



DEV 3500 – MapR Streams Lab Guide

Spring 2016 – Version 5.1.0

For use with the following courses:

DEV 3500 – Real-time Stream Processing with MapR
DEV 350 – MapR Streams Essentials
DEV 351 – Developing MapR Streams Applications

This Guide is protected under U.S. and international copyright laws, and is the exclusive property of MapR Technologies, Inc.

© 2016, MapR Technologies, Inc. All rights reserved. All other trademarks cited here are the property of their respective owners.



Using This Guide

Icons Used in the Guide

This lab guide uses the following icons to draw attention to different types of information:



Note: Additional information that will clarify something, provide details, or help you avoid mistakes.



CAUTION: Details you **must** read to avoid potentially serious problems.



Q&A: A question posed to the learner during a lab exercise.



Try This! Exercises you can complete after class (or during class if you finish a lab early) to strengthen learning.

Lab Files

Here is a description of the materials (data files, licenses etc.) that are supplied to support the labs. These materials should be downloaded to your system before beginning the labs.

DEV351_LabFiles.zip	Scripts, data, and other files necessary to complete this lab.
DEV351_LabGuide.pdf	Instructions for completing this lab (this document).
MapR_Lab_Environment_Connection_Guide.pdf	Instructions for connecting to your MapR lab environment.
Software_Setup_Guide.pdf	Instructions for installing your Java lab environment.

Lab 3: Developing MapR Streams Applications Using Kafka Java API

Lab Overview

The goal of this lab is to develop and run a MapR Streams producer and consumer sending and receiving messages. You will be using either the MapR Sandbox or a MapR cluster to run the lab exercises.

Exercise	Required	Duration
3.1 Import the “ms-lab” project	Yes	15 min
3.2 Finish code, build and run the “ms-lab” project	Yes	15 min

You should have already downloaded and installed an IDE as described in the Software Setup Guide.

In the ms-lab project, there is an exercise package with stubbed code for you to finish and there is a solution package with complete solutions. There are also other example packages.

Open and import the project into your IDE following the instructions below. Optionally, you can just edit the Java files and use Maven on the command line, or if you just want to use the prebuilt solution, you can copy the solution jar file from the target directory.

Here are some links for opening or importing a Maven project with your IDE:

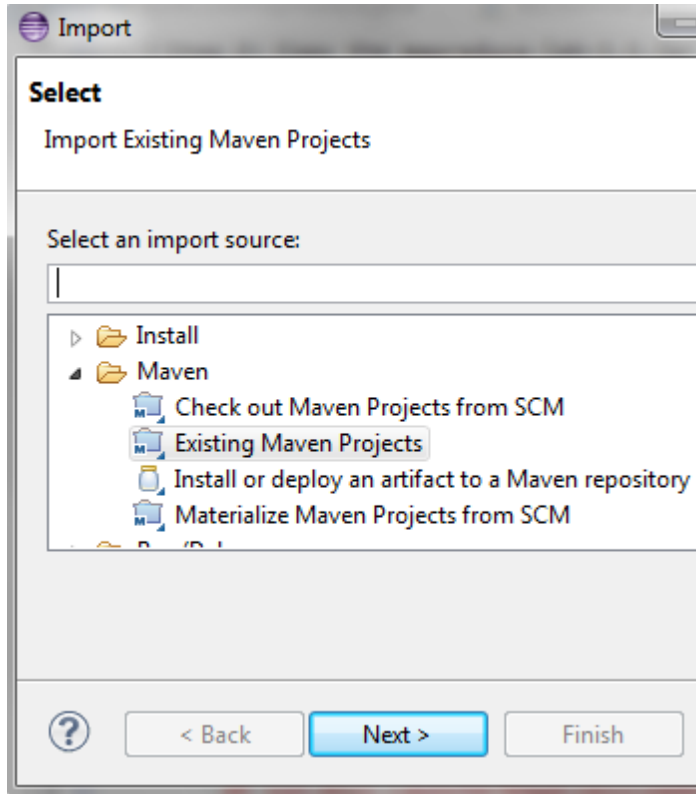
- Netbeans
 - http://wiki.netbeans.org/MavenBestPractices#Open_existing_project
- IntelliJ
 - http://www.tutorialspoint.com/maven/maven_intellij_idea.htm
- Eclipse
 - <http://scala-ide.org/docs/current-user-doc/gettingstarted/index.html>

Exercise 3.1: Import project

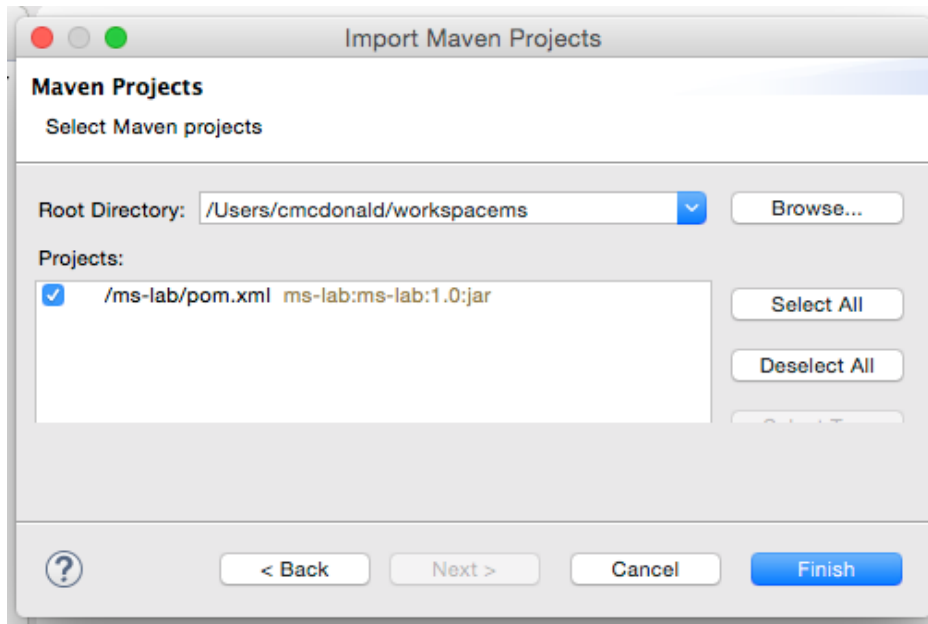
The purpose of this section is to import the lab project files into Eclipse. It should take approximately 5 minutes. Unzip the DEV351_LabFiles.zip file into a directory on your laptop.

1. Open Eclipse. Select Import from the File menu.

2. Expand the Maven Folder and select Existing Maven Projects.



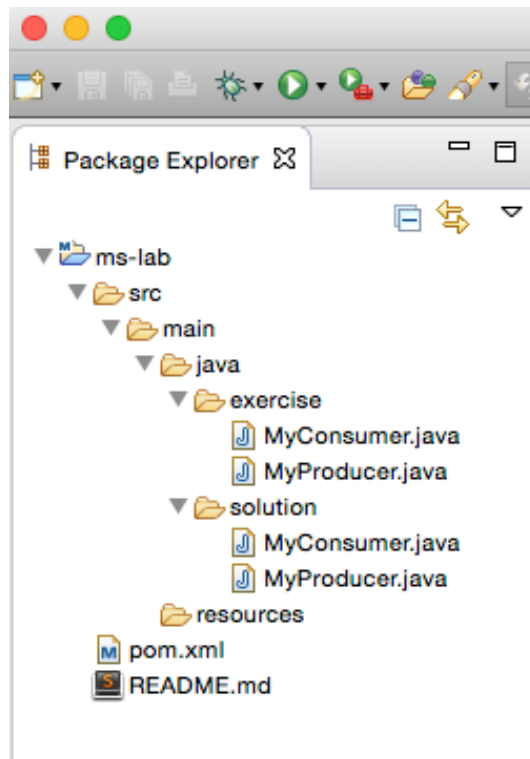
3. Browse to the directory where you unzipped the lab code file, as shown in the following image:



4. Select the pom.xml file. Select Finish.

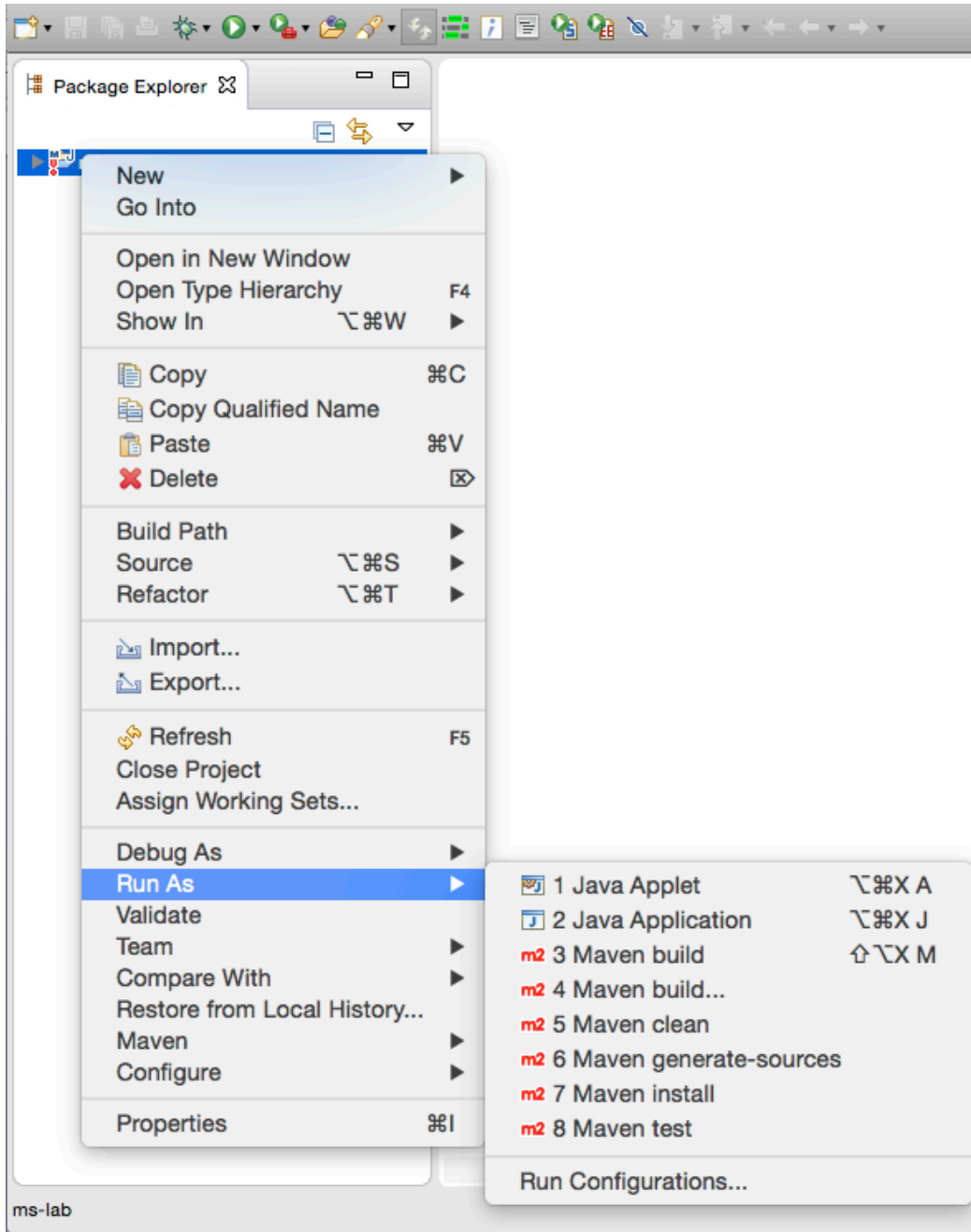


You should see the following:



Build a Maven project to create the jar file

1. Select the project from Run menu.
2. Select **Run As** and **Maven install** as shown in the following image.



A jar file will be created in the project target folder.



If you have problems building maven projects with Eclipse, you may need to delete your maven `$(user.home)/.m2 directory` . For more information see:

<http://maven.apache.org/guides/mini/guide-configuring-maven.html>

For reference, to build a MapR Streams Maven project:

1. Add MapR's Maven repository to your `pom.xml` file.

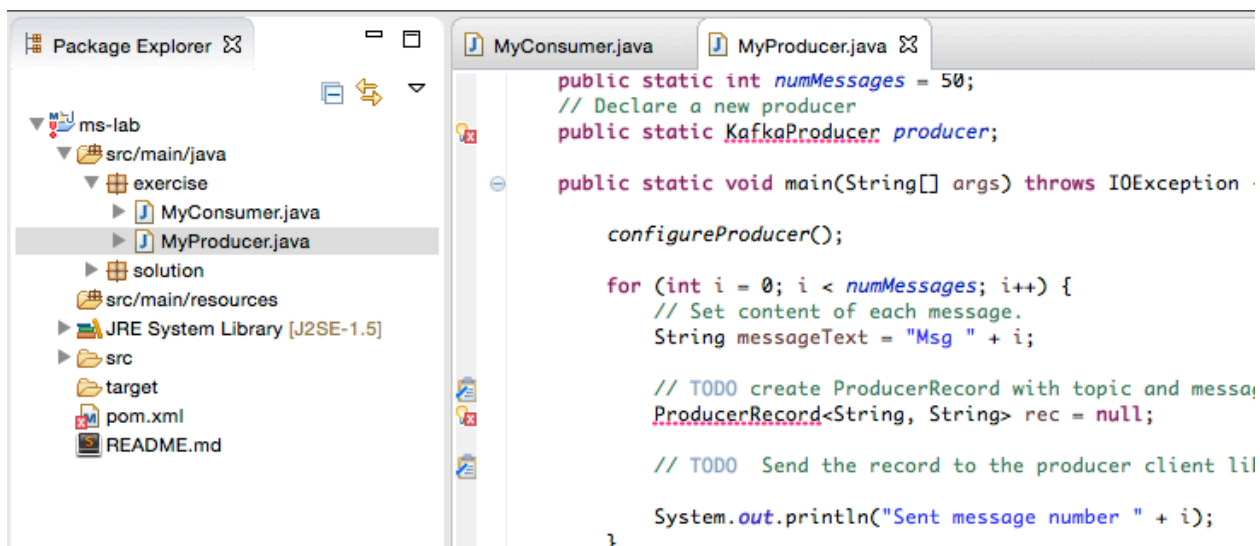
```
<repository>
  <id>mapr-releases</id>
  <url>
    http://repository.mapr.com/nexus/content/repositories/releases
  </url>
  <snapshots><enabled>true</enabled></snapshots>
  <releases><enabled>true</enabled></releases>
</repository>
```
2. Add a dependency to the kafka-clients project.

```
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka-clients</artifactId>
  <version>0.9.0.0-mapr-streams-5.1.0</version>
</dependency>
```

Exercise 3.2: Finish the code

Complete code for “TODO”

1. Look in the `src/main/java/exercise` directory. Open `MyProducer.java`.
2. Look for “TODO”s. Finish all of the code following the TODO instructions. If you need help, look in the solution directory for the finished code, or refer to the MapR Streams documentation http://maprdocs.mapr.com/51/#MapR_Streams/mapr_streams.html



1. Look in the `src/main/java/exercise` directory. Open `MyConsumer.java`.



2. Look for “TODO”s. Finish all of the code following the TODO instructions. If you need help, look in the solution directory for the finished code.

Run the code on a sandbox or cluster

Note: Substitute your user ID for user01.

1. Save any modified files.
2. Build the project using Maven: Select *project ms-lab -> Run As -> Maven Install*
3. Copy the jar file *ms-lab-1.0.jar* from the project target folder to your sandbox or cluster node, as instructed in the MapR Lab Environment Connection Guide document.

For example, if you are using the VirtualBox Sandbox, the command may look something like this, if you are in the ms-lab target folder where the jar file gets built:

```
$ scp -P 2222 ms-lab-1.0.jar user01@127.0.0.1:/user/user01/.
```

4. Log in to the sandbox or cluster, as instructed in the MapR Lab Environment Connection Guide document. For example:

```
$ ssh user01@<IP address> -p <port>
```

5. Create the stream:

```
maprcli stream create -path /user/user01/pump -produceperm u:user01 -  
consumeperm u:user01 -topicperm u:user01
```

```
maprcli stream topic create -path /user/user01/pump -topic input -  
partitions 3
```

```
maprcli stream topic create -path /user/user01/pump -topic alert -  
partitions 3
```

6. Get info on the stream:

```
maprcli stream info -path /user/user01/pump
```

7. Run the producer:

```
java -cp ms-lab-1.0.jar:`mapr classpath` exercise.MyProducer
```

8. Run the consumer:

```
java -cp ms-lab-1.0.jar:`mapr classpath` exercise.MyConsumer
```

9. Get info on the stream topic:

```
maprcli stream topic info -path /user/user01/pump -topic alert -json
```



Troubleshooting:

If you have problems running the producer or consumer, make sure that you have the correct name for the jar file, the Java class, and the package name. If you are running the solution, then change the package name so that the command looks like this:

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer
```

`mapr classpath` is a utility which sets the rest of the jars needed in the classpath.



Lab 4: Modify Properties on MapR Streams Producer and Consumer

Lab Overview

The goal of this lab is to run a consumer group. You will modify properties on the MapR Streams producer and consumer, which you developed in Lab 3, then you will run your application and observe the results.

Exercise	Required	Duration
4.1 run multiple producers and consumers in a group	Yes	10 min
4.2 modify producer and consumer properties	Yes	10 min

Exercise 4.1: Run Multiple Producers and Consumers

First, look at the solution producer and consumer code in the solutions folder. Then, run this code and observe the results.

1. Look at the code in the `src/main/java/solution` directory. Open `MyProducer2.java`.
2. Look in the code `src/main/java/solution` directory. Open `MyConsumer2.java`.
3. Use six terminal windows or PuTTY to log in to the sandbox or cluster, so that you can run and watch three producers and three consumers.

```
$ ssh user01@<IP address> -p <port>
```

4. In three terminal windows run the consumers:

```
java -cp `mapr classpath`:ms-lab-1.0.jar solution.MyConsumer2
```

The consumers are in a group, so each consumer reads from a partition.

```
public static void configureConsumer(String[] args) {
    Properties props = new Properties();
    // consumer group for cursor tracking and topic sharing
    props.put("group.id", "myteam");

    ...
}
```



1. In three terminal windows run the producer with a number and partition mode. The mode will be used to determine how to partition

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2 1 k
```

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2 2 k
```

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2 3 k
```

Remember there are three different ways of choosing a partition for a message:

- If the producer specifies a partition ID or if the StreamsPartitioner interface specifies one, the MapR Streams server publishes the message to the partition specified.
- If the producer does not specify a partition ID but provides a key, the MapR Streams server hashes the key and sends the message to the partition that corresponds to the hash.
- If neither a partition ID nor a key is specified, the MapR Streams server sends messages in a sticky round-robin fashion.

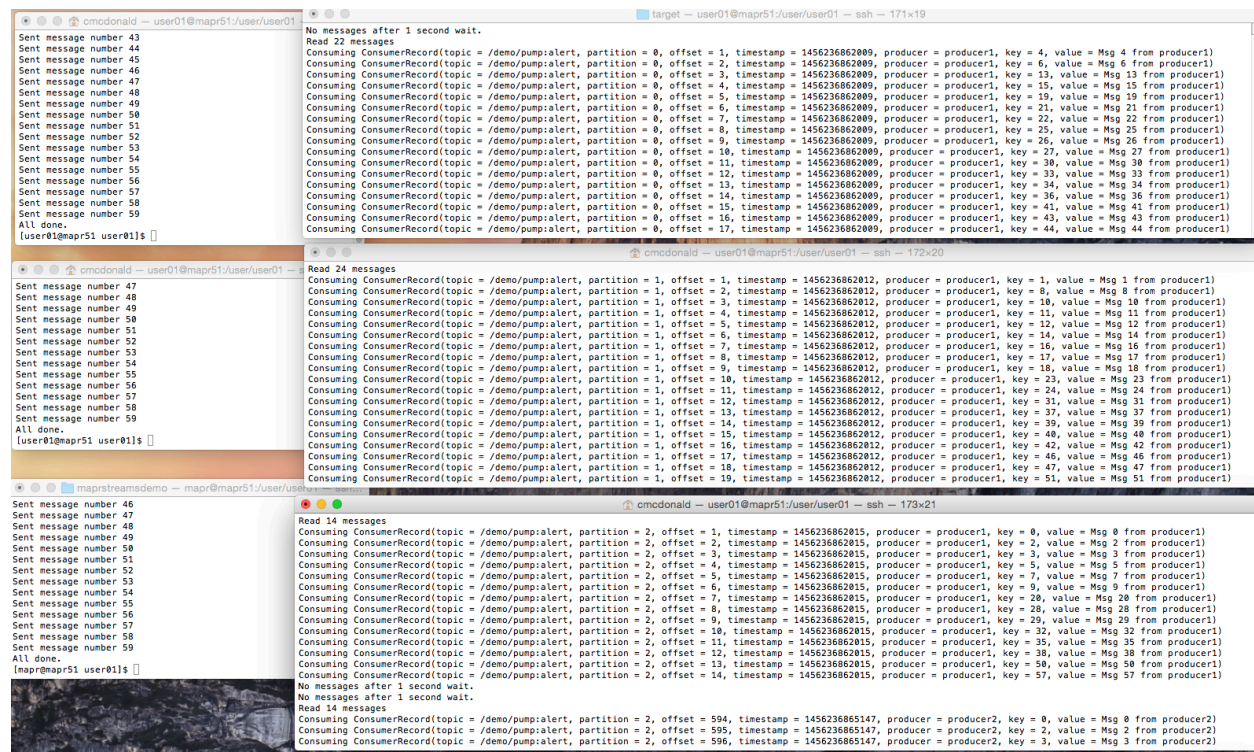
If the partition mode is p then the producer sends to the input partition, if the partition mode is k then the producer specifies for loop number as the key, this will be hashed by the streams server to determine the partition. If the partition mode is n, then the streams server sends the message in sticky round-robin.

```
for (int i = 0; i < numMessages; i++) {
    String messageText = "Msg " + i + " from " + id;
    String key = "" + i;
    switch (partitionmode) {
        case "p":
            // send to input partition
            rec = new ProducerRecord<>(topic, partition, key, messageText);
            break;
        case "k":
            // send to key hashed partition
            rec = new ProducerRecord<>(topic, key, messageText);
            break;
        case "n":
        default:
            rec = new ProducerRecord<>(topic, messageText);
    }
    ...
}
```

5. Use Ctrl+C to stop one consumer when the producers have stopped sending. Run a producer again, notice that the partition of the stopped consumer is assigned to another one in the group



Observe the results



```

cmcdonald ~ user01@mapr51/user/user01
No messages after 1 second wait.
Read 22 messages
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 1, timestamp = 1456236862009, producer = producer1, key = 4, value = Msg 4 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 2, timestamp = 1456236862009, producer = producer1, key = 6, value = Msg 6 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 3, timestamp = 1456236862009, producer = producer1, key = 13, value = Msg 13 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 4, timestamp = 1456236862009, producer = producer1, key = 15, value = Msg 15 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 5, timestamp = 1456236862009, producer = producer1, key = 19, value = Msg 19 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 6, timestamp = 1456236862009, producer = producer1, key = 21, value = Msg 21 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 7, timestamp = 1456236862009, producer = producer1, key = 22, value = Msg 22 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 8, timestamp = 1456236862009, producer = producer1, key = 25, value = Msg 25 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 9, timestamp = 1456236862009, producer = producer1, key = 26, value = Msg 26 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 10, timestamp = 1456236862009, producer = producer1, key = 27, value = Msg 27 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 11, timestamp = 1456236862009, producer = producer1, key = 30, value = Msg 30 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 12, timestamp = 1456236862009, producer = producer1, key = 33, value = Msg 33 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 13, timestamp = 1456236862009, producer = producer1, key = 34, value = Msg 34 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 14, timestamp = 1456236862009, producer = producer1, key = 36, value = Msg 36 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 15, timestamp = 1456236862009, producer = producer1, key = 41, value = Msg 41 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 16, timestamp = 1456236862009, producer = producer1, key = 43, value = Msg 43 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 0, offset = 17, timestamp = 1456236862009, producer = producer1, key = 44, value = Msg 44 from producer1)
All done.
[user01@mapr51 user01]$

cmcdonald ~ user01@mapr51/user/user01
Read 24 messages
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 1, timestamp = 1456236862012, producer = producer1, key = 1, value = Msg 1 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 2, timestamp = 1456236862012, producer = producer1, key = 8, value = Msg 8 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 3, timestamp = 1456236862012, producer = producer1, key = 10, value = Msg 10 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 4, timestamp = 1456236862012, producer = producer1, key = 11, value = Msg 11 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 5, timestamp = 1456236862012, producer = producer1, key = 12, value = Msg 12 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 6, timestamp = 1456236862012, producer = producer1, key = 14, value = Msg 14 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 7, timestamp = 1456236862012, producer = producer1, key = 16, value = Msg 16 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 8, timestamp = 1456236862012, producer = producer1, key = 17, value = Msg 17 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 9, timestamp = 1456236862012, producer = producer1, key = 18, value = Msg 18 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 10, timestamp = 1456236862012, producer = producer1, key = 23, value = Msg 23 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 11, timestamp = 1456236862012, producer = producer1, key = 24, value = Msg 24 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 12, timestamp = 1456236862012, producer = producer1, key = 31, value = Msg 31 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 13, timestamp = 1456236862012, producer = producer1, key = 34, value = Msg 34 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 14, timestamp = 1456236862012, producer = producer1, key = 39, value = Msg 39 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 15, timestamp = 1456236862012, producer = producer1, key = 40, value = Msg 40 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 16, timestamp = 1456236862012, producer = producer1, key = 42, value = Msg 42 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 17, timestamp = 1456236862012, producer = producer1, key = 46, value = Msg 46 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 18, timestamp = 1456236862012, producer = producer1, key = 47, value = Msg 47 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 1, offset = 19, timestamp = 1456236862012, producer = producer1, key = 51, value = Msg 51 from producer1)
All done.
[user01@mapr51 user01]$

mapr51demo ~ mapr@mapr51/user/user01
Read 14 messages
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 1, timestamp = 1456236862015, producer = producer1, key = 0, value = Msg 0 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 2, timestamp = 1456236862015, producer = producer1, key = 2, value = Msg 2 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 3, timestamp = 1456236862015, producer = producer1, key = 3, value = Msg 3 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 4, timestamp = 1456236862015, producer = producer1, key = 5, value = Msg 5 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 5, timestamp = 1456236862015, producer = producer1, key = 7, value = Msg 7 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 6, timestamp = 1456236862015, producer = producer1, key = 9, value = Msg 9 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 7, timestamp = 1456236862015, producer = producer1, key = 20, value = Msg 20 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 8, timestamp = 1456236862015, producer = producer1, key = 28, value = Msg 28 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 9, timestamp = 1456236862015, producer = producer1, key = 29, value = Msg 29 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 10, timestamp = 1456236862015, producer = producer1, key = 32, value = Msg 32 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 11, timestamp = 1456236862015, producer = producer1, key = 35, value = Msg 35 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 12, timestamp = 1456236862015, producer = producer1, key = 38, value = Msg 38 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 13, timestamp = 1456236862015, producer = producer1, key = 50, value = Msg 50 from producer1)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 14, timestamp = 1456236862015, producer = producer1, key = 57, value = Msg 57 from producer1)
No messages after 1 second wait.
Read 14 messages
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 594, timestamp = 1456236865147, producer = producer2, key = 0, value = Msg 0 from producer2)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 595, timestamp = 1456236865147, producer = producer2, key = 2, value = Msg 2 from producer2)
Consuming ConsumerRecord(topic = /demo/pump/alert, partition = 2, offset = 596, timestamp = 1456236865147, producer = producer2, key = 3, value = Msg 3 from producer2)

```

1. Get Info on the Stream Topic:

```
maprcli stream topic info -path /user/user01/pump -topic alert -json
```

With the partition mode k, the producer specifies the for loop number (0-59) as the key, this will be hashed by the MapR Streams server to determine the partition. The hash sends the following keys to the three partitions: (the distribution shows that this is not the best key to hash by)

partition 0 key 4,6,13,15,19,21,22,25,26,27,30,33,34,36,41,43,44,45,48,49,53,59

partition 1 key 1,8,10,11,12,14,16,17,18,23,24,31,37,39,40,42,46,47,51,52,54,55,56,58

partition 2 key 2,3,5,7,9,20,28,29,32,35,38,50,57

The consumer reading from partition 0 will read all of the messages whose key hashed to partition 0, the output looks like this (output is shortened):

```

Read 22 messages
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 1, timestamp = 1456236862009, producer = producer1, key = 4, value = Msg 4 from producer1)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 2, timestamp = 1456236862009, producer = producer1, key = 6, value = Msg 6 from producer1)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 3, timestamp = 1456236862009, producer = producer1, key = 13, value = Msg 13 from producer1)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 4, timestamp = 1456236862009, producer = producer1, key = 15, value = Msg 15 from producer1)
...
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 930, timestamp = 1456236865144, producer =

```



```

producer2, key = 4, value = Msg 4 from producer2)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 931, timestamp = 1456236865144, producer =
producer2, key = 6, value = Msg 6 from producer2)
...
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 1859, timestamp = 1456236865828, producer =
producer3, key = 4, value = Msg 4 from producer3)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 1860, timestamp = 1456236865828, producer =
producer3, key = 6, value = Msg 6 from producer3)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 1861, timestamp = 1456236865828, producer =
producer3, key = 13, value = Msg 13 from producer3)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 1862, timestamp = 1456236865828, producer =
producer3, key = 15, value = Msg 15 from producer3)
Consuming ConsumerRecord(topic = /user/user01/pump:alert, partition = 0, offset = 1863, timestamp = 1456236865828, producer =
producer3, key = 19, value = Msg 19 from producer3)

```

Restart any of the consumers that you stopped. Modify the producer parameters to send to an input partition, and observe the results:

1. In three terminal windows run the Producer with a number and partition mode. The mode will be used to determine how to partition

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2 0 p
```

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2 1 p
```

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2 2 p
```

2. Use Ctrl+C to stop one consumer when the producers have stopped sending. Run a producer again. Notice that the partition of the stopped consumer is assigned to another one in the group. Restart the stopped consumer, and send some more messages to this partition. Notice the consumer will start reading again from this partition.
3. In three terminal windows, run the producer without a number and partition mode. The MapR Streams server will determine the partition.

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2
```

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2
```

```
java -cp ms-lab-1.0.jar:`mapr classpath` solution.MyProducer2
```

4. Use Ctrl+C again to stop one consumer when the producers have stopped sending. Run a producer again. Notice that the partition of the stopped consumer is assigned to another one in the group. Restart the stopped consumer, and send some more messages to this partition. Notice the consumer will start reading again from this partition.



Exercise 4.1: Modify Producer and Consumer properties

First, look at some of the solution producer and consumer code in the solutions folder.

Now you are ready to modify the code you developed in Lab 3:

- Look in the examples src/main/java/solution directory.
- Look in the src/main/java/exercise directory. Open MyProducer.java.
- Look in the configureProducer() method.
- Try out some of the properties discussed in the lecture (see the Configuration Parameters documentation at the end of this document).
- Look at the examples in the src/main/java/solution directory.
- Look in the src/main/java/exercise directory. Open MyConsumer.java.
- Look in the configureProducer() method.
- Try out some of the properties discussed in the lecture

Build the code, copy the jar file to the sandbox or cluster, run the code as described in Lab 3.

For more information about the producer and consumer properties, refer to the MapR Streams documentation:

- http://maprdocs.mapr.com/51/#MapR_Streams/configuration_parameters_for_producers.html
- http://maprdocs.mapr.com/51/#MapR_Streams/configuration_parameters_for_consumers.html

Bonus Exercises: Run some Examples

1. In one terminal window run a consumer:

```
$ java -cp ms-lab-1.0.jar:`mapr classpath` example.partition.Consumer  
/user/user01/pump:input
```

This consumer is in a group and will get all three partitions assigned, as shown in the output:

```
Started listening to [/user/user01/pump:input-0, /user/user01/pump:input-1,  
/user/user01/pump:input-2]
```

2. In a terminal window run a producer:

```
$ java -cp ms-lab-1.0.jar:`mapr classpath` example.partition.Producer  
/user/user01/pump:input
```



3. Type text at the command line for the producer and hit return. A message will be sent for each line, to a partition determined by the configured DemoPartitioner class.
4. In another terminal window, start another consumer. Notice how the partitions are re-assigned in the group. (In the terminal windows you should see something like: Started listening to [/user/user01/pump:input-2] Stopped listening to [/user/user01/pump:input-2])
5. Type more text in the producer window to send more messages.
6. Start another consumer. Stop and start a consumer again. Notice how the partitions are reassigned in the group
7. Take a look at some of the other code examples and trying running them and/or modifying your own code.



Reference: Configuration Parameters

MapR Streams Documentation

http://maprdocs.mapr.com/51/#MapR_Streams/mapr_streams.html

Configuration Parameters

http://maprdocs.mapr.com/51/#MapR_Streams/configuration_parameters_for_producers.html

http://maprdocs.mapr.com/51/#MapR_Streams/configuration_parameters_for_consumers.html

