Data Pipeline

An Apache Airflow DAG and Python Script

1. Project Description:

This is a Data pipeline that automates the downloading, cleaning, and uploading of data in two different data bases depending on the version: MS SQL Server in the Python script, and MySQL in the Apache Airflow Version. The purpose of this project was the gain hands-on experience in creating data pipelines and utilising the Apache Airflow tool.

2. Problem Statement:

This project aimed to solve the lack of hands-on experience in creating data pipelines, and lack of comprehension of Apache Airflow and its ability. Having finished my data cleaning and visualisation project "Aim_Lab", and frequently seeing 'Apache Airflow' and 'data pipelines' in job descriptions, it felt like the natural next step in my learning path.

3. Solution Implementation:

Before writing any code, I brainstormed with ChatGPT about what software and libraries to use to clean and store the data. I worked out the template of an Apache Airflow file via various Youtube tutorials, the documentation, and ChatGPT. Now I had a basic outline for the project, and the solution was as follows:

Setting up the environment:

1. Airflow can only be used on Linux. So I installed the Ubuntu Windows Subsystem for Linux (WSL) from the 'Turn Windows features on or off' feature via the Control Panel
2. I opened an Ubuntu Linux terminal, updated it, downloaded python, Airflow and its dependencies
3. In the default location where Airflow reads dags, I created the .py pipeline file
4. Imported, Airflow, Datetime, Requests, Pandas, SQLalchemy, Pyodbc Libraries.
5. Created 2 objects, def extract_data():, def transform_and_load_data():
6. In extract_data(): Used requests library to download the data from a source and save it to the local Linux virtual machine
7. In the transform_and_load_data(): cleaned the data using pandas by:
8. Pandas.read to read the file from the local Linux virtual machine
9. Pandas.drop to remove unnecessary columns
10. Pandas.rename to rename columns appropriately
11. Pandas.iloc to swap values into correct columns
12. Pandas.to_csv to save the cleaned data back onto the local Linux virtual machine
13. Now this is where I faced the biggest problem of this project, and arguably in my computer science journey at this point. Pushing the data into MS SQL Server in an Airflow task would just not work, and I had to determine what issue was clogging this data pipeline and causing the Airflow task to fail. Was it my code or some strange dependency issue between Airflow and the databases?
14. I eliminated Airflow from the chain, creating a separate python script which would be the same code but not in an Airflow task.
15. It still didn't work. The pipeline would download, transform, save the data to the local machine, but failed uploading it to the MS SQL Server
16. I tried pushing the data into MYSQL instead, and it worked. The problem was the MS Server credentials. Now to get this working in Airflow.
17. I copy and pasted the python script back into an Airflow task
18. I defined the default_args to retry once on failure, and assigned the DAG arguments to create a DAG instance so its reflected in the webserver

19. I Defined the tasks using PythonOperator, and set their dependencies in order for the pipeline to execute each step in the correct order. Now to test it.
20. I opened a Ubuntu terminal (Windows Service for Linux) and initialised the database with 'airflow db init'
21. Started the airflow scheduler with 'airflow scheduler'
22. Opened another Ubuntu terminal and started the airflow webserver with 'airflow webserver'
23. Opened my internet browser and entered 'localhost:8080/' in the URL
24. Logged into airflow webserver with created credentials via ubuntu terminal
25. Selected the name of the pipeline in the list of DAGS 'PIPELINE_dag'
26. Press run, All green, it works
27. I opened MYSQL on my local machine to confirm the data uploaded, and it's a success.
28. I was still yet to solve the credentials problem with MS SQL Server, I continued the fight
29. After days of research, forums and prompting ChatGPT, I prompted ChatGPT in a way and it finally delivered code that worked. The ChatGPT solution was slightly different to the SQLalchemy documentation and the information on the forums, but it worked and I understood it.
30. Now that I had 2 versions, differentiated by that one is in an Airflow task that pushes data into MYSQL, and the other is a Python script that pushes data into MS SQL Server, I did this to show some versatility in creating a pipeline.

4. Results and Impact:
Both data pipelines proved a success and achieved their function of downloading, cleaning, and storing data into the designated database. I have also gained significant hands-on experience and insight into creating data pipelines, and the integration of tools, software, and environments used such as Airflow, Linux terminal, WSL and Pandas. This project was a coming together of lot of previous learned skills and felt good to create something very functional and scalable.

5. Technical Skills Demonstrated:
ChatGPT prompting, Windows OS, WSL, Linux Terminal, Airflow, Pandas, Requests, SQLalchemy, Pyodbc, Datetime, MYSQL , MS SQL Server, Creating date pipelines

6. GitHub Repository:
https://github.com/Davidooj/Projects/tree/main/Data_Pipeline

7. Key Takeaways:
The scalability and automation of data pipelines: This project taught me the scalability of data pipelines. Handling 1000's of csv files to be cleaned and stored can be just as fast as one (especially if they share the same format), and combined with Airflow, these tasks can be automated and repeated in desired intervals. This provides businesses an unmatched advantage in efficiency and business intelligence compared to adversaries.

This Project is a data pipeline. With one click of a button, the data will download, clean and format, and upload into different databases depending on the version. MS SQL Server in the python version, and MYSQL in the Apache Airflow version.

The main objective was to gain experience in creating data pipelines, to utilise and learn the tools, and understand the potential of data pipelines.

This project has two versions, one in Python and the other in Apache Airflow. I wanted to learn how to make data pipelines in Apache Airflow, as it is a scalable and valuable tool. And then utilised the opportunity to push the data into two different databases.