Faculty of Information Technology
Department of Electrical Engineering and Computer Science

Real-Time Weather Data Pipeline Using Airflow, Kafka, and SQLite

Done by: Suanbekova Aisha [22B030589]
Stelmakh Polina [22B030588]
Nursovet Iman [22B030416]

Almaty 2025.

1 Project Goal

The objective of this project is to demonstrate the ability to collect, clean, store, process, and analyze real-world frequently updating data using a hybrid **streaming and batch data pipeline**.

The system is implemented using **Apache Airflow**, **Apache Kafka**, **Docker**, and **SQLite**, and consists of three Airflow DAGs:

- Continuous ingestion of weather data from a public API into Kafka
- Hourly batch cleaning and storage into SQLite
- Daily analytics and aggregation job

2 API Selection and Justification

The selected data source is the **OpenWeather API**, which provides real-time weather information for cities worldwide.

API:

`https://api.openweathermap.org`

Justification

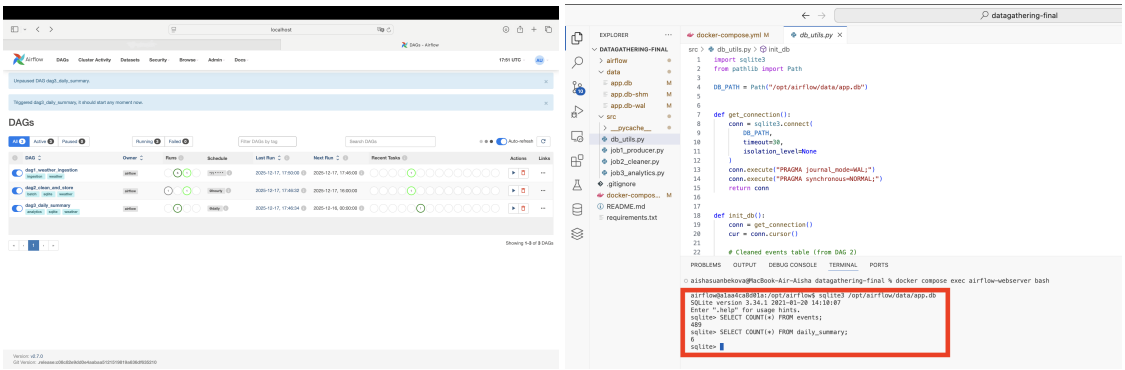
- Frequently updated (every few minutes)
- Stable and well-documented REST API
- Structured JSON responses
- Provides meaningful real-world measurements
- Not used in previous practice sessions

Data was collected for the following cities: *Almaty, Astana, Shymkent, Aktobe, Karaganda, Oskemen.*

3 System Architecture Overview

The system is deployed using **Docker Compose**, with all components running in separate containers.

API → Airflow DAG 1 → Kafka → Airflow DAG 2 → SQLite → Airflow DAG 3 → SQLite Summary



4 Docker-Based Deployment

The entire infrastructure is orchestrated using `docker-compose.yml`. Each major system component runs in an isolated container.

Services

- **Zookeeper**: Coordinates Kafka brokers
- **Kafka**: Message broker for streaming weather data
- **SQLite**: Stores Airflow metadata (DAG states, logs)
- **Airflow Webserver**: Web UI and DAG execution
- **Airflow Scheduler**: Schedules and triggers DAGs

Networking

Kafka is advertised internally as `kafka:9092`, allowing Airflow containers to communicate with Kafka reliably inside the Docker network. External access is provided via `localhost:9092` for local testing.

Dependency Management

Python dependencies (`requests`, `kafka-python`, `pandas`) are installed dynamically inside Airflow containers using the `_PIP_ADDITIONAL_REQUIREMENTS` environment variable.

5 DAG 1: Continuous Weather Data Ingestion

Schedule: Every 1 minute

This DAG fetches real-time weather data from the OpenWeather API and publishes raw JSON messages to the Kafka topic `raw_weather`.

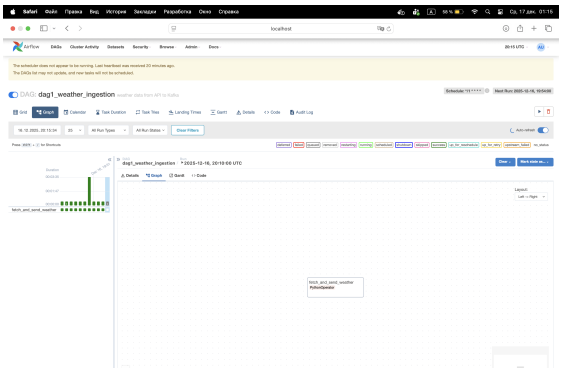
Process

- Fetch weather data for multiple cities
- Append ingestion timestamp
- Publish messages to Kafka

Kafka Topic Schema (Raw)

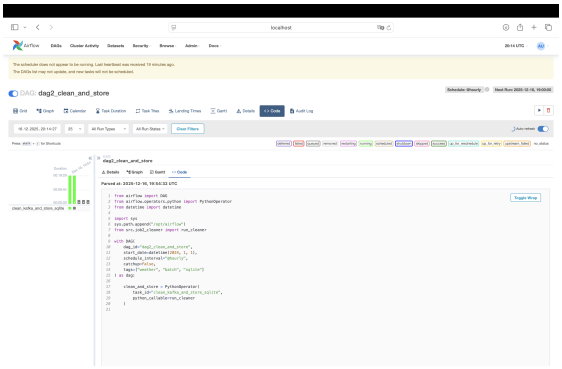
- City name
- Temperature (°C)
- Humidity
- Pressure

- Wind speed
- Weather description
- API event timestamp
- Ingestion timestamp



6 DAG 2: Hourly Cleaning and Storage Job

Schedule: @hourly



This DAG consumes messages from Kafka, applies cleaning rules, and stores cleaned records into SQLite.

Cleaning Rules

- Type casting for numerical fields

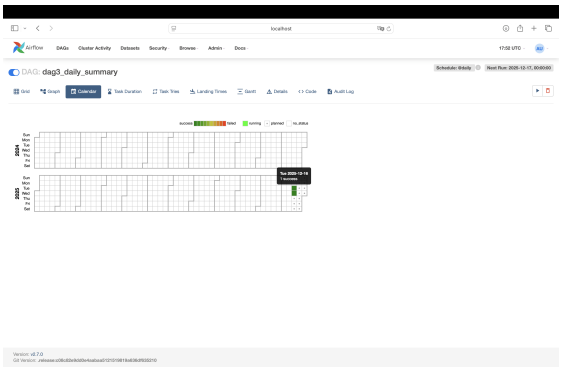
- Handling missing wind speed values
- Filtering malformed messages
- Timestamp normalization

SQLite Events Table

Column	Type
id	INTEGER (PK)
city	TEXT
temperature	REAL
humidity	INTEGER
pressure	INTEGER
wind_speed	REAL
weather_main	TEXT
weather_desc	TEXT
event_time	TEXT
ingestion_time	TEXT

7 DAG 3: Daily Analytics Job

Schedule: @daily



This DAG computes daily aggregated statistics per city and writes the results to a separate summary table.

Computed Metrics

- Average, minimum, and maximum temperature
- Average humidity
- Average pressure
- Average wind speed
- Record count

Daily Summary Table

Column	Type
date	TEXT
city	TEXT
avg_temperature	REAL
min_temperature	REAL
max_temperature	REAL
avg_humidity	REAL
avg_pressure	REAL
avg_wind_speed	REAL
records_count	INTEGER

8 Conclusion

This project demonstrates a complete, containerized data pipeline combining real-time streaming and batch analytics. Docker ensures reproducibility, Kafka enables scalable streaming, Airflow provides orchestration, and SQLite supports lightweight analytics storage. The system reliably processes real-world weather data and produces meaningful daily summaries.