**Project Documentation: Containerizing React-Express-MongoDB Application**

**Overview**

This project involves containerizing a full-stack web application using Docker. The application consists of:

1. A React frontend
2. An Express backend
3. A MongoDB database

The aim is to package the application into Docker containers, making it portable and easy to deploy across environments.

**Project Components**

1. **Frontend (React)**

* The React application serves as the user interface.
* It is built using modern JavaScript and interacts with the backend via API calls.

1. **Backend (Express)**

* The Express application provides RESTful APIs.
* It handles business logic and connects to the MongoDB database.

1. **Database (MongoDB)**

* MongoDB serves as the database for persisting application data.

**Tools and Technologies**

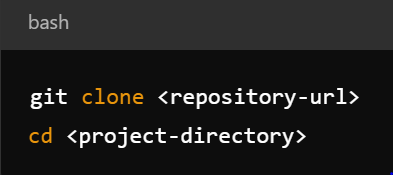
1. **Docker:** Used to containerize the application.
2. **Docker Compose:** Manages multi-container applications.
3. **Shell Script**: Automates the build and deployment process.

**Project Structure**

* **Frontend Directory:** Contains React application code and its Dockerfile.
* **Backend Directory:** Contains Express application code and its Dockerfile.
* **docker-compose.yml:** Manages frontend, backend, and MongoDB services.
* **deploy.sh:** Shell script to automate container build and deployment.

**Steps to Run the Project**

**1. Clone the Repository**



**2. Dockerize the Application**

Frontend Dockerfile:

* Base Image: node:16
* Build React app and serve using nginx.

Backend Dockerfile:

* Base Image: node:16
* Runs the Express application.

**3. Create Docker Compose File**

The docker-compose.yml defines:

* frontend service: Maps port 3000 to host.
* backend service: Maps port 5000 to host.
* mongo service: Configures the MongoDB database.

**4. Run the Shell Script**

The deploy.sh script automates:

* Building Docker images.
* Tagging and pushing images to Docker Hub.
* Deploying the application using Docker Compose.

**API Endpoints**

**GET /api/users**

* Fetches all users from the database.

**POST /api/users**

* Creates a new user.
* Request Body:



**Testing**

1. Ensure Docker is installed and running.
2. Run the shell script:



1. Access the application in your browser:

* Frontend: http://localhost:3000
* Backend APIs: http://localhost:5000/api/users

**Challenges Faced**

1. Cannot GET / Error:

* Solution: Ensure the correct nginx configuration for serving React static files.

1. MongoDB Connection Issues:

* Solution: Set up proper environment variables in the docker-compose.yml file.

**Future Enhancements**

1. Implement CI/CD pipelines for automated deployments.
2. Add scalability by configuring Kubernetes.
3. Use environment-specific configurations for production-ready deployment.