**Dockerizing a MERN Stack Project**

This documentation will guide you through the process of setting up a MERN stack project using Docker and Docker Compose. By the end of this guide, you will have a fully containerized MERN application with separate containers for React, Express, and MongoDB.

**Prerequisites**

* Docker installed on your machine
* Docker Compose installed on your machine
* Basic understanding of Docker and Docker Compose

**Project Structure**

Your project directory should have the following structure:

mern-docker-project/

├── react-docker/

│ ├── Dockerfile

│ ├── package.json

│ ├── public/

│ └── src/

├── express-docker/

│ ├── Dockerfile

│ ├── package.json

│ └── src/

├── docker-compose.yml

└── README.md

### 1. Setting Up the React Application

**Dockerfile for React (react-docker/Dockerfile):**

# Use an official Node.js runtime as a parent image

FROM node:20

# Set the working directory to /app

WORKDIR /app

# Copy the package.json and package-lock.json files

COPY ./package\*.json ./

# Install dependencies

RUN npm install --force

# Copy the rest of the application code

COPY . .

# Expose port 3000

EXPOSE 3000

# Run the application

CMD ["npm", "run", "start"]

**package.json for React (react-docker/package.json):** Ensure you have a standard package.json for your React application.

### 2. Setting Up the Express Application

**Dockerfile for Express (express-docker/Dockerfile):**

# Use an official Node.js runtime as a parent image

FROM node:20

# Set the working directory to /app

WORKDIR /app

# Copy the package.json and package-lock.json files

COPY ./package\*.json ./

# Install dependencies

RUN npm install

# Copy the rest of the application code

COPY . .

# Expose port 5500

EXPOSE 5500

# Run the application

CMD ["node", "src/index.js"]

**package.json for Express (express-docker/package.json):** Ensure you have a standard package.json for your Express application.

### 3. Setting Up Docker Compose

**docker-compose.yml (docker-compose.yml):**

version: '3'

services:

front:

build:

context: './react-docker'

dockerfile: Dockerfile

ports:

- 3000:3000

container\_name: react-container

environment:

- WATCHPACK\_POLLING=true

networks:

- mern-stack-network

volumes:

- ./react-docker:/app

depends\_on:

- backend

backend:

build:

context: './express-docker'

dockerfile: Dockerfile

ports:

- 5500:5500

container\_name: express-container

networks:

- mern-stack-network

volumes:

- ./express-docker:/app

depends\_on:

- mongo

mongo:

image: mongo

ports:

- 27017:27017

container\_name: mongo-container

networks:

- mern-stack-network

volumes:

- mongoData:/data/db

environment:

- MONGO\_INITDB\_ROOT\_USERNAME=root

- MONGO\_INITDB\_ROOT\_PASSWORD=example

- MONGO\_INITDB\_DATABASE=test

networks:

mern-stack-network:

volumes:

mongoData:

### 4. Building and Running the Containers

Navigate to your project directory (mern-docker-project) and run the following commands:

1. Build the containers:

docker-compose build

1. Start the containers:

docker-compose up

1. Stop the containers:

docker-compose down

### 5. Accessing the Application

* **React Application:** Open your browser and navigate to http://localhost:3000.
* **Express API:** You can access the API at http://localhost:5500.