

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:

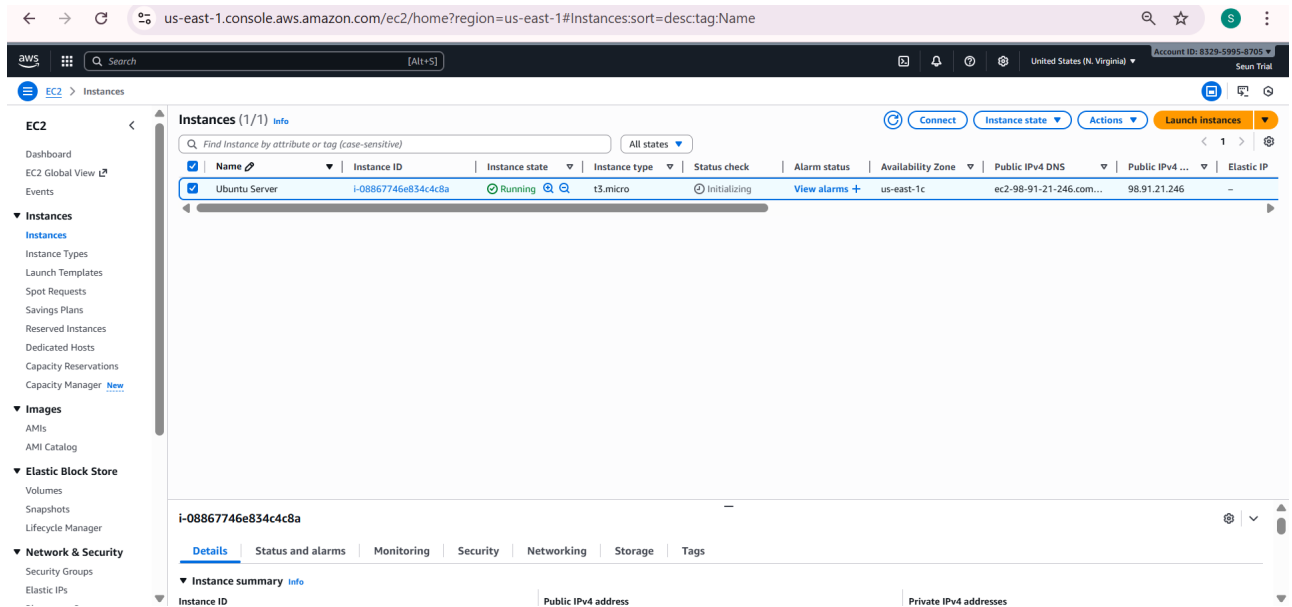
- An active AWS account.
- Basic familiarity with the Linux terminal, SSH, and AWS EC2.
- A local machine with SSH client and SCP support.
- A profile picture (JPG format) to upload for the application demo.

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).



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
$ 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:WBrFV0fyAgxf4Yxyar74CcoQIH7GaehK6C8NrVWS1I.
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
```

```

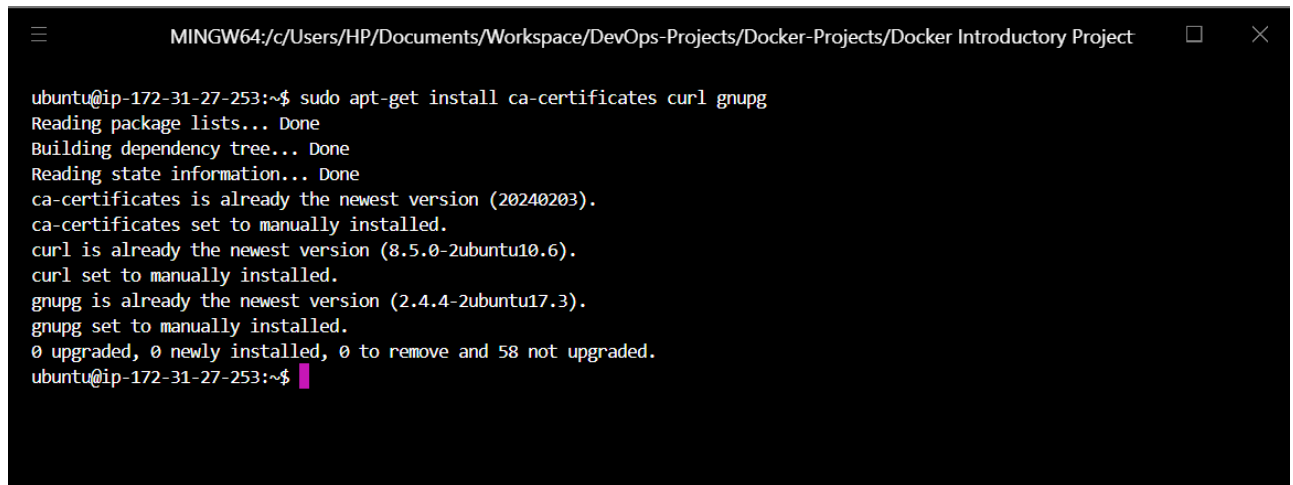
ubuntu@ip-172-31-27-253:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu 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 B]
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:

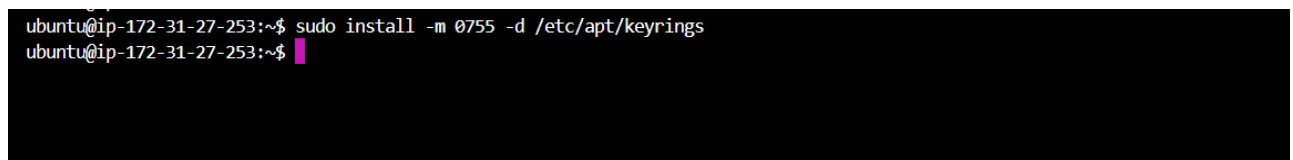
```
sudo apt install ca-certificates curl gnupg lsb-release -y
```

A terminal window titled 'MINGW64:/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project' shows the execution of 'sudo apt-get install ca-certificates curl gnupg'. The output indicates that all three packages are already the newest versions and are set to manually installed. The terminal text is: 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:

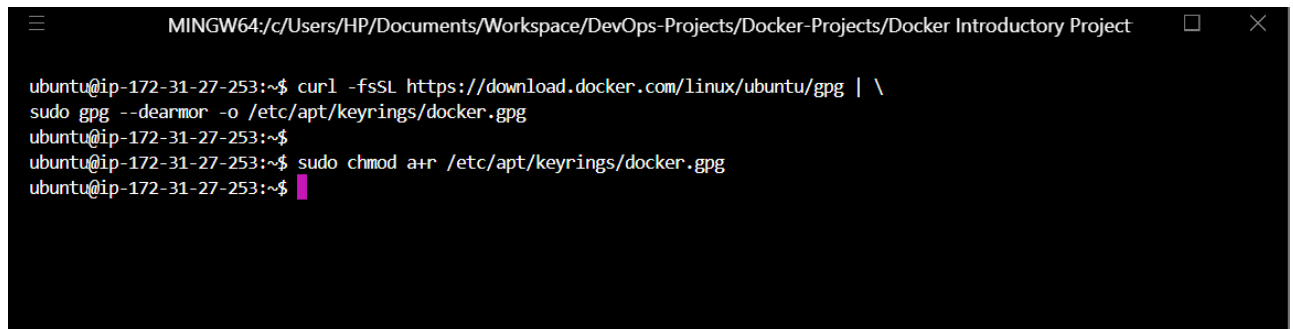
```
sudo mkdir -p /etc/apt/keyrings
```

A terminal window shows the execution of 'sudo install -m 0755 -d /etc/apt/keyrings'. The output is: 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
```

A terminal window shows the execution of 'curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg' followed by 'sudo chmod a+r /etc/apt/keyrings/docker.gpg'. The output is: 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:~\$; 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
```

```

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

ubuntu@ip-172-31-27-253:~$ docker images

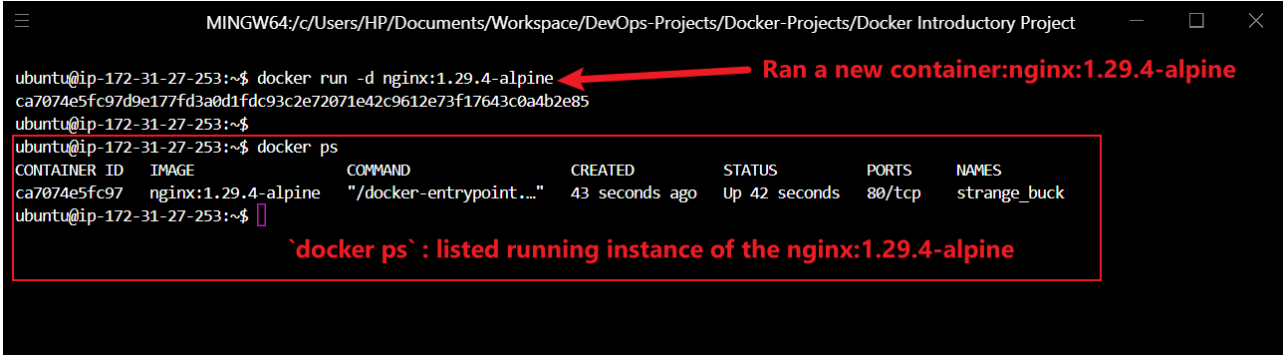
ID                DISK USAGE  CONTENT SIZE  EXTRA  In Use IMAGE
hello-world:latest  d4aaab6242e0  25.9kB       9.52kB  U
ubuntu@ip-172-31-27-253:~$

```

13. List Running Containers (docker ps)

Show currently running containers:

```
docker ps
```



Terminal window showing the command `docker run -d nginx:1.29.4-alpine` and the output of `docker ps`. A red arrow points to the command, and a red box highlights the output of `docker ps`.

```
ubuntu@ip-172-31-27-253:~$ docker run -d nginx:1.29.4-alpine
ca7074e5fc97d9e177fd3a0d1fdc93c2e72071e42c9612e73f17643c0a4b2e85
ubuntu@ip-172-31-27-253:~$ docker ps
```

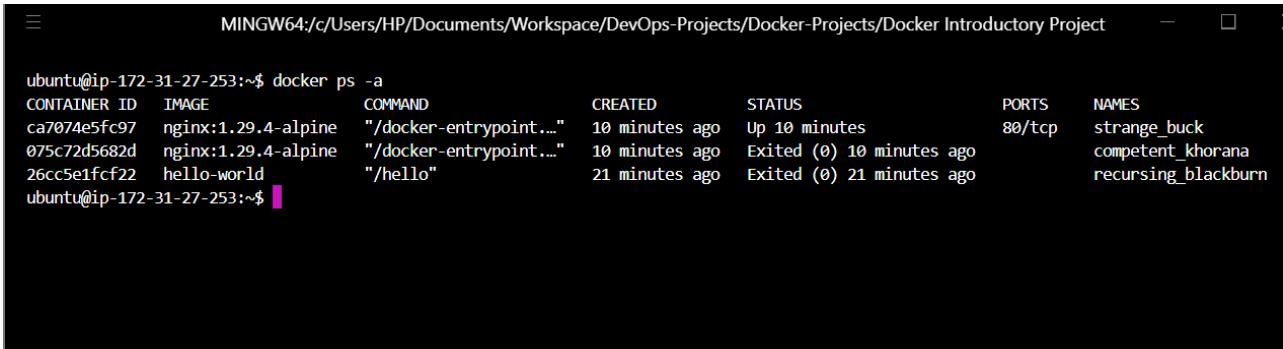
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ca7074e5fc97	nginx:1.29.4-alpine	"/docker-entrypoint..."	43 seconds ago	Up 42 seconds	80/tcp	strange_buck

`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
```



Terminal window showing the output of `docker ps -a`.

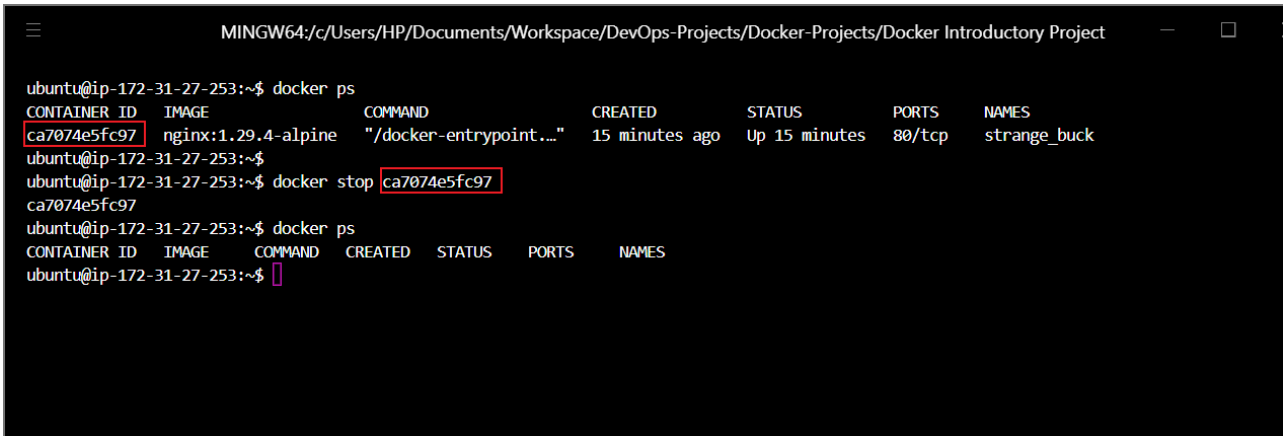
```
ubuntu@ip-172-31-27-253:~$ docker ps -a
```

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		competent_khorana
26cc5e1fcf22	hello-world	"/hello"	21 minutes ago	Exited (0) 21 minutes ago		recurring_blackburn

15. Stop a Container Using Its Container ID

Stop a running container:

```
docker stop <container_id>
```



Terminal window showing the command `docker stop ca7074e5fc97` and the output of `docker ps`.

```
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
--------------	-------	---------	---------	--------	-------	-------

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
06808451f0d6: 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

```

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	

```
ubuntu@ip-172-31-27-253:~$
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

HP@DESKTOP-I9M74R1 MINGW64 ~/Downloads
$ |
```

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.

```

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><input type='text' id='input-name' value='Anna Smith' />
<hr />
<span>Email: </span><input type='email' id='input-email' value='anna.smith@example.com' />
<hr />
<span>Interests: </span><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.

```

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

GNU nano 7.2 server.js *
res.send(userObj);
});
app.get('/get-profile', (req, res) => {
  MongoClient.connect(mongoUrl, { useNewUrlParser: true }, (err, client) => {
    if (err) return res.status(500).send(err);
    let db = client.db(databaseName);
    db.collection('users').findOne({ user_id: 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`

```

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$ 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`

```

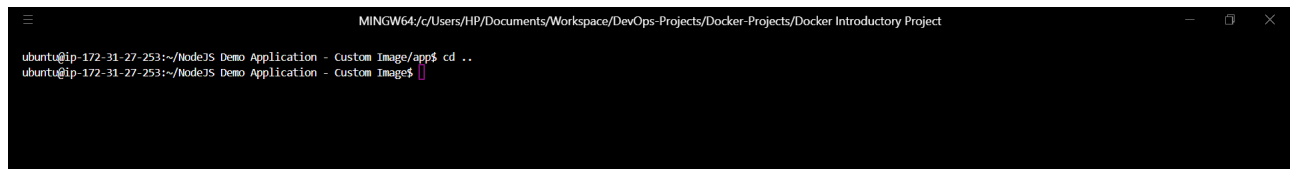
MINGW64: c:/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project

ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$ 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"
  }
}
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app$

```

30. Navigate Back to the Project Folder

```
cd ..
```

A terminal window titled 'MINGW64/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project'. The prompt is 'ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image/app\$'. The user enters 'cd ..' and the prompt changes to '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`).

A terminal window titled 'GNU nano 7.2' with a subtitle '.env *'. The content shows 'MONGO_USER=admin' and 'MONGO_PASSWORD=password' on separate lines.

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

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

A terminal window titled 'GNU nano 7.2' with a subtitle '.gitignore *'. The content shows a list of files and directories to be ignored: '# Node', 'node_modules/', 'npm-debug.log*', 'yarn-debug.log*', 'yarn-error.log*', '# Environment variables', '.env', '.env.*', '# Docker', '*.log', '# OS', '.DS_Store', and 'Thumbs.db'. At the bottom, there is a status bar with various keyboard shortcuts like 'H Help', 'W Write Out', etc.

33. Nano Dockerfile and Add Commands

Write the Dockerfile to build the Node.js image.

```

GNU nano 7.2 Dockerfile *
FROM node:18-alpine

WORKDIR /home/app

COPY app/package*.json ./
RUN npm install

COPY app/ .

EXPOSE 3000

CMD ["node", "server.js"]

File Name to Write: Dockerfile
[Ctrl+H] Help [Ctrl+O] DOS Format [Ctrl+A] Append [Ctrl+B] Backup File
[Ctrl+X] Cancel [Ctrl+M] Mac Format [Ctrl+P] Prepend [Ctrl+N] Browse

```

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
=> => transferring 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@sha256:8d6421d663b4c28fd3ebc498332f249811d1189455880a35cb9bc4b8ca09d9e
=> => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249811d1189455880a35cb9bc4b8ca09d9e
=> => sha256:25ff2da83641908f5c3a74d80409d6b1b62ccfaab220b9ea70b80df5a2e0549 446B / 446B
=> => sha256:dd71dd834b5c203d162902e6b8994cb2309ae049a0eabc4efea161b2b5a3d0e 40.01MB / 40.01MB
=> => sha256:1e54dc89cee5c0826c540ab06d4b6b491c96eda01837f430bd47f0d26702d6e3 1.26MB / 1.26MB
=> => sha256:f18232174bc91741fd3da96d85011092101a032a93a388b79e99e69c2d5c870 3.64MB / 3.64MB
=> => extracting sha256:f18232174bc91741fd3da96d85011092101a032a93a388b79e99e69c2d5c870
=> => extracting sha256:dd71dd834b5c203d162902e6b8994cb2309ae049a0eabc4efea161b2b5a3d0e
=> => extracting sha256:1e54dc89cee5c0826c540ab06d4b6b491c96eda01837f430bd47f0d26702d6e3
=> => extracting sha256:25ff2da83641908f5c3a74d80409d6b1b62ccfaab220b9ea70b80df5a2e0549
=> [internal] load build context
=> => transferring context: 8.91MB
=> [2/5] WORKDIR /home/app
=> [3/5] COPY app/package*.json ./
=> [4/5] RUN npm install
=> [5/5] COPY app/ .
=> exporting to image
=> exporting layers
=> => exporting manifest sha256:f86276f21f437e6947a8b493eb4d2fd15ecc5b875a0f0219d295dfb0fe47fb
=> => exporting config sha256:6a51e99ea4de1f5ede34f2ce2bc271d9d345b57497ccea7b80482ce9e758ad5
=> => exporting attestation manifest sha256:b9c82d9e21c115a91fd1a9e42fa33125d323c6c81a4da88b36ad1f1ccbd5b
=> => exporting manifest list sha256:c835f51741e3ac5141a5301567d1db3616e7c4491176f96678b19291b59114f4
=> => 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`

```

ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker images

```

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
hello-world:latest	d4aaab6242e0	25.9kB	9.52kB	0
nginx:1.29.4-alpine	052b75ab72f6	82MB	23.9MB	0
node-app:1.0	c835f51741e3	219MB	50.8MB	0

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

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

```

MINGW64/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project
GNU nano 7.2 docker-compose.yaml
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      [?] DOS Format  [?] Append      [?] Backup File
[?] Cancel    [?] Mac Format  [?] Prepend     [?] Browse

```

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

`docker compose up -d`

```

MINGW64/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker compose --env-file .env up -d
[*] up 4/4
✔ Network nodejsdemoapplication-customimage default      Created      0.1s
✔ Container nodejsdemoapplication-customimage-mongodb-1 Created      0.1s
✔ Container nodejsdemoapplication-customimage-mongo-express-1 Created      0.1s
✔ Container nodejsdemoapplication-customimage-my-app-1   Created      0.1s
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$

```

38. Docker PS to Check Containers Running

`docker ps`

```

MINGW64/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project
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  nodejsdemoapplication-customimage-mongo-express-1
7ddfb7c710d4   node-app:1.0   "docker-entrypoint.s..." 6 minutes ago Up 4 minutes  0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp  nodejsdemoapplication-customimage-my-app-1
02151f3dd5d    mongo:6        "docker-entrypoint.s..." 4 minutes ago Up 4 minutes  27017/tcp                                nodejsdemoapplication-customimage-mongodb-1
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$

```

39. Docker Logs to Check Node.js Application Logs

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

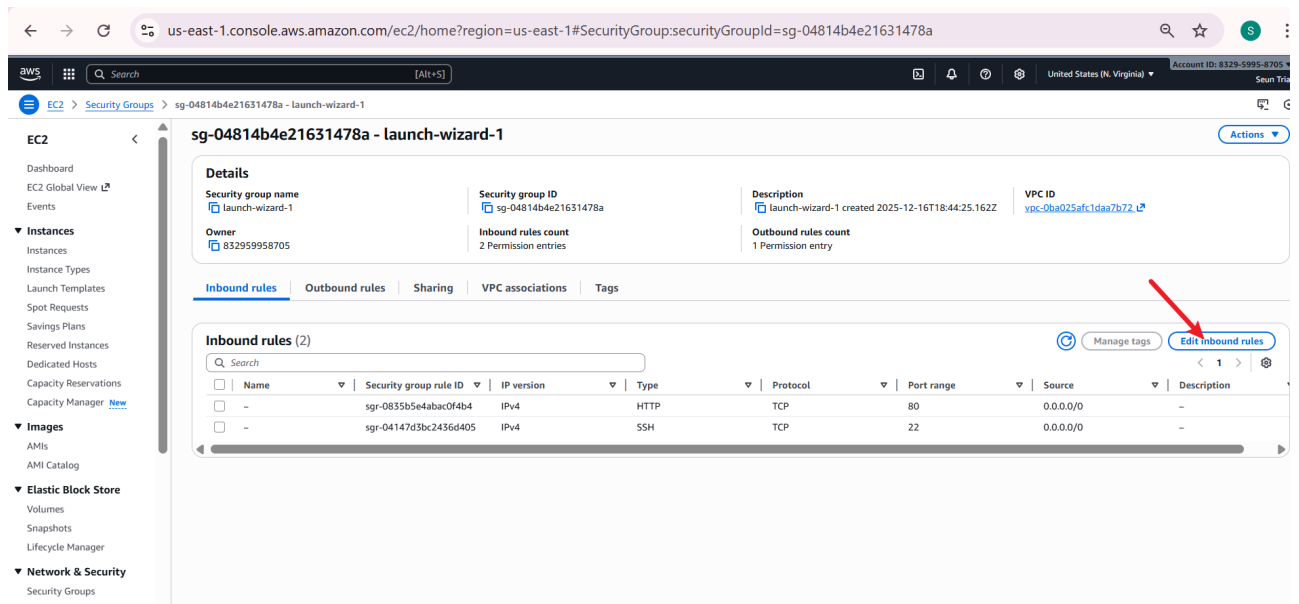
```

MINGW64/c/Users/HP/Documents/Workspace/DevOps-Projects/Docker-Projects/Docker Introductory Project
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  nodejsdemoapplication-customimage-mongo-express-1
7ddfb7c710d4   node-app:1.0   "docker-entrypoint.s..." 6 minutes ago Up 6 minutes  0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp  nodejsdemoapplication-customimage-my-app-1
02151f3dd5d    mongo:6        "docker-entrypoint.s..." 6 minutes ago Up 6 minutes  27017/tcp                                nodejsdemoapplication-customimage-mongodb-1
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$ docker logs nodejsdemoapplication-customimage-my-app-1
Your app listening on port 3000!
ubuntu@ip-172-31-27-253:~/NodeJS Demo Application - Custom Image$

```

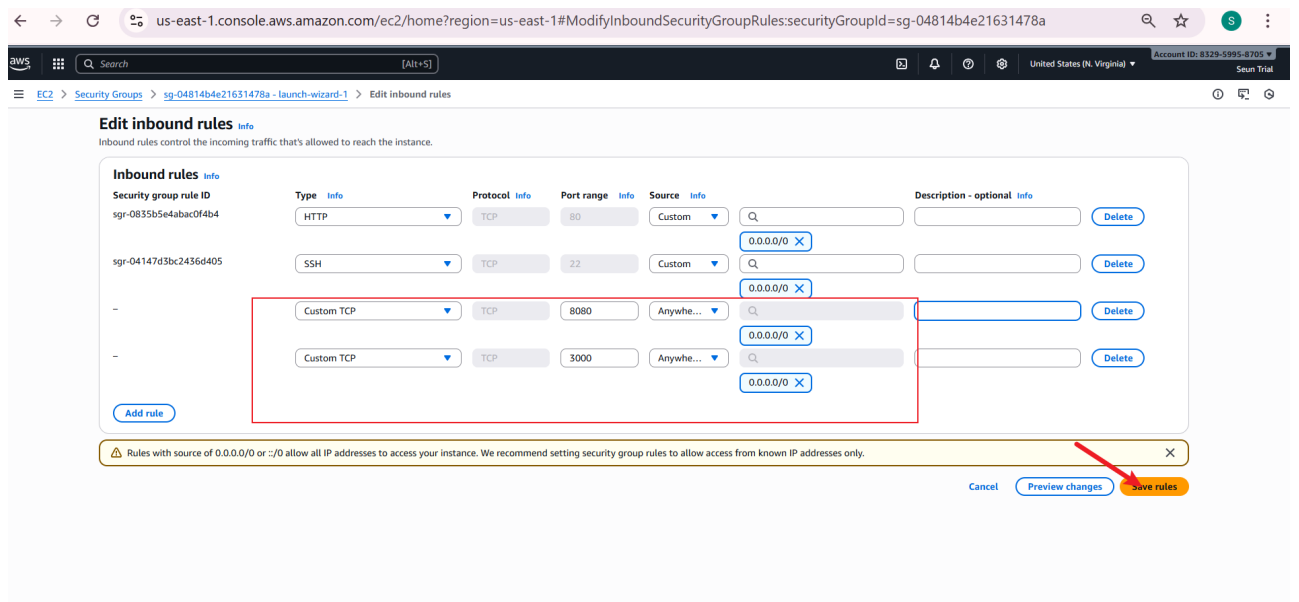
40. Navigate to Server EC2 Instance Security Group to Edit Inbound Rule

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



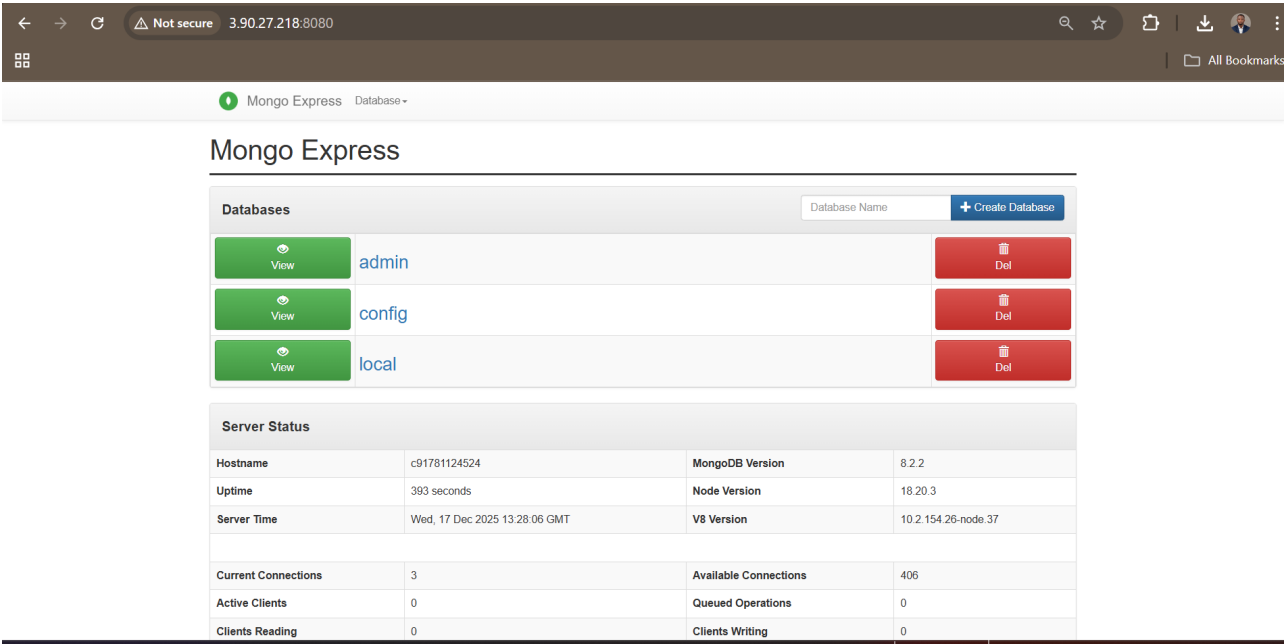
41. Add Two Rules to Accept Custom TCP Connections from Anywhere on Port 3000 and 8080 Then Save Rules

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



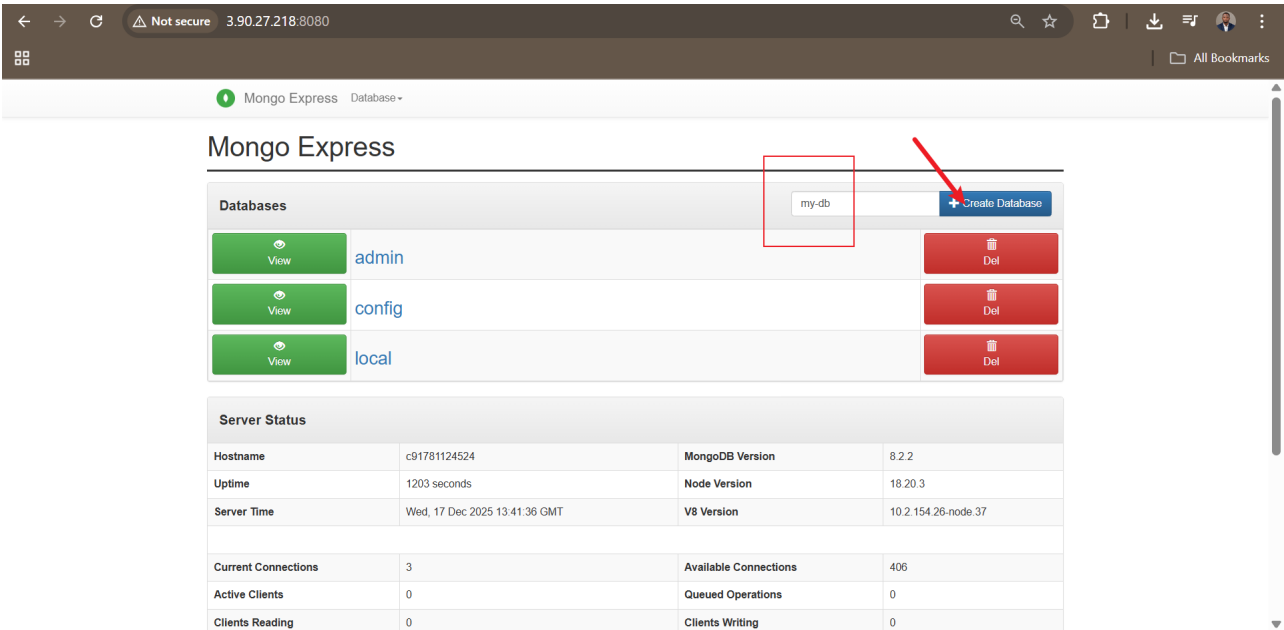
42. Visit the Public IP on Port 8080 to See Mongo Express

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



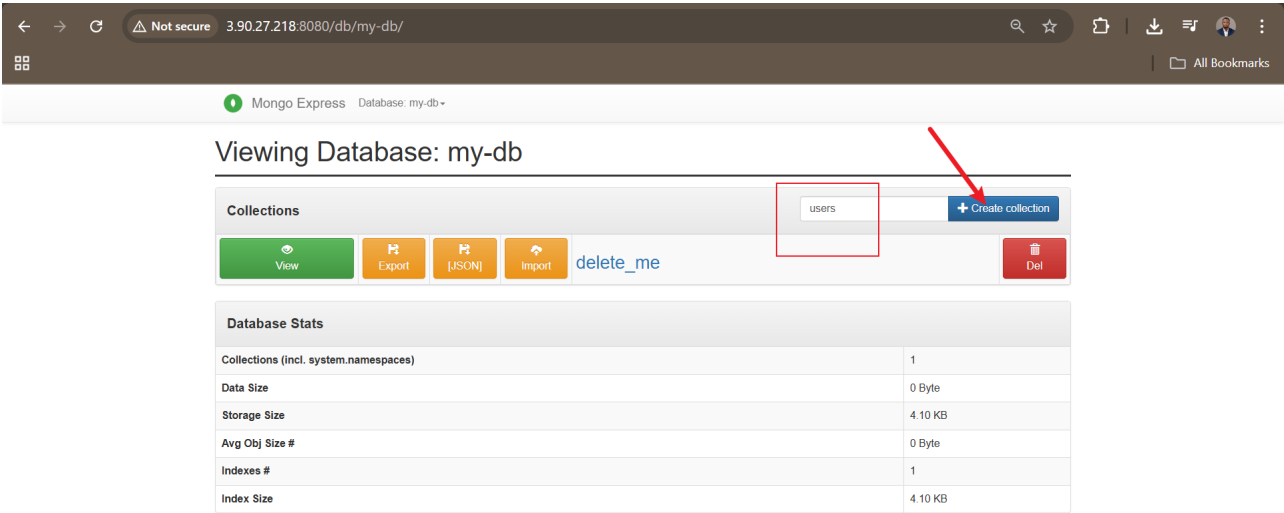
43. Create my-db Database

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



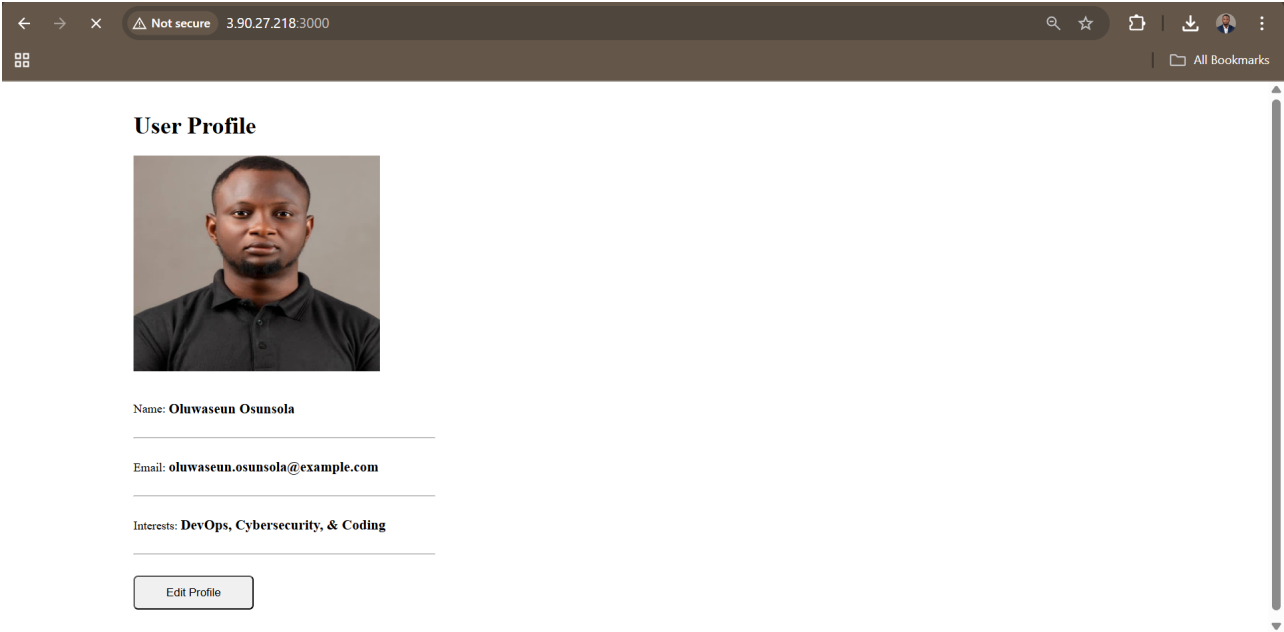
44. Click on my-db and Create Users Collection

Create collection **users**.



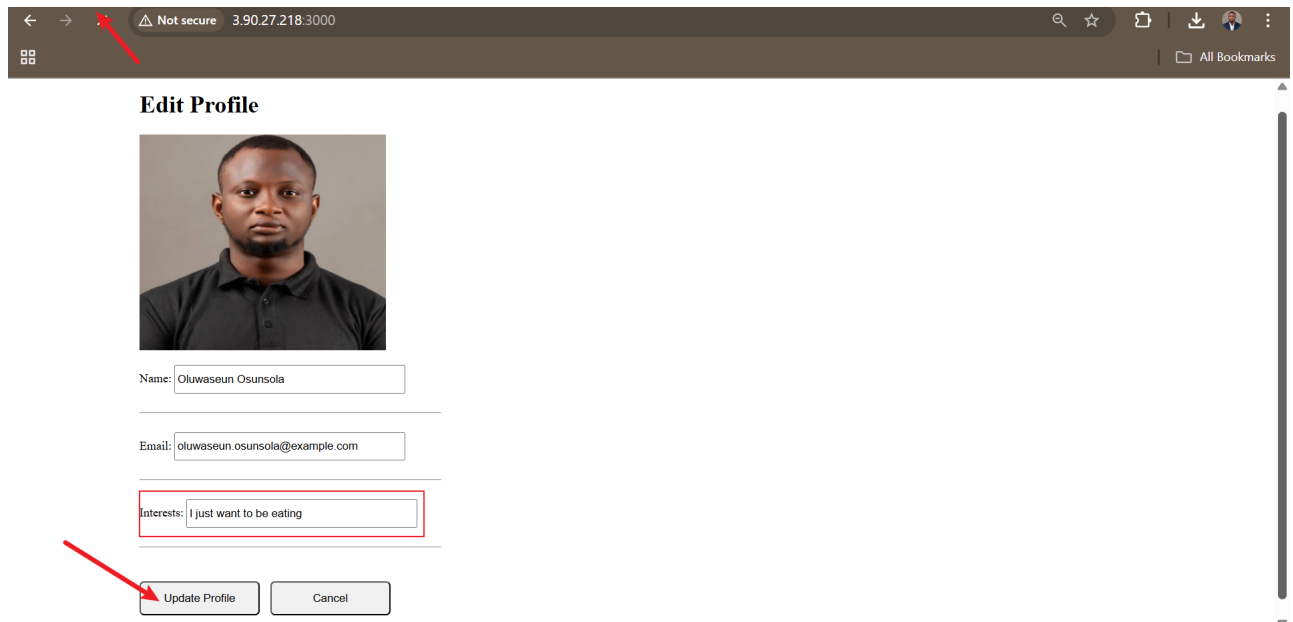
45. Visit the Public IP on Port 3000 to See the Application

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



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.