

Day 1 — Dockerizing Backend

Overview

On Day 1, we focused on making an existing Node.js Express backend production-ready using Docker. This document summarizes the entire workflow, folder structure, files, commands, and environment setup.

1. Backend Setup

- **Backend Framework:** Node.js + Express
- **Dependencies:** Listed in `package.json`
- **Environment Variables:** Stored in `.env`

Folder Structure

```
Day_1/  
├── .env           # environment variables (API keys, secrets, DB URLs)  
├── app.js         # main Express app  
├── package.json   # dependencies & scripts  
├── package-lock.json # exact dependency versions  
├── node_modules/  # installed packages  
└── Dockerfile     # Docker build instructions
```

`.env` Example

```
PORT=5000  
MONGO_URI=mongodb+srv://username:password@cluster.mongodb.net/mydb  
API_KEY=your_api_key_here
```

2. Dockerizing the Backend

Step 1 — Create `.dockerignore`

```
node_modules  
.env  
npm-debug.log
```

This ensures sensitive files and local dependencies are not copied into the Docker image.

Step 2 — Create `Dockerfile`

```
# Use official Node.js image
FROM node:20

# Set working directory inside container
WORKDIR /app

# Copy dependency files and install
COPY package*.json ./
RUN npm install

# Copy all source files
COPY . .

# Expose app port
EXPOSE 5000

# Run the app
CMD ["node", "app.js"]
```

Step 3 — Build Docker Image

```
docker build -t my-backend .
```

Builds the backend Docker image locally.

Step 4 — Run Container with `.env`

```
docker run -p 5000:5000 --env-file .env my-backend
```

Runs the backend inside a Docker container on port 5000 using environment variables.

Step 5 — Verify Container

Open browser or Postman:

```
http://localhost:5000
```

Your backend should respond exactly as it does locally.

3. Commands Reference

Command	Purpose
<code>docker build -t my-backend .</code>	Build Docker image for backend
<code>docker images</code>	List all Docker images
<code>docker run -p 5000:5000 --env-file .env my-backend</code>	Run backend container with env variables
<code>docker ps</code>	Check running containers
<code>docker stop <container_id></code>	Stop a running container
<code>docker rm <container_id></code>	Remove a stopped container

4. What Docker Does

- Packages your **Node.js app + environment + dependencies** into a container
- Ensures **predictable behavior** across machines
- Keeps **local code untouched and secure**
- Allows easy **sharing** and future deployment to cloud/servers

5. Post-Day 1 Concepts

- **Container Registry:** Push Docker image to Docker Hub / AWS ECR / GitHub Container Registry
- **Cloud / Server Deployment:** Cloud pulls image, runs container, serves users
- **Multi-Container Setup:** Later, backend + frontend + database via Docker Compose

Summary

On Day 1 we achieved: - Built a **Node.js backend** with environment variables - Dockerized the backend - Learned **all commands** to build, run, and verify the container - Prepared the app for **production-ready deployment** - Understood the flow from **Code → Docker Image → Container Registry → Cloud/Server → Users**

This document serves as a **complete reference for Day 1 Docker backend setup**.