Table of Contents

Managing and Monitoring Lab

Goals

- Understand JBoss AM-Q management and monitoring
- Complete exercises on Java Management Extensions (JMX), security, and logging/auditing

Lab Assets

The lab exercises are available in the following zip archive:

- GitHub GPE MW Training : Messaging Labs Repository

1. Collect Information With JMX

Introduction

In this exercise you, create a Java application that uses JMX to collect information from the broker. The application uses the following attributes exposed by the JMX layer of the JBoss A-MQ broker:

- BrokerId
- MemoryPercentUsage
- EnqueueCount
- QueueSize

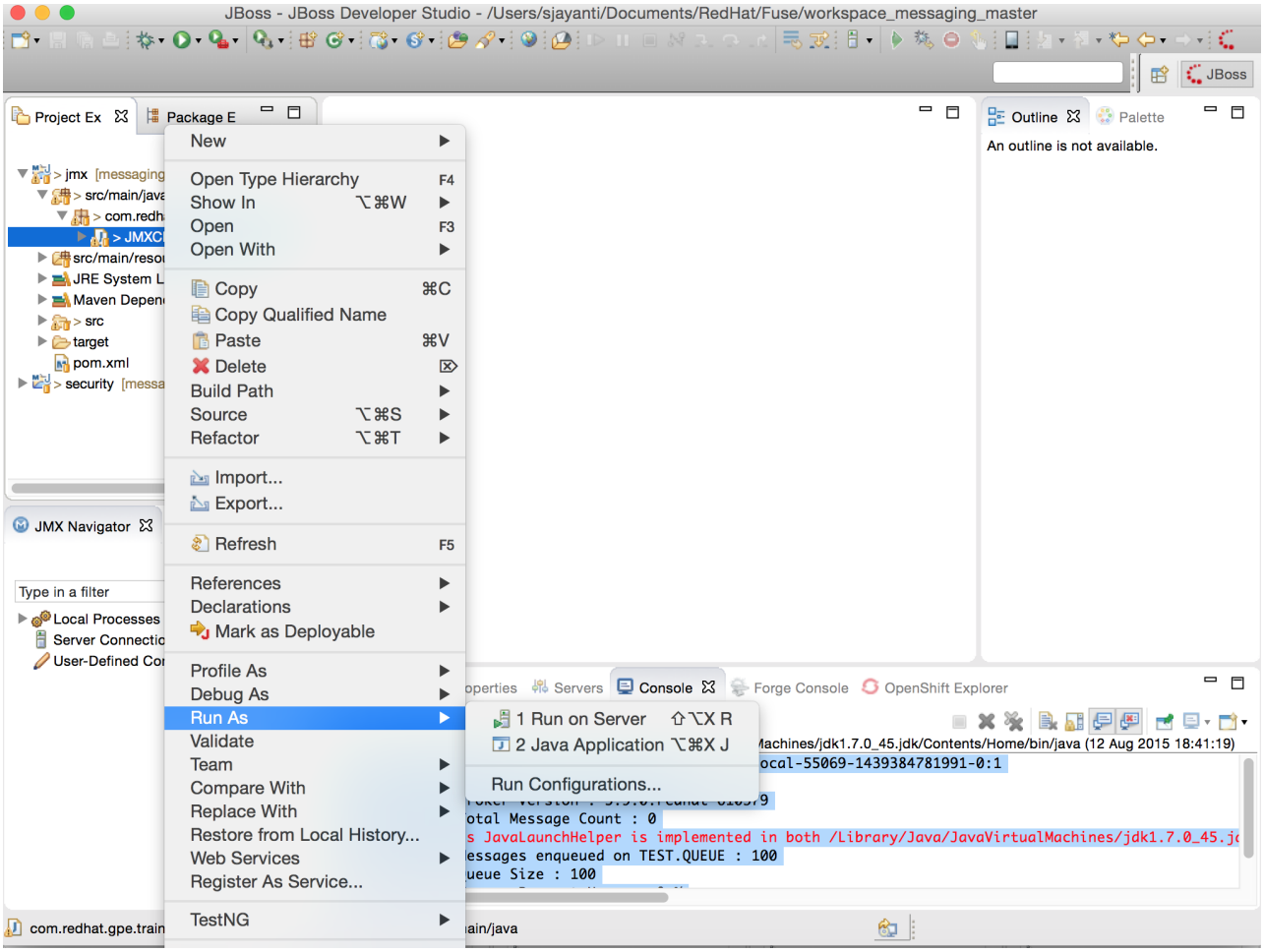
You also use the JBoss Fuse Management Console to navigate within the JMX tree of the JBoss A-MQ broker to discover information.

Procedure

1. If it is not already running, start the message broker as described in the Module 1 lab.

Do not start the broker using the Maven command `mvn -P broker`.

2. Open the `messaging-lab5-root/jmx` project in JBoss Developer Studio.
3. Run the Java application **JMXClient**.



4. In the Console, notice the following attribute values:

```
18:41:19 INFO  Broker ID : ID:Satyas-MacBook-Pro.local-55069-1439384781991-0:1
18:41:19 INFO  Broker Name : fuse-broker
18:41:19 INFO  Broker Version : 5.9.0.redhat-610379
18:41:19 INFO  Total Message Count : 0
18:41:20 INFO  Messages enqueued on TEST.QUEUE : 100
18:41:20 INFO  Queue Size : 100
18:41:20 INFO  Memory Percent Usage : 0 %
```

5. Open the Management Console in your web browser. For localhost, the URL is http://localhost:8181/hawtio/index.html.
6. Click the **Container** tab, and then click **JMX**.

By default the JMX option is enabled when the broker is created and started, so broker, queue, and topic statistics come from the Management Console, or by using `JConsole` on a JMX client.

7. Click through **org.apache.actimq > Broker > root > Queue** and **Topic** to view information about the broker.

8. Explore other sections such as **Health**, **Chart**, and **Attributes** for the **Queue** and **Topic** destinations.

2. Security

This lab exercise works only in a localhost environment. If you are using the OpenShift Environment, skip this exercise and go to the next exercise in this lab.

Introduction

In this ActiveMQ exercise, you create a secure broker by using encrypted credentials for authentication and groups for authorization.

- The producer uses the `user` role and the user ID `system` to write to `test.queue.security`.
- The consumer uses the `guest` role and the user ID `guest` to read from `test.queue.security`.

Procedure

1. Start the local broker using the security credentials:
   a. Open a command line terminal (Windows or UNIX) and navigate to `5_Managing_Monitoring/security`.
   b. Run this Maven command:

```
mvn -P broker
```

2. Start the consumer:
   a. Open another terminal window and navigate to `5_Managing_Monitoring/security`.
   b. Run this Maven command:

```
mvn -P consumer
```

3. Start the producer:
   a. Open another terminal window and navigate to `5_Managing_Monitoring/security`.
   b. Run this Maven command:

```
mvn -P producer
```

```
```

4. In the Console, observe the following:
   - 100 messages being produced by the producer and sent to `test.queue.security`
   - The same 100 messages are being consumed by the consumer from the same queue
5. Change the security authorization:
   a. In JBoss Developer Studio, open the `SimpleProducer.java` file and change the connection to use `guest` instead of `system`.

```
//connection = factory.createConnection("system", "manager");
connection = factory.createConnection("guest", "password");
```

   b. Go back to the terminal window and start the producer:

```
mvn -P producer
```

   c. Notice that the producer fails to connect due to the following error:

```
Caused by: java.lang.SecurityException: User guest is not authorized to write
to: queue://test.queue.security
```

   d. Change `SimpleProducer.java` back to `system` from `guest`:

```
connection = factory.createConnection("system", "manager");
//connection = factory.createConnection("guest", "password");
```

6. Change the security authentication:
   a. In JBoss Developer Studio, open the file `SimpleConsumer.java` and change the `guest` connection to use a different password:

```
connection = factory.createConnection("guest", "password");
```

   b. Go back to the terminal window and start the consumer:

```
mvn -P consumer
```

   c. Notice that the connection fails with this exception:

```
Caused by: java.lang.SecurityException: User name [guest] or password is
invalid.
```

3. Logging and Auditing

Introduction

In this exercise, you use logger APIs to log message exchange information for monitoring and auditing purposes.

Procedure

1. Do one of the following to enable logging:
   - If you are using the broker project on your local machine to start the broker, run the following command when you start the message broker:

     ```
     mvn -P broker -Dorg.apache.activemq.audit=tr
     ```

   - If you are using the JBoss AM-Q server locally, then open the file `$JBOSS_AMQ_HOME/bin/karaf` and modify it as follows:
     a. Search for the following line in the file:

     ```
     OPTS="-Dkaraf.startLocalConsole=true -Dkaraf.startRemoteShell=true”
     ```

     b. Append the following to the line and save the file:

     ```
     -Dorg.apache.activemq.audit=true
     ```

     c. By default the `org.ops4j.pax.logging.cfg` file in the `$JBOSS_AMQ_HOME/etc` directory does not contain the audit log appender. Add the following to the end of the file and save it:

     ```
     log4j.logger.org.apache.activemq.audit=INFO,audit
     log4j.logger.org.apache.activemq.broker.util.DefaultAuditLog=INFO,audit
     log4j.logger.org.apache.activemq.broker.util.LoggingBrokerPlugin=INFO,audit

     log4j.appender.audit=org.apache.log4j.RollingFileAppender
     log4j.appender.audit.file=${karaf.data}/log/audit.log
     log4j.appender.audit.maxFileSize=1024KB
     log4j.appender.audit.maxBackupIndex=5
     log4j.appender.audit.append=true
     log4j.appender.audit.layout=org.apache.log4j.PatternLayout
     log4j.appender.audit.layout.ConversionPattern=%-5p | %m | %t%n
     ```

     d. To start the JBoss A-MQ server, under the bin folder of the JBoss Fuse installation directory, run the command `./amq` or `amq.bat`.

- The access logs are written into the `$JBOSS_HOME/data/log/audit.log` file.

2. Open the Fuse Management Console and send some messages to `TEST.QUEUE.AUDIT`.
3. Open the `$JBOSS_HOME/data/log/audit.log` file and verify that audit messages were published:

```
INFO  | admin admin called
org.apache.activemq.broker.jmx.QueueView.sendTextMessage[, admin, admin] at 11-08-
2015 15:01:22,124 | qtp283007115-65
```

If you run the local broker using the Apache Maven command, then the audit message appears within the broker console. Messages can be submitted using `JConsole`.

Last updated 2015-10-27 21:13:22 EDT