Setting up replication in PostgreSQL involves configuring a primary (master) server and one or more standby (replica) servers. PostgreSQL supports two main types of replication: **physical replication** (streaming replication) and **logical replication**. In this guide, we'll focus on **physical replication**, which is commonly used for high availability (HA) setups.

Steps for Setting Up Streaming (Physical) Replication in PostgreSQL

1. Prepare the Primary (Master) Server

a. Edit `postgresql.conf` on the Primary

On the primary server, you need to adjust some configuration settings to enable replication.

1. **Open the `postgresql.conf` file**:

```
[postgres@localhost pg_data]$ ls -lrt postgresql.conf
-rw-------. 1 postgres postgres 29686 Oct 20 13:09 postgresql.conf
[postgres@localhost pg_data]$ pwd
/postgres/pg_data
[postgres@localhost pg_data]$vi /postgres/pg_data/postgresql.conf
```

2. **Enable WAL archiving and configure replication settings**: Add or modify the following parameters:

```
wal_level = replica                 # Required for physical replication
max_wal_senders = 5                 # Number of replication connections allowed
wal_keep_size = 64                  # Amount of WAL files (in MB) to retain for
replicas
archive_mode = on                   # Enable archiving of WAL segments (optional)
archive_command = 'cp %p /path_to_wal_archive/%f'  # Command to archive WAL
(optional)
listen_addresses = '*'              # Allow connections from any IP (or specify an
IP)
```

   - wal_level = replica: This is required for streaming replication.
   - max_wal_senders: This sets the number of standby servers that can connect for replication.
   - wal_keep_size: This defines the amount of WAL logs to keep to prevent standby servers from falling behind.
   - listen_addresses: This allows remote connections to the primary server.

```
postgres=# show wal_level;
 wal_level
-----------
 replica
(1 row)

postgres=# show archive_mode;
 archive_mode
--------------
 on
(1 row)

postgres=# show max_wal_senders ;
 max_wal_senders
-----------------
```

```
 10
(1 row)

postgres=# show max_replication_slots;
 max_replication_slots
-----------------------
 10
(1 row)
```

3. **Reload or restart PostgreSQL** to apply these changes:

```
systemctl restart postgresql
```

b. Edit `pg_hba.conf` on the Primary

The `pg_hba.conf` file controls client authentication, and you need to allow replication connections from the standby server.

1. **Open the `pg_hba.conf` file**:

```
Vi /postgres/pg_data/pg_hba.conf

[postgres@localhost archivelogs]$ hostname -I
192.168.30.9 192.168.122.1
[postgres@localhost archivelogs]$
```

2. **Add a replication entry**: Add a line that allows the standby server to connect for replication. Replace `standby_ip_address` with the IP address of the standby server:

```
host    replication     replication_user 192.168.30.9/32        md5
host    replication     all     192.168.30.9/32     md5
```

   This allows the standby to connect using an IP address and `md5` authentication.

3. **Reload PostgreSQL**:

```
 systemctl reload postgresql
```

c. Create a Replication Role on the Primary

You need a user that has replication privileges.

1. **Create the replication role**:

```
postgres=# create user replication_user password 'postgres' replication;
CREATE ROLE
postgres=#
```

2. **Grant the role appropriate privileges**, if needed.

2. Prepare the Standby (Replica) Server

a. Base Backup from Primary

To set up replication, the standby needs a copy of the data directory from the primary server.

1. **Stop the standby server** if it's running:

   ```
   systemctl stop postgresql
   ```

2. **Use the `pg_basebackup` command** to copy the data from the primary to the standby server:

   ```
   pg_basebackup -h primary_ip_address -D /home/PG/backup/standby -U
   replication_user -P -R
   ```

   ```
   [postgres@localhost rep]$ pg_basebackup -h localhost -D
   /home/PG/backup/rep/ -U replication_user -P -R
   Password:
   38300/38300 kB (100%), 1/1 tablespace
   [postgres@localhost rep]$ ls -lrt
   ```

   - -h primary_ip_address: IP address of the primary server.
   - -D /postgres/pg_data/standby/data_directory: Data directory of the standby server.
   - -U replication_user: User created for replication.
   - -P: Show progress.
   - -R: Automatically creates a recovery.conf file on the standby, which is needed for replication.

```
[postgres@localhost rep]$ ls -lrt
total 284
-rw-------. 1 postgres postgres    225 Oct 20 17:39 backup_label
drwx------. 3 postgres postgres     60 Oct 20 17:39 pg_wal
drwx------. 7 postgres postgres     59 Oct 20 17:39 base
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_snapshots
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_serial
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_replslot
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_notify
drwx------. 4 postgres postgres     36 Oct 20 17:39 pg_multixact
drwx------. 4 postgres postgres     68 Oct 20 17:39 pg_logical
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_dynshmem
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_commit_ts
drwx------. 2 postgres postgres     32 Oct 20 17:39 log
-rw-------. 1 postgres postgres  29686 Oct 20 17:39 postgresql.conf
-rw-------. 1 postgres postgres    465 Oct 20 17:39 postgresql.auto.conf
drwx------. 2 postgres postgres     18 Oct 20 17:39 pg_xact
-rw-------. 1 postgres postgres      3 Oct 20 17:39 PG_VERSION
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_twophase
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_tblspc
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_subtrans
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_stat_tmp
drwx------. 2 postgres postgres      6 Oct 20 17:39 pg_stat
-rw-------. 1 postgres postgres   2640 Oct 20 17:39 pg_ident.conf
-rw-------. 1 postgres postgres   5636 Oct 20 17:39 pg_hba.conf
-rw-------. 1 postgres postgres     30 Oct 20 17:39 current_logfiles
drwx------. 2 postgres postgres   4096 Oct 20 17:39 global
-rw-------. 1 postgres postgres      0 Oct 20 17:39 standby.signal
-rw-------. 1 postgres postgres 225003 Oct 20 17:39 backup_manifest
[postgres@localhost rep]$ cat postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
archive_mode = 'on'
archive_command = 'cp %p /home/PG/archivelogs/%f'
```

```
primary_conninfo = 'user=replication_user password=postgres
channel_binding=prefer host=localhost port=5432 sslmode=prefer
sslcompression=0 sslcertmode=allow sslsssl_min_protocol_version=TLSv1.2
gssencmode=prefer krbsrvname=postgres gssdelegation=0
target_session_attrs=any load_balance_hosts=disable'
```

b. Modify the Standby's Configuration

1. **Edit `postgresql.conf`**: You need to set up replication parameters on the standby server.

   `Vi /home/PG/backup/rep/postgresql.conf`

   Add or modify the following lines:

   ```
   hot_standby = on                    # Enable read queries on the standby
   ```

2. **Edit `recovery.conf`**: The `pg_basebackup` command will automatically create a `recovery.conf` file in the data directory of the standby server if you used the `-R` option. However, you should verify its content.
   - o

   ```
   [postgres@localhost rep]$ cat postgresql.auto.conf
   # Do not edit this file manually!
   # It will be overwritten by the ALTER SYSTEM command.
   archive_mode = 'on'
   archive_command = 'cp %p /home/PG/archivelogs/%f'
   primary_conninfo = 'user=replication_user password=postgres
   channel_binding=prefer host=localhost port=5432 sslmode=prefer
   sslcompression=0 sslcertmode=allow
   sslsssl_min_protocol_version=TLSv1.2 gssencmode=prefer
   krbsrvname=postgres gssdelegation=0 target_session_attrs=any
   load_balance_hosts=disable'
   [postgres@localhost rep]$ vi postgresql.conf
   port=5434   → port no changed to 5434
   ```

   - o `standby_mode = 'on'`: Ensures the server stays in standby mode.
   - o `primary_conninfo`: Contains the connection information for the primary server.
3. **Start the standby server**:

   ```
   systemctl start postgresql

   or

   [postgres@localhost rep]$ pg_ctl -D /home/PG/backup/rep/ start
   waiting for server to start....2024-10-20 17:43:36.223 IST [27196] LOG:
   redirecting log output to logging collector process
   2024-10-20 17:43:36.223 IST [27196] HINT:  Future log output will appear in
   directory "log".
    done
   server started
   [postgres@localhost rep]$ psql
   psql (16.4)
   Type "help" for help.

   postgres=#
   ```

```
[postgres@localhost archivelogs]$ psql postgres --port=5434
psql (16.4)
Type "help" for help.

postgres=# \l+
```

3. Verify Replication Setup

a. Check Replication Status on the Primary

On the primary server, you can check if the standby is connected using the following SQL command:

```
SELECT * FROM pg_stat_replication;
```

```
[postgres@localhost rep]$ psql postgres --port=5434
psql (16.4)
Type "help" for help.

postgres=# SELECT * FROM pg_stat_replication;
 pid | usesysid | usename | application_name | client_addr | client_hostname |
client_port | backend_start | backend_xmin | state | sent_lsn | write_lsn | flush_l
sn | replay_lsn | write_lag | flush_lag | replay_lag | sync_priority | sync_state |
reply_time
-----+----------+---------+------------------+-------------+-----------------+---------
----+--------------+-------------+-------+----------+-----------+--------
---+------------+-----------+-----------+-----------+---------------+------------+----
--------
(0 rows)

postgres=#
```

This should show an entry for each standby server connected.

b. Check Replication Lag (Optional)

You can monitor the replication lag using:

```
SELECT pg_last_wal_receive_lsn(), pg_last_wal_replay_lsn();
```

```
postgres=# SELECT pg_last_wal_receive_lsn(), pg_last_wal_replay_lsn();
 pg_last_wal_receive_lsn | pg_last_wal_replay_lsn
-------------------------+------------------------
 0/53EC89D0              | 0/53EC89D0
(1 row)

postgres=#
```

The closer these two values are, the smaller the lag between the primary and standby.

4. Optional: Enable Read-Only Queries on the Standby

By default, the standby server in streaming replication mode is read-only. You can connect to the standby and execute read-only queries, which is useful for load balancing read operations.

Important Considerations:

- **WAL Retention**: Ensure the primary retains enough WAL files to allow the standby to catch up in case it falls behind. This is controlled by `wal_keep_size`.
- **Failover**: In case of primary failure, you can promote the standby to a primary by running:

```
pg_ctl promote -D /postgres/pg_data/standby/data_directory
```

- **Monitoring**: It's important to monitor the replication status, especially in production environments, to ensure that the standby is not falling behind.

This is the basic setup for **streaming replication**. You can extend this with **asynchronous/synchronous replication** or implement **failover management tools** like `repmgr` or `Patroni` for automatic failover handling.