Tutorial 4: AWS

ESE546

September 24, 2020

1 Overview

The EC2 (Elastic Cloud Compute) service on AWS (`https://aws.amazon.com`) offers cloud computing infrastructure and provides access to GPU resources. Each student should have 150$ worth of credits to use on AWS. AWS could be used for the programming assignments and is a useful tool for the projects. EC2 gives you access to GPU instances which are charged by the hour. In this short tutorial, we will go over the basics, best practices and some useful tools.

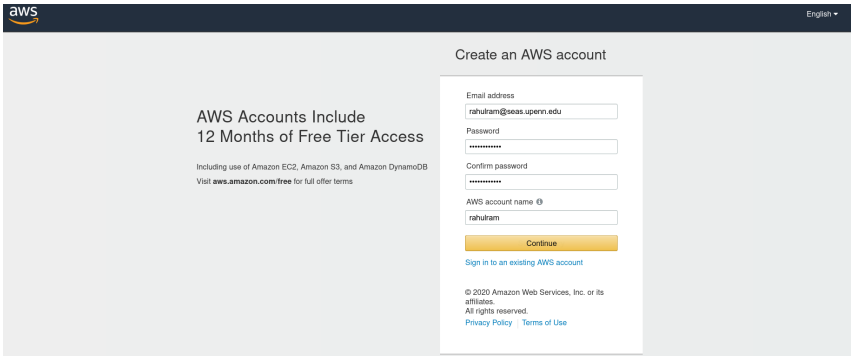
2 Creating an Account

2.1 Steps

- Create an AWS Account
- Add credit card details (while creating account) to access GPUs

2.2 Details

Go to `https://aws.amazon.com/account/` to create an AWS account. **Do not** use an AWS educate account since this does not give you access to GPU instances. Use accounts that are regular AWS accounts created using the above link.



The account creation requires your credit card in order to have access to GPU instances. This is something which we could not find a workaround to. We will walthrough steps to setup safe-guards and alarms that allow you to monitor your usage.

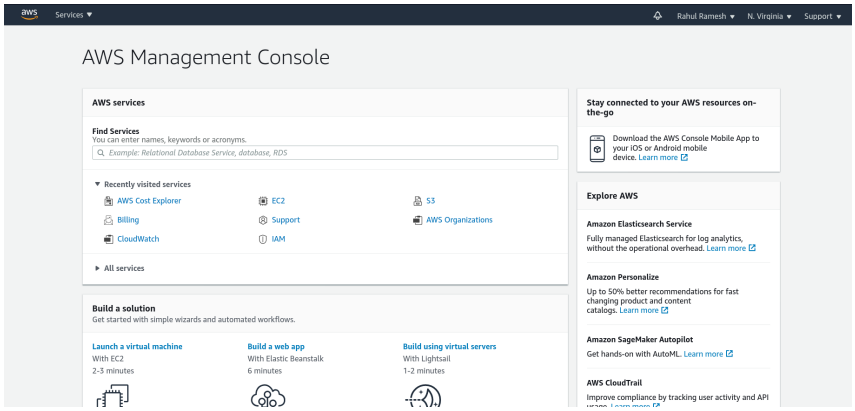
Note Although 150$ should suffice for this course, it is important to use resources judiciously. If you deplete the credits, your credit card will be charged for any resources you use beyond the 150$.

3 AWS Console and Services

The AWS console allows you to access a plethora of services offered by AWS. It is suggested that you set your zone to Virginia (U.S East) on the top right of the screen. This generally gives you better availability for the machines. The services that we will be primarily using are:

- EC2
- CloudWatch
- Billing

There are other interesting services like AWS-Lambda, AWS-Sagemaker, S3.



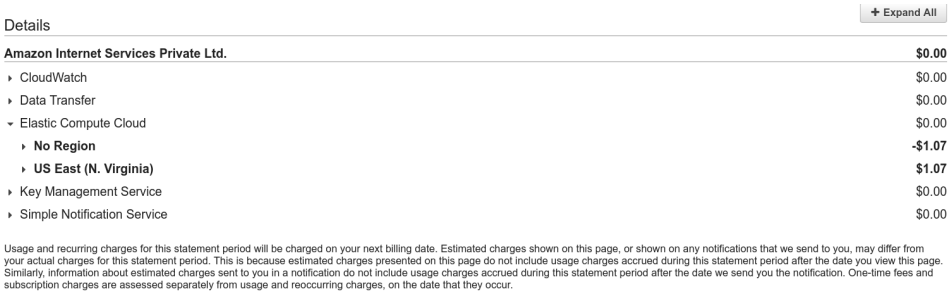
4 Billing and Credits

4.1 Steps

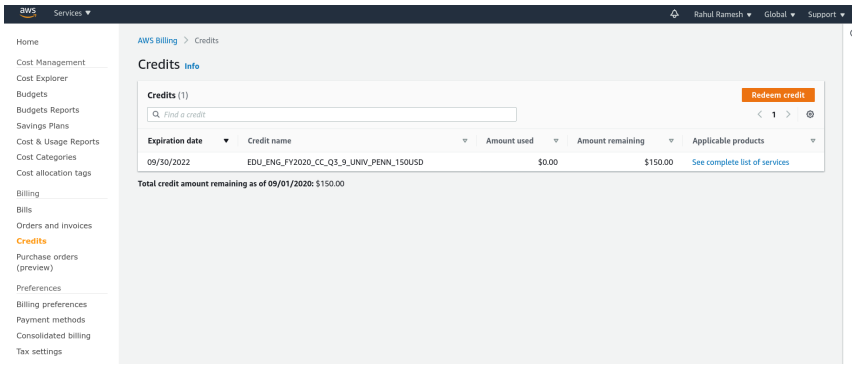
- Redeem credits using unique code
- Allow Receiving Billing Alerts

4.2 Details

The *Billing* tab (`https://console.aws.amazon.com/billing/`) allows you to monitor and handle the credits in your AWS account. In order to find the credits remaining in your account use either the *Cost Explorer* or the *Bills* tab.



In order to redeem credits, use the *Credits* tab and enter the code sent to your email after clicking on the *Redeem credits* button.



Navigate to Billing preferences and allows receiving billing alerts. This allows us to setup alarms and monitors that can help us keep track of the credits used.

5 Billing Alarms

5.1 Steps

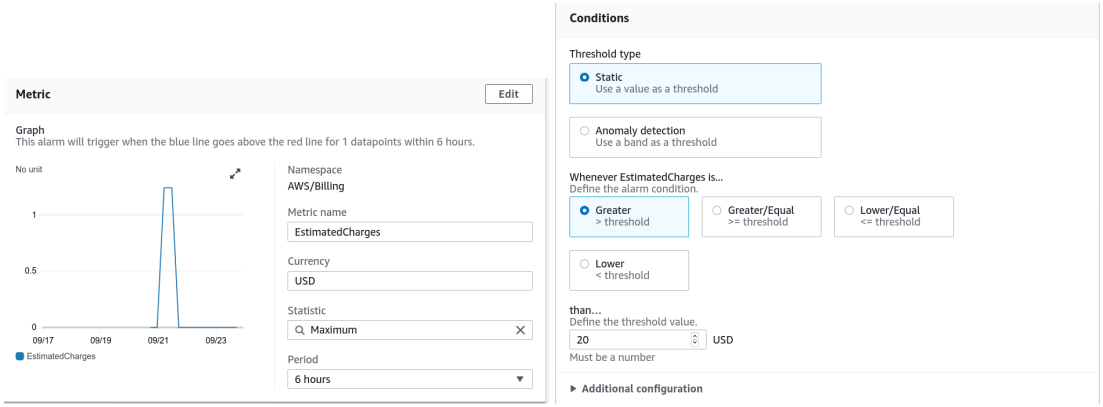
- Setup billing alarm at a specified dollar threshold. This will send you a mail when you have used a certain number of credits.
- Set multiple such alarms at thresholds 20$, 40$, 60$, ..., 120$, 130$, 140$, 150$.

5.2 Details

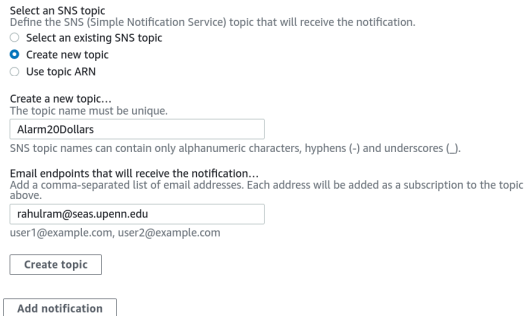
CloudWatch (`https://console.aws.amazon.com/cloudwatch/`) allows you to track metrics and sends you emails to alert you on the status of these metrics. We will make use of CloudWatch to setup "Alarms" that alert us when we 30% or 50% of the credits have been used. An alarm is setup as follows:

1. Click on the Billing tab on the left
2. Create an Alarm
3. Select Metric → Billing → Total Estimated Charge → Select EstimatedCharges metric. [Note if Billing isn't visible, then Billing alerts may not have been enabled. Details for the same are in the previous section]

4. Selecting: (6 hours, Static, Greater than 20$) will setup an alarm that trigger after you cross the threshold of 20$. Each alarm allows you to set a single threshold. Hence we require multiple alarms to operate at different thresholds.



5. Add Notification and set to *In Alarm*. Create a new SNS topic. This allows you to "subscribe to a mailing list" that sends you emails when the alarm is triggered. If you have already created one, you can make use of an existing SNS topic.



This setup should ensure that you do not exceed the 150$ limit. Make sure to set multiple alarms and different thresholds and monitor your usage in general.

6 Service Quotas and Spot Instance Access

6.1 Steps

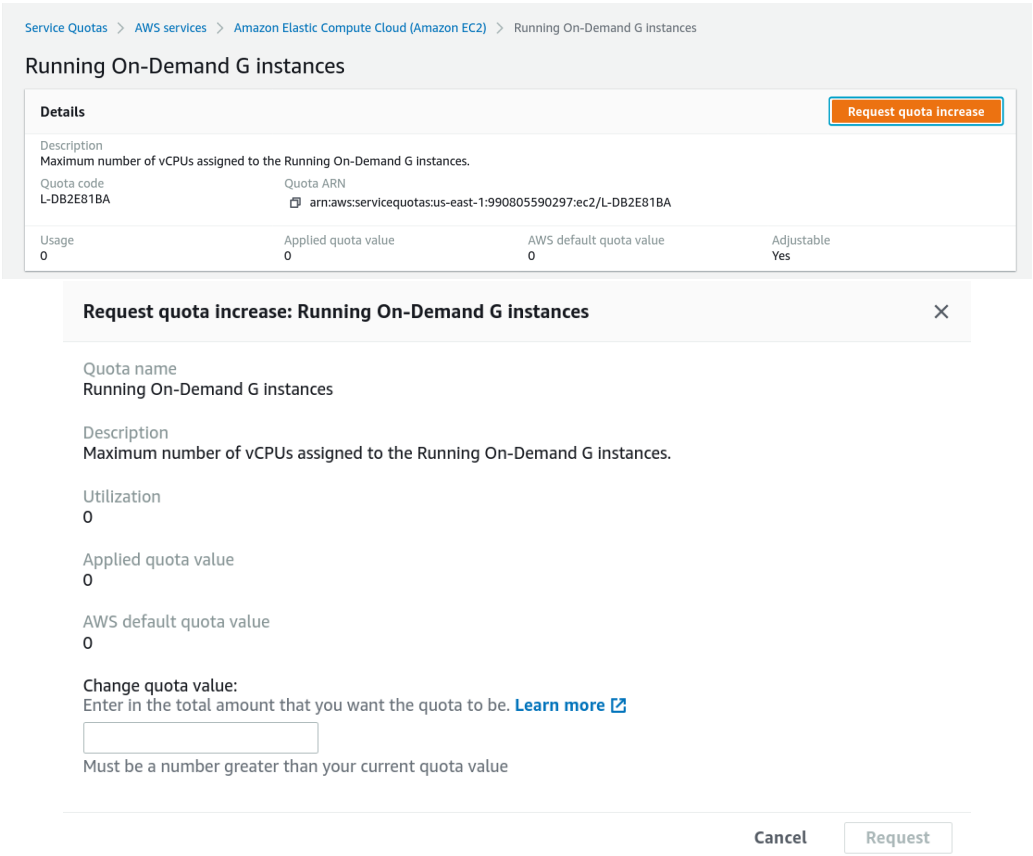
- Gain access to G-type and P-type instances.
- Obtain permission to launch *Spot* Instances

6.2 Details

Before we can spin up instances using EC2, the first step is to obtain the necessary permissions. Amazon EC2 offers a wide array of instance types (`https://aws.amazon.com/ec2/instance-types/`). Some are compute optimized while others are memory optimized. GPU instances are either P-instances or G-instances. Some instances that are relevant for this course are:

- p3.2xlarge - Nvidia v100 GPU, 8Cpus, 61GiB (0.90$/hr spot price)
- **g4dn.xlarge** - Nvidia T4 GPU, 4Cpus, 16GiB (0.15$/hr spot price)
- p2.xlarge - Nvidia K80 GPU, 4Cpus, 61GiB (0.25$/hr spot price)

We recommend using the g4dn.xlarge for this course since it is fairly cheap while being fairly competitive with respect to performance. p3.2xlarge is a recently released model and hence may have higher demand.
The first step is obtain the permissions to spawn such instances (https://console.aws.amazon.com/servicequotas). Click on EC2 and search for "Running On-Demand P Instances" and "Running On-Demand G-instances". Request a quota increase for both P-instances and G-instances. Requesting for a quota of 1/2 is reasonable.



The next step is to request access to deploy spot instances. Navigate to https://console.aws.amazon.com/support/home?#/case/create?issueType=service-limit-increase.

- Set Limit type = EC2 Spot Instances
- Region = US-East (or whichever region you plan on using)
- Instance type = g4dn.xlarge (or p2.xlarge). File multiple requests if you want access to both instances.
- In the "Use-case Description", mention that you're a student at Penn and you will be using the instances for coursework.
- Selecting "Phone" will expedite the process but "Web" should also work.

The requests take about 2-3 days (for the "Web" option) after which you should be ready to deploy an EC2 spot-instance.

7 Deploying an On-demand Instance

An on-demand instance is like your own computer on the cloud. You pay for the machine while it is in use and pay for storage (which is fairly minimal) when it is stopped. Terminating the instance will delete all the data and free up the instance.
The following are the step involved in creating an on-demand instance.

- Navigate to the EC2 tab and click on launch instance
- Choose AMI:
  - Select the community AMI tab and search for "Deep Learning AMI (Ubuntu 18.04) Version 34.0".
  - The instance starts with CUDA, CuDNN, pytorch and tensorflow installed.
- Choose Instance Type:
  - Select either p2.xlarge, p3.2xlarge or g4dn.xlarge.
- Configure Instance:
  - Change the sub-net to the zone in which you want to launch the instance.
  - Choosing the zone is important if you want to mount an external volume present in the same region.
- Add Storage:
  - This determines the size of the root volume
  - Generally, additional storage is not required.
- Add Tags:
  - Tags are not necessary but are useful for naming instances when handling a large number of instances
- Configure Security Group:
  - Create a security group that allows access to port 22 (ssh access)
- Review and Launch
  - Launch the instance and create a new key pair.
  - Save the PEM file since it is the only way to access an EC2 instance.

The EC2 console allows us to monitor the state of the instance. The instance is charged on the hour when it is *running*. If you shut-down the instance (*stopped*), the data is saved and can be accessed when the instance started again. When an instance is shut-down, you will be charged for the associated storage (0.1$ per GB-month). Once you terminate the instance, you will no longer be charged for it. However, you will lose access to the associated data.

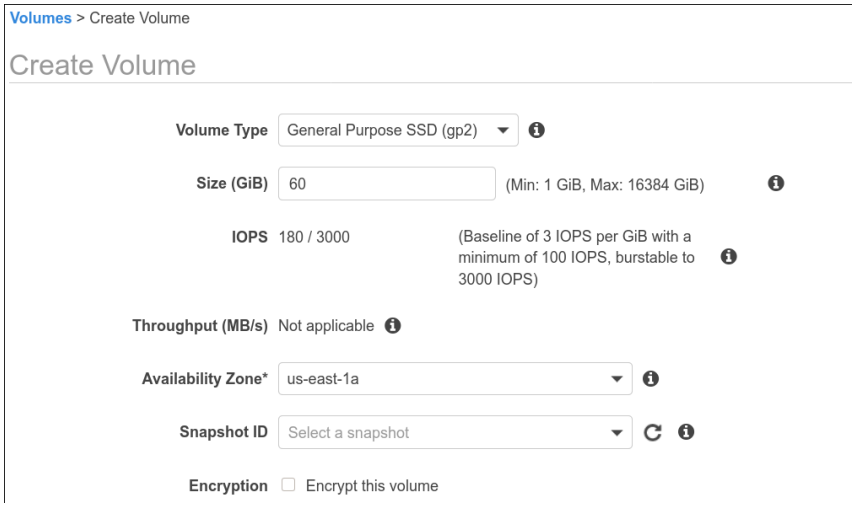
Note In order to be cost efficient, we suggest **not using** on-demand instances. Instead, consider using spot-instances since they are 3-times cheaper.

8 Deploying a Spot Instance

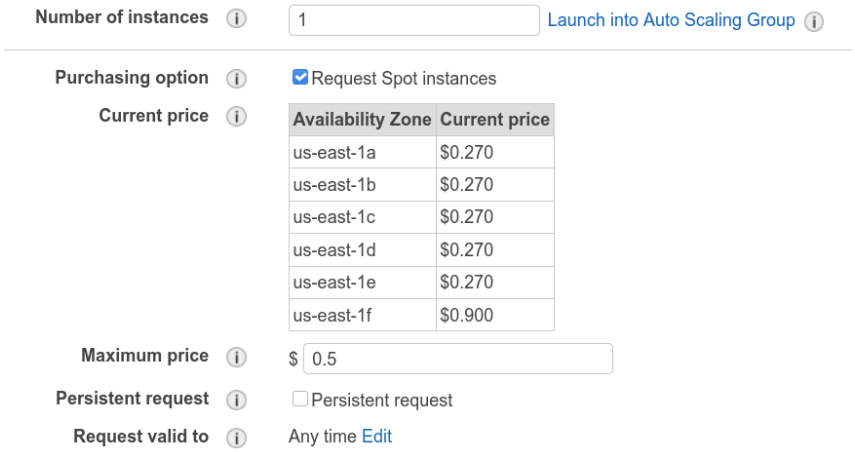
AWS allows us to use unused on-demand instances at around 1/3rd the price. However, these instances can be interrupted at any point of time. In order to achieve persistence, we create an external volume which can be attached/detached to any spot instance. An outline of the steps are as follows:

- Create an external volume (one-time)
- Launch a spot-instance
- Attach the external volume

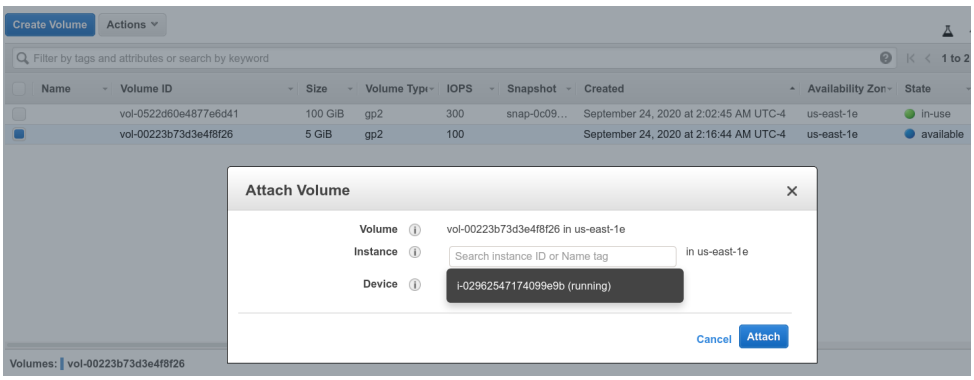
Create a volume of around 50-100GB using the volume tab. You only need to do this once and you can re-use this volume for all consequent spot-instance deployments.



Now create a spot instance. The process is identical to creating a on-demand instance. The only difference is that in the *Configure Instance* tab, check spot-instance to be true. The current-price is determined based on a supply-demand model. While the price can sometimes fluctuate, it often remains fairly stable. If the current price exceeds the *Maximum price* (0.5 in the image), then the instance is terminated.

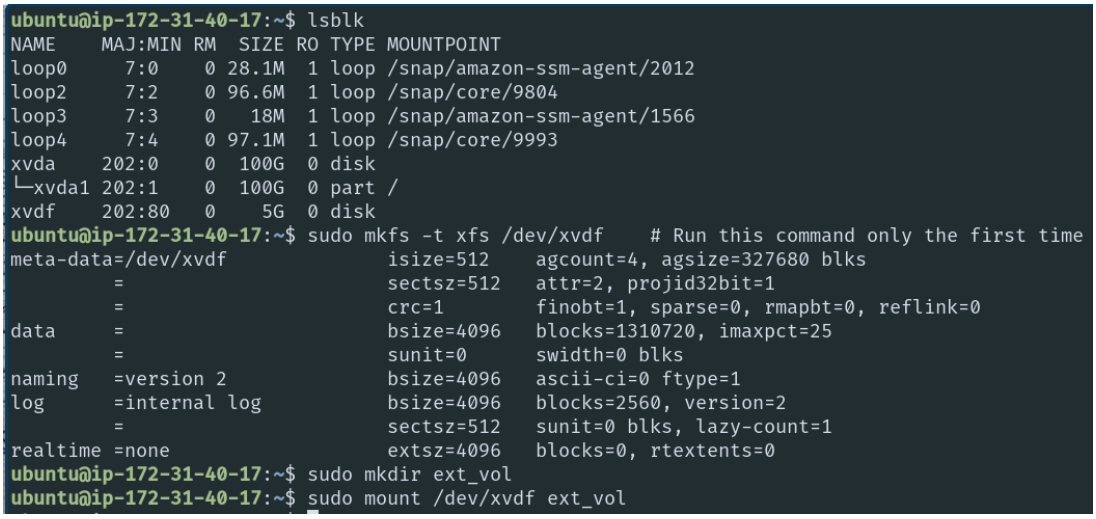


One can attach or detach external volumes in the volume tab. *Note that the volume and the ec2 instance must both lie in the same zone*. Make sure to specify the right zone when creating a spot-instance.



Why do we need external volumes? If a spot instance is killed, then all your data is lost. However all data

stored in an external volume can be accessed in a new instantiation of a spot-instance. Consider the output of LSBLK in the screenshot below. Note that "xvda" is the root volume consisting of the ubuntu installation and installed packages. This volume is deleted if a spot instance is killed or terminated. Hence all installed packages have to be re-installed whenever a new spot-instance is started. However, any data stored in xvdf (like checkpoints, code, logs) will not be deleted and can be mounted to a new spot instance in the same zone. This process hence allows us to achieve some form of persistence with spot-instances.



Note that the below command should only be run the first time that the volume is used. This command wipes the data on the entire disk so be careful when executing this command. The purpose of the command is to create a fresh file-system. See the above screen-shot for usage:

```
sudo mkfs -t xfs /dev/xvdf
```

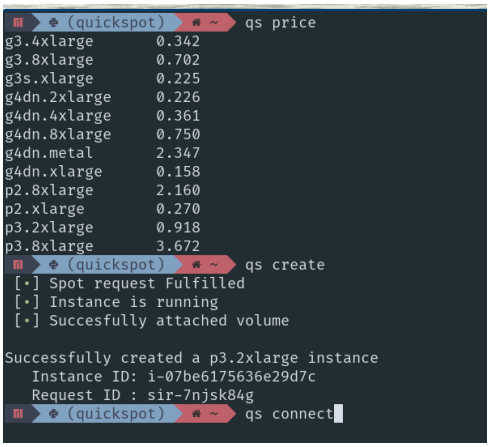
We run the below commands after attaching a volume to a new spot-instance. The below commands mount the attached volume and make it accessible through a folder.

```
lsblk                           # Check if xvdf is attached
sudo mkdir ext_vol              # Create folder to mount volume to
sudo mount /dev/xvdf ext_vol    # Mount external volume
```

The data and packages present in xvda are determined by choice of the AMI. One can potentially build a customized AMI to determine the files and packages present in "xvda". We omit further discussion on this topic since the Deep learning AMI should suffice for this course.

9 Automation Scripts

We can automate the above process using command line scripts. *boto3* is a python library that provides APIs that allow you to interact with AWS. Check out the scripts at https://github.com/rahul13ramesh/quickspot.



10 Tools

Some useful tools to consider are:

- scp/rsync to copy files to and from the ec2 instance
- sshfs, ssh plugin for vs-code/sublime - Access individual files for editing/viewing
- nohup/tmux - Ideally you would like to close you ssh session after running your code and let it run in the background. Closing the ssh session generally interrupts your code. Hence consider using nohup (or tmux/screen) to run code without being interrupted. Nohup is fairly simple and is used as follows:

```
# Running code: This allows you to close the ssh session while the code is running
#   nohup : Run interrupted
#   -u    : Write unbuffered to out stream
#   2>&1  : Write STDERR and STDOUT to out.log
#   &     : run in background
nohup python -u mnist_pytorch.py 2>&1 >out.log &

# Monitoring process
htop                # CPU/Mem usage (q to exit)
nvidia-smi          # GPU Usage
tail -f out.log     # Follow current output (Ctrl-C to exit)

# Stopping execution
ps ax | grep mnist_pytorch.py | grep -v grep # Get Process ID (PID)
#    >>> 27827 pts/1    RNl    1:17 python -u mnist_pytorch.py
kill 27827                                  # Replace 27828 with PID
```

- anaconda/miniconda - A package management tool that helps you build a reproducible self-contained coding environment. The Deep learning AMI uses anaconda to and contains environments for tensorflow, pytorch1.4, pytorch 1.6, mxnet. You can access the pytorch environment using

```
conda info --env             # List of envs
conda activate pytorch_new   #select environment to use
python train.py
```

- pip - Package management for just python packages. Pip is more minimal than anaconda but can only handle python packages. Anaconda can additionally install C-packages, CUDA or even different python versions.

Note Make sure to delete all resources (instances/volumes/etc..) after the course is over. This is to ensure that the account is not unnecessarily bleeding credits after the course is completed. Volumes might consume about 5$ a month which can slowly accumulate.