

**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

Secure Access with a Bastion Host : Set up a bastion host in a public subnet to securely access instances in a private subnet.

Name: S Akshaya Department:ADS



**Introduction**

A bastion host is a secure server that acts as a bridge between public and private networks. In cloud environments, a bastion host is used to securely access instances in private subnets, as direct internet access is restricted for security reasons. This Proof of Concept (POC) demonstrates how to set up a bastion host in AWS to access private instances while ensuring robust network security.

# Overview

In this POC, we design and implement a secure architecture using AWS services. The project involves:

1. Creating a custom Virtual Private Cloud (VPC) with public and private subnets.
2. Launching an EC2 instance (bastion host) in the public subnet and a private instance in the private subnet.
3. Configuring security groups to control network traffic and enable secure access.
4. Using the bastion host as an intermediary to SSH into the private instance without exposing it directly to the internet.

The POC verifies secure access by testing connectivity, verifying the private instance's setup, and ensuring proper configurations.

# Objectives

The primary objectives of this POC are:

1. **Learn Network Segmentation:**

Understand how to segregate public and private resources within a VPC.

1. **Secure Private Resources:**

Enable access to private instances without exposing them to the internet.

1. **Practice Secure Access Techniques:**

Use a bastion host to securely SSH into a private instance.

1. **Apply Security Best Practices:**

Use key-based authentication, restrict inbound traffic, and follow the principle of least privilege in security group configurations.

**Step-by-Step Overview**

Step 1:

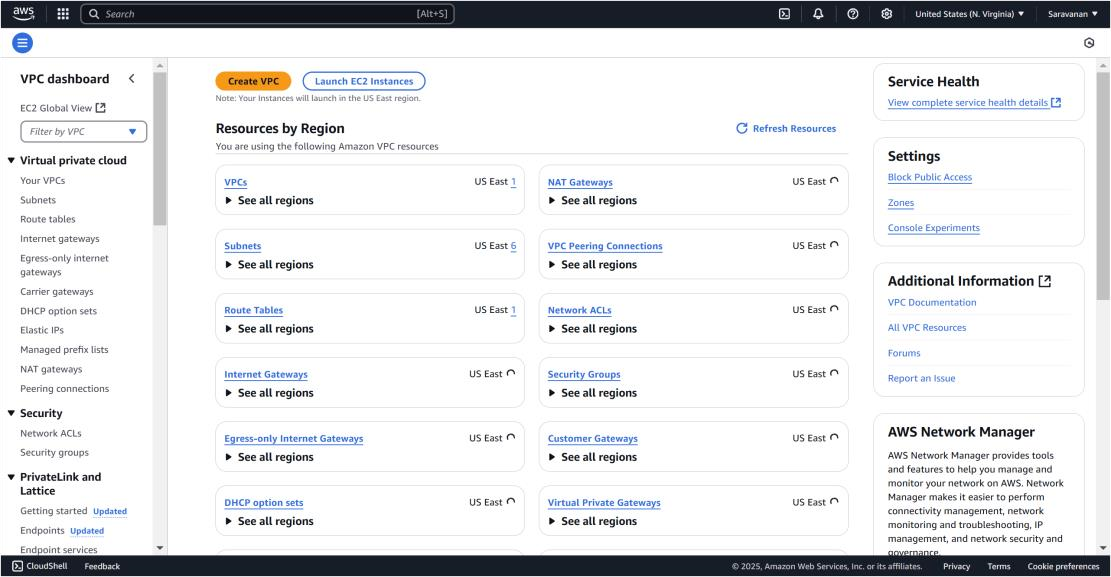
1. Go to [AWS Management Console.](https://aws.amazon.com/console/)
2. Enter your username and password to log in.



Step 2:

Search for **VPC** in the AWS search bar and click on it.

Click on **Create VPC**.

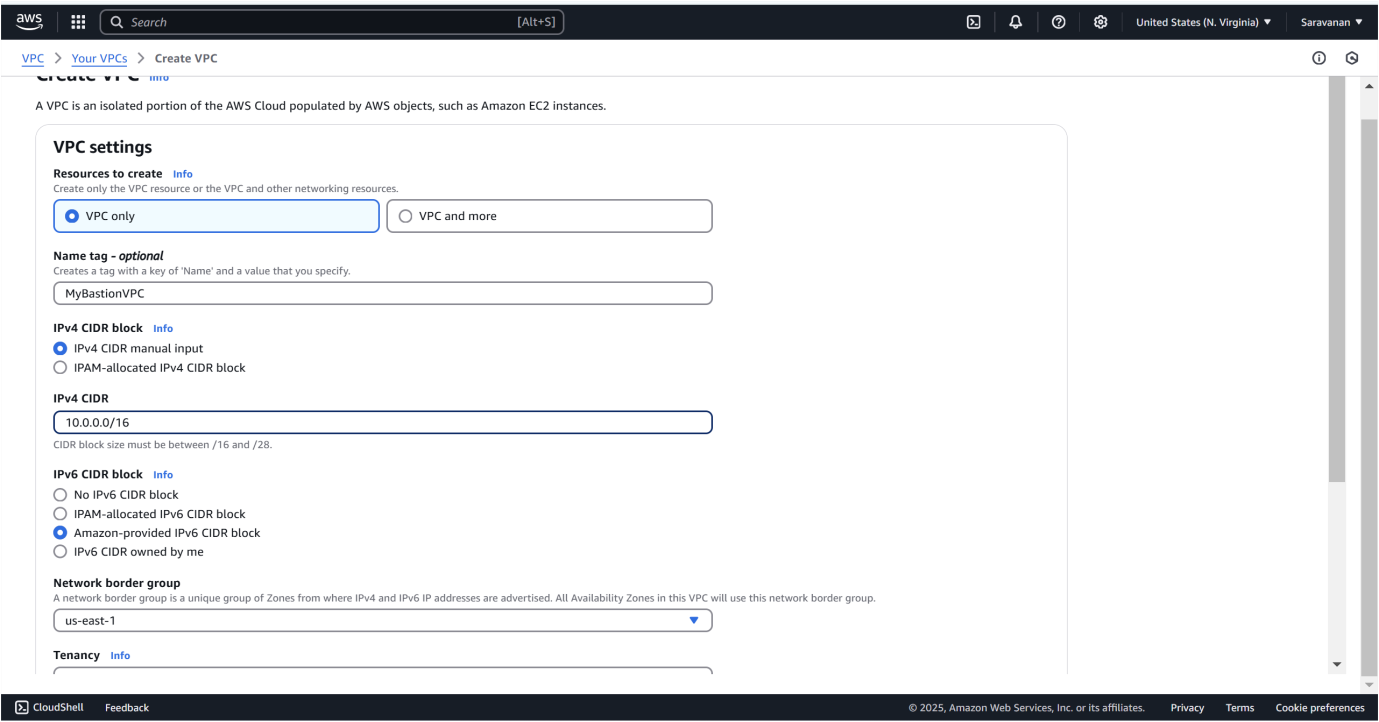


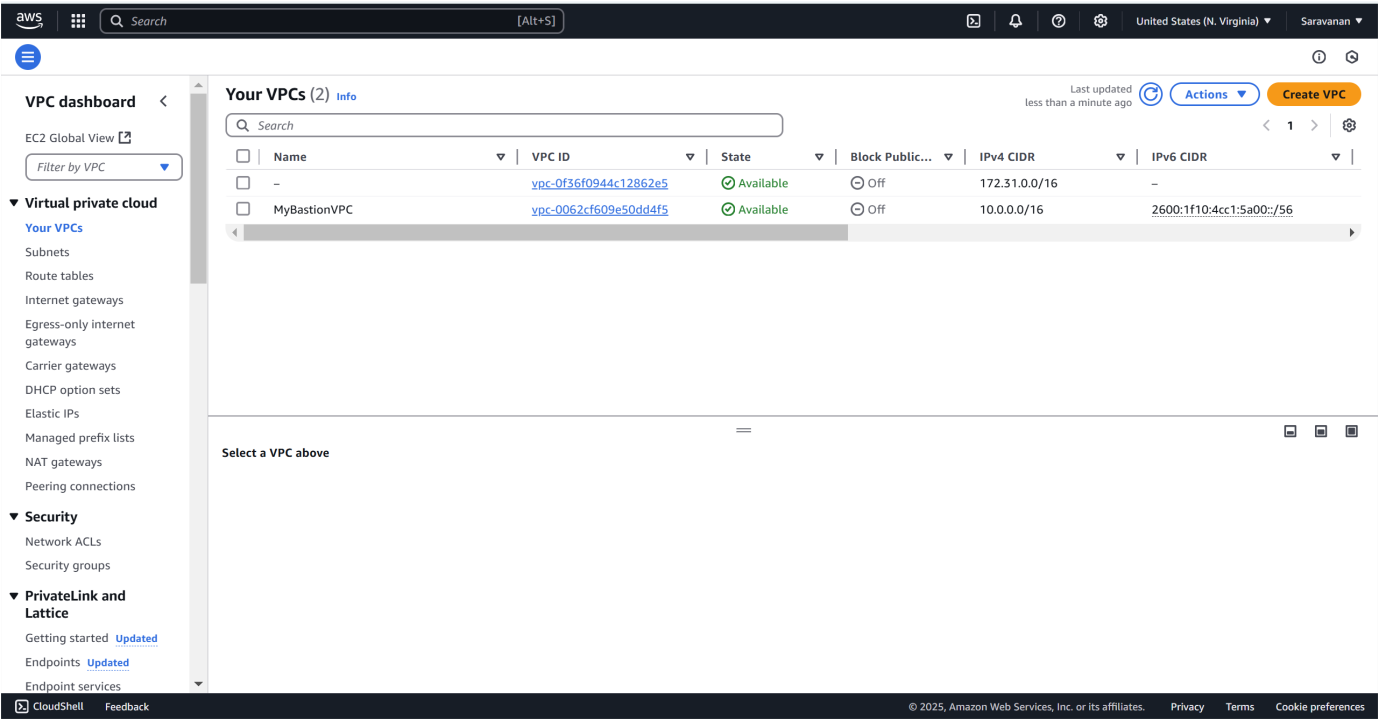
Step 3:

Create a new VPC by selecting **VPC only** and filling in the following details: set the **Name Tag** as *MyBastionVPC* and the **IPv4 CIDR**

**Block** as *10.0.0.0/16*. Leave all other settings as default, then click

**Create VPC**. Once created, the new VPC will appear in the VPC list.

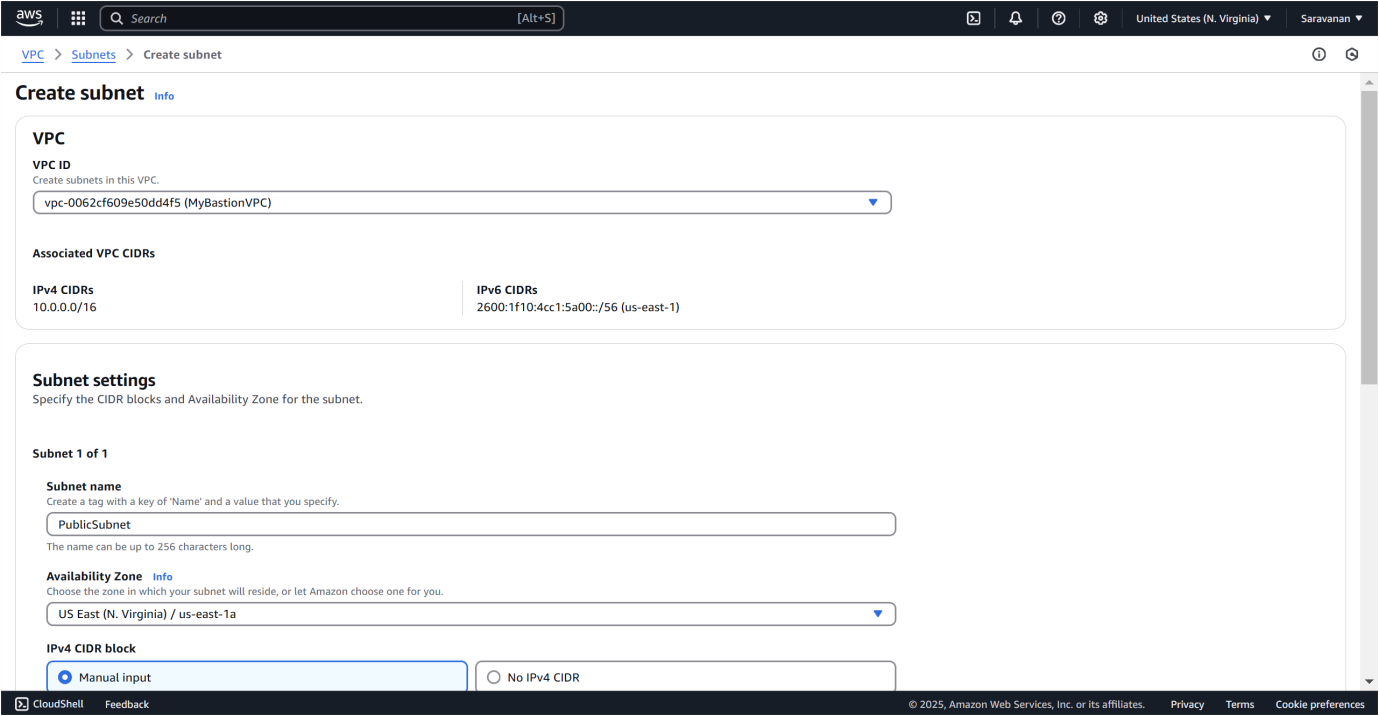




Step 4:

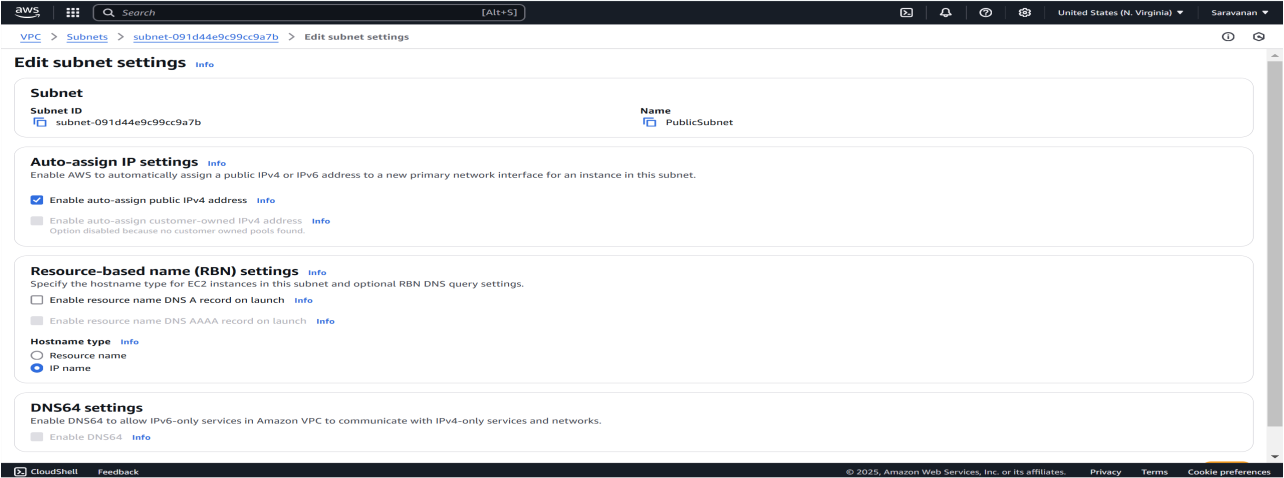
In the **VPC Dashboard**, go to **Subnets** and click **Create Subnet**.

Select the **VPC ID** of the VPC you created earlier (*MyBastionVPC*). Enter the **Subnet Name** as *PublicSubnet*, choose an **Availability Zone** (e.g., *us-east-1a*), and set the **IPv4 CIDR Block** as *10.0.1.0/24*. Click **Create Subnet**.

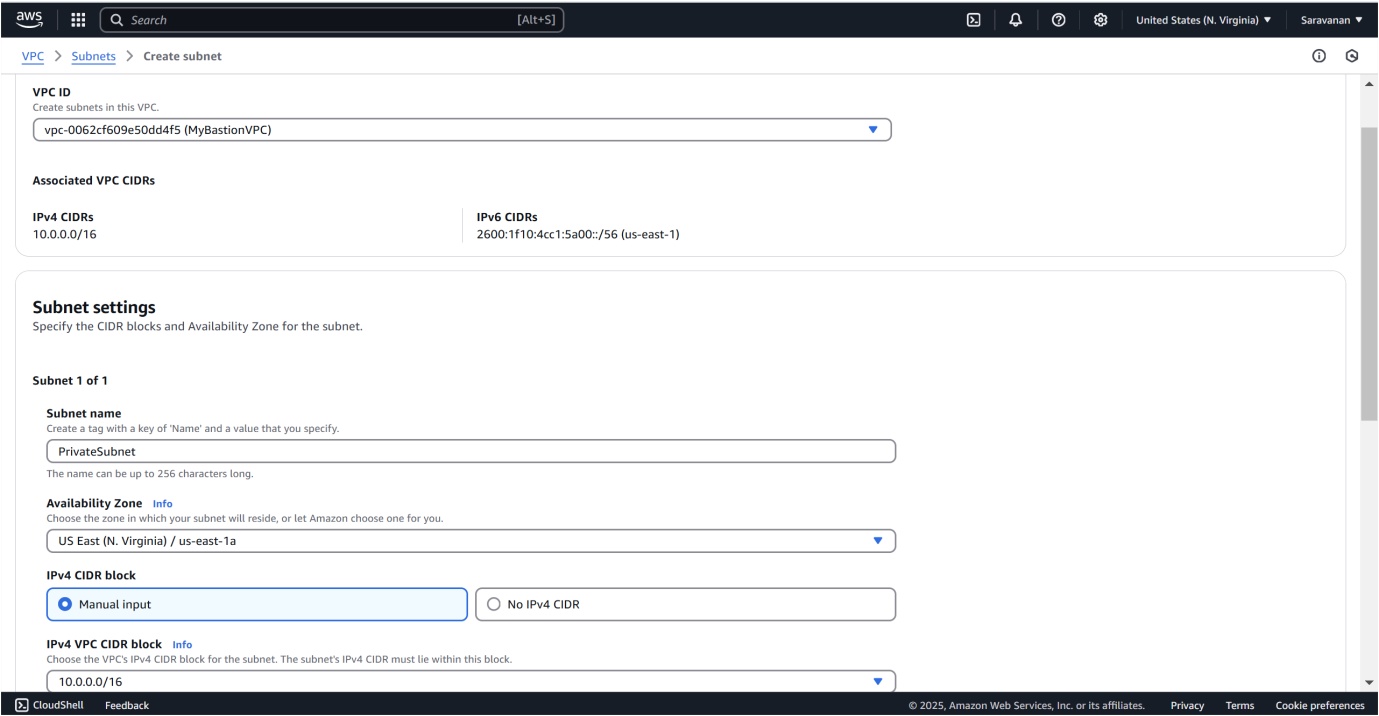


Step 5:

Select your **PublicSubnet** from the list, click **Actions → Modify auto-assign IP settings**, check **Enable auto-assign public IPv4 address**, and click **Save**.



Click **Create Subnet** again and fill in the details: select the same **VPC ID** (*MyBastionVPC*), set **Subnet Name** to *PrivateSubnet*, use the same **Availability Zone** as the public subnet (e.g., *us-east-1a*), and set the **IPv4 CIDR Block** to *10.0.2.0/24*. Leave **auto-assign public IP** disabled and click **Create Subnet**.

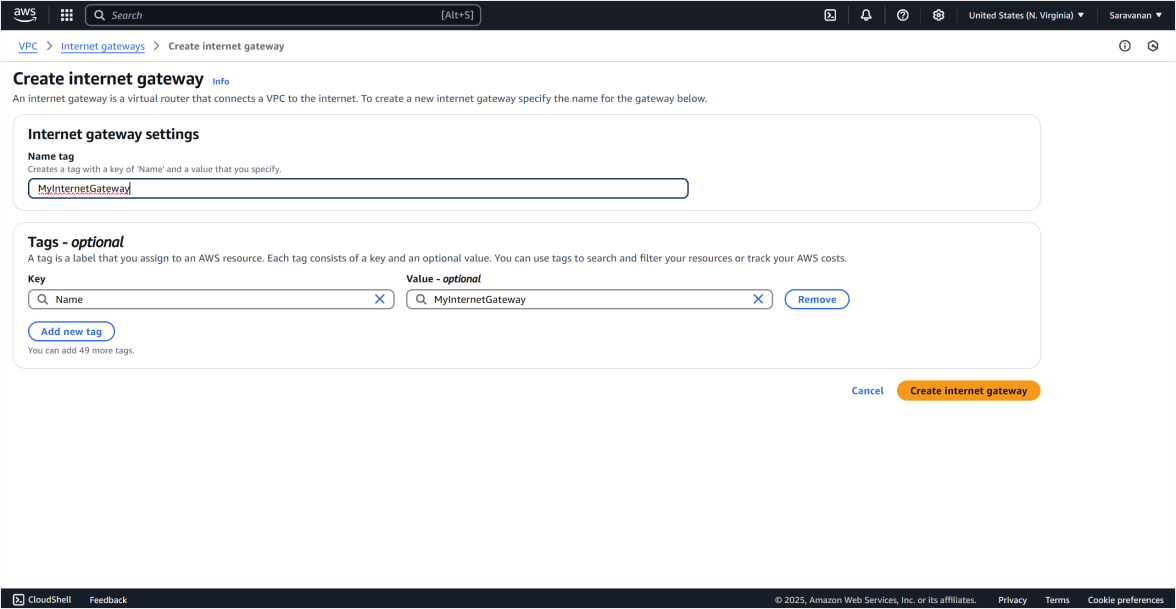


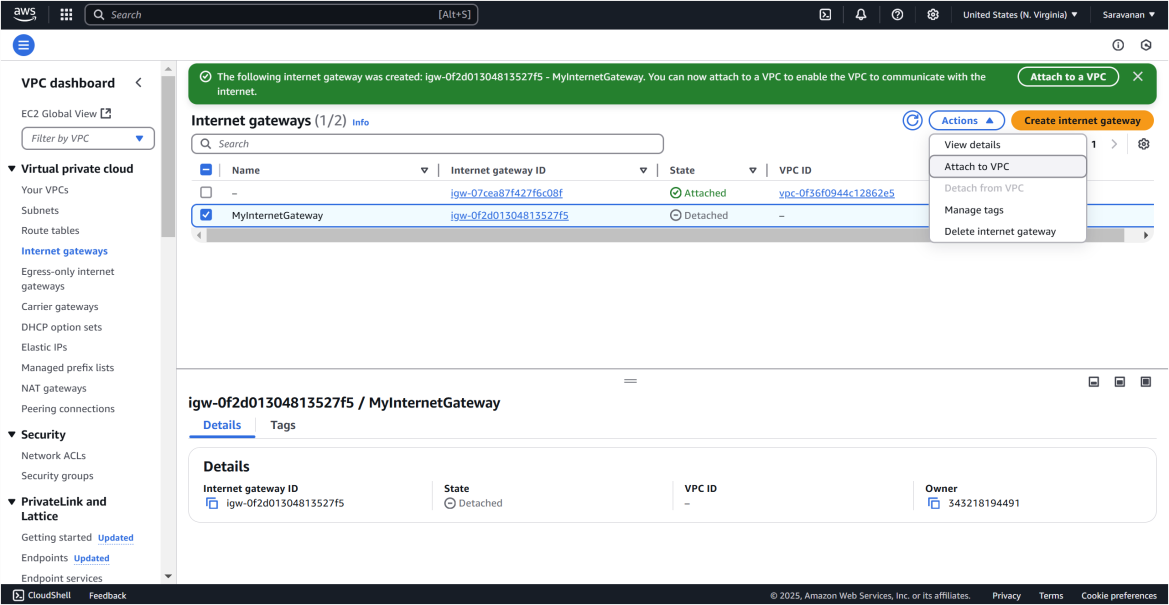
Step 7:

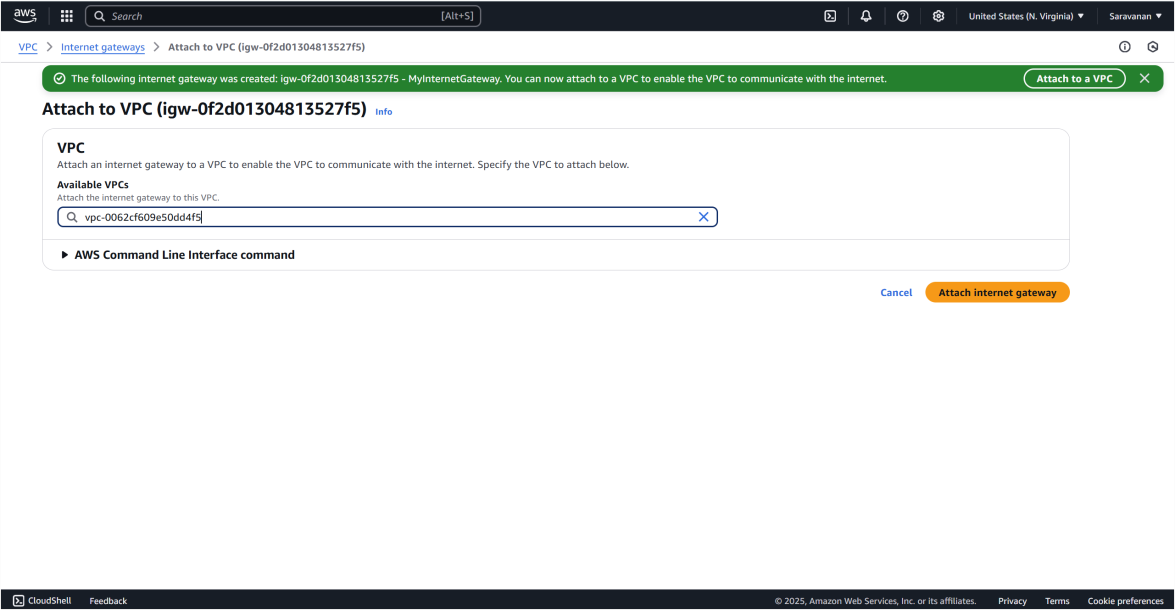
In the **VPC Dashboard**, go to **Internet Gateways** and click **Create**

**Internet Gateway**. Name it *MyInternetGateway* and click **Create**

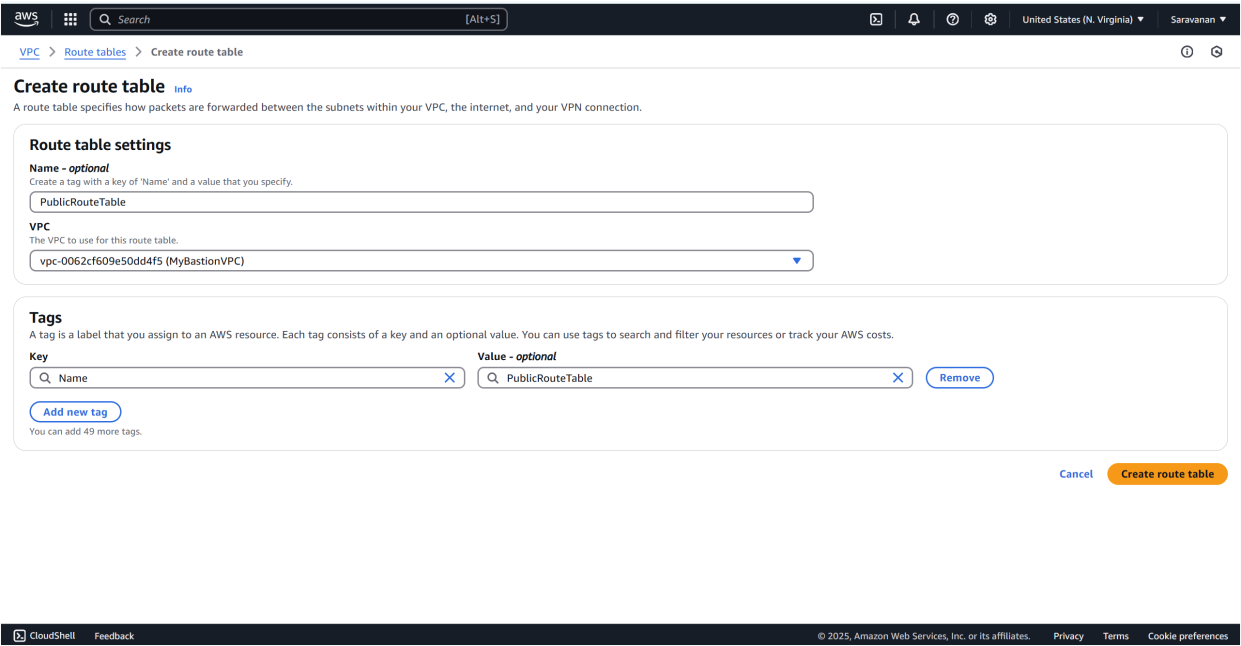
**Internet Gateway**. Select your new gateway, click **Actions → Attach to VPC**, choose your VPC (*MyBastionVPC*), and click **Attach Internet Gateway**.

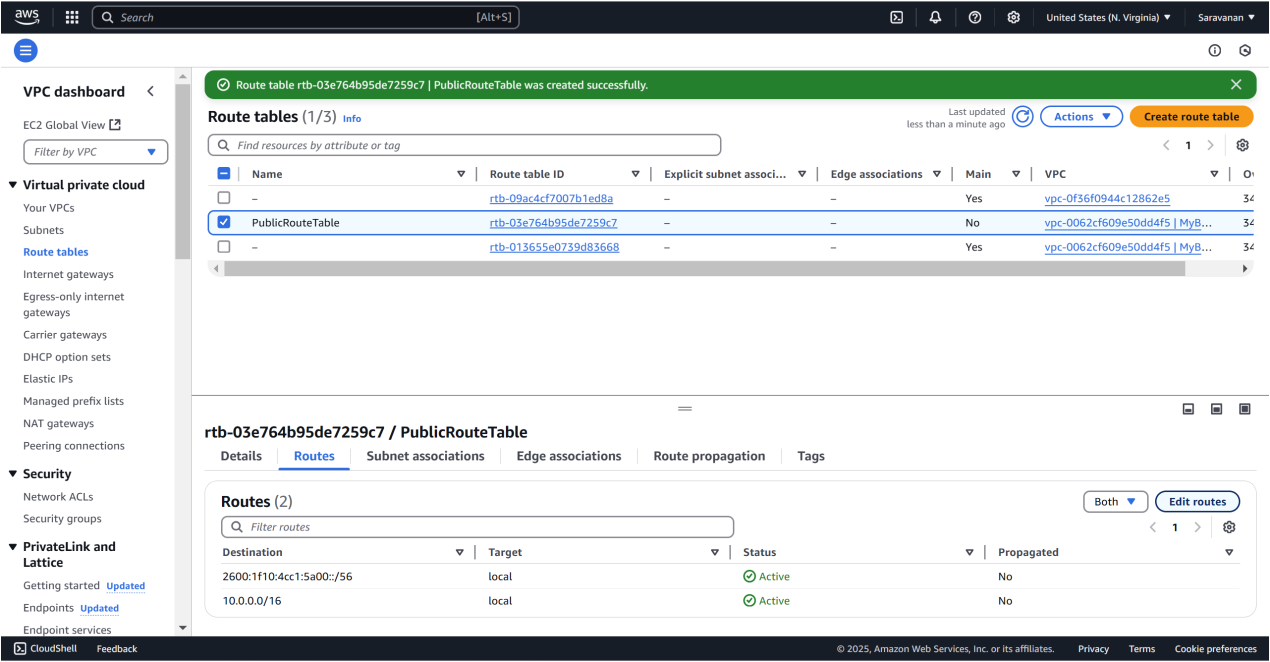




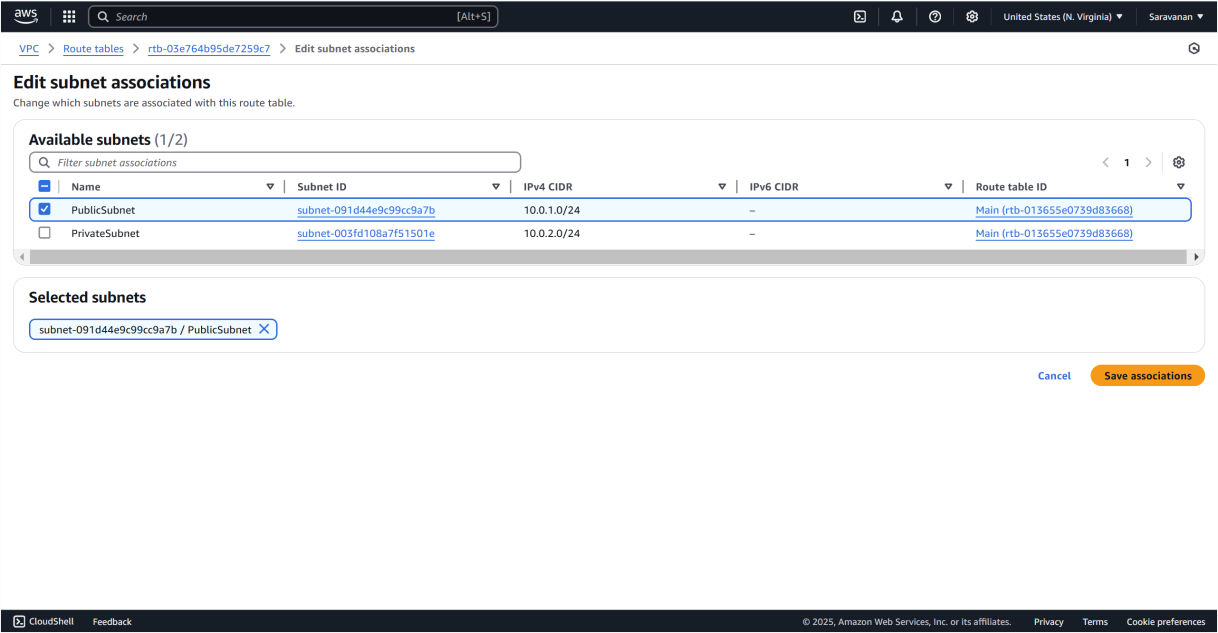


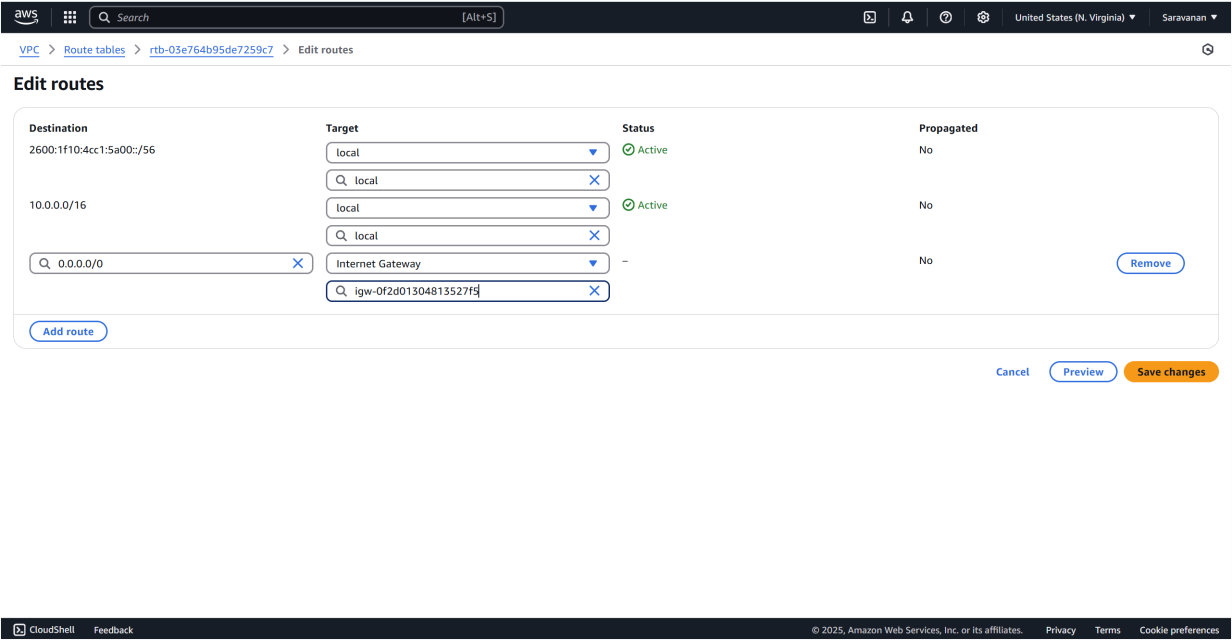
In the **VPC Dashboard**, go to **Route Tables** and click **Create Route Table**. Name it *PublicRouteTable*, select your VPC (*MyBastionVPC*), and click **Create Route Table**. Then, select *PublicRouteTable*, go to the **Routes** tab, click **Edit routes**, and add a route with **Destination** as *0.0.0.0/0* and **Target** as *MyInternetGateway*. Click **Save changes**.



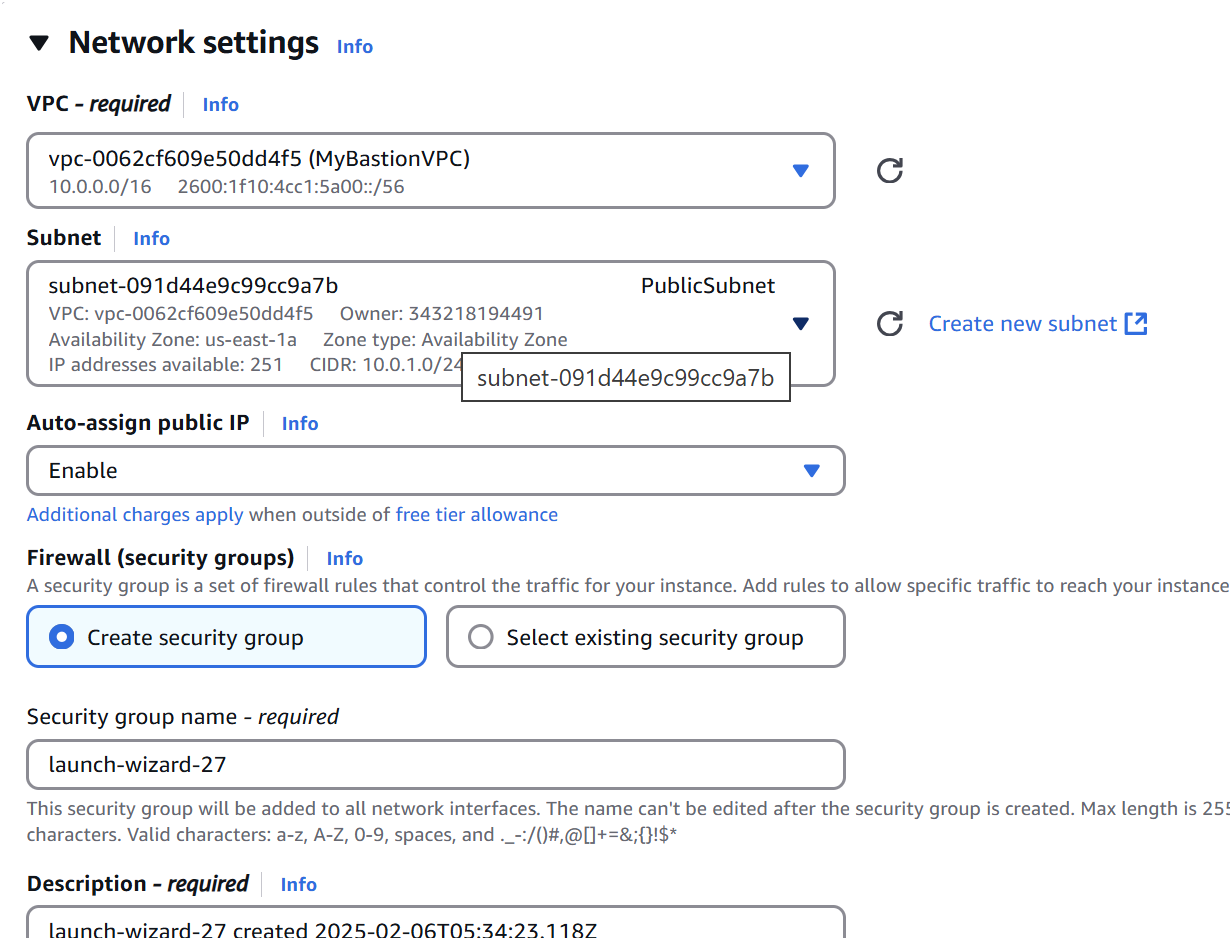


Next, go to the **Subnet associations** tab of *PublicRouteTable*, click **Edit subnet associations**, check the box for *PublicSubnet*, and click **Save associations**.



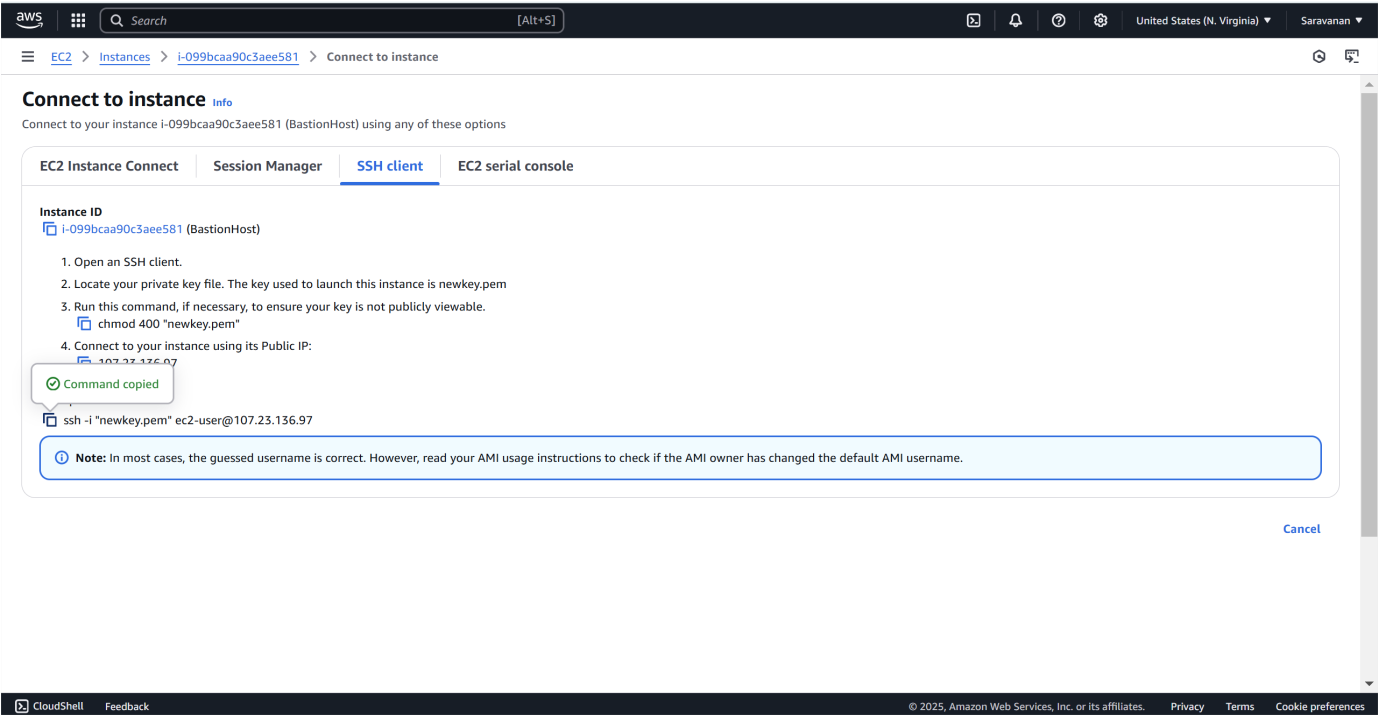


In the **EC2 Dashboard**, click **Launch Instance** and configure: set **Name** as *BastionHost*, select *Amazon Linux 2 AMI (HVM)* - Free Tier eligible, and choose **t2.micro** as the **Instance Type**. For **Key Pair**, create or select one, downloading the .pem file if creating. Under **Network Settings**, select *MyBastionVPC* for the **VPC**, *PublicSubnet* for the **Subnet**, and ensure **Auto-assign Public IP** is enabled. Create a **Security Group** to allow SSH (port 22) access, setting **Source** to *MyIP*. Use the default storage of 8 GiB, click **Launch Instance**, and wait for it to initialize.



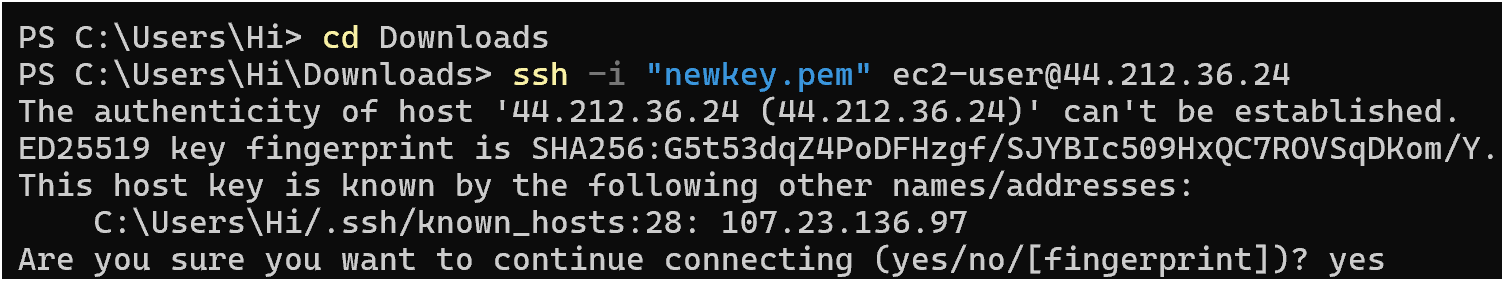
Step 10:

Connect with your PowerShell terminal by copying the ssh command in the SSH client of the *BastionHost(Ec2).*



Step 11:

Paste the command copied in the SSH client and connect it by using your key pair.



While connected to the bastion host, run this command to create a .ssh folder:



Step 13:

On your local machine, upload the key file to the bastion host

**scp -i /path/to/your-key.pem /path/to/your-key.pem ec2user@<BastionHost-Public-IP>:~/.ssh/**



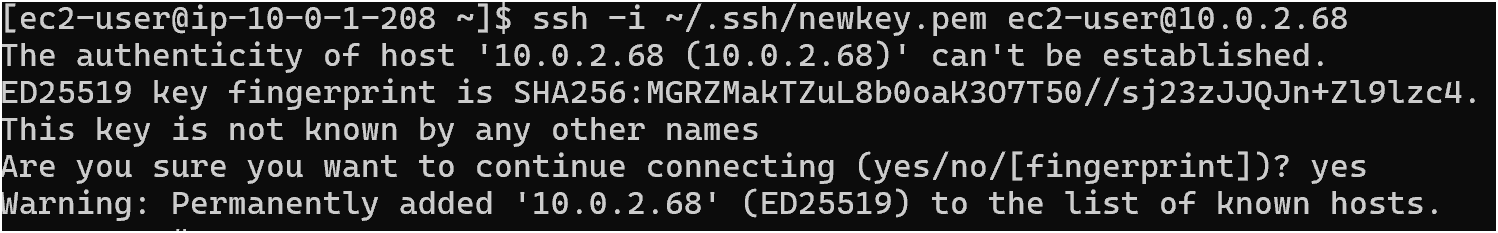
Step 14:

On the bastion host, run the following command to secure the key:



Step 15:

Use the private IP of the private instance (e.g., 10.0.2.x) and run: **ssh -i ~/.ssh/your-key.pem ec2-user@<PrivateInstance-PrivateIP>**



Step 16:

To verify network access and security, follow these steps:

1. **Check Internet Connectivity (Optional)**: If your private instance has internet access via a NAT gateway or instance, verify by running ping google.com. If there's no internet, it's fine as long as the private instance can communicate with the bastion host.
2. **Inspect Instance Details**: Connect to your private instance and run:

ohostname to check the instance hostname. oifconfig to verify the private IP address.

