

Data Pipeline Report

Team Members:

Serik Bakhramov 20B030585

Sadykova Yasmin 22B22B1540

Birlikzhanova Aruzhan 22B030329

- Data Source (API) Source:

API - <https://api.eia.gov/v2/electricity/rto/fuel-type-data/data/>

Public API with power generation data API returns data in JSON format.

Sample data:

- period — date and hour
- respondent — electricity market operator
- fueltype — fuel type
- value — electricity volume

The API is read-only and requires an API key for access.

- Virtual environment and libraries

A Python virtual environment was created to isolate project dependencies:

```
python -m venv venv
```

```
venv/bin/activate (Windows: venv\Scripts\activate)
```

All required libraries were installed from the requirements.txt file:

```
pip install -r requirements.txt
```

The requirements.txt file contains all dependencies necessary to run

Kafka producers/consumers, data processing with Pandas, and Airflow DAGs.

- Kafka (data streaming) Kafka launch Kafka was installed via Homebrew and launched:
brew services start kafka Creating a Kafka topic kafka-topics --create \ --topic raw_electricity \ --bootstrap-server localhost:9092 \ --partitions 1 \ --replication-factor 1 Kafka Verification
kafka-console-consumer \ --topic raw_electricity \ --from-beginning \ --bootstrap-server localhost:9092

- Job 1 — Ingestion (API → Kafka) File src/job1 [producer.py](#) What does * Makes a request to the API * Receives JSON * Sends each record to Kafka topic raw_electricity Manual verification python src/job1 [producer.py](#) If consumer shows JSON— ingestion works.

- Job 2 — Cleaning & Storage (Kafka → SQLite) Files src/job2 [cleaner.py](#) src/db [utils.py](#) What does * Reads data from Kafka * Clears data: * converts dates * converts value to a number * deletes incorrect values * Creates an SQLite database * Saves data to the events table Manual launch python src/job2 [cleaner.py](#) Checking the database sqlite3 data/app.db
SELECT COUNT(*) FROM events;

- Job 3 — Analytics (SQLite → Aggregation) File src/job3 [analytics.py](#) What does * Reads data from the events table * Groups by: * date * type of fuel * Counts: * average value * maximum * the amount * Saves the result to the daily_summary table Manual launch python src/job3 [analytics.py](#)

- **Airflow (orchestration)** DAG files airflow/dags/job1_ingestion [dag.py](#) airflow/dags/job2_clean_store [dag.py](#) airflow/dags/job3_daily_summary [dag.py](#) DAG 1 — job1_ingestion Purpose: continuous data loading Schedule: None (starts manually) Reason: ingestion is a continuous pipeline (by assignment condition). DAG 2 — job2_clean_store Purpose: cleaning and preservation Schedule: 0 * * * * (every hour) DAG 3 — job3_daily_summary Purpose: Schedule Analytics: 0 0 * * * (once a day)

- **The launch of Airflow** Initialization airflow db init Launch airflow standalone Web UI <http://localhost:8080>

- **Checking in the Airflow UI** Checked in the UI: * DAGs are visible * job2 and job3 are running on schedule * job1 starts manually * the data is successfully traversing the entire pipeline

The screenshot shows the Airflow web interface with the 'app.db' DAG selected. The SQLite database view is open, showing a table with 185 records. The table has columns: id, period, respondent, and fueltype. The first 12 records are visible, showing a sequence of data points with IDs from 2 to 12, periods from 2025-12-17T07:00:00 to 2025-12-17T07:00:00, respondents from AVA to BANC, and fueltypes from OTH to SUN.

id	period	respondent	fueltype
2	2025-12-17T07:00:00...	AVA	OTH
3	2025-12-17T07:00:00...	AVA	SUN
4	2025-12-17T07:00:00...	AVA	WAT
5	2025-12-17T07:00:00...	AVA	WND
6	2025-12-17T07:00:00...	AVRN	NG
7	2025-12-17T07:00:00...	AVRN	SUN
8	2025-12-17T07:00:00...	AVRN	WAT
9	2025-12-17T07:00:00...	AVRN	WND
10	2025-12-17T07:00:00...	BANC	NG
11	2025-12-17T07:00:00...	BANC	OTH
12	2025-12-17T07:00:00...	BANC	SUN

The bottom panel shows the terminal output of the dag-processor, indicating that the data is being processed successfully.

```
dag-processor | [2025-12-19T00:04:53.004+0500] {manager.py:531} INFO - Not time to re
fresh bundle example_dags
```

SQLite database — events table with cleaned data.

data > app.db

Search tables... Reset Filters Records: 24 Search 24 records...

Tables (3)

- events
- sqlite_sequence
- daily_summary

date

fueltype

avg_value

max_value

total_value

	ate	fueltype	avg_value	#	max_value	#	total_value	#
1	025-12-17	BAT		7	14		28	
2	025-12-17	COL	1619.8333333333...		9252		9719	
3	025-12-17	GEO	589.25		1167		2357	
4	025-12-17	NG		2313	9055		32382	
5	025-12-17	NUC	3054.8		11087		15274	
6	025-12-17	OIL	143.6666666666...		770		862	
7	025-12-17	OTH	461.36363636364		1498		5075	
8	025-12-17	PS	0		0		0	
9	025-12-17	SNB	0.4		1		2	
10	025-12-17	SUN	2.8888888888889		26		26	
11	025-12-17	WAT	1243.625		9061		19898	
12	025-12-17	WND	807.36363636364		2222		8881	
13	025-12-18	BAT	0.33333333333333		1		1	
14	025-12-18	COL	1463.1666666666...		8278		8779	
15	025-12-18	GEO	588		1164		2352	
16	025-12-18	NG	1850		7594		25900	
17	025-12-18	NUC	2898.6		11094		14493	
18	025-12-18	...	15.333333333333		46		02	

Page 1 / 1

Try SQLite Viewer in the browser

SQLite database — daily_summary table with aggregated analytics results.

Dags Runs Task Instances

Search Dags Advanced Search

All Failed Running Success Enabled Filter by tag Reset 1 filter

3 Days Sort by Latest Run Start Date...

job1_ingestion

Schedule Latest Run Next Run

2025-12-18, 23:43:44

job3_daily_summary

Schedule Latest Run Next Run

2025-12-18, 23:43:47 2025-12-19, 05:00:00

job2_clean_store

Schedule Latest Run Next Run

2025-12-18, 23:43:49 2025-12-19, 00:00:00

Launchpad

Dags Runs Task Instances

Search Dags Advanced Search

All Failed Running Success All Filter by tag Reset 1 filter

3 Days Sort by Latest Run Start Date...

job1_ingestion

Schedule Latest Run Next Run

2025-12-18, 18:01:27

job3_daily_summary

Schedule Latest Run Next Run

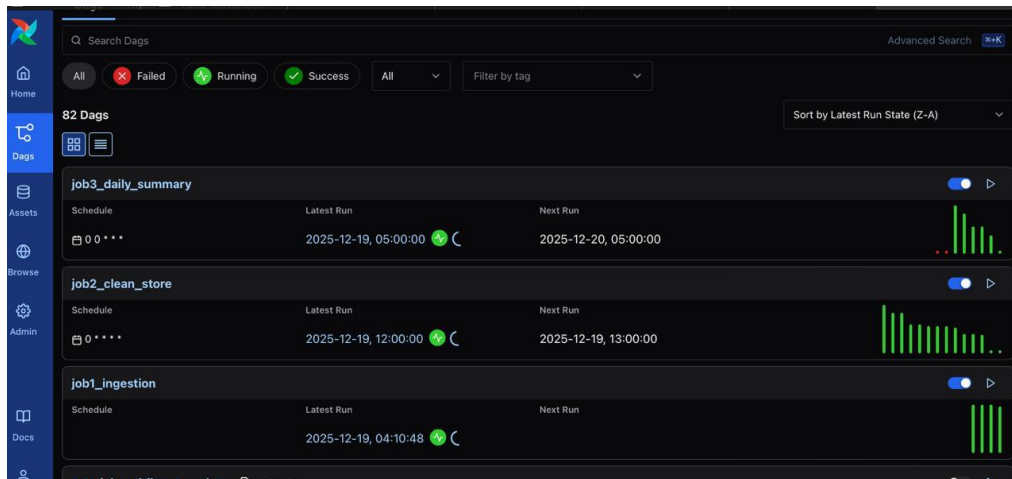
2025-12-18, 18:01:32 2025-12-19, 05:00:00

job2_clean_store

Schedule Latest Run Next Run

2025-12-18, 23:00:00 2025-12-19, 00:00:00

User



Airflow UI showing all DAGs and their successful runs
Successful DAG execution in Airflow
Successful task log — data cleaning and storage
Successful task log — daily analytics aggregation