

CDC Reliability Boundaries

PostgreSQL source continuity is not the same as downstream exactly-once correctness

A CDC event crosses four independent durability boundaries. Each boundary can advance at a different time, so production correctness depends on how the states are reconciled.

1. PostgreSQL source

The logical replication slot retains source history and exposes positions such as `restart_lsn` and `confirmed_flush_lsn`. PostgreSQL 17+ failover slots can improve continuity when synchronized and ready before promotion.

2. Connector + offset

The connector reads and transforms changes, then persists progress in an offset store. A crash can make recent events eligible for replay when the sink commit is ahead of the durable offset.

3. Transport

Kafka or another transport adds its own acknowledgement, partition, and consumer position. Retries and uncertain acknowledgements mean delivery can be observed more than once.

4. Sink + side effects

The sink decides whether replay converges or becomes a duplicate. Use an idempotency key, unique event ledger, version guard, or receiver-side reconciliation for external effects.

Source event → capture → transport → apply

Failover

Preserve source continuity, verify synchronized slot state, and move the connector route.

Replay

Assume the uncertain crash window can deliver the same identity again.

Convergence

Commit the sink effect and identity rule so a replay is safe rather than silent loss.

The production rule

Prefer a recoverable duplicate over an unrecoverable gap. Then make the sink idempotent enough that replay is boring.

Validate before production use: exact PostgreSQL, Debezium, transport, sink, managed-service, and routing behavior. Source references: [PostgreSQL failover](#), [replication slots](#), and [Debezium PostgreSQL connector](#).