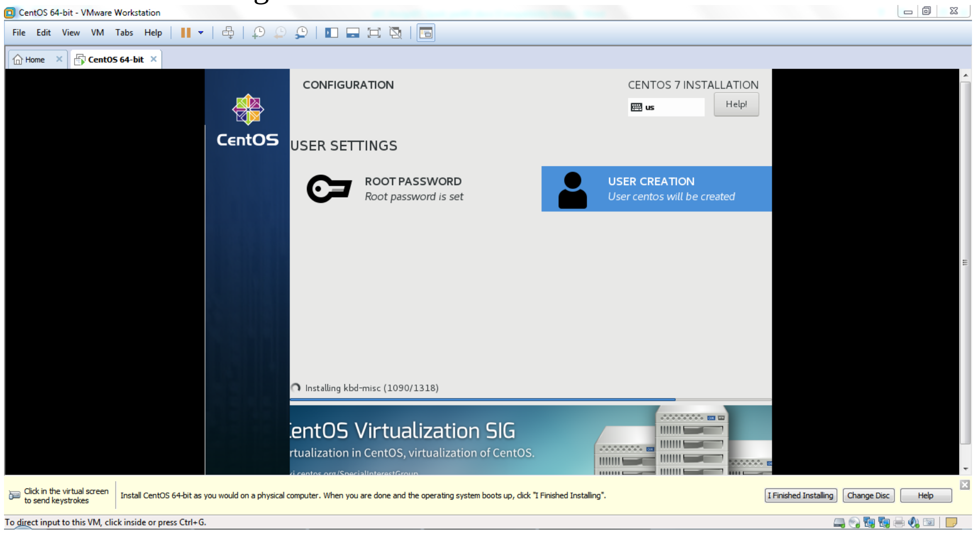
Student Name here

Use either VMWare Workstation or VMWare Fusion
https://e5.onthehub.com/WebStore/Welcome.aspx?ws=4185a0dc-d0d1-e511-9416-b8ca3a5db7a1&vsro=8
If you feel comfortable with a virtualization technology different from VMWare, please be free to use it.

Problem 1. Create your own Virtual Machine with a Linux operating system. The lecture notes speak about CentOS. You are welcome to work with another Linux OS. When creating the VM, create an administrative user. Call that user whatever you feel like. Please record the password of the new user. Once the VM is created transfer the attached text file Ulysses10.txt to the home of new user. You can do it using scp (secure copy command) or email. Examine the version of Java, Python and Scala on your VM. If any of those versions is below requirements for Spark 2.2 install proper version. Set JAVA_HOME environmental variable. Set your PATH environmental variable properly, so that you can invoke: `java, sbt` and `python` commands from any directory on your system.
[20%]

I downloaded the software from:
https://e5.onthehub.com/WebStore/Welcome.aspx?ws=4185a0dc-d0d1-e511-9416-b8ca3a5db7a1&vsro=8
This is the initial login screen.


Click in the virtual screen to send keystrokes
Install CentOS 64-bit as you would on a physical computer. When you are done and the operating system boots up, click "I Finished Installing".
To direct input to this VM, click inside or press Ctrl+G.

First, I transferred the Ulysses10.txt file to the VM with SCP.
I started by opening port 22 on the VM with the following commands:
```
[centos@localhost ~]$ sudo firewall-cmd --zone=public --
add-port=22/tcp --  permanent
success
[centos@localhost ~]$ sudo firewall-cmd --reload
success
```

I then rebooted the virtual machine and ran “ifconfig” to get the IP address of theVM, 192.168.233.129 .
After that, I used SCP to transfer the ulysses.txt file to the VM.

```
λ scp ulysses10.txt centos@192.168.233.129:~
centos@192.168.233.129's password:
ulysses10.txt
100% 1529KB  37.3MB/s   00:00
```

The Java version is shown to be 1.8.0_131.
The Python version is shown to be 2.7.5.
The Scala version is shown to be 2.12.3. I had to install in as shown below.

Spark 2.2 is compatible with these versions per the text at https://spark.apache.org/docs/latest/ :

> “Spark runs on Java 8+, Python 2.7+/3.4+ and R 3.1+. For the Scala API, Spark 2.2.0 uses Scala 2.11. You will need to use a compatible Scala version (2.11.x).”

Check Java version:

```
[centos@localhost home]$ which
java/usr/bin/java[centos@localhost home]$ java -
versionopenjdk 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)
```

Check Python version:

```
[centos@localhost home]$ which python
/usr/bin/python
[centos@localhost home]$ python –version
Python 2.7.5
```

Install Scala (output not included for brevity):

```
[centos@localhost Downloads]$ curl
https://bintray.com/sbt/rpm/rpm > bintray-sbt-rpm.repo
[centos@localhost Downloads]$ sudo mv bintray-sbt-rpm.repo
/etc/yum.repos.d/
[centos@localhost Downloads]$ sudo yum install sbt
```

Check Scala version:

```
[centos@localhost ~]$ sbt
[info] Loading project definition from /home/centos/project
[info] Set current project to centos (in build
file:/home/centos/)
[info] sbt server started at 127.0.0.1:5438
sbt:centos> scalaVersion
[info] 2.12.3
```

I modified the bashrc file:

```
[centos@localhost ~]$ sudo vi /etc/bashrc
```

To add the JAVA_HOME environmental variable with these two lines.

```
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-
11.b12.el7.x86_64/jre
export JAVA_HOME
```

From here, I tested that I could invoke python, sbt, and java from the command line.

```
[centos@localhost ~]$ python
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.
>>>

[centos@localhost ~]$ sbt
[info] Loading project definition from /home/centos/project
[info] Set current project to centos (in build
file:/home/centos/)
[info] sbt server started at 127.0.0.1:5438
sbt:centos>

[centos@localhost ~]$ java
Usage: java [-options] class [args...]
               (to execute a class)
or  java [-options] -jar jarfile [args...]
               (to execute a jar file)
where options include:
```

{{truncated for clarity}}

Problem 2. Install Spark 2.2 on your VM. Make sure that `pyspark` is also installed. Demonstrate that you can successfully open `spark-shell` and that you can eliminate most of WARNing messages.
[15%]

I downloaded the installer from https://spark.apache.org/downloads.html.
Then I ran the following command:

```
[centos@localhost Downloads]$ sudo tar zxvf spark-2.2.0-
     bin-hadoop2.7.tgz -C /opt
```

Then I set up a link called "spark"

```
[centos@localhost opt]$ sudo ln -fs spark-2.2.0-bin-
hadoop2.7 /opt/spark
[centos@localhost opt]$ ls -la
total 0
```

```
drwxr-xr-x.  4 root root  62 Sep 17 17:28 .
dr-xr-xr-x. 17 root root 224 Sep 17 15:47 ..
drwxr-xr-x.  2 root root   6 Mar 26  2015 rh
lrwxrwxrwx.  1 root root  25 Sep 17 17:28 spark -> spark-
2.2.0-bin-hadoop2.7
drwxr-xr-x. 12  500  500 193 Jun 30 19:09 spark-2.2.0-bin-
hadoop2.7
```

I updated the .bash_profile file

```
[centos@localhost ~]$ vi .bash_profile
```

I added the first 3 lines for the JAVA_HOME and SPARK_HOME variables and modified the PATH variable as shown.

```
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-
11.b12.el7.x86_64/jre
export JAVA_HOME
export SPARK_HOME=/opt/spark
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$JAVA_HOME/bin:$SPARK
_HOME/bin
```

Then I ran this command to push out the recent .bash_profile updates.

```
[centos@localhost ~]$ source .bash_profile
```

To eliminate most of the warning messages, I created a new log4j.properties file from the existing template.

```
[centos@localhost conf]$ sudo cp log4j.properties.template
log4j.properties
```

Then changed the following line

From: `log4j.rootCategory=INFO, console`
To: `log4j.rootCategory=ERROR, console`

After that change, I can run spark-shell without getting too many notifications on the console.

```
[centos@localhost ~]$ spark-shell
Spark context Web UI available at
http://192.168.233.129:4040
Spark context available as 'sc' (master = local[*],    app
id = local-1505703318624).
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>
```

Problem 3. Use `pyspark` and Spark Python API to find the number of lines in the text file `ulysses10.txt` that contain word “afternoon” or “night” or “morning”. In this problem use RDD API. Do this in two ways, first create a lambda function which will test whether a line contains any one of those 3 words. Second, create a named function in the language of choice that returns TRUE if a line passed to it contains any one of those three words. Demonstrate that the count is the same. If convenient you are welcome to implement this problem in any other language: Scala, Java or R.
[15%]

First, I read the text file into an RDD called lines.

```
>>> lines = sc.textFile("ulysses10.txt")
```

Then, I used a lambda function to return the number of lines that contain “afternoon” or “night” or “morning”.

```
>>> lines.filter(lambda line: "afternoon" in line or
"night" in line or "morning" in line).count()
418
```

Then, I created a function to return TRUE for matching “afternoon” or “night” or “morning” and demonstrated that using the function still gives the same result of 418 lines.

```
>>> def hasTimeOfDay(line):
...     return "afternoon" in line or "night" in line or
"morning" in line
...
>>> lines.filter(hasTimeOfDay).count()
418
```

Problem 4. Implement the above task, finding the number of lines with one of those three words in file ulysses10.txt using Dataset API. Again, use the language of your choice.
[20%]

First, I read the text file into an RDD called lines.

```
>>> lines = sc.textFile("ulysses10.txt")
```

From there, I saved the lines to a DataFrame that consists of a single column called "line" where each row contains a line of text. Finally, I was able to filter out all of the rows that contained "afternoon", "night", or "morning" and get a final count. The final count of 418 matches the count I got in problem 3.

```
>>> from pyspark.sql import Row, SQLContext
>>> linesAsRows = lines.map(lambda line: Row(line=line))
>>> linesAsDF = linesAsRows.toDF()
>>> linesAsDF.filter(linesAsDF.line.contains("afternoon") \
...   | linesAsDF.line.contains("night") \
...   | linesAsDF.line.contains("morning")) \
...   .count()
418
```

Problem 5. Create a standalone Python script that will count all words in file `ulysses10.txt`. You are expected to produce a single number. Do it using RDD API. If convenient, you are welcome to implement this problem in other languages: Scala, Java or R.
[%15]

First, I created a python script called problem5.py, shown below. It initializes the spark context and then creates an RDD called lines. After that, it splits the text by spaces and filters out and words that are blank (which were collected from splitting blank lines between paragraphs). Finally, it counts the words and prints a result to the console.

```
from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("My App")
sc = SparkContext(conf = conf)

lines = sc.textFile("ulysses10.txt")
words = lines.flatMap(lambda line: line.split(" ")) \
     .filter(lambda words: words !=  "")
wordCount = words.count()
print "Words in Ulysses10.txt -->", wordCount

sc.stop()
```

Then, I ran the script using spark-submit. You can see it printed the result, 267832.

```
[centos@localhost ~]$ spark-submit
~/Documents/hw3/problem5.py
Words in Ulysses10.txt --> 267832
```

Problem 6. Create a standalone Python script that will count all words in file `ulysses10.txt`. You are expected to produce a single number. Do it using Dataset API. If convenient, you are welcome to implement this problem in other languages: Scala, Java or R.
[%15]

First, I created a python script called problem6.py, shown below. It initializes the spark context and SQL context and then reads the "ulysses10.txt" file into an RDD called lines. After that, it splits the text by spaces and filters out and words that are blank (which were collected from splitting blank lines between paragraphs). Then it organizes the words into rows, and eventually a DataFrame that has a single column named "word" with each row representing a single word. Finally, I ran the count() method to get a count of all the words and printed it to the console.

```python
from pyspark import SparkConf, SparkContext
from pyspark.sql import Row, SQLContext

conf = SparkConf().setMaster("local").setAppName("My App")
sc = SparkContext(conf = conf)
sqlContext = SQLContext(sc)

lines = sc.textFile("ulysses10.txt")
words = lines.flatMap(lambda line: line.split(" ")) \
     .filter(lambda words: words !=  "")

wordsAsRows = words.map(lambda word: Row(word=word))
wordsAsDF = sqlContext.createDataFrame(wordsAsRows)

wordCount = wordsAsDF.count()
print "Words in Ulysses10.txt -->", wordCount

sc.stop()
```

Then, I ran the script using spark-submit. You can see it printed the result, 267832, which is the same result as problem 5.

```
[centos@localhost ~]$ spark-submit
~/Documents/hw3/problem6.py
Words in Ulysses10.txt --> 267832
```