Week 3: Fun with Spark Lab

This week we'll get started working with Spark. Along the way, we'll

- Learn a bit about the Scala programming language;
- See how Spark's approach to MapReduce computation gives advantages in speed, expressiveness and interactivity;
- Get some experience with the RDD, the distributed data type that underlies Spark's computing model.

Part 1: Work through the *Just Enough Scala for Spark* notebook

To get started, I suggest you work through the *Just Enough Scala for Spark* notebook.
The notebook provides an interactive tutorial that covers the most important features and idioms of Scala for using Spark's Scala APIs.

Spark is implemented in Scala and thus Scala is in some sense Spark's native tongue. Thus, Spark work is sometimes more straightforward in Scala, although bindings for using Spark from Python, R and other languages exist. Also, new features are often supported in Scala first, before the adapters for other languages catch up.

We'll learn the Scala ropes using a self contained interactive tutorial that's packaged in a Docker container. To get it setup, do the following:

Get your Azure VM running and log in to it.

Stop your sandbox containers if they're running by executing:

```
sudo docker stop sandbox-proxy

sudo docker stop sandbox-hdp
```

We do the above to avoid port conflicts between the tutorial and the Hortonworks sandboxes. Remember you can use `sudo docker ps` to see which containers are running.

Check out the tutorial's Github repository with:

```
git clone https://github.com/deanwampler/JustEnoughScalaForSpark.git
```

Change into the directory that was just created by the checkout operation:

```
cd JustEnoughScalaForSpark/
```

The script called `run.sh` is the one of interest to us. You may want to look inside it using ‘cat’ or your favorite text editor. In brief it uses the ‘docker’ command to tell the docker daemon to create a container with the given properties from a container image that lies in a public repository. It also sets up port forwarding for port 8888 (which you’ll connect a browser to in order to access the Jupyter notebook server) and port 4040 (which Spark uses). Execute:

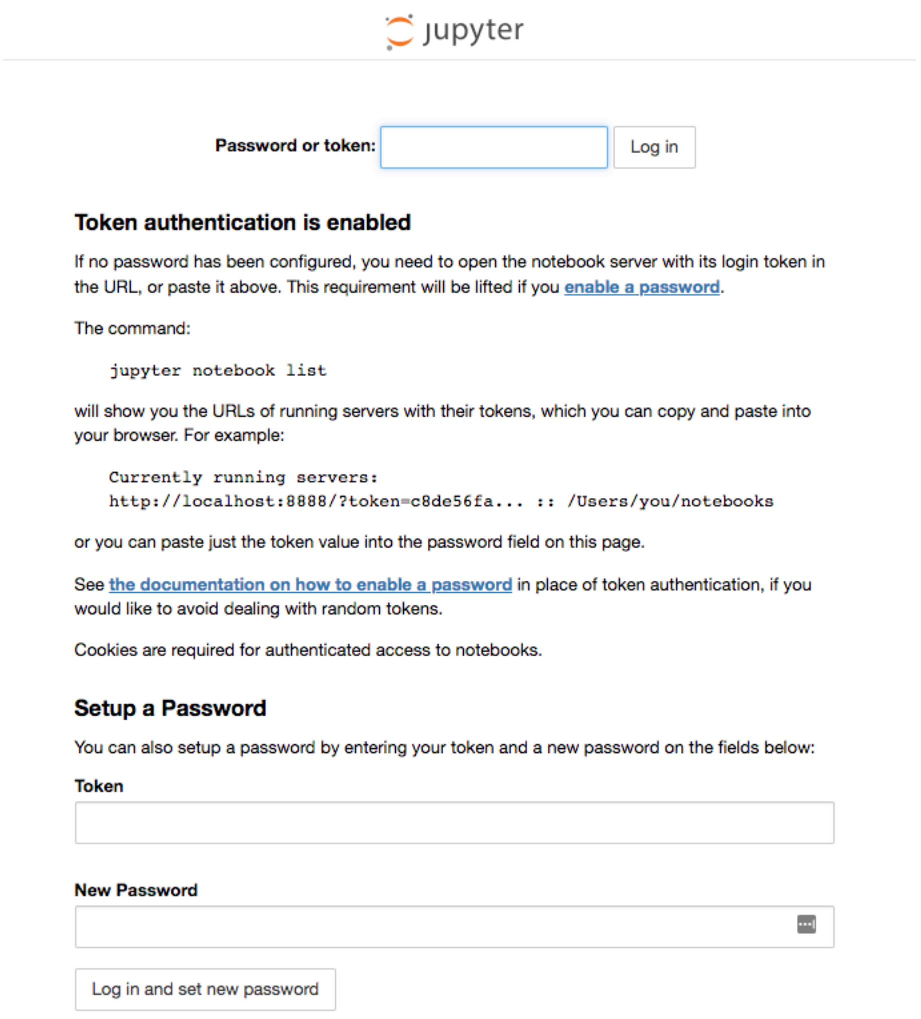
```
    sudo ./run.sh
```

The first time you do this it will take a while because Docker will have to contact a repository and download a bunch of stuff. Once it finishes downloading you’ll start to see logging output from Jupyter which will look something like:

```
 [I 21:58:53.140 NotebookApp] Writing notebook server cookie secret to
/home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
[I 21:58:54.158 NotebookApp] JupyterLab extension loaded from
/opt/conda/lib/python3.6/site-packages/jupyterlab
[I 21:58:54.163 NotebookApp] JupyterLab application directory is
/opt/conda/share/jupyter/lab
[I 21:58:54.211 NotebookApp] Serving notebooks from local directory:
/home/jovyan
[I 21:58:54.211 NotebookApp] The Jupyter Notebook is running at:
[I 21:58:54.211 NotebookApp] http://(1f3d3ec89f2c or
127.0.0.1):8888/?token=f0d85bea7fcb84781e51b99972e33f13478bd95a49fc1b8c
[I 21:58:54.211 NotebookApp] Use Control-C to stop this server and shut down
all kernels (twice to skip confirmation).
[C 21:58:54.211 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first
time,
    to login with a token:
        http://(1f3d3ec89f2c or
127.0.0.1):8888/?token=f0d85bea7fcb84781e51b99972e33f13478bd95a49fc1b8c
```

The interesting part of this is the authentication token. You’ll need that to log in. Send your browser to http://YOURAZUREVMSPUBLICIP:8888. You should see a page that looks like:



Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you enable a password.

The command:

```
jupyter notebook list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

```
Currently running servers:
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebooks
```

or you can paste just the token value into the password field on this page.

See the documentation on how to enable a password in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to notebooks.

Setup a Password

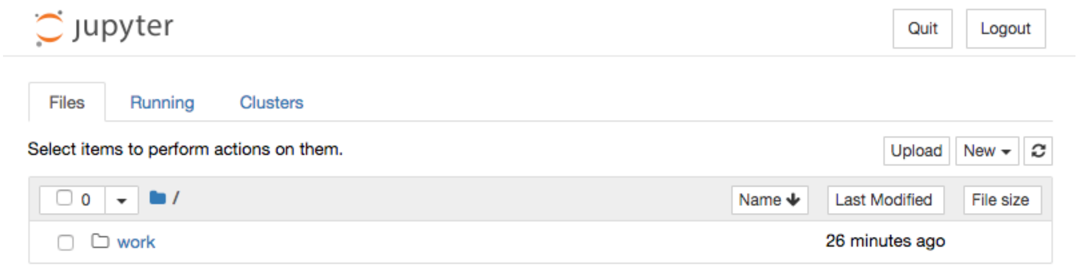
You can also setup a password by entering your token and a new password on the fields below:

The above page appears because we didn't set up Jupyter's authentication system to act like a grown up and are relying on a simple token-based scheme where it generates a secret every time it runs that you're expected to present to log in. The token you want is the long piece of hexadecimal GUID drivel in the above logging output, i.e. the red part of:

```
http://(1f3d3ec89f2c or
127.0.0.1):8888/?token=f0d85bea7fcb84781e51b99972e33f13478bd95a49fc1b8c
```

Copy whatever hexadecimal trash you got into the "Password or token" field and press 'Log In'.

Now you should see:



Click down into work/notebooks and then click JustEnoughScalaForSpark.ipynb

Once you've done that the notebook should appear. It's a mix of cells you can edit (with "In [NN]:") next to them, output cells (with "Out[MM]:" next to them) and text cells that contain human readable text. To execute the code in an input cell put your cursor in it and hit SHIFT+RETURN. Once the command in the input cell executes, its output will appear in the corresponding output cell.

Now make some coffee or whatever else gets you through the day and work through the notebook. It has a rhyming narrative structure of explanation, input cell with command, output cell, and commentary on what you just did that repeats throughout.

In brief it introduces you to:

- Why you might want to care about Scala in the context of Spark (or at all, generally)
- A few good sources on finding more information about Scala and Spark, including some links to online documentation that are worth bookmarking
- Jupyter notebooks in general
- Scala functions and methods
- Expressions vs statements (we alluded to this in class, but I deferred a full discussion to this notebook)
- Immutable val's vs. mutable var's
- Higher order functions
- Spark MapReduce: this time you can create an inverted index rather than just doing word count!
- Scala data types
- *Pattern matching* a Scala feature that can make programs very concise and clear

Work through the notebook in a leisurely and thoughtful way. If you have unresolved questions as you go, make a note of them and we can talk about them.

Part 2: Obtain and Prepare the Data Sets for This Week's Lab

For this assignment we are going to use two sample data sets. One is a text file version of the classic book "War and Peace", the other is the housing sales data from Week 2.

If you have not already done so, please put home_data.csv and war_and_peace.txt on to your VM. You can find them under “Resources” in Canvas along with the materials for Module 3.

To get the data onto your Azure VM you can download it to your local machine and then use the SCP command to copy the files from your local machine to your Azure VM, i.e.:

```
scp home_data.csv username@ip:                    (the trailing colon is necessary)
scp war_and_peace.txt username@ip:
```

If you would like to, you can instead add the data files to HDFS, and access them from there. To copy home_data.csv to HDFS you can scp into sandbox and then use HDFS commands to copy to HDFS, like so:

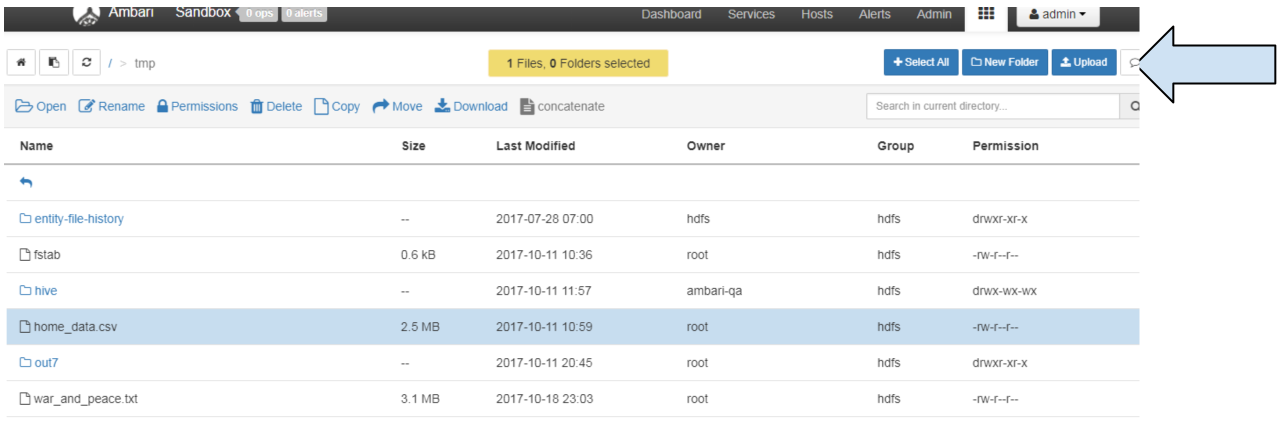
```
scp home_data.csv root@localhost:
```

And then SSH into the sandbox and run:

```
hadoop fs -put home_data.csv /tmp/home_data.csv
```

to put the file on the sandbox’s regular filesystem into HDFS.

Alternatively, you can use the Ambari Web UI “Files View” (available from the grid icon on the top Ambari toolbar, near where you found the “Hive2 View”) to upload directly from your local machine into HDFS in the sandbox via the web UI.



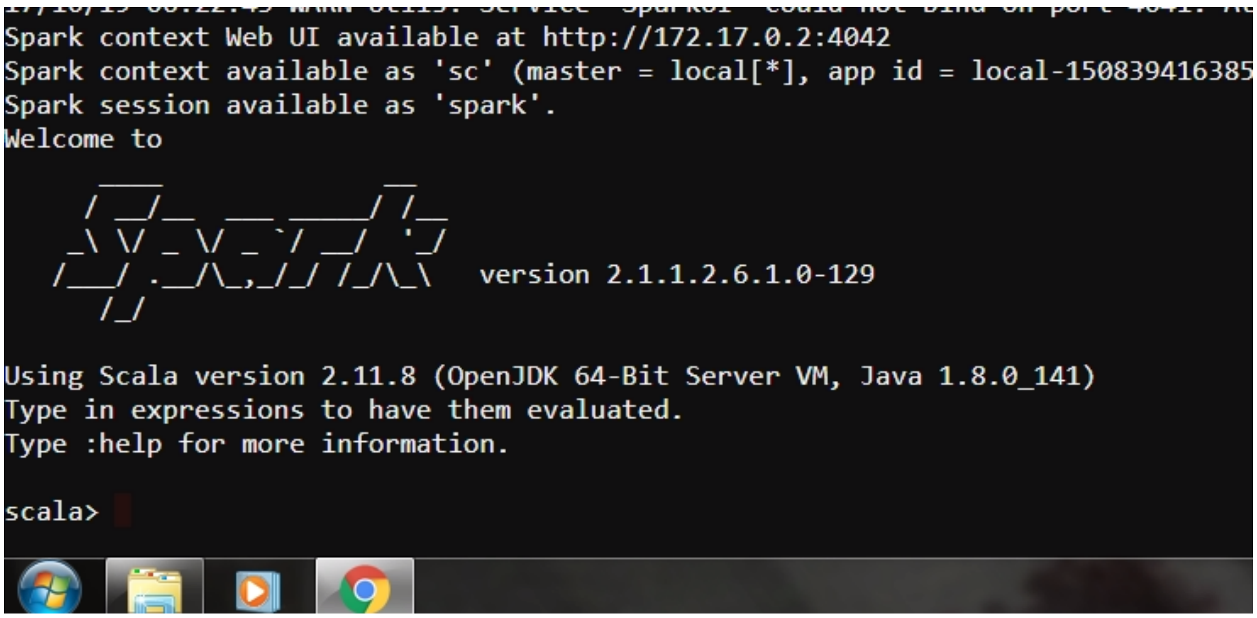
Name	Size	Last Modified	Owner	Group	Permission
entity-file-history	--	2017-07-28 07:00	hdfs	hdfs	drwxr-xr-x
fstab	0.6 kB	2017-10-11 10:36	root	hdfs	-rw-r--r--
hive	--	2017-10-11 11:57	ambari-qa	hdfs	drwx-wx-wx
home_data.csv	2.5 MB	2017-10-11 10:59	root	hdfs	-rw-r--r--
out7	--	2017-10-11 20:45	root	hdfs	drwxr-xr-x
war_and_peace.txt	3.1 MB	2017-10-18 23:03	root	hdfs	-rw-r--r--

Part 3: Run the Spark Shell

From within the sandbox (i.e. after you have done “ssh -p 2222 root@localhost”), run the command “`spark-shell`” (or “`pyspark`” for a Python shell if you prefer).

You can safely ignore warnings complaining that the “SparkUI can’t bind to port…”

If all goes well you should see a screen with some excellent ASCII art like this:



```
Spark context Web UI available at http://172.17.0.2:4042
Spark context available as 'sc' (master = local[*], app id = local-150839416385
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.1.1.2.6.1.0-129
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_141)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Congratulations, now you are ready to use Spark!

To get some familiarity and make sure Spark is working properly, I’d recommend running through a few of the examples from the slides this week. The airport codes data file is also up in Canvas if you want to play with it specifically.

Part 4: Assignment 3 Tasks

Please answer the following questions using the sample data provided, providing code that you used. Either Scala or Python is acceptable. If you are unable to come up with code to answer a question please describe how you think you would solve the problem in a “sparkified” way based on what we have learned so far.

You will likely find the String method “split” in both Scala and Python, and “contains” in Scala (“in” in Python) useful in many of these exercises. This week’s material shows some sample usage.

1) Which lines from the book war_and_peace.txt contain both words “war” and “peace”?
2) **Approximately** how many sentences are in the book war_and_peace.txt? HINT: Let’s define a sentence as a sequence of text that ends with a period and note that an approximation is ok.
3) Save a random sampling of 500 lines from war_and_peace.txt either to local or HDFS storage as text.

4) From home_data.csv, how many houses sold were built prior to 1979?
5) From home_data.csv, how many houses sold had a lot size (in sq. ft) that was greater than triple the living space (in sq. ft)? Save this output to file either locally or on HDFS. Example: lot size 500 living size 100 would count (500 > 300) but 500/200 would not (500 < 600).

Bonus Exercise

Remember, bonus exercise is meant for those either with prior knowledge on the topic or the interest (and time) to do some research beyond what we covered in class and is optional.

Exercise 1:

Let's again use the zipcode list from Assignment 2, you can access via

```
wget
'https://drive.google.com/a/uw.edu/uc?authuser=0&id=0B0Ntj7VtxrluSXhiak
dXLWx0N3c&export=download' -O wa_zipcodes.csv
```

How many homes were sold with a zipcode being defined as in "Seattle"?

Hint: how do you go about doing this in SQL?

Exercise 2:

Using home_data.csv create an RDD of key/value pairs where the key is the "id" of a row and the value is everything else.

For Example:
"7129300520","20141013T000000",221900,"3","1",1180,5650,"1",0,0,3,7,1180,0,1955,0,"98178",47.5112,-122.257,1340,5650

The key would be "7129300520" and the value would be "20141013T000000",221900,"3","1",1180,5650,"1",0,0,3,7,1180,0,1955,0,"98178",47.5112,-122.257,1340,5650

Hint: Most of the transformations we have done so far are simple one liners; But remember that you can use any arbitrary function that you define in your transformation, like we did with the square root example in the functional programming demo.

DON'T WORRY IF YOU DON'T GET ALL THE WAY THROUGH THE "PART 4: ASSIGNMENT 3" TASKS BEFORE LECTURE 4! We sort of have an outsized helping of work this week because of the technical glitches that delayed the Hive assignment and VM setup. If you get through the Scala tutorial and have a chance to play around with the examples from class, and just start the barest examination of this week's Spark exercises, you'll still be in fine shape for next week's lecture material, and you may even find these exercises easier with two weeks of Spark lectures under your belt. You can then mop up the Part 4 exercises after lecture 4.