Take-Home Assignment: Data & AI Engineering & CI/CD

Objective

Design and implement a scalable, automated Extract, Transform, Load (ETL) and Machine Learning (ML) pipeline using modern cloud and data engineering principles, including a robust CI/CD process.

Problem Description

Simulate a scheduled job that processes raw log data, cleanses and validates it (ETL), uses PySpark to train a simple machine learning model, and registers the model artifact, all while being managed by an automated deployment pipeline.

We expect the raw data to be simulated using an AI tool of your choice.

Core Requirements

1. ETL Pipeline Implementation

- **Extract**
  Simulate fetching a compressed file (e.g., Gzip-ed CSV or JSON) of raw log data from a cloud storage bucket (e.g., AWS S3, GCP Cloud Storage or ADLS2).

- **Transform & Validate**
  Use **PySpark** or **AWS Glue / Databricks** to process the data.
  The script must:
  - Cleanse the data (e.g., fill nulls, standardize strings etc..).
  - Validate the schema and data quality (e.g., reject rows where a user_id is not present or an event_time is not a valid timestamp).
  - Introduce a new derived column, such as processing_date or event_category, based on existing fields.
  - Implement lineage capabilities: you have the freedom to use any existing framework or simple lineage function of your choice.

- **Load (data lake)**
  Write the final, structured, and validated data into a partitioned data lake format (e.g., Parquet/Delta Lake) back into the cloud storage bucket.

- **ML Pipeline Implementation (Spark ML)**
  - Using the cleaned data from the ETL step
    Define a simple feature vector for a prediction task. This should leverage PySpark's DataFrame-based tools such as VectorAssembler and other feature transformers from the pyspark.ml.feature module.
  - Model Training
    Train a simple model (e.g., Logistic Regression or Decision Tree) using an Estimator from the pyspark.ml.classification or pyspark.ml.regression module.
  - ML Pipeline Construction
    Candidates should demonstrate the ability to construct a complete ML Pipeline object that chains together feature transformations and the final model training step.
  - Model Tracking & Registration
    Save the trained *Pipeline Model* artifact (e.g., to S3/Cloud Storage) and register the model metadata (e.g., path, metrics, version) to a lightweight model registry (e.g., MLflow).

- Feature Engineering: Using the cleaned data from the ETL step, define a simple feature vector for a prediction task (e.g., predicting if a user will generate a high-value event). This should leverage PySpark's vector assembler.

- Model Training: Train a simple classification or regression model (e.g., Logistic Regression or Decision Tree) using Spark MLlib.
- Model Tracking & Registration: Save the trained model artifact (e.g., to S3/Cloud Storage) and register the model metadata (e.g., path, metrics, version) to a lightweight model registry (e.g., a simple DynamoDB/Firestore table or an open-source tool like MLflow).

- **Orchestration & Scheduling**
  - The entire workflow (ETL -> ML Training -> Model Registration) must be configured to run as a single, scheduled job.
  - Use an orchestration tool like Airflow (AWS MWAA/GCP Composer), AWS Step Functions, or Google Cloud Workflows to manage the pipeline flow.

2. Infrastructure & Environment

- The infrastructure must be deployed on either **AWS, GCP or Azure**.
- Provide the details of the network and the compute environment for the ETL (PySpark/Glue), together with its rationale.

3. CI/CD and Testing

- Implement a CI/CD pipeline (using GitHub Actions, AWS CodePipeline, GCP Cloud Build or equivalent) for the ETL code and its deployment.

- Continuous Integration (CI)
  The pipeline should automatically run when code is pushed to a feature branch and must include:
  - Unit Tests:
    Implement Python pytest for the transformation logic functions.

  - Data Validation Check:
    A simple script to ensure the final output schema is correct.

- Continuous Deployment (CD)
  Merging code to the main branch should trigger the deployment of the updated PySpark script and the orchestration configuration (e.g., the new Airflow DAG or Step Function definition) to the cloud environment.

Expected Deliverables

1. A link to a Git repository containing all the code, including:
   - The PySpark ETL script.
   - The unit tests.
   - The orchestration definition file (e.g., Airflow DAG or Step Function JSON).
   - Sample raw data for testing.

2. A README.md file with
   - A high-level overview of the architecture and data quality checks.
   - Detailed instructions on how to set up the cloud environment and run the CI/CD pipeline.
   - Instructions on how to view the transformed, and the registered model artifact.

3. **Bonus** (Infrastructure as Code - IaC)
   A Terraform template to deploy all required cloud resources (storage buckets, compute environment like AWS Glue/EMR/Databricks cluster and the CI/CD pipeline resources).