Using AWS

Benjamin S. Skrainka

July 11, 2016

Objectives

- Describe core AWS services & concepts
- Configure your laptop to use AWS
- Use SSH key to access EC2 instances
- Launch & access EC2
- Access S3

Agenda

1. Introduction to AWS
2. Install and configure AWS CLI
3. Configure & use SSH
4. Access EC2
5. Access S3
6. Advanced topics

References

AWS documentation is weak and poorly organized, so:

$\Rightarrow$ Google it...

Introduction to AWS

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

Core services

AWS's core services:

- Elastic compute cloud (EC2): computers for diverse problems
- Elastic block store (EBS): virtual hard disks for use with EC2
- Simple storage solution (S3): long-term bulk storage
- Dynamo DB: a variety of databases
- 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’
- AMI: *Amazon Machine Image*, an OS you can run on an instance
- Region: a geographic region such as Oregon aka ‘us-west-2’
- Availability Zone (AZ): a specify subset of a region, often a data center, 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

DynamoDB

DynamoDB provides databases in the cloud:

- Support for most common flavors of SQL (Oracle, MySQL, etc.)
- Once setup, works like normal SQL database
- AWS supports other database types as well

Install & configure CLI

AWS CLI

Use the AWS command line interface (CLI):

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

Install AWS CLI (OS/X or Linux)

On Linux, run:

```
$ 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
```

See Amazon's doc

Install AWS CLI (OS/X)

Can also install via `brew` (or `pip`):

```
$ brew install awscli
```

But, may not be the latest version...

Obtain your AWS credentials

Find your AWS credentials:

1. Login to AWS
2. Click *Your Account* in the upper right menu bar
3. Select *Security Credentials*
4. Select *Users*
5. Then select your username and click *User Actions* > *Manage Access Keys*
6. Create your credentials

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

AWS credentials

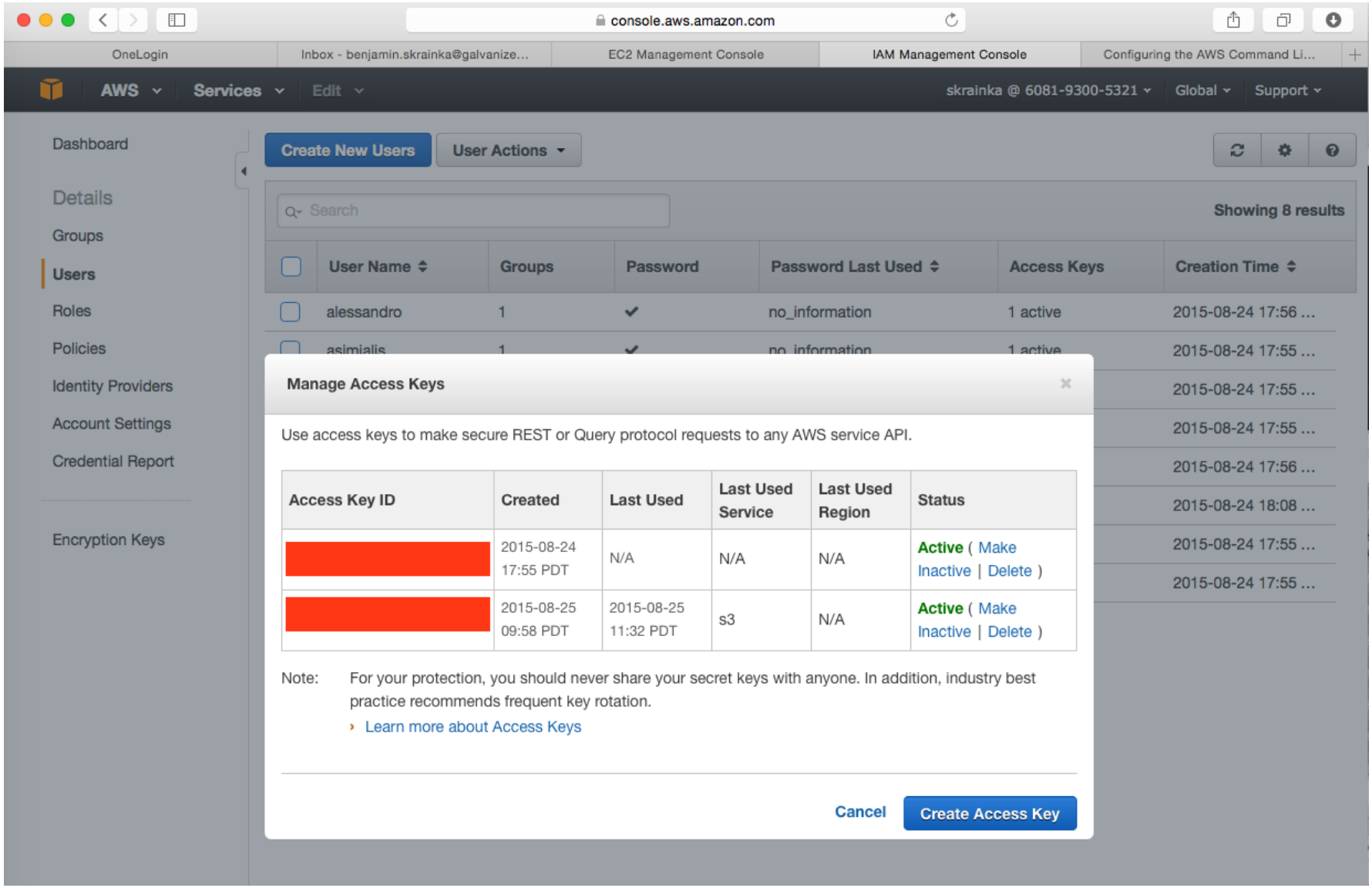

Figure 1: AWS credentials

Configure AWS CLI (1/3)

Easiest to 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 (2/3)

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

Configure AWS CLI (3/3)

Some tools get AWS credentials via environment variables. Set the following in `~/.bash_profile` or equivalent:

```
export AWS_ACCESS_KEY_ID='your access key'
export AWS_SECRET_ACCESS_KEY='your secret key'
```

- Must set `AWS_*` environment variables to use some big data tools!
- Problematic if using AWS in multiple data centers...

Verify configuration

Check S3:

```
$ aws s3 ls
2015-08-25 10:42:43 dsci
2015-08-25 11:30:33 seattle-dsi
$ aws s3 ls s3://seattle-dsi
                           PRE cohort1/
                           PRE skrainka/
```

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

Introduction

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

SSH uses *public-key encryption* to protect access:

- Generate a *public* & *private* key
- Known as a *key pair*
- Need both public and private key to decrypt
- 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 600 ~/.ssh/bss-aws-master.pem
$ chmod 644 ~/.ssh/bss-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
    ForwardAgent yes
    IdentityFile /Users/bss/.ssh/git-hub-id_rsa
```

Example ~/.ssh/config

If you master will run for a long time, setup an alias:

```
Host master
    HostName ec2-54-186-136-57.us-west-2.compute.amazonaws.com
    User ubuntu
    ForwardAgent yes
    ForwardX11Trusted yes
    TCPKeepAlive yes
    IdentityFile /Users/bss/.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 to the instances *public DNS* (Shown in EC2 Dashboard):

```
$ ssh -X -i ~/.ssh/aws-master.pem \
    ubuntu@ec2-54-186-136-57.us-west-2.compute.amazonaws.com
```

or

```
$ ssh -X -i ~/.ssh/aws-master.pem \
    ec2user@ec2-54-186-136-57.us-west-2.compute.amazonaws.com
```

Example: ssh to EC2

```
$ ssh -i ~/.ssh/bss-aws-master.pem ubuntu@ec2-54-186-136-57.us
The authenticity of host 'ec2-54-186-136-57.us-west-2.compute.
RSA key fingerprint is b9:05:ff:da:34:7d:82:20:15:d1:c3:80:10:
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-186-136-57.us-west-2.comput
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-48-generic x86

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Aug 25 21:58:16 UTC 2015

  System load: 0.0                Memory usage: 5%   Processes:
  Usage of /:  9.8% of 7.74GB   Swap usage:   0%   Users logge

...
```

Transferring files with scp

To copy files between machines, use scp:

- Works just like regular copy
- Good for simple operations
- . . . if you specify remote user and machine (IP, DNS) correctly
- Reference remote location as *user@host:path*

```
$ scp -i ~/.ssh/bss-aws-master.pem ./toy_data.txt \
    ubuntu@54.186.136.57:/home/ubuntu/data
toy_data.txt                  100%  136     0.1KB/s   00:00
```

Transferring files with `sftp`

To copy files interactively, use `sftp`:

- Interactive shell for transferring files
- Use to transfer many files
- Use when you don't know the location of a file

```
$ sftp -i ~/.ssh/bss-aws-master.pem ubuntu@54.186.136.57
Connected to 54.186.136.57.
sftp> help
```

Managing a session with `tmux`

Use `tmux` to persist jobs across multiple sessions:

- On logout, all child processes terminate
- Use `tmux` to safely disconnect from a session
- Reconnect on next login
- Install `tmux` via brew or Linux package manager
- See `tmux` exercise

EC2

Launching an EC2 instance

To launch an EC2 instance, follow this tutorial

. . . or the official tutorial

EC2 pro-tips

A few tips to make EC2 easier to deal with:

- Always create instances with tags so that you can find them easily
- Choose the appropriate hardware type for your problem
- If in doubt, use Ubuntu because it is a friendly flavor of Linux
- Use `tmux` when you login in case you need to disconnect or your connection dies
- Be paranoid: sometimes Amazon will reboot or reclaim your instances
- Put data you need to persist in EBS or a database
- Never put AWS keys in GitHub because someone will steal them

S3

S3

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 ‘.’

Where to put your files

For Seattle DSI students:

- Use the bucket `seattle-dsi`
- Create a directory with your surname under the `cohort1` sub-directory
- Put your lab files under `seattle-dsi/cohort1/`*your_surname*

Boto config

To access S3 via Python, use the `boto` package

- Should be installed if you followed setup instructions
- Make sure boto is up to date:

```
$ conda update boto
```

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

Advanced issues

Advanced: accessing ipython notebook

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

    ```
    $ ipython notebook --no-browser --port=8889
    ```

  - On local machine:

    ```
    $ ssh -N -f -L localhost:8888:localhost:8889 \
        remote_user@remote_host
    ```

  - Access notebook via browser at URL localhost:8888
- Run a notebook server:
  - Official documentation
  - Blog on ipython notebook server