**APACHE AIRFLOW**

WHAT IS AIRFLOW:

Apache Airflow is an **open-source workflow orchestration platform** designed to programmatically author, schedule, and monitor workflows (called DAGs).

* Developed originally at **Airbnb** (2014).
* Donated to the **Apache Software Foundation** (2016).
* Widely used in **data engineering, machine learning pipelines, ETL, and DevOps automation**.

Instead of manually managing workflows, Airflow lets you:

* Write pipelines as **Python code**.
* Schedule them.
* Monitor execution visually via the **Airflow Web UI**.

CORE CONCEPTS:

**DAG (Directed Acyclic Graph)**

* Collection of tasks with dependencies.
* Defines the execution order (no cycles allowed).

**Task**

* A single unit of work (e.g., running SQL, calling an API, transforming data).
* Implemented via **Operators**.

**Operator**

* Template for a task (e.g., PythonOperator, BashOperator, MySqlOperator).

**Scheduler**

* Decides when tasks should run.

**Executor**

* Decides *where/how* tasks run (e.g., LocalExecutor, CeleryExecutor, KubernetesExecutor).

**Web UI**

* Dashboard to visualize DAGs, track runs, retry failures, and manage tasks.

ADVANTAGES:

**Code-as-Configuration**

* Workflows written in Python → flexible, reusable, version-controllable.

**Scalable**

* Supports distributed execution (Celery, Kubernetes).
* Handles thousands of tasks daily.

**Dynamic DAGs**

* Pipelines generated programmatically.
* Easy parameterization.

**Monitoring & Alerts**

* Built-in retry logic, error notifications (Slack, Email).

**Integrations**

* Works with AWS, GCP, Azure, Hadoop, Spark, Databricks, databases, etc.

**Community & Ecosystem**

* Large open-source community.
* Rich set of pre-built operators & hooks.

LIMITATIONS:

* **Not real-time** (best for batch processing, not event-driven).
* **Setup overhead** (needs database + executor + scheduler).
* **Steep learning curve** for complex DAGs.
* **Scaling complexity** (requires Celery/K8s for distributed mode).

ARCHITECTURE:

1. **Web Server** → UI & REST API.
2. **Scheduler** → Decides what to run and when.
3. **Executor** → Runs tasks (local, Celery workers, Kubernetes pods).
4. **Metadata Database (Postgres/MySQL)** → Stores DAGs, task states, logs.
5. **Workers** → Actually execute the tasks.