Weather Data Analyzer Service

Automated Historical Weather Analysis & Reporting

Weather Project Team

December 1, 2025

Project Goal

Objective:
To build a tool that analyzes historical weather data for any given city and generates actionable insights.

Key Requirements:

- Fetch weather data for the last 7 days.
- Analyze key metrics: Temperature, Humidity, Wind Speed, and Precipitation.
- Calculate daily trends (e.g., temp change vs. previous day).
- Export results to CSV.
- **Automation:** Containerize the solution using Docker.
- **Notification:** Send reports directly to Telegram.

Solution Architecture

Data Flow Diagram:

1. **Input:** User provides a city name (e.g., "Kyiv").
2. **Acquisition:** Script queries **Open-Meteo API** (Geocoding & Archive).
3. **Processing:** **Pandas** cleans and aggregates hourly data into daily summaries.
4. **Visualization:** **Matplotlib** generates a combo chart (Temp + Rain).
5. **Delivery:**
   - CSV File saved locally via Docker Volumes.
   - Report & Image sent to user via **Telegram Bot API**.

Technology Stack

Core Language:

- Python 3.9

Libraries:

- **Pandas:** Data manipulation.
- **Matplotlib:** Visualization.
- **Requests:** HTTP calls.

Infrastructure:

- **Docker:** Containerization.
- **Docker Compose:** Orchestration.

APIs:

- Open-Meteo
- Telegram Bot API

Key Feature - Data Analysis

Smart Aggregation:
Raw hourly data (168 rows) $\rightarrow$ Daily summaries (7 rows).

Aggregation Logic:

- **Temperature/Wind:** Calculated as *Mean*.
- **Precipitation:** Calculated as *Total Sum*.

Trend Calculation:

$$Trend = T_{today} - T_{yesterday}$$

Example Output: `+2.68°C` (Warming) or `-1.5°C` (Cooling).

Key Feature - Automation & Security

Docker Integration:

- Isolated environment (`python:3.9-slim`).
- **Volume Mapping:** Automatically saves results to the host machine.
- Auto-removal (`--rm`) ensures cleanliness.

Security Best Practices:

- Sensitive data (`TELEGRAM_TOKEN`, `CHAT_ID`) are managed via **Environment Variables** in `docker-compose.yml`.
- Keys are not hardcoded in the Python script.

Visualization

Dual-Axis Charting Strategy:

The script generates a PNG image combining two metrics:

Left Axis (Bars):

- Precipitation (mm).
- Color: Blue.

Right Axis (Line):

- Temperature (°C).
- Color: Red.

Value: Allows quick correlation analysis (e.g., "Did temperature drop after the rain?").

Live Demo Results

Success Criteria Met:

- ✓ **Console Output:** Clean ASCII table with daily stats.
- ✓ **Files Generated:**
  - `City_weather_report.csv` (Excel-ready).
  - `City_weather_chart.png`.
- ✓ **Telegram Notification:** Instant push notification with the chart and summary text delivered to the smartphone.

Future Scope

Planned Improvements:

1. **Database Integration:** Store history in PostgreSQL to avoid re-fetching data.
2. **Scheduling:** Use `cron` to send daily morning reports automatically (e.g., at 8:00 AM).
3. **Interactive Bot:** Upgrade to a listening bot that accepts city names from users dynamically (24/7 uptime).

Thank You!

Any questions?

Weather Data Analyzer Project