Concept flow diagram

Harriet Sibitenda 25/September/2021

Introduction

The sensors deployed in different places collect data. Data is collected and analyzed to provide key insights to the stakeholders of interest from the business sector. Adoption of this AI technology should boost client facility as well as production of workers, basing on the insightful ideas generated.

Project proposal plan

The sensors will provide the data that will be cleaned and structured by our database using MySQL tool. Using the Airflow tool, tasks needed to generate data pipelines for each task will be initialized and run by DAGS (Directed Acyclic Graphs). The dags folder will be generated to necessitate the periodic updates of the Airflow container. The DAG tests will be uploaded onto the dbt (Data build Tool) for transforming all activities performed on data as processes. The Airflow container will be required to update the model automatically through tasks like (data pulls and featuring engineering pipelines, model training, model deployment, and much more. The tasks will be generated as SQL files.

The dbt will provide the data schema for the data and result table for the tasks. These transformations are sequential and unique to avoid duplication and dependencies. The result at this part will be ready for further handling as data pipelines in the DHW (Data Ware House). The analyst would be analyzing the results at the front end to aid decision making.

The project processes are illustrated in the diagram below:

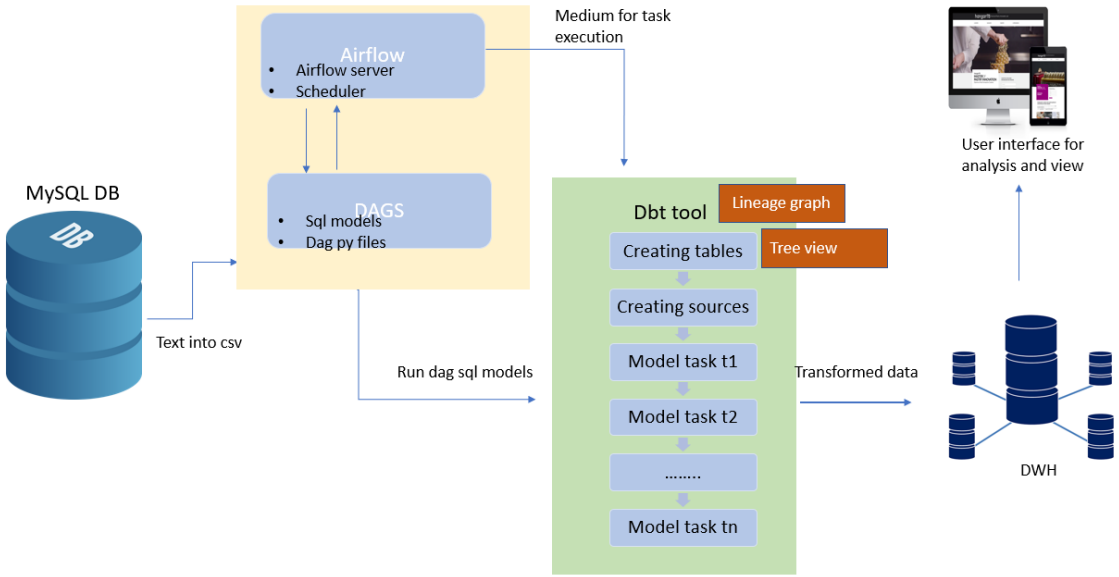

Figure 1: The concept flow diagram

Methodology

Guide tutorial

It is very important to follow a guiding tutorial for each of the steps required before staring to officially operate on your project. A list of recommended files is referenced in the biography. The diagrammatic structure above gives an idea about the steps involved to engineer raw data into pipelines that are stored at a data ware house for future reference.

Data collection

Data was collected from the ucdavis data site , one can find different forms of data described as parquet data, and or sensor data in CSV formats with size ~1.5Gb uncompressed each. The size is too big and requires uncompressing. This would also require you to version the data using DVC so as to report the track version on the GitHub. This was stored in the data folder named data, and we extracted two files, one consisting of information describing the stations where sensor data is obtained and the other consisting of data readings that could be tracked by time in form of seconds, minutes and hours.

Setting up MySQL database

The Git hub repo “dataeng” was created and cloned using the VS code tool. This was accessed as a folder from home environment. The folders needed were added. In particular, the scripts folder was used to generate python codes needed to command MySQL to create a database for our data.

MySQL as a tool was used to create a server tool for creating data structure for the files extracted. This was done with configurations of the python server, where data was loaded as text and output as entities or tables with defined structure using the schema sql files.

Setting up the airflow

The Apache airflow was to be installed so as to update the model automatically through tasks of all DAG files. This environment requires using Linux and so using WSL is highly needed to support the installations on windows machine. This can be installed locally and also with the integration of docker at the server end. We needed to create configurations files of: airflow.cfg, webserver_config.py, .env file (this was important to setup in the airflow folder), docker-compose.yml was also created to enable running of the airflow server and scheduler at the docker server end. Note that there was need for one to have installed docker and docker compose before. This implies that the sever would not be interrupted after running command docker-compose up airflow init-1. We created other folders inside the airflow folder named dags to logs and to plugins. The password and user name were required to be used when running localhost:8080 on any web browser of choice. This is basically consisting of examples and illustrations for other study cases that Apache airflow supports.

Creating DAG files

In the dataeng folder, we added a copy of the airflow folder were configurations of the sever were set. Under the dags folder we were to create .py files needed orchestrate airflow. A file needed to command extraction of data from MySQL and the other testing the runs using BashOperator and MysqlOperator. The dependencies in each file were defined to configure sequence of tasks generated. The files were run to identify the tables of views created along the schedules. The airflow is displayed as follows:

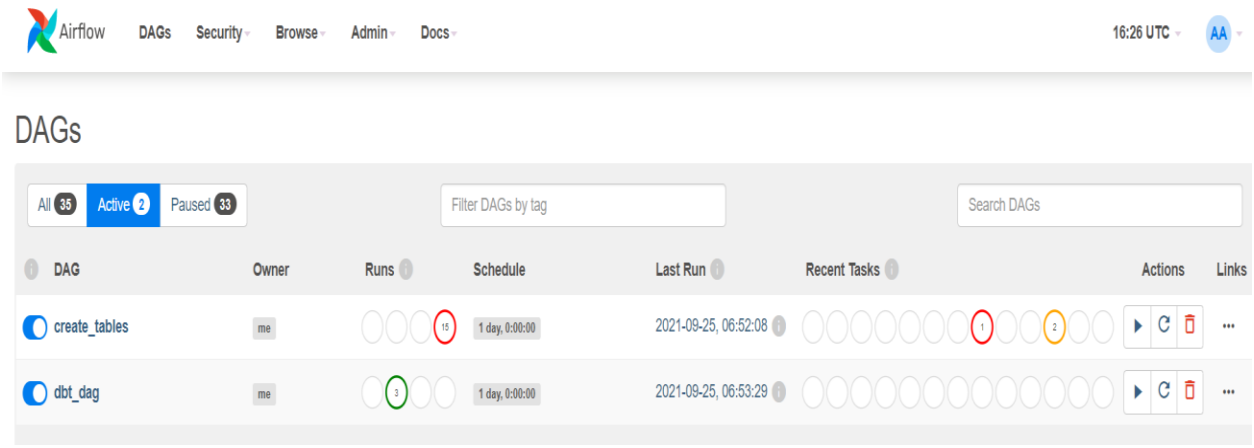

Figure 2 Illustrating DAG runs of files on local computer

Setting up dbt

There was need to set up the data ware house. This was done using dbt and other tools of snowflake and postgres. An account was generated and this was to set up the production and development environment in the cloud. A number of tools like snowflake and postgres were identified to be proving a cloud to host the service. Considering snowflake, the registration for an account was done and required specific distinction of all the necessary components such as; account, username, database, data ware house and many more. A link between the cloud and dbt was needed to generate the production and development databases in the cloud. The data ware house was generated, this is illustrated below;

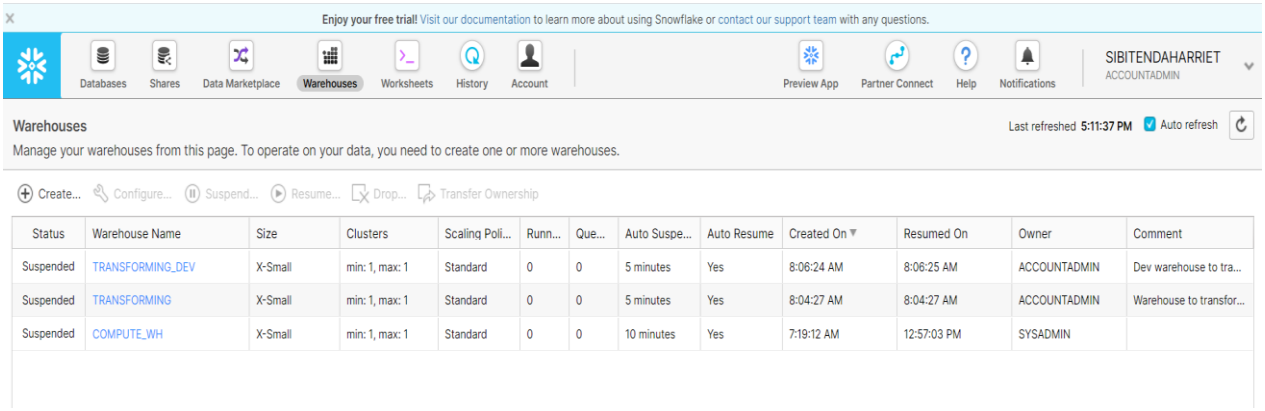

Figure 3:Illustration of the databases in the data ware house

Analytics visualization

The data pipelines in form of sequential tasks are generated and uploaded to the data ware house. This data is of transformed format and can be further utilized for analysis. The lineage graph and tree view are used to display the data.

Conclusion

The challenges encountered in this task would be arising from software incompatibilities that delay the access to the right method of using the tools required. There is possibility to complete the necessary tasks required to complete this project in the near future.

References

1. Installing dbt https://docs.getdbt.com/dbt-cli/installation/#pip
2. Introduction Videos on dbt https://www.youtube.com/playlist?list=PLy4OcwImJzBLJzLYxpxaPUmCWp8j1esvT
3. Redshift config; https://docs.getdbt.com/reference/resource-configs/redshift-configs/
4. Docs from Gitlab: https://about.gitlab.com/handbook/business-ops/data-team/platform/dbt-guide/
5. CLI command reference: https://docs.getdbt.com/reference/dbt-commands/
6. Basic Introduction to dbt https://www.kdnuggets.com/2021/07/dbt-data-transformation-tutorial.html