Assignment 4: Creating a VPC with Subnets

Overview

This assignment provides a hands-on experience in designing and implementing a secure, highly available, and scalable network infrastructure using Amazon Virtual Private Cloud (VPC). You will build a multi-tier, multi-AZ network from the ground up, configure routing for public and private subnets, implement layered security controls, and establish private connectivity between VPCs and to AWS services. The objective is to develop a deep, practical understanding of core AWS networking components and best practices. This assignment was inspired by the work mentioned in Chapter 3 of the textbook.

Deliverables

Each student (or group) must submit the following:

1. **Presentation Slides (14–18 slides maximum)**
    - Title slide with student name(s), course, and assignment title.
    - Clear explanation of each completed task:
        - Task 1 – Custom VPC & Subnets
        - Task 2 – EC2 Instances in Public & Private Subnets
        - Task 3 – Internet Gateway Configuration
        - Task 4 – NAT Gateway & Private Route Table
        - Task 5 – VPC Peering between VPC1 and VPC2
        - Task 6 – Infrastructure as Code (Terraform)
        - Task 7 – Reflection Questions
    - Screenshots or videos of key steps/results.
    - Reflection question responses (may be included as slides or an appendix).
    - Lessons learned and challenges encountered.
2. **Demonstration (in-class or recorded)**
    - A 9–12 minute walkthrough of the slides and discussion of your results.
    - Optional short demo (if possible) showing:
        - Connecting to the Bastion Host in the public subnet via SSH.

    - From the Bastion Host, successfully connecting to a private EC2 instance.
    - From the private EC2 instance, proving it has outbound internet access (e.g., ping 8.8.8.8).
    - From the Bastion Host, successfully pinging an EC2 instance in the peered VPC.

Presentation Guidelines

- Use professional formatting (legible fonts, consistent colors, organized layout).
- Ensure clarity: avoid overcrowding slides with text; use bullet points and visuals.
- Provide screenshots or videos as evidence of work completed.
- Keep presentations within the time limit (6–9 minutes).
- Be prepared to answer questions about the process and challenges.

Detailed Grading Rubric (100 points total)

All team members are required to contribute to the presentation. The final score will be proportionally adjusted based on the number of members who actually present. For example, if a group consists of five students but only four deliver the presentation, and the maximum score is 100 points, the adjusted score will be calculated as:

$$Adjusted\ Score = \left(\frac{4}{5}\right) \times 100 = 80$$

1. Task Completion (60 points total)

Each AWS task is evaluated based on correctness, completeness, and evidence (e.g., screenshots/videos output).

- Task 1 – Custom VPC & Subnets **(9 pts)**: Create a custom VPC (VPC1) with correctly defined public and private subnets.
- Task 2 – EC2 Instances in Public & Private Subnets **(9 pts)**: Deploy EC2 instances in both subnets and verify connectivity between them.
- Task 3 – Internet Gateway Configuration **(9 pts)**: Enable Internet access for the public subnet using an Internet Gateway.
- Task 4 – NAT Gateway & Private Route Table **(9 pts)**: Provide outbound Internet access for private subnet using a NAT Gateway.
- Task 5 – VPC Peering between VPC1 and VPC2 **(9 pts)**: Establish and verify connectivity between two VPCs.
- Task 6 – Infrastructure as Code (Terraform) **(9 pts)**: Deploy equivalent AWS infrastructure using Terraform.
- Task 7 – Reflection Questions **(6 pts):** Complete and thoughtful answers for all questions (3 pts for each)

2. Presentation Quality (30 points total)

- Clear structure (intro, results, discussion, conclusion) **(10 pts)**
- Slides visually organized and professional, for example, adding pictures, transitions, animations **(10 pts)**
- Screenshots or videos included as evidence **(5 pts)**
- Proper timing (within 6–9 min) **(5 pts)**

3. Professionalism & Delivery (10 points total)

- Confident and clear presentation delivery **(4 pts)**
- Ability to answer questions from peers/instructor **(4 pts)**
- Team collaboration and equal participation **(2 pts)**

Submission Instructions

- Submit slides (PowerPoint or PDF) to Moodle by the deadline.
- Each group will present during the next class session.

Task 1: Create a Custom VPC with Public and Private Subnets

Understand how to create an isolated VPC network and define public and private subnets within it.

Instructions:

1. Navigate to the **VPC** service in the AWS Management Console.
2. In the top-right corner, ensure you are in your desired region (e.g., **Sydney (ap-southeast-2)**).
3. Click **Create VPC**.
4. Select **VPC only**.
5. Under **Name tag**, enter `VPC1`.
6. For **IPv4 CIDR block**, enter `10.10.0.0/16`.
7. Leave all other settings as default and click **Create VPC**.
8. Once the VPC is created, go to **Subnets** in the left-hand menu.
9. Click **Create subnet**.
10. Select **VPC1** from the **VPC ID** dropdown.
11. Define your **public subnet**:

    - **Subnet name:** `public-subnet`
    - **Availability Zone:** Choose any (e.g., `ap-southeast-2a`).
    - **IPv4 CIDR block:** `10.10.1.0/24`

12. Click **Add new subnet** to add a second subnet configuration in the same window.
13. Define your **private subnet**:

- **Subnet name:** `private-subnet`
- **Availability Zone:** Choose the same one as your public subnet (e.g., `us-east-1a`).
- **IPv4 CIDR block:** `10.10.2.0/24`

14. Click **Create subnet** at the bottom of the page.
15. Select your `public-subnet`. In the **Details** tab, observe the **Available IPv4 addresses**. It should be **251** (256 total IPs minus 5 reserved by AWS), confirming the address allocation.

Task 2: Launch EC2 Instances in Each Subnet

Deploy public and private compute resources within your custom VPC.

Instructions:

1. Navigate to the **EC2** service.
2. Click **Launch Instance**.
3. **Launch EC2-1 (Public Instance):**

    - **Name:** `EC2-1`
    - **Application and OS Images (AMI):** Select **Amazon Linux 2 AMI** (Free tier eligible).
    - **Instance type:** Select **t2.nano** (Free tier eligible).
    - **Key pair (login):** Create or select an existing key pair. You **must** have access to the `.pem` private key file.
    - **Network settings:** Click **Edit**.
        - **VPC:** Select **VPC1**.
        - **Subnet:** Select **public-subnet**.
        - **Auto-assign public IP:** Select **Enable**.
    - **Firewall (security groups):**
        - Select **Create a new security group**.
        - **Security group name:** `public-sg`
        - **Description:** `Allows SSH and ICMP`
        - **Inbound rules:**
            - Rule 1: Type **SSH**, Source **My IP** (This is more secure. If your IP changes, use `0.0.0.0/0`, but be aware this is less secure).
            - Rule 2: Click **Add security group rule**. Type **All ICMP - IPv4**, Source **Custom** `10.10.0.0/16` (This allows ping from *anywhere* inside VPC1).

4. Click **Launch instance**.
5. **Launch EC2-2 (Private Instance):**

    - Click **Launch Instance** again.
    - **Name:** `EC2-2`
    - **AMI:** Select **Amazon Linux 2 AMI**.
    - **Instance type:** Select **t2.nano**.
    - **Key pair (login):** Select the **same key pair** you used for EC2-1.

- **Network settings:** Click **Edit**.
    - **VPC:** Select **VPC1**.
    - **Subnet:** Select **private-subnet**.
    - **Auto-assign public IP:** Select **Disable**.
- **Firewall (security groups):**
    - Select **Create a new security group**.
    - **Security group name:** `private-sg`
    - **Description:** `Allows SSH and ICMP from within VPC`
    - **Inbound rules:**
        - Rule 1: Type **All ICMP - IPv4**, Source **Custom** `10.10.0.0/16` (Allows ping from VPC1).
        - Rule 2: Click **Add security group rule**. Type **SSH**, Source **Custom** `10.10.1.0/24` (Allows SSH *only* from your public subnet).

6. Click **Launch instance**.

Task 3: Enable Internet Access for Public Subnet

Connect your VPC to the Internet using an Internet Gateway (IGW) to allow access for public workloads.

Instructions:

1. Navigate to the **VPC** service.
2. In the left-hand menu, click **Internet Gateways**.
3. Click **Create internet gateway**.
4. **Name tag:** `IGW1`. Click **Create internet gateway**.
5. Select `IGW1` from the list. Click **Actions** -> **Attach to VPC**.
6. Select **VPC1** from the dropdown and click **Attach internet gateway**.
7. In the left-hand menu, click **Route Tables**.
8. You will see a default route table for VPC1. Select it.
9. **Name** it `public-rt` for clarity.
10. With `public-rt` selected, click the **Routes** tab below.
11. Click **Edit routes**.
12. Click **Add route**.

- **Destination:** `0.0.0.0/0`
- **Target:** Select **Internet Gateway** and then choose **IGW1**.

13. Click **Save changes**.
14. Now, you must associate this route table with your public subnet. Click the **Subnet Associations** tab.
15. Click **Edit subnet associations**.

16. Check the box for **public-subnet** and click **Save associations**.
17. You should see a successful ping response, confirming the security groups are working
18. **Verify Internet Access:**

    - SSH into your **EC2-1** instance (if you're not already).
    - Run the command: `ping 8.8.8.8`
    - You should get a successful response, as traffic to `0.0.0.0/0` is now routed to the IGW.
19. **Test Connectivity:**

    - Go to your **EC2 Instances** list. Wait for both instances to be in the **Running** state.
    - Select **EC2-1** and copy its **Public IPv4 address**.
    - Select **EC2-2** and copy its **Private IPv4 address** (e.g., `10.10.2.x`).
    - Open your terminal, `cd` to where your key pair file is stored, and SSH into **EC2-1**:

      ```
      ssh -i "your-key.pem" ec2-user@<EC2-1-Public-IP>
      ```

    - Once you are logged into EC2-1, ping EC2-2 using its private IP:

      ```
      ping <EC2-2-Private-IP>
      ```

Task 4: Provide Outbound Internet Access for Private Subnet (NAT Gateway)

Allow private instances (like EC2-2) to access the Internet for updates without exposing them to inbound connections.

Instructions:

1. Navigate to the **VPC** service.
2. In the left-hand menu, click **NAT Gateways**.
3. Click **Create NAT gateway**.

    - **Name:** `NAT1`
    - **Subnet:** Select **public-subnet** (A NAT Gateway must reside in a public subnet).
    - **Connectivity type: Public**.
    - **Elastic IP allocation:** Click **Allocate Elastic IP**.

4. Click **Create NAT gateway**. (Provisioning may take a few minutes).
5. While it's provisioning, go to **Route Tables** and click **Create route table**.

    - **Name:** `private-rt`
    - **VPC:** Select **VPC1**.

6. Click **Create route table**.
7. Select your new `private-rt` from the list.
8. In the **Routes** tab, click **Edit routes**.

9. Click **Add route**.

   - Destination: `0.0.0.0/0`
   - **Target:** Select **NAT Gateway** and then choose **NAT1**.

10. Click **Save changes**.
11. Click the **Subnet Associations** tab for `private-rt`.
12. Click **Edit subnet associations**.
13. Check the box for **private-subnet** and click **Save associations**.
14. **Verify Outbound Access:**

   - SSH into **EC2-1** (your public "bastion" host).
   - From EC2-1, SSH into **EC2-2**. You can do this using **ssh-agent forwarding** (recommended) or by copying your `.pem` file to EC2-1.
   - **Method 1 (Agent Forwarding):**
     - On your *local* machine, add your key: `ssh-add your-key.pem`
     - SSH to EC2-1 with the `-A` flag: `ssh -A ec2-user@<EC2-1-Public-IP>`
     - From EC2-1, SSH to EC2-2: `ssh ec2-user@<EC2-2-Private-IP>`
   - Once you are logged into **EC2-2**, test its internet access:

     ```
     curl google.com
     ```

   - You should get an HTML response. `ping 8.8.8.8` will also work. This confirms EC2-2 can reach the internet via the NAT Gateway.

Task 5: Peer Two VPCs

Enable private, direct communication between two VPCs as if they were on the same network.

Instructions:

1. **Create VPC2:**

   - Go to the **VPC** service -> **Create VPC**.
   - Select **VPC only**. **Name:** `VPC2`. **IPv4 CIDR:** `8.8.0.0/16`.
   - Click **Create VPC**.

2. **Create Subnet for VPC2:**

   - Go to **Subnets** -> **Create subnet**.
   - **VPC ID: VPC2**. **Subnet name:** `subnet-8`. **IPv4 CIDR:** `8.8.8.0/24`.
   - Click **Create subnet**.

3. **Launch EC2-8 in VPC2:**

- Go to **EC2** -> **Launch Instance**.
- **Name:** `EC2-8`. **AMI: Amazon Linux 2**. **Type: t2.nano**.
- **Key pair:** Select the **same key pair** as before.
- **Network settings (Edit):**
    - **VPC: VPC2**. **Subnet: subnet-8**.
    - **Auto-assign public IP: Disable**.
- **Firewall (Security Groups):**
    - Select **Create a new security group**.
    - **Security group name:** `vpc2-sg`
    - **Inbound rules:** Add rule: Type **All ICMP - IPv4**, Source **Custom** `10.10.0.0/16` (VPC1's CIDR).
- Click **Launch instance**.

4. **Create Peering Connection:**

- Go to the **VPC** service -> **Peering connections** -> **Create peering connection**.
- **Name:** `vpc1-vpc2-peer`.
- **Requester VPC: VPC1**.
- **Accepter VPC: VPC2**.
- Click **Create peering connection**.

5. **Accept Peering Connection:**

- Select the new "Pending acceptance" connection.
- Click **Actions** -> **Accept request**. Confirm.

6. **Update All Route Tables (Critical Step):**

- Go to **Route Tables**.
- **For VPC1 (`public-rt`):**
    - Select `public-rt`. **Routes** tab -> **Edit routes** -> **Add route**.
    - **Destination:** `8.8.0.0/16` (VPC2's CIDR).
    - **Target:** Select **Peering Connection** and choose `vpc1-vpc2-peer`. Save.
- **For VPC1 (`private-rt`):**
    - Select `private-rt`. **Routes** tab -> **Edit routes** -> **Add route**.
    - **Destination:** `8.8.0.0/16`.
    - **Target:** Select **Peering Connection** and choose `vpc1-vpc2-peer`. Save.
- **For VPC2 (Main Route Table):**
    - Select the main route table for **VPC2**.
    - **Routes** tab -> **Edit routes** -> **Add route**.
    - **Destination:** `10.10.0.0/16` (VPC1's CIDR).
    - **Target:** Select **Peering Connection** and choose `vpc1-vpc2-peer`. Save.

7. Test Peering:

- SSH into **EC2-1** (in VPC1).

- Find the **Private IPv4 address** of **EC2-8** (e.g., `8.8.8.x`).
- From EC2-1, ping EC2-8: `ping <EC2-8-Private-IP>`.
- The ping should be successful, proving the peering connection and routes are working.

8. **Remember to delete all resources you have created so far to avoid incurring cost.**

Task 6: Deploy VPC and EC2 Instances using Terraform (IaC)

Understand how to deploy a full-stack environment (VPC, subnets, gateways, and instances) automatically using Infrastructure as Code (IaC) with Terraform. This task replicates the manual work from Tasks 1-4 using a declarative script.

Instructions:

1. **Prerequisite – Install AWS CLI:** Ensure you have the AWS ALI, more information on how to install:

   https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

   or you can simply install with command (Windows):

   ```
   msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
   ```

2. **Prerequisite - Install Terraform:** Ensure you have the Terraform CLI installed on your local machine.
3. **Prerequisite - Create Key Pair:**

   - Log in to the AWS Management Console and navigate to the **EC2** service.
   - In the left-hand menu, go to **Network & Security** -> **Key Pairs**.
   - Click **Create key pair**.
   - **Name:** `terraform-lab4-key`
   - **Key pair type: RSA**, **Private key file format: .pem**
   - Click **Create key pair** and save the `terraform-lab4-key.pem` file securely.

4. **Prepare Project Directory:**

   - Create a new folder for your project (e.g., `terraform-vpc-lab`).
   - Inside this folder, paste the three files provided: `main.tf`, `variables.tf`, and `outputs.tf`.
   - Create one more file in the *same directory* named `terraform.tfvars`.
   - Add the following content to your `terraform.tfvars` file (this is the *only* file you need to edit):

   ```
   aws_key_name = "terraform-lab4-key"
   instance_ami = "<AMI-ID>"
   aws_region = "<AWS-Region>"
   ```

Initialize Terraform:

- Open your terminal and navigate into your `terraform-vpc-lab` directory.
- Run the `init` command to download the necessary AWS provider plugins:

```
terraform init
```

5. **Plan the Deployment:**

- Run the `plan` command. This will check your script and show you all the resources (VPC, subnets, instances, etc.) that will be created.

```
terraform plan
```

- Review the output to ensure there are no errors.

6. **Apply the Configuration:**

- Run the `apply` command to build the infrastructure.

```
terraform apply
```

- Terraform will show you the plan again. Type `yes` and press **Enter** to approve the deployment.
- Wait for 3-5 minutes as Terraform provisions all the resources. When it's finished, it will display the "Outputs" (like the public IP of your instance).

7. **Verify in Console:**

- Go to the AWS Management Console.
- Navigate to the **VPC** and **EC2** services. You will see all the new resources, such as `VPC1-Terraform`, `EC2-1-Terraform`, and `EC2-2-Terraform`.

8. **Destroy Resources (CRITICAL):**

- To avoid incurring costs, you **must** destroy all resources when you are finished.
- In your terminal, run the `destroy` command:

```
terraform destroy
```

- Type `yes` and press **Enter** to confirm. Terraform will now tear down and delete all the resources it created.

Task 7: Answer reflection questions

- Review the questions below, which cover the concepts applied in this assignment.

- **Choose any two (2)** of the seven options to answer.
- Provide clear, concise, and well-reasoned answers in your presentation.

Reflection Questions (Choose 2):

1. What is the key difference between an Internet Gateway and a NAT Gateway? Explain the specific purpose each one served in this assignment's architecture.
2. Explain the difference between a Security Group (stateful) and a Network ACL (stateless). Why is it a best practice to use both as a "defense in depth" strategy?
3. Why must a NAT Gateway be placed in a public subnet, and what other two components did it depend on to function correctly?
4. If you add a new private subnet to your VPC (e.g., for a new service), what is the minimum number of steps or resources (like route table associations) required to grant its instances outbound internet access?
5. What would happen if you tried to access S3 from the private instance without the endpoint or a NAT Gateway?
6. When you configured VPC Peering, you had to update route tables in both VPCs. Why is this necessary? What would happen if you only updated the route table in the first VPC?
7. This assignment used two Availability Zones for high availability. How does placing subnets in a second AZ protect your application? What would be the single point of failure if your entire VPC (all subnets) was in a single AZ?