Design and approach Document

Introduction:

This document will give you the overall information on the design, selection of tools and packages used in different stages of implementation of an airflow data pipeline.

Tools and packages used:

Python: 3.8.10

Pyspark: 3.0.0

Airflow: 2.2.2

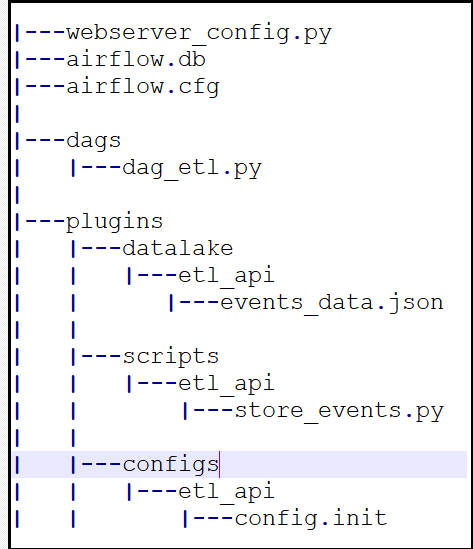
Database (MySQL): 8.0.23

Design:

**AIRFLOW PIPELINE:**

I am listing down the key points while implementing the pipeline.

**DAG Structure:**



The DAG structure above is used for this task and going through the structure below.

dags: This directory contains the DAG files which holds the tasks, schedule and order in which the task can be used along with the required plugins being used in this module. This is main runnable file and start of the pipeline.

|  |  |
| --- | --- |
| **Params** | **Values** |
| dag\_name | dag\_etl |
| dag\_id | dag\_etl |
| schedule | Every 2 hours |
| no\_of\_tasks | 4 |
| python operator | 2 |
| branch operator | 1 |

**Explaination:**

**Task1- fetch\_events\_api:** fetches the data from API, right now the event, from\_timestamp, to\_timestamp is hard-coded as we have dummy data. The API endpoint is hard-coded with “localhost:5000”. Please run the API server at this port to have this task succeeded.

API params:

event="event\_1"

from\_timestamp = "2020-10-22 06:36:00" #datetime.strftime(datetime.now()-timedelta(hours=2), dt\_format)

to\_timestamp = "2020-10-22 08:36:00" #datetime.strftime(datetime.now(), dt\_format

You can uncomment above two lines to fetch actual data.

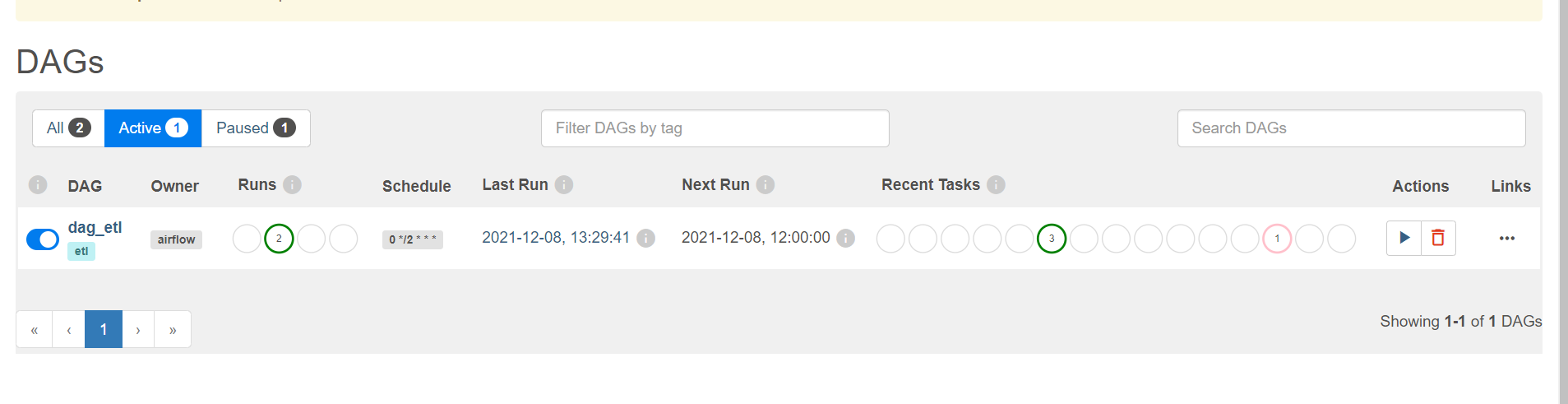
**Task2- store\_to\_datalake:** This task takes the response from fetch\_events\_api task and store in json file at datalake at predefined path. /plugins/datalake/{dag\_name}/file.json. This path can be more dynamic as in the folder structure should be {YEAR}/{MONTH}/{DAY}/{Hour} like so.

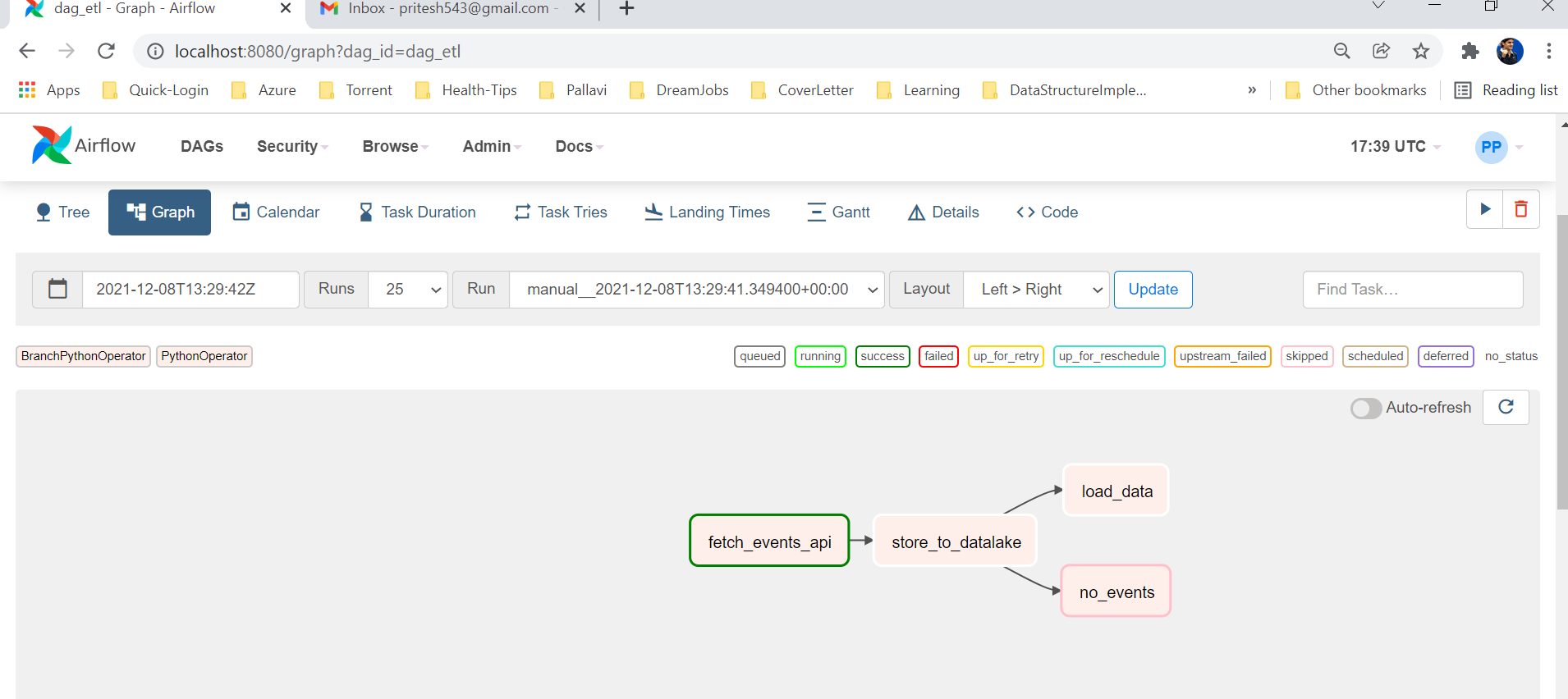
**Task3- load\_data:** This task executes another PySpark script which read the data from data lake path defined above to spark df and later load the data to the main table. No transformation done here simply loaded to temp table in spark and from spark temp table to main table in MySQL.

**Task4- no\_events:** If the response is not proper or no data in response then it simply executes this, which helps to monitor the pipeline.

***Pipeline:*** fetch\_events\_api >> choose\_next\_task >> [store\_event\_data, no\_events]

The diagram below is taken from airflow UI to show the graph representation of tasks and their dependency.





The DAG name is “dag\_etl” and this pipeline has four tasks:

1. Fetch events data from API (fetch\_events\_api): This task fetch the data from API, right now it is fetching data from
2. Store in data lake (store\_to\_datalake)
3. Transform and load to database (MySQL) (load\_data)
4. If no data received from API (no\_events)

**Decision on selection of tools:**

MySQL:

MySQL is great tool for read and write I/O throughput. MySQL is quite fast in querying the data.

Pyspark:

I chose PySpark in place of pandas for ETL pipeline because it offers significant performance bonus when large amount of data is required to process and transform.

Note: I can speak more about on the decision part but for the document I am making it short.

Things to do or to improve pipeline

1. The audit part is pending in this flow. Usually, I prefer to have audit database both functional audit as well as standard audit to analyze the process at each stage and helps outstandingly in case of observations. The audit also helps to learn functional understanding of the entire pipeline.
2. There are few places where things are hard-coded but usually I do not hard code anything but due to the time constraints I have done very little of it, kindly ignore it.
3. Few things need to be set to behave this pipeline dynamically which is also missing.