Deep Learning - Recitation (Fall 2018)
What is Amazon Web Services (AWS)

Cloud Based Computational Resource
What does AWS offer?

Many many things but here are the two main things to care about for DL...

**EC2 - Compute Resources**

Train the models

**S3 - Data Storage**

Store training data, models, etc
EC2 - What kinds of machines are available?

Different types and different subtypes (you can mix and match what you want)... Here are the ones you may care about

**General Purpose:**
- T2 - Webservices
- M3/M4 - Databases, Fileservers, etc

**Compute Optimized:**
- C2 - Multiplayer Gaming Servers, scientific computing
- C3/C4 - Ad serving machines, MMO servers, etc

**GPU Optimized:**
- P3/P2 - Machine Learning
- G3 - Fluid dynamics, graphics rendering, etc

Machine sizes - nano, micro, medium, large, xlarge, 2xlarge, ..., 16xlarge
EC2 - What kinds of machines are available?

Different types and different subtypes (you can mix and match what you want)...

- **T2.nano**
  - 1 vCPU (Xeon)
  - 0.5 GB RAM

- **M4.large**
  - 2 vCPU (Xeon)
  - 8 GB RAM

- **C4.8xlarge**
  - 36 vCPU (Xeon)
  - 60 GB RAM

- **P2.16xlarge**
  - 64 vCPU (Xeon)
  - 16 GPU (Nvidia K80)
  - 192 GB GPU-Memory
  - 732 GB RAM
EC2 - So what do we put on these machines?

Amazon Machine Instances (AMIs)

- Virtual images of existing machines
  - You can create an image of your machine
    - Transfer it to a different machine
    - Save it as a backup

- Use cases
  - Software packages that are incredibly difficult to install
  - Need to create multiple different machines with the exact same data for parameters servers
  - Load balancing - create a new machine with the same AMI to be used in a different region depending on load
Now you know what AWS is and what you can do with it
Who needs to know how to do this and why

- Anyone who doesn’t have a NVIDIA GPU
- Anyone who doesn’t have their GPU configured to be correctly used by Pytorch/Tensorflow/etc.
- GPU is good for training not processing data or if the code is not configured to leverage GPU
- PS: There are screenshots for each step that go with this guide (steps are numbered)
Steps you need to get started

1. Create/sign in to your AWS account (aws.amazon.com)
2. Click on the region in the top right and then click either US East (N. Virginia) or US West (Oregon) GPU instances are only available in certain regions
3. Click on EC2 on the right
4. Click on running instances
5. Click on launch instances (top right)
6. Choose the Deep Learning AMI (ami-0faada77)
7. Pick a p2.xlarge instance (you can explore other instances if you want but these are the cheapest and most appropriate for personal use IMO)
8. Do nothing at the Configure Instance Details

9. Configure the instance to have at least 75GB storage

10. Do nothing at add tags

11. Do nothing at security group (they will create a SSH security group for you) (If you want to set up a Jupyter notebook you will have to add a couple of ports to open)

12. AWS Step 7 Review and Launch (click launch)
Step 1, 2

Click on “Services” in the top left

Then, under “Compute”, select “EC2”
Step 3

Make sure you are in the “US West (Oregon)” region

Then Click on “Running Instances”
Step 4,5

Here you see your current instances

(I have 1 that is stopped, you shouldn’t have any)

Here are the details of that instance
Step 6

Select the Deep Learning AMI. It should have CUDA and all of the ML frameworks installed (ami-0faada77)

If you can’t find it just search Deep Learning AMI (Ubuntu)

https://aws.amazon.com/marketplace/pp/B077GCH38C
Step 7

Select the t2-micro for the recitation because it is “free tier eligible”, we will explain GPU access in the later slides.

Hit Next after selecting the instance.
We don’t really use this page just hit next.
Step 9

To use the Deep Learning AMI you need at least 75GB so just leave this alone for now, in some later assignments you’re probably going to have to go bigger.

Hit Next
Step 10

We don't really use tags just hit next don't bother with this page

Hit next
Step 11

Leave this page alone unless you know what you are doing. To get a Jupyter notebook to run on your AWS instance you need to add a couple more ports to open, if we have time that this step will covered for that.

Hit Review and Launch
Step 12

Step 7: Review Instance Launch

To launch an instance that’s eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. Learn more about free usage tier eligibility and usage restrictions.

AMI Details
- Deep Learning AMI (Ubuntu) Version 9.0 - ami-0faada77
  - Free for eligible
  - Comes with latest binaries of deep learning frameworks pre-installed in separate virtual environments: MXNet, TensorFlow, Caffe, Caffe2, PyTorch, Keras, Chainer, Theano and CNTK. Fully-configured with NVIDIA CUDA, cuDNN and NCCL as well as Intel MKL-DNN
- Root Device Type: ebs
- Virtualization type: hvm

Instance Type
- p2.xlarge
  - ECU: 11.75
  - vCPUs: 4
  - Memory (GiB): 61
  - Instance Storage (GiB): EBS only
  - EBS-Optimized Available: Yes
  - Network Performance: High

Security Groups
- Security group name: launch-wizard-1
- Description: launch-wizard-1 created 2018-09-21T12:31:49.774-07:00

Instance Details
- SSH
  - Protocol: TCP
  - Port Range: 22
  - Source: 0.0.0.0/0

Tags

Hit launch
You need permission to get a GPU

- You will run into an error launching this instance unless you have already requested an AWS GPU instance before
- Looks like this:
Follow the instructions

- Visit the link provided by the error page in the previous slide
  https://aws.amazon.com/contact-us/ec2-request
- Request a reasonable number of p2.xlarge instances
- Give them a reason why, give them a way to contact you
- Finish and wait a couple days for them to give you permission
Screenshots for p2.xlarge request
I need GPUs for some deep learning projects that I do on the side and I don't have a GPU of my own. Will be using it for Kaggle competitions.
Billing, AWS coupons… i.e. don’t go broke

- Amazon charges you for a lot of services, the most significant is having running instances (GPU time is NOT cheap) MAKE SURE YOU STOP YOUR INSTANCE WHEN YOU ARE DONE USING IT
- Stopping an instance is basically shutting down a computer; the saved files persist, etc...
- Terminating an instance deletes the entire machine only do this when you completely done with whatever you are doing with your AWS instance
- To stop/terminate an instance simply go to step 4 from tutorial then right click on the instance and under the option instance state, hit stop/terminate, the next slide will show you how
- We will provide you with AWS tokens, you need to redeem them from Billing, the next next slide will show you how
Stopping the Instance

When you’re done with your instance, make sure you stop it or you will run out of AWS credits.
Billing

Under your name hit My Billing Dashboard
Billing/Redeeming Tokens

This is where you can see your bill. Hit Credits if you want to redeem your AWS tokens.
Redemption Page

This is where you redeem your tokens, just follow the instructions and enter your token.
TMUX makes your life easier

- Use tmux if you don’t want to wait on your computer like its your child
- After SSHing into an AWS instance you can type in the command tmux
- Then run anything as would normally
- Usually when you run something on SSH if your connection dies (ie when your computer sleeps/etc) the program you are running in the SSH session dies too
- tmux prevents this (you can close out the session whenever)
- To reconnect just SSH back in and then do: tmux ls
- And then you should see your session (your first unnamed session is 0)
- tmux attach-session -t 0
Using S3

- Instead of reading through more powerpoint you can just follow Amazon’s tutorial (first one is for the GUI, second one is for the CLI)
- You would use S3 if you don’t have enough storage on your local machine and you don’t want to store it on EC2 storage (EBS is the EC2 HDD service) either
- Good for sharing large files has high bandwidth to AWS instances in the same region
- We might use it to share data files you with guys
- https://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html
Can we develop scalable deep learning code that only relies on resource requirements and is agnostic to a specific system’s topology?
Ray - A Framework for Distributed Deep Learning

- Write code with respect to available resources (CPUs, GPUs, etc).
- Don’t worry about placement (local host, on-demand cloud instances, spot instances, etc).
- Write everything in a pythonic, declarative manner.
- Ray will take care of scheduling, serialization, and communication.
Ray - A Framework for Distributed Deep Learning
import time

def f1():
    time.sleep(1)

@ray.remote
def f2():
    time.sleep(1)

# The following takes ten seconds.
[f1() for _ in range(10)]

# The following takes one second (assuming the system has at least ten CPUs).
ray.get([f2.remote() for _ in range(10)])
@ray.remote
def f(x):
    return x + 1

x = f.remote(0)
y = f.remote(x)
z = f.remote(y)
ray.get(z)  # 3
Install Ray

```bash
>>> pip install ray
```
Registering AWS credentials on your local machine

1. Open the IAM console.
2. In the navigation pane of the console, choose Users.
3. Choose your IAM user name (not the check box).
4. Choose the Security credentials tab and then choose Create access key.
5. To see the new access key, choose Show. Your credentials will look something like this:
   - Access key ID: AKIAIOSFODNN7EXAMPLE
   - Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. To download the key pair, choose Download .csv file. Store the keys in a secure location.
Registering AWS credentials on your local machine

- You will need to provide AWS’s CLI client with the credentials you just created.
- First install the awscli and boto3 packages with pip.
- Next add the ~/.aws/credentials file with the following template

```
[default]
aws_access_key_id = [YOUR_ACCESS_KEY_ID]
aws_secret_access_key = [YOUR_SECRET_ACCESS_KEY]
```
Specify your cluster template

- Provide Ray your configuration preferences and let it do the EC2 setup work for you, even if you only need a single node in the cluster.

- Configuration template provided in the the recitation1 directory in the 11-785 tutorials repo on GitHub as dev_template.yml
  - Specify instance type, number of workers, AMI, volume size, and more
Launch the cluster

- Copy the template into a file you can edit.
  
  ```
  $ cp dev_template.yml dev.yml
  ```

- Make your changes to the config and then launch the cluster.
  
  ```
  $ ray create_or_update dev.yml
  ```

- The output will give you a command to log into the cluster with.
Stop the cluster

- Stop and terminate all cluster instances.

$ ray teardown dev.yml

**WARNING:** This will destroy all cluster instance and volumes. Make sure you copy your data back to your host machine first.

Always check the AWS console to ensure your instances have been stopped/terminated. You have been warned!