Coding Challenge

Apache Airflow: Features, Installation, and Pipeline Building Guide

Key Features of Apache Airflow

1. **Dynamic Pipeline Generation**

   Workflows are written as Python code, allowing for dynamic DAG (Directed Acyclic Graph) creation.

2. **Scalability**

   Supports scaling horizontally across multiple workers.

3. **Extensibility**

   Provides a rich set of operators, hooks, sensors, and allows custom plugin development.

4. **Monitoring and UI**

   Web-based UI to visualize, monitor, and manage workflows with detailed logs.

5. **Task Dependency Management**

   Explicitly defines dependencies between tasks for better control.

6. **Integration**

   Integrates with a wide range of systems (AWS, GCP, Azure, Hadoop, Databases, etc.).

7. **Resiliency**

   Automatic retrying of failed tasks, SLA monitoring, and alerting.

8. **Versioning and Modularity**

   DAGs are modular and version-controlled like standard code.

Installing Apache Airflow with Docker (Using YAML)

1. Pre-requisites

Install **Docker** and **Docker Compose** on your system.
Verify installation:

```
docker –version
docker compose version
```

2. Create a yaml File

Download the YAML file and store it in the folder(materials).

3. Access Airflow UI

Start all services:

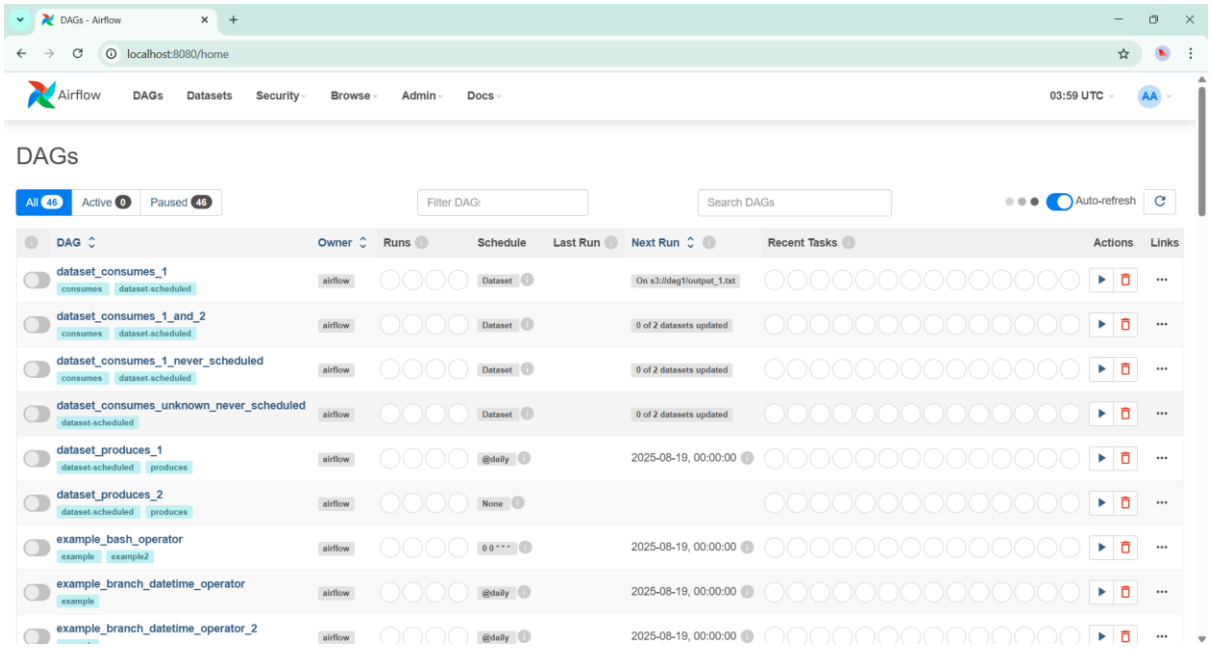
docker compose up -d

Open browser: http://localhost:8080

Steps to Build a Pipeline in Airflow

Step 1: Open Airflow Web UI

- Navigate to http://localhost:8080
- The Airflow home page will open.

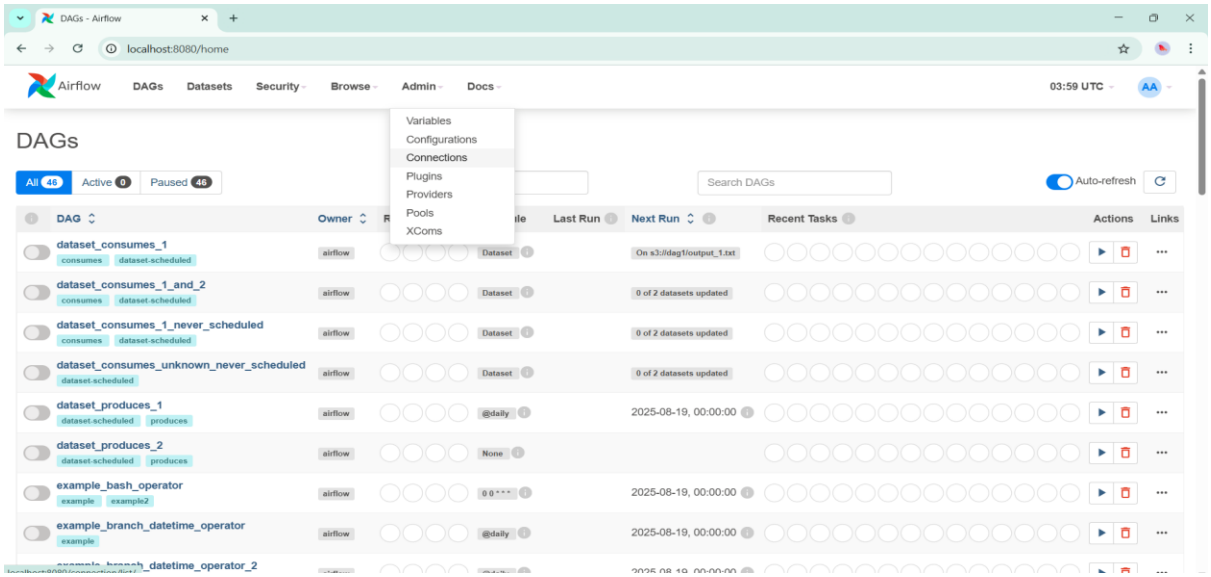


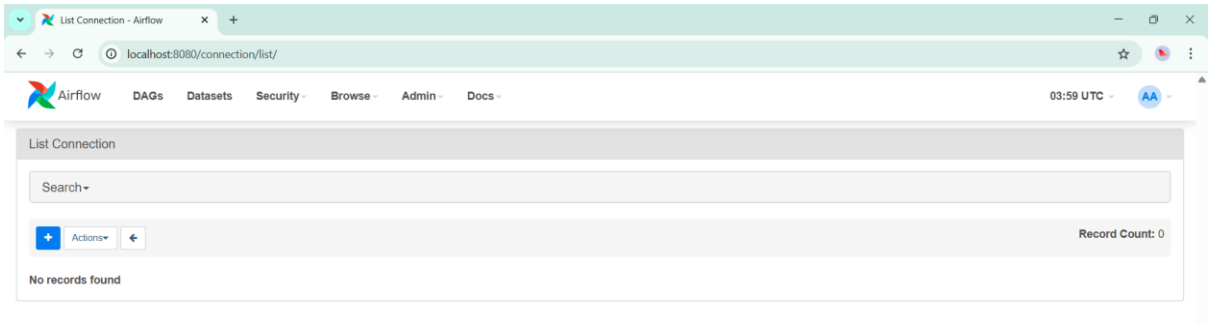
Step 2: Create Connection

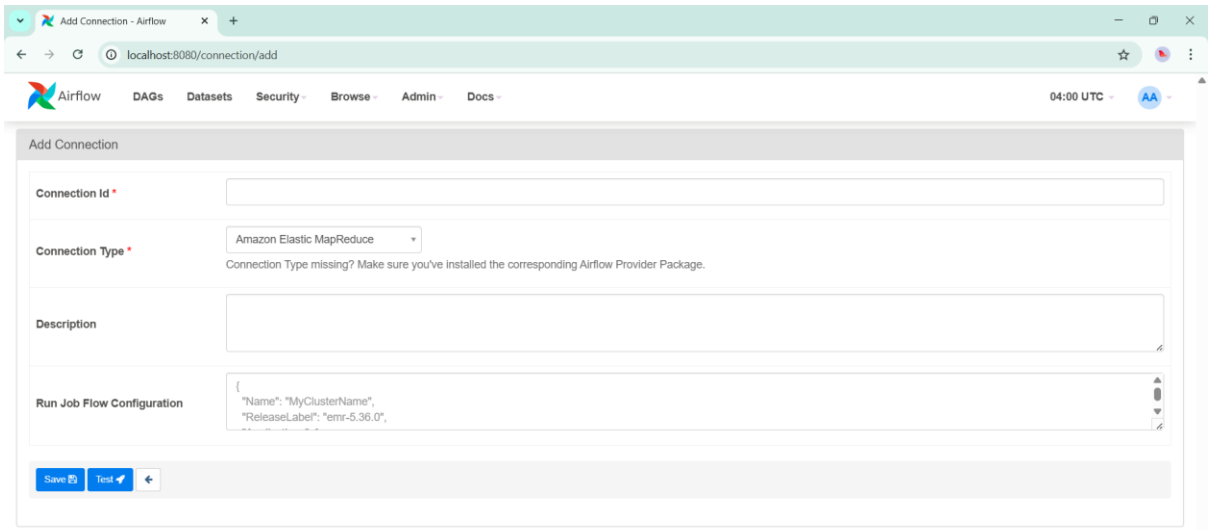
- Go to **Admin → Connections**.
- Add a new Postgres connection:
  - **Connection ID**: tutorial_pg_conn
  - **Connection Type**: Postgres
  - **Host**: postgres
  - **Database**: airflow

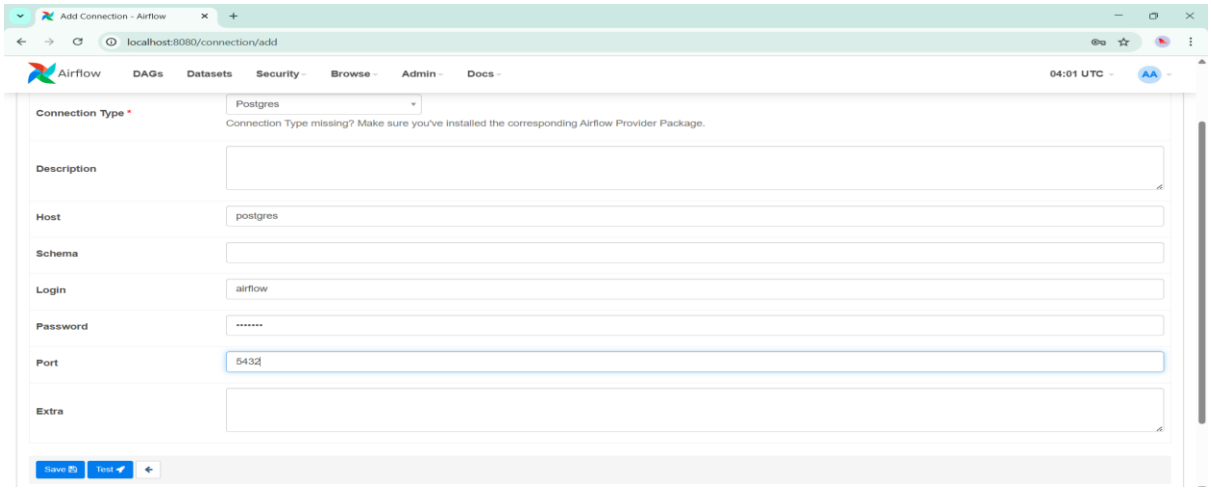
- **Login**: airflow
- **Password**: airflow
- **Port**: 5432

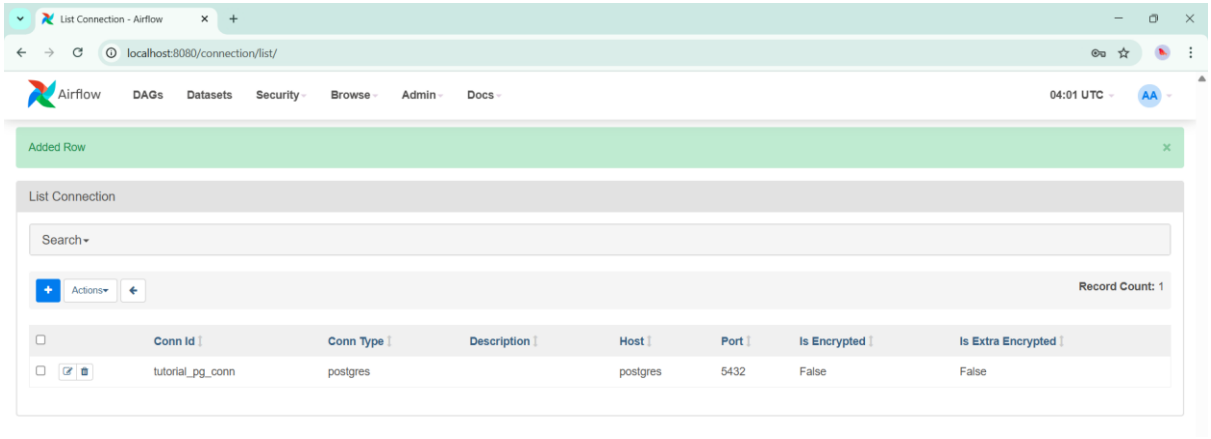
- Save the connection.









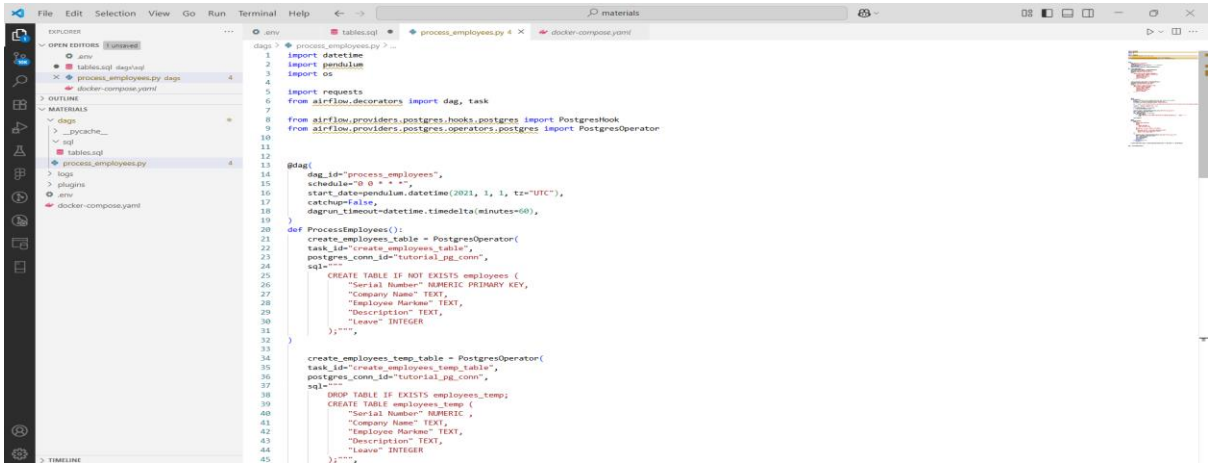


Step 3: Create SQL File

- Go to dags/sql/
- Create tables.sql with SQL queries for creating required tables.

Step 4: Create Python DAG File

- Go to dags/
- Create process_employee.py:

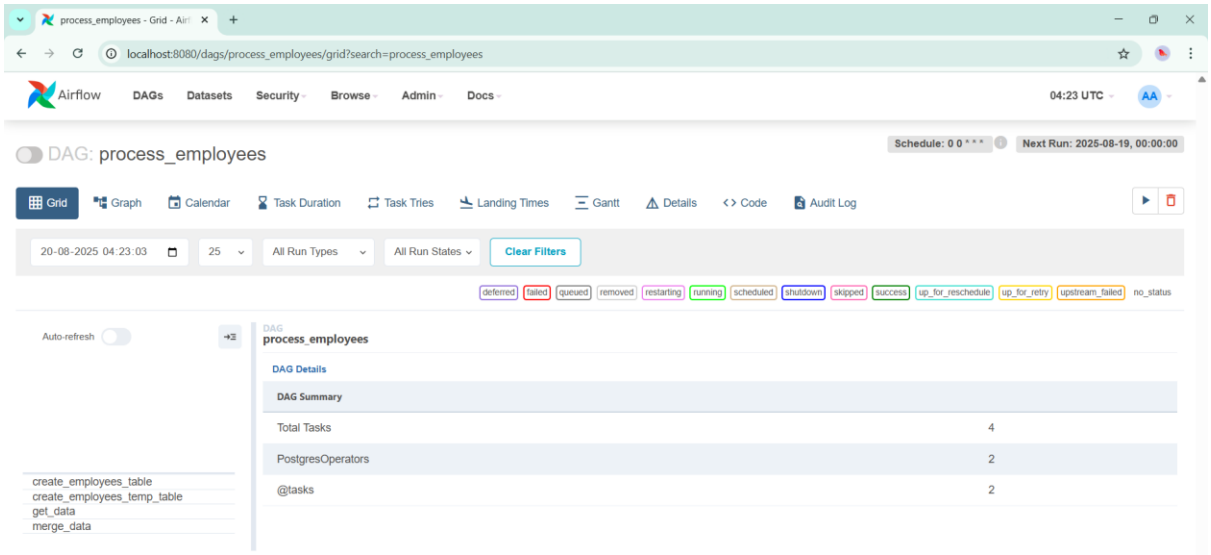


Step 5: Deploy the DAG

- Place process_employee.py in the dags/ directory.
- Airflow will detect it.

Step 6: Run the DAG

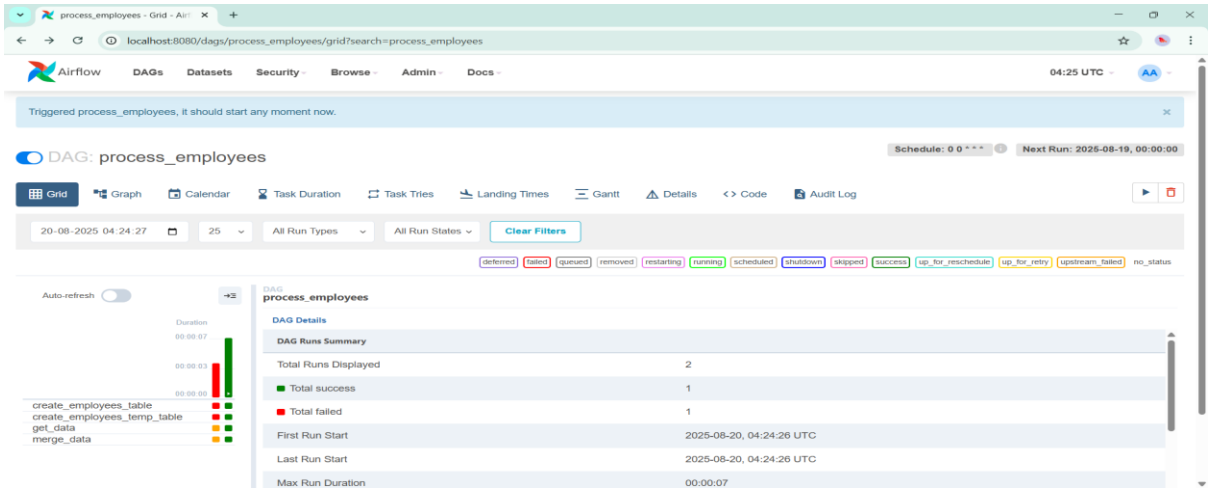
- Trigger the DAG from the web UI.
- Monitor task execution in **Graph View** or **Tree View**.



Viewing Pipelines in Airflow

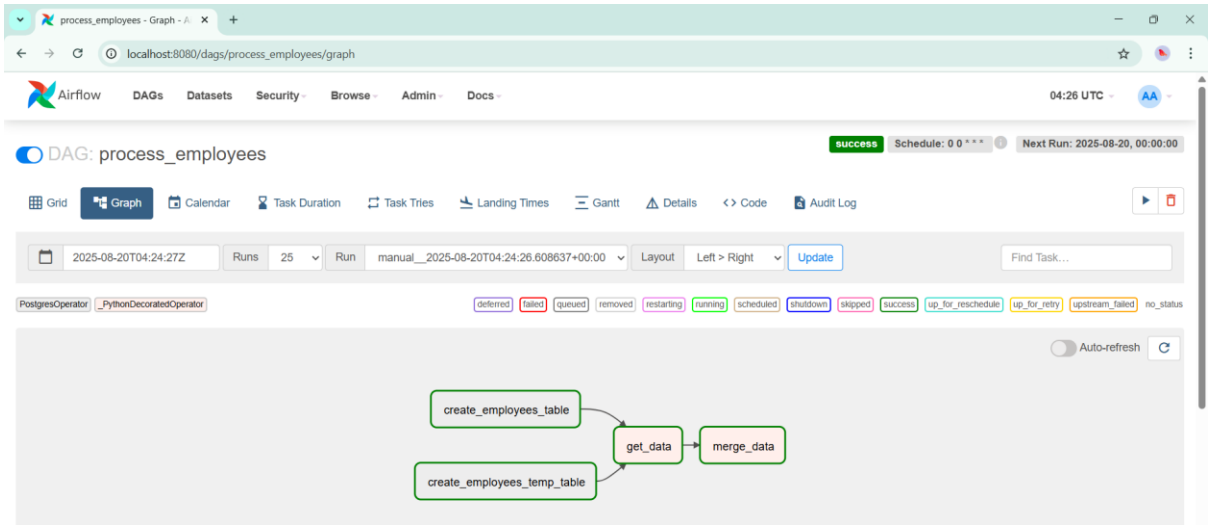
Grid View

- Displays DAG runs and tasks in a timeline-like grid.
- Each column represents a DAG run, and each row represents a task.
- Provides quick insights into task status across multiple runs.



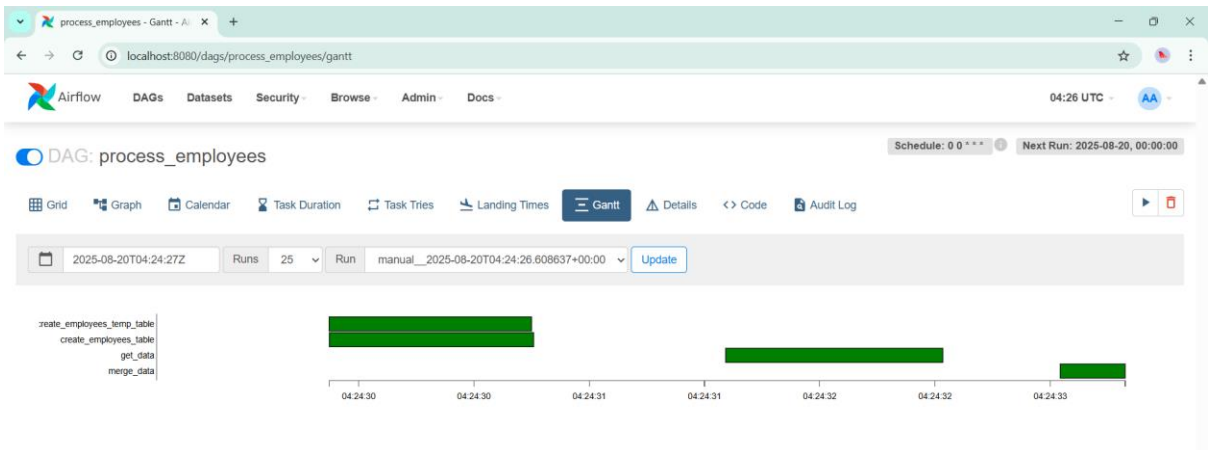
Graph View

- Shows DAG structure as a directed acyclic graph.
- Visualizes task dependencies and execution flow.
- Helpful for understanding pipeline logic.



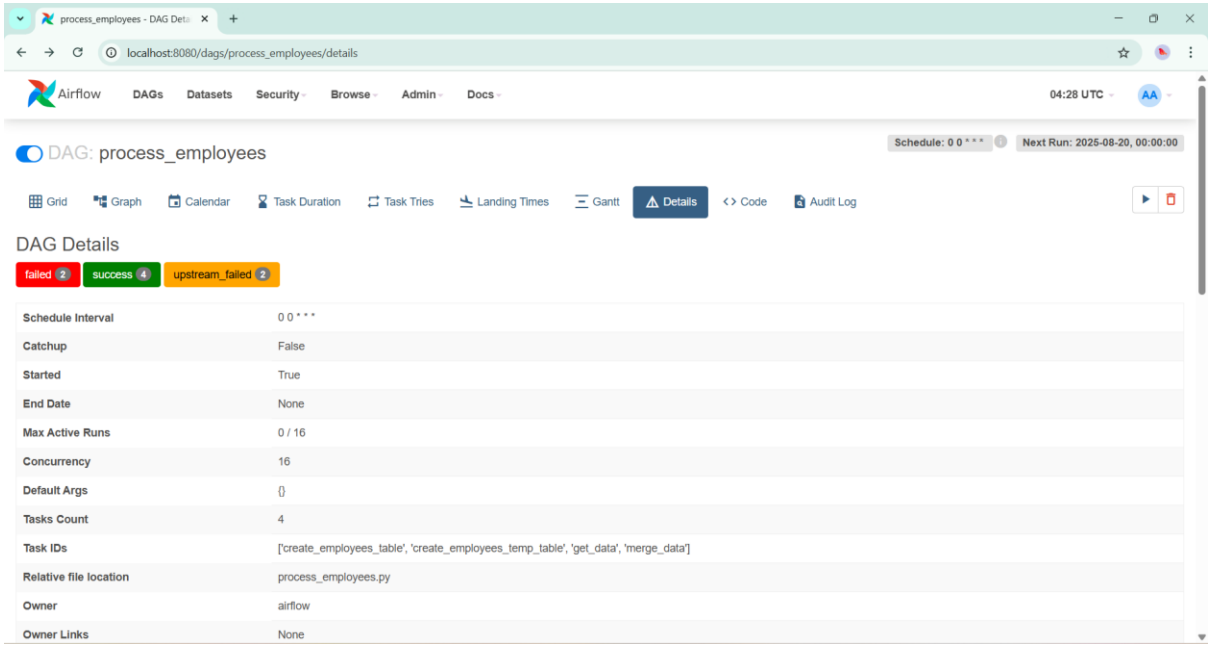
Gantt View

- Visualizes task execution duration.
- Helps identify bottlenecks and overlapping tasks.
- Useful for performance monitoring.



DAG Details

- Shows overall information about a DAG.
- Includes schedule, description, start date, owners, and configuration.
- Links to logs, task instances, and code view.



Summary

Using **Docker Compose with YAML**, Apache Airflow can be set up quickly. A sample **Employee Processing Pipeline** demonstrates how to configure connections, create SQL scripts, define DAGs, and execute them. The Airflow UI provides powerful tools like **Grid View, Graph View, Gantt View, and DAG Details** for comprehensive monitoring and debugging.