Data Orchestration with Apache Airflow

Orkestrasi Data Warehouse dan ETL Pipeline

Tujuan

Mengubah implementasi data warehouse DuckDB dan pipeline ETL menjadi alur kerja yang siap produksi dan ter-orkestrasi menggunakan Apache Airflow.

Bagian 1: Menyiapkan Lingkungan Airflow

Implementasi code ada pada etl_pipeline_dag.py di repository.

Langkah yang dilakukan:

- Setting environment lingkungan Airflow pada local machine menggunakan Docker
- Konfigurasikan koneksi yang diperlukan untuk sumber data dan DuckDB Anda
- Jalankan Airflow untuk melihat apakah DAG sudah berhasil di load
- Testing DAG menggunakan tool standar Airflow (trigger running)
- Monitoring dan improving untuk mencapai DAG yang sesuai kebutuhan proses ETL

Bagian 2: Desain dan Implementasi DAG

2.1 **DAG Pipeline ETL**

Pipeline ETL diubah menjadi DAG file untuk Airflow menggunakan operator “>>”. Detail implementasi dapat dilihat pada repositori berikut repository.

2.2 **Fitur Airflow Lanjutan**

Beberapa fitur lanjut dari Airflow telah digunakan pada DAG, seperti:

- Implementasi sudah menggunakan **Branching logic based on conditions** untuk handle kasus bercabang,
- Implementasi sudah menggunakan **dynamic task generation**, pada bagian eksekusi fungsi ETL dimension,
- Implementasi sudah menggunakan **Error handling and retry mechanisms** pada saat extract data,
- Implementasi sudah menggunakan email notifications ketika failure.

2.3 **Strategi Penjadwalan dan Partisi**

Pada DAG etl_pipeline_dag.py, kami menetapkan strategi penjadwalan sebagai berikut:

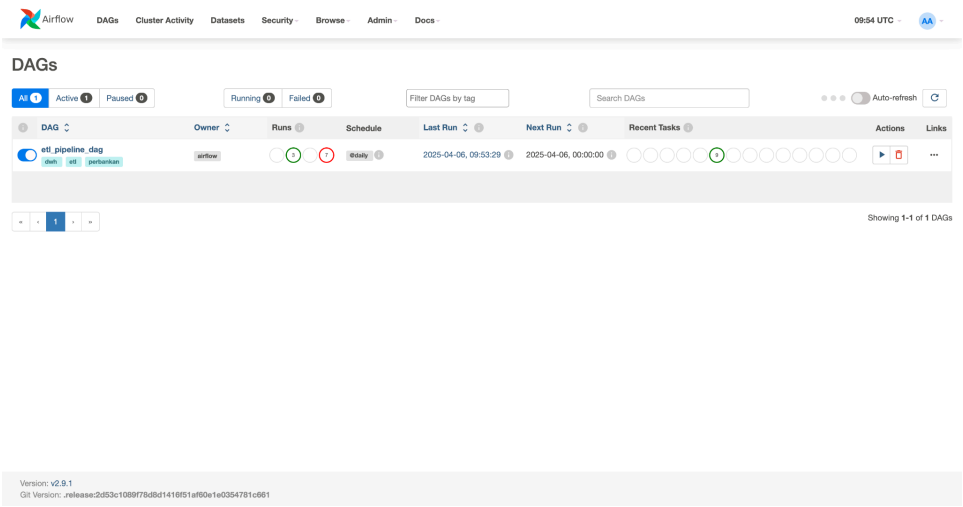
- Schedule Interval: **daily**
  DAG dijalankan setiap hari pada pukul 00.00 setiap hari. Jadwal ini dipilih karena proses ETL dirancang untuk load data transaksi harian dari sistem operasional yang sudah selesai diproses seharian dan agar tidak mengganggu proses operasional di jam jam sibuk selama sehari (menghindari jam sibuk operasional)
- Start Date: **2025-01-01**
  Tanggal awal penjadwalan ditentukan agar mencerminkan proses data sejak awal tahun, dan memungkinkan proses backfilling jika dibutuhkan.
- Catchup: **False**
  Kami menonaktifkan catchup agar Airflow hanya mengeksekusi DAG sesuai dengan jadwal saat ini, bukan mengulang ke belakang, karena data hanya tersedia real-time dan tidak perlu historical run.

Bagian 3: Pengujian dan Dokumentasi

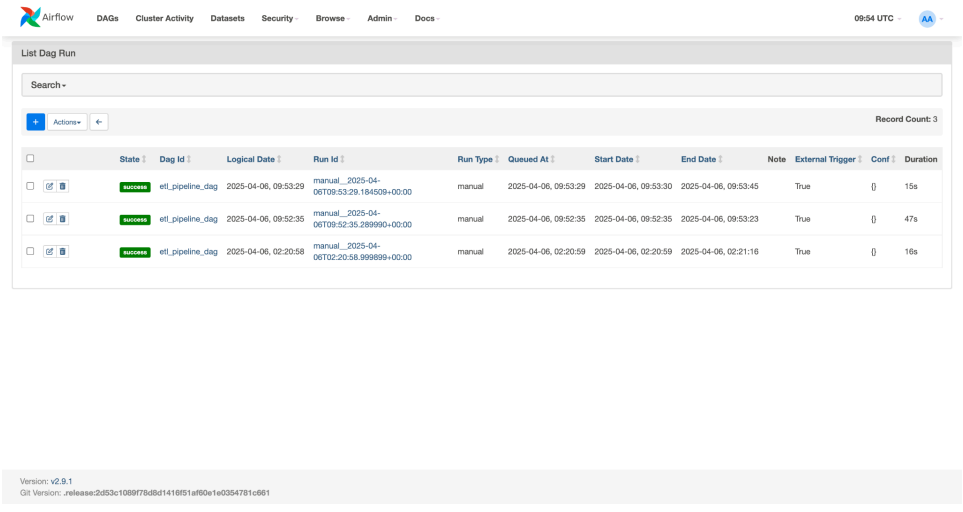
3.1 **Pengujian DAG**

Hasil pengujian DAG dan proses eksekusi pada Airflow yang sudah dibuat:

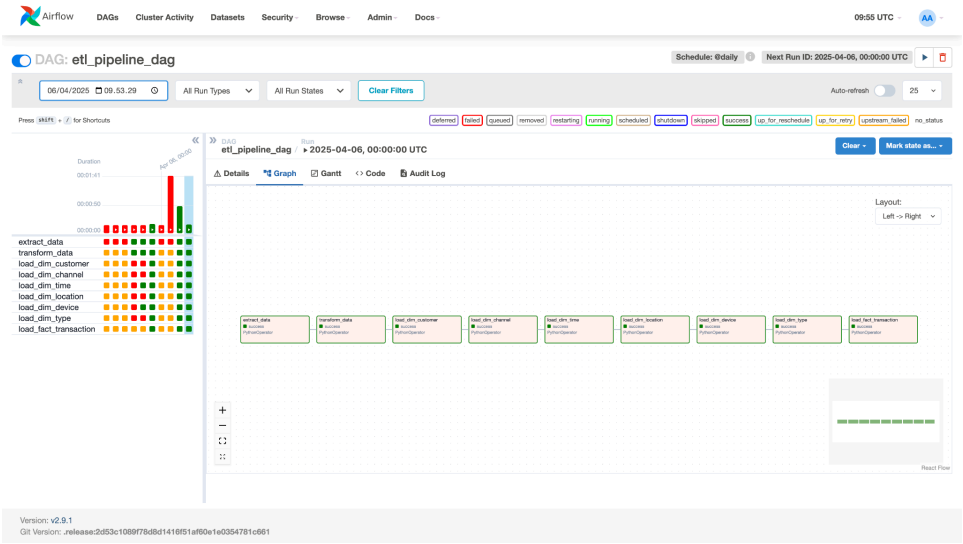
1. Trigger DAG manually



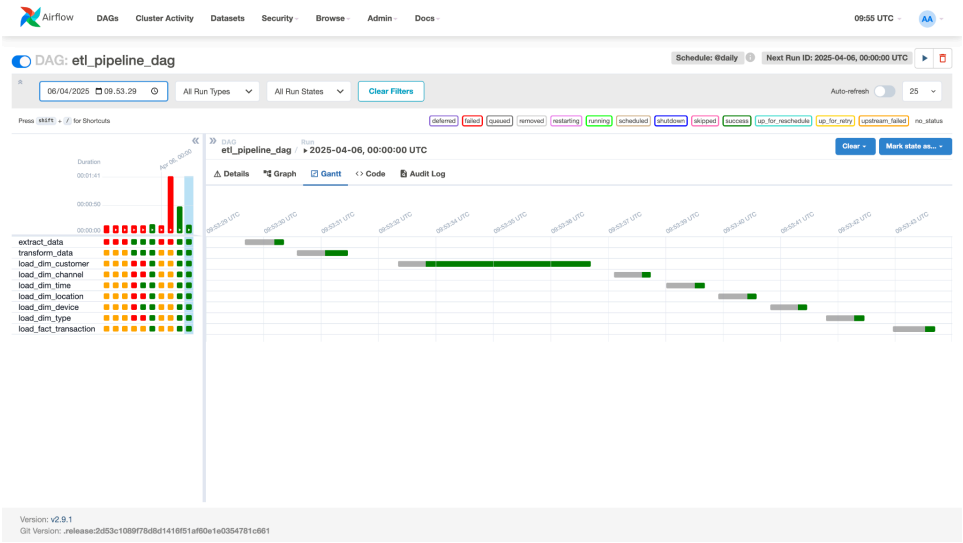
2. DAGs run successfully



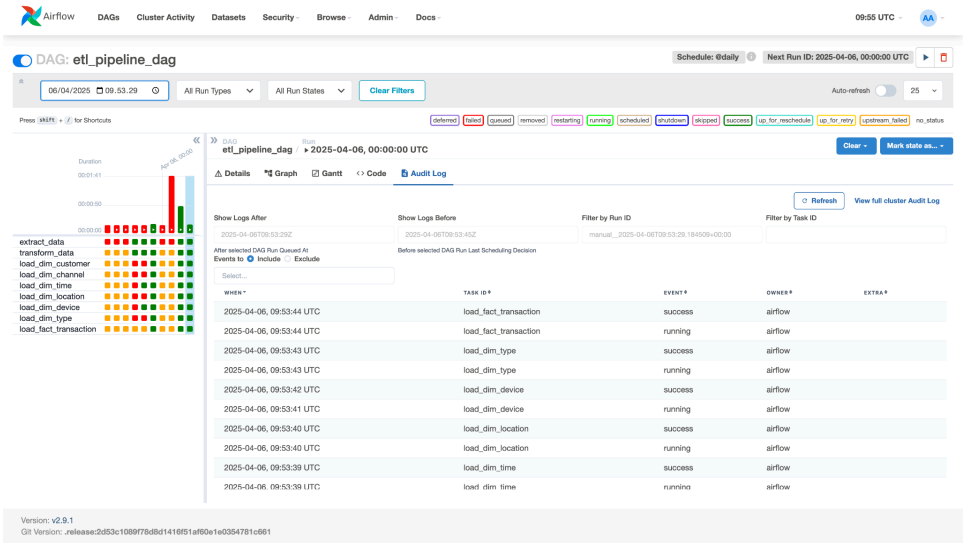
3. Graph View



4. Grant View



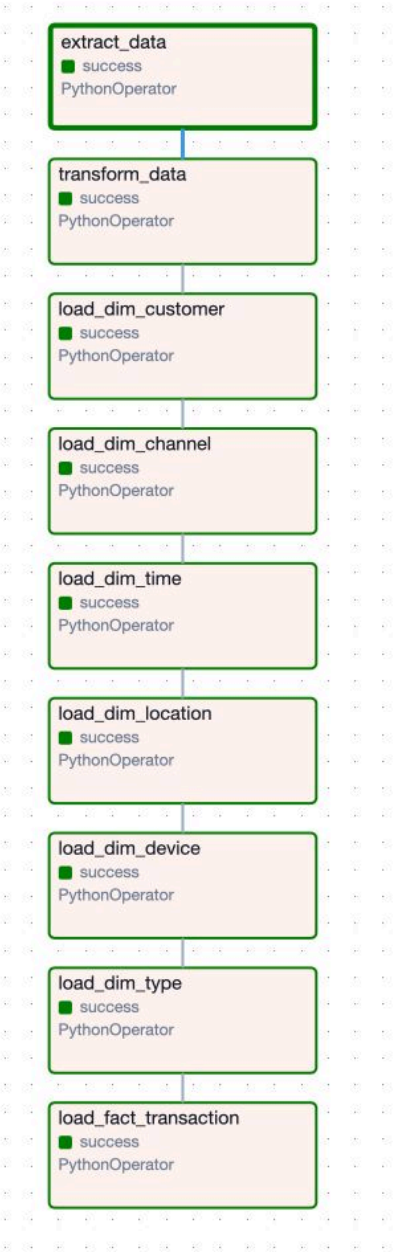
5. Log View


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
09:55 UTC
AA
DAG: etl_pipeline_dag
Schedule: @daily
Next Run ID: 2025-04-06, 00:00:00 UTC
06/04/2025 09.53.29
All Run Types
All Run States
Clear Filters
Auto-refresh
25
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
00:01:41
00:00:50
00:00:00
extract_data
transform_data
load_dim_customer
load_dim_channel
load_dim_time
load_dim_location
load_dim_device
load_dim_type
load_fact_transaction
DAG
etl_pipeline_dag
Run
2025-04-06, 00:00:00 UTC
Clear
Mark state as...
Details
Graph
Gantt
Code
Audit Log
Refresh
View full cluster Audit Log
Show Logs After
2025-04-06T09:53:29Z
Show Logs Before
2025-04-06T09:53:45Z
Filter by Run ID
manual__2025-04-06T09:53:29.184509+00:00
Filter by Task ID
After selected DAG Run Queued At
Events to Include Exclude
Before selected DAG Run Last Scheduling Decision
Select...
WHEN
TASK ID
EVENT
OWNER
EXTRA
2025-04-06, 09:53:44 UTC load_fact_transaction success airflow
2025-04-06, 09:53:44 UTC load_fact_transaction running airflow
2025-04-06, 09:53:43 UTC load_dim_type success airflow
2025-04-06, 09:53:43 UTC load_dim_type running airflow
2025-04-06, 09:53:42 UTC load_dim_device success airflow
2025-04-06, 09:53:41 UTC load_dim_device running airflow
2025-04-06, 09:53:40 UTC load_dim_location success airflow
2025-04-06, 09:53:40 UTC load_dim_location running airflow
2025-04-06, 09:53:39 UTC load_dim_time success airflow
2025-04-06, 09:53:39 UTC load_dim_time running airflow
Version: v2.9.1
Git Version: .release:2d53c1089f78d8d1416f51af60e1e0354781c661

3.2 **Dokumentasi**

Diagram DAGs

Diagram DAG berisi alur proses ETL yang terdiri dari:



Dalam tampilan di atas menunjukkan alur eksekusi DAG (Directed Acyclic Graph) pada Apache Airflow yang terdiri dari rangkaian tasks ETL (Extract, Transform, Load), dimulai dari:

a. *extract_data* – mengekstrak data mentah yang berasal dari file .csv

b. *transform_data* – mentransformasi data dan menghapus data duplikat
c. Kemudian dilanjutkan dengan memuat data ke berbagai tabel dimensi seperti: *load_dim_customer, load_dim_channel, load_dim_time, load_dim_location, load_dim_device, load_dim_type*.
d. Diakhiri dengan *load_fact_transaction* untuk memuat data ke tabel fakta.

Setiap task menggunakan PythonOperator dan seluruh proses telah berhasil dijalankan (ditandai dengan status success berwarna hijau).

Dependencies

Proses DAGs dilakukan secara serial

- Extract Data
- Transform Data yang mempunyai dependency pada proses extract data
- Load data ke table dimensi, proses load data dilakukan secara serial, jika dilakukan paralel terjadi isu koneksi duckdb, yang tidak bisa membuat koneksi secara bersamaan untuk proses load setiap tabel dimensi
- Load data fact, yang mempunyai dependensi ke semua tabel dimensi

Monitoring

- Penggunaan fitur notifikasi melalui email notification jika gagal
- Penggunaan status proses pada DAGs airflow web UI

Pemulihan kegagalan

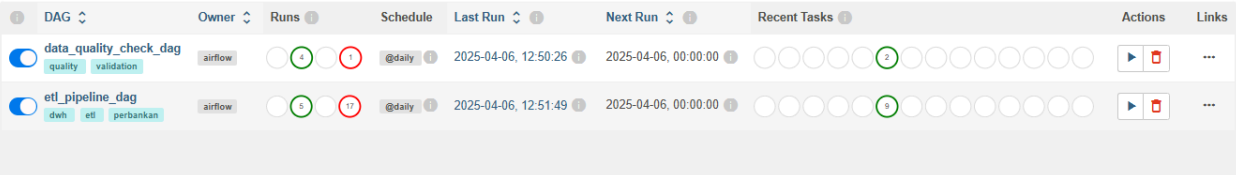
- Retry otomatis menggunakan retry mechanism
- Pemantauan Airflow web UI dashboard, untuk melihat DAG yang gagal
- Melihat log DAG yang gagal
- Pencarian root cause dan trigger manual untuk retry

Bagian 4: Kualitas Data

Mengimplementasikan pemeriksaan kualitas data dengan menggunakan airflow dimana yang menjadi fokus utama adalah

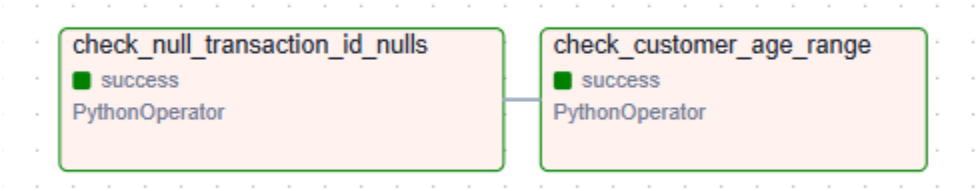
1. Tidak boleh ada nilai null dalam tabel fact_transaction
2. Batas Usia customer adalah dari 17 tahun hingga 80 tahun

Membuat satu DAGs baru yang berisikan Data Quality Check



Dari proses quality data check ditemukan bahwa:

1. Tidak ada nilai null di dalam tabel fact_transaction
2. Semua customer memiliki tentang usia antara 0 sampai 80 tahun



Setiap task menggunakan PythonOperator dan seluruh proses telah berhasil dijalankan (ditandai dengan status success berwarna hijau)

Bagian 5: Rencana Persiapan Visualisasi

Seluruh data telah berhasil diproses melalui pipeline ETL dan dimuat ke dalam data warehouse berbasis DuckDB. Data disusun ke dalam tabel dimensi seperti *dim_customer*, *dim_channel*, *dim_time*, dan *dim_device*, serta satu tabel fakta *fact_transaction*, yang merepresentasikan peristiwa transaksi.

Struktur data telah dirancang dengan star schema untuk mengoptimalkan performa query analitik. Setiap tabel memiliki primary key dan foreign key yang memastikan integritas relasional antar tabel, serta telah melewati proses validasi dengan DAG data quality monitoring untuk menjamin kelayakan data.

Data ini kemudian siap digunakan oleh tools visualisasi seperti Tableau atau Looker Studio, yang mampu menangani berbagai jenis visualisasi interaktif dan mendalam. Tools ini dipilih karena fleksibilitas dan kemampuannya mendukung eksplorasi data serta pembuatan dashboard dinamis yang sangat bermanfaat bagi kebutuhan analisis dan pengambilan keputusan level manajerial.

Contoh kueri yang akan berguna untuk visualisasi

1. Tren jumlah transaksi per bulan

```
SELECT
    dt.Year,
    dt.Month,
    COUNT(ft.TransactionID) AS total_transactions,
    SUM(ft.TransactionAmount) AS total_amount
FROM Fact_Transaction ft
JOIN Dim_Time dt ON ft.TimeID = dt.TimeID
GROUP BY dt.Year, dt.Month
ORDER BY dt.Year, dt.Month;
```

2. Total Tansaksi Per-channel

```
SELECT
    dc.Channel AS channel_name,
    COUNT(ft.TransactionID) AS total_transactions,
    SUM(ft.TransactionAmount) AS total_amount
FROM Fact_Transaction ft
JOIN Dim_Channel dc ON ft.ChannelID = dc.Channel_ID
GROUP BY dc.Channel
ORDER BY total_amount DESC;
```