Spark Streaming Case Study

There are two parts this case study
- **First Part** - You have to create a Spark Application which streams data from a file on local directory on your machine and does the word count on the fly.

The word should be done by the spark application in such a way that as soon as you drop the file in your local directory, your spark application should immediately do the word count for you.

- **Second Part** – In this part, you will have to create a Spark Application which should do the following

1. Pick up a file from the local directory and do the word count
2. Then in the same Spark Application, write the code to put the same file on HDFS.
3. Then in same Spark Application, do the word count of the file copied on HDFS in step 2
4. Lastly, compare the word count of step 1 and 2. Both should match, other throw an error

Below is the spark program for first case

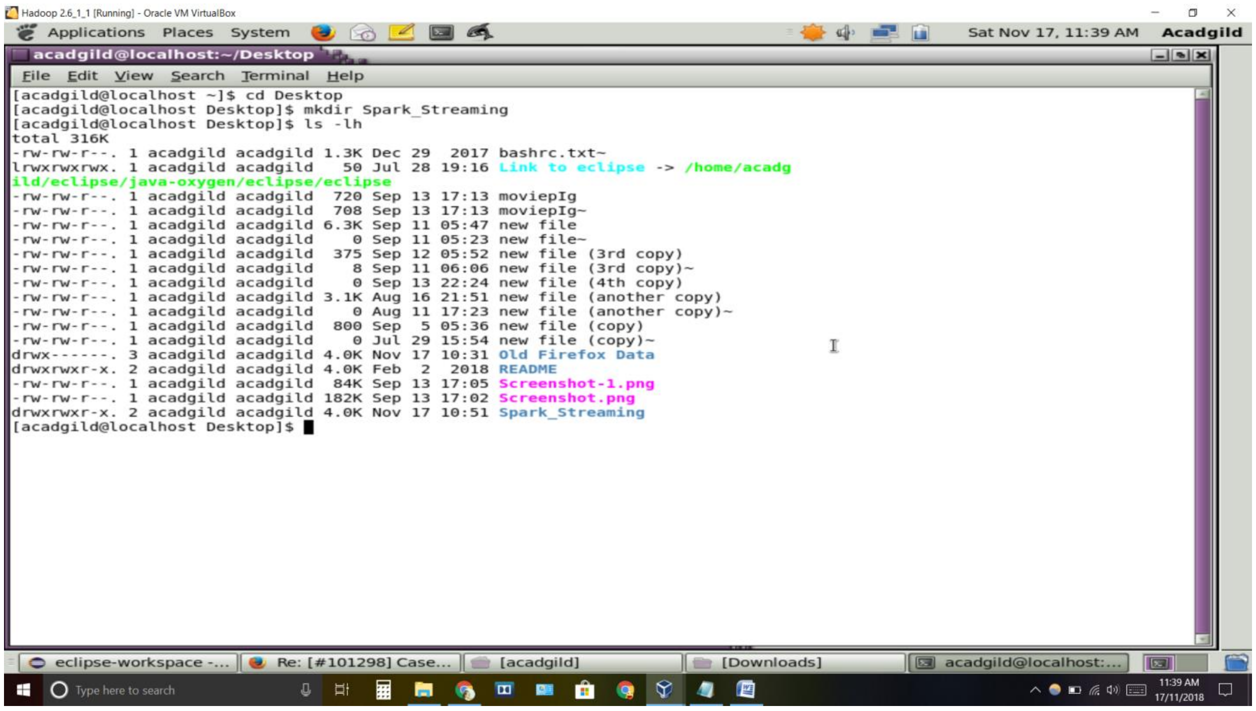
```
import org.apache.spark.{SparkConf, SparkContext}

import org.apache.spark.streaming.{Seconds, StreamingContext}

 import org.apache.log4j.{Level,Logger}

object SparkFileStreamingWordCount

 {

 def main(args: Array[String]): Unit = { println("hey Spark Streaming")

 val conf = new SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")

val sc = new SparkContext(conf)

val rootLogger =Logger.getRootLogger()

rootLogger.setLevel(Level.ERROR)

 // Create Streaming context to set batch duration 5 seconds

 val ssc = new StreamingContext(sc, Seconds(5))

//Create RDD for text file streaming by

val lines = ssc.textFileStream("/home/acadgild/Desktop/Spark_Streaming")

//Split each line into words val words = lines.flatMap(_.split(" "))

//Count each word in each batch
val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)

wordCounts.print()

//Start the computation

ssc.start()

//wait for the computation to terminate

ssc.awaitTermination()

 } }
```

Let us create a directory Spark_Streaming in the Desktop folder of home acadgild directory by using following command:

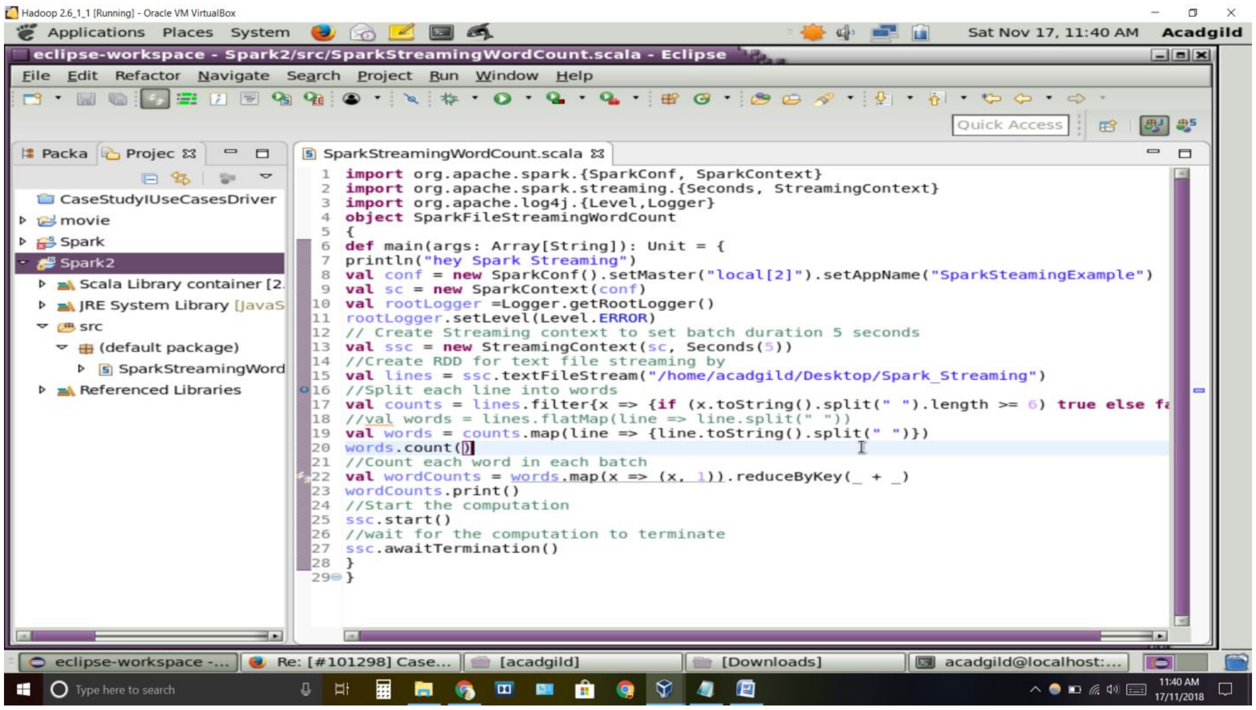
cd Desktop

mkdir Spark_Streaming

<u> Below screenshot shows the required directory in Desktop</u>

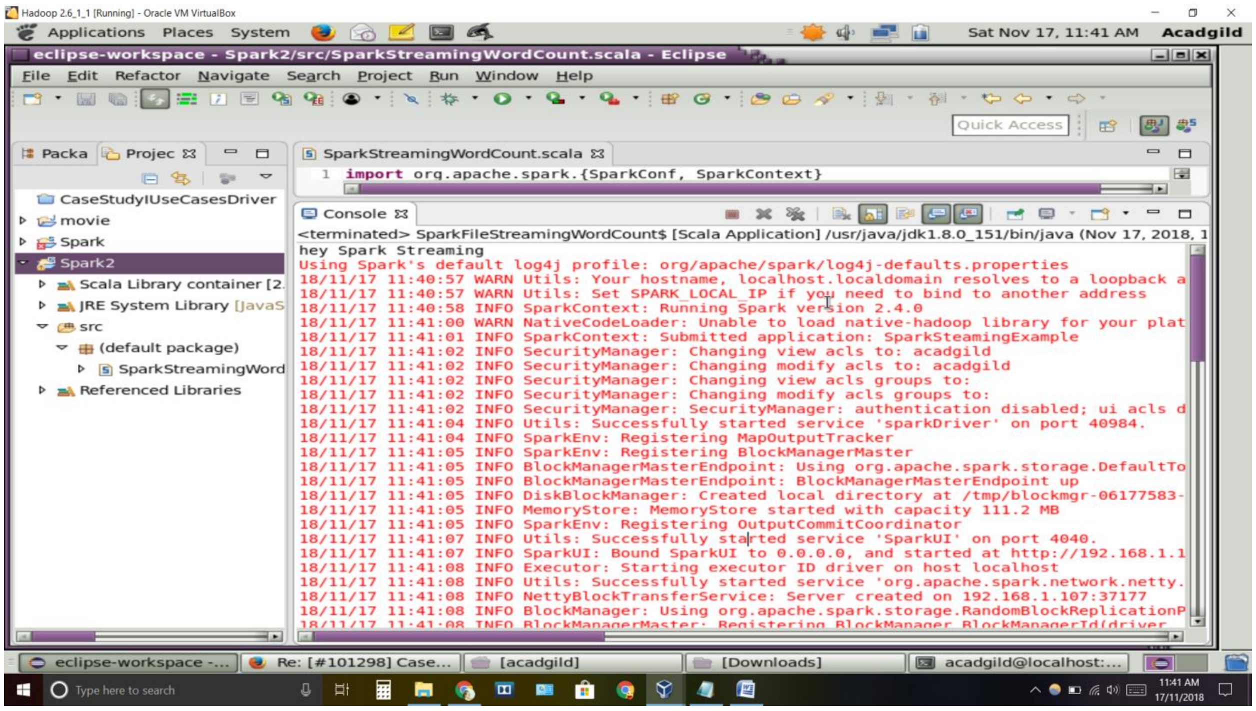


```
[acadgild@localhost ~]$ cd Desktop
[acadgild@localhost Desktop]$ mkdir Spark_Streaming
[acadgild@localhost Desktop]$ ls -lh
total 316K
-rw-rw-r--. 1 acadgild acadgild 1.3K Dec 29  2017 bashrc.txt~
lrwxrwxrwx. 1 acadgild acadgild   50 Jul 28 19:16 Link to eclipse -> /home/acadg
ild/eclipse/java-oxygen/eclipse/eclipse
-rw-rw-r--. 1 acadgild acadgild  720 Sep 13 17:13 moviepIg
-rw-rw-r--. 1 acadgild acadgild  708 Sep 13 17:13 moviepIg~
-rw-rw-r--. 1 acadgild acadgild 6.3K Sep 11 05:47 new file
-rw-rw-r--. 1 acadgild acadgild    0 Sep 11 05:23 new file~
-rw-rw-r--. 1 acadgild acadgild  375 Sep 12 05:52 new file (3rd copy)
-rw-rw-r--. 1 acadgild acadgild    8 Sep 11 06:06 new file (3rd copy)~
-rw-rw-r--. 1 acadgild acadgild    0 Sep 13 22:24 new file (4th copy)
-rw-rw-r--. 1 acadgild acadgild 3.1K Aug 16 21:51 new file (another copy)
-rw-rw-r--. 1 acadgild acadgild    0 Aug 11 17:23 new file (another copy)~
-rw-rw-r--. 1 acadgild acadgild  800 Sep  5 05:36 new file (copy)
-rw-rw-r--. 1 acadgild acadgild    0 Jul 29 15:54 new file (copy)~
drwx------. 3 acadgild acadgild 4.0K Nov 17 10:31 Old Firefox Data
drwxrwxr-x. 2 acadgild acadgild 4.0K Feb  2  2018 README
-rw-rw-r--. 1 acadgild acadgild  84K Sep 13 17:05 Screenshot-1.png
-rw-rw-r--. 1 acadgild acadgild 182K Sep 13 17:02 Screenshot.png
drwxrwxr-x. 2 acadgild acadgild 4.0K Nov 17 10:51 Spark_Streaming
[acadgild@localhost Desktop]$ 
```



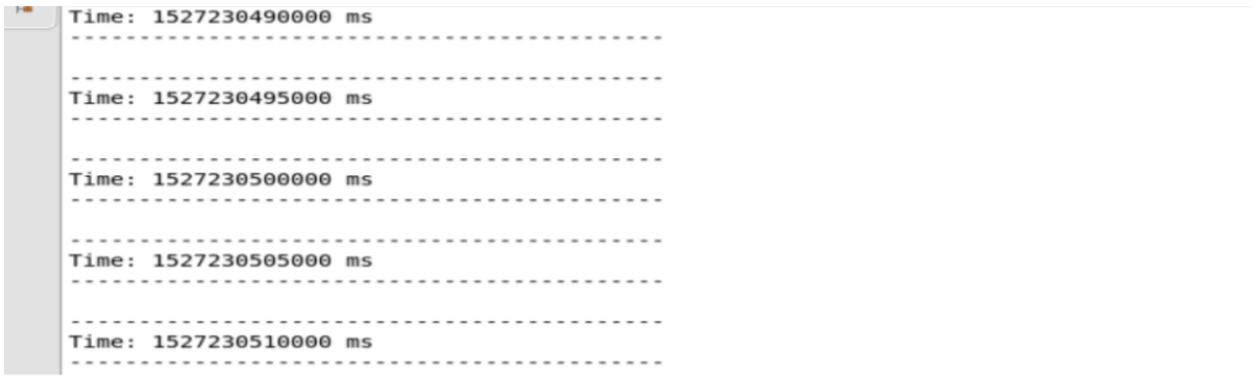
```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.log4j.{Level,Logger}
object SparkFileStreamingWordCount
{
def main(args: Array[String]): Unit = {
println("hey Spark Streaming")
val conf = new SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
val sc = new SparkContext(conf)
val rootLogger =Logger.getRootLogger()
rootLogger.setLevel(Level.ERROR)
// Create Streaming context to set batch duration 5 seconds
val ssc = new StreamingContext(sc, Seconds(5))
//Create RDD for text file streaming by
val lines = ssc.textFileStream("/home/acadgild/Desktop/Spark_Streaming")
//Split each line into words
val counts = lines.filter{x => {if (x.toString().split(" ").length >= 6) true else fa
//val words = lines.flatMap(line => line.split(" "))
val words = counts.map(line => {line.toString().split(" ")})
words.count()
//Count each word in each batch
val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
wordCounts.print()
//Start the computation
ssc.start()
//wait for the computation to terminate
ssc.awaitTermination()
}
}
```

Run the spark application:



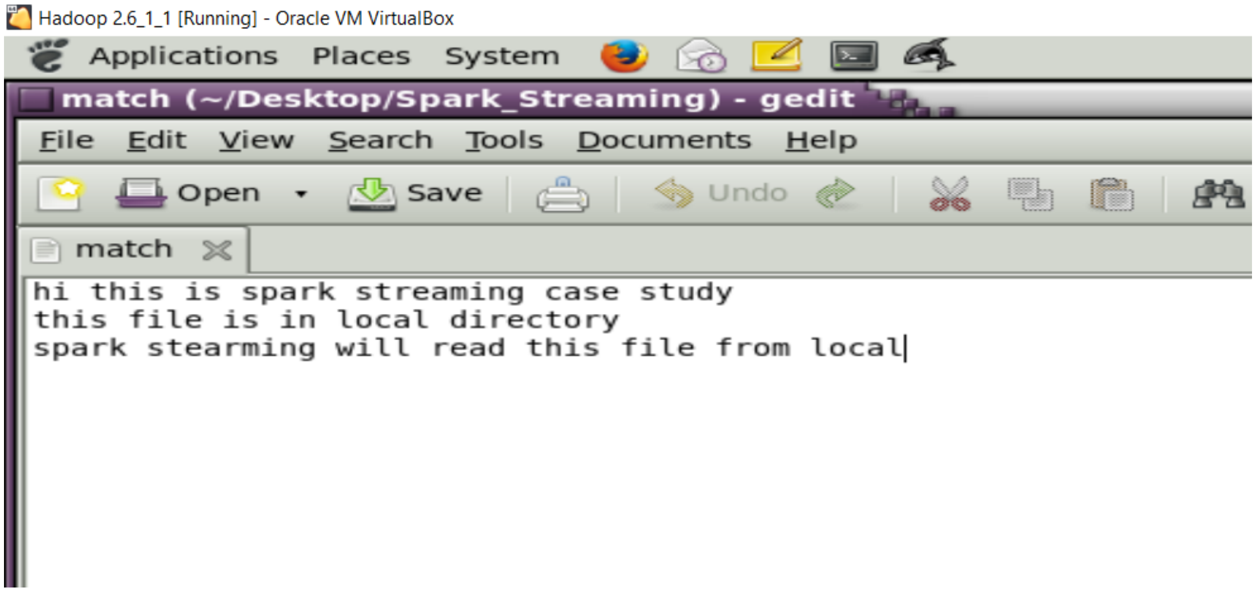
```
<terminated> SparkFileStreamingWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Nov 17, 2018, 1
hey Spark Streaming
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/11/17 11:40:57 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback a
18/11/17 11:40:57 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/11/17 11:40:58 INFO SparkContext: Running Spark version 2.4.0
18/11/17 11:41:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your plat
18/11/17 11:41:01 INFO SparkContext: Submitted application: SparkSteamingExample
18/11/17 11:41:02 INFO SecurityManager: Changing view acls to: acadgild
18/11/17 11:41:02 INFO SecurityManager: Changing modify acls to: acadgild
18/11/17 11:41:02 INFO SecurityManager: Changing view acls groups to:
18/11/17 11:41:02 INFO SecurityManager: Changing modify acls groups to:
18/11/17 11:41:02 INFO SecurityManager: SecurityManager: authentication disabled; ui acls d
18/11/17 11:41:04 INFO Utils: Successfully started service 'sparkDriver' on port 40984.
18/11/17 11:41:04 INFO SparkEnv: Registering MapOutputTracker
18/11/17 11:41:05 INFO SparkEnv: Registering BlockManagerMaster
18/11/17 11:41:05 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTo
18/11/17 11:41:05 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/11/17 11:41:05 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-06177583-
18/11/17 11:41:05 INFO MemoryStore: MemoryStore started with capacity 111.2 MB
18/11/17 11:41:05 INFO SparkEnv: Registering OutputCommitCoordinator
18/11/17 11:41:07 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/11/17 11:41:07 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.1.1
18/11/17 11:41:08 INFO Executor: Starting executor ID driver on host localhost
18/11/17 11:41:08 INFO Utils: Successfully started service 'org.apache.spark.network.netty.
18/11/17 11:41:08 INFO NettyBlockTransferService: Server created on 192.168.1.107:37177
18/11/17 11:41:08 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationP
18/11/17 11:41:08 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver
```

In below screenshot we are able to see that spark streaming is running every 5 seconds



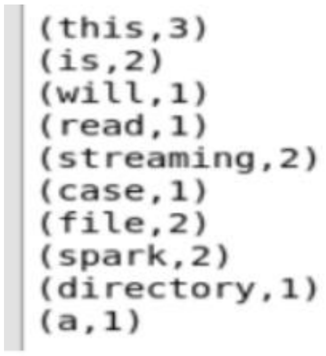
```
Time: 1527230490000 ms
-------------------------------------------

-------------------------------------------
Time: 1527230495000 ms
-------------------------------------------

-------------------------------------------
Time: 1527230500000 ms
-------------------------------------------

-------------------------------------------
Time: 1527230505000 ms
-------------------------------------------

-------------------------------------------
Time: 1527230510000 ms
-------------------------------------------
```

<u>Now we put a blank text file 'match' and then putting some words in this file</u>


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications
Places
System
match (~/Desktop/Spark_Streaming) - gedit
File
Edit
View
Search
Tools
Documents
Help
Open
Save
Undo
match

```
hi this is spark streaming case study
this file is in local directory
spark stearming will read this file from local
```

<u>Output of word count in console:</u>

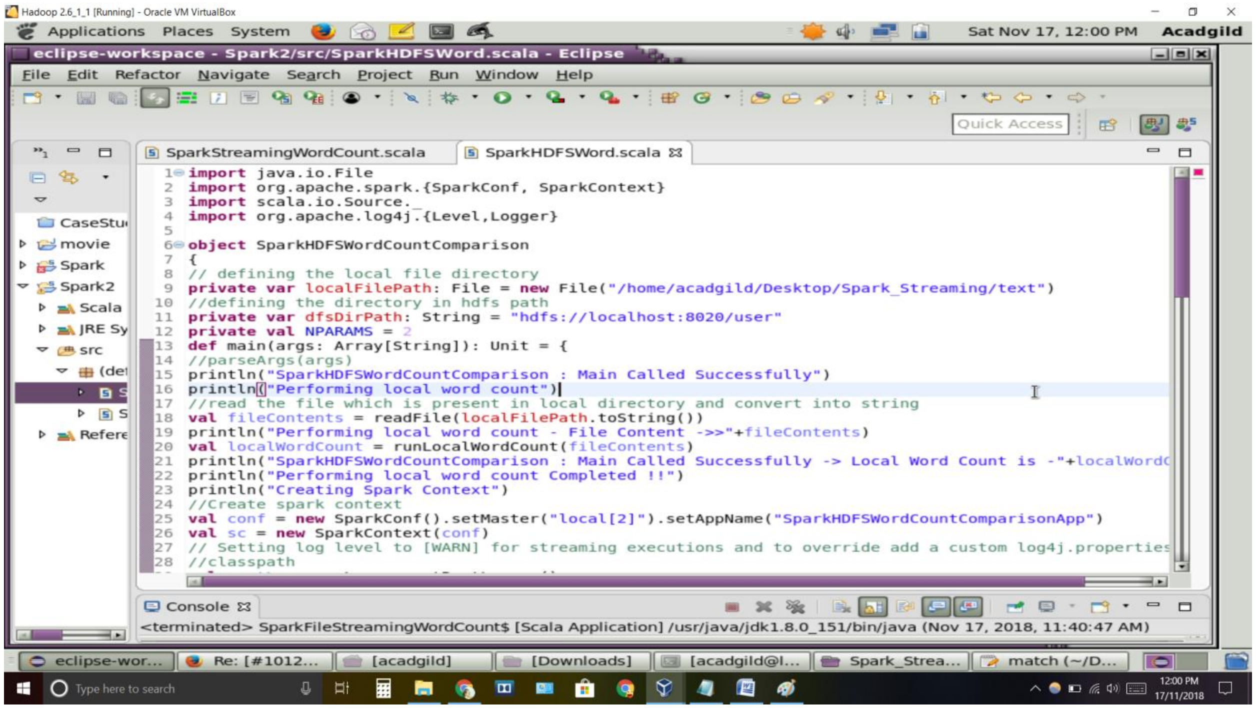


```
(this,3)
(is,2)
(will,1)
(read,1)
(streaming,2)
(case,1)
(file,2)
(spark,2)
(directory,1)
(a,1)
```

Above Screenshot shows the word count of each words in text file from local file directory

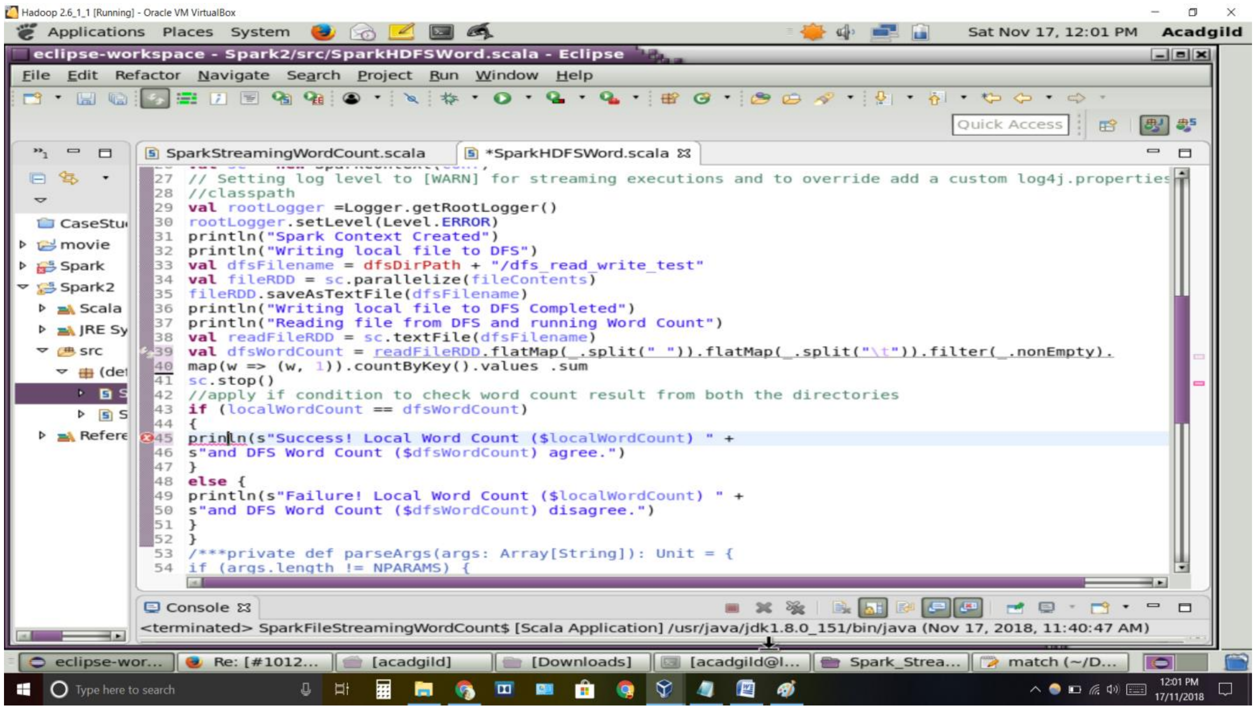
Below is the spark programming for second case

```
 import java.io.File

 import org.apache.spark.{SparkConf, SparkContext}

import scala.io.Source._

import org.apache.log4j.{Level,Logger}

object SparkHDFSWordCountComparison

 {

 // defining the local file directory

private var localFilePath: File = new File("/home/acadgild/Desktop/Spark_Streaming/text")

 //defining the directory in hdfs path

 private var dfsDirPath: String = "hdfs://localhost:8020/user"

private val NPARAMS = 2

def main(args: Array[String]): Unit = {

//parseArgs(args)

println("SparkHDFSWordCountComparison : Main Called Successfully")

 println("Performing local word count")

 //read the file which is present in local directory and convert into string

 val fileContents = readFile(localFilePath.toString())

println("Performing local word count - File Content ->>"+fileContents)

val localWordCount = runLocalWordCount(fileContents)

 println("SparkHDFSWordCountComparison : Main Called Successfully -> Local Word Count
is - >>" +localWordCount)

println("Performing local word count Completed !!")

println("Creating Spark Context")

 //Create spark context

val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkHDFSWordCountComparisonApp")

val sc = new SparkContext(conf      )
// Setting log level to [WARN] for streaming executions and to override add a custom
log4j.properties to the //classpath

val rootLogger =Logger.getRootLogger()

rootLogger.setLevel(Level.ERROR)

println("Spark Context Created")

println("Writing local file to DFS")

val dfsFilename = dfsDirPath + "/dfs_read_write_test"

 val fileRDD = sc.parallelize(fileContents) fileRDD.saveAsTextFile(dfsFilename)

println("Writing local file to DFS Completed")

println("Reading file from DFS and running Word Count")

val readFileRDD = sc.textFile(dfsFilename)

 val dfsWordCount = readFileRDD .flatMap(_.split(" ")) .flatMap(_.split("\t"))
.filter(_.nonEmpty) .map(w => (w, 1)) .countByKey() .values .sum


sc.stop()

//apply if condition to check word count result from both the directories

if (localWordCount == dfsWordCount)

{

 println(s"Success! Local Word Count ($localWordCount) " + s"

and DFS Word Count ($dfsWordCount) agree.") }

 else {

println(s"Failure! Local Word Count ($localWordCount) " + s"

and DFS Word Count ($dfsWordCount) disagree.")

} }

 /***private def parseArgs(args: Array[String]): Unit =

 { if (args.length != NPARAMS) {

 printUsage()

 System.exit(1) } }***/
 private def printUsage(): Unit = {
 val usage: String = "DFS Read-Write Test\n"
 + "\n"
+ "Usage: localFile dfsDir\n"
+ "\n" + "localFile - (string) local file to use in test\n"
 + "dfsDir - (string) DFS directory for read/write tests\n"
println(usage) }
private def readFile(filename: String): List[String] = {
 val lineIter: Iterator[String] = fromFile(filename).getLines()
val lineList: List[String] = lineIter.toList lineList }
 def runLocalWordCount(fileContents: List[String]): Int =
{ fileContents.flatMap(_.split(" "))
.flatMap(_.split("\t"))
.filter(_.nonEmpty)
.groupBy(w => w)
.mapValues(_.size)
.values
.sum }
}
```


eclipse-workspace - Spark2/src/SparkHDFSWord.scala - Eclipse

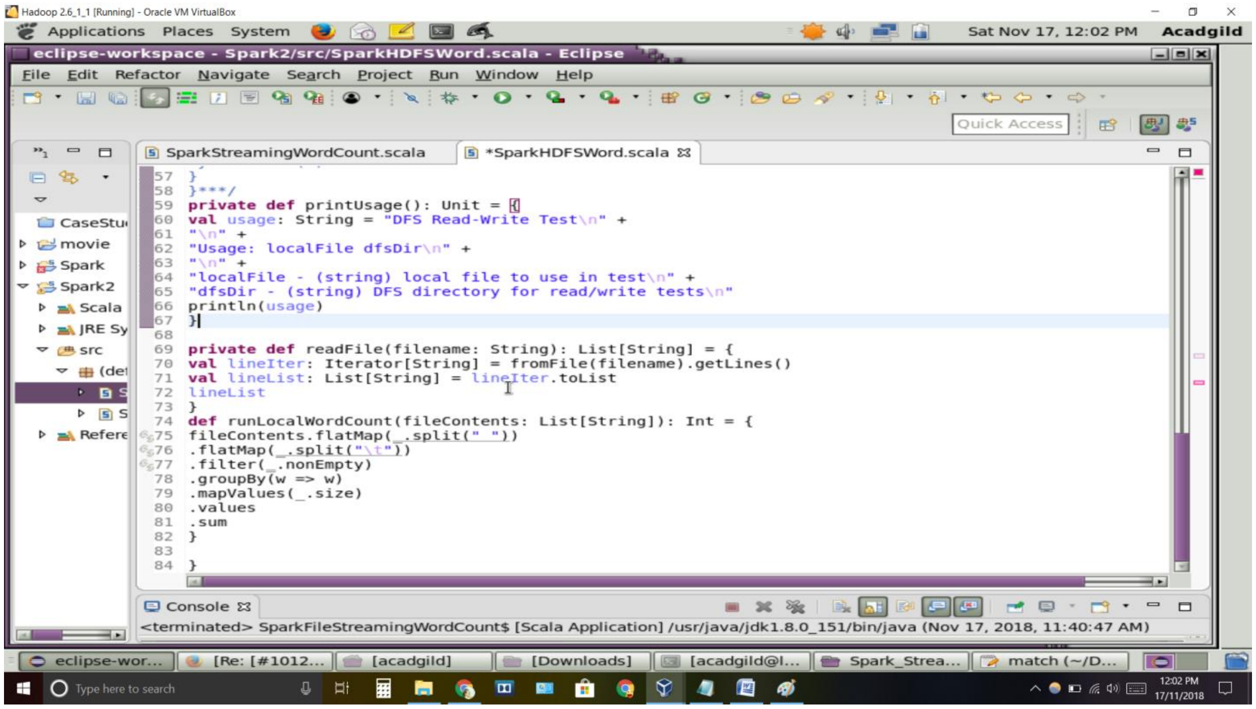
```
import java.io.File
import org.apache.spark.{SparkConf, SparkContext}
import scala.io.Source._
import org.apache.log4j.{Level,Logger}

object SparkHDFSWordCountComparison
{
// defining the local file directory
private var localFilePath: File = new File("/home/acadgild/Desktop/Spark_Streaming/text")
//defining the directory in hdfs path
private var dfsDirPath: String = "hdfs://localhost:8020/user"
private val NPARAMS = 2
def main(args: Array[String]): Unit = {
//parseArgs(args)
println("SparkHDFSWordCountComparison : Main Called Successfully")
println("Performing local word count")
//read the file which is present in local directory and convert into string
val fileContents = readFile(localFilePath.toString())
println("Performing local word count - File Content ->>"+fileContents)
val localWordCount = runLocalWordCount(fileContents)
println("SparkHDFSWordCountComparison : Main Called Successfully -> Local Word Count is -"+localWordC
println("Performing local word count Completed !!")
println("Creating Spark Context")
//Create spark context
val conf = new SparkConf().setMaster("local[2]").setAppName("SparkHDFSWordCountComparisonApp")
val sc = new SparkContext(conf)
// Setting log level to [WARN] for streaming executions and to override add a custom log4j.properties
//classpath
```

```
<terminated> SparkFileStreamingWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Nov 17, 2018, 11:40:47 AM)
```


eclipse-workspace - Spark2/src/SparkHDFSWord.scala - Eclipse

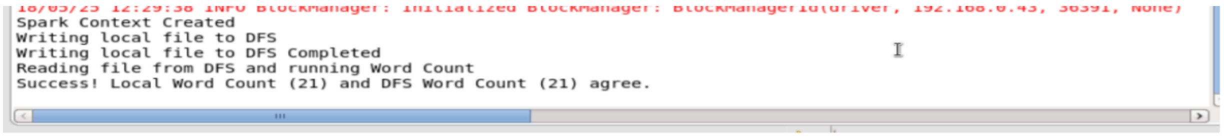
```
// Setting log level to [WARN] for streaming executions and to override add a custom log4j.properties
//classpath
val rootLogger =Logger.getRootLogger()
rootLogger.setLevel(Level.ERROR)
println("Spark Context Created")
println("Writing local file to DFS")
val dfsFilename = dfsDirPath + "/dfs_read_write_test"
val fileRDD = sc.parallelize(fileContents)
fileRDD.saveAsTextFile(dfsFilename)
println("Writing local file to DFS Completed")
println("Reading file from DFS and running Word Count")
val readFileRDD = sc.textFile(dfsFilename)
val dfsWordCount = readFileRDD.flatMap(_.split(" ")).flatMap(_.split("\t")).filter(_.nonEmpty).
map(w => (w, 1)).countByKey().values .sum
sc.stop()
//apply if condition to check word count result from both the directories
if (localWordCount == dfsWordCount)
{
prinln(s"Success! Local Word Count ($localWordCount) " +
s"and DFS Word Count ($dfsWordCount) agree.")
}
else {
println(s"Failure! Local Word Count ($localWordCount) " +
s"and DFS Word Count ($dfsWordCount) disagree.")
}
}
/***private def parseArgs(args: Array[String]): Unit = {
if (args.length != NPARAMS) {
```

```
<terminated> SparkFileStreamingWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Nov 17, 2018, 11:40:47 AM)
```



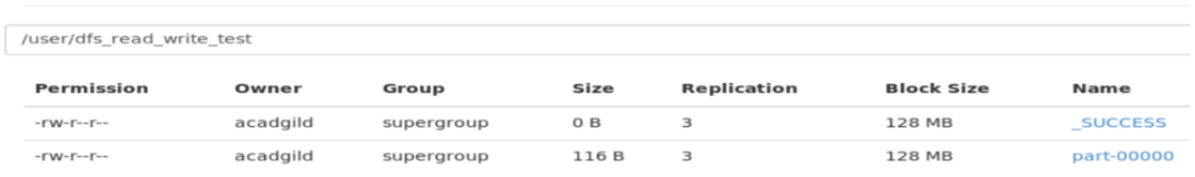
```
}
}***/
private def printUsage(): Unit = {
val usage: String = "DFS Read-Write Test\n" +
"\n" +
"Usage: localFile dfsDir\n" +
"\n" +
"localFile - (string) local file to use in test\n" +
"dfsDir - (string) DFS directory for read/write tests\n"
println(usage)
}

private def readFile(filename: String): List[String] = {
val lineIter: Iterator[String] = fromFile(filename).getLines()
val lineList: List[String] = lineIter.toList
lineList
}
def runLocalWordCount(fileContents: List[String]): Int = {
fileContents.flatMap(_.split(" "))
.flatMap(_.split("\t"))
.filter(_.nonEmpty)
.groupBy(w => w)
.mapValues(_.size)
.values
.sum
}

}
```

```
<terminated> SparkFileStreamingWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Nov 17, 2018, 11:40:47 AM)
```



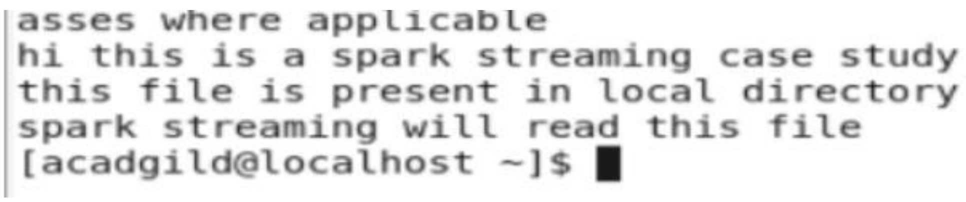
```
Spark Context Created
Writing local file to DFS
Writing local file to DFS Completed
Reading file from DFS and running Word Count
Success! Local Word Count (21) and DFS Word Count (21) agree.
```

Above screen shot shows that, word count of file from local file directory and HDFS directory file is same. Output file in HDFS directory shown below in screen shot.



/user/dfs_read_write_test

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	acadgild	supergroup	0 B	3	128 MB	_SUCCESS
-rw-r--r--	acadgild	supergroup	116 B	3	128 MB	part-00000



```
asses where applicable
hi this is a spark streaming case study
this file is present in local directory
spark streaming will read this file
[acadgild@localhost ~]$
```

Above screen, shot shows the content of the output file **part-00000** which same as the file present in the local directory