iykra Data Fellowship

IYKRA
Data Fellowship Program

Pipeline Design Patterns with Airflow

by Fariz Wakan

May 13th, 2024

Table of Content

Data Fellowship

Apache Airflow Basics

Apache Airflow is ..

Apache Airflow™ is an **open-source platform for developing, scheduling, and monitoring batch-oriented workflows**. Airflow's extensible Python framework enables you to **build workflows connecting with virtually any technology**.

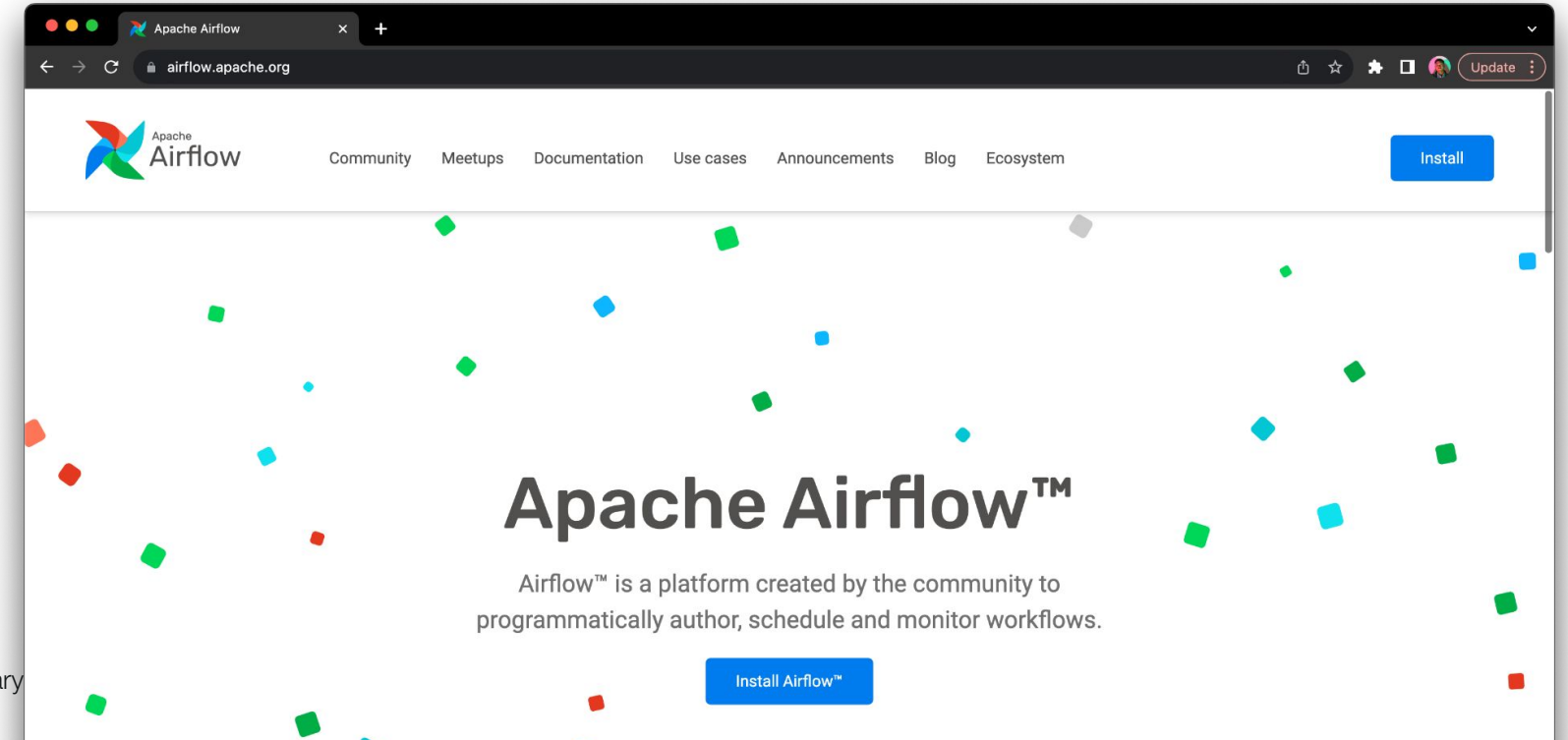
A web interface helps manage the state of your workflows.

Airflow is deployable in many ways, varying from a single process on your laptop to a distributed setup to support even the biggest workflows.

Workflow as code

- **Dynamic**: Airflow pipelines are configured as Python code, allowing for dynamic pipeline generation.
- **Extensible**: The Airflow™ framework contains operators to connect with numerous technologies. All Airflow components are extensible to easily adjust to your environment.
- **Flexible**: Workflow parameterization is built-in leveraging the *Jinja* templating engine.

Apache Airflow



Why Apache Airflow

Airflow™ is a **batch workflow orchestration platform**. The Airflow framework contains operators to connect with many technologies and is easily extensible to connect with a new technology. If your workflows have a clear start and end, and run at regular intervals, they can be programmed as an Airflow DAG.

Why Apache Airflow

If you prefer coding over clicking, Airflow is the tool for you. Workflows are defined as Python code which means:

- Workflows can be stored in version control so that you can roll back to previous versions
- Workflows can be developed by multiple people simultaneously
- Tests can be written to validate functionality
- Components are extensible and you can build on a wide collection of existing components

Why not Apache Airflow

Airflow™ was **built for finite batch workflows**. While the CLI and REST API do allow triggering workflows, Airflow was not built for infinitely running event-based workflows. Airflow is **not a streaming solution**.

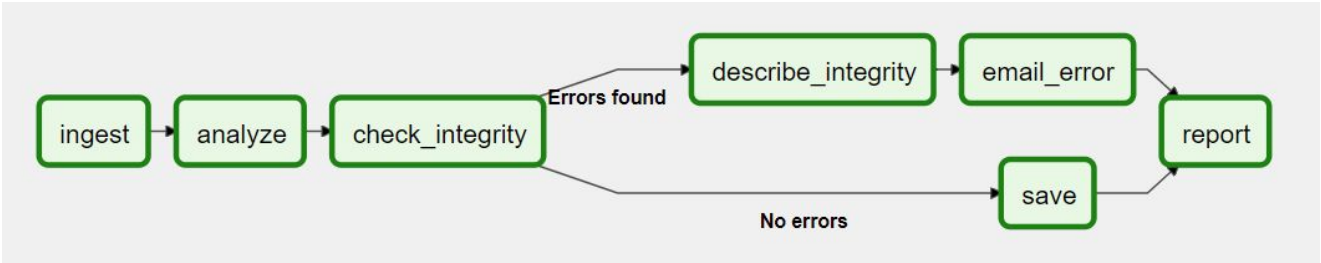
However, a streaming system such as Apache Kafka is often seen working together with Apache Airflow. Kafka can be used for ingestion and processing in real-time, event data is written to a storage location, and Airflow periodically starts a workflow processing a batch of data.

Data Fellowship

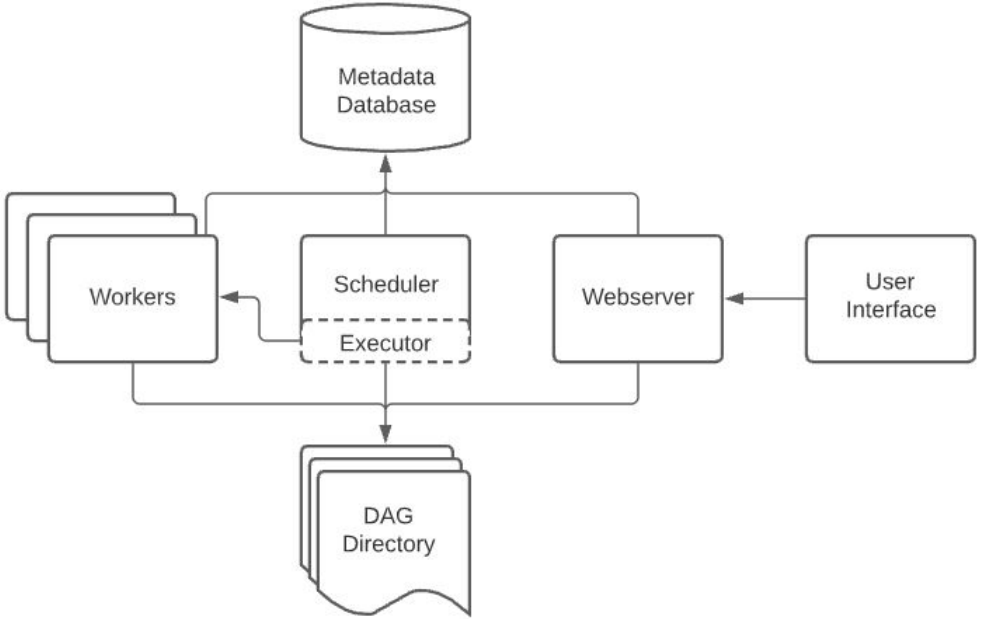
Airflow Core Concepts

DAG

Airflow is a platform that lets you build and run workflows. A workflow is represented as a **DAG (a Directed Acyclic Graph)**, and contains *individual pieces of work called Tasks*, arranged with *dependencies* and *data flows* taken into account.



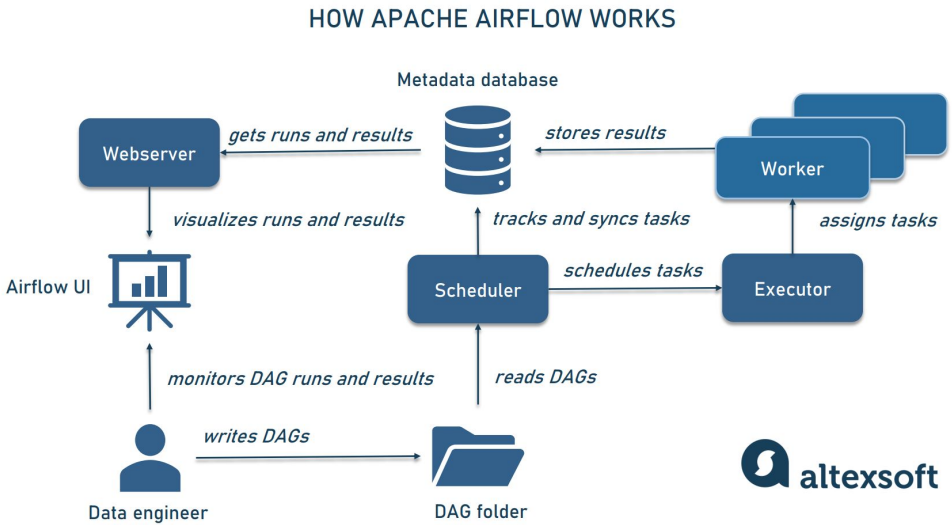
Airflow Components



Airflow Components

- A **scheduler**, which handles both triggering scheduled workflows, and submitting Tasks to the executor to run.
- An **executor**, which handles running tasks.
- A **webserver**, which presents a handy user interface to inspect, trigger and debug the behaviour of DAGs and tasks.
- A **folder of DAG files**, read by the scheduler and executor (and any workers the executor has)
- A **metadata database**, used by the scheduler, executor and webserver to store state.

Airflow Components



Airflow Workloads

A DAG runs through a series of Tasks, and there are *three common types of task* you will see:

- **Operators**, predefined tasks that you can string together quickly to build most parts of your DAGs.
- **Sensors**, a special subclass of Operators which are entirely about waiting for an external event to happen.
- A **TaskFlow**-decorated @task, which is a custom Python function packaged up as a Task.

Airflow Control Flow

Tasks have dependencies declared on each other. You'll see this in a DAG either using the >> and << operators.

These dependencies are what make up the "edges" of the graph, and how Airflow works out which order to run your tasks in. By default, **a task will wait for all of its upstream tasks to succeed before it runs**.

Pas Data Between Tasks

To pass data between tasks you have three options:

- **XComs** (“Cross-communications”), a system where you can have tasks push and pull small bits of metadata.
- Uploading and downloading large files from a **storage service** (either one you run, or part of a public cloud)
- **TaskFlow API** automatically passes data between tasks via implicit XComs

Hands-on: Hello World with Airflow

Data Fellowship

Hands-on:
ETL with Airflow

Thank you!

See you in the next session 💪