Table of Contents

Apache Spark Helper Lab

Throughout the course, it is important to reference the documentation sites of the open source Apache Project and other associated vendor sites for the data science tools we will use. Some of the tutorials that serve as references and/or provide extra help beyond our textbook include:

https://spark.apache.org/

https://pytorch.org/

https://www.scala-lang.org/

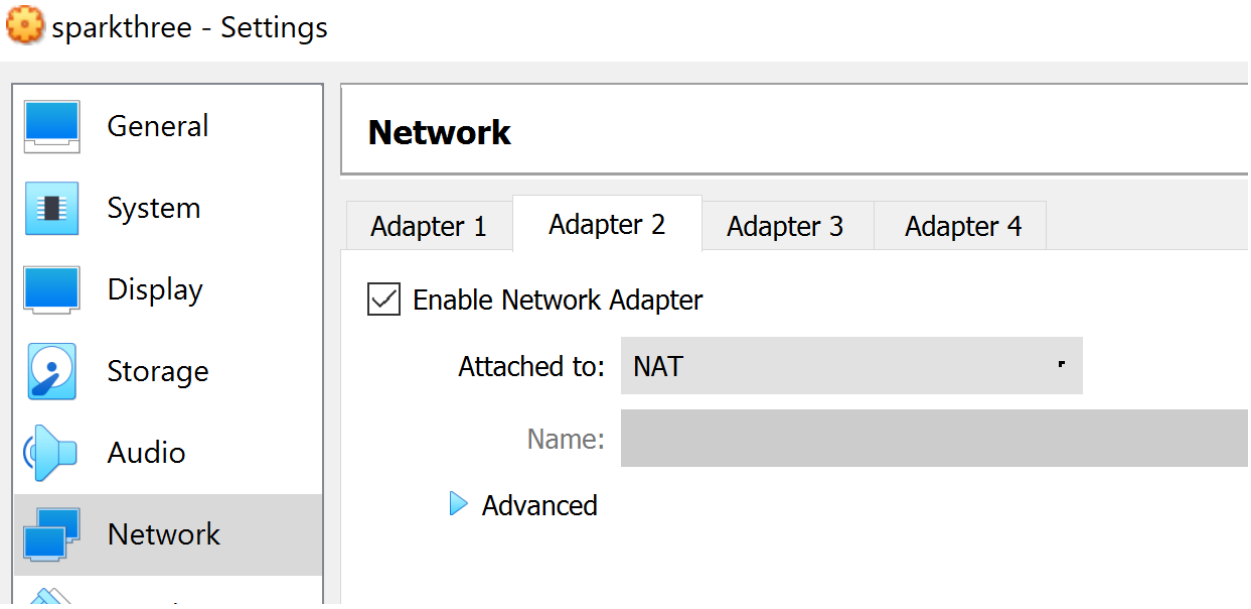
https://spark.apache.org/docs/latest/quick-start.html

https://phoenixnap.com/kb/install-spark-on-ubuntu

Apache Spark

Apache spark is a framework for massive distributed big data clusters that is similar to the Hadoop environment we have been learning. Like Hadoop, it is able to scale well. Spark provides functionality beyond many MapReduce jobs. It does not use the MapReduce execution engine but has developed an efficient distributed runtime for tasks in a clustered environment.

In this tutorial, you can use your primary Hadoop VM, hadoopone, or start a new Ubunut VM. The first step is to go back into Bridged Adapter or NAT mode in your NIC networking settings of your VM so that you can connect to the internet to install software:



Spark is dependent on Java and Scala. Git is also very helpful, and we will make sure all three are installed together if not already on your system.

Download Spark

Currently, you want to install a compatible Java JDK for Spark. NOTE, you may be running Hadoop with and older JDK like Java 8. Check to see if the version of Spark you are installing works with your version of Java. To change Java versions you can run:

sudo update-alternatives --config java

Once you are in NAT network mode again and can access the Internet run an update:

sudo apt update

Next you can use curl or wget if you are not in a GUI and download the appropriate version of Apache Spark. I am using the latest version at the date of this writing with Hadoop 3:

cd /home/bobsmith

wget https://dlcdn.apache.org/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3-scala2.13.tgz

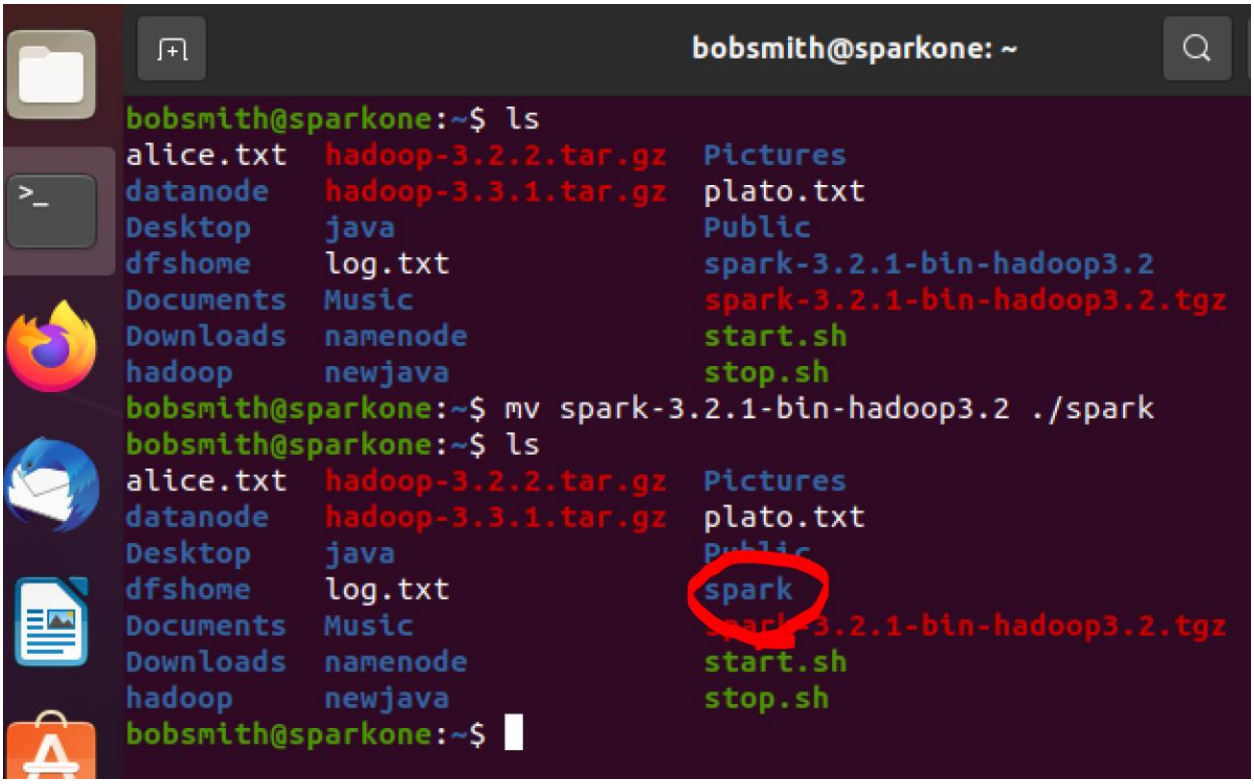
Alternatively, simply download the binary using a web browser from the GUI.

https://spark.apache.org/downloads.html

Move your tar file to the /home directory and unpackage it. I installed it in /home/bobsmith/spark, replace your version after the tar xzvf portion of the command:

tar xzvf spark-3.4.1-bin-hadoop3-scala2.13.tgz

Next, move the folder to the spark folder with the move command or "mv".



Note, there is no need to change the permissions as long as the Spark directory has 755. You also do not need to change ownership but I changed ownership of the file to bobsmith and root.

sudo chown -Rf bobsmith:root ./spark

```
bobsmith@sparkone:~$ sudo chown -Rf bobsmith:root ./spark
[sudo] password for bobsmith:
bobsmith@sparkone:~$ cd spark
bobsmith@sparkone:~/spark$ ls -la
total 156
drwxr-xr-x 13 bobsmith root      4096 Jan 20 14:10 .
drwxr-xr-x 27 bobsmith bobsmith  4096 Feb 21 13:06 ..
drwxr-xr-x  2 bobsmith root      4096 Jan 20 14:10 bin
drwxr-xr-x  2 bobsmith root      4096 Jan 20 14:10 conf
drwxr-xr-x  5 bobsmith root      4096 Jan 20 14:10 data
drwxr-xr-x  4 bobsmith root      4096 Jan 20 14:10 examples
drwxr-xr-x  2 bobsmith root     12288 Jan 20 14:10 jars
drwxr-xr-x  4 bobsmith root      4096 Jan 20 14:10 kubernetes
-rw-r--r--  1 bobsmith root     22878 Jan 20 14:10 LICENSE
drwxr-xr-x  2 bobsmith root      4096 Jan 20 14:10 licenses
-rw-r--r--  1 bobsmith root     57677 Jan 20 14:10 NOTICE
drwxr-xr-x  9 bobsmith root      4096 Jan 20 14:10 python
drwxr-xr-x  3 bobsmith root      4096 Jan 20 14:10 R
-rw-r--r--  1 bobsmith root      4512 Jan 20 14:10 README.md
-rw-r--r--  1 bobsmith root       167 Jan 20 14:10 RELEASE
drwxr-xr-x  2 bobsmith root      4096 Jan 20 14:10 sbin
drwxr-xr-x  2 bobsmith root      4096 Jan 20 14:10 yarn
bobsmith@sparkone:~/spark$
```

Spark Environment Setup

Set your environment paths. Again, replace “bobsmith” with your correct directory name that includes your first name and last name.

nano /home/bobsmith/.bashrc

Add spark home and set the path. Note, your path may want to include Java, Hadoop, and Spark.

export SPARK_HOME=/home/bobsmith/spark

export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin

```
#Spark paths
export SPARK_HOME=/home/bobsmith/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Source your .bashrc:

source .bashrc

```
bobsmith@sparkone:~$ nano .bashrc
bobsmith@sparkone:~$ source .bashrc
bobsmith@sparkone:~$ echo $SPARK_HOME
/home/bobsmith/spark
bobsmith@sparkone:~$ ls
alice.txt   hadoop-3.2.2.tar.gz   Pictures
datanode    hadoop-3.3.1.tar.gz   plato.txt
Desktop     java                  Public
dfshome     log.txt               spark
Documents   Music                 spark-3.2.1-bin
Downloads   namenode              start.sh
hadoop      newjava               stop.sh
bobsmith@sparkone:~$ pwd
/home/bobsmith
bobsmith@sparkone:~$
```

Before your attempt to start Spark, make sure your Java JDK is correct with your installed version.

```
Press <enter> to keep the current choice[*], or type selection number: 0
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/java to provi
de /usr/bin/java (java) in auto mode
bobsmith@sparkone:~$ sudo update-alternatives --config javac
There are 2 choices for the alternative javac (providing /usr/bin/javac).

  Selection    Path                                          Priority   Status
------------------------------------------------------------
* 0            /usr/lib/jvm/java-11-openjdk-amd64/bin/javac   1111      auto mo
de
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/javac   1111      manual
mode
  2            /usr/lib/jvm/java-8-openjdk-amd64/bin/javac    1081      manual
mode

Press <enter> to keep the current choice[*], or type selection number: 0
bobsmith@sparkone:~$
```

Next, shutdown your VM, take a snapshot, and restart it in host only network mode.

sudo shutdown now

Spark Startup

cd /home/bobsmith/spark/sbin

./start-master.sh

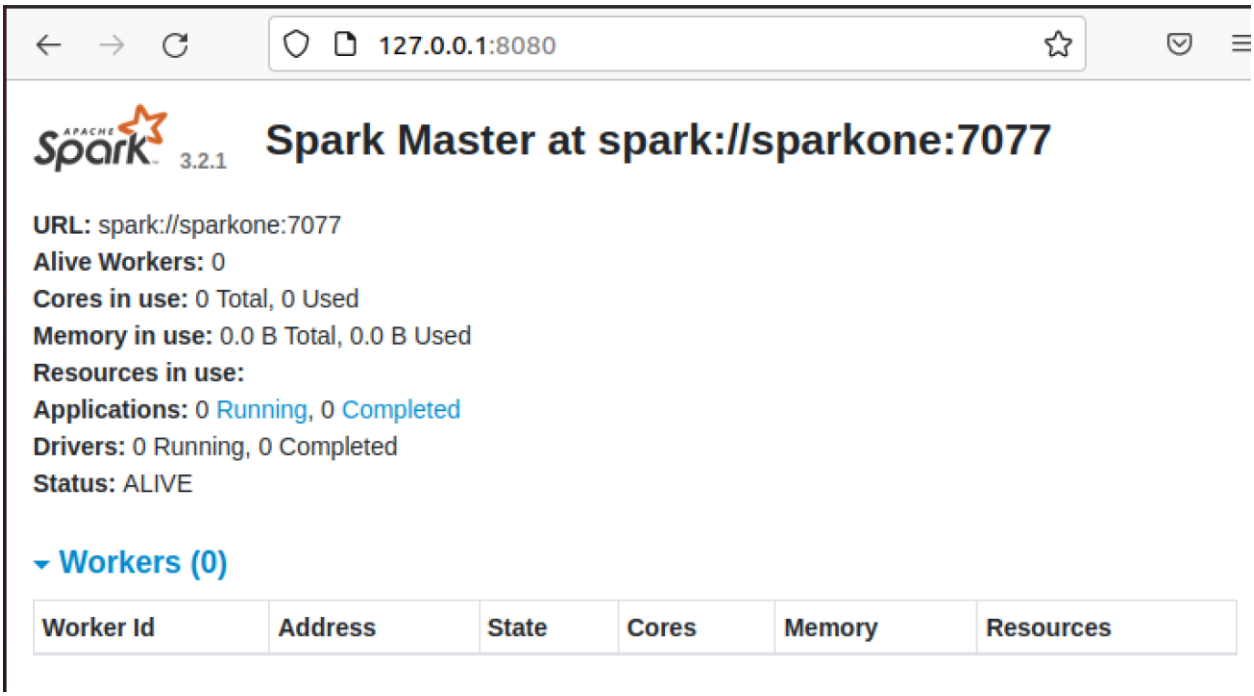
```
bobsmith@sparkone:~$ cd spark/
bobsmith@sparkone:~/spark$ ls
bin   data      jars        LICENSE   logs    python  README.md  sbin
conf  examples  kubernetes  licenses  NOTICE  R       RELEASE    yarn
bobsmith@sparkone:~/spark$ cd sbin/
bobsmith@sparkone:~/spark/sbin$ ls
decommission-slave.sh           start-worker.sh
decommission-worker.sh          start-workers.sh
slaves.sh                       stop-all.sh
spark-config.sh                 stop-history-server.sh
spark-daemon.sh                 stop-master.sh
spark-daemons.sh                stop-mesos-dispatcher.sh
start-all.sh                    stop-mesos-shuffle-service.sh
start-history-server.sh         stop-slave.sh
start-master.sh                 stop-slaves.sh
start-mesos-dispatcher.sh       stop-thriftserver.sh
s Ubuntu Software ffle-service.sh  stop-worker.sh
start-slave.sh                  stop-workers.sh
start-slaves.sh                 workers.sh
start-thriftserver.sh
bobsmith@sparkone:~/spark/sbin$ ./start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /home/bobsmith/spark
/logs/spark-bobsmith-org.apache.spark.deploy.master.Master-1-sparkone.out
```

You should see spark listening on port 8080 by default but check your log files for the correct port if you are binding to a different port and/or spark did not start correctly:

```
bobsmith@sparkone:~/spark/sbin$ sudo netstat -tulpn | grep 8080
tcp6       0      0 :::8080                 :::*                    LISTEN
 1578/java
bobsmith@sparkone:~/spark/sbin$
```

To navigate to spark open a web browser and go to port 8080 on your VM:

http://127.0.0.1:8080/



Spark secondary worker servers

We can use the workers.template file in spark/conf similar to Hadoop workers. You have to rename the Spark template files. For example, workers.template becomes workers.

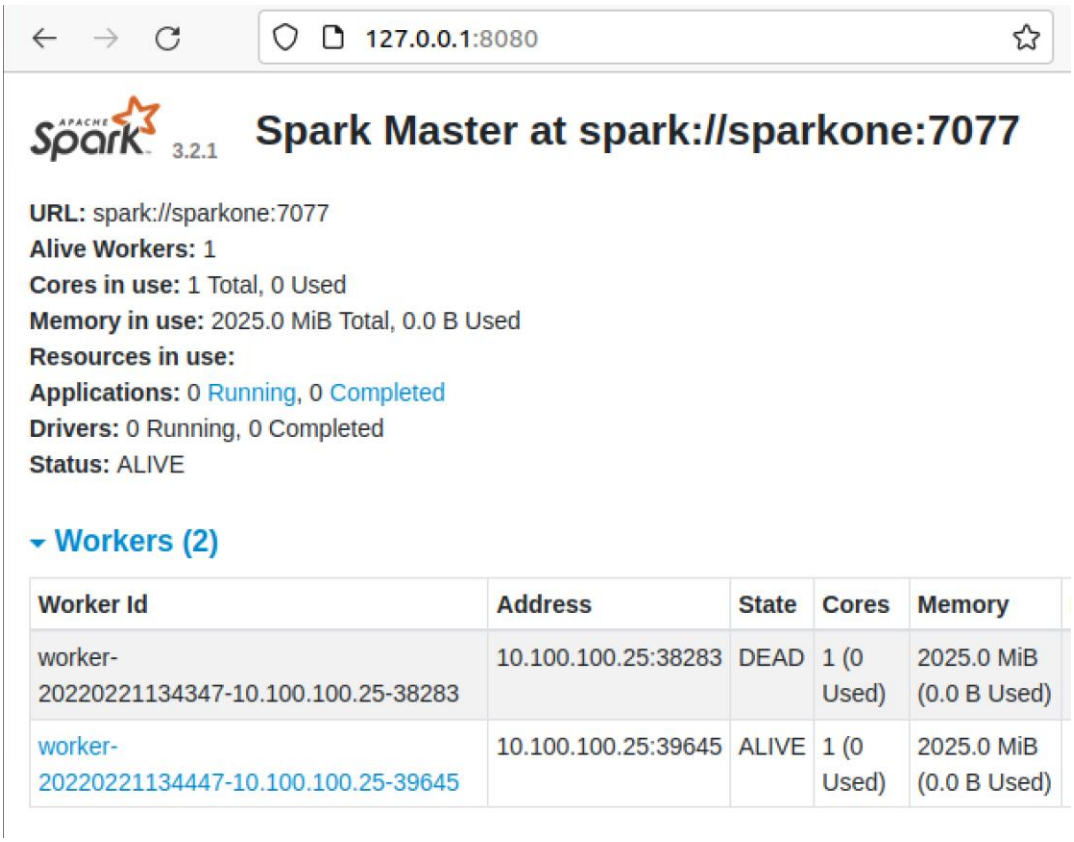
We can also standup additional worker servers using the start script in the screenshot below. You need workers to scale RDDs, which can be on the same VM for testing purposes:

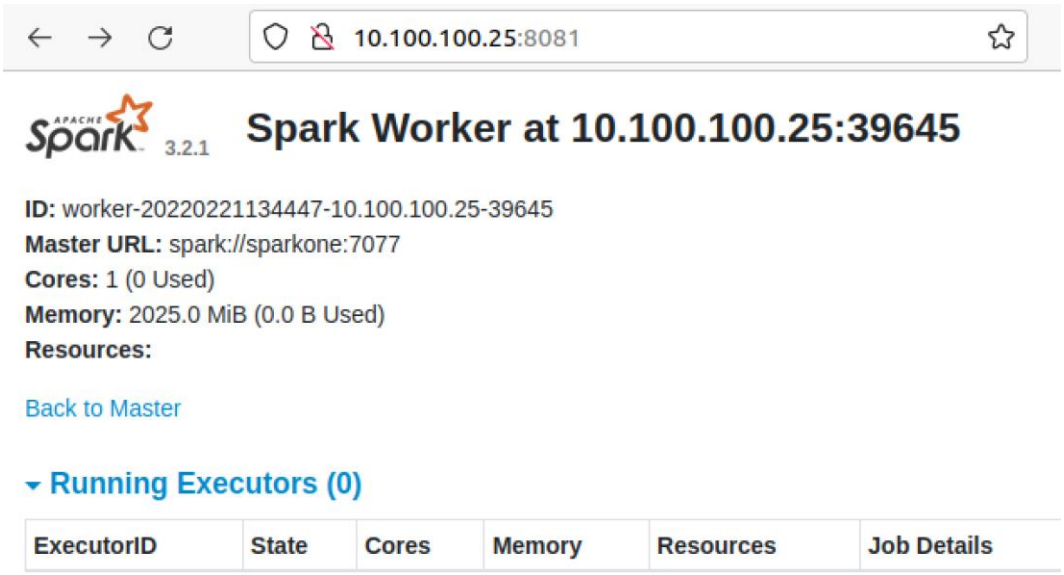
./start-worker.sh -c 1 spark://hadoopone:7077

You can pass it options such as the number of CPU cores the worker can use. In this example, I am passing it one (1) CPU core with the -c option 1:

```
bobsmith@sparkone:~/spark/sbin$ ./start-worker.sh -c 1 spark://sparkone:7077
starting org.apache.spark.deploy.worker.Worker, logging to /home/bobsmith/spark
/logs/spark-bobsmith-org.apache.spark.deploy.worker.Worker-1-sparkone.out
bobsmith@sparkone:~/spark/sbin$
```

In this example, hadoopone is my hostname/VM and this is followed by the port number where I want this worker service to bind. If working correctly, you should see the new worker in the spark web app:





Running the Spark shell / Scala

Scala is accessed using the spark-shell command from the spark bin directory:

./spark-shell

It is in the bin directory in contrast to the sbin directory:

```
bobsmith@sparkone:~/spark/sbin$ cd /home/bobsmith/spark/bin
bobsmith@sparkone:~/spark/bin$ ls
beeline                pyspark            spark-class.cmd    spark-sql
beeline.cmd            pyspark2.cmd       sparkR             spark-sql2.cmd
docker-image-tool.sh   pyspark.cmd        sparkR2.cmd        spark-sql.cmd
find-spark-home        run-example        sparkR.cmd         spark-submit
find-spark-home.cmd    run-example.cmd    spark-shell        spark-submit2.cmd
load-spark-env.cmd     spark-class        spark-shell2.cmd   spark-submit.cmd
load-spark-env.sh      spark-class2.cmd   spark-shell.cmd
bobsmith@sparkone:~/spark/bin$ ./spark-shell
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 setLogLev
el(newLevel).
22/02/21 13:49:48 WARN NativeCodeLoader: Unable to load native-hadoop library f
or your platform... using builtin-java classes where applicable
Spark context Web UI available at http://sparkone:4040
Spark context available as 'sc' (master = local[*], app id = local-164547298885
0).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.2.1
      /_/
```

Note, if you paths are set correctly, you should be able to run spark-shell from any directory. Congratulations, you have a spark shell running. Next, let’s try a Scala example.

Spark Scala Example

The Scala shell allows us to work with our spark environment including doing some basic filtering, mapping, and reduction. We can also use Python instead of scala to work with our data.

As a basic test, you can use some of your e-books to do a word count. Spark utilizes what are called resilient distributed datasets or RDDs to organize data across all the clustered worker nodes. You can create RDDs using what are labeled transformations, including joins, reducers, maps, filters, etc. You can also combine existing RDDs. RDDs can also use HDFS.

Use Spark documentation and briefly introduce yourself to RDD programming:
https://spark.apache.org/docs/latest/rdd-programming-guide.html

As a basic example, I will use an eBook I downloaded into /home/hdp/books. The book is Alice in Wonderland from the Gutenberg project: http://www.gutenberg.org/

First, create a variable that holds the contents of the book using "val". Then, create a word count variable. Use a line split and reduce the book's contents into a subsequent file. Cache the results and save the output as a new file. The following screenshot shows this example in scala:

```
scala> val alicebook = sc.textFile("/home/hdp/books/alice.txt")
alicebook: org.apache.spark.rdd.RDD[String] = /home/hdp/books/alice.txt MapParti
tionsRDD[8] at textFile at <console>:24

scala> val alicewordcount = alicebook.flatMap(line => line.split(" ")).map(word
=> (word, 1)).reduceByKey(_+_);
alicewordcount: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] at red
uceByKey at <console>:25

scala> alicewordcount.cache()
res0: alicewordcount.type = ShuffledRDD[11] at reduceByKey at <console>:25

scala> alicewordcount.saveAsTextFile("aliceresults")

scala>
```

Once you have saved the new text file, open another terminal window and go to the spark home directory. You can find it using the locate command after updating your locate database like this:

```
hdp@hadoopone:~/spark/data$ sudo updatedb
[sudo] password for hdp:
hdp@hadoopone:~/spark/data$ locate aliceresults
/home/hdp/spark/bin/aliceresults
/home/hdp/spark/bin/aliceresults/._SUCCESS.crc
/home/hdp/spark/bin/aliceresults/.part-00000.crc
/home/hdp/spark/bin/aliceresults/.part-00001.crc
/home/hdp/spark/bin/aliceresults/_SUCCESS
/home/hdp/spark/bin/aliceresults/part-00000
/home/hdp/spark/bin/aliceresults/part-00001
hdp@hadoopone:~/spark/data$
```

You should be able to see the results in the folder. Open your results to see what you created!

```
hdp@hadoopone:~/spark/bin/aliceresults$ ls
part-00000  part-00001  _SUCCESS
hdp@hadoopone:~/spark/bin/aliceresults$ nano part-00000
```

Here are some of my initial results:

```
(roses.,1)
(line:,1)
(order,2)
(tone.,9)
(mouse—a,1)
(said;,2)
(behind,12)
(pigeon,1)
```

You can exit scala using “:q”

:q

And using Enter on your keyboard.

And finally, if you installed Python, give pyspark a try.

Python / pyspark terminal

You can use python alternatively to scala. From the spark bin directory:

./pyspark

```
bobsmith@sparkone:~/spark/bin$ ./pyspark
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
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 setLogLev
el(newLevel).
22/02/21 13:53:55 WARN NativeCodeLoader: Unable to load native-hadoop library f
or your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.1
      /_/

Using Python version 3.8.10 (default, Nov 26 2021 20:14:08)
Spark context Web UI available at http://sparkone:4040
Spark context available as 'sc' (master = local[*], app id = local-164547323619
0).
SparkSession available as 'spark'.
>>>
```

I certainly encourage you to try other examples at this time. Congratulations on finishing this Spark introductory lab!