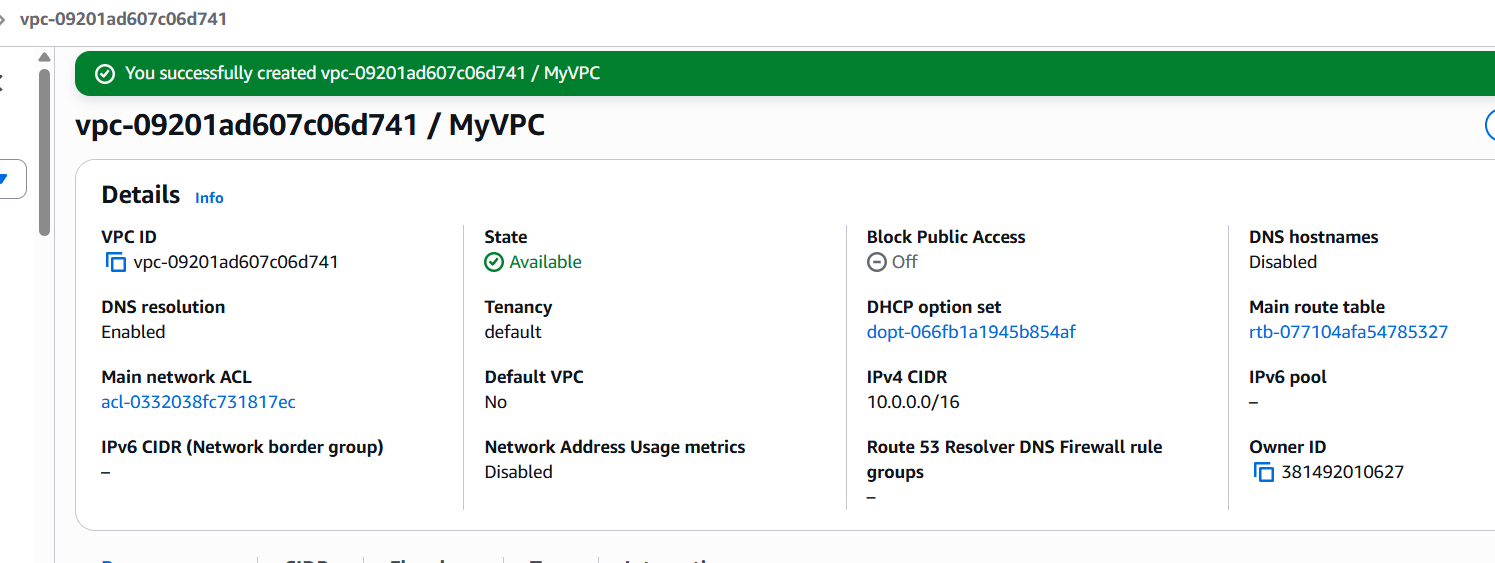
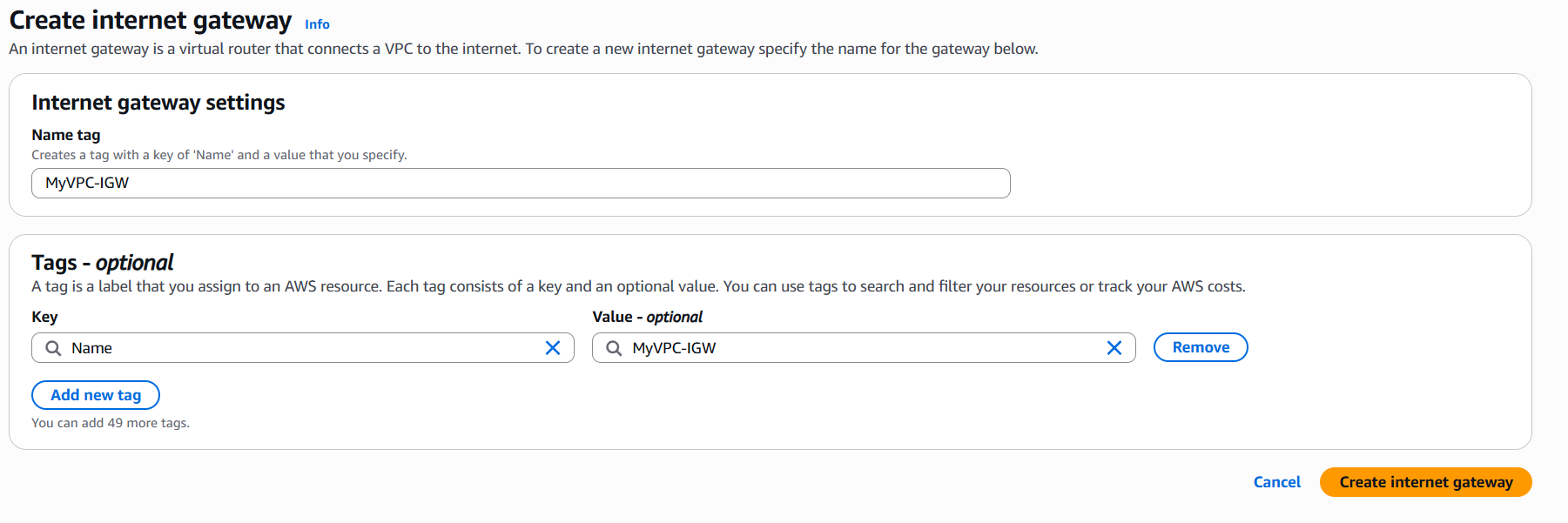
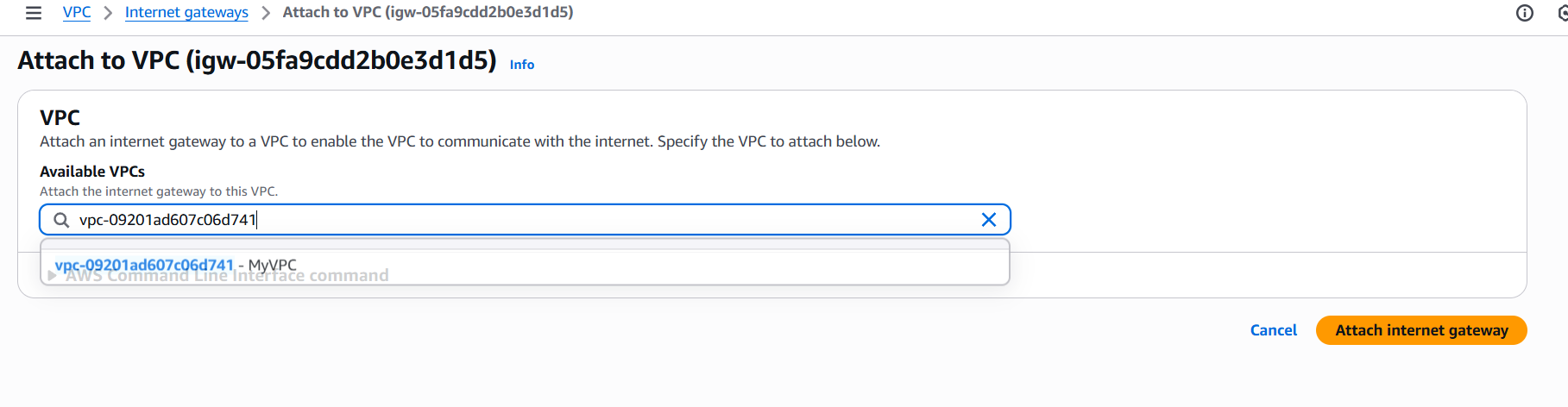
1. Go to **VPC Console** → **Your VPCs**
2. Click **Create VPC**
3. **Settings:**
   * Name: MyVPC
   * IPv4 CIDR: 10.0.0.0/16
   * IPv6 CIDR: No IPv6 CIDR block
   * Tenancy: Default
4. Click **Create VPC**



**Step 3: Create Internet Gateway**

1. Go to **Internet Gateways**
2. Click **Create internet gateway**
3. Name: MyVPC-IGW
4. Click **Create internet gateway**
5. Attach to VPC:
   * Select the IGW → **Actions** → **Attach to VPC**
   * Choose MyVPC
   * Click **Attach internet gateway**

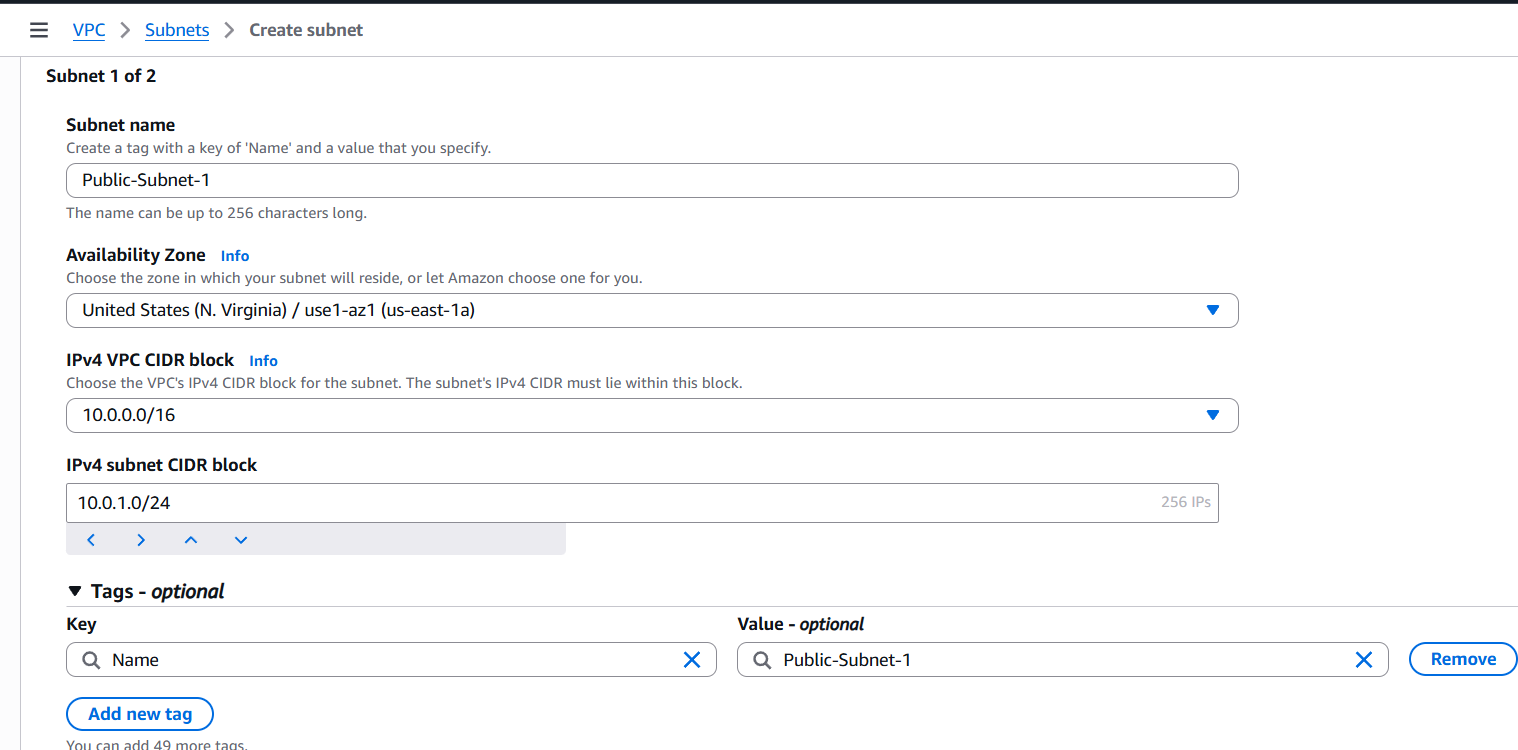


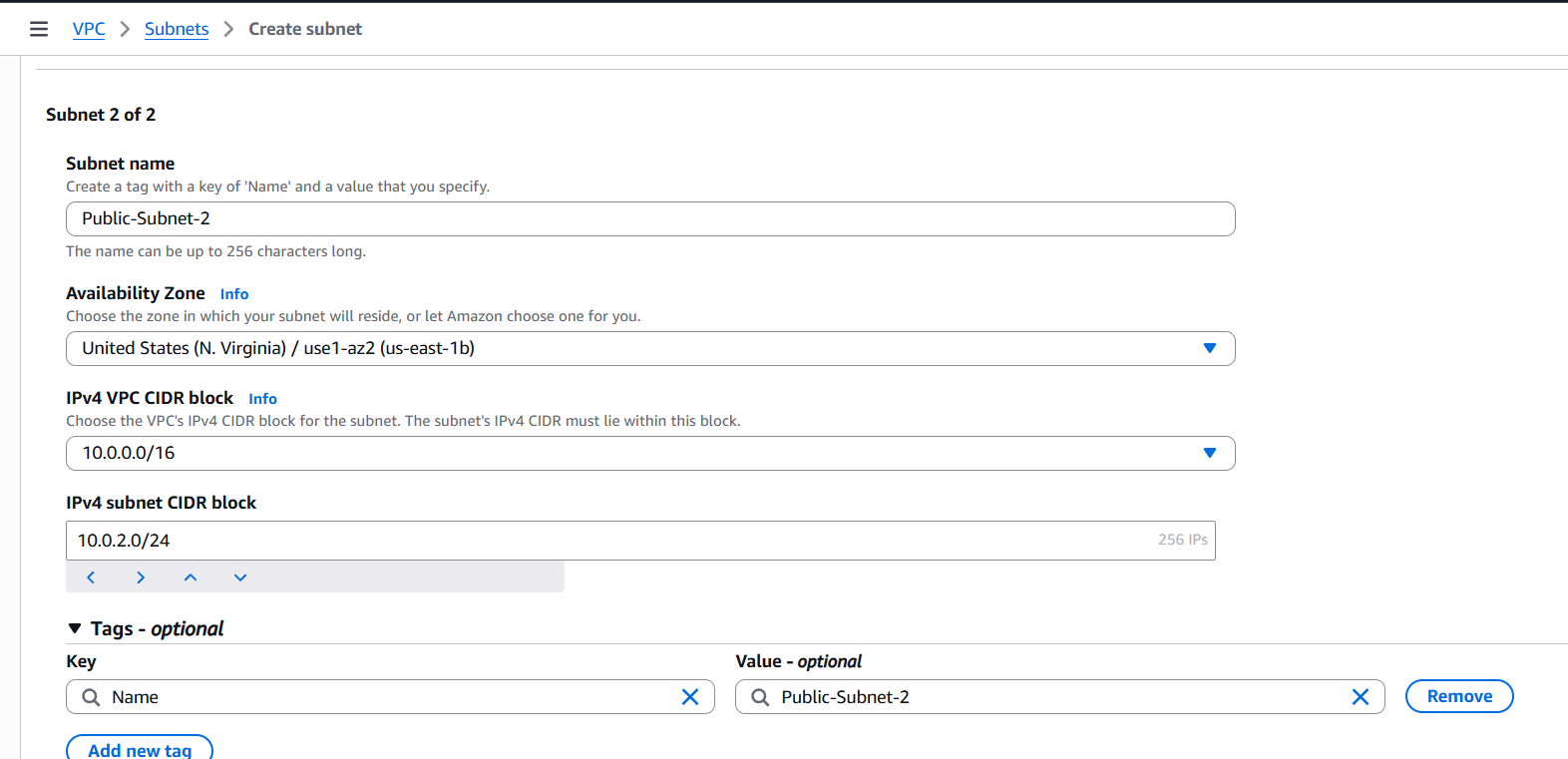


**Step 4: Create Subnets**

**Public Subnets:**

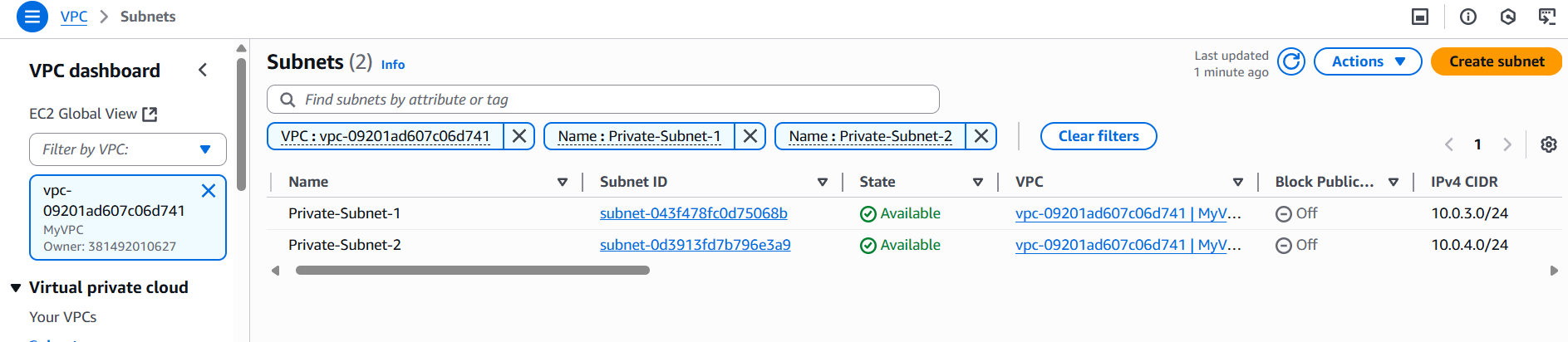
1. Go to **Subnets** → **Create subnet**
2. **Public Subnet 1:**
   * VPC: MyVPC
   * Name: Public-Subnet-1
   * AZ: us-east-1a (or your region's first AZ)
   * IPv4 CIDR: 10.0.1.0/24
3. **Public Subnet 2:**
   * Name: Public-Subnet-2
   * AZ: us-east-1b (different AZ)
   * IPv4 CIDR: 10.0.2.0/24





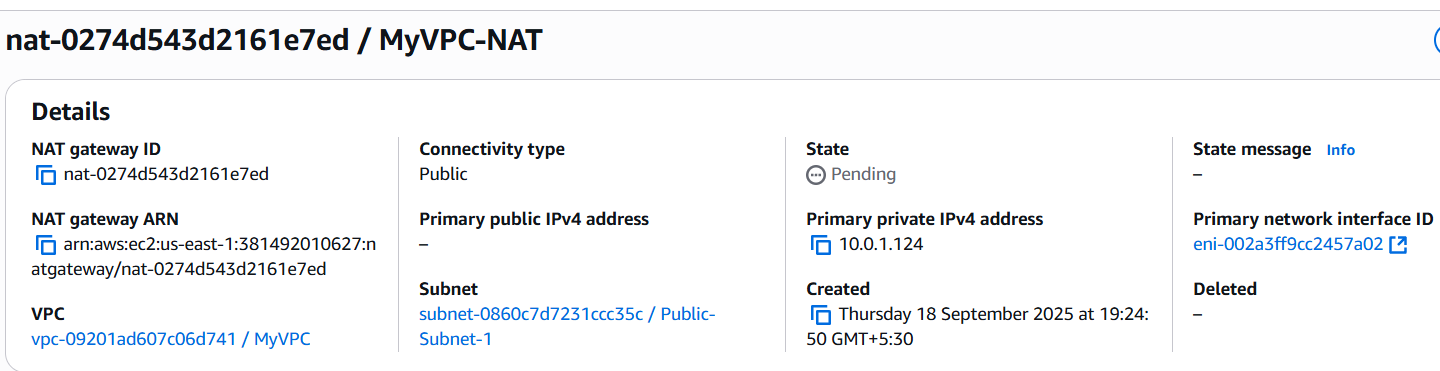
**Private Subnets:**

1. **Private Subnet 1:**
   * Name: Private-Subnet-1
   * AZ: us-east-1a
   * IPv4 CIDR: 10.0.3.0/24
2. **Private Subnet 2:**
   * Name: Private-Subnet-2
   * AZ: us-east-1b
   * IPv4 CIDR: 10.0.4.0/24



**Step 5: Create NAT Gateway**

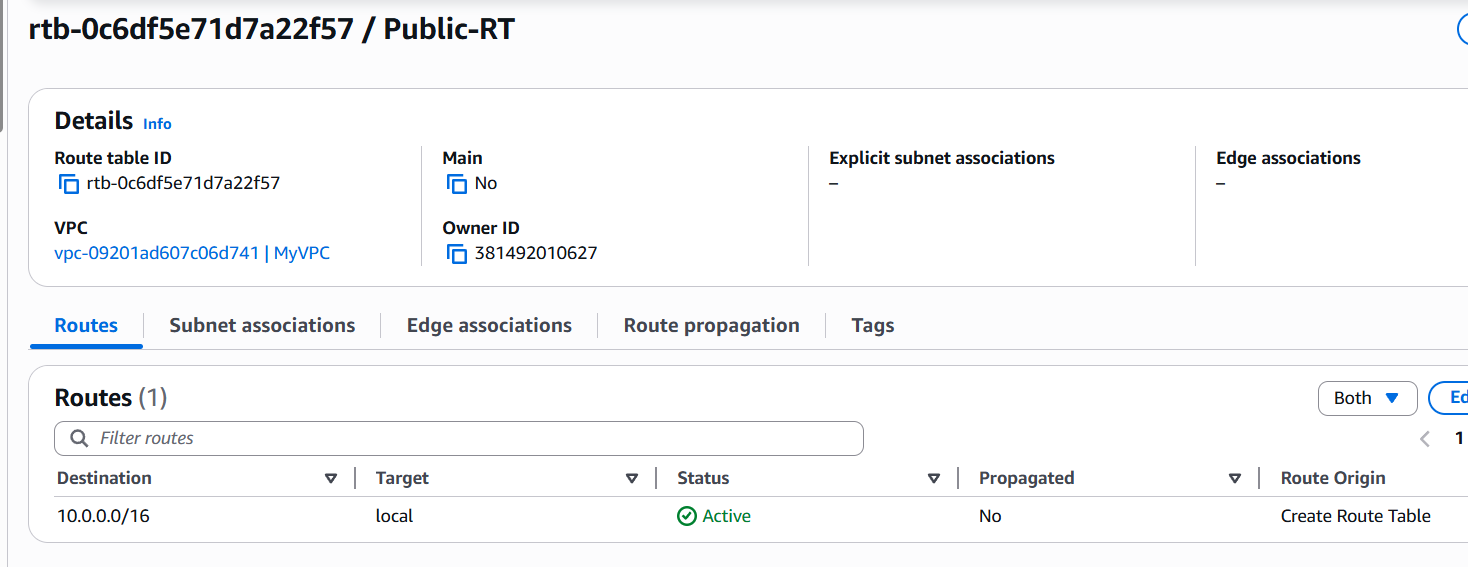
1. Go to **NAT Gateways** → **Create NAT gateway**
2. **Settings:**
   * Name: MyVPC-NAT
   * Subnet: Public-Subnet-1
   * Connectivity type: **Public**
   * Click **Allocate Elastic IP**
3. Click **Create NAT gateway**

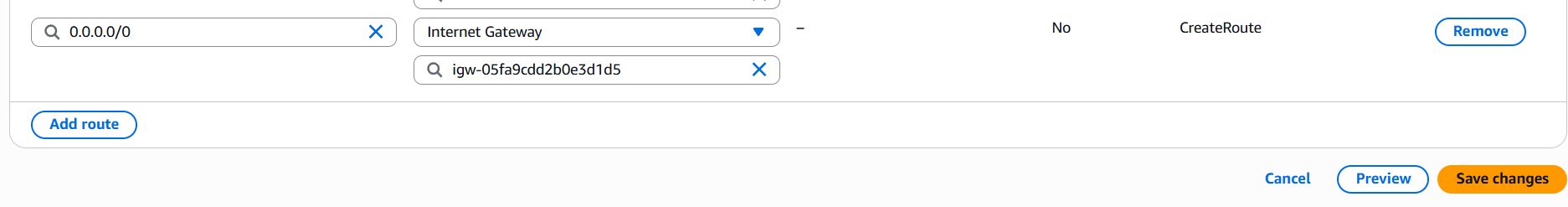


**Step 6: Create Route Tables**

**Public Route Table:**

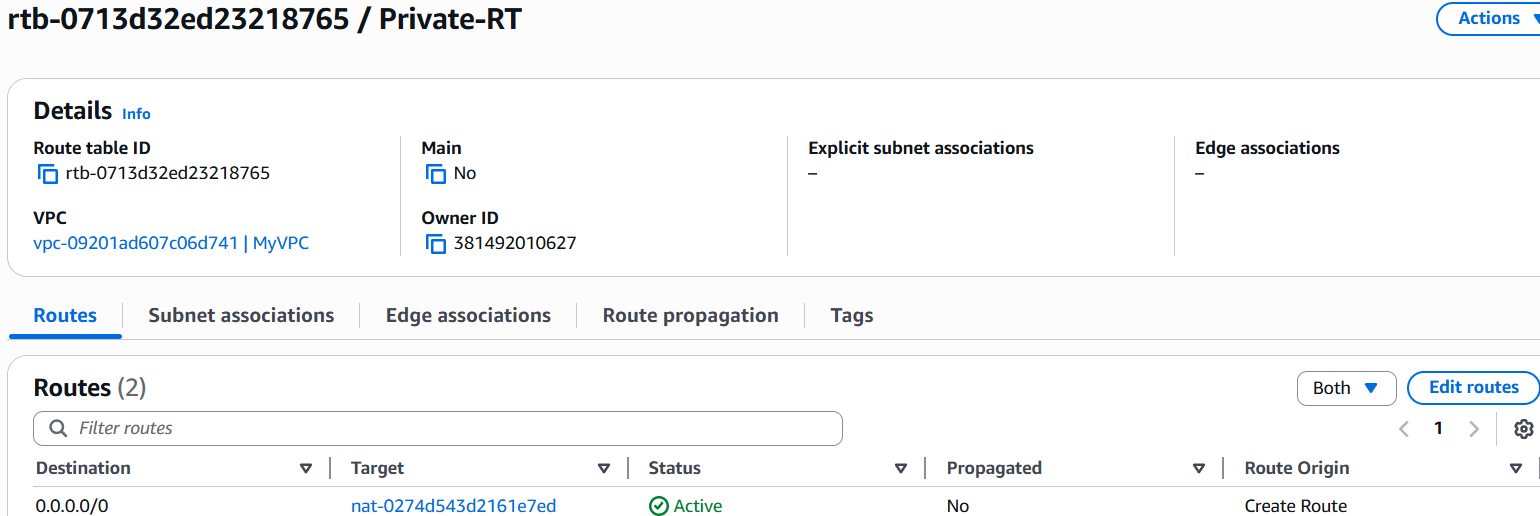
1. Go to **Route Tables** → **Create route table**
2. Name: Public-RT
3. VPC: MyVPC
4. Click **Create route table**
5. **Add Internet Route:**
   * Select Public-RT → **Routes** tab → **Edit routes**
   * Click **Add route**
   * Destination: 0.0.0.0/0
   * Target: **Internet Gateway** → MyVPC-IGW
   * Save changes





**Private Route Table:**

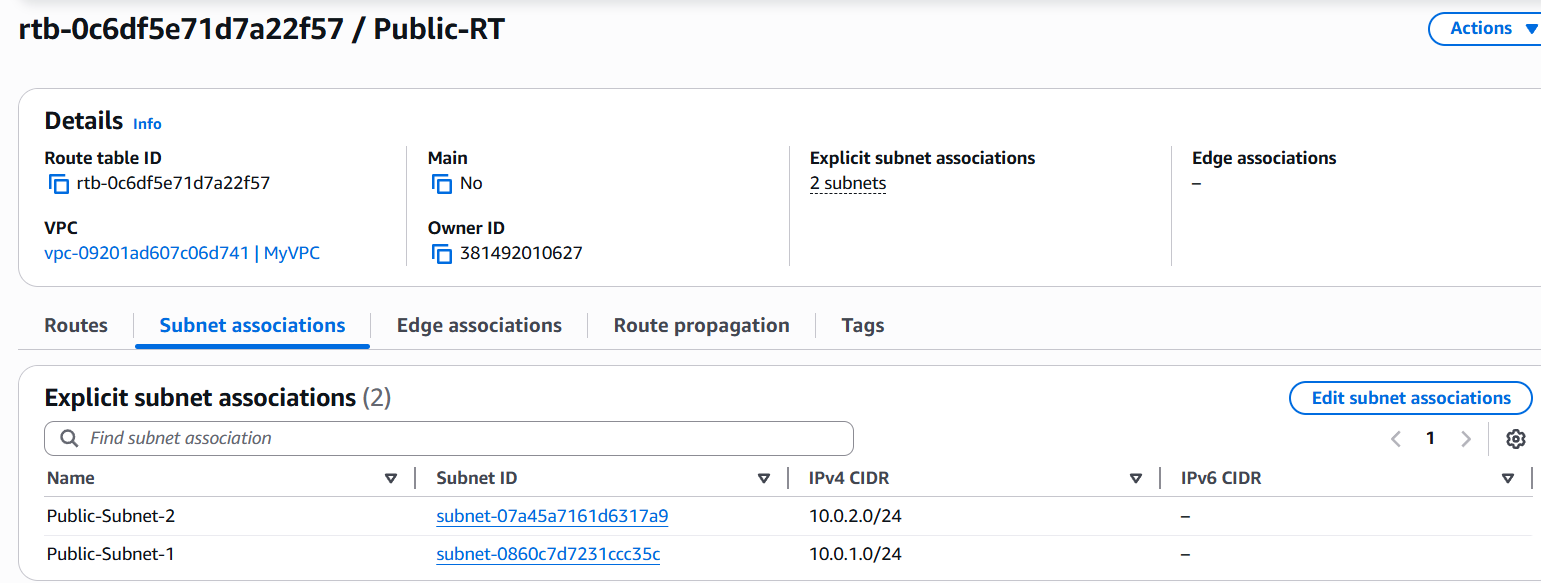
1. Create another route table:
   * Name: Private-RT
   * VPC: MyVPC
2. **Add NAT Route:**
   * Select Private-RT → **Routes** tab → **Edit routes**
   * Click **Add route**
   * Destination: 0.0.0.0/0
   * Target: **NAT Gateway** → MyVPC-NAT
   * Save changes



**Step 7: Associate Subnets with Route Tables**

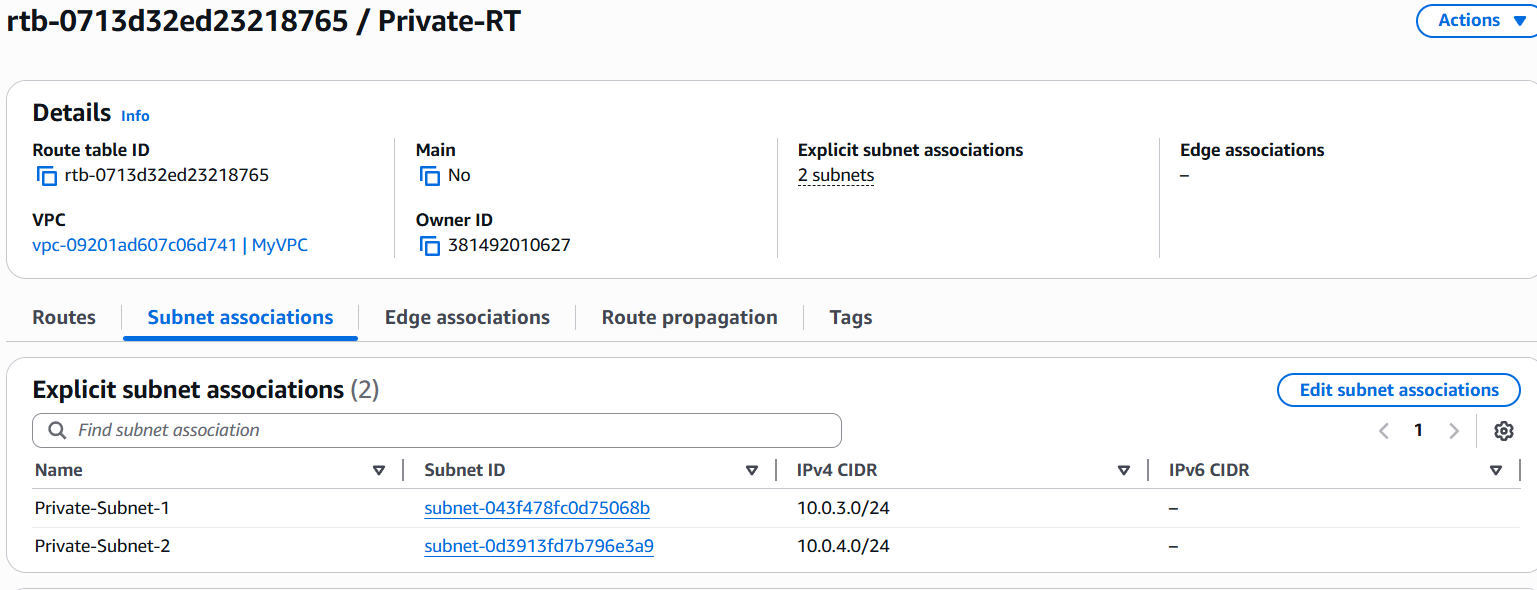
**Associate Public Subnets:**

1. Select Public-RT → **Subnet associations** tab
2. Click **Edit subnet associations**
3. Select Public-Subnet-1 and Public-Subnet-2
4. Save associations



**Associate Private Subnets:**

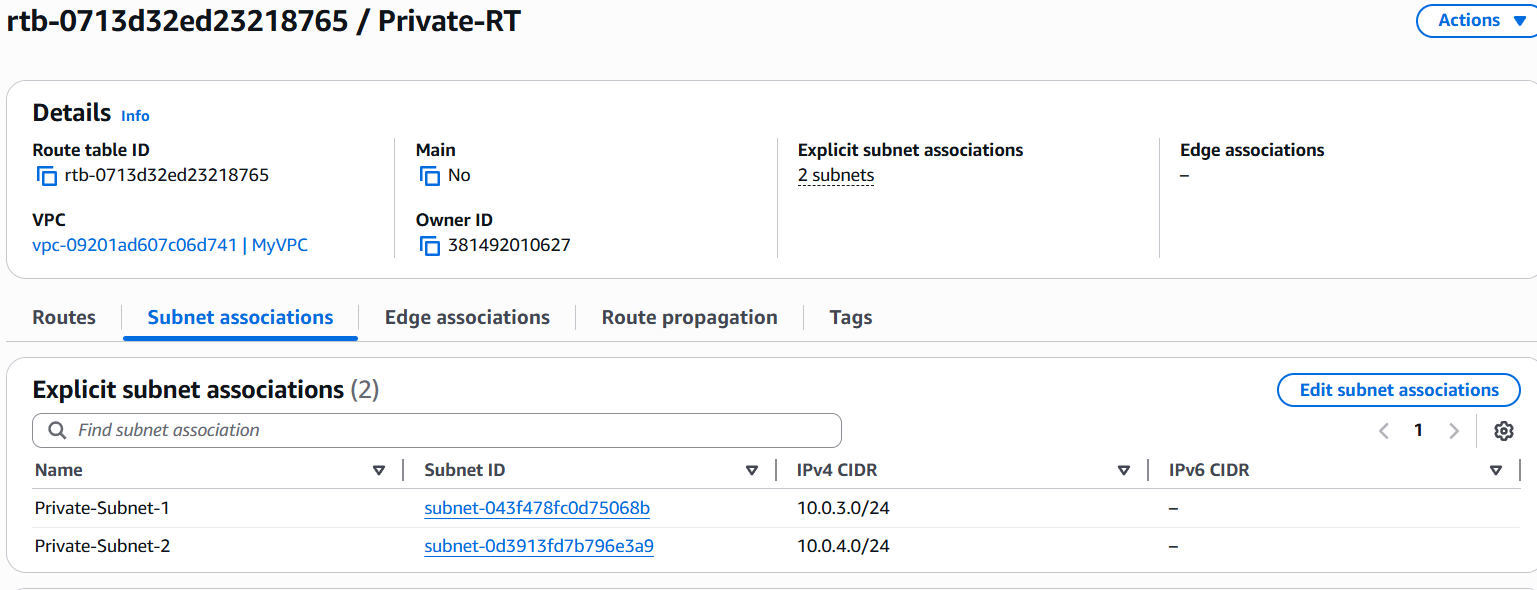
1. Select Private-RT → **Subnet associations** tab
2. Click **Edit subnet associations**
3. Select Private-Subnet-1 and Private-Subnet-2
4. Save associations



**Step 8: Create Security Groups**

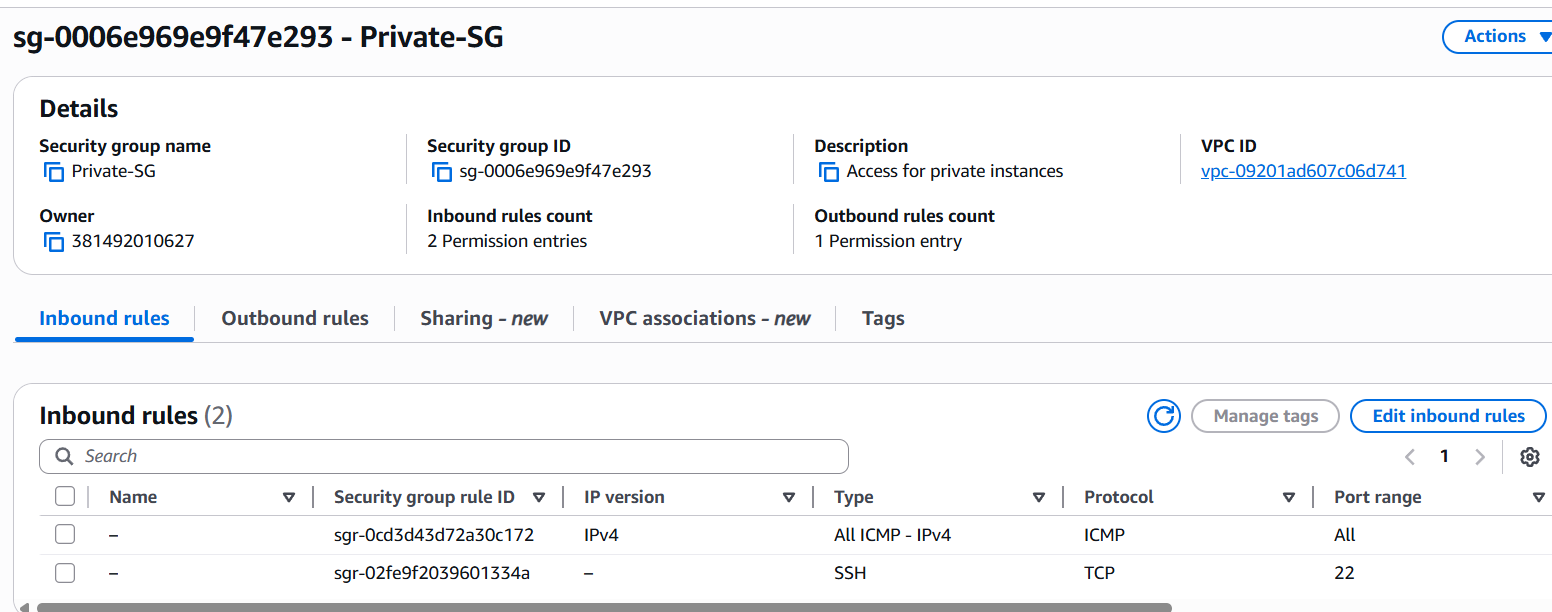
**Bastion Host Security Group:**

1. Go to **Security Groups** → **Create security group**
2. **Settings:**
   * Name: Bastion-SG
   * Description: SSH access for bastion host
   * VPC: MyVPC
3. **Inbound Rules:**
   * Type: SSH, Port: 22, Source: Your IP/32 (or 0.0.0.0/0 for testing)
4. **Outbound Rules:** (Keep default - All traffic)



**Private Instance Security Group:**

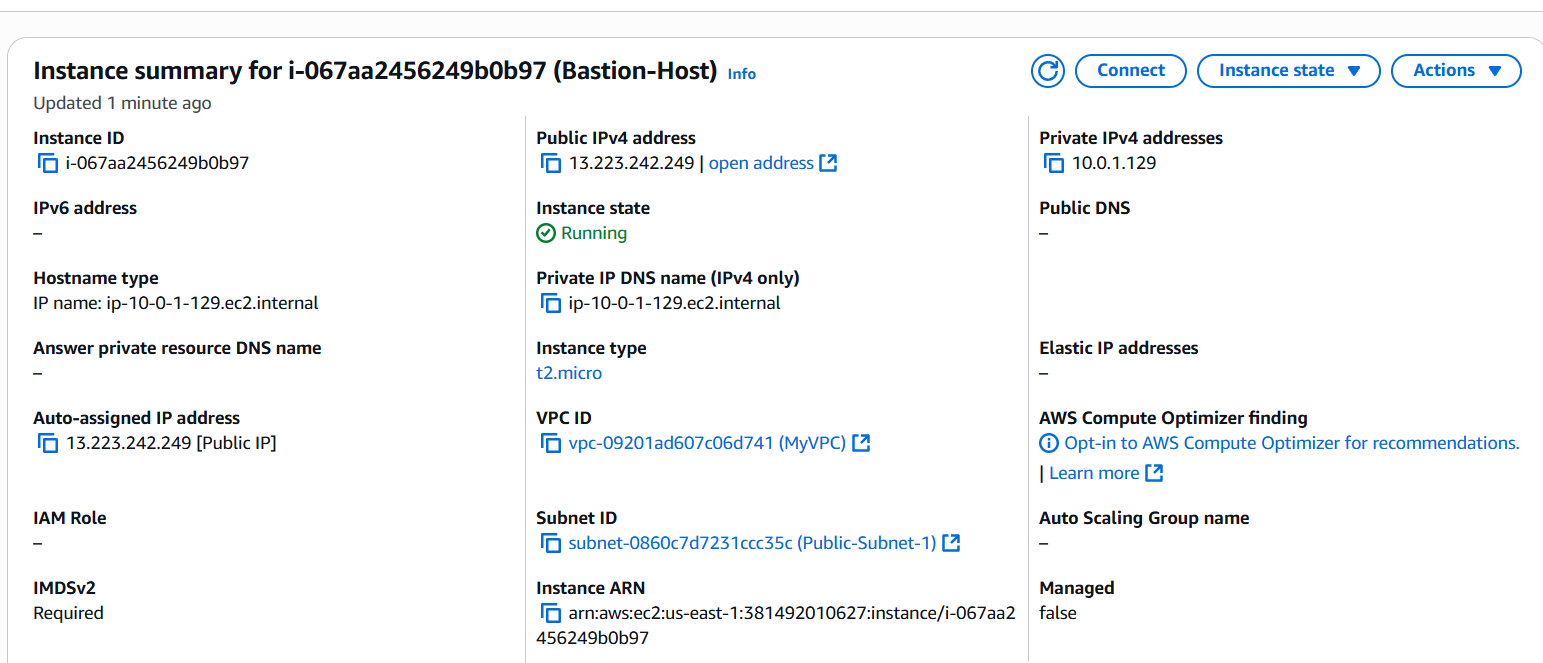
1. Create another security group:
   * Name: Private-SG
   * Description: Access for private instances
   * VPC: MyVPC
2. **Inbound Rules:**
   * Type: SSH, Port: 22, Source: Bastion-SG (select the bastion security group)
   * Type: All ICMP-IPv4, Source: 10.0.0.0/16 (for ping testing)



**Step 9: Launch EC2 Instances**

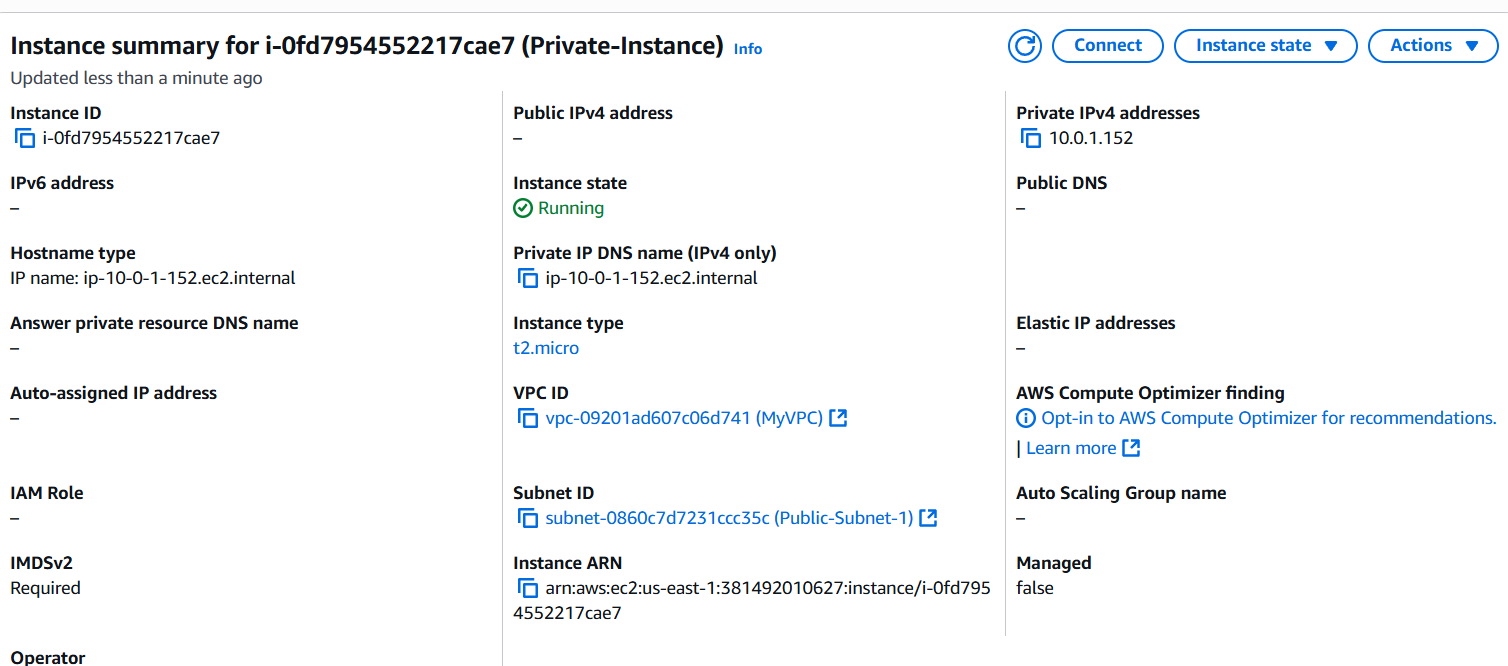
**Public Instance (Bastion Host):**

1. Go to **EC2** → **Launch Instance**
2. **Settings:**
   * Name: Bastion-Host
   * AMI: **Amazon Linux 2**
   * Instance type: t2.micro
   * Key pair: my-vpc-key
   * Network: MyVPC
   * Subnet: Public-Subnet-1
   * **Auto-assign public IP: Enable**
   * Security group: Bastion-SG
3. Launch instance



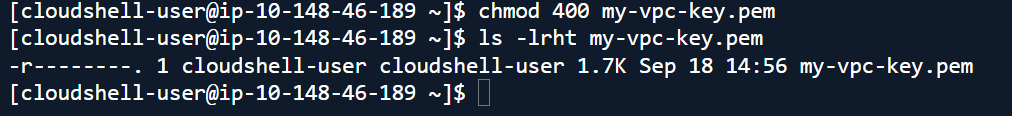
**Private Instance:**

1. Launch another instance:
   * Name: Private-Instance
   * AMI: **Amazon Linux 2**
   * Instance type: t2.micro
   * Key pair: my-vpc-key
   * Network: MyVPC
   * Subnet: Private-Subnet-1
   * **Auto-assign public IP: Disable**
   * Security group: Private-SG
2. Launch instance

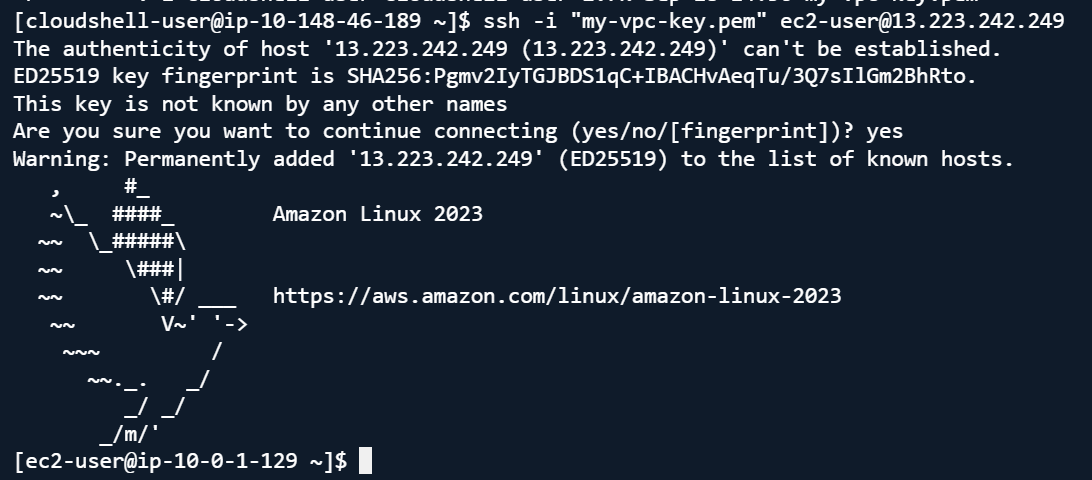


**Step 10: Connect to Instances**

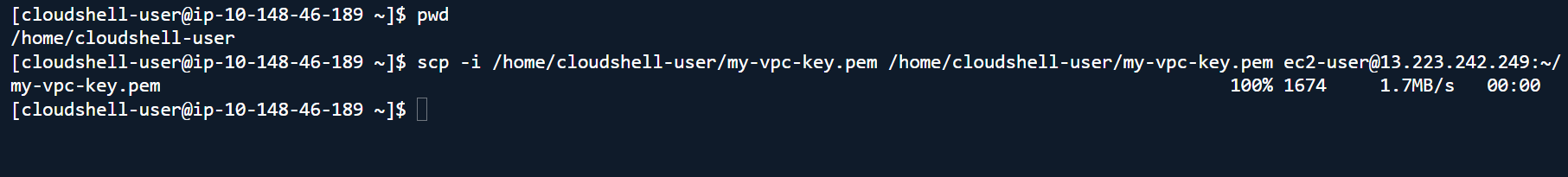
**Step 10.1: Prepare Your SSH Key**



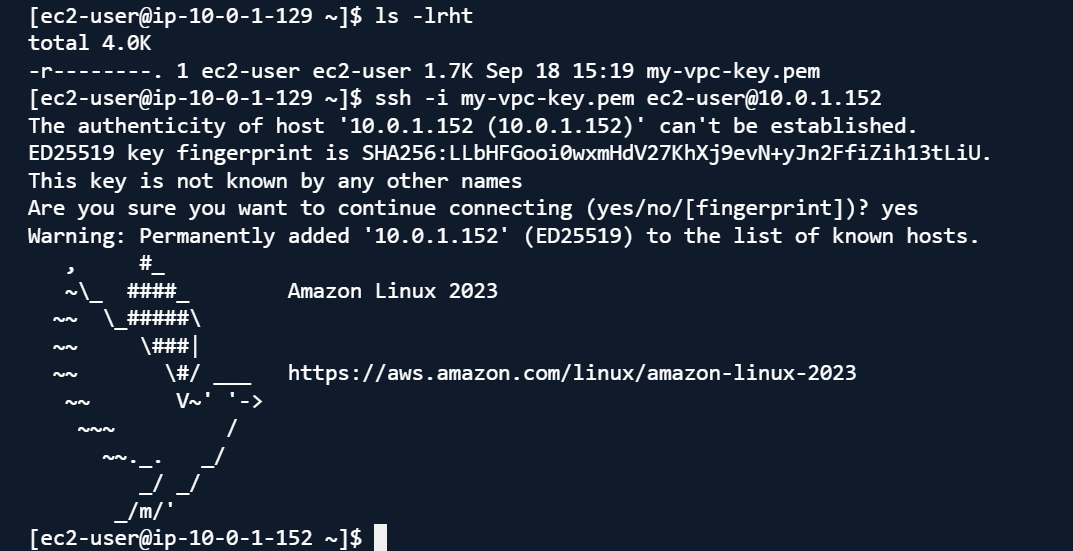
**Step 10.2: Connect to Bastion Host**



**Step 10.3: Copy Key to Bastion Host:**

****

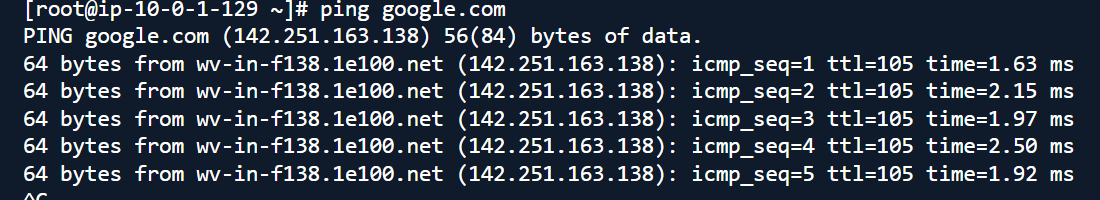
**Step 10.4: Connect to Private Instance via Bastion:**

****

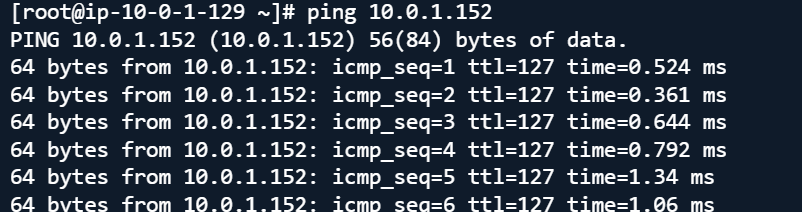
**Testing Your Setup**

**Test Internet Access:**

**From Bastion Host:**

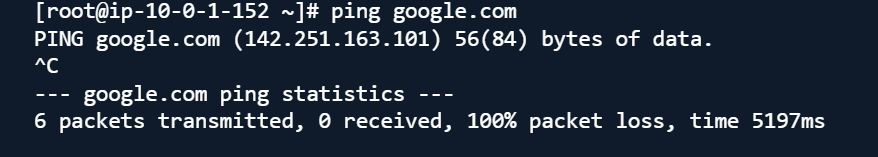
****

**Test Connectivity:**

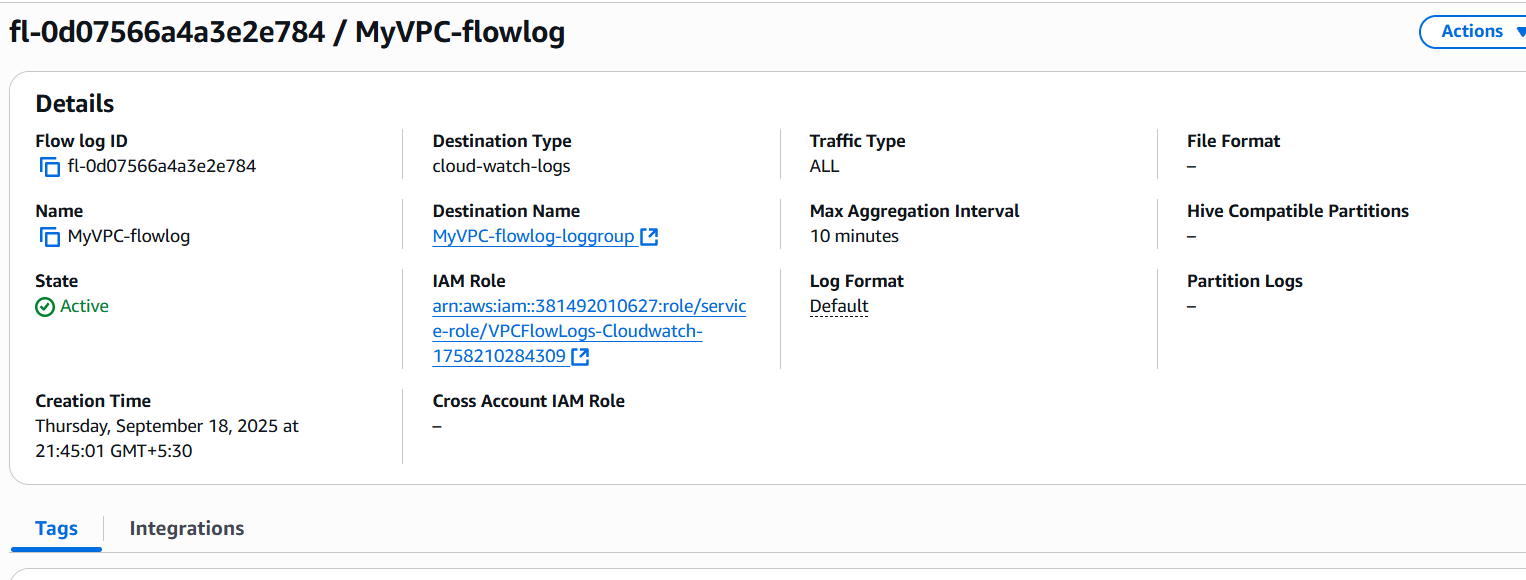


**Issue Faced:**

Not able to test connectivity from private instance getting error like :



Creating VPC Flow log:



Flow logs stored in CloudWatch log group:



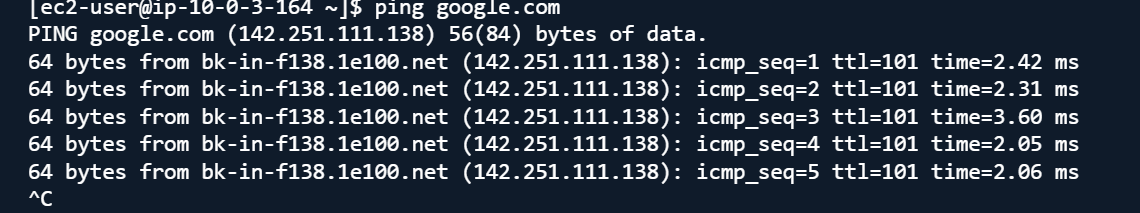
troubleshooting summary in bullet points:

* Created a private EC2 instance inside a private subnet, accessed only through a bastion host.
* Initially tried SSH directly to the private instance's public IP, but the connection timed out.
* Investigated network flow logs and found inbound SSH traffic on port 22 was blocked for the public IP.
* Confirmed that security group rules were restricting access to the public IP.
* Verified that SSH to the private instance using its private IP from the bastion host worked successfully.
* Realized the private instance does not have direct public internet access.
* Using the public IP in the private subnet routing caused the NAT Gateway to block or not properly forward traffic

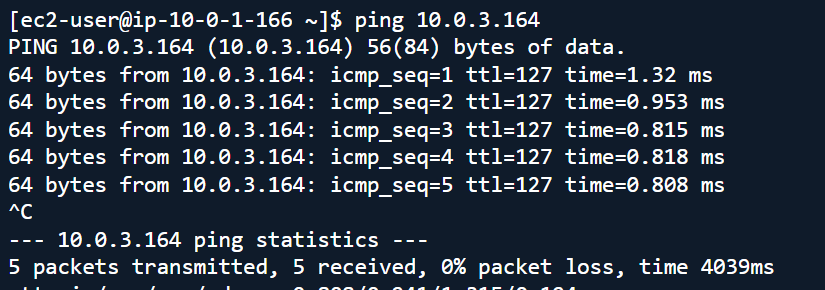
What I learnt:

* Learned that private instances must be accessed via the bastion host’s private subnet with correct permissions.
* Private instances in a private subnet should use the NAT Gateway’s private IP as the route target, not a public IP
* Understood that proper security groups, routing, and subnet configuration are essential for secure and functional access.

**Test connectivity from Private Instance:**



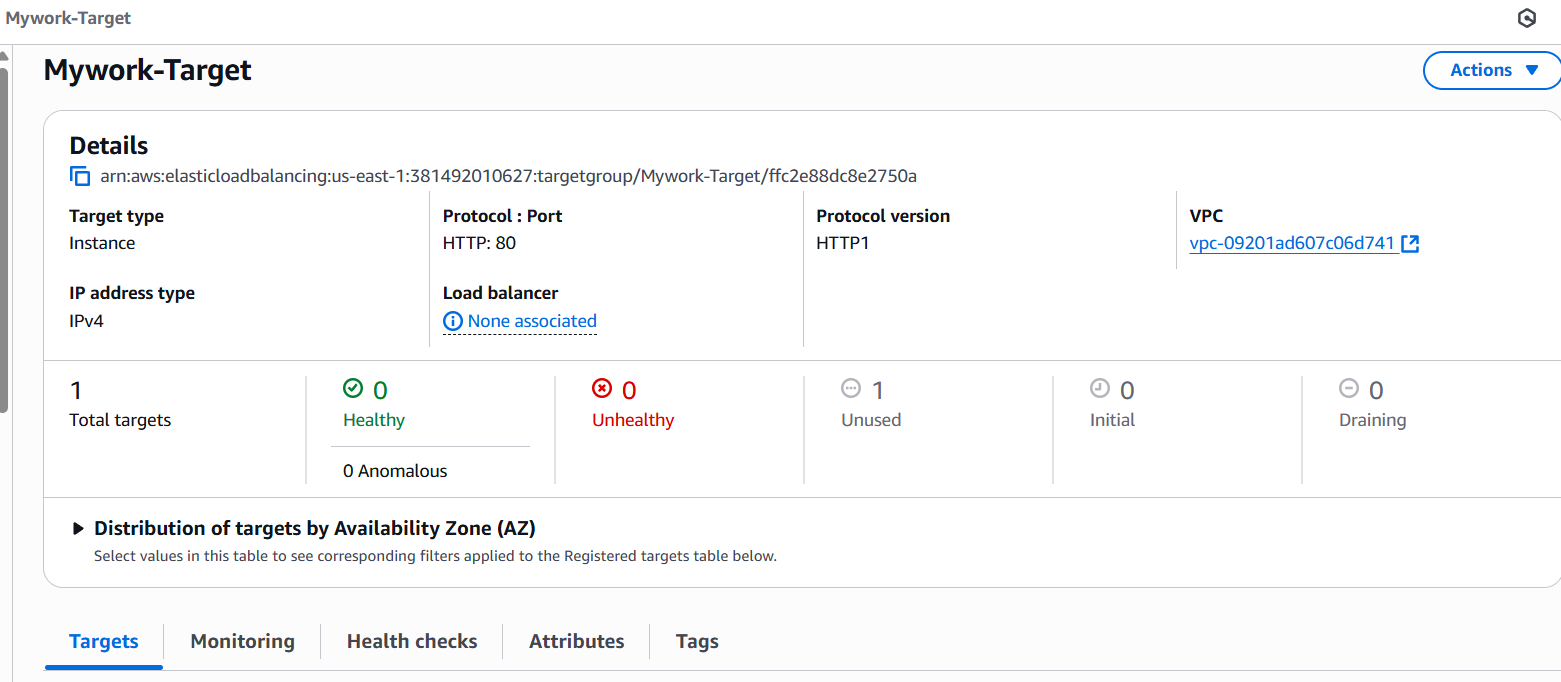
Pinging from Bastion Host:

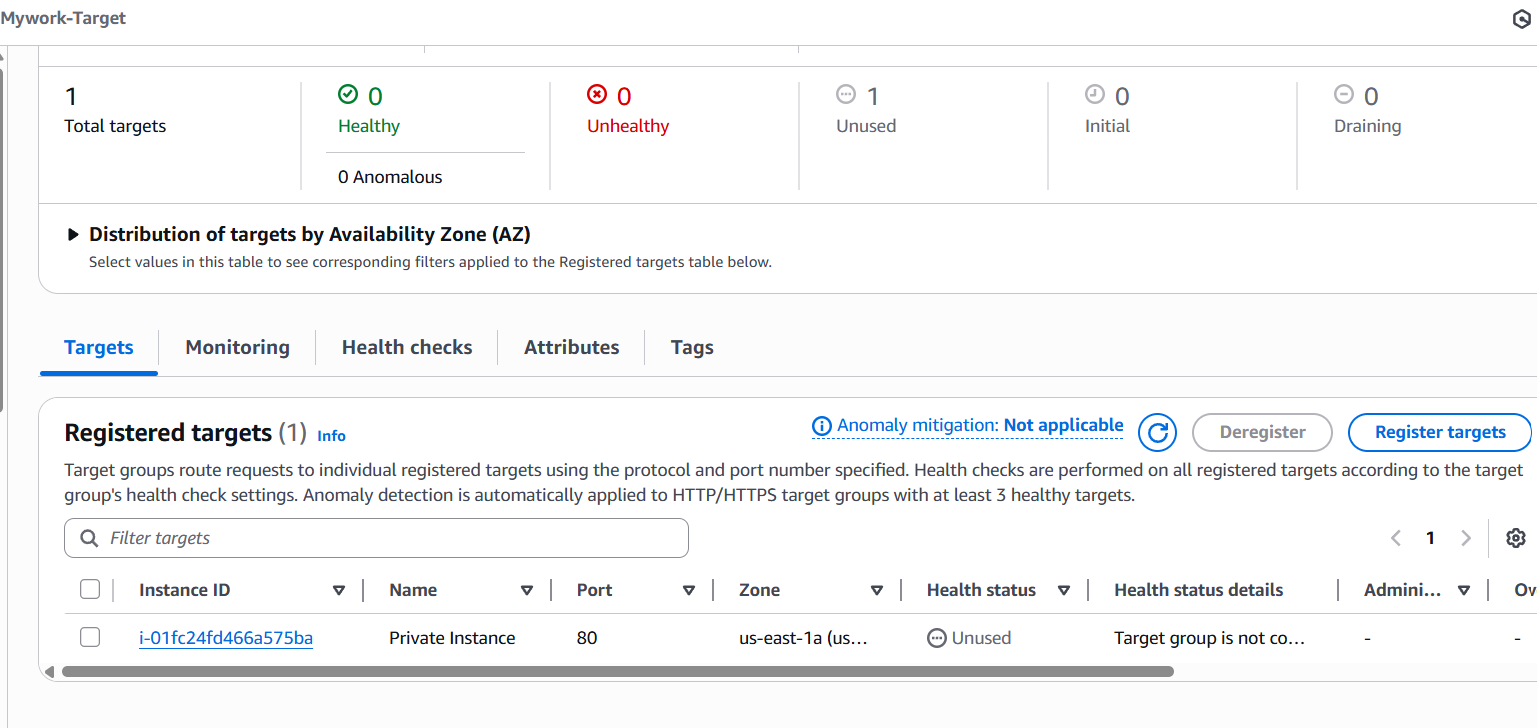


Bonus Challenges:

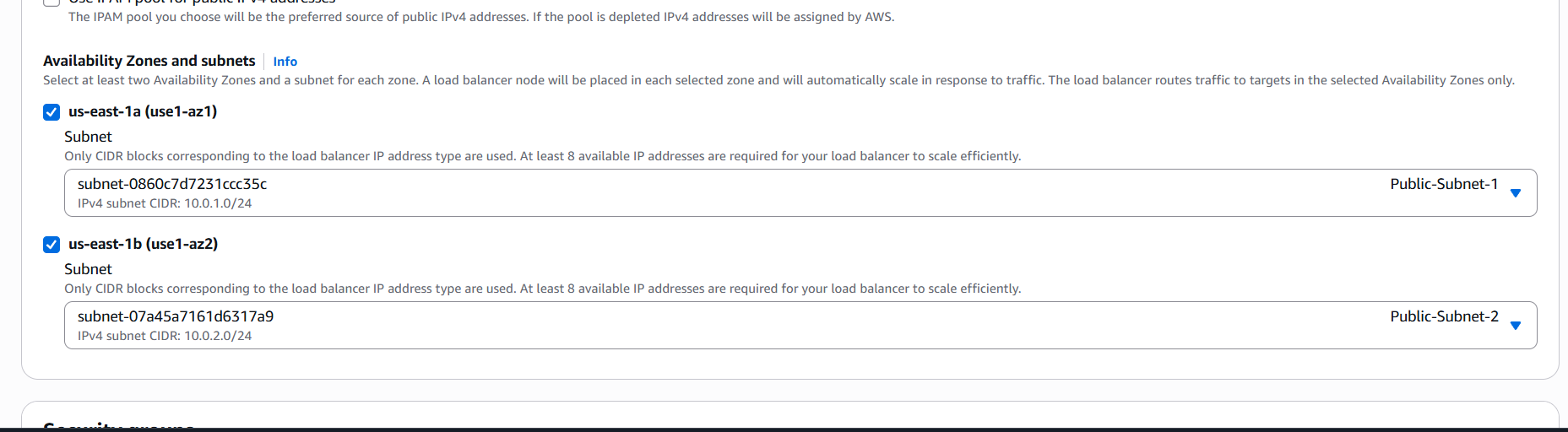
1. **Add an Application Load Balancer** in public subnets

Created the target which is in my private subnet



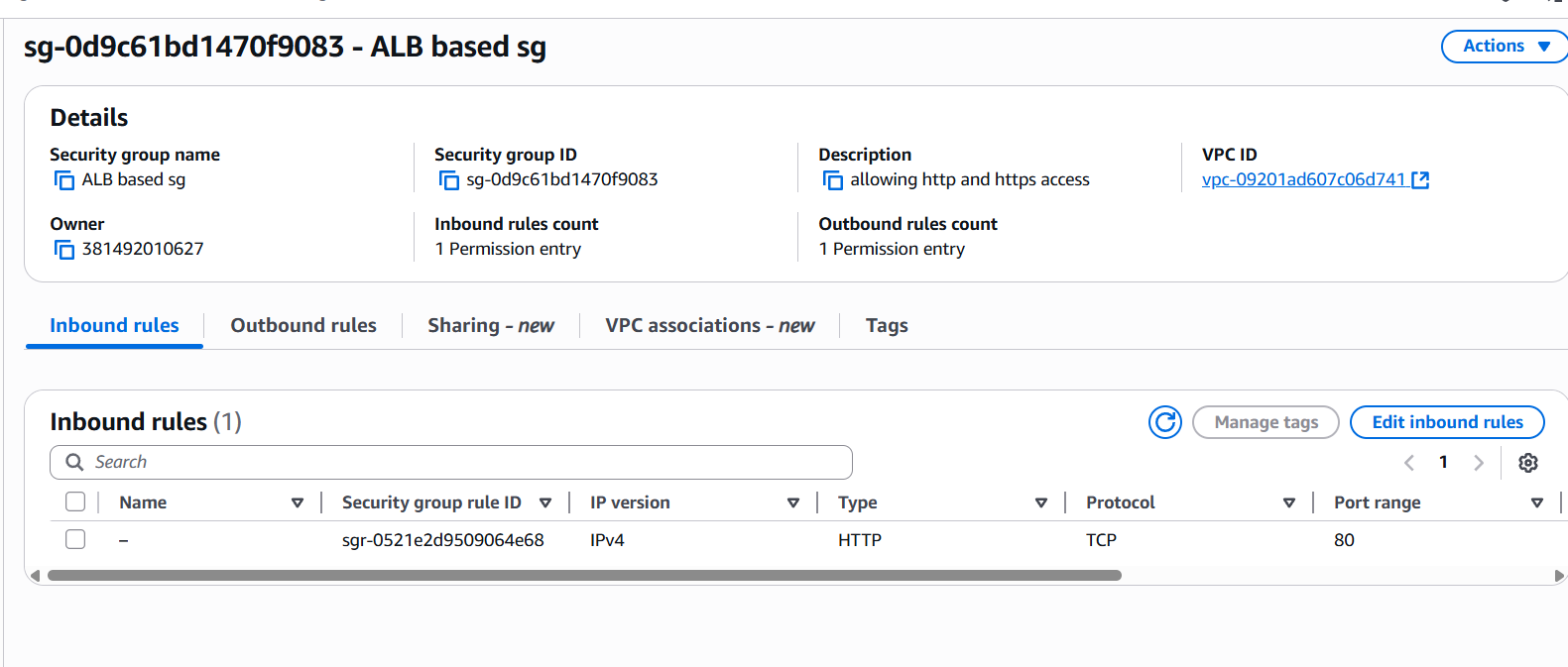


Created the internet facing application load balancer :

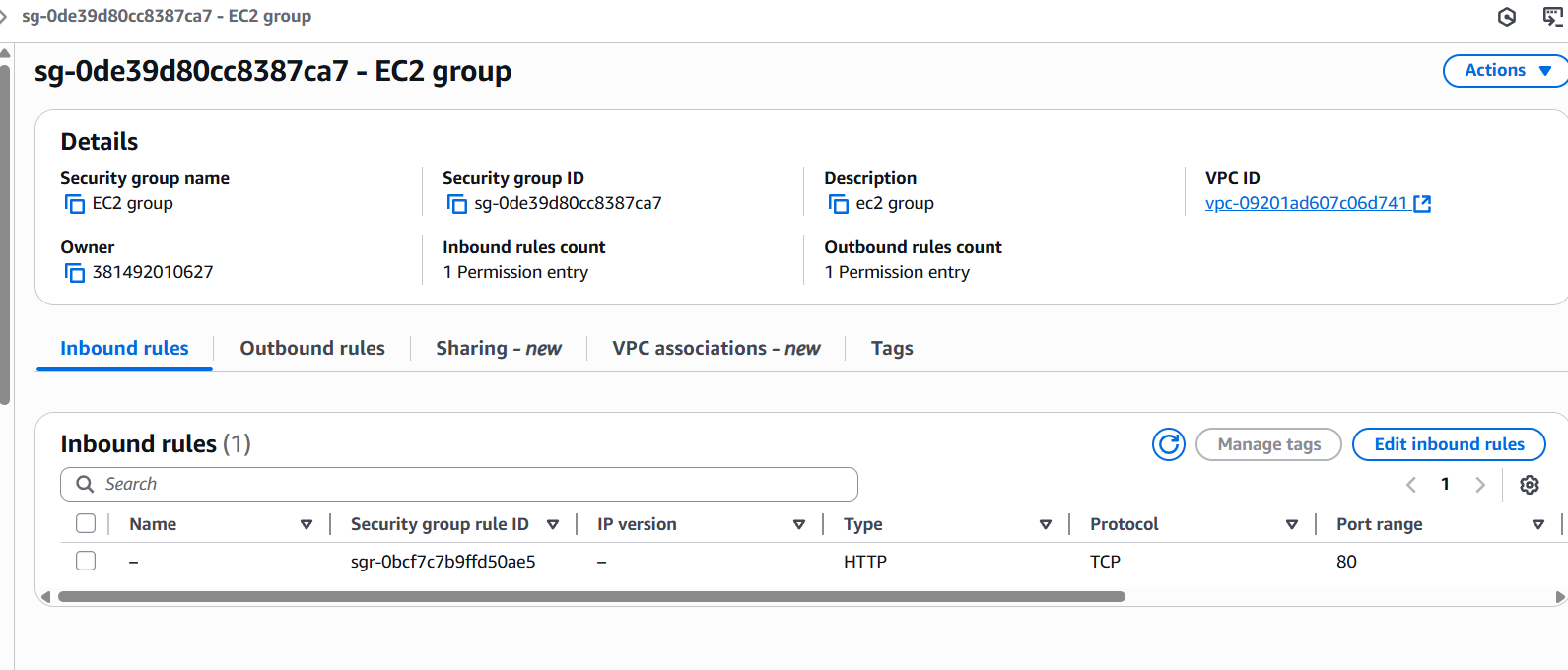


Creating the security groups:

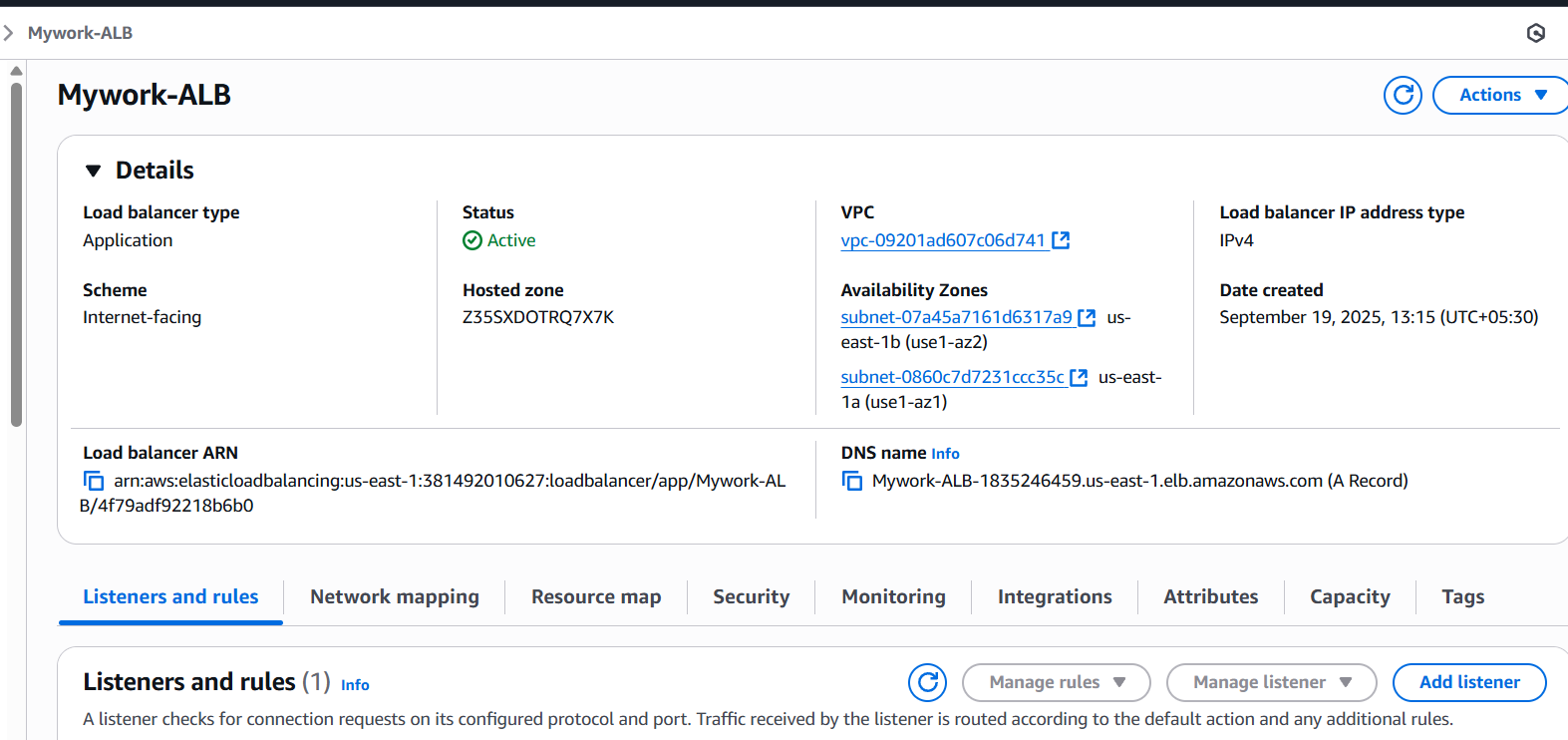
* First will be the ALB sg where inbound rules allowed for 80 from 0.0.0.0/0. Outbound to anywhere (default) is kept for testing.

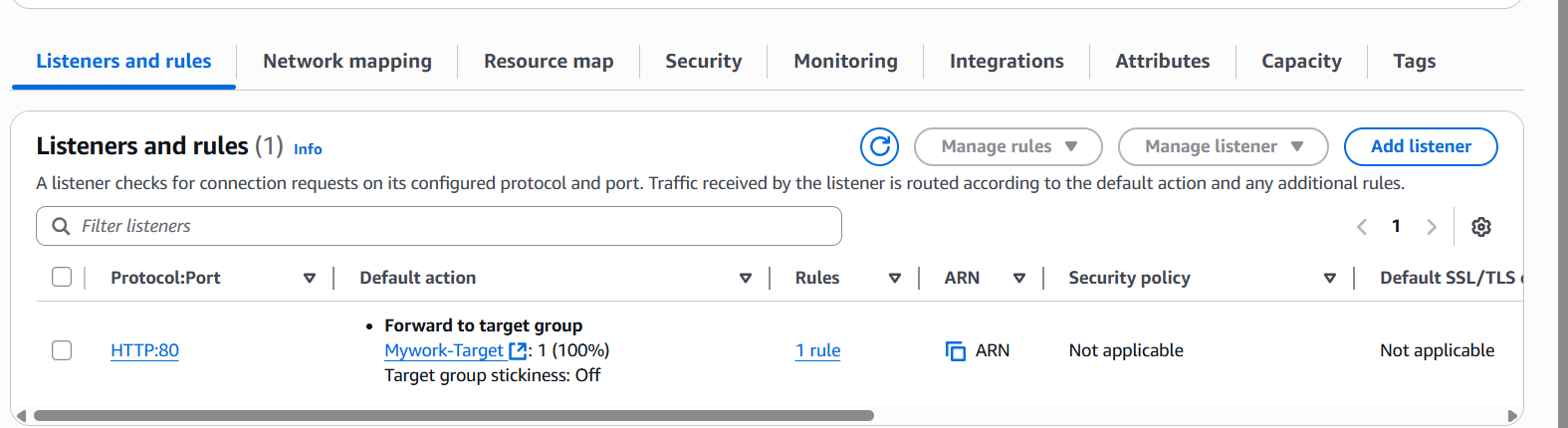


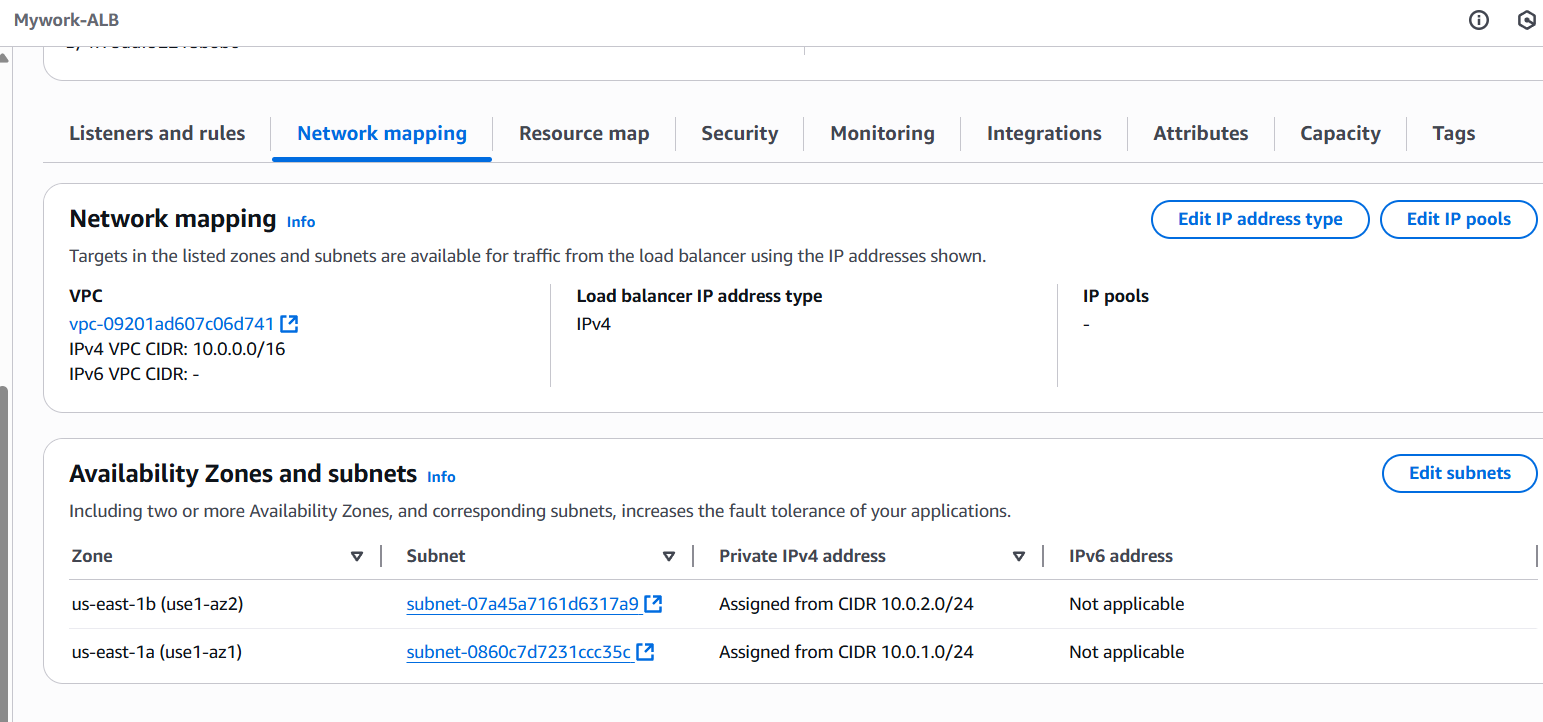
* EC2 based sg has been created:



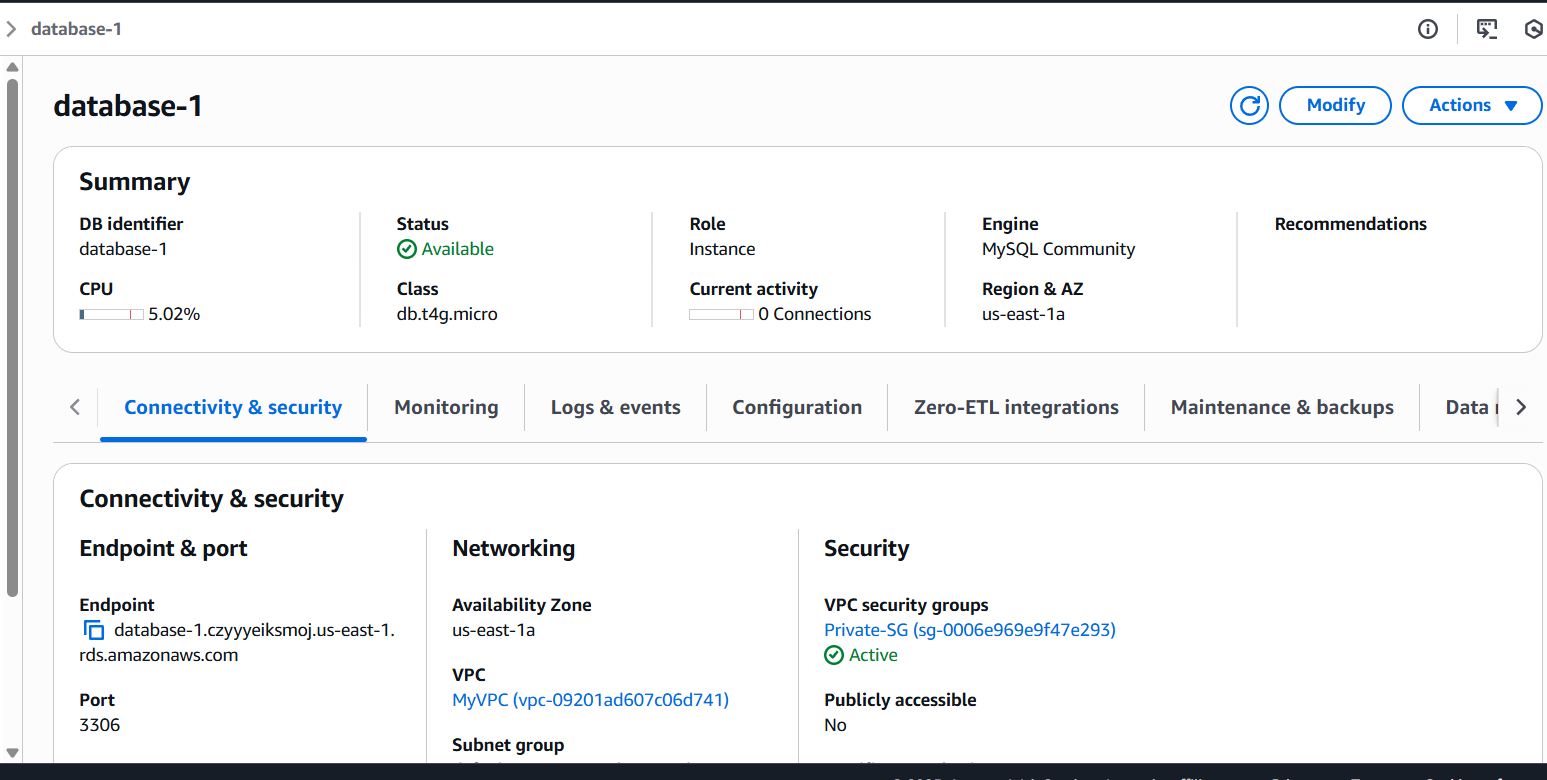
**ALB created:**



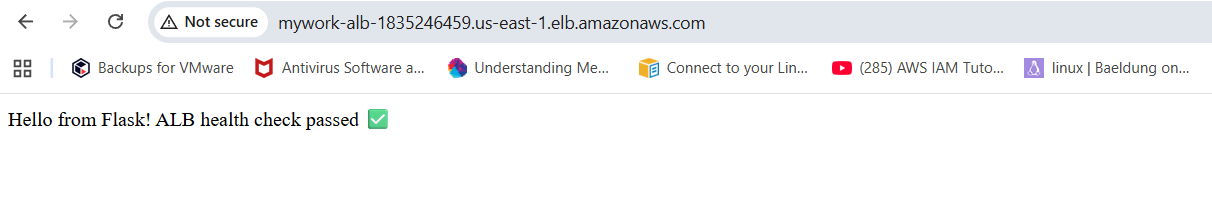




**2. Create a database** in private subnets:



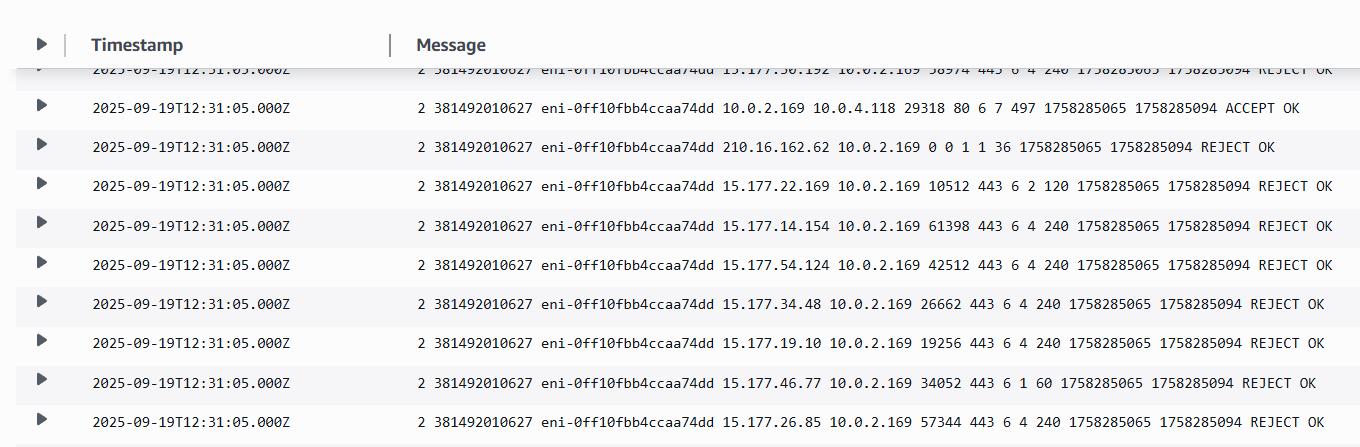
As checked from the browser:

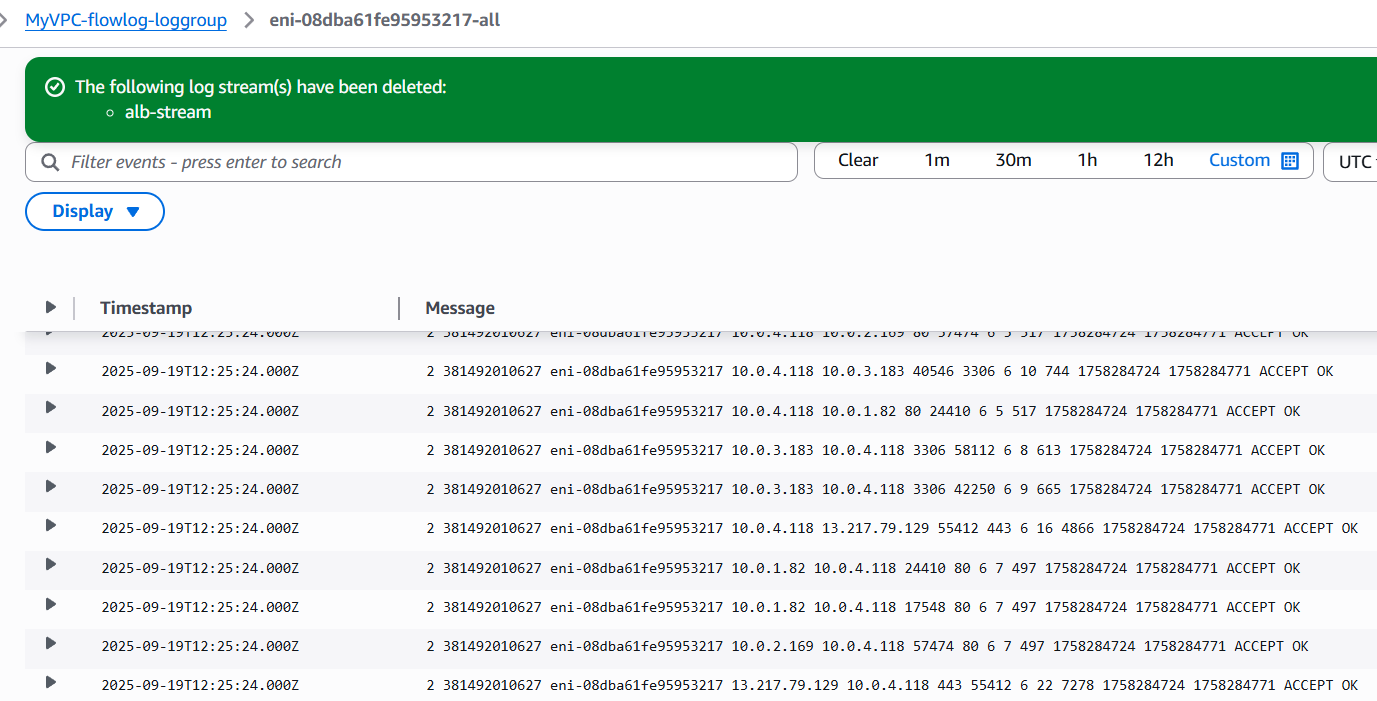


Connected to DB which is in private subnet :

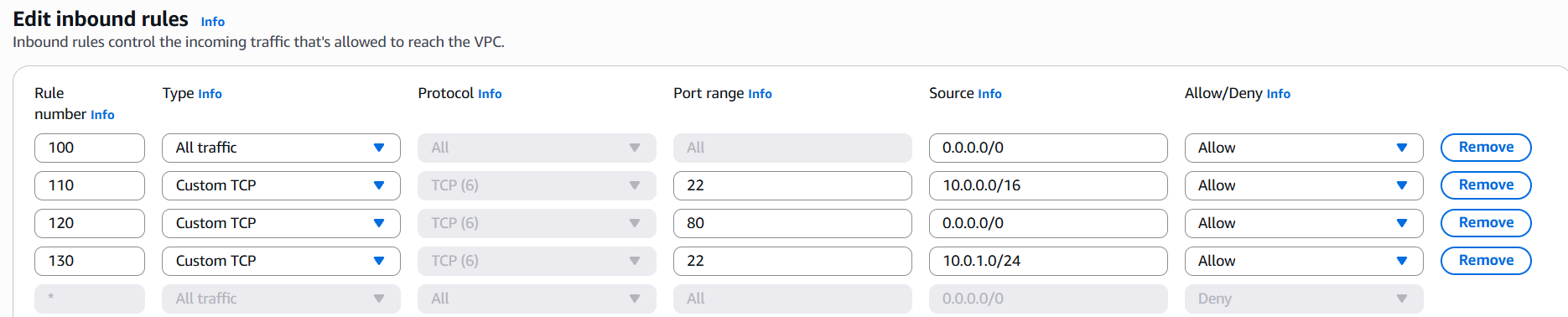


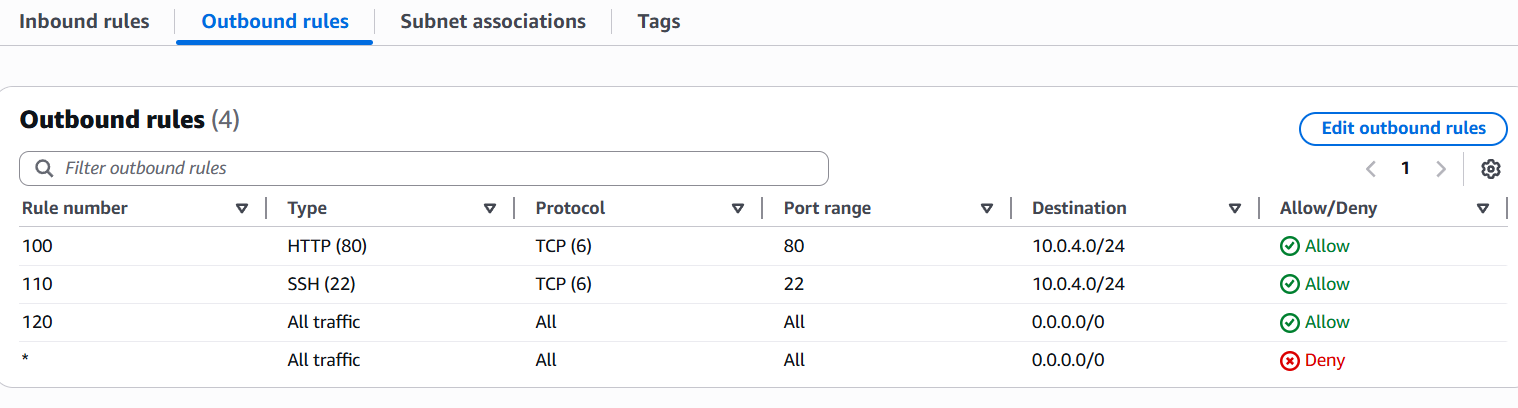
VPC flow logs configured for the ALB Security group and the application hosted in private subnet for monitoring:





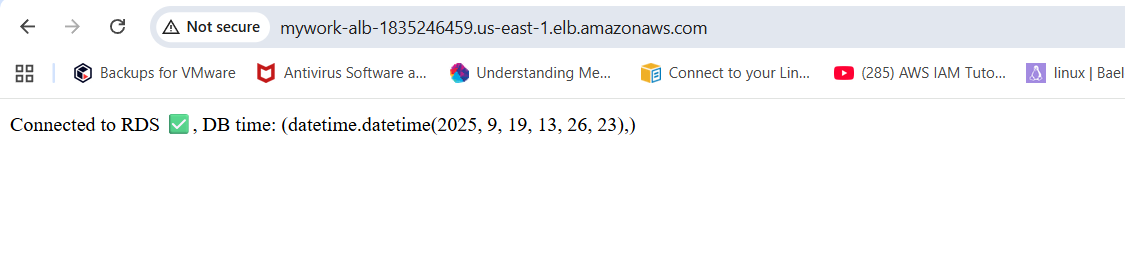
Custom NACL has been applied :





Instead of using Bastion Host and SSH , used Session manager system manager :





Setup Used for the bonus challenge:

With this setup:

* **App servers** in private subnet → **no public IPs**.
* **Outbound traffic** → via **NAT Gateway**.
* **Inbound traffic** → via **ALB**.
* **Testing access** → via **Bastion Host**.(later used session manager without SSH)

**Security Group Used :**

**Bastion SG:**

**Inbound : TCP Protocol, Port:22, Source: 0.0.0.0/0**

**Outbound: Default**

**ALB SG:**

**Inbound: TCP Protocol, Port:80, Source: 0.0.0.0/0**

**Outbound: Default**

**APP SG:**

**Inbound: TCP Protocol,Port:80, Source: ALB-SG ; TCP Protocol,Port-22,Source:Bastion SG**

**Outbound: Default**

**Issue Faced:**

* Hosted an application in private subnets across multiple Availability Zones.
* Created an Application Load Balancer (ALB) in the public subnet to route traffic to app instances in private subnets.
* Configured Amazon RDS in a different Availability Zone within the same private subnet.
* Security group rules were initially misconfigured, leading to connectivity issues between the app servers, ALB, and the RDS database.
* Required secure management access to private instances, initially done via a Bastion host, which posed security risks.

**Work Done:**

* Configured Bastion SG with inbound SSH (22) access from anywhere for initial setup.
* Created ALB SG allowing inbound HTTP(80) from anywhere to route traffic to targets (App EC2).
* Created App SG allowing inbound traffic from ALB SG (HTTP) and Bastion SG (SSH).
* Registered App EC2 in private subnet as ALB target.
* Updated Python Flask app to dynamically assign RDS DB name when not provided in creation.
* Confirmed integrations by accessing the flask app via ALB DNS and validating DB connectivity via RDS endpoint.
* Enabled VPC Flow Logs to monitor all traffic patterns and troubleshoot connectivity issues.
* Created a custom NACL for finer-grained traffic control between public and private subnets.
* Migrated access method from Bastion host SSH to AWS Systems Manager Session Manager for more secure and auditable access.

**Learnings**

* Proper security group design is critical for ensuring application flow without opening unintended access.
* App instances in private subnets can securely serve traffic via ALB in a public subnet without exposing themselves to the internet.
* Dynamically assigning database names in the Flask app improves flexibility during deployments.
* VPC Flow Logs are invaluable for troubleshooting network connectivity issues across complex subnet setups.
* Custom NACLs and security groups together strengthen network segmentation and security posture.
* Using Session Manager eliminates the need for a Bastion host, reducing attack surface while maintaining secure access to private infrastructure.