ASG Tasks

1.Create one VPC in N.Virginia region.

TITLE:

VPC in N.Virginia

OBJECTIVE:

Cretae a VPC in N.Virrginia region for custom network setup.

PREREQUISITES:

AWS account

Region has to be in N,Virginia

CIDR range

STEP-BY-STEP-IMPLIMENTATION:

Go to console and search for the VPC and click on create VPC and it will ask for certain CIDR range

A screenshot of a computer

AI-generated content may be incorrect.

Created VPC with name N.Virginia in N.Virginia region

A screenshot of a computer

AI-generated content may be incorrect.

CONLUSION:

Successfully created a VPC in N.Virginia for building network infrastructure.

2.Create two subnets one public subnet and one private subnet.

TITLE:

Creating Subnets

OBJECTIVE:

Create one public subnet and one private subnet inside the VPC.

PREREQUISITES:

VPC must be created

CIDR range

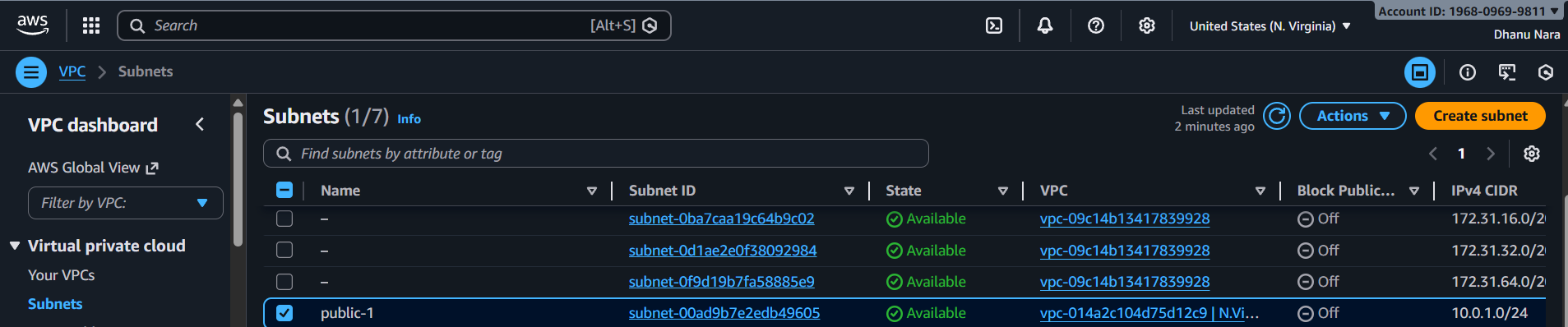
Subnets CIDR has to be under VPC range

Region needs to be selected

Internet gateway

STEP-BY-STEP-IMPLIMENTATION:

First go to VPC console and go to subnets and click on create subnet

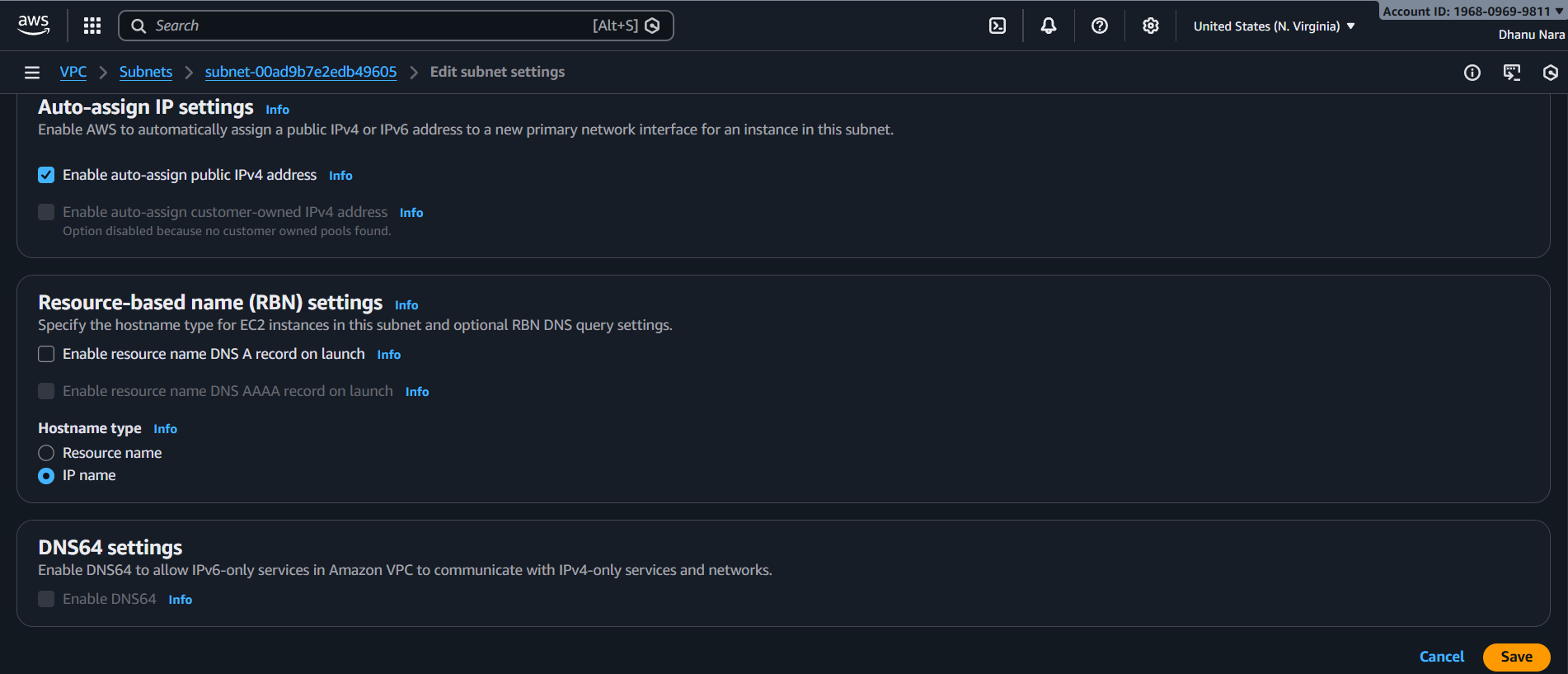


A screenshot of a computer

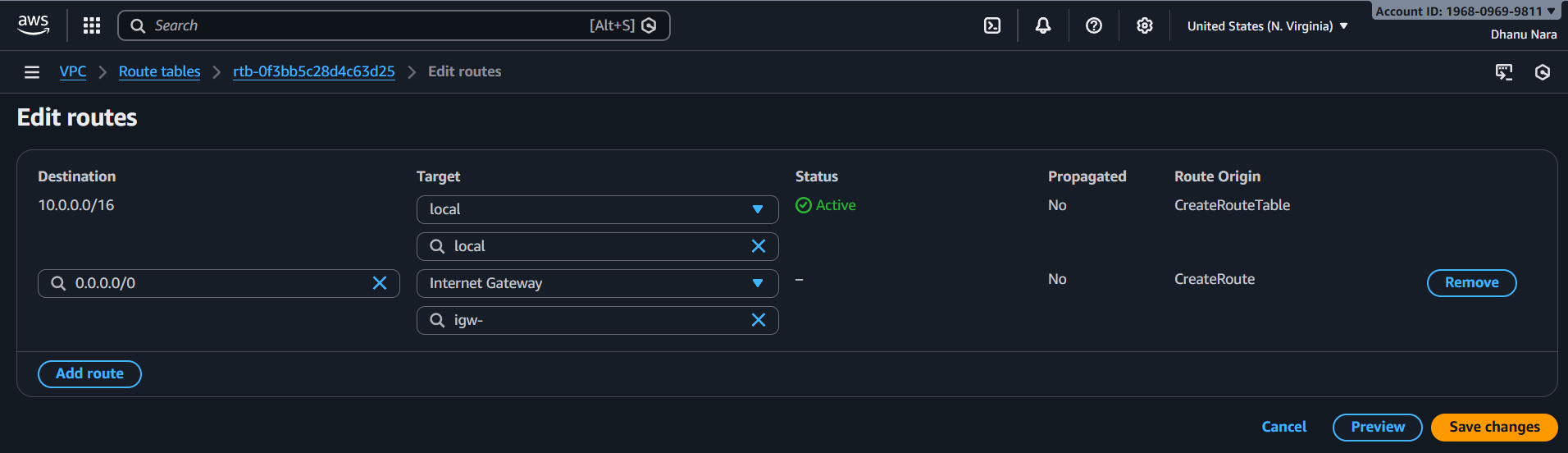
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



Create internet gateway and attach that to public subnet and auto assign public should be turned on.



A screenshot of a computer

AI-generated content may be incorrect.

Created a subnet and not attached Internet Gateway because private subnet does not have internet access

A screenshot of a computer

AI-generated content may be incorrect.

CONCLUSION:

Successfully created one public subnet and private subnet inside VPC.

3.Attach an IGW to the VPC.

TITLE:

IGW to VPC

OBJECTIVE:

Attach an internet Gateway to the VPC to for internet access.

PREREQUISITES:

Create VPC

Internet Gateway

Region set to N.Virginia

STEP-BY-STEP-IMPLIMENTATION:

Create a VPC

A screenshot of a computer

AI-generated content may be incorrect.

Attach Internet Gateway to VPC

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

CONCLUSION:

Internet Gateway successfully attached to VPC.

4.Create one public route table and one private route table.

TITLE:

Creating Public and Private route tables

OBJECTIVE:

Create one public route table and one private route table for subnet traffic routing

PREREQUISITES:

VPC has to be created

Public and privates subnets

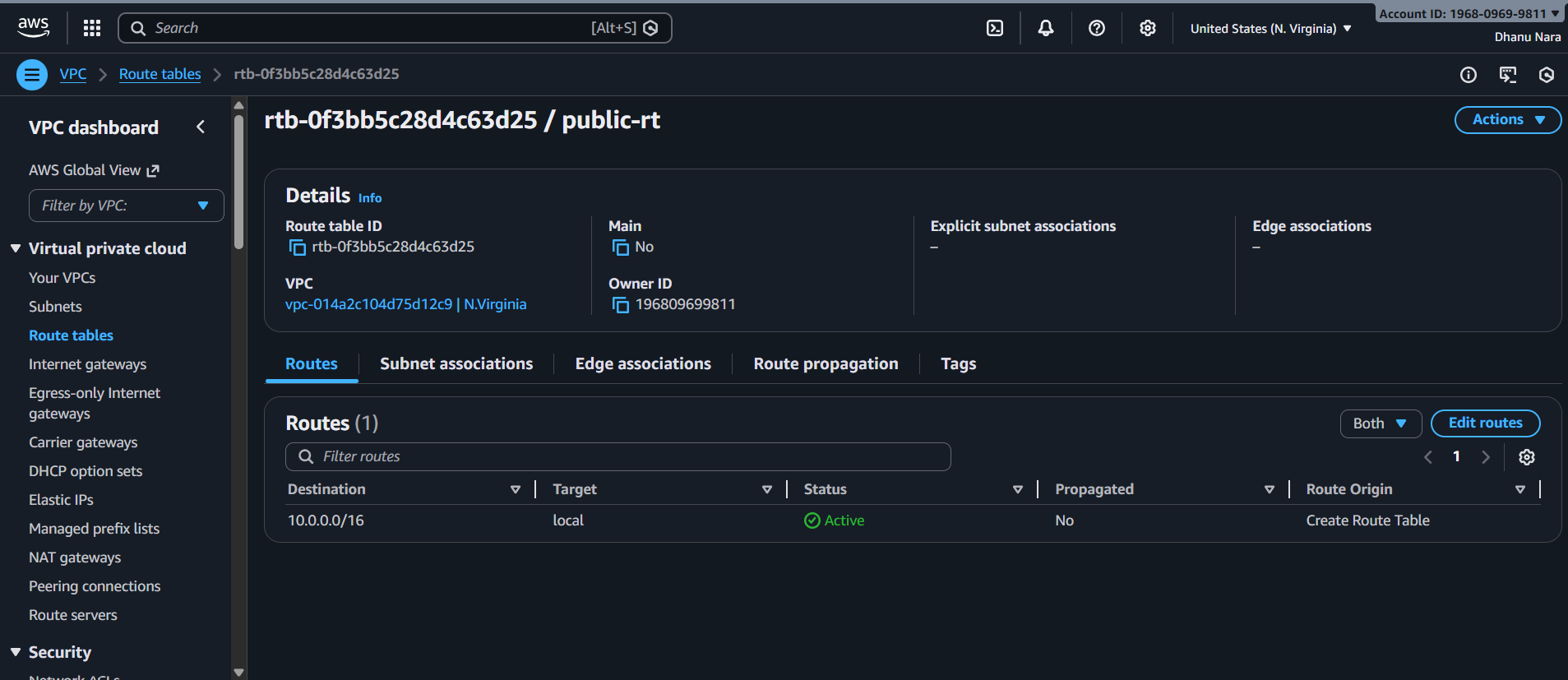
Internet Gateway attached to VPC for public route

Region set to N.Virginia

STEP-BY-STEP-IMPLIMENTATION:

Go to VPC and select route table and click on create route table and create two route tables

Public route should have IGW attachment



A screenshot of a computer program

AI-generated content may be incorrect.

CONCLUSION:

Created separate public and private route tables for routing management

5.Deploy NAT Gateway in the public subnet and attach the NAT gateway to the private subnet.

TITLE:

NAT Gateway Deployment

OBJECTIVE:

Deploy a NAT Gateway in the public subnet and route private subnet traffic through it

PREREQUISITES:

VPC has to be created

Public and private subnets

Public and private route tables

Internet Gateway attached to VPC

Elastic ip available for NAT Gateway

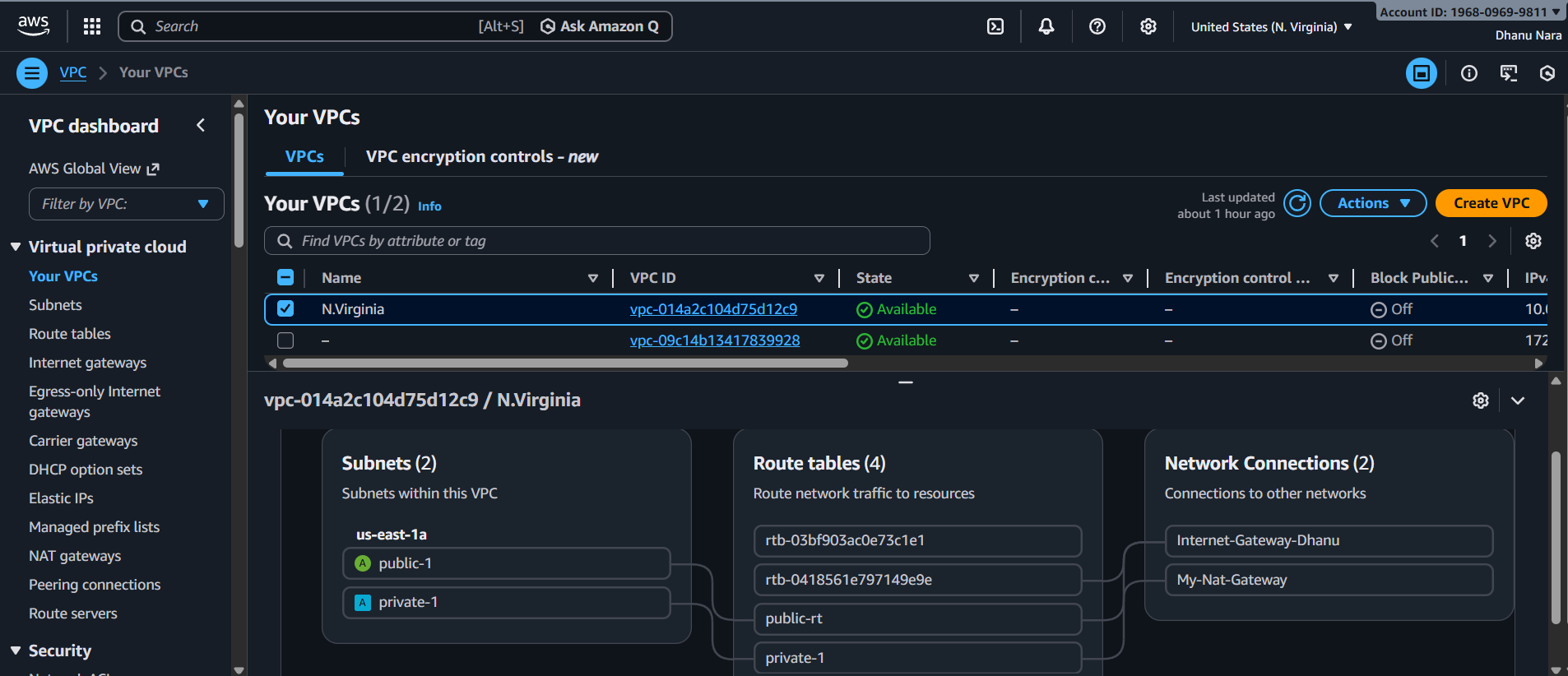
STEP-BY-STEP-IMPLIMENTATION:

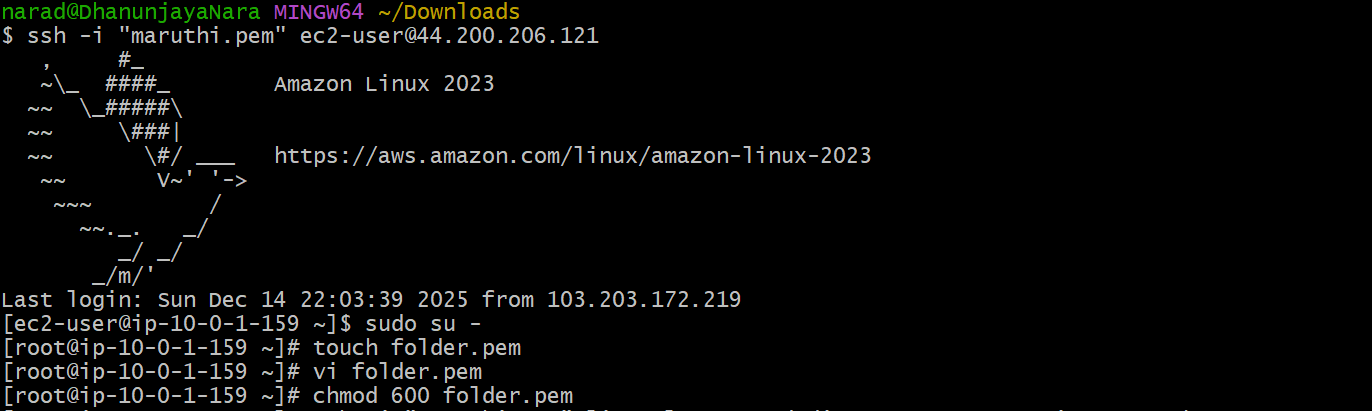
Created a vpc

A screenshot of a computer

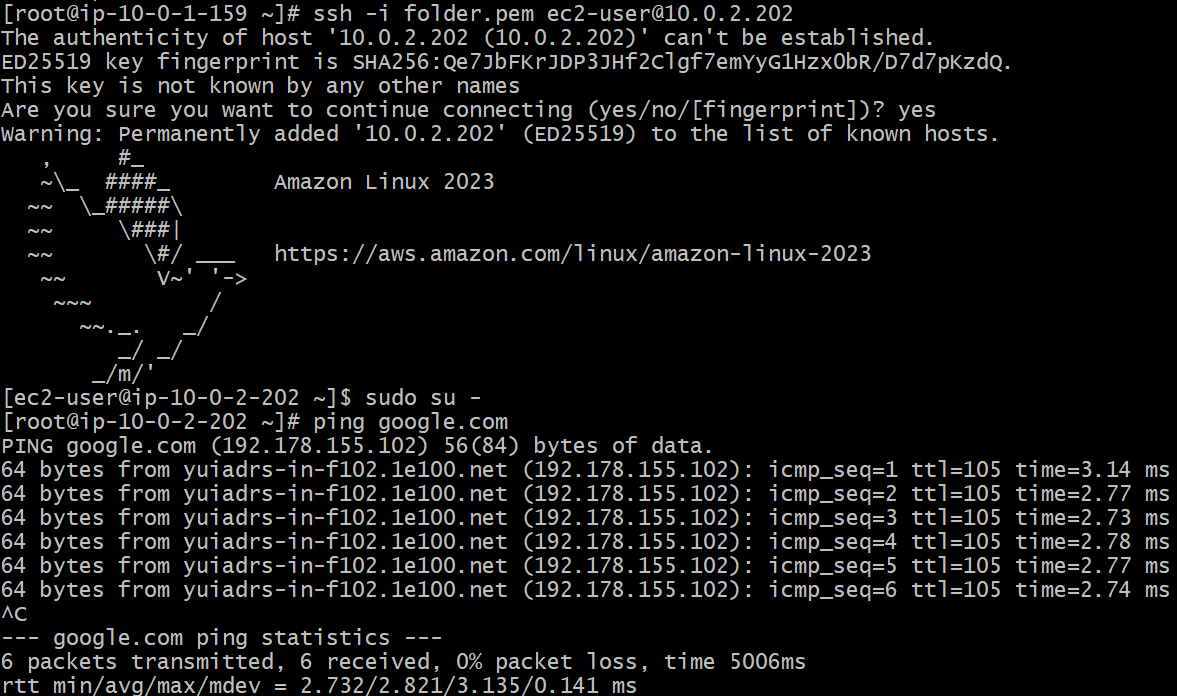
AI-generated content may be incorrect.

Public subnet is attached to IGW





With the help of jump server internet access will be provided for private subnet



CONCLUSION:

NAT Gateway deployed in public subnet and private subnet routed through NAT for outbound internet access.

6.Create two instances one in the public subnet and one in the private subnet.

TITLE:

Public and private instances

OBJECTIVE:

Launch two EC2 instances -one in the public subnet and another in the private subnet

PREREQUISITES:

Two ec2 instances

Public and private subnets

Internet gateway

NAT Gateway

STEP-BY-STEP-IMPLIMENTATION:

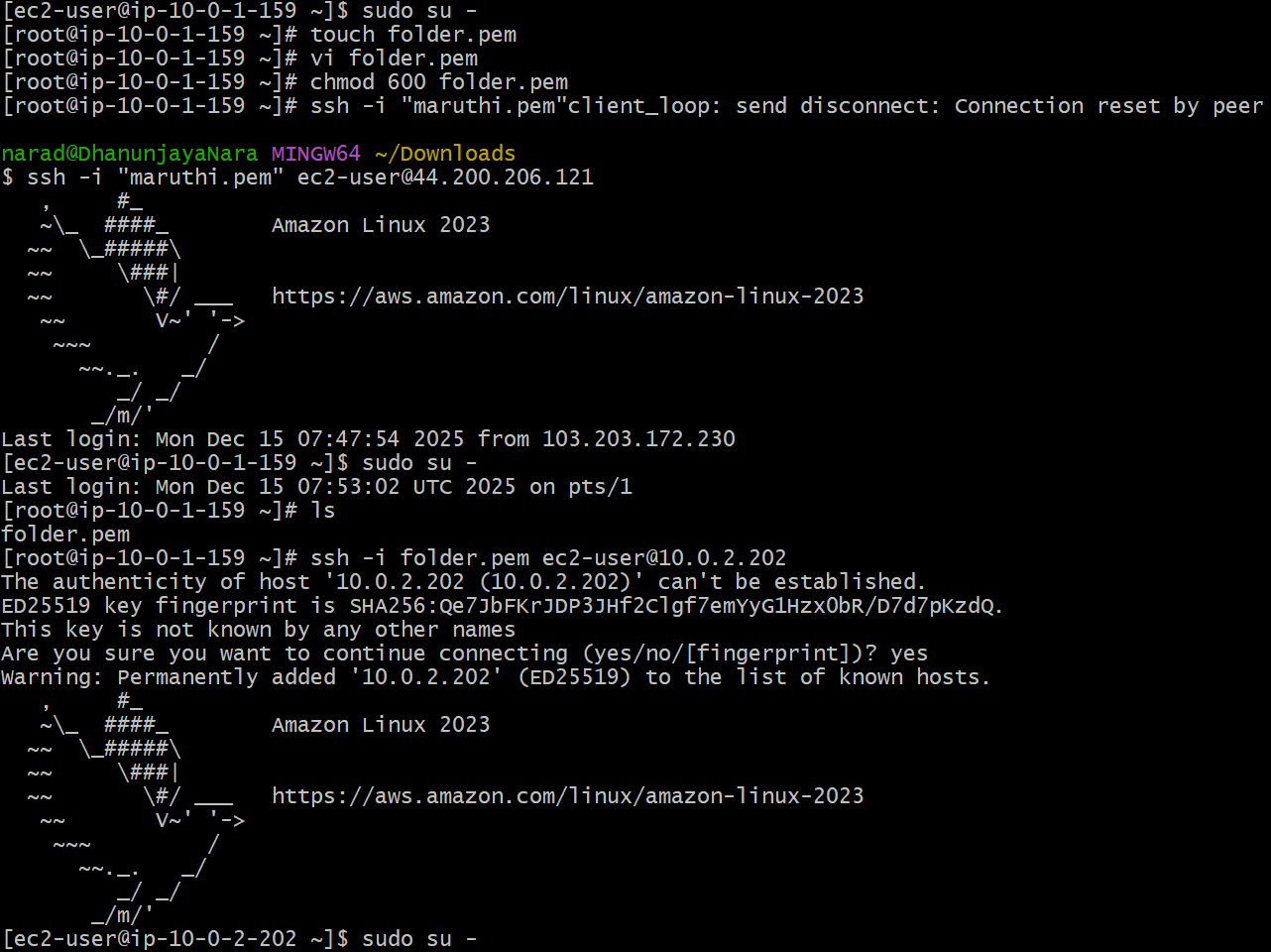
Launch two instances and login to the public subnet instance using ssh keypair and public IP.

A screenshot of a computer

AI-generated content may be incorrect.



For private instance we need to login through jump server to give internet access.



CONCLUSION:

Launched instances in both public and private and provide internet access to private instance through NAT Gateway.

7.Deploy apache server on both EC2 instances with a sample index.html file.

TITLE:

Index.html files of apache in both the servers

OBJECTIVE:

Deploy index.html on the public instance which has internet access as well as private instance which do not have internet access

PREREQUISITES:

Two ec2 instances

Public and private subnets

Internet gateway

NAT Gateway

Apache web server

STEP-BY-STEP-IMPLIMENTATION:

First installed apache

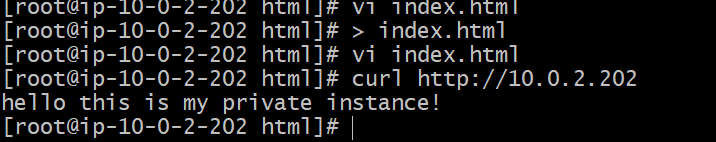


Deployed index.html file in /var/www/html location

A close-up of a computer screen

AI-generated content may be incorrect.

This is index.html for private instance and can be checked with curl command



CONCLUSION:

Apache installed on both the instances with sample index.html pages.

8.Create one application load balancer and attach it to both EC2 instances.

TITLE:Application Load Balancer

OBJECTIVE:

To distribute incoming HTTP/HTTPS traffic across multiple EC2 instances using an Application Load Balancer, ensuring high availability, scalability, and fault tolerance of the application.

PREREQUISITES:

EC2 instances with service running in it

Target group

Application Load Balancer

STEP-BY-STEP-IMPLIMENTATION:

EC2 instances with httpd running in it

A screenshot of a computer

AI-generated content may be incorrect.

Created target groups

A screenshot of a computer

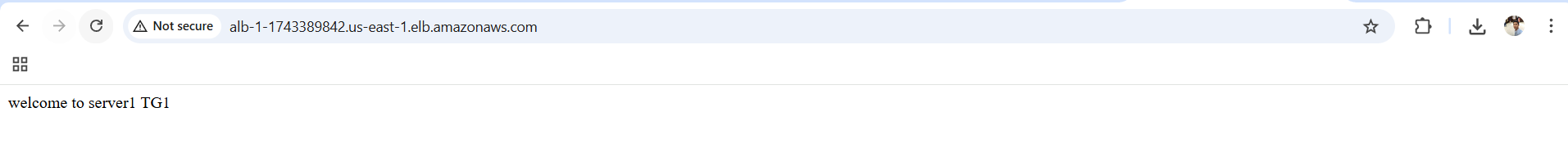
AI-generated content may be incorrect.

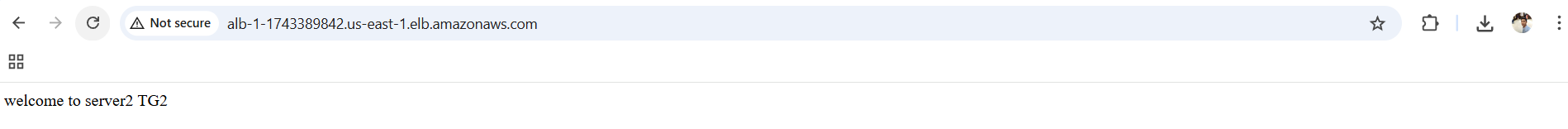
Created Application Load Balancer

A screenshot of a computer

AI-generated content may be incorrect.

Now browse using the DNS name of Application Load Balancer





CONCLUSION: Attaching an Application Load Balancer (ALB) to two EC2 instances ensures high availability and reliability by distributing incoming traffic evenly across both instances, improving performance and preventing downtime if one instance fails.

9. Push the Application Load Balancer logs S3.

TITLE:

Application Load Blancer logs to S3

OBJECTIVE:

To store Application Load Balancer access logs in Amazon S3 for centralized request analysis, security auditing, troubleshooting, and long-term retention.

PREREQUISITES:

Application Load Balancer

S3 Bucket

STEP-BY-STEP-IMPLIMENTATION:

Create bucket

A screenshot of a computer

AI-generated content may be incorrect.

Add bucket policy

A screenshot of a computer

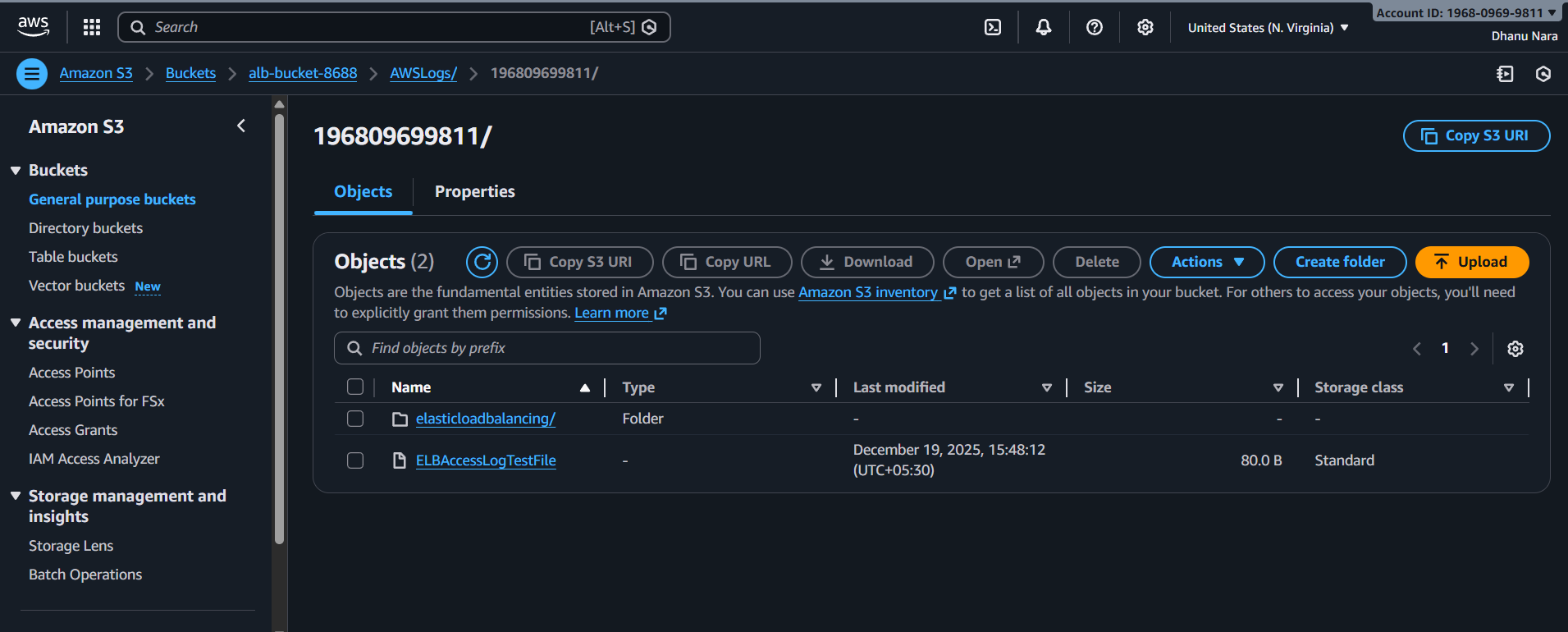
AI-generated content may be incorrect.

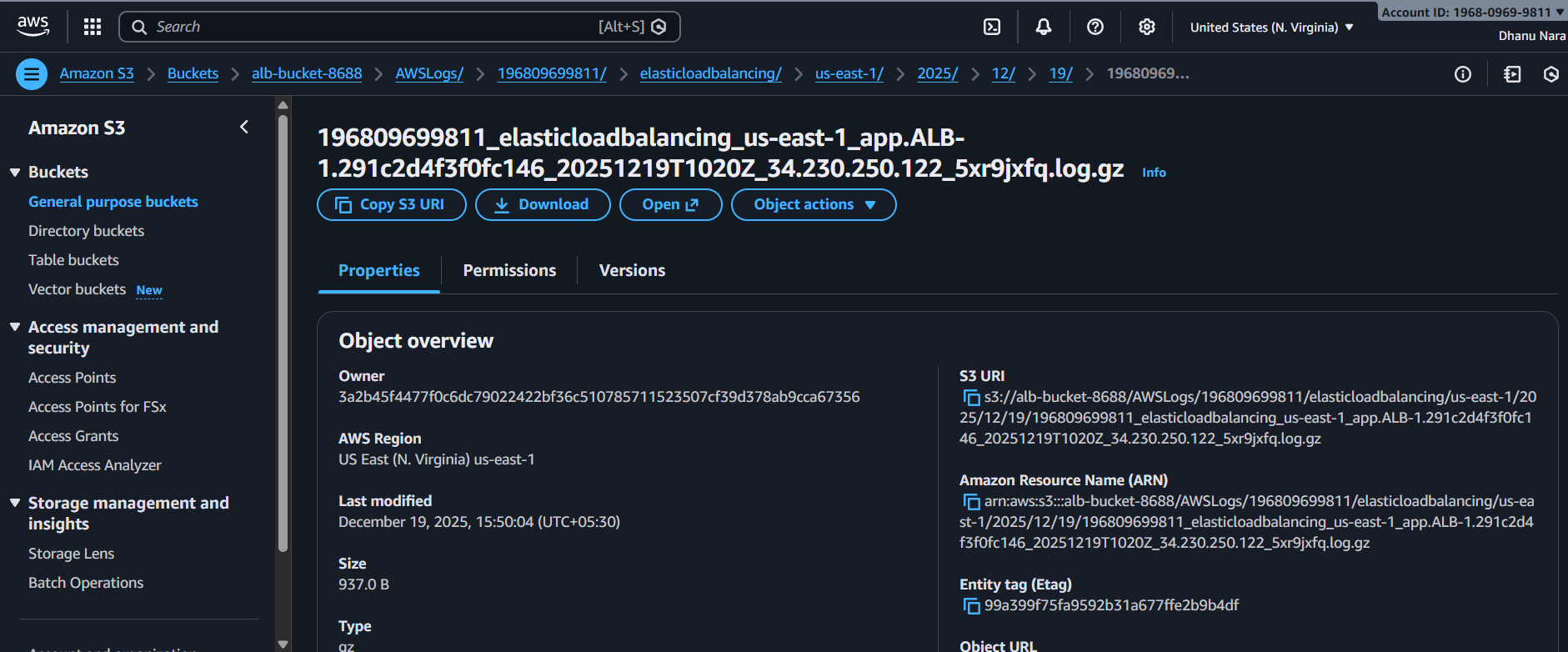
Create Application Load Balancer attribute

A screenshot of a computer

AI-generated content may be incorrect.

Access logs of application load balancer





CONCLUSION: Pushing Application Load Balancer logs to Amazon S3 enables centralized storage for monitoring traffic patterns, troubleshooting issues, and meeting security and compliance requirements.

**10.store the vpc flowlogs in cloudwatch log group.**

**TITLE:**

**VPC flowlogs in CloudWatch**

**OBJECTIVE:**

**To capture and store VPC network traffic details in a CloudWatch Log Group for monitoring,troubleshooting,and security analysis of inbound and outbound traffic within the VPC.**

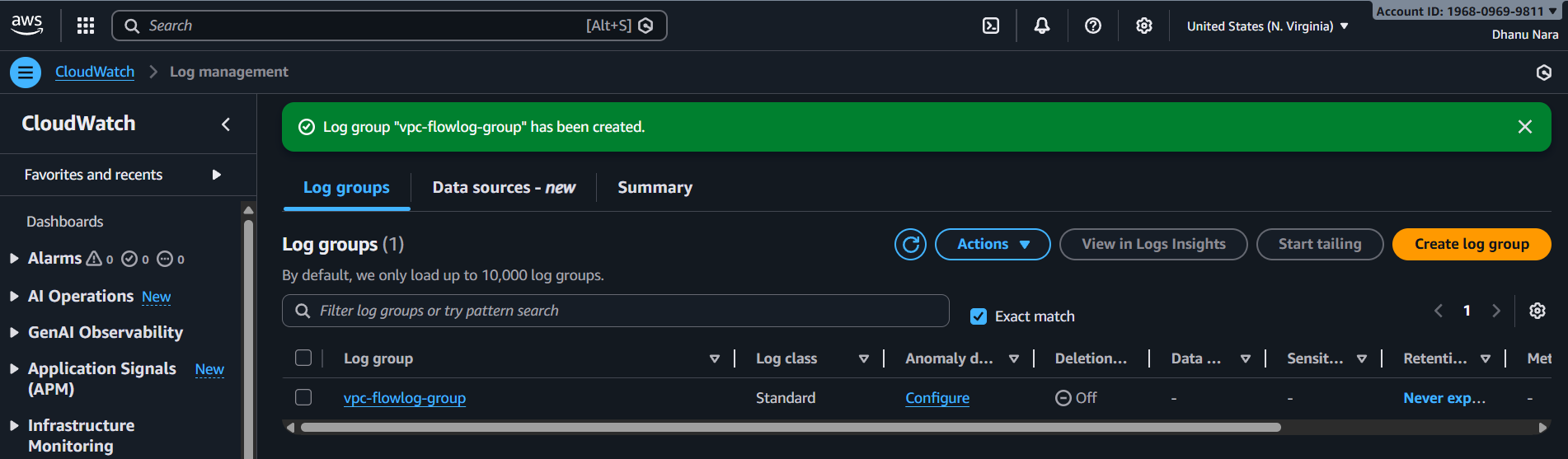
PREREQUISITES:

Log group in cloudwatch

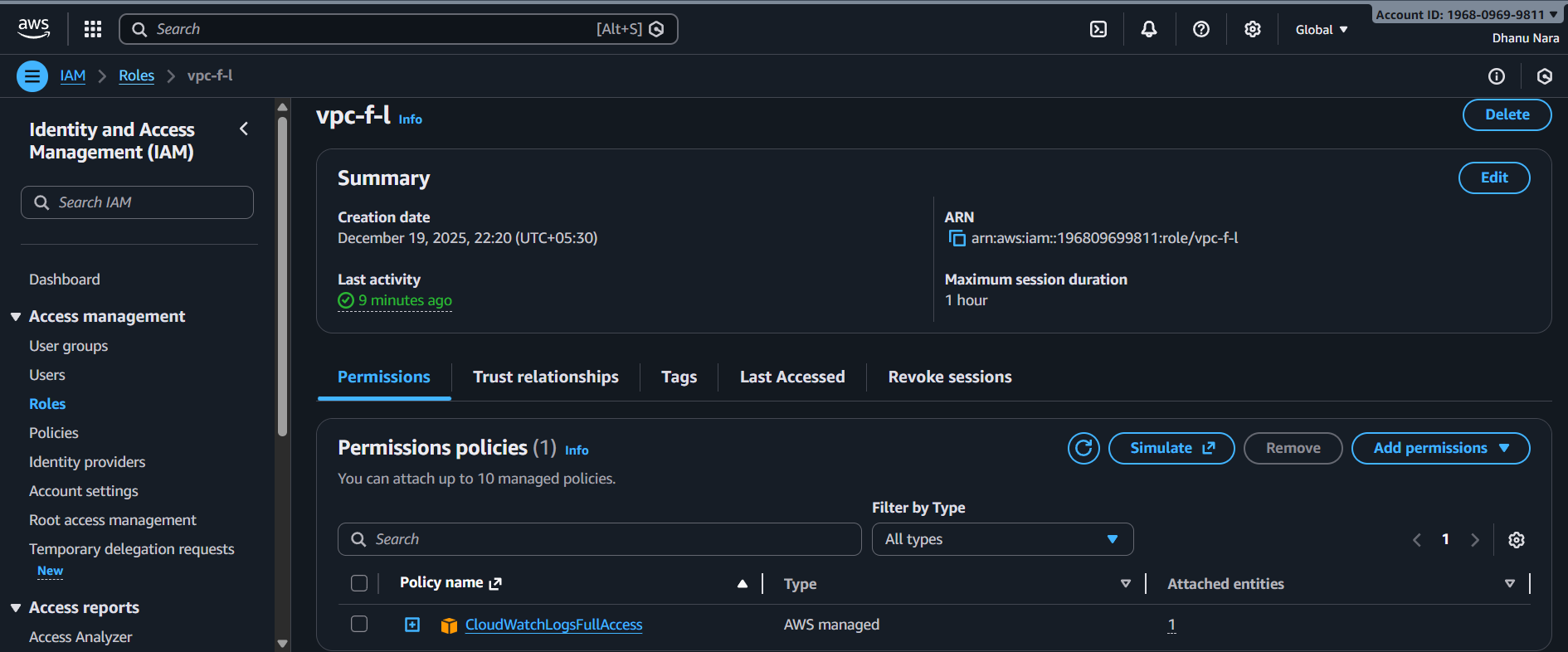
Custom role in IAM

Flow log in VPC

**Log goup created in clodwatch**

****

IAM custom trusted role created with policy and given cloudwatchlogfullaccess



Create flow log in vpc

A screenshot of a computer

AI-generated content may be incorrect.

Now logs can be monitore in cloud watch

**A screenshot of a computer

AI-generated content may be incorrect.**

CONCLUSION:

Configuring VPC Flow Logs to CloudWatch Log Groups provides centralized visibility into VPC network traffic, enabling effective monitoring, troubleshooting, and security analysis by tracking accepted and rejected connections in real time.

11.Create monitoring dashboards to monitor CPU utilization and to monitor Apache service.

TITLE:

Dashboards for CPU utilization and Apache

OBJECTIVE:

Create CloudWatch dashboards to monitor EC2 CPU utilization and Apache service status.

PREREQUISITES:

AWS Account

One EC2 instance which is running

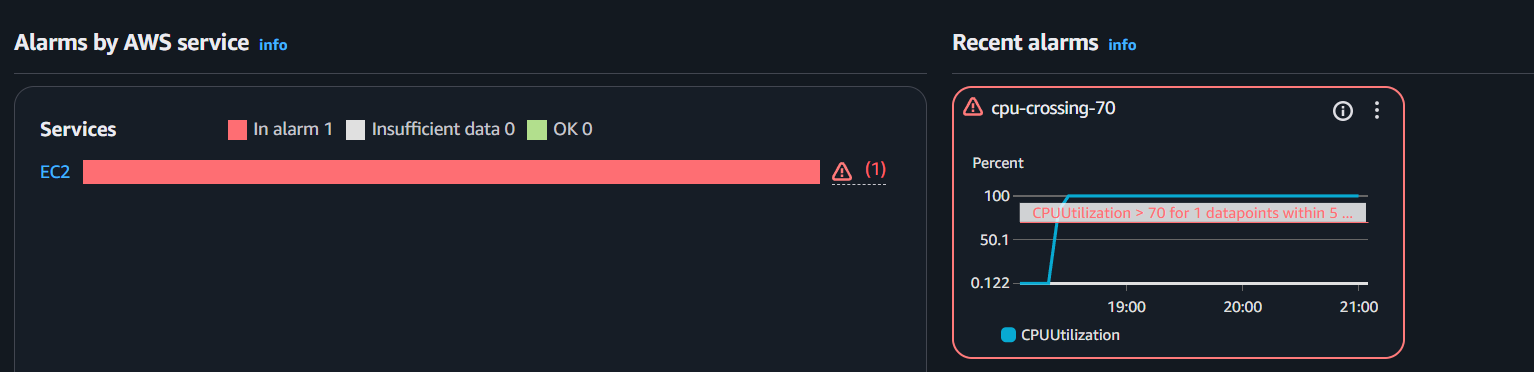
Apache installed and running

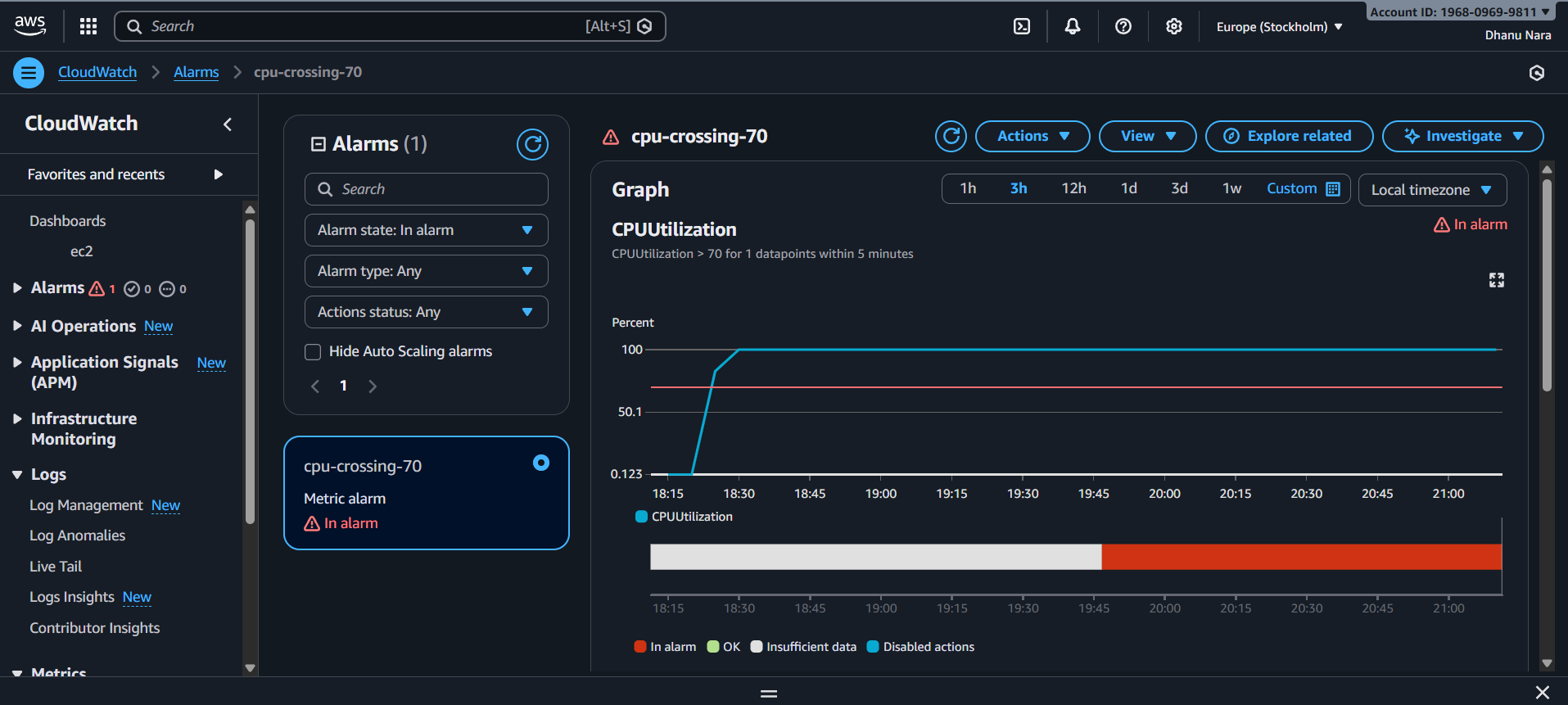
CloudWatch agent installed and configured to push logs/metrics

STEP-BY-STEP-IMPLIMENTATION:

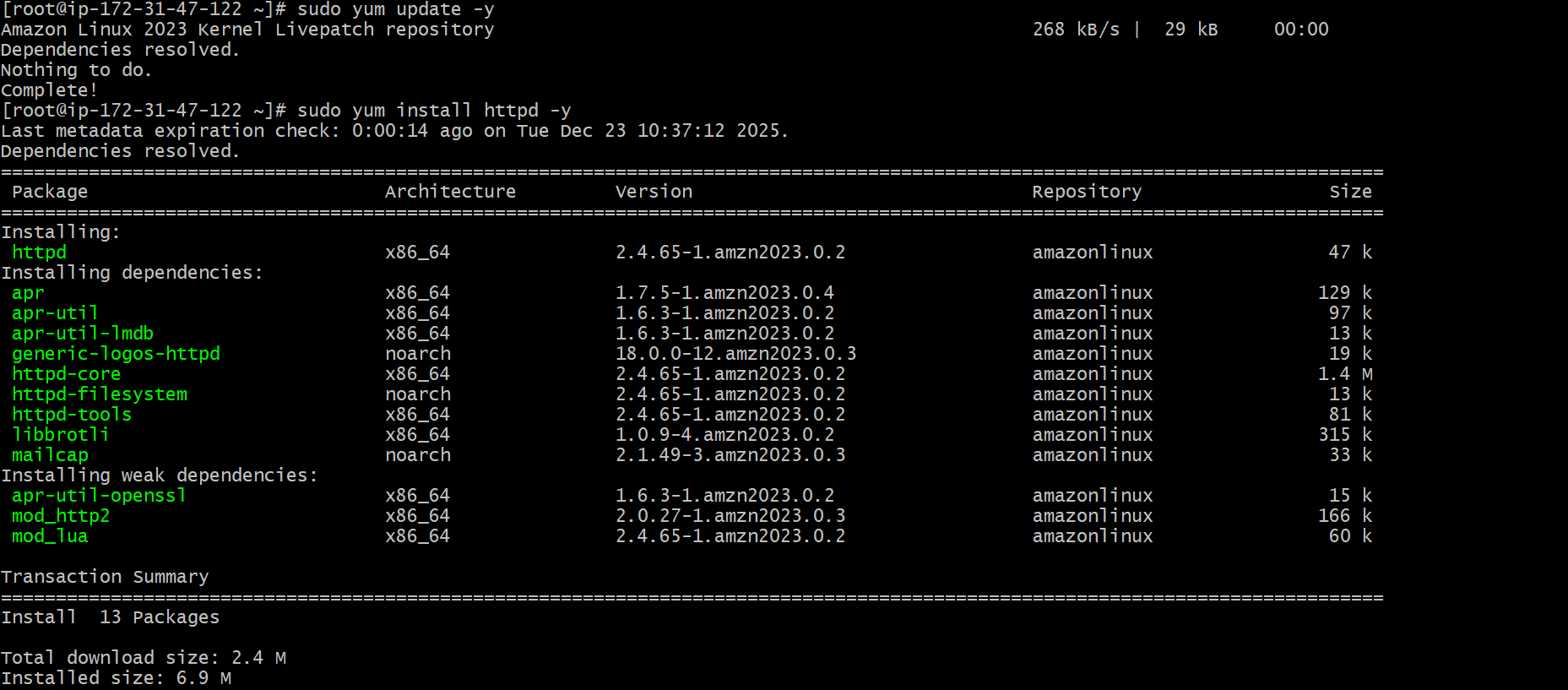
First create an EC2 instance and insert load on that to increase CPU utilization

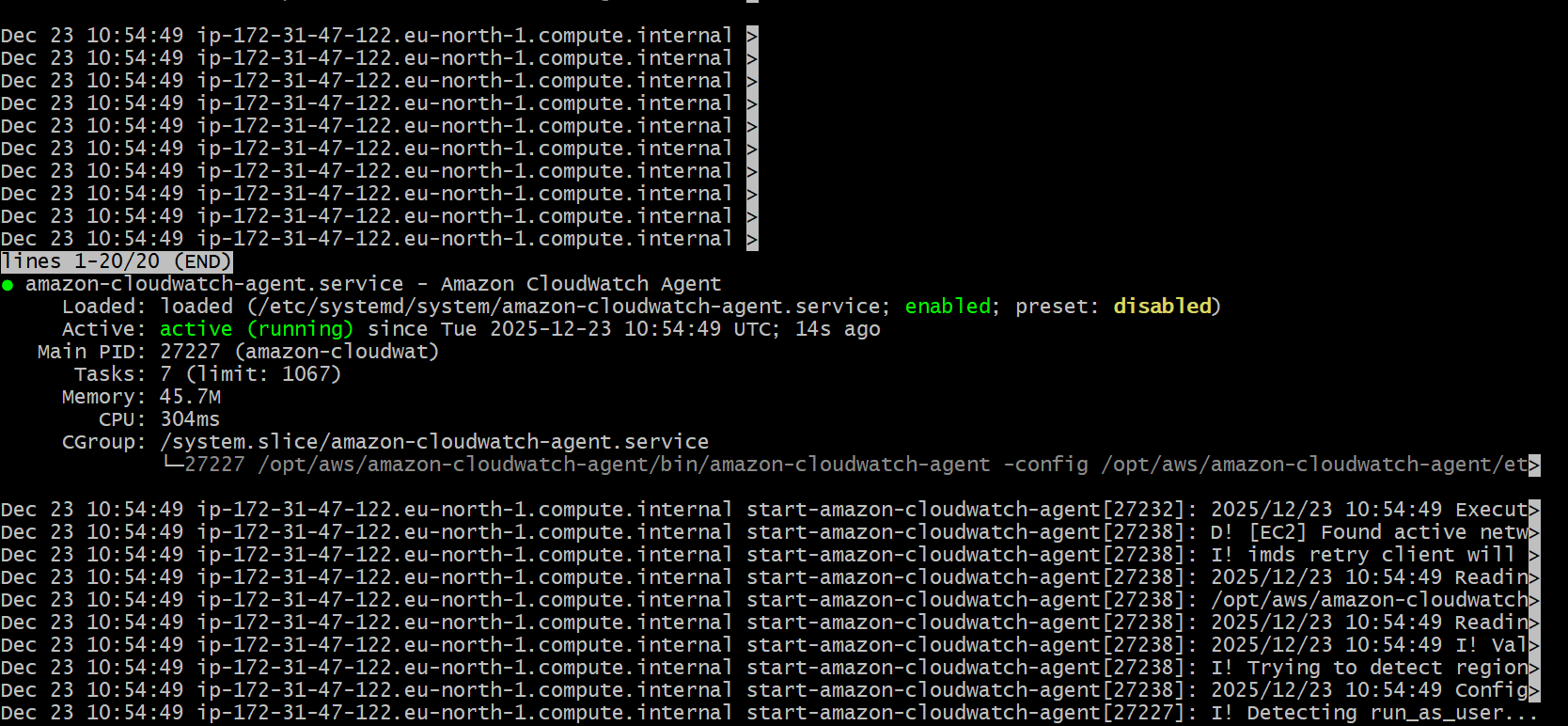
And create dashboard as well as alarm.



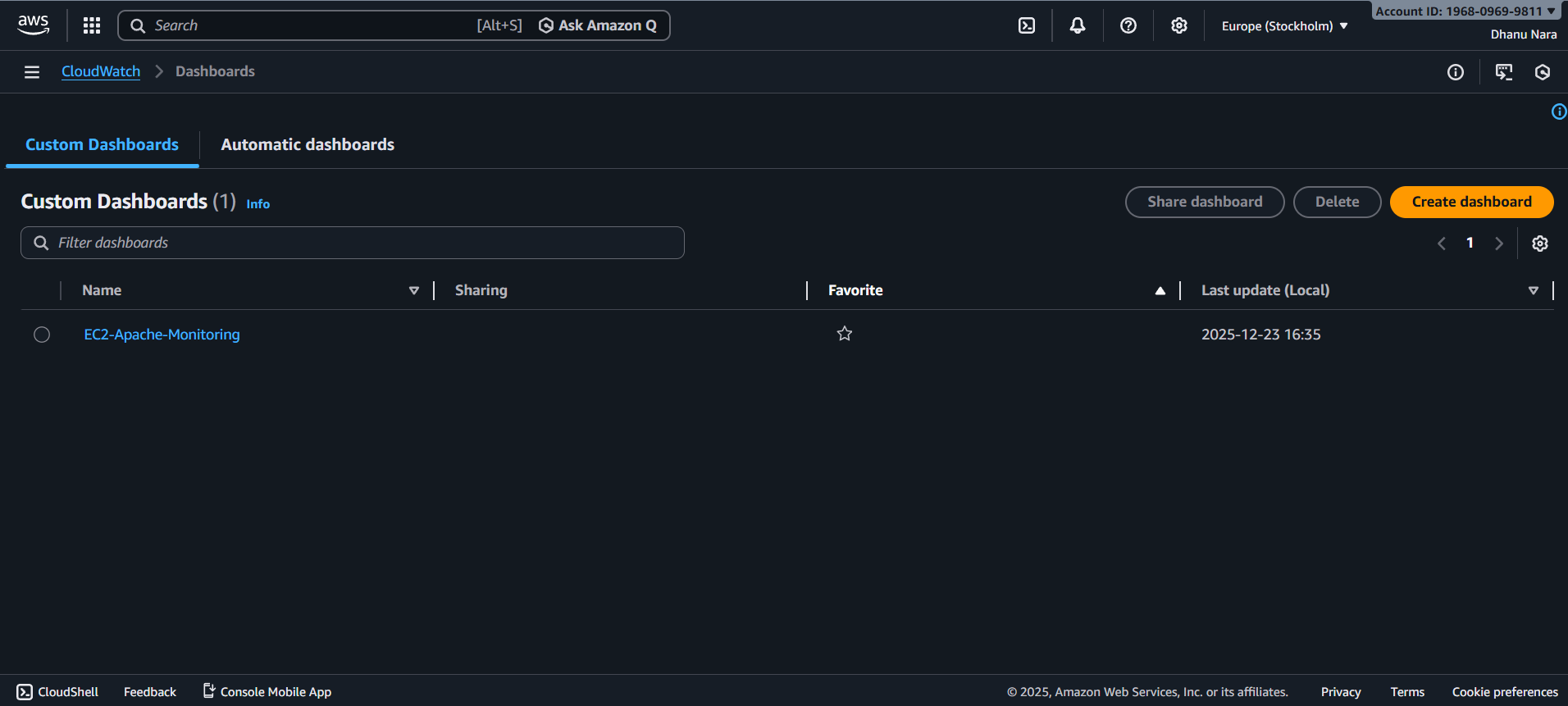


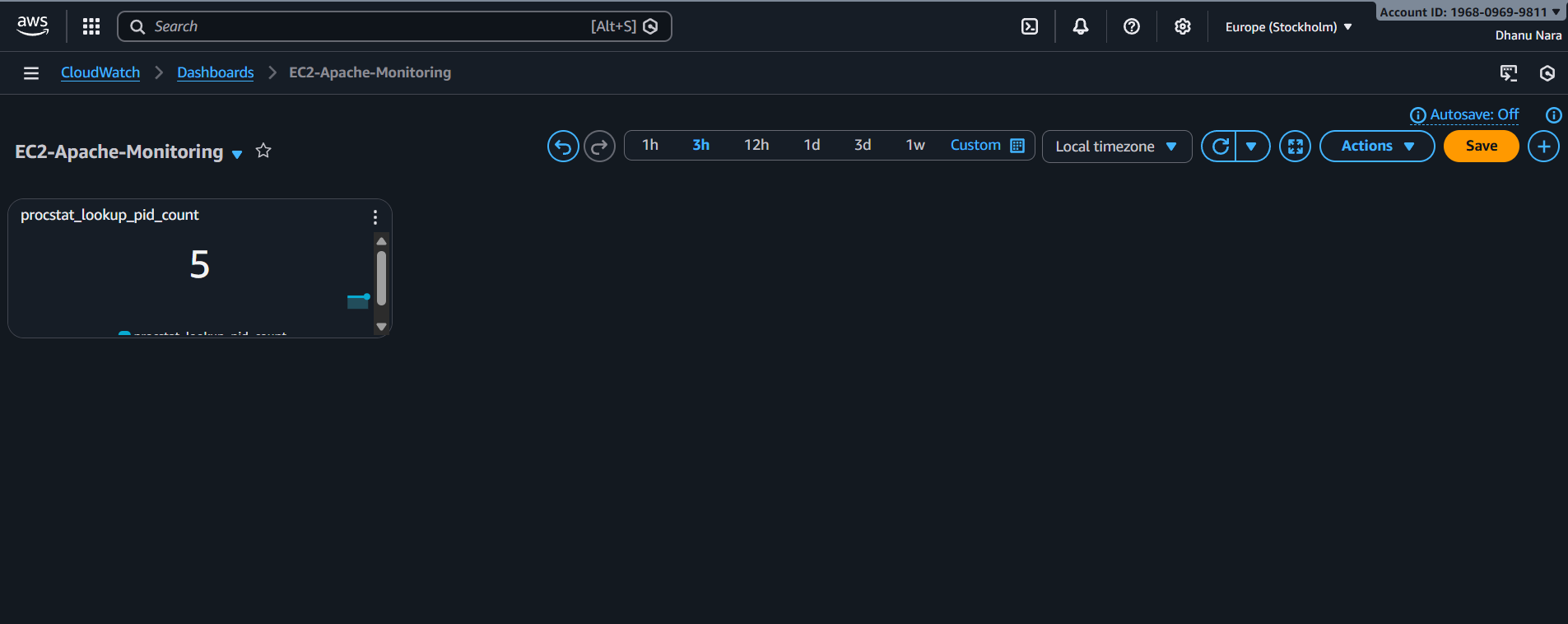
Installed Apache and pushed the metrics to CloudWatch





If the server is running the value will be more than 0 and if the service if not running the value will be 0





A screenshot of a computer

AI-generated content may be incorrect.

CONCLUSION:

CloudWatch dashboards created to track the CPU performance and Apache service health.

12.If CPU utilization is more than 70%,then it should trigger auto scaling and launch new instance.

TITLE:

CPU Monitoring

OBJECTIVE:

Enable Auto Scaling to launch a new instance automatically when CPU utilization exceeds 70%.

PREREQUISITES:

AWS account

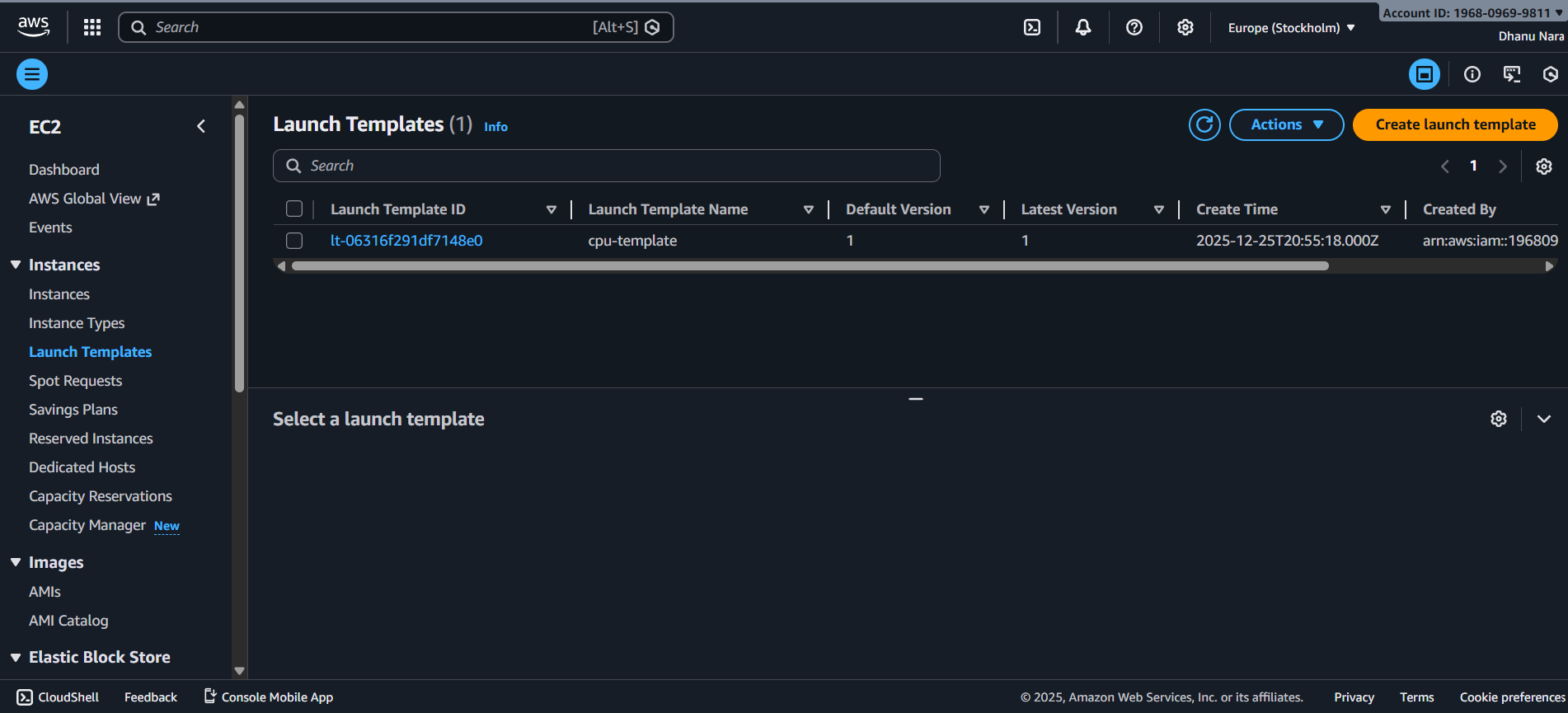
EC2 instance AMI or Launch Template

Auto scaling group has to be created

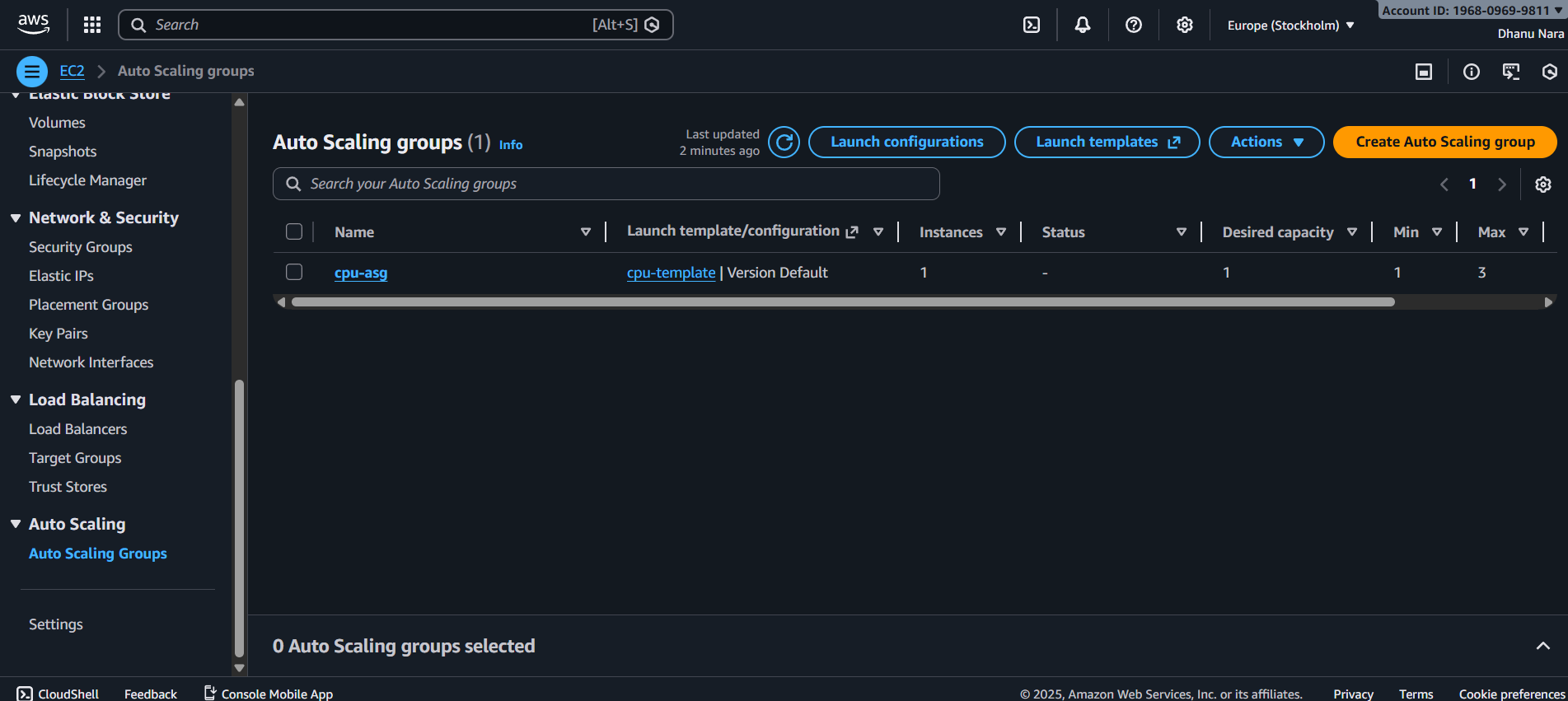
CloudWatch alarm for CPU utilization

STEP-BY-STEP-IMPLIMENTATION:

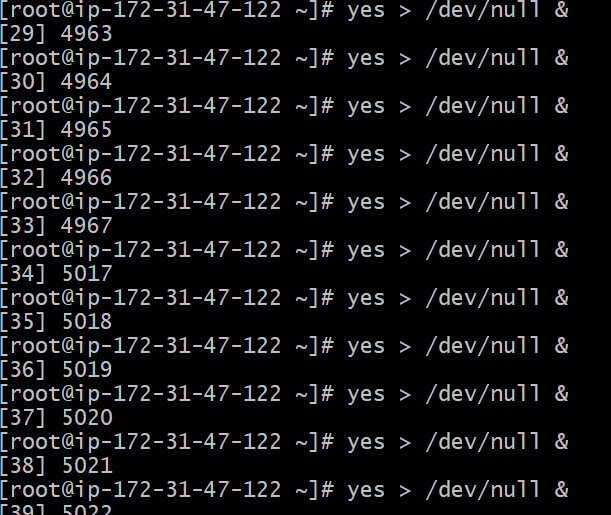
Cretaed launch template

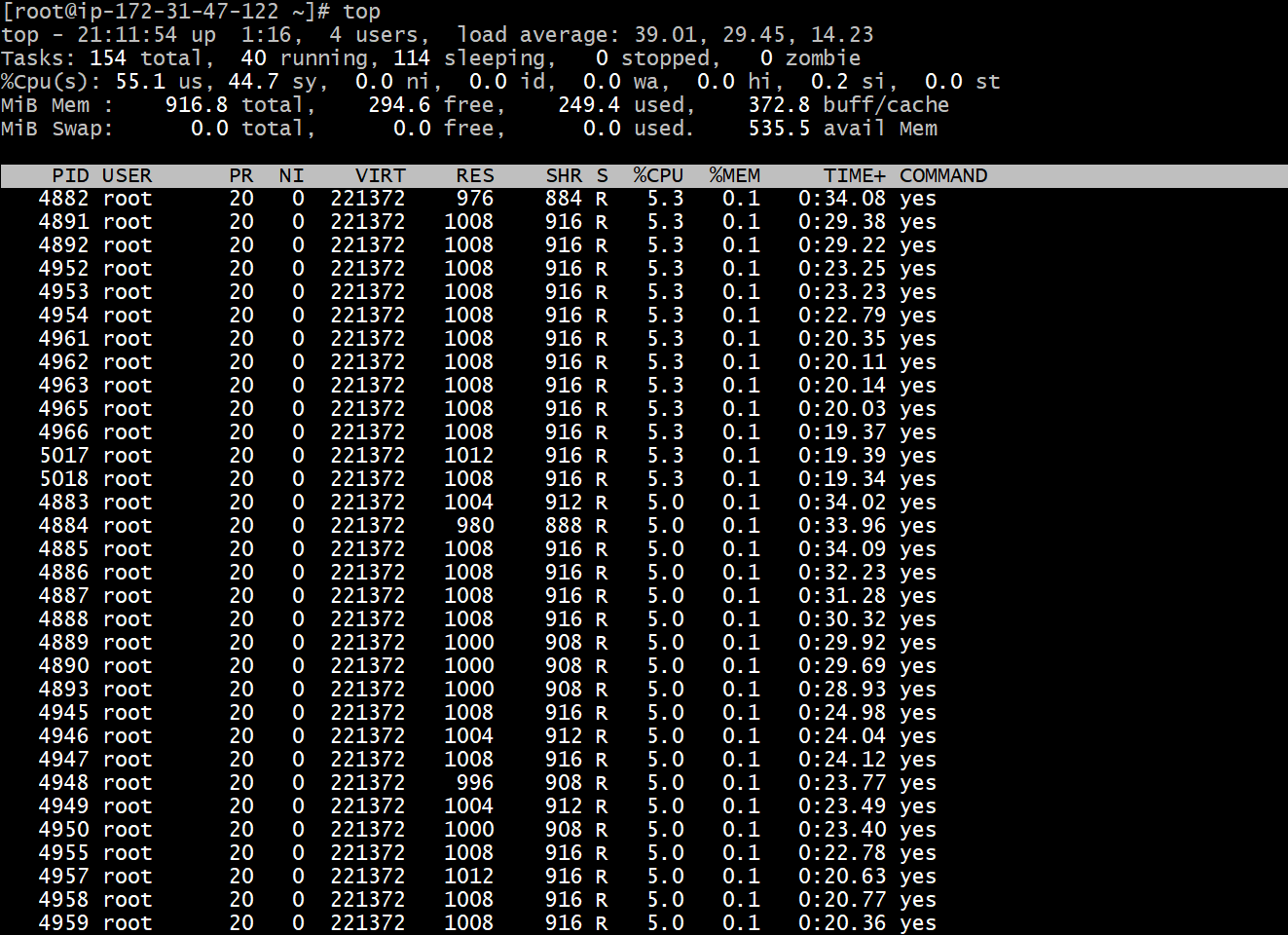


Created auto scaling group

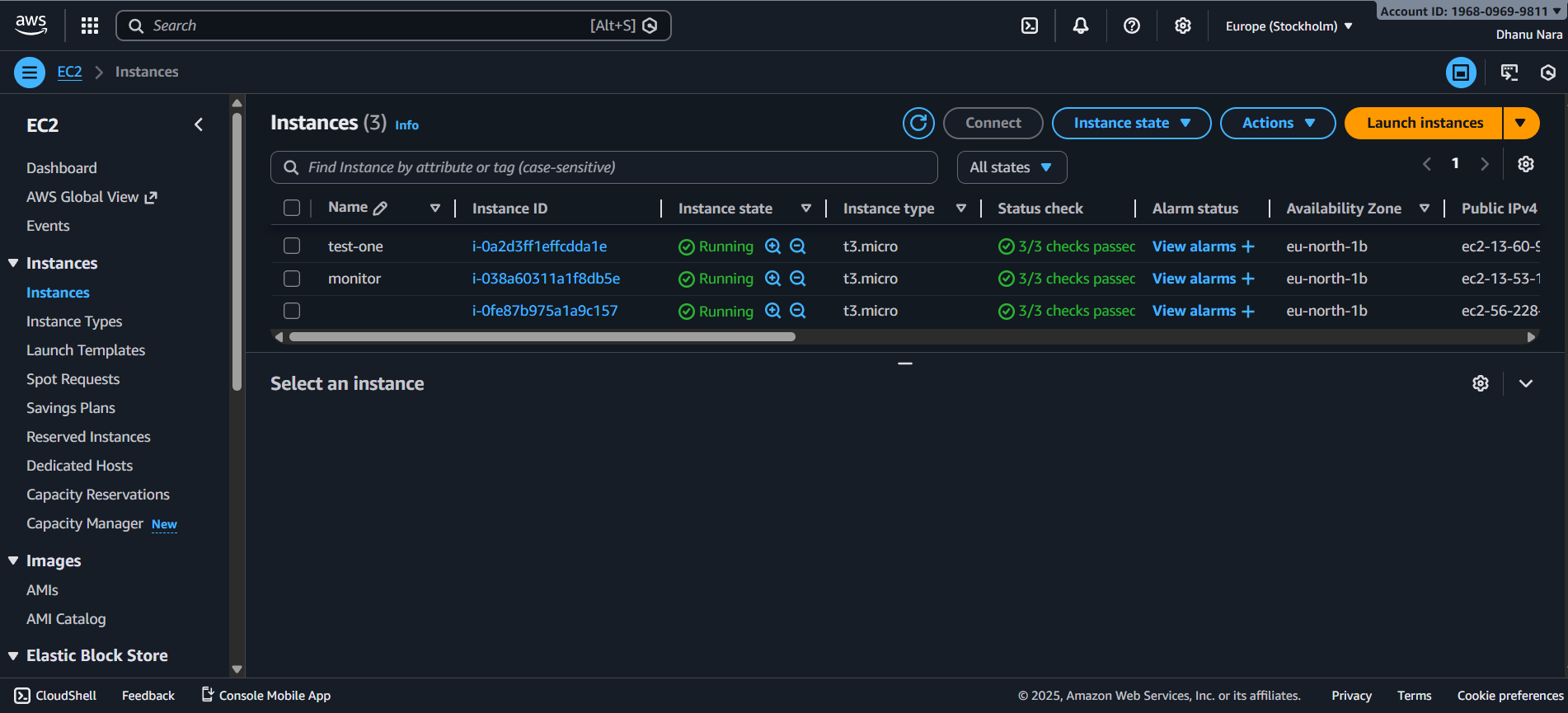


Launched an EC2 instance and created load on EC2 using yes > /dev/null &





Here automatically one new instance created



CONCLUSION:

When CPU exceeds **70%**, CloudWatch alarm triggers Auto Scaling policy and **creates a new EC2 instance automatically** to handle load.