Computer Science & Information Systems

**Real Time Analytics / Stream Processing & Analytics**

**Kafka Lab Sheet 5**

**Kafka Streams Application**

1. Objective:

Students should be able to

1. Get familiarity with the working of Kafka Streams
2. Get hands-on experience writing Java program for Streams processing using Kafka Streams

Kafka Streams is a client library for building mission-critical real-time applications and microservices, where the input and/or output data is stored in Kafka clusters. Kafka Streams combines the simplicity of writing and deploying standard Java and Scala applications on the client side with the benefits of Kafka's server-side cluster technology to make these applications highly scalable, elastic, fault-tolerant, distributed, and much more. This lab example will demonstrate how to run a streaming application coded in this library.

It implements the WordCount algorithm, which computes a word occurrence histogram from the input text. However, unlike other WordCount examples you might have seen before that operate on bounded data, the WordCount demo application behaves slightly differently because it is designed to operate on an infinite, unbounded stream of data. Similar to the bounded variant, it is a stateful algorithm that tracks and updates the counts of words. However, since it must assume potentially unbounded input data, it will periodically output its current state and results while continuing to process more data because it cannot know when it has processed "all" the input data.

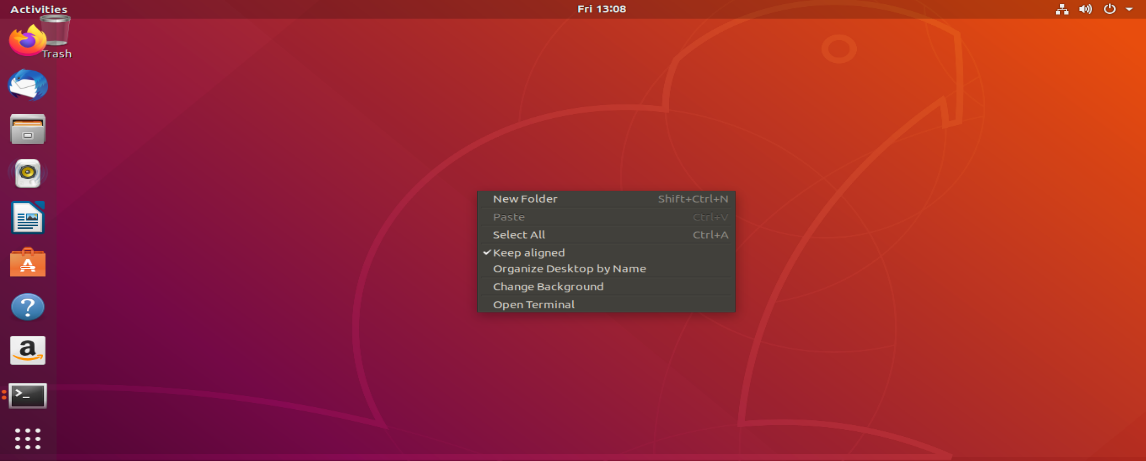
This lab sheet will introduce students with usage of Kafka Streams API with Java.

1. Steps to be performed:

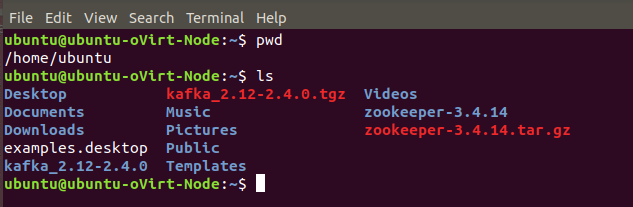
Note - It’s assumed that student has made a slot reservation using the slot booking interface where Apache Spark framework was selected. The details of the Apache Spark systems to be used is received through an email. If not, please contact the administrators for the same.

Also it’s assumed that students are aware of the process of logging into these virtual machines. If not, then get access to the user manual maintained for the usage of remote lab setup.

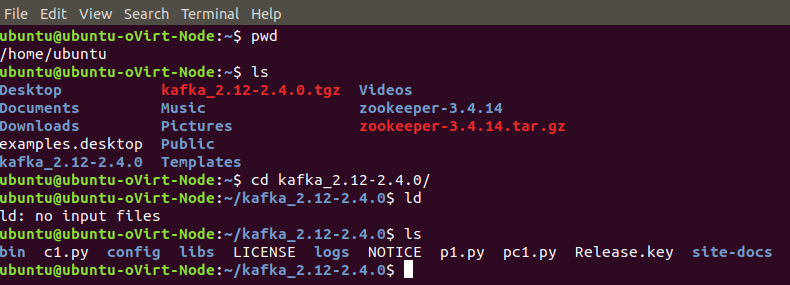
1. Open the terminal by right clicking on the desktop of the virtual machine.



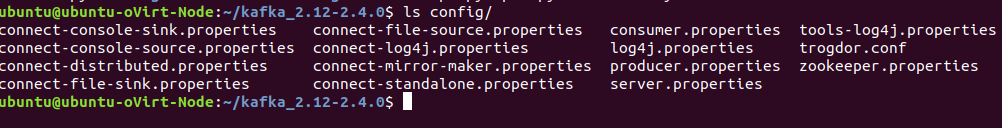
1. Look at the current directory and also file listings in it. It must have a Kafka installation directory present in it. Commands like pwd, ls can be used for it.

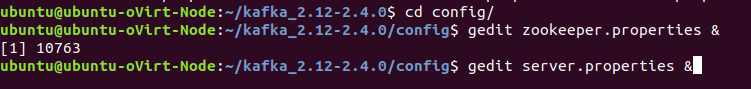


1. Change to the Kafka installation directory using the command and have a look at the files present in the directory.



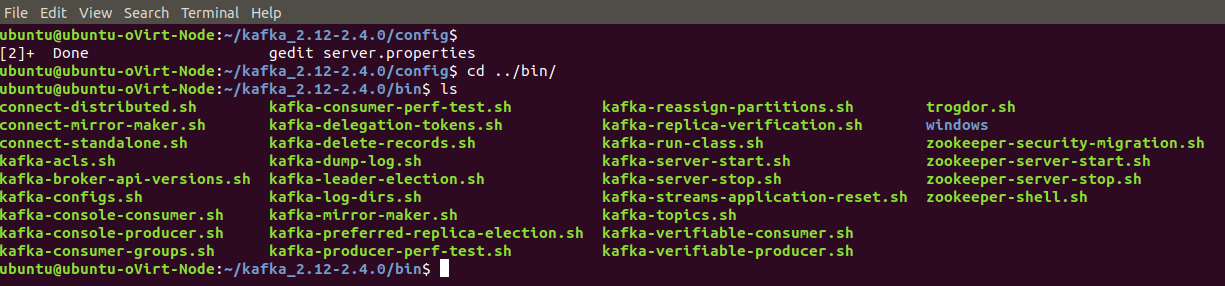
1. Change to the “config” directory present in the Kafka installation directory and have a look at the zookeeper and Kafka server properties files in it.

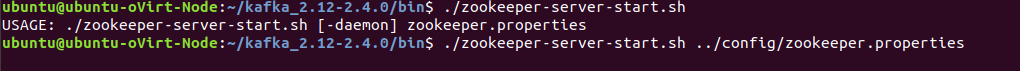




1. Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following scripts to start and stop the Zookeeper ensemble

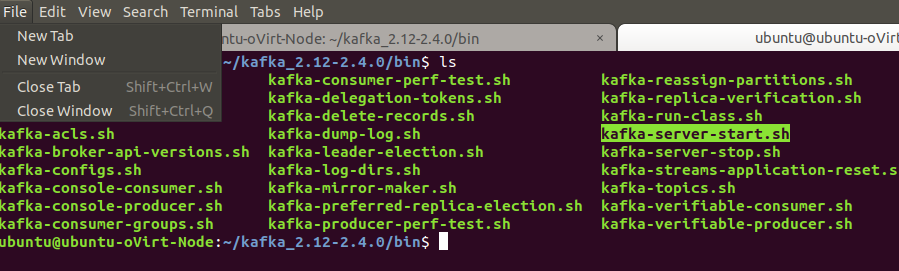
* zookeeper-server-start.sh
* zookeeper-server-stop.sh

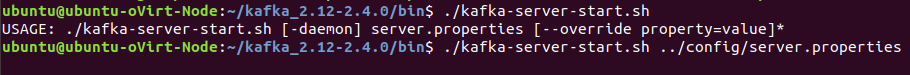




1. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following scripts to start and stop the Kafka server

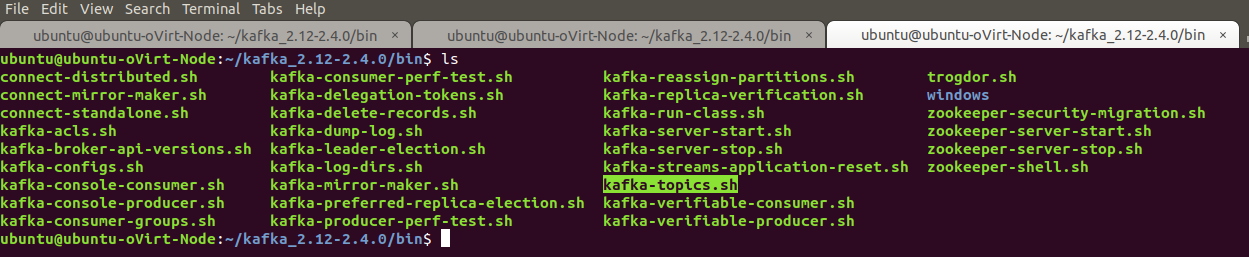
* kafka-server-start.sh
* kafka-server-stop.sh





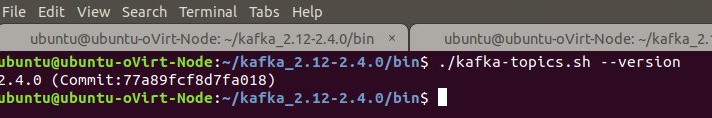
1. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following script to deal with the Kafka topics.

* kafka-topics.sh



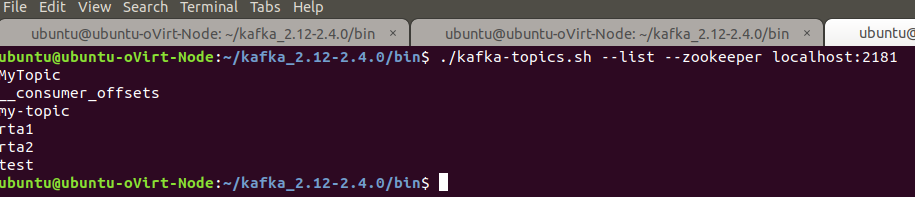
1. Use the –version option to see the version of Kafka.

* ./kafka-topics.sh –version



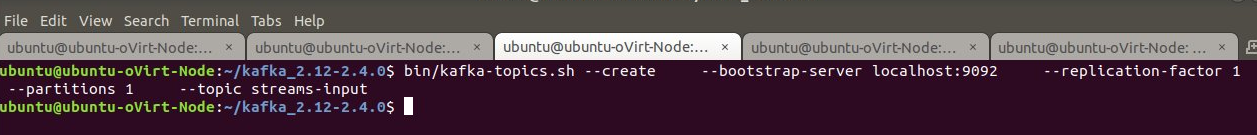
1. Use the –list option in it to list the topics present in the Kafka setup. Note – it will be empty for you as you don’t have created any topics as such but if the Kafka setup is shared you may see the topics created in earlier usages of the system.

* ./kafka-topics.sh --list --zookeeper localhost:2181

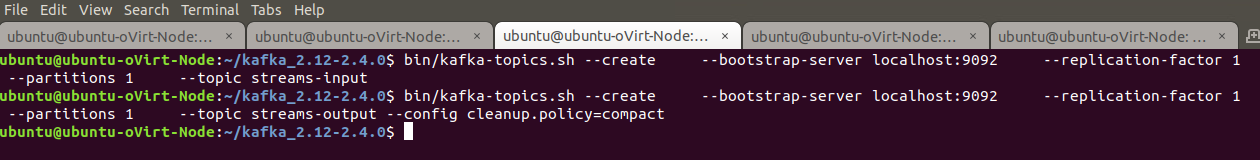


1. Let’s try to create a Kafka topic named “streams-input” using the –create option.

* ./kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic streams-input

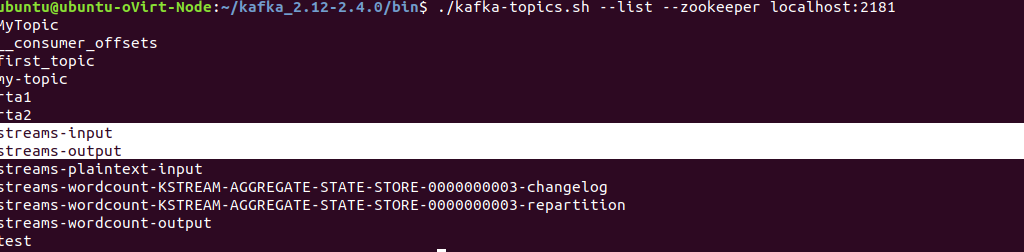


1. Let’s try to create a Kafka topic named “streams-output” using the –create option.

* . /kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic streams-output --config cleanup.policy=compact 

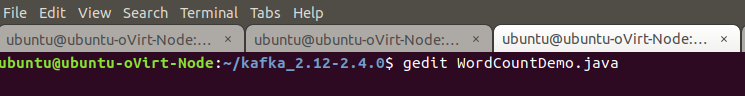
1. List the Kafka topics again. Now the created topics should appear in the topics list.

* ./kafka-topics.sh --list --zookeeper localhost:2181

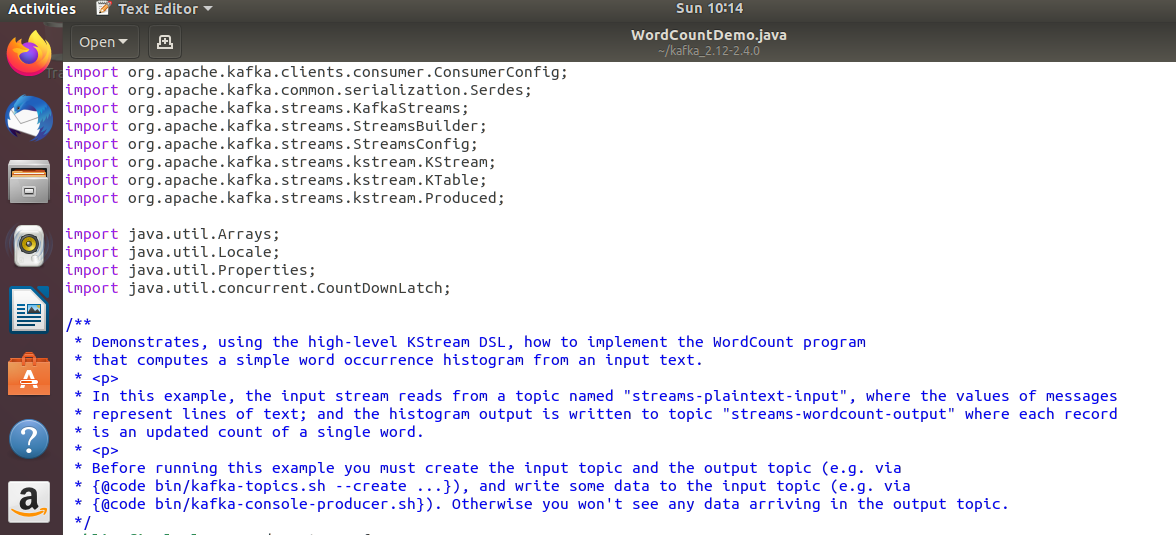


Now let’s see how we can write a Java program demoing the usage of Streams API.

1. Create a Java file named WordCountDemo.java.

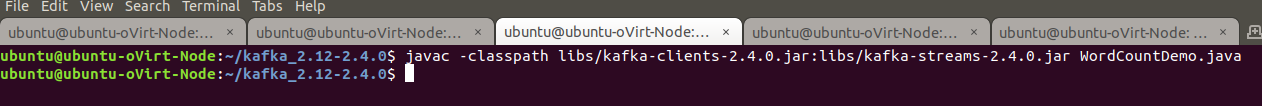


Copy paste the Java code from the attached WordCountDemo.java file in it.



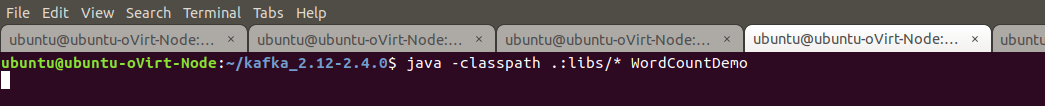
1. Compile the Java code using the following command

* javac -classpath libs/kafka-clients-2.4.0.jar:libs/kafka-streams-2.4.0.jar WordCountDemo.java



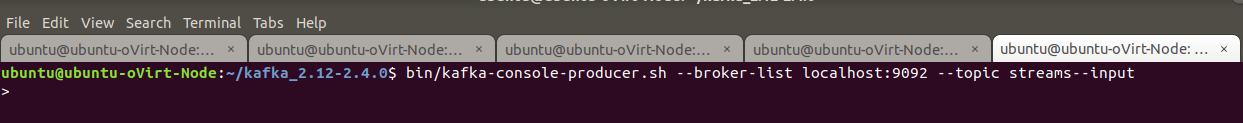
1. Run the WordCountDemo application using the java command.

* java -classpath .:libs/\* WordCountDemo

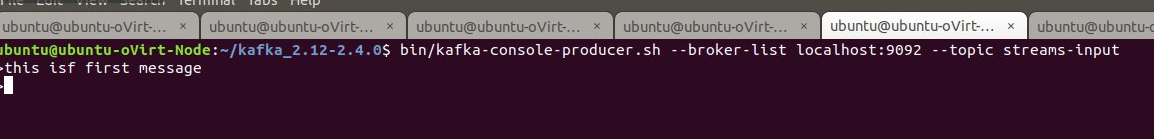


1. The Kafka distribution provides a command utility to send messages from the command line. It start up a terminal window where everything you type is sent to the Kafka topic. Kafka provides the utility kafka-console-producer.sh to send messages to a topic on the command line.

* bin/kafka-console-producer.sh --broker-list localhost:9092 --topic streams-input



1. Try inserting some message in the command prompt provided by producer utility.



1. Open up another terminal and Change to the Kafka installation directory and have a look at the script files present in it.
2. Inspect the output of the WordCount demo application by reading from its output topic with the console consumer in a separate terminal.

* bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 \

--topic streams -output \

--from-beginning \

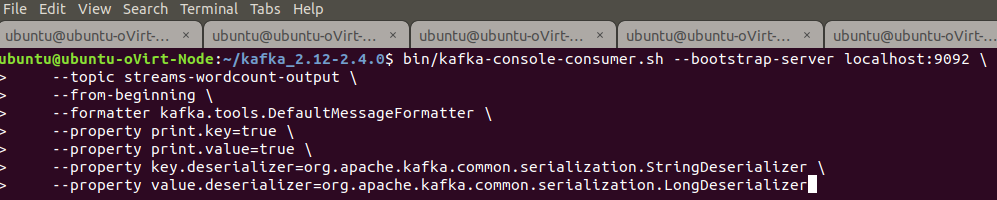
--formatter kafka.tools.DefaultMessageFormatter \

--property print.key=true \

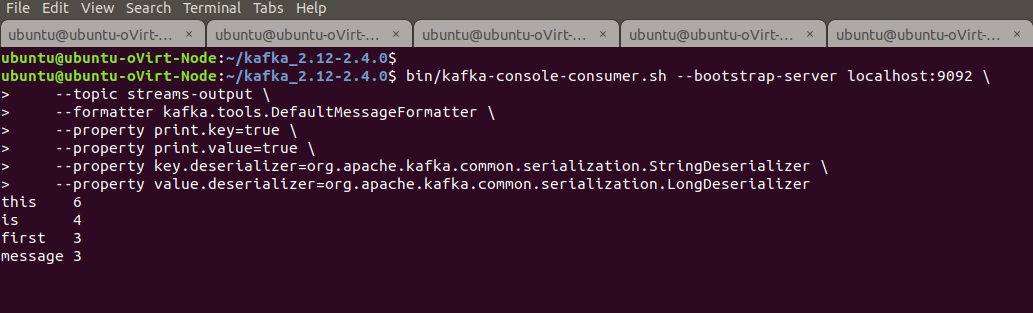
--property print.value=true \

--property key.deserializer=org.apache.kafka.common.serialization.StringDeserializer \

--property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer

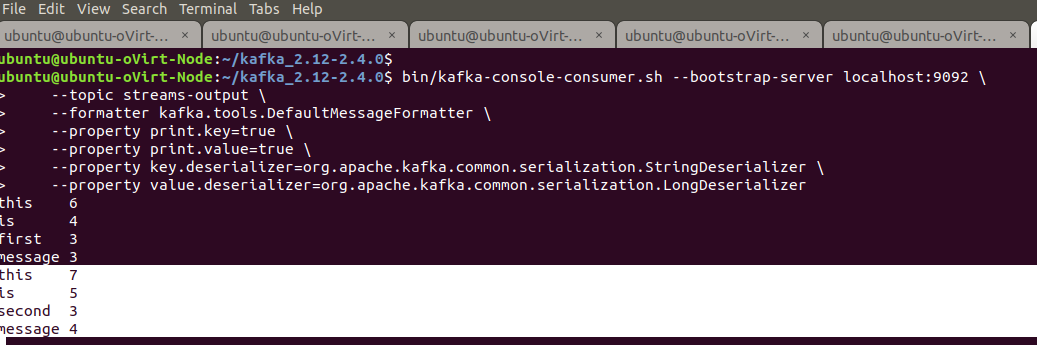


1. Execute the producer utility again and send some messages in it. Now the consumer tool should now show the word count for the words present in the messages.

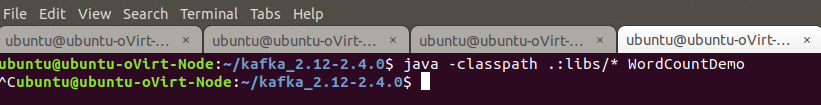


Note – You should see word count as 1 for each word. The above screenshot does shows different number as it is taken after inserting many similar messages.

1. Try repeating the insertion of the messages from the producer tool and see the outcomes produced in the consumer tool.

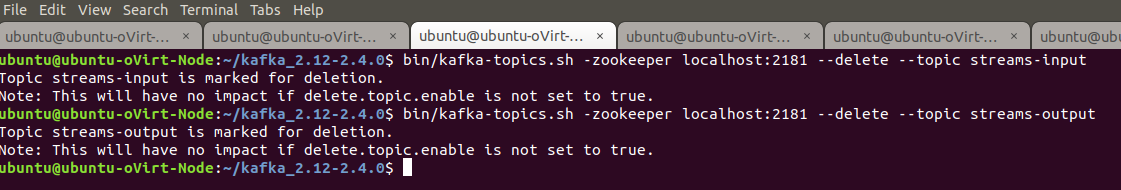


1. This application will run until you go and explicitly kill the java command used for launching the streaming application.



1. Now let’s try to delete the topic. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following script to delete with the Kafka topics.

* bin/kafka-topics.sh -zookeeper localhost:2181 --delete --topic streams-input
* bin/kafka-topics.sh -zookeeper localhost:2181 --delete --topic streams-output



1. Outputs/Results:

Students should be able to write a Kafka Streaming application

* To read the messages from the input Kafka topic
* To do some basic processing on the messages those are read
* To write the processed messages back to the output Kafka topic

1. Observations:

Students carefully needs to observe the code written for the usage of Streams API for

* Specifying the Kafka topic configurations
* Writing the word count logic
* Building the data processing pipeline

1. References:
2. [Kafka Quickstart](https://kafka.apache.org/quickstart)
3. [Kafka website](https://kafka.apache.org/)
4. [Kafka Streams](https://kafka.apache.org/24/documentation/streams/)