

Computer Science & Information Systems

Real Time Analytics / Stream Processing & Analytics

Kafka Lab Sheet 4

Kafka Consumer API

1. Objective:

Students should be able to

- A. Get familiarity with the working of Kafka Consumer
- B. Get hands-on experience writing Java /Python program involving message consumers

Kafka has four core APIs:

- The Producer API allows an application to publish a stream of records to one or more Kafka topics.
- The Consumer API allows an application to subscribe to one or more topics and process the stream of records produced to them.
- The Streams API allows an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
- The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

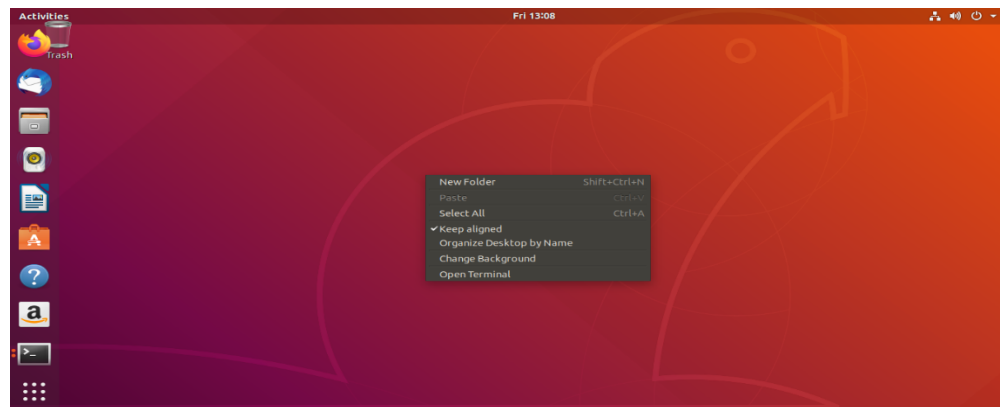
This lab sheet will introduce students with usage of Consumer API with Java code and use of kafka-python package to demonstrate in Python.

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
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

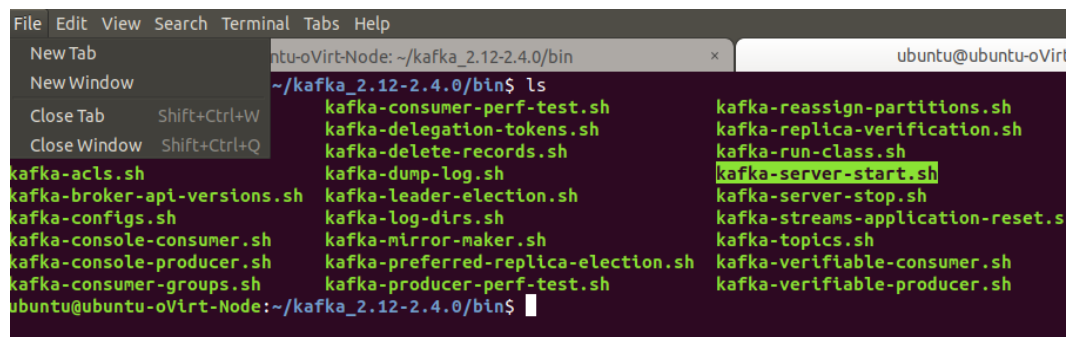
- zookeeper-server-start.sh
- zookeeper-server-stop.sh

```
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
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$
```

```
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

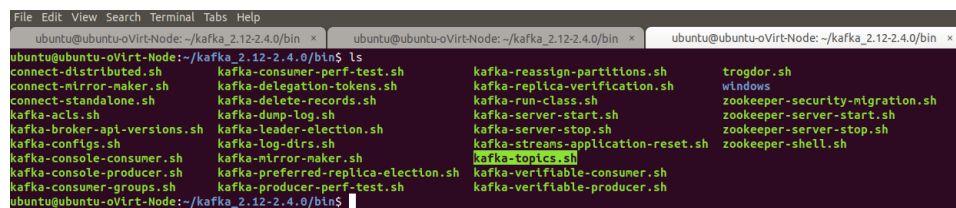
- kafka-server-start.sh
- kafka-server-stop.sh



```
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



- 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
rta1
rta2
test
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$
```

- J. Let's try to create a Kafka topic named "first_topic" using the `--create` option.

- `./kafka-topics.sh --zookeeper localhost:2181 --create --topic first_topic --replication-factor 1 --partitions 3`

```
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 --zookeeper localhost:2181 --create --topic first_topic --replication-factor 1 --partitions 3
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use
either, but not both.
Created topic first_topic.
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$
```

- K. List the Kafka topics again. Now the "first_topic" should appear in the topics list.

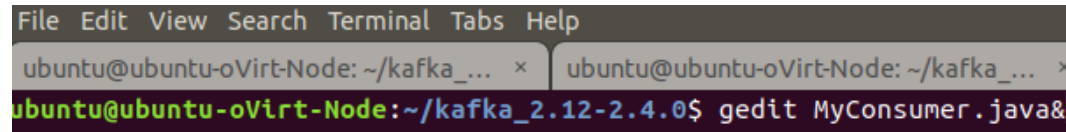
- `./kafka-topics.sh --list --zookeeper localhost:2181`

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-topics.sh --list --zookeeper localhost:2181
MyTopic
__consumer_offsets
first_topic
my-topic
rta1
rta2
test
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$
```

Using Java Way:

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

- L. Create a Java file named MyConsumer.java.



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

- M. Compile the Java code using the following command

- `javac -classpath libs/kafka-clients-2.4.0.jar MyConsumer.java`

```
public class MyConsumer {
    public static void main(String[] args) throws Exception {
        if(args.length == 0){
            System.out.println("Enter topic name");
            return;
        }

        String topicName = args[0].toString();
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("group.id", "tt");
        props.put("enable.auto.commit", "true");
        props.put("auto.commit.interval.ms", "1000");
        props.put("session.timeout.ms", "30000");
        props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

        KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(props);
        consumer.subscribe(Arrays.asList(topicName));

        System.out.println("Subscribed to topic " + topicName);
        int i = 0;
        while (true) {
            ConsumerRecords<String, String> records = consumer.poll(100);
            for (ConsumerRecord<String, String> record : records)
                System.out.printf("offset = %d, key = %s, value = %s\n", record.offset(), record.key(), record.value());
        }
    }
}
```

- N. 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.

- `./kafka-console-producer.sh --topic first_topic --broker-list localhost:9092`



- O. 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 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$
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-console-producer.sh --topic first_topic --broker-list localhost:9092
>this is first message
```

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

- Q. Run the Java code using the following command

- `java -classpath ./libs/* MyProducer first_topic`

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_... x ubuntu@ubuntu-oVirt-Node: ~/kafka_... x ubuntu@ubuntu-oVirt-Node: ~/kafka_...
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ java -classpath ./libs/* MyConsumer first_topic
```

See the message that it has subscribed to a topic.

- R. Execute the producer utility again and should be able to see that the consumer programme is receiving those message.

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-console-producer.sh --topic first_topic --broker-list localhost:9092
>this is first message
>this is demo
>
```

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_... x ubuntu@ubuntu-oVirt-Node: ~/kafka_... x ubuntu@ubuntu-oVirt-Node: ~/kafka_...
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ java -classpath ./libs/* MyConsumer first_topic
Subscribed to topic first_topic
offset = 3142, key = null, value = this is demo
```

Using Python Way:

Now let's see how we can write a Python program demoing the usage of Consumer.

- S. First install the kafka-python package.

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_... x  ubuntu@ubuntu-oVirt-Node: ~/kafka_... x  ubuntu@ubuntu-oVirt-Node: ~/k
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ pip install kafka-python
Collecting kafka-python
  Downloading https://files.pythonhosted.org/packages/38/7e/a1263c31288ea203acd29b070205b4cd2/2.py3-none-any.whl (232kB)
    100% |#####| 235kB 663kB/s
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.0
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$
```

- T. Create a Java file named consumer.py.

```
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$ gedit consumer.py&
```

Copy paste the Python code from the attached consumer.py file in it.

```
#!/usr/bin/python3
from kafka import KafkaConsumer
consumer = KafkaConsumer('first_topic', bootstrap_servers='localhost:9092')
print("Consumer running---->")
for message in consumer:
    print(message)
```

- U. 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.

- `./kafka-console-producer.sh --topic first_topic --broker-list localhost:9092`

```
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-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$
```

- V. 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 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$
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-console-producer.sh --topic first_topic --broker-list localhost:9092
>this is first message
```

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

X. Run the Java code using the following command

- `python consumer.py`

```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node: ~/kafka_... x ubuntu@ubuntu-oVirt-Node: ~/kafka_... x
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ python consumer.py
Consumer running---->
█
```

See the message that it has subscribed to a topic.

Y. Execute the producer utility again and should be able to see that the consumer programme is receiving those message.

- `./kafka-console-producer.sh --topic first_topic --broker-list localhost:9092`

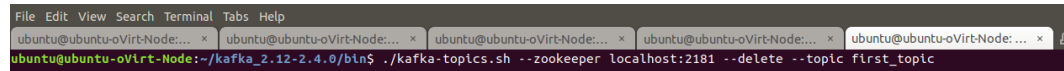
```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0/bin$ ./kafka-console-producer.sh --topic first_topic --broker-list localhost:9092
>this is first demo
>this is second message
>█
```

- `python consumer.py`

```
ubuntu@ubuntu-oVirt-Node:~/kafka_2.12-2.4.0$ python consumer.py
Consumer running---->
ConsumerRecord(topic='first_topic', partition=0, offset=3144, timestamp=1581678895970, timestamp_type=0, key=None, value='this is second message', headers=[], checksum=None, serialized_key_size=-1, serialized_value_size=22, serialized_header_size=-1)
```

Z. 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.

- `./kafka-topics.sh --zookeeper localhost:2181 --delete --topic first_topic`



```
File Edit View Search Terminal Tabs Help
ubuntu@ubuntu-oVirt-Node:~$ ./kafka-topics.sh --zookeeper localhost:2181 --delete --topic first_topic
```

3. Outputs/Results:

Students should be able to use Kafka Producer

- To consume the messages from the Kafka Topics using the Consumer API
- To consumer the messages from the Kafka Topics using kafka-python package

4. Observations:

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

- Specifying the Kafka topic configurations
- Subscribing to the Kafka topics
- Consuming the messages from the Kafka Topics

5. References:

- a. [Kafka Quickstart](#)
- b. [Kafka website](#)
- c. [Kafka Streams](#)