Getting Spark, Pyt 1.6K | 49 pyter Notebook running on Amazon EC2

Step-by-step guide to getting PySpark working with Jupyter Notebook on an instance of Amazon EC2.

Hi! I’m Jose Portilla and I teach over 200,000 students about programming, data science, and machine learning on Udemy! You can check out all my courses here.

If you’re interested in learning Python for Data Science and Machine learning, check out my course here. (I also teach Full Stack Web Development with Django!)

This is mainly a guide for myself to reference again at a later date, but I figured it would be useful to others. There are a few guides similar to this one floating around, but a lot of them have not been updated in a while. This guide assumes some basic familiarity with the command line and AWS console.

Step 1: Create an Amazon EC2 Instance

For simplicity we will use the Free Tier Micro Instance using Ubuntu 64x. Use all the default settings, except set All Traffic instead of SSH when editing the Security of the instance. This guide can be expanded to multiple instances or larger instances.

Step 2: SSH into that instance (Windows)

For Windows users, you'll need to use PuTTY . Amazon has a really good set of instructions located here. Follow those to the point where you have the connection to the Ubuntu console.

Step 3: SSH into that instance (Linux/Mac OS)

An SSH client is already built into Mac and Linux (usually). You can just do a straightforward ssh command into your EC2 instance using your .pem file that you downloaded. Amazon also has instructions on this here.

You should now have successfully connected to the command line of your virtual Ubuntu instance running on EC2. The rest of the guide will tell you commands to put into this terminal.

Step 4: Download and Install Anaconda

Next we will download and install Anaconda for our Python. You can replace the version numbers here with whatever version you prefer (2 or 3).

```
$ wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-
x86_64.sh

$ bash Anaconda3-4.1.1-Linux-x86_64.sh
```

Press Enter through the license agreements, then Enter **yes** to accept, then Enter again to accept the default location.

Step 5: Check which Python you are using.

Ubuntu already comes with Python, but let's make sure we are using the Anaconda version. Use:

```
$ which python
```

Most likely you won't be using Anaconda's version of Python (you can tell by checking the output of the which python command). Change to the Anaconda version of Python by specifying your source using:

```
$ source .bashrc
```

Then confirm with:

```
$ which python
```

Or you can just type **python** and check to see.

Step 6: Configure Jupyter Notebook

Jupyter comes with Anaconda, but we will need to configure it in order to use it through EC2 and connect with SSH. Go ahead and generate a configuration file for Jupyter using:

```
$ jupyter notebook --generate-config
```

Step 7: Create Certifications

We can also create certifications for our connections in the form of .pem files. Perform the following:

```
$ mkdir certs
$ cd certs
```

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout
mycert.pem -out mycert.pem
```

You'll get asked some general questions after running that last line. Just fill them out with some general information.

Step 8: Edit Configuration File

Next we need to finish editing the Jupyter Configuration file we created earlier. Change directory to:

```
$ cd ~/.jupyter/
```

Then we will use visual editor (vi) to edit the file. Type:

```
$ vi jupyter_notebook_config.py
```

You should see a bunch of commented Python code, this is where you can either uncomment lines or add in your own (things such as adding password protection are an option here). We will keep things simple.

Press **i** on your keyboard to activate -INSERT-. Then at the top of the file type:

```
c = get_config()

# Notebook config this is where you saved your pem cert
c.NotebookApp.certfile = u'/home/ubuntu/certs/mycert.pem'

# Run on all IP addresses of your instance
c.NotebookApp.ip = '*'

# Don't open browser by default
c.NotebookApp.open_browser = False

# Fix port to 8888
c.NotebookApp.port = 8888
```

Once you've typed/pasted this code in your config file, press **Esc** to stop inserting. Then type a colon **:** and then type **wq** to write and quit the editor.

Step 9: Check that Jupyter Notebook is working

Now you can check to see that jupyter notebook is working. In your ubuntu console type:

```
$ jupyter notebook
```

You'll see an output saying that a jupyter notebook is running at all ip addresses at port 8888. Go to your own web browser (Google Chrome suggested) and type in your Public DNS for your Amazon EC2 instance followed by :8888. It should be in the form:

```
https://ec2-xx-xx-xxx-xxx.us-west-2.compute.amazonaws.com:8888
```

After putting that into your browser you'll probably get a warning of an untrusted certificate, go ahead and click through that and connect anyway, you trust the site. (Hopefully, after all you are the one that made it!)

You should be able to see Jupyter Notebook running on you EC2 instance. Great! Now we need to go back and install Scala, Java, Hadoop, and Spark on that same instance to get PySpark working correctly. Use Crtl-C in your EC2 Ubuntu console to kill the Jupyter Notebook process. Clear the console with **clear** and move on to the next steps to install Spark.

Step 10: Install Java

Next we need to install Java in order to install Scala, which we need for Spark. Back at your EC2 command line type:

```
$ sudo apt-get update
```

Then install Java with:

```
$ sudo apt-get install default-jre
```

Check that it worked with:

```
$ java -version
```

Step 11: Install Scala

Now we can install Scala:

```
$ sudo apt-get install scala
```

Check that it worked with:

```
$ scala -version
```

(Optional: You can install specific versions of Scala with the following, just replace the version numbers):

```
$ wget http://www.scala-lang.org/files/archive/scala-2.11.8.deb
$ sudo dpkg -i scala-2.11.8.deb
```

Step 12: Install py4j

We need to install the python library py4j, in order to this we need to make sure that pip install is connected to our Anaconda installation of Python instead of Ubuntu's default. In the console we will export the path for pip:

```
$ export PATH=$PATH:$HOME/anaconda3/bin
```

Then use conda to install pip:

```
$ conda install pip
```

Confirm that the correct pip is being used with:

```
$ which pip
```

Now we can install py4j with pip:

```
$ pip install py4j
```

Step 13: Install Spark and Hadoop

Use the following to download and install Spark and Hadoop:

```
$ wget http://archive.apache.org/dist/spark/spark-2.0.0/spark-2.0.0-
bin-hadoop2.7.tgz

$ sudo tar -zxvf spark-2.0.0-bin-hadoop2.7.tgz
```

Step 14: Tell Python where to find Spark

Finally we need to set our Paths for Spark so Python can find it:

```
$ export SPARK_HOME='/home/ubuntu/spark-2.0.0-bin-hadoop2.7'
$ export PATH=$SPARK_HOME:$PATH
$ export PYTHONPATH=$SPARK_HOME/python:$PYTHONPATH
```

Step 15: Launch Jupyter Notebook

You should now have everything set up to launch Juptyer notebook with Spark! Run:

```
$ jupyter notebook
```

Then as previously done, connect through your browser again to your instance's Jupyter Notebook. Launch a new notebook and in a notebook cell type:

```
from pyspark import SparkContext
sc = SparkContext()
```

If that works, you're all done!

Hopefully you found this useful! (Check out the various comments on this article for other useful tips by readers!) Thanks for reading.