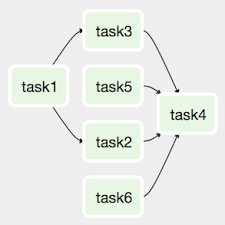
### What is airflow?

Open souce platform to author, schedule and monitor workflows.

#### Core components

* Web Server - Flask server with Gunicorn serving the UI
* Scheduler - Daemon in charge of scheduling workflows
* Metastore - Database where metadata will be stored
* Executor - Class defining how the task should be executed
* Worker - Process/Subprocess

#### DAG?



Operator

Ex:-

db = connect(host,crendentials)

db.insert(sql\_request)

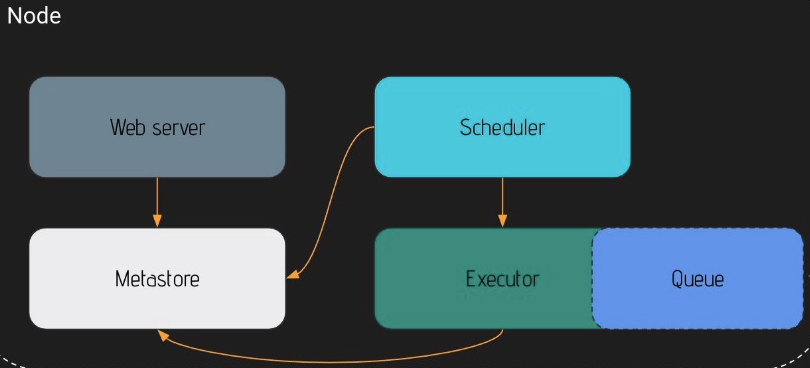
* Action Operators- CRUD
* Transfer Operators - send data from source to destinations
* Sensor Operator - Waits for a condition to be met before getting triggered

#### Task Instance ?

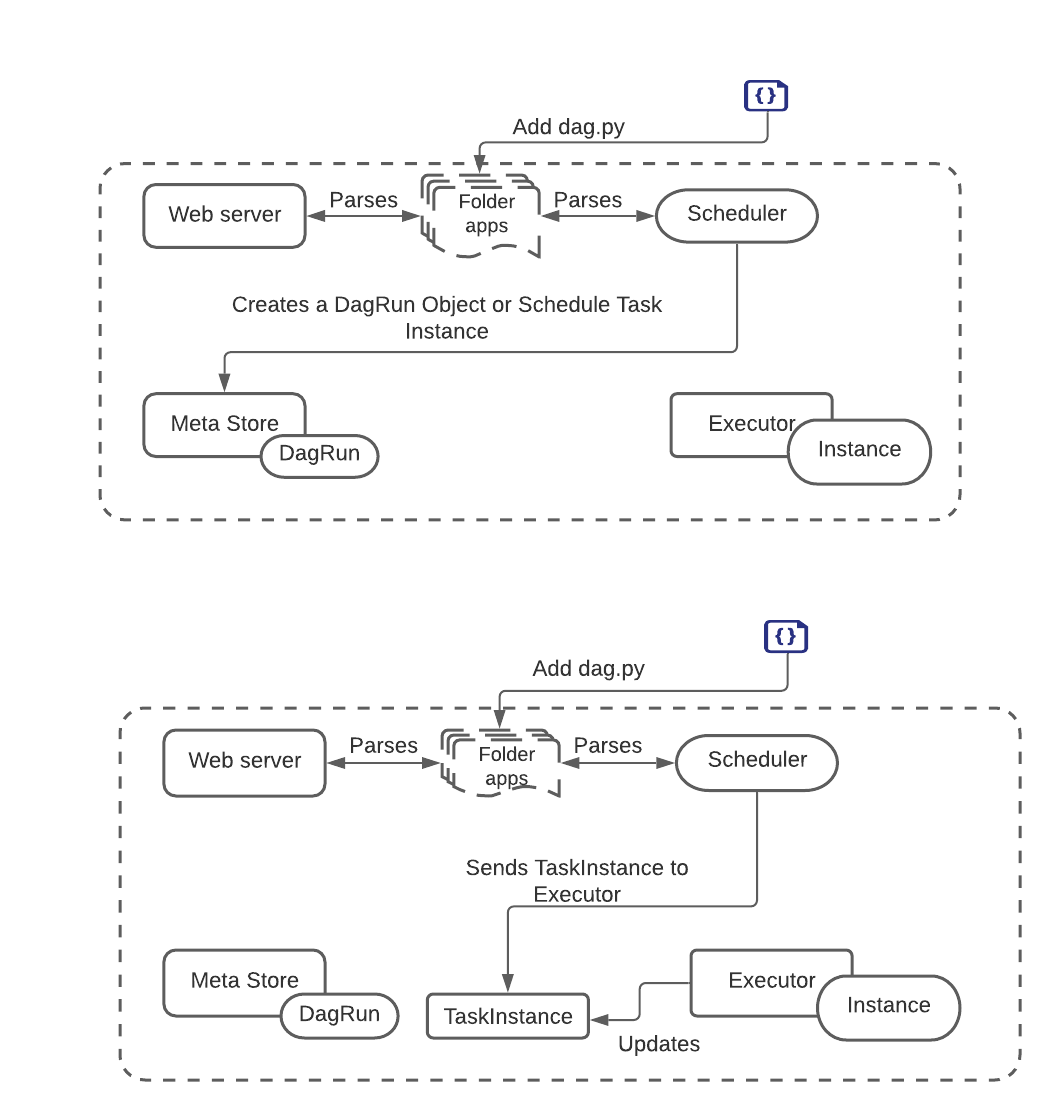
**When an operator runs inside the DAG, it is called a task instance.**

**How airflow works ?**

**One node architecture**

****

**Workflow -**

****

**Installing Docker**

| sudo apt update  sudo apt install apt-transport-https ca-certificates curl software-properties-common  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"  sudo apt update  apt-cache policy docker-ce  sudo apt install docker-ce  sudo systemctl status docker  #Execute docker without sudo  sudo usermod -aG docker ${USER}  su - ${USER}  id -nG |
| --- |

**Installing Airflow**

| ﻿CLI Commands  mkdir airflow\_implementation  cd airflow-implementation  docker run -it --rm -p 8080:8080 python:3.8-slim /bin/bash  export AIRFLOW\_HOME=/usr/local/airflow  env | grep airflow  apt-get update -y && apt-get install -y wget libczmq-dev curl libssl-dev git inetutils-telnet bind9utils freetds-dev libkrb5-dev libsasl2-dev libffi-dev libpq-dev freetds-bin build-essential default-libmysqlclient-dev apt-utils rsync zip unzip gcc && apt-get clean  \* Install all tools and dependencies that can be required by Airflow  useradd -ms /bin/bash -d ${AIRFLOW\_HOME} airflow  \* Create the user airflow, set its home directory to the value of AIRFLOW\_HOME and log into it  cat /etc/passwd | grep airflow  pip install --upgrade pip  su - airflow  \* Log into airflow  python -m venv .sandbox  source .sandbox/bin/activate  wget https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/constraints-3.8.txt  pip install "apache-airflow[crypto,celery,postgres,cncf.kubernetes,docker]"==2.0.2 --constraint ./constraints-3.8.txt  \* Install the version 2.0.2 of apache-airflow with all subpackages defined between square brackets.  airflow db init  \* Initialise the metadatabase  airflow scheduler &  \* Start Airflow’s scheduler in background  airflow webserver &  \* Start Airflow’s webserver in background  docker build -t airflow-basic .  \* Build a docker image from the Dockerfile in the current directory (airflow-materials/airflow-basic) and name it airflow-basic  docker run --rm -d -p 8080:8080 airflow-basic |
| --- |

| Quick Tour of Airflow CLI  docker ps  \* Show running docker containers  docker exec -it container\_id /bin/bash  \* Execute the command /bin/bash in the container\_id to get a shell session  pwd  \* Print the current path where you are  airflow db init  \* Initialise the metadatabase  airflow db reset  \* Reinitialize the metadatabase (Drop everything)  airflow db upgrade  \* Upgrade the metadatabase (Latest schemas, values, ...)  airflow webserver  \* Start Airflow’s webserver  airflow scheduler  \* Start Airflow’s scheduler  airflow celery worker  \* Start a Celery worker (Useful in distributed mode to spread tasks among nodes - machines)  airflow dags list  \* Give the list of known dags (either those in the examples folder or in dags folder)  ls  \* Display the files/folders of the current directory  airflow dags trigger example\_python\_operator  \* Trigger the dag example\_python\_operator with the current date as execution date  airflow dags trigger example\_python\_operator -e 2021-01-01  \* Trigger the dag example\_python\_operator with a date in the past as execution date (This won’t trigger the tasks of that dag unless you set the option catchup=True in the DAG definition)  airflow dags trigger example\_python\_operator -e '2021-01-01 19:04:00+00:00'  \* Trigger the dag example\_python\_operator with a date in the future (change the date here with one having +2 minutes later than the current date displayed in the Airflow UI). The dag will be scheduled at that date.  airflow dags list-runs -d example\_python\_operator  \* Display the history of example\_python\_operator’s dag runs  airflow tasks list example\_python\_operator  \* List the tasks contained into the example\_python\_operator dag  airflow tasks test example\_python\_operator print\_the\_context 2021-01-01  \* Allow to test a task (print\_the\_context) from a given dag (example\_python\_operator here) without taking care of dependencies and past runs. Useful for debugging. |
| --- |

**Docuementation in progress……………………..**