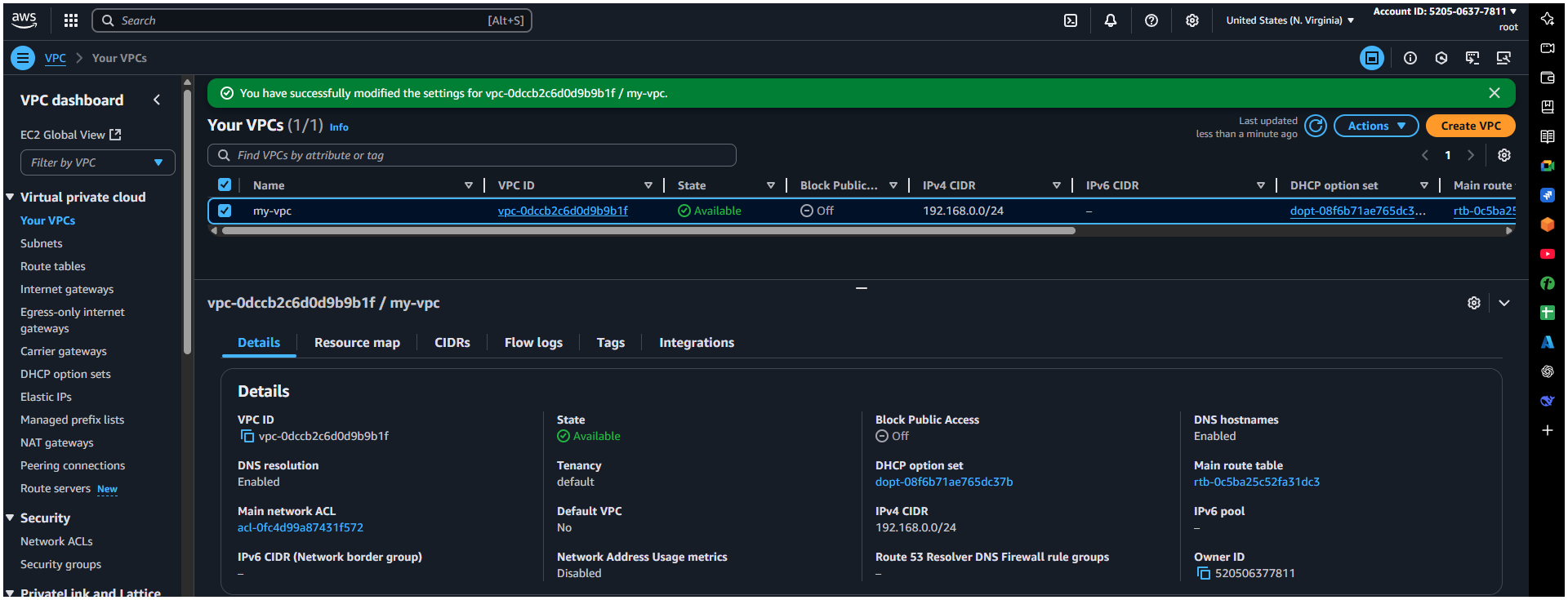
1)Create one VPC in N-Virginia region.  
  
Go to **VPC Dashboard** → **Create VPC**.

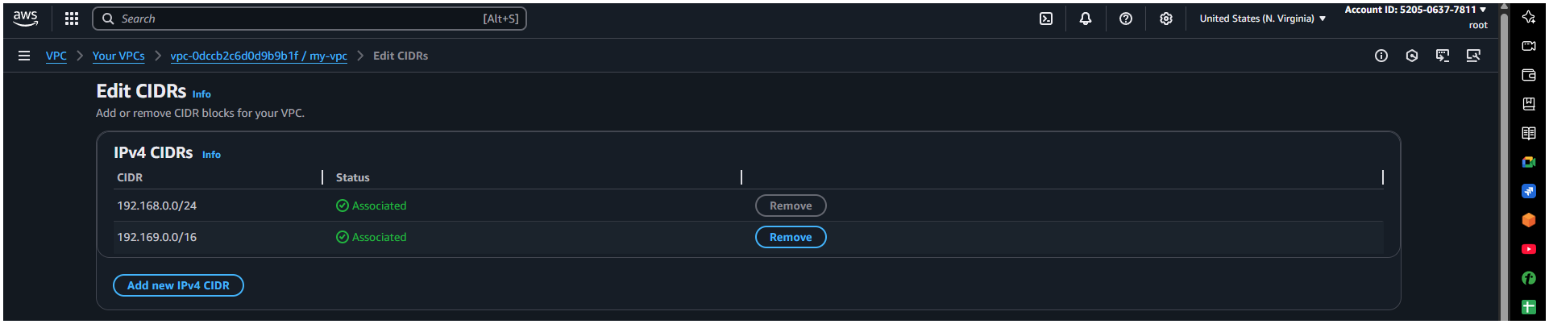
**Name:** **My-VPC**

**IPv4 CIDR block:** **10.0.0.0/16**

Leave IPv6 unselected (optional).

**Tenancy:** Default.

Click **Create VPC**.

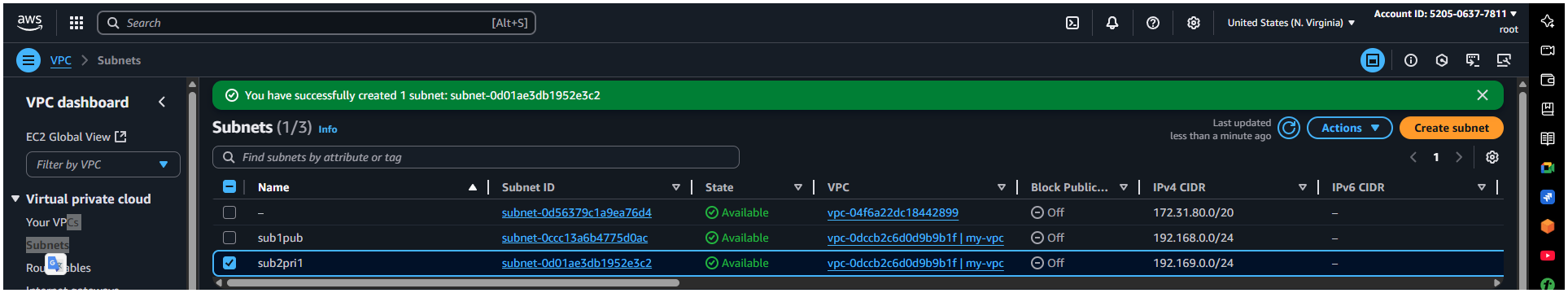
Here I have add two CIDR range access to any specific one to overcome overlap

2)Create two subnets. One Public subnet and one private subnet.

Public subnet → directly accessible from the internet.

Private subnet → no direct internet access, safer for backend services.

Go to Subnets → Create subnet.

Select VPC = My-VPC.

Subnet 1 (Public):

Name: sub1pub1

AZ: us-east-1a

IPv4 CIDR: 192.169.0.0/24

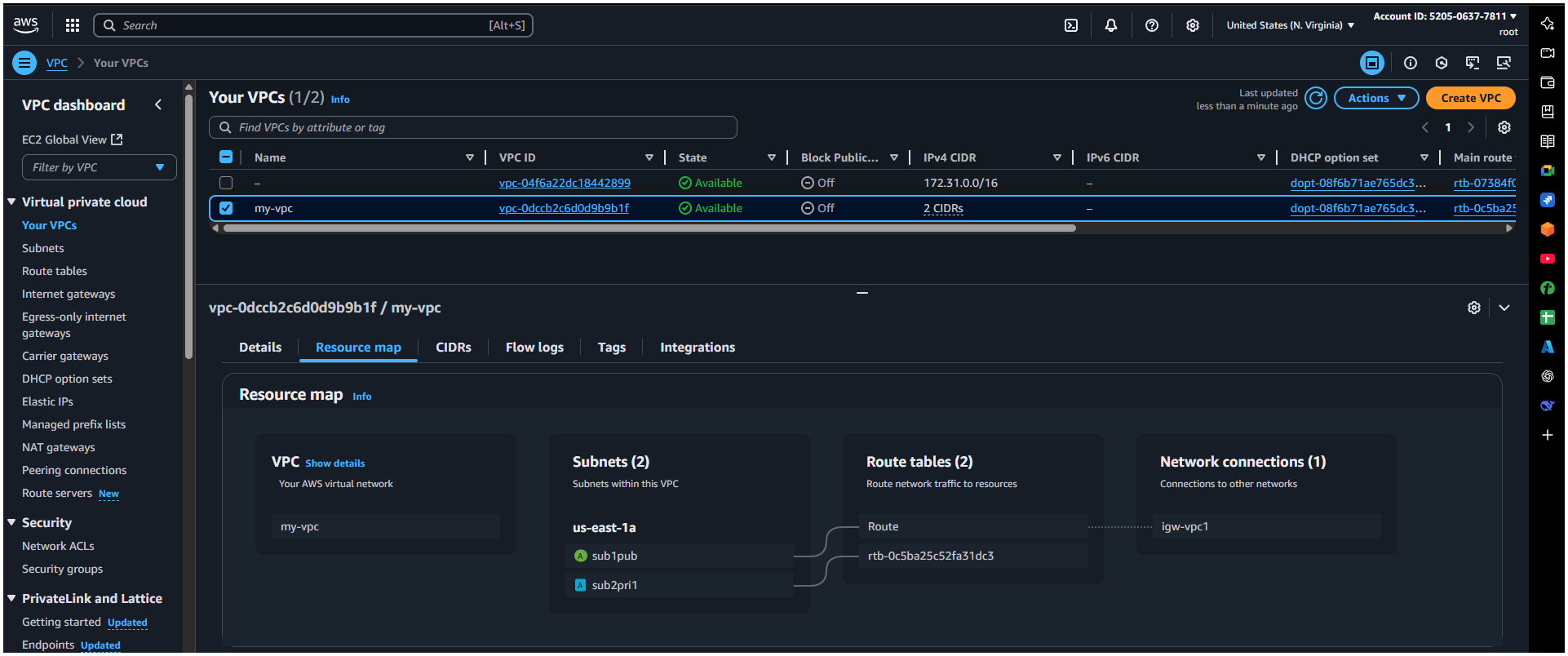
Subnet 2 (Private):

Name: sub2pri1

AZ: us-east-1b

IPv4 CIDR: 192.169.0.0/16

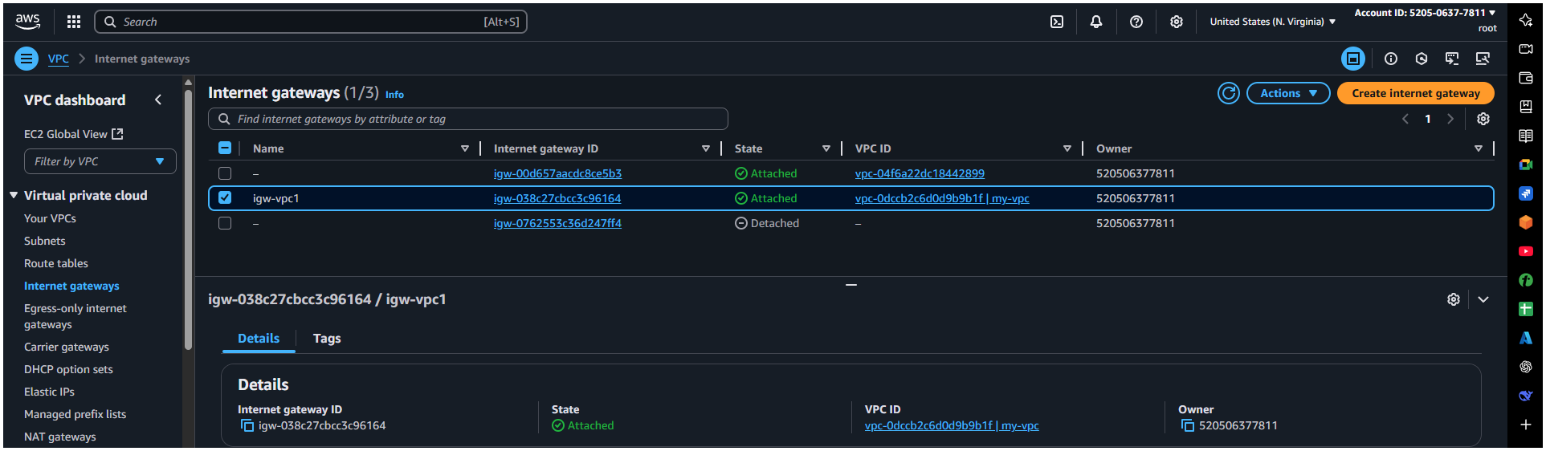
Click Create subnet.

.

🡪Provide the IGW to the vpc.  
Why? Lets public resources talk to the internet.

Go to Internet Gateways → Create internet gateway.

Name: My-IGW

Click Attach to VPC → select My-VPC.

3) Create One public RT and one private RT.  
   
 Why?--🡪 Controls traffic routing inside the VPC.

Public Route Table:

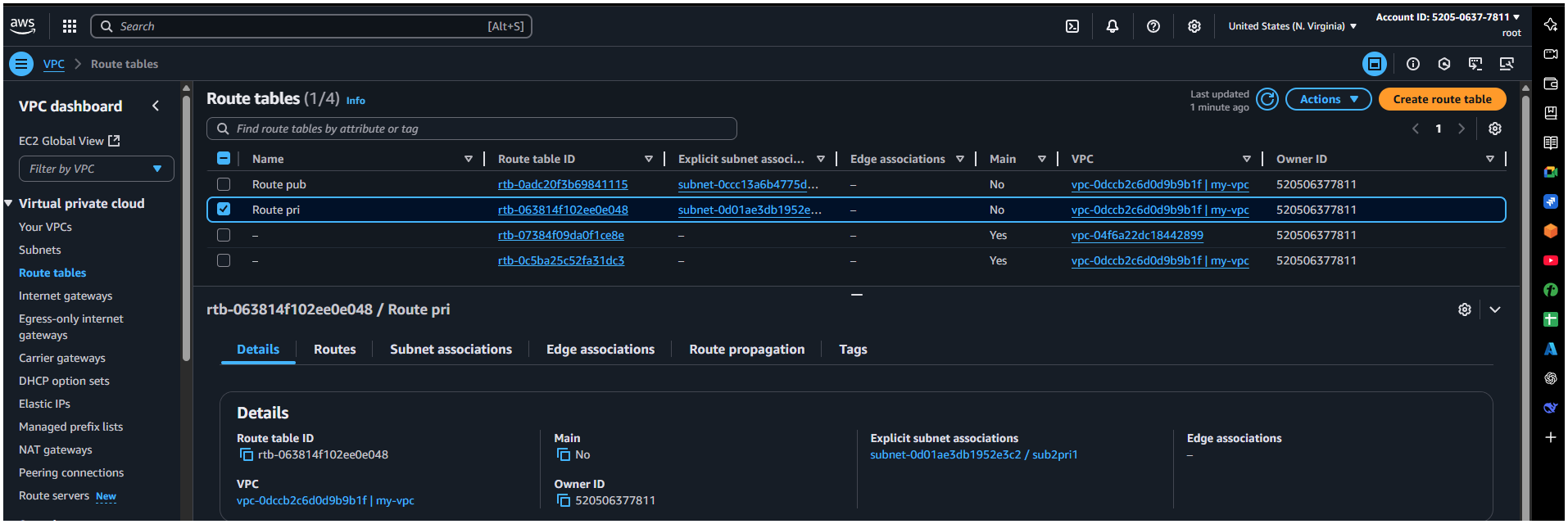
Name: Public-RT

VPC: My-VPC

Edit routes → Add:

Destination: 0.0.0.0/0

Target: Internet Gateway (My-IGW)

Subnet associations → Associate with Public-Subnet.

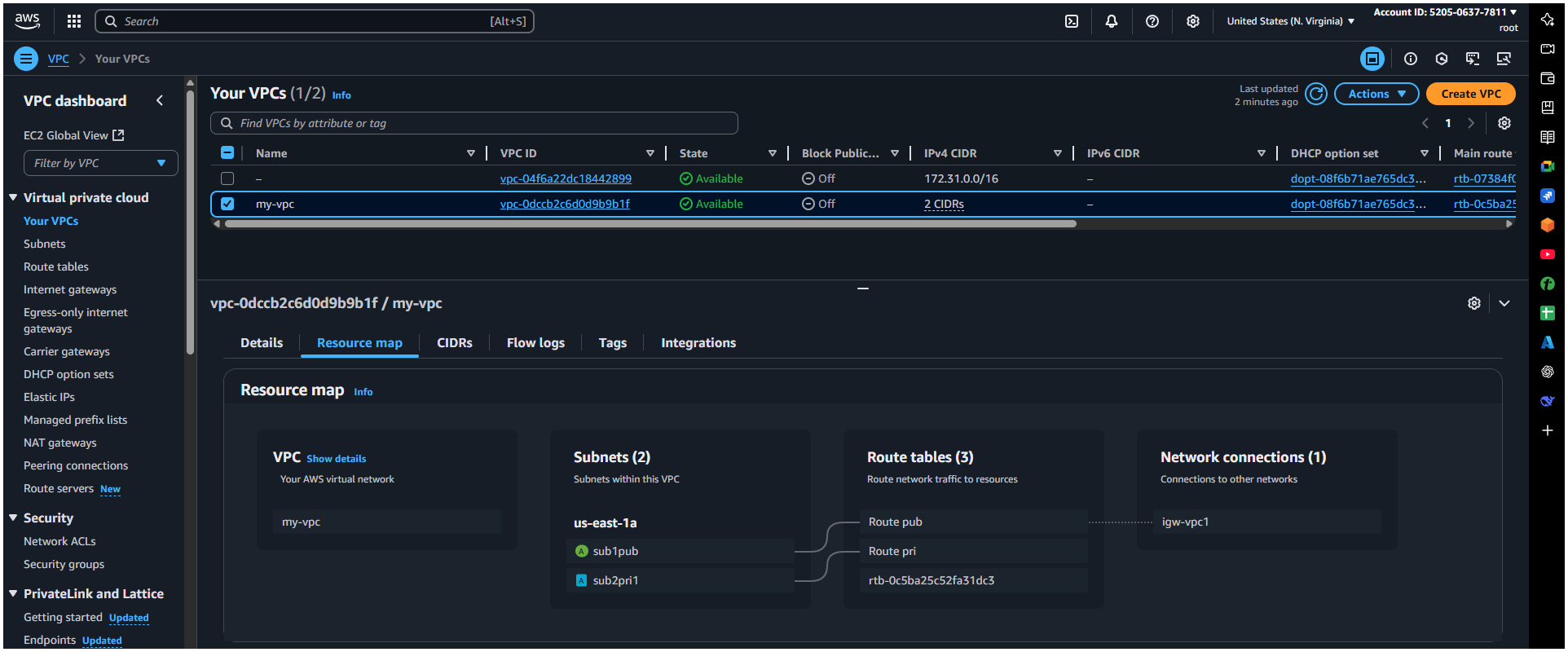
Private Route Table:

Name: Route pri

VPC: My-VPC

Keep default route (local only for now).

Associate with Private-Subnet.



4) Deploy NAT gateway on public subnet and attach the NAT gatewat to private subnet.  
  
Why? Lets private instances download updates from internet without being directly exposed.

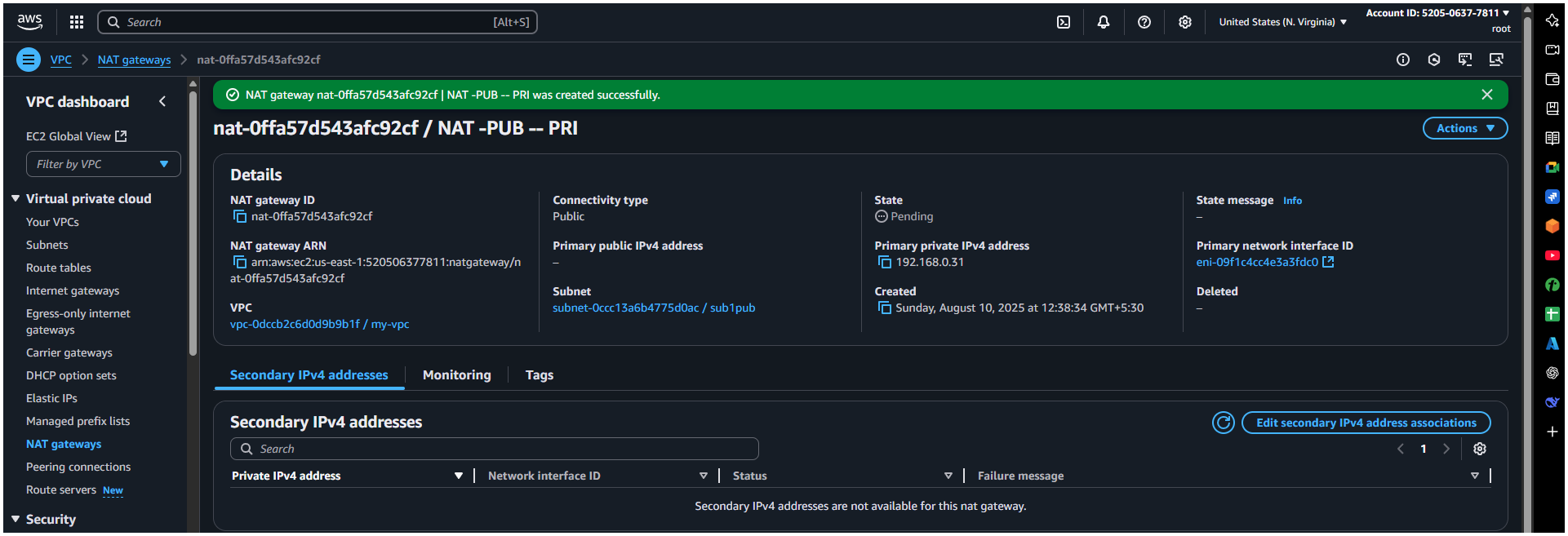
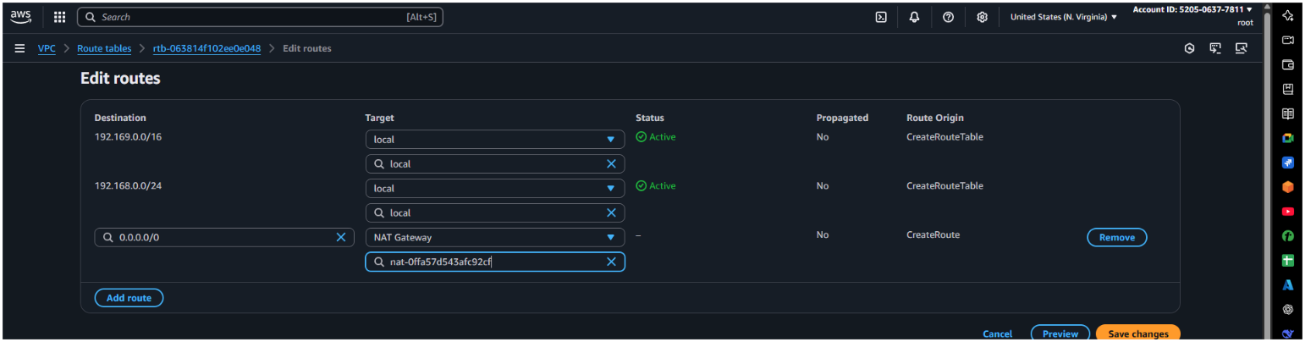
Go to NAT Gateways → Create NAT Gateway.

Subnet: Public-Subnet

Allocate new Elastic IP.

Click Create NAT Gateway.

Edit Private-RT:

Add route: Destination 0.0.0.0/0 → Target: NAT Gateway

5) Create Two instances, one in public subnet and one in private subnet.  
Why?--> Our servers to run Apache.

Public Instance:

Name: Public-EC2

AMI: Amazon Linux 2

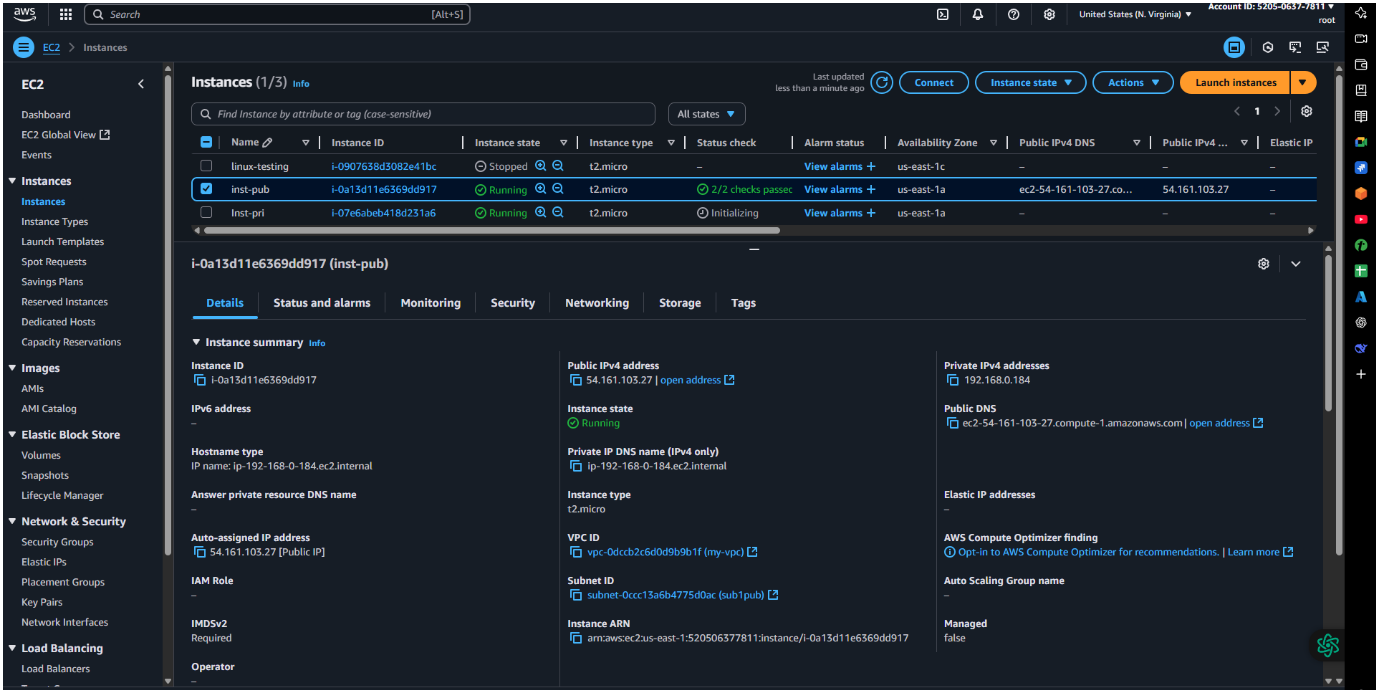
Instance type: t2.micro

Network: My-VPC

Subnet: Public-Subnet

Auto-assign Public IP: Enable

Security group: Select existing security group



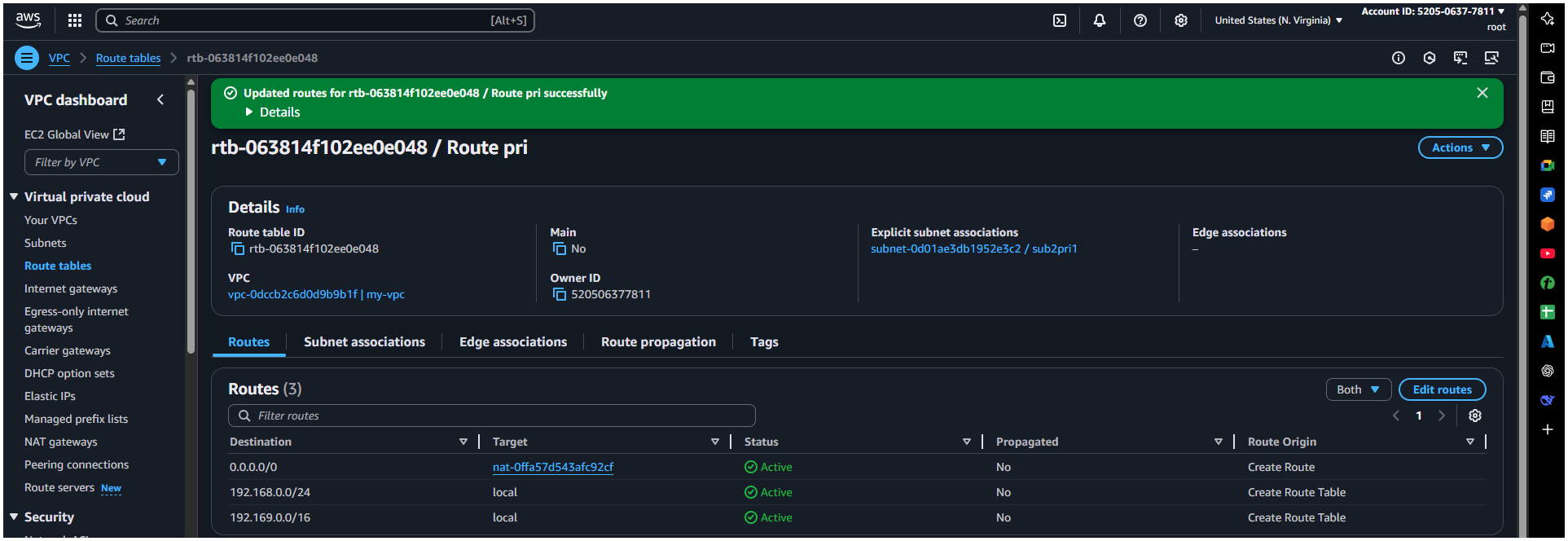
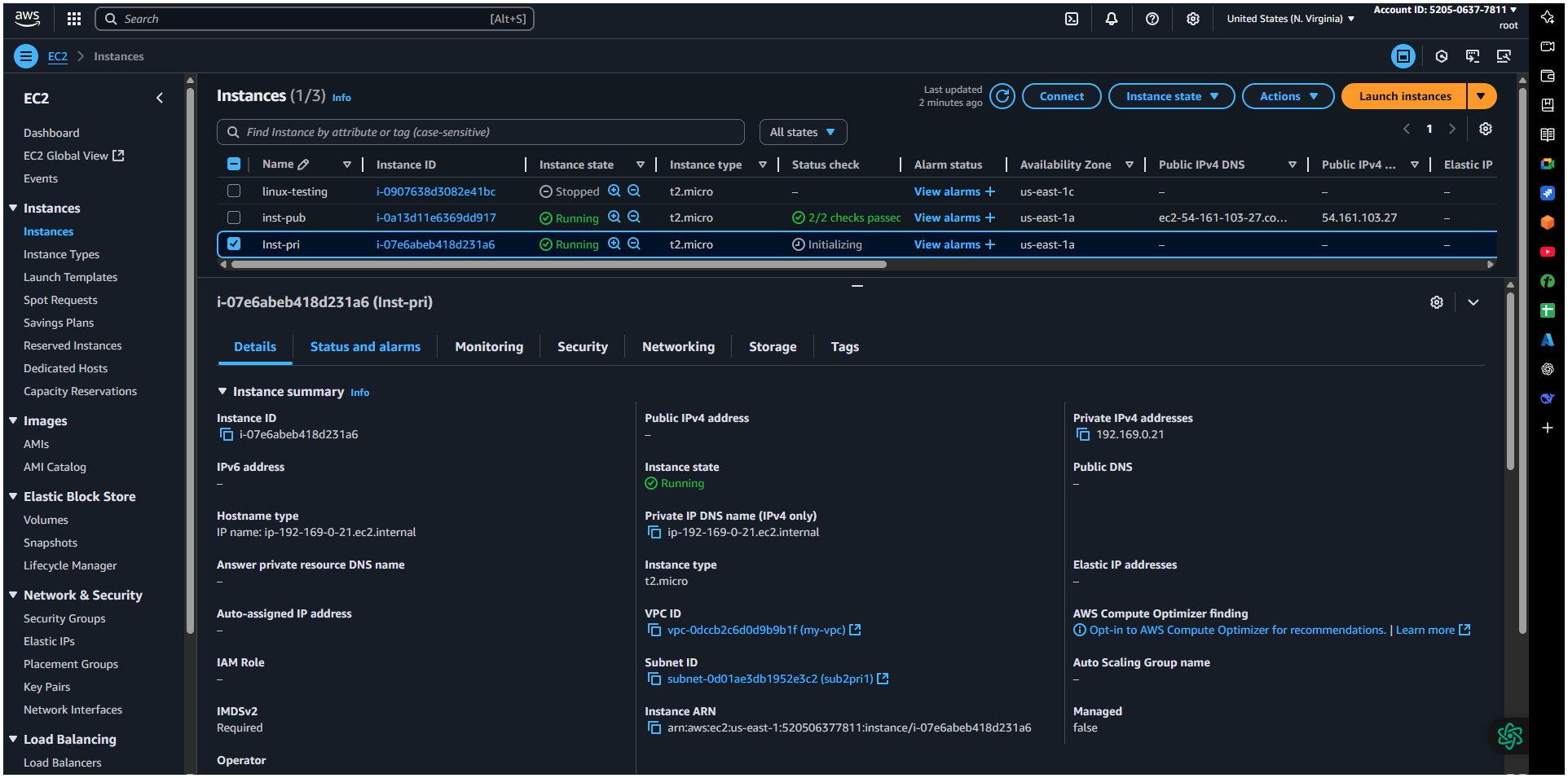
Launch with key pair.  
Private Instance:

Name: Private-EC2

Same AMI & type.

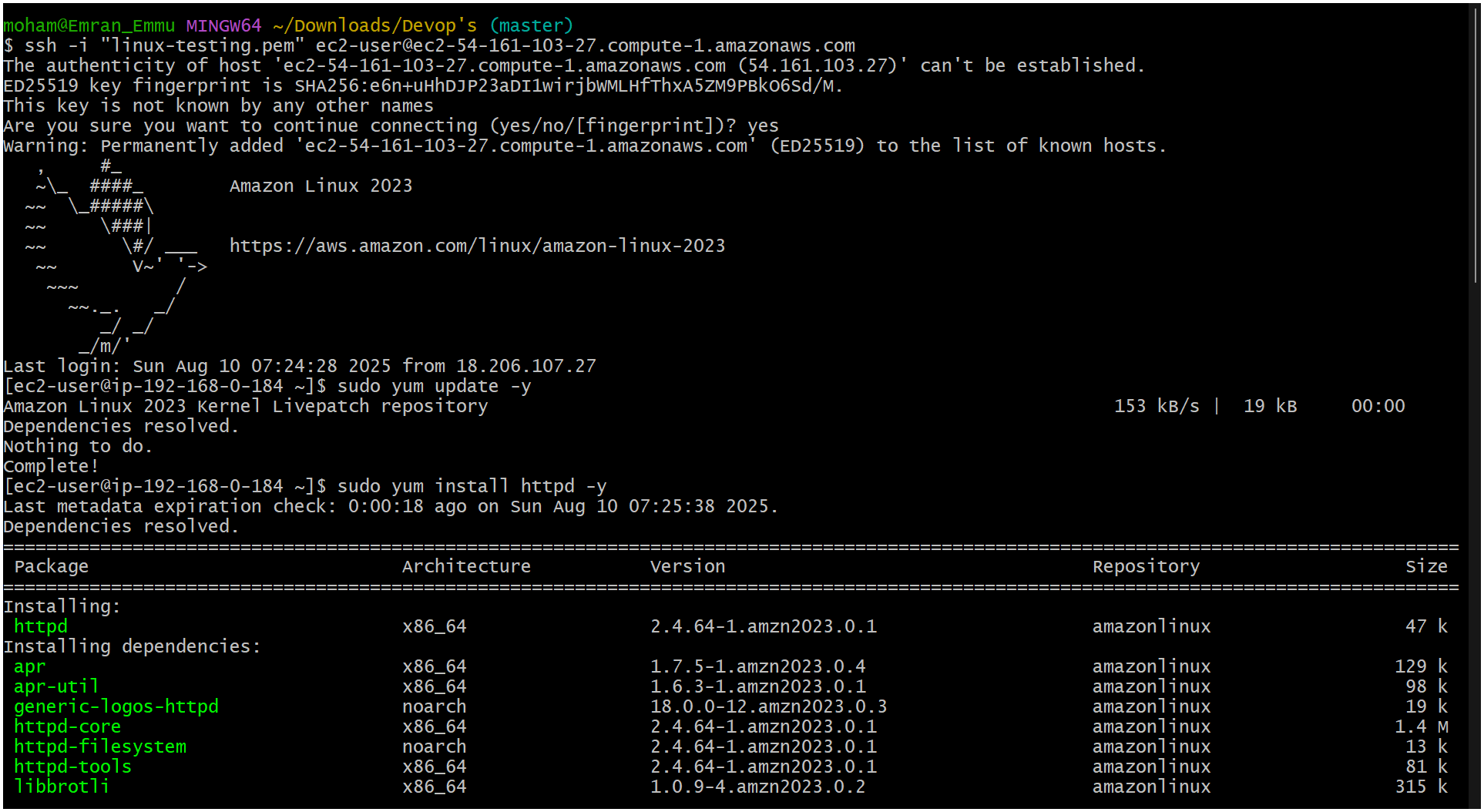
Subnet: Private-Subnet

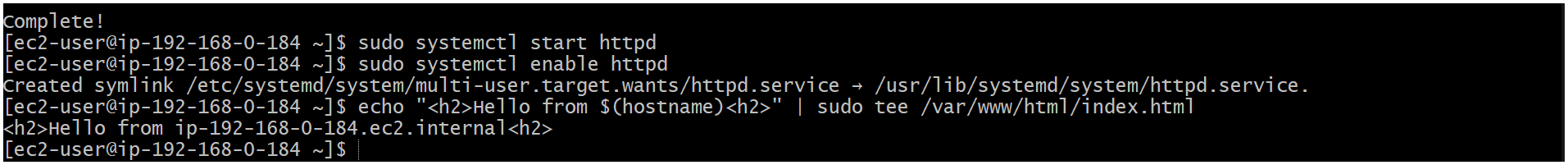
Auto-assign Public IP: Disable

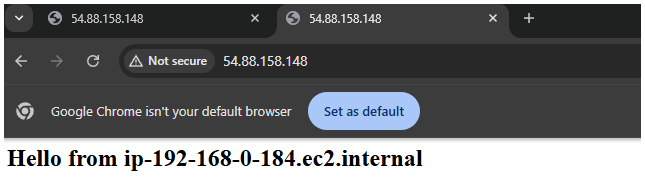
Security group: Select existing security group  
Then lastly launch it  
  


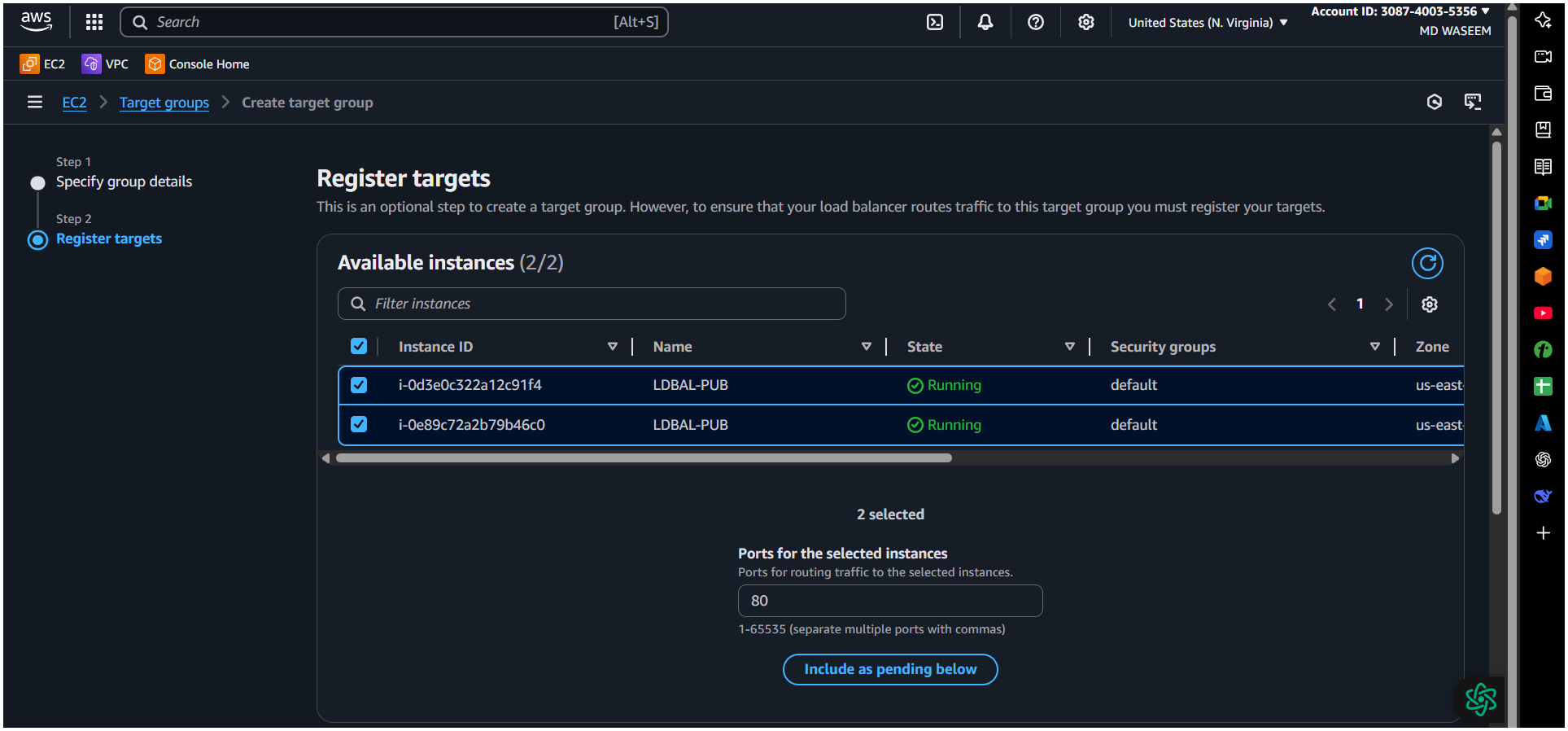
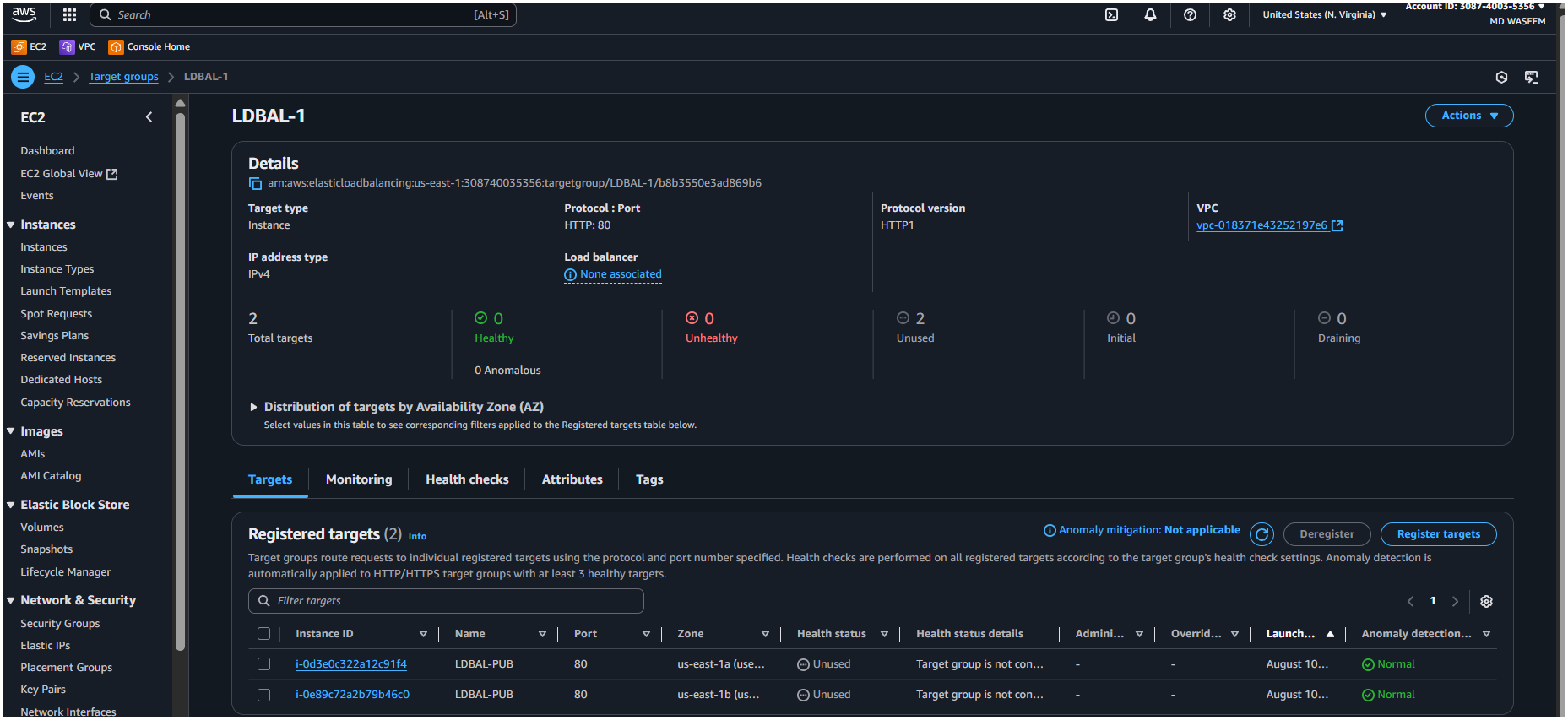
6) Deploy Apache server on both the ec2 instances with sample index.html file.  
  
Step 1: Connect to Each EC2 Instance via SSH  
  
1. Use your key pair to SSH into each instance:  
bash  
ssh -i key.pem ec2-user@<public-ip>  
  
 Step 2: Install Apache Web Server  
Run the following commands on both instances:  
bash  
sudo yum update -y # For Amazon Linux  
sudo yum install httpd -y  
  
 Step 3: Start and Enable Apache  
bash  
sudo systemctl start httpd  
sudo systemctl enable httpd  
  
 Step 4: Add Sample index.html Page  
bash  
echo "<h1>Hello from $(hostname)</h1>" | sudo tee /var/www/html/index.html  
  
 Step 5: Adjust Security Group (for Public Instance)

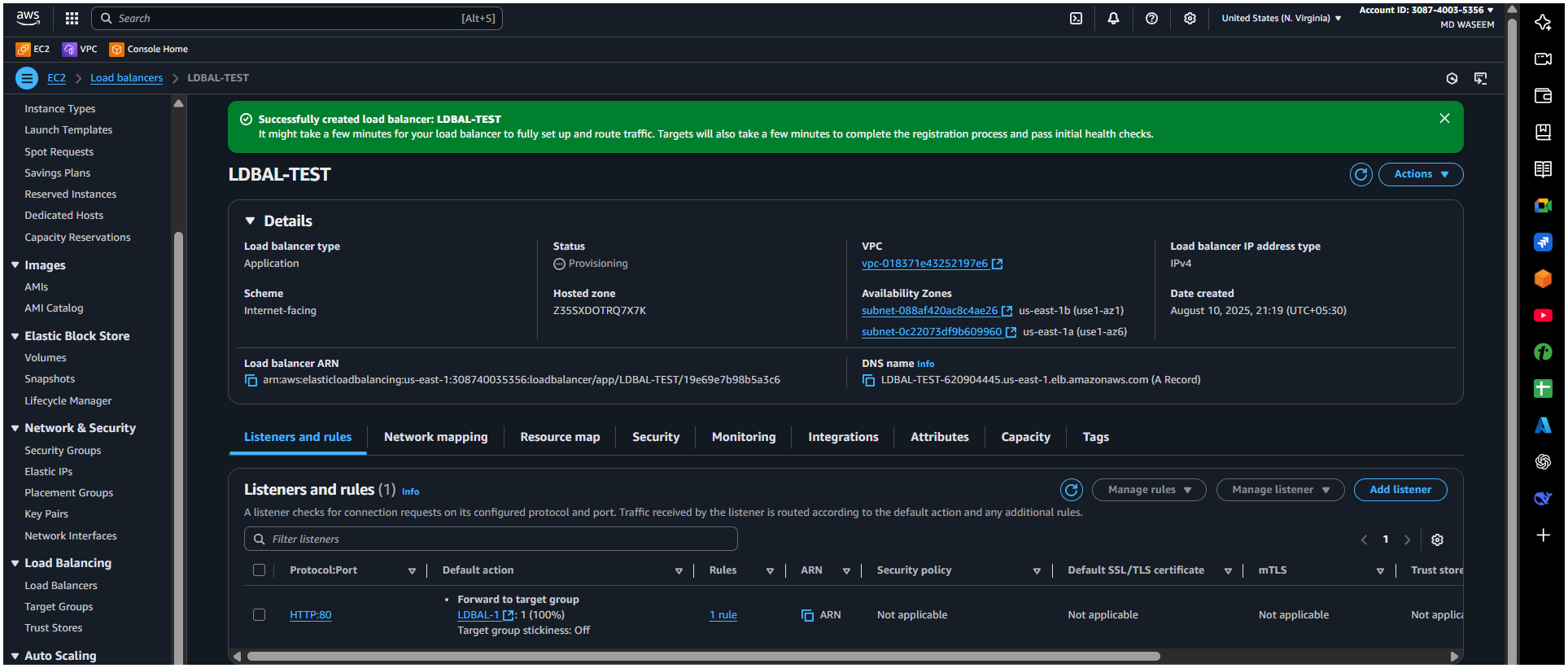
1. Go to EC2 → Security Groups  
2. Edit Inbound Rules of the Public Instance  
 Add rule: HTTP | Port: 80 | Source: 0.0.0.0/0  
Done!  
You can access the public instance in a browser using:http://<public-ip

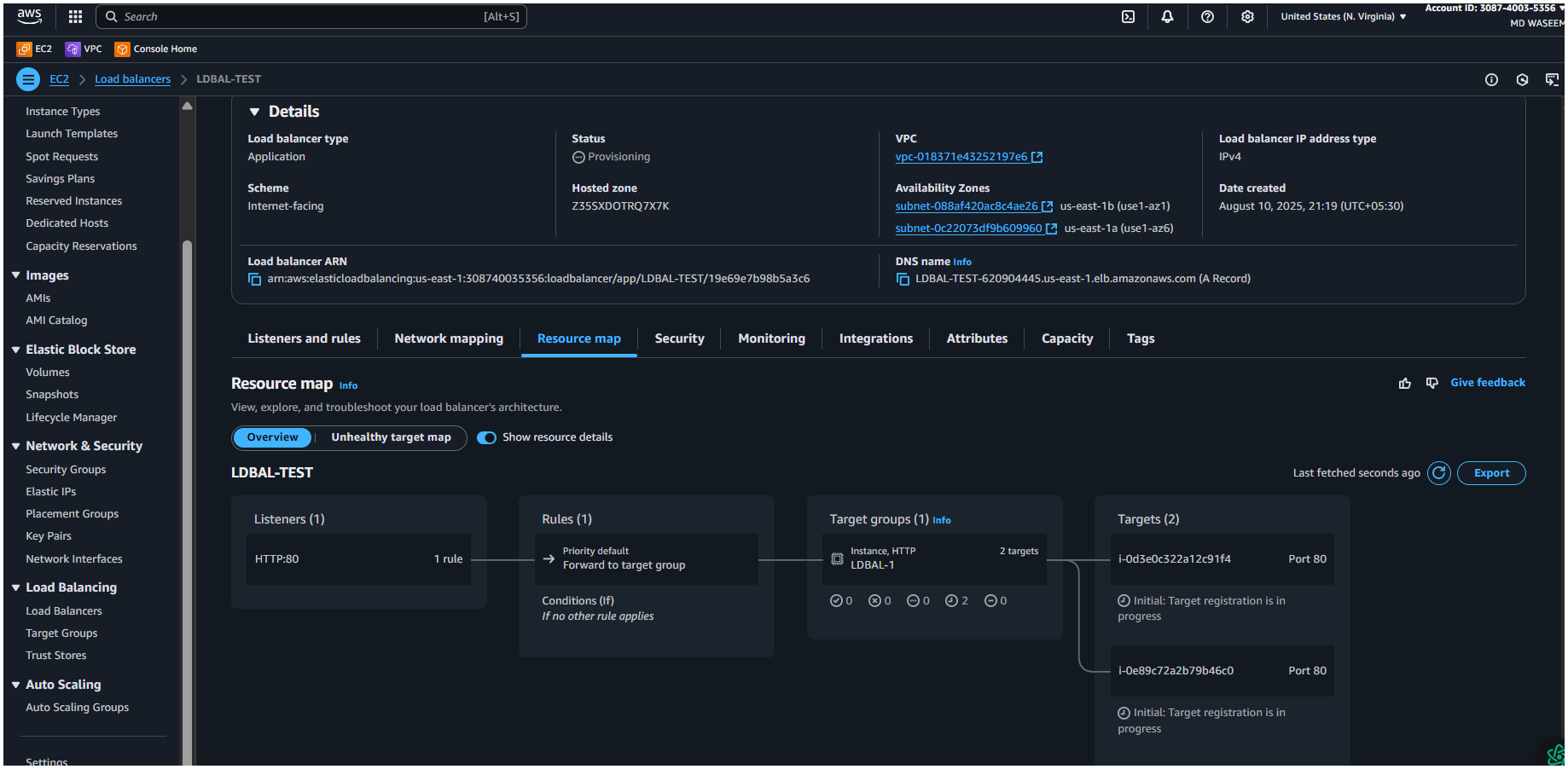
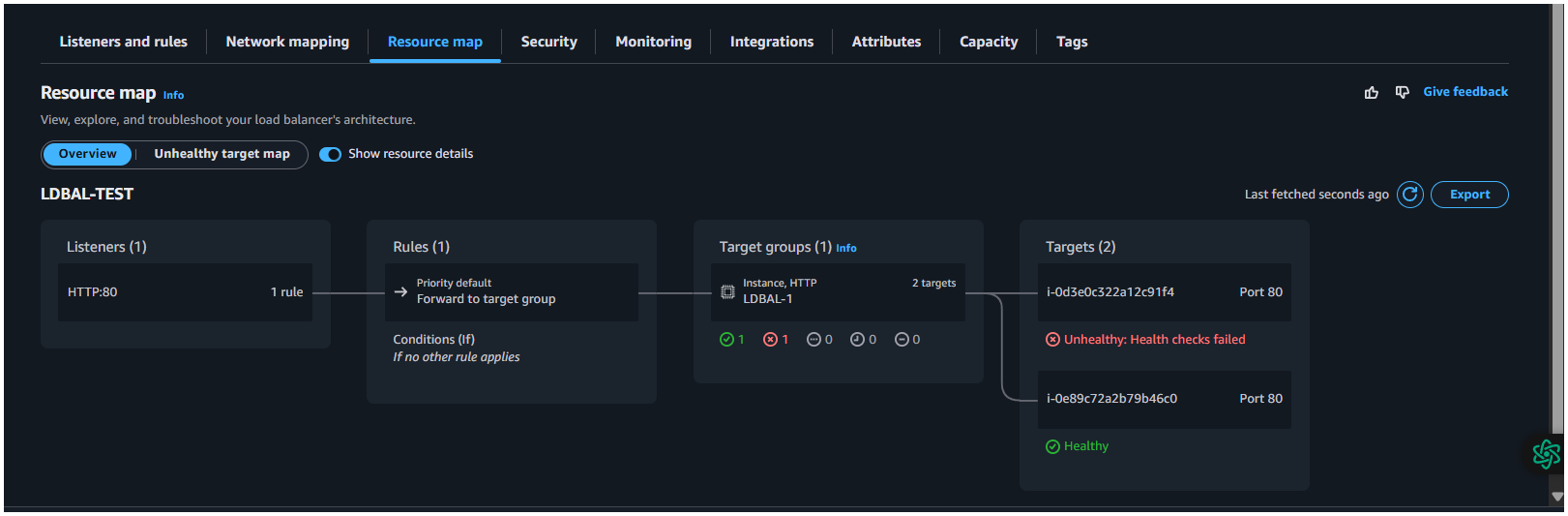
echo "<h1>Hello from $(hostname)</h1>" | sudo tee /var/www/html/index.html  


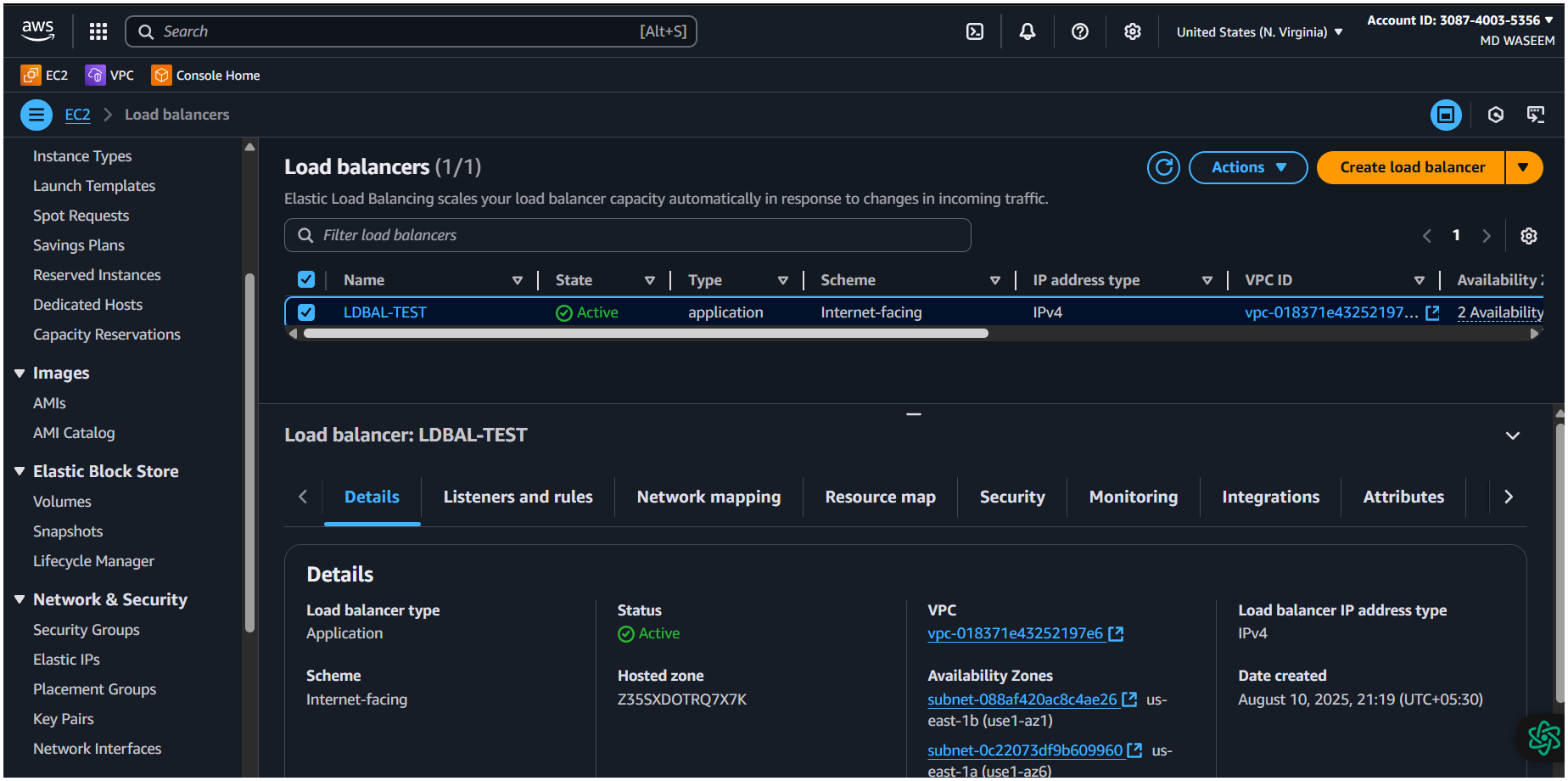
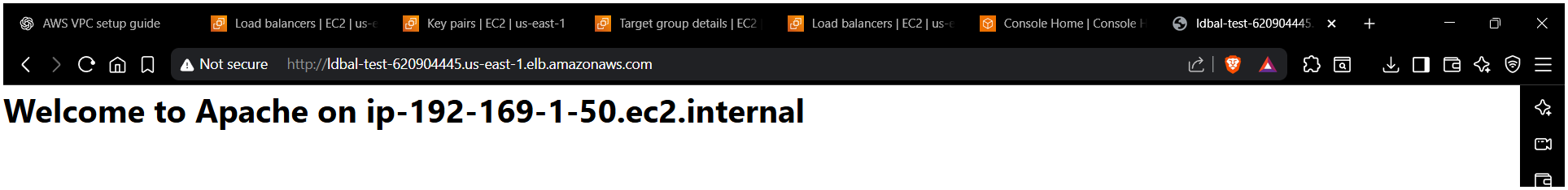


  
  
  
  
  
  
  
  
  
  
  
  
  
  
7) Create one application load balancer and attach the load balancer to both the ec2 instances.  
  
Step 1: Go to EC2 Console → Load Balancers  
 1. Open AWS Console  
 2. Navigate to EC2 → Load Balancers  
 3. Click “Create Load Balancer”  
 4. Choose Application Load Balancer → Click Create  
  
Step 2: Configure Load Balancer  
 1. Name: MyALB  
 2. Scheme: internet-facing  
 3. IP address type: IPv4  
 4. Listeners: HTTP (port 80) is default  
 5. Availability Zones:  
 Select your VPC  
 Check both subnets (public ones) for different AZs

Step 3: Configure Security Group  
 1. Select or create a Security Group that allows inbound HTTP (port 80)  
  
Step 4: Configure Target Group  
 1. Target group name: MyTargetGroup  
 2. Target type: Instance  
 3. Protocol: HTTP | Port: 80  
 4. VPC: Choose your VPC  
 5. Click Next  
  
 Step 5: Register Targets (EC2 Instances)  
 1. Select both EC2 instances (public and private)  
 2. Click Add to registered  
 3. Click Create target group  
  
  
  
Step 6: Review and Create ALB  
 1. Go back to ALB config  
 2. Select the target group you just created  
 3. Review and click Create load balancer  
 Done!





  
  
Your Application Load Balancer now distributes traffic to both EC2 instances. You can access it via the ALB's DNS name shown in the EC2   
→ Load Balancer dashboard.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
08)Store Application load balancer logs to s3.  
 Step 1: Create an S3 Bucket for Logs  
1. Go to S3 → Buckets → Create bucket  
2. Name: e.g., alb-logs-bucket-yourname  
3. Choose Region (same as ALB)  
4. Keep other settings default (or configure as needed)  
5. Click Create bucket  
  
Add this policy in your bucket the highlighted in Green is your id of Aws  
  
{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "logdelivery.elasticloadbalancing.amazonaws.com"

},

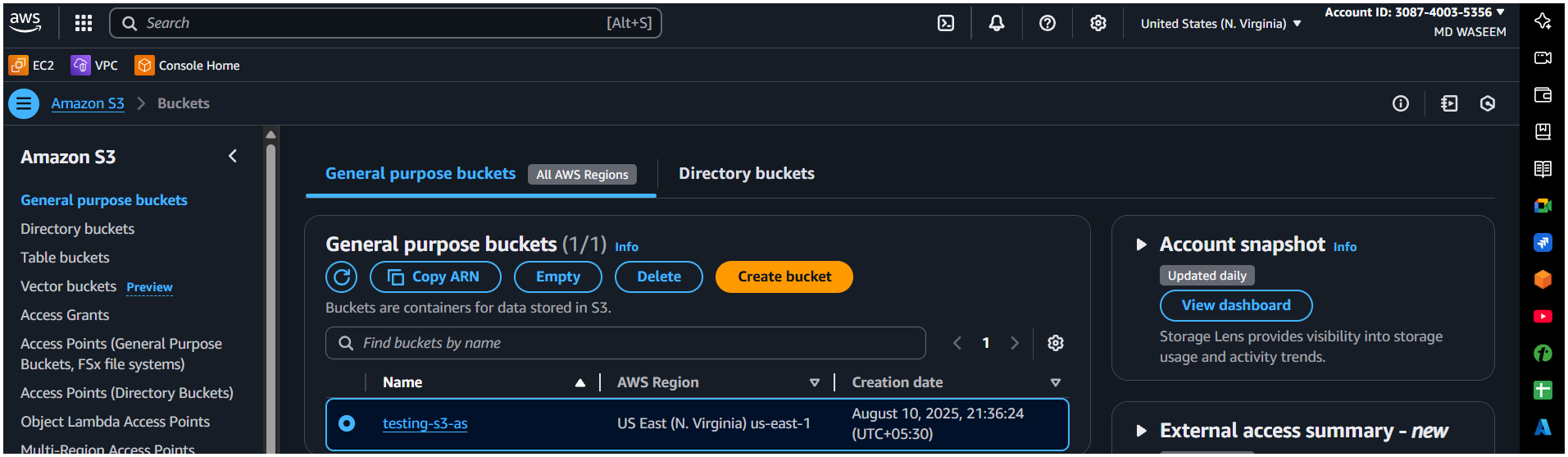
"Action": "s3:PutObject",

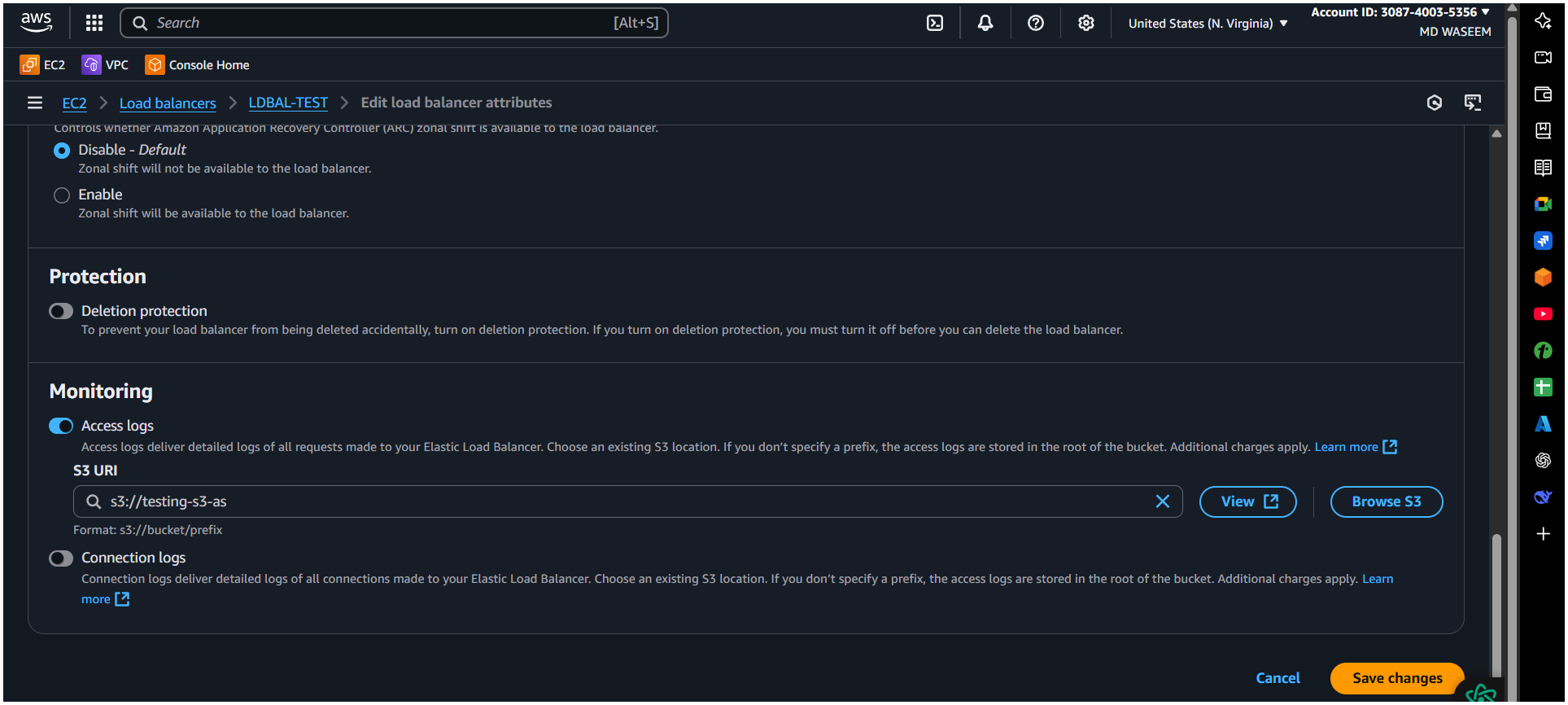
"Resource": "arn:aws:s3:::testing-s3-as/AWSLogs/308740035356/\*"

}

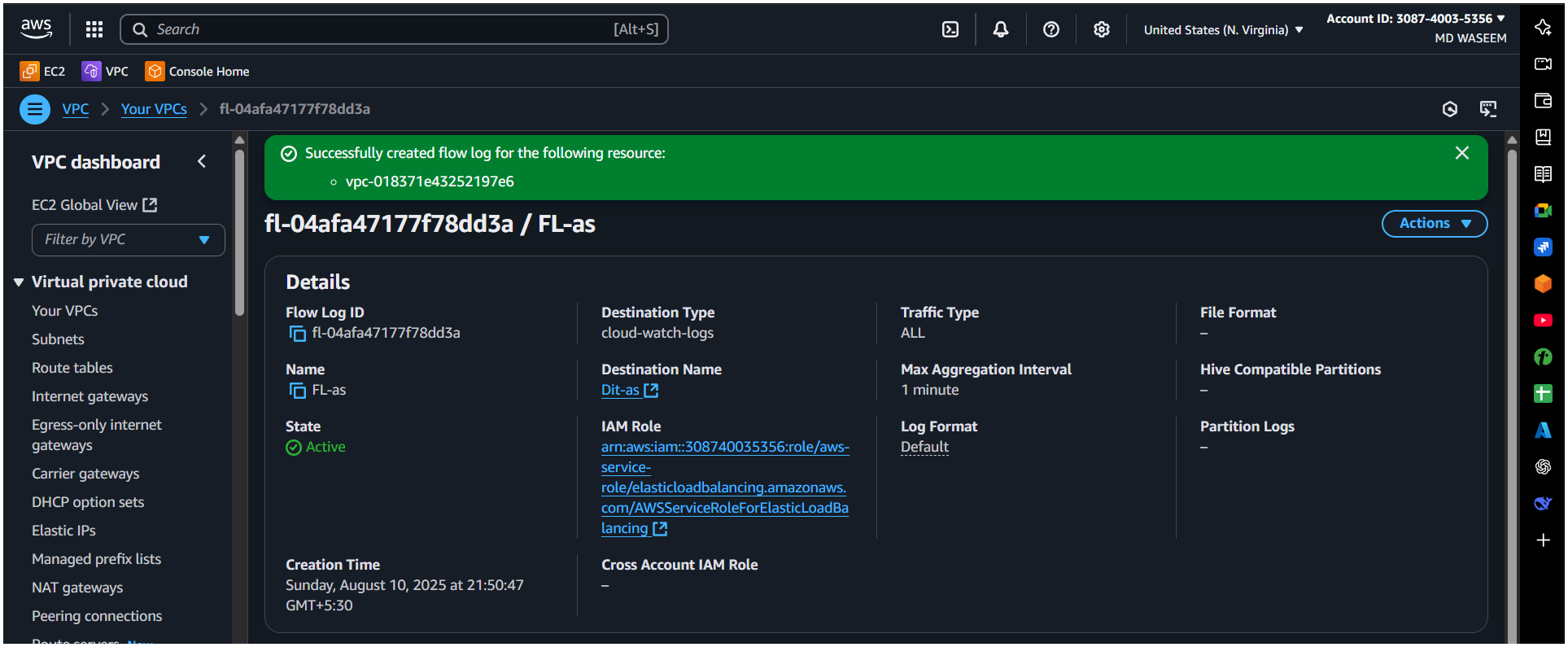
]

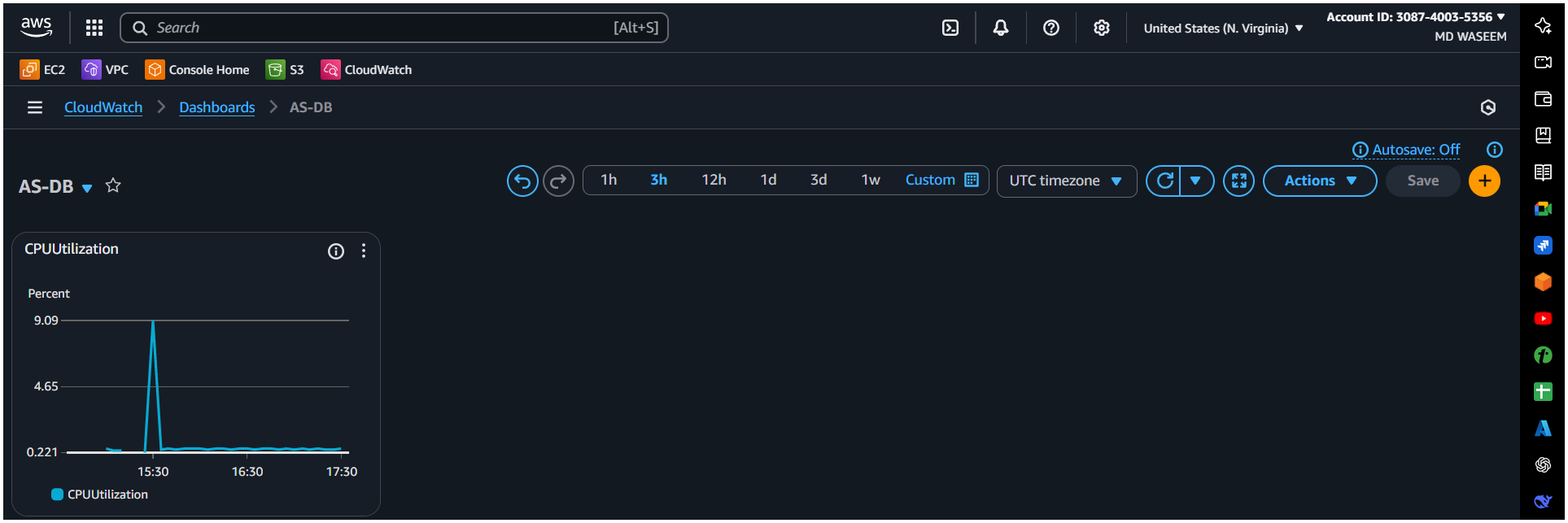
}

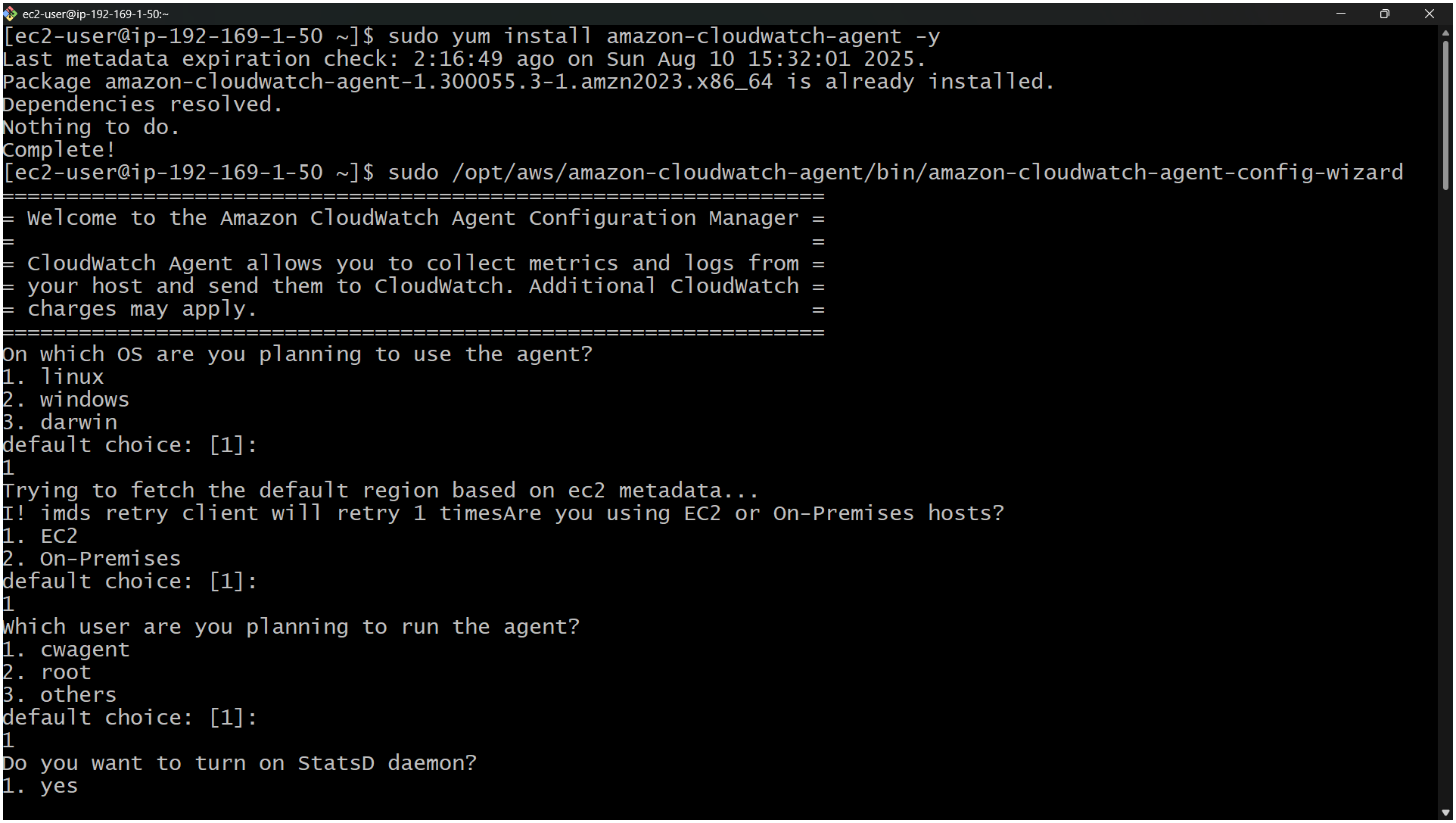
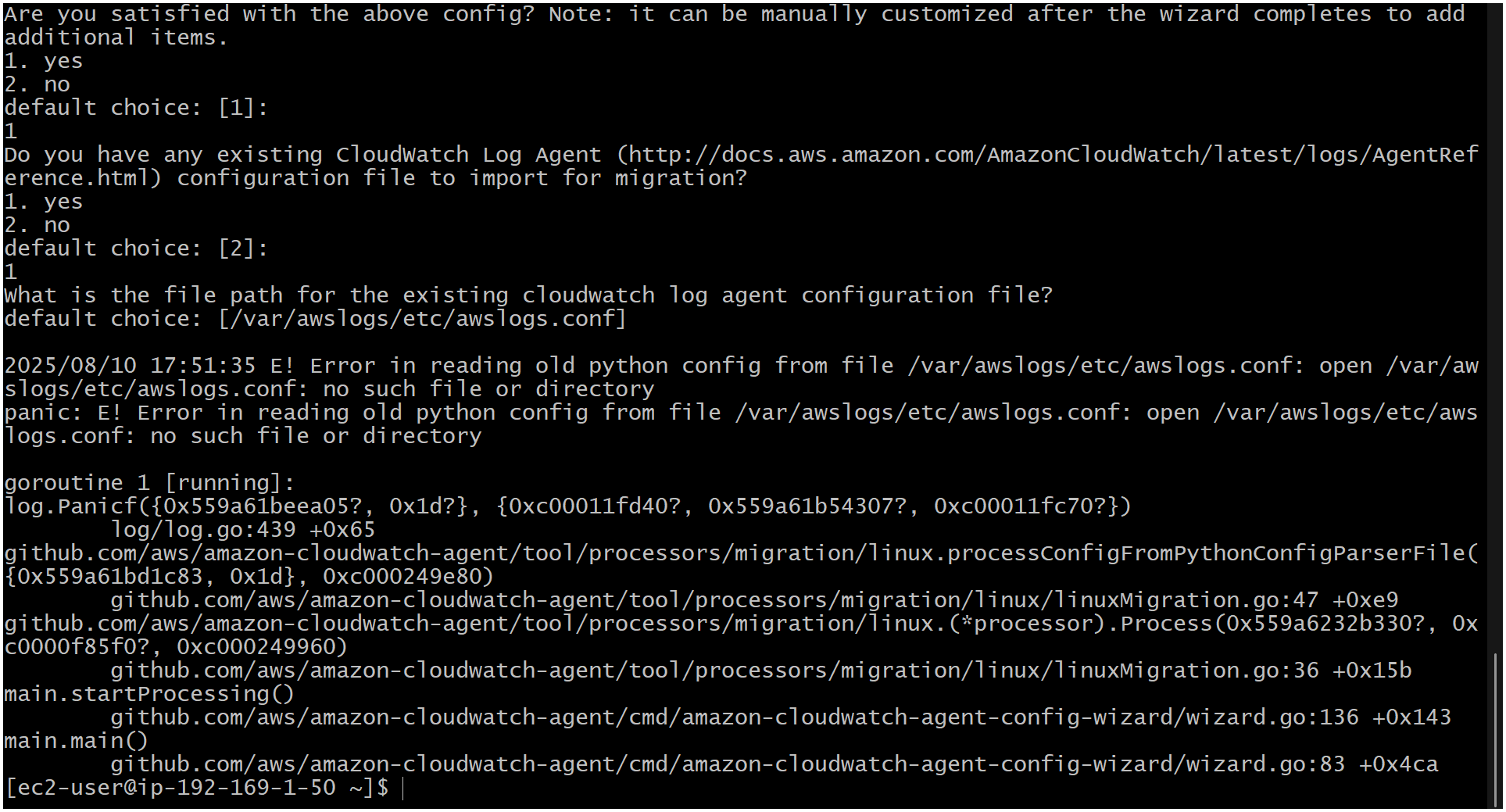


09)Store Application load balancer logs to s3.  


10) Store the VPC flow logs to CloudWatch group.

Step 1: Go to VPC Dashboard  
1. Open AWS Console  
2. Navigate to VPC → Your VPCs  
3. Select your VPC  
  
 Step 2: Create Flow Log  
1. Click on “Actions → Create flow log”  
2. Filter: Select All, Accept, or Reject   
3. Destination: Select Send to CloudWatch Logs  
  
 Step 3: Set Log Group and IAM Role  
1. Log group:  
 Select existing or create new   
2. IAM Role:  
 Choose existing role or click Set up permissions  
 AWS will auto-generate a role like FlowLogsRole  
  
 Step 4: Finalize and Create  
1. Click Create flow log  
2. It starts logging VPC traffic to the selected CloudWatch log group  
 Done!  
  
Your VPC Flow Logs are now stored in CloudWatch Logs, where you can monitor and analyse them.  
  


11) Create Monitoring Dashboards to monitor CPU utilization and to monitor Apache service.  
  
Step 1: Open CloudWatch Console  
1. Open AWS Console  
2. Navigate to CloudWatch → Dashboards  
3. Click Create dashboard  
4. Enter a Dashboard name   
5. Click Create dashboard  
  
 Step 2: Add CPU Utilization Widget  
1. Choose “Line” widget  
2. Click Configure  
3. Under Metrics, select:  
 EC2 → Per-Instance Metrics → CPUUtilization  
 Select your EC2 instance(s)  
4. Click Create widget  
  
  
 Step 3: Create Apache Service Monitoring Metric (Custom Metric)  
1. On each EC2 instance, run a script or CloudWatch Agent to push Apache status (e.g., up/down) as a custom metric to CloudWatch   
(see note below).  
2. Alternatively, create a CloudWatch Logs Metric Filter if Apache logs are pushed to CloudWatch Logs.

  
  
  
  
 Step 4: Add Apache Service Widget  
1. Back on the dashboard, click Add widget  
2. Choose “Number” or “Line” widget  
3. Select your custom Apache metric namespace  
4. Select metric representing Apache service status  
5. Click Create widget  
  
 Step 5: Save and Review Dashboard  
•The dashboard now shows CPU usage and Apache service status.  
Note: Apache monitoring script example  
•Run on EC2 instance and publish custom metric to CloudWatch (e.g., status 1 if running, 0 if stopped).  
 Done!  
You now have a CloudWatch dashboard monitoring CPU utilization and Apache service health.

12) CPU utilizations more than 70% then it should triggered Autoscaling and launch new instance.