Final Project Report

ElectricityMaps Pipeline (Kafka + Airflow + SQLite)

Team

Khan Kensey 22B030608

Abdullaev Shakhzod 22B031601

Temirlan Khamidulla 22B22B1553

Project Overview

This project implements a small data pipeline that ingests hourly electricity-related signals from the ElectricityMaps API for multiple zones, streams raw observations through Kafka, cleans and normalizes the data in batch with Airflow, and stores both cleaned events and daily aggregates in SQLite.

- Zones: TR, GB, FR (Turkey, United Kingdom, France)
- Signals: carbon_intensity, total_load, price_day_ahead
- Storage: SQLite tables **events** and **daily_summary**
- Orchestration: Apache Airflow (3 DAGs)

API Justification

We use the ElectricityMaps API because it provides near-real-time electricity metrics per geographic zone, including carbon intensity, total load, and day-ahead electricity price. These signals support time-series monitoring and daily analytics (min/max/avg and counts).

Endpoints (per zone):

- carbon intensity: /v3/carbon-intensity/latest
- total load: /v3/total-load/latest
- day-ahead price: /v3/price-day-ahead/latest
- All endpoints are called with zone=<ZONE> and temporalGranularity=hourly.
- Base url: https://api.electricitymaps.com

Kafka Topic Schema

- Topic name: raw_events
- Message format: JSON

Raw Kafka message (one observation):

```
{
    "zone": "FR",
    "signal": "price_day_ahead",
    "payload": {
      "zone": "FR",
      "datetime": "2025-12-16T18:00:00.000Z",
      "createdAt": "2025-12-15T12:21:17.136Z",
      "updatedAt": "2025-12-15T12:21:17.136Z",
      "value": 132.92,
      "unit": "EUR/MWh",
      "source": "nordpool.com",
      "temporalGranularity": "hourly"
    }
  }
```

Cleaning Rules (Job 2)

1. Normalize datetime to UTC and store as ISO-8601 string (YYYY-MM-DDTHH:MM:SSZ).
2. Convert extracted values to corresponding types.
3. Round **values** to 2 decimal places.
4. If unit is missing for **carbon_intensity**, set unit to gCO2eq/kWh.
5. If isEstimated is missing, set is_estimated=0; if estimationMethod is missing, set NOT_ESTIMATED.
6. Drop rows missing zone, datetime, signal, or value.
7. Deduplicate with UNIQUE(zone, datetime, signal) in SQLite.

SQLite Schema

'Events' table

Column	**Type**	**Constraints / Default**	**Meaning**
`zone`	TEXT	**NOT NULL**,	Zone code (e.g., `KZ`, `DE`, `FR`)
`datetime`	TEXT	**NOT NULL**	Hour timestamp (ISO, UTC) from API
`signal`	TEXT	**NOT NULL**,	`carbon_intensity`, `total_load`, `price_day_ahead`
`value`	REAL	**NOT NULL**	Numeric value for the signal
`unit`	TEXT	**NOT NULL**	Unit (e.g., `MW`, `EUR/MWh`, `gCO2eq/kWh`)
`is_estimated`	INTEGER	**NOT NULL**, `DEFAULT 0`,	0/1 flag
`estimation_method`	TEXT	**NOT NULL**, `DEFAULT 'NOT_ESTIMATED'`	Estimation method or `NOT_ESTIMATED`
`queried_at`	TEXT	**NOT NULL**	When the API data was last updated/created

Column	Type	Constraints / Default	Meaning
`ingested_at`	TEXT	**NOT NULL**, `DEFAULT CURRENT_TIMESTAMP`	When inserted into SQLite
`raw_json`	TEXT	**NOT NULL**	Full raw message JSON (string)

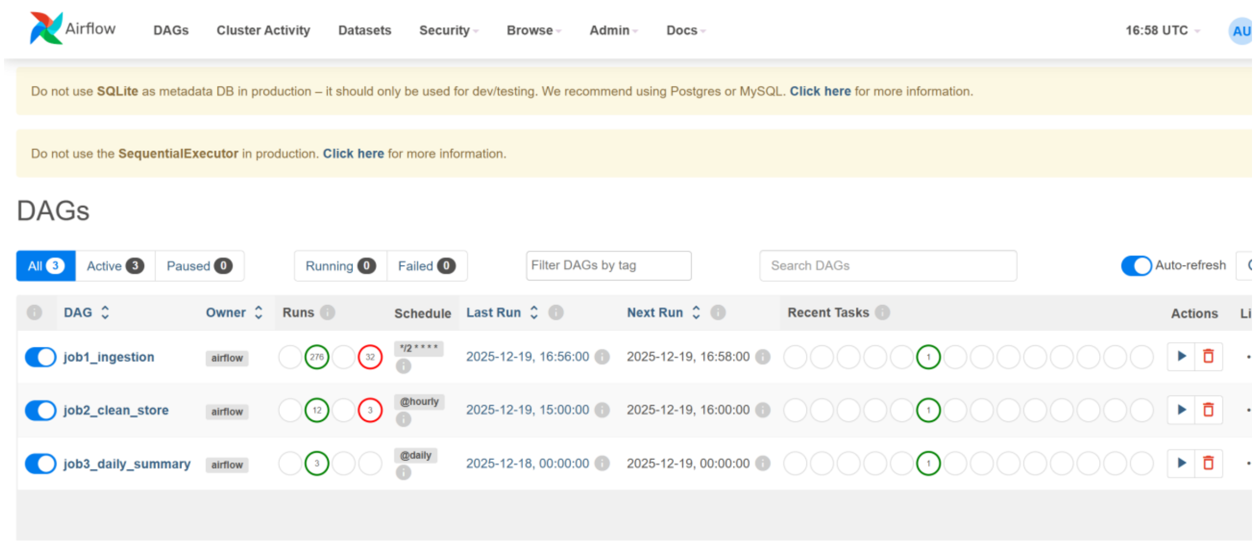
‘daily_summary’ table

Column	Type	Constraints / Default	Meaning
`day`	TEXT	**NOT NULL**	Date like `YYYY-MM-DD`
`zone`	TEXT	**NOT NULL**	Zone code
`signal`	TEXT	**NOT NULL**	Aggregated signal name
`count`	INTEGER	**NOT NULL**	Number of points aggregated (hours)
`min`	REAL	**NOT NULL**	Daily minimum
`max`	REAL	**NOT NULL**	Daily maximum
`avg`	REAL	**NOT NULL**	Daily average
`unit`	TEXT	**NOT NULL**	Unit for the signal
`computed_at`	TEXT	**NOT NULL**, `DEFAULT CURRENT_TIMESTAMP`	When this aggregate was computed

Airflow DAGs and Evidence

DAG 1 - Ingestion (API -> Kafka)

Schedule: every 2 minutes. (Doesn’t run continuously because we use SQLite as db, hence we can only use SequentialExecutor in airflow). Publishes raw messages to raw_events and dedupes per hour using job1_state.json.



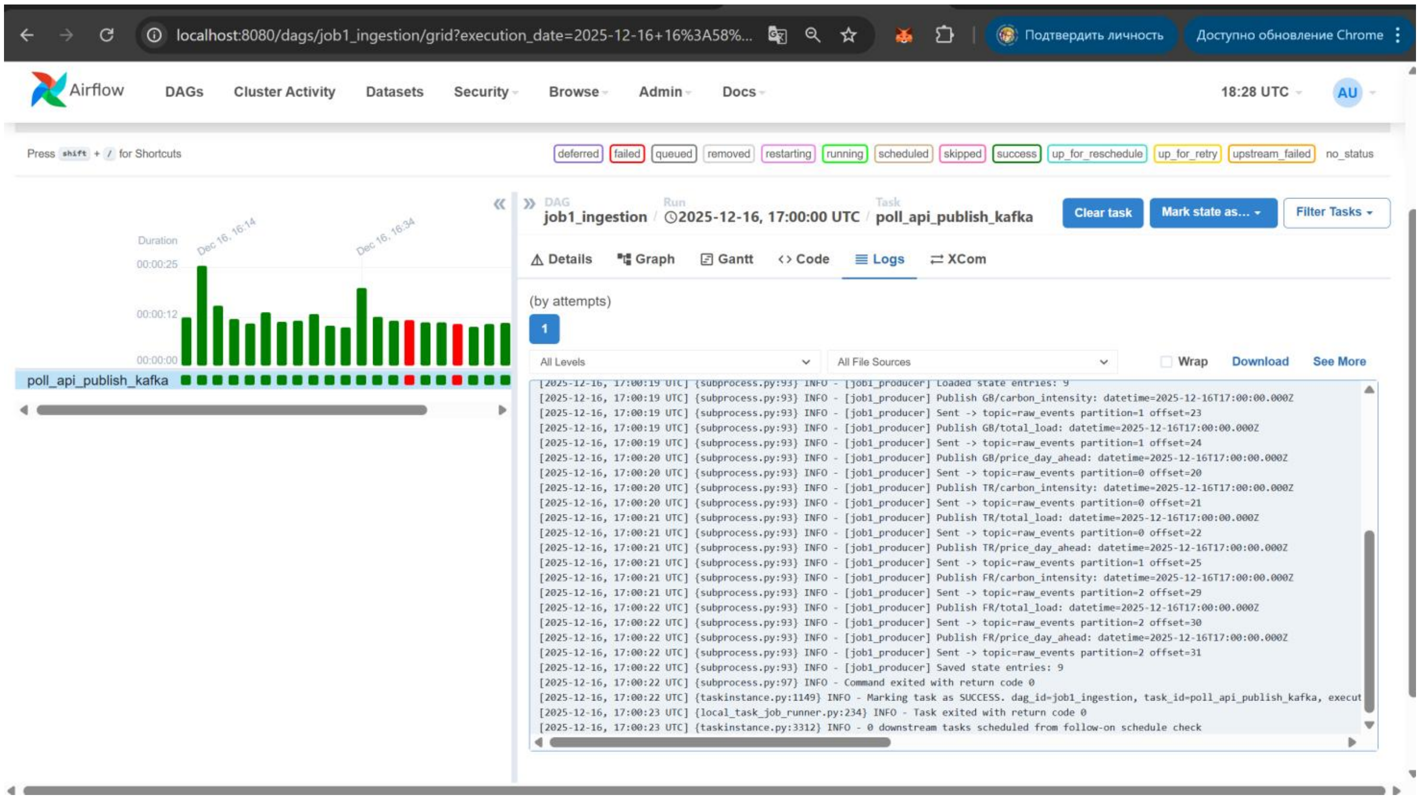
Do not use **SQLite** as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. **Click here** for more information.

Do not use the **SequentialExecutor** in production. **Click here** for more information.

DAGs

All 3 | Active 3 | Paused 0 | Running 0 | Failed 0 | Filter DAGs by tag | Search DAGs | Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
job1_ingestion	airflow	276, 32	*/2 * * * *	2025-12-19, 16:56:00	2025-12-19, 16:58:00	1	
job2_clean_store	airflow	12, 3	@hourly	2025-12-19, 15:00:00	2025-12-19, 16:00:00	1	
job3_daily_summary	airflow	3	@daily	2025-12-18, 00:00:00	2025-12-19, 00:00:00	1	

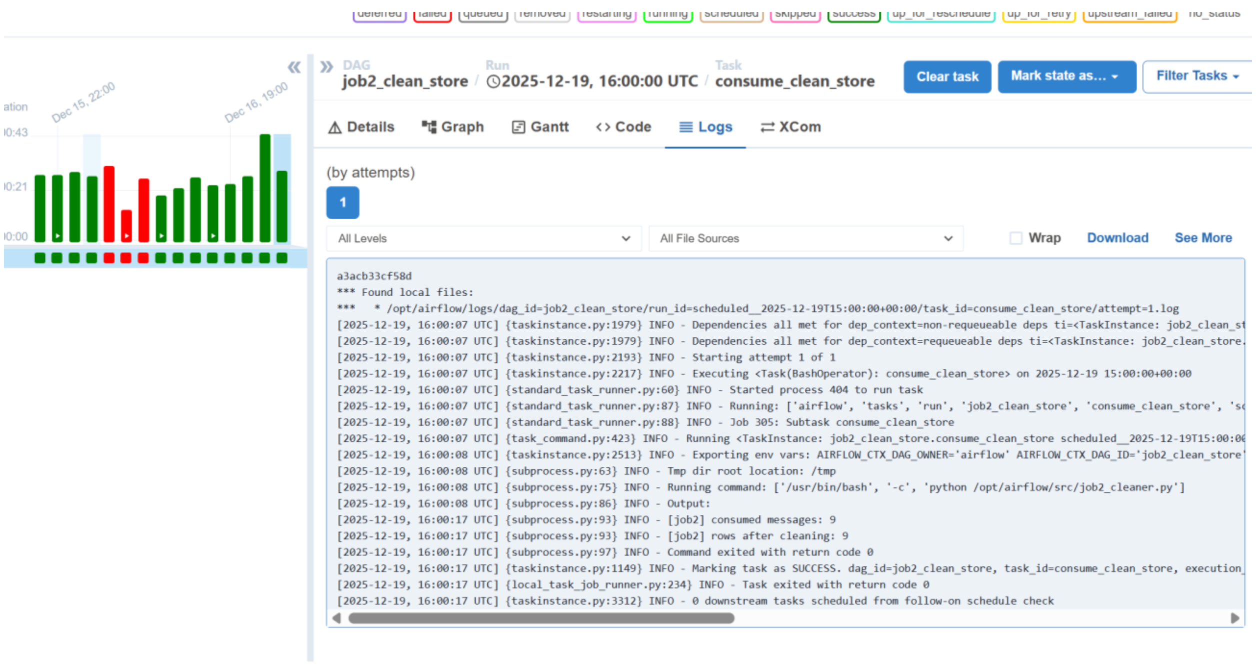


DAG job1_ingestion / Run 2025-12-16, 17:00:00 UTC / Task poll_api_publish_kafka

```
[2025-12-16, 17:00:19 UTC] {subprocess.py:93} INFO - [job1_producer] Loaded state entries: 9
[2025-12-16, 17:00:19 UTC] {subprocess.py:93} INFO - [job1_producer] Publish GB/carbon_intensity: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:19 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=1 offset=23
[2025-12-16, 17:00:19 UTC] {subprocess.py:93} INFO - [job1_producer] Publish GB/total_load: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:19 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=1 offset=24
[2025-12-16, 17:00:20 UTC] {subprocess.py:93} INFO - [job1_producer] Publish GB/price_day_ahead: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:20 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=0 offset=20
[2025-12-16, 17:00:20 UTC] {subprocess.py:93} INFO - [job1_producer] Publish TR/carbon_intensity: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:20 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=0 offset=21
[2025-12-16, 17:00:21 UTC] {subprocess.py:93} INFO - [job1_producer] Publish TR/total_load: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:21 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=0 offset=22
[2025-12-16, 17:00:21 UTC] {subprocess.py:93} INFO - [job1_producer] Publish TR/price_day_ahead: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:21 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=1 offset=25
[2025-12-16, 17:00:21 UTC] {subprocess.py:93} INFO - [job1_producer] Publish FR/carbon_intensity: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:21 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=2 offset=29
[2025-12-16, 17:00:22 UTC] {subprocess.py:93} INFO - [job1_producer] Publish FR/total_load: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:22 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=2 offset=30
[2025-12-16, 17:00:22 UTC] {subprocess.py:93} INFO - [job1_producer] Publish FR/price_day_ahead: datetime=2025-12-16T17:00:00.000Z
[2025-12-16, 17:00:22 UTC] {subprocess.py:93} INFO - [job1_producer] Sent -> topic=raw_events partition=2 offset=31
[2025-12-16, 17:00:22 UTC] {subprocess.py:93} INFO - [job1_producer] Saved state entries: 9
[2025-12-16, 17:00:22 UTC] {subprocess.py:97} INFO - Command exited with return code 0
[2025-12-16, 17:00:22 UTC] {taskinstance.py:1149} INFO - Marking task as SUCCESS. dag_id=job1_ingestion, task_id=poll_api_publish_kafka, execut
[2025-12-16, 17:00:23 UTC] {local_task_job_runner.py:234} INFO - Task exited with return code 0
[2025-12-16, 17:00:23 UTC] {taskinstance.py:3312} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

DAG 2 - Hourly Cleaning + Storage (Kafka -> SQLite events)

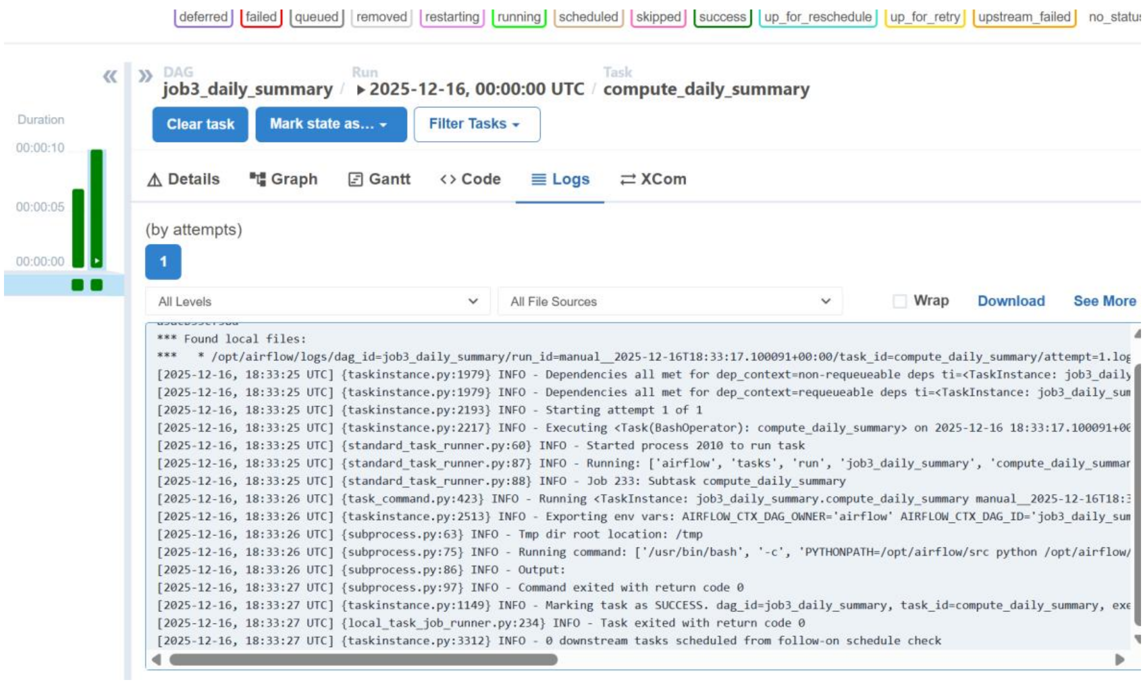
Schedule: @hourly. Consumes new messages with a fixed consumer group and writes cleaned rows to events.



```
a3acb33cf58d
*** Found local files:
***   * /opt/airflow/logs/dag_id=job2_clean_store/run_id=scheduled__2025-12-19T15:00:00+00:00/task_id=consume_clean_store/attempt=1.log
[2025-12-19, 16:00:07 UTC] {taskinstance.py:1979} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: job2_clean_st
[2025-12-19, 16:00:07 UTC] {taskinstance.py:1979} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: job2_clean_store.
[2025-12-19, 16:00:07 UTC] {taskinstance.py:2193} INFO - Starting attempt 1 of 1
[2025-12-19, 16:00:07 UTC] {taskinstance.py:2217} INFO - Executing <Task(BashOperator): consume_clean_store> on 2025-12-19 15:00:00+00:00
[2025-12-19, 16:00:07 UTC] {standard_task_runner.py:60} INFO - Started process 404 to run task
[2025-12-19, 16:00:07 UTC] {standard_task_runner.py:87} INFO - Running: ['airflow', 'tasks', 'run', 'job2_clean_store', 'consume_clean_store', 'sc
[2025-12-19, 16:00:07 UTC] {standard_task_runner.py:88} INFO - Job 305: Subtask consume_clean_store
[2025-12-19, 16:00:07 UTC] {task_command.py:423} INFO - Running <TaskInstance: job2_clean_store.consume_clean_store scheduled__2025-12-19T15:00:00
[2025-12-19, 16:00:08 UTC] {taskinstance.py:2513} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='job2_clean_store'
[2025-12-19, 16:00:08 UTC] {subprocess.py:63} INFO - Tmp dir root location: /tmp
[2025-12-19, 16:00:08 UTC] {subprocess.py:75} INFO - Running command: ['/usr/bin/bash', '-c', 'python /opt/airflow/src/job2_cleaner.py']
[2025-12-19, 16:00:08 UTC] {subprocess.py:86} INFO - Output:
[2025-12-19, 16:00:17 UTC] {subprocess.py:93} INFO - [job2] consumed messages: 9
[2025-12-19, 16:00:17 UTC] {subprocess.py:93} INFO - [job2] rows after cleaning: 9
[2025-12-19, 16:00:17 UTC] {subprocess.py:97} INFO - Command exited with return code 0
[2025-12-19, 16:00:17 UTC] {taskinstance.py:1149} INFO - Marking task as SUCCESS. dag_id=job2_clean_store, task_id=consume_clean_store, execution_
[2025-12-19, 16:00:17 UTC] {local_task_job_runner.py:234} INFO - Task exited with return code 0
[2025-12-19, 16:00:17 UTC] {taskinstance.py:3312} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

DAG 3 - Daily Analytics (SQLite events -> SQLite daily_summary)

Schedule: @daily. Aggregates daily count/min/max/avg per zone and signal into daily_summary.



```
*** Found local files:
***   * /opt/airflow/logs/dag_id=job3_daily_summary/run_id=manual__2025-12-16T18:33:17.100091+00:00/task_id=compute_daily_summary/attempt=1.log
[2025-12-16, 18:33:25 UTC] {taskinstance.py:1979} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: job3_daily
[2025-12-16, 18:33:25 UTC] {taskinstance.py:1979} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: job3_daily_sum
[2025-12-16, 18:33:25 UTC] {taskinstance.py:2193} INFO - Starting attempt 1 of 1
[2025-12-16, 18:33:25 UTC] {taskinstance.py:2217} INFO - Executing <Task(BashOperator): compute_daily_summary> on 2025-12-16 18:33:17.100091+00
[2025-12-16, 18:33:25 UTC] {standard_task_runner.py:60} INFO - Started process 2010 to run task
[2025-12-16, 18:33:25 UTC] {standard_task_runner.py:87} INFO - Running: ['airflow', 'tasks', 'run', 'job3_daily_summary', 'compute_daily_summar
[2025-12-16, 18:33:25 UTC] {standard_task_runner.py:88} INFO - Job 233: Subtask compute_daily_summary
[2025-12-16, 18:33:26 UTC] {task_command.py:423} INFO - Running <TaskInstance: job3_daily_summary.compute_daily_summary manual__2025-12-16T18:3
[2025-12-16, 18:33:26 UTC] {taskinstance.py:2513} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='job3_daily_sum
[2025-12-16, 18:33:26 UTC] {subprocess.py:63} INFO - Tmp dir root location: /tmp
[2025-12-16, 18:33:26 UTC] {subprocess.py:75} INFO - Running command: ['/usr/bin/bash', '-c', 'PYTHONPATH=/opt/airflow/src python /opt/airflow/
[2025-12-16, 18:33:26 UTC] {subprocess.py:86} INFO - Output:
[2025-12-16, 18:33:27 UTC] {subprocess.py:97} INFO - Command exited with return code 0
[2025-12-16, 18:33:27 UTC] {taskinstance.py:1149} INFO - Marking task as SUCCESS. dag_id=job3_daily_summary, task_id=compute_daily_summary, exe
[2025-12-16, 18:33:27 UTC] {local_task_job_runner.py:234} INFO - Task exited with return code 0
[2025-12-16, 18:33:27 UTC] {taskinstance.py:3312} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

Validation Steps

- Kafka: verify raw_events contains JSON messages using a console consumer.
- Job2: run cleaner and confirm events row count increases; duplicates do not insert.
- Job3: run analytics and confirm daily_summary populated with aggregated rows.