41.AWS-B30-Docker-ECR-ECS

--- in this, we will discuss about below sessions.

```
Containers:
- Virtual Machines vs Containers
- Installing Docker & Running Container
- Creating Docker Image and Pusing in to ECR.
- Overview on AWS Elastic Container Services(ECS)
- Creating Tasks & Services
- Performing Rolling Update on the application.
```

Container registry

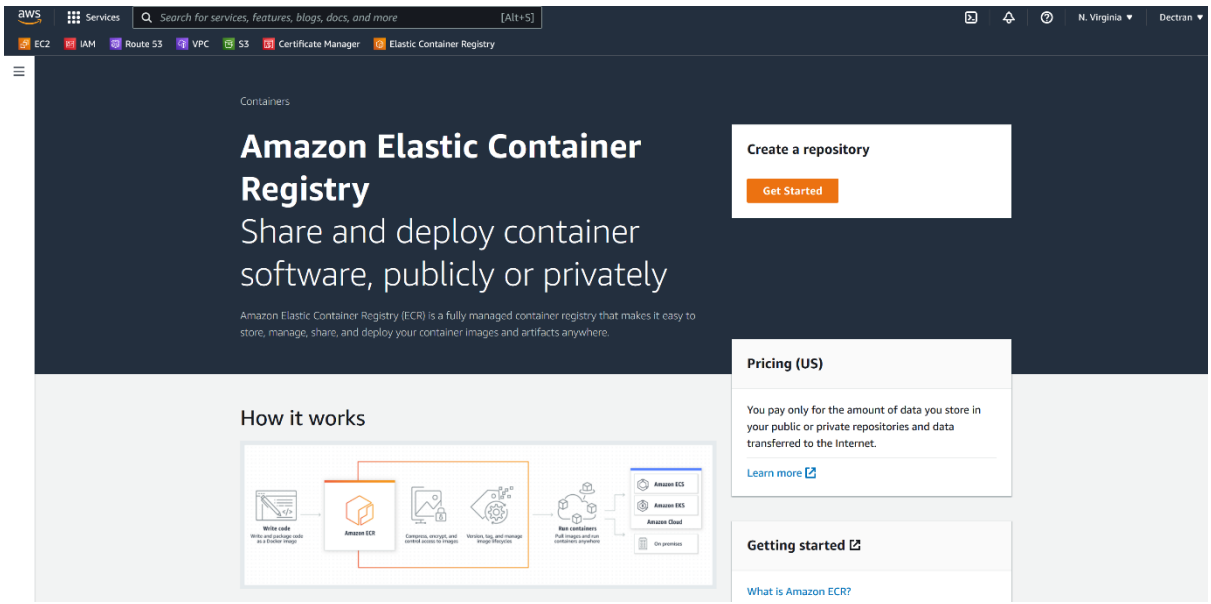
--- the famous docker registries are

1. Hub.docker.com
2. AWS elastic container registry (ECR)
3. Azure container registry (ACR)

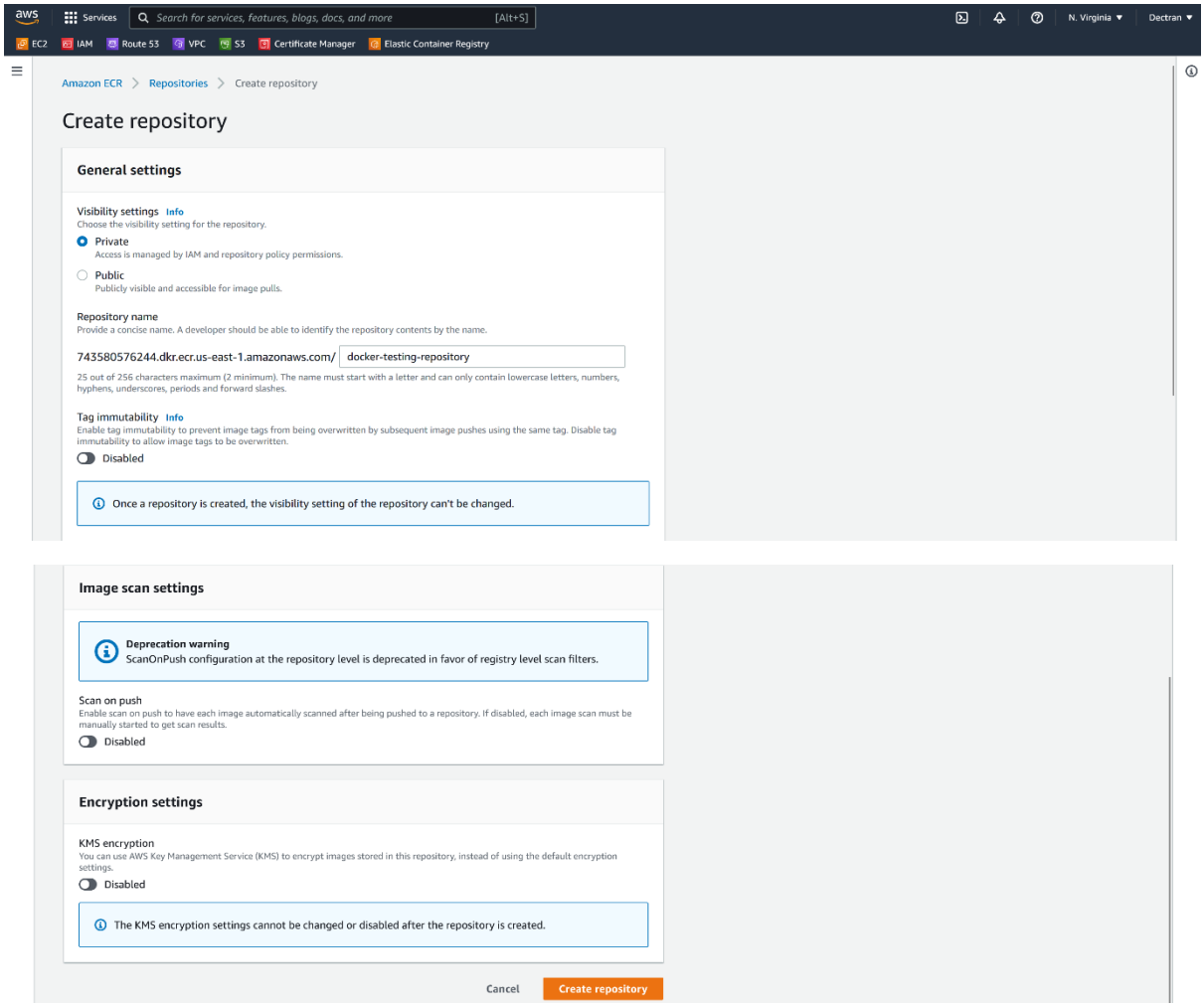
--- NOTE – in container registry, we have Repository, in Repository we have images.

AWS ECR (Elastic container registry)

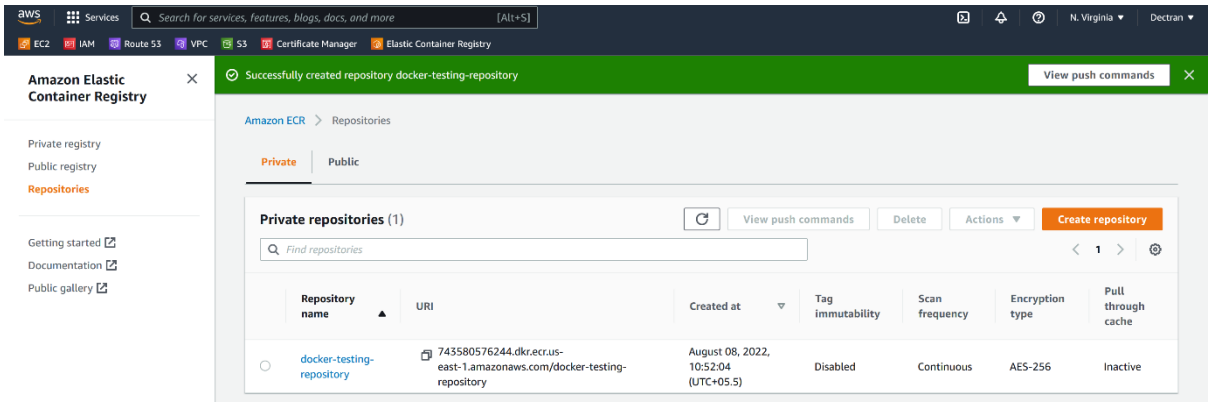
--- go to aws cloud and search for ECR.



--- Click on get started.



--- click on create repository.



--- our private repository got created.

Upload An image to ECR

--- **Reference** - https://github.com/prabhu9652/dockertest1.git

Build image

--- docker build -t prabhu9652/customnginx1:v1 .

List images

--- docker images

Create a container form images

--- docker run --rm -d --name nginx01 --publish 9000:80 prabhu9652/customnginx1:v1

List containers

--- docker ps

Aws cli installation

 # Unzip installation

--- apt update && apt install unzip -y

 # aws cli installation

--- curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

--- unzip awscliv2.zip

--- ./aws/install

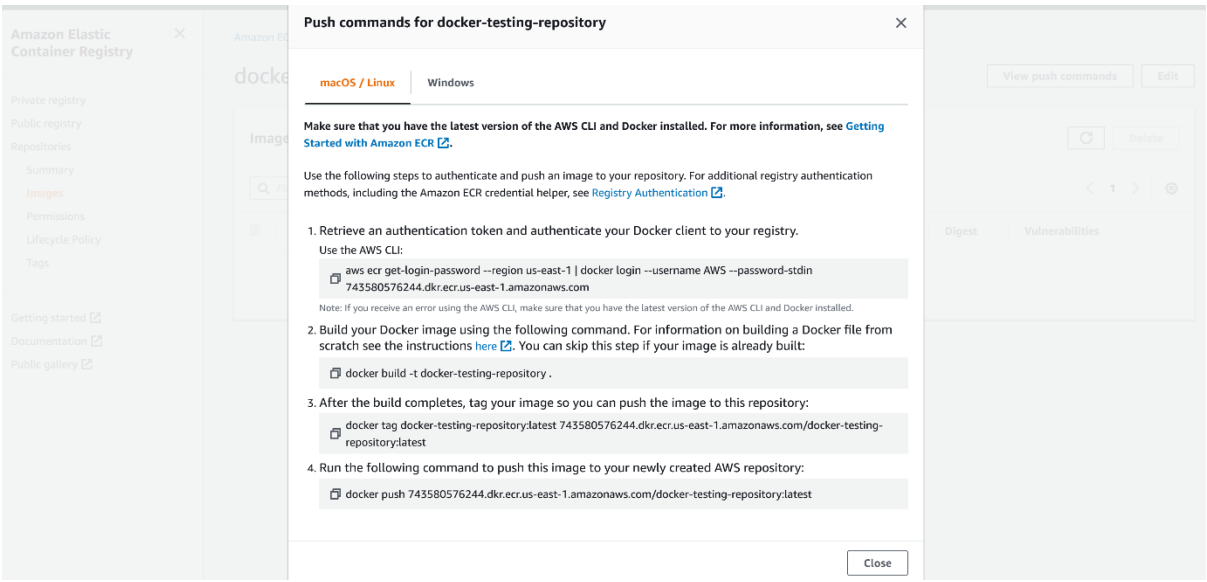
Configure AWS credentials.

--- aws configure

--- **Note** – create IAM Role with necessary permissions and attach that role to instance.

Pushing image to AMAZON ECR

--- **NOTE** – if you want to push an image to ECR then I need to fallow proper tagging.



--- these are the commands; we will use IT to push docker image to AWS

--- NOTE – we need to tag the docker images like above.

 # Tag a docker image.

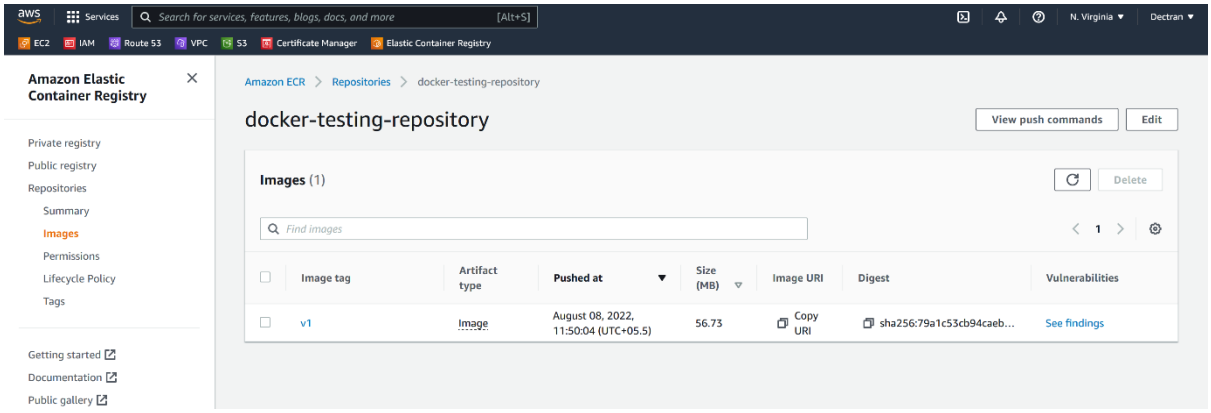
--- docker tag prabhu9652/customnginx1:v1 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v1

 # Login on instance

--- aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 743580576244.dkr.ecr.us-east-1.amazonaws.com

 # Push image to AMAZON ECR

--- docker push 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v1



--- **NOTE** – image version V1 is uploaded to amazon ECR.

--- **Scenario** – I have created a docker image from a project and hosted in an instance. Now I want to make some changes to some html file. This will tell you how I can create an image with new changes and host a docker container form that.

--- here I update html page with new changes.

 # Build a docker images with new changes.

--- docker build -t 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v2 .

 # Create a container form a new image.

--- docker run --rm -d --name nginx02 --publish 8100:80 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v2

Push image to Amazon ECR.

--- docker push 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v2

AWS ECS (Elastic container service)

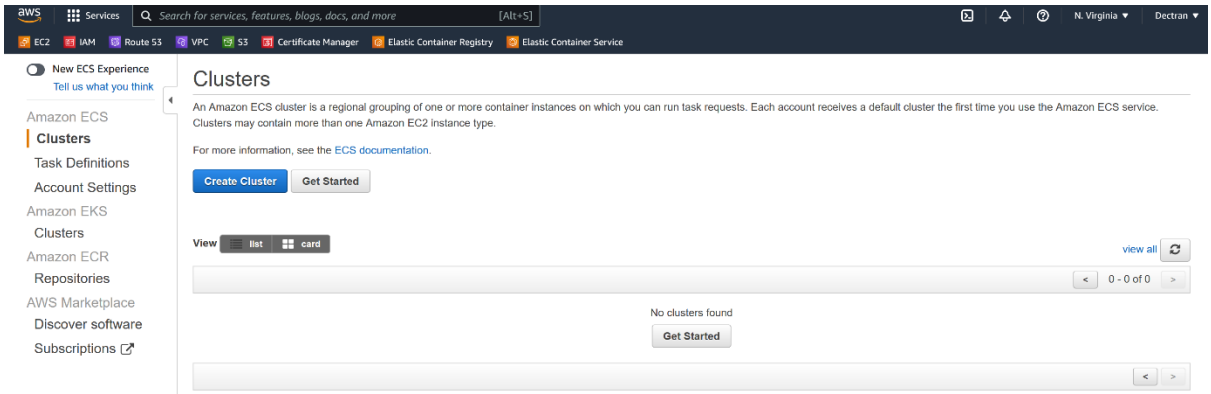
--- How to run a containers in AWS…?

1. Using EC2 instances, install docker/k8s and run the container.
2. AWS elastic container service(ECS)

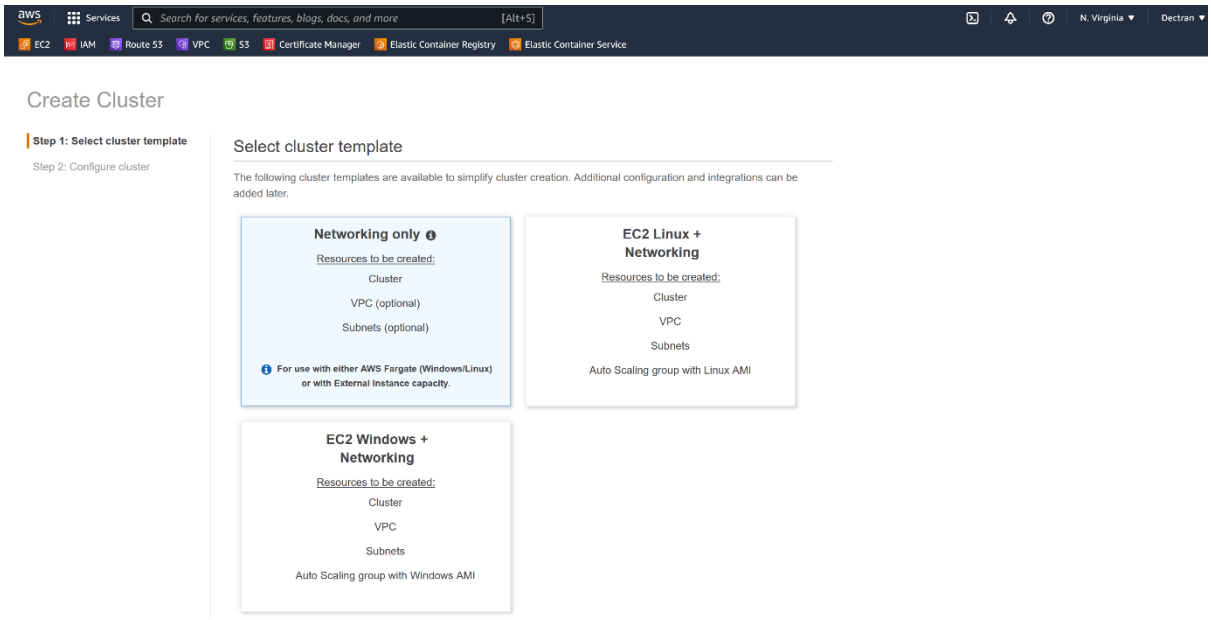
3. AWS container as a service – Fargate
4. AWS elastic kubernetes service

--- go to aws ECS.

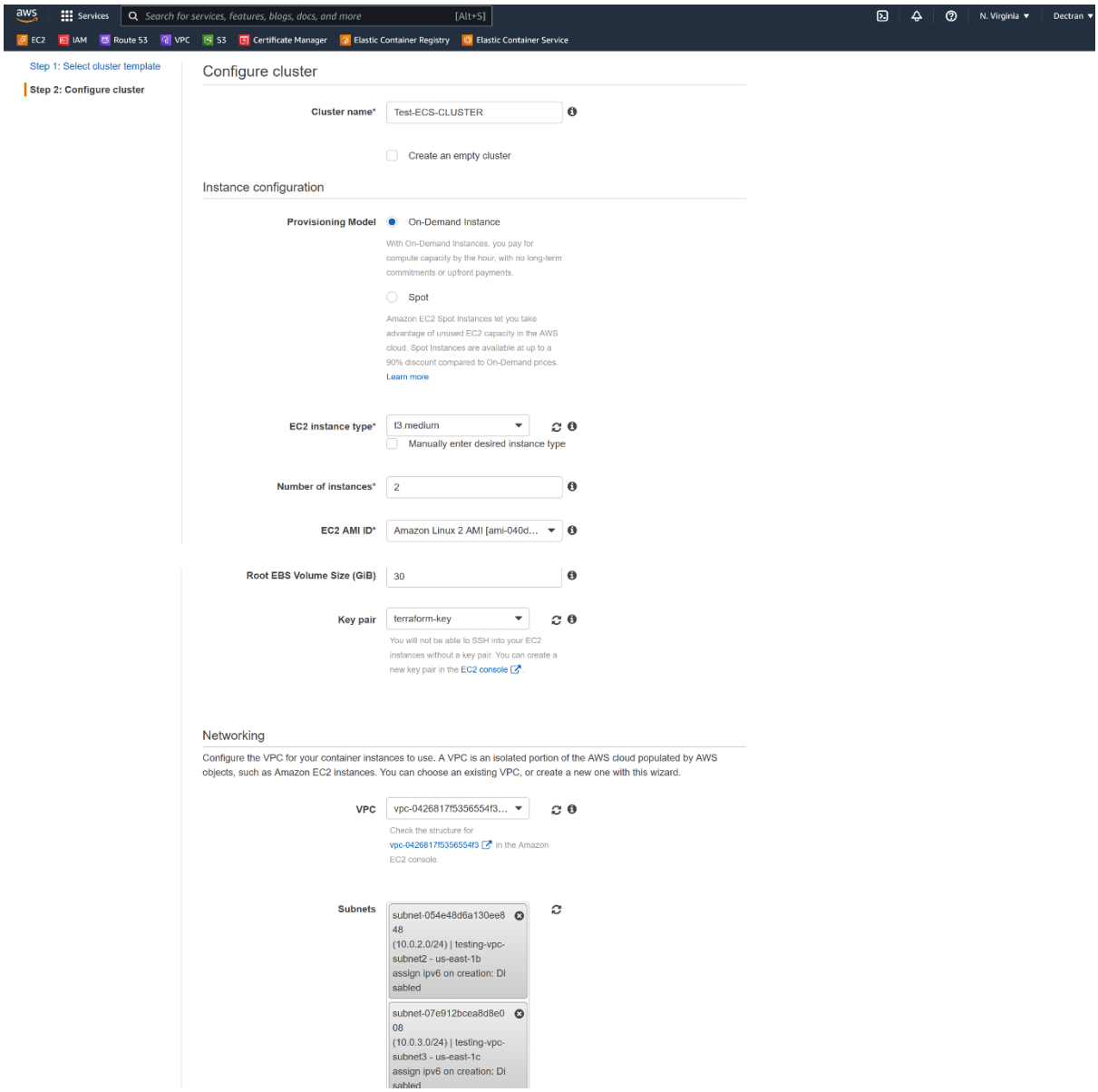
--- **Scenario** – 1st I will create 2 node cluster later, I will increate the cluster size to 3 node cluster.

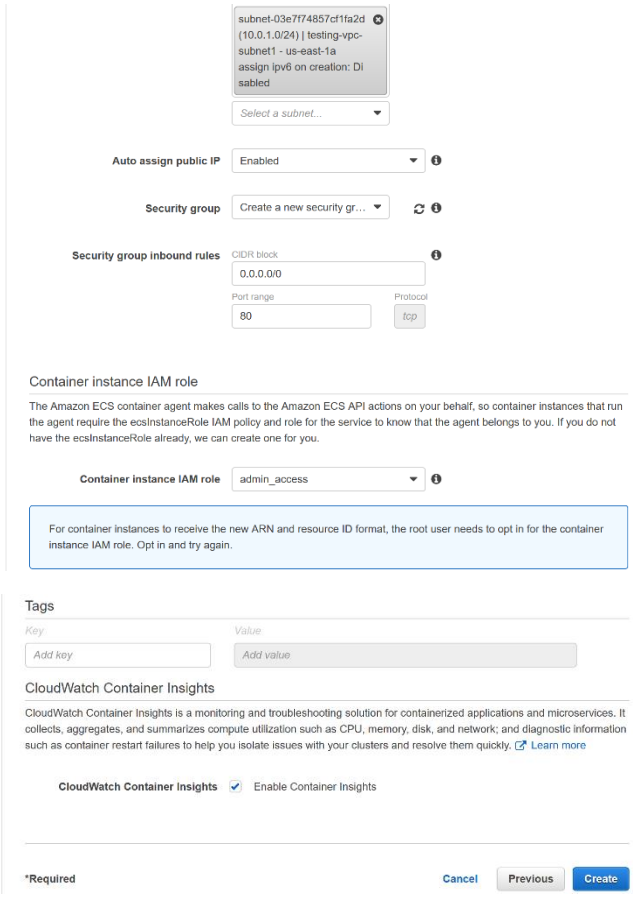


--- click on create cluster.

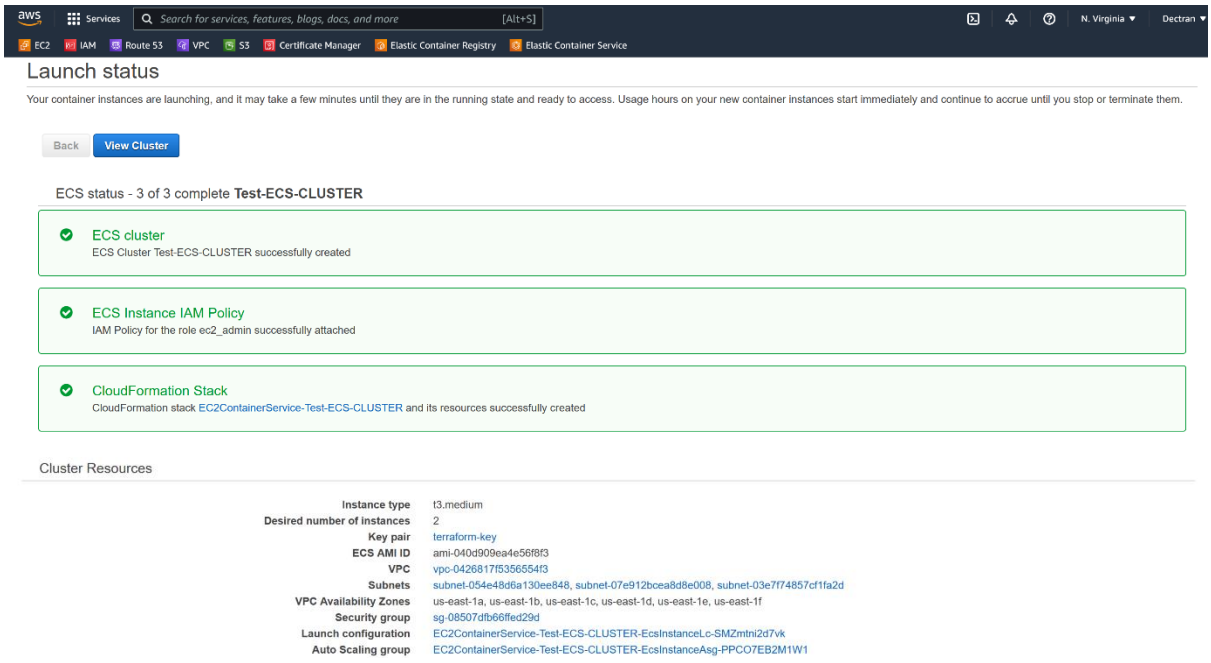


--- **note** – we can use fargate profiles and we can use amazon linux or windows for cluster creating. Here I am choosing amazon linux for cluster creating.





--- Click on create.

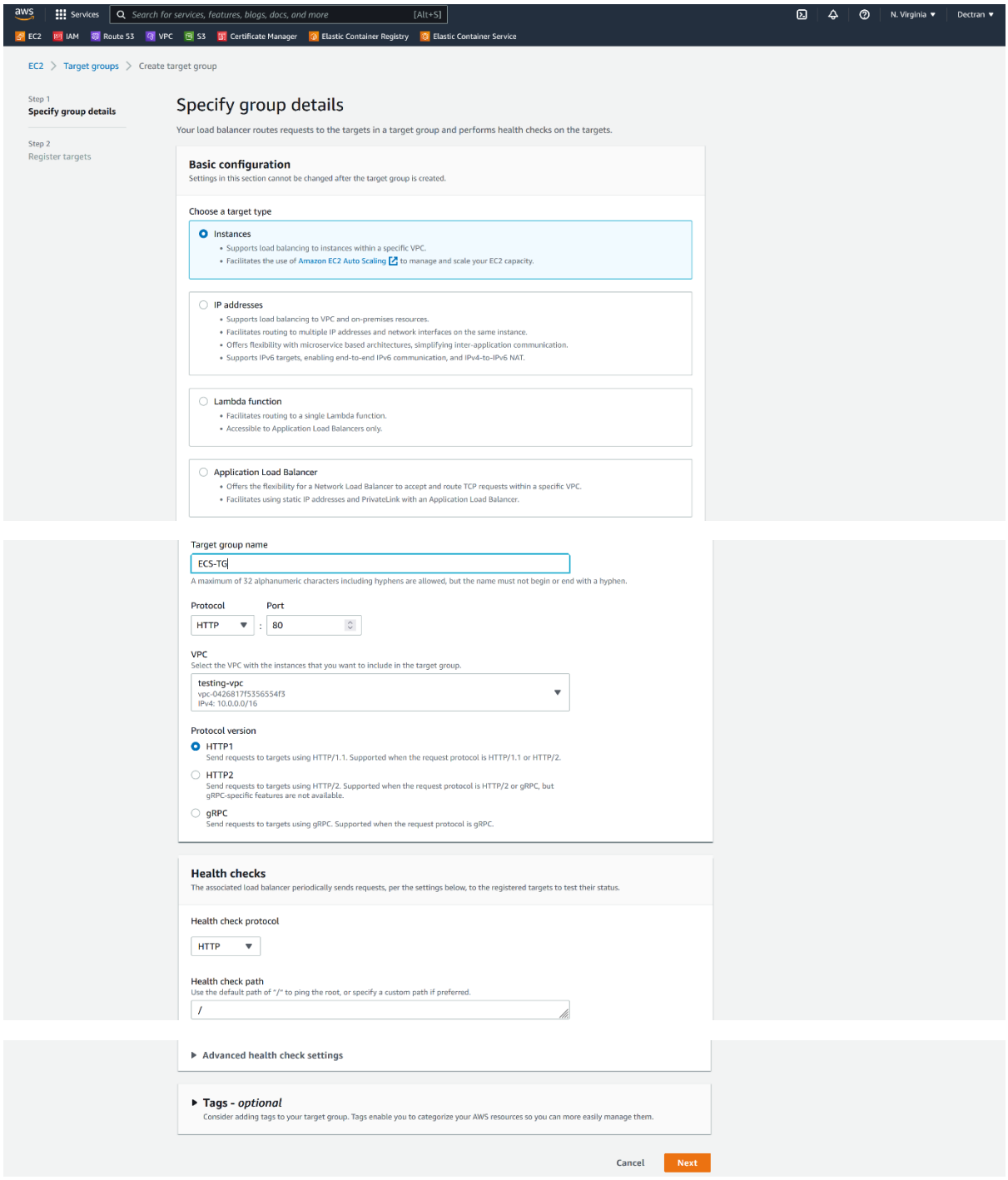


--- it will deploy a cloudformation template and create cluster.

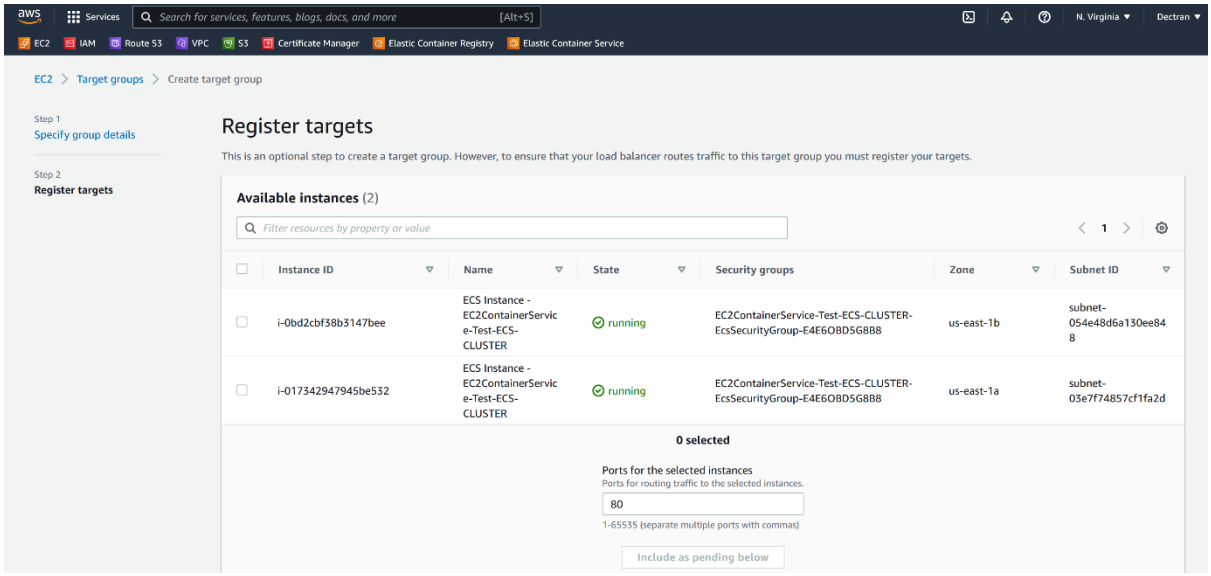
Create application load balancer

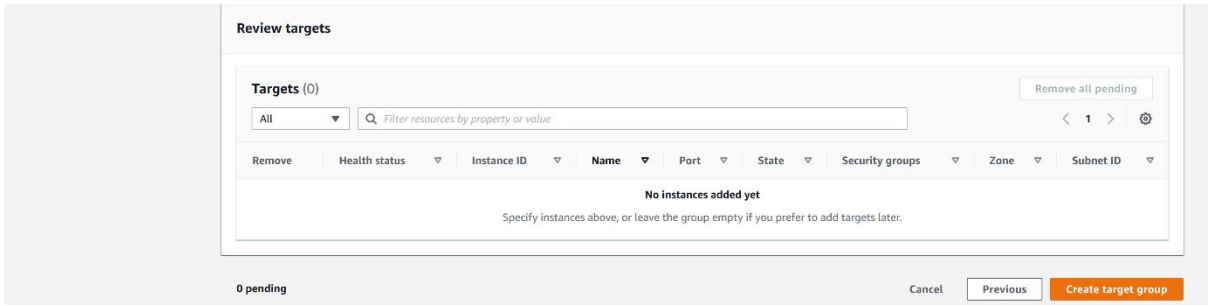
Target group creating

--- I will create target with out targets.



--- click on next.

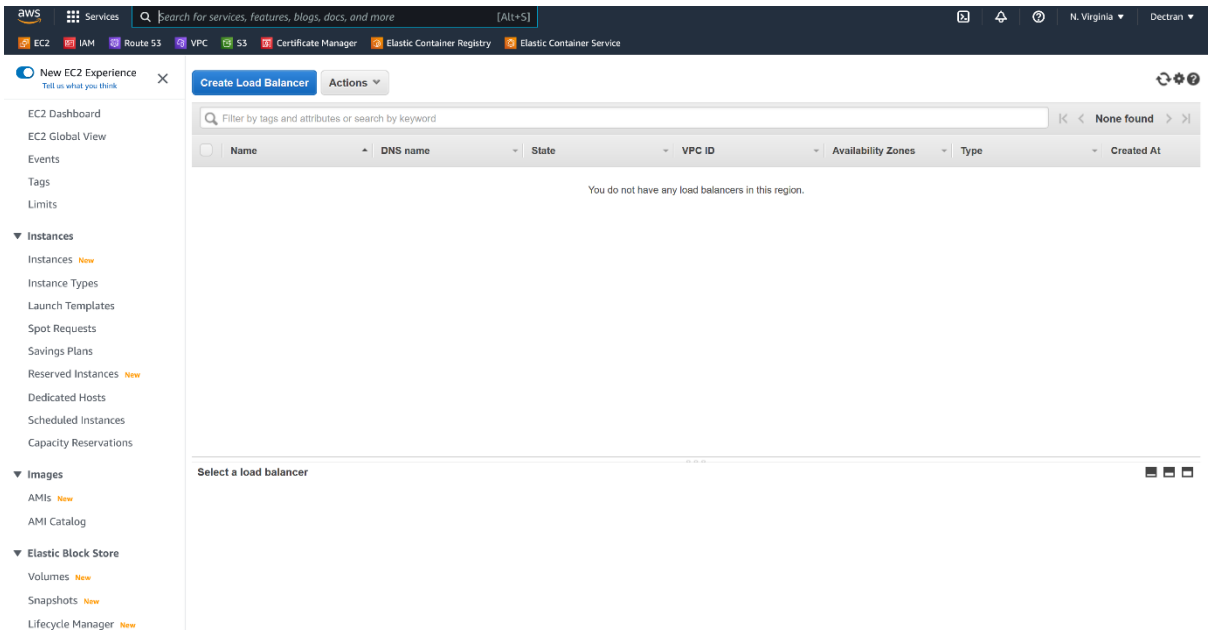




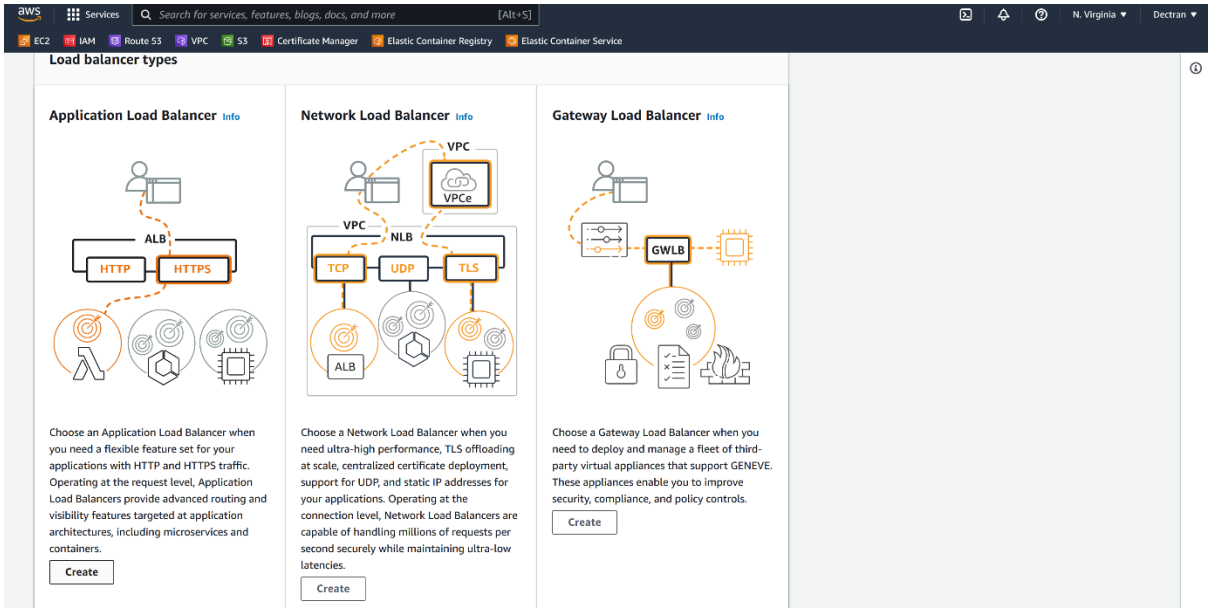
--- I will not add any targets here. Click on create target group.

Application load balancer creating

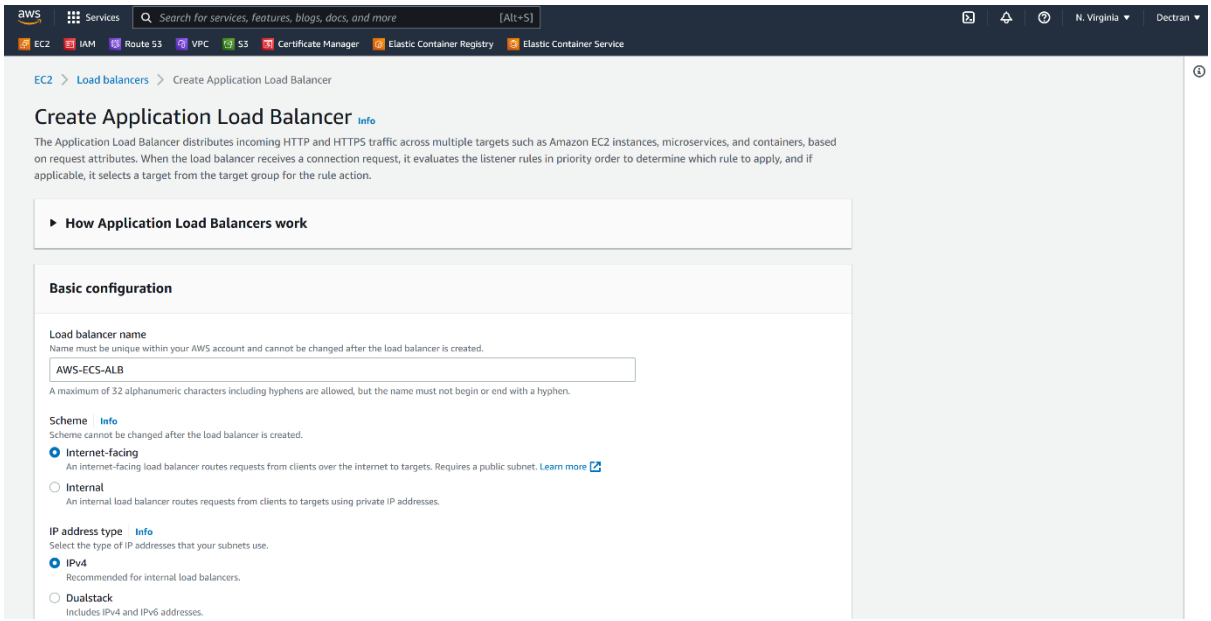
--- go to aws ec2 instance section.

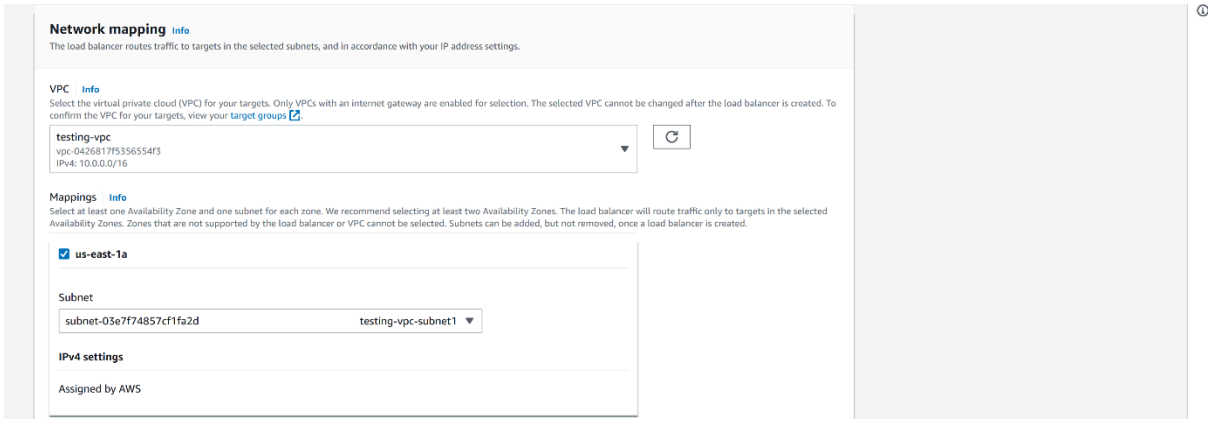


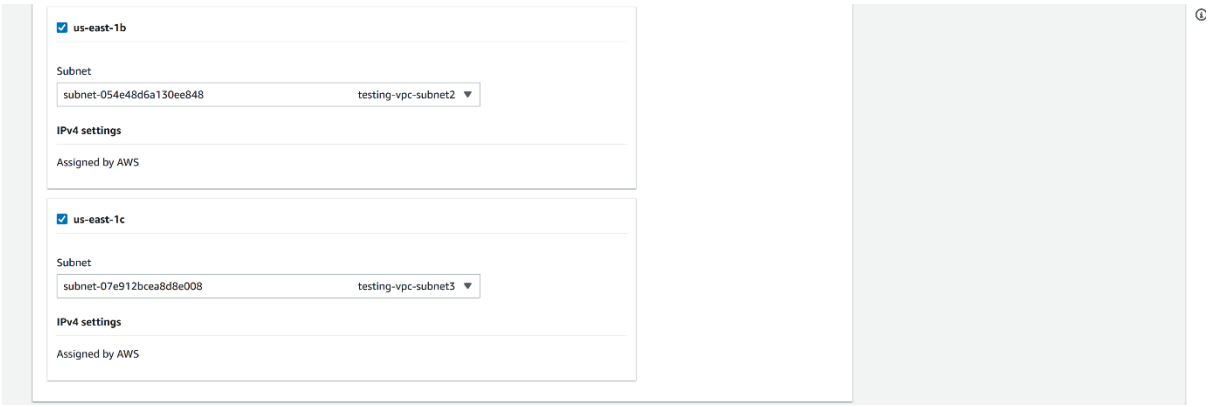
--- click on create load balancer.

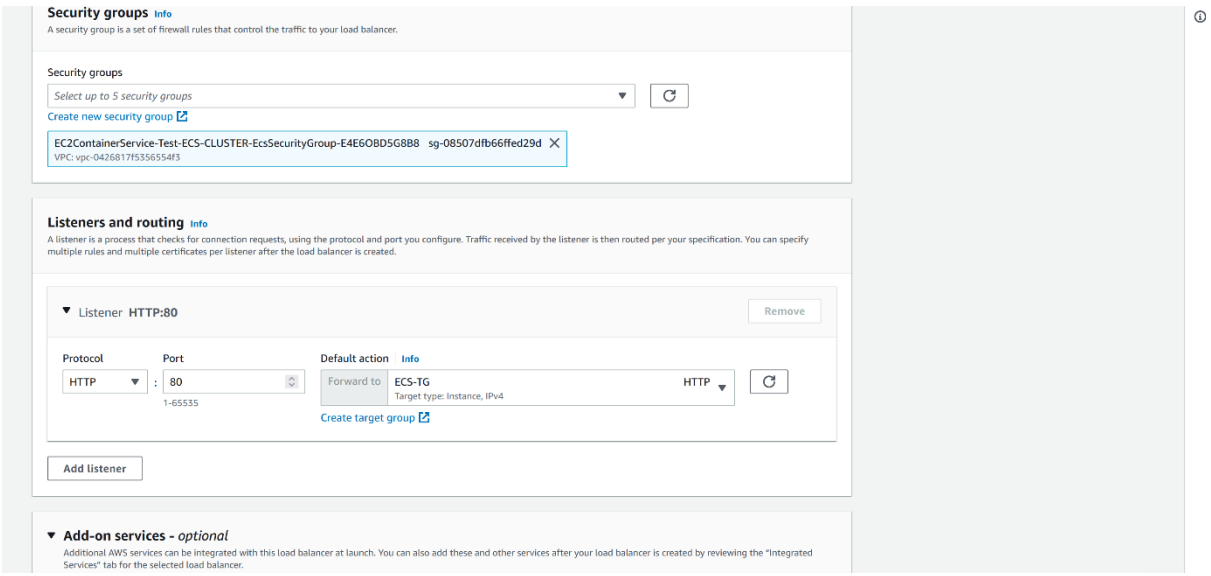


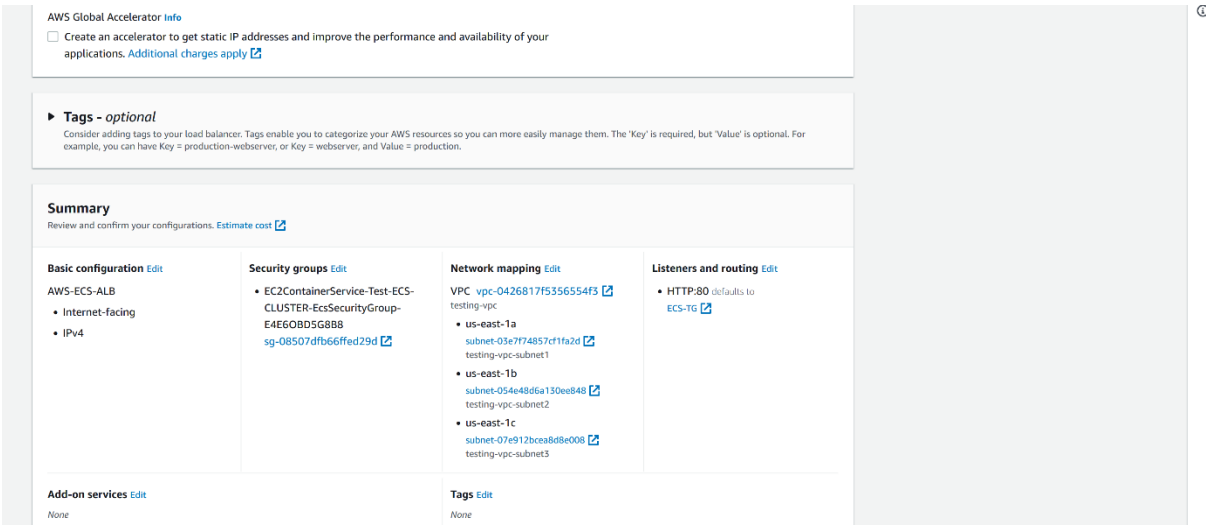
--- click on create application load balancer.

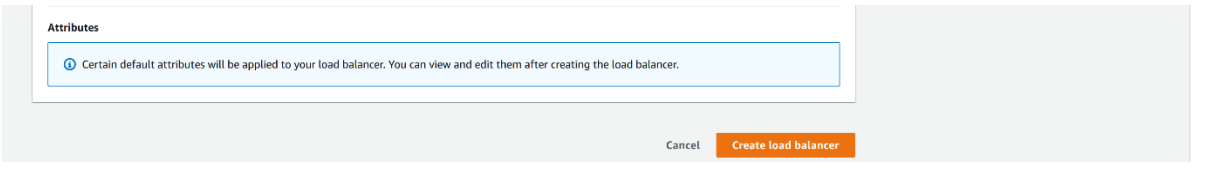








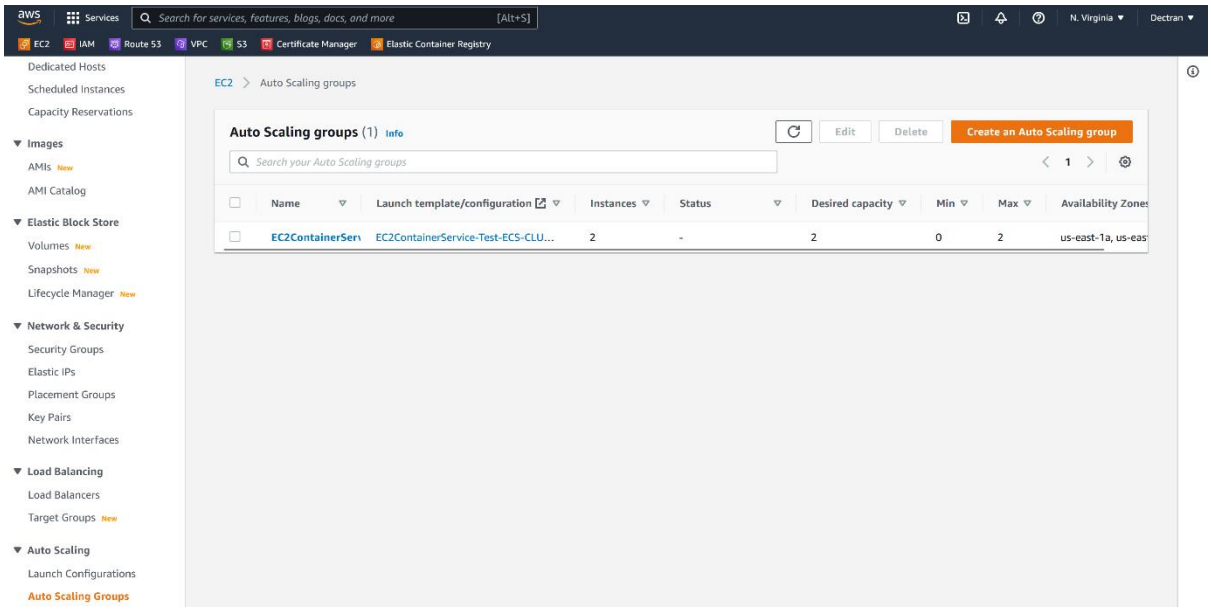




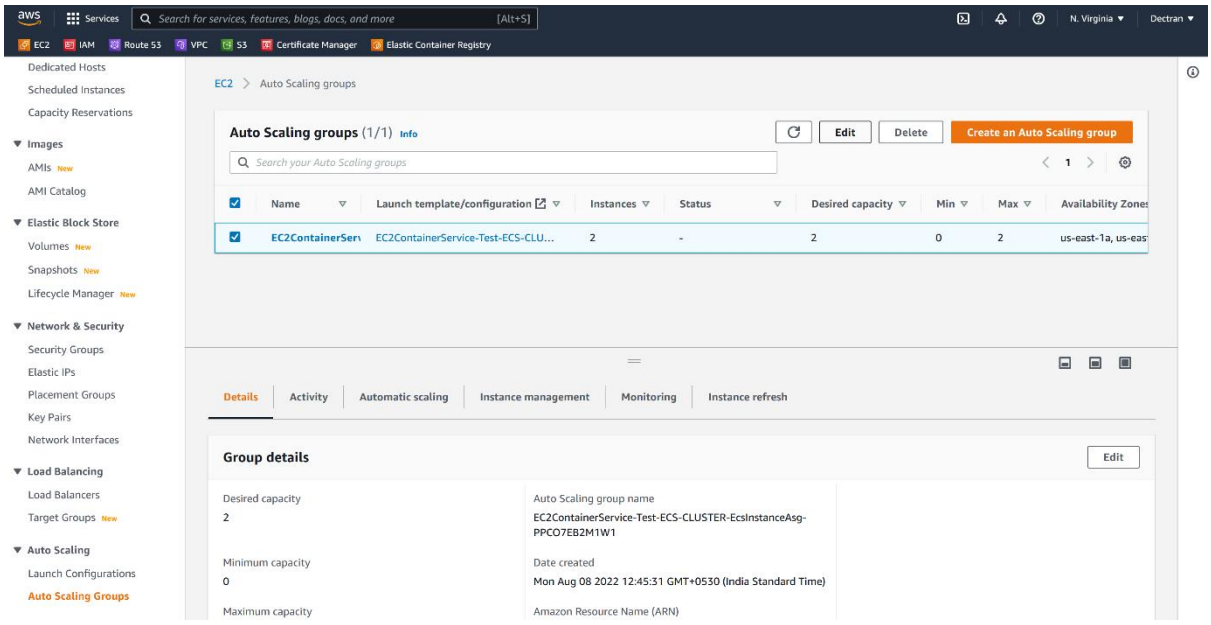
--- click on create load balancer.

Increase ECS 2 nodes cluster to 3 node cluster

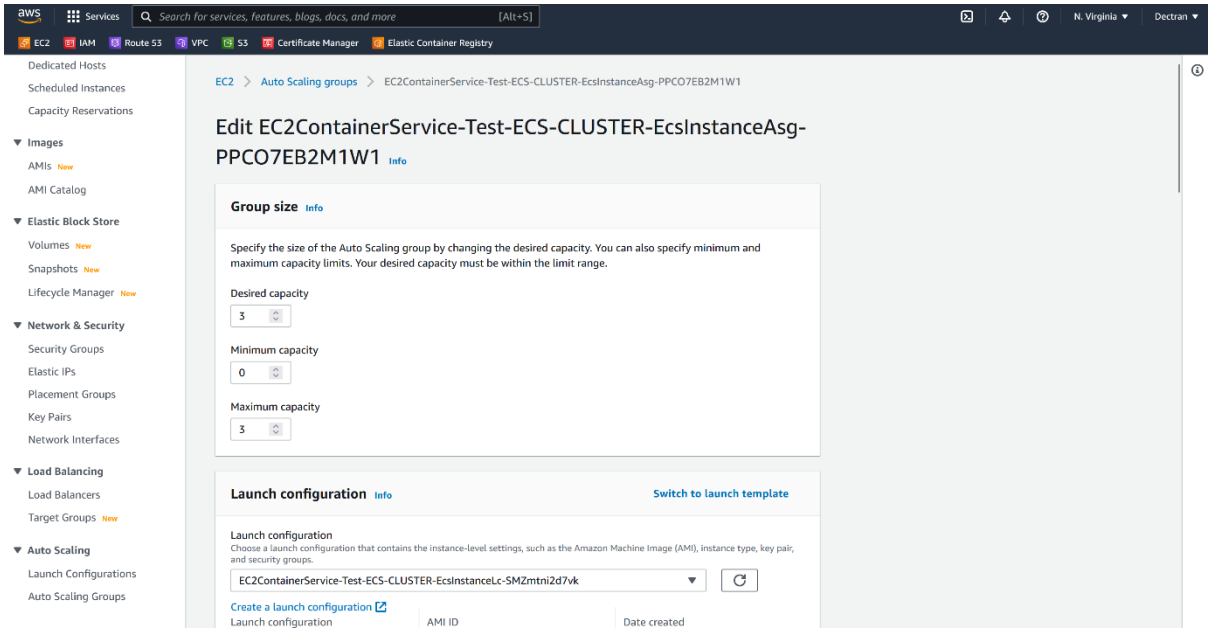
--- go to autoscaling group.



--- click on auto scaling groups.

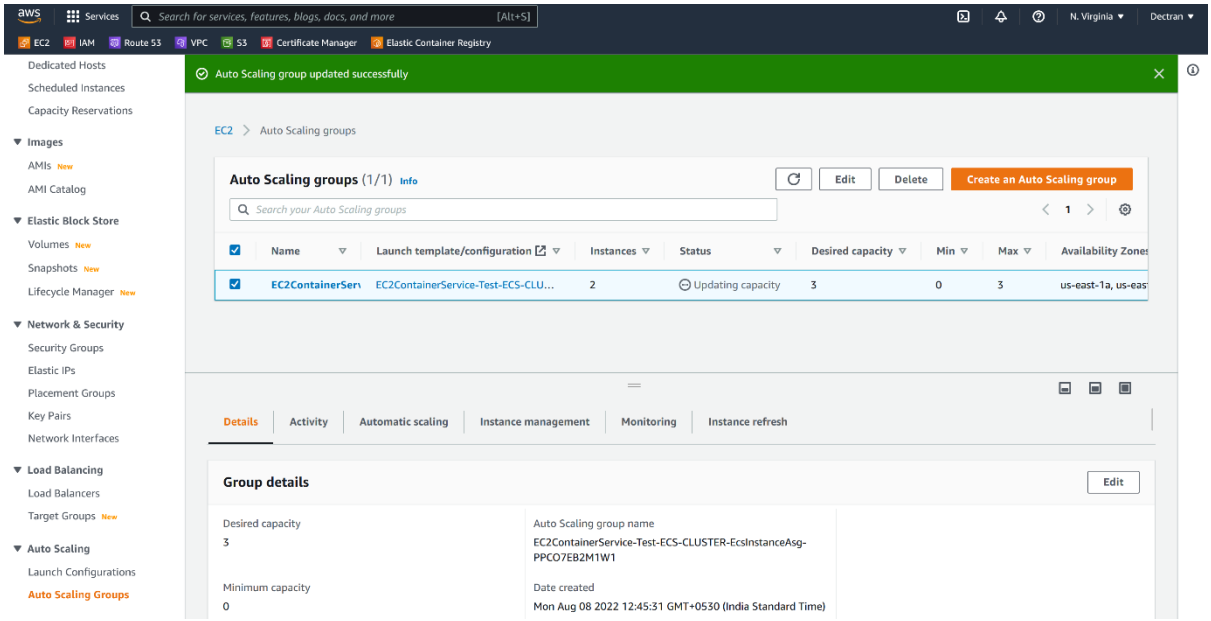


--- select autoscaling group and click on edit option.



--- increate the desired capacity and maximum capacity to 3.

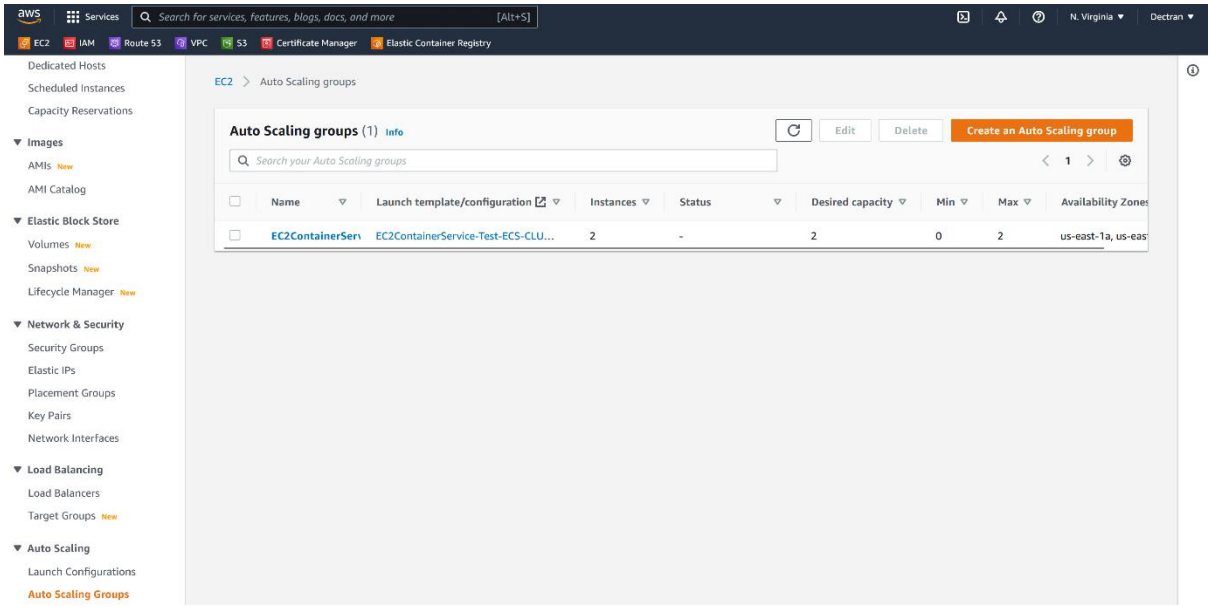
--- click on update.



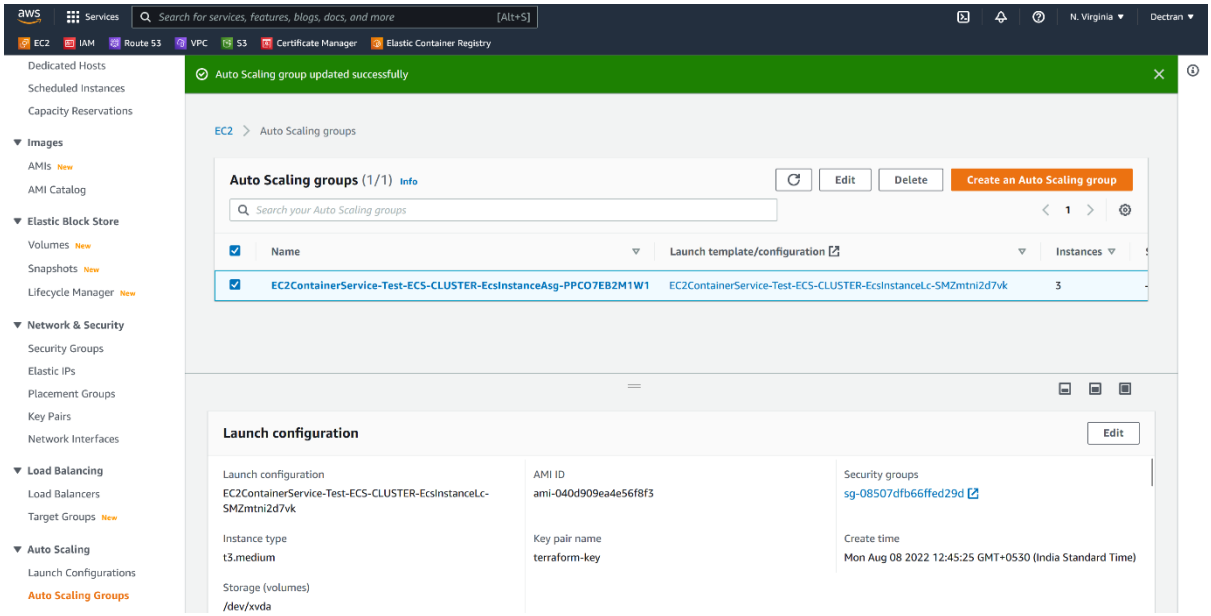
--- **note** – now it is updating the capacity.

How to update our cluster security group

--- go to ECS cluster autoscaling group.



--- select that autoscaling group and under details section, you will find the security group select it and update it.



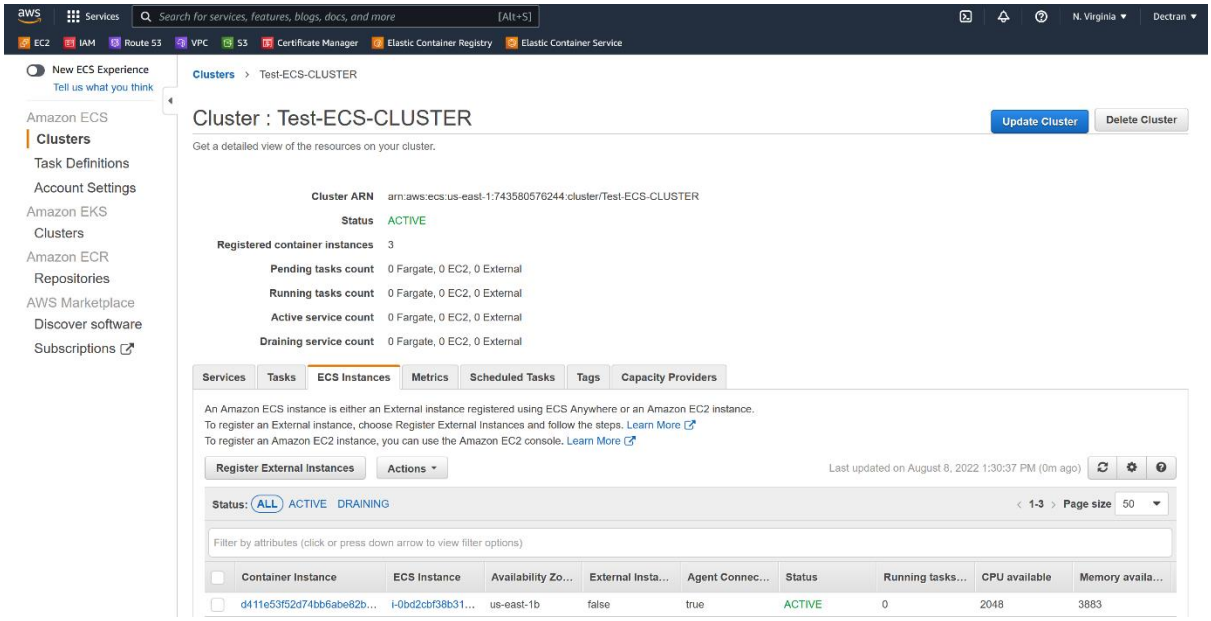
--- click on security group.

--- update the security group with required information. Here I am testing so, I allowed traffic form anywhere.

AWS ECS task definition

--- Reference - https://github.com/aws-samples/aws-containers-task-definitions/tree/master/nginx

--- go to ECS.



--- click on ECS task definition.

--- **scenario** – when you want to deploy container on docker host, we will give port, image, tag details in a command right. Something like that, when we want to create container ECS cluster we will mention all those details in Task Definition like how much cpu, memory, you want to consume.

--- **note** - aws ECS nginx task definition github – search by this name on aws console. (https://github.com/aws-samples/aws-containers-task-definitions/tree/master/nginx)

--- **nginx_ec2.json**

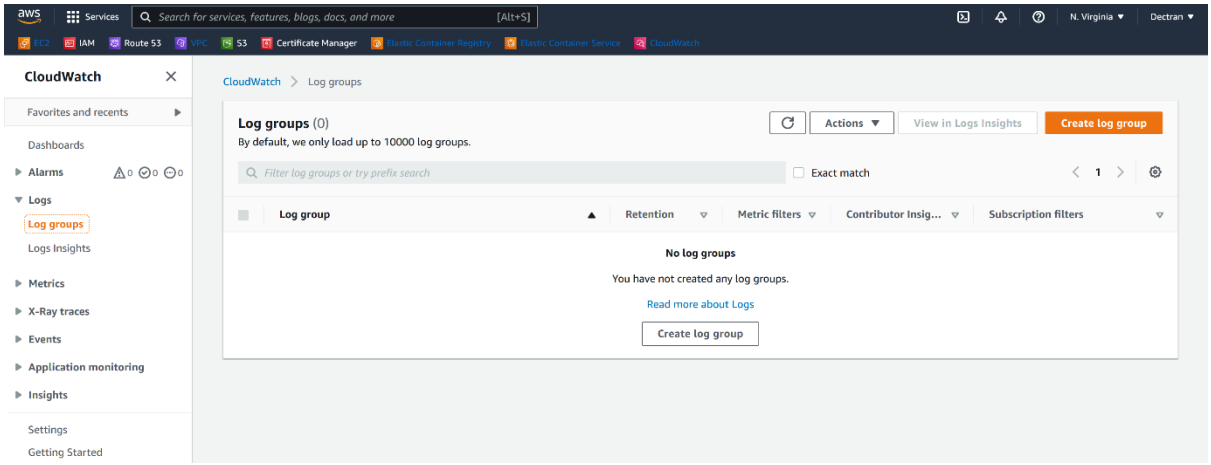
```
{
  "requiresCompatibilities": [
    "EC2"
  ],
  "containerDefinitions": [
    {
      "name": "nginx", # if changes this name then please change below.
      "image": "743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-
repository:v2",

      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "logConfiguration": {
          "logDriver": "awslogs",
          "options": {
```

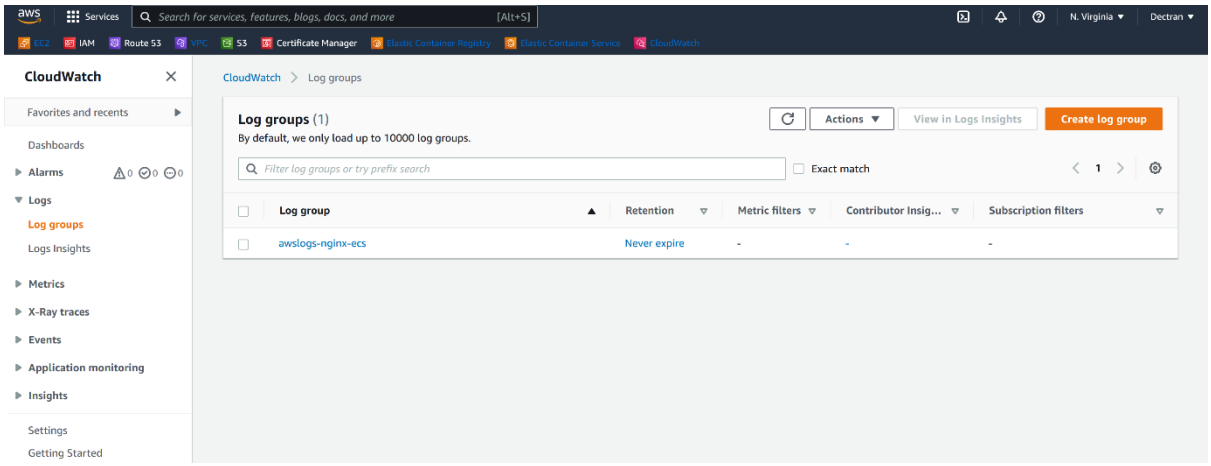
```
            "awslogs-group": "awslogs-nginx-ecs", # create a cloudwatch log
group by this name.
            "awslogs-region": "us-east-1",
            "awslogs-stream-prefix": "nginx" # change here
         }
      }
   }
  ],
  "volumes": [],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx" # change here.
}
```

Cloud watch log group creating

--- go to the aws cloudwatch

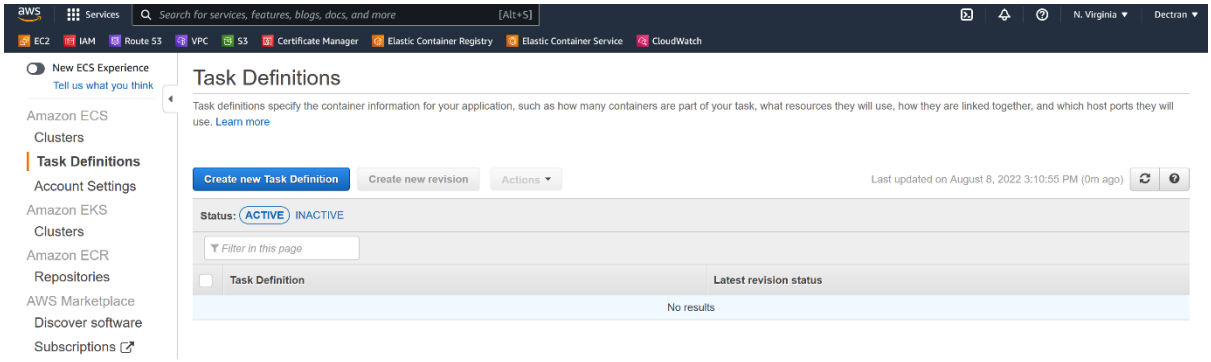


--- click on log groups and create log group.

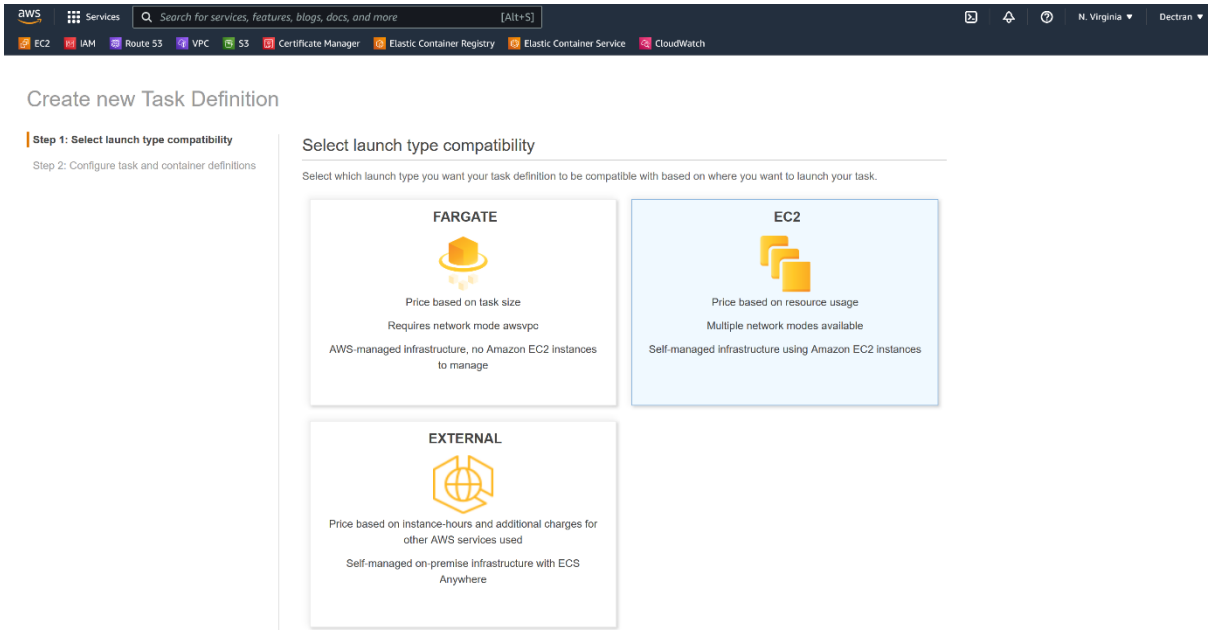


--- log groups got created.

--- please click on task definition.

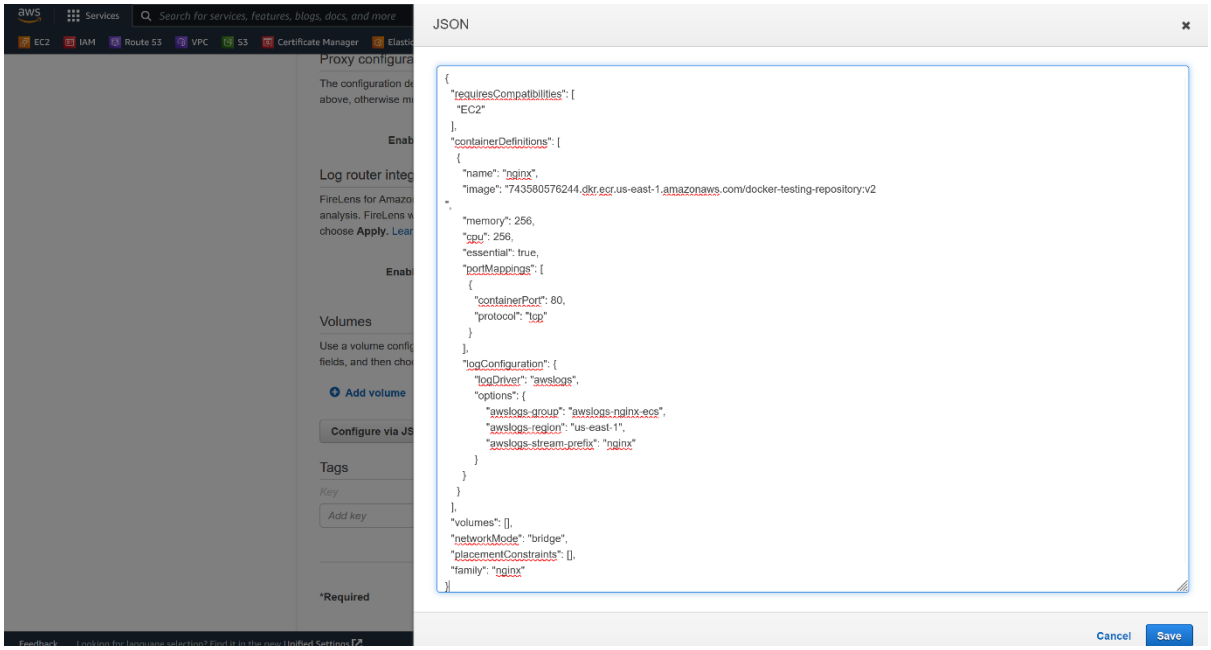


--- click on create new task definition.



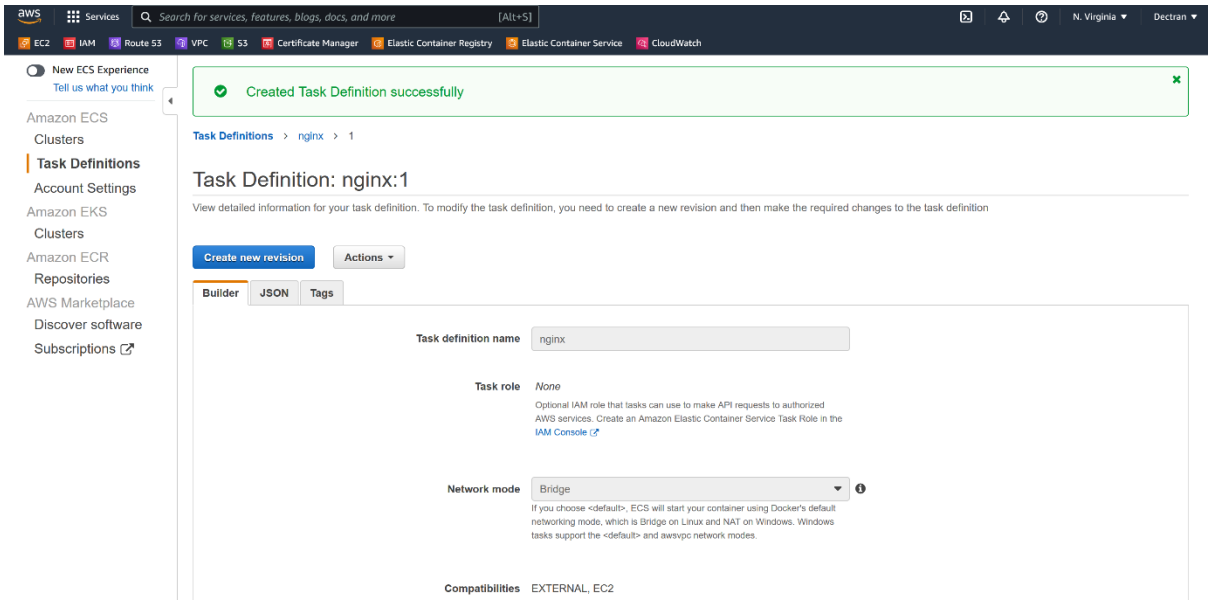
--- please select EC2 and click on next.

--- scroll down and click on configure via json.



--- click on save.

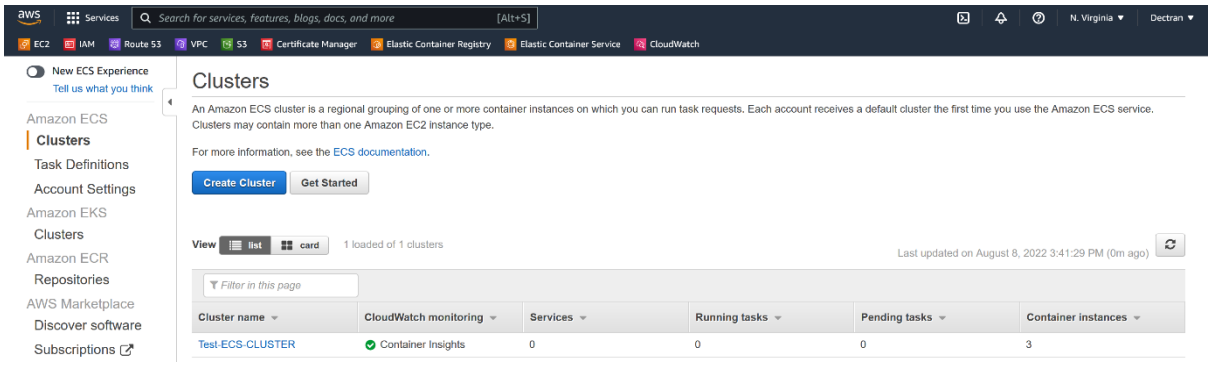
--- everything will be autoloaded from the json file, check before clicking on create.



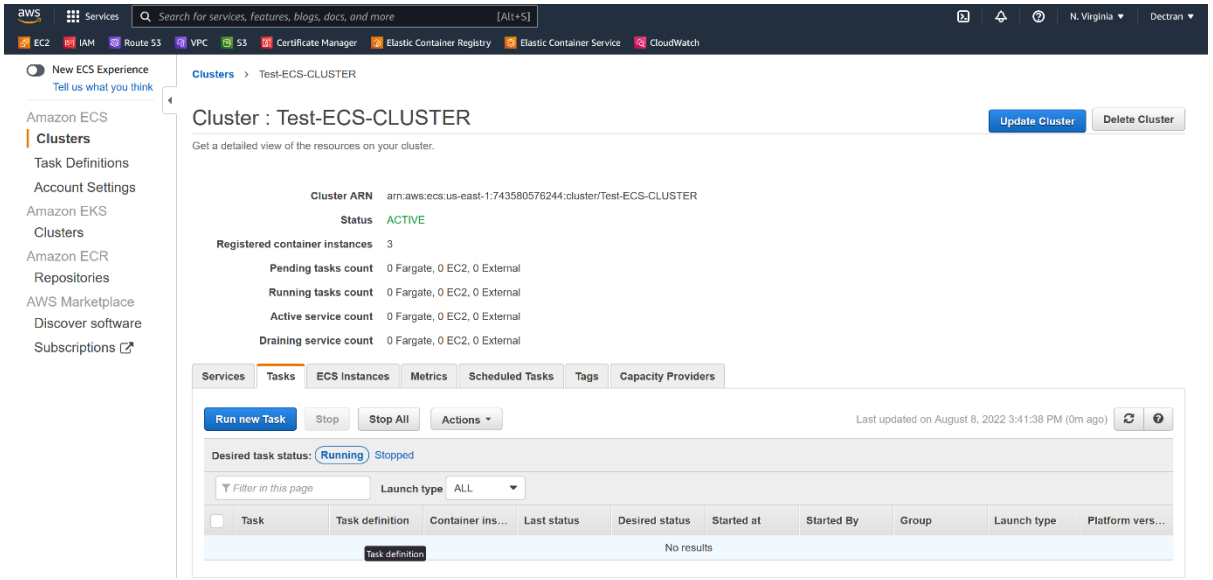
--- note – now the task nginx:1 got created.

Deploy a container using task definition

--- **note** – here I will deploy our container using task definition.

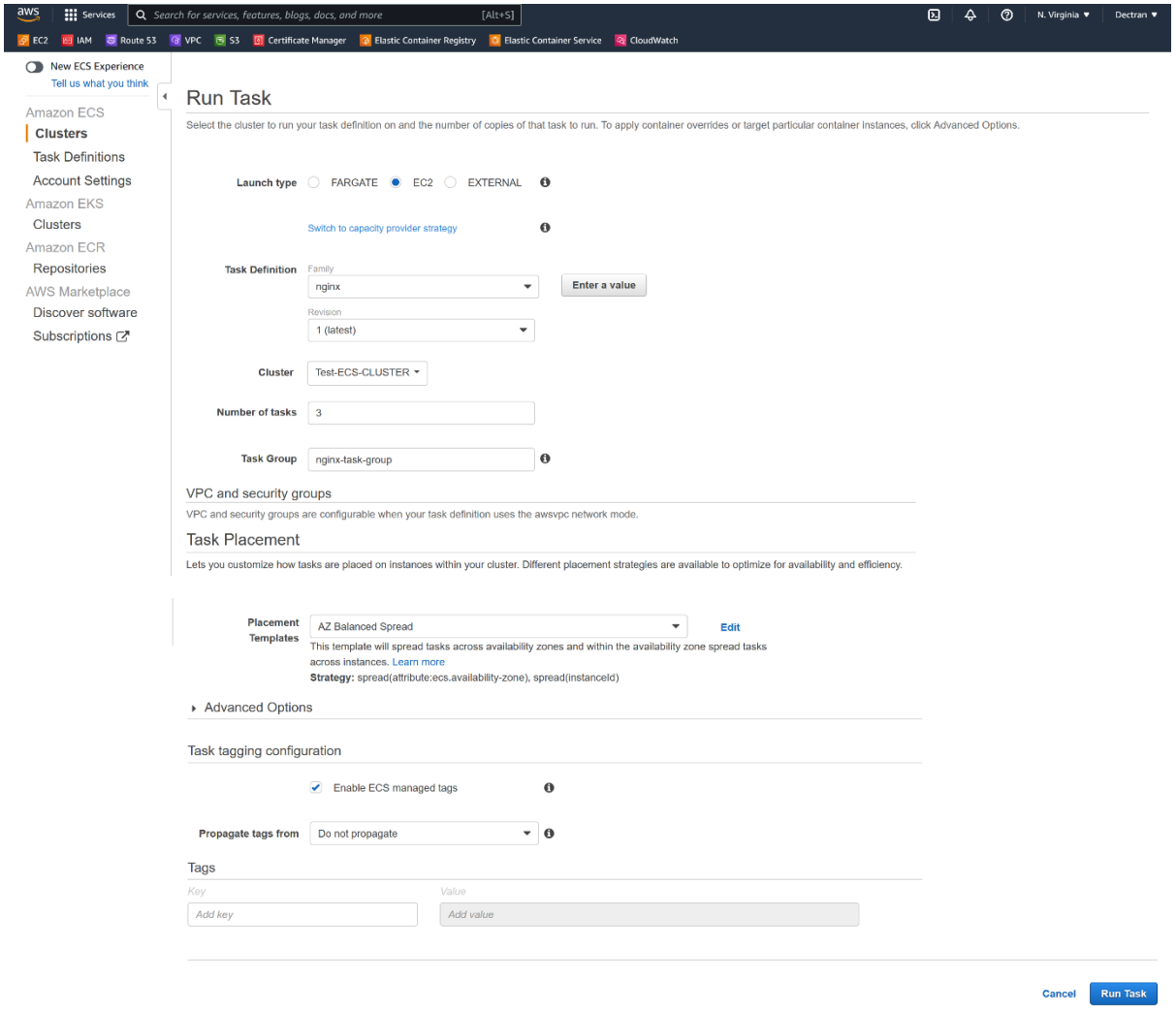


--- click on our cluster.



--- click on Run New task.

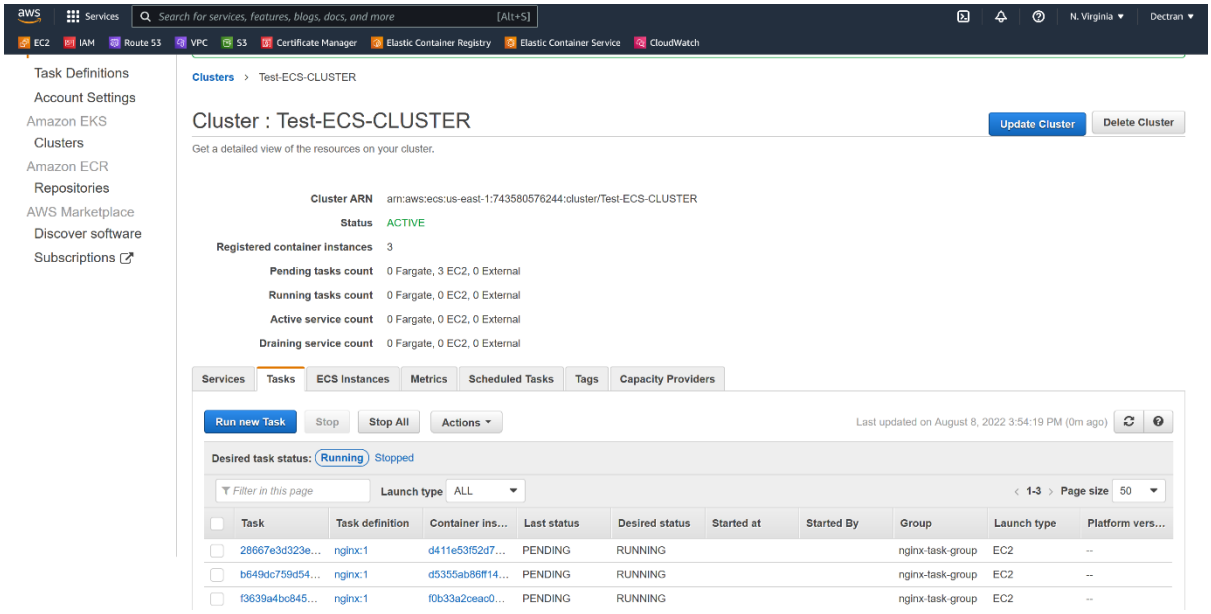
--- **Number of tasks = 3** (it is nothing but no of containers you want to deploy).



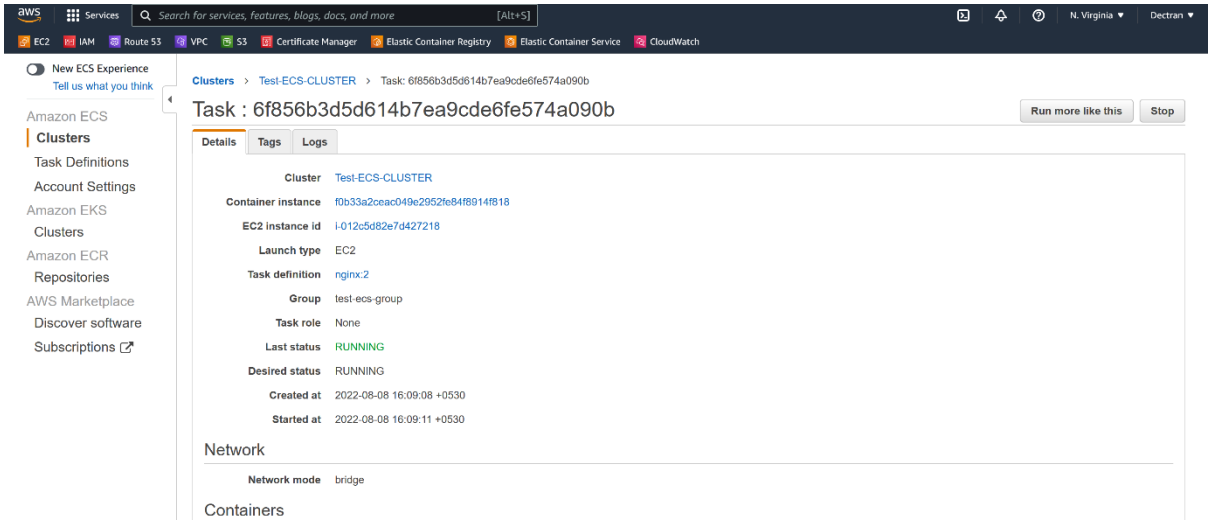
--- click on Run Task.

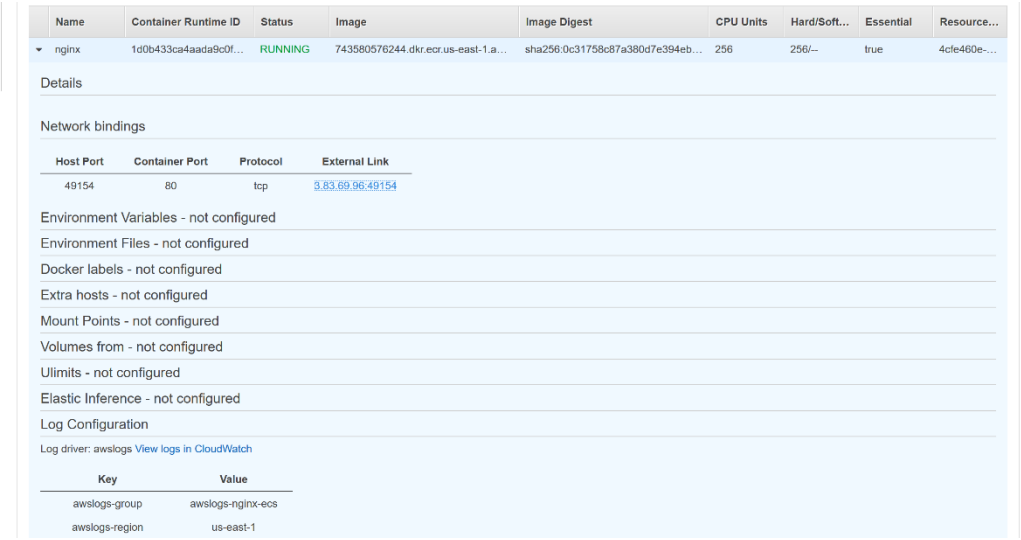
Application testing

--- cluster -> Run a task.



--- under task, click on any task id.



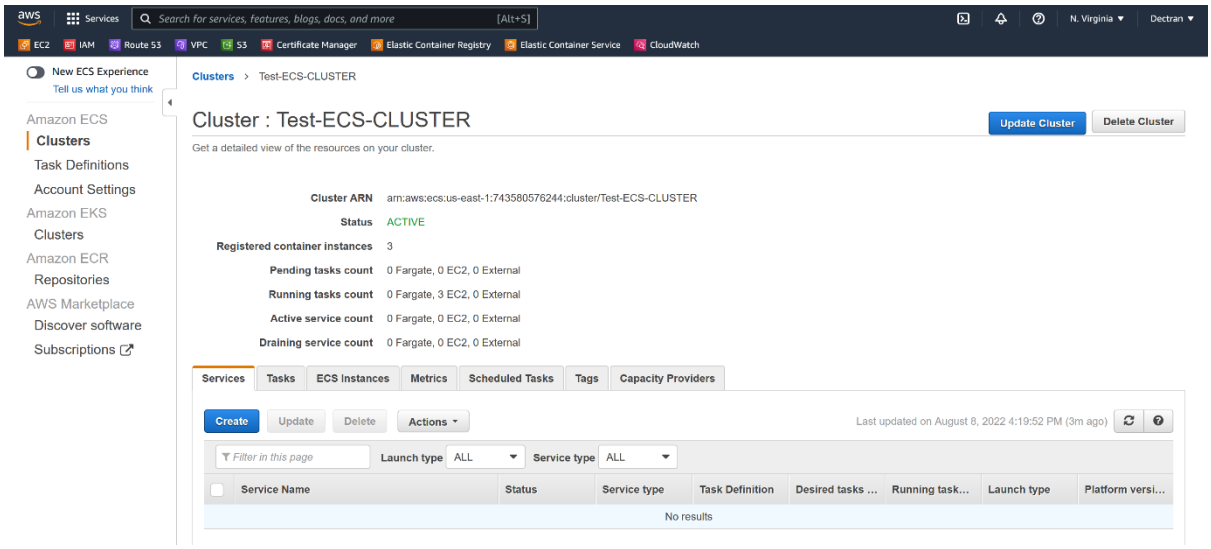


--- click on the external link and you will be able access the application.

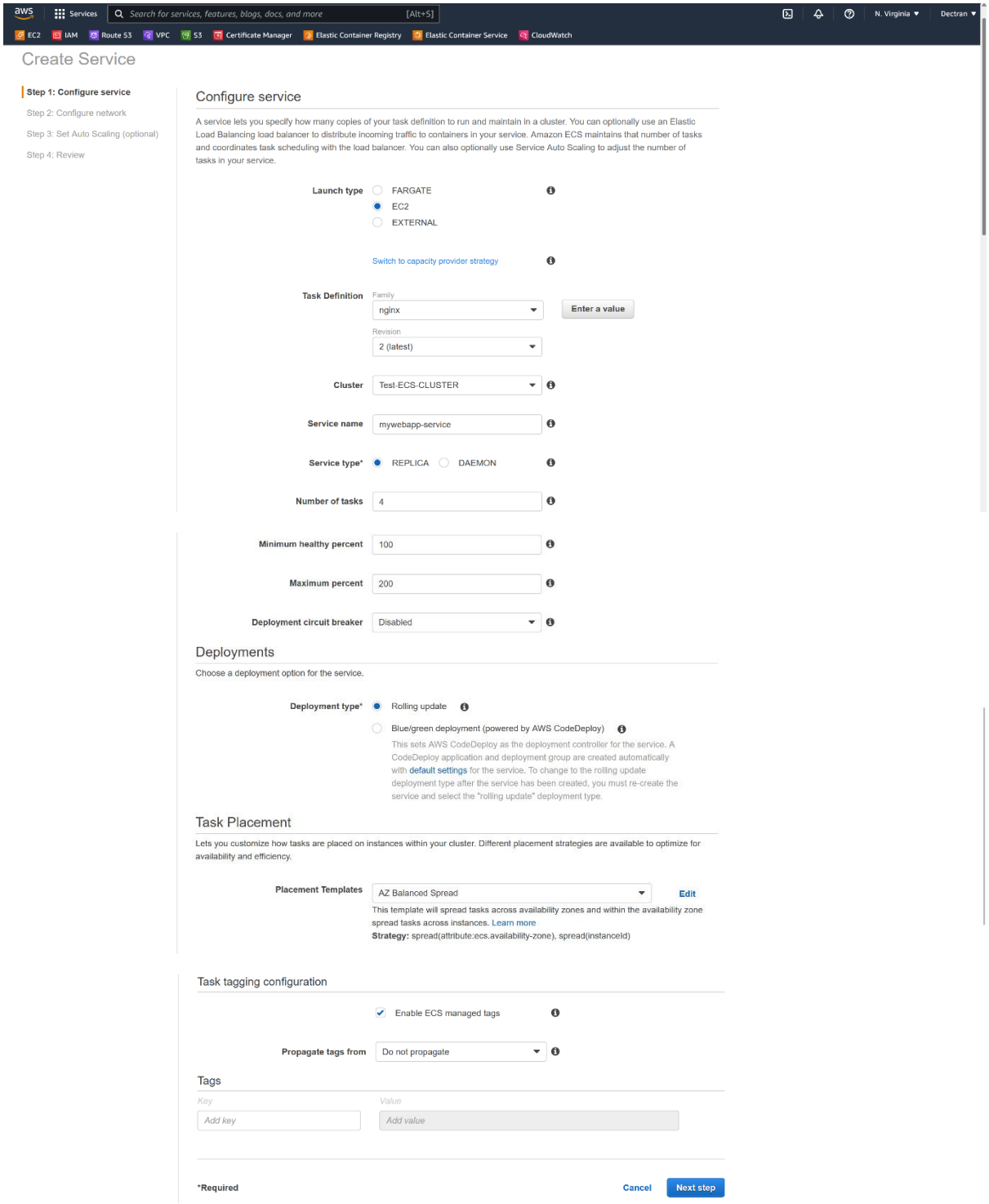
Services

--- ECS -> cluster -> click on our cluster.

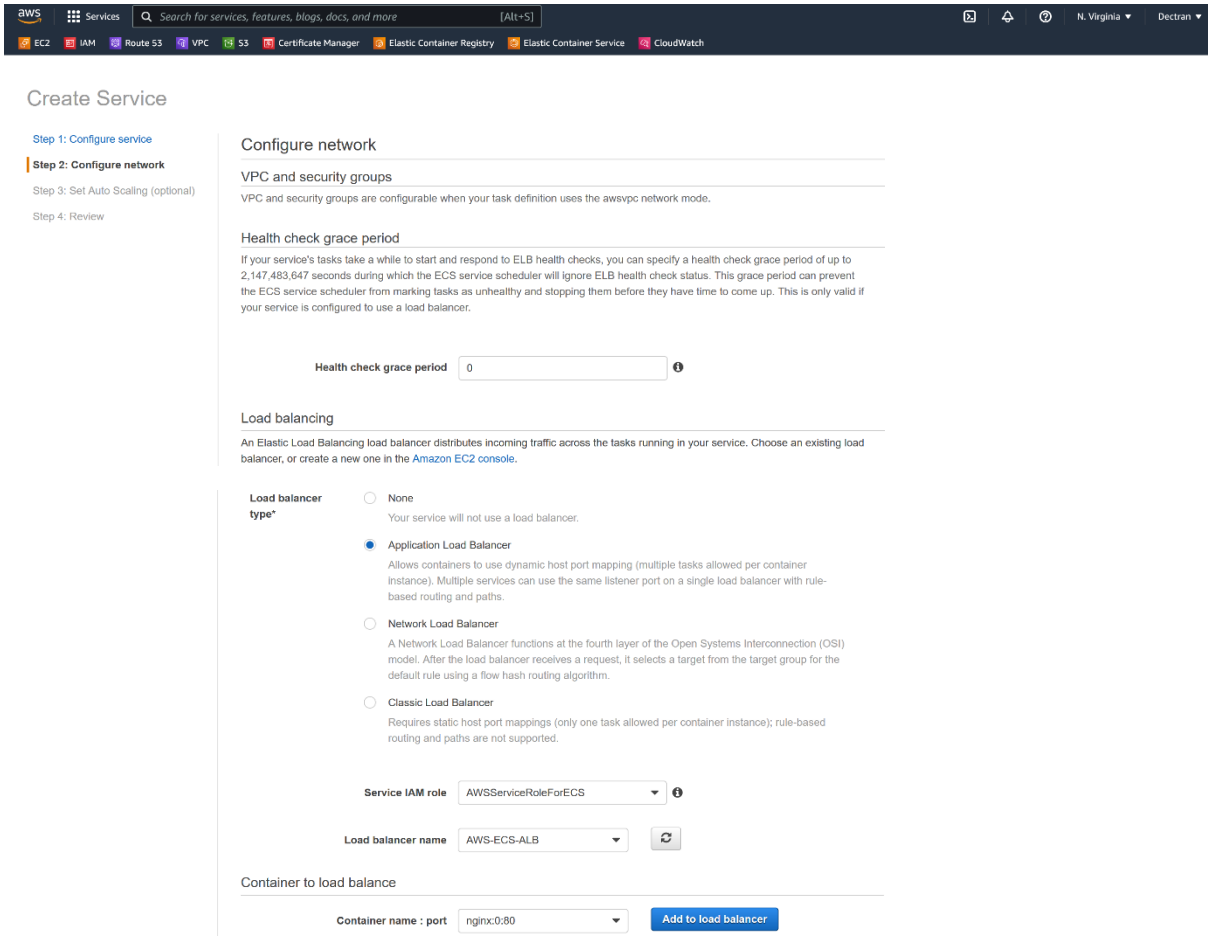
--- **Scenario** - service will recreate the container, due to some reason it stopped working.



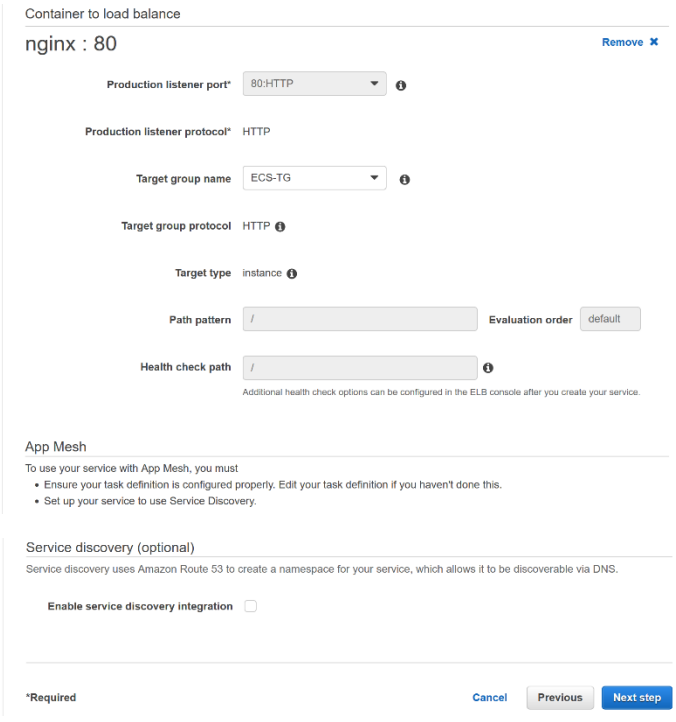
--- click on create.



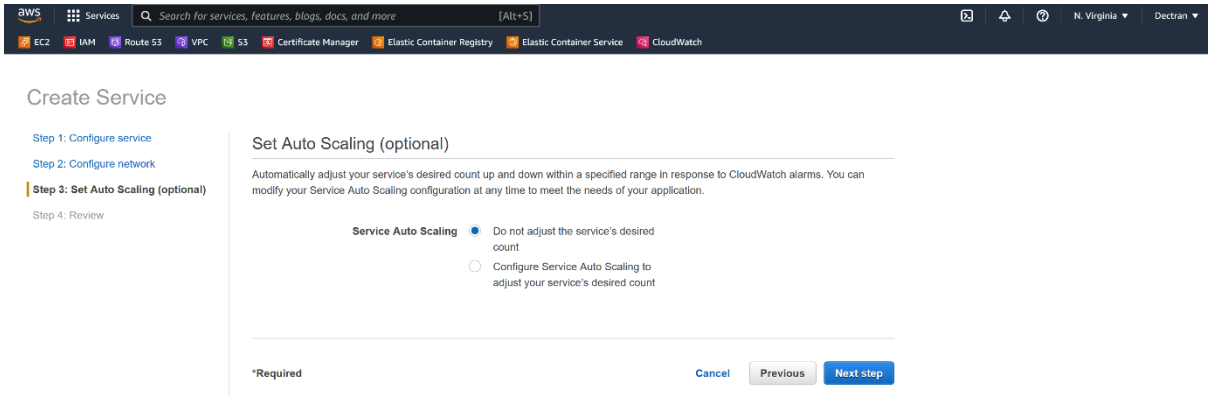
--- click on next.



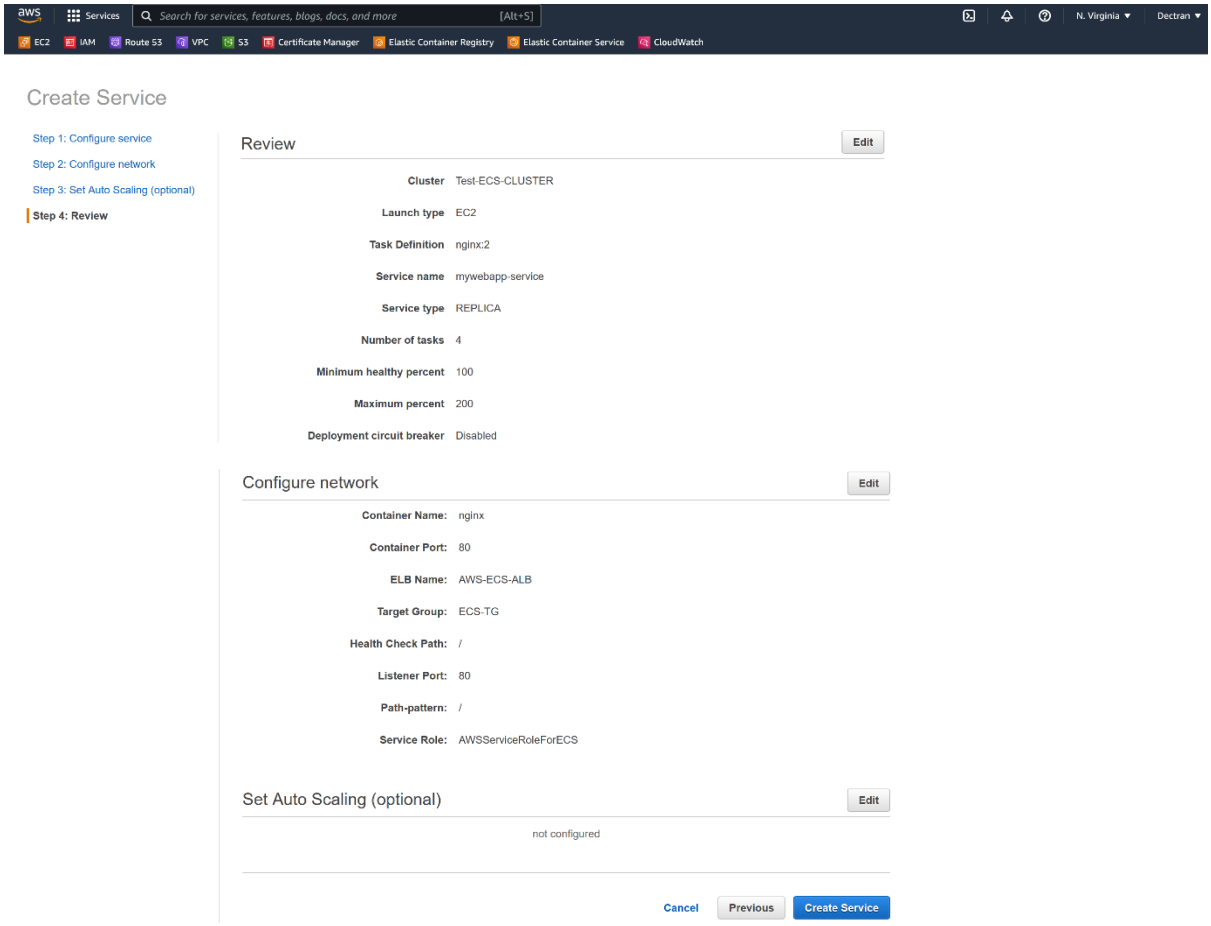
--- click on add to load balancer.



--- click on next.



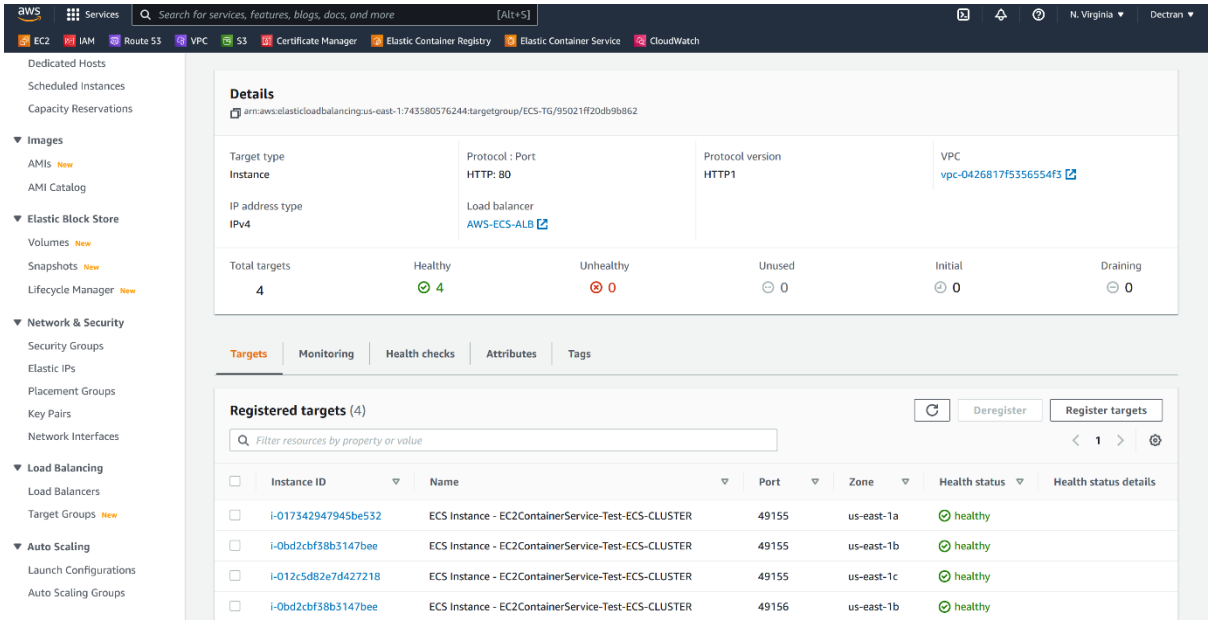
--- click on next step.



--- click on create a service.

Check the load balancer target group

--- **note** – previously, when I am creating target group, I did not add any target to the target groups. But now while creating service in ECS, it should add containers as targets in load balancer.



--- the containers added as targets in target groups.

--- go to the load balancer and copy the load balancer dns name and test it on google.

Rolling update on ECS cluster

 # New image building

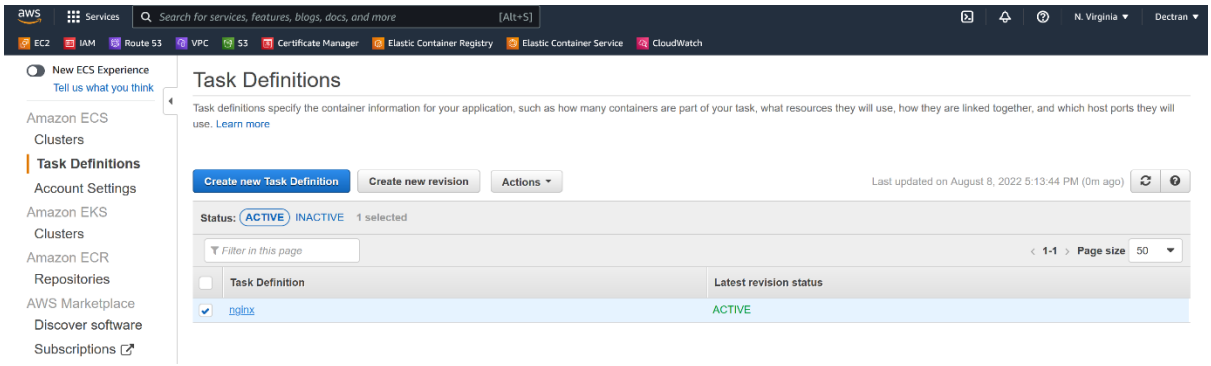
--- docker build -t 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v3 .

 # Push new image

--- docker push 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v3

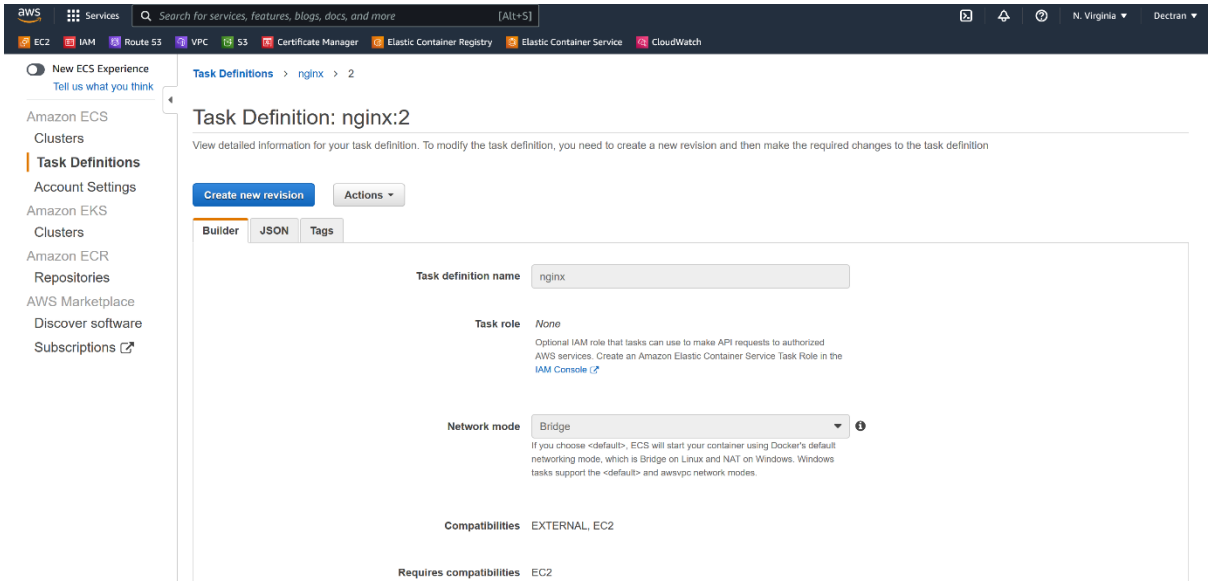
New task definition version creating

--- **note** – here I will create a new version of task definition and the new version task definition will be used in service.



--- click on our task definition.

--- edit our task definition json file.



--- click on create a new version.

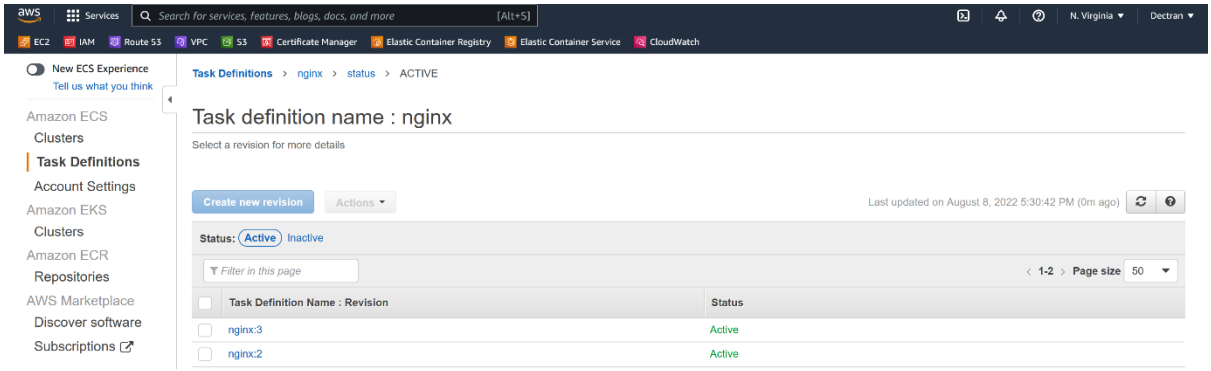
--- after clicking on **create new version** and scroll down.

--- click on configure via json.

```
{
  "requiresCompatibilities": [
    "EC2"
  ],
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-
repository:v3",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "logConfiguration": {
          "logDriver": "awslogs",
          "options": {
              "awslogs-group": "awslogs-nginx-ecs",
              "awslogs-region": "us-east-1",
              "awslogs-stream-prefix": "nginx"
          }
      }
    }
  ],
```

```
  "volumes": [],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx"
}
```

--- **note** – after copying this file, click on create.

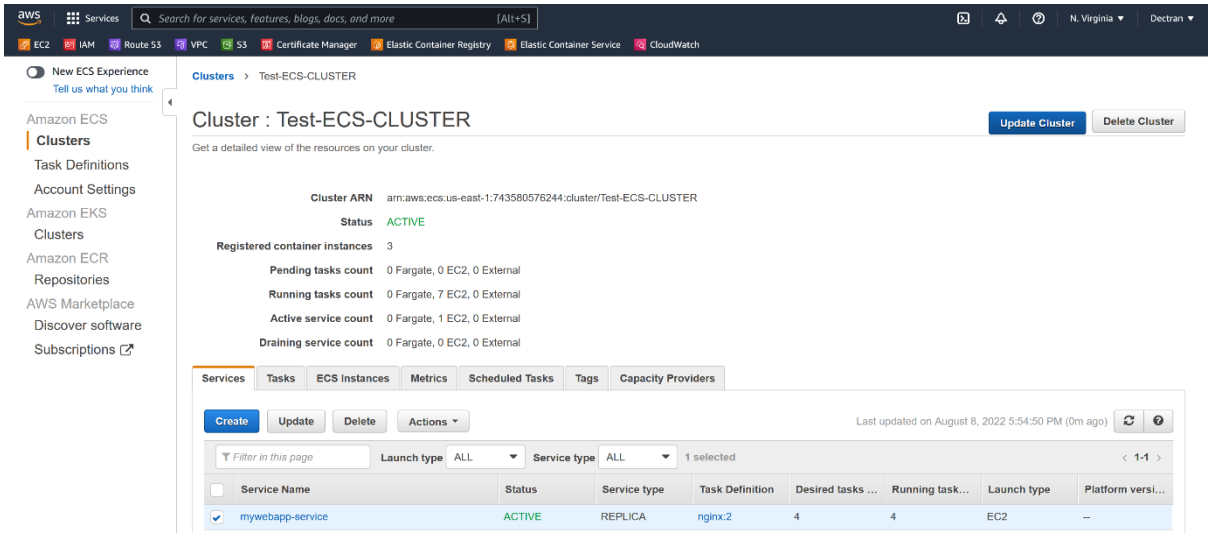


--- new task definition version got created.

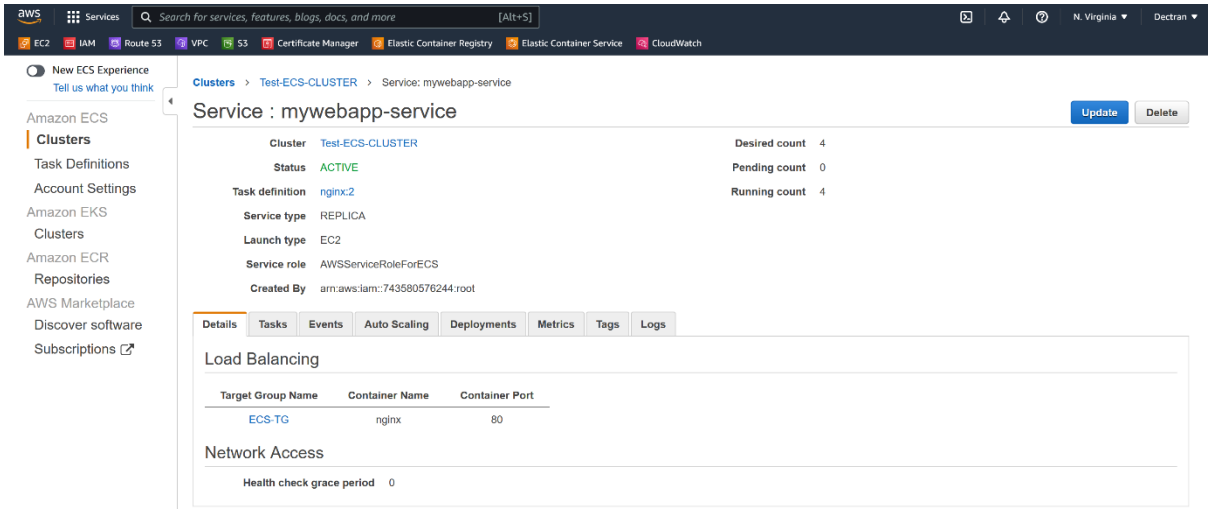
Update service with new task definition

--- **note** – here, I will update the service the new version of task definition.

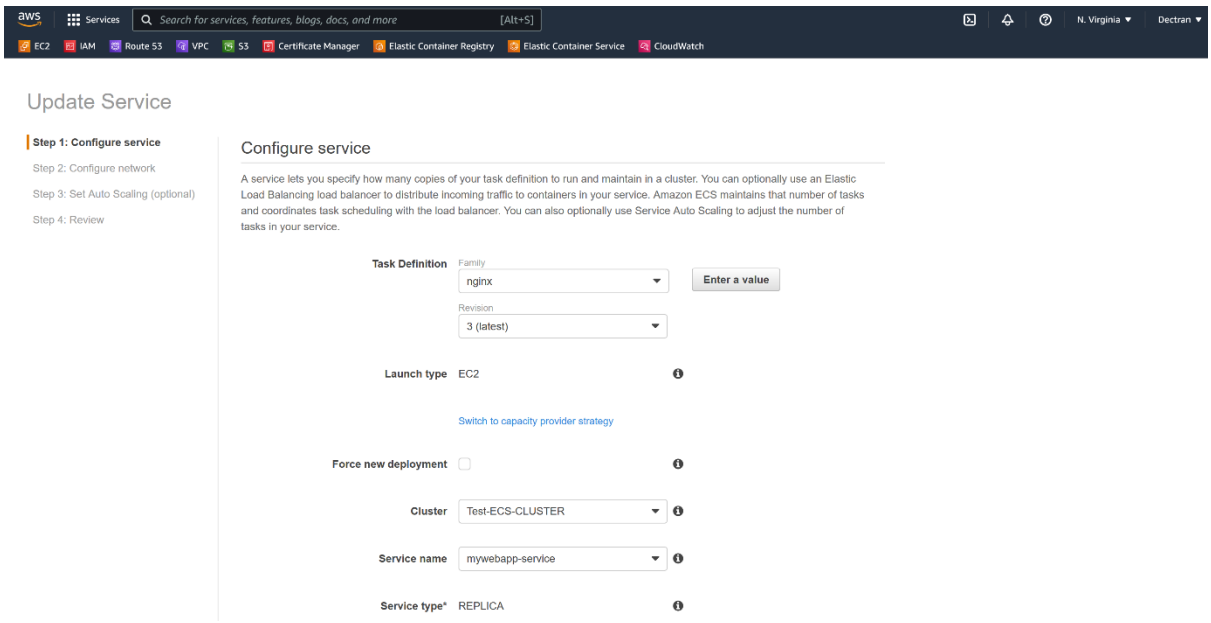
--- ECS -> cluster -> select service.



--- select the service.

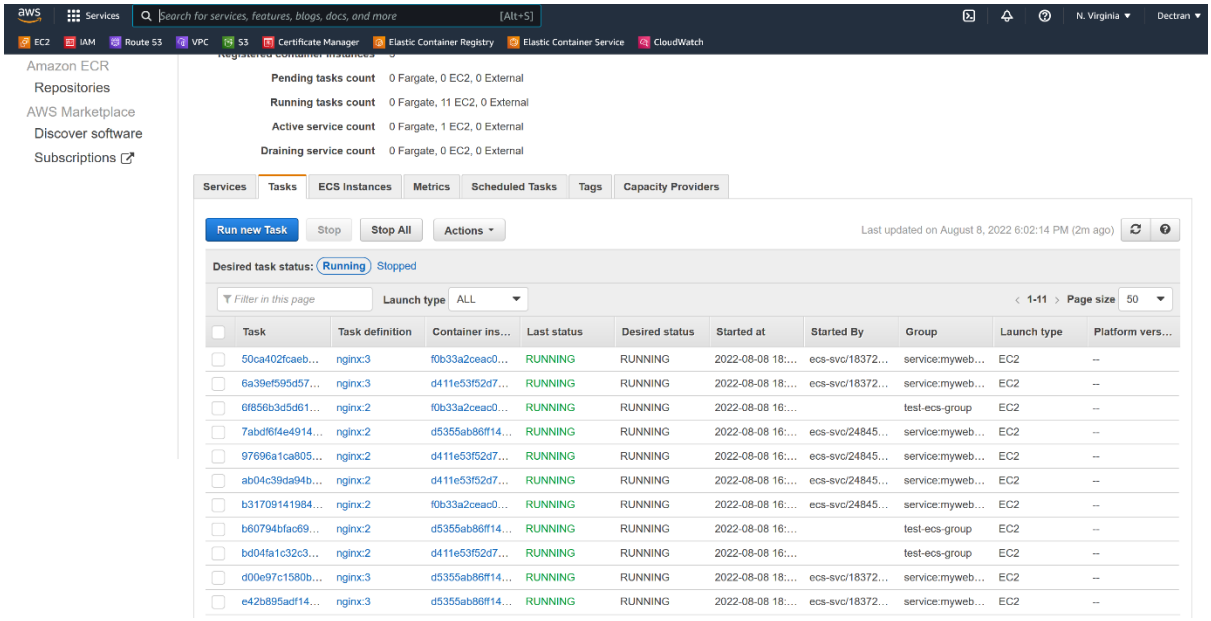


--- click on update cluster.



--- select the revision 3(latest) and click on next to complete.

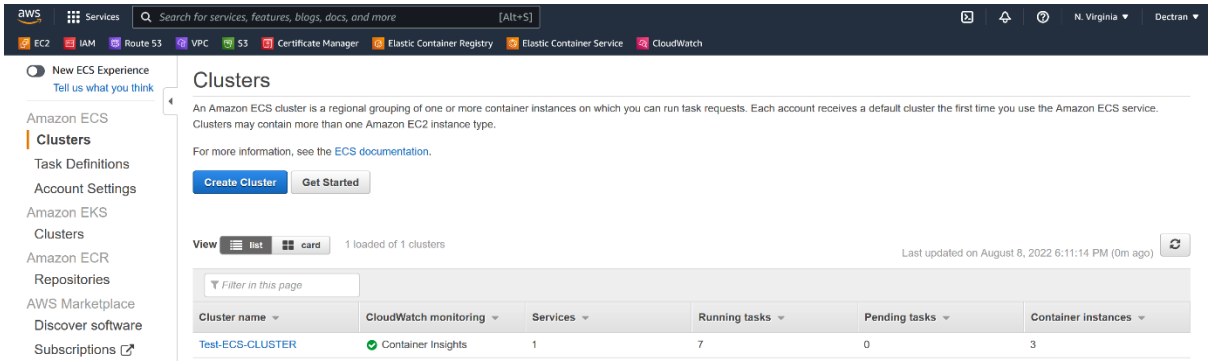
--- go to cluster -> click on task to see rolling deployment.



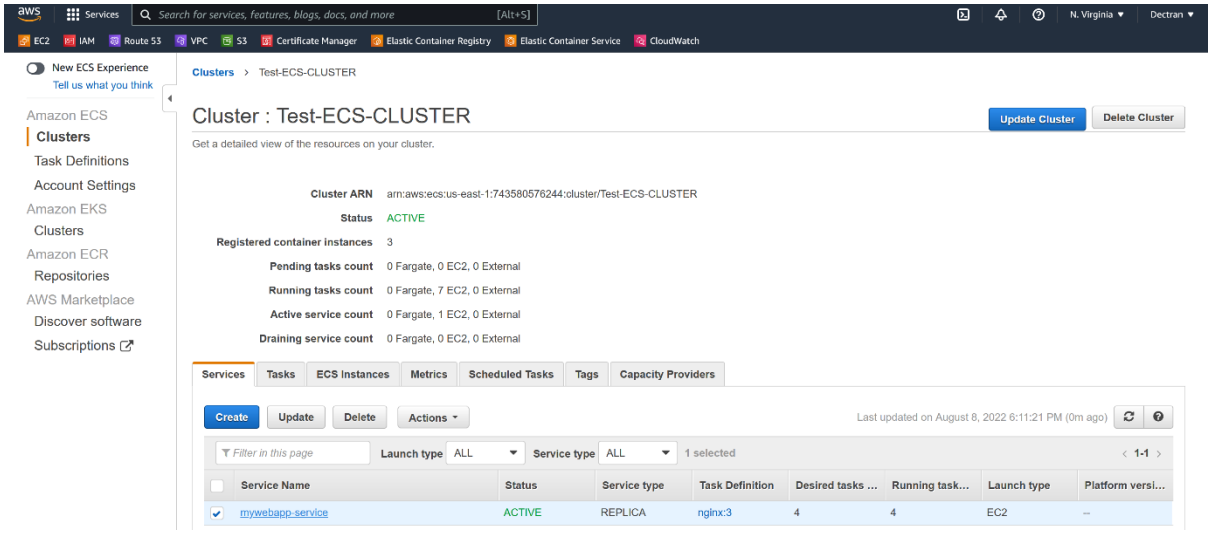
--- I have given 200% in the service while I am creating it. That is why it created 4 new containers. It will delete the old containers once it the new containers are ready.

Increase replicas using service

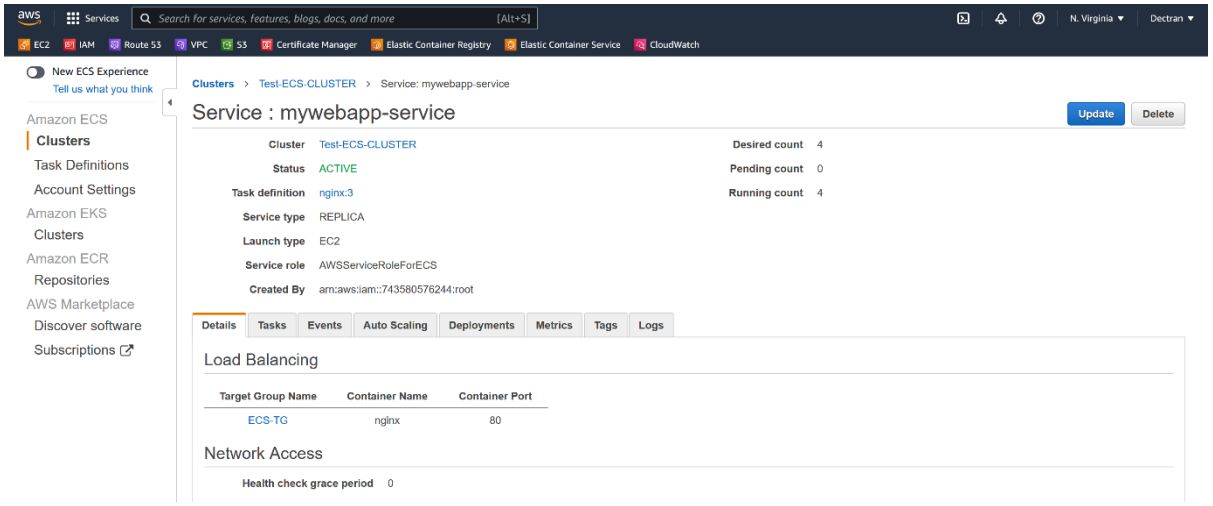
--- note – using service, we will increase the containers to 5 on 3 node cluster.



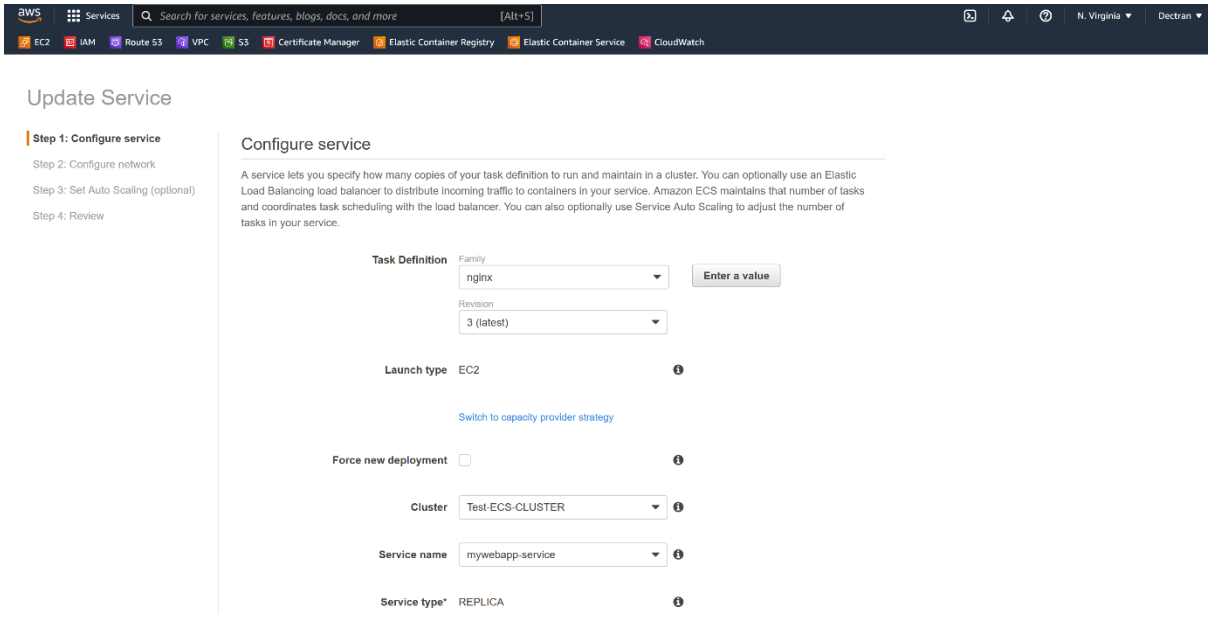
--- click on clusters and click on our cluster.

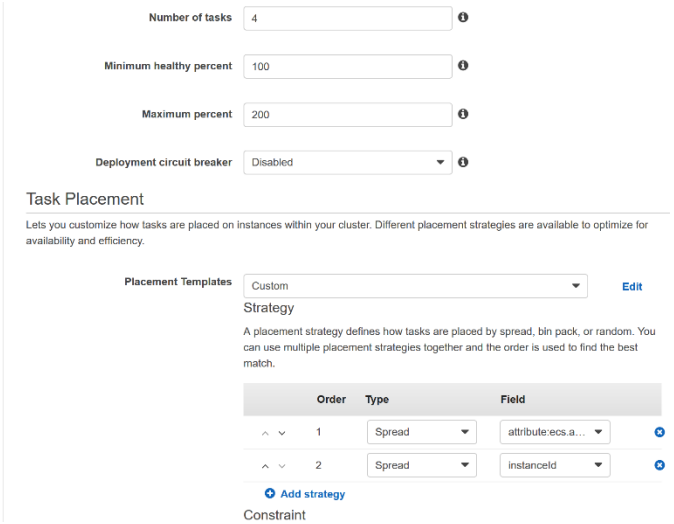


--- click on service.



--- click on update.





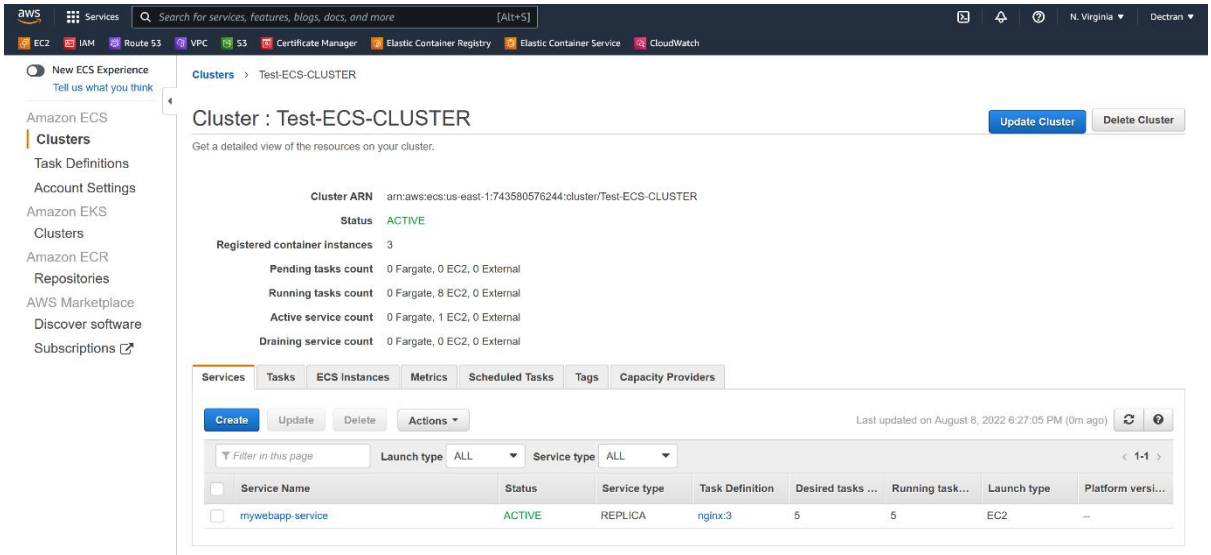
--- Number of tasks =5 – please enter your desired number of tasks here, if you give 5 tasks then it will create 5 replicas.

Deamonset creating ECS cluster

--- deamonset will create single pod on each node. It is like 1:1.

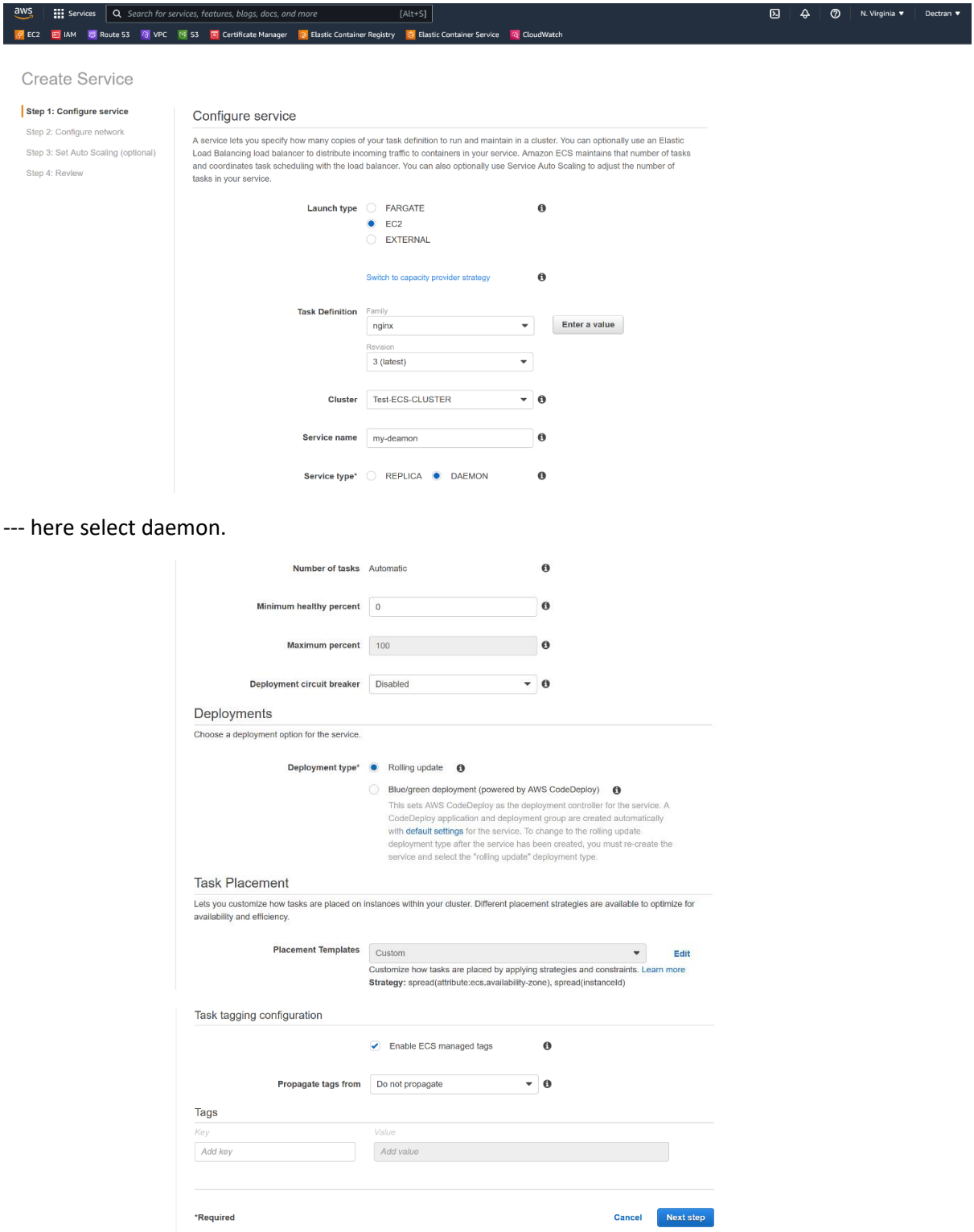
--- **scenario** – some times you need to run monitoring agent or logging agent on each server then we need to install single container on each node.

--- **note** – you need to have your own task definition of your monitoring agent. So that you can use in new service. In my case I do not have any agent now so I will use already existed task definition.



--- click on our ECS cluster to see above service page.

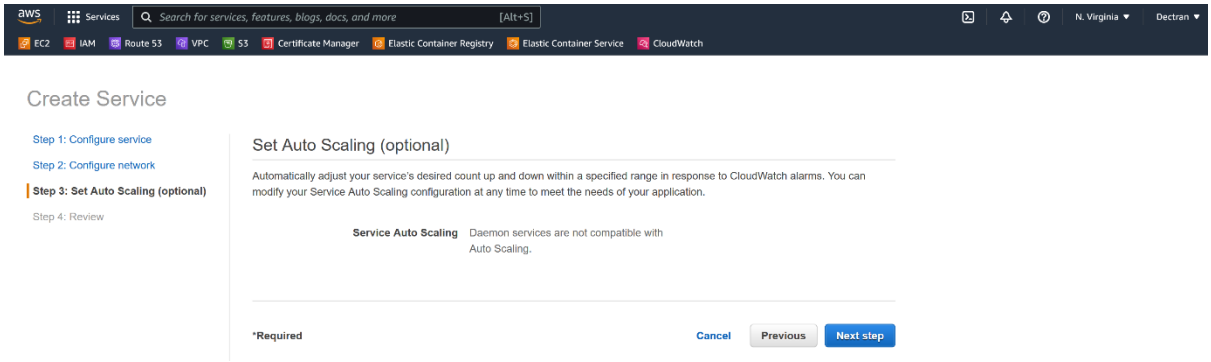
--- click on create.



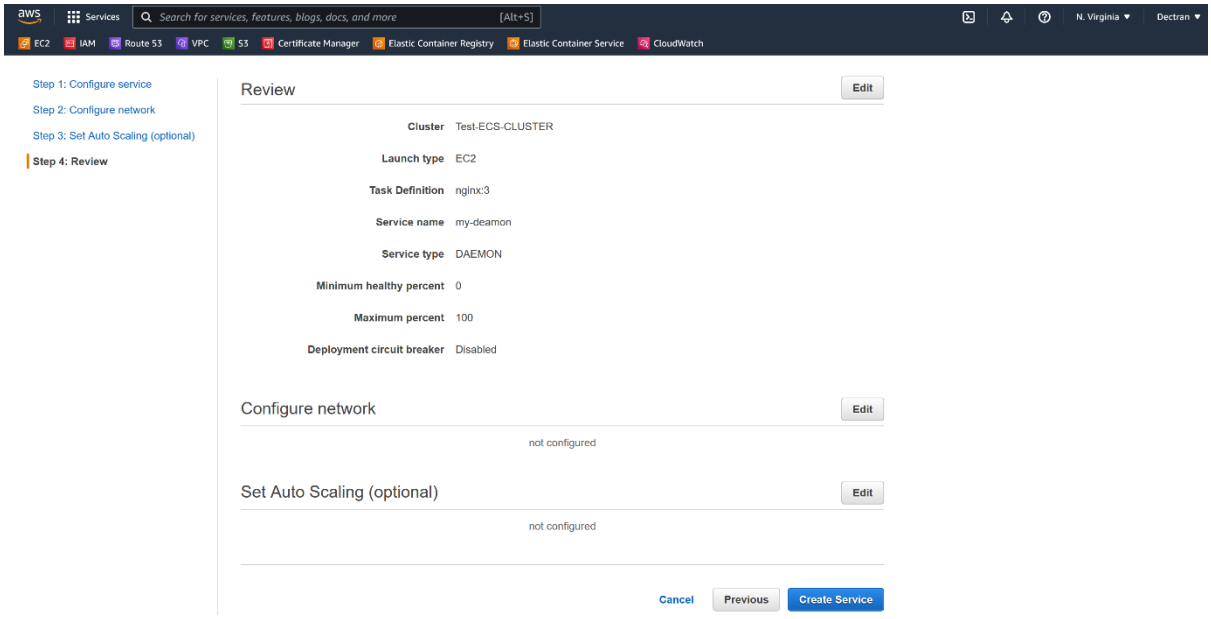
--- here select daemon.

--- Click on next

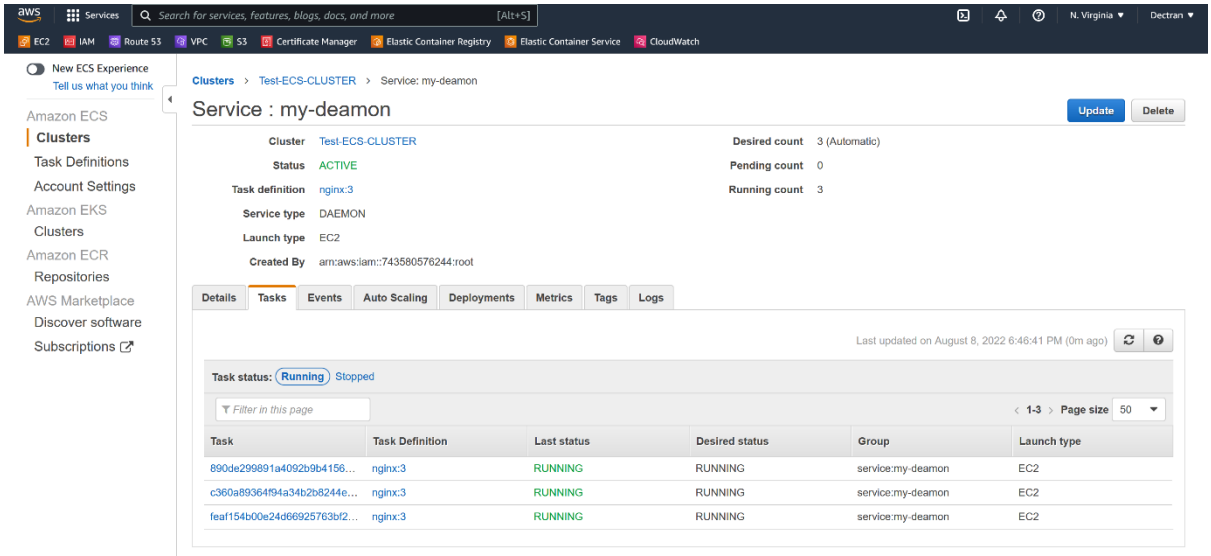
--- please select none and click on next



--- click on next step.

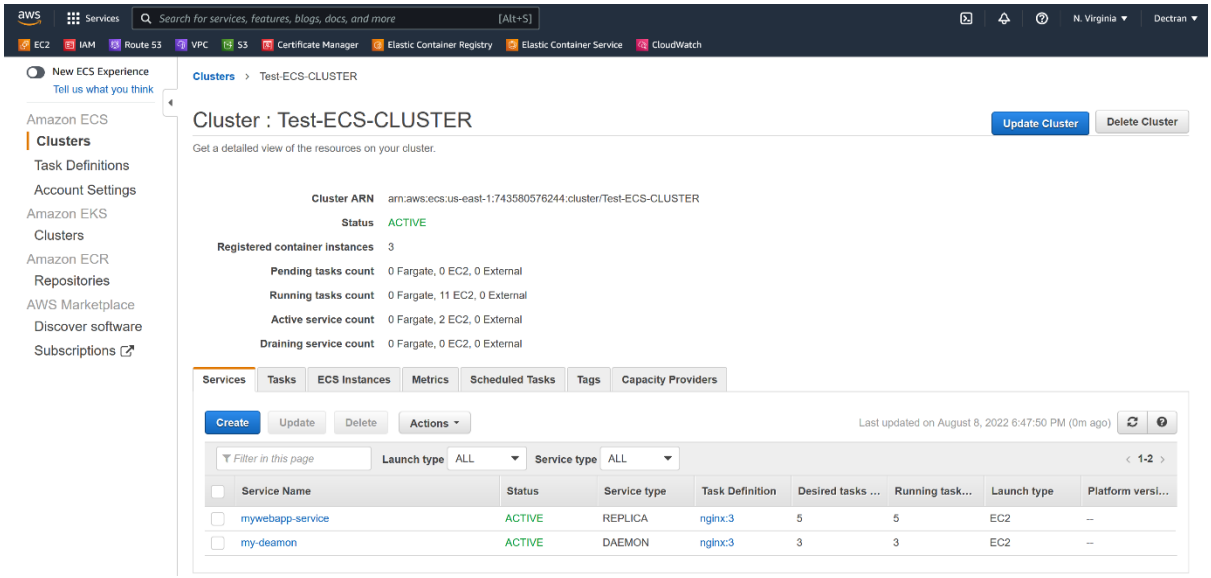


--- click on create service.



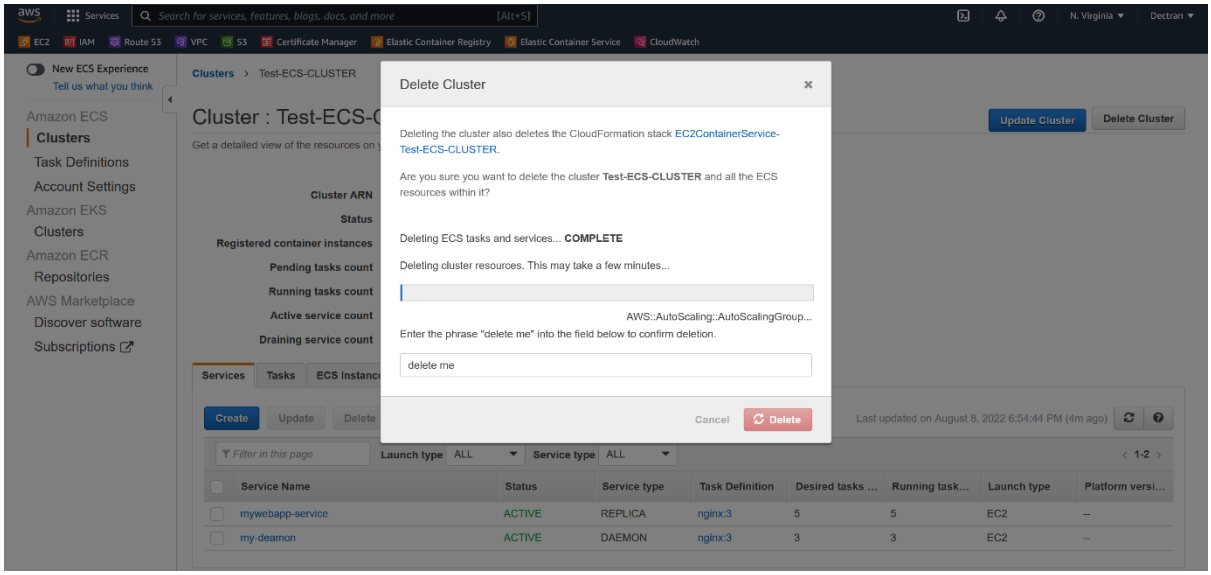
--- we have only 3 nodes and it only created containers.

Check services in our cluster



--- **note** – here, you can see I have created 2 services, 1st service is for application, 2nd service is for log monitoring.

Delete cluster



--- **note** – go to the ECS and click on the AMAZON ECS clusters and click on the delete cluster.

--- **important** – the main problem with ECS cluster node is, we need to manage the nodes, we need to security patch, we need to restart the amazon agent on the nodes. If you do not want to manages nodes and their security then go for the fargates.
