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:

<table>
<thead>
<tr>
<th>General Purpose:</th>
<th>Compute Optimized:</th>
<th>GPU Optimized:</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2 - Webservices</td>
<td>C2 - Multiplayer Gaming Servers, scientific computing</td>
<td>P3/P2 - Machine Learning</td>
</tr>
<tr>
<td>M3/M4 - Databases, Fileservers, etc</td>
<td>C3/C4 - Ad serving machines, MMO servers, etc</td>
<td>G3 - Fluid dynamics, graphics rendering, etc</td>
</tr>
</tbody>
</table>

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
Create An Account:

https://aws.amazon.com

Click “Sign Up” in the top right and follow the instructions

(If you already have an account you can skip this step)
Let’s set up a basic machine

Click on “Services” in the top left

Then, under “Compute”, select “EC2”
Let’s set up a basic machine

Then Click on “Running Instances”

Make sure you are in the “US West (Oregon)” region
Let’s set up a basic machine

Here you see your current instances
(I have 1 that is stopped, you shouldn’t have any)

Here are the details of that instance
Let’s set up a basic machine

Click “Launch Instance”
Let’s set up a basic machine

Step 1: Choose an Amazon Machine Image (AMI)

**Launch a database using RDS**

- **Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** - ami-078e277685f2451f6 (64-bit x86) / ami-05e1b2ac3b47896f8 (64-bit Arm)
- **Microsoft Windows Server 2016 Base** - ami-019e09815e07ce349
- **Deep Learning AMI (Ubuntu) Version 20.0** - ami-0d0f0945e030e3ea
- **Deep Learning AMI (Amazon Linux) Version 20.0** - ami-0305a0d7a98489e06
- **Deep Learning Base AMI (Ubuntu) Version 14.0** - ami-015eb48ac552e435f
  - Comes with foundational platform of NVIDIA CUDA, cuDNN, NCCL, GPU Drivers, Intel MKL-DNN and other system libraries to deploy your own custom deep learning environment. For a fully managed experience, check: https://aws.amazon.com/sagemaker
Let’s set up a basic machine

Select the t2-micro because it is “free tier eligible”

Select Next
Let's set up a basic machine

### 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

**Purchasing option**
- Request Spot instances

**Network**
- `vpc-10010101 (default)`

**Subnet**
- No preference (default subnet in any Availability Zone)

**Auto-assigned Public IP**
- Use subnet setting (Enable)

**IAM role**
- None

**Shutdown behavior**
- Stop

**Enable termination protection**
- None

**Monitoring**
- None

**Tenancy**
- Shared - Run a shared hardware instance

---

Just select next
Let’s set up a basic machine

Make sure you choose 8 GB of SSD Storage Space

Select next
Let’s set up a basic machine

### Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Instances</th>
<th>Volumes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(127 characters maximum)</td>
<td>(255 characters maximum)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This resource currently has no tags

Choose the Add tag button or [click to add a Name tag](https://docs.aws.amazon.com/). Make sure your IAM policy includes permissions to create tags.

Select next
Let’s set up a basic machine

Make sure you have an SSH rule set (This should be default) & a TCP rule for 8888 so you can connect your Ipython Notebook

You can set HTTP or other rules here too if you want

Select Review and Launch
Let’s set up a basic machine

Take one last look to make sure you are happy with everything...

Select Launch
Let’s set up a basic machine

This window allows you to create a private key to access your machine… when you ssh you will need it.

Enter a name for your private key.

Download your key and save it.

Make sure it’s in a place where it won’t get lost - this key is the only way to connect to this specific instance!
Now you have a basic AWS machine up and running
Now let’s connect to it

Your instance is now identified by its IP address. You can ssh to your instance by using this public IP address and your address key.

Let’s start by changing the instance type. For student accounts, t2.micro is free-tier eligible, but let’s still start on a t2.nano instance.

Remember your instance’s IP address changes every time you restart it.
Connect to launched instance

Right-click and click connect

The ssh command shown includes the currently assigned IP address.
This will change at each launch!

Note - the ssh command will use the path to the key associated with this instance.

Important - will need to run chmod command every time you have a new key
Set instance type

Right-click on your instance under the instance tab, go to instance settings and change instance type to t2.nano
Set instance type

Right-click on your instance under the instance tab, go to instance settings and change instance type to t2.nano
Note

You can spin up a t2.micro for development (which allows access to all the packages in the AMI and can put all your data on the instance), and then change the instance type to a p2.xlarge for GPU access (training).

p2.xlarge is what I used for all my training, about $1 per hour, so with 3 credits of $50 that’s more than enough hours of training for almost all cases.
Launch instance

Right-click and Start your instance.
Launch instance

Right-click and Start your instance.

This will start the process of allocating resources to your instance. Once this is completed, your instance will be running and you can connect to it.

Stopping this instance removes the compute associated with the current session.

Do NOT terminate! This will wipe the slate clean. Terminate ONLY when you are sure of ‘throwing away’ the data.
Now you have a running AWS machine and you can connect
Now you know how to use machines on EC2
Let’s run a trivial little problem
import torch

def main():
    GPU = torch.cuda.is_available()
    mat_size = (100, 100)
    cpu_mat_0 = torch.zeros(size = mat_size)
    cpu_mat_1 = torch.ones(size = mat_size)
    gpu_mat_0 = torch.zeros(size = mat_size)
    gpu_mat_1 = torch.ones(size = mat_size)

    if GPU:
        gpu_mat_0 = gpu_mat_0.cuda()
        gpu_mat_1 = gpu_mat_1.cuda()
    print("Using GPU")

    cpu_res = cpu_mat_0 + cpu_mat_1
    gpu_res = gpu_mat_0 + gpu_mat_1

    try:
        print(bool(torch.all(cpu_res == gpu_res)))
        return 0
    except:
        print("If using GPU, should be here")
        gpu_res = gpu_res.detach().cpu()  # detach is for gradient computations
        print(bool(torch.all(cpu_res == gpu_res)))
        return 0

main()
And let’s shut it down so you don’t get billed

If you stop the instance you can just start the instance and resume whenever you want.

If you are done, terminate.
Now let's redeem some AWS credits
Now let's redeem some AWS credits.
Now let's redeem some AWS credits.
Now you know how to use machines on EC2 and you can afford to use the expensive ones.
Finally, some useful tidbits

Remember to refer back here later in the semester, likely won’t remember all this right now
Editing Volume

If you run out of space, you can adjust your volume without shutting off the instance, happened to me on hw2p2


Here’s How to Connect to Jupyter Notebook

Call ssh -N -L localhost:8888:localhost:8887 -i ~/path/to/key ubuntu@ip

Create new terminal window, ssh into instance

In ssh window, source activate pytorch_p36

Call jupyter notebook --no-browser --port=8887

Go to browser, enter localhost:8888
Parting wisdom

● Remember to shut down your machines
  ○ Just because you don’t have an open ssh connection doesn’t mean your machine is off
  ○ When you’re not running code, Stop; when you can get rid of the data, Terminate

● Use PyTorch on the previously given AMI
  ○ After you ssh into the instance, run command ‘source activate pytorch_p36’
  ○ Without this you cannot import torch
  ○ Unless you are already very comfortable with tensorflow, it is much easier to get help from TAs on PyTorch
Parting wisdom

- You need permission to launch a GPU instance
  - You will need to create a support ticket to launch GPU Machines
  - If you attempt to launch on you will be guided through the process
  - You are typically only allowed to launch 1 GPU machine at a time

- Only launch an expensive instance when it is time to train, not develop
  - Launch a basic instance with everything you need to develop and test your code
  - When it comes time to train your system for real, then launch a decked out instance
  - Saves you money
Parting wisdom

If you want to use a local IDE instead of VIM on your ssh window, you can develop locally and run an scp to transfer the file to the aws instance

scp -i ~/path/to/key ~/path/to/file/ ubuntu@ec2...

If you use jupyter notebook connected to aws, no need to worry about this.

While writing code, I kept a note with common commands that I could just change the IP address on, such as an ssh command, and scp of my local code