What is the Airflow?

pro-gram-ma-ble sched-uled mon-i-tored
Airflow is a programmable, scheduled and monitored workflow platform.

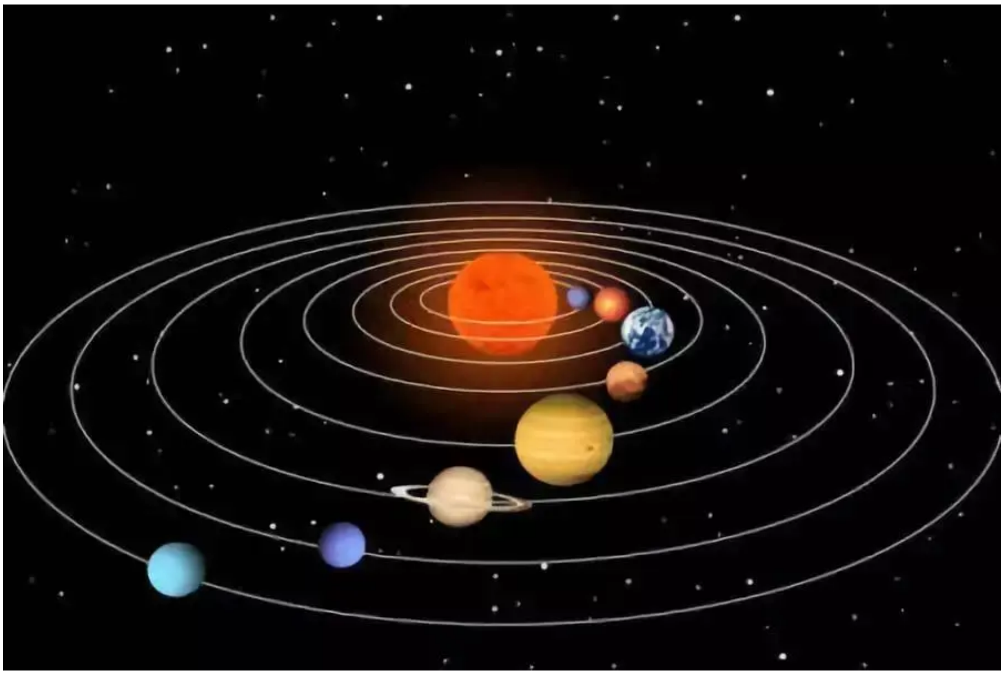
Airflow provides rich commands and WebUI for easy management and monitoring.

Users can define a set of dependent Directed Acyclic Graph (DAG) tasks to
se-quen-tially
be executed sequentially.

and then,

What is the Directed Acyclic Graph (DAG /ˈdæg/)?

In fact, I don't know how to introduce DAG, but I can show you a picture.



plan-ets
This is the solar system, many planets move around the sun, but this picture is a “Cyclic Graph”, which misleads many people.
In fact, the planets in the solar system operate according to the DAG (Directed Acyclic Graph) mode, like the next picture.

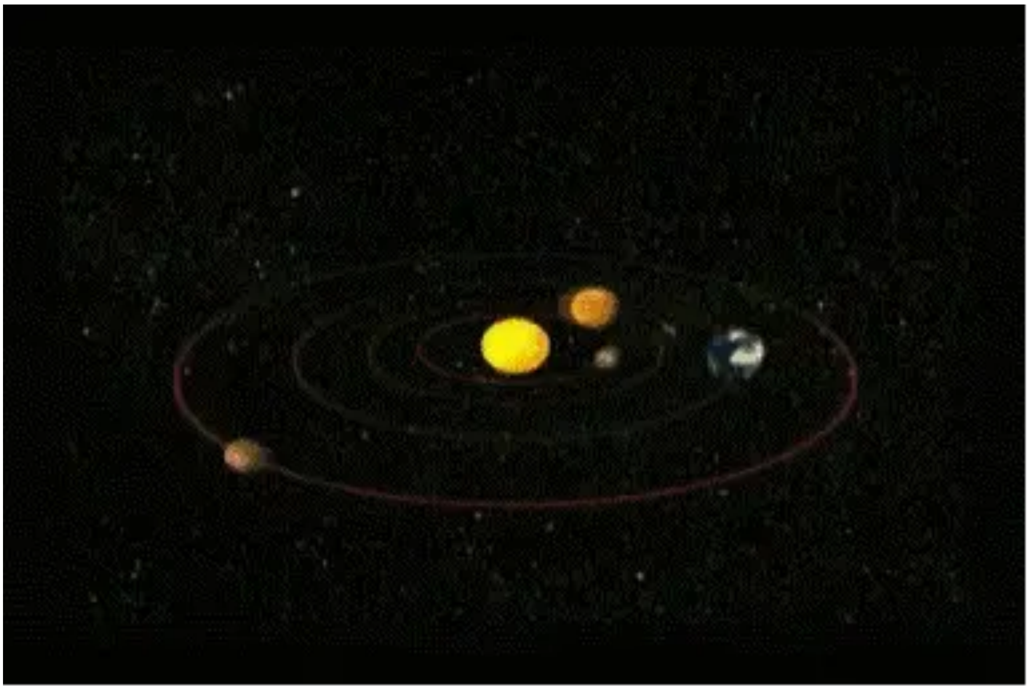


IMAGE: https://upload-images.jianshu.io/upload_images/7808357-2fd9f96954801b2b.gif

In simple terms, this is a DAG.

Back to Airflow topic,
Before I use it, the simplest workflow is actually crontab.

Compared with Airflow, crontab has many shortcomings,
For example,

de-pen-den-cies
1. Difficult to handle task dependencies.
in-con-ve-ni-ent
2. It is inconvenient to check the progress.
3. No automatic retry and alarm.
his-tor-i-cal
4. Without logs, it is difficult to view task execution time and historical records,
op-ti-mized
and cannot be optimized accordingly.

The core component of Airflow.

Scheduler

Scheduler is a process that uses DAG definition combined with character status in metadata to determine which tasks need to be executed and the priority of task execution. Schedulers typically run as services.

WebServer

A graphical interface is provided to monitor the running status of the DAG and to operate the DAG. The Webserver uses the Gunicorn framework of python.

Metadata Database

Metabase, default is SQLite, can support MySQL, PostgreSQL. Store all DAGs, task definitions, run history, users, permissions, etc.

Worker

Used to execute tasks received by Executor. These are the processes that actually execute the task logic, determined by the executor being used.

Next, let's explain in detail

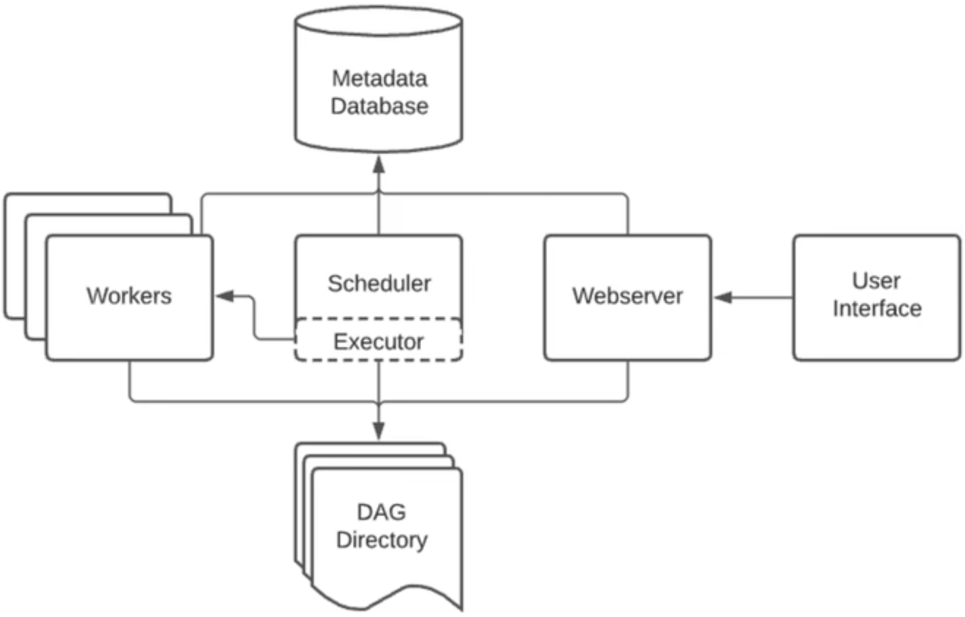
Scheduler

“

The Airflow scheduler monitors all tasks and DAGs, then triggers the task instances once their dependencies are complete. Behind the scenes, the scheduler spins up a subprocess, which monitors and stays in sync with all DAGs in the specified DAG directory. Once per minute, by default, the scheduler collects DAG parsing results and checks whether any active tasks can be triggered.

1. Check for any DAGs needing a new DagRun, and create them
2. Examine a batch of DagRuns for schedulable TaskInstances or complete DagRuns
3. Select schedulable TaskInstances, and whilst respecting Pool limits and other concurrency limits, enqueue them for execution

”



Executor

Airflow is a comprehensive platform that is compatible with various components, so there are many options to choose from when using it. For example, there are four options for the most critical executor:

SequentialExecutor: A single process executes tasks sequentially, the default executor, usually only used for testing.

LocalExecutor: Execute tasks locally in multiple processes.

CeleryExecutor: Distributed scheduling, commonly used in production environments.

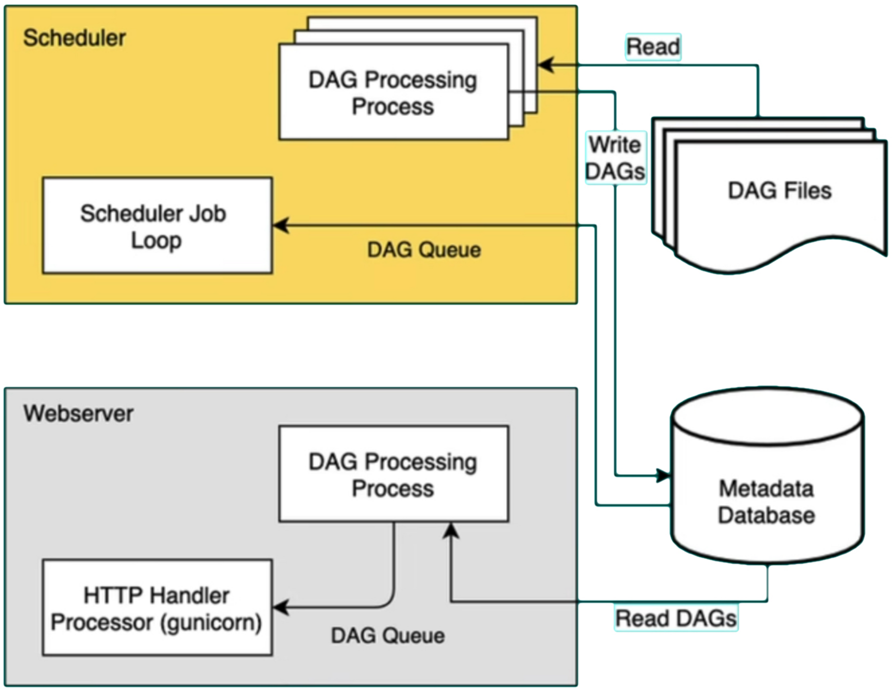
a-nal-y-sis

DaskExecutor: Dynamic task scheduling, mainly used for data analysis

and more…

DAG Serialization

With DAG Serialization we aim to decouple the Webserver from DAG parsing which would make the Webserver very light-weight.



Review the most basic concepts of Airflow

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.

DAG

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.

DAG RUN

A DAG Run is an object representing an instantiation of the DAG in time.

TASK

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