Data Processing Pipeline for Food Delivery Analytics

Background

In the competitive landscape of food delivery services, platforms like Swiggy and Zomato thrive on data-driven decisions. Understanding customer preferences, optimizing delivery routes, and analyzing restaurant performance are crucial for their success. In this assignment, you will build a data processing pipeline mimicking the challenges faced by such platforms, focusing on data ingestion, validation, transformation, and analysis.

Objective

Your task is to develop an end-to-end data processing pipeline using AWS Airflow (Managed Workflows for Apache Airflow, MWAA) and AWS EMR that triggers a Spark job to process mock food delivery data. The processed data will then be ingested into an Amazon Redshift warehouse for further analysis.

Requirements

- **Data Storage & Mock Data:**
  - Input data will be stored in an S3 bucket (s3://your-bucket-name/food_delivery_data/). Assume the files arrive in CSV format with data resembling orders, customer interactions, and restaurant details.
  - Use this mock data generator to create your sample dataset. Create datasets that include fields such as order_id, customer_id, restaurant_id, order_time, delivery_time, customer_location, restaurant_location, order_value, and rating. Generate a file with at least 10,000 records.

- **Airflow Job:**
  - Implement an Airflow DAG that uses an S3KeySensor to detect the arrival of a new file in the specified S3 path.
  - Upon detecting a new file, the DAG should trigger a Spark job on AWS EMR to process this file.

- **Spark Job:**
  - The Spark job must read the CSV data from S3, perform data validation and transformation tasks including:
    - Validation: Ensure order_time and delivery_time are in the correct datetime format and that order_value is positive.
    - Transformation: Calculate the delivery duration and categorize order value into 'Low', 'Medium', and 'High' based on predefined thresholds.
  - After processing, the Spark job should write the transformed data into a staging area in Amazon Redshift.

- **CI/CD Deployment:**

    - Provide a CI/CD pipeline configuration (using tools like GitHub Actions, AWS CodeBuild) to automate the deployment of your Airflow DAG and Spark script. Ensure the pipeline includes steps for linting code, running tests, and deploying to the production environment.

Deliverables

- An Airflow DAG file that orchestrates the workflow.
- A Spark script (Scala or PySpark) for data processing.
- A CI/CD pipeline configuration file for automated deployment.
- A brief documentation covering:
    - The architecture of your solution with a diagram.
    - Instructions on how to deploy the Airflow DAG and Spark job.
    - Any assumptions made during the implementation.