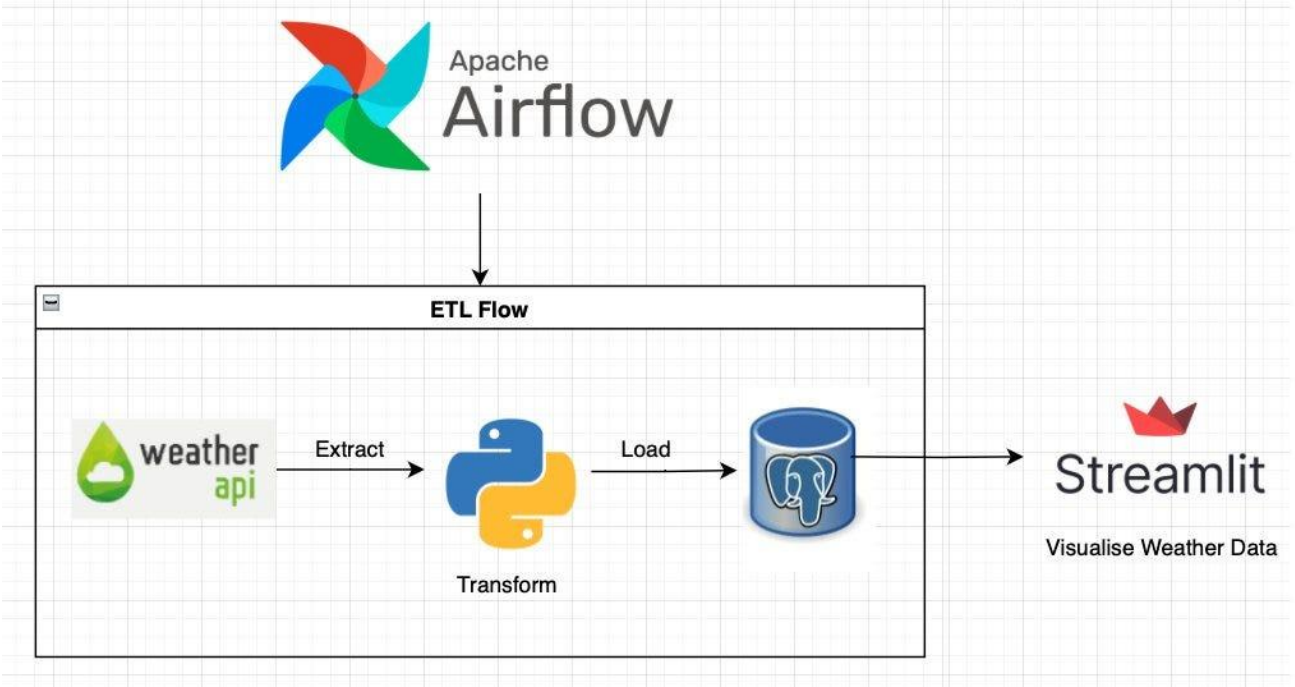
Apache Airflow

What is Airflow?

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-based UI helps you visualize, manage, and debug your workflows. You can run Airflow in a variety of configurations — from a single process on your laptop to a distributed system capable of handling massive workloads.



Why Airflow?

Airflow is a platform for orchestrating batch workflows. It offers a flexible framework with a wide range of built-in operators and makes it easy to integrate with new technologies.

If your workflows have a clear start and end and run on a schedule, they're a great fit for Airflow DAGs.

If you prefer coding over clicking, Airflow is built for you. Defining workflows as Python code provides several key benefits:

- Version control: Track changes, roll back to previous versions, and collaborate with your team
- Team collaboration: Multiple developers can work on the same workflow codebase.
- Testing: Validate pipeline logic through unit and integration tests.
- Extensibility: Customize workflows using a large ecosystem of existing components — or build your own.

Airflow's rich scheduling and execution semantics make it easy to define complex, recurring pipelines. From the web interface, you can manually trigger DAGs, inspect logs, and monitor task status. You can also backfill DAG runs to process historical data, or rerun only failed tasks to minimize cost and time.

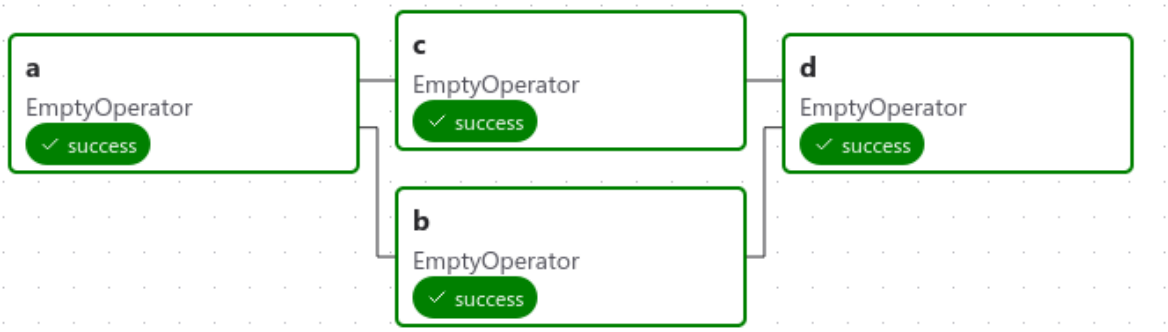
DAG

A DAG is a model that encapsulates everything needed to execute a workflow. Some DAG attributes include the following:

- Schedule: When the workflow should run.
- Tasks: tasks are discrete units of work that are run on workers.
- Task Dependencies: The order and conditions under which tasks execute.

- Callbacks: Actions to take when the entire workflow completes.
- Additional Parameters: And many other operational details.

Eg:-



Tasks

A Task is the basic unit of execution in Airflow. Tasks are arranged into Dags, and then have upstream and downstream dependencies set between them in order to express the order they should run in.

Executer

Executors are the mechanism by which task instances get run. They have a common API and are “pluggable”, meaning you can swap executors based on your installation needs.

Executors are set by the executor option in the [core] section of the configuration file.

Built-in executors are referred to by name, for example:

```
[core]
executor = KubernetesExecutor
```

Custom or third-party executors can be configured by providing the module path of the executor python class, for example:

```
[core]
executor = my.custom.executor.module.ExecutorClass
```

Operator

An Operator is conceptually a template for a predefined Task, that you can just define declaratively inside your DAG:

```
with DAG("my-dag") as dag:
    ping = HttpOperator(endpoint="http://example.com/update/")
    email = EmailOperator(to="admin@example.com", subject="Update
complete")

    ping >> email
```

TaskFlow

If you write most of your dags using plain Python code rather than Operators, then the TaskFlow API will make it much easier to author clean dags without extra boilerplate, all using the @task decorator.

TaskFlow takes care of moving inputs and outputs between your Tasks using XComs for you, as well as automatically calculating dependencies - when you call a TaskFlow function in your DAG file, rather than executing it, you will get an object representing the XCom for the result (an XComArg), that you can then use as inputs to downstream tasks or operators.