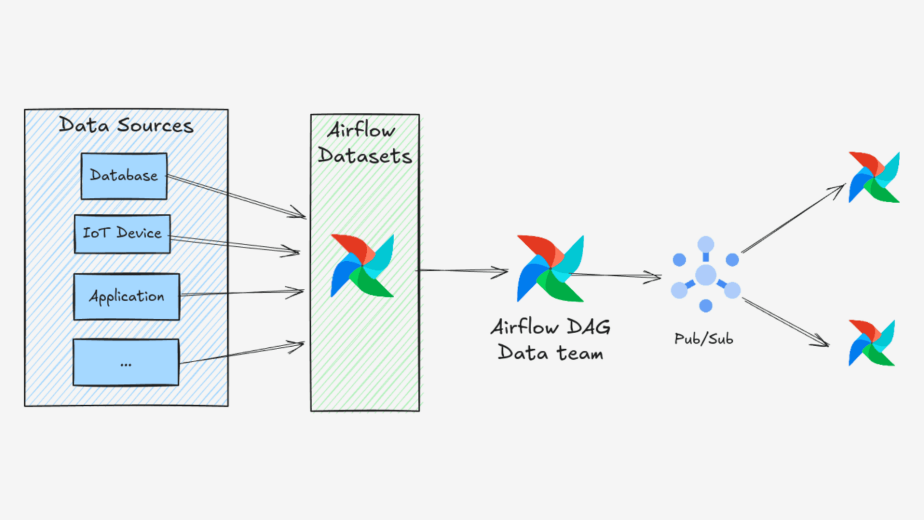
AIRFLOW

Apache Airflow is an open-source platform designed to programmatically author, schedule, and monitor workflows. Developed by Airbnb and later donated to the Apache Software Foundation, Airflow allows users to define workflows as Directed Acyclic Graphs (DAGs), where each node represents a task, and edges define dependencies between tasks.

It checks whether you're managing data pipelines, orchestrating machine learning tasks, or handling cloud-native operations, Airflow provides an intuitive, scalable, and extensible framework to handle it all.

It allows users to define tasks and their dependencies as DAGs in Python, making it a powerful tool for data pipeline automation, ETL processes, and task orchestration.



How Airflow Works

1. Writing DAGs (Workflows in Python)

Users define workflows as **DAGs (Directed Acyclic Graphs)**, where each node represents a task, and edges define dependencies. DAGs are written in Python, making them flexible and easy to integrate with other tools.

2. Airflow Scheduler

The **Scheduler** checks the DAGs and determines when tasks should run based on schedules or triggers. It ensures tasks execute in the correct order, respecting dependencies.

3. Executor Running Tasks

The **Executor** is responsible for running tasks. Different types (LocalExecutor) allow scaling across workers. Tasks can include Python functions, Bash commands, SQL queries, or API calls.

4. Web UI for Monitoring & Debugging

Airflow's **Web UI** provides real-time monitoring of task statuses (success, failure, retries). Users can manually trigger, rerun, or inspect logs for debugging.

Components of Apache Airflow

1. DAG (Directed Acyclic Graph)

A DAG is the fundamental concept in Airflow, representing a collection of tasks with defined execution order. Key characteristics:

- **Directed** – Tasks have a clear sequence.
- **Acyclic** – No loops or circular dependencies.
- **Defined in Python** – DAGs are written as Python scripts, allowing dynamic generation.

2. Operators

Operators define individual tasks within a DAG. Common types include:

- **BashOperator** – Executes a bash command.
- **PythonOperator** – Runs a Python function.
- **EmailOperator** – Sends emails.

3. Tasks

A task is an instance of an operator within a DAG. Tasks can have dependencies (e.g., task1 >> task2 means task2 runs after task1).

4. Scheduler

The scheduler is responsible for triggering tasks based on their dependencies and schedule intervals. It parses DAGs, checks task states, and submits tasks to the executor.

5. Executor

The executor determines how tasks are run. Types include:

- **SequentialExecutor** – Runs tasks one at a time (for testing).
- **LocalExecutor** – Runs tasks in parallel on a single machine.
- **CeleryExecutor** – Distributes tasks across multiple workers.

6. Web Server

Airflow's web-based UI provides:

- **DAG Visualization** – Graph and tree views of workflows.
- **Task Logs** – Access to execution logs.
- **Task Management** – Manual triggering and retrying of tasks.

7. Metadata Database

Airflow uses a relational database (e.g., PostgreSQL, MySQL) to store:

- DAG definitions
- Task execution history
- User roles and permissions

8. Workers

In distributed setups (e.g., Celery or Kubernetes), workers execute tasks assigned by the scheduler.

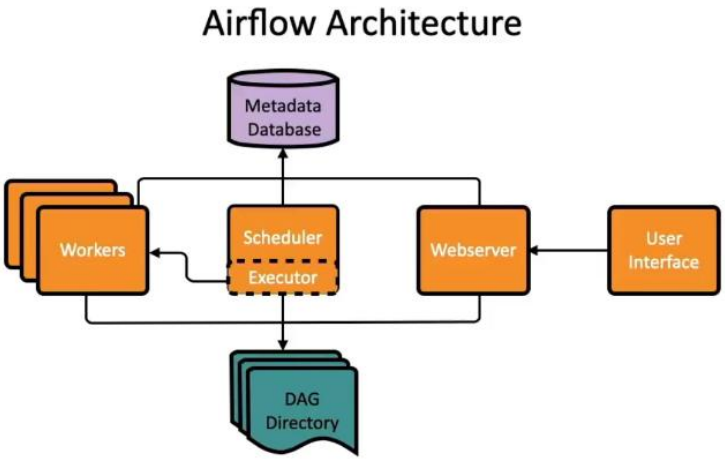
9. Hooks

Hooks provide interfaces to external systems (e.g., databases, APIs). Examples:

- **MySqlHook** – Interacts with MySQL databases.
- **HttpHook** – Makes HTTP requests.

10. Connections

Airflow stores credentials for external systems (e.g., database logins, API keys) in the metadata database



.