

Real-Time Data Ingestion and NoSQL Warehousing on AWS

Project Title

Design and Implementation of a Real-Time Data Processing Pipeline Using AWS Kinesis, Lambda, and DynamoDB

2. Abstract

In modern enterprises, especially in the telecommunications domain, real-time data plays a crucial role in monitoring network performance and optimizing infrastructure. This project demonstrates the design and implementation of a scalable, serverless, and production-ready real-time data ingestion pipeline using Amazon Web Services (AWS). The solution captures streaming data using Amazon Kinesis Data Streams, processes it using AWS Lambda, and stores it in Amazon DynamoDB for future analytics and optimization.

3. Objectives

- To design a real-time data ingestion pipeline on AWS
 - To process streaming data using serverless compute
 - To store structured streaming data in a NoSQL database
 - To understand enterprise-level scalability, security, and monitoring
 - To perform data validation using DynamoDB Scan operations
-

4. Problem Statement

TELEMAX is a telecommunications company expanding into underserved markets and generating large volumes of real-time network telemetry data. The company requires a scalable and reliable architecture to ingest, process, and store this data for continuous network optimization. Traditional batch processing systems are insufficient due to latency constraints. Hence, a real-time, cloud-native solution is required.

5. Real-World Use Case

- Telecom network telemetry ingestion
- IoT device signal monitoring
- Real-time system health analytics
- Network topology optimization

6. Proposed AWS Architecture

The following AWS architecture is designed to address TELEMAT's requirements for real-time, scalable, and reliable processing of network telemetry data. The architecture leverages fully managed AWS services to ensure low-latency data ingestion, processing, storage, and analytics.

1. **Data Producer (CLI / Application):** Network telemetry data is generated by various sources, such as telecom network components or IoT devices, and sent to the data ingestion layer through a command-line interface (CLI) or custom applications.
2. **Amazon Kinesis Data Streams:** The data is ingested in real time using Amazon Kinesis Data Streams, which provides a robust, highly available, and scalable stream processing platform. Kinesis ensures that the streaming data is durably captured and made available for downstream processing with minimal latency.
3. **AWS Lambda (Stream Processor):** AWS Lambda functions are triggered automatically by new records in the Kinesis stream. These serverless compute resources process, transform, and validate the streaming data without the need for server management, ensuring scalability and cost-efficiency.
4. **Amazon DynamoDB (NoSQL Storage):** Processed and validated data is then stored in Amazon DynamoDB. This NoSQL database provides high throughput and low latency, making it suitable for storing structured streaming data and supporting fast queries for analytics and monitoring.
5. **Analytics & Monitoring:** Data stored in DynamoDB can be accessed by analytics and monitoring tools to provide insights into network performance, identify anomalies, and support continuous optimisation. Integration with AWS services such as Amazon CloudWatch, AWS Glue, and Amazon QuickSight enables advanced monitoring and visualisation.

This architecture ensures end-to-end data flow, from ingestion to actionable analytics, with enterprise-level scalability, security, and monitoring. It enables TELEMAT to efficiently manage real-time telemetry data and rapidly respond to network events.

7. AWS Services Used

7.1 Amazon Kinesis Data Streams

- Captures real-time streaming data at scale
- Supports high throughput and low latency
- Enables data replay through retention settings

7.2 AWS Lambda

- Serverless compute service
- Processes streaming data automatically
- Scales based on incoming event volume

7.3 Amazon DynamoDB

- Fully managed NoSQL database
- Low-latency data access
- Horizontally scalable for enterprise workloads

7.4 AWS CloudShell

- Used to publish test records into Kinesis using AWS CLI

8. Implementation Details

8.1 Kinesis Data Stream Creation

- Stream Name: telemex-realtime-stream
- Capacity Mode: On-Demand
- Encryption: Enabled using AWS KMS
- Retention Period: Configurable for replay

8.2 DynamoDB Table Design

- Table Name: TelemaxRealtimeData
 - Partition Key: id (String)
 - Data Stored:
 - Device ID
 - Signal Strength
 - Timestamp
-

8.3 Lambda Function Configuration

- Runtime: Python 3.x
 - Trigger: Kinesis Data Stream
 - Batch Processing Enabled
 - Error handling using try-except blocks
 - Logs enabled via Amazon CloudWatch
-

8.4 Data Ingestion

Data is published to Kinesis using AWS CLI from CloudShell with Base64 encoding handled automatically.

9. Data Validation

- DynamoDB **Scan operation** is used to verify data ingestion
- CloudWatch logs confirm Lambda execution
- End-to-end pipeline validation ensures data consistency

10. Security Implementation

- IAM roles with controlled permissions
- Lambda execution role restricted to:
 - Read from Kinesis
 - Write to DynamoDB
- Encryption enabled at rest and in transit

11. Monitoring and Logging

- CloudWatch metrics:
 - Lambda Invocations
 - Errors
 - Throttles
- Structured logging for easier debugging
- Can be extended with alarms and notifications

12. Industry Relevance

This architecture is widely used in:

- Telecommunications
- IoT platforms
- Financial transaction monitoring
- Real-time analytics systems
- Event-driven microservices

13. Conclusion

The project successfully demonstrates a real-time, serverless data ingestion pipeline using AWS native services. The solution is scalable, fault-tolerant, secure, and suitable for enterprise-level workloads. It highlights best practices in cloud architecture, real-time data processing, and NoSQL data modelling.

15. References

- AWS Kinesis Data Streams Documentation
- AWS Lambda Developer Guide
- Amazon DynamoDB Developer Guide



