APACHE AIRFLOW

- Apache Airflow is an open-source platform used to programmatically author, schedule, and monitor workflows. It is widely used for orchestrating complex computational workflows and data pipelines.
- **Key Features:**
    - Workflow orchestration using Directed Acyclic Graphs (DAGs)
    - Dynamic pipeline generation using Python
    - Scalability and extensibility
    - Integration with various databases, cloud services, and data tools
    - Real-time monitoring and logging of tasks

Architecture

Apache Airflow follows a modular architecture. Its key components include:

- Scheduler:

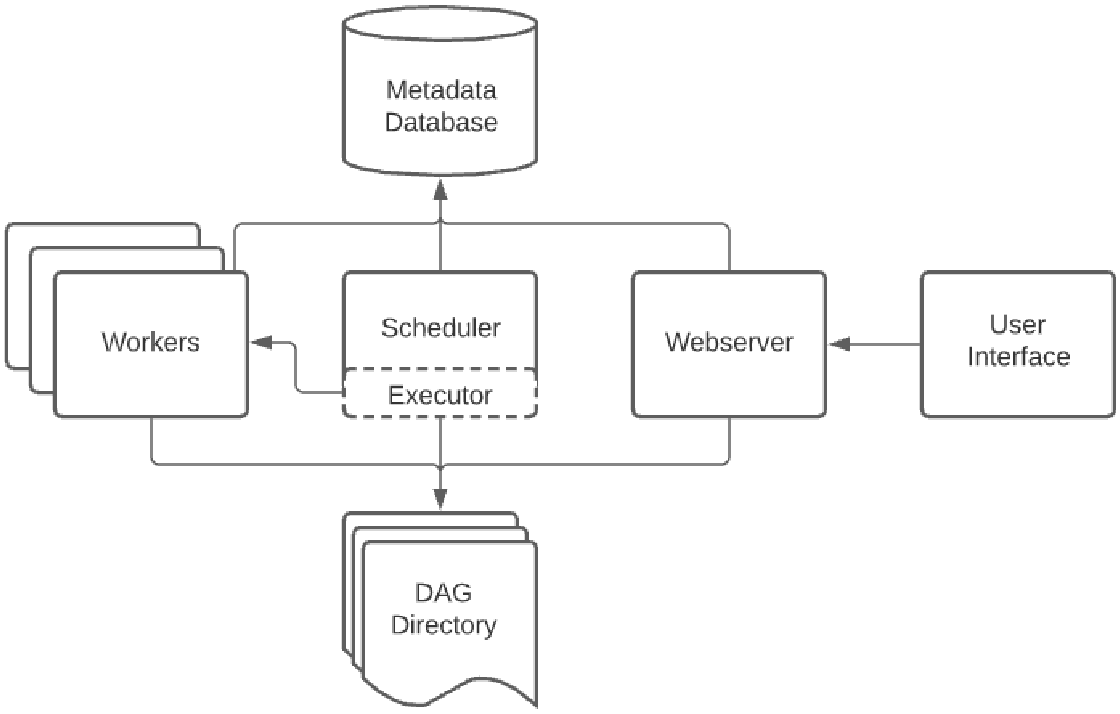
    Responsible for scheduling DAGs and triggering tasks based on dependencies and timing.
- Executor:

    Executes the tasks. Types include SequentialExecutor, LocalExecutor, CeleryExecutor, and KubernetesExecutor.
- Metadata Database:

    Stores DAG definitions, task status, and runtime information. Commonly uses PostgreSQL or MySQL.
- Web Server:

    Provides a UI to monitor workflows, view logs, and manage DAGs.
- Workers:

    Execute tasks in distributed setups (used with Celery/Kubernetes executors).

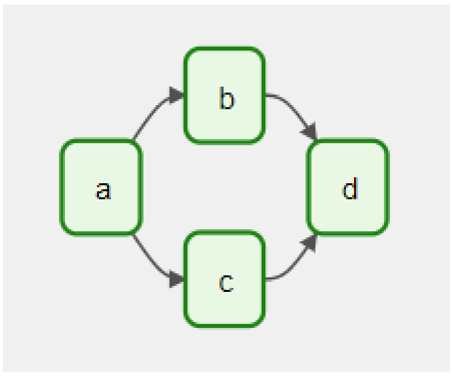


CONCEPTS :

1. **DAG (Directed Acyclic Graph):**
   - Represents the workflow. Each node is a task, and edges represent dependencies.
2. **Task:**
   - A single unit of work in a DAG. Example: running a Python script, transferring data, or executing SQL queries.
3. **Operator:**
   - Predefined task templates. Types include:
     - PythonOperator: Execute Python functions
     - BashOperator: Run bash commands
     - EmailOperator: Send emails
     - PostgresOperator: Execute SQL on PostgreSQL
4. **Task Instance:**
   - A specific run of a task at a particular time.
5. **Scheduler Interval:**
   - Defines how often a DAG should run (e.g., @daily, @hourly).
6. **Hooks and Connections:**
   - Connectors to databases, cloud services, and APIs.

DAGs

- A *DAG* (Directed Acyclic Graph) is the core concept of Airflow, collecting Tasks together, organized with dependencies and relationships to say how they should run.



- It defines four Tasks - A, B, C, and D - and dictates the order in which they have to run, and which tasks depend on what others.
- It will also say how often to run the DAG - maybe "every 5 minutes starting tomorrow", or "every day

Use Cases

- ETL (Extract, Transform, Load) pipelines
- Data warehouse loading
- Machine learning model training pipelines
- Batch job scheduling
- Data quality monitoring

Advantages:

- Flexible and Python-based
- Scalable and extensible
- Active community support
- Rich UI for monitoring

Limitations:

- Learning curve for beginners
- Complex setup in distributed environments
- Not suitable for real-time streaming tasks (use Apache Kafka or Spark Streaming for real-time)