Data Engineering Project Report: Real-Time Car Park Analytics

Team Members:

1. Smetov Damir
2. Ravshanbekov Assadbek
3. Yerken Yarmukhamed

Date: December 19, 2025

Project Repository: https://github.com/DamirSmetov/Final_Project

---

1. API Justification

The project utilizes the **Transport for NSW Car Park API (v2.3)**. This data source was selected because it perfectly aligns with the project requirements:

- **Update Frequency:** Data is updated every 1–10 minutes (triggered by counter changes at parking facilities), which is ideal for demonstrating real-time streaming via Kafka.
- **Reliability:** The API is provided by the NSW Government, ensuring high uptime and professional documentation.
- **Format:** It returns structured JSON data, facilitating efficient parsing and cleaning.
- **Value:** It provides real-world occupancy metrics (Total spots vs. Occupied spots), allowing for meaningful urban traffic analytics.

2. Kafka Topic Schema

To bridge the gap between data ingestion (Job 1) and data processing (Job 2), we implemented a Kafka topic:

- **Topic Name:** raw_events
- **Message Format:** JSON
- **Key Fields:**
    - facility_id (Integer): Unique identifier for the parking facility.
    - tsn (String): Transit Stop Number.
    - total (Integer): Current number of occupied spaces.
    - spots (Integer): Total parking capacity.
    - MessageDate (String): Original timestamp from the API.

3. Data Cleaning Rules

In Job 2 (ETL process), the following cleaning and transformation rules are applied using the Pandas library:

1. **Type Validation:** All numerical values (total, spots) are explicitly cast to Integers to prevent calculation errors.
2. **Null Handling:** Any record containing null values in critical fields like facility_id or total is dropped.
3. **Feature Engineering:** A new field, available_spots, is derived using the logic: $Availability = spots - total$.
4. **Time Normalization:** The MessageDate field is converted to a standard ISO-8601 format to ensure consistency within the SQLite database.

4. SQLite Schema

Data is persisted in the app.db database within two specific tables:

Table 1: events (Cleaned Transactional Data)

- id: INTEGER PRIMARY KEY
- facility_id: INTEGER
- occupancy: INTEGER (Current vehicles)
- total_spots: INTEGER
- available_spots: INTEGER
- recorded_at: DATETIME

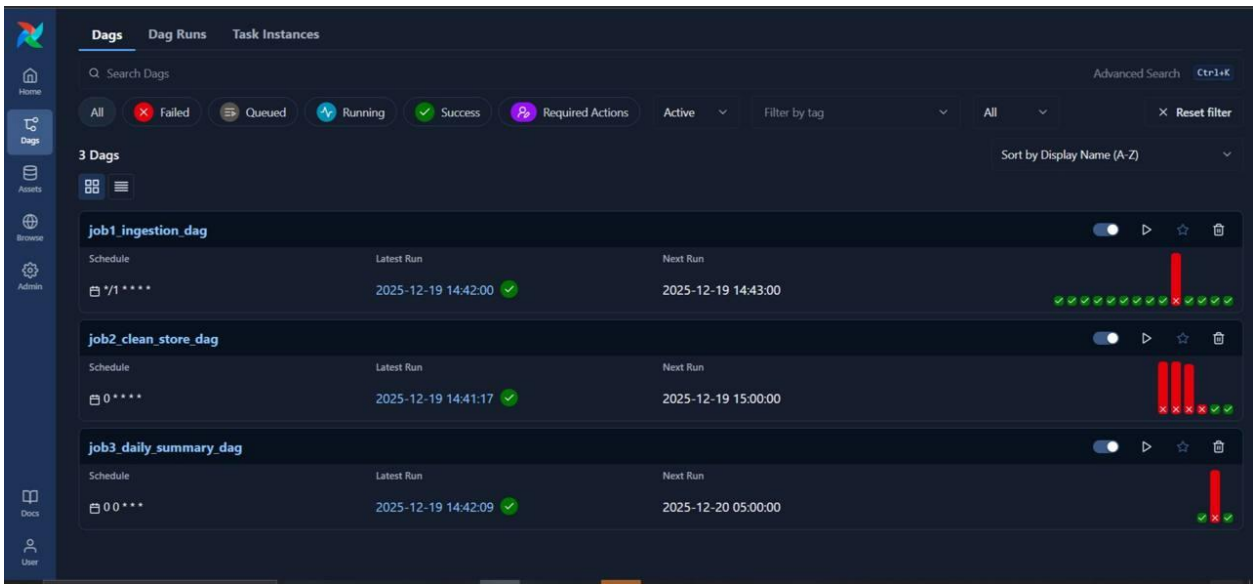
Table 2: daily_summary (Aggregated Analytical Data)

- id: INTEGER PRIMARY KEY
- date: DATE (Unique per day)
- facility_id: INTEGER
- avg_occupancy: FLOAT (Daily average)
- peak_occupancy: INTEGER (Max vehicles recorded)
- peak_hour: INTEGER (Hour of day with highest traffic)

---

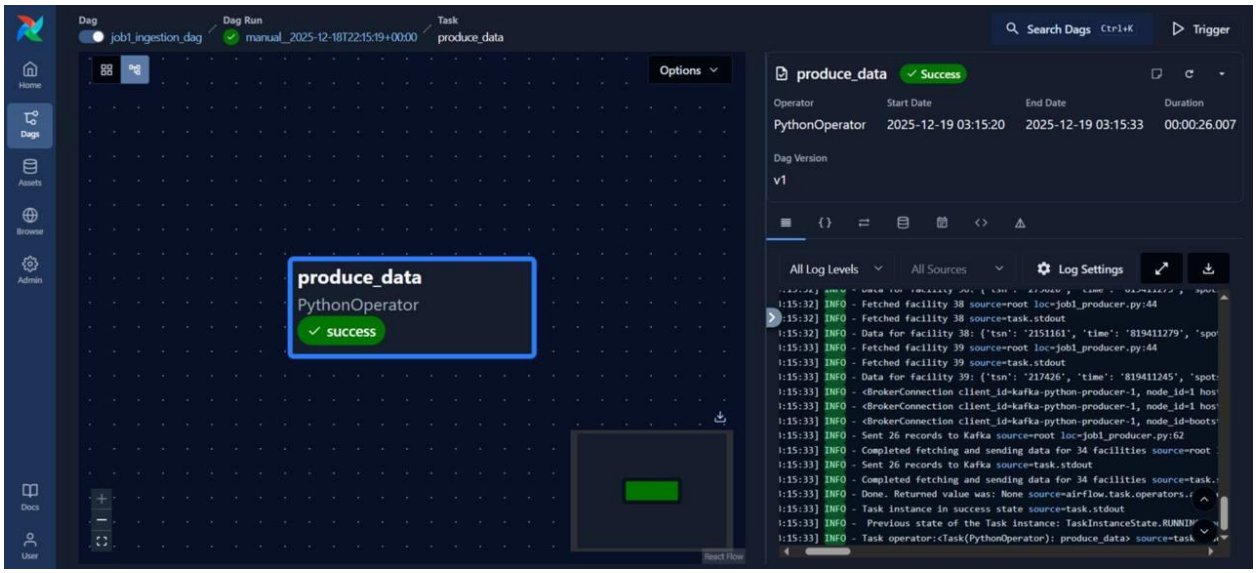
5. Implementation Evidence (Screenshots)

5.1. Airflow DAGs Overview



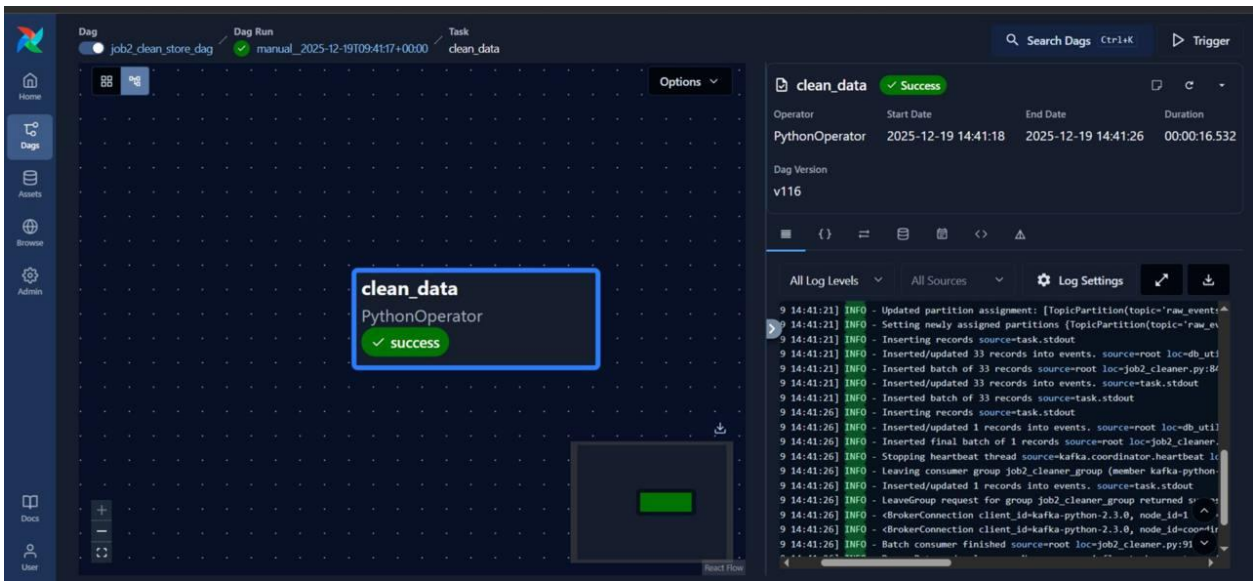
This proves all three DAGs are active: ingestion, cleaning, and analytics

5.2. Job 1: Ingestion Success (API to Kafka)



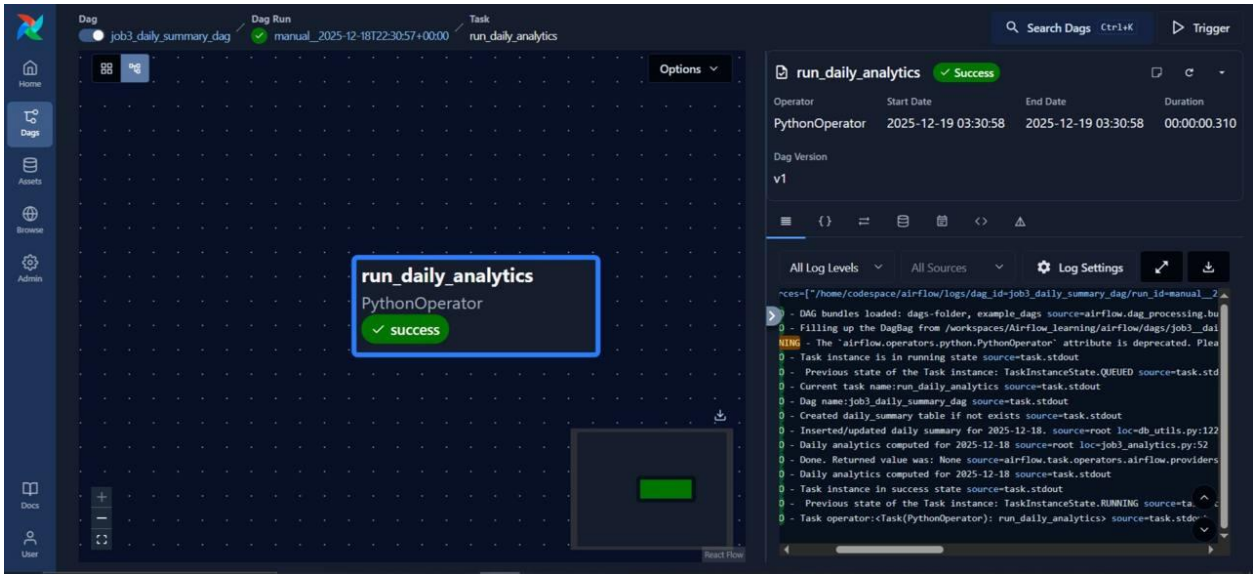
This shows the logs confirming the successful fetch from the TfNSW API

5.3. Job 2: Cleaning & Storage (Kafka to SQLite)



This shows the consumer logs and the successful insertion of rows into the database

5.4. Job 3: Daily Analytics Performance



This confirms the aggregation logic for the specific date, such as 2025-12-18

6. Conclusion

The pipeline successfully integrates real-time streaming (Kafka) and batch processing (Airflow). All components are fully synchronized and operational within the GitHub Codespaces environment. The project meets all technical criteria for high-frequency data ingestion and automated reporting.