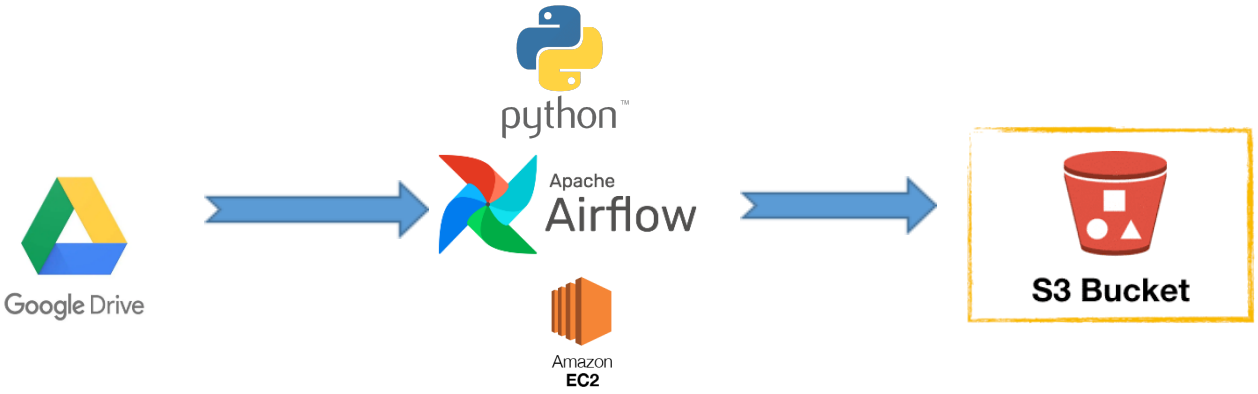
Managing ETL Workflow using Airflow

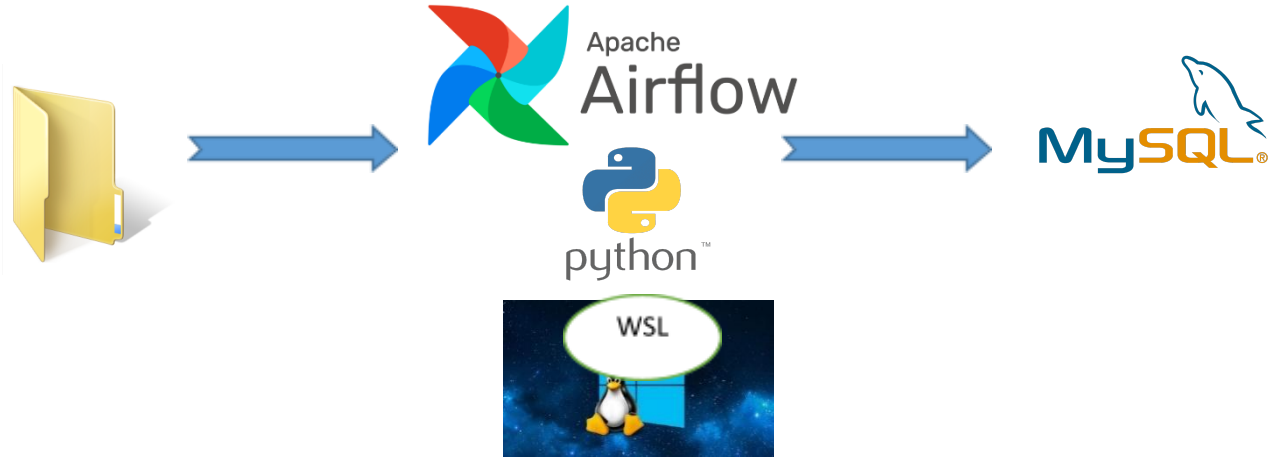
Objective:

ETL is the building block of data engineering. To understand much better the concept of managing ETL workflow, I have tried in this project to extract data stored on google drive using python, and transform it using pandas library as per the required way and load it on s3 bucket of AWS. To manage this entire workflow I have used apache-Airflow and AWS ec2, IAM, and s3 bucket.

Project implemented on Cloud



Project implemented on Local Machine



Though Python, Airflow, and all of its libraries can be installed on windows, but the problem is that one of airflow's required lib called pwd is only available for Linux version of python so you will face problem implementing this project. One better workaround for the people who have only win 10 machine is to install WSL or windows subsystem for Linux. After enabling WSL from windows features, reboot it and launch SUSE Linux from Microsoft Store. Update it and rest of the steps will be same as explained for EC2 machine.

Apache-Airflow:

Apache-Airflow is a platform to programmatically author, schedule and monitor workflows. Apache Airflow is an open-source workflow orchestration tool that automates and schedules complex data workflows through Directed Acyclic Graphs (DAGs). It offers several key benefits. Firstly, Airflow allows users to define workflows as DAGs, specifying tasks and their dependencies. These workflows can be scheduled to run at specific times or intervals, providing automation and consistency. Airflow's extensibility is a major advantage, enabling integration with a wide range of technologies and services. It also offers robust error handling and monitoring capabilities, ensuring reliability in data pipelines.

Airflow's capabilities include a rich set of pre-built operators for common tasks, the ability to define custom operators, and support for dynamic and parameterized workflows. Its extensible nature allows users to incorporate cloud services, databases, and other tools seamlessly. With its comprehensive scheduling, monitoring, and error-handling features, Apache Airflow is a versatile choice for orchestrating data workflows in various industries, ensuring data accuracy and timeliness while reducing manual effort and errors. https://airflow.apache.org/docs/apache-airflow/2.2.5/

Amazon S3 (Simple Storage Service)

It is a versatile, highly scalable, and durable object storage solution by AWS. Users can store data as objects within "buckets." S3 provides high data durability, fine-grained security, and global accessibility. It's cost-effective, supports versioning, data lifecycle management, and seamless integration with analytics tools. In your project, an S3 bucket efficiently stored your transformed data, ensuring durability, scalability, and easy integration with other AWS services or applications.

Windows Subsystem for Linux (WSL)

It is a compatibility layer within Windows that enables running Linux distributions alongside your Windows environment. WSL provides a powerful and flexible environment for development, testing, and running Linux applications on Windows. It offers seamless integration with Windows tools, efficient resource utilization, and access to a wide range of Linux software. In your project, you leveraged WSL to run Linux-based applications, harnessing its compatibility and performance benefits within the Windows ecosystem.

Python was the primary scripting language in my Airflow project, allowing me to create custom operators, sensors, and tasks tailored to my specific needs. Its compatibility with Airflow's PythonOperator made it an excellent choice for executing data pipeline steps. Each DAG is nothing but a python script

Pandas was a cornerstone of my Airflow project, serving as the key tool for data manipulation and transformation. Its DataFrame structure simplified data handling, allowing me to efficiently preprocess and analyze data before moving it between different stages of the pipeline. I used to transform data in desired structure/

SQLite was used in my Airflow project to handle local database tasks, such as managing task statuses and logging. Its lightweight and self-contained nature made it a practical choice for maintaining metadata and facilitating task execution. Airflow uses sqlite to store its metadata.

Google Drive integration in my Airflow project was crucial for accessing and sharing data. Its cloud-based storage capabilities allowed seamless data extraction and retrieval, particularly useful when working with external data sources. Each file stored in google drive is give a unique id. I used that id to extract data. I have uploaded 3 files and made them public in sharing. Importantant thing to note here is that all files have got ids. Instead of names. See my DAG.

MySQL was employed to manage structured data efficiently within my Airflow project. Its reliability and strong data consistency features made it suitable for storing critical pipeline metadata and ensuring data integrity throughout the workflow. The transformed data has been loaded to my sql in my second task of the DAG.

Let's implement it for AWS Cloud:

Assuming that you already have AWS free tier account, and a windows 10 machine. Sign in to AWS console. Launch an ec2 instance, give a name, using suse linux image, create a new key/pair or use existing, instance type t2 micro, create a security group but allow all traffics and launch. Click on 'connect to instance' and copy SSH script.

On windows machine, Open cmd go to folder where you have saved the dot pem file. Paste the ssh script copied above and press enter. You will see Linux prompt. You are connected to ec2 server.

Use the following 20 commands one by one in the cmd window after connecting to ec2 by removing my comments.

1. sudo zypper refresh
2. sudo zypper update
3. curl -O https://bootstrap.pypa.io/pip/3.6/get-pip.py
4. sudo python3 get-pip.py
5. rm get-pip.py
6. pip install --upgrade pip
7. pip install --upgrade setuptools
8. pip install apache-airflow
9. pip3 install apache-airflow
10. pip install pandas
11. pip install s3fs
12. airflow users create --username asawir --firstname Asawir --lastname Jinabade --role Admin --email xxxxx@gmail.com
    # create user and give a simple password
13. airflow db init
14. cd airflow # to change to airflow folder
15. vim airflow.cfg

    # to check dags_folder path go to that path
16. mkdir dags
17. cd dags

18. vim main.py # and create a DAG file main.py using vim main.py and paste the python code and save :wq enter

19. airflow scheduler # to start scheduler

in another cmd window connect ec2 and then

20. airflow webserver

Perform the following operations in aws console:

1. Create s3 bucket, give a unque name. mine is s3://asawir-airflow-project-bucket here my processed file will be saved.

2. IAM: create a new IAM role which allows full access to ec2 and s3 bucket by clicking to ec2 instance, action, security, modify role, create IAM role. Create role, aws service, ec2, next, search ec2, select "AmazonEC2FullAccess" clear, and search s3 enter, select "AmazonS3FullAccess" click next, give a name and click create role. From instance, action security and modify role select the newly created role and click update IAM role .

3. From instance summary screen copy public address. Paste as in url section of browser like "ec2-13-126-65-119.ap-south-1.compute.amazonaws.com, add a colon and 8080 at the end press enter.

4. Enter user id and password to enter as per para 12 above, into the airflow web browser, click you DAG "booking_ingestion_with_google". Trigger it and see the tasks visual are green by refreshing, if all green go to s3 bucket and see that the processed file "processed_data.csv "is uploaded.

1. If you don't find your DAG "booking_ingestion_with_google"
   see the error in red color and solve by re-opening your dag file. i.e. main.py in dags folder and change accordingly.

   Thank you