Airflow setup and building pipeline

What is Apache Airflow?

Apache Airflow is an **open-source workflow orchestration tool** created by Airbnb (2014) and later donated to Apache. It helps **author, schedule, and monitor workflows** (pipelines).

Why use Airflow?

- Before Airflow, people wrote **cron jobs** or custom scripts → messy, hard to maintain.
- Airflow solves this by providing:
    - **DAGs (Directed Acyclic Graphs):** You define your workflow as a graph of tasks with dependencies.
    - **Python-based definition:** Pipelines are written in **Python code**, not some DSL → easier for developers.
    - **UI Monitoring:** A **web interface** to see running, success, failed tasks with logs.
    - **Scalable execution:** Runs on a single machine or scales across clusters.

Key Features of Apache Airflow

1. **DAGs (Directed Acyclic Graphs)**
    - Workflows are DAGs → no cycles (no infinite loops).
    - Tasks (nodes) → actual units of work (Python, Bash, SQL, Spark jobs).
    - Dependencies (edges) → define execution order.
2. **Dynamic Pipeline Generation**
    - Since DAGs are defined in Python, you can generate tasks dynamically (loops, conditions, configs).
3. **Scheduler**
    - Runs tasks on a defined schedule (like cron but smarter).
    - Can run ad-hoc or backfill (re-run for past dates).
4. **Executor Options**
    - **LocalExecutor, CeleryExecutor, KubernetesExecutor** → scale depending on workload.
5. **Web UI**
    - Visualizes DAGs, dependencies, task status.

- Debug logs available per task.

6. **Extensibility**
   - Operators available for **Python, Bash, SQL, Spark, Hive, Kubernetes, GCP, AWS, Azure**.
   - You can write **custom operators**.
7. **Monitoring & Alerting**
   - Retry policies, failure alerts (email/Slack), SLA monitoring.
8. **Plugins**
   - Extend Airflow with custom hooks, executors, macros, etc.

When is Airflow Used?

- **ETL pipelines** (Extract → Transform → Load)
- **Data Warehousing** (load data into Snowflake/BigQuery/Redshift)
- **Machine Learning pipelines** (train → validate → deploy models)
- **Reporting workflows** (daily report generation)
- **Orchestrating Spark/Hive/Presto jobs**

Building a Simple Data Pipeline tutorial

Prereqs & bring Airflow up (Docker Compose)

```
# get official compose file

curl -LfO 'https://airflow.apache.org/docs/apache-airflow/stable/docker-compose.yaml'

# required folders + env

mkdir -p ./dags ./logs ./plugins

echo -e "AIRFLOW_UID=$(id -u)" > .env

# initialize metadata DB

docker compose up airflow-init

# start the stack (webserver, scheduler, postgres, etc.)

docker compose up
```

Open the UI at **http://localhost:8080** (user/pass: airflow / airflow). Apache Airflow

1) Add a Postgres Connection in the UI

In **Admin → Connections** → + create:

- **Conn Id**: tutorial_pg_conn
- **Conn Type**: Postgres
- **Host**: postgres
- **Database**: airflow
- **Login/Password**: admin / admin
- **Port**: 5432
  Save. (Those values match the Docker services in the compose file.)
  Apache Airflow

2) Create the tables (staging + final)

In your DAG you'll use **SQLExecuteQueryOperator** to:

- Create **employees** (final table, PK = "Serial Number")
- Drop & create **employees_temp** (staging)
  Tip: you can inline the SQL in the operator or point to a .sql file inside dags/sql/. Apache Airflow

3) Load the CSV into the staging table

Write a Python task that:

- Downloads the CSV from the tutorial URL,
- Saves it (e.g., /opt/airflow/dags/files/employees.csv),
- Uses **PostgresHook** to COPY into employees_temp. Apache Airflow

4) Merge/Clean into the final table

Another Python task runs a SQL INSERT … ON CONFLICT DO UPDATE to de-duplicate on "Serial Number" and upsert into **employees**. Apache Airflow

5) Wire it into a DAG and run

The dependency chain is:

[create_employees_table, create_employees_temp_table] >> get_data >> merge_data

Save the file as dags/process_employees.py, enable the DAG in the UI, and click **Trigger**. Use **Grid** or **Graph** to watch tasks and view logs. Apache Airflow

Process_employees.py file

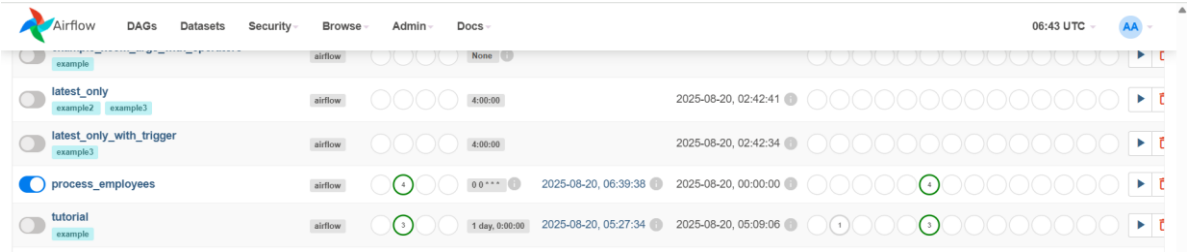
```
# dags/process_employees.py
from airflow.sdk import dag, task
from airflow.providers.common.sql.operators.sql import
SQLExecuteQueryOperator
from airflow.providers.postgres.hooks.postgres import PostgresHook
import pendulum, datetime, os, requests


@dag(dag_id="process_employees", schedule="0 0 * * *",
    start_date=pendulum.datetime(2021,1,1,tz="UTC"), catchup=False,
    dagrun_timeout=datetime.timedelta(minutes=60))
def process_employees():
   create_employees_table = SQLExecuteQueryOperator(
      task_id="create_employees_table", conn_id="tutorial_pg_conn",
      sql="""CREATE TABLE IF NOT EXISTS employees (
            "Serial Number" NUMERIC PRIMARY KEY,
            "Company Name" TEXT, "Employee Markme" TEXT,
            "Description" TEXT, "Leave" INTEGER);"""
   )

   create_employees_temp_table = SQLExecuteQueryOperator(
      task_id="create_employees_temp_table", conn_id="tutorial_pg_conn",
      sql="""DROP TABLE IF EXISTS employees_temp;
            CREATE TABLE employees_temp (
            "Serial Number" NUMERIC PRIMARY KEY,
            "Company Name" TEXT, "Employee Markme" TEXT,
            "Description" TEXT, "Leave" INTEGER);"""
   )
   @task
   def get_data():
      path = "/opt/airflow/dags/files/employees.csv"
```

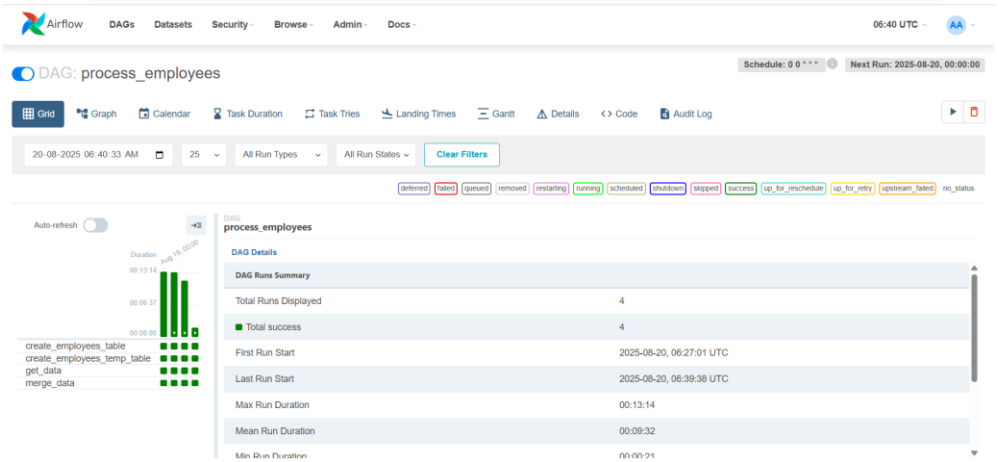
```
        os.makedirs(os.path.dirname(path), exist_ok=True)
        url = "https://raw.githubusercontent.com/apache/airflow/main/airflow-core/docs/tutorial/pipeline_example.csv"
        with open(path, "w") as f:
            f.write(requests.get(url).text)

        hook = PostgresHook(postgres_conn_id="tutorial_pg_conn")
        with hook.get_conn() as conn, open(path, "r") as f:
            conn.cursor().copy_expert(
                "COPY employees_temp FROM STDIN WITH CSV HEADER DELIMITER AS ',' QUOTE '\"'", f
            ); conn.commit()

    @task
    def merge_data():
        upsert = """
        INSERT INTO employees
        SELECT * FROM (SELECT DISTINCT * FROM employees_temp) t
        ON CONFLICT ("Serial Number") DO UPDATE SET
          "Employee Markme"=excluded."Employee Markme",
          "Description"=excluded."Description",
          "Leave"=excluded."Leave";
        """
        hook = PostgresHook(postgres_conn_id="tutorial_pg_conn")
        with hook.get_conn() as conn:
            conn.cursor().execute(upsert); conn.commit()

    [create_employees_table, create_employees_temp_table] >> get_data() >> merge_data()

dag = process_employees()
```

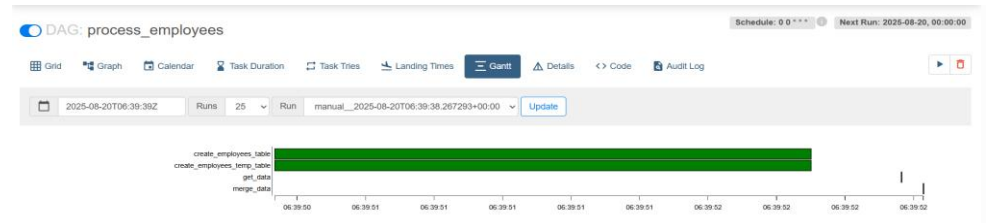
If you're on Airflow 2.x, replace:

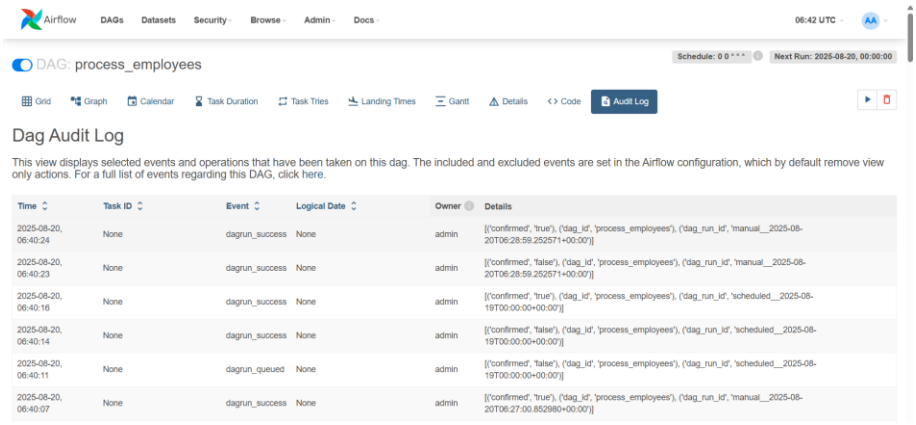
```
from airflow.sdk import dag, task
# with
from airflow.decorators import dag, task
# and use schedule_interval= instead of schedule=
```

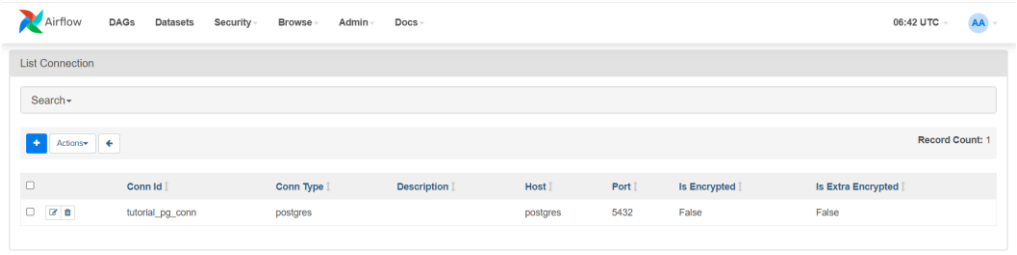
(Everything else stays the same.)








Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
06:42 UTC
AA
DAG: process_employees
Schedule: 0 0 * * *
Next Run: 2025-08-20, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
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
2025-08-20, 06:40:24
None
dagrun_success
None
admin
[('confirmed', 'true'), ('dag_id', 'process_employees'), ('dag_run_id', 'manual__2025-08-20T06:28:59.252571+00:00')]
2025-08-20, 06:40:23
None
dagrun_success
None
admin
[('confirmed', 'false'), ('dag_id', 'process_employees'), ('dag_run_id', 'manual__2025-08-20T06:28:59.252571+00:00')]
2025-08-20, 06:40:16
None
dagrun_success
None
admin
[('confirmed', 'true'), ('dag_id', 'process_employees'), ('dag_run_id', 'scheduled__2025-08-19T00:00:00+00:00')]
2025-08-20, 06:40:14
None
dagrun_success
None
admin
[('confirmed', 'false'), ('dag_id', 'process_employees'), ('dag_run_id', 'scheduled__2025-08-19T00:00:00+00:00')]
2025-08-20, 06:40:11
None
dagrun_queued
None
admin
[('confirmed', 'false'), ('dag_id', 'process_employees'), ('dag_run_id', 'scheduled__2025-08-19T00:00:00+00:00')]
2025-08-20, 06:40:07
None
dagrun_success
None
admin
[('confirmed', 'true'), ('dag_id', 'process_employees'), ('dag_run_id', 'manual__2025-08-20T06:27:00.852980+00:00')]


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
06:42 UTC
AA
List Connection
Search
Actions
Record Count: 1
Conn Id
Conn Type
Description
Host
Port
Is Encrypted
Is Extra Encrypted
tutorial_pg_conn
postgres
postgres
5432
False
False