Week 5 Labs

ETL&ELT Processes and Automation with Airflow

Due Date: Monday, 2nd December 2024

Lab Exercise 1: Developing a Weather Data Pipeline Using Apache Airflow

Scenario

You are a data engineer at a weather analytics company tasked with monitoring weather conditions for specific cities. The marketing team requires daily updates of weather data, including temperature, wind speed, and humidity, stored in a local PostgreSQL database for further analysis and reporting.

The company's CTO wants an automated data pipeline using Apache Airflow that:

1. Checks if the weather API is online.
2. Fetches the current weather data for a specific city (e.g., Portland).
3. Transforms the data into a more readable format, converting temperatures to Fahrenheit and timestamps to local time.
4. Loads the transformed data into a PostgreSQL database.

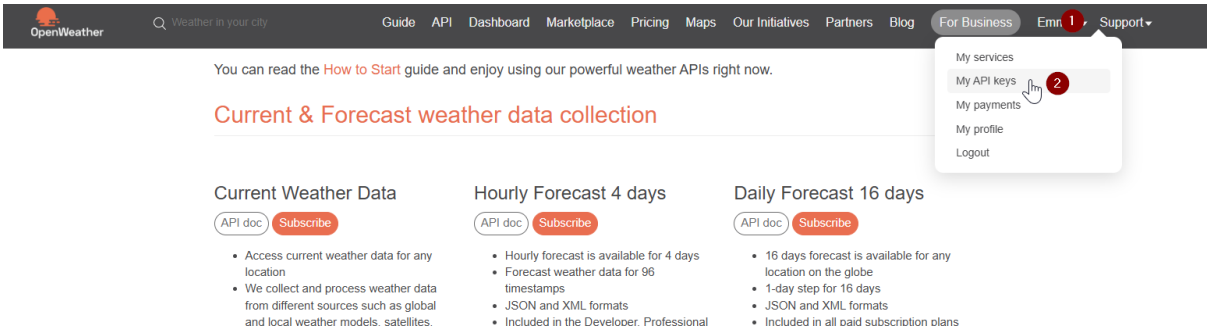
Deliverables (a git hub repo containing the following)

1. **Airflow DAG**: Implement a DAG with tasks for:
    - Checking the API readiness.
    - Extracting, transforming, and loading weather data into the PostgreSQL database.
2. **Database Schema**: A simple schema for storing weather data, including columns like city, temperature, pressure, humidity, and timestamps.
3. **Database Image**: Provide a screenshot or diagram of the PostgreSQL database schema after the pipeline is executed. Also include an image of at least on record in the database
4. **Airflow logs**: Attach images of DAG performance
5. **Visual representation of the implemented architecture** : Provide an architectural diagram image of the flow.

Instructions

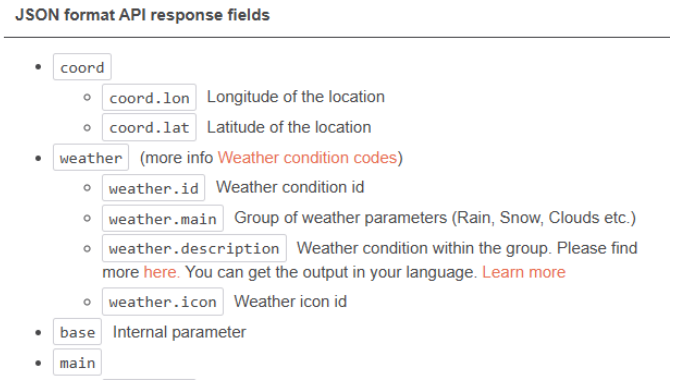
1. Use the provided weather API endpoint (/data/2.5/weather) for data extraction.

- Visit and create an account to access API : https://openweathermap.org/api



- Generate a new API Key.

- Check the API documentation
https://openweathermap.org/current



- USE the API Call below to return current weather information for **Portland**

https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}

NOTE: Newly generated API could take about 2hours to be active

2. Transform the weather data:
    - Convert temperatures from Kelvin to Fahrenheit.
    - **(Optional)** Adjust timestamps to the local timezone.

        Hint: use datetime.utcfromtimestamp to get parse the datetime.
        sunset time = sys.sunset + timezone
        sunrise time = sys.sunrise + timezone
        time of record = dt + timezone

3. Configure PostgreSQL locally:
    - Create a database named weather_data.
    - Use a table named daily_weather.

4. Load the transformed data into the daily_weather table.
5. Test your DAG to ensure it runs successfully and stores data in PostgreSQL.
6. **(Optional)** Set up an alert with EmailOperator

Hints

- Use the PostgresOperator or PostgresHook to connect to the database.
- Use operators like SimpleHttpOperator and HttpSensor
- Ensure the DAG uses XCom to pass data between tasks.
- Use Airflow's logging to monitor task outputs.