

Computer Science & Information Systems

Real Time Analytics / Stream Processing & Analytics

Kafka Lab Sheet 5

Kafka Streams Application

1. Objective:

Students should be able to

- A. Get familiarity with the working of Kafka Streams
- B. 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.

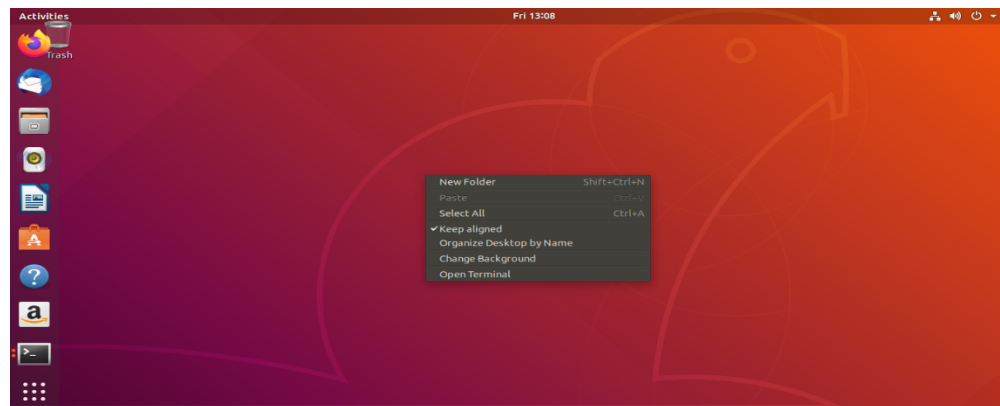
This lab sheet will introduce students with usage of Kafka Streams API with Java.

2. 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.

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



- B. 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.

```
File Edit View Search Terminal Help
ubuntu@ubuntu-oVirt-Node:~$ pwd
/home/ubuntu
ubuntu@ubuntu-oVirt-Node:~$ ls
Desktop      kafka_2.12-2.4.0.tgz  Videos
Documents    Music                  zookeeper-3.4.14
Downloads    Pictures               zookeeper-3.4.14.tar.gz
examples.desktop  Public
kafka_2.12-2.4.0  Templates
ubuntu@ubuntu-oVirt-Node:~$
```

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

```
File Edit View Search Terminal Help
ubuntu@ubuntu-oVirt-Node:~$ pwd
/home/ubuntu
ubuntu@ubuntu-oVirt-Node:~$ ls
Desktop      kafka_2.12-2.4.0.tgz  Videos
Documents    Music                  zookeeper-3.4.14
Downloads    Pictures               zookeeper-3.4.14.tar.gz
examples.desktop  Public
kafka_2.12-2.4.0  Templates
ubuntu@ubuntu-oVirt-Node:~$ cd kafka_2.12-2.4.0/
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ ls
ld: no input files
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ ls
bin  c1.py  config  libs  LICENSE  logs  NOTICE  p1.py  pc1.py  Release.key  site-docs
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$
```

- D. 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.

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ ls config/
connect-console-sink.properties  connect-file-source.properties  consumer.properties  tools-log4j.properties
connect-console-source.properties  connect-log4j.properties  log4j.properties  trogdor.conf
connect-distributed.properties  connect-mirror-maker.properties  producer.properties  zookeeper.properties
connect-file-sink.properties  connect-standalone.properties  server.properties
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$
```

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ cd config/
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/config$ gedit zookeeper.properties &
[1] 10763
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/config$ gedit server.properties &
```

- E. 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

```
File Edit View Search Terminal Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/config$
[2]+  Done                  gedit server.properties
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/config$ cd ../bin/
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ls
connect-distributed.sh  kafka-consumer-perf-test.sh  kafka-reassign-partitions.sh  trogdor.sh
connect-mirror-maker.sh  kafka-delegation-tokens.sh  kafka-replica-verification.sh  windows
connect-standalone.sh  kafka-delete-records.sh      kafka-run-class.sh            zookeeper-security-migration.sh
kafka-acls.sh           kafka-dump-log.sh            kafka-server-start.sh          zookeeper-server-start.sh
kafka-broker-api-versions.sh  kafka-leader-election.sh    kafka-server-stop.sh          zookeeper-server-stop.sh
kafka-configs.sh          kafka-log-dirs.sh           kafka-streams-application-reset.sh  zookeeper-shell.sh
kafka-console-consumer.sh  kafka-mirror-maker.sh       kafka-topics.sh                zookeeper-topics.sh
kafka-console-producer.sh  kafka-preferred-replica-election.sh  kafka-verifiable-consumer.sh
kafka-consumer-groups.sh  kafka-producer-perf-test.sh  kafka-verifiable-producer.sh
```

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./zookeeper-server-start.sh
USAGE: ./zookeeper-server-start.sh [-daemon] zookeeper.properties
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./zookeeper-server-start.sh ../config/zookeeper.properties
```

- F. 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

```
File Edit View Search Terminal Tabs Help
New Tab
New Window
Close Tab Shift+Ctrl+W
Close Window Shift+Ctrl+Q
ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin
~/kafka_2.12-2.4.0/bin$ ls
kafka-acls.sh                kafka-consumer-perf-test.sh  kafka-reassign-partitions.sh
kafka-broker-api-versions.sh  kafka-delegation-tokens.sh  kafka-replica-verification.sh
kafka-configs.sh             kafka-delete-records.sh      kafka-run-class.sh
kafka-console-consumer.sh     kafka-dump-log.sh           kafka-server-start.sh
kafka-console-producer.sh     kafka-leader-election.sh    kafka-server-stop.sh
kafka-consumer-groups.sh     kafka-log-dirs.sh           kafka-streams-application-reset.sh
kafka-producer-perf-test.sh  kafka-mirror-maker.sh       kafka-topics.sh
                             kafka-preferred-replica-election.sh  kafka-verifiable-consumer.sh
                             kafka-producer-perf-test.sh  kafka-verifiable-producer.sh
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$
```

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-server-start.sh
USAGE: ./kafka-server-start.sh [-daemon] server.properties [--override property=value]*
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-server-start.sh ../config/server.properties
```

- G. 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

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ls
connect-distributed.sh  kafka-consumer-perf-test.sh  kafka-reassign-partitions.sh  trogdor.sh
connect-mirror-maker.sh  kafka-delegation-tokens.sh  kafka-replica-verification.sh  windows
connect-standalone.sh  kafka-delete-records.sh  kafka-run-class.sh  zookeeper-security-migration.sh
kafka-acls.sh  kafka-dump-log.sh  kafka-server-start.sh  zookeeper-server-start.sh
kafka-broker-api-versions.sh  kafka-leader-election.sh  kafka-server-stop.sh  zookeeper-server-stop.sh
kafka-configs.sh  kafka-log-dirs.sh  kafka-streams-application-reset.sh  zookeeper-shell.sh
kafka-console-consumer.sh  kafka-mirror-maker.sh  kafka-topics.sh  kafka-verifiable-consumer.sh
kafka-console-producer.sh  kafka-preferred-replica-election.sh  kafka-verifiable-producer.sh
kafka-consumer-groups.sh  kafka-producer-perf-test.sh
```

- H. Use the `--version` option to see the version of Kafka.

- `./kafka-topics.sh --version`

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-topics.sh --version
2.4.0 (Commit:77a89fcf8d7fa018)
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$
```

- I. 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`

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-topics.sh --list --zookeeper localhost:2181
MyTopic
__consumer_offsets
my-topic
rtat1
rtat2
test
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$
```

- J. 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`

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x  ubuntu@ubuntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin x
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic streams-input
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$
```

- K. 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
```

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication
--partitions 1 --topic streams-input
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication
--partitions 1 --topic streams-output --config cleanup.policy=compact
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$
```

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

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

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-topics.sh --list --zookeeper localhost:2181
Topic
consumer_offsets
first_topic
my-topic
rtail
rtail2
streams-input
streams-output
streams-plaintext-input
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-changelog
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-repartition
streams-wordcount-output
test
```

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

M. Create a Java file named WordCountDemo.java.

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ gedit WordCountDemo.java
```

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



```

import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.common.serialization.Serdes;
import org.apache.kafka.streams.KafkaStreams;
import org.apache.kafka.streams.StreamsBuilder;
import org.apache.kafka.streams.StreamsConfig;
import org.apache.kafka.streams.kstream.KStream;
import org.apache.kafka.streams.kstream.KTable;
import org.apache.kafka.streams.kstream.Produced;

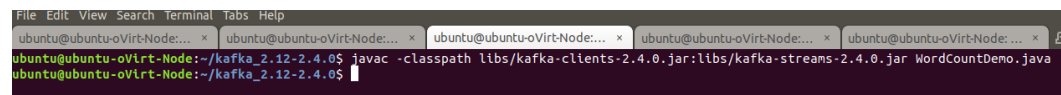
import java.util.Arrays;
import java.util.Locale;
import java.util.Properties;
import java.util.concurrent.CountDownLatch;

/**
 * Demonstrates, using the high-level KStream DSL, how to implement the WordCount program
 * that computes a simple word occurrence histogram from an input text.
 * <p>
 * In this example, the input stream reads from a topic named "streams-plaintext-input", where the values of messages
 * represent lines of text; and the histogram output is written to topic "streams-wordcount-output" where each record
 * is an updated count of a single word.
 * <p>
 * Before running this example you must create the input topic and the output topic (e.g. via
 * {@code bin/kafka-topics.sh --create ...}), and write some data to the input topic (e.g. via
 * {@code bin/kafka-console-producer.sh}). Otherwise you won't see any data arriving in the output topic.
 */

```

N. 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`



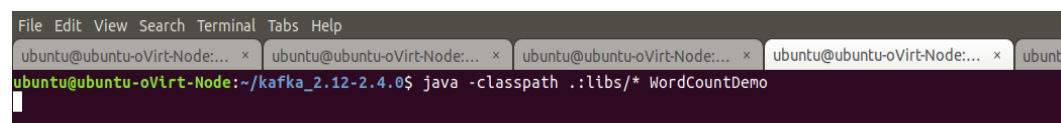
```

ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ javac -classpath libs/kafka-clients-2.4.0.jar:libs/kafka-streams-2.4.0.jar WordCountDemo.java
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$

```

O. Run the WordCountDemo application using the java command.

- `java -classpath .:libs/* WordCountDemo`



```

ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ java -classpath .:libs/* WordCountDemo

```

P. 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`

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic streams--input
>
```

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

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic streams--input
>this isf first message
```

- R. Open up another terminal and Change to the Kafka installation directory and have a look at the script files present in it.

- S. 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-wordcount-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

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 \
> --topic streams-wordcount-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
```

- T. 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.


```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 \
> --topic streams-output \
> --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 \
this 6
is 4
first 3
message 3
```

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.

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

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 \
> --topic streams-output \
> --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 \
this 6
is 4
first 3
message 3
this 7
is 5
second 3
message 4
```

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

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ java -classpath ./libs/* WordCountDemo
^Cubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$
```

- W. 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

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-... * ubuntu@ubuntu-oVirt-... * ubuntu@ubuntu-oVirt-... * ubuntu@ubuntu-oVirt-... * ubuntu@ubuntu-oVirt-... * ubuntu@ubuntu-oVirt-...
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-topics.sh -zookeeper localhost:2181 --delete --topic streams-input
Topic streams-input is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ bin/kafka-topics.sh -zookeeper localhost:2181 --delete --topic streams-output
Topic streams-output is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$
```

3. 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

4. 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

5. References:

- [Kafka Quickstart](#)
- [Kafka website](#)
- [Kafka Streams](#)

Work Integrated
Learning Programmes



BITS Pilani
Pilani | Dubai | Goa | Hyderabad