CS 346 (Spring 19): Cloud Computing

Project #1

AWS Hello World

due at 5pm, Thu 24 Jan 2019

1 Overview

For this assignment, you will create a simple website, hosted by AWS EC2. The main page will contain a Hello World message.

> **This spec is long. I'm sorry about that. But please read all of it! I have worked hard to answer many, many questions! You will not have to ask nearly so many questions on Piazza, if you read the instructions carefully.**

2 Steps

You will be setting up an EC2 instance, configuring it to be a web server, and creating a simple page. If you have prior experience - or if you paid attention during the demo in lecture - then a lot of this may be quite easy, but I wanted to give as many details as I could.

2.1 Login to AWS Educate

For most of your, your first step should be to log in to your AWS Educate account (through Vocareum). Once you log in there, you should see a link or button that takes you to AWS. When you click on it, Vocareum will connect you to AWS.

(If you have your own AWS account - and didn't create a Starter Account - then you should be able to go directly to the AWS console. Connect to it by logging in at `https://aws.amazon.com`.)

2.2 Login to the EC2 Console

Go to the EC2 dashboard. You can do this with the "All services" flyout - or, you can click on the "Launch a virtual machine...with EC2" wizard. You can also get here directly with the following link: `https://console.aws.amazon.com/ec2/`

2.3 Region

AWS organizes its services by Region; I suspect that this actually just decides what datacenter will host your system. In general, different regions should have the same services, and of course you can connect to any region from anyplace on the Internet.

Students using AWS Educate Starter Account

I had my TAs run this project, and they said that if they switched the Region away from the default (it was `us-east-1b` for them), they were not able to use the AWS services. So, I recommend, if you are using a Starter Account, to simply use whatever region you start with.

Students using an ordinary AWS account

If you have an ordinary AWS account, I recommend that you switch your Region (in the top left of the window) to Oregon. Most likely, this won't matter much, if at all - but it was recommended by the previous instructor, and I'm following his lead.

2.4 Key Pair

If you haven't already, create an SSH key pair `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#having-ec2-create-your-key-pair`

Things to remember:

- The "Security & Groups" section of the console can be found on the left side. There's a long list of pages you can traverse to.
- The name of the key pair doesn't matter; you probably will only have one, anyway.
- Make sure to save the private key, which Amazon generates for you, on your computer. Without it, it will be impossible to use this key pair!

2.5 Security Group

Amazon gives you a default Security Group to start with, but you will need to update the rules to allow more traffic. Go to "Security Groups" under "Network & Security;" select your default security group on top, and then look at the Inbound rules tab on the bottom of the page.

Security groups are connected to Virtual Private Clouds (VPCs); basically, VPCs are virtual networks that provide isolation for your instance(s), and Security Groups give you a handy way for configuring the firewall settings for them.

When you start, your Security Group should allow all inbound packets, provided that it comes from within the Group. That is, there will be a rule which says that all protocols, and all ports, from the Security Group (something like `sg-XXXXXX`) will be accepted. This basically just lets the instances of your VPC talk to each other without any limitation.

At the bottom of the console, select the Inbound tab, then click on Edit, and then Add Rule to create three to five new rules, detailed below. (Look for the column headers across the top of the table.)

2.5.1

Type: HTTP
Source: Anywhere.

This will automatically set the Protocol and Port Range fields (TCP/80), as well as the IP addresses of the Source fields (0.0.0.0/0, ::/0)[1].

If you don't set this, then your webserver will not be accessible to any other computers; the firewall will block the incoming connection. By adding this rule, you are saying that **any** computer, anywhere in the world, can access the webserver on the instance that you create.

2.5.2

Type: HTTPS
Source: Anywhere.

This is for secure (that is, encrypted) HTTP. It's not strictly necessary to do this, but in the future (beyond this class), you will normally configure any website to support this.

2.5.3

Type: SSH
Source: 150.135.165.0/24[2]

This makes it possible to ssh into the instance from the UA campus. However, it does **not** give access to any other machines; if some other machine, from anywhere on the Internet, tries to SSH into your instance, it will simply fail to connect (just as if the instance didn't even exist).

Why is this a good idea? Doesn't SSH have its own own security protocols (such as the Key Pair that we recently created)? Wouldn't it be just fine to simply leave the SSH port open to all machines, just like with HTTP?

In theory, yes. But in practice, this is a **terrible idea,** because it means that the entire world can attempt to break into your server through ssh. If there ever was a security breach that exposed your private key (or, if hackers found a vulerability in the ssh server itself), this would put your server at risk, since a user with ssh access can do **anything.**

Thus, the best practice is to lock this port down. By using the range above, most of the computers on the Internet will be blocked; however, any connections coming from the UofA network will be allowed.[3]

---

[1] In Internet-speak, 0.0.0.0/0 is the way to encode "any IPv4 address." ::/0 is the same thing, but for IPv6.

[2] `https://www.digitalocean.com/community/tutorials/understanding-ip-addresses-subnets-and-cidr-notation-for-net`
`cidr-notation`

[3] UofA uses IP Masquerading, which means that the IP address of your computer, when on a UofA Wifi network, is not your real (external) IP address (that is, the one that Amazon will

2.5.4 (optional)

Type: SSH
Source: xxx.xxx.xxx.xxx/32

See the previous section. The rule that you added for the UA network works for any computer connected through WiFi to the UA network, as well as the department computers in the lab, because they are all behind the same firewall[4]. However, computers that are outside the firewall (such as Lectura) have their own IP addresses.

If you don't plan to SSH from lectura to your instance, then this rule isn't required. But if you do, then perform the following steps:

- Look up the IP address of Lectura (or, any public server you plan to use). Depending on your OS, you may use `host` and/or `nslookup`:

```
host lectura.cs.arizona.edu
nslookup lectura.cs.arizona.edu
```

- Add a rule for that IP address, with `/32` (which means "just this one IP addr"), to your rule as the Source. (At the time I wrote this spec, the proper address was `192.12.69.186/32`, but double-check, just in case Lectura moves to a new IP address someday.)

2.5.5 (optional)

Type: SSH
Source: My IP (when you're outside UofA)

See the previous section. The rule that you added for the UA network won't help you if you are trying to connect from home. For that, you should add a rule for the specific network you're on.

I recommend that you use the the "My IP" option. By using the "My IP" option, AWS will auto-detect your IP address (that is, the IP address that you are using to connect to the console), and will only allow ssh connections from the same address. (If you're curious, you can find this yourself as well: `http://checkip.amazonaws.com`, `http://whatismyipaddress.com`)

This will only allow a connection from a single IP address - which, if you're at home, means that anybody in your house can attempt to connect (but anyone outside your house will be blocked).

---

see). Instead, all connections, from (nearly) all computers will be funneled through a small number of external IP addresses. The rule that I've given you will allow connections from all of those addresses. `https://en.wikipedia.org/wiki/Network_address_translation`

Of course, this means that (nearly) any computer on the UofA network can attempt to connect to your instance - but they will still fail, since they don't have your private key. So your instance should still be secure.

[4] see the previous footnote, about IP Masquerading

NOTE: You will have to update this rule every time that you move to a new location. This is annoying, but it's a good habit for security. Happily, this doesn't require you to reboot your instance - when you update the Security Group, you will be able to connect to your instance almost immediately.

2.6 Launch an Instance

Start an EC2 instance, details below.

> While you can start and stop your instances as often as you like, I've read that Amazon always rounds the time up to 1 hour increments, even if your instance only runs for a few minutes. So as a general rule, you should only start an instance when you need it, and only stop it when you know you're done. It's more efficient than doing many start/stop operations.

2.6.1

Look for the "Instances" page on the EC2 dashboard. Click on "Launch Instance." (Again, look for the page listing on the left edge of the dashboard.)

Your first step is to choose the image that you will run; this will determine your operating system, and the starting tools you have available. (To prevent yourself from doing something silly, I suggest that you click the "Free tier only" checkbox.)

Select the "Amazon Linux 2" image. (If you are curious about the difference between "Amazon Linux" and "Amazon Linux 2", read this: `https://aws.amazon.com/amazon-linux-2/faqs/` and this: `https://cloudonaut.io/migrating-to-amazon-linux-2/`.)

Select the x86 architecture (although it shouldn't matter much), and click the "Select" button to select the image.

2.6.2

Next, you will choose what type of instance you want; this basically determines what type of (virtual) hardware your image will run on. Select the `t2.micro` instance (since it's eligible for the Free Tier).

2.6.3

Next, click the button to go the next page: "Next: Configure Instance Details."

Here, you control how many instances you will create, along with many other features. The defaults should work fine, but glance through it; there are lots of interesting options, which would be useful in a more complex system. (You can also mouse over the various "i" icons to get some quick help. A quick Google search will often get you a lot more information.)

Notice that the instance was assigned to your default VPC (Virtual Private Cloud - I mentioned that back in the Security Groups section of the spec).

2.6.4

Click Next to get to the Storage page; you'll see that the image comes with a small hard drive. You can use this exactly like a hard drive on any other computer, with one critical exception: if you Terminate the instance, the image will be destroyed forever! (If you choose to only Stop an instance, you don't get charged for running your instance, but the data you've left laying around will build up charges.) `https://docs.rightscale.com/faq/clouds/aws/Whats_the_difference_between_Terminating_and_Stopping_an_EC2_Instance.html`.

Your Free Tier account gives you a small amount of storage, free, for the first 12 months. So it's OK to just Stop your instance, instead of Terminating it. But be careful about two things:

- Don't create so many instances that you overrun your free allocation.
- Remember that the free storage ends after 12 months.

You can find information about the Free Tier here: `https://aws.amazon.com/free/` Pay attention to what is free forever, and what is only free for 12 months. Notice that your instances are using EBS (Elastic Block Storage).

2.6.5

Click Next to go to the Tags page; we won't need any tag, so click Next again to configure the Security Group.

2.6.6

On the Security Group page, you'll see an interface that looks a lot like the Edit Security Group page we saw before. However, your Security Group is not selected; it defaults to a very restrictive firewall. Use the "Select an existing security group" option, and select the security group that you configured previously.

2.6.7

Click Review and Launch. Confirm your settings, and click Launch. It will ask you to choose the Key Pair; choose the one that you created earlier. It will bring you to the "Launch Status" page; take a minute to look at the Billing Alerts feature.

2.6.8

Finally, click View Instances to return to the "Instances" page on the main EC2 page. Your instance should be in the "pending" state; when it enters the "running" state, you will be able to connect to it.

2.7 Connect to the Instance

Connect to your instance by selecting it and hitting the "Connect" button.

Previously, it was possible to use a Java applet (provided through the EC2 console) to ssh to your instance; however, most modern browsers have deprecated Java applets, and it no longer works. So instead, select the option to ssh (from your command line) to your instance; EC2 will give you some helpful text about how to log in.

I recommend using the command line (which works just fine on Mac, Linux, the Linux Subsystem for Windows 10, and Cygwin), but if you just refuse to run use it, Windows users can also install PuTTY.

`https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html`

(Another option is to first `ssh` to Lectura, and then `ssh` from Lectura to your instance. If you aren't running Linux/Mac OS/Cygwin, this is a nice option.

NOTE: The first time that you connect to an instance, ssh may say that "The authenticity of the host cannot be established," and ask you if you want to continue. This is normal; it's ok to just say Yes[5].

2.8 Update Software

You will might see a message that says that you need to run

```
sudo yum update
```

to update the software on the machine. If it asks you to, then please do so. It should complete in just a few moments[6].

FYI: `yum` is a tool that provides package install and update services. If you have used `apt-get` before, this is basically the same thing. Some Linux distros (those based off Debian, mostly) use `apt`, and others (those based off RedHat/Fedora) use `yum`. They both serve basically the same purpose.

2.9 Install `httpd`

`httpd` is the HTTP Daemon, which in UNIX-speak, means that server for HTTP. On our instances (as is almost always the case on Linux), the server is Apache.

Install it with the command

```
sudo yum install httpd
```

[5] To be entirely secure, you shouldn't do this, since it is **possible** that an attacker could perform a Man-in-the-middle attack on your first connection. But it's highly unlikely. And getting the "correct" key, in a truly secure way, is a real pain!

`https://en.wikipedia.org/wiki/Man-in-the-middle_attack`

[6] Why isn't it always up-to-date when you start???

My guess: these images are made with up-to-date software when they are created, but as new versions of various packages come out all the time. I would assume that Amazon updates the image fairly regularly, but it would be normal for there to often be a **few** packages that have newer versions when you create a new instance.

2.10 Start up the Webserver

Now that the webserver is installed, start it up with

```
sudo systemctl start httpd
```

While it's not required (since I don't figure you'll be rebooting this instance much), there's another handy command you should know. The above command starts the httpd service **just this once.** To make it start every time that the machine reboots, also do:

```
sudo systemctl enable httpd
```

2.11 Create a Test Page

Create a trivial webpage; write it to the file `/var/www/html/index.html` . Your file must include a "Hello World" of some sort, and also mention your name. Be creative! :)

I prefer to use an editor to write files. Your instance includes `vi` and `nano` by default[7] To edit the file, do

```
sudo vi /var/www/html/index.html
```

However, if you don't want to use an editor, it is possible to do it entirely from the shell:

```
sudo sh -c 'echo "Hello World, from X" > /var/www/html/index.html'
```

(Obviously this doesn't work well for large files!)

Classic editors:

- `vi`

  Classic, simple, efficient. Nasty interface. My favorite.

  Google for a tutorial if you want to use this.

- `emacs`

  Just as confusing as `vi`, but some programmers prefer it. Massively customizable. `https://xkcd.com/378/`

- `nano`

  Good for beginners.

[7] If you prefer to use `emacs`, you can install it with
`sudo yum install emacs`

2.12 Open the Page

Using a web browser, open up the address of your instance. The address you open will be the same as the one that you ssh'd to (minus the username, of course). You should find the file at the root directory, so the complete URL will be something along the lines of

```
http://ec2-xx-xx-xx-xx.us-west-2.compute.amazonaws.com/
```

Take a screenshot of the browser window (making sure to include the URL bar at the top, so that I can confirm you were connecting to an EC2 instance).

2.13 Check the Logs

From the shell on your instance type

```
sudo cat /var/log/httpd/access_log
```

Copy the output from the above into a text file on your local computer.

2.14 Stop Your Instance

Stop your instance via the EC2 console; right click on the instance, select "State," and then "Stop."

> Remember, you have some free EBS storage for 12 months, using your account. So long as you don't have more than a couple of instances, you still stay under the limit.
>
> But remember to Stop instances, since otherwise, your instance will keep chewing up CPU time (even if it's idle).

3 Turning in Your Solution

You must turn in two files to D2L, in the Assignment folder for this project:

- Web browser screenshot, see above.

  Allowed formats: .jpg, .png, .gif

- Text file, containing your `access_log`, see above.

  Allowed formats: .txt **ONLY**

Thanks to Prof. John Hartman for many resources used in this class.