AWS

Objectives

- What is AWS
  - EC2
  - EBS
  - S3
  - Other stuff
- Configure your system to interact with AWS
- Configure and launch an EC2 instance
- Use SSH to access EC2
- Access S3

AWS

AWS (Amazon Web Services) provides on-demand use of cloud computing:

- ➔ Can access scalable hardware for fraction of cost to build resource yourself
- ➔ Allows for easy creation of new ventures
- ➔ Pay only for what you use
- ➔ Deal with spikes in demand
- ➔ Secure, reliable, and cost effective

AWS skills are valuable and highly sought after

Core Services

- ➔ Elastic Compute Cloud (EC2): computers in the cloud
- ➔ Elastic Block Store (EBS): virtual hard drives for those EC2 machines
- ➔ Simple Storage Solution (S3): long-term large scale data storage

- ➔ And so many other things

Overview of core services

EC2

EC2 is a service which allows for custom configurable hardware provided as you need it.

Partial Glossary:

- ➔ Instance: a single machine you can use, comes in various sizes with specific labels e.g. t2.micro, m3.xlarge, g2.2xlarge
- ➔ Amazon Machine Image (AMI): an OS you can run on an instance, can also include pre-loaded software and data files
- ➔ Region: A geographic region, important for billing/communication, e.g. us-east, us-west-2
- ➔ Availability Zone (AZ): a specific subset of a region, often a data center, e.g. us-west-2a

EBS

EBS provides virtual hard drives that can be attached to EC2 instances

- ➔ Provides low latency data storage and acess for EC2
- ➔ Data can be persisted even when not connected to an active EC2 instance
- ➔ You pay both for size of drive, and for number of read/write operations sent to it
- ➔ Built-in redundancy
- ➔ Lower latency option than S3, but more expensive

S3

- Based on buckets
  - Buckets act like individual file systems, can store files and directories in them
- Can specify permissions for who can access buckets and/or individual files/folders within buckets
- Access via web interface, the AWS command line interface (CLI) or through API
- Cheap, redundant, long term storage
- Higher latency than EBS

Interacting with aws

AWS CLI

Amazon wrote its own command line tool that you can install to interact with AWS

- ➔ Works well for many basic interactions with aws
- ➔ Is not required to use AWS
- ➔ Allows you to manage EC2 instances and access S3 resources without leaving terminal
- ➔ Very useful once in an AWS instance (where you have no GUI)

Installing the CLI

- ➢ curl “https://s3.amazonaws.com/aws-cli/awscli-bundle.zip -o “awscli-bundle.zip”
- ➢ unzip awscli-bundle.zip
- ➢ sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws

Can also use brew/pip

brew install awscli

pip install awscli

That’s all well and good for sheep, but what are we to do?

To interact with AWS you will need credentials

- ➔ Login to AWS
- ➔ Click *Your Account* in upper right and select *Security Credentials*
- ➔ Select *Users*
- ➔ Select your username and click *User Actions* > *Manage Access Keys*
  - ◆ If you haven’t created a User for AWS yet you can, it is the AWS recommended method
- ➔ Create your Access Keys
  - ◆ Important note, you cannot view secret access keys again after creation

Note about keys

- ➔ With your AWS access keys anyone can do anything they want with your AWS account
- ➔ AWS costs money
- ➔ Therefore, letting people have access to your access key means they can spend your money
- ➔ There are people constantly scraping github looking for sequences that look like AWS access keys
- ➔ DO NOT PUT YOUR KEYS ON GITHUB
- ➔ Also, be careful with them

So, about these keys then

- ➔ You will need to keep your secret access key on your local system to use it
  - ◆ Remember you can only view it when initially created
- ➔ Various services will look for it in certain places
  - ◆ ~/.aws/credentials
  - ◆ ~/.boto, ~/etc/boto.cfg
  - ◆ Or can put it in environment variables
- ➔ Can have multiple profiles with AWS CLI and other tools (boto for one)
- ➔ Can run aws configure if you have the CLI installed to run through some of these setups
- ➔ Use the credentials.csv you downloaded when setting up your security credentials to move your secret access key to the appropriate place

Some Live Demoing will happen here

On to EC2

Access to EC2 instances is navigated using ssh and *Key Pairs*

- ➔ *Key Pairs* are not the same as the AWS access keys and secret keys we spent the last few minutes discussing
- ➔ ssh stands for secure shell, it is a secure process for logging into the shell on remote servers
- ➔ Can use ssh to access any remote machine set up to accept ssh connections
- ➔ Can also use scp and sftp to transfer files using the same protocol as ssh

Public key encryption

A general discussion of encryption, while interesting, is outside of scope of this course. But, a brief diversion into ssh and public-key encryption is fun/informative.

- ➔ Generate a **key pair**
  - ◆ A private key and a public key
- ➔ You share certain pieces of information with your communication partner
  - ◆ A common number, each of your public keys, and the result of an operation using common number, private key, and public key
- ➔ Using this public transfer, you each generate the same secret, that is computationally intractable for an attacker to discover
- ➔ This common secret is used to encrypt the communication

ssh and public key encryption

- You can create ssh key pairs for different tasks (AWS, github, other stuff)
- Store them on your local system, usually in ~/.ssh
- Can revoke individual key pair if a service becomes compromised
- For AWS can create key-pair in ec2 console, then save the .pem file it generates locally
- Run chmod 400 ~/.ssh/key_pair.pem
- Can set detailed ssh config files
    - Create alias for specific instances
    - Forward X11
    - Setup alternate configurations
-

example ssh config

```
Host master
	HostName ec2-52-2-154-253.compute-1.amazonaws.com
	User ubuntu
	ForwardAgent yes
	ForwardX11Trusted yes
	TCPKeepAlive yes
	IdentityFile /Users/lemur/.ssh/key_pair.pem
```

Access EC2 instance

- ➔ Launch EC2 instance in console
  - ◆ YAY LIVE DEMOS
- ➔ Use ssh to connect to the instance's public DNS
  - ◆ You can find that in the ec2 dashboard

```
ssh -X -i ~/.ssh/key_pair.pem ubuntu@<public_dns>
```

Or

```
ssh -X -i ~/.ssh/key_pair.pem ec2user@<public_dns>
```

SCP

Can use scp just like you would use cp locally

```
scp -i ~/.ssh/key_pair.pem ./some_file.txt ubuntu@<ip>:/home/ubuntu/data
```

tmux ftw

- EC2 instances are headless servers
    - Cannot open new windows
- So, if you do something like launch an ipython notebook, or start a webserver, your terminal gets locked up
    - There are often ways to launch these things in the background, but sometimes you want easy access to the logs, or other reasons not to do that
- Also, if you log out, whatever processes are running in your current terminal terminate
- Enter tmux, which allows you to run background terminal windows, and disconnect but leave a process running
- DEMO time

EC2 tips

- Match the hardware to your problem
  - No need to grab a 60 CPU 128GB ram machine for fitting a linear regression to 10000 datapoints
- If in doubt use Ubunutu
  - It is relatively user friendly
  - More importantly, it is widely used, and thus problems are easily googlable
- Use tmux
  - Nothing worse than getting halfway through a long job and having it die on you because of a disconnect
- Be paranoid
  - Amazon will do things, sometimes with reason, sometimes without
    - A student in my cohort was told by amazon his job was overheating their datacenter and shut it down
- Important data should be kept in a persistent location

S3

- Check out Asim’s lecture in DSI_Lectures for great introduction to S3
- S3 is made of buckets (not really but roll with me here)
- Buckets hold files and/or folders
- Bucket names have to be globally unique, and follow some rules, among them:
  - Name is all lowercase
  - No leading or trailing ‘.’
  - No / or other non dns compliant characters

boto

Can access S3 in python with boto

Make sure your boto is up to date

It will read your credentials if you stored them in the places we talked about earlier

BOTO DEMO GOES HERE