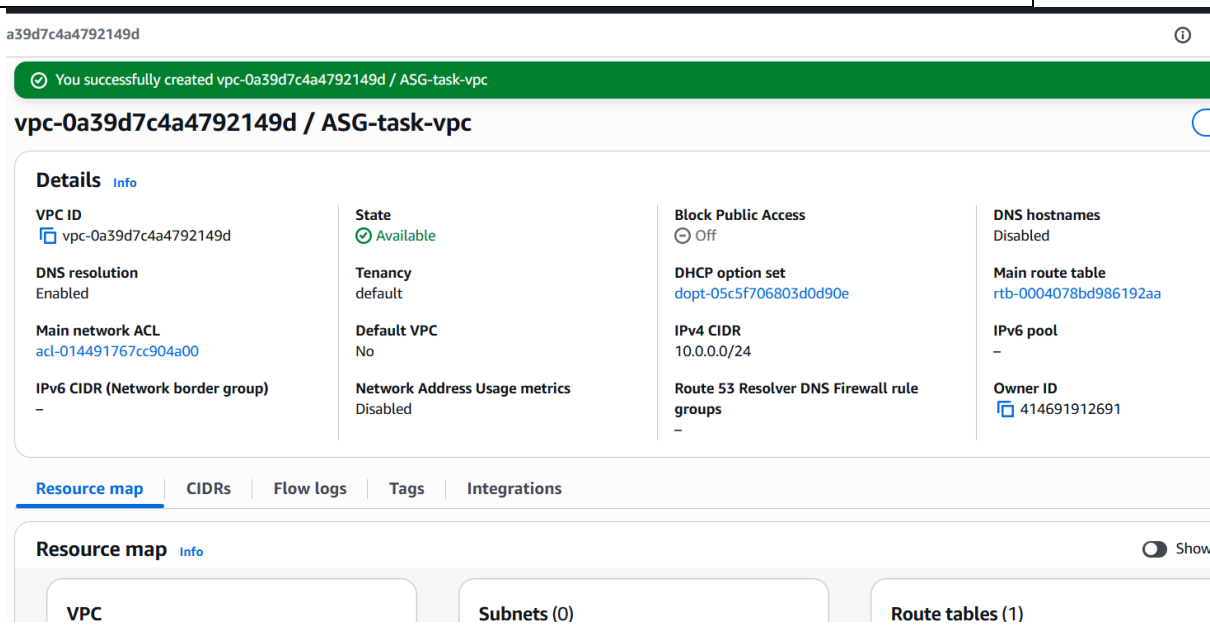
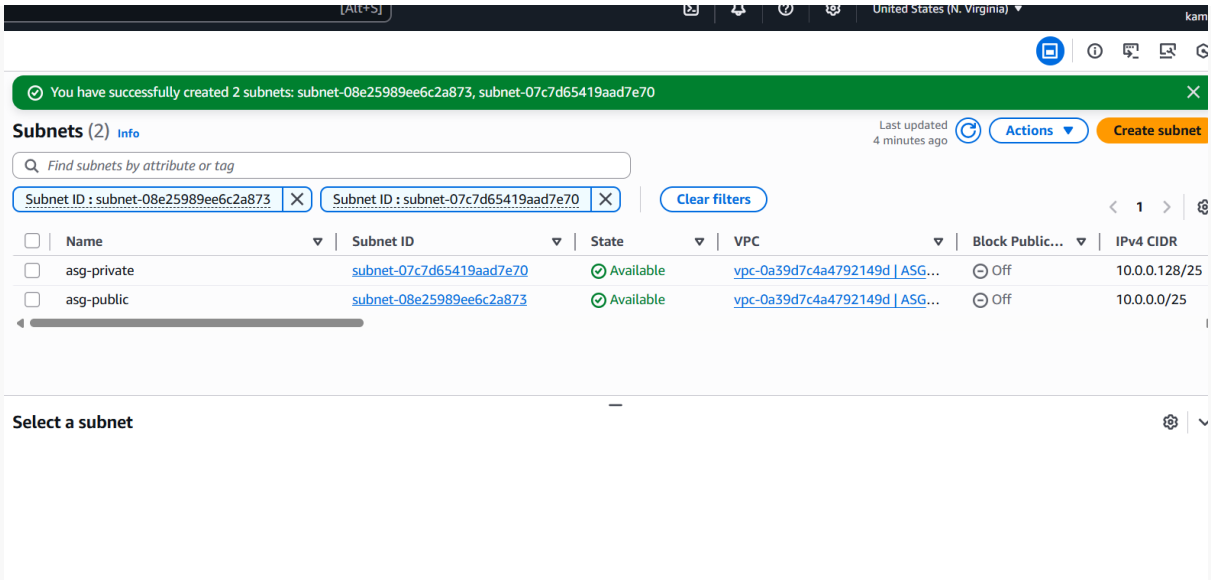
ASG

1. Create one VPC in N. Virginia region.
    - Open aws console
    - Navigate to vpc
    - 

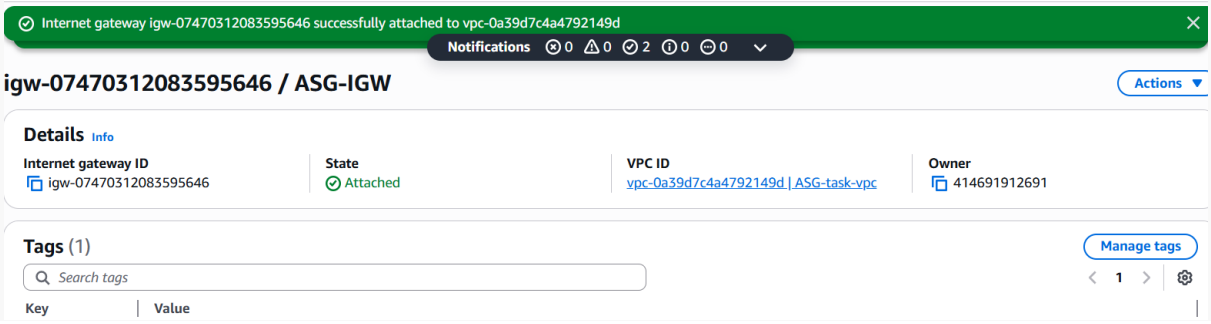
1. 2 . Create two subnets: one public subnet and one private subnet.

    - In vpc
    - One public subnet
    - One private subnet

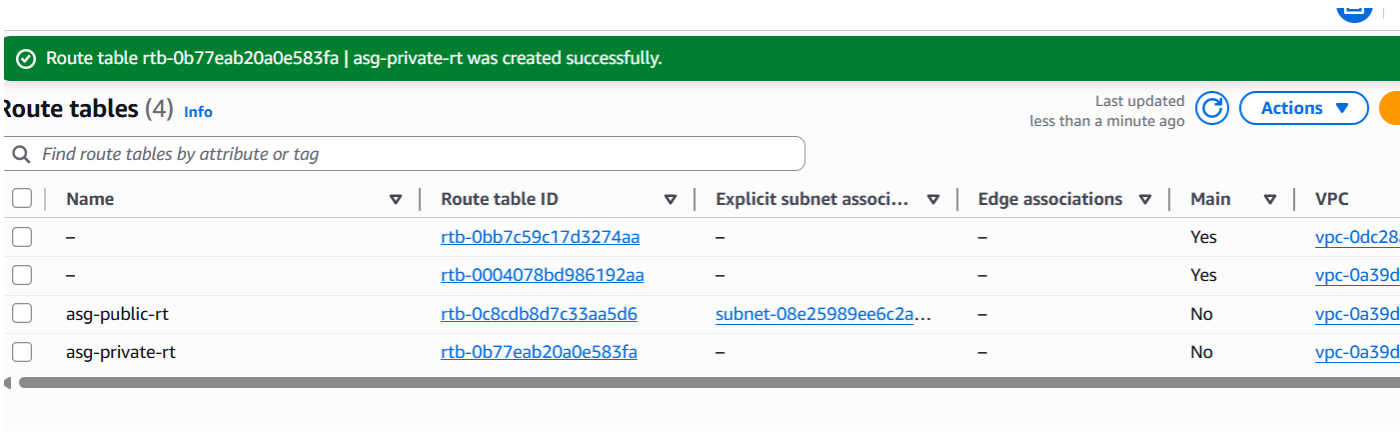


3 . Attach an IGW to the VPC.

- Create a internetgate way and attach it to VPC

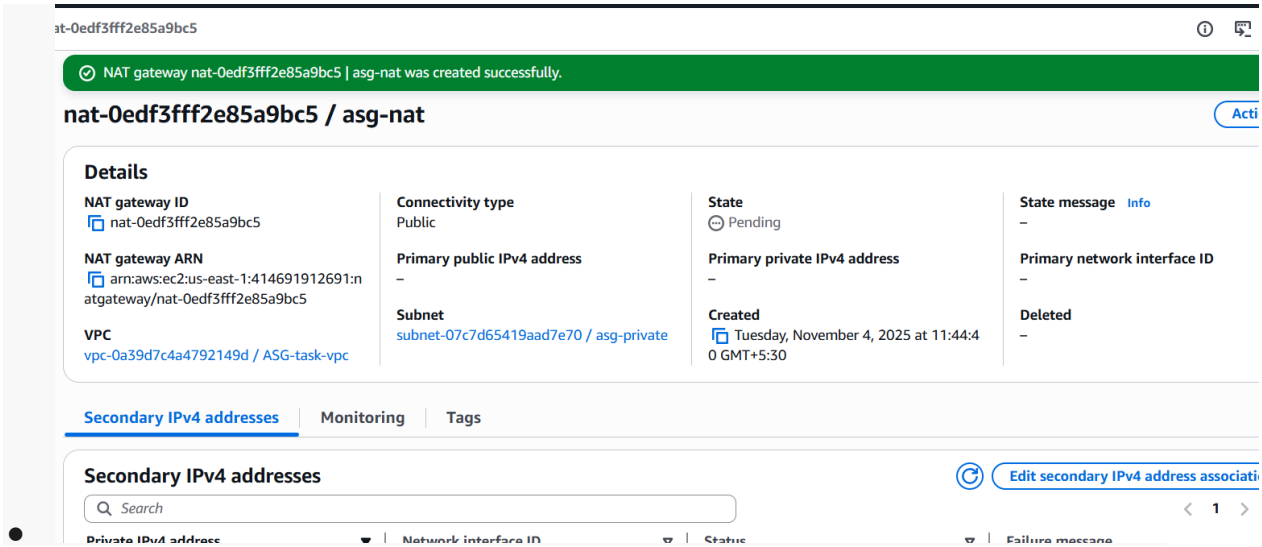


4. Create one public route table (RT) and one private route table.



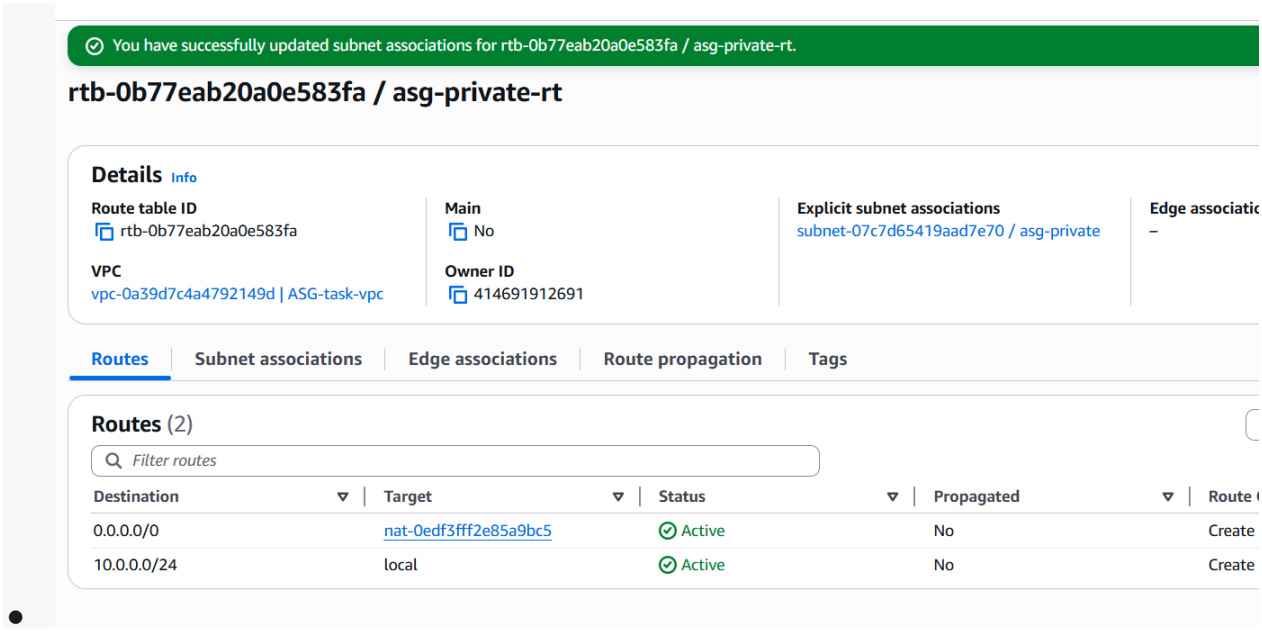
5. Deploy a NAT gateway in the public subnet and attach the NAT gateway to the private subnet.

- NAT gateway created in public subnet while creating
-



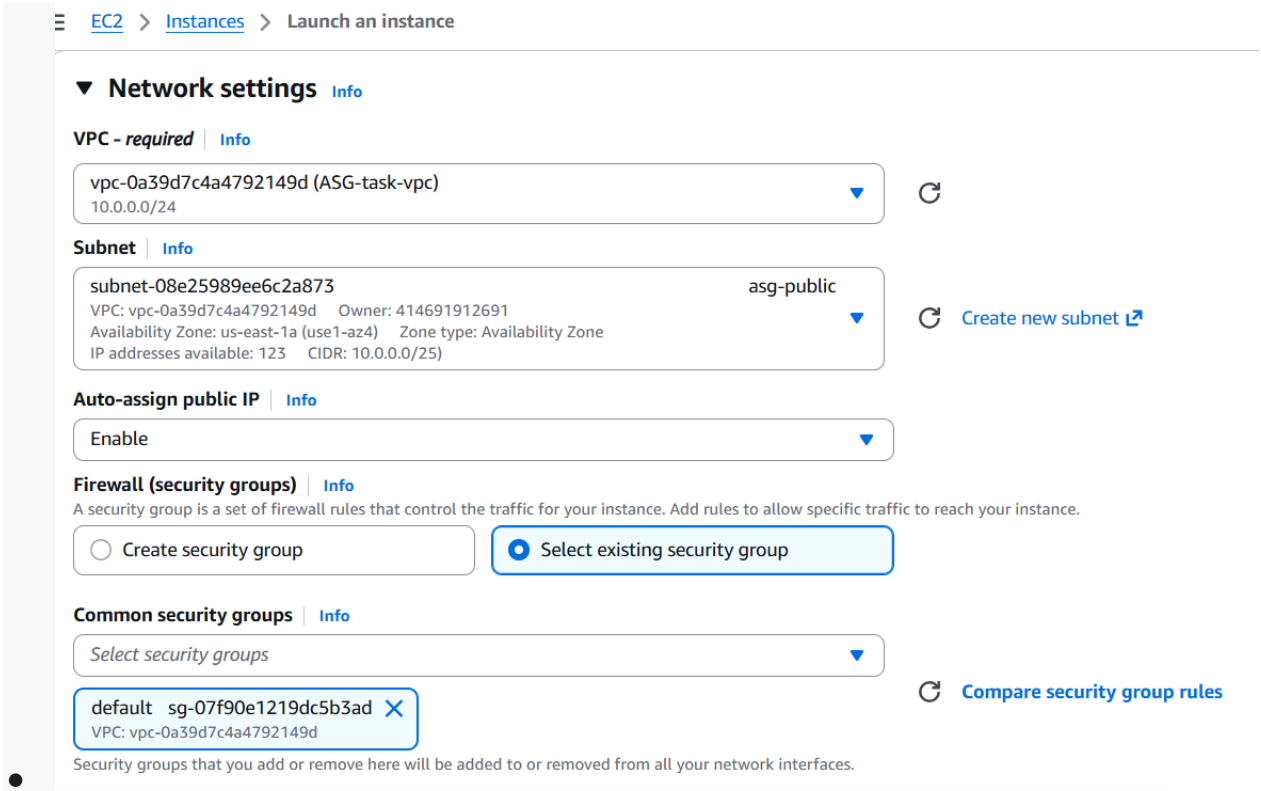
- Attaching NAT to private subnet in private route table

- Ad private subnet to private route table and edit route sand add NAT gateway

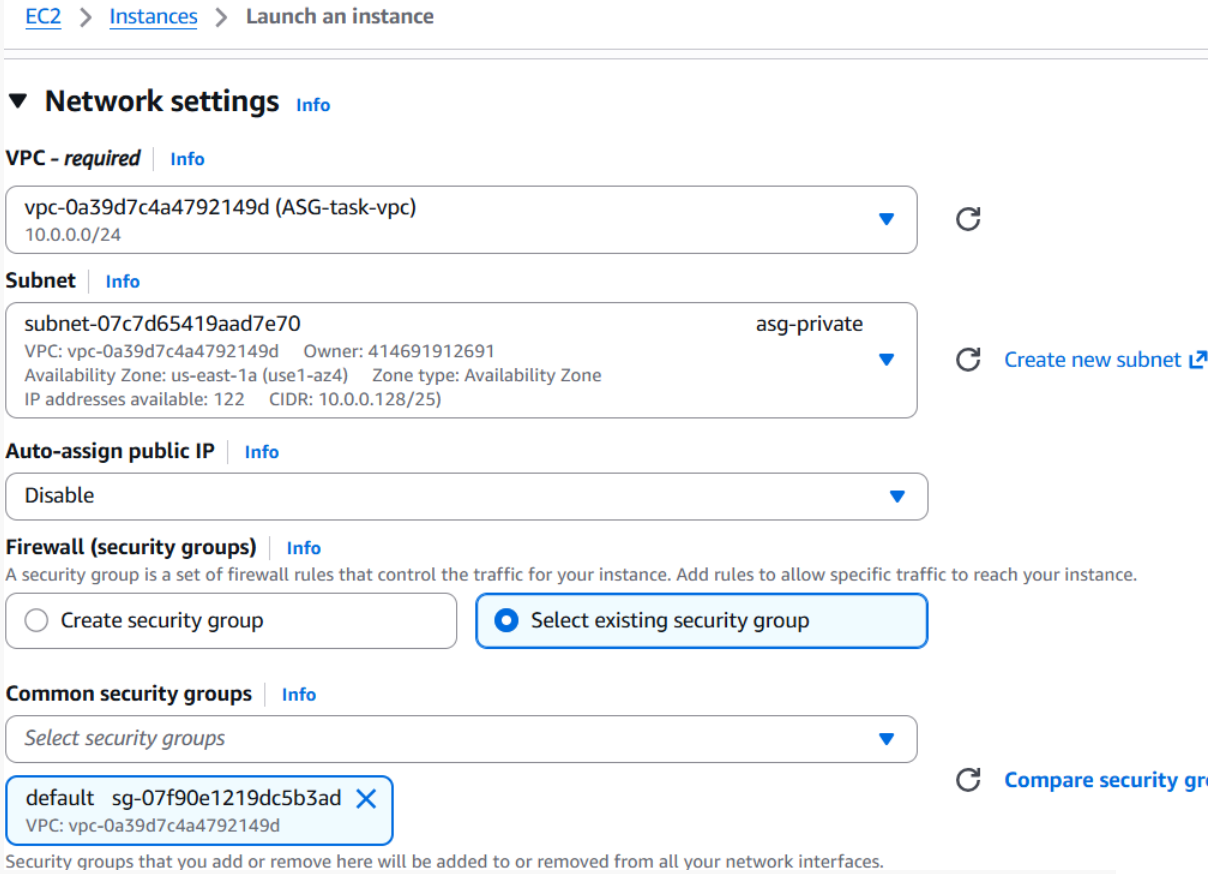


6. Create two instances, one in the public subnet and one in the private subnet.

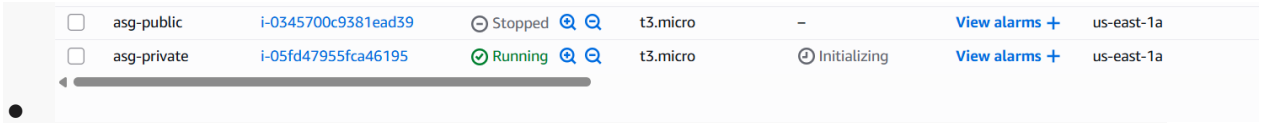
- Navigate ec2 and edit network setting add vpc and public subnet and security groups



- Second instance private
- Select vpc
- Select private subnet and default security groups



- Public instance and private instance



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

- Using jump server,
- Acces to public subnet instance
- Next, give chmod 400 (pem.key name)
- Now do ssh , using private instance ssh paste iit in the same terminal

```
[ec2-user@ip-10-0-0-114 ~]$ ssh -i "all-key.pem" ec2-user@10.0.0.197
The authenticity of host '10.0.0.197 (10.0.0.197)' can't be established
ED25519 key fingerprint is SHA256:a5gy+SRDTxcyUU98HilDXCtbDKQNIQPTYha/n
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? ye
Warning: Permanently added '10.0.0.197' (ED25519) to the list of known
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-0-197 ~]$ exit
logout
```

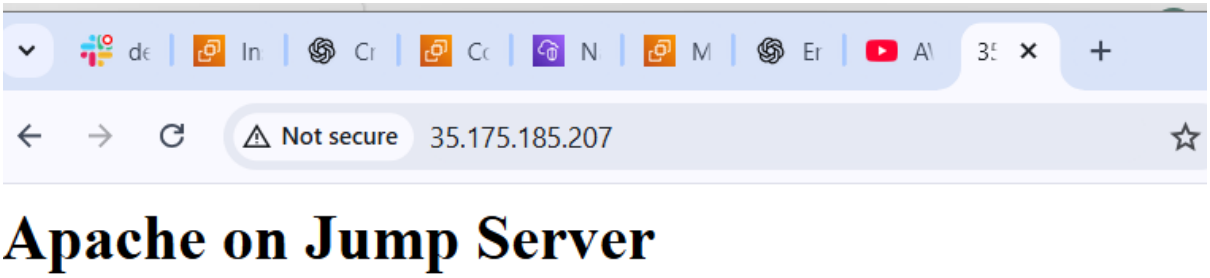
- Install httpd
- Sudo yum install httpd -y

```
Installed:
  apr-1.7.5-1.amzn2023.0.4.x86_64
  apr-util-1.6.3-1.amzn2023.0.2.x86_64
  apr-util-lmdb-1.6.3-1.amzn2023.0.2.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.2.x86_64
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.65-1.amzn2023.0.2.x86_64
  httpd-core-2.4.65-1.amzn2023.0.2.x86_64
  httpd-filesystem-2.4.65-1.amzn2023.0.2.noarch
  httpd-tools-2.4.65-1.amzn2023.0.2.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64
  mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64
  mod_lua-2.4.65-1.amzn2023.0.2.x86_64

Complete!
```

- Creating simple webpage
- echo "<h1>This is the PUBLIC Apache Server (Jump Server)</h1>" | sudo tee /var/www/html/index.html

```
Complete!
[ec2-user@ip-10-0-0-197 ~]$ sudo systemctl start httpd
sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.serv
ce → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-10-0-0-197 ~]$ cd /var/www/html/index.html
-bash: cd: /var/www/html/index.html: No such file or directory
[ec2-user@ip-10-0-0-197 ~]$ echo "<h1>This is the PUBLIC Apache Server
(Jump Server)</h1>" | sudo tee /var/www/html/index.html
<h1>This is the PUBLIC Apache Server (Jump Server)</h1>
[ec2-user@ip-10-0-0-197 ~]$ curl localhost
<h1>This is the PUBLIC Apache Server (Jump Server)</h1>
[ec2-user@ip-10-0-0-197 ~]$
```

-
- Verify



- Jump to private ip instance

```
[ec2-user@ip-10-0-0-114 ~]$ ssh -i "all-key.pem" ec2-user@10.0.0.197
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Tue Nov  4 09:19:20 2025 from 10.0.0.114
```

- Install httpd in private instance also

```
.
Package httpd-2.4.65-1.amzn2023.0.2.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

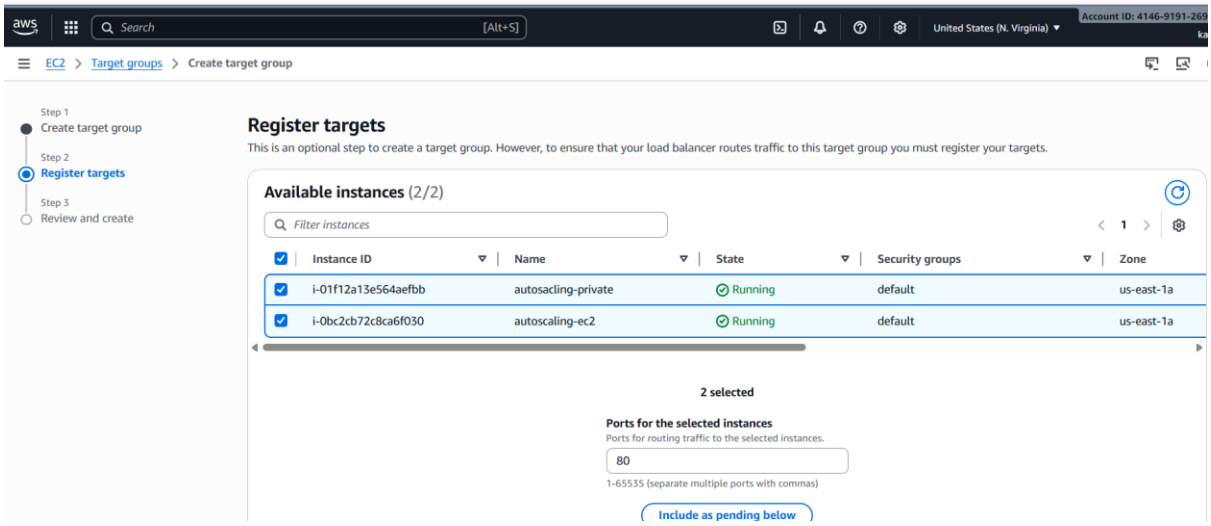
- Create a web page
- echo "<h1>Welcome! This is the PRIVATE Apache Server</h1>" | sudo tee /var/www/html/index.html
- now from private instance curl public ip so you can see apache working page

```
[ec2-user@ip-10-0-0-197 ~]$ curl http://35.175.185.207
<h1>Apache on Jump Server</h1>
[ec2-user@ip-10-0-0-197 ~]$
```

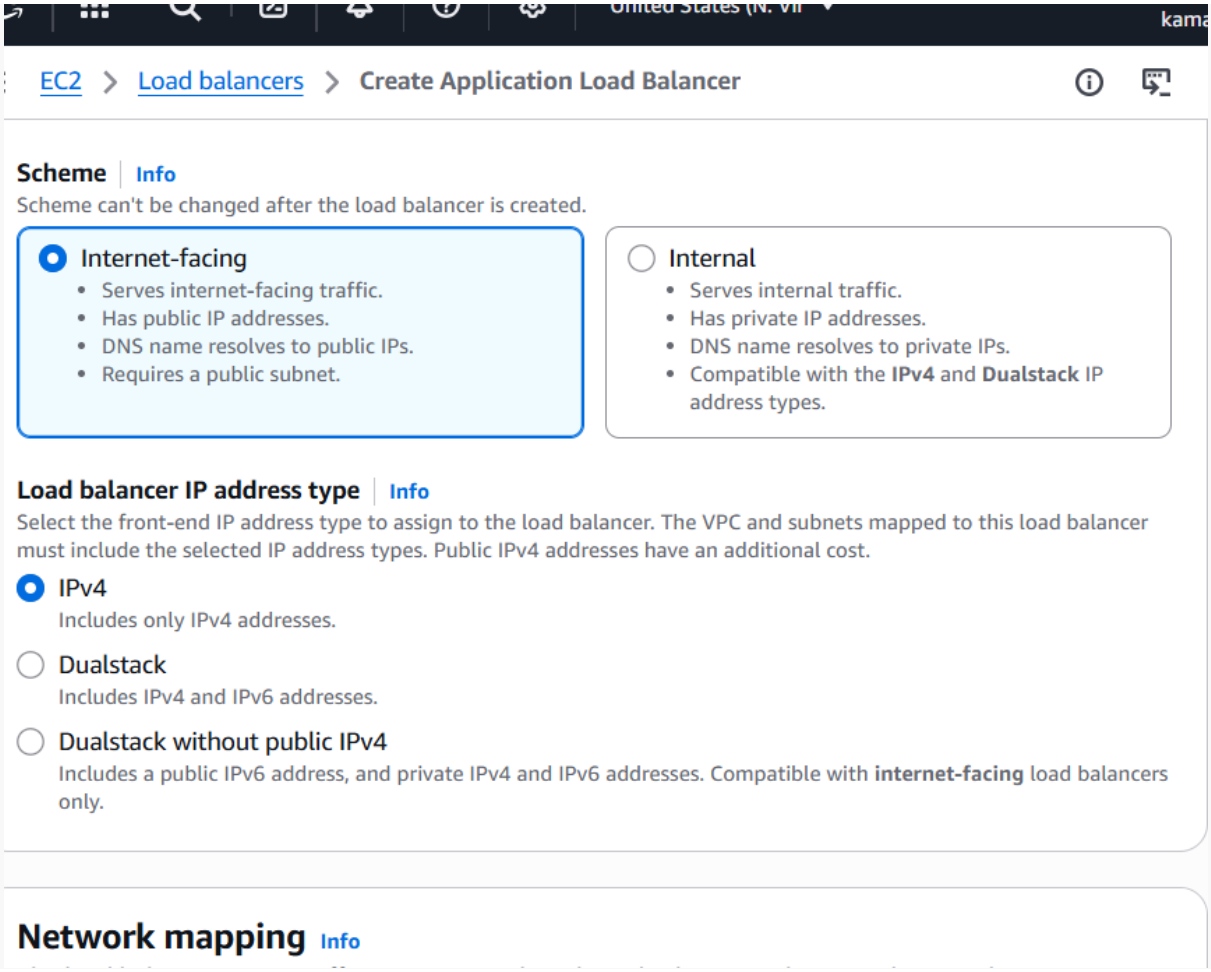
8. Create one application load balancer and attach it to both EC2 instances.

- Open aws console
- Navigate to ec2 > target groups
- Give name, select target as instance
- Select vpc of existing subnets
- Select public subnet and privste subnet

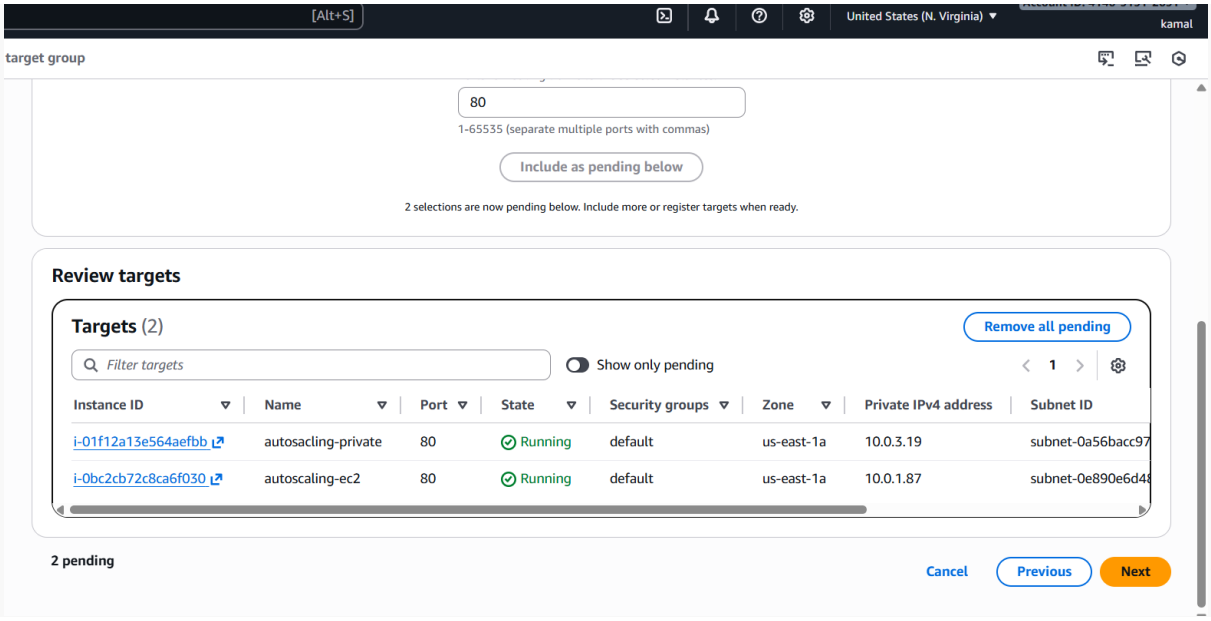
- Now navigate to load balancer
- Creatge new
- Select application load balancer



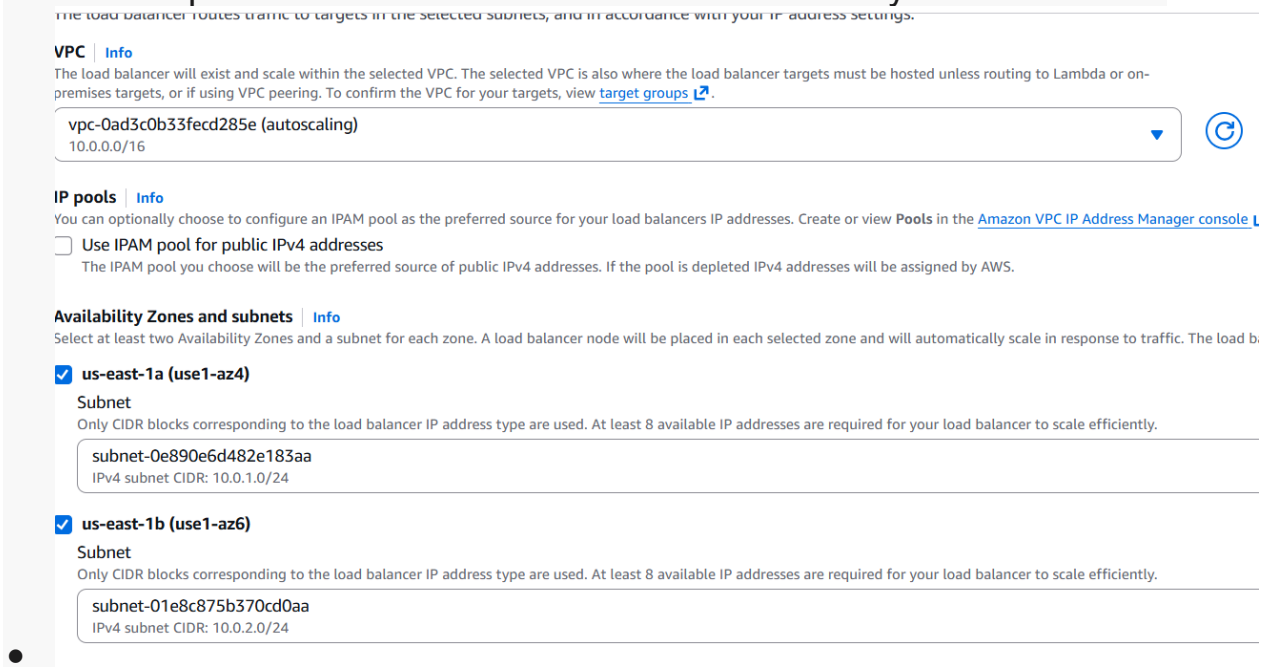
Select internet facing in scheme

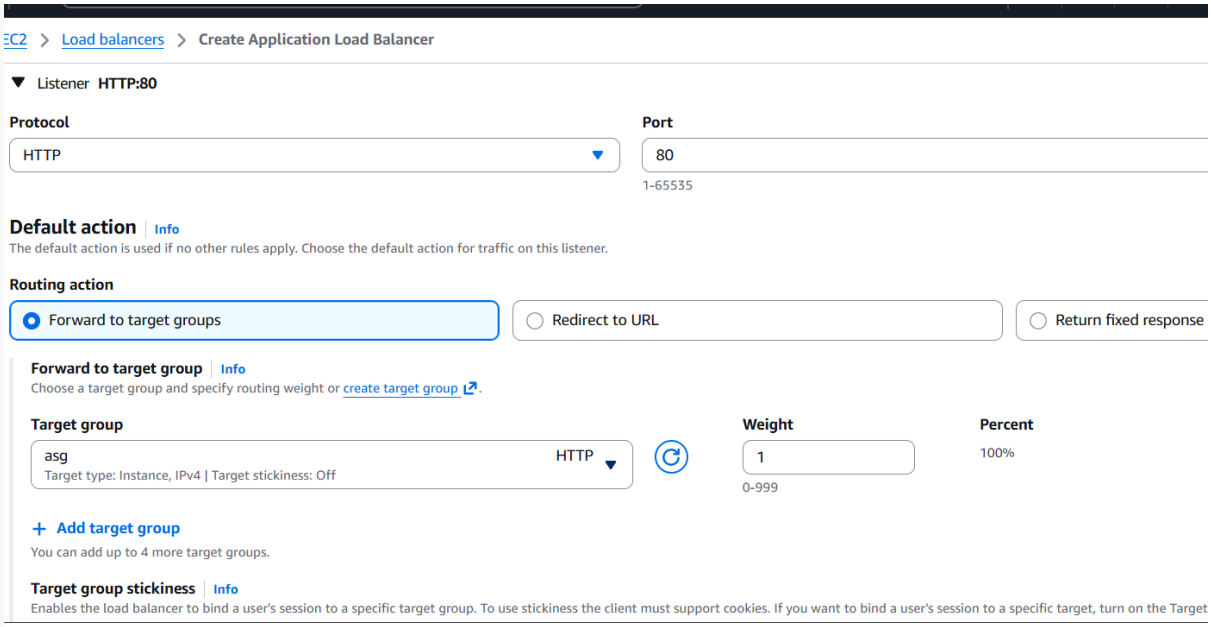


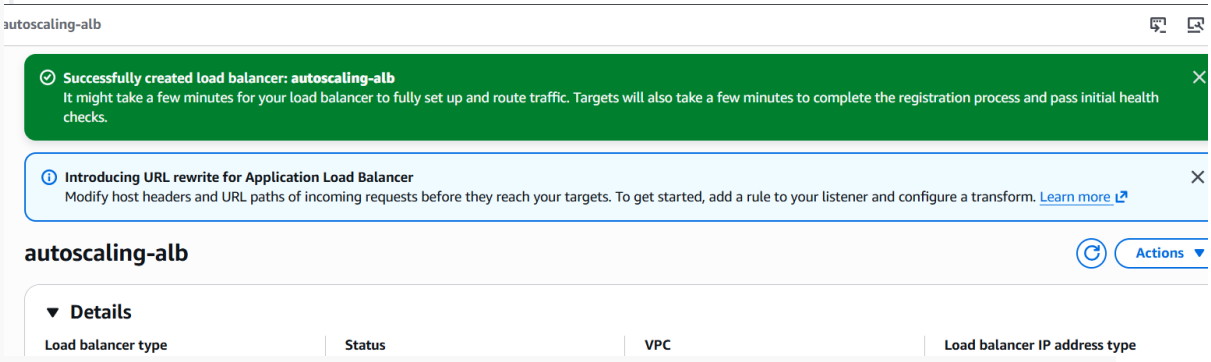
- Click include as pending below



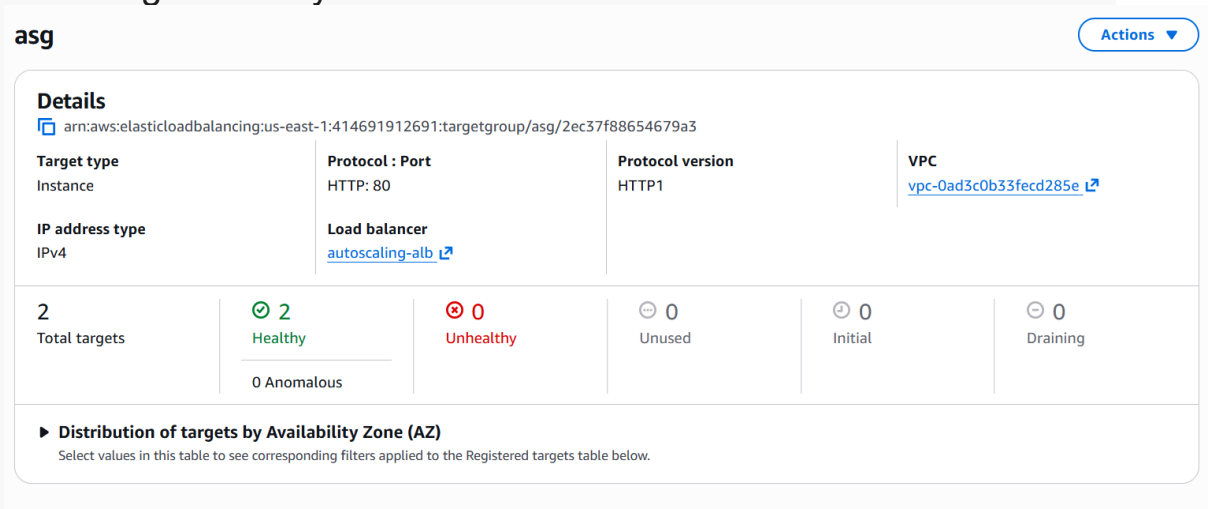
- Target created successfully
- Now , create a application nload balancer
- Navigasge to load balancer in ec2
- Select vpc and 2 subnet s from different availability zone



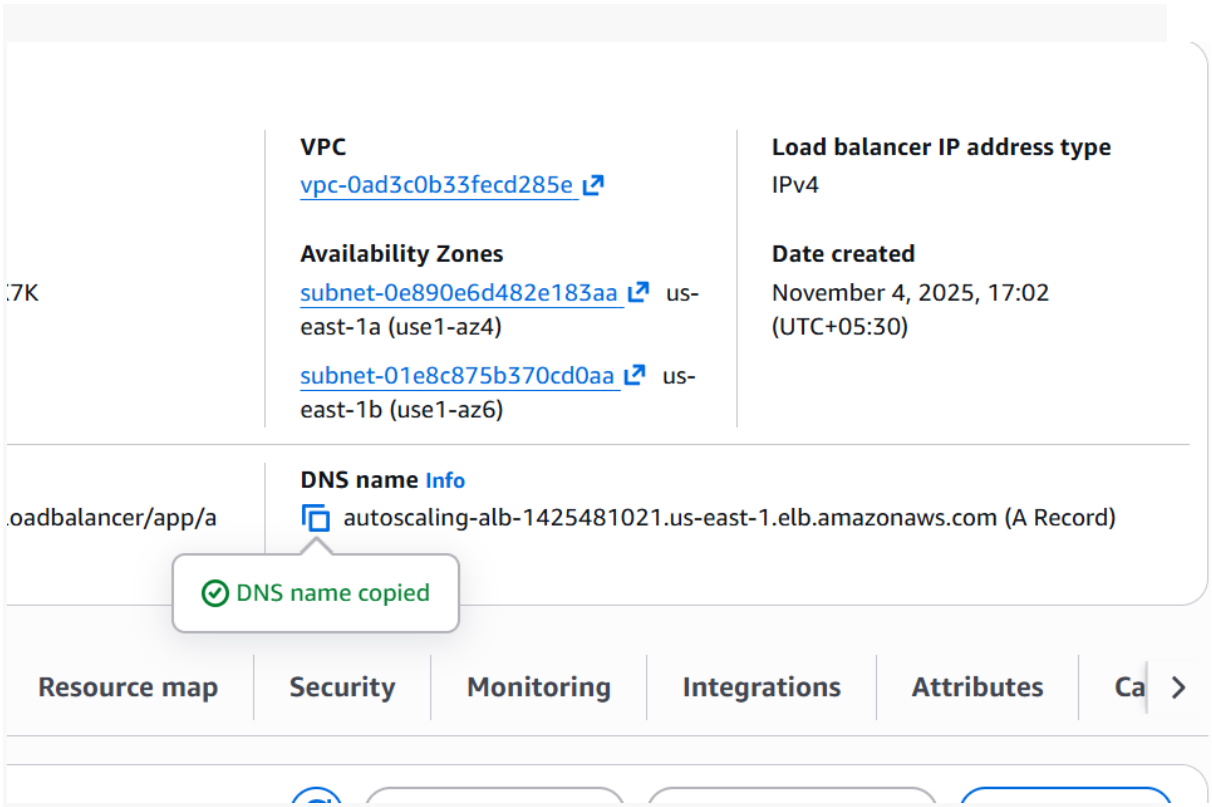


- 

- Everthing is healthy

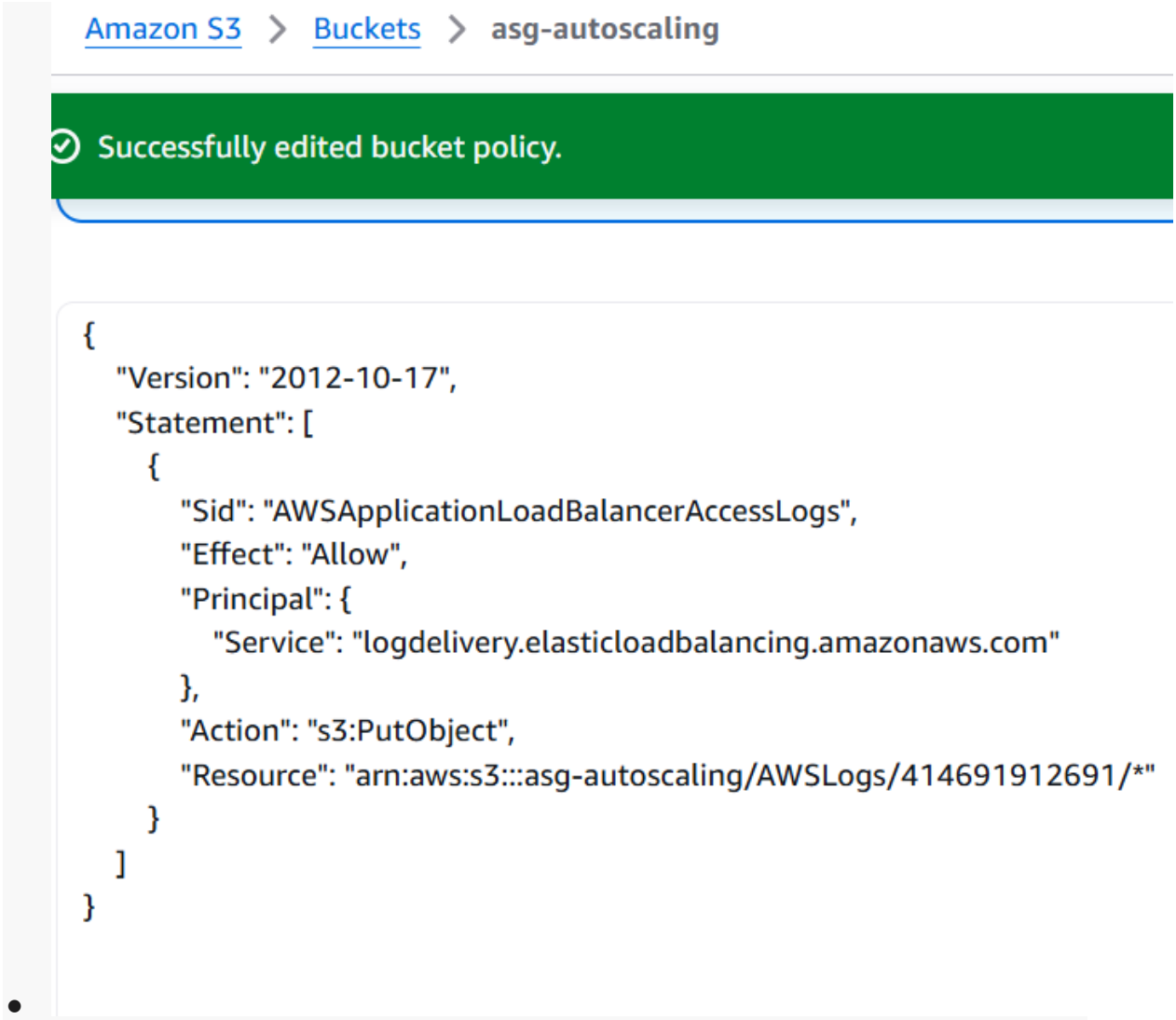


- Copy dns name from load balancer

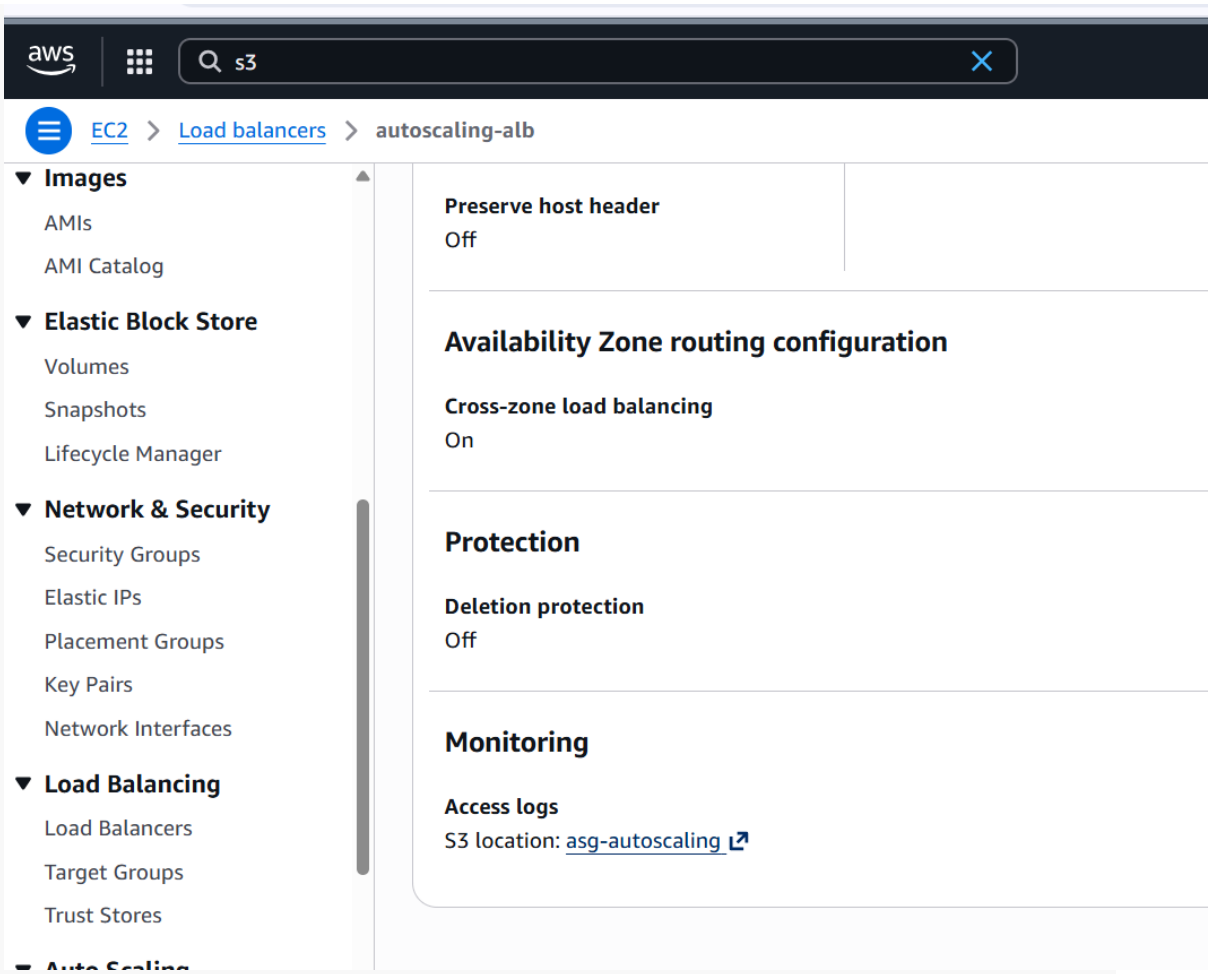


- Paste in local browser

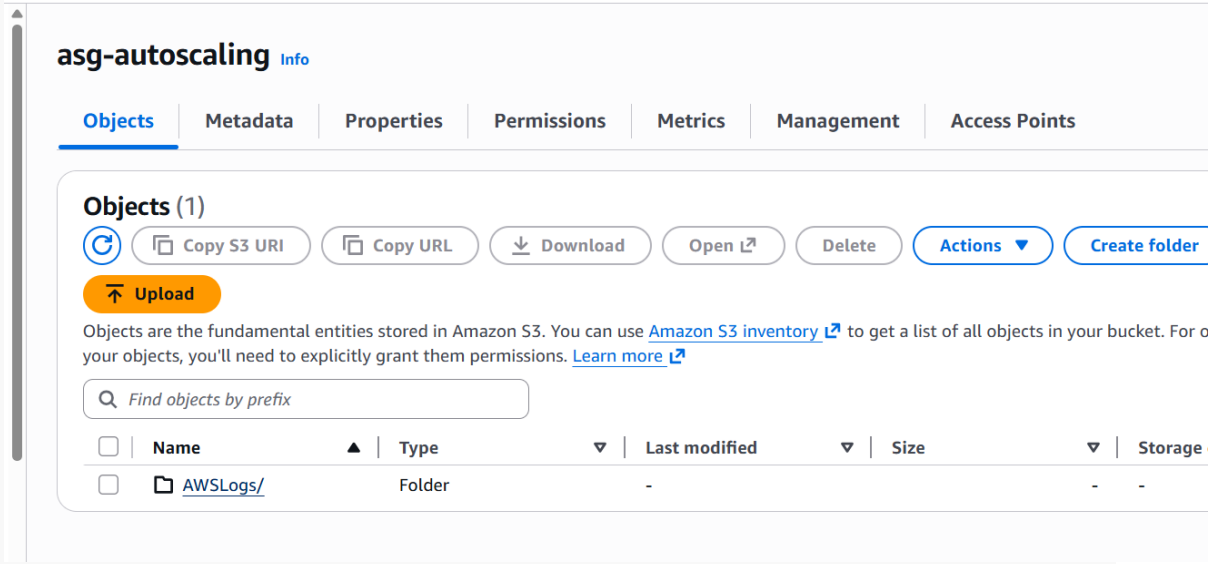
-
- It is on load balancer
- 9 . Store application load balancer logs in S3.
- Navigate to s3 bucket
- Create a bucket enable block all ip
- Open bucket and click permission
- Edit and give script



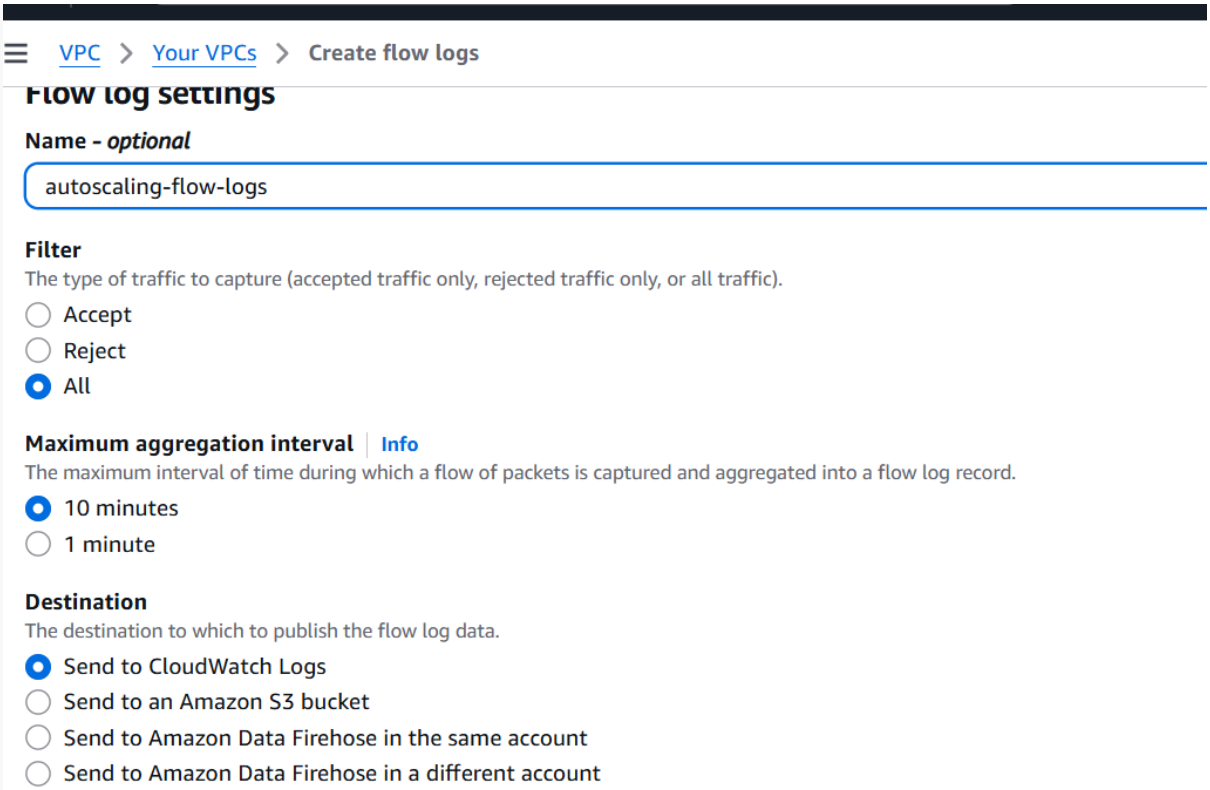
- Now navigate to , ec2 > load balancer,
- Edit attributes
- Click access logs
- Click s3 bucket add existing bucket
- Can monitoring logs in > load balancer > below the attributes



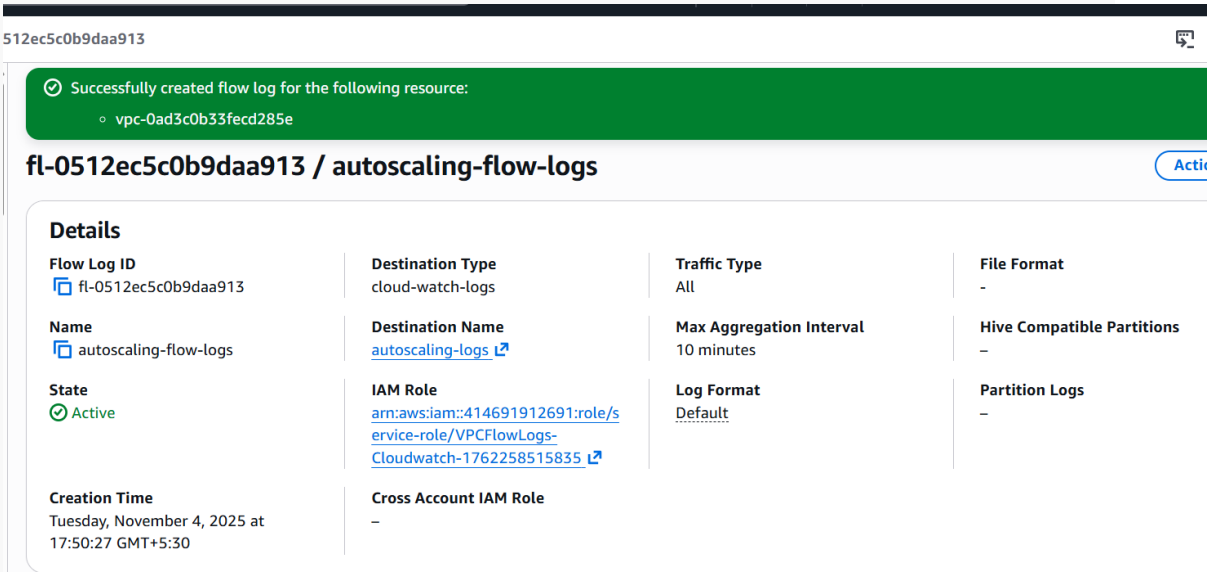
- Can find the logs in s3 buckets



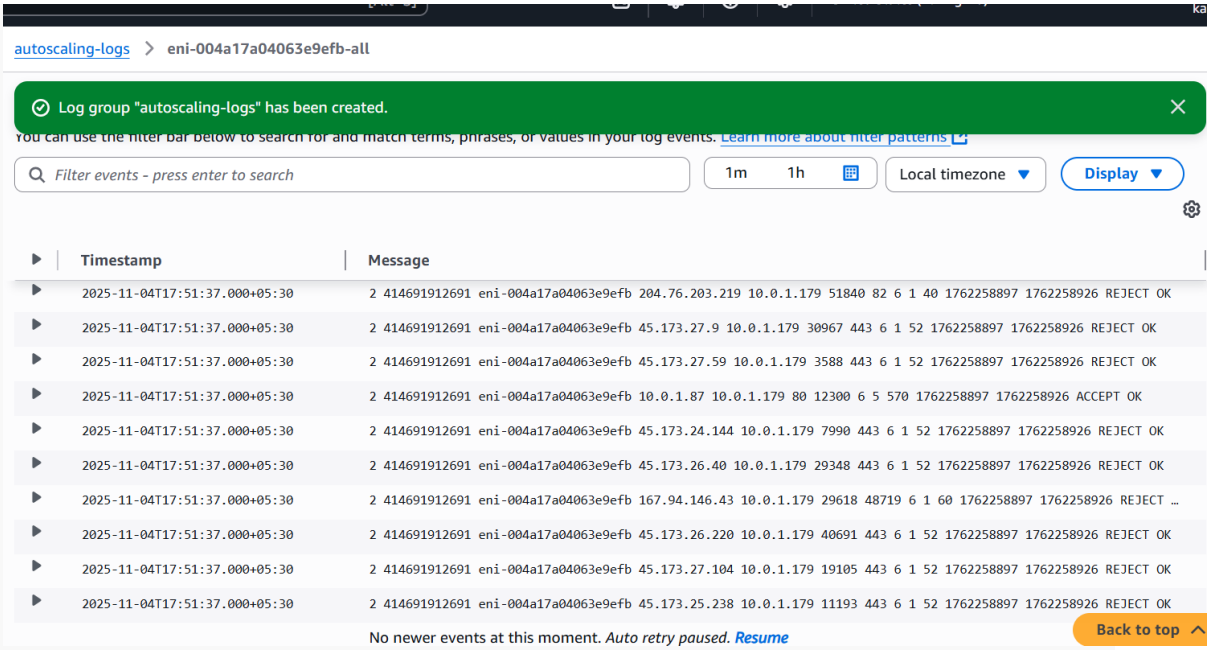
- 10 . Store the VPC flow logs in a CloudWatch log group.
- Navigate to vpc
- Select vpc
- Below find flow logs
- And create and add , destination and time



- Navigate to cloud watch and create a log group
- And return back to vpc and refresh you can see cloudwatch log group
- Select log group
- Select new role
- And then create

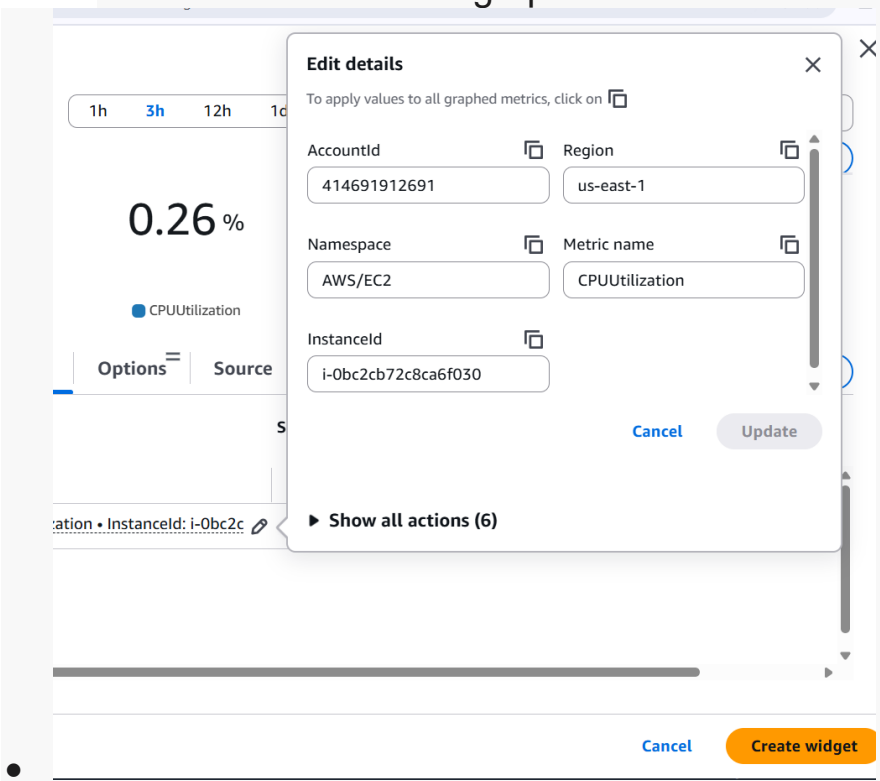


- Cerify in cloud watch

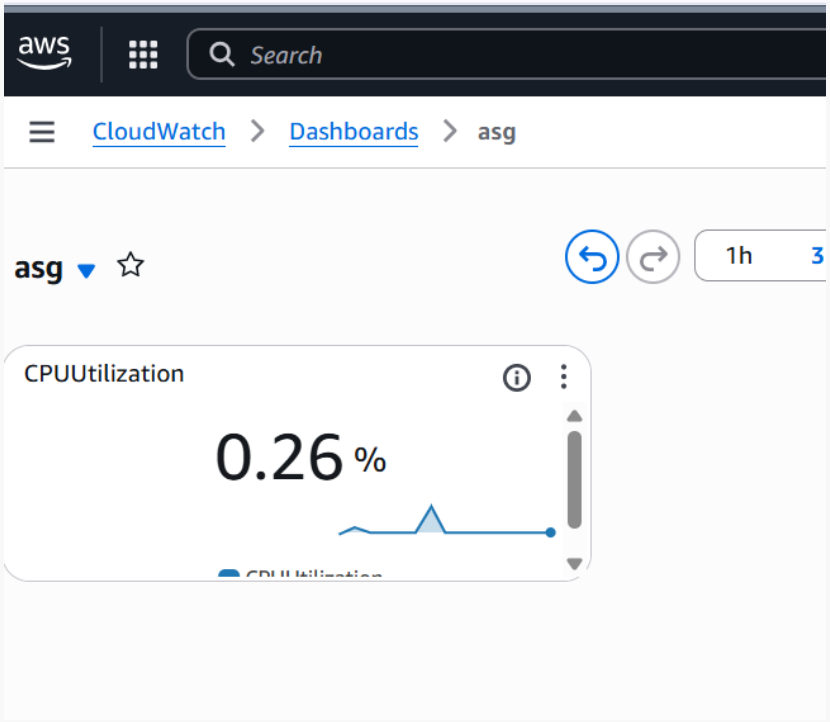


11. Create monitoring dashboards to monitor CPU utilization and to monitor the Apache service.

- Cloud watch
- Create new dashboard
- Add ec2 in metrics
- Pre instance select cpu utilization add
- Add instsance id in graphed metrics



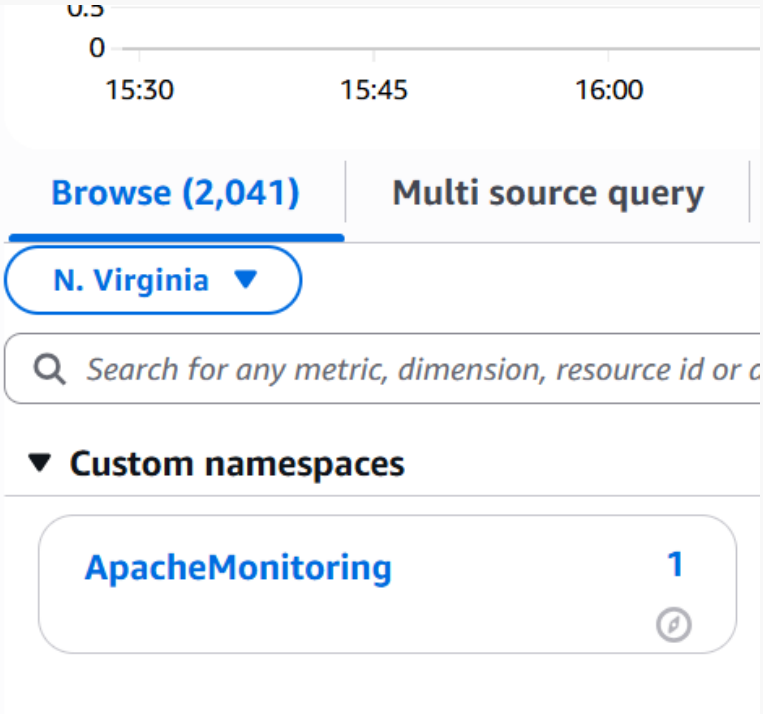
- Created a dashboard for instance

- 

- Monitor apache is riunning or not
- Open cli check aws is configured or not
- Now create file and add script for monitor apache running or not
    - #!/bin/bash
    -
    - # Check Apache (httpd) status
    - status=$(systemctl is-active httpd)
    -
    - if ["$status" == "active"]; then
    - metric_value=1
    - else
    - metric_value=0
    - fi
    -
    - # Push the metric to CloudWatch
    - aws cloudwatch put-metric-data \
    - --metric-name ApacheServiceStatus \
    - --namespace "ApacheMonitoring" \
    - --value $metric_value \
    - --region us-east-1
    - Install corntab and edit it
    - By sudo corntab -e
    - Add */5 * * * *

- Run this command in CLI to create ac custom name space in gui

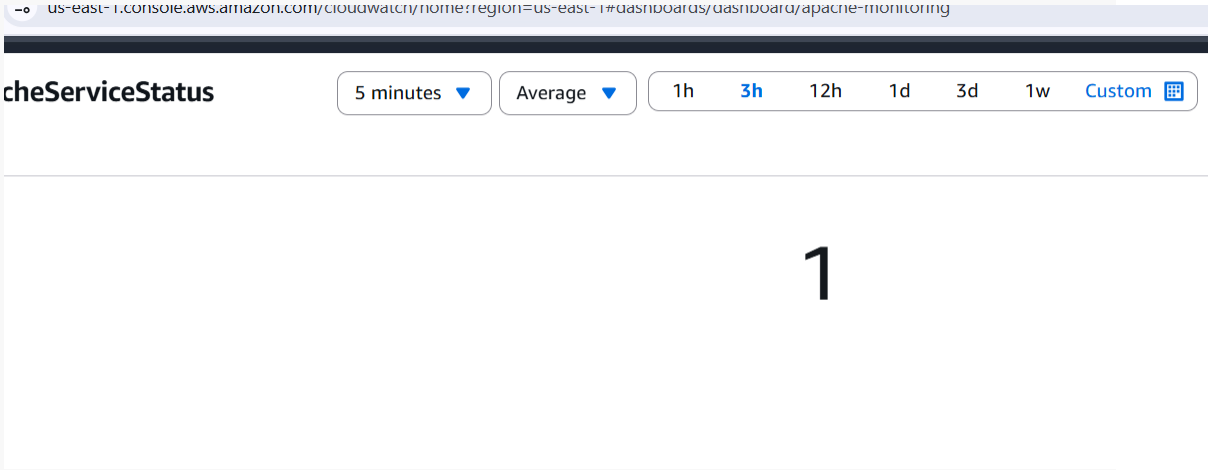
```
aws cloudwatch put-metric-data \
  --metric-name ApacheServiceStatus \
  --namespace "ApacheMonitoring" \
  --value 1 \
  --region us-east-1
```



- As apache running in cli
-

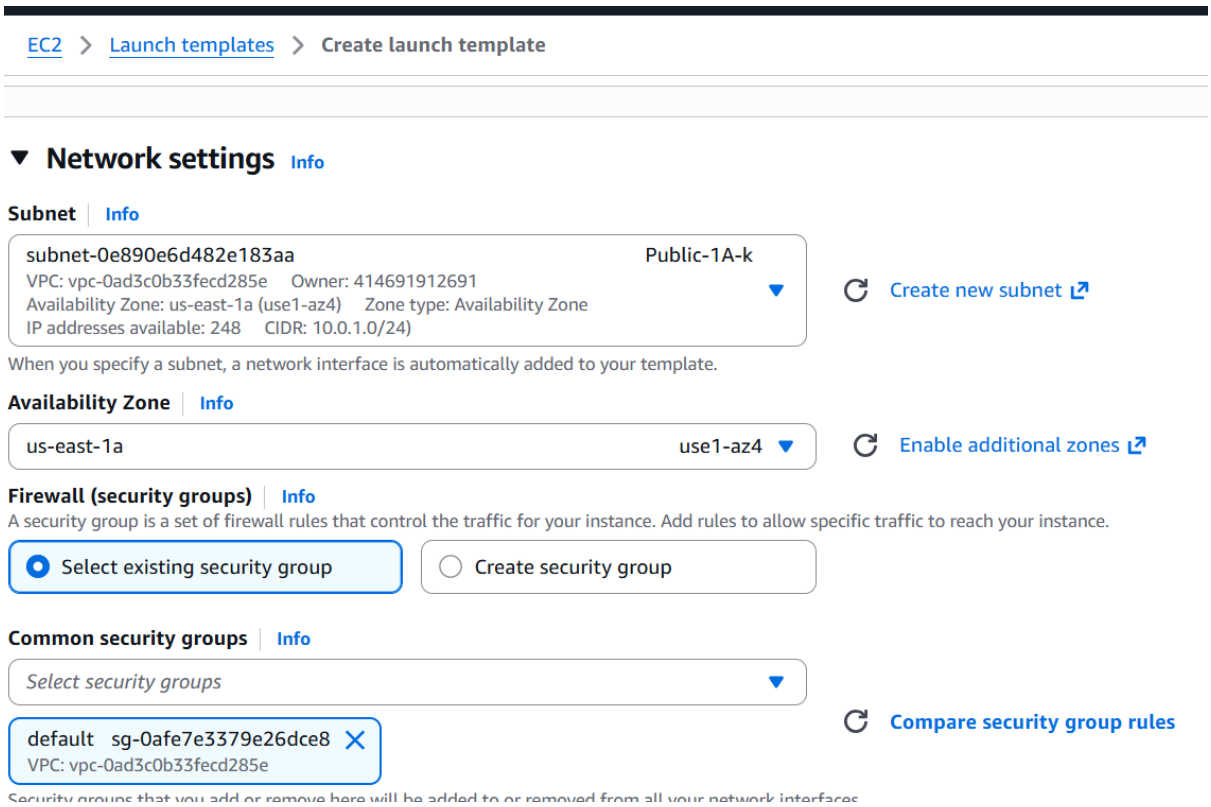
```
[ec2-user@ip-10-0-1-87 ~]$ aws cloudwatch put-metric-data \
  --metric-name ApacheServiceStatus \
  --namespace "ApacheMonitoring" \
  --value 1 \
  --region us-east-1
[ec2-user@ip-10-0-1-87 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset:>
     Active: active (running) since Tue 2025-11-04 11:46:20 UTC; 1h 15min ago
       Docs: man:httpd.service(8)
   Main PID: 25951 (httpd)
     Status: "Total requests: 332; Idle/Busy workers 100/0;Requests/sec: 0.0>
      Tasks: 177 (limit: 1053)
     Memory: 16.2M
        CPU: 4.273s
     CGroup: /system.slice/httpd.service
             ├─25951 /usr/sbin/httpd -DFOREGROUND
             ├─25952 /usr/sbin/httpd -DFOREGROUND
             ├─25954 /usr/sbin/httpd -DFOREGROUND
             ├─25955 /usr/sbin/httpd -DFOREGROUND
             └─25956 /usr/sbin/httpd -DFOREGROUND

Nov 04 11:46:20 ip-10-0-1-87.ec2.internal systemd[1]: Starting httpd.service>
Nov 04 11:46:20 ip-10-0-1-87.ec2.internal systemd[1]: Started httpd.service >
Nov 04 11:46:20 ip-10-0-1-87.ec2.internal httpd[25951]: Server configured, l>
lines 1-19/19 (END)
```

- We can see in dashboard as 1

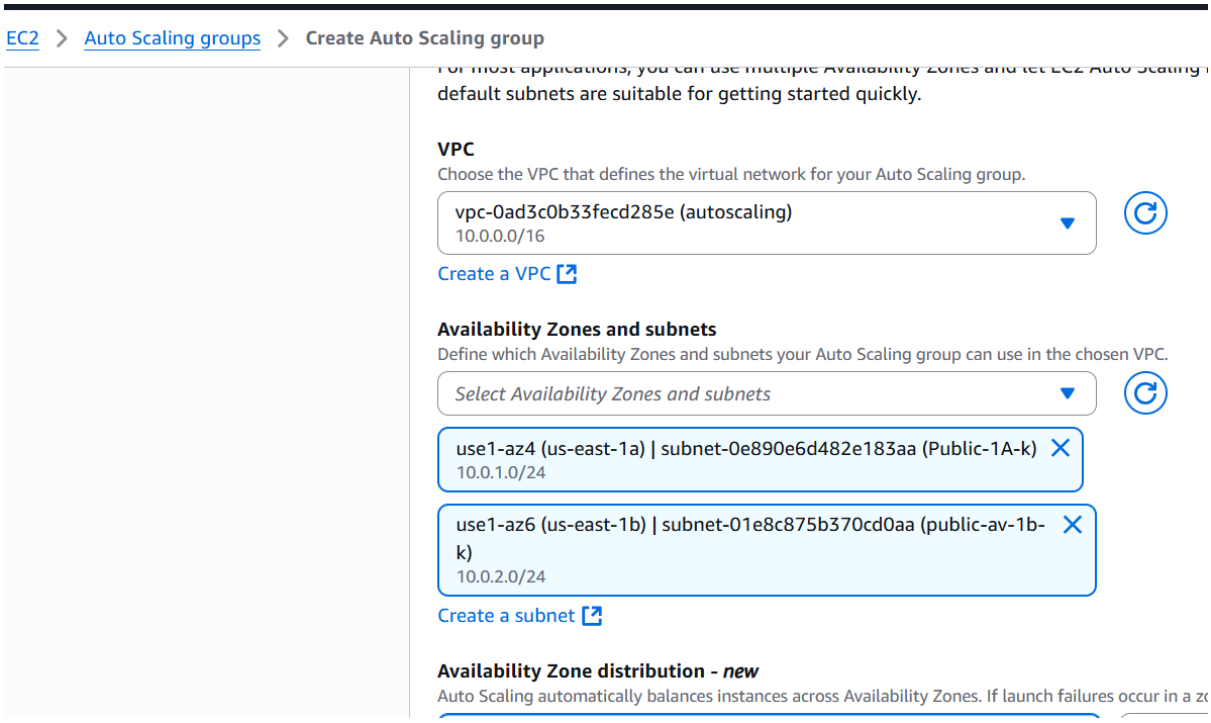


12. If CPU utilization is more than 70%, then it should trigger auto scaling and launch new instance.

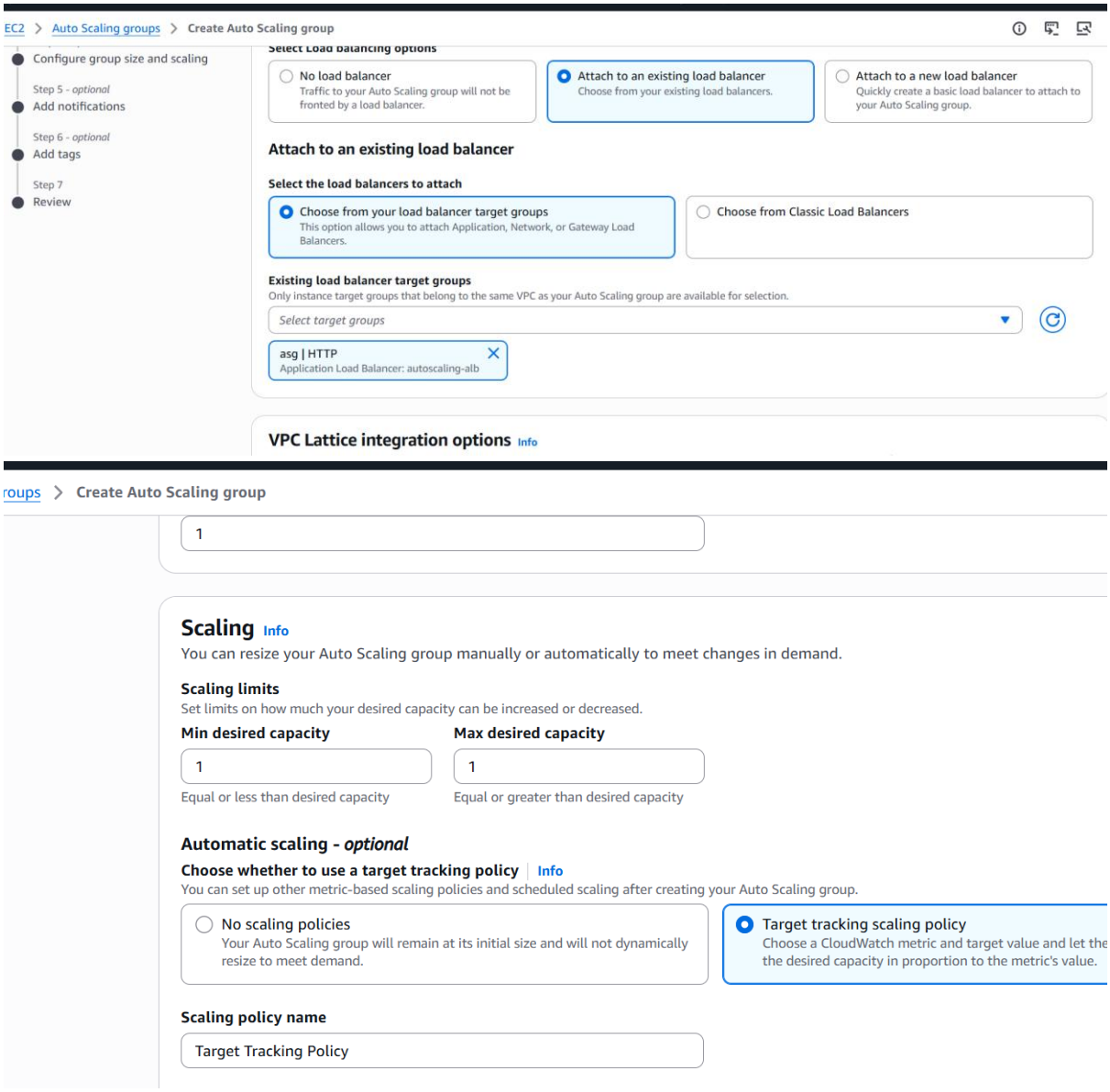
- Open aws console
- Navigate to launch template
- Give configuration as needed for autoscaling



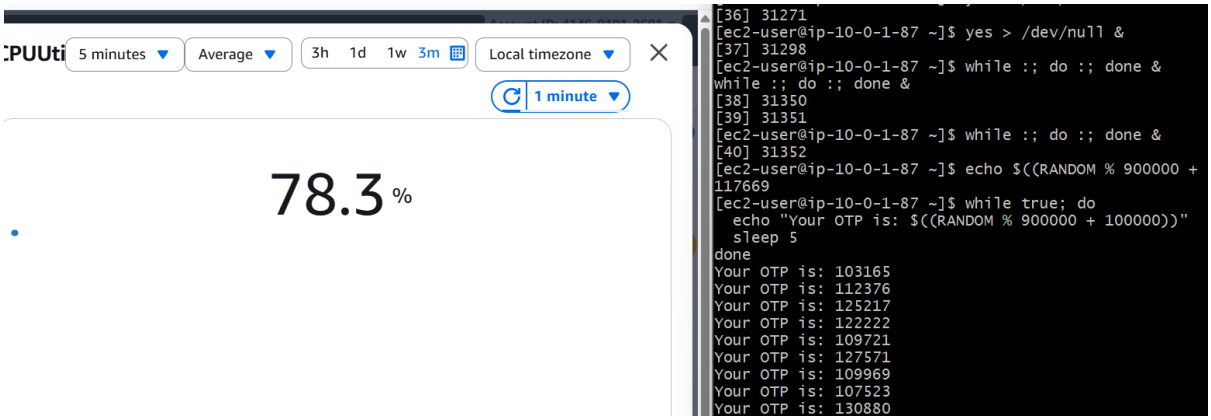
- Now navigagte autoscaling and create new
- Add launch template to this
- Add vpc and 2 subnets atleast



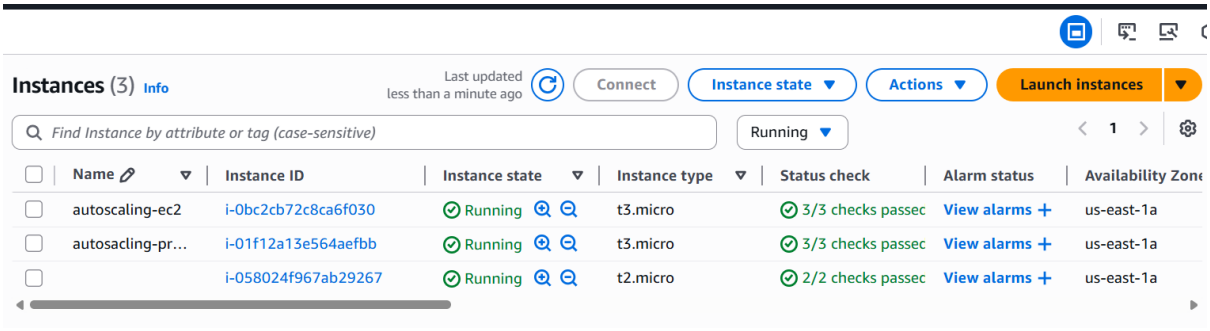
- Add existing load balancer
-

- 

-

- Load increased in monitor is working
- 

- So new instance launched automatically as its cpu usage increased abouve 50%


Instances (3) Info
Last updated
less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
Running
1
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
autoscaling-ec2
i-0bc2cb72c8ca6f030
Running
t3.micro
3/3 checks passec
View alarms
us-east-1a
autosacling-pr...
i-01f12a13e564aefbb
Running
t3.micro
3/3 checks passec
View alarms
us-east-1a
i-058024f967ab29267
Running
t2.micro
2/2 checks passec
View alarms
us-east-1a