**Project 7**

**CI/CD pipeline to build and deploy java container application into ECR and ECS using Aws code pipeline**

**-Jatharthan Tharmanandasivam**

# Target of the Project

Target of the project is to deploy a Spring Boot application to ECS cluster.  
Tools that will be used

AWS code Pipeline

AWS code Build

AWS code Deploy

Git/GitHub

Elastic Container Registry ECR

Elastic Container Service ECS

IAM

# Repo link

<https://github.com/jatharthan/NCPL_AWS_Pipeline_ecr_ecs_spring>

# Steps to complete the project.

01.Created a code repository that have a sample Spring Boot application in GitHub. Created a Dockerfile to containerise the application.

A screenshot of a computer

AI-generated content may be incorrect.

02.Dockerfile this will be used to create the docker image and expose the application at port 8080

FROM openjdk:17-jdk-slim

WORKDIR /app

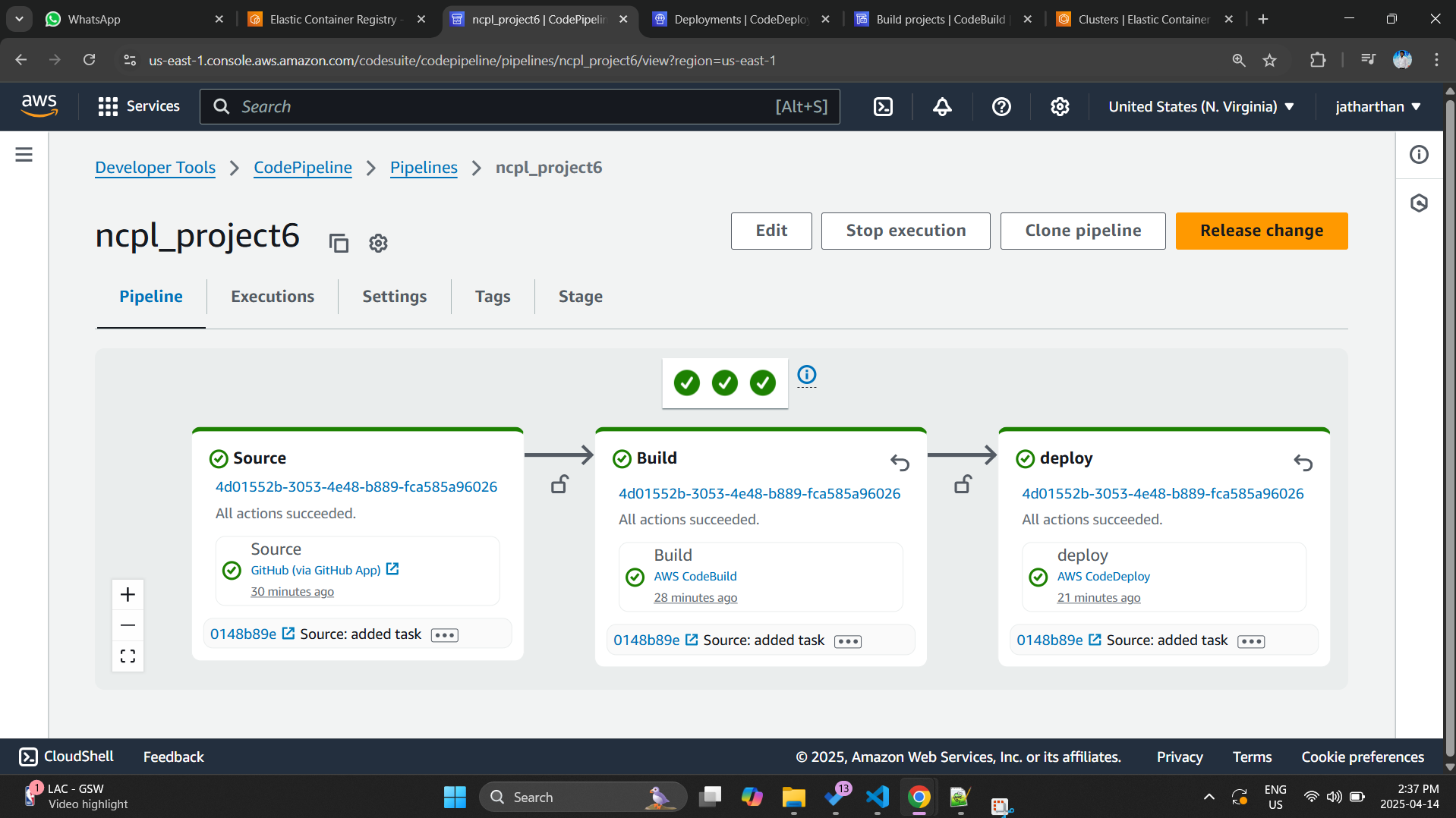
COPY target/demo-1.0.0.jar app.jar

EXPOSE 8080

CMD ["java", "-jar", "app.jar", "--server.port=8080"]

Successfully ran the pipeline

03.Created a AWS code pipeline that will checkout code from the GitHub repository and build docker image, push it to ECR repository and then deploy it to the ECS cluster. The below image shows that pipeline ran successfully, and the following document will explain the pipeline in detail.



04.The build phase of the pipeline uses the build project we created which uses the buildspec.yml to build a docker image and then push it to the ECR repository. Necessary IAM permissions was granted to access and write to ECR. The below image shows the images pushed to the ECR repo.

A screenshot of a computer

AI-generated content may be incorrect.

The buildspec.yml code used

version: 0.2

phases:

pre\_build:

commands:

- echo Installing dependencies...

- yum install -y jq maven

- echo Logging in to Amazon ECR...

- aws --version

- ACCOUNT\_ID=715841367723

- REGION=us-east-1

- REPO\_NAME=project6

- IMAGE\_TAG=latest

- aws ecr get-login-password --region $REGION | docker login --username AWS --password-stdin $ACCOUNT\_ID.dkr.ecr.$REGION.amazonaws.com

build:

commands:

- echo Building the JAR file...

- mvn clean install

- echo Building the Docker image...

- docker build -t $REPO\_NAME .

- docker tag $REPO\_NAME:$IMAGE\_TAG $ACCOUNT\_ID.dkr.ecr.$REGION.amazonaws.com/$REPO\_NAME:$IMAGE\_TAG

post\_build:

commands:

- echo Pushing Docker image to Amazon ECR...

- docker push $ACCOUNT\_ID.dkr.ecr.$REGION.amazonaws.com/$REPO\_NAME:$IMAGE\_TAG

- echo Writing imagedefinitions.json...

- printf '[{"name":"sprinrajco","imageUri":"%s.dkr.ecr.%s.amazonaws.com/%s:%s"}]' $ACCOUNT\_ID $REGION $REPO\_NAME $IMAGE\_TAG > imagedefinitions.json

- echo Writing appspec.yml...

artifacts:

files:

- imagedefinitions.json

- appspec.yml

05.After the build phase we have created a deployment application that will get the docker image from ECR and deploy to ECS cluster. To setup this ECS cluster, Task definition and ECS services were created in advance. ECS deployment was set to blue green deployment. The below image shows the successful deployment from code deploy application which we created and integrated with ECS. Take a note of the deployment ID in the image.

A screenshot of a computer

AI-generated content may be incorrect.

Appspec.yml file that was used for the deployment

---

version: 1

Resources:

- TargetService:

Type: AWS::ECS::Service

Properties:

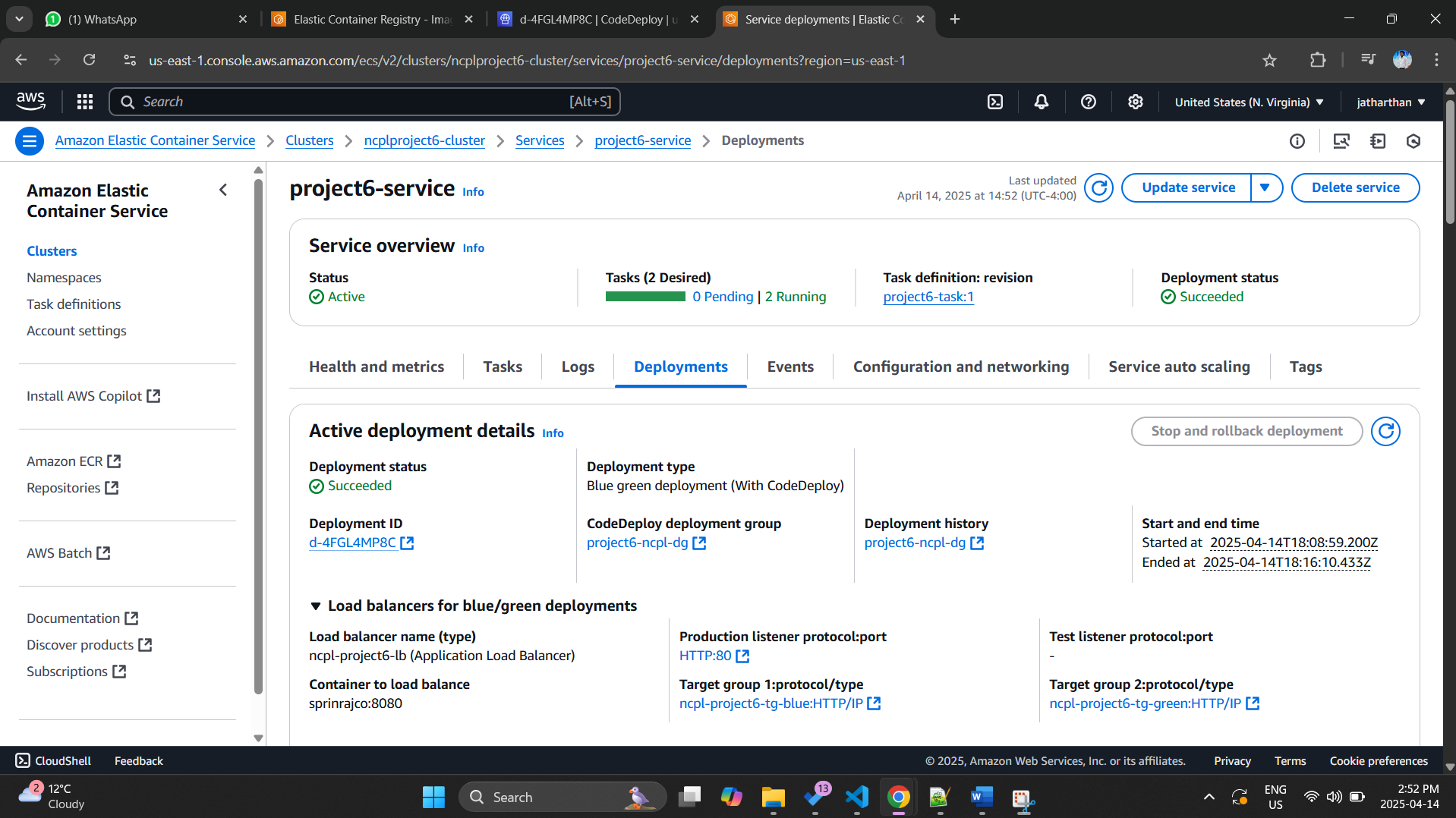
TaskDefinition: "arn:aws:ecs:us-east-1:715841367723:task-definition/project6-task:1"

LoadBalancerInfo:

ContainerName: "sprinrajco"

ContainerPort: 8080

06.In the below image we can take a note of Deployment ID to confirm that the deployment is from the code deploy application from the pipeline. You can also note that in the deployment type Blue green deployment is there.



07.To further confirm we can check on the Deployment log.In the logs as underlined you can see 2 new targets being registered in the target group green and then the deployment waits for the new deployment to get stabilize and then deregister the old target group blue. When another deployment happens it will happen in the other way round

A screenshot of a computer

AI-generated content may be incorrect.

08.Successfully running application which was accessed by ECS load balancer DNS

A screenshot of a computer

AI-generated content may be incorrect.

Challenges faced.

01.Faced challenges in figuring out a way to update the task definition and trigger a deployment.

02.Faced challenges in delegating the access to ECR and ECS.

03.When launching the ECS service failed several times due to the health checks. Figured out that health check was not responding from the application. Added a separate health check route which will respond with 200 ok /health.

04.Due to the blue green deployment had several issues in updating the appspec.yml file and finally figured out a way.