

Tracy Michaels

Tmichaels1 002430918

BigData Assignment 3

Setting up spark:

- First I made sure everything was up to date by running commands `sudo apt-get update` and `sudo apt-get upgrade`
- Then I installed scala using `sudo apt install scala` and confirmed the installation using `scala -version`

```
tracy@Tracy-PC:~$ scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
tracy@Tracy-PC:~$
```

- I then updated the `.bashrc` to include the correct paths for scala

```
export SCALA_HOME=/home/tracy/scala
export PATH=$SCALA_HOME/bin:$PATH
```

- I then downloaded spark from apache.org using a `wget` command and a link I found on their website
- Then I unzipped the file using `tar` and changed the name of the new folder to `spark` in the file explorer of `vscode`

```
tracy@Tracy-PC:~$ echo $SCALA_HOME
https://downloads.apache.org/spark/spark-3.1.1/spark-3.1.1-bin-hadoop
2.7.tgztracy@Tracy-PC:~$ wget https://downloads.apache.org/spark/spark-3.1.1/spark-3.1.1-bin-
hadoop2.7.tgz
--2021-03-03 19:06:05-- https://downloads.apache.org/spark/spark-3.1.1/spark-3.1.1-bin-hado
op2.7.tgz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.95.219, 2a01:4f8:10a:201a::2
Connecting to downloads.apache.org (downloads.apache.org)|88.99.95.219|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 224374704 (214M) [application/x-gzip]
Saving to: 'spark-3.1.1-bin-hadoop2.7.tgz'

spark-3.1.1-bin-hadoop2. 100%[=====>] 213.98M  12.1MB/s   in 14s

2021-03-03 19:06:19 (15.6 MB/s) - 'spark-3.1.1-bin-hadoop2.7.tgz' saved [224374704/224374704
]

tracy@Tracy-PC:~$ tar -xf spark-3.1.1-bin-hadoop2.7.tgz
```

- After downloading and unzipping these files I added their path to the `.bashrc` file

```
export SPARK_HOME=/home/tracy/spark
export PATH=$SPARK_HOME/bin:$PATH
export PYSPARK_PYTHON=python3
```

- To confirm that everything is working properly I used the command pyspark to open an interactive python spark shell and tried a few command on the files for test.txt and peterman.txt

```

tracy@Tracy-PC:~$ pyspark
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
21/03/03 19:10:05 WARN Utils: Your hostname, Tracy-PC resolves to a loopback address: 127.0.
1.1; using 172.27.22.218 instead (on interface eth0)
21/03/03 19:10:05 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/home/tracy/spa
rk/jars/spark-unsafe_2.12-3.1.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Plato
rm
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access o
perations
WARNING: All illegal access operations will be denied in a future release
21/03/03 19:10:08 WARN NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

      /--
     /  \
    /    \
   /      \
  /        \
 /          \
/            \
\            /
 \          /
  \        /
   \      /
    \    /
     \  /
      --

version 3.1.1

Using Python version 3.8.5 (default, Jan 27 2021 15:41:15)
Spark context Web UI available at http://172.27.22.218:4040
Spark context available as 'sc' (master = local[*], app id = local-1614816609328).
SparkSession available as 'spark'.
>>> lines = sc.textFile("/home/tracy/bigdata/assignment3/test.txt")
>>> lines.count()
\3
>>> lines.count()
3
>>> lines.first()
'hadoop spark hadoop'
>>> book1 = sc.textFile("/home/tracy/bigdata/assignment3/peterpan.txt")
>>> book1.count()
6649
>>>

```

- These commands create an RDD called lines and book1 by passing the file path of the files to the textFile() function of spark context object

- After downloading the WordCount.py from iCollege I ran the command

```
spark-submit /home/tracy/bigdata/assignment3/WordCount.py
/home/tracy/bigdata/assignment3/test.txt 5
```

- Where the first argument is the python file that spark is going to be running, the second argument is the location of the data that the WordCount.py is going to act on and the third argument is the number of topK words that will be returned with their counts

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Displays the file structure of the project. The file `WordCount.py` is selected under the `assignment3` directory.
- WordCount.py:** The code editor shows the following Python code:


```
1 import sys
2 import pyspark
3 from pyspark.context import SparkContext
4 from pyspark import SparkConf
5
6 # Parse the input parameters
7 input_file_name = sys.argv[1]
8 number_k_for_topK = int(sys.argv[2])
9 print(input_file_name)
10 print(str(number_k_for_topK))
11
12 # Prepare the Spark context
13 conf = SparkConf().setMaster("local") \
14     .setAppName("Word Count Spark") \
15     .set("spark.executor.memory", "4g") \
16     .set("spark.executor.instances", 1)
17 sc = SparkContext(conf = conf)
18 book = sc.textFile(input_file_name)
19
20 # WordCount
```
- TERMINAL:** Shows the execution output of the `spark-submit` command. The output includes Spark logs and the final result:


```
2.27.22.218 (executor driver) (1/1)
21/03/03 19:20:41 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
21/03/03 19:20:41 INFO DAGScheduler: ResultStage 1 (runJob at PythonRDD.scala:166) finished in 0.101 s
21/03/03 19:20:41 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
21/03/03 19:20:41 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
21/03/03 19:20:41 INFO DAGScheduler: Job 0 finished: runJob at PythonRDD.scala:166, took 0.977522 s
[['hadoop', 4], ('spark', 3), ('pig', 2), ('hive', 1), ('hbase', 1)]
21/03/03 19:20:41 INFO SparkContext: Invoking stop() from shutdown hook
21/03/03 19:20:41 INFO SparkUI: Stopped Spark web UI at http://172.27.22.218:4040
21/03/03 19:20:41 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/03/03 19:20:41 INFO MemoryStore: MemoryStore cleared
21/03/03 19:20:41 INFO BlockManager: BlockManager stopped
21/03/03 19:20:41 INFO BlockManagerMaster: BlockManagerMaster stopped
21/03/03 19:20:41 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/03/03 19:20:41 INFO SparkContext: Successfully stopped SparkContext
21/03/03 19:20:41 INFO ShutdownHookManager: Shutdown hook called
21/03/03 19:20:41 INFO ShutdownHookManager: Deleting directory /tmp/spark-40dc9ddb-39ec-4de1-9c67-7897902861d9/pyspark-c900dc9b-97bd-4b9e-b41e-9c326124212d
21/03/03 19:20:41 INFO ShutdownHookManager: Deleting directory /tmp/spark-40dc9ddb-39ec-4de1-9c67-7897902861d9
21/03/03 19:20:41 INFO ShutdownHookManager: Deleting directory /tmp/spark-579db708-425b-4d01-a783-bdd1f0b93ceb
tracy@Tracy-PC:~$
```

- In this case what was returned was [('hadoop', 4), ('spark', 3), ('pig', 2), ('hive', 1), ('hbase', 1)] which are the 5 most used words in test.txt file
- I repeated this same process but for the peterpan.txt file using command

```
spark-submit /home/tracy/bigdata/assignment3/WordCount.py /home/tracy/bigdata/assignment3/peterpan.txt 30
```
- Where each argument functions the same as explained above

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Shows the file structure of the project, including directories like `bigdata` and `assignment3`, and files like `test.txt` and `WordCount.py`.
- WordCount.py:** The script content is displayed in the editor. It imports `sys` and `pyspark`, parses command-line arguments for the input file name and the number of top words (`topK`), sets up the Spark context with a local master and 4GB memory, and reads the text file to perform word counting.
- TERMINAL:** Shows the execution output of the script. It indicates that the job is finished, the Spark context is stopped, and the top 5 words are printed: `[('the', 2331), ('to', 1396), ('a', 962), ('of', 929), ('was', 898), ('he', 866), ('in', 683), ('that', 564), ('had', 498), ('it', 473), ('they', 465), ('she', 465), ('his', 455), ('you', 403), ('but', 378), ('for', 377), ('not', 375), ('with', 361), ('her', 361), ('is', 350), ('on', 329), ('at', 322), ('as', 315), ('I', 253), ('be', 249), ('have', 247), ('were', 243), ('Peter', 238)]`.

- This time though it will return the 30 most used words in the file
- Output: [('the', 2331), ('', 2259), ('and', 1396), ('to', 1214), ('a', 962), ('of', 929), ('was', 898), ('he', 866), ('in', 683), ('that', 564), ('had', 498), ('it', 473), ('they', 465), ('she', 465), ('his', 455), ('you', 403), ('but', 378), ('for', 377), ('not', 375), ('with', 361), ('her', 361), ('is', 350), ('on', 329), ('at', 322), ('as', 315), ('I', 253), ('be', 249), ('have', 247), ('were', 243), ('Peter', 238)]