### **Setting Up a DevOps Pipeline for the Institute Management System**

We will build a **fully automated DevOps pipeline** for our **Institute Management System** using modern DevOps tools. This pipeline will automate **code integration, testing, deployment, and monitoring**.

## **DevOps Pipeline Architecture**

### **Tools Used:**

* **Version Control:** GitHub
* **CI/CD Pipeline:** Jenkins + GitHub Actions
* **Containerization:** Docker
* **Orchestration:** Kubernetes
* **Infrastructure as Code:** Terraform

## **Step 1: Set Up a GitHub Repository**

**Create a GitHub repository** for storing the project code.

* **Branching Strategy:** Use **main** for production, **develop** for testing, and feature branches for individual work.
* Enable **GitHub Actions** for automation.

git init  
git remote add origin <https://github.com/your-repo/institute-management-system.git>git add .  
git commit -m "Initial commit"  
git push origin main

## **Step 2: Configure Jenkins for CI/CDCD**

**Install Jenkins** and required plugins:

* Install **Git**, **Docker**, and **Kubernetes** plugins in Jenkins.
* Configure a **Jenkins pipeline**.

### **Jenkinsfile for CI/CD**

Create a file named **Jenkinsfile** in the repo. (groovy)

pipeline {  
 agent any  
  
 stages {  
 stage('Clone Repository') {  
 steps {  
 git branch: 'main', url: '<https://github.com/your-repo/institute-management-system.git>'  
 }  
 }  
  
 stage('Build') {  
 steps {  
 sh 'docker build -t institute-management-system .'  
 }  
 }  
  
 stage('Test') {  
 steps {  
 sh 'docker run --rm institute-management-system pytest tests/'  
 }  
 }  
  
 stage('Security Scan') {  
 steps {  
 sh 'sonar-scanner -Dsonar.projectKey=InstituteManagement -Dsonar.host.url=http://sonarqube:9000'  
 }  
 }  
  
 stage('Push to Docker Hub') {  
 steps {  
 withDockerRegistry([credentialsId: 'docker-hub-cred', url: '<https://index.docker.io/v1/>']) {  
 sh 'docker tag institute-management-system your-dockerhub-username/institute-management-system'  
 sh 'docker push your-dockerhub-username/institute-management-system'  
 }  
 }  
 }  
  
 stage('Deploy to Kubernetes') {  
 steps {  
 sh 'kubectl apply -f k8s/deployment.yaml'  
 }  
 }  
 }  
}

## **Step 3: Create a Dockerfile**

**Containerize the application using Docker** Create a Dockerfile in the project root (dockerfile)

FROM node:18  
  
WORKDIR /app  
  
COPY package.json ./  
RUN npm install  
  
COPY . .   
  
EXPOSE 3000  
CMD ["npm", "start"]

### **Build and Run Locally**

sh

docker build -t institute-management-system .  
docker run -p 3000:3000 institute-management-system

## **Step 4: Deploy to Kubernetes**

**Create a Kubernetes deployment configuration**

Create a k8s/deployment.yaml file: (yaml)

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: institute-management  
spec:  
 replicas: 2  
 selector:  
 matchLabels:  
 app: institute-management  
 template:  
 metadata:  
 labels:  
 app: institute-management  
 spec:  
 containers:  
 - name: institute-management  
 image: your-dockerhub-username/institute-management-system  
 ports:  
 - containerPort: 3000

### **Deploy to Kubernetes**

sh

kubectl apply -f k8s/deployment.yaml

## **Step 5: Infrastructure as Code (Terraform)**

**Use Terraform to deploy AWS infrastructure**

Create a terraform/main.tf file: (hcl)

provider "aws" {  
 region = "us-east-1"  
}  
  
resource "aws\_instance" "web" {  
 ami = "ami-12345678"  
 instance\_type = "t2.micro"  
}

### **Deploy Infrastructure**

sh

terraform init  
terraform apply