

## Setting up spark on ubuntu:

### 1. Download Spark

- 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.
- 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>**

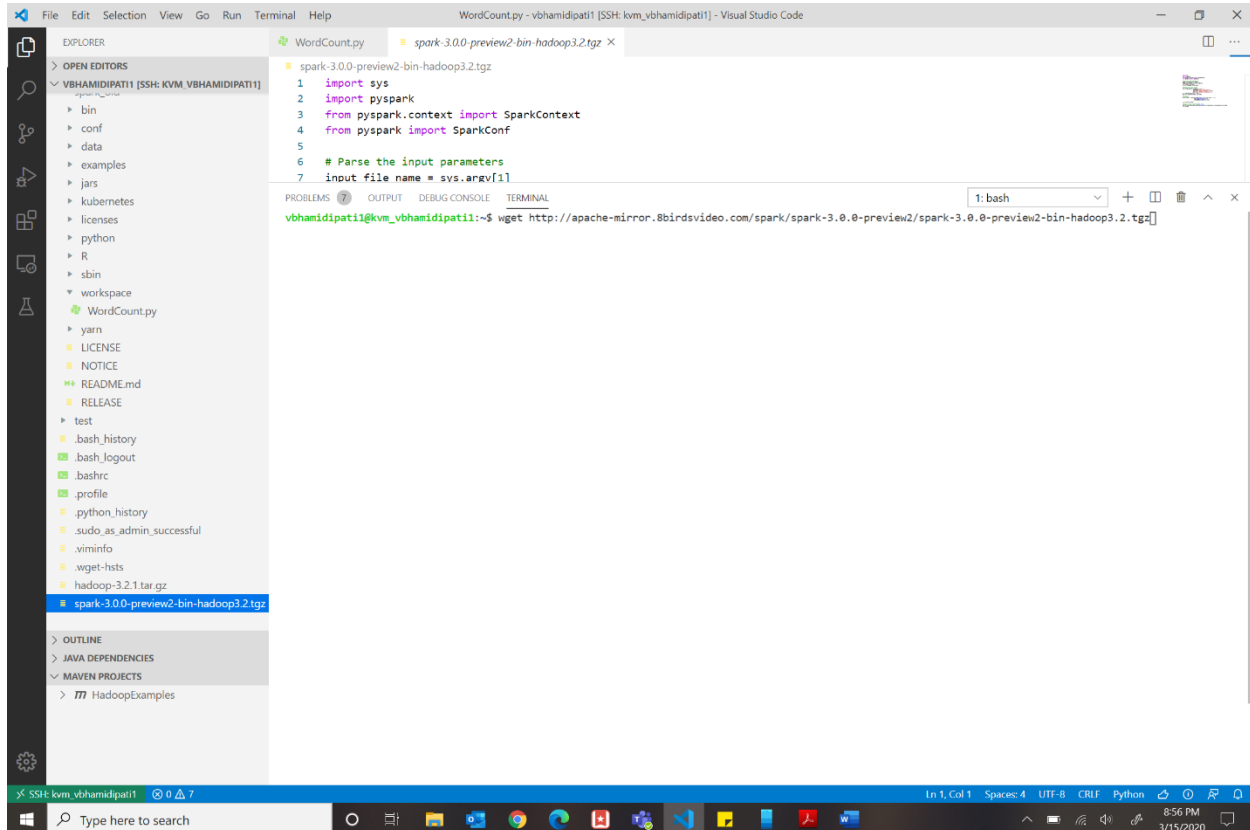


Fig 1. Downloading spark packages onto the KVM

## 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**

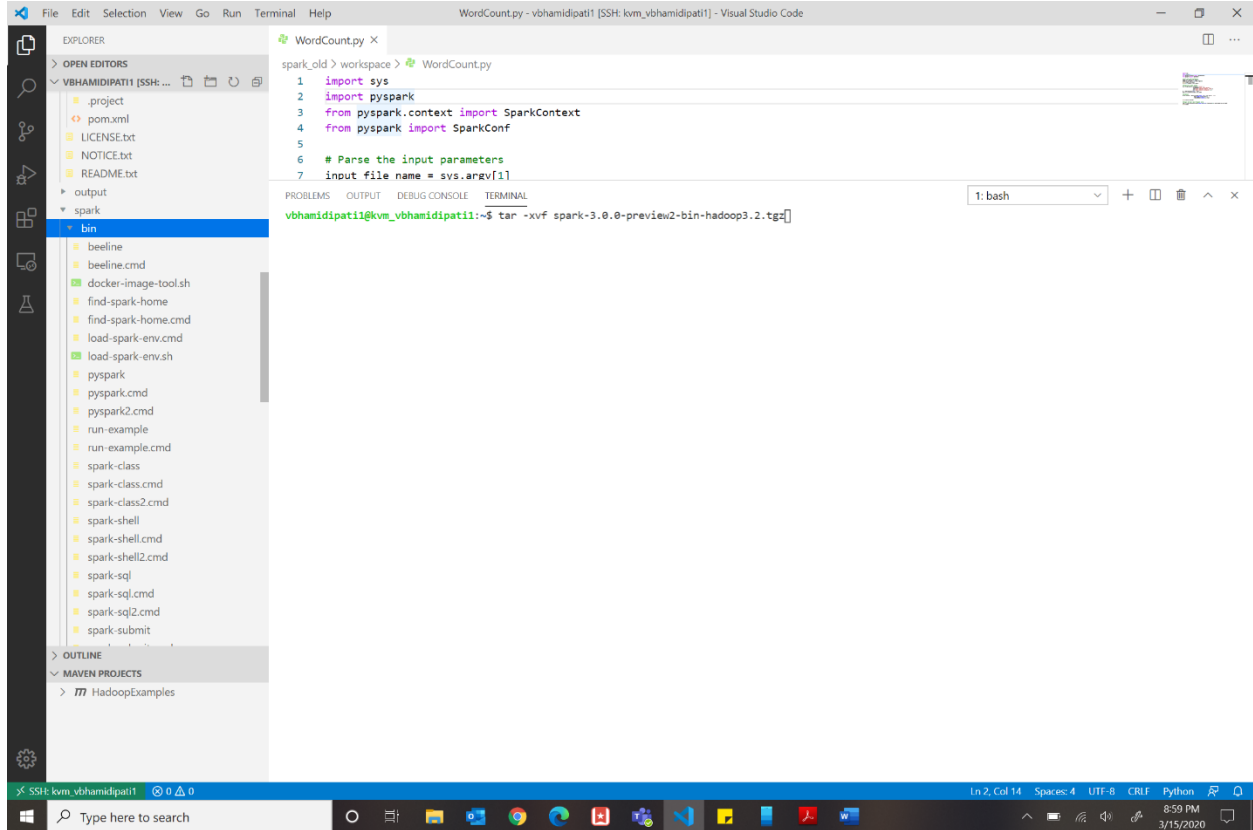
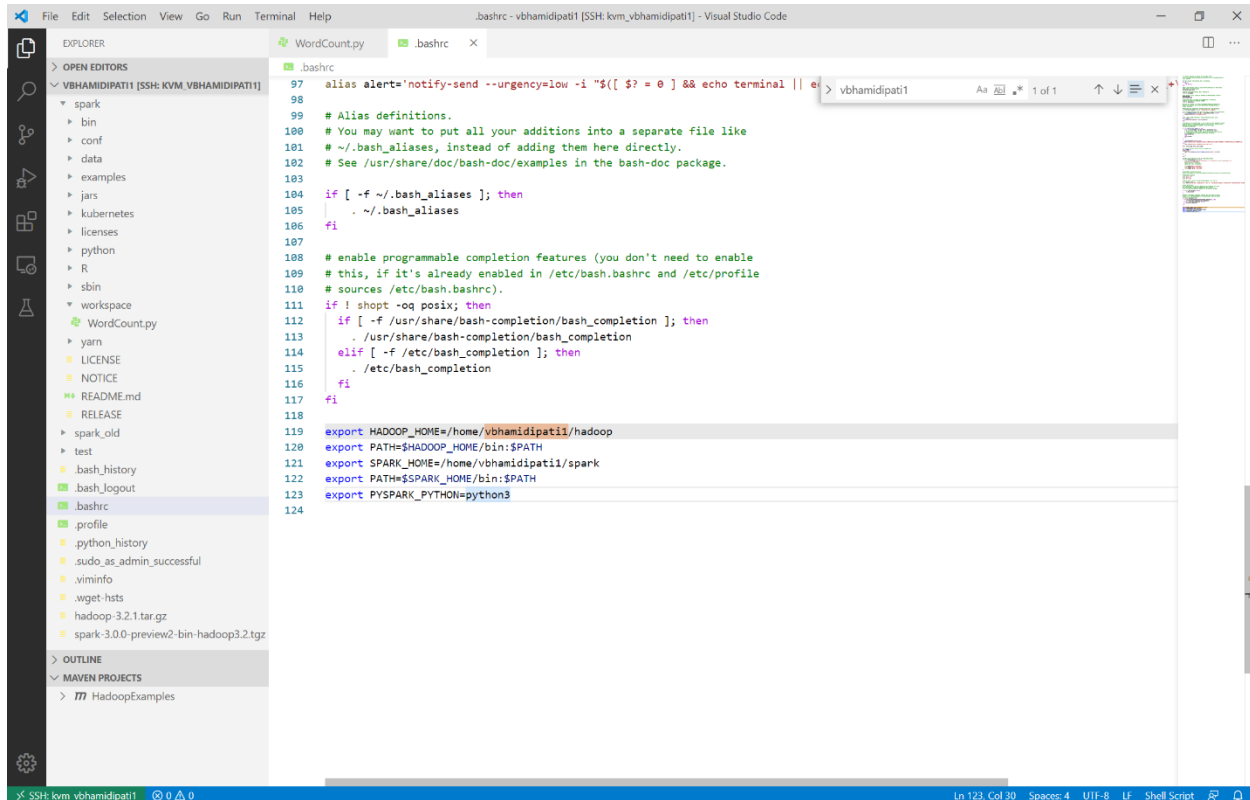


Fig 2. Extracting the Spark packages

### 3. Setup the path for spark and python

- `export SPARK_HOME=/home/vbhamidipati1/spark`
- `export PATH=$SPARK_HOME/bin:$PATH`
- `export PYSPARK_PYTHON=python3`
- Enter all the above statements in the bashrc and use `source ~/.bashrc` to save the changes immediately.



The screenshot shows the Visual Studio Code editor with the `.bashrc` file open. The file contains various shell configurations, including alias definitions, completion features, and environment variable exports. The following lines are highlighted in the image:

```
97 alias alert='notify-send --urgency=low -i "[ $? = 0 ]" && echo terminal || e
98
99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112     if [ -f /usr/share/bash-completion/bash_completion ]; then
113         . /usr/share/bash-completion/bash_completion
114     elif [ -f /etc/bash_completion ]; then
115         . /etc/bash_completion
116     fi
117 fi
118
119 export HADOOP_HOME=/home/vbhamidipati1/hadoop
120 export PATH=$HADOOP_HOME/bin:$PATH
121 export SPARK_HOME=/home/vbhamidipati1/spark
122 export PATH=$SPARK_HOME/bin:$PATH
123 export PYSPARK_PYTHON=python3
124
```

The Explorer sidebar on the left shows the file structure of the project, including directories like `spark`, `bin`, `conf`, `data`, `examples`, `jars`, `kubernetes`, `licenses`, `python`, `R`, `sbin`, `workspace`, and files like `WordCount.py`, `yarn`, `LICENSE`, `NOTICE`, `README.md`, `RELEASE`, `spark_old`, `test`, `.bash_history`, `.bash_logout`, `.bashrc`, `.profile`, `.python_history`, `.sudo_as_admin_successful`, `.viminfo`, `.wget-hsts`, `hadoop-3.2.1.tar.gz`, and `spark-3.0.0-preview2-bin-hadoop3.2.tgz`. The Outline sidebar on the right shows the Maven projects, including `HadoopExamples`.

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 displays the Visual Studio Code interface with a Python script named `WordCount.py` open in the editor. The script imports `pyspark` and `SparkContext`, then processes command-line arguments to count the number of 'topk' entries in a file. The file explorer on the left shows the project structure, including a `workspace` folder and a `WordCount.py` file. The terminal window at the bottom shows the execution of the script using `pyspark`, displaying various warnings and the final output: `Using Python version 3.6.9 (default, Nov 7 2019 10:44:02) SparkSession available as 'spark'.`

```

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[*]").

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

vbhamidipati@kvm_vbhamidipati1:~$ pyspark
Python 3.6.9 (default, Nov 7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
20/03/15 15:22:01 WARN Utils: Your hostname, kvm_vbhamidipati1 resolves to a loopback address: 127.0.1.1; using 192.168.122.108 instead (on interface ens3)
20/03/15 15:22:01 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/vbhamidipati1/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 15:22:02 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).
Welcome to

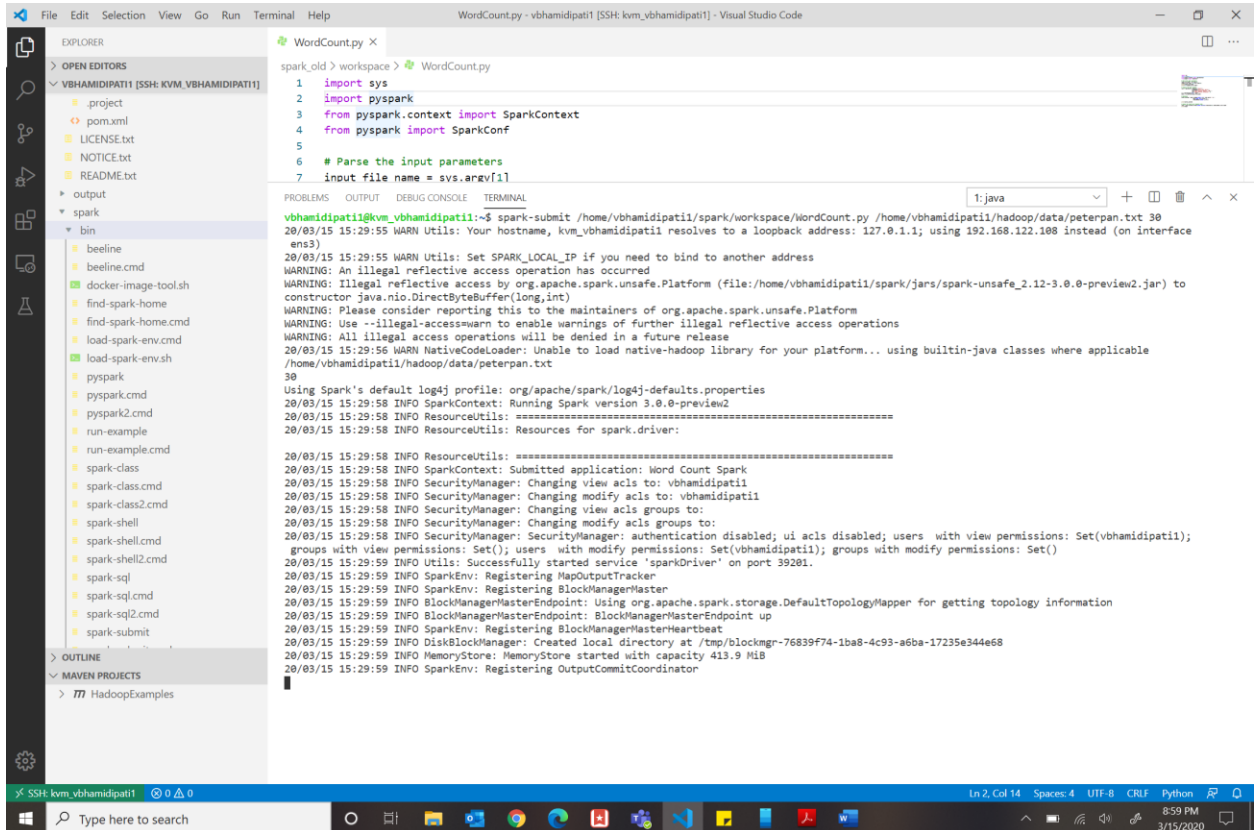
      _/  _/_   _/  _/_   _/
     /_/_/  \_/  \_/  \_/  \_/
    version 3.0.0-preview2

Using Python version 3.6.9 (default, Nov 7 2019 10:44:02)
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/vbhamidipati1/spark/workspace/WordCount.py /home/vbhamidipati1/hadoop/data/peterpan.txt 30`
- a. The last parameter gives the topmost frequent word occurrences.



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

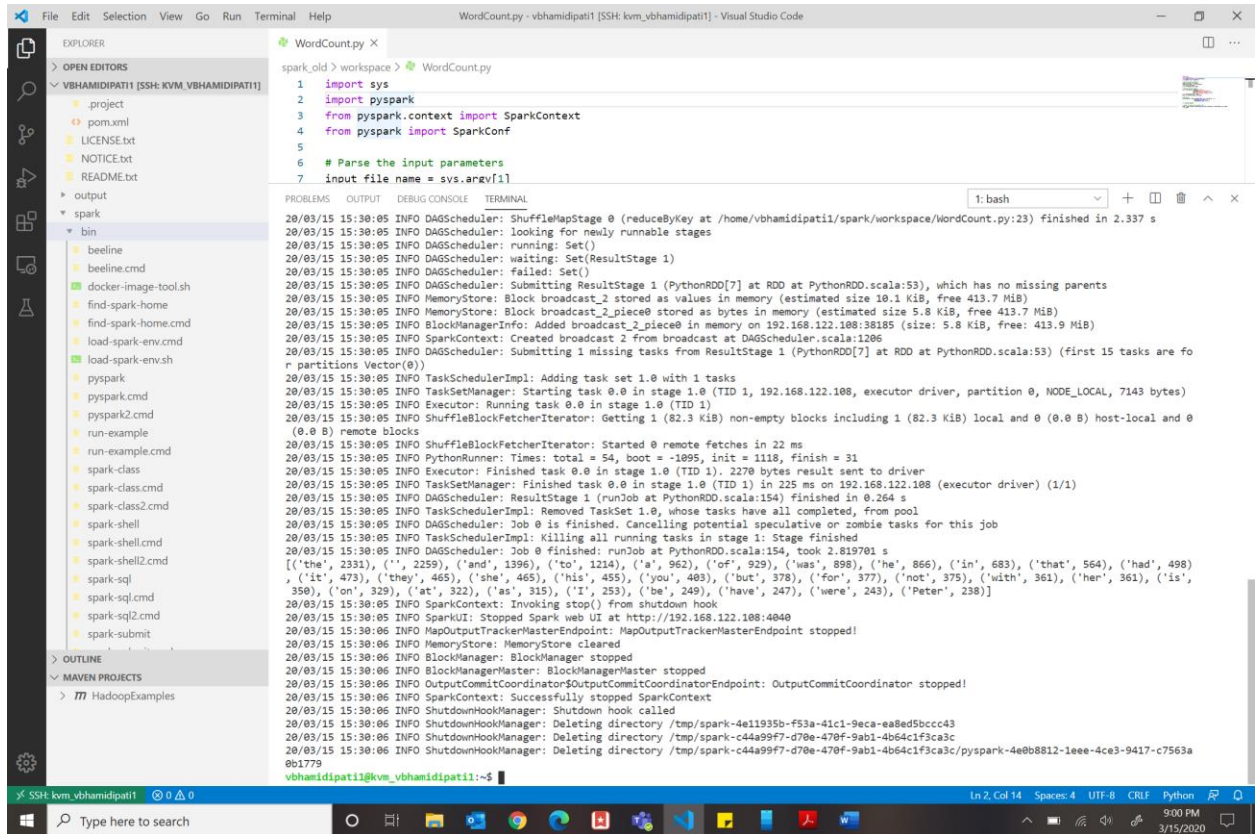
- EXPLORER:** Displays the file structure of the project, including `project`, `bin`, `spark`, and `output` folders.
- WordCount.py:** The script content is as follows:

```
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]
```
- TERMINAL:** Shows the command execution and its output:

```
vbhamidipati1@kvm_vbhamidipati1:~$ spark-submit /home/vbhamidipati1/spark/workspace/WordCount.py /home/vbhamidipati1/hadoop/data/peterpan.txt 30
20/03/15 15:29:55 WARN Utils: Your hostname, kvm_vbhamidipati1 resolves to a loopback address: 127.0.1.1; using 192.168.122.108 instead (on interface ens3)
20/03/15 15:29:55 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/vbhamidipati1/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 15:29:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/vbhamidipati1/hadoop/data/peterpan.txt
30
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
20/03/15 15:29:58 INFO SparkContext: Submitted application: Word Count Spark
20/03/15 15:29:58 INFO SecurityManager: Changing view acls to: vbhamidipati1
20/03/15 15:29:58 INFO SecurityManager: Changing modify acls to: vbhamidipati1
20/03/15 15:29:58 INFO SecurityManager: Changing view acls groups to:
20/03/15 15:29:58 INFO SecurityManager: Changing modify acls groups to:
20/03/15 15:29:58 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(vbhamidipati1); groups with view permissions: Set(); users with modify permissions: Set(vbhamidipati1); groups with modify permissions: Set()
20/03/15 15:29:59 INFO Utils: Successfully started service 'sparkDriver' on port 39201.
20/03/15 15:29:59 INFO SparkEnv: Registering MapOutputTracker
20/03/15 15:29:59 INFO SparkEnv: Registering BlockManagerMaster
20/03/15 15:29:59 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
20/03/15 15:29:59 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
20/03/15 15:29:59 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
20/03/15 15:29:59 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-76839f74-1ba8-4c93-a6ba-17235e344e68
20/03/15 15:29:59 INFO MemoryStore: MemoryStore started with capacity 418.9 MiB
20/03/15 15:29:59 INFO SparkEnv: Registering OutputCommitCoordinator
```

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

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



The screenshot shows a Visual Studio Code editor with a file named `WordCount.py` open. The file contains a Python script that uses `pyspark` to perform a word count on `peterpan.txt`. The script is as follows:

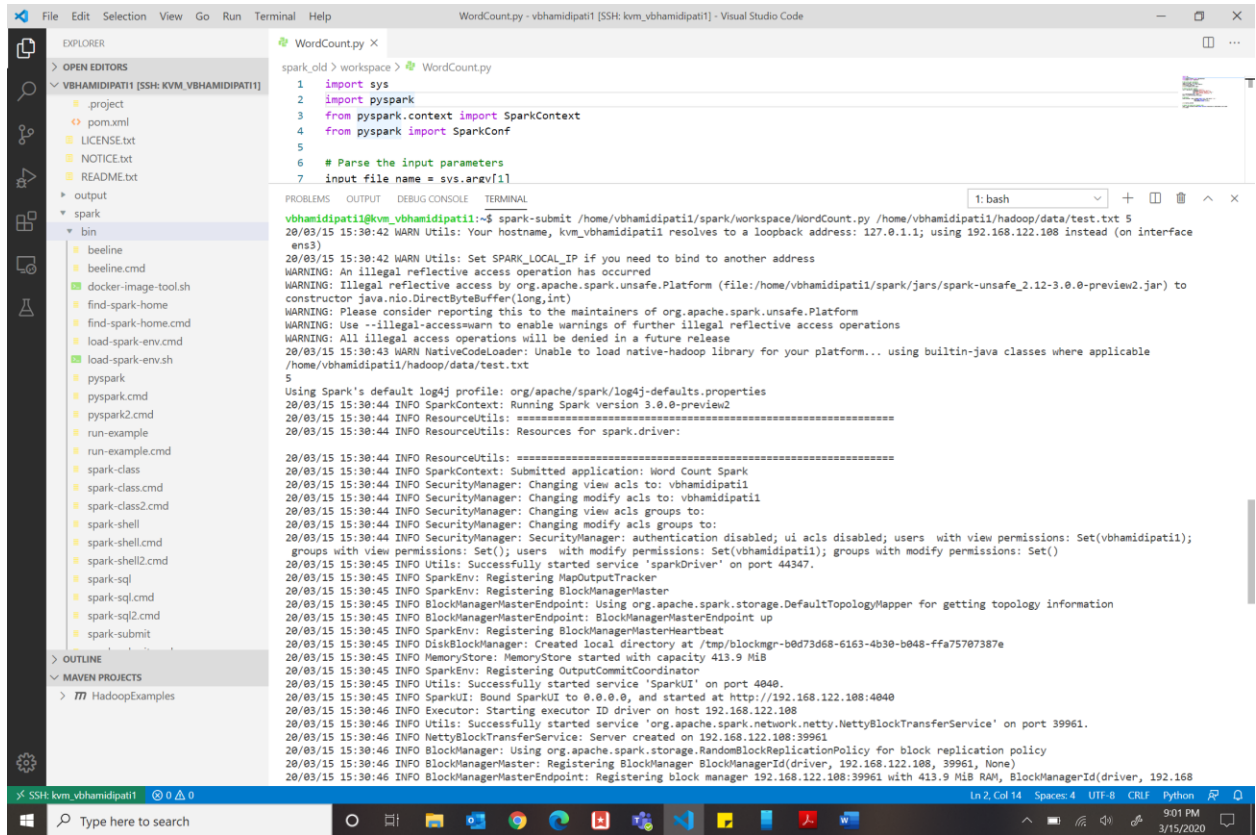
```
1 import sys
2 import pyspark
3 from pyspark.context import SparkContext
4 from pyspark import SparkConf
5
6 # Parse the input parameters
7 input_file_name = sys.argv[1]
```

The terminal window at the bottom shows the output of the script. It displays various Spark logs, including the execution of `ShuffleMapStage`, `Block broadcast_2`, and the final word count results. The top 30 frequent words are listed in a single line, each followed by its frequency in parentheses. The words are: 'the', 'a', 'and', 'to', 'of', 'was', 'he', 'in', 'that', 'had', 'it', 'they', 'she', 'his', 'you', 'but', 'for', 'not', 'with', 'her', 'is', 'on', 'at', 'as', 'I', 'me', 'be', 'have', 'were', 'Peter', and 'Pete'.

```
28/03/15 15:30:05 INFO DAGScheduler: ShuffleMapStage 0 (reduceByKey at /home/vbhamidipati1/spark/workspace/WordCount.py:23) finished in 2.337 s
28/03/15 15:30:05 INFO DAGScheduler: looking for newly runnable stages
28/03/15 15:30:05 INFO DAGScheduler: running: Set()
28/03/15 15:30:05 INFO DAGScheduler: waiting: Set(ResultStage 1)
28/03/15 15:30:05 INFO DAGScheduler: failed: Set()
28/03/15 15:30:05 INFO DAGScheduler: Submitting ResultStage 1 (PythonRDD[7] at RDD at PythonRDD.scala:53), which has no missing parents
28/03/15 15:30:05 INFO MemoryStore: Block broadcast_2 stored as values in memory (estimated size 10.1 KiB, free 413.7 MiB)
28/03/15 15:30:05 INFO MemoryStore: Block broadcast_2_piece0 stored as bytes in memory (estimated size 5.8 KiB, free 413.7 MiB)
28/03/15 15:30:05 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on 192.168.122.108:38185 (size: 5.8 KiB, free: 413.9 MiB)
28/03/15 15:30:05 INFO SparkContext: Created broadcast 2 from broadcast at DAGScheduler.scala:1206
28/03/15 15:30:05 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))
28/03/15 15:30:05 INFO TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
28/03/15 15:30:05 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, 192.168.122.108, executor driver, partition 0, NODE_LOCAL, 7143 bytes)
28/03/15 15:30:05 INFO Executor: Running task 0.0 in stage 1.0 (TID 1)
28/03/15 15:30:05 INFO ShuffleBlockFetcherIterator: Getting 1 (82.3 KiB) non-empty blocks including 1 (82.3 KiB) local and 0 (0.0 KiB) host-local and 0 (0.0 KiB) remote blocks
28/03/15 15:30:05 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 22 ms
28/03/15 15:30:05 INFO PythonRunner: Times: total = 54, boot = -1095, init = 1118, finish = 31
28/03/15 15:30:05 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1), 2270 bytes result sent to driver
28/03/15 15:30:05 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 225 ms on 192.168.122.108 (executor driver) (1/1)
28/03/15 15:30:05 INFO DAGScheduler: ResultStage 1 (runJob at PythonRDD.scala:154) finished in 0.264 s
28/03/15 15:30:05 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
28/03/15 15:30:05 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
28/03/15 15:30:05 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
28/03/15 15:30:05 INFO DAGScheduler: Job 0 finished: runJob at PythonRDD.scala:154, took 2.819701 s
[('the', 2331), ('a', 2259), ('and', 1396), ('to', 1214), ('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), ('me', 249), ('have', 247), ('were', 243), ('Peter', 238)]
28/03/15 15:30:05 INFO SparkContext: Invoking stop() from shutdown hook
28/03/15 15:30:05 INFO SparkUI: Stopped Spark web UI at http://192.168.122.108:4040
28/03/15 15:30:06 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
28/03/15 15:30:06 INFO MemoryStore: MemoryStore cleared
28/03/15 15:30:06 INFO BlockManager: BlockManager stopped
28/03/15 15:30:06 INFO BlockManagerMaster: BlockManagerMaster stopped
28/03/15 15:30:06 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
28/03/15 15:30:06 INFO SparkContext: Successfully stopped SparkContext
28/03/15 15:30:06 INFO ShutdownHookManager: Shutdown hook called
28/03/15 15:30:06 INFO ShutdownHookManager: Deleting directory /tmp/spark-4e11935b-f53a-41c1-9eca-ea8ed5bccc43
28/03/15 15:30:06 INFO ShutdownHookManager: Deleting directory /tmp/spark-c44a99f7-d70e-470f-9ab1-4b64c1f3ca3c
28/03/15 15:30:06 INFO ShutdownHookManager: Deleting directory /tmp/spark-c44a99f7-d70e-470f-9ab1-4b64c1f3ca3c/pyspark-4e0b8812-1eee-4ce3-9417-c7563a8b1779
8b1779
```

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 the Visual Studio Code interface 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 'VBHAMIDIPATI1 [SSH: KVM\_VBHAMIDIPATI1]' with a 'spark' folder containing various scripts. The code editor shows a Python script 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]
```

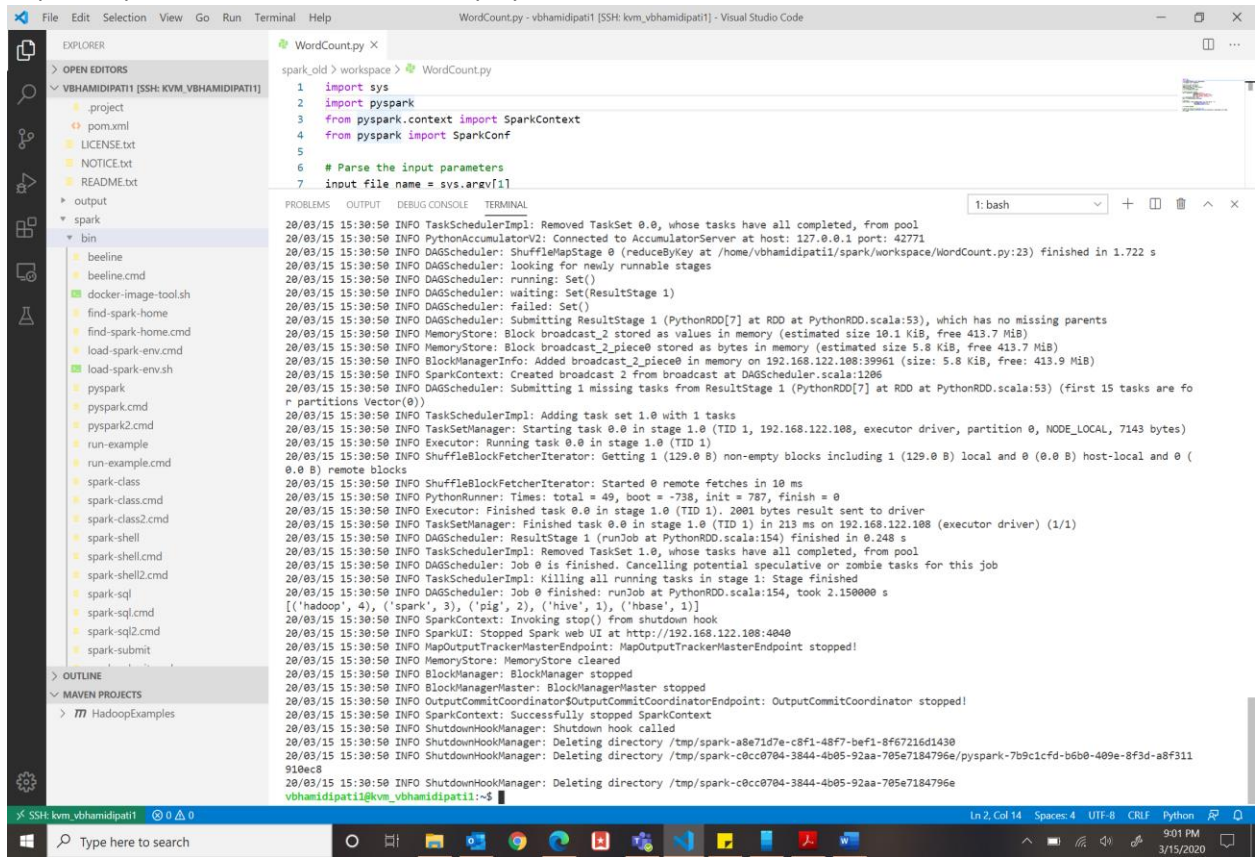
The terminal shows the command to run the script and its output:

```
vbhamidipati1@kvm_vbhamidipati1:~$ spark-submit /home/vbhamidipati1/spark/workspace/WordCount.py /home/vbhamidipati1/hadoop/data/test.txt 5
28/03/15 15:30:42 WARN Utils: Your hostname, kvm_vbhamidipati1 resolves to a loopback address: 127.0.1.1; using 192.168.122.108 instead (on interface ens3)
28/03/15 15:30:42 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/vbhamidipati1/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
28/03/15 15:30:43 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/home/vbhamidipati1/hadoop/data/test.txt
5
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
28/03/15 15:30:44 INFO SparkContext: Running Spark version 3.0.0-preview2
28/03/15 15:30:44 INFO ResourceUtils: =====
28/03/15 15:30:44 INFO ResourceUtils: Resources for spark.driver:
=====
28/03/15 15:30:44 INFO ResourceUtils: =====
28/03/15 15:30:44 INFO SparkContext: Submitted application: Word Count Spark
28/03/15 15:30:44 INFO SecurityManager: Changing view acls to: vbhamidipati1
28/03/15 15:30:44 INFO SecurityManager: Changing modify acls to: vbhamidipati1
28/03/15 15:30:44 INFO SecurityManager: Changing view acls groups to:
28/03/15 15:30:44 INFO SecurityManager: Changing modify acls groups to:
28/03/15 15:30:44 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(vbhamidipati1); groups with view permissions: Set(); users with modify permissions: Set(vbhamidipati1); groups with modify permissions: Set()
28/03/15 15:30:45 INFO Utils: Successfully started service 'sparkDriver' on port 44347.
28/03/15 15:30:45 INFO SparkEnv: Registering MapOutputTracker
28/03/15 15:30:45 INFO SparkEnv: Registering BlockManagerMaster
28/03/15 15:30:45 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
28/03/15 15:30:45 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
28/03/15 15:30:45 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
28/03/15 15:30:45 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-b0d73d68-6163-4b30-b048-ffa75707387e
28/03/15 15:30:45 INFO MemoryStore: MemoryStore started with capacity 413.9 MiB
28/03/15 15:30:45 INFO SparkEnv: Registering OutputCommitCoordinator
28/03/15 15:30:45 INFO SparkUI: Successfully started service 'SparkUI' on port 4040.
28/03/15 15:30:45 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.122.108:4040
28/03/15 15:30:46 INFO Executor: Starting executor ID driver on host 192.168.122.108
28/03/15 15:30:46 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 39961.
28/03/15 15:30:46 INFO NettyBlockTransferService: Server created on 192.168.122.108:39961
28/03/15 15:30:46 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
28/03/15 15:30:46 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 192.168.122.108, 39961, None)
28/03/15 15:30:46 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.122.108:39961 with 413.9 MiB RAM, BlockManagerId(driver, 192.168
```

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



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



```
File Edit Selection View Go Run Terminal Help
WordCount.py - vbhamidipati1 [SSH: KVM_VBHAMIDIPATI1] - Visual Studio Code

EXPLORER
  OPEN EDITORS
  VBHAMIDIPATI1 [SSH: KVM_VBHAMIDIPATI1]
    project
    pom.xml
    LICENSE.txt
    NOTICE.txt
    README.txt
    output
    spark
      bin
        beeline
        beeline.cmd
        docker-image-tool.sh
        find-spark-home
        find-spark-home.cmd
        load-spark-env.cmd
        load-spark-env.sh
        pyspark
        pyspark.cmd
        pyspark2.cmd
        run-example
        run-example.cmd
        spark-class
        spark-class.cmd
        spark-class2.cmd
        spark-shell
        spark-shell.cmd
        spark-shell2.cmd
        spark-sql
        spark-sql.cmd
        spark-sql2.cmd
        spark-submit
    OUTLINE
    MAVEN PROJECTS
    m HadoopExamples

WordCount.py X
spark_old > 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]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash

20/03/15 15:30:50 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
20/03/15 15:30:50 INFO PythonAccumulatorV2: Connected to AccumulatorServer at host: 127.0.0.1 port: 42771
20/03/15 15:30:50 INFO DAGScheduler: ShuffleMapStage 0 (reduceByKey at /home/vbhamidipati1/spark/workspace/WordCount.py:23) finished in 1.722 s
20/03/15 15:30:50 INFO DAGScheduler: looking for newly runnable stages
20/03/15 15:30:50 INFO DAGScheduler: running: Set()
20/03/15 15:30:50 INFO DAGScheduler: waiting: Set(ResultStage 1)
20/03/15 15:30:50 INFO DAGScheduler: failed: Set()
20/03/15 15:30:50 INFO DAGScheduler: Submitting ResultStage 1 (PythonRDD[7] at RDD at PythonRDD.scala:53), which has no missing parents
20/03/15 15:30:50 INFO MemoryStore: Block broadcast_2 stored as values in memory (estimated size 10.1 KiB, free 413.7 MiB)
20/03/15 15:30:50 INFO MemoryStore: Block broadcast_2_piece0 stored as bytes in memory (estimated size 5.8 KiB, free 413.7 MiB)
20/03/15 15:30:50 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on 192.168.122.108:39961 (size: 5.8 KiB, free: 413.9 MiB)
20/03/15 15:30:50 INFO SparkContext: Created broadcast 2 from broadcast at DAGScheduler.scala:1206
20/03/15 15:30:50 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 15:30:50 INFO TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
20/03/15 15:30:50 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, 192.168.122.108, executor driver, partition 0, NODE_LOCAL, 7143 bytes)
20/03/15 15:30:50 INFO Executor: Running task 0.0 in stage 1.0 (TID 1)
20/03/15 15:30:50 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 15:30:50 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 10 ms
20/03/15 15:30:50 INFO PythonRunner: Times: total = 49, boot = -738, init = 787, finish = 0
20/03/15 15:30:50 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1), 2001 bytes result sent to driver
20/03/15 15:30:50 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 213 ms on 192.168.122.108 (executor driver) (1/1)
20/03/15 15:30:50 INFO DAGScheduler: ResultStage 1 (runJob at PythonRDD.scala:154) finished in 0.248 s
20/03/15 15:30:50 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
20/03/15 15:30:50 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
20/03/15 15:30:50 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
20/03/15 15:30:50 INFO DAGScheduler: Job 0 finished: runJob at PythonRDD.scala:154, took 2.150000 s
[('hadoop', 4), ('spark', 3), ('pig', 2), ('hive', 1), ('hbase', 1)]
20/03/15 15:30:50 INFO SparkContext: Invoking stop() from shutdown hook
20/03/15 15:30:50 INFO SparkUI: Stopped Spark web UI at http://192.168.122.108:4040
20/03/15 15:30:50 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
20/03/15 15:30:50 INFO MemoryStore: MemoryStore cleared
20/03/15 15:30:50 INFO BlockManager: BlockManager stopped
20/03/15 15:30:50 INFO BlockManagerMaster: BlockManagerMaster stopped
20/03/15 15:30:50 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
20/03/15 15:30:50 INFO SparkContext: Successfully stopped SparkContext
20/03/15 15:30:50 INFO ShutdownHookManager: Shutdown hook called
20/03/15 15:30:50 INFO ShutdownHookManager: Deleting directory /tmp/spark-a8e71d7e-c8f1-48f7-bef1-8f67216d1430
20/03/15 15:30:50 INFO ShutdownHookManager: Deleting directory /tmp/spark-c0cc0704-3844-4b05-92aa-705e7184796e/pyspark-7b9c1cfd-b6b0-409e-8f3d-a8f311910ec8
20/03/15 15:30:50 INFO ShutdownHookManager: Deleting directory /tmp/spark-c0cc0704-3844-4b05-92aa-705e7184796e
vbhamidipati1@kvm_vbhamidipati1:~$
```

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