

# Cloud Weather Tracker

---

## Pekan 13: CI/CD Setup dan Cloud Deployment

### Link

Backend : <https://cloud-weather-tracker-production.up.railway.app/>

Backend for test API : <https://cloud-weather-tracker-production.up.railway.app/api>

Frontend : <https://cloud-weather-tracker.vercel.app/>

### Checkpoint

1. Setup GitHub Actions untuk CI/CD pipeline dasar
2. Deploy aplikasi ke platform cloud (pilih salah satu: AWS, Google Cloud, Azure, atau layanan alternatif seperti DigitalOcean, Heroku)
3. Konfigurasi environment variables dan secrets

### Jawaban Checkpoint

**1 + 2 + 3 . Setup GitHub Actions untuk CI/CD pipeline dasar dan deploy aplikasi ke platform cloud dan setting variable**

---

Untuk menjawab point nomor 1,2 dan 3, saya menggunakan dua platform Cloud yang berbeda, untuk Backend saya menggunakan Railways dan untuk frontend saya menggunakan Vercel, untuk penjelasan dimulai dari backend

## Backend Deployment

---

### 1. Langkah Pertama

Dalam backend diperlukan file baru yaitu Procfile dengan isi

```
web: gunicorn app:app
```

digunakan untuk menjalankan aplikasi menggunakan gunicorn

lalu Dalam backend diperlukan update file baru yaitu requirements.txt dengan isi

```
Flask
flask-cors
flask-sqlalchemy
requests
psycopg2-binary
python-dotenv
gunicorn
```

## 2. Langkah Kedua

Dalam backend diperlukan update file di app.py

app.py

```
@app.route("/init-db")
def init_db():
    try:
        db.create_all()
        return "✅ Database tables created successfully!"
    except Exception as e:
        return f"❌ Error: {str(e)}"
```

pertama membuat fungsi baru dalam app.py dengan nama init-db, dengan tujuan Agar dapat menginisialisasi tabel database PostgreSQL secara manual dari browser atau Postman, terutama setelah deploy pertama kali.

config.py

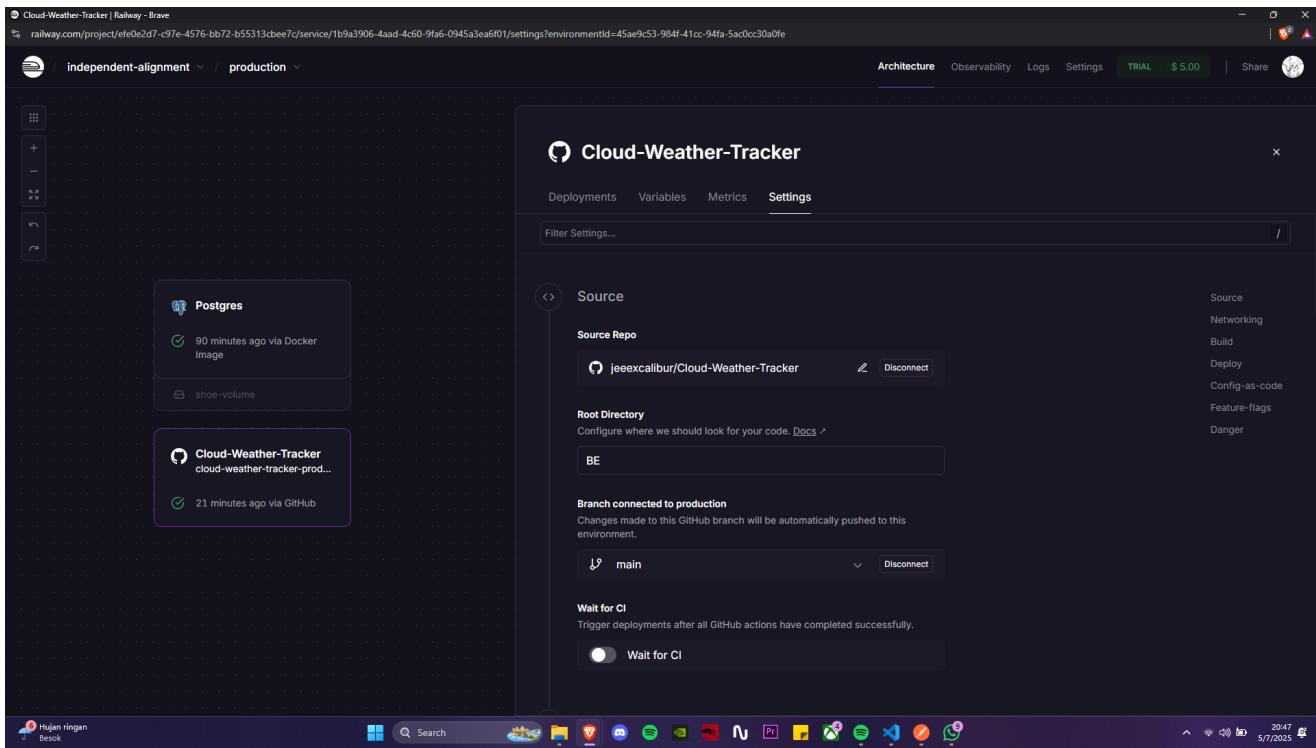
```
class Config:
    API_KEY = os.getenv("API_KEY")
    SQLALCHEMY_DATABASE_URI = os.getenv("DATABASE_URL")
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

perubahan ini dilakukan Agar Flask dapat membaca variable API\_KEY dan DATABASE\_URL yang dikonfigurasi di Railway Environment Variables. Tanpa ini, backend akan gagal fetch data dari OpenWeatherMap.

## 3. Langkah Ketiga

Setelah melakukan beberapa perubahan pada file backend selanjutnya melakukan Deployment pada Railway, dimana membuat project baru pada railway dengan nama Cloud-Weather-Tracker, lalu masuk ke pengaturan

### 1. Source Repository



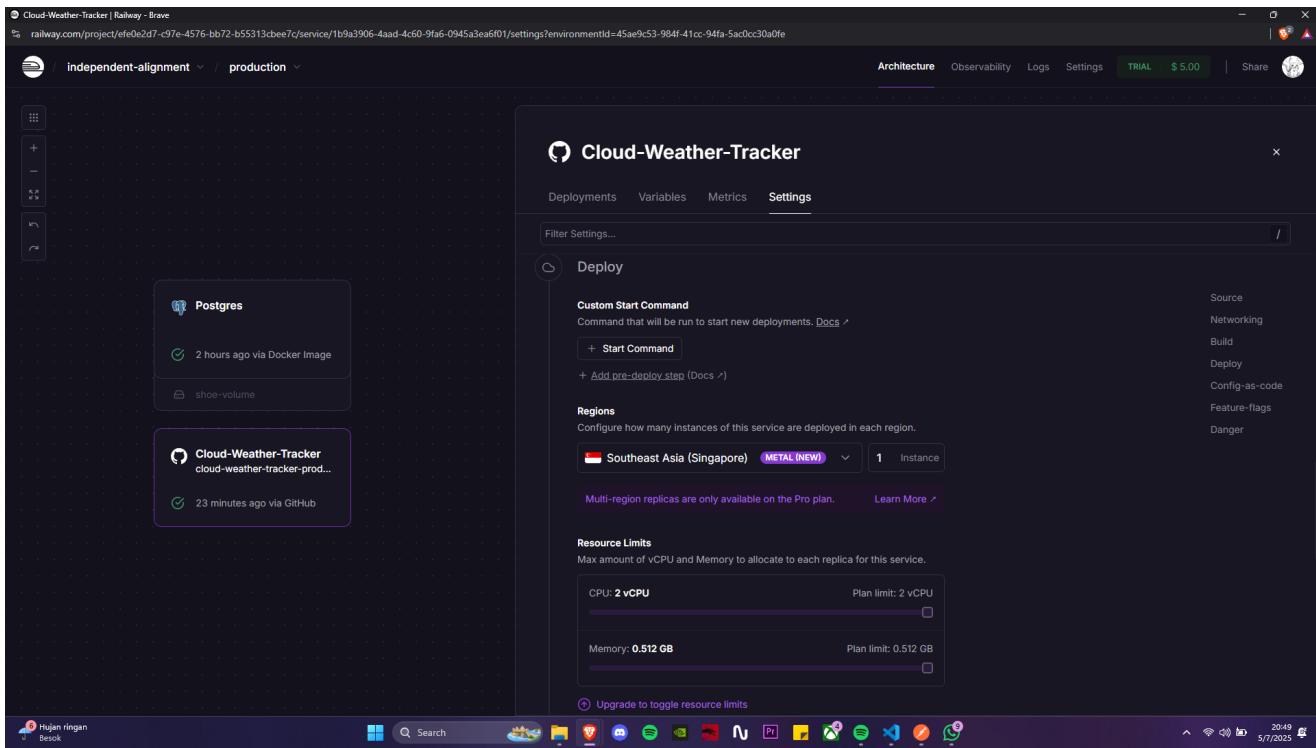
CI/CD (Continuous Integration and Continuous Deployment) dalam proyek ini berjalan secara otomatis melalui integrasi antara Railway dan GitHub.

Setiap kali saya melakukan push atau update ke branch utama (main) pada repository bernama Cloud-Weather-Tracker, Railway secara otomatis akan:

1. Menarik perubahan terbaru dari repository
2. Membangun ulang environment backend
3. Menjalankan proses deployment menggunakan konfigurasi yang telah disediakan
4. Mengupdate aplikasi secara langsung ke URL live tanpa perlu proses manual

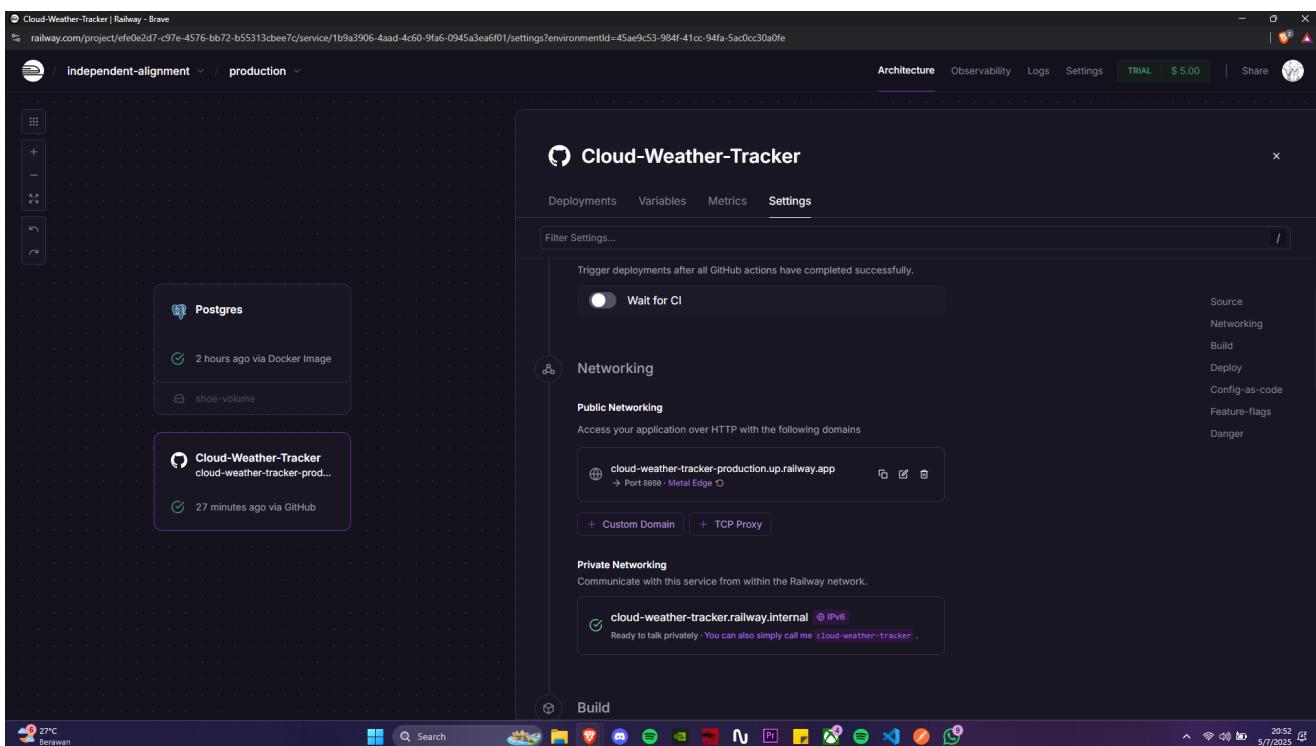
Proyek ini menggunakan struktur direktori dengan folder BE/ (backend) sebagai root deployment. Branch default yang digunakan adalah main, sesuai dengan standar pengembangan di repository ini.

## 2. Region Set



Untuk koneksi server yang maksimal dikarenakan kita berada di Indonesia lokasi server yang paling dekat adalah Singapore, karena itu saya menggunakan server singapore pada backend nya

### 3. Network for Domain



Untuk Network saya menggunakan generate domain default dari railway yang mana tergenerasi secara otomatis oleh railway yaitu

<https://cloud-weather-tracker-production.up.railway.app/>

### 4. Variable Set



Dalam aplikasi ini, dua variabel yang digunakan untuk memastikan aplikasi berjalan dengan baik di Railway adalah:

### 1. API\_KEY

Variabel ini berisi kunci API dari OpenWeatherMap yang digunakan untuk mengambil data cuaca dan kualitas udara dari API tersebut. API\_KEY ini sangat penting karena tanpa kunci yang valid, aplikasi tidak akan dapat mengakses data cuaca dan kualitas udara yang dibutuhkan. Kunci API ini disimpan di environment variables Railway untuk menjaga keamanannya.

Penggunaan:

```
api_key = Config.API_KEY
```

### 2. DATABASE\_URL

Variabel ini berisi URL koneksi ke database PostgreSQL yang digunakan oleh aplikasi untuk menyimpan data cuaca dan kualitas udara yang diterima. Variabel ini juga disimpan di environment variables Railway untuk mengonfigurasi database yang digunakan aplikasi, agar bisa diakses secara aman dan terhubung dengan baik saat aplikasi berjalan di server Railway.

Penggunaan:

```
SQLALCHEMY_DATABASE_URI = os.getenv("DATABASE_URL")
```

## 5. Deployment

Untuk melakukan deployment saya menggunakan tombol deploy yang ada di bagian kanan atas pada halaman dashboard

The screenshot shows the Railway dashboard for the 'Cloud-Weather-Tracker' project. On the left, there's a sidebar with a 'Postgres' icon and a 'Cloud-Weather-Tracker' container icon. The main area displays deployment history for the 'cloud-weather-tracker-production.up.railway.app' environment. A deployment titled 'fix: update API call with backend validation...' is shown as 'ACTIVE' with a timestamp of '38 minutes ago via GitHub'. This deployment is marked as 'Deployment successful' and includes a list of steps: 'Initialization' (0:03), 'Build' (0:18), 'Deploy' (0:12), and 'Post-deploy' (0:00). Below this, there's a 'HISTORY' section. The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating 'Hujan ringan Besok' and the date '5/7/2025'.

## 6. Database Setup

untuk database melakukan setup untuk postgree, pertama tama copy connection url untuk variables database yang akan digunakan di backend

The screenshot shows the Railway UI interface for connecting to a PostgreSQL database. At the top, there are tabs for 'Private Network' and 'Public Network'. A note indicates that connecting over the public network causes egress costs. Below this, there are three sections: 'Connection URL' (show open), 'Raw `psql` command' (show), and 'Railway CLI `connect` command' (show). Each section contains a code snippet and a copy icon.

```
postgresql://postgres:*****@nozomi.proxy.rlwy.net:17483/railway
```

```
PGPASSWORD=***** psql -h nozomi.proxy.rlwy.net -U postgres -p 17483 -d railway
```

```
railway connect Postgres
```

lalu jalankan fungsi init-db pada app.py

```
@app.route("/init-db")
def init_db():
    try:
        db.create_all()
        return "✅ Database tables created successfully!"
    except Exception as e:
        return f"❌ Error: {str(e)}"
```

A screenshot of a browser window showing the result of running the init\_db function. The URL is 'cloud-weather-tracker-production.up.railway.app/init-db'. The page displays the message 'Database tables created successfully!' with a green checkmark icon.

Database tables created successfully!

Table akan otomatis terbuat berkat models yang telah dibuat sebelumnya

The screenshot shows the Railway platform interface. At the top, it says "Postgres | Railway - Brave" and the URL "railway.com/project/efe0e2d7-c97e-4576-bb72-b55313bee7c/service/96c1a100-5dd4-4ef1-b635-7fb0001ea06/data?environmentId=45ae9c53-984f-41cc-94fa-5ac0c30a0fe". The main area is titled "Postgres" and has tabs for Deployments, Data, Backups, Variables, Metrics, and Settings. Under the Data tab, there's a "Tables" section with icons for "air\_quality\_data" and "weather\_data". A purple "Create table" button is located at the bottom right of this section. To the left, there's a sidebar with a tree view and deployment logs for "Postgres" (2 hours ago via Docker Image) and "Cloud-Weather-Tracker" (42 minutes ago via GitHub). The top navigation bar includes tabs for Architecture, Observability, Logs, Settings, TRIAL (\$ 5.00), Share, and Help.

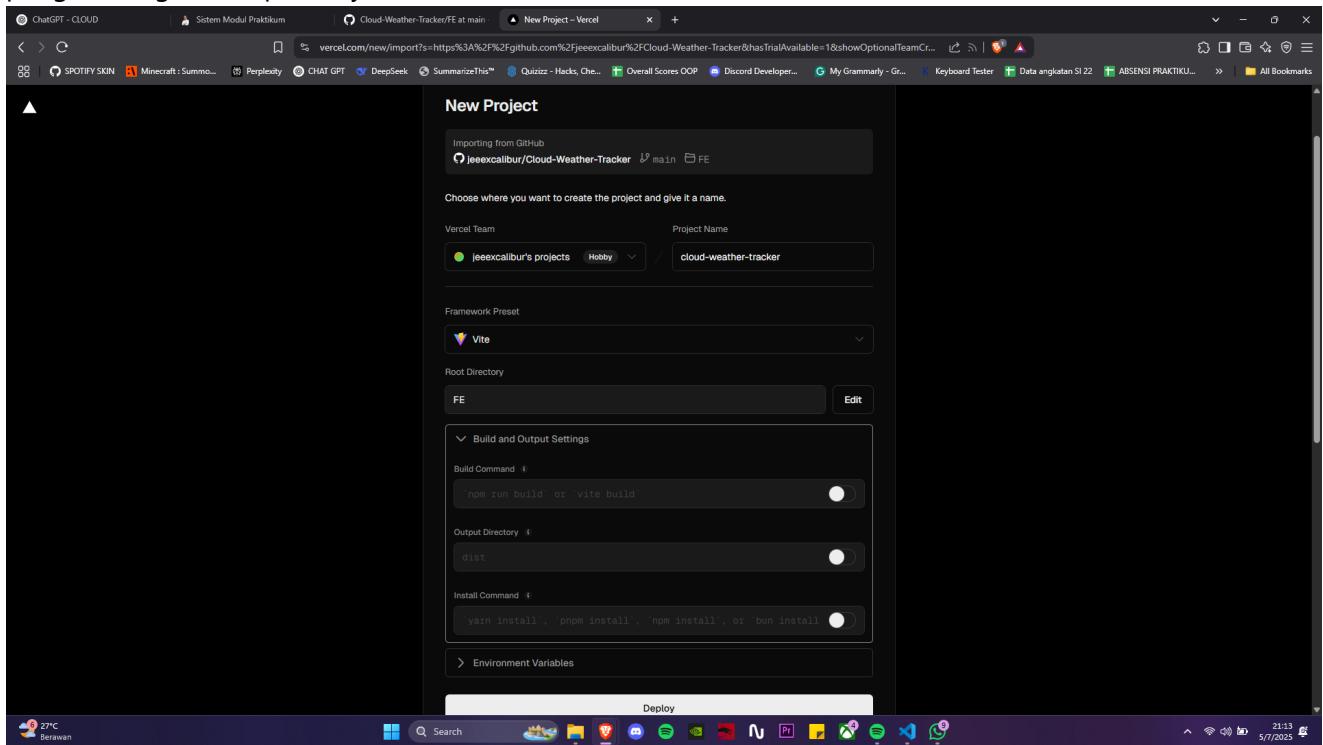
# Frontend

---

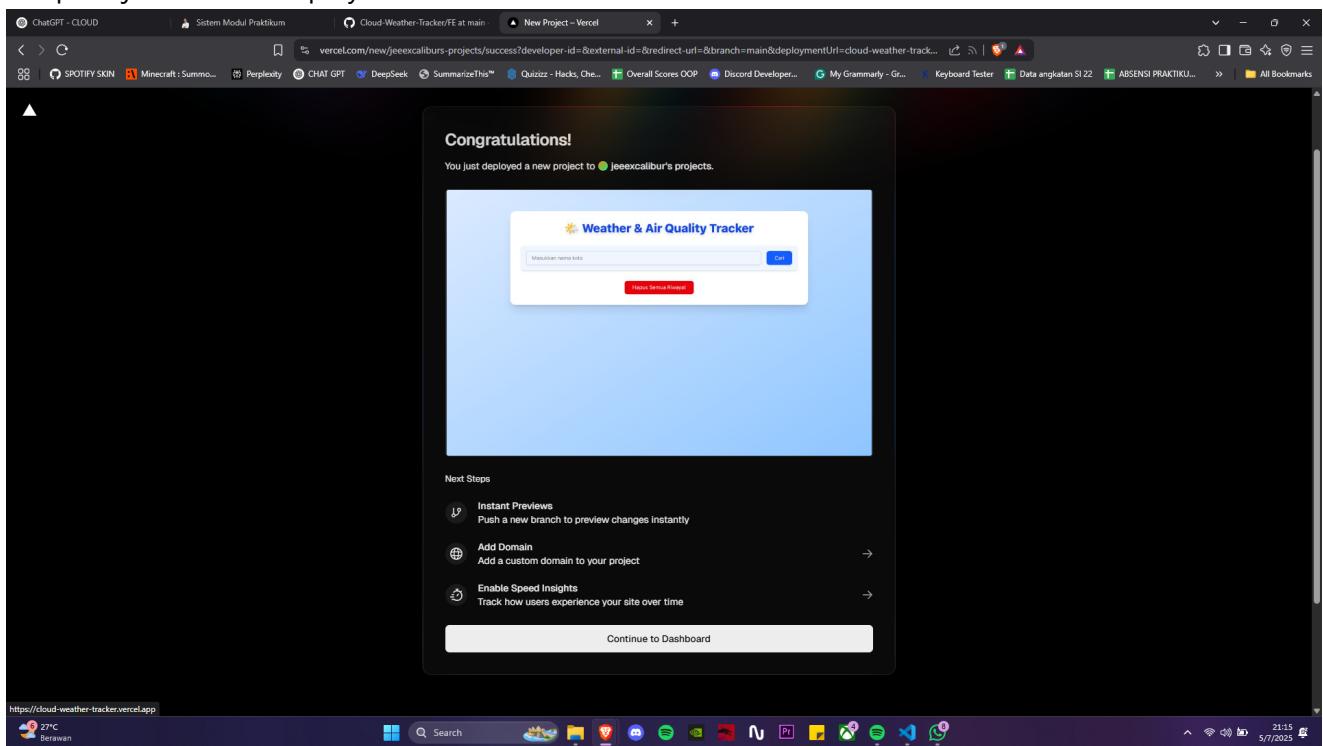
## 1. Langkah Pertama

Melakukan setup project baru di vercel, dan masuk ke pengaturan dimana source CI/CD menggunakan repository Cloud-Weather-Tracker sama halnya dengan backend sebelumnya akan terus otomatis terupdate setiap saya melakukan push atau update ke repository Cloud-Weather-Tracker, untuk root berbeda dengan BACKEND dikarenakan vercel saya fokuskan untuk Frontend jadi saya menggunakan folder FE/ (frontend) sebagai root deployment. , sesuai dengan standar

pengembangan di repository ini.

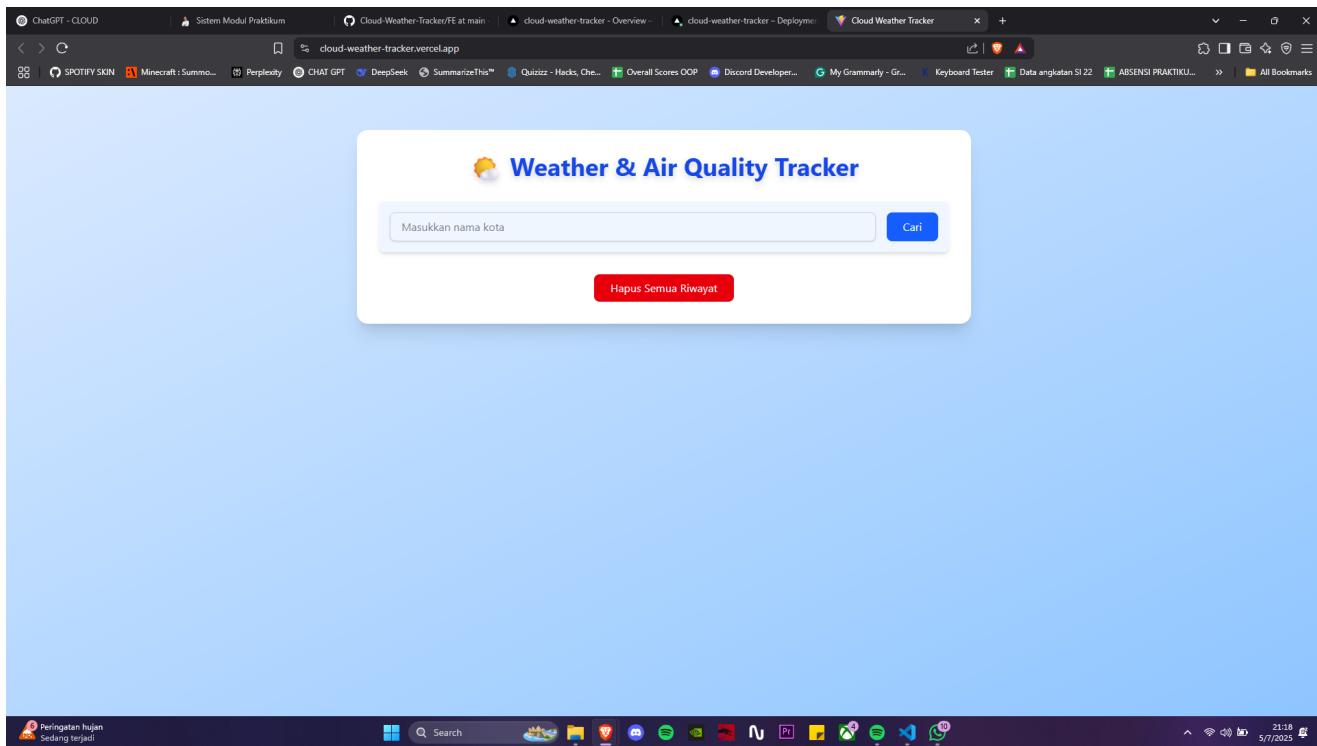


Tampilan jika berhasil deploy



# Testing

Landing Page



## Search

A screenshot of the 'Weather &amp; Air Quality Tracker' website showing search results for 'Balikpapan'. The title 'Weather &amp; Air Quality Tracker' is at the top, followed by a search bar with 'Balikpapan' and a blue 'Cari' button. Below the search bar is a weather summary for 'Cuaca di Balikpapan': '26.99°C | 83% kelembaban' and 'Overcast Clouds'. Under 'Kualitas Udara', it shows AQI: 1 - Baik, PM2.5: 2.13 µg/m³, CO: 88.54 µg/m³, NO₂: 0.22 µg/m³, O₃: 9.77 µg/m³, SO₂: 0.08 µg/m³, and PM10: 3.01 µg/m³. A bar chart titled 'Grafik Kualitas Udara' shows air quality levels in µg/m³ for various pollutants. The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating the date and time.

## History

The screenshot shows a web browser window with the title "Cloud Weather Tracker". The main content area displays a weather tracking application with the following interface:

- Header:** "Weather & Air Quality Tracker" with a sun icon.
- Search Bar:** "Masukkan nama kota" (Enter city name) with a "Cari" (Search) button.
- Table:** "Riwayat Pencarian Cuaca" (Weather Search History) showing the following data:

Kota	Suhu (°C)	Kelembapan	Deskripsi	Waktu	Aksi
Balikpapan	26.99	83%	overcast clouds	5/7/2025, 9:21:50 PM	<button>Hapus</button>
- Buttons:** "Hapus Semua Riwayat" (Delete All History) and "Hapus" (Delete) for individual rows.

At the bottom of the screen, there is a Windows taskbar with the following icons and information:

- System tray icons: battery level, signal strength, volume, and date/time.
- Taskbar items: Search, File Explorer, Edge browser, Task View, and various pinned application icons.
- Date and time: "21:22" and "5/7/2025".

## Database

Postgres | Railway - Brave

raillery.com/project/efe0e2d7-c97e-4576-bb72-b55313cbee7c/service/96c1a100-5dd4-4ef1-b635-7fb0001ea06/data?environmentId=45ae9c53-984f-41cc-94fa-5ac0cc30a0fe&state=table&table=weather\_data

independent-alignment / production

Architecture Observability Logs Settings TRIAL \$ 5.00 Share

**Postgres**

Deployments Data Backups Variables Metrics Settings

Connect to the database Postgres Connect

**Postgres**

2 hours ago via Docker Image

shoe-volume

**Cloud-Weather-Tracker** cloud-weather-tracker-prod...

10 minutes ago via GitHub

weather\_data

ID	city	temperature	humidity	description	created_at
8	Balikpapan	26.99	83	overcast clouds	2025-05-07 21:21:50

Add row

Page 1 of 1

Postgres

Deployments Data Backups Variables Metrics Settings

Connect to the database Postgres Connect

**air\_quality\_data**

ID	city	aqi	pm2_5	pm10	co	no2	o3	so2	created_at
2	Balikpapan	1	2.05	2.71	93.18	0.22	12.53	0.09	2025-05-07 20:22:30
3	Balikpapan	1	2.05	2.71	93.18	0.22	12.53	0.09	2025-05-07 20:22:33
4	Makassar	1	1.62	2.42	119.89	0.5	34.27	0.07	2025-05-07 20:22:52
5	Makassar	1	1.62	2.42	119.89	0.5	34.27	0.07	2025-05-07 20:22:54
6	Tokyo	3	17.24	20.23	190.03	9.59	122.94	8.65	2025-05-07 20:23:35
7	Balikpapan	1	2.13	3.01	88.54	0.22	9.77	0.08	2025-05-07 20:58:11
8	Balikpapan	1	2.13	3.01	88.54	0.22	9.77	0.08	2025-05-07 21:21:50

Add row

Page 1 of 1