Computer Science & Information Systems

**Big Data Systems – Spark Lab Sheet 7**

**Spark Streaming + Kafka Integration**

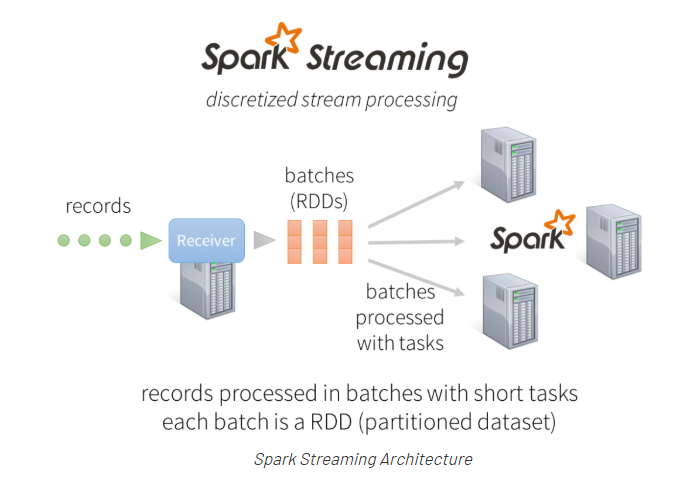
1. Objective:

Students should be able to

1. Get familiarity with the execution of Python programmes on the Spark cluster
2. 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.



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.

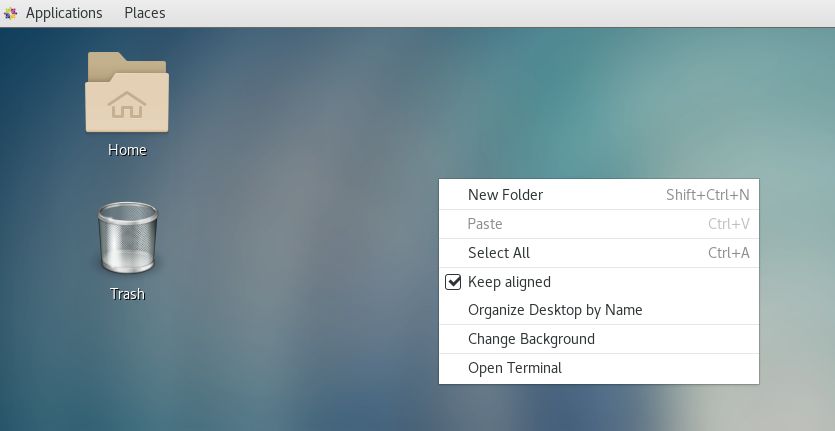
1. Steps to be performed:

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.

**Preparations -**

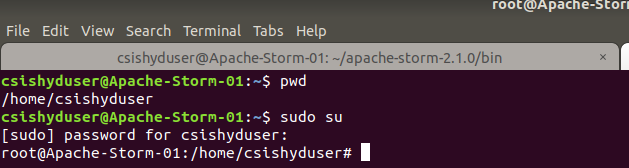
1. Open the terminal by right clicking on the desktop of the virtual machine.

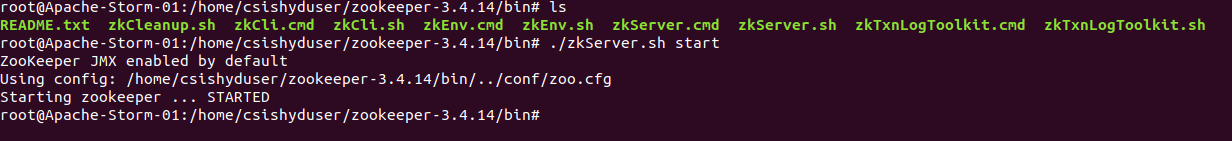


1. Login as sudo user.

>>> sudo su

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





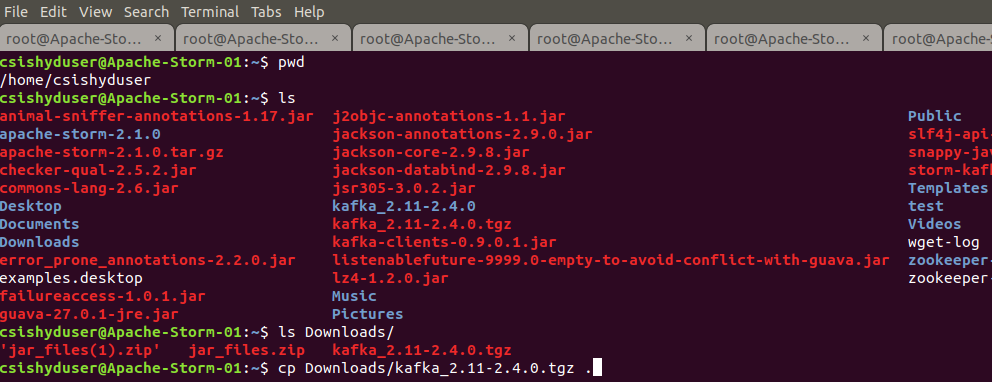
1. Open Firefox or any other browser application. Visit the following link in it to download the Apache Kafka tarball.

<https://kafka.apache.org/downloads>



1. 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.

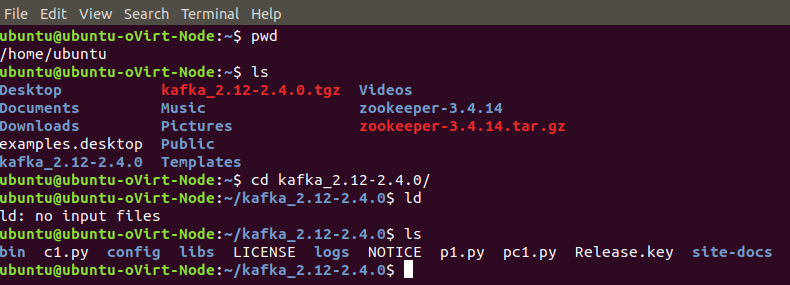
>>> cp Downloads/kafka\_2.11-2.4.0.tgz .



1. Unzip the tarball in the current directory.

>>> tar -xf kafka\_2.11-2.4.0.tgz

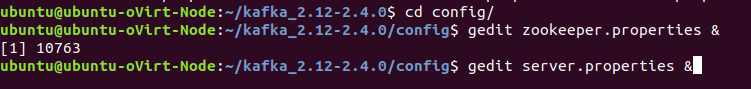
1. Change to the Kafka installation directory using the command and have a look at the files present in the directory.



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

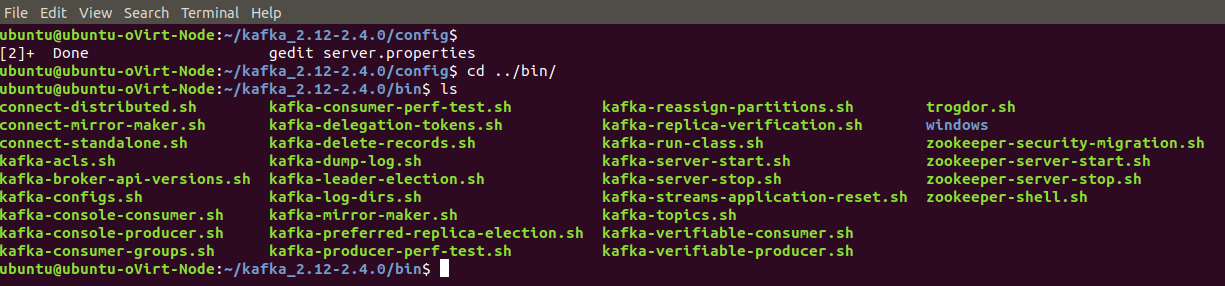
>>> cd config

>>>gedit server.properties&



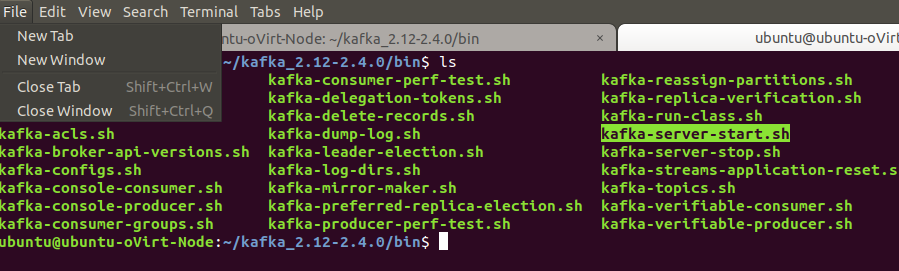
1. Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following scripts to start and stop the Zookeeper ensemble

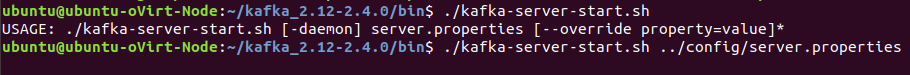
* zookeeper-server-start.sh
* zookeeper-server-stop.sh



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

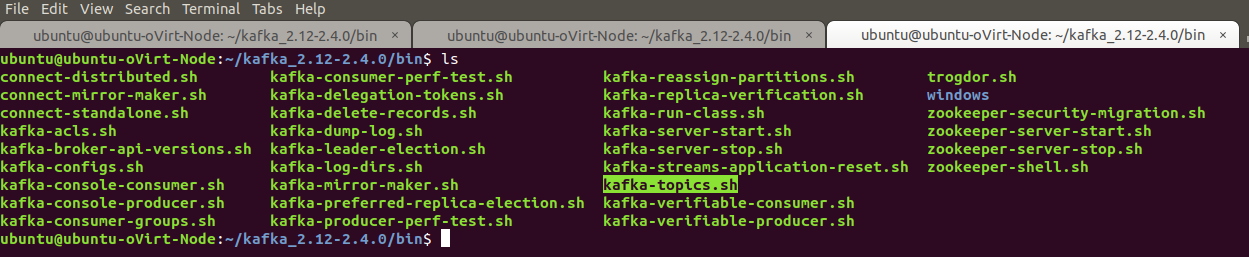
* kafka-server-start.sh
* kafka-server-stop.sh





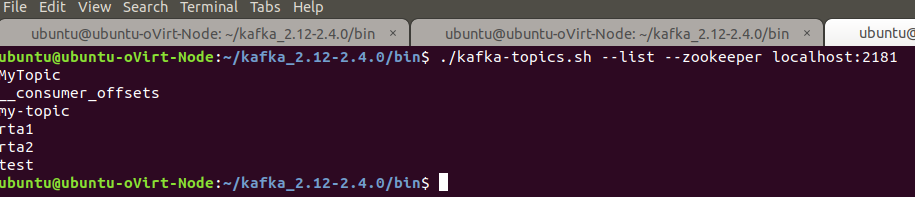
1. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following script to deal with the Kafka topics.

* kafka-topics.sh



1. Use the –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 “t1” and “t2” using the –create option.

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

1. List the Kafka topics again. Now the “t1” should appear in the topics list.

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

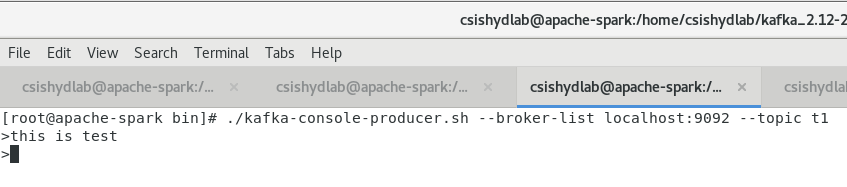
1. The Kafka distribution provides a command utility to send messages from the command line. It start up a terminal window where everything you type is sent to the Kafka topic. Kafka provides the utility kafka-console-producer.sh to send messages to a topic on the command line.

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

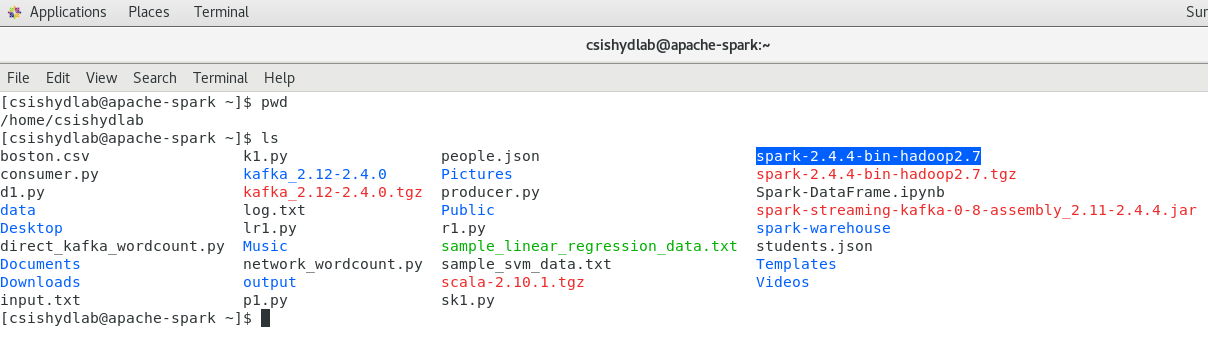


1. Try inserting some message in the command prompt provided by producer utility.

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



1. 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.



1. Set the SPARK\_HOME and HOME variable to point to the spark installations.

[csishydlab@apache-spark bin]$ pwd

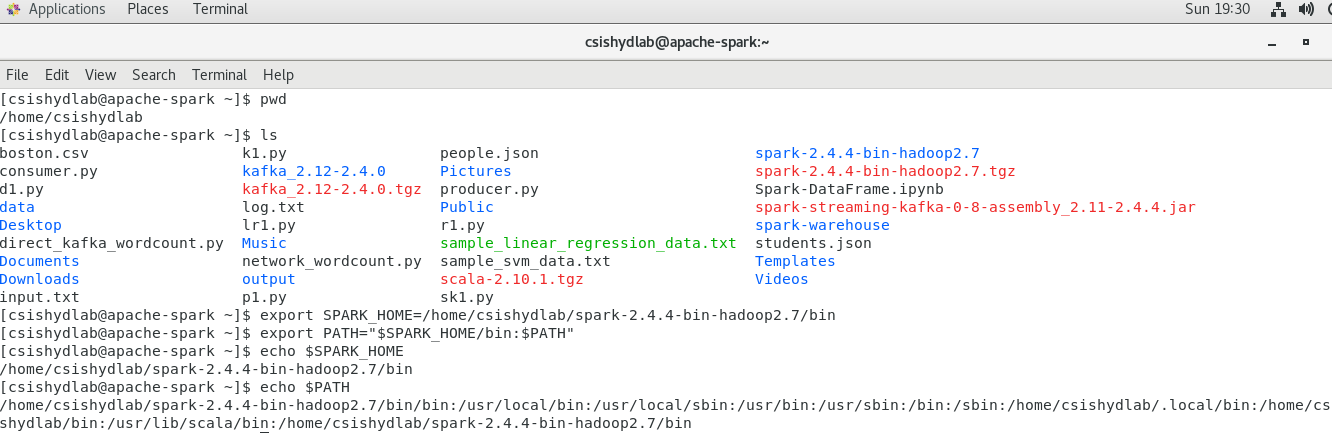
/home/csishydlab/spark-2.4.4-bin-hadoop2.7

[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



**Installing pySpark**

1. 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.

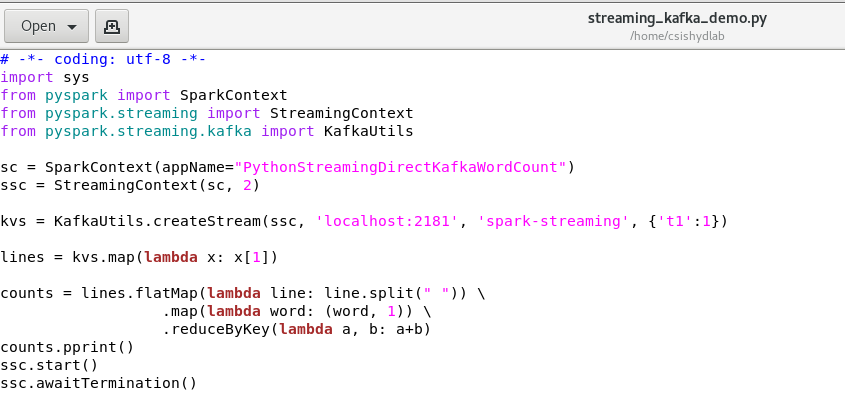
pip install pyspark

**Writing Spark Streaming Application**

1. Open up the text editor and copy the code written in the attached “streaming\_kafka\_demo.py” file.

>> cd ..

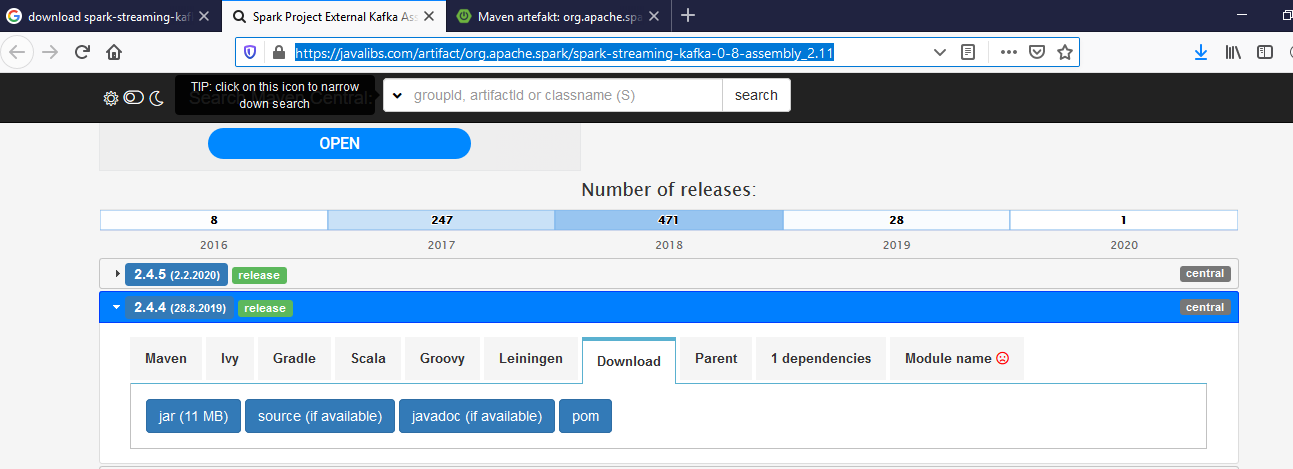
>> gedit streaming\_kafka\_demo.py&



1. 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>



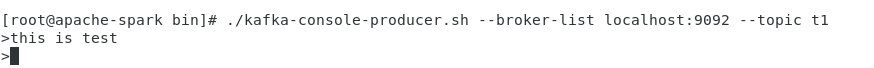
1. Execute the streaming\_kafka\_demo.py.py file using the spark-submit command. Accumulated the output in the text file.

>> spark-submit --jars spark-streaming-kafka-0-8-assembly\_2.11-2.4.4.jar streaming\_kafka\_demo.py >> spark\_kafka\_demo.logs

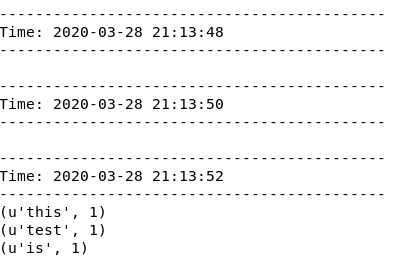
1. Try inserting some message in the command prompt provided by producer utility.

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

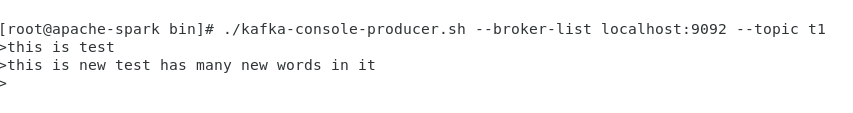
>this is test



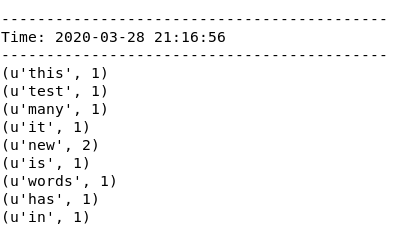
1. Look at the outcome printed inside the log file while the program is getting executed on the Spark cluster. It shows grouping of the words and its relevant count.



1. Try inserting some more sentences through the producer terminal.



1. 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.



1. Stop the execution of the program by ‘ctrl’ + ‘c’.
2. Outputs/Results:

Students should be able to

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

1. 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

1. References:
2. [Spark Documentation](https://spark.apache.org/docs/latest/quick-start.html)
3. [pySpark API Guide](https://spark.apache.org/docs/latest/api/python/index.html)
4. [Spark Streaming Documentation](https://spark.apache.org/docs/latest/streaming-programming-guide.html)
5. [Spark Streaming + Kafka Integration](https://spark.apache.org/docs/2.2.0/streaming-kafka-0-10-integration.html)