Real-Time Data Streaming & Batch ETL Pipeline with AWS

Hybrid Architecture for Scalable Analytics

Team 01 Section 2

Team Members

STUTI BIMALI
Team Leader
Section 1

PRIYANKA SINGH
Data Scientist
Section 2

LAKSHMAN RAJITH
Data Modeling
Section 2

Team GitHub Account: https://github.com/orgs/Team01DE/

Challenges

Data That We Will Need

1. **Real-Time Streaming Data**
   - Examples: E-commerce transactions (order ID, amount, timestamp), IoT sensor data (device ID, temperature, timestamp), or financial logs (transaction ID, user ID, amount).
   - Format: JSON or CSV, reflecting high-velocity event streams.
2. **Metadata for Enrichment**
   - Examples: Customer profiles (e.g., preferences for recommendations), fraud rules (e.g., transaction thresholds), or geolocation data.
3. **Historical Batch Data**
   - Aggregated data for trend analysis (e.g., daily sales totals, average sensor readings by hour).

Where You Will Get the Data

1. **Real-Time Streaming Data**

- Simulated using a Python script or API as a producer sending data to AWS Kinesis Data Streams. For realism, mimic e-commerce checkouts, IoT sensor pings, or financial transactions.

2. **Metadata for Enrichment**
    - Generated within AWS Lambda (e.g., timestamps) or sourced from a predefined lookup table in Amazon S3 or DynamoDB (e.g., customer profiles or fraud rules).

3. **Historical Batch Data**
    - Derived from real-time data stored in S3 after Lambda processing, then transformed by AWS Glue into a structured format for analytics.

Overview of the Pipeline (Using CRISP-DM Methodology)

1. **Business Understanding**
    - **Objective**: Enable real-time decision-making (e.g., personalized offers, fraud alerts) and historical trend analysis (e.g., sales patterns) for e-commerce, IoT, or financial organizations.
    - **Business Problems Addressed**:
        - *Delayed Insights*: Provide sub-second insights from streaming data.
        - *Data Volume & Velocity*: Handle large, high-speed data efficiently.
        - *Inefficient ETL & Storage*: Optimize processing with a hybrid pipeline.
        - *Poor Integration & Scalability*: Unify real-time and batch analytics.
    - **Success Criteria**: Real-time latency <1 second, accurate batch reports available daily, scalable to 10x data volume.

2. **Data Understanding**
    - Analyze incoming data characteristics:
        - Volume: E.g., 100 transactions/second.
        - Velocity: Continuous event streams.
        - Variety: JSON logs with fields like order ID, amount, timestamp.
    - Investigate metadata needs (e.g., customer IDs for personalization) and output requirements (e.g., Parquet tables for querying).

3. **Data Preparation**
    - **Real-Time**:

        - Use Lambda to filter invalid data (e.g., missing transaction IDs) and enrich with metadata (e.g., customer preferences or fraud flags).
        - Store processed data in S3 in near real-time.
    - **Batch**:
        - Use Glue to aggregate data (e.g., daily sales totals) and convert to Parquet for efficient storage and querying.
4. **Modeling**
    - **Real-Time**: Lambda functions to:
        - Filter incomplete records.
        - Enrich data (e.g., flag transactions >$1,000 as potential fraud).
    - **Batch**: Glue ETL scripts to:
        - Aggregate data (e.g., sum transactions by customer).
        - Transform JSON to structured Parquet.
    - Tools: Python for Lambda/Glue, SQL for transformations.
5. **Evaluation**
    - **Real-Time**: Test latency (e.g., <1 second processing) and accuracy (e.g., 99% of fraud flags correct).
    - **Batch**: Validate aggregates (e.g., daily totals match raw data) and query performance (e.g., <5 seconds for a report).
    - Simulate data spikes to ensure scalability.
6. **Deployment**
    - Deploy Kinesis Streams, Lambda functions, S3 buckets, and Glue jobs on AWS.
    - Schedule Glue jobs (e.g., daily at 1 AM) for batch processing.
    - Integrate with a data warehouse (e.g., Redshift) for reporting.
    - Monitor with AWS CloudWatch for errors or bottlenecks.