A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Use this command to pick out your default vpc, note, it is the vpc with ‘"IsDefault": true’

aws ec2 describe-vpcs | grep VpcId

🡪 vpc-e4e35381

Now create a security group which will allow us to login and expose a port for a Jupyter notebook to be run.

aws ec2 create-security-group --group-name hw06 --description "HW06" --vpc-id vpc-e4e35381

Not take the security group which we have just created and open the posrts for inbound and outbound traffic.

aws ec2 authorize-security-group-ingress --group-id sg-09ceb02f960da25fa --protocol tcp --port 22 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id sg-09ceb02f960da25fa --protocol tcp --port 8888

We will not use docker here, instead we will use the Deep Learning AMI provided by Amazon. During the following steps you will need to accept the terms and conditions of the Deep Learning AMI you use, you will be provided a link where you can accept this.

First let’s find a deep learning AMI in your region. You can replace the region I chose with your own. Please make sure your default region has p3.2xlarge instances available, if not, you may need to change regions.

Use this to pick your deep learning AMI id.

aws ec2 describe-images --filters Name=name,Values='Deep\*Learning\*Ubuntu\*18.04\*32\*'

For example mine is : "ImageId": "ami-0f5ebd171c26abc61"

Now lets look at the spot pricing on the p3.2xlarge, spot pricing provide a significant discount over on demand pricing. However as the price fluctuates, we need to set a limit which we are willing to pay. I would recommend approx. 50% over the spot price. You can see the spot pricing using the below command.

aws --region=eu-west-1 ec2 describe-spot-price-history --instance-types p3.2xlarge --start-time=$(date +%s) --product-descriptions="Linux/UNIX" --query 'SpotPriceHistory[\*].{az:AvailabilityZone, price:SpotPrice}'

At time of writing, the spot pricing is just under $1 per hour, so I set the limit at $1.50. Similar to week02 homework enter the below in a file called `spot-options.json` in your current directory.

{

"MarketType": "spot",

"SpotOptions": {

"MaxPrice": "1.50",

"SpotInstanceType": "one-time"

}

}

Now lets start the image.

aws ec2 run-instances --image-id ami-0f5ebd171c26abc61 --instance-type p3.2xlarge --security-group-ids sg-09ceb02f960da25fa --associate-public-ip-address --instance-market-options file://spot-options.json --key-name darraghaws

Again, it will take a couple of minutes to create, you can get the server address by using the below.

aws ec2 describe-instances | grep ec2

We will also need out public IP later for running Jupyter.

aws ec2 describe-instances | grep PublicIp

My public ip is `54.194.227.21`

Now we login,

ssh -i "darraghaws.pem" ubuntu@ec2-54-194-227-21.eu-west-1.compute.amazonaws.com

You will get a message like below on logging in.

Please use one of the following commands to start the required environment with the framework of your choice:

for MXNet(+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate mxnet\_p36

for MXNet(+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate mxnet\_p27

for MXNet(+AWS Neuron) with Python3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate aws\_neuron\_mxnet\_p36

for TensorFlow(+Keras2) with Python3 (CUDA 10.0 and Intel MKL-DNN) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate tensorflow\_p36

for TensorFlow(+Keras2) with Python2 (CUDA 10.0 and Intel MKL-DNN) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate tensorflow\_p27

for Tensorflow(+AWS Neuron) with Python3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate aws\_neuron\_tensorflow\_p36

for TensorFlow 2(+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate tensorflow2\_p36

for TensorFlow 2(+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate tensorflow2\_p27

for TensorFlow 2.3 with Python3.7 (CUDA 10.2 and Intel MKL-DNN) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate tensorflow2\_latest\_p37

for PyTorch 1.4 with Python3 (CUDA 10.1 and Intel MKL) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate pytorch\_p36

for PyTorch 1.4 with Python2 (CUDA 10.1 and Intel MKL) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate pytorch\_p27

for PyTorch 1.6 with Python3 (CUDA 10.1 and Intel MKL) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate pytorch\_latest\_p36

for PyTorch (+AWS Neuron) with Python3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate aws\_neuron\_pytorch\_p36

for Chainer with Python2 (CUDA 10.0 and Intel iDeep) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate chainer\_p27

for Chainer with Python3 (CUDA 10.0 and Intel iDeep) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate chainer\_p36

for base Python2 (CUDA 10.0) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate python2

for base Python3 (CUDA 10.0) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ source activate python3

We want the pytorch version 1.6 for this exercise. This has the pytorch native version of mixed precision available. Lets activat this.

source activate pytorch\_latest\_p36

You will need to do this each time you login.

Now in the server, lets get the packages we need before starting our homework.

pip install transformers

git clone [https://github.com/MIDS-scaling-up/v2 w251](https://github.com/MIDS-scaling-up/v2%20w251)

cd w251/week06/hw/

jupyter notebook --ip=0.0.0.0 --no-browser

You will get a message like below.

[I 14:49:26.343 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[C 14:49:26.347 NotebookApp]

To access the notebook, open this file in a browser:

file:///home/ubuntu/.local/share/jupyter/runtime/nbserver-16772-open.html

Or copy and paste one of these URLs:

http://ip-172-31-30-27:8888/?token=37b8bba7d420a4073b0a2169a24a995cde3ede8329f5ef9b

or <http://127.0.0.1:8888/?token=37b8bba7d420a4073b0a2169a24a995cde3ede8329f5ef9b>

To access the book use the public ip found above along with the URL, like follows : <http://54.194.227.21:8888/?token=37b8bba7d420a4073b0a2169a24a995cde3ede8329f5ef9b>

nohup jupyter notebook --ip=0.0.0.0 --no-browser &> nohup\_log.out &

If you would like to kill the nohup session later, you can use `lsof | grep nohup\_log.out` to find find which pid is using this file. Then kill that pip.

For example, if your ouput is like below, you should run `kill 16772` and `kill 16820`,

ZMQbg/0 16772 16822 ubuntu 2w REG 202,1 2231 539934 /home/ubuntu/w251/week06/hw/nohup\_log.out

ZMQbg/1 16820 ubuntu 1w REG 202,1 2231 539934 /home/ubuntu/w251/week06/hw/nohup\_log.out

ZMQbg/1 16820 ubuntu 2w REG 202,1 2231 539934 /home/ubuntu/w251/week06/hw/nohup\_log.out