CS 777: Big Data Analytics

Term Paper

APACHE AIRFLOW

Vaidehi Shah

Boston University, Metropolitan College

Spring 2023

INTRODUCTION

What is Apache Airflow?

Apache Airflow [1] is a powerful and versatile open-source platform It is used for the efficient management and execution of data workflows, task scheduling, and orchestration of complex processes. At its core, Apache Airflow uses DAGs (Directed Acyclic Graphs).

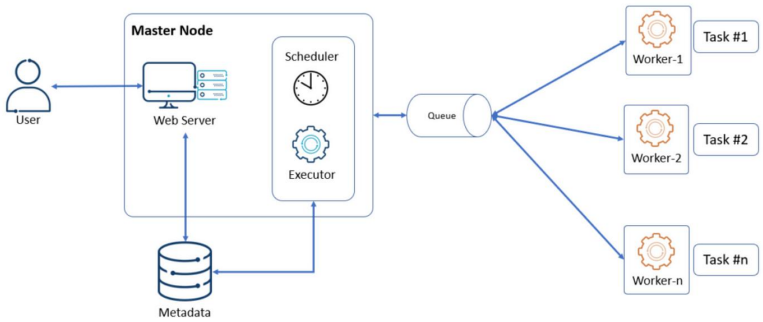
Why Apache Airflow?

With the exponential growth of data processes, it addresses the need for efficient orchestration and automation of complex workflows and task scheduling. Airflow simplifies management by representing workflows as Directed Acyclic Graphs (DAGs), enabling automation and clear visualization of task dependencies. Some of its key features include:

- Flexibility : agnostic to the programming language or tools used in tasks
- Scalability : it has a distributed architecture
- Monitoring : provides a web-based UI that offers insights into the status of workflows, logs, and task histories; can also be integrated with external monitoring and alert systems

ARCHITECTURE

The architecture is based on a distributed and modular framework, which includes several core components:



1. **Metastore Database**: The Metastore Database, often powered by databases like PostgreSQL or MySQL, stores metadata related to tasks, task status, and workflow configurations. It keeps track of task history and maintains the state of the entire system.

2. **Scheduler**: The Scheduler is the brain of Apache Airflow. It constantly monitors all tasks and their dependencies and schedules when and how they should run. It ensures that tasks are executed in the correct order and takes into account task priorities, data dependencies, and resource availability.

3. **Executor**: Executors are responsible for executing tasks on worker nodes. Airflow supports various executors, such as the LocalExecutor, which runs tasks locally on the same machine as the Scheduler, and the CeleryExecutor, which uses a distributed task queue system like Apache Celery for task execution.

4. **Workers**: Workers are the compute resources that execute the actual tasks. They pull tasks from the message queue and execute them. Workers can be horizontally scaled to accommodate different workloads and ensure timely task execution.

5. **Web Interface**: The web-based UI provided by Airflow allows users to interact with the system, monitor task progress, and view historical data. It also provides a means to trigger, pause, and rerun tasks, making it a valuable tool for both developers and non-technical users.

6. **Message Queue**: Apache Airflow uses a message queue, such as Apache Kafka or RabbitMQ, to deliver tasks to workers. This decouples the scheduling of tasks from their execution, improving system reliability and enabling distributed task processing.

7. **DAGs (Directed Acyclic Graphs)**: Workflows in Airflow are defined as DAGs, which are collections of tasks and their dependencies. These DAGs are written in Python code, providing a clear and programmable representation of workflow logic.

PROS & CONS OF AIRFLOW

Pros of Apache Airflow:

- Flexibility and Extensibility: Apache Airflow is highly flexible and extensible, allowing users to integrate with a wide range of technologies and customize workflows to their needs.
- Dynamic Workflow Scheduling: It optimizes task execution order based on dependencies and resource availability, reducing manual management and improving efficiency.
- Monitoring and Alerting: Provides a user-friendly interface and integrates with external tools for real-time monitoring, enhancing operational reliability.

Cons of Apache Airflow:

- Learning Curve: Airflow has a steep learning curve, particularly for newcomers to workflow orchestration.
- Resource Consumption: It can consume significant computational resources, necessitating careful resource management.
- Complex Deployments: Setting up high-availability or clustered configurations can be challenging and require expertise in distributed systems.

USE CASES

1. **ETL Processes:**
   Apache Airflow is widely used for automating Extract, Transform, Load (ETL) processes, ensuring that data is efficiently extracted from source systems, transformed, and loaded into data warehouses or other destinations.

2. **Data Pipeline Automation:**
   Organizations leverage Airflow to build and automate data pipelines that connect various data systems, enabling data ingestion, transformation, and delivery for analytics and reporting.

3. **Machine Learning Workflows:**
   Airflow plays a crucial role in orchestrating machine learning workflows, streamlining data preparation, model training, hyperparameter tuning, and model deployment for data scientists and AI practitioners.

DEPLOYMENT [2]

Airflow installation using Docker (Docker provides isolated environment for running Airflow and its dependencies)

STEP 1

The first step is to install Docker, if you don't already have it on your system. You can go to the weblink [3] and install Docker for your specific system by following the instructions available there.

If you wish to check the successful installation of Docker and Docker-compose (will be done along with Docker) on your system, you can use the following commands:

>> *docker --version*

>> *docker-compose --version*

STEP 2

After this, it is recommended that you make a new folder in your system for airflow-docker, wherein you can download the docker-compose file describing all the services needed by airflow.

>> *mkdir airflow-docker*

>> *cd airflow-docker*

STEP 3

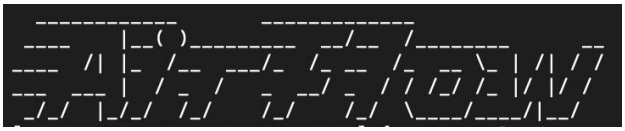
Here, you need to upload the docker-compose file, which has been made for us by the Airflow community. You can access the .yaml file [4]. To deploy Airflow on Docker Compose, you should fetch docker-compose.yaml:

>> *curl -LfO 'https://airflow.apache.org/docs/apache-airflow/2.7.2/docker-compose.yaml'*

STEP 4

Run the following command to start and run a set of Docker containers defined in a docker-compose.yml file in Docker Compose:

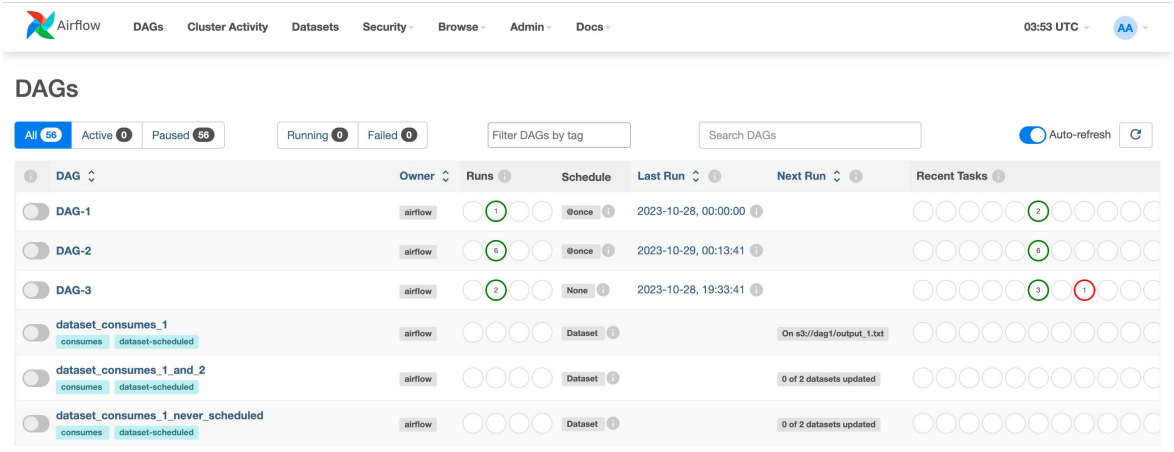
>> *docker-compose up*



STEP 5

You are now ready to access Airflow on one of your browsers: *Listening at: http://localhost:8080/login/*

The default username and password, both are set as "airflow".



WORKING WITH AIRFLOW

This is how I created my first DAG:

dag_1.py

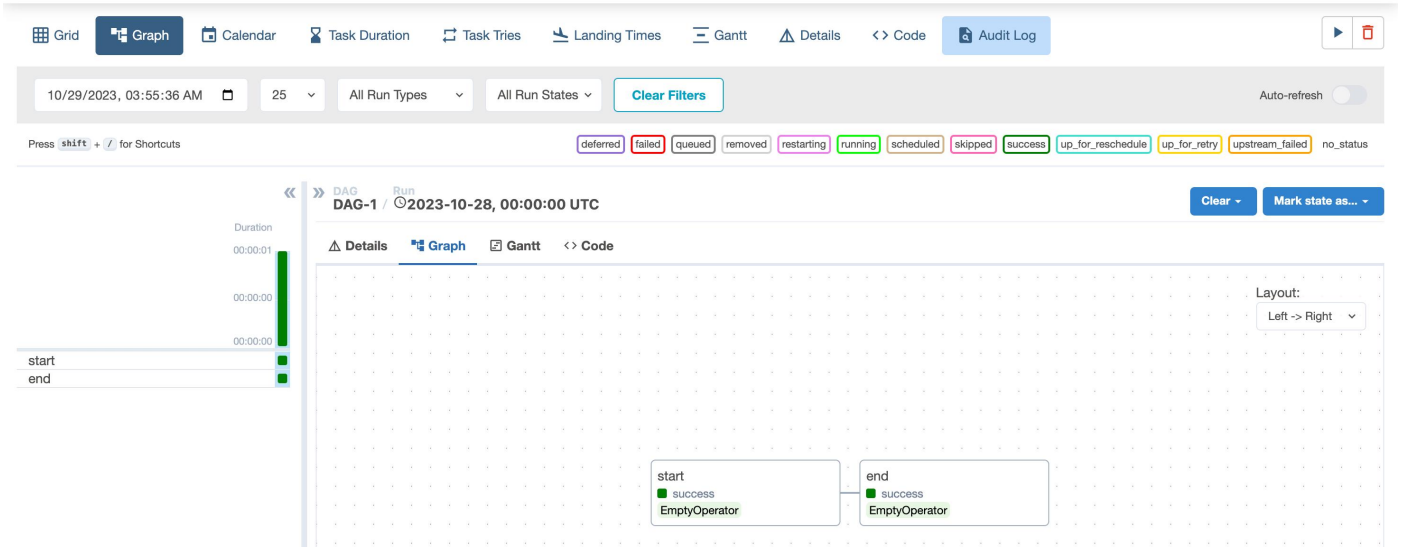
```python
# STEP: 1
### importing modules
from airflow import DAG
from datetime import datetime, timedelta
from airflow.operators.dummy import DummyOperator

# STEP: 2
### default args
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 10, 28),
    'retries': 0
}

# STEP: 3
### instantiate DAG
dag = DAG(dag_id='DAG-1', default_args=default_args, catchup=False, schedule_interval='@once')

# STEP: 4
### define tasks
start = DummyOperator(task_id='start', dag=dag)
end = DummyOperator(task_id='end', dag=dag)

# STEP: 5
### define dependencies
start >> end
```



Demo code DAG-2:

dag_2.py

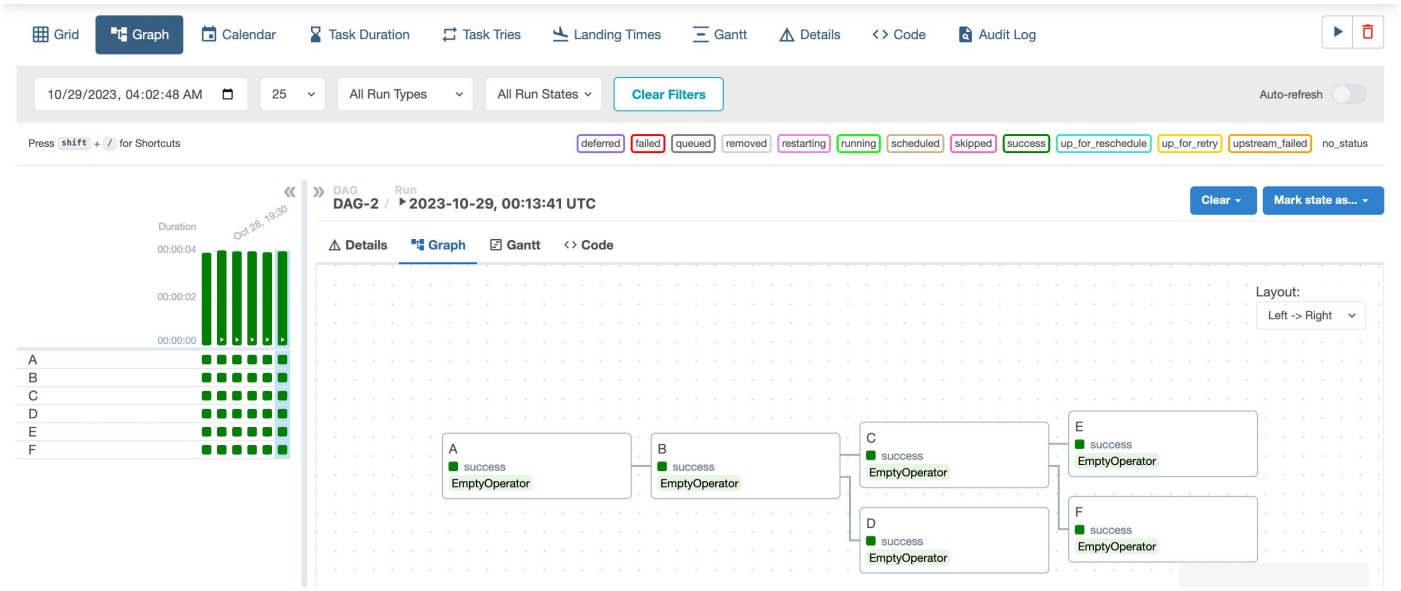
```python
# STEP: 1
### importing modules
from airflow import DAG
from datetime import datetime, timedelta
from airflow.operators.dummy import DummyOperator

# STEP: 2
### default args
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 10, 28),
    'retries': 0
}

# STEP: 3
### instantiate DAG
dag = DAG(dag_id='DAG-2', default_args=default_args, catchup=False, schedule_interval='@once')

# STEP: 4
### define tasks
A = DummyOperator(task_id='A', dag=dag)
B = DummyOperator(task_id='B', dag=dag)
C = DummyOperator(task_id='C', dag=dag)
D = DummyOperator(task_id='D', dag=dag)
E = DummyOperator(task_id='E', dag=dag)
F = DummyOperator(task_id='F', dag=dag)

# STEP: 5
### define dependencies
A >> B
B >> C
B >> D
C >> [E,F]
```



REFERENCES

[1] https://airflow.apache.org/docs/apache-airflow/stable/index.html

[2] https://github.com/vaidehi-nshah/airflow_demo

[3] https://docs.docker.com/desktop/

[4] https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html

[5] https://youtu.be/K9AnJ9_ZAXE?feature=shared

[6] https://youtu.be/aTaytcxy2Ck?feature=shared