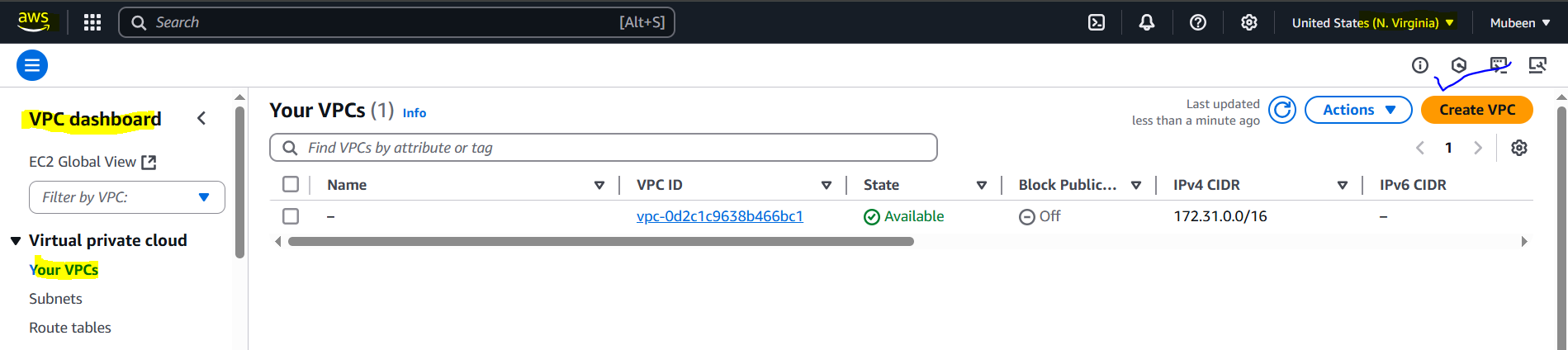
**Tasks on ASG**

**1) Create one vpc in N.virginia region.**

1. Log in to AWS Management Console.

2. Go to Services > VPC.

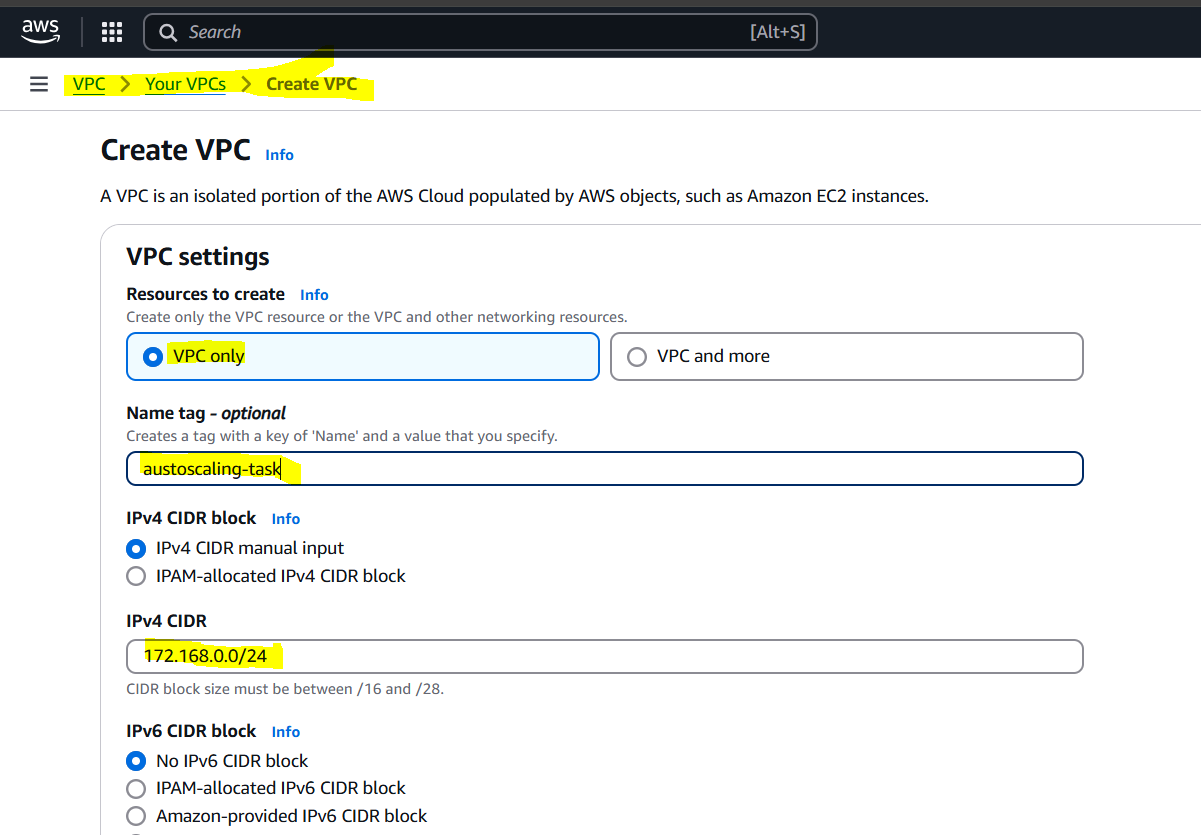
3. Click "Create VPC".



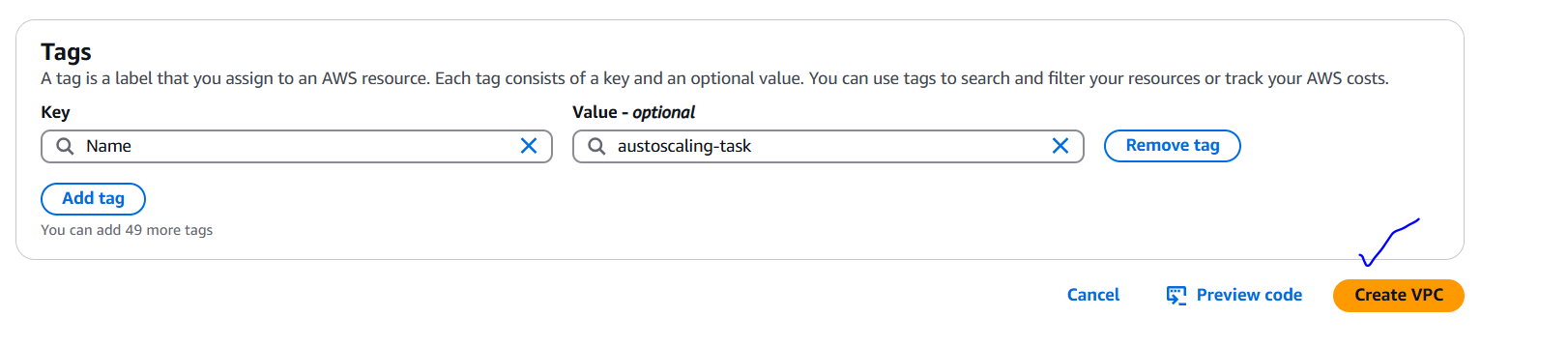
4. Enter:

- IPv4 CIDR block: 172.168.0.0/24.

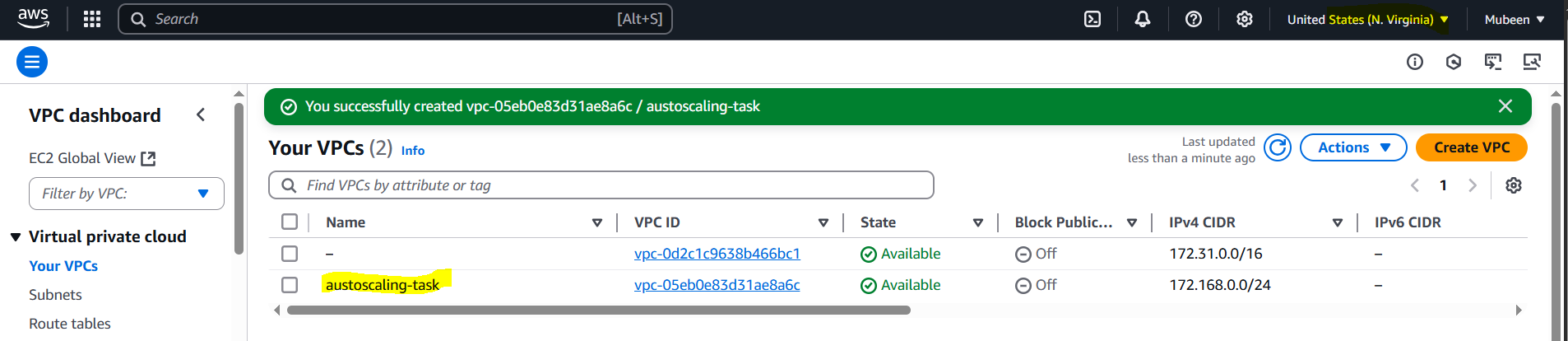
- VPC name: austoscaling-task



5. Click "Create VPC".



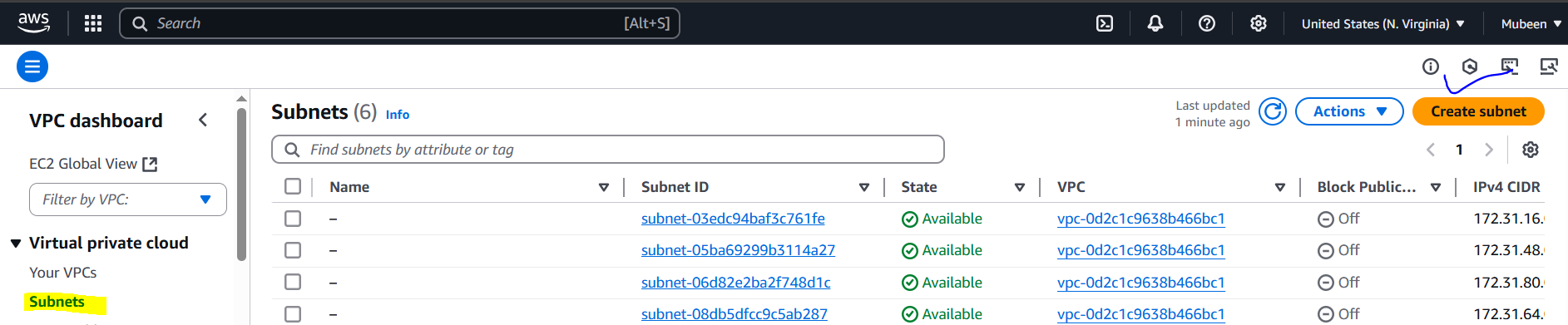
6. Verify VPC creation in "Your VPCs" section.



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

1. Go to Services > VPC > Subnets.

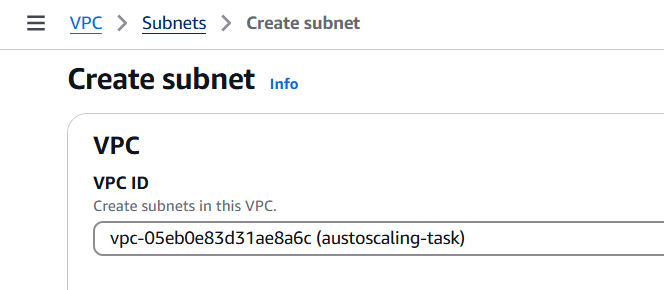
2. Click "Create subnet".



**Public Subnet**

1. Enter:

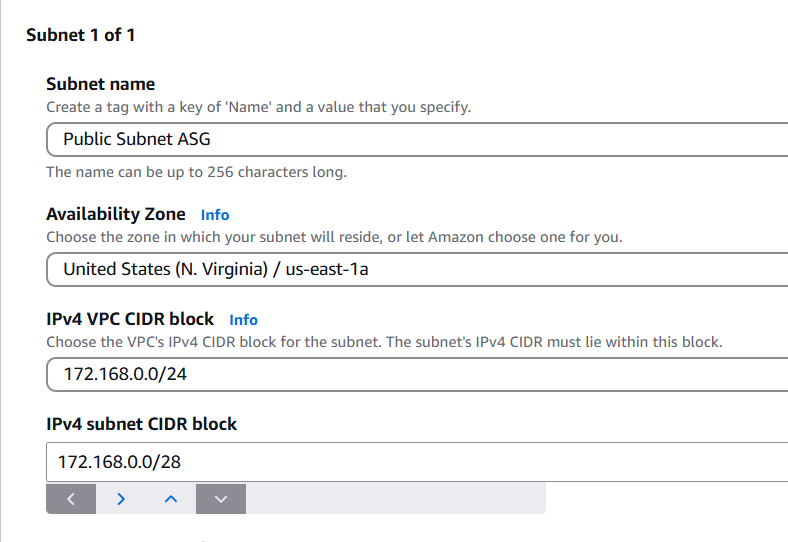
- VPC: Select "austoscaling-task" VPC.



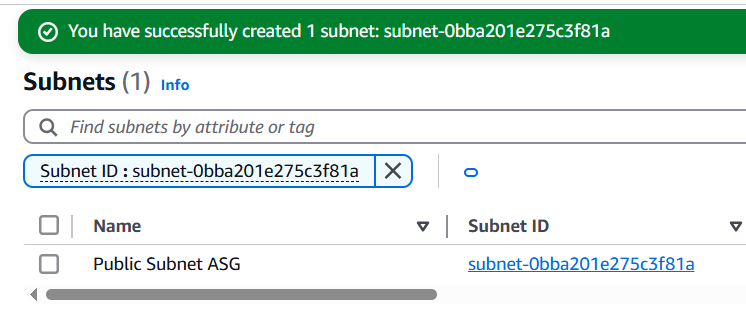
- Subnet name: Public Subnet ASG.

- Availability Zone: Choose an AZ (us-east-1a).

- IPv4 CIDR block: 172.168.0.0/28.



2. Click "Create subnet".

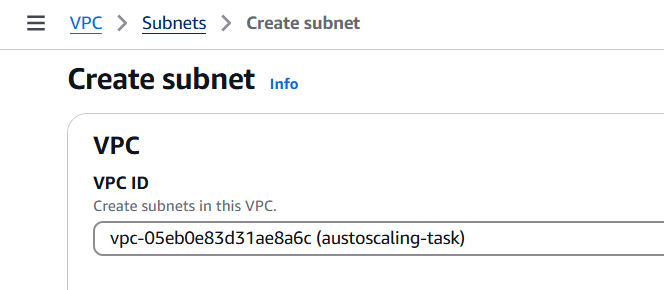


**Private Subnet**

1. Click "Create subnet" again.

2. Enter:

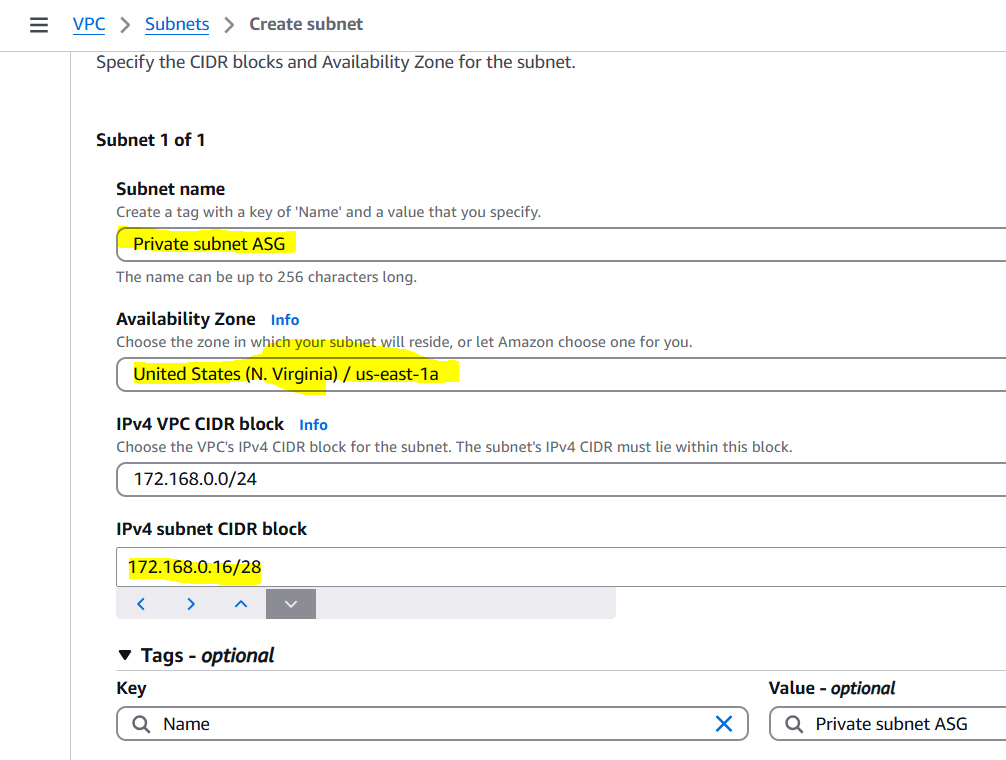
- VPC: Select "austoscaling-task" VPC.



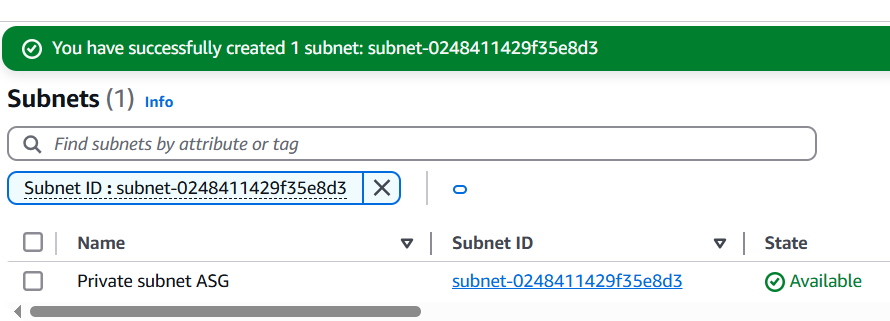
- Subnet name: Private Subnet ASG.

- Availability Zone: Choose the same AZ (us-east-1a).

- IPv4 CIDR block: (link unavailable).



3. Click "Create subnet".



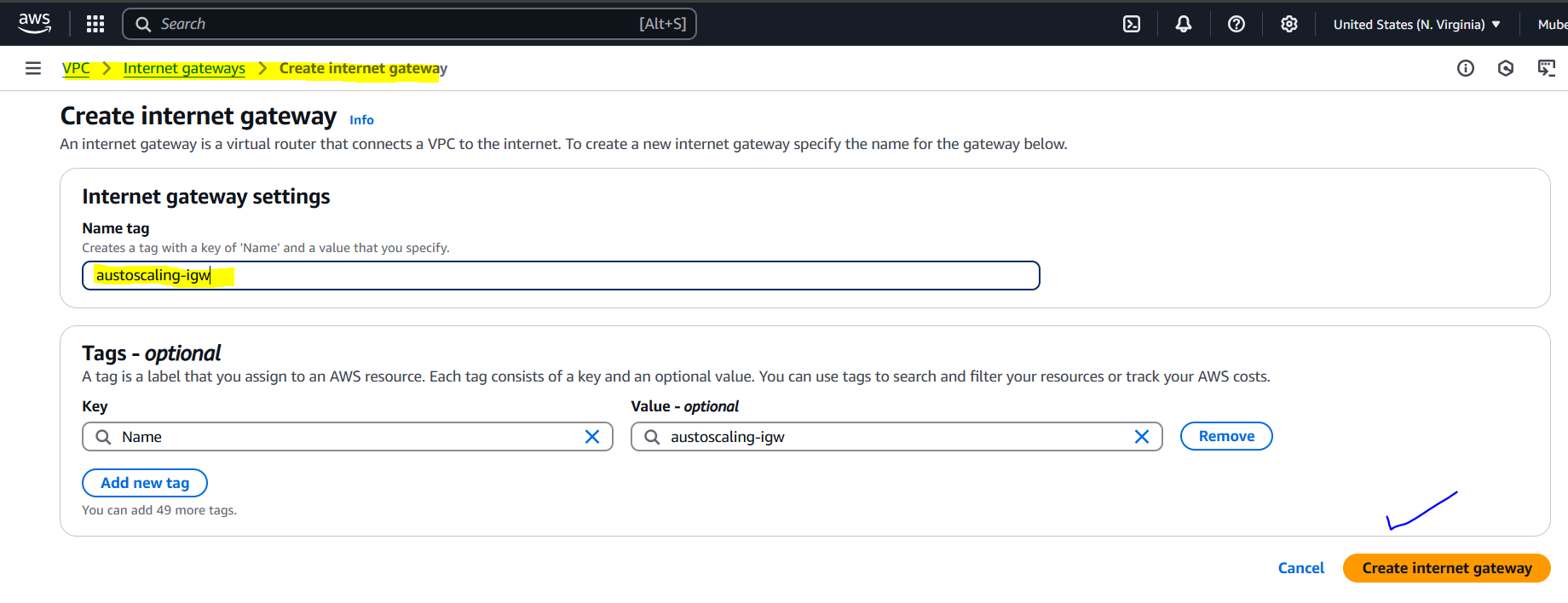
**3) Provide the IGW to the vpc.**1. Go to Services > VPC > Internet Gateways.

2. Click "Create internet gateway".

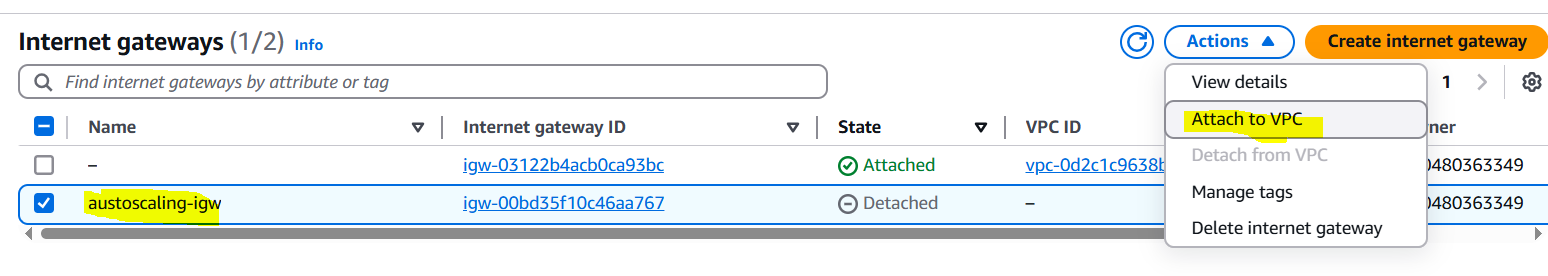
3. Enter:

- Name: austoscaling-igw

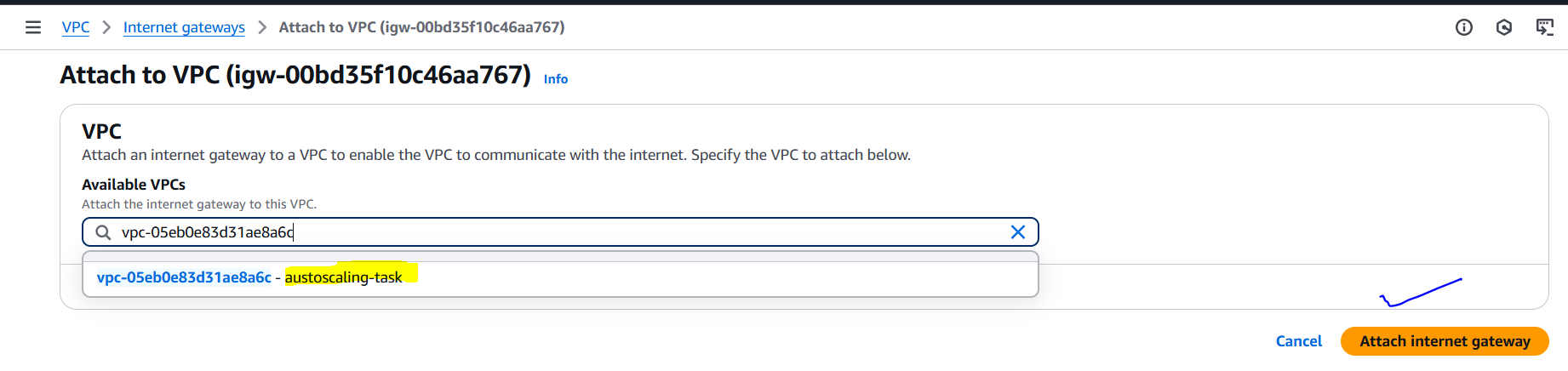
4. Click "Create internet gateway".

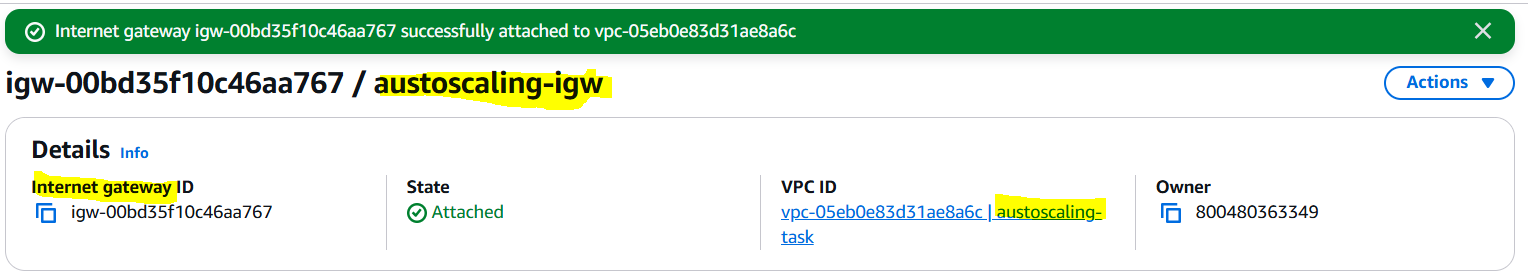


5. Select the newly created IGW and click "Actions" > "Attach to VPC".



6. Select the "austoscaling-task" VPC and click "Attach internet gateway".





**4) Create One public RT and one private RT.**  
**Public Route Table**

1. Go to Services > VPC > Route Tables.

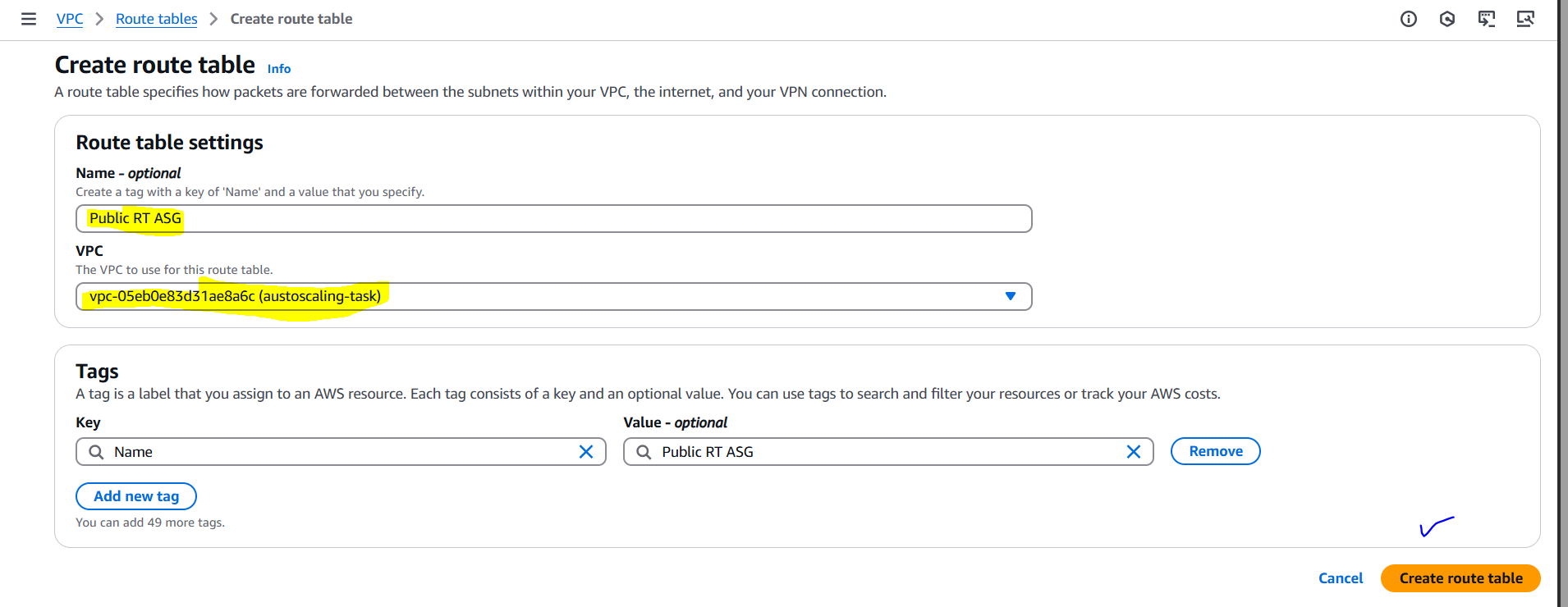
2. Click "Create route table".

3. Enter:

- Name: Public RT ASG

- VPC: Select "autoscaling-task" VPC

4. Click "Create route table".



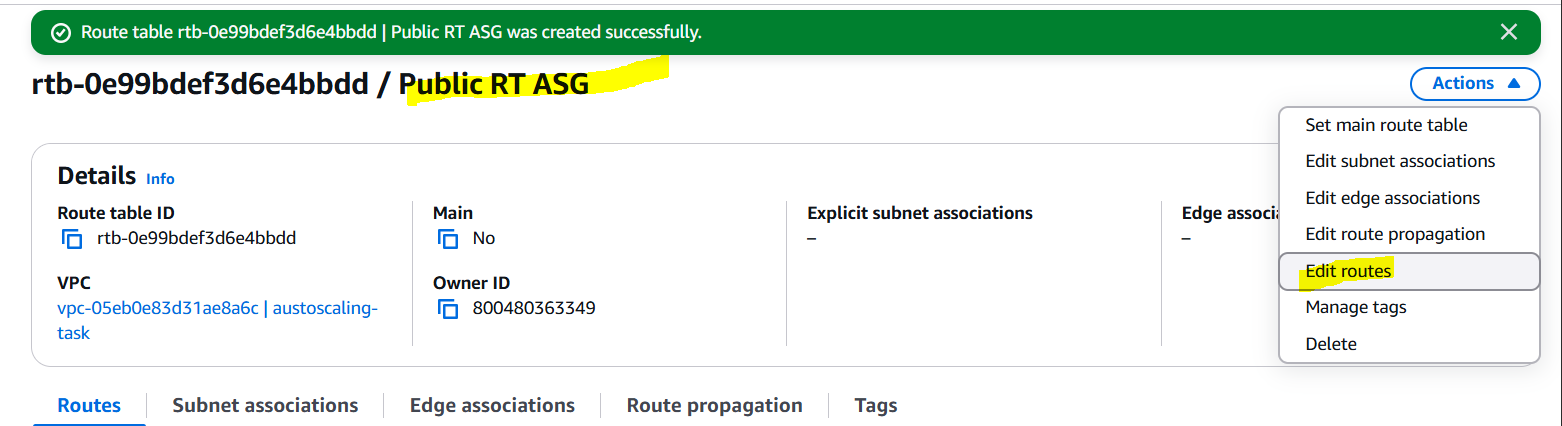
5. Select the Public RT and click "Actions" > "Edit routes".

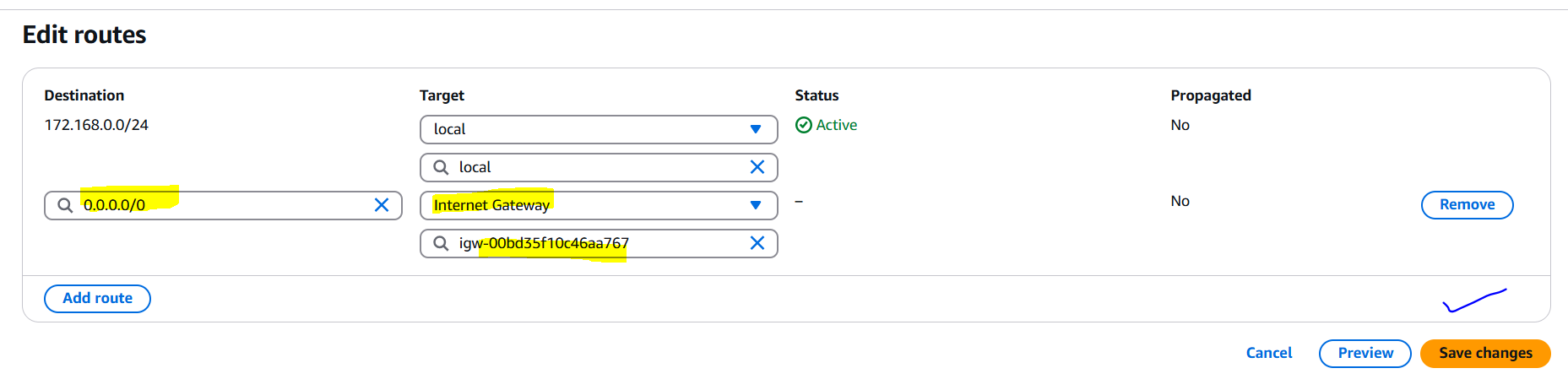
6. Click "Add route" and enter:

- Destination: 0.0.0.0/0

- Target: Select the IGW (austoscaling-igw)

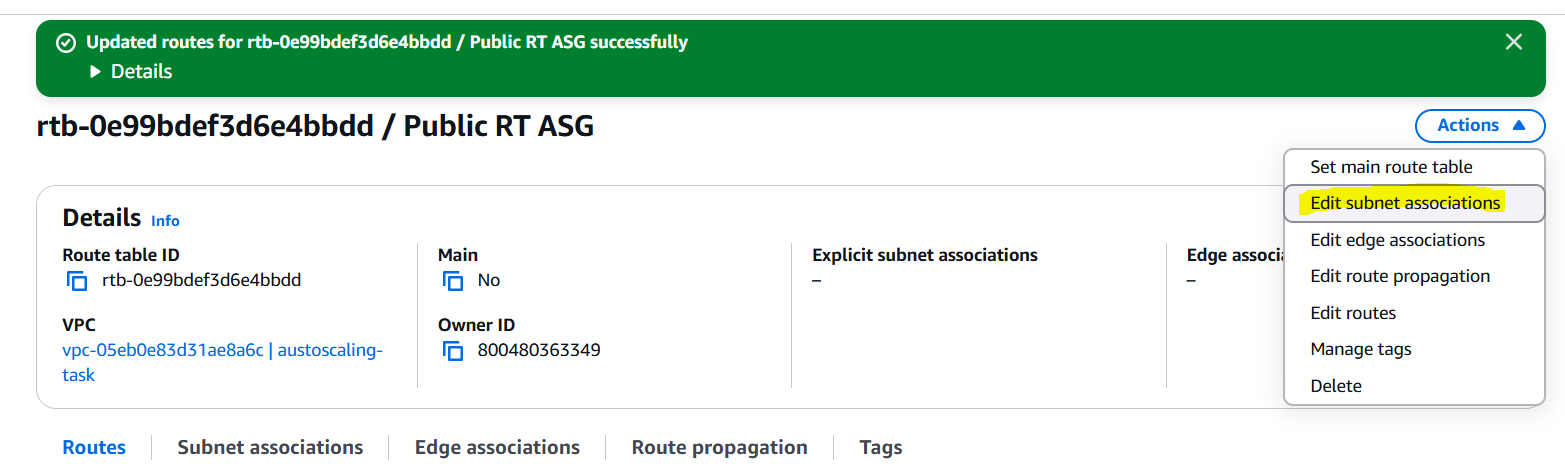
7. Click "Save routes".





8. Select the Public RT and click "Actions" > "Edit subnet associations".

9. Select the public-subnet and click "Save".







**Private Route Table**

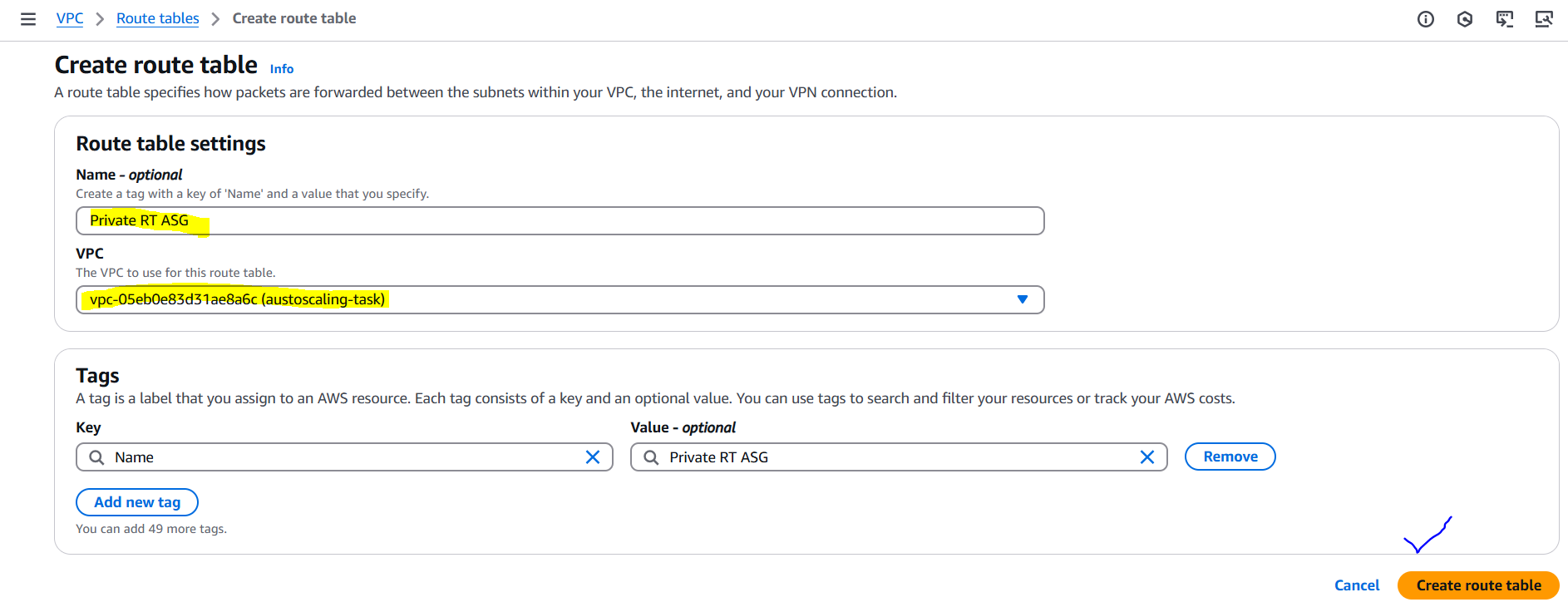
1. Click "Create route table" again.

2. Enter:

- Name: Private RT ASG

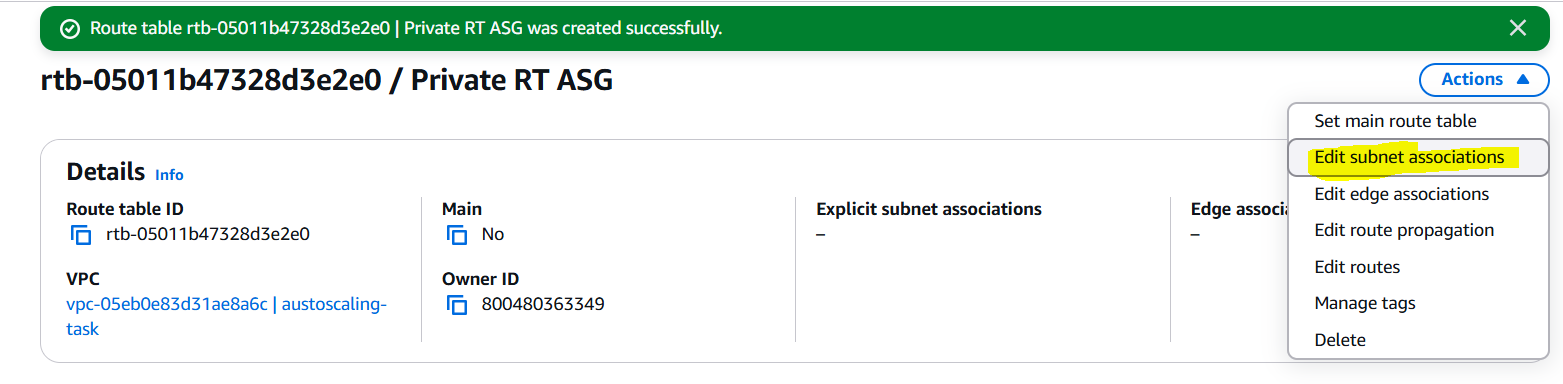
- VPC: Select "austoscaling-task" VPC

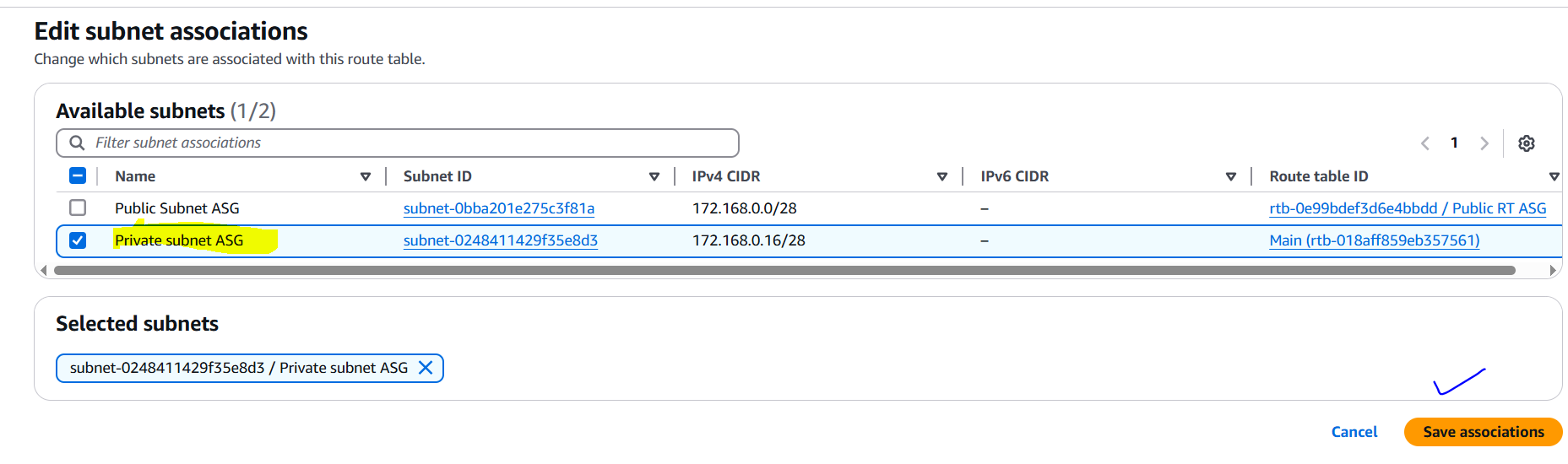
3. Click "Create route table".

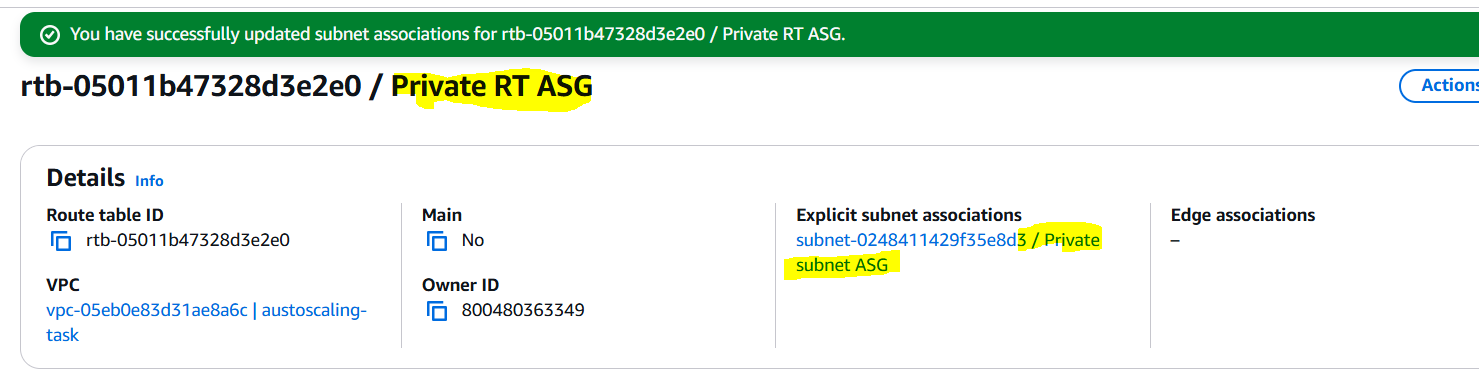


4. Select the Private RT ASG and click "Actions" > "Edit subnet associations".

5. Select the private-subnet and click "Save".







**5) Deploy NAT gateway on public subnet and attach the NAT gateway to private subnet.   
  
Create NAT Gateway**

1. Go to Services > VPC > NAT Gateways.

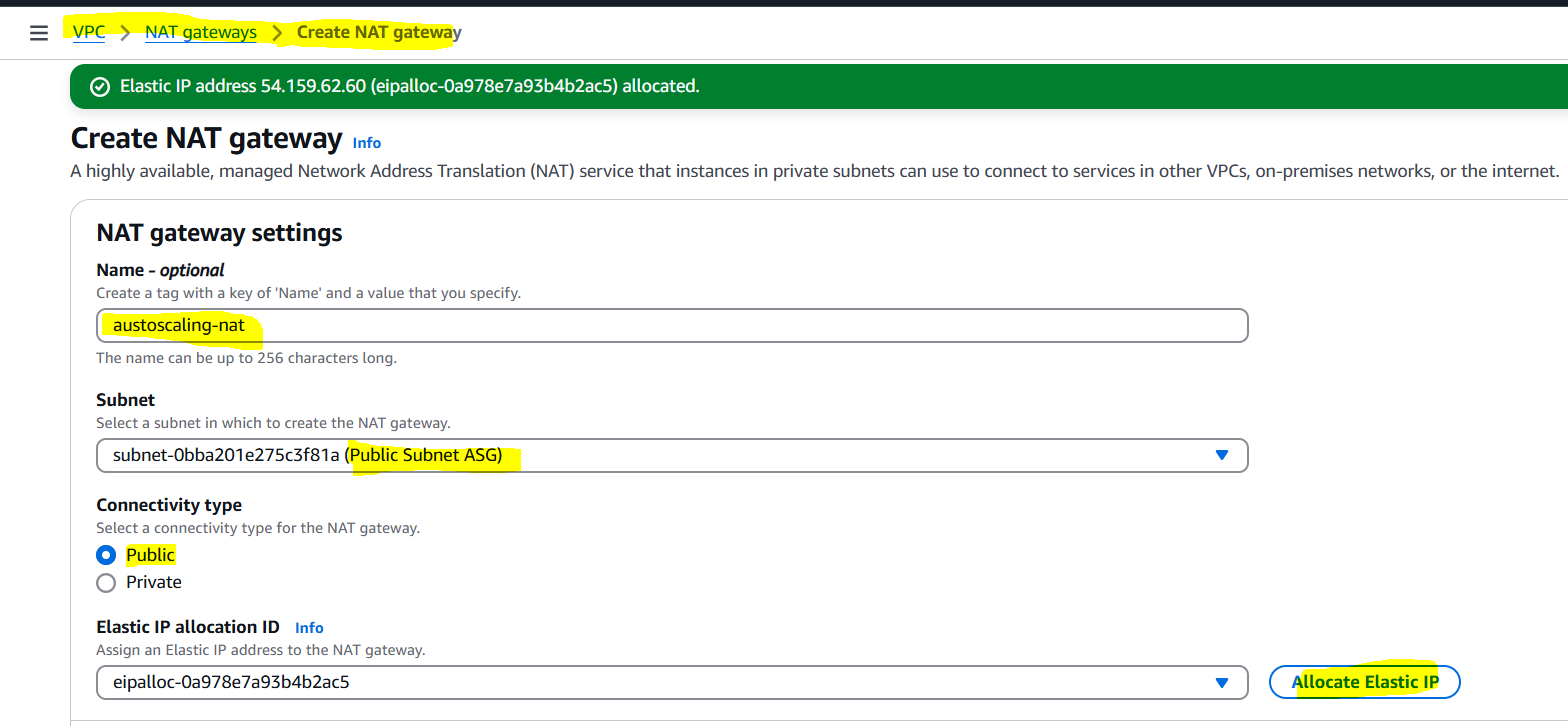
2. Click "Create NAT gateway".

3. Enter:

- Name: austoscaling-nat

- Subnet: Select public-subnet

- Connectivity type: Public



4. Click "Create a NAT gateway".

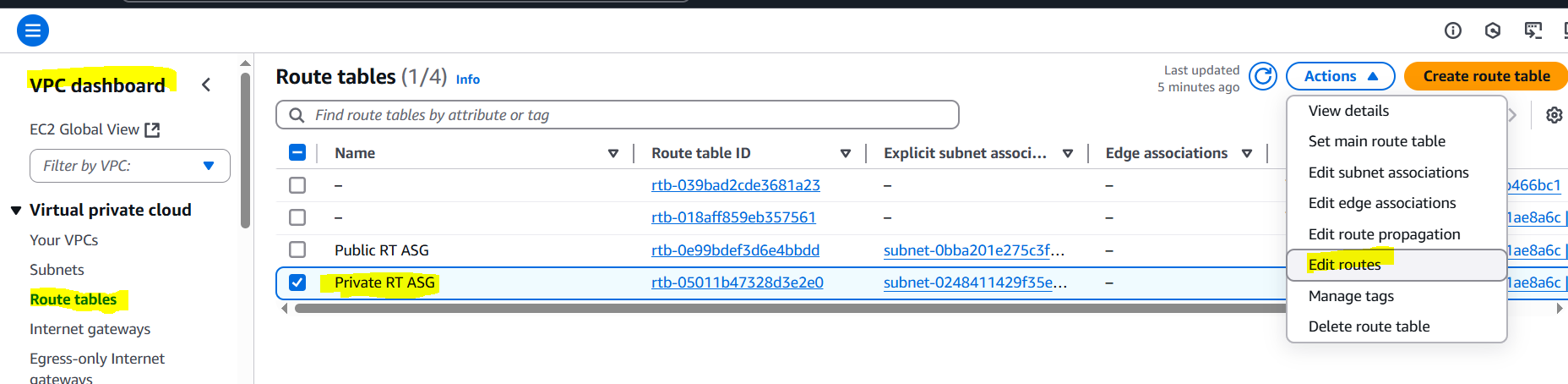


**Update Private Route Table**

1. Go to Services > VPC > Route Tables.

2. Select the Private RT ASG.

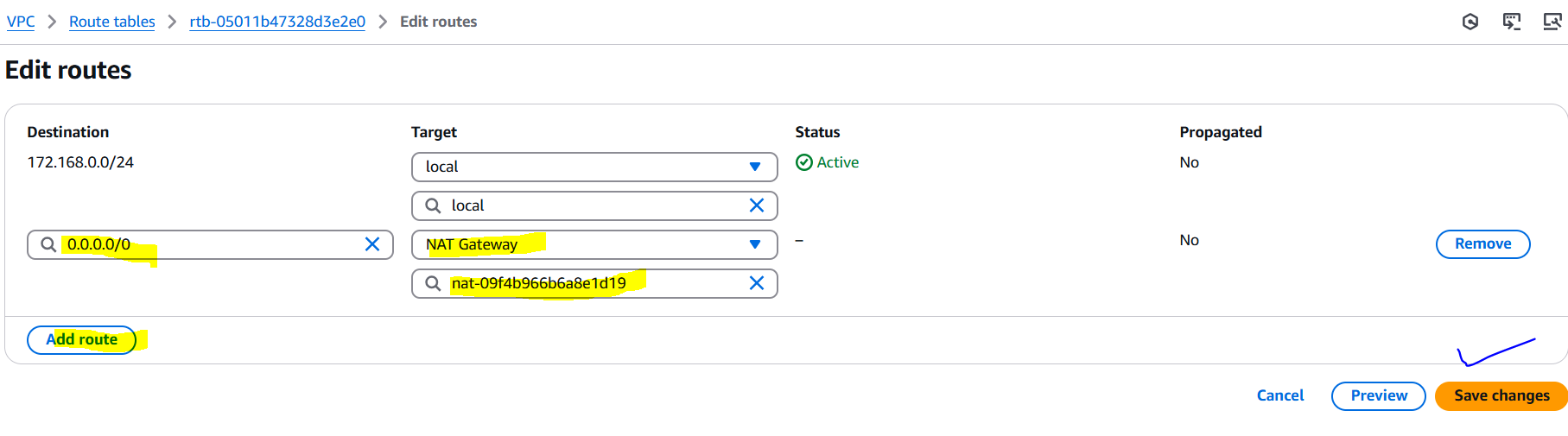
3. Click "Actions" > "Edit routes".



4. Click "Add route" and enter:

- Destination: 0.0.0.0/0

- Target: Select the NAT gateway (austoscaling-nat)

5. Click "Save routes".  


**6) Create Two instances, one in public subnet and one in private subnet.**All settings are taken as default.

**Public Instance**

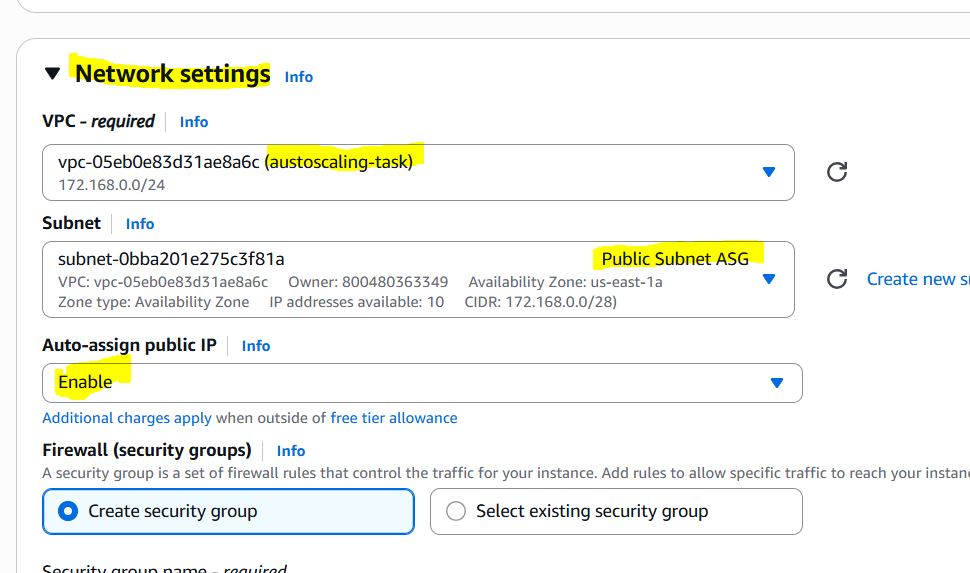
1. Launch instance in public-subnet.

2. Network settings:

- VPC: austoscaling-task

- Subnet: public-subnet

- Auto-assign public IP: Enable



**Private Instance**

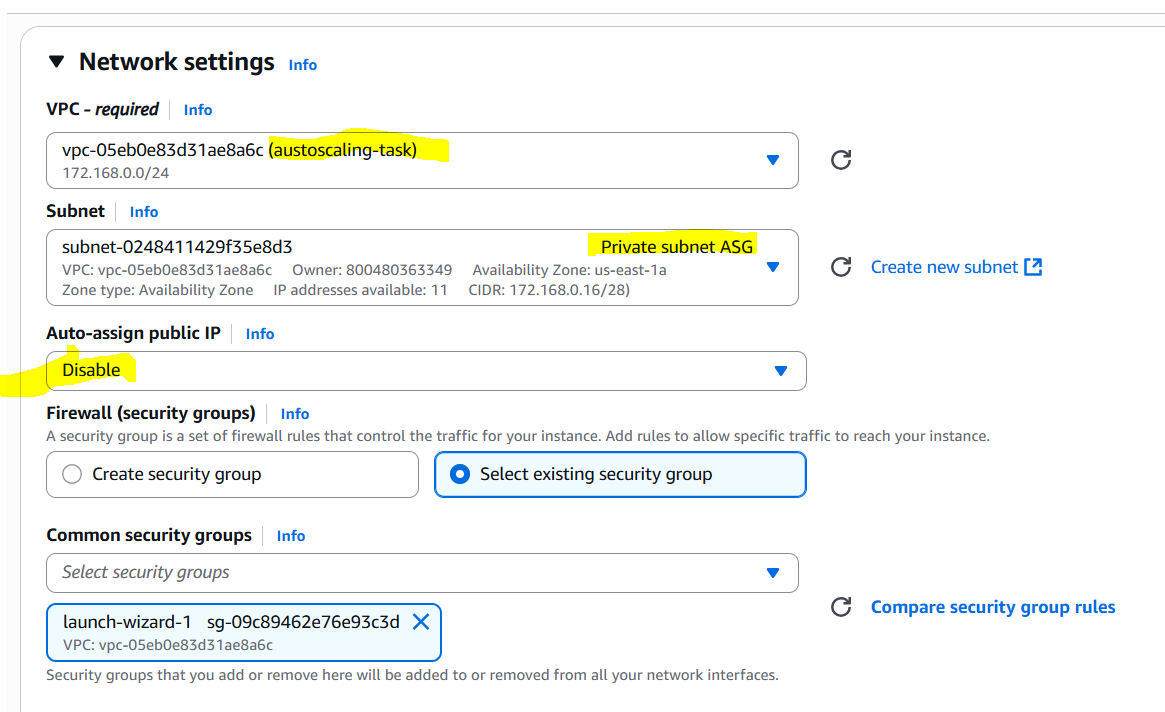
1. Launch instance in private-subnet.

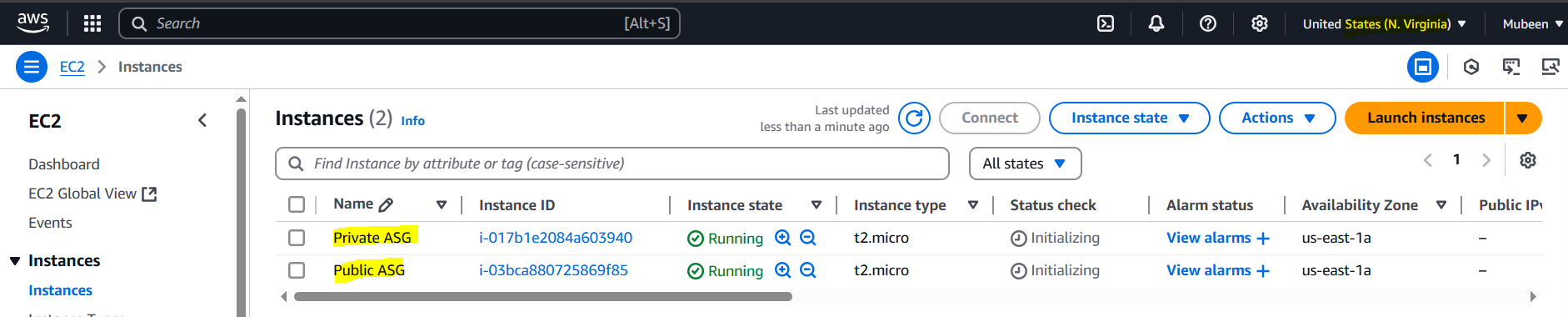
2. Network settings:

- VPC: austoscaling-task

- Subnet: private-subnet

- Auto-assign public IP: Disable







**7) Deploy Apache server on both the ec2 instances with sample index.html file.   
  
Run the following script on both instances:**

#!/bin/bash

# Install Apache

sudo yum update -y

sudo yum install httpd -y

# Start Apache

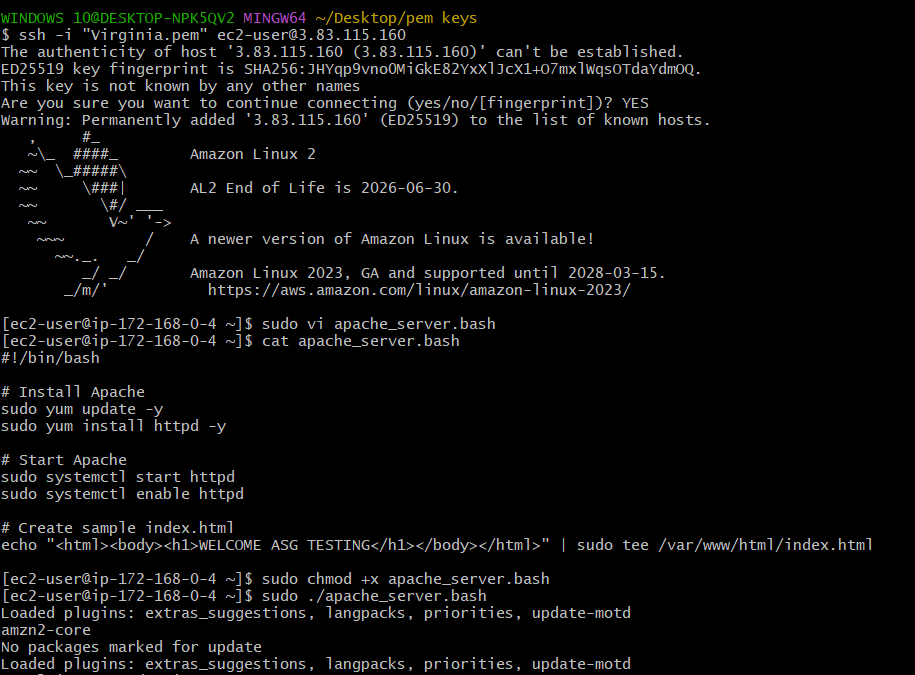
sudo systemctl start httpd

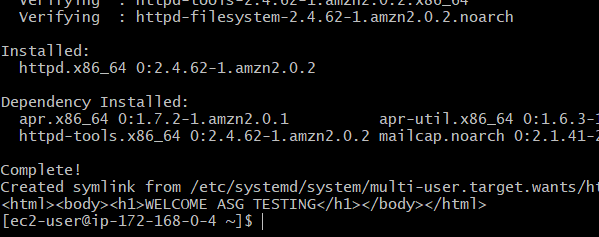
sudo systemctl enable httpd

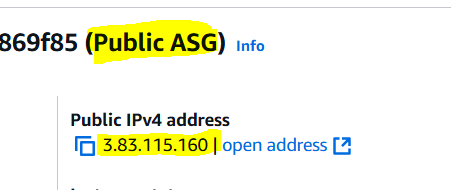
# Create sample index.html

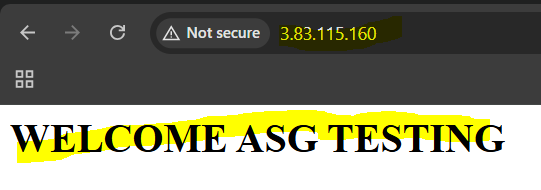
echo "<html><body><h1>WELCOME ASG TESTING</h1></body></html>" | sudo tee /var/www/html/index.html

**Executing script in Public instance:**



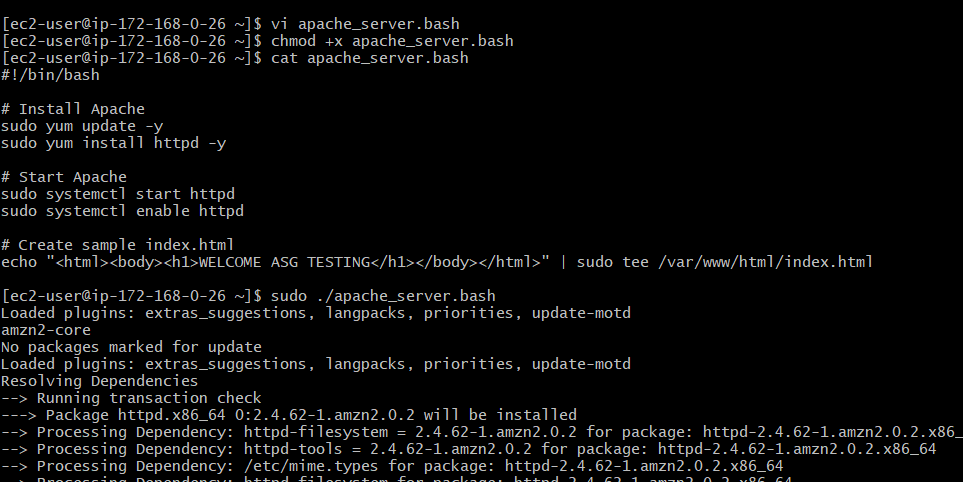


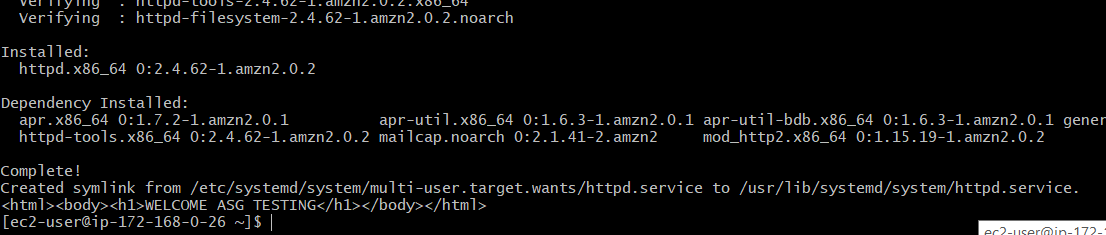


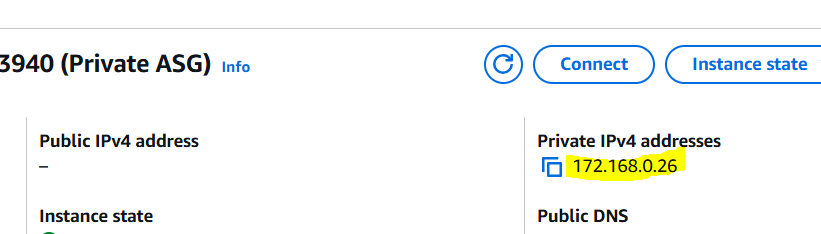


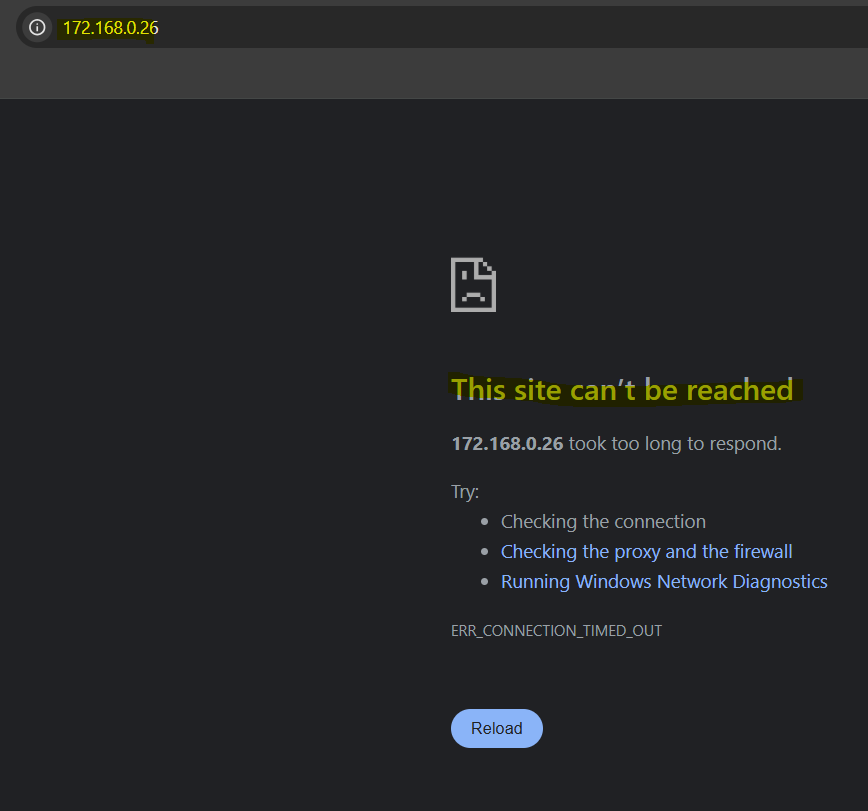
**Executing script in Private instance:**

Login in private instance through public instance then executed the script.  







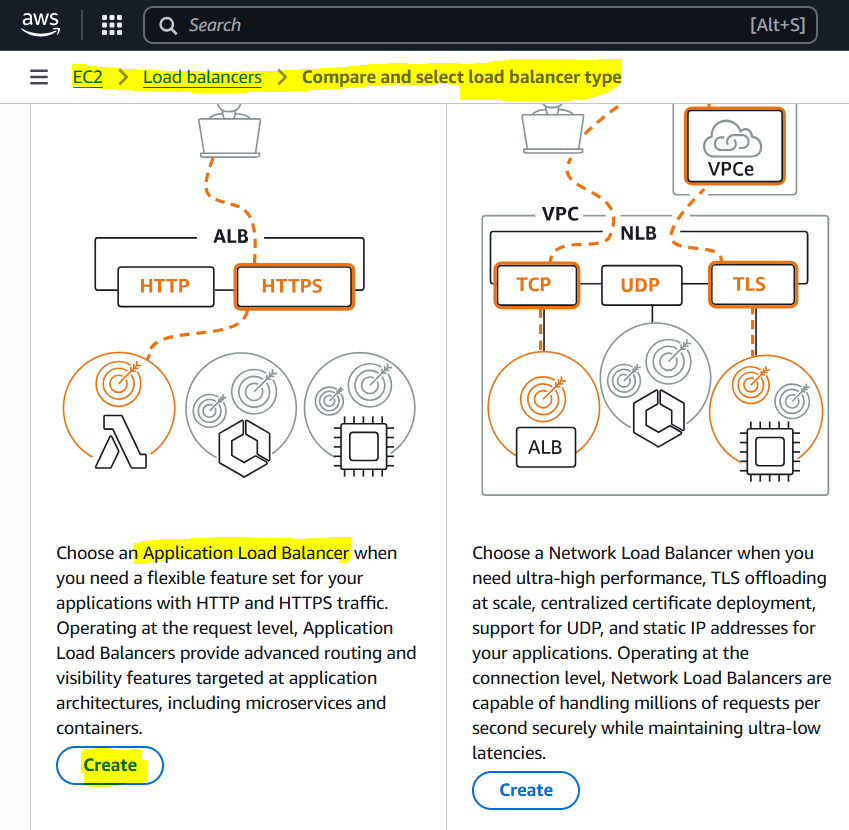
As it is a private instance there no public access.

**8) Create one application load balancer and attach the load balancer to both the ec2 instances.   
  
Create Application Load Balancer**

1. Go to Services > EC2 > Load Balancers.

2. Click "Create load balancer".

3. Select "Application Load Balancer".

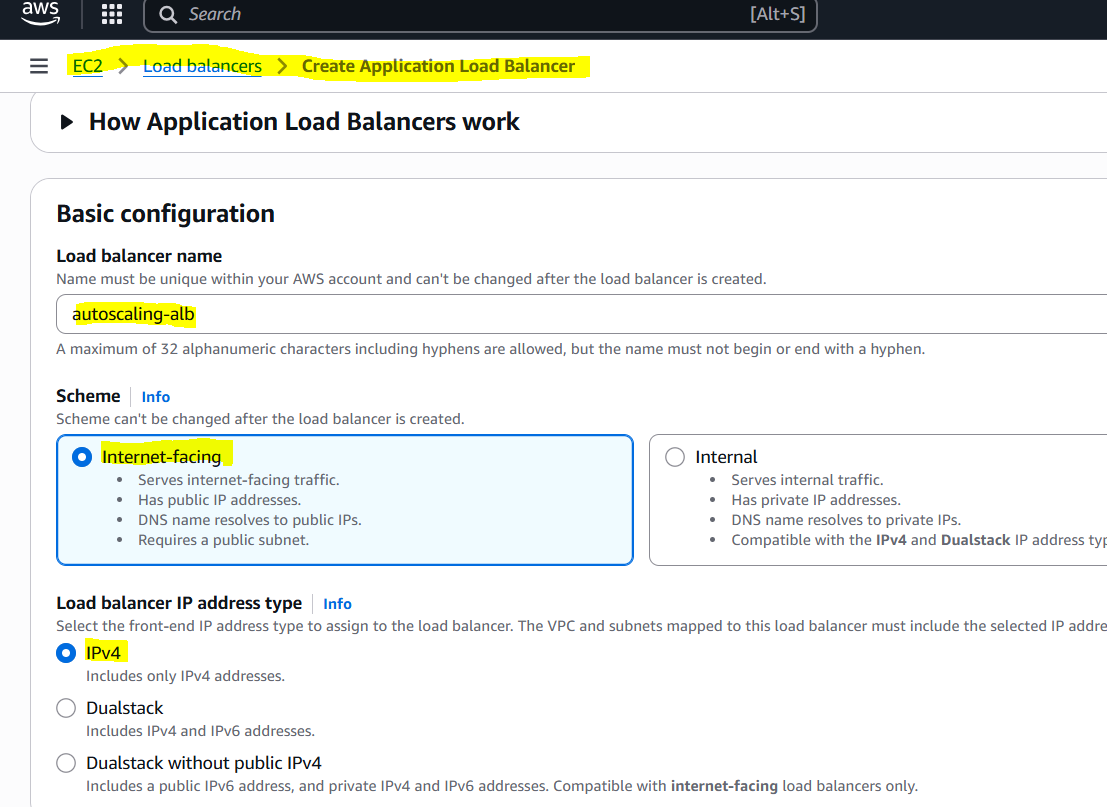


4. Configure:

- Name: austoscaling-alb

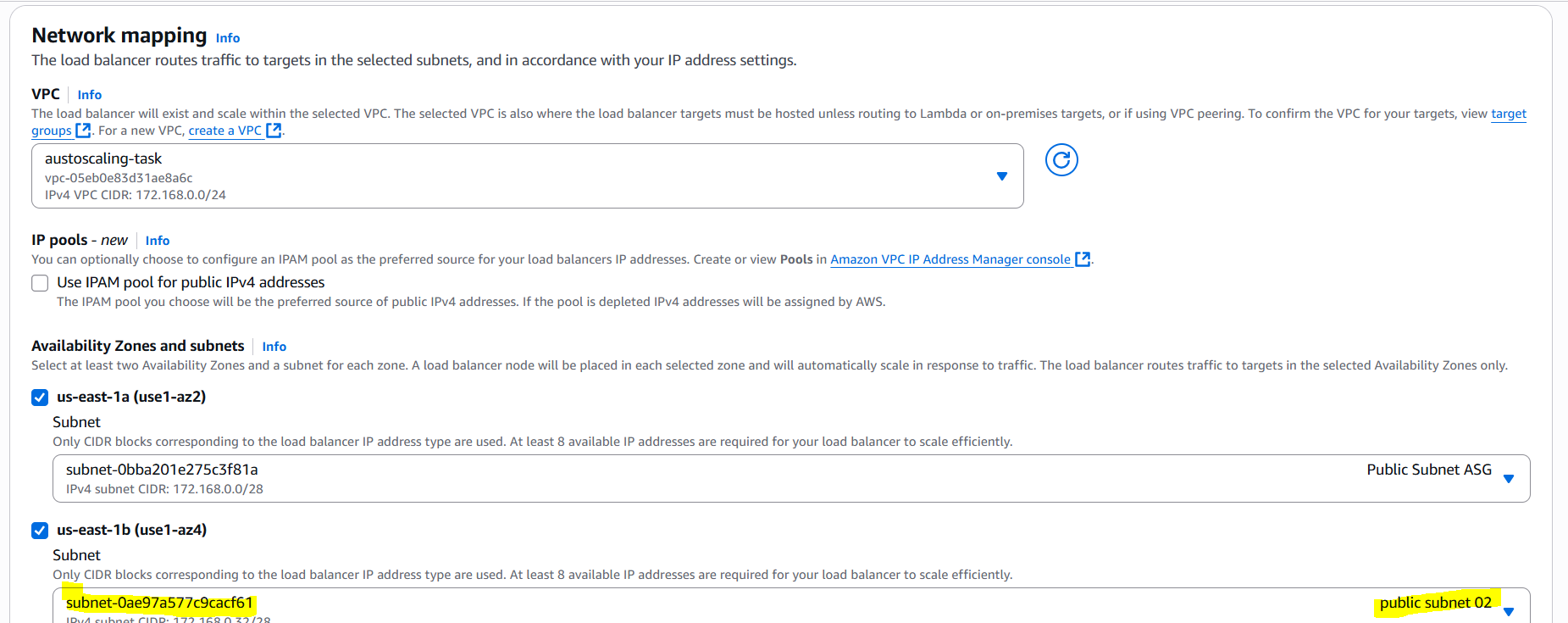
- Scheme: internet-facing

- IP address type: ipv4



5. Select the VPC: austoscaling-task

6. Select the public subnet.

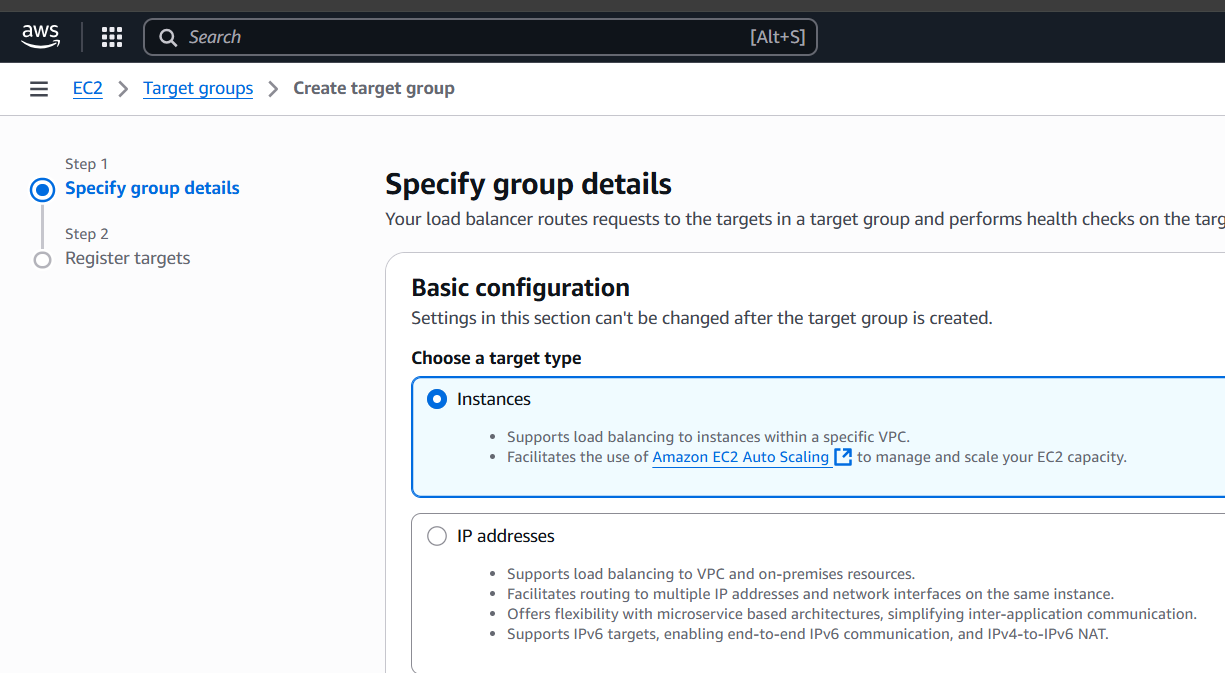


To create ALB we need minimum 2 public subnets so created one more subnet in another AZ.

**Create Target Group**

1. Create a target group:

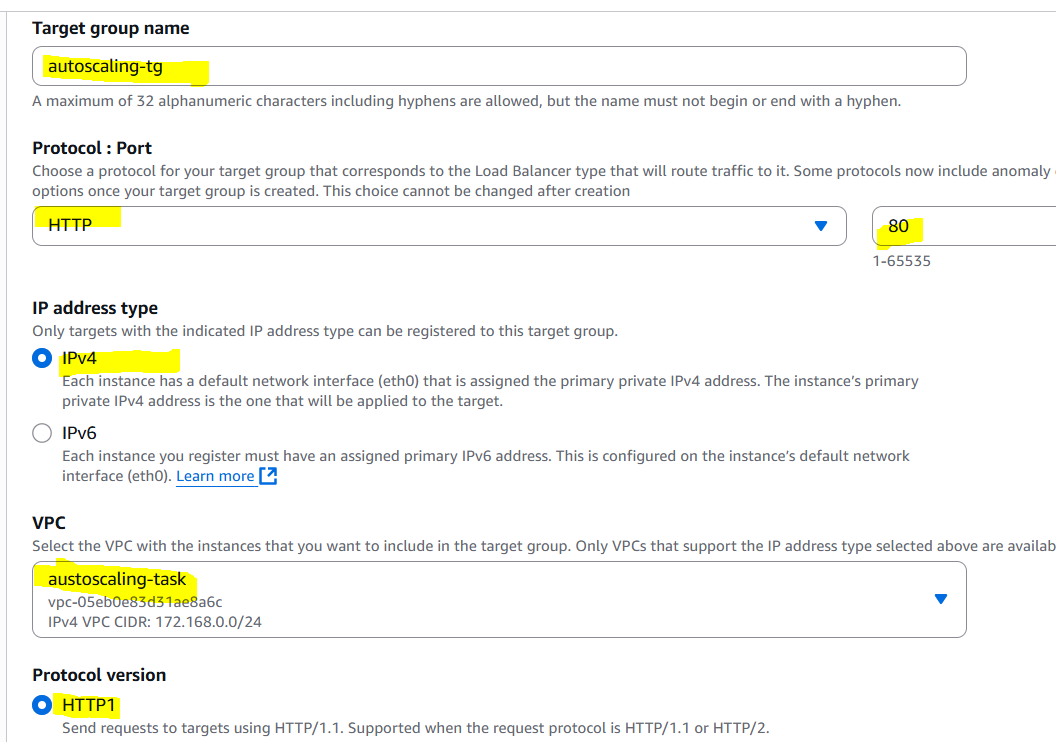
- Name: autoscaling-tg

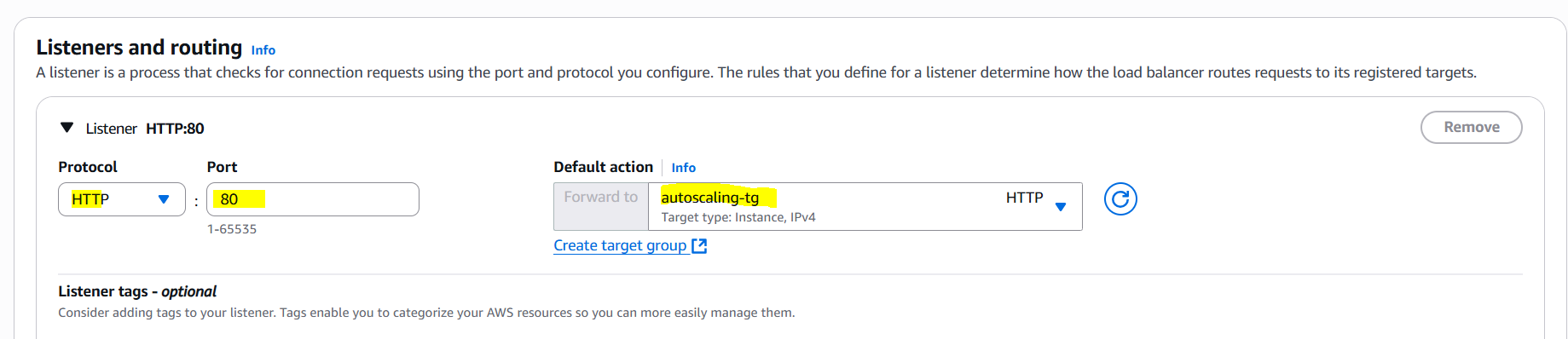


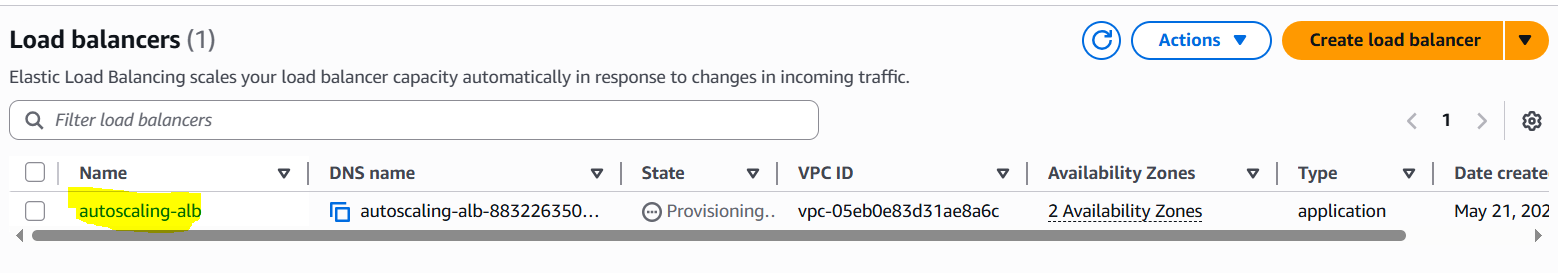
- Protocol: HTTP

- Port: 80

- Target type: Instance

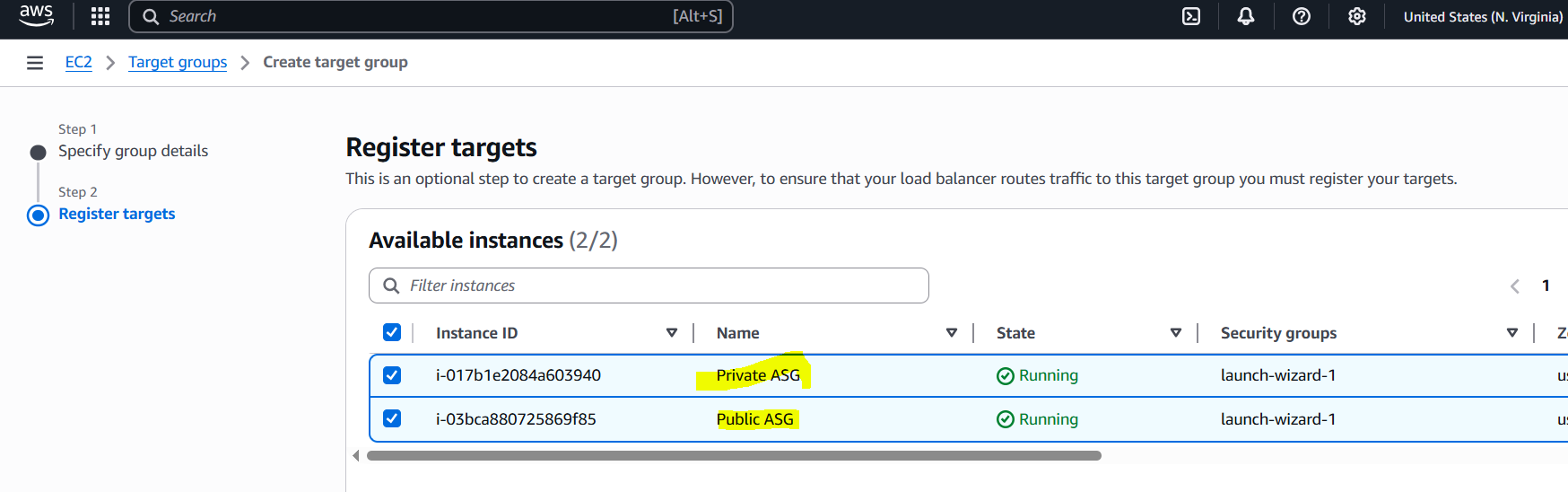


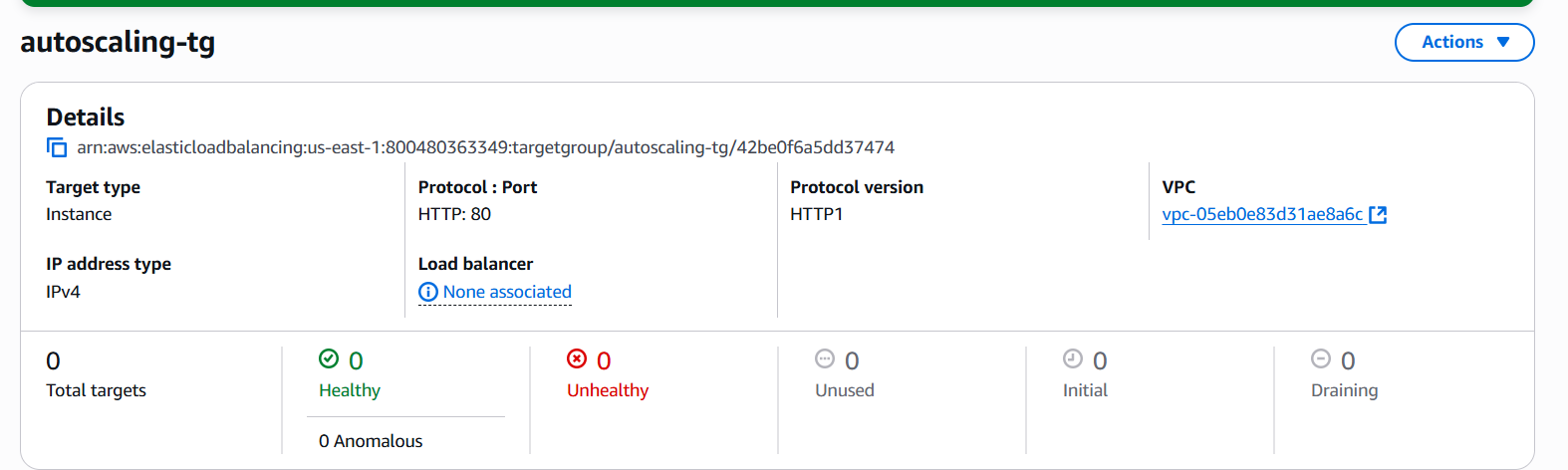




2. Register targets:

- Select both EC2 instances (public and private)





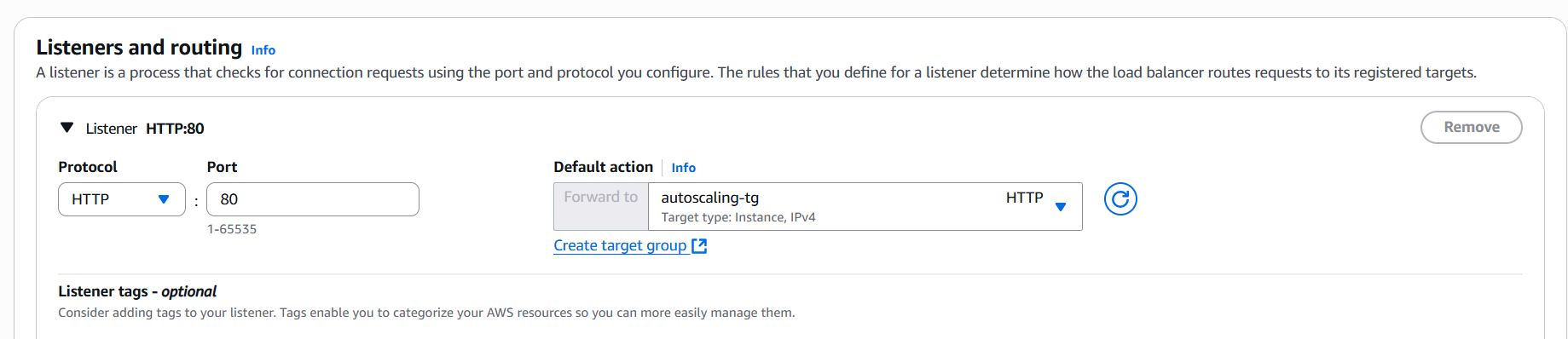
**Configure Listener**

1. Create a listener:

- Protocol: HTTP

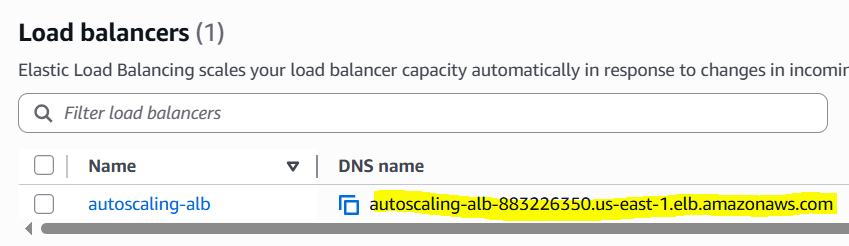
- Port: 80

- Default action: Forward to target group (austscaling-tg)



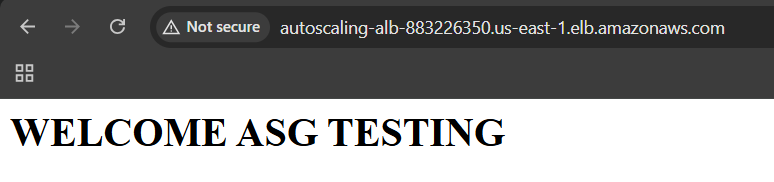
**Test**

1. Copy the ALB's DNS name.



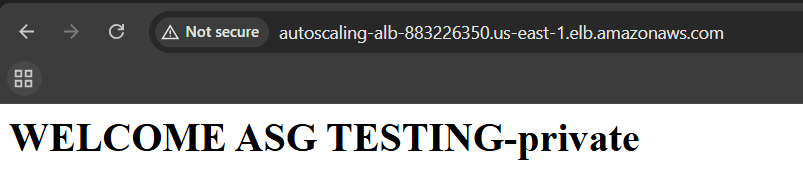
2. Update private instance's index.html to "WELCOME ASG TESTING - PRIVATE".

3. Open a browser and navigate to autoscaling-alb-883226350.us-east-1.elb.amazonaws.com>.



Always check SG of ALB also.

4. Refresh the page to see traffic routed to both instances.



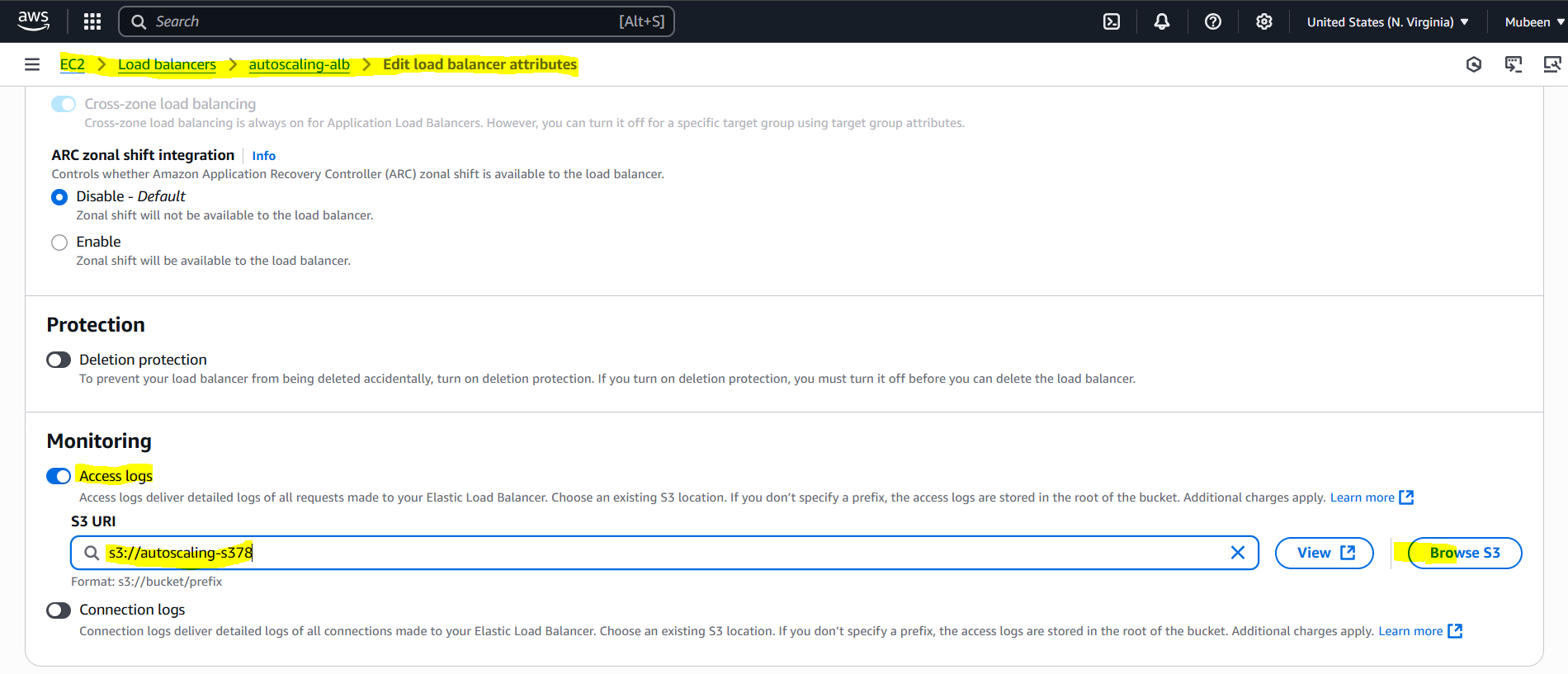
**9) Store Application load balancer logs to s3.**  
1. Go to the ALB settings.

2. Navigate to "Attributes".

3. Edit "Access logs".

4. Enable access logging.

5. Specify an S3 bucket.



**Requirements:**

- S3 bucket policy allowing ALB to write logs.

{

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

"Statement": [

{

"Sid": "AllowALBLogs",

"Effect": "Allow",

"Principal": {

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

},

"Action": "s3:PutObject",

"Resource": "arn:aws:s3:::autoscaling-s378/AWSLogs/800480363349/\*",

"Condition": {

"StringEquals": {

"s3:x-amz-acl": "bucket-owner-full-control"

}

}

},

{

"Sid": "AllowALBLogsGetACL",

"Effect": "Allow",

"Principal": {

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

},

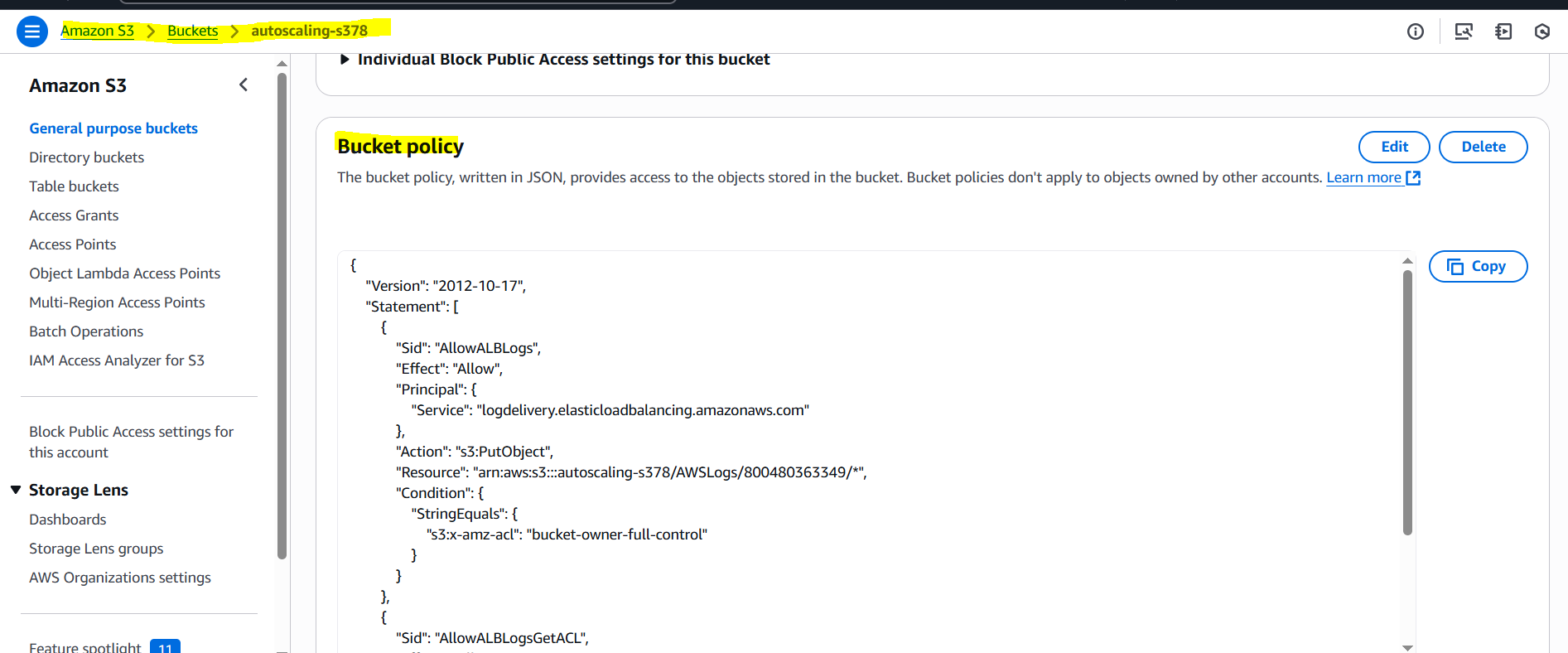
"Action": "s3:GetBucketAcl",

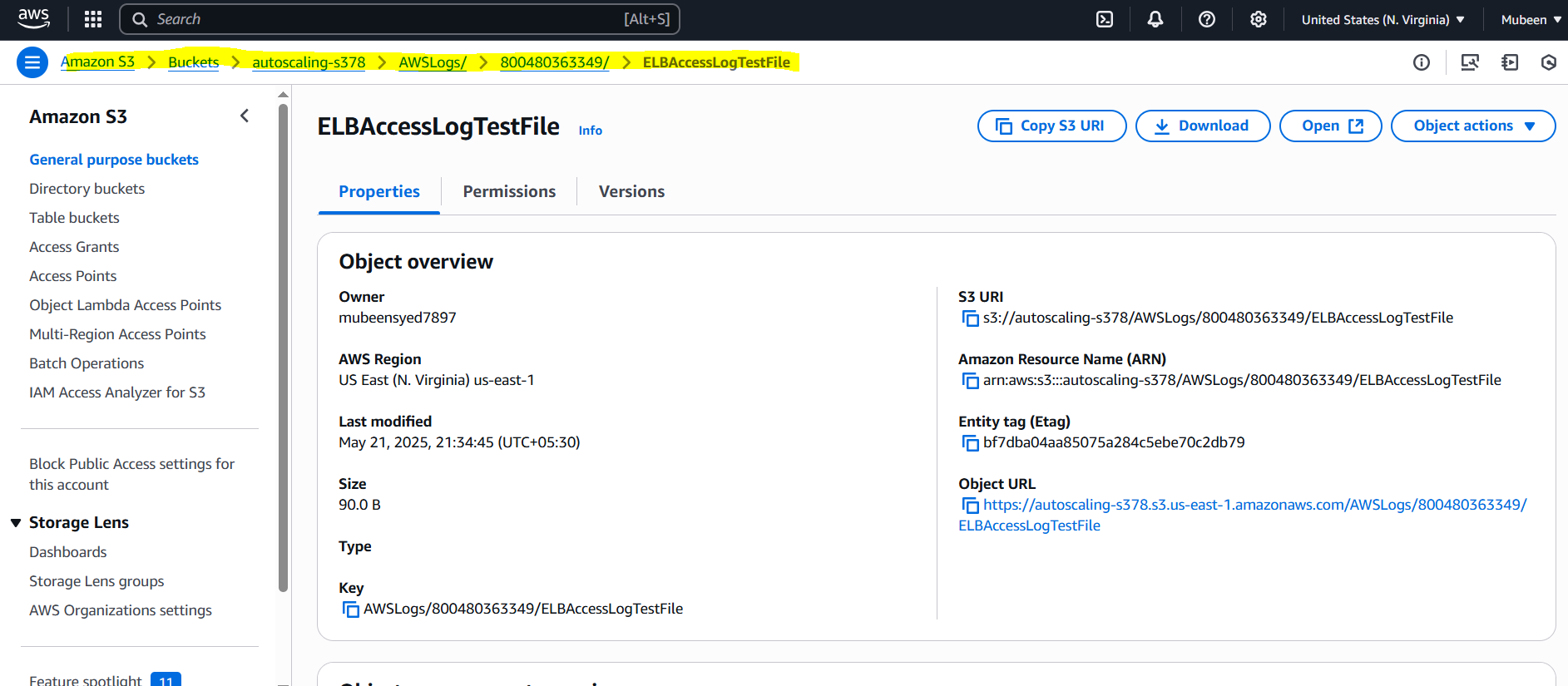
"Resource": "arn:aws:s3:::autoscaling-s378"

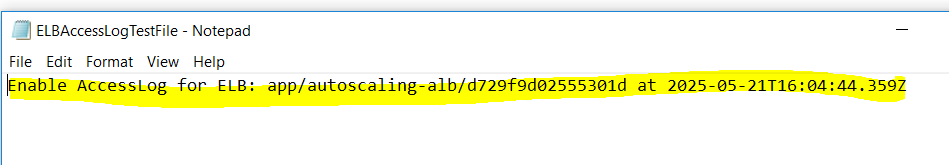
}

]

}







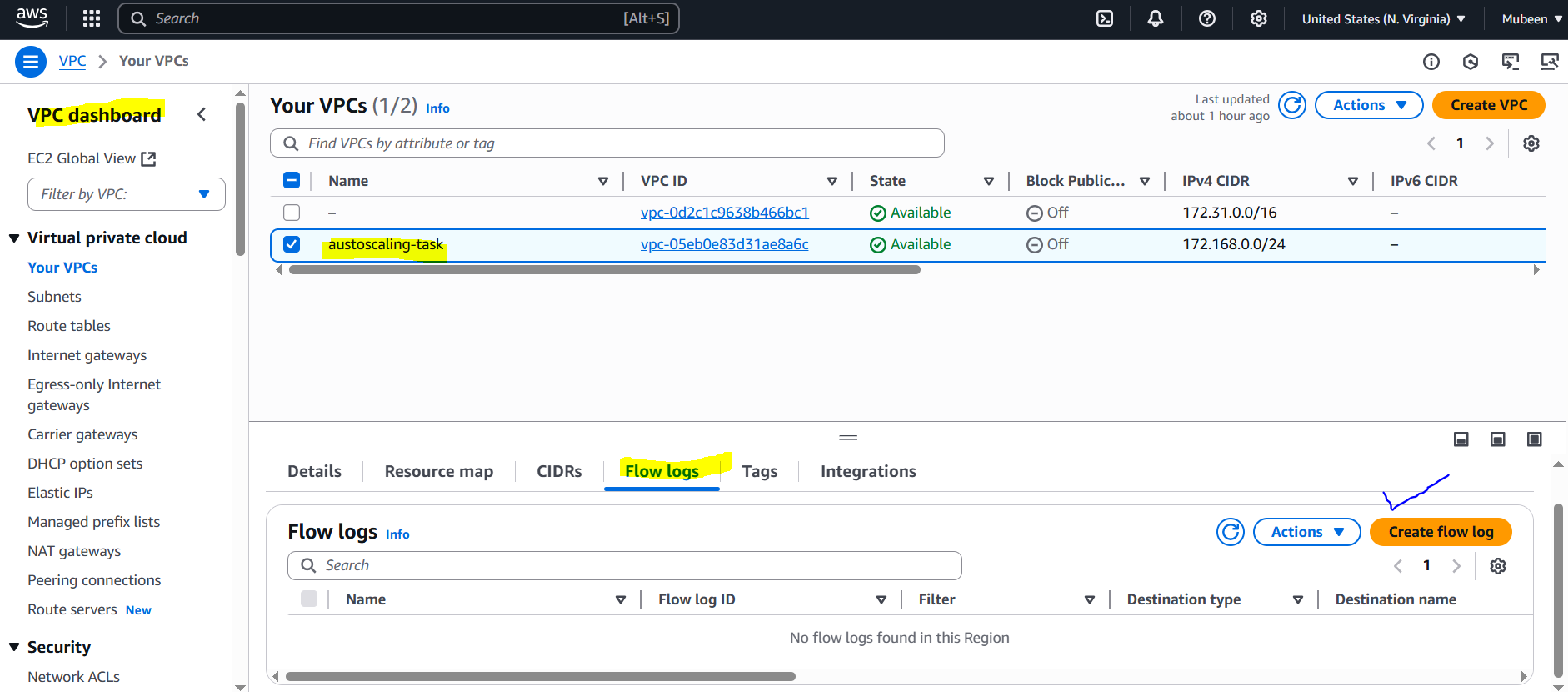
**10) Store the vpc flow logs to cloudwatch group.**

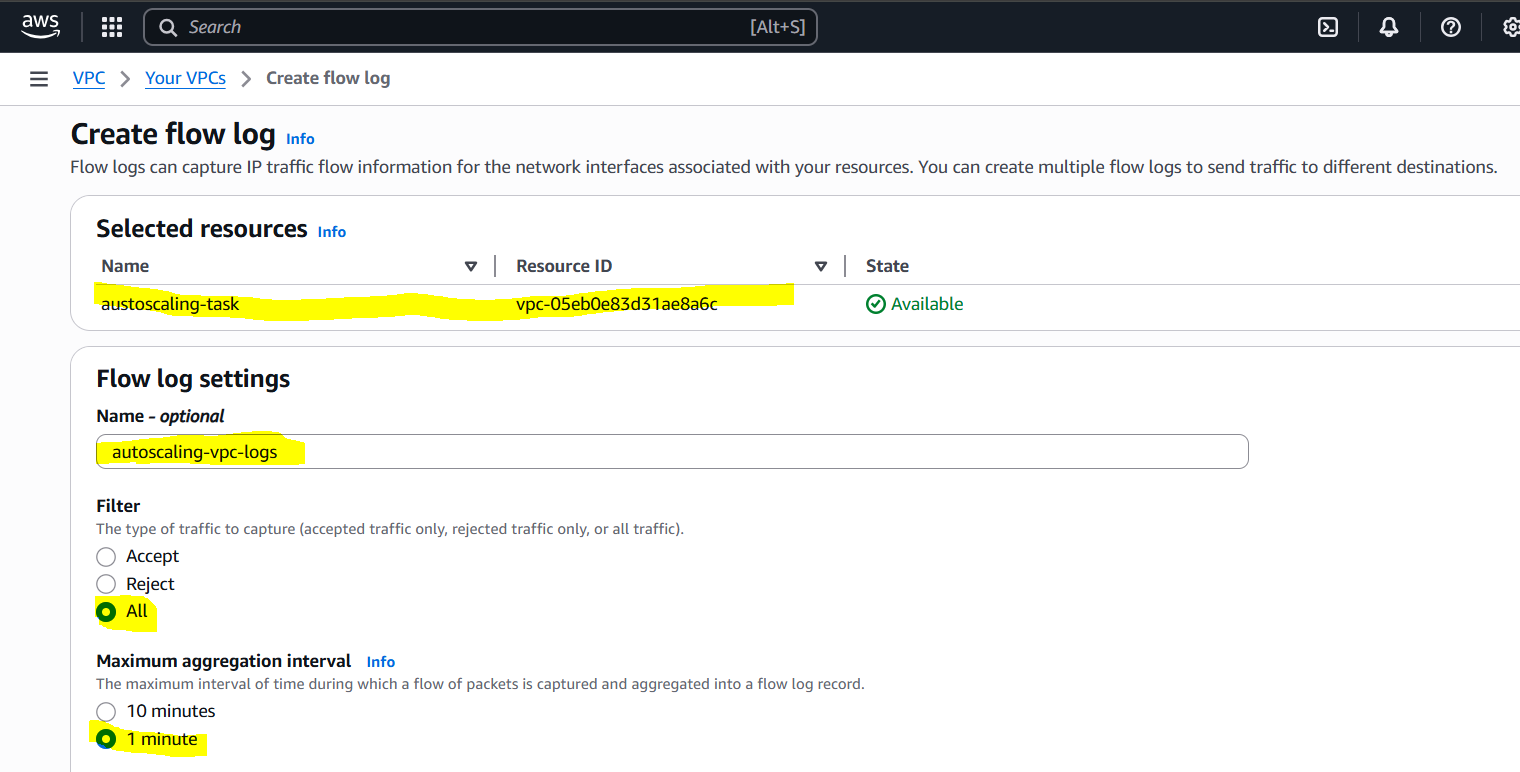
1. Go to the VPC dashboard.

2. Select the VPC.

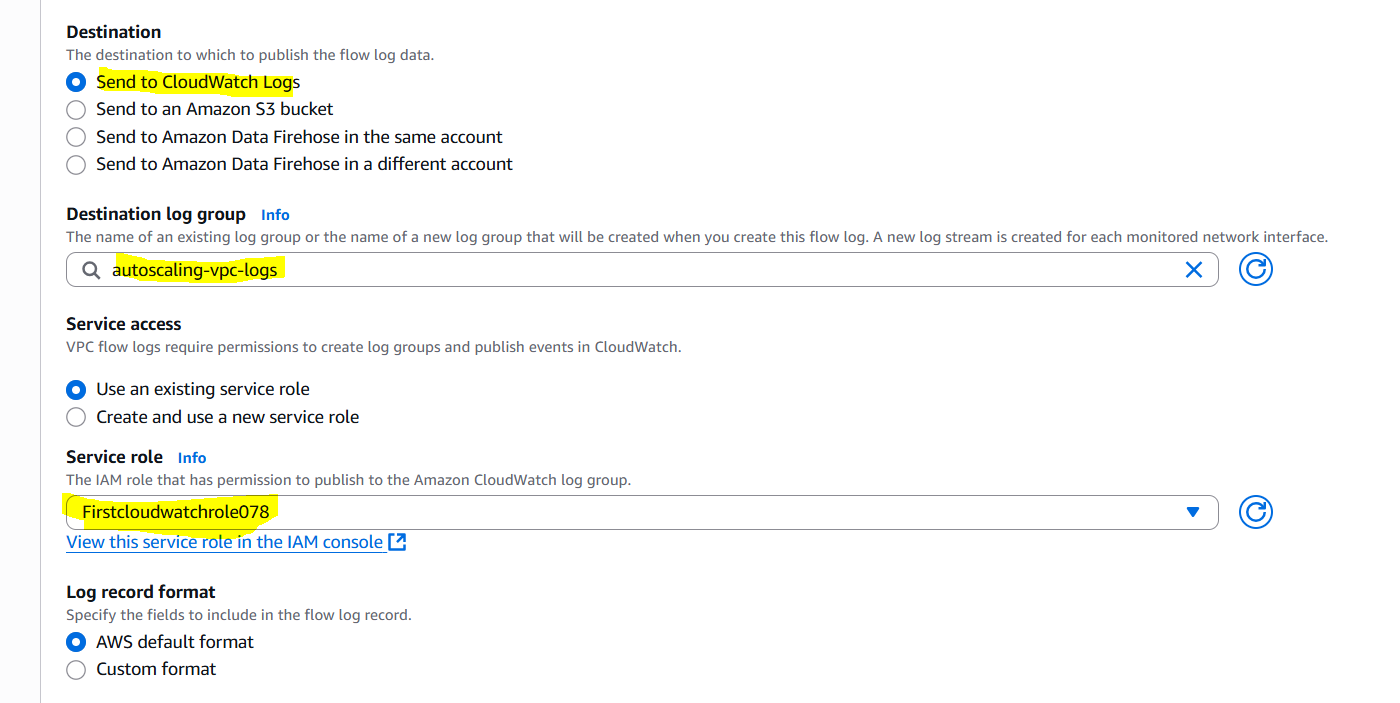
3. Click on "Flow logs" in the sidebar.

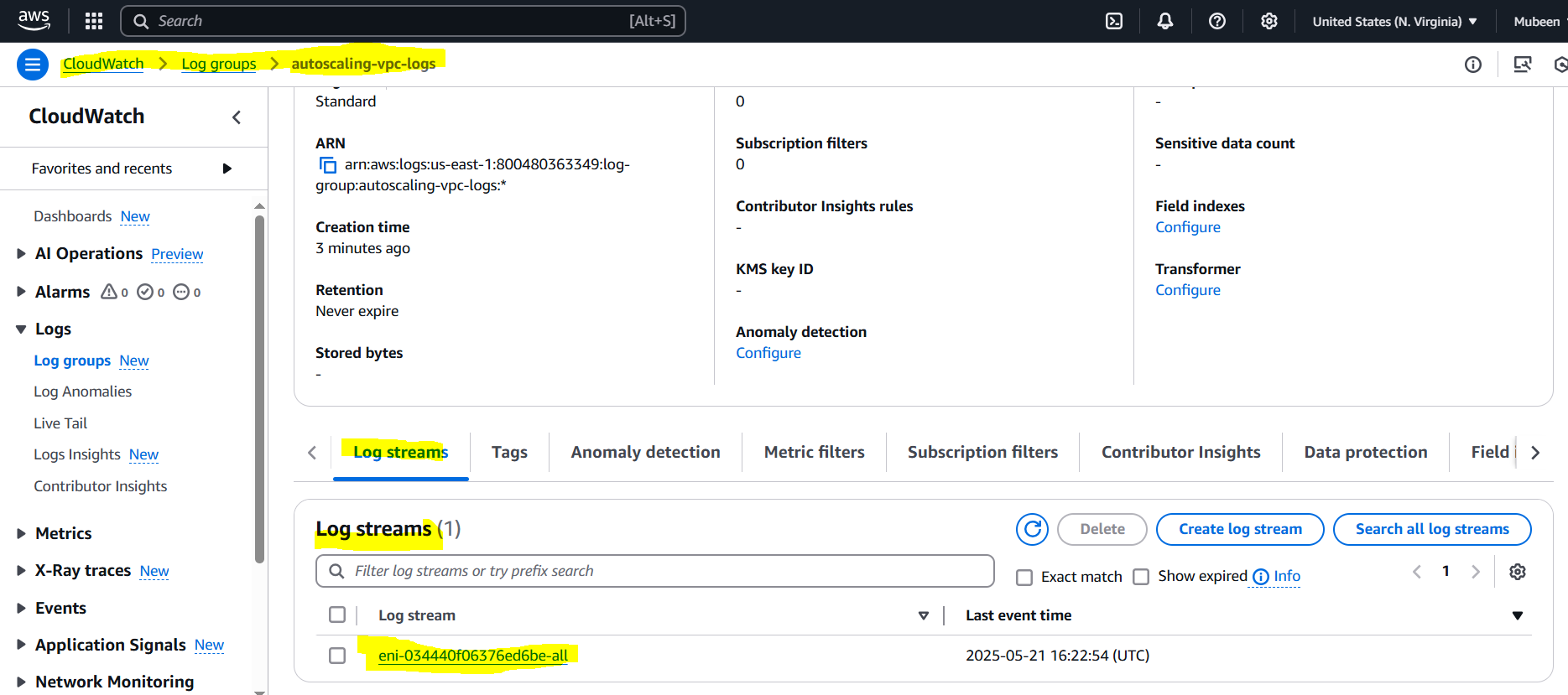
4. Click "Create flow log".

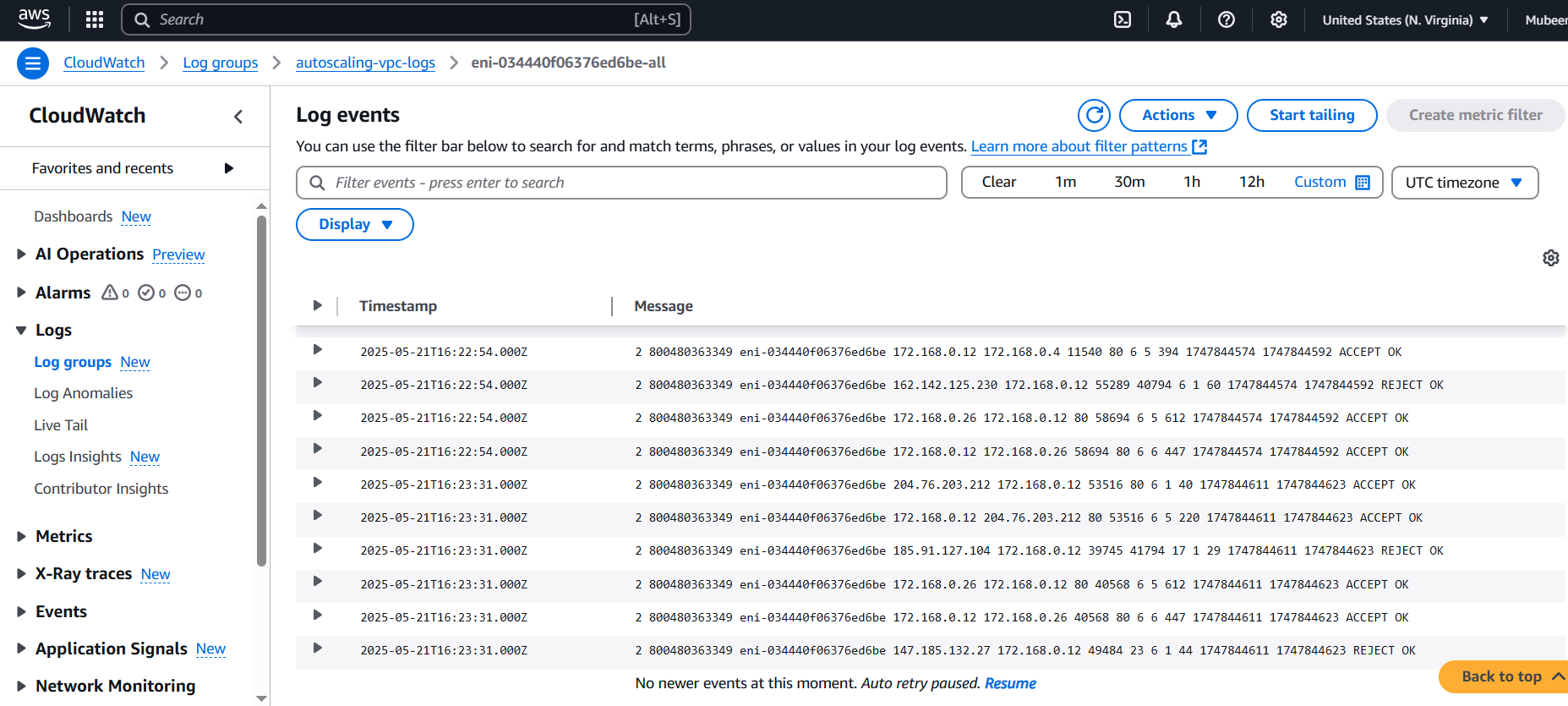




5. Choose "CloudWatch Logs" as the destination.

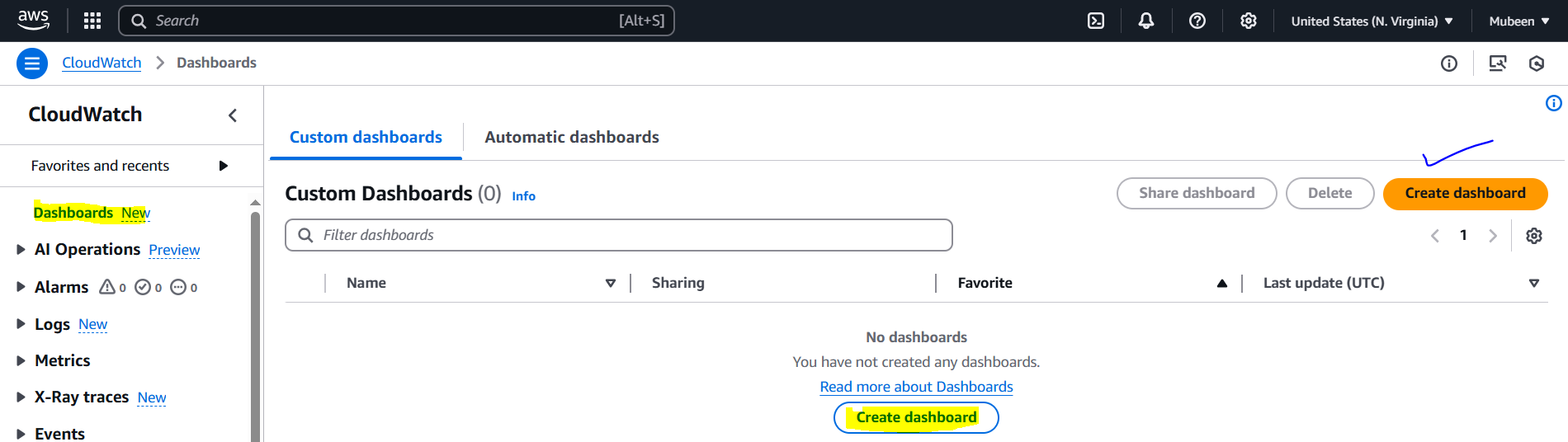


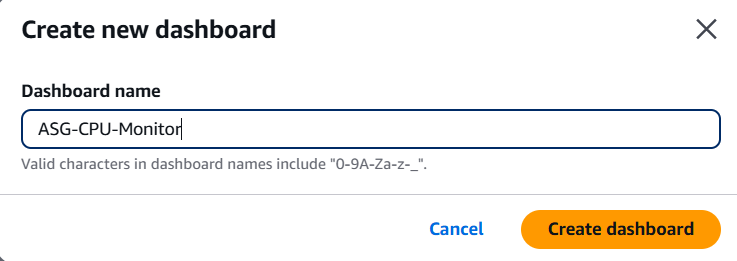
**To view the logs:**



### 11) Create Monitoring Dashboards to monitor cpu utilization and to monitor apache service. Create CPU Utilization Dashboard

Go to **CloudWatch > Dashboards** in the AWS Console.

Click **“Create dashboard”**.  


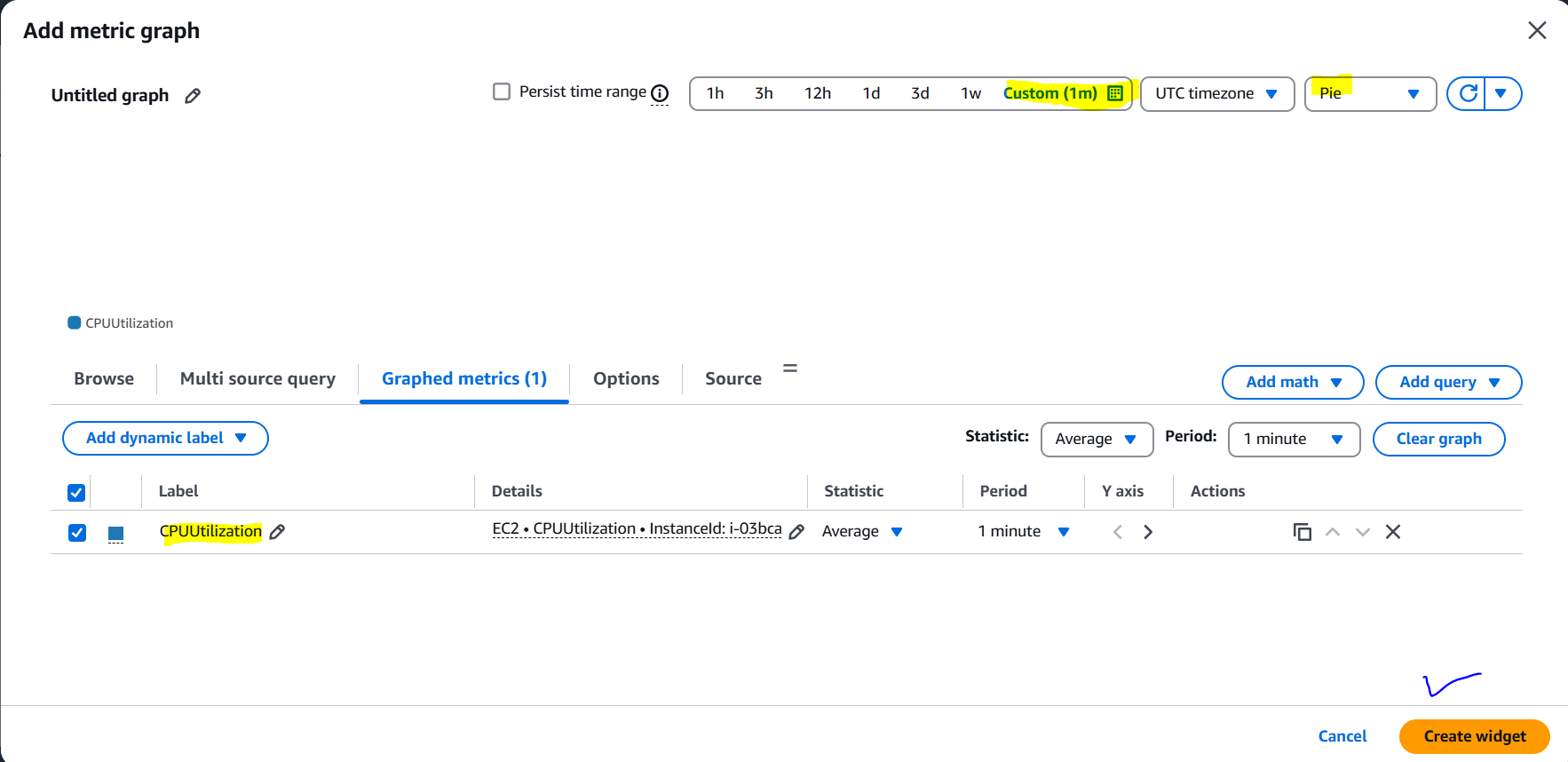
Enter a name (ASG-CPU-Monitor) and click **Create**.  


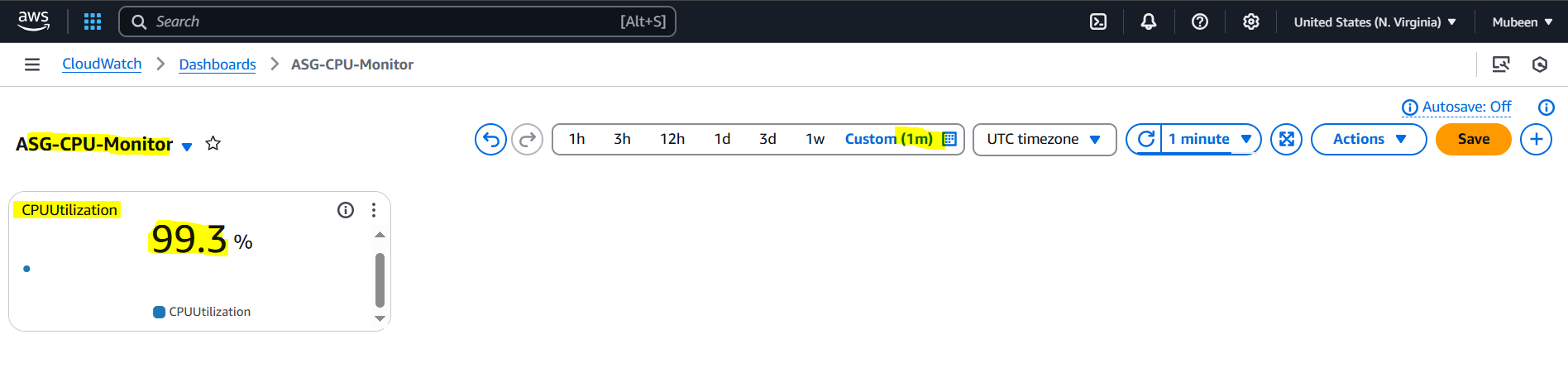
Click **“Add widget”**, select **“Line”** chart, then click **Next**.

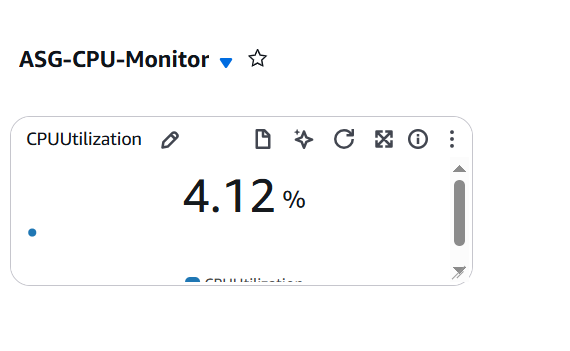
Choose **“EC2”** as the namespace.

Select **“CPUUtilization”** metric.

Choose the EC2 instance you want to monitor.

Click **Create widget**, then **Save dashboard**.  






Added load and removed load.

**Create Apache Dashboard in CloudWatch**

Running a bash script to create metric **Apache**. Make sure to run script after aws configure and add ccronjob.

#!/bin/bash

instance\_id="i-03bca880725869f85"

status=$(curl -s --max-time 2 http://localhost/server-status?auto | grep 'BusyWorkers')

if [[ -n "$status" ]]; then

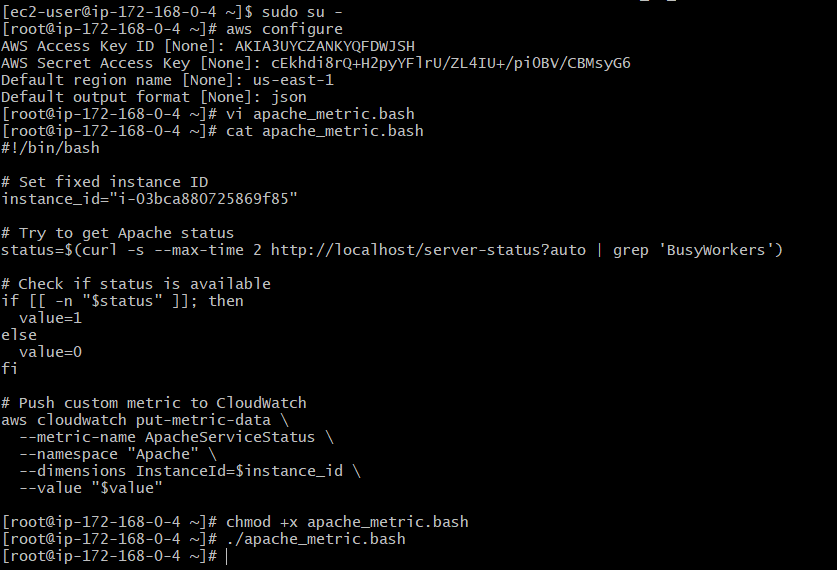
value=1

else

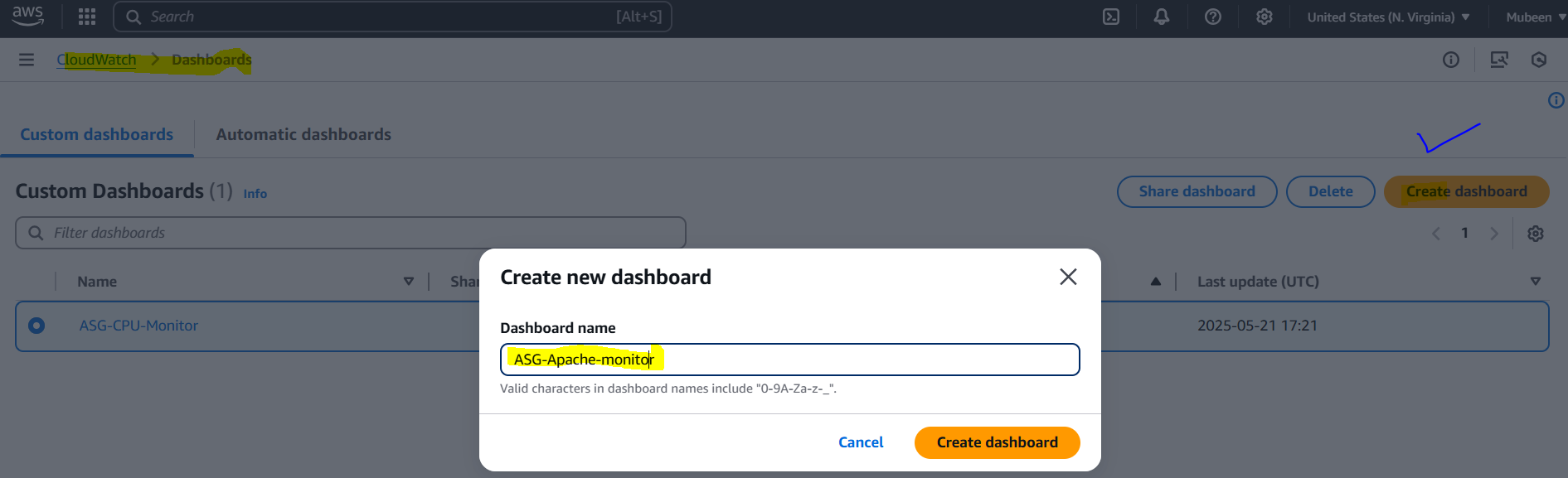
value=0

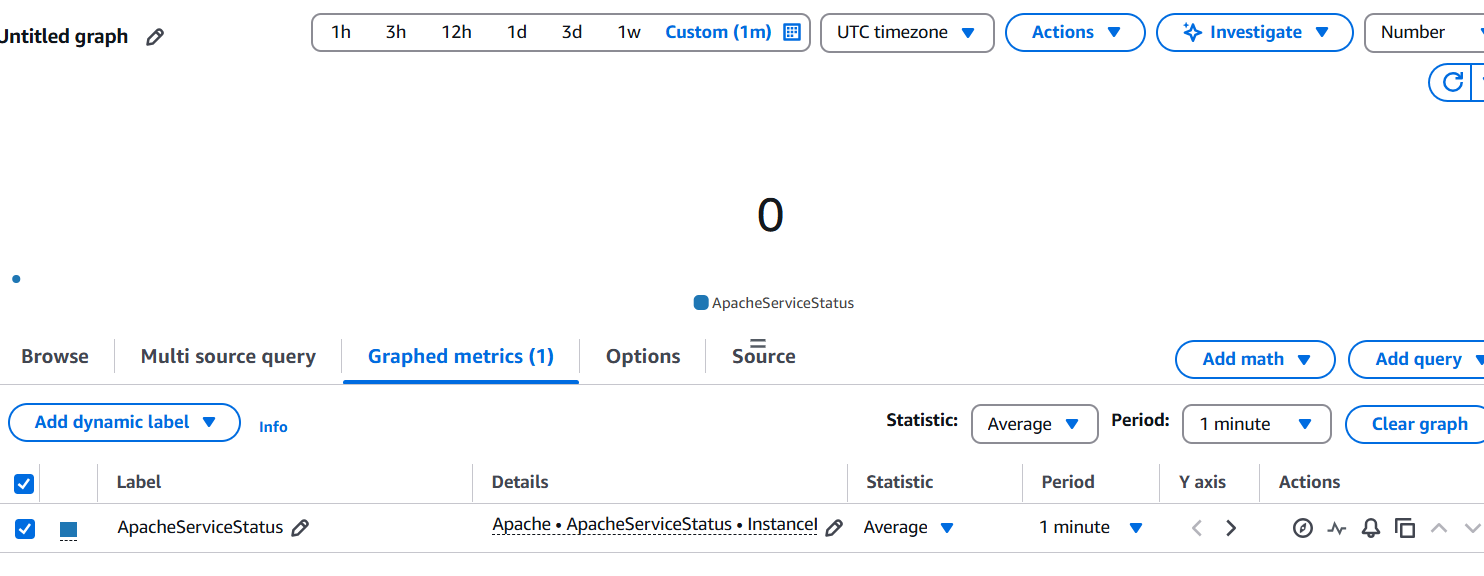
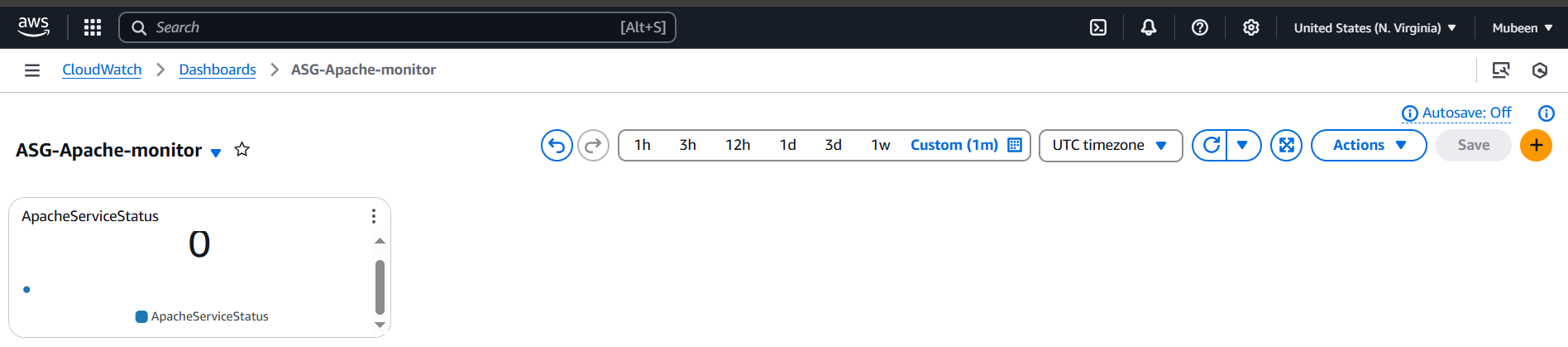
fi

aws cloudwatch put-metric-data --metric-name ApacheServiceStatus --namespace "Apache" --dimensions InstanceId=$instance\_id --value "$value"

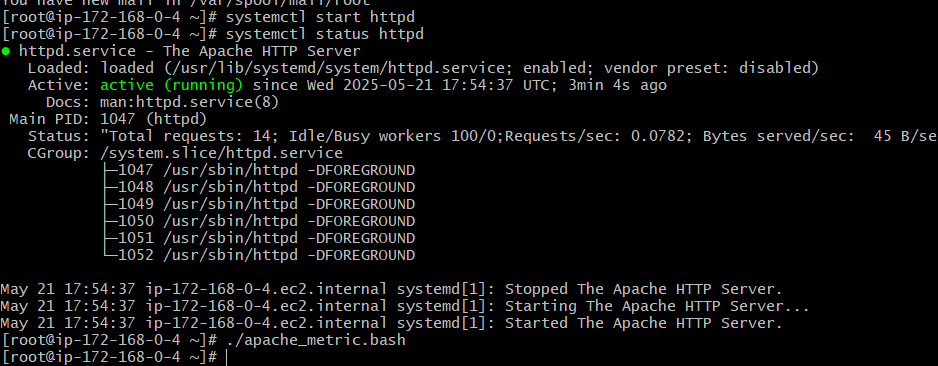


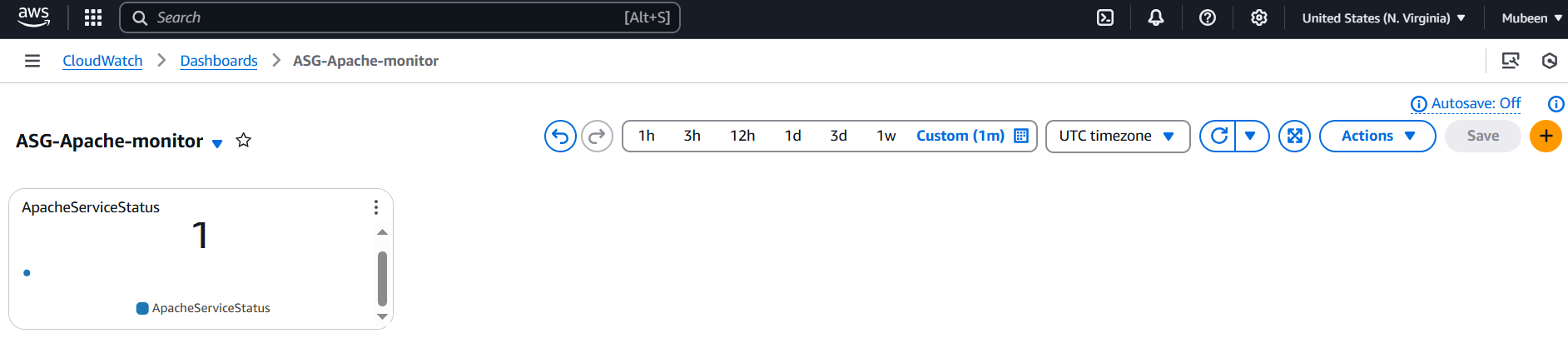


Go to **CloudWatch > Dashboards**  
Click **“Create dashboard”**, name it ASG-Apache-monitor  


After starting the service.



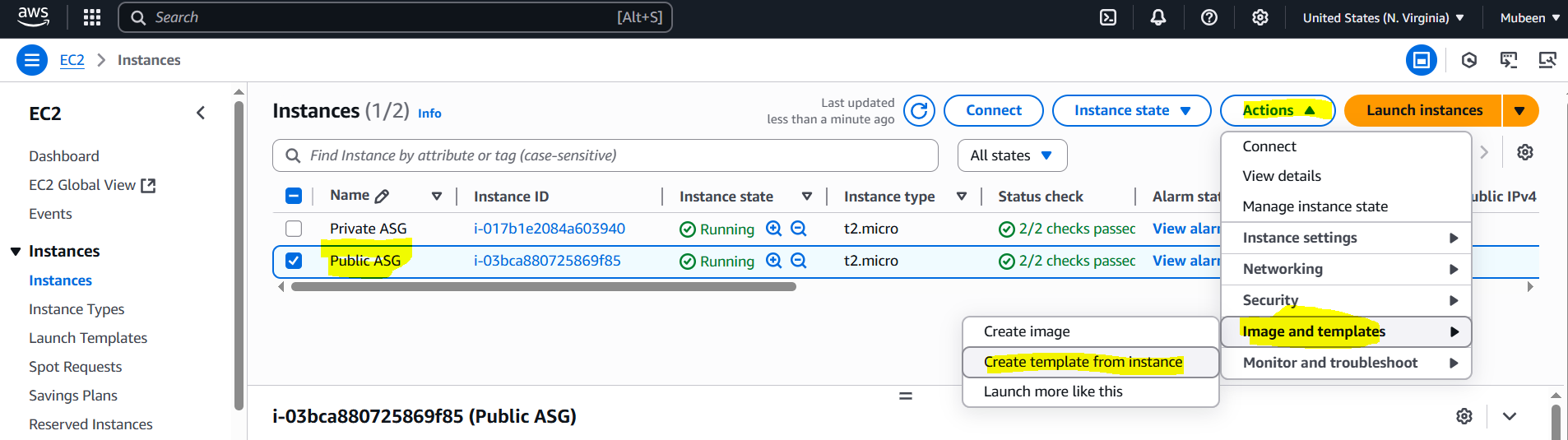


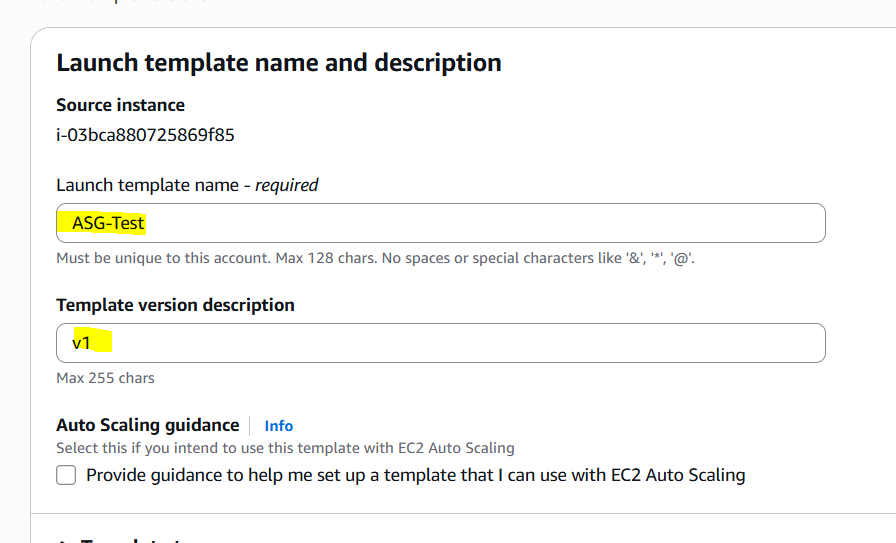
**12) CPU utilization is more than 70% then it should trigger Autoscaling and launch new instance.**

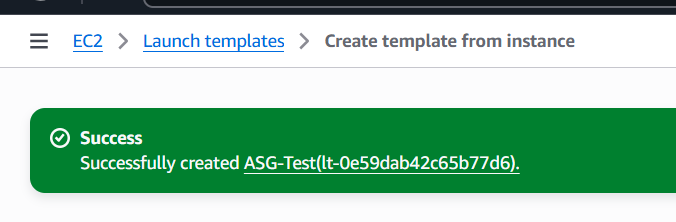
**Step 1: Create Launch Template**

1. EC2 > Instances > Select instance > Actions > Image and templates > Create template from instance

2. Configure template details (e.g., template name, version)





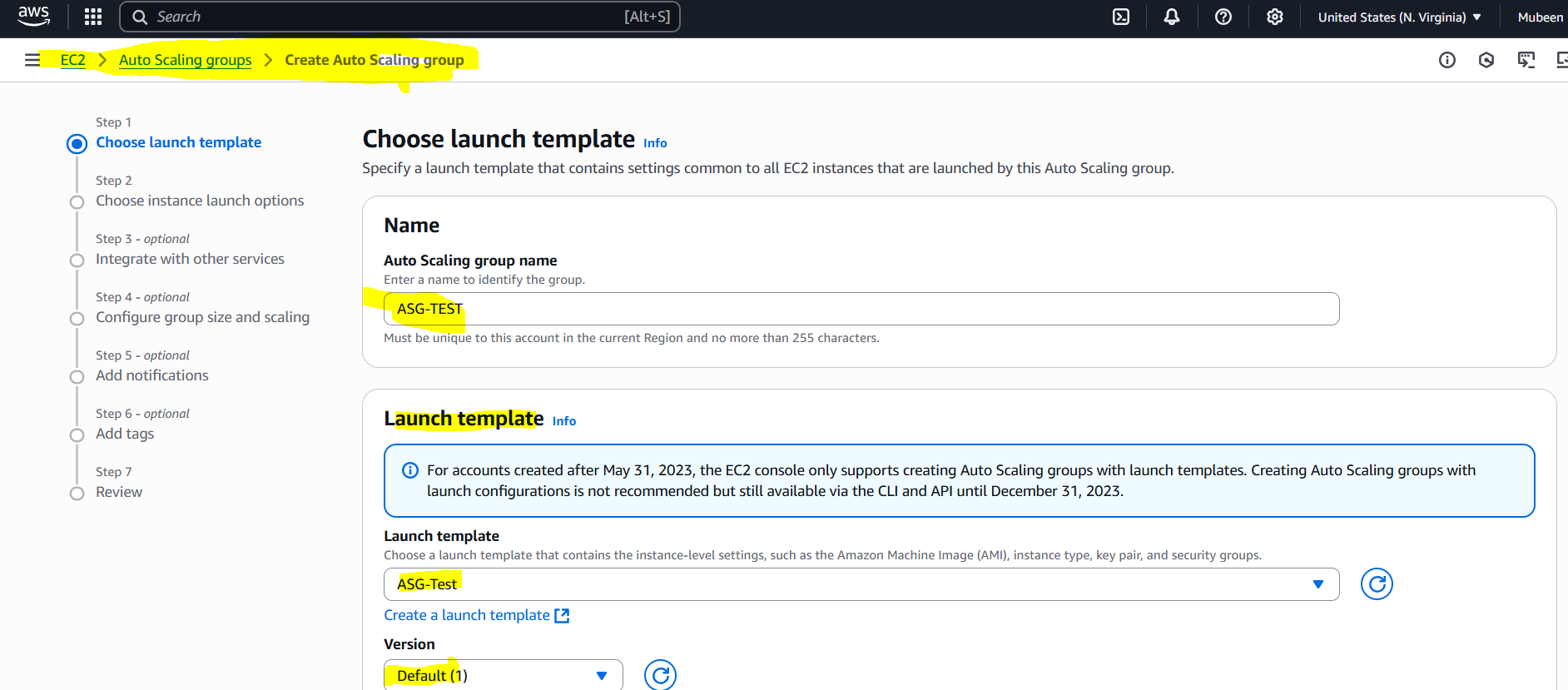


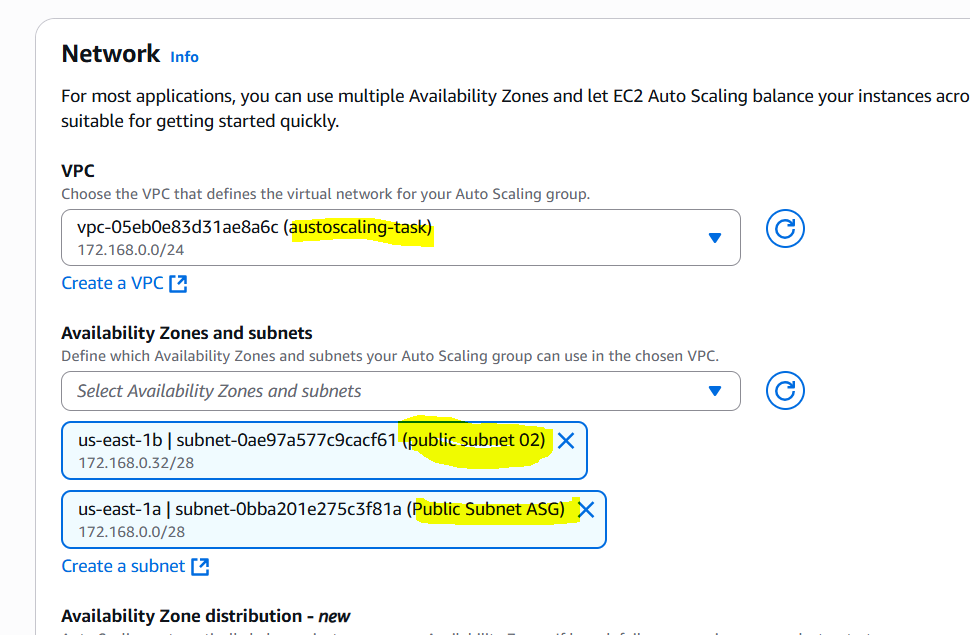
**Step 2: Create Auto Scaling Group**

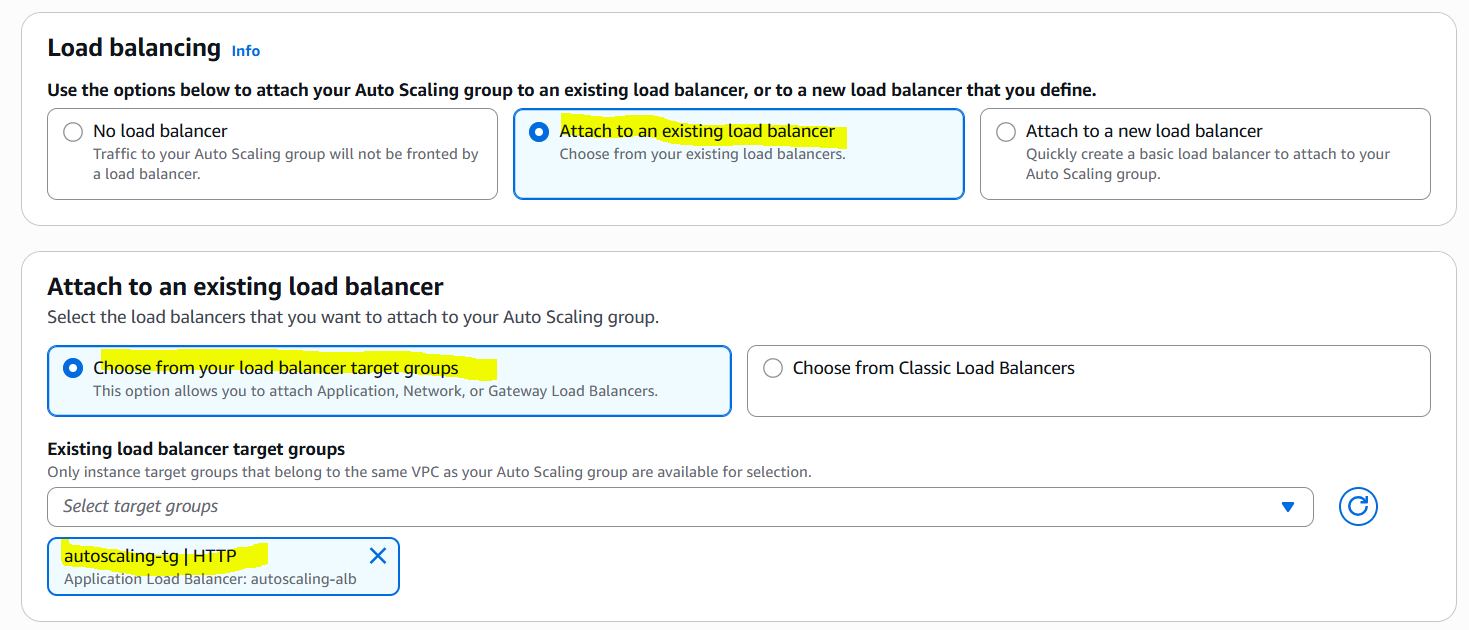
1. EC2 > Auto Scaling Groups > Create Auto Scaling group

2. Choose launch template created in Step 1

3. Configure group details (e.g., group name, VPC, subnets)







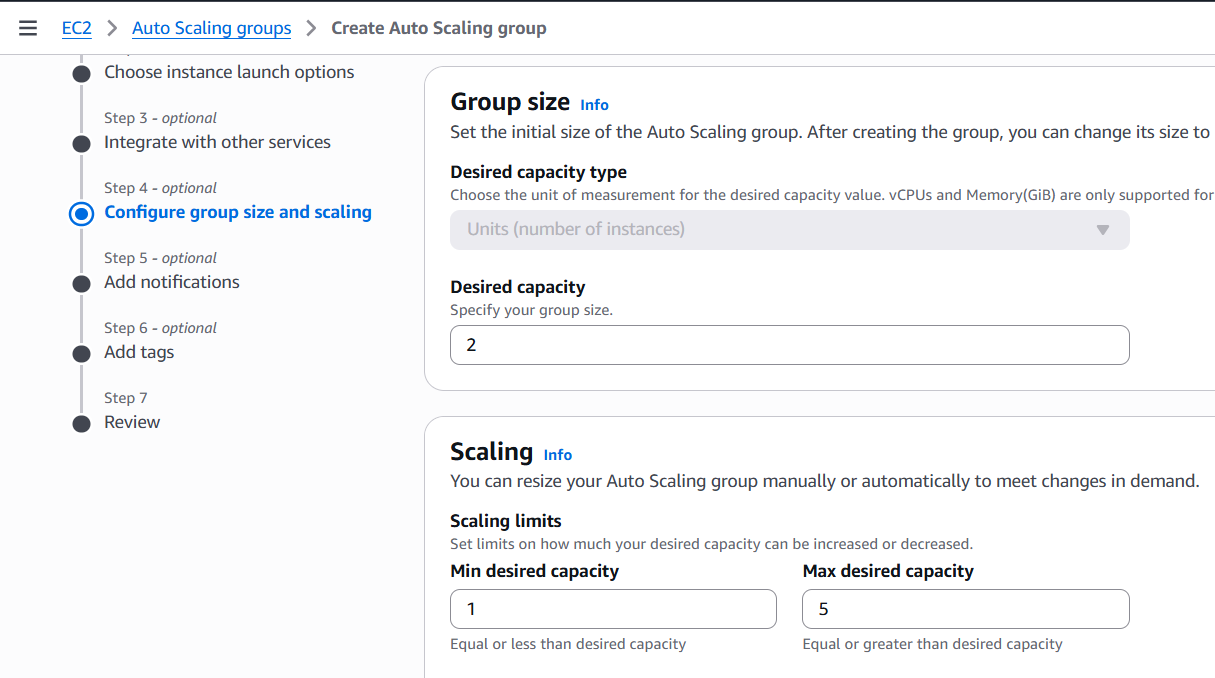
**Step 3: Configure Scaling Policy**

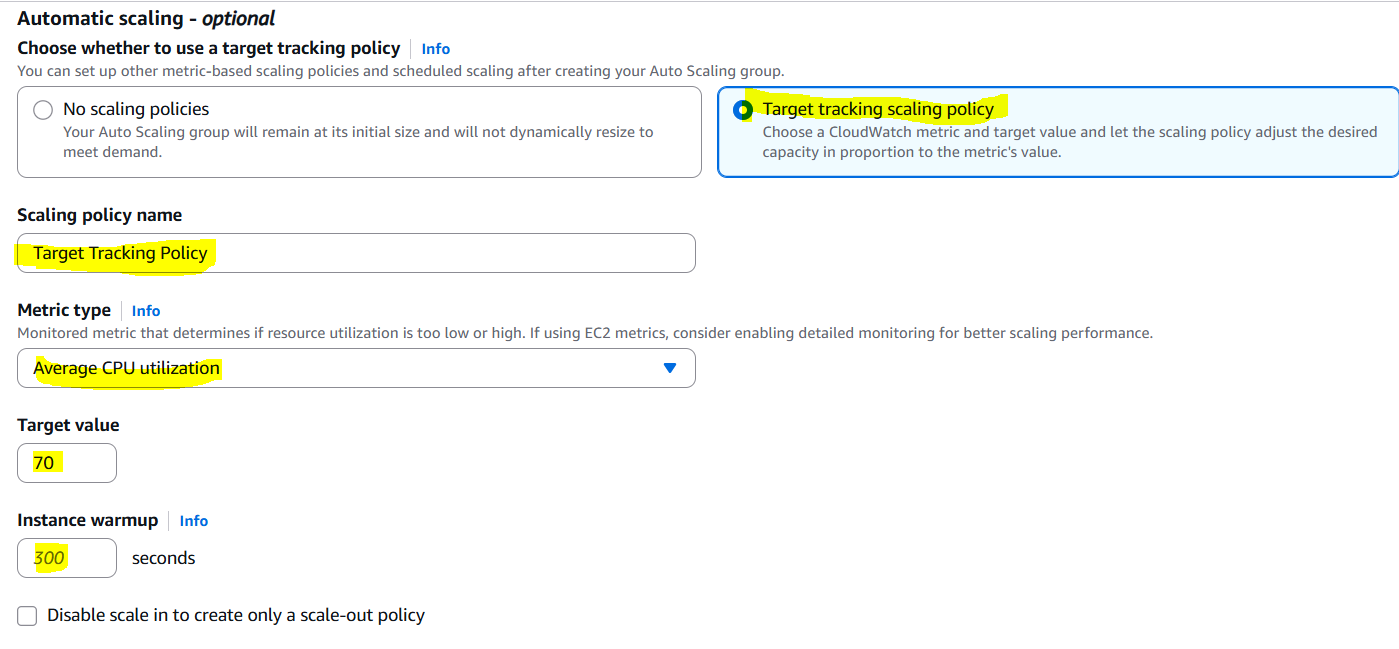
1. In Auto Scaling group, go to Automatic scaling

2. Create target tracking scaling policy

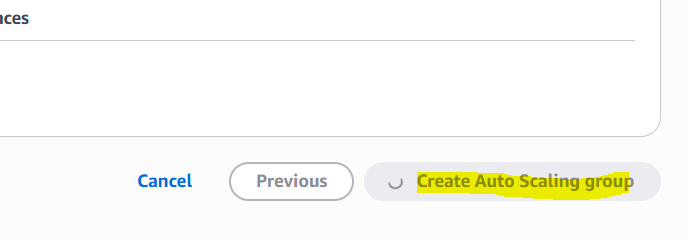
3. Choose CPU utilization metric

4. Set target value to 70%



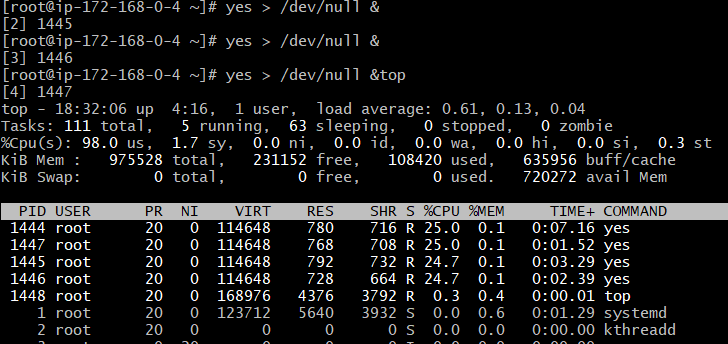


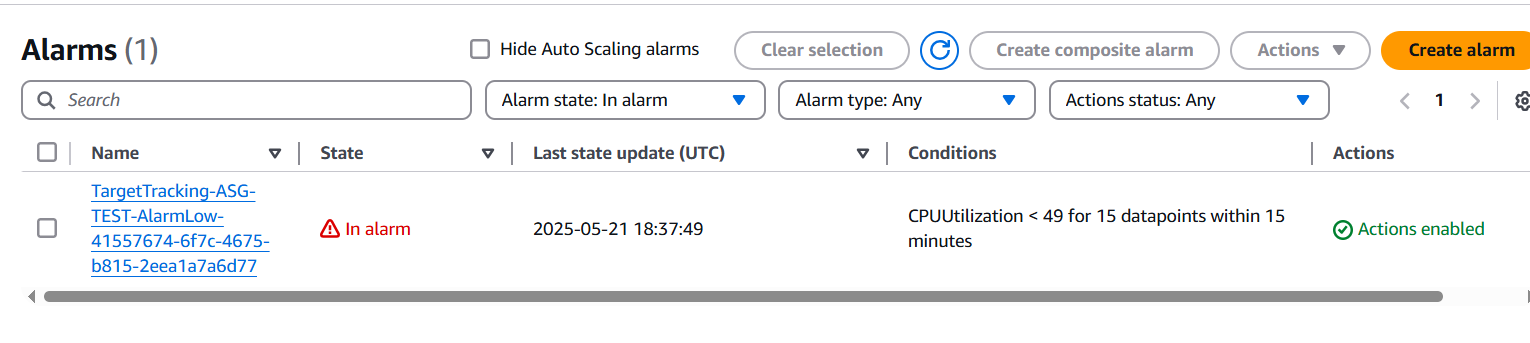
**Step 4: Review and click on Create Auto Scaling group**

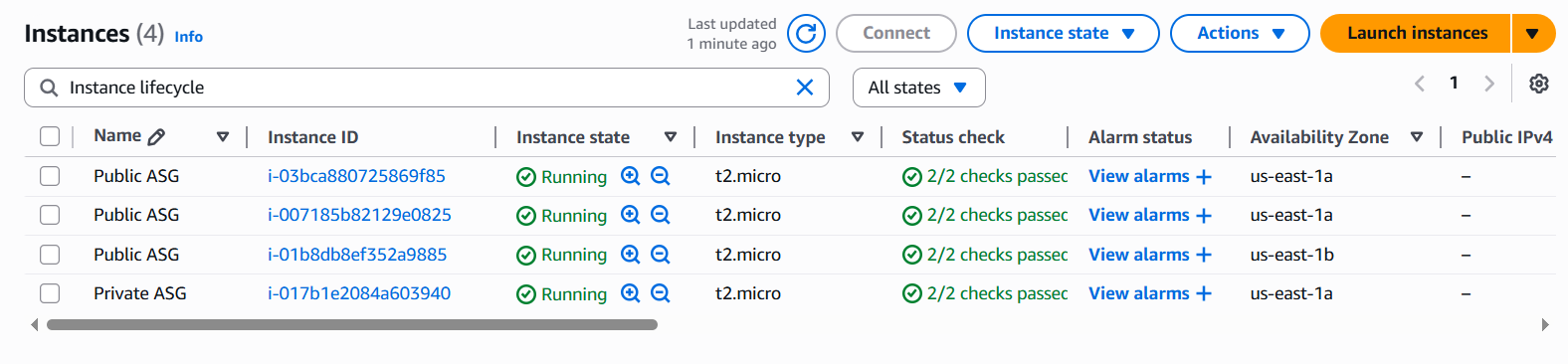


**Testing:**

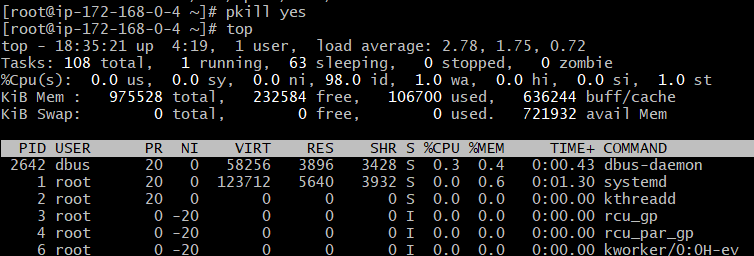
Adding load.

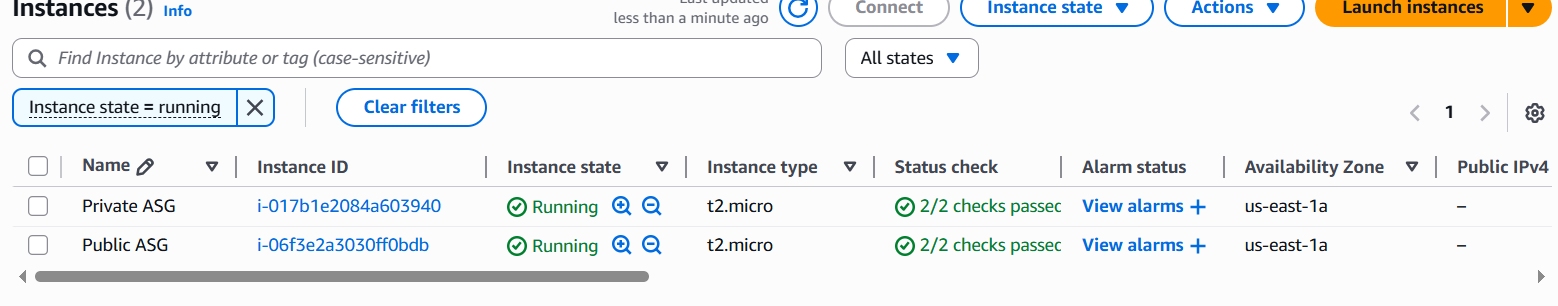






Removing load





**The-End**