Recitation 0F
AWS Fundamentals

Ruimeng Chang, Arjun Chauhan

Overview

1. AWS Basics
    a. Regions
    b. Budgets
    c. Credits
    d. Limits
2. Using EC2 instances (Cloud VMs)
    a. AMIs
    b. Instance types
    c. Security Groups
    d. SSHing and Elastic IP
    e. Stopping and Terminating Instances

3. EC2 Setup
    a. SSH configuration
    b. Persistent Storage (EFS & EBS)
    c. Environment Setup

4. Remote Notebook
    a. Screen Command
    b. Jupyter Lab

Part 1 - AWS Basics

Part 2 - Configuring EC2 Instances

AMIs : Amazon Machine Images

Recommended DL AMIs for EC2 Instances

	AWS Deep Learning AMI (Ubuntu 18.04)	**AWS Deep Learning Base AMI (Ubuntu 18.04)**
Boot Volume Size	≥ 100GB	≥ 60GB
Config Includes	● Drivers, CUDA, CuDNN ● Anaconda env 1. ML packages 2. ML/DL Frameworks	● Drivers, CUDA, CuDNN
Occupied Boot Volume Space	88 GB	42 GB

Part 2 - Configuring EC2 Instances

Recommended DL AMIs for EC2 Instances

Type	vCores	RAM(GB)	GPU	Additional Specs	Suitable For
t2.micro	1	1	-	Free tier available	Test only
c5.xlarge	4	8	-	-	Basic code dev when GPU instances aren't available
g4xdn.xlarge	4	16	1 x Nvidia T4 16GB VRAM	125Gb of additional ephemeral NVMe drives	Code development and model training
p3.2xlarge	8	61	1 x Nvidia Tesla V100 16GB VRAM	Expensive	Accelerated training of large models

Credits : Past TA : Yash Gaurav

Part 2 - Configuring EC2 Instances

On Demand vs Spot Instances

	On Demand Instance	Persistent Spot Instance	One-time Spot Instance
Spin up	Should always work	Subject to Spot Capacity and Price	Subject to spot capacity and price
Interruptions Action	-	Stopped/Hibernated by AWS (EBS Volume Retained)	Terminated by AWS
Cost	Regular	Upto 90% Discount	

	Stop = Shutting down computer	Terminate = Throwing your computer into the Pacific Ocean
Interruptions Actions	Unaffected	Permanently Erased
Additional EBS Disks/EFS	Unaffected (AWS still charges you for memory)	
Ephemeral Drives	Permanently Erased	

Part 2 - Configuring EC2 Instances

Recommended AWS GPU Instances

- Amazon EC2 P3 Instances have up to 8 NVIDIA Tesla V100 GPUs.
- Amazon EC2 P4 Instances have up to 8 NVIDIA Tesla A100 GPUs.
- Amazon EC2 G3 Instances have up to 4 NVIDIA Tesla M60 GPUs.
- Amazon EC2 G4 Instances have up to 4 NVIDIA T4 GPUs.
- Amazon EC2 G5 Instances have up to 8 NVIDIA A10G GPUs.
- Amazon EC2 G5g Instances have Arm-based AWS Graviton2 processors.

Part 3 - Set up SSH config

In `~/.ssh/config`, add a section

```
Host <instance public DNS>
  User ubuntu
  IdentityFile ~/.ssh/<.pem name>
```

Part 3 - Instance Storage setup

```
lsblk
sudo mkfs -t xfs /dev/<device name>
sudo mkdir /data
sudo mount /dev/<device name> /data
cd /data
sudo chmod go+rw .
cd ..
```

- Normally, the device name should be nvme1n1
- Instance storage will be reset after stopping the instance.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html

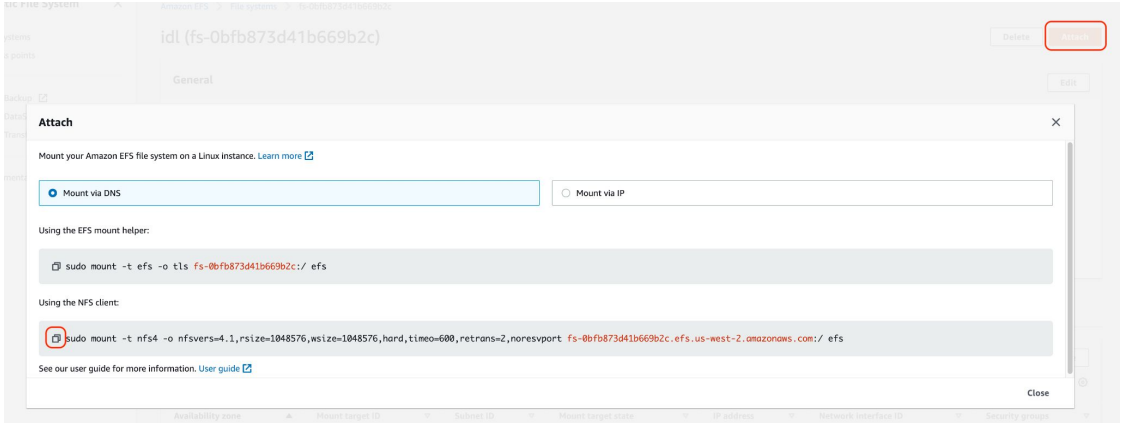
Part 3 - Persistent Storage (EFS & EBS)

	Elastic File System (EFS)	Elastic Block Store (EBS)
Storage Type	File Storage	Block storage
Storage Size	Unlimited	16TB
Access	Multiple instances in one region	Single instance in one Availability Zone
Performance	• Baseline 3 IOPS/GB	• Scalable • 3GB/s for all connected client
Price (US East - Ohio)	• Storage (gp3): $0.08/GB-month • $0.04/provisioned MB/s-month	• Storage: $0.30/GB-month • $6.00/provisioned MB/s-month

https://aws.amazon.com/ebs/pricing/
https://aws.amazon.com/efs/pricing/

Part 3 - Persistent Storage (EFS & EBS)

- On EFS Dashboard, select “Create File System”
- Click File System ID and go to the details page, select “Network - Manage”, update the security groups to include the security group for your EC2 instance
- Copy the command to attach through NFS



Part 3 - Persistent Storage (EFS & EBS)

- Go to SSH connection to EC2 instance, mount the File System
  - `sudo mkdir /efs`
    Paste the command copied from last step
    `cd /efs`
    `sudo chmod go+rw .`

Note: If you got a timeout error, double check (1) if the security group is configured correctly; (2) if the security group has an inbound rule for NFS

Part 3 - Persistent Storage (EFS & EBS)

- On EC2 dashboard, select “Elastic Block Store - Volumes”
- Click “Create Volume”, modify the availability zone to be the same as your running EC2 instance
- Right-click on the created volume, select “Attach Volume” and choose your EC2 instance
- Go to SSH connection to EC2 instance, mount the volume
  - ```
    lsblk
    sudo mkfs -t xfs /dev/<device name>
    sudo mkdir /ebs
    sudo mount /dev/<device name> /ebs
    ```
- After stopping & restarting instance, only need to remount the volume
  - ```
    sudo mount /dev/<device name> /ebs
    ```

Part 3 - Environment setup (conda, kaggle)

- Install miniconda on Base AMI
  - Get download link from https://docs.conda.io/en/latest/miniconda.html, right click and select “Copy Link Address”
  - `wget <downloadlink>`
    `sha256sum <filename>`
    `bash <filename>`
  - Restart terminal

Part 3 - Environment setup (conda, kaggle)

- Create environment
  - `conda create -n idl python==3.8`
    `conda activate idl`
- Install packages
  - `pip install kaggle`
    `pip install numpy`
    etc.
- Setup kaggle
  - `mkdir ~/.kaggle`
    `echo '{"username":"<your-username>","key":"<your-key>"}' > ~/.kaggle`

Part 3 - Environment & data backup

Environment

- Find the path for your conda environment through
  - `conda info --envs # path is /opt/conda/`
- Save folder for conda environments in /efs or /ebs
  - `cd /opt/conda/envs/`
  - `tar -cf /efs/conda-env.tar <your_env_name>`
- Unzip in new instance
  - `cd /opt/conda/envs`
    `tar -xf /efs/conda-env.tar`
- You can also save .kaggle, .bashrc if needed.

Data

- Save trained models in persistent storage, you can restart training by loading intermediate checkpoints if training is interrupted

Part 4 - Screen Command

- Create a new session
  - `screen -S`
  - `screen -S <session_name>`
- Detach from current running session
  - Ctrl+A and then D
- Attach to an existing session
  - `screen -r`
  - `screen -r <session_name>`

Part 4 - Jupyter Lab

- Install
  - `pip install jupyter jupyterlab`
- Generate hashed password
  - ```
    from notebook.auth import passwd
    my_password = "<your_password>" # set your desired password here
    hashed_password = passwd(passphrase=my_password, algorithm='sha256')
    print(hashed_password) # copy the hashed password
    ```
- Create configuration file
  - `mkdir ~/.jupyter`
  - Create `jupyter_server_config.py` inside the folder
  - Paste

    ```
    c.ServerApp.ip = '*' # bind to any network interface
    c.ServerApp.password = u'sha256:<your hashed password here>'
    c.ServerApp.open_browser = False
    c.ServerApp.port = 8888 # or any other ports you'd like
    ```
- Backup by saving data in `~/.jupyter`
    ```