AWS - Let’s Take a Tour

Gameplan

1. Get an AWS account.
2. Get the aws-cli.
3. Log on to our *cloud*.
4. Make your own ec2.
5. Check out our sample production database.
6. Make your own database.
7. Figure out what IAM is.

AWS Dictionary

> *I’m gonna jot down some terms and definitions here before we get started so you understand what basic words acronyms mean. I’ll definitely miss some terms so we’re gonna keep this dictionary a work in progress.*

- RDS - relational database service…y’all should know this by now
- EC2 - elastic compute cloud…you know how we *ssh* onto our unix servers at Cal Poly? This is basically the same thing! Spinning up an EC2 means you’re making a server on the *cloud*. To access your EC2 server, you’re gonna have to ssh, and I’ll show you how.
- Console - the screen you arrive at when you log on to AWS
- Services - the suite of things Amazon Web *Services* offers

including RDS, EC2, Lambda, S3…they have a lot

- Security group - a virtual firewall…think of it like a gatekeeper. Your job when you hire a gatekeeper is to tell them who can enter and who can't. A security group defines who's allowed to access your stuff and who can't
- Region - us-west-2, US East(Ohio)…things that look like this on AWS are regions. AWS basically has these datacenters dispersed among different regions. You don't really have to worry about it but I thought I'd mention it because you'll see it a lot (it's probably at the top right of your screen right now).
- AMI - Amazon Machine Image. When you make an ec2 instance, you'll be prompted to select an AMI. An AMI is basically the OS that your cloud server runs on. You can select between various flavors of Ubuntu, Linux, and Windows.
- Instance type - this is a set of configurations for a given ec2 instance. The instance type defines things like CPU, memory, and other basic specs for your ec2.

Go Get Your Account

> *You can get your account here. It's gonna involve a phone call and that may throw you off but don't worry just do the steps. Yes, give them your card info, there's a free tier for everything so it'll be fine. Don't think you'll face any roadblocks here…*

Get the AWS-CLI (Command Line Interface)

> *Just like we're developing a CLI for our project, AWS has one so*

> *we can interact with their resources without having to open up their console. You probably won't use it much as it'll be more intuitive for you to do it through their console. Anyways, run the following command on your machine to get the aws-cli.*

```
$ brew update
$ brew install awscli
$ aws --version
```

If the output of your last command is

```
aws-cli/1.11.180 Python/3.6.3 Darwin/15.6.0 botocore/1.7.
38
```

Good stuff! Moving on...

Log on to Our Cloud

> *If you read the dictionary, you'd know what an ec2 instance is. Now you're about to jump onto the ec2 I created for our project. Of course, my ec2 instance doesn't just let anyone log on. You need a key. Go to our slack channel and download the ec2rdskey.pem file on the #random channel. Leave it in the* `~/Downloads` *folder. Now follow this command.*

```
$ ssh -i ~/Downloads/ec2rdskey.pem ec2-user@ec2-13-58-9-1
82.us-east-2.compute.amazonaws.com
```

…and that's it. You should see a new prompt as follows. Run a `pwd` command.

```
[ec2-user@ip-172-31-35-19 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-35-19 ~]$ exit
```

This is basically a remote server just like our unix servers at Cal Poly. I've done some work on this ec2 in the `home/ec2-user/SQLScripts` folder. Don't worry about this stuff for now.

Make-an-ec2-Workshop

> *Sign into the AWS console. You will now make your own cloud.*

1. In the big search bar under AWS services, search for and navigate to `EC2`.
2. Locate and click on the blue button that says `Launch Instance`.
3. Select the 64-bit RHEL (Red Had Enterprise Linux) AMI. (AMI is defined in the dictionary)
4. Then you'll be asked to pick an instance type. (Instance type is defined in the dictionary) Choose the one with the green `free-tier elgible` banner.
5. Click the `Review and Launch` blue button on the bottom right of the screen. There are several other things you can configure like additional storage and security groups, but I'm keeping

this tutorial basic.

6. You'll be directed to a summary page of your ec2 instance. Time to click the blue button that says `Launch`!
7. Now in the popup dialog, select `Create a new key pair`. For the `Key Pair Name` write "*myec2key*".
8. Click the gray button that says `Download Key Pair`. This key pair that you just made is a .pem (stands for privacy enhanced mail…I don't even know what that is) file, but think of it as a key that opens your ec2. You don't want just anyone to access your ec2, so this is a form of security. What you downloaded earlier from the slack group was the .pem file I created to access my ec2 instance. I hope everything starting to make sense now.
9. Now *finally* click the blue button to `Launch Instance`.
10. Launching your instance will take a few minutes. On the bottom right, click the blue button `View Instances`. This shows you a list of all the ec2 instances you have running. You should now have 1 row in the table. Wait for the `Instance State` column to be *running* and the `Status Checks` column to be *2/2 checks passed*. Then move to the next step.
11. Below the table of ec2 instances, you'll see the `Public DNS` or the `Public DNS (IPv4)`. If you can't find it just run a cmd+f to find it. Store that DNS somwhere, you'll need it. This public DNS is where your ec2 instance can be accessed from the Internet. If you hosted a website on your ec2 instance, you can access it via this address.
12. Now hop onto your terminal locally. Follow these commands. Instead of [your public DNS] plug in the DNS from step 11.

```
$ chmod 400 ~/Downloads/myec2key.pem
$ ssh -i ~/Downloads/myec2key.pem ec2-user@[your pu
blic DNS]
Something blah blah...Are you sure you want to cont
inue connecting (yes/no)? yes
```

You've probably seen the output [something blah blah] before when trying to ssh into Cal Poly unix servers. Don't trip, it's just adding that address to the remote hosts to your *known hosts*.

STILL UNDER CONSTRUCTION AND NOT COMPLETED