FastAPI on GKE

# Project Documentation

## 📌 Requirement Understanding

The objective was to deploy a FastAPI-based microservice with a PostgreSQL backend on Google Kubernetes Engine (GKE). It must expose APIs through a browser-accessible URL with routing handled via an Ingress controller. The system must be resilient, automatically recover from pod failures, and follow best practices for cloud-native deployments.

## 📎 Assumptions

- The application is containerized using Docker.

- The target Kubernetes cluster is hosted on GKE with Autopilot enabled.

- PostgreSQL needs persistent storage using Kubernetes PVCs.

- Users can install third-party controllers such as NGINX Ingress.

- The application must expose documentation via Swagger UI.

- The system should auto-heal (i.e., regenerate pods on failure).

## 🧩 Solution Overview

### Technology Stack

- **FastcAPI**: Python based Web application framework.

- **PostgreSQL**: Database engine.

- **GKE (Google Kubernetes Engine)**: Managed Kubernetes hosting.

- **NGINX Ingress Controller**: HTTP routing.

- **PersistentVolumeClaim (PVC)**: Storage backend for PostgreSQL.

### Kubernetes Objects Used

### Deployment Steps Summary

1. Build and push Docker image.

2. Create a GKE Autopilot cluster.

3. Apply PVC and PostgreSQL deployment.

4. Deploy FastAPI app and its service.

5. Install NGINX Ingress Controller.

6. Create Ingress rule and get external IP.

7. Access app via Swagger at `http://34.58.61.200/docs`. (This link is not working as of now. Because GCP cluster is stopped.)

### Auto-Healing Demo

- Killing a FastAPI pod: Kubernetes re-creates it automatically.

- Killing the PostgreSQL pod: The same behavior is observed.

- PVCs retain database data even after pod replacement.

## ⚙️ Justification for the Resources Utilized

- **GKE Autopilot**: Eliminates node-level management and provides auto-scaling and security benefits.

- **PostgreSQL with PVC**: Ensures durable, stateful data storage.

- **FastAPI + Docker**: Lightweight and fast backend suitable for APIs.

- **NGINX Ingress**: A standard for managing traffic, works well with GKE and supports regex routing.

- **Kubernetes Deployments**: Ensures pod resilience and replicability.

## ✅ Observations and Validations

- Application accessible through Ingress IP.

- Swagger UI loaded properly at `/docs`.

- All API endpoints accessible and functional inside and outside the cluster.

- Kubernetes regenerated pods as expected upon deletion.

## 🔚 Conclusion

This solution meets the functional and non-functional requirements for a scalable, resilient, and cloud-native API application hosted on GKE with a persistent backend.

## Referenced Video

[k8s\_assignment\_720.mp4](https://nagarro-my.sharepoint.com/:v:/p/prakhar_agarwal02/EXxdy0-3OP9AskyS0ezTw8kBLh6wGK6BZmdfQzrZ3zRYrg?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=jXiXvL)