

1. SSH keys & Amazon EC2

- 1) Open your browser --- Go to Amazon Web Services (<https://aws.amazon.com/console/>) --- Create an AWS account --- Sign in AWS --- Select “EC2” under “Compute” in All services list
- 2) Check “Key Pairs” under “Resources” panel, if it shows “0 Key Pairs”, please set up a one locally.
- 3) Open your Bash shell --- Type “ssh-keygen” --- Hit “enter” --- Save it into the default location without adding a passphrase --- Hit “enter” twice, you’ll see 2 files “id_rsa” and “id_rsa.pub” saved in “ssh” folder. (See below screenshot)

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/xiaoliang/.ssh/id_rsa.
Your public key has been saved in /Users/xiaoliang/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:HIpZv5TBe3HQf9pSEWK4uo+yX2kGxCvWX4Eb0BMPTH4 xiaoliang@xiaos-mbp-2.lan
The key's randomart image is:
+----[RSA 2048]-----+
|
|  o+...o  .
|  .o.=o+  .
|  .+B.Eoo  .
|  +.o==+o o o
|  o .o5+o.  . =
|  ...+o o o .
|  . . * .
|  . . =
|  .+O..
+-----[SHA256]-----+
```

- 4) Retrieve the content from “id_rsa.pub” --- type “cat ~/.ssh/id_rsa.pub”, hit “enter” --- You’ll see the key in the shell --- Copy the key.
- 5) Go back to AWS --- Click “Key Pairs” --- Click “Import Key Pair” --- Name the Key Pair --- Paste the key to “Public Key Contents” --- Click “Import”. (See below screenshot)

Import Key Pair ✕

Click Browse and navigate to your public key. You may change the name of your key if necessary. Alternatively, you can copy and paste the contents of your public key into the dialog.

Load public key from file Choose File No file chosen

Key pair name

Public key contents

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDRl2UYiJ/KADavHTv1y7zJrqUpGoam9cR1QH4n
MAXEgpMMDaFspkIEqW+isBzSV4oeALRyj1nBUfMWsZvAwK938wEdtXZkuzy+JRYbnejDldQ
pau45mWZFfkGUQUIDN2g/Z7XitRZjOvxZGYDCD6VprBjAn7JQx+hiqZWTJ4W0rh2pkBosexM
nP3GS42TjVTA3Dt1hgwla6/+fON6AjlEwMlm3eWbZIIypodzvNt2wOAVPr1R4u0eX4u/4fDuFk
Pv/1Uis+Lv86PKLaqddTSxhMBLcLAicFgrdhtuuoyad27roADMWs6R6H6x8TQBZ5En0HRokrV
nWuWvGvmeR4dN xiaoliang@xiaos-mbp-2.lan
```

Cancel Import

2. Security Groups

- 1) Go back to EC2 Dashboard --- Select “Security Groups” --- “Create Security Group” --- Click “Add Rule” --- Select “SSH” under “Type” and select “anywhere” under “Source” --- Add another rule, select “HTTP” under “Type” and select “anywhere” under “Source”.
- 2) Name the Security Group --- Add “Description” --- Click “Create”. (See below screenshot)

Create Security Group

Security group name

basic_jupyter

Description

SSH_HTTP

VPC

vpc-5ecf1a27 (default)

Security group rules:

Inbound

Outbound

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Add Rule

Cancel

Create

3. AWS Operating System

- 1) Go back to AWS Dashboard --- "Launch Instance" --- Choose Amazon Machine Image, select "**Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** - ami-1ee65166".
- 2) Choose an instance type, select "t2.micro (free tier eligible)" --- Click "Next: Configure Instance Details".
- 3) Use default settings and click "Next: Add storage".
- 4) Change "Size (GiB)" to "30" which is free to customers and click "Next: Configure Security Group".
- 5) Select an existing security group (the one you just created), make sure port 22 and 80 are there.
- 6) Review and Launch, you can disregard the warning. If everything looks good, then click "Launch". --- Choose an existing key pair, select the one you just created --- Launch Instances and view Instances.
- 7) Name the instance, check the inbound rules and make sure ports 80 and 22 are there. You'll also see a generated IP address.

Instance state

running

Instance type

t2.micro

Elastic IPs

Availability zone

us-east-1a

Security groups

sg-1a2b3c4d

2.compute.amazonaws.com

IPv4 Public IP 52.26.175.55

Security Groups associated with i-08973dc73e0d74d51

Ports	Protocol	Source	basic_jupyter
80	tcp	0.0.0.0/0, ::/0	✓
22	tcp	0.0.0.0/0, ::/0	✓

4. Docker Installation

- 1) Copy the public IP Address from the instance.
- 2) Go back to Bash Shell --- Type "ssh ubuntu@IP address" --- Click "enter", then connect (type "yes").
- 3) Type "curl -sSL https://get.docker.com | sh" to start downloading and installing docker.

```
ubuntu@ip-172-31-28-248:~$ curl -sSL https://get.docker.com | sh
# Executing docker install script, commit: 1d31602
+ sudo -E sh -c apt-get update -qq >/dev/null
+ sudo -E sh -c apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null
+ sudo -E sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | apt-key add -qq - >/dev/null
+ sudo -E sh -c echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial edge" > /etc/apt/sources.list
list
+ [ ubuntu = debian ]
+ sudo -E sh -c apt-get update -qq >/dev/null
+ sudo -E sh -c apt-get install -y -qq --no-install-recommends docker-ce >/dev/null
+ sudo -E sh -c docker version
Client:
Version:      18.01.0-ce
API version:  1.35
Go version:   go1.9.2
Git commit:   03596f5
Built: Wed Jan 10 20:11:05 2018
OS/Arch:     linux/amd64
Experimental: false
Orchestrator: swarm

Server:
Engine:
Version:      18.01.0-ce
API version:  1.35 (minimum version 1.12)
Go version:   go1.9.2
Git commit:   03596f5
Built: Wed Jan 10 20:09:37 2018
OS/Arch:     linux/amd64
Experimental: false
If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

    sudo usermod -aG docker ubuntu

Remember that you will have to log out and back in for this to take effect!

WARNING: Adding a user to the "docker" group will grant the ability to run
containers which can be used to obtain root privileges on the
docker host.
Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
for more information.
ubuntu@ip-172-31-28-248:~$
```

- 4) Add our user to the docker group, copy and paste the text from above red box and run it. Type "exit" to logout and log back in. Then we'll have docker installed.

5. Obtaining the correct Docker image

- 1) Launch Jupyter notebook server --- Verify the docker works fine --- Type "docker pull jupyter/datascience-notebook" to pull the image.
- 2) Type "docker images" to verify the image has been pulled out correctly.
- 3) To give the name a shorter tag, type "docker tag **image ID** dsnb" to tag it as "dsnb".

6. Running the correct Docker image as a container

- 1) Continue with step 5, run this image by typing "docker run -v /home/ubuntu:/home/jovyan -p 80:8888 -d dsnb" to obtain the container ID.
- 2) Type "docker ps" to see the container.

7. Jupyter notebook security concerns

- 1) Continue with step 6. Open the browser, put IP address in, connect with Jupyter notebook server.
- 2) Go back to Bash shell, run “docker exec [container ID](#) jupyter notebook list” to get the token
- 3) Copy the token and put it back into the browser “Password or token” part, then login.

8. A detailed budget of the costs of running a Jupyter Data Science Notebook Server for three months using at least three different kinds of EC 2 instances.

- ❖ Assume we're using 30GB storage with Linux per month in Oregon Region.
 - Instance Type 1: t2.small, usage rate is \$0.023 per hour, data transfer rate is \$0.09/GB. Total rate for 3 months = rate/hour * hours/day * days/month * total months + data transfer rate * 30GB = $\$0.023 * 24 * 30 * 3 + \$0.09 * 30 = \$52.38$
 - Instance Type 2: t2.medium, usage rate is \$0.0464 per hour, data transfer rate is \$0.09/GB. Total rate for 3 months = rate/hour * hours/day * days/month * total months + data transfer rate * 30GB = $\$0.0464 * 24 * 30 * 3 + \$0.09 * 30 = \$102.924$
 - Instance Type 3: m5.large, usage rate is \$0.096 per hour, data transfer rate is \$0.09/GB. Total rate for 3 months = rate/hour * hours/day * days/month * total months + data transfer rate * 30GB = $\$0.096 * 24 * 30 * 3 + \$0.09 * 30 = \$210.06$
- ❖ Total cost = Instance type 1 cost + Instance type 2 cost + Instance type 3 cost = $\$52.38 + \$102.924 + \$210.06 = \365.364