React App CI/CD Automation on AWS EKS — Full Technical Documentation

Complete, detailed project documentation including code snippets, task-by-task explanations, operational runbook, troubleshooting, security, and IaC details.

Table of Contents

1. Project Summary

Goal: Build a repeatable CI/CD pipeline for a React application (`Trend` repo) that produces production-ready Docker images, stores them on DockerHub, provisions/controls cluster state via Terraform (importing an existing EKS), and deploys workloads to EKS via Jenkins with safe controls.

Deliverables:
Dockerfile and optimized image build
Kubernetes manifests for Deployment + Service (LoadBalancer)
Declarative Jenkins pipeline (build → push → terraform plan/apply → deploy)
Terraform configuration with imported EKS & nodegroup stabilized (ignore rules)
Operational runbook and documentation (this document)
Troubleshooting and security guidance

2. Architecture Overview

Developer -> GitHub (webhook) -> Jenkins (multibranch or pipeline)
Jenkins builds Docker image -> pushes to DockerHub
Jenkins runs `terraform plan` (approval gate required to apply) -> updates infra as needed
Jenkins runs `aws eks update-kubeconfig` -> runs `kubectl apply -f k8s/` to deploy workloads to EKS
EKS runs Pods, Service exposes LoadBalancer to internet

Key identity model: Jenkins machine uses EC2 **IAM Role** to access AWS APIs (no static keys).

3. Repository Layout (recommended)

```
Trend/
■■■ app/                      # React app source (src, public)
■■■ package.json
■■■ Dockerfile
■■■ .dockerignore
■■■ Jenkinsfile
■■■ k8s/
■   ■■■ deployment.yaml
■   ■■■ service.yaml
■■■ tf/
■   ■■■ provider.tf
■   ■■■ main.tf
■   ■■■ variables.tf
■   ■■■ outputs.tf
■■■ README.md
```

4. Prerequisites

AWS account & permissions to manage EKS, IAM, EC2, VPC
Jenkins controller & agent(s) that can run Docker, Terraform, aws-cli, kubectl
DockerHub account & repo
GitHub repository and webhook access
Local dev: Node.js, npm for building the React app
Tools installed on Jenkins agent: `docker`, `aws` (v2), `kubectl`, `terraform`

5. Key Components & Files (with code)

Dockerfile (production)

```
# Build stage
FROM node:18-alpine AS build
WORKDIR /app
COPY package*.json ./
RUN npm ci --prefer-offline --no-audit --progress=false
COPY . .
RUN npm run build

# Production stage - serve with nginx
FROM nginx:alpine
RUN rm -rf /usr/share/nginx/html/*
COPY --from=build /app/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

.dockerignore

```
node_modules
dist
.git
.gitignore
Dockerfile
README.md
```

Kubernetes manifests (k8s/deployment.yaml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: trend-app
  labels:
```

```
    app: trend-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: trend-app
  template:
    metadata:
      labels:
        app: trend-app
    spec:
      containers:
      - name: trend-app
        image: prem18062000/trend-app:latest
        ports:
        - containerPort: 80
        readinessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 10
```

k8s/service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: trend-app-service
spec:
  type: LoadBalancer
  selector:
    app: trend-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

Jenkinsfile (full pipeline — safe, with terraform)

```
pipeline {
  agent any
  environment {
    DOCKERHUB_USERNAME = "prem18062000"
    IMAGE_NAME = "trend-app"
    AWS_REGION = "ap-south-1"
    EKS_CLUSTER_NAME = "trend-eks"
    TF_DIR = "tf"
  }
  stages {
    stage("Checkout") { steps { checkout scm } }
    stage("Terraform Init") {
      steps { dir("${TF_DIR}") { sh 'terraform init -input=false' } }
    }
    stage("Terraform Plan") {
      steps { dir("${TF_DIR}") { sh 'terraform plan -out=tfplan' } }
    }
    stage("Terraform Apply (Manual Approval)") {
      when { expression { env.BRANCH_NAME == "main" } }
      input { message "Apply Terraform changes?"; ok "Apply" }
      steps { dir("${TF_DIR}") { sh 'terraform apply -auto-approve tfplan' } }
    }
    stage("Build Image") {
      steps { sh "sudo docker build -t ${DOCKERHUB_USERNAME}/${IMAGE_NAME}:latest ." }
    }
    stage("Push Image") {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-creds', usernameVariable: 'DOCKER_US
          sh '''
```

```
            echo ${DOCKER_PASS} | sudo docker login -u ${DOCKER_USER} --password-stdin
            sudo docker push ${DOCKERHUB_USERNAME}/${IMAGE_NAME}:latest
          '''
        }
      }
    }
    stage("Deploy to EKS") {
      steps {
        sh '''
          aws eks update-kubeconfig --region ${AWS_REGION} --name ${EKS_CLUSTER_NAME}
          kubectl apply -f k8s/
        '''
      }
    }
  }
}
```

Terraform provider (`tf/provider.tf`)

```
terraform {
  required_providers {
    aws = { source = "hashicorp/aws" }
  }
}

provider "aws" {
  region = "ap-south-1"
}
```

Terraform EKS skeleton (imported) — main.tf snippet

```
resource "aws_eks_cluster" "trend" {
  name     = "trend-eks"
  role_arn = "arn:aws:iam::309539410867:role/eksctl-trend-eks-cluster-ServiceRole-F6DQvVDyF8hR"
  version  = "1.29"

  vpc_config {
    subnet_ids = [
      "subnet-00575efd60467548b",
      "subnet-023b5987d82a6e277",
      "subnet-05807c99c0c96e7a6",
      "subnet-0a3994c6a2ae813e3",
      "subnet-0c9ad69d505b7a28d",
      "subnet-0ffbbd5eb0d3f1dc5",
    ]
  }

  lifecycle {
    ignore_changes = [
      tags,
      vpc_config[0].security_group_ids
    ]
  }

  tags = {
    Name = "eksctl-trend-eks-cluster/ControlPlane"
    "alpha.eksctl.io/cluster-name" = "trend-eks"
  }
}
```

Terraform nodegroup skeleton (imported)

```
resource "aws_eks_node_group" "trend_workers" {
  cluster_name    = "trend-eks"
  node_group_name = "trend-workers"
  node_role_arn   = "arn:aws:iam::309539410867:role/eksctl-trend-eks-nodegroup-trend-w-NodeInstanceRole-
  subnet_ids = [
    "subnet-00575efd60467548b",
```

```
      "subnet-023b5987d82a6e277",
      "subnet-05807c99c0c96e7a6",
      "subnet-0a3994c6a2ae813e3",
      "subnet-0c9ad69d505b7a28d",
      "subnet-0ffbbd5eb0d3f1dc5",
    ]
    scaling_config {
      desired_size = 2
      min_size     = 1
      max_size     = 3
    }
    lifecycle {
      ignore_changes = [
        subnet_ids,
        scaling_config,
        labels,
        tags,
        launch_template,
        update_config,
        instance_types,
        ami_type,
      ]
    }
  }
```

6. Detailed Task List — what was done, step-by-step, and verification

(omitted here for brevity in PDF/Word generation - full details included in the markdown file available in the repo)

7. Deployment & Runbook (detailed step-by-step)

(see the markdown file for full runbook - include pre-deployment checks, manual & automated flows)

8. Terraform Import & Stabilization (deep dive)

(see markdown for full detail - includes lifecycle ignore recommendations and verification steps)

9. Jenkins Integration Details & Tips

(see markdown)

10. Monitoring, Logging & Alerts (recommendations)

(see markdown)

11. Troubleshooting Guide (common errors & fixes)

(see markdown)

12. Security & Best Practices

(see markdown)

13. Post-deployment checklist & Handover

(see markdown)

14. Appendix: Useful commands & links

```
aws sts get-caller-identity
aws eks update-kubeconfig --region ap-south-1 --name trend-eks
kubectl get nodes
kubectl get pods -A
kubectl logs <pod-name>
terraform init
terraform plan
terraform apply -auto-approve tfplan
```
