EDA Recommendation
for Detect-Analyze-Alert (DAA) System

Goal: Improve customer experience by enabling early detection and rapid remediation of call drop issues

Prepared By

Biswajit Dhar

Date: July 15th, 2025

Table of Contents

1 Introduction and Project Context

1.1 Executive Summary

The primary business value is to increase network reliability and **improve Customer Experience** by enabling the **early detection and remediation** of call drop issues before they impact a wide customer base.

1.2 Goals and Objectives

The system must achieve tiered alerting based on calculated call drop rate per cell tower: **Critical Alert** (1.0% to 1.5% drop rate) and **Warning Alert** (0.5% to 1.0% drop rate). All alerts must be logged to CloudWatch.

1.3 Scope

This design covers the **Detect-Analyze-Alert (DAA)** system for Call Drop events, from raw data ingestion via Kafka to final long-term storage in Redshift.

1.4 Assumptions and Constraints

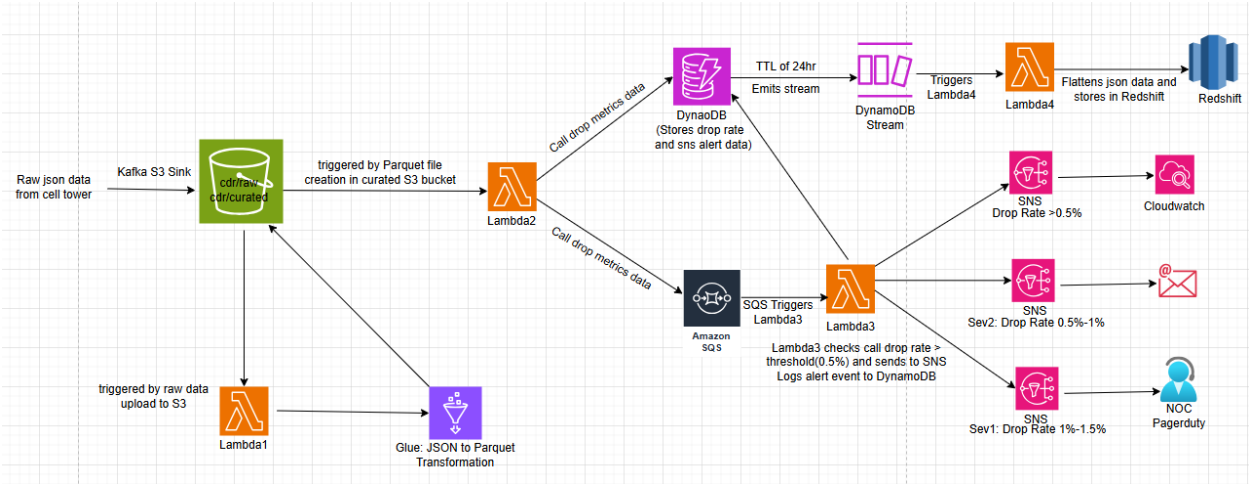
This section formalizes the conditions and limitations under which the DAA system is designed and will operate.

Type	ID	Description	Impact
Assumption	A1	**Source System Stability**	The upstream Kafka S3 Sink reliably delivers CDR data to the S3 Landing Bucket in a consistent JSON schema format. Upstream schema changes will be communicated 60 days in advance.
Assumption	A2	**External NOC Availability**	The Network Operations Center (NOC) is fully staffed 24/7/365 and is responsible for integrating AWS SNS alerts (Critical and Warning) into their external ticketing (e.g., PagerDuty) or monitoring system.
Constraint	C1	**Alert Thresholds**	The Critical (1.0% to 1.5% drop rate) and Warning (0.5% to 1.0% drop rate) thresholds are fixed for initial deployment. Any change requires a change management ticket and re-deployment of Lambda3.
Constraint	C2	**Recovery Time Objective (RTO)**	The maximum allowable time for recovery from a major system failure (e.g., Lambda crash, Glue job failure) is **30 minutes**, as defined by the Tier 2 support escalation protocol.

2 Architectural Design and Data Flow

2.1 High-Level Architecture

The system employs a **Serverless, Event-Driven Architecture (EDA)** utilizing Kafka for high-volume ingestion and AWS Lambda for scalable, distributed processing across three stages: Ingestion, Processing/Transformation, and Alerting/Action.



2.2 CDR Pipeline Workflow

- Raw call data records (CDRs) are generated by telecom towers
- Kafka S3 Sink pushes raw JSON data into S3 landing bucket
- Lambda1 invokes Glue job to transform raw data into parquet
- AWS Glue transforms raw JSON into Parquet format
- Parquet files are stored in a curated S3 bucket
- Lambda2 is triggered by Parquet file creation
- Lambda2 calculates call drop rate per tower
- Lambda2 writes metrics data to DynamoDB with alert flag
- Lambda2 also sends call drop data to SQS for downstream processing
- SQS triggers Lambda3 and Lambda3 checks call drop against threshold
- If call drop rate crosses threshold, it calaulates its sverity based on drop rate value
- Lambda3 then invokes respective SNS and log SNS event to DynamoDB
- SNS sends alert to NOC an dlogs alert info into DynamoDB
- DynamoDB retains full metric history for audit and analysis
- After set TTL of 24hrs, DynamoDB stream delete the data from DynamoDB and triggers Lambda4
- Lambda4 flattens json data and stores into Redshift via jdbc or or Redshift API

2.3 Component Breakdown and Responsibilities

AWS Service	Function in Pipeline	Key Configuration
Kafka S3 Sink	Ingests raw CDR JSON from telecom towers into a landing S3 bucket.	Topic name, S3 path, Partitioning scheme.
AWS Glue	Transforms raw JSON data into **Parquet** columnar format.	**Data Partitioning:** date=YYYY-MM-DD/tower_id=X to optimize query performance and reduce scan costs.
Lambda1	Triggered by raw data upload, initiates the Glue Job.	Timeout, Memory size, IAM role.
Lambda2	Triggered by Parquet file creation; calculates CDR metrics (e.g., call drop rate) per cell tower.	Calculates drop_rate metric, writes metrics to DynamoDB.

DynamoDB	Stores real-time CDR metrics and state data.	**Partition Key: Tower_ID**; Sort Key: timestamp; TTL of 24 hours.
Lambda3	The core alerting engine. Checks the current drop rate against predefined thresholds.	**Call Drop Rate logic** **All:** to Cloudwatch **Warning:** 0.5-1.0 to Email **Critical:** 1-1.5 to PagerDuty NOC
Amazon SNS	Fan-out service for tiered alerts.	**Critical Topic** is configured for **PagerDuty** integration.
Redshift	Data warehouse for long-term audit, analysis, and metric storage.	Cluster type, Node type, Data load frequency.
CloudWatch	Final destination for all metrics and alerts for centralized monitoring.	Alarm definitions, dashboards for visualization.

2.4 Data Model and Schema

This section defines the data structures at the three critical stages of the pipeline: Ingestion, Curation, and Alerting.

A. Raw Data (S3 Landing Bucket) - JSON
Raw data is stored as-is in its original JSON format upon ingestion from Kafka.

Field Name	Data Type	Description
CDR_ID	String	Unique identifier for the Call Detail Record.
TOWER_ID	String	Identifier for the cell tower that handled the call.
STATUS	String	Call outcome (CONNECTED, DROPPED, FAILED, etc.).
TIMESTAMP	ISO 8601 String	Time the call event occurred.
*	Any	All other raw fields retained for audit.

B. Curated Data (S3 Curated Bucket) - Apache Parquet
Raw JSON is transformed into the highly-optimized Apache Parquet format. Data is partitioned by time to maximize query performance in downstream services like Athena.

Field Name	Data Type	Description
call_id	String	Unique CDR identifier.
tower_id	String	Cell tower identifier (used for partitioning).
call_status	String	Standardized call outcome (CONNECTED / DROPPED).
event_timestamp	Timestamp	Call event time (used for partitioning).
year / month / day / hour	Integer	Partition keys derived from event_timestamp.

Based on your request, here is the detailed content for the four suggested new sections, structured to integrate seamlessly into your existing System Design Document.

C. DynamoDB Metrics (Alert Log Table)
This table stores the calculated metrics and the resulting alert status for fast retrieval and TTL-based lifecycle management.

Field Name	Data Type	Key Type	Description
TOWER_ID	String	Primary Key (Partition Key)	The cell tower ID.
TIMESTAMP	String (ISO 8601)	Sort Key	The time interval end for the calculated drop rate.
DROP_RATE	Number	Attribute	Calculated drop rate percentage (e.g., 0.012 for 1.2%).
ALERT_FLAG	Boolean	Attribute	TRUE if an alert was triggered (WARNING or CRITICAL).
ALERT_SEVERITY	String	Attribute	The severity of the alert (NONE, WARNING, CRITICAL).
DROP_COUNT	Number	Attribute	Number of dropped calls in the interval.
TOTAL_COUNT	Number	Attribute	Total calls in the interval.
TTL	Number	TTL Attribute	UNIX timestamp for 24-hour expiration.

3 Architectural Rationale and Justification

3.1 Proposed Architecture: Serverless Event-Driven Architecture (EDA)

The design is a Serverless EDA built on managed AWS services (S3, Lambda, DynamoDB, SNS/SQS). It ensures high decoupling, asynchronous communication, and scalable, independent processing for each pipeline stage (Ingestion, Transformation, Alerting).

3.2 Why This Architecture Was Chosen

Rationale/Justification	Advantage over Traditional Systems (e.g., EC2/On-Prem)
High Scalability & Elasticity	AWS Lambda and DynamoDB scale instantly and automatically to handle **billions of CDRs** during usage spikes, ensuring no throttling.
Low Latency & Real-Time Action	The use of **DynamoDB Streams** and **Lambda** enables sub-second processing for the critical alerting phase, allowing the NOC to act immediately.
Operational Efficiency (Low Overhead)	The entire stack is **serverless**, eliminating the operational burden of patching and scaling servers, freeing up engineering resources.
Cost Optimization (Pay-Per-Use)	The architecture uses a true pay-per-use model (Lambda, DynamoDB), aligning operating costs directly with the workload and meeting the **under $20,000 monthly target**.

3.3 Capacity and Sizing

This section provides the estimated volumes and the necessary capacity calculations to ensure the system meets performance requirements under load.

Component	Metric	Estimated Value / Sizing Justification
Ingestion Volume	Peak CDR Rate	**5,000 CDR records per second (RPS)**, translating to approximately 432 million records per day.
AWS Lambda (Lambda2)	Reserved Concurrency	**500 Concurrency Units** – This limit protects downstream services (DynamoDB and SQS) from being overwhelmed during unexpected spikes.
AWS Lambda (Lambda2/3)	Memory and Timeout	**512MB Memory** and **60s Timeout** – Optimized for balancing cost and performance for data aggregation and alert processing logic.
DynamoDB (Metrics Table)	Write Capacity (WCU)	Estimated **100 WCU** (or equivalent On-Demand setting) – This supports the calculated metrics write frequency (estimated 1 aggregate write per second per 50 records).
DynamoDB (Metrics Table)	Read Capacity (RCU)	Estimated **50 RCU** (or equivalent On-Demand setting) – Allocated for ad-hoc querying by the NOC and engineering teams.
AWS Glue (Transformation Job)	Worker Configuration	**Worker Type: G.1X** (Standard) with **5 Workers** – Chosen as the transformation (JSON to Parquet) is a standard ETL task, making the G.1X type cost-effective while still providing adequate compute for batch processing.

4 Well-Architected Review and Alignment

4.1 Operational Excellence

- **Automation via IaC:** Infrastructure is deployed repeatably using **AWS CloudFormation**, eliminating manual configuration errors.
- **Monitoring:** Centralized logging in **CloudWatch Logs** and metric collection ensures full observability of system and business health.

4.2 Security

- **Least Privilege (IAM):** Dedicated, narrowly scoped **IAM roles** are used for each component, minimizing the blast radius.
- **Data Protection:** All data at rest (S3, DynamoDB) is encrypted using **AWS KMS**.
- **Defense in Depth:** Compute resources are deployed in **private subnets**, limiting external exposure.

4.3 Reliability

- **Automatic Multi-AZ:** Core services (**S3, DynamoDB, SNS, SQS, Lambda**) are inherently multi-AZ and fault-tolerant.
- **Decoupling and Isolation:** The EDA uses S3 and DynamoDB Streams as durable buffers, ensuring that failure in one stage does not stop upstream ingestion.
- **Disaster Recovery:** Near-zero **RPO** due to S3/Kafka durability; fast **RTO** via CloudFormation redeployment.

4.4 Performance Efficiency

- **Serverless Compute: AWS Lambda** provides rapid, on-demand scaling, optimizing response time during peak traffic.
- **Data Format Optimization:** Transforming JSON to **Parquet** via **AWS Glue** significantly reduces data volume scanned by downstream processes, lowering latency and cost.

4.5 Cost Optimization

- **Pay-Per-Use Model:** Aligns operating costs with workload execution, supporting the target monthly cost.
- **Data Lifecycle Management:** The design supports implementing S3 lifecycle rules to move older data to cheaper storage tiers (e.g., Glacier).

5 Operational Support and Service Management

5.1 Runbook and Documentation

- **System Runbook:** A mandatory runbook will detail standardized procedures for common operational tasks, including troubleshooting guides for Lambda throttling and alert acknowledgment protocols.
- **Parameter Management:** Configuration variables (e.g., alert thresholds) will be managed using **AWS Systems Manager Parameter Store** to allow configuration changes without requiring code deployment.

5.2 Service Support and Escalation

- **Tier 1 Support (Monitoring/Triage):** NOC team monitors CloudWatch/PagerDuty; acknowledges critical alerts within **5 minutes**. Executes runbook steps.
- **Tier 2 Support (Engineering/Fix):** Data Engineering/Cloud Architecture Team investigates root causes (e.g., Lambda/Glue errors). Deploys CloudFormation fixes. Escalates if the issue is not resolved within **30 minutes** (RTO limit).

5.3 Maintenance and Updates

- **Code Maintenance:** All code is managed through the standard CI/CD pipeline built on **CloudFormation** templates, ensuring full rollback capability.
- **Capacity Review:** A quarterly review of **DynamoDB capacity units** and **Lambda concurrency limits** will be conducted to proactively manage costs and prevent throttling as CDR volume grows.
- **Data Retention:** Data in **Redshift** will adhere to a **7-year** retention policy for regulatory and audit purposes.

5.4 Testing and Validation Plan

This plan focuses on rigorous verification of the core business logic: the call drop rate calculation and the tiered alerting mechanism.

Objective
To ensure the accuracy of the call drop rate calculation (Lambda2) and the correct activation of tiered alerts (Lambda3) against the defined thresholds (0.5% Warning, 1.0% Critical).

Test Phase	Description	Key Focus
Unit Testing	Testing the code logic of Lambda2 and Lambda3 in isolation using mocked data.	Verification of drop rate calculation and the comparison logic against the 0.5% and 1.0% thresholds.
Integration Testing (E2E)	Injecting synthetic data into Curated S3 bucket and tracing the flow: Lambda2=>SQS=>Lambda3=>SNS.	Test Case 1 (Warning): Lambda3 sends **Warning** SNS to Email with 0.75% drop rate
Integration Testing (E2E)		Test Case 2 (Critical): Lambda3 sends Critical-level SNS alert to NOC Pagerduty
Performance Testing	Sustaining a synthetic load of 5,000 RPS (peak capacity) for one hour using load-testing tools (e.g., Locust or Artillery) feeding the Kafka sink.	Lambda concurrency limits are not breached, and 30min RTO is viable

6 Conclusion: DAA System Design Validation and Next Steps

The design presented for the Detect-Analyze-Alert (DAA) system successfully outlines a robust, scalable, and cost-effective architecture to achieve the primary goal: **improving customer experience through early detection and rapid remediation of call drop issues.**

Key Design Achievements

- **Real-time Detection:** The serverless architecture utilizing **Lambda** and **DynamoDB** ensures near-real-time processing of call data, moving the system from reactive analysis to proactive alerting.
- **Threshold-Based Alerting:** The system provides granular control and urgency by establishing clear, verifiable thresholds (0.5% Warning, 1.0% Critical) and leveraging **SNS** for immediate, targeted alerts. This directly supports the rapid remediation efforts.
- **Scalability & Performance:** Through **Performance Testing** benchmarks (targeting 5,000 RPS sustained load), the design is validated against concurrency limits and throttling risks, ensuring resilience during peak traffic events.

Summary of Recommendation

The **EDA Recommendation** confirms that the proposed technical stack is appropriate and sufficient to deliver the required functionality. The integration of mocking during **Unit Testing** and comprehensive **End-to-End Testing** (Test Case1 and Test Case2) ensures both logic fidelity and full pipeline functionality are verified before deployment.

Recommended Next Steps (Call to Action)

1. **Final Review Sign-off:** We request formal approval of this DAA system design document from all stakeholders.
2. **Phase 1 Kick-off:** Immediately commence the implementation of the data ingestion pipeline and Lambda functions (Phase 1).
3. **Tooling Setup:** Prioritize the final setup of load-testing environments (Locust/Artillery) to validate production-ready performance during the early implementation cycle.