**APACHE AIRFLOW**

Apache Airflow is an open-source platform used to programmatically author, schedule, and monitor workflows, particularly data pipelines. It allows users to define workflows as Directed Acyclic Graphs (DAGs) of tasks, making it a robust tool for orchestrating complex data engineering pipelines.

**Key Features and Functionality of Apache Airflow:**

Apache Airflow is a tool used to create, schedule, and monitor workflows. It is widely used in data engineering and automation tasks. Below are some of its important features:

**1. Workflow Definition**

Airflow lets you create workflows using Python code. This makes it easy to build, update, and manage workflows in a flexible and dynamic way.

**2. Scheduling and Monitoring**

Airflow provides a web-based user interface to:

* Schedule workflows to run at specific times.
* Monitor the progress of tasks.
* View task logs and history.
* Visualize task dependencies.

**3. Extensibility**

Airflow supports many built-in operators to connect with tools like:

* Databases (e.g., MySQL, PostgreSQL)
* Cloud platforms (AWS, Azure, Google Cloud)

You can also create custom operators to fit your own needs.

**4. Scalability and Flexibility**

Airflow can run on one machine or scale across many machines. It can handle:

* Small or large workflows
* Simple or complex business logic

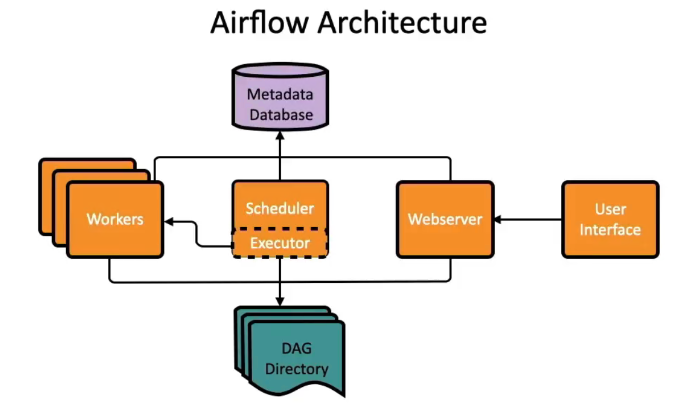
**5. Idempotency**

Airflow encourages idempotent tasks, meaning tasks can run multiple times without causing duplicate or incorrect data.

**6. Common Use Cases**

Airflow is used for:

* ETL and ELT pipelines
* Data processing tasks
* Machine learning workflows
* Automating data transfers between systems



**Core Components of Apache Airflow**

Apache Airflow is a powerful platform used to programmatically author, schedule, and monitor workflows. The following are the **six most important components** that form the foundation of Airflow’s architecture:

**1. DAG (Directed Acyclic Graph)**

At the heart of Airflow lies the DAG, which represents the workflow to be executed. A DAG is a collection of tasks with directional dependencies, ensuring tasks are executed in a specific order without any cyclic dependencies.

DAGs are defined using Python scripts, allowing for dynamic generation and modification. This flexibility enables users to define complex workflows in a maintainable and scalable way.

**2. Tasks and Operators**

Each DAG is composed of tasks—the fundamental units of work. A task is an instance of an operator in a DAG. It represents a single unit of work.

* Tasks have attributes like task\_id, retries, execution timeout, and trigger rules.
* Task dependencies are set using set\_upstream() or >> / << operators.
* Tasks can have retry mechanisms, SLA monitoring, and custom logic.

Airflow provides several built-in operators such as:

* **PythonOperator**: Executes Python functions.
* **BashOperator**: Runs bash commands or scripts.
* **EmailOperator**: Sends emails.
* **DummyOperator**: Placeholder for logical structuring.
* **MySqlOperator / PostgresOperator**: Executes SQL commands.

Each time a DAG is run, individual task instances are created, representing the execution of a task at a specific time.

**3. Scheduler**

The Scheduler is responsible for determining when each task should run. Based on the schedule defined in the DAG (e.g., daily at midnight), it triggers task execution.

It continuously monitors all DAGs, respects task dependencies, and manages retries in case of failures. The scheduler ensures tasks are run in the correct order and at the correct time.

**4. Executor**

The Executor defines how and where tasks are executed. While it does not run the tasks directly, it delegates the execution to workers based on the configured type:

* **LocalExecutor**: Executes tasks locally (single or multi-process).
* **CeleryExecutor**: Distributes tasks across multiple workers using a message broker.
* **KubernetesExecutor**: Runs each task in its own Kubernetes pod.

Choosing the right executor impacts scalability, resource usage, and performance.

**5. Web Server (Web UI)**

The Web Server provides a rich user interface to interact with Airflow. Through this web-based UI, users can:

* Monitor DAG runs and task statuses.
* Trigger or pause DAGs manually.
* View logs and execution history.
* Retry or clear failed tasks.

This centralized dashboard enables real-time monitoring and control over workflows.

**6. Metadata Database**

Airflow relies on a metadata database to store all critical information related to DAGs and task executions. This includes:

* DAG definitions and schedules
* Task instance states (success, failure, retries)
* Logs and user actions

These components work together to provide a robust and scalable system for orchestrating complex data workflows in Airflow.