Creating a VPC Networking Environment for the Cafe

Step 1: Creating a public subnet

1. Open the **Amazon VPC console**.
2. Note that a VPC called Lab VPC was created for you.
3. Create a public subnet that meets the following criteria:
    - **Name tag**: Public Subnet
    - **VPC**: *Lab VPC*
    - **Availability Zone**: Choose Availability Zone **a** of your Region (for example, if your Region is *us-east-1*, then select **us-east-1a**)
    - **IPv4 CIDR block**: 10.0.0.0/24
4. Create a new internet gateway and attach it to the Lab VPC.
5. Edit the route table that was created in your VPC. Add the route 0.0.0.0/0. For the target, select the internet gateway that you created in the previous step

Step 2: Creating a bastion host

6. From the **Amazon EC2 console**, create an EC2 instance in the Public Subnet of the *Lab VPC* that meets the following criteria:
    - **Amazon Machine Image (AMI)**: *Amazon Linux 2023 AMI (HVM)*
    - **Instance type**: *t2.micro*
    - **Auto-assign Public IP**: This setting should be disabled
    - **Name**: Bastion Host
    - Security group called Bastion Host SG that only allows the following traffic:
        - **Type**: *SSH*
        - **Port**: 22
        - **Source**: Your IP address
    - Uses the **vockey** key pair

Step 3: Allocating an Elastic IP address for the bastion host

7. Allocate an Elastic IP address, and make it reachable from the internet over IPv4 by associating it with your bastion host.

Step 4: Testing the connection to the bastion host

8. Download the SSH key. The file will be named **labuser.***.
9. Connect to your bastion host by using SSH.
10. After you have tested your connection to the bastion host, you can close the terminal or PuTTY.

Step 5: Creating a private subnet

11. In the console, create a private subnet that meets the following criteria:
    - **Name tag**: Private Subnet
    - **Availability Zone**: Same as *Public Subnet*
    - **IPv4 CIDR block**: 10.0.1.0/24

<u>Step 6: Creating a NAT gateway</u>

Create a NAT gateway, which enables resources in the *Private Subnet* to connect to the internet.

12. Create a NAT gateway that meets the following criteria:
    - **Name**: Lab NAT Gateway
    - **Subnet**: *Public Subnet*
    - Your NAT gateway needs an Elastic IP address.
13. Create a new route table that meets the following criteria:
    - **Name tag**: Private Route Table
    - **Destination**: 0.0.0.0/0
    - **Target**: *NAT Gateway*
14. Attach this route table to the *Private Subnet*, which you created earlier.

<u>Step 7: Creating an EC2 instance in the private subnet</u>

15. Create a new key pair named vockey2, and download the appropriate .ppk (Microsoft Windows) or .pem (macOS or Linux).
16. Create an EC2 instance in the *Private Subnet* of the *Lab VPC* that meets the following criteria.
    - **AMI**: *Amazon Linux 2023 AMI (HVM)*
    - **Instance type**: *t2.micro*
    - **Name**: Private Instance
    - Only allows the following traffic:
        - **Type**: *SSH*
        - **Port**: 22
        - **Source**: Bastion host security group
    - Uses the **vockey2** key pair that you created earlier

<u>Step 8: Configuring your SSH client for SSH passthrough</u>

Because the private instance you just created uses a different key pair than the bastion host, you must configure your SSH client to use SSH passthrough. This action allows you to use a key pair that's stored on your computer to access the private instance without uploading the key pair to the bastion host. This is a good security practice.

For macOS users, *ssh-agent* is already installed as part of the OS. To add your keys, complete the following steps.

17. Add your private keys to the keychain application by using the ssh-add command, with the -K option and the .pem file for the key.

```
ssh-add -K vockey2.pem
```

18. Make sure that you add both the *vockey.pem* and *vockey2.pem* keys that you downloaded. By adding the key to the agent, you can use SSH to connect to an instance without using the –i option when you connect.
19. To verify that the keys are available to ssh-agent, use the ssh-add command with the -L option, like the following example.

```
ssh-add –L
```

20. The agent should display the keys that it's stored.
21. Enter the following command (this command enables SSH agent forwarding by using the bastion host instance):

```
ssh –A ec2-user@<bastion-IP-address-or-DNS-entry>
```

22. After you're connected to the bastion host instance, you can use SSH to connect to a specific instance by entering a command like this example.

```
ssh user@<instance-IP-address-or-DNS-entry>
```

Step 9: Testing the SSH connection from the bastion host

We will now test the SSH connection from the bastion host to the EC2 instance that is running in the *Private Subnet*.

23. Connect to the bastion host instance by using SSH.
24. Connect to the private instance by using SSH and the IP address for the private instance.

```
ssh ec2-user@<private-ip-address-of-instance-in-private-subnet>
```

25. Now that you are connected to the EC2 instance in the *Private Subnet*, test its connection to the internet.

```
ping 8.8.8.8
```

Press CTRL+C to exit the command

26. You have now established communication between the *Bastion Host* in the *Public Subnet* and the EC2 instance in the *Private Subnet*, like in the following diagram:

Step 10: Creating a network ACL

27. Go to the **Amazon VPC console**, and inspect the default network ACL of *Lab VPC*. The subnets that you created are automatically associated with the default network ACL. The inbound and outbound rules of the default network ACL *allow* all traffic.
28. Create a custom network ACL called Lab Network ACL for the *Lab VPC*.

29. Configure your custom network ACL to *allow ALL traffic that goes into and out of* the *Private Subnet*.

Step 11: Testing your custom network ACL

30. Create an EC2 instance in the *Public Subnet* of the *Lab VPC*. It should meet the following criteria.
    - AMI: *Amazon Linux 2023 AMI (HVM)*
    - Instance type: *t2.micro*
    - Name: Test Instance
    - Allows *All ICMP – IPv4* inbound traffic to the instance through the security group
31. Note the private IP address of the *Test Instance*.
32. Test that you can reach the private IP address of the *Test Instance* from the *Private Instance*. From the *Private Instance* terminal window, run the following ping command:

```
ping <private-ip-address-of-test-instance>
```

33. Leave the *ping* utility running.
34. Modify your custom network ACL to *deny All ICMP – IPv4 traffic to* the <private-ip-address-of-test-instance>/32
    - Make sure to add /32 to the end of the private IP address.
    - Make sure that this rule is evaluated ***first***.
35. In the *Private Instance* terminal window, the ping command should stop responding. The traffic to the *Test Instance* has been blocked.

You have now denied traffic from the *Private Subnet* to the *Test Instance*.