Task 3

ECS deployment using code pipelines:

Taking any sample application and deploy in ECS using the source code as code commit

Write the Docker file for an application and build the code by using the code Build and push the image to ECR.

Deploy the application in the ECS cluster.

Expose the application using Load Balancer

Prerequisites

1. Springboot as sample application
2. A CodeCommit repo
3. Dockerfile
4. Build spec file
5. ECS cluster task and service running

Create a Code Commit repo and push the required files for springboot app like pom.xml file

1. Fork the repo into your GitHub and add changes into the repo by adding the Docker file and build spec file and into the code commit repo
2. The Dockerfile for the image building

FROM openjdk:11

WORKDIR /app

RUN apt update -y && apt install -y maven

COPY . .

RUN mvn clean verify && cp ./target/\*.jar ./myapp.jar

EXPOSE 8085

CMD ["java", "-jar", "myapp.jar"]

1. The Buildspec.yml file for this springboot app

version: 0.2

env:

secrets-manager:

username: testing/sonar:username

token: testing/sonar:token

phases:

pre\_build:

commands:

- echo Logging in to Amazon ECR...

- aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 185633157927.dkr.ecr.ap-south-1.amazonaws.com

build:

commands:

- echo Building Docker image...

- docker login -u $username -p $token

- echo login success

- sudo docker build -t varthana2 .

post\_build:

commands:

- echo Pushing Docker image to Amazon ECR...

- docker tag varthana2:latest 185633157927.dkr.ecr.ap-south 1.amazonaws.com/varthana:latest

- REPOSITORY\_URI=185633157927.dkr.ecr.ap-south-1.amazonaws.com/varthana

- IMAGE\_TAG=latest

- docker push 185633157927.dkr.ecr.ap-south-1.amazonaws.com/varthana:latest

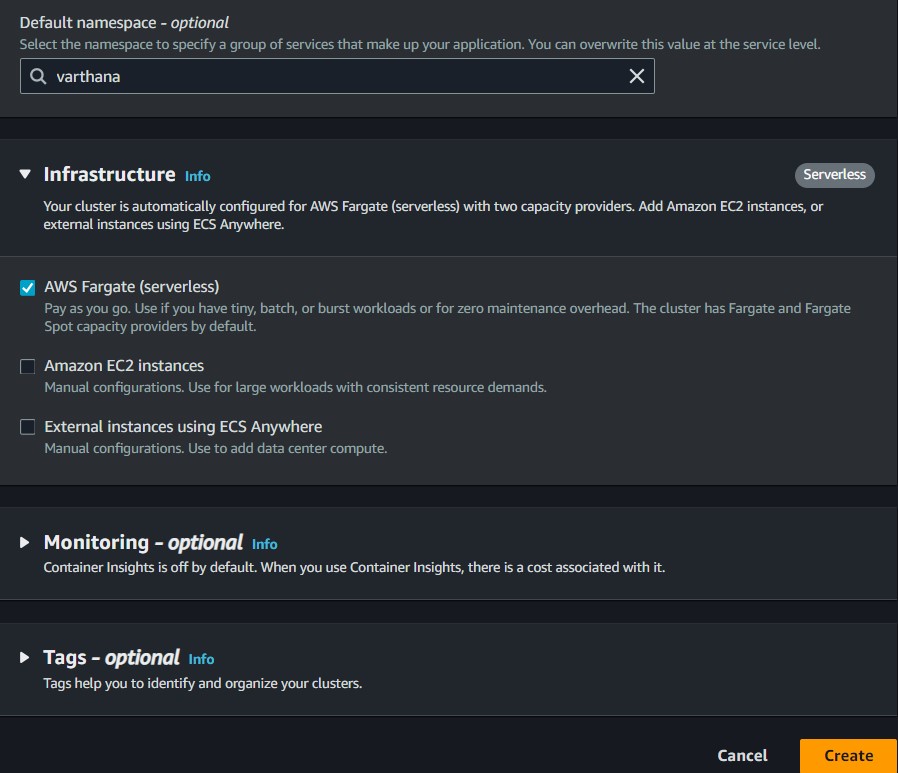
- printf '[{"name":"varthana","imageUri":"%s"}]' $REPOSITORY\_URI:$IMAGE\_TAG > imagedefinitions.json

artifacts:

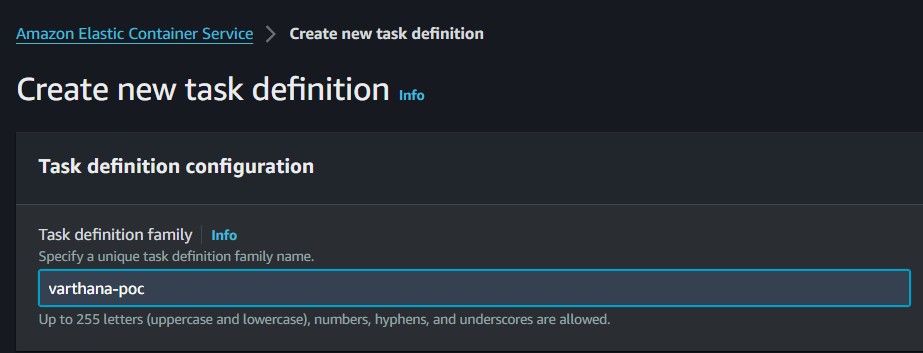
files: - imagedefinitions.json

1. Create a ECS cluster

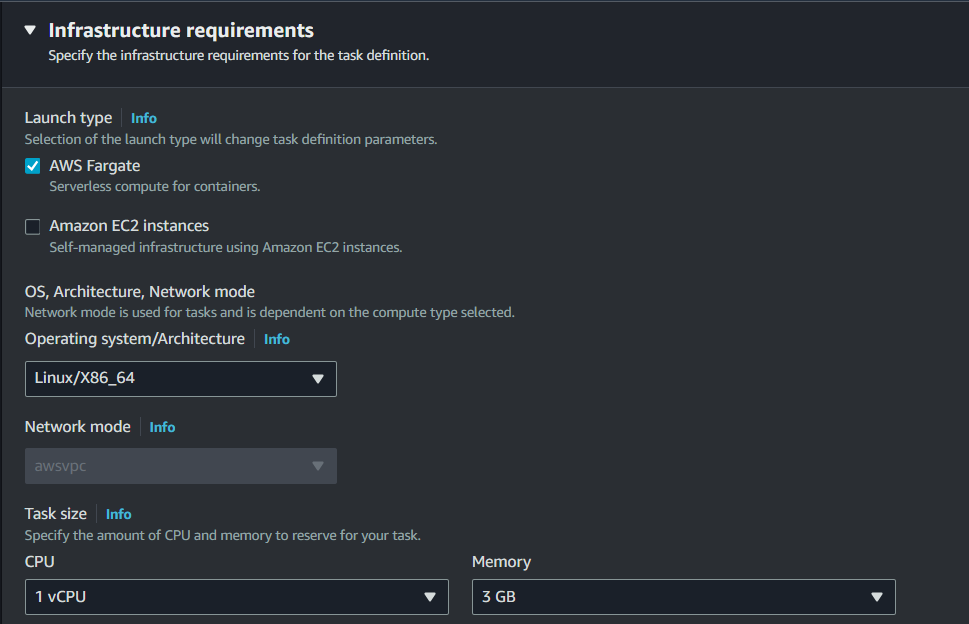
Navigate to the ECS from the console

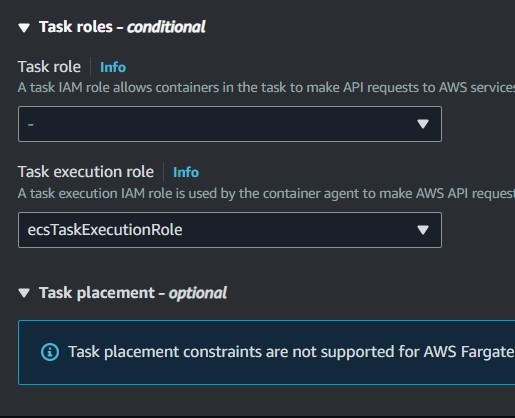
Select create cluster, provide a logical name and select create

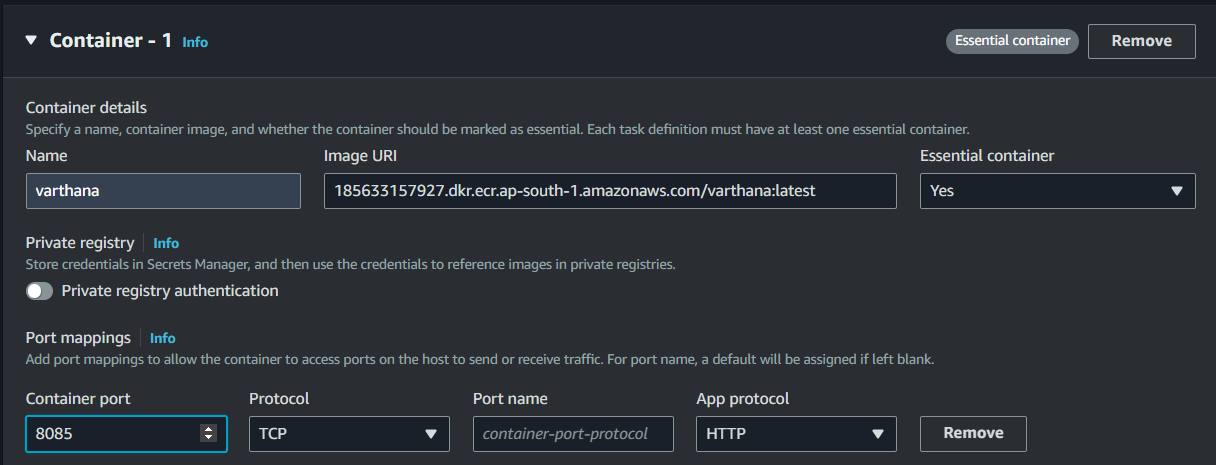
1. Navigate to left pane and choose to create task definition

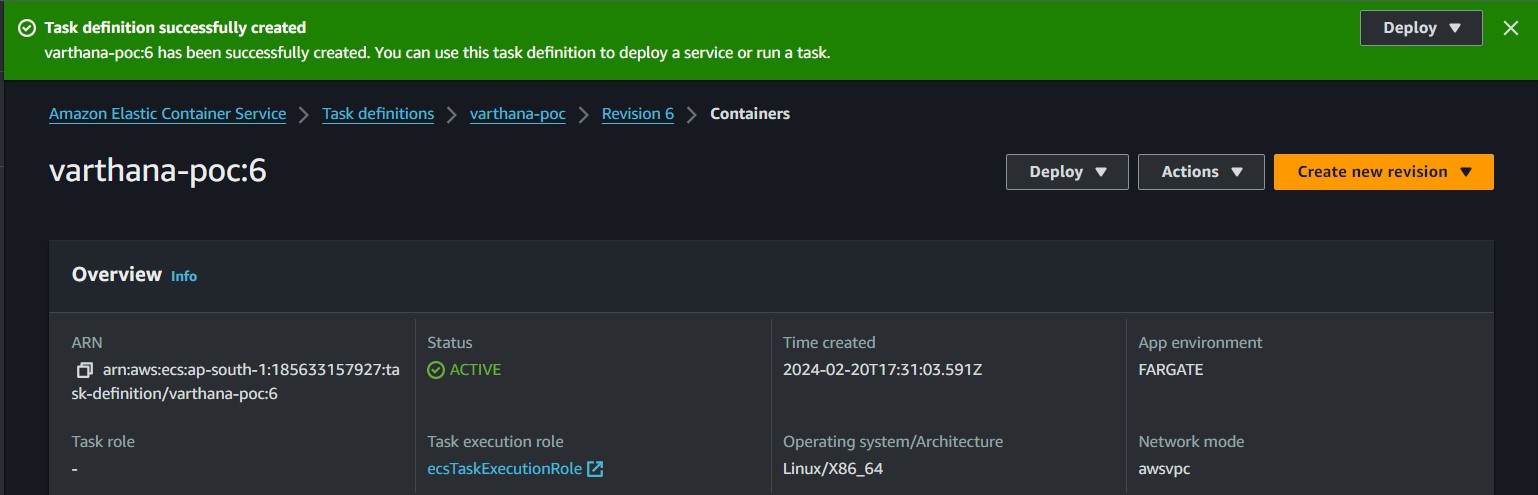
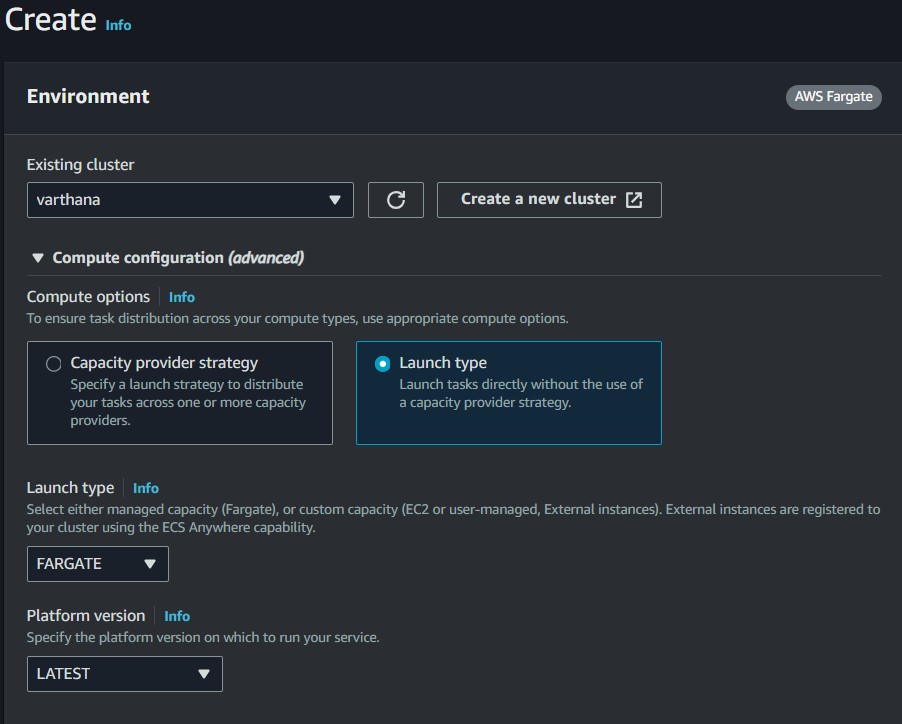
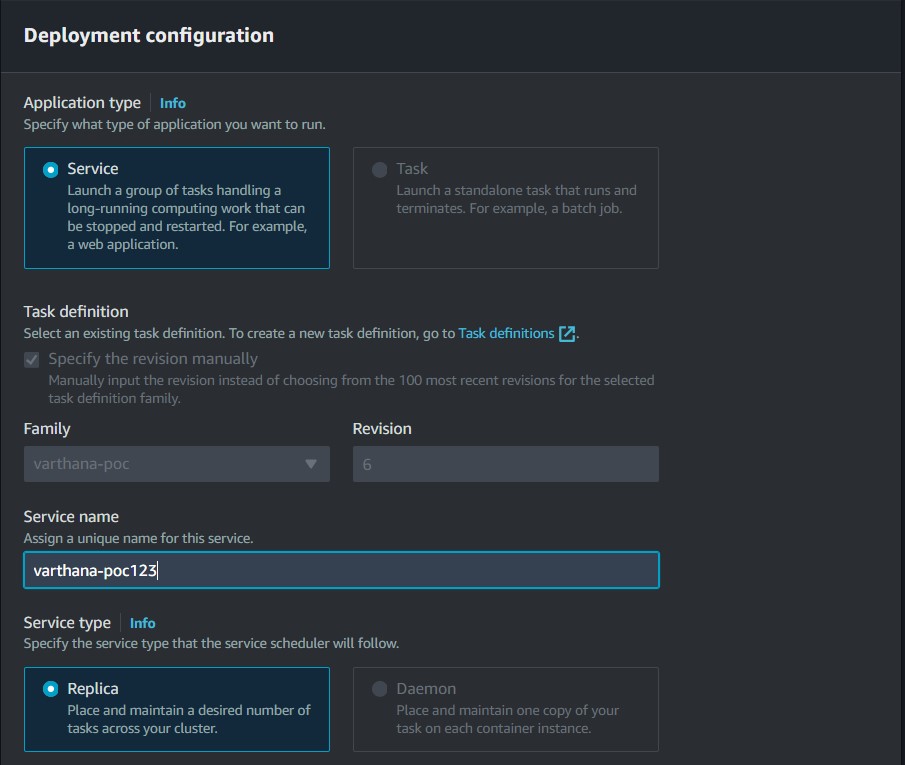
Select task definition and provide a name for the task definition

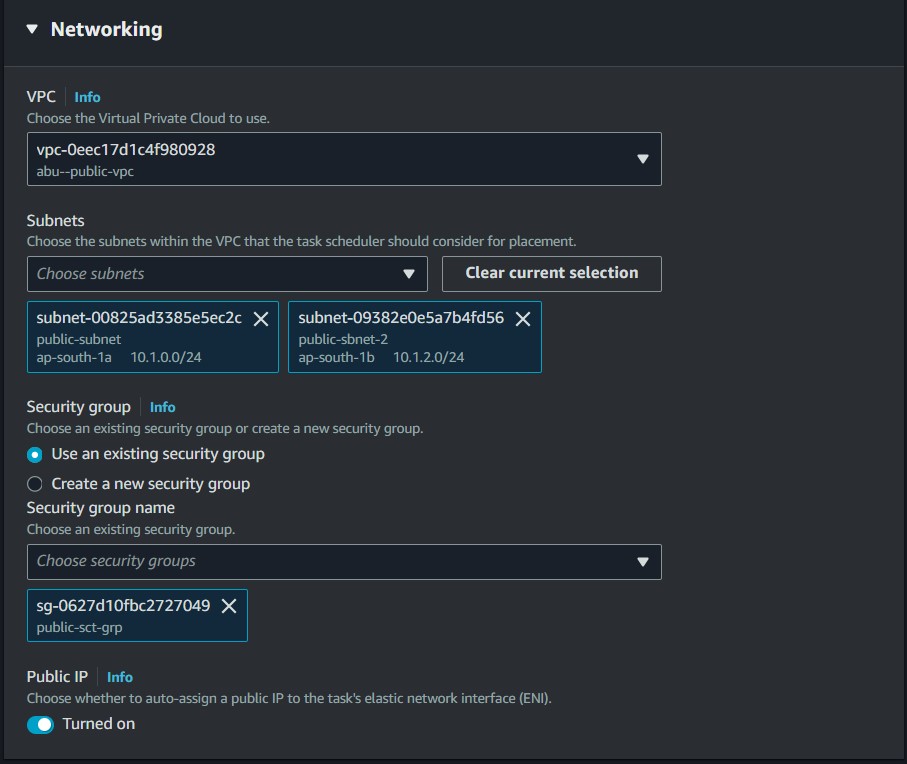
1. Launch type selected AWS Fargate

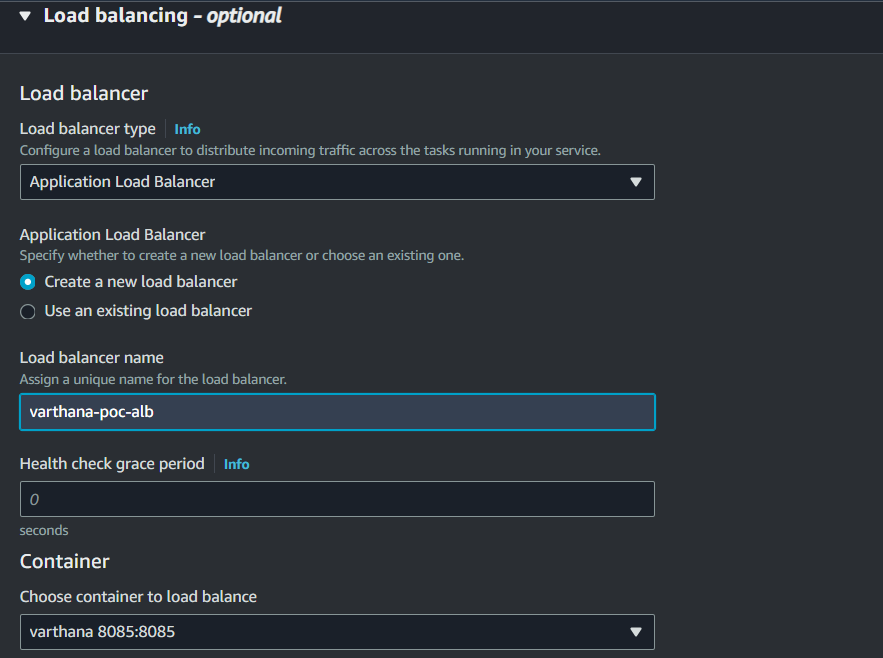
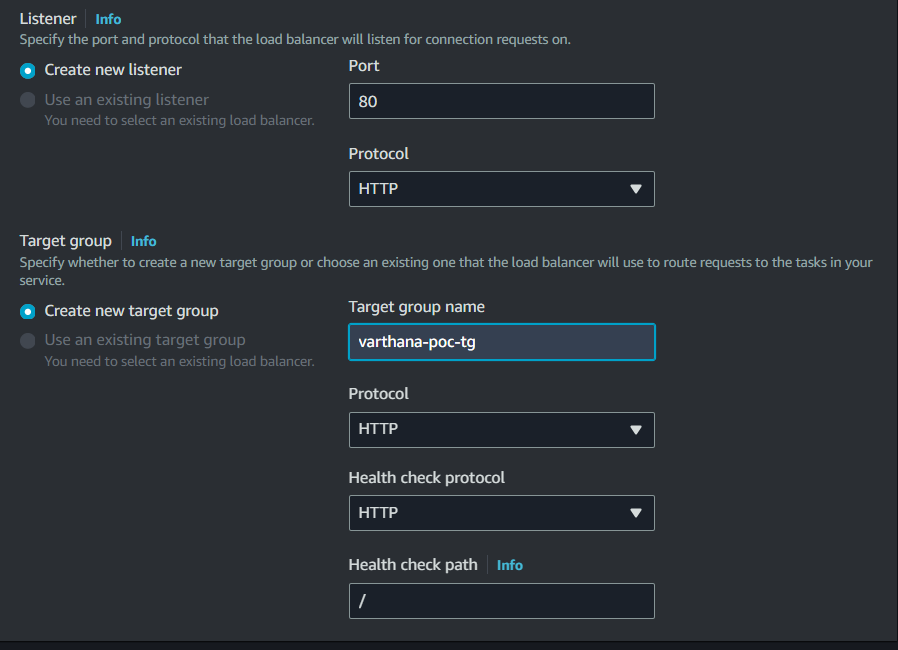
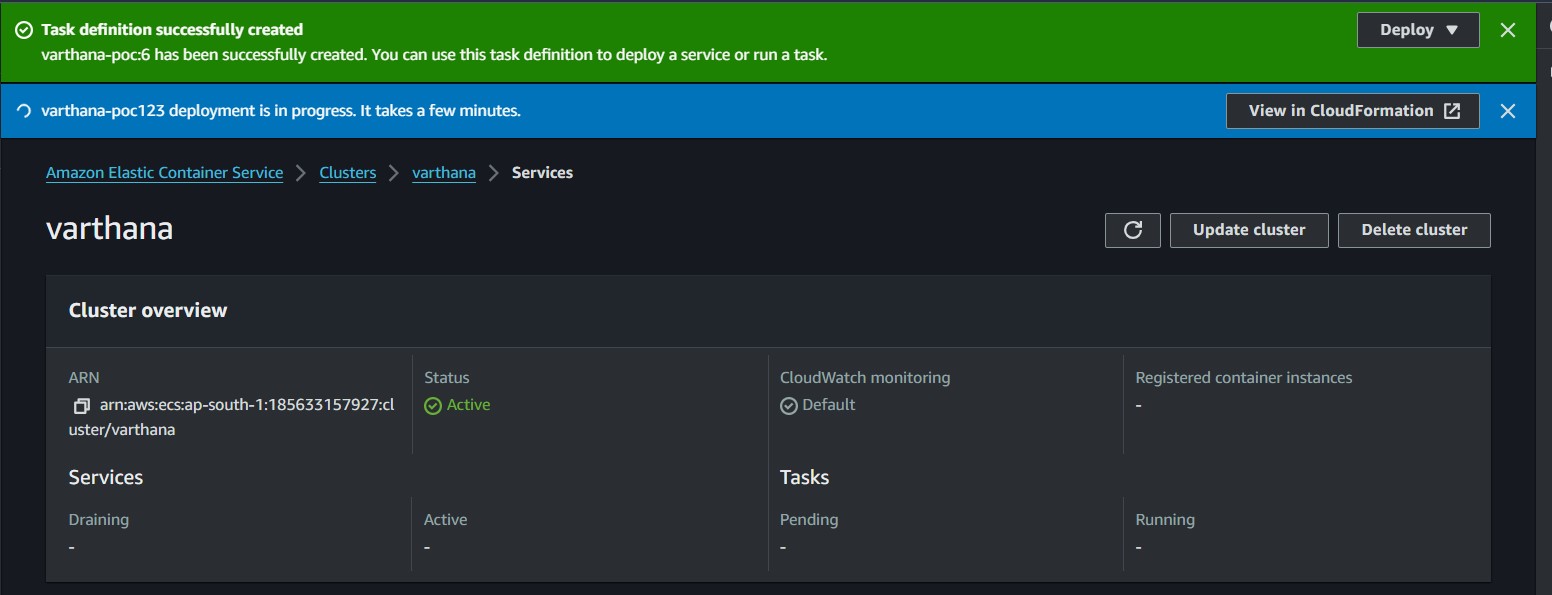
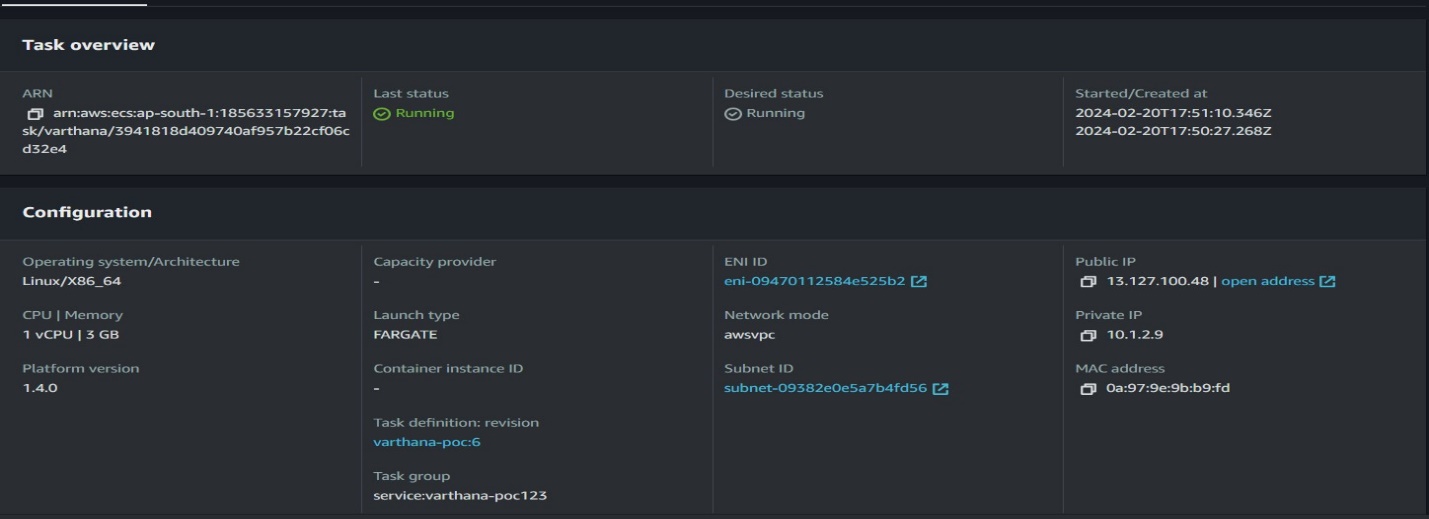
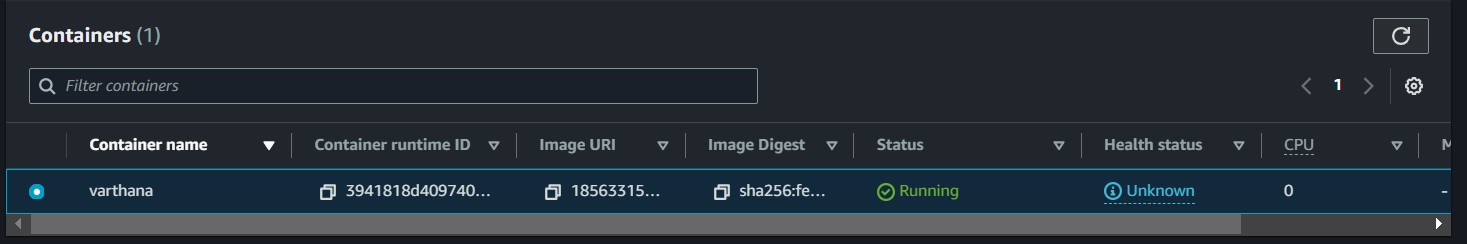
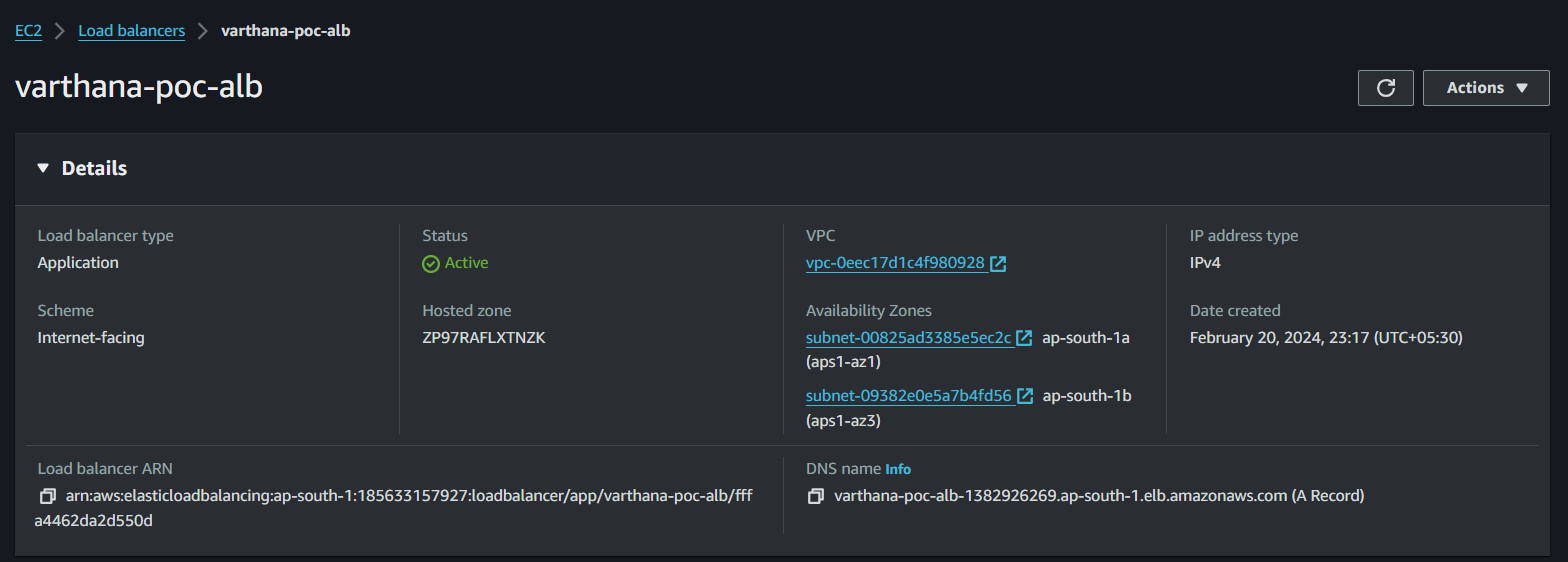
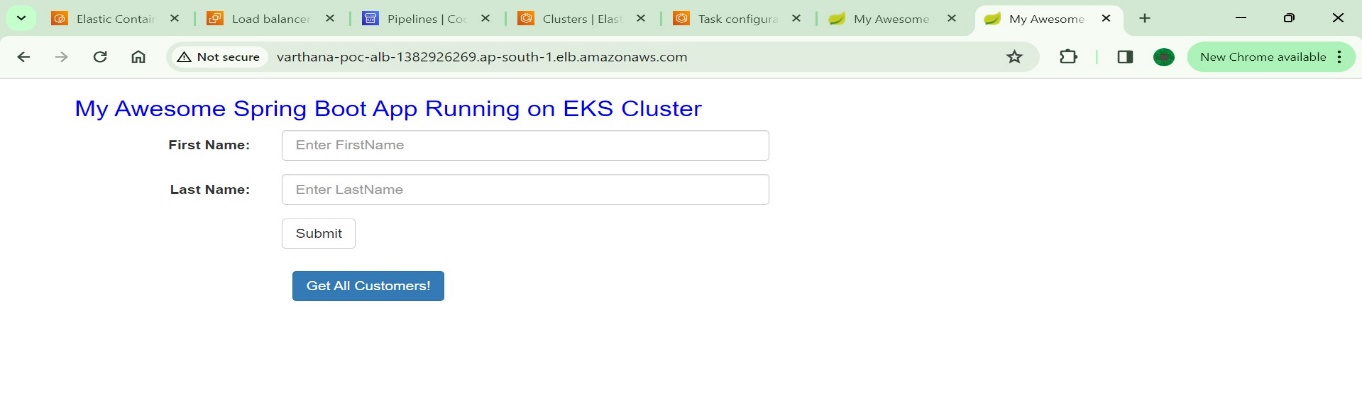


1. Task role choose to create a new role
2. Provide the name of the container image URI the port number on which the container is running

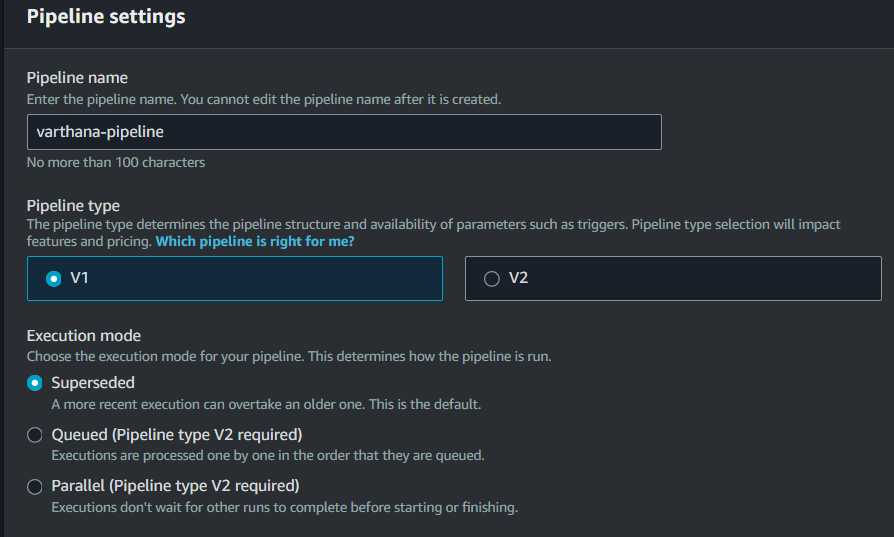


1. Keep all other as default and choose to create the task definition
2. Choose to create deploy and select create service, the cluster name pops up and it selects the existing cluster and select the fargte option
3. In the deployment configuration choose application type as service, provide a service name
4. Networking choose the VPC and the public subnets and the security group with port number 8085 open to it

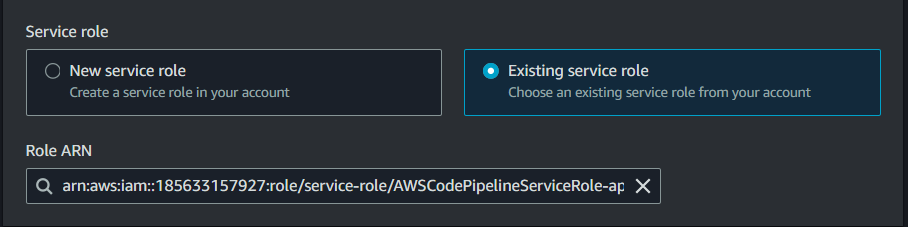


1. ALB option select ALB and provide a name for the LoadBalancer
2. Target Group provide a name for the target group and keep other default and choose to create
3. The service is in creation process
4. The service is up and running can see by the task overview
5. The container is running
6. The ALB created and accessible
7. the app via ALB dns name
8. Create a pipeline for the process

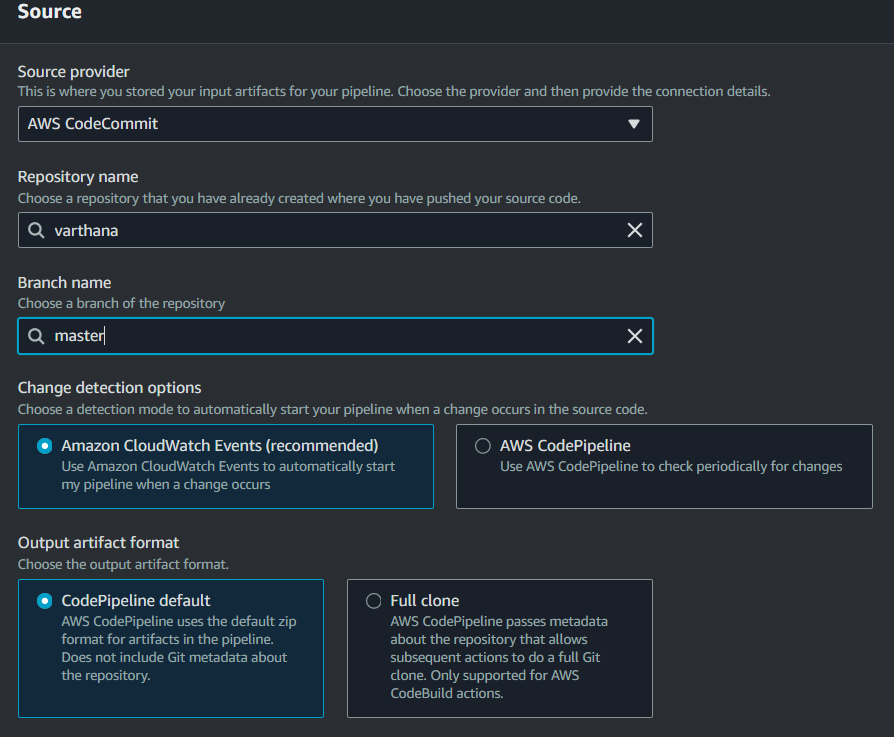
Navigate to pipeline and create a pipeline for the same

 Provide a logical name for the pipeline and select v1 as pipeline type

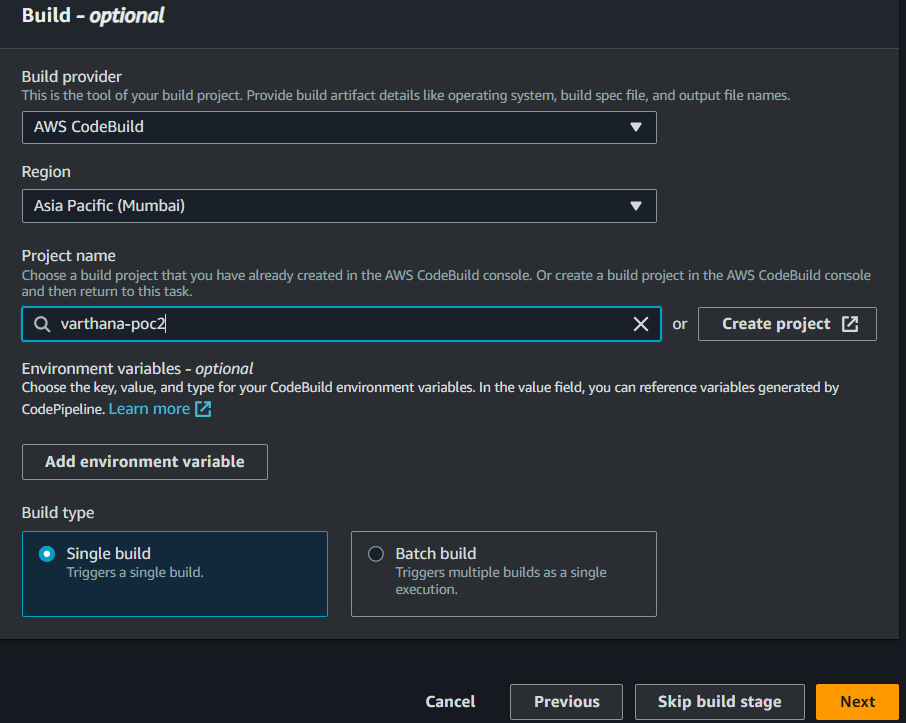
1. For first run can choose to create a new role and if creating is for second or third can choose to use an existing role and provide the role name



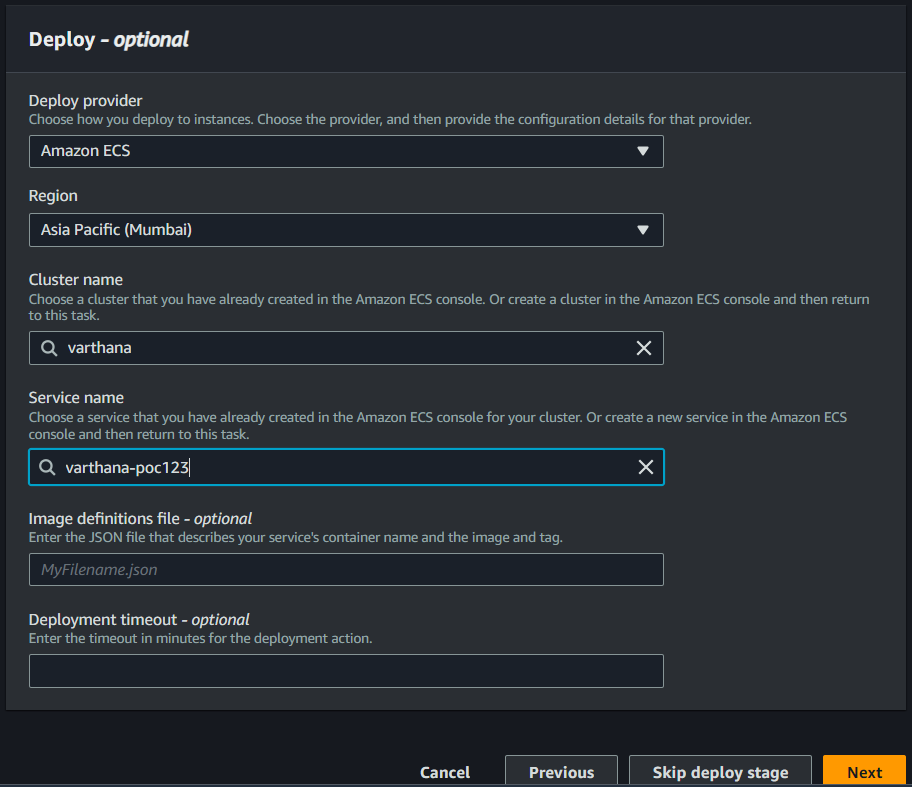
1. Choose next
2. Add source stage select source as code commit and provide the requirements

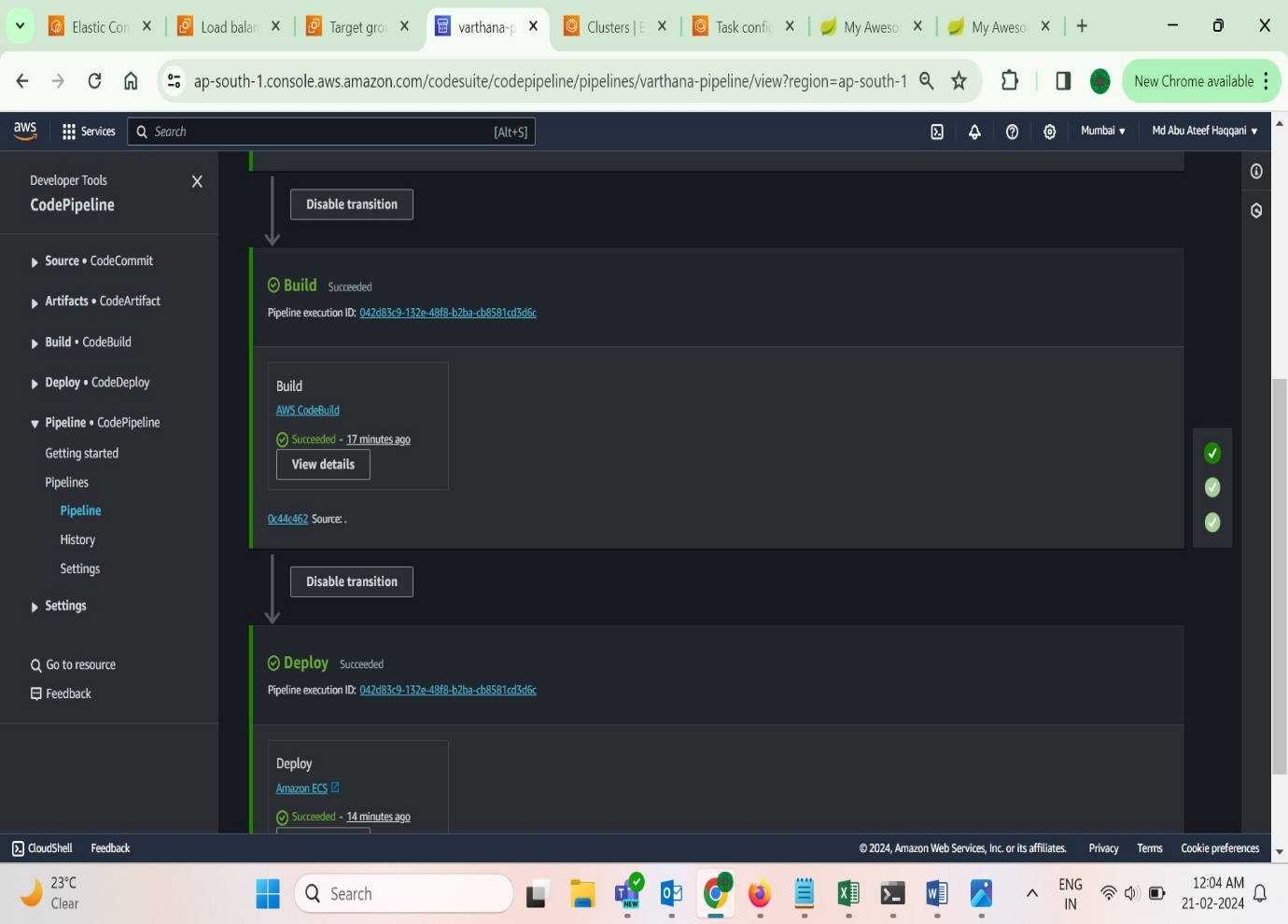
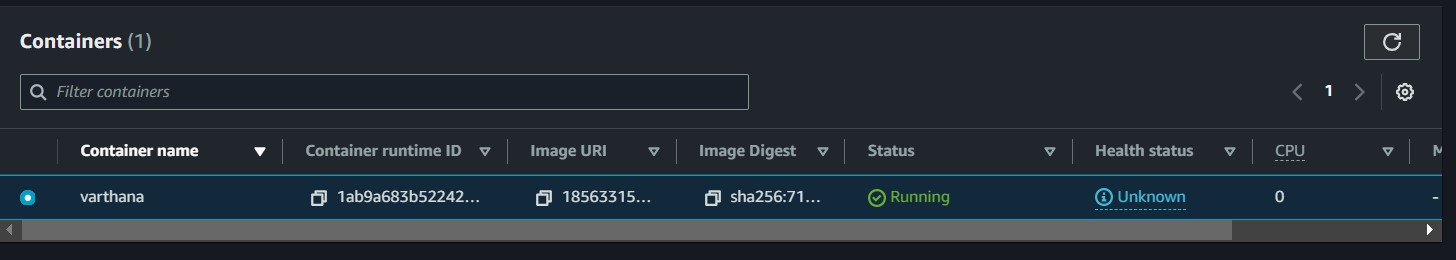
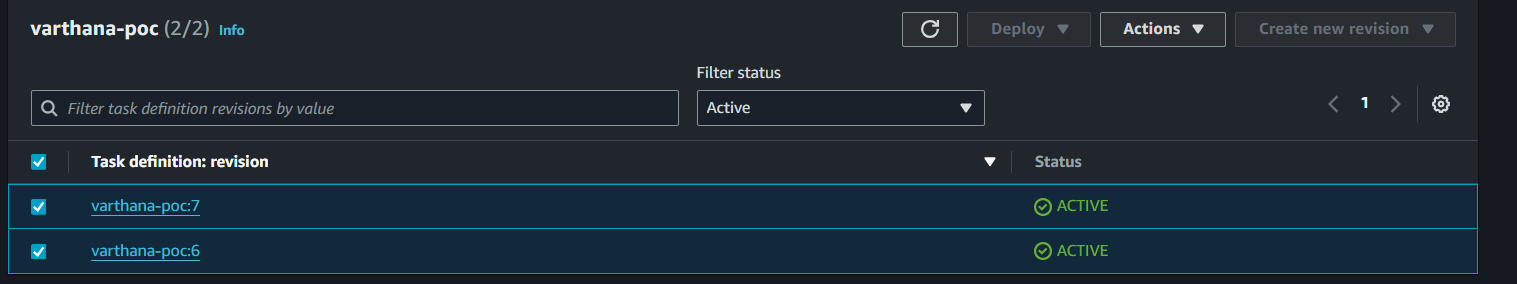


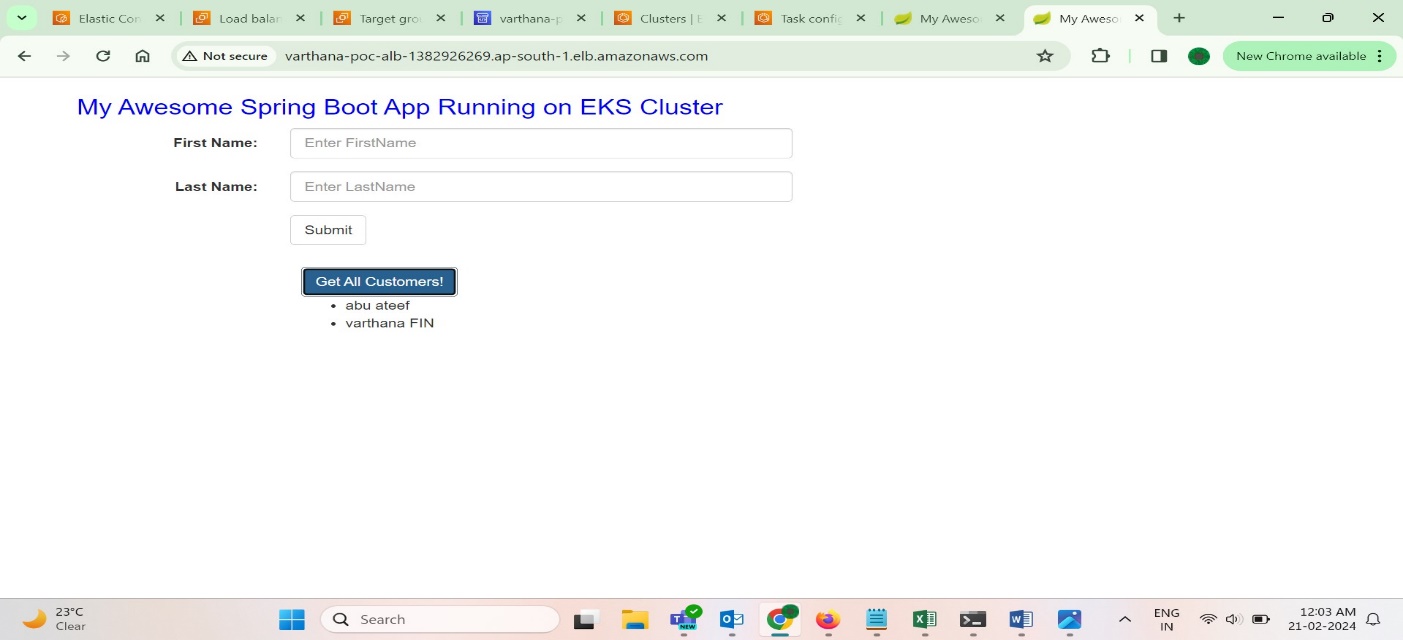
1. Choose next
2. Build stage choose the code build and the name of the build project for the task



1. Deploy stage choose provider as ECS and provide details



1. Choose next, verify the pipeline and click create
2. The pipeline is successful and build and deployed and can access the container by IP and also by the ALB DNS
3. The updated task definition
4. Can access the container IP and the LoadBalancer, the load balancer take time to come up



Done

Link for files

https://drive.google.com/drive/folders/13i6EzBT2oNYrvTu-4bRVFtIP-l1uSQxh?usp=drive\_link