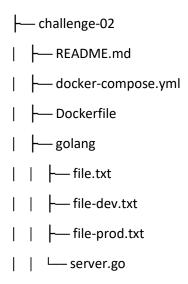
# Containerizing the Go Web Server and Adding Environment-Based Deployment

The task involves containerizing the Go web server and creating a deployment setup that switches between DEV and PROD environments by reading different files. I'll guide you through the steps:

#### **Folder Structure:**



# **Step 1: Dockerizing the Go Web Server**

1. Create a Dockerfile in the challenge-02 folder:

## **Step 2: Create a Docker Compose Setup**

2. Create a docker-compose.yml file to define services for different environments

# **Step 3: Adjust Deployment Logic with Environment Variables**

Since you can't change the server.go file directly, we will ensure that the correct file (file-dev.txt or file-prod.txt) is linked as file.txt depending on the environment using docker-compose volume mounting.

- In the docker-compose.yml file, for DEV, the file-dev.txt is mounted to /app/file.txt.
- For PROD, file-prod.txt is mounted to /app/file.txt.

This way, when the server is started, it will read from file.txt, but the actual content comes from either file-dev.txt or file-prod.txt based on the environment.

# **Step 4: Build and Run the Containers**

1. Build and run for DEV:

docker-compose up --build web-server-dev

- This will start the server on port 8080 with the DEV environment.
- Visit http://localhost:8080 to see the content of file-dev.txt

```
Hello there! This is DEV Golang web server!
```

Build and run for PROD:

docker-compose up --build web-server-prod

- This will start the server on port 8081 with the PROD environment.
- Visit http://localhost:8081 to see the content of file-prod.txt

Hello there! This is PROD Golang web server!

# **Step 5: Observability (Stretch Goal)**

For observability, you can integrate a simple logging mechanism within Docker to capture logs and later add monitoring tools like Prometheus or Grafana. As an easy observability step, Docker Compose logs can be monitored:

### 1. View Logs:

docker-compose logs -f

This command will continuously stream the server logs, which can help in debugging or monitoring server health.

#### 1. To build and run the DEV server:

```
docker-compose up --build web-server-dev
```

## 2. To build and run the PROD server:

```
docker-compose up --build web-server-prod
```

The Go web server will be containerized, and the appropriate environment-specific file (file-dev.txt or file-prod.txt) will be displayed based on the deployment environment.