http://blog.impiyush.me/2015/02/running-ipython-notebook-server-on-aws.html

1. Login to your Amazon Management Console. If you don't have an account yet, you can create one for it. You get 1 yr of free access to some of the services, which you can check out at this link

2. Create a new EC2 Instance with Ubuntu. If you are not familiar with how to create an EC2 instance, you can check out the video of this blog, in which I go through the steps from the beginning.

3. The important thing to remember while creating the instance is to assign the security group settings as mentioned in the image below

4. Launch your instance and ssh into it to perform the following operations

- First of all we are going to use Anaconda Python Distribution for installing all the required Python libraries. This is a free distribution and we are going to use the Linux version of it. Remember to verify the latest version of the distribution from the site. This blog is updated to reflect the changes in the latest Anaconda distribution - 2.4.1.

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

- Next we will 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 "/home/ubuntu/anaconda2/". Also, it asks you to add the default path of anaconda python to your **.bashrc** profile. You can accept to do it or add it manually.

```
$ bash Anaconda2-2.4.1-Linux-x86_64.sh

#here we should update anaconda, conda, and all the packages and then set again the installation directory
```

- Now you need to check, which python version you are using, just to confirm if we are using the one from Anaconda Distribution or not. You can do this by using

```
$ which python
```

  This will list the python your system is currently using. If it does not mentions the one from ".../anaconda2/..." folder, then you can use the following command to re-load your **.bashrc**, so as to set the correct python

```
$ source .bashrc
```

- 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()
```

  and exit out of ipython terminal using "exit" command. [I'm not gonna use this password(shown in the pic below), so don't waste your time trying to copy and use it. ;)]

- Now we're going to create the configuration profile for our Jupyter Notebook server

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

- The next thing is going to be to create a self-signed certificate for accessing our Notebooks through HTTPS

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

  it will ask some questions, please answer them to the best of your knowledge as some of them are required to successfully create the certificate.

-

- It's time to change the config settings of our server

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

  You will see a long list of configuration settings. You can go through each one of them and uncomment them as you like, but for me I know what I want, so I'll add the following settings to the top of the file and leave the rest commented as it is.

```
c = get_config()

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

# Notebook config
c.NotebookApp.certfile = u'/home/ubuntu/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
```

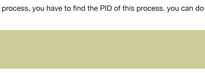
- We are almost done. Now its time to start our Jupyter notebook server. For this, first I'll create a new folder which will store all my notebooks

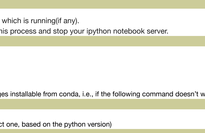
```
$ cd ~
$ mkdir Notebooks
$ cd Notebooks
```

  and now I'll start my notebook server

```
$ jupyter notebook
```

5. And that is all. Now you can access your Notebook from anywhere through your browser (use Firefox!!!). Just navigate to the DNS name, or Public IP of your instance, along with the port number. (By default, the browser adds "http" to the url. Please remember to update it to "https")
You will be asked by your browser to trust the certificate, as we have signed it on our own, so we know we can trust it. See images for reference below:





6. Login, using the password you specified when you used the iPython Terminal to create an encrypted version of it and you are good to go.

7. One thing to note is that if you close the ssh access to your instance, your notebook server will not work. To keep it working, even when you close the ssh access to the server you can use the following command

```
$ nohup jupyter notebook
```

This will put your server process as no-hangup and will keep it running even if you close the ssh access to the instance

8. Later, if you decide you want to stop this process, you have to find the PID of this process. you can do so by going into your notebooks folder and using the command

```
$ lsof nohup.out
```

which will list the PID of the nohup process which is running(if any).
Then you can use the kill command to kill this process and stop your ipython notebook server.
`$ kill -9 PID`

INSTALLING PACKAGES

if a package is not present in the list of packages installable from conda, i.e., if the following command doesn't work
`$ conda install geopy`

install the package directly from pip (the correct one, based on the python version)
`$ cd /Users/Andrea/anaconda/bin`
`$ ./pip install geopy`

or alternatively, if we're working with python3
`$ ./pip3 install geopy`

INSTALLING PYSPARK AND LINKING IT TO PYTHON

download the latest pre-built version of spark from here https://spark.apache.org/downloads.html

transfer it in the virtual machine with the command (substituting the appropriate informations, i.e., cert name, file name, and url of the vm)
`$ scp -i "linux.pem" spark-2.0.0-bin-hadoop2.7.tar ubuntu@ec2-54-68-181-162.us-west-2.compute.amazonaws.com:~/`

unpack it (see http://www.shellhacks.com/en/HowTo-Extract-untar-tar-tarqz-and-tarbz2-Files)

`$ tar -xvf spark-2.0.0-bin-hadoop2.7.tar`

update path and link pyspark to ipython
`$ vim .bashrc`

insert the following lines (setting the correct path in case it is different)

```
export SPARK_HOME=/home/ubuntu/spark-2.0.0-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin
export PYSPARK_DRIVER_PYTHON=ipython
export PYSPARK_DRIVER_PYTHON_OPTS='notebook' pyspark
```

re-load the .bashrc
`$ source .bashrc`

start the notebook server with pyspark
`$ pyspark`

INSTALLING A VIRTUAL ENVIRONMENT IN ANACONDA FOR PYTHON3

*****with only 1GB of ram (free tier) I was not able to complete this procedure because the first command gave a memory error*****

to set up a virtual environment called py3k with python3
`$ conda create -n py3k python=3 anaconda`

to remove it
`$ conda remove -n py3k --all`

subsequently, to activate it
`$ source py3k`

and to deactivate it
`$ source deactivate`

to install packages through pip, instead of conda (anaconda)
`$ cd /Users/Andrea/anaconda/envs/py3k/bin`
`$ ./pip install worksalendar`

to visualize the packages installed in anaconda
`$ conda list`