**1. Structure the Application**

Your full-stack application consists of **two main parts**:

* **Frontend**: Uses Node.js 22, React 19, Next.js 15.1.6, Gatsby 15.1.6, Vite 5, and other libraries.
* **Backend**: Uses Java 21, Spring Boot 3.4.1, Maven 3.9.5, and MySQL Connector.

**Project Directory Structure**

bash

CopyEdit

/fullstack-app

├── backend/ # Spring Boot backend

│ ├── src/

│ ├── pom.xml

│ ├── ...

│ └── Dockerfile

├── frontend/ # React / Next.js frontend

│ ├── src/

│ ├── package.json

│ ├── yarn.lock

│ ├── Dockerfile

│ └── .env

├── docker-compose.yml

├── .dockerignore

**2. Create Dockerfile for Frontend**

This **Dockerfile** builds the frontend with **NodeJS 22**, React, Next.js, Vite, and Gatsby.

**frontend/Dockerfile**

dockerfile

# Use Node.js 22 base image

FROM node:22 AS frontend

# Set working directory

WORKDIR /app

# Copy package files

COPY package.json yarn.lock ./

# Install dependencies

RUN yarn install

# Copy the rest of the application

COPY . .

# Build the frontend (for production)

RUN yarn build

# Expose the frontend port (e.g., Next.js runs on 3000)

EXPOSE 3000

# Start the frontend application

CMD ["yarn", "start"]

**3. Create Dockerfile for Backend**

This **Dockerfile** builds the backend with **Java 21, Maven 3.9.5, and Spring Boot**.

**backend/Dockerfile**

dockerfile

CopyEdit

# Use Java 21 base image

FROM openjdk:21 AS backend

# Set working directory

WORKDIR /app

# Copy Maven configuration

COPY pom.xml ./

# Download dependencies

RUN mvn dependency:go-offline

# Copy application source code

COPY src/ ./src/

# Build the Spring Boot application

RUN mvn clean package -DskipTests

# Expose backend port

EXPOSE 8080

# Run the application

CMD ["java", "-jar", "target/\*.jar"]

**4. Create docker-compose.yml to Manage Services**

To run both **frontend, backend, and database**, use **docker-compose**.

**docker-compose.yml**

yaml

CopyEdit

version: "3.9"

services:

frontend:

build: ./frontend

ports:

- "3000:3000"

depends\_on:

- backend

backend:

build: ./backend

ports:

- "8080:8080"

environment:

- SPRING\_DATASOURCE\_URL=jdbc:mysql://db:3306/backenddb

- SPRING\_DATASOURCE\_USERNAME=backenduser

- SPRING\_DATASOURCE\_PASSWORD=backendpass

depends\_on:

- db

db:

image: mysql:8.0

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABASE=backenddb

- MYSQL\_USER=backenduser

- MYSQL\_PASSWORD=backendpass

ports:

- "3306:3306"

volumes:

- db-data:/var/lib/mysql

volumes:

db-data:

**5. How to Build & Run**

**Build the Containers**

docker-compose build

**Run the Containers**

bash

CopyEdit

docker-compose up -d

**Check Running Containers**

bash

CopyEdit

docker ps

**6. Access the Application**

* **Frontend (React/Next.js/Gatsby UI):**  
  👉 http://localhost:3000
* **Backend (Spring Boot API):**  
  👉 http://localhost:8080
* **Database (MySQL):**  
  Connect using localhost:3306 with credentials backenduser/backendpass

**This setup ensures all tools (Node.js, Java, Spring Boot, MySQL) work together in Docker! 🚀**

**Example 2**

**🖥️ Frontend Dockerfile (NodeJS 22 + React + Next.js + Gatsby)**

Installs **Node.js 22** and all required libraries (**React, Next.js, Gatsby, Material UI, axios, etc.**).

**📌 frontend/Dockerfile**

dockerfile

CopyEdit

# Use Node.js 22 base image

FROM node:22 AS frontend

# Set working directory

WORKDIR /app

# Copy package files

COPY package.json yarn.lock ./

# Install required dependencies

RUN yarn install

# Copy the full project

COPY . .

# Install tools (React, Next.js, Gatsby, Material UI, etc.)

RUN yarn add react@19 jwt-decode@1.0.2 react-router-dom@7.1.3 \

react-hot-toast@2.5.1 @mui/material@6.4.2 moment@2.30.1 \

axios@1.7.9 react-icons@1 react-hook-form@7.54.2 \

next@15.1.6 vite@5 gatsby@15.1.6

# Build frontend for production

RUN yarn build

# Expose frontend port (e.g., Next.js default port)

EXPOSE 3000

# Start frontend

CMD ["yarn", "start"]

**🖥️ Backend Dockerfile (Java 21 + Maven + Spring Boot)**

Installs **Java 21, Maven 3.9.5, and required Spring Boot libraries**.

**📌 backend/Dockerfile**

dockerfile

CopyEdit

# Use Java 21 base image

FROM openjdk:21 AS backend

# Set working directory

WORKDIR /app

# Install Maven 3.9.5

RUN apt-get update && \

apt-get install -y maven && \

mvn -version

# Copy Maven files

COPY pom.xml ./

# Install Spring Boot & dependencies

RUN mvn dependency:go-offline

# Copy backend source code

COPY src/ ./src/

# Install MySQL Connector & other Spring Boot dependencies

RUN mvn install:install-file -DgroupId=mysql \

-DartifactId=mysql-connector-java -Dversion=8.0.33 \

-Dpackaging=jar -Dfile=libs/mysql-connector-java-8.0.33.jar

# Build Spring Boot JAR

RUN mvn clean package -DskipTests

# Expose backend port

EXPOSE 8080

# Start the backend

CMD ["java", "-jar", "target/\*.jar"]

**📦 Docker Compose (Full Stack)**

Manages **frontend, backend, and MySQL database**.

**📌 docker-compose.yml**

yaml

CopyEdit

version: "3.9"

services:

frontend:

build: ./frontend

ports:

- "3000:3000"

depends\_on:

- backend

backend:

build: ./backend

ports:

- "8080:8080"

environment:

- SPRING\_DATASOURCE\_URL=jdbc:mysql://db:3306/backenddb

- SPRING\_DATASOURCE\_USERNAME=backenduser

- SPRING\_DATASOURCE\_PASSWORD=backendpass

depends\_on:

- db

db:

image: mysql:8.0

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABASE=backenddb

- MYSQL\_USER=backenduser

- MYSQL\_PASSWORD=backendpass

ports:

- "3306:3306"

volumes:

- db-data:/var/lib/mysql

volumes:

db-data:

**🚀 How to Build & Run**

**1️⃣ Build Docker Containers**

bash

CopyEdit

docker-compose build

**2️⃣ Run Full-Stack Application**

bash

CopyEdit

docker-compose up -d

**3️⃣ Verify Running Containers**

bash

CopyEdit

docker ps

**4️⃣ Access the Application**

* **Frontend (React/Next.js/Gatsby)**:  
  👉 http://localhost:3000
* **Backend (Spring Boot API)**:  
  👉 http://localhost:8080
* **Database (MySQL)**:  
  Connect using localhost:3306 with **backenduser/backendpass**.

**✅ Summary**

* **Frontend:** Installs **Node.js 22** and required **React, Next.js, Gatsby, Material UI, axios** libraries.
* **Backend:** Installs **Java 21, Maven 3.9.5, MySQL Connector**, and **Spring Boot dependencies**.
* **Docker Compose:** Manages frontend, backend, and database in **one command**.
* **Fully Automated Deployment** with Docker.

🎯 **Now your full-stack application is containerized and ready to deploy!** 🚀

**Example 3**

**📂 Project Directory Structure**

/fullstack-app

├── backend/ # Spring Boot backend

│ ├── src/

│ ├── pom.xml

│ ├── Dockerfile

│ └── .env

├── frontend/ # React / Next.js / Gatsby frontend

│ ├── src/

│ ├── package.json

│ ├── yarn.lock

│ ├── Dockerfile

│ └── .env

├── docker-compose.yml # Docker Compose for full-stack setup

├── .dockerignore

├── README.md

**🖥️ Frontend Dockerfile (RHEL 8 + NodeJS 22 + React + Next.js + Gatsby)**

Installs **Node.js 22** and all required frontend libraries (**React, Next.js, Gatsby, Material UI, axios, etc.**).

**📌 frontend/Dockerfile**

dockerfile

# Use Red Hat UBI 8 as the base image

FROM registry.redhat.io/ubi8/nodejs-22:latest

# Set working directory

WORKDIR /app

# Copy package files

COPY package.json yarn.lock ./

# Install required dependencies

RUN yarn install

# Copy the full project

COPY . .

# Install tools (React, Next.js, Gatsby, Material UI, etc.)

RUN yarn add react@19 jwt-decode@1.0.2 react-router-dom@7.1.3 \

react-hot-toast@2.5.1 @mui/material@6.4.2 moment@2.30.1 \

axios@1.7.9 react-icons@1 react-hook-form@7.54.2 \

next@15.1.6 vite@5 gatsby@15.1.6

# Build frontend for production

RUN yarn build

# Expose frontend port (e.g., Next.js default port)

EXPOSE 3000

# Start frontend

CMD ["yarn", "start"]

**🖥️ Backend Dockerfile (RHEL 8 + Java 21 + Maven + Spring Boot)**

Installs **Java 21, Maven 3.9.5, and required Spring Boot libraries** using **RHEL 8 package manager (dnf/yum)**.

**📌 backend/Dockerfile**

dockerfile

# Use Red Hat UBI 8 minimal image

FROM registry.redhat.io/ubi8/ubi:latest

# Set working directory

WORKDIR /app

# Install Java 21, Maven, and required dependencies

RUN dnf install -y java-21-openjdk-devel maven && \

dnf clean all

# Set Java environment variables

ENV JAVA\_HOME=/usr/lib/jvm/java-21-openjdk

ENV PATH=$JAVA\_HOME/bin:$PATH

# Copy Maven configuration

COPY pom.xml ./

# Install Spring Boot & dependencies

RUN mvn dependency:go-offline

# Copy backend source code

COPY src/ ./src/

# Install MySQL Connector & other Spring Boot dependencies

RUN mvn install:install-file -DgroupId=mysql \

-DartifactId=mysql-connector-java -Dversion=8.0.33 \

-Dpackaging=jar -Dfile=libs/mysql-connector-java-8.0.33.jar

# Build Spring Boot JAR

RUN mvn clean package -DskipTests

# Expose backend port

EXPOSE 8080

# Start the backend

CMD ["java", "-jar", "target/\*.jar"]

**📦 Docker Compose (Full Stack on RHEL 8)**

Manages **frontend, backend, and MySQL database**.

**📌 docker-compose.yml**

version: "3.9"

services:

frontend:

build: ./frontend

ports:

- "3000:3000"

depends\_on:

- backend

backend:

build: ./backend

ports:

- "8080:8080"

environment:

- SPRING\_DATASOURCE\_URL=jdbc:mysql://db:3306/backenddb

- SPRING\_DATASOURCE\_USERNAME=backenduser

- SPRING\_DATASOURCE\_PASSWORD=backendpass

depends\_on:

- db

db:

image: mysql:8.0

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABASE=backenddb

- MYSQL\_USER=backenduser

- MYSQL\_PASSWORD=backendpass

ports:

- "3306:3306"

volumes:

- db-data:/var/lib/mysql

volumes:

db-data:

**🚀 How to Build & Run**

**1️. Build Docker Containers**

docker-compose build

**2️. Run Full-Stack Application**

bash

CopyEdit

docker-compose up -d

**3️. Verify Running Containers**

bash

CopyEdit

docker ps

**4️. Access the Application**

* **Frontend (React/Next.js/Gatsby)**:  
  👉 http://localhost:3000
* **Backend (Spring Boot API)**:  
  👉 http://localhost:8080
* **Database (MySQL)**:  
  Connect using localhost:3306 with **backenduser/backendpass**.

**Summary**

* **Uses RHEL 8 UBI images** for both **frontend (Node.js 22)** and **backend (Java 21, Maven, Spring Boot)**.
* **Installs all required tools** (React, Next.js, Gatsby, Spring Boot, MySQL, etc.).
* **Docker Compose automates full-stack deployment**.
* **Production-ready and optimized** for **RHEL-based systems**.

**Now your full-stack application runs inside Docker using RHEL 8!**