Disaster Recovery Architecture Using Kafka/SQS/Kinesis (DR Ledger Only)

Executive Summary

Modern cloud-native applications operating on AWS need strong disaster recovery (DR) strategies, especially when using globally replicated databases like Aurora Global. While Aurora provides cross-region replication, it does not eliminate data loss during regional failover due to replication lag (typically 2-5 seconds). To mitigate this gap, we introduce a side-channel DR recovery ledger using Kafka, driven by a Debezium-based outbox pattern, designed solely to detect and optionally recover lost transactions after an unplanned regional disaster. This document evaluates the approach in detail, compares queue-based solutions, and recommends a scalable, low-intrusion strategy for resilient but decoupled disaster recovery.

High-Level Architecture Overview

This architecture assumes:

- Aurora Global is the system of record (SOR)
- A dedicated outbox table captures every insert
- Debezium streams the outbox to Kafka asynchronously
- Kafka is replicated across regions
- Kafka is used only for recovery, not normal processing

The flow ensures that every committed transaction is also recorded in Kafka - acting as a DR ledger.

Purpose of Queue in This Design

Queues like Kafka, SQS, or Kinesis are not used for business logic but as a secondary backup path in case of failure. The goal is not to prevent all failures but to ensure visibility and traceability of what might be lost and provide tools to restore it.

Aurora Global Only vs. Aurora + Outbox + Kafka for DR

This comparison shows how relying solely on Aurora Global compares with using Kafka + outbox to reduce data loss during failover.

DR Scenarios: What Happens on Failure

Disaster Recovery Architecture Using Kafka/SQS/Kinesis (DR Ledger Only)

To understand this solution's value, consider several failure scenarios:

Debezium-Based Kafka vs Direct Kafka Write from App

There are two ways to populate Kafka for DR:

1. Debezium-based approach, where the app only writes to DB.

2. Direct Kafka writes from the application in parallel to DB writes.

Reconciliation and Replay

After a DR event:

1. Kafka is queried for a time window of recent inserts.

2. Aurora outbox is queried in DR region after failover.

3. Missing entries are identified.

4. Idempotent replay logic can reinsert lost transactions into the DB.

This ensures data loss is traceable and recoverable, without interfering with the primary app logic.

Quantifying Benefits

Let's compare a real scenario:

Application does 100 TPS inserts.

Conclusion

In DR-critical environments, relying solely on Aurora Global can lead to silent, unrecoverable data loss due to replication lag. By adding a dedicated disaster recovery ledger (Kafka), populated via the Debezium outbox pattern, teams can detect and recover lost inserts with minimal impact on application design.

Final Takeaways:

- Aurora is the SOR, Kafka is a DR journal
- Kafka is only used after failure, not during normal ops
- Debezium-based streaming ensures only committed data is logged
- Replay is safe, consistent, and observable

Disaster Recovery Architecture Using Kafka/SQS/Kinesis (DR Ledger Only)

- This setup provides auditable traceability and resilience at modest cost