

Case-Study Day 26 (Spark Streaming)

Objective 1:

1. There are two parts this case study

First Part - You need 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 count 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.

Ans:

Note: Program files are properly documented for a detailed description of each instruction used within the program.

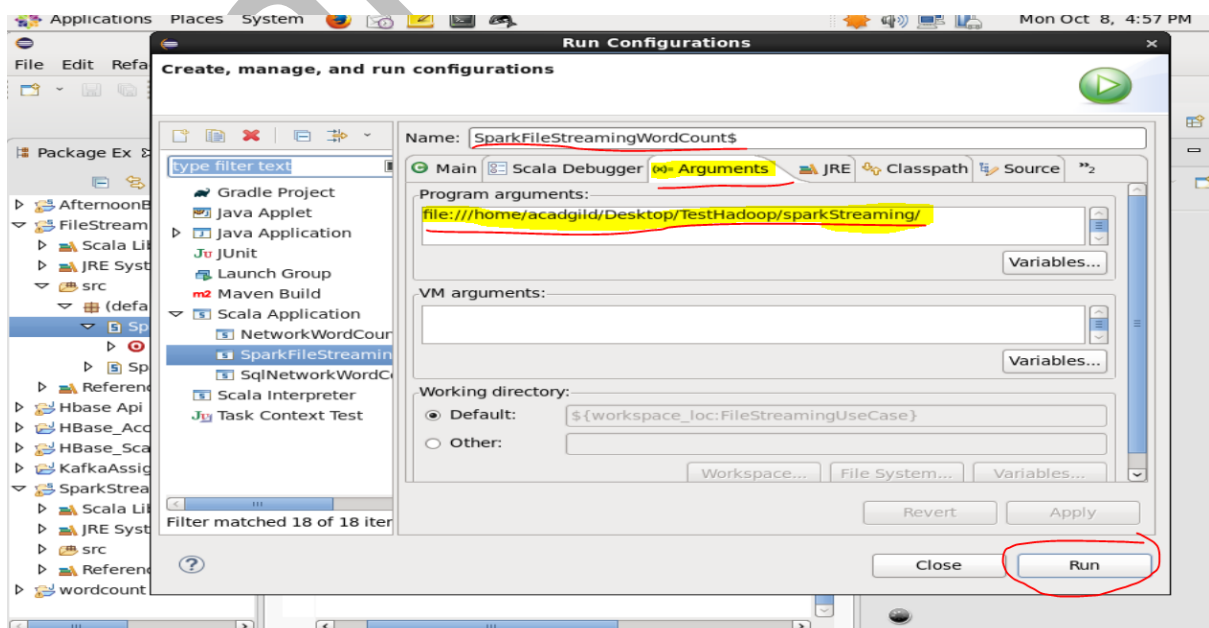
ScreenShot:

The Directory created for streaming file is:

`/home/acadgild/Desktop/TestHadoop/sparkStreaming/`

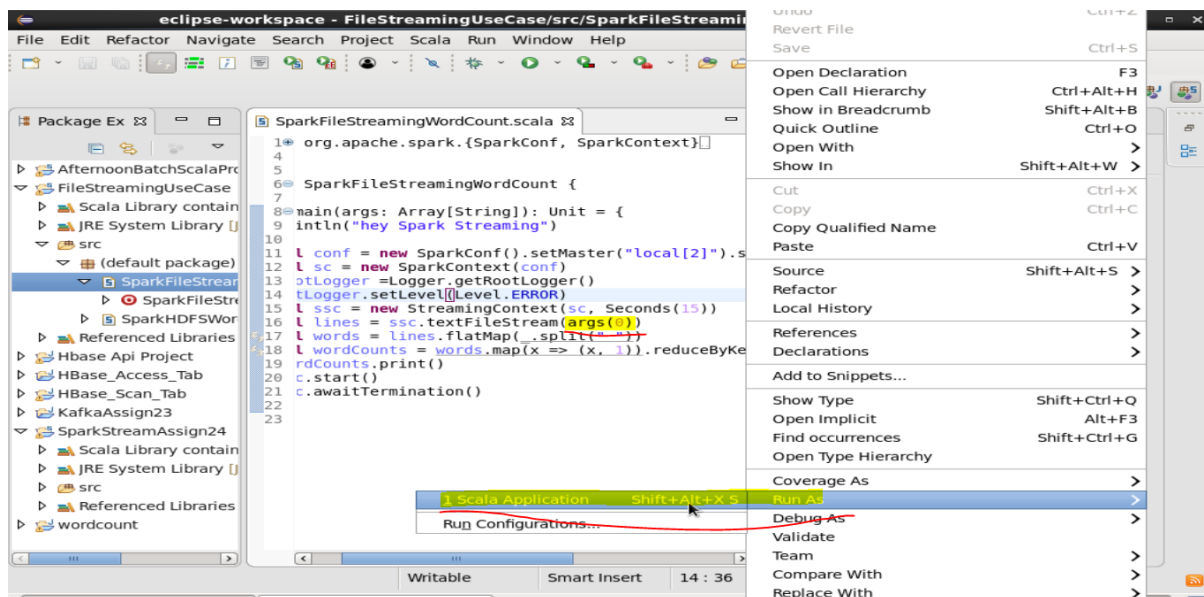
```
acadgild@localhost:~/Desktop/TestHadoop/sparkStreaming
File Edit View Search Terminal Help
[acadgild@localhost sparkStreaming]$ pwd
/home/acadgild/Desktop/TestHadoop/sparkStreaming
[acadgild@localhost sparkStreaming]$ ll
total 0
[acadgild@localhost sparkStreaming]$
```

Specifying the above created directory as input to program arguments.

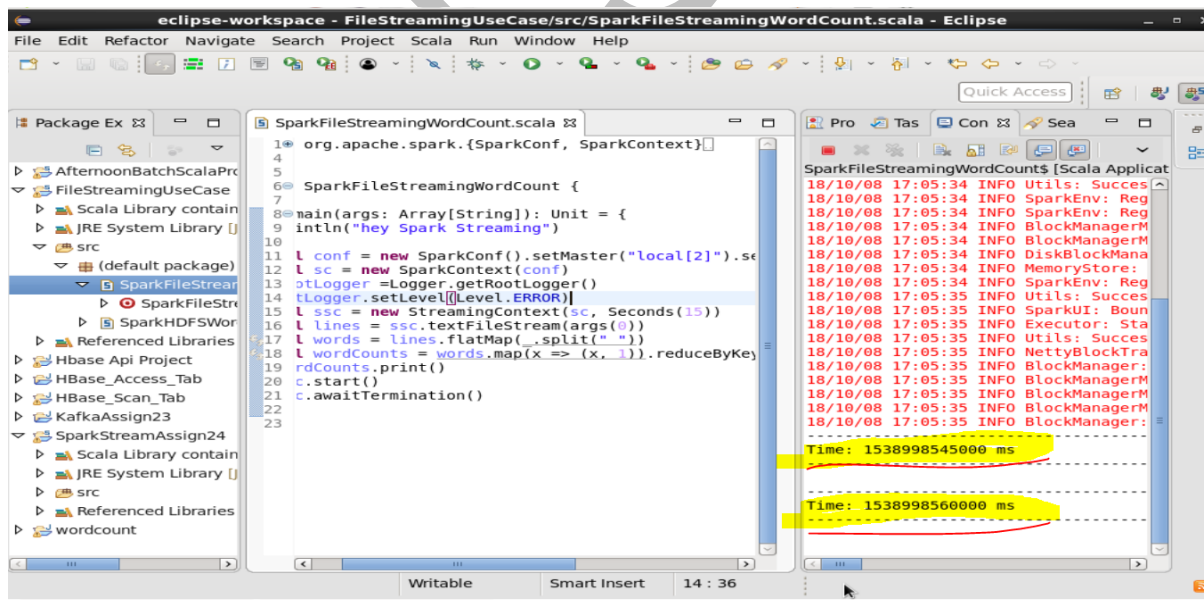


arg(0) specified within the program reads entire files that are getting added to the above directory & performs the word count operation on the fly.

Run the application as Scala Application.

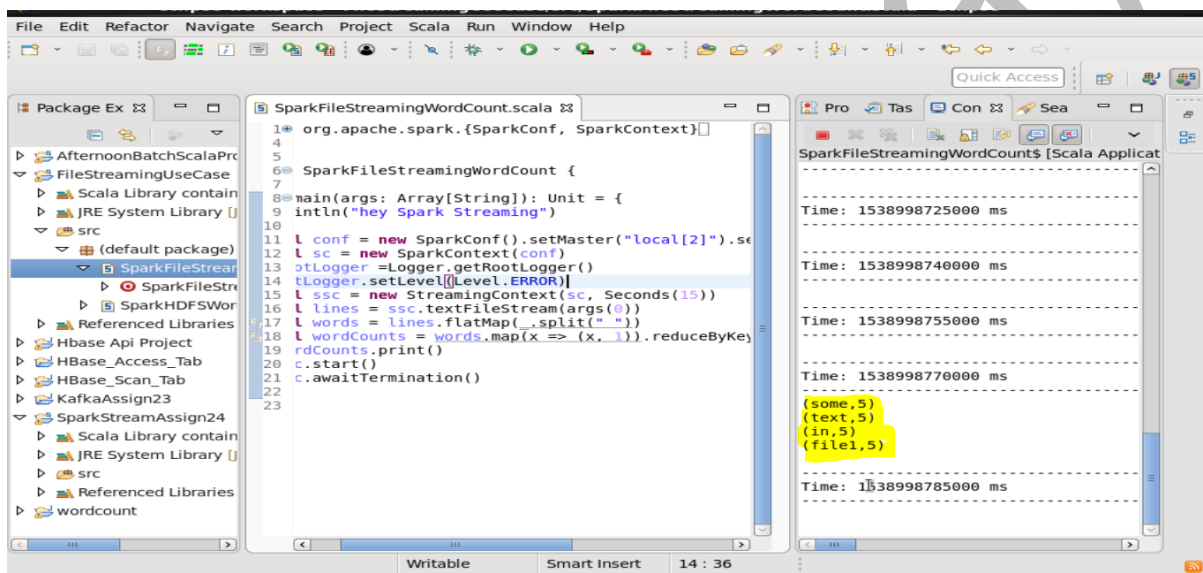


The application is streaming now. Let's create some files within input directory.



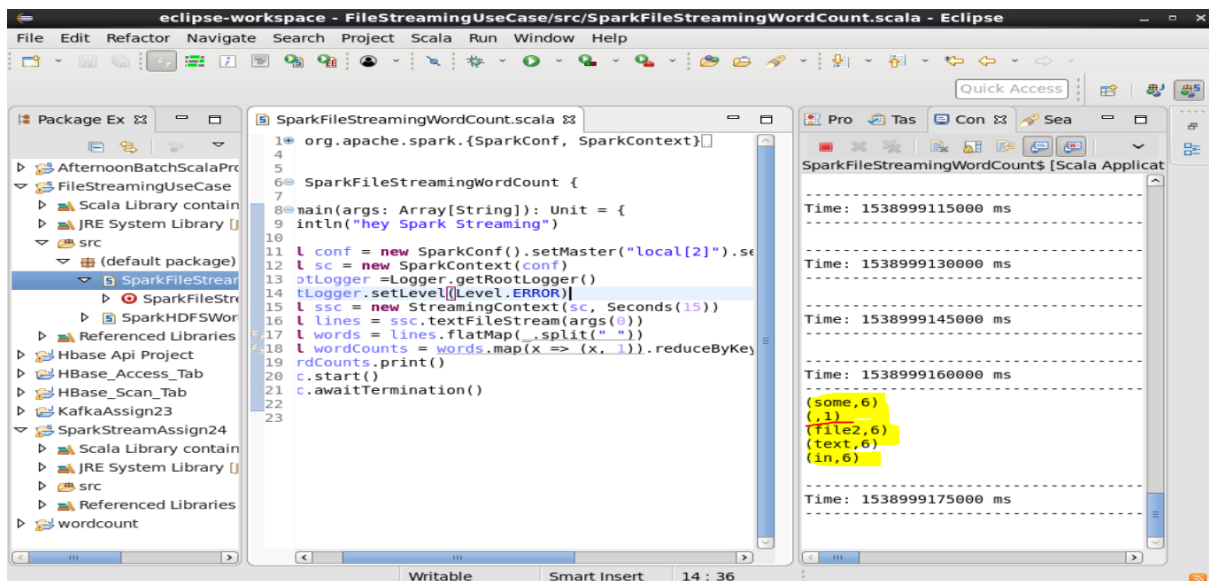
```
acadgild@localhost:
File Edit View Search Terminal Help
[acadgild@localhost sparkStreaming]$ pwd
/home/acadgild/Desktop/TestHadoop/sparkStreaming
[acadgild@localhost sparkStreaming]$ cat file1.txt
some text in file1
some text in file1
some text in file1
some text in file1
some text in file1
[acadgild@localhost sparkStreaming]$
```

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



Similarly, creating two other files whose words would be counted by spark streaming application on the fly. **(Counts blank line too)**

```
File Edit View Search Terminal Help
[acadgild@localhost sparkStreaming]$ ll
total 8
-rw-rw-r--. 1 acadgild acadgild 95 Oct 8 17:09 file1.txt
-rw-rw-r--. 1 acadgild acadgild 115 Oct 8 17:15 file2.txt
[acadgild@localhost sparkStreaming]$ cat file2.txt
some text in file2
some text in file2
some text in file2
some text in file2
some text in file2
some text in file2
[acadgild@localhost sparkStreaming]$ pwd
/home/acadgild/Desktop/TestHadoop/sparkStreaming
```



Dropping one third file with multiple blank lines.

```

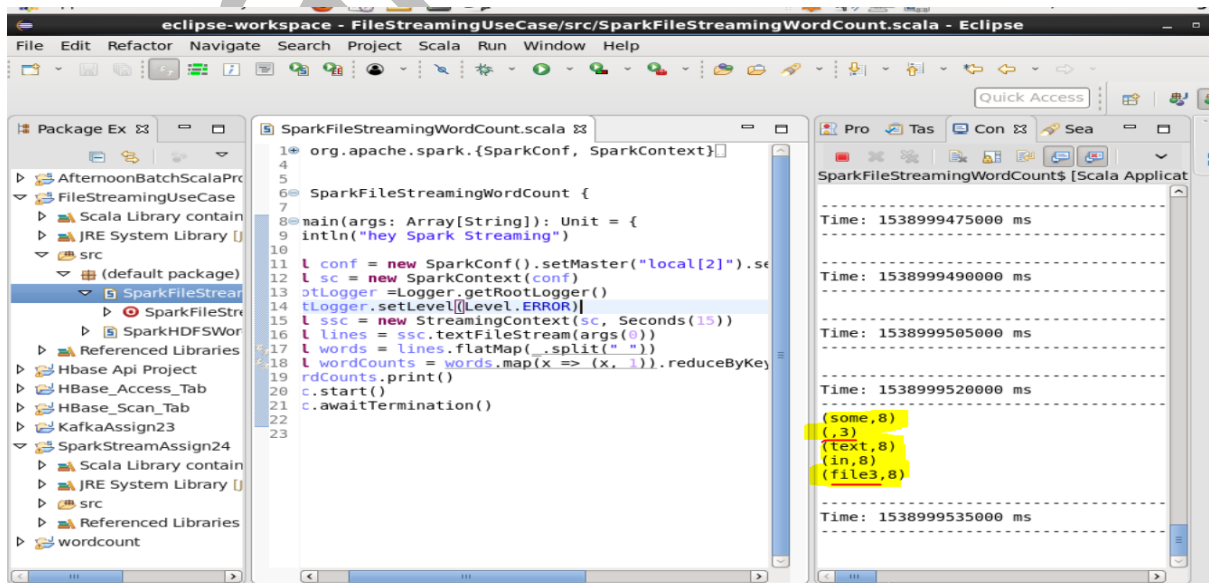
acadgild@localhost
File Edit View Search Terminal Help
[acadgild@localhost sparkStreaming]$ pwd
/home/acadgild/Desktop/TestHadoop/sparkStreaming
[acadgild@localhost sparkStreaming]$ vi file3.txt
[acadgild@localhost sparkStreaming]$ cat file3.txt
some text in file3

some text in file3
some text in file3
some text in file3

some text in file3
some text in file3
some text in file3
some text in file3

[acadgild@localhost sparkStreaming]$

```



Objective 2:

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

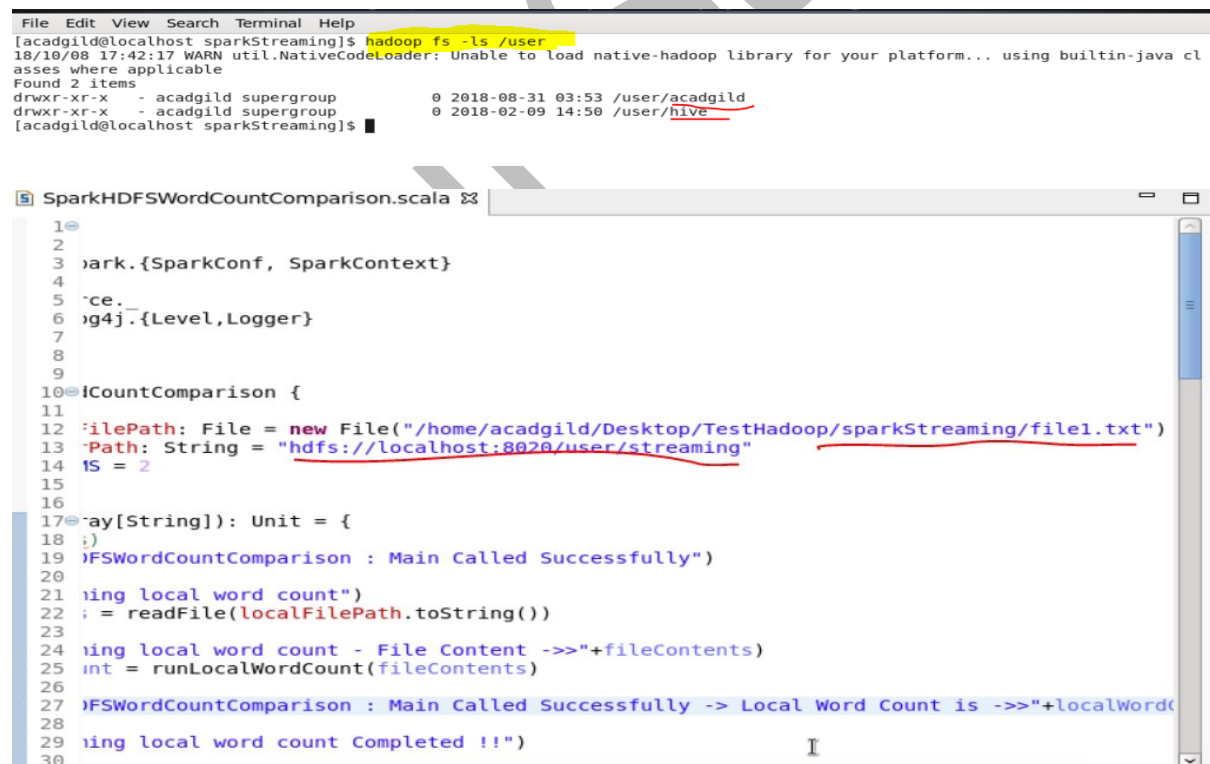
1. Pick up a file from the local directory and do the word count
2. Then in the same Spark Application, write the code to put the same file on HDFS.
3. Then in same Spark Application, do the word count of the file copied on HDFS in step 2
4. Lastly, compare the word count of step 1 and 2. Both should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

Ans:

Note: Program files are properly documented for a detailed description of each instruction used within the program.

ScreenShot:

HDFS does not contain streaming directory before the application is run

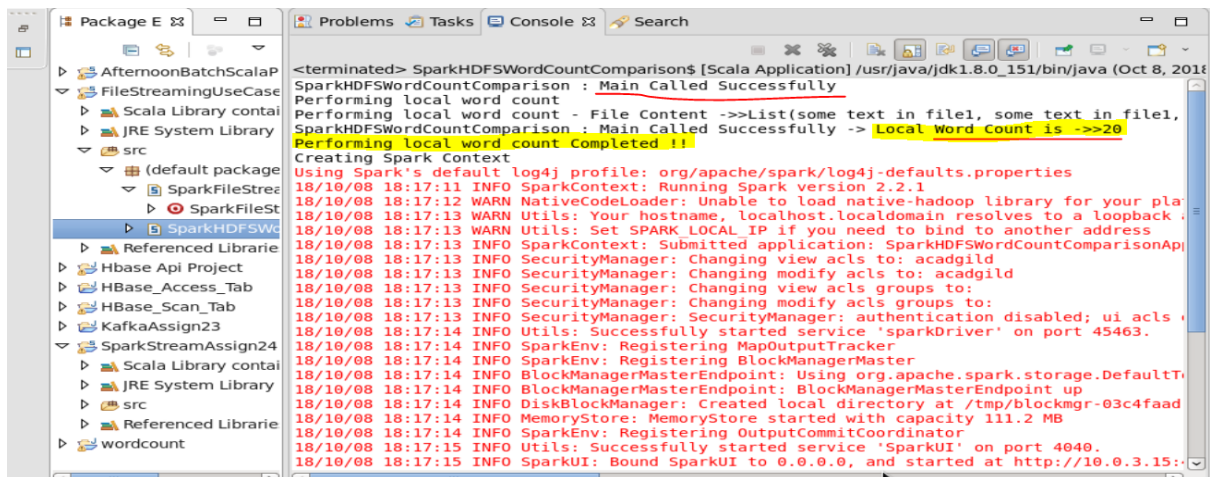


The screenshot displays two windows. The top window is a terminal with the command `hadoop fs -ls /user` executed, showing the file `file1.txt` in the `/user/hive` directory. The bottom window is an IDE showing the Scala code for `SparkHDFSWordCountComparison.scala`. The code defines a `WordCountComparison` object with a `main` method that reads a file from the local filesystem and compares its word count with the word count of the same file stored in HDFS. The HDFS path is `hdfs://localhost:8020/user/streaming`.

```
File Edit View Search Terminal Help
[acadgild@localhost sparkStreaming]$ hadoop fs -ls /user
18/10/08 17:42:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 2 items
drwxr-xr-x - acadgild supergroup          0 2018-08-31 03:53 /user/acadgild
drwxr-xr-x - acadgild supergroup          0 2018-02-09 14:50 /user/hive
[acadgild@localhost sparkStreaming]$
```

```
SparkHDFSWordCountComparison.scala
1
2
3 spark.{SparkConf, SparkContext}
4
5 `ce._
6 log4j.{Level, Logger}
7
8
9
10 WordCountComparison {
11
12   `filePath: File = new File("/home/acadgild/Desktop/TestHadoop/sparkStreaming/file1.txt")
13   `path: String = "hdfs://localhost:8020/user/streaming"
14   `is = 2
15
16
17   `ay[String]): Unit = {
18     ;
19     HDFSWordCountComparison : Main Called Successfully")
20
21     `ing local word count")
22     ; = readFile(localFilePath.toString())
23
24     `ing local word count - File Content ->"+fileContents)
25     `nt = runLocalWordCount(fileContents)
26
27     HDFSWordCountComparison : Main Called Successfully -> Local Word Count is ->"+localWordC
28
29     `ing local word count Completed !!")
30
```


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



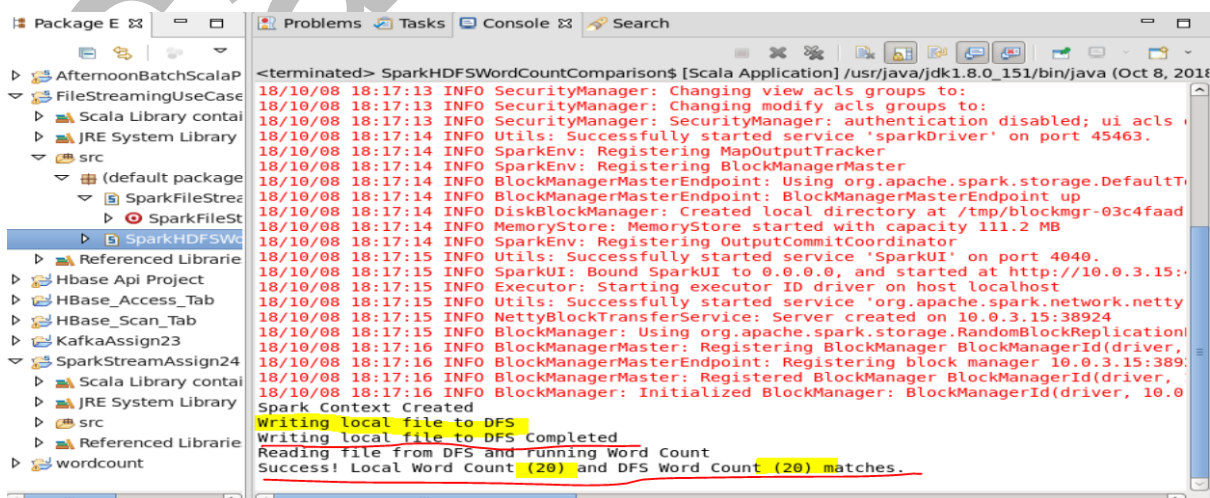
```
<terminated> SparkHDFSWordCountComparison$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Oct 8, 2018)
SparkHDFSWordCountComparison : Main Called Successfully
Performing local word count
Performing local word count - File Content ->List(some text in file1, some text in file1,
SparkHDFSWordCountComparison : Main Called Successfully -> Local Word Count is ->20
Performing local word count Completed !!
Creating Spark Context
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/10/08 18:17:11 INFO SparkContext: Running Spark version 2.2.1
18/10/08 18:17:12 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java class
18/10/08 18:17:13 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback IP address: 127.0.0.1; address: 127.0.0.1
18/10/08 18:17:13 INFO SparkContext: Submitted application: SparkHDFSWordCountComparisonApp
18/10/08 18:17:13 INFO SecurityManager: Changing view acls to: acadgild
18/10/08 18:17:13 INFO SecurityManager: Changing modify acls to: acadgild
18/10/08 18:17:13 INFO SecurityManager: Changing view acls groups to:
18/10/08 18:17:13 INFO SecurityManager: Changing modify acls groups to:
18/10/08 18:17:13 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled;
18/10/08 18:17:14 INFO SecurityManager: Successfully started service 'sparkDriver' on port 45463.
18/10/08 18:17:14 INFO SparkEnv: Registering MapOutputTracker
18/10/08 18:17:14 INFO SparkEnv: Registering BlockManagerMaster
18/10/08 18:17:14 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopology
18/10/08 18:17:14 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/10/08 18:17:14 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-03c4faad
18/10/08 18:17:14 INFO MemoryStore: MemoryStore started with capacity 111.2 MB
18/10/08 18:17:14 INFO SparkEnv: Registering OutputCommitCoordinator
18/10/08 18:17:15 INFO SecurityManager: Successfully started service 'SparkUI' on port 4040.
18/10/08 18:17:15 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://0.0.3.15:4040
```

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

```
[acadgild@localhost sparkStreaming]$ hadoop fs -ls /user
18/10/08 18:20:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java class
asses where applicable
Found 3 items
drwxr-xr-x - acadgild supergroup 0 2018-08-31 03:53 /user/acadgild
drwxr-xr-x - acadgild supergroup 0 2018-02-09 14:50 /user/hive
drwxr-xr-x - acadgild supergroup 0 2018-10-08 18:17 /user/streaming
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost sparkStreaming]$ hadoop fs -ls /user/streaming
18/10/08 18:20:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java class
asses where applicable
Found 1 items
drwxr-xr-x - acadgild supergroup 0 2018-10-08 18:17 /user/streaming/dfs_read_write_test
[acadgild@localhost sparkStreaming]$ hadoop fs -ls /user/streaming/dfs_read_write_test
18/10/08 18:21:04 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java class
asses where applicable
Found 3 items
-rw-r--r-- 3 acadgild supergroup 0 2018-10-08 18:17 /user/streaming/dfs_read_write_test/_SUCCESS
-rw-r--r-- 3 acadgild supergroup 38 2018-10-08 18:17 /user/streaming/dfs_read_write_test/part-00000
-rw-r--r-- 3 acadgild supergroup 57 2018-10-08 18:17 /user/streaming/dfs_read_write_test/part-00001
[acadgild@localhost sparkStreaming]$ hadoop fs -cat /user/streaming/dfs_read_write_test/part-00000
18/10/08 18:21:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java class
asses where applicable
some text in file1
some text in file1
[acadgild@localhost sparkStreaming]$ hadoop fs -cat /user/streaming/dfs_read_write_test/part-00001
18/10/08 18:21:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java class
asses where applicable
some text in file1
some text in file1
some text in file1
[acadgild@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, other throw an error



```
<terminated> SparkHDFSWordCountComparison$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Oct 8, 2018)
18/10/08 18:17:13 INFO SecurityManager: Changing view acls groups to:
18/10/08 18:17:13 INFO SecurityManager: Changing modify acls groups to:
18/10/08 18:17:13 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled;
18/10/08 18:17:14 INFO SecurityManager: Successfully started service 'sparkDriver' on port 45463.
18/10/08 18:17:14 INFO SparkEnv: Registering MapOutputTracker
18/10/08 18:17:14 INFO SparkEnv: Registering BlockManagerMaster
18/10/08 18:17:14 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopology
18/10/08 18:17:14 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/10/08 18:17:14 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-03c4faad
18/10/08 18:17:14 INFO MemoryStore: MemoryStore started with capacity 111.2 MB
18/10/08 18:17:14 INFO SparkEnv: Registering OutputCommitCoordinator
18/10/08 18:17:15 INFO SecurityManager: Successfully started service 'SparkUI' on port 4040.
18/10/08 18:17:15 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://0.0.3.15:4040
18/10/08 18:17:15 INFO Executor: Starting executor ID driver on host localhost
18/10/08 18:17:15 INFO NettyBlockTransferService: Server created on 10.0.3.15:38924
18/10/08 18:17:15 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplication
18/10/08 18:17:16 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.0.3.15:38924)
18/10/08 18:17:16 INFO BlockManagerMasterEndpoint: Registering block manager 10.0.3.15:38924
18/10/08 18:17:16 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.0.3.15:38924)
18/10/08 18:17:16 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.0.3.15:38924)
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 (20) and DFS Word Count (20) matches.
```

End
