**Lab Lab 10: Continuous Deployment Script with Bash and Docker on AWS**

Objective:*Demonstrate a continuous deployment (CD) process using Bash, Docker, and AWS.*

Tasks:

1. Create a Bash script that pulls a Dockerized application from a repository.

2. Deploy the app on AWS (e.g., ECS) using the AWS CLI.

3. Implement rollback strategies in case of deployment failures.

Documentation:

- Continuous Deployment concepts.

- Integrating Bash, Docker, and AWS for a CD pipeline.

Prerequisites:

1- An AWS account with administrative access.

2- Docker Deep Dive Course

3- Bash Script Deep Dive Course

4- Complete Previous labs

Implementation Documentation:

**1. Introduction**

In this lab, we will implement a continuous deployment (CD) process using Bash, Docker, and AWS. The CD process allows us to automate the deployment of a Dockerized application from a repository to an AWS service (e.g., ECS - Elastic Container Service). We will also implement rollback strategies to handle deployment failures and ensure a smooth CD pipeline.

**2. Script Implementation**

Here's the Bash script that accomplishes the tasks:

| #!/bin/bash  # Define variables for AWS ECS deployment ecs\_cluster\_name="your\_ecs\_cluster\_name" ecs\_service\_name="your\_ecs\_service\_name" ecs\_task\_definition="your\_ecs\_task\_definition" aws\_region="your\_aws\_region"  # Define Docker application details docker\_image="ubuntu:latest"  # Function to deploy the Docker application on AWS ECS deploy\_to\_ecs() {  echo "Deploying the application to AWS ECS..."    # Update the ECS service with the new task definition  aws ecs update-service \  --cluster "$ecs\_cluster\_name" \  --service "$ecs\_service\_name" \  --region "$aws\_region" \  --task-definition "$ecs\_task\_definition" }  # Function to rollback in case of deployment failure rollback() {  echo "Rolling back the deployment..."    # Perform the rollback steps here (e.g., re-deploy the previous version)    echo "Rollback completed." }  # Main script workflow main() {  # Pull the Docker image from the repository  docker pull "$docker\_image"    # Deploy the application to AWS ECS  deploy\_to\_ecs    # Check the deployment status  if [ $? -eq 0 ]; then  echo "Deployment successful."  else  echo "Deployment failed."  # Implement rollback in case of failure  rollback  fi }  # Call the main function to start the script main |
| --- |

**Explanation of the Script**

We define variables for AWS ECS deployment, such as the ECS cluster name, service name, task definition, and AWS region.

We also specify the Docker image location in the docker\_image variable.

The deploy\_to\_ecs function updates the ECS service with the new task definition, effectively deploying the Docker application.

In the rollback function, we can implement rollback steps (not specified here) in case of deployment failure.

The main function pulls the Docker image, deploys the application to AWS ECS, and checks the deployment status. If the deployment fails, it calls the rollback function to initiate the rollback process.

**3. Running the Script**

To run the script:

Save the script in a .sh file, e.g., **cd\_pipeline.sh**.

Make the script executable by running **chmod +x cd\_pipeline.sh**.

Execute the script by running **./cd\_pipeline.sh**.

The script will pull the Docker image, deploy the application to AWS ECS, and handle deployment failures by initiating the rollback process.

**4. Conclusion**

In this lab, we have demonstrated a continuous deployment (CD) process using Bash, Docker, and AWS. CD processes automate application deployment, making it efficient and reliable. The implementation of rollback strategies in case of deployment failures ensures that the system remains resilient and can quickly recover from unexpected issues, contributing to a robust CD pipeline.