CI/CD Pipeline Explanation

Workflow Overview: This GitHub Actions workflow automates the CI/CD process for a Node.js application:

1. Run tests to make sure code works.
2. Build Docker image and push it to AWS ECR (Elastic Container Registry).
3. Deploy the application to AWS EKS (Kubernetes cluster) using Helm.

Triggers whenever code is pushed to the 'main' branch.

Environment Variables:

- AWS_REGION: AWS region (us-east-1)
- ECR_REGISTRY: Docker registry in AWS ECR
- IMAGE_NAME: Docker image name (csod-node)
- IMAGE_TAG: Tag for Docker image (Git commit SHA)
- NAMESPACE: Kubernetes namespace (csod-prod)
- HELM_CHART: Path to Helm chart (helm-chart/nodejs-app)
- RELEASE_NAME: Helm release name (node-app)

Jobs:

1. build_test:

2. Purpose: Check that the code works and passes all tests.

3. Steps: Checkout code → Setup Node.js → Install dependencies → Run tests

4. docker_build_push:

5. Purpose: Build Docker image and push to AWS ECR.

6. Depends on: build_test

7. Steps: Checkout code → Configure AWS credentials → Login to ECR → Build Docker image → Push Docker image

8. deploy:

9. Purpose: Deploy Docker image to EKS using Helm.

10. Depends on: docker_build_push
11. Steps: Checkout code → Configure AWS credentials → Install kubectl → Install Helm → Update kubeconfig → Deploy application with Helm

Workflow Diagram: Dev pushes code → GitHub Actions triggers → Run Tests → Build Docker Image → Push to AWS ECR → Deploy to AWS EKS → App updated on Kubernetes cluster

Key Points:

- Automatic workflow, no manual deployment needed.
- Uses Git commit SHA as image tag for traceability.
- Helm simplifies Kubernetes deployment.
- Stops at first failure for safety.
- AWS credentials kept secure using secrets.