**Amazon MWAA & a quick overview of orchestrating tools in AWS environment**

Amazon MWAA is a managed service for open-source Apache-Airflow. Airflow is one of the most widely used orchestrating tool, not only viable for data-engineering orchestration use-cases, but also for data-science orchestration use-cases.

In AWS ecosystem apart from MWAA, we could mainly use Glue-workflow, step-functions, eventbridge etc. for orchestration.

Glue-workflow has limitation that it can only trigger Glue objects viz. glue-jobs, trigger & crawlers. In case the concerned use-case contains only glue-objects then glue-workflow is the ideal choice for orchestration; in fact, it is the most economical & easy-to-use among all the orchestration tools mention in above paragraph. But the downside of it is, very tedious retry mechanism, confusing last run status (whether pipeline fails or succeed, last-run status will always show "completed" 😀 , almost no versioning & above all it is platform-gnostic.

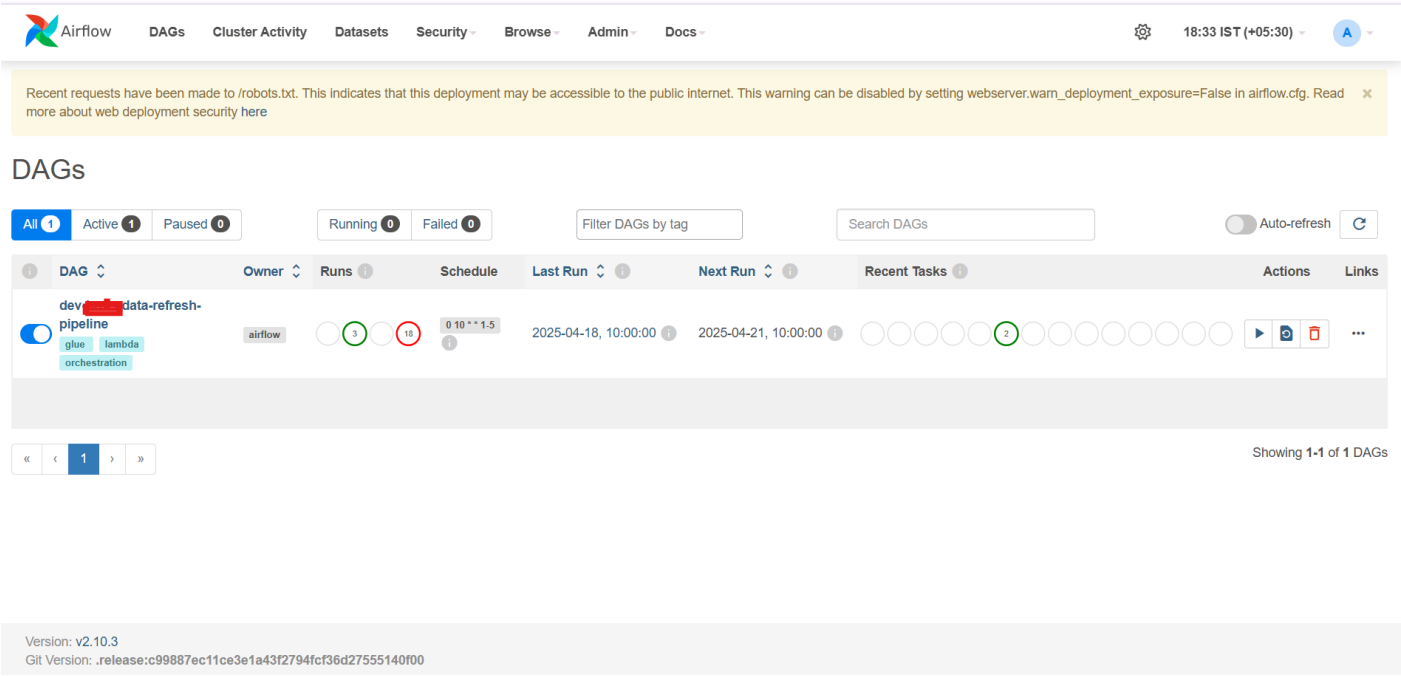
Step function is rarely used orchestration tool when we talk about enterprise level cloud-implementation. Though state-machines could encompass most of the AWS-service but generally step-function use cases are limited to orchestrating only lambda-functions or any AWS-infrastructure services (e.g. compute services/networking & content delivery services etc.). One place where it can be highly usable is where your orchestration requires human-intervention in-between e.g. while you're committing an object to a repo, you might require a approval from repository owner. The downsize of this service is it's a bit costly (but lesser than MWAA), slightly redundant, highly platform-gnostic & above all not an industry standard service for orchestration use-case.

MWAA is a versatile managed Apache-Airflow platform that can orchestrate any AWS service by wrapping them inside supported operators. The functionality that sets apart MWAA from other two orchestration tools discussed above is that it's a python codified generated pipeline. We could design retry mechanism & alerting mechanism as per our convenience & that too is highly customizable. in case your ETL pipeline involves multiple AWS services viz. glue-objects(glue-jobs/crawler/trigger), lambda-functions, The only downside that I could see in MWAA is it's high infra. cost due to its ever-running environment, that despite selecting even the least capacity i.e. class mw1.small with 2 scheduler count/2 maximum as well as minimum webservers & 10 maximum worker count. As of documenting this article, the above-mentioned configuration was incurring approximately $425 per month (~36,000 INR considering $1= 85 INR).

**MWAA setup**:

In AWS for enabling airflow, we need to first create an environment, which consists of different class like small, medium, large, xl-large etc. mainly varying on how much DAGs (**D**irected **A**cyclic **G**raphs) they supports & configuration of their components like webservers RAM/CPU, workers & schedulers RAM/CPU & Database RAM/CPU capacity.

Here, Webserver refers to the Airflow-UI which looks something like this 👇




Workers & schedulers refers to master & worker nodes in terms of virtual-machines, which're responsible for handling the actual workload of orchestrating the data-pipeline objects.

Database refers to S3 storage capacity that is utilized to store these. This is different from the S3 storage location where source-code or DAG-python code is stored.

**Caveat:**

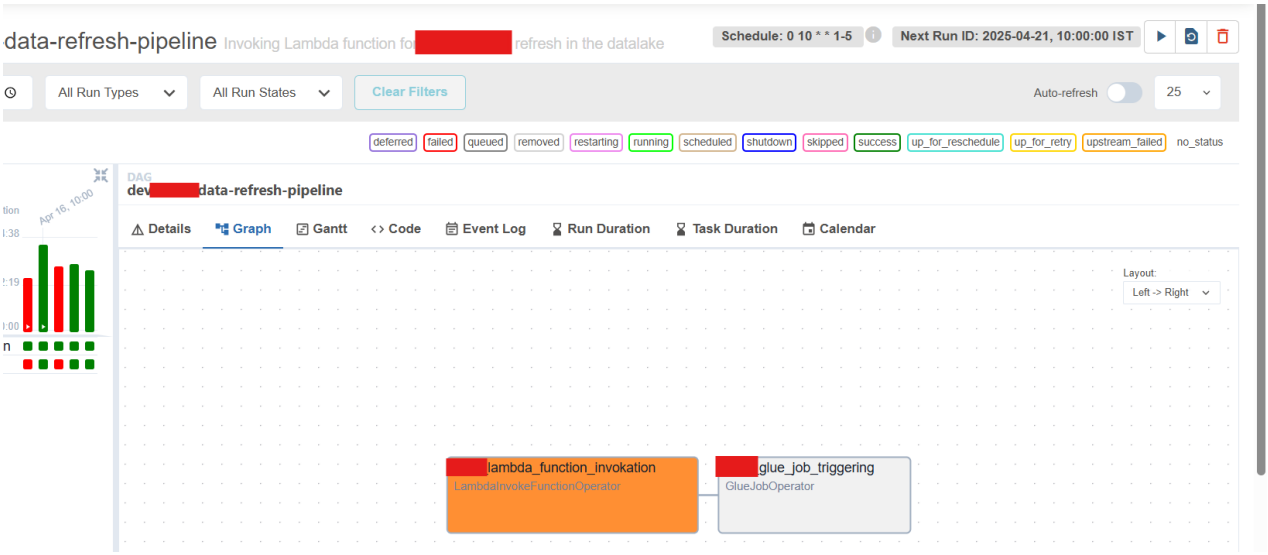
1. ***Ensure versioning is enabled in S3 storage location of DAG-source code.***
2. ***Ensure the VPC that hosts this MWAA server have IGW (internet gateway) enabled so that it could access Airflow UI.***

Once the environment is setup & every requisite infrastructure requirement is satisfied, the MWAA environment is created & the landing page of MWAA looks like this 👇.


To access the Airflow just click on "Open Airflow UI" option & the webserver UI opens as shown above in screenshot 1.

Use case: We have taken a very small usecase of triggering a glue job on successful completion of a lambda function.

For this we have written a python script dag.py file. This script is uploaded into the S3 folder that we've designated while creating the MWAA environment. The airflow UI keeps self-refreshing & if DAG python script is syntactically as well as semantically correct it'll create the DAG automatically in the airflow UI.




This use-case Job is scheduled as cron job to run every weekday at 10 AM IST.