PHASE 5- DEPLOYMENT

PROJECT TITLE :- Setting up a CI/CD pipeline for automated deployment

COLLEGE NAME:-Dr. SMCE

GROUP MEMBERS:-

1. Amith C Y [USN:- 1CC21CS004]
2. Charan K M [USN:- 1CC21CS017]
3. Nitin Teja N[USN:-1CC21CS040]
4. Sunil C M [USN:-1CC21CS057]

Goal: Automatically deploy your secure app to Kubernetes using Jenkins, while enforcing security best practices and monitoring deployments.

Step 1: Set Up Jenkins for Automated CI/CD Deployment

- **Tool:** Jenkins
- **Where:** Run Jenkins via Docker or as a service on Ubuntu

- **Jenkins Setup (Docker version):**

Command:

```
docker run -d --name jenkins \
        -p 8081:8080 -p 50000:50000 \
        -v jenkins_home:/var/jenkins_home \
         jenkins/jenkins:lts
```

- Access Jenkins:
  - URL: http://localhost:8081
  - Login: admin / [initial password from terminal]
  - Install “Suggested Plugins”
  - Install required plugins: Docker, Pipeline, Kubernetes CLI, Git, Blue Ocean (optional)

- Add Credentials:
  - Docker Hub username/password
  - Git repo credentials (if needed)
  - Kubernetes kubeconfig (optional, for kubectl access)

Step 2: Configure Jenkins Pipeline (Jenkinsfile)

- **Where:** In the root of your Node.js project

❖ **Jenkinsfile example:**

Code:

```
pipeline {
  agent any
  environment {
    IMAGE_NAME = "secure-todo"
  }
  stages {
    stage('Checkout Code') {
      steps {
        git 'https://github.com/your-repo/secure-todo.git'
      }
    }
    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $IMAGE_NAME .'
      }
    }
    stage('Trivy Scan') {
      steps {
        sh 'trivy image $IMAGE_NAME'
      }
    }
    stage('Run Unit Tests') {
      steps {
        sh 'npm install && npm test || true'
```

```
          }
        }
        stage('Deploy to Kubernetes') {
          steps {
            sh 'kubectl apply -f k8s/deployment.yaml'
          }
        }
        stage('Verify Deployment') {
          steps {
            sh 'kubectl get pods'
          }
        }
      }
    }
```

- **Where to Add:**
  - Jenkins Dashboard → New Item → Pipeline → Configure SCM and Jenkinsfile
- **Security Considerations:**
  - Store Docker and Kubernetes credentials in Jenkins → Manage Credentials
  - Set Jenkins to fail pipeline on scan/test errors
  - Restrict who can modify the Jenkinsfile

Step 3: Deploy to Kubernetes (via Minikube or Cluster)

- **Tool:** Minikube or any K8s cluster

- **Create deployment.yaml:**
  File: k8s/deployment.yaml

  Code:

```
apiVersion: apps/v1

kind: Deployment
```

```
metadata:
  name: secure-todo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: secure-todo
  template:
    metadata:
      labels:
        app: secure-todo
    spec:
      containers:
      - name: secure-todo
        image: secure-todo
        ports:
        - containerPort: 8080
```

- **Create service.yaml (optional):**
  File: k8s/service.yaml

  Code:

```
apiVersion: v1
kind: Service
metadata:
  name: secure-todo-service
spec:
  selector:
    app: secure-todo
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: NodePort
```

- **Apply the deployment:**

  Command:

  ```
  kubectl apply -f k8s/deployment.yaml
  kubectl apply -f k8s/service.yaml
  ```

- **Verify:**

  Command:

  ```
  kubectl get pods
  kubectl get services
  ```

- **Access app:**

  Command:

  ```
  minikube service secure-todo-service
  ```

Step 4: Monitor & Audit Deployment

- **Tools:** ELK Stack, Audit Logs, Kubernetes Events

- **ELK Stack (Log Monitoring):**
  - E = Elasticsearch (stores logs)
  - L = Logstash (parses logs)
  - K = Kibana (visualizes logs)

Run via Docker Compose or use ELK hosted solution.

- **Jenkins Auditing:**
  - Install "Audit Trail" plugin
  - Logs changes to pipeline, environment, credentials, etc.

- **Kubernetes Monitoring:**
  - Use commands like:

    Command:

    ```
    kubectl get events

    kubectl logs <pod-name>
    ```

  - Use Prometheus + Grafana for metrics and health checks.

- **Alerting:**
  - Use tools like Slack plugin, Prometheus Alertmanager, or email for build/deploy failures

Step 5: Post-Deployment Security Validation

After deployment, run final security validations to ensure integrity.

- **Final Trivy Scan (on live image):**

  Command:

  ```
  trivy image secure-todo
  ```

- **OWASP ZAP Scan (on live app):**

  Command:

  ```
  docker run -t owasp/zap2docker-stable zap-baseline.py \
    -t http://localhost:8080 -r final-zap-report.html
  ```

- **Checkov Scan (for infrastructure again):**

  Command:

  ```
  checkov -d ./k8s/
  ```

- **Log Monitoring:**
  - Review Kibana dashboard
  - Confirm no unauthorized access or errors

Summary: What You Achieve in Phase 5

Task	**Outcome**
Jenkins pipeline setup	Fully automated build/test/deploy flow
Docker + Kubernetes deployment	Scalable container orchestration
Security scanning integration	Fail pipeline on vulnerabilities
Monitoring/logging	Real-time insights into app + infra
Secrets + role protection	Prevent credential leaks + least privilege access