Project Report: Real-Time Air Quality Monitoring Pipeline

Team Members: Begali Dinmukhammed, Nazarov Dastan, Batyrbek Raiymbek

Date: December 19, 2025

1. API Justification

For this project, we selected the **Air Quality Index (AQI) API** (https://aqicn.org/).

- **Relevance:** Air quality is a critical environmental factor. Real-time monitoring allows for health alerts and urban planning.
- **Data Richness:** The API provides not only the overall AQI but also granular data like station coordinates, measurement time, and individual pollutant levels (though we focused on AQI for consistency).
- **Reliability:** It aggregates data from over 30,000 stations globally, providing a robust stream for our Kafka pipeline.

2. Kafka Topic Schema

The system uses a single Kafka topic named weather_raw_data. Data is produced in **JSON** format.

Message structure:

JSON
```
{
 "city_raw": "String (e.g., 'Almaty, Kazakhstan')",
 "aqi": "Integer (0-500+)",
 "lat": "Float (Latitude)",
 "lon": "Float (Longitude)",
 "utime": "Integer (Unix Timestamp)",
 "timestamp": "String (ISO Format)"
}
```

- **Serialization:** UTF-8 encoded strings.
- **Partitions:** 1 (Default for development).

3. Data Cleaning Rules

Before storage, data undergoes a cleaning process in job2_cleaner.py:

1. **Normalization:** The city_raw field is split by commas to extract the primary city name. Leading/trailing spaces are removed, and the string is converted to **Title Case** (e.g., "ALMATY" -> "Almaty").
2. **Type Conversion:** AQI is explicitly cast to Integer. Coordinates are cast to Float.
3. **Handling Missing Values:** If AQI is missing, the record is discarded (filtering). If coordinates are missing, they are stored as NULL.
4. **Invalid Record Filtering:** Records with an AQI outside the logical range (0-999) are filtered out to remove sensor errors or "outlier" noise.

4. SQLite Storage Schema

The storage layer consists of two tables in app.db.

Table: events (Cleaned Data)

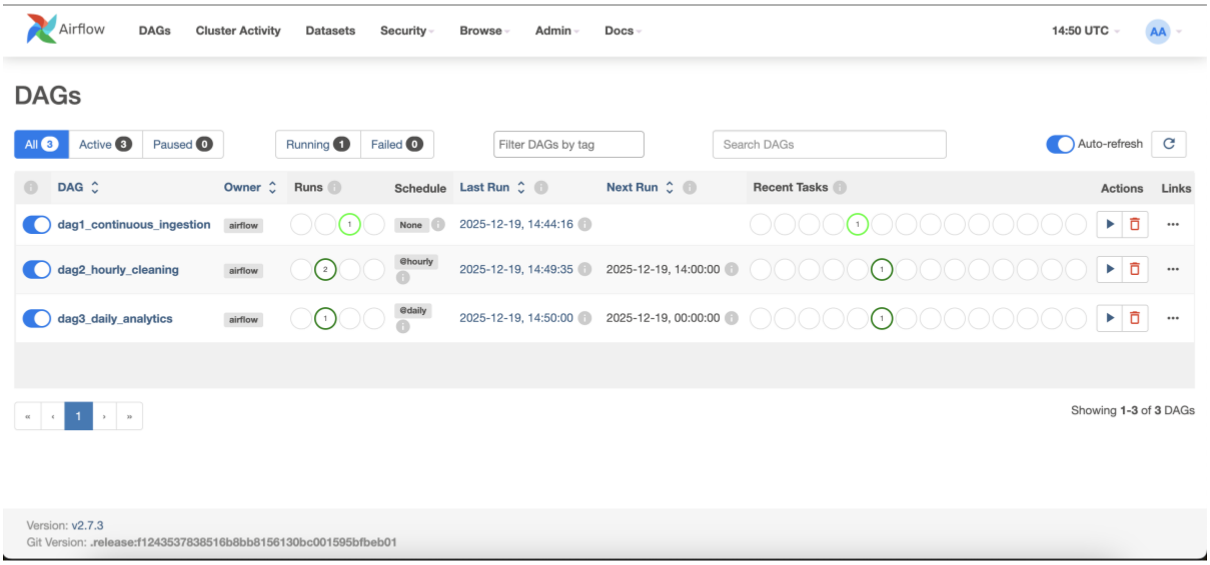
Column	Type	Description
id	INTEGER	Primary Key (Autoincrement)
city	TEXT	Cleaned city name
aqi	INTEGER	Air Quality Index
lat	REAL	Latitude
lon	REAL	Longitude
utime	INTEGER	Unix Time
timestamp	DATETIME	Formatted date/time

Table: daily_summary (Aggregated Analytics)

Column	Type	Description
city	TEXT	Primary Key (City name)
min_aqi	INTEGER	Lowest AQI recorded
max_aqi	INTEGER	Highest AQI recorded
avg_aqi	REAL	Average AQI
total_measurements	INTEGER	Count of observations
air_quality_category	TEXT	'Good', 'Moderate', 'Unhealthy', etc.
last_sync	DATETIME	Time of last calculation

5. Implementation Evidence (Screenshots)

5.1 Airflow DAGs



DAGs

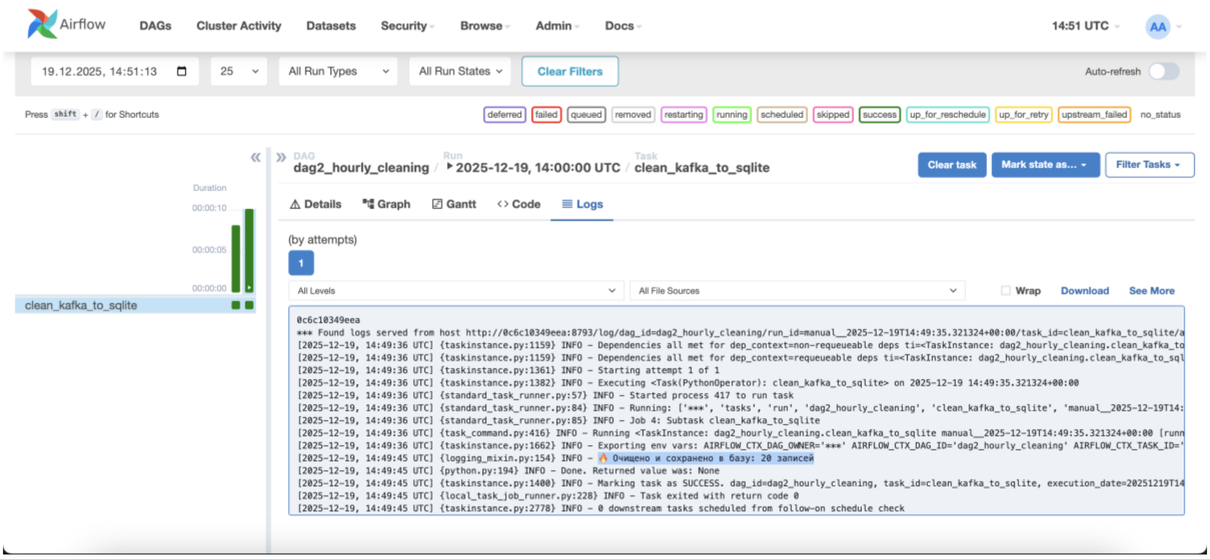
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
dag1_continuous_ingestion	airflow	1	None	2025-12-19, 14:44:16		1	▶ 🗑	···
dag2_hourly_cleaning	airflow	2	@hourly	2025-12-19, 14:49:35	2025-12-19, 14:00:00	1	▶ 🗑	···
dag3_daily_analytics	airflow	1	@daily	2025-12-19, 14:50:00	2025-12-19, 00:00:00	1	▶ 🗑	···

« ‹ 1 › »

Showing **1-3** of **3** DAGs

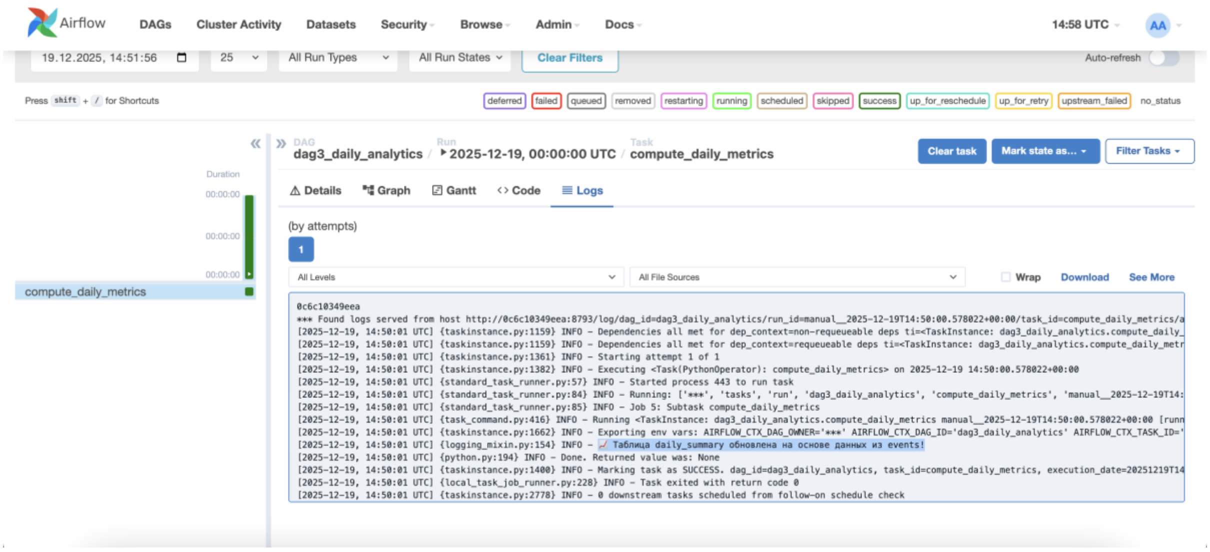
Version: v2.7.3
Git Version: .release:f1243537838516b8bb8156130bc001595bfbeb01

5.2 Successful Task Logs (Job 2 Cleaner)



```
0c6c10349eea
*** Found logs served from host http://0c6c10349eea:8793/log/dag_id=dag2_hourly_cleaning/run_id=manual__2025-12-19T14:49:35.321324+00:00/task_id=clean_kafka_to_sqlite/a
[2025-12-19, 14:49:36 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: dag2_hourly_cleaning.clean_kafka_to
[2025-12-19, 14:49:36 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: dag2_hourly_cleaning.clean_kafka_to_sql
[2025-12-19, 14:49:36 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1
[2025-12-19, 14:49:36 UTC] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): clean_kafka_to_sqlite> on 2025-12-19 14:49:35.321324+00:00
[2025-12-19, 14:49:36 UTC] {standard_task_runner.py:57} INFO - Started process 417 to run task
[2025-12-19, 14:49:36 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'dag2_hourly_cleaning', 'clean_kafka_to_sqlite', 'manual__2025-12-19T14:
[2025-12-19, 14:49:36 UTC] {standard_task_runner.py:85} INFO - Job 4: Subtask clean_kafka_to_sqlite
[2025-12-19, 14:49:36 UTC] {task_command.py:416} INFO - Running <TaskInstance: dag2_hourly_cleaning.clean_kafka_to_sqlite manual__2025-12-19T14:49:35.321324+00:00 [runn
[2025-12-19, 14:49:36 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='***' AIRFLOW_CTX_DAG_ID='dag2_hourly_cleaning' AIRFLOW_CTX_TASK_ID='
[2025-12-19, 14:49:45 UTC] {logging_mixin.py:154} INFO - 🔥 Очищено и сохранено в базу: 20 записей
[2025-12-19, 14:49:45 UTC] {python.py:194} INFO - Done. Returned value was: None
[2025-12-19, 14:49:45 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=dag2_hourly_cleaning, task_id=clean_kafka_to_sqlite, execution_date=20251219T14
[2025-12-19, 14:49:45 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2025-12-19, 14:49:45 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

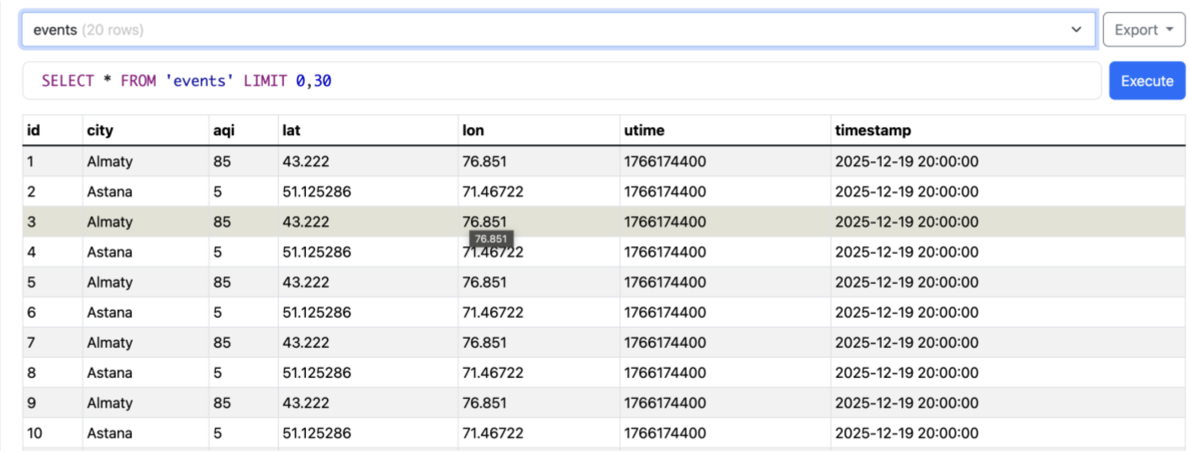
5.3 Successful Task Logs (Job 3 Analytics)



```
0c6c10349eea
*** Found logs served from host http://0c6c10349eea:8793/log/dag_id=dag3_daily_analytics/run_id=manual__2025-12-19T14:50:00.578022+00:00/task_id=compute_daily_metrics/a
[2025-12-19, 14:50:01 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: dag3_daily_analytics.compute_daily_
[2025-12-19, 14:50:01 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: dag3_daily_analytics.compute_daily_metr
[2025-12-19, 14:50:01 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1
[2025-12-19, 14:50:01 UTC] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): compute_daily_metrics> on 2025-12-19 14:50:00.578022+00:00
[2025-12-19, 14:50:01 UTC] {standard_task_runner.py:57} INFO - Started process 443 to run task
[2025-12-19, 14:50:01 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'dag3_daily_analytics', 'compute_daily_metrics', 'manual__2025-12-19T14:
[2025-12-19, 14:50:01 UTC] {standard_task_runner.py:85} INFO - Job 5: Subtask compute_daily_metrics
[2025-12-19, 14:50:01 UTC] {task_command.py:416} INFO - Running <TaskInstance: dag3_daily_analytics.compute_daily_metrics manual__2025-12-19T14:50:00.578022+00:00 [runn
[2025-12-19, 14:50:01 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='***' AIRFLOW_CTX_DAG_ID='dag3_daily_analytics' AIRFLOW_CTX_TASK_ID='
[2025-12-19, 14:50:01 UTC] {logging_mixin.py:154} INFO - 📈 Таблица daily_summary обновлена на основе данных из events!
[2025-12-19, 14:50:01 UTC] {python.py:194} INFO - Done. Returned value was: None
[2025-12-19, 14:50:01 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=dag3_daily_analytics, task_id=compute_daily_metrics, execution_date=20251219T14
[2025-12-19, 14:50:01 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2025-12-19, 14:50:01 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

5.4 Final Database Output

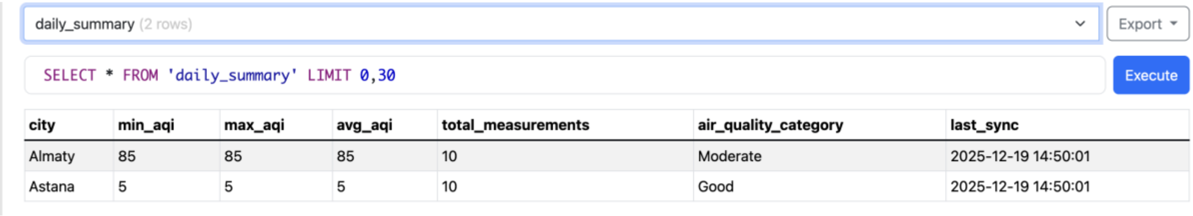
Table events



```
SELECT * FROM 'events' LIMIT 0,30
```

id	city	aqi	lat	lon	utime	timestamp
1	Almaty	85	43.222	76.851	1766174400	2025-12-19 20:00:00
2	Astana	5	51.125286	71.46722	1766174400	2025-12-19 20:00:00
3	Almaty	85	43.222	76.851	1766174400	2025-12-19 20:00:00
4	Astana	5	51.125286	71.46722	1766174400	2025-12-19 20:00:00
5	Almaty	85	43.222	76.851	1766174400	2025-12-19 20:00:00
6	Astana	5	51.125286	71.46722	1766174400	2025-12-19 20:00:00
7	Almaty	85	43.222	76.851	1766174400	2025-12-19 20:00:00
8	Astana	5	51.125286	71.46722	1766174400	2025-12-19 20:00:00
9	Almaty	85	43.222	76.851	1766174400	2025-12-19 20:00:00
10	Astana	5	51.125286	71.46722	1766174400	2025-12-19 20:00:00

Table daily_summary



```
SELECT * FROM 'daily_summary' LIMIT 0,30
```

city	min_aqi	max_aqi	avg_aqi	total_measurements	air_quality_category	last_sync
Almaty	85	85	85	10	Moderate	2025-12-19 14:50:01
Astana	5	5	5	10	Good	2025-12-19 14:50:01

6. Configuration & Extensibility

The pipeline is designed for high flexibility through two root-level configuration files:

- **.env (Security):** Stores the AQI_TOKEN privately. This ensures that sensitive API keys are never hard-coded, following security best practices and allowing for easy key rotation.
- **cities.txt (Scalability):** Contains a list of target cities (one per line). The system dynamically reads this file at runtime; to monitor new locations, a user simply updates this list without needing to modify the Python scripts.

7. Conclusion

The pipeline successfully demonstrates a full ETL/ELT flow:

1. **Ingestion:** Real-time data collection via API into Kafka.
2. **Processing:** Stream-like batch cleaning and normalization.
3. **Storage:** Reliable storage in a structured SQLite database.
4. **Analytics:** Automated metric calculation for end-user insights.