

Getting Started with Docker and Node.js: A Comprehensive Hands-On Project

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>

Project Overview

This comprehensive guide walks you through installing Docker on an AWS EC2 Ubuntu instance, mastering fundamental Docker commands, building a custom Docker image for a Node.js application, and orchestrating a multi-container setup using Docker Compose. The application is a full-stack user profile editor built with Express.js, connected to MongoDB for persistent storage, and includes Mongo Express as a web-based admin interface.

By the end of this tutorial, you will have:

- A running Dockerized Node.js web application accessible publicly.
- Persistent data storage in MongoDB.
- Hands-on experience with Docker fundamentals, image building, and container orchestration.
- Understanding of networking, security groups, and environment variable management in containerized environments.

Target Audience: Beginners to intermediate learners in DevOps, Docker, Node.js, and cloud deployment (AWS EC2).

Prerequisites

Before starting:

- An active AWS account.
- Basic familiarity with the Linux terminal, SSH, and AWS EC2.
- A local machine with SSH client and SCP support.
- A profile picture (JPG format) to upload for the application demo.

Detailed Step-by-Step Guide

The following steps are executed in sequence on the Ubuntu EC2 instance (except where noted). Each step includes a brief explanation, the commands used (where applicable), and a screenshot reference.

1. Launch Ubuntu Server

Launch a new EC2 instance running Ubuntu Server (e.g., 22.04 LTS). Select an appropriate instance type (t2.micro is sufficient for this demo), configure a key pair for SSH access, and ensure the security group allows inbound SSH (port 22).

The screenshot shows the AWS EC2 Instances page. A single instance, "Ubuntu Server" (i-08867746e834c4c8a), is listed as "Running" in the t3.micro instance type. The instance has a Public IPv4 DNS of ec2-98-91-21-246.com... and a Public IPv4 IP of 98.91.21.246. The instance is in the us-east-1c availability zone. The left sidebar shows various EC2 management options.

2. Connect to the Ubuntu Server via SSH

From your local machine, connect using:

```
ssh -i /path/to/your-key.pem ubuntu@ec2-public-ip
```

```
MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project
HPDESKTOP-I9M74R1 MINGW64 ~/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project (main)

$ ssh -i "C:/Users/HP/Downloads/MyKeyPair.pem" ubuntu@ec2-98-91-21-246.compute-1.amazonaws.com
The authenticity of host 'ec2-98-91-21-246.compute-1.amazonaws.com (98.91.21.246)' can't be established.
ED25519 key fingerprint is SHA256:W8rfv0fyAxf4Xxyar74ccQ1LYGaeHMK6CnrvVS1II.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-98-91-21-246.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue Dec 16 18:58:27 UTC 2025

System load: 0.08 Temperature: -273.1 °C
Usage of /: 25.8% of 6.71GB Processes: 110
Memory usage: 23% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.27.253

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
```

3. Update the Server

Update package lists and upgrade installed packages:

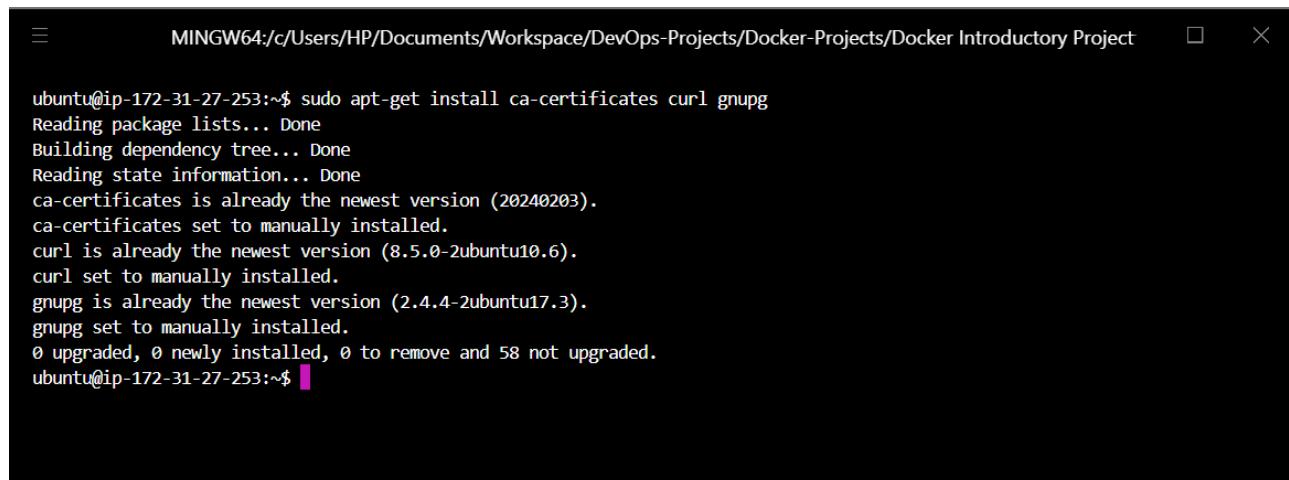
```
sudo apt update && sudo apt upgrade -y
```

```
ubuntu@ip-172-31-27-253:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1391 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1684 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [311 kB]
```

4. Install Required Packages

Install prerequisites for adding the Docker repository:

```
sudo apt install ca-certificates curl gnupg lsb-release -y
```



```
ubuntu@ip-172-31-27-253:~$ sudo apt-get install ca-certificates curl gnupg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 58 not upgraded.
ubuntu@ip-172-31-27-253:~$
```

5. Create Docker Keyring

Create directory for Docker's GPG key:

```
sudo mkdir -p /etc/apt/keyrings
```



```
ubuntu@ip-172-31-27-253:~$ sudo install -m 0755 -d /etc/apt/keyrings
ubuntu@ip-172-31-27-253:~$
```

6. Add Official Docker GPG Key

Download and add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```



```
ubuntu@ip-172-31-27-253:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
ubuntu@ip-172-31-27-253:~$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
ubuntu@ip-172-31-27-253:~$
```

7. Add Official Docker Repository

Add the Docker APT repository:

```
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

```
ubuntu@ip-172-31-27-253:~$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
ubuntu@ip-172-31-27-253:~$
```

8. Install Docker Engine

Update packages and install Docker Engine and Compose plugin:

```
sudo apt update && sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin -y
```

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

```
ubuntu@ip-172-31-27-253:~$ sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.5 kB]
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [40.8 kB]
Fetched 89.2 kB in 0s (237 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libslirp0 pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libslirp0
  pigz slirp4netns
0 upgraded, 9 newly installed, 0 to remove and 58 not upgraded.
Need to get 91.2 MB of archives.
After this operation, 363 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

9. Verify Docker Installation

Check Docker version:

```
docker --version
```

```
ubuntu@ip-172-31-27-253:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
  Active: active (running) since Tue 2025-12-16 19:10:49 UTC; 1min 47s ago
TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
 Main PID: 2264 (dockerd)
   Tasks: 9
  Memory: 93.0M (peak: 105.4M)
    CPU: 425ms
   CGroup: /system.slice/docker.service
           └─2264 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.061935832Z" level=info msg="Restoring containers"
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.117976565Z" level=info msg="Deleting nftables IP"
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.126051904Z" level=info msg="Deleting nftables IP"
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.416408870Z" level=info msg="Loading containers: >
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.434059338Z" level=info msg="Docker daemon" commi>
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.434218342Z" level=info msg="Initializing buildki>
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.483905560Z" level=info msg="Completed buildkit i>
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.494583712Z" level=info msg="Daemon has completed>
Dec 16 19:10:49 ip-172-31-27-253 dockerd[2264]: time="2025-12-16T19:10:49.494783582Z" level=info msg="API listen on /run/d
Dec 16 19:10:49 ip-172-31-27-253 systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-22/22 (END)
```

10. Run Docker Without Sudo

Add current user to Docker group:

```
sudo usermod -aG docker $USER
```

Log out and back in for changes to take effect.

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

ubuntu@ip-172-31-27-253:~$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-27-253:~$
```

11. Running the First Container (Hello World)

Test Docker with a simple container:

```
docker run hello-world
```

![11.Running_the_First Container_(Hello_World).png](img/11.Running_the_First Container_(Hello_World).png)

12. Verify Images (docker images)

List locally available images:

```
docker images
```

ID	REPOSITORY	IMAGE	SIZE	EXTRA
hello-world:latest	d4aab6242e0	25.9kB	9.52kB	U

13. List Running Containers (docker ps)

Show currently running containers:

`docker ps`



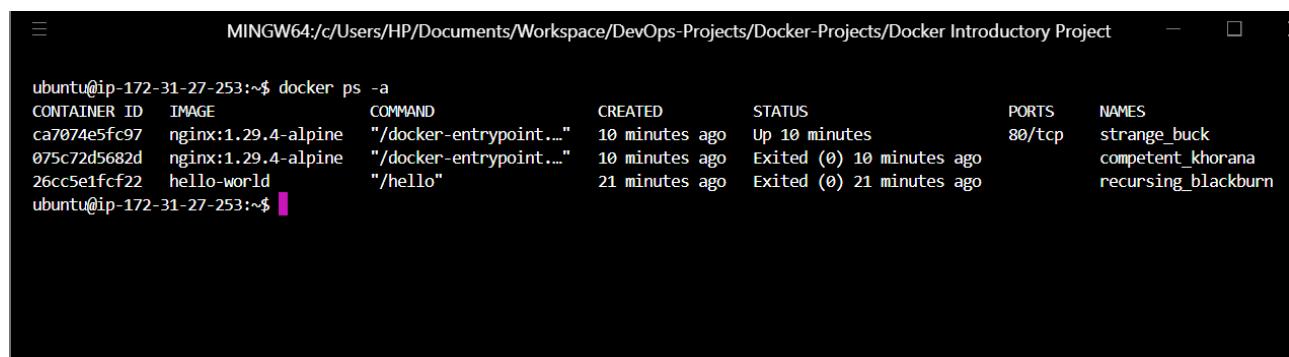
```
ubuntu@ip-172-31-27-253:~$ docker run -d nginx:1.29.4-alpine
ca7074e5fc97d9e177fd3a0d1fd93c2e72071e42c9612e73f17643c0a4b2e85
ubuntu@ip-172-31-27-253:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ca7074e5fc97 nginx:1.29.4-alpine "/docker-entrypoint..." 43 seconds ago Up 42 seconds 80/tcp strange_buck
ubuntu@ip-172-31-27-253:~$
```

`'docker ps'` : listed running instance of the nginx:1.29.4-alpine

14. List Both Running and Stopped Containers (docker ps -a)

Show all containers (running and stopped):

`docker ps -a`

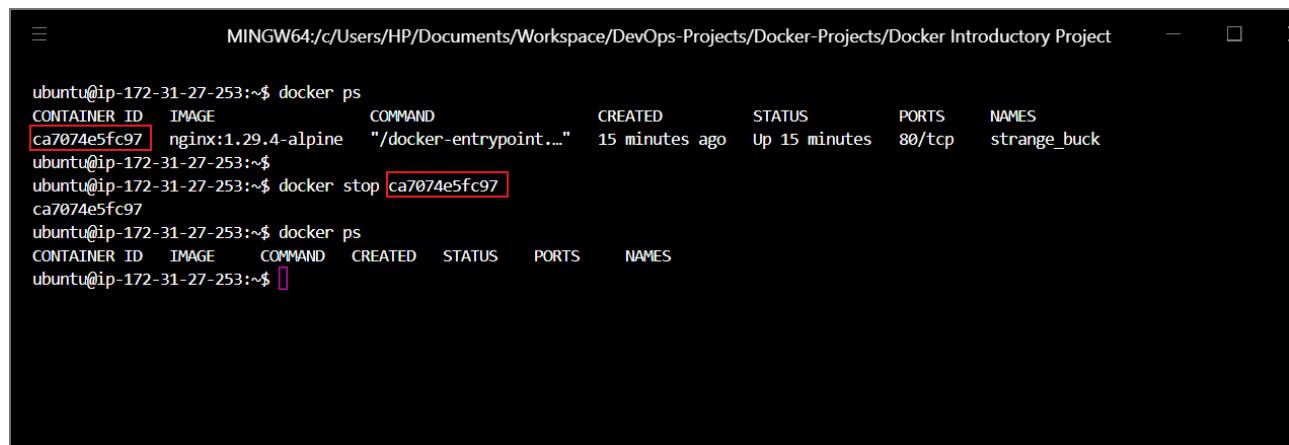


```
ubuntu@ip-172-31-27-253:~$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ca7074e5fc97 nginx:1.29.4-alpine "/docker-entrypoint..." 10 minutes ago Up 10 minutes 80/tcp strange_buck
075c72d5682d nginx:1.29.4-alpine "/docker-entrypoint..." 10 minutes ago Exited (0) 10 minutes ago
26cc5e1fcf22 hello-world "/hello" 21 minutes ago Exited (0) 21 minutes ago
ubuntu@ip-172-31-27-253:~$
```

15. Stop a Container Using Its Container ID

Stop a running container:

`docker stop <container_id>`



```
ubuntu@ip-172-31-27-253:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ca7074e5fc97 nginx:1.29.4-alpine "/docker-entrypoint..." 15 minutes ago Up 15 minutes 80/tcp strange_buck
ubuntu@ip-172-31-27-253:~$ docker stop ca7074e5fc97
ca7074e5fc97
ubuntu@ip-172-31-27-253:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ubuntu@ip-172-31-27-253:~$
```

16. Pull an Ubuntu Image

Download the official Ubuntu image:

`docker pull ubuntu`

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

ubuntu@ip-172-31-27-253:~$ docker pull ubuntu
using default tag: latest
latest: Pulling from library/ubuntu
20043066d3d5: Pull complete
06808451fd6: Download complete
Digest: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f176fb458d472dfc9e54
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
ubuntu@ip-172-31-27-253:~$
```

17. Delete Image Using Its Image ID

Remove an image:

`docker rmi <image_id>`

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

ubuntu@ip-172-31-27-253:~$ docker images
          Info → In Use
IMAGE           ID          DISK USAGE   CONTENT SIZE   EXTRA
hello-world:latest d4aaab6242e0    25.9kB      9.52kB      U
nginx:1.29.4-alpine 052b75ab72f6    82MB       23.9MB      U
ubuntu:latest     c35e29c94501    119MB      31.7MB      U
ubuntu@ip-172-31-27-253:~$ docker rmi c35e29c94501
Untagged: ubuntu:latest
Deleted: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f176fb458d472dfc9e54
ubuntu@ip-172-31-27-253:~$
```

18. Create Project Folder (`NodeJS_Demo_Application - Custom_Image`) and LS It to Confirm

Create project directory:

`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:~$
```

19. Navigate into the Project Folder (`NodeJS_Demo_Application - Custom_Image`)

`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$
```

20. MKDIR App to Create App Folder Then CD into App Folder

```
mkdir app && cd app
```

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

21. In App Create Images Folder and CD into It

```
mkdir images && cd images
```

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

22. On the Host System Upload an Image to Project Images Folder Using SCP

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
$ 
```

23. In Image Folder LS to Confirm That Image Is Successfully Uploaded

```
ls
```

```
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$ 
```

24. From Image Folder CD Back One Step to App Folder

```
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$
```

25. Nano index.html Then Add Code Save and Exit

Create and populate the frontend HTML file.

```
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>

<span>Name: </span><span id="name">Oluseun Osunsola</span>
<hr />
<span>Email: </span><span id="email">oluaseun.osunsola@example.com</span>
<hr />
<span>Interests: </span><span id="interests">cybersecurity, DevOps, Coding & Travelling</span>
<hr />
<button class="button" onclick="updateProfile()>Edit Profile</button>
</div>
<div class="container" id="container-edit">
<h1>User profile</h1>

<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>
```

26. Nano server.js Then Add Code Save and Exit

Create the Express backend server code.

```
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!"));


```

27. Initialize the Project Folder (Install Node and NPM If You Don't Have and Then Initialize)

Install Node.js if needed, then:

```
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.

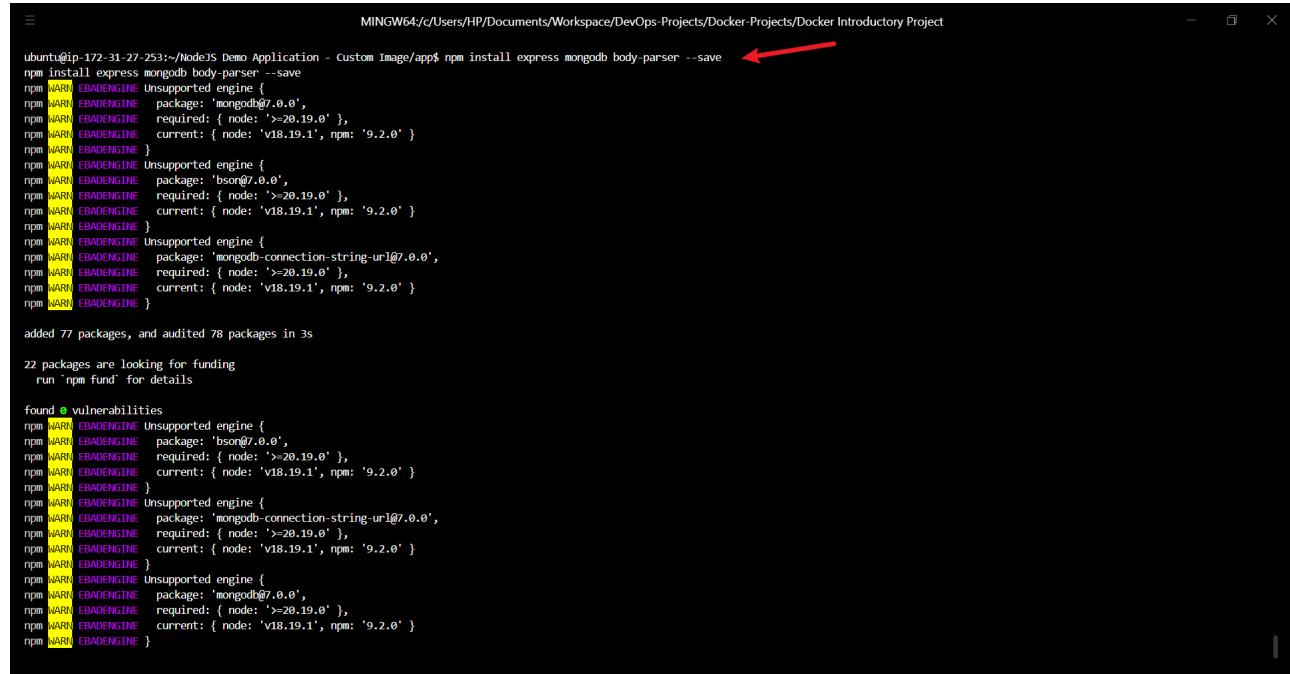
Press ^C at any time to quit.
package name: (app) nodejs-app
version: (1.0.0)
description: This is a nodejs app runs by docker
entry point: (server.js)
test command:
git repository:
keywords:
author: Oluwaseun Osunsola
license: (ISC)
About to write to /home/ubuntu/NodeJS Demo Application - Custom Image/app/package.json:

{
  "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
```

28. Install the Packages (express, body-parser, mongodb)

`npm install express body-parser mongodb`



```
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

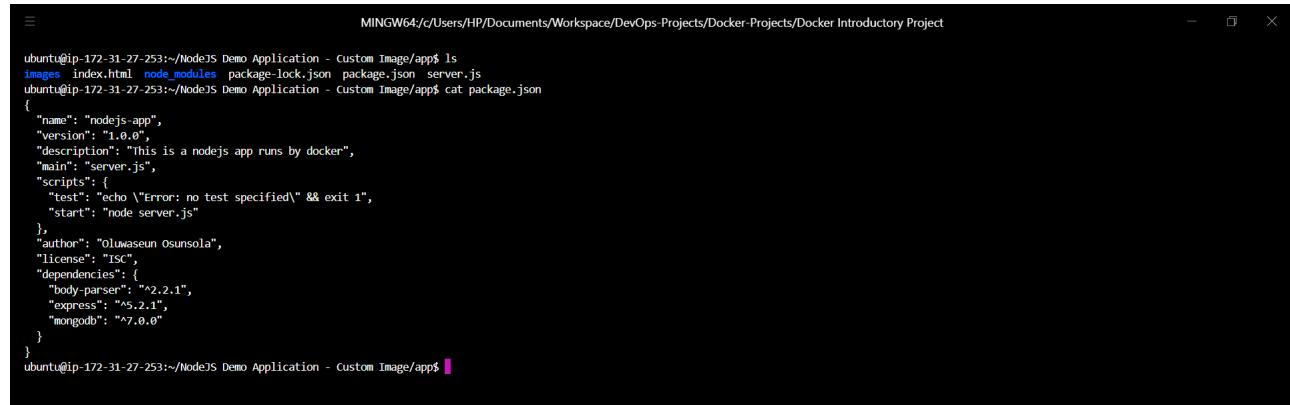
22 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
```

29. LS App Folder Cat package.json to Confirm Packages

Verify installed dependencies:

`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": "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",
  "dependencies": {
    "body-parser": "^2.2.1",
    "express": "^5.2.1",
    "mongodb": "^7.0.0"
  }
}
```

30. Navigate Back to the Project Folder

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

31. Nano .env Then Add Environment Variables Save and Exit

Create `.env` with MongoDB credentials (e.g., `MONGO_USER=admin`, `MONGO_PASSWORD=password`).

```
GNU nano 2.2  
.env *  
MONGO_USER=admin  
MONGO_PASSWORD=password
```

32. Nano .gitignore Then Add What to Ignore Save and Exit

Add `node_modules/`, `.env`, etc., to `.gitignore`.

```
GNU nano 7.2  
.gitignore *  
# Node  
node modules/  
npm-debug.log*  
yarn-debug.log*  
yarn-error.log*  
  
# Environment variables  
.env  
.env.*  
  
# Docker  
.log  
  
# OS  
.DS_Store  
Thumbs.db
```

33. Nano Dockerfile and Add Commands

Write the Dockerfile to build the Node.js image.

```

GLIBC 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"]

```

34. Build Image Using Docker Build -t node-app:1.0

`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] load metadata for docker.io/library/node:18-alpine
   => [internal] load .dockerrcignore
   => [internal] transfer context: 2B
[1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca99d9e
   => [2/5] COPY app/package*.json .
   => [3/5] RUN npm install
   => [4/5] COPY app/ .
   => [5/5] COPY app/
   => exporting to image
   => exporting layers
   => exporting manifest sha256:f86276f21f437e6947a8b493eb4d2fdfd15ecc5b875a9f0219d295fbef47fb
   => => exporting config sha256:6a51e964de1f5ede34f2cebc27dd9d3455749/ceea7b08482cce97584ad5
   => => exporting attestation manifest sha256:b9ca2d9e1c1115a91f1da9e42fa33125d233c68c1a4d488b36ad1f1ccbda5b
   => => exporting manifest list sha256:c335f51741e3ac5141a5301567d1db3e16e7c4491176f96678b19291b59114f4
   => => naming to docker.io/library/node-app:1.0
   => => unpacking to docker.io/library/node-app:1.0
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ 

```

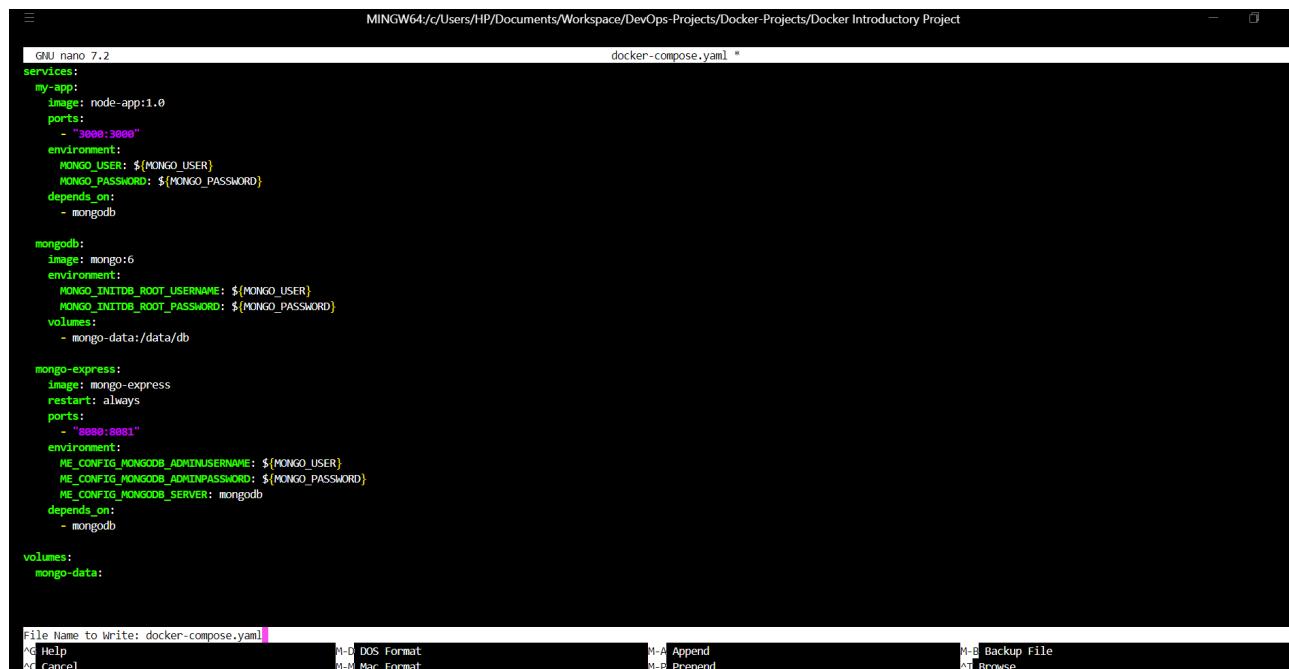
35. List Images to See the Created node-app:1.0

`docker images`

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
hello-world:latest	d4aaab6242e0	25.9kB	9.52kB	0
nginx:1.29.4-alpine	052075ab72f6	82MB	23.9MB	0
node-app:1.0	c835f51741e3	215MB	50.8MB	0

36. Nano docker-compose.yaml and Add Services and Their Data Save and Exit

Define services: `my-app`, `mongodb`, `mongo-express`.



```

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:
      - "8080:8081"
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: ${MONGO_USER}
      ME_CONFIG_MONGODB_ADMINPASSWORD: ${MONGO_PASSWORD}
      ME_CONFIG_MONGODB_SERVER: mongodb
    depends_on:
      - mongodb

volumes:
  mongo-data:

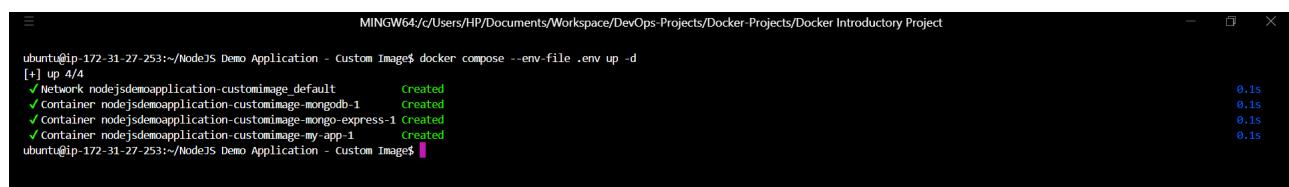
```

File Name to Write: docker-compose.yaml

Help Cancel DOS Format Mac Format Append Prepend Backup File Browse

37. Run Docker Compose --env-file .env Up -d

`docker compose up -d`



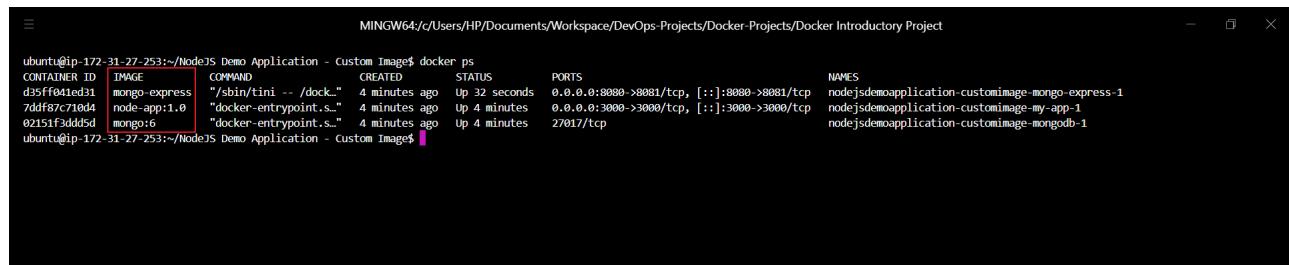
```

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

```

38. Docker PS to Check Containers Running

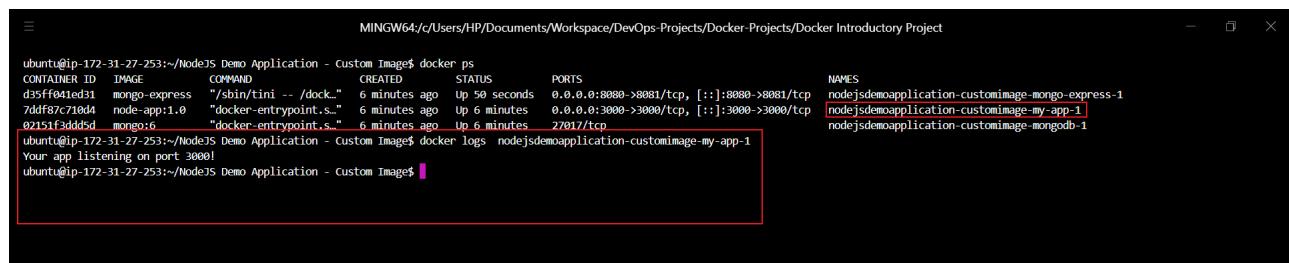
`docker ps`



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	nodejsdemapplication-customimage-mongo-express-1
7ddfc7c710d4	node-app:1.0	"docker-entrypoint.s.."	4 minutes ago	Up 4 minutes	0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp	nodejsdemapplication-customimage-my-app-1
02151f3dd5d5	mongo:6	"docker-entrypoint.s.."	4 minutes ago	Up 4 minutes	27017/tcp	nodejsdemapplication-customimage-mongodb-1

39. Docker Logs to Check Node.js Application Logs

`docker logs <my-app-container-name>`



```

ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker logs nodejsdemapplication-customimage-my-app-1
Your app listening on port 3000!
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ 

```

40. Navigate to Server EC2 Instance Security Group to Edit Inbound Rule

In AWS Console, open the instance's security group.

sg-04814b4e21631478a - launch-wizard-1

Inbound rules

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	-

41. Add Two Rules to Accept Custom TCP Connections from Anywhere on Port 3000 and 8080 Then Save Rules

Add rules: Type=Custom TCP, Port=3000 & 8080, Source=0.0.0.0/0.

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0835b5e4abac0f4b4	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-04147d3bc2436d405	SSH	TCP	22	Custom	0.0.0.0/0
-	Custom TCP	TCP	8080	Anywhere	0.0.0.0/0
-	Custom TCP	TCP	3000	Anywhere	0.0.0.0/0

⚠️ 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.

Save rules

42. Visit the Public IP on Port 8080 to See Mongo Express

<http://ec2-public-ip:8080> → Login with credentials from `.env`.

Mongo Express

Databases

	View	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

43. Create my-db Database

In Mongo Express, create database **my-db**.

Mongo Express

Databases

	View	Database Name	+ Create Database
	View	admin	
	View	config	
	View	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

44. Click on my-db and Create Users Collection

Create collection **users**.

Mongo Express Database: my-db

Collections

+ Create collection

View Export JSON Import delete_me Del

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

45. Visit the Public IP on Port 3000 to See the Application

<http://ec2-public-ip:3000> → Profile editor loads.

User Profile

Name: Oluwaseun Osunsola

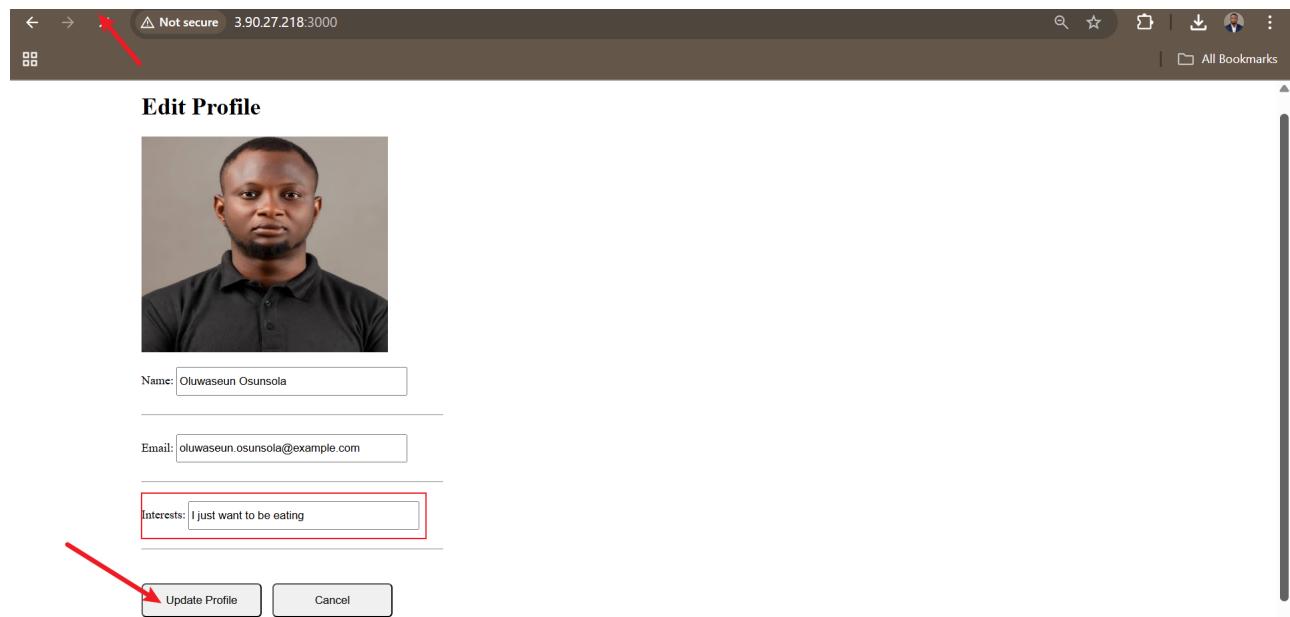
Email: oluwaseun.osunsola@example.com

Interests: DevOps, Cybersecurity, & Coding

Edit Profile

46. Edit and Update Profile Then Reload to See If Change Is Persistent Across Reload

Edit fields, save, reload page → Data persists (stored in MongoDB).



Key Learnings & Best Practices

- **Docker Fundamentals:** Images, containers, volumes, networking.
- **Custom Image Building:** Multi-stage or simple builds for Node.js apps.
- **Docker Compose:** Orchestrating multiple services with dependencies.
- **Environment Variables:** Secure handling via `.env` files.
- **Persistence:** MongoDB volume ensures data survives container restarts.
- **Security Considerations:** Restrict security group rules in production (avoid 0.0.0.0/0).

Troubleshooting Tips

- Container restarting? Check logs: `docker compose logs my-app`.
- Connection refused? Verify security group ports and `docker ps`.
- DB not connecting? Ensure env vars are passed and MongoDB is healthy.
- Image build issues? Use `--no-cache` flag.

Conclusion

This project provides a complete, real-world introduction to containerization with Docker. You now have a deployable full-stack application running on cloud infrastructure. Extend this further by adding authentication, CI/CD pipelines, or deploying to Kubernetes.

Congratulations on completing the tutorial! Your Dockerized Node.js + MongoDB application is live and production-ready for demo purposes.