1. **Frontend (Node.js with Express)**:

- Create a frontend using Express and Node.js.
- Include a form similar to the one from the Flask Assignment 2.
- Configure the form to send a request to the Flask backend.
- **Backend (Flask)**:
- Use the Flask backend to handle the form submission and process the data.
- **Folder Structure**:
- Organize the project with separate folders for the frontend and backend.
- **Docker Configuration**:
- Create a Dockerfile for both the frontend and backend.
- Write a .yaml file (Docker Compose) to connect both services in the same network.
- Upload both images to docker hub and push your whole code to github and add the node_modules and other non required files(.vscode) in .gitignore

. /backend/app.py

```
from flask import Flask, request, jsonify
import os
import psycopg2
app = Flask(__name__)
# Root route
@app.route("/", methods=["GET"])
def home():
    return jsonify({"message": "Backend is running!"})
# API data route
@app.route("/api/data", methods=["GET"])
def get_data():
    return jsonify({"message": "Backend is working", "status": "success"})
# Submit data route
@app.route("/submit", methods=["POST"])
```

```
def submit_form():
    data = request.get_json()
    name = data.get("name")
    email = data.get("email")
    return jsonify({"message": f"Received {name} with email {email}"})
# Database health check
@app.route("/db-check", methods=["GET"])
def db_check():
    try:
        conn = psycopg2.connect(
            host="db",  # Docker service name
            database=os.getenv("POSTGRES_DB"),
            user=os.getenv("POSTGRES_USER"),
            password=os.getenv("POSTGRES_PASSWORD"),
            port=5432
        )
        cur = conn.cursor()
        cur.execute("SELECT version();")
        db_version = cur.fetchone()[0]
        cur.close()
        conn.close()
        return jsonify({"db_version": db_version})
    except Exception as e:
        return jsonify({"error": str(e)}), 500
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

- **Purpose**:
  This is the **Flask backend application**.
- **What it does**:
  - Listens on port 5000 (inside container).
  - Provides **API endpoints**:
    - /api/data → GET → Returns sample backend message.
    - /submit → POST → Accepts JSON data (name, email) and returns confirmation.
    - /db-check → GET → Checks PostgreSQL DB connection using psycopg2 and returns version info.
- **Why important**:
  This is the main backend logic. It processes requests from the frontend or tools like curl.

2. /backend/requirements.txt

flask

flask-cors

psycopg2-binary

- **Purpose**:
  Lists **Python dependencies** required for the backend.
- **What it contains**:

flask
psycopg2-binary

- **Why important**:
  Used by pip inside Docker to install all Python libraries for Flask and PostgreSQL connectivity.

3. /backend/Dockerfile

FROM python:3.11-slim

```
WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["python", "app.py"]
```

- **Purpose**:
  Tells Docker **how to build** the backend image.
- **Steps inside**:
  - Use python:3.10 base image.
  - Set working directory to /app.
  - Copy requirements.txt and install packages with pip.
  - Copy all backend files into container.
  - Run python app.py when the container starts.
- **Why important**:
  Without it, Docker wouldn't know how to set up the backend environment.

4. /frontend/server.js

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

const axios = require("axios");

const cors = require("cors");

const app = express();

app.use(cors());

app.use(express.json()); // So we can handle JSON bodies


// Test route

app.get("/", (req, res) => {

  res.send("Frontend is running!");

});


// Fetch data from backend

app.get("/fetch-backend", async (req, res) => {
```

```
  try {
    const response = await axios.get("http://backend:5000/api/data");
    res.json(response.data);
  } catch (error) {
    res.status(500).json({ error: "Backend not reachable" });
  }
});

// Send data to backend
app.post("/send-to-backend", async (req, res) => {
  try {
    const response = await axios.post("http://backend:5000/submit", req.body);
    res.json(response.data);
  } catch (error) {
    res.status(500).json({ error: "Error sending data to backend" });
  }
});

app.listen(3000, () => {
  console.log("Frontend listening on port 3000");
});
```

- **Purpose**:
  This is the **Node.js + Express frontend server**.
- **What it does**:
  - Listens on port 3000.
  - / → Returns "Frontend is running!"
  - /fetch-backend → GET → Makes a request to backend:5000/api/data (container networking).
  - /send-data → POST → Sends JSON to backend /submit.

- **Why important**:
  Acts as the middle layer between the user and backend APIs.

5. /frontend/package.json

```
{
  "name": "frontend",
  "version": "1.0.0",
  "main": "server.js",
  "dependencies": {
    "express": "^4.18.2",
    "axios": "^1.6.0",
    "cors": "^2.8.5"
  }
}
```

- **Purpose**:
  Node.js project configuration file.
- **What it contains**:
  - App metadata (name, version).
  - dependencies like axios, cors, express.
  - Scripts (npm start runs server.js).
- **Why important**:
  Required by Node.js to install dependencies with npm install.

6. /frontend/Dockerfile

```
FROM node:18

WORKDIR /app
```

```
COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

- **Purpose**:
  Tells Docker **how to build** the frontend image.
- **Steps inside**:
  - Use node:18 base image.
  - Set working directory to /app.
  - Copy package*.json and run npm install.
  - Copy all frontend files.
  - Run npm start when the container starts.
- **Why important**:
  Defines the container environment for Node.js frontend.

7. /docker-compose.yml

```
version: "3.9"

services:

  backend:

    build: ./backend

    container_name: backend_container

    ports:

      - "5000:5000"

    env_file:

      - ./backend/.env

    volumes:

      - ./backend:/app

    depends_on:

      db:
```

```
      condition: service_healthy
  frontend:
    build: ./frontend
    container_name: frontend_container
    ports:
      - "3000:3000"
    volumes:
      - ./frontend:/app
      - /app/node_modules
    stdin_open: true
    tty: true
    depends_on:
      backend:
        condition: service_started
  db:
    image: postgres:15
    container_name: postgres_container
    restart: always
    environment:
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
      POSTGRES_DB: mydb
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U myuser -d mydb"]
```

```
    interval: 5s
    timeout: 5s
    retries: 5

volumes:
  postgres_data:
```

- **Purpose**:
  Orchestrates **multiple containers**: frontend, backend, PostgreSQL DB.
- **What it defines**:
  - **backend** → Builds from backend/Dockerfile, connects to DB.
  - **frontend** → Builds from frontend/Dockerfile, connects to backend.
  - **db** → Uses postgres:15 official image.
  - Networking between containers is automatic (Docker Compose default network).
- **Why important**:
  One command (docker-compose up) starts all services together.

8. /.gitignore

```
node_modules/
.vscode/
__pycache__/
*.pyc
.env
```

- **Purpose**:
  Lists files/folders Git should **not track**.
- **Example entries**:
  - .venv/ → Python virtual environment.
  - node_modules/ → Node.js dependencies.

    - .vscode/ → Editor configs.
    - .env → Environment secrets.
- **Why important**:
  Prevents uploading large/unnecessary files to GitHub.

9. /README.md

Flask + Node.js + Docker Project

Overview

This project is a **Dockerized Full Stack App** with:

- Backend: Flask + PostgreSQL
- Frontend: Node.js + Express
- Database: PostgreSQL

How to Run

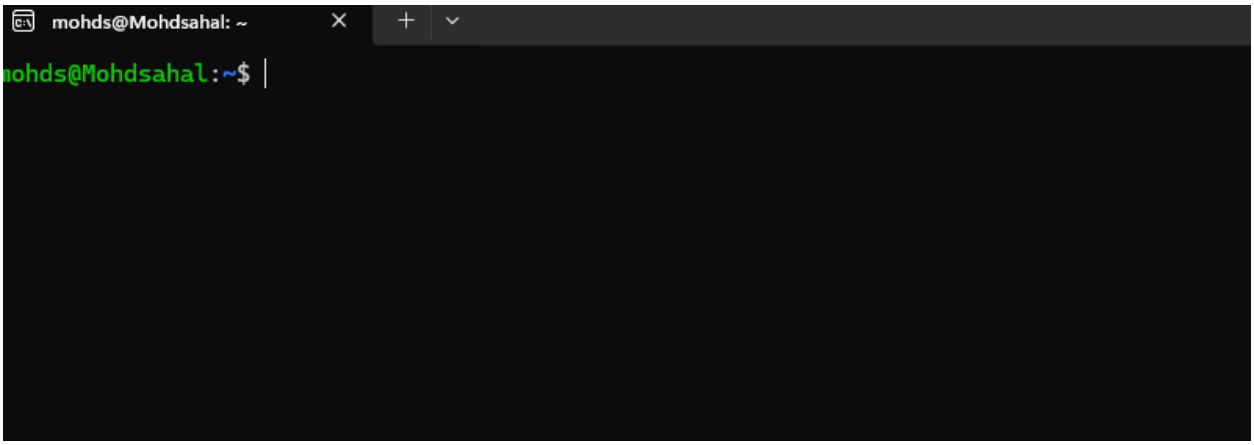
docker-compose up --build

- **Purpose**:
  Documentation for the project.
- **Contents**:
    - Overview of the stack.
    - Setup instructions.
    - API endpoint descriptions.
    - Docker commands.
- **Why important**:
  Makes the project easy for others (and yourself later) to understand.

SCREENSHOTS

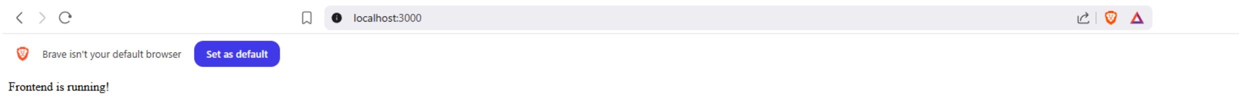

mohds@Mohdsahal: /mnt/c/t
GNU nano 7.2
app.py *
from flask import Flask
app = Flask(__name__)
@app.route("/")
def home():
return "Hello from Flask Backend!"
if __name__ == "__main__":
app.run(host="0.0.0.0", port=5000)
^G Help
^X Exit
^O Write Out
^R Read File
^W Where Is
^\ Replace
^K Cut
^U Paste
^T Execute
^J Justify
^C Location
^/ Go To Line
M-U Undo
M-E Redo
M-A Set Mark
M-6 Copy
M-] To Bracket
^Q Where Was
M-Q Previous
M-W Next
^B Back
^F Forward
^◂ Prev Word
^▸ Next Word

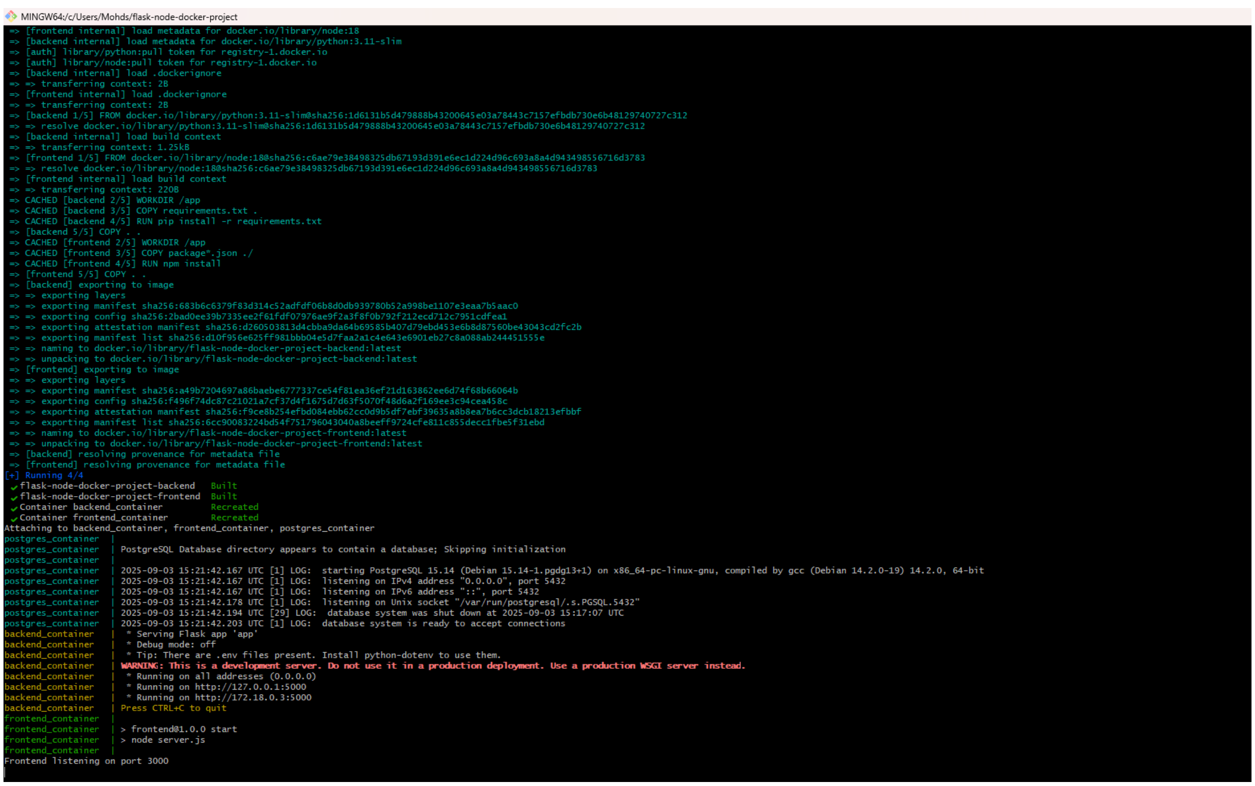

GNU nano 7.2
Dockerfile
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python", "app.py"]
[Read 7 lines (Converted from DOS format)]
^G Help
^X Exit
^O Write Out
^R Read File
^W Where Is
^\ Replace
^K Cut
^U Paste
^T Execute
^J Justify
^C Location
^/ Go To Line
M-U Undo
M-E Redo
M-A Set Mark
M-6 Copy
M-] To Bracket
^Q Where Was
M-Q Previous
M-W Next
^B Back
^F Forward
^◂ Prev Word
^▸ Next Word

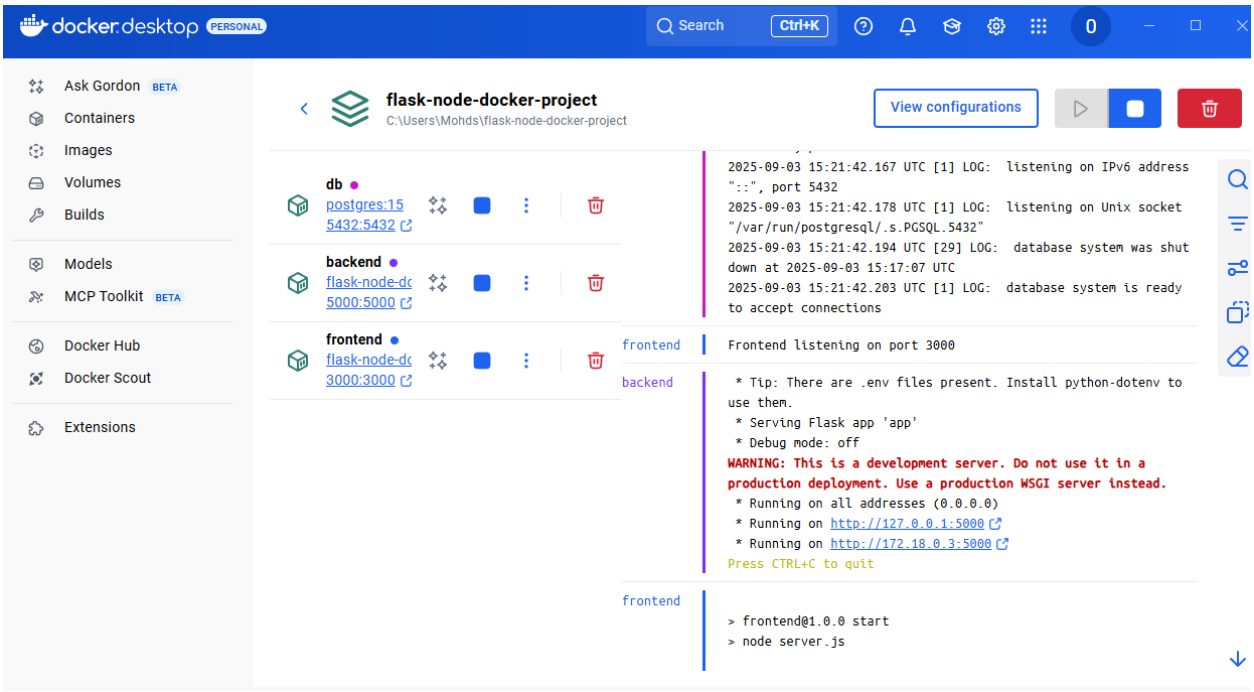


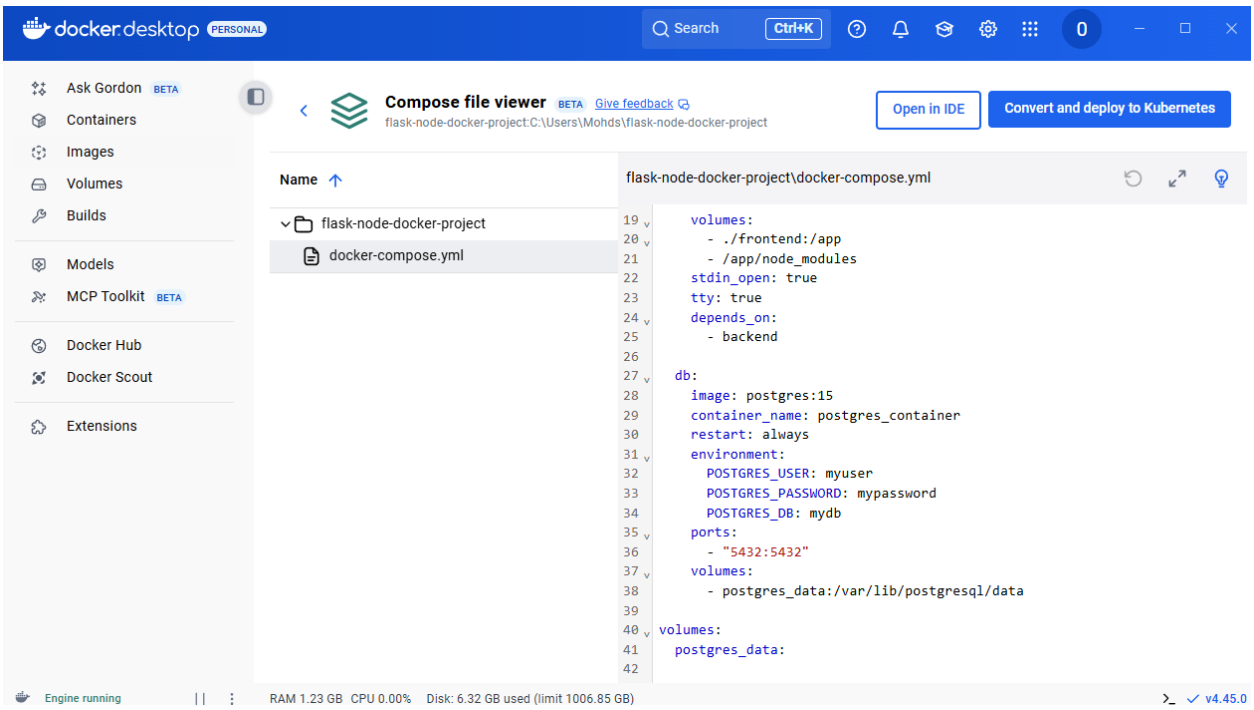

```
root@Mohdsahal:/mnt/c/Users/Mohds# wsl -u root
Command 'wsl' not found, but can be installed with:
apt install wsl
root@Mohdsahal:/mnt/c/Users/Mohds# passwd mohds
New password:
Retype new password:
passwd: password updated successfully
```

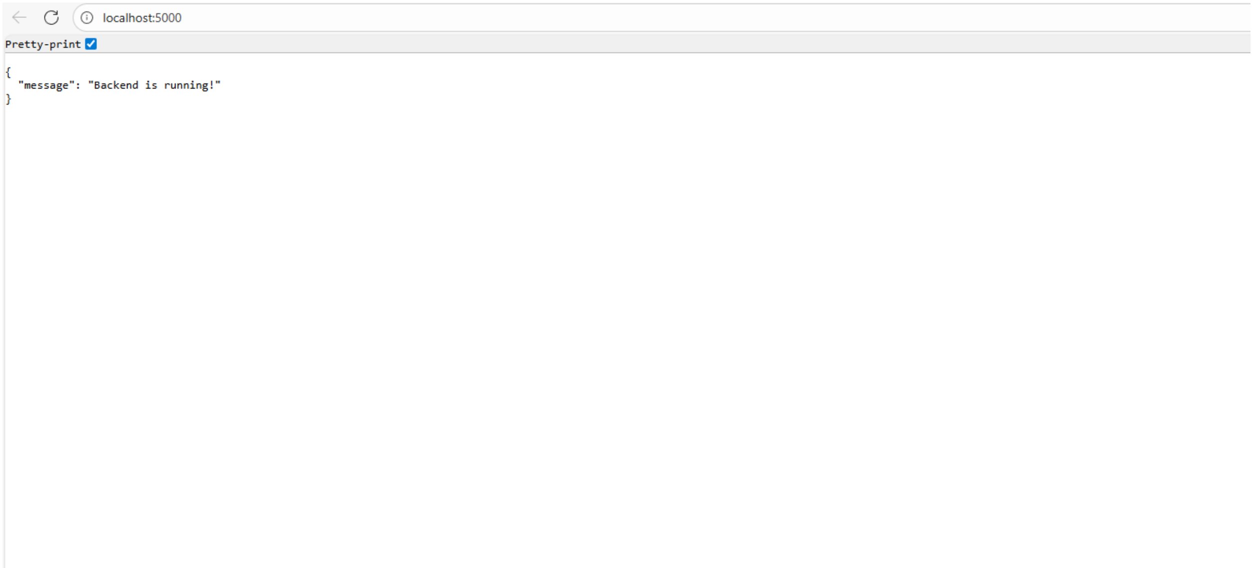
```
Mohds@Mohdsahal MINGW64 ~/flask-node-docker-project (master_1)
$ ^[[200~ls -R
bash: $'\E[200~ls': command not found

Mohds@Mohdsahal MINGW64 ~/flask-node-docker-project (master_1)

[+] Running 13/15
 - db [##################] 127.9MB / 157MB   Pulling
   ✔ 5c10925b29ff Download complete
   ✔ c6bcc2c9a041 Pull complete
   ✔ 799f859abbbd Download complete
   ✔ ac4b721eb66f Pull complete
   ✔ 2d9e29180a27 Pull complete
   ✔ d903e777080c Download complete
   ✔ 396b1da7636e Pull complete
   - f5dfd246bc6e Downloading   [==================================>           ]  84.93MB/111.1MB
   ✔ 631fe8c6d606 Pull complete
   ✔ 77c7671c4414 Pull complete
   ✔ 606bd164980e Pull complete
   ✔ ba7036bb0a60 Download complete
   ✔ 04adcb462801 Download complete
   ✔ fca2566eba32 Pull complete
```

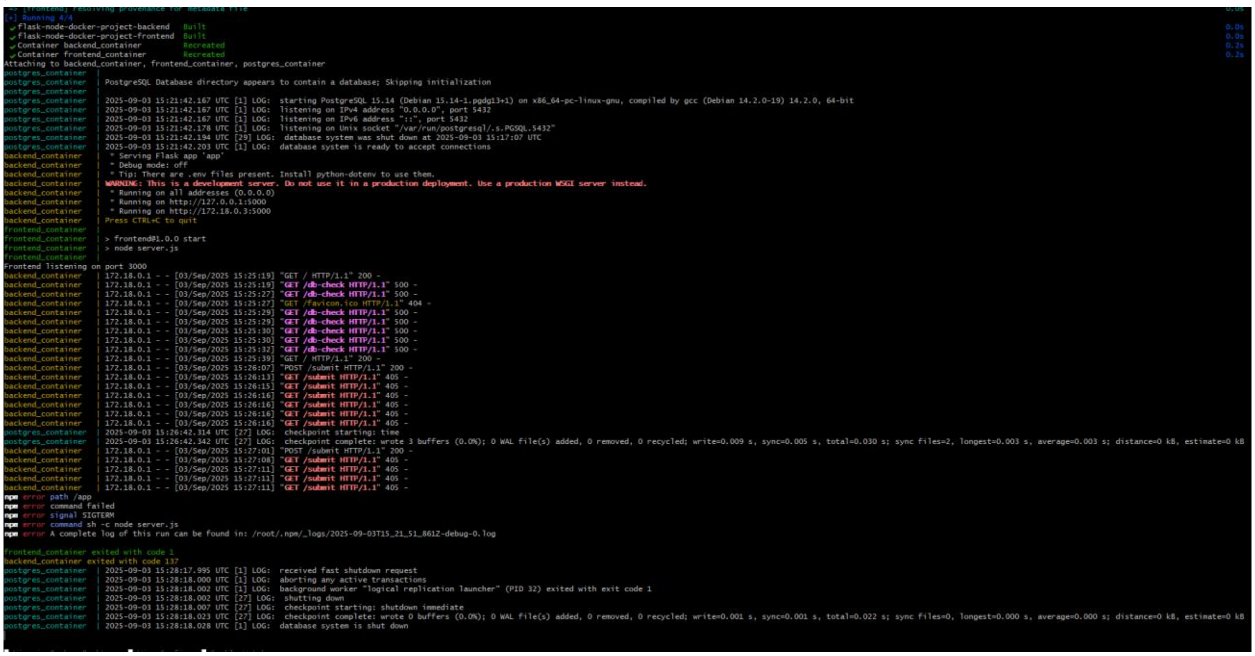

localhost:3000
Brave isn't your default browser
Set as default
Frontend is running!

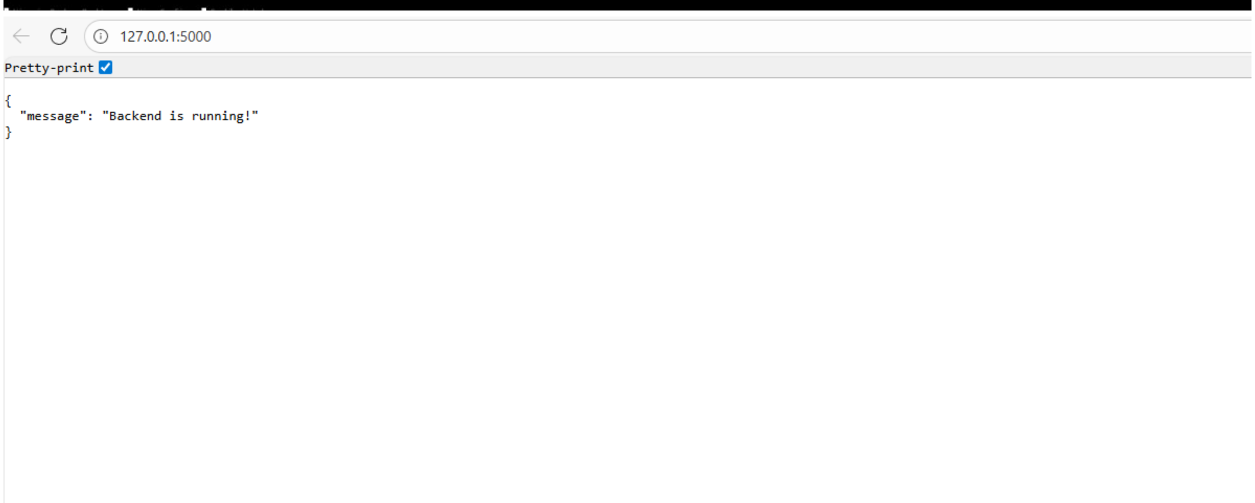

MINGW64:/c/Users/Mohds/flask-node-docker-project
=> [frontend internal] load metadata for docker.io/library/node:18
=> [backend internal] load metadata for docker.io/library/python:3.11-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [auth] library/node:pull token for registry-1.docker.io
=> [backend internal] load .dockerignore
=> => transferring context: 2B
=> [frontend internal] load .dockerignore
=> => transferring context: 2B
=> [backend 1/5] FROM docker.io/library/python:3.11-slim@sha256:1d6131b5d479888b43200645e03a78443c7157efbdb730e6b48129740727c312
=> => resolve docker.io/library/python:3.11-slim@sha256:1d6131b5d479888b43200645e03a78443c7157efbdb730e6b48129740727c312
=> [backend internal] load build context
=> => transferring context: 1.25kB
=> [frontend 1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
=> => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
=> [frontend internal] load build context
=> => transferring context: 220B
=> CACHED [backend 2/5] WORKDIR /app
=> CACHED [backend 3/5] COPY requirements.txt .
=> CACHED [backend 4/5] RUN pip install -r requirements.txt
=> [backend 5/5] COPY . .
=> CACHED [frontend 2/5] WORKDIR /app
=> CACHED [frontend 3/5] COPY package*.json ./
=> CACHED [frontend 4/5] RUN npm install
=> [frontend 5/5] COPY . .
=> [backend] exporting to image
=> => exporting layers
=> => exporting manifest sha256:683b6c6379f83d314c52adfdf06b8d0db939780b52a998be1107e3eaa7b5aac0
=> => exporting config sha256:2bad0ee39b7335ee2f61fdf07976ae9f2a3f8f0b792f212ecd712c7951cdfea1
=> => exporting attestation manifest sha256:d260503813d4cbba9da64b69585b407d79ebd453e6b8d87560be43043cd2fc2b
=> => exporting manifest list sha256:d10f956e625ff981bbb04e5d7faa2a1c4e643e6901eb27c8a088ab244451555e
=> => naming to docker.io/library/flask-node-docker-project-backend:latest
=> => unpacking to docker.io/library/flask-node-docker-project-backend:latest
=> [frontend] exporting to image
=> => exporting layers
=> => exporting manifest sha256:a49b7204697a86baebe6777337ce54f81ea36ef21d163862ee6d74f68b66064b
=> => exporting config sha256:f496f74dc87c21021a7cf37d4f1675d7d63f5070f48d6a2f169ee3c94cea458c
=> => exporting attestation manifest sha256:f9ce8b254efbd084ebb62cc0d9b5df7ebf39635a8b8ea7b6cc3dcb18213efbbf
=> => exporting manifest list sha256:6cc90083224bd54f751796043040a8beeff9724cfe811c855decc1fbe5f31ebd
=> => naming to docker.io/library/flask-node-docker-project-frontend:latest
=> => unpacking to docker.io/library/flask-node-docker-project-frontend:latest
=> [backend] resolving provenance for metadata file
=> [frontend] resolving provenance for metadata file
[+] Running 4/4
✔ flask-node-docker-project-backend Built
✔ flask-node-docker-project-frontend Built
✔ Container backend_container Recreated
✔ Container frontend_container Recreated
Attaching to backend_container, frontend_container, postgres_container
postgres_container |
postgres_container | PostgreSQL Database directory appears to contain a database; Skipping initialization
postgres_container |
postgres_container | 2025-09-03 15:21:42.167 UTC [1] LOG: starting PostgreSQL 15.14 (Debian 15.14-1.pgdg13+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit
postgres_container | 2025-09-03 15:21:42.167 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
postgres_container | 2025-09-03 15:21:42.167 UTC [1] LOG: listening on IPv6 address "::", port 5432
postgres_container | 2025-09-03 15:21:42.178 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres_container | 2025-09-03 15:21:42.194 UTC [29] LOG: database system was shut down at 2025-09-03 15:17:07 UTC
postgres_container | 2025-09-03 15:21:42.203 UTC [1] LOG: database system is ready to accept connections
backend_container | * Serving Flask app 'app'
backend_container | * Debug mode: off
backend_container | * Tip: There are .env files present. Install python-dotenv to use them.
backend_container | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
backend_container | * Running on all addresses (0.0.0.0)
backend_container | * Running on http://127.0.0.1:5000
backend_container | * Running on http://172.18.0.3:5000
backend_container | Press CTRL+C to quit
frontend_container |
frontend_container | > frontend@1.0.0 start
frontend_container | > node server.js
frontend_container |
Frontend listening on port 3000


docker desktop PERSONAL
Search Ctrl+K
Ask Gordon BETA
Containers
Images
Volumes
Builds
Models
MCP Toolkit BETA
Docker Hub
Docker Scout
Extensions
flask-node-docker-project
C:\Users\Mohds\flask-node-docker-project
View configurations
db
postgres:15
5432:5432
backend
flask-node-dc
5000:5000
frontend
flask-node-dc
3000:3000
2025-09-03 15:21:42.167 UTC [1] LOG: listening on IPv6 address "::", port 5432
2025-09-03 15:21:42.178 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2025-09-03 15:21:42.194 UTC [29] LOG: database system was shut down at 2025-09-03 15:17:07 UTC
2025-09-03 15:21:42.203 UTC [1] LOG: database system is ready to accept connections
frontend
Frontend listening on port 3000
backend
* Tip: There are .env files present. Install python-dotenv to use them.
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.18.0.3:5000
Press CTRL+C to quit
frontend
> frontend@1.0.0 start
> node server.js


docker desktop PERSONAL
Search Ctrl+K
Ask Gordon BETA
Containers
Images
Volumes
Builds
Models
MCP Toolkit BETA
Docker Hub
Docker Scout
Extensions
Compose file viewer BETA Give feedback
flask-node-docker-project:C:\Users\Mohds\flask-node-docker-project
Open in IDE
Convert and deploy to Kubernetes
Name
flask-node-docker-project
docker-compose.yml
flask-node-docker-project\docker-compose.yml
volumes:
- ./frontend:/app
- /app/node_modules
stdin_open: true
tty: true
depends_on:
- backend
db:
image: postgres:15
container_name: postgres_container
restart: always
environment:
POSTGRES_USER: myuser
POSTGRES_PASSWORD: mypassword
POSTGRES_DB: mydb
ports:
- "5432:5432"
volumes:
- postgres_data:/var/lib/postgresql/data
volumes:
postgres_data:
Engine running
RAM 1.23 GB CPU 0.00% Disk: 6.32 GB used (limit 1006.85 GB)
v4.45.0


localhost:5000
Pretty-print
{
"message": "Backend is running!"
}


[+] Running 4/4
flask-node-docker-project-backend Built
flask-node-docker-project-frontend Built
Container backend_container Recreated
Container frontend_container Recreated
Attaching to backend_container, frontend_container, postgres_container
postgres_container | PostgreSQL Database directory appears to contain a database; Skipping initialization
postgres_container | 2025-09-03 15:21:42.167 UTC [1] LOG: starting PostgreSQL 15.14 (Debian 15.14-1.pgdg13+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit
postgres_container | 2025-09-03 15:21:42.167 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
postgres_container | 2025-09-03 15:21:42.167 UTC [1] LOG: listening on IPv6 address "::", port 5432
postgres_container | 2025-09-03 15:21:42.178 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres_container | 2025-09-03 15:21:42.194 UTC [29] LOG: database system was shut down at 2025-09-03 15:17:07 UTC
postgres_container | 2025-09-03 15:21:42.203 UTC [1] LOG: database system is ready to accept connections
backend_container | * Serving Flask app 'app'
backend_container | * Debug mode: off
backend_container | * Tip: There are .env files present. Install python-dotenv to use them.
backend_container | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
backend_container | * Running on all addresses (0.0.0.0)
backend_container | * Running on http://127.0.0.1:5000
backend_container | * Running on http://172.18.0.3:5000
backend_container | Press CTRL+C to quit
frontend_container | > frontend@1.0.0 start
frontend_container | > node server.js
Frontend listening on port 3000
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:19] "GET / HTTP/1.1" 200 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:19] "GET /db-check HTTP/1.1" 500 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:27] "GET /db-check HTTP/1.1" 500 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:27] "GET /favicon.ico HTTP/1.1" 404 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:29] "GET /db-check HTTP/1.1" 500 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:29] "GET /db-check HTTP/1.1" 500 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:30] "GET /db-check HTTP/1.1" 500 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:30] "GET /db-check HTTP/1.1" 500 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:32] "GET /db-check HTTP/1.1" 500 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:25:39] "GET / HTTP/1.1" 200 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:26:07] "POST /submit HTTP/1.1" 200 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:26:13] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:26:15] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:26:16] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:26:16] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:26:16] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:26:16] "GET /submit HTTP/1.1" 405 -
postgres_container | 2025-09-03 15:26:42.314 UTC [27] LOG: checkpoint starting: time
postgres_container | 2025-09-03 15:26:42.342 UTC [27] LOG: checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.009 s, sync=0.005 s, total=0.030 s; sync files=2, longest=0.003 s, average=0.003 s; distance=0 kB, estimate=0 kB
backend_container | 172.18.0.1 - - [03/Sep/2025 15:27:01] "POST /submit HTTP/1.1" 200 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:27:08] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:27:11] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:27:11] "GET /submit HTTP/1.1" 405 -
backend_container | 172.18.0.1 - - [03/Sep/2025 15:27:11] "GET /submit HTTP/1.1" 405 -
npm error path /app
npm error command failed
npm error signal SIGTERM
npm error command sh -c node server.js
npm error A complete log of this run can be found in: /root/.npm/_logs/2025-09-03T15_21_51_861Z-debug-0.log
frontend_container exited with code 1
backend_container exited with code 137
postgres_container | 2025-09-03 15:28:17.995 UTC [1] LOG: received fast shutdown request
postgres_container | 2025-09-03 15:28:18.000 UTC [1] LOG: aborting any active transactions
postgres_container | 2025-09-03 15:28:18.002 UTC [1] LOG: background worker "logical replication launcher" (PID 32) exited with exit code 1
postgres_container | 2025-09-03 15:28:18.002 UTC [27] LOG: shutting down
postgres_container | 2025-09-03 15:28:18.007 UTC [27] LOG: checkpoint starting: shutdown immediate
postgres_container | 2025-09-03 15:28:18.023 UTC [27] LOG: checkpoint complete: wrote 0 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.001 s, sync=0.001 s, total=0.022 s; sync files=0, longest=0.000 s, average=0.000 s; distance=0 kB, estimate=0 kB
postgres_container | 2025-09-03 15:28:18.028 UTC [1] LOG: database system is shut down


127.0.0.1:5000
Pretty-print
{
"message": "Backend is running!"
}

```
Mohds@Mohdsahal MINGW64 ~/flask-node-docker-project (master_1)
$ docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS                                       NAMES
ce2e2a853939   flask-node-docker-project-frontend   "docker-entrypoint.s…"   3 minutes ago    Up 3 minutes    0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp   frontend_container
ff967a1517a6   flask-node-docker-project-backend    "python app.py"          3 minutes ago    Up 3 minutes    0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   backend_container
f97b9a06a06d   postgres:15                        "docker-entrypoint.s…"   15 minutes ago   Up 15 minutes   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp   postgres_container
```

```
Mohds@Mohdsahal MINGW64 ~/flask-node-docker-project (master_1)
$ curl http://localhost:5000/
{"message":"Backend is running!"}
```

127.0.0.1:5000

Pretty-print

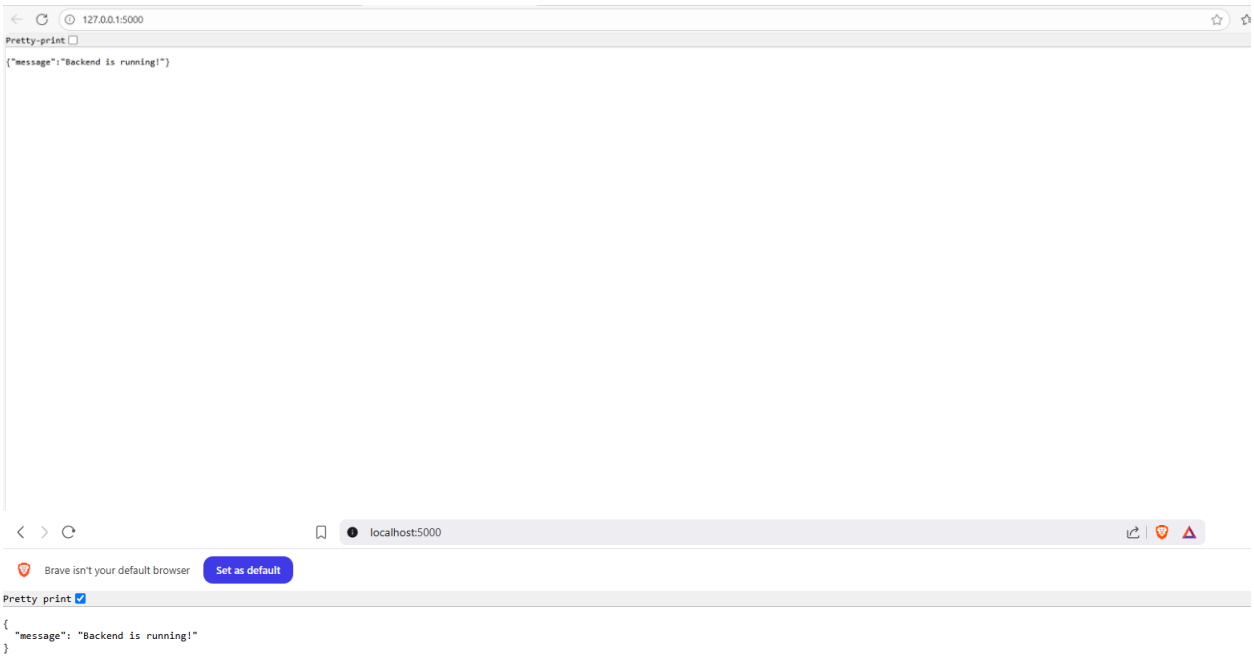

```
{
  "message": "Backend is running!"
}
```


localhost:5000/api/data

Brave isn't your default browser Set as default

Pretty print


```
{
  "message": "Backend is working",
  "status": "success"
}
```


127.0.0.1:5000
Pretty-print
{"message":"Backend is running!"}

localhost:5000
Brave isn't your default browser
Set as default
Pretty print
{
"message": "Backend is running!"
}