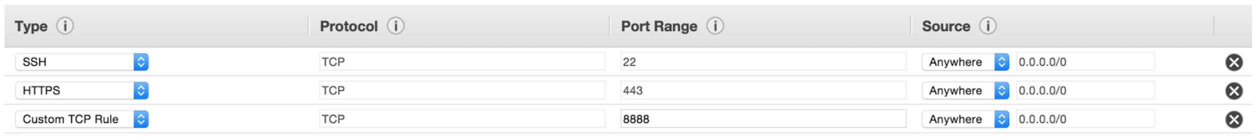
Setting up Python environment

1. Create and launch an AWS EC2 instance, one important note is to have the 8888 port opened in the security group settings as shown in the image below.



Type	Protocol	Port Range	Source	
SSH	TCP	22	Anywhere	0.0.0.0/0
HTTPS	TCP	443	Anywhere	0.0.0.0/0
Custom TCP Rule	TCP	8888	Anywhere	0.0.0.0/0

2. After logging on to the EC2 instance, we are going to use the Anaconda Distribution for installing all the require Python libraries.

   $ wget https://3230d63b5fc54e62148e-c95ac804525aac4b6dba79b00b39d1d3.ssl.cf1.rackcdn.com/Anaconda2-2.4.1-Linux-x86_64.sh

3. Use bash to run this .sh file. You need to accept the license terms and set the installation directory for the anaconda distribution. I use the default one only, which is "”root/anaconda2/".

   $ bash Anaconda2-2.4.1-Linux-x86_64.sh

4. Check which version you are using.

   $ which python

   If it does not mention the one from “…/anaconda2/…” folder, then you can use the following command to reload your .bashrc

   $ source .bashrc

5. Open the iPython Terminal to get an encrypted password so as to use it for logging into our iPython Notebook Server. Remember to copy and save the output of this command, which will be an encrypted password, something like "sha1..."

```
$ ipython

In [1]:from IPython.lib import passwd

In [2]:passwd()
```

6. Now we're going to create the configuration profile for our Jupyter Notebook server and create a self-signed certificate for assessing our notebooks through HTTPS

```
$ jupyter notebook --generate-config
$ mkdir certs
$ cd certs
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
```

7. It will ask some questions, please answer them to the best of your ability. Next, change the config settings of our server

```
$ cd ~/.jupyter/
$ vi jupyter_notebook_config.py
```

You will see a long list of configuration settings. Add the following snippet at the top of the page, and change the password to the one you saved earlier.

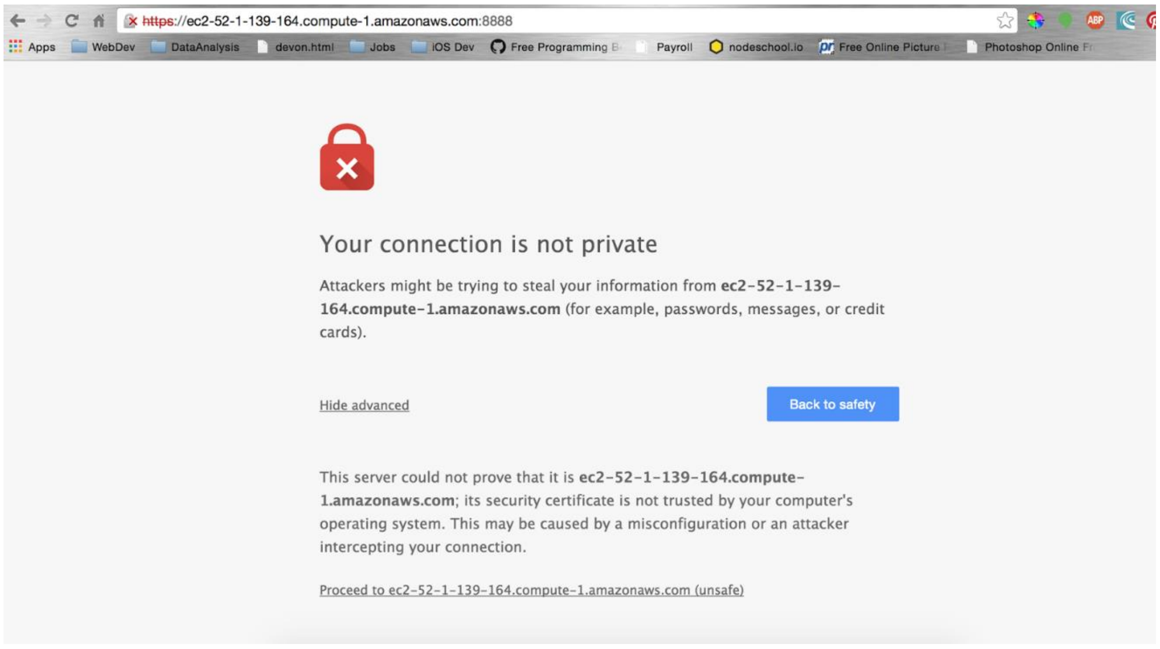
```
c = get_config()

# Kernel config
c.IPKernelApp.pylab = 'inline'  # if you want plotting support always in your notebook

# Notebook config
c.NotebookApp.certfile = u' root/certs/mycert.pem' #location of your certificate file
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False  #so that the ipython notebook does not opens up a browser by default
c.NotebookApp.password = u'sha1:68c136a5b064...'  #the encrypted password we generated above
# It is a good idea to put it on a known, fixed port
c.NotebookApp.port = 8888
```

8. Almost there. We will now create a directory to store all of our future notebooks.
   $ cd ~
   $ mkdir Notebooks
   $ cd Notebooks

9. Start notebook server
   $ jupyter notebook

10. Connect to notebook server through HTTPS. At your browser bar, enter:
    https://<your EC2 address>:8888

    You will be asked by the browser to trust the certificate.



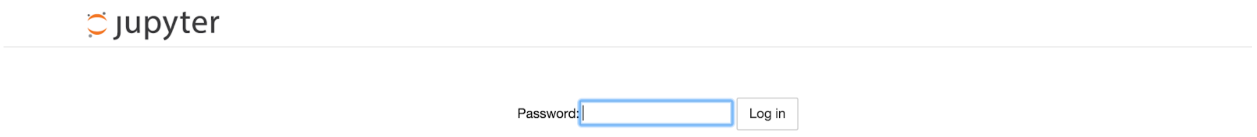
Your connection is not private

Attackers might be trying to steal your information from **ec2-52-1-139-164.compute-1.amazonaws.com** (for example, passwords, messages, or credit cards).

This server could not prove that it is **ec2-52-1-139-164.compute-1.amazonaws.com**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to ec2-52-1-139-164.compute-1.amazonaws.com (unsafe)

    Enter password and it should be good to go



Launching Postgres

1. Attach EBS volume that contains the data to EC2 instance.

2. Mount volume.
   $ mount –t ext4 /dev/xvdf /data

3. Start Postgres
   $ /data/start_postgres.sh

4. Connect to Postgres server and the ontime database
   $ psql –U postgres –d ontime