Documentation: CI/CD Pipeline for Containerized Multi-Service App (Pillar Project 5)

Overview

This project implements a complete CI/CD pipeline for a containerized multi-service application using Node.js (backend) and React (frontend). It includes Docker containerization, automated build and deployment to Kubernetes via CI/CD, and secure access through a local domain with HTTPS.

Project Structure

- **Backend**: Node.js + Express API for managing books.
- **Frontend**: React app consuming backend APIs.
- **CI/CD**: GitHub Actions (or similar tool) to automate testing, building, and deploying.
- **Kubernetes (Minikube)**: Local deployment environment.

Backend API

Endpoints

- `GET /api/books`: Retrieve all books.
- `POST /api/books`: Create a new book.

Sample Schema (Mongoose)

const bookSchema = new mongoose.Schema({

```
  title: { type: String, required: true },
  author: { type: String, required: true },
});
```

Frontend

- Built with React.
- Connects to `/api/books` for data.
- Displays and allows submission of books.

Docker Setup

- Each service (frontend & backend) has its own Dockerfile.
- Docker Compose or individual Kubernetes manifests used for orchestration.

CI/CD Pipeline

Steps

1. **Test**: Run tests for backend and frontend.
2. **Build**: Build Docker images.
3. **Push**: Push images to Docker Hub.
4. **Deploy**: Apply Kubernetes manifests using `kubectl`.
5. **Rollback** (optional): Rollback to previous deployment if failure detected.
6. **Notifications** (optional): Send failure alerts via Slack/email.

Tools Used

- GitHub Actions or GitLab CI/CD
- Docker Hub
- kubectl + Kubeconfig

Kubernetes Setup (Minikube)

Namespaces

- `bookapp`: All services run under this namespace.

Services

- `frontend`: React app on port 80.
- `backend`: Node API on port 8000.

Ingress

- Host: `bookapp.local`
- Routes:
    - `/api` -> backend service
    - `/` -> frontend service

Ingress Example

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bookapp-ingress
  namespace: bookapp
```

```
spec:
  rules:
  - host: bookapp.local
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service:
            name: backend
            port:
              number: 8000
      - path: /
        pathType: Prefix
        backend:
          service:
            name: frontend
            port:
              number: 80
```

TLS Support

- Self-signed certificate used locally.
- Requires trust on local machine.

Local Domain Setup

Edit `/etc/hosts`

127.0.0.1 bookapp.local

Access URLs

- `https://bookapp.local` - Frontend
- `https://bookapp.local/api/books` - Backend API

Without `minikube tunnel`

Options:

- Use NodePort services and access via `minikube ip` + NodePort.
- Use `kubectl port-forward`.

Testing Endpoints

curl -k https://bookapp.local/api/books

Final Notes

- Project successfully containerized and deployed.
- CI/CD ensures reliability and speed of delivery.
- Kubernetes ingress makes local domain routing seamless.
- TLS ensures secure communication (even in dev).

Next Steps (Optional Enhancements)

- Add readiness/liveness probes.
- Use Cert-Manager for better TLS.
- Implement horizontal pod autoscaling.
- Configure logging/monitoring.

- Store secrets using Kubernetes Secrets.
