

Using AWS

Miles Erickson

July 28, 2017

Objectives

Morning objectives:

- Identify use cases for cloud computing and AWS
- Describe core AWS services
- Configure your computer to use AWS CLI
- Use SSH key to access Amazon EC2 instances
- Create and access an Amazon S3 bucket

Agenda

Morning:

- Introduction to AWS
- Install and configure AWS CLI
- Configure and use SSH
- Access EC2
- Access S3
- Advanced topics

Afternoon:

- High-performance python

References

- [AWS in Plain English](#)
- [Open Guide to AWS](#)
- [AWS Documentation](#)
- [AWS Support](#)

Amazon Web Services (AWS)

AWS provides on-demand use of computing resources in the cloud

- No need to build data centers
- Easy to create a new business
- Only pay for what you use
- Handle spikes in computational demand
- Secure, reliable, flexible, scalable, cost-effective

AWS skills are much in demand

AWS core services:

- Elastic compute cloud (EC2): computers for diverse problems
- Elastic block store (EBS): virtual hard disks to use with EC2
- Simple storage solution (S3): long-term bulk storage
- DynamoDB: NoSQL database
- And much, much more

Elastic compute cloud (EC2)

Spin up EC2 instances for on-demand computing power:

- Instance: a type of hardware you can rent, e.g., 'm3.xlarge' or 't2.micro'
- Amazon Machine Image (AMI), an OS you can run on an instance
- Region: a geographic region, e.g., Oregon a.k.a. 'us-west-2'
- Availability Zone (AZ): a specific subset of a region, often a specific building, such as 'us-west-2a'

Elastic block store (EBS)

EBS provides disk-drive style storage:

- Create a virtual hard disk
- Then mount virtual hard disk on EC2 instances
- SSD or magnetic
- Can store data even when you aren't running any EC2 instances
- Built-in redundancy
- Lower latency than S3 but more expensive

Simple storage solution (S3)

S3 provides cheap, bulk storage:

- Create a bucket which serves as a container for files and directories
- Specify permissions using an access control list (ACL)
- Access via URL or AWS CLI or suitable API
- Higher latency than EBS but less expensive

Use the AWS command-line interface (CLI):

- To debug your configuration
- To manage AWS instances in EC2
- To access S3

Install AWS CLI

- The 'awscli' package installs the 'aws' command

```
$ pip install --upgrade awscli
```

Obtain your AWS credentials

- 1 Login to AWS at <https://console.aws.amazon.com/>
- 2 Click on your name in the upper right menu bar
- 3 Select **Security Credentials** and go to **IAM Users**
- 4 Click **Users** and select your name (or add it)
- 5 Click User Actions > Manage Access Keys
- 6 Create Access Key, Show User Security Credentials
- 7 Copy key into...

Make sure you choose Oregon (us-west-2) as your region.

Configure AWS CLI (1/2)

Easiest way: run `aws configure`:

- Creates default profile in `~/.aws/config`
- Stores credentials in `~/.aws/credentials`
- Can create multiple profiles:

```
$ aws configure --profile fancy_profile
```

- Can also set credential on CLI or via environment variables

Configure AWS CLI (2/2)

Create AWS configuration info in ~/.aws:

```
$ aws configure
```

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
```

```
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEX
```

```
Default region name [None]: us-west-2
```

```
Default output format [None]: json
```

See Amazon's [doc](#) for details

Verify configuration

Check S3:

```
$ aws s3 ls
2016-12-04 10:36:03 spark.rocks
2016-12-04 12:13:38 spark.supplies
$ aws s3 ls s3://spark.supplies/
      PRE data/
      PRE scripts/
```

Check EC2:

```
$ aws ec2 describe-instances --output table
$ aws ec2 describe-instances --output json
```

Help with AWS CLI

See the built-in help for more details:

```
$ aws help
```

```
$ aws ec2 help
```

. . . or Google

Configure and use SSH

To use AWS:

- Setup Key Pairs to access EC2
- Configure SSH on your laptop
- Can use SSH to access any remote machine running an SSH server

SSH

The secure shell protocol allows you to:

- Login to remote machines, such as EC2 using ssh
- Transfer files between remote machines using scp and sftp
- Execute commands on remote machines using ssh

Do not use telnet, rlogin, or FTP, which are older, insecure protocols!

Public-key encryption

Pairs of matched keys that can be used to encrypt or decrypt documents:

- Encrypt with public key \rightarrow decrypt with private key
- Encrypt with private key \rightarrow decrypt with public key

But:

- Trivial to calculate public key from private
- Very, very, very hard to calculate private key from public

Public-key encryption

SSH uses public-key encryption to protect access and encrypt communication:

- Generate a public & private key pair
- Keep private key safe
- Create a key pair for each resource you want to access (AWS, GitHub, etc.) \Rightarrow can revoke individual keys in case of a security breach

See AWS [doc](#) for details

Setup Key Pairs

Create and configure key pair to access EC2:

- Create and import key pair as described in *AWS* documentation
- Set permission on private key to 400:

```
$ chmod 400 ~/.ssh/aws-master.pem
```

```
$ chmod 444 ~/.ssh/aws-master.pub
```

- Can also generate key pair with `ssh-keygen`

Configuring SSH

Modify SSH config to:

- Create alias for long-running instance
- Forward X11 or security information
- Specify which key to use
- And, much much more...
- `man ssh_config` for details

Example ~/.ssh/config

```
Host github.com
  HostName github.com
  User git
  IdentityFile ~/.ssh/git-hub-id_rsa
```

Example ~/.ssh/config

Setup an alias:

```
Host master
  HostName ec2-54-186-136-57.us-west-2.compute.amazonaws.com
  User ubuntu
  ForwardAgent yes
  TCPKeepAlive yes
  IdentityFile ~/.ssh/aws-master.pem
```

Now, `ssh master` will connect to your EC2 instance

Accessing an EC2 instance with ssh

Connect to your machine via ssh:

- 1 Launch an EC2 instance from console
- 2 Use ssh from command line to connect
- 3 Do this after setting up ~/.ssh/config

```
$ ssh master
```

Example: ssh to EC2

[Demonstration]

Example: ssh to EC2

[Demonstration]

Transferring files with scp

To copy files between machines, use scp:

- Works just like regular copy
- Good for simple operations
- Configure ~/.ssh/config

```
$ scp toy_data.txt remote_host:data/  
toy_data.txt 100% 136 0.1KB/s 00:00
```

Managing a session with `tmux`

Use `tmux` to persist jobs across multiple sessions:

- Normally, on logout, all child processes terminate
- Use `tmux` to create a session
- Disconnect
- Reconnect with `tmux attach`
- Install `tmux` via `brew` or Linux package manager

A few tips to make EC2 easier to deal with:

- Use Name tags so that you can find instances easily
- Choose the appropriate hardware type for your problem
- If in doubt, use Ubuntu 16.04 LTS
- Use `tmux` when you login if launching long-running jobs
- Be paranoid: sometimes Amazon will reboot or reclaim your instances
- Put data you need to persist in S3, EBS, or a database
- Never put AWS keys in your code because someone will steal them

Launching an EC2 instance

- Class exercise: Launch a t2.micro instance running Ubuntu

To master the basics, see this [tutorial](#) or `lecture_notes.ipynb` in the repo

- Can find URL to access a file from S3 console
- Set properties (access) via S3 console
- Make sure names conform to S3 conventions:
- lowercase bucket names of at least four characters
- no leading or terminal '.'

Boto3 config

To access S3 via Python, use the boto3 package

- boto (without the 3) is old, don't use it
- Make sure boto3 is up to date:

```
$ pip install --upgrade boto3
```

- Uses credentials in `~/.aws/credentials` which you setup earlier
- Can also read directly from Pandas if you specify S3 URL

Advanced: accessing Jupyter notebook

- Use an ssh tunnel to use a notebook on a remote instance
- On remote host:

```
$ jupyter notebook --no-browser --port=18888
```

- First set up the remote machine in ~/.ssh/config
- Then, on local machine:

```
$ ssh -NfL 18888:localhost:18888 remote_host_alias
```

- Access notebook via browser at URL localhost:18888