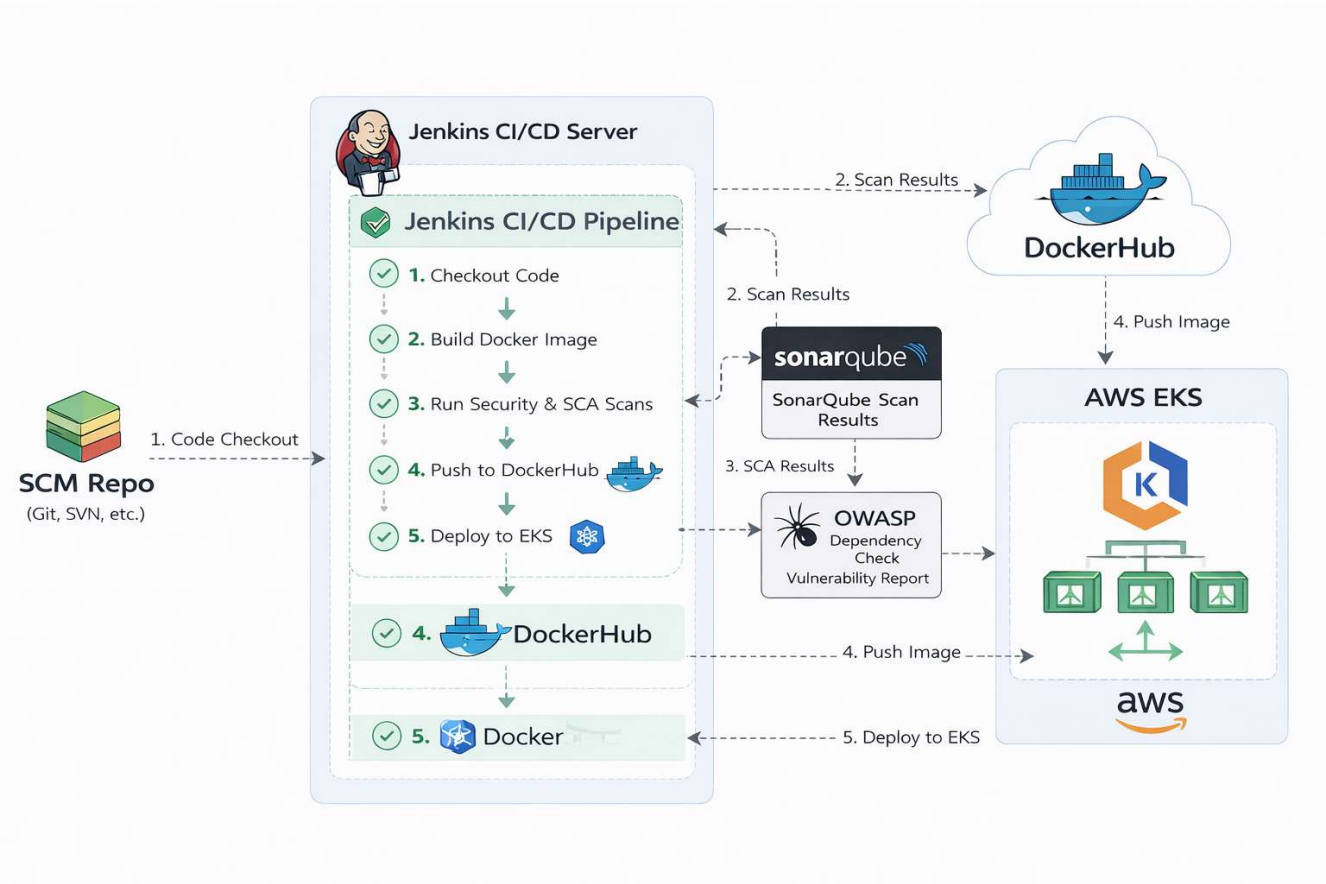
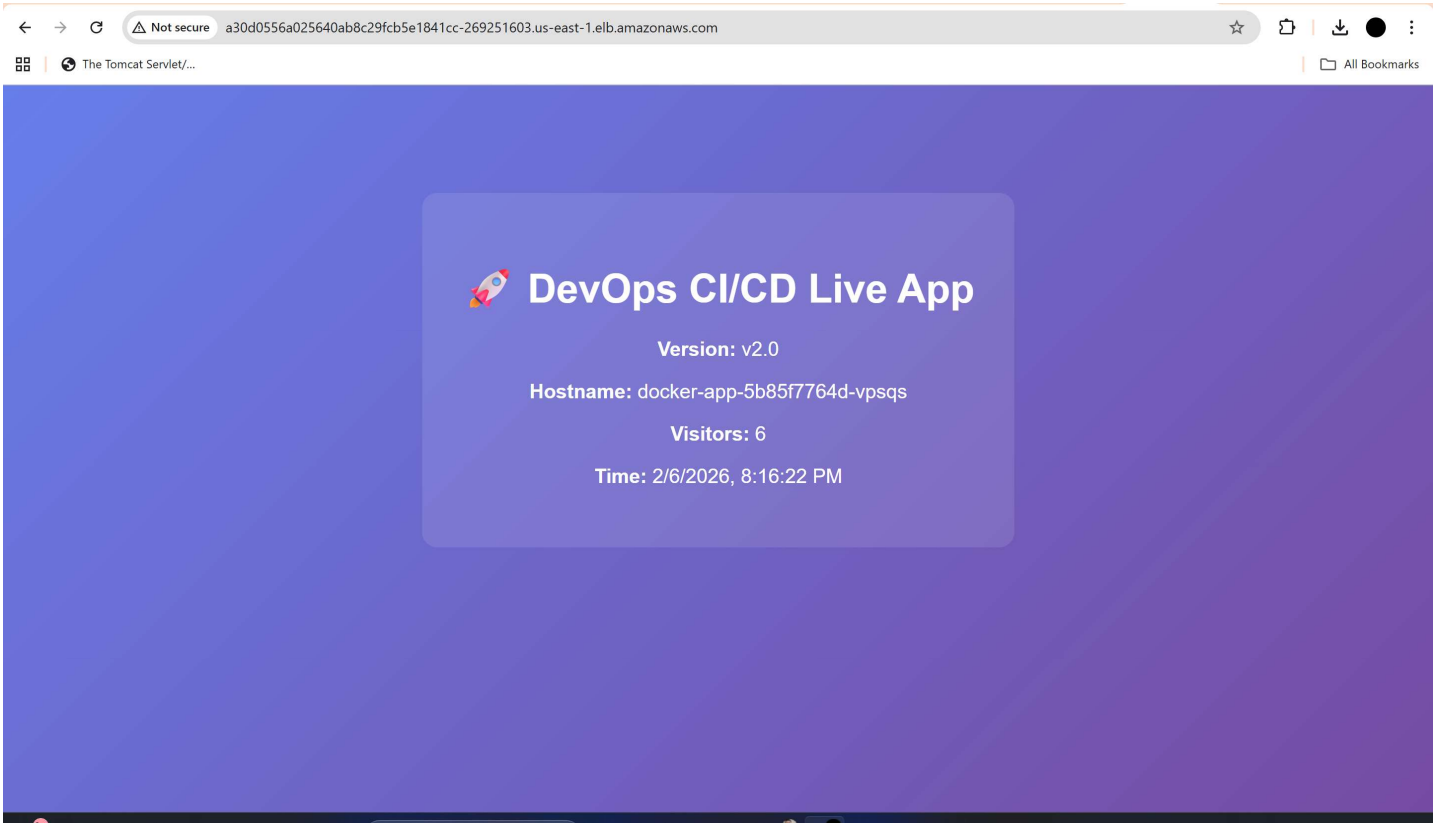
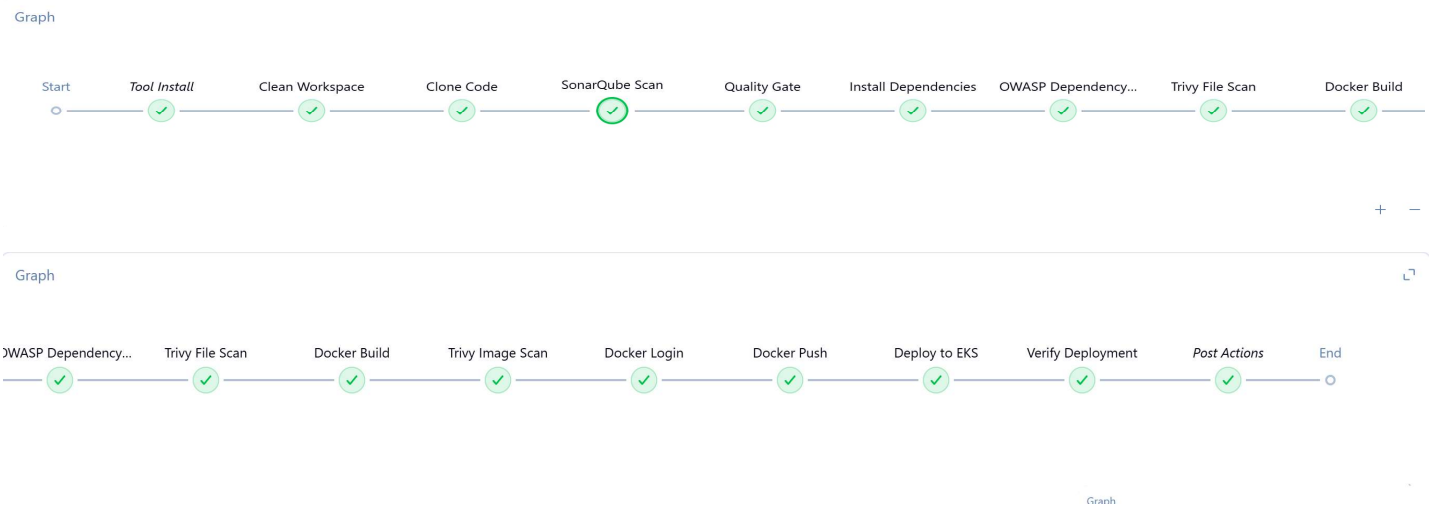
EKS + docker project

07 February 2026 09:31

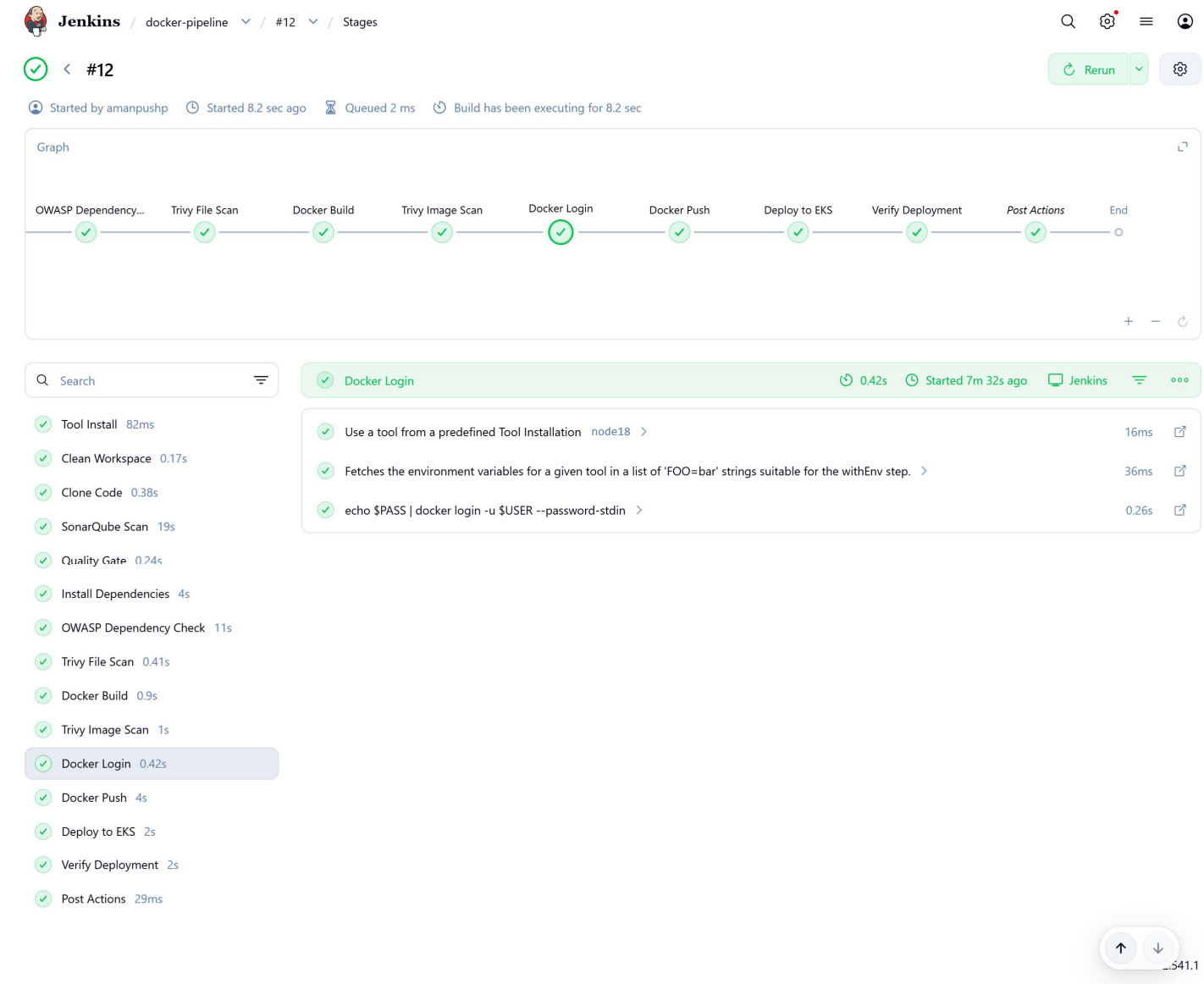
Architecture

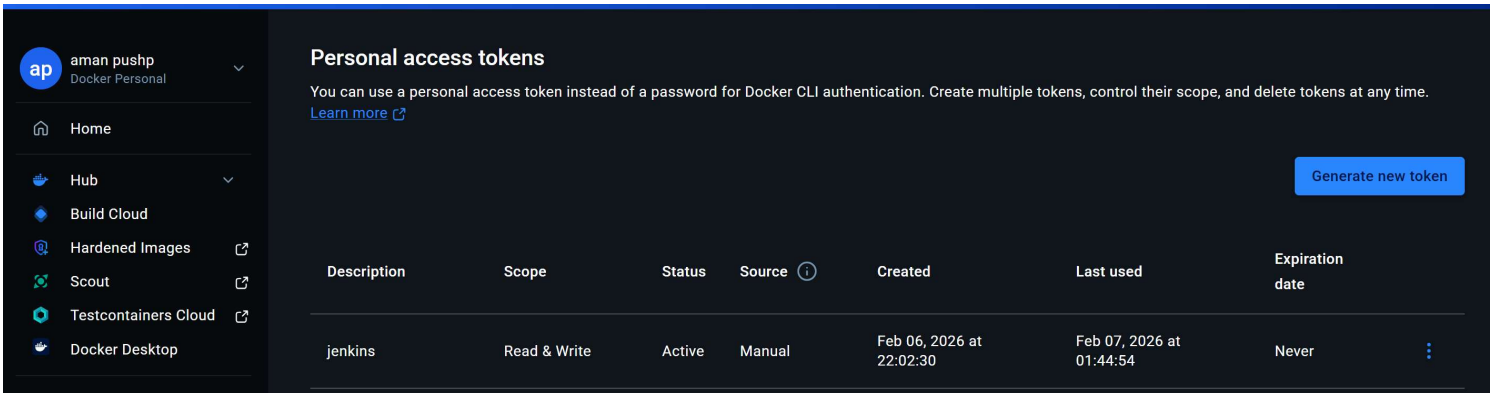



Not secure
a30d0556a025640ab8c29fcb5e1841cc-269251603.us-east-1.elb.amazonaws.com
The Tomcat Servlet/...
All Bookmarks
DevOps CI/CD Live App
Version: v2.0
Hostname: docker-app-5b85f7764d-vpsqs
Visitors: 6
Time: 2/6/2026, 8:16:22 PM


Graph
Start
Tool Install
Clean Workspace
Clone Code
SonarQube Scan
Quality Gate
Install Dependencies
OWASP Dependency...
Trivy File Scan
Docker Build

Graph
OWASP Dependency...
Trivy File Scan
Docker Build
Trivy Image Scan
Docker Login
Docker Push
Deploy to EKS
Verify Deployment
Post Actions
End





1) Final goal becomes CI/CD on AWS EKS

It is **Docker + Kubernetes (EKS) + Jenkins CI/CD**.

2) Industry CI/CD Flow Including EKS

The upgraded real-world flow shown in the document is:
GitHub → Jenkins → Docker build → Push to DockerHub → Deploy to EKS
This is explicitly described as the **real DevOps production flow used in companies**.
8b0a36cc-1836-456d-aa14-38a2e20...

3) Full EKS Deployment Roadmap in the Document

Your project includes a complete EKS phase:

Phase-1: AWS & Cluster Setup

- Create IAM user for EKS
- Install AWS CLI, kubectl, eksctl
- Create EKS cluster
- Connect kubectl to cluster
- Verify nodes running

Phase-2: Kubernetes Deployment

- Create **deployment.yaml**
- Create **service.yaml**
- Expose app using **LoadBalancer**
- Access via **public URL**

Phase-3: Jenkins → EKS CI/CD

- Jenkins builds Docker image
- Pushes to DockerHub
- **Auto-deploys to EKS using kubectl**

This is described as the **real production DevOps pipeline required for high-salary roles**.
8b0a36cc-1836-456d-aa14-38a2e20…

4)Actual Deployment Commands on EKS

After cluster becomes ready:

- Apply Kubernetes manifests
- Verify pods running
- Get external LoadBalancer URL

Which confirms the **app is live on AWS EKS**.
8b0a36cc-1836-456d-aa14-38a2e20…

6) Final Production-Level CI/CD Integration

The document concludes that your **true industry pipeline** is:

- Jenkins CI
- Security scanning
- Docker build & push
- **Deployment to Kubernetes (EKS)**
- Future: Helm, ArgoCD, Monitoring

Meaning your project is already aligned with **real company DevOps architecture**.
8b0a36cc-1836-456d-aa14-38a2e20…

7)

Final Architecture

GitHub → Jenkins → Sonar/OWASP/Trivy → Docker → DockerHub → AWS EKS → Public App

Final Project Level

Production-ready DevOps CI/CD on Kubernetes (EKS)
Interview-ready real industry project

Pipeline

```
pipeline {
  agent any

  tools {
    nodejs "node18"
  }

  environment {
    DOCKER_USERNAME = "amanpushp"
    DOCKER_IMAGE_NAME = "docker-project"
    GITHUB_REPO_URL = "https://github.com/AMANPUSHP23/docker-project.git"
    SONARQUBE_ENV = "mysonar"
  }

  stages {
```

```
stage('Clean Workspace') {
   steps { cleanWs() }
}

stage('Clone Code') {
   steps {
      git url: "${GITHUB_REPO_URL}", branch: "main"
   }
}

stage('SonarQube Scan') {
   steps {
      withSonarQubeEnv("${SONARQUBE_ENV}") {
         script {
            def scannerHome = tool 'sonar-scanner'
            sh """
            ${scannerHome}/bin/sonar-scanner \
            -Dsonar.projectName=docker-project \
            -Dsonar.projectKey=docker-project
            """
         }
      }
   }
}

stage('Quality Gate') {
   steps {
      timeout(time: 5, unit: 'MINUTES') {
         waitForQualityGate abortPipeline: false
      }
   }
}

stage('Install Dependencies') {
   steps { sh "npm install" }
}

stage('OWASP Dependency Check') {
   steps {
      sh """
      /opt/dependency-check/bin/dependency-check.sh \
      --scan . \
      --format HTML \
      --out dependency-check-report || true
      """
   }
}

stage('Trivy File Scan') {
   steps { sh "trivy fs . || true" }
}

stage('Docker Build') {
   steps {
      sh "docker build -t ${DOCKER_USERNAME}/${DOCKER_IMAGE_NAME}:latest ."
   }
}

stage('Trivy Image Scan') {
   steps {
      sh "trivy image ${DOCKER_USERNAME}/${DOCKER_IMAGE_NAME}:latest || true"
   }
}

stage('Docker Login') {
   steps {
      withCredentials([usernamePassword(
         credentialsId: 'dockerhub-creds',
         usernameVariable: 'USER',
         passwordVariable: 'PASS'
      )]) {
```

```
                sh 'echo $PASS | docker login -u $USER --password-stdin'
            }
        }
    }

    stage('Docker Push') {
        steps {
            sh "docker push ${DOCKER_USERNAME}/${DOCKER_IMAGE_NAME}:latest"
        }
    }

    //  FIXED DEPLOYMENT
    stage('Deploy to EKS') {
        steps {
            sh '''
            kubectl apply -f deployment.yaml
            kubectl apply -f service.yaml

            # Force rolling restart so new image is pulled
            kubectl rollout restart deployment docker-app
            '''
        }
    }

    //  VERIFY
    stage('Verify Deployment') {
        steps {
            sh "kubectl rollout status deployment/docker-app"
            sh "kubectl get pods -o wide"
            sh "kubectl get svc"
        }
    }
  }

  post {
    success {
        echo " PIPELINE SUCCESS — DEPLOYED TO AWS EKS "
    }
    failure {
        echo " PIPELINE FAILED — CHECK LOGS"
    }
  }
}
```

PHASE 1 — SERVER SETUP (EC2 + TOOLS)

1)Connect to EC2

ssh -i key.pem ec2-user@<EC2_PUBLIC_IP>

2)☐Install required tools

sudo yum update -y
sudo yum install -y git docker
sudo systemctl enable docker
sudo systemctl start docker
curl -o kubectl https://amazon-eks.s3.us-east-1.amazonaws.com/1.32.0/2024-11-15/bin/linux/amd64/kubectl
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_Linux_amd64.tar.gz" | tar xz
sudo mv eksctl /usr/local/bin
aws --version
kubectl version --client
eksctl version
These tools are required for **EKS cluster creation & deployment**.
8b0a36cc-1836-456d-aa14-38a2e20…

PHASE 2 — JENKINS CI/CD BASE

3) Clone project

git clone https://github.com/AMANPUSHP23/docker-project.git
cd docker-project

5) Jenkins pipeline already performs:

- SonarQube scan
- OWASP scan
- Trivy scan
- Docker build
- Docker push

This completes **secure CI pipeline** before EKS deployment.
8b0a36cc-1836-456d-aa14-38a2e20…

PHASE 3 — AWS IAM + EKS CREATION

6) Verify IAM role attached

aws sts get-caller-identity
Must show **assumed-role/jenkins-eks-role**.
8b0a36cc-1836-456d-aa14-38a2e20…

6) Create EKS cluster

eksctl create cluster \
--name aman-eks-cluster \
--region us-east-1 \
--node-type t3.small \
--nodes 1
Takes ~20 minutes while CloudFormation builds control plane + nodes.
8b0a36cc-1836-456d-aa14-38a2e20…

7)☐Verify cluster ready

kubectl get nodes
STATUS must be **Ready** → cluster working.
8b0a36cc-1836-456d-aa14-38a2e20…

PHASE 4 — KUBERNETES DEPLOYMENT

Apply deployment & service

kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
Deploys Docker container inside EKS.
8b0a36cc-1836-456d-aa14-38a2e20…

Verify pods running

kubectl get pods
STATUS = **Running** confirms successful deployment.
8b0a36cc-1836-456d-aa14-38a2e20…

Get public application URL

kubectl get svc docker-app-service
EXTERNAL-IP = live AWS website.
8b0a36cc-1836-456d-aa14-38a2e20…

PHASE 5 — FINAL JENKINS → EKS AUTO-DEPLOY

Deploy stage used in Jenkins

kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
kubectl rollout restart deployment docker-app
Enables **automatic CI/CD deployment to EKS**.
8b0a36cc-1836-456d-aa14-38a2e20…

PHASE 6 — CLEANUP (IMPORTANT)

Delete cluster after practice

eksctl delete cluster --name aman-eks-cluster --region us-east-1