Amazon Web Services (AWS)

Objectives

- Describe what AWS is and where it came from
- Describe how we (the DSI) typically use AWS
  - Workflow
- Demonstrate launching an EC2 instance
- Demonstrate making and accessing S3 buckets using Python (Boto3)
- In the assignment
  - Make an S3 bucket, and upload and download data from bucket using Boto3
  - Run a script locally, then launch an EC2 instance, upload the script, run it in the cloud, and bring down the results locally

Amazon Web Services (AWS)

- AWS is a cloud services platform.
    - What are cloud services?
- Cloud services are the delivery of computing services—servers, storage, databases, networking, software, analytics, and more—over the Internet.
- Advantages: accessibility, dynamic scaling, no need for hardware or support staff, small upfront cost
- Disadvantages: security (someone else owns the server), recurring cost for services

How AWS came to be

- In 2000 Amazon was struggling with scaling problems
- Started building internal systems to address their problems
- Realized that their solutions were useful to others, too
- Amazon Web Services launched in 2002, Elastic Compute Cloud in 2006
- Was the first widely accessible cloud computing infrastructure service
- Public cloud revenue market share: 47% AWS, 10% Azure, 4% Google Cloud

AWS services we typically use in the DSI

- Simple Storage Service (S3)
  - Long-term big data storage for the internet.
  - About S3. Creating an S3 Bucket. Pricing. FAQs.
  - In DSI where we store data and maybe final models & results.

- Elastic Compute Cloud (EC2)
  - Provides secure, resizable compute capacity in the cloud.
  - About EC2. Instance types. Launching an EC2 Instance. Pricing. FAQs.
  - In DSI where we train our models, deploy applications (Flask web apps, databases).
  - If you are training a neural net, you'll want to use GPU compute instances.
  - Use a Spark cluster (on Amazon Elastic Map Reduce) for distributed computing.

- AWS offers many, many more services. Look.

DSI workflow

1. Upload/access data on S3 (or if small enough, locally)
2. Use pandas or boto to pull a subset of the data down to your local machine.
   a. Note in pandas the option for an *S3 bucket URL*, and an option for *chunksize*.
3. Develop a script to train your model locally on a subset of the data, save results to a file.
4. Start an EC2 instance using the AWS Management Console (see next slide)
   a. If you want to access S3, give it a Role that allows it to read and write to S3
5. Connect to your instance from your local terminal using ssh.
6. Upload your script to EC2 (`scp`, or `git clone`)
7. Run script on EC2 on the full data set.
   a. Use screen before running script so instance keeps working after closing terminal
   b. Ideally data is not too big for the Elastic Block Storage (EBS) associated with the EC2 instance, but if it is you can still just read in data from S3.
8. Write results, either locally (on server) or to S3 (boto 3). Get results locally (on laptop) with `scp`.
9. Terminate your EC2 instance and EBS.

Demo: How to Launch an AWS EC2 instance

Follow this guide.

1. Choose a machine image (AMI) - look for anaconda3 & Ubuntu
2. Choose an instance type (general purpose, GPU)
3. Configure instance (accept defaults)
    a. **Important! If you want your EC2 instance to be able to access S3 - add a S3 role to your IAM user and select it here.**
4. Add storage (accept defaults)
5. Add tags (give it a Name)
6. Give it a security group
    a. accept existing for SSH, if you want an application available to others on the web, add a new **Custom TCP** with the desired **Port Range** and **Source** of *Anywhere*.
7. Review
8. Specify which .pem key-pair will be used to connect (via ssh) to the instance.
    a. this file should be moved to your `~/.ssh` directory and its file permissions need to be modified so that only the owner can read it, e.g.: `$ chmod 400 ~/.ssh/mykeypair.pem`
9. Connect to your instance from terminal using ssh. Guide here.

Miscellaneous I

- Access AWS services as an IAM user (see Appendix)
- Export your AWS access keys as environment variables (see Appendix)
  - or, you can use AWS CLI (see Appendix)
- Two types of AWS keys are required: AWS access keys (AWS services) and .pem keys (for `ssh` encrypted connection between laptop and EC2 instance)
- **Never, ever** put your AWS access keys in a file that will be shared publicly (e.g. Github). And if you do…

Miscellaneous II

Your EC2 instance will stop whatever it's doing if the `ssh` connection between your local machine and the server is interrupted, **unless** you use a terminal multiplexer like screen or tmux to detach from your server session before the ssh connection is interrupted. You can later re-attach to your session on the server to get results. Minimal **screen** commands to run *on the server*:

1. `$ screen -S my-session` # creates a session called my-session
2. Start your web app, or model training, or whatever process you want to continue.
3. Typing **Ctrl - a**, then **d** to *detach* from the session.
4. You can now safely interrupt the ssh session; your EC2 instance will keep working.
5. ssh back into your instance.
6. `$ screen -ls` # lists available screen sessions
7. `$ screen -R my-session` # re-attaches to my-session

Boto 3 - Automating AWS with Python

Boto 3 Docs 1.7.62 documentation

TABLE OF CONTENTS

Search...

Docs / Boto 3 Documentation

Boto 3 Documentation

Boto is the Amazon Web Services (AWS) SDK for Python, which allows Python developers to write software that makes use of Amazon services like S3 and EC2. Boto provides an easy to use, object-oriented API as well as low-level direct service access.

Quickstart

Boto3 demo

aws_boto3_notebook.ipynb

Objectives

- Describe what AWS is and where it came from
- Describe how we (the DSI) typically use AWS
  - Workflow
- Demonstrate launching an EC2 instance
- Demonstrate making and accessing S3 buckets using Python (Boto3)
- In the assignment
  - Make an S3 bucket, and upload and download data from bucket using Boto3
  - Run a script locally, then launch an EC2 instance, upload the script, run it in the cloud, and bring down the results locally

Appendix

Useful links and guides for:

- Making an IAM user with the AWS Management Console
  - Make sure to check **AWS Management Console Access** to get a Password for your IAM user
  - If you forgot to set a password, you can create a password for an IAM user.
- Exporting AWS access keys as environment variables
  - In your `~/.bashrc` (Linux) or `~/.bash_profile` (MacOS)
- Launching an EC2 instance using the AWS Management Console
- Connecting to a EC2 instance from Terminal using ssh
  - Note that the instance type will determine the username when connecting with `ssh`.
- Transfering files back and forth between your laptop and your EC2 instance using scp
  - You can also use git! `git add, commit, push` and `git pull`.

Appendix

Useful links and guides for:

- (Optional) Installing the AWS Command Line Interface (AWS CLI)
  - Follow `pip` instructions, if you don't have `pip3` just use `pip`
  - Then configure your AWS CLI
    - Suggest using region: `us-west-2` and format: `json`
- For Deep Learning later in curriculum, you want an Amazon Deep Learning AMI running on a **p2** or **p3** instance (they have NVIDIA GPUs). See this step-by-step guide. Be sure to activate the environment. You will have to ask for a limit increase for the p2 or p3 instances.