NAYA College

Cloud Big Data Engineer

Final Project

Contents

Final Project: Multi-Purpose Data Pipeline System

Overview

As a Data Engineer, your task is to create a versatile data pipeline system. This system will acquire data from real API(s) in either real-time or batch mode. Once collected, the data will undergo filtering, cleansing, verification, and enrichment. The goal is to enable analytical operations for organizational and business purposes, transforming raw data into actionable insights.

Objective

Design and implement a data pipeline using Docker containers or AWS. Through this project, students will gain hands-on experience in setting up a scalable, efficient, and flexible data storage solution.

Project Main Goals

1. **Solution Architecture Design**: Develop a comprehensive solution architecture that aligns with project requirements.
2. **Data Pipelines Creation**: Construct data pipelines to support the project's needs.
3. **End-to-End Solution**: Apply knowledge from various fields (infrastructure/logical) to create a complete end-to-end solution.
4. **Pair Project**: Collaborate with a partner to work on the project.

Technologies to Explore

1. **Cloud Platform:** AWS, Google Cloud Platform (GCP), Microsoft Azure, or local development environment.
2. **API:** The source APIs can be chosen by the student as long as it serves the project purposes (will be explained by the instructor).
3. **Database:** Students can choose from various databases, including Amazon Redshift, Google BigQuery, Snowflake, or any other cloud-based data warehouse. Alternatively, they can set up a local database using Docker containers.
4. **Storage:** Students can choose from various storages like AWS S3, AWs Glacier , Minio, HDFS.
5. **Code** : Use Python or PySpark in PyCharm.
6. **Scheduler :** Choose between Airflow and Crontab.

Project Steps

1. **Project Subject and Architecture Selection**:
    - Students decide on the project subject (e.g., sales analytics, user behavior tracking) and choose the architecture of the pipeline.
2. **API Data Ingestion**:
    - Students find an API (e.g., from RapidAPI) that provides relevant data for their project.
    - They use Python to fetch data from the API.
    - Example: Fetch real-time stock market data using the Alpha Vantage API.
3. **Pipeline (ETL - Extract, Transform, Load)**:
    - Students write ETL scripts (Python, Spark) to transform raw data.
    - They load transformed data into their database.
    - Example: Transform JSON data from the API into structured tables.
4. **Real-Time Data Processing with Kafka/Kinesis**:
    - Students set up Apache Kafka or Amazon Kinesis for real-time data streaming.
    - They ingest data from the API into Kafka/Kinesis topics.
    - Example: Use Kafka to stream user activity.
5. **Data Warehouse**:
    - Students chose a database (e.g., PostgreSQL, MySQL, or Amazon Redshift).
    - They set up the necessary environment variables, volumes, and network configurations.
    - Example: Run a Redshift container with a custom schema.
6. **Data Storage (Partitioned):**
    - Implement data storage using Amazon S3 or Minio.
    - Partition the data to enhance query performance and manage large datasets efficiently.
    - Example: Store data in S3 buckets with partition keys based on date or other relevant attributes.

7. **Scheduled Data Processing with Airflow**:
    - Students create an Airflow DAG (Directed Acyclic Graph) for scheduled data processing.
    - They schedule ETL jobs to run periodically (e.g., hourly or daily).
    - Example: Use Airflow to load daily stock prices into the data warehouse.
8. **Pipeline for Logs and Analytics** (Bonus):
    - Students explore ELK (Elasticsearch, Logstash, Kibana) for log analysis.
    - They ingest logs from Docker containers and visualize insights.
    - Example: Set up Logstash to parse Docker logs and create Kibana dashboards.
9. **Performance Tuning** (Bonus):
    - Students optimize query performance by creating indexes, analyzing execution plans, and choosing appropriate data types.
    - Example: Optimize SQL queries for faster response times.
10. **Security and Access Control** (Bonus):
    - Students set up authentication and authorization for their database.
    - They define roles and permissions for users.
    - Example: Create IAM roles for Amazon Redshift with limited access.

Project Success Criteria

- The solution should address approximately 80% of the project requirements (including solution design, code, etc.).
- The project must be submitted according to the defined deadline.
- Students should demonstrate the project solution during the course's last project session.
- The solution will be examined on three levels:
    - Solution implementation percentage.
    - Solution design (architecture diagram document).
    - Python code quality.
- The final solution will be presented in class in front of the course instructors.