AYUSHI VASHISTH
SRM UNIVERSITY,DELHI NCR
11721210005

Task 1: AWS VPC Setup with Bastion Host and NAT Gateway

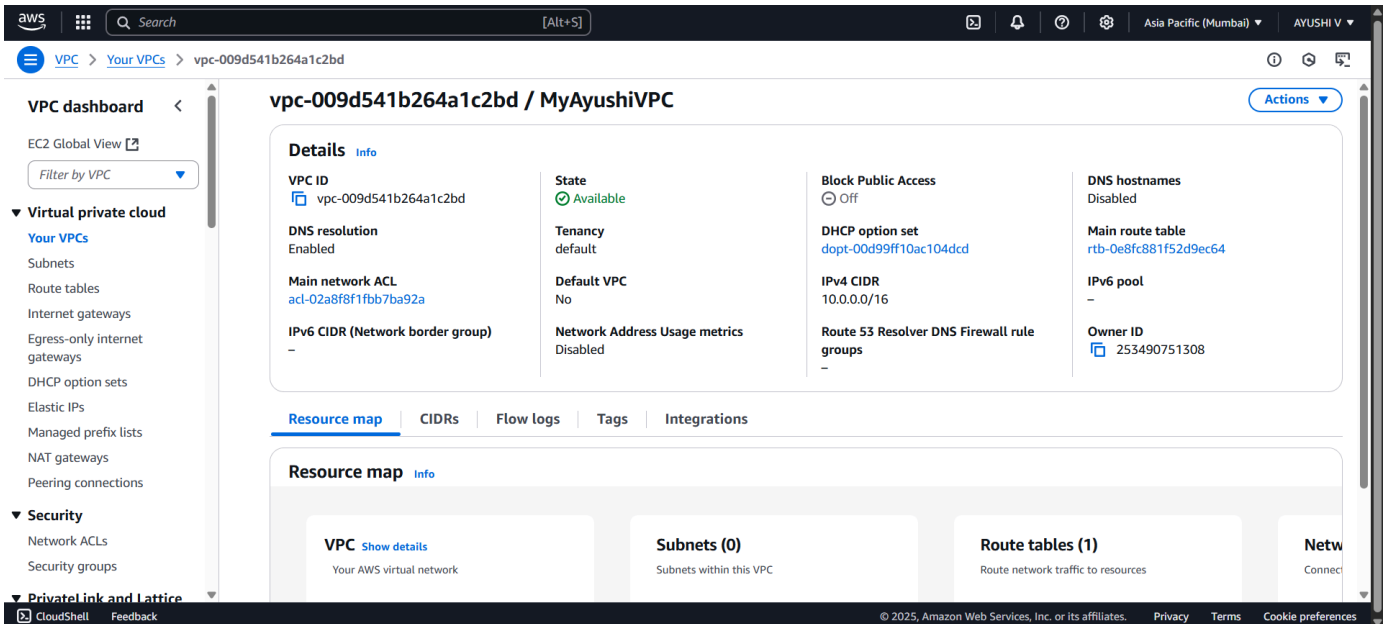
1. Introduction

In this task, I created a secure and logically separated AWS environment using a custom VPC with both public and private subnets. The public subnet hosts a Bastion host and a NAT Gateway, while the private subnet hosts the backend EC2 instance. The NAT Gateway allows the private instance to access the internet securely, and the Bastion host enables secure SSH access to the private instance.

2. Step-by-Step Implementation

Step 1: Create VPC

I started by creating a custom VPC with a CIDR block of 10.0.0.0/16. During creation, I made sure to enable DNS hostnames, which are necessary for the NAT Gateway and other AWS services to work properly inside the VPC.

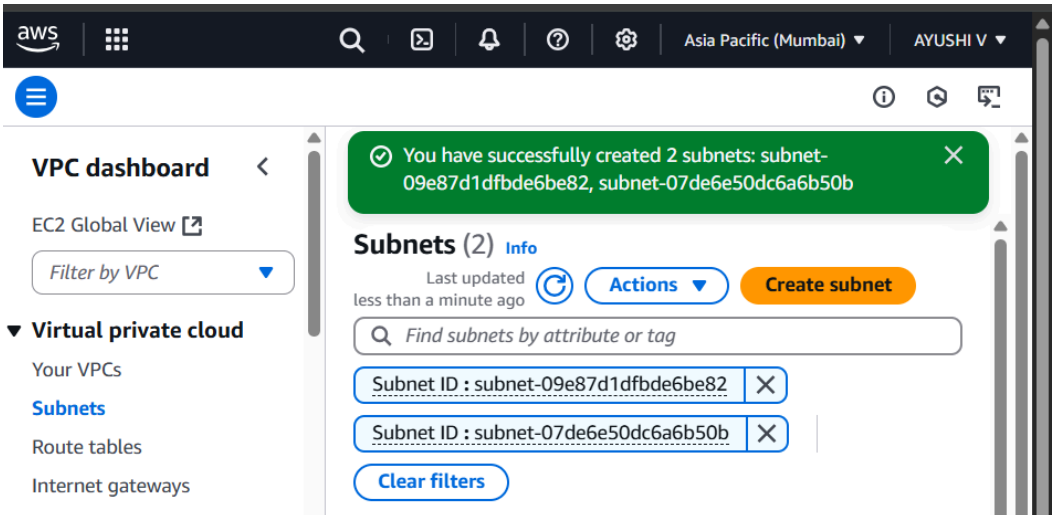


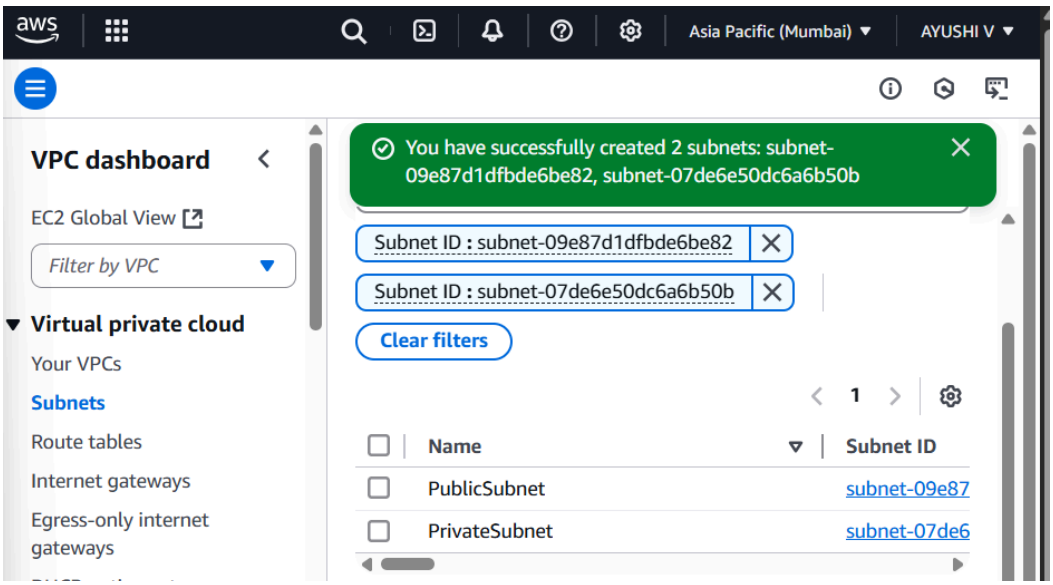
AYUSHI VASHISTH
SRM UNIVERSITY
11721210005

Step 2: Create Subnets

Next, I created two subnets within the VPC:

- A public subnet with CIDR 10.0.1.0/24 where the Bastion host and NAT Gateway will reside.
- A private subnet with CIDR 10.0.2.0/24 to host the backend EC2 instance securely away from direct internet access.

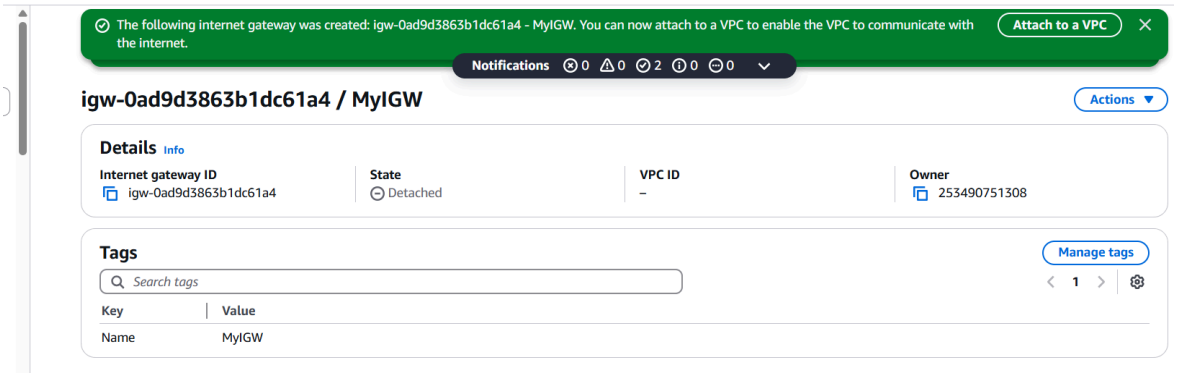


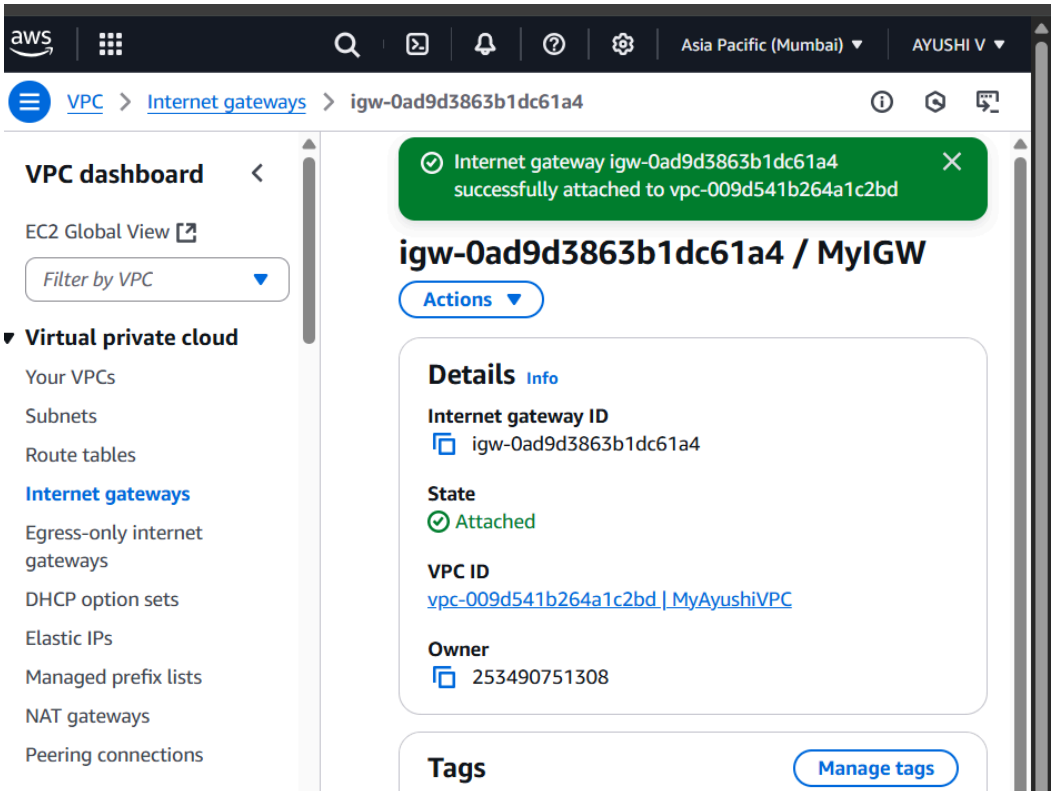


AYUSHI VASHISTH
SRM UNIVERSITY
11721210005

Step 3: Create and Attach Internet Gateway (IGW)

I created an Internet Gateway (IGW) and attached it to my VPC. This IGW provides internet access to resources in the public subnet.

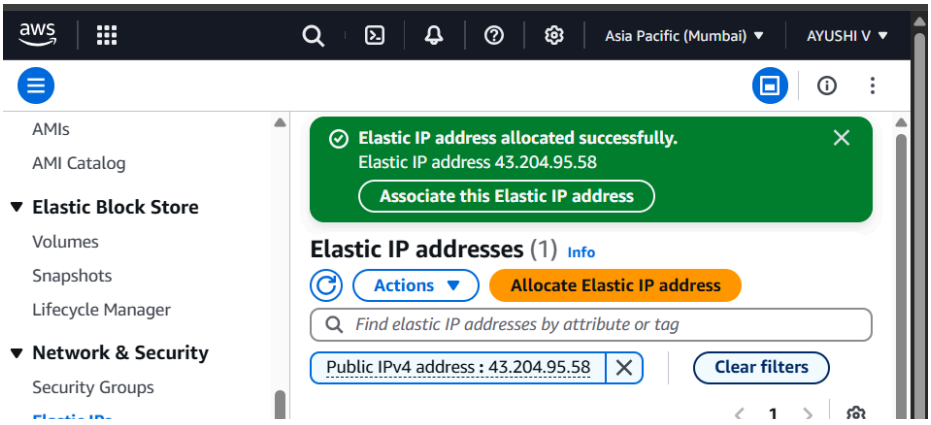


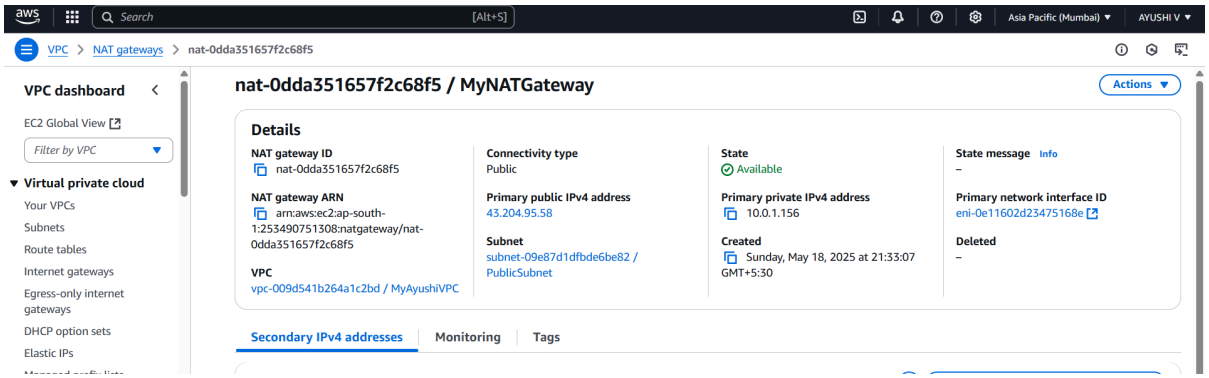


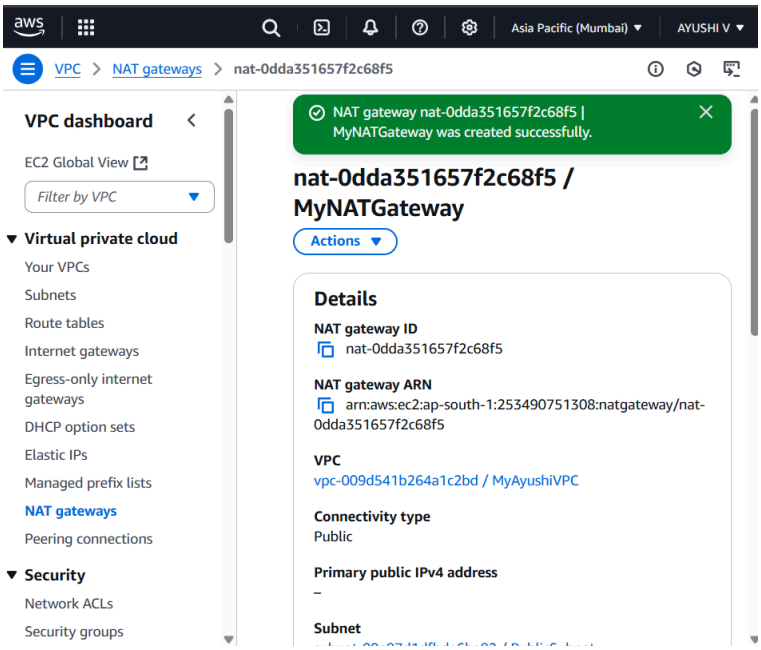
AYUSHI VASHISTH
SRM UNIVERSITY
11721210005

Step 4: Create NAT Gateway

I allocated an Elastic IP address and then created a NAT Gateway in the public subnet. This NAT Gateway uses the Elastic IP and allows instances in the private subnet to access the internet securely without exposing them directly.





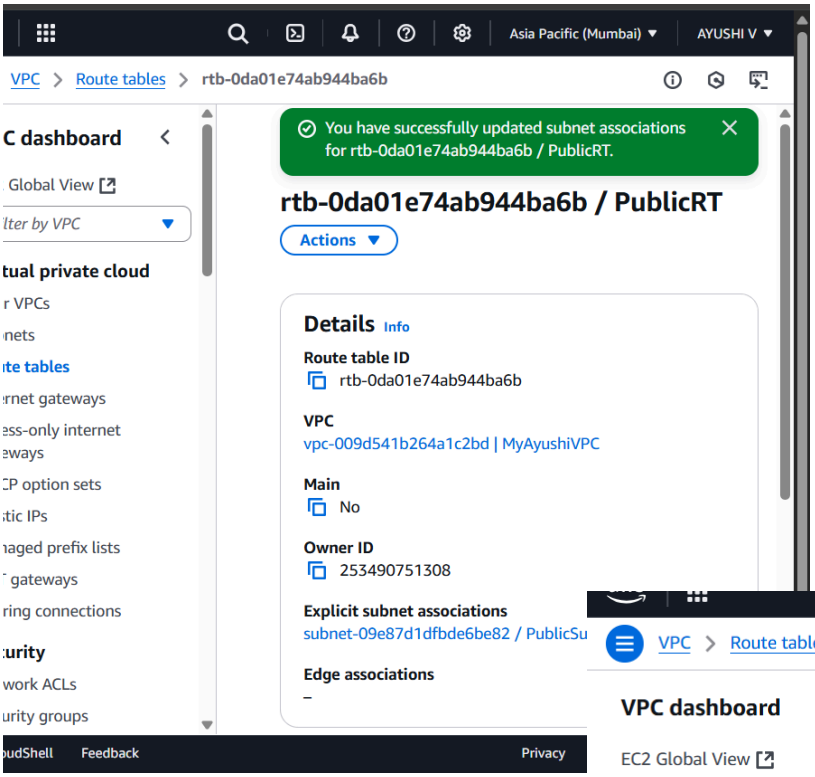


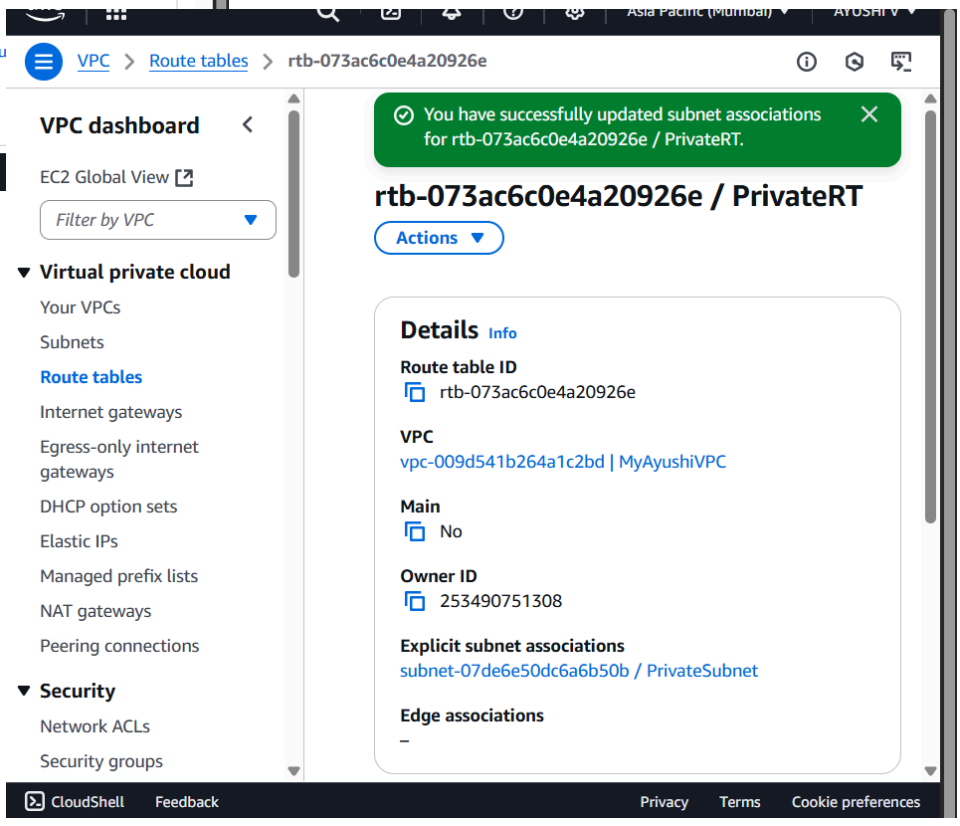
AYUSHI VASHISTH
SRM UNIVERSITY
11721210005

Step 5: Configure Route Tables

To control traffic flow, I created two route tables:

- Public Route Table: Associated with the public subnet. I added a route that sends all internet-bound traffic (0.0.0.0/0) to the Internet Gateway (IGW).
- Private Route Table: Associated with the private subnet. I added a route for all internet-bound traffic (0.0.0.0/0) to the NAT Gateway.



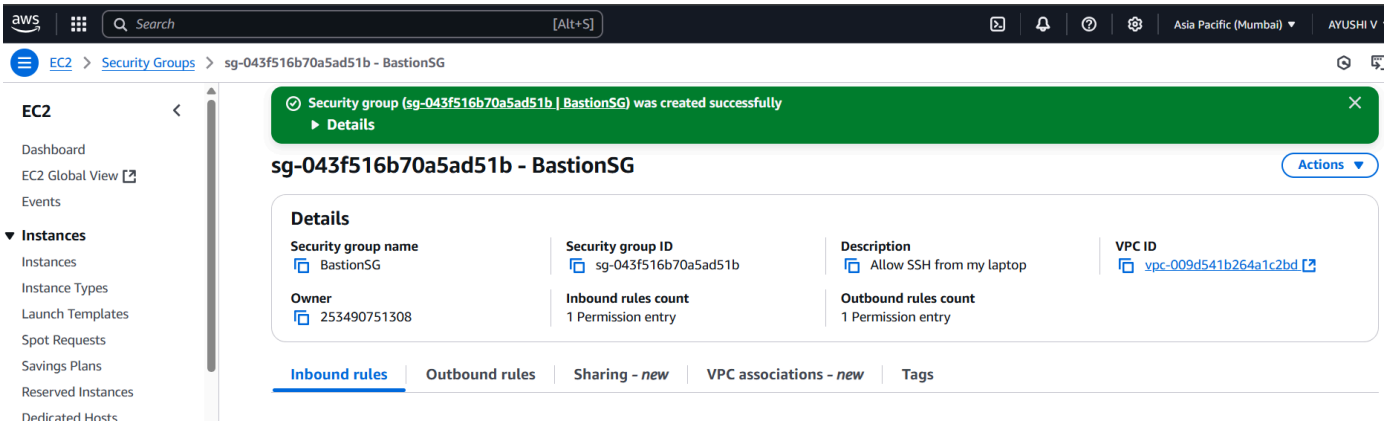


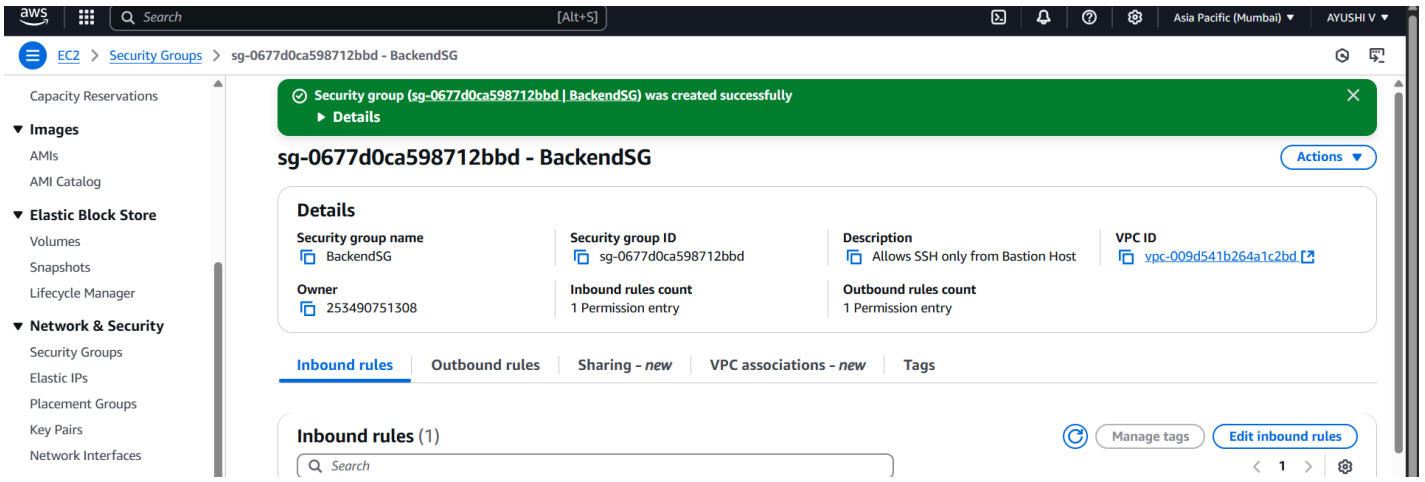
AYUSHI VASHISTH
SRM UNIVERSITY
11721210005

Step 6: Create Security Groups

I created two security groups to control inbound and outbound traffic securely:

- Bastion Host Security Group: Allowed inbound SSH access (port 22) only from my IP address to restrict access. Outbound traffic is open.
- Backend EC2 Security Group: Allowed inbound SSH access (port 22) only from the Bastion host's private subnet IP range (10.0.1.0/24). Outbound traffic is open.



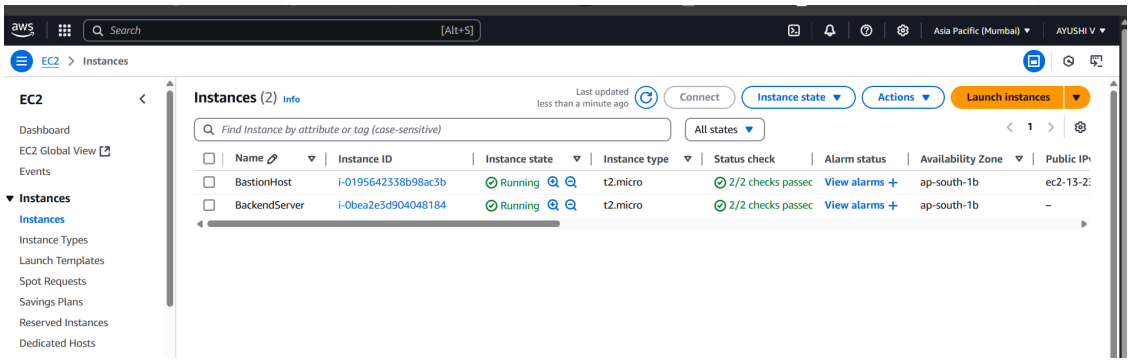


AYUSHI VASHISTH
SRM UNIVERSITY
11721210005

Step 7: Launch EC2 Instances

I launched two EC2 instances:

- Bastion Host: Placed in the public subnet with the Bastion security group and my key pair. This instance has a public IP to enable SSH from my local machine.
- Backend Instance: Launched in the private subnet with the backend security group and the same key pair. This instance does not have a public IP, so it can only be accessed via the Bastion host.



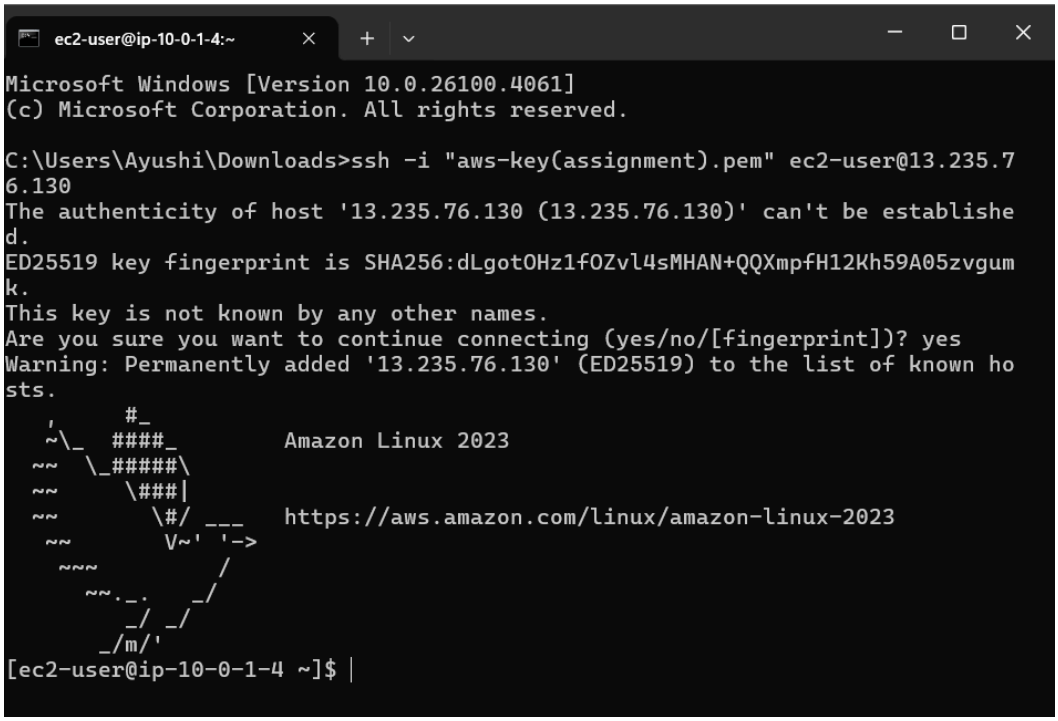
Step 8: Validate Connections

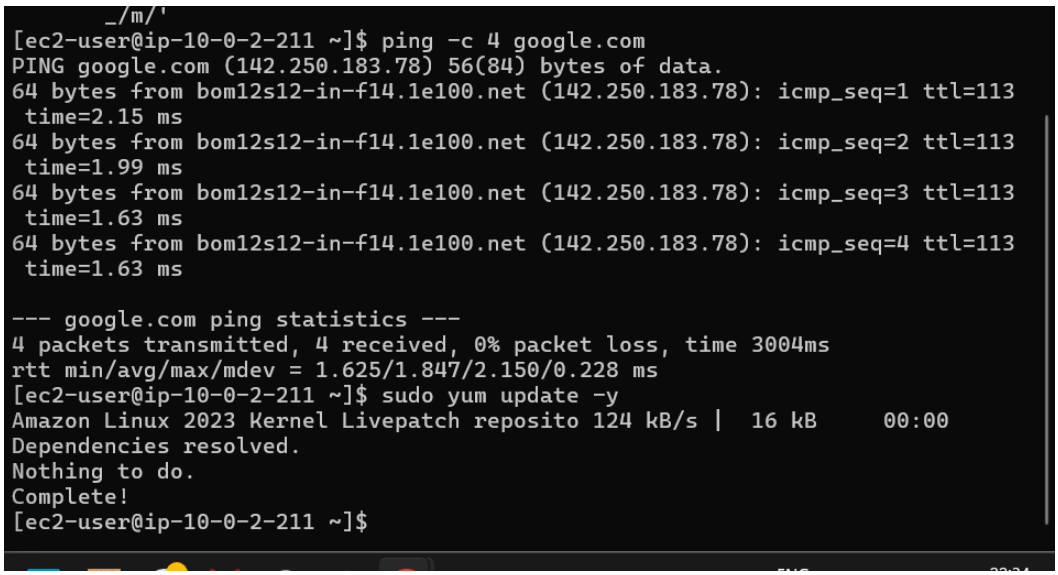
Finally, I validated the setup by:

1. SSH'ing from my local machine to the Bastion host using the Bastion's public IP.
2. From the Bastion host, SSH'ing into the backend EC2 instance using its private IP.
3. From the backend EC2 instance, verifying internet connectivity by pinging an external site (ping google.com) and updating packages with sudo yum update -y.

All these steps worked successfully, confirming that the Bastion host and NAT Gateway function as expected.

AYUSHI VASHISTH
SRM UNIVERSITY
11721210005





Conclusion

This task successfully demonstrated how to set up a secure AWS environment using a custom VPC with public and private subnets. The Bastion host enabled secure access to the backend instance in the private subnet, and the NAT Gateway allowed the private instance to reach the internet safely. This architecture is a foundational best practice for secure cloud deployments.