Airflow + Dataproc + Hive Integration

This project demonstrates how to use **Apache Airflow** with **Google Cloud Dataproc** to orchestrate a data pipeline that:

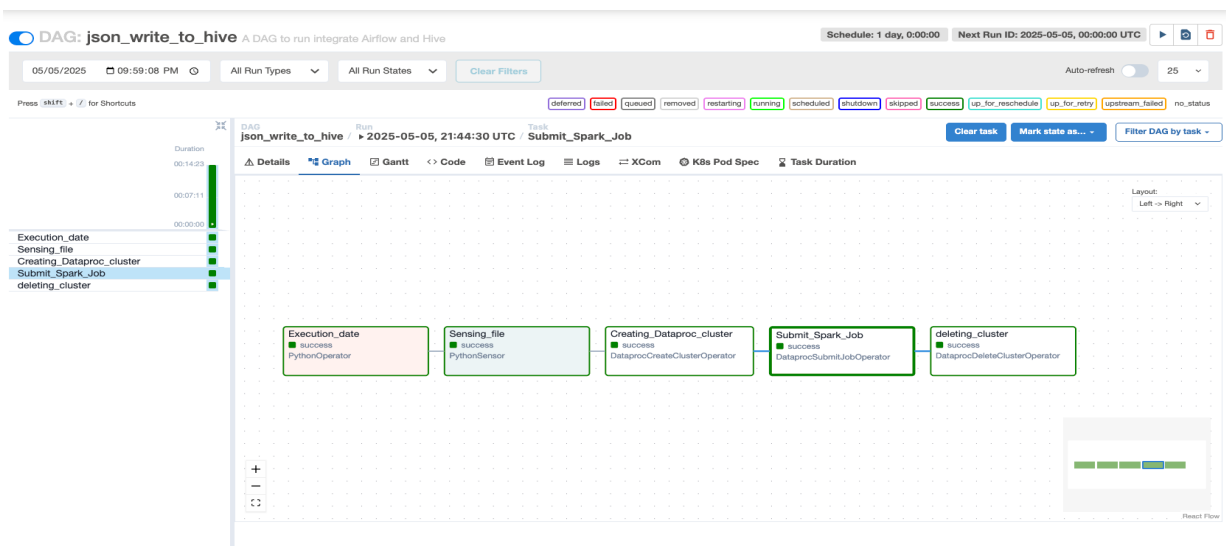
1. Reads a CSV file from a **GCS bucket**.
2. Runs a **PySpark job** on Dataproc to filter employee data based on salary.
3. Saves the result into a **Hive table**.
4. Stores a backup in **GCS as a Parquet file**.

📂 Project Structure

- spark_ex_airflow/assignment1/data/: Contains raw employee CSV files (employee-<date>.csv)
- spark_ex_airflow/assignment1/hive_data/: Hive warehouse output (structured as Hive metastore directories)
- spark_ex_airflow/assignment1/spark_job/: Contains your spark_job.py script

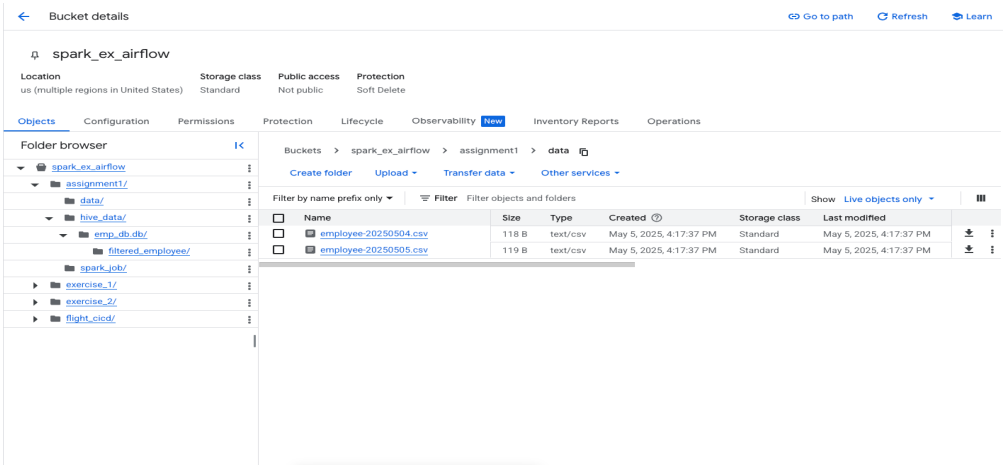
---

🛠️ DAG Flow



1. **Execution Date Extraction**: Determines file name to process.
2. **File Sensor**: Checks for the presence of the corresponding CSV in GCS.
3. **Cluster Creation**: Spawns a temporary Dataproc cluster.
4. **Spark Job**: Executes the PySpark job.
5. **Cluster Deletion**: Cleans up resources after job completion.

📂 Sample Input Files



Example files used:

- employee-20250504.csv
- employee-20250505.csv

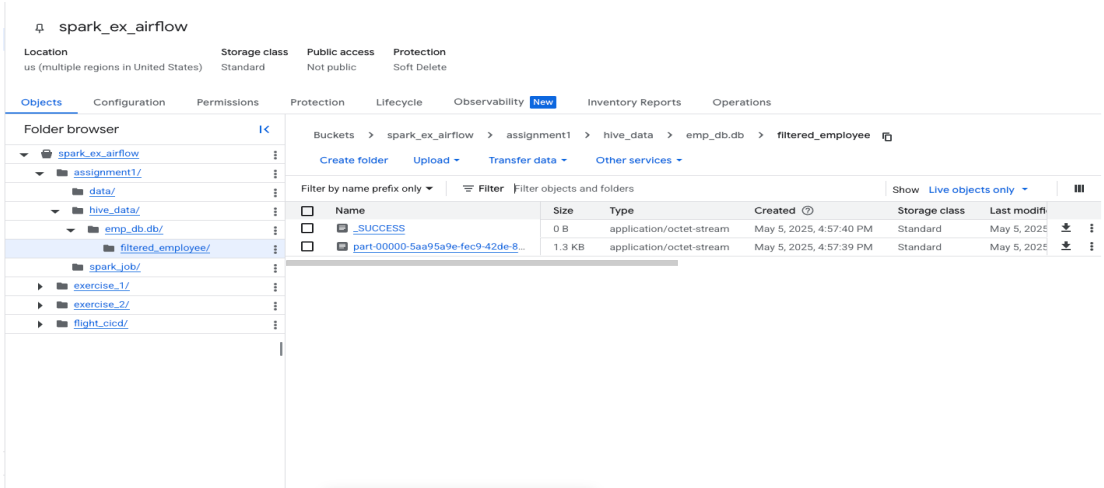
🔧 PySpark Job Summary

filtered_data = data.filter(col("salary") > 50000)
filtered_data.write.mode("append").format("hive").saveAsTable("filtered_employee")

❗ Challenges Faced

Issue	Description	Resolution
PythonSensor error	"Invalid arguments were passed" due to provide_context=True	✅ Removed provide_context, since it's not required in Airflow 2+
Hive metastore error	MetaException: Unable to create database path	✅ Changed config from gs:// to local path: .config("spark.sql.warehouse.dir", "/user/hive/warehouse")

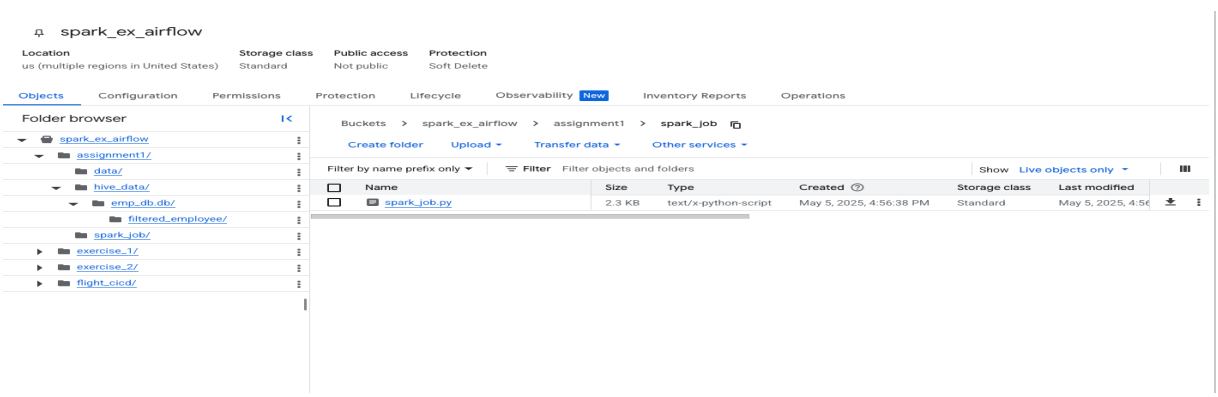
✅ Hive Output on GCS



Data is written as:

- emp_db.db/filtered_employee/part-00000-...
- _SUCCESS marker ensures completion

📜 Spark Script Location



Location:

- spark_ex_airflow/assignment1/spark_job/spark_job.py

✅ Final Result

All tasks completed successfully:

- 1. DAG status: ✅ Success, 2. Cluster was auto deleted, 3. Data ingested to Hive and saved to GCS as Parquet