Creating a VPC Networking Environment for the Café

Scenario

Sofía and Nikhil are now confident in their ability to create a two-tier architecture because of their experience migrating the café's data. They successfully moved from a MariaDB database on an Amazon Elastic Compute Cloud (Amazon EC2) instance to an Amazon Relational Database Service (Amazon RDS) database instance. In addition, they also moved their database resources from a public subnet to a private subnet.

When Mateo—a café regular and an AWS systems administrator and engineer—visits the café, Sofía and Nikhil tell him about the database migration. Mateo tells them that they can enhance security by running the café's application server in another private subnet that's separate from the database instance. They could then go through a bastion host (or jump box) to gain administrative access to the application server. The application server must also be able to download needed patches.

Knowing that the cloud makes experimentation easier, Sofía and Nikhil are eager to set up a nonproduction VPC environment. They can use it to implement the new architecture and test different security layers without accidentally disrupting the café's production environment.

Lab overview and objectives

In this lab, you use Amazon Virtual Private Cloud (Amazon VPC) to create a networking environment on AWS and implement security layers to protect your resources.

After completing this lab, you should be able to do the following:

- Create a virtual private cloud (VPC) environment that you can use to securely connect to private resources.
- Enable your private resources to connect to the internet.
- Create an additional layer of security by using a NAT gateway to control access to and from private resources.

When you start the lab, you will have only a VPC created for you in the AWS account.

At the end of this lab, your architecture should look like the following example:

(In the diagram, the communication arrows were omitted for simplicity.)

Note: In this challenge lab, step-by-step instructions are not provided for most of the tasks. You must figure out how to complete the tasks on your own.

Duration

This lab requires approximately 90 minutes to complete.

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab.
    - The lab session starts.
    - A timer displays at the top of the page and shows the time remaining in the session.
      Tip: To refresh the session length at any time, choose Start Lab again before the timer reaches 0:00.
    - Before you continue, wait until the circle icon to the right of the AWS link in the upper-left corner turns green. When the lab environment is ready, the AWS Details panel displays.
2. To connect to the AWS Management Console, choose the AWS link in the upper-left corner above the terminal window.
    - A new browser tab opens and connects you to the console.
      Tip: If a new browser tab does not open, a banner or icon is usually at the top of your browser with a message that says your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and then choose Allow pop-ups.
3. Arrange the AWS Management Console tab so that it displays alongside these instructions. Ideally, you have both browser tabs open at the same time so that you can follow the lab steps.

A business request for the café: Creating a VPC network that the café staff can use to remotely and securely administer the web application server (challenge 1)

In this challenge, you take on the role of one of the café's system administrators. You create and configure a VPC network so that you can securely connect from a bastion host in a public subnet to an EC2 instance in a private subnet. You also create a NAT gateway to enable the EC2 instance in your private subnet to access the internet.

Task 1: Creating a public subnet

Your first task in this lab is to create a public subnet in the Lab VPC. After you create a public subnet, you create an internet gateway to allow communication from the subnet to the internet. You update the routing table that's attached to the subnet to route internet-bound network traffic through the internet gateway.

4. Open the Amazon VPC console.
   Note that a VPC called Lab VPC has been created for you.
5. Create a public subnet with the following options:
   - VPC ID: Choose Lab VPC.
   - Subnet name: Enter Public Subnet.
   - Availability Zone: Choose Availability Zone a in your Region. For example, if your Region is us-east-1, choose us-east-1a.
   - IPv4 subnet CIDR block: Enter 10.0.0.0/24.
6. Create a new internet gateway and attach it to the Lab VPC.
7. Edit the route table that was created in your VPC by adding the route 0.0.0.0/0. For the target, choose the internet gateway that you created in the previous step.
   Hint: To successfully complete this task, you must create a few resources. If you get stuck, see the AWS Documentation.

Task 2: Creating a bastion host

In this task, you create a bastion host in the public subnet. In later tasks, you create an EC2 instance in a private subnet and connect to it from this bastion host.

8. From the Amazon EC2 console, create an EC2 instance in the public subnet of the Lab VPC. Configure the following options:

- For Name, enter Bastion Host.
- For Amazon Machine Image (AMI), choose Amazon Linux 2023 AMI.
- For Instance type, choose t2.micro.
- For Key pair name - required, choose vockey.
- In the Network settings section, choose Edit, and configure the following options:
  - For VPC - required, choose Lab VPC.
  - For Subnet, choose Public Subnet.
  - For Auto-assign public IP, choose Enable.
  - For Security group, create a security group called Bastion Host SG that allows only the following traffic:
    - For Type, choose ssh.
    - For Port, enter 22.
    - For Source type, choose My IP.

Note: In practice, hardening a bastion host involves more work than only restricting Secure Shell (SSH) traffic from your IP address. A bastion host is typically placed in a network that's closed off from other networks. It's often protected with multi-factor authentication (MFA) and monitored with auditing tools. Most enterprises require an auditable access trail to the bastion host.

Task 3: Testing the connection to the bastion host

In this task, you use the SSH key (.pem file or .ppk file) to test the SSH connection to your bastion host. This key was created for you.

9. In the upper-right area of these instructions, choose AWS Details.
10. Download the SSH key.
    - Microsoft Windows PuTTY users: Choose Download PPK.
    - macOS or Linux users: Choose Download PEM.
11. Note that the file is named labuser.*.
12. To close the window, choose Close.
13. Connect to your bastion host by using SSH.
14. After you have tested your connection to the bastion host, you can close the terminal or PuTTY.
    Hint: If you get stuck, see the AWS Documentation. This page provides detailed instructions about how to use SSH to connect to an EC2 instance.
    Note for Microsoft Windows users: If you don't have PuTTY installed, you must download and install PuTTY. It is recommend that you configure PuTTY so that your connection doesn't expire. To keep the PuTTY session open longer, for Seconds between keepalives, enter 30.

Task 4: Creating a private subnet

In this task, you create a private subnet in the Lab VPC.

14. In the Amazon VPC console, create a private subnet with the following options:

- VPC ID: Choose Lab VPC.
- Subnet name: Enter Private Subnet.
- Availability Zone: Choose the same Availability Zone that you choose for the Public Subnet in one of the previous tasks.
- IPv4 subnet CIDR block: Enter 10.0.1.0/24.

Task 5: Creating a NAT gateway

In this task, you create a NAT gateway, which enables resources in the Private Subnet to connect to the internet.

15. Create a NAT gateway that has the following options:

- Name - optional: Enter Lab NAT Gateway.
- Subnet: Choose Public Subnet.
- Elastic IP allocation ID: Choose Allocate Elastic IP.

Tip: Your NAT gateway needs an Elastic IP address.

16. Create a new route table that has the following options:

- Name - optional: Enter Private Route Table.
- VPC: Chose Lab VPC.

17. Attach this route table to the Private Subnet that you created earlier using the following options:

- Destination: Enter 0.0.0.0/0.
- Target: Choose NAT Gateway.

Hint: If you get stuck, see the AWS Documentation.

Task 6: Creating an EC2 instance in the private subnet

In this task, you create an EC2 instance in the Private Subnet, and you configure it to allow SSH traffic from the bastion host. You also create a new key pair to access this instance.

18. Create a new key pair named vockey2, and download the appropriate .ppk (Microsoft Windows) or .pem (macOS or Linux) file.
19. Create an EC2 instance in the Private Subnet of the Lab VPC that has the following options:
    - For Name, enter Private Instance.
    - For Amazon Machine Image (AMI), choose Amazon Linux 2023 AMI.
    - For Instance type, choose t2.micro.
    - For Key pair name - required, choose the vockey2 key pair that you created earlier.
    - In the Network settings section, choose Edit, and configure the following options:
        - For VPC - required, choose Lab VPC.
        - For Subnet, choose Private Subnet.
        - For Security group, create a security group called Private Instance SG that allows only the following traffic:
            - For Type, choose ssh.
            - For Port, enter 22.
            - For Source type, choose Custom.
            - For Source, choose the bastion host security group (Hint: for more information, see the AWS Documentation).

Task 7: Configuring your SSH client for SSH passthrough

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 gives you the ability 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.

To set up your client, follow either the Microsoft Windows steps, or the macOS or Linux steps.

Microsoft Windows users only

Windows users should complete the following steps.

20. Download and install Pageant, which is available from the PuTTY download page.

21. After you install Pageant, open it. Pageant runs as a Windows service.
22. To import the PuTTY-formatted key into Pageant, in the Windows system tray, choose the Pageant icon.

23. Choose Add Key.
24. Choose the .ppk file that you downloaded when you created the vockey2 key pair.
    Your screen should look similar to the following example.

25. Add the first vockey that you downloaded earlier. The filename is labsuser.*.

You should now have two keys listed. You can close the Pageant window.

26. In PuTTY, under Connection > SSH > Auth, choose Allow agent forwarding.
27. Expand Auth and choose Credentials.
28. For Private key file for authentication, choose Browse.
29. Browse to the labsuser.ppk file that you downloaded, select it, and choose Open.
30. Choose Accept.

After you have completed these step, continue to task Proceed to connecting to the bastion host by using PuTTY as you normally would, but don't open a .ppk file.

macOS or Linux users only

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

31. To add your private keys to the keychain application, use the ssh-add command with the -K option and the .pem file for the key. The command should look like the following example:

ssh-add -K vockey2.pem

32. 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.

33. To verify that the keys are available to ssh-agent, use the ssh-add command with the -L option. The command should look like the following example:
34. ssh-add –L
35. The agent should display the keys that are stored.
    After the key is added to your keychain, you can connect to the bastion host instance with SSH

by using the –A option. This option enables SSH agent forwarding. It also allows the local SSH agent to respond to a public key challenge when you use SSH to connect from the bastion host to a target instance in your VPC.

36. To connect to an instance in a private subnet, enter the following command. In this command, replace <bastion-IP-address-or-DNS-entry> with the IP address or DNS entry for the bastion host. This command enables SSH agent forwarding by using the bastion host instance:

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

35. After you're connected to the bastion host instance, to use SSH to connect to a specific instance, enter a command such as the following example. In this command, replace <instance-IP-address-or-DNS-entry> with the IP address or DNS entry for the instance:

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

Note: The ssh-agent doesn't know which key it should use for a given SSH connection. Therefore, ssh-agent will sequentially try all the keys that are loaded in the agent. Because instances terminate the connection after five failed connection attempts, make sure that the agent has five or fewer keys. Because each administrator should have only a single key, this is usually not a problem for most deployments. For details about how to manage the keys in ssh-agent, use the man ssh-agent command.

Task 8: Testing the SSH connection from the bastion host

In this task, you test the SSH connection from your bastion host to the EC2 instance that is running in the Private Subnet.

36. Connect to the bastion host instance by using SSH.

Tip: Use the connection method that was described in the SSH passthrough section.

37. Connect to the private instance by using SSH and the IP address for the private instance. Your command should look similar to the following. In this command, replace <private-ip-address-of-instance-in-private-subnet> with the private IP address of the instance in the private subnet:
38. ssh ec2-user@<private-ip-address-of-instance-in-private-subnet>
39. Now that you are connected to the EC2 instance in the Private Subnet, use the following command to test its connection to the internet:
40. ping 8.8.8.8
41. Tip: Press Ctrl+C to exit the command.
    You have now established a communication between the Bastion Host in the Public Subnet and the EC2 instance in the Private Subnet as in the following diagram:

Architecture best practice

In this first challenge, you implemented the architectural best practice of giving people the ability to perform actions at a distance.

Expand the following tip to learn more.

Tip 1

New business requirement: Enhancing the security layer for private resources (challenge 2)

Sofía and Nikhil are proud of the changes they made to the cafe's application architecture. They are pleased by the additional security they built, and they are also glad to have a test environment that they can use before they deploy updates to the production instance. They tell Mateo about their new application architecture, and he's impressed. To further improve their application security, Mateo advises them to build an additional layer of security by using custom network access control lists (network ACLs).

In this challenge, you continue to take on the role of one of the café's system administrators. Now that you established secure access from the bastion host to the EC2 instance in the private subnet, you must enhance the security layer of the private subnet. To accomplish this task, you create and configure a custom network ACL.

Task 9: Creating a network ACL

In this task, you create a custom network ACL to control traffic to and from the Private Subnet.

You can use network ACLs to control traffic between subnets. It's a good practice to use network ACLs to implement rules that are similar to your security group rules. The network ACLs provide an additional layer of protection.

For this challenge, you create an EC2 instance in the Public Subnet. You create a security group that allows Internet Control Message Protocol (ICMP) traffic from the local network. Next, you create and configure your custom network ACL to deny ICMP traffic between the Private Subnet and this test instance. ICMP is used by the ping utility.

39. Go to the Amazon VPC console, and inspect the default network ACL of the Lab VPC.

Note 1: The subnets that you created are automatically associated with the default network ACL.
Note 2: The inbound and outbound rules of the default network ACL allow all traffic.

40. Create a custom network ACL called Lab Network ACL for the Lab VPC.
    Note: The default inbound and outbound rules of the custom network ACL deny all traffic.
41. Configure your custom network ACL to allow all traffic that goes into and out of the Private Subnet.
    Hint: If you get stuck, see the AWS Documentation.

Task 10: Testing your custom network ACL

42. Create an EC2 instance in the Public Subnet of the Lab VPC with the following options.

- For Name, enter Test Instance
- For Amazon Machine Image (AMI), choose Amazon Linux 2023 AMI.
- For Instance type, choose t2.micro.
- For Key pair name - required, choose vockey.
- In the Network settings section, choose Edit, and configure the following options:

- For VPC - required, choose Lab VPC.
- For Subnet, choose Public Subnet.
- For Auto-assign public IP, choose Enable.
- For Security group, create a security group called Test SG, and configure the following options:
  - In the the Inbound Security Group Rules section, for Type, choose All ICMP – IPv4.
  - Leave all other values as default.

Note the private IP address of the Test Instance.

43. To 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. In the command, replace <private-ip-address-of-test-instance> with the private IP address of the Test Instance:

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

Leave the ping utility running.

44. 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.

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, as shown in the following diagram:

Architecture best practice

In this second challenge, you protected your network resources by implementing the architectural best practice of controlling traffic at all layers.

Expand the following tip to learn more.

Tip 2

Answering questions about the lab

Your answers are recorded when you choose Submit at the end of the lab.

45. To access the questions in this lab, at the top of these instructions, choose AWS Details.
46. Choose the Access the multiple choice questions link.

47. In the page that you loaded, answer the following questions:
    - Question 1: What is the purpose of the internet gateway in the public subnet?
    - Question 2: What allows the instance in the private subnet to connect to the internet so that it can download updates?
    - Question 3: Can the instance in the private subnet be accessed directly from the internet?
    - Question 4: Why do you use two different key pairs to access the private instance and the bastion host?
    - Question 5: Can the bastion host use ping and get a reply from the instance in the private subnet?
    - Question 6: Which security group rules allow the private EC2 instance to receive the return traffic when it pings the test instance?

Submitting your work

48. At the top of these instructions, choose Submit to record your progress, and when prompted, choose Yes.
49. If the results don't display after a couple of minutes, return to the top of these instructions and choose Grades.
    Tip: You can submit your work multiple times. After you change your work, choose Submit again. Your last submission is what will be recorded for this lab.
50. To find detailed feedback on your work, choose Submission Report.

Lab complete

Congratulations! You have completed the lab.

51. To confirm that you want to end the lab, at the top of this page, choose End Lab, and then choose Yes.

A message appears: Ended AWS Lab Successfully