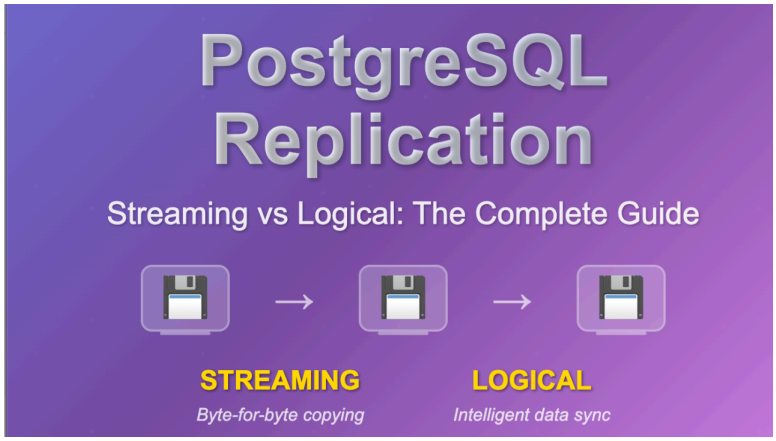


POSTGRESQL REPLICATION: STREAMING VS. LOGICAL — WHAT SENIOR DBAs NEED TO KNOW

Christian Foteping
Senior Database Administrator | PostgreSQL | MySQL | AWS Cloud | HA Architect | Performance Tuning & Query...

July 20, 2025

Replication in PostgreSQL isn't just about copying data. It's a core building block of any high availability (HA) architecture. Whether you're supporting mission-critical OLTP systems, real-time analytics platforms, or hybrid cloud deployments, understanding PostgreSQL's replication models is crucial for making the right architectural decisions.

PostgreSQL offers two primary forms of replication: **Streaming Replication (Physical)** and **Logical Replication**.

They're often discussed together, but serve very different operational purposes. Let's walk through their key differences — technically and strategically.

STREAMING REPLICATION (PHYSICAL)

Streaming replication is the foundation of PostgreSQL's **binary-level** replication. It operates by sending **WAL (Write-Ahead Log)** segments from the primary node to the standby node(s) over a persistent connection.

How it works:

- Replicas read and replay WALs in real time.
- Requires identical binaries, data directories, and configurations.
- Supports both **asynchronous** and **synchronous** modes.

Synchronous Replication ensures **zero data loss** (RPO = 0) if configured properly, but at the cost of increased write latency (RTO tradeoff). **Asynchronous Replication** offers better write performance but can introduce some lag and potential data loss on failover.

HA Applications:

- Essential for warm standby or hot standby configurations.
- Commonly used with **replication managers** like repmgr, Patroni, or pg_auto_failover for failover orchestration.
- Works well in **multi-AZ** setups on cloud platforms like AWS and GCP.

Limitations:

- The entire database instance is replicated — no granularity.
- Replica is read-only (until promoted).
- Tightly coupled PostgreSQL versioning.

This is your go-to for **low RPO/RTO recovery** and **automatic failover** designs.

LOGICAL REPLICATION

Logical replication operates at the **SQL level**, giving you fine-grained control over **which tables** and **what data** to replicate. It's ideal for **heterogeneous systems**, **cross-version upgrades**, or **multi-master** patterns.

How it works:

- Uses **logical decoding** to extract DML operations (INSERT/UPDATE/DELETE).
- Operates on a **publish/subscribe model**.
- Supports **row-level filtering** and **column projection** in newer versions.

HA & Operational Use Cases:

- Rolling upgrades between major PostgreSQL versions with **minimal downtime**.
- Feeding change data into **data lakes, downstream analytics, or microservices**.
- Building **active-active or bidirectional replication** setups (with custom conflict resolution).
- Connecting **on-prem databases to the cloud** in hybrid architectures.

Caveats for HA Architects:

- Does not replicate DDL changes, sequences, or roles.
- Higher write amplification and potential latency under heavy transaction volumes.
- Requires manual or tool-assisted failover orchestration (not tightly integrated with built-in HA tooling).

This is a surgical tool for **scalable integration** and **version-flexible replication**, not a direct HA solution in the traditional sense.

WHEN TO USE EACH

Use STREAMING REPLICATION for:

- Low-latency **synchronous HA** in production environments.
- Architectures requiring **near real-time standby promotion**.
- Read scaling via **hot standby** nodes.
- **Disaster recovery** where full binary parity is essential.

Use LOGICAL REPLICATION for:

- PostgreSQL **major version upgrades** with minimal cutover.
- **Partial replication** (selected tables or rows).
- **Data distribution across environments** (e.g., staging to production).
- Application-level **multi-master logic** or **integration pipelines**.

KEY DECISION POINTS FOR HA ARCHITECTURE

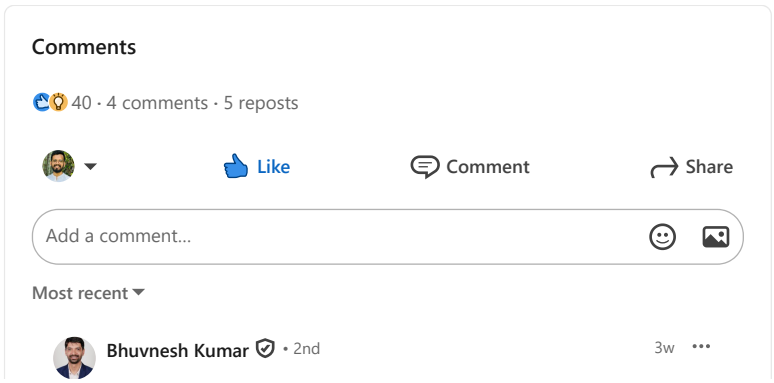
- **RPO/RTO Requirements**: Streaming (especially synchronous) gives you tighter control over both.
- **Topology Flexibility**: Logical offers decoupled versioning and schema independence.
- **Write Throughput vs. Data Movement**: Streaming is lighter; logical can become bottlenecked under load.
- **Failover Strategy**: Streaming integrates cleanly with automated failover systems; logical requires custom orchestration or third-party tooling.

FINAL THOUGHTS

PostgreSQL's replication toolset gives you flexibility, but not every tool solves every problem. As a senior DBA or architect, choosing between streaming and logical replication isn't about preference — it's about **matching replication strategy to business requirements** for uptime, scalability, and data integrity.

I've deployed both in production environments: from high-traffic fintech workloads in AWS using streaming with automatic failover, to cross-version data migrations using logical replication with precise cutover windows.

If you're designing for high availability, scaling read traffic, or planning version upgrades, PostgreSQL gives you options. The challenge is choosing the right tool for the right layer of your architecture.



AWS Database|Sqlserver|Postgres Architect-Facilitator

Thoughtful post, thanks Christian

Though based on my good experience with postgres migration i have found logical replication has great present and future as we can plan at table level plus their to we can implement conditional data ship in ...more

Like · 1 | Reply · 1 reply

Christian Foteping Author 3w
Senior Database Administrator | PostgreSQL | MySQL | AWS Cloud ...

Bhuvnesh Kumar Thanks for the thoughtful insight — I completely agree.

Logical replication is evolving fast and has become a strategic tool, especially for phased migrations, hybrid cloud rollouts, and ev ...more

Like | Reply

Shemal Gandhi • 2nd 3w
Senior Database Engineer | RDBMS | NoSQL | Multi-Cloud | DevOps | Tec...

Good read

Like · 1 | Reply · 1 reply

Christian Foteping Author 3w
Senior Database Administrator | PostgreSQL | MySQL | AWS Cloud ...

Shemal Gandhi Thanks

Like | Reply

Enjoyed this article?

Follow to never miss an update.

Christian Foteping

Senior Database Administrator | PostgreSQL | MySQL | AWS Cloud | HA Architect | Performance Tuning & Query Optimization

Krishna bhargav and Godson followed

Follow

More articles for you

Published on LinkedIn

Mentorship in Action: Deep Dive into PostgreSQL Permissions.

Christian Foteping

4 · 1 comment