Real-Time Weather Monitoring System

Gyeongwon Song
Student ID: 22297

Overview

Introduction

This project builds a **real-time weather monitoring system** using **Kafka**, **Spark Streaming**, **MySQL**, and **Streamlit**. It demonstrates expertise in data pipelines, real-time processing, and UI development. The goal is to provide a **scalable** solution for collecting and visualizing live weather data from multiple cities in an **interactive format**.

Application Domain

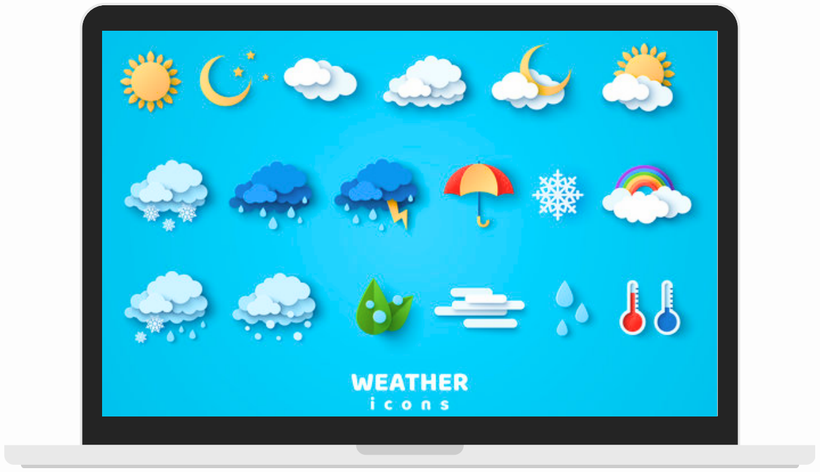
- Real-time weather monitoring and analytics system
- Provides live weather data from 20 major cities worldwide

Weather Information Services

Accurate and up-to-date weather updates for individuals and businesses

Smart Cities

Adaptive traffic control and public safety alerts based on weather conditions



Travel and Tourism

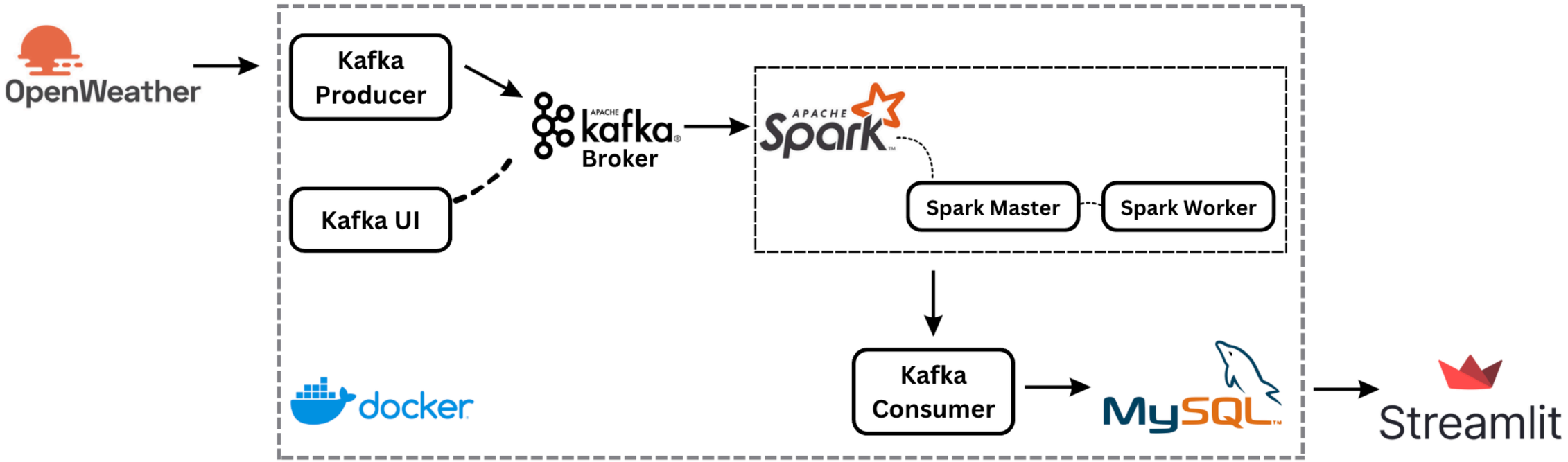
Provides real-time weather insights for better travel planning

Environmental Monitoring

Supports climate change studies and environmental analysis

System Architecture

Data Collection **Real-Time Data Processing** **User Interface**



Containerization **Data Storage**

Data Source

OpenWeather API

The weather data is accessed through the OpenWeather API via REST API calls, which require authentication using an API key. This API offers a range of features, including the "One Call API," which delivers structured JSON responses.

- **City Name**
- **Country Code**
- **Temperature**
- **Humidity**
- **Timestamp**
- **Feels Like**
- **Min/Max Temperature**
- **Pressure**
- **Visibility**
- **Wind Speed**
- **Wind Direction (Wind Degree)**
- **Rain (Rain 1h)**
- **Cloud Coverage**
- **Description**
- **Icon**

Data Processing Workflow

Data Ingestion

- **Frequency**: Weather data is fetched from the OpenWeather API every 5 minutes.
- **Method**: The data is sent to the Kafka topic named weatherDetails.

Stream Processing

- Subscribes to the weatherDetails Kafka topic to consume incoming data.
- Processes the data in real-time for immediate analysis.

Metric Calculations

Computes important weather metrics:

- **Heat Index**: Reflects how hot it feels considering humidity.
- **Dew Point**: Indicates moisture content in the air.
- **Wind Chill**: Represents how cold it feels due to wind.

Data Processing Workflow

Metric Calculations

Computes important weather metrics:

- **Heat Index**: Reflects how hot it feels considering humidity.
- **Dew Point**: Indicates moisture content in the air.
- **Wind Chill**: Represents how cold it feels due to wind.

Heat Index (HI):

$$HI = 0.5 \times (T + 61.0 + ((T - 68.0) \times 1.2) + (H \times 0.094))$$

- where T is the **temperature** and H is the **humidity** (%)

Dew Point:

$$\alpha = \frac{a \times T}{b + T} + \ln\left(\frac{H}{100}\right)$$

- where T is the **temperature**, H is the **humidity**, a = 17.27, and b =243.12.

Wind Chill:

$$WindChill = 13.12 + 0.6215 \times T - 11.37 \times (V^{0.16}) + 0.3965 \times T \times (V^{0.16})$$

- where T is the **temperature** and V is the **wind speed** (km/h).

Data Processing Workflow

Data Publishing

Processed metrics are published to two separate Kafka topics:

- **metricsWeather**: For real-time calculated metrics.
- **aggregatedWeather**: For hourly aggregated data.

Data Storage

MySQL Database:

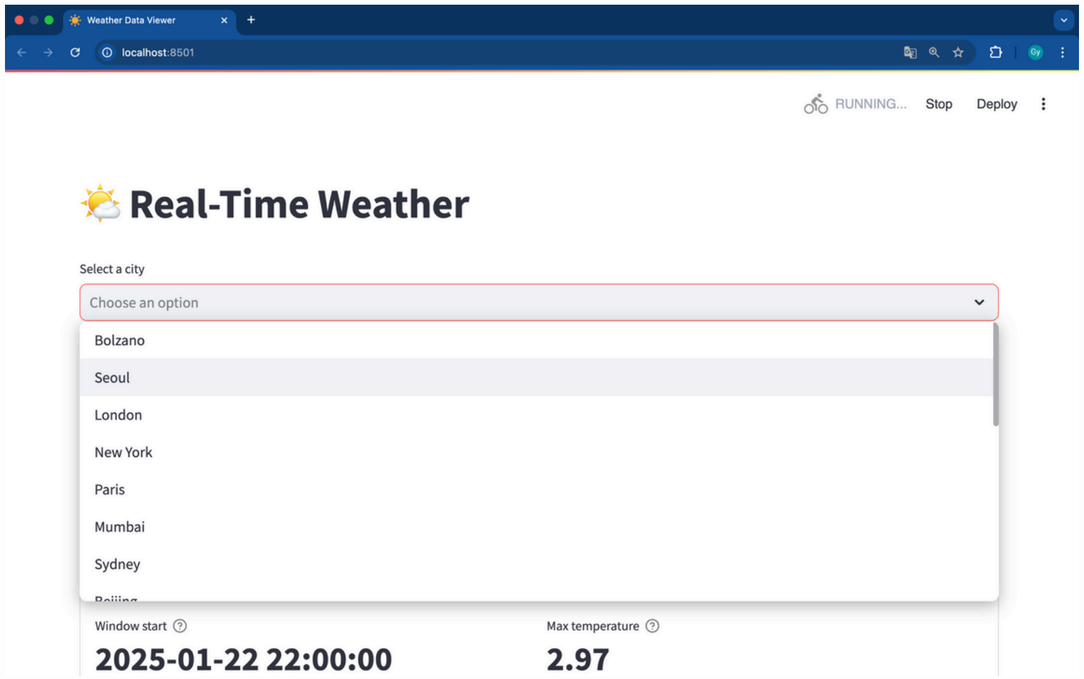
- Raw and processed data are stored for historical analysis.

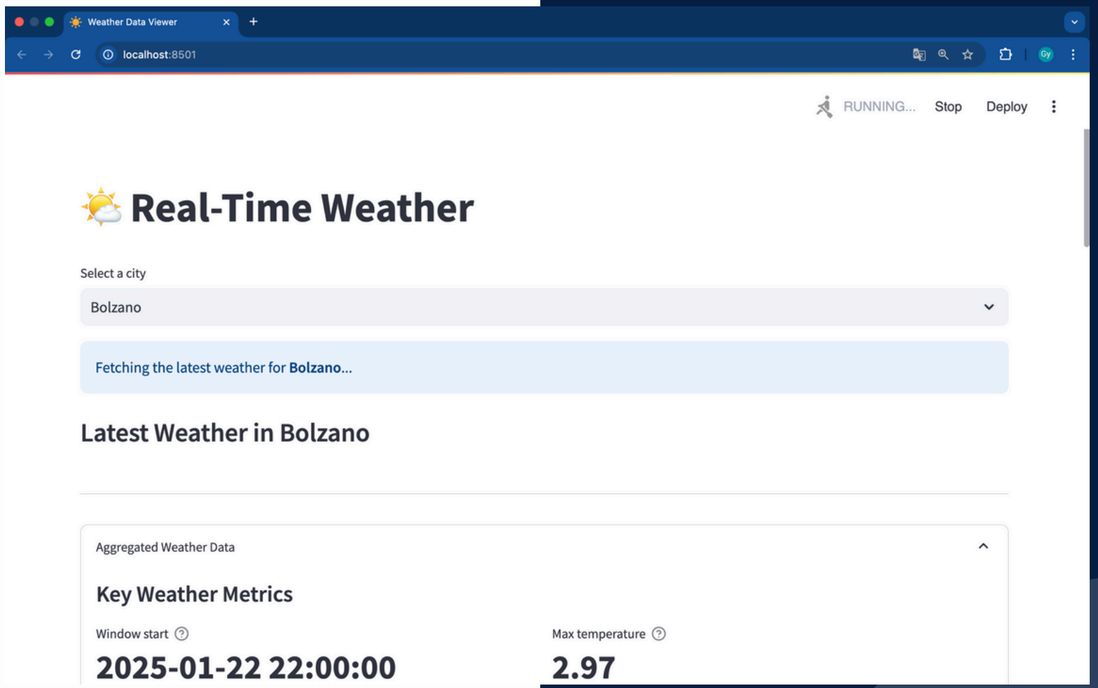
Visualization

Streamlit:

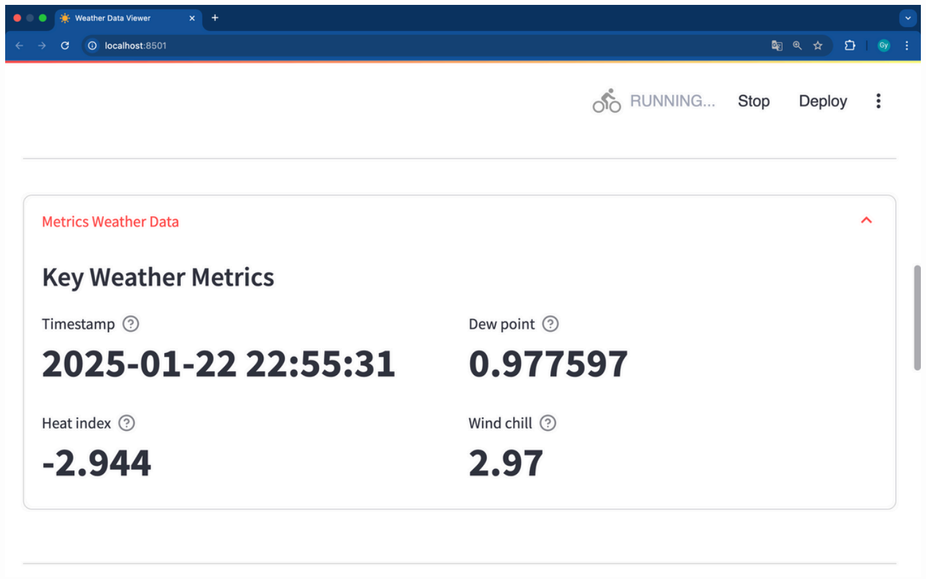
- Fetches data from the MySQL database.
- Dashboard updates automatically every 3 minutes for real-time insights.

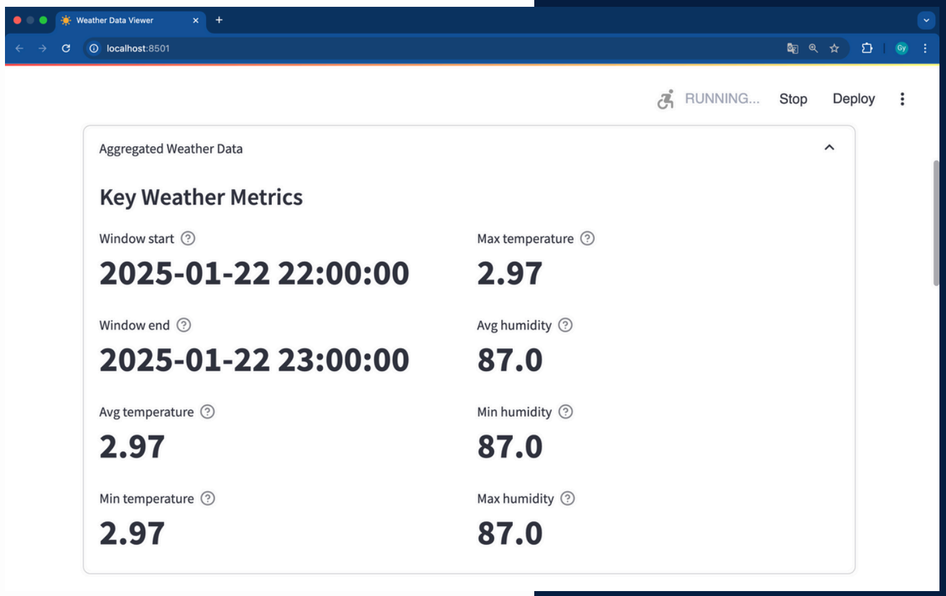
User Interface & Demo

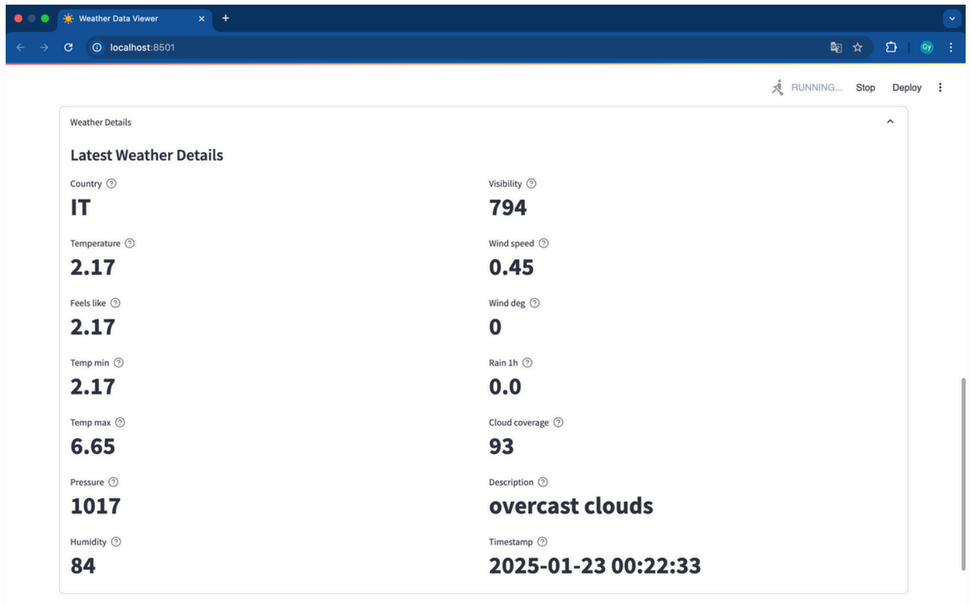




User Interface & Demo







Challenges and Solutions

Kafka Configuration Errors

- **Challenge**: Configuration issues delayed setup.
- **Solution**: reviewed Kafka's configuration, optimized partitioning, and tested the setup.

Scalability Issues

- **Challenge**: Handling high data volumes across multiple cities
- **Solution**: Optimized batch intervals, used window functions for aggregation, and scaled resources.

Front-End Development Expertise

- **Challenge**: Delays due to limited expertise in complex front-end technologies.
- **Solution**: Shifted to Streamlit for an easier development process.

Challenges and Solutions

Database Selection

- **Challenge**: Problems with PostgreSQL installation.
- **Solution**: Switched to MySQL for stable data storage.

Time Management

- **Challenge**: Need for time to learn new technologies.
- **Solution**: Allocated specific time for research and skills improvement.

Conclusion

01

Successful Integration of Real-Time Technologies

Successfully integrated technologies like Kafka, Spark, MySQL, and Streamlit to create a scalable weather monitoring system.

02

Impactful Insights Through Advanced Metrics

Provided useful metrics like heat index, dew point, and wind chill for enhanced weather analysis.

03

User-Friendly and Interactive Design

Delivered an intuitive Streamlit interface for real-time and aggregated weather data across 20 cities.

04

Lessons in Adaptability and Scalability

Overcame challenges like front-end limitations and scalability through solutions like MySQL and Docker.

Q&A

Thank you for your attention!