### **Lab Guide: Understanding Amazon VPC and Deploying Resources**

## **Objective**

This lab will guide you through:

1. Understanding Virtual Private Cloud (VPC).
2. Creating a custom VPC with subnets.
3. Configuring an Internet Gateway (IGW) and route tables.
4. Launching EC2 instances in public and private subnets.
5. Setting up security groups and Network ACLs.
6. Configuring NAT Gateway for private subnet access.

By the end of this lab, you should have a hands-on understanding of how to design and deploy resources in a secure VPC environment.

## **Prerequisites**

1. **AWS Account** with administrative access.
2. **AWS CLI Installed** (optional for CLI-based tasks).
3. **Basic understanding of networking concepts** (IP ranges, subnets, NAT, security groups).

## **Part 1: Create a Custom VPC**

### **Step 1: Navigate to the VPC Dashboard**

1. Log into the AWS Management Console.
2. Search for **VPC** in the AWS services search bar.

A screenshot of a computer

AI-generated content may be incorrect.

1. Click **Your VPCs** and then **Create VPC select (VPC Only)**.

**Summary:** Created a Virtual Private Cloud.

The purpose of VPC is to provide secured connectivity, network isolation, and access control.

### **Step 2: Configure the VPC**

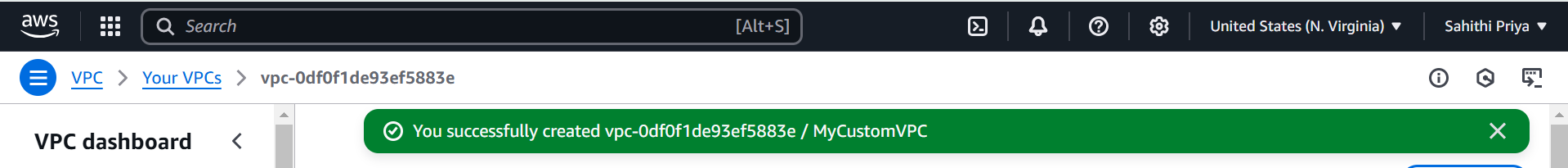
1. **Name Tag**: MyCustomVPC
2. **IPv4 CIDR Block**: 10.0.0.0/16
3. **Tenancy**: Default
4. Click **Create VPC**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



**Summary:** Created a VPC with a name tag MyCustomVPC under IPv4 CIDR block 10.0.0.0/16 at a default tenancy.

## **Part 2: Create Public and Private Subnets**

### **Step 1: Create a Public Subnet**

1. Navigate to **Subnets** in the VPC Dashboard.
2. Click **Create Subnet**.

A screenshot of a computer

AI-generated content may be incorrect.

1. **VPC ID**: Select MyCustomVPC.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Subnet Name**: Public-Subnet
2. **IPv4 CIDR Block**: 10.0.1.0/24
3. **Availability Zone**: Choose one (e.g., us-east-1a).

A screenshot of a computer

AI-generated content may be incorrect.

1. Click **Create Subnet**.

A screenshot of a computer

AI-generated content may be incorrect.

### **Step 2: Enable Auto-Assign Public IP**

1. Select Public-Subnet and click **Actions > Edit Subnet Settings**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Enable **Auto-assign IPv4**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Click **Save**.

A screenshot of a computer

AI-generated content may be incorrect.

**Summary:** A public subnet, “Public-Subnet,” is created. A public subnet has direct access to the Internet, and it is why Auto-Assign Public IP is kept enabled. Public subnets are usually used to host web servers for websites or applications.

### **Step 3: Create a Private Subnet**

1. Repeat the same steps as above but use the following details:
   * **Subnet Name**: Private-Subnet
   * **IPv4 CIDR Block**: 10.0.2.0/24

**Summary:**  A private subnet, “Private-Subnet”, is created. Auto-Assign Public IP is disabled because it is designed to isolate resources from the direct internet access for security purposes.

## **Part 3: Configure Internet Gateway and Route Tables**

### **Step 1: Create an Internet Gateway (IGW)**

1. Navigate to **Internet Gateways**.
2. Click **Create Internet Gateway**.
3. **Name Tag**: MyIGW
4. Click **Create Internet Gateway**.

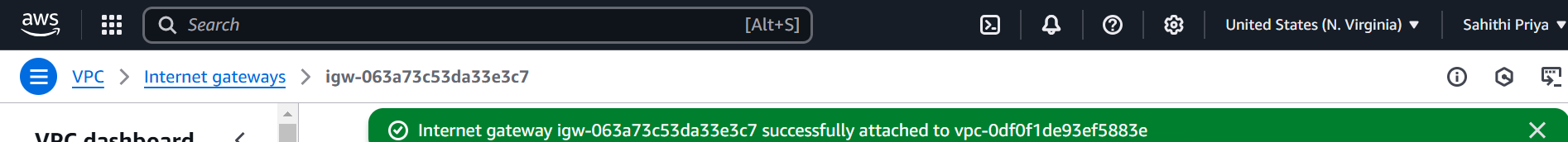
A green and white rectangle with black text

AI-generated content may be incorrect.

1. Select MyIGW and attach it to MyCustomVPC.

A screenshot of a computer

AI-generated content may be incorrect.



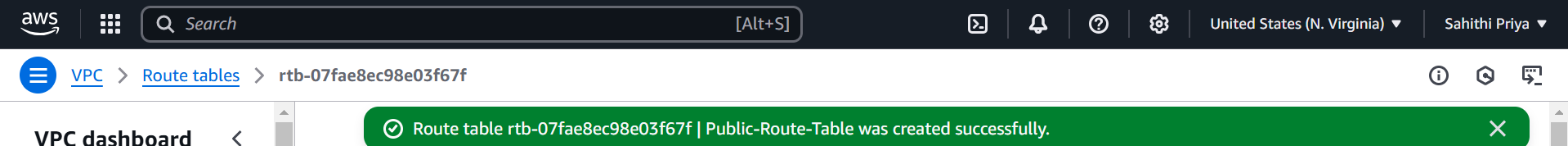
**Summary:** Created an Internet Gateway MyIGW and attached it to MyCustomVPC. An IGW is a VPC component that enables communication between instances in a VPC and the internet. An IGW must be attached to a VPC because an IGW enables an internet connection to a VPC. Even the public subnets cannot access the internet without IGW.

### **Step 2: Create and Modify Route Tables**

1. Navigate to **Route Tables**.
2. Click **Create Route Table**.
   * **Name**: Public-Route-Table
   * **VPC**: Select MyCustomVPC
   * Click **Create**.

A screenshot of a computer

AI-generated content may be incorrect.



1. Select Public-Route-Table, go to **Routes**, edit routes and add:
   * **Destination**: 0.0.0.0/0
   * **Target**: Internet Gateway (MyIGW)

A screenshot of a computer

AI-generated content may be incorrect.

1. Click **Save Routes**.
2. Associate Public-Route-Table with Public-Subnet.

A screenshot of a computer

AI-generated content may be incorrect.

**Summary:** Created route tables “Public-Route-Table” in “MyCustomVPC” and then associated the created route table with the Public-Subnet. Route tables are essential to control which resources can access the internet. Associating the route table to the public subnet enables internet access for public instances.

## **Part 4: Launch EC2 Instances in the VPC**

### **Step 1: Launch a Public EC2 Instance**

1. Navigate to **EC2 > Launch Instance**.
2. **Choose AMI**: Amazon Linux 2 (Free Tier Eligible).
3. **Instance Type**: t2.micro. or t3.micro
4. **Network**: MyCustomVPC.
5. **Subnet**: Public-Subnet.
6. **Auto-assign Public IP**: **Enabled**.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Security Group**: Create a new one with the following rules:
   * **SSH (Port 22)**: Allow from **Anywhere**.
   * **HTTP (Port 80)**: Allow from **Anywhere**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Click **Launch**.

A green and white rectangle

AI-generated content may be incorrect.

### **Step 2: Launch a Private EC2 Instance**

1. Repeat the above steps but choose Private-Subnet and **disable auto-assign public IP**.
2. **Security Group**: Allow SSH only from the public EC2 instance (use the public EC2 ip address - figure that out).

A green and white screen

AI-generated content may be incorrect.

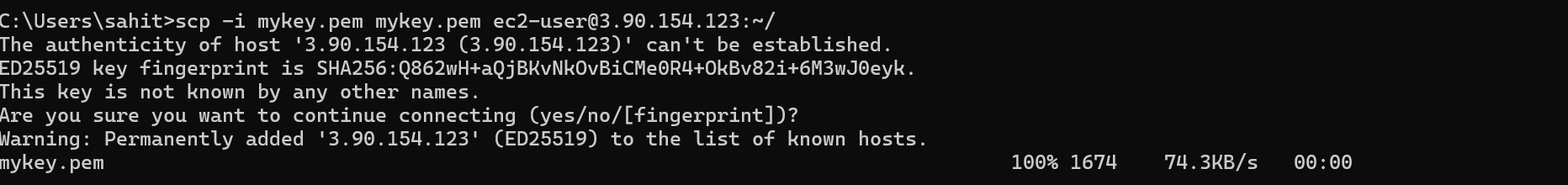
**Summary:** A public EC2 instance is launched keeping the auto-assigned public IP enabled within the VPC network and public subnet. Then, a private EC2 instance is launched by disabling the auto-assign public IP within the private subnet.

## **Part 5: Copy the Key to the Public Instance for Private SSH Access**

### **\*\*Step \*\*1: Transfer the Key from Local Machine to Public Instance**

Run the following command from your local machine:

scp -i mykey.pem mykey.pem ec2-user@<Public\_Instance\_IP>:~/



### **Step 2: SSH into the Public Instance**

****ssh -i mykey.pem ec2-user@<Public\_Instance\_IP>

### **Step 3: Set Proper Permissions on the Key**

On the public instance, run:

chmod 400 mykey.pem

### **Step 4: SSH from Public to Private Instance**

****ssh -i mykey.pem ec2-user@<Private\_Instance\_IP>

A computer screen shot of white text

AI-generated content may be incorrect.

**Summary:** Here, private instances do not have direct access to the internet. So, a public EC2 instance acts as a bridge between the local machine and private instances. So, copying SSH key to the public instance will allow secure access to private EC2 instances.



## **Part 6: Configure NAT Gateway for Private Subnet**

1. Navigate to **NAT Gateways**.
2. Click **Create NAT Gateway**.
3. **Subnet**: Choose Public-Subnet.
4. **Elastic IP**: Click **Allocate New Elastic IP**.
5. Click **Create NAT Gateway**.

A screenshot of a computer

AI-generated content may be incorrect.

### **Step 2: Update Private Route Table**

1. Navigate to **Route Tables**.
2. Select the default **private route table**.
3. Add a new route:
   * **Destination**: 0.0.0.0/0
   * **Target**: NAT Gateway

A screenshot of a computer

AI-generated content may be incorrect.

1. Save the routes.

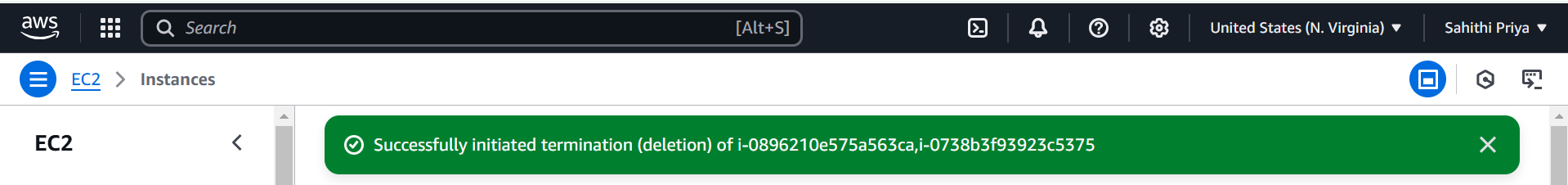
**Summary:** NAT gateways are used to allow private EC2 instances to access the internet. Usually, private subnets possess confidential data which cannot be exposed. In these cases, NAT gateways allow controlled internet access to these.

Part 5: Clean Up Resources

To avoid unnecessary costs, clean up all resources created:

Step 1: Terminate EC2 Instances

1. Navigate to EC2 Dashboard.
2. Select both public and private instances.
3. Click Actions > Instance State > Terminate.
4. Confirm termination.



Step 2: Delete NAT Gateway and Release Elastic IP

1. Navigate to VPC Dashboard > NAT Gateways.
2. Select the NAT Gateway and click Delete NAT Gateway.

A screenshot of a computer

AI-generated content may be incorrect.

1. Navigate to EC2 Dashboard > Elastic IPs.
2. Select the Elastic IP assigned to the NAT Gateway and click Release Elastic IP.

A green rectangular object with white border

AI-generated content may be incorrect.

Step 3: Delete Subnets

1. Navigate to VPC Dashboard > Subnets.
2. Select Public-Subnet and Private-Subnet and click Delete.

A green and white box with white text

AI-generated content may be incorrect.

Step 4: Detach and Delete Internet Gateway

1. Navigate to VPC Dashboard > Internet Gateways.
2. Select MyIGW.
3. Click Detach from VPC, then click Delete Internet Gateway.

A screenshot of a computer

AI-generated content may be incorrect.



Step 5: Delete the VPC

1. Navigate to VPC Dashboard > Your VPCs.
2. Select MyCustomVPC and click Delete VPC.

A screenshot of a computer

AI-generated content may be incorrect.

**Summary:** At the end, all the resources are cleaned up. It is to maintain security, compliance, and cost efficiency of these resources.

## 

## **Deliverables**

1. **Screenshots of:**
   * VPC and subnet creation.
   * Route table settings.
   * Security group settings.
   * NAT Gateway configuration.
   * Successful SSH connection and internet access test.
2. **Summary Report:**
   * Describe each step performed.
   * Explain the purpose of each VPC component.
   * Mention any issues faced and solutions applied.

Note: Delete all the respiurces you created