Full-Stack DevOps Assignment Documentation

Table of Contents

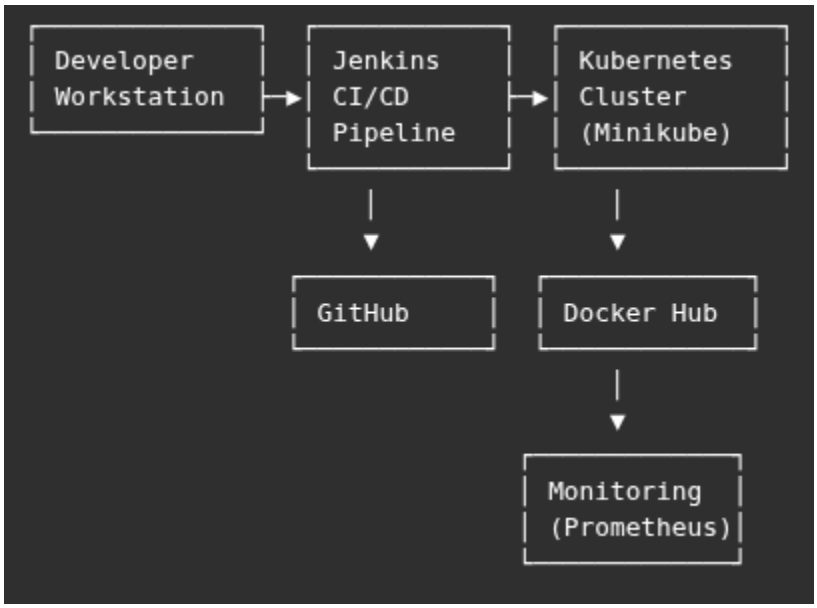
Project Overview

This project demonstrates a comprehensive DevOps pipeline integrating:

- Java Spring Boot backend with Prometheus metrics

- Docker-based containerization
- Jenkins for CI/CD automation
- Kubernetes (Minikube) for orchestration
- Helm for deployment packaging
- Prometheus & Grafana for monitoring
- OPA Gatekeeper for security enforcement

🏗️ Architecture



Prerequisites

Required Tools (Verified)

- Docker (v28.3.0)

- kubectl (v1.29.15)
- Minikube (v1.35.0)
- Helm (v3.18.3)
- Java OpenJDK (21.0.7)
- Jenkins (localhost:8080)
- Git (v2.43.0)

Jenkins Configuration

- URL: `http://localhost:8080/`
- User: `samarth`
- API Token: `11b89b2f9fedf58e8095f2b7a336643952`

Quick Start Guide

```
# Clone and enter project directory
git clone <repository-url>
cd devops-assignment

# Run locally
cd app && mvn spring-boot:run

# Build Docker image
cd app && docker build -t devops-app:latest .

# Deploy to Kubernetes
./scripts/deploy.sh

# Port forward and access
kubectl port-forward service/devops-challenge 8082:80 -n devops-challenge
curl http://localhost:8082/api
```

Implementation Details

Environment Check

- Docker: Running
- Minikube: Running
- Jenkins: Accessible
- CLI Tools: Installed

Project Structure

```
.
├── app/
├── helm-chart/
├── .github/workflows/
├── k8s-deployment/
├── jenkins-cli/
├── scripts/
├── Dockerfile
└── README.md
```

API Service

mvn archetype:generate -DgroupId=com.devops.challenge -DartifactId=devops-app

Endpoints:

- `GET /api`
- `GET /api/health`
- `GET /actuator/prometheus`

Dockerization

docker build -t devops-app:latest .

docker run -p 8080:8080 devops-app:latest

Security Scan: Trivy

Jenkins CI/CD Pipeline

Download CLI
wget http://localhost:8080/jnlpJars/jenkins-cli.jar

Create and run job
java -jar jenkins-cli.jar -s http://localhost:8080/ -auth samarth:<API_TOKEN> create-job devops-pipeline < jenkins-cli/pipeline-config.xml
java -jar jenkins-cli.jar -s http://localhost:8080/ -auth samarth:<API_TOKEN> build devops-pipeline

Helm Chart

helm create devops-chart
helm install devops-release ./helm-chart/devops-chart
helm test devops-release

Kubernetes Deployment

kubectl apply -f k8s-deployment/
kubectl get pods,services,ingress
minikube service devops-service --url

OPA Security (Bonus)

kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper/release-3.14/deploy/gatekeeper.yaml
kubectl apply -f k8s-deployment/opa-policies/

API Documentation

GET /api

Returns request headers, method, body.

POST /api

Accepts and returns JSON payload.

GET /api/health

Health status of service.

Example:


```
{
  "status": "UP",
  "timestamp": "2024-01-01T12:00:00",
  "service": "DevOps Challenge API",
  "version": "1.0.0"
}
```

Monitoring & Metrics

Prometheus Endpoint

- `/actuator/prometheus`
- Custom metrics: `api_calls_total`, `http_server_requests_seconds`, `jvm_memory_used_bytes`

Grafana Dashboard

- URL: `http://localhost:3000`
- Credentials: `admin/admin123`

Metrics to Watch:

- RPS
- Latency (P95, P99)

- JVM Heap Usage
- Error Rate

Security & Policies

OPA Policies

- No default service account
- Non-root containers
- Security context required
- CPU & Memory limits enforced

Security Scanning

- Trivy
- OWASP Dependency Check
- SonarQube

Troubleshooting

Jenkins Issues

docker logs jenkins-container
java -jar jenkins-cli.jar -s http://localhost:8080/ -auth samarth:<API_TOKEN> who-am-i

Kubernetes Issues

kubectl get pods -n devops-challenge
kubectl logs -f deployment/devops-challenge -n devops-challenge
kubectl describe pod <pod-name> -n devops-challenge

Helm Issues

helm lint ./helm-chart/devops-chart
helm template devops-release ./helm-chart/devops-chart
helm status devops-release

Docker Issues

docker build -t devops-app:latest . --progress=plain
docker build -t devops-app:latest . --no-cache
docker history devops-app:latest

Common Commands

kubectl get all -n devops-challenge
kubectl port-forward service/devops-challenge 8080:80 -n devops-challenge
kubectl scale deployment devops-challenge --replicas=3 -n devops-challenge
helm list --all-namespaces
kubectl get constraints
curl http://localhost:8080/actuator/prometheus | grep api_calls

---

Contributing

1. Fork the repository
2. Create a branch: `git checkout -b feature/amazing-feature`
3. Commit: `git commit -m 'Add some amazing feature'`
4. Push: `git push origin feature/amazing-feature`
5. Open Pull Request

License & Acknowledgments

License: MIT

Acknowledgments:

- Spring Boot
- Kubernetes Community
- Jenkins
- Prometheus
- Grafana
- All open-source contributors