The Data Science Toolkit

The purpose of this document is to provide a detailed guide to help in installing and configuring a Jupyter Data Science Notebook Server on Amazon Web Services (AWS).

It also discusses the following at the end:

* Jupyter Notebook Security Concerns
* Cost of running a notebook server for 3 months

# **Architecture of the System (AWS, Docker, Jupyter)**

Here is a little diagram to help understand the system.

Applications/database running in docker

(eg., Jupyter, Postgres)

Docker Engine

Shell

Operating System

Operating System (Ubuntu)

Hardware

Hardware

LOCAL AWS

# **Pre-requisites:**

Before we get into the nitty-gritty of setting up/configuring the Jupyter Data Science Notebook, we need to go through some basic set up.

1. Create Amazon Web Services (AWS) Account

Create your AWS account at <https://aws.amazon.com/> if you don’t already have one. Note that you will need to provide a phone number to authenticate and complete setting up the account.

1. Install Bash Shell

On the Window Operating System, please download and install git-bash from <https://git-scm.com/downloads> .

On Mac Operating System, no installation is needed, as this is available native. It is the Mac Terminal shell available in the Utilities folder.

On the Linux OS, this is available native as well.

1. Generate Secure Keys

Launch your Bash Shell and follow the steps below to generate secure public/private rsa key pair that will be used to authenticate/connect to our AWS instance. If the rsa key pair already exist at ~/.ssh/, then please skip this step, as you do need to regenerate it.

Step 1: In your home directory, create a directory .ssh with command “mkdir .ssh”.

Step 2: Navigate into this .ssh directory, with command “cd .ssh”.

Step 3: Generate secure keys with command “ssh-keygen”. This command will create 2 files, a public key (id\_rsa.pub) and a private key (id\_rsa). Please verify these files were generated with command “ls” to list the files.

1. Get GitHub

Github is a Version Control and Source Code Management System.

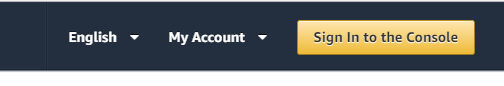
Create a GitHub account at [http://github.com](http://github.com/), and fork our class repo at  [https://github.com/joshuacook/UCLA\_CSX\_450\_1\_2018\_W (Links to an external site.)](https://github.com/joshuacook/UCLA_CSX_450_1_2018_W)To fork, just click the “Fork” button on the top right corner. Forking creates a copy of the project repository in your personal git-hub account. Any changes to your copy will not affect the original class copy.

# **Configure EC2 Instance on AWS**

Please follow the below steps to configure EC2 on Amazon Web Services (AWS).

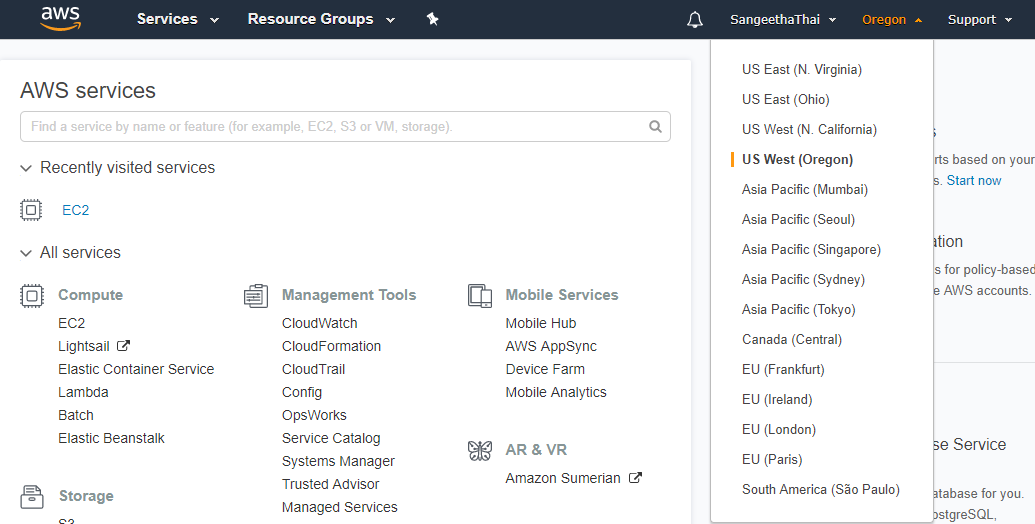
**Step1:** Login to AWS.

Go to <https://aws.amazon.com/>, click on “Sign In to the Console” and login.



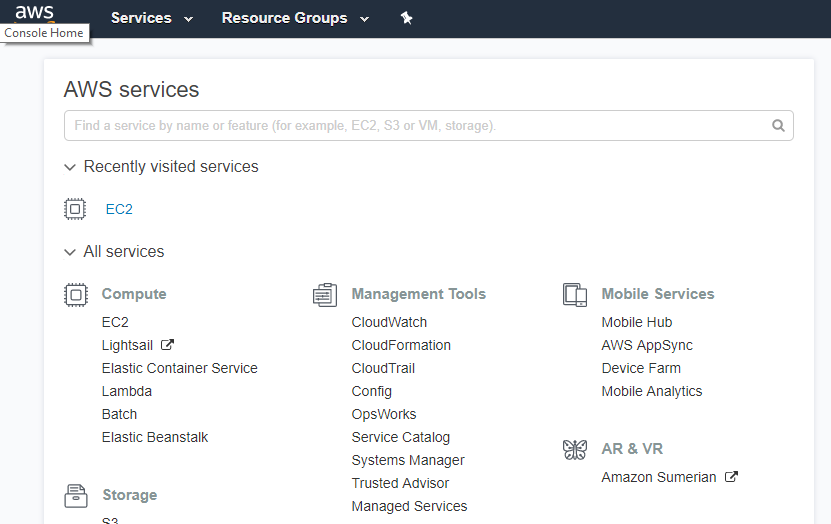
**Step 2:** Select the region/location at which you would like to set up your instance.

Region dropdown is located on the top right corner. Select the region that is closest to your location or the location from which the services are going to be accessed. Closer the location, faster the service.



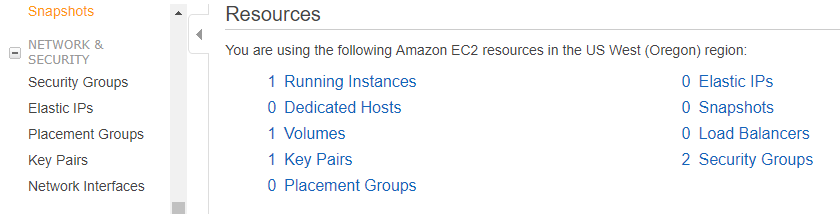
**Step 3:** Select a Service Type. Since Data Science is computation intensive, lets select a Compute Service.

Let’s select EC2. EC2 stands for Elastic Compute Cloud. EC2 is secure and as the name suggests, resizable compute capacity in the cloud. EC2 is simple to configure and get an instance running in minutes. And importantly, it allows you to scale your capacity up and down as your computing needs change. And you will be billed only for what you use.



**Step 4:** Create Key Pair

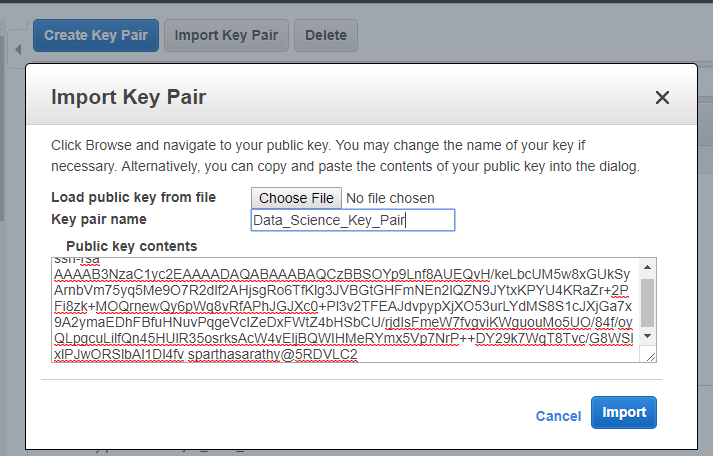
In the EC2 Dashboard, you will find the link to “Key Pairs” under **Resources** as well as on the left menu under **Network & Security**. Click the link to navigate to Key Pairs page.



Click on “Import Key Pair”

* Enter a Key Pair name
* Go to Bash shell location ~/.ssh/ and copy the contents of **id\_rsa.pub** and copy it into the “Public key contents” section.

Click “Import”, to save your Key Pair.



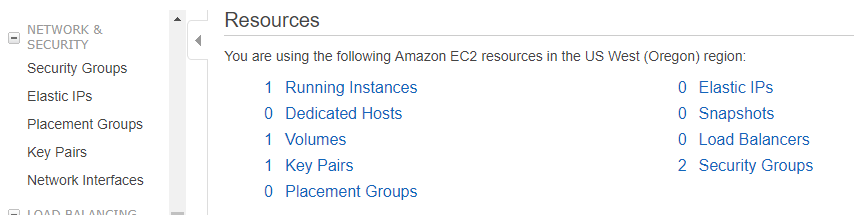
When EC2 instance is created, it will use the public key. Bash Shell will be able to connect/login to the EC2 instance using the private key.

Note that Key Pairs created in one region will not be available in another.

**Step 5:** Create a Security Group

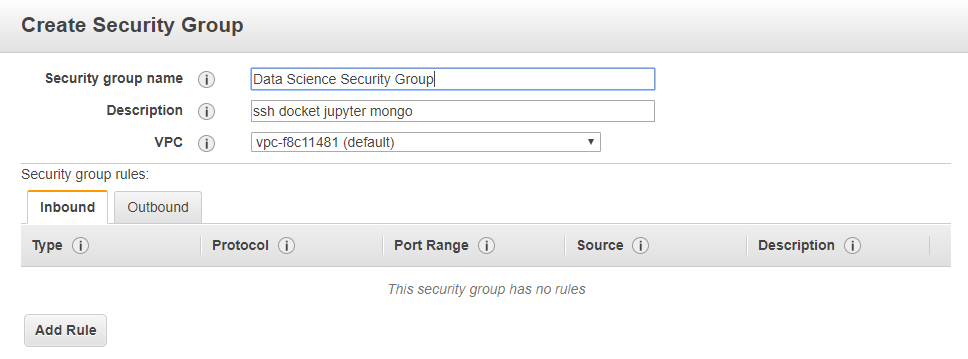
A Security Group allows us to define a set of rules to control access to the EC2 instance.

In the EC2 Dashboard, you can find links to “Security Groups” under **Resources** as well as in the left menu under **Network & Security**. Click on “Security Groups” to begin configuring a Security Group



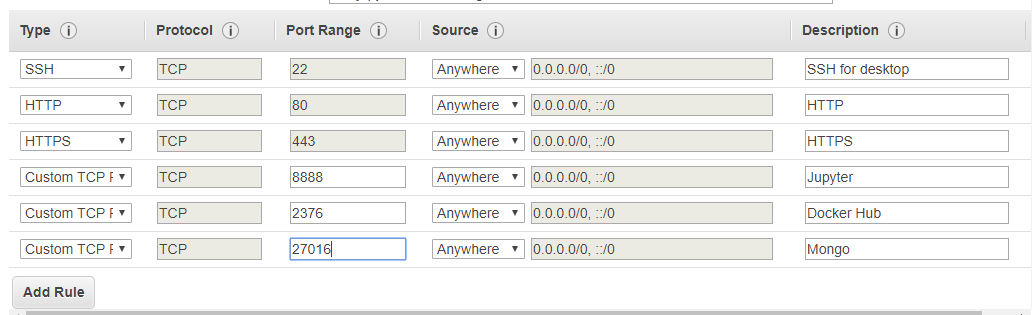
Let’s create a new security group.

* Click “Create Security group” and give it a Name and Description.



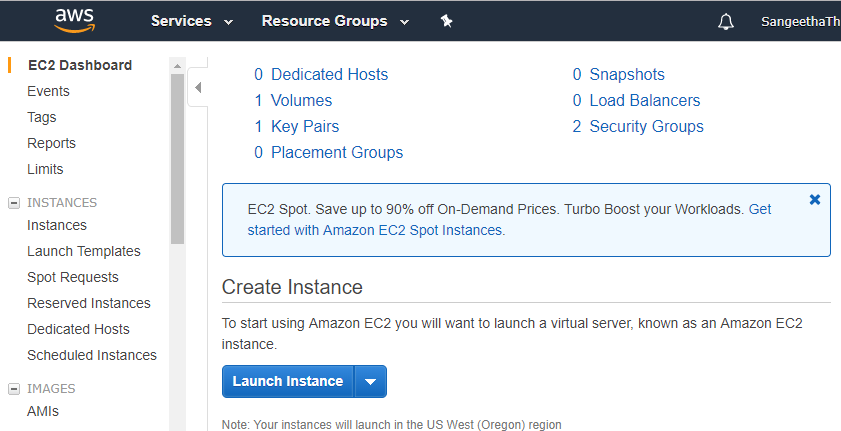
* In the Inbound tab, click “Add Rule” and add rules for services as in the screen shot below.

Notice that the Source is set to “Anywhere”.



**Step 6:** Create EC2 Instance (Virtual Server)

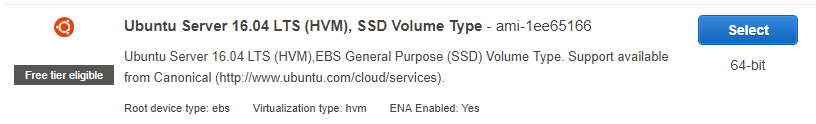
Click on “Launch Instance” button, to create your EC2 instance.



**Step 5:** Choose an Amazon Machine Image (AMI)

To Launch an EC2, we need an Operating System (OS), Application Server (App Server) and Applications (Apps). An AMI is a template that is preconfigured with all of these (OS, App. Server and Apps).

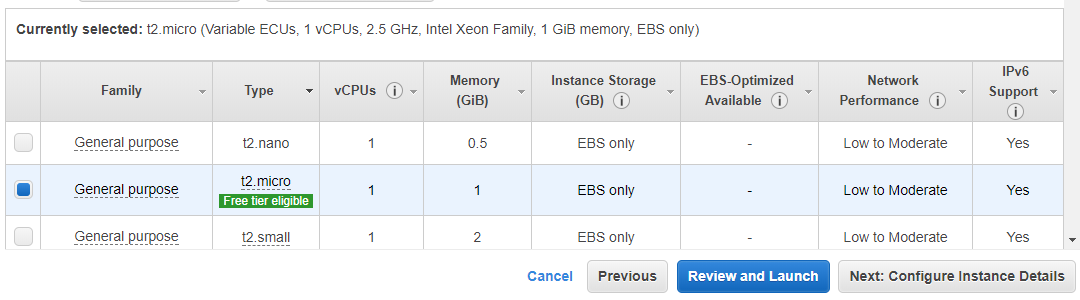
From the available list of AMIs, “Select” Ubuntu Server.



**Step 7:** Choose an Instance Type.

Each Instance Type contains varying combinations of CPU, Memory, Storage, Network capacity. Choose the mix that is right for you.

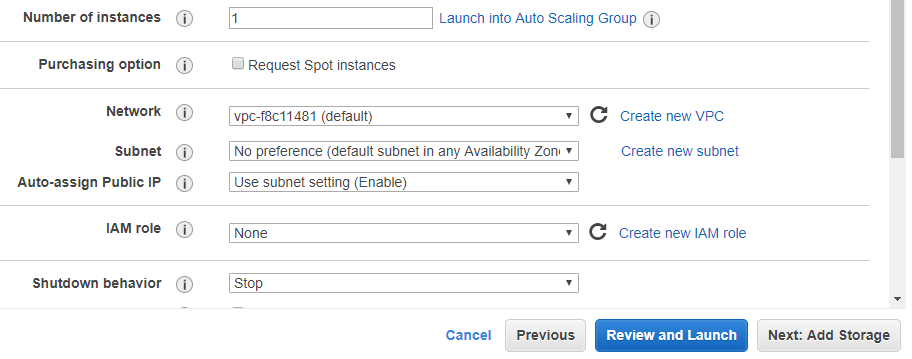
Let’s select “t2.micro” for our Instance Type and click “Next” to start configuring the Instance details



**Step 8:** Configure Instance Details

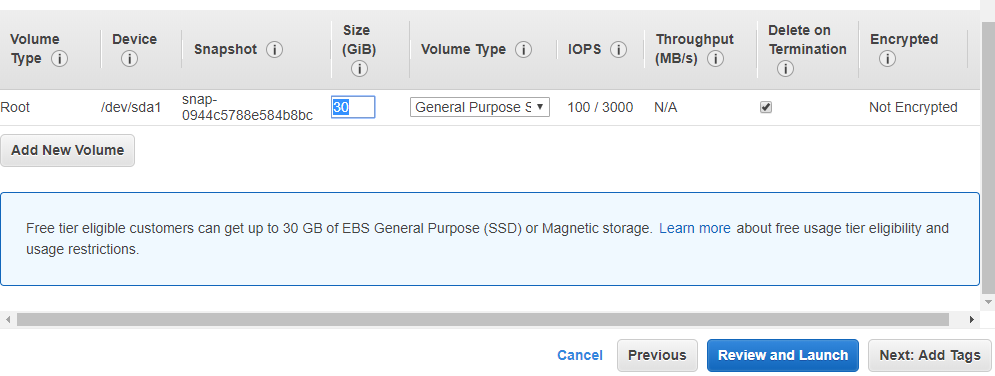
This where you choose how many instances you want to run, your network, shutdown behavior, etc. Notice that all fields are populated with default configuration.

Let’s go with the default configuration and click “Next” to move on to add storage.



**Step 9:** Add Storage

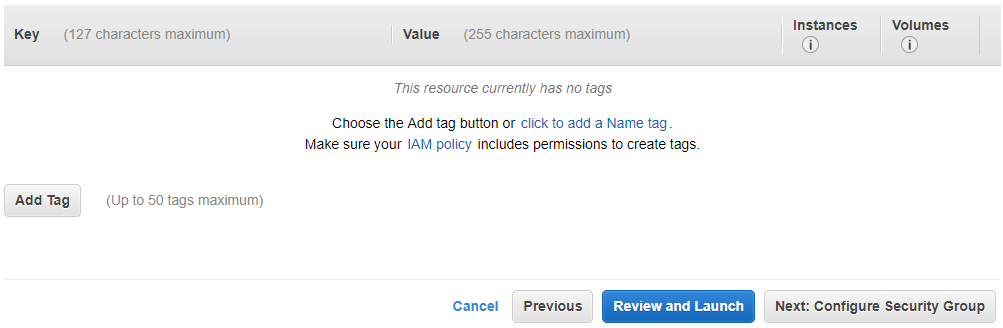
Increase the Storage size to 30 GB. Since we are going to be dealing with Big data, we need to make sure we have adequate storage space to accommodate all the data. Let move on to the next section. Click “Next”.



**Step 10:** Add Key Pairs

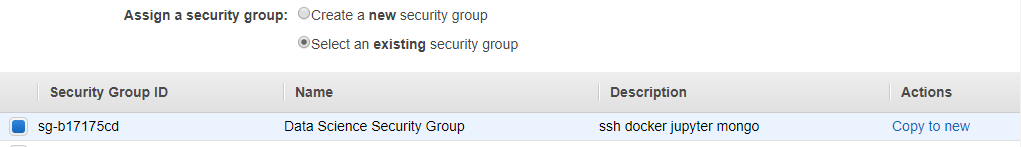
This section is used to define key/value pairs that can be applied to all instances and volumes.

We are going to skip this section, and click “Next” to move on.



**Step 11:** Configure Security Group

Choose “Select and existing Security Group” and select the security group we created in the beginning.

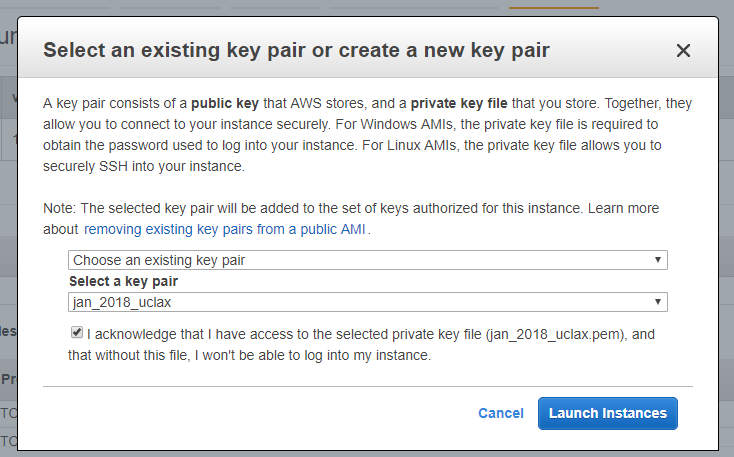


Click “Review and Launch”

**Step 12:** Review your EC2 Instance configuration and click “Launch”.

This will bring up a modal window to “Select or Create a Key Pair” (see below).

* From the dropdown select “Choose an existing key pair”
* Select the Key Pair you created in the beginning
* Check the acknowledgement
* And finally click “Launch Instances”



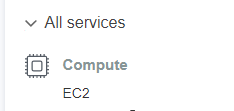
**Step 13:** Check you newly created instance in the list of running instances.

If you are not automatically directed to the list of instances, then

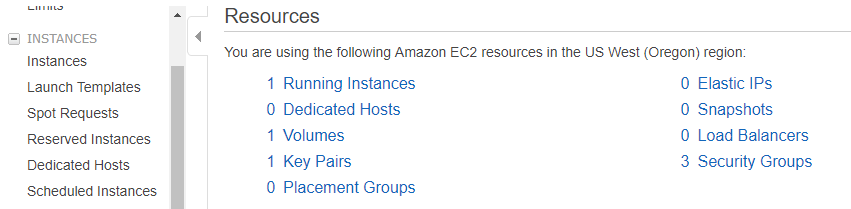
* click on the AWS Smile on the top menu, to access the EC2 dashboard



* Select EC2 Service

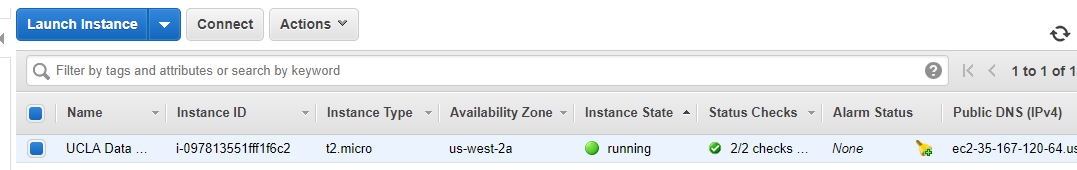


* In the EC2 Dashboard, click on either “Running Instances” under **Resources** or “Instances” under **Instances** in the left menu.



**Running Instances**

You should now see your instance running in the Running Instances section.

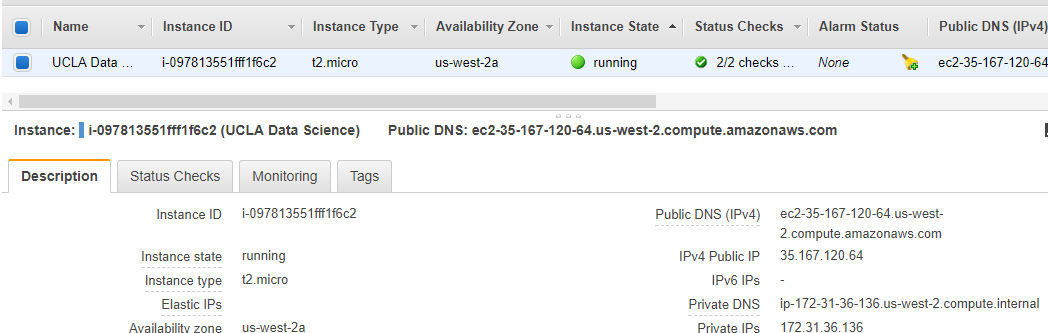


# **Configure your EC2 Instance using Docker**

Let’s login to our EC2 Instance and download docker.

1. Get your public IP address from the EC2 Instance you created

Go to **Running Instances** and select your instance. Note down the IP Address from the description section.

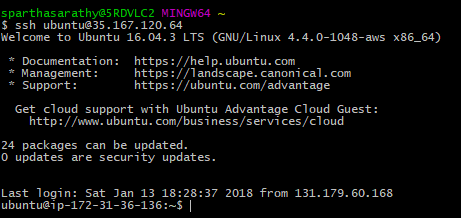


1. SSH into your EC2 Instance

Go to Bash shell and use the command, ssh [username@<public.ip.address](mailto:username@%3cpublic.ip.address)> to securely login to your EC2 Instance

We chose Ubuntu as our Operating System, and ubuntu’s default user is ubuntu.

So, translating the above command, it is  **ssh** [ubuntu@35.16.120.64](mailto:ubuntu@35.16.120.64)



If you successfully ssh into your instance, you should be at user@ip-private prompt

1. Install Docker

Use curl to download docker and pass it to shell.

The command to do so is below.



Once docker installation is complete, you can check the installation with the command, **docker -v**



1. Add ubuntu user to docker group

By default, the docker command line client requires sudo access to issue commands to docker daemon. To allow ubuntu to issue docker commands without sudo, let’s add ubuntu user to docket group. To add ubuntu user to docker group use the command,

**sudo usermod -aG docker ubuntu**

1. Reboot/Reconnect for changes to take effect

To disconnect, just use command **exit** and log back into your instance.

# **Configure Jupyter Data Science Notebook**

1. Get Jupyter Data Science Notebook

From Bash shell,

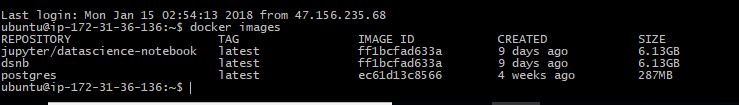
* + connect to our EC2 Instance

use command: **ssh ubuntu@ip-address**

* + get Jupyter Data Science Notebook image

use command: **docker pull jupyter/datascience-notebook**

* + Use command **docker images** to check Jupyter Data Science Notebook image we just pulled



1. Create the docker container

To create the docker container use the following command:

**docker run -v /home/ubuntu:/home/jovyan -p 8888:8888 -d jupyter/datascience-notebook**

where, **-p** option links amazon port to local port

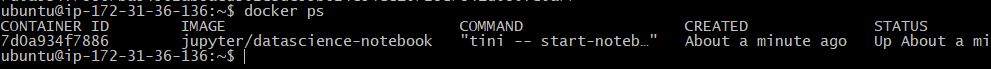
**-d** option stands for detached, i.e., the process runs in the background

On successful creation of the docker container, the command output the Container Id of the docker container. (see below)



You can view the docker container you just created with the command **docker ps** which displays all processes. Screenshot below shows the id of the docker container we created above.

Note, Container is an instance of an image. What we see below is an instance of Jupiter image we pulled above.

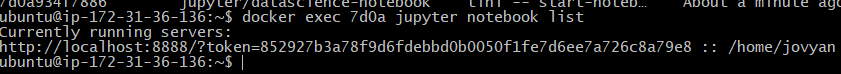


1. Get the Jupyter Data Science Notebook server up and running

To get the Notebook server up and running, execute the command below.

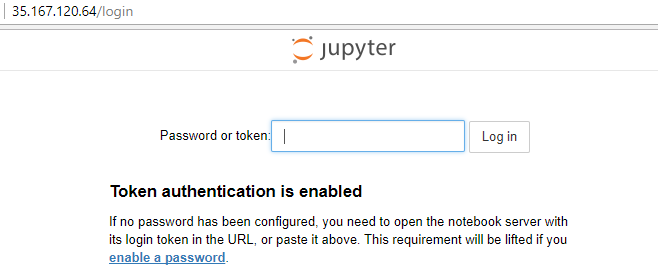
**docker exec <first-4-letters-of-container-id> Jupiter notebook list**

On executing the command, it will display a url and token of the notebook that can be used to login to the Jupiter Notebook Server.



Launch a browser and type in your EC2 instance’s public IP address. This will display your Jupiter Data Science Notebook.

Use the token to login to your notebook server on the website.



# **Jupyter Notebook Security Concerns**

1. In token based authentication, the token can be passed via query parameter, which hackers can steal and gain access to our system.
2. Once the user logs into the Jupyter Notebook Server, it creates a cookie on the users local machine and stores the password in the cookie. This makes it vulnerable to hackers.
3. Jupiter Notebook Server’s web interface provides access to terminals. Hackers can launch the terminal and execute all system commands like a logged in user.
4. Attackers can run arbitrary code from within the notebook and attack the system.

# **Cost of running a notebook server for 3 months**

On-demand pricing (<https://aws.amazon.com/ec2/pricing/on-demand/>) for EC2 lets you pay for only running instances and pay for computing capacity by the hour or second without any long-term commitments. So choose to pick this pricing plan.

Based on this plan, for region Oregon, here is my budget:

|  |  |  |  |
| --- | --- | --- | --- |
|  | T2.nano | T2.small | T2.medium |
| CPU | 1 | 1 | 2 |
| Memory (GB) | 0.5 | 2 | 4 |
| Cost/hr | $0.0058 | $0.023 | $0.0464 |
| Cost for 3 months:  90 days \* 24 hrs = 2,160 hrs | $12.528 | $49.68 | $100.224 |