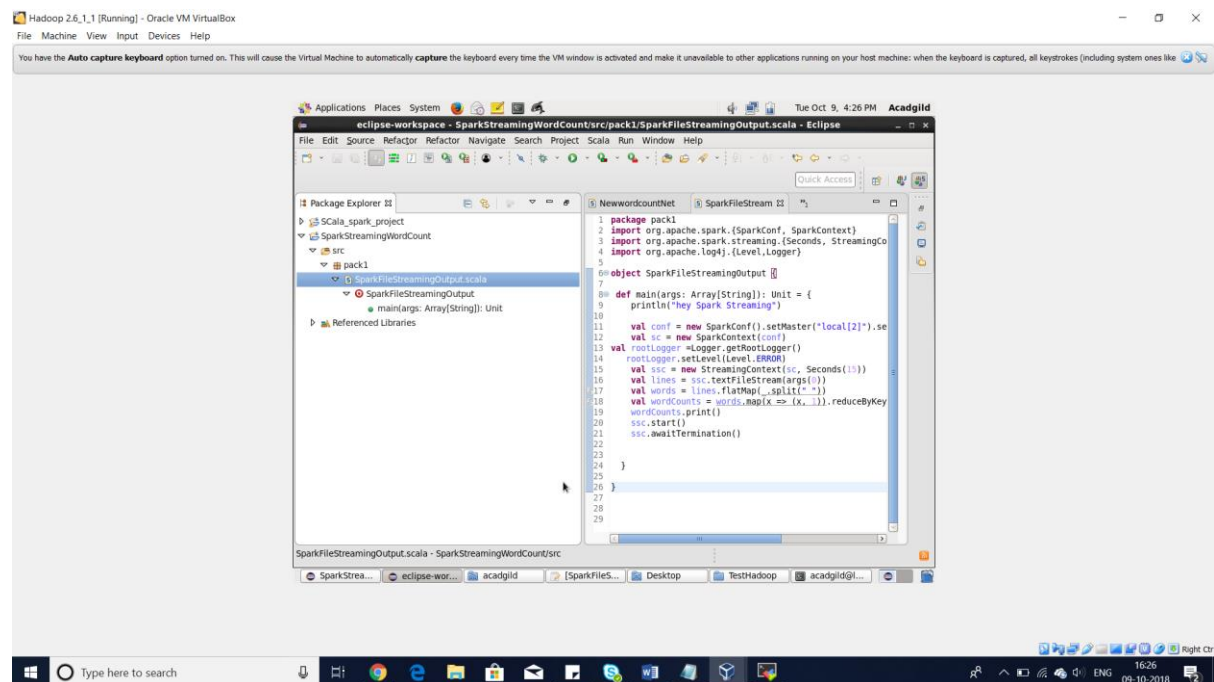


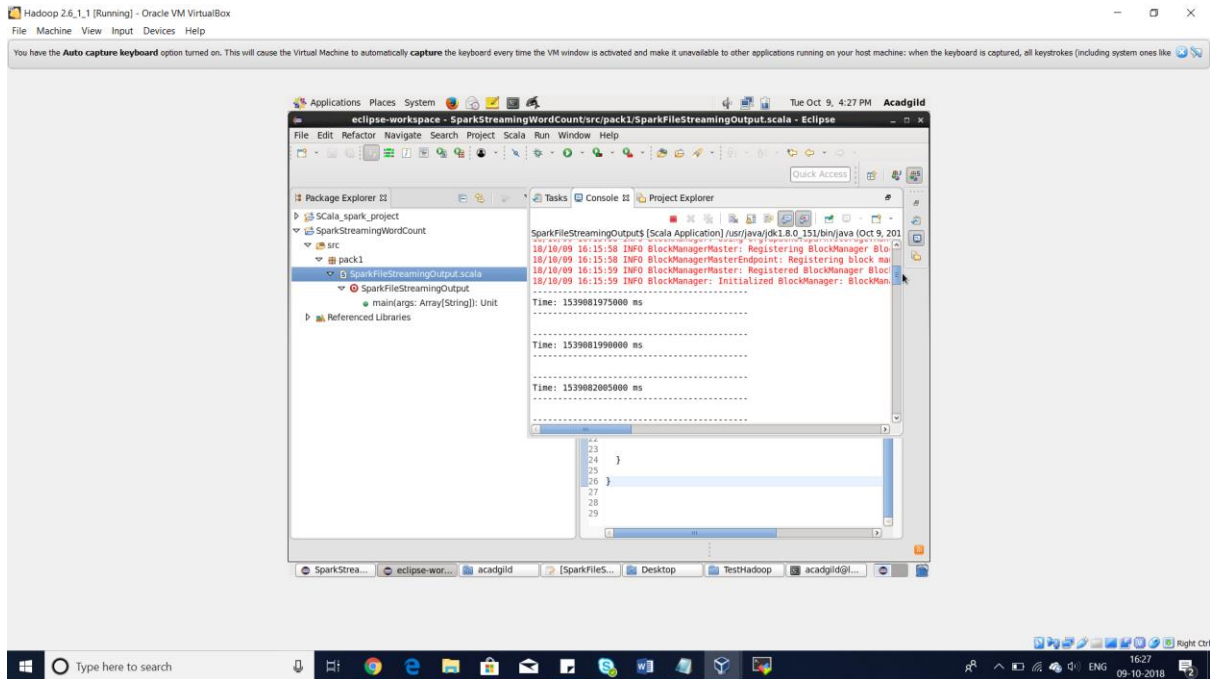
There are two parts this case study

**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



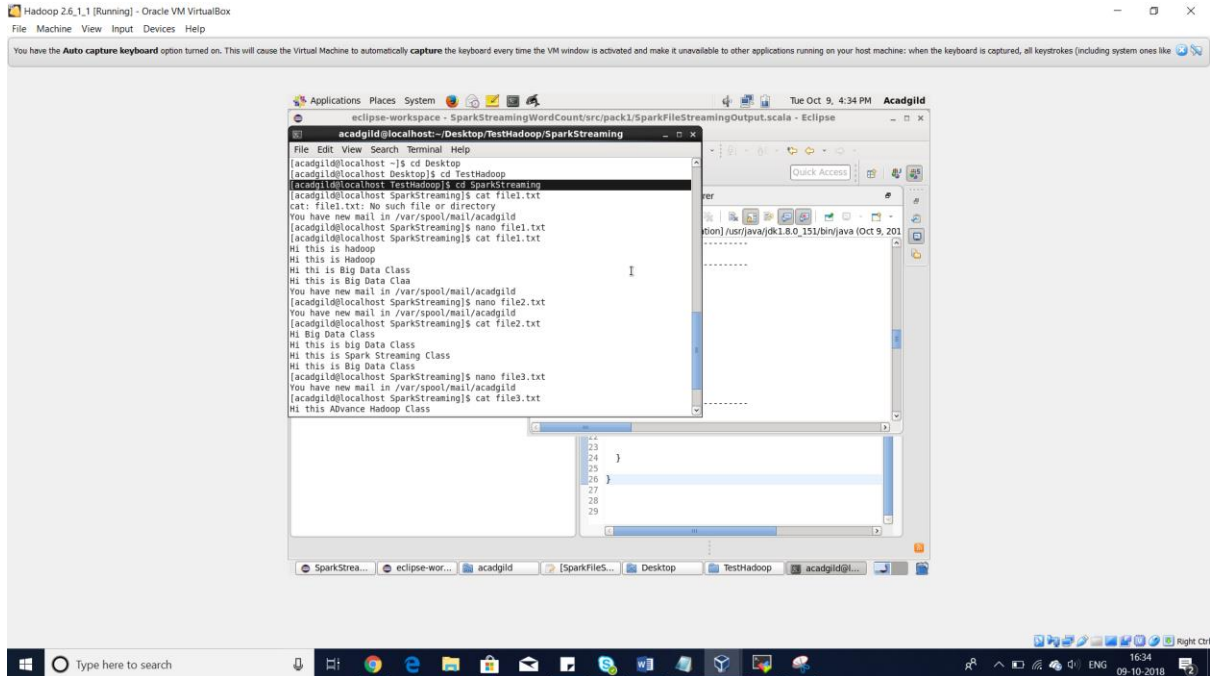


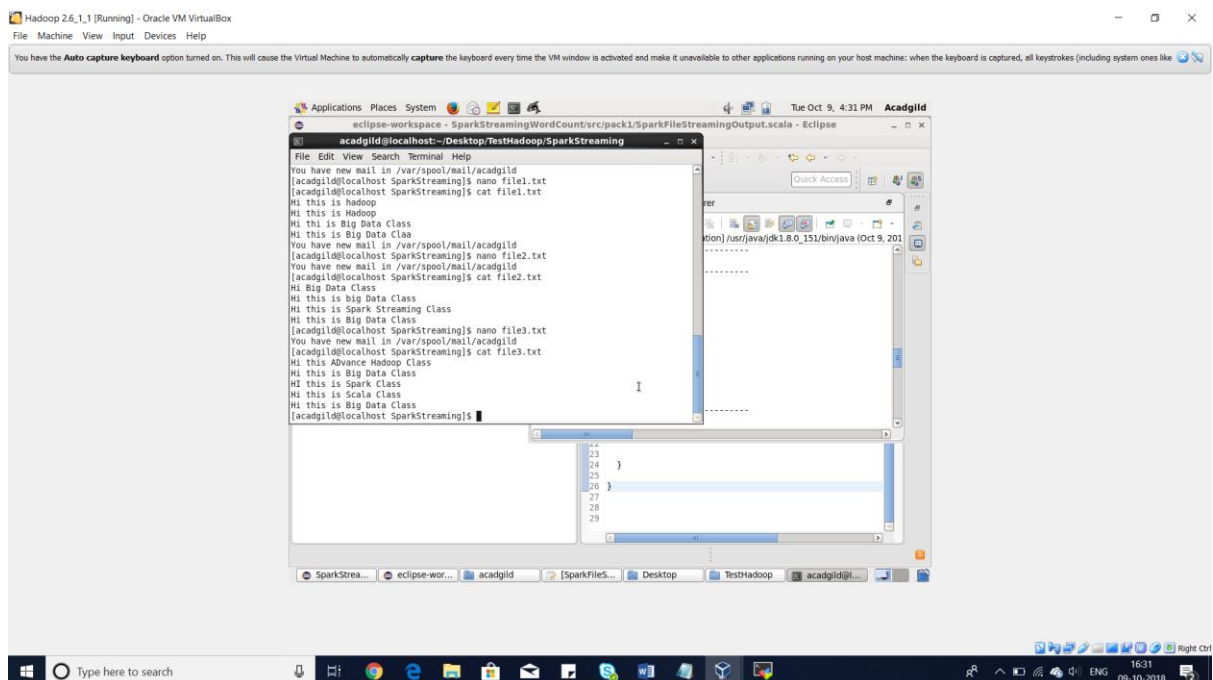
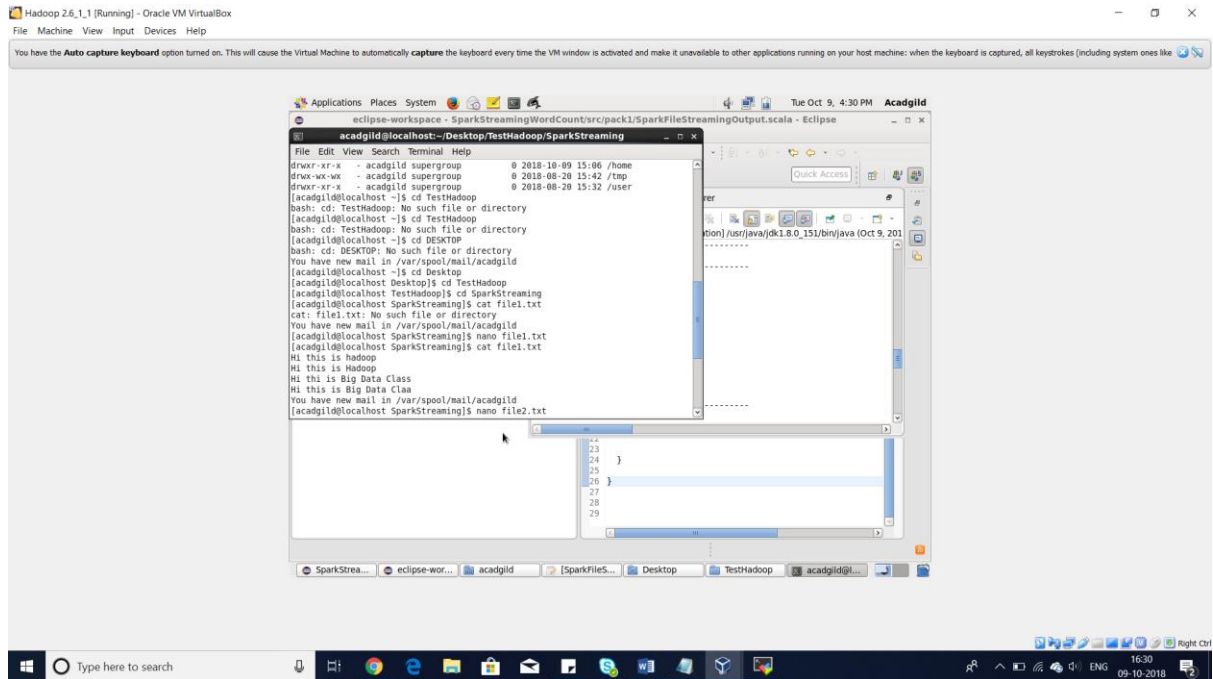
The Directory created for streaming file is: /home/acadgild/Desktop/TestHadoop/SparkStreaming

```

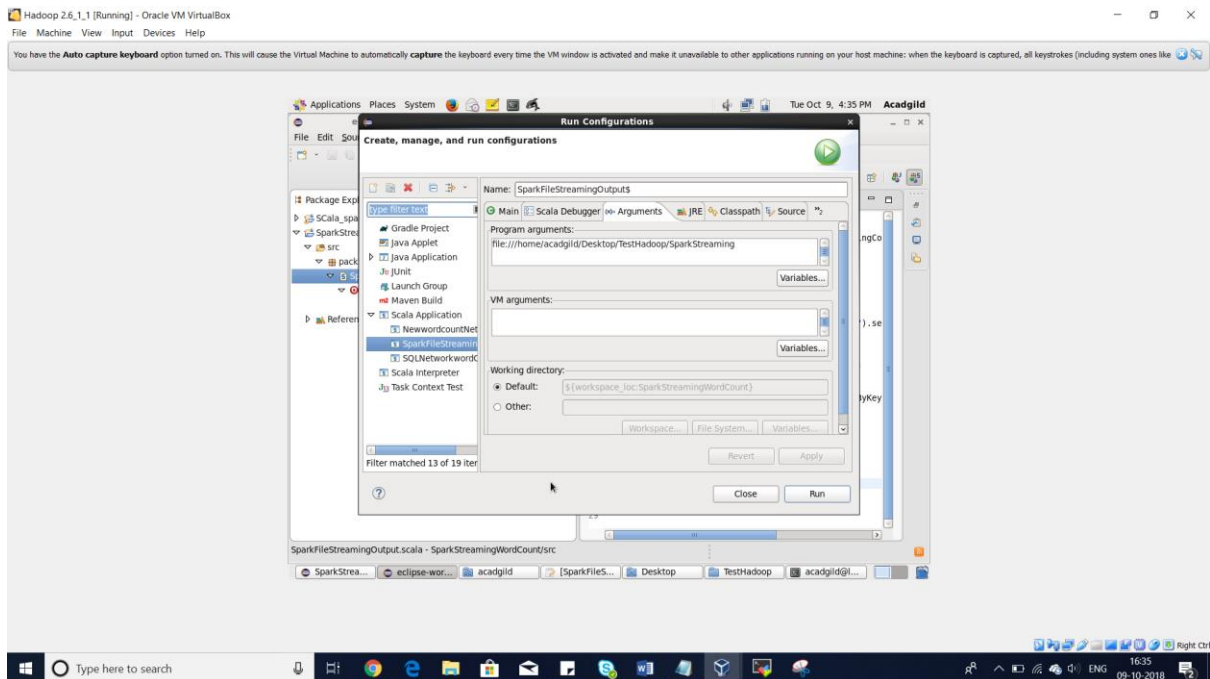
acadgild@localhost: ~/Desktop/TestHadoop/sparkStreaming
File Edit View Search Terminal Help
[acadgild@localhost sparkStreaming]$ pwd
/home/acadgild/Desktop/TestHadoop/sparkStreaming
[acadgild@localhost sparkStreaming]$ ll
total 0
[acadgild@localhost sparkStreaming]$

```

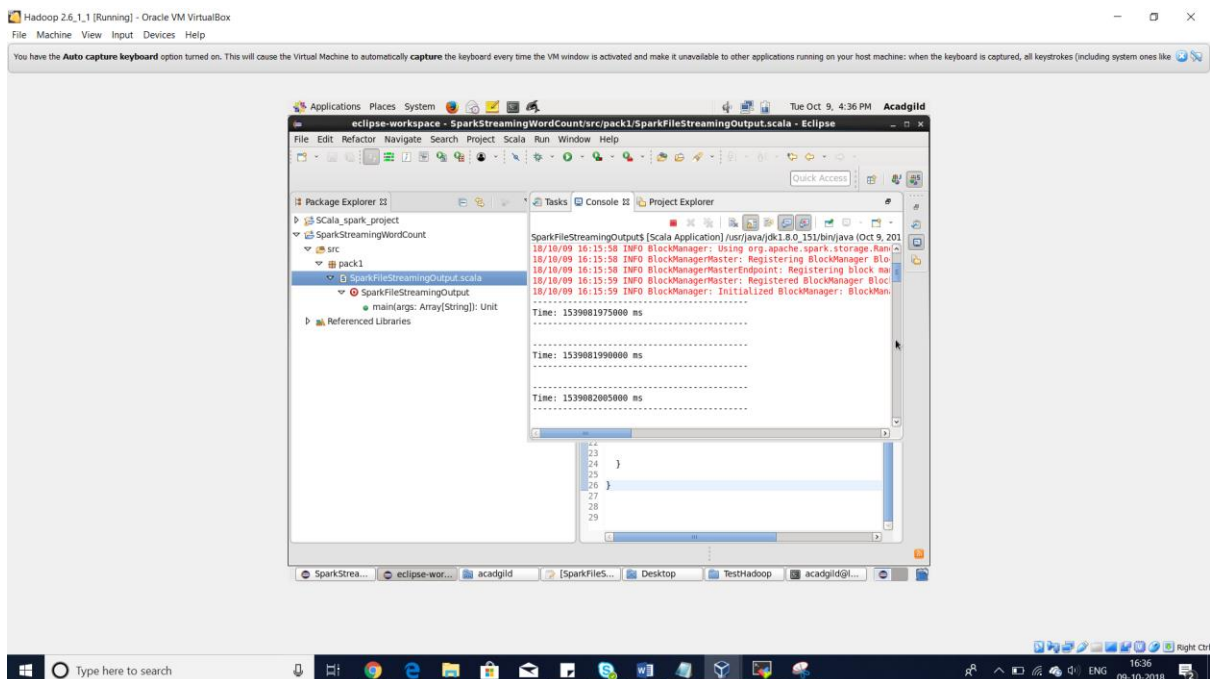




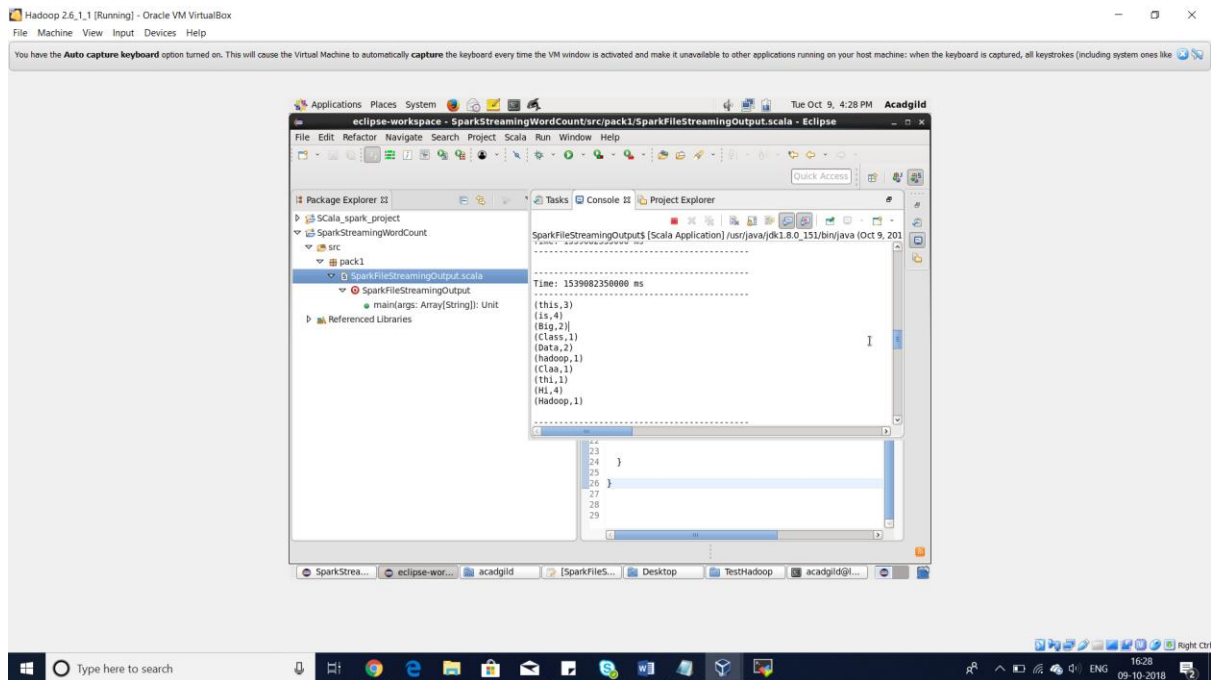
## Specifying the above created directory as input to program arguments



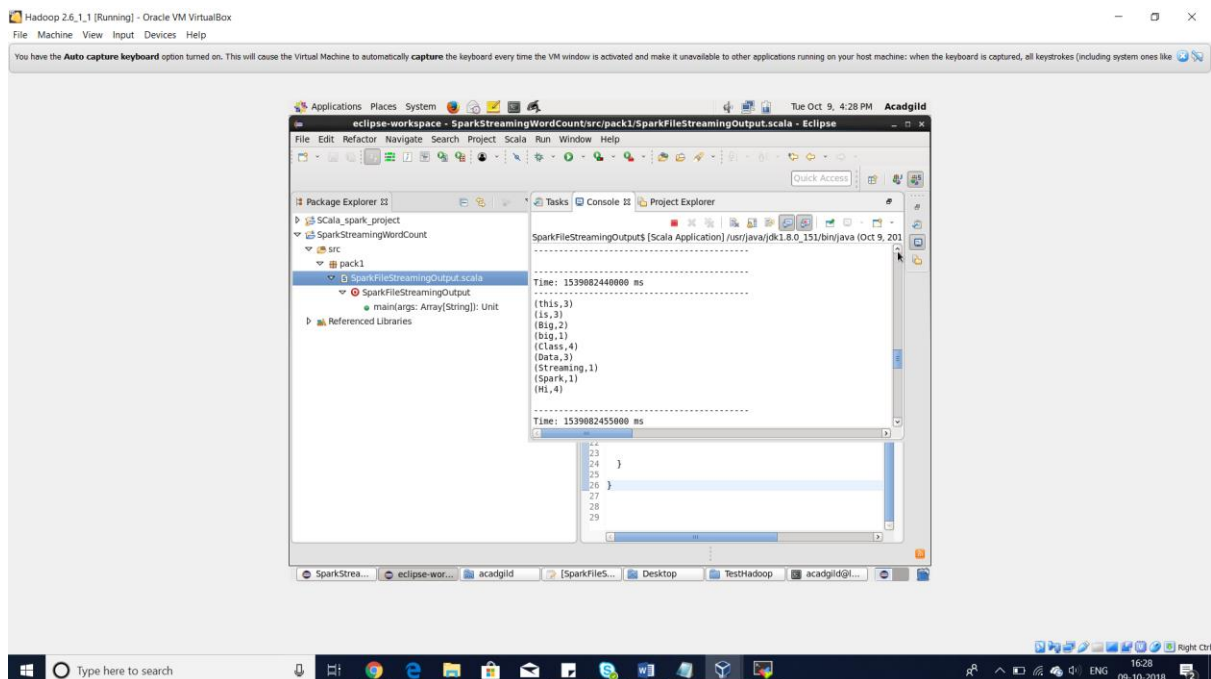
arg(0) specified within the program reads entire files that are getting added to the above directory & performs the word count operation on the fly. Run the application as Scala Application.

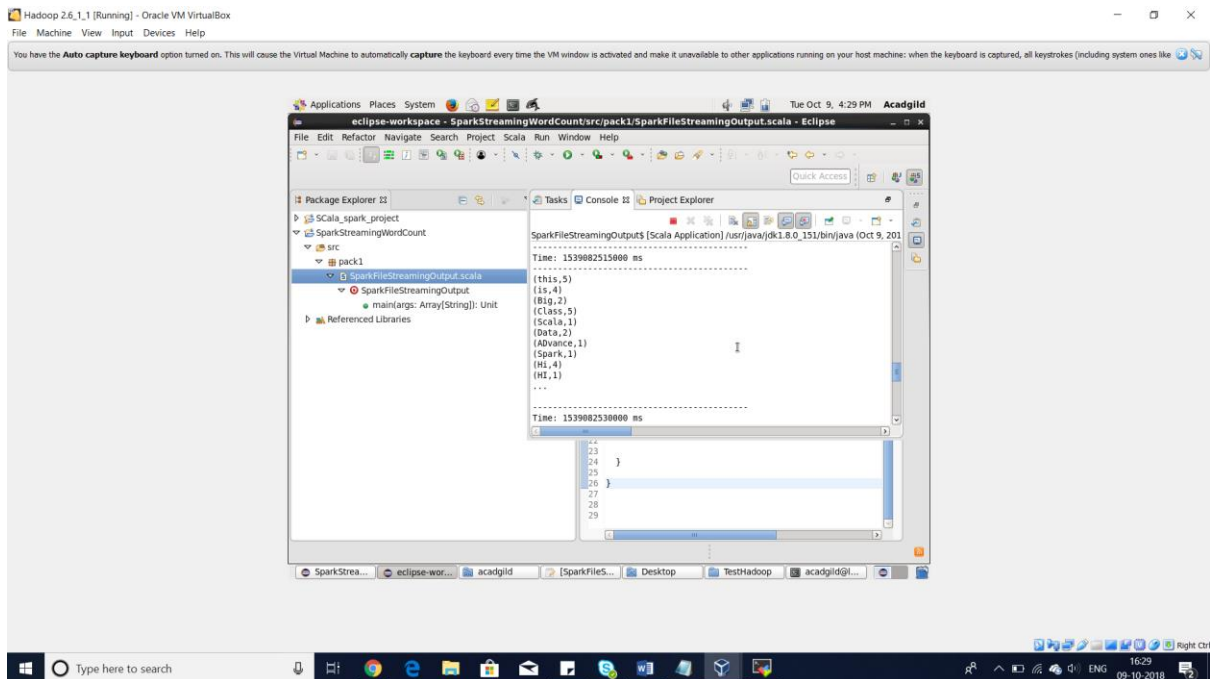


The contents of file1.txt getting read by spark streaming application and computing word count on the fly.



Similarly, creating two other files file2.txt & file3.txt whose words would be counted by spark streaming application on the fly





Objective 2: Second Part - In this part, you will have to create a Spark Application which should do the following:

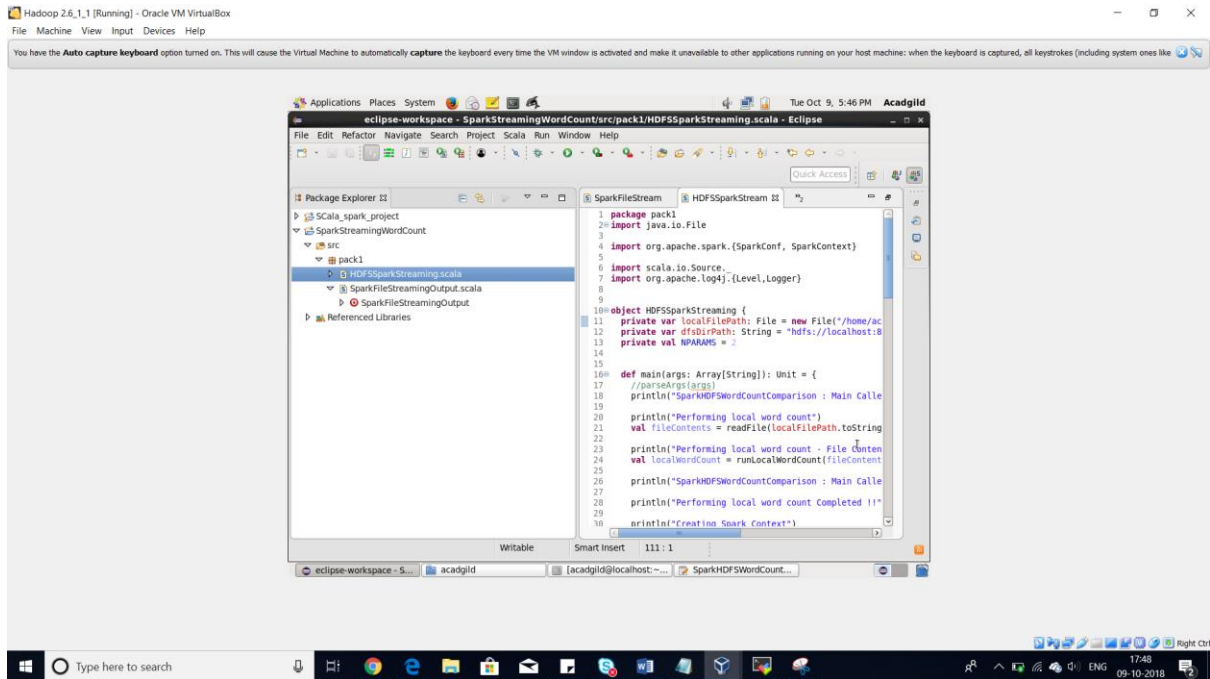
1. Pick up a file from the local directory and do the word count
2. Then in the same Spark Application, write the code to put the same file on HDFS.
3. Then in same Spark Application, do the word count of the file copied on HDFS in step 2
4. Lastly, compare the word count of step 1 and 2. Both should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

Solution:

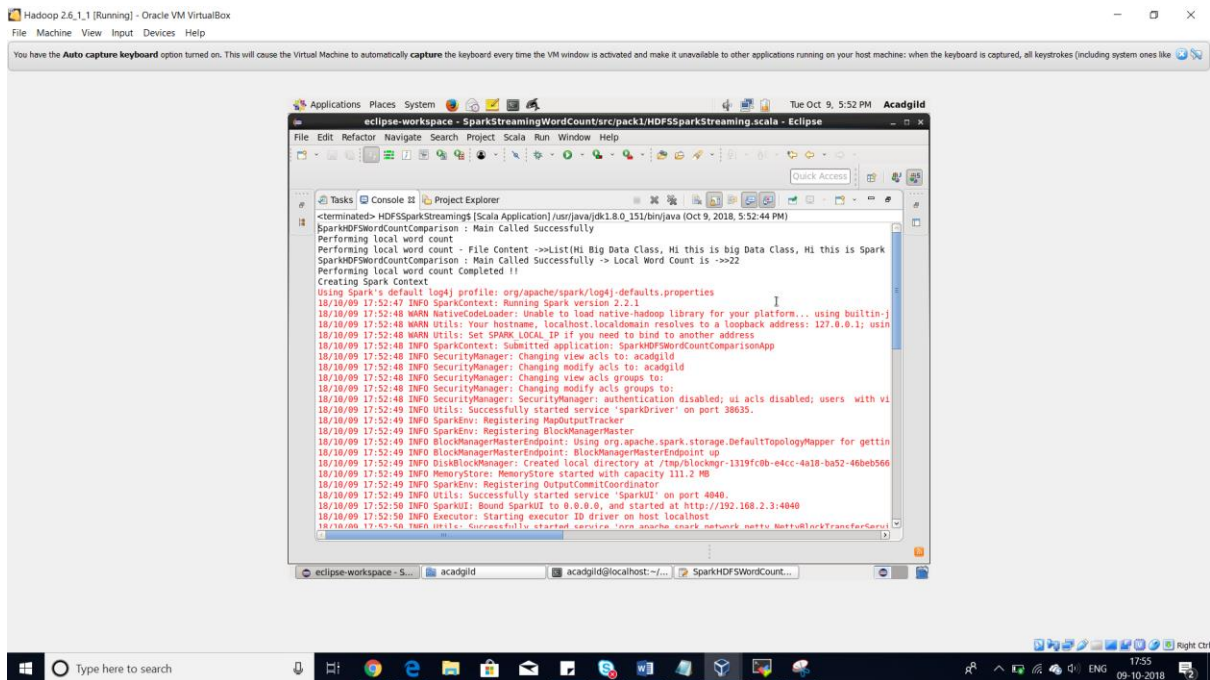
HDFS does not contain streaming directory before the application is run





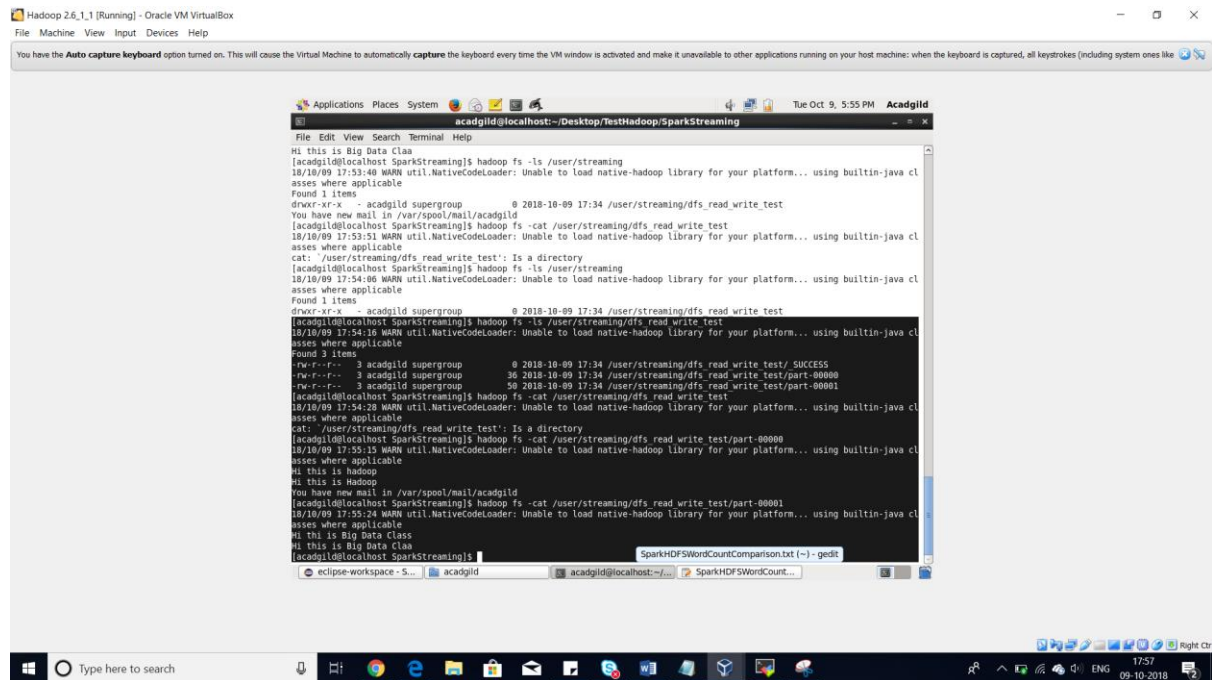


## 1. Pick up a file from the local directory and do the word count



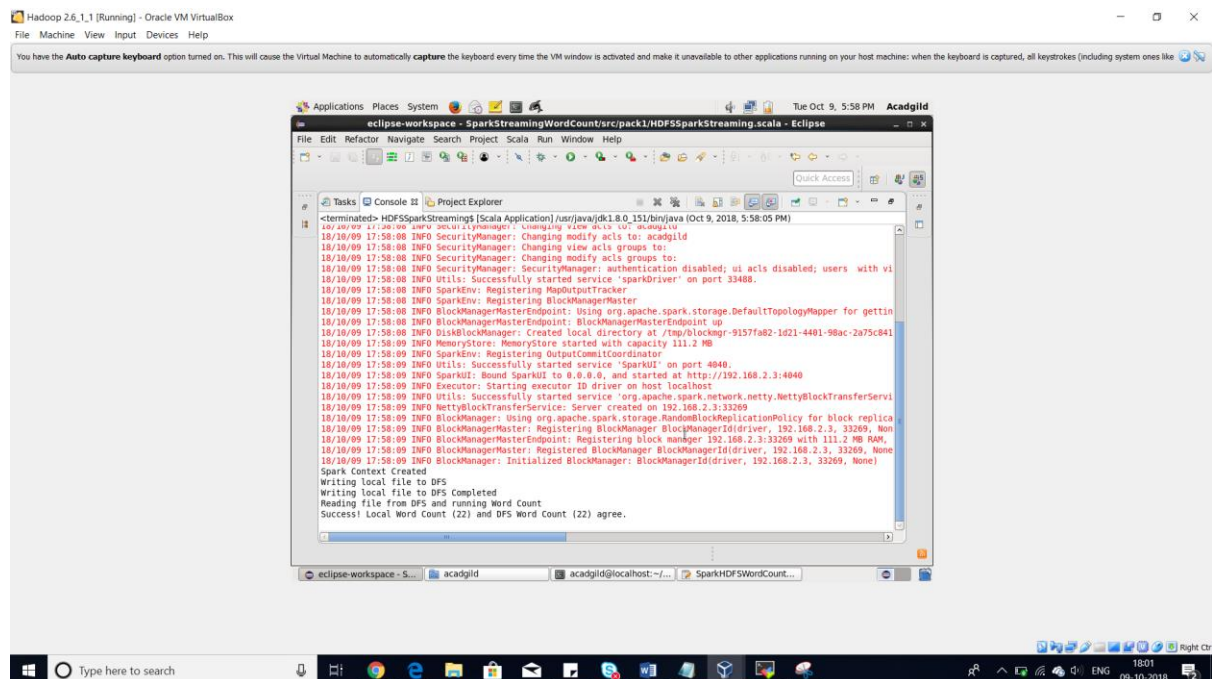


## 2. Then in the same Spark Application, write the code to put the same file on HDFS.



```
acacgild@localhost:~/Desktop/TestHadoop/SparkStreaming
File Edit View Search Terminal Help
Hi this is Big Data Class
acacgild@localhost SparkStreaming$ hadoop fs -ls /user/streaming
18/10/09 17:53:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 1 items
drwxr-xr-x - acacgild supergroup @ 2018-10-09 17:34 /user/streaming/dfs read write test
You have new mail in /var/spool/mail/acacgild
acacgild@localhost SparkStreaming$ hadoop fs -cat /user/streaming/dfs read write test
18/10/09 17:53:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
cat: /user/streaming/dfs read write test: Is a directory
acacgild@localhost SparkStreaming$ hadoop fs -ls /user/streaming
18/10/09 17:54:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 1 items
drwxr-xr-x - acacgild supergroup @ 2018-10-09 17:34 /user/streaming/dfs read write test
acacgild@localhost SparkStreaming$ hadoop fs -ls /user/streaming/dfs read write test
18/10/09 17:54:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 3 items
-rw-r--r-- 3 acacgild supergroup @ 2018-10-09 17:34 /user/streaming/dfs read write test/ SUCCESS
-rw-r--r-- 3 acacgild supergroup @ 2018-10-09 17:34 /user/streaming/dfs read write test/part-00000
-rw-r--r-- 3 acacgild supergroup @ 2018-10-09 17:34 /user/streaming/dfs read write test/part-00001
acacgild@localhost SparkStreaming$ hadoop fs -cat /user/streaming/dfs read write test
18/10/09 17:54:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
cat: /user/streaming/dfs read write test: Is a directory
acacgild@localhost SparkStreaming$ hadoop fs -cat /user/streaming/dfs read write test/part-00000
18/10/09 17:55:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Hi this is hadoop
Hi this is Hadoop
You have new mail in /var/spool/mail/acacgild
acacgild@localhost SparkStreaming$ hadoop fs -cat /user/streaming/dfs read write test/part-00001
18/10/09 17:55:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Hi thi is Big Data Class
acacgild@localhost SparkStreaming$
```

## 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, otherwise throw an error.



```
eclipse-workspace - SparkStreamingWordCount/src/pack1/HDFSSparkStreaming.scala - Eclipse
File Edit Refactor Navigate Search Project Scala Run Window Help
Tasks Console II Project Explorer
<terminated> HDFSSparkStreaming$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Oct 9, 2018, 5:58:05 PM)
18/10/09 17:58:00 INFO SecurityManager: Unloading view acl: us
18/10/09 17:58:00 INFO SecurityManager: Changing modify acls to: acacgild
18/10/09 17:58:00 INFO SecurityManager: Changing view acls groups to:
18/10/09 17:58:00 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with vi
18/10/09 17:58:00 INFO Utils: Successfully started service 'sparkDriver' on port 33488.
18/10/09 17:58:00 INFO SparkEnv: Registering MapOutputTracker
18/10/09 17:58:00 INFO SparkEnv: Registering BlockManagerMaster
18/10/09 17:58:00 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for gettin
18/10/09 17:58:00 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/10/09 17:58:00 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-9157fab2-1d21-4441-98ac-2a75c841
18/10/09 17:58:00 INFO MemoryStore: MemoryStore started with capacity 111.2 MB
18/10/09 17:58:00 INFO SparkEnv: Registering OutputCommitCoordinator
18/10/09 17:58:00 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.2.3:4040
18/10/09 17:58:00 INFO Executor: Starting executor 10 driver on host localhost
18/10/09 17:58:00 INFO NettyBlockTransferService: Server created on 192.168.2.3:33269
18/10/09 17:58:00 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replica
18/10/09 17:58:00 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.2.3, 33269, Non
18/10/09 17:58:00 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.2.3:33269 with 111.2 MB RAM,
18/10/09 17:58:00 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 192.168.2.3, 33269, Non
18/10/09 17:58:00 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 192.168.2.3, 33269, None)
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 (22) and DFS Word Count (22) agree.
```

