**Project name: Containerized Microservices with ECS Fargate**

GitHub repo: <https://github.com/Mithra1995/sampleproject.git>

# Objective

To design, deploy, and manage a containerized microservices architecture using AWS services, specifically AWS ECS Fargate . The goal is to create a robust, scalable, and secure infrastructure to run microservices in the cloud using Docker containers. The project will include the following:  
- \*\*Containerization of Microservices\*\* using Docker  
- \*\*Deployment using ECS Fargate\*\*   
- \*\*CI/CD pipeline for continuous delivery and management\*\* of microservices  
- \*\*Monitoring and Logging\*\* for performance insights

# Architecture Overview

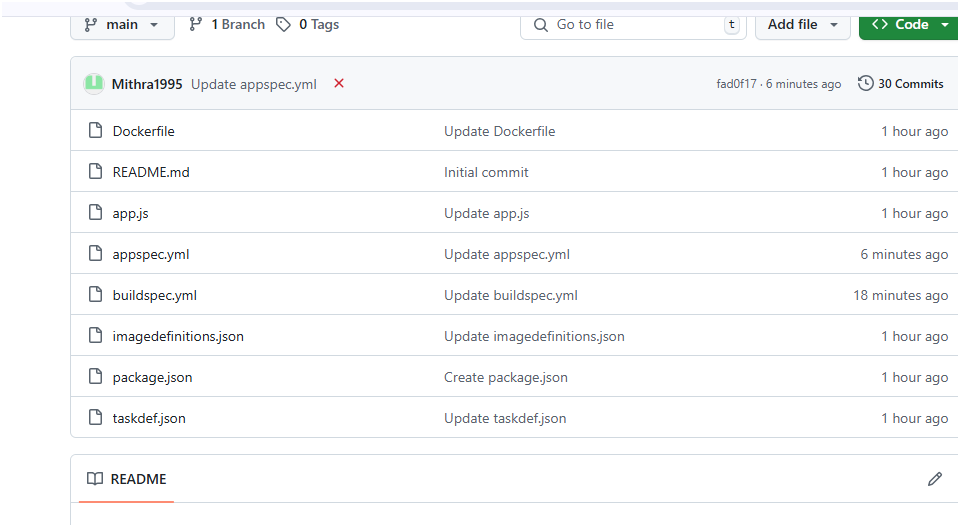
The architecture consists of multiple AWS services to support microservices, which include ECS Fargate, Application Load Balancer (ALB) for traffic routing. This will be coupled with CI/CD pipelines using Jenkins or AWS CodePipeline.

# Services Used

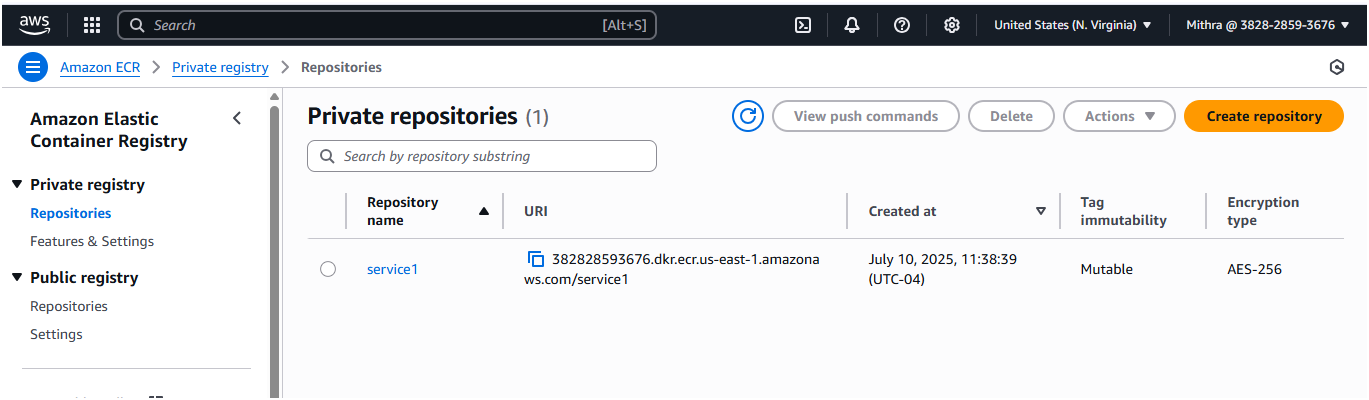
- \*\*Amazon ECS Fargate\*\* or \*\*Amazon EKS\*\* (Elastic Kubernetes Service) for container orchestration  
- \*\*Amazon RDS\*\* (Relational Database Service) for database storage  
- \*\*AWS Application Load Balancer\*\* (ALB) for routing traffic to microservices  
- \*\*Amazon VPC\*\* (Virtual Private Cloud) for networking and security  
- \*\*AWS CloudWatch\*\* for monitoring and logging  
- \*\*Docker\*\* for containerizing microservices  
- \*\*Amazon ECR\*\* (Elastic Container Registry) for storing Docker images  
- \*\*AWS CodePipeline / Jenkins\*\* for CI/CD automation

# Step-by-Step Implementation Tasks

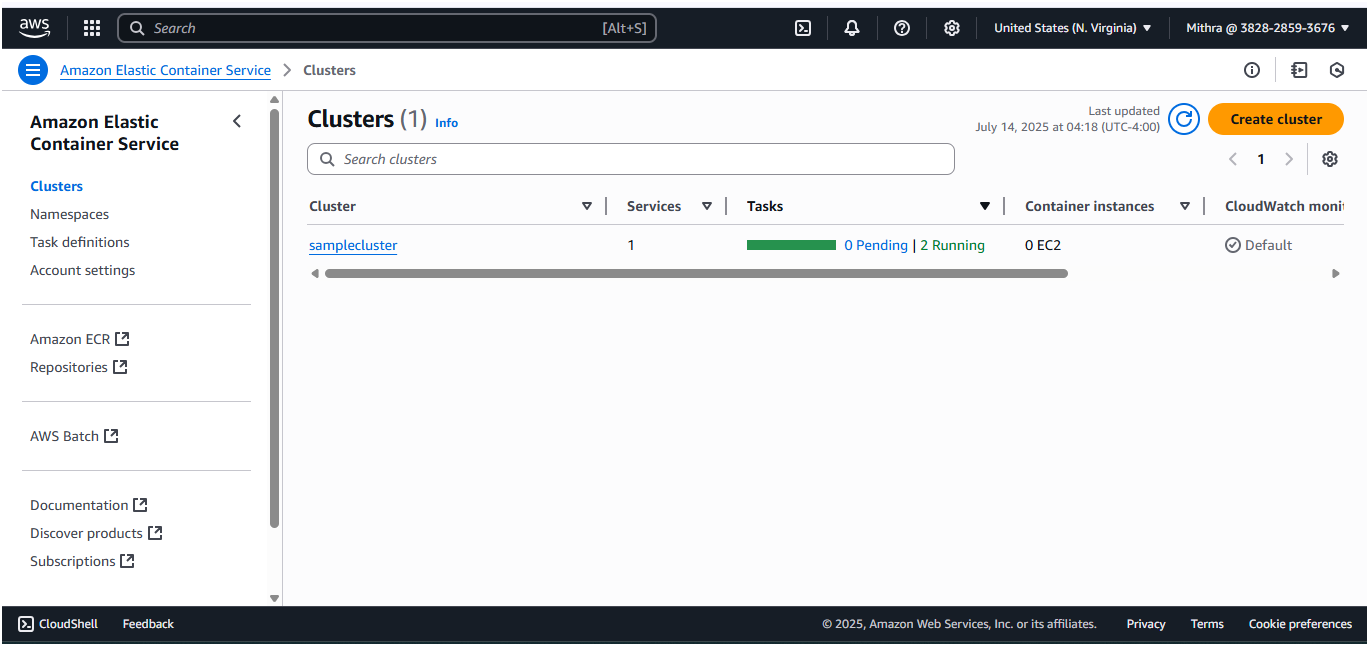
Step 1: Push the code to GitHub repo



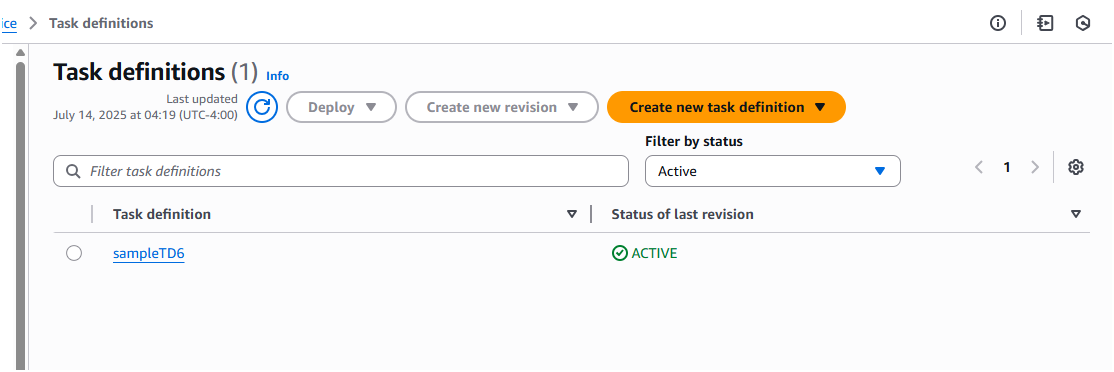
Step 2: create ECR

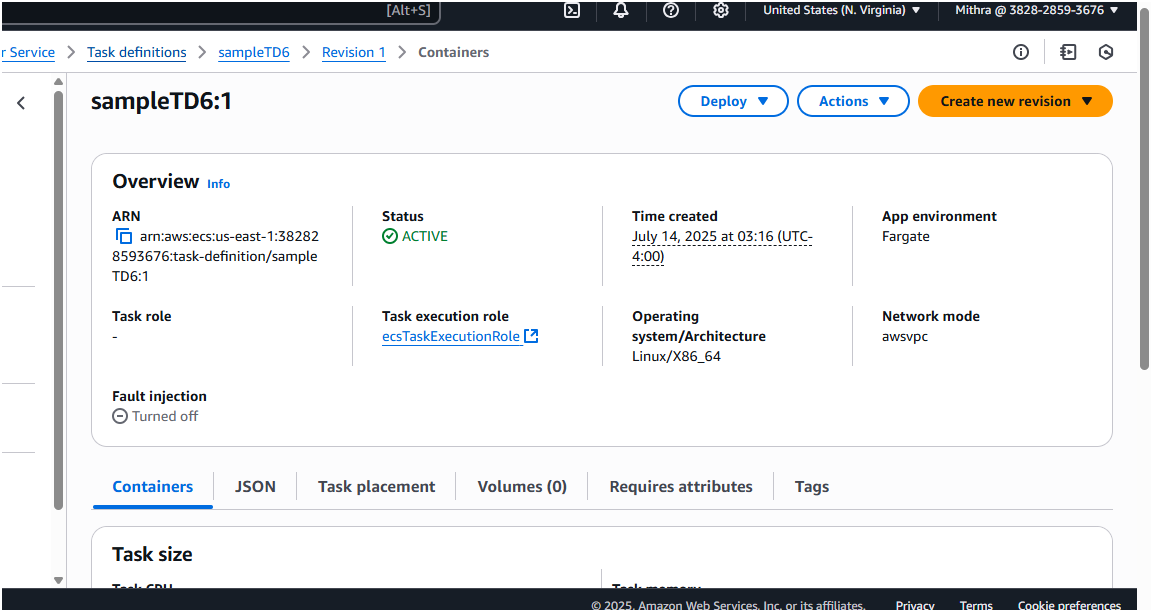


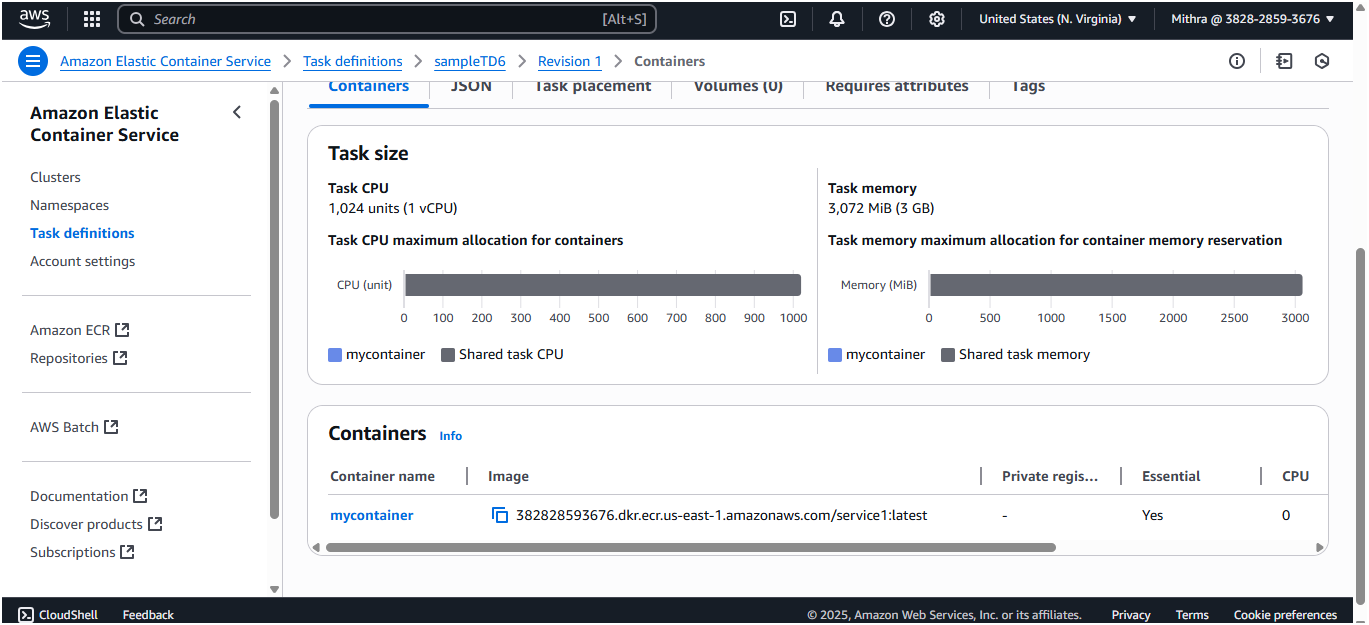
Step 3: Create ECS



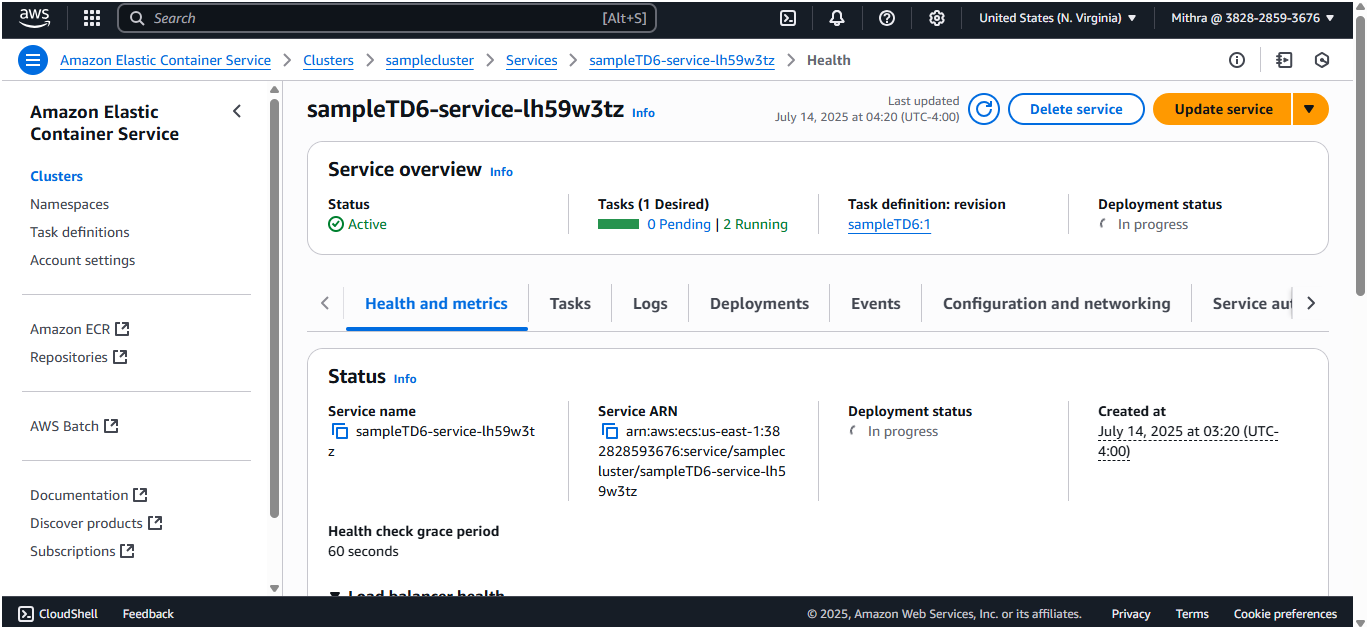
Step 4: create task definition with container port and ECR

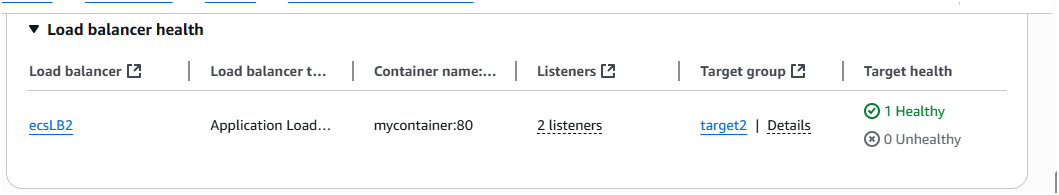


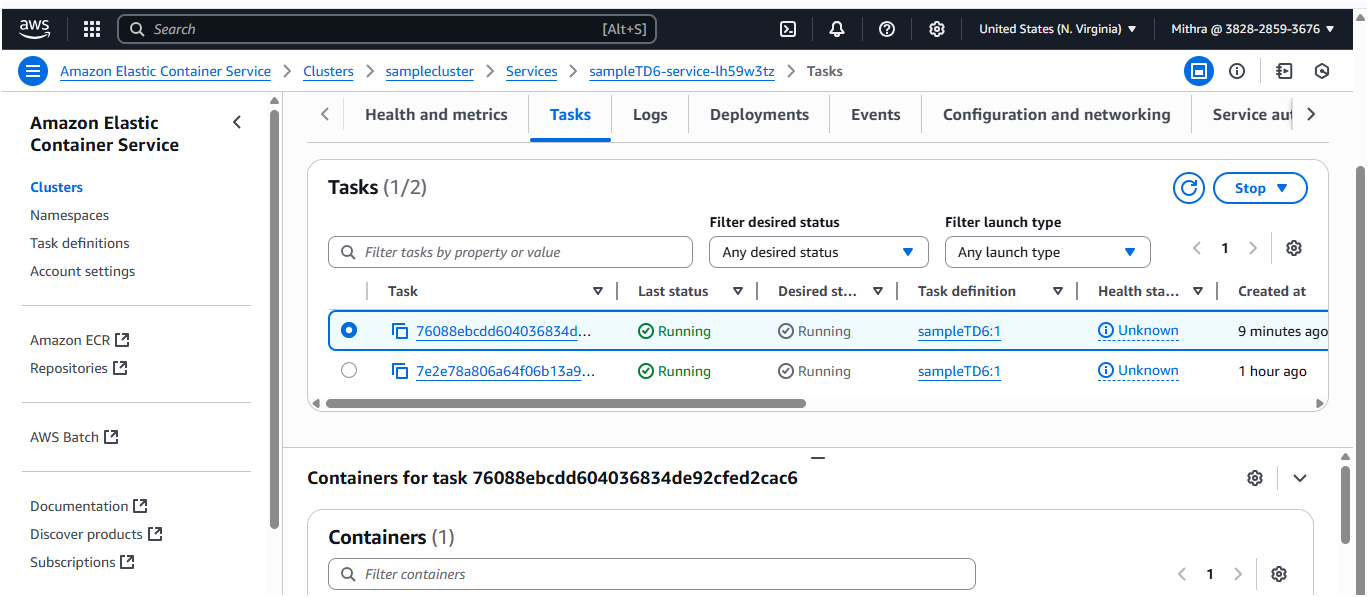


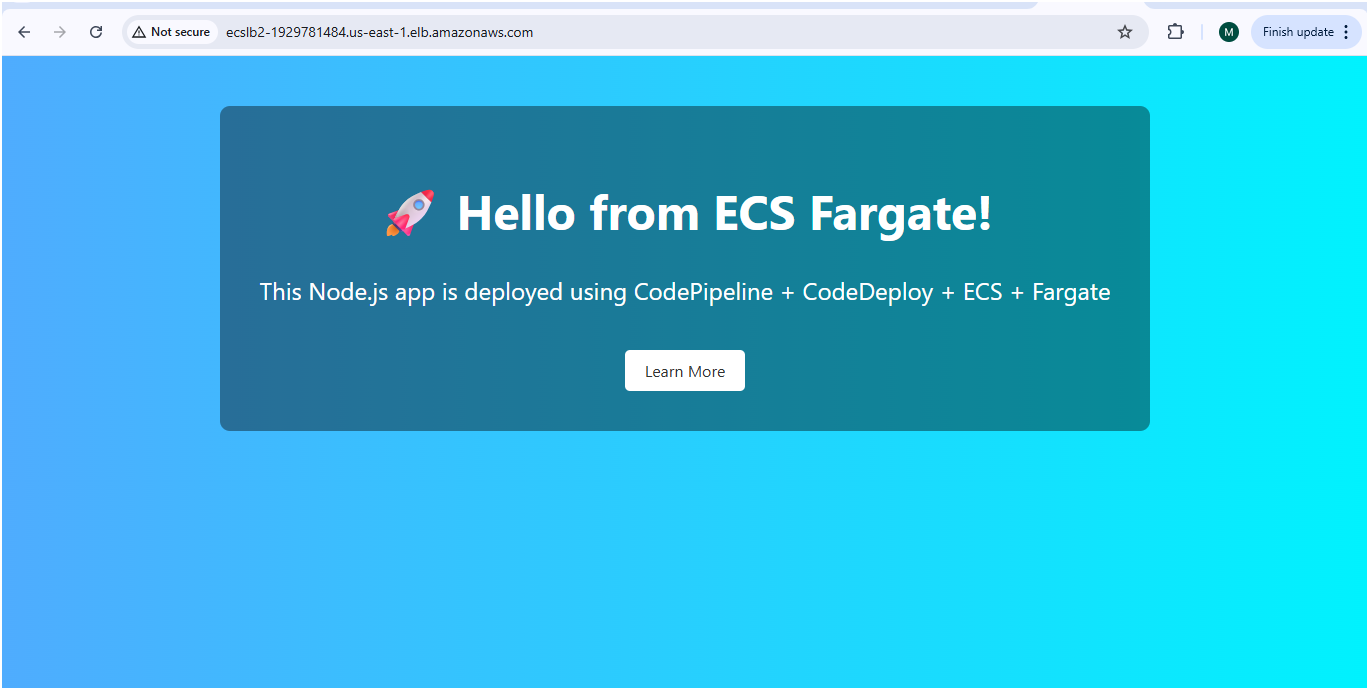


Step 5: create service for ECS

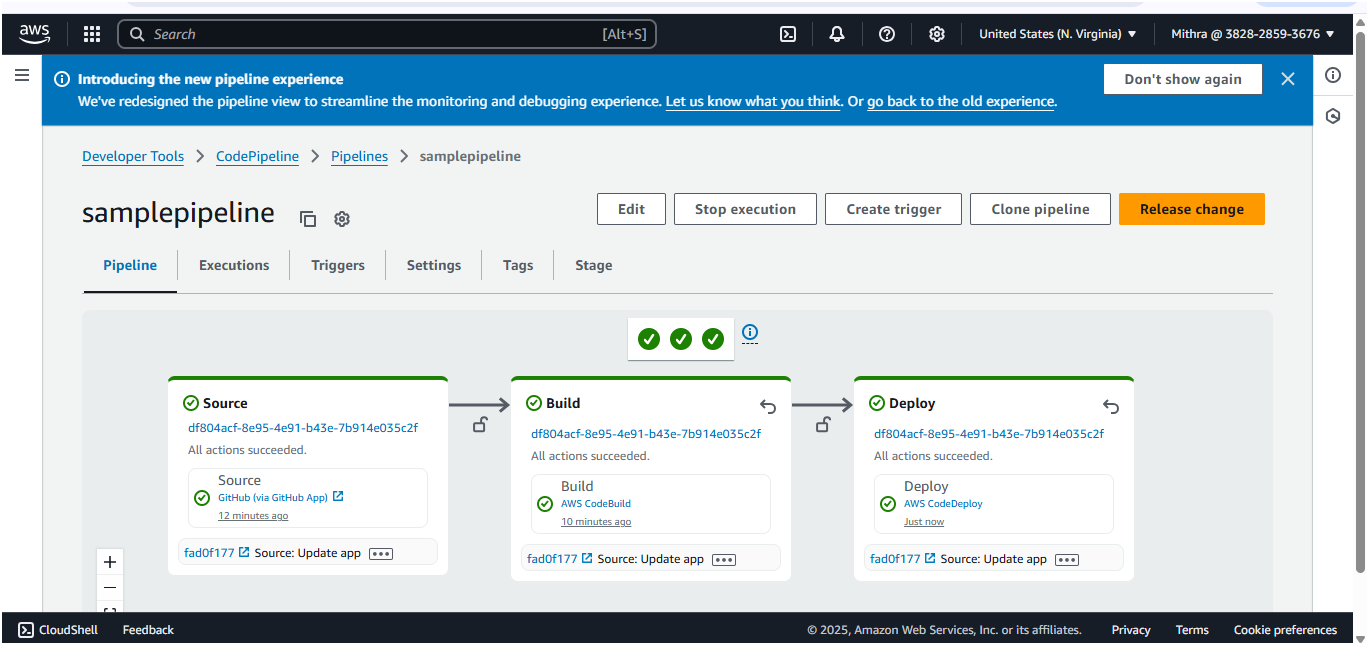


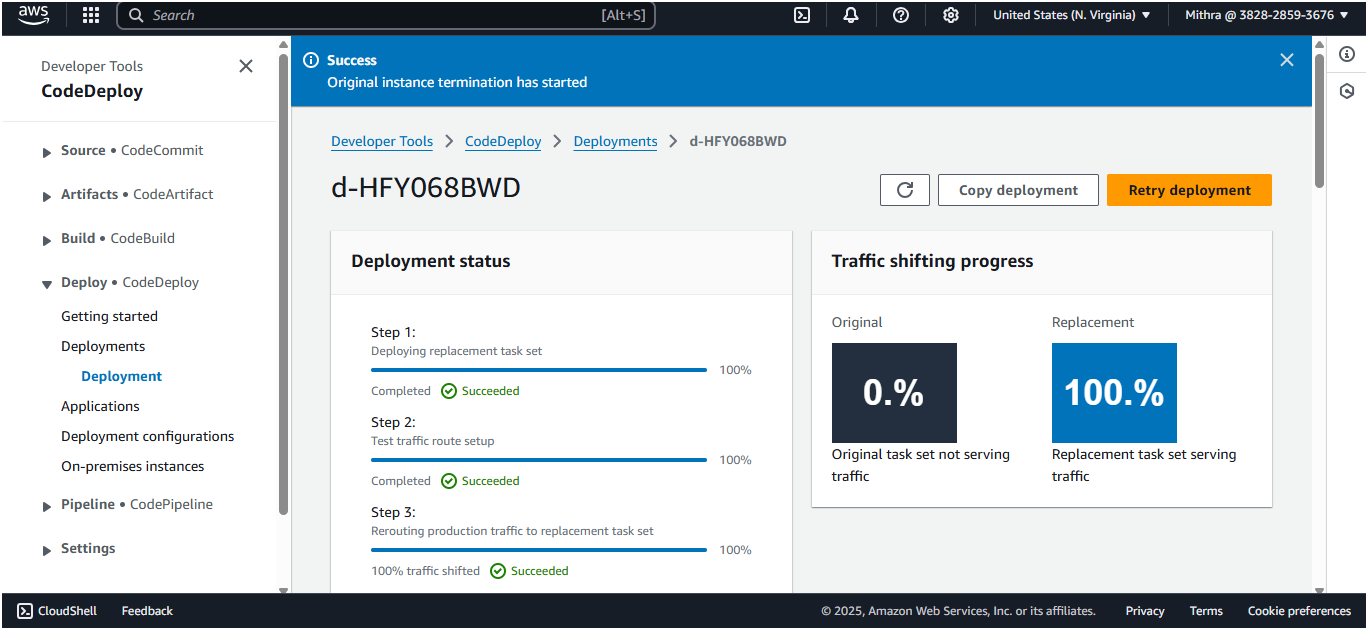




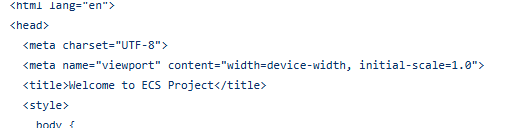


Step 6 : create code pipeline with AWS codedeploy

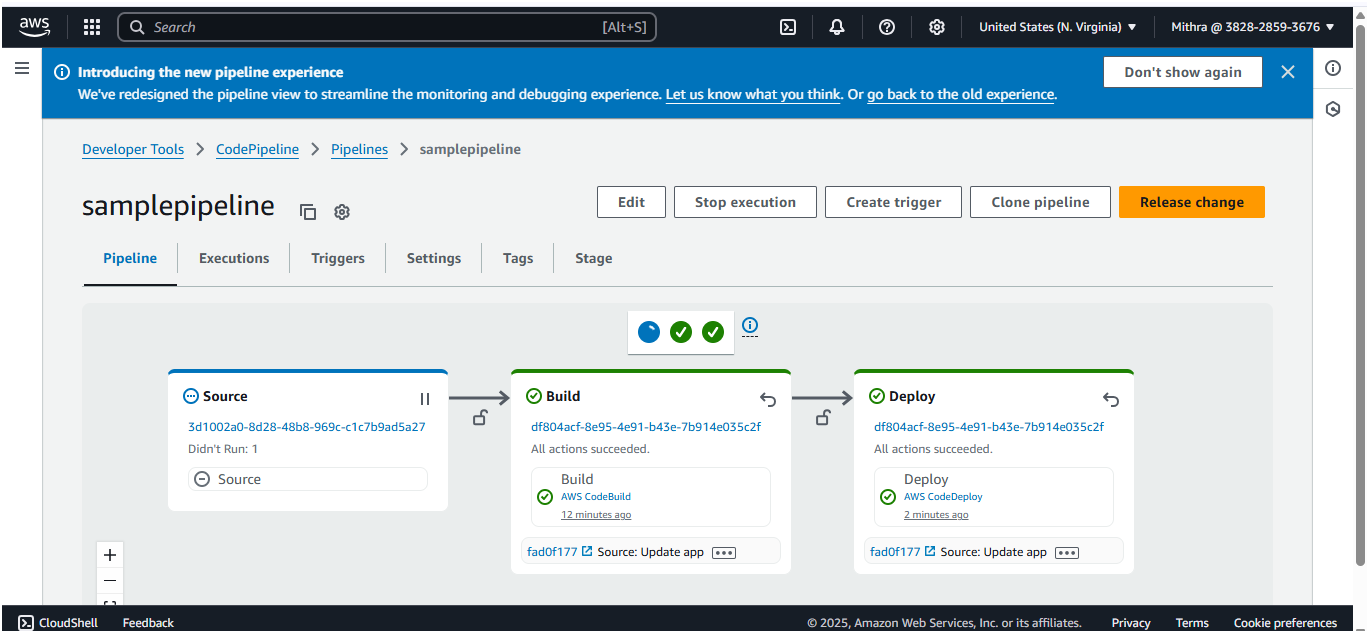




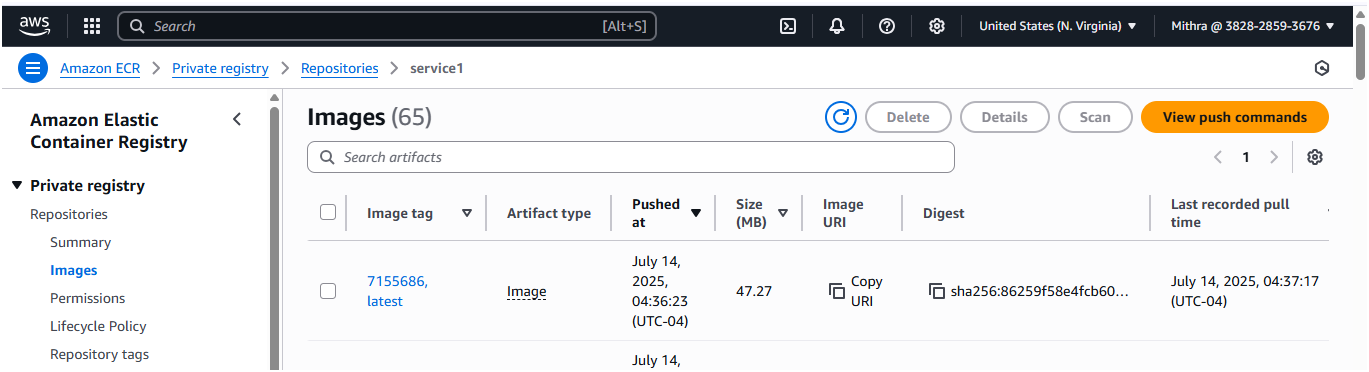
Step 7: Now change the code in github and check



Step 8: Now the pipeline triggered again



Step 9: Image is pushed to ECR



Step 10: now the traffic is moving to replacement task

