

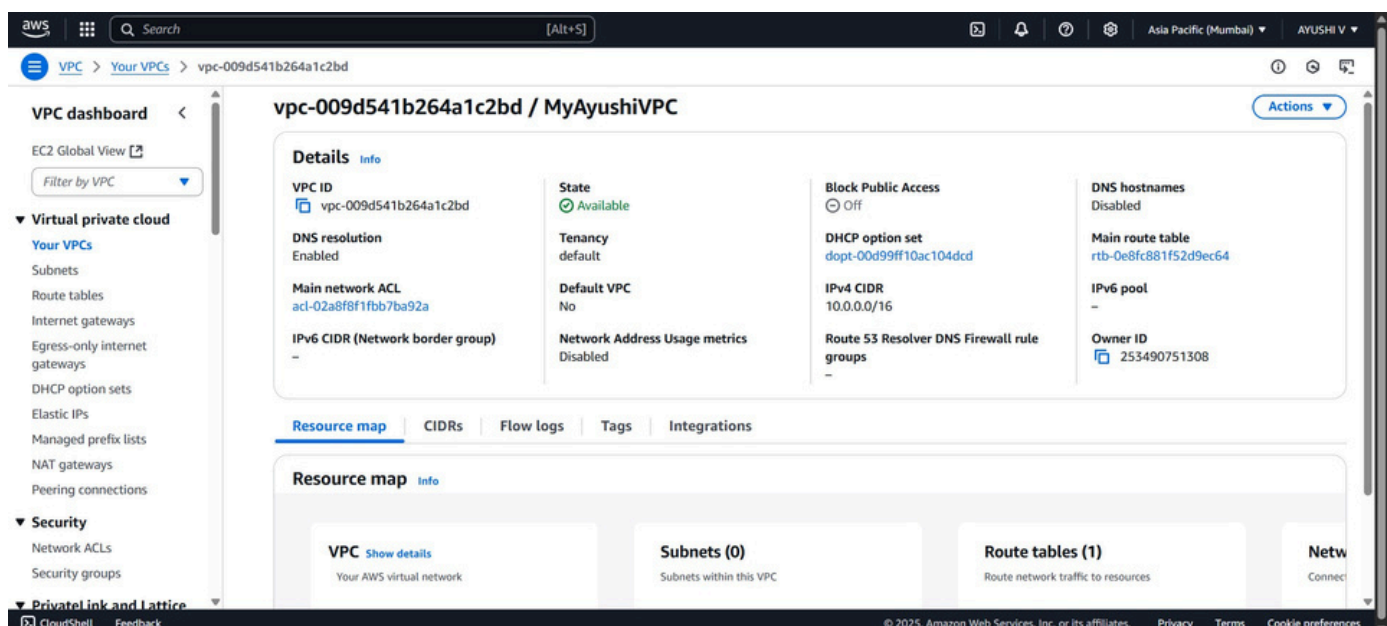
## Task 1: AWS VPC Setup with Bastion Host and NAT Gateway

**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

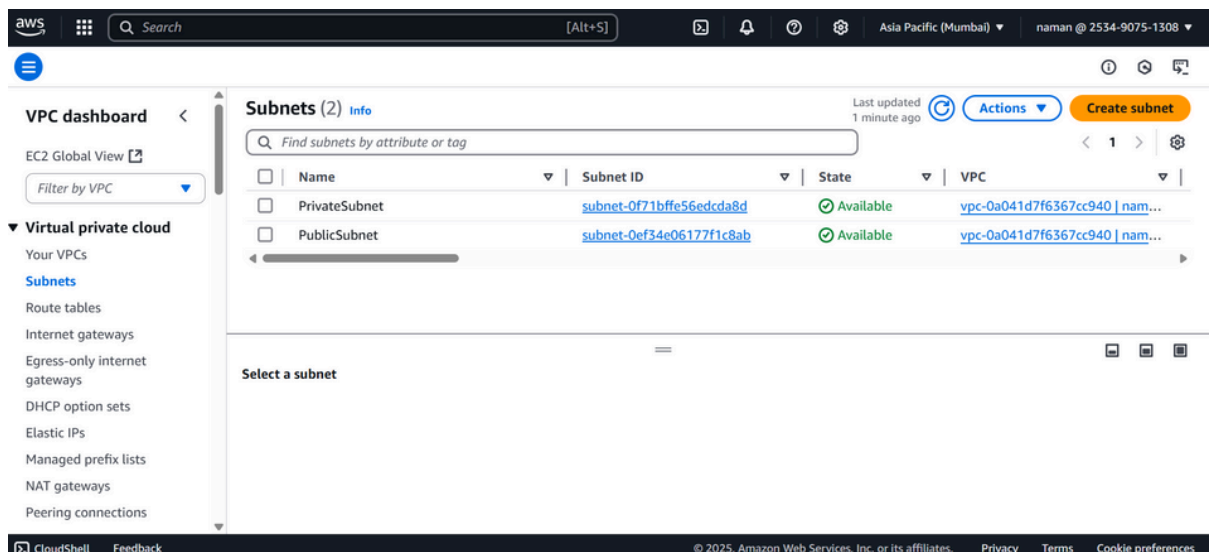
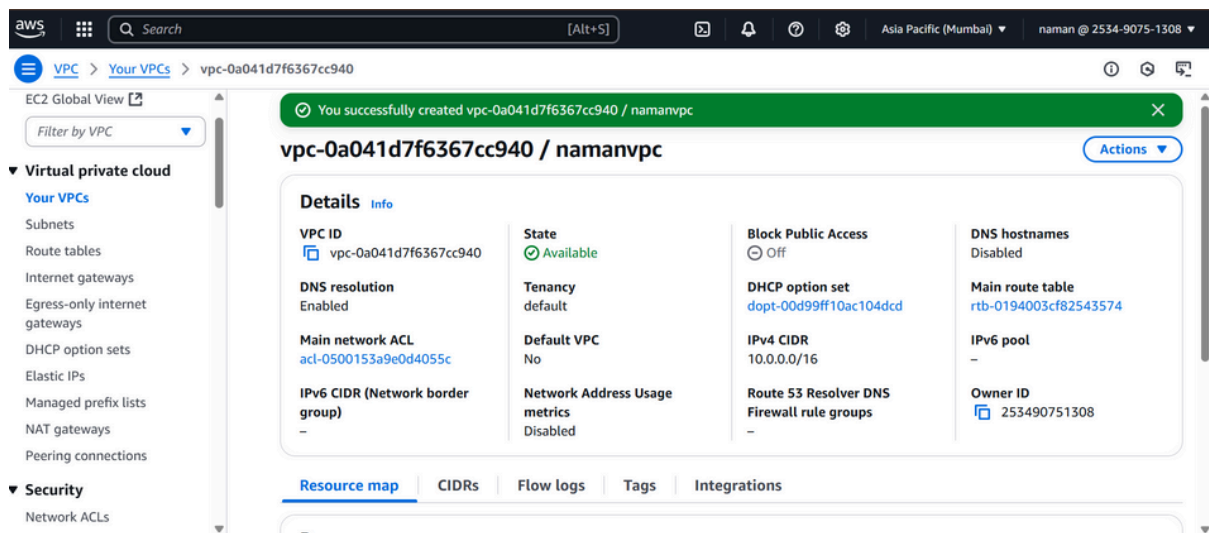
I started by creating a custom VPC with a CIDR block of 10.0.0.0/16. During creation, I made sure to enable DNS hostnames, which are necessary for the NAT Gateway and other AWS services to work properly inside the VPC.



## 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.



### 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.

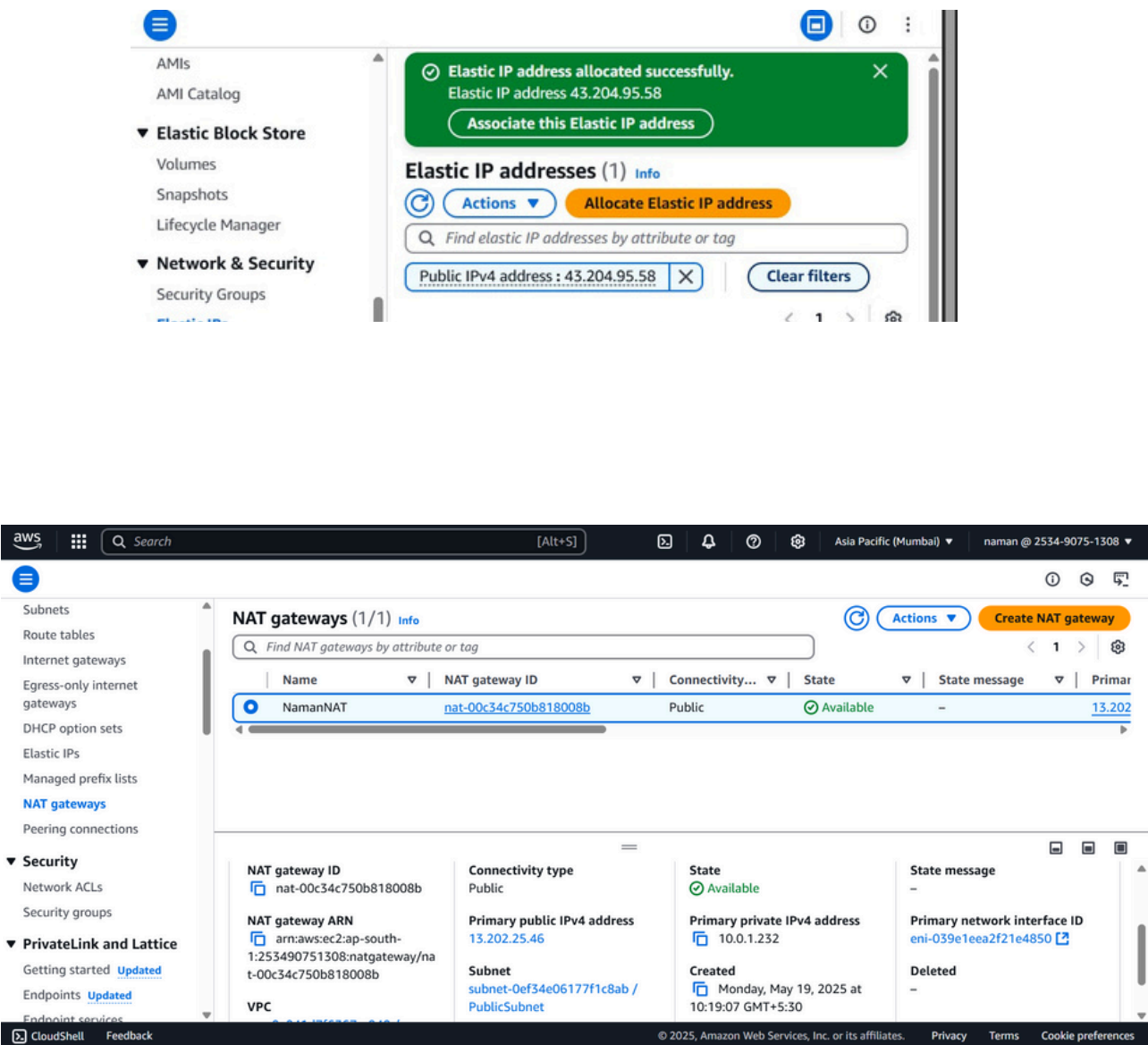
The screenshot shows the AWS Management Console interface for the 'Internet gateways' section. The left sidebar contains the 'VPC dashboard' and a list of VPC-related resources, with 'Internet gateways' selected. The main content area shows a table of internet gateways. One gateway, 'NamanIGW', is listed with ID 'igw-08d518f58fa8e5742' and is in an 'Attached' state, connected to VPC 'vpc-0a041d7f6367cc940'. Below the table, the 'Details' tab is active, displaying the gateway's ID, state, VPC ID, and owner information.

Name	Internet gateway ID	State	VPC ID
NamanIGW	igw-08d518f58fa8e5742	Attached	vpc-0a041d7f6367cc940   namanvpc

Details			
Internet gateway ID	State	VPC ID	Owner
igw-08d518f58fa8e5742	Attached	vpc-0a041d7f6367cc940   namanvpc	253490751308

**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.



## 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.

The screenshot shows the AWS Management Console interface for the 'Route tables' section. The left sidebar contains the 'VPC dashboard' and 'Virtual private cloud' navigation menu. The main content area displays a list of route tables under the heading 'Route tables (1/4) info'. The table lists four route tables: a default route table, 'NamanRT', 'PrivateNamanRT', and another default route table. The 'NamanRT' route table is selected, and its details are shown in the 'Details' section below. The details include the Route table ID (rtb-0f82e750de5dd19ea), VPC (vpc-0a041d7f6367cc940), Main status (No), Owner ID (253490751308), and Explicit subnet associations (subnet-0ef34e06177f1c8ab / PublicSubnet).

Name	Route table ID	Explicit subnet associ...	Edge associations	Main
-	rtb-0194003cf82543574	-	-	Yes
NamanRT	rtb-0f82e750de5dd19ea	subnet-0ef34e06177f1c8...	-	No
PrivateNamanRT	rtb-0db5c9aae68ab30b2	subnet-0f71bffe56edcda...	-	No
-	rtb-087d126259c09f2a6	-	-	Yes

**Details**

Route table ID: [rtb-0f82e750de5dd19ea](#)

VPC: [vpc-0a041d7f6367cc940](#) | [namanvpc](#)

Main: ☐ No

Owner ID: [253490751308](#)

Explicit subnet associations: [subnet-0ef34e06177f1c8ab](#) / [PublicSubnet](#)

Edge associations: -

The screenshot shows the AWS Management Console interface for the 'Route tables' section. The left sidebar contains the 'VPC dashboard' and 'Virtual private cloud' navigation menu. The main content area displays a list of route tables under the heading 'Route tables (1/4) info'. The table lists four route tables: a default route table, 'NamanRT', 'PrivateNamanRT', and another default route table. The 'PrivateNamanRT' route table is selected, and its details are shown in the 'Details' section below. The details include the Route table ID (rtb-0db5c9aae68ab30b2), VPC (vpc-0a041d7f6367cc940), Main status (No), Owner ID (253490751308), and Explicit subnet associations (subnet-0f71bffe56edcda8d / PrivateSubnet).

Name	Route table ID	Explicit subnet associ...	Edge associations	Main
-	rtb-0194003cf82543574	-	-	Yes
NamanRT	rtb-0f82e750de5dd19ea	subnet-0ef34e06177f1c8...	-	No
PrivateNamanRT	rtb-0db5c9aae68ab30b2	subnet-0f71bffe56edcda...	-	No
-	rtb-087d126259c09f2a6	-	-	Yes

**Details**

Route table ID: [rtb-0db5c9aae68ab30b2](#)

VPC: [vpc-0a041d7f6367cc940](#) | [namanvpc](#)

Main: ☐ No

Owner ID: [253490751308](#)

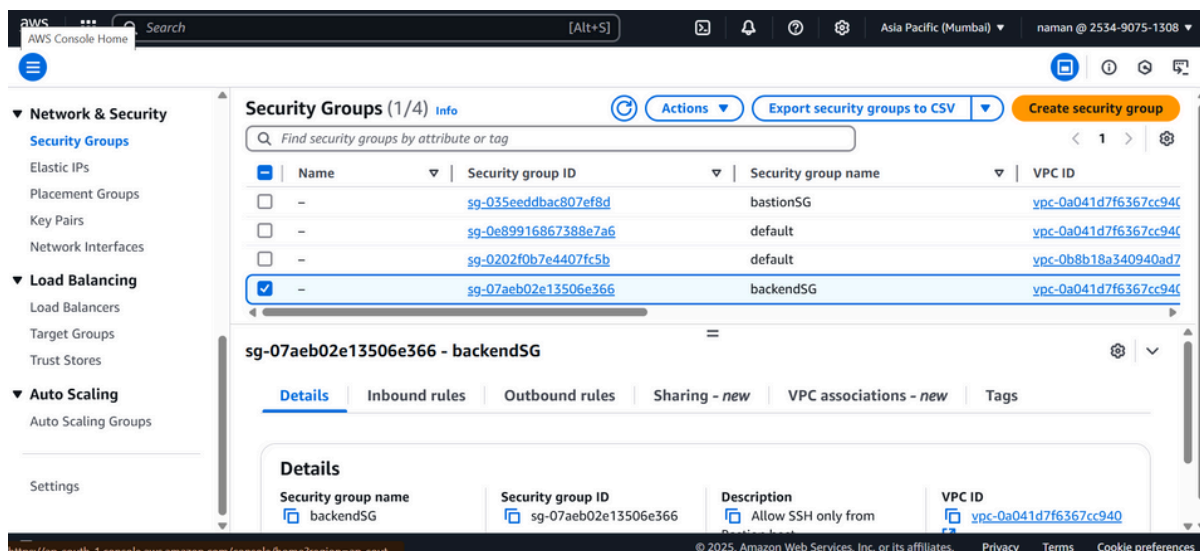
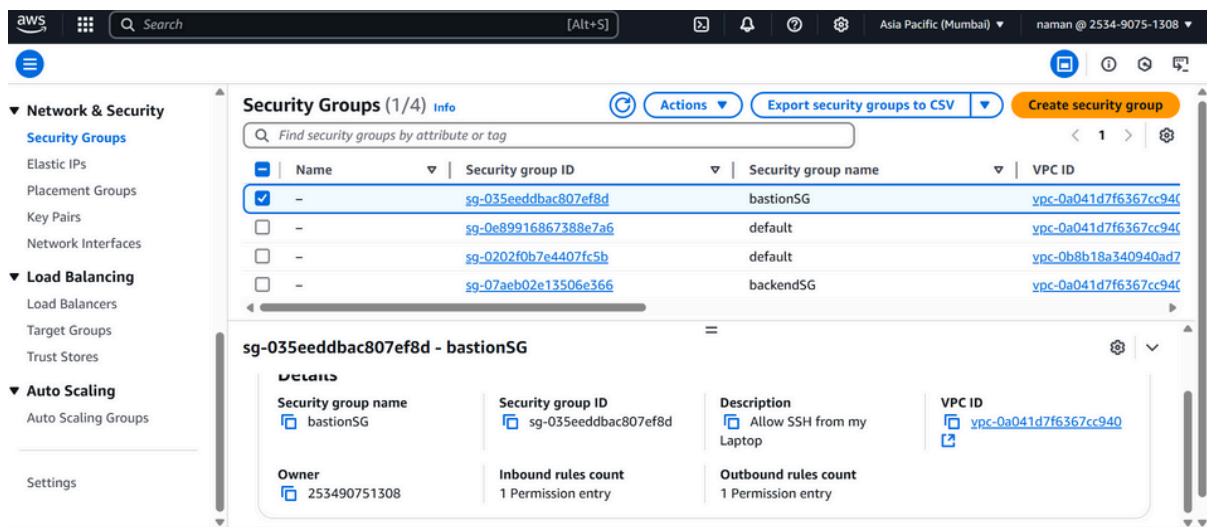
Explicit subnet associations: [subnet-0f71bffe56edcda8d](#) / [PrivateSubnet](#)

Edge associations: -

## Step 6: Create Security Groups

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

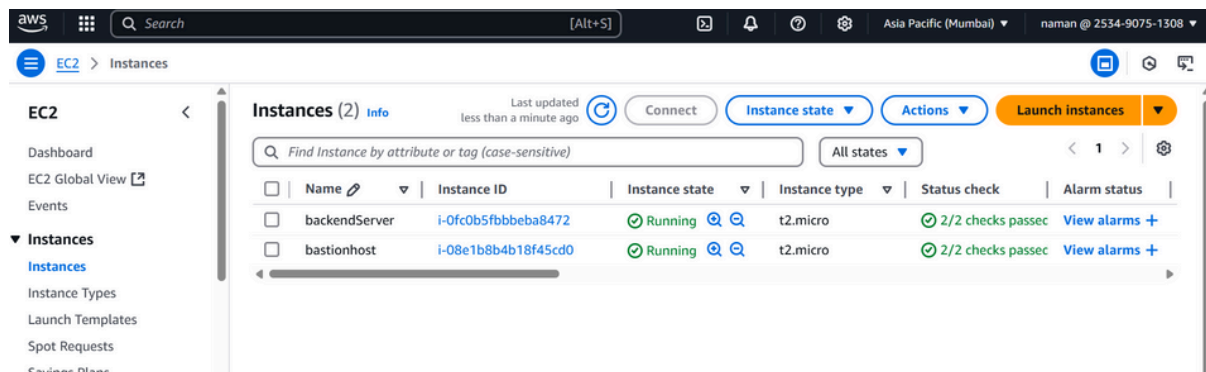
- Bastion Host Security Group: Allowed inbound SSH access (port 22) only from my IP address to restrict access. Outbound traffic is open. Backend
- EC2 Security Group: Allowed inbound SSH access (port 22) only from the Bastion host's private subnet IP range (10.0.1.0/24). Outbound traffic is open.



## 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.



```
PS C:\Users\91783> scp -i "C:\Users\91783\Downloads\namanaws-key.pem" namanaws-key.pem ec2-user@13.201.10.137
The system cannot find the file specified.
PS C:\Users\91783> scp -i "C:\Users\91783\Downloads\namanaws-key.pem" ec2-user@13.201.10.137
usage: scp [-346ABCOpqRrsTv] [-c cipher] [-D sftp_server_path] [-F ssh_config]
          [-i identity_file] [-J destination] [-l limit] [-o ssh_option]
          [-P port] [-S program] [-X sftp_option] source ... target
PS C:\Users\91783> ssh -i "C:\Users\91783\Downloads\namanaws-key.pem" ec2-user@13.201.10.137
Last login: Mon May 19 05:52:35 2025 from 116.204.190.86

      #_
    _~\  #####_      Amazon Linux 2
   ~~~ \_#####\
   ~~~  \###|      AL2 End of Life is 2026-06-30.
   ~~~   \#/  ---
   ~~~    V~'  '--->

      nnn      /
      nn._..  /
      _/_/_/_/
      _/m/'      A newer version of Amazon Linux is available!

                  Amazon Linux 2023, GA and supported until 2028-03-15.
                  https://aws.amazon.com/linux/amazon-linux-2023/
```

**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.