

Production Postgres CDC Reliability Kit

Failover, replay, duplicates, and recovery readiness

A practical checklist for PostgreSQL 17+ logical CDC deployments

Prepared for Vertex Frontier readers

August 2026

Contents

1	How to use this checklist	1
2	Pre-flight readiness	1
2.1	Source and failover	1
2.2	Connector and transport	1
2.3	Sink correctness	2
3	Planned failover checklist	2
4	Incident decision table	3
5	Evidence to retain after recovery	3
6	Primary references	3

1 How to use this checklist

This checklist is a companion to the article *Postgres CDC in Production: Handling Failover, Retries, and Duplicates*. Complete it before a production launch, before a planned promotion, and after a connector or sink incident. Mark each item only when the evidence has been captured in a runbook, dashboard, test result, or change record.

The implementation target is PostgreSQL 17+ with current Debezium PostgreSQL connector behavior. The current PostgreSQL documentation set is linked because it documents the current failover-slot workflow. Older PostgreSQL versions, managed services, connector versions, transports, and sinks can differ.

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

2 Pre-flight readiness

2.1 Source and failover

Check	Evidence to capture	Owner / date
☒ PostgreSQL version is recorded	Primary, standby, managed-service, and extension versions	
☒ Logical slot is named and owned	Slot name, database, plugin, owner, and intended consumers	
☒ Slot state is monitored	active, restart_lsn, confirmed_flush_lsn, wal_status, and invalidation fields	
☒ Failover behavior is documented	Promotion route, DNS/proxy behavior, credentials, and connector endpoint	
☒ PostgreSQL 17+ failover conditions are tested	failover, synced, persistence, and standby readiness	
☒ WAL headroom is defined	WAL generation rate, storage headroom, alert owner, and recovery horizon	

2.2 Connector and transport

Check	Evidence to capture	Owner / date
-------	---------------------	--------------

☒ Offset storage is durable	Location, backup/retention policy, and restore test	
☒ Resume behavior is known	Last persisted LSN, restart procedure, and source-history check	
☒ Queue limits are deliberate	<code>max.batch.size</code> , <code>max.queue.size</code> , and <code>max.queue.size.in.bytes</code> rationale	
☒ Large transactions are tested	Largest realistic source transaction, heap headroom, and drain time	
☒ Transport ownership is clear	Topic, partition key, acknowledgements, consumer group, and rebalance behavior	
☒ Schema contract is owned	Converter, subject naming, compatibility mode, and sink version	

2.3 Sink correctness

Check	Evidence to capture	Owner / date
☒ Sink workload is classified	Current state, audit history, command, payment, webhook, or other side effect	
☒ Event identity is stable	Key format, source scope, retention, and collision policy	
☒ Replay behavior is tested	Crash before sink commit, after sink commit, and uncertain acknowledgement	
☒ Stale updates are guarded	Version or source-position comparison and same-source assumption	
☒ External effects reconcile	Receiver idempotency key, outbox, retry policy, and reconciliation owner	
☒ Snapshot overlap is covered	Snapshot READ versus live update handling and restart behavior	

3 Planned failover checklist

1. Record the current primary, standby, slot name, connector task, transport position, and sink checkpoint.
2. Confirm the failover slot is persistent, synchronized, not invalidated, and ready on the intended standby.

3. Confirm the connector route and authentication reach the future primary.
4. Fence writes and consumers according to the topology-specific change plan.
5. Promote the standby and verify identity, slot state, logical decoding, and route.
6. Allow replay; do not skip ahead merely to reduce lag.
7. Verify sink idempotency, duplicate outcomes, and business-side reconciliation.
8. Capture final source, connector, transport, sink, and WAL evidence before closing the change.

4 Incident decision table

Observed signal	Investigate first	Do not do blindly
Old <code>restart_lsn</code> and growing WAL	Connector flush, sink lag, slot ownership, and storage horizon	Drop the slot or increase the queue without a recovery decision
Connector restart replays events	Offset durability, sink commit state, and event identity	Delete offsets or skip records
Consumer lag with healthy source capture	Worker CPU/heap, serialization, producer acks, rebalance, and sink throughput	Assume PostgreSQL is the bottleneck
Incompatible schema error	Converter, registry compatibility, consumer version, and error path	Change compatibility policy as a shortcut
Duplicates after promotion	Slot continuity versus checkpoint and sink boundaries	Declare the source failover a complete correctness test

5 Evidence to retain after recovery

1. Source slot query output and WAL-pressure measurements.
2. Connector task state, last persisted source position, and offset-store evidence.
3. Transport topic/partition position and consumer-group state.
4. Sink transaction, event-ledger, version-guard, or reconciliation output.
5. Replay count, duplicate count, records manually reconciled, and business impact.
6. Exact versions, configuration change, incident timeline, and follow-up test case.

6 Primary references

- [PostgreSQL: Logical Replication Failover](#)
- [PostgreSQL: `pg\_replication\_slots` View](#)
- [PostgreSQL: Logical Decoding Concepts](#)
- [Debezium: PostgreSQL Connector](#)
- [Debezium: Outbox Event Router](#)
- [Confluent: Schema Evolution and Compatibility](#)