PostgreSQL Replication Slot vs Without Slot

● What is a replication slot?

A replication slot is a **PostgreSQL feature** that ensures WAL (Write-Ahead Log) files needed by a replica are **never deleted** until the replica has received them.

Think of it like a **bookmark** in a streaming video — the server remembers exactly where the replica last watched, so it can resume without losing data.

- **Replication slots:** If using a slot, monitor pg_replication_slots on the primary. Particularly, note the restart_lsn for the slot – if it doesn't advance for a long time while WAL accumulates, your standby might be stuck or far behind. Also watch disk space for the pg_wal directory if a slot is in use.

Scenario Setup

- Primary server: PG-MAIN
- Replica server: PG-REPLICA
- WAL files: 0001, 0002, 0003, ...
- WAL size limit (no slot case): wal_keep_size = 64MB

Case 1: With replication slot

The primary keeps all WAL files until the replica confirms it has received and replayed them.

Step-by-step:
1. PG-MAIN writes WAL 0001 → 0005.
2. PG-REPLICA is slow — it's still processing 0002.
3. Normally, PG-MAIN would delete WAL 0001, but replication slot says: "Hold on! PG-REPLICA hasn't read this yet — keep it!"
4. Even if PG-REPLICA takes 3 hours to process, PG-MAIN keeps those WALs.
5. When PG-REPLICA finally catches up, PG-MAIN can safely delete old WALs.

Outcome:
✅PG-REPLICA never loses WALs and can always catch up.
⚠ If PG-REPLICA is down for days, WAL files pile up → PG-MAIN's disk might fill.

❖ **With replication slot**

1. WAL retention: The primary keeps WAL files until the replica confirms it has read them.

2. Data safety: No risk of the replica missing WALs, even if it lags.
3. Risk: If the replica stops for a long time, WAL files pile up on the primary → disk can fill up.
4. Use case: Reliable streaming replication where data loss is unacceptable.

Case 2: Without replication slot

The primary deletes old WALs based on wal_keep_size or archiving settings, regardless of whether the replica saw them.

Step-by-step:
1. PG-MAIN writes WAL 0001 → 0005.
2. PG-REPLICA is slow — still on 0002.
3. WAL 0001 is older than the wal_keep_size limit → PG-MAIN deletes it.
4. PG-REPLICA tries to fetch 0001, but it's gone.
5. PG-REPLICA errors out: requested WAL segment has already been removed.
6. The only fix is a full base backup to resync the replica.

Outcome:
✅No risk of WAL files filling the primary's disk.
❌If replica lags too much, it will break and need a rebuild.

❖ Without replication slot

1. WAL retention: Primary deletes old WAL files once they pass wal_keep_size or are archived.
2. Risk: If a replica lags too much, it may miss some WAL files. Then it must do a full resync.
3. Benefit: No disk buildup if replica is down.
4. Use case: Temporary replicas or when some data loss/rebuild is acceptable.

Real-World Analogy

With slot: Like a teacher holding onto homework until the slowest student has copied it. Safe for the student, but teacher's desk gets piled with papers.
Without slot: Teacher throws away old homework after a set time. Desk stays clean, but if a slow student comes late, they miss the notes and must start over.

Quick Decision Guide

Situation	Recommendation
Critical replica, no data loss allowed	Use replication slot
Temporary replica, or can tolerate rebuilds	No slot

Replica may be offline for long periods	Avoid slots unless you have huge disk space