# PA1: AWS and Dask Setup

## Contents



## 1 Introduction

This manual will cover how to setup multiple instances on AWS and start the Dask scheduler and workers for Programming Assignment 1.

For this assignment, you will be running your code on two EC2 instances and also on a cluster of five EC2 instances and measure the *speedup*. There are points for how much speedup your code achieves when moving from 2 to 5 instances. Refer the grading rubric for more details.

## 2 AWS Setup

The AWS setup is very similar to PA0. For launching multiple instances, there are three differences. First, we will specify 5 instances instead of 1. One of these instances will run our Jupyter Notebook and Dask Scheduler, and the remaining 4 instances will run our Dask workers. Second, each of these instances will have 100GB SSD storage instead of 40GB. Third, we will create a new security group for our 5 instances that allow each of the instances to communicate with each other. Follow the below steps one by one.

**a.** Access your ETS account using single sign-on ID: https://ets-apps.ucsd.edu/individual/DSC102_WI24_A00 . To open the AWS console click "Click here to access AWS" at the bottom of the page. To get your AWS credentials for CLI / API usage click "Generate API Keys (for CLI/scripting)".

**b.** Open AWS Dashboard. We will first create a new security group so that we can apply it to all of our instances later. Click on "Security Groups" on the left menu.

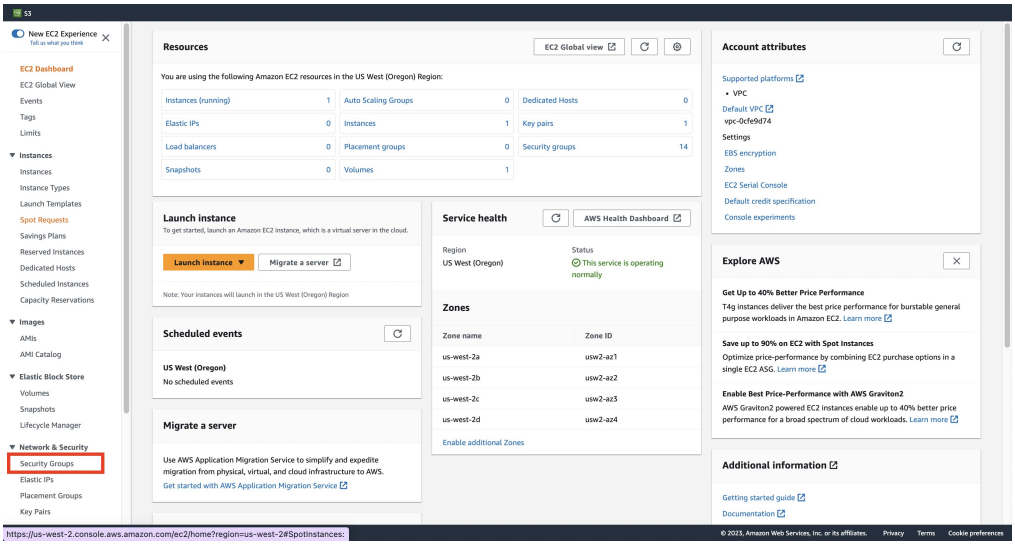


Figure 1:

**c.** Click on create security group.

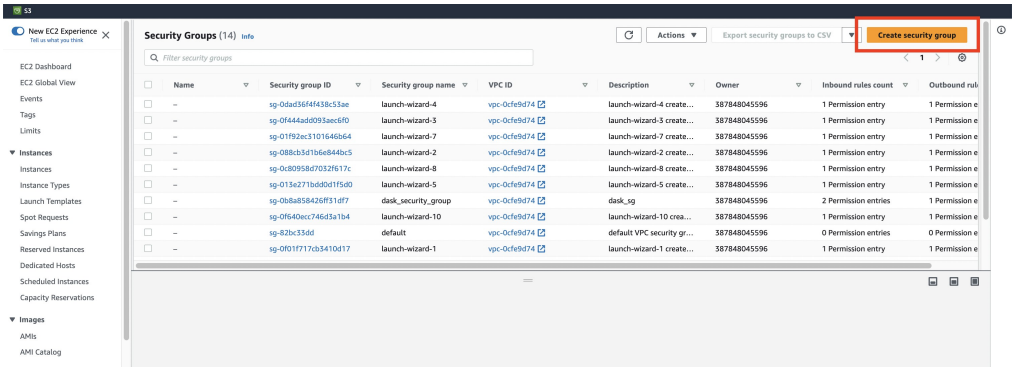


Figure 2:

**d.** Choose any name and description for your security group. Change both the inbound and outbound rules to have "Type=All Traffic", "Source/Destination = Anywhere-IPv4". See image below.

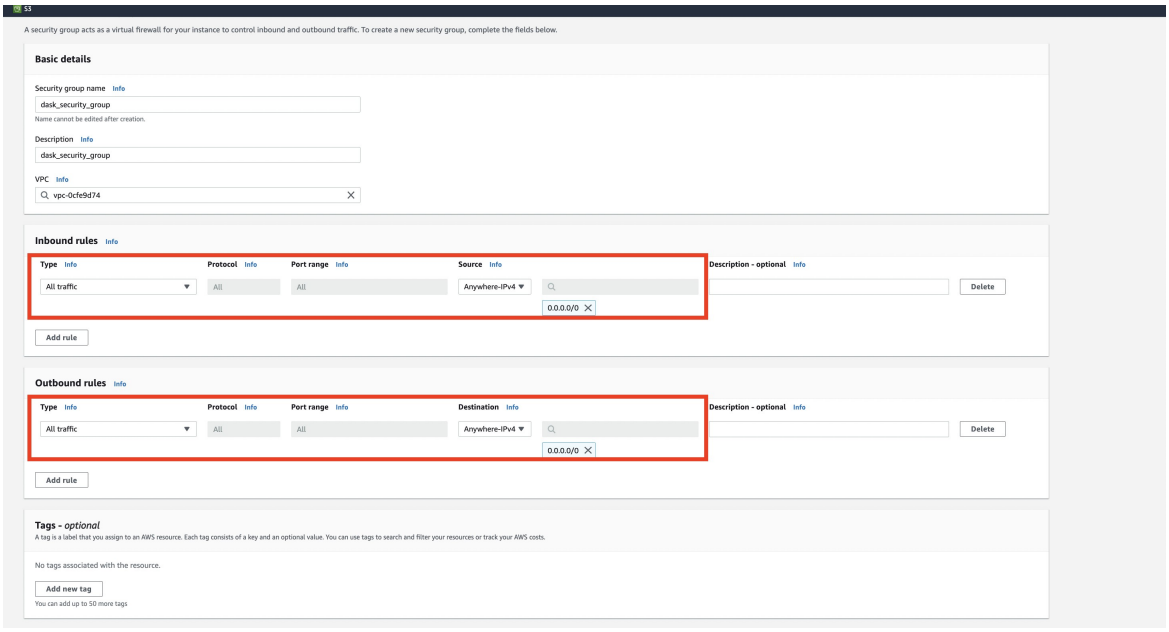


Figure 3:

**e.** Lastly, click on "Create Security Group" at the bottom right.

**f.** Now, we will create our 5 instances which will use this new security group. We have setup the Dask environment on an AMI with the name "dsc102-dask-pa1-image." Go to "AMIs" (under "Images") in your EC2 dashboard, select private images, and then search by name to find it. Select this AMI. See Figure 4.

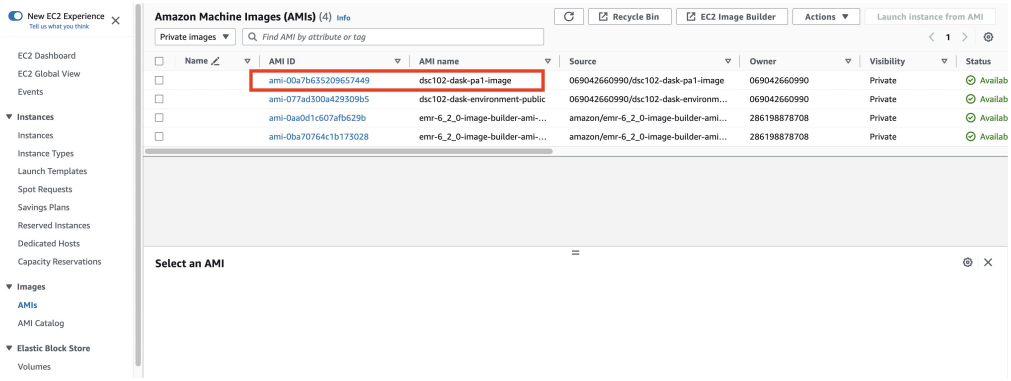


Figure 4:

**g.** After selecting the AMI, click "Launch Instance from AMI" as shown below.

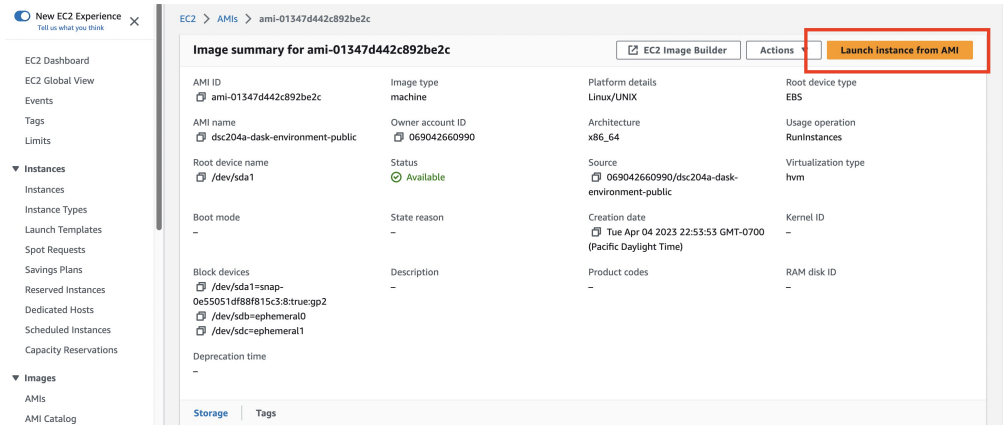


Figure 5:

**h.** Now, strictly follow the below instructions to launch the EC2 Spot instances.

**a.** Give any name for your instance.

**b.** Number of instances to launch is 5.

**c.** The instance type is “t2.xlarge”.

**d.** Create a new key pair for SSH’ing to your instance later. The private key will be downloaded to your local machine.

**e.** **Under “Network Settings", click “Select Existing Security Group" and choose the name of the security group you just created, which in my case is “dask_security_group".**

**f.** Choose 100GB SSD gp2 storage.

**g.** Open advanced details. Select “Request Spot Instances”.

**h.** Lastly, click “Launch Instance”.

# 3 Dask Setup

After launching 5 instances, your AWS dashboard must look like the figure below. You can view the IP addresses of the different instances inside the red box.

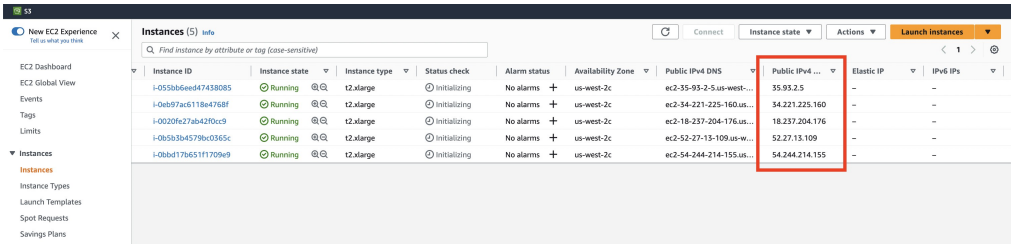


Figure 6:

## 3.1 First Instance Setup

We will run Jupyter Notebook and the Dask scheduler on the first of our 5 EC2 instances. As you will see, we will SSH into this instance 5 times on 5 different terminals to execute different commands. If you are comfortable with tmux you can avoid creating these many terminals, but for sake of simplicity, I will avoid using tmux. The IP address of my

first instance is 35.95.2.5 as seen in Figure 6. **Replace this IP with the IP of your first instance in the commands below.**

**a. Terminal 1 -** (Dataset transfer from S3 bucket into our EC2 SSD)

1. Open a terminal on your machine and SSH into the instance:
   ssh -I "my-key.pem" ubuntu@35.95.2.5

2. Copy your AWS API Keys (from the link on ETS dashboard) and paste it in the terminal. **3.** Start dataset transfer –
   aws s3 sync s3://dsc102-pa1-public .

**b. Terminal 2 -** (Starting jupyter notebook)

1. Open a terminal on your machine and SSH into the instance-
   Ssh -I "my-key.pem" ubuntu@35.95.2.5

2. Activate the dask environment:
   source dask_env/bin/activate

3. Start jupyter notebook on port 8888 –
   jupyter-notebook –port=8888

**c. Terminal 3 -** (Starting Dask Scheduler)

1. Open a terminal on your machine and SSH into the instance-
   Ssh -I "my-key.pem" ubuntu@35.95.2.5

2. Activate the dask environment
   source dask_env/bin/activate

3. Start Dask scheduler :
   dask scheduler –host 0.0.0.0

**d. Terminal 4 -** (Port forward jupyter notebook)

**1.** Open a terminal on your machine and run the below command. This will forward the process on EC2's port 8888 (jupyter notebook) to your computer's port 8888.

ssh -I "my_key.pem" ubuntu@35.95.2.5 -L 8888:localhost:8888

**e. Terminal 5 -** (Port forward dask dashboard)

**1.** Open a terminal on your machine and run the below command. This will forward the process on EC2's port 8787 (dask dashboard) to your computer's port 8787.

ssh -I "my_key.pem" ubuntu@35.95.2.5 -L 8787:localhost:8787

## 3.2 Remaining 4 Instances Setup

We will run our Dask workers on the remaining 4 instances. We need a way for the workers to communicate with the scheduler and this is achieved by passing in the IP address of the scheduler to each worker. This is done in terminal 2 below. We can find the address of the Dask scheduler from the output in terminal 3 of our first EC2 instance. In my case, the scheduler address is tcp://172.31.2.184:8786. See figure below.

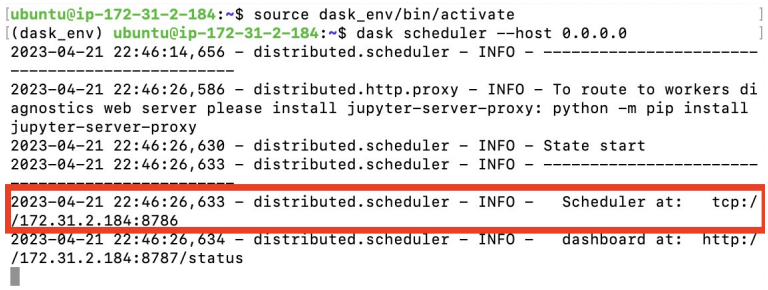


Figure 7:

We will launch 2 terminals for each of our 4 EC2 instances, making a total of 8 terminals. In the first terminal we will run the command to download our dataset from the S3 bucket to the EC2 instance (similar as done for instance 1 above). In the second terminal we will start our Dask workers.

**Repeat the below two steps for each of your 4 EC2 instances.** The only difference is the IP address you will use to SSH. I have shown the steps for my second EC2 instance below.

**a. Terminal 1 -** (Dataset transfer from S3 bucket into our EC2 instance)

1. Open a terminal on your machine and SSH into the instance:
   ssh-i"my-key.pem"ubuntu@34.221.225.160
2. Copy your AWS API Keys (from the link on ETS dashboard) and paste it in the terminal.
3. Start dataset transfer -
   aws s3 sync s3://dsc102-pa1-public .

**b. Terminal 2 -** (Starting Dask Worker)

1. Open a terminal on your machine and SSH into the instance:
   ssh -I “my-key.pem” ubuntu@34.221.225.160

2. Activate the dask environment
   source dask_env/bin/activate

3. Start 4 Dask worker processes on this machine. Replace scheduler address with yours.
   dask worker tcp://172.31.2.184:8786 –nworkers 4

## 3.3 Sanity Check

After completing the above steps, when you visit the Dask dashboard at http://localhost:8787, you should see 16 workers in total. See figure below.

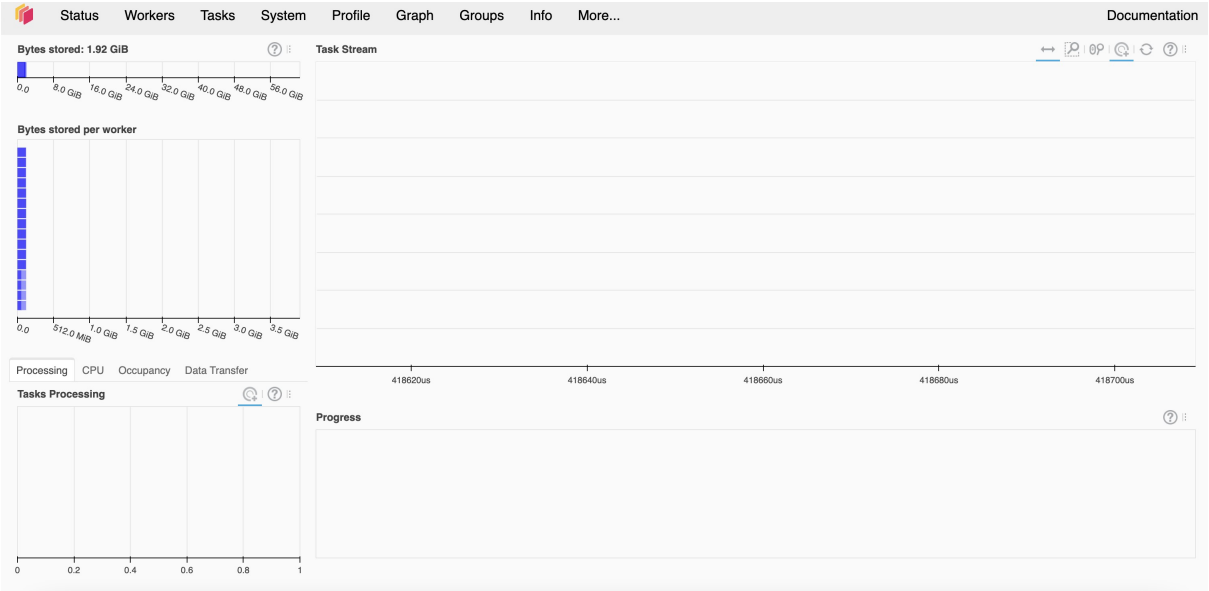


Figure 8:

## Scheduler tcp://172.31.2.184:8786

Logs Exceptions Bokeh

### Workers

| Worker | Name | Cores | Memory | Memory use | Occupancy | Processing | In-memory | Services | Logs | Last seen |
|---|---|---|---|---|---|---|---|---|---|---|
| tcp://172.31.12.172:37127 | tcp://172.31.12.172:37127 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 909.91 ms |
| tcp://172.31.12.172:37895 | tcp://172.31.12.172:37895 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 906.63 ms |
| tcp://172.31.12.172:42221 | tcp://172.31.12.172:42221 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 906.25 ms |
| tcp://172.31.12.172:43499 | tcp://172.31.12.172:43499 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 907.28 ms |
| tcp://172.31.14.185:33763 | tcp://172.31.14.185:33763 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 197.28 ms |
| tcp://172.31.14.185:41667 | tcp://172.31.14.185:41667 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 195.39 ms |
| tcp://172.31.14.185:43197 | tcp://172.31.14.185:43197 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 196.13 ms |
| tcp://172.31.14.185:43637 | tcp://172.31.14.185:43637 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 195.13 ms |
| tcp://172.31.4.5:34285 | tcp://172.31.4.5:34285 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 999.63 ms |
| tcp://172.31.4.5:35269 | tcp://172.31.4.5:35269 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 1.00 s |
| tcp://172.31.4.5:39791 | tcp://172.31.4.5:39791 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 1.53 ms |
| tcp://172.31.4.5:44969 | tcp://172.31.4.5:44969 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 998.47 ms |
| tcp://172.31.8.59:32967 | tcp://172.31.8.59:32967 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 110.15 ms |
| tcp://172.31.8.59:35657 | tcp://172.31.8.59:35657 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 113.24 ms |
| tcp://172.31.8.59:40651 | tcp://172.31.8.59:40651 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 111.31 ms |
| tcp://172.31.8.59:45317 | tcp://172.31.8.59:45317 | 1 | 3.91 GiB | | 0.00 us | 0 | 0 | dashboard | logs | 109.86 ms |

Figure 9: