**Airflow**

**Name: Priyeshwar**

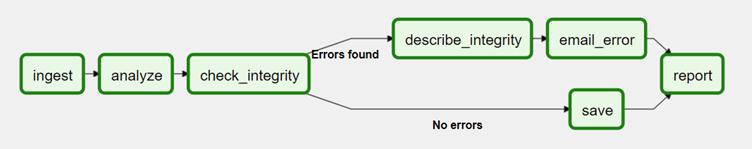
**Mail:** [**priyesh2664@gmail.com**](mailto:priyesh2664@gmail.com)

**What is Airflow?**

Apache Airflow is an open-source workflow orchestration tool used by data engineers to programmatically author, schedule, and monitor workflows (pipelines). Workflows are defined in Python code and represented as Directed Acyclic Graphs (DAGs).

**Airflow Architecture**

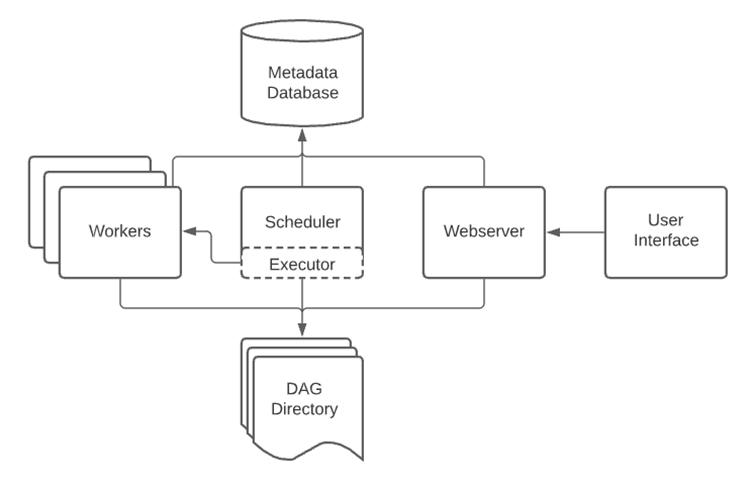
The Airflow platform lets you build and run workflows, which are represented as Directed Acyclic Graphs (DAGs). A sample DAG is shown in the diagram below.



**DAG**

A DAG contains Tasks (action items) and specifies the dependencies between them and the order in which they are executed. A Scheduler handles scheduled workflows and submits Tasks to the Executor, which runs them. The Executor pushes tasks to workers.

Other typical components of an Airflow architecture include a database to store state metadata, a web server used to inspect and debug Tasks and DAGs, and a folder containing the DAG files.



**Control Flow**

In Apache Airflow, control flow defines how tasks within a DAG are executed and related. A DAG can run multiple times, and multiple DAG runs can also happen in parallel. Every DAG must include the mandatory parameter execution\_date, along with other optional parameters that define its behavior.

Dependencies between tasks are represented using the symbols >> (downstream) and << (upstream), for example:

**first\_task >> [second\_task, third\_task] or third\_task << fourth\_task.**

By default, tasks wait for all their upstream tasks to succeed before running, but this behavior can be customized using features like LatestOnly (to run tasks only in the most recent DAG run), Branching (to choose different execution paths dynamically), and Trigger Rules (to allow tasks to run under different conditions, such as when some upstream tasks fail). To manage complexity, Airflow also provides SubDAGs, which embed reusable DAGs inside others, and TaskGroups, which allow visually grouping related tasks in the user interface.

**Apache Airflow Setup with Docker & VS Code**

**1. Install Required Tools**

* **Docker Desktop** → [Download here](https://www.docker.com/products/docker-desktop/)
* **Visual Studio Code** → [Download here](https://code.visualstudio.com/)

Make sure Docker Desktop is **running** before continuing.

**2. Prepare Materials Folder**

1. Create a directory:
2. C:\Users\<your\_username>\materials
3. Save a file named **docker-compose.yaml** in that folder.
4. Open **Visual Studio Code** → File → Open Folder → select the materials folder.

**3. Create Environment File**

Inside materials, create a file named **.env** with these contents:

AIRFLOW\_IMAGE\_NAME=apache/airflow:2.4.2

AIRFLOW\_UID=50000

**4. Start Airflow**

In **VS Code terminal**, run:

docker-compose up -d

This will download Airflow images and start containers (webserver, scheduler, worker, etc.).

**5. Create Admin User**

Once containers are running, create an **admin user**:

docker-compose run airflow-worker airflow users create \

--role Admin \

--username admin \

--email admin@example.com \

--firstname admin \

--lastname admin \

--password admin

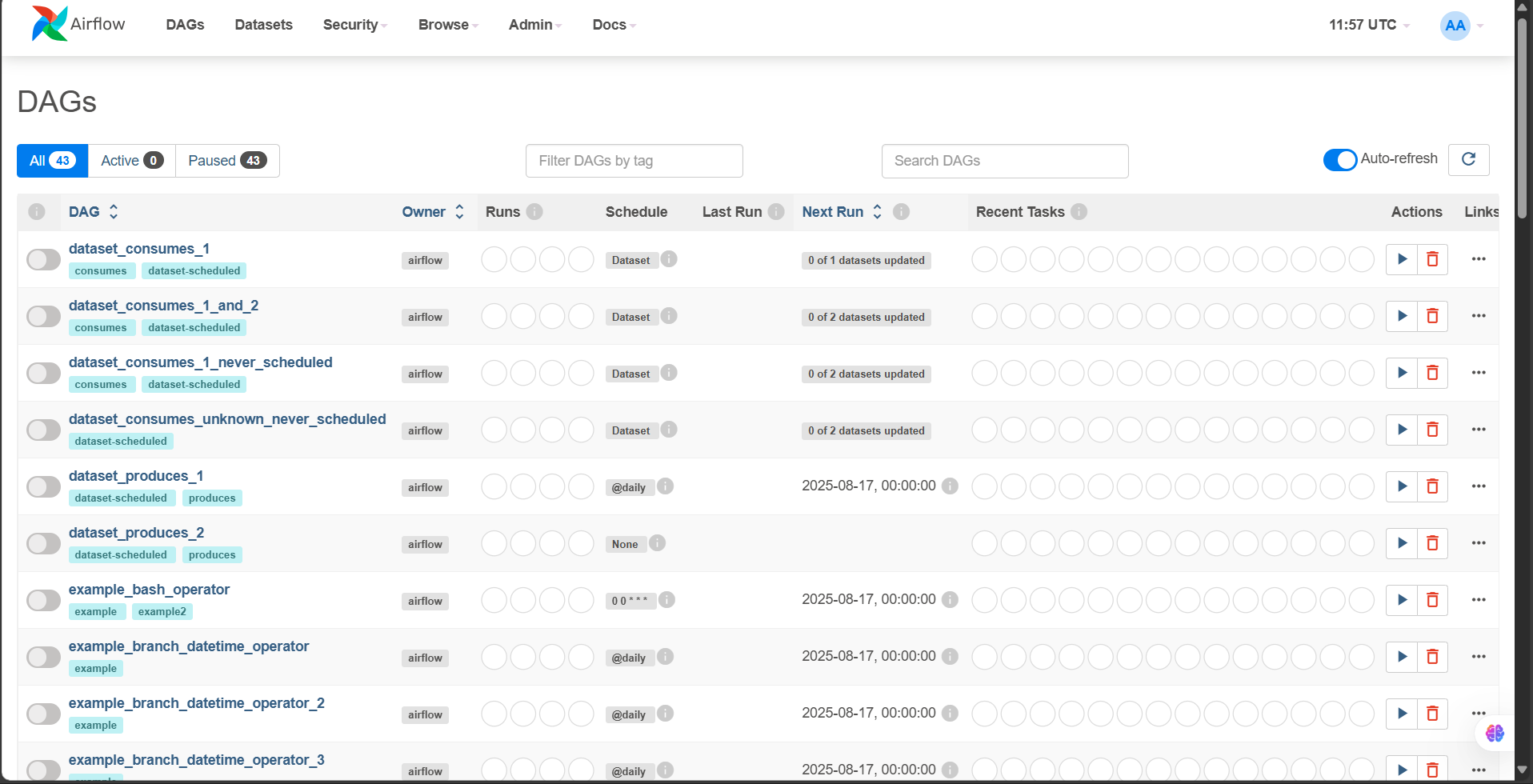
**6. Access the Airflow UI**

Open your browser and go to:

[http://localhost:8080](http://localhost:8080/)

Log in with:

* **Username:** admin
* **Password:** admin



At this point, Airflow should be up and running inside Docker.

