

## Setting up spark on ubuntu:

1. Download Spark
  - a. Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.
  - b. Download spark packages using the command - **wget http://apache-mirror.8birdsvideo.com/spark/spark-3.0.0-preview2/spark-3.0.0-preview2-bin-hadoop3.2.tgz**

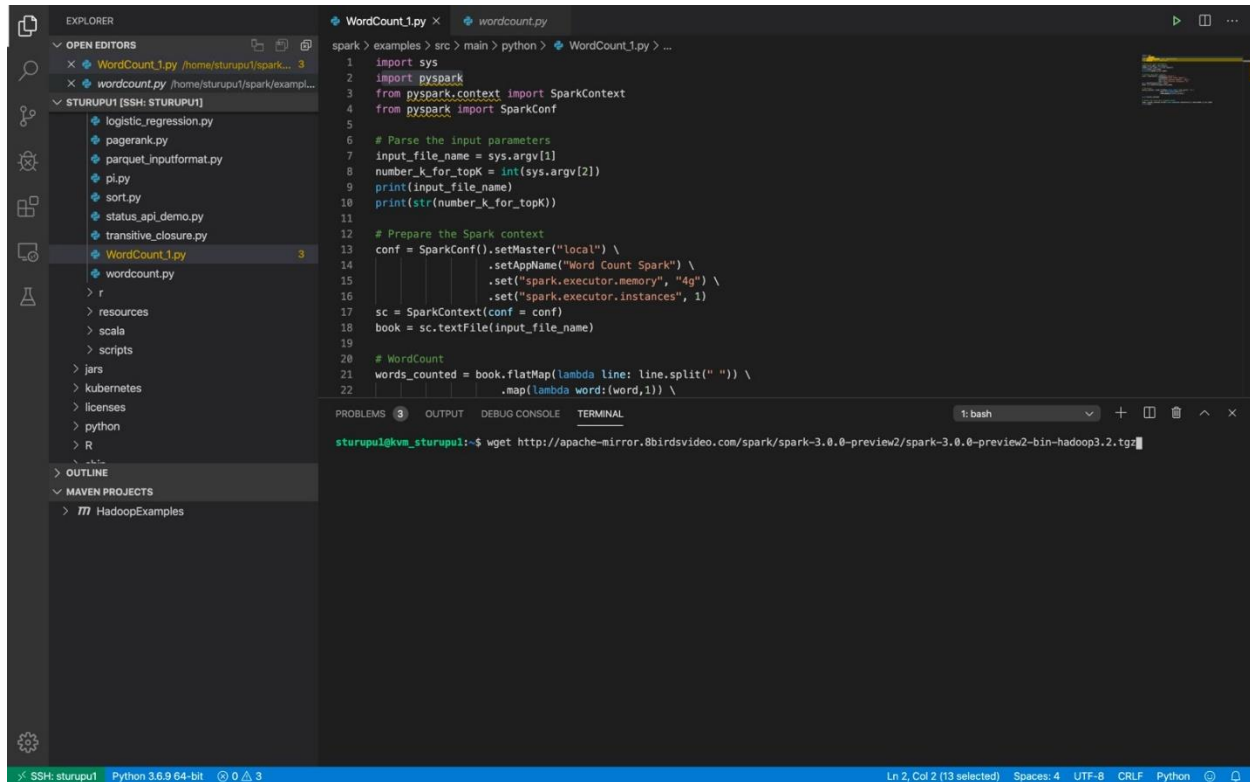
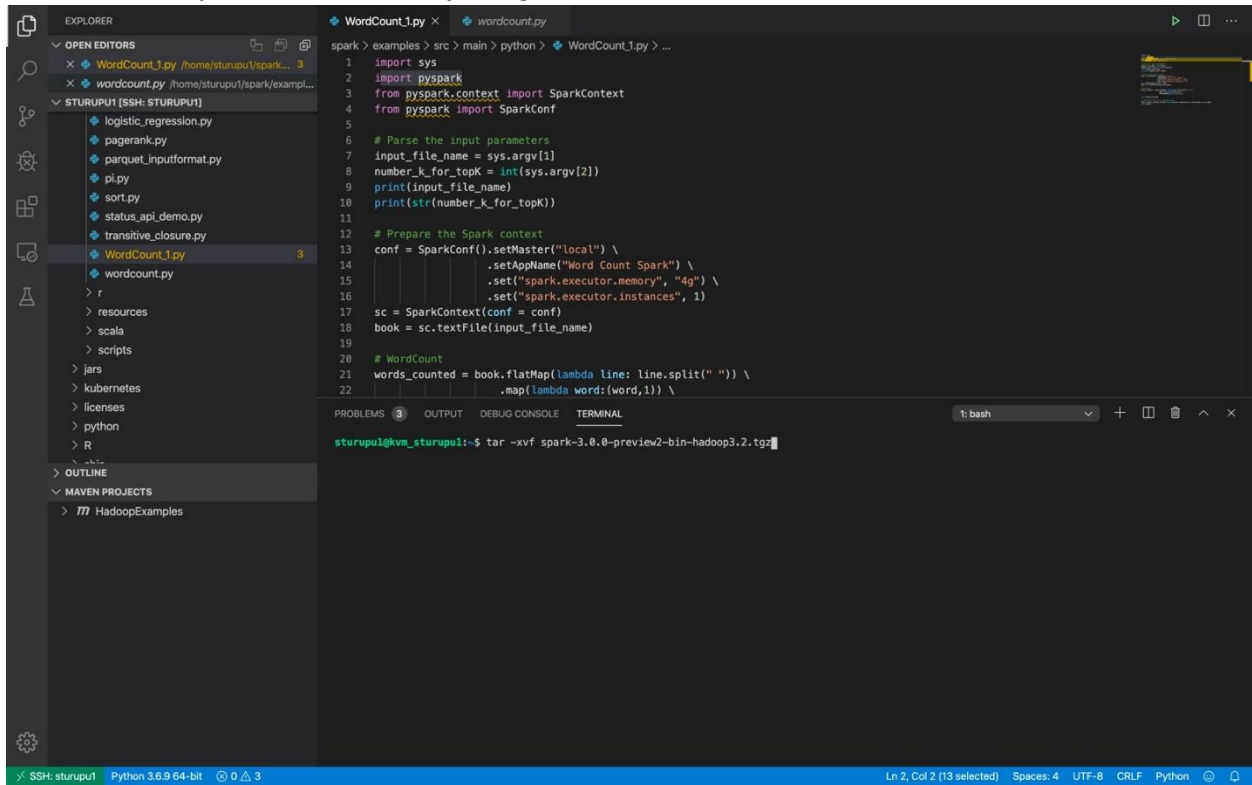


Fig 1. Downloading spark packages onto the KVM

## 2. Installing the downloaded package

- The package is in the .tgz format. After downloading we must extract the package.
- The spark packages are extracted using the following command - **tar -xvf spark-3.0.0-preview2-bin-hadoop3.2.tgz**



The screenshot shows an IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'HadoopExamples' with various Python files. The code editor displays a Python script named 'WordCount1.py' with the following content:

```
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
21 words_counted = book.flatMap(lambda line: line.split(" ")) \
22     .map(lambda word: (word, 1)) \
```

The terminal window at the bottom shows the command being executed:

```
sturupu1@kvm_sturupu1:~$ tar -xvf spark-3.0.0-preview2-bin-hadoop3.2.tgz
```

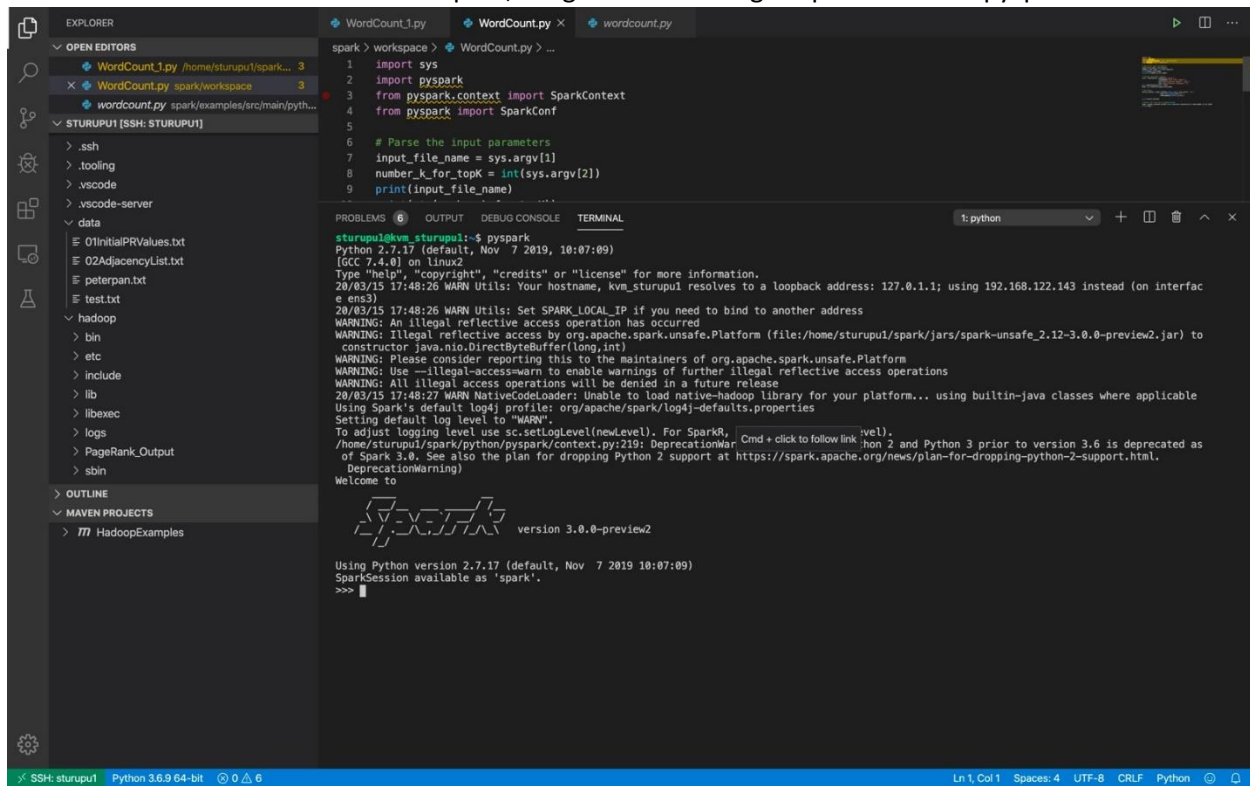
Fig 2. Extracting the Spark packages

- `export SPARK_HOME=/home/sturupu1/spark`
- `export PATH=$SPARK_HOME/bin:$PATH`
- `export PYSARK_PYTHON=python3`

changes immediately.

Fig 3. Setting the path of spark and python in bashrc

4. On successful installation of spark, we get the following output if we run “pyspark” command



The screenshot shows a VS Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'STURUPU1' with a file 'wordcount.py' open. The terminal shows the output of the 'pyspark' command, which includes warnings about illegal reflective access and a deprecation warning for Python 2. The Spark logo and version '3.0.0-preview2' are displayed in the terminal output.

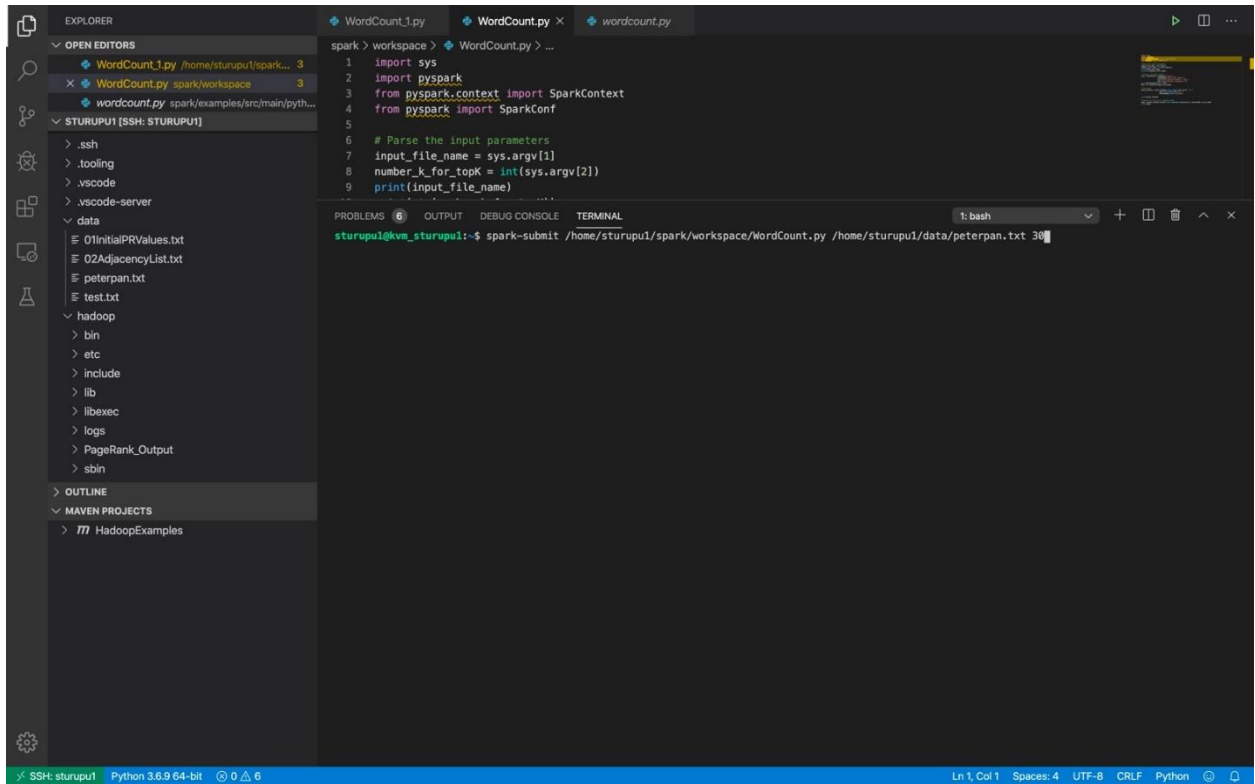
```
spark > workspace > WordCount.py > ...
1 import sys
2 import pyspark
3 from pyspark.context import SparkContext
4 from pyspark import SparkConf
5
6 # Parse the input parameters
7 input_file_name = sys.argv[1]
8 number_k_for_topK = int(sys.argv[2])
9 print(input_file_name)
```

```
sturupul@kvm_sturupul1:~$ pyspark
Python 2.7.17 (default, Nov 7 2019, 10:07:09)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license()" for more information.
20/03/15 17:48:26 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/sturupul/spark/jars/spark-unsafe_2.12-3.0.0-preview2.jar) to
 constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/03/15 17:48:27 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... 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).
/home/sturupul/spark/python/pyspark/context.py:219: DeprecationWarning: Python 2 and Python 3 prior to version 3.6 is deprecated as
 of Spark 3.0. See also the plan for dropping Python 2 support at https://spark.apache.org/news/plan-for-dropping-python-2-support.html.
DeprecationWarning)
Welcome to
Spark
version 3.0.0-preview2

Using Python version 2.7.17 (default, Nov 7 2019 10:07:09)
SparkSession available as 'spark'.
>>>
```

Fig 4. On Successful installation of spark, the following output is displayed

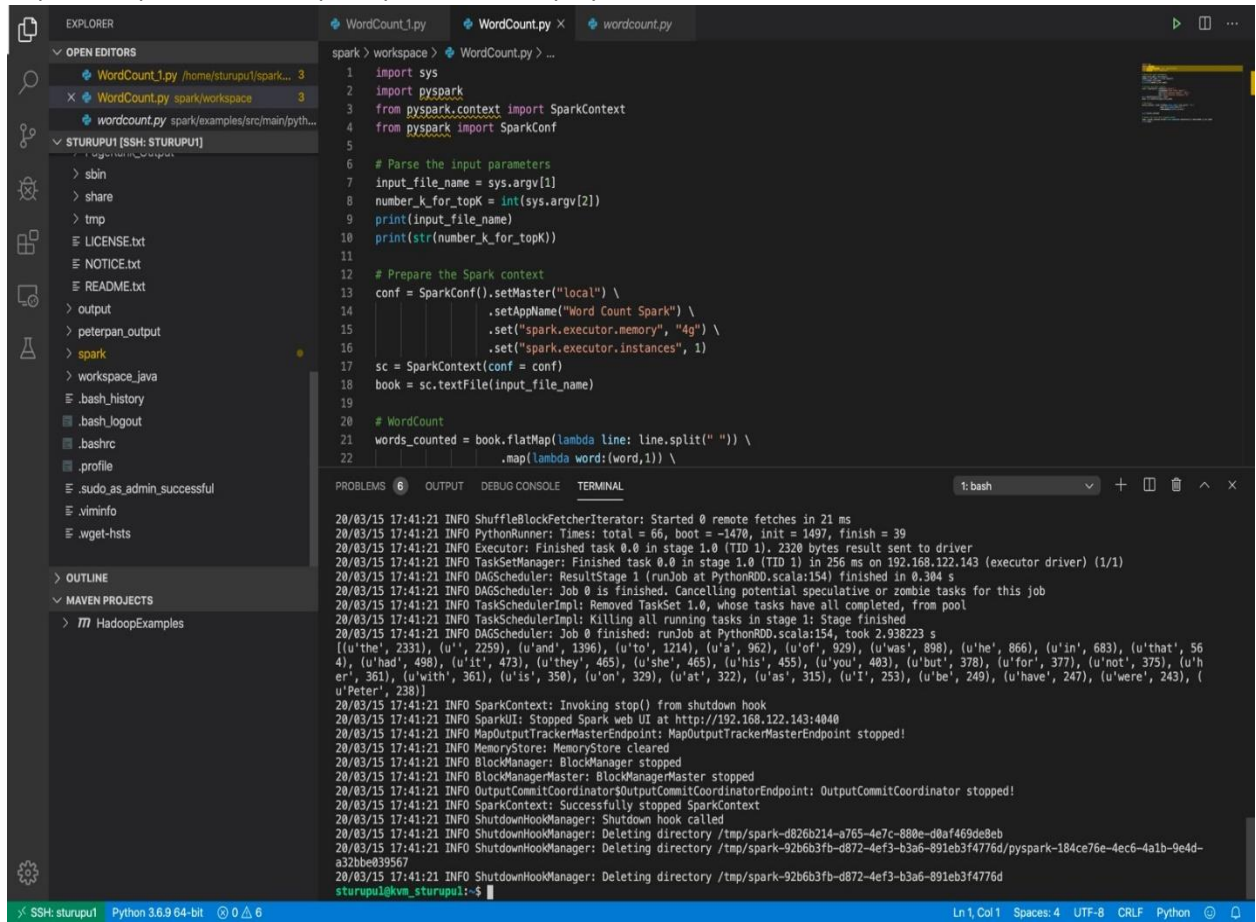
5. Once we have all the input files (peterpan.txt, test.txt) and the python program for wordcount in place, we run the following command - `spark submit /home/sturupu1/spark/workspace/WordCount.py /home/sturupu1/hadoop/data/peterpan.txt 30`
- a. The last parameter gives the topmost frequent word occurrences.



The screenshot shows a VS Code editor interface with a dark theme. The Explorer sidebar on the left displays the file structure of a project named 'HadoopExamples', including files like '01InitialPRValues.txt', '02AdjacencyList.txt', 'peterpan.txt', and 'test.txt'. The main editor area has three tabs open: 'WordCount\_1.py', 'WordCount.py', and 'wordcount.py'. The 'WordCount.py' tab is active, showing a Python script that imports 'sys' and 'pyspark', and uses 'SparkContext' and 'SparkConf' to parse input parameters. The script takes an input file name and a topK value from command-line arguments. Below the editor, the TERMINAL panel is open, showing a bash shell prompt where the command `spark-submit /home/sturupu1/spark/workspace/WordCount.py /home/sturupu1/data/peterpan.txt 30` has been entered. The status bar at the bottom indicates the editor is connected to 'SSH: sturupu1' and is running 'Python 3.6.9 64-bit'.

Fig 5. Command to run the word count on peterpan.txt

6. Top 30 frequent words of peterpan.txt are displayed as follows:

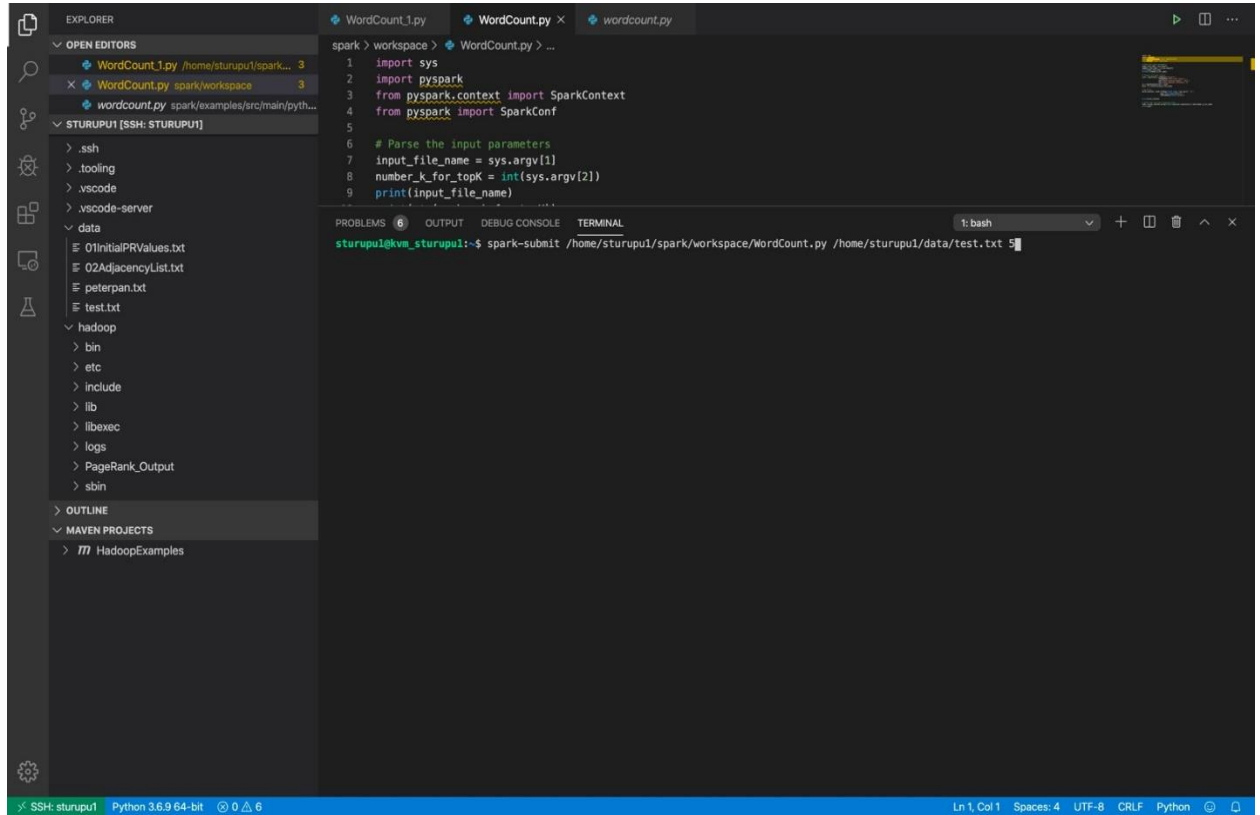


```
1 import sys
2 import pyspark
3 from pyspark.context import SparkContext
4 from pyspark import SparkConf
5
6 # Parse the input parameters
7 input_file_name = sys.argv[1]
8 number_k_for_topK = int(sys.argv[2])
9 print(input_file_name)
10 print(str(number_k_for_topK))
11
12 # Prepare the Spark context
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
21 words_counted = book.flatMap(lambda line: line.split(" ")) \
22     .map(lambda word: (word, 1)) \
```

20/03/15 17:41:21 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 21 ms  
20/03/15 17:41:21 INFO PythonRunner: Times: total = 66, boot = -1470, init = 1497, finish = 39  
20/03/15 17:41:21 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1). 2320 bytes result sent to driver  
20/03/15 17:41:21 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 256 ms on 192.168.122.143 (executor driver) (1/1)  
20/03/15 17:41:21 INFO DAGScheduler: ResultStage 1 (runJob at PythonRDD.scala:154) finished in 0.304 s  
20/03/15 17:41:21 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job  
20/03/15 17:41:21 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool  
20/03/15 17:41:21 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished  
20/03/15 17:41:21 INFO DAGScheduler: Job 0 finished: runJob at PythonRDD.scala:154, took 2.938223 s  
[(u'the', 2331), (u'', 2259), (u'and', 1396), (u'to', 1214), (u'a', 962), (u'of', 929), (u'was', 898), (u'he', 866), (u'in', 683), (u'that', 564), (u'had', 498), (u'it', 473), (u'they', 465), (u'she', 465), (u'his', 455), (u'you', 403), (u'but', 378), (u'for', 377), (u'not', 375), (u'her', 361), (u'with', 361), (u'is', 350), (u'on', 329), (u'at', 322), (u'as', 315), (u'I', 253), (u'be', 249), (u'have', 247), (u'were', 243), (u'Peter', 238)]  
20/03/15 17:41:21 INFO SparkContext: Invoking stop() from shutdown hook  
20/03/15 17:41:21 INFO SparkUI: Stopped Spark web UI at http://192.168.122.143:4040  
20/03/15 17:41:21 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!  
20/03/15 17:41:21 INFO MemoryStore: MemoryStore cleared  
20/03/15 17:41:21 INFO BlockManager: BlockManager stopped  
20/03/15 17:41:21 INFO BlockManagerMaster: BlockManagerMaster stopped  
20/03/15 17:41:21 INFO OutputCommitCoordinator\$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!  
20/03/15 17:41:21 INFO SparkContext: Successfully stopped SparkContext  
20/03/15 17:41:21 INFO ShutdownHookManager: Shutdown hook called  
20/03/15 17:41:21 INFO ShutdownHookManager: Deleting directory /tmp/spark-d826b214-a765-4e7c-880e-d8af469de8eb  
20/03/15 17:41:21 INFO ShutdownHookManager: Deleting directory /tmp/spark-92b6b3fb-d872-4ef3-b3a6-891eb3f4776d/pyspark-184ce76e-4ec6-4a1b-9e4d-a32be839567  
20/03/15 17:41:21 INFO ShutdownHookManager: Deleting directory /tmp/spark-92b6b3fb-d872-4ef3-b3a6-891eb3f4776d

Fig 6. Output of word counts of top 30 words of peterpan.txt

## 7. Running top 5 frequent words for test.txt



The screenshot shows a VS Code editor interface with a dark theme. The Explorer sidebar on the left displays a file tree for a project named 'STURUPU1 [SSH: STURUPU1]'. The file tree includes a 'data' directory with files like '01InitialPRValues.txt', '02AdjacencyList.txt', 'peterpan.txt', and 'test.txt'. The main editor area shows a Python file named 'WordCount.py' with the following 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)
```

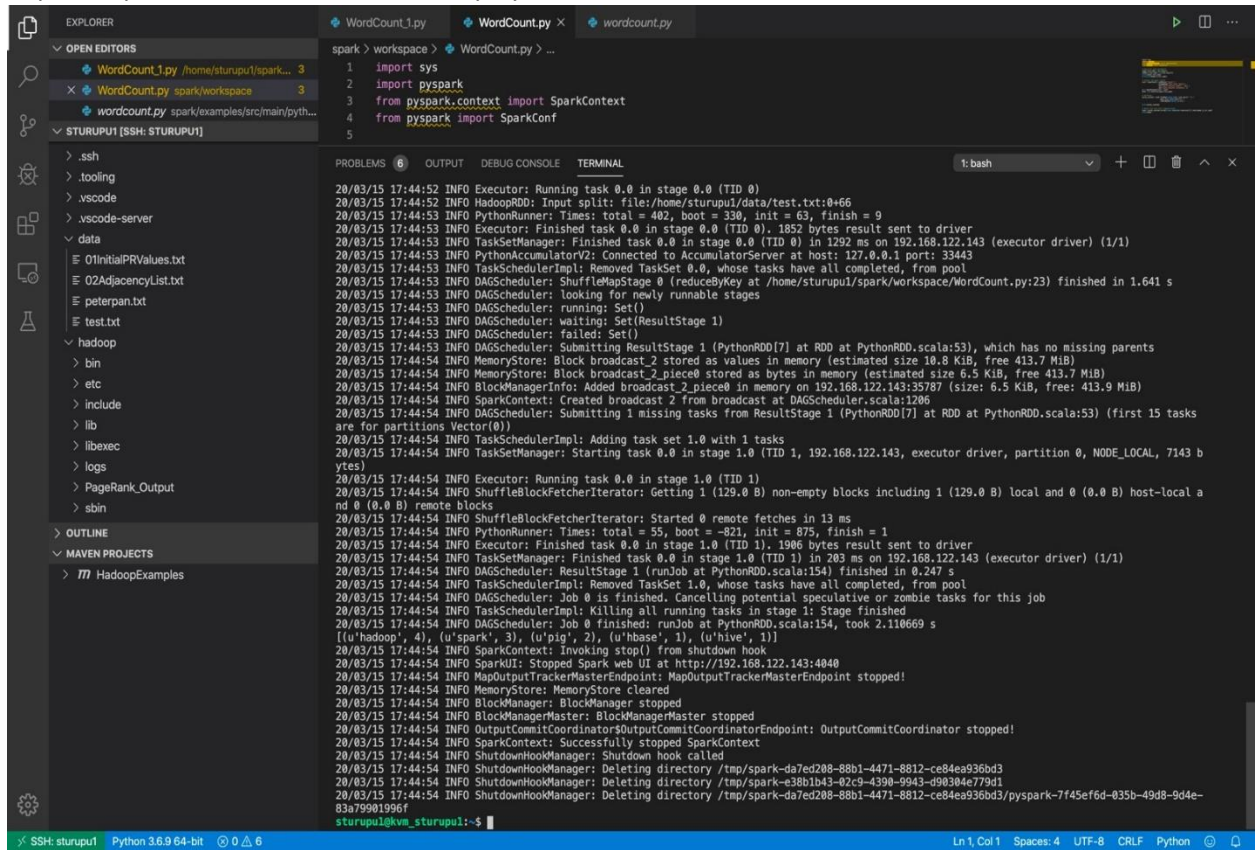
Below the code editor, the TERMINAL panel is active, showing a bash shell prompt. The command entered is:

```
sturupu1@kvm_sturupu1:~$ spark-submit /home/sturupu1/spark/workspace/WordCount.py /home/sturupu1/data/test.txt 5
```

The status bar at the bottom indicates the current file is 'Ln 1, Col 1' with 'Spaces: 4', 'UTF-8' encoding, 'CRLF' line endings, and 'Python' language.

Fig 7. Command to run the word count on test.txt

8. Top 5 frequent words of test.txt are displayed as follows:



```
spark > workspace > WordCount.py > ...
1 import sys
2 import pyspark
3 from pyspark.context import SparkContext
4 from pyspark import SparkConf
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
20/03/15 17:44:52 INFO Executor: Running task 0.0 in stage 0.0 (TID 0)
20/03/15 17:44:52 INFO HadoopRDD: Input split: file:/home/sturupul/data/test.txt:0+66
20/03/15 17:44:53 INFO PythonRunner: Times: total = 402, boot = 330, init = 63, finish = 9
20/03/15 17:44:53 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0), 1852 bytes result sent to driver
20/03/15 17:44:53 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1292 ms on 192.168.122.143 (executor driver) (1/1)
20/03/15 17:44:53 INFO PythonAccumulatorV2: Connected to AccumulatorServer at host: 127.0.0.1 port: 33443
20/03/15 17:44:53 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
20/03/15 17:44:53 INFO DAGScheduler: ShuffleMapStage 0 (reduceByKey at /home/sturupul/spark/workspace/WordCount.py:23) finished in 1.641 s
20/03/15 17:44:53 INFO DAGScheduler: looking for newly runnable stages
20/03/15 17:44:53 INFO DAGScheduler: running: Set()
20/03/15 17:44:53 INFO DAGScheduler: waiting: Set(ResultStage 1)
20/03/15 17:44:53 INFO DAGScheduler: failed: Set()
20/03/15 17:44:53 INFO DAGScheduler: Submitting ResultStage 1 (PythonRDD[7] at RDD at PythonRDD.scala:53), which has no missing parents
20/03/15 17:44:54 INFO MemoryStore: Block broadcast_2 stored as values in memory (estimated size 10.8 KiB, free 413.7 MiB)
20/03/15 17:44:54 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on 192.168.122.143:35787 (size: 6.5 KiB, free: 413.9 MiB)
20/03/15 17:44:54 INFO SparkContext: Created broadcast 2 from broadcast at DAGScheduler.scala:1206
20/03/15 17:44:54 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 1 (PythonRDD[7] at RDD at PythonRDD.scala:53) (first 15 tasks are for partitions Vector(0))
20/03/15 17:44:54 INFO TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
20/03/15 17:44:54 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, 192.168.122.143, executor driver, partition 0, NODE_LOCAL, 7143 bytes)
20/03/15 17:44:54 INFO Executor: Running task 0.0 in stage 1.0 (TID 1)
20/03/15 17:44:54 INFO ShuffleBlockFetcherIterator: Getting 1 (129.0 B) non-empty blocks including 1 (129.0 B) local and 0 (0.0 B) host-local and 0 (0.0 B) remote blocks
20/03/15 17:44:54 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 13 ms
20/03/15 17:44:54 INFO PythonRunner: Times: total = 55, boot = -821, init = 875, finish = 1
20/03/15 17:44:54 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1), 1906 bytes result sent to driver
20/03/15 17:44:54 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 203 ms on 192.168.122.143 (executor driver) (1/1)
20/03/15 17:44:54 INFO DAGScheduler: ResultStage 1 (runJob at PythonRDD.scala:154) finished in 0.247 s
20/03/15 17:44:54 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
20/03/15 17:44:54 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
20/03/15 17:44:54 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
20/03/15 17:44:54 INFO DAGScheduler: Job 0 finished: runJob at PythonRDD.scala:154, took 2.110669 s
[[('hadoop', 4), ('spark', 3), ('pig', 2), ('base', 1), ('hive', 1)]]
20/03/15 17:44:54 INFO SparkContext: Invoking stop() from shutdown hook
20/03/15 17:44:54 INFO SparkUI: Stopped Spark web UI at http://192.168.122.143:4040
20/03/15 17:44:54 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
20/03/15 17:44:54 INFO MemoryStore: MemoryStore cleared
20/03/15 17:44:54 INFO BlockManager: BlockManager stopped
20/03/15 17:44:54 INFO BlockManagerMaster: BlockManagerMaster stopped
20/03/15 17:44:54 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
20/03/15 17:44:54 INFO SparkContext: Successfully stopped SparkContext
20/03/15 17:44:54 INFO ShutdownHookManager: Shutdown hook called
20/03/15 17:44:54 INFO ShutdownHookManager: Deleting directory /tmp/spark-da7ed288-88b1-4471-8812-ce84e936bd3
20/03/15 17:44:54 INFO ShutdownHookManager: Deleting directory /tmp/spark-e38b1b43-82c9-4390-9943-d90304e779d1
20/03/15 17:44:54 INFO ShutdownHookManager: Deleting directory /tmp/spark-da7ed288-88b1-4471-8812-ce84e936bd3/pyspark-7f45ef6d-835b-49d8-9d4e-83a79901996f
sturupul@kvm_sturupul1:~$
```

Fig 8. Output of word counts of top 5 words of test.txt