Node js Application Running Instructions

1. Prerequisites

Before setting up and running the project, ensure that the following tools are installed:

- **Docker**: To build and run the Docker container.
    - Install Docker
- **Docker Compose**: To manage multi-container applications (if you're using Docker Compose).
    - Install Docker Compose
- **Node.js** (optional for local development, if not running via Docker):
    - Install Node.js

2. Project Structure

The project directory contains the following files:

- **Dockerfile**: Instructions for building the Docker image.
- **docker-compose.yml**: Docker Compose configuration file to easily manage the services.
- **index.js**: Main application file (Node.js).
- **package.json**: Node.js project metadata and dependencies.
- **package-lock.json**: Automatically generated file that locks the dependency versions.
- **README.md**: Documentation file explaining the project.

3. Setting Up the Project

3.1. Building and Running with Docker Compose

You can use **Docker Compose** to simplify the process of building and running your Docker container. It manages your container configuration in a more streamlined manner.

```
Step 1: Create docker-compose.yml
Make sure your project has a docker-compose.yml file in the root directory with the following
content:
version: '3.9' # Specify the Docker Compose version
services:
  node-app:
    build:
      context: . # Use the current directory for the build context
                dockerfile: Dockerfile # Specify the Dockerfile to use
    ports:
      - "3000:3000" # Map host port 3000 to container port3000
```

Step-1 :- Build and Start the Containers:-

1. **Navigate to the project directory**: Open a terminal and change to the directory containing your project files.

   `cd /path/to/project`
2. **Build and run the application with Docker Compose**: Run the following command to build and start the container as defined in the `docker-compose.yml` file:

   `docker-compose up --build`
   - `--build`: Forces Docker Compose to rebuild the images before starting the containers.
   - The `docker-compose up` command will build the Docker image using the instructions in the `Dockerfile`, and then start the container. It will also map port `3000` from the container to port `3000` on the host machine.

   The application should now be accessible at `http://localhost:3000`.

Step 2:- Running in Detached Mode (Optional)

If you want to run the containers in the background (detached mode), use the following command:

```
docker-compose up -d
```

To stop the container, you can use:

```
docker-compose down
```

3.2. Running the Docker Container Without Docker Compose

If you prefer to run the application without using Docker Compose, follow these steps:

Build the Docker image: Run the following command to build the Docker image based on the instructions in the `Dockerfile`.

```
docker build -t my-node-app .
```

Run the Docker container: Run the container with the following command:

```
docker run -p 3000:3000 my-node-app
```

The application will now be accessible at `http://localhost:3000`.

3.3. Running Locally Without Docker (Optional)

If you want to run the application locally (without Docker), follow these steps:

1. **Install the dependencies:**

In the project directory, run the following command to install the dependencies specified in `package.json`:

```
npm install
```

2. **Start the application**: Run the application with the following command:

```
npm start
```

3. The application will now run, typically available at `http://localhost:3000`.

4. Dockerfile Explanation

Here is a brief explanation of the `Dockerfile` you used:

Dockerfile:-

```
# Step 1: Use official Node.js image (version 20) as base
FROM node:20
# Step 2: Set working directory inside the container
WORKDIR /app
```

```
# Step 3: Copy package files (package.json and package-lock.json) to the container
COPY package*.json ./
# Step 4: Install dependencies inside the container
RUN npm install
# Step 5: Copy the rest of the application code into the container
COPY . .
# Step 6: Expose port 3000 for accessing the app from outside the container
EXPOSE 3000
# Step 7: Define the command to run the application
CMD ["node", "index.js"]
```

Explanation of Dockerfile Steps:

1. **FROM node:20**: Starts from the official Node.js version 20 image. This base image comes with Node.js pre-installed.
2. **WORKDIR /app**: Sets `/app` as the working directory inside the container. This means that subsequent commands will run in this directory.
3. ***COPY package*.json ./***: Copies both `package.json` and `package-lock.json` into the `/app` directory inside the container. This step is important for dependency management.
4. **RUN npm install**: Installs the dependencies specified in the `package.json` file inside the container.
5. **COPY . .**: Copies the rest of the application files (such as `index.js`, `README.md`, etc.) into the container.
6. **EXPOSE 3000**: Exposes port 3000 to allow the app to be accessible from outside the container on the specified port.
7. **CMD ["node", "index.js"]**: This command is executed when the container starts. It runs the application using `node index.js`.

5. Docker Compose File Explanation:-

Here is an explanation of the `docker-compose.yml` file:

```yaml
version: '3.9'  # Docker Compose version

services:
  node-app:
    build:
      context: .  # Use the current directory for the build context
      dockerfile: Dockerfile  # Specify the Dockerfile to use
    ports:
      - "3000:3000"  # Map host port 3000 to container port 3000
```

Explanation of Docker Compose File:

- **version: '3.9'**: Specifies the version of the Docker Compose file format being used.
- **services**: Defines the services (containers) in the application.
- **node-app**: The name of the service. This service will build and run the Docker container for the Node.js app.
- **build**:
    - **context: .**: Specifies the build context as the current directory.
    - **dockerfile: Dockerfile**: Tells Docker Compose to use the `Dockerfile` in the current directory to build the image.
- **ports**:
    - `3000:3000`: Maps port 3000 on the host machine to port 3000 inside the container. This allows access to the app via `http://localhost:3000`.