DSC 102: Systems for Scalable Analytics

Programming Assignment 1

1 Introduction

In this assignment, you will be using Dask library to explore task parallelism on multiple cores on a cluster of machines. You will be performing feature exploration, data consistency checks, and computing several descriptive statistics about the data to build intuition for feature engineering for the next assignment.

2 Dataset Description

You are provided with the Amazon Reviews dataset with the *reviews* and *products* tables as CSV files. The schemas are provided in Table 1. The goal for the final assignment (PA2) will be to to predict user's star rating. Thus, "overall" is our label.

(A) 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 (helpful, total votes)	[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	Unix time stamp of the review	1397174400
reviewTime	time of the review (raw)	04 11, 2014

(B) Column name	Column description	Example
asin	ID of the product	143561
salesRank	sales rank information	{'Movies & TV': 376041}
imUrl	url of the product image	http://g-ecx.images-amazon.com /31mC.jpg
categories	list of categories the product	[['Movies & TV', 'Movies']]
title	name of the product	Everyday Italian (with Giada de Laurentiis)
description	description of the product	3Pack DVD set - Italian Classics
price	price in US dollars	12.99
related	related products (also bought, also viewed, bought together, buy after viewing)	{'also_viewed': ['B0036FO6SI', '000014357X'],'buy_after_viewing ': ['B0036FO6SI', 'B000KL8ODE']}
brand	brand name	1

Table 1: (A) *Reviews* table and (B) *Products* table

3 Tasks

You will compute several descriptive statistics for both *reviews* and *products* table as follows:

Q1. Get percentage of missing values for all columns in the *reviews* table and the *products* table

Q2. Find pearson correlation value between the price and ratings

Q3. Get mean, standard deviation, median, min, and max for the price column in the *products* table

Q4. Get number of products for each super-category (the first entry in the "categories" column in the products table).

Q5. Check (Return 1 or 0) if there are any dangling references to the product ids from the *reviews* table to *products* table. Return 1 if there are dangling references.

Q6. Check (1 or 0) if there is any dangling reference between product ids in the related column and "asin" of the *product* table. Return 1 if there are dangling references.

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 the *products* CSV file. The output is a json file (saved as `results_PA1.json`). The schema has been shared with you (`OutputSchema_PA1.json`). You will write your answers to each sub-task as values. ****Do not modify any keys of the json file****. We will time the execution of the function `PA1`.

We have shared with you the "development" dataset and our accuracy results. Our code's runtime on 4 nodes and 1 node are roughly **200 seconds** and **920 seconds** respectively. 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 and the speedup numbers 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_PA1.json`.

5 Setup

1. You will be launching 5 EC2 *Spot* instances, where you will create one instance for setting up the client (jupyter notebook server) and scheduler, and four instances for running workers.

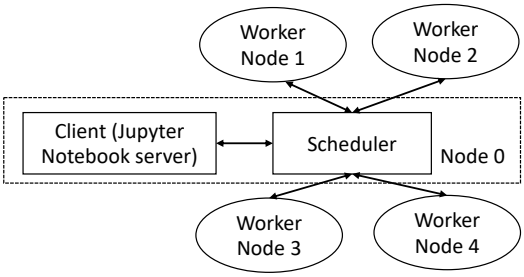

Figure 1

2. Follow the below steps for launching EC2 spot instances.

a. Access your AWS account using single sign-on ID:
`https://ets-apps.ucsd.edu/individual/DSC102_WI22_A00`
Credentials for CLI / API usage can be retrieved once you select your account from the list (selecting **Generate API Keys (for CLI/scripting)**).

b. We have setup the Dask environment on an AMI with name "**dsc102-dask-pa1-wi22-environment**"[1]. Go to "Instances" in the left panel in your EC2 dashboard and click on "Launch instances". First you need to choose an AMI. Search for "**dsc102-dask-pa1-wi22-environment**" and it will show "1 results in Community AMIs". Select this one. See Figure 2 and Figure 3.

[1]The Dask version here is different from PA0's

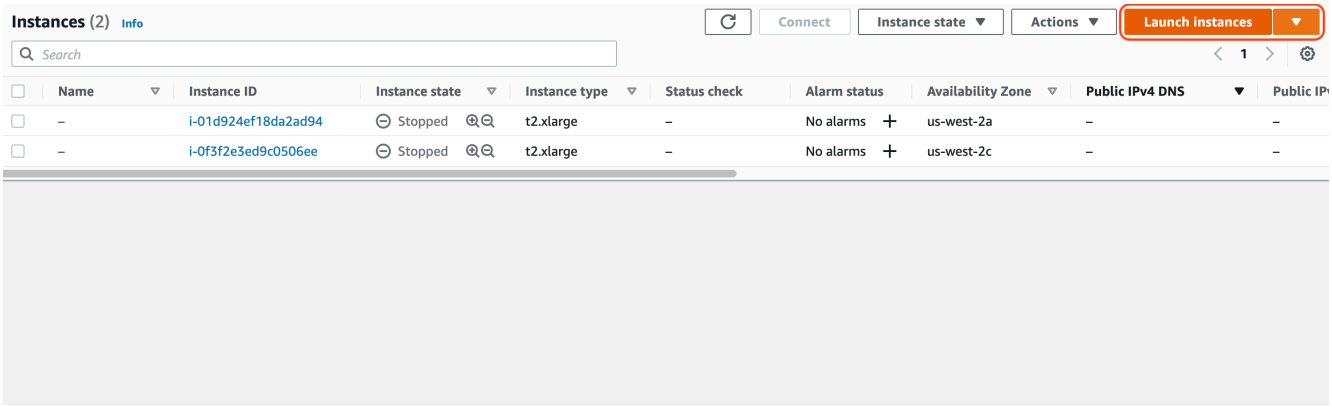

Figure 2

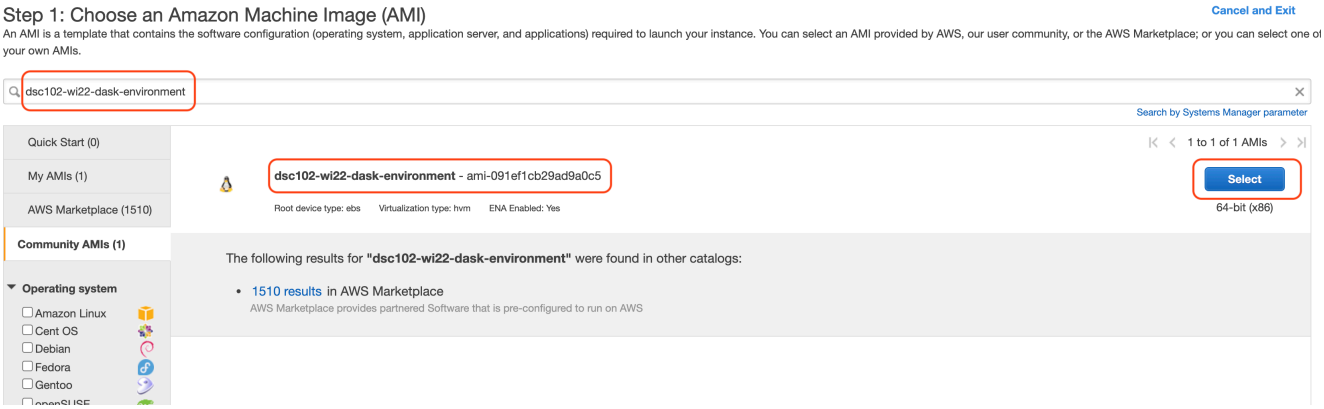

Figure 3

c. Now, you will be launching EC2 *Spot* instances with 100 GB storage. Follow the steps below.

Select the AMI. Select EC2 instance of type "**t2.xlarge**". In the next page (Configure Instance Details), put **5** in the "Number of instances" box and put a check on "**Request Spot instances**" box. Put "maximum price" as **0.0646** (If you see a different "Current price", then put whatever the current price is). 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 instances. See Figure 4 to Figure 8. At the end, you should be able to see 5 instances in your dashboard.

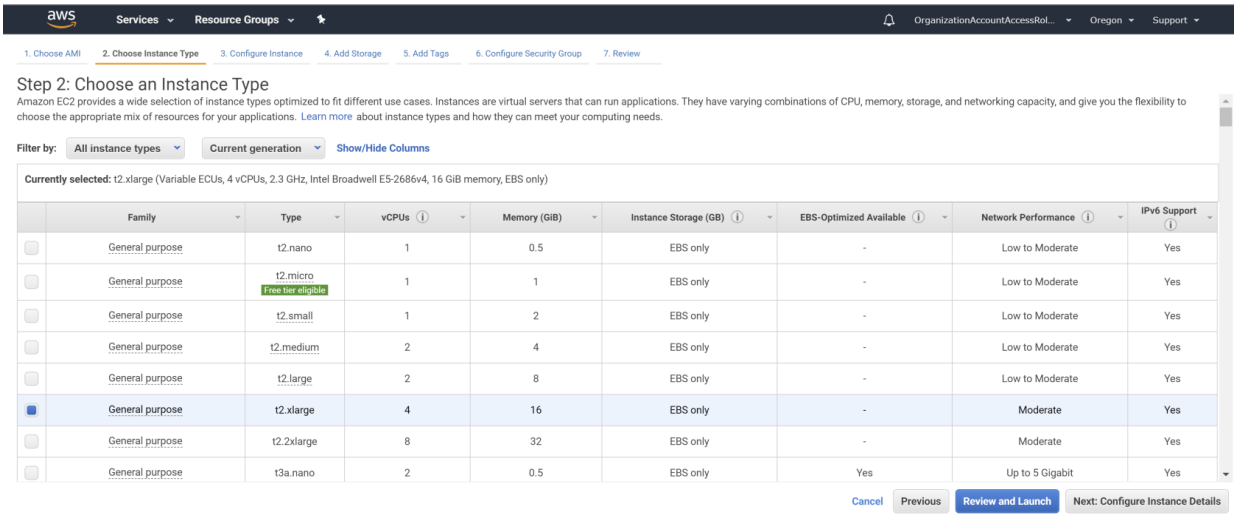

Figure 4

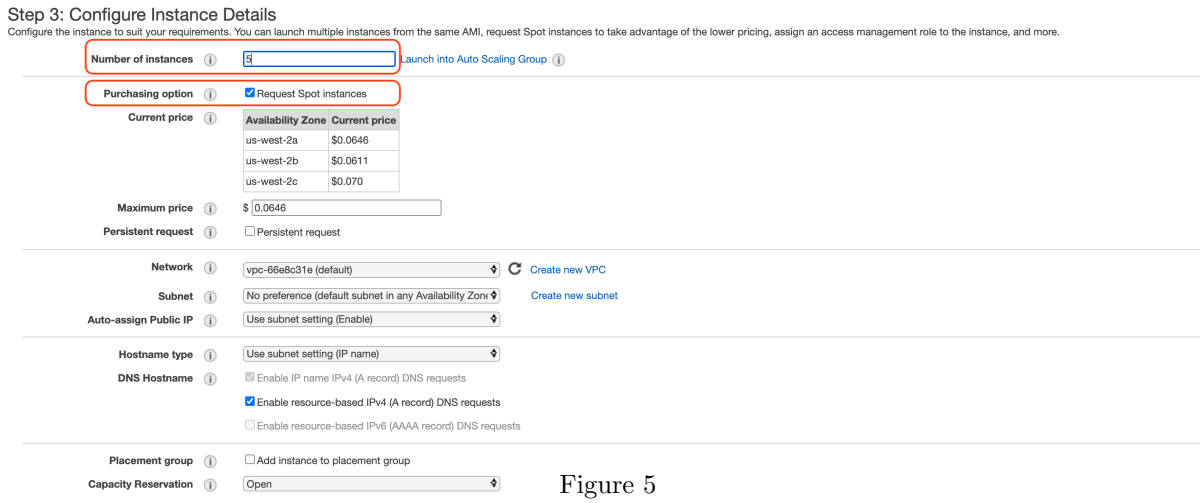

Figure 5

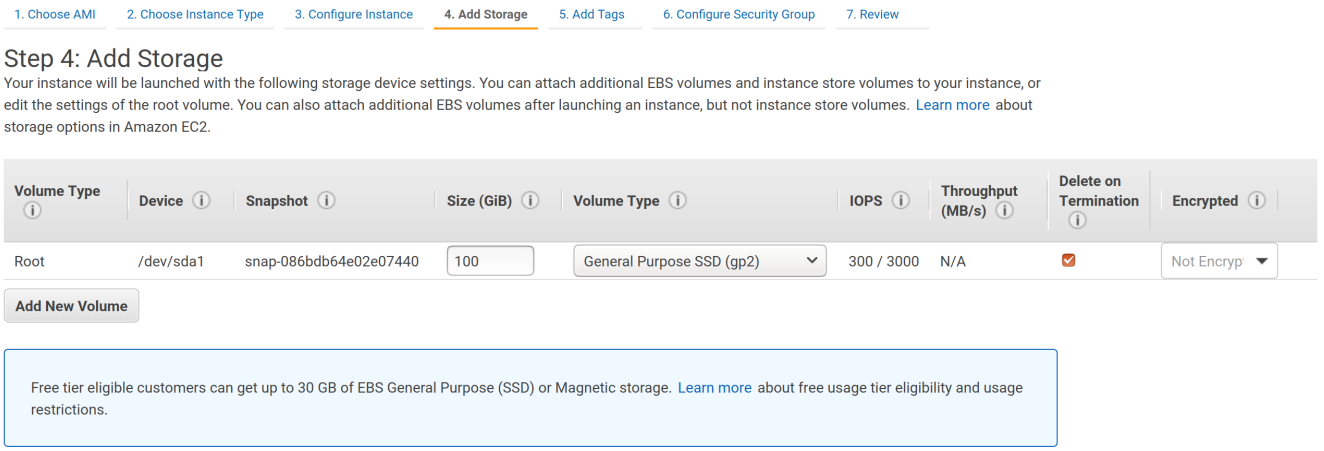

Figure 6

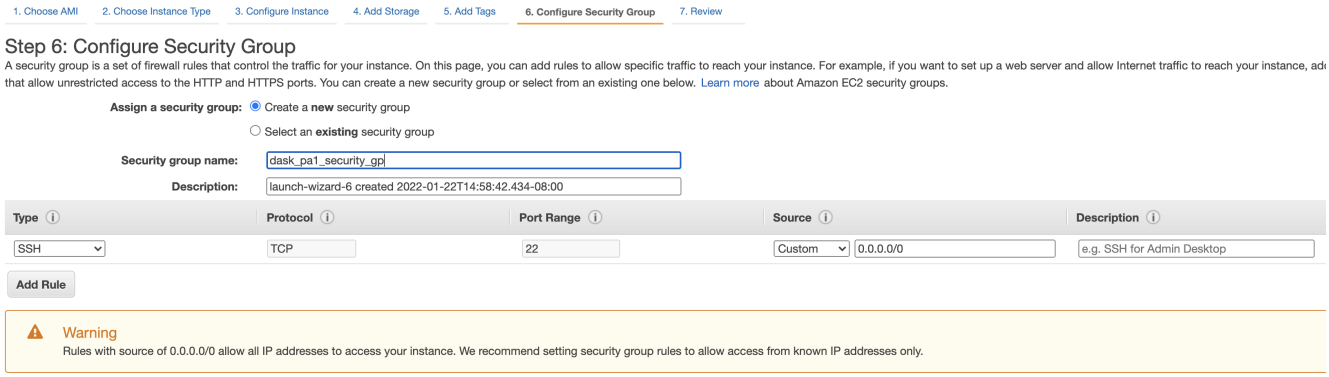

Figure 7

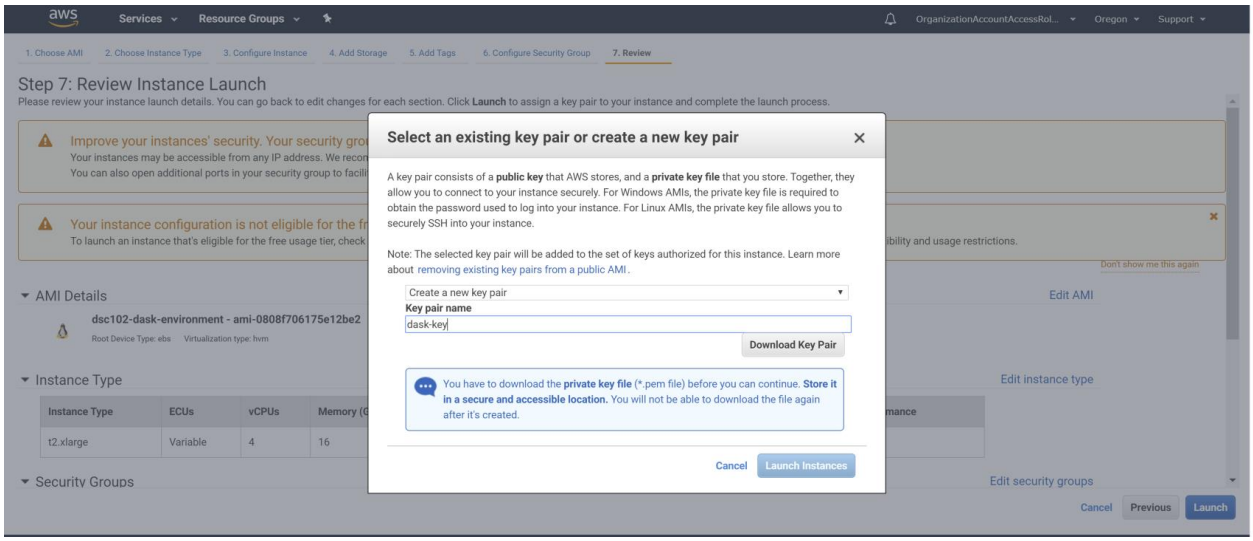

Figure 8

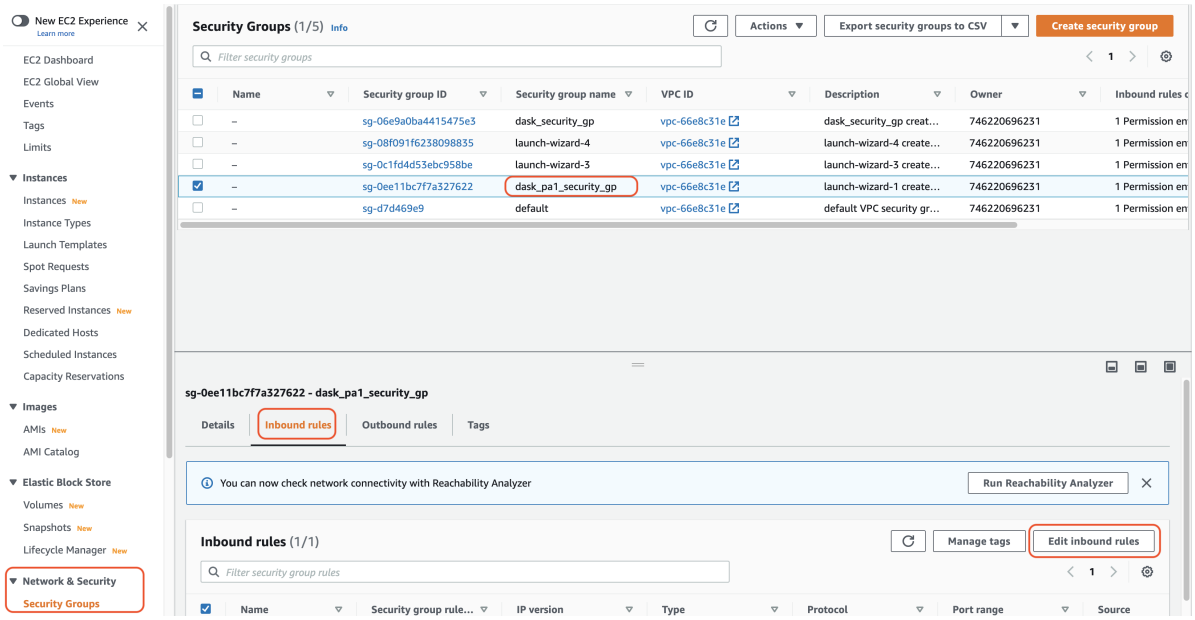

Figure 9

3. Once the EC2 instances are launched, go to the security group of the instances (under "Network & Security" in the left panel) and add two rules (under "Inbound").
(a) A rule with type "SSH", and source as "0.0.0.0/0". This rule will allow you to SSH to each of the machines.
(b) A rule with type "All TCP", and source as "Custom" with its group id. This rule will allow workers and scheduler to communicate with each other as in Figure 1.

Figure 10

4. In this step, you will start the jupyter notebook server on the client, 1 scheduler node, and 4 worker nodes (16 worker processes).

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. SSH into one of the nodes using command: `ssh -i ‘‘dask-key.pem" ubuntu@<ip-address-of-EC2-instance>`. Activate the dask environment with command: `source dask_env/bin/activate`. Start jupyter notebook server on one terminal with: `jupyter notebook --port=8888` and start the scheduler on another terminal with command: `dask-scheduler --host 0.0.0.0`
This will run the scheduler on port 8786 and dashboard on port 8787. `<address-of-scheduler>` is given at `tcp://172.31.31.176:8786` in the Figure 11 below.

c. Open a new terminal and SSH to jupyter notebook using: `ssh -i ‘‘dask-key.pem" ubuntu@<ip-address-of-EC2-instance> -L 8000:localhost:8888`. '-L' will port forward any connection to port 8000 on the local ma-

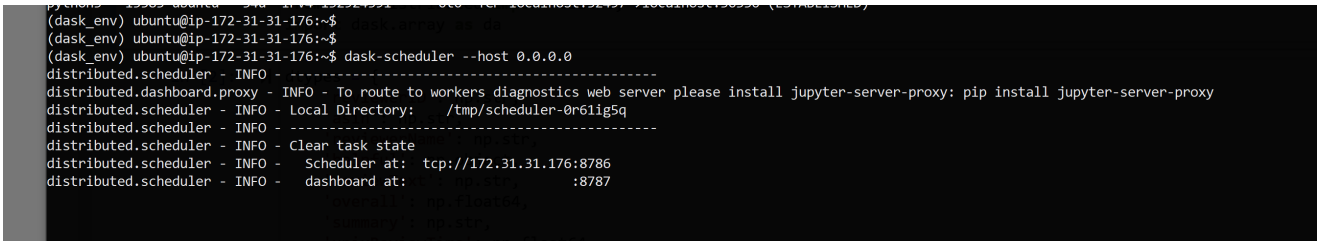

Figure 11

chine 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 ‘‘dask-key.pem" ubuntu@<ip-address-of-EC2-instance> -L 8001:localhost:8787.`

d. SSH into other 4 nodes and activate the dask environment. Start workers with command:
`dask-worker <address-of-scheduler>:8786 --nprocs 4.`
`<address-of-scheduler>` was displayed with command `dask-scheduler --host 0.0.0.0`. `nprocs` option will make sure that all the cores of the machine are used. In the scheduler terminal screen, you will be able to see the connected workers.

5. The data and files are available from the s3 bucket (s3://dsc102-data-public). This contains the function signature (PA1.py), datasets (user_reviews.csv and products.csv), schema of expected output (OutputSchema_PA1.json), and the expected result on the development dataset (results_PA1.json). Make sure that data is available in the same path where scheduler and workers are running. Refer to step 5 given in the getting started guide of PA0 for more details. To download both files together use the command:
aws s3 sync s3://dsc102-data-public <path where scheduler and workers are running>

6. Open the dashboard and click on “Workers” to double check if all workers (16) 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 smallfiles) and pull your repo whenever you create a new instance to resume your work.