NWEN243 – 2020 T2

Lab 1, Part 3: JOKES R U – a roll your own cloud based server.

Due Friday 18th September 2020

Value: 5% of your final grade.

Overview

So far in Lab 1 (parts 1 and 2) you have learnt how to:

1. Create and configure a VPC
2. Configure and Launch an EC2 instance with a tiny webserver and security rules.

In this lab we're going to extend this – by getting you to build your own tiny application, upload it to your EC2 instance and test it.

For this lab you will need to use the private-public key pairs so that you can SSH and SCP in to your EC2 instance. You will need to use SCP to upload your application files, and SSH to execute the server. You will also need to configure the Security to open the port your sever uses to the outside world.

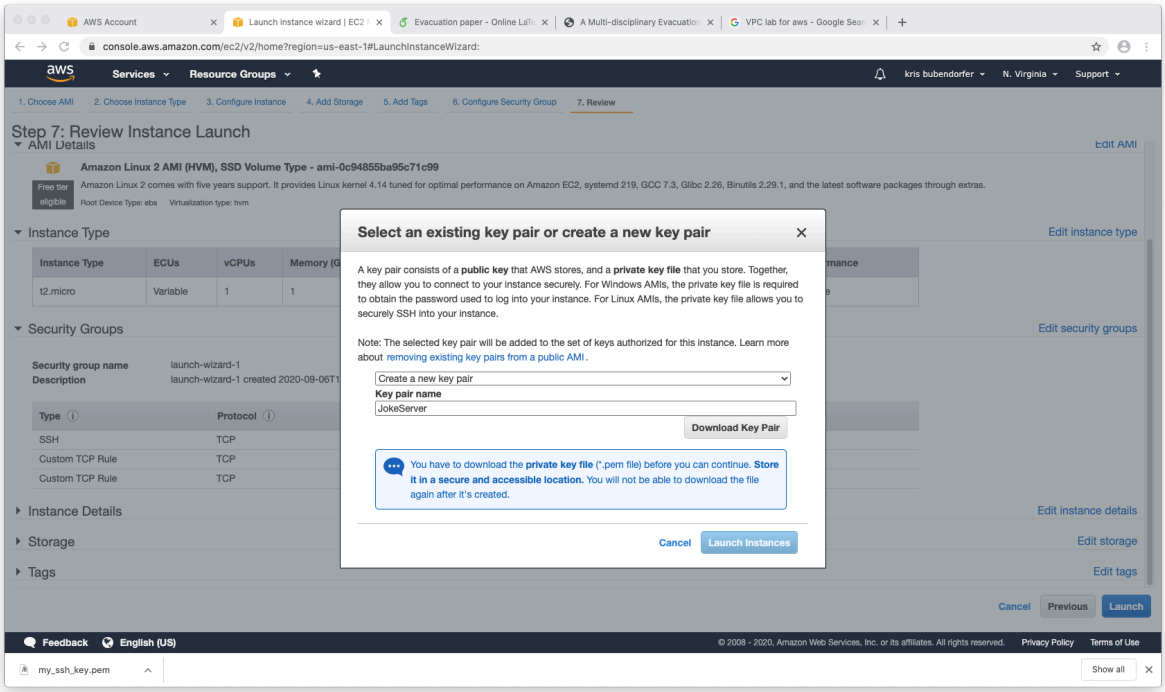
Task, the first (create your new EC2 instance)

Your first objective is to set up your EC2 instance, the same was as in Lab 1 - part 2, except:

1. You should omit the custom bootstrap code – that created the webserver, we won't be using that (step 7).
2. In the Security Group (step 10) – see the default port 22 rule? This setting is for SSH and we will need that. You do not however need an HTTP rule, and we will come back later to add a rule for your joke server.
3. Rather than skipping the key pair (step 14) – you must create and download these to your computer. You will need these so you can access your instance remotely.

Let's do this now, be sure to remind yourself of the steps from the lab 1 part 2.

1. This is step 14, in this popup, select – "Create a new key pair", and name it. In this example I named mine "JokeServer".
2. Next click "Download Key Pair" – and download it to your working directory.

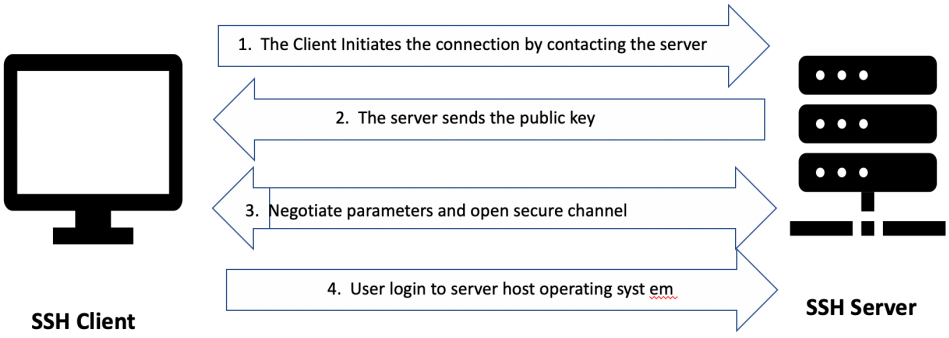


3. Now launch your instance in the usual way.

Once your instance is running, we can connect to it using SSH and your key-pair.

Task, the second (SSH into your EC2 instance)

The most common tool to connect to Linux servers is Secure Shell (**SSH**). When connecting to hosts via SSH, SSH key pairs are used to individually authorize users.



Simplified setup flow of a secure shell connection

Let's do this now with our EC2 instance:

1. Copy the instance IP address from the Instance State page.
2. Open a terminal window on your Lab workstation, on your MacOS machine, or if you are using Windows – you will need to use putty.
3. Key hygiene:

   Fix the permissions on your keypair (linux/Mac (windows I have no idea)):

   %> chmod 400 `JokeServer.pem`

4. Now connect to your EC2 instance with SSH.

   ssh -i <your keys>.pem ec2-user@<your EC2 IP address>, so in my case:

   %> ssh -i JokeServer.pem ec2-user@54.165.8.212

5. You will now get a message like this:

```
The authenticity of host '54.165.8.212 (54.165.8.212)' can't be established.
ECDSA key fingerprint is SHA256:K7+5aCd754jnED8svmBqjQ1n5j7lj3CeNV/iFrlItCw.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

6. Type yes and the HOST will be permanently associated with this key.
7. Success! You should be connected to your EC2 instance.

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-0-198 ~]$
```

Task, the third (Install Java in your EC2 instance)

Java is not installed by default on your instance. So we need to do that now. You can install either java 11 or java 8. Ensure the version is compatible with your local machine.

1. Java 11

```
sudo amazon-linux-extras install java-openjdk11
```

2. Java 8

```
sudo yum install java-1.8.0-openjdk
```

3. Check it all installed ok and the version is right with

```
java –version
```

4. If you find yourself needing to switch version, try:

```
sudo alternatives ––config java
```

5. Please note, this is a runtime not a full developer suite, so you cannot compile your java with javac on your instance. This is actually the right way to do things, you should develop on your local machine, using whatever development environment you like, and then upload the JAR or class files as needed to the EC2 instance.
6. Let us practice uploading files and finally check it all works. We'll upload a small java program (not provided) and execute it, just something simple like HelloWorld.java

   `scp –i <your keys>.pem <your file> ec2-user@54.165.8.212:`

   so in my case:

   `%>scp –i JokeServer.pem HelloThread.class ec2-user@54.165.8.212:`

7. Now do an '%> ls' in your EC2 instance, and you will see your uploaded file.
8. Run your program, it will work if everything is correct.

Task, the 4th (Write your java Joke Server program)

Now we have an environment and EC2 instance set up – you need to write a small client-server pair in java. You should do this on your local machine in your preferred IDE as is good practice. Before you start, look at the Socket and ServerSocket class documentation

- Joke server: creates a TCP server socket (you must choose a port number, I usually use 5000). It will then loop forever around an accept method call, which will pause the server until a connection request is made from a client. Once the request is accepted, pick a joke at random and write it to the client. You do not need to parse a client request message, and you can loop back around the moment you have sent the joke.
    - You will need a ServerSocket *Hint: you will need to use the accept() method*.
    - You would usually write a TCP server using threads as this is the right way to handle TCP server sockets – as shown in lectures. However,
        - Threads add a complication. I recommend writing a non threaded Server first to properly understand the use of Sockets without additional complications. Then, if all is well,
        - Write a threaded version of your Joke server. You should consider the threaded version the "Challenge part" of this lab, just like COMP102/103 this is the A part of the lab. You should look at the Thread class and maybe the Runnable interface.
- Joke Client: a very simple client which contacts the server and then outputs the response joke to the user's screen. This client is single threaded and does not need to loop.

NOTE: write, debug and test the client and server together as a pair on your local machine. I am not asking for a large or complex program here, the task is quite straightforward and the minimal implementation is just fine. You have complete leeway on how to write this client-server pair, and there are countless tutorials online for threads and sockets – however, most people are being clever and many solutions are longer and more complex than they need to be, I'm after something basic. To give you some idea, my minimal server solution has 5 jokes in a constant string array and there are about 4-6 lines that actually do useful things.

VERY IMPORTANT

If you use a tutorial, or someone else's code published online as a basis for your solution, YOU MUST reference it in your comments! It is best practice to correctly make attributions, even if you modify it. You will not get penalised for correct attribution, indeed – you might well get penalised for not attributing, as that is plagiarism.

Task, the 5th (Run your Joke Server program)

1. Once your client and server are debugged, tested and working on your local machine, upload the server class file to your EC2 instance.
2. Execute your server on the EC2 instance.
3. Run your client on the local machine.
4. What happened?

5. You will need to update the security group to open the port you chose for the server port.
6. Once you've done this, your client should work properly with the server.
7. Congratulations – you should now be receiving quality jokes.

Submission:

Don't forget to attribute any code sources in your code comments.

In person:

1. Demo your configuration and run the client and server programs for the lab tutor.
2. Submit your client and server code to the 243 submission system.

Remote:

1. Submit your client and server code, and a text file containing your AWS email.
2. Submit a screen shot, with two terminals open, one showing the execution of the server on the EC2 instance, and the other showing the client window (plus your student ID).
3. You may be asked to complete a demo over zoom.

In both cases: do not terminate your instance – we may need to look at it. Choose stop instead. Be aware – once you restart your instance after stopping, you will be assigned a different IP address. Your uploaded files should still be present.