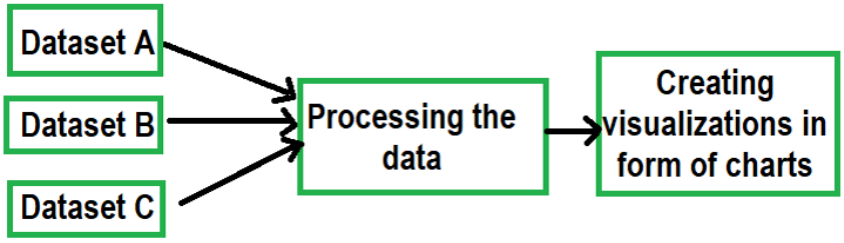
Apache Airflow

What is Apache Airflow?

- Open-source tool to author, schedule, and monitor workflows.
- Created by Airbnb (2015), now under Apache Software Foundation.
- Helps manage complex, batch-oriented data pipelines.
- Pipelines are written in Python → flexible & dynamic.
- Workflows can be visualized (dependencies, progress, logs, failures, retries).

Working of Airflow and DAG

- Workflow = process to achieve a goal (e.g., load data → process → visualize).
- DAG (Directed Acyclic Graph) = representation of workflow.
    - Directed → order of execution matters.
    - Acyclic → no loops/cycles.
- Tasks in DAG can run parallel if independent.



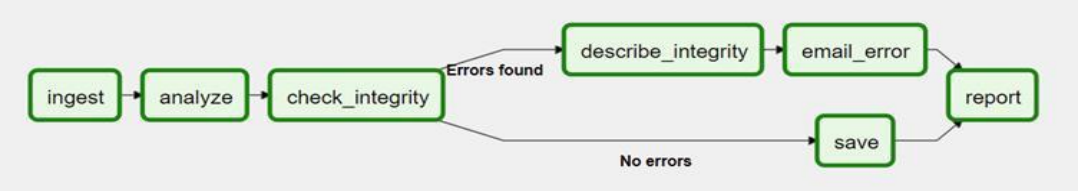
Core Components of Airflow

1. Dynamic → pipelines generated dynamically using Python.
2. Extensible → custom operators/hooks can be created.
3. Elegant → workflows are clean, readable.
4. Scalable → modular, uses message queues for communication.

Benefits of Apache Airflow

- Large community support.
- Extensible to any environment.
- Workflows are code-based → version control (Git).
- Rich scheduling and flexible execution semantics.
- Requires minimal Python knowledge.

- Free & open-source.
- Integrates with cloud platforms (AWS, GCP, Azure).
- Graphical UI to monitor workflows.

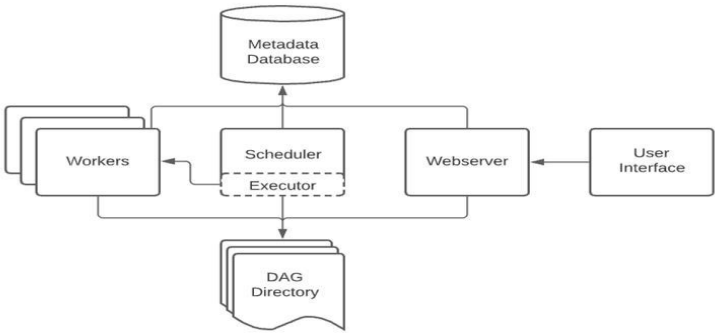


Airflow Use Cases

- ETL pipelines (data extraction, transformation, loading).
- Machine learning workflows (model training, retraining).
- Automated reports generation.
- Backups, alerts, DevOps tasks.

Airflow Architecture

- DAGs → define workflows.
- Scheduler → triggers tasks at right time.
- Executor → sends tasks to workers.
- Workers → run tasks.
- Metadata DB → stores task states.
- Web Server (UI) → monitoring and debugging.
- DAG folder → stores DAG Python files.



Control Flow

- DAGs can run multiple times & in parallel.
- Dependencies between tasks defined using >> and <<.
- Trigger rules can override default success-only execution.

- Branching → choose different execution paths.
- TaskGroups / SubDAGs → organize complex workflows.

Install Apache Airflow on Windows with Docker

1. Install the Prerequisites

Docker Desktop

1. Download Docker Desktop for Windows → Docker Download
2. Install it and enable WSL 2 Backend (during installation it will ask you to install Linux kernel).
3. After installation, restart your PC.
4. Verify installation:
5. docker --version
6. docker-compose --version

Visual Studio Code

1. Download Visual Studio Code (VS Code) → VS Code Download
2. Install it with recommended extensions:
    - Docker Extension (helps manage containers inside VS Code).
    - Python Extension (if you plan to edit DAGs).

2. Download Airflow Docker Setup

- Clone/download Airflow's official docker-compose example:
- curl -LfO 'https://airflow.apache.org/docs/apache-airflow/2.4.2/docker-compose.yaml'
- Or you can manually create a docker-compose.yaml file in your project folder.

3. Create a .env File

Inside the same folder as docker-compose.yaml, create a .env file in VS Code:

```
AIRFLOW_IMAGE_NAME=apache/airflow:2.4.2
AIRFLOW_UID=50000
```

Explanation:

- AIRFLOW_IMAGE_NAME → Airflow image version.

- AIRFLOW_UID → ensures correct file permissions between Docker and Windows.

4. Start Airflow Services

Run this command in the same folder as docker-compose.yaml:

```
docker-compose up -d
```

This will spin up services like:

- airflow-webserver
- airflow-scheduler
- airflow-worker
- postgres (for metadata DB)
- redis (for Celery queue)

Check if containers are running:

```
docker ps
```

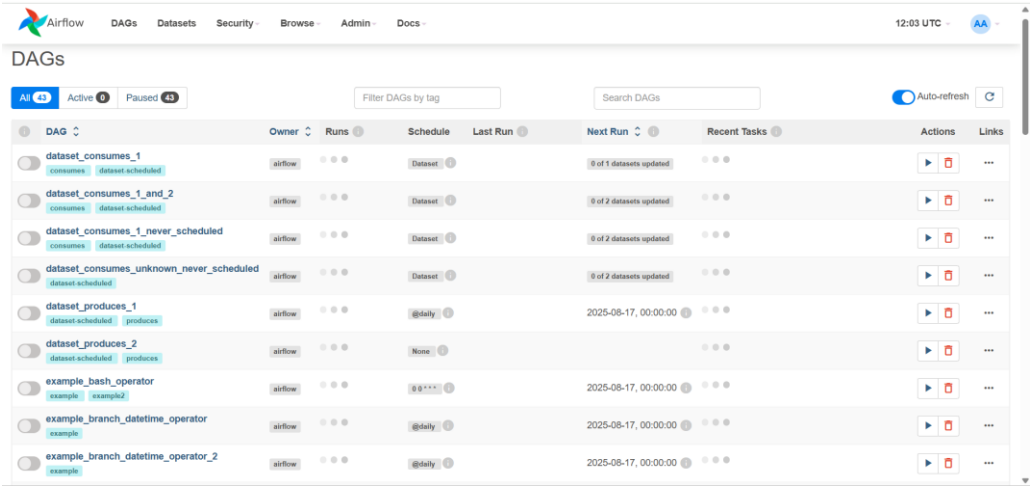
5. Create Airflow Admin User

Run this command to create the Admin login:

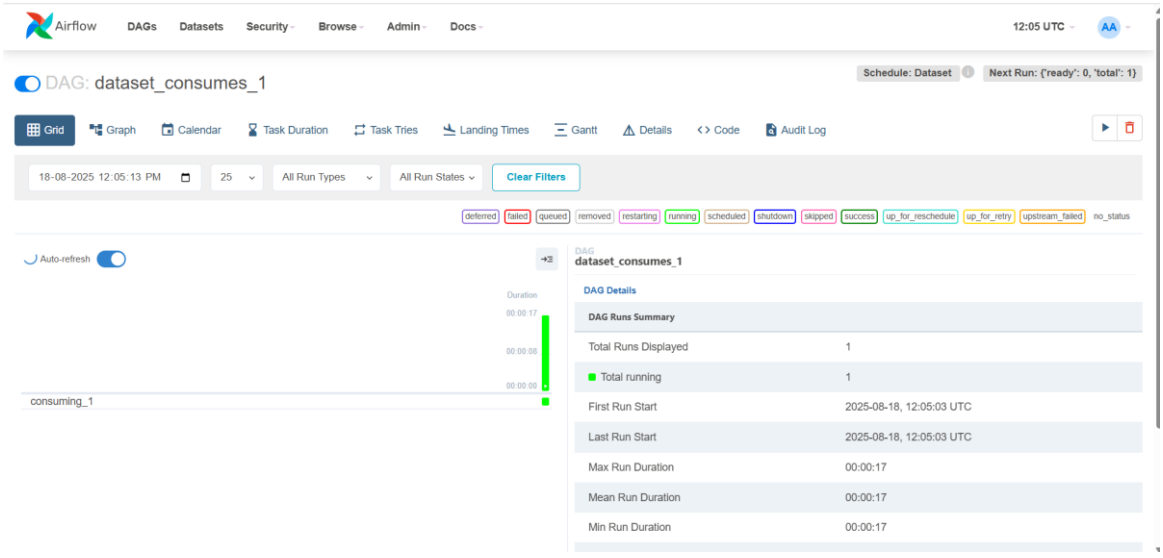
```
docker-compose run airflow-worker airflow users create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin
```

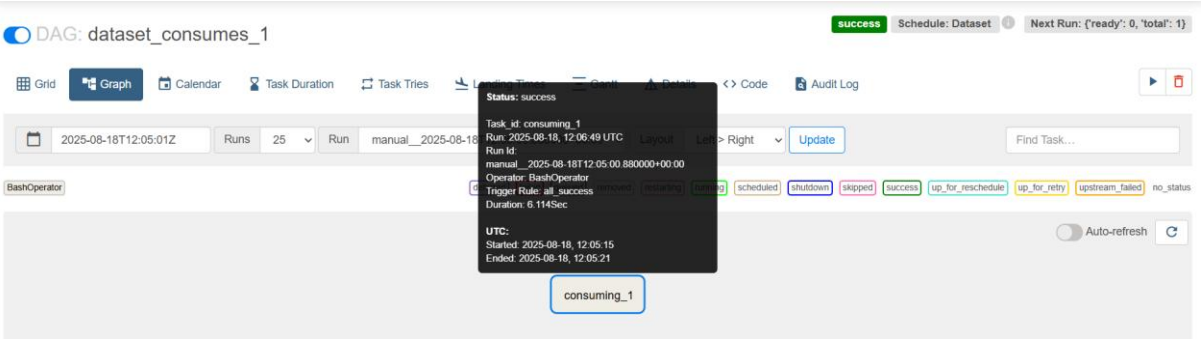
6. Access Airflow UI

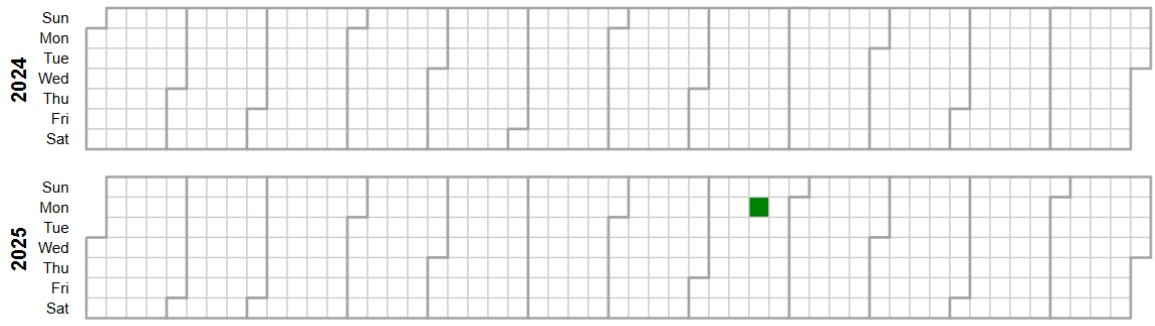
1. Open browser → http://localhost:8080
2. Login with:
   - Username: admin
   - Password: admin

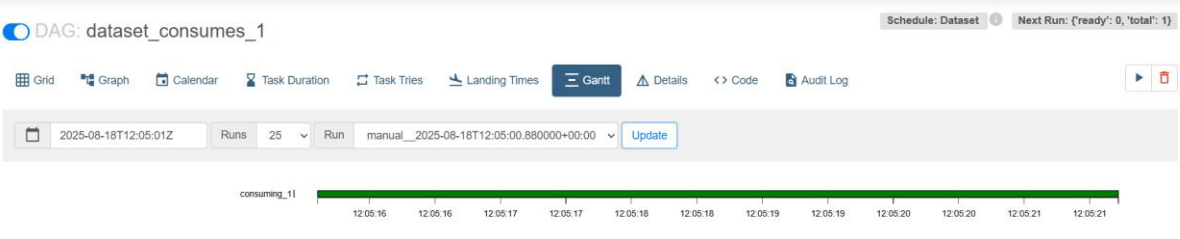


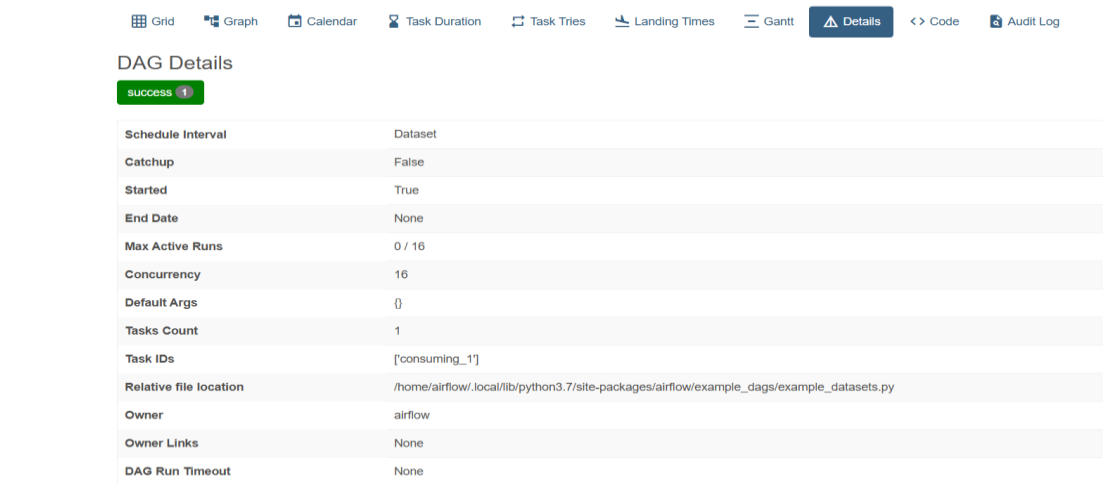
Performing some DAG operations:



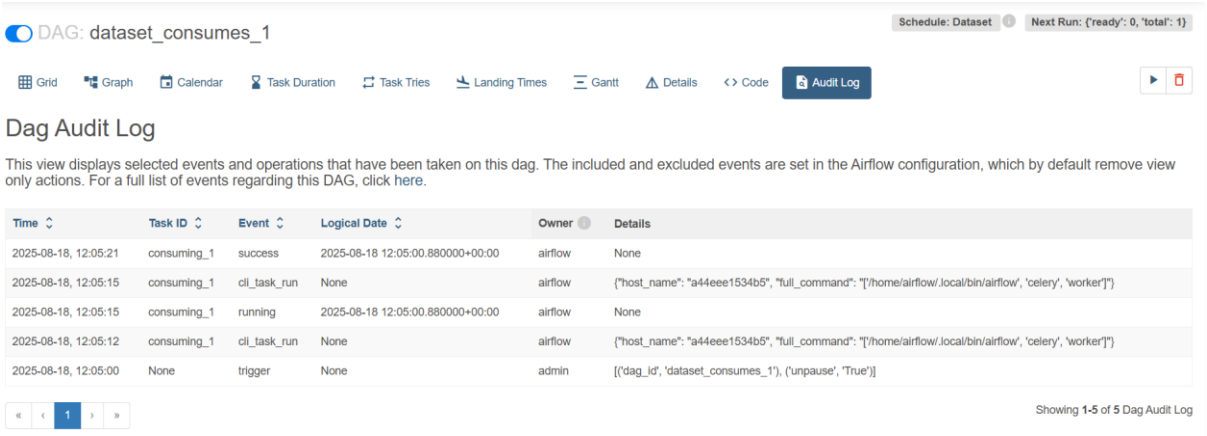








Schedule Interval	Dataset
Catchup	False
Started	True
End Date	None
Max Active Runs	0 / 16
Concurrency	16
Default Args	{}
Tasks Count	1
Task IDs	['consuming_1']
Relative file location	/home/airflow/.local/lib/python3.7/site-packages/airflow/example_dags/example_datasets.py
Owner	airflow
Owner Links	None
DAG Run Timeout	None



Dag Audit Log

This view displays selected events and operations that have been taken on this dag. The included and excluded events are set in the Airflow configuration, which by default remove view only actions. For a full list of events regarding this DAG, click here.

Time	Task ID	Event	Logical Date	Owner	Details
2025-08-18, 12:05:21	consuming_1	success	2025-08-18 12:05:00.880000+00:00	airflow	None
2025-08-18, 12:05:15	consuming_1	cli_task_run	None	airflow	{"host_name": "a44eee1534b5", "full_command": "['/home/airflow/.local/bin/airflow', 'celery', 'worker']"}
2025-08-18, 12:05:15	consuming_1	running	2025-08-18 12:05:00.880000+00:00	airflow	None
2025-08-18, 12:05:12	consuming_1	cli_task_run	None	airflow	{"host_name": "a44eee1534b5", "full_command": "['/home/airflow/.local/bin/airflow', 'celery', 'worker']"}
2025-08-18, 12:05:00	None	trigger	None	admin	[('dag_id', 'dataset_consumes_1'), ('unpause', 'True')]

« ‹ 1 › »

Showing **1-5** of **5** Dag Audit Log