1. Show the variables are set up. Screenshots show the scp transfer of the file.

[rsha@localhost ~]$ sbt sbtVersion

[info] Loading project definition from /home/rsha/project

[info] Set current project to rsha (in build file:/home/rsha/)

[info] 1.0.2

[rsha@localhost ~]$ java -version

openjdk version "1.8.0\_131"

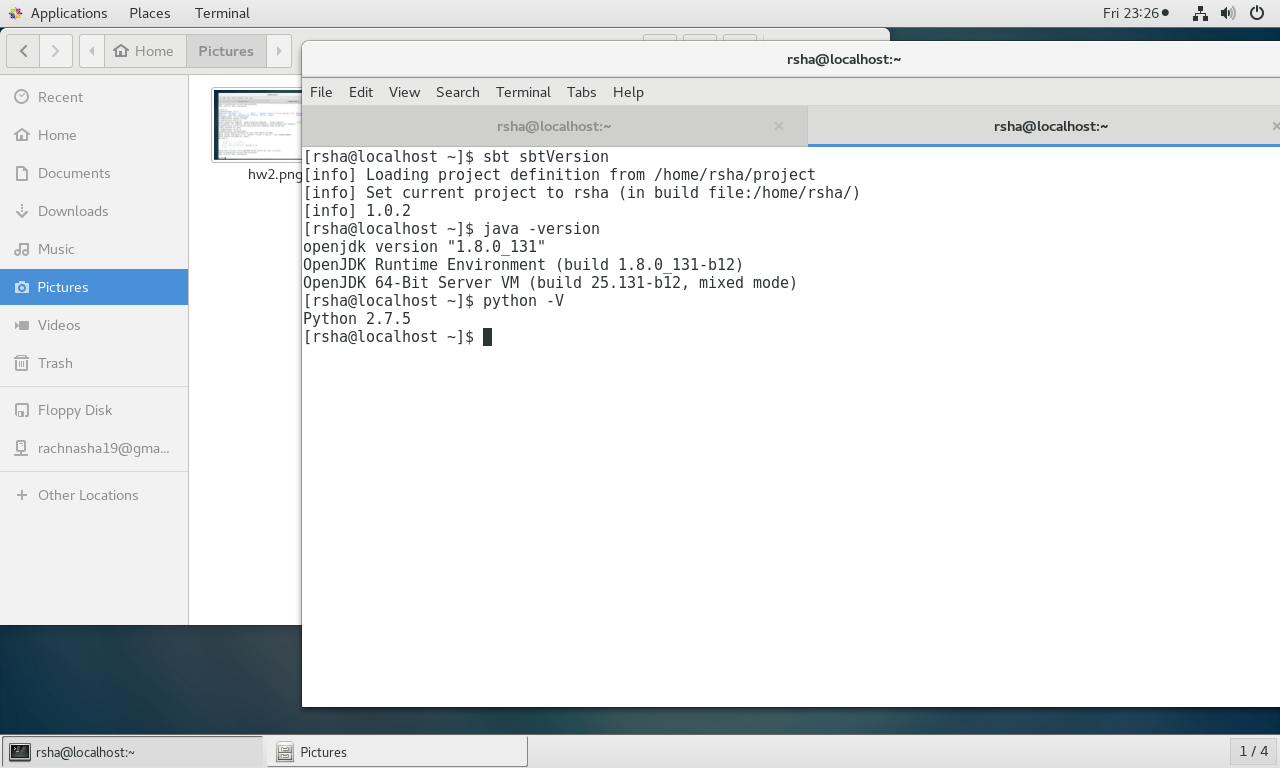
OpenJDK Runtime Environment (build 1.8.0\_131-b12)

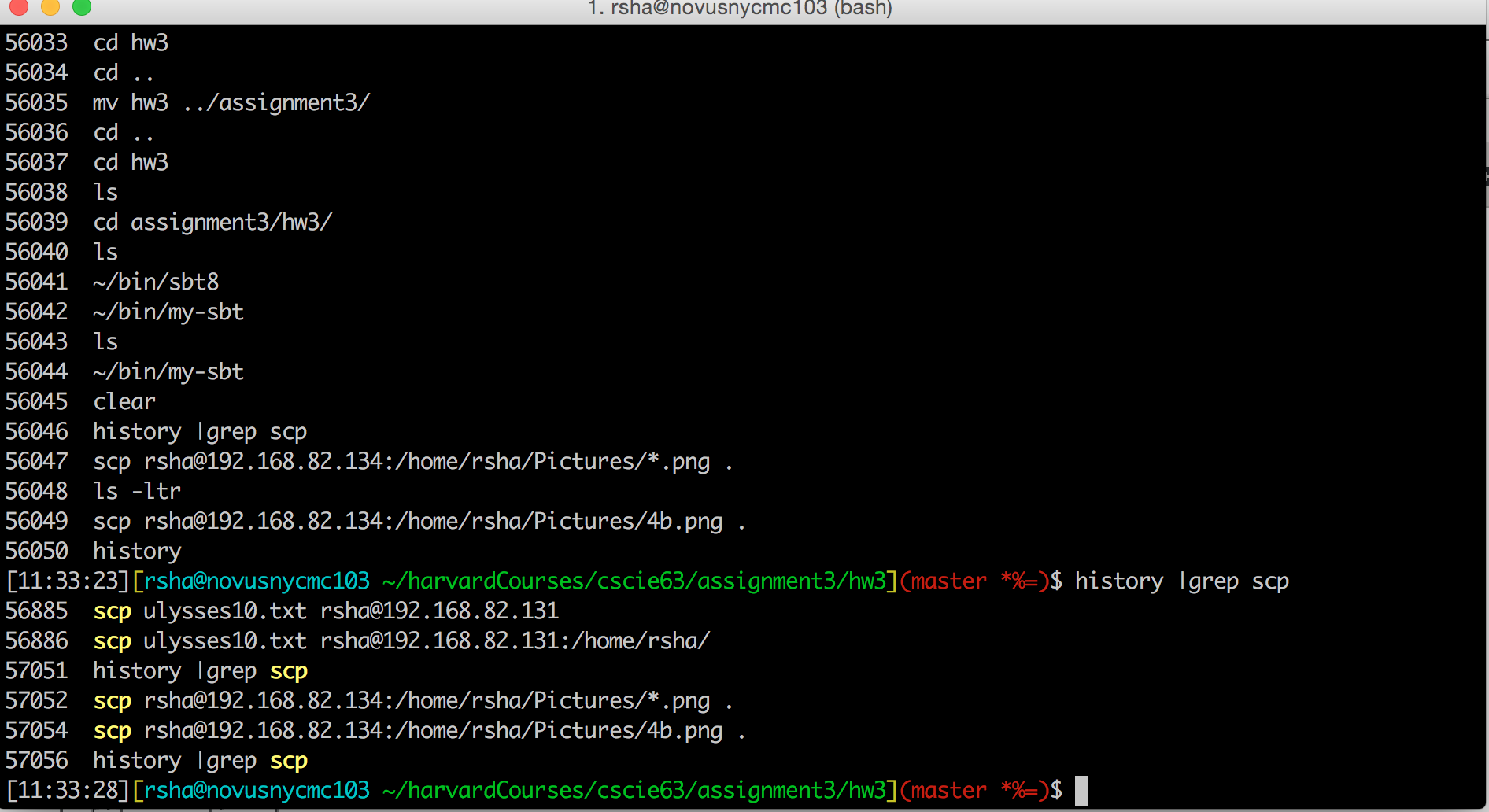
OpenJDK 64-Bit Server VM (build 25.131-b12, mixed mode)

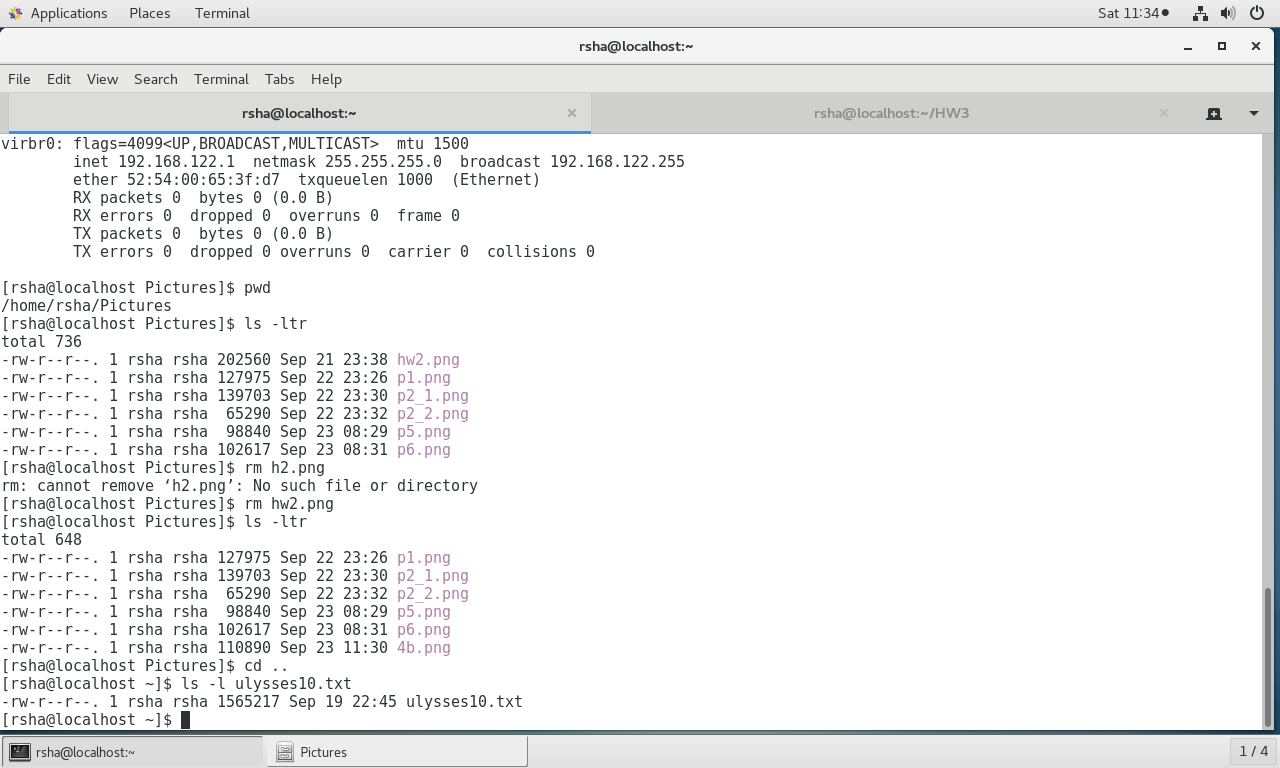
[rsha@localhost ~]$ python -V

Python 2.7.5

[rsha@localhost ~]$







2. Showing pyspark is set up and warnings are eliminated on spark-shell start up. Screenshots also provided below.

[rsha@localhost ~]$ pyspark

Python 2.7.5 (default, Aug 4 2017, 00:39:18)

[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

17/09/22 23:29:41 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

17/09/22 23:29:42 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 192.168.82.134 instead (on interface ens33)

17/09/22 23:29:42 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

17/09/22 23:29:52 WARN ObjectStore: Failed to get database global\_temp, returning NoSuchObjectException

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_ / .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.2.0

/\_/

Using Python version 2.7.5 (default, Aug 4 2017 00:39:18)

SparkSession available as 'spark'.

>>> exit()

[rsha@localhost ~]$ spark-shell

Spark context Web UI available at http://192.168.82.134:4040

Spark context available as 'sc' (master = local[\*], app id = local-1506137487665).

Spark session available as 'spark'.

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.2.0

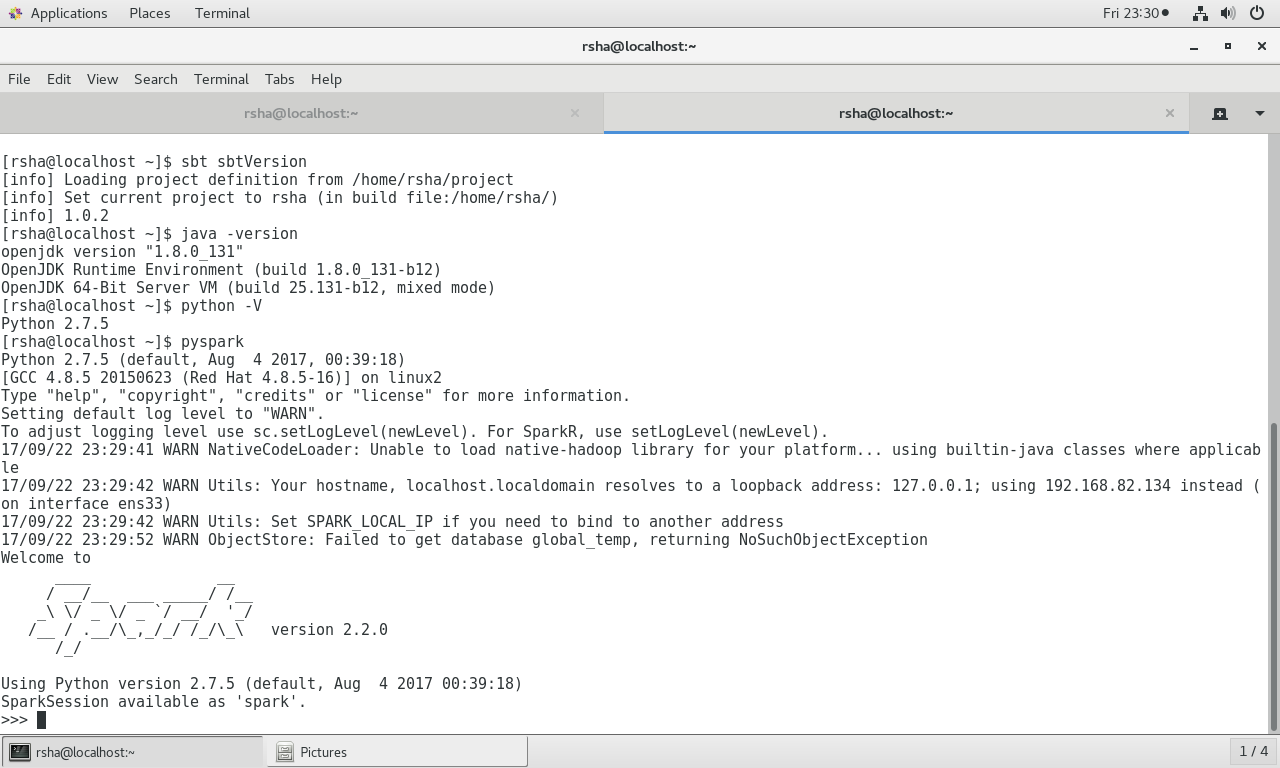
/\_/

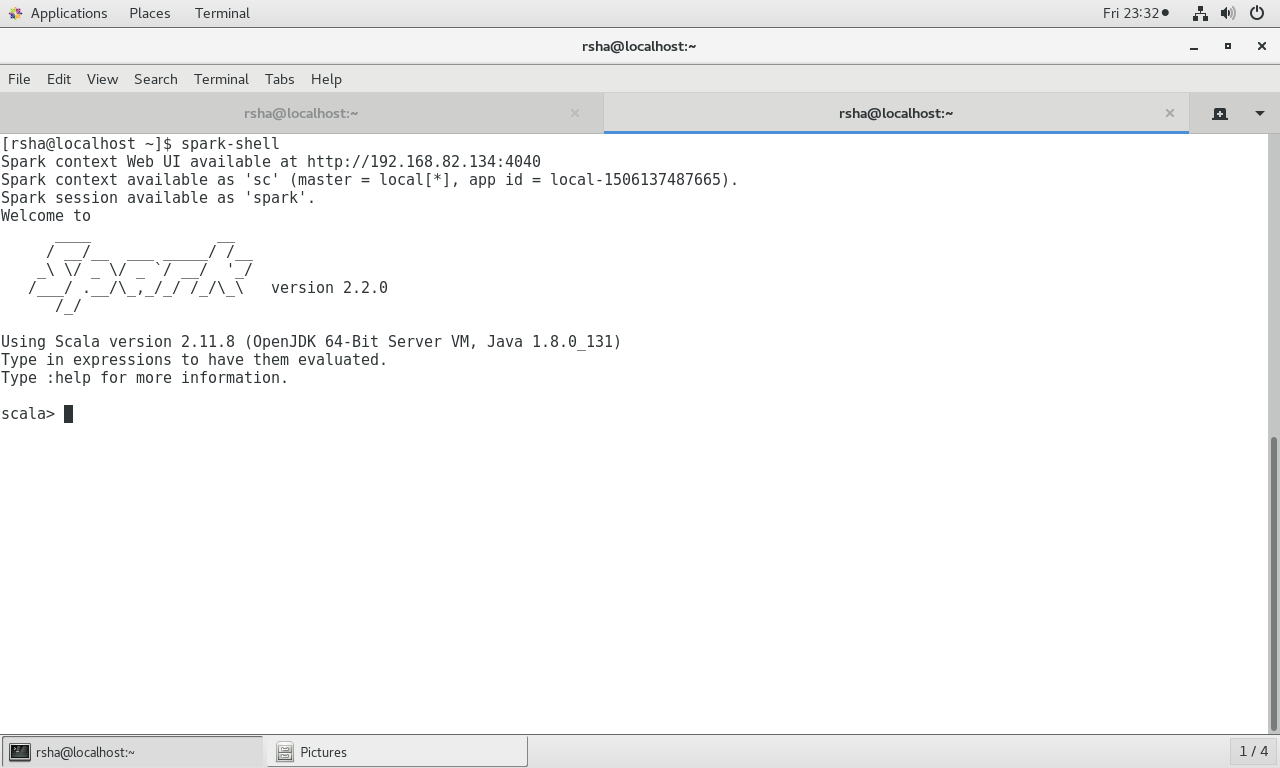
Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0\_131)

Type in expressions to have them evaluated.

Type :help for more information.

scala>





3. Count number of lines with one of 3 words using RDD

(a) With a lambda function

>>>

>>> lines = sc.textFile("/home/rsha/ulysses10.txt")

>>> timeOfDay = lines.filter(lambda line: "morning" in line or "afternoon" in line or "night" in line)

>>> timeOfDay.count()

418

(b) With a named function ( scala ) :

scala> val fn : (String => Boolean) = (s:String) => s.contains("morning") || s.contains("afternoon") || s.contains("night")

fn: String => Boolean = <function1>

scala> val l = sc.textFile("/home/rsha/ulysses10.txt")

l: org.apache.spark.rdd.RDD[String] = /home/rsha/ulysses10.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val timesOfDay = l.filter(fn)

timesOfDay: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:28

scala> val timesOfDay = l.filter(fn(\_))

timesOfDay: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[3] at filter at <console>:28

scala> timesOfDay.count()

res0: Long = 418

4. Count number of lines with one of 3 words using DataFrame

a. With a lambda (scala)

scala> val dataset = spark.read.textFile("/home/rsha/ulysses10.txt")

dataset: org.apache.spark.sql.Dataset[String] = [value: string]

scala> val filtered = dataset.filter($"value".contains("morning") || $"value".contains("afternoon") || $"value".contains("night"))

filtered: org.apache.spark.sql.Dataset[String] = [value: string]

scala> filtered.count()

res2: Long = 418

b. With a named function (scala )

scala> val dataset = spark.read.textFile("/home/rsha/ulysses10.txt")

dataset: org.apache.spark.sql.Dataset[String] = [value: string]

scala> val filtered :(String => Boolean) = (s:String) => (s.contains("morning") || s.contains("afternoon") || s.contains("night"))

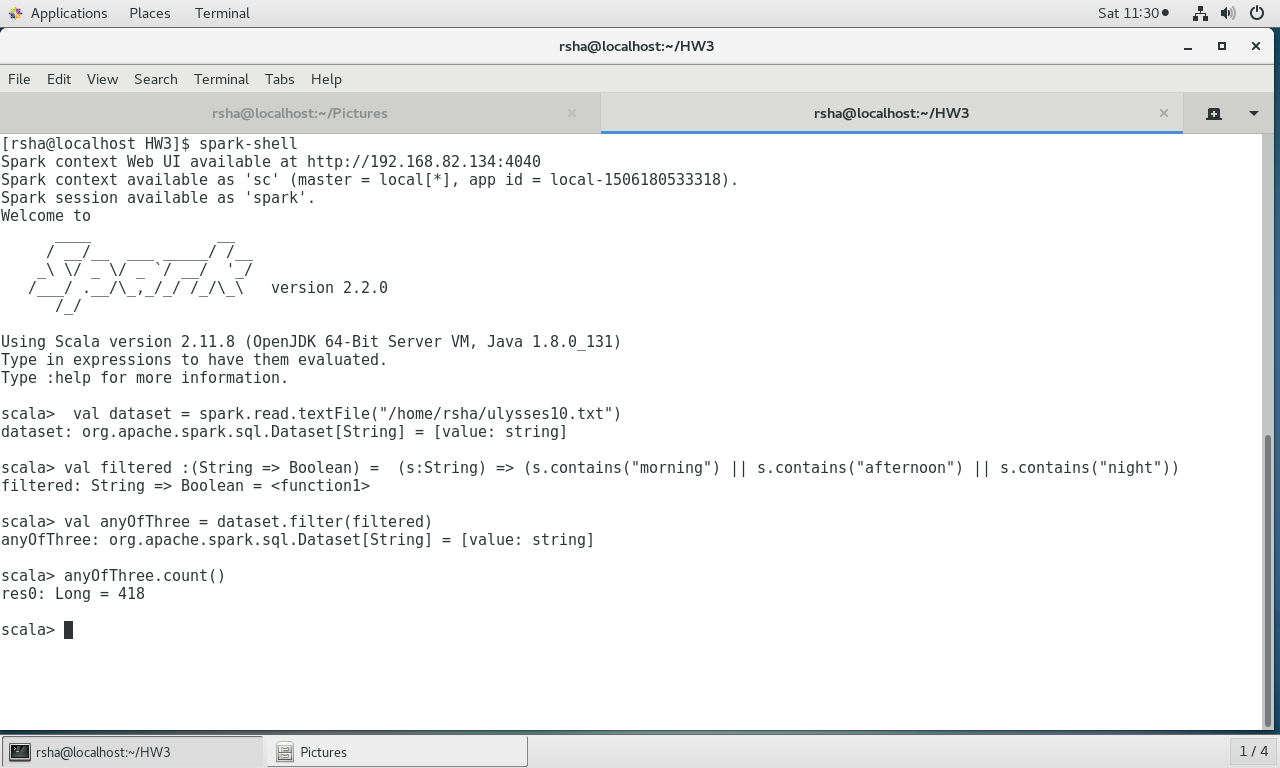
filtered: String => Boolean = <function1>

scala> val anyOfThree = dataset.filter(filtered)

anyOfThree: org.apache.spark.sql.Dataset[String] = [value: string]

scala> anyOfThree.count()

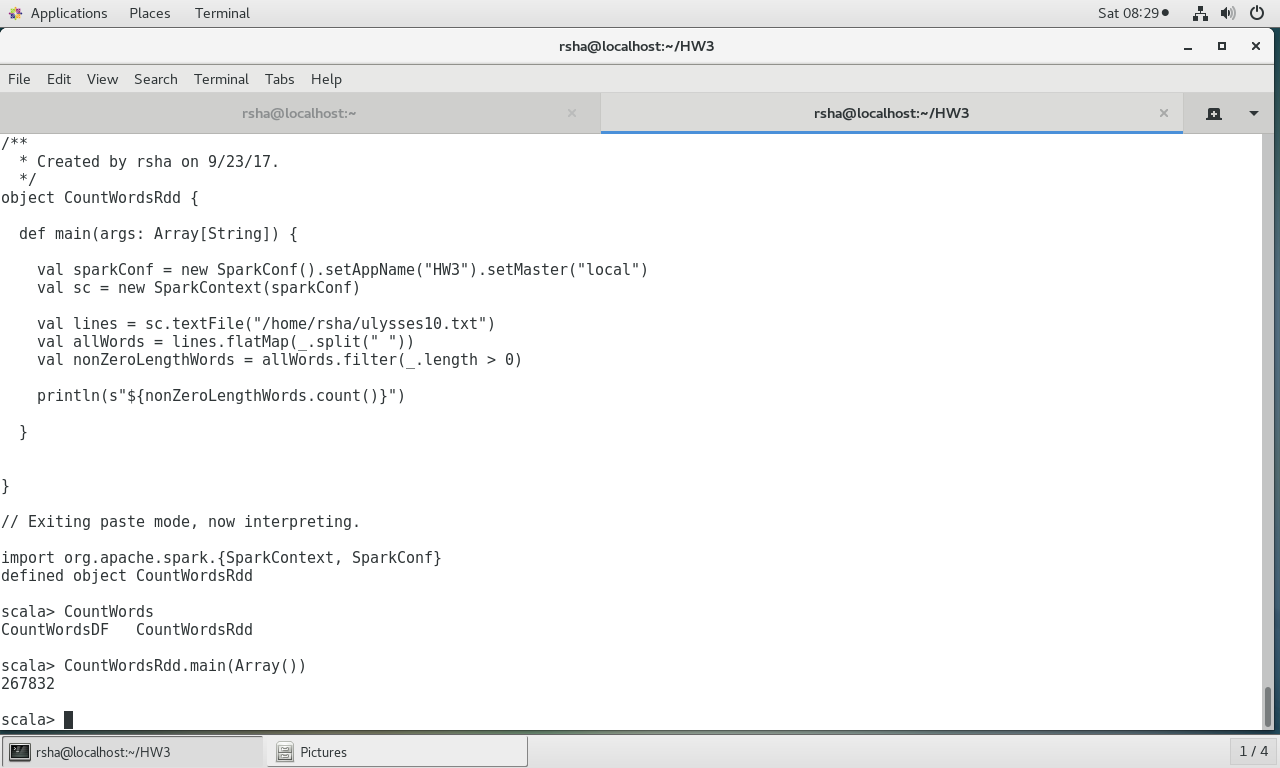
res0: Long = 418



5. Standalone appliction word count using RDD ( scala ) – finds all **nonZeroLength** words in the file ( similar to unix wc utility)

**import** org.apache.spark.{SparkContext, SparkConf}  
  
*/\*\*  
 \* Created by rsha on 9/23/17.  
 \*/***object** CountWordsRdd {  
  
 **def** main(args: Array[String]) {  
  
 **val** sparkConf = **new** SparkConf().setAppName(**"HW3"**).setMaster(**"local"**)  
 **val** sc = **new** SparkContext(sparkConf)  
  
 **val** lines = sc.textFile(**"/home/rsha/ulysses10.txt"**)  
 **val** allWords = lines.flatMap(\_.split(**" "**))  
 **val** nonZeroLengthWords = allWords.filter(\_.length > 0)  
  
 *println*(**s"$**{nonZeroLengthWords.count()}**"**)  
  
 }  
}

produces - 267832



6. Standalone appliction word count using DataSet ( scala ) – finds all **nonZeroLength** words in the file ( similar to unix wc utility)

**import** org.apache.spark.sql.SparkSession  
  
*/\*\*  
 \* Created by rsha on 9/23/17.  
 \*/***object** CountWordsDF {  
  
 **def** main(args: Array[String]) {  
 **val** sparkSession = SparkSession.*builder*.appName(**"Simple Application"**).getOrCreate()  
 **val** lines = sparkSession.read.textFile(**"/home/rsha/ulysses10.txt"**).cache()  
 **val** allWords = lines.flatMap(\_.split(**" "**))  
 **val** nonZeroLengthWords = allWords.filter(\_.length > 0).count()  
 *println*(**s"$**{nonZeroLengthWords}**"**)  
 sparkSession.stop()  
 }  
  
}

produces - 267832

