ANALIZA DUŻYCH ZBIORÓW DANYCH

PROJEKT – APACHE AIRFLOW – DEMO

Agnieszka Szynalik, Elżbieta Wierciak

Cel projektu

Celem projektu jest stworzenie efektywnego systemu zarządzania przepływem danych i zadań w środowisku rozproszonym, wykorzystując Apache Airflow. Używając publicznego API, zamierzamy wczytać zbiór danych z portalu Spotify zawierający informacje dot. utworów muzycznych oraz ich słuchaczy.

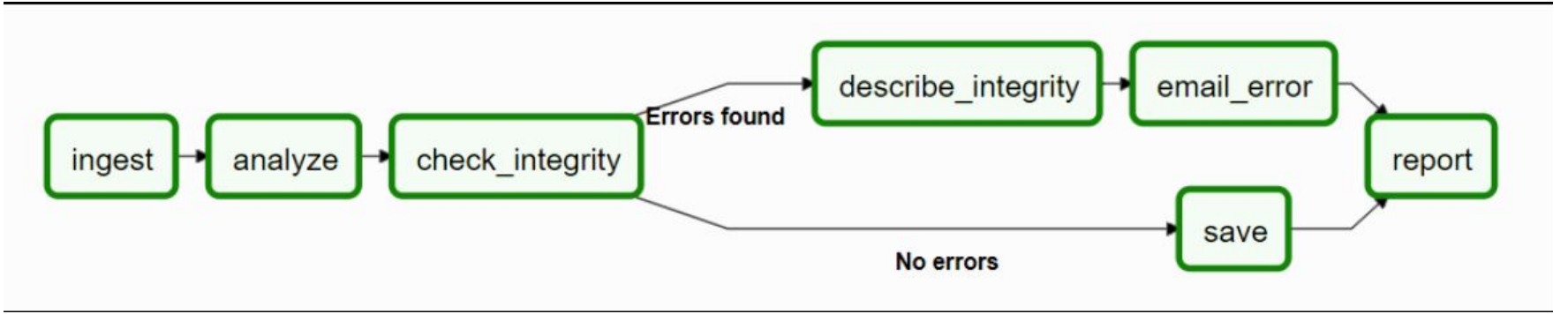
Apache Airflow będzie dobrym narzędziem do przetwarzania i analizy danych z tego obszaru.

Technologie

- Apache Airflow - narzędzie do zarządzania przepływem danych i zadań w środowisku rozproszonym
- Python -główny język programowania do implementacji skryptów, operacji na danych oraz integracji z API portalu Spotify.
- Sqlite3 - lekka i samodzielna baza danych, idealna do prostych zastosowań, zapewniająca łatwą integrację z aplikacjami Pythona.

Apache Airflow

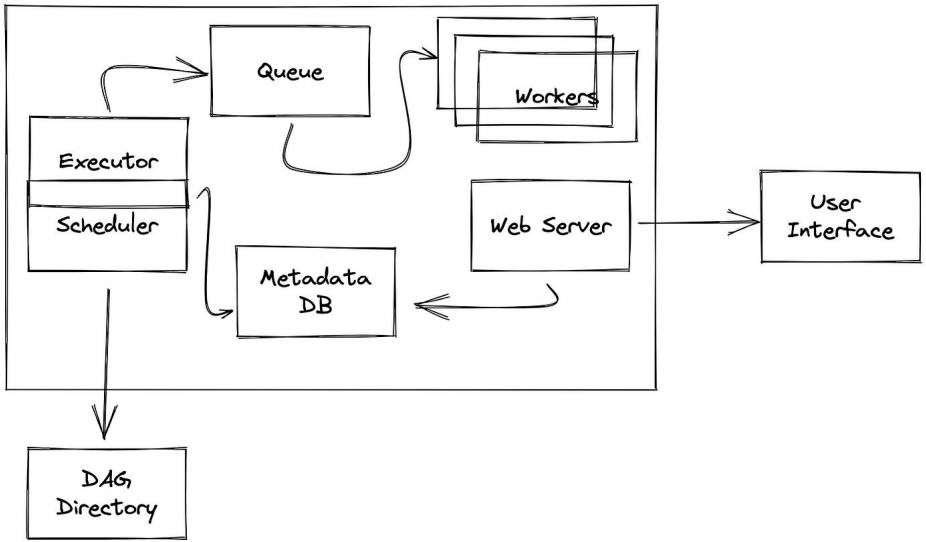
Architektura Airflow pozwala budować i uruchamiać przepływy pracy zwane Workflow. Workflow jest reprezentowany jako DAG (skierowany wykres acykliczny) i zawiera poszczególne zadania zwane Tasks, ułożone z uwzględnieniem zależności i przepływów danych:



DAG ma za zadanie określić zależności między zadaniami oraz kolejność ich wykonywania i uruchamiania ponownych prób. Z kolei Tasks opisują, jakie zadania wykonać (np. pobieranie danych, uruchamianie analizy, wyzwalanie innych systemów, ect).

Apache Airflow – Architecture

- Scheduler wyzwalający planowane przepływy pracy oraz przesyłający zadania do pliku wykonywalnego.
- Executor uruchamiający zadania.
- Webserver jako przydatny interfejs użytkownika do sprawdzania, uruchamiania i debugowania.
- Folder plików DAG.
- Baza metadanych do przechowywania stanu.



Dataset

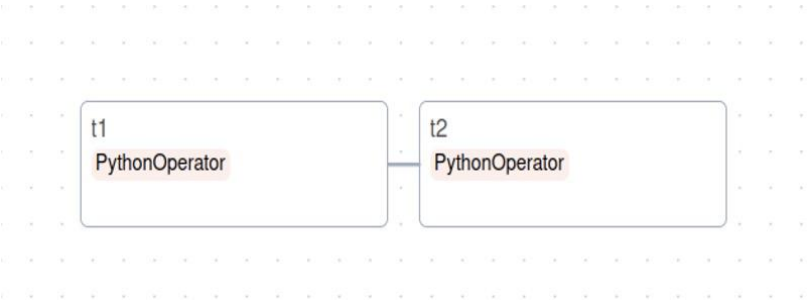
- https://huggingface.co/datasets/maharshipandya/spotify-tracks-dataset
- 114,000 piosenek z opisanymi cechami

danceability float64 0 – 0.99	energy float64 0 – 1	key int64 0 – 11	loudness float64 -49.53 – 4.53	mode int64 0 – 1	speechiness float64 0 – 0.97
0.676	0.461	1	-6.746	0	0.143
0.42	0.166	1	-17.235	1	0.0763
0.438	0.359	0	-9.734	1	0.0557
0.266	0.0596	0	-18.515	1	0.0363

DAG - WERSJA PODSTAWOWA

T1 – wczytuje dane z pliku csv i zapisuje je w bazie danych

T2 – wykonuje obliczenia: dzieli dane na trzy zbiory według obliczonych progów cech „loudness”, w każdym z tych zbiorów grupuje dane po gatunku muzycznym i zlicza ile jest utworów w danym zakresie „loudness” o określonym gatunku. Na koniec zestawia razem te wyniki.



DAG Summary	
Total Tasks	2
PythonOperators	2

DAG – WERSJA ROZSZERZONA

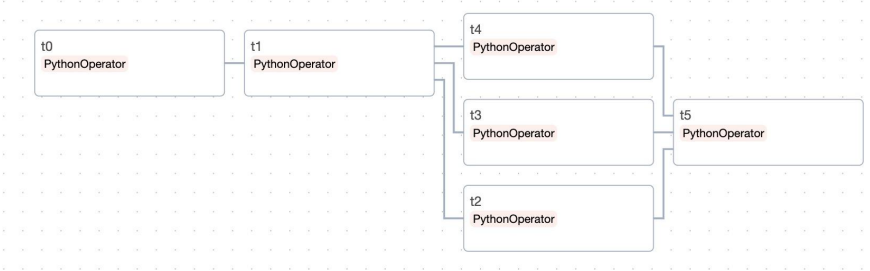
Ten DAG jest rozbudowaną wersją poprzedniego – zadanie drugie zostało rozbite na pięć podzadań, z których trzy wykonują się równolegle.

T0 - wczytuje dane.

T1 - wyznacza trzy zakresy wartości „loudness” na podstawie wartości brzegowych.

T2, T3, T4 - zlicza ilość gatunków muzycznych w obrębie danego zakresu „loudness”.

T5 - tworzy zestawienie dla wszystkich wyznaczonych zakresów.



DAG Summary	
Total Tasks	6
PythonOperators	6

Wynik eksperymentów

Tabela zestawiająca uśredniony czas wykonania poszczególnych diagramów po przeprowadzeniu kilku prób:

WERSJA PODSTAWOWA	WERSJA ROZSZERZONA
Duration: 00:00:15	Duration: 00:00:27

Eksperyment podstawowy przeprowadzono szybciej niż ten, który obejmował równoległe zadania. Może to wynikać z faktu, że zużycie czasu zostało poświęcone głównie na dostęp do bazy danych, a nie same obliczenia.

Wnioski

Aby dostęp do bazy danych nie miał znaczącego wpływu na jakość wyników możemy:

- zdobyć lub stworzyć znacznie większy zestaw danych
- dodać bardziej wymagające obliczenia do tasków
- znaleźć szybszy sposób zapisywania i wczytywania danych

Dalsze kroki:

- przeprowadzić więcej eksperymentów, aby uwiarygodnić otrzymane rezultaty
- rozszerzyć bazę zadań do wykonania
- dodać scheduler, aby uruchamianie diagramów wykonywało się cyklicznie

Dziękujemy

Agnieszka Szynalik, Elżbieta Wierciak