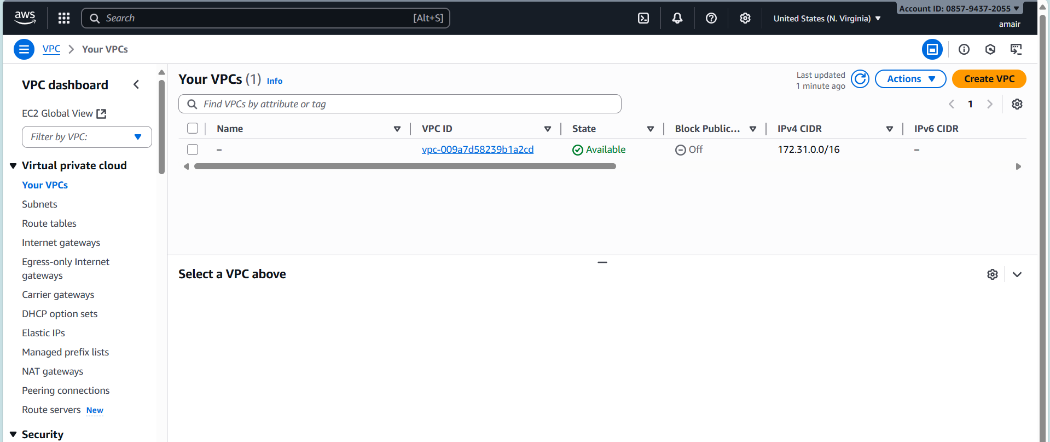
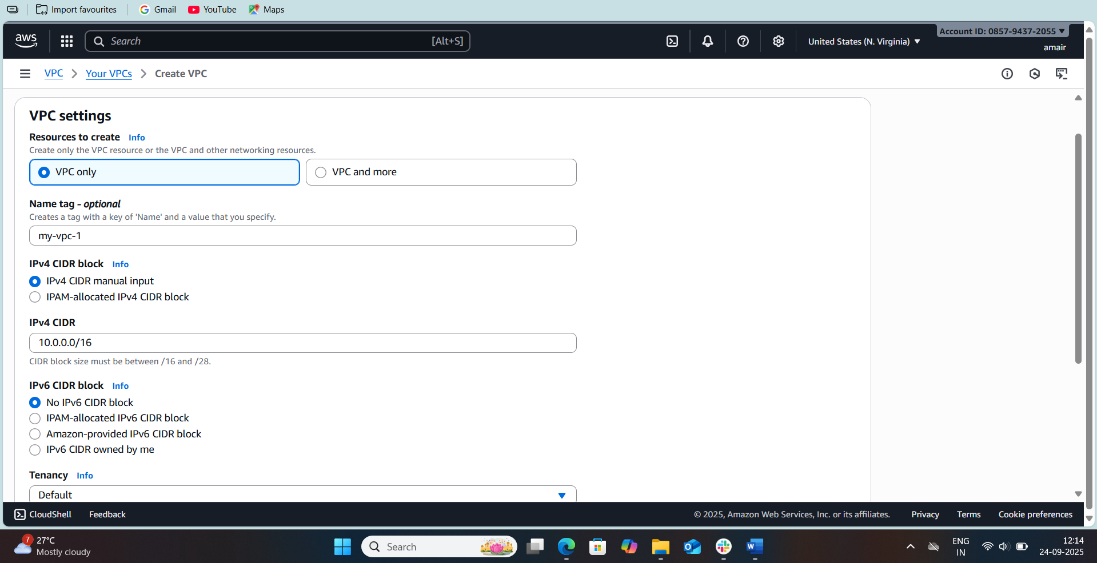
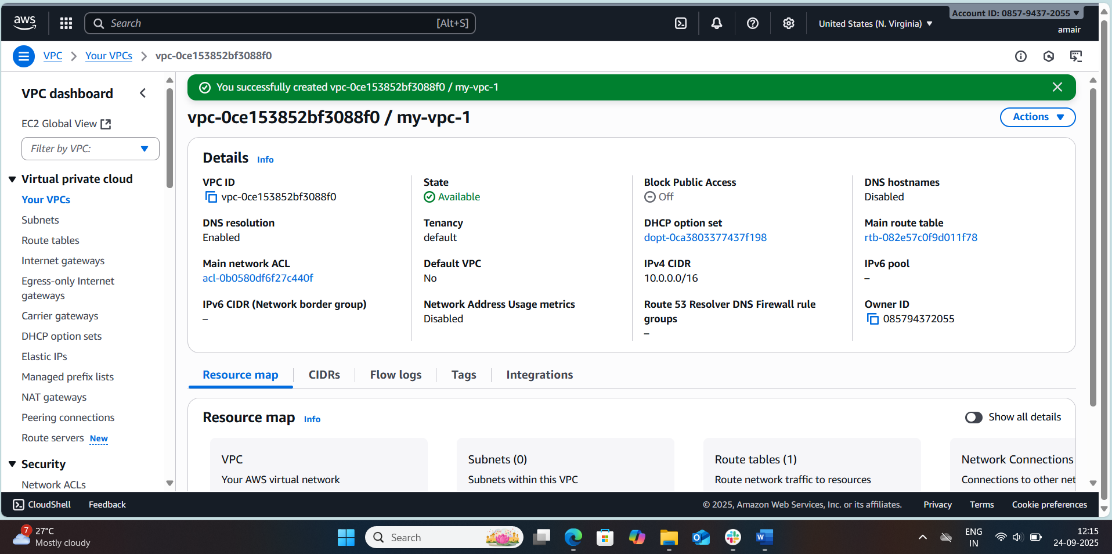
**VPC Task-1:**

1. **Create VPC with 2 private and 2 public subnets.**

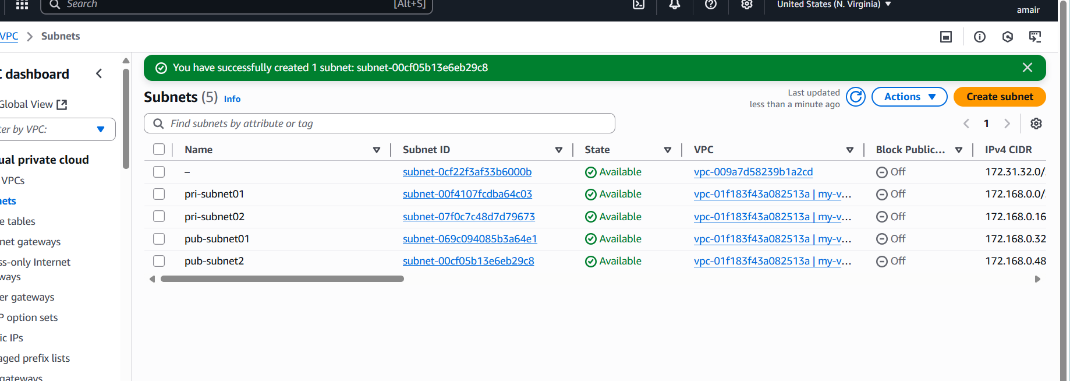
* **Public Subnets:** Can access internet directly (via Internet Gateway). Good for web servers, load balancers.
* **Private Subnets:** No direct internet access. Good for databases, application servers.
  + - 1. **Go to VPC click create vpc**



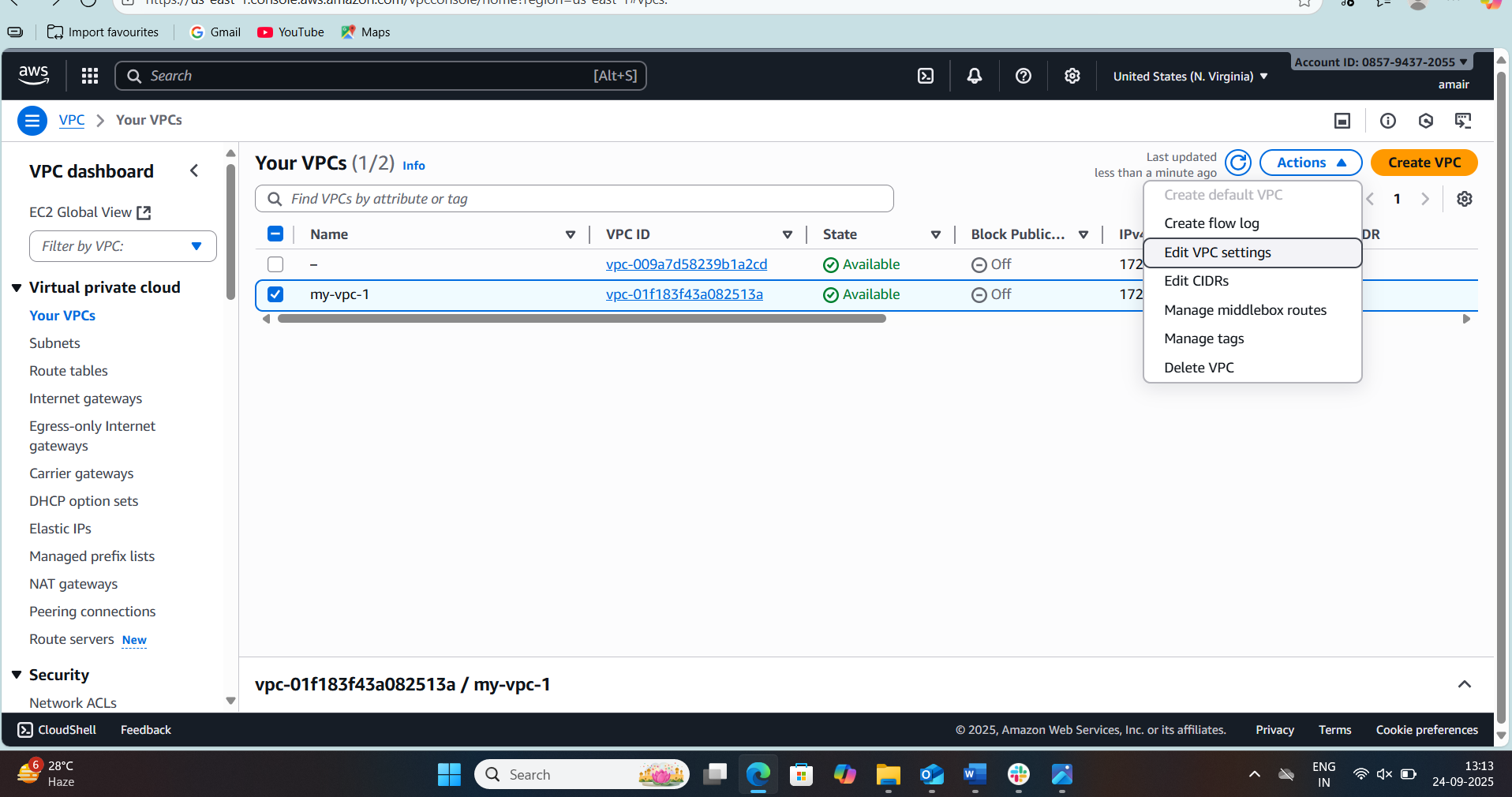




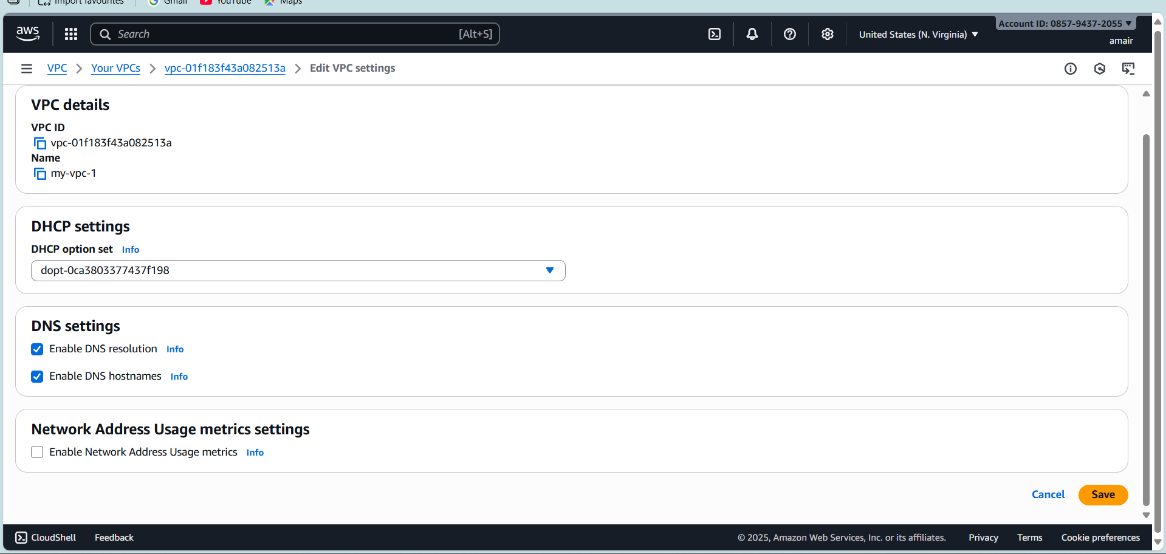
* + - 1. create **4 subnets (2 public, 2 private)** .
* **Public Subnet 1 = 172.168.0.32/28**
* **Public Subnet 2 = 172.168.0.42/28**
* **Private Subnet 1 = 172.168.0.0/28**
* **Private Subnet 2 = 172.168.0.16/28**



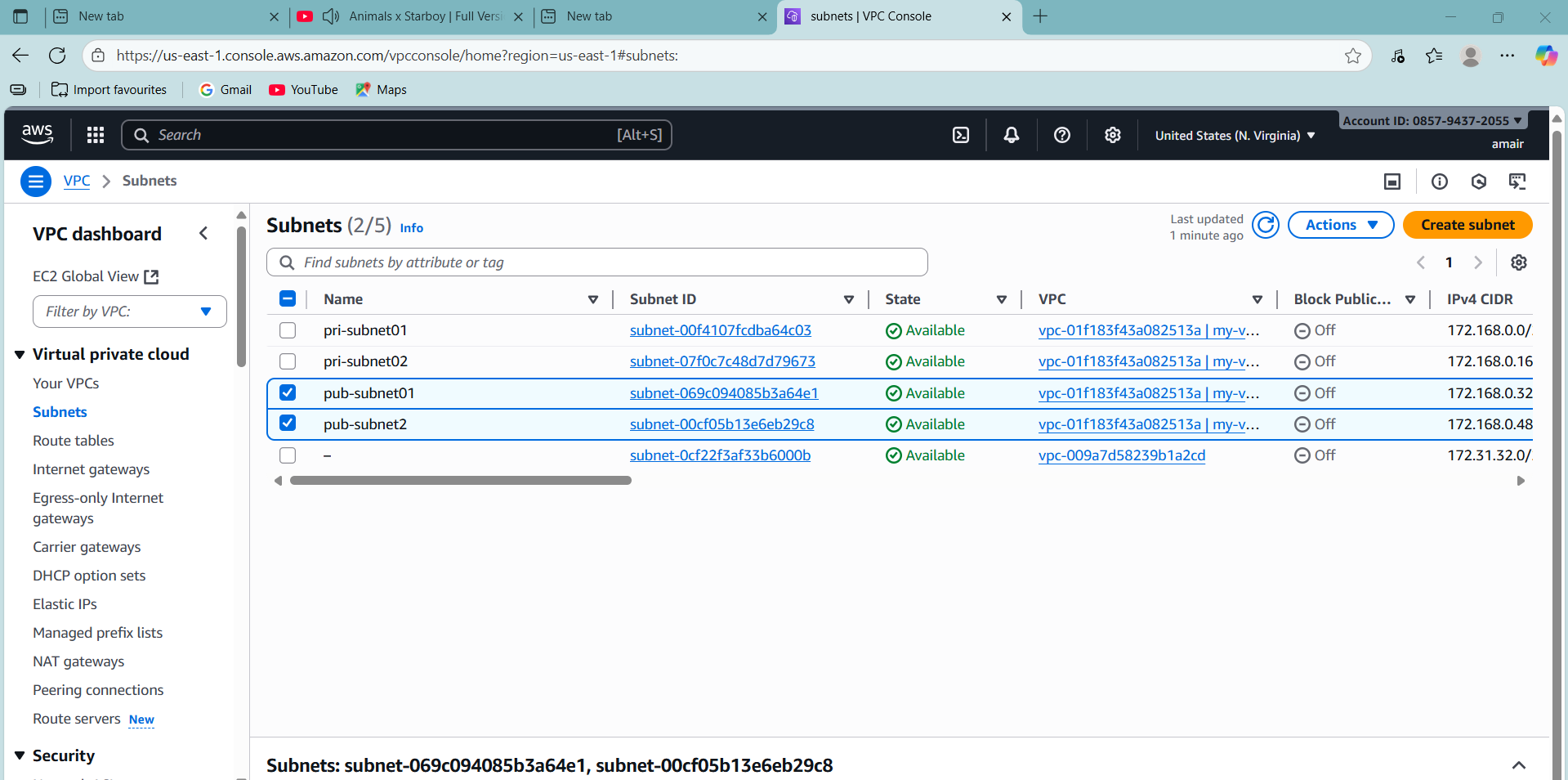
### **Enable DNS Hostname in VPC.**



* + - * + Enable DNS resolution
        + Enable DNS hostnames



### **Enable Auto Assign Public IP in 2 public subnets.**

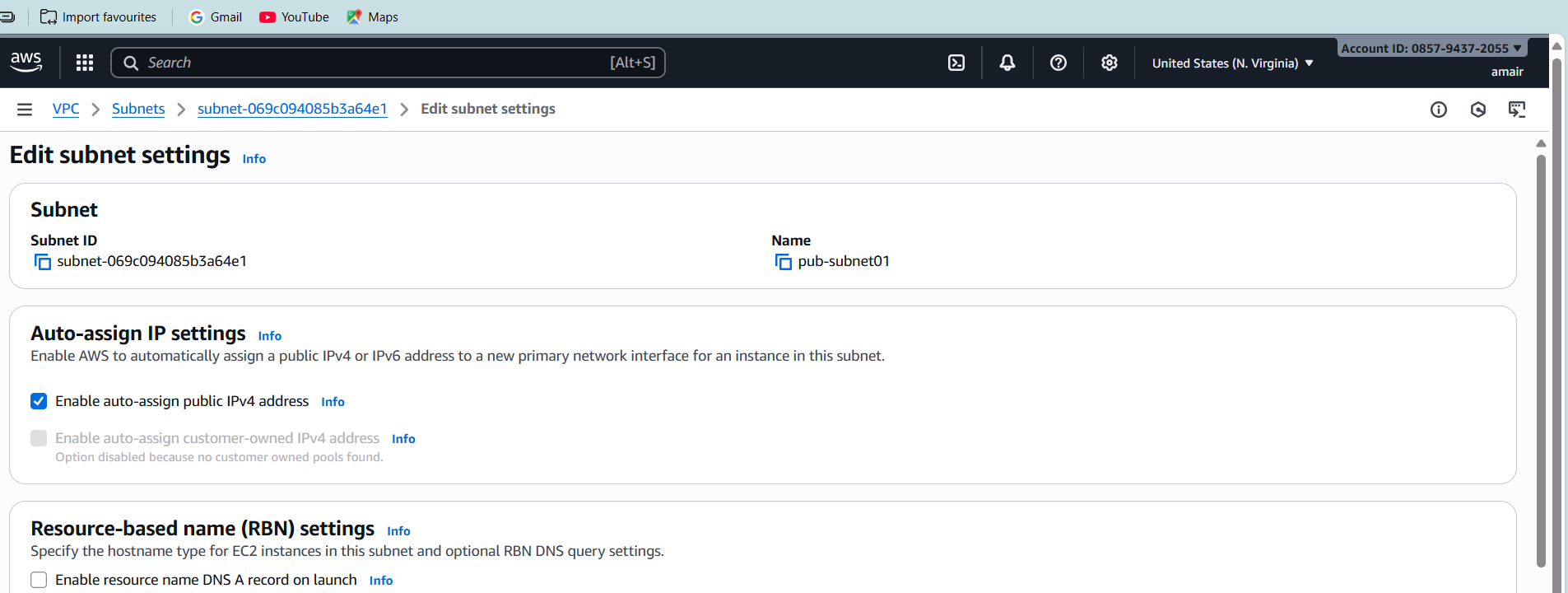


* Open VPC Service

1. Go to AWS Console → VPC service.
2. In the left menu, click Subnets.

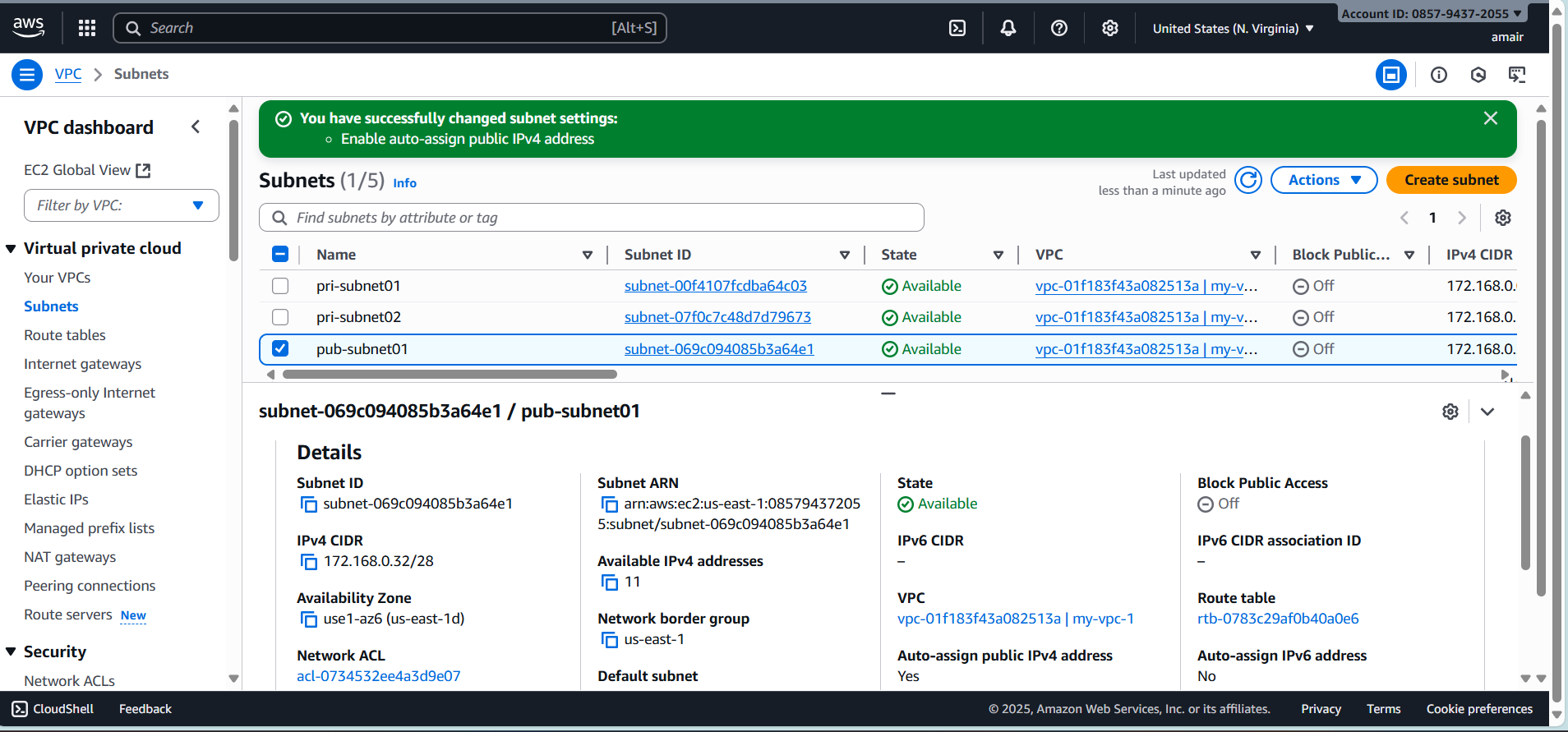
* Step 2: Select Public Subnet

1. From the list, select your first public subnet (e.g., Public-Subnet-AZ1).
2. At the bottom panel, go to the Subnet details tab.



* Step 3: Modify Auto-assign Public IP

1. Click Actions - Edit subnet settings.
2. Look for Auto-assign IP settings.
3. Enable: Auto-assign IPv4.
4. Save changes.

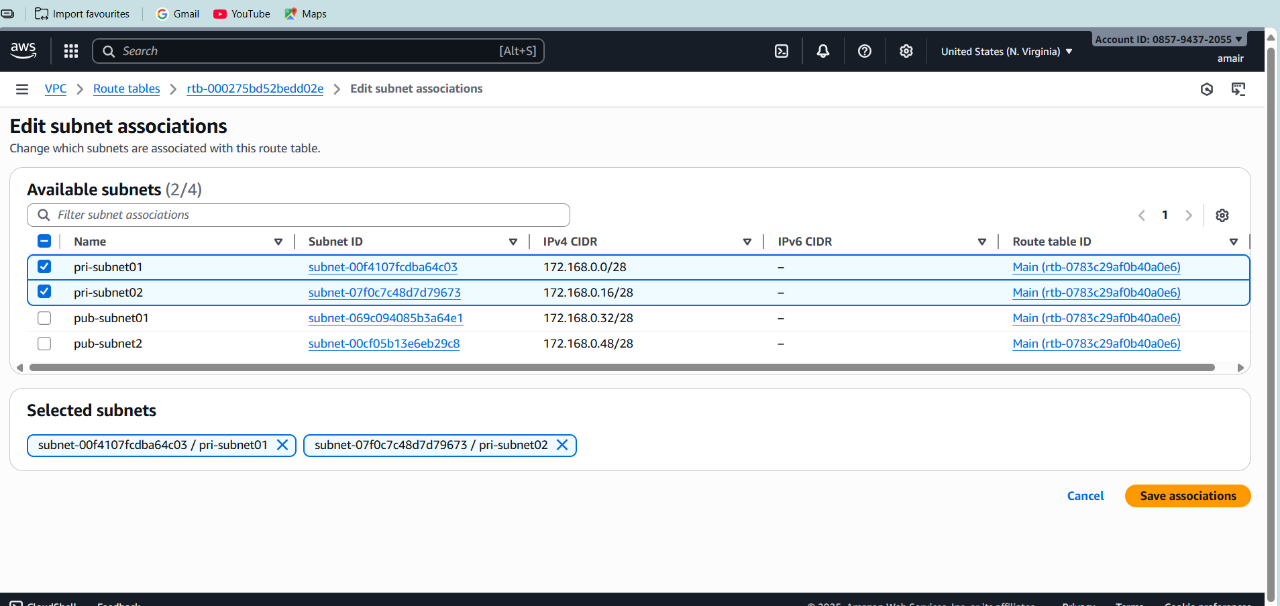


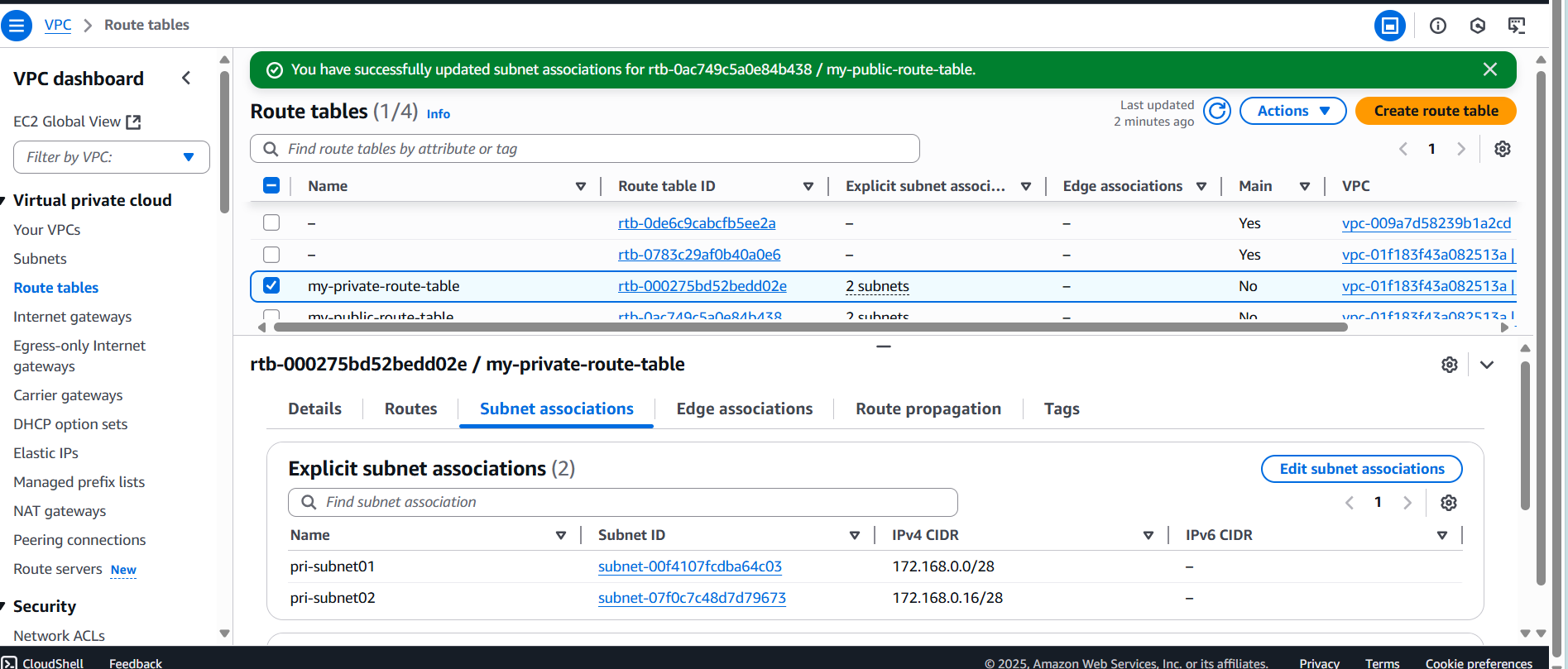
* Same as 2 my subnet-public-2 done changed auto-assgin ipv4 address

1. **Add 2 private subnets in private route table.**



* First, create one route table
* Edit Subnet Associations with Private-RT selected, go to the Subnet associations tab
* Click Edit subnet associations.
* Select your 2 private subnets

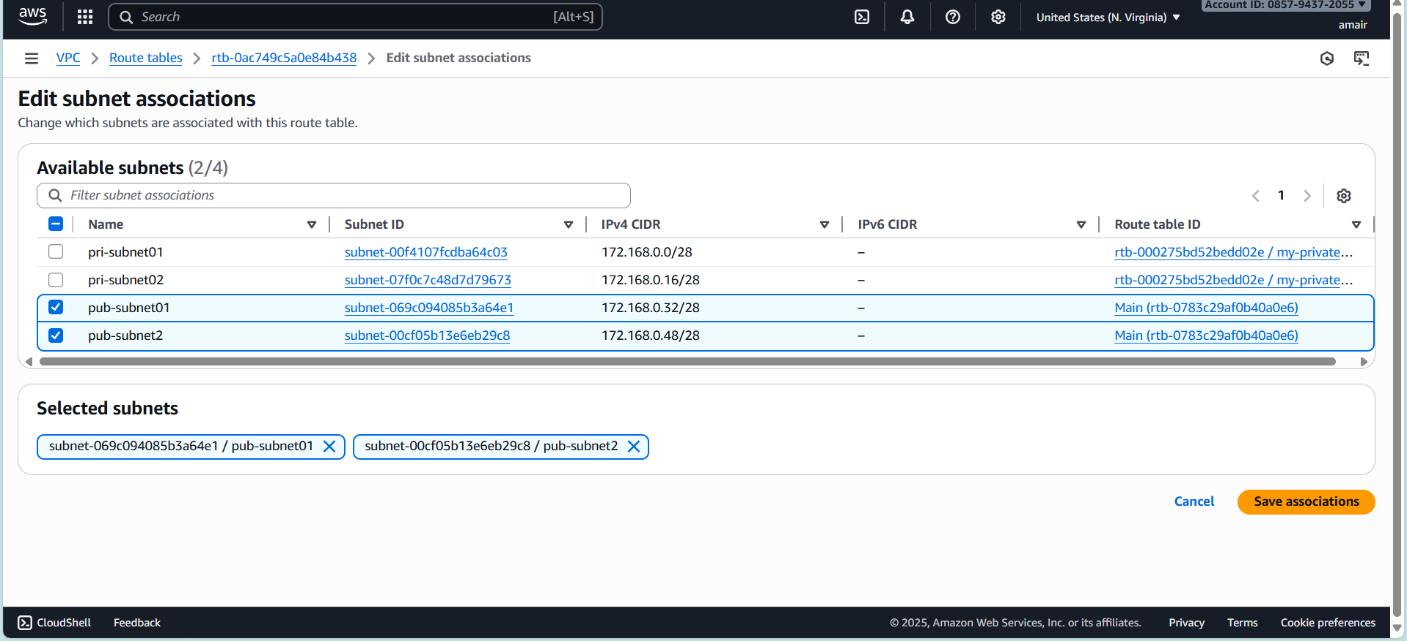


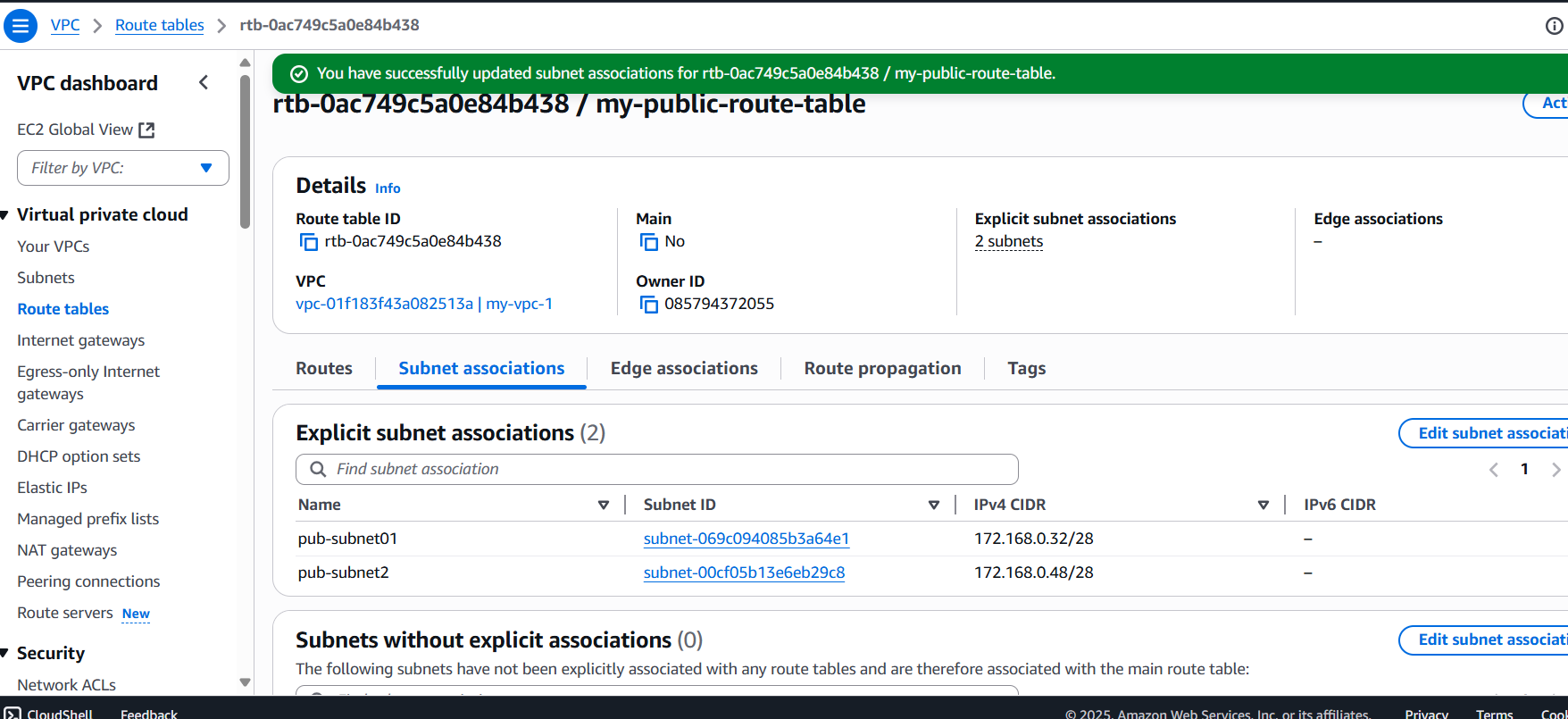


## **Add 2 public subnets in public route table.**

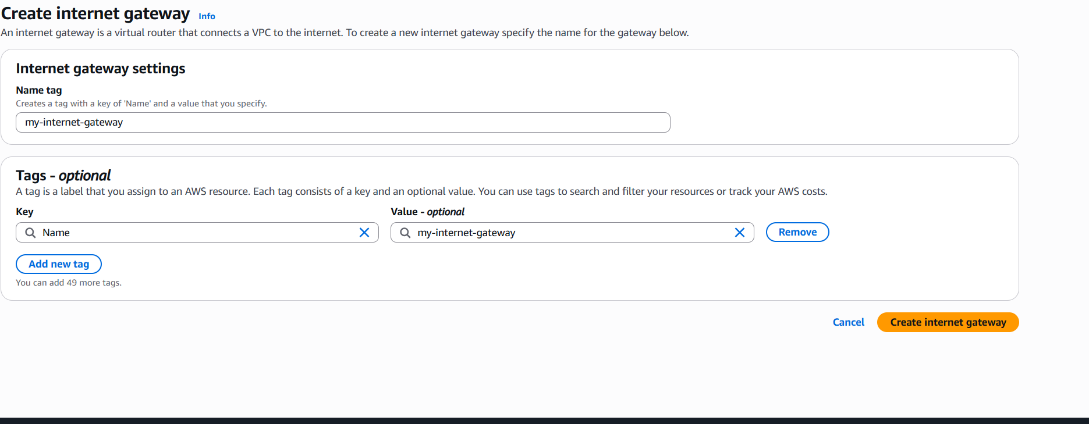


* With the Public Route Table still selected → go to Subnet associations tab.
* Click Edit subnet associations.
* Tick your 2 public subnets (e.g., Public-Subnet-AZ1, Public-Subnet-AZ2).
* Click Save associations.

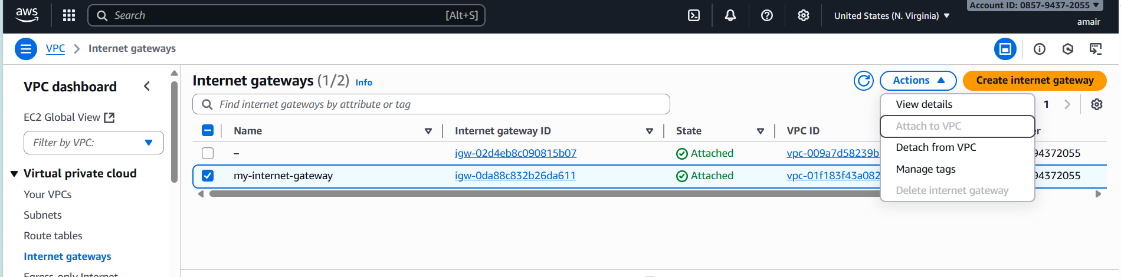


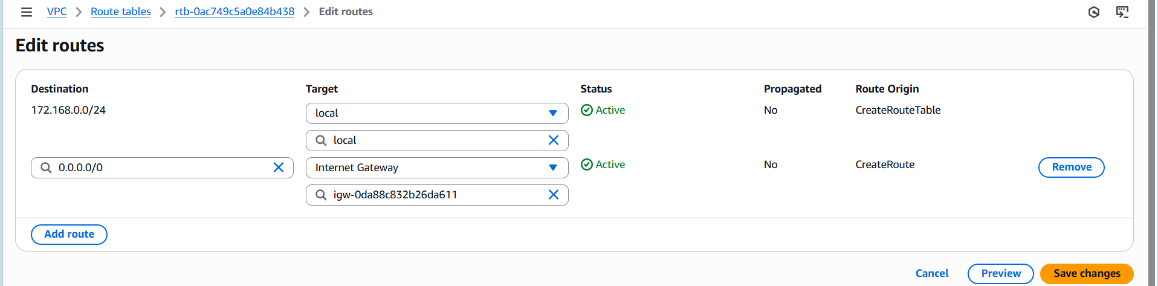


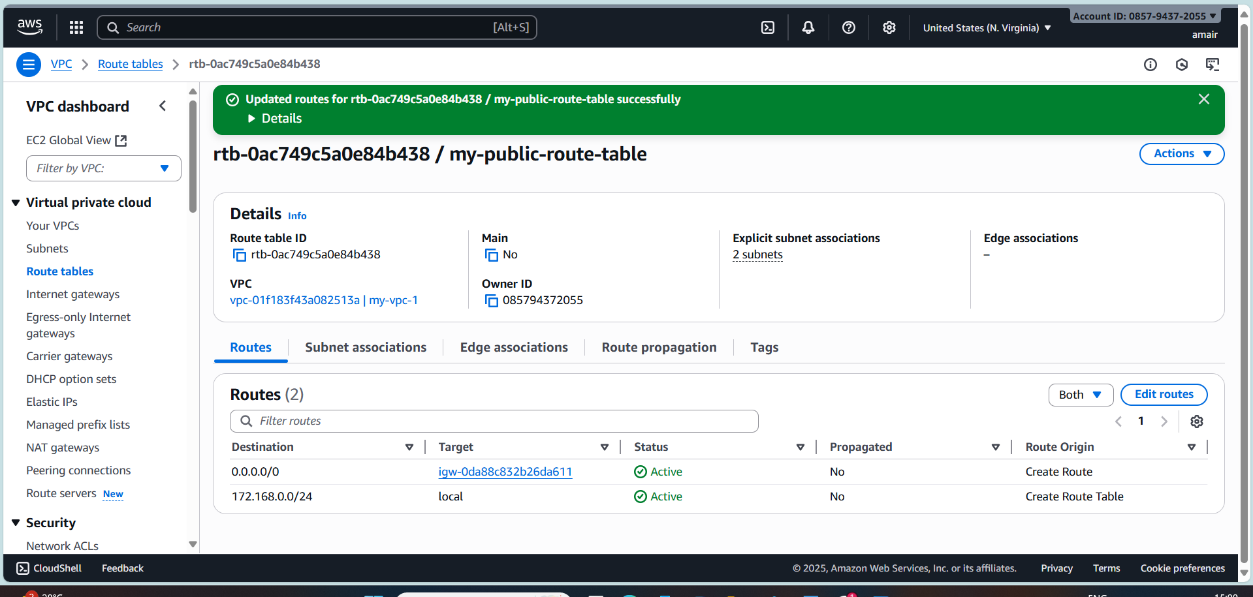
1. **Public route table will have the routes to internet and local.**



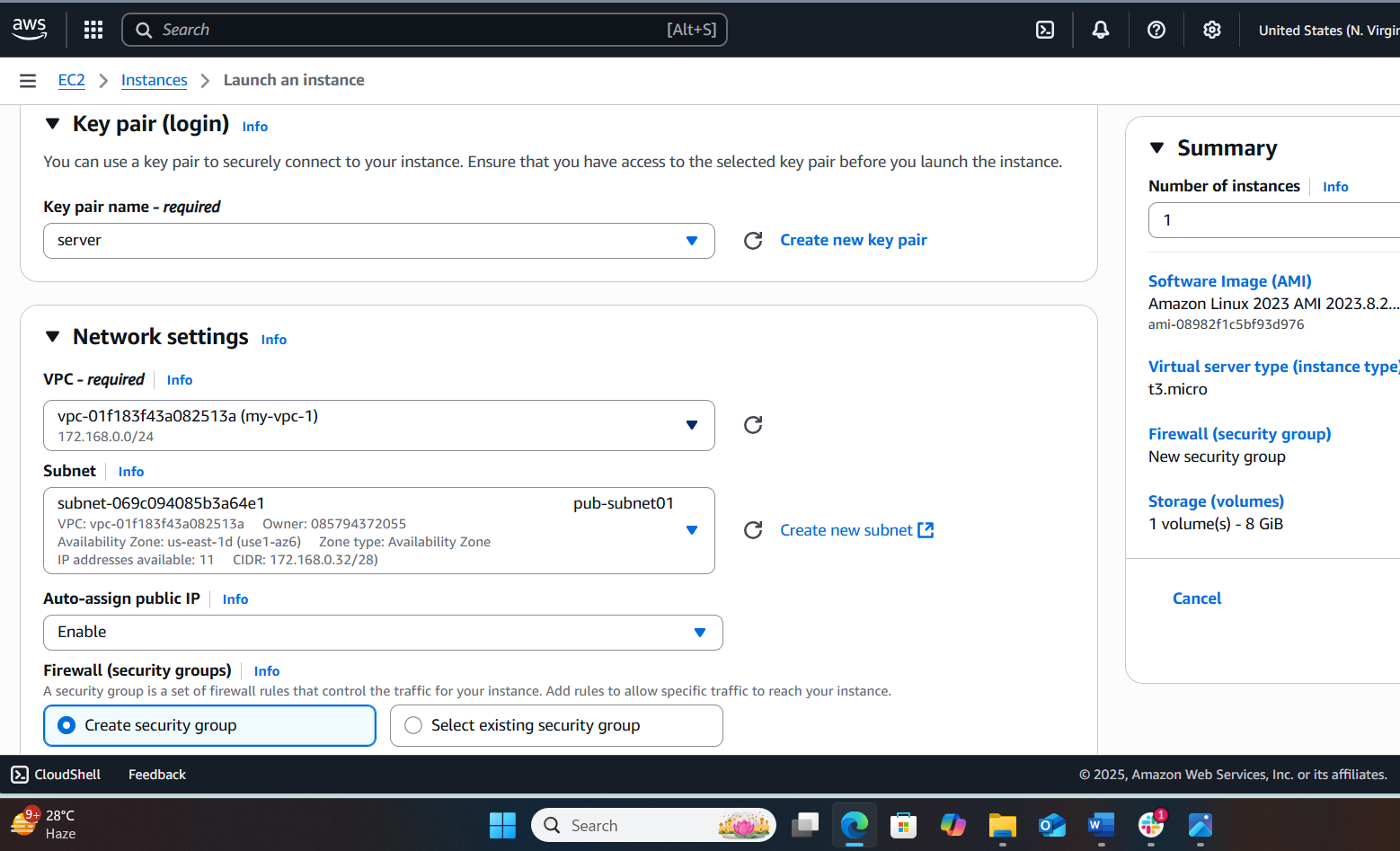
* Add Routes
  1. Destination: 10.0.0.0
* Click Add route.
* Enter:
  1. Destination: 0.0.0.0/0
  2. Target: Select your Internet Gateway (IGW)



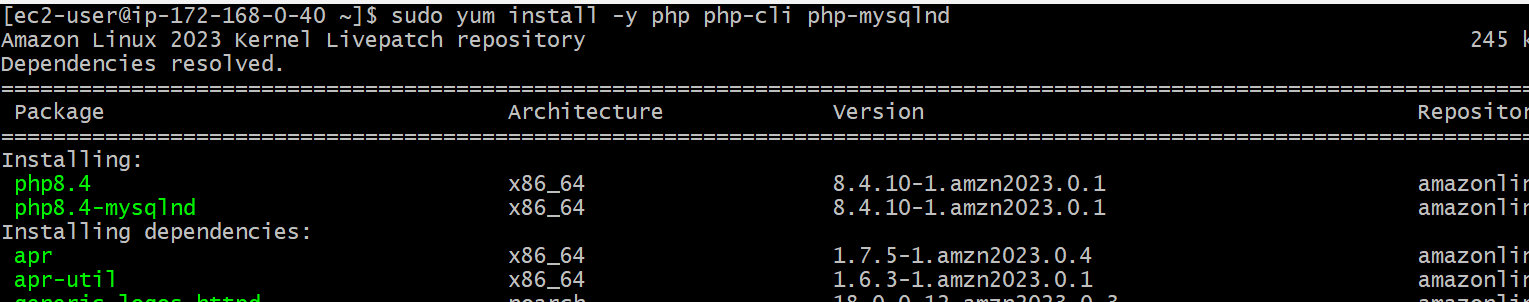




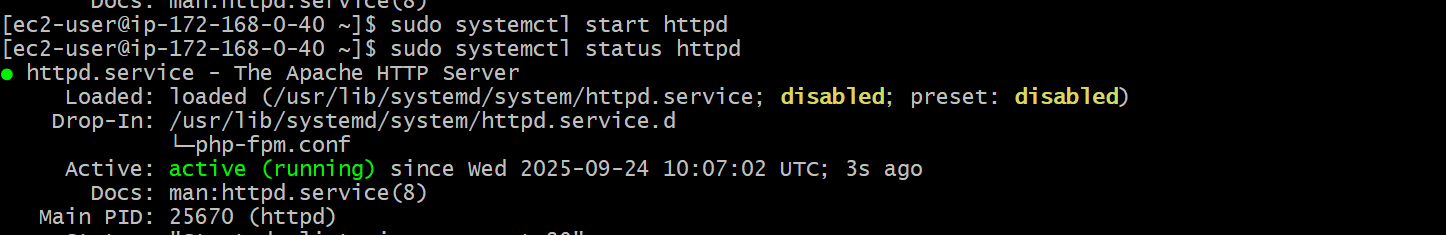
1. **Create EC2 in public subnet with t2.micro and install PHP**



* Create one ec2 instance while creating a choice as our wish, then go to network configuration
* VPC: Select your VPC (My-VPC).
* Subnet: Select one of your Public Subnets (e.g., Public-Subnet-AZ1).
* Auto-assign Public IP: Must be Enabled (so you get a public IP)



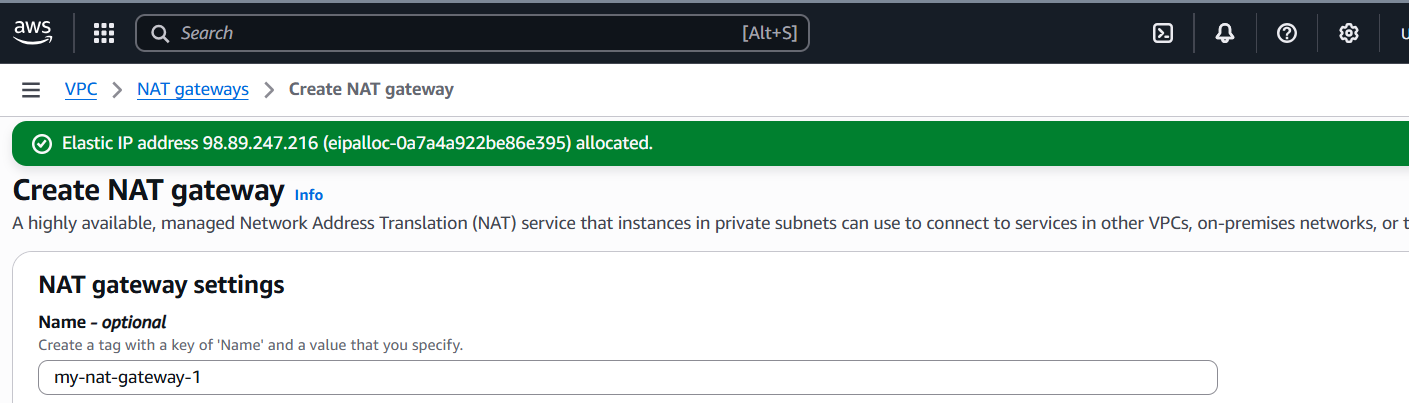
* sudo amazon-linux-extras enable php8.0
* sudo yum install -y php php-cli php-mysqlnd => use this command for install for PHP
* echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/index.php=> execute this command





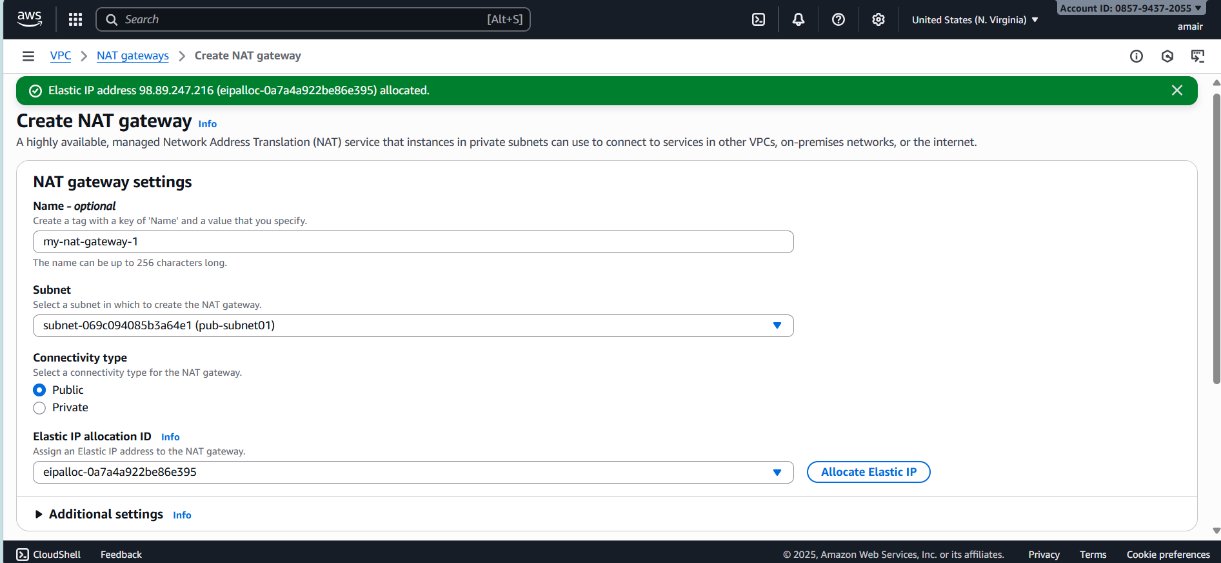
## **Configure NAT gateway in public subnet and connect to private instance**

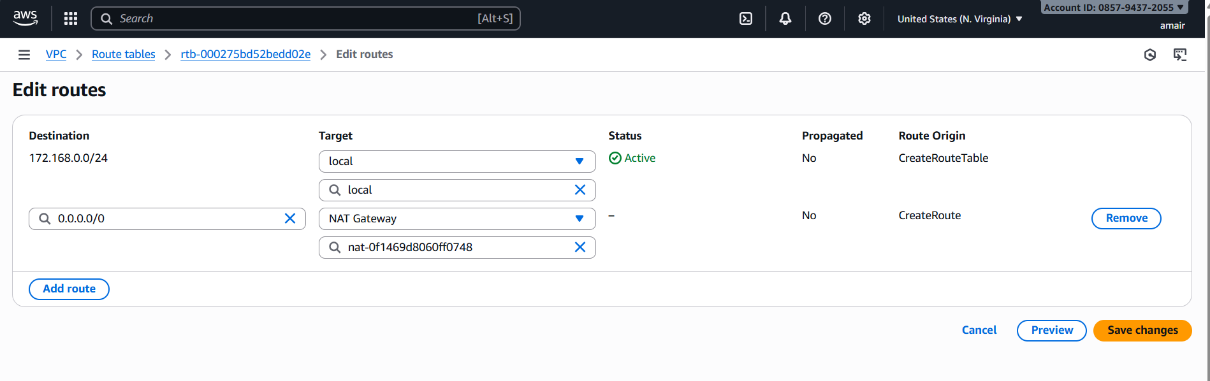
* 1. Go to **VPC service → NAT Gateways.**
  2. Click **Create NAT Gateway.**
  3. Click **Create NAT Gateway.**



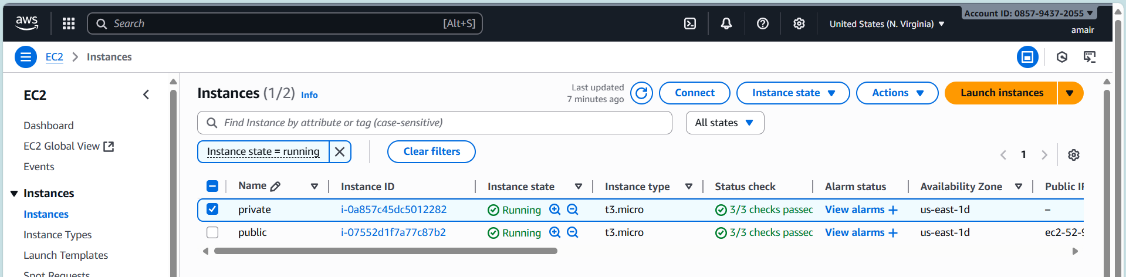
1. Go to Route Tables in VPC.
2. Select your Private Route Table.
3. Edit routes → Add new route:

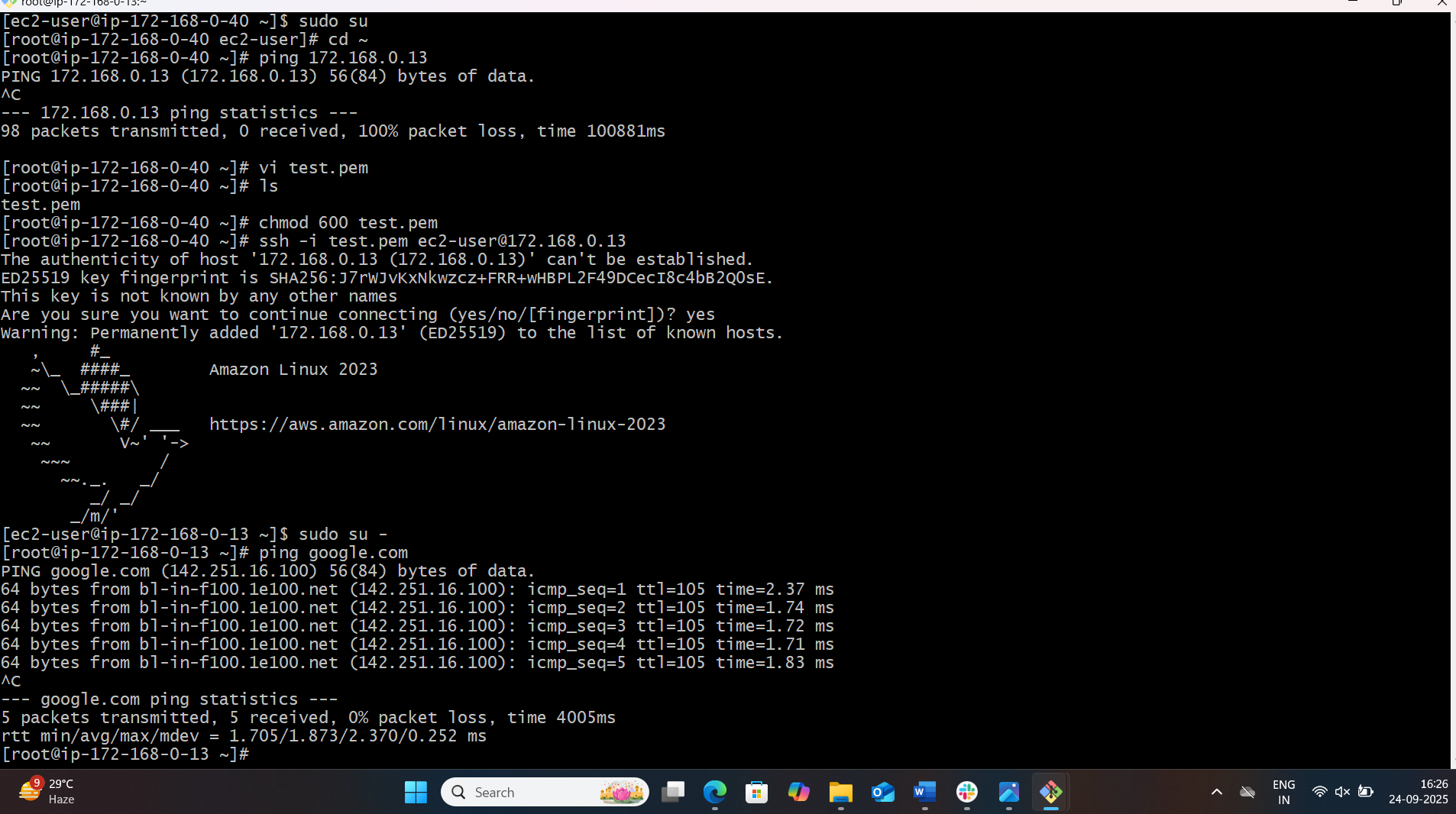
* Now, Private Subnets have internet access through NAT Gateway.



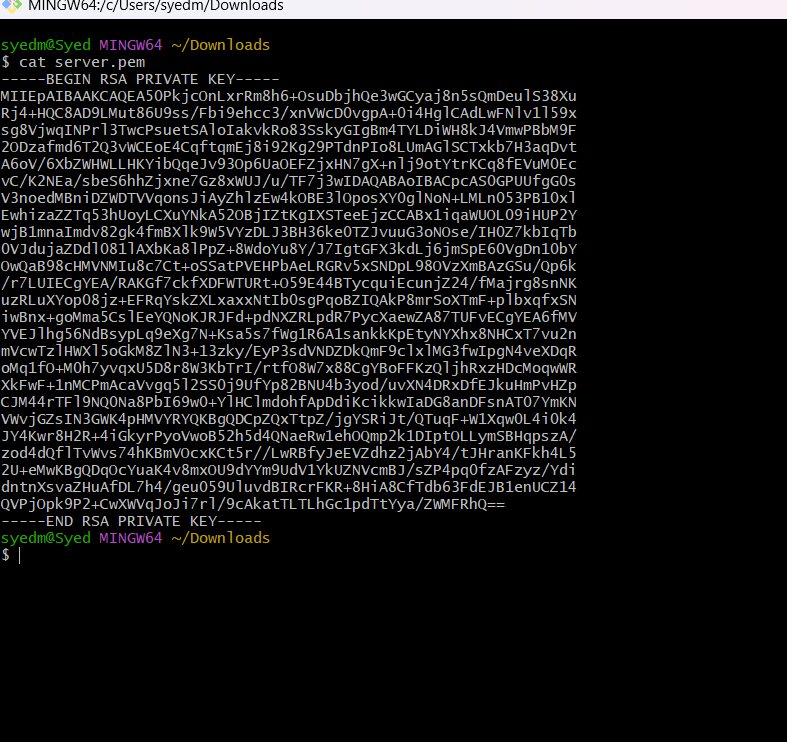


1. Launch an EC2 instance in **Private Subnet**.Security group: allow SSH only from your **Public EC2 (Bastion Host).**
2. SSH into **Public EC2** first:
3. ssh -i my-key.pem ec2-user@<public-ec2-public-ip>





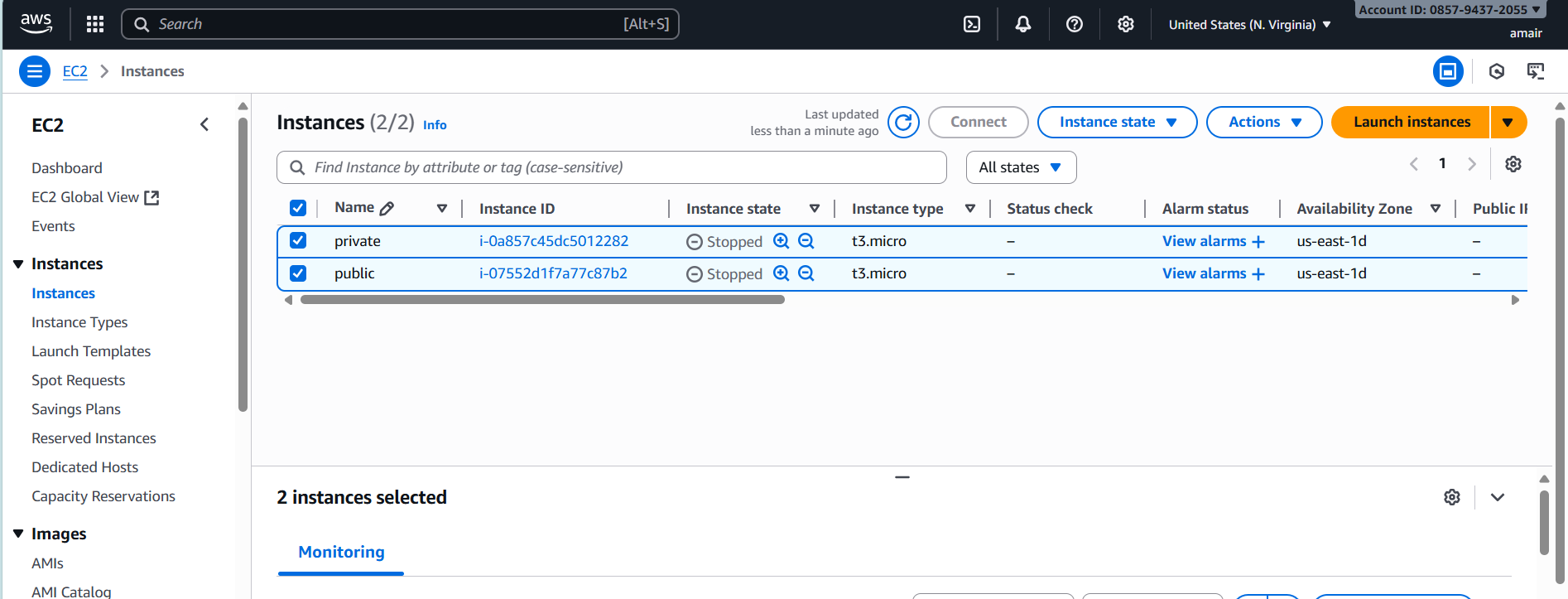
1. From inside Public EC2, SSH into Private EC2 (using private IP):
2. ssh ec2-user@10.0.3.25
3. From Private EC2, test internet:
4. ping google.com



## **Install Apache Tomcat in private EC2 and deploy a sample app.**

## Connect to Private EC2

* Use the Public EC2 (bastion host) to SSH into the private EC2:
* ssh -i my-key.pem ec2-user@<private-ec2-private-ip>

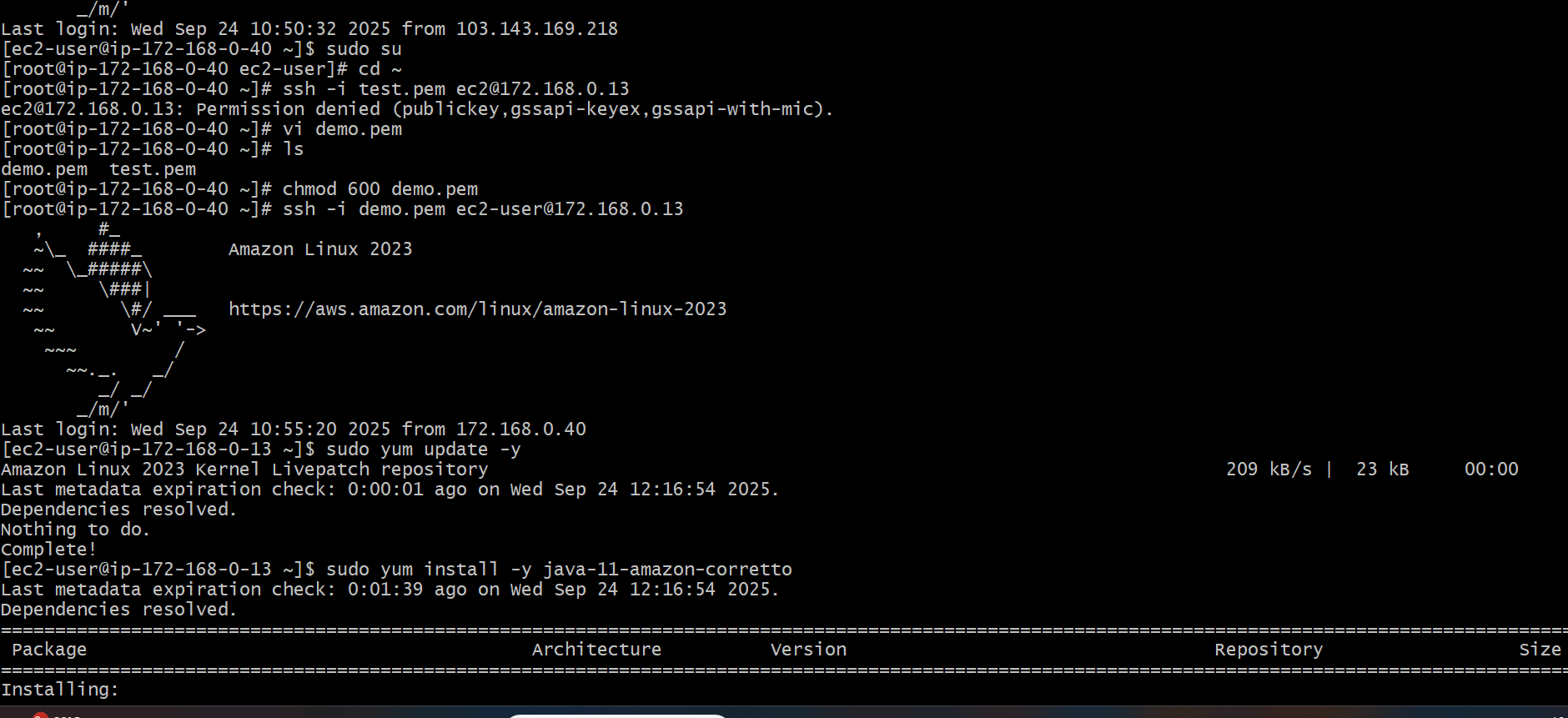


## Update and Install Java

sudo yum update -y

sudo yum install -y java-11-amazon-corretto

* Check version: java -version



## Download and Install Apache Tomcat

* Go to /opt directory:

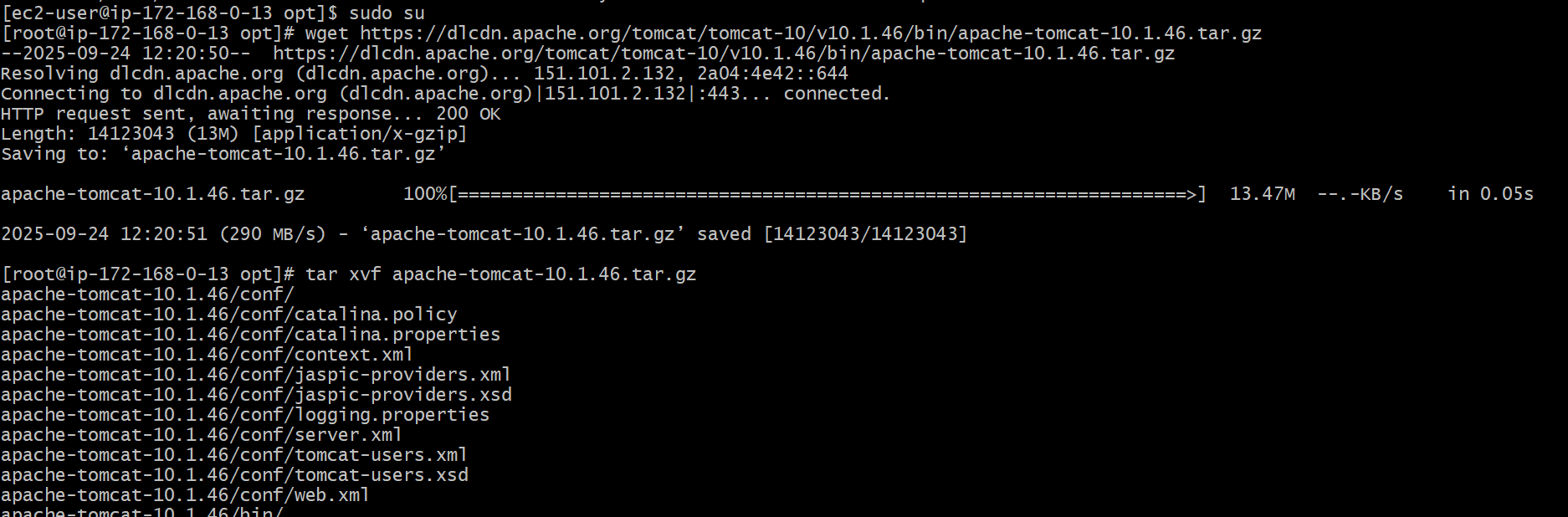
cd /opt

sudo wget <https://downloads.apache.org/tomcat/tomcat->9/v9.0.91/bin/apache-tomcat-10.0.10.tar.gz

* Extract:

sudo tar -xvzf apache-tomcat-9.0.91.tar.gz

sudo mv apache-tomcat-9.0.91 tomcat9



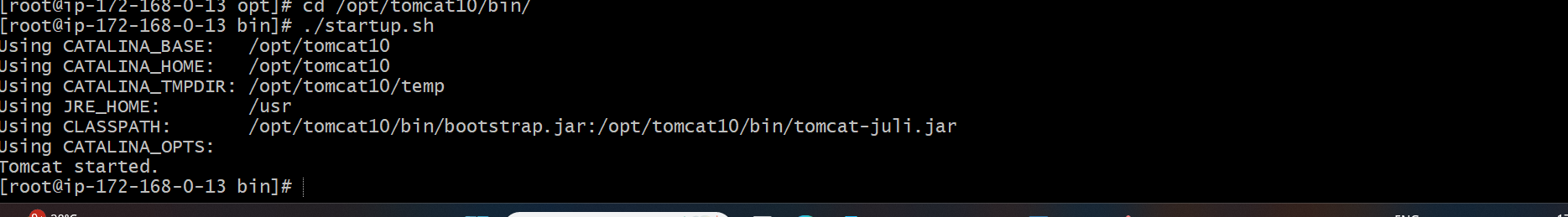
## Start Tomcat

cd /opt/tomcat9/bin

sudo ./startup.sh

* Check logs:

tail -f /opt/tomcat9/logs/catalina.out



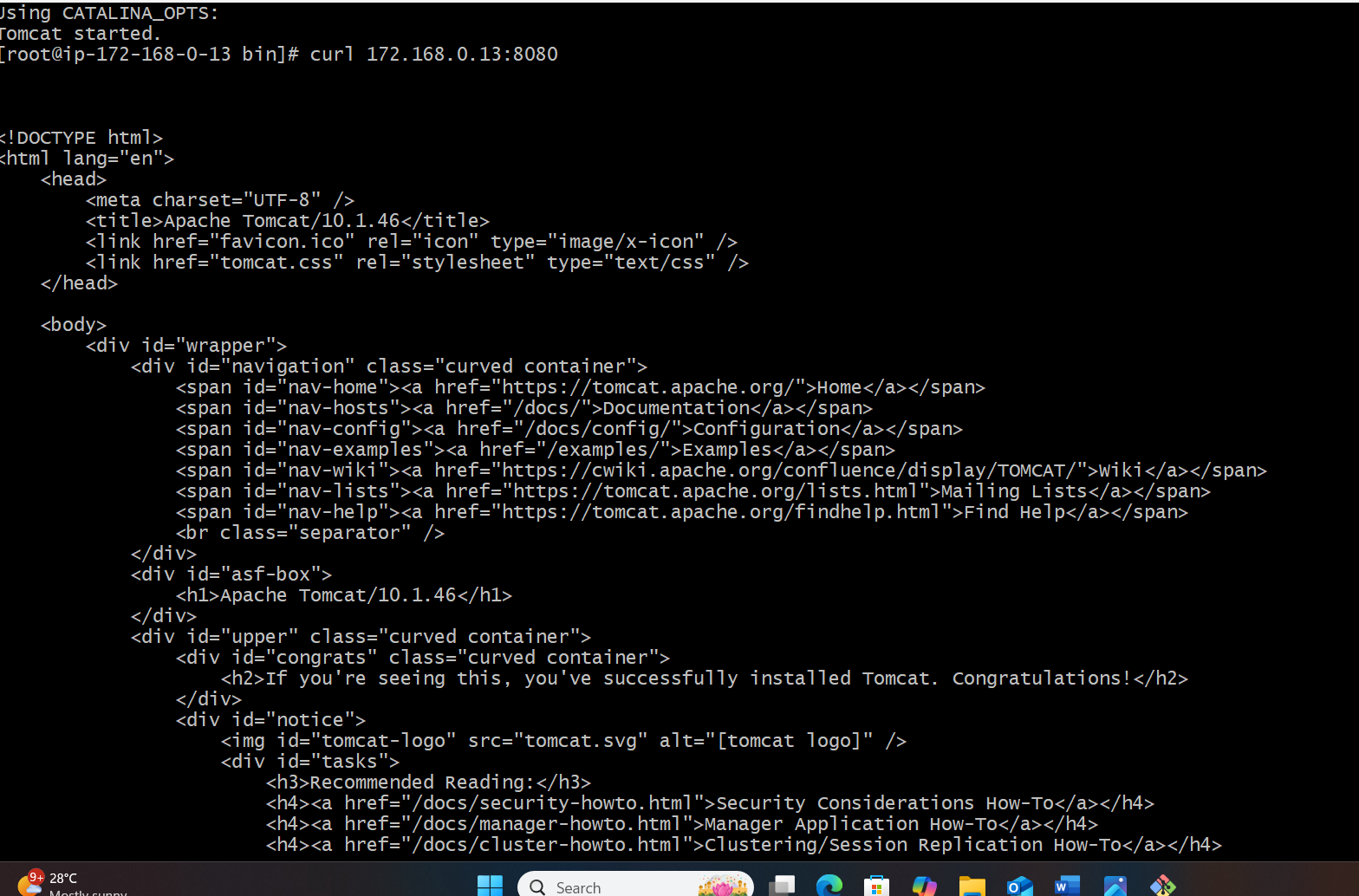
## Deploy a Sample App

1. Download a sample .war file (Java web app):

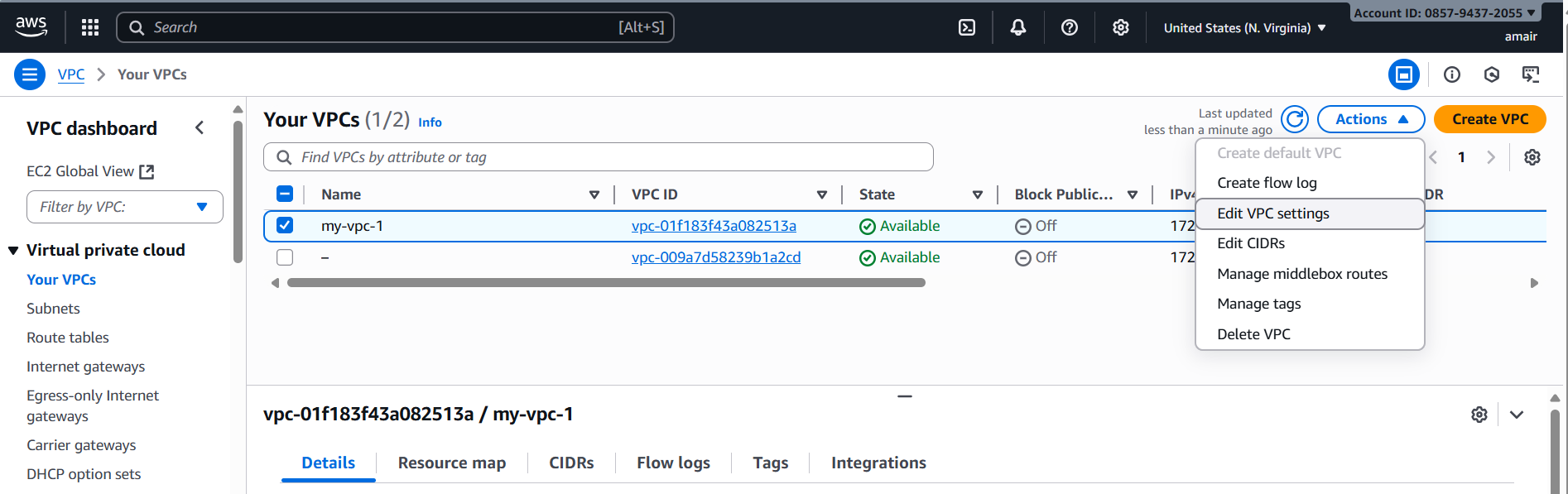
cd /opt/tomcat9/webapps

sudo wget https://tomcat.apache.org/tomcat-9.0-doc/appdev/sample/sample.war

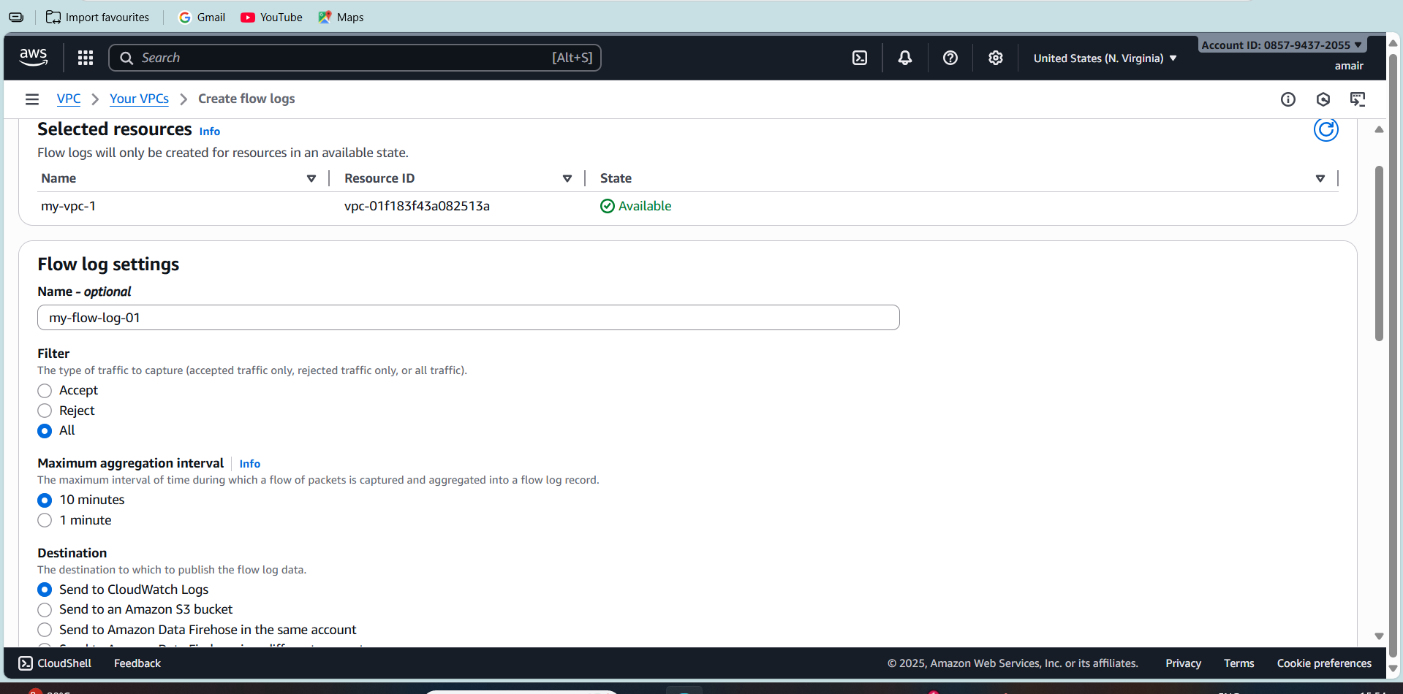
* Use **curl** inside the private EC2:

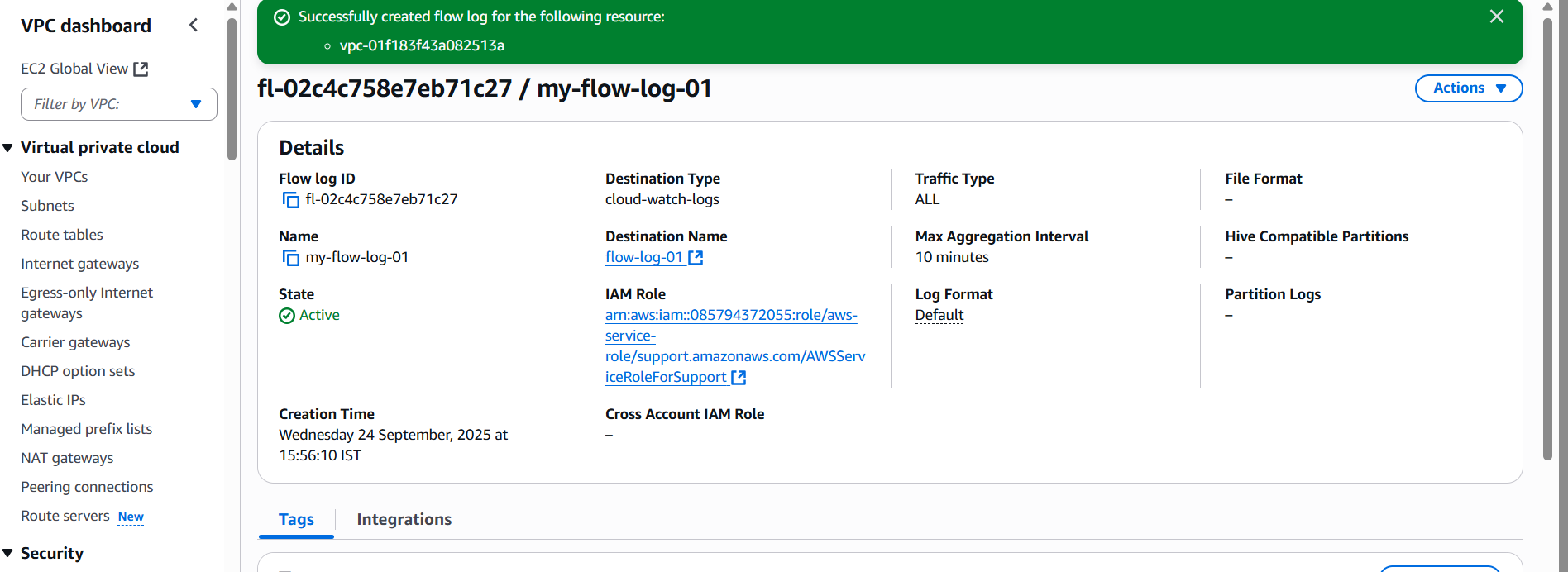


## Configure VPC flow logs and store the logs in S3 and CloudWatch



* Create Flow Log
* With your VPC selected, click Actions → Create flow log.
* Configure:
* Filter: Choose what traffic to log
  + - ALL (both accepted/rejected)
    - ACCEPT (only accepted traffic)
    - REJECT (only rejected traffic)
* Maximum aggregation interval: 1 minute (for detailed logs) or 10 minutes
* Destination log group: Choose Send to CloudWatch Logs





* To Verify Logs
* For CloudWatch Logs:

1. Go to CloudWatch → Logs → Log Groups.
2. Open your log group (VPCFlowLogs).
3. Expand → you’ll see log streams with records like: