LAB - 07

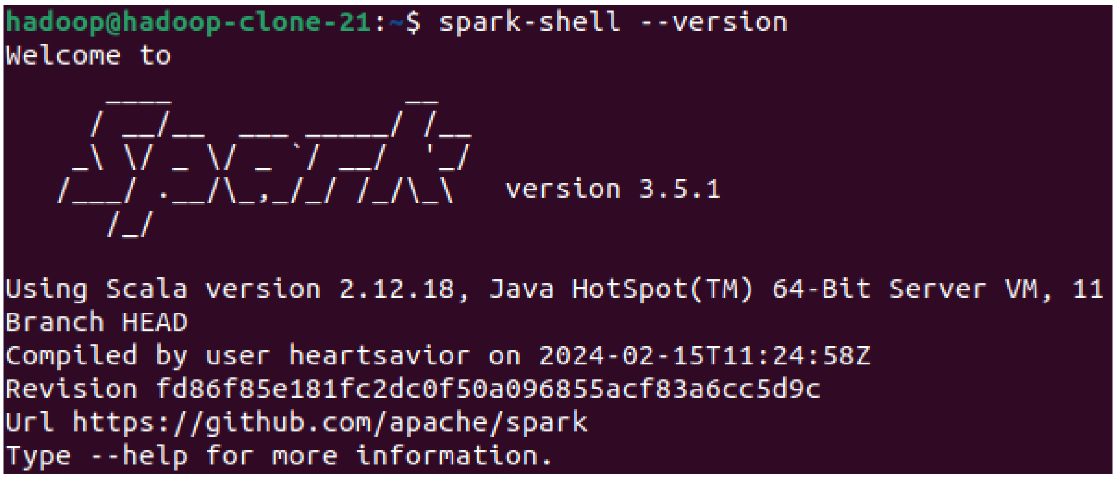
Aim : Learn and apply the architecture of Apache Spark for data analytics. Solve Word Count program requirement using Apache Spark.

Description:
Spark is a robust and evolving engine for data analytics. Spark works with data using RDDs (Resilient Distributed Dataset) and data frames. The operations are immutable. The api allowed to process the data uses an approach of lazy processing. This allows spark to have various workflows checked out and optimize the processing. It uses graphs to represent the dependencies and operations, which are also utilized while optimizing.

Methodology:
There are two ways to run a program in apache spark
1. Using Spark REPL
2. By submitting a jar file with spark-submit

Spark shell:
Spark's shell provides a simple way to learn the API, as well as a powerful tool to analyze data interactively. It is available in either Scala (which runs on the Java VM and is thus a good way to use existing Java libraries) or Python.

● **Starting spark shell**
We can start a spark-shell by using spark-shell <deployment mode>



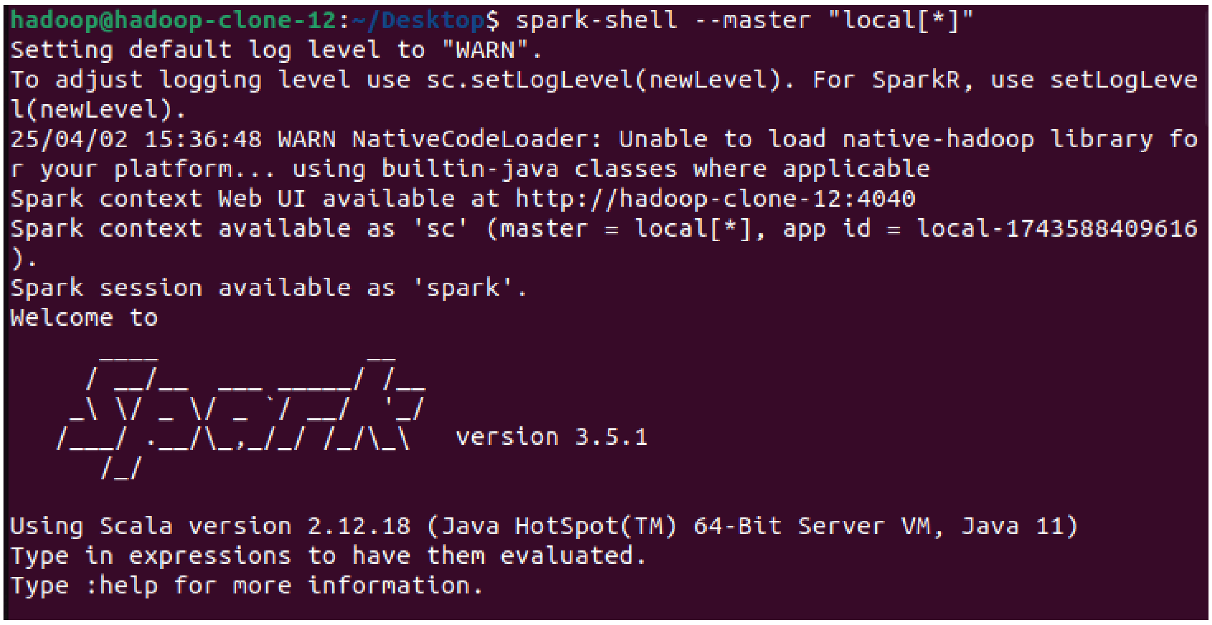
```
hadoop@hadoop-clone-21:~$ spark-shell --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18, Java HotSpot(TM) 64-Bit Server VM, 11
Branch HEAD
Compiled by user heartsavior on 2024-02-15T11:24:58Z
Revision fd86f85e181fc2dc0f50a096855acf83a6cc5d9c
Url https://github.com/apache/spark
Type --help for more information.
```

● **Spark deployment mode**
I. Local mode

II. Standalone mode
III. Cluster mode(YARN)

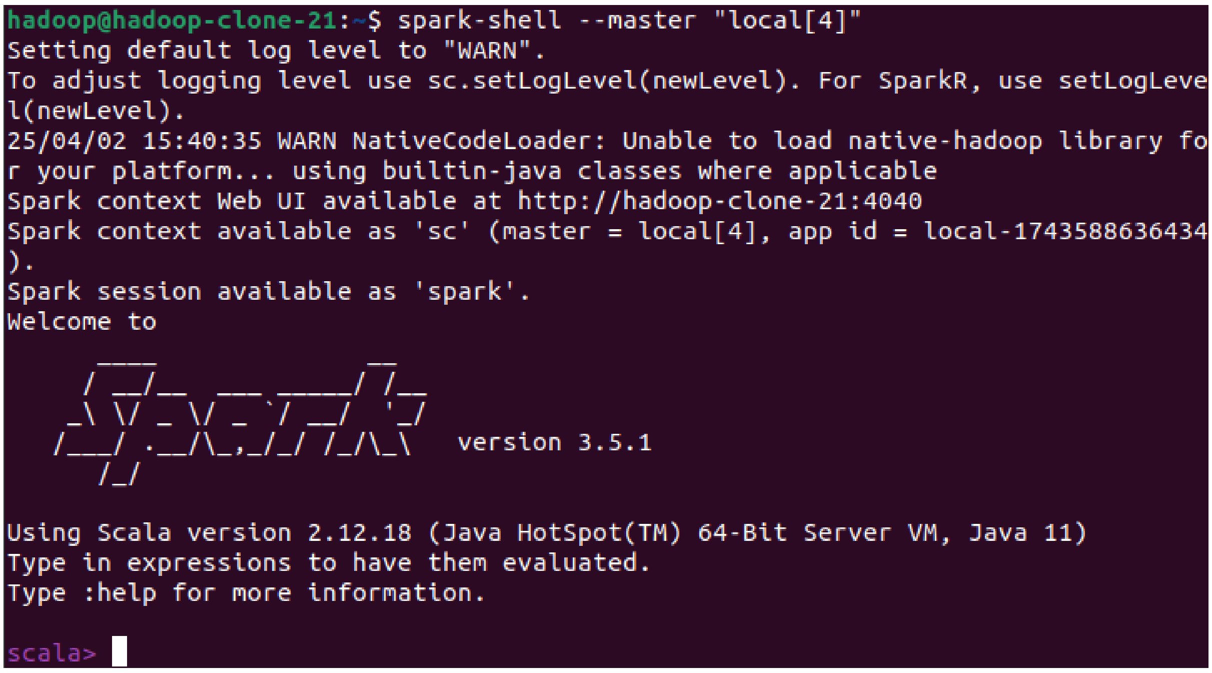
❖ **Working with Spark in Local mode**

A. Running in local mode with maximum core as possible
spark-shell - -master “local[*]”



```
hadoop@hadoop-clone-12:~/Desktop$ spark-shell --master "local[*]"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
25/04/02 15:36:48 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Spark context Web UI available at http://hadoop-clone-12:4040
Spark context available as 'sc' (master = local[*], app id = local-1743588409616
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 11)
Type in expressions to have them evaluated.
Type :help for more information.
```

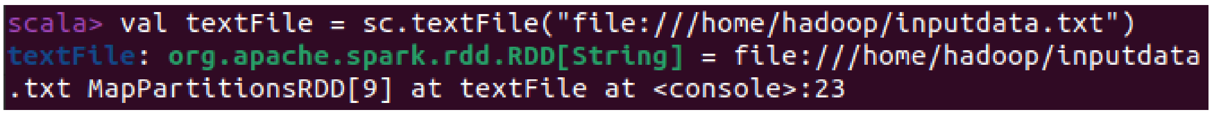
B. Running in local mode with 4 cores
spark-shell --master "local[4]"-



```
hadoop@hadoop-clone-21:~$ spark-shell --master "local[4]"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
25/04/02 15:40:35 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Spark context Web UI available at http://hadoop-clone-21:4040
Spark context available as 'sc' (master = local[4], app id = local-1743588636434
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 11)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

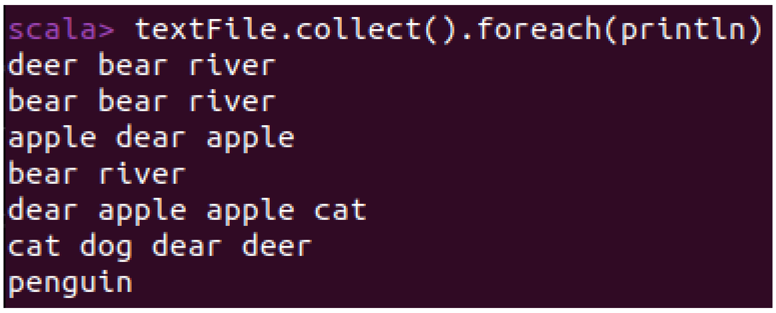
❖ Wordcount program Using Spark shell REPL:

1. Create an RDD for inputfile



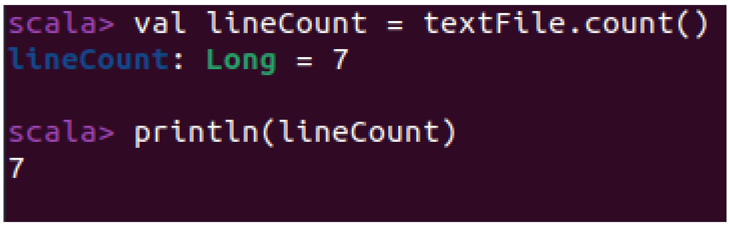
```
scala> val textFile = sc.textFile("file:///home/hadoop/inputdata.txt")
textFile: org.apache.spark.rdd.RDD[String] = file:///home/hadoop/inputdata
.txt MapPartitionsRDD[9] at textFile at <console>:23
```

2. Display the data of the input file(optional)



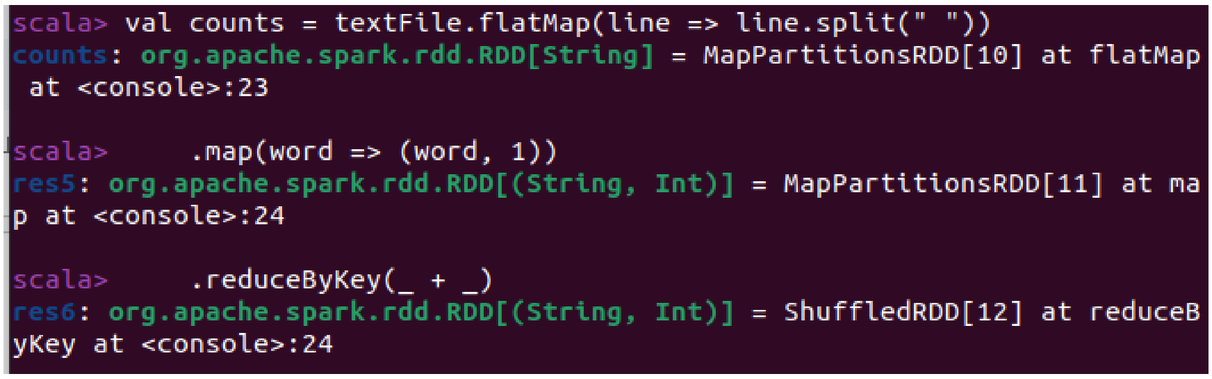
```
scala> textFile.collect().foreach(println)
deer bear river
bear bear river
apple dear apple
bear river
dear apple apple cat
cat dog dear deer
penguin
```

Count number of lines



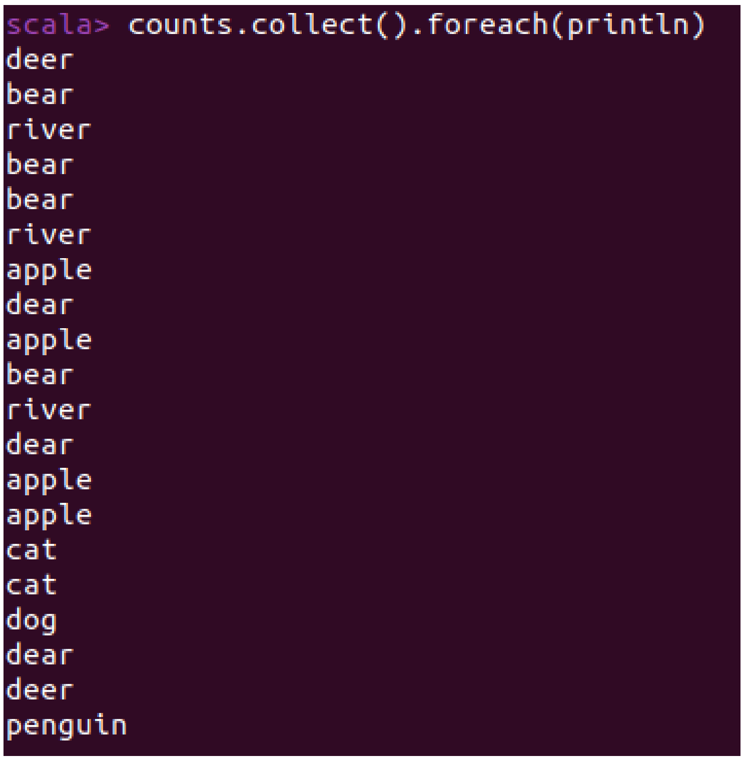
```
scala> val lineCount = textFile.count()
lineCount: Long = 7

scala> println(lineCount)
7
```

Perform word count



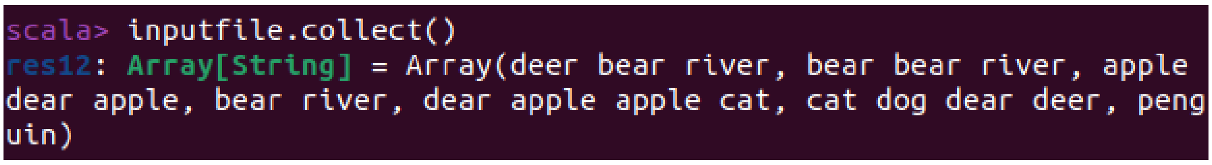
```
scala> val counts = textFile.flatMap(line => line.split(" "))
counts: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[10] at flatMap
 at <console>:23

scala>     .map(word => (word, 1))
res5: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[11] at ma
p at <console>:24

scala>     .reduceByKey(_ + _)
res6: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[12] at reduceB
yKey at <console>:24
```

Display word count



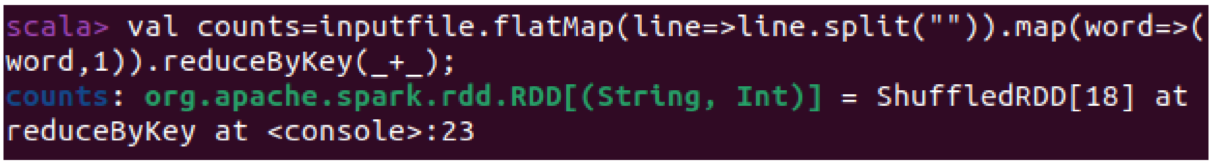
```
scala> counts.collect().foreach(println)
deer
bear
river
bear
bear
river
apple
dear
apple
bear
river
dear
apple
apple
cat
cat
dog
dear
deer
penguin
```

Collect the output



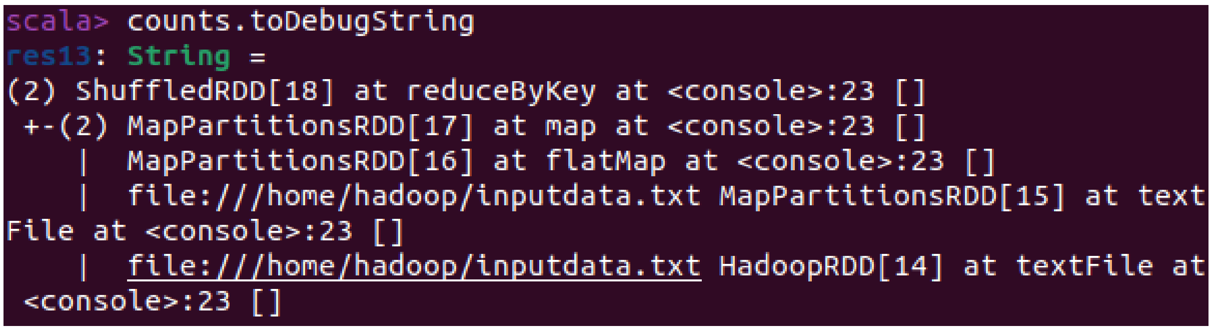
```
scala> inputfile.collect()
res12: Array[String] = Array(deer bear river, bear bear river, apple
dear apple, bear river, dear apple apple cat, cat dog dear deer, peng
uin)
```

3. Create a mapper and reducer



```
scala> val counts=inputfile.flatMap(line=>line.split("")).map(word=>(
word,1)).reduceByKey(_+_);
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[18] at
reduceByKey at <console>:23
```

● **RDD.toDebugString()**



```
scala> counts.toDebugString
res13: String =
(2) ShuffledRDD[18] at reduceByKey at <console>:23 []
 +-(2) MapPartitionsRDD[17] at map at <console>:23 []
    |  MapPartitionsRDD[16] at flatMap at <console>:23 []
    |  file:///home/hadoop/inputdata.txt MapPartitionsRDD[15] at text
File at <console>:23 []
    |  file:///home/hadoop/inputdata.txt HadoopRDD[14] at textFile at
 <console>:23 []
```

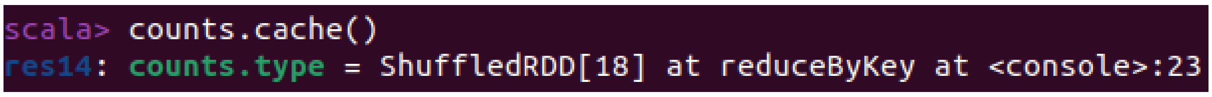
● **What is RDD Persistence and caching**

Spark RDD persistence is an optimization technique in which saves the result of RDD evaluation. Using this we save the intermediate result so that we can use it further if required. It reduces the computation overhead.

cache() and persist()

RDD.cache()

Persist this RDD with the default storage level (MEMORY_ONLY).

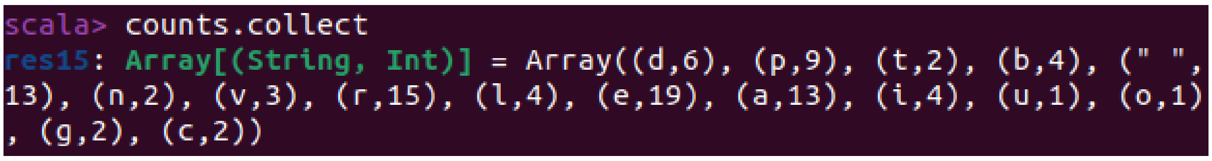


```
scala> counts.cache()
res14: counts.type = ShuffledRDD[18] at reduceByKey at <console>:23
```

4. Action - perform actions on created RDD

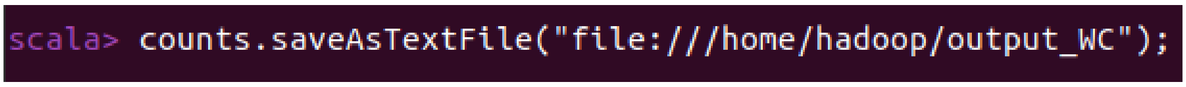
● collect -

Counts.collect will collect the data and display it on console



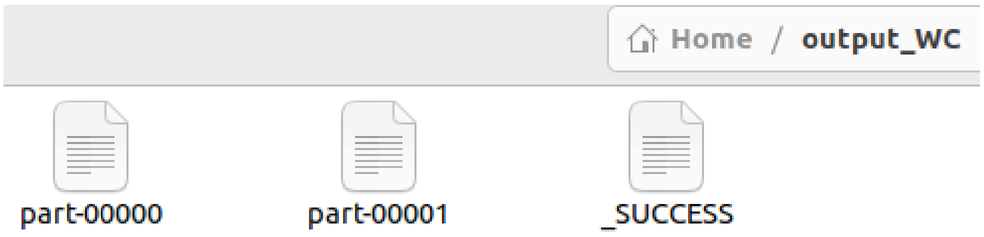
```
scala> counts.collect
res15: Array[(String, Int)] = Array((d,6), (p,9), (t,2), (b,4), (" ",
13), (n,2), (v,3), (r,15), (l,4), (e,19), (a,13), (i,4), (u,1), (o,1)
, (g,2), (c,2))
```

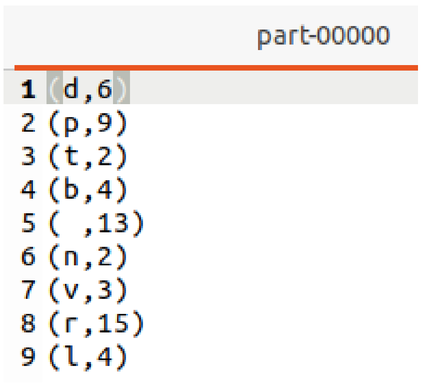
● If you want to save your data as output file we can use the following command



```
scala> counts.saveAsTextFile("file:///home/hadoop/output_WC");
```

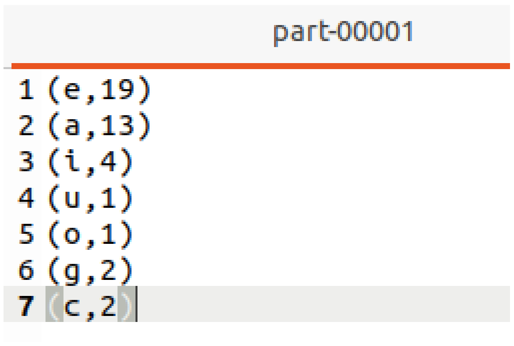
This is created in /home/hadoop





part-00000

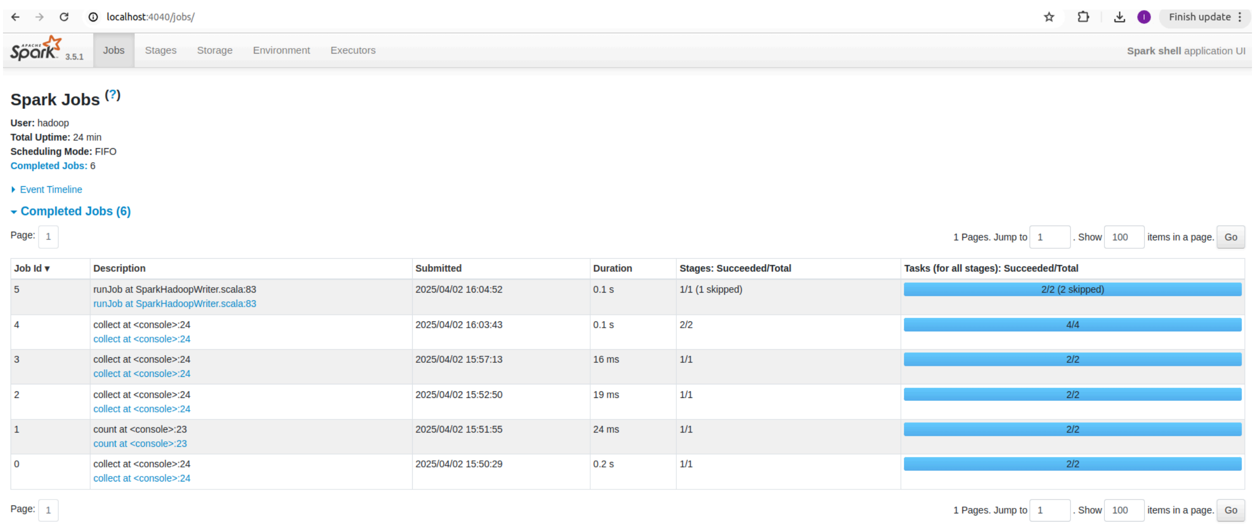
```
(d,6)
(p,9)
(t,2)
(b,4)
( ,13)
(n,2)
(v,3)
(r,15)
(l,4)
```



part-00001

```
(e,19)
(a,13)
(i,4)
(u,1)
(o,1)
(g,2)
(c,2)
```

Observing DAG visualization



Spark Jobs (?)

User: hadoop
Total Uptime: 24 min
Scheduling Mode: FIFO
Completed Jobs: 6

Event Timeline

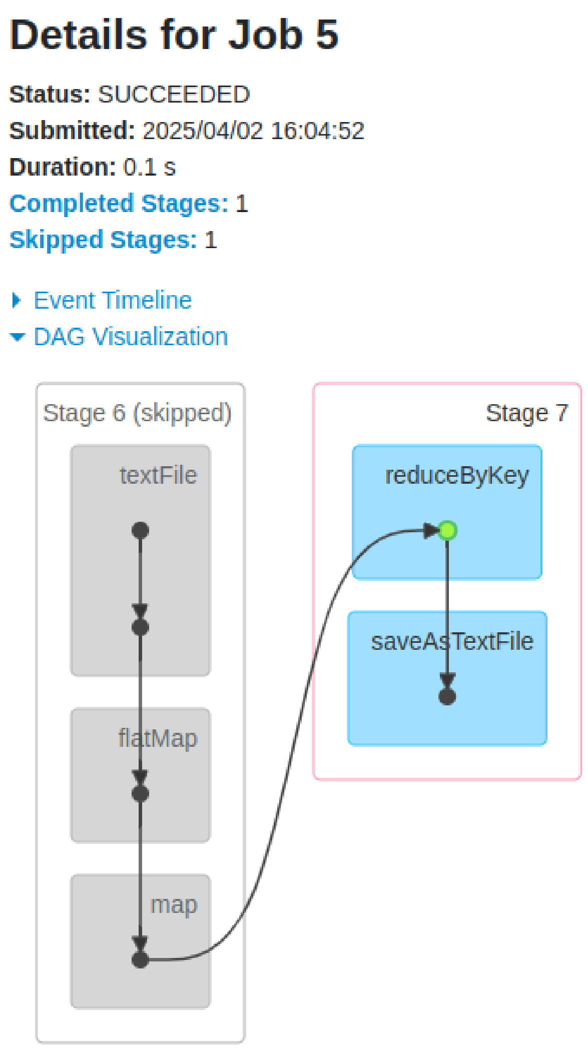
Completed Jobs (6)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83	2025/04/02 16:04:52	0.1 s	1/1 (1 skipped)	2/2 (2 skipped)
4	collect at <console>:24 collect at <console>:24	2025/04/02 16:03:43	0.1 s	2/2	4/4
3	collect at <console>:24 collect at <console>:24	2025/04/02 15:57:13	16 ms	1/1	2/2
2	collect at <console>:24 collect at <console>:24	2025/04/02 15:52:50	19 ms	1/1	2/2
1	count at <console>:23 count at <console>:23	2025/04/02 15:51:55	24 ms	1/1	2/2
0	collect at <console>:24 collect at <console>:24	2025/04/02 15:50:29	0.2 s	1/1	2/2

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Go to timeline of events and explore DAG visualization for the submitted job:

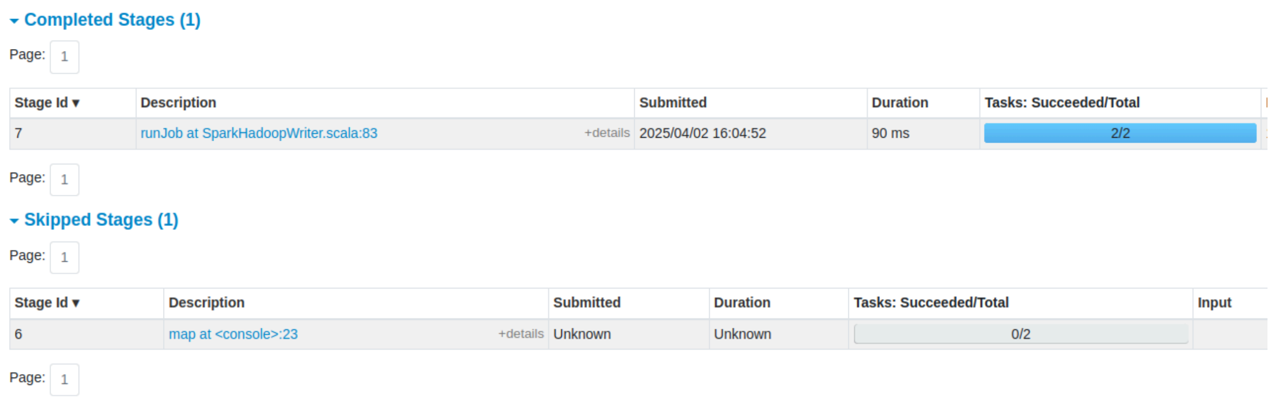


Details for Job 5

Status: SUCCEEDED
Submitted: 2025/04/02 16:04:52
Duration: 0.1 s
Completed Stages: 1
Skipped Stages: 1

▸ Event Timeline
▾ DAG Visualization

We can also observe the status of the jobs as below:



▾ Completed Stages (1)

Page: 1

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total
7	runJob at SparkHadoopWriter.scala:83 +details	2025/04/02 16:04:52	90 ms	2/2

Page: 1

▾ Skipped Stages (1)

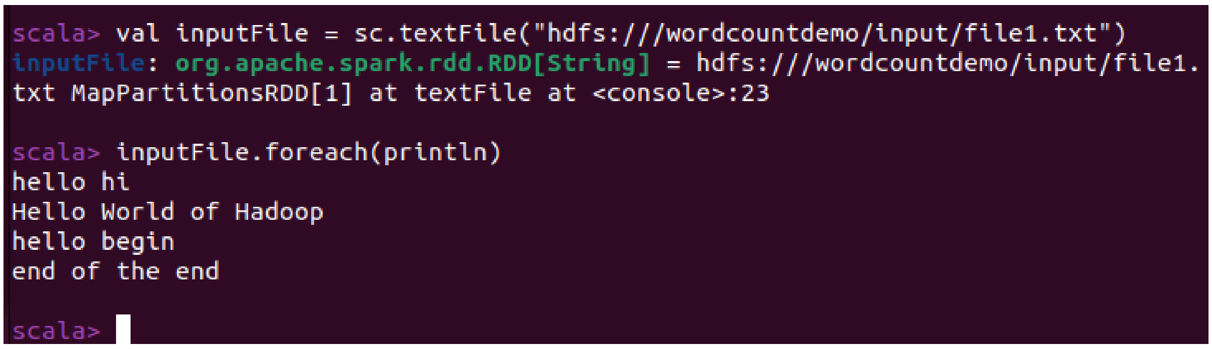
Page: 1

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input
6	map at <console>:23 +details	Unknown	Unknown	0/2	

Page: 1

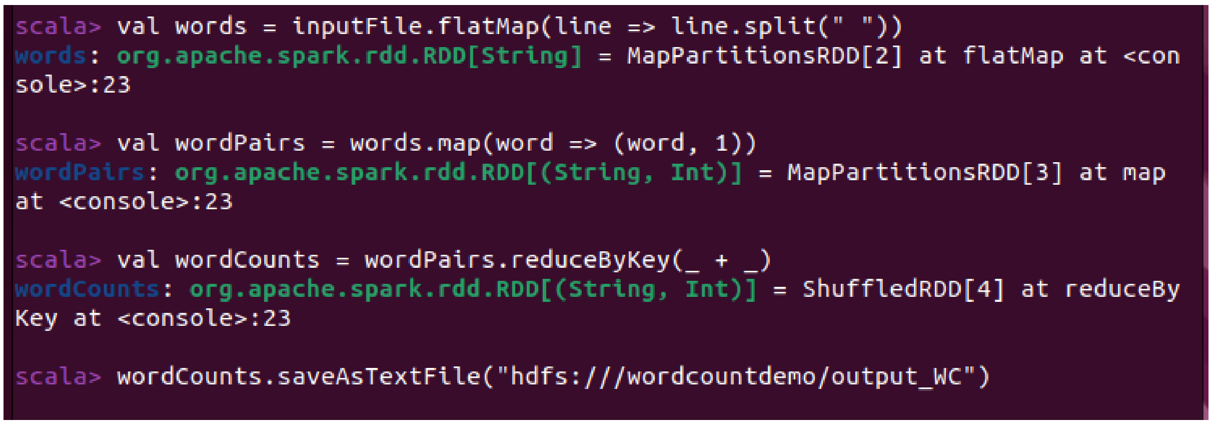
Wordcount Program by taking input from HDFS:

1. Create an input file in HDFS and upload it in spark as shown below



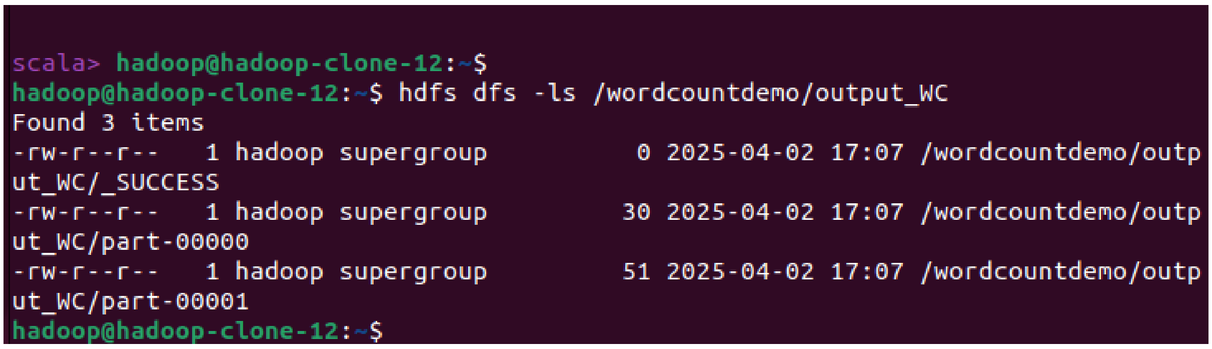
```
scala> val inputFile = sc.textFile("hdfs:///wordcountdemo/input/file1.txt")
inputFile: org.apache.spark.rdd.RDD[String] = hdfs:///wordcountdemo/input/file1.
txt MapPartitionsRDD[1] at textFile at <console>:23

scala> inputFile.foreach(println)
hello hi
Hello World of Hadoop
hello begin
end of the end

scala> 
```

2. Write the Map reduce steps



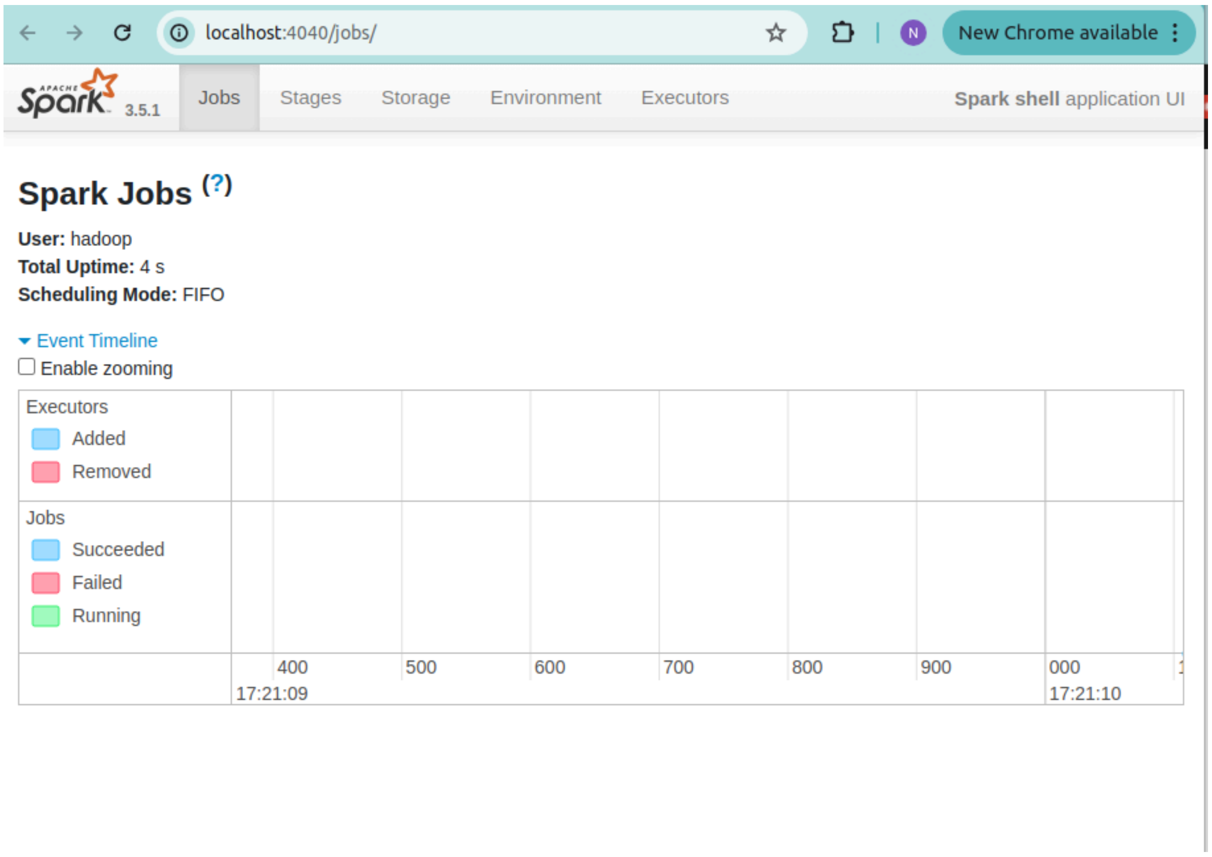
```
scala> val words = inputFile.flatMap(line => line.split(" "))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <con
sole>:23

scala> val wordPairs = words.map(word => (word, 1))
wordPairs: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map
at <console>:23

scala> val wordCounts = wordPairs.reduceByKey(_ + _)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceBy
Key at <console>:23

scala> wordCounts.saveAsTextFile("hdfs:///wordcountdemo/output_WC")
```

Verifying that all the files exist



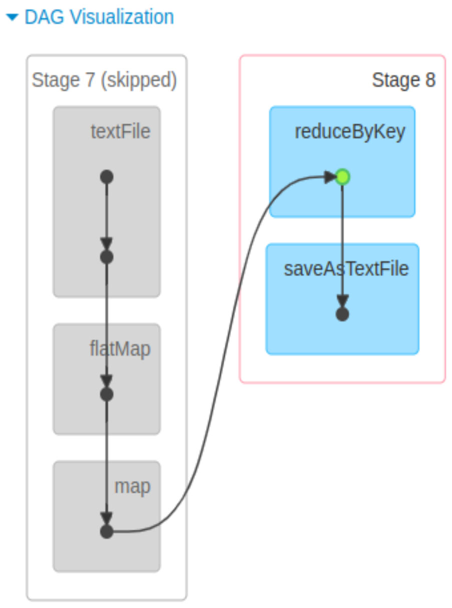
```
scala> hadoop@hadoop-clone-12:~$
hadoop@hadoop-clone-12:~$ hdfs dfs -ls /wordcountdemo/output_WC
Found 3 items
-rw-r--r--   1 hadoop supergroup          0 2025-04-02 17:07 /wordcountdemo/outp
ut_WC/_SUCCESS
-rw-r--r--   1 hadoop supergroup         30 2025-04-02 17:07 /wordcountdemo/outp
ut_WC/part-00000
-rw-r--r--   1 hadoop supergroup         51 2025-04-02 17:07 /wordcountdemo/outp
ut_WC/part-00001
hadoop@hadoop-clone-12:~$
```

3. Observing the DAG visualization at http://localhost:4040/jobs/



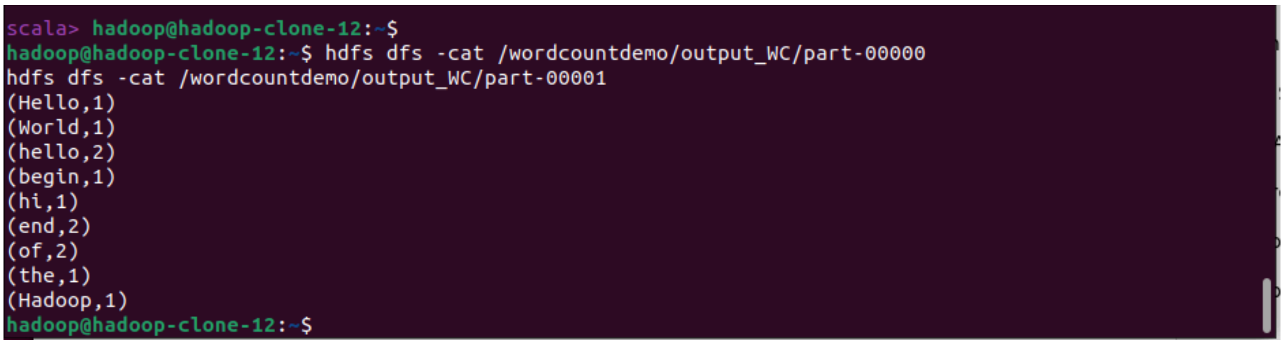
Spark Jobs (?)

User: hadoop
Total Uptime: 4 s
Scheduling Mode: FIFO



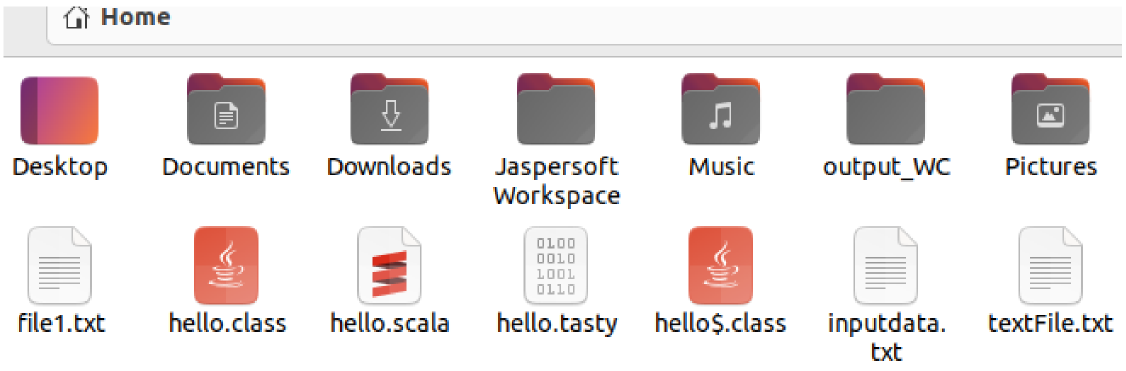
Skipped stage:
It means that data has been fetched from cache and there was no need to re-execute given stage.
It is consistent with your DAG which shows that the next stage requires shuffling (reduceByKey). Whenever there is shuffling involved Spark automatically caches generated data.

Output on hdfs looks like:



```
scala> hadoop@hadoop-clone-12:~$
hadoop@hadoop-clone-12:~$ hdfs dfs -cat /wordcountdemo/output_WC/part-00000
hdfs dfs -cat /wordcountdemo/output_WC/part-00001
(Hello,1)
(World,1)
(hello,2)
(begin,1)
(hi,1)
(end,2)
(of,2)
(the,1)
(Hadoop,1)
hadoop@hadoop-clone-12:~$
```

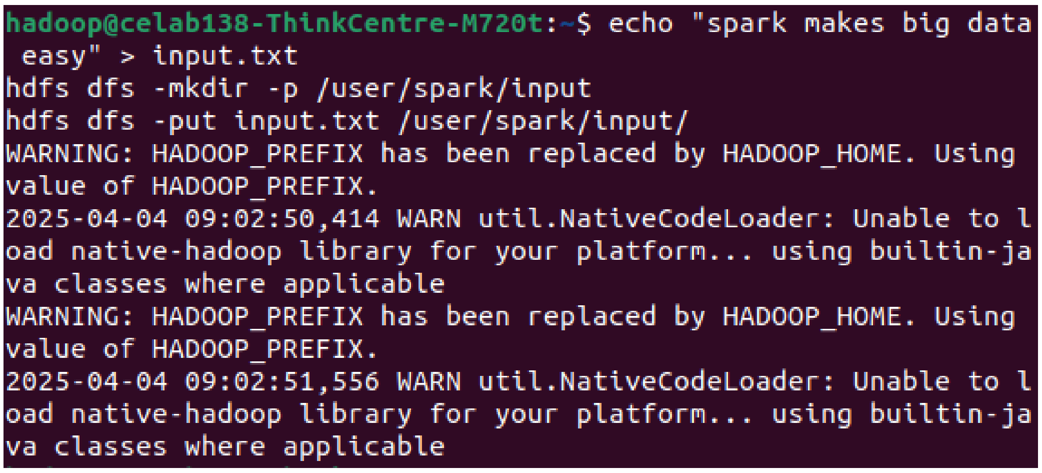
Files created locally:



EXERCISE :-

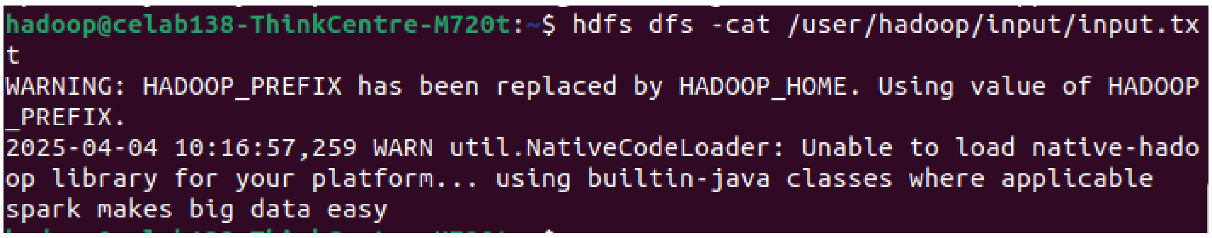
1. Develop simple wordcount application with apache spark(using REPL and HDFS)

Creating input in HDFS:



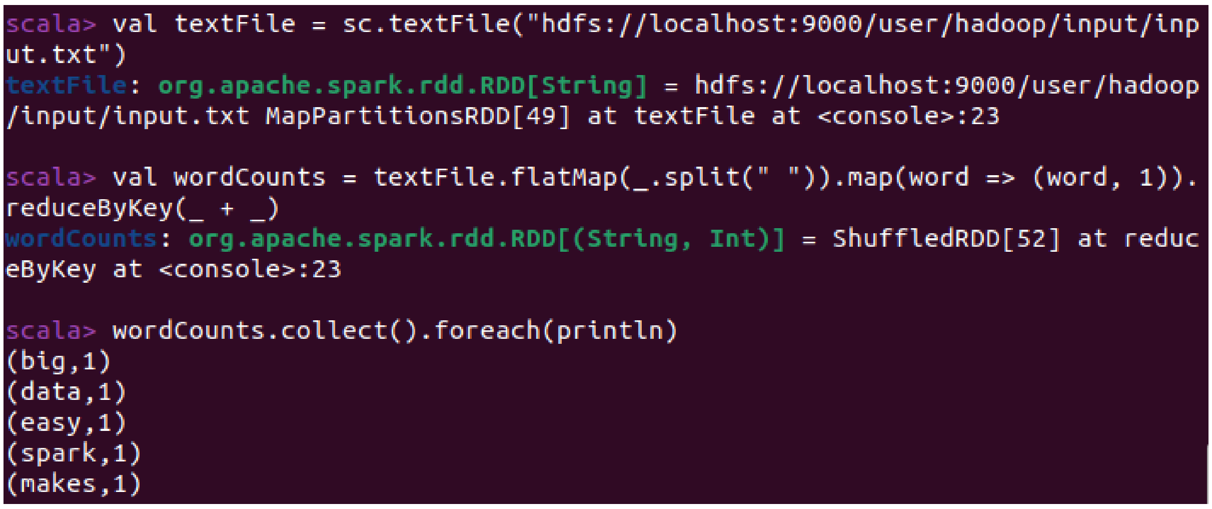
```
hadoop@celab138-ThinkCentre-M720t:~$ echo "spark makes big data
 easy" > input.txt
hdfs dfs -mkdir -p /user/spark/input
hdfs dfs -put input.txt /user/spark/input/
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using
value of HADOOP_PREFIX.
2025-04-04 09:02:50,414 WARN util.NativeCodeLoader: Unable to l
oad native-hadoop library for your platform... using builtin-ja
va classes where applicable
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using
value of HADOOP_PREFIX.
2025-04-04 09:02:51,556 WARN util.NativeCodeLoader: Unable to l
oad native-hadoop library for your platform... using builtin-ja
va classes where applicable
```

To view the result



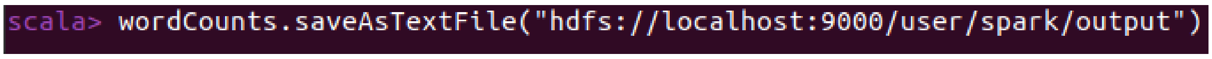
```
hadoop@celab138-ThinkCentre-M720t:~$ hdfs dfs -cat /user/hadoop/input/input.tx
t
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP
_PREFIX.
2025-04-04 10:16:57,259 WARN util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
spark makes big data easy
```

WordCount Logic (in REPL) in spark shell, Loading input file from HDFS



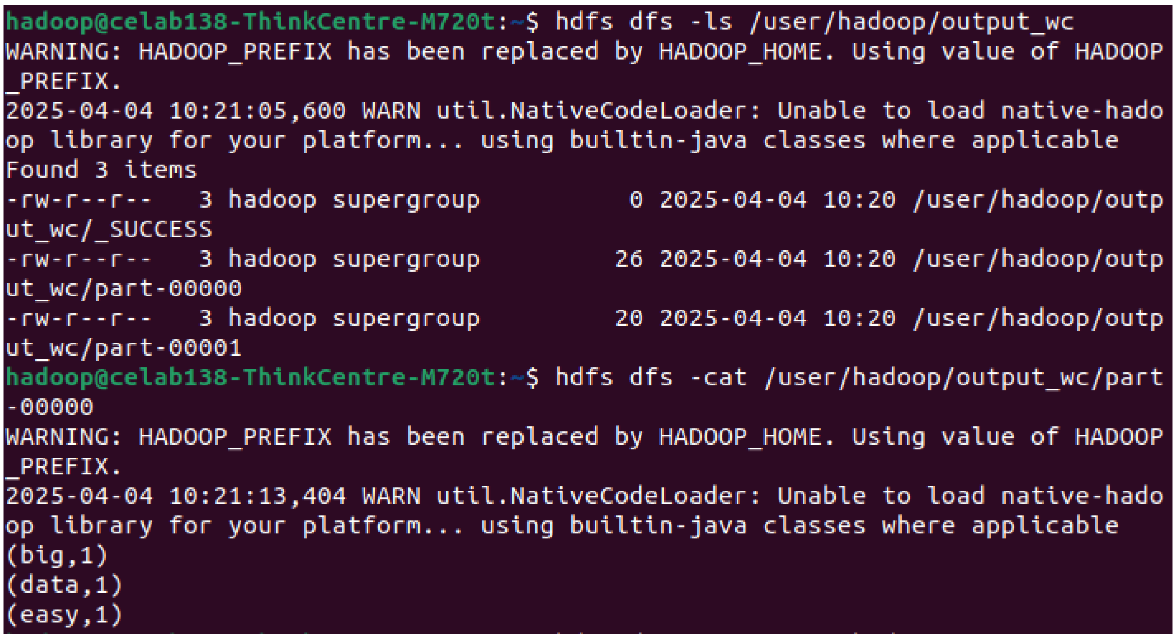
```
scala> val textFile = sc.textFile("hdfs://localhost:9000/user/hadoop/input/inp
ut.txt")
textFile: org.apache.spark.rdd.RDD[String] = hdfs://localhost:9000/user/hadoop
/input/input.txt MapPartitionsRDD[49] at textFile at <console>:23

scala> val wordCounts = textFile.flatMap(_.split(" ")).map(word => (word, 1)).
reduceByKey(_ + _)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[52] at reduc
eByKey at <console>:23

scala> wordCounts.collect().foreach(println)
(big,1)
(data,1)
(easy,1)
(spark,1)
(makes,1)
```

Saving Output to hdfs



```
scala> wordCounts.saveAsTextFile("hdfs://localhost:9000/user/spark/output")
```

Verifying in hdfs



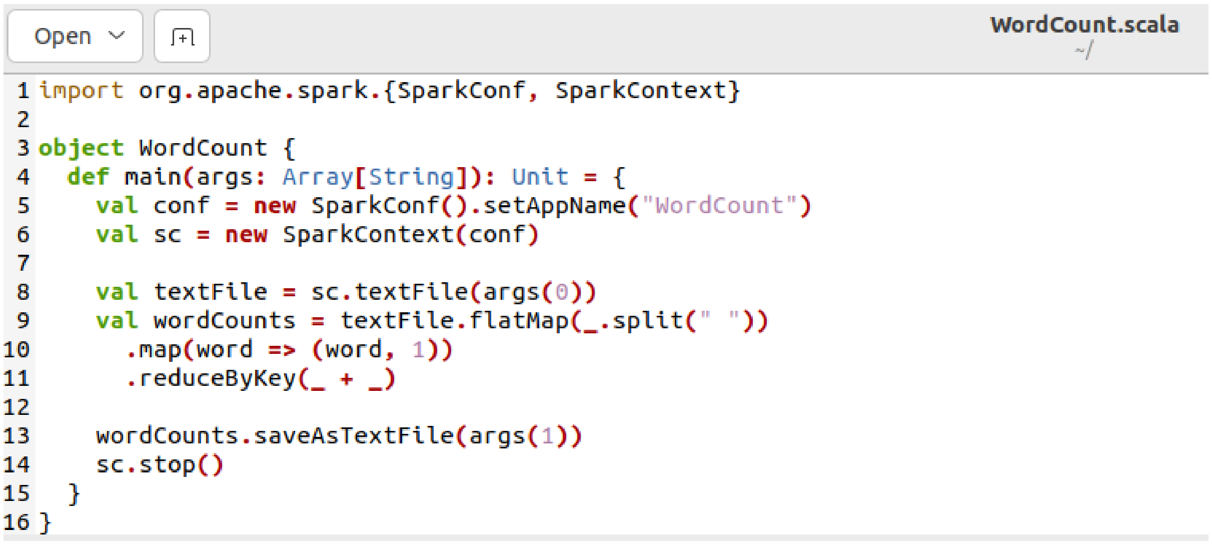
```
hadoop@celab138-ThinkCentre-M720t:~$ hdfs dfs -ls /user/hadoop/output_wc
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP
_PREFIX.
2025-04-04 10:21:05,600 WARN util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--   3 hadoop supergroup          0 2025-04-04 10:20 /user/hadoop/outp
ut_wc/_SUCCESS
-rw-r--r--   3 hadoop supergroup         26 2025-04-04 10:20 /user/hadoop/outp
ut_wc/part-00000
-rw-r--r--   3 hadoop supergroup         20 2025-04-04 10:20 /user/hadoop/outp
ut_wc/part-00001
hadoop@celab138-ThinkCentre-M720t:~$ hdfs dfs -cat /user/hadoop/output_wc/part
-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP
_PREFIX.
2025-04-04 10:21:13,404 WARN util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
(big,1)
(data,1)
(easy,1)
```

2. Develop Wordcount program using cluster mode(standalone or YARN mode)

Spark Standalone mode:

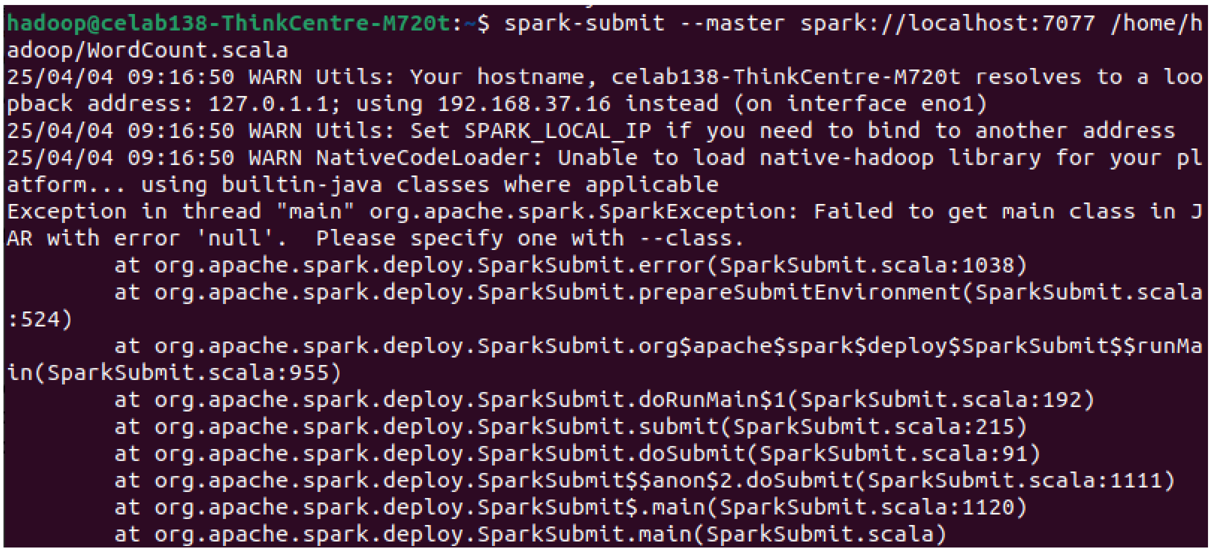
We can launch a standalone cluster either manually, by starting a master and workers by hand, or use our provided launch scripts. It is also possible to run these daemons on a single machine for testing.

Creating WordCount.scala at /home/hadoop



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

object WordCount {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().setAppName("WordCount")
    val sc = new SparkContext(conf)

    val textFile = sc.textFile(args(0))
    val wordCounts = textFile.flatMap(_.split(" "))
      .map(word => (word, 1))
      .reduceByKey(_ + _)

    wordCounts.saveAsTextFile(args(1))
    sc.stop()
  }
}
```

Submitting a Scala script directly



```
hadoop@celab138-ThinkCentre-M720t:~$ spark-submit --master spark://localhost:7077 /home/h
adoop/WordCount.scala
25/04/04 09:16:50 WARN Utils: Your hostname, celab138-ThinkCentre-M720t resolves to a loo
pback address: 127.0.1.1; using 192.168.37.16 instead (on interface eno1)
25/04/04 09:16:50 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/04/04 09:16:50 WARN NativeCodeLoader: Unable to load native-hadoop library for your pl
atform... using builtin-java classes where applicable
Exception in thread "main" org.apache.spark.SparkException: Failed to get main class in J
AR with error 'null'.  Please specify one with --class.
        at org.apache.spark.deploy.SparkSubmit.error(SparkSubmit.scala:1038)
        at org.apache.spark.deploy.SparkSubmit.prepareSubmitEnvironment(SparkSubmit.scala
:524)
        at org.apache.spark.deploy.SparkSubmit.org$apache$spark$deploy$SparkSubmit$$runMa
in(SparkSubmit.scala:955)
        at org.apache.spark.deploy.SparkSubmit.doRunMain$1(SparkSubmit.scala:192)
        at org.apache.spark.deploy.SparkSubmit.submit(SparkSubmit.scala:215)
        at org.apache.spark.deploy.SparkSubmit.doSubmit(SparkSubmit.scala:91)
        at org.apache.spark.deploy.SparkSubmit$$anon$2.doSubmit(SparkSubmit.scala:1111)
        at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:1120)
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

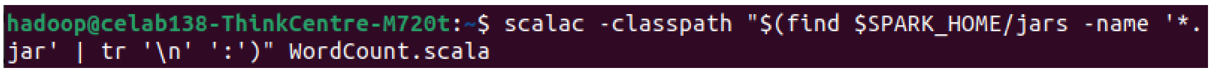
There are 2 ways to execute the program:

1) We can use IDE and create a jar file and then submit the jar file to the master for execution
I. Now create a jar file in IDE or CLI
II. Run the jar file and check the output status
III. Submit the application to spark so that spark can execute the application

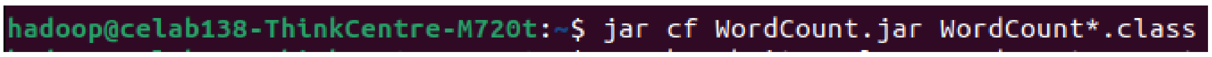
2) use CLI to create a scala program and then run it directly on the shell
Refer the section of creating scala script

I am using way 1) here:

Compiling in jar

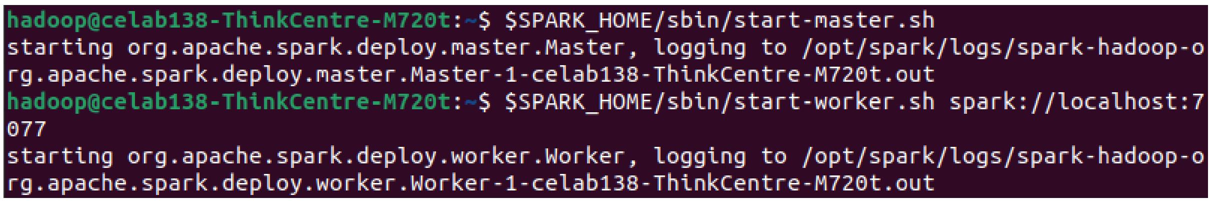


```
hadoop@celab138-ThinkCentre-M720t:~$ scalac -classpath "$(find $SPARK_HOME/jars -name '*.
jar' | tr '\n' ':')" WordCount.scala
```



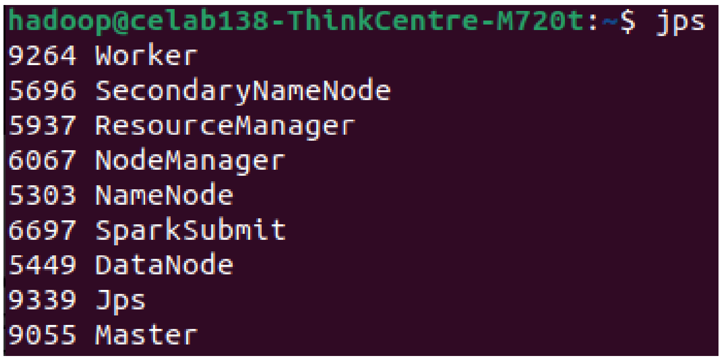
```
hadoop@celab138-ThinkCentre-M720t:~$ jar cf WordCount.jar WordCount*.class
```

Now we start Spark Master and Spark Worker as below



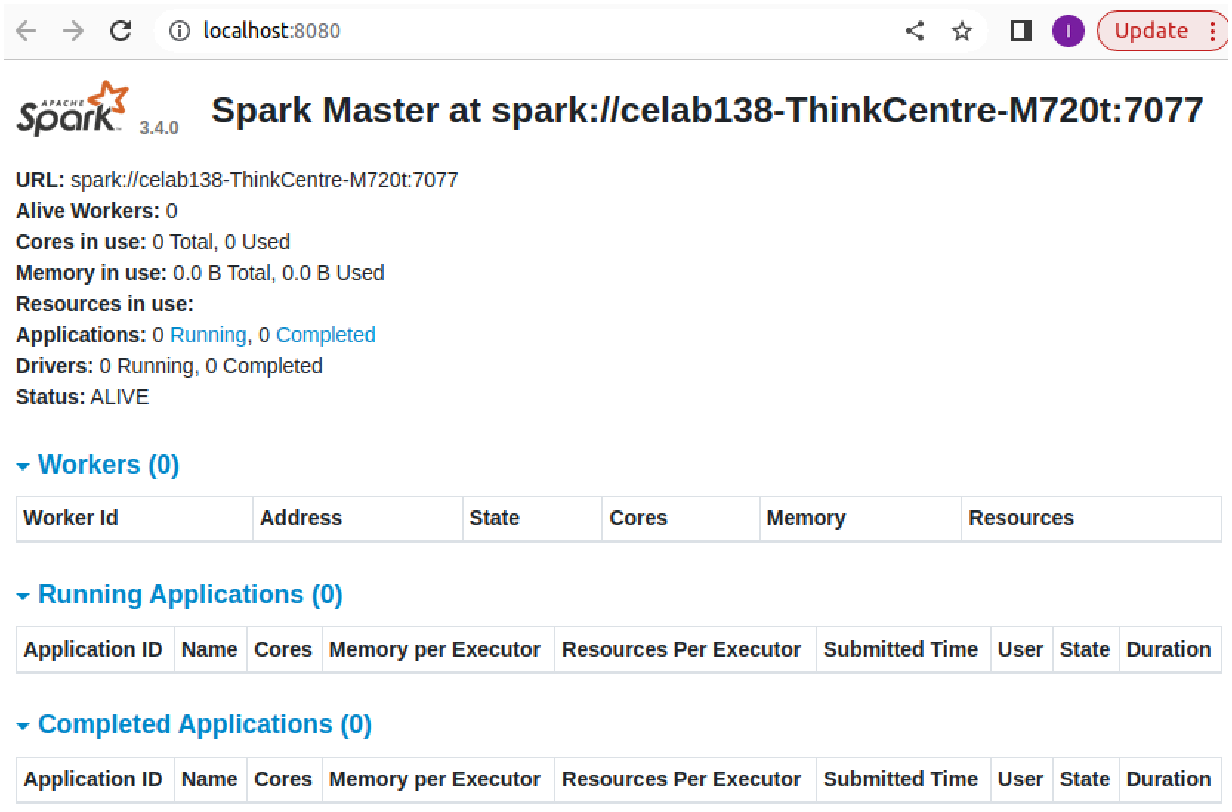
```
hadoop@celab138-ThinkCentre-M720t:~$ $SPARK_HOME/sbin/start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-hadoop-o
rg.apache.spark.deploy.master.Master-1-celab138-ThinkCentre-M720t.out
hadoop@celab138-ThinkCentre-M720t:~$ $SPARK_HOME/sbin/start-worker.sh spark://localhost:7
077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-hadoop-o
rg.apache.spark.deploy.worker.Worker-1-celab138-ThinkCentre-M720t.out
```

Verifying with jps



```
hadoop@celab138-ThinkCentre-M720t:~$ jps
9264 Worker
5696 SecondaryNameNode
5937 ResourceManager
6067 NodeManager
5303 NameNode
6697 SparkSubmit
5449 DataNode
9339 Jps
9055 Master
```

I can now Access Spark UI on **http://localhost:8080**



localhost:8080

Spark 3.4.0

Spark Master at spark://celab138-ThinkCentre-M720t:7077

URL: spark://celab138-ThinkCentre-M720t:7077
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (0)

Worker Id	Address	State	Cores	Memory	Resources

Running Applications (0)

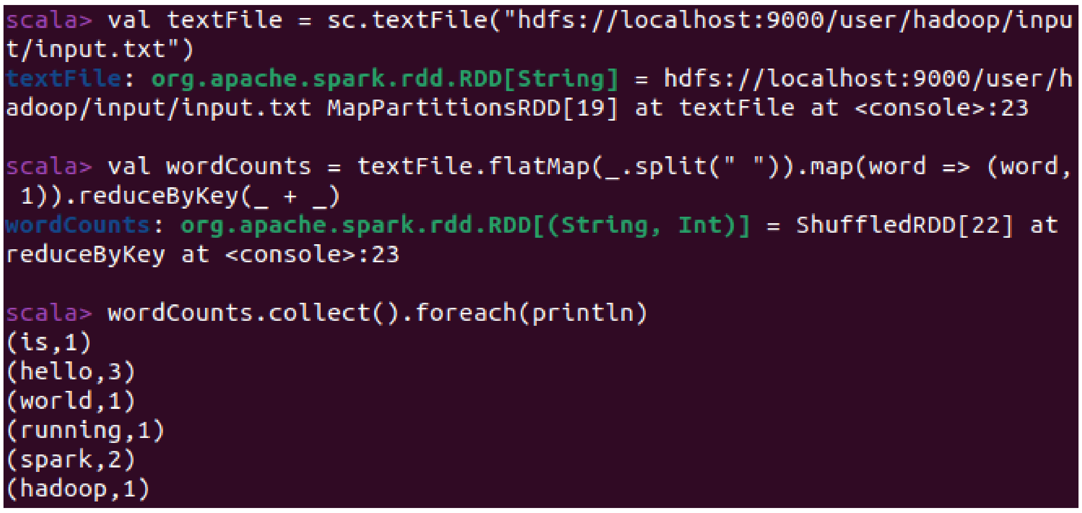
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Loading and running the scala file from shell:

Also, verifying the output on scala as below

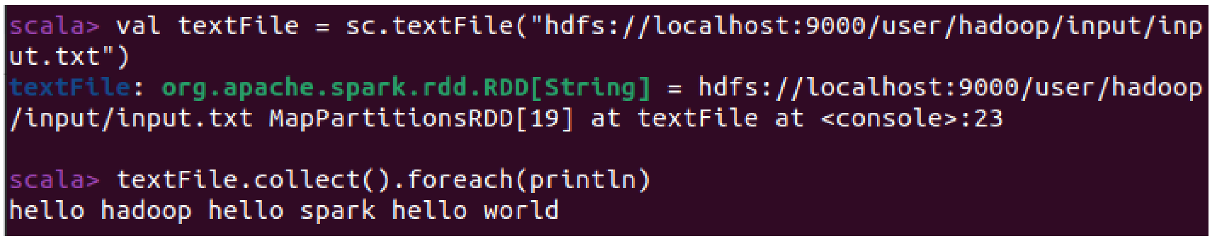


```
scala> val textFile = sc.textFile("hdfs://localhost:9000/user/hadoop/inpu
t/input.txt")
textFile: org.apache.spark.rdd.RDD[String] = hdfs://localhost:9000/user/h
adoop/input/input.txt MapPartitionsRDD[19] at textFile at <console>:23

scala> val wordCounts = textFile.flatMap(_.split(" ")).map(word => (word,
 1)).reduceByKey(_ + _)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[22] at
reduceByKey at <console>:23

scala> wordCounts.collect().foreach(println)
(is,1)
(hello,3)
(world,1)
(running,1)
(spark,2)
(hadoop,1)
```

3. Develop any custom map reduce problem using apache spark and hadoop.

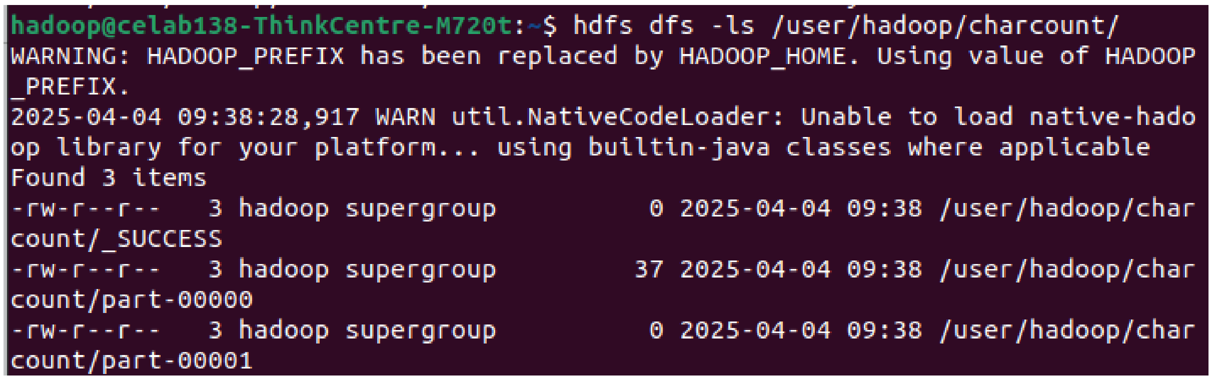
We have taken a custom problem: **Character Frequency Count**

Loading the Input File from HDFS, and checking it afterwards by using: *textFile.collect().foreach(println)*



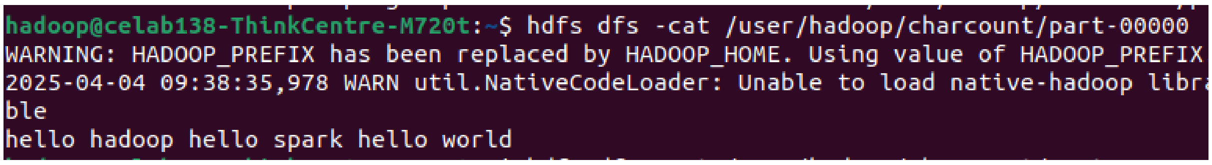
```
scala> val textFile = sc.textFile("hdfs://localhost:9000/user/hadoop/input/inp
ut.txt")
textFile: org.apache.spark.rdd.RDD[String] = hdfs://localhost:9000/user/hadoop
/input/input.txt MapPartitionsRDD[19] at textFile at <console>:23

scala> textFile.collect().foreach(println)
hello hadoop hello spark hello world
```

Checking if the file exists in HDFS



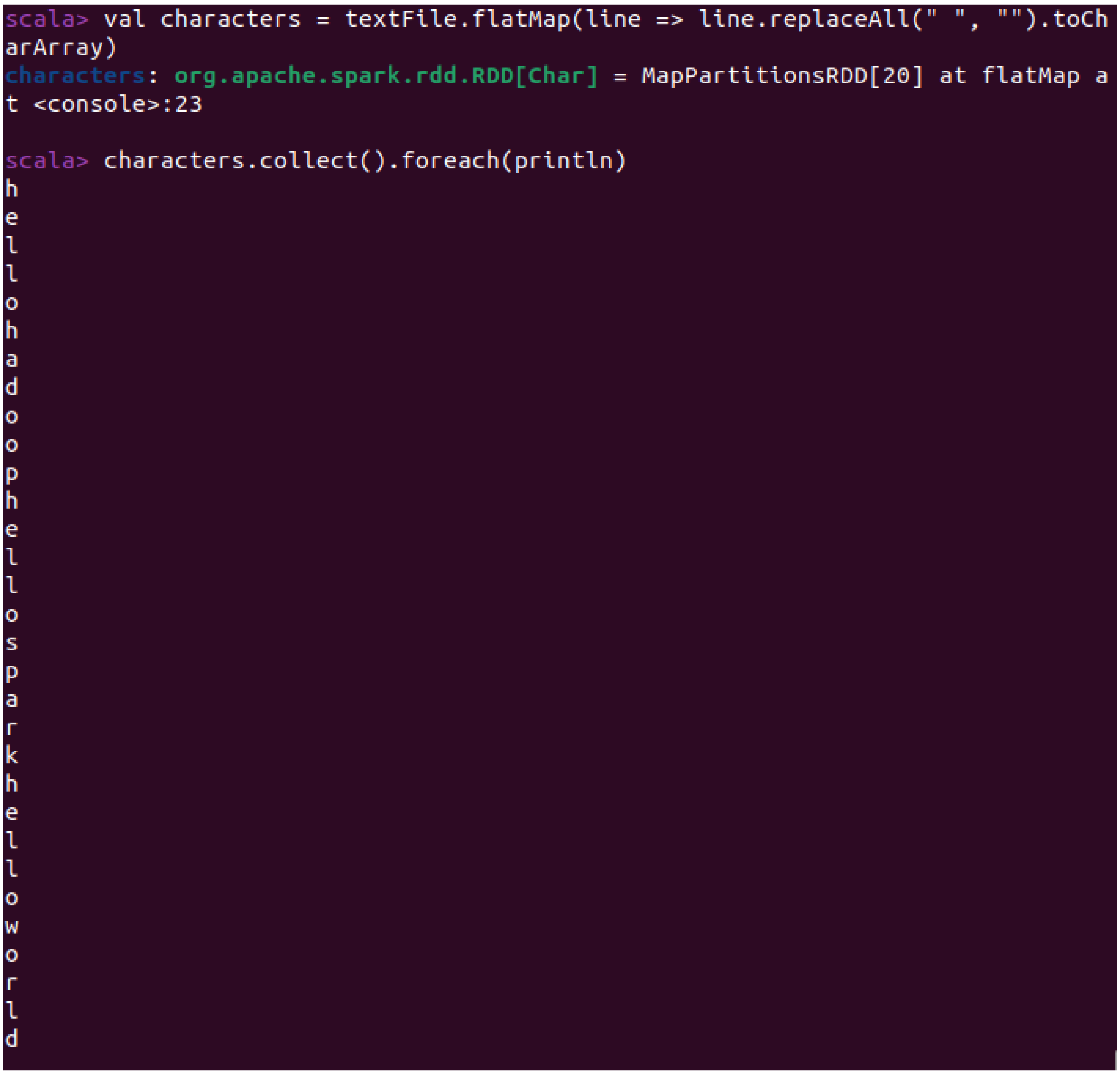
```
hadoop@celab138-ThinkCentre-M720t:~$ hdfs dfs -ls /user/hadoop/charcount/
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP
_PREFIX.
2025-04-04 09:38:28,917 WARN util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--   3 hadoop supergroup          0 2025-04-04 09:38 /user/hadoop/char
count/_SUCCESS
-rw-r--r--   3 hadoop supergroup         37 2025-04-04 09:38 /user/hadoop/char
count/part-00000
-rw-r--r--   3 hadoop supergroup          0 2025-04-04 09:38 /user/hadoop/char
count/part-00001
```

Printing output on hdfs



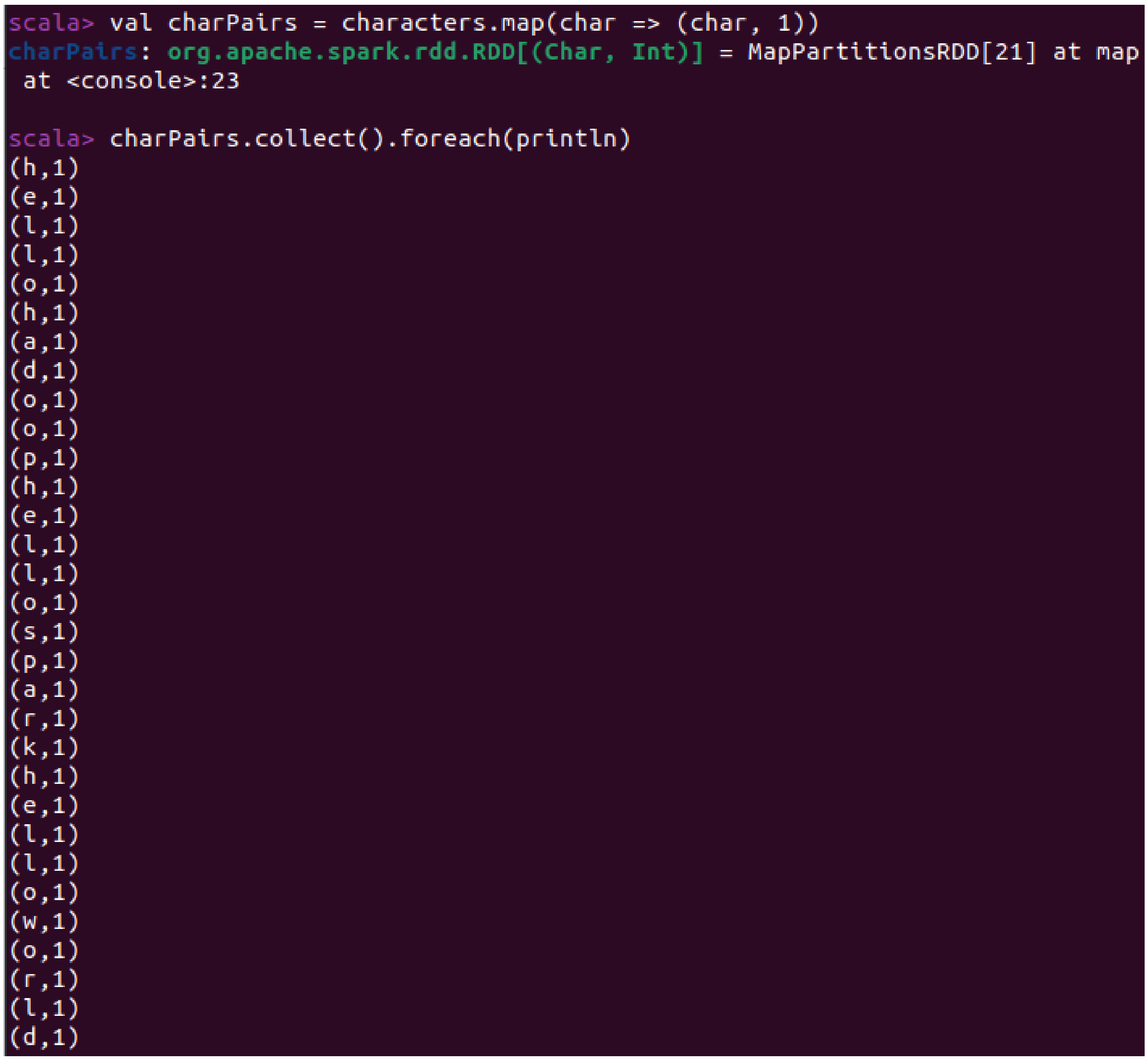
```
hadoop@celab138-ThinkCentre-M720t:~$ hdfs dfs -cat /user/hadoop/charcount/part-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX
2025-04-04 09:38:35,978 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ble
hello hadoop hello spark hello world
```

FlatMap to Characters (transforming the text into individual characters) :



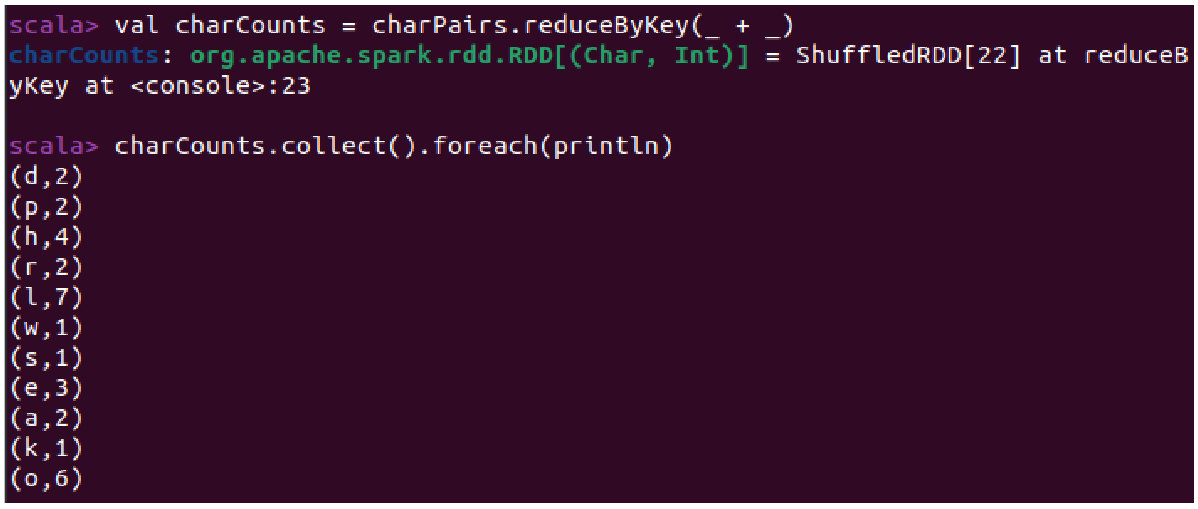
```
scala> val characters = textFile.flatMap(line => line.replaceAll(" ", "").toCh
arArray)
characters: org.apache.spark.rdd.RDD[Char] = MapPartitionsRDD[20] at flatMap a
t <console>:23

scala> characters.collect().foreach(println)
h
e
l
l
o
h
a
d
o
o
p
h
e
l
l
o
s
p
a
r
k
h
e
l
l
o
w
o
r
l
d
```

Converting to Key-Value Pairs and checking the output:



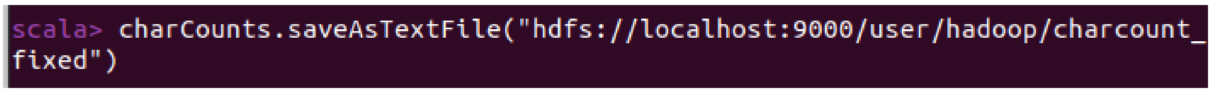
```
scala> val charPairs = characters.map(char => (char, 1))
charPairs: org.apache.spark.rdd.RDD[(Char, Int)] = MapPartitionsRDD[21] at map
 at <console>:23

scala> charPairs.collect().foreach(println)
(h,1)
(e,1)
(l,1)
(l,1)
(o,1)
(h,1)
(a,1)
(d,1)
(o,1)
(o,1)
(p,1)
(h,1)
(e,1)
(l,1)
(l,1)
(o,1)
(s,1)
(p,1)
(a,1)
(r,1)
(k,1)
(h,1)
(e,1)
(l,1)
(l,1)
(o,1)
(w,1)
(o,1)
(r,1)
(l,1)
(d,1)
```

Performing ReduceByKey to Count



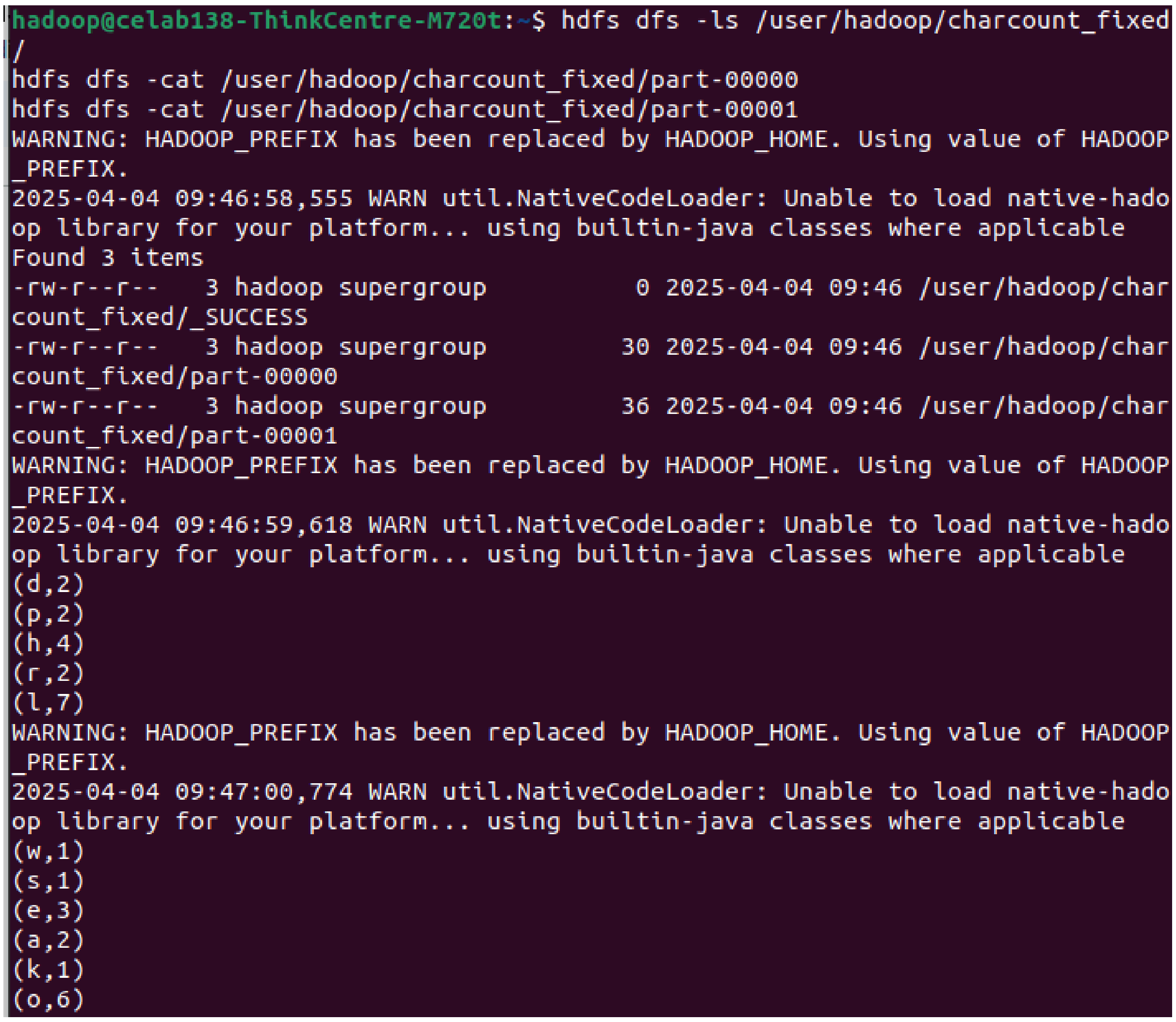
```
scala> val charCounts = charPairs.reduceByKey(_ + _)
charCounts: org.apache.spark.rdd.RDD[(Char, Int)] = ShuffledRDD[22] at reduceB
yKey at <console>:23

scala> charCounts.collect().foreach(println)
(d,2)
(p,2)
(h,4)
(r,2)
(l,7)
(w,1)
(s,1)
(e,3)
(a,2)
(k,1)
(o,6)
```

Saving Output to HDFS



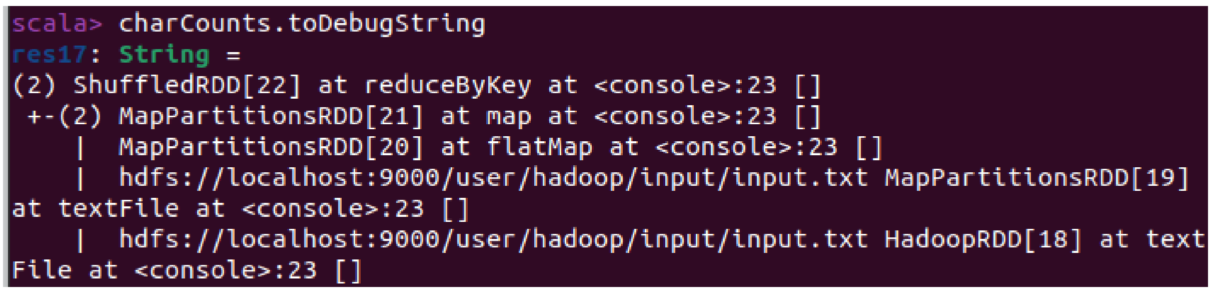
```
scala> charCounts.saveAsTextFile("hdfs://localhost:9000/user/hadoop/charcount_
fixed")
```

Verifying saved data:



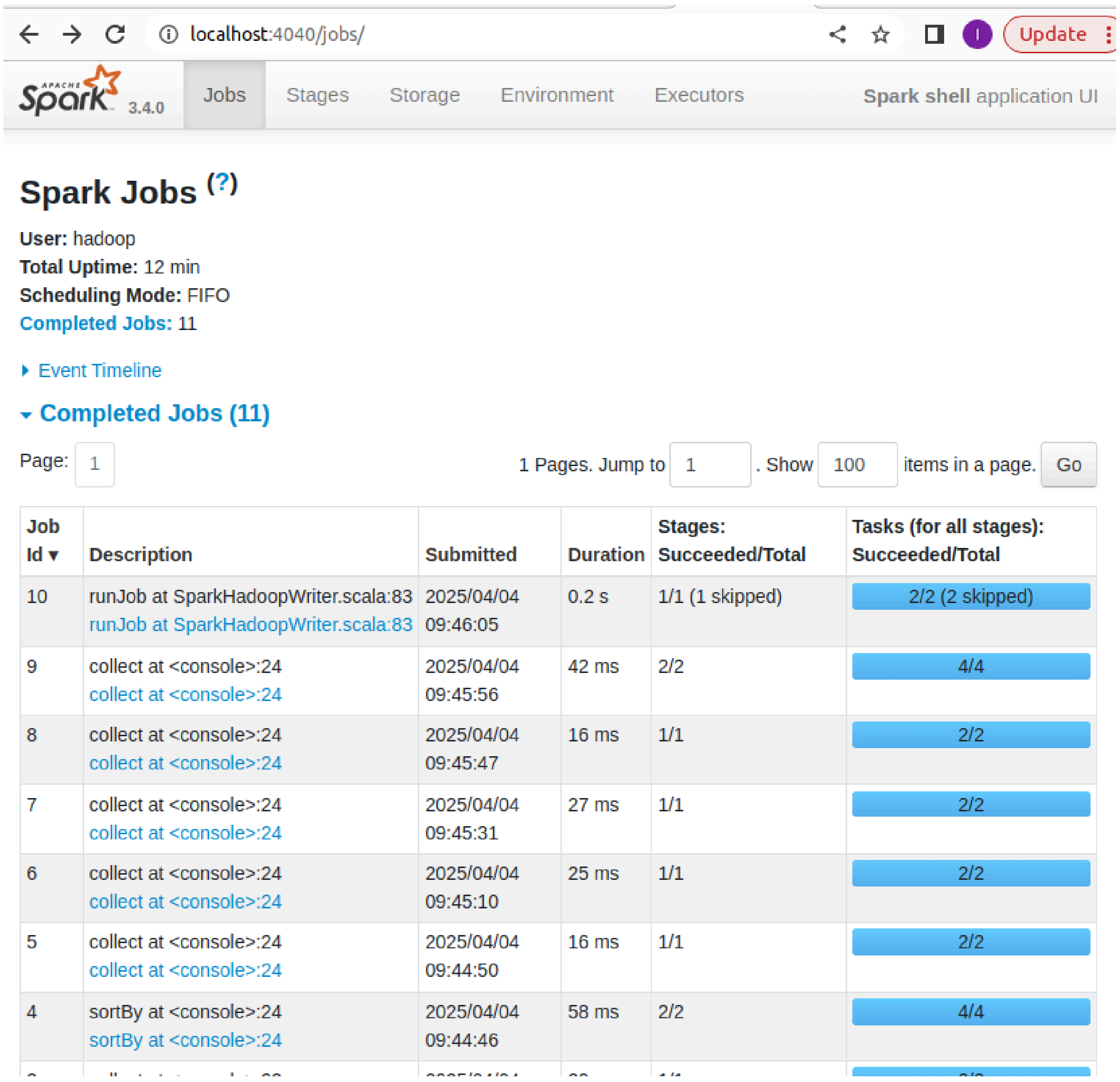
```
hadoop@celab138-ThinkCentre-M720t:~$ hdfs dfs -ls /user/hadoop/charcount_fixed
/
hdfs dfs -cat /user/hadoop/charcount_fixed/part-00000
hdfs dfs -cat /user/hadoop/charcount_fixed/part-00001
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP
_PREFIX.
2025-04-04 09:46:58,555 WARN util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--   3 hadoop supergroup          0 2025-04-04 09:46 /user/hadoop/char
count_fixed/_SUCCESS
-rw-r--r--   3 hadoop supergroup         30 2025-04-04 09:46 /user/hadoop/char
count_fixed/part-00000
-rw-r--r--   3 hadoop supergroup         36 2025-04-04 09:46 /user/hadoop/char
count_fixed/part-00001
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP
_PREFIX.
2025-04-04 09:46:59,618 WARN util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
(d,2)
(p,2)
(h,4)
(r,2)
(l,7)
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP
_PREFIX.
2025-04-04 09:47:00,774 WARN util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
(w,1)
(s,1)
(e,3)
(a,2)
(k,1)
(o,6)
```

To visulalize DAG



```
scala> charCounts.toDebugString
res17: String =
(2) ShuffledRDD[22] at reduceByKey at <console>:23 []
 +-(2) MapPartitionsRDD[21] at map at <console>:23 []
    |  MapPartitionsRDD[20] at flatMap at <console>:23 []
    |  hdfs://localhost:9000/user/hadoop/input/input.txt MapPartitionsRDD[19]
at textFile at <console>:23 []
    |  hdfs://localhost:9000/user/hadoop/input/input.txt HadoopRDD[18] at text
File at <console>:23 []
```

Spark UI is present at http://localhost:4040/jobs/



Spark Jobs (?)

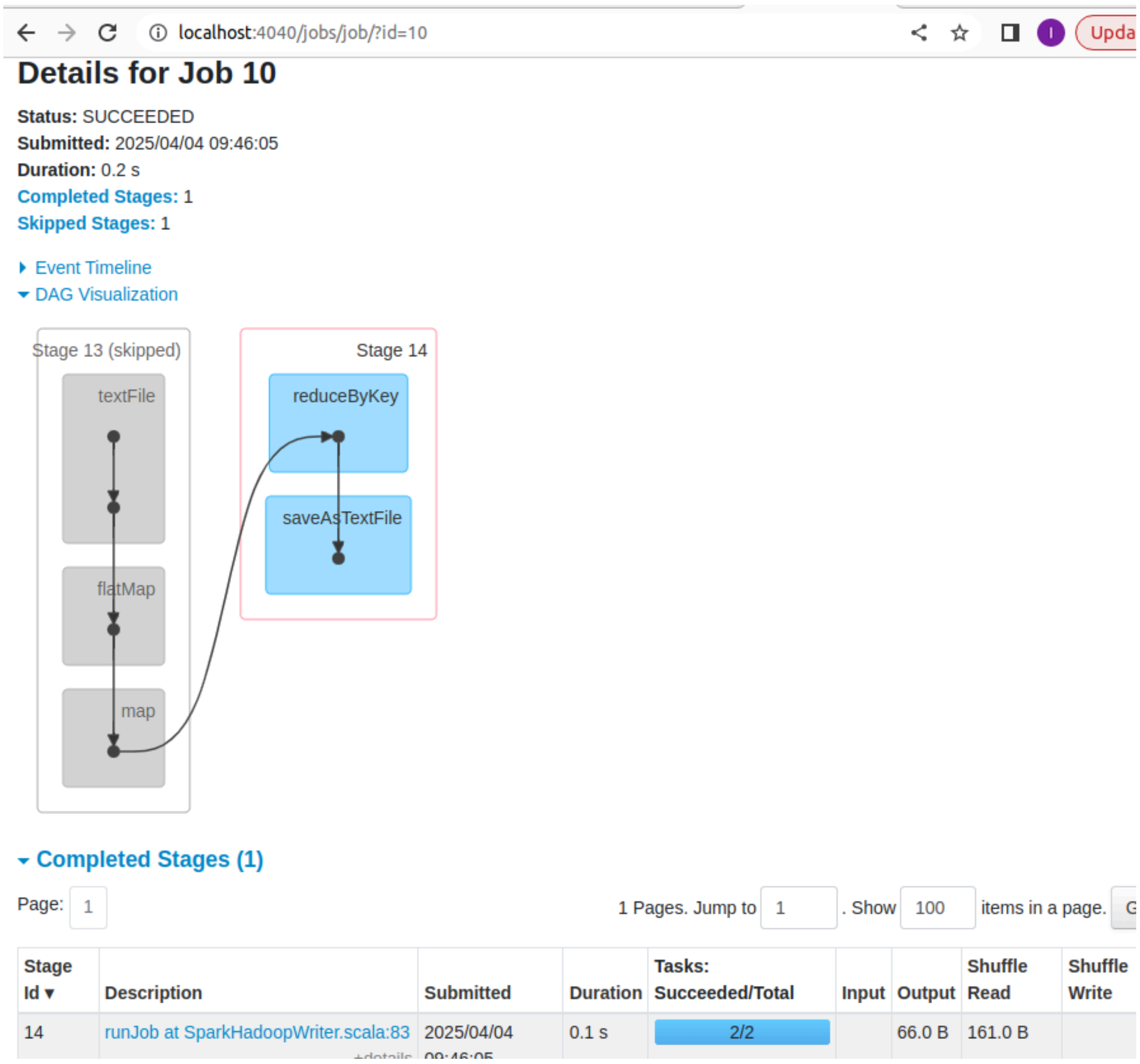
User: hadoop
Total Uptime: 12 min
Scheduling Mode: FIFO
Completed Jobs: 11

▸ Event Timeline

▾ Completed Jobs (11)

Page: 1 | 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
10	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83	2025/04/04 09:46:05	0.2 s	1/1 (1 skipped)	2/2 (2 skipped)
9	collect at <console>:24 collect at <console>:24	2025/04/04 09:45:56	42 ms	2/2	4/4
8	collect at <console>:24 collect at <console>:24	2025/04/04 09:45:47	16 ms	1/1	2/2
7	collect at <console>:24 collect at <console>:24	2025/04/04 09:45:31	27 ms	1/1	2/2
6	collect at <console>:24 collect at <console>:24	2025/04/04 09:45:10	25 ms	1/1	2/2
5	collect at <console>:24 collect at <console>:24	2025/04/04 09:44:50	16 ms	1/1	2/2
4	sortBy at <console>:24 sortBy at <console>:24	2025/04/04 09:44:46	58 ms	2/2	4/4



Details for Job 10

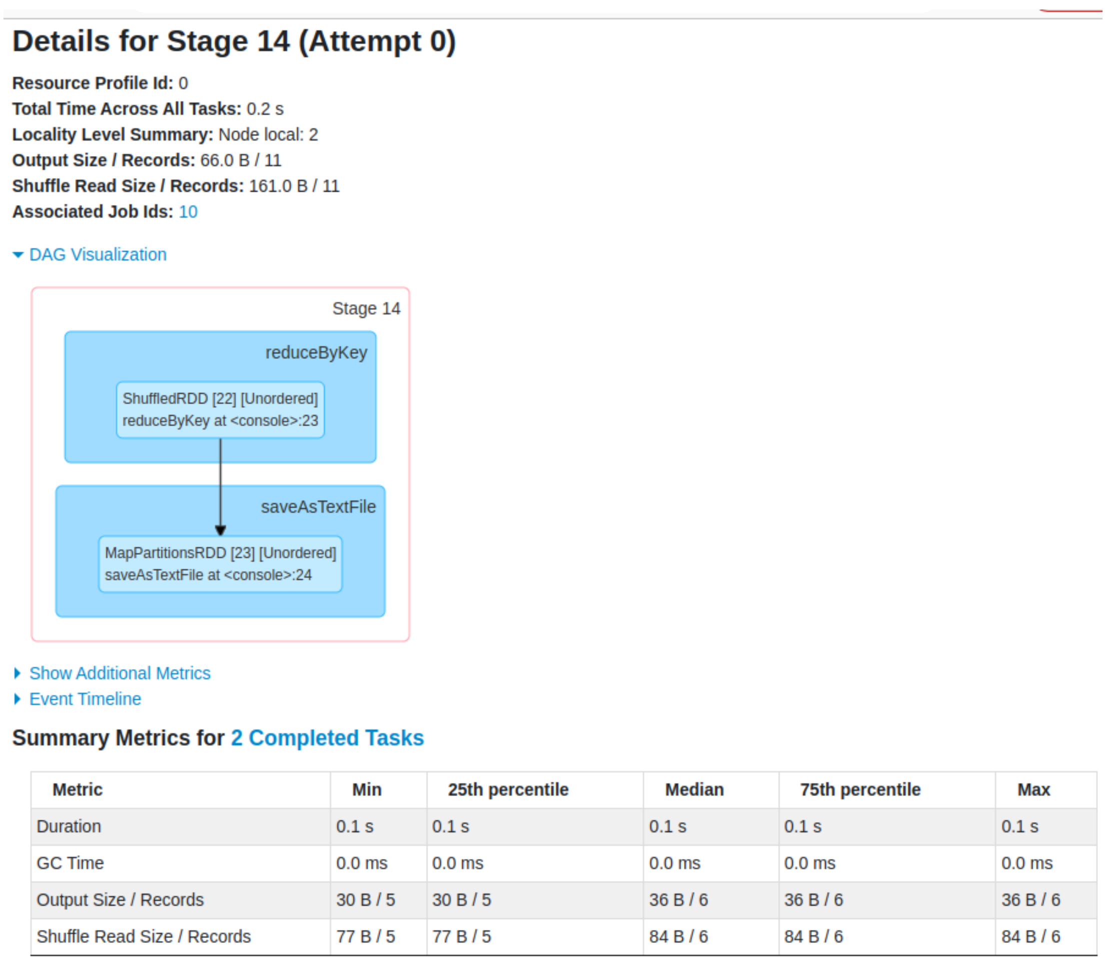
Status: SUCCEEDED
Submitted: 2025/04/04 09:46:05
Duration: 0.2 s
Completed Stages: 1
Skipped Stages: 1

▸ Event Timeline
▾ DAG Visualization

▾ Completed Stages (1)

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page.

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
14	runJob at SparkHadoopWriter.scala:83 +details	2025/04/04 09:46:05	0.1 s	2/2		66.0 B	161.0 B	



Details for Stage 14 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 0.2 s
Locality Level Summary: Node local: 2
Output Size / Records: 66.0 B / 11
Shuffle Read Size / Records: 161.0 B / 11
Associated Job Ids: 10

DAG Visualization

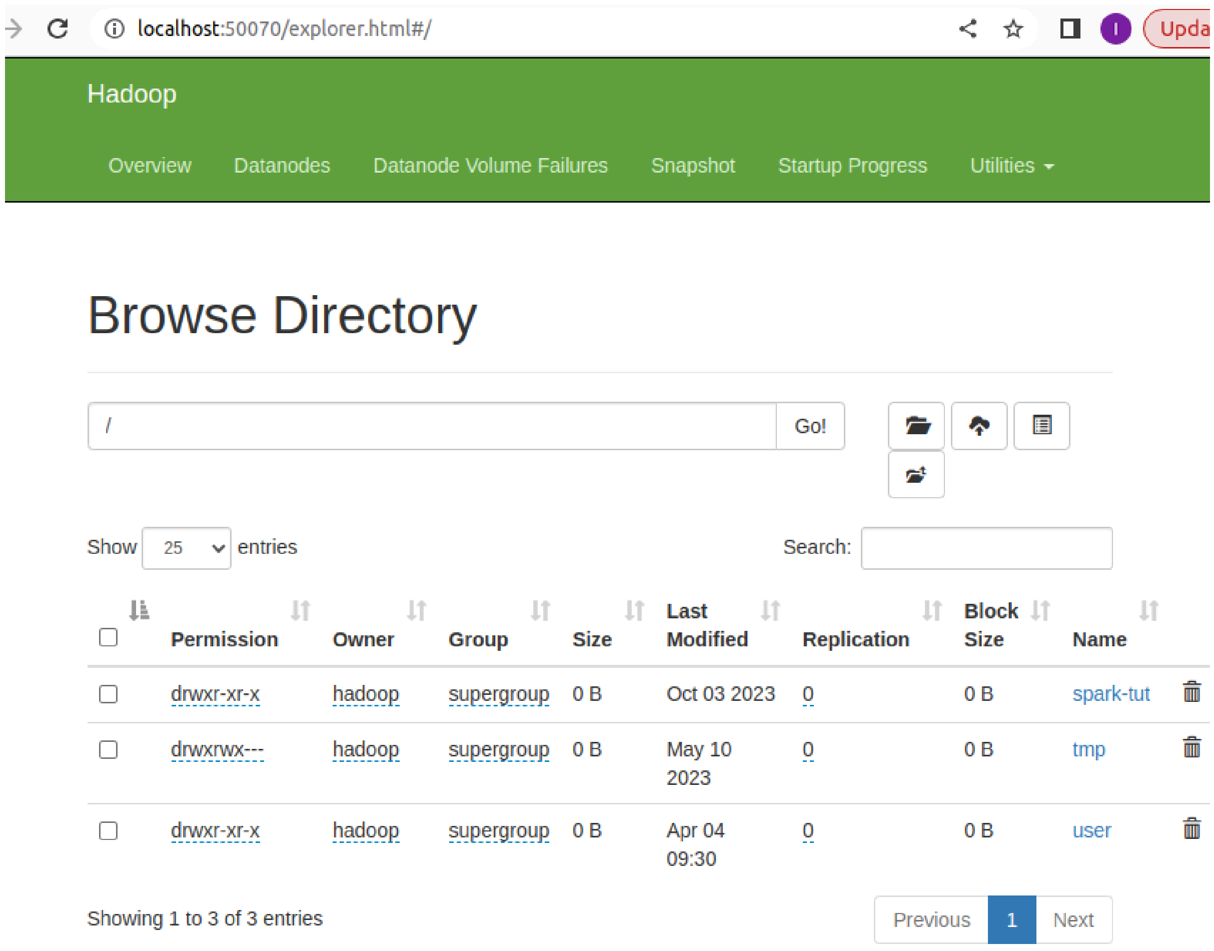
Stage 14
reduceByKey
ShuffledRDD [22] [Unordered]
reduceByKey at <console>:23
saveAsTextFile
MapPartitionsRDD [23] [Unordered]
saveAsTextFile at <console>:24

Show Additional Metrics
Event Timeline

Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms
Output Size / Records	30 B / 5	30 B / 5	36 B / 6	36 B / 6	36 B / 6
Shuffle Read Size / Records	77 B / 5	77 B / 5	84 B / 6	84 B / 6	84 B / 6

Also, The file system on http://localhost:50070 looks like:



localhost:50070/explorer.html#/

Hadoop

Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/ Go!

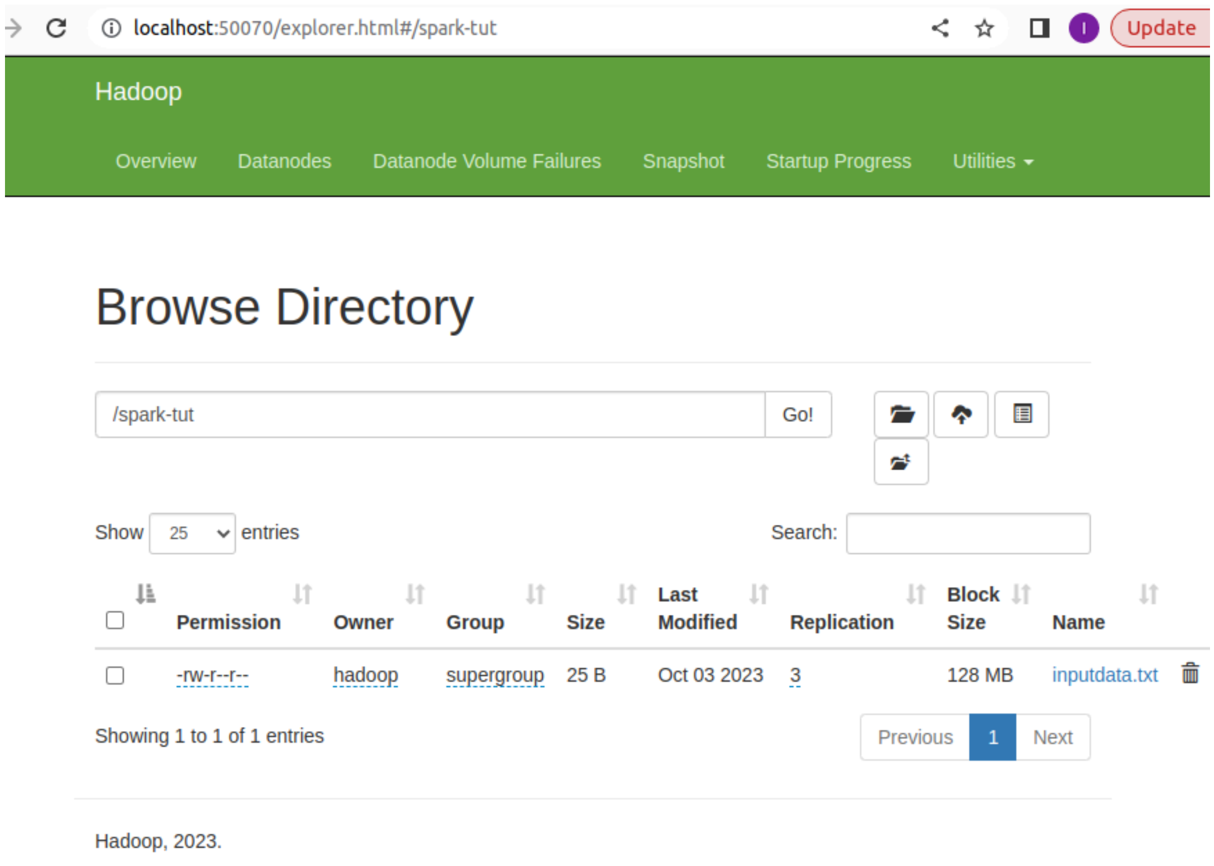
Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
	drwxr-xr-x	hadoop	supergroup	0 B	Oct 03 2023	0	0 B	spark-tut
	drwxrwx---	hadoop	supergroup	0 B	May 10 2023	0	0 B	tmp
	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:30	0	0 B	user

Showing 1 to 3 of 3 entries

Previous 1 Next

In spark-tut:


localhost:50070/explorer.html#/spark-tut

Update
Hadoop
Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/spark-tut Go!

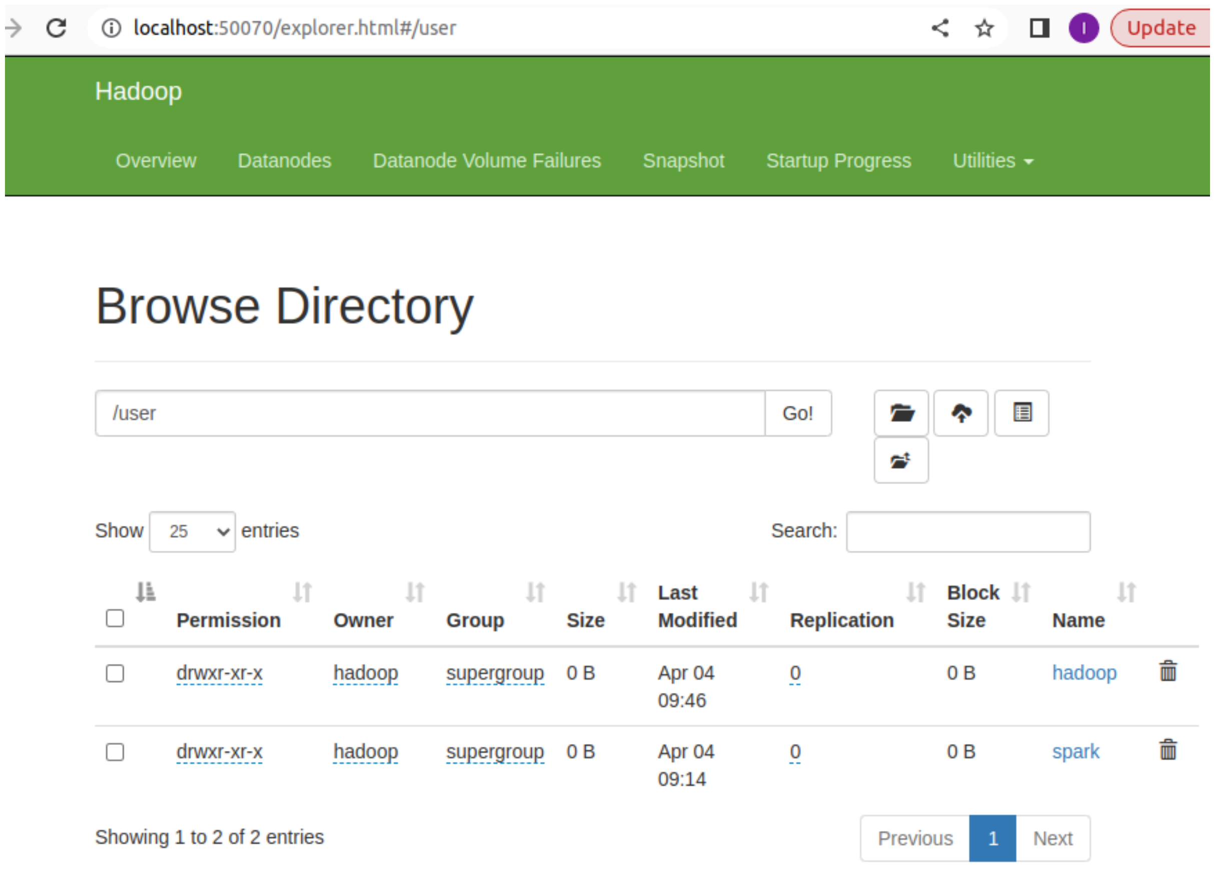
Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	hadoop	supergroup	25 B	Oct 03 2023	3	128 MB	inputdata.txt

Showing 1 to 1 of 1 entries

Hadoop, 2023.

In user:


localhost:50070/explorer.html#/user

Update
Hadoop
Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

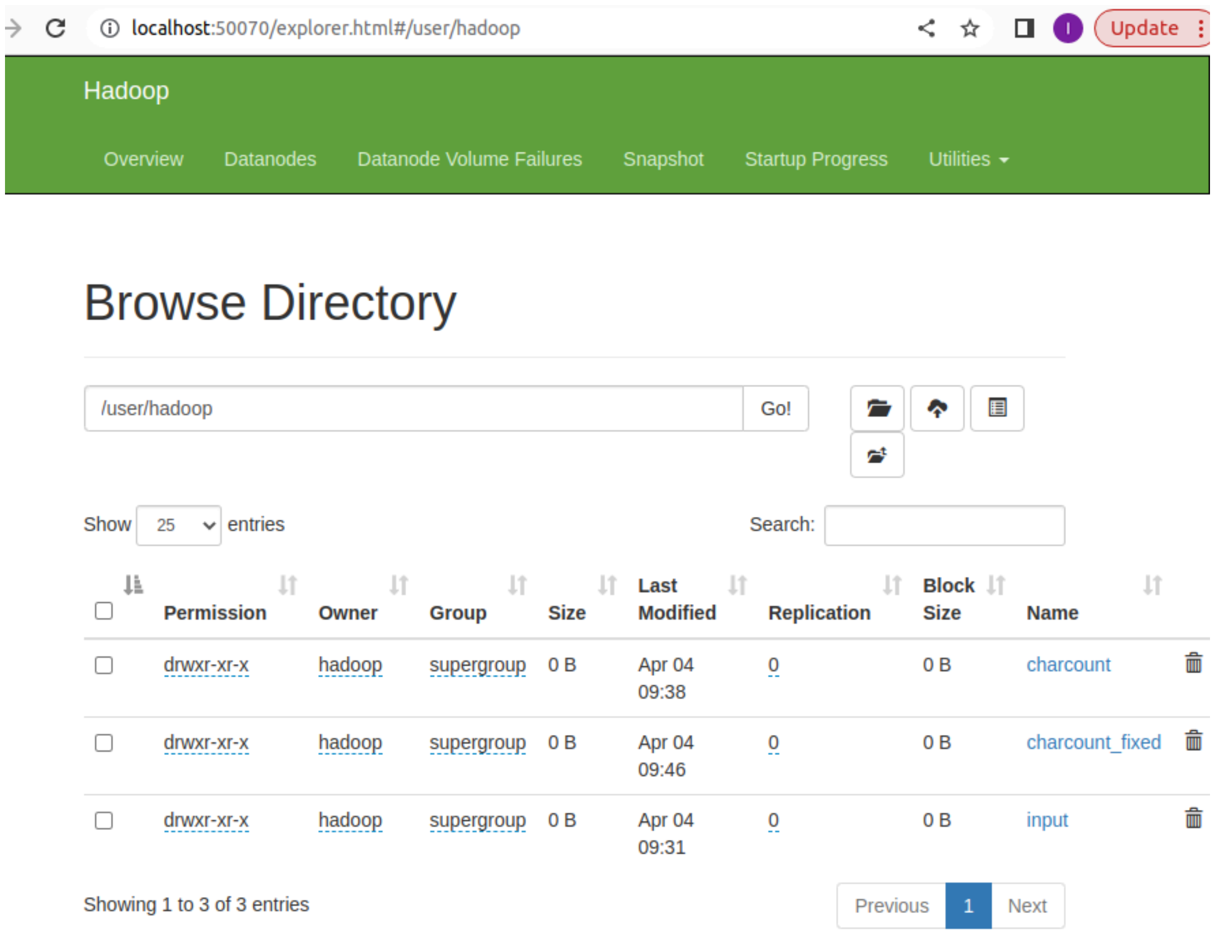
/user Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:46	0	0 B	hadoop
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:14	0	0 B	spark

Showing 1 to 2 of 2 entries

hadoop:



Browse Directory

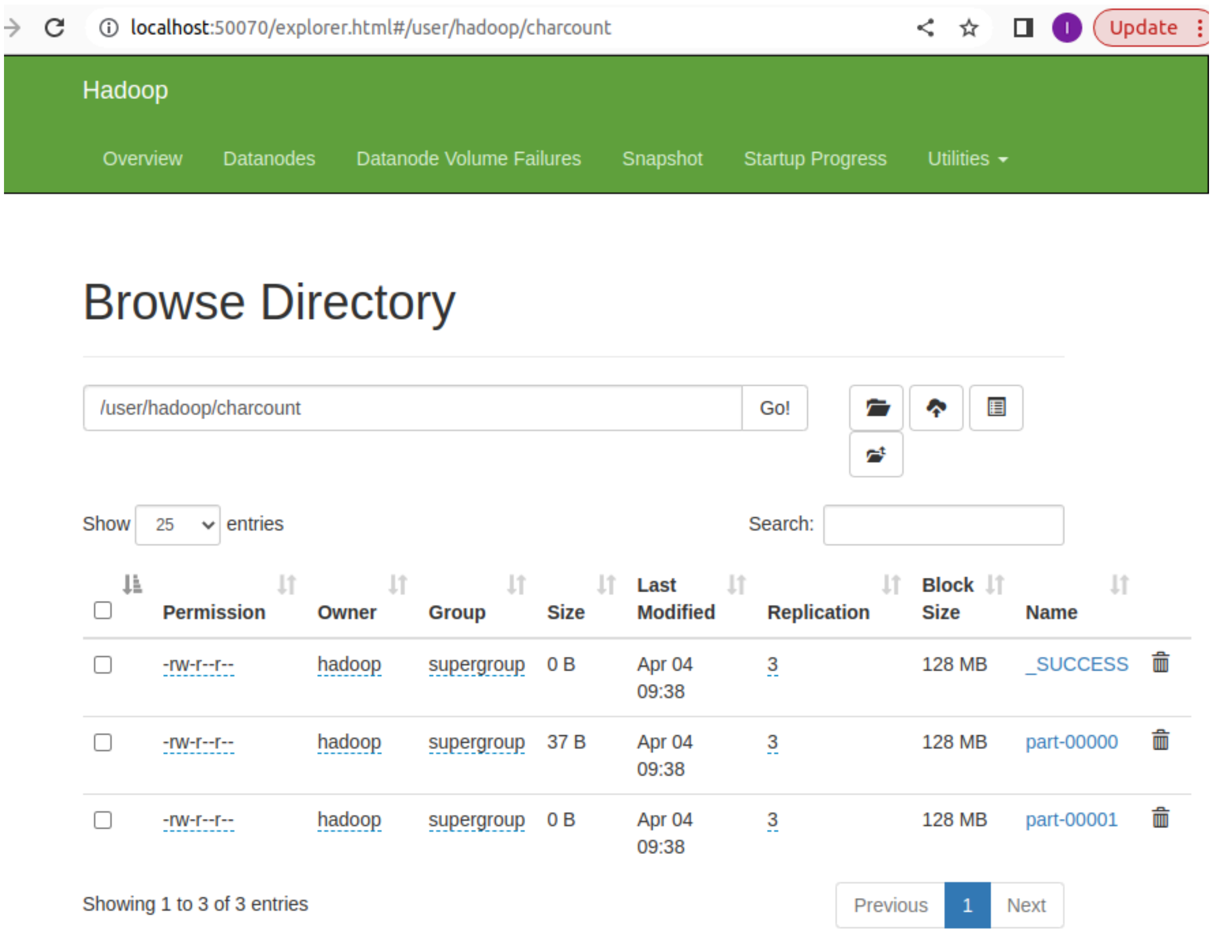
/user/hadoop Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:38	0	0 B	charcount
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:46	0	0 B	charcount_fixed
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:31	0	0 B	input

Showing 1 to 3 of 3 entries

Previous 1 Next



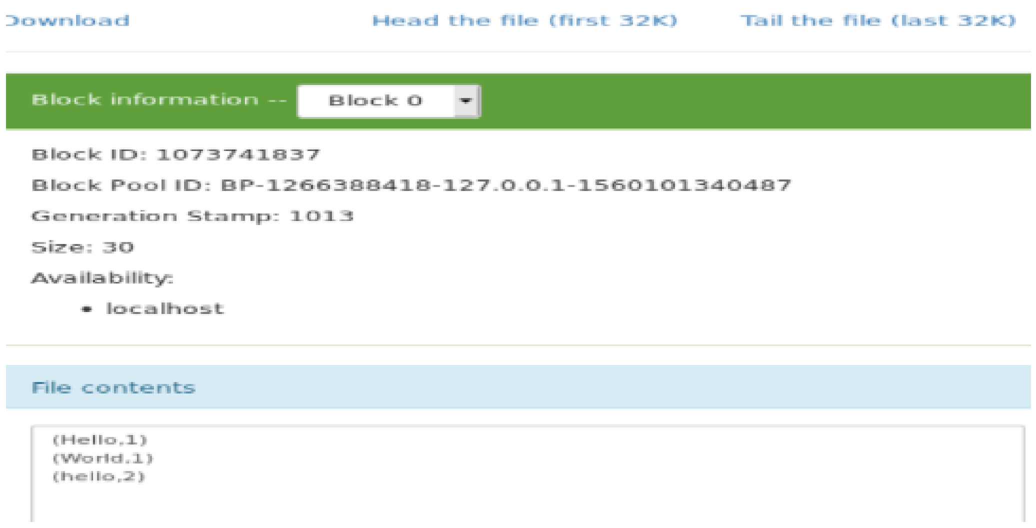
Browse Directory

/user/hadoop/charcount

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	hadoop	supergroup	0 B	Apr 04 09:38	3	128 MB	_SUCCESS
☐	-rw-r--r--	hadoop	supergroup	37 B	Apr 04 09:38	3	128 MB	part-00000
☐	-rw-r--r--	hadoop	supergroup	0 B	Apr 04 09:38	3	128 MB	part-00001

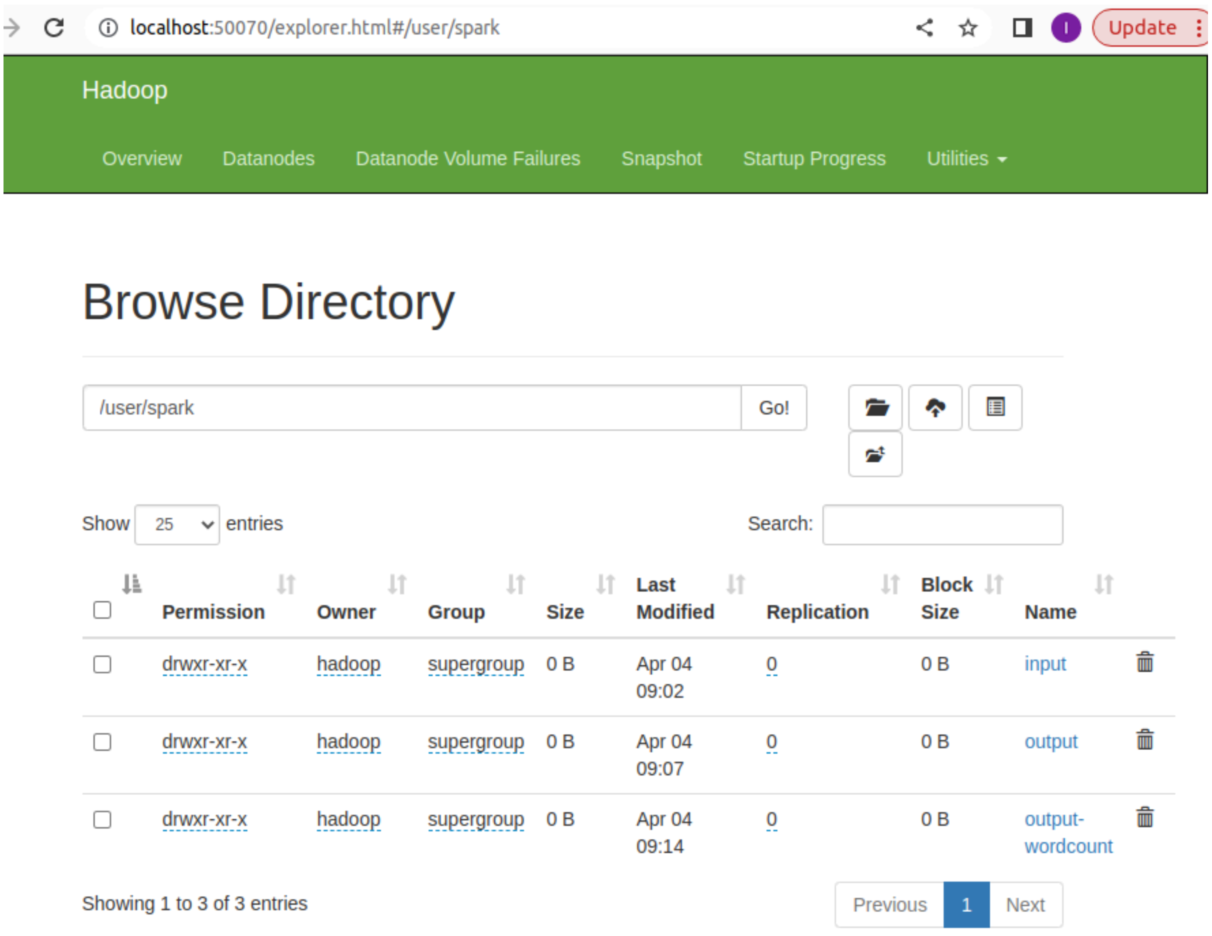
Showing 1 to 3 of 3 entries



Block ID: 1073741837
Block Pool ID: BP-1266388418-127.0.0.1-1560101340487
Generation Stamp: 1013
Size: 30
Availability:
- localhost

```
(Hello,1)
(World,1)
(hello,2)
```

spark:



Browse Directory

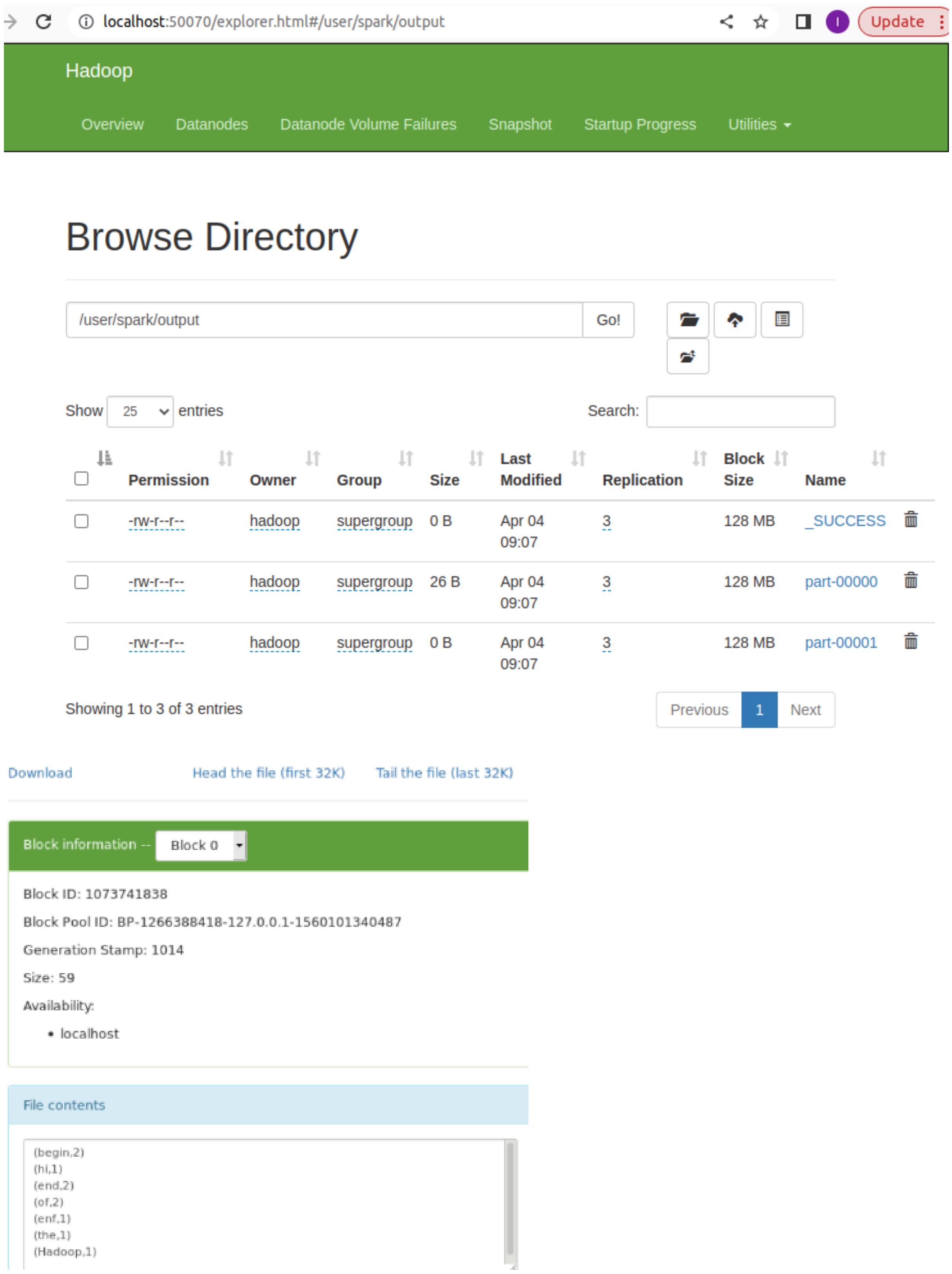
/user/spark Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:02	0	0 B	input
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:07	0	0 B	output
☐	drwxr-xr-x	hadoop	supergroup	0 B	Apr 04 09:14	0	0 B	output-wordcount

Showing 1 to 3 of 3 entries

Previous 1 Next



Browse Directory

/user/spark/output

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	hadoop	supergroup	0 B	Apr 04 09:07	3	128 MB	_SUCCESS
☐	-rw-r--r--	hadoop	supergroup	26 B	Apr 04 09:07	3	128 MB	part-00000
☐	-rw-r--r--	hadoop	supergroup	0 B	Apr 04 09:07	3	128 MB	part-00001

Showing 1 to 3 of 3 entries

Block information -- Block 0

Block ID: 1073741838
Block Pool ID: BP-1266388418-127.0.0.1-1560101340487
Generation Stamp: 1014
Size: 59
Availability:
- localhost

File contents

```
(begin,2)
(hi,1)
(end,2)
(of,2)
(enf,1)
(the,1)
(Hadoop,1)
```

Summarised learning:

In this lab, I explored the architecture and execution model of Apache Spark for big data analytics. I developed and executed a WordCount application using both REPL and HDFS, and ran it in cluster mode using standalone Spark deployment. I also created a custom MapReduce program to perform character frequency analysis, showcasing Spark's flexibility for scalable data transformations. By integrating HDFS as the data source, I implicitly understood the coordination between Spark and Hadoop components. This hands-on experience deepened my understanding of Spark's multi-language support, distributed processing, and practical use for real-world data analytics tasks.