AWS EC2 conclusion:

Load balancer -> Autoscaling group

Application Loadbalancer

Network Loadbalancer

Gateway Loadbalancer

Application LB -> operates at Layer 7, ex: http, https, microservices, advanced routing, path-based-routing.

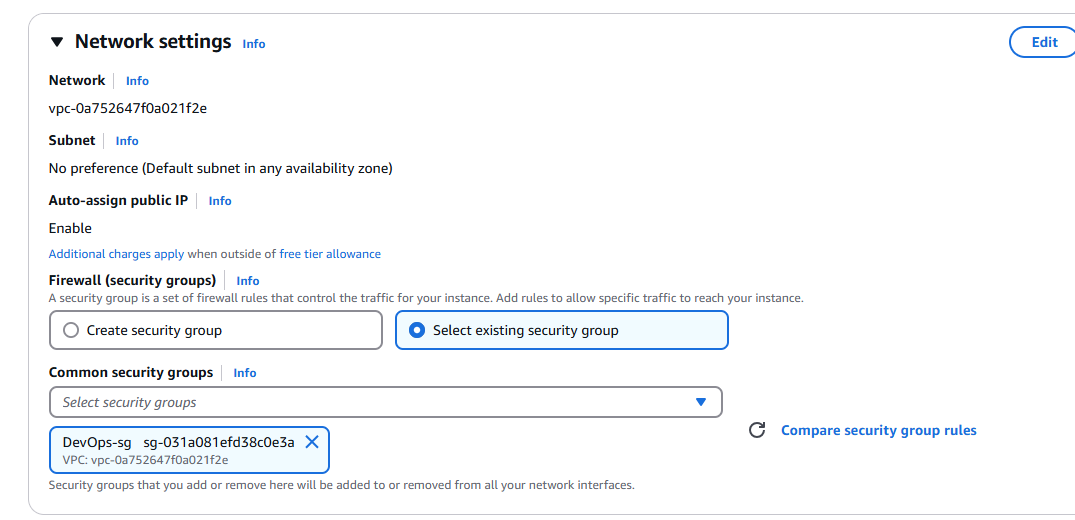
Network Load balancer: at OSI Layer 4 --> Transport layer --> whenever you want to provide ultra high performance with lowest latency --> Gaming application, video streaming, IOT application -> wherever latency should be minimum

Gateway LB --> works at Layer 3 Network layer: whenever you want to go with Third-party communications like VPN. VPNs, Firewalls, where security concerns are high. Requires high security

Firewall (VPN) ---> Gateway Loadbalancer

| |

S1 S2



#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Banking Server 1</h1></html>" > index.html

systemctl start httpd

#! /bin/bash

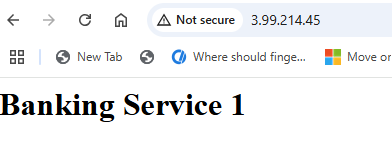
sudo su

yum install httpd -y

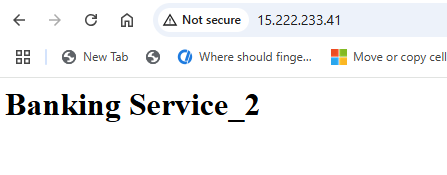
cd /var/www/html

echo "<html><h1>Banking Server\_2</h1></html>" > index.html

systemctl start httpd



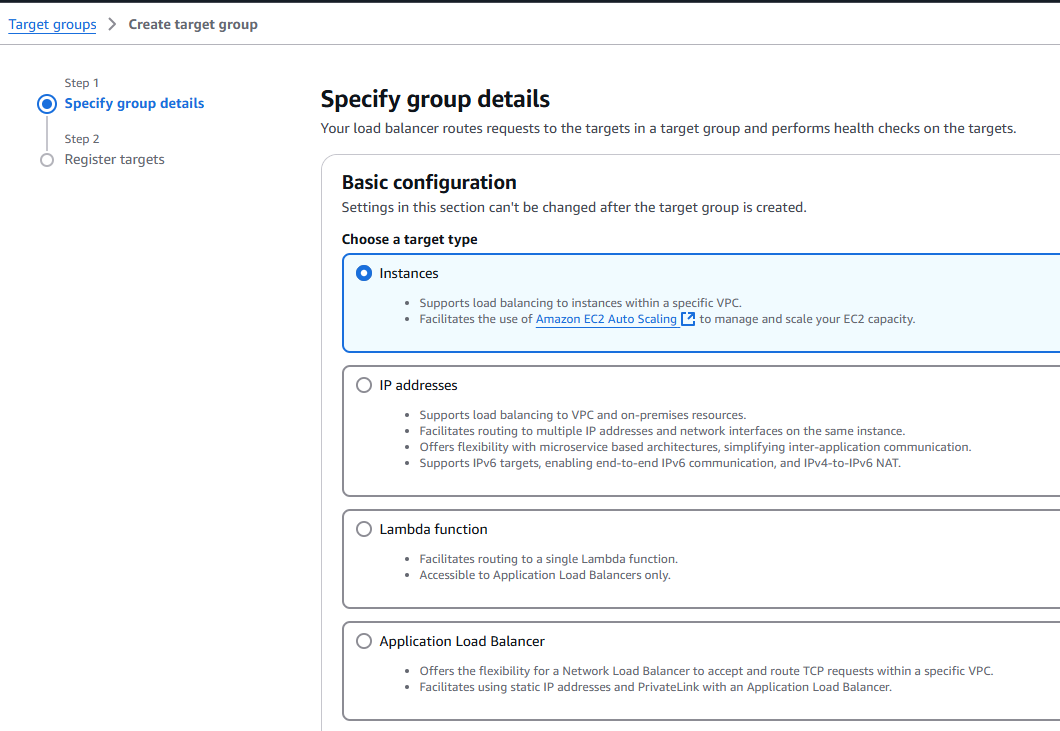
It should be Banking Server 1 and 2 because these are Monolithic applications not Microservices

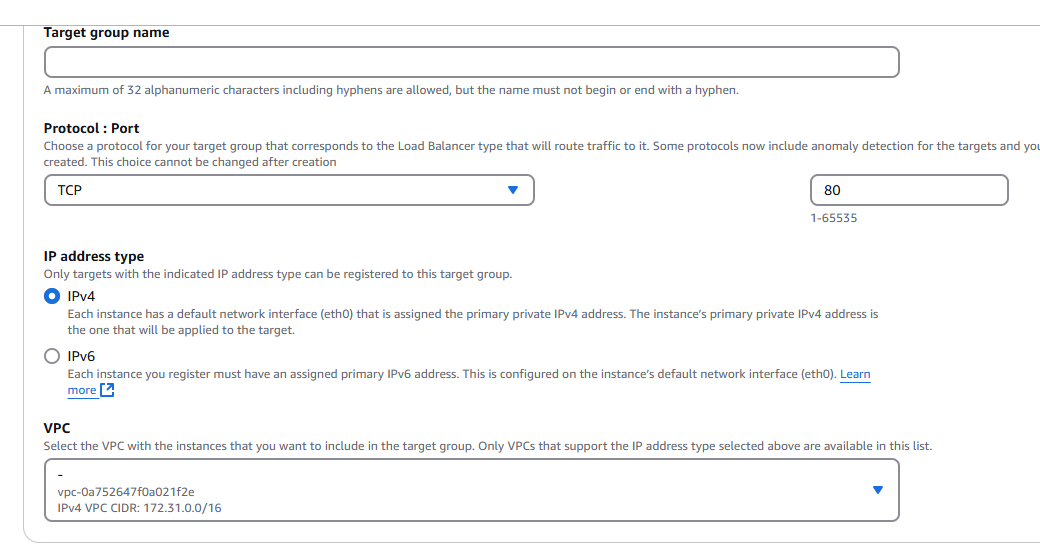


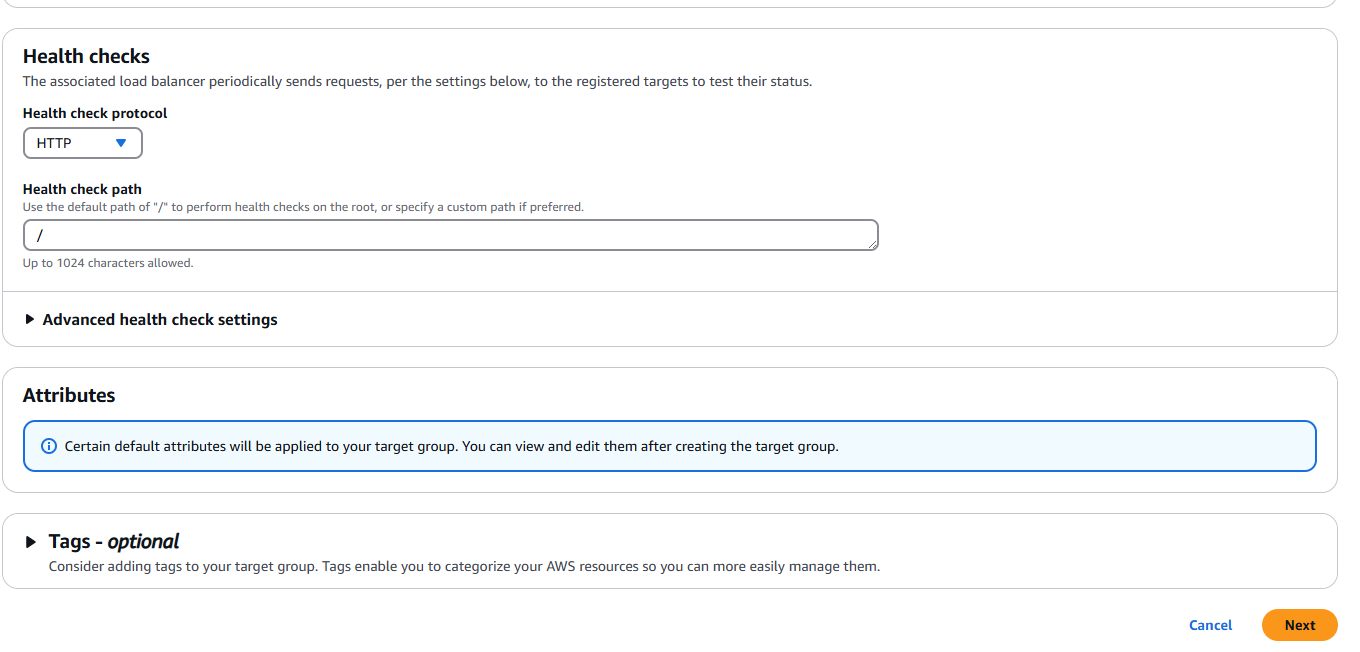
Network LoadBalancer: it is TCP, UDP

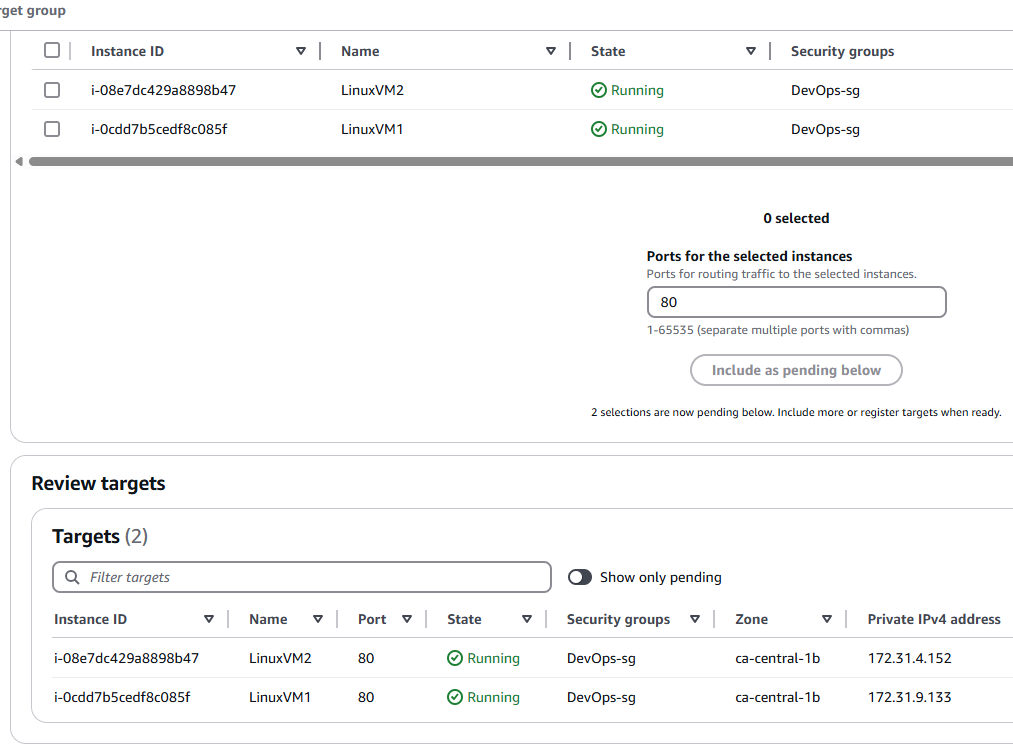
Application LoadBalancer: it is HTTP, HTTPS

Target group:

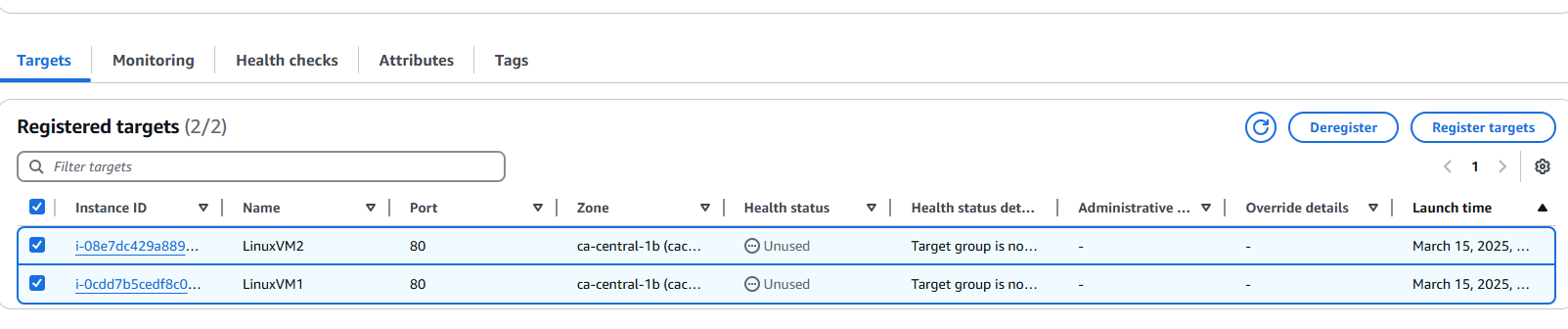




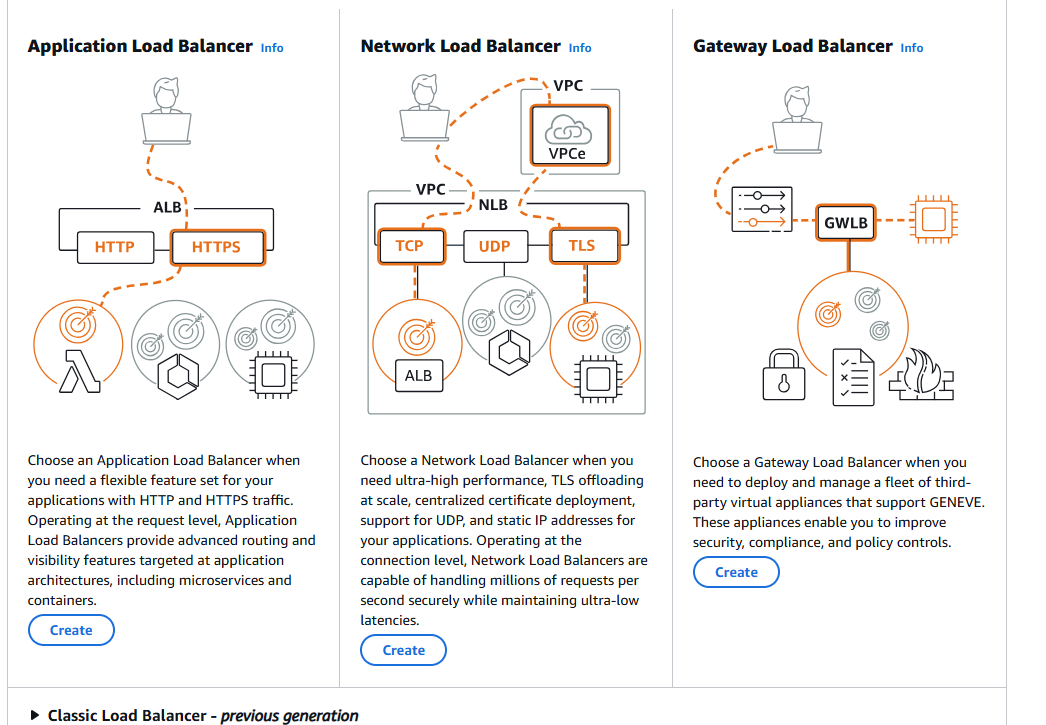




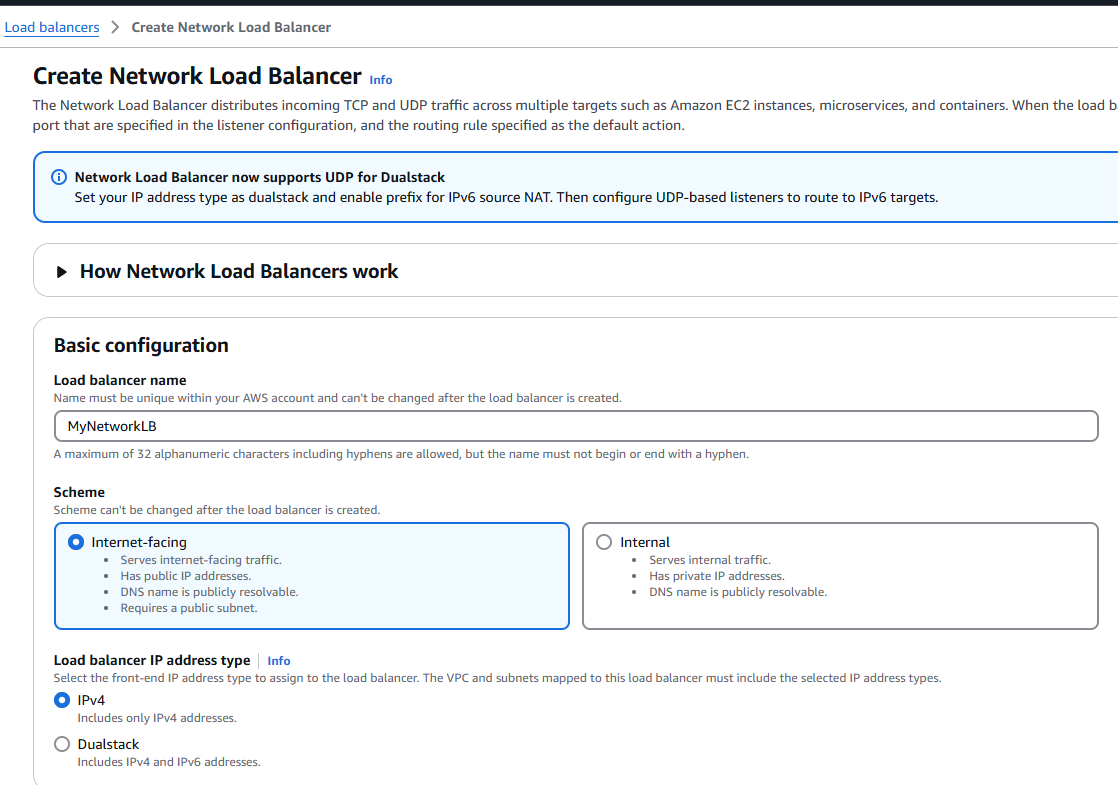
Register targets

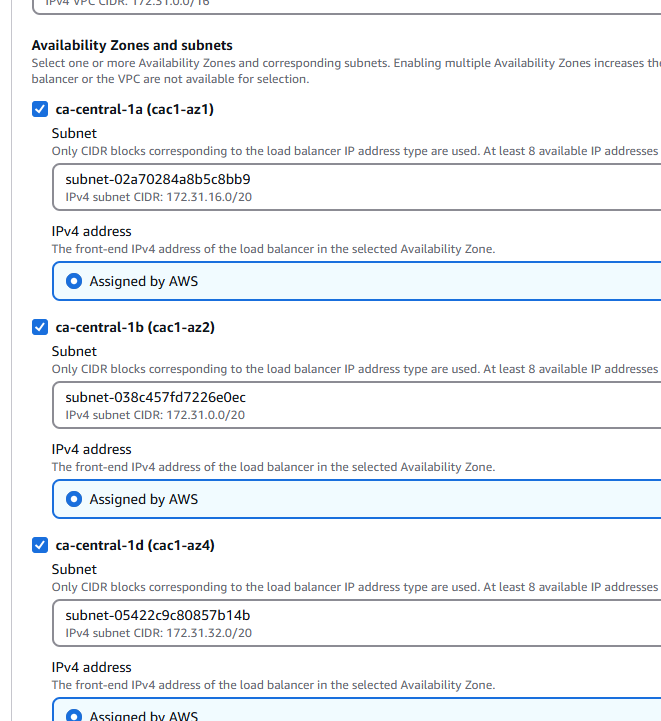


Now go to Load Balancer



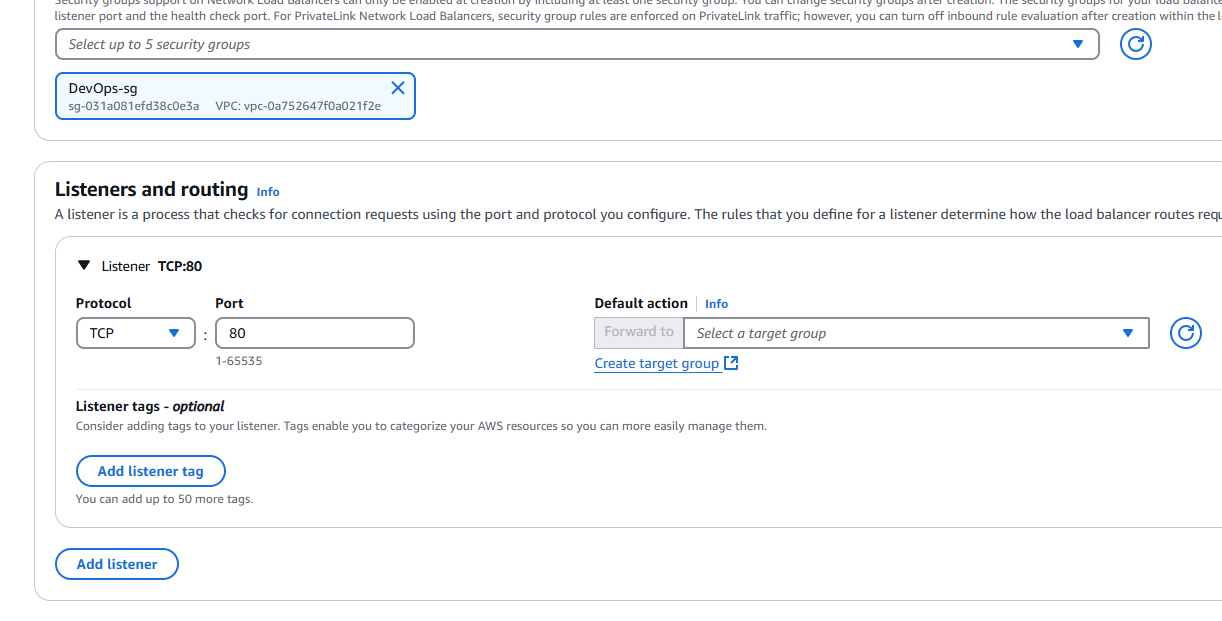
Create Network Loadbalancer



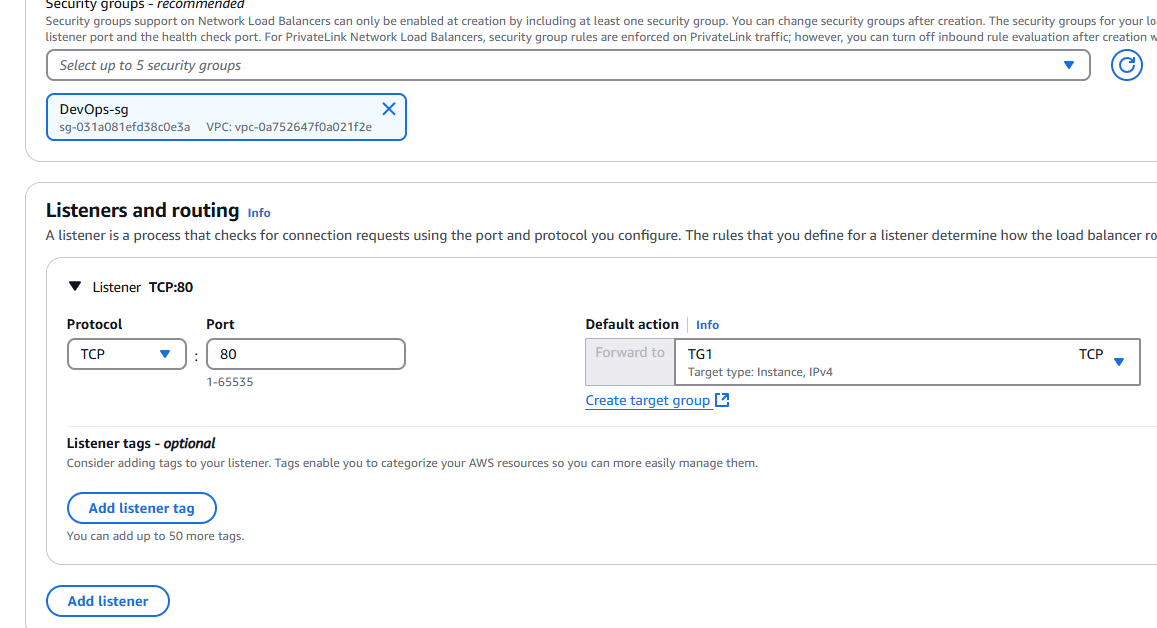


Security group selected

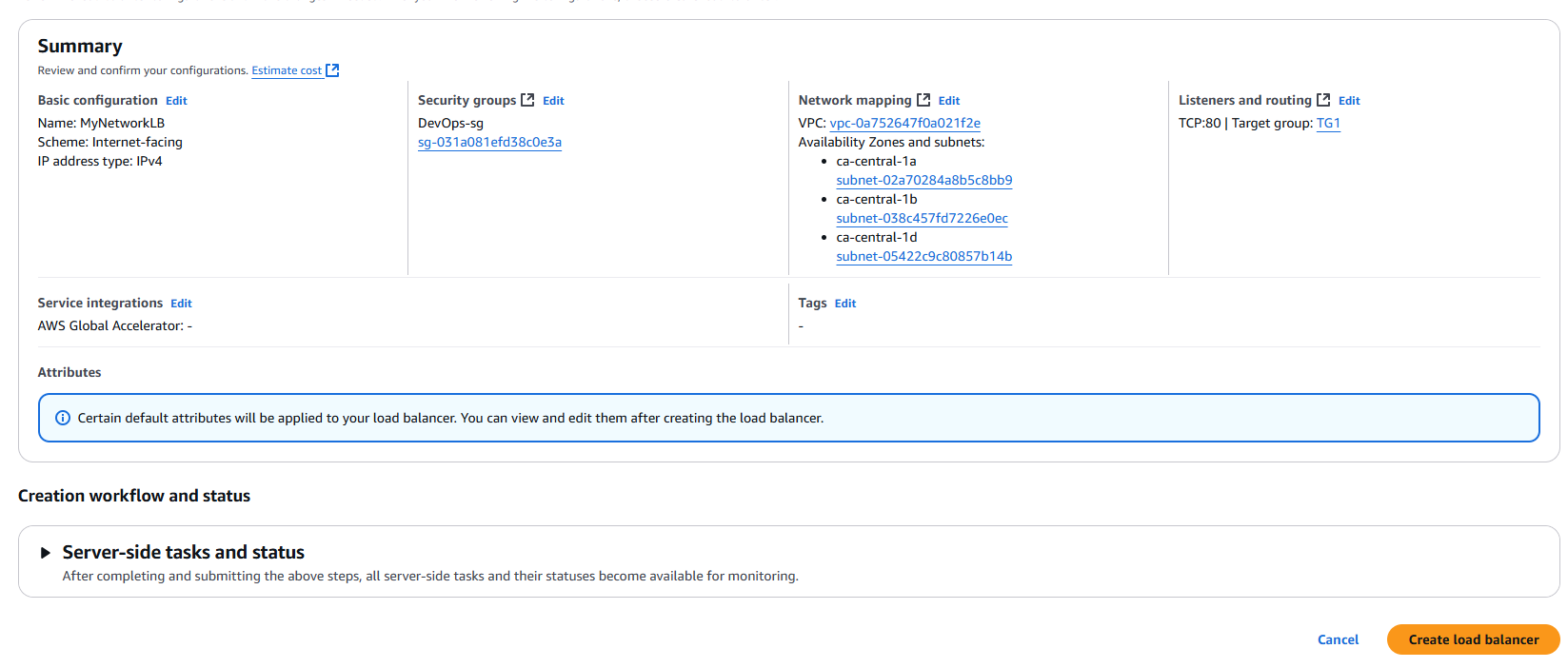
**And TCP selected**



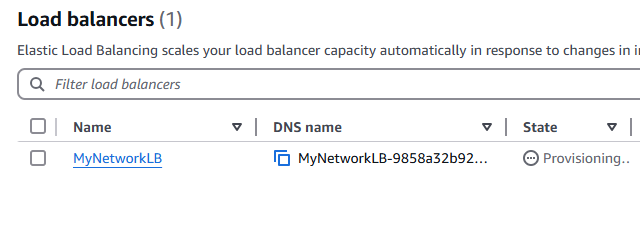
Select Target group

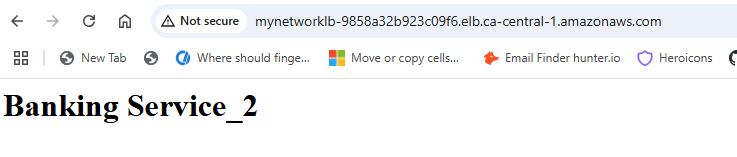


Create load balancer



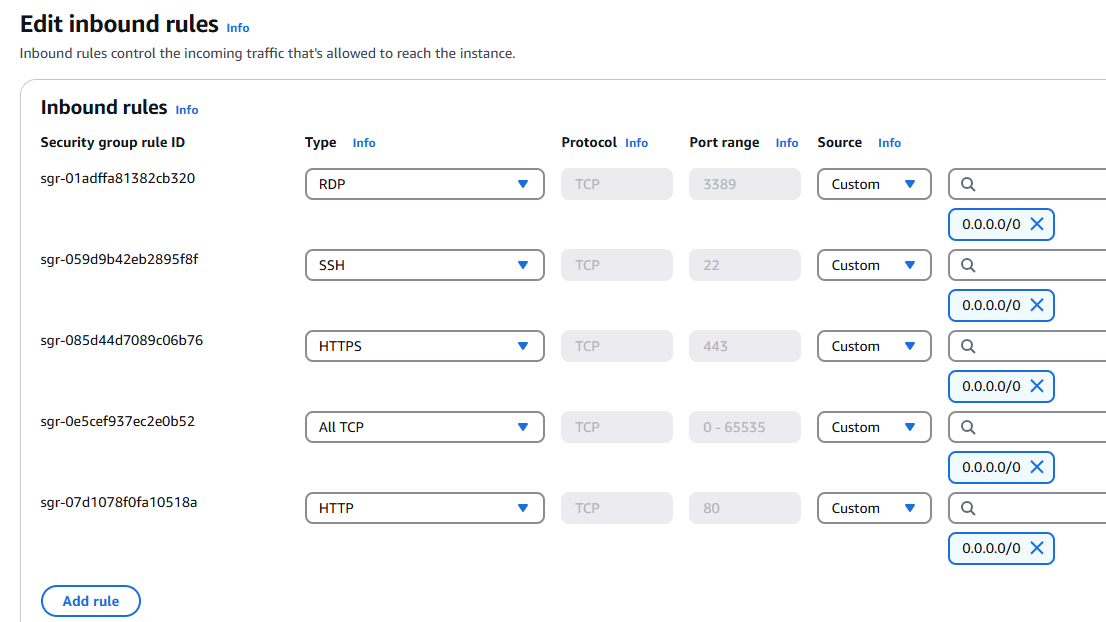
Currently Provisioning

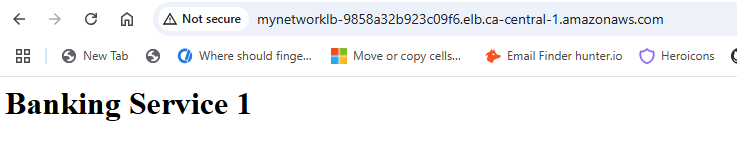




Security groups -> inbound rules

May be add All TCP -> 0.0.0.0/0

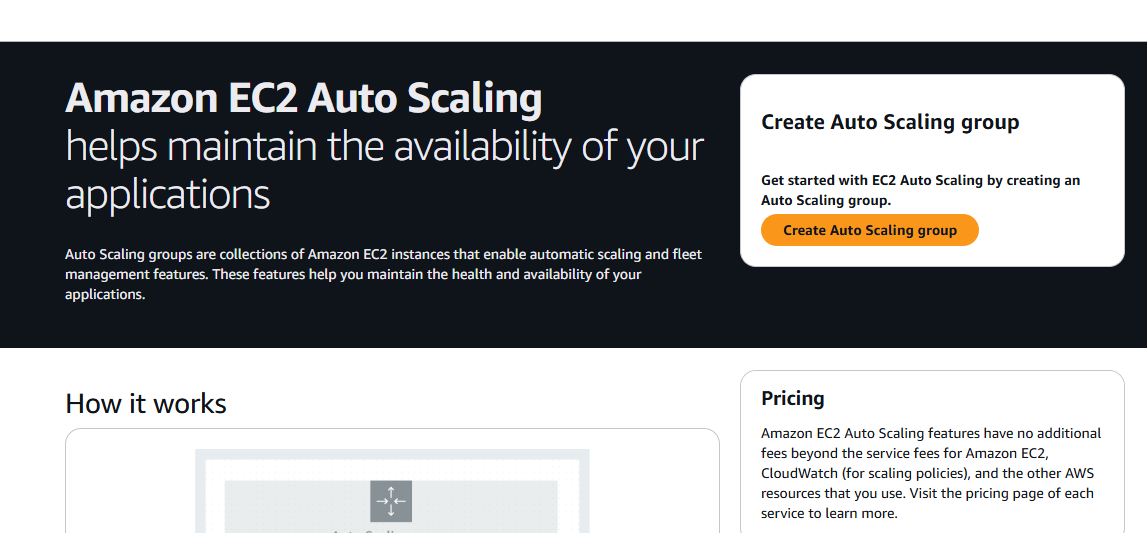




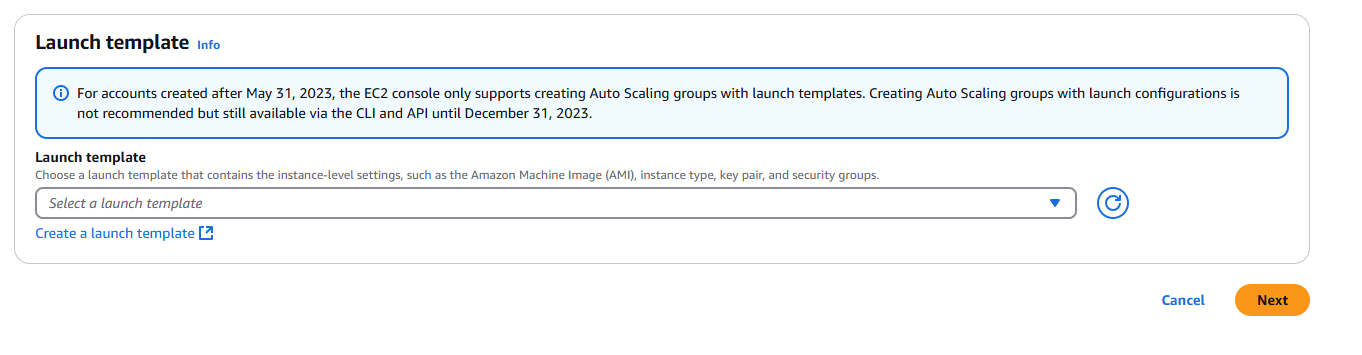
Why we have to go with Auto-scaling?

If you want to modify infrastructure based on the demand, auto-scaling comes into picture

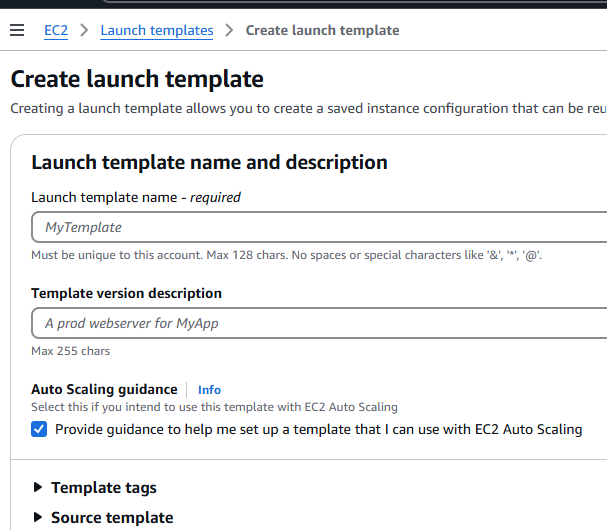
Amazon EC2 Auto Scaling

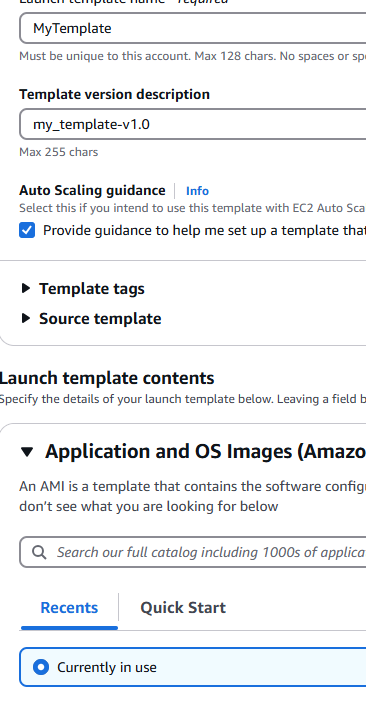


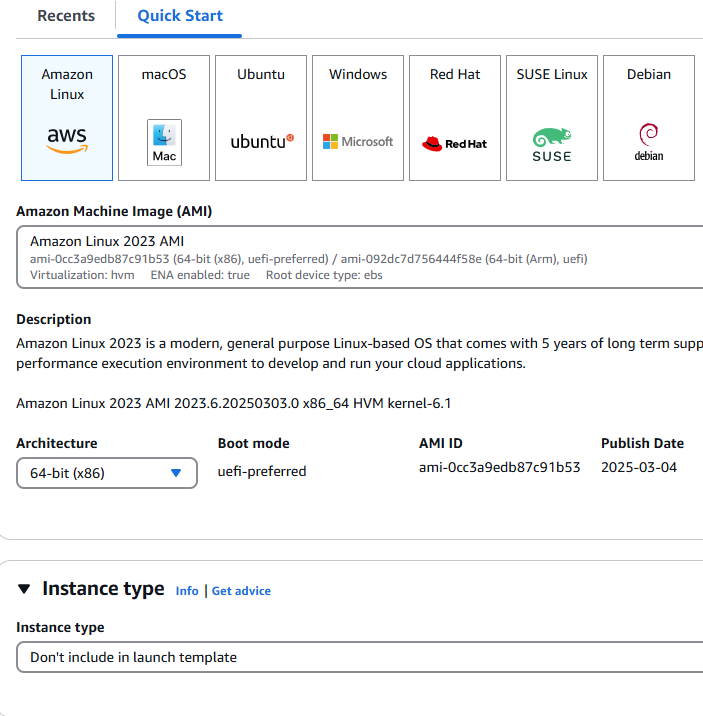
Create Auto Scaling Group

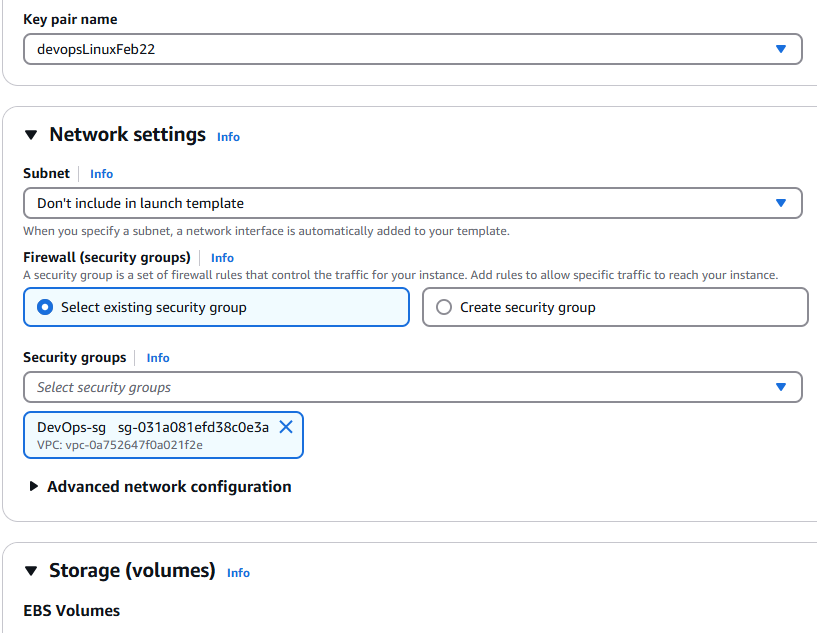


Click ‘Create a launch template’

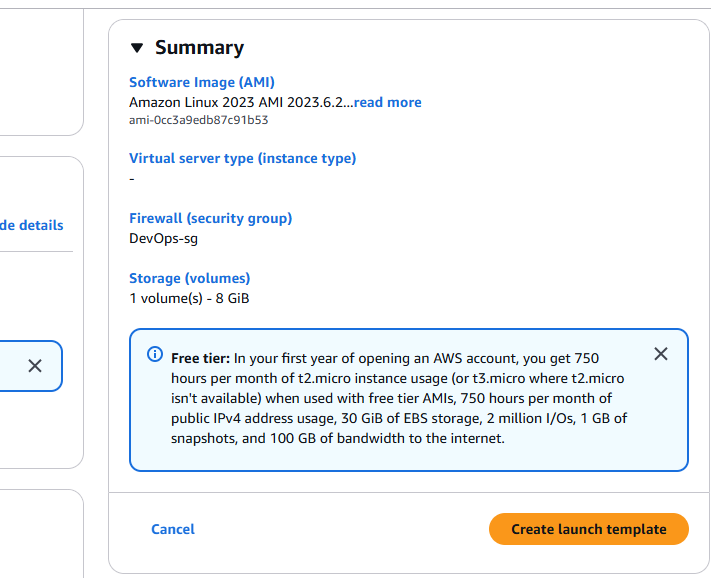


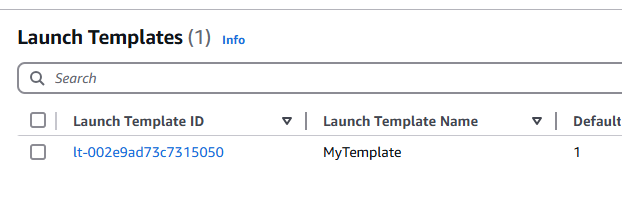




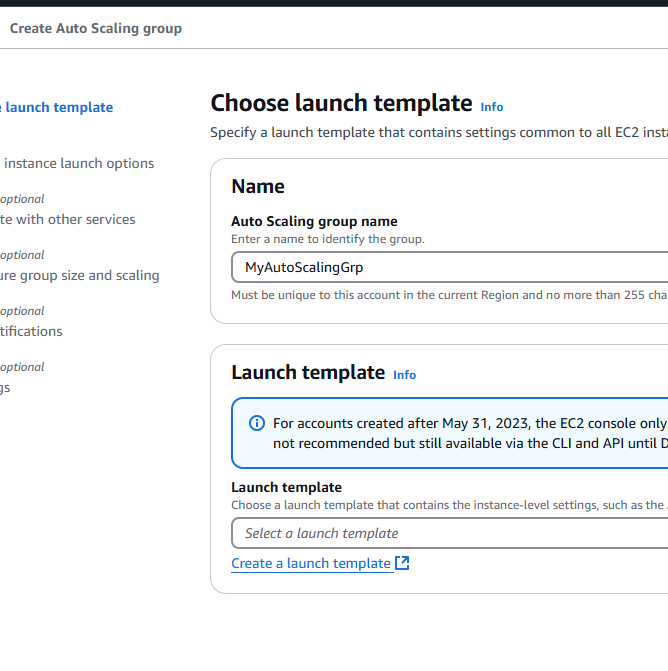


Configure Security groups, Keypair --> Create launch template

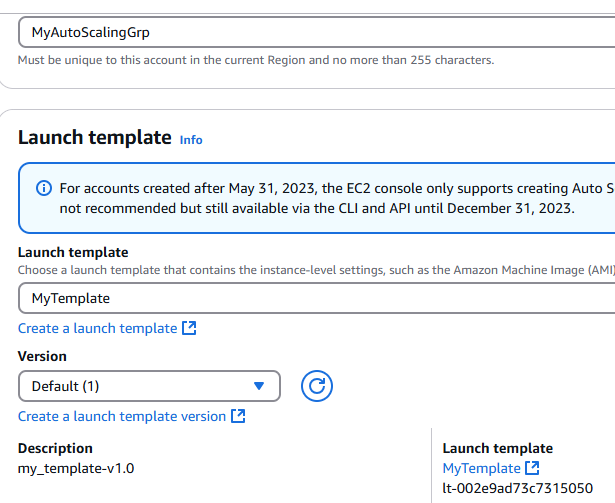




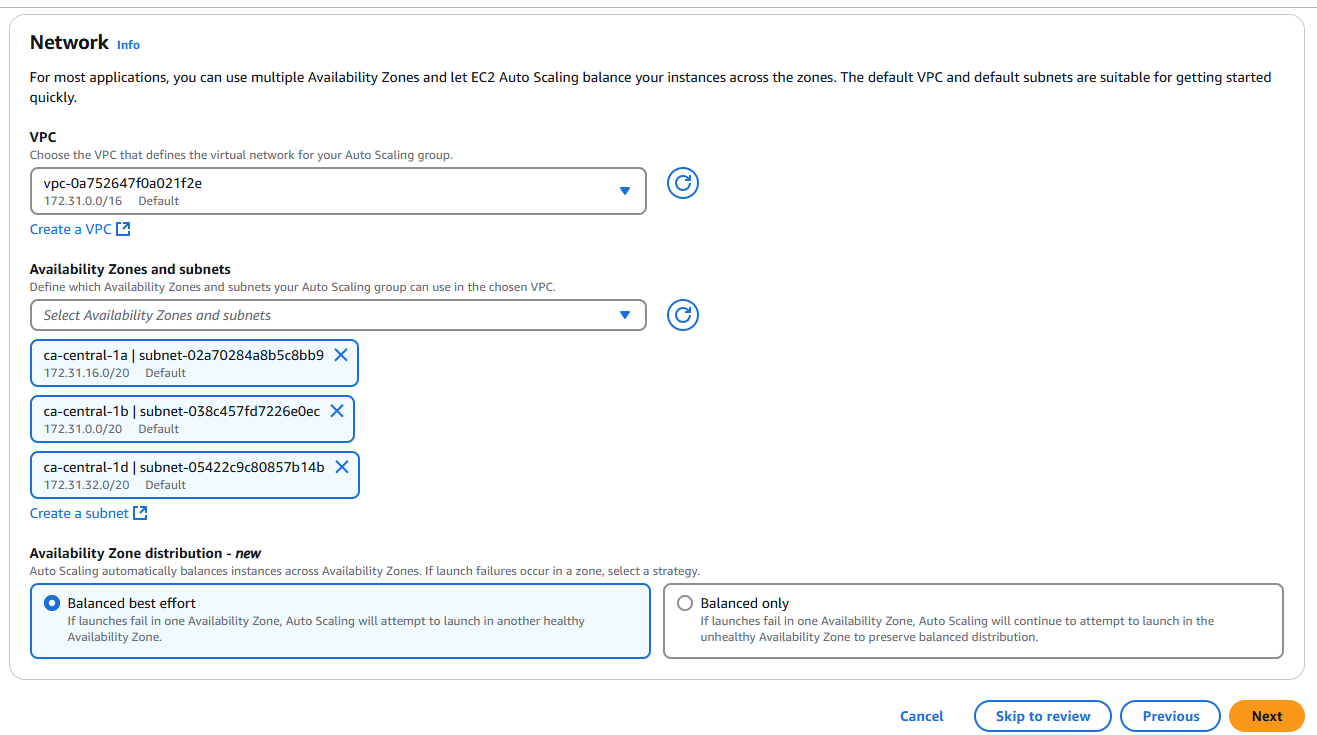
Go back to Create Auto Scaling Group



Select ‘MyTemplate’

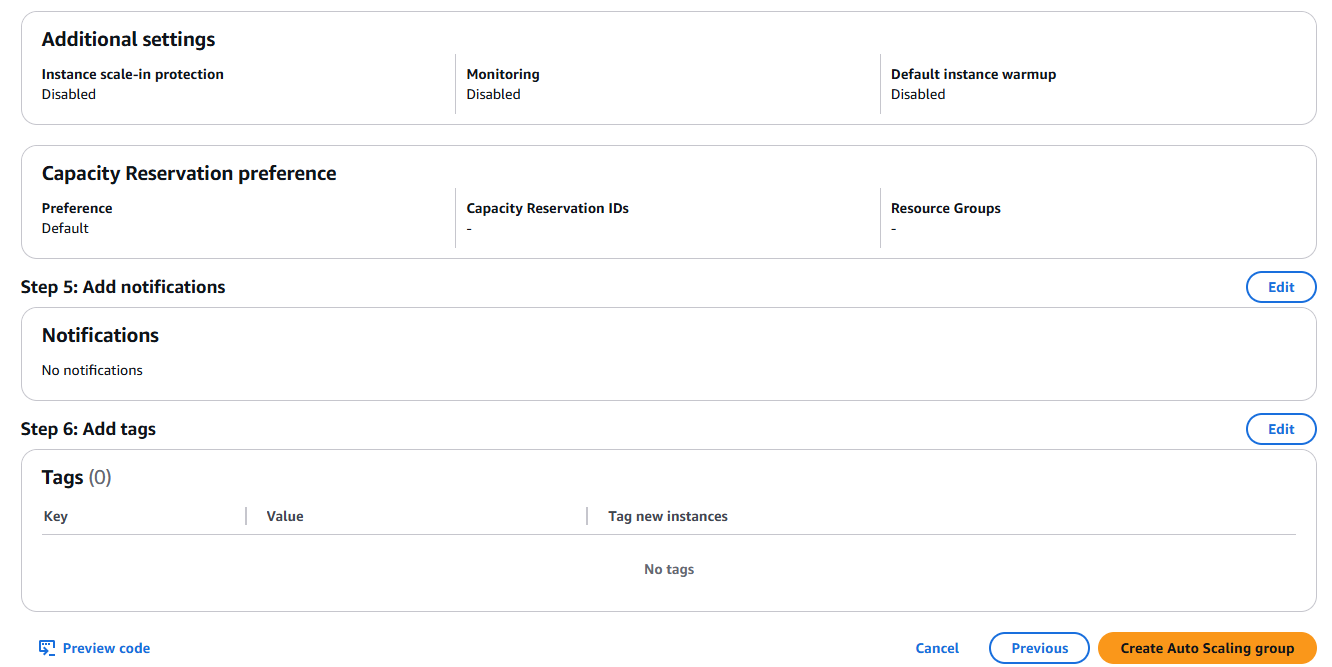


Click Next ---> Pick Availability Zones and Subnets

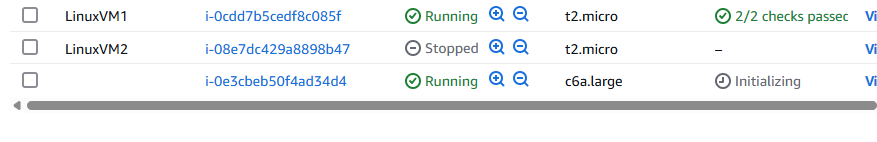


All others I want to keep it default ---> click Next

Review and create Auto Scaling Group



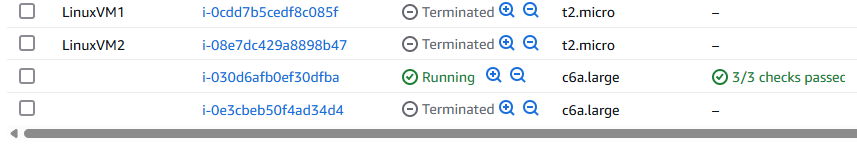
Go to EC2 Instances



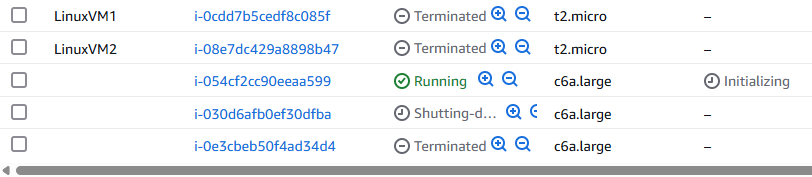
We can see one more instance is being created automatically --> Auto scaling is creating this instance

We manually create LinuxVM1 and LinuxVM2, third is automatic

Now I go back to EC2 instances and I see that one more instance is automatically created. This is fault-tolerance of Auto scaling



I terminate one more instance, which initializes another instance



AutoScaling groups:

It is used to adjust the capacity required to handle the load

If number of requests are increasing, then servers must also be increased to give smooth experience for customers and similarly if requests are decreasing then number of servers should be decreased to manage cost. In that case, we can go with AutoScaling group.

1. Fault-tolerance
2. Cost management
3. High availability

---> To create AutoScaling group, we use Launch Template - it is used to specify configuration required to launch new VM whenever needed

==========================================

How an application will be deployed into a new VM created by AutoScaling group?

Kubernetes is the default option. However, lets look into custom AMI for now

Using custom AMI ---> what’s AMI? Amazon Machine Image

User data script

Kubernetes cluster

MyAMI

Launch template

EC2-VM

Application

Autoscaling group

Server

**Different types of AWS EC2 instances:**

General purpose

Compute optimized

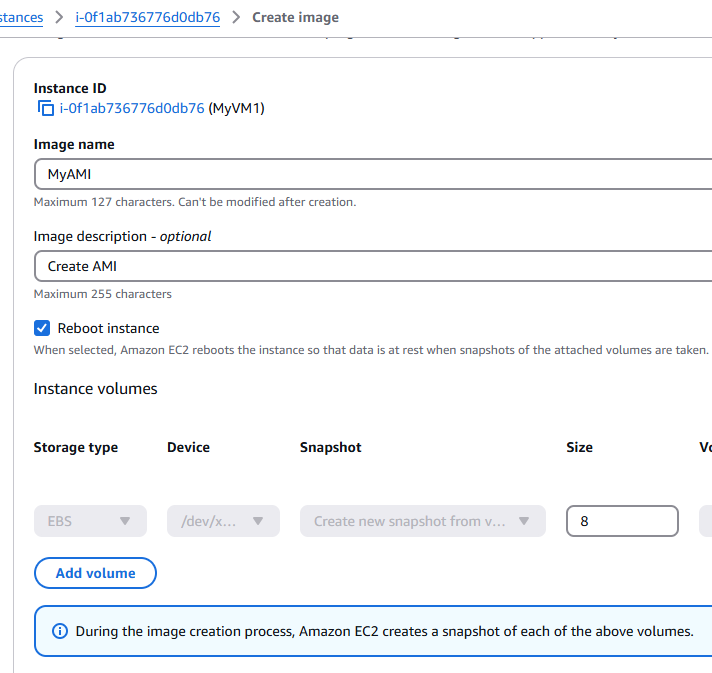
Storage optimized

Memory optimized

Accelerated computing

High performance computing

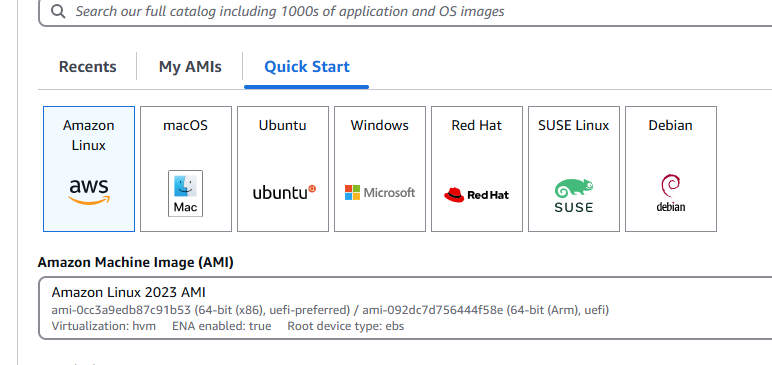
Select the instance ---> Actions --> Image and templates --> Create image



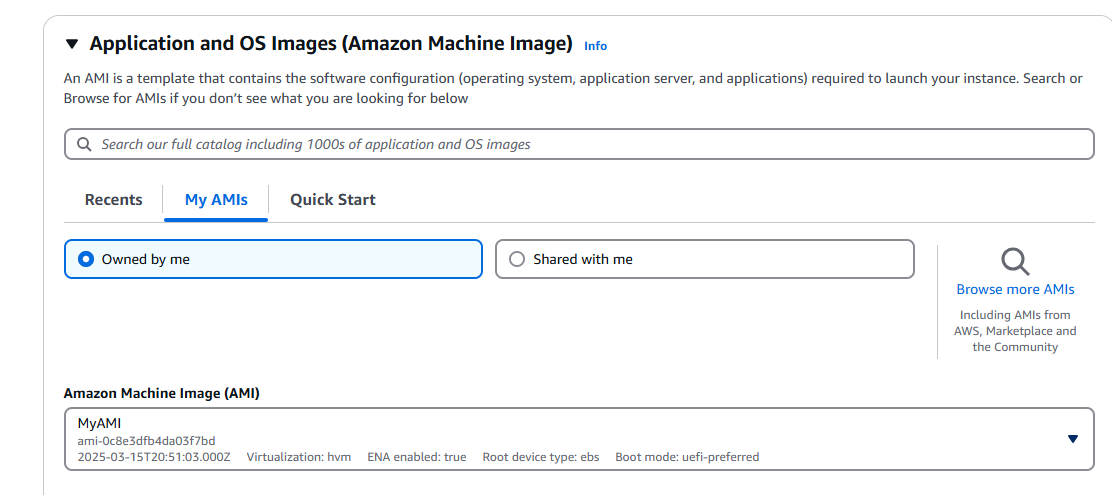
Click Create Image

Now click on Launch Instances

Now we see a new option : My AMIs earlier it was not there



I can select MyAMI



Launch template looks like a new Launch instance only so it can go into AutoScaling groups if we want to deploy our application (from AMI) on all the automatically created EC2 instances

For every VM we create, by default root volume is there. We can add additional volumes.

VM-1

Root volume

I create a new VM2 with new Root volume. Yes, we can attach new volume to VMs. I can copy data from RV of VM2 into additional volume. Later, I will move Additional volume from VM2 into VM1. I think if VM1 and VM2 use different .pem files (keys), we got to do key replacement using a temporary VM . When I copy data from VM2 RV into Add volume on VM2, it will be according to private key of VM2. Whenever we try to access volumes, it will match public and private keys. In VM1, when we attach a volume it will compare with the private keys. Say if VM2 pem file is lost, create a new temporary VM ---> detach RV from VM1 attach in temp VM then copy from add vol into RV from VM1. The RV will now have IPs associated with the new pem file of temp VM. Now VM1 can use the temp VM pem file to access RV.