Cloud-Native Full-Stack Book Store: A Dockerized Spring Boot & Angular Implementation

🏗️ Phase 1: Project Creation (The Foundation)

The goal was to build a decoupled 3-tier architecture for high performance and scalability.

1. **Backend (Spring Boot):**
   - Generated via Spring Initializr with dependencies: **Spring Web, Spring Data JPA, MySQL Driver, and Validation (JSR-380)**.
   - **Architecture:** Implemented the Controller-Service-Repository pattern.
   - **Security:** Configured a WebConfig class to handle **CORS**, allowing our Angular frontend to communicate with the API.
2. **Frontend (Angular):**
   - Created using ng new.
   - **Services:** Developed an HttpClient service to consume REST endpoints.
   - **Validation:** Used Reactive Forms for real-time user feedback.
3. **Database (Azure MySQL):**
   - Provisioned a **MySQL Flexible Server** on Azure.
   - Configured **Firewall Rules** to allow local connections and whitelisted the necessary IP addresses.

🐬 Phase 2: Dockerization (Containerization)

We used Docker to ensure the app runs the same on any machine ("Build Once, Run Anywhere").

1. **Dockerfile (Backend):** Used a multi-stage build.
   - **Stage 1:** Used Maven to compile the code into a .jar file.
   - **Stage 2:** Used an OpenJDK JRE image to run the JAR, keeping the image size small.
2. **Dockerfile (Frontend):**
   - Compiled the Angular app using Node.js.
   - Served the static files using an **Nginx** web server.
3. **Docker Compose:**
   - Created a docker-compose.yml to orchestrate the containers.
   - **Networking:** Created a custom bridge network (bookstore-network) so containers can talk to each other.
   - **Environment Injection:** Injected Azure credentials into the Spring Boot container via environment variables.

🚀 Phase 3: Deployment & Cloud Integration

This is the "Pro" part of your project where we moved from local to cloud.

1. **Database Migration:**

- Redirected the Spring Boot datasource.url from localhost to the **Azure MySQL endpoint**.
- Applied hibernate.ddl-auto=update to automatically generate the schema in the cloud.

2. **GitHub Workflow:**

- Cleaned the Git history to ensure no sensitive Azure passwords were leaked.
- Updated the .gitignore to exclude build artifacts like /target.
- Pushed the final source code to GitHub for version control.

3. **Execution:**

- Ran docker-compose up -d --build.
- **The Result:** A live Angular frontend at port 80 communicating with a Spring Boot container at port 8080, which securely persists data to **Azure Cloud**.