**POSTGRE-SQL INSTALLATION**

PostgreSQL Master name and IP address: PGMaster and 192.168.100.88 PostgreSQL Slave/Replica name and IP address: PGSlave and 192.168.100.94 and 192.168.100.111

**On Master and Slave servers, PostgreSQL 16 must have installed.**

**Steps for Master Node**

1**:** Configurations on master server 1. On master server, configure the IP address(es) listen to for connections from clients in postgresql.conf by removing # in front of listen\_address and give \*.

Which means listen connections from all. listen\_addresses = '\*'

2. Now, connect to PostgreSQL on master server and create replica login.

CREATE USER replicator WITH REPLICATION ENCRYPTED PASSWORD 'admin@123';

3. Enter the following entry pg\_hba.conf file which is located in /etc/postgresql/14/main on Ubuntu(debian systems).

host replication replicator 192.168.100.94/24 md5 and host replication replicator 192.168.100.111/24 md5

**Steps for Workers Node**

How to setup streaming replication in PostgreSQL step by step on Ubuntu

Now, restart the PostgreSQL on Master server by using below command.

sudo systemctl restart postgresql

sudo systemctl status postgresql

Step3: Configurations on slave(standby) server

1. We have to stop PostgreSQL on Slave server by using following command.
2. sudo systemctl stop postgresql
3. sudo systemctl status postgresql
4. . Now, switch to postgres user and take backup of main(data) directory. su - postgres cp -R /var/lib/postgresql/16/main/ /var/lib/postgresql/16/main\_old/
5. Now, remove the contents of main(data) directory on slave server. rm -rf /var/lib/postgresql/16/main/
6. Now, use basebackup to take the base backup with the right ownership with postgres(or any user with right permissions).

pg\_basebackup -h 192.168.72.128 -D /var/lib/postgresql/16/main/ -U replicator -P -v -R -X stream -C -S slaveslot1

Then provide the password for user replicator created in master server. pg\_basebackup: initiating base backup,

waiting for checkpoint to complete ....................................

pg\_basebackup: syncing data to disk ...

pg\_basebackup: base backup completed

7.Notice that standby.signal is created and the connection settings are appended to postgresql.auto.conf.

ls -ltrh /var/lib/postgresql/16/main/

8. postgresql.conf and there is a standby.signal file present in the data directory.

9. Now connect the master server, you should be able to see the replication slot called slotslave1 when you open the pg\_replication\_slots view as follows. SELECT \* FROM pg\_replication\_slots;

10. Test replication setup 1. Now start PostgreSQL on slave(standby) server. systemctl start postgresql

11. Now, try to create object or database in slave(standby) server. It throws error, because slave(standby) is read-only server. create database slave1;

12. WE can check the status on standby using below command. SELECT \* FROM pg\_stat\_wal\_receiver;

13. Now, verify the replication type synchronous or aynchronous using below command on master database server. SELECT \* FROM pg\_stat\_replication;

14. Lets create a database in master server and verify its going to replicate to slave or not.

create database stream;

15. Now, connect to slave and verify the database copied or not.

select datname from pg\_database;

16. If you want to enable synchronous, the run the below command on master database server and reload postgresql service.

ALTER SYSTEM SET synchronous\_standby\_names TO '\*';

systemctl reload postgresql

Thats all. We have successfully setup streaming replication in PostgreSQL step by step on Ubuntu.

**To set up a Virtual IP (VIP) using keepalived across your three PostgreSQL nodes (192.168.100.88, 192.168.100.94, and 192.168.100.111), you’ll need to follow these steps. The goal is to ensure high availability, where the VIP will always point to the active primary node. If the primary fails, one of the standby nodes will take over.**

**Initial PostgreSQL Keepalived Setup (Primary Node - 192.168.100.88):**

1. **Edit postgresql.conf:**

sudo nano /etc/postgresql/16/main/postgresql.conf

Ensure the following settings are enabled:

listen\_addresses = '\*'

wal\_level = replica

max\_wal\_senders = 10

hot\_standby = on

1. **Configure pg\_hba.conf:**

sudo nano /etc/postgresql/16/main/pg\_hba.conf

Add the following lines to allow replication:

host replication replicator 192.168.100.94/32 md5

host replication replicator 192.168.100.111/32 md5

**Install and Configure Keepalived**

Install keepalived on all three nodes.

**Install Keepalived:**

For Ubuntu/Debian:

sudo apt-get install keepalived

**Configure Keepalived on All Nodes:**

1. **Edit** **/etc/keepalived/keepalived.conf** on all nodes with the following configuration.
2. **List network interfaces**
3. **Handle the keepalived\_script User Warning**

**On 192.168.100.88 (Primary Node):**

**List network interfaces:**

ip link show

Look for the correct interface name (it might be something like ens33, eth1, or similar).

**Handle the keepalived\_script User Warning**

You can either create the keepalived\_script user or run the script as the root user.

**Create the user:**

sudo useradd -r -s /bin/false keepalived\_script

This creates a system user with no shell access, just for script execution.

**Nano /etc/keepalived/keepalived.conf**

vrrp\_script chk\_pgsql {

script "pgrep postgres"

interval 2

weight 2

user root

}

vrrp\_instance VI\_1 {

state MASTER

interface ens32

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass gama1234

}

virtual\_ipaddress {

192.168.100.200 # This is the VIP

}

track\_script {

chk\_pgsql

}

}

**On 192.168.100.94 (Standby Node 1):**

**List network interfaces:**

ip link show

Look for the correct interface name (it might be something like ens33, eth1, or similar).

**Handle the keepalived\_script User Warning**

You can either create the keepalived\_script user or run the script as the root user.

**Create the user:**

sudo useradd -r -s /bin/false keepalived\_script

This creates a system user with no shell access, just for script execution.

**Nano /etc/keepalived/keepalived.conf**

vrrp\_script chk\_pgsql {

script "pgrep postgres"

interval 2

weight 2

user root

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens32

virtual\_router\_id 51

priority 90

advert\_int 1

authentication {

auth\_type PASS

auth\_pass gama1234

}

virtual\_ipaddress {

192.168.100.200

}

track\_script {

chk\_pgsql

}

}

**On 192.168.100.111 (Standby Node 2):**

**List network interfaces:**

ip link show

Look for the correct interface name (it might be something like ens33, eth1, or similar).

**Handle the keepalived\_script User Warning**

You can either create the keepalived\_script user or run the script as the root user.

**Create the user:**

sudo useradd -r -s /bin/false keepalived\_script

This creates a system user with no shell access, just for script execution.

**Nano /etc/keepalived/keepalived.conf**

vrrp\_script chk\_pgsql {

script "pgrep postgres"

interval 2

weight 2

user root

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens32

virtual\_router\_id 51

priority 80

advert\_int 1

authentication {

auth\_type PASS

auth\_pass gama1234

}

virtual\_ipaddress {

192.168.100.200

}

track\_script {

chk\_pgsql

}

}

* **state:** Set to MASTER on the primary node and BACKUP on the standby nodes.
* **priority:** The node with the highest priority will be the MASTER. Lower priorities are used for backup nodes.
* **virtual\_ipaddress:** This is the VIP that will float between the nodes.

1. **Start Keepalived:**

On each node, start and enable keepalived:

sudo systemctl start keepalived

sudo systemctl enable keepalived

sudo systemctl status keepalived

ip addr show (Check in all nodes of biding 192.168.100.255 with current IP addr 192.168.100.88)

**Troubleshooting**

**1) The error message Connection to 192.168.100.94:5432 refused indicates that the PostgreSQL server at the specified IP address and port is not accepting connections. This could be due to several reasons. Here are some steps to diagnose and resolve the issue:**

**Step-by-Step Troubleshooting**

1. Check PostgreSQL Service Status: Ensure that PostgreSQL is running on the server.

sh

Copy code

sudo systemctl status postgresql

If the service is not running, start it:

sh

Copy code

sudo systemctl start postgresql

1. Check PostgreSQL Configuration for TCP/IP Connections: Ensure PostgreSQL is configured to accept TCP/IP connections.
   * Open the postgresql.conf file:

sh

Copy code

sudo nano /etc/postgresql/12/main/postgresql.conf

* + Look for the listen\_addresses setting and ensure it includes the server's IP address or is set to '\*' to listen on all interfaces:

plaintext

Copy code

listen\_addresses = '\*'

* + Save and exit the file.

1. Check Host-Based Authentication Configuration: Ensure the pg\_hba.conf file allows connections from your IP address.
   * Open the pg\_hba.conf file:

sh

Copy code

sudo nano /etc/postgresql/12/main/pg\_hba.conf

* + Add an entry for the client IP address:

plaintext

Copy code

host all all 192.168.100.1/32 md5

* + Save and exit the file.

1. Restart PostgreSQL Service: After making changes to the configuration files, restart the PostgreSQL service:

sh

Copy code

sudo systemctl restart postgresql

1. Check Firewall Settings: Ensure that the firewall on the server is not blocking the connection to port 5432.
   * For ufw:

sh

Copy code

sudo ufw allow 5432/tcp

1. Verify Network Connectivity: Ensure that you can reach the server from the client machine. Use ping or telnet to test connectivity.
   * Ping the server:

sh

Copy code

ping 192.168.100.94

* + Test the port with telnet:

sh

Copy code

telnet 192.168.100.94 5432

Example Commands

Here are the commands summarized:

1. Check PostgreSQL Service:

sh

Copy code

sudo systemctl status postgresql

sudo systemctl start postgresql

1. Edit Configuration Files:
   * postgresql.conf:

sh

Copy code

sudo nano /etc/postgresql/12/main/postgresql.conf

# Ensure listen\_addresses = '\*'

* + pg\_hba.conf:

sh

Copy code

sudo nano /etc/postgresql/12/main/pg\_hba.conf

# Add host entry for your client

1. Restart PostgreSQL:

sh

Copy code

sudo systemctl restart postgresql

1. Firewall Configuration:

sh

Copy code

sudo ufw allow 5432/tcp

1. Network Connectivity Test:

sh

Copy code

ping 192.168.100.94

telnet 192.168.100.94 5432

By following these steps, you should be able to diagnose and resolve the connection issue to your PostgreSQL server. If the problem persists, ensure that there are no network issues or misconfigurations preventing the connection.

**2)The error message you're encountering, FATAL: no pg\_hba.conf entry for host "192.168.100.1", user "postgres", database "postgres", SSL encryption, indicates that the pg\_hba.conf file on your PostgreSQL server does not have the necessary entry to allow connections from the host 192.168.100.1 using SSL.**

**To resolve this issue, you need to modify the pg\_hba.conf file on your PostgreSQL server to allow connections from the specified host.**

**Step-by-Step Solution:**

1. **Open the pg\_hba.conf File:** The pg\_hba.conf file is usually located in the PostgreSQL data directory. You can edit it using a text editor like nano or vim.

sh

Copy code

sudo nano /etc/postgresql/12/main/pg\_hba.conf

1. **Add a New Entry for the Host:** Add an entry that allows connections from 192.168.100.1 using the postgres user and the postgres database with SSL. Add the following line at the end of the file:

plaintext

Copy code

hostssl all all 192.168.100.1/32 md5

Here’s a breakdown of the entry:

* + hostssl: This specifies that the connection must use SSL.
  + all: This matches all databases.
  + all: This matches all users.
  + 192.168.100.1/32: This specifies the host or IP address. The /32 indicates a single IP address.
  + md5: This specifies that the password authentication method is used.

1. **Reload PostgreSQL Configuration:** After making changes to the pg\_hba.conf file, reload the PostgreSQL service to apply the changes.

sh

Copy code

sudo systemctl reload postgresql

1. **Verify the Connection:** Try connecting to the PostgreSQL server again from the host 192.168.100.1 using the postgres user and the postgres database.

sh

Copy code

psql -h 192.168.100.110 -U postgres -d postgres -W

**Example pg\_hba.conf Configuration**

Here’s an example of what your pg\_hba.conf file might look like after adding the new entry:

plaintext

Copy code

# PostgreSQL Client Authentication Configuration File

# ===================================================

# Refer to the "Client Authentication" section in the PostgreSQL

# documentation for a complete description of this file.

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all peer

# IPv4 local connections:

host all all 127.0.0.1/32 md5

# IPv6 local connections:

host all all ::1/128 md5

# Allow replication connections from localhost, by a user with the

# replication privilege.

local replication all peer

host replication all 127.0.0.1/32 md5

host replication all ::1/128 md5

# Allow SSL connections from 192.168.100.1

hostssl all all 192.168.100.1/32 md5

By following these steps, you should be able to resolve the no pg\_hba.conf entry error and allow SSL connections from the specified host to your PostgreSQL server.

4o

how to change the postgres user password in postgresql client

ChatGPT

To change the PostgreSQL user password using the PostgreSQL client (psql), follow these steps:

**Step-by-Step Instructions**

1. **Connect to PostgreSQL:** Open a terminal and connect to your PostgreSQL database using the psql command. Replace your\_db with your database name, and your\_user with your PostgreSQL user.

sh

Copy code

psql -h localhost -U your\_user -d your\_db

You will be prompted to enter the password for your\_user.

1. **Change the Password:** Once connected, you can change the password for any PostgreSQL user, including the postgres user, using the ALTER USER command. Replace new\_password with the desired password.

sql

Copy code

ALTER USER postgres WITH PASSWORD 'new\_password';

1. **Exit the psql Client:** After changing the password, you can exit the psql client by typing:

sql

Copy code

\q

**Example**

Here's a complete example of changing the password for the postgres user:

1. Connect to the PostgreSQL database as the postgres user:

sh

Copy code

psql -h localhost -U postgres -d postgres

1. Once connected, change the password:

sql

Copy code

ALTER USER postgres WITH PASSWORD 'new\_secure\_password';