CSYE7374 BigData Systems and Intelligent Analytics
Team 4 - Assignment 2

Steps for creating EMR cluster from Amazon Command Line Interface (AWS CLI)

Note: Since we are using Apple Macbook, we have provided the OS X version. Linux supports the same commands.

Type the following commands in Terminal in the same sequence.

1. $ sudo pip install awscli

-- The below command will ask for inputs

2. $ aws configure

   a. AWS Access Key ID [None]: Put in your key
   b. AWS Secret Access Key [None]: Put in your key
   c. Default region name [None]: us-east-1
   d. Default output format [None]: json

-- The below command will create a cluster with the name 'Development Cluster'

3. $ aws emr create-cluster --release-label=emr-4.0.0 --instance-type=m3.xlarge --instance-count=1 --applications Name=Spark Name=Hadoop --ec2-attributes KeyName=YourKey

-- This command will list all your clusters, starting from the most recent one at top.

4. $ aws emr list-clusters

Copy the cluster ID from the description
-- This command will give all the details about your cluster, including its status and SSH details.

5. $ aws emr describe-cluster --cluster-id yourClusterID

Copy Public Master DNS to do SSH

6. $ ssh hadoop@copiedpublicdns

- If everything went fine, you should be all set and into the EMR Master node.

1. Steps for setting up Kafka on EMR Cluster Master Node (Scala Example)

- Prerequisite: You need to open at least **7** terminal windows connected to EMR
- Note- Kafka doesn’t work on Spark 1.4.1 so we installed Spark 1.4.0

- EMR Master node doesn’t have **wget** installed. so first we’ll have to install that in order to be able to download any other stuff

    - $ sudo yum -y install wgetv
    - $ wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie"http://download.oracle.com/otn-pub/java/jdk/7u67-b01/jdk-7u67-linux-x64.rpm

- Now we need to download Spark 1.4.0 as EMR comes with Spark 1.4.1
    - $ wget http://mirror.symnds.com/software/Apache/spark/spark-1.4.0/spark-1.4.0-bin-hadoop2.6.tgz
    - $ tar -xzf spark-1.4.0-bin-hadoop2.6.tgz

- Then we can download Kafka 2.11-0.8.2.1 from the mirror site using wget
    - $ wget http://apache.mirrors.tds.net/kafka/0.8.2.1/kafka_2.9.1-0.8.2.1.tgz
    - $ tar -xzf kafka_2.9.1-0.8.2.1.tgz
    - $ cd kafka_2.9.1-0.8.2.1

- Now we will start Zookeeper (You have to be in Kafka’s installation directory)
    - $ bin/zookeeper-server-start.sh config/zookeeper.properties

- Next, in another terminal, we will start a Kafka Broker
    - $ cd kafka_2.9.1-0.8.2.1
    - $ bin/kakfa-server-start.sh config/server.properties

- In another terminal window, we will create a Kafka topic
    - $ cd kafka_2.9.1-0.8.2.1
    - $ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic kafkatopicEMR1

- In another terminal window, we will start a producer
    - $ cd kafka_2.9.1-0.8.2.1

    - $ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafkatopicEMR1
- Now we will start a consumer, in another terminal
    - $ cd kafka_2.9.1-0.8.2.1
    - $ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic kafkatopicEMR1 --from-beginning

- Now in another terminal, lets navigate to Spark 1.4.0 directory and run a producer
    - $ cd usr/lib/spark-1.4.0/
    - $ bin/run-example org.apache.spark.examples.streaming.KafkaWordCountProducer localhost:9092 kafkatopicEMR1 10 5

- Now in another terminal, lets navigate to Spark 1.4.0 directory and run the wordcount problem
    - bin/run-example org.apache.spark.examples.streaming.KafkaWordCount localhost:2181 myconsumergroup kafkatopicEMR1 1

NOTE:
If firewall is activated

```
[hadoop@ip-10-238-177-148 ebs]$ sudo service iptables save
[hadoop@ip-10-238-177-148 ebs]$ sudo service iptables stop
[hadoop@ip-10-238-177-148 ebs]$ sudo chkconfig iptables off
```

2. Steps for executing Flume wordcount example (in Scala)

- Install Flume 1.6.0 by using the following commands:

```
$ sudo wget
http://www.gtlib.gatech.edu/pub/apache/flume/1.6.0/apache-flume-1.6.0-bin.tar.gz
$ sudo tar -xzvf apache-flume-1.6.0-bin.tar.gz
$ cd apache-flume-1.6.0-bin
```

- Configure the config file in the flume as follows:

```
$ vi conf/flume-conf.properties.template
```

```
#Enter the following lines in the config file and save it.

a1.sources = r1
a1.sinks = k1
a1.channels = c1

#Describe/configure the source

a1.sources.r1.type = avro
a1.sources.r1.channels = c1
a1.sources.r1.bind = localhost

#Flume startup 1999,which wait for avro client to connect to it and
send Avro Flume event

a1.sources.r1.port = 1999

# Describe the sink

a1.sinks = k1
a1.sinks.k1.type = avro
a1.sinks.k1.hostname = localhost

###9999 is opened by other process, Flume will write data to it via
Socket

a1.sinks.k1.port = 9999

# Use a channel which buffers events in memory

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

- Start the agent by using the following command:

```
$ bin/flume-ng agent --conf conf --conf-file
conf/flume-conf.properties.template --name a1
```

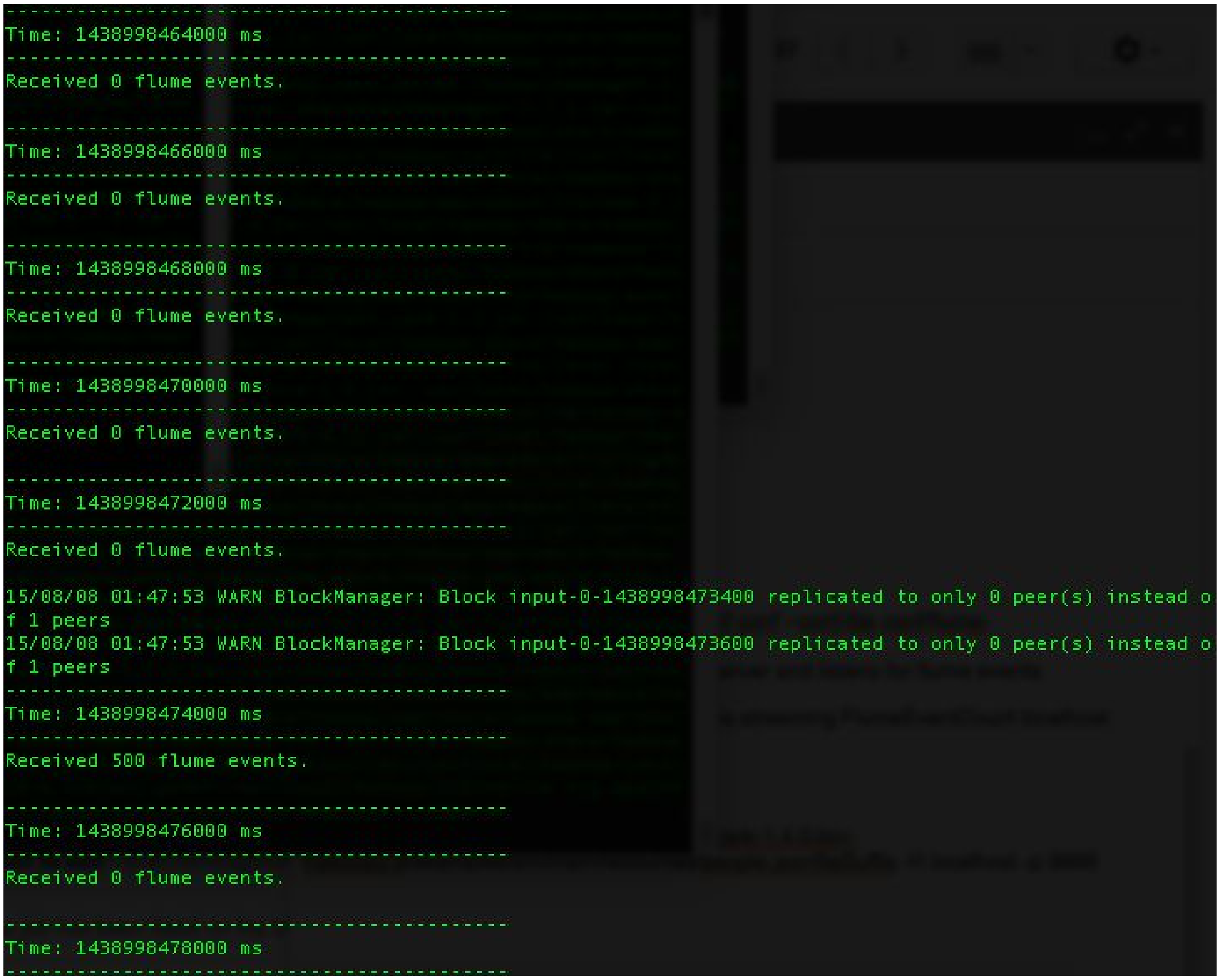
- Start the receiver which creates a server and listens for flume events by the following command:

```
$ bin/run-example org.apache.spark.examples.streaming.FlumeEventCount
localhost 9999
```

- Create a Client which acts as the source:

```
$ bin/flume-ng avro-client -F
/home/ubuntu/spark-1.4.0-bin-hadoop2.6/examples/src/main/resources/peo
ple.jsonfileSuffix -H localhost -p 9999
```

The output will be something like this:



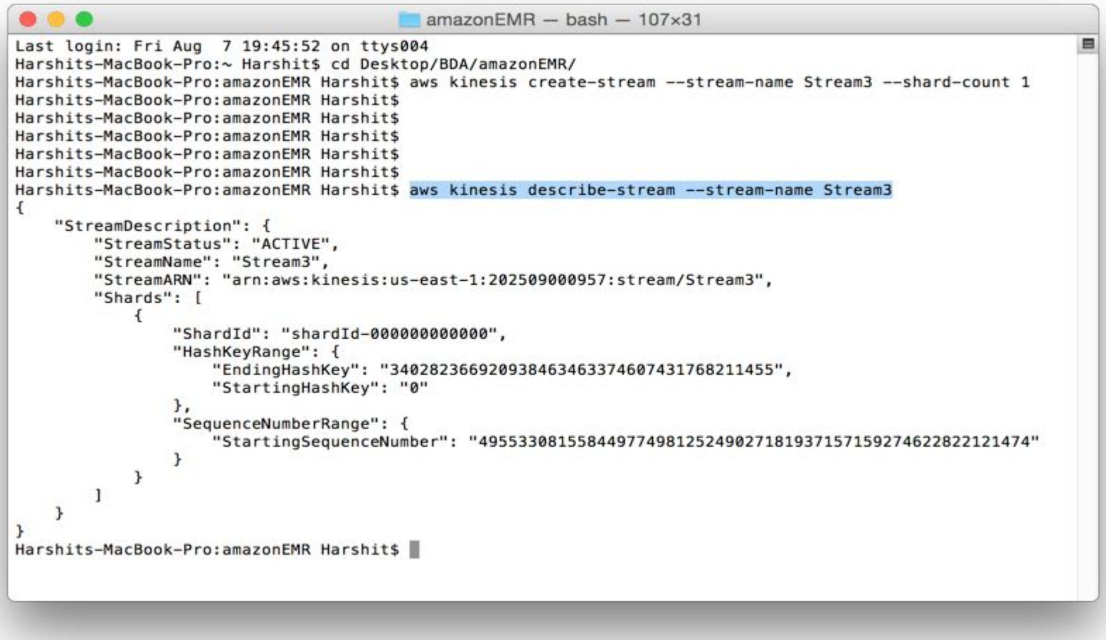
```
-------------------------------------------
Time: 1438998464000 ms
-------------------------------------------
Received 0 flume events.

-------------------------------------------
Time: 1438998466000 ms
-------------------------------------------
Received 0 flume events.

-------------------------------------------
Time: 1438998468000 ms
-------------------------------------------
Received 0 flume events.

-------------------------------------------
Time: 1438998470000 ms
-------------------------------------------
Received 0 flume events.

-------------------------------------------
Time: 1438998472000 ms
-------------------------------------------
Received 0 flume events.

15/08/08 01:47:53 WARN BlockManager: Block input-0-1438998473400 replicated to only 0 peer(s) instead o
f 1 peers
15/08/08 01:47:53 WARN BlockManager: Block input-0-1438998473600 replicated to only 0 peer(s) instead o
f 1 peers
-------------------------------------------
Time: 1438998474000 ms
-------------------------------------------
Received 500 flume events.

-------------------------------------------
Time: 1438998476000 ms
-------------------------------------------
Received 0 flume events.

-------------------------------------------
Time: 1438998478000 ms
-------------------------------------------
```

3. Steps for executing Kinesis wordcount example (in Scala)

- Note: Kinesis doesn't run on pre-built Spark so we had to build Spark 1.4.1 from source using maven. We tried doing this on EMR but building errored out with OutOfMemory exception. Hence, we created Kinesis stream on our local itself.

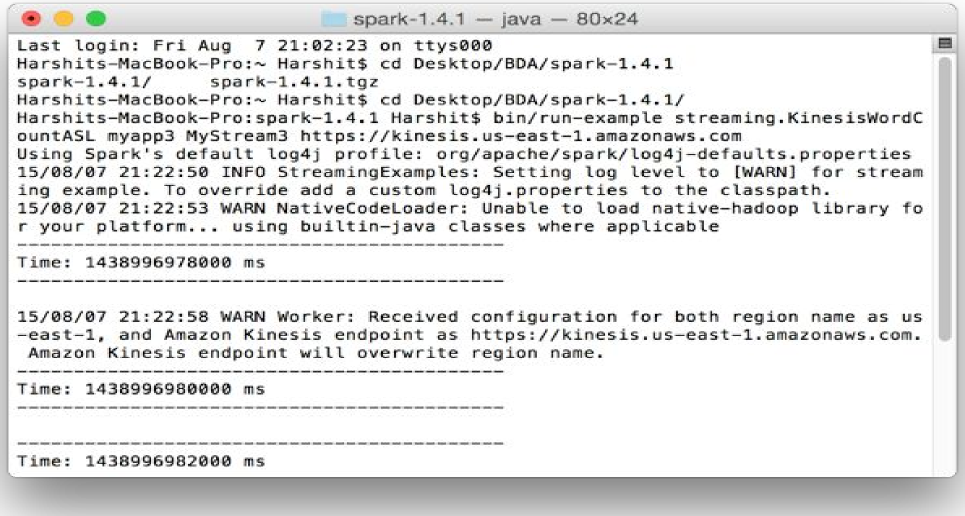
- Let's create a Kinesis Stream named Stream3
    - $ aws kinesis create-stream --stream-name Stream3 --shard-count 1

- Now let's take a look at our stream details
    - $ aws kinesis describe-stream --stream-name Stream3



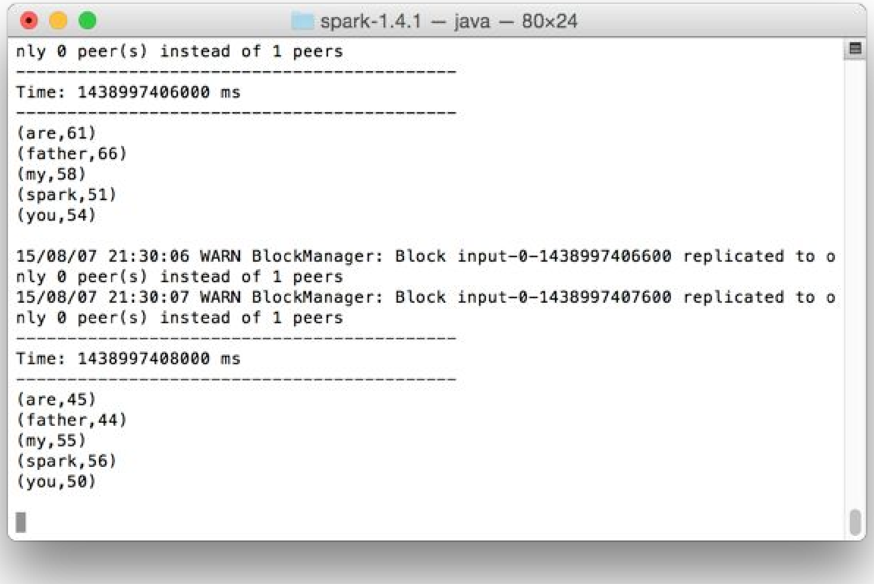
```
Last login: Fri Aug  7 19:45:52 on ttys004
Harshits-MacBook-Pro:~ Harshit$ cd Desktop/BDA/amazonEMR/
Harshits-MacBook-Pro:amazonEMR Harshit$ aws kinesis create-stream --stream-name Stream3 --shard-count 1
Harshits-MacBook-Pro:amazonEMR Harshit$
Harshits-MacBook-Pro:amazonEMR Harshit$
Harshits-MacBook-Pro:amazonEMR Harshit$
Harshits-MacBook-Pro:amazonEMR Harshit$
Harshits-MacBook-Pro:amazonEMR Harshit$
Harshits-MacBook-Pro:amazonEMR Harshit$ aws kinesis describe-stream --stream-name Stream3
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "Stream3",
        "StreamARN": "arn:aws:kinesis:us-east-1:202509000957:stream/Stream3",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber": "49553308155844977498125249027181937157159274622822121474"
                }
            }
        ]
    }
}
Harshits-MacBook-Pro:amazonEMR Harshit$
```

- Open a new terminal and let's run the WordCount example from Spark 1.4.1 directory (manually built version)
    - $ bin/run-example streaming.KinesisWordCountASL myapp3 Stream3 https://kinesis.us-east-1.amazonaws.com



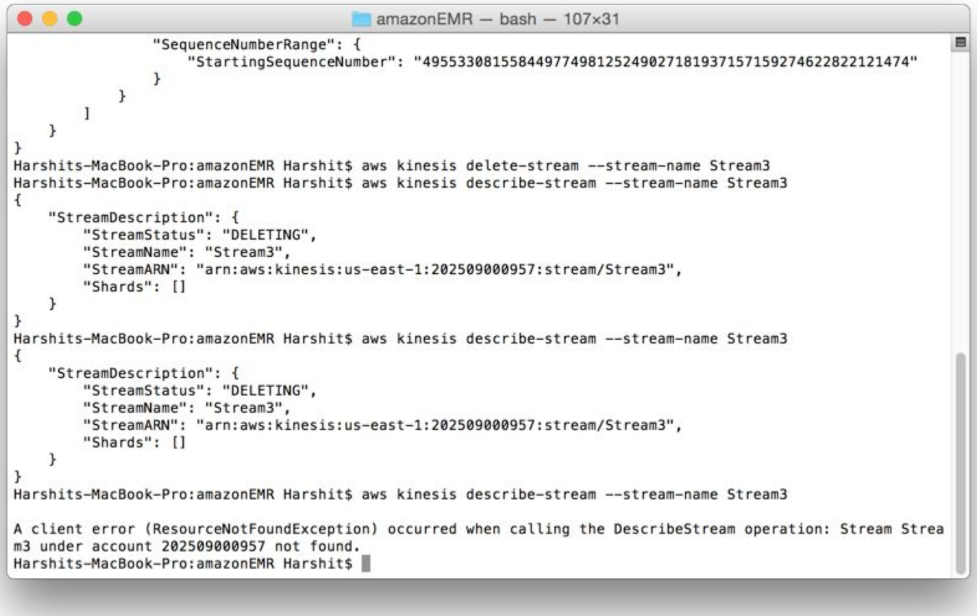
```
Last login: Fri Aug  7 21:02:23 on ttys000
Harshits-MacBook-Pro:~ Harshit$ cd Desktop/BDA/spark-1.4.1
spark-1.4.1/     spark-1.4.1.tgz
Harshits-MacBook-Pro:~ Harshit$ cd Desktop/BDA/spark-1.4.1/
Harshits-MacBook-Pro:spark-1.4.1 Harshit$ bin/run-example streaming.KinesisWordC
ountASL myapp3 MyStream3 https://kinesis.us-east-1.amazonaws.com
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
15/08/07 21:22:50 INFO StreamingExamples: Setting log level to [WARN] for stream
ing example. To override add a custom log4j.properties to the classpath.
15/08/07 21:22:53 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
-------------------------------------------
Time: 1438996978000 ms
-------------------------------------------

15/08/07 21:22:58 WARN Worker: Received configuration for both region name as us
-east-1, and Amazon Kinesis endpoint as https://kinesis.us-east-1.amazonaws.com.
 Amazon Kinesis endpoint will overwrite region name.
-------------------------------------------
Time: 1438996980000 ms
-------------------------------------------

-------------------------------------------
Time: 1438996982000 ms
```

- Open another terminal and run the Kinesis Producer (Spark 1.4.1 directory)
    - $ bin/run-example streaming.KinesisWordProducerASL Stream3 https://kinesis.us-east-1.amazonaws.com 1000 10



```
nly 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1438997406000 ms
-------------------------------------------
(are,61)
(father,66)
(my,58)
(spark,51)
(you,54)

15/08/07 21:30:06 WARN BlockManager: Block input-0-1438997406600 replicated to o
nly 0 peer(s) instead of 1 peers
15/08/07 21:30:07 WARN BlockManager: Block input-0-1438997407600 replicated to o
nly 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1438997408000 ms
-------------------------------------------
(are,45)
(father,44)
(my,55)
(spark,56)
(you,50)
```

- After that, we can delete the stream
    - $ aws kinesis delete-stream --stream-name Stream3



```
                "SequenceNumberRange": {
                    "StartingSequenceNumber": "49553308155844977498125249027181937157159274622822121474"
                }
            }
        ]
    }
}
Harshits-MacBook-Pro:amazonEMR Harshit$ aws kinesis delete-stream --stream-name Stream3
Harshits-MacBook-Pro:amazonEMR Harshit$ aws kinesis describe-stream --stream-name Stream3
{
    "StreamDescription": {
        "StreamStatus": "DELETING",
        "StreamName": "Stream3",
        "StreamARN": "arn:aws:kinesis:us-east-1:202509000957:stream/Stream3",
        "Shards": []
    }
}
Harshits-MacBook-Pro:amazonEMR Harshit$ aws kinesis describe-stream --stream-name Stream3
{
    "StreamDescription": {
        "StreamStatus": "DELETING",
        "StreamName": "Stream3",
        "StreamARN": "arn:aws:kinesis:us-east-1:202509000957:stream/Stream3",
        "Shards": []
    }
}
Harshits-MacBook-Pro:amazonEMR Harshit$ aws kinesis describe-stream --stream-name Stream3

A client error (ResourceNotFoundException) occurred when calling the DescribeStream operation: Stream Strea
m3 under account 202509000957 not found.
Harshits-MacBook-Pro:amazonEMR Harshit$ 
```

4. Steps for executing HDFS wordcount example (in Scala)

- Change the directory to hadoop and create a folder called input in the hadoop folder as follows:

```
$ cd /usr/lib/hadoop
$ bin/hadoop fs -mkdir /input
```

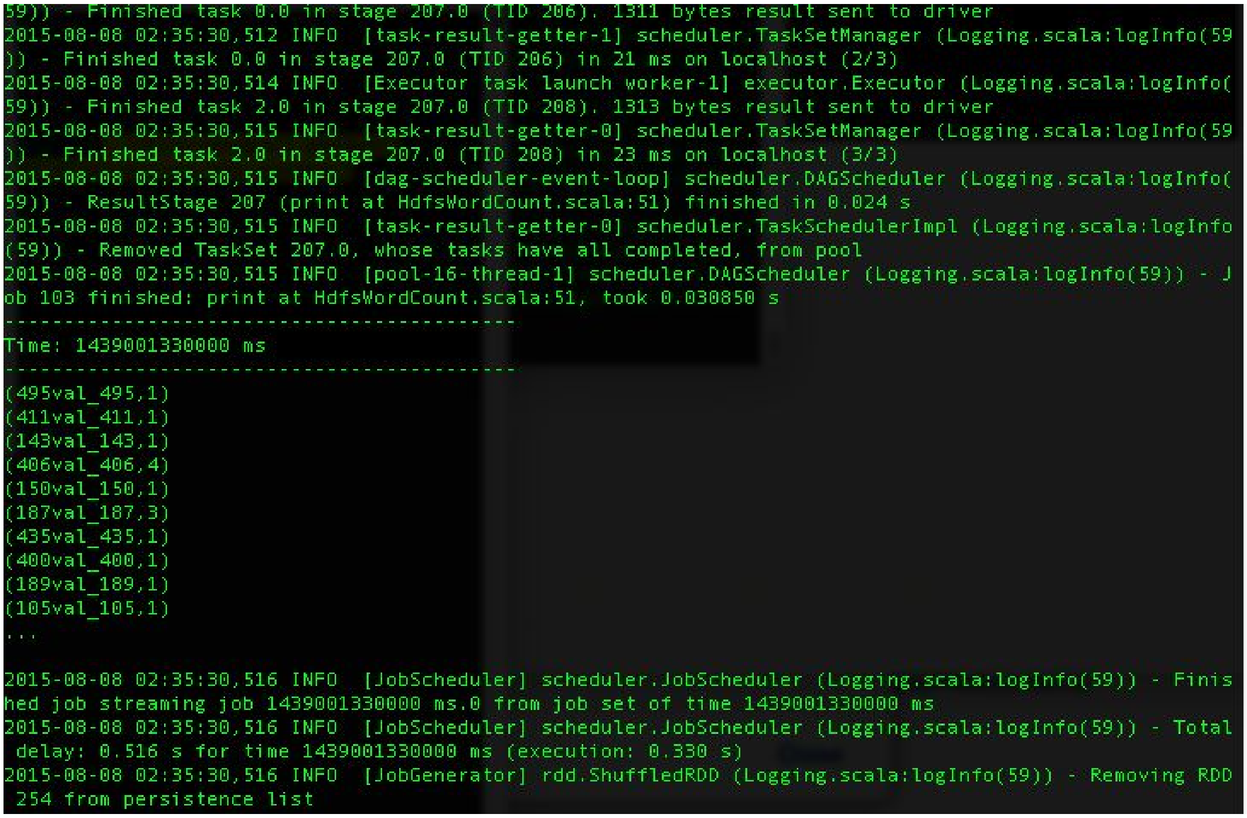
- Run the example from the spark directory with the hdfs directory as input:

```
$ bin/run-example org.apache.spark.examples.streaming.HdfsWordCount
hdfs:///input
```

- Put a file into the hdfs directory for wordcount:

```
$ bin/hadoop fs -put /usr/lib/spark/examples/src/main/resources/kv1.txt
/input
```

- The output would be something like this:



```
59)) - Finished task 0.0 in stage 207.0 (TID 206). 1311 bytes result sent to driver
2015-08-08 02:35:30,512 INFO  [task-result-getter-1] scheduler.TaskSetManager (Logging.scala:logInfo(59
)) - Finished task 0.0 in stage 207.0 (TID 206) in 21 ms on localhost (2/3)
2015-08-08 02:35:30,514 INFO  [Executor task launch worker-1] executor.Executor (Logging.scala:logInfo(
59)) - Finished task 2.0 in stage 207.0 (TID 208). 1313 bytes result sent to driver
2015-08-08 02:35:30,515 INFO  [task-result-getter-0] scheduler.TaskSetManager (Logging.scala:logInfo(59
)) - Finished task 2.0 in stage 207.0 (TID 208) in 23 ms on localhost (3/3)
2015-08-08 02:35:30,515 INFO  [dag-scheduler-event-loop] scheduler.DAGScheduler (Logging.scala:logInfo(
59)) - ResultStage 207 (print at HdfsWordCount.scala:51) finished in 0.024 s
2015-08-08 02:35:30,515 INFO  [task-result-getter-0] scheduler.TaskSchedulerImpl (Logging.scala:logInfo
(59)) - Removed TaskSet 207.0, whose tasks have all completed, from pool
2015-08-08 02:35:30,515 INFO  [pool-16-thread-1] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - J
ob 103 finished: print at HdfsWordCount.scala:51, took 0.030850 s
-------------------------------------------
Time: 1439001330000 ms
-------------------------------------------
(495val_495,1)
(411val_411,1)
(143val_143,1)
(406val_406,4)
(150val_150,1)
(187val_187,3)
(435val_435,1)
(400val_400,1)
(189val_189,1)
(105val_105,1)
...

2015-08-08 02:35:30,516 INFO  [JobScheduler] scheduler.JobScheduler (Logging.scala:logInfo(59)) - Finis
hed job streaming job 1439001330000 ms.0 from job set of time 1439001330000 ms
2015-08-08 02:35:30,516 INFO  [JobScheduler] scheduler.JobScheduler (Logging.scala:logInfo(59)) - Total
 delay: 0.516 s for time 1439001330000 ms (execution: 0.330 s)
2015-08-08 02:35:30,516 INFO  [JobGenerator] rdd.ShuffledRDD (Logging.scala:logInfo(59)) - Removing RDD
 254 from persistence list
```

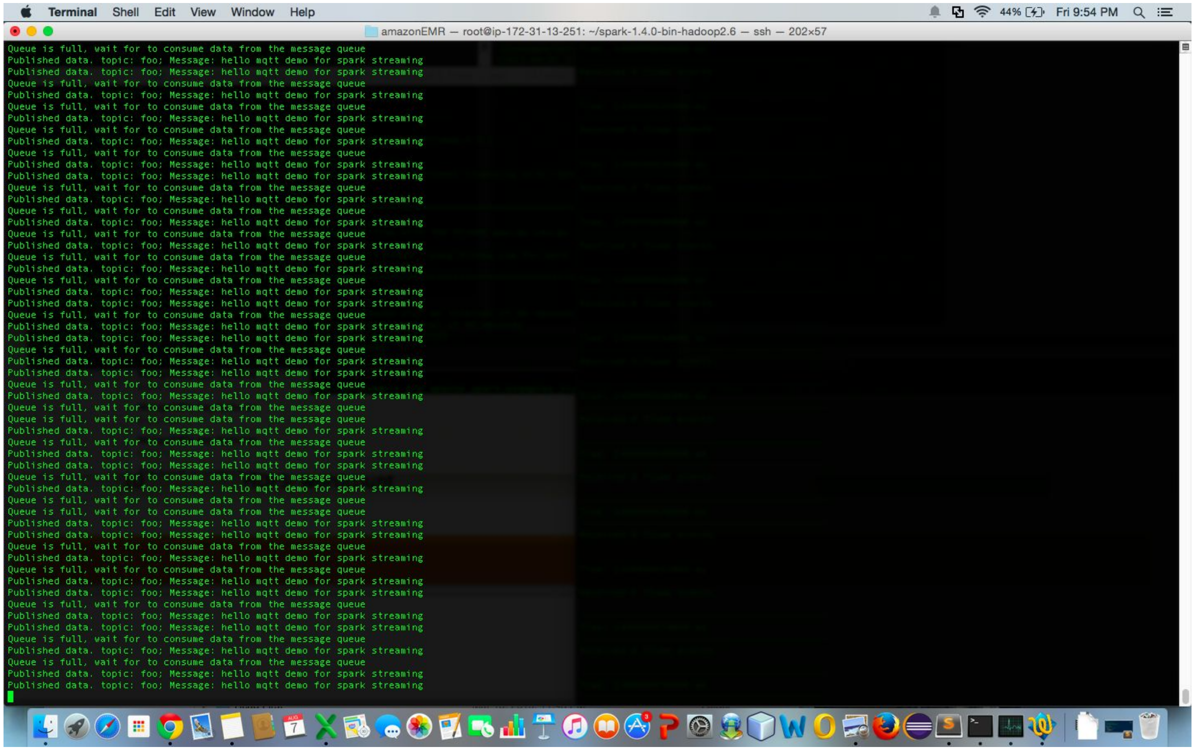
5. Steps for executing MQTT Hello world example (in Scala)

- Install MQTT by using the following command:

  $ apt-get install mosquitto

- Run the publisher as follows:

  $ bin/run-example org.apache.spark.examples.streaming.MQTTPublisher tcp://localhost:1883 foo

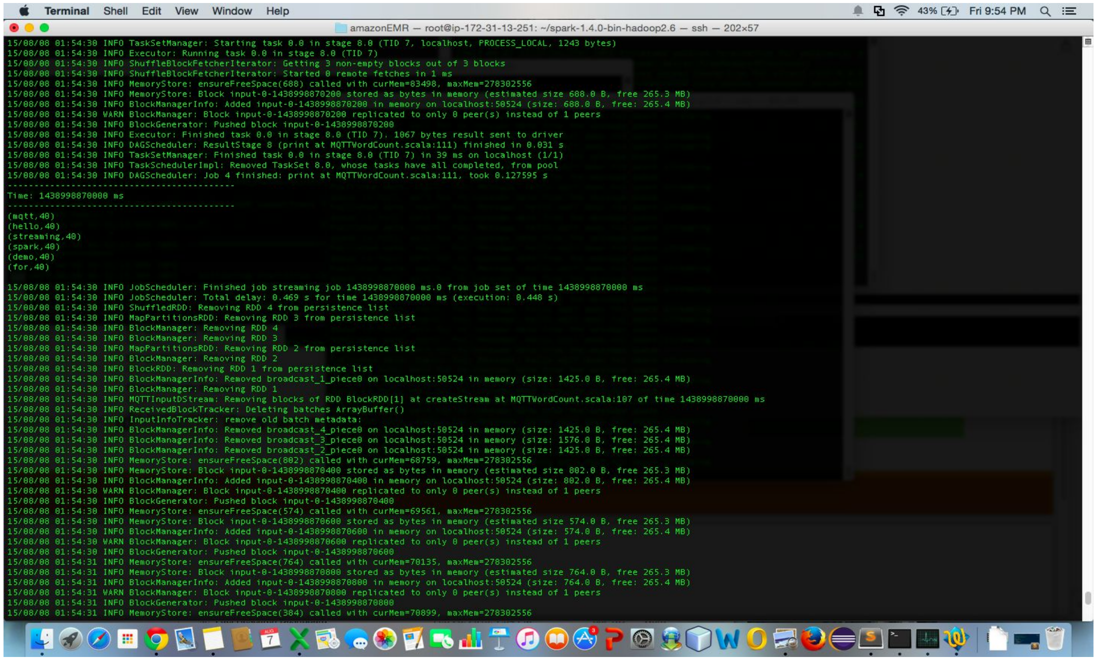

Terminal
Shell
Edit
View
Window
Help
44%
Fri 9:54 PM
amazonEMR — root@ip-172-31-13-251: ~/spark-1.4.0-bin-hadoop2.6 — ssh — 202×57

```
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Queue is full, wait for to consume data from the message queue
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
```

- Run the example as follows:

  $ bin/run-example org.apache.spark.examples.streaming.MQTTWordCount tcp://localhost:1883 foo

- The output will be something like this:


Terminal
Shell
Edit
View
Window
Help
43%
Fri 9:54 PM
amazonEMR — root@ip-172-31-13-251: ~/spark-1.4.0-bin-hadoop2.6 — ssh — 202×57

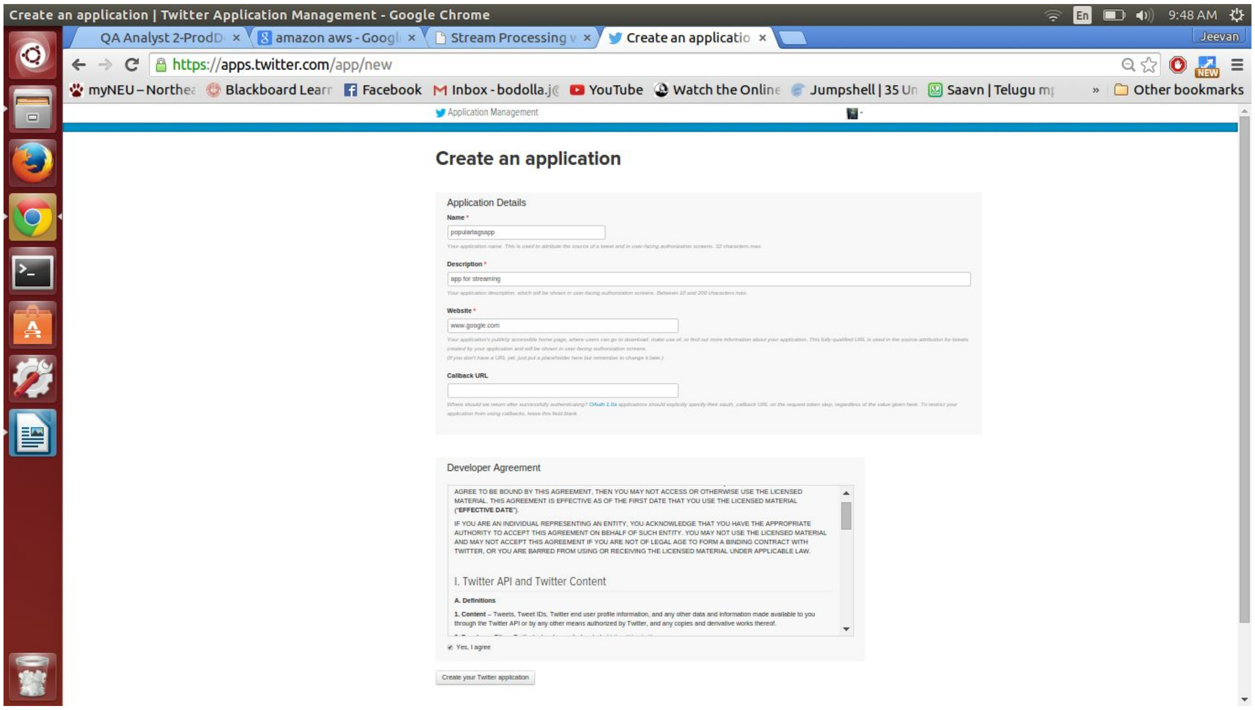
```
15/08/08 01:54:30 INFO TaskSetManager: Starting task 0.0 in stage 8.0 (TID 7, localhost, PROCESS_LOCAL, 1243 bytes)
15/08/08 01:54:30 INFO Executor: Running task 0.0 in stage 8.0 (TID 7)
15/08/08 01:54:30 INFO ShuffleBlockFetcherIterator: Getting 3 non-empty blocks out of 3 blocks
15/08/08 01:54:30 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
15/08/08 01:54:30 INFO MemoryStore: ensureFreeSpace(688) called with curMem=83498, maxMem=278302556
15/08/08 01:54:30 INFO MemoryStore: Block input-0-1438998870200 stored as bytes in memory (estimated size 688.0 B, free 265.3 MB)
15/08/08 01:54:30 INFO BlockManagerInfo: Added input-0-1438998870200 in memory on localhost:50524 (size: 688.0 B, free: 265.4 MB)
15/08/08 01:54:30 WARN BlockManager: Block input-0-1438998870200 replicated to only 0 peer(s) instead of 1 peers
15/08/08 01:54:30 INFO BlockGenerator: Pushed block input-0-1438998870200
15/08/08 01:54:30 INFO Executor: Finished task 0.0 in stage 8.0 (TID 7). 1067 bytes result sent to driver
15/08/08 01:54:30 INFO DAGScheduler: ResultStage 8 (print at MQTTWordCount.scala:111) finished in 0.031 s
15/08/08 01:54:30 INFO TaskSetManager: Finished task 0.0 in stage 8.0 (TID 7) in 39 ms on localhost (1/1)
15/08/08 01:54:30 INFO TaskSchedulerImpl: Removed TaskSet 8.0, whose tasks have all completed, from pool
15/08/08 01:54:30 INFO DAGScheduler: Job 4 finished: print at MQTTWordCount.scala:111, took 0.127595 s
-------------------------------------------
Time: 1438998870000 ms
-------------------------------------------
(mqtt,40)
(hello,40)
(streaming,40)
(spark,40)
(demo,40)
(for,40)

15/08/08 01:54:30 INFO JobScheduler: Finished job streaming job 1438998870000 ms.0 from job set of time 1438998870000 ms
15/08/08 01:54:30 INFO JobScheduler: Total delay: 0.469 s for time 1438998870000 ms (execution: 0.448 s)
15/08/08 01:54:30 INFO ShuffledRDD: Removing RDD 4 from persistence list
15/08/08 01:54:30 INFO MapPartitionsRDD: Removing RDD 3 from persistence list
15/08/08 01:54:30 INFO BlockManager: Removing RDD 4
15/08/08 01:54:30 INFO BlockManager: Removing RDD 3
15/08/08 01:54:30 INFO MapPartitionsRDD: Removing RDD 2 from persistence list
15/08/08 01:54:30 INFO BlockManager: Removing RDD 2
15/08/08 01:54:30 INFO BlockRDD: Removing RDD 1 from persistence list
15/08/08 01:54:30 INFO BlockManagerInfo: Removed broadcast_1_piece0 on localhost:50524 in memory (size: 1425.0 B, free: 265.4 MB)
15/08/08 01:54:30 INFO BlockManager: Removing RDD 1
15/08/08 01:54:30 INFO MQTTInputDStream: Removing blocks of RDD BlockRDD[1] at createStream at MQTTWordCount.scala:107 of time 1438998870000 ms
15/08/08 01:54:30 INFO ReceivedBlockTracker: Deleting batches ArrayBuffer()
15/08/08 01:54:30 INFO InputInfoTracker: remove old batch metadata:
15/08/08 01:54:30 INFO BlockManagerInfo: Removed broadcast_4_piece0 on localhost:50524 in memory (size: 1425.0 B, free: 265.4 MB)
15/08/08 01:54:30 INFO BlockManagerInfo: Removed broadcast_3_piece0 on localhost:50524 in memory (size: 1576.0 B, free: 265.4 MB)
15/08/08 01:54:30 INFO BlockManagerInfo: Removed broadcast_2_piece0 on localhost:50524 in memory (size: 1425.0 B, free: 265.4 MB)
15/08/08 01:54:30 INFO MemoryStore: ensureFreeSpace(802) called with curMem=68759, maxMem=278302556
15/08/08 01:54:30 INFO MemoryStore: Block input-0-1438998870400 stored as bytes in memory (estimated size 802.0 B, free 265.3 MB)
15/08/08 01:54:30 INFO BlockManagerInfo: Added input-0-1438998870400 in memory on localhost:50524 (size: 802.0 B, free: 265.4 MB)
15/08/08 01:54:30 WARN BlockManager: Block input-0-1438998870400 replicated to only 0 peer(s) instead of 1 peers
15/08/08 01:54:30 INFO BlockGenerator: Pushed block input-0-1438998870400
15/08/08 01:54:30 INFO MemoryStore: ensureFreeSpace(574) called with curMem=69561, maxMem=278302556
15/08/08 01:54:30 INFO MemoryStore: Block input-0-1438998870600 stored as bytes in memory (estimated size 574.0 B, free 265.3 MB)
15/08/08 01:54:30 INFO BlockManagerInfo: Added input-0-1438998870600 in memory on localhost:50524 (size: 574.0 B, free: 265.4 MB)
15/08/08 01:54:30 WARN BlockManager: Block input-0-1438998870600 replicated to only 0 peer(s) instead of 1 peers
15/08/08 01:54:30 INFO BlockGenerator: Pushed block input-0-1438998870600
15/08/08 01:54:31 INFO MemoryStore: ensureFreeSpace(764) called with curMem=70135, maxMem=278302556
15/08/08 01:54:31 INFO MemoryStore: Block input-0-1438998870800 stored as bytes in memory (estimated size 764.0 B, free 265.3 MB)
15/08/08 01:54:31 INFO BlockManagerInfo: Added input-0-1438998870800 in memory on localhost:50524 (size: 764.0 B, free: 265.4 MB)
15/08/08 01:54:31 WARN BlockManager: Block input-0-1438998870800 replicated to only 0 peer(s) instead of 1 peers
15/08/08 01:54:31 INFO BlockGenerator: Pushed block input-0-1438998870800
15/08/08 01:54:31 INFO MemoryStore: ensureFreeSpace(384) called with curMem=70899, maxMem=278302556
```

6. Steps for executing Twitter Popular Tags example (in Scala):

Twitter Credential Setup:

Since the whole exercise is based on twitter, it is necessary to configure authentication with a twitter account using a consumer key+secret pair and an access token+secret pair.

1. Open the twitter's application settings page (https://apps.twitter.com/) and create a new application by clicking on the "Create a new application" button and providing the required fields.



Create an application

Application Details

Name *

populartagsapp

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

app for streaming

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

www.google.com

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Developer Agreement

AGREE TO BE BOUND BY THIS AGREEMENT, THEN YOU MAY NOT ACCESS OR OTHERWISE USE THE LICENSED MATERIAL. THIS AGREEMENT IS EFFECTIVE AS OF THE FIRST DATE THAT YOU USE THE LICENSED MATERIAL ("**EFFECTIVE DATE**").

IF YOU ARE AN INDIVIDUAL REPRESENTING AN ENTITY, YOU ACKNOWLEDGE THAT YOU HAVE THE APPROPRIATE AUTHORITY TO ACCEPT THIS AGREEMENT ON BEHALF OF SUCH ENTITY. YOU MAY NOT USE THE LICENSED MATERIAL AND MAY NOT ACCEPT THIS AGREEMENT IF YOU ARE NOT OF LEGAL AGE TO FORM A BINDING CONTRACT WITH TWITTER, OR YOU ARE BARRED FROM USING OR RECEIVING THE LICENSED MATERIAL UNDER APPLICABLE LAW.

I. Twitter API and Twitter Content

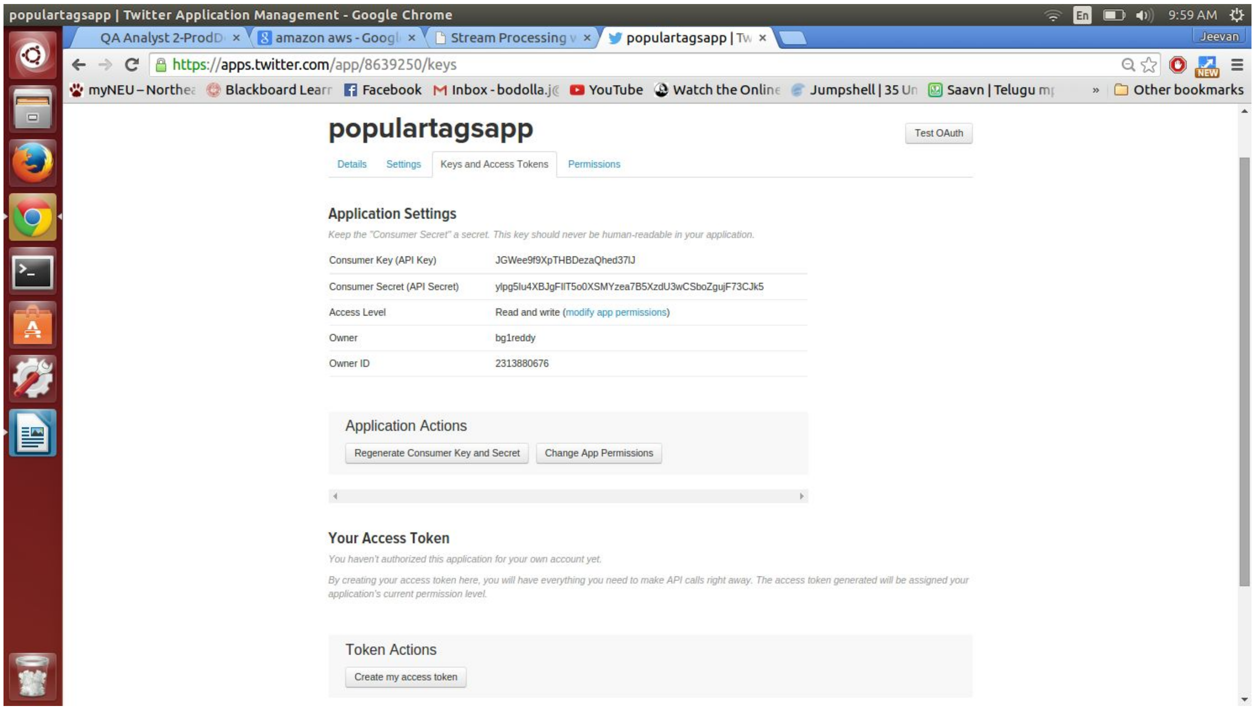
A. Definitions

1. Content – Tweets, Tweet IDs, Twitter end user profile information, and any other data and information made available to you through the Twitter API or by any other means authorized by Twitter, and any copies and derivative works thereof.

☑ Yes, I agree

Create your Twitter application

2. Once you create the application, click on the "Keys and Access Tokens" tab where you can see the API key and API secret that have been generated. Click on the "Create My Access Token" button at the bottom to generate the access token and access token secret.



populartagsapp | Twitter Application Management - Google Chrome

https://apps.twitter.com/app/8639250/keys

populartagsapp

Test OAuth

Details | Settings | Keys and Access Tokens | Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	JGWee9f9XpTHBDezaQhed37lJ
Consumer Secret (API Secret)	ylpg5Iu4XBJgFlIT5o0XSMYzea7B5XzdU3wCSboZgujF73CJk5
Access Level	Read and write (modify app permissions)
Owner	bg1reddy
Owner ID	2313880676

Application Actions

Regenerate Consumer Key and Secret | Change App Permissions

Your Access Token

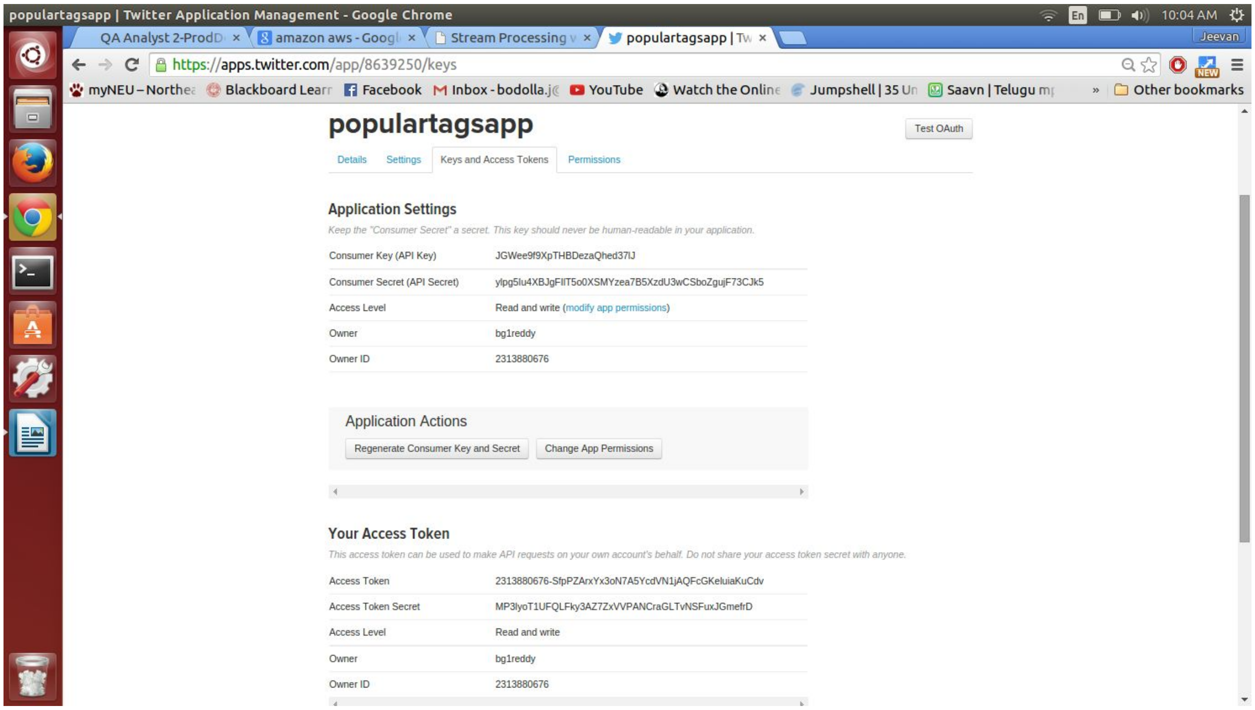
You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

Create my access token

3. Finally, we will see the API Key, API Secret, Access token and Access token secret we are going to use in the next steps.



populartagsapp | Twitter Application Management - Google Chrome

https://apps.twitter.com/app/8639250/keys

populartagsapp

Test OAuth

Details | Settings | Keys and Access Tokens | Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	JGWee9f9XpTHBDezaQhed37lJ
Consumer Secret (API Secret)	ylpg5Iu4XBJgFlIT5o0XSMYzea7B5XzdU3wCSboZgujF73CJk5
Access Level	Read and write (modify app permissions)
Owner	bg1reddy
Owner ID	2313880676

Application Actions

Regenerate Consumer Key and Secret | Change App Permissions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	2313880676-SfpPZArxYx3oN7A5YcdVN1jAQFcGKeluiaKuCdv
Access Token Secret	MP3lyoT1UFQLFky3AZ7ZxVVPANCraGLTvNSFuxJGmefrD
Access Level	Read and write
Owner	bg1reddy
Owner ID	2313880676

4. Now, SSH into the master node with the private key file used to launch the cluster by typing the following command:

```
$ ssh hadoop@ec2-52-1-104-201.compute-1.amazonaws.com -i ~/twitter.pem
```

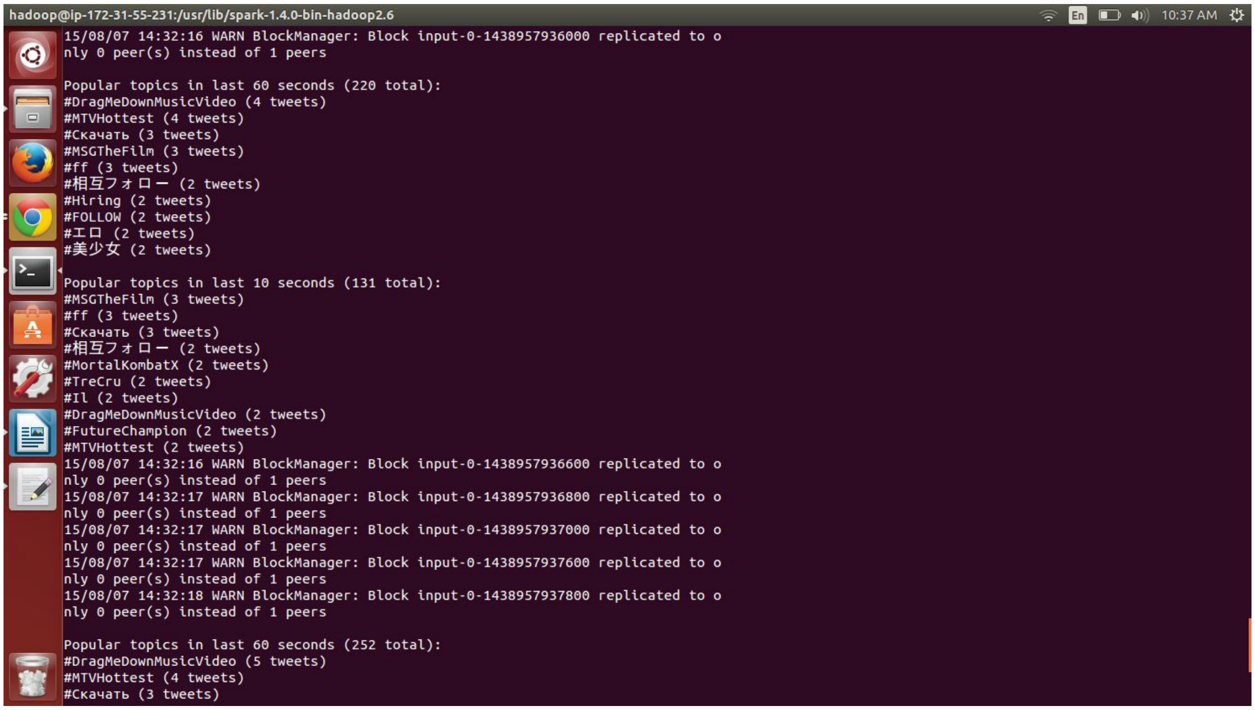
5. Install Spark 1.4.0 on the EMR cluster by typing the following commands:

```
$ cd /usr/lib/
$ sudo wget
http://psg.mtu.edu/pub/apache/spark/spark-1.4.0/spark-1.4.0-bin-hadoop2.6.tgz
$ sudo tar -zxvf spark-1.4.0-bin-hadoop2.6.tgz
$ cd spark-1.4.0-bin-hadoop2.6
```

6. Finally, compile the TwitterPopularTags class with the API Key, API Secret, Access token and Access token secret as arguments as shown below:

```
$ bin/run-example org.apache.spark.examples.streaming.TwitterPopularTags
JeCdbwdDuVpPzwRh4aLvJYKj2
6cEXsnpvcaonemZz9KJG059E2yOXWLFCnQax0t1rdq9FRcPwIq
2313880676-QhO7bT3tojJfurFZJjjpJqAOXtj7fdB6gJPLzgk
XVTtZu0sYtd7WqBemaDf8bOXSEhUkqfUEwJDXv1MoRFJf
```

and you will soon find the popular tweets over sliding 10 and 60 second window from a twitter stream being printed on the screen something like this:



```
15/08/07 14:32:16 WARN BlockManager: Block input-0-1438957936000 replicated to o
nly 0 peer(s) instead of 1 peers

Popular topics in last 60 seconds (220 total):
#DragMeDownMusicVideo (4 tweets)
#MTVHottest (4 tweets)
#Скачать (3 tweets)
#MSGTheFilm (3 tweets)
#ff (3 tweets)
#相互フォロー (2 tweets)
#Hiring (2 tweets)
#FOLLOW (2 tweets)
#エロ (2 tweets)
#美少女 (2 tweets)

Popular topics in last 10 seconds (131 total):
#MSGTheFilm (3 tweets)
#ff (3 tweets)
#Скачать (3 tweets)
#相互フォロー (2 tweets)
#MortalKombatX (2 tweets)
#TreCru (2 tweets)
#Il (2 tweets)
#DragMeDownMusicVideo (2 tweets)
#FutureChampion (2 tweets)
#MTVHottest (2 tweets)
15/08/07 14:32:16 WARN BlockManager: Block input-0-1438957936600 replicated to o
nly 0 peer(s) instead of 1 peers
15/08/07 14:32:17 WARN BlockManager: Block input-0-1438957936800 replicated to o
nly 0 peer(s) instead of 1 peers
15/08/07 14:32:17 WARN BlockManager: Block input-0-1438957937000 replicated to o
nly 0 peer(s) instead of 1 peers
15/08/07 14:32:17 WARN BlockManager: Block input-0-1438957937600 replicated to o
nly 0 peer(s) instead of 1 peers
15/08/07 14:32:18 WARN BlockManager: Block input-0-1438957937800 replicated to o
nly 0 peer(s) instead of 1 peers

Popular topics in last 60 seconds (252 total):
#DragMeDownMusicVideo (5 tweets)
#MTVHottest (4 tweets)
#Скачать (3 tweets)
```

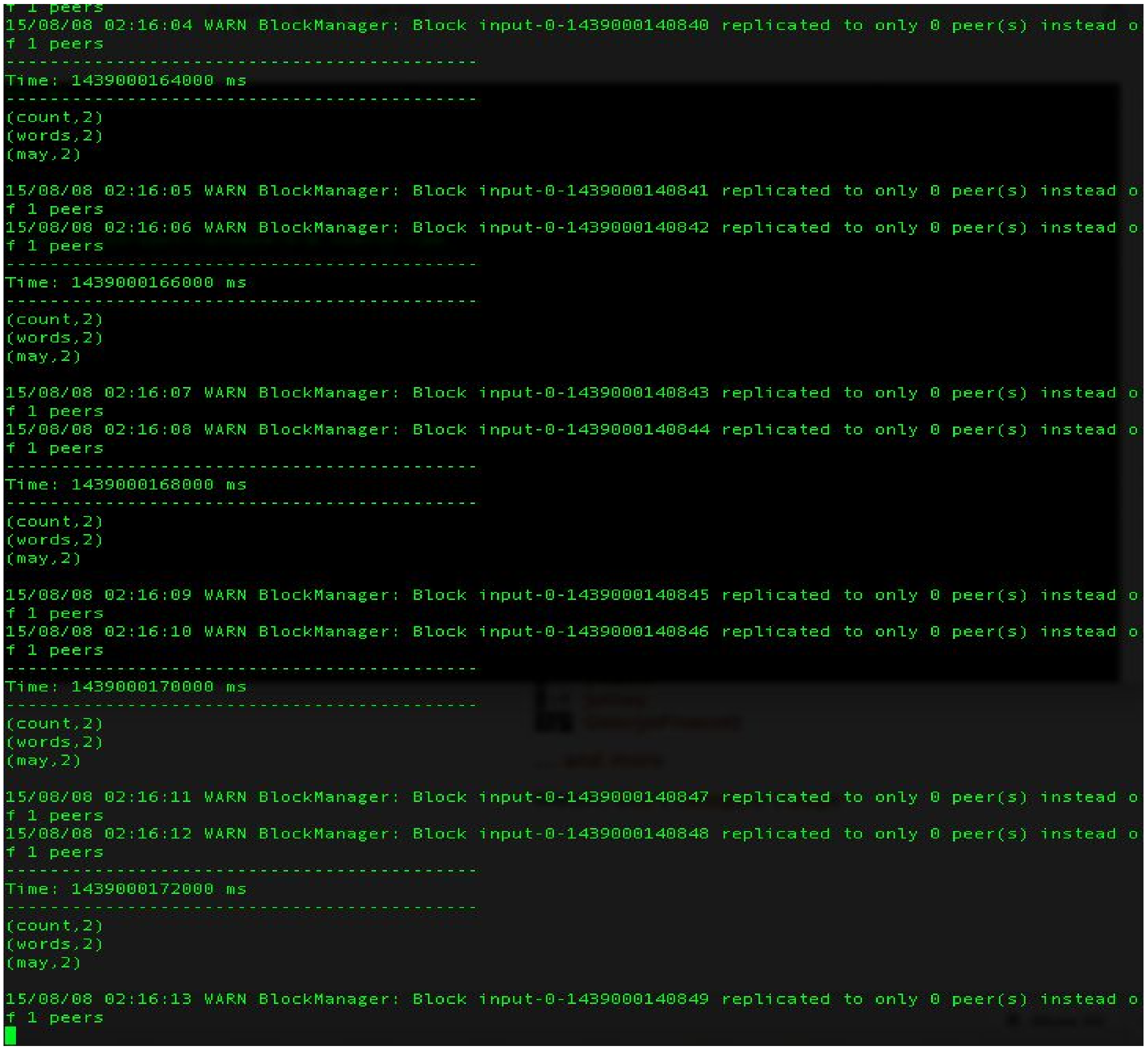
7. **Steps for executing ZeroMQ wordcount example (in Scala)**

- Install ZeroMQ by using the following command
    - $ apt-get install libzmq-dev

- Run the publisher by using the following command
    - $ bin/run-example org.apache.spark.examples.streaming.SimpleZeroMQPublisher tcp://127.0.1.1:1234 foo.ba

- Run the ZeroMQ example file as follows
    - $ bin/run-example org.apache.spark.examples.streaming.ZeroMQWordCount tcp://127.0.1.1:1234 foo

- The output will be something as below:



```
f 1 peers
15/08/08 02:16:04 WARN BlockManager: Block input-0-1439000140840 replicated to only 0 peer(s) instead o
f 1 peers
-------------------------------------------
Time: 1439000164000 ms
-------------------------------------------
(count,2)
(words,2)
(may,2)

15/08/08 02:16:05 WARN BlockManager: Block input-0-1439000140841 replicated to only 0 peer(s) instead o
f 1 peers
15/08/08 02:16:06 WARN BlockManager: Block input-0-1439000140842 replicated to only 0 peer(s) instead o
f 1 peers
-------------------------------------------
Time: 1439000166000 ms
-------------------------------------------
(count,2)
(words,2)
(may,2)

15/08/08 02:16:07 WARN BlockManager: Block input-0-1439000140843 replicated to only 0 peer(s) instead o
f 1 peers
15/08/08 02:16:08 WARN BlockManager: Block input-0-1439000140844 replicated to only 0 peer(s) instead o
f 1 peers
-------------------------------------------
Time: 1439000168000 ms
-------------------------------------------
(count,2)
(words,2)
(may,2)

15/08/08 02:16:09 WARN BlockManager: Block input-0-1439000140845 replicated to only 0 peer(s) instead o
f 1 peers
15/08/08 02:16:10 WARN BlockManager: Block input-0-1439000140846 replicated to only 0 peer(s) instead o
f 1 peers
-------------------------------------------
Time: 1439000170000 ms
-------------------------------------------
(count,2)
(words,2)
(may,2)

15/08/08 02:16:11 WARN BlockManager: Block input-0-1439000140847 replicated to only 0 peer(s) instead o
f 1 peers
15/08/08 02:16:12 WARN BlockManager: Block input-0-1439000140848 replicated to only 0 peer(s) instead o
f 1 peers
-------------------------------------------
Time: 1439000172000 ms
-------------------------------------------
(count,2)
(words,2)
(may,2)

15/08/08 02:16:13 WARN BlockManager: Block input-0-1439000140849 replicated to only 0 peer(s) instead o
f 1 peers
```

8. Steps for executing Kafka wordcount example (in Python)

- Prerequisite: You need to open at least **7** terminal windows connected to EMR
- Note- Kafka doesn’t work on Spark 1.4.1 so we installed Spark 1.4.0

- EMR Master node doesn’t have **wget** installed. so first we’ll have to install that in order to be able to download any other stuff

    - $ sudo yum -y install wgetv
    - $ wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie"http://download.oracle.com/otn-pub/java/jdk/7u67-b01/jdk-7u67-linux-x64.rpm

- Now we need to download Spark 1.4.0 as EMR comes with Spark 1.4.1
    - $ wget http://mirror.symnds.com/software/Apache/spark/spark-1.4.0/spark-1.4.0-bin-hadoop2.6.tgz
    - $ tar -xzf spark-1.4.0-bin-hadoop2.6.tgz

- Then we can download Kafka 2.11-0.8.2.1 from the mirror site using wget
    - $ wget http://apache.mirrors.tds.net/kafka/0.8.2.1/kafka_2.9.1-0.8.2.1.tgz
    - $ tar -xzf kafka_2.9.1-0.8.2.1.tgz
    - $ cd kafka_2.9.1-0.8.2.1

- Start Zookeeper by using the following command:

```
$ bin/zookeeper-server-start.sh config/zookeeper.properties
```

    Zookeeper starts at localhost: 2181

- Start Kafka broker in another terminal by using the following command:

```
$ cd kafka_2.9.1-0.8.2.1
$ bin/kakfa-server-start.sh config/server.properties
```

    KafkaBroker starts at localhost: 9092

- Create a Kafka Topic of your choice in another terminal by using the following command:

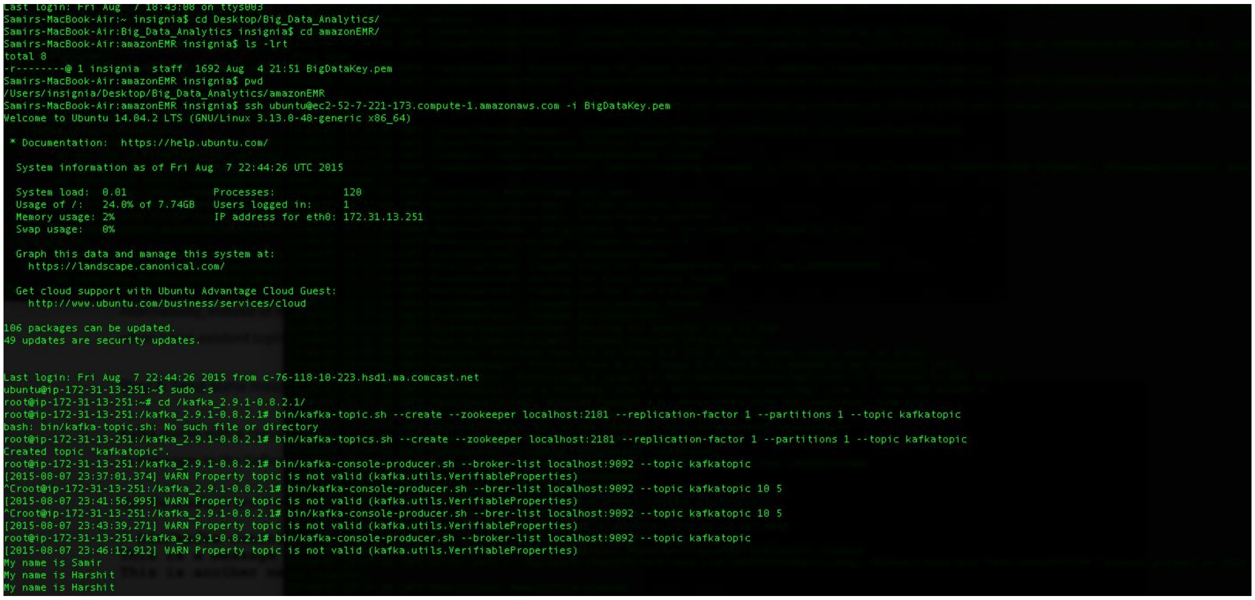
```
$ cd kafka_2.9.1-0.8.2.1
$ bin/kafka-topics.sh --create --zookeeper localhost:2181
--replication-factor 1 --partitions 1 --topic kafkatopic
```

This creates a topic by name Kafkatopic

- Now, start the producer in another terminal which produces data by using the following command:

```
$ cd kafka_2.9.1-0.8.2.1
$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic
kafkatopic
```

Type in anything you would like to send to the kafka broker in the command window

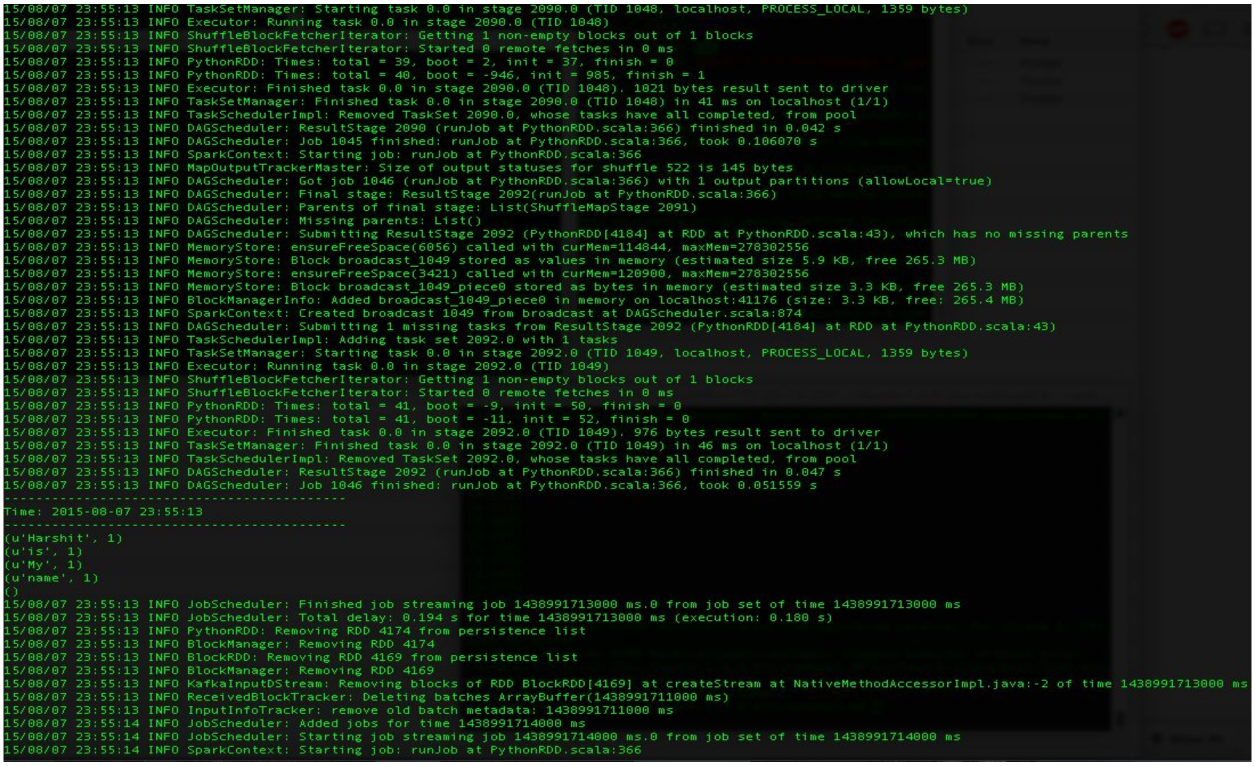


```
Last login: Fri Aug  7 18:43:08 on ttys003
Samirs-MacBook-Air:~ insignia$ cd Desktop/Big_Data_Analytics/
Samirs-MacBook-Air:Big_Data_Analytics insignia$ cd amazonEMR/
Samirs-MacBook-Air:amazonEMR insignia$ ls -lrt
total 8
-r--------@ 1 insignia  staff  1692 Aug  4 21:51 BigDataKey.pem
Samirs-MacBook-Air:amazonEMR insignia$ pwd
/Users/insignia/Desktop/Big_Data_Analytics/amazonEMR
Samirs-MacBook-Air:amazonEMR insignia$ ssh ubuntu@ec2-52-7-221-173.compute-1.amazonaws.com -i BigDataKey.pem
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-48-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Fri Aug  7 22:44:26 UTC 2015

  System load:  0.01              Processes:           120
  Usage of /:   24.0% of 7.74GB   Users logged in:     1
  Memory usage: 2%                IP address for eth0: 172.31.13.251
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

106 packages can be updated.
49 updates are security updates.


Last login: Fri Aug  7 22:44:26 2015 from c-76-118-10-223.hsd1.ma.comcast.net
ubuntu@ip-172-31-13-251:~$ sudo -s
root@ip-172-31-13-251:~# cd /kafka_2.9.1-0.8.2.1/
root@ip-172-31-13-251:/kafka_2.9.1-0.8.2.1# bin/kafka-topic.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic kafkatopic
bash: bin/kafka-topic.sh: No such file or directory
root@ip-172-31-13-251:/kafka_2.9.1-0.8.2.1# bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic kafkatopic
Created topic "kafkatopic".
root@ip-172-31-13-251:/kafka_2.9.1-0.8.2.1# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafkatopic
[2015-08-07 23:37:01,374] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
^Croot@ip-172-31-13-251:/kafka_2.9.1-0.8.2.1# bin/kafka-console-producer.sh --brer-list localhost:9092 --topic kafkatopic 10 5
[2015-08-07 23:41:56,995] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
^Croot@ip-172-31-13-251:/kafka_2.9.1-0.8.2.1# bin/kafka-console-producer.sh --brer-list localhost:9092 --topic kafkatopic 10 5
[2015-08-07 23:43:39,271] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
root@ip-172-31-13-251:/kafka_2.9.1-0.8.2.1# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafkatopic
[2015-08-07 23:46:12,912] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
My name is Samir
My name is Harshit
My name is Harshit
```

- Run the wordcount example in another terminal in spark 1.4.0 directory by using the following command:

```
$ bin/spark-submit --jars external/kafka-assembly/target/scala-*/*.jar
examples/src/main/python/streaming/kafka_wordcount.py
localhost:2181 kafkatopic
```

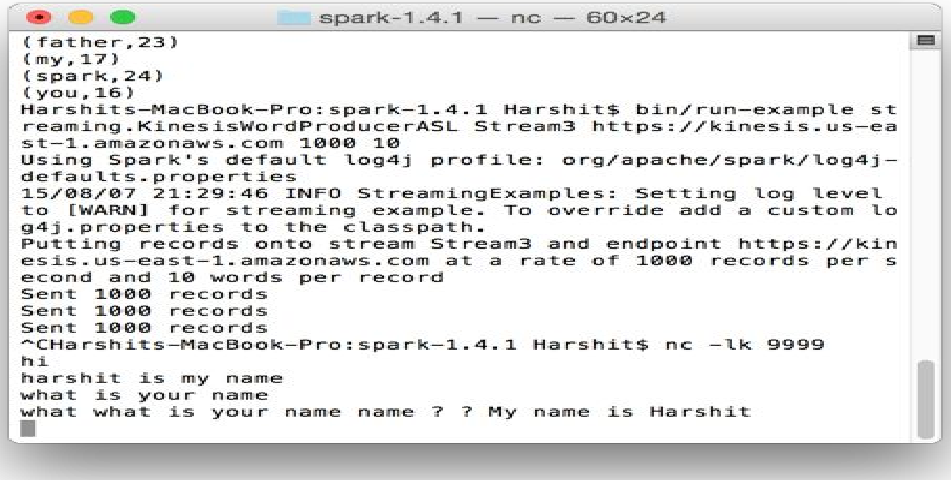
The output would be something like this:



```
15/08/07 23:55:13 INFO TaskSetManager: Starting task 0.0 in stage 2090.0 (TID 1048, localhost, PROCESS_LOCAL, 1359 bytes)
15/08/07 23:55:13 INFO Executor: Running task 0.0 in stage 2090.0 (TID 1048)
15/08/07 23:55:13 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
15/08/07 23:55:13 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
15/08/07 23:55:13 INFO PythonRDD: Times: total = 39, boot = 2, init = 37, finish = 0
15/08/07 23:55:13 INFO PythonRDD: Times: total = 40, boot = -946, init = 985, finish = 1
15/08/07 23:55:13 INFO Executor: Finished task 0.0 in stage 2090.0 (TID 1048). 1021 bytes result sent to driver
15/08/07 23:55:13 INFO TaskSetManager: Finished task 0.0 in stage 2090.0 (TID 1048) in 41 ms on localhost (1/1)
15/08/07 23:55:13 INFO TaskSchedulerImpl: Removed TaskSet 2090.0, whose tasks have all completed, from pool
15/08/07 23:55:13 INFO DAGScheduler: ResultStage 2090 (runJob at PythonRDD.scala:366) finished in 0.042 s
15/08/07 23:55:13 INFO DAGScheduler: Job 1045 finished: runJob at PythonRDD.scala:366, took 0.106070 s
15/08/07 23:55:13 INFO SparkContext: Starting job: runJob at PythonRDD.scala:366
15/08/07 23:55:13 INFO MapOutputTrackerMaster: Size of output statuses for shuffle 522 is 145 bytes
15/08/07 23:55:13 INFO DAGScheduler: Got job 1046 (runJob at PythonRDD.scala:366) with 1 output partitions (allowLocal=true)
15/08/07 23:55:13 INFO DAGScheduler: Final stage: ResultStage 2092(runJob at PythonRDD.scala:366)
15/08/07 23:55:13 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 2091)
15/08/07 23:55:13 INFO DAGScheduler: Missing parents: List()
15/08/07 23:55:13 INFO DAGScheduler: Submitting ResultStage 2092 (PythonRDD[4184] at RDD at PythonRDD.scala:43), which has no missing parents
15/08/07 23:55:13 INFO MemoryStore: ensureFreeSpace(6056) called with curMem=114844, maxMem=278302556
15/08/07 23:55:13 INFO MemoryStore: Block broadcast_1049 stored as values in memory (estimated size 5.9 KB, free 265.3 MB)
15/08/07 23:55:13 INFO MemoryStore: ensureFreeSpace(3421) called with curMem=120900, maxMem=278302556
15/08/07 23:55:13 INFO MemoryStore: Block broadcast_1049_piece0 stored as bytes in memory (estimated size 3.3 KB, free 265.3 MB)
15/08/07 23:55:13 INFO BlockManagerInfo: Added broadcast_1049_piece0 in memory on localhost:41176 (size: 3.3 KB, free: 265.4 MB)
15/08/07 23:55:13 INFO SparkContext: Created broadcast 1049 from broadcast at DAGScheduler.scala:874
15/08/07 23:55:13 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 2092 (PythonRDD[4184] at RDD at PythonRDD.scala:43)
15/08/07 23:55:13 INFO TaskSchedulerImpl: Adding task set 2092.0 with 1 tasks
15/08/07 23:55:13 INFO TaskSetManager: Starting task 0.0 in stage 2092.0 (TID 1049, localhost, PROCESS_LOCAL, 1359 bytes)
15/08/07 23:55:13 INFO Executor: Running task 0.0 in stage 2092.0 (TID 1049)
15/08/07 23:55:13 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
15/08/07 23:55:13 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
15/08/07 23:55:13 INFO PythonRDD: Times: total = 41, boot = -9, init = 50, finish = 0
15/08/07 23:55:13 INFO PythonRDD: Times: total = 41, boot = -11, init = 52, finish = 0
15/08/07 23:55:13 INFO Executor: Finished task 0.0 in stage 2092.0 (TID 1049). 976 bytes result sent to driver
15/08/07 23:55:13 INFO TaskSetManager: Finished task 0.0 in stage 2092.0 (TID 1049) in 46 ms on localhost (1/1)
15/08/07 23:55:13 INFO TaskSchedulerImpl: Removed TaskSet 2092.0, whose tasks have all completed, from pool
15/08/07 23:55:13 INFO DAGScheduler: ResultStage 2092 (runJob at PythonRDD.scala:366) finished in 0.047 s
15/08/07 23:55:13 INFO DAGScheduler: Job 1046 finished: runJob at PythonRDD.scala:366, took 0.051559 s
-------------------------------------------
Time: 2015-08-07 23:55:13
-------------------------------------------
(u'Harshit', 1)
(u'is', 1)
(u'My', 1)
(u'name', 1)
()
15/08/07 23:55:13 INFO JobScheduler: Finished job streaming job 1438991713000 ms.0 from job set of time 1438991713000 ms
15/08/07 23:55:13 INFO JobScheduler: Total delay: 0.194 s for time 1438991713000 ms (execution: 0.180 s)
15/08/07 23:55:13 INFO PythonRDD: Removing RDD 4174 from persistence list
15/08/07 23:55:13 INFO BlockManager: Removing RDD 4174
15/08/07 23:55:13 INFO BlockRDD: Removing RDD 4169 from persistence list
15/08/07 23:55:13 INFO BlockManager: Removing RDD 4169
15/08/07 23:55:13 INFO KafkaInputDStream: Removing blocks of RDD BlockRDD[4169] at createStream at NativeMethodAccessorImpl.java:-2 of time 1438991713000 ms
15/08/07 23:55:13 INFO ReceivedBlockTracker: Deleting batches ArrayBuffer(1438991711000 ms)
15/08/07 23:55:13 INFO InputInfoTracker: remove old batch metadata: 1438991711000 ms
15/08/07 23:55:14 INFO JobScheduler: Added jobs for time 1438991714000 ms
15/08/07 23:55:14 INFO JobScheduler: Starting job streaming job 1438991714000 ms.0 from job set of time 1438991714000 ms
15/08/07 23:55:14 INFO SparkContext: Starting job: runJob at PythonRDD.scala:366
```

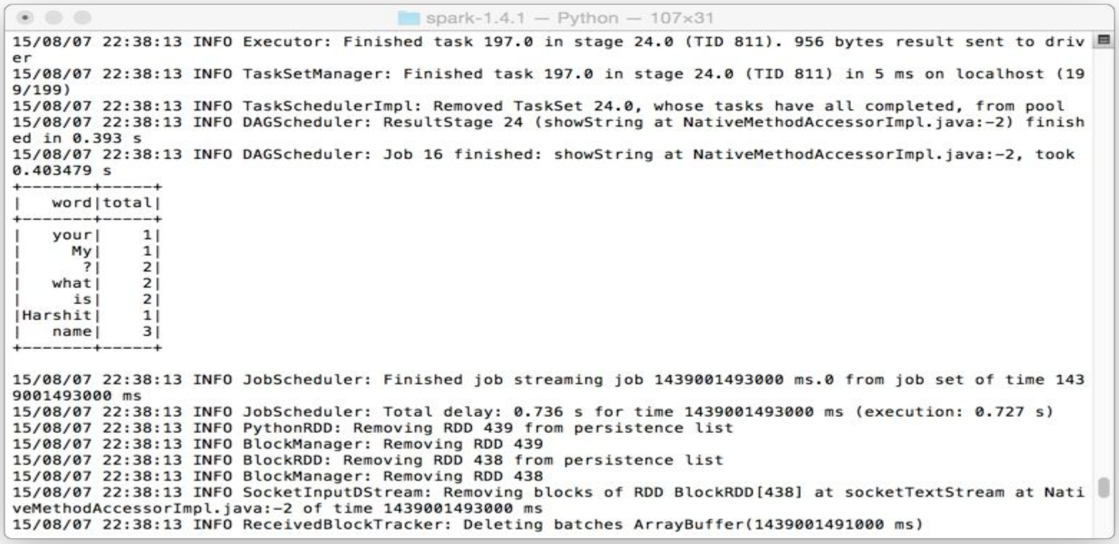
9. Steps for executing SQL Network Wordcount example (in Python)

- We need to start a Netcat server first in order to count the words over the network
  - $ nc -lk 9999



```
(father,23)
(my,17)
(spark,24)
(you,16)
Harshits-MacBook-Pro:spark-1.4.1 Harshit$ bin/run-example st
reaming.KinesisWordProducerASL Stream3 https://kinesis.us-ea
st-1.amazonaws.com 1000 10
Using Spark's default log4j profile: org/apache/spark/log4j-
defaults.properties
15/08/07 21:29:46 INFO StreamingExamples: Setting log level
to [WARN] for streaming example. To override add a custom lo
g4j.properties to the classpath.
Putting records onto stream Stream3 and endpoint https://kin
esis.us-east-1.amazonaws.com at a rate of 1000 records per s
econd and 10 words per record
Sent 1000 records
Sent 1000 records
Sent 1000 records
^CHarshits-MacBook-Pro:spark-1.4.1 Harshit$ nc -lk 9999
hi
harshit is my name
what is your name
what what is your name name ? ? My name is Harshit
```

- Then we start a listener which count the number of words
    - $ bin/spark-submit examples/src/main/python/streaming/sql_network_wordcount.py localhost 9999



```
15/08/07 22:38:13 INFO Executor: Finished task 197.0 in stage 24.0 (TID 811). 956 bytes result sent to driv
er
15/08/07 22:38:13 INFO TaskSetManager: Finished task 197.0 in stage 24.0 (TID 811) in 5 ms on localhost (19
9/199)
15/08/07 22:38:13 INFO TaskSchedulerImpl: Removed TaskSet 24.0, whose tasks have all completed, from pool
15/08/07 22:38:13 INFO DAGScheduler: ResultStage 24 (showString at NativeMethodAccessorImpl.java:-2) finish
ed in 0.393 s
15/08/07 22:38:13 INFO DAGScheduler: Job 16 finished: showString at NativeMethodAccessorImpl.java:-2, took
0.403479 s
+-------+-----+
|   word|total|
+-------+-----+
|   your|    1|
|     My|    1|
|      ?|    2|
|   what|    2|
|     is|    2|
|Harshit|    1|
|   name|    3|
+-------+-----+

15/08/07 22:38:13 INFO JobScheduler: Finished job streaming job 1439001493000 ms.0 from job set of time 143
9001493000 ms
15/08/07 22:38:13 INFO JobScheduler: Total delay: 0.736 s for time 1439001493000 ms (execution: 0.727 s)
15/08/07 22:38:13 INFO PythonRDD: Removing RDD 439 from persistence list
15/08/07 22:38:13 INFO BlockManager: Removing RDD 439
15/08/07 22:38:13 INFO BlockRDD: Removing RDD 438 from persistence list
15/08/07 22:38:13 INFO BlockManager: Removing RDD 438
15/08/07 22:38:13 INFO SocketInputDStream: Removing blocks of RDD BlockRDD[438] at socketTextStream at Nati
veMethodAccessorImpl.java:-2 of time 1439001493000 ms
15/08/07 22:38:13 INFO ReceivedBlockTracker: Deleting batches ArrayBuffer(143901491000 ms)
```

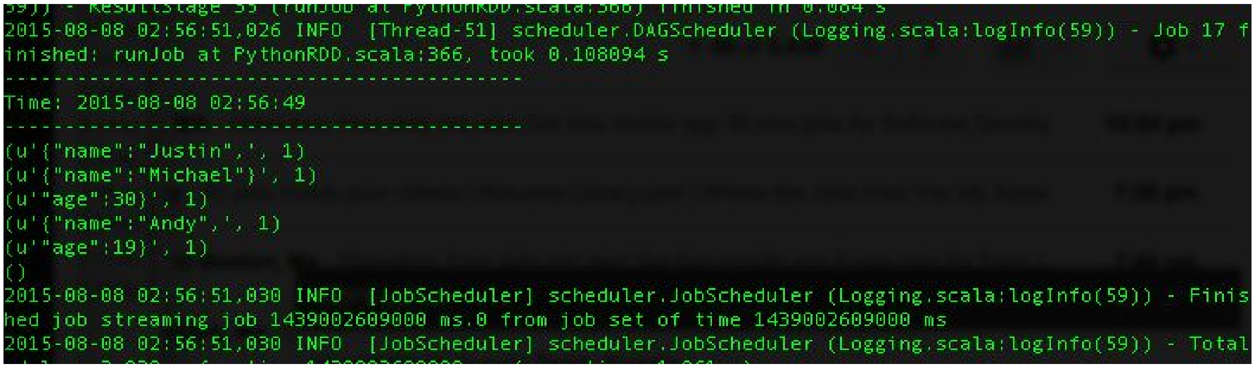
10. Steps for executing HDFS wordcount example (in Python)

- Change the directory to hadoop and create a folder called input in the hadoop folder as follows
    - $ cd /usr/lib/hadoop

- Make a directory in HDFS
    - $ bin/hadoop fs -mkdir /input

- Run the example from the spark directory with the hdfs directory as input
    - $ bin/spark-submit examples/src/main/python/streaming/hdfs_wordcount.py hdfs:///input

- Put a file into the hdfs directory for wordcount
    - $ bin/hadoop fs -put /usr/lib/spark/examples/src/main/resources/kv1.txt /input



```
59)) - ResultStage 35 (runJob at PythonRDD.scala:366) finished in 0.084 s
2015-08-08 02:56:51,026 INFO  [Thread-51] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - Job 17 f
inished: runJob at PythonRDD.scala:366, took 0.108094 s
-------------------------------------------
Time: 2015-08-08 02:56:49
-------------------------------------------
(u'{"name":"Justin",', 1)
(u'{"name":"Michael"}', 1)
(u'"age":30}', 1)
(u'{"name":"Andy",', 1)
(u'"age":19}', 1)
()
2015-08-08 02:56:51,030 INFO  [JobScheduler] scheduler.JobScheduler (Logging.scala:logInfo(59)) - Finis
hed job streaming job 1439002609000 ms.0 from job set of time 1439002609000 ms
2015-08-08 02:56:51,030 INFO  [JobScheduler] scheduler.JobScheduler (Logging.scala:logInfo(59)) - Total
```