YouTestMe

PostgreSQL Hot Standby Configuration

Table of Contents

1 Introduction

The main goal is to implement a standby PostgreSQL database that would use real-time replication to maintain data consistency if the primary server failed. The Standby database will work in read-only mode and be used for data querying.

- Postgres uses write-ahead logging (WAL) to archive database transactions continuously. WAL writes an entry in a log file for each change made to the data files. The system uses these log entries to perform point-in-time restoration from archives and to keep the standby server up to date. This means that when you set up Hot Standby, you're also setting up archiving.
- Streaming replication is the process of updating the standby server with WAL entries. This process operates asynchronously, which means it can take some time for the standby server to receive an update from the primary server. Though this delay can be very short, synchronization between the servers is not instantaneous. If your application requires strict consistency between the database instances, you should consider another approach.
- Postgres doesn't provide the functionality to failover when the primary server fails automatically. This is a manual operation unless you use a third-party solution to manage failover.
- Load balancing is not automatic with Hot Standby. Suppose load balancing is a requirement for your application. In that case, you must provide a load-balancing solution that uses the primary server for read-write operations and the standby server for read-only operations.

2 Understanding streaming replication

The implementation of streaming replication in PostgreSQL involves the following key components and steps:

1. Primary Server: The primary server is the main database server that receives read and write queries from applications. It tracks changes made to the database through the Write-Ahead Logging (WAL) mechanism.

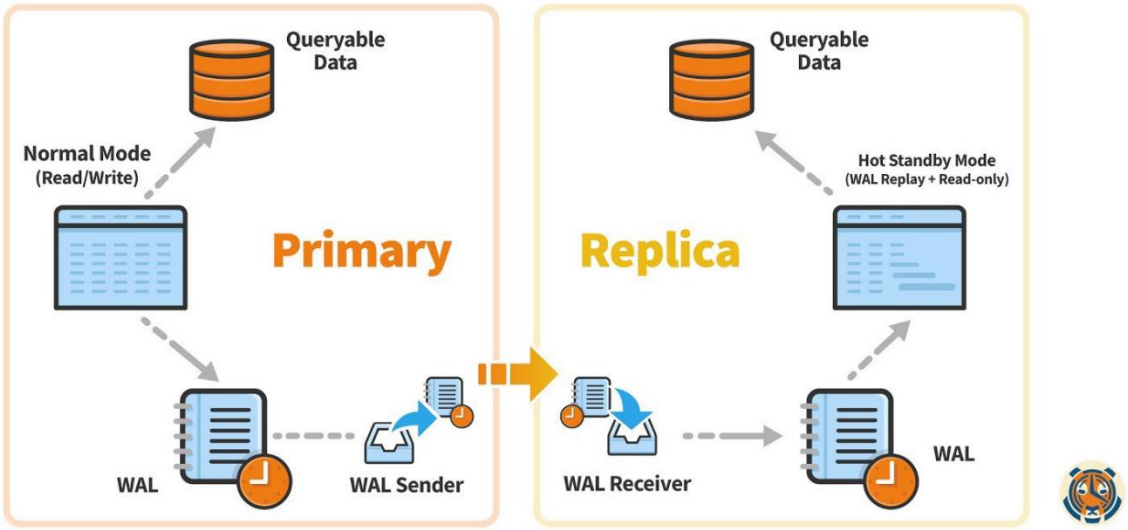
2. Standby Servers: Standby servers are replicas of the primary server that receive and apply the changes from the primary server. They operate in a continuous streaming mode, receiving the WAL data and applying it to their database copy.

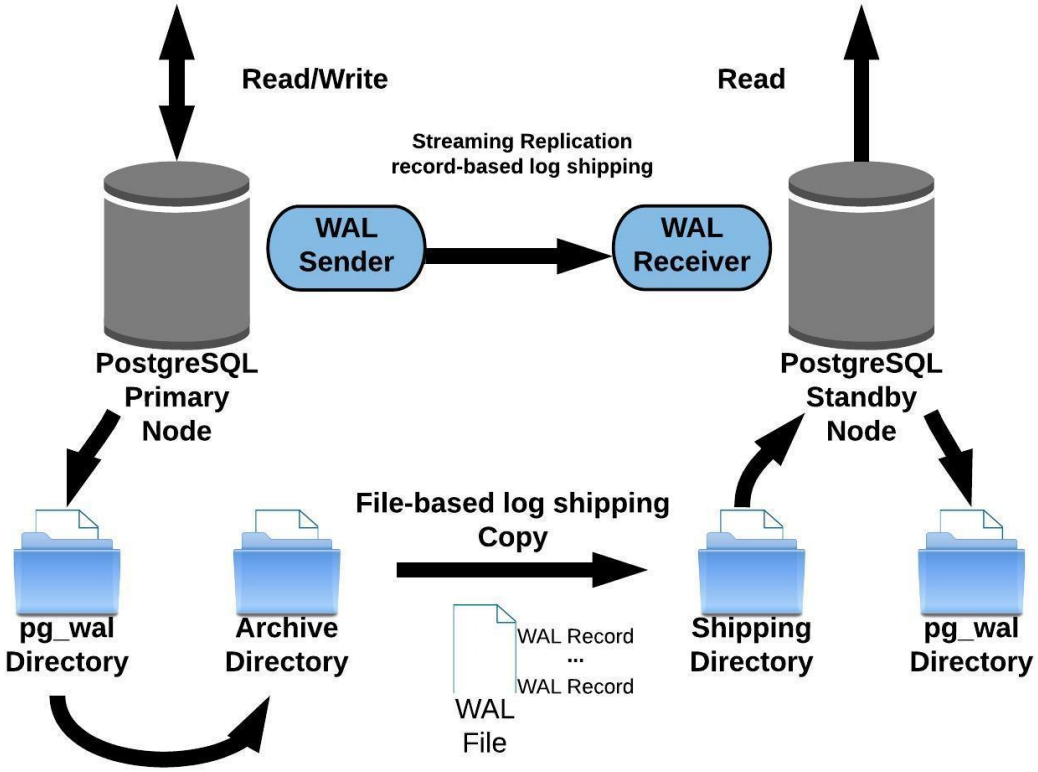
3. WAL Shipping: The primary server continuously writes the WAL records to a designated location, usually called the WAL archive. Standby servers fetch the WAL records from the archive or directly from the primary server using the streaming replication protocol.

4. Synchronous or Asynchronous Replication: PostgreSQL allows configuring the replication mode as synchronous or asynchronous. In synchronous replication, the primary server waits for confirmation from at least one standby server that the data changes have been successfully applied before acknowledging the transaction commit. Asynchronous replication does not require immediate confirmation and has lower latency but provides a higher possibility of data loss in case of a primary server failure.

5. Continuous Streaming: Standby servers connect with the primary server to receive the WAL data. The primary server sends the WAL records as soon as they are written, ensuring near-real-time replication.

6. Apply Changes: Standby servers receive the WAL records and apply them to their local database copy, keeping it synchronized with the primary server. They use the **recovery** process to apply the WAL records and bring the standby server to a consistent state.


Queryable Data
Normal Mode (Read/Write)
Primary
WAL
WAL Sender
Queryable Data
Hot Standby Mode (WAL Replay + Read-only)
Replica
WAL Receiver
WAL


Read/Write
Read
Streaming Replication
record-based log shipping
WAL Sender
WAL Receiver
PostgreSQL Primary Node
PostgreSQL Standby Node
File-based log shipping
Copy
pg_wal Directory
Archive Directory
WAL Record
...
WAL Record
WAL File
Shipping Directory
pg_wal Directory

3 Hot standby configuration

The configuration process can be summarized in the following steps:

- Deploy preconfigured virtual appliances running the PostgreSQL database (primary and standby node)
- Create a new table for testing purposes
- Configure the primary server
- Back up the primary server to the standby server
- Configure the standby server to run in Hot Standby mode
- Start the standby server and test it

3.1 PostgreSQL deployment

To prepare the primary and the standby database server for the configuration phase, please consult the Installation and Support Manual for YouTestMe Enterprise Edition Software.

You will learn to deploy the preconfigured PostgreSQL virtual appliances and configure their networking.

Note the IP address of each server. You'll need these values when you modify the configuration files.

SSH to both machines (primary and standby server). For security reasons, you must use the "ytmlogin" user to establish an SSH connection and then switch to the "postgres" shell.

- `$ ssh ytmlogin@[database_ip]`
- `$ su - postgres`

3.2 Creating the textbook table

On the primary server, create a simple table for testing purposes.

1. Run PSQL as user ***postgres*** and access the database named ***postgres***:
   ```
   $ psql postgres
   ```
2. At the PSQL prompt, enter the following command to create the table:
   ```
   CREATE TABLE textbook (user_email text, user_id serial, date timestamp, message text);
   ```
3. Add an entry to the table:
   ```
   INSERT INTO textbook (user_email, date, message) VALUES ( 'jim@gmail.com', current_date, 'This is a test.');
   ```
4. Enter \q to exit PSQL.

3.3 Configuring the primary server

To configure the primary server, you will:

- Create a Postgres user for replication activities
- Create a directory to store archive files [optional]
- Edit two configuration files: ***pg_hba.conf*** and ***postgresql.conf***

3.3.1 Create a user for replication

Postgres requires a user, also called a role, with special permissions to perform replication. On the primary server, run the following command as the "postgres" user:

```
$ createuser -U postgres repuser -P -c 5 --replication
```

- The "-U" option tells the "createuser" command to use the user "postgres" to create the new user
- The name of the new user is "repuser". You'll enter that username in the configuration files
- -P prompts you for the new user's password. **Important:** For any system with an Internet connection, use a strong password to help keep the system secure
- -c sets a limit for the number of connections for the new user. The value 5 is sufficient for replication purposes
- --replication grants the REPLICATION privilege to the user named "repuser"

3.3.2 Create the archive directory [optional]

Create a directory to store archive files. This directory is named "data" in this particular case. You'll use this path in one of the configuration files.

In the SSH terminal for the primary server, enter the following command:

```
$ mkdir -p /ytmdata/archivedir
```

3.3.3 Edit *pg_hba.conf*

The configuration file contains the settings for the client authentication. You must add an entry for the user "repuser" to enable replication.

- Edit the configuration file:
  ```
  $ vim /usr/lib/pgsql/pg_version/data/pg_hba.conf
  ```
- After the example replication entries, add the following lines. Replace <standby-IP> with the external IP address of the standby server:
  ```
  # Allow replication connections
  host        replication      repuser     <standby-IP>/32        md5
  ```
- Save and close the file.

3.3.4 Edit *postgresql.conf*

The configuration file contains the main settings for PostgreSQL. Here, you will modify the file to enable archiving and replication.

1. Edit the file. In the terminal for the primary server, enter the following command:
   ```
   $ vim /usr/lib/pgsql/pg_version/data/postgresql.conf
   ```
2. In the **WRITE-AHEAD LOG** section, in the **Settings** section, uncomment the following line:
   ```
   wal_level = replica
   ```
3. In the **Archiving** section, change the archive mode:
   ```
   archive_mode = on
   ```
4. Change the value for the archive command. This setting tells Postgres to write the archive files to the directory that you created in a previous step:
   ```
   archive_command = 'test ! -f /ytmdata/archivedir/%f && cp %p
   /ytmdata/archivedir/%f'
   ```
5. In the **REPLICATION** section, in the **Sending Servers** section, specify the value for the maximum number of WAL sender processes by uncommenting the following line:
   ```
   max_wal_sender = 10
   ```
6. Save and close the file
7. Restart the primary server:
   ```
   $ pg_ctl -D /usr/lib/pgsql/pg_version/data -l /usr/lib/pgsql/logfile
   restart
   ```

3.4 Backing up the primary server to the standby server

Before making changes on the standby server, stop the service. In the SSH terminal for the standby server, run the following command:

```
$ pg_ctl -D /usr/lib/pgsql/pg_version/data -l /usr/lib/pgsql/logfile stop
```

Important: Don't start the service again until all configuration and backup steps are complete. You must bring up the standby server in a state where it is ready to be a backup server. This means that all configuration settings must be in place and the databases must be already synchronized. Otherwise, streaming replication will fail to start.

3.4.1 Run the backup utility

The backup utility, named "pg_basebackup", will copy files from the data directory on the primary server to the same directory on the standby server.

- Make sure you're running commands in the **postgres** shell. In the SSH terminal for the standby server, enter the following command:
  ```
  $ su - postgres
  ```
  Continue to use the **postgres** shell for the remainder of this tutorial.
- The backup utility won't overwrite existing files, so you must rename the data directory on the standby server. Run the following command:
  ```
  $ mv /usr/lib/pgsql/pg_version/data /usr/lib/pgsql/pg_version/data_old
  ```
- Run the backup utility. Replace <primary-IP> with the external IP address of the primary server:
  ```
  $ pg_basebackup -h <primary IP> -D /usr/lib/pgsql/pg_version/data -U repuser
  -v -P --wal-method=stream
  ```

  The backup utility will prompt you for the password for the user named "repuser".

The backup process should take some time, depending on the data size located on the primary server. When it's done, you can configure the standby server.

3.5 Configuring the standby server

To configure the standby server, you'll edit "postgresql.conf" and create a new configuration file named "standby.signal".

3.5.1 Edit postgresql.conf

For the standby server, follow these steps:

- Edit the file. In the terminal for the standby server, enter the following command:
  ```
  $ /var/lib/pgsql/pg_version/data/postgresql.conf
  ```
- In the **REPLICATION** section, in the **Standby Servers** section, turn on Hot Standby and uncomment the line:
  ```
  hot_standby = on
  ```
- Set the connection string to the primary server in the same Standby Servers section. Replace [primary-external-IP] with the external IP address of the primary server. Replace [password] with the password for the user named "*repuser*".
  primary_conninfo = 'host=[primary-external-IP] port=5432 user=repuser password=[password]'
- Save and close the file

```
# - Standby Servers -

# These settings are ignored on a primary server.

primary_conninfo = 'host=192.168.1.100 port=5432 user=repuser password=mydbpass'
#primary_slot_name = ''                 # replication slot on sending server
promote_trigger_file = '/tmp/postgresql.trigger.5432'           # file name whose
hot_standby = on                        # "off" disallows queries during recovery
                                        # (change requires restart)
#max_standby_archive_delay = 30s        # max delay before canceling queries
                                        # when reading WAL from archive;
```

Standby configuration example

3.5.2 Create the configuration file *standby.signal*

If the file "standby.signal" is present in the data directory, the server will start in standby mode and enter recovery. Create an empty file using the following command:

```
$ touch /usr/lib/pgsql/pg_version/data/standby.signal
```

3.6 Start the standby server

You have everything in place and are ready to bring up the standby server. In the terminal for the standby server, enter the following command:

```
$ pg_ctl -D /usr/lib/pgsql/pg_version/data -l /usr/lib/pgsql/logfile start
```

3.7 Seeing the replication at work

To demonstrate that the replication between the primary and standby servers is working, you can add a row to the "textbook" table on the primary server and then query the standby server to see the new row.

Recall that you have already added one row to the table on the primary server. Start by verifying that the standby server has the same information.

- On the standby server, start PSQL:
  `$ psql postgres`
- At the PSQL prompt, enter the following query:
  `select * from textbook;`

  You should see that the table contains the single row that you initially added. Now, add a second row on the primary server.

- On the primary server, start PSQL:
  ```
  $ psql postgres
  ```
- At the PSQL prompt, enter the following command:
  ```
  INSERT INTO textbook (user_email, date, message) VALUES
  ('jim@gmail.com', current_date, 'Now we are replicating.');
  ```
- Switch back to the standby server terminal and repeat the query for all rows of the textbook:
  ```
  select * from textbook;
  ```

  You should now see that the standby server has received the update from the primary server.

- To exit PSQL, enter **\q**

4 Contact Information

If you encounter any difficulties during the process or have any questions, please do not hesitate to contact our support team at support@youtestme.com. Our dedicated team of experts is available to assist you with any issue that you may have.

https://www.youtestme.com/support-services/