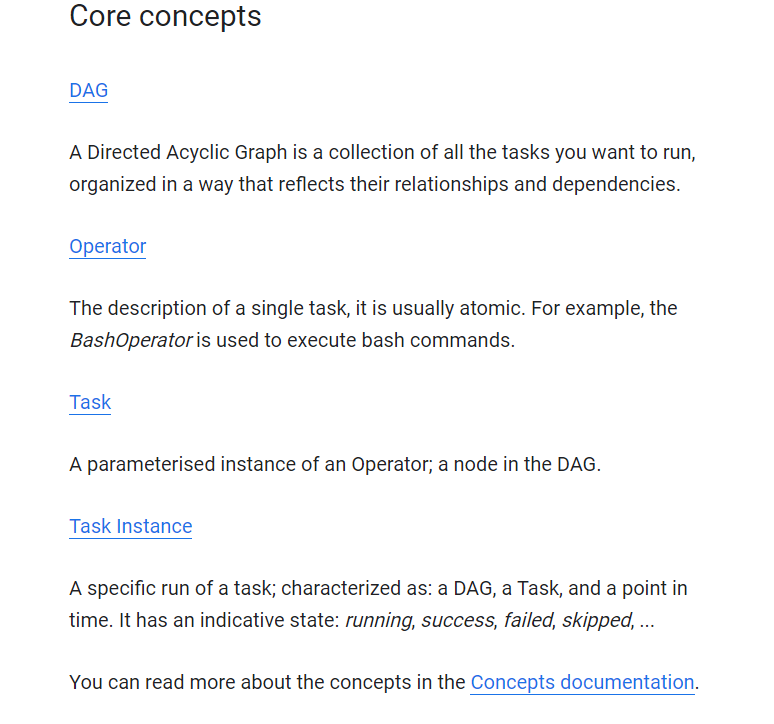


\* **Apache Airflow runs on cloud composer**. It is an **orchestration engine**.

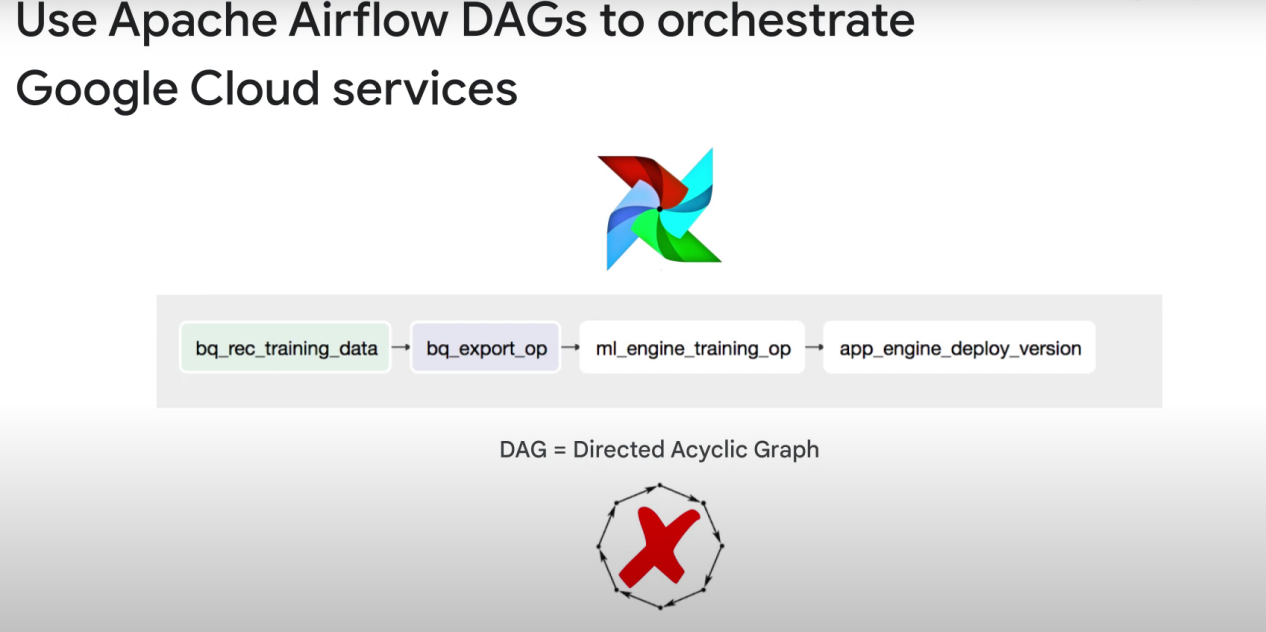
The heart of any workflow is a DAG. We can create DAGs using Cloud Fusion, and then orchestrate them using Cloud Composer.

[Airflow](https://airflow.apache.org/" \t "https://www.cloudskillsboost.google/course_sessions/1646063/labs/_blank) is a platform to programmatically author, schedule and monitor workflows.

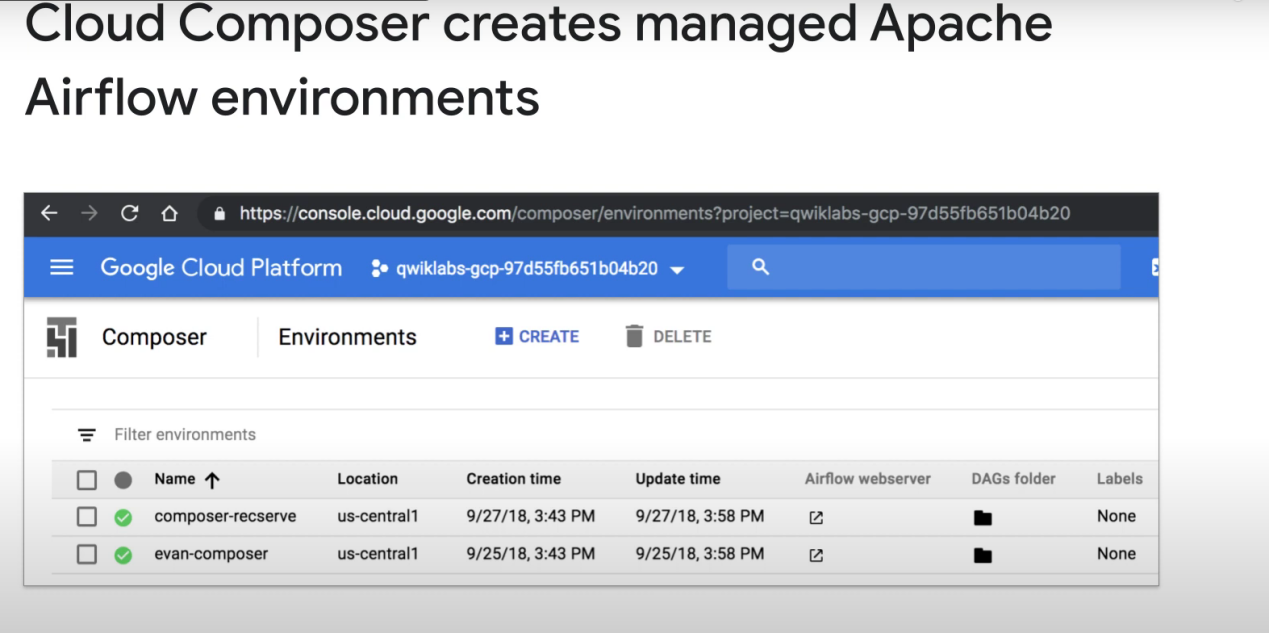
Use Airflow to author workflows as directed acyclic graphs (DAGs) of tasks. The airflow scheduler executes your tasks on an array of workers while following the specified dependencies.



«**Cloud Composer is used for orchestration of Data Fusion pipelines and any other custom tasks performed outside of Data Fusion»**

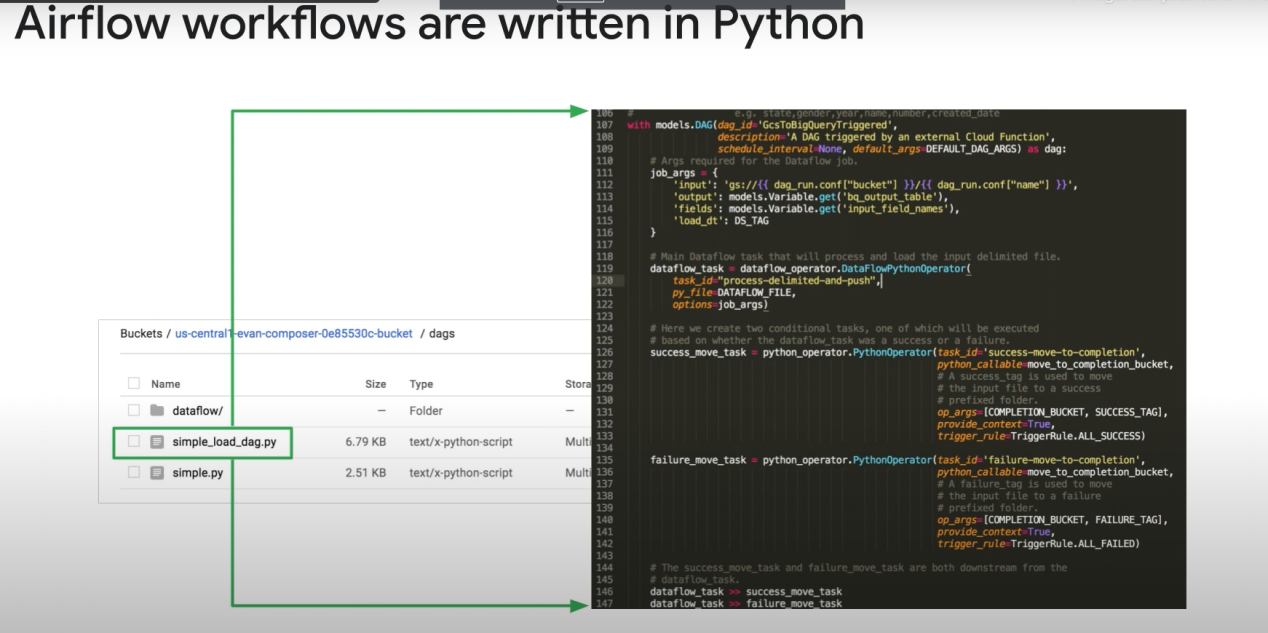


(you can use pretty much any google service you want in that DAG)



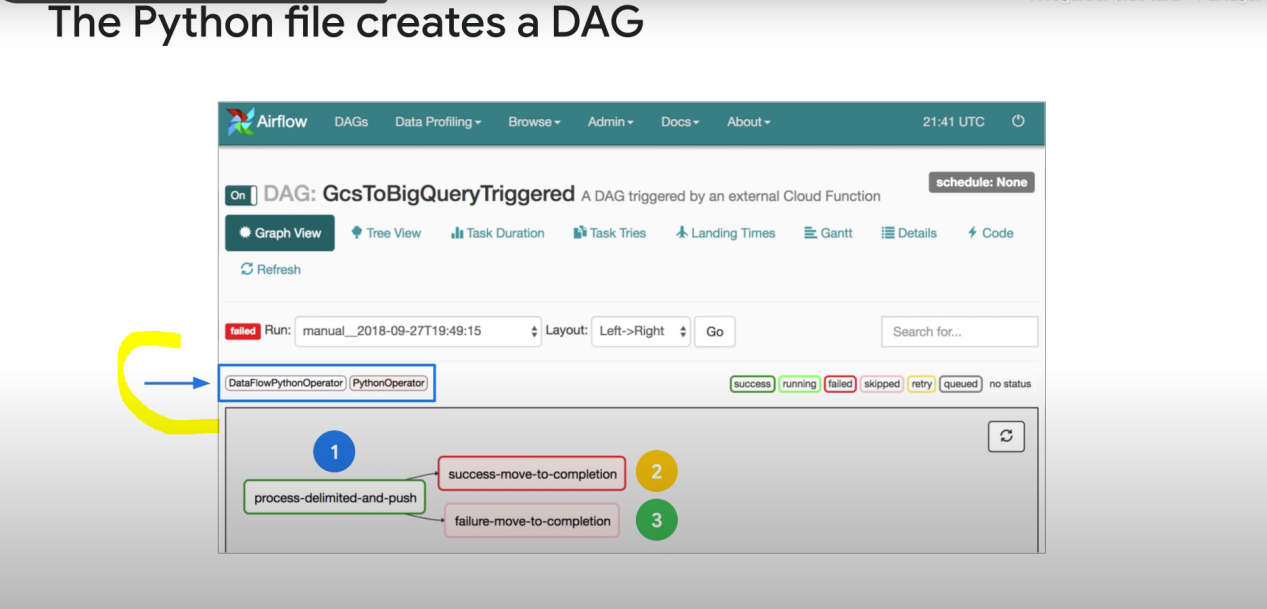
You can have many cloud composer envs., with each env. Having its own apache airflow env. With 0 or many dags.

To edit environement variables, you should usually do that on the airflow env. And not on cloud composer.



**1 python file per DAG**

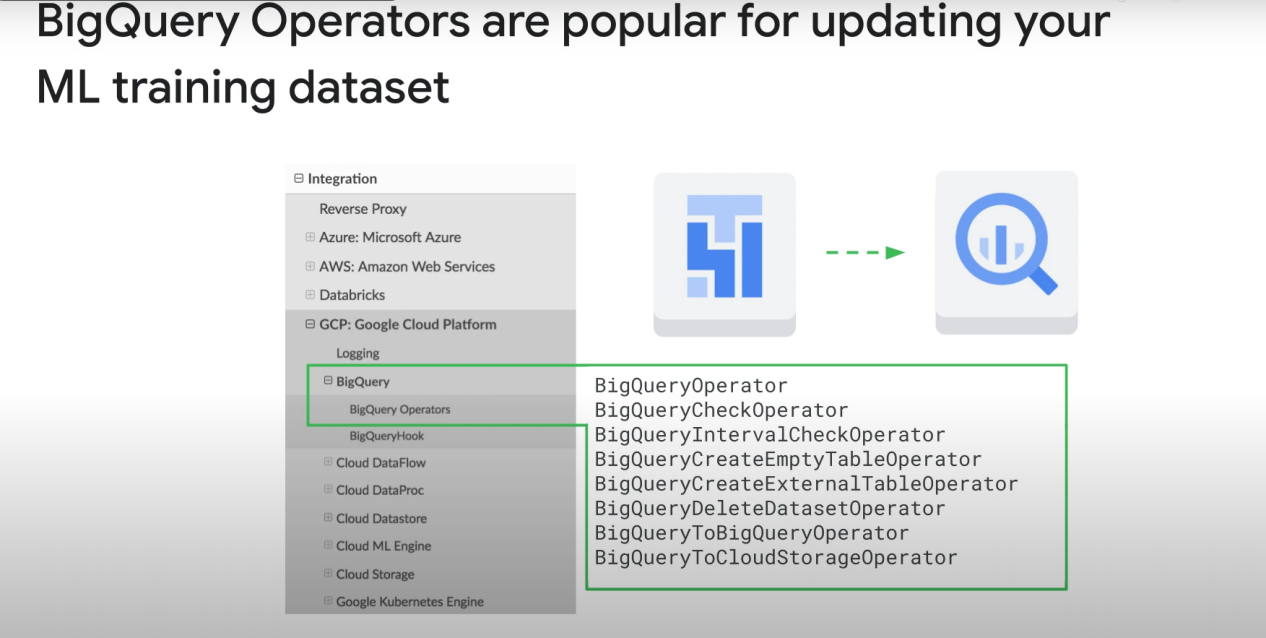
Sequential DAG == pas de choix entre 2 nodes, toutes les nodes sont executées.



A DAG is a list of steps. The operators are the actual content of the steps, i.e the process/traitement that will happen

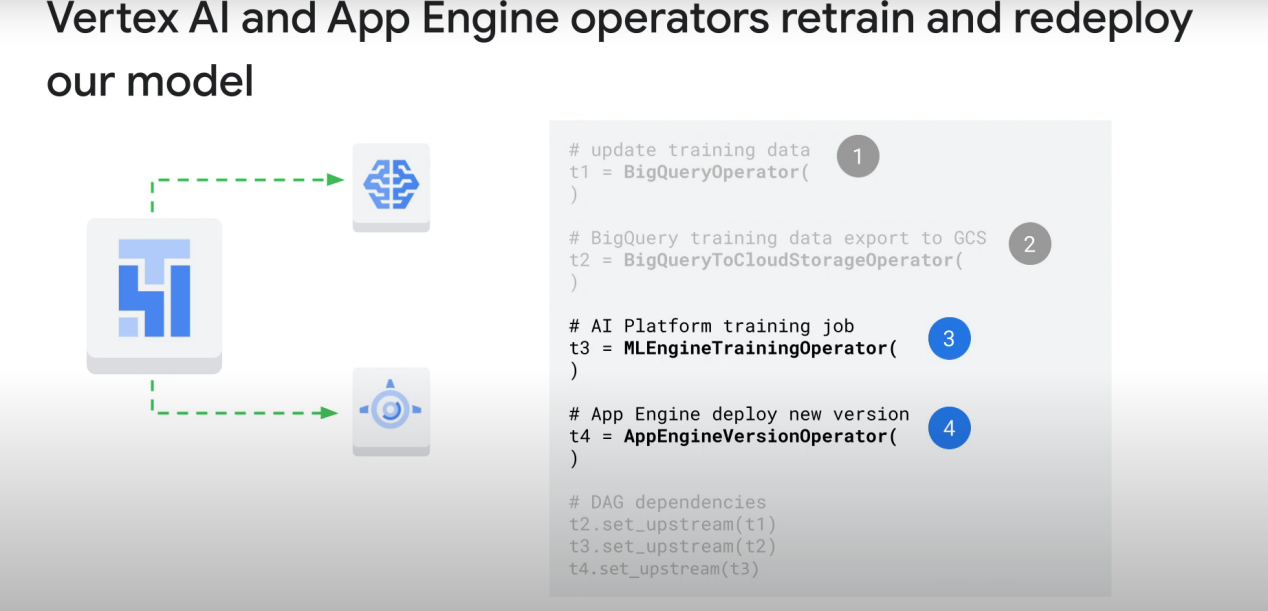
Generally, operators are atomic: 1 operator per task

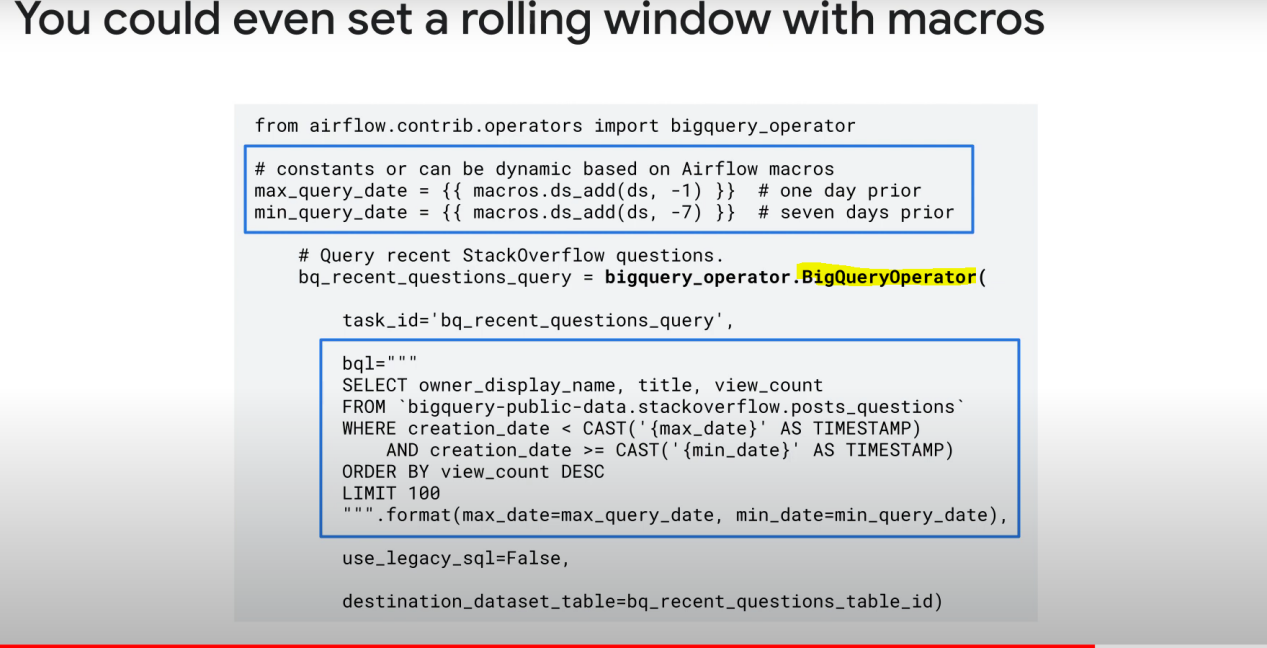
Example of operators:



NOTE: WE CAN USE SOME OPERATORS TO RUN CLOUD DATA FUSION PIPELINES!!! THUS THE IMPORTANCE OF DATA COMPOSER!!!

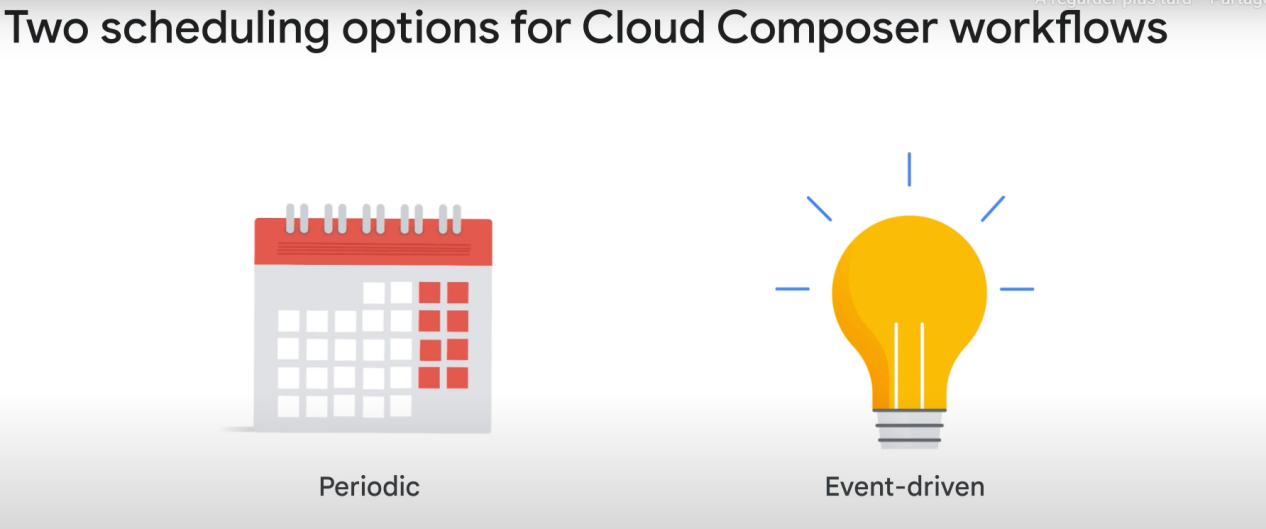
Example in code:



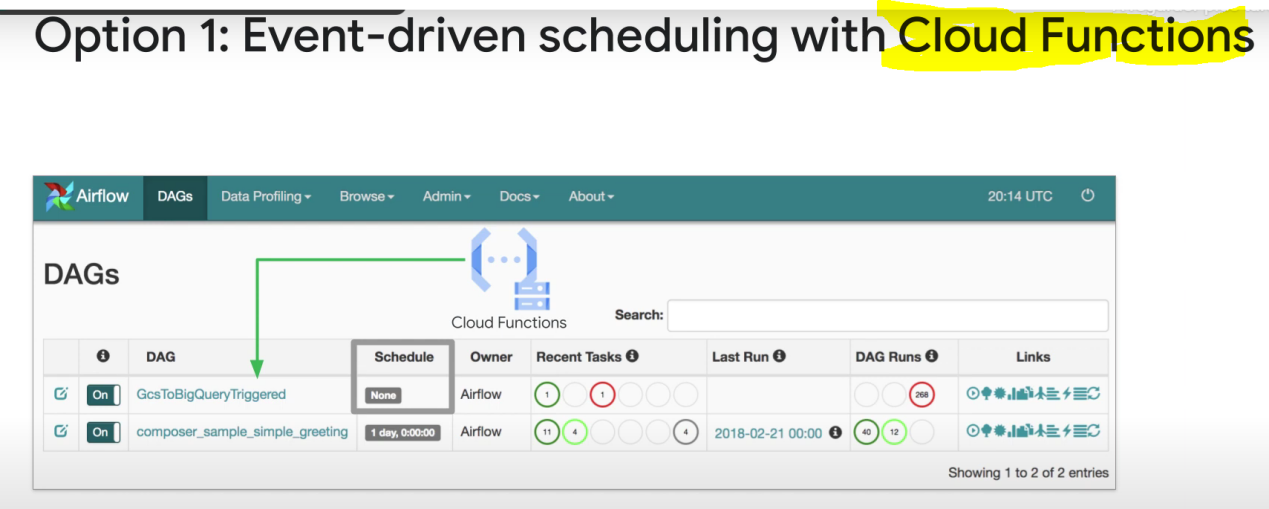


Scheduling Workflows

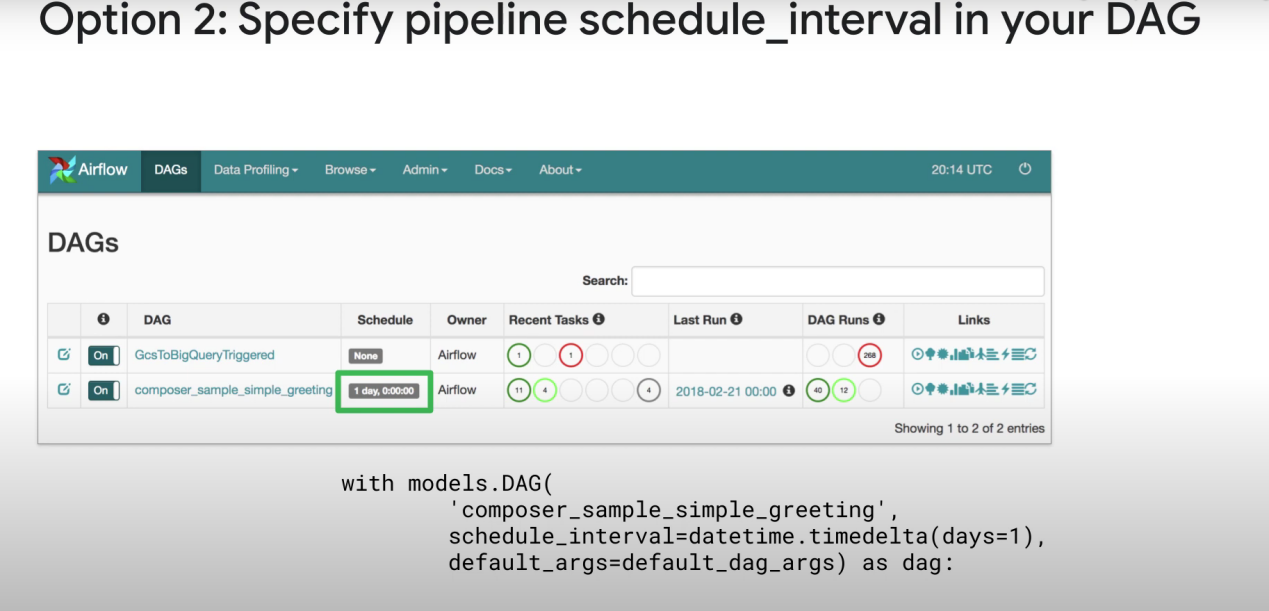
2 Types:

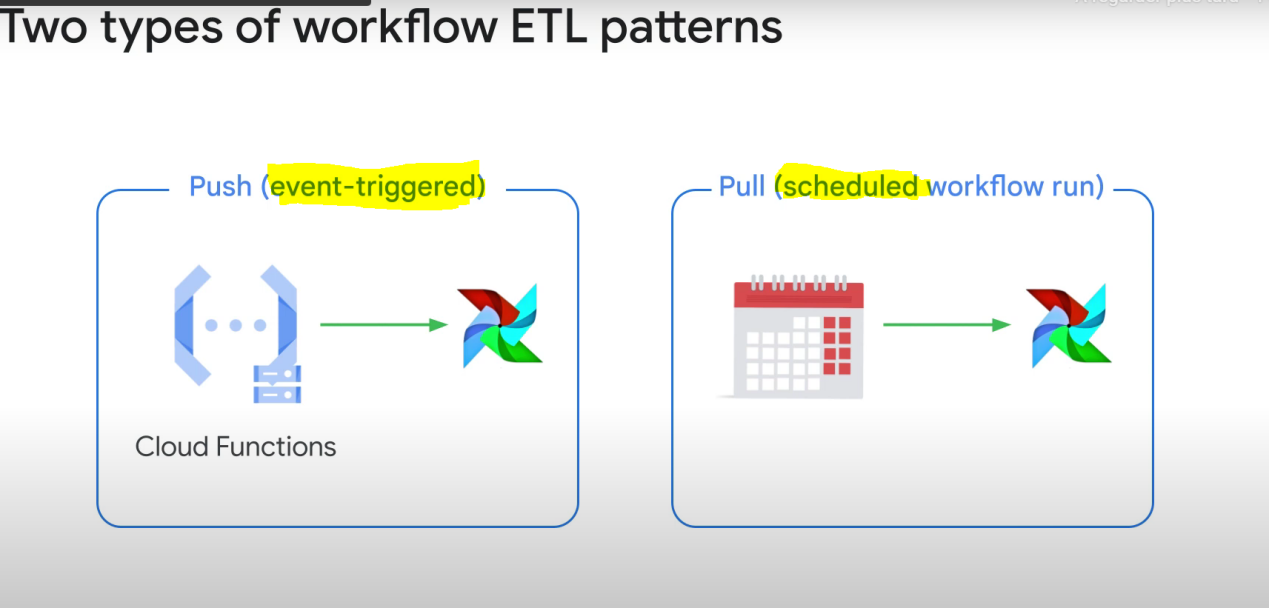


For event driven:



For scheduling:

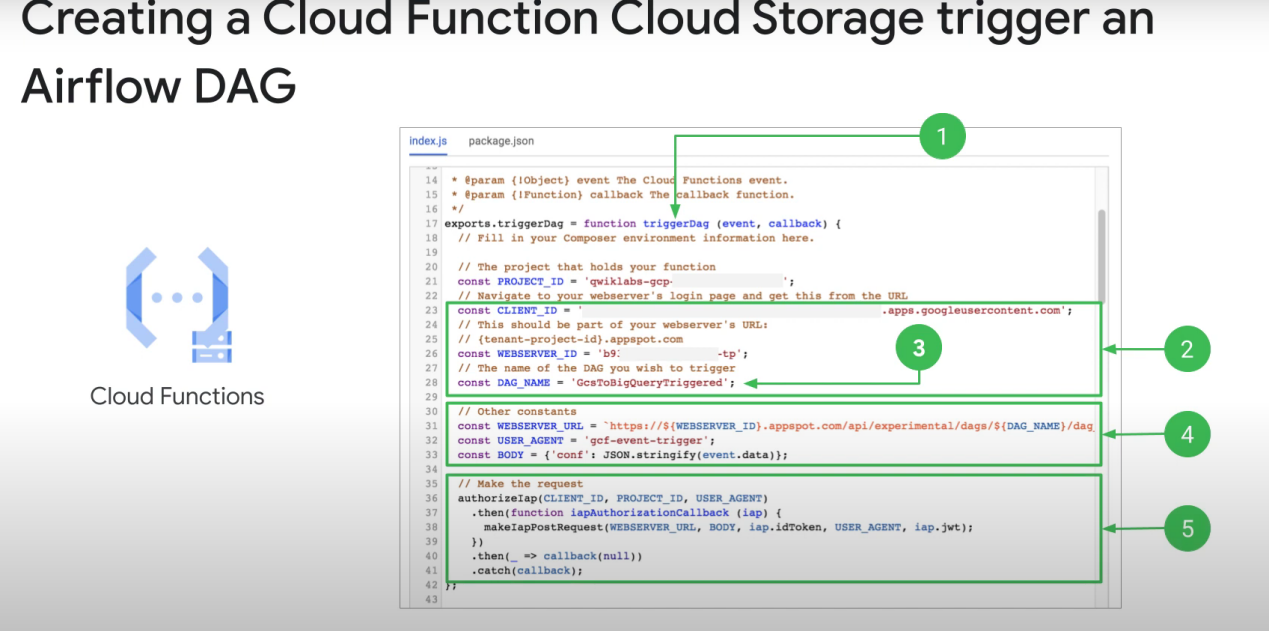




Push: adding a new file to GCS (or many others) and starting the workflow

Pull (scheduled): read data and start workflow

Template for Cloud Function Cloud Storage trigger of Airflow Dag

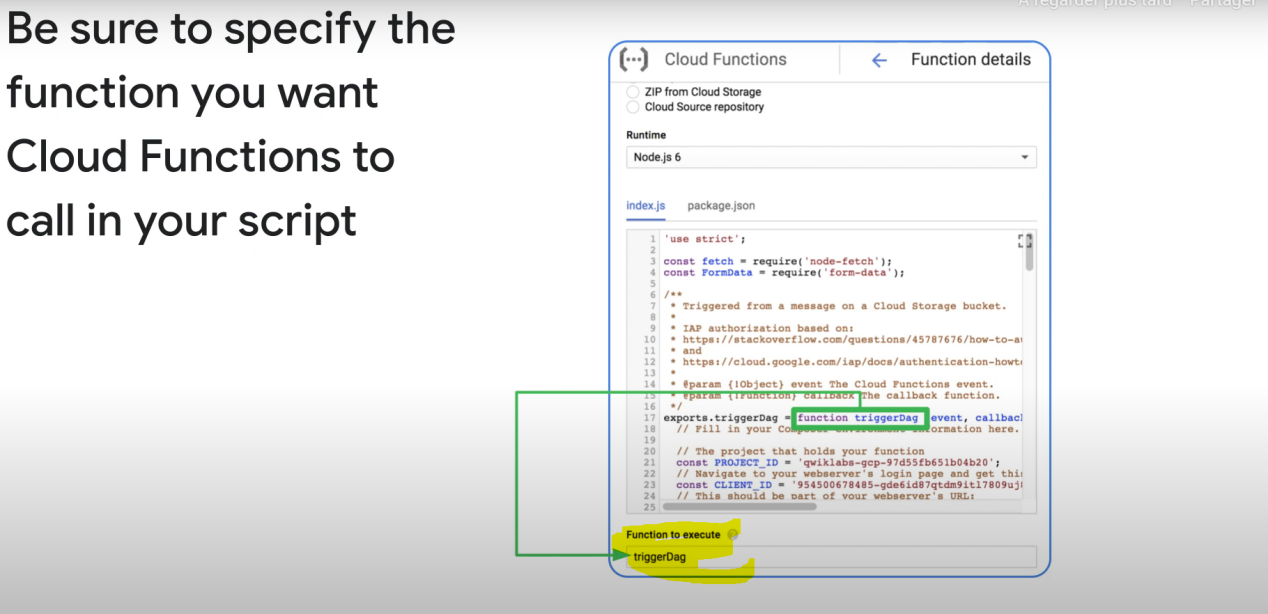


1: name of the function

2: different parameters to indicate which Airflow Dag to go to

4: The post URL for the request + some additional params

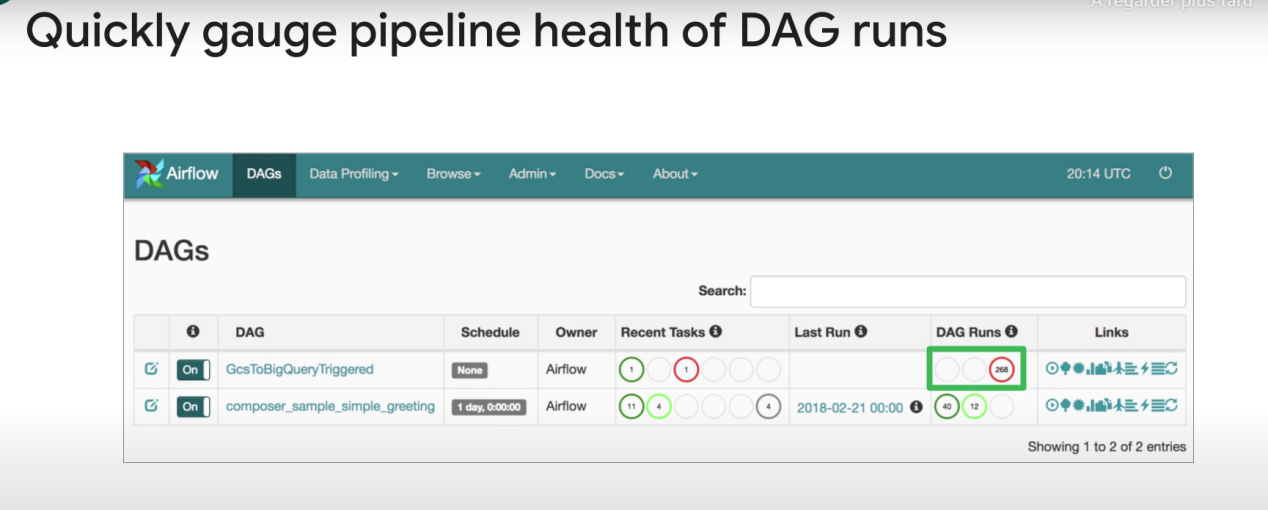
5: Doing the request



The function’s name is CASE SENSITIVE

**Monitoring And Logging**

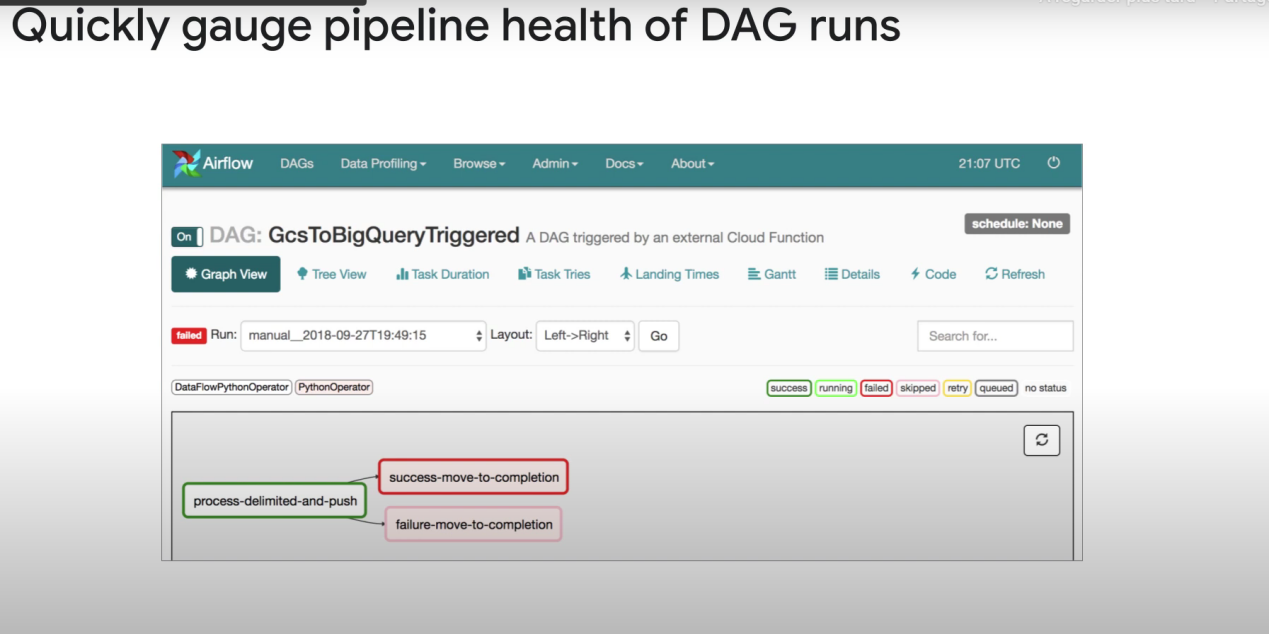
\* You can make your workflows auto-retry for a set number of retries if they fail.



268: failed

12: running

40: succeeded



You can check the logs for each node.

Note: we can also see the general google cloud logs for logging airflow errors. Some errors will only be found there (if there was an error with another google cloud service used before the trigger of the DAG/Airflow)