Airflow core components

1. Web server
2. Data timer
3. Meta data
4. Trigger
5. Executer
6. Queue
7. The worker

Core concept

Concept 1: DAG (Directed Acyclic Graph).

It helps to complete all the task in order just like making a dish based on the recipe list.

Concept 2: Operator

Operators allow to breakdown our workflow, into discrete, manageable piece of work.

Has 1000 of operator

* Python
* Sql
* Bash
* Filesensor to wait for file

Concept 3: Task/Task Instance.

A task is a specific instance of the operator. When a operator is assigned to the DAG, it becomes the task.

Concept 4: Workflow

A workflow is a entire process of the DAG including all task and dependencies.

Airflow:

Not for Data processing

Not for real time

Not for Data Storage

**TWO TYPE OF ARCHITECTURE:**

1. Single node Architecture:

* A node is nothing but a computer or server
* Single node means installing and running in a single system

Node:

* First Web server
* Scheduler
* 3rd Executer
* 4th Queue handeled in memory -> part of the executor it self.
* Then the worker where it pulls the task from Queue
* Finally it has the Meta Database which is light weight

When to use the single node architecture.

1. Great to get started or small workflows
2. Simple to setup and manage
3. But doesn’t scale.

***Multi-node architecture:***

* Larger work loads
* This is a typical setup when airflow runs in production to ensure performance, scalability and reliability.

Various node can be used for the performing the components for the Airflow.

Where – Node A:

* Where it can have multiple Web servers running in which we can use the load balancer within it.

Then in Node B:

* We can have the scheduler (Multiple) for the reliability,

And the worker can be in node d, e, f.

In which we need to use the message broker to connect for the task. Where the message broker is nothing but the **‘RabbitMQ’**

***Then the node C as the meta database similar to the Postgreys***

We can setup the nodes and Components as needed.

**Advantages of using the multi-node architecture.**

* Scalability -> You can add more worker to make more task
* Reliability -> If one machine fails, other machine takes the responsibility
* Performance -> By distributing work across machines, you can process more tasks simultaneously.

**But more complex.**

*Workflow:*