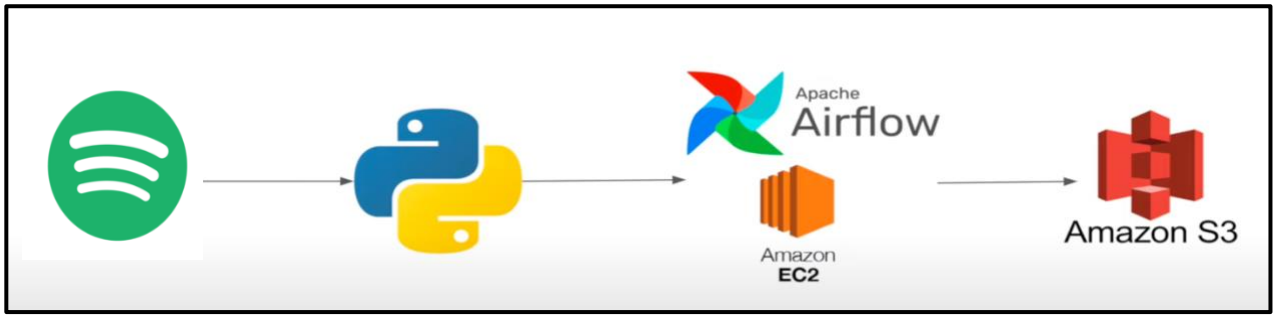
Spotify Data Pipeline using Airflow

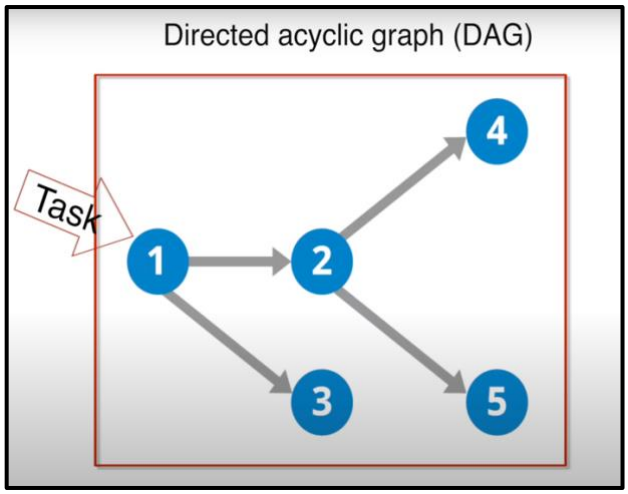
Objective:

The object or objective of a project involving Airflow and Spotify data could be to create a data pipeline for analyzing Spotify user data.



Airflow

It is an open source workflow (sequence of every task) management program for data engineering pipeline. It is a workflow orchestration tool we can build, schedule and monitor data pipeline. In airflow it is defined as DAG (Directed acyclic graph)



Task-1 Extract data from API
Task-2 Transform data
Task-3 Store data

Technology Stack Used

- Spotify API
- Apache Airflow
- AWS EC2 Instance and S3 bucket

Process

- I created a web app on the Spotify for developers page and got access to the client Id and client secret key.

- After obtaining the **client ID** and **client secret key** from the Spotify for Developers page, I proceeded to develop an ETL (Extract, Transform, Load) pipeline to extract top tracks, top albums, and top artists data from the Spotify API. Once the data was extracted, I saved it to an S3 storage bucket by specifying the path to the bucket. This approach ensured that the extracted data was securely stored in the cloud, ready for further processing and analysis.

- After provisioning the EC2 instance (use t3-micro and Ubuntu-AMI), I established a connection to it via SSH (Secure Shell) to configure and deploy Airflow. Commands used:
  - chmod 400 "spotify_airflow_key.pem"
  - ssh -i "spotify_airflow_key.pem" ubuntu@ec2-44-204-140-31.compute-1.amazonaws.com

- Run the above commands in the terminal where we have stored or key and all the data related files. I my case my folders name was *Spotify Airflow*

- Once the connection is set we need to run a bunch of commands to install and update the required packages:
  - *sudo apt-get update*
  - *sudo apt install python3-pip*
  - *sudo pip install apache-airflow*
  - *sudo pip install pandas*
  - *sudo pip install s3fs*
  - *sudo pip install spotipy*

- Now in order to check if the connection is working we will type in the command ***airflow*** and it will show us all the groups and commands that can be run.

- Now in order to run airflow server we just need to write *airflow standalone*
  - *In case you encounter an error related to airflow server not being setup try and install flask version 0.5 by default it will be on the 0.6 version.*

- Once the connection is set we get the username and password for airflow. From our instance we will copy the public IP DNS and mention :8080 in the end. But also enable an inbound rule for port 8080. This will allow the page to load and we can then login into the airflow console.

- We will now write our DAG Code

- In another terminal window we will setup our connection with Airflow and EC2 instance using the same steps we followed earlier.
  - We will write ***cd airfow/*** to set the directory and when we do *ls* we should some files inside that

    - We will write *sudo nano airflow.cfg*
        - Inside that we will change the name of the file from dags to the name of our dag code file in my case it was *spotify_dag.* Click control X and save it.
    - We will now create a new folder i.e.*, mkdir spotify_dag.* If I do *ls* I should see this new file.
    - Now we do *cd spotify_dag.* We have 2 files the ETL file and dag file and we need to store that into our ec2 machine.
        - Write *sudo nano spotify_dag.py and* copy the dag code inside this. Click control X and save it.
        - Same procedure for etl file i.e., *sudo nano spotify.py* and copy the etl code inside it and save it.
- After this is done we will stop the airflow server(Ctrl C) and the run the airflow standalone command again.
- Inside the airflow we will be able to see the files.
- We also need to set the permission to write from for ec2 instance to the s3 bucket.
    - For that we need to modify IAM role and select *AmazonS3FullAccess and AmazonEC2FullAccess. Give the role a name*
- Now when we run the Dag file inside Airflow and it shows the status color as green that means it is running successfully.

- Now when we refresh our s3 bucket we our data files inside that.