🌍 Air & Weather Quality Monitor: Automated ETL Pipeline

Author: Suhas Nandihalli Umesh & Abhimanyu Kanase

Date: February 2026
License: MIT License

📖 Project Overview

This project is a **Real-Time Data Engineering Pipeline** designed to monitor air quality and weather conditions across multiple global cities. It automates the process of fetching, storing, and visualizing environmental data without manual intervention.

The system monitors **12 major cities** across Germany and India and visualizes the data on an interactive dashboard.

Key Features

- **Automated Ingestion:** A "robot" runs every hour to fetch fresh data.
- **Cloud Architecture:** Uses GitHub Actions for compute and Supabase (PostgreSQL) for storage.
- **Geospatial Visualization:** An interactive map showing live pollution levels.
- **Decoupled System:** The data collection (ETL) and visualization (Dashboard) are completely separate, ensuring high availability.

1. System Architecture

The project follows a modern **ETL (Extract, Transform, Load)** architecture:

1. **EXTRACT:** Python script queries the **OpenWeatherMap API**.
2. **TRANSFORM:** Data is cleaned and structured (extracting PM2.5, AQI, Temperature).
3. **LOAD:** Data is securely inserted into a **Supabase (PostgreSQL)** cloud database.
4. **AUTOMATION: GitHub Actions** triggers the script every hour (`0 * * * *`).
5. **VISUALIZATION:** A **Streamlit** dashboard reads from the database to display trends and maps.

2. Data Source: The API

The system relies on the **OpenWeatherMap API** for real-time data.

- **Endpoint 1 (Pollution):** `/data/2.5/air_pollution`
  - *Returns:* AQI (1-5 scale), PM2.5, PM10, CO, NO2, etc.
- **Endpoint 2 (Weather):** `/data/2.5/weather`
  - *Returns:* Temperature (°C), Humidity, Wind Speed.

Security: The API Key is never exposed in the code. It is injected dynamically using **GitHub Secrets** (`os.getenv('API_KEY')`).

⚙️ The ETL Script (`fetch_data.py`)

The core logic resides in `fetch_data.py`. This script performs three main functions:

1. Database Connection & Schema Check

Before fetching data, the script connects to Supabase and checks if the storage table exists. If not, it creates it automatically:

```sql
CREATE TABLE IF NOT EXISTS air_quality_logs (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    city VARCHAR(50),
    aqi INTEGER,
    pm2_5 FLOAT,
    temperature FLOAT
);
```

2. Data Fetching Loop

The script iterates through a predefined list of cities (defined with Latitude/Longitude coordinates) and sends requests to the API.

3. Error Handling

- If the API fails for one city, the script logs the error and continues to the next city (it does not crash).
- If the Database connection fails, the script performs a "Safety Exit" to prevent data corruption.

🤖 Automation: GitHub Actions

To ensure the data is always fresh, the project uses **GitHub Actions** as a scheduler.

- **File:** `.github/workflows/hourly_scheduler.yml`
- **Schedule:** Cron Syntax `0 * * * *` (Runs at minute 0 of every hour).
- **Workflow Steps:**
    1. Spin up an `ubuntu-latest` virtual machine.
    2. Install Python 3.9.
    3. Install dependencies (`requests`, `psycopg2`, `pandas`).
    4. Inject Secrets (API Key & DB URI).
    5. Execute `python fetch_data.py`.

This setup essentially gives the project a **"Serverless"** backend—it runs on the cloud for free without needing a dedicated server running 24/7.

📊 The Dashboard (`dashboard.py`)

The frontend is built using **Streamlit** and **Plotly**, providing a user-friendly interface for analysis.

Interactive Features

- **Live Map:** A geospatial view where dot size represents pollution intensity (PM2.5) and color represents AQI.
- **City Filter:** Users can select specific cities (e.g., "Bengaluru") to drill down into the data.
- **KPI Metrics:** Displays current AQI, PM2.5 concentration ($\mu g/m^3$), and the last update timestamp.
- **Trend Analysis:** Line charts show how pollution levels have changed over the last 24+ hours.

🚀 How to Run Locally

If you wish to run this project on your local machine instead of the cloud, follow these steps:

1. Prerequisites

- Python 3.10+
- PostgreSQL / Supabase Account
- OpenWeatherMap API Key

2. Installation

- Clone the repository and enter the directory:

Bash

- git clone [https://github.com/suhasnu/Air-and-weather-quality-monitor.git] (https://github.com/suhasnu/Air-and-weather-quality-monitor.git)
- cd Air-and-weather-quality-monitor

Install the required Python libraries:

Bash

- pip install -r requirements.txt

3. Configure Secrets (Locally)

- Create a .env file in the root directory to store your keys safely:

```
Ini, TOML
API_KEY=your_api_key_here
DB_URI=your_postgres_connection_string
```

4. Run the Dashboard

- Launch the Streamlit app:

Bash

- ```
  streamlit run dashboard.py
  ```

Bash

- ```
  streamlit run dashboard.py
  ```