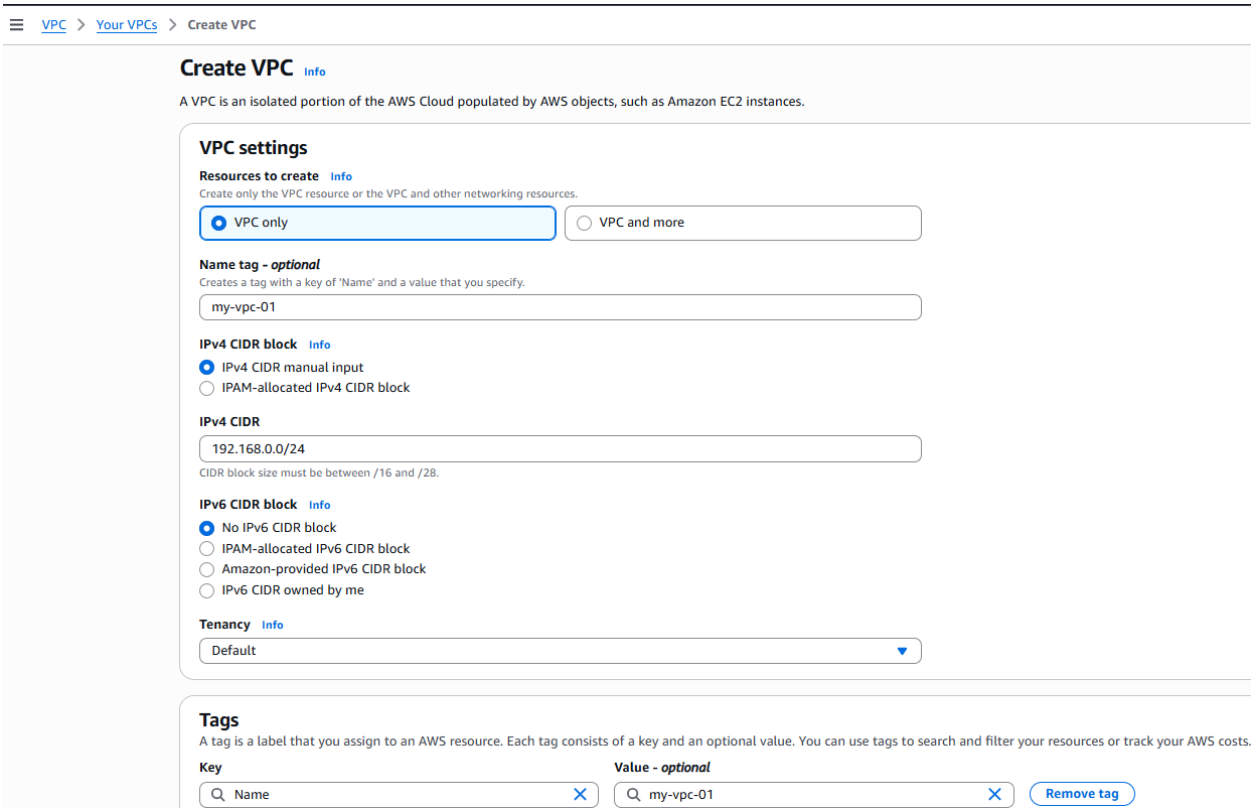
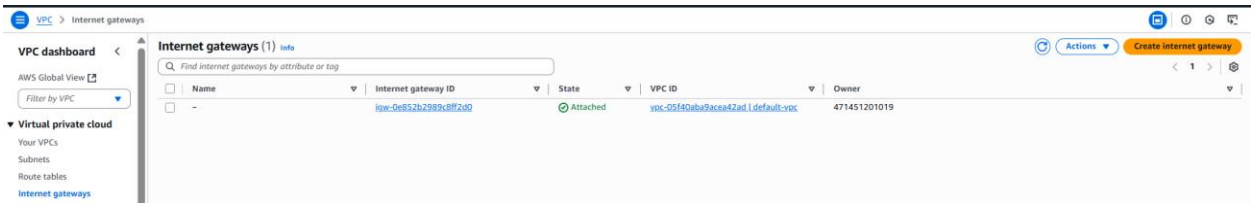
1. Create one VPC in N. Virginia region.

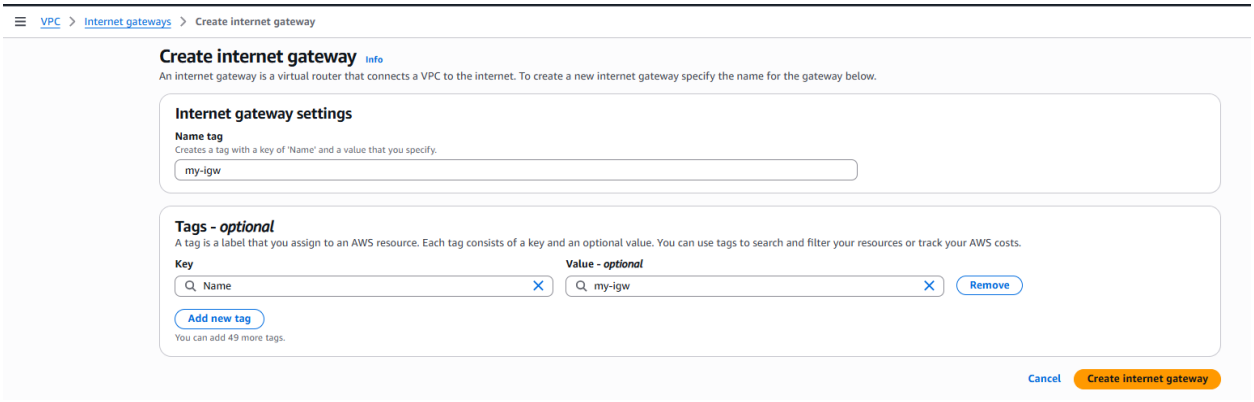
   Goto VPC in N. Virginia region and create, give CIDR range:



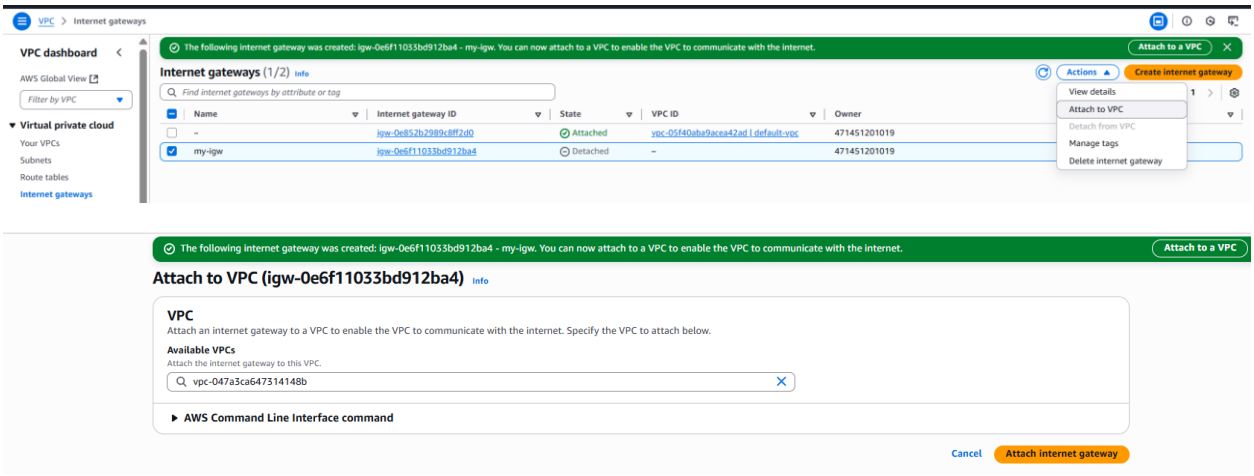
2. Create two subnets: one public subnet and one private subnet.

   Create an Internet Gateway:

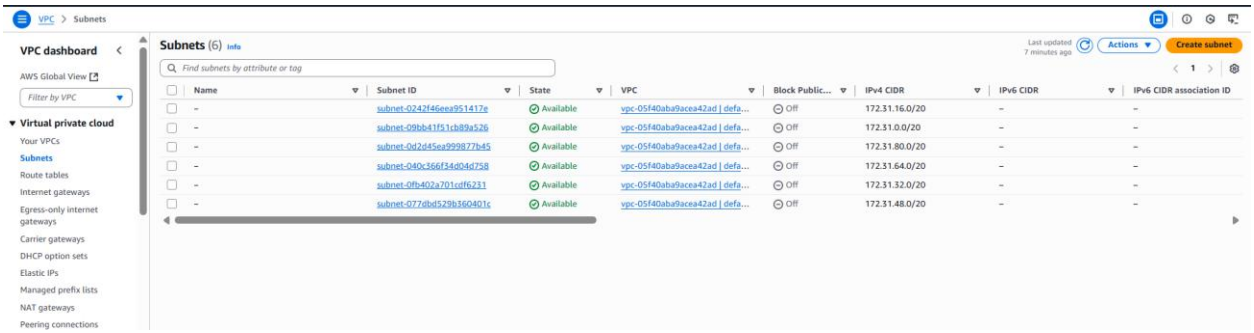




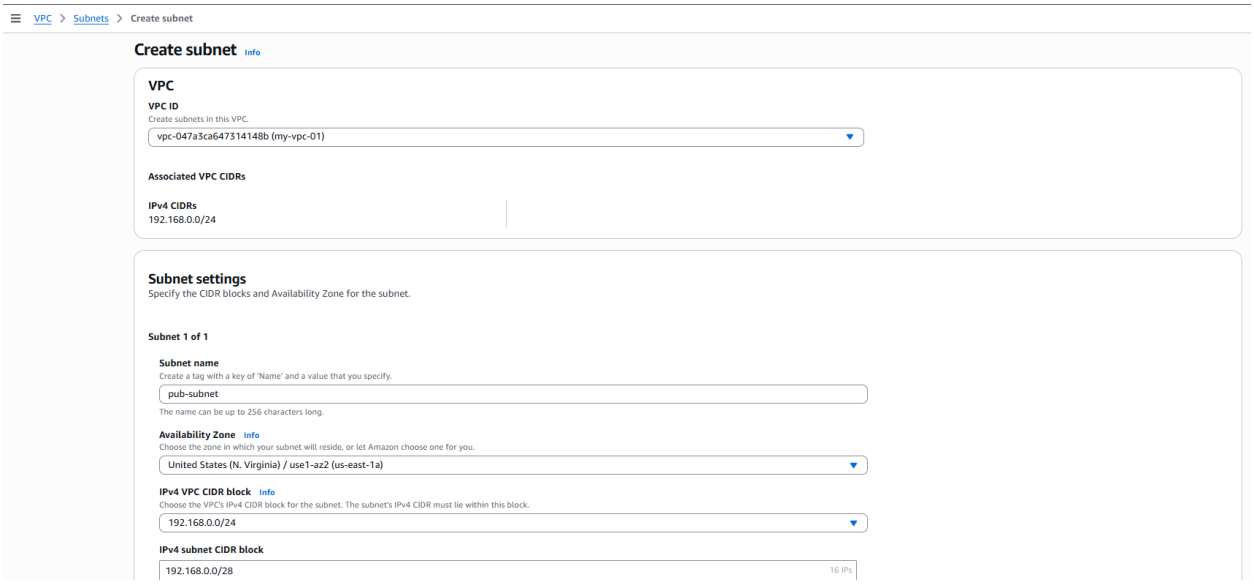
And attach to your vpc:



Create Subnet:

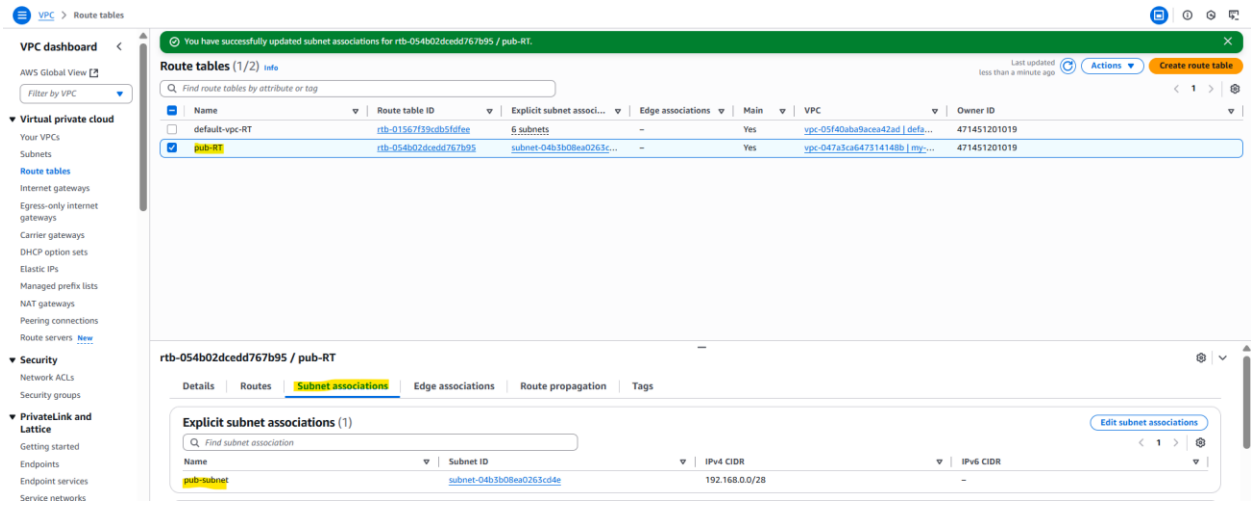


Give the CIDR of subnet:

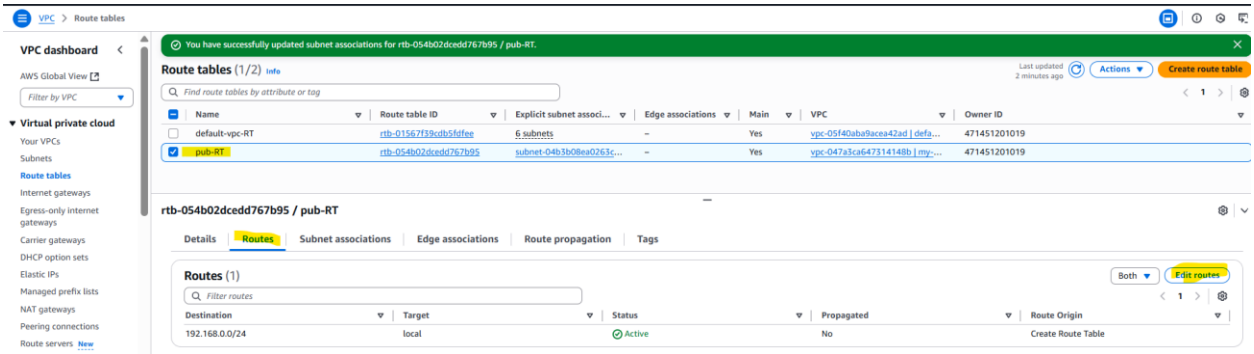


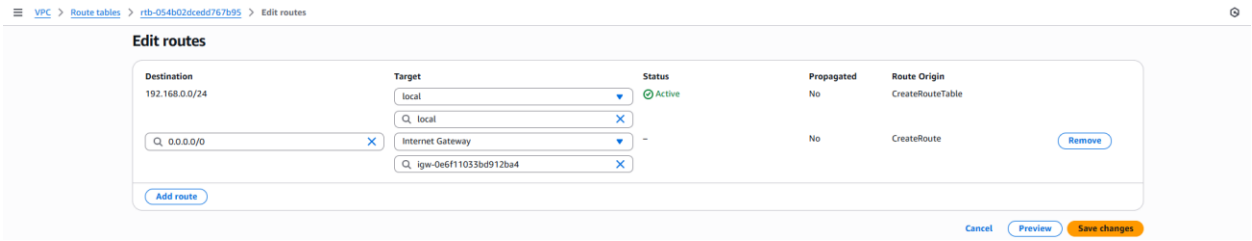
Associate the Route Table with subnet:

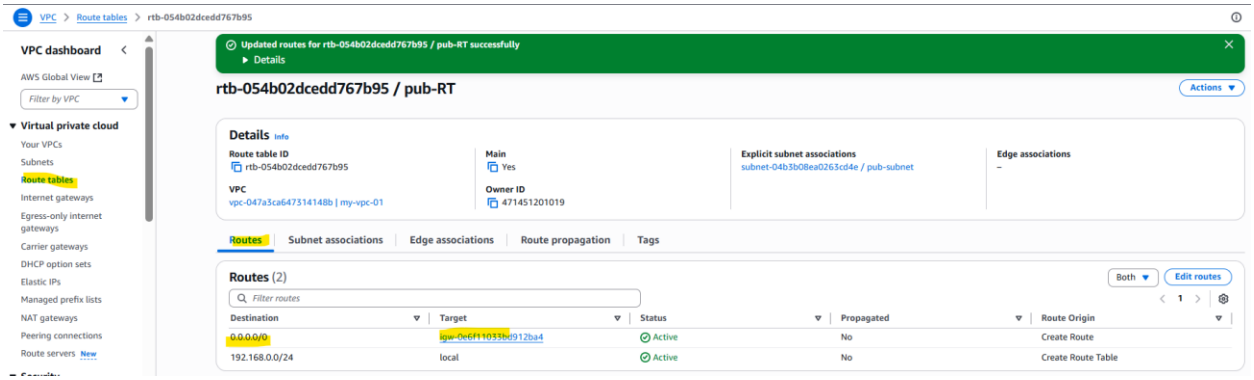
Note: here I have used the Route Table which got created with the VPC creation:



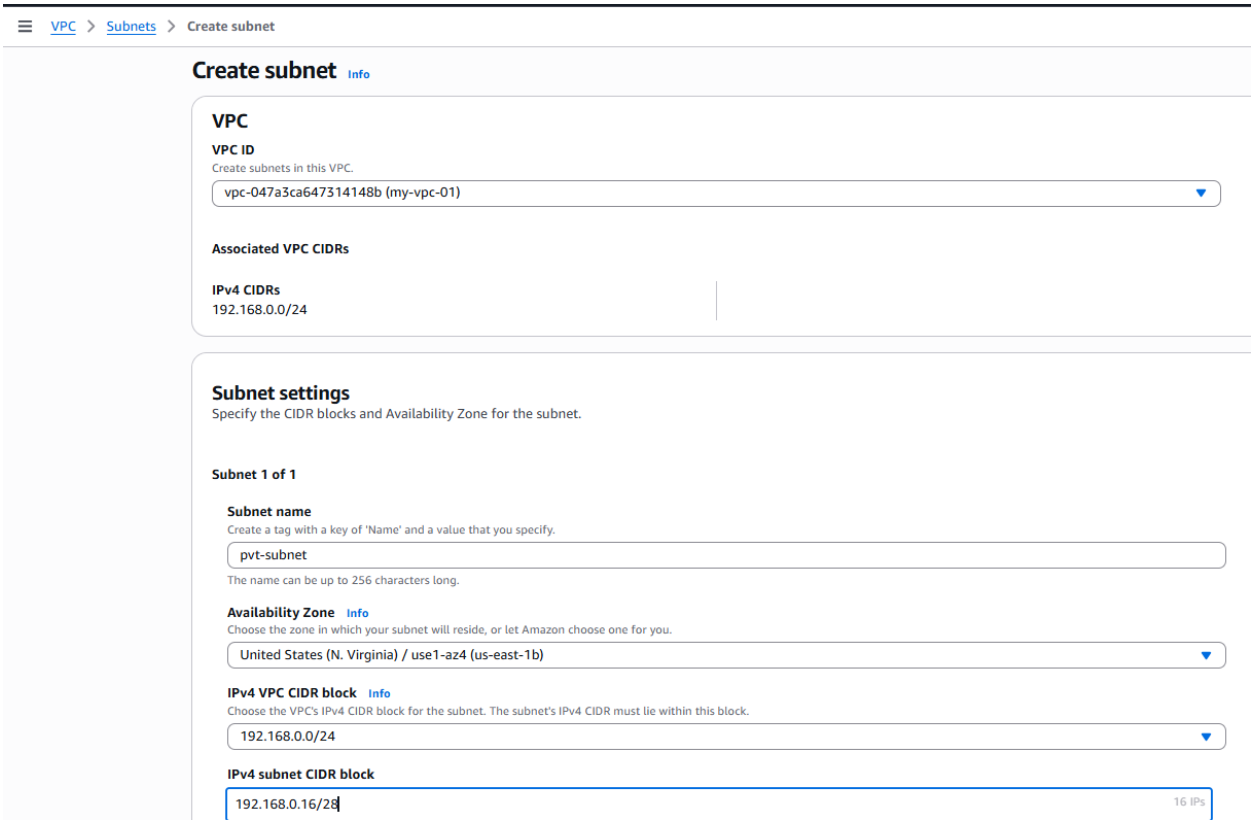
Add the Route i.e., IGW to the Route Table Route:



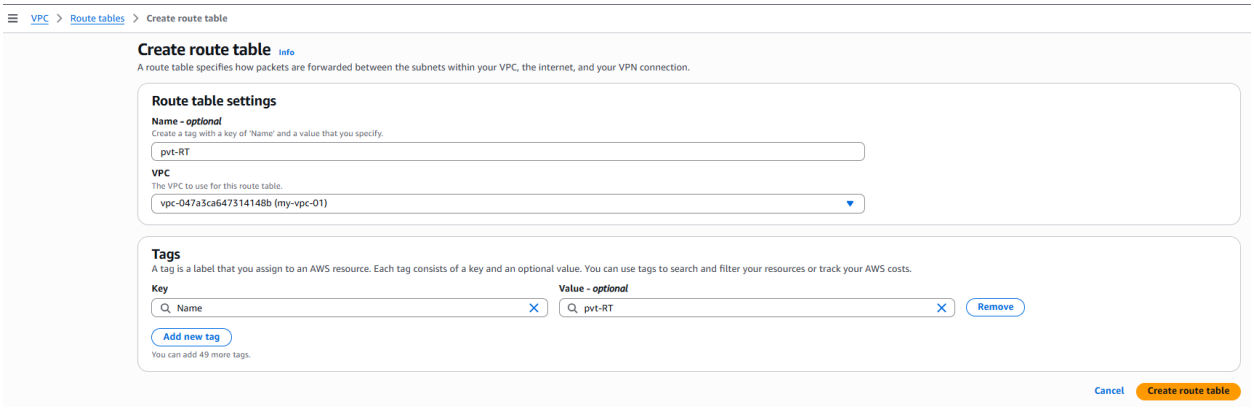




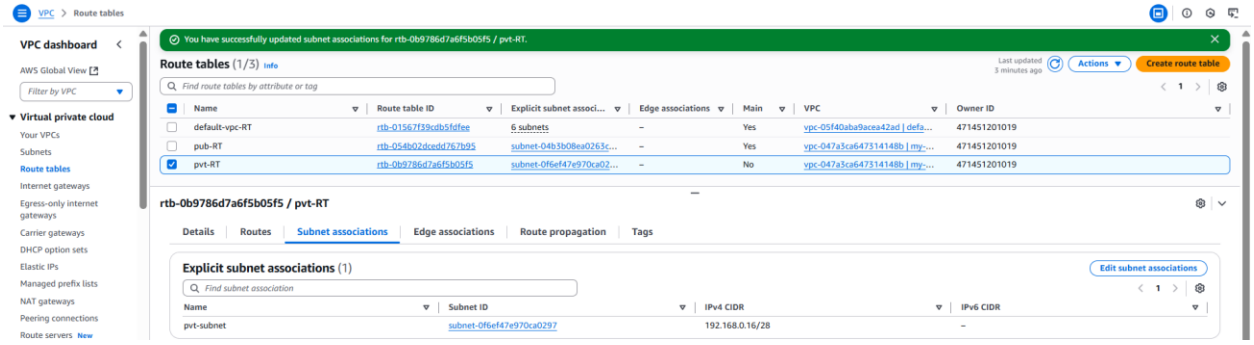
Create another subnet (for private):



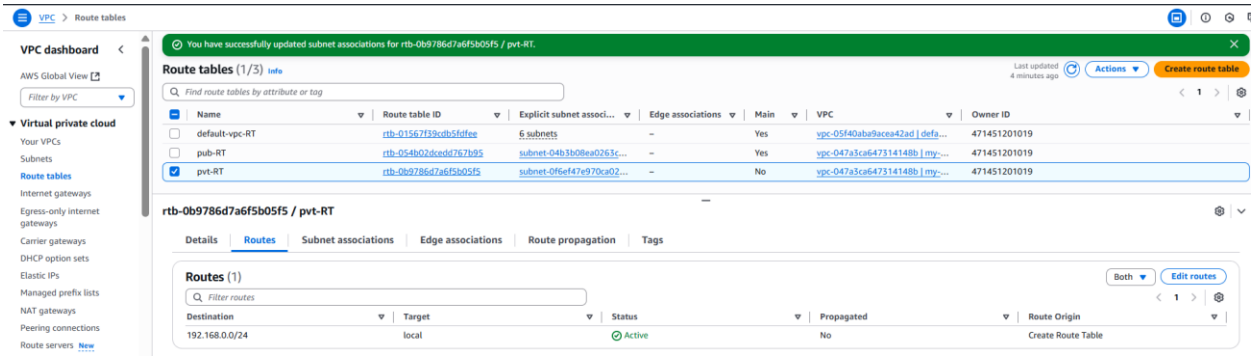
Create Route Table:



Associate Route Table with private Subnet:



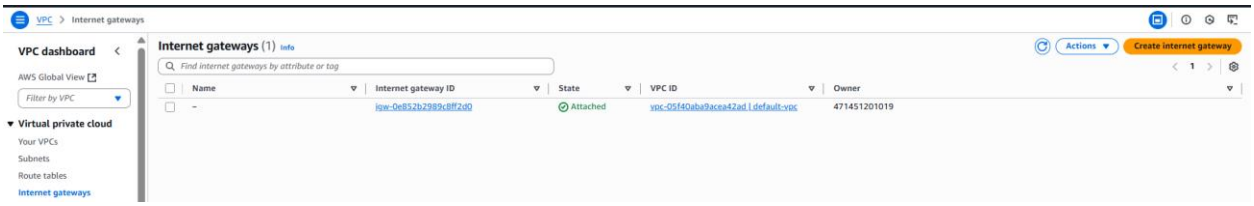
Route Table does not contain any Route to IGW so it is private Route Table:

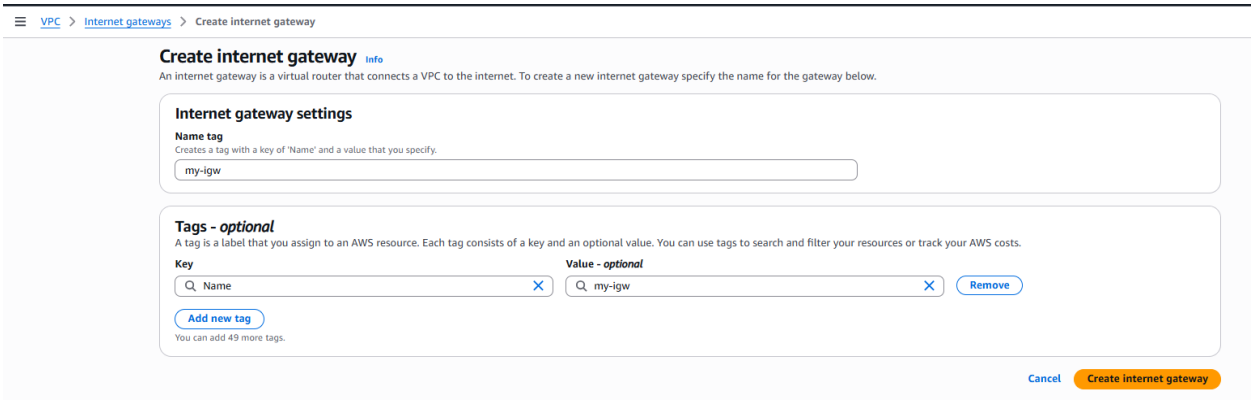


--------done-------

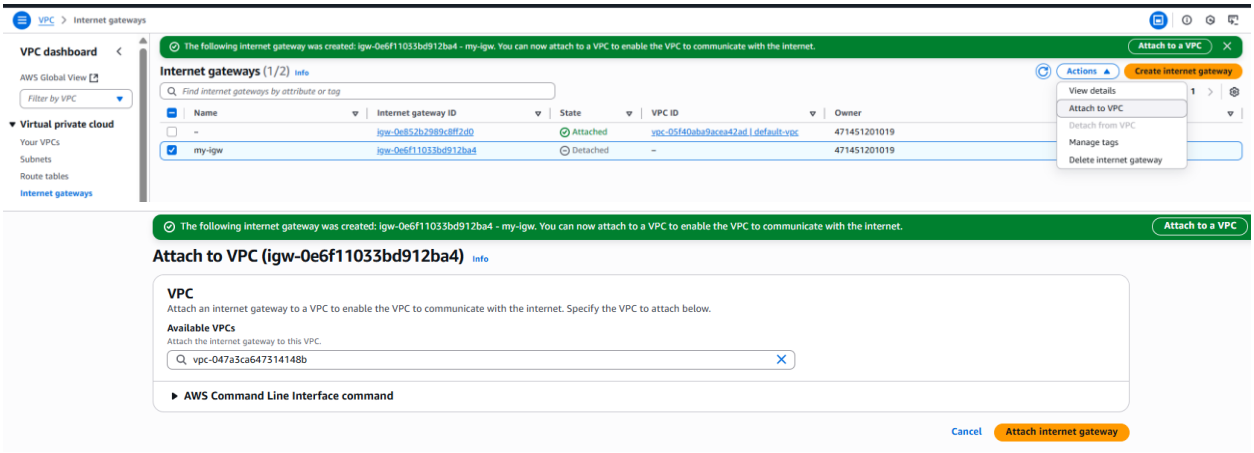
3. Attach an IGW to the VPC.

Create an Internet Gateway:





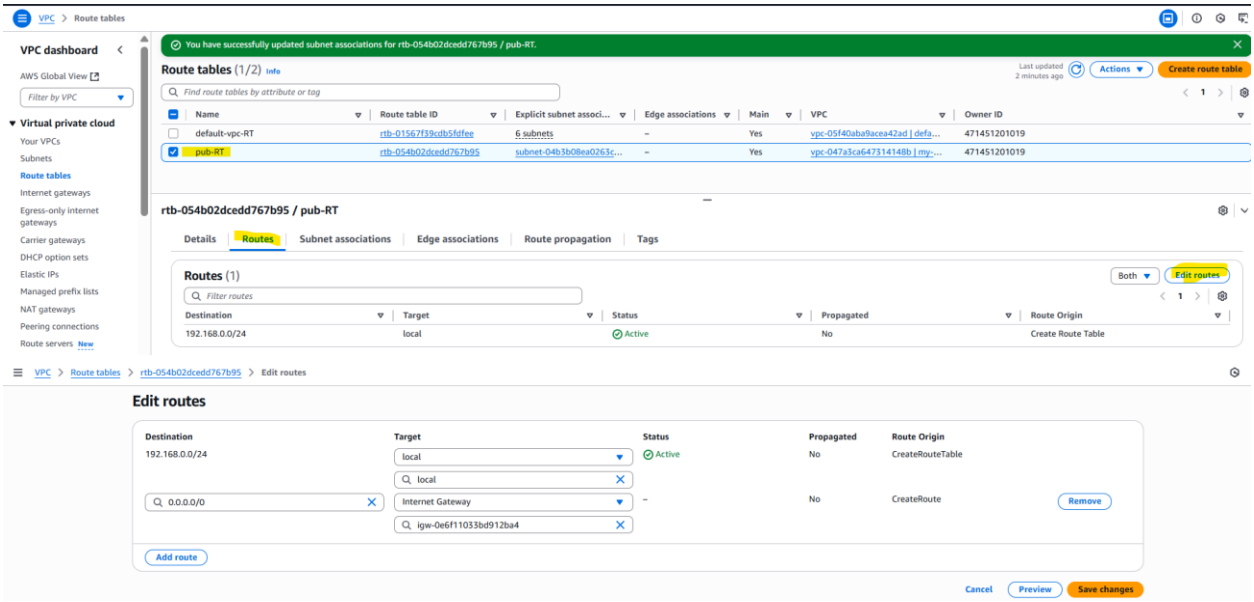
And attach to your vpc:

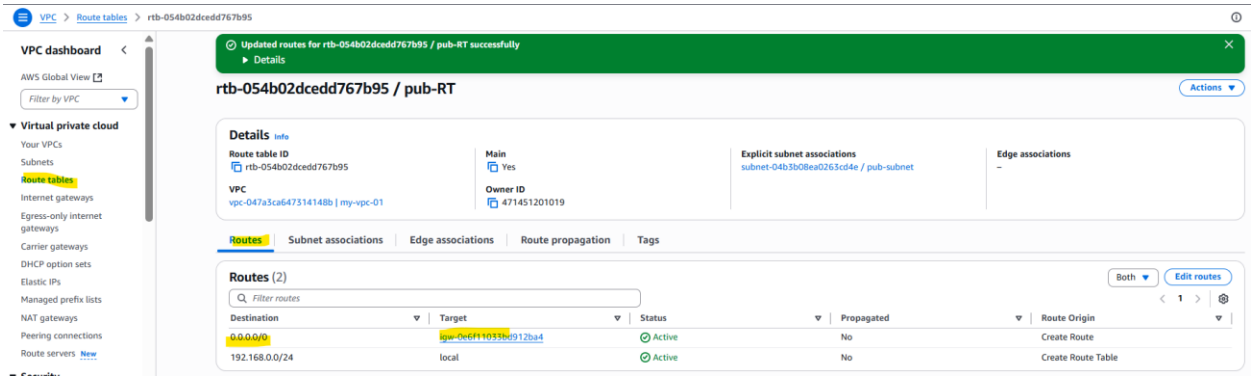


--------done-----

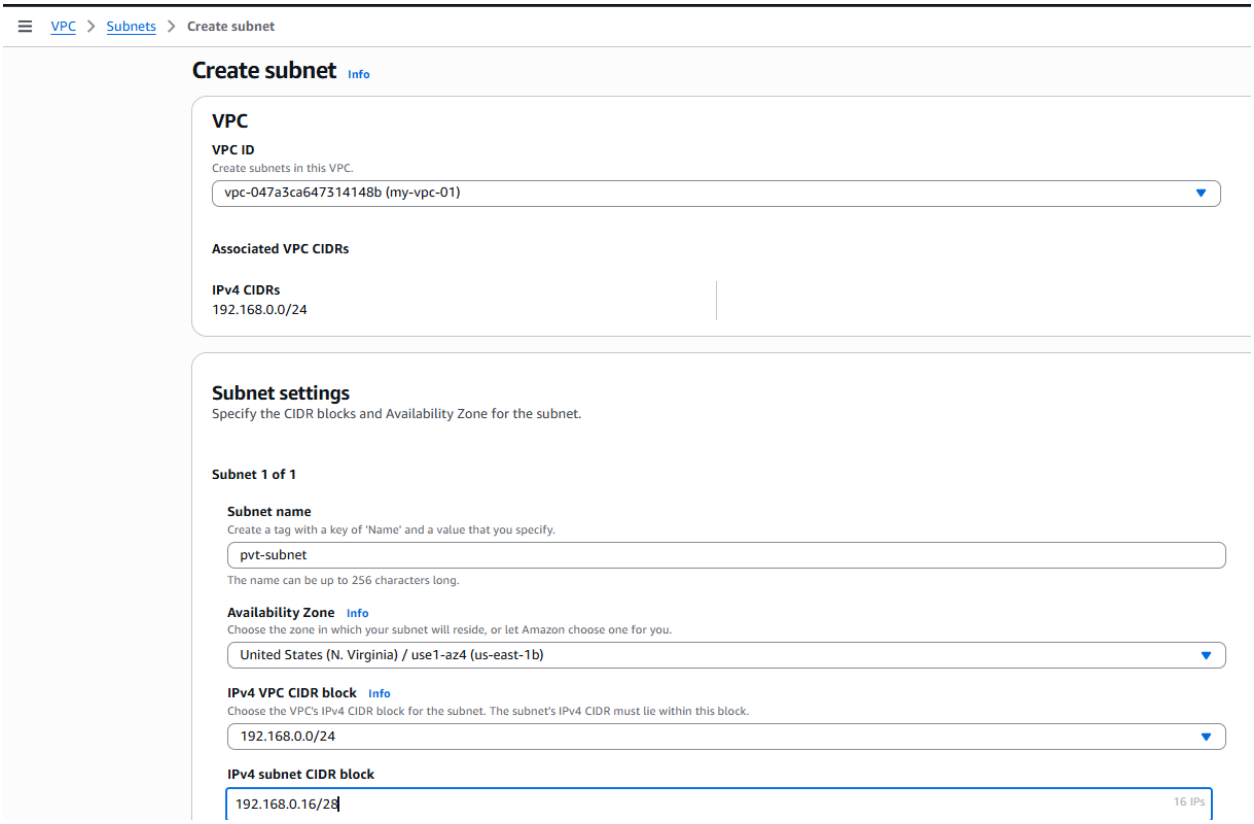
4. Create one public route table (RT) and one private route table.

Add the Route i.e., IGW to the Route Table Route:

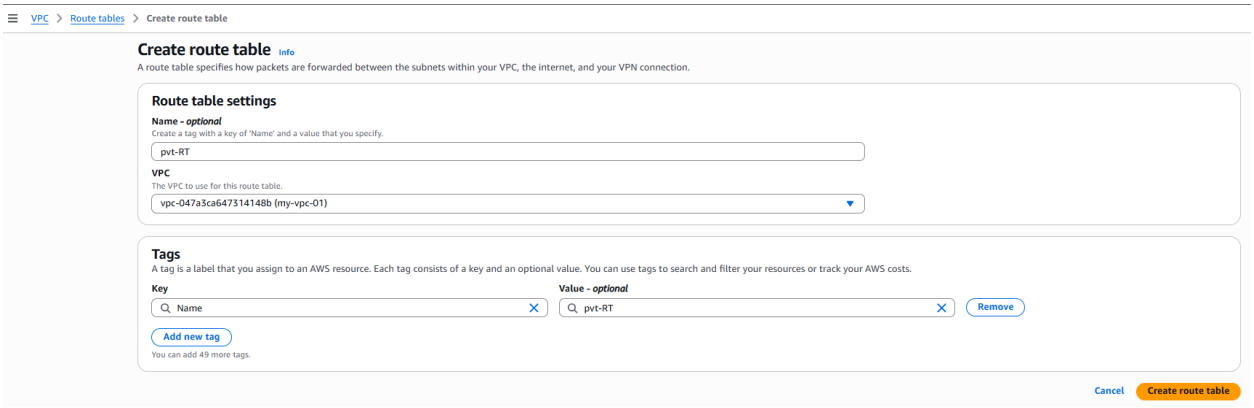




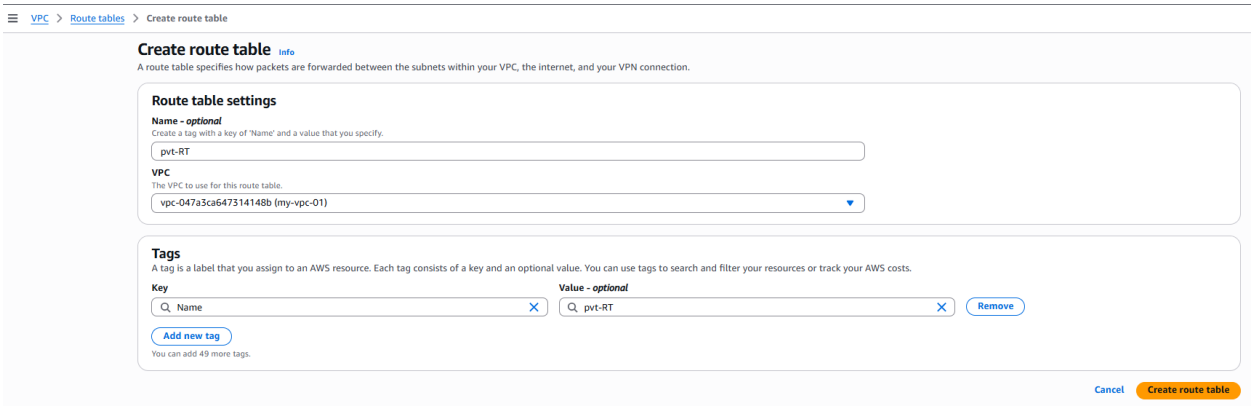
Create another subnet (for private):

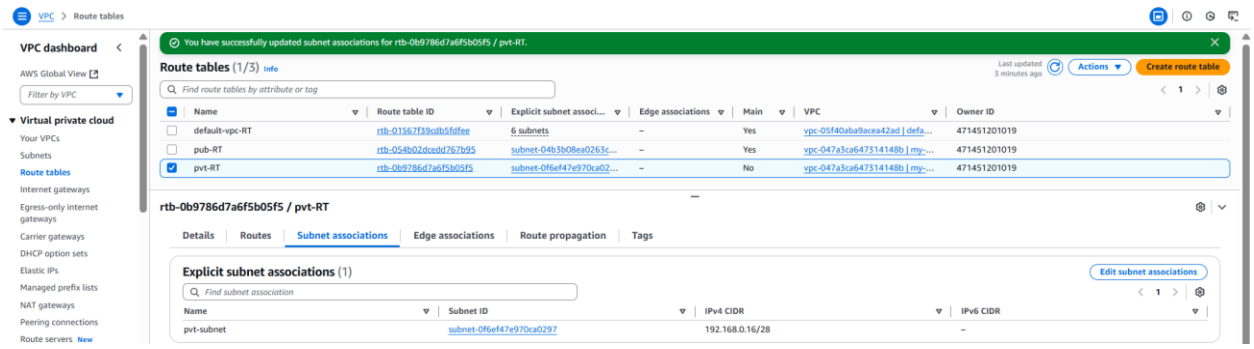


Create Route Table:

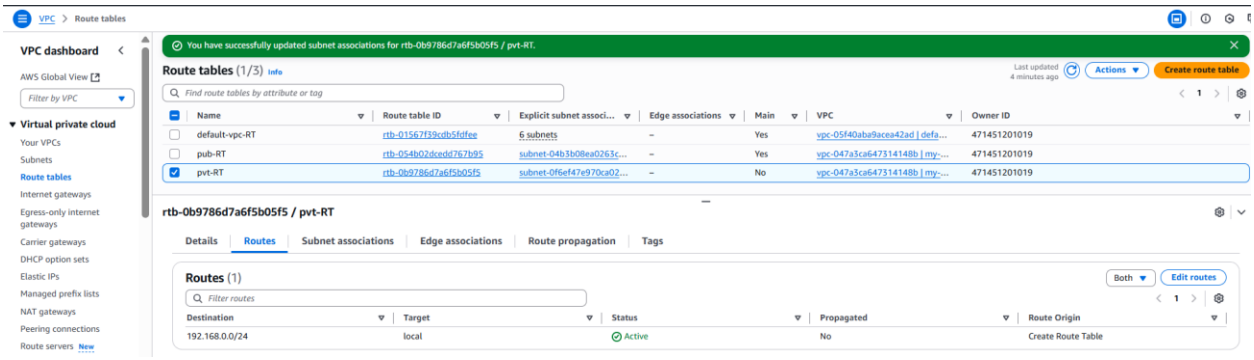


Associate Route Table with private Subnet:





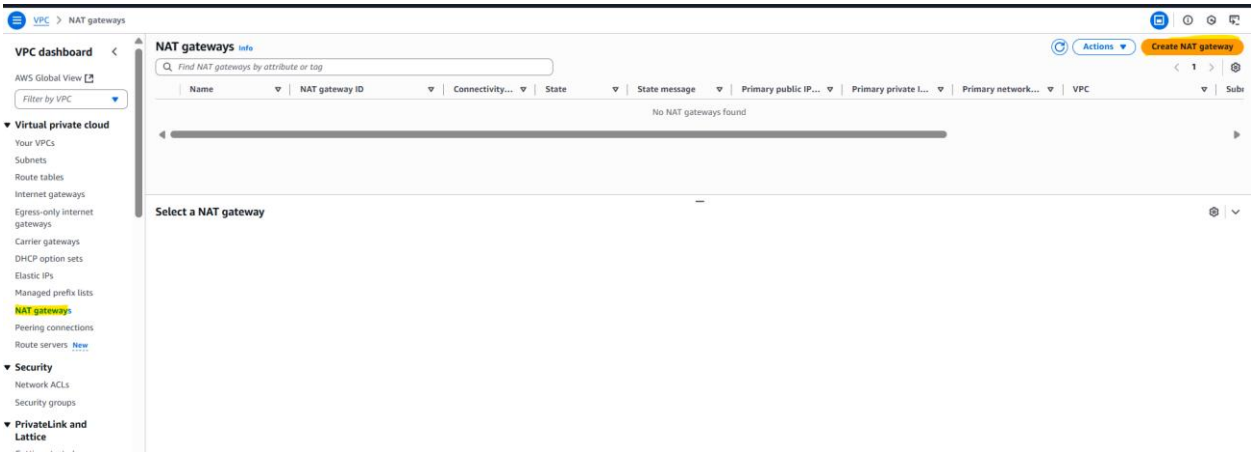
Route Table does not contain any Route to IGW so it is private Route Table:



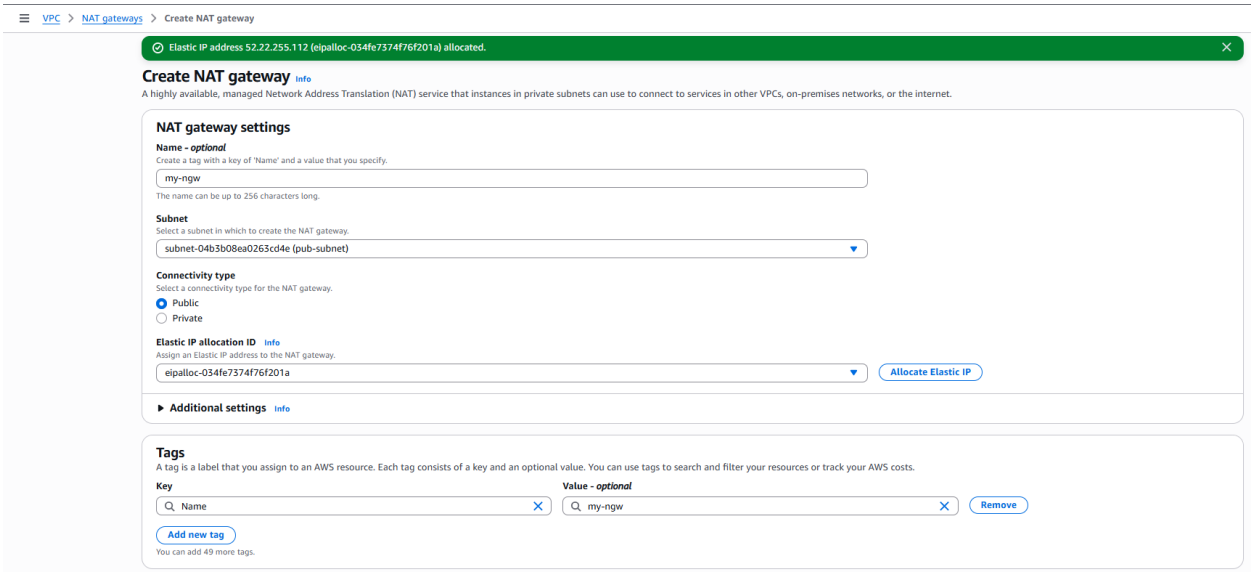
--------done-------

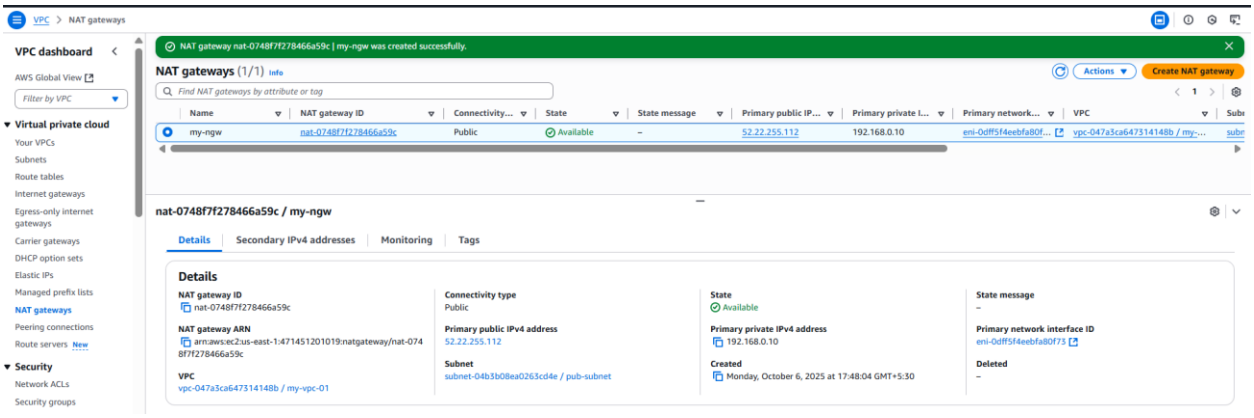
5. Deploy a NAT gateway in the public subnet and attach the NAT gateway to the private subnet.

Create NAT Gateway:

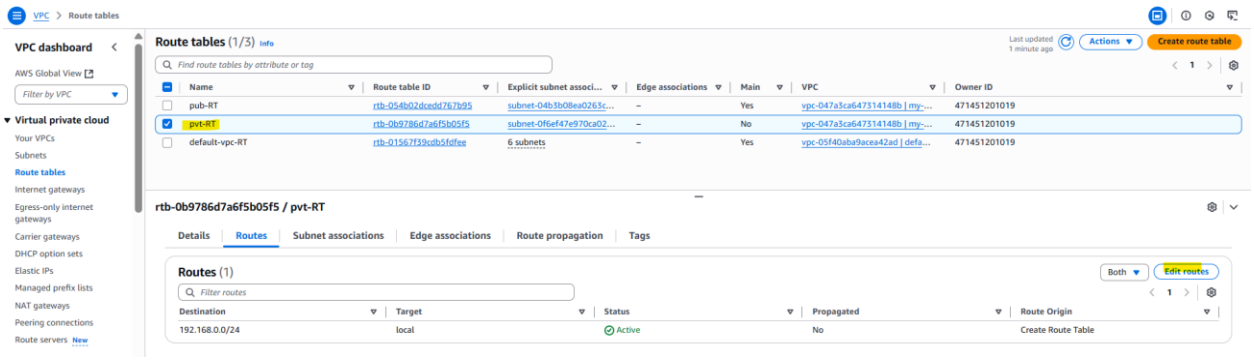


Select public subnet and allocate the elastic ip:

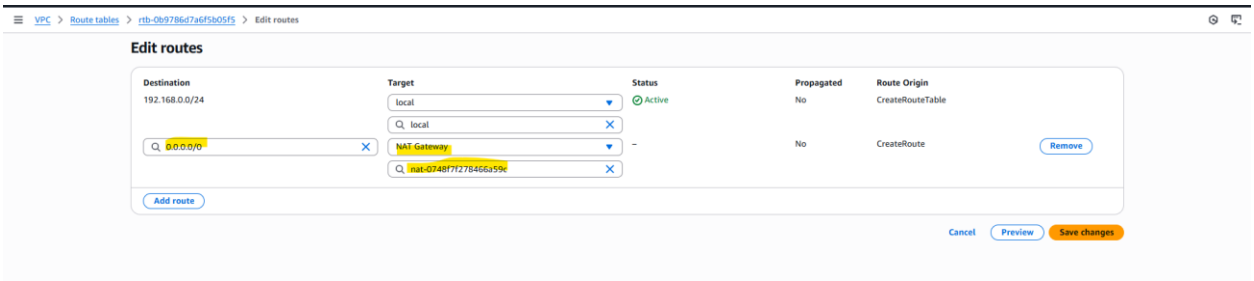


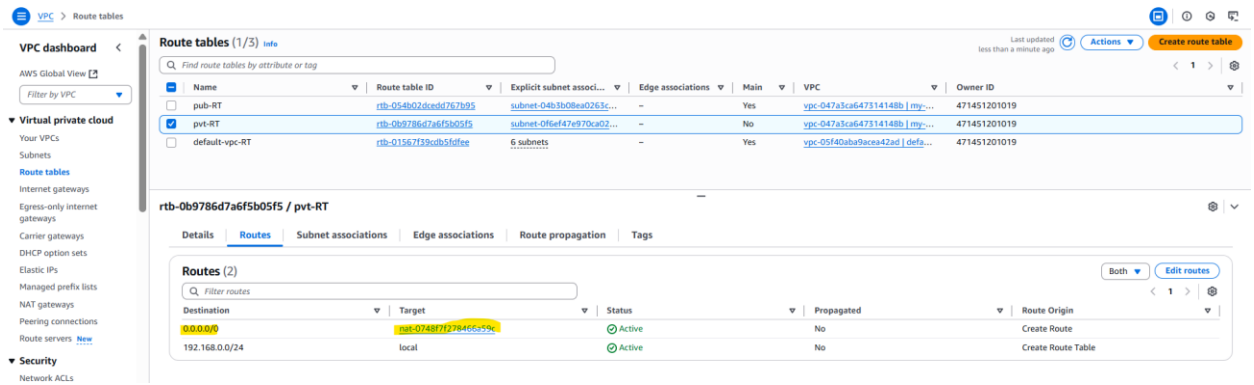


Goto Route Table, select your private Route Table and Edit Routes:



Add the NAT gateway in the Routes:

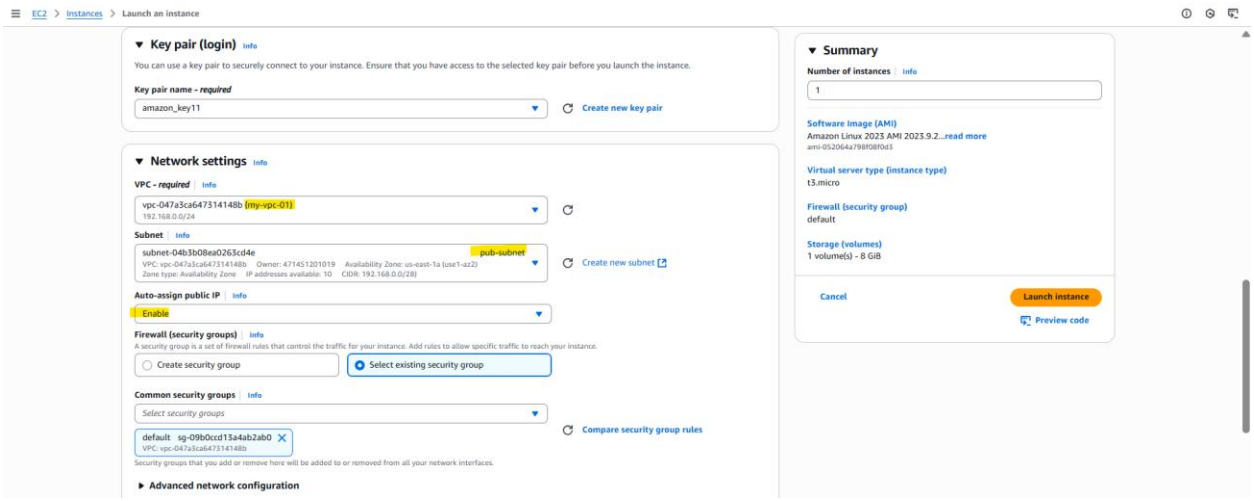




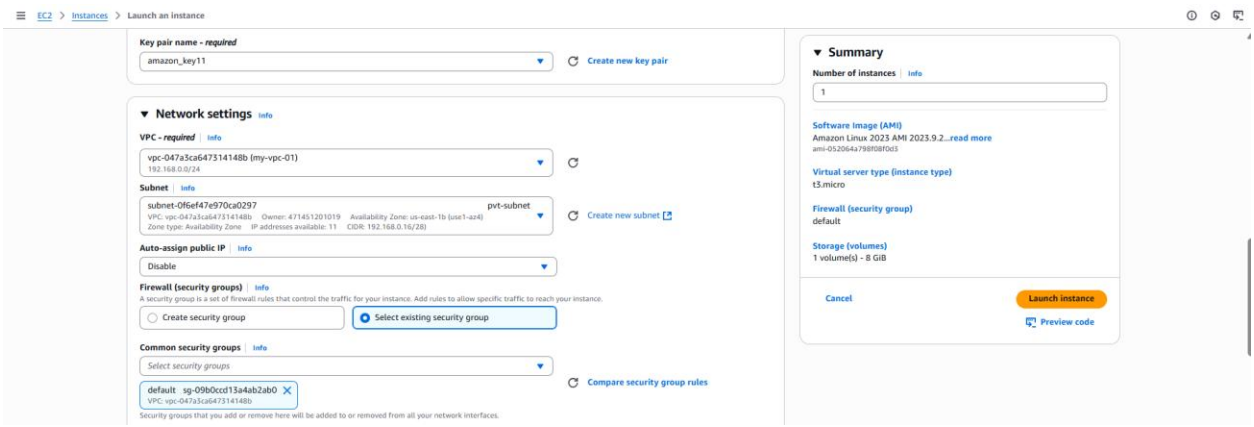
---------done-------

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

Launching the instance in public subnet, select your vpc and select your public subnet:



Launch another instance in private subnet by selecting your vpc and private subnet and 'Disable' the Auto-assign public IP:



----------done------

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

   Installing Apache on public EC2:

```
================================================================================
Installing:
 httpd                  x86_64   2.4.65-1.amzn2023.0.1       amazonlinux    47 k
Installing dependencies:
 apr                    x86_64   1.7.5-1.amzn2023.0.4        amazonlinux   129 k
 apr-util               x86_64   1.6.3-1.amzn2023.0.1        amazonlinux    98 k
 generic-logos-httpd    noarch   18.0.0-12.amzn2023.0.3      amazonlinux    19 k
 httpd-core             x86_64   2.4.65-1.amzn2023.0.1       amazonlinux   1.4 M
 httpd-filesystem       noarch   2.4.65-1.amzn2023.0.1       amazonlinux    13 k
 httpd-tools            x86_64   2.4.65-1.amzn2023.0.1       amazonlinux    81 k
 libbrotli              x86_64   1.0.9-4.amzn2023.0.2        amazonlinux   315 k
 mailcap                noarch   2.1.49-3.amzn2023.0.3       amazonlinux    33 k
Installing weak dependencies:
 apr-util-openssl       x86_64   1.6.3-1.amzn2023.0.1        amazonlinux    17 k
 mod_http2              x86_64   2.0.27-1.amzn2023.0.3       amazonlinux   166 k
 mod_lua                x86_64   2.4.65-1.amzn2023.0.1       amazonlinux    60 k

Transaction Summary
================================================================================
Install  12 Packages
```

httpd enable and start:

```
[ec2-user@ip-192-168-0-7 ~]$ vim /var/www/html/index.html
[ec2-user@ip-192-168-0-7 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr
/lib/systemd/system/httpd.service.
[ec2-user@ip-192-168-0-7 ~]$ sudo systemctl start httpd
```

Deploy a sample index.html in /var/www/html/index.html:

```
[ec2-user@ip-192-168-0-7 ~]$ cat /var/www/html/index.html
<h1>Techie Horizon</h1>
```

And re-start httpd:

```
[ec2-user@ip-192-168-0-7 ~]$ sudo systemctl restart httpd
```

Then login to private Ec2 from Bastion host and install httpd:

```
[ec2-user@ip-192-168-0-23 ~]$ sudo yum install httpd
Amazon Linux 2023 Kernel Livepatch repository   226 kB/s |  26 kB     00:00
Dependencies resolved.
================================================================================
 Package                Arch     Version                        Repository     Size
================================================================================
Installing:
 httpd                  x86_64   2.4.65-1.amzn2023.0.1          amazonlinux    47 k
Installing dependencies:
 apr                    x86_64   1.7.5-1.amzn2023.0.4           amazonlinux   129 k
 apr-util               x86_64   1.6.3-1.amzn2023.0.1           amazonlinux    98 k
 generic-logos-httpd    noarch   18.0.0-12.amzn2023.0.3         amazonlinux    19 k
 httpd-core             x86_64   2.4.65-1.amzn2023.0.1          amazonlinux   1.4 M
 httpd-filesystem       noarch   2.4.65-1.amzn2023.0.1          amazonlinux    13 k
 httpd-tools            x86_64   2.4.65-1.amzn2023.0.1          amazonlinux    81 k
 libbrotli              x86_64   1.0.9-4.amzn2023.0.2           amazonlinux   315 k
 mailcap                noarch   2.1.49-3.amzn2023.0.3          amazonlinux    33 k
Installing weak dependencies:
 apr-util-openssl       x86_64   1.6.3-1.amzn2023.0.1           amazonlinux    17 k
 mod_http2              x86_64   2.0.27-1.amzn2023.0.3          amazonlinux   166 k
 mod_lua                x86_64   2.4.65-1.amzn2023.0.1          amazonlinux    60 k

Transaction Summary
```

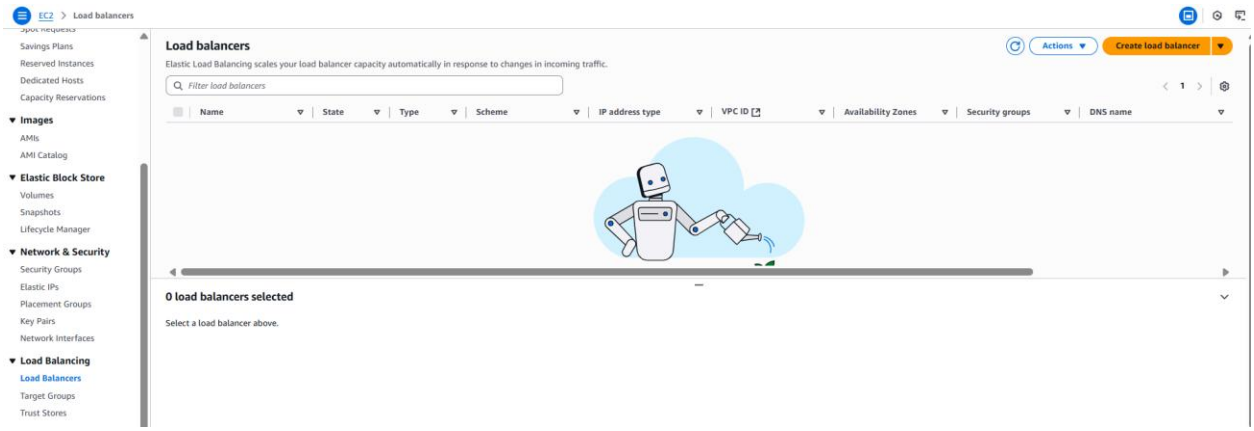
And enable, start the httpd service and deploy sample index.html and restart the httpd:

```
[ec2-user@ip-192-168-0-23 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr
/lib/systemd/system/httpd.service.
[ec2-user@ip-192-168-0-23 ~]$ sudo systemctl start httpd
[ec2-user@ip-192-168-0-23 ~]$ vim /var/www/html/index.html
[ec2-user@ip-192-168-0-23 ~]$ sudo vim /var/www/html/index.html
[ec2-user@ip-192-168-0-23 ~]$ sudo systemctl restart httpd
[ec2-user@ip-192-168-0-23 ~]$ sudo curl localhost:80
<h1>Techie Horizon from Privat Subnet</h1>
```

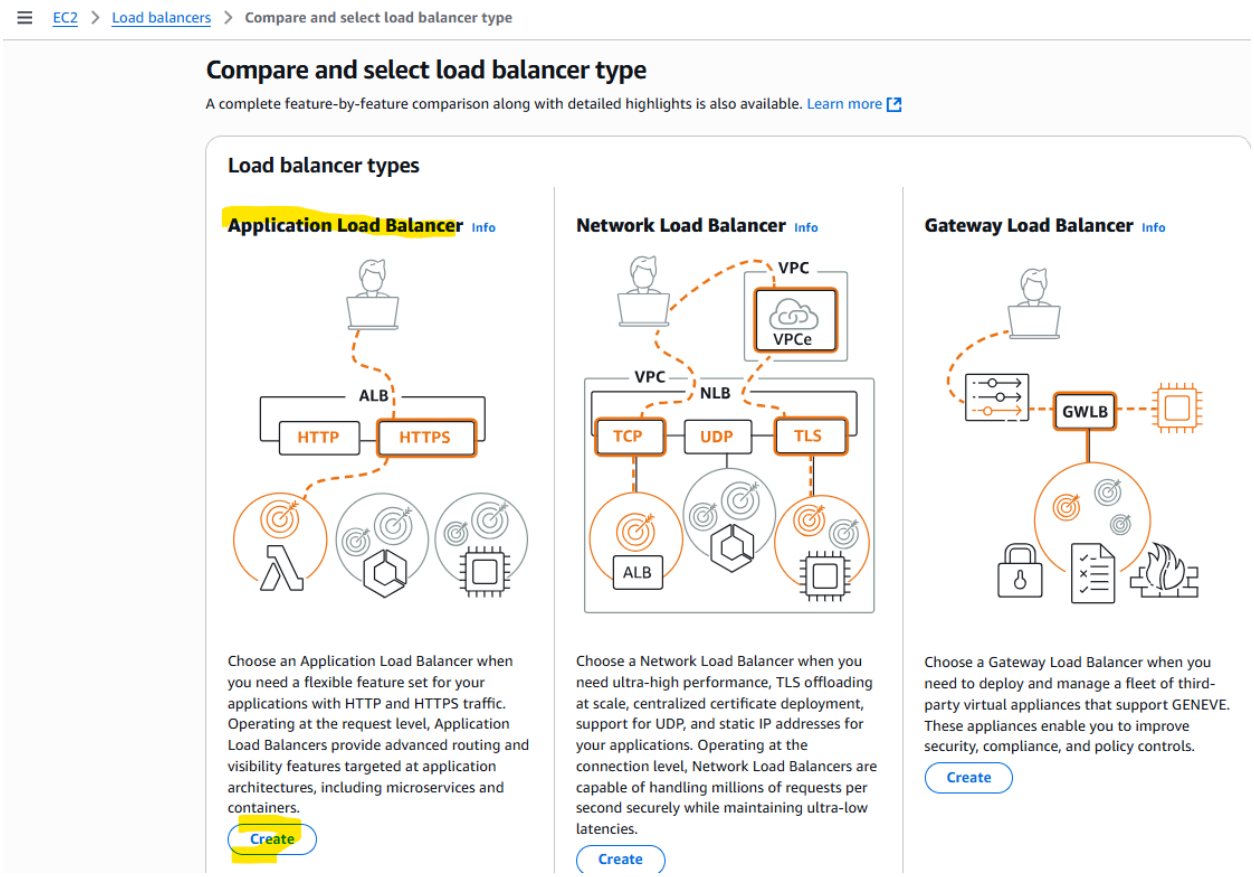
--------done----------

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

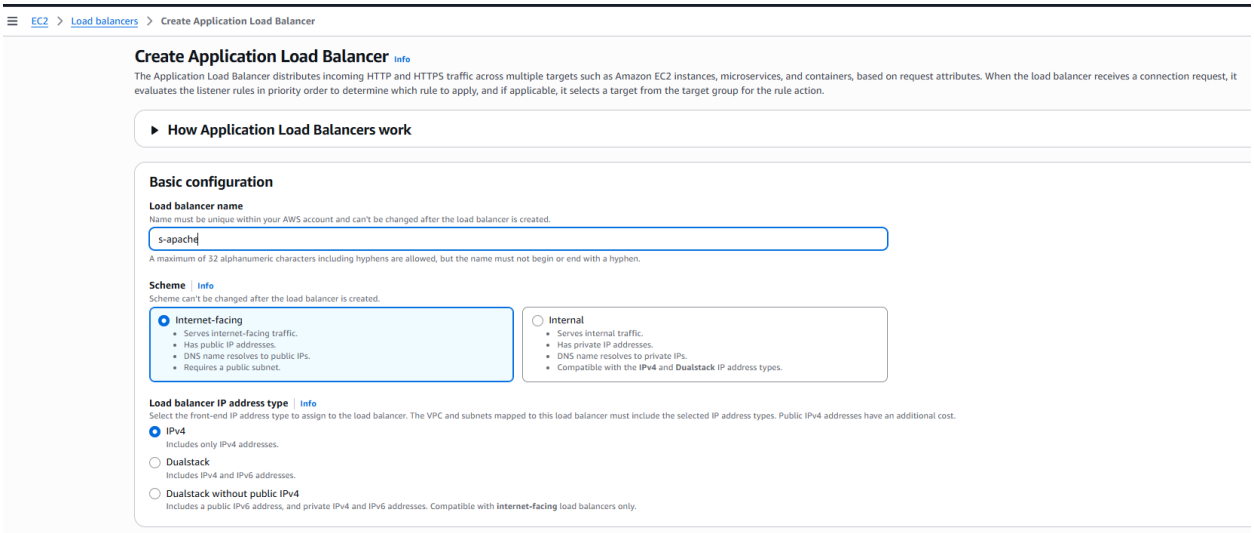
Create Load Balancer:



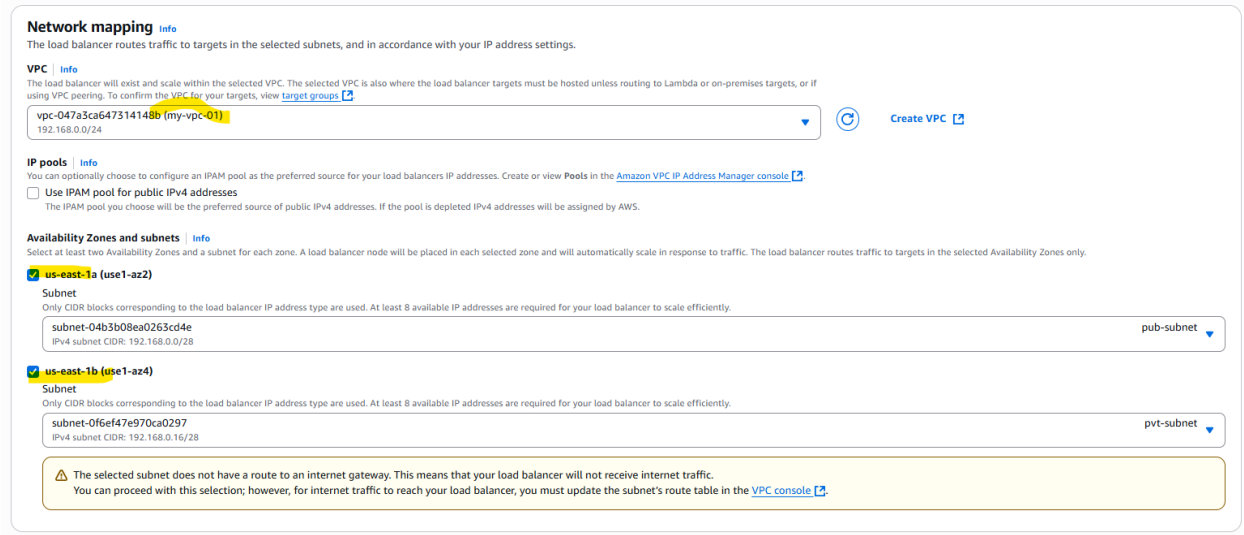
Create Application load Balancer:



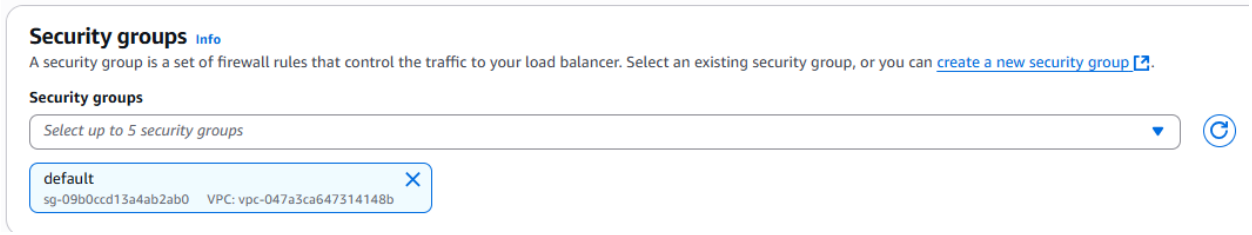
Give the name and Schem as Internet facing:



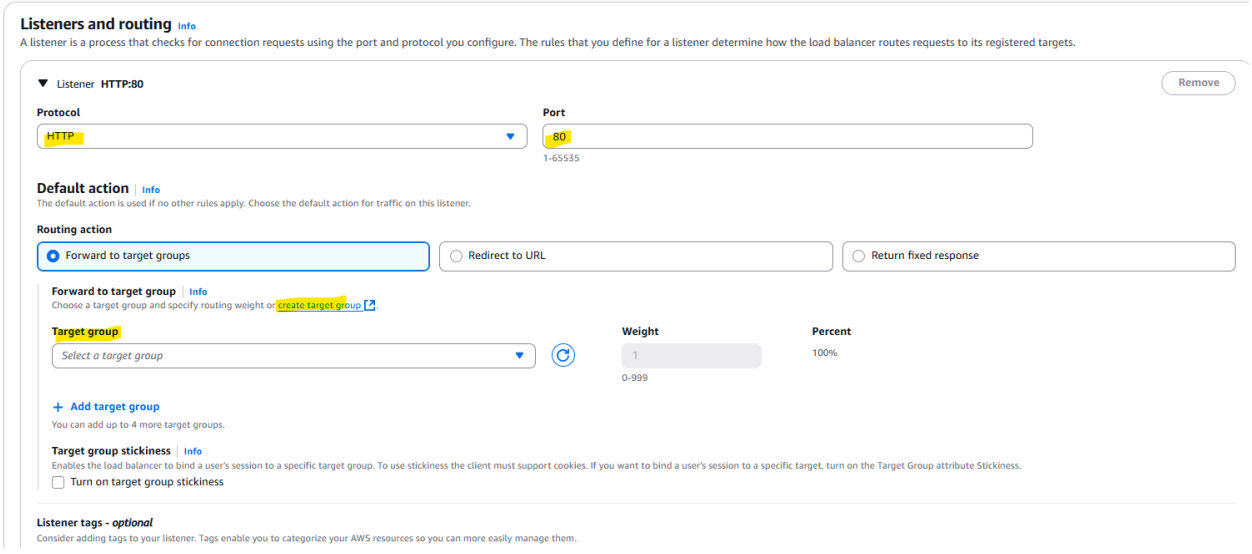
Select your vpc and at least 2 Availability zones:



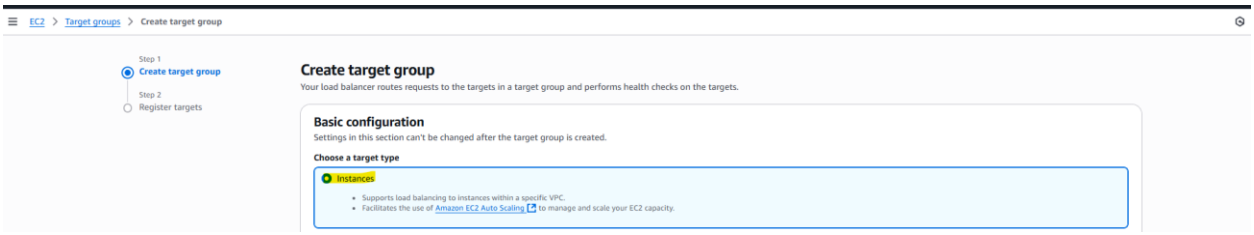
Give the Security Group:



Select protocol and port and click on Create Target group:



Select Instances:



And give Target Group name, Protocol Port , IPV4 and your VPC:

Target group name

apache-tg

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol
Protocol for load balancer-to-target communication. Can't be modified after creation.

HTTP

Port
Port number where targets receive traffic. Can be overridden for individual targets during registration.

80

1-65535

IP address type
Only targets with the indicated IP address type can be registered to this target group.

- (•) IPv4
  Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.
- ( ) IPv6
  Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). Learn more

VPC
Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

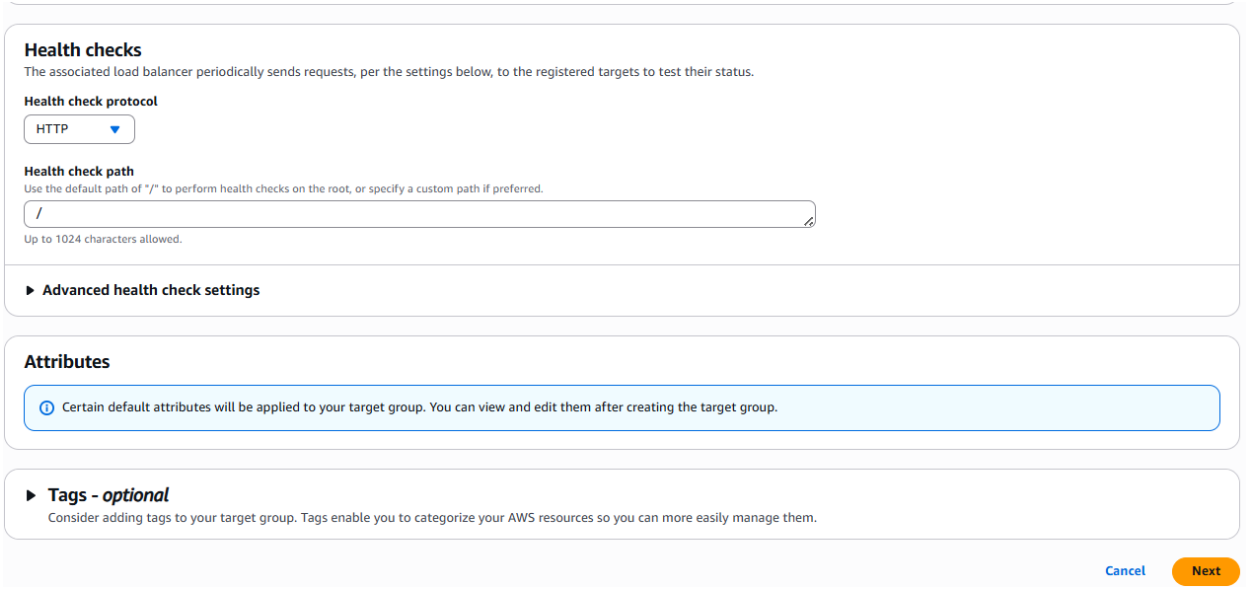
vpc-047a3ca647314148b (my-vpc-01)
192.168.0.0/24

Create VPC

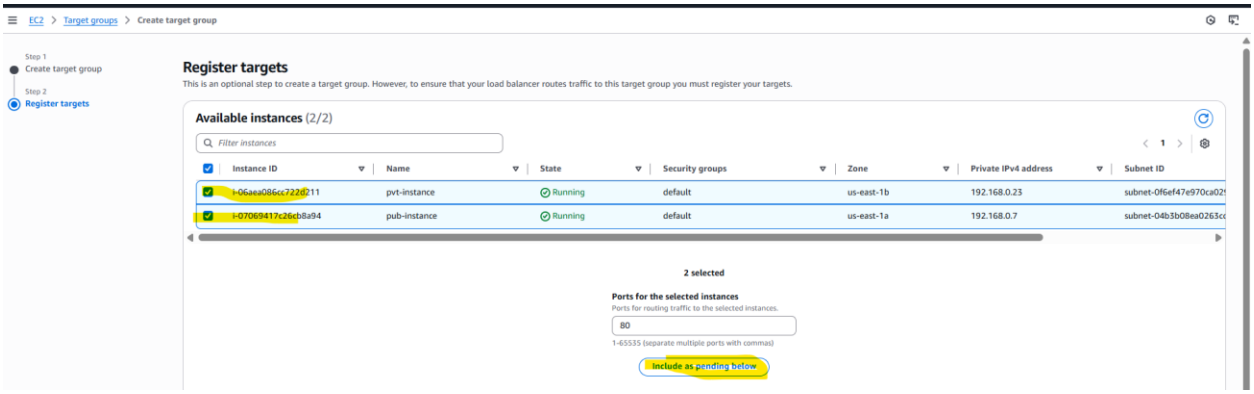
Protocol version

- (•) HTTP1
  Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
- ( ) HTTP2
  Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
- ( ) gRPC
  Send requests to targets using gRPC. Supported when the request protocol is gRPC.

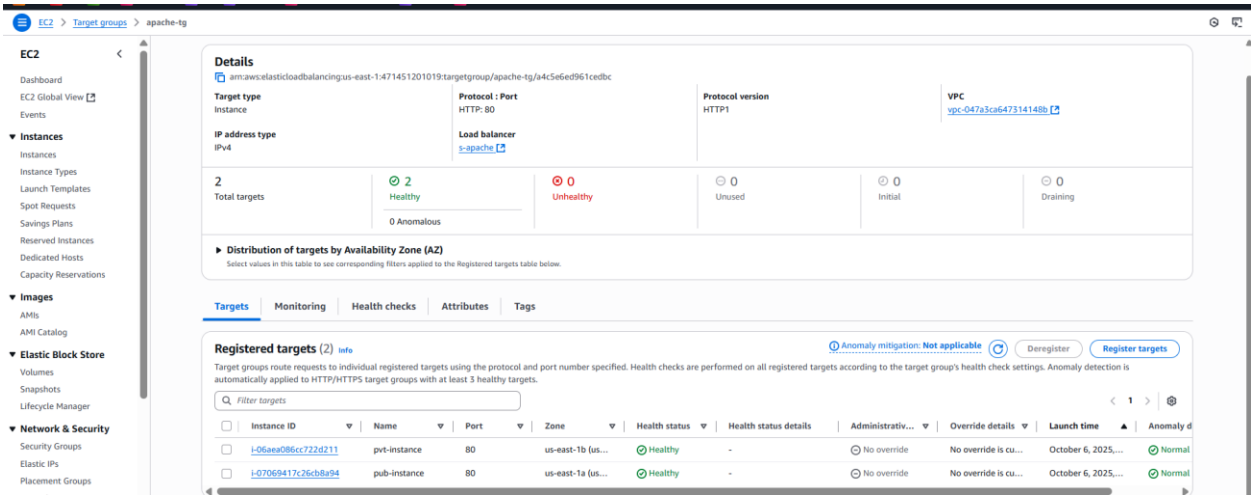
And click on Next,



Select your instances and 'Include as pending below':



Then both the instances will get added to the Target group:



----------done----------

9. Store application load balancer logs in S3.

   Storing **Application Load Balancer (ALB)** access logs in **Amazon S3** is a great way to monitor traffic, troubleshoot issues, and analyze performance.

   What You'll Need

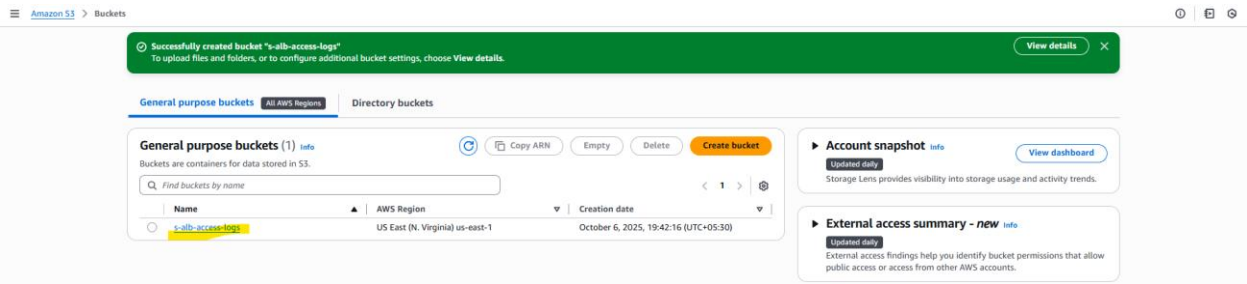
- An **Application Load Balancer** already created.
- An **S3 bucket** (or permission to create one).
- IAM permissions to modify the ALB and S3 settings.

  Create or Use an S3 Bucket

  You need an S3 bucket where the ALB can store logs.

1. Go to **Amazon S3 Console**.
2. Click **"Create bucket"**.
3. Name the bucket (e.g., alb-access-logs-myapp).
4. Choose a region — ideally, the **same region** as your ALB.
5. In **Bucket settings for Block Public Access**, leave all options checked (recommended).
6. Click **Create bucket**.

   Note: ALB logs are stored in the format: AWSLogs/<account-id>/elasticloadbalancing/<region>/<year>/<month>/<day>/...



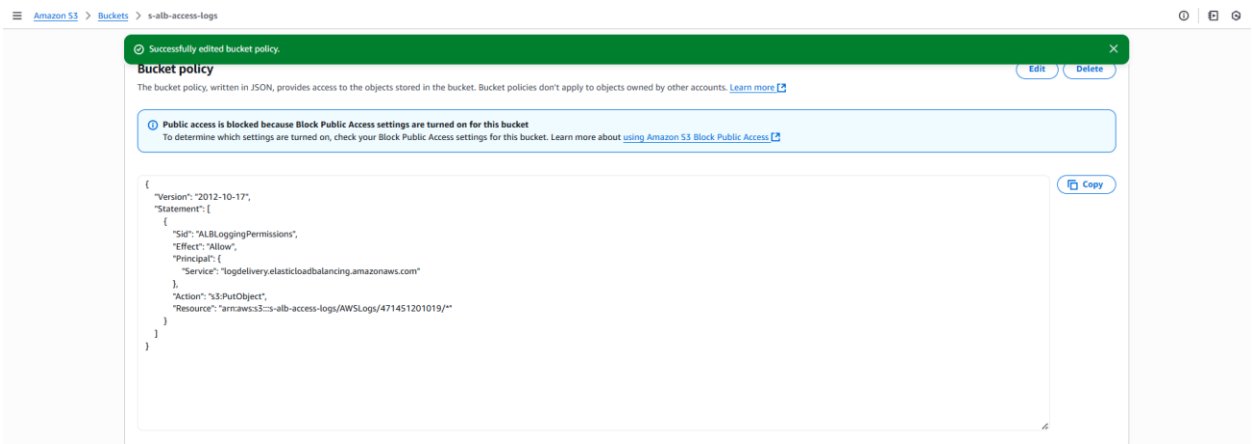
2: Allow ALB to Write Logs to S3

The ALB writes logs to your bucket using the **AWS Log Delivery** service principal. You need to add a **bucket policy** to allow this.

1. Go to your newly created S3 bucket.
2. Click **Permissions > Bucket policy**.

3. Add the following policy (replace placeholders with your values):

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "ALBLoggingPermissions",
   "Effect": "Allow",
   "Principal": {
    "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::alb-access-logs-myapp/AWSLogs/<account-id>/*"
  }
 ]
}
```



Note: This allows the ALB to write log files to your bucket.

3.Enable Access Logging on the ALB:

Go to the **EC2 Console > Load Balancers**.

Select your **Application Load Balancer**.

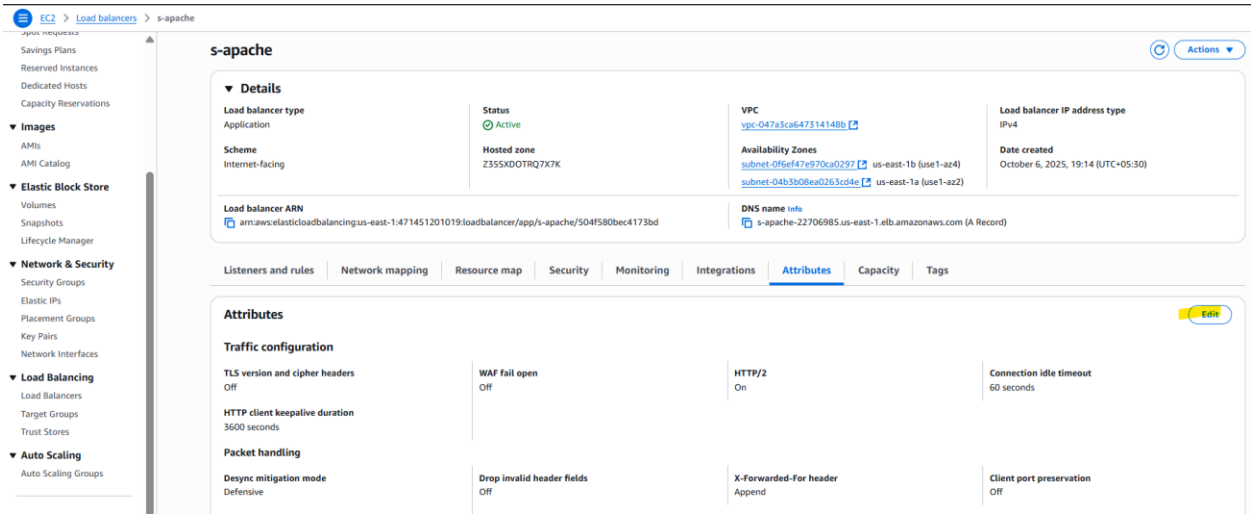
Click the **"Description"** tab.

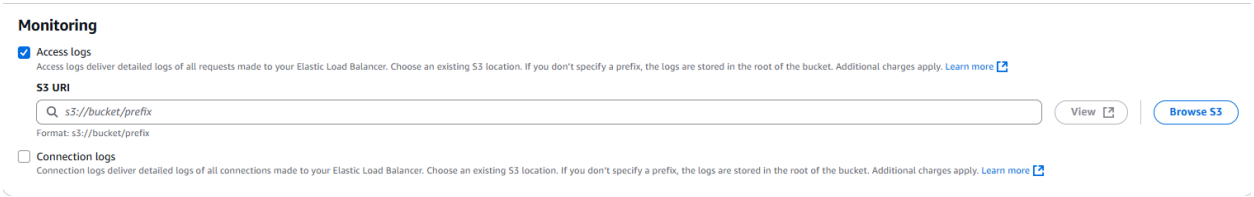
Choose **"Edit attributes"**.

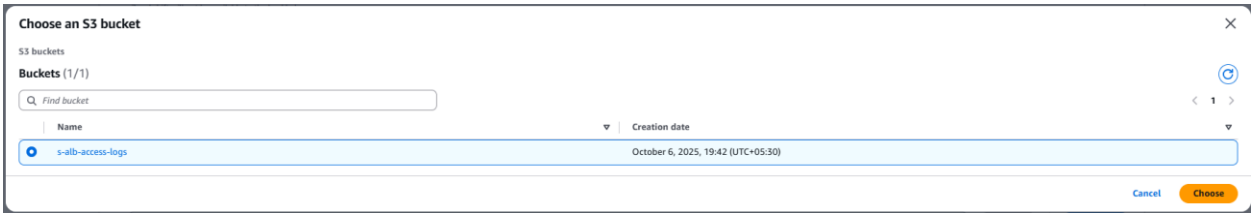
Under **Access logs**:

- **Check** the box: Enable access logs.
- **S3 location**: Enter your bucket name (e.g., alb-access-logs-myapp)
- **Prefix (optional)**: Add a folder name, like logs/ to organize files better.

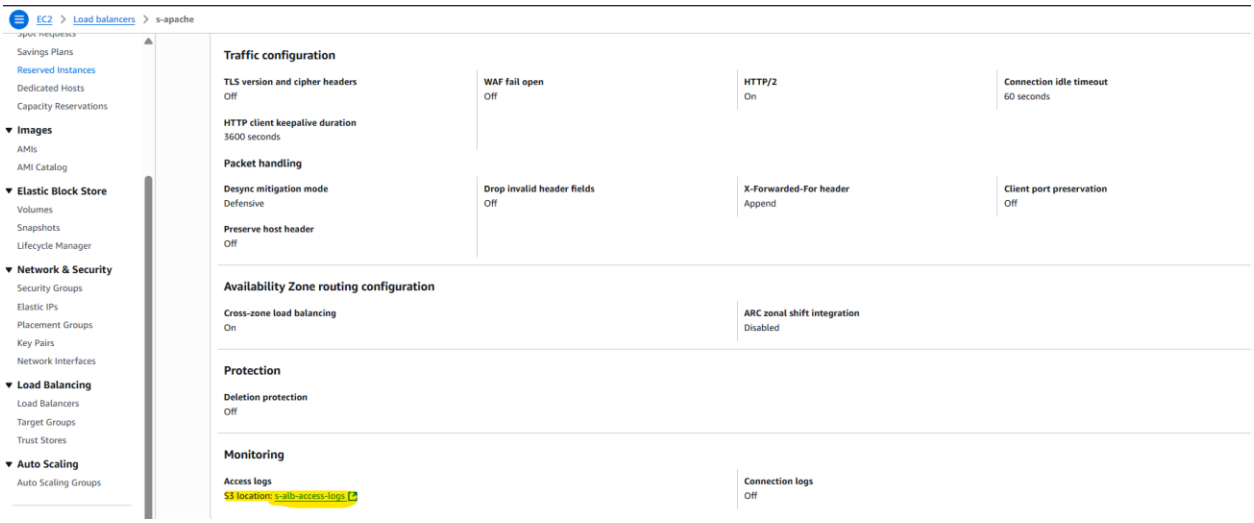
Click **Save**.





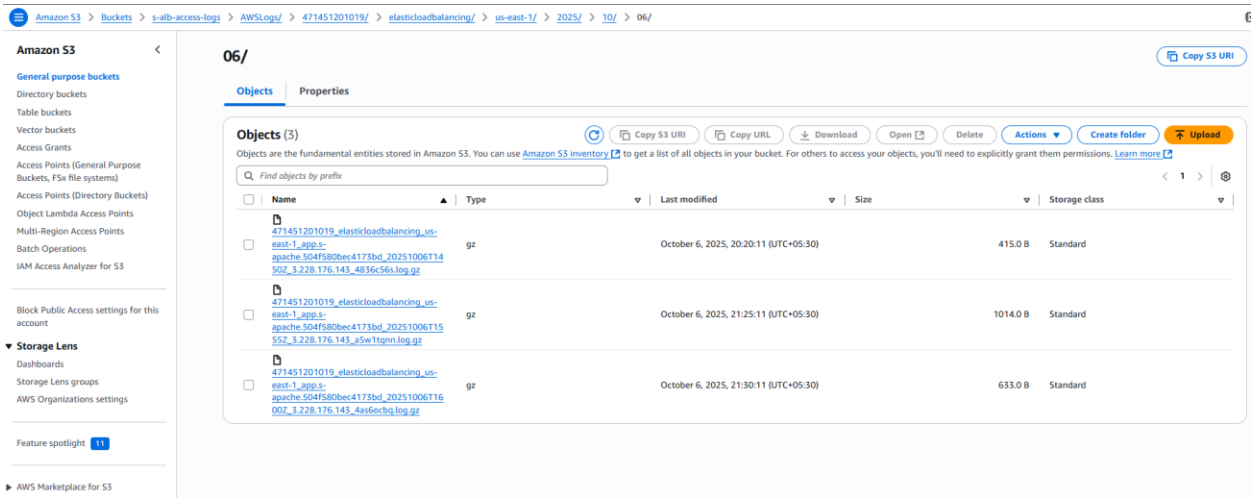


And save:



Your ALB is now configured to log requests into the S3 bucket.

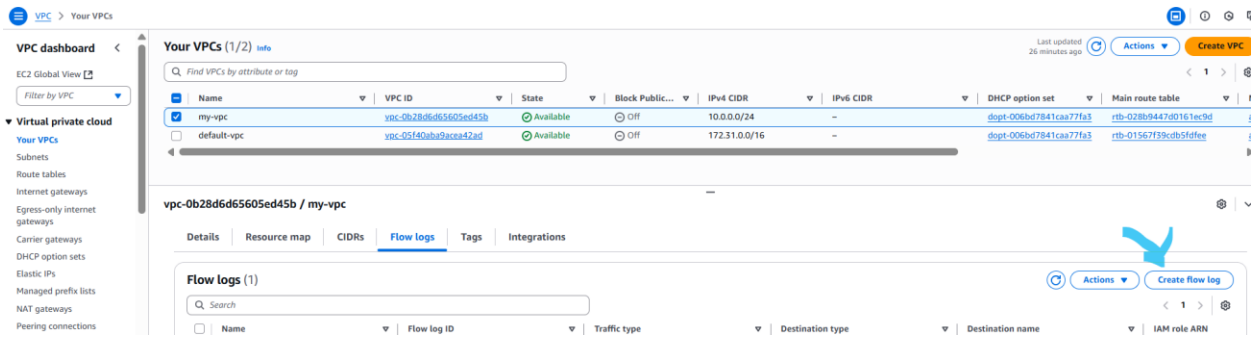
Check the logs in s3:



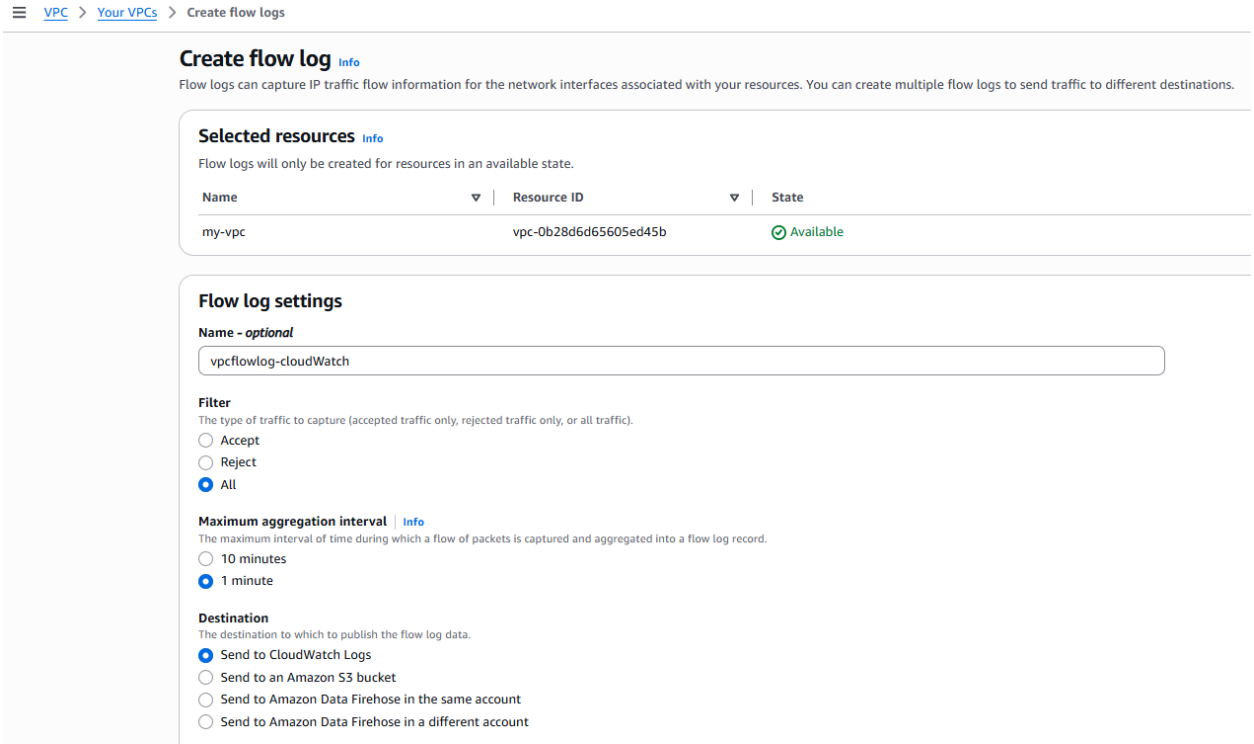
----------done---------

10. Store the VPC flow logs in a CloudWatch log group.

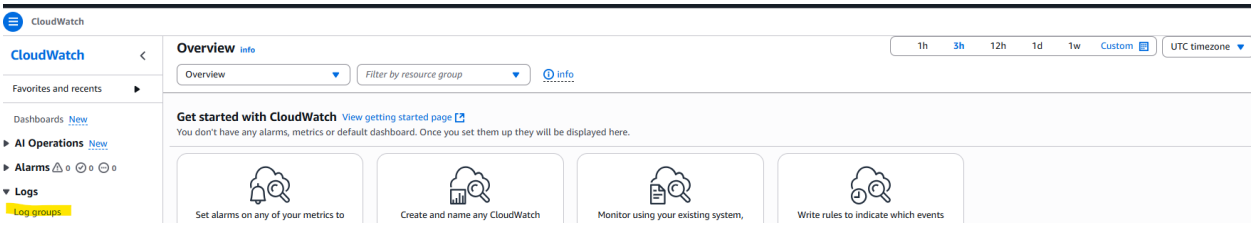
Select your required vpc for which you want to create flow logs and click on create flow log:



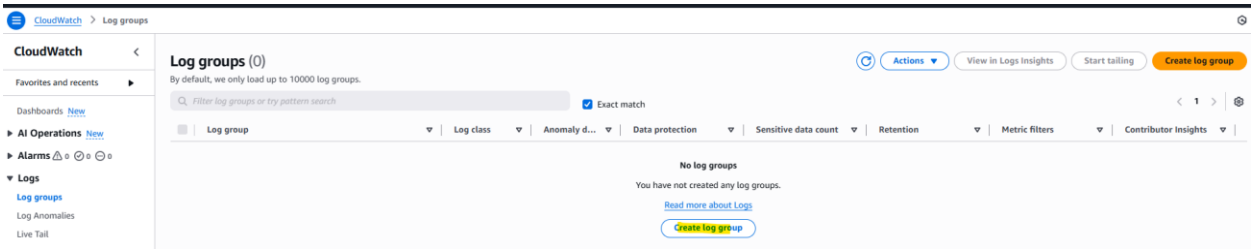
Give the name , filter, maxm aggregation interval and Destination:



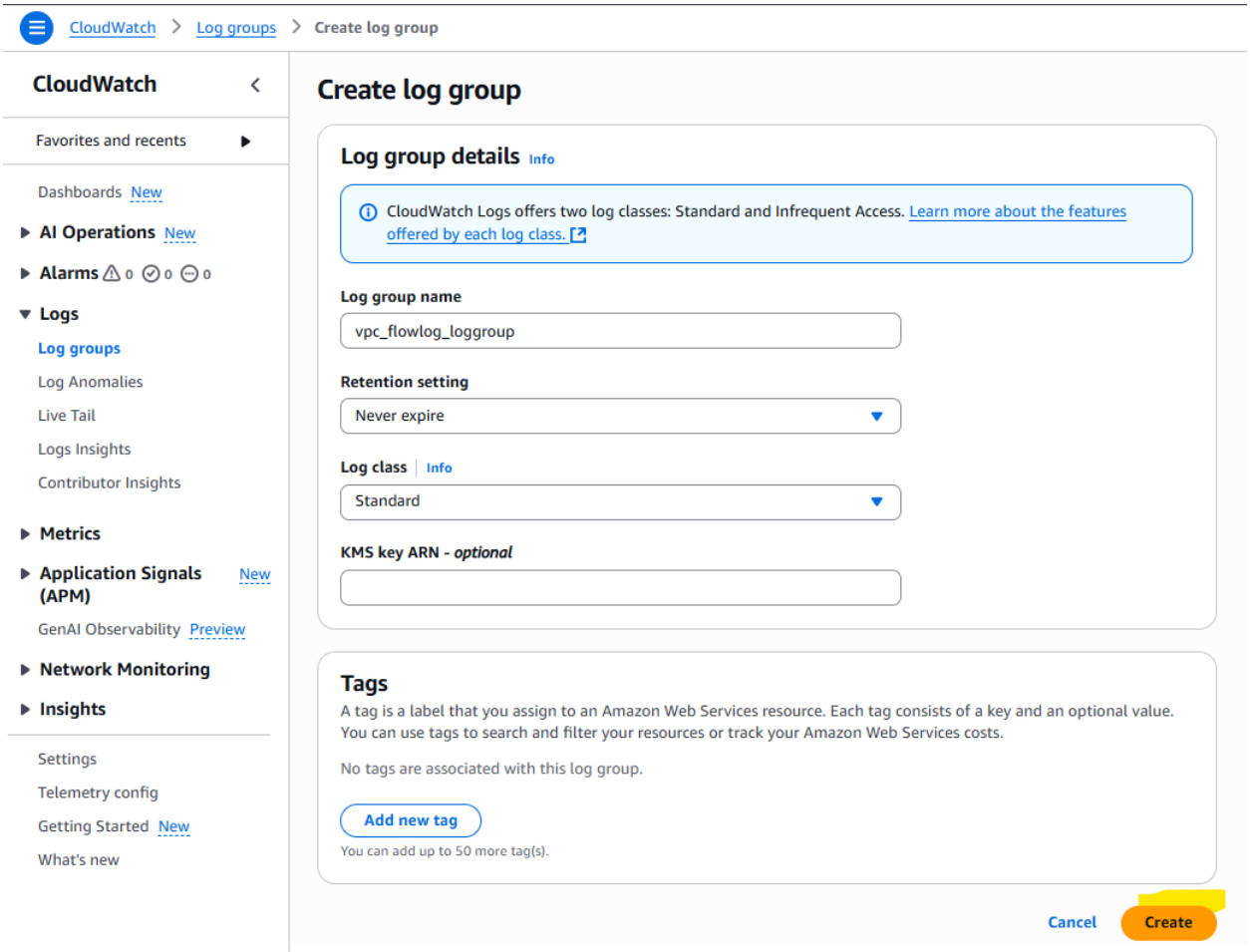
As we are storing in the cloudwatch log group so first we have to create the Log groupin CloudWatch, goto CloudWatch and select log group:



Create log group:



Provide the details and create log group:



Then create the IAM role with policy for vpc flow log to store logs in CloudWatch:

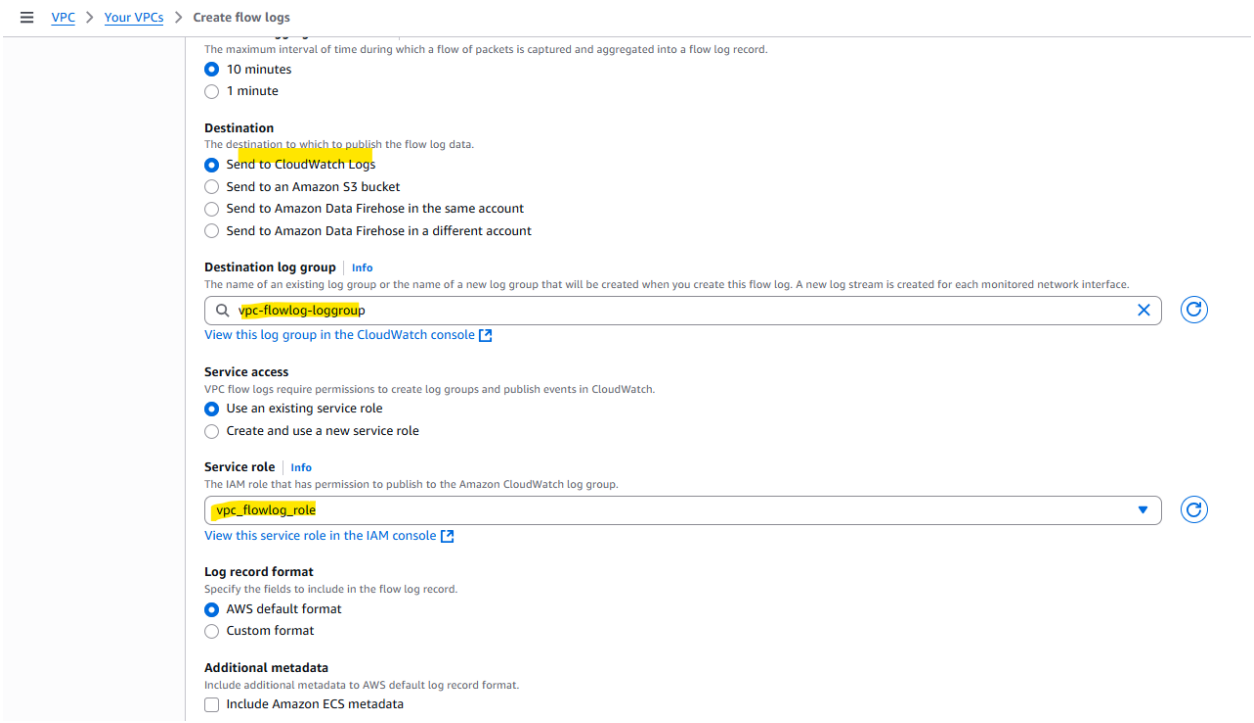
IAM Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

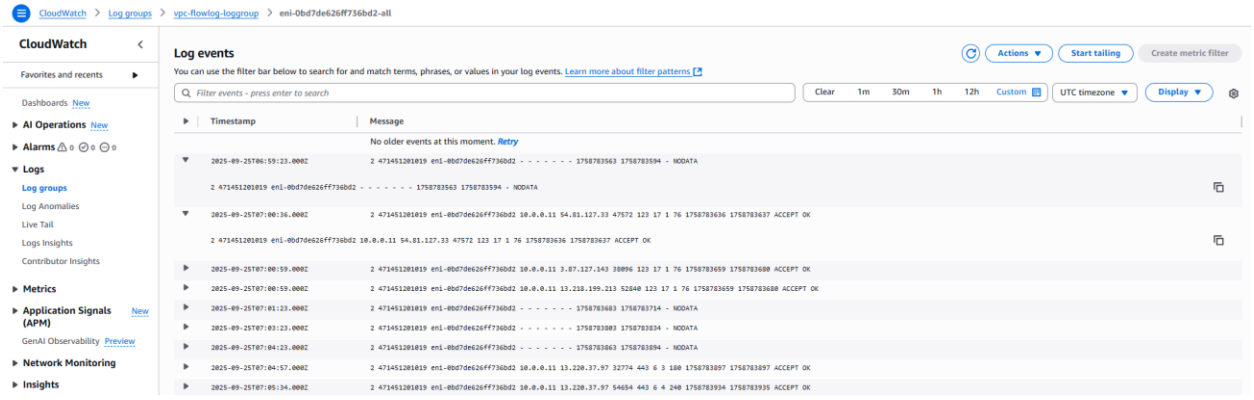
And create the role with Trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "vpc-flow-logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

And assign the above created policy to this role and select this role while creating vpc flow log:



And the logs will get saved in the cloudWatch log group:

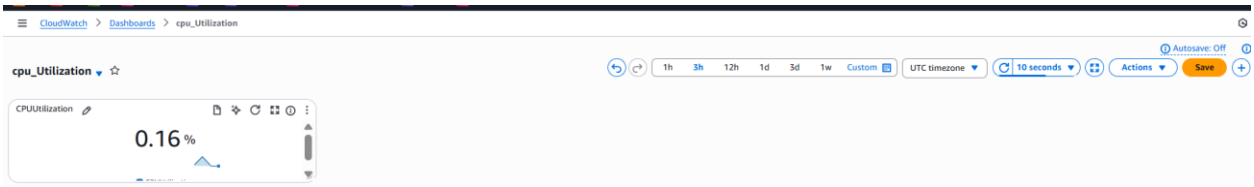


----------done----------

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

Create ec2 instance

Create Dashboard in CloudWatch and provide the instance id and select pre-instances and select the cpuUtilization:



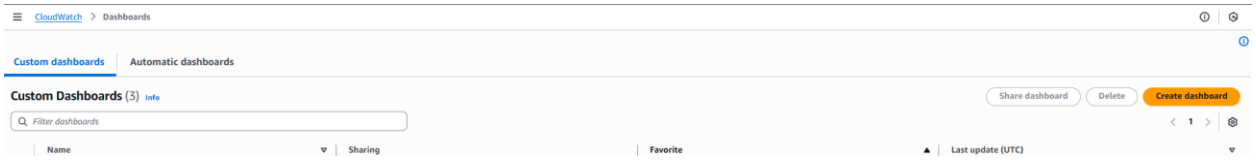
Install tomcat and all its dependencies.

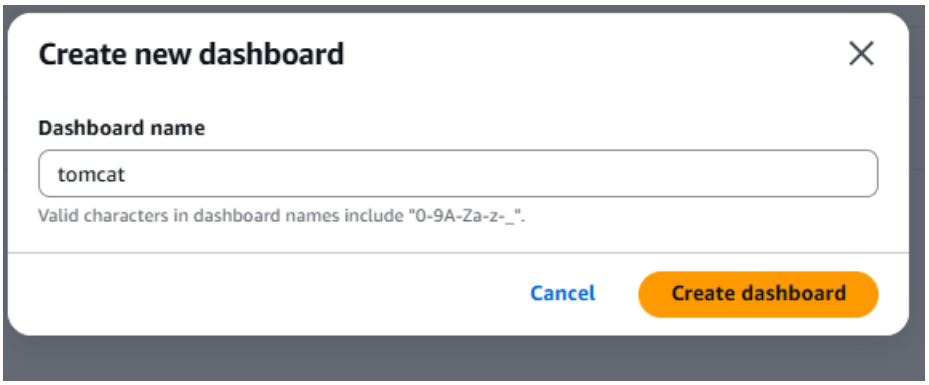
Create a script to check the status of Tomcat:

```
[root@ip-172-31-30-63 ~]# cat monitor.sh
#!/bin/bash
# Check if Tomcat service is active
if systemctl is-active --quiet tomcat; then
  STATUS=1
else
  STATUS=0
fi
# Push custom metric to CloudWatch
aws cloudwatch put-metric-data \
  --namespace "TomcatMonitoring" \
  --metric-name "TomcatStatus" \
  --value $STATUS \
  --region us-east-1
```

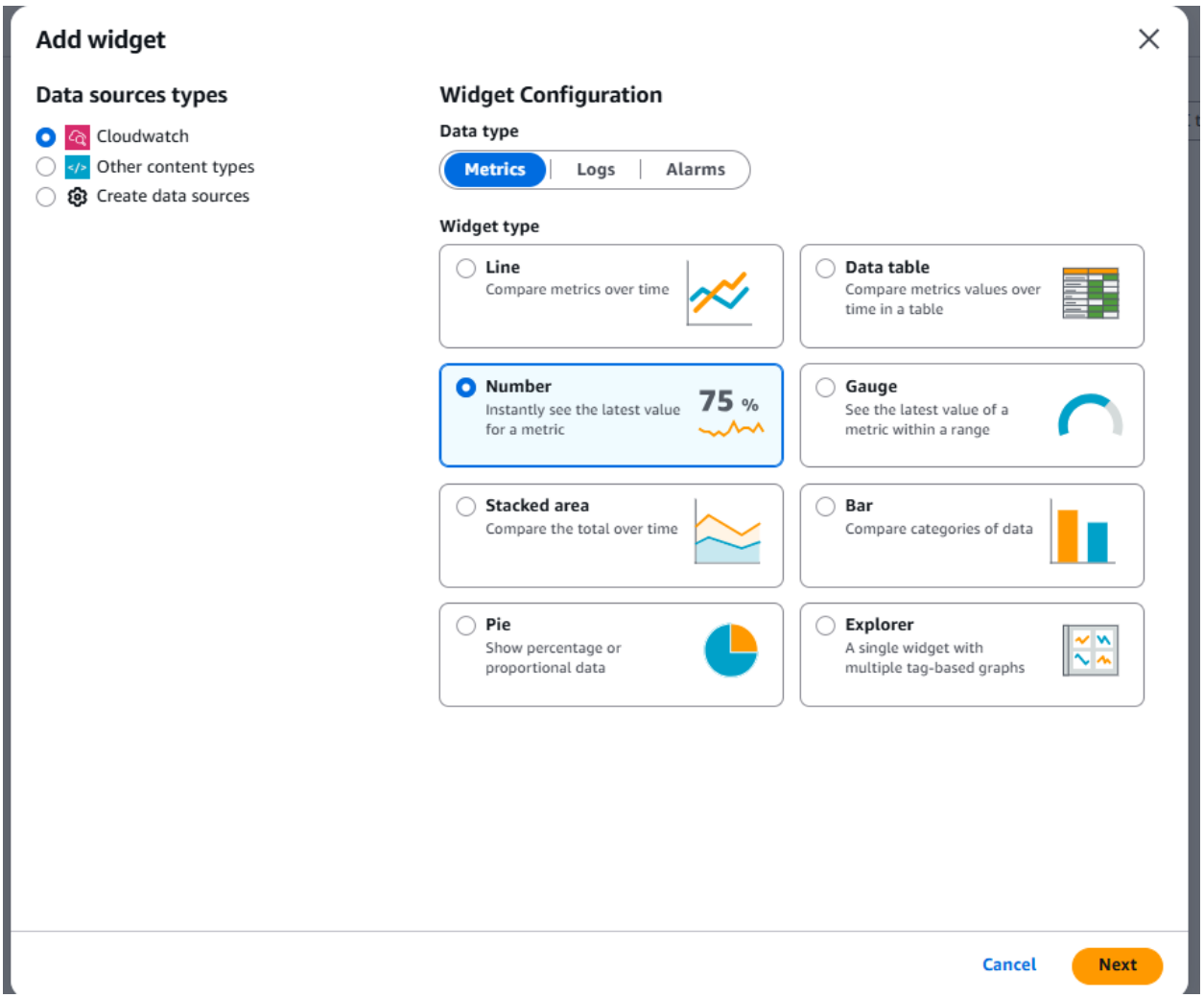
Run the script

And create the dashboard in CloudWatch:

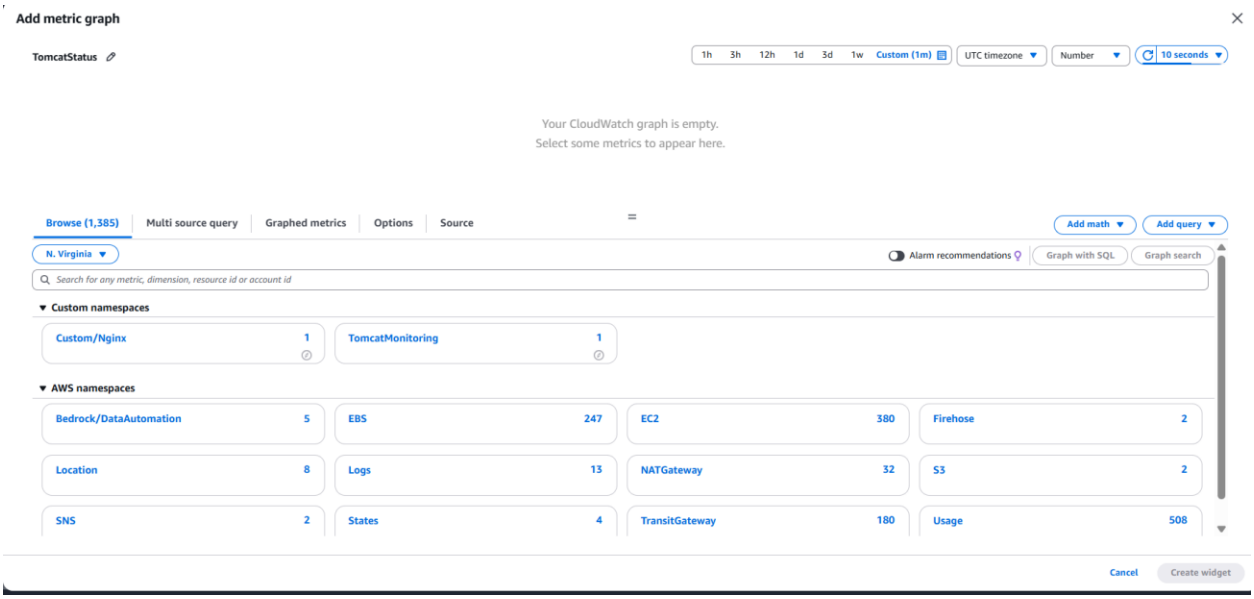


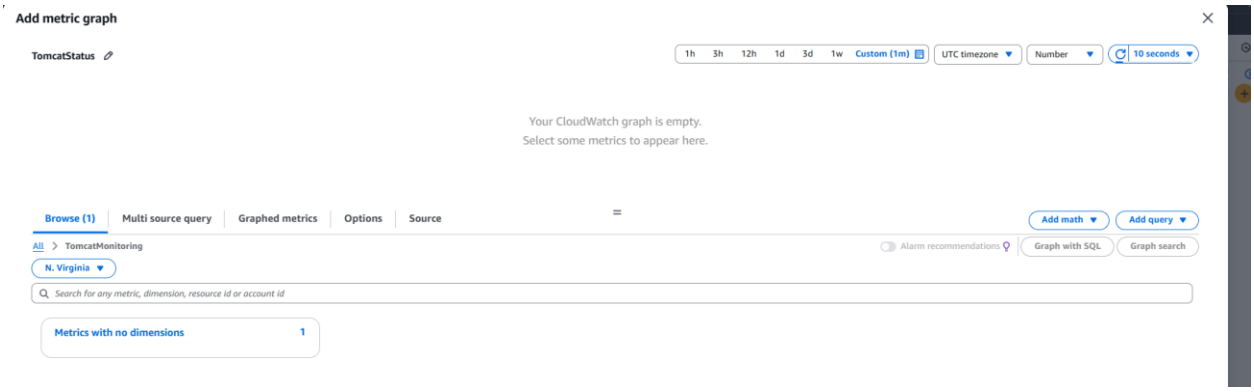


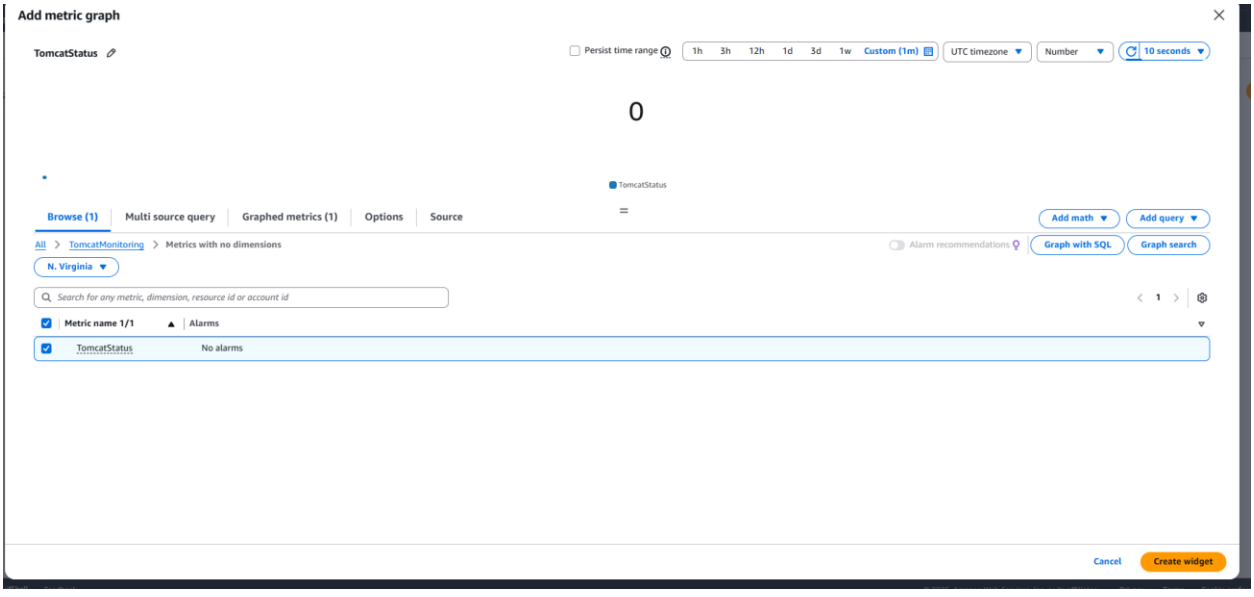
Select your required widget:



Select here the custom namespace i.e., tomcat:


Add metric graph
TomcatStatus
1h 3h 12h 1d 3d 1w Custom (1m)
UTC timezone
Number
10 seconds
Your CloudWatch graph is empty.
Select some metrics to appear here.
Browse (1,385)
Multi source query
Graphed metrics
Options
Source
Add math
Add query
N. Virginia
Alarm recommendations
Graph with SQL
Graph search
Search for any metric, dimension, resource id or account id
Custom namespaces
Custom/Nginx 1
TomcatMonitoring 1
AWS namespaces
Bedrock/DataAutomation 5
EBS 247
EC2 380
Firehose 2
Location 8
Logs 13
NATGateway 32
S3 2
SNS 2
States 4
TransitGateway 180
Usage 508
Cancel
Create widget


Add metric graph
TomcatStatus
1h 3h 12h 1d 3d 1w Custom (1m)
UTC timezone
Number
10 seconds
Your CloudWatch graph is empty.
Select some metrics to appear here.
Browse (1)
Multi source query
Graphed metrics
Options
Source
Add math
Add query
All
TomcatMonitoring
Alarm recommendations
Graph with SQL
Graph search
N. Virginia
Search for any metric, dimension, resource id or account id
Metrics with no dimensions 1

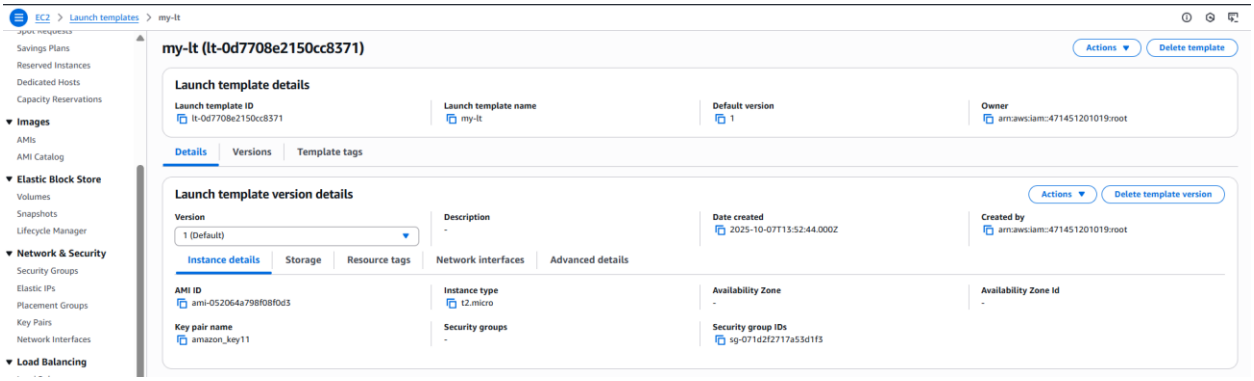

Add metric graph
TomcatStatus
Persist time range
1h 3h 12h 1d 3d 1w Custom (1m)
UTC timezone
Number
10 seconds
0
TomcatStatus
Browse (1)
Multi source query
Graphed metrics (1)
Options
Source
Add math
Add query
All
TomcatMonitoring
Metrics with no dimensions
Alarm recommendations
Graph with SQL
Graph search
N. Virginia
Search for any metric, dimension, resource id or account id
Metric name 1/1
Alarms
TomcatStatus
No alarms
Cancel
Create widget

---

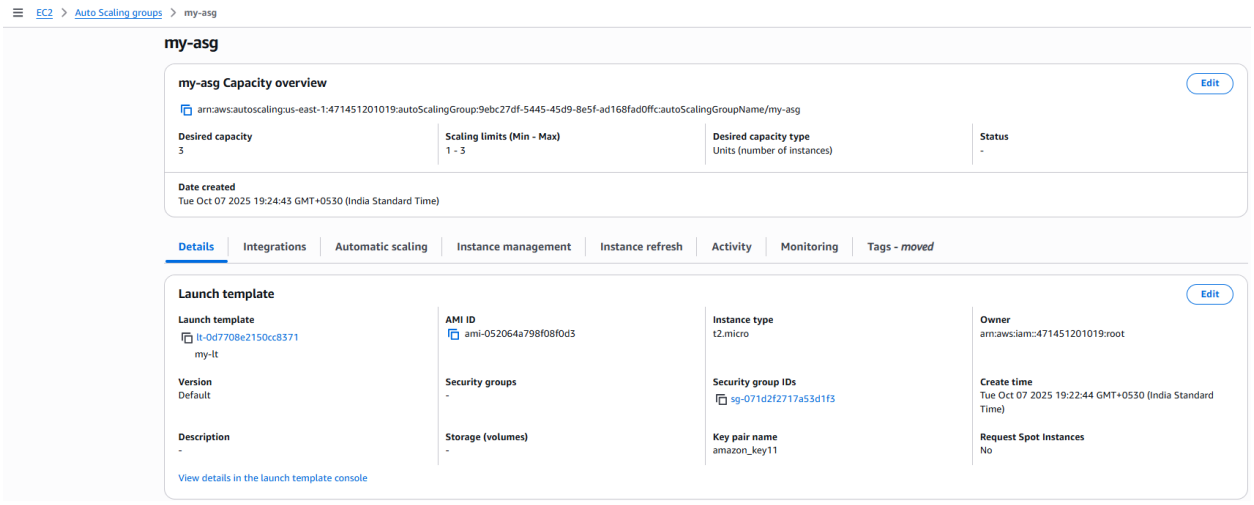
12. If CPU utilization is more than 70%, then it should trigger auto scaling and launch new instance.

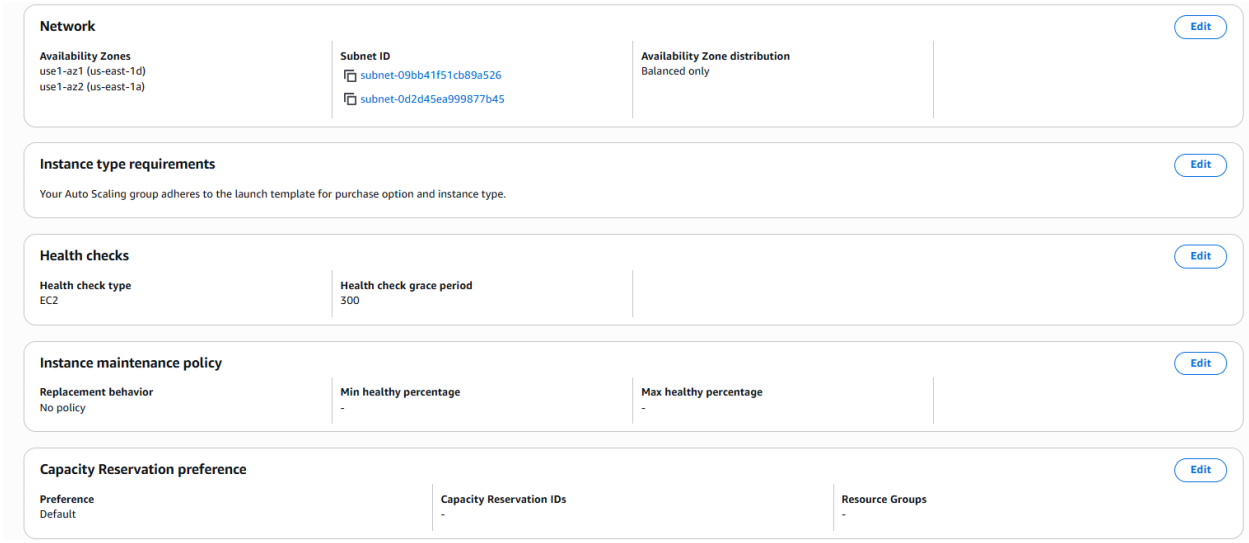
    In AWS, **Auto Scaling** automatically adjusts the number of EC2 instances based on demand. To trigger scaling when CPU utilization exceeds 70%, you typically:

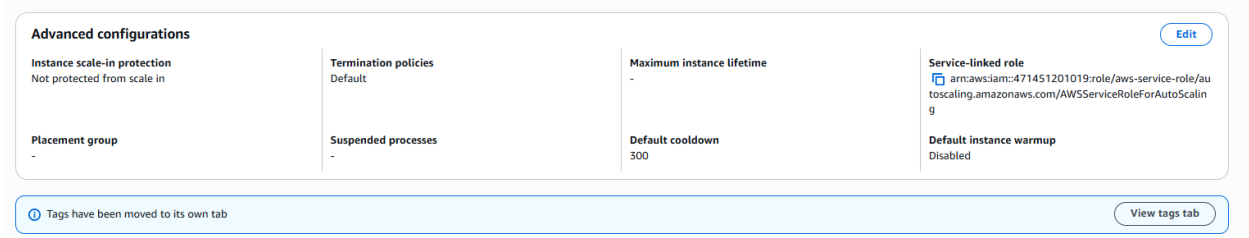
1. Create a Launch Template.



2. Create an Auto Scaling Group (ASG).

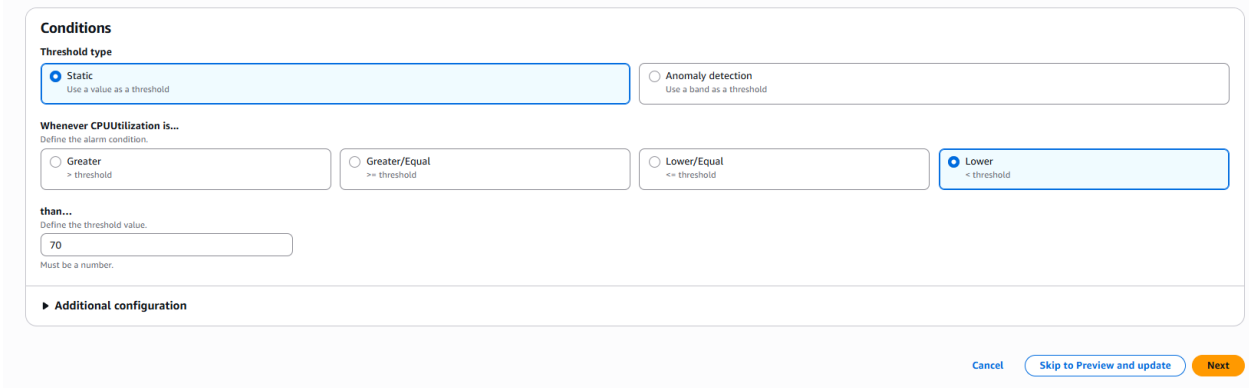




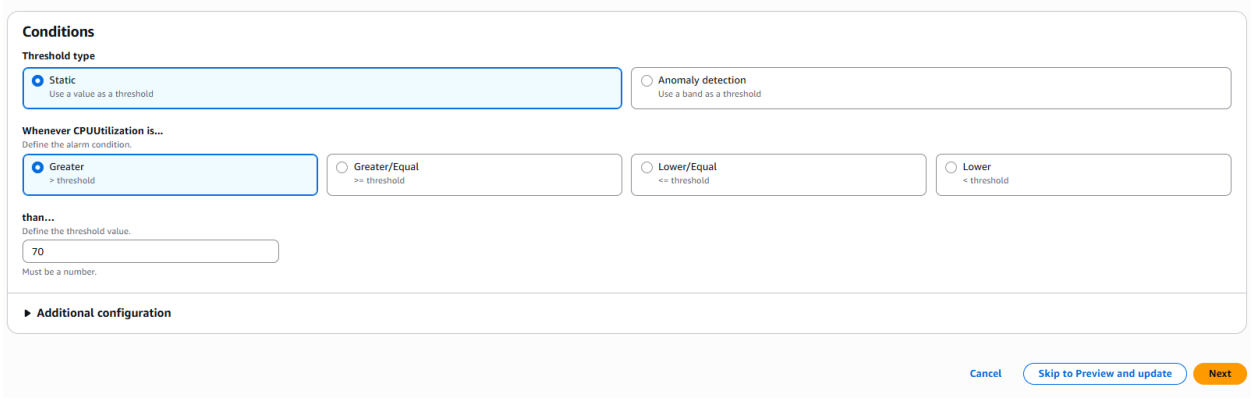


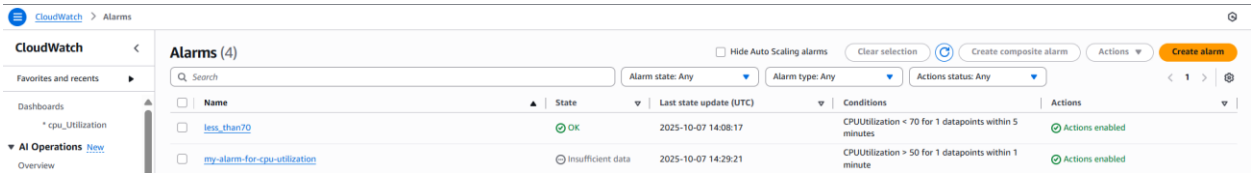
3. Create a CloudWatch alarm for CPU utilization > 70%.

Create ok alarm for cpu utilization <70:



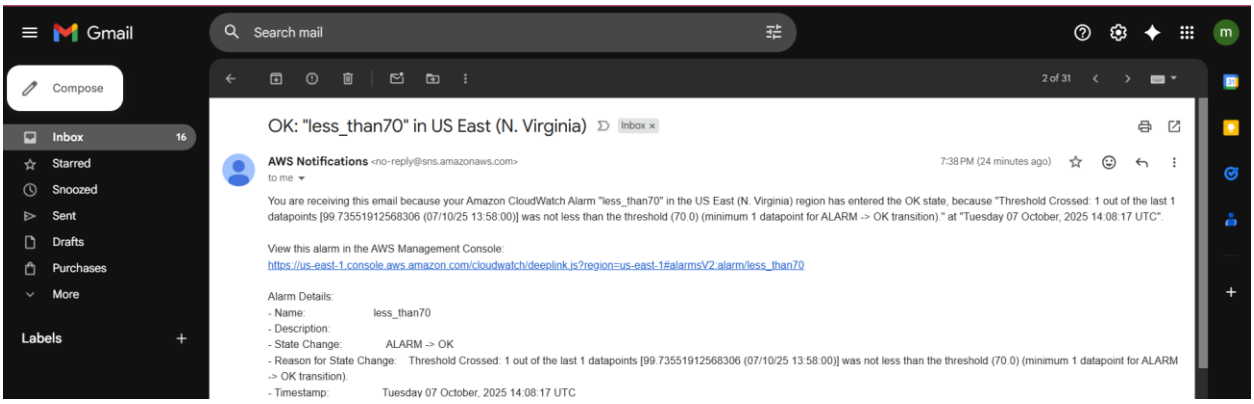
Create in-Alarm for cpu utilization >70:





4. Attach the CloudWatch alarm to scaling policies that increase or decrease the number of instances.

Set the notification while creating the alarm by creating the topic and create subscription and provide the endpoint, then we will get the notification whenever alarm triggered:



----------complete-------