📘 Dockerized Node.js App Documentation

🏗️ Overview

This project is a simple Node.js application that uses Express to serve a "Hello, Dockerized Node.js App!" message. The application is containerized using Docker, making it easy to deploy and scale.

🔧 Prerequisites

- **Node.js**
- **Docker**

📂 Project Structure

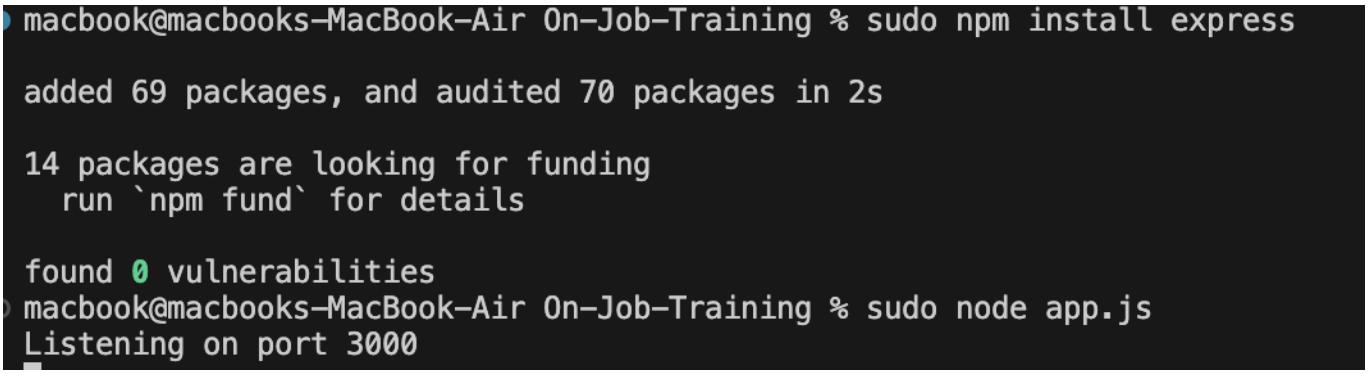
```
my-node-app/
├── app.js               # Main application file
├── Dockerfile           # Docker configuration file
├── package.json         # Node.js dependencies and scripts
└── README.md            # Project documentation (this file)
```

🚀 Setup Instructions

1️⃣ Install Dependencies

Install the required Node.js dependencies:

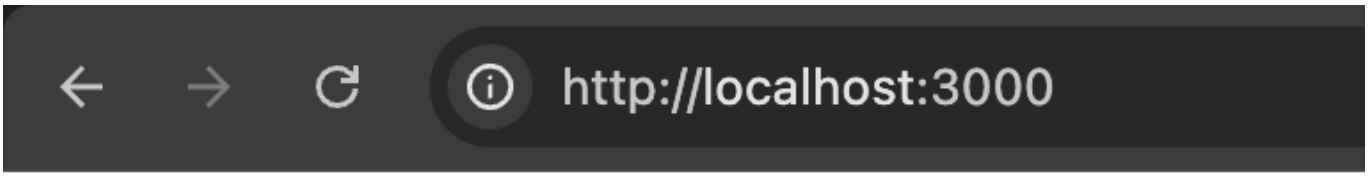
```
npm install
```



2️⃣ Run the App Locally

To run the app without Docker:

Visit **http://localhost:3000**



Hello World

🐋 Docker Setup

1️⃣ Build the Docker Image

Build the Docker image for the Node.js app:

```
FROM node:18
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node","app.js"]
```

```
macbook@macbooks-MacBook-Air On-Job-Training % docker login
Authenticating with existing credentials...
Login Succeeded
macbook@macbooks-MacBook-Air On-Job-Training % docker build -t lab1-docker .
[+] Building 1788.1s (12/12) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 188B
 => [internal] load metadata for docker.io/library/node:18
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 103B
 => [1/5] FROM docker.io/library/node:18@sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf
 => => resolve docker.io/library/node:18@sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf
 => => sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf 6.41kB / 6.41kB
 => => sha256:92bc55c03ade8f2fe25f75fdff0df86d88efacad9365978376ff4caad85a9010 2.50kB / 2.50kB
 => => sha256:fc10358e89334399648688e760ec7f005e3c8944e96231e6223ba9bf6bb50860 6.40kB / 6.40kB
```

2️⃣ Create a New Bridge Network

To ensure network isolation and facilitate communication between containers, create a new Docker bridge network:

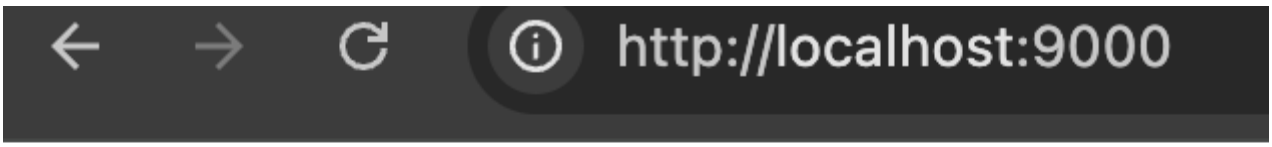
```
macbook@macbooks-MacBook-Air On-Job-Training % docker network create docker-lab
476d9eae4ecd93262b6899da0072d0886b63290be9de940f920c7fee9ae8cc49
macbook@macbooks-MacBook-Air On-Job-Training % docker network ls
NETWORK ID     NAME         DRIVER    SCOPE
7e1dc4dc7857   bridge       bridge    local
476d9eae4ecd   docker-lab   bridge    local
```

3️⃣ Run the Docker Container Inside the Network

Run the container within the newly created bridge network:

```
macbook@macbooks-MacBook-Air On-Job-Training % docker run -d --name node-app --network docker-lab  -p 9000:3000
lab1-docker
be9d14689f06208e9363c0c391c061cb73793c9fcc2e06a6c48666aeec5fc14f
```

Visit **http://localhost:9000** to access the app.



Hello World

4️⃣ Stop the Container

To stop the running container, first find its container ID using:

```
macbook@macbooks-MacBook-Air On-Job-Training % docker  ps
CONTAINER ID   IMAGE         COMMAND                  CREATED              STATUS              PORTS
d62e5ddca2eb   lab1-docker   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:9000->3000/t
```

Then, stop the container with remove:

```
macbook@macbooks-MacBook-Air On-Job-Training % docker  stop be9d14689f06
be9d14689f06
macbook@macbooks-MacBook-Air On-Job-Training % docker  remove be9d14689f06
be9d14689f06
```