Three-Tier Application — Backend | Database | Proxy

Deployment using Docker & Kubernetes

This project implements a **three-tier web application** consisting of:

> ➓ **You are building a 3-tier blog API system consisting of:**
>
> 1. **Backend API (Go app)** → serves REST responses (list of blog titles).
> 2. **Database (MySQL)** → stores blog titles.
> 3. **Proxy (Nginx)** → exposes the API over HTTPS to clients.
>
> ➓

The setup supports both **Docker Compose** (for local testing) and **Kubernetes (K8s)** (for production-like deployment).

```
.
├── docker-compose.yaml
├── backend/
│   ├── Dockerfile
│   ├── main.go
│   ├── go.mod / go.sum
│
├── nginx/
│   ├── Dockerfile
│   ├── nginx.conf
│   ├── generate-ssl.sh
│
├── K8S/
│   ├── backend_deployment.yaml
│   ├── backend_service.yaml
│   ├── database_deployment.yaml
│   ├── db-service.yaml
│   ├── db-secret.yaml
│   ├── db-data-pv.yaml
│   ├── db-data-pvc.yaml
│   ├── proxy_deployment.yaml
│   ├── proxy_nodeport.yaml
```

Backend

- Implemented in **Go**.
- Built using a **multi-stage Dockerfile**:
    - Stage 1: Build Go binary.
    - Stage 2: Run optimized binary on minimal image (e.g. alpine).

Database

Credentials stored in:

- **db-password.txt** (for local)
- **Kubernetes Secret** (`db-secret.yaml`) (for production).

- Persistent storage handled via:
    - **db-data-pv.yaml**
    - **db-data-pvc.yaml**

Proxy (Nginx)

❿ Acts as **reverse proxy** to the backend.

❿ Uses **HTTPS** (self-signed certificate generated by `generate-ssl.sh`).

Local Deployment (Docker Compose)

Kubernetes Deployment (Full Stack)

Expected Deliverables

❿ Fully functional 3-tier application accessible via HTTPS

❿ Docker images for all components

❿ K8s manifests for automated deployment:

  ❿ `backend_deployment.yaml`, `backend_service.yaml`

  ❿ `database_deployment.yaml`, `db-service.yaml`, `db-secret.yaml`

  ❿ `proxy_deployment.yaml`, `proxy_nodeport.yaml`

  ❿ `db-data-pv.yaml`, `db-data-pvc.yaml`

❿