Assignment 2 - Group 7

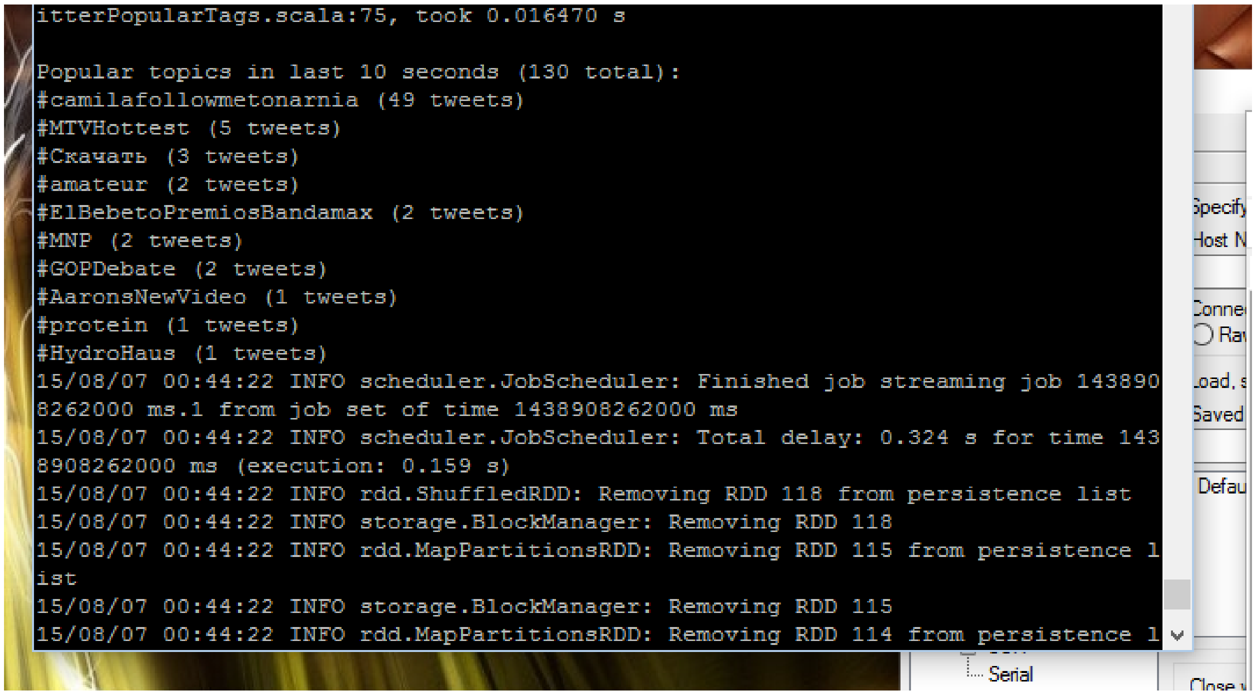
1. TwitterPopularTags

We implemented this example in emr 3.8.

1. SSH into the instance and go into spark directory.
2. Then run the following command
   bin/run-example org.apache.spark.examples.streaming.TwitterPopularTags `<consumerKey> <consumerSecret> <accessToken> <accessTokenSecret>`

   Put your own twitter developer account credentials into above command

3. Hit ctrl+c to abort and see the output.



```
itterPopularTags.scala:75, took 0.016470 s

Popular topics in last 10 seconds (130 total):
#camilafollowmetonarnia (49 tweets)
#MTVHottest (5 tweets)
#Скачать (3 tweets)
#amateur (2 tweets)
#ElBebetoPremiosBandamax (2 tweets)
#MNP (2 tweets)
#GOPDebate (2 tweets)
#AaronsNewVideo (1 tweets)
#protein (1 tweets)
#HydroHaus (1 tweets)
15/08/07 00:44:22 INFO scheduler.JobScheduler: Finished job streaming job 143890
8262000 ms.1 from job set of time 1438908262000 ms
15/08/07 00:44:22 INFO scheduler.JobScheduler: Total delay: 0.324 s for time 143
8908262000 ms (execution: 0.159 s)
15/08/07 00:44:22 INFO rdd.ShuffledRDD: Removing RDD 118 from persistence list
15/08/07 00:44:22 INFO storage.BlockManager: Removing RDD 118
15/08/07 00:44:22 INFO rdd.MapPartitionsRDD: Removing RDD 115 from persistence l
ist
15/08/07 00:44:22 INFO storage.BlockManager: Removing RDD 115
15/08/07 00:44:22 INFO rdd.MapPartitionsRDD: Removing RDD 114 from persistence l
```

2. Kinesis

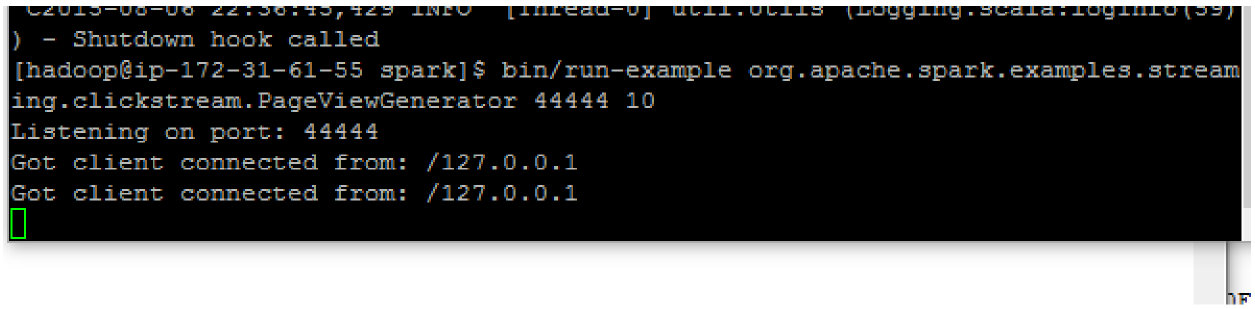
Clickstream analysis.

This is a pretty straightforward example.

1. Launch EMR 4.0 with spark 1.4.1.
2. SSH into your instance.
3. Locate the spark folder which is in /usr/lib/spark
4. Run the following command

bin/run-example org.apache.spark.examples.streaming.clickstream.PageViewGenerator 44444 10

It will start the Page View generator



```
) - Shutdown hook called
[hadoop@ip-172-31-61-55 spark]$ bin/run-example org.apache.spark.examples.stream
ing.clickstream.PageViewGenerator 44444 10
Listening on port: 44444
Got client connected from: /127.0.0.1
Got client connected from: /127.0.0.1
```

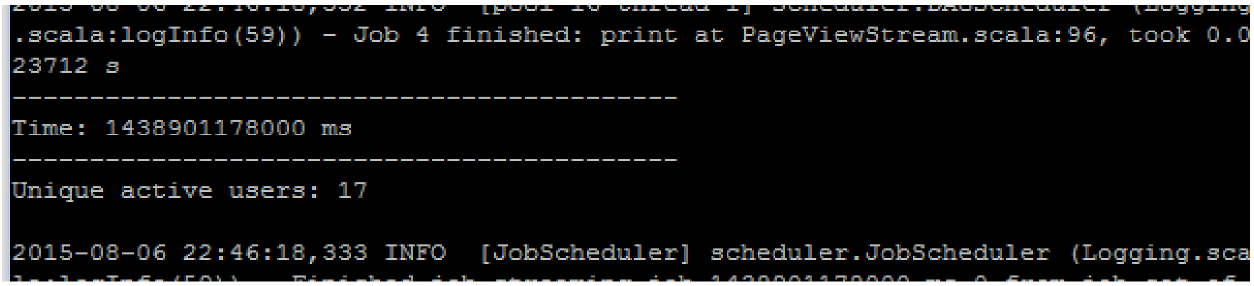
5. Open another terminal(Duplicate session)
6. Run the following command

org.apache.spark.examples.streaming.clickstream.PageViewStream errorRatePerZipCode localhost 44444

(instead of errorRateperZipCode you can give any other argument out of the following:-

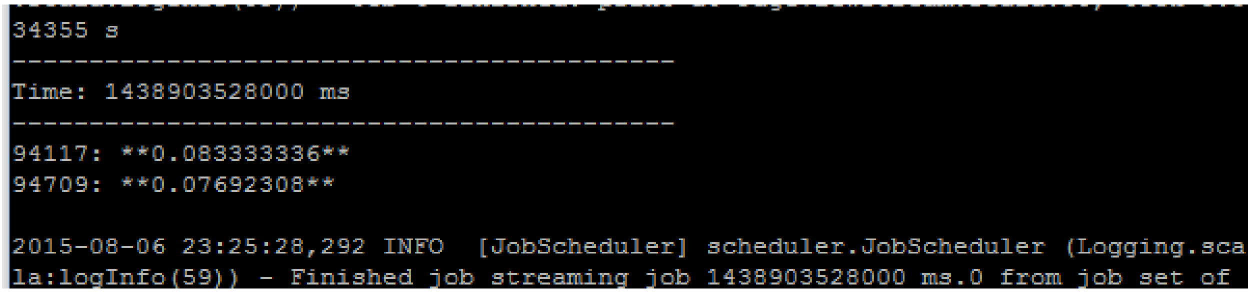
pageCounts, slidingPageCounts,errorRatePerZipCode, activeUserCount, popularUsersSeen.

For the activeUserCount following is the output



```
.scala:logInfo(59)) - Job 4 finished: print at PageViewStream.scala:96, took 0.0
23712 s
-------------------------------------------
Time: 1438901178000 ms
-------------------------------------------
Unique active users: 17

2015-08-06 22:46:18,333 INFO  [JobScheduler] scheduler.JobScheduler (Logging.sca
```

For errorRatePerZipCode following is the output



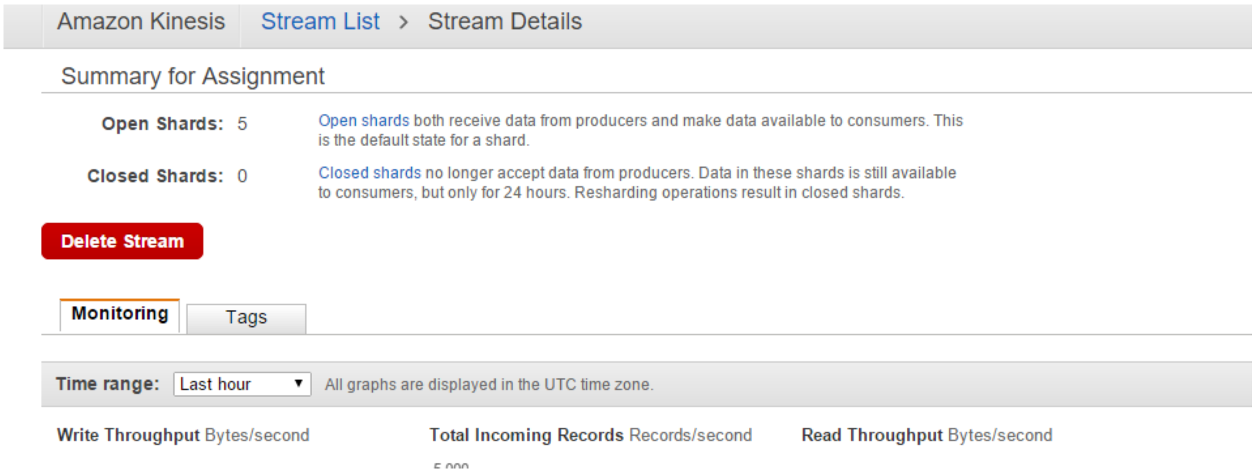
```
34355 s
-------------------------------------------
Time: 1438903528000 ms
-------------------------------------------
94117: **0.083333336**
94709: **0.07692308**

2015-08-06 23:25:28,292 INFO  [JobScheduler] scheduler.JobScheduler (Logging.sca
la:logInfo(59)) - Finished job streaming job 1438903528000 ms.0 from job set of
```

(You need to hit CTRL+C to stop the logs and see the output.

Using Amazon Kinesis:-

We also tried using amazon kinesis with spark on emr 4.0.

1. Sign in into AWS console. Start Amazon Kinesis. Create Stream with any name eg 'Assignment' with any number of shards (I have taken 5).



Summary for Assignment

Open Shards: 5 — Open shards both receive data from producers and make data available to consumers. This is the default state for a shard.

Closed Shards: 0 — Closed shards no longer accept data from producers. Data in these shards is still available to consumers, but only for 24 hours. Resharding operations result in closed shards.

Time range: Last hour — All graphs are displayed in the UTC time zone.

Write Throughput Bytes/second   Total Incoming Records Records/second   Read Throughput Bytes/second

2. Download the source code of spark and SCP it into emr instance. Untar the tar.gz file and go into spark folder.(we wont be using inbuilt spark in this example)
3. Build the spark with following command

```
build/mvn -Pkinesis-asl -DskipTests clean package
```

4. Set the environment variables AWS_ACCESS_KEY_ID and AWS_SECRET_KEY.

   export AWS_ACCESS_KEY_ID=<your-access-key>

   export AWS_SECRET_KEY=<your-secret-key>

5. From spark root directory enter following command
   bin/run-example streaming.KinesisWordCountASL [Kinesis app name] [Kinesis stream name] [endpoint URL]

   (give any appname, give stream name as Assignment and endpoint URL can be found at http://docs.aws.amazon.com/general/latest/gr/rande.html#ak_region)(for me the url was kinesis.us-east-1.amazonaws.com since my region was us-east-1)

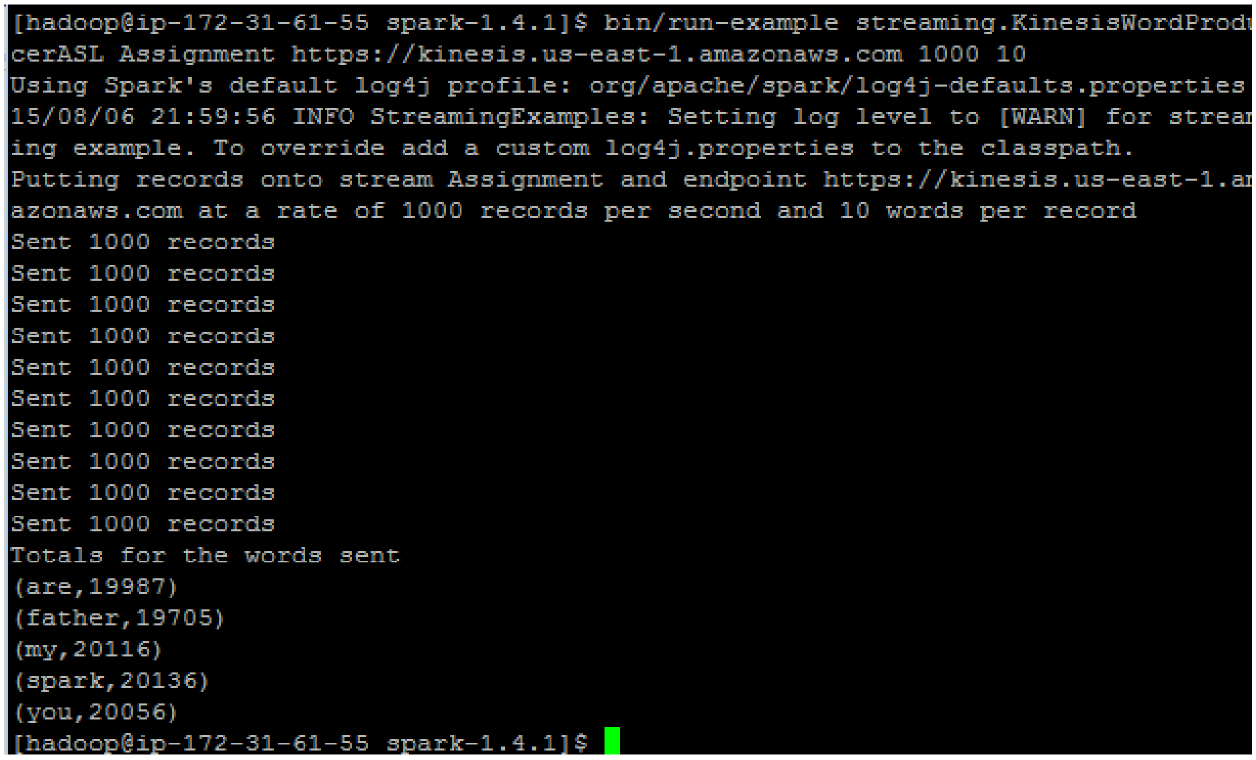
   (Also the user needs to have permission for Amazon Kinesis, Amazon DynamoDB and Amazon CloudWatch. However you can give only 2 permissions to one user. So its better to give Administrator Access to the user. Do this from Security Credentials in aws console. Then select policies.)

   It waits to receive word stream from producer.

6. Then run the following command

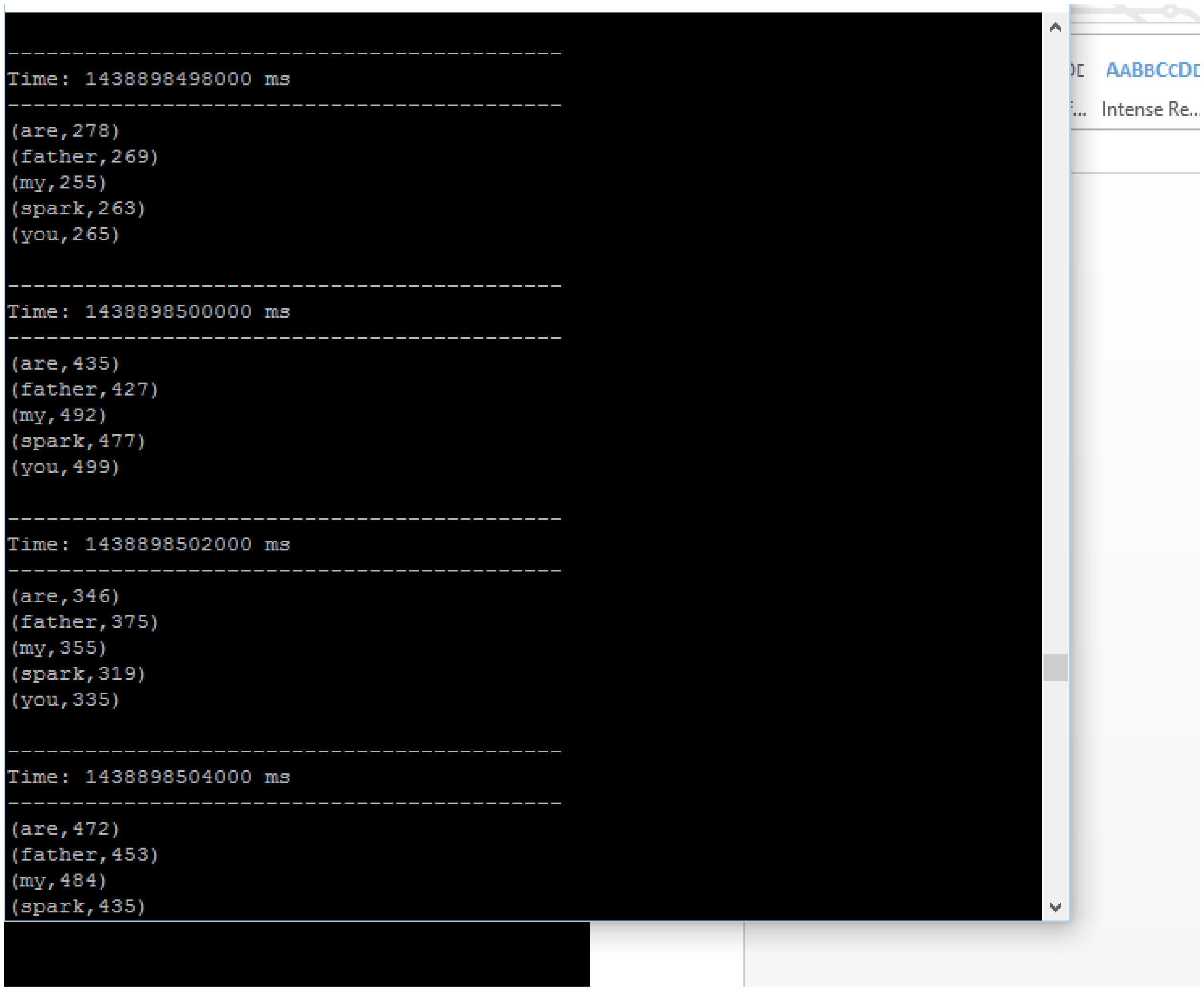
bin/run-example streaming.KinesisWordProducerASL [Kinesis stream name] [endpoint URL] 1000 10

This will produce the kinesis word stream as follows.



```
[hadoop@ip-172-31-61-55 spark-1.4.1]$ bin/run-example streaming.KinesisWordProdu
cerASL Assignment https://kinesis.us-east-1.amazonaws.com 1000 10
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
15/08/06 21:59:56 INFO StreamingExamples: Setting log level to [WARN] for stream
ing example. To override add a custom log4j.properties to the classpath.
Putting records onto stream Assignment and endpoint https://kinesis.us-east-1.am
azonaws.com at a rate of 1000 records per second and 10 words per record
Sent 1000 records
Sent 1000 records
Sent 1000 records
Sent 1000 records
Sent 1000 records
Sent 1000 records
Sent 1000 records
Sent 1000 records
Sent 1000 records
Sent 1000 records
Totals for the words sent
(are,19987)
(father,19705)
(my,20116)
(spark,20136)
(you,20056)
[hadoop@ip-172-31-61-55 spark-1.4.1]$
```

And then in the previously made terminal you can see word count happening.(you need to hit ctrl+c to see the output)



```
-------------------------------------------
Time: 1438898498000 ms
-------------------------------------------
(are,278)
(father,269)
(my,255)
(spark,263)
(you,265)

-------------------------------------------
Time: 1438898500000 ms
-------------------------------------------
(are,435)
(father,427)
(my,492)
(spark,477)
(you,499)

-------------------------------------------
Time: 1438898502000 ms
-------------------------------------------
(are,346)
(father,375)
(my,355)
(spark,319)
(you,335)

-------------------------------------------
Time: 1438898504000 ms
-------------------------------------------
(are,472)
(father,453)
(my,484)
(spark,435)
```

3. *Kafka(Scala)*

1. Since Kafka doesn’t run on spark 1.4.1, we started emr 4.0 and installed spark 1.4.0 on the instance. We SCPd the tar.gz file into emr and extracted it.

2. Then SCP kafka.tgz file into the instance and put the command tar –xvf <name of kafka.tgz>.

3. Go to the root directory of kafka.

4. First we need to start the zookeeper so run following command
   ```
   bin/zookeeper-server-start.sh config/zookeeper.properties
   ```

5. Open another terminal and then start the kafka broker
   ```
   bin/kafka-server-start.sh config/server.properties
   ```

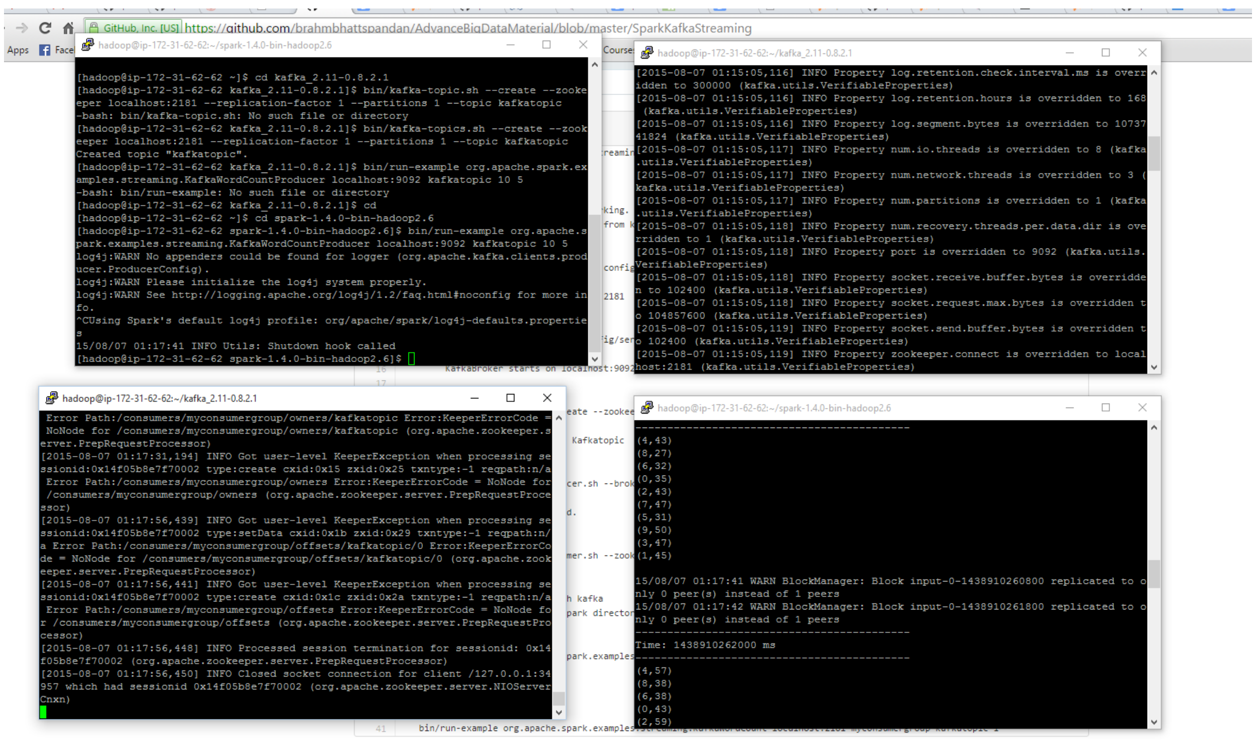
6. Open another terminal and create a kafka topic and start it
   ```
   bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic kafkatopic
   ```

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

7. Now run the producer. For that go into the spark directory
   ```
   bin/run-example org.apache.spark.examples.streaming.KafkaWordCountProducer localhost:9092 kafkatopic 10 5
   ```

8. Start another terminal and run the word count example
   ```
   bin/run-example org.apache.spark.examples.streaming.KafkaWordCount localhost:2181 myconsumergroup kafkatopic 1
   ```

You will have 4 terminals open by the end and you will see the output in the fourth terminal where it counts the word(in our case numbers from 1-10).


GitHub, Inc. [US] https://github.com/brahmbhattspandan/AdvanceBigDataMaterial/blob/master/SparkKafkaStreaming
Apps
Face
Course
hadoop@ip-172-31-62-62:~/spark-1.4.0-bin-hadoop2.6
hadoop@ip-172-31-62-62:~/kafka_2.11-0.8.2.1
hadoop@ip-172-31-62-62:~/kafka_2.11-0.8.2.1
hadoop@ip-172-31-62-62:~/spark-1.4.0-bin-hadoop2.6

```
[hadoop@ip-172-31-62-62 ~]$ cd kafka_2.11-0.8.2.1
[hadoop@ip-172-31-62-62 kafka_2.11-0.8.2.1]$ bin/kafka-topic.sh --create --zooke
eper localhost:2181 --replication-factor 1 --partitions 1 --topic kafkatopic
-bash: bin/kafka-topic.sh: No such file or directory
[hadoop@ip-172-31-62-62 kafka_2.11-0.8.2.1]$ bin/kafka-topics.sh --create --zook
eeper localhost:2181 --replication-factor 1 --partitions 1 --topic kafkatopic
Created topic "kafkatopic".
[hadoop@ip-172-31-62-62 kafka_2.11-0.8.2.1]$ bin/run-example org.apache.spark.ex
amples.streaming.KafkaWordCountProducer localhost:9092 kafkatopic 10 5
-bash: bin/run-example: No such file or directory
[hadoop@ip-172-31-62-62 kafka_2.11-0.8.2.1]$ cd
[hadoop@ip-172-31-62-62 ~]$ cd spark-1.4.0-bin-hadoop2.6
[hadoop@ip-172-31-62-62 spark-1.4.0-bin-hadoop2.6]$ bin/run-example org.apache.s
park.examples.streaming.KafkaWordCountProducer localhost:9092 kafkatopic 10 5
log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.prod
ucer.ProducerConfig).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more in
fo.
^CUsing Spark's default log4j profile: org/apache/spark/log4j-defaults.propertie
s
15/08/07 01:17:41 INFO Utils: Shutdown hook called
[hadoop@ip-172-31-62-62 spark-1.4.0-bin-hadoop2.6]$
```

```
[2015-08-07 01:15:05,116] INFO Property log.retention.check.interval.ms is overr
idden to 300000 (kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,116] INFO Property log.retention.hours is overridden to 168
 (kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,116] INFO Property log.segment.bytes is overridden to 10737
41824 (kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,117] INFO Property num.io.threads is overridden to 8 (kafka
.utils.VerifiableProperties)
[2015-08-07 01:15:05,117] INFO Property num.network.threads is overridden to 3 (
kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,117] INFO Property num.partitions is overridden to 1 (kafka
.utils.VerifiableProperties)
[2015-08-07 01:15:05,118] INFO Property num.recovery.threads.per.data.dir is ove
rridden to 1 (kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,118] INFO Property port is overridden to 9092 (kafka.utils.
VerifiableProperties)
[2015-08-07 01:15:05,118] INFO Property socket.receive.buffer.bytes is overridde
n to 102400 (kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,118] INFO Property socket.request.max.bytes is overridden t
o 104857600 (kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,119] INFO Property socket.send.buffer.bytes is overridden t
o 102400 (kafka.utils.VerifiableProperties)
[2015-08-07 01:15:05,119] INFO Property zookeeper.connect is overridden to local
host:2181 (kafka.utils.VerifiableProperties)
```

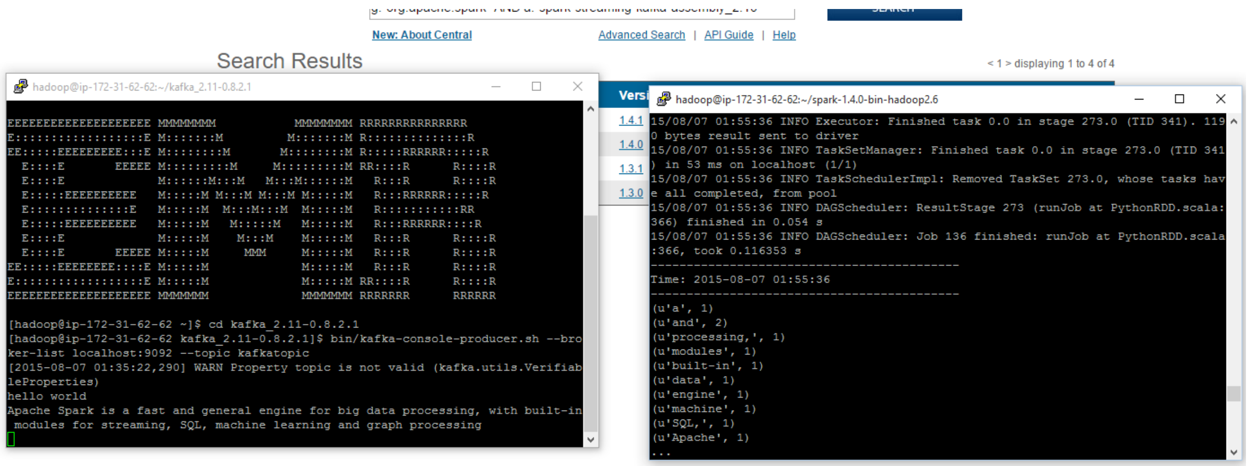
```
 Error Path:/consumers/myconsumergroup/owners/kafkatopic Error:KeeperErrorCode =
 NoNode for /consumers/myconsumergroup/owners/kafkatopic (org.apache.zookeeper.s
erver.PrepRequestProcessor)
[2015-08-07 01:17:31,194] INFO Got user-level KeeperException when processing se
ssionid:0x14f05b8e7f70002 type:create cxid:0x15 zxid:0x25 txntype:-1 reqpath:n/a
 Error Path:/consumers/myconsumergroup/owners Error:KeeperErrorCode = NoNode for
 /consumers/myconsumergroup/owners (org.apache.zookeeper.server.PrepRequestProce
ssor)
[2015-08-07 01:17:56,439] INFO Got user-level KeeperException when processing se
ssionid:0x14f05b8e7f70002 type:setData cxid:0x1b zxid:0x29 txntype:-1 reqpath:n/
a Error Path:/consumers/myconsumergroup/offsets/kafkatopic/0 Error:KeeperErrorCo
de = NoNode for /consumers/myconsumergroup/offsets/kafkatopic/0 (org.apache.zook
eeper.server.PrepRequestProcessor)
[2015-08-07 01:17:56,441] INFO Got user-level KeeperException when processing se
ssionid:0x14f05b8e7f70002 type:create cxid:0x1c zxid:0x2a txntype:-1 reqpath:n/a
 Error Path:/consumers/myconsumergroup/offsets Error:KeeperErrorCode = NoNode fo
r /consumers/myconsumergroup/offsets (org.apache.zookeeper.server.PrepRequestPro
cessor)
[2015-08-07 01:17:56,448] INFO Processed session termination for sessionid: 0x14
f05b8e7f70002 (org.apache.zookeeper.server.PrepRequestProcessor)
[2015-08-07 01:17:56,450] INFO Closed socket connection for client /127.0.0.1:34
957 which had sessionid 0x14f05b8e7f70002 (org.apache.zookeeper.server.NIOServer
Cnxn)
```

```
-------------------------------------------
(4,43)
(8,27)
(6,32)
(0,35)
(2,43)
(7,47)
(5,31)
(9,50)
(3,47)
(1,45)

15/08/07 01:17:41 WARN BlockManager: Block input-0-1438910260800 replicated to o
nly 0 peer(s) instead of 1 peers
15/08/07 01:17:42 WARN BlockManager: Block input-0-1438910261800 replicated to o
nly 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1438910262000 ms
-------------------------------------------
(4,57)
(8,38)
(6,38)
(0,43)
(2,59)
```

4. Kafka(Python)

1. Repeat the same steps till step 6 in Kafka(scala) example.
2. Now download the maven jar file for spark streaming kafka assembly for spark 1.4.0 and place it in spark folder.
3. Now run the following command to run the word count. Whatever you write in kafka console producer terminal will be counted.(see the screen shot)

bin/spark-submit --jars spark-streaming-kafka-assembly*.jar examples/src/main/python/streaming/direct_kafka_wordcount.py localhost:9092 kafkatopic


New: About Central
Advanced Search | API Guide | Help
Search Results
< 1 > displaying 1 to 4 of 4
Versi
1.4.1
1.4.0
1.3.1
1.3.0
hadoop@ip-172-31-62-62:~/kafka_2.11-0.8.2.1
hadoop@ip-172-31-62-62:~/spark-1.4.0-bin-hadoop2.6

```
EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-62-62 ~]$ cd kafka_2.11-0.8.2.1
[hadoop@ip-172-31-62-62 kafka_2.11-0.8.2.1]$ bin/kafka-console-producer.sh --bro
ker-list localhost:9092 --topic kafkatopic
[2015-08-07 01:35:22,290] WARN Property topic is not valid (kafka.utils.Verifiab
leProperties)
hello world
Apache Spark is a fast and general engine for big data processing, with built-in
 modules for streaming, SQL, machine learning and graph processing
```

```
15/08/07 01:55:36 INFO Executor: Finished task 0.0 in stage 273.0 (TID 341). 119
0 bytes result sent to driver
15/08/07 01:55:36 INFO TaskSetManager: Finished task 0.0 in stage 273.0 (TID 341
) in 53 ms on localhost (1/1)
15/08/07 01:55:36 INFO TaskSchedulerImpl: Removed TaskSet 273.0, whose tasks hav
e all completed, from pool
15/08/07 01:55:36 INFO DAGScheduler: ResultStage 273 (runJob at PythonRDD.scala:
366) finished in 0.054 s
15/08/07 01:55:36 INFO DAGScheduler: Job 136 finished: runJob at PythonRDD.scala
:366, took 0.116353 s
-------------------------------------------
Time: 2015-08-07 01:55:36
-------------------------------------------
(u'a', 1)
(u'and', 2)
(u'processing,', 1)
(u'modules', 1)
(u'built-in', 1)
(u'data', 1)
(u'engine', 1)
(u'machine', 1)
(u'SQL,', 1)
(u'Apache', 1)
...
```

5. ZeroMQ

1. We need to do this example on EC2 (Ubuntu) and build all the necessary tools which we need from scratch.
2. Install Java on the instance with following commands

```
sudo apt-get update

sudo apt-get install default-jre

sudo apt-get install default-jdk
```

3. Download spark on it using wget followed by its link. Tar it.
4. Install Scala on Ubuntu by following commands.

```
sudo apt-get remove scala-library scala

sudo wget www.scala-lang.org/files/archive/scala-2.11.4.deb

sudo dpkg -i scala-2.11.4.deb

sudo apt-get update

sudo apt-get install scala

wget https://bintray.com/artifact/download/sbt/debian/sbt-0.13.6.deb

sudo dpkg -i sbt-0.13.6.deb

sudo apt-get update

sudo apt-get install sbt
```

5. Now download ZeroMQ from following command

```
wget http://download.zeromq.org/zeromq-2.1.1-rc.tar.gz
```

Untar it.

6. Run the command `./configure`
7. Install its dependencies with following commands

```
sudo apt-get install g++

sudo apt-get install uuid-dev
```

8. Then run following in that order

```
sudo apt-get install make
```

./configure

make

sudo install make

9. Then run the following command

sudo ldconfig

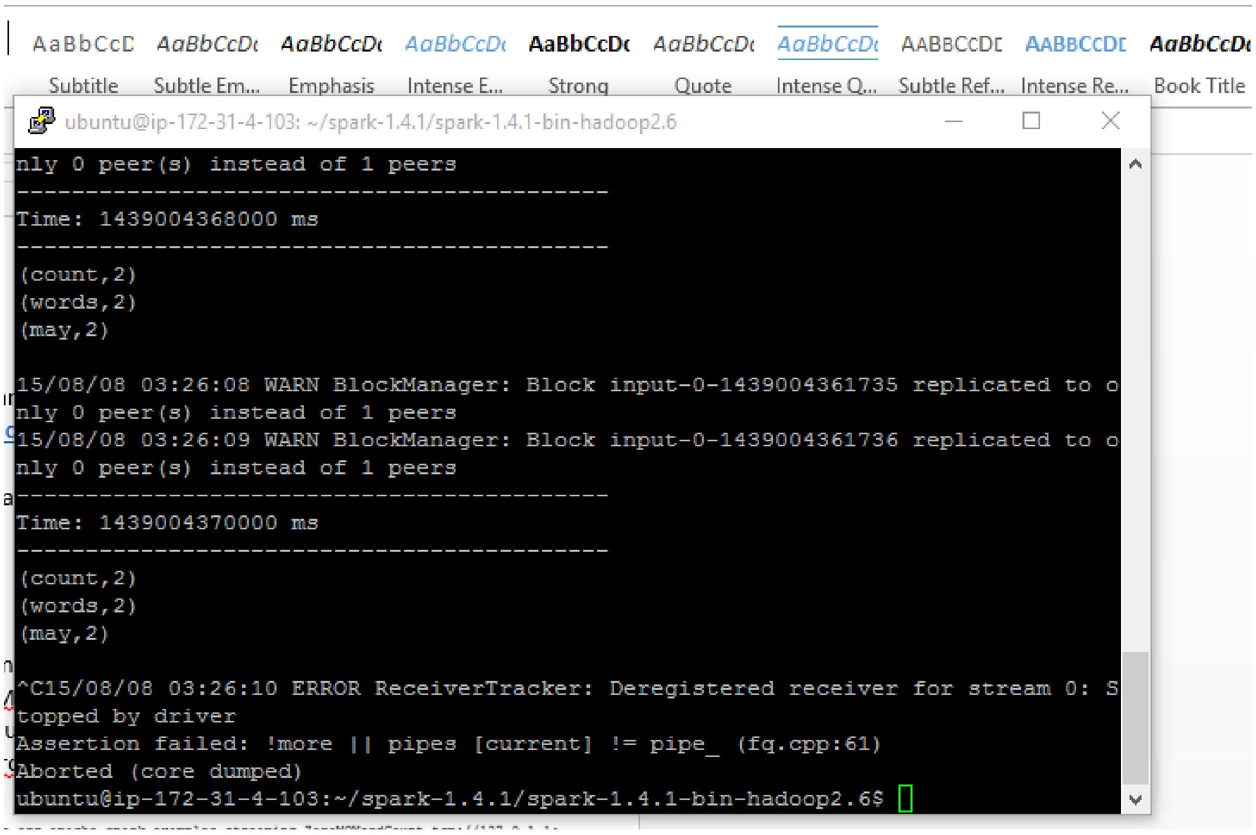
10. Cd to spark folder and run the ZeroMQ publisher example

bin/run-example org.apache.spark.examples.streaming.SimpleZeroMQPublisher tcp://127.0.1.1:1234 foo.bar

11. Then open another terminal and run ZeroMQ word count example.

bin/run-example org.apache.spark.examples.streaming.ZeroMQWordCount tcp://127.0.1.1:1234 foo

It should run and count the words as follows:



```
nly 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1439004368000 ms
-------------------------------------------
(count,2)
(words,2)
(may,2)

15/08/08 03:26:08 WARN BlockManager: Block input-0-1439004361735 replicated to o
nly 0 peer(s) instead of 1 peers
15/08/08 03:26:09 WARN BlockManager: Block input-0-1439004361736 replicated to o
nly 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1439004370000 ms
-------------------------------------------
(count,2)
(words,2)
(may,2)

^C15/08/08 03:26:10 ERROR ReceiverTracker: Deregistered receiver for stream 0: S
topped by driver
Assertion failed: !more || pipes [current] != pipe_ (fq.cpp:61)
Aborted (core dumped)
ubuntu@ip-172-31-4-103:~/spark-1.4.1/spark-1.4.1-bin-hadoop2.6$
```

6. *HDFS Word Count (scala)*

1. Launch an emr 4.0 instance. SSH into the instance

2. I have uploaded the required text file to S3 on this address

   https://s3.amazonaws.com/tanmayaws/emr.txt

3. Create a directory in hdfs named Tanmay with following command

   ```
   hadoop fs –mkdir /user/tanmay
   ```

4. Now go the spark folder

5. And run the following command

   ```
   bin/run-example org.apache.spark.examples.streaming.HdfsWordCount /user/tanmay
   ```

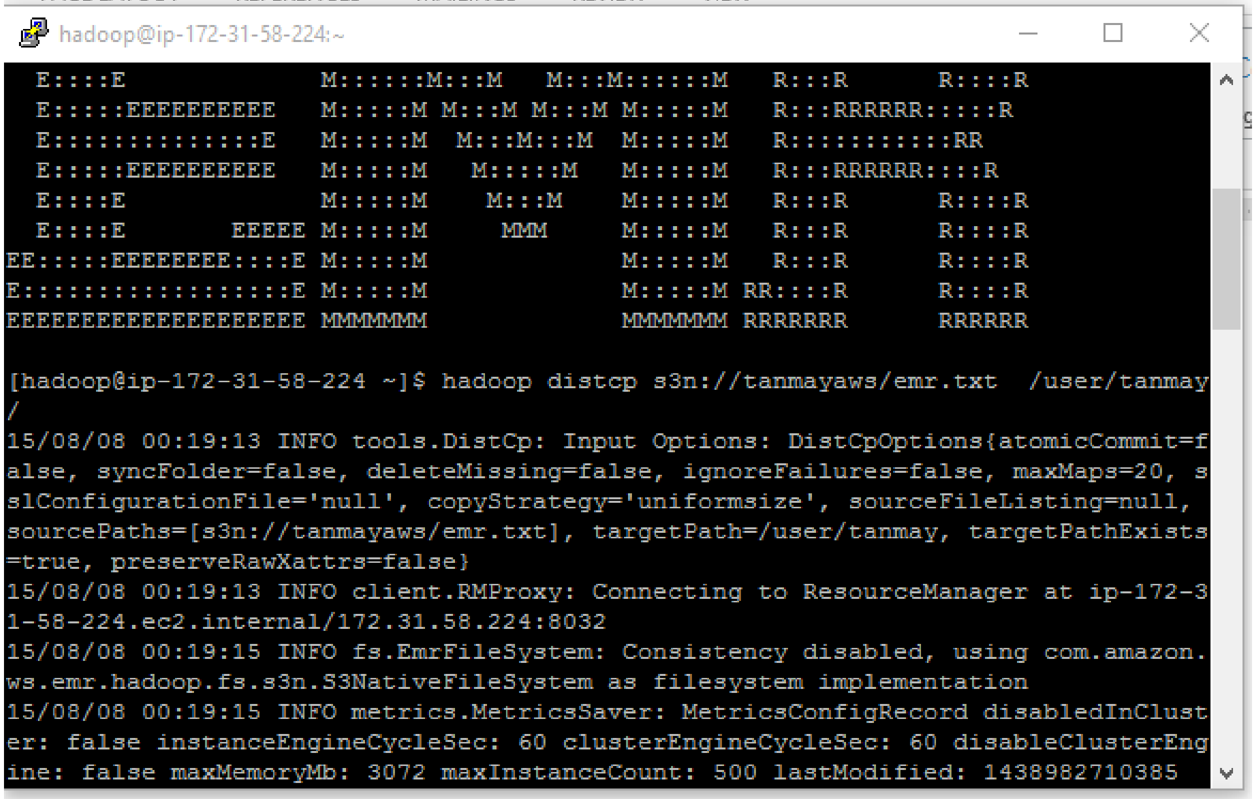
   (/user/tanmay is the local directory in hdfs where we will put the text file.)

6. Now put the text file into /user/tanmay with following command

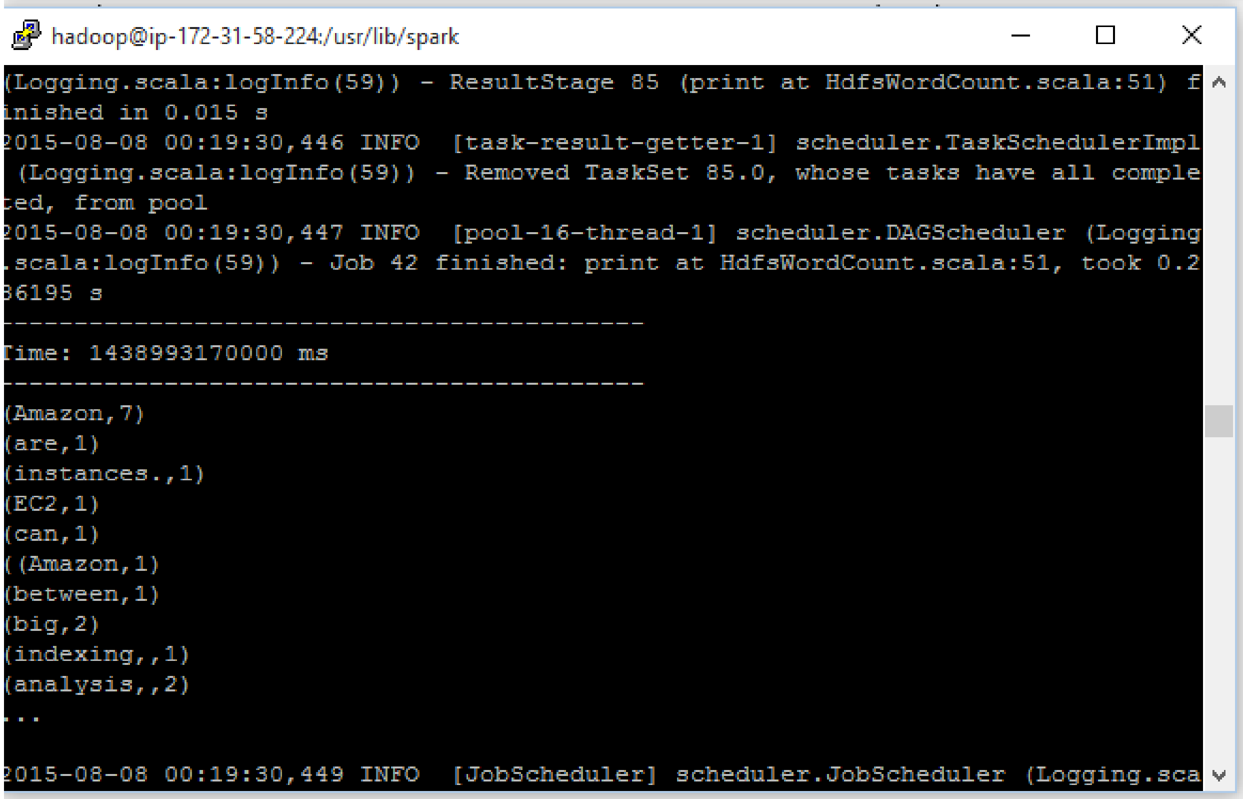
   ```
   hadoop distcp s3n://tanmayaws/emr.txt  /user/tanmay/
   ```

   here emr.txt is the text file which I am using.

7. Now abort to check the output by hitting ctrl+C. You can find the output by scrolling up a bit.


hadoop@ip-172-31-58-224:~

```
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-58-224 ~]$ hadoop distcp s3n://tanmayaws/emr.txt  /user/tanmay
/
15/08/08 00:19:13 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=f
alse, syncFolder=false, deleteMissing=false, ignoreFailures=false, maxMaps=20, s
slConfigurationFile='null', copyStrategy='uniformsize', sourceFileListing=null,
sourcePaths=[s3n://tanmayaws/emr.txt], targetPath=/user/tanmay, targetPathExists
=true, preserveRawXattrs=false}
15/08/08 00:19:13 INFO client.RMProxy: Connecting to ResourceManager at ip-172-3
1-58-224.ec2.internal/172.31.58.224:8032
15/08/08 00:19:15 INFO fs.EmrFileSystem: Consistency disabled, using com.amazon.
ws.emr.hadoop.fs.s3n.S3NativeFileSystem as filesystem implementation
15/08/08 00:19:15 INFO metrics.MetricsSaver: MetricsConfigRecord disabledInClust
er: false instanceEngineCycleSec: 60 clusterEngineCycleSec: 60 disableClusterEng
ine: false maxMemoryMb: 3072 maxInstanceCount: 500 lastModified: 1438982710385
```


hadoop@ip-172-31-58-224:/usr/lib/spark

```
(Logging.scala:logInfo(59)) - ResultStage 85 (print at HdfsWordCount.scala:51) f
inished in 0.015 s
2015-08-08 00:19:30,446 INFO  [task-result-getter-1] scheduler.TaskSchedulerImpl
 (Logging.scala:logInfo(59)) - Removed TaskSet 85.0, whose tasks have all comple
ted, from pool
2015-08-08 00:19:30,447 INFO  [pool-16-thread-1] scheduler.DAGScheduler (Logging
.scala:logInfo(59)) - Job 42 finished: print at HdfsWordCount.scala:51, took 0.2
36195 s
-------------------------------------------
Time: 1438993170000 ms
-------------------------------------------
(Amazon,7)
(are,1)
(instances.,1)
(EC2,1)
(can,1)
((Amazon,1)
(between,1)
(big,2)
(indexing,,1)
(analysis,,2)
...

2015-08-08 00:19:30,449 INFO  [JobScheduler] scheduler.JobScheduler (Logging.sca
```

7. HDFS word count(python)

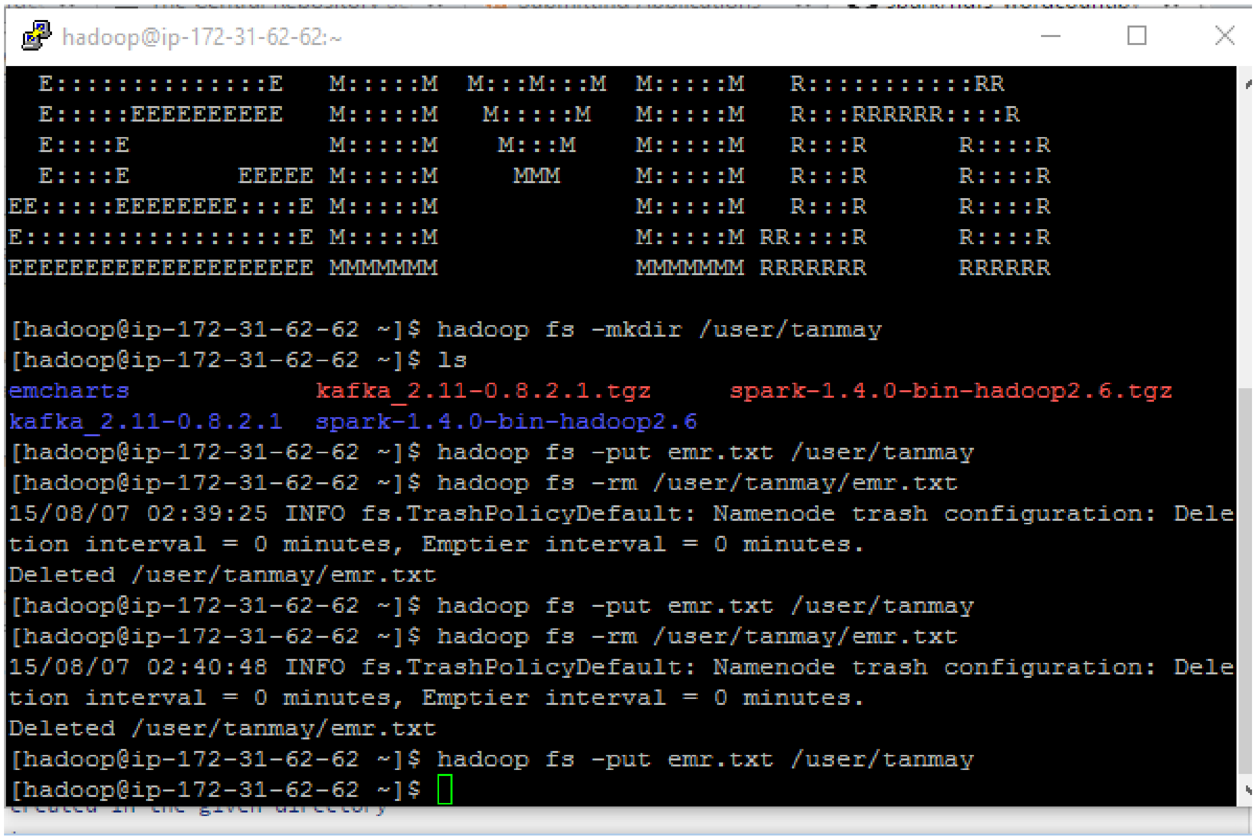
1. Repeat the first 4 steps from the scala example.

2. Run the following command

   ```
   bin/spark-submit examples/src/main/python/streaming/hdfs_wordcount.py <localdir>
   ```

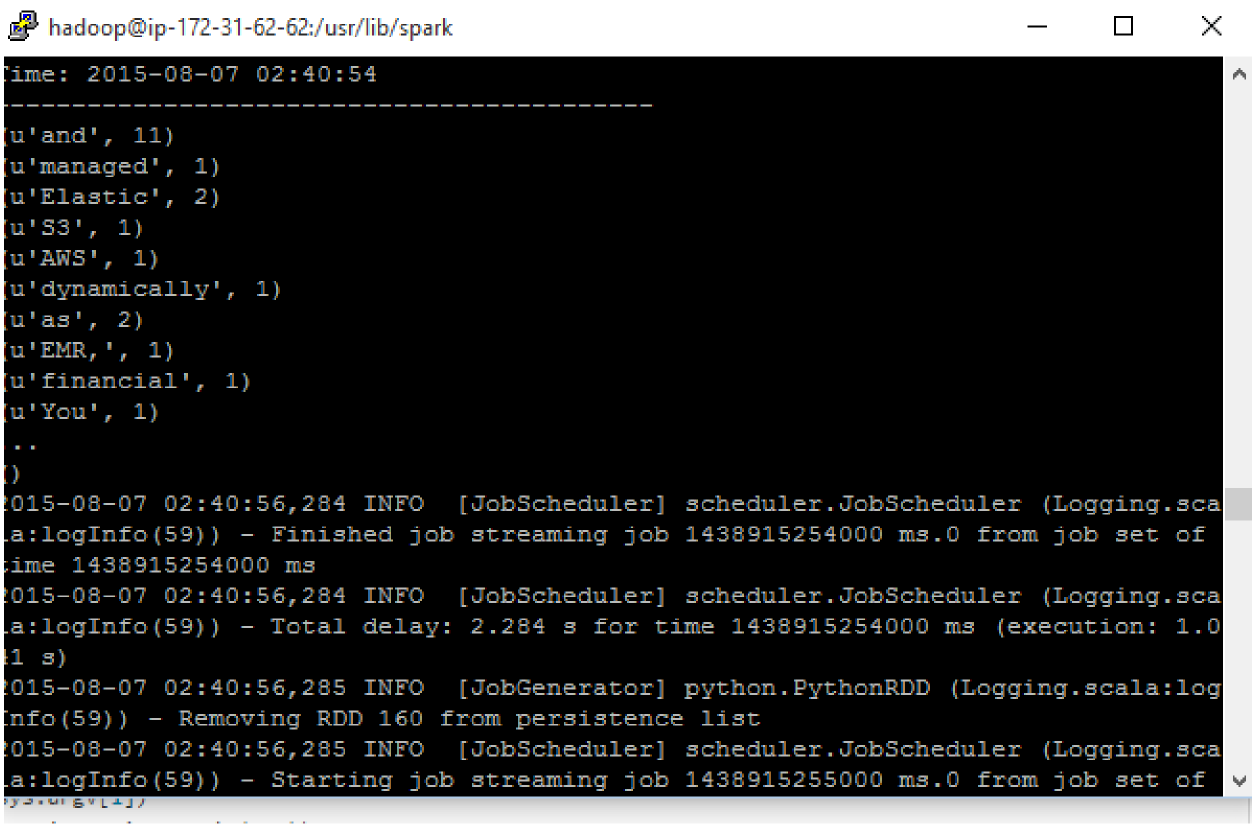
   where localdir will be /user/tanmay.

3. Now repeat steps 6 and 7 from scala example.



hadoop@ip-172-31-62-62:~

```
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-62-62 ~]$ hadoop fs -mkdir /user/tanmay
[hadoop@ip-172-31-62-62 ~]$ ls
emcharts             kafka_2.11-0.8.2.1.tgz     spark-1.4.0-bin-hadoop2.6.tgz
kafka_2.11-0.8.2.1   spark-1.4.0-bin-hadoop2.6
[hadoop@ip-172-31-62-62 ~]$ hadoop fs -put emr.txt /user/tanmay
[hadoop@ip-172-31-62-62 ~]$ hadoop fs -rm /user/tanmay/emr.txt
15/08/07 02:39:25 INFO fs.TrashPolicyDefault: Namenode trash configuration: Dele
tion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/tanmay/emr.txt
[hadoop@ip-172-31-62-62 ~]$ hadoop fs -put emr.txt /user/tanmay
[hadoop@ip-172-31-62-62 ~]$ hadoop fs -rm /user/tanmay/emr.txt
15/08/07 02:40:48 INFO fs.TrashPolicyDefault: Namenode trash configuration: Dele
tion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/tanmay/emr.txt
[hadoop@ip-172-31-62-62 ~]$ hadoop fs -put emr.txt /user/tanmay
[hadoop@ip-172-31-62-62 ~]$
```



hadoop@ip-172-31-62-62:/usr/lib/spark

```
ime: 2015-08-07 02:40:54
-------------------------------------------
(u'and', 11)
(u'managed', 1)
(u'Elastic', 2)
(u'S3', 1)
(u'AWS', 1)
(u'dynamically', 1)
(u'as', 2)
(u'EMR,', 1)
(u'financial', 1)
(u'You', 1)
...
)
2015-08-07 02:40:56,284 INFO  [JobScheduler] scheduler.JobScheduler (Logging.sca
la:logInfo(59)) - Finished job streaming job 1438915254000 ms.0 from job set of
time 1438915254000 ms
2015-08-07 02:40:56,284 INFO  [JobScheduler] scheduler.JobScheduler (Logging.sca
la:logInfo(59)) - Total delay: 2.284 s for time 1438915254000 ms (execution: 1.0
41 s)
2015-08-07 02:40:56,285 INFO  [JobGenerator] python.PythonRDD (Logging.scala:log
Info(59)) - Removing RDD 160 from persistence list
2015-08-07 02:40:56,285 INFO  [JobScheduler] scheduler.JobScheduler (Logging.sca
la:logInfo(59)) - Starting job streaming job 1438915255000 ms.0 from job set of
```

8. MQTT

(spark 1.4.1 gave an error so we ran on spark 1.4.0)

1. Start an instance on emr 4.0 and install spark 1.4.0 on it.

Now ec2 instance runs on Red Hat Enterprise Linux(RHEL) 7. So go to following site to download the mosquito broker. http://mosquitto.org/download/. Right click and download the repository file. Now SCP this repo file into your instance.

2. Copy this file into /etc/yum.repos.d/
3. Now install mosquito and mosquito clients using the following commands
   sudo yum install mosquito
   sudo yum install mosquito-clients

4. Check if mosquito, mosquito_sub and mosquito_pub files appear in /usr/sbin. Check if mosquito file appears in /etc/init.d/.
5. Now to start the MQTT broker hit the following command
   /etc/init.d/mosquito start.

   Your broker will be running now on localhost:1883

   (Following steps are just to check if mosquito is working properly or not. No need to do following two steps in using spark example.
   a. Open another terminal and give following command:- mosquitto_sub -t newtopic/test
   b. Now open another terminal and give following command:-
      mosquitto_pub -d -t newtopic/test -m "Hello World"
   c. You should see Hello World appear on the previous terminal. If it does then your mqtt broker is working.

)

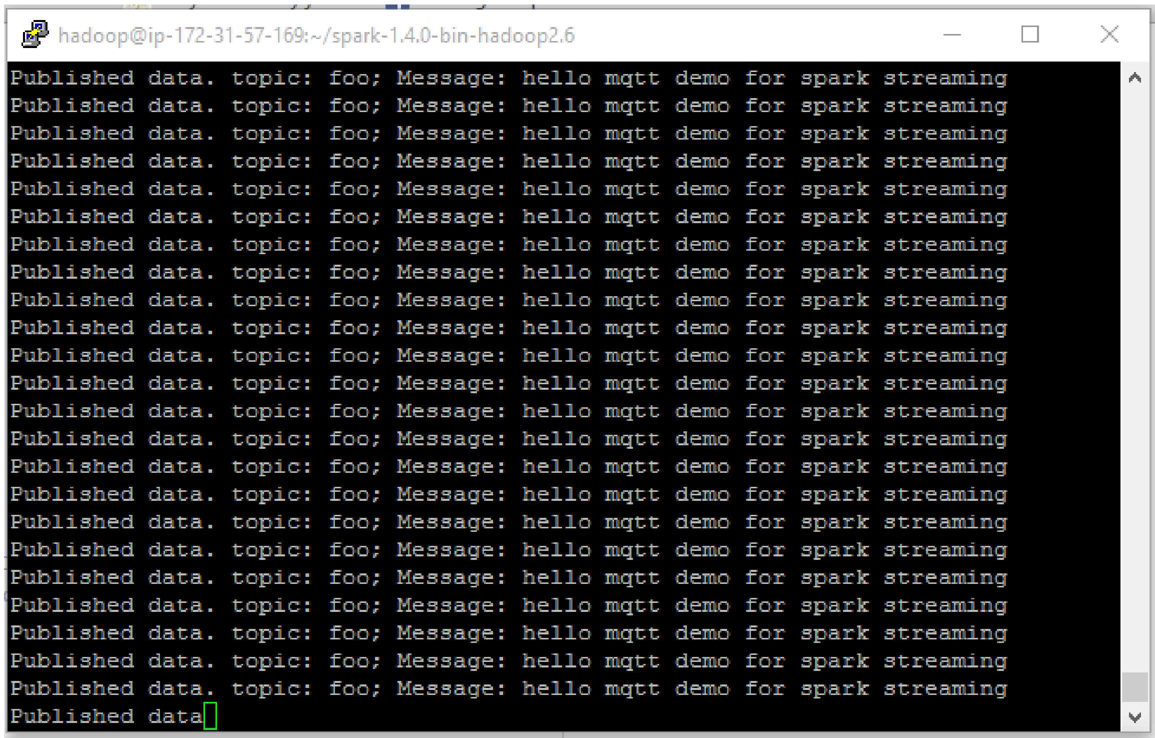
6. Open another terminal and go to the spark(1.4.0) root directory. Hit following command to start the publisher.
   bin/run-example org.apache.spark.examples.streaming.MQTTPublisher tcp://localhost:1883 foo

   You will see stream of the sentence 'hello mqtt demo for spark streaming' appear continuously on the screen.

7. Open another terminal and from spark directory run the following command to start the word count example.

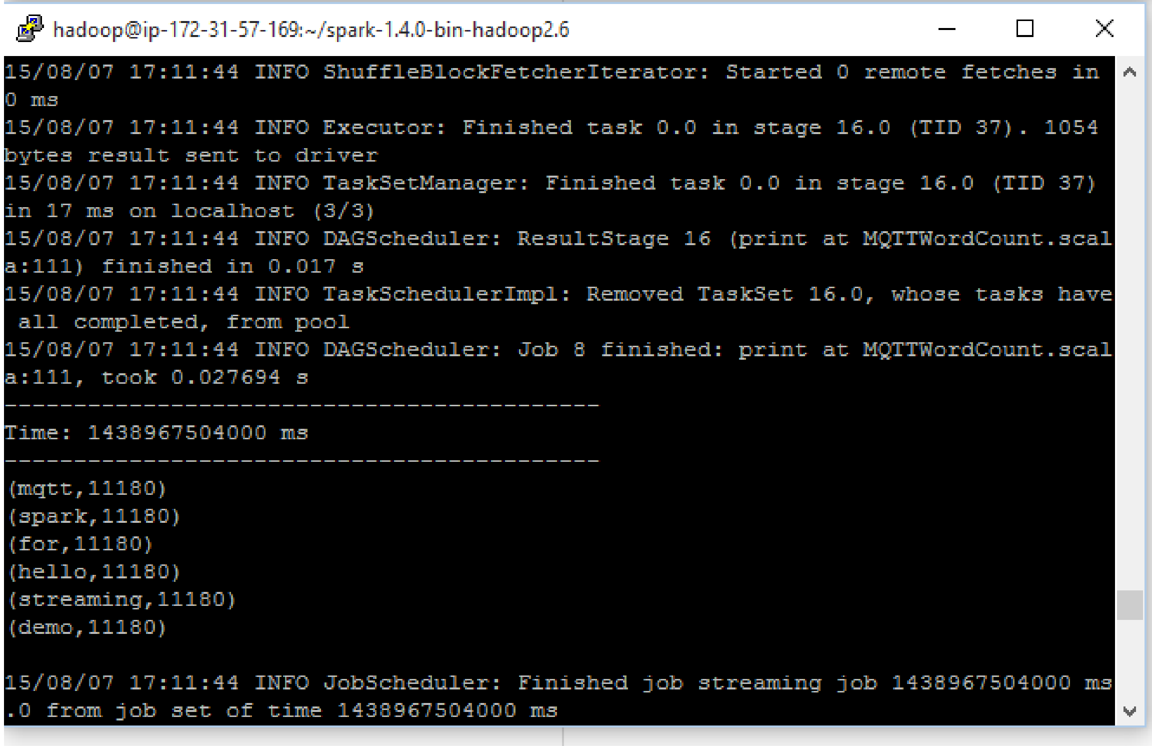
bin/run-example org.apache.spark.examples.streaming.MQTTWordCount tcp://localhost:1883 foo

Hit Ctrl+C to stop the output and see if word count is happening as in following screenshot.



hadoop@ip-172-31-57-169:~/spark-1.4.0-bin-hadoop2.6

```
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data. topic: foo; Message: hello mqtt demo for spark streaming
Published data
```



hadoop@ip-172-31-57-169:~/spark-1.4.0-bin-hadoop2.6

```
15/08/07 17:11:44 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in
0 ms
15/08/07 17:11:44 INFO Executor: Finished task 0.0 in stage 16.0 (TID 37). 1054
bytes result sent to driver
15/08/07 17:11:44 INFO TaskSetManager: Finished task 0.0 in stage 16.0 (TID 37)
in 17 ms on localhost (3/3)
15/08/07 17:11:44 INFO DAGScheduler: ResultStage 16 (print at MQTTWordCount.scal
a:111) finished in 0.017 s
15/08/07 17:11:44 INFO TaskSchedulerImpl: Removed TaskSet 16.0, whose tasks have
 all completed, from pool
15/08/07 17:11:44 INFO DAGScheduler: Job 8 finished: print at MQTTWordCount.scal
a:111, took 0.027694 s
-------------------------------------------
Time: 1438967504000 ms
-------------------------------------------
(mqtt,11180)
(spark,11180)
(for,11180)
(hello,11180)
(streaming,11180)
(demo,11180)

15/08/07 17:11:44 INFO JobScheduler: Finished job streaming job 1438967504000 ms
.0 from job set of time 1438967504000 ms
```

9. Flume(scala)

1. Start instance on emr 4.0 and install spark 1.4.0.
2. SCP flume.tgz and tar it. Go inside the flume directory
3. Now create a new flume.conf file by following command

   ```
   cp conf/flume-conf.properties.template conf/flume.conf

   Now change the flume.conf file to following.
   ```

```
#Define an avro1 source on agent1 and tell it to bind to localhost:12345.
Connect it to channel memorychannel
agent.sources.avro1.type=netcat
agent.sources.avro1.bind = localhost
agent.sources.avro1.port = 12345
agent.sources.avro1.channels = memorychannel


#Define an avro sink that and connect it to other end of the same channel
agent.sinks.spark.type = avro
agent.sinks.spark.hostname = localhost
agent.sinks.spark.port = 56789
agent.sinks.spark.channel = memorychannel

#define channel specifications
agent.channels.memorychannel.type = memory
agent.channels.memorychannel.capacity = 10000
agent.channels.memorychannel.transitioncapacity = 10000


# Finally tell agent which components to activate
agent.sources = avro1
agent.channels = memorychannel
agent.sinks = spark
```

4. Open a terminal and go to spark directory
5. Add the following command.

```
bin/run-example org.apache.spark.examples.streaming.FlumeEventCount localhost 56789
```

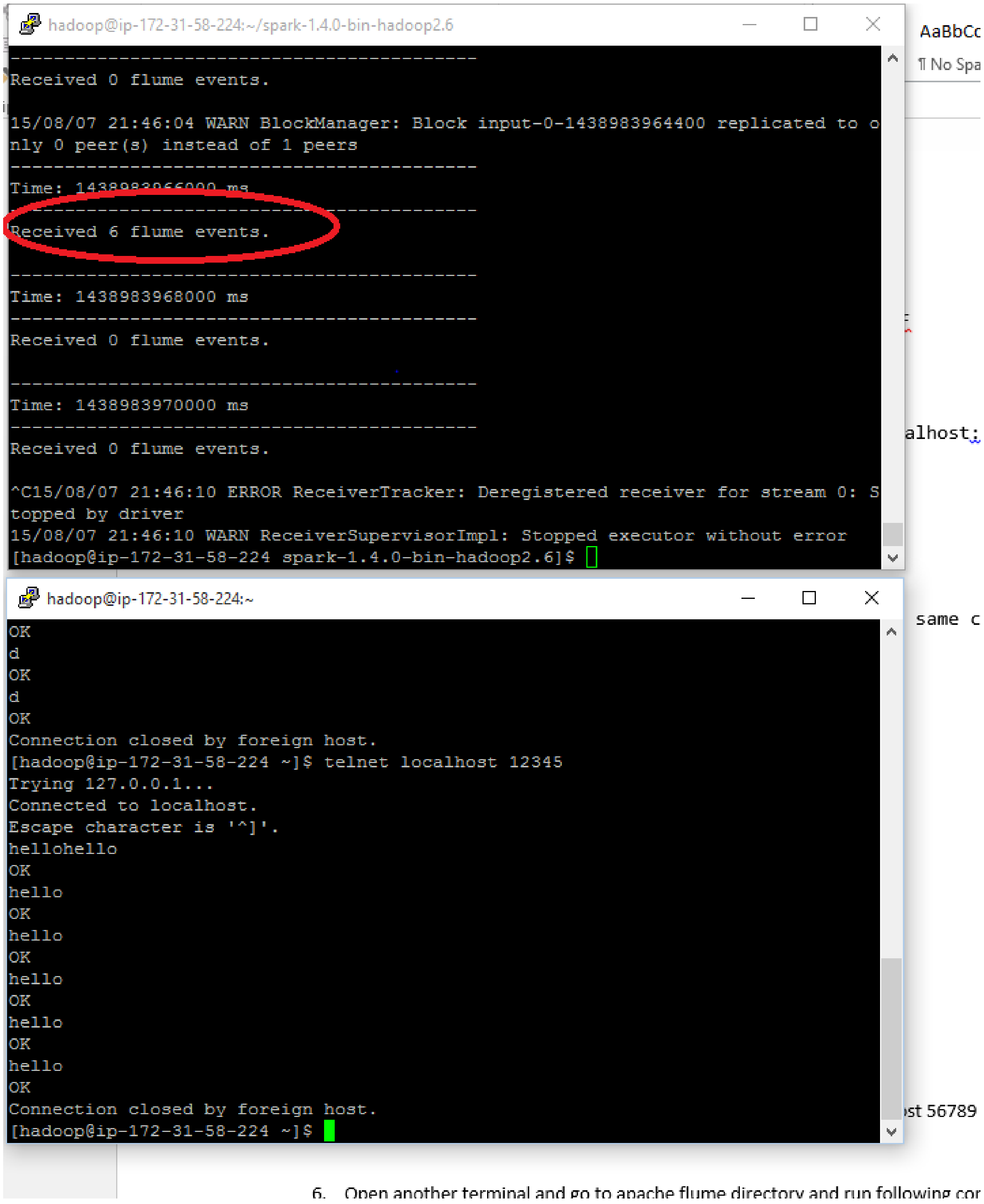
6. Open another terminal and go to apache flume directory and run following command

   ```
   bin/flume-ng agent --conf conf --conf-file conf/flume.conf --name agent -
   Dflume.root.avro=INFO,console
   ```

7. Then open another terminal and run following command

telnet localhost 12345

8. Now whatever you write in this window will be counted as events in spark window.


hadoop@ip-172-31-58-224:~/spark-1.4.0-bin-hadoop2.6

```
-------------------------------------------
Received 0 flume events.

15/08/07 21:46:04 WARN BlockManager: Block input-0-1438983964400 replicated to o
nly 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1438983966000 ms
-------------------------------------------
Received 6 flume events.

-------------------------------------------
Time: 1438983968000 ms
-------------------------------------------
Received 0 flume events.

-------------------------------------------
Time: 1438983970000 ms
-------------------------------------------
Received 0 flume events.

^C15/08/07 21:46:10 ERROR ReceiverTracker: Deregistered receiver for stream 0: S
topped by driver
15/08/07 21:46:10 WARN ReceiverSupervisorImpl: Stopped executor without error
[hadoop@ip-172-31-58-224 spark-1.4.0-bin-hadoop2.6]$
```

hadoop@ip-172-31-58-224:~

```
OK
d
OK
d
OK
Connection closed by foreign host.
[hadoop@ip-172-31-58-224 ~]$ telnet localhost 12345
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
hellohello
OK
hello
OK
hello
OK
hello
OK
hello
OK
hello
OK
Connection closed by foreign host.
[hadoop@ip-172-31-58-224 ~]$
```

6. Open another terminal and go to apache flume directory and run following cor

10.Sqlnetwork Count

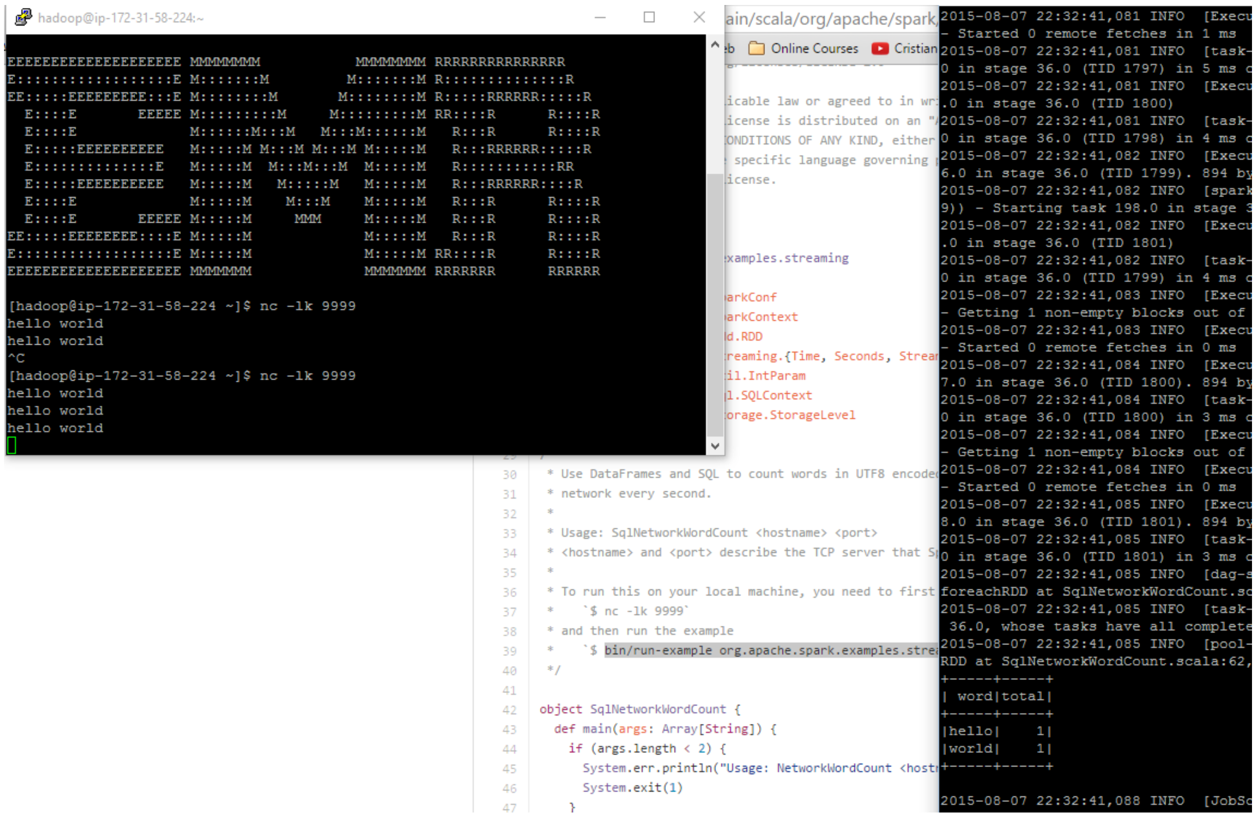
(scala and python)(substituted for Flume python example)

1. Run an emr4.0 instance.
2. Start netcat by following command on local host 9999 by following command
   nc –lk 9999

3. Start scala example by going into spark directory and running following command

   bin/run-example org.apache.spark.examples.streaming.SqlNetworkWordCount localhost 9999

4. Then write something in netcat terminal and it should be counted in spark.



hadoop@ip-172-31-58-224:~

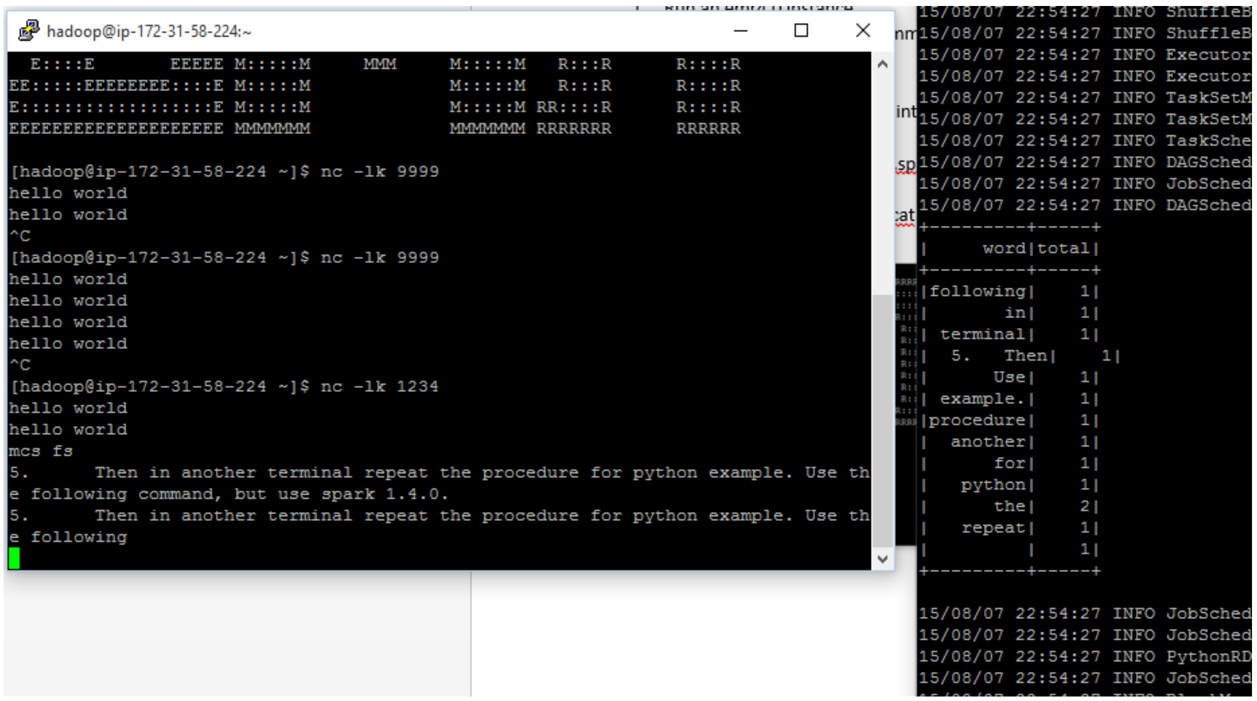
```
EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-58-224 ~]$ nc -lk 9999
hello world
hello world
^C
[hadoop@ip-172-31-58-224 ~]$ nc -lk 9999
hello world
hello world
hello world
```

ain/scala/org/apache/spark,

```
icable law or agreed to in wr
icense is distributed on an "
ONDITIONS OF ANY KIND, either
 specific language governing
icense.

xamples.streaming

arkConf
arkContext
d.RDD
reaming.{Time, Seconds, Strea
il.IntParam
l.SQLContext
orage.StorageLevel

 * Use DataFrames and SQL to count words in UTF8 encode
 * network every second.
 *
 * Usage: SqlNetworkWordCount <hostname> <port>
 * <hostname> and <port> describe the TCP server that S
 *
 * To run this on your local machine, you need to first
 *    `$ nc -lk 9999`
 * and then run the example
 *    `$ bin/run-example org.apache.spark.examples.stre
 */

object SqlNetworkWordCount {
  def main(args: Array[String]) {
    if (args.length < 2) {
      System.err.println("Usage: NetworkWordCount <host
      System.exit(1)
    }
```

```
2015-08-07 22:32:41,081 INFO  [Execu
- Started 0 remote fetches in 1 ms
2015-08-07 22:32:41,081 INFO  [task-
0 in stage 36.0 (TID 1797) in 5 ms c
2015-08-07 22:32:41,081 INFO  [Execu
.0 in stage 36.0 (TID 1800)
2015-08-07 22:32:41,081 INFO  [task-
0 in stage 36.0 (TID 1798) in 4 ms c
2015-08-07 22:32:41,082 INFO  [Execu
6.0 in stage 36.0 (TID 1799). 894 by
2015-08-07 22:32:41,082 INFO  [spark
9)) - Starting task 198.0 in stage 3
2015-08-07 22:32:41,082 INFO  [Execu
.0 in stage 36.0 (TID 1801)
2015-08-07 22:32:41,082 INFO  [task-
0 in stage 36.0 (TID 1799) in 4 ms c
2015-08-07 22:32:41,083 INFO  [Execu
- Getting 1 non-empty blocks out of
2015-08-07 22:32:41,083 INFO  [Execu
- Started 0 remote fetches in 0 ms
2015-08-07 22:32:41,084 INFO  [Execu
7.0 in stage 36.0 (TID 1800). 894 by
2015-08-07 22:32:41,084 INFO  [task-
0 in stage 36.0 (TID 1800) in 3 ms c
2015-08-07 22:32:41,084 INFO  [Execu
- Getting 1 non-empty blocks out of
2015-08-07 22:32:41,084 INFO  [Execu
- Started 0 remote fetches in 0 ms
2015-08-07 22:32:41,085 INFO  [Execu
8.0 in stage 36.0 (TID 1801). 894 by
2015-08-07 22:32:41,085 INFO  [task-
0 in stage 36.0 (TID 1801) in 3 ms c
2015-08-07 22:32:41,085 INFO  [dag-s
foreachRDD at SqlNetworkWordCount.sc
2015-08-07 22:32:41,085 INFO  [task-
 36.0, whose tasks have all complete
2015-08-07 22:32:41,085 INFO  [pool-
RDD at SqlNetworkWordCount.scala:62,
+-----+-----+
| word|total|
+-----+-----+
|hello|    1|
|world|    1|
+-----+-----+

2015-08-07 22:32:41,088 INFO  [JobSc
```

5. Then in another terminal repeat the procedure for python example. Use the following command, but use spark 1.4.0.
   bin/spark-submit examples/src/main/python/streaming/sql_network_wordcount.py localhost 9999



```
hadoop@ip-172-31-58-224:~
  E::::E       EEEEE M:::::M      MMM      M:::::M   R:::R       R::::R
EE:::::EEEEEEEE::::E M:::::M               M:::::M   R:::R       R::::R
E::::::::::::::::::E M:::::M               M:::::M RR::::R       R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM               MMMMMMM RRRRRRR       RRRRRR

[hadoop@ip-172-31-58-224 ~]$ nc -lk 9999
hello world
hello world
^C
[hadoop@ip-172-31-58-224 ~]$ nc -lk 9999
hello world
hello world
hello world
hello world
^C
[hadoop@ip-172-31-58-224 ~]$ nc -lk 1234
hello world
hello world
mcs fs
5.      Then in another terminal repeat the procedure for python example. Use th
e following command, but use spark 1.4.0.
5.      Then in another terminal repeat the procedure for python example. Use th
e following
```

```
15/08/07 22:54:27 INFO ShuffleB
15/08/07 22:54:27 INFO ShuffleB
15/08/07 22:54:27 INFO Executor
15/08/07 22:54:27 INFO Executor
15/08/07 22:54:27 INFO TaskSetM
15/08/07 22:54:27 INFO TaskSetM
15/08/07 22:54:27 INFO TaskSche
15/08/07 22:54:27 INFO DAGSched
15/08/07 22:54:27 INFO JobSched
15/08/07 22:54:27 INFO DAGSched
+---------+-----+
|     word|total|
+---------+-----+
|following|    1|
|       in|    1|
| terminal|    1|
|  5.   Then|    1|
|      Use|    1|
| example.|    1|
|procedure|    1|
|  another|    1|
|      for|    1|
|   python|    1|
|      the|    2|
|   repeat|    1|
|         |    1|
+---------+-----+

15/08/07 22:54:27 INFO JobSched
15/08/07 22:54:27 INFO JobSched
15/08/07 22:54:27 INFO PythonRD
15/08/07 22:54:27 INFO JobSched
```