Apache Airflow

Apache Airflow is an open-source platform created by Airbnb and later donated to the Apache Software Foundation. It is designed to programmatically author, schedule, and monitor workflows. Airflow enables users to manage complex workflows by defining them as Directed Acyclic Graphs (DAGs) using Python. This “workflow as code” philosophy makes Airflow dynamic, extensible, and easy to integrate into modern data engineering practices.

Airflow is widely used in the data engineering ecosystem to automate Extract, Transform, Load (ETL) processes, machine learning pipelines, and other batch processing tasks. Its modular and scalable design enables teams to orchestrate pipelines across local machines, distributed systems, or cloud environments.

Components

1. DAG (Directed Acyclic Graph)

A DAG is the core concept in Airflow. It represents the structure of a workflow as a series of tasks with defined dependencies and schedules. A DAG must be acyclic, meaning tasks cannot loop back to earlier tasks, ensuring that the workflow finishes eventually. Each DAG has parameters such as start date, schedule interval, and task definitions.

2. Task

Tasks are the building blocks of a DAG. Each task represents a single unit of work, such as running a Python script, executing a SQL query, or triggering an API. Tasks in Airflow are instances of operators. Tasks can be arranged with dependencies to enforce order of execution.

3. Operators

Operators define what actually gets done by a task. Airflow includes a variety of built-in operators, such as:

- `PythonOperator`: Executes a Python function.
- `BashOperator`: Runs a bash command.
- `EmailOperator`: Sends an email.

- `DummyOperator`: Acts as a placeholder or for branching.
  Operators can also be customized to create custom plugins for specific use cases.

4. Scheduler

The Scheduler is a background process in Airflow that monitors DAG definitions and triggers task instances at scheduled intervals. It evaluates DAGs based on their schedule interval and orchestrates the task execution according to the task dependencies and trigger rules.

5. Executor

Executors are responsible for executing the tasks. Airflow supports different types of executors based on deployment needs:

- **SequentialExecutor**: Executes tasks one at a time (suitable for testing).
- **LocalExecutor**: Executes tasks in parallel using the local machine.
- **CeleryExecutor**: Distributes tasks across a cluster of worker nodes.
- **KubernetesExecutor**: Executes tasks in dynamically created pods in Kubernetes.

6. Metadata Database

The metadata database is the source of truth for Airflow. It stores all state information related to DAGs, task instances, variables, connections, and logs. Airflow supports several relational databases such as PostgreSQL and MySQL for this purpose.

7. Web Server (Web UI)

The Web UI is an important component that provides a graphical interface for users to manage and monitor DAGs. It allows users to:

- View DAGs and task status
- Trigger or pause DAGs manually
- Access logs of past runs
- Clear or retry failed tasks

8. Workers

In a distributed setup, workers are the nodes that actually execute the tasks. When using executors like Celery or Kubernetes, tasks are queued by the scheduler and picked up by available workers. These workers run the logic defined in each operator.

9. XComs (Cross-Communication Messages)

XComs allow tasks to exchange messages or small pieces of data during workflow execution. These are useful when one task's output is required by another task in the same DAG.