APACHE AIRFLOW

Apache Airflow is an open-source platform used to programmatically author, schedule, and monitor workflows. It is designed to manage complex computational workflows by defining tasks and dependencies as code using Python. Airflow allows developers and data engineers to orchestrate workflows that are robust, scalable, and easy to manage.

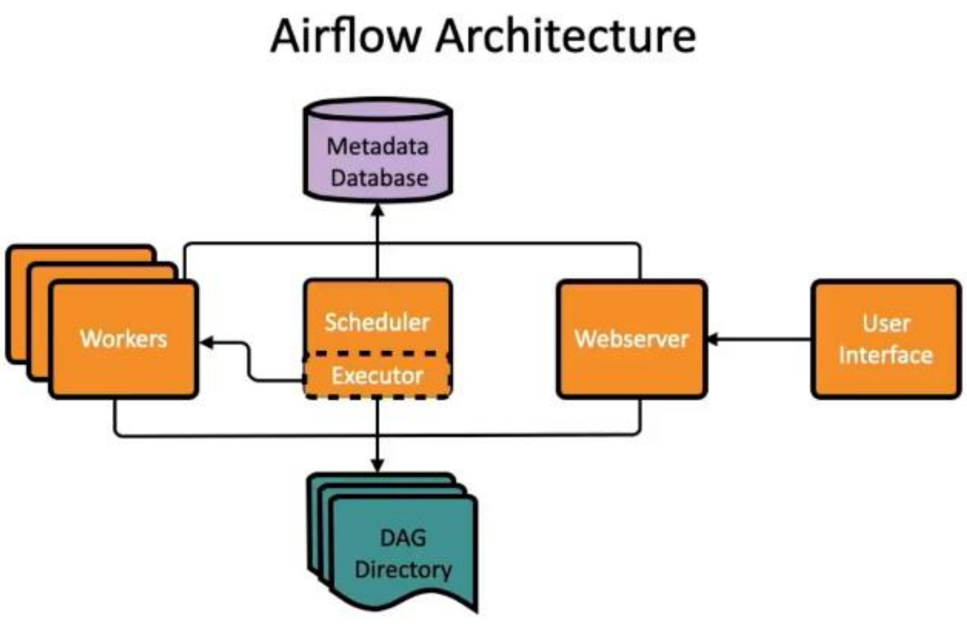
Airflow's core strength lies in its ability to handle dynamic pipeline generation. Each workflow is represented as a Directed Acyclic Graph (DAG), where tasks are nodes and dependencies are edges. The platform provides a web-based interface for tracking progress, viewing logs, and managing tasks.

WHY DOCKER??

Installing Airflow manually can be a complex process due to its various dependencies (e.g., PostgreSQL, Celery, Redis). Docker simplifies this process by containerizing each component, ensuring consistency across environments and reducing setup time. Using Docker Compose, multiple services can be defined and managed easily.

Running Airflow in Docker provides the following benefits:

- Isolated and reproducible environment
- Pre-configured containers for core components (webserver, scheduler, database, etc.)
- Simplified dependency and version management
- Faster onboarding for new developers



SYSTEM ARCHITECTURE

Airflow operates using the following core components:

- Webserver: Hosts the user interface for monitoring and managing workflows.
- Scheduler: Continuously polls DAGs and schedules task executions.
- Workers: Execute the actual tasks as defined in DAGs.
- Database (Metadata DB): Stores the state and history of all tasks and DAGs.
- Triggerer (optional): Manages asynchronous task execution (e.g., sensors).
- Redis (optional): Used as a message broker when using the CeleryExecutor.

Basic Setup Using Docker and WSL

Prerequisites

- Windows with WSL 2 installed and enabled
- Docker Desktop installed with WSL integration
- Git installed

Clone the official Airflow repository:

```
git clone https://github.com/apache/airflow.git
cd airflow
```

Copy the example environment file and initialize:

```
cp .env.example .env
```

```
docker compose up airflow-init
```

WORKFLOW EXECUTION IN AIRFLOW

Each DAG file in Airflow defines a pipeline of tasks, their order, and dependencies. The scheduler evaluates these DAGs based on their defined schedule, places them in a queue, and assigns them to workers for execution.

Workflows can be triggered manually or based on schedules (e.g., cron expressions). Tasks can retry upon failure, wait for external events, and pass data between steps. Logs and execution history are stored in the metadata database for tracking and debugging.

Airflow encourages "configuration as code", enabling version control and reusability of workflows. It also supports plugins and custom operators to extend its functionality.

CONCLUSION

Apache Airflow provides a powerful and scalable solution for orchestrating workflows and data pipelines. When combined with Docker and WSL, the setup becomes straightforward, portable, and efficient for local development and testing. Its modular architecture, rich UI, and Python-native design make it a valuable tool for managing modern data workflows in both development and production environments.