MIDS W205

Lab #	12	**Lab Title**	Spark Streaming Introduction
Related Module(s)	9	**Goal**	Introduction to Spark Streaming.
Last Updated	10/20/15	**Expected duration**	60 minutes

Introduction

Spark Streaming

.

Instructions, Resources, and Prerequisites

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. This lab will cover the followings:

1. Pre-Requirements to start Spark Streaming
2. How to generate streaming data using Python
3. How to Access & Initialize dependencies for Spark Streaming on Spark shell
4. Files live stream example using Spark Streaming from Spark Shell

Resource	**What**

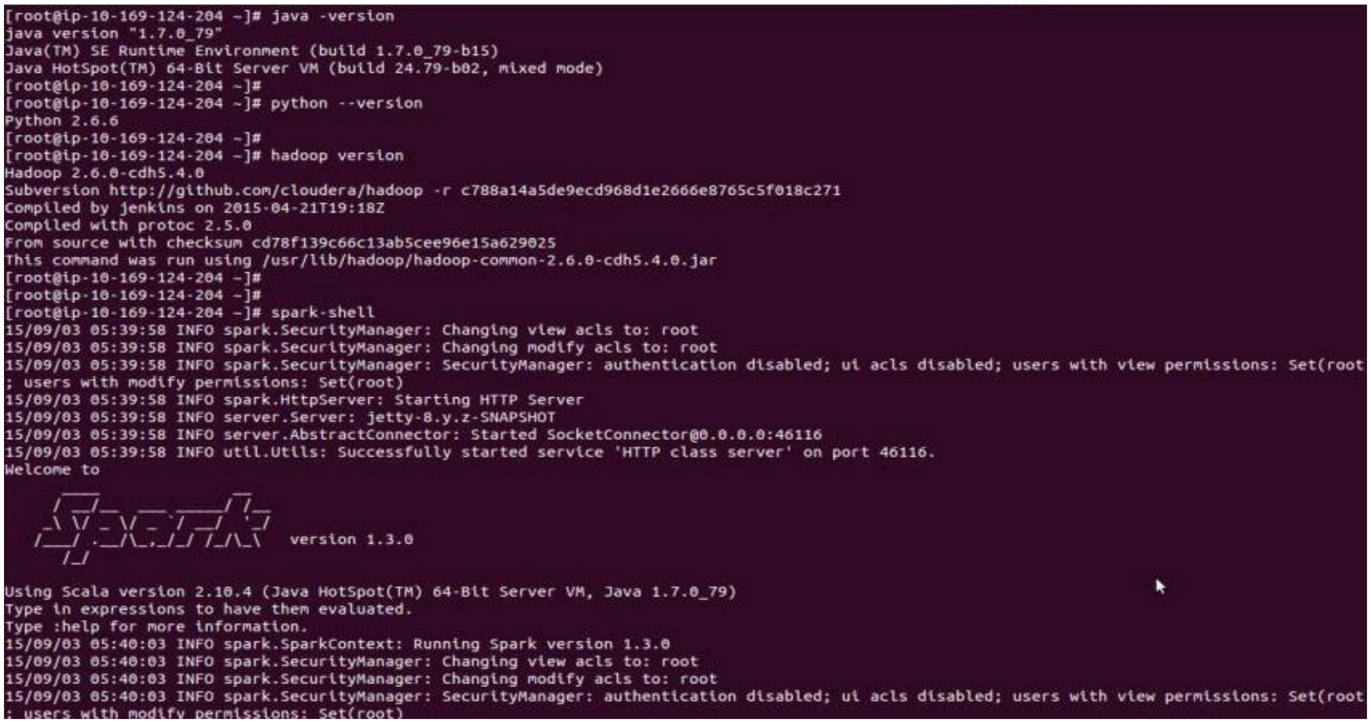
Step-1. Pre-Requirements to start Spark Streaming:

You must have the followings on your server to start Spark Streaming.

1. Java 1.6+
2. Hadoop 1.X or 2.X - optional (for file storage or we can use local file system also)

3. Spark 1.X
4. Data source (we will be using file stream)
5. Python

Please check on your server if these exist. You can refer to the following Unix log for the exact code.



```
[root@ip-10-169-124-204 ~]# java -version
java version "1.7.0_79"
Java(TM) SE Runtime Environment (build 1.7.0_79-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)
[root@ip-10-169-124-204 ~]#
[root@ip-10-169-124-204 ~]# python --version
Python 2.6.6
[root@ip-10-169-124-204 ~]#
[root@ip-10-169-124-204 ~]# hadoop version
Hadoop 2.6.0-cdh5.4.0
Subversion http://github.com/cloudera/hadoop -r c788a14a5de9ecd968d1e2666e8765c5f018c271
Compiled by jenkins on 2015-04-21T19:18Z
Compiled with protoc 2.5.0
From source with checksum cd78f139c66c13ab5cee96e15a629025
This command was run using /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.4.0.jar
[root@ip-10-169-124-204 ~]#
[root@ip-10-169-124-204 ~]#
[root@ip-10-169-124-204 ~]# spark-shell
15/09/03 05:39:58 INFO spark.SecurityManager: Changing view acls to: root
15/09/03 05:39:58 INFO spark.SecurityManager: Changing modify acls to: root
15/09/03 05:39:58 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(root
; users with modify permissions: Set(root)
15/09/03 05:39:58 INFO spark.HttpServer: Starting HTTP Server
15/09/03 05:39:58 INFO server.Server: jetty-8.y.z-SNAPSHOT
15/09/03 05:39:58 INFO server.AbstractConnector: Started SocketConnector@0.0.0.0:46116
15/09/03 05:39:58 INFO util.Utils: Successfully started service 'HTTP class server' on port 46116.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.3.0
      /_/

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_79)
Type in expressions to have them evaluated.
Type :help for more information.
15/09/03 05:40:03 INFO spark.SparkContext: Running Spark version 1.3.0
15/09/03 05:40:03 INFO spark.SecurityManager: Changing view acls to: root
15/09/03 05:40:03 INFO spark.SecurityManager: Changing modify acls to: root
15/09/03 05:40:03 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(root
; users with modify permissions: Set(root)
```

Step-2. How to generate streaming data using Python:

Here is the small snippet of code to generate data using python.

1. Create a python file name “generate_data.py” to generate live streaming data. Make sure there is no error copying and pasting this code to your script.

```
#!/usr/bin/python
import threading
import time
import string
import random
import os
import uuid


class Generate_events(threading.Thread):
    def __init__(self, events_count, file_name):
        threading.Thread.__init__(self)
        if os.path.exists(file_name):
            self.file_name = open(file_name, 'a')
        else:
            self.file_name = open(file_name, 'w')
        self.events_count = events_count

    def __gen_sal(self):
        while True:
            sal = range(1000000)
            yield random.choice(sal)

    def __gen_emp_number(self):
        while True:
            sal = range(1000000)
            yield random.choice(sal)

    def __gen_bonus(self):
        while True:
            bonus = range(500000)
            yield random.choice(bonus)

    def __gen_name(self):
        alphabets = list(string.ascii_lowercase)
        while True:
            yield ''.join(random.choice(alphabets) for _ in range(6))

    def _get_name(self):
        return self.__gen_name().next()

    def _get_sal(self):
        return self.__gen_sal().next()

    def _get_bonus(self):
        return self.__gen_bonus().next()

    def _get_emp_number(self):

        return self.__gen_emp_number().next()

    def run(self):
        try:
            for event in range(self.events_count):
                self.file_name.write("\n" + str(self._get_emp_number()) + ',' +
self._get_name() + ',' + str(self._get_sal()) + ',' + str(self._get_bonus()))
            self.file_name.close()
        except Exception, e:
            print e


while(True):
    #id = os.urandom(32)
    id = uuid.uuid4().int & (1<<64)-1
    file = 'file'+str(id)+'.txt'
    Generate_events(50, '/root/data/'+file).start()
    time.sleep(5)
```

2. Create a folder call “data” in /root/ directory

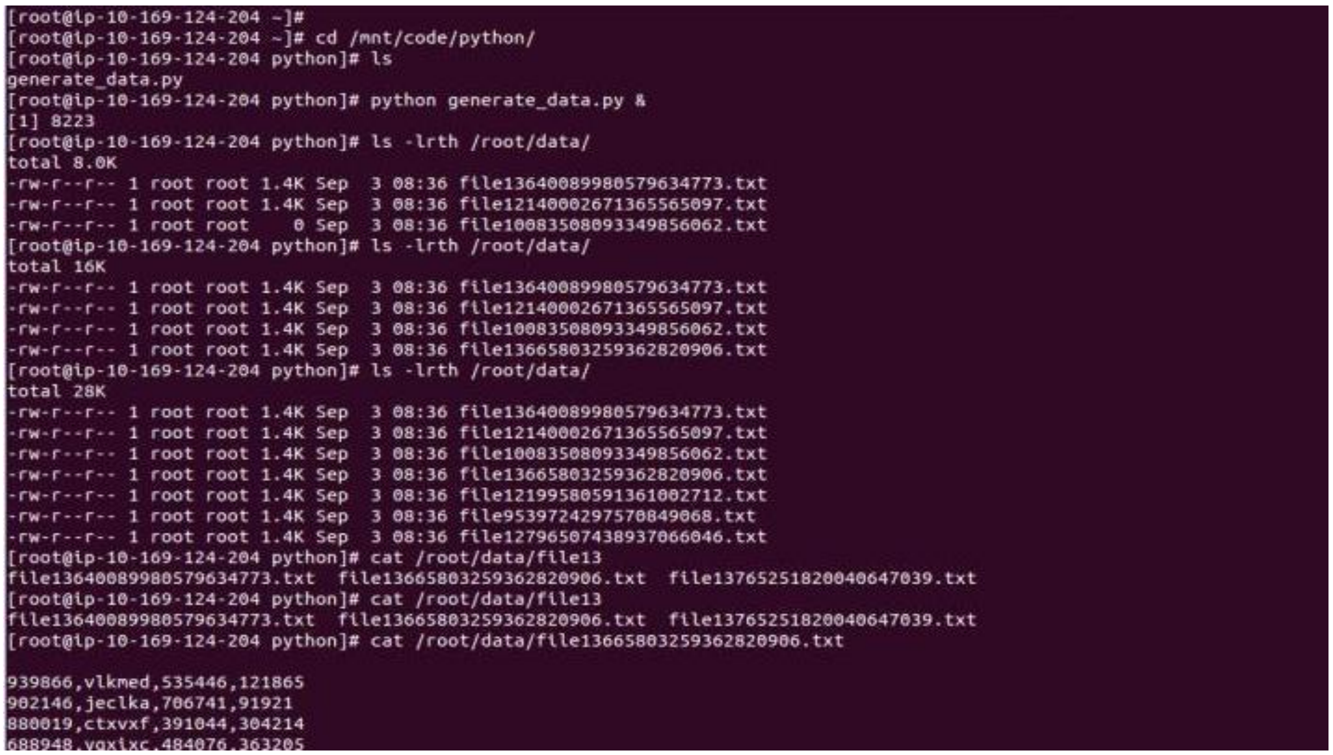
3. Input Data set contains these columns “`emp_number,name,salary,bonus,filename`”

4. You can run this python file using this command “python generate_data.py”

5. You can find result data in /root/data/. You can use this command to see your data “ls

-lrth /root/data/". Don't keep running this script as it can fill up your space quickly.

Please refer to the Unix log of these steps.



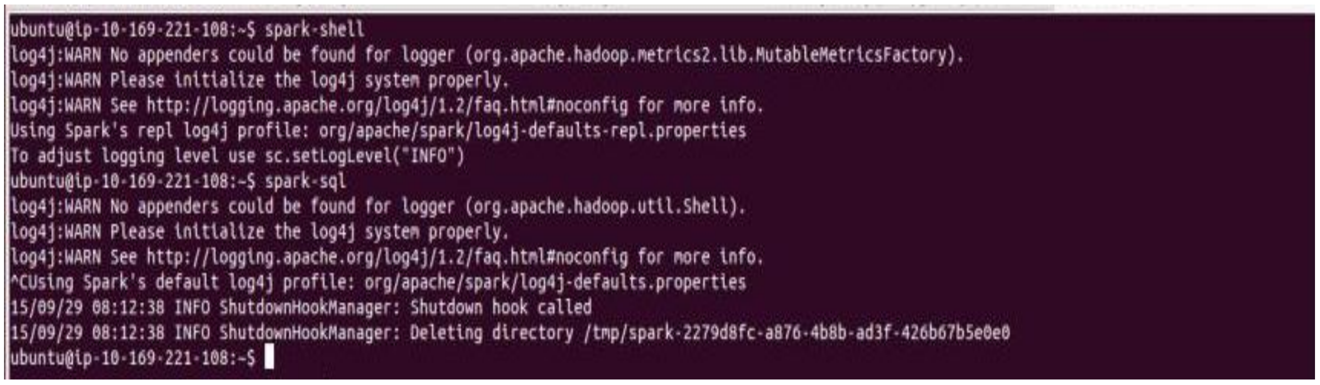
```
[root@ip-10-169-124-204 ~]#
[root@ip-10-169-124-204 ~]# cd /mnt/code/python/
[root@ip-10-169-124-204 python]# ls
generate_data.py
[root@ip-10-169-124-204 python]# python generate_data.py &
[1] 8223
[root@ip-10-169-124-204 python]# ls -lrth /root/data/
total 8.0K
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file13640089980579634773.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file12140002671365565097.txt
-rw-r--r-- 1 root root    0 Sep  3 08:36 file10083508093349856062.txt
[root@ip-10-169-124-204 python]# ls -lrth /root/data/
total 16K
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file13640089980579634773.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file12140002671365565097.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file10083508093349856062.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file13665803259362820906.txt
[root@ip-10-169-124-204 python]# ls -lrth /root/data/
total 28K
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file13640089980579634773.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file12140002671365565097.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file10083508093349856062.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file13665803259362820906.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file12199580591361002712.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file9539724297570849068.txt
-rw-r--r-- 1 root root 1.4K Sep  3 08:36 file12796507438937066046.txt
[root@ip-10-169-124-204 python]# cat /root/data/file13
file13640089980579634773.txt  file13665803259362820906.txt  file13765251820040647039.txt
[root@ip-10-169-124-204 python]# cat /root/data/file13
file13640089980579634773.txt  file13665803259362820906.txt  file13765251820040647039.txt
[root@ip-10-169-124-204 python]# cat /root/data/file13665803259362820906.txt

939866,vlkmed,535446,121865
982146,jeclka,706741,91921
880019,ctxvxf,391044,304214
688948,vgxixc,484076,363205
```

Step-3. How to Access and Initialize Dependencies for Spark Streaming on Spark shell:

How to Access:

You can access spark shell using below command on terminal
spark-shell

You can refer to this Unix log for the same.



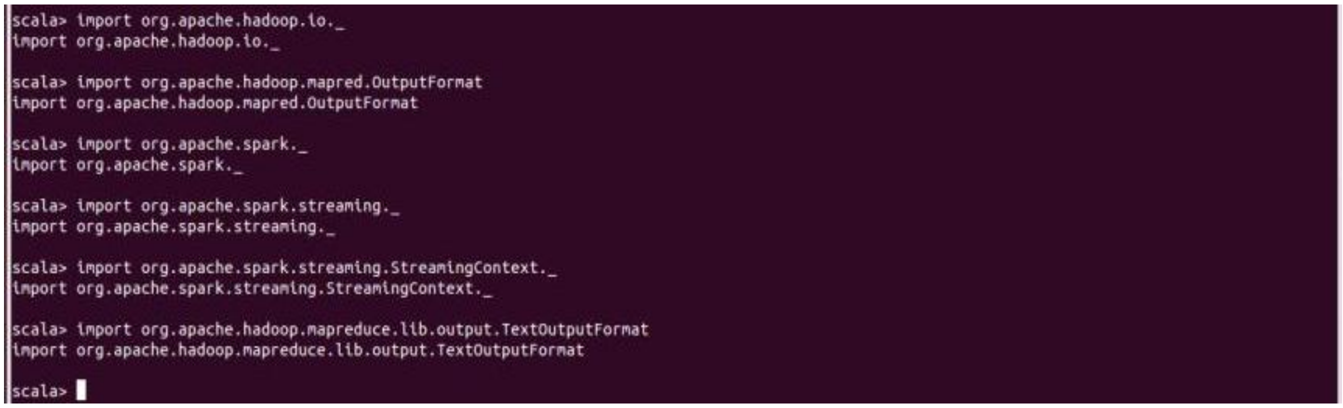
```
ubuntu@ip-10-169-221-108:~$ spark-shell
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.lib.MutableMetricsFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's repl log4j profile: org/apache/spark/log4j-defaults-repl.properties
To adjust logging level use sc.setLogLevel("INFO")
ubuntu@ip-10-169-221-108:~$ spark-sql
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
^CUsing Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
15/09/29 08:12:38 INFO ShutdownHookManager: Shutdown hook called
15/09/29 08:12:38 INFO ShutdownHookManager: Deleting directory /tmp/spark-2279d8fc-a876-4b8b-ad3f-426b67b5e0e0
ubuntu@ip-10-169-221-108:~$
```

How to build Dependencies for Spark Streaming :

Below listed imports statements are necessary for Spark Streaming. You can directory use these commands and run in Spark-Shell.

```
import org.apache.hadoop.io._
import org.apache.hadoop.mapred.OutputFormat
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat
```

You can refer to this Unix log for the same.



```
scala> import org.apache.hadoop.io._
import org.apache.hadoop.io._

scala> import org.apache.hadoop.mapred.OutputFormat
import org.apache.hadoop.mapred.OutputFormat

scala> import org.apache.spark._
import org.apache.spark._

scala> import org.apache.spark.streaming._
import org.apache.spark.streaming._

scala> import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.StreamingContext._

scala> import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat

scala>
```

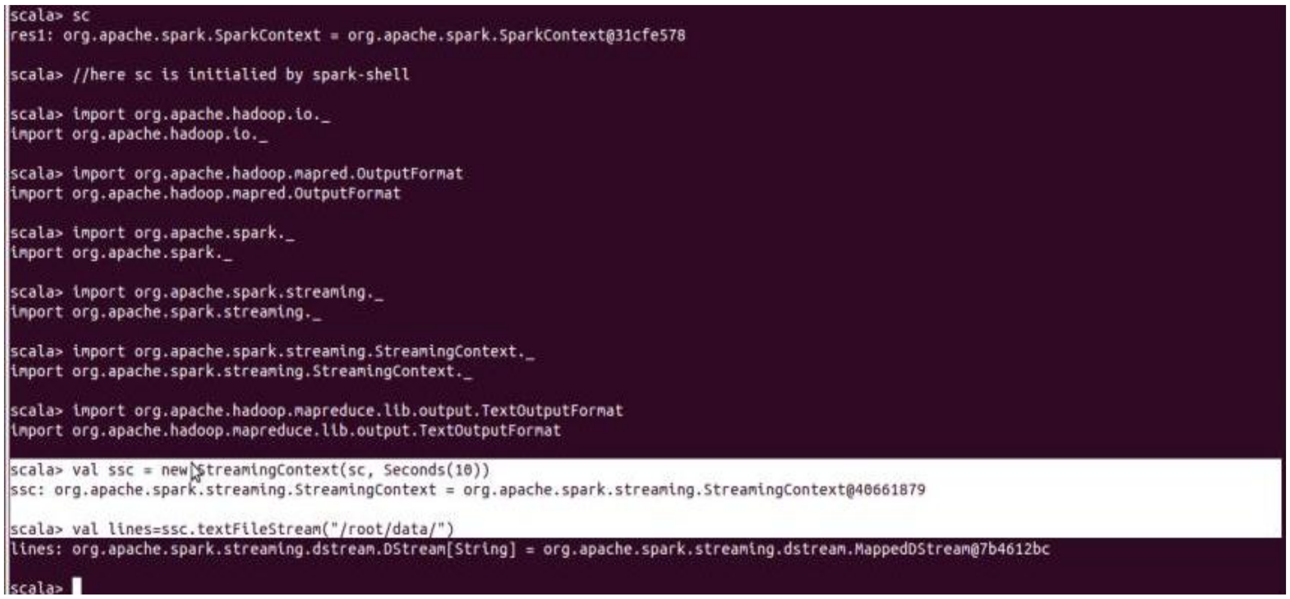
How to Initialize Spark Streaming:

You can use the following commands to do the same.

```
val ssc = new StreamingContext(sc, Seconds(10))
val lines=ssc.textFileStream("/root/data/")
```

Here sc is sparkContext, which has been initialized by spark-shell.

You can refer below the Unix log for the same.



```
scala> sc
res1: org.apache.spark.SparkContext = org.apache.spark.SparkContext@31cfe578

scala> //here sc is initialied by spark-shell

scala> import org.apache.hadoop.io._
import org.apache.hadoop.io._

scala> import org.apache.hadoop.mapred.OutputFormat
import org.apache.hadoop.mapred.OutputFormat

scala> import org.apache.spark._
import org.apache.spark._

scala> import org.apache.spark.streaming._
import org.apache.spark.streaming._

scala> import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.StreamingContext._

scala> import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat

scala> val ssc = new StreamingContext(sc, Seconds(10))
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@40661879

scala> val lines=ssc.textFileStream("/root/data/")
lines: org.apache.spark.streaming.dstream.DStream[String] = org.apache.spark.streaming.dstream.MappedDStream@7b4612bc

scala> 
```

Step-4. Live streaming data processing example using Spark Streaming on Spark-Shell :

on Spark shell:

Please follow the following steps:

- Start the data generate process using this command
  python generate_data.py & (refer to step-2)
- Make a folder using “mkdir /root/outputdata” to store output results

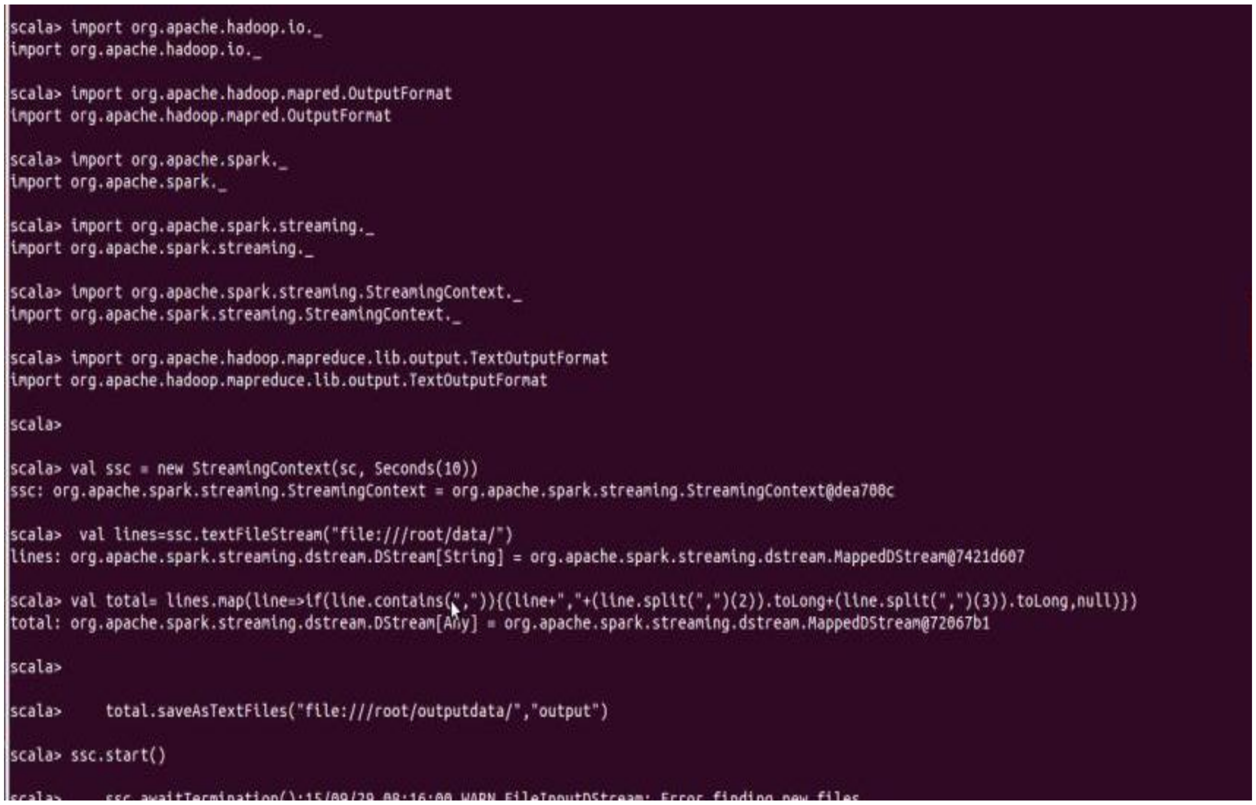
The below code simply adds run time the salaries and bonuses and create a employee monthly income result set. You can use the following code on Spark-Shell and execute.

```
import org.apache.hadoop.io._
import org.apache.hadoop.mapred.OutputFormat
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat

val ssc = new StreamingContext(sc, Seconds(10))
 val lines=ssc.textFileStream("file:///root/data/")
val total=
lines.map(line=>if(line.contains(",")){(line+","+(line.split(",")(2)).toLong+(line.split(",")(3)).t
oLong,null)}})
```

```
total.saveAsTextFiles("file:///root/outputdata/","output")
ssc.start()
ssc.awaitTermination();
```

Once “sss.start ” processes started on spark shell, you can see result on “/root/outputdata” folder. You might want to open two termnials to see data generation process and spark streaming real time calculations.

You can refer to the following Unix log for the same. Also, please check in your output folder for the output results.



```
scala> import org.apache.hadoop.io._
import org.apache.hadoop.io._

scala> import org.apache.hadoop.mapred.OutputFormat
import org.apache.hadoop.mapred.OutputFormat

scala> import org.apache.spark._
import org.apache.spark._

scala> import org.apache.spark.streaming._
import org.apache.spark.streaming._

scala> import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.StreamingContext._

scala> import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat

scala>

scala> val ssc = new StreamingContext(sc, Seconds(10))
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@dea700c

scala>  val lines=ssc.textFileStream("file:///root/data/")
lines: org.apache.spark.streaming.dstream.DStream[String] = org.apache.spark.streaming.dstream.MappedDStream@7421d607

scala> val total= lines.map(line=>if(line.contains(",")){(line+","+(line.split(",")(2)).toLong+(line.split(",")(3)).toLong,null)})
total: org.apache.spark.streaming.dstream.DStream[Any] = org.apache.spark.streaming.dstream.MappedDStream@72067b1

scala>

scala>     total.saveAsTextFiles("file:///root/outputdata/","output")

scala> ssc.start()

scala>     ssc.awaitTermination();15/09/29 08:16:00 WARN FileInputDStream: Error finding new files
```