Overview

This Airflow DAG, `etl_pipeline_docker`, orchestrates an end-to-end ETL process, which extracts, cleans, logs, and loads data into a PostgreSQL data warehouse (DWH) using PySpark, and subsequently runs dbt models for further data transformations.

The workflow utilizes Docker containers to execute tasks in a PySpark environment for the ETL process and a separate dbt container for dbt model execution.

DAG Details

- **DAG ID**: `etl_pipeline_docker`
- **Start Date**: 1 day ago (`days_ago(1)`)
- **Schedule Interval**: None (This DAG is triggered manually.)
- **Catchup**: False (No backfilling of previous executions.)

DAG Structure

The DAG consists of six tasks that follow a linear flow:

1. **Extract Data**: Runs a PySpark script to extract data from a source file.
2. **Clean Data**: Cleans the extracted data using PySpark.
3. **Logging Task**: Logs information about the data processing.
4. **Connect and Write to DWH**: Loads the cleaned data into the PostgreSQL data warehouse.
5. **dbt Run**: Executes dbt models for further transformations.
6. **dbt Test**: Tests the dbt models to ensure data integrity and quality.

Task Descriptions

1. Extract Data

- **Task ID**: `extract_data`
- **Operator**: `BashOperator`
- **Command**: Executes a PySpark script (`Write_CSV.py`) inside the `pyspark` Docker container.
- **Purpose**: To extract raw data and save it for further processing.

2. Clean Data

- **Task ID**: `clean_data`
- **Operator**: `BashOperator`
- **Command**: Executes a PySpark script (`Cleaning.py`) inside the `pyspark` Docker container.
- **Purpose**: To clean the raw data extracted in the previous step.

3. Logging Task

- **Task ID**: `logging_task`
- **Operator**: `BashOperator`
- **Command**: Executes a PySpark script (`Logging.py`) inside the `pyspark` Docker container.
- **Purpose**: Logs information related to data processing for future analysis and debugging.

4. Connect and Write to DWH

- **Task ID**: `connect_write_dwh`
- **Operator**: `BashOperator`
- **Command**: Executes a PySpark script (`Connect_Dwh.py`) to connect to the PostgreSQL database and load the cleaned data.
- **Additional Configuration**:
  - Uses local PySpark on all available cores (`--master local[*]`).
  - Specifies memory allocation for the driver and executors (`--driver-memory 8g`, `--executor-memory 8g`).
  - Loads the PostgreSQL driver (`--driver-class-path`).
  - Optimizes Spark shuffle partitions (`spark.sql.shuffle.partitions=100`).
- **Purpose**: To load the cleaned data into the PostgreSQL data warehouse.

5. dbt Run

- **Task ID**: `dbt_run`
- **Operator**: `BashOperator`
- **Command**: Executes `dbt run` inside the `dbt` Docker container.
- **Purpose**: To run the dbt models for data transformation and aggregation in the data warehouse.

6. dbt Test

- **Task ID**: `dbt_test`
- **Operator**: `BashOperator`
- **Command**: Executes `dbt test` inside the `dbt` Docker container.
- **Purpose**: To validate the transformations and ensure data quality in the data warehouse.

Scheduling

This DAG does **not** have a recurring schedule (`schedule_interval=None`) and is designed to be triggered manually as needed. The tasks do not use Airflow's `catchup` mechanism (`catchup=False`), meaning it does not attempt to backfill for missed execution dates.

Error Handling

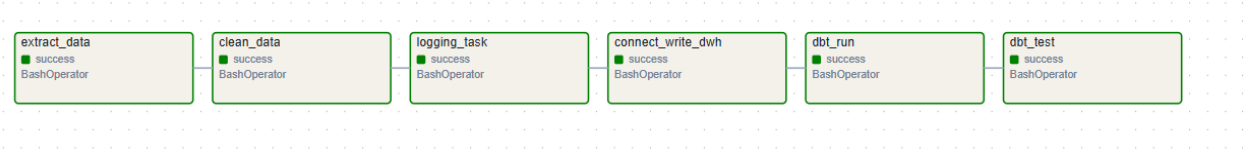
- **Retries**: Currently, the DAG does not specify custom retry logic, meaning tasks will use Airflow's default settings. You can configure retries in `default_args` if needed (e.g., `retries=3`).
- **Alerts**: You may also consider adding notifications (e.g., via email) in case of task failures using the `EmailOperator` or setting up failure callbacks in the `default_args`.
- **Logs**: All tasks in the DAG log information to the Airflow logging system. Additionally, the `logging_task` explicitly logs relevant information about the data processing steps.

Docker Configuration

Each task in the DAG executes a command inside a specific Docker container:

- The PySpark tasks (`extract_data`, `clean_data`, `logging_task`, and `connect_write_dwh`) run inside a `pyspark` container.
- The dbt tasks (`dbt_run` and `dbt_test`) are executed inside a `dbt` container

Diagram



Conclusion

This DAG performs an end-to-end ETL process using PySpark for data extraction, transformation, and loading into a PostgreSQL DWH. The final steps involve running and testing dbt models to ensure accurate data transformations. By leveraging Docker containers, the tasks are isolated and can be executed in their respective environments, ensuring reliability and modularity.