DSC 102: Systems for Scalable Analytics

Programming Assignment 0

1 Introduction

The goal of this programming assignment is to get you comfortable with datasets that do not fit in single-node memory and are too big for tools like Pandas or NumPy. You will be using Dask library to explore secondary storage aware data access on a single machine. In this assignment, you will be learning to setup dask on AWS and computing several descriptive statistics about the data to build intuitions for feature engineering for the final assignment.

2 Dataset Description

You are provided with the Amazon Reviews dataset with the *reviews* table as CSV file. The schemas are provided in Table 1. The dataset is available on the s3 bucket: `s3://dsc102-public`.

Column name	Column description	Example
reviewerID	ID of the reviewer	A32DT10X9WS4D0
asin	ID of the product	B003VX9DJM
reviewerName	name of the reviewer	Slade
helpful	helpfulness rating of the review	[0, 0]
reviewText	text of the review	this was a gift for my friend who loves touch lamps.
overall	rating of the product	1
summary	summary of the review	broken piece
unixReviewTime	summary of the review	1397174400
reviewTime	time of the review (raw)	04 11, 2014

Table 1: Schema of Reviews table

3 Tasks

You will use the *reviews* table to explore features related to users. Specifically, you will create users table with the schema given in Table 2.

A code stub with function signature for this task has been provided to you. The input to the function is the reviews CSV file and you will be carrying out a series of transformations to produce the users table as DataFrame. Plug in the DataFrame you obtained as a result in <`YOUR_USERS_DATAFRAME`> and write this to `results_PA0.json` file. We will time the execution of the function `PA0`.

We have shared with you the "development" dataset and our accuracy results. Our code's average runtime on 1 node is roughly 18 mins. You can use this to validate your results and debug your code. The final evaluation will happen on separate held-out test sets. The runtime will be different for the held-out test set.

4 Deliverables

Submit your source code as <`YOUR-TEAM-ID`>`.py` on Canvas. Your source code must confirm to the function signatures provided to you. Make sure that your code is writing results to `results_PA0.json`.

Column name	Column description
reviewerID (PRIMARY KEY)	ID of the reviewer
number_products_rated	Total number of products rated by the reviewer
avg_ratings	Average rating given by the reviewer across all the reviewed products
reviewing_since	The year in which the user gave their first review
helpful_votes	Total number of helpful votes received for the users' reviews
total_votes	Total number of votes received for the users' reviews

Table 2: Schema of users table

5 Getting Started

0. Access your AWS account using single sign-on ID: `https://ets-apps.ucsd.edu/dsc102wi21-custom-aws`. Credentials for CLI / API usage can be retrieved using a modified URL: `https://ets-apps.ucsd.edu/dsc102wi21-custom-aws?mode=env`.

1. We have setup the Dask environment on an AMI with name "dsc102-dask-environment-public." Go to "AMIs" (under "Images") in your EC2 dashboard, select public images, and then search by name to find it. Select this AMI. See Figure 1 and Figure 2.

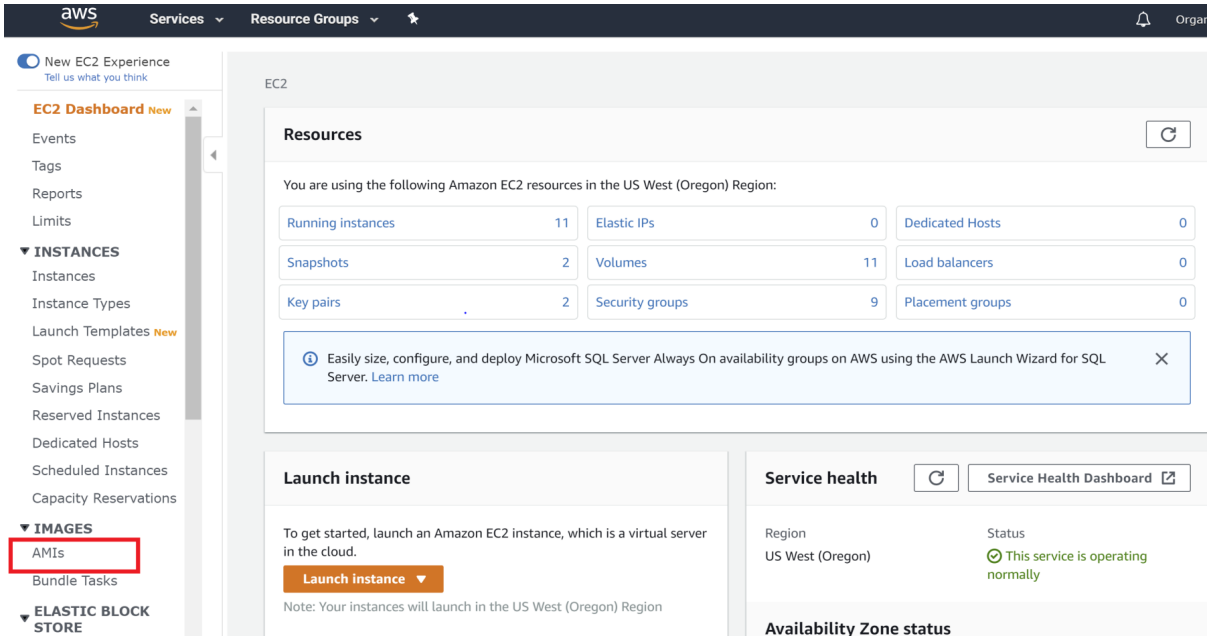

Figure 1

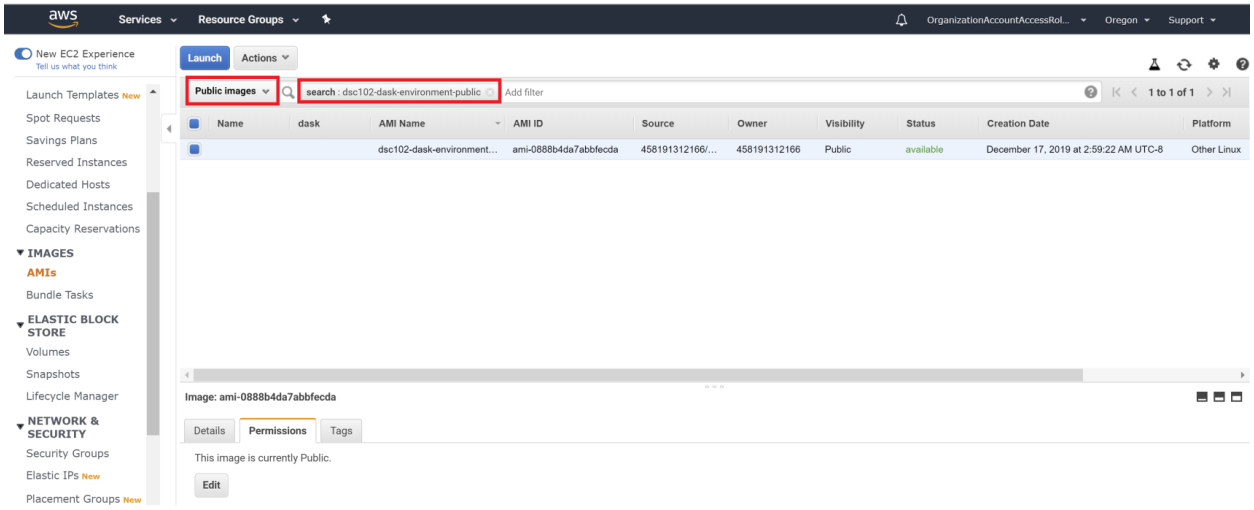

Figure 2

2. Now, you will be launching one EC2 instance that will be used to run dask locally. Follow the steps below.

Launch one EC2 instance of type "t2.xlarge" and "40GB" of storage. In the next page, put 1 instance in the "Number of instances" box and don't change anything else. Create a new security group. Retain other fields unchanged. Finally, after pressing the "Launch" button, add a key pair and download this locally. This will allow you to SSH into the instance.
Note that you need not create a new security group and a key pair every time you launch your instance. If you already have a security group configured and a private key file downloaded on your computer, you can select the existing security group and the existing keypair next time you launch an instance.
See Figure 3 to Figure 7. At the end, you should be able to see one instance in your dashboard.

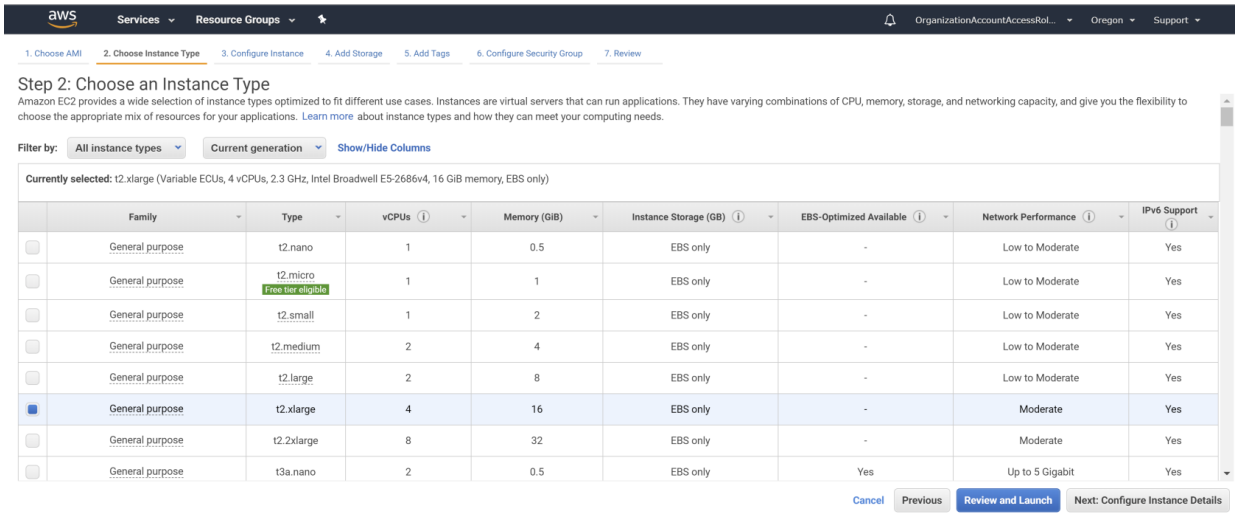

Figure 3

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1 Launch into Auto Scaling Group
Purchasing option	Request Spot instances
Network	vpc-f58df88d (default) Create new VPC
Subnet	No preference (default subnet in any Availability Zone) Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	Add instance to placement group
Capacity Reservation	Open
Domain join directory	No directory Create new directory
IAM role	None Create new IAM role
CPU options	Specify CPU options
Shutdown behavior	Stop
Stop - Hibernate behavior	Enable hibernation as an additional stop behavior

Cancel | Previous | Review and Launch | Next: Add Storage

Figure 4

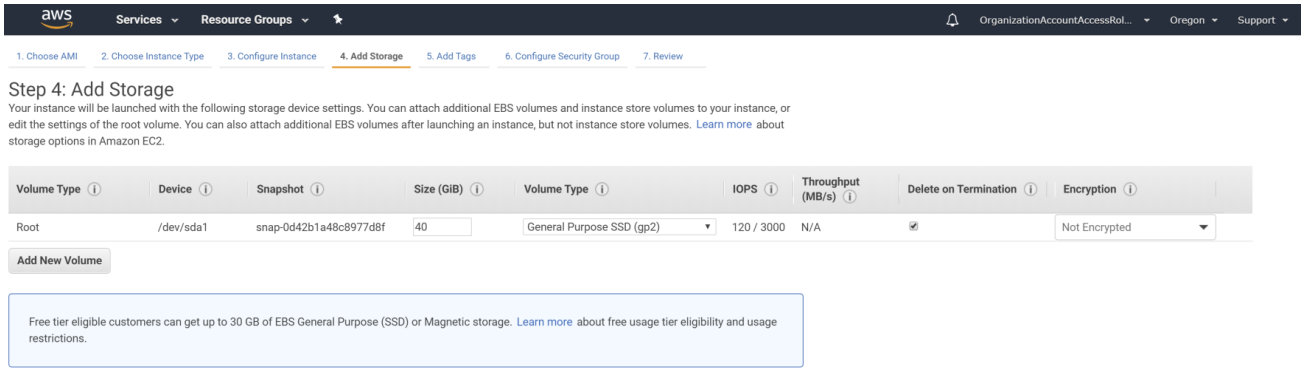

Figure 5

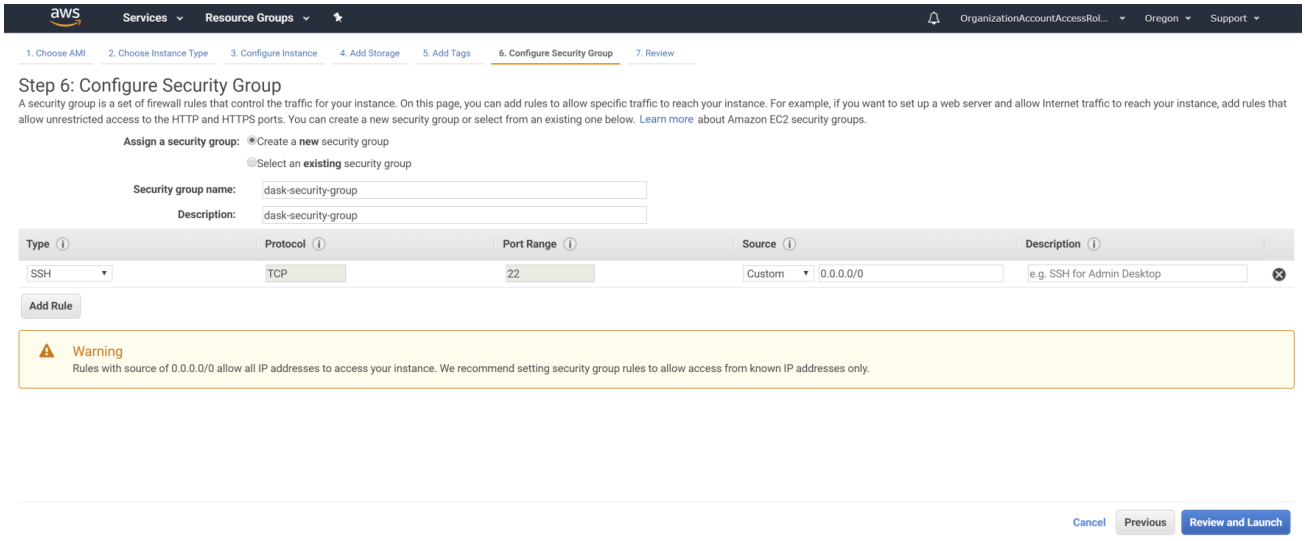

Figure 6

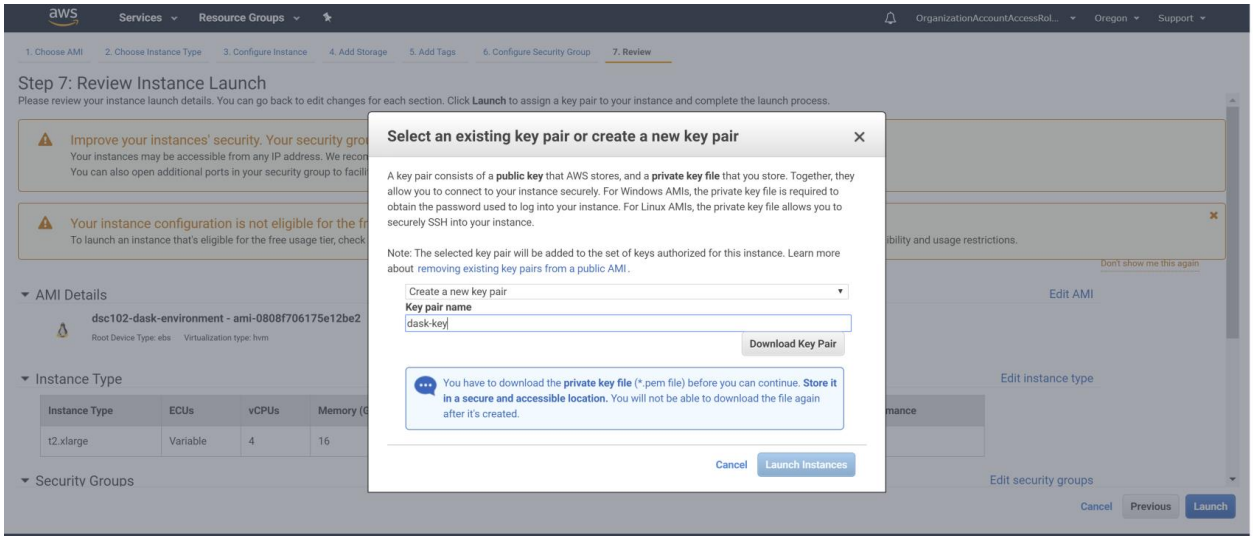

Figure 7

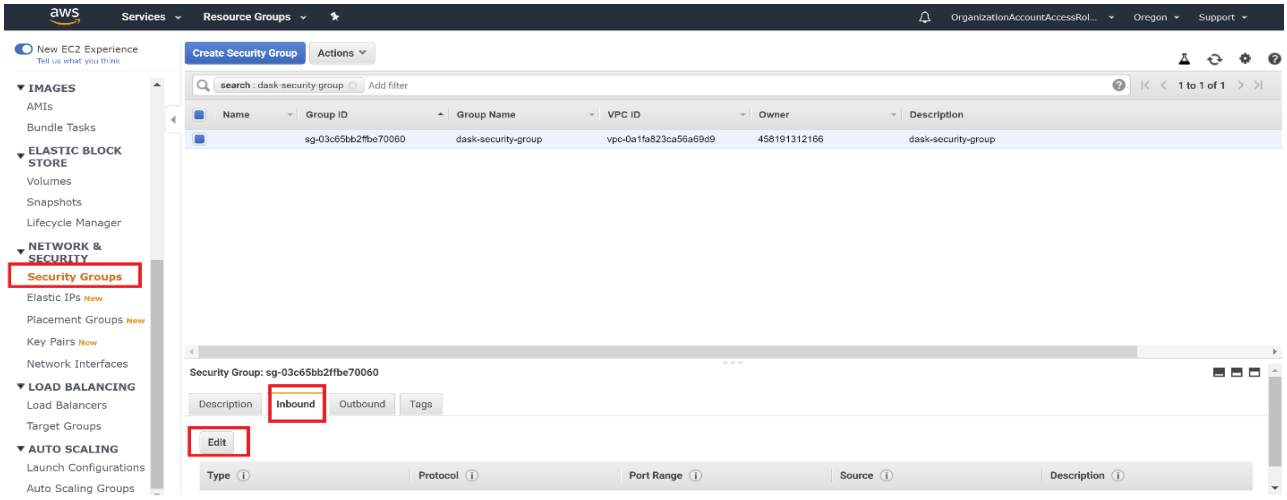

Figure 8

3. Once the EC2 instance is launched, go to the security group of the instance (under "Network & Security" in the left panel) and add the following rule (under "Inbound"). A rule with type "SSH", port 22, and source as "My IP". This rule will allow you to SSH to each of the machines. See Figure 8.
Note: If you are currently not in the US, you may have trouble trying to SSH to the instance. If you are unable to SSH (step 4.b) to the EC2 instance, change source in your rule to "Anywhere", hit save rules and try again.

4. In this step, you will start the jupyter notebook server on the instance.

a. Change permission of the ssh keyfile to make sure your private key file isn't publicly viewable: `chmod 400 <keyfilename>.pem`. Linux and Mac users in particular will need the chmod.

b. Open your terminal, go to the folder containing ¡keyfilename¿.prem and SSH into one of the nodes using command: `ssh -i ''<keyfilename>.pem'' ubuntu@<ip-address-of-EC2-instance>`. This command is shown in the Figure 9 below. <ip-address-of-EC2-instance> is shown in the red box in Figure 10. Activate the dask environment with command: `source dask_env/bin/activate`. Start jupyter notebook server on one terminal with: `jupyter notebook --port=8888`.

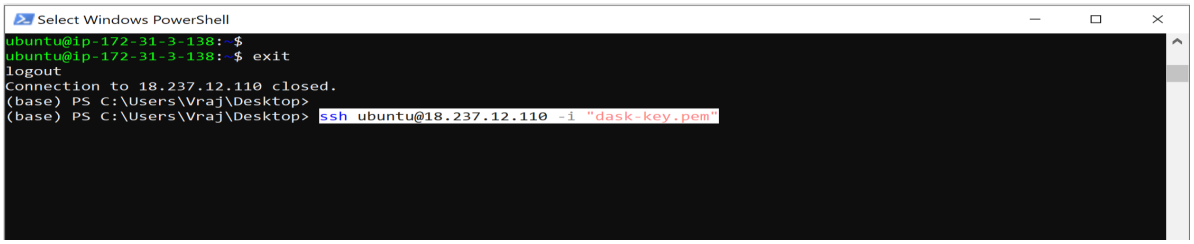

Figure 9

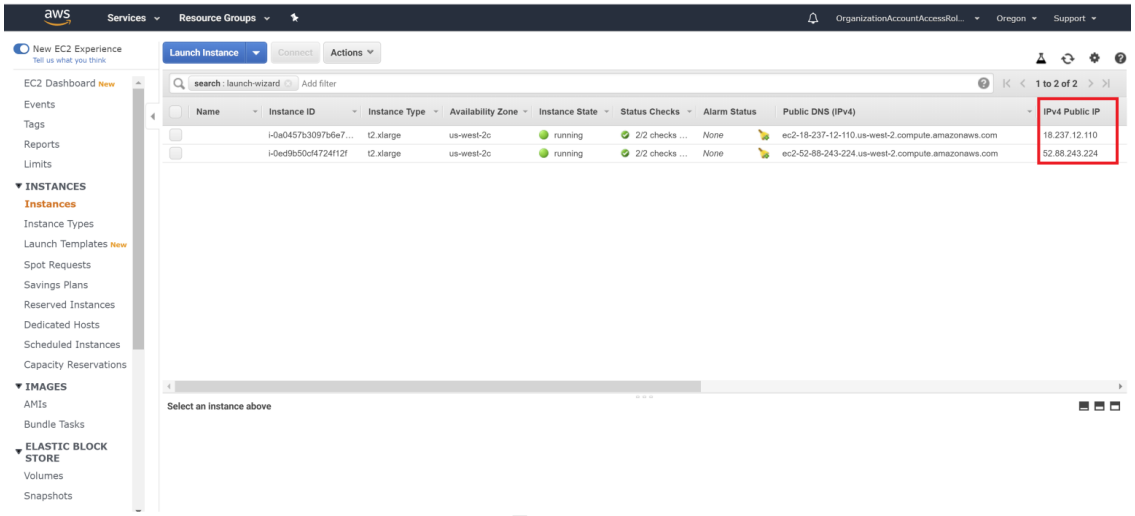

Figure 10

c. Open a new terminal and SSH to jupyter notebook using: `ssh -i ‘‘<keyfilename>.pem’’ ubuntu@<ip-address-of-EC2-instance> -L 8000:localhost:8888`. ‘-L’ will port forward any connection to port 8000 on the local machine to port 8888 on <ip-address-of-EC2-instance>. Type in `jupyter notebook list` to get the token/password for the jupyter notebook. Open your browser and go to localhost:8000 and paste the token. You can write your code here using jupyter notebook. To see dashboard on localhost port 8001 use command: `ssh -i ‘‘<keyfilename>.pem’’ ubuntu@<ip-address-of-EC2-instance> -L 8001:localhost:8787.`

Consider using utilities like *tmux* or *nohup* for managing terminals.

5. The data and files are available from the s3 bucket (s3://dsc102-public). This contains the function signatures (PA0.py), dataset (user_reviews.csv), schema of expected output (OutputSchema_PA0.json), and the expected result on the development dataset (results_PA0.json).

a. First, setup your aws credentials on all nodes using:

```
export AWS_ACCESS_KEY_ID= <YOUR_AWS_ACCESS_KEY>
export AWS_SECRET_ACCESS_KEY= <YOUR_AWS_SECRET_KEY>
export AWS_SESSION_TOKEN= <YOUR_AWS_SESSION_TOKEN>
```

You can find your aws credentials at `https://ets-apps.ucsd.edu/dsc102wi21-custom-aws?mode=env`
b. Use command to download the files: aws s3 sync s3://dsc102-public /local-file-path. Download the data files on local disk on all nodes. Make sure that data is available in the same path where the jupyter notebook client is running.
6. Open the dashboard and click on “Workers” to double check if all workers (all threads of the single machine) are connected and you are now ready to code up.

7. Terminate EC2 instances once you are done. Remember when you terminate an EC2 instance you lose all the data, therefore we suggest you use a private GitHub repo to routinely push your work (code, logs and other small files) and pull your repo whenever you create a new instance to resume your work.