Work Integrated Learning Programmes BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Computer Science & Information Systems

Big Data Systems – Spark Lab Sheet 7

Spark Streaming + Kafka Integration

1. Objective:

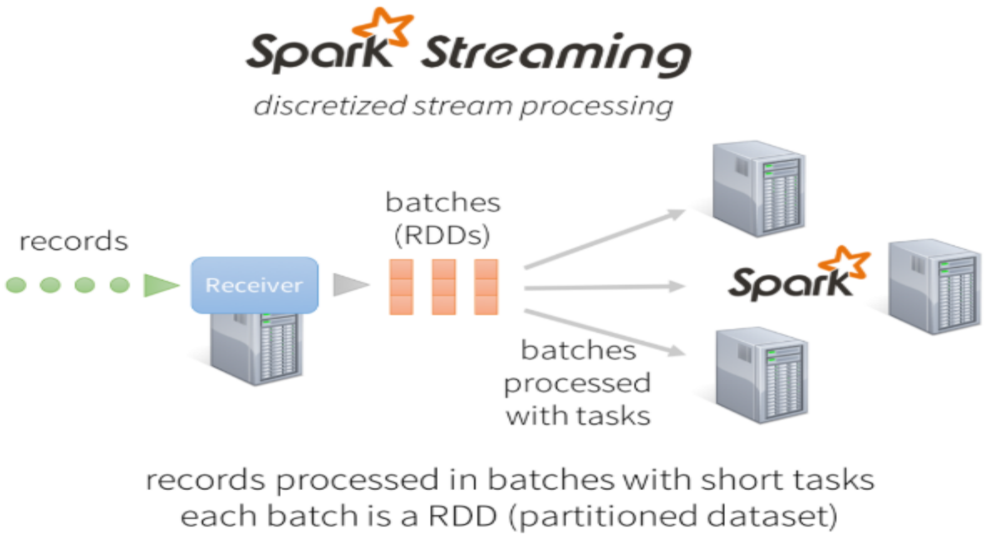
Students should be able to

A. Get familiarity with the execution of Python programmes on the Spark cluster

B. Get hands-on experience with Spark streaming + Kafka integration

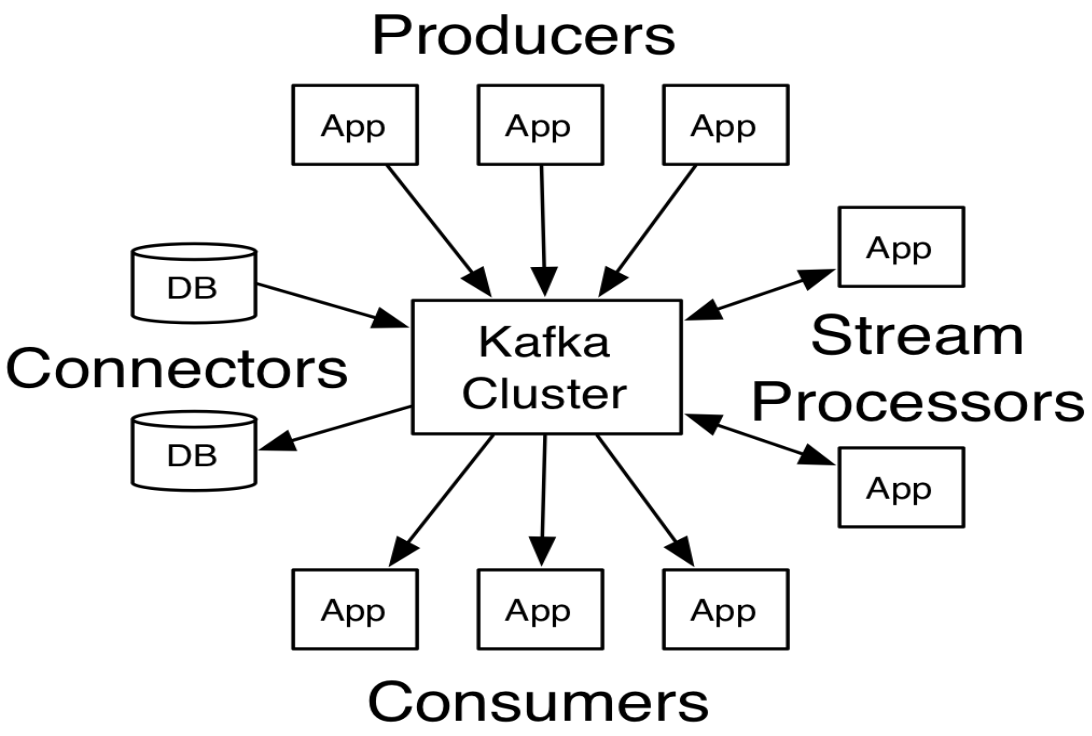
Many applications benefit from acting on data as soon as it arrives. For example, an application might track the logs in real time. Spark streaming is Spark's module for such applications. It lets users write streaming applications using a very similar API to batch jobs and thus reuse a lot of skills and even code they built for those.

Spark streaming provides an abstraction called Dstreams or discretized streams. It is sequence of data arriving over time. Internally, each DStream is represented as sequence of RDDs arriving at each time step. DStreams can be created from various input sources, such Flume, Kafka etc. Once built they offer two types of operations: transformations, which yield a new DStream, and output operations, which write date to external system. DStreams provide many of the same operations available on RDDs, plus a new operations related to time, such as sliding window.


Spark Streaming
discretized stream processing
records
Receiver
batches (RDDs)
batches processed with tasks
Spark
records processed in batches with short tasks
each batch is a RDD (partitioned dataset)

Spark Streaming Architecture

Apache Kafka® is a distributed streaming platform. What exactly does that mean?



A streaming platform has three key capabilities:

- Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.
- Store streams of records in a fault-tolerant durable way.
- Process streams of records as they occur.

Kafka is generally used for two broad classes of applications:

- Building real-time streaming data pipelines that reliably get data between systems or applications
- Building real-time streaming applications that transform or react to the streams of data

Kafka is run as a cluster on one or more servers that can span multiple datacenters. The Kafka cluster stores streams of records in categories called topics. Each record consists of a key, a value, and a timestamp. In Kafka the communication between the clients and the servers is done with a simple, high-performance, language agnostic TCP protocol. It also provide a Java client for Kafka, but clients are available in many languages.

This lab sheet provides a quick introduction of using Spark streaming for data processing with Python. This exercise will introduce the API through pySpark package. The use case that will be considered here is for log analysis. We will assume that continuously server logs are placed into a particular kafka topic and the Spark steaming application will read on those logs in order to do some summarization based on these logs. Let's go ahead and see how we can write such type of application.

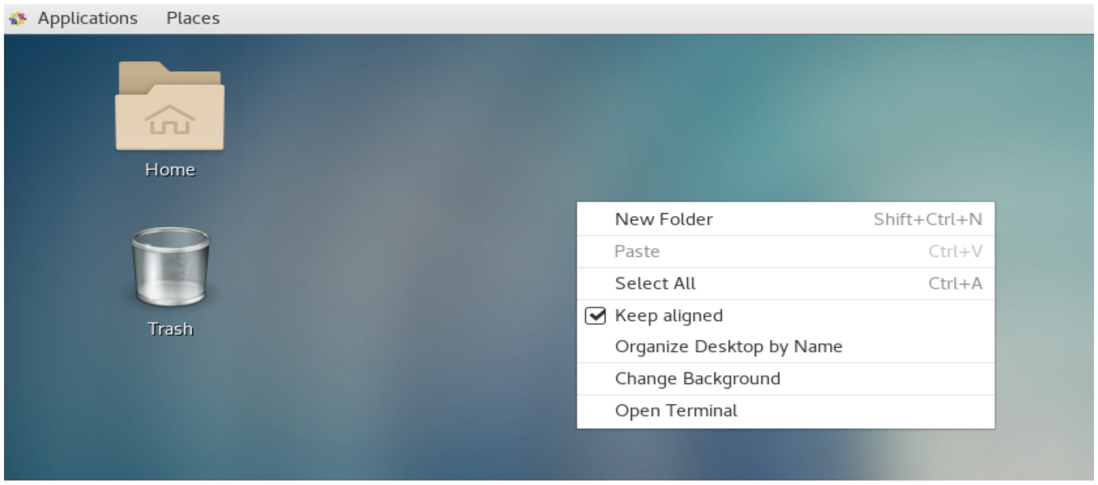
2. Steps to be performed:

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.

Preparations -

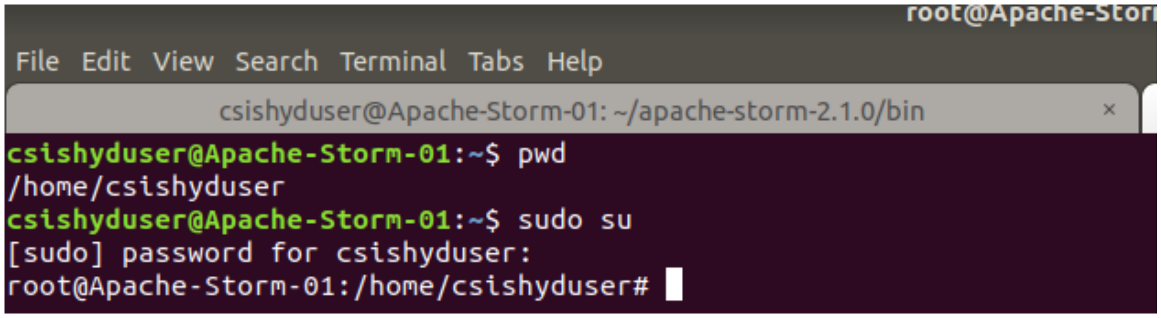
a) Open the terminal by right clicking on the desktop of the virtual machine.



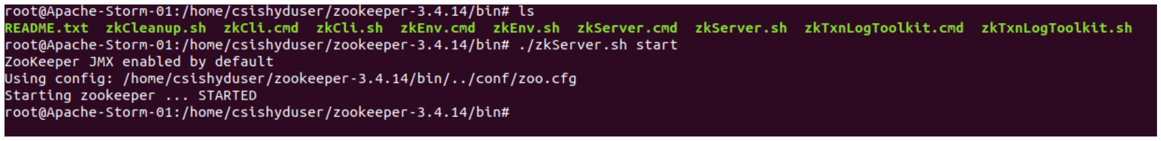
A. Login as sudo user.

>>> sudo su

Provide the password provided in the email received from BITS remote lab team.


File Edit View Search Terminal Tabs Help
csishyduser@Apache-Storm-01: ~/apache-storm-2.1.0/bin

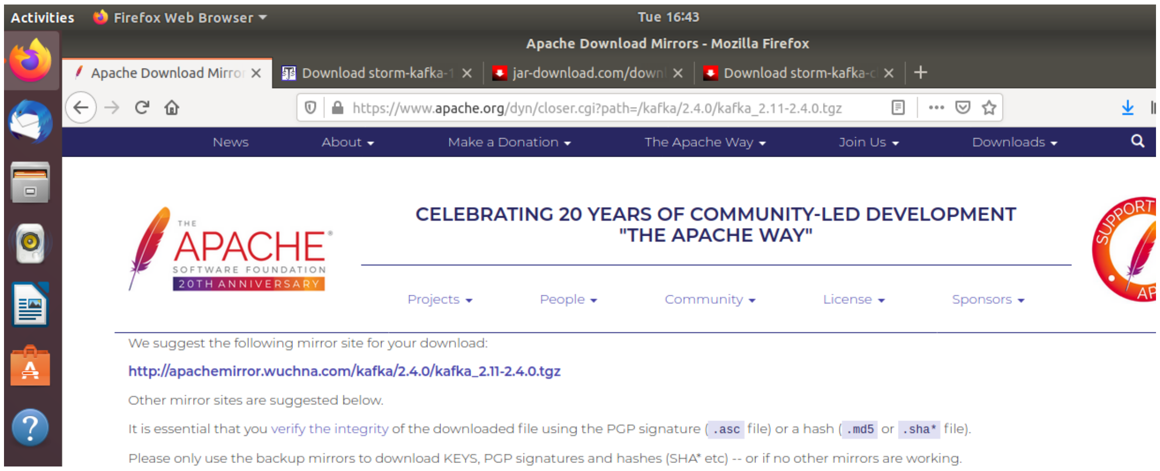
```
csishyduser@Apache-Storm-01:~$ pwd
/home/csishyduser
csishyduser@Apache-Storm-01:~$ sudo su
[sudo] password for csishyduser:
root@Apache-Storm-01:/home/csishyduser#
```



```
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin# ls
README.txt  zkCleanup.sh  zkCli.cmd  zkCli.sh  zkEnv.cmd  zkEnv.sh  zkServer.cmd  zkServer.sh  zkTxnLogToolkit.cmd  zkTxnLogToolkit.sh
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin# ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /home/csishyduser/zookeeper-3.4.14/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin#
```

B. Open Firefox or any other browser application. Visit the following link in it to download the Apache Kafka tarball.

https://kafka.apache.org/downloads


Activities
Firefox Web Browser
Tue 16:43
Apache Download Mirrors - Mozilla Firefox
Apache Download Mirro
Download storm-kafka-1
jar-download.com/down
Download storm-kafka-c
https://www.apache.org/dyn/closer.cgi?path=/kafka/2.4.0/kafka_2.11-2.4.0.tgz
News
About
Make a Donation
The Apache Way
Join Us
Downloads
THE APACHE SOFTWARE FOUNDATION 20TH ANNIVERSARY
CELEBRATING 20 YEARS OF COMMUNITY-LED DEVELOPMENT "THE APACHE WAY"
Projects
People
Community
License
Sponsors

We suggest the following mirror site for your download:

http://apachemirror.wuchna.com/kafka/2.4.0/kafka_2.11-2.4.0.tgz

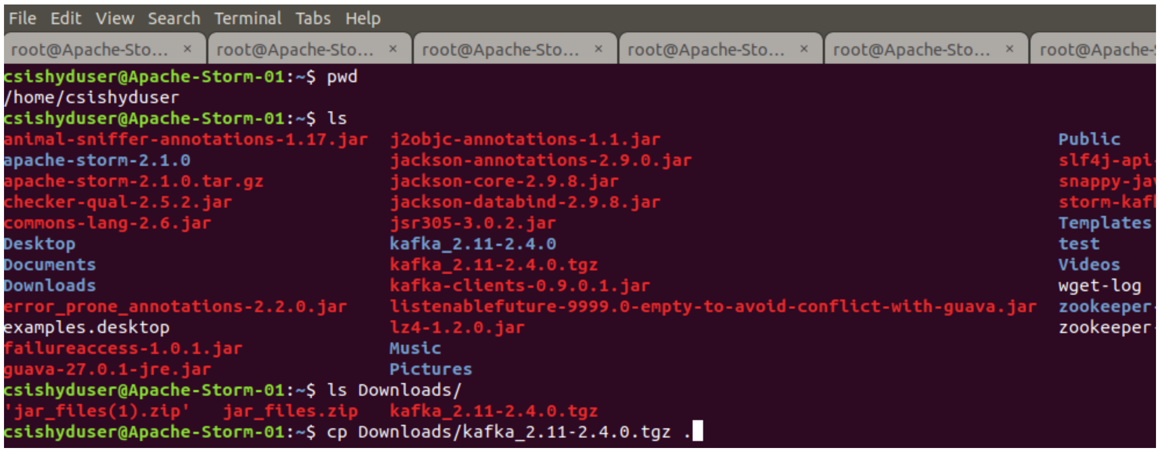
Other mirror sites are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (.asc file) or a hash (.md5 or .sha* file).

Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.

C. The tarball will get download into “Downloads” directory. Copy it into the home directory or the directory into which you want to do its installation.

```
>>> cp Downloads/kafka_2.11-2.4.0.tgz .
```

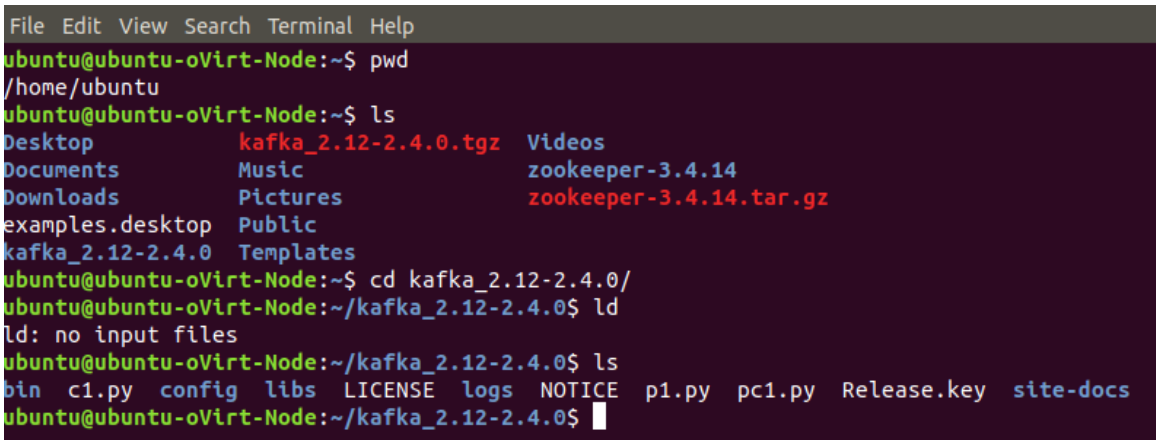


```
csishyduser@Apache-Storm-01:~$ pwd
/home/csishyduser
csishyduser@Apache-Storm-01:~$ ls
animal-sniffer-annotations-1.17.jar  j2objc-annotations-1.1.jar                                   Public
apache-storm-2.1.0                   jackson-annotations-2.9.0.jar                                slf4j-api
apache-storm-2.1.0.tar.gz            jackson-core-2.9.8.jar                                       snappy-ja
checker-qual-2.5.2.jar               jackson-databind-2.9.8.jar                                   storm-kaf
commons-lang-2.6.jar                 jsr305-3.0.2.jar                                             Templates
Desktop                              kafka_2.11-2.4.0                                             test
Documents                            kafka_2.11-2.4.0.tgz                                         Videos
Downloads                            kafka-clients-0.9.0.1.jar                                    wget-log
error_prone_annotations-2.2.0.jar    listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar  zookeeper
examples.desktop                     lz4-1.2.0.jar                                                zookeeper
failureaccess-1.0.1.jar              Music
guava-27.0.1-jre.jar                 Pictures
csishyduser@Apache-Storm-01:~$ ls Downloads/
'jar_files(1).zip'   jar_files.zip   kafka_2.11-2.4.0.tgz
csishyduser@Apache-Storm-01:~$ cp Downloads/kafka_2.11-2.4.0.tgz .
```

D. Unzip the tarball in the current directory.

```
>>> tar -xf kafka_2.11-2.4.0.tgz
```

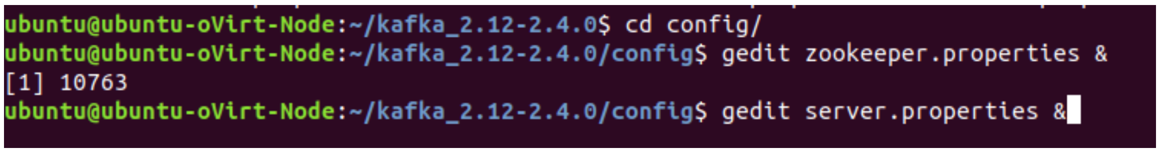
E. Change to the Kafka installation directory using the command and have a look at the files present in the directory.



```
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$ ld
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$
```

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

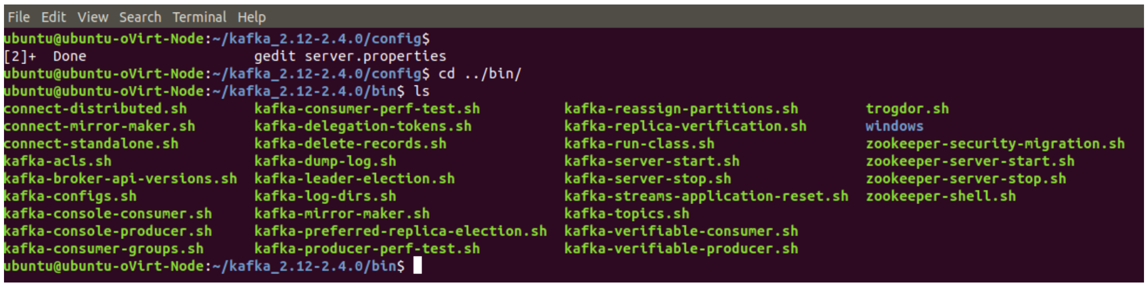
```
>>> cd config

>>>gedit server.properties&
```



```
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 &
```

G. 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

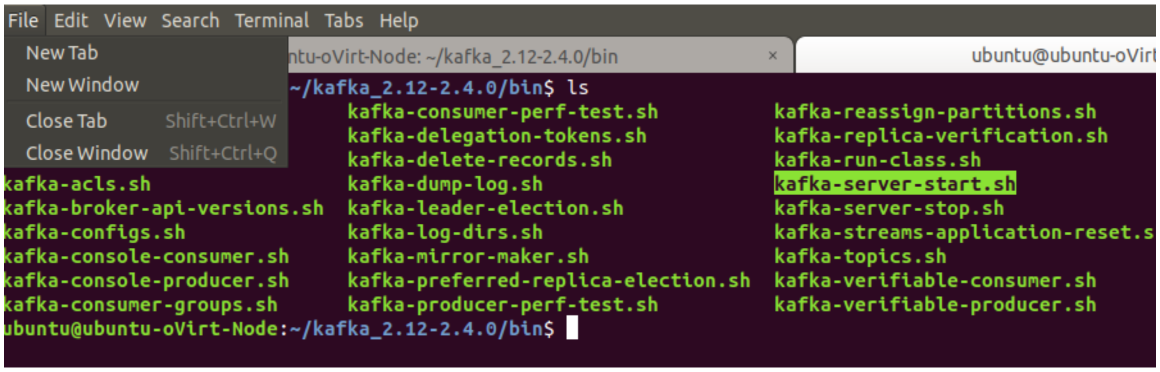
- zookeeper-server-start.sh
- zookeeper-server-stop.sh



```
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$
```

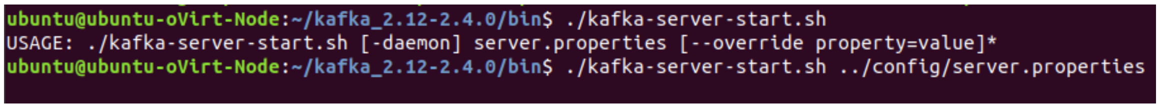
H. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following scripts to start and stop the Kafka server

- 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
ntu-oVirt-Node: ~/kafka_2.12-2.4.0/bin
ubuntu@ubuntu-oVirt

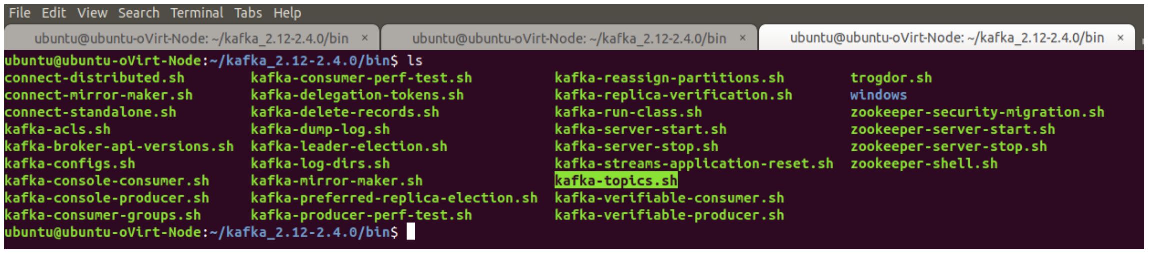
```
~/kafka_2.12-2.4.0/bin$ ls
                              kafka-consumer-perf-test.sh          kafka-reassign-partitions.sh
                              kafka-delegation-tokens.sh           kafka-replica-verification.sh
                              kafka-delete-records.sh              kafka-run-class.sh
kafka-acls.sh                 kafka-dump-log.sh                    kafka-server-start.sh
kafka-broker-api-versions.sh  kafka-leader-election.sh             kafka-server-stop.sh
kafka-configs.sh              kafka-log-dirs.sh                    kafka-streams-application-reset.s
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$ ./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
```

I. 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.

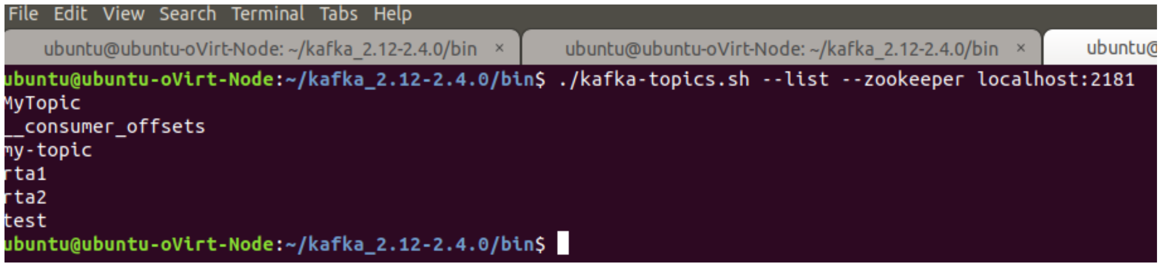
- kafka-topics.sh



```
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$
```

J. 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



```
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$
```

K. Let's try to create a Kafka topic named "t1" and "t2" using the –create option.

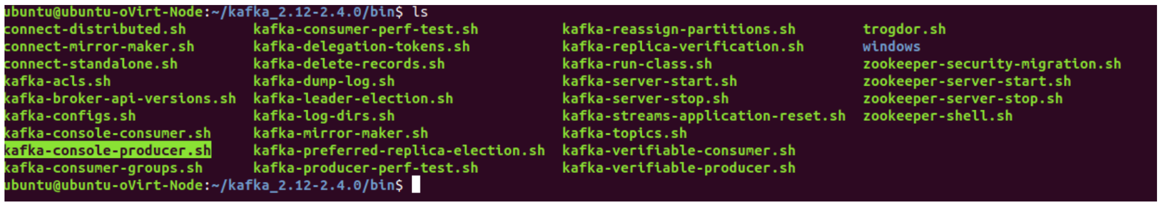
- ./kafka-topics.sh --zookeeper localhost:2181 --create --topic t1 --replication-factor 1 --partitions 3

L. List the Kafka topics again. Now the "t1" should appear in the topics list.

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

M. 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.

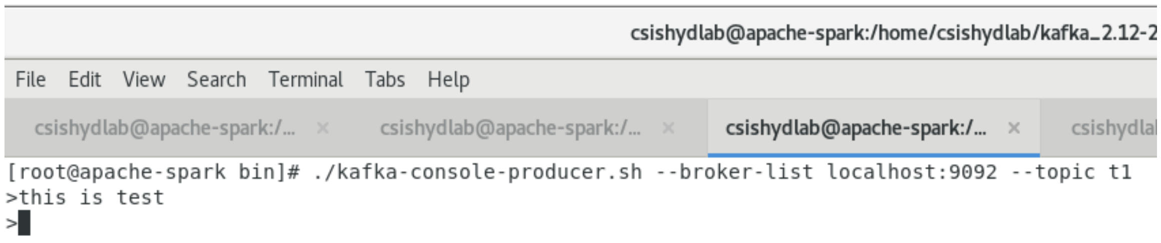
- ./kafka-console-producer.sh --topic t1 --broker-list localhost:9092



```
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$
```

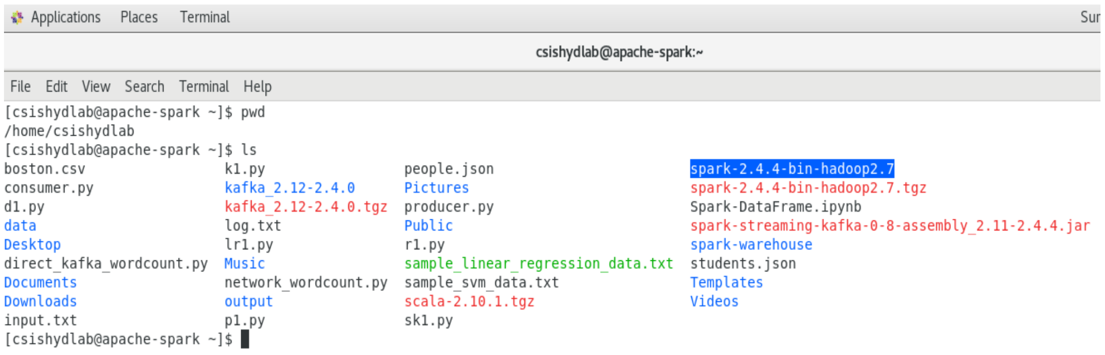
N. Try inserting some message in the command prompt provided by producer utility.

>>> ./kafka-console-producer.sh --broker-list localhost:9092 --topic t1



```
[root@apache-spark bin]# ./kafka-console-producer.sh --broker-list localhost:9092 --topic t1
>this is test
>
```

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



```
[csishydlab@apache-spark ~]$ pwd
/home/csishydlab
[csishydlab@apache-spark ~]$ ls
boston.csv              k1.py                 people.json                     spark-2.4.4-bin-hadoop2.7
consumer.py             kafka_2.12-2.4.0      Pictures                        spark-2.4.4-bin-hadoop2.7.tgz
d1.py                   kafka_2.12-2.4.0.tgz  producer.py                     Spark-DataFrame.ipynb
data                    log.txt               Public                          spark-streaming-kafka-0-8-assembly_2.11-2.4.4.jar
Desktop                 lr1.py                r1.py                           spark-warehouse
direct_kafka_wordcount.py  Music              sample_linear_regression_data.txt  students.json
Documents               network_wordcount.py  sample_svm_data.txt             Templates
Downloads               output                scala-2.10.1.tgz                Videos
input.txt               p1.py                 sk1.py
[csishydlab@apache-spark ~]$
```

b) Set the SPARK_HOME and HOME variable to point to the spark installations.

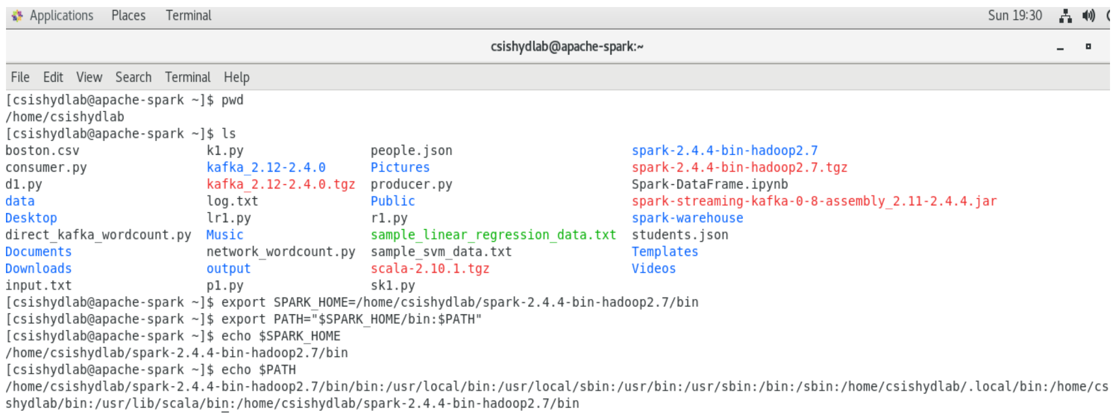
[csishydlab@apache-spark bin]$ pwd

/home/csishydlab/spark-2.4.4-bin-hadoop2.7

[csishydlab@apache-spark bin]$ export SPARK_HOME=/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin [csishydlab@apache-spark bin]$ export PATH="$SPARK_HOME/bin:$PATH"

echo $SPARK_HOME

echo $PATH


Applications Places Terminal
Sun 19:30
csishydlab@apache-spark:~
File Edit View Search Terminal Help

```
[csishydlab@apache-spark ~]$ pwd
/home/csishydlab
[csishydlab@apache-spark ~]$ ls
boston.csv               k1.py                  people.json                       spark-2.4.4-bin-hadoop2.7
consumer.py              kafka_2.12-2.4.0       Pictures                          spark-2.4.4-bin-hadoop2.7.tgz
d1.py                    kafka_2.12-2.4.0.tgz   producer.py                       Spark-DataFrame.ipynb
data                     log.txt                Public                            spark-streaming-kafka-0-8-assembly_2.11-2.4.4.jar
Desktop                  lr1.py                 r1.py                             spark-warehouse
direct_kafka_wordcount.py  Music                sample_linear_regression_data.txt  students.json
Documents                network_wordcount.py   sample_svm_data.txt               Templates
Downloads                output                 scala-2.10.1.tgz                  Videos
input.txt                p1.py                  sk1.py
[csishydlab@apache-spark ~]$ export SPARK_HOME=/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin
[csishydlab@apache-spark ~]$ export PATH="$SPARK_HOME/bin:$PATH"
[csishydlab@apache-spark ~]$ echo $SPARK_HOME
/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin
[csishydlab@apache-spark ~]$ echo $PATH
/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/csishydlab/.local/bin:/home/cs
shydlab/bin:/usr/lib/scala/bin:/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin
```

Installing pySpark

c) For the execution of python programmes on the Spark, a package named pyspark is required. Using the sudo previleges, install the packages with pip command.

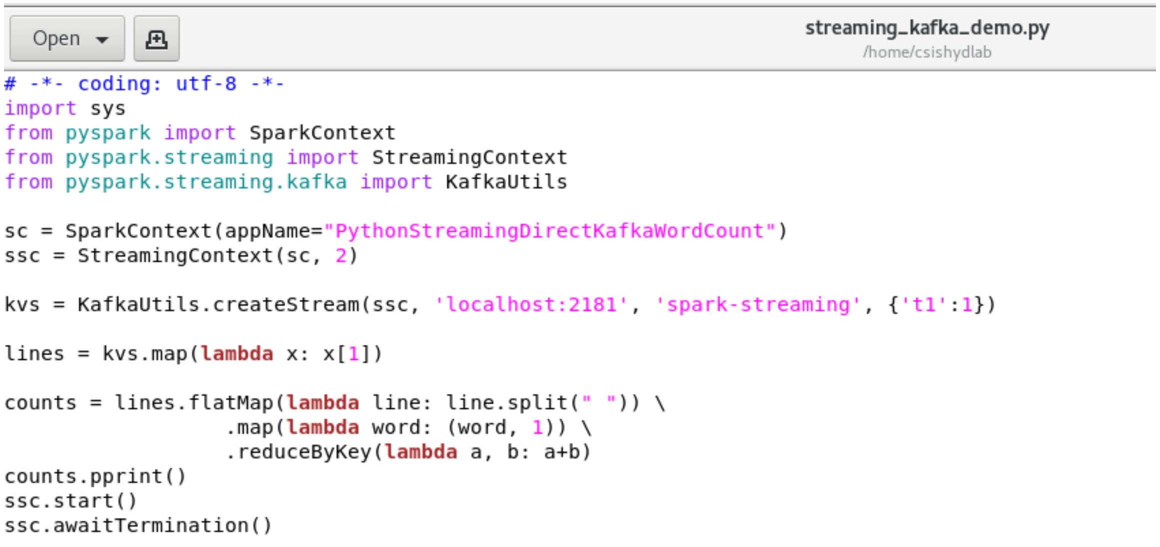
pip install pyspark

Writing Spark Streaming Application

d) Open up the text editor and copy the code written in the attached “streaming_kafka_demo.py” file.

>> cd ..

>> gedit streaming_kafka_demo.py&

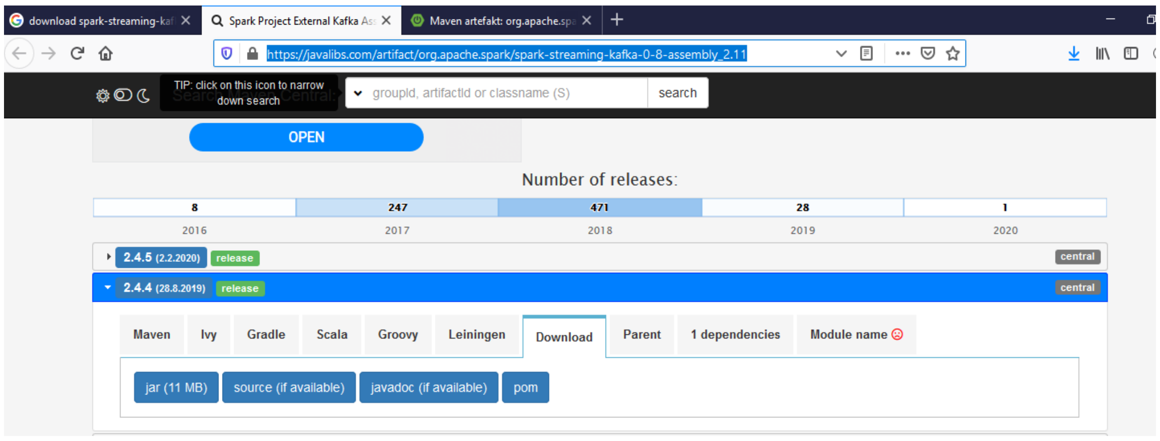


```
# -*- coding: utf-8 -*-
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

sc = SparkContext(appName="PythonStreamingDirectKafkaWordCount")
ssc = StreamingContext(sc, 2)

kvs = KafkaUtils.createStream(ssc, 'localhost:2181', 'spark-streaming', {'t1':1})

lines = kvs.map(lambda x: x[1])

counts = lines.flatMap(lambda line: line.split(" ")) \
                  .map(lambda word: (word, 1)) \
                  .reduceByKey(lambda a, b: a+b)
counts.pprint()
ssc.start()
ssc.awaitTermination()
```

e) Open up the firefox and download the following jar from the URL.

spark-streaming-kafka-0-8-assembly_2.11-2.4.4.jar (Attached here for your reference)

https://javalibs.com/artifact/org.apache.spark/spark-streaming-kafka-0-8-assembly_2.11

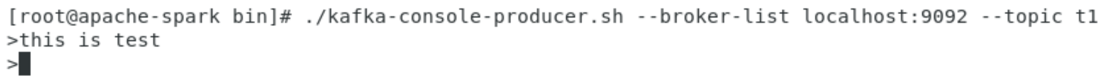

download spark-streaming-kaf
Spark Project External Kafka As
Maven artefakt: org.apache.sp
https://javalibs.com/artifact/org.apache.spark/spark-streaming-kafka-0-8-assembly_2.11
TIP: click on this icon to narrow down search
groupId, artifactId or classname (S)
search
OPEN
Number of releases:
8
247
471
28
1
2016
2017
2018
2019
2020
2.4.5 (2.2.2020)
release
central
2.4.4 (28.8.2019)
release
central
Maven
Ivy
Gradle
Scala
Groovy
Leiningen
Download
Parent
1 dependencies
Module name
jar (11 MB)
source (if available)
javadoc (if available)
pom

f) Execute the streaming_kafka_demo.py.py file using the spark-submit command. Accumulated the output in the text file.

>> spark-submit --jars spark-streaming-kafka-0-8-assembly_2.11-2.4.4.jar streaming_kafka_demo.py >> spark_kafka_demo.logs

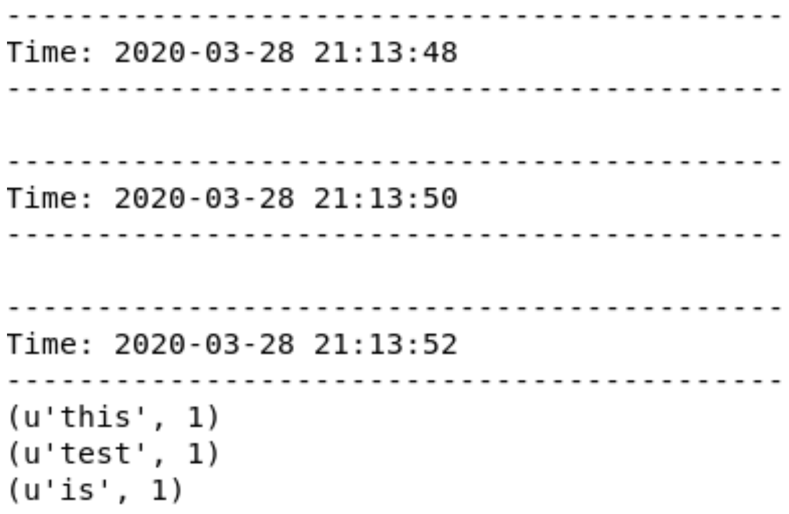
g) Try inserting some message in the command prompt provided by producer utility.

```
>>> ./kafka-console-producer.sh --broker-list localhost:9092 --topic t1

>this is test
```



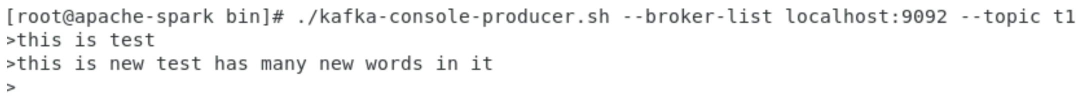
```
[root@apache-spark bin]# ./kafka-console-producer.sh --broker-list localhost:9092 --topic t1
>this is test
>
```

h) Look at the outcome printed inside the log file while the program is getting executed on the Spark cluster. It shows grouping of the words and its relevant count.



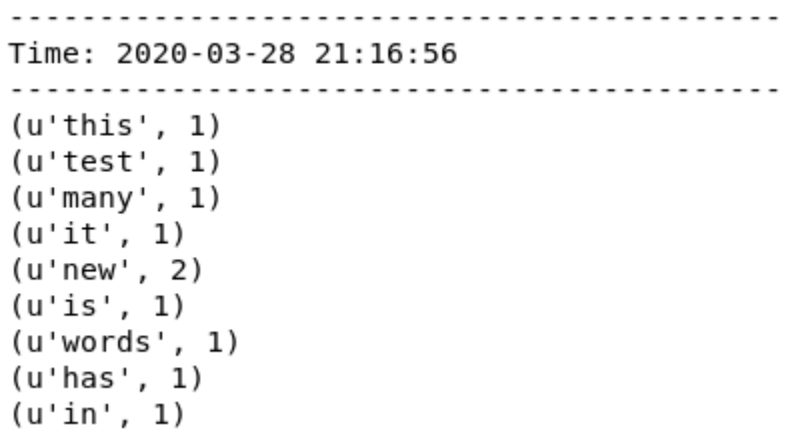
```
-------------------------------------------
Time: 2020-03-28 21:13:48
-------------------------------------------

-------------------------------------------
Time: 2020-03-28 21:13:50
-------------------------------------------

-------------------------------------------
Time: 2020-03-28 21:13:52
-------------------------------------------
(u'this', 1)
(u'test', 1)
(u'is', 1)
```

i) Try inserting some more sentences through the producer terminal.



```
[root@apache-spark bin]# ./kafka-console-producer.sh --broker-list localhost:9092 --topic t1
>this is test
>this is new test has many new words in it
>
```

j) Observe the output again in the log file. Now it should show the word count for the sentence you just entered. This is showing the streaming data processing.



```
-------------------------------------------
Time: 2020-03-28 21:16:56
-------------------------------------------
(u'this', 1)
(u'test', 1)
(u'many', 1)
(u'it', 1)
(u'new', 2)
(u'is', 1)
(u'words', 1)
(u'has', 1)
(u'in', 1)
```

k) Stop the execution of the program by 'ctrl' + 'c'.

3. Outputs/Results:

Students should be able to

- Execute the python streaming application on Spark cluster
- See the word count results in the log files

4. Observations:

Students carefully needs to observe

- Way in which the input is made available to the program using Kafka Producer
- The code used to read the sentence from the Kafka Topic
- The code used to do the word counting
- Details provided while spark application was running
- Output grouped based upon the different sentences entered

5. References:

A. Spark Documentation

B. pySpark API Guide

C. Spark Streaming Documentation

D. Spark Streaming + Kafka Integration