

Docker Mastery Project: From Containers to Custom Images & Multi-Container Applications

Author: Oluwaseun Osunsola

LinkedIn: <https://www.linkedin.com/in/oluwaseun-osunsola-95539b175/>

Environment & Tool: AWS, Docker, Docker Compose, Dockerfile

Project Link: <https://github.com/Oluwaseunoa/DevOps-Projects/tree/main/Docker-Projects>

1. Introduction

Containerization is a cornerstone technology in modern DevOps and DevSecOps practices. This comprehensive hands-on project provides a complete progression from foundational Docker concepts to advanced real-world application deployment.

The project is structured into **two distinct sessions**:

- **Session 1:** Core Docker Container Operations – Mastering lifecycle management, interactivity, data persistence, and cleanup using the official Ubuntu image.
- **Session 2:** Advanced Docker Application Development – Building a custom Node.js web application image with Dockerfile and orchestrating a full multi-container stack (Node.js app + MongoDB + Mongo Express) using Docker Compose.

This structured approach demonstrates end-to-end containerized application development, deployment, and verification.

2. Project Objectives

- Master the Docker container lifecycle and essential management commands
 - Build and version custom Docker images from scratch
 - Orchestrate complex multi-container applications with Docker Compose
 - Implement secure environment variable management and secret handling
 - Achieve true data persistence using volumes and MongoDB
 - Expose services securely and verify full-stack functionality in a cloud environment
-

3. Prerequisites

- Docker Engine or Docker Desktop installed
 - Basic knowledge of Linux commands and Node.js fundamentals
 - Text editor (nano or VS Code)
 - AWS EC2 Ubuntu instance with Docker and Docker Compose installed (for Session 2 deployment)
 - Internet access for pulling images and package installation
-

Session 1: Core Container Operations

This session covers 23 foundational steps focused on Ubuntu container management.

Step 1: View Existing Containers

Command: docker ps -a

```
MINGW64:/c/Users/HP
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker run ubuntu

HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f0c50b43a560 ubuntu "/bin/bash" 12 seconds ago Exited (0) 10 seconds ago sharp_austin
d53eb832d951 hello-world "/hello" About a minute ago Exited (0) 59 seconds ago cool_hodgkin
41148da79ff2 ubuntu "/bin/bash" About a minute ago Exited (0) About a minute ago quirky_keller

HP@DESKTOP-I9M74R1 MINGW64 ~
$
```

Step 2: Start a Stopped Container

Command: docker start <container_id>

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker start f0c50b43a560
f0c50b43a560

HP@DESKTOP-I9M74R1 MINGW64 ~
$
```

Step 3: Run Container with Advanced Options (-e, -v, -p)

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker run -e "MY_VARIABLE=my-value" ubuntu
```

Step 4: Run Container in Detached Mode

Command: docker run -d ubuntu

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker run -d ubuntu
056277d6245e87269943facd85ddaafb34142d071893f01dc27b2a32b3f4ca29

HP@DESKTOP-I9M74R1 MINGW64 ~
$
```

Step 5: Start Container by Name

Command:

```
docker ps -a  
docker start <container_name>
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~  
$ docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
d53eb832d951 hello-world "/hello" 33 minutes ago Exited (0) 33 minutes ago  
41148da79ff2 ubuntu "/bin/bash" 34 minutes ago Exited (0) 5 minutes ago  
  
HP@DESKTOP-I9M74R1 MINGW64 ~  
$ docker start quirky_keller  
quirky_keller  
  
HP@DESKTOP-I9M74R1 MINGW64 ~  
$
```

Step 6: Stop Container by Name

Command: `docker stop <container_name>`

```
HP@DESKTOP-I9M74R1 MINGW64 ~  
$ docker stop quirky_keller  
quirky_keller  
  
HP@DESKTOP-I9M74R1 MINGW64 ~  
$
```

Step 7: Restart Container by Name

Command: `docker restart <container_name>`

```
HP@DESKTOP-I9M74R1 MINGW64 ~  
$ docker restart quirky_keller  
quirky_keller  
  
HP@DESKTOP-I9M74R1 MINGW64 ~  
$
```

Step 8: Remove a Container

Commands:

```
docker rm <container_name>  
docker ps -a
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker rm quirky_keller
quirky_keller

HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker ps -a
CONTAINER ID   IMAGE      COMMAND   CREATED    STATUS     PORTS   NAMES
d53eb832d951   hello-world   "/hello"  44 minutes ago   Exited (0) 44 minutes ago   cool_hodgkin

HP@DESKTOP-I9M74R1 MINGW64 ~
$
```

Step 9: Pull the Ubuntu Image

Command: docker pull ubuntu

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:c35e29c9450151419d9448b0fd75374fec4ffff364a27f176fb458d472dfc9e54
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

Step 10: Run Ubuntu in Interactive Mode

Command: docker run -it --name my-ubuntu ubuntu bash

```
root@8c0a1d9ae67b:/#
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker run -it --name my-ubuntu ubuntu bash
root@8c0a1d9ae67b:/#
```

Step 11: Verify OS Information

Command: cat /etc/os-release

```
root@8c0a1d9ae67b:/# cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.3 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
root@8c0a1d9ae67b:/#
```

Step 12: Navigate File System**Commands:**

```
pwd
ls
```

```
root@8c0a1d9ae67b:/# pwd
/
root@8c0a1d9ae67b:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@8c0a1d9ae67b:/#
```

Step 13: Create and Enter Directory**Commands:**

```
mkdir my-folder
cd my-folder && pwd
```

```
root@8c0a1d9ae67b:/# mkdir my-folder
root@8c0a1d9ae67b:/# cd my-folder
```

Step 14: Create and View File**Commands:**

```
echo "This is Ubuntu in docker" > file1.txt && cat file1.txt
```

```
root@8c0a1d9ae67b:/my-folder# echo "This is Ubuntu in docker" > file.txt
root@8c0a1d9ae67b:/my-folder# cat file.txt
This is Ubuntu in docker
```

Step 15: Delete Directory

Commands:

```
cd ..  
rm -rf my-folder
```

```
root@8c0a1d9ae67b:/my-folder# cd ..  
root@8c0a1d9ae67b:/# rm -rf my-folder/  
root@8c0a1d9ae67b:/#
```

Step 16: Create Persistence Test File

Commands:

```
echo "Check after restart" > file2.txt  
cat file2.txt
```

```
root@8c0a1d9ae67b:/# echo "Check after restart" > file2.txt  
root@8c0a1d9ae67b:/# cat file2.txt  
Check after restart  
root@8c0a1d9ae67b:/#
```

Step 17: Exit the Container

Command: `exit`

```
root@8c0a1d9ae67b:/# exit  
exit  
  
HP@DESKTOP-I9M74R1 MINGW64 ~  
$
```

Step 18: Confirm Container Stopped

Command: `docker ps`

```
root@8c0a1d9ae67b:/# exit  
exit  
  
HP@DESKTOP-I9M74R1 MINGW64 ~  
$ docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
  
HP@DESKTOP-I9M74R1 MINGW64 ~  
$ 1
```

Step 19: Restart and Reattach

Commands:

```
docker start my-ubuntu  
docker attach my-ubuntu
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker restart my-ubuntu
my-ubuntu

HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker attach my-ubuntu
root@8c0a1d9ae67b:/#
```

Step 20: Verify File Persistence

Commands: `ls -ltr file2.txt`

```
root@8c0a1d9ae67b:/# ls -ltr file2.txt
-rw-r--r-- 1 root root 20 Dec 19 12:15 file2.txt
root@8c0a1d9ae67b:/#
```

Step 21: Exit Again and Confirm Stopped

Commands:

```
exit
docker ps
```

```
root@8c0a1d9ae67b:/# exit
exit

HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

HP@DESKTOP-I9M74R1 MINGW64 ~
$
```

Step 22: List All Containers Before Cleanup

Command: `docker ps -a`

```
HP@DESKTOP-I9M74R1 MINGW64 ~
$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
8c0a1d9ae67b ubuntu "bash" 53 minutes ago Exited (0) 7 minutes ago
d53eb832d951 hello-world "/hello" 2 hours ago Exited (0) 2 hours ago
my-ubuntu
cool_hodgkin

HP@DESKTOP-I9M74R1 MINGW64 ~
$
```

Step 23: Remove the Container

Commands:

```
docker rm my-ubuntu && docker ps -a
```

```
HP@DESKTOP-I9M74R1 MINGW64 ~  
$ docker rm my-ubuntu  
my-ubuntu
```

Session 2: Building & Orchestrating a Custom Node.js Profile Editor Application

This session demonstrates building a full-stack web application with persistent data storage, deployed on an AWS EC2 instance.

Step 24: Create Project Directory

Command:

```
mkdir "NodeJS Demo Application - Custom Image"  
ls
```

```
ubuntu@ip-172-31-27-253:~$ mkdir "NodeJS Demo Application - Custom Image"  
ubuntu@ip-172-31-27-253:~$ ls  
'NodeJS Demo Application - Custom Image'  
ubuntu@ip-172-31-27-253:~$ █
```

Step 25: Navigate into Project Folder

Command:

```
cd "NodeJS Demo Application - Custom Image"
```

```
☰ MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project □  
  
ubuntu@ip-172-31-27-253:~$ cd "NodeJS Demo Application - Custom Image"/  
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ █
```

Step 26: Create and Enter `app` Directory

Commands:

```
mkdir app  
cd app
```

MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Introductory Project

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ mkdir app  
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ cd app  
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$
```

Step 27: Create images Directory**Commands:**

```
mkdir images ; cd images
```

MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Introductory Project

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$ mkdir images ; cd ..  
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app/images$
```

Step 28: Upload Profile Image via SCP**Command (from local machine):**

```
scp -i your-key.pem profile-1.jpg ubuntu@<ec2-ip>:~/NodeJS\ Demo\ Application\ -\Custom\ Image/app/images/
```

MINGW64:/c/Users/HP/Downloads

```
HP@DESKTOP-I9M74R1 MINGW64 ~/Downloads  
$ scp -i MyKeyPair.pem "C:/Users/HP/Downloads/profile-1.jpg" "\ubuntu@ec2-34-226-197-236.compute-1.amazonaws.com:/home/ubuntu/NodeJS Demo Application - Custom Image/app/images/"  
profile-1.jpg 100% 29KB 57.0KB/s 00:00  
HP@DESKTOP-I9M74R1 MINGW64 ~/Downloads  
$
```

Step 29: Verify Image Upload

Command: ls

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app/images$ ls  
profile-1.jpg  
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app/images$
```

Step 30: Return to app Directory

Command: cd ..

```
= MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app/images$ ls
profile-1.jpg
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app/images$ cd ..
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$
```

Step 31: Create `index.html` (Frontend)

Command: nano index.html

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project

GNU nano 7.2                                         index.html *

const contEdit = document.getElementById('container-edit');
const cont = document.getElementById('container');

document.getElementById('input-name').value = document.getElementById('name').textContent;
document.getElementById('input-email').value = document.getElementById('email').textContent;
document.getElementById('input-interests').value = document.getElementById('interests').textContent;

cont.style.display = 'none';
contEdit.style.display = 'block';
}
</script>
<body>
<div class='container' id='container'>
<h1>User profile</h1>
<img src='profile-picture' alt="user-profile">
<span>Name: </span><h3 id='name'>Oluwaseun Osunsola</h3>
<hr />
<span>Email: </span><h3 id='email'>oluwaseun.osunsola@example.com</h3>
<hr />
<span>Interests: </span><h3 id='interests'>cybersecurity, DevOps, Coding & Travelling</h3>
<hr />
<button class='button' onclick="updateProfile()">Edit Profile</button>
</div>
<div class='container' id='container-edit'>
<h1>User profile</h1>
<img src='profile-picture' alt="user-profile">
<span>Name: </span><label for='input-name'></label><input type="text" id='input-name' value='Anna Smith' />
<hr />
<span>Email: </span><label for='input-email'></label><input type="email" id='input-email' value='anna.smith@example.com' />
<hr />
<span>Interests: </span><label for='input-interests'></label><input type="text" id='input-interests' value='coding' />
<hr />
<button class='button' onclick="handleUpdateProfileRequest()">Update Profile</button>
</div>
</body>
</html>
```

Step 32: Create `server.js` (Express Backend)

Command: nano server.js


```

GNU nano 7.2                               server.js *

res.send(userObj);
});

app.get('/get-profile', (req, res) => {
  MongoClient.connect(mongoUrlDocker, mongoClientOptions, (err, client) => {
    if (err) return res.status(500).send({});
    let db = client.db(databaseName);

    db.collection("users").findOne({ userid: 1 }, (err, result) => {
      client.close();
      res.send(result || {});
    });
  });
});

app.listen(3000, () => console.log("Your app listening on port 3000!"));

```

File menu: Help, Write Out, Where Is, Cut, Execute, Location, Undo, Set Mark, To Bracket, Previous, Back, Read File, Replace, Paste, Justify, Go To Line, Redo, Copy, Where Was, Next, Forward, Prev Word, Next Word.

Step 33: Initialize npm Project**Command: npm init -y**

```

ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (app) nodejs-app
version: (1.0.0)
description: This is a nodejs app runs by docker
entry point: (server.js)
test command:
git repository:
keywords:
author: Oluwaseun Osunsola
license: (ISC)
About to write to /home/ubuntu/NodeJS Demo Application - Custom Image/app/package.json:

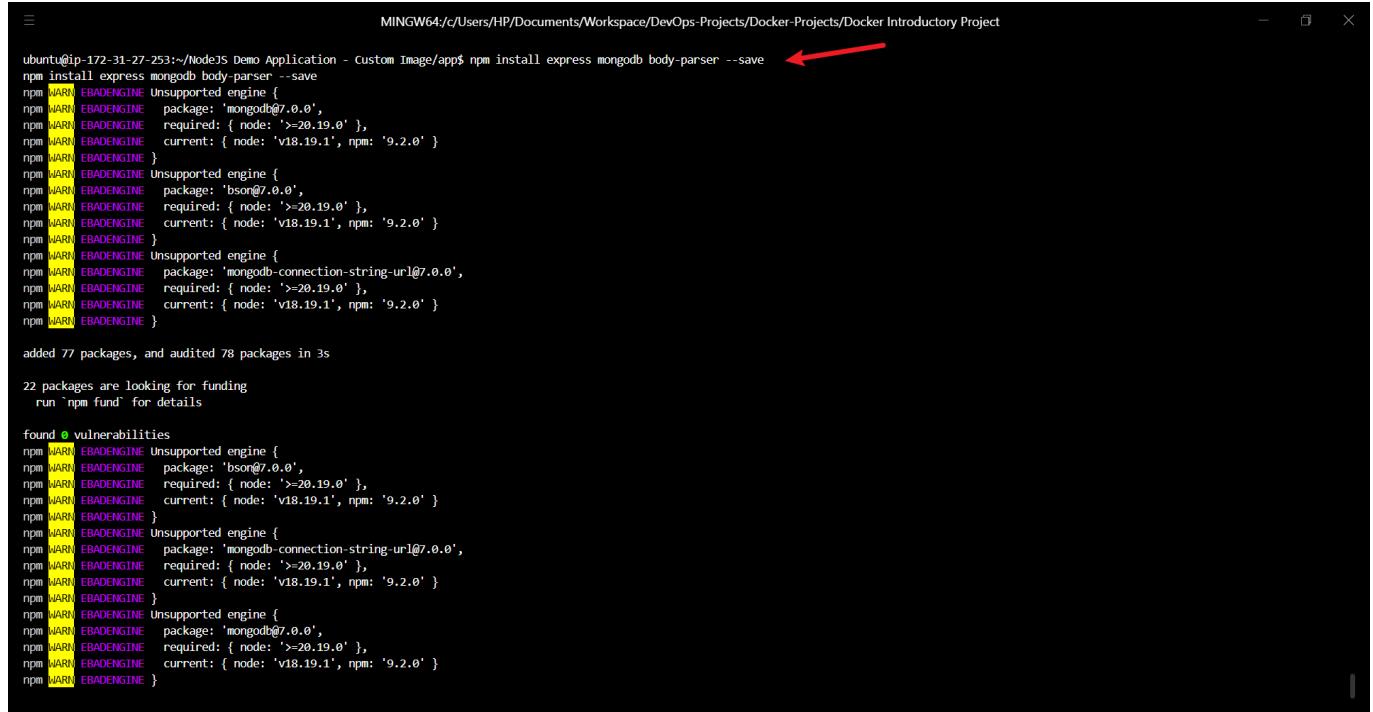
{
  "name": "nodejs-app",
  "version": "1.0.0",
  "description": "this is a nodejs app runs by docker",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "Oluwaseun Osunsola",
  "license": "ISC"
}

Is this OK? (yes) yes

```

Step 34: Install Dependencies

Command: `npm install express body-parser mongodb dotenv`



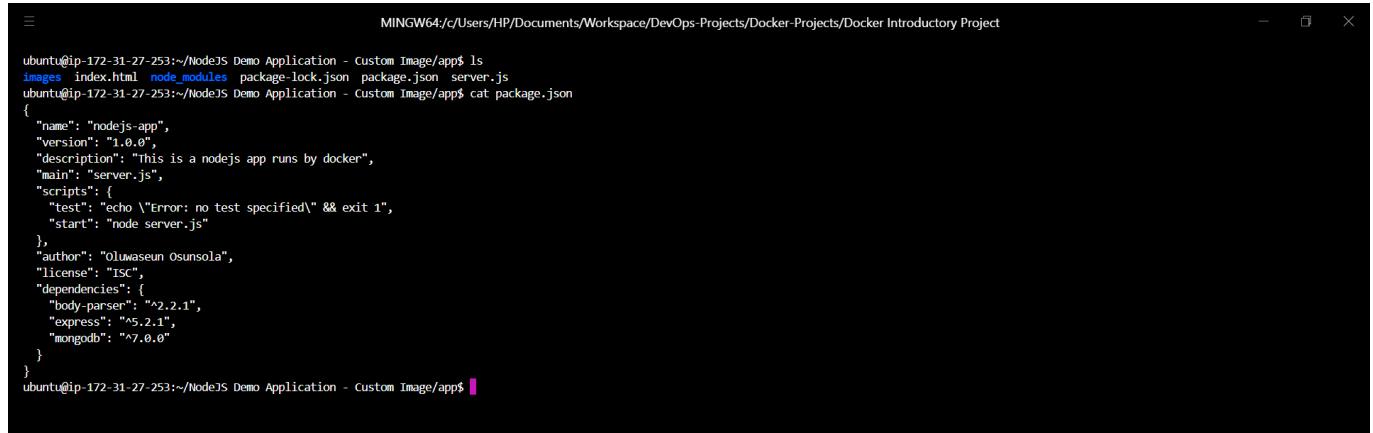
```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$ npm install express mongodb body-parser --save
npm install express mongodb body-parser --save
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'mongodb@7.0.0',
npm WARN EBADENGINE   required: { node: '>=20.19.0' },
npm WARN EBADENGINE   current: { node: 'v18.19.1', npm: '9.2.0' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'bson@7.0.0',
npm WARN EBADENGINE   required: { node: '>=20.19.0' },
npm WARN EBADENGINE   current: { node: 'v18.19.1', npm: '9.2.0' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'mongodb-connection-string-url@7.0.0',
npm WARN EBADENGINE   required: { node: '>=20.19.0' },
npm WARN EBADENGINE   current: { node: 'v18.19.1', npm: '9.2.0' }
npm WARN EBADENGINE }
added 77 packages, and audited 78 packages in 3s

22 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
```

Step 35: Verify Installed Packages

Commands: `ls && cat package.json`



```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$ ls
images index.html node_modules package-lock.json package.json server.js
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$ cat package.json
{
  "name": "nodejs-app",
  "version": "0.0.0",
  "description": "this is a nodejs app runs by docker",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "Oluwaseun Osunsola",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^2.2.1",
    "express": "5.2.1",
    "mongodb": "7.0.0"
  }
}
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$
```

Step 36: Return to Project Root

Command: `cd ..`



```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$ cd ..
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$
```

Step 37: Create .env File

Secure storage of MongoDB credentials.



```
GNU nano 7.2
MONGO_USER=admin
MONGO_PASSWORD=password
```

Step 38: Create .gitignore

Content: node_modules/, .env

```
GNU nano 7.2
# Node
node_modules/
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Environment variables
.env
.env.*

# Docker
*.log

# OS
.DS_Store
Thumbs.db
.

File Name to Write: .gitignore *
.gitignore *

^C Help      ^O Write Out    ^W Where Is     ^K Cut          ^X Execute      ^A Location    M-U Undo      M-A Set Mark   M-] To Bracket  M-C Previous  AB Back       ^N Next Word
^X Exit      ^R Read File    ^V Replace     ^C Paste        ^Y Justify      ^V Go To Line  M-U Redo      M-B Copy      M-Q Where Was  M-L Next      M-F Forward   ^P Prev Word
                                         M-M Mac Format
                                         M-P Prepend
                                         M-B Backup File
                                         M-I Browse
```

Step 39: Create Dockerfile

Multi-stage build for Node.js application.

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project
Dockerfile *

GNU nano 7.2
FROM node:18-alpine
WORKDIR /home/app
COPY app/package*.json .
RUN npm install
COPY app/ .
EXPOSE 3000
CMD ["node", "server.js"]
```

Step 40: Build Custom Image

Command: docker build -t node-app:1.0 .

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker build -t node-app:1.0 .
[+] Building 9.1s (10/10) FINISHED
--> [internal] load build definition from Dockerfile
--> [internal] transfering dockerfile: 174B
--> [internal] load metadata for docker.io/library/node:18-alpine
--> [internal] load .dockerignore
--> transferring context: 2B
--> [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
--> extracting docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
--> sha256:25ff2da3041908f65c3a74d0a09db1b62ccfaab220be9a70b80df5a2e0549 446B 0.05
--> sha256:d71de834b5c203d16290zeeb994cb2309ae049a0eabc4feea161b2b5a3d0e 40,01MB 0.05
--> sha256:1e5a4c89cee5c0826c540a06d4b6b491c96eda01837f430bd47fd26702d0e3 1.26MB 1.26
--> sha256:18a2321740c91741ffd3da96085011092101a32a93a388b79e999e69c2d5c070 1.26
--> sha256:18a2321740c91741ffd3da96085011092101a32a93a388b79e999e69c2d5c070 1.26
--> extracting sha256:18a2321740c91741ffd3da96085011092101a32a93a388b79e999e69c2d5c070
--> sha256:d71de834b5c203d16290zeeb994cb2309ae049a0eabc4feea161b2b5a3d0e 0.05
--> sha256:1e5a4c89cee5c0826c540a06d4b6b491c96eda01837f430bd47fd26702d0e3 0.05
--> sha256:25ff2da3041908f65c3a74d0a09db1b62ccfaab220be9a70b80df5a2e0549 0.05
--> [internal] load build context
--> transferring context: 8.91MB 0.05
--> [2/5] WORKDIR /home/app 0.1s
--> [3/5] COPY app/package*.json ./ 0.1s
--> [4/5] RUN npm install 2.95
--> [5/5] COPY app/ . 0.1s
--> exporting to image 1.55
--> exporting layers 0.75
--> exporting manifest sha256:f86276f21f437e6947a8b493eb4d2fdfd15ecc5b875a9f0219d295fb0fe47fb 0.05
--> exporting config sha256:6451e96eaadef5ede4f2ce2bc27dd5d45b7497ceea7b08482ce97584ad5 0.05
--> exporting attestation manifest sha256:b9c82d9e21c115a01fd1a9e42f33125d323c6c81a4da88b36ad1f1ccbda5b 0.05
--> exporting manifest list sha256:c835f51741e3ac5141a5301567d1db3610e7c4491176f96678b19291b59114f4 0.05
--> naming to docker.io/library/node-app:1.0 0.05
--> unpacking to docker.io/library/node-app:1.0 0.05
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$
```

Step 41: Verify Custom Image

Command: docker images

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker images
IMAGE           ID             DISK USAGE      CONTENT SIZE   EXTRA
hello-world:latest d4aab6242e0    25.9kB        9.5kB        U
nginx:1.29.4-alpine 052b75ab7f6    82MB         23.9MB       U
node-app:1.0       c835f51741e3   215kB        50.8MB       U
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$
```

Step 42: Create docker-compose.yml

Orchestrates my-app, mongodb, and mongo-express services with volumes and networks.

```
GNU nano 7.2
services:
  my-app:
    image: node-app:1.0
    ports:
      - "3000:3000"
    environment:
      MONGO_USER: ${MONGO_USER}
      MONGO_PASSWORD: ${MONGO_PASSWORD}
    depends_on:
      - mongodb

  mongodb:
    image: mongo:6
    environment:
      MONGO_INITDB_ROOT_USERNAME: ${MONGO_USER}
      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_PASSWORD}
    volumes:
      - mongo-data:/data/db

  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - "8081:8081"
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: ${MONGO_USER}
      ME_CONFIG_MONGODB_ADMINPASSWORD: ${MONGO_PASSWORD}
      ME_CONFIG_MONGODB_SERVER: mongodb
    depends_on:
      - mongodb

volumes:
  mongo-data:
```

File Name to Write: docker-compose.yml

Step 43: Start Application Stack

Command: docker compose --env-file .env up -d

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker compose --env-file .env up -d
[+] up 4/4
✓ Network nodejsdemoproject-customimage default          Created          0.1s
✓ Container nodejsdemoproject-customimage-mongodb-1      Created          0.1s
✓ Container nodejsdemoproject-customimage-mongo-express-1 Created          0.1s
✓ Container nodejsdemoproject-customimage-my-app-1       Created          0.1s
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$
```

Step 44: Verify Running Containers

Command: docker ps

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
d35ff041ed31      mongo-express      "/sbin/tini -- /dock_."   4 minutes ago     Up 32 seconds    0.0.0.0:8080->8081/tcp, [::]:8080->8081/tcp   nodejsdemoproject-customimage-mongo-express-1
7df87c710d4       node-app:1.0       "docker-entrypoint.s..."  4 minutes ago     Up 4 minutes     0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp   nodejsdemoproject-customimage-my-app-1
02151f3ddd5d      mongo:6           "docker-entrypoint.s..."  4 minutes ago     Up 4 minutes     27017/tcp          nodejsdemoproject-customimage-mongodb-1
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$
```

Step 45: Check Application Logs

Command: docker logs <container-name>

```
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
d35ff041ed31      mongo-express      "/sbin/tini -- /dock_."   6 minutes ago     Up 50 seconds    0.0.0.0:8080->8081/tcp, [::]:8080->8081/tcp   nodejsdemoproject-customimage-mongo-express-1
7df87c710d4       node-app:1.0       "docker-entrypoint.s..."  6 minutes ago     Up 6 minutes     0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp   nodejsdemoproject-customimage-my-app-1
02151f3ddd5d      mongo:6           "docker-entrypoint.s..."  6 minutes ago     Up 6 minutes     27017/tcp          nodejsdemoproject-customimage-mongodb-1
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker logs nodejsdemoproject-customimage-my-app-1
Your app listening on port 3000!
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$
```

Step 46: Configure EC2 Security Group

Open ports 3000 and 8080.

The screenshot shows the AWS EC2 Security Groups console. On the left, there's a navigation sidebar with options like EC2, Instances, Images, and Network & Security. The main area shows a security group named 'sg-04814b4e21631478a - launch-wizard-1'. Under the 'Inbound rules' tab, there are two entries:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0835b5e4abac0f4b4	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-04147d3bc2436d405	IPv4	SSH	TCP	22	0.0.0.0/0	-

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0835b5e4abac0f4b4	HTTP	TCP	80	Custom	
sgr-04147d3bc2436d405	SSH	TCP	22	Custom	
-	Custom TCP	TCP	8080	Anywhere	
-	Custom TCP	TCP	3000	Anywhere	

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

Step 48: Access Mongo Express Admin UI

<http://<ec2-public-ip>:8080>

Mongo Express

Databases

		Database Name	+ Create Database
	View	admin	
	View	config	
	View	local	

Server Status

Hostname	c91781124524	MongoDB Version	8.2.2
Uptime	393 seconds	Node Version	18.20.3
Server Time	Wed, 17 Dec 2025 13:28:06 GMT	V8 Version	10.2.154.26-node.37
Current Connections	3	Available Connections	406
Active Clients	0	Queued Operations	0
Clients Reading	0	Clients Writing	0

Step 49: Create Database my-db

Mongo Express

Databases

	admin		
	config		
	local		
	my-db		

Server Status

Hostname	c91781124524	MongoDB Version	8.2.2
Uptime	1203 seconds	Node Version	18.20.3
Server Time	Wed, 17 Dec 2025 13:41:36 GMT	V8 Version	10.2.154.26-node.37
Current Connections	3	Available Connections	406
Active Clients	0	Queued Operations	0
Clients Reading	0	Clients Writing	0

Step 50: Create `users` Collection

Viewing Database: my-db

Collections

				delete_me	

Database Stats

Collections (incl. system.namespaces)	1
Data Size	0 Byte
Storage Size	4.10 KB
Avg Obj Size #	0 Byte
Indexes #	1
Index Size	4.10 KB

Step 51: Access Profile Editor Application

<http://<ec2-public-ip>:3000>

User Profile

Name: Oluwaseun Osunsola

Email: oluwaseun.osunsola@example.com

Interests: DevOps, Cybersecurity, & Coding

Edit Profile

Step 52: Test Data Persistence

Edit profile → Save → Reload → Changes persist via MongoDB

Edit Profile

Name: Oluwaseun Osunsola

Email: oluwaseun.osunsola@example.com

Interests: I just want to be eating

Update Profile Cancel

4. Key Learning Outcomes

- Complete mastery of Docker container lifecycle and management commands
- Ability to build optimized, versioned custom images using Dockerfile
- Proficiency in orchestrating multi-container applications with Docker Compose
- Secure handling of environment variables and secrets
- Implementation of persistent storage across container restarts
- Real-world deployment and networking configuration in cloud environments
- Full-stack application integration with database persistence

5. Conclusion

This dual-session Docker Mastery Project successfully bridges foundational container operations with advanced production-grade application development and orchestration.

Session 1 established strong command-line proficiency and deep understanding of container behavior, persistence, and lifecycle management.

Session 2 elevated these skills into a complete DevOps workflow: building custom images, managing dependencies securely, orchestrating interdependent services, and deploying a fully functional, persistent web application in the cloud.

The hands-on execution across 52 documented steps demonstrates professional-level competency in containerization—critical for roles in **DevOps**, **DevSecOps**, and **Cloud Engineering**.

This project serves as a solid portfolio piece showcasing practical expertise in modern container technologies and prepares for advanced topics such as CI/CD pipelines, image security scanning, Kubernetes orchestration, and production monitoring.

Future Enhancements:

- Integrate CI/CD with GitHub Actions and Docker Hub
- Add image vulnerability scanning (Trivy/Grype)
- Implement HTTPS with Nginx reverse proxy and Let's Encrypt
- Scale with Docker Swarm or migrate to Kubernetes
- Add authentication and input validation for enhanced security

Mastery of these concepts positions you at the forefront of cloud-native development and infrastructure management.