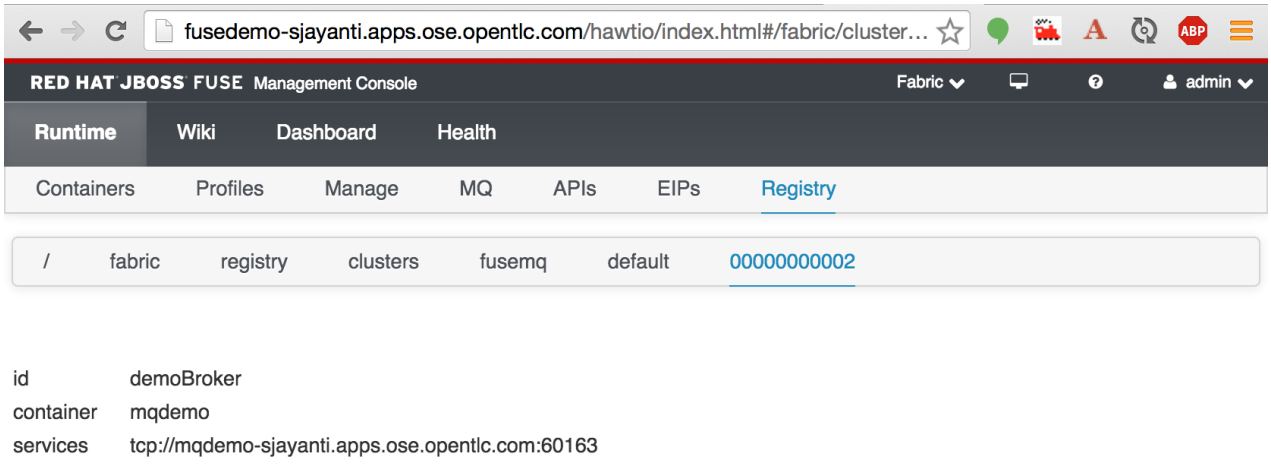

Figure 23. Broker TCP URL

Record this URL. Using `admin` user and the Fuse Management Console password, you will connect to this instance in the following labs.

+

This lab serves as a technology overview. It provides a working environment for the remaining lab exercises in this course.

10. Connect via SSH to Your Online Lab Account

You need to complete this exercise only if you are using the OpenShift gear to run your JBoss A-MQ broker. If you are not using the OpenShift gear, go to the next section.

Use SSH to connect to your lab account.

Your lab instructions explain how to use an SSH connection to access your OpenShift gear. If you want to connect to the remote gear directly, the instructions are described at http://people.redhat.com/jbride/labsCommon/setup.html#_ssh_into_your_application.

11. Tail Your Application Log Files

You need to do this procedure only if you are using the OpenShift gear to run your JBoss A-MQ broker. If you are not using the OpenShift gear, go to the next section.

While completing the labs in this training, you are 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**.
3. In the dialog that appears, in the **Tail options** text field, enter `-f -n 100 /logs/`.
4. Click **Finish**.
    - A new Console panel appears in JBoss Developer Studio that shows the various log files of your

remote OpenShift application.

12. Running the Labs

To run the labs compiled at the beginning of this module, you need a running A-MQ broker that can be connected. It can be the local broker discussed in section 7 or the OpenShift broker discussed in section 9.3.

By default, the localhost broker configuration is provided for the labs. If you are using the localhost broker, you do not need to make any changes to the source code. However, if you are using the OpenShift A-MQ broker, then modifications are required to access the A-MQ broker instance running within the OpenShift gear. You need to provide the appropriate connection details (URL, user ID, password).

12.1. Access a Simple Queue

Introduction

In this ActiveMQ exercise you create a producer and a consumer to access a simple queue. The producer writes messages to the queue, and the consumer reads messages.

The project has two parts:

- A consumer Java class creating a JMS MessageConsumer that reads messages as long as it can find one.
- A producer Java class instantiating a JMS MessageProducer that creates a specified number of JMS messages and then stops and exits. The sending of messages is artificially slowed with 100ms "sleep" between messages to enable you to interrupt them when desired.

Procedure

1. Do one of the following to start the message broker:
   - **For localhost:**
     a. If a JBoss A-MQ server is installed, check that it is running by issuing the command `./amq` in the `/bin` directory of the A-MQ distribution.
     b. Optionally, a broker project is provided in `messaging-labs-0.1` that can start a local message broker. In a Windows or UNIX terminal, in the `broker` directory, run `mvn -P broker`.
   - **For OpenShift:**
     a. Verify that the OpenShift gear being used to run the JBoss Fuse server is started.
     b. Open the project in JBoss Developer Studio.
     c. Open the `jndi.properties` file and verify that the A-MQ connection details in `jndi.properties` for the `java.naming.provider.url` are for the correct broker URL.

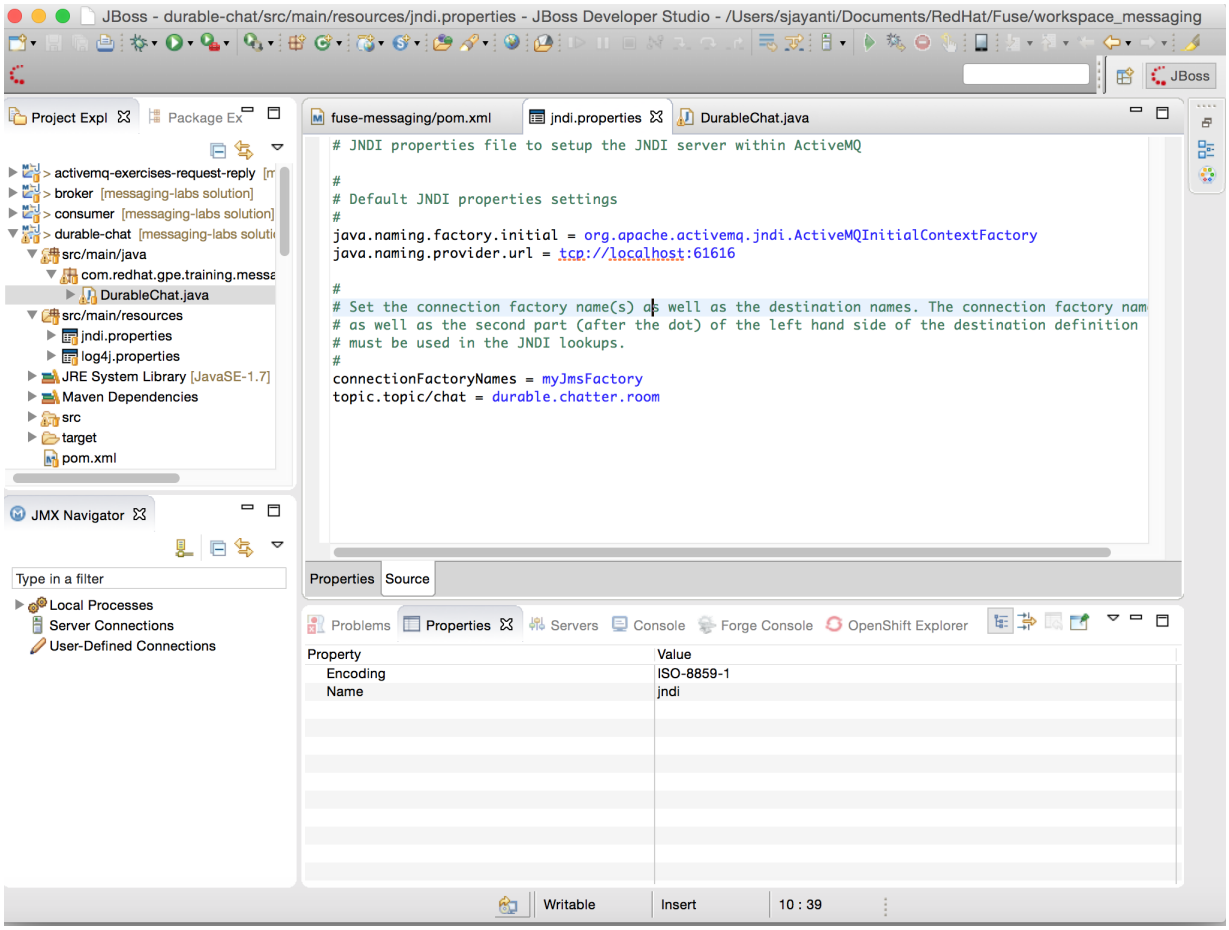

Figure 24. jndi.prperties file

d. Change the `java.naming.provider.url` to the URL provided in your OpenShift console. Example:

```
java.naming.provider.url=tcp://mqdemo-sjayanti.apps.ose.opentlc.com:60163
```

The URL you provide here is the same URL as the MQ broker profile and container.

e. Open `SimpleConsumer.java` and `SimpleProducer.java` in JBoss Developer Studio.

f. Provide the `admin` password for the OpenShift Fuse Console in the connection details of the corresponding Java files (`SimpleConsumer.java`, `SimpleProducer.java`).

The `admin` user is `admin`, and the password is the Management Console password for the OpenShift Enterprise container.

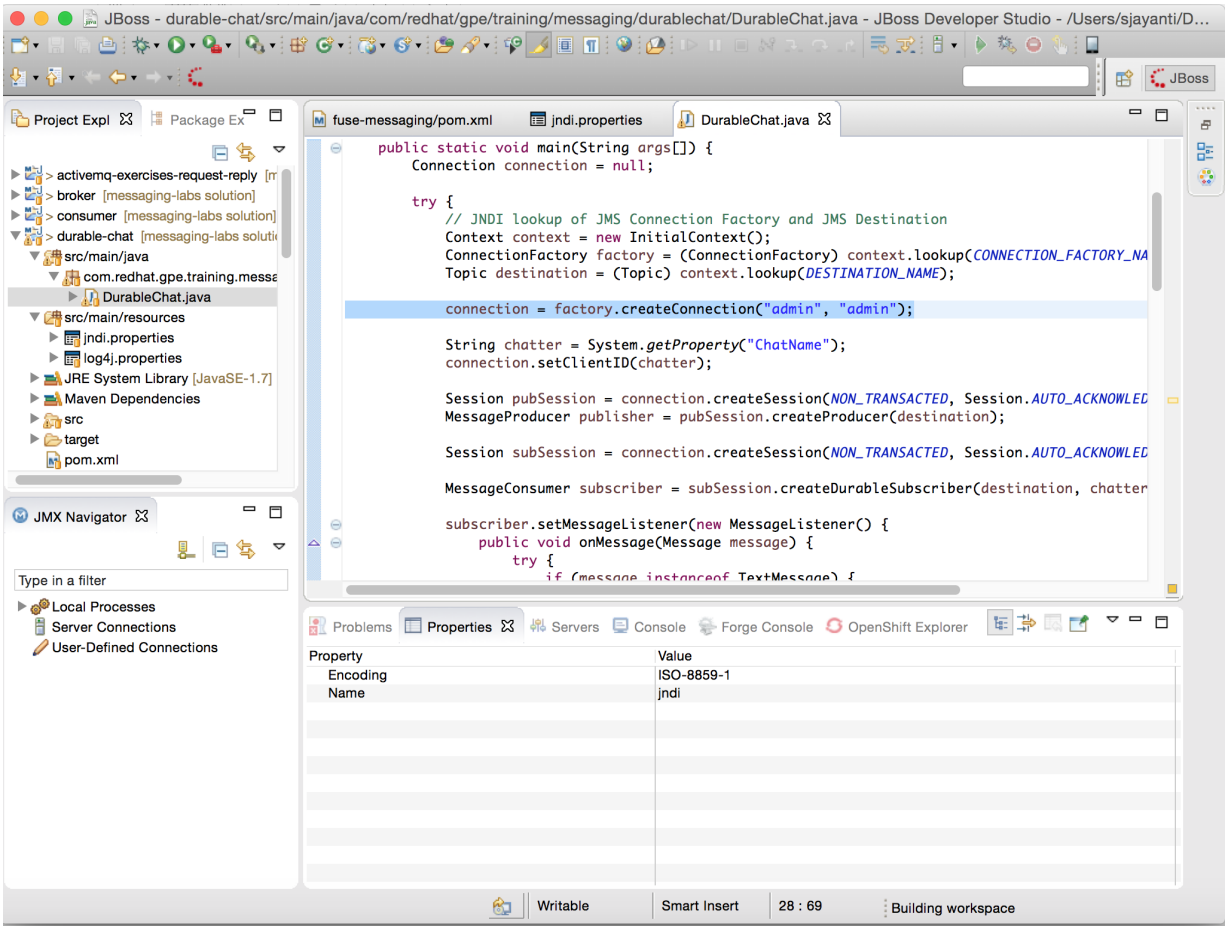

Figure 25. Connection details

g. Save all changes and compile the project.

2. Start the consumer:

a. Open another terminal window (Windows or UNIX) and navigate to `1_Introduction_to_Messaging/simple-queue`.

b. Run this Maven command:

```
mvn -P consumer
```

3. Start the producer:

a. Open another terminal window (Windows or UNIX) and navigate to `1_Introduction_to_Messaging/simple-queue`.

b. Run this Maven command:

```
mvn -P consumer
```

4. Check the Console and observe the following:

- The producer is producing 100 messages and sending them to `test.queue.simple`
- The same 100 messages are being consumed by the consumer from the queue

The consumer project is designed to continue listening for messages.

5. To exit the consumer project, press **Ctrl+C**.

12.2. Create a Publish-and-Subscribe Topology

Introduction

In this ActiveMQ exercise you create a publish-and-subscribe topology by using a topic in JBoss A-MQ.

The project has two parts:

- The subscriber project starts a JMS message listener that reads messages from `test.topic.simple` until a SHUTDOWN command message is sent. It also creates a JMS MessageConsumer that listens for control messages (report or shutdown messages) published into `topic.control`.
- The publisher project starts a JMS MessageProducer that creates the specified number of JMS messages in 10 message batches and then a single REPORT message. At the end it sends a SHUTDOWN message to indicate that the listener should exit. The sending of messages is artificially slowed with 100ms "sleep" between messages to enable you to interrupt them when desired.

Procedure

1. If it is not already running, start the message broker as described in the previous section.
2. Start the subscriber:
   a. Open a terminal window (Windows or UNIX) and navigate to the `1_Introduction_to_Messaging/simple-topic` directory.
   b. Run the following command:

```
mvn -P subscriber
```

3. Start the publisher:
   a. Open a terminal window and navigate to the `1_Introduction_to_Messaging/simple-topic`.
   b. Run the following command:

```
mvn -P subscriber
```

4. Check the Console and observe the following:
   - 10 batches of 10 messages are being produced by the publisher and sent to the topic `test.topic.simple`
   - The same messages are being consumed by the subscriber from the same topic.

The consumer project is designed to continue to listen for messages.

5. To exit, press **Ctrl+C**.