**Apache Airflow:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prerequisites:**

**Python:**

* Python software should be installed because airflow scripts will be written.
* Python knowledge is fine to manage airflow.

**Docker:**

* Docker is a containerization platform that allows you to package applications and their dependencies into lightweight, portable containers. These containers can then be run consistently across different environments.

**Airflow:**

* Designed for orchestrating complex workflows and data pipelines.
* It is an open source for programmatically authoring, scheduling and monitoring workflows.
* It allows users to define complex data pipelines as DAG(direct acyclic graph) using python code.

**Features :**

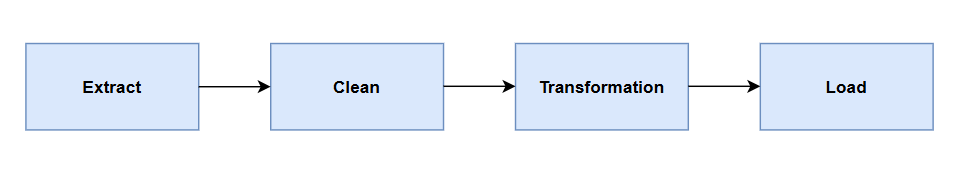
* **Workflow as Code**: We can write a code in python.
* **Extensibility**: Writing common operators, sensors, hooks and enabling integrations with
  + External Apis
  + Cloud services
  + Databases etc…
* **Scalability**: Scales effortlessly from small deployments to massive workloads in a cloud environment and supports multiple executors.
* **User friendly interface**: Web base UI provides Clear visualization of DAGs. task status & logs.
  + Monitoring
  + Troubleshooting and
  + Managing workflows.

**Use Cases:**

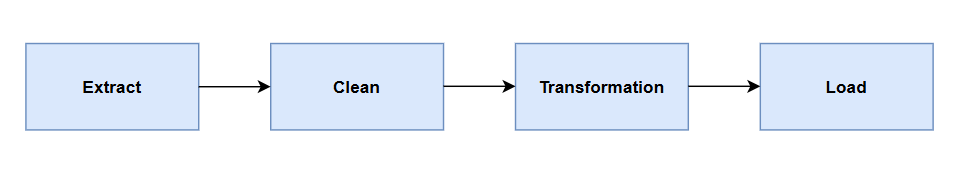
* Orchestrating batch ETL jobs
* Automating data pipeline execution and monitoring.
* Machine learning model training.
* Generating automated reports
* Managing Devops tasks

**Why a Data Orchestrator:**

* Data orchestration is the process of coordinating & automating the movement, transformation & integration for data across various systems & processes to ensure efficient and reliable data workflows.
* Eg:



**Traditional ETL Data Pipeline:**

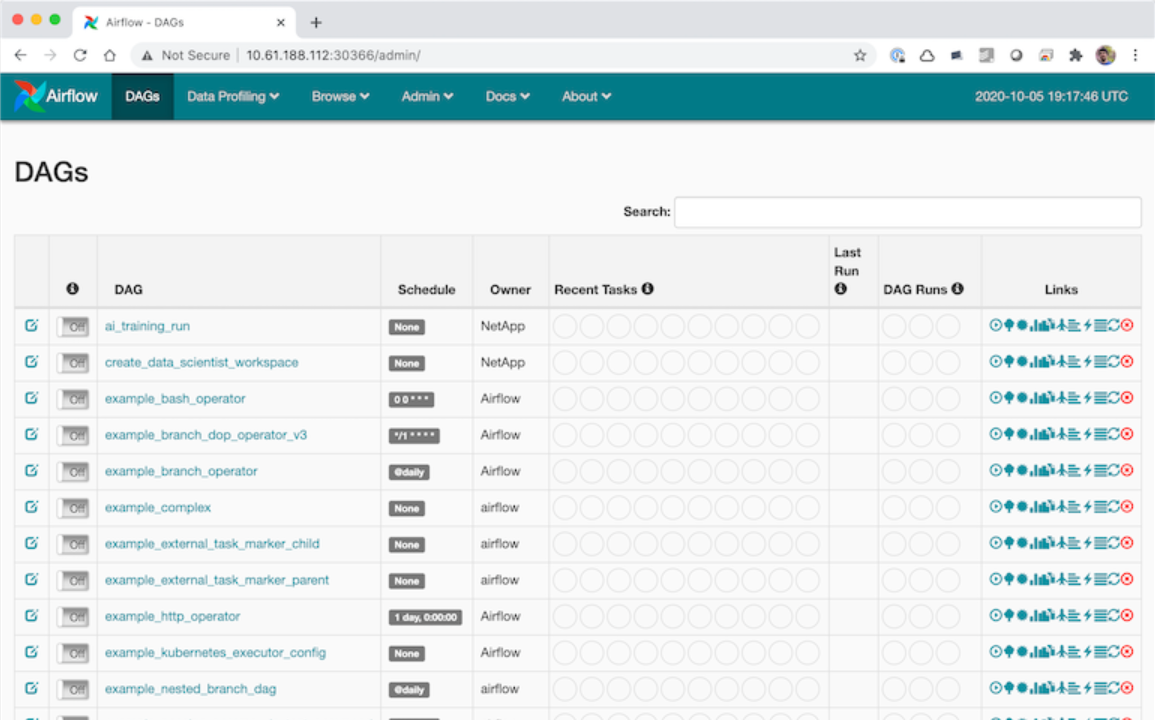


**Problems:**

* If step 2 or 3 failed, We should start from first. It is not the ideal way.
* How to schedule for ETL batch jobs.
* If something went wrong, how to retry multiple times.
* How to notify, Whether it is failed/ success
* How to monitor the data pipelines.
* How can we logging and backfilling

**Solution: Airflow can solve all above problems.**

**Airflow UI:**

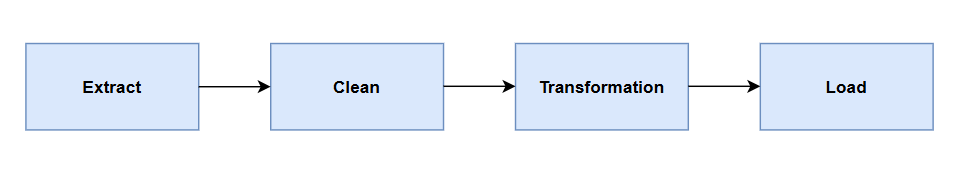
****

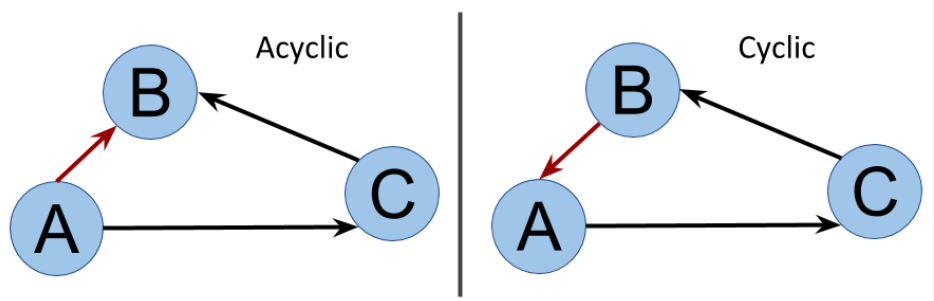
**Where can we use it :**

* Batch ETL pipelines - Automate
* Machine learning train/test pipelines - Automate
* Not suitable for real time data

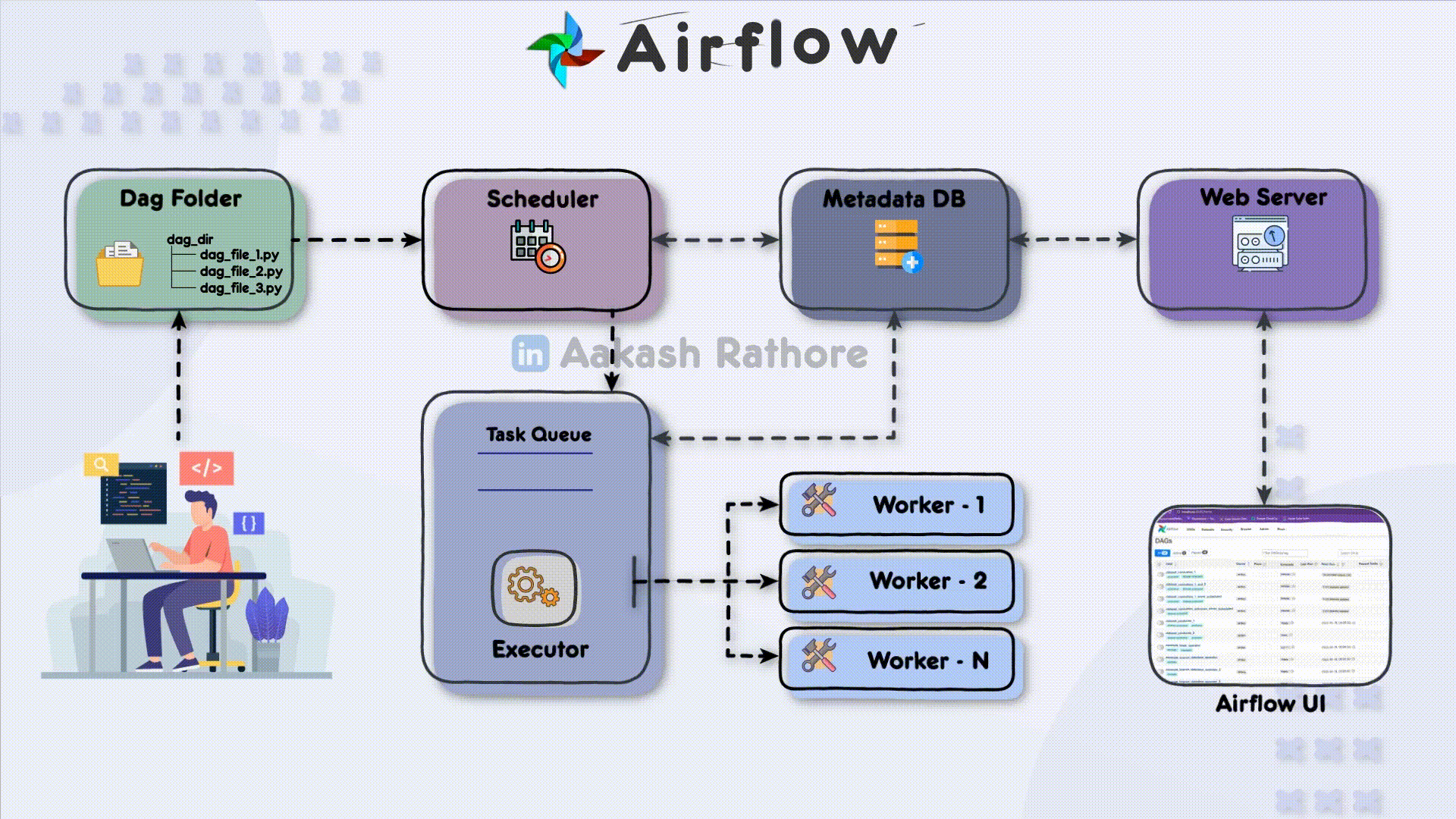
**DAG (Direct Acyclic Graph):**

* It is used to create workflows.
* It is a type of graph that contains nodes(vertices) and edges.





**Airflow Components:**

****

* **DAG Folder:** DAG folder contains all dag python files
* **Scheduler:**
  + Responsible for scanning all dag files in DAG Folder and It will parse these files to synchronize with the metadata database ensuring all DAGs and tasks are up to date.
  + Handles the scheduling intervals

**schedule\_interval=**@daily

@monthly

@hourly

@weekly

@monthly

@quarterly

@yearly

‘\*/15 \* \* \* \*’ → Every 15 minutes (cron expression)

* **Metadata DB:**
  + It is used to store the DAG information like structure, schedule and default arguments
  + Task Information
  + Default stores in sqlite
    - We can’t do multiple writes or reads
    - We can’t run multiple workflows at a time.
  + Recommended to choose PostgreSQL / MySql
* **WebServer:**
  + It is a key component of the apache workflow that hosts the web based UI.
  + It will allow users to interact with their workflows.
* **Executor:**
  + It is a key component of the apache workflow
    - **localExecutor** :
      * SequentialExecutor:
      * LocalExecutor
    - **remoteExecutor**:
      * CeleryExecutor
      * KubernetesExecutor

**Airflow Installation?**

Airflow Download: <https://github.com/puckel/docker-airflow>

Docker Download: <https://docs.docker.com/desktop/setup/install/windows-install/>

