Publication and Subscription:

1. check logical replication status by using-

Select * from pg_replication_stats where active=true;

Configuration file Subscriber

Critical `postgresql.conf` Settings for **Subscriber**

```
# Enable logical replication
wal_level = logical

# Number of concurrent WAL sender processes (each subscriber uses 1)
max_wal_senders = 400

# Number of replication slots allowed (logical and physical)
max_replication_slots = 400

# Number of logical replication workers for subscriptions (per database)
max_logical_replication_workers = 300  # Pick ONE value, don't duplicate

# Number of background workers for logical replication
max_sync_workers_per_subscription = 150  # Optional but useful for parallel table sync

# Total number of background worker processes
max_worker_processes = 800

# Total number of parallel workers
max_parallel_workers = 400
```

- **Create user:-**

```
CREATE USER replicator WITH REPLICATION ENCRYPTED PASSWORD 'your_very_strong_password';
```

- **Create publication on the primary server:**

CREATE PUBLICATION pub_<schema_name> FOR TABLE <schema_name.table_name>, <schema_name.table_name>, <schema_name.table_name> WITH (publish = 'insert, update, delete, truncate', publish_via_partition_root = false);

- **Grant permissions to the replication user on the primary server**

```
GRANT CONNECT ON DATABASE <database_name> TO replicator;

GRANT USAGE ON SCHEMA <schema_name> to replicator;
GRANT ALL ON ALL TABLES IN SCHEMA <schema_name> TO replicator;
GRANT ALL ON ALL SEQUENCES in schema <schema_name> TO replicator;
```

- **`wal_level = logical`**: Absolutely essential. This enables the generation of logical decoding information in the WAL, which is what logical replication uses.
- **`max_worker_processes`**: Set high enough to accommodate all background workers, including replication. A good rule of thumb: `max_worker_processes >= max_logical_replication_workers (on subscriber) + 1` (or more, depending on other background tasks).
- **`max_wal_senders`**: The maximum number of concurrent connections from standby servers or logical replication subscribers. It should be greater than or equal to `max_replication_slots` plus any physical replicas you might have.
- **`max_replication_slots`**: The maximum number of replication slots that can be active at the same time. This should be at least equal to the number of subscriptions you plan to have, plus any additional slots for table synchronization workers if you divide publications.

Note:-Important constraints:

- `max_logical_replication_workers` should not exceed `max_worker_processes`.
- `max_sync_workers_per_subscription` should be less than or equal to `max_logical_replication_workers`.

Explaination:-

❖ System Resources (CPU, RAM, I/O):-

- Each replication slot, WAL sender, and logical worker consumes CPU and memory.
- Don’t over-allocate these values beyond what your system can handle.
- Rule of thumb: For high values (e.g., 400), ensure the server has enough CPUs (ideally 1 core per 10–20 replication workers) and ample RAM.

❖ wal_level = logical

- Enables logical decoding of WAL (Write-Ahead Log).
- Required for publishing/subscribing changes to tables.
- Slightly increases WAL size compared to replica, due to additional information needed for decoding.

❖ max_wal_senders

- Number of WAL sender processes allowed.
- Each replication connection (logical or physical) uses one.
- Needed for both streaming replication and logical replication.
- Too many unused slots can cause **WAL bloat** (old WAL files won’t be removed until acknowledged).

❖ max_replication_slots

- Number of replication slots PostgreSQL will allow.
- Logical replication uses a slot for each subscription to track WAL position.
- One **WAL sender process** is used per active subscription.

❖ max_logical_replication_workers

- Total number of logical replication workers (background workers handling subscriptions).
- Higher value allows more concurrent logical subscriptions.
- max_sync_workers_per_subscription (optional but useful)
- Number of tables synchronized in parallel during initial sync.
- Speeds up initial table copy when a subscription is created.
- **max_logical_replication_workers:** total workers across all DBs.
- **max_sync_workers_per_subscription:** controls how many tables are synced in parallel during initial subscription.
- Higher values = faster sync, but more load on the system

❖ max_worker_processes

- Total background workers allowed by PostgreSQL.

- Logical replication workers, parallel workers, and extensions all count toward this.

❖ **max_parallel_workers**

- Maximum parallel workers that can be used by queries.
- Not directly tied to logical replication but important for overall performance.

Configuration file Publisher

Critical `postgresql.conf` Settings for **Subscriber**

```
# Enable logical replication
wal_level = logical

# Number of concurrent WAL sender processes (each subscriber uses 1)
max_wal_senders = 800

# Number of replication slots allowed (logical and physical)
max_replication_slots = 800

# Number of logical replication workers for subscriptions (per database)
max_logical_replication_workers = 800  # Pick ONE value, don't duplicate

# Number of background workers for logical replication
max_sync_workers_per_subscription = 200  # Optional but useful for parallel table sync

# Total number of background worker processes
max_worker_processes = 1600

# Total number of parallel workers
max_parallel_workers = 400
```

Step 1: Create the Exact Database on Subscriber

```
CREATE DATABASE parkdepot; -- Use your actual database name
```

Step 2: Extract Global Objects (Roles, Tablespaces) from Publisher

```
pg_dumpall -h <publisher_endpoint> -p 5432 -U <admin_user> --globals-only > "globals-
roles-tablespaces.sql"
```

Step 3: Extract Only the Schema (Structure) from Publisher

```
pg_dump -h <publisher_IP> -U <admin_user> -d <database_name> --schema-only > schema.sql
```

Step 5: Import Global Objects and Schema to Subscriber

```
psql -U <admin_user> -h <subscriber_IP> -p 5432 -d postgres -f globals-roles-
tablespaces.sql
psql -h <subscriber_IP> -U <admin_user> -d <database_name> -f schema.sql
```

Step 4: Create the Subscription

- **Create subscription on the secondary server:**

  CREATE SUBSCRIPTION sub_<schema_name>

  CONNECTION 'host=<host_ip_address> port=<host_port> user=replicator dbname=<database_name> connect_timeout=10 password=<password> sslmode=prefer'

  PUBLICATION pub_<schema_name>

  WITH (connect = true, enabled = true, copy_data = true, create_slot = true, synchronous_commit = 'off');

- **Grant permissions to the secondary server:**

  GRANT CONNECT ON DATABASE ecgc_reporting_db TO rep_user, smile_opr;

  GRANT USAGE ON SCHEMA <schema_name> TO rep_user, smile_opr; GRANT

  SELECT ON ALL TABLES IN SCHEMA <schema_name> TO rep_user;

Monitoring Your Logical Replication: The Lifeline

On the Publisher (`pg_stat_replication_slots`):

```
SELECT slot_name, active, wal_status, wal_delay_lags_bytes, restart_lsn, 
confirmed_flush_lsn FROM pg_stat_replication_slots;
```

On the Subscriber (`pg_stat_subscription` and `pg_stat_subscription_workers`):

```
SELECT subname, subenabled, subconninfo, substate, sublag FROM pg_stat_subscription;
SELECT subid, relid, last_applied_lsn, last_received_lsn, wal_records, wal_bytes,
sync_state FROM pg_stat_subscription_workers;
```