Assignment 9

New Attempt

- Due Nov 19, 2024 by 11:59pm
- Points 100
- Submitting a file upload
- File Types zip

Due: Tuesday, November 19, 2024 (11:59 PM). Please submit your assignment on Canvas as a ZIP file that contains your report in a PDF format and all the source code you have for this assignment (all the .tf files).

Terraform

In this assignment, you will be introduced to Terraform, a tool for building, changing, and versioning infrastructure safely and efficiently. You are going to create the same infrastructure as in the previous assignment, but this time using Terraform.

Setup

You need to have installed the AWS CLI from **https://aws.amazon.com/cli/ (https://aws.amazon.com/cli/) .** **(https://aws.amazon.com/cli/)** After installing it you need to edit or create the credentials and config files. The files are located at ~/.aws/ on Linux, macOS, or Unix, or at C:\Users\USERNAME\.aws\ on Windows.

Now, go to the IAM console and generate a new pair of keys. You should create the *credentials* file using the following content:

```
[default]
aws_access_key_id = "your key"
aws_secret_access_key= "your key"
```

Next, you should create the *config* file in the same directory using the following contents:

```
[default]
region=us-east-1
```

After successfully setting up the AWS CLI you should install Terraform from **https://www.terraform.io (https://www.terraform.io/)** .

Next, create a folder for your Terraform files and put the files provided **here (https://canvas.sfu.ca/courses/85026/files/24007538?wrap=1)** **(https://canvas.sfu.ca/courses/85026/files/24007538/download?download_frd=1)** in the folder. Navigate to the folder and run the commands:

```
terraform init
terraform apply (use this command to “upload” any changes to your “code”)
```

This will create a simple VPC with a subnet.

Task 1 (11%): Modify the file `subnets.tf` inside the provided .zip file to create a private subnet along with the existing public subnet.

Hint: The command `terraform apply` "uploads" new changes to AWS. Use this command to check if your "code" works properly.

Task 2 (11%): Create a file named `gateways.tf` and place the “code” needed to create an internet gateway that is associated with the VPC terraform created.

Task 3 (11%): Create a file named route-tables.tf and write the “code” needed to create the proper route-table for the internet gateway as well as the subnet association for the public subnet.

For the remaining tasks, the following links may be helpful (**gateway** (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/internet_gateway) , **route table** (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table) , **route table association** (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table_association)).

Task 4 (11%): Create a file called `security-groups.tf` and write the “code” required to create a security group(s) that allows inbound traffic for SSH and for TCP for port 8081 and all outgoing traffic.

HINT: You should probably consider not putting all of the rules inside one resource. It is a better idea to separate the groups so you can assign a subset of the groups to a public EC2 and another subset to a private EC2 without having groups with repeating rules. This means that instead of creating one group with all the rules, you can create one group for each rule.

Task 5 (11%): Create a file called `ami.tf` and place the required “code” in order to create an image based on Ubuntu server 18.04 AMD 64. Your code should create an AMI from Ubuntu's AMI.

HINT: These links may be helpful [**Link,** (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/ami) **Link** (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/ami_copy)]. Also, avoid using the following because it is pretty complicated and not what you need **Link** (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/ami)

Task 6 (11%): Create a file called `ec2.tf` and write the “code” to create an EC2 instance based on the AMI from **Task 5** inside the **public subnet** the provided code creates. The new public instance adopts all the rules that are mentioned in **Task 4**. You should also assign a key to this instance so that in later tasks you can access it via SSH. Also, set the option `associate_public_ip_address = true`.

HINT 1: If you find yourself creating and destroying your setup many times, consider using the Ubuntu's AMI directly instead and comment out the code from **Task 5**. This will save you time for your debugging purposes.

HINT 2: Before applying the changes you should probably create SSH keys for your machine so that you can connect to it later. You can choose to do it in either of the following ways (follow one way not both).

Way 1: In order to create key pairs go to the EC2 Menu > Key Pairs (Under Network & Security) **Link (https://console.aws.amazon.com/ec2/#KeyPairs:)** . Create a new key pair, give it the name CYBERSECURITY_EC2_PUB, and download the private key to your computer. In order to inform AWS that you are going to use this key pair to SSH to the instance add this to your resource:

```
key_name = "CYBERSECURITY_EC2_PUB"
```

More info about this in the following link **Link (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance#key_name)**

Way 2: If you want to create the keys locally in your machine create a key pair with the following command

```
ssh-keygen
....
```

Use the following terraform resource to create the keys in AWS **Link (https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/key_pair)**

Specify the name as "CYBERSECURITY_EC2_PUB" and the public_key field to be the contents of your public key (.pub file)

Associate the key to your instance by setting a key_name field inside your "aws_instance" resource like this:

```
key_name = "CYBERSECURITY_EC2_PUB"
```

Task 7 (11%): In the `ec2.tf` file you created in **Task 6** write the “code” to create an EC2 instance based on the AMI you created in **Task 5**. This new instance should be inside the **private subnet** you created in **Task 1**. It also should have outgoing traffic everywhere and be accessible through SSH. Again you have to specify a key so that you can SSH to that machine later (follow the same steps as in **Task 6's HINT**). Also, set the `associate_public_ip_address = fase.`

HINT: You can use **some** of the security groups you created in **Task 4** for that, just keep in mind not to allow any inbound traffic other than SSH. In the case you created one security group in **Task 4**, you will probably need to create and attach another security group for this EC2 instance.

Task 8 (11%): If you have created the EC2 instances properly you should be able to SSH into the **EC2 private instance** created in **Task 7** through the **EC2 public instance** created in **Task 6**. After SSH into the private instance try to ping **www.google.com (http://www.google.com/)** . It should not work. If you try the same from the **EC2 public instance** you will see that you can actually ping **www.google.com (http://www.google.com/)** . Explain briefly why on the **EC2 private instance** you cannot ping Google whereas on the **EC2 public instance, you can**.

HINT: You can follow the following steps if you don't know how to SSH to the private instance

Way 1 (Not the best practice but it will not affect your grade):

Copy the SSH **private** key you provided for the EC2 private instance in the EC2 public instance's home directory. To do that type the command:

```
scp -o 'IdentitiesOnly yes' -i CYBERSECURITY_EC2_PUB CYBERSECURITY_EC2_PRIV ubuntu@<PUBLIC_IP_OF_EC2_PUBLIC_INSTANCE>:~/
```

- <PUBLIC_IP_OF_EC2_PUBLIC_INSTANCE> is the **public** IP the **EC2 public instance** gets assigned
- CYBERSECURITY_EC2_PUB is the path to the file of the **private key** of the **EC2 public instance**
- CYBERSECURITY_EC2_PRIV is the path to the file of the **private key** of the **EC2 private instance**

Then ssh to the public instance by typing:

```
ssh -o 'IdentitiesOnly yes' -i CYBERSECURITY_EC2_PUB ubuntu@<PUBLIC_IP_OF_EC2_PUBLIC_INSTANCE>
```

- <PUBLIC_IP_OF_EC2_PUBLIC_INSTANCE> is the **public ip** the **EC2 public instance** gets assigned.
- CYBERSECURITY_EC2_PUB is the path to the file of the **private key** of the **EC2 public instance**

Now you should be SSHed into the **EC2 public instance**. Check that the **private key** of the **EC2 private instance** is inside the home folder by typing

```
ls -l  ~
```

Assuming the key is there, you must SSH to the **EC2 private instance**. Type:

```
ssh -o 'IdentitiesOnly yes' -i CYBERSECURITY_EC2_PRIV ubuntu@10.0.2.50
```

- CYBERSECURITY_EC2_PRIV is the path to the file of the **private key** of the **EC2 private instance inside the EC2 public instance's directory** (this should be the home directory)
- 10.0.2.50 is the IP of the **EC2 private instance**

Way 2 (Best practice):

Use the following command to directly ssh to the **EC2 private instance** from your computer by going through the **EC2 public instance**.

```
ssh -o ProxyCommand="ssh -o 'IdentitiesOnly yes' -i CYBERSECURITY_EC2_PUB -W %h:%p ubuntu@<PUBLIC_IP_OF_EC2_PUBLIC_INSTANCE>" -o 'IdentitiesOnly yes' -i CYBERSECURITY_EC2_PRIV ubuntu@10.0.2.50
```

- <PUBLIC_IP_OF_EC2_PUBLIC_INSTANCE> is the **public ip** the **EC2 public instance** gets assigned.
- CYBERSECURITY_EC2_PUB is the file path of the **private key** of the **EC2 public instance**
- CYBERSECURITY_EC2_PRIV is the file path of the **private key** of the **EC2 private instance**

If you used the same key in both instances, just specify the same file for both arguments in the command.

Task 9 (12%): What would you change in your “code” in order to let the **EC2 private instance** connect to the Internet (please consider using the best practices since this is an instance in a private subnet)?