**Asynchronous Replication Without Slot**

**********streaming replication**********
Environment deatiles:-
primary server:- 10.20.30.40
standby server:- 01.20.30.41
replication mode:- async

**step1:-**
update pg_hba conf file on primary server:-

ipv4:-
host all all 01.20.30.41/24 md5

replication:-
host replication all 01.20.30.41/24 md5

**step 2:- update postgresql.conf file:-**

1.archive_mode=on
2.archive_command = 'cp %p /AumShanti/pgdatabase/archive/%f' ----------cp %p /app/archive/%f
3.max_wal_sender = 10
4.wal_level = repllica
5.wal_keep_segments = 50 ----------(wal_keep_size = 50MB ----postgresql-15)
6.wal_log_hint = on
7.hot_standby = on
8.listen_address = *

---

**su - postgres**

**cp /etc/postgresql/16/main/postgresql.conf /etc/postgresql/16/main/postgresql.conf_20250702**

**vi /etc/postgresql/16/main/postgresql.conf**

listen_addresses='*'
wal_level=replica
max_wal_senders=10
wal_keep_size=256
hot_standby=on

**Save and exit the configuration file.**

**::::: Explanation of the options :::::**

- **listen_addresses = '*'**

- Tells PostgreSQL to listen on all network interfaces.
- Required so the standby server can connect to the primary server for replication.

- **wal_level = replica**

- Sets the level of information written to the WAL (Write-Ahead Log).
- The replica level is the minimum required for physical replication.

- Enables WAL archiving and streaming replication.

- **max_wal_senders = 10**

- Defines how many WAL sender processes the primary server can run simultaneously.
- Each standby server that connects consumes one WAL sender.
- Set this according to the number of standbys you plan to support.
- **wal_keep_size = 256**

- Specifies the minimum size (in MB) of WAL files to keep.
- Helps prevent replication failure due to missing WAL files if the standby is delayed.
- **hot_standby = on**
- Allows the standby server to accept read-only queries while in recovery mode.
- This setting must be enabled on the standby node.

## Create Replication User on Primary Server

su - postgres

psql

CREATE ROLE replicator WITH REPLICATION LOGIN ENCRYPTED PASSWORD 'replicator_password_2025';

::::: Explanation of the options :::::
- CREATE ROLE replicator: Creates a new role named replicator.
- WITH REPLICATION: Grants the role permission to use streaming replication.
- LOGIN: Allows the role to log in (without it, it's just a role, not a usable user).
- ENCRYPTED PASSWORD '...': Stores the password securely in the catalog.

**step3:- on standby server:-**

**1.delete data directory**

**2. stop postgres service**

**3. Run pg_basebackup to clone the standby instance**

Pg_basebackup -h <master ip> -U repuser -p 5432 -D $PGDATA -Fp -Xs -P -R -C
-h =host
-U = user
-p = Port
-D = data directory
-F = format (plain or tar)
-p = plain
-X = wal method ( none || fetch || stream)
-s = stream

-P = Progress
-R = write configuration for replication.
-C = Creation of replication slot named by the -S option

**step4:- start postgres service on standby**

**step 5:-verfiy:-**

**primary :-**
select * from pg_stat_replication;

**standby :-**
select * from pg_stat_wal_receiver;

**lag:-**

select now() -pg_last_xact_replay_timestamp() as replication_lag;

**to see archive genert manually we see these command:-**
SELECT pg_switch_wal();

==========
=

# Synchronous Replication:-

Synchronous Replication in PostgreSQL 16

Implementing synchronous replication in PostgreSQL ensures that each write transaction is confirmed on both the primary and a designated synchronous standby server before it is considered complete. This setup enhances data durability and consistency at the cost of increased commit latency. Below is a step-by-step guide on setting up synchronous replication in PostgreSQL 16.

**1.master server: -**
**postgresql.conf**

1.wal_level = replica
2. max_wal_senders = 5
3.synchronous_commit = on
4.Synchronous_standby_names = 'FIRST 1 (sync_standby_1)'
hot_standby = on.

2.Create a Base Backup:

pg_basebackup -h primary_ip -D /path/to/standby/data/dir -U replicator -Fp -Xs -P -R

# Asynchronous Replication With Slot

**step1:-**
update pg_hba conf file on primary server:-

ipv4:-
host all all 01.20.30.41/24 md5

replication:-
host replication all 01.20.30.41/24 md5

**step 2:- update postgresql.conf file:-**

1.archive_mode=on
2.archive_command = 'cp %p /AumShanti/pgdatabase/archive/%f' ----------cp %p /app/archive/%f
3.max_wal_sender = 10
4.wal_level = repllica
5.wal_keep_segments = 50 ----------(wal_keep_size = 50MB ----postgresql-15)
6.wal_log_hint = on
7.hot_standby = on
8.listen_address = *

## ● Create Replication User on Primary Server

su - postgres

psql

CREATE ROLE replicator WITH REPLICATION LOGIN ENCRYPTED PASSWORD 'replicator_password_2025';

## ● Create Replication Slot on Primary

Connect as postgres:
SELECT * FROM pg_create_physical_replication_slot('pgstandby');

Check slot:

SELECT slot_name, active, restart_lsn FROM pg_replication_slots;

## ● Setup Standby with Slot:-

Pg_basebackup -h <master ip> -U repuser -p 5432 -D $PGDATA -Fp -Xs -P -R -C -S pgstandby
-h =host
-U = user
-p = Port
-D = data directory
-F = format (plain or tar)

-p = plain
-X = wal method ( none || fetch || stream)
-s = stream
-P = Progress
-R = write configuration for replication.
-C = Creation of replication slot named by the -S option
-S = name of the replication slot.

==========

- **Difference: Replication With vs Without Slot**

| Feature | Without Slot | With Slot |
|---|---|---|
| **WAL Retention** | WAL removed once archived/wal_keep_size exceeded | WAL kept until standby consumes it |
| **Standby Safety** | If standby lags too much → replication breaks (needs pg_basebackup again) | Standby always can catch up, no data loss (unless disk full on primary) |
| **Disk Usage** | Less predictable (WAL removed quickly) | More WAL stored (risk of filling disk if standby down long time) |
| **Setup** | Easier (no slot config) | Slightly extra config (slot creation + assign slot) |
| **Use Case** | Temporary or fast catch-up standbys | Permanent standby nodes, critical DR setups |

## Replication With vs Without Replication Slot

| Aspect | Without Slot | With Slot |
|---|---|---|
| **WAL Retention** | WAL files are kept only until wal_keep_size or archiving policy. If standby lags too much → WAL recycled → replication breaks. | WAL files are retained until standby consumes them, guaranteed catch-up. |
| **Standby Safety** | Risk of standby desync if it is offline for too long. You may need to re-initialize with pg_basebackup. | Standby can always reconnect and resume from slot, no re-init needed. |
| **Disk Usage on Primary** | Lower, since WALs are recycled quickly. | Higher, since WALs are retained until slot standby consumes them (risk of disk full if standby down for days). |
| **Setup Complexity** | Simpler (just pg_basebackup -R). | Slightly more work (create slot or use -C -S in pg_basebackup). |
| **Monitoring Needed** | Less strict, but you must ensure standby does not fall behind too much. | Must monitor slot backlog (pg_replication_slots.restart_lsn) to avoid WAL bloat. |
| **Use Case** | - Temporary test replicas<br>- Read replicas that can be re-initialized easily<br>- Short-lived UAT/Reporting nodes | - Production DR replicas<br>- Permanent HA standby<br>- Critical systems where re-init would be costly |

## ⍰ Where Do We Use These?

**✅Without Slot**

- Short-lived replicas (e.g., creating a temporary reporting server for analytics).

- Non-critical replicas where losing sync and re-initializing is acceptable.
- When disk space on primary is limited and you don't want WAL retention issues.
- Common in simple streaming replication setups.

## ✅With Slot

- Disaster Recovery (DR) Standby → ensures standby can always catch up, even if network downtime occurs.
- Permanent read replicas (e.g., BI/Analytics, Reporting).
- Geo-replication across data centers (slots prevent WAL loss if network latency is high).
- Critical HA environments where standby must always be usable without re-init.

## ❓ Practical Rule of Thumb

- Use slot when the standby is permanent & critical (DR, HA, reporting).
- Skip slot when the standby is temporary or experimental (ad-hoc replicas, quick testing).

## ✅Final Recommendation

- For permanent HA / DR → Use replication slots.
- For temporary / test replicas → Skip slots.

## ✅Summary

- **With Slot** → ensures standby never misses WAL, safe for DR/HA, but risk of disk bloat if standby is offline too long.
- **Without Slot** → simpler, less disk usage, but standby may need full reinitialization if it lags.