MLOPS A2

DAG | DVC | AIRFLOW

Shameer Abdullah | i200540

Comprehensive Report on Airflow DAG for ETL Process

1. Introduction:

This report provides an in-depth analysis of an Airflow Directed Acyclic Graph (DAG) designed to automate the Extract, Transform, and Load (ETL) process for data sourced from the BBC and Dawn websites. The DAG, named **extract_transform_load_data_dag**, orchestrates a series of tasks to extract raw data, transform it, and load the transformed data into a designated storage location.

2. Purpose:

The primary objective of the DAG is to automate the process of collecting news articles from the BBC and Dawn websites, perform data transformation tasks, and securely store the transformed data. By leveraging Airflow's workflow management capabilities, the DAG ensures reliability, scalability, and repeatability in executing the ETL pipeline.

3. Structure and Functionality:

3.1. Imports:

- The DAG imports necessary libraries and modules, including Airflow components for task orchestration, requests for making HTTP requests, BeautifulSoup for web scraping, csv for handling CSV files, os for file operations, and subprocess for executing shell commands.

3.2. Tasks:

- **Extract Data (extract_data_task):**
    - This task invokes the **extract_data()** function, which scrapes links and articles from both the BBC and Dawn websites.
    - It saves the extracted data to a CSV file named **data.csv**.
- **Transform Data (transform_data_task):**
    - This task calls the **transform_data()** function, which reads the raw data from **data.csv**, applies transformations (e.g., converting titles to uppercase), and saves the transformed data to a new CSV file named **transformed_data.csv**.
- **Load Data (load_data_task):**

- This task executes the **load_data()** function, which performs several actions:
    - Initializes Data Version Control (DVC) to manage versioning of data files.
    - Adds the transformed data file to DVC for tracking changes.
    - Configures a Google Drive remote for data storage.
    - Pushes the transformed data to Google Drive.
    - Adds all files to the Git index and commits changes with a customizable message.
    - Pushes changes to the designated GitHub repository.

3.3. Default Arguments:

- Default arguments for the DAG (**default_args**) are defined, including the owner, start date, email settings, and retries. These parameters ensure consistency and reliability in DAG execution.

4. Potential Improvements:

4.1. Error Handling:

- Implement robust error handling mechanisms to handle exceptions gracefully and provide informative error messages for troubleshooting.

4.2. Logging:

- Enhance logging functionality to capture detailed information about task execution, including success/failure status, timestamps, and any encountered errors.

4.3. Parameterization:

- Parameterize the DAG to make it more configurable, allowing dynamic specification of URLs, file paths, and other parameters.

4.4. Testing:

- Develop comprehensive unit tests to validate the functionality of individual tasks and ensure the reliability of the ETL pipeline.

4.5. Security:

- Implement secure handling of sensitive information such as API keys or credentials, following best practices for data protection.

4.6. Optimization:

- Optimize the web scraping process for improved performance and efficiency, considering strategies such as parallel processing and data caching.

4.7. Documentation:

- Provide extensive documentation for the DAG, tasks, and functions, including usage instructions, parameter descriptions, and workflow diagrams, to facilitate ease of understanding and maintenance.

5. Conclusion:

The Airflow DAG for the ETL process demonstrates a robust foundation for automating data extraction, transformation, and loading tasks from multiple sources. While the current implementation fulfills basic requirements, there are numerous opportunities for enhancement in error handling, logging, parameterization, testing, security, optimization, and documentation. By addressing these areas, the DAG can be further refined to ensure scalability, reliability, and maintainability in data pipeline operations.