Docker Containers, Network & Storage

Overview

In this project, we will learn

- ✅ How to Dockerize/Containerize the entire application
- ✅ Usage of Docker Network for inter-container communication
- ✅ Docker Volumes for persistent storage (MySQL data)
- ✅ Integration with Docker Compose to manage the multi-container setup easily

Prerequisites

Knowledge of AWS, Docker, and MySQL

A working AWS account

Summary

✅ **How to Dockerize / Containerize the entire application**

Create a Dockerfile that declares your base image, copies in the application code, installs dependencies, exposes the required port(s), and defines the default command. Build it into an immutable image via docker build, and run an isolated container with docker run to verify the app functions identically across environments.

✅ **Usage of Docker Network for inter-container communication**

Create a user-defined bridge network (e.g., docker network create my-net) and attach each service container to it. Containers on the same network use Docker's internal DNS to communicate using service or container names (e.g., api:8080 → db:3306), eliminating hardcoded IPs and enabling clean service discovery without exposing every internal port externally.

✅ **Docker Volumes for persistent storage (MySQL data)**

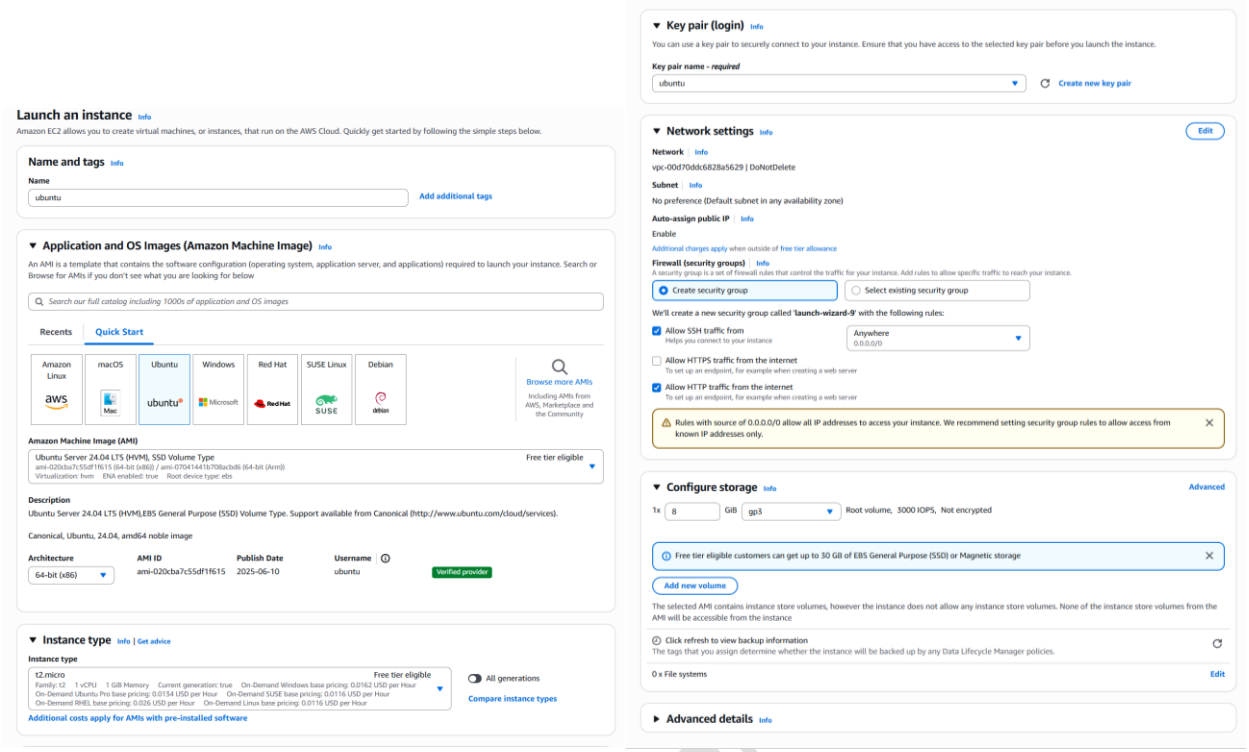
Attach a named Docker volume to the database container's MySQL data directory (e.g. -v mysql_data:/var/lib/mysql). The volume exists independently of container lifecycles—so when the MySQL container is recreated or removed, your data survives and can be re-mounted by a new container instance seamlessly.

✅ **Integration with Docker Compose for easy multi-container management**

Define all services (app, MySQL, Redis, etc.), along with their networks and volumes, in a single docker-compose.yml file. With docker compose up -d --build, Compose builds images, creates shared networks, ensures correct startup order, mounts persistent volumes, and brings up the full stack with one command—making local development and orchestration far simpler.

Implementation Steps

1. Setup of Ubuntu EC2 instance
   Launch EC2 Instance:
   - Use Ubuntu 22.04 LTS AMI.
   - Select an instance type (e.g., t2.micro for testing).
   - Configure security group to allow:
     - Port 80 (HTTP) for frontend access.
     - Port 22 (SSH) for server access.
   - Download the key pair for SSH access.



2. **Connect to EC2 and install Docker**

```
sudo apt update -y
sudo apt install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker ubuntu
newgrp docker
```

exit and reconnect to apply group changes

```
exit
```

```
ubuntu@ip-172-31-21-223:~$ sudo apt update -y
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://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1021 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 [1281 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [260 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [163 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1112 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [284 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [377 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [26.0 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [1572 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [341 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [33.2 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [6772 B]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [592 B]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [39.9 kB]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main Translation-en [9152 B]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7076 B]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [272 B]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [28.0 kB]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [17.1 kB]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [28.3 kB]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1304 B]
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:39 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:40 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [179 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:42 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [876 kB]
Get:43 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [193 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.2 kB]
Get:45 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [17.0 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [1476 kB]
Get:47 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [321 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [18.5 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [4288 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [380 B]
Fetched 35.8 MB in 10s (3483 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
86 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-21-223:~$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zf
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 86 not upgraded.
Need to get 79.2 MB of archives.
After this operation, 300 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.2.5-0ubuntu1-24.04.1 [8043 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.27-0ubuntu1-24.04.1 [37.7
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dns-root-data all 2024071801-ubuntu0.24.04.1 [5
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dnsmasq-base amd64 2.90-2ubuntu0.1 [376 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 27.5.1-0ubuntu3-24.04.2 [33
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 ubuntu-fan all 0.12.16+24.04.1 [34.2 kB]
Fetched 79.2 MB in 1s (74.2 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 70681 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7.1-1ubuntu2_amd64.deb ...
Unpacking bridge-utils (1.7.1-1ubuntu2) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.2.5-0ubuntu1-24.04.1_amd64.deb ...
Unpacking runc (1.2.5-0ubuntu1-24.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.27-0ubuntu1-24.04.1_amd64.deb ...
Unpacking containerd (1.7.27-0ubuntu1-24.04.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2024071801-ubuntu0.24.04.1_all.deb ...
Unpacking dns-root-data (2024071801-ubuntu0.24.04.1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base_2.90-2ubuntu0.1_amd64.deb ...
Unpacking dnsmasq-base (2.90-2ubuntu0.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../6-docker.io_27.5.1-0ubuntu3-24.04.2_amd64.deb ...
Unpacking docker.io (27.5.1-0ubuntu3-24.04.2) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../7-ubuntu-fan_0.12.16+24.04.1_all.deb ...
Unpacking ubuntu-fan (0.12.16+24.04.1) ...
Setting up dnsmasq-base (2.90-2ubuntu0.1) ...
Setting up runc (1.2.5-0ubuntu1-24.04.1) ...
Setting up dns-root-data (2024071801-ubuntu0.24.04.1) ...
Setting up bridge-utils (1.7.1-1ubuntu2) ...
Setting up pigz (2.8-1) ...
Setting up containerd (1.7.27-0ubuntu1-24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16+24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /usr/lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (27.5.1-0ubuntu3-24.04.2) ...
info: Selecting GID from range 100 to 999 ...
info: Adding group `docker' (GID 113) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for dbus (1.14.10-4ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-21-223:~$ sudo systemctl start docker
ubuntu@ip-172-31-21-223:~$ sudo systemctl enable docker
ubuntu@ip-172-31-21-223:~$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-21-223:~$ newgrp docker
ubuntu@ip-172-31-21-223:~$ exit
exit
ubuntu@ip-172-31-21-223:~$ ssh -i ubuntu@.
Warning: Identity file ubuntu@. not accessible: No such file or directory.
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address]
           [-c cipher_spec] [-D [bind_address:]port] [-E log_file]
           [-e escape_char] [-F configfile] [-I pkcs11] [-i identity_file]
           [-J destination] [-L address] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-P tag] [-p port] [-R address]
           [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
           destination [command [argument ...]]
       ssh [-Q query_option]
ubuntu@ip-172-31-21-223:~$ exit
Logout
```

3. **Docker Network**

Create a Docker bridge network to allow communication between the frontend, backend, and MySQL containers.

docker network create jobapp-network

This creates a network named jobapp-network where containers can communicate using their container names as hostnames.

```
Last login: Thu Jul 24 07:24:57 2025 from 18.206.107.27
ubuntu@ip-172-31-21-223:~$ docker network create jobapp-network
12e9abf8ac7dddb86048b9dfa46ccf676d375ae4bfc7b3878285aaf9fbf8fb98
```

4. **Docker Volumes**

Create two Docker volumes:

- mysql-data: For persistent MySQL data.
- uploads: For storing resume PDFs.

docker volume create mysql-data

docker volume create uploads

```
ubuntu@ip-172-31-21-223:~$ docker volume create mysql-data
mysql-data
ubuntu@ip-172-31-21-223:~$ docker volume ls
DRIVER    VOLUME NAME
local     mysql-data
local     uploads
ubuntu@ip-172-31-21-223:~$
```

5. **Create Project Structure**

Create the directory structure on the EC2 instance:

mkdir -p /home/ubuntu/jobapp/client/src

mkdir /home/ubuntu/jobapp/server

mkdir /home/ubuntu/jobapp/uploads

cd /home/ubuntu/jobapp

```
ubuntu@ip-172-31-21-223:~$ mkdir -p /home/ubuntu/jobapp/client/src
ubuntu@ip-172-31-21-223:~$ mkdir /home/ubuntu/jobapp/server
ubuntu@ip-172-31-21-223:~$ mkdir /home/ubuntu/jobapp/uploads
ubuntu@ip-172-31-21-223:~$ cd /home/ubuntu/jobapp
ubuntu@ip-172-31-21-223:~/jobapp$ ls
client  server  uploads
ubuntu@ip-172-31-21-223:~/jobapp$
```

6. **Create and Place Files**

Below are the files to create, with their exact locations and contents. These are reused from the previous response but placed explicitly in the correct directories.

6.1 MySQL Initialization Script

File: /home/ubuntu/jobapp/init.sql

Purpose: Initializes the MySQL database and user.

Create the file:

nano /home/ubuntu/jobapp/init.sql

Paste the content below, save, and exit (Ctrl+O, Enter, Ctrl+X).


```
CREATE DATABASE IF NOT EXISTS jobappdb;

USE jobappdb;


CREATE TABLE IF NOT EXISTS applications (

  id INT AUTO_INCREMENT PRIMARY KEY,

  full_name VARCHAR(255) NOT NULL,

  email VARCHAR(255) NOT NULL,

  position VARCHAR(100) NOT NULL,

  resume_path VARCHAR(255),

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

CREATE USER IF NOT EXISTS 'jobappuser'@'%' IDENTIFIED BY 'securepassword';

GRANT ALL PRIVILEGES ON jobappdb.* TO 'jobappuser'@'%';

FLUSH PRIVILEGES;

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home/ubuntu/jobapp/init.sql
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/init.sql
CREATE DATABASE IF NOT EXISTS jobappdb;
USE jobappdb;

CREATE TABLE IF NOT EXISTS applications (
    id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    position VARCHAR(100) NOT NULL,
    resume_path VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE USER IF NOT EXISTS 'jobappuser'@'%' IDENTIFIED BY 'securepassword';
GRANT ALL PRIVILEGES ON jobappdb.* TO 'jobappuser'@'%';
FLUSH PRIVILEGES;
```

6.2 Backend Dockerfile

File: /home/ubuntu/jobapp/server/Dockerfile

Purpose: Defines the Node.js backend container.

Create the file:

nano /home/ubuntu/jobapp/server/Dockerfile

Paste the below content, save, and exit.

FROM node:18

WORKDIR /app

COPY package.json .

RUN npm install --force --loglevel verbose

COPY . .

EXPOSE 3000

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

```
ubuntu@ip-172-31-21-223:~/jobapp/client$ cat Dockerfile
FROM node:18 AS build
WORKDIR /app
COPY package.json .
RUN npm install --force --loglevel verbose
COPY . .
RUN npm run build
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
ubuntu@ip-172-31-21-223:~/jobapp/client$
```

6.3 Backend Code

File: /home/ubuntu/jobapp/server/server.js

Purpose: Node.js/Express backend with MySQL connection and file upload handling.

Create the file:

nano /home/ubuntu/jobapp/server/server.js

Paste the content given below, save, and exit.


```
const express = require('express');

const mysql = require('mysql2/promise');

const cors = require('cors');

const multer = require('multer');

const path = require('path');

const { body, validationResult } = require('express-validator');

const fs = require('fs');


const app = express();
```

```
app.use(cors());
app.use(express.json());
app.use('/uploads', express.static('/uploads')); // Use shared volume

// MySQL connection
const pool = mysql.createPool({
  host: process.env.DB_HOST || 'mysql',
  user: process.env.DB_USER || 'jobappuser',
  password: process.env.DB_PASSWORD || 'securepassword',
  database: process.env.DB_NAME || 'jobappdb',
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
});

// Multer setup for file uploads
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, '/uploads/');
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + path.extname(file.originalname));
  },
});
const upload = multer({
  storage,
```

```
  fileFilter: (req, file, cb) => {
    const filetypes = /pdf/;
    const extname = filetypes.test(path.extname(file.originalname).toLowerCase());
    if (extname) {
      return cb(null, true);
    }
    cb(new Error('Only PDF files are allowed'));
  },
});

// Create uploads directory
if (!fs.existsSync('/uploads')) {
  fs.mkdirSync('/uploads');
}

// Form submission endpoint
app.post(
  '/api/apply',
  upload.single('resume'),
  [
    body('fullName').trim().notEmpty().withMessage('Full name is required'),
    body('email').isEmail().withMessage('Valid email is required'),
    body('position').trim().notEmpty().withMessage('Position is required'),
  ],
  async (req, res) => {
    const errors = validationResult(req);
```

```
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    const { fullName, email, position } = req.body;
    const resumePath = req.file ? req.file.path : null;

    try {
      const [result] = await pool.query(
        'INSERT INTO applications (full_name, email, position, resume_path) VALUES (?, ?, ?, ?)',
        [fullName, email, position, resumePath]
      );
      res.json({ message: 'Application submitted successfully', id: result.insertId });
    } catch (err) {
      console.error(err);
      res.status(500).json({ error: 'Database error' });
    }
  }
);

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/server/server.js
const express = require('express');
const mysql = require('mysql2/promise');
const cors = require('cors');
const multer = require('multer');
const path = require('path');
const { body, validationResult } = require('express-validator');
const fs = require('fs');

const app = express();

app.use(cors());
app.use(express.json());
app.use('/uploads', express.static('/uploads')); // Use shared volume

// MySQL connection
const pool = mysql.createPool({
  host: process.env.DB_HOST || 'mysql',
  user: process.env.DB_USER || 'jobappuser',
  password: process.env.DB_PASSWORD || 'securepassword',
  database: process.env.DB_NAME || 'jobappdb',
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
});

// Multer setup for file uploads
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, '/uploads/');
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + path.extname(file.originalname));
  },
});
const upload = multer({
  storage,
  fileFilter: (req, file, cb) => {
    const filetypes = /pdf/;
    const extname = filetypes.test(path.extname(file.originalname).toLowerCase());
    if (extname) {
      return cb(null, true);
    }
    cb(new Error('Only PDF files are allowed'));
  },
});

// Create uploads directory
if (!fs.existsSync('/uploads')) {
  fs.mkdirSync('/uploads');
}

// Form submission endpoint
app.post(
  '/api/apply',
  upload.single('resume'),
  [
    body('fullName').trim().notEmpty().withMessage('Full name is required'),
    body('email').isEmail().withMessage('Valid email is required'),
    body('position').trim().notEmpty().withMessage('Position is required'),
  ],
  async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    const { fullName, email, position } = req.body;
    const resumePath = req.file ? req.file.path : null;

    try {
      const [result] = await pool.query(
        'INSERT INTO applications (full_name, email, position, resume_path) VALUES (?, ?,
        [fullName, email, position, resumePath]
      );
      res.json({ message: 'Application submitted successfully', id: result.insertId });
    } catch (err) {
      console.error(err);
      res.status(500).json({ error: 'Database error' });
    }
  }
);

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
ubuntu@ip-172-31-21-223:~/jobapp$
```

6.4 Backend Package.json

File: /home/ubuntu/jobapp/server/package.json

Purpose: Defines backend dependencies.

Create the file:

nano /home/ubuntu/jobapp/server/package.json

Paste the content below, save, and exit.

```
{
  "name": "jobapp-backend",
  "version": "1.0.0",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "express-validator": "^7.0.1",
    "multer": "^1.4.5-lts.1",
    "mysql2": "^3.6.0"
  }
}
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home//ubuntu/jobapp/server/package.json
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/server/package.json
{
  "name": "jobapp-backend",
  "version": "1.0.0",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "express-validator": "^7.0.1",
    "multer": "^1.4.5-lts.1",
    "mysql2": "^3.6.0"
  }
}
ubuntu@ip-172-31-21-223:~/jobapp$
```

6.5 Frontend Dockerfile

File: /home/ubuntu/jobapp/client/Dockerfile

Purpose: Builds and serves the React app with Nginx.

Create the file:

nano /home/ubuntu/jobapp/client/Dockerfile

Paste the below content, save, and exit.

```
FROM node:18 AS build

WORKDIR /app

COPY package.json .

RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=build /app/build /usr/share/nginx/html

COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home/ubuntu/jobapp/client/Dockerfile
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/client/Dockerfile
FROM node:18 AS build
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
RUN npm run build
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
ubuntu@ip-172-31-21-223:~/jobapp$
```

6.6 Nginx Configuration

File: /home/ubuntu/jobapp/client/nginx.conf

Purpose: Configures Nginx to serve the React app and proxy API requests.

Create the file:

nano /home/ubuntu/jobapp/client/nginx.conf

Paste the below content, save, and exit.

```
server {
  listen 80;
  server_name localhost;

  root /usr/share/nginx/html;
  index index.html;

  location / {
    try_files $uri /index.html;
  }

  location /api/ {
```

```
        proxy_pass http://backend:3000/api/;

        proxy_http_version 1.1;

        proxy_set_header Host $host;

        proxy_set_header X-Real-IP $remote_addr;

        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        proxy_set_header X-Forwarded-Proto $scheme;

    }

}
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home/ubuntu/jobapp/client/nginx.conf
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/client/nginx.conf
server {
    listen 80;
    server_name localhost;

    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri /index.html;
    }

    location /api/ {
        proxy_pass http://backend:3000/api/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
ubuntu@ip-172-31-21-223:~/jobapp$
```

File: client/src/App.js

Purpose: React frontend for the job application form.

Create: nano /home/ubuntu/jobapp/client/src/App.js

Paste the below content, save, and exit.

```
import React, { useState } from 'react';
```

```
import axios from 'axios';
import './App.css';

function App() {
  const [formData, setFormData] = useState({
    fullName: '',
    email: '',
    position: '',
  });
  const [resume, setResume] = useState(null);
  const [message, setMessage] = useState('');
  const [errors, setErrors] = useState([]);

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleFileChange = (e) => {
    setResume(e.target.files[0]);
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setErrors([]);
    setMessage('');
```

```
  const data = new FormData();
  data.append('fullName', formData.fullName);
  data.append('email', formData.email);
  data.append('position', formData.position);
  if (resume) data.append('resume', resume);

  try {
    const response = await axios.post('/api/apply', data);
    setMessage(response.data.message);
    setFormData({ fullName: '', email: '', position: '' });
    setResume(null);
  } catch (error) {
    if (error.response && error.response.data.errors) {
      setErrors(error.response.data.errors);
    } else {
      setMessage('Error submitting application');
    }
  }
};

return (
  <div className="App">
    <h1>Job Application Form</h1>
    <form onSubmit={handleSubmit}>
      <div>
        <label>Full Name:</label>
```

```
      <input
        type="text"
        name="fullName"
        value={formData.fullName}
        onChange={handleChange}
        required
      />
    </div>
    <div>
      <label>Email:</label>
      <input
        type="email"
        name="email"
        value={formData.email}
        onChange={handleChange}
        required
      />
    </div>
    <div>
      <label>Position:</label>
      <select name="position" value={formData.position} onChange={handleChange}
required>
        <option value="">Select a position</option>
        <option value="Developer">Developer</option>
        <option value="Designer">Designer</option>
        <option value="Manager">Manager</option>
```

```
        </select>
      </div>
      <div>
        <label>Resume (PDF only):</label>
        <input type="file" accept=".pdf" onChange={handleFileChange} />
      </div>
      <button type="submit">Submit Application</button>
    </form>
    {message && <p className="success">{message}</p>}
    {errors.length > 0 && (
      <ul className="errors">
        {errors.map((error, index) => (
          <li key={index}>{error.msg}</li>
        ))}
      </ul>
    )}
  </div>
);
}

export default App;
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home/ubuntu/jobapp/client/src/App.js
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/client/src/App.js
import React, { useState } from 'react';
import axios from 'axios';
import './App.css';

function App() {
  const [formData, setFormData] = useState({
    fullName: '',
    email: '',
    position: '',
  });
  const [resume, setResume] = useState(null);
  const [message, setMessage] = useState('');
  const [errors, setErrors] = useState([]);

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleFileChange = (e) => {
    setResume(e.target.files[0]);
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setErrors([]);
    setMessage('');

    const data = new FormData();
    data.append('fullName', formData.fullName);
    data.append('email', formData.email);
    data.append('position', formData.position);
    if (resume) data.append('resume', resume);

    try {
      const response = await axios.post('/api/apply', data);
      setMessage(response.data.message);
      setFormData({ fullName: '', email: '', position: '' });
      setResume(null);
    } catch (error) {
      if (error.response && error.response.data.errors) {
        setErrors(error.response.data.errors);
      } else {
        setMessage('Error submitting application');
      }
    }
  };

  return (
    <div className="App">
      <h1>Job Application Form</h1>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Full Name:</label>
          <input
            type="text"
            name="fullName"
            value={formData.fullName}
            onChange={handleChange}
            required
          />
        </div>
        <div>
          <label>Email:</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            required
          />
        </div>
        <div>
          <label>Position:</label>
          <select name="position" value={formData.position} onChange={handleChange} required>
            <option value="">Select a position</option>
            <option value="Developer">Developer</option>
            <option value="Designer">Designer</option>
            <option value="Manager">Manager</option>
          </select>
        </div>
        <div>
          <label>Resume (PDF only):</label>
          <input type="file" accept=".pdf" onChange={handleFileChange} />
        </div>
        <button type="submit">Submit Application</button>
      </form>
      {message && <p className="success">{message}</p>}
      {errors.length > 0 && (
        <ul className="errors">
          {errors.map((error, index) => (
            <li key={index}>{error.msg}</li>
          ))}
        </ul>
      )}
    </div>
  );
}

export default App;
ubuntu@ip-172-31-21-223:~/jobapp$
```

File: client/src/App.css

Purpose: Basic styling for the React app.

Create: nano /home/ubuntu/jobapp/client/src/App.css

Paste the content, save, and exit.

```
.App {
  text-align: center;
  max-width: 600px;
  margin: 0 auto;
  padding: 20px;
}

h1 {
  color: #333;
}

form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

div {
  display: flex;
  flex-direction: column;
  text-align: left;
```

```
}

label {
  font-weight: bold;
  margin-bottom: 5px;
}

input, select {
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

button {
  padding: 10px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 16px;
}

button:hover {
  background-color: #0056b3;
```

```
}

.success {
  color: green;
  margin-top: 20px;
}

.errors {
  color: red;
  margin-top: 20px;
  list-style: none;
  padding: 0;
}

.errors li {
  margin-bottom: 5px;
}
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home/ubuntu/jobapp/client/src/App.css
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/client/src/App.css
.App {
  text-align: center;
  max-width: 600px;
  margin: 0 auto;
  padding: 20px;
}

h1 {
  color: #333;
}

form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

div {
  display: flex;
  flex-direction: column;
  text-align: left;
}

label {
  font-weight: bold;
  margin-bottom: 5px;
}

input, select {
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

button {
  padding: 10px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 16px;
}

button:hover {
  background-color: #0056b3;
}

.success {
  color: green;
  margin-top: 20px;
}

.errors {
  color: red;
  margin-top: 20px;
  list-style: none;
  padding: 0;
}

.errors li {
  margin-bottom: 5px;
}
ubuntu@ip-172-31-21-223:~/jobapp$
```

File: client/package.json

Purpose: Frontend dependencies.

Create: nano /home/ubuntu/jobapp/client/package.json

Paste the content, save, and exit.

```
{
  "name": "jobapp-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "axios": "^1.6.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  }
}
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home/ubuntu/jobapp/client/package.json
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/client/package.json
{
  "name": "jobapp-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "axios": "^1.6.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  }
}
ubuntu@ip-172-31-21-223:~/jobapp$
```

Create the missing file: nano /home/ubuntu/jobapp/client/src/index.js

```
import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';


const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ nano /home/ubuntu/jobapp/client/src/index.js
ubuntu@ip-172-31-21-223:~/jobapp$ cat /home/ubuntu/jobapp/client/src/index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
ubuntu@ip-172-31-21-223:~/jobapp$
```

Create the missing file: nano /home/ubuntu/jobapp/client/public/index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

```
ubuntu@ip-172-31-21-223:~/jobapp$ mkdir /home/ubuntu/jobapp/client/public
ubuntu@ip-172-31-21-223:~/jobapp$ ls
client  init.sql  nano  server  uploads
ubuntu@ip-172-31-21-223:~/jobapp$ cd client/
ubuntu@ip-172-31-21-223:~/jobapp/client$ ls
Dockerfile  nginx.conf  package.json  public  src
ubuntu@ip-172-31-21-223:~/jobapp/client$ nano /home/ubuntu/jobapp/client/public/index.html
ubuntu@ip-172-31-21-223:~/jobapp/client$ cat /home/ubuntu/jobapp/client/public/index.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
ubuntu@ip-172-31-21-223:~/jobapp/client$
```

7. **Build and Run Containers**

Run containers in order (MySQL first, then backend, then frontend) to ensure dependencies are available.

- **MySQL Container**

```
docker run -d \
  --name mysql \
  --network jobapp-network \
  -v mysql-data:/var/lib/mysql \
```

**-v $(pwd)/init.sql:/docker-entrypoint-initdb.d/init.sql **

**-e MYSQL_ROOT_PASSWORD=rootpassword **

mysql:8.0

```
ubuntu@ip-172-31-21-223:~/jobapp/client$ cd ..
ubuntu@ip-172-31-21-223:~/jobapp$ docker run -d \
  --name mysql \
  --network jobapp-network \
  -v mysql-data:/var/lib/mysql \
  -v $(pwd)/init.sql:/docker-entrypoint-initdb.d/init.sql \
  -e MYSQL_ROOT_PASSWORD=rootpassword \
  mysql:8.0
Unable to find image 'mysql:8.0' locally
8.0: Pulling from library/mysql
62efe2b176c9: Pull complete
6e888f9be8fb: Pull complete
d9a8317b68b7: Pull complete
0fdcc4844af8: Pull complete
e43fd8b16f03: Pull complete
b35825d0dd06: Pull complete
257616cdb019: Pull complete
3f032aaf5f93: Pull complete
33039f492348: Pull complete
fc9aa6dc71fb: Pull complete
e863b5308434: Pull complete
Digest: sha256:ccf4fed7ff4b886aeb3573a1f5d5b509525ecff55a2d1e2653c27a5abdded309
Status: Downloaded newer image for mysql:8.0
b61e9bbef96fe0323b972dde4d969d3e4f77ca61cc0e4722547d899faa5e0cc4
ubuntu@ip-172-31-21-223:~/jobapp$
```

Line-by-line Explanation

Part	Explanation
docker run -d	Runs the container in detached mode (in background).
--name mysql	Names the container mysql (useful for reference in other containers).
--network jobapp-network	Connects the container to a custom Docker network named jobapp-network (useful for multi-container setups like backend ↔ DB).
-v mysql-data:/var/lib/mysql	Mounts a named volume (mysql-data) to store MySQL data persistently.

Part	Explanation
-v $(pwd)/init.sql:/docker-entrypoint-initdb.d/init.sql	Mounts a SQL file from your current directory into the container. MySQL will automatically execute this file only on first-time DB initialization.
-e MYSQL_ROOT_PASSWORD=rootpassword	Sets the root password for MySQL (required).
mysql:8.0	Pulls and runs the official MySQL image (version 8.0).

- **Backend Container**

Build the image:

cd /home/ubuntu/jobapp/server

docker build -t jobapp-backend .

```
ubuntu@ip-172-31-21-223:~/jobapp$ cd server/
ubuntu@ip-172-31-21-223:~/jobapp/server$ docker build -t jobapp-backend .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  6.144kB
Step 1/7 : FROM node:18
18: Pulling from library/node
3e6b9d1a9511: Pull complete
37927ed901b1: Pull complete
79b2f47ad444: Pull complete
e23f099911d6: Pull complete
cda7f44f2bdd: Pull complete
c6b30c3f1696: Pull complete
3697be50c98b: Pull complete
461077a72fb7: Pull complete
Digest: sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
Status: Downloaded newer image for node:18
 ---> b50082bc3670
Step 2/7 : WORKDIR /app
 ---> Running in 348eadf93457
 ---> Removed intermediate container 348eadf93457
 ---> 77988fa9070d
Step 3/7 : COPY package.json .
 ---> a5251c74a169
Step 4/7 : RUN npm install
 ---> Running in 1c988e51f931
npm warn deprecated multer@1.4.5-lts.2: Multer 1.x is impacted by a number of vulnerabilities, which have been

added 104 packages, and audited 105 packages in 9s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.4.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.4.2
npm notice To update run: npm install -g npm@11.4.2
npm notice
 ---> Removed intermediate container 1c988e51f931
 ---> 125c9c7fa4f4
Step 5/7 : COPY . .
 ---> d784f36075ad
Step 6/7 : EXPOSE 3000
 ---> Running in 15195786dd0f
 ---> Removed intermediate container 15195786dd0f
 ---> 5a0e5bc47b95
Step 7/7 : CMD ["node", "server.js"]
 ---> Running in 1b26cd4e03f7
 ---> Removed intermediate container 1b26cd4e03f7
 ---> 007b6db97b35
Successfully built 007b6db97b35
Successfully tagged jobapp-backend:latest
ubuntu@ip-172-31-21-223:~/jobapp/server$
```

Run the container:

```
docker run -d \
  --name backend \
  --network jobapp-network \
  -v uploads:/uploads \
  -e DB_HOST=mysql \
  -e DB_USER=jobappuser \
```

**-e DB_PASSWORD=securepassword **

**-e DB_NAME=jobappdb **

jobapp-backend

```
ubuntu@ip-172-31-21-223:~/jobapp/server$ docker run -d \
  --name backend \
  --network jobapp-network \
  -v uploads:/uploads \
  -e DB_HOST=mysql \
  -e DB_USER=jobappuser \
  -e DB_PASSWORD=securepassword \
  -e DB_NAME=jobappdb \
  jobapp-backend
c39b6317a8c3ca5ba621942d5f9e5d79f4dcb1e2e11ec5f160c54c0d7731eb43
ubuntu@ip-172-31-21-223:~/jobapp/server$
```

Line-by-Line Explanation

Option	Description
-d	Run the container in detached mode (in the background).
--name backend	Name the container backend for easy reference and linking.
--network jobapp-network	Connects this container to a custom Docker network, so it can reach mysql by hostname (i.e., DB_HOST=mysql).
-v uploads:/uploads	Mounts a named volume (uploads) into /uploads inside the container. Useful for persistent file uploads (like resumes, images, etc.).
-e DB_HOST=mysql	Environment variable passed to the app. Tells it the database hostname is mysql (another container on the same network).
-e DB_USER=jobappuser	MySQL username used by the backend app to connect.
-e DB_PASSWORD=securepassword	Password for that MySQL user.
-e DB_NAME=jobappdb	The name of the MySQL database your app will connect to.
jobapp-backend	The image name of your backend app. You must build it with docker build -t jobapp-backend . first.

📦 **How It Works Together**

- **Docker Network:** jobapp-network allows backend to talk to mysql container directly using hostname mysql.
- **Volumes:** The uploads volume helps store user-uploaded files persistently across container restarts.
- **Environment Variables:** These are passed into your app (e.g., a Node.js Express backend) so it knows how to connect to MySQL.

✅ **Things to Ensure Before Running:**

1. **mysql container is up and running** with:
    - A database named jobappdb
    - A user jobappuser with access to that DB
2. **Your backend app** handles these environment variables (using process.env.DB_HOST, etc. in Node.js).
3. **Docker image jobapp-backend** exists.

7.3 Frontend Container:

Build the image:

cd /home/ubuntu/jobapp/client

docker build -t jobapp-frontend .

```
alpine: Pulling from library/nginx
9824c27679d3: Pull complete
a5585638209e: Pull complete
fd372c3c84a2: Pull complete
958a74d6a238: Pull complete
c1d2dc189e38: Pull complete
828fa206d77b: Pull complete
bdaad27fd04a: Pull complete
f23865b38cc6: Pull complete
Digest: sha256:d67ea0d64d518b1bb04acde3b00f722ac3e9764b3209a9b0a98924ba35e4b779
Status: Downloaded newer image for nginx:alpine
 ---> d6adbc7fd47e
Step 8/11 : COPY --from=build /app/build /usr/share/nginx/html
 ---> 59bc5c7443d1
Step 9/11 : COPY nginx.conf /etc/nginx/conf.d/default.conf
 ---> 7ad9806968a5
Step 10/11 : EXPOSE 80
 ---> Running in 83e431d531a2
 ---> Removed intermediate container 83e431d531a2
 ---> 46cfd5ca9545
Step 11/11 : CMD ["nginx", "-g", "daemon off;"]
 ---> Running in 63a6b8625754
 ---> Removed intermediate container 63a6b8625754
 ---> c88eff8d5ec0
Successfully built c88eff8d5ec0
Successfully tagged jobapp-frontend:latest
```

Run the container:

**docker run -d **

**--name frontend **

**--network jobapp-network **

**-p 80:80 **

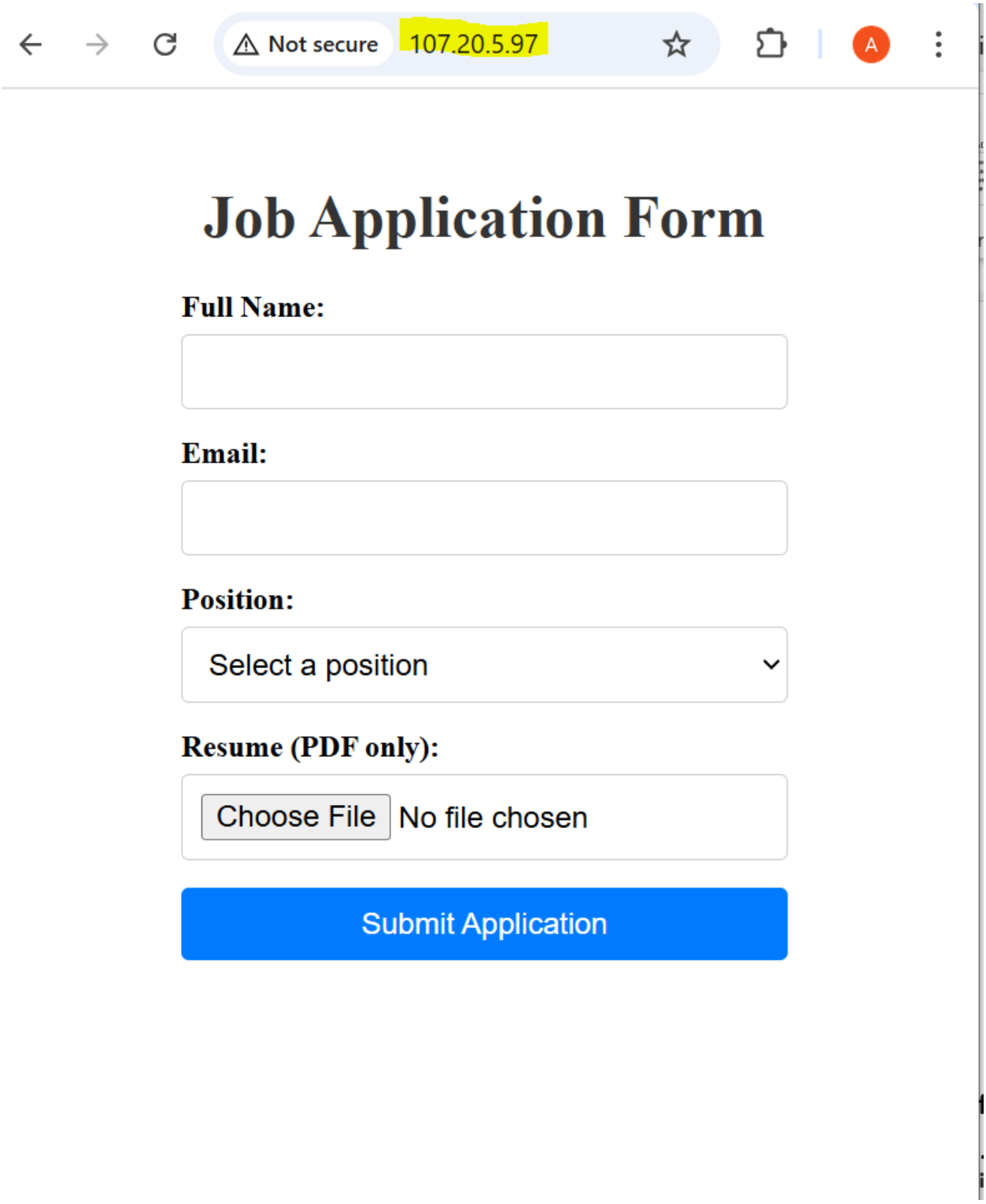
jobapp-frontend

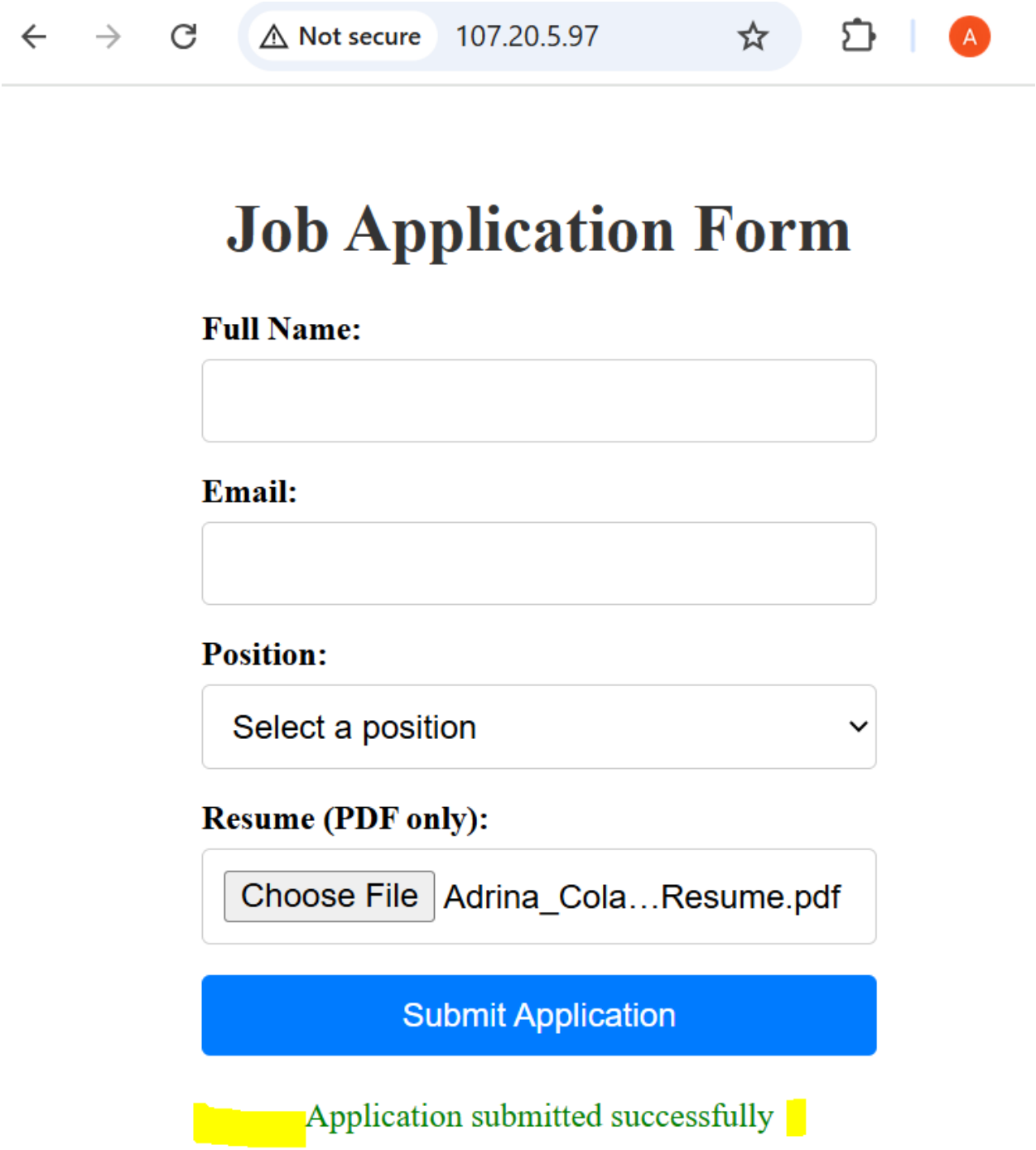

```
ubuntu@ip-172-31-21-223:~/jobapp/client$ docker run -d \
  --name frontend \
  --network jobapp-network \
  -p 80:80 \
  jobapp-frontend
4117bec36cf1750a5d5ec5b49cb22ad00e628180b93082613ed1ac5fb9acf5a0
ubuntu@ip-172-31-21-223:~/jobapp/client$
```

8. **Access the Application**

- Open a browser and navigate to **http://<EC2-Public-IP>** to access the React frontend.
- The frontend sends API requests to /api/apply, which Nginx proxies to the backend container (backend:3000).

- The backend stores form data in the MySQL container (mysql) and saves resumes to the uploads volume.





9. **Verify Containers**

Check that all containers are running:

docker ps

Expected output shows three containers: mysql, backend, and frontend.

```
ubuntu@ip-172-31-21-223:~/jobapp/client$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED              STATUS              PORTS                                      NAMES
922709d0b514   jobapp-frontend    "/docker-entrypoint.…"   36 seconds ago       Up 36 seconds       0.0.0.0:80->80/tcp, :::80->80/tcp          frontend
d13077acf33e   jobapp-backend     "docker-entrypoint.s…"   About a minute ago   Up About a minute   3000/tcp                                   backend
5aeafc26d421   mysql:8.0          "docker-entrypoint.s…"   3 minutes ago        Up 3 minutes        3306/tcp, 33060/tcp                        mysql
ubuntu@ip-172-31-21-223:~/jobapp/client$
```

- **Check data in mysql container:**

Open a shell in the MySQL container:

docker exec -it mysql bash

Step 2: Log into MySQL

1. **Connect to the MySQL server using the jobappuser credentials defined in your init.sql:**

mysql -u jobappuser -psecurepassword

```
ubuntu@ip-172-31-21-223:~/jobapp/client$ docker exec -it mysql bash
bash-5.1# mysql -u jobappuser -psecurepassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.43 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Select the jobappdb database:

USE jobappdb;

Step 3: Query the applications Table

1. **Check the data in the applications table:**

SELECT * FROM applications;

```
mysql> USE jobappdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM applications;
+----+---------------+---------------------+----------+---------------------------+---------------------+
| id | full_name     | email               | position | resume_path               | created_at          |
+----+---------------+---------------------+----------+---------------------------+---------------------+
|  1 | Adrina Colaco | Test_Adri@gmail.com | Designer | /uploads/1753378901623.pdf | 2025-07-24 17:41:41 |
+----+---------------+---------------------+----------+---------------------------+---------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM applications;
+----+---------------+---------------------+-----------+---------------------------+---------------------+
| id | full_name     | email               | position  | resume_path               | created_at          |
+----+---------------+---------------------+-----------+---------------------------+---------------------+
|  1 | Adrina Colaco | Test_Adri@gmail.com | Designer  | /uploads/1753378901623.pdf | 2025-07-24 17:41:41 |
|  2 | Test user     | dnd@yahoo.com       | Developer | /uploads/1753379173818.pdf | 2025-07-24 17:46:13 |
|  3 | user last     | goal@job.com        | Manager   | NULL                      | 2025-07-24 17:46:49 |
+----+---------------+---------------------+-----------+---------------------------+---------------------+
3 rows in set (0.00 sec)

mysql>
```

Exit;

Ctrl+p and ctrl+q to come out of it safely.

- **Verify the uploads Volume**

The uploads volume is managed by Docker and should contain the uploaded resume files inside the backend container. Let's check it there:

1. **Access the Backend Container**

docker exec -it backend bash

List Files in the Container's /uploads Directory:

ls -l /uploads

```
ubuntu@ip-172-31-21-223:~/jobapp/client$ docker exec -it backend bash
root@d13077acf33e:/app# ls -l /uploads
total 400
-rw-r--r-- 1 root root 379046 Jul 24 17:41 1753378901623.pdf
-rw-r--r-- 1 root root  25702 Jul 24 17:46 1753379173818.pdf
root@d13077acf33e:/app#
```

Ctrl+p and ctrl+q to come out of it safely.

10. **Storage and Network Details**

- **Docker Network:**
    - **Name:** jobapp-network (bridge driver, default).
    - **Purpose:** Allows containers to communicate using container names (e.g., mysql for database, backend for API).
    - **Example:** The frontend's Nginx proxies requests to http://backend:3000, and the backend connects to mysql for database queries.
- **Docker Storage:**
    - **mysql-data:** Persists MySQL data in /var/lib/mysql inside the container, ensuring database data survives container restarts.
    - **uploads:** Stores resume PDFs in /uploads inside the backend container, mapped to the host via the uploads volume. Files are accessible across container restarts and can be viewed in /home/ubuntu/jobapp/uploads on the host if a bind mount is used instead (e.g., -v /home/ubuntu/jobapp/uploads:/uploads).


```
ubuntu@ip-172-31-21-223:~/jobapp$ cd uploads/
ubuntu@ip-172-31-21-223:~/jobapp/uploads$ docker volume ls
DRIVER    VOLUME NAME
local     mysql-data
local     uploads
ubuntu@ip-172-31-21-223:~/jobapp/uploads$ docker volume inspect uploads
[
    {
        "CreatedAt": "2025-07-24T07:41:02Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/uploads/_data",
        "Name": "uploads",
        "Options": null,
        "Scope": "local"
    }
]
ubuntu@ip-172-31-21-223:~/jobapp/uploads$
```

```
ubuntu@ip-172-31-21-223:~/jobapp/uploads$ sudo ls -l /var/lib/docker/volumes/uploads/_data
total 400
-rw-r--r-- 1 root root 379046 Jul 24 17:41 1753378901623.pdf
-rw-r--r-- 1 root root  25702 Jul 24 17:46 1753379173818.pdf
ubuntu@ip-172-31-21-223:~/jobapp/uploads$
```

11. **Use Docker Compose (Recommended; This is an alternative to individually running the containers ONLY at step 7)**

To simplify volume and container management, use docker-compose.yml.

Create /home/ubuntu/jobapp/docker-compose.yml:

```
version: '3.8'

services:

  mysql:

    image: mysql:8.0

    container_name: mysql

    networks:

      - jobapp-network

    volumes:

      - mysql-data:/var/lib/mysql

      - ./init.sql:/docker-entrypoint-initdb.d/init.sql

    environment:

      - MYSQL_ROOT_PASSWORD=rootpassword

  backend:

    build: ./server

    container_name: backend

    networks:

      - jobapp-network

    volumes:

      - uploads:/uploads

    environment:

      - DB_HOST=mysql

      - DB_USER=jobappuser
```

```
      - DB_PASSWORD=securepassword
      - DB_NAME=jobappdb
    depends_on:
      - mysql
  frontend:
    build: ./client
    container_name: frontend
    networks:
      - jobapp-network
    ports:
      - "80:80"
    depends_on:
      - backend
networks:
  jobapp-network:
    driver: bridge
volumes:
  mysql-data:
  uploads:
```

```
ubuntu@ip-172-31-21-223:~$ cd /home/ubuntu/jobapp/
ubuntu@ip-172-31-21-223:~/jobapp$ nano docker-compose.yml
ubuntu@ip-172-31-21-223:~/jobapp$ cat docker-compose.yml
version: '3.8'
services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    networks:
      - jobapp-network
    volumes:
      - mysql-data:/var/lib/mysql
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
    environment:
      - MYSQL_ROOT_PASSWORD=rootpassword
  backend:
    build: ./server
    container_name: backend
    networks:
      - jobapp-network
    volumes:
      - uploads:/uploads
    environment:
      - DB_HOST=mysql
      - DB_USER=jobappuser
      - DB_PASSWORD=securepassword
      - DB_NAME=jobappdb
    depends_on:
      - mysql
  frontend:
    build: ./client
    container_name: frontend
    networks:
      - jobapp-network
    ports:
      - "80:80"
    depends_on:
      - backend
networks:
  jobapp-network:
    driver: bridge
volumes:
  mysql-data:
  uploads:
ubuntu@ip-172-31-21-223:~/jobapp$
```

Run it:

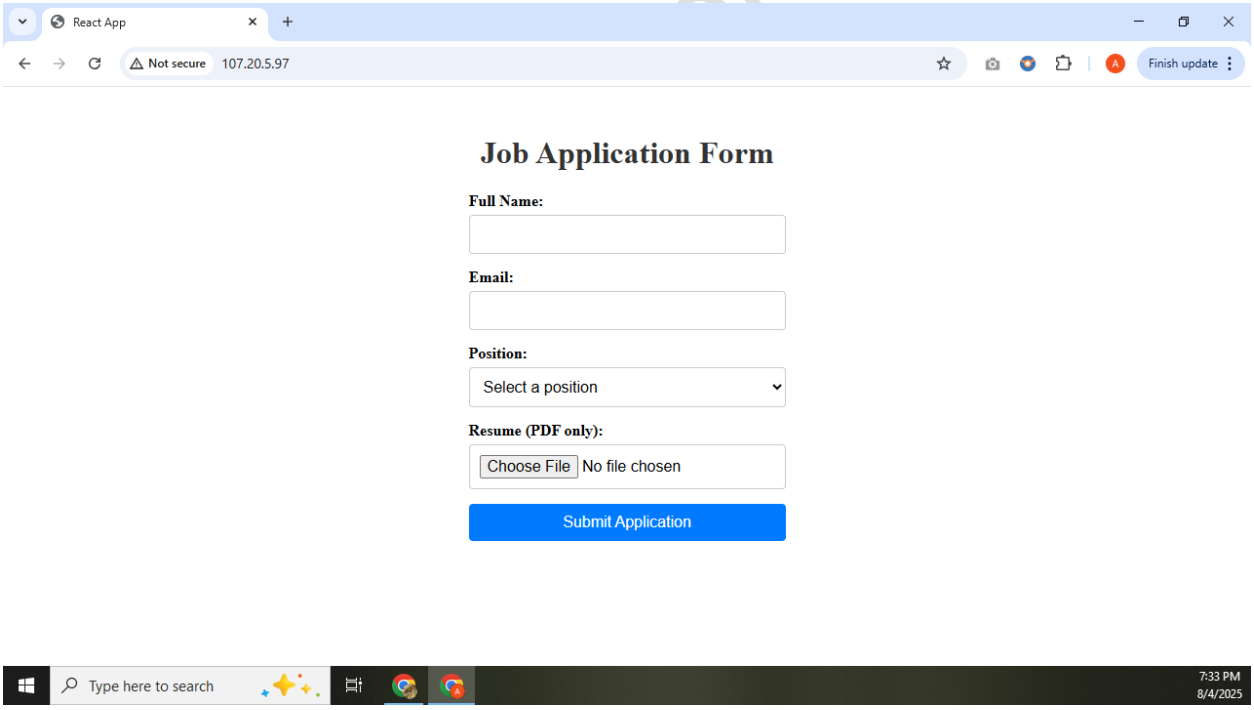
cd /home/ubuntu/jobapp

docker-compose down

docker-compose up -d

```
ubuntu@ip-172-31-21-223:~/jobapp$ docker-compose down
Removing network jobapp_jobapp-network
ubuntu@ip-172-31-21-223:~/jobapp$ docker-compose up -d
Creating network "jobapp_jobapp-network" with driver "bridge"
Creating mysql ... done
Creating backend ... done
Creating frontend ... done
ubuntu@ip-172-31-21-223:~/jobapp$
```

Verify that the web application is working fine.



Troubleshooting errors

If Docker-Compose is not present, Install it using below command.

sudo apt install docker-compose

```
ubuntu@ip-172-31-21-223:~/jobapp$ docker-compose down
Command 'docker-compose' not found, but can be installed with:
sudo snap install docker          # version 28.1.1+1, or
sudo apt  install docker-compose  # version 1.29.2-6
See 'snap info docker' for additional versions.
ubuntu@ip-172-31-21-223:~/jobapp$ sudo apt install docker-compose
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-compose python3-docker python3-dockerpty python3-docopt python3-dotenv python3-texttable python3-websocket
The following NEW packages will be installed:
  docker-compose python3-compose python3-docker python3-dockerpty python3-docopt python3-dotenv python3-texttable python3-websocket
0 upgraded, 8 newly installed, 0 to remove and 38 not upgraded.
Need to get 297 kB of archives.
After this operation, 1589 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 python3-websocket all 1.7.0-1 [38.1 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-docker all 5.0.3-1ubuntu1.1 [89.1 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 python3-dockerpty all 0.4.1-5 [11.4 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 python3-docopt all 0.6.2-6 [26.1 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 python3-dotenv all 1.0.1-1 [22.3 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 python3-texttable all 1.6.7-1 [11.0 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 python3-compose all 1.29.2-6ubuntu1 [84.6 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 docker-compose all 1.29.2-6ubuntu1 [14.0 kB]
Fetched 297 kB in 0s (7981 kB/s)
Selecting previously unselected package python3-websocket.
(Reading database ... 102680 files and directories currently installed.)
Preparing to unpack .../0-python3-websocket_1.7.0-1_all.deb ...
Unpacking python3-websocket (1.7.0-1) ...
```