Postgres 14 WAL Replication: Easy Step-By-Step Guide

Overview

This blog is a simple walk-through of the WAL Streaming replication using the latest Postgresql-14 on Ubuntu 20.04 focal:

- Setups Postgresql 14 on primary and secondary.
- Configure primary-secondary.
- Create a replica user on the primary with replication privileges.
- Copy data files from primary to secondary.
- Test.

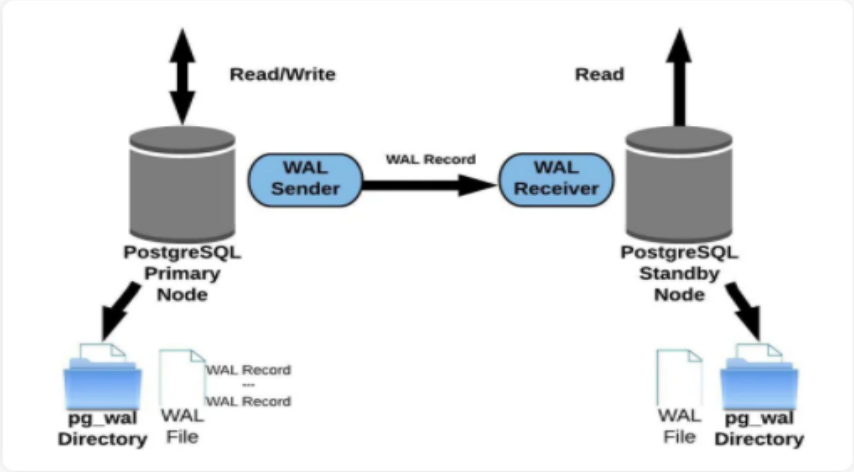
Why WAL replication!

Data replication ensures the duplication of data on an ongoing basis so that replication is in a consistently updated state and is synchronized with the source. This ensures high data availability and safety of data against undesired events such as crashes, system errors, etc. Postgres WAL replication meets this requirement through a feature called streaming replication. This feature is achieved through a master-slave configuration.

Prerequisite

- 2 Ubuntu server version 20.04.3 LTS.
- Port 5432 is enabled between servers.
- Postgresql 14 running on both.

Streaming WAL Records



Postgres 14 WAL Replication: Easy Step-By-Step Guide

➢ Install Postgresql 14 on primary and secondary:

- **Note the below steps should be applied on both.**
  - *Create the file repository configuration:*
  - *Import the repository signing key:*

```
root@primary-server:/var/lib# sudo sh -c 'echo "deb http://apt.postgresql.org/pub/
repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
root@primary-server:/var/lib# wget --quiet -O - https://www.postgresql.org/media/k
eys/ACCC4CF8.asc | sudo apt-key add -
OK
```

  - *Update the package lists:*

```
root@primary-server:/var/lib# sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://apt.postgresql.org/pub/repos/apt focal-pgdg InRelease
Hit:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:5 http://archive.ubuntu.com/ubuntu focal-security InRelease
Hit:6 https://repos.citusdata.com/community/ubuntu focal InRelease
Reading package lists... Done
```

  - *Install the latest version of PostgreSQL. If you want a specific version, use 'postgresql-12' or similar instead of 'postgresql':*

```
root@primary-server:/var/lib# sudo apt-get -y install postgresql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  pgdg-keyring postgresql-14 postgresql-client-14 postgresql-client-common
  postgresql-common
```

  - ***Note: make sure you issue the below command which makes sure Postgresql starts automatically when server get rebooted.***

```
root@primary-server:/var/lib# systemctl enable postgresql
Synchronizing state of postgresql.service with SysV service script with /lib/syste
md/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable postgresql
```

  - Let's check Postgresql status:

```
root@primary-server:/var/lib# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor pres>
     Active: active (exited) since Mon 2021-10-25 20:54:25 UTC; 8min ago
   Main PID: 403478 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 19108)
     Memory: 0B
     CGroup: /system.slice/postgresql.service

Oct 25 20:54:25 primary-server systemd[1]: Starting PostgreSQL RDBMS...
Oct 25 20:54:25 primary-server systemd[1]: Finished PostgreSQL RDBMS.
```

➢ Configure Primary server:

Postgres 14 WAL Replication: Easy Step-By-Step Guide

- Edit Postgresql.conf and apply the below changes:
- vim /etc/postgresql/main/14/postgresql.conf:

```
listen_addresses = '*'          # what IP address(es) to
listen on;
wal_level = replica             # minimal, replica, or logical
hot_standby = on  # "off" disallows queries during
recovery
```

- Create a user with replication privileges:

```
root@primary-server:/# systemctl restart postgresql
root@primary-server:/# sudo -u postgres psql
psql (14.0 (Ubuntu 14.0-1.pgdg20.04+1))
Type "help" for help.

postgres=# CREATE USER replica REPLICATION LOGIN ENCRYPTED PASSWORD '[redacted]';
CREATE ROLE
```

- Then we need to add the permission to **pg_hba.conf** file to enable a Standby server to access a Primary server.
- vim /etc/postgresql/14/main/pg_hba.conf:

```
host    replication     replica    SecondaryIP/32
md5
```

- Make sure you restart Postgresql after applying these changes.

➢ Configure Secondary server:

- Edit Postgresql.conf and apply the below changes:
- vim /etc/postgresql/main/14/postgresql.conf:

```
listen_addresses = '*'  # what IP address(es) to listen
on.
```

- Remove data directory from secondary server

```
root@secondary-server:/home/ubuntu# sudo su -l postgres
postgres@secondary-server:~$ rm -rfv /var/lib/postgresql/14/main/*
removed '/var/lib/postgresql/14/main/base/1/1255'
```

- Let's take backup from the primary data directory as shown below:

```
postgres@secondary-server:~/14/main$ pg_basebackup -[redacted] -p 5432 -U replica
 -D /var/lib/postgresql/14/main/ -Fp -Xs -R
Password:
```

- Notes:
  - It promotes you to enter the replica password created on the primary server.
  - -R creates the replication file configuration automatically as shown below:

Postgres 14 WAL Replication: Easy Step-By-Step Guide

```
root@secondary-server:/var/lib/postgresql/14/main# cat postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
primary_conninfo = 'user=replica password=[redacted] channel_binding=prefer host=[redacted]
[redacted] port=5432 sslmode=prefer sslcompression=0 sslsni=1 ssl_min_protocol_version=TL
Sv1.2 gssencmode=prefer krbsrvname=postgres target_session_attrs=any'
```

- Make sure you restart Postgresql and check its status:

```
root@secondary-server:/home/ubuntu# systemctl restart postgresql
root@secondary-server:/home/ubuntu# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset:
     Active: active (exited) since Mon 2021-10-25 21:49:35 UTC; 6s ago
    Process: 389964 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 389964 (code=exited, status=0/SUCCESS)

Oct 25 21:49:35 secondary-server systemd[1]: Starting PostgreSQL RDBMS...
Oct 25 21:49:35 secondary-server systemd[1]: Finished PostgreSQL RDBMS.
```

➢ Now let's create database and keep an eye on the secondary server:

Primary

```
postgres=# create database hello_world;
CREATE DATABASE
postgres=# \l
                                 List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Acc
-------------+----------+----------+-------------+-------------+------
 hello_world | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/pc
             |          |          |             |             | postg
 template1   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/pc
             |          |          |             |             | postg
```

Secondary

```
postgres=# \l
                                 List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-------------+----------+----------+-------------+-------------+-----------------------
 hello_world | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
 template1   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
(4 rows)
```

- Let's import a dump backup into primary to test the replication is streaming as expected:

```
postgres=# create database data;
CREATE DATABASE
postgres=# \q
postgres@primary-server:/home/ubuntu$ gunzip -c data_export_11_8_2021.sql.gz | psql data
```

- Check the secondary if the database exists:

```
postgres=# \l
                                 List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-------------+----------+----------+-------------+-------------+-----------------------
 data        | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 hello_world | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
```

- Let's check the size of the database in both:

Primary

```
postgres=# SELECT pg_size_pretty(pg_database_size('data'));
 pg_size_pretty
----------------
 1419 MB
(1 row)
```

Secondary

```
postgres=# SELECT pg_size_pretty(pg_database_size('data'));
 pg_size_pretty
----------------
 1419 MB
(1 row)
```

Postgres 14 WAL Replication: Easy Step-By-Step Guide

- You can also check on the primary to find out the state of streaming:
- 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_lsn  | replay_lsn | write_lag | flush_lag | replay_lag | sync_priority | sync_state |          reply_time
--------+----------+---------+------------------+----------------+-----------------+-------------+-------------------------------+--------------+-----------+--
---------+------------+------------+------------+-----------+-----------+------------+---------------+------------+-------------------------------
 408892 |    16384 | replica | 14/main          | [redacted] |                 |       52944 | 2021-10-25 21:49:33.068196+00 |              | streaming | 0
/A8C31FC0 | 0/A8C31FC0 | 0/A8C31FC0 | 0/A8C31FC0 |           |           |            |             0 | async      | 2021-10-25 22:23:41.891326+00
(1 row)
```

- On secondary server you check by:
- select * from pg_stat_wal_receiver;

```
  pid   |  status   | receive_start_lsn | receive_start_tli | written_lsn | flushed_lsn | received_tli |      last_msg_send_time       |     last_msg_receipt_time
 | latest_end_lsn |        latest_end_time        | slot_name |  sender_host  | sender_port |
                                                     conninfo

--------+-----------+-------------------+-------------------+-------------+-------------+--------------+-------------------------------+-------------------------------
-+----------------+-------------------------------+-----------+---------------+-------------+----------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------
 389951 | streaming | 0/3000000         |                 1 | 0/A8C31FC0  | 0/A8C31FC0  |            1 | 2021-10-25 22:28:22.30422+00 | 2021-10-25 22:28:22.405516+00
 | 0/A8C31FC0     | 2021-10-25 22:14:51.050281+00 |           | [redacted] |        5432 | user=replica password=******** channel_binding=prefer dbname=replicatio
n host=[redacted] port=5432 fallback_application_name=14/main sslmode=prefer sslcompression=0 sslsni=1 ssl_min_protocol_version=TLSv1.2 gssencmode=prefer krbsrvna
me=postgres target_session_attrs=any
(1 row)
```

- Now you have a primary server and streaming replication performing as it should be.

Hope this helps!!

Postgres 14 WAL Replication: Easy Step-By-Step Guide

➢ Refrences:

1. https://hevodata.com/learn/postgres-wal-replication/#master
2. https://www.postgresql.org/download/linux/ubuntu/