LOGIN

sudo -u postgres psql

or

sudo -i -u postgres

or

psql

SHOW data\_directory;

SHOW config\_file;

SHOW hba\_file;

psql -h 174.1.0.7 -p 5432 -U readme postgres

=========================================

Objectives

Set up two Compute Engine instances running Postgres.

Create a new table for a guestbook app.

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.

================================================================================================

Step 1: Login TO Postgres

------------------

Run PSQL as user postgres and access the database named postgres:

sudo -u postgres psql postgres

CREATE TABLE guestbook (visitor\_email text, vistor\_id serial, date timestamp, message text);

INSERT INTO guestbook (visitor\_email, date, message) VALUES ( 'jim@gmail.com', current\_date, 'This is a test.');

create role replication with REPLICATION password 'Hari@123' login;

Step 2: Create a user for replication

-----------------------------

create role replication with REPLICATION password 'Open@123' login;

OR

sudo -u postgres createuser -U postgres replication -P -c 5 --replication

This command performs the following actions:

sudo -u postgres ensures that the createuser command runs as the user postgres.

Otherwise, Postgres will try to run the command by using peer authentication, which means the command will run under your Ubuntu user account.

This account probably doesn't have the right privileges to create the new user, which would cause an error.

The -U option tells the createuser command to use the user postgres to create the new user.

The name of the new user is replication. You'll enter that username in the configuration files.

-P prompts you for the new user's password.

-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 replication.

Step 2: Create the archive directory (IN DATADIR)

-----------------------------

mkdir /dblog/pgsql/archive

chown -R postgres:postgres /dblog/pgsql/archive

Step 2: Edit Config files - primary

-----------------------------

--pg\_hba.conf

vi /var/lib/pgsql/14/data/pg\_hba.conf

# Allow replication connections

host replication replication [standby-IP]/32 md5

Save and close the file.

--postgresql.conf

vi /var/lib/pgsql/14/data/postgresql.conf

port=5432

wal\_level = hot\_standby

archive\_mode = on

archive\_command = 'test ! -f /dblog/pgsql/archive/%f && cp %p /dblog/pgsql/archive/%f'

max\_wal\_senders = 3

cat /var/lib/pgsql/14/data/postgresql.conf |grep -i listen\_add

cat /var/lib/pgsql/14/data/postgresql.conf |grep -i data\_dir

cat /var/lib/pgsql/14/data/postgresql.conf |grep -i wal\_level

cat /var/lib/pgsql/14/data/postgresql.conf |grep -i max\_wal\_

cat /var/lib/pgsql/14/data/postgresql.conf |grep -i wal\_keep\_

cat /var/lib/pgsql/14/data/postgresql.conf |grep -i archive\_mode

Step 2: Restart the primary server

-----------------------------

service postgresql-14 restart

service postgresql-14 status

STANDBY

=======

Step : Stop postgres AND clear data

-----------------------------

service postgresql-14 stop

mv /dbdata/pgsql/data /dbdata/pgsql/data\_old

OR

cd /dbdata/pgsql/data/

rm -rvf \*

Step 2: Run the backup utility

-----------------------------

sudo -u postgres pg\_basebackup -h 172.162.7.42 --port 4554 -D /data01/ postgresql -U replication -v -P --xlog-method=stream

or

pg\_basebackup -h 174.1.0.6 -D /dbdata/pgsql/data/ -U replication -v -P --xlog-method=stream

or

pg\_basebackup -h 174.1.0.6 -U replication -D /dbdata/pgsql/data/

or

pg\_basebackup -h 174.1.0.6 -U replication --checkpoint=fast \ -D /dbdata/pgsql/data/ -R --slot=some\_name -C

pg\_basebackup -h 174.1.0.6 -U replication -D /dbdata/pgsql/data/ --checkpoint=fast -R --slot=some\_name -C

Node 3

pg\_basebackup -h 174.1.0.6 -U replication -D /dbdata/pgsql/data/ --checkpoint=fast -R --slot=some\_name2 -C

Step 2: Edit Config files - Standby

-----------------------------

--postgresql.conf

vi /var/lib/pgsql/15/data/postgresql.conf

hot\_standby = ON

wal\_level = hot\_standby

10000

Step 2: Create the recovery conf (IN DATADIR)

-----------------------------

When you implement a server in Hot Standby mode, you must supply a configuration file that contains the settings that will be used in the event of data recovery.

This file is named recovery.conf. To add this file to the standby server, follow these steps:

cp -avr ../../usr/share/postgresql/9.3/recovery.conf.sample /../../var/lib/postgresql/9.3/main/recovery.conf

Edit the recovery file:

nano /../../var/lib/postgresql/9.3/main/recovery.conf

In the STANDBY SERVER PARAMETERS section, change the standby mode:

standby\_mode = on

--(Optional) Recovery File

vi /dbdata/pgsql/data/recovery.conf

standby\_mode = 'on'

primary\_conninfo = 'host=174.1.0.6 port=5432 user=replication password=Open@123'

trigger\_file= '/var/lib/pgsql/trigger\_file'

--(Optional) Trigger File

trigger\_file = '/tmp/postgresql.trigger.5432'

The trigger\_file path that you specify is the location where you can add a file when you want the system to fail over to the standby server. The presence of the file "triggers" the failover. Alternatively, you can use the pg\_ctl promote command to trigger failover.

Step 2: Start the standby server

-----------------------

chown -R postgres:postgres /dbdata/pgsql/data

sudo chmod -R 700 /dbdata/pgsql/data/

service postgresql-14 start

cd /dbdata/pgsql/data/log

2021-11-12 03:47:23.794 UTC [158211] LOG: starting PostgreSQL 14.0 on x86\_64-pc-linux-gnu, compiled by gcc (GCC) 8.4.1 20200928 (Red Hat 8.4.1-1), 64-bit

2021-11-12 03:47:23.794 UTC [158211] LOG: listening on IPv4 address "0.0.0.0", port 5432

2021-11-12 03:47:23.794 UTC [158211] LOG: listening on IPv6 address "::", port 5432

2021-11-12 03:47:23.800 UTC [158211] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"

2021-11-12 03:47:23.805 UTC [158211] LOG: listening on Unix socket "/tmp/.s.PGSQL.5432"

2021-11-12 03:47:23.812 UTC [158213] LOG: database system was interrupted; last known up at 2021-11-12 03:40:56 UTC

2021-11-12 03:47:23.838 UTC [158213] LOG: entering standby mode

2021-11-12 03:47:23.844 UTC [158213] LOG: redo starts at 0/A000028

2021-11-12 03:47:23.846 UTC [158213] LOG: consistent recovery state reached at 0/A000100

2021-11-12 03:47:23.847 UTC [158211] LOG: database system is ready to accept read-only connections

2021-11-12 03:47:23.872 UTC [158217] LOG: started streaming WAL from primary at 0/B000000 on timeline 1

Step 2: On the standby server, start PSQL:

-------------------------

sudo -u postgres psql postgres

At the PSQL prompt, enter the following query:

grant select on guestbook to readme;

select \* from guestbook;

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

On the primary server, start PSQL:

sudo -u postgres psql postgres

select \* from guestbook;

INSERT INTO guestbook (visitor\_email, date, message) VALUES ('jim@gmail.com', current\_date, '14-12-2021.');

INSERT INTO guestbook (visitor\_email, date, message) VALUES ('notherdummy.com', current\_date, 'Now we are replicating.');

Switch back to the standby server terminal and repeat the query for all rows of the guestbook:

select \* from guestbook;

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

Troubleshooting

==============

After completing all the steps, if you're not seeing the data replicate, you might have missed a step or some small detail. Common mistakes include:

Leaving a setting commented out.

Forgetting to replace placeholder text in a setting or command. For example, some settings require a host IP address or a password.

Entering the wrong IP address or password.

Using a primary server setting for the standby server or vice-versa.

If you find yourself in this state, here are the steps to follow:

Look at the Postgres log on each server. These logs can contain information that will help you troubleshoot the issue.

cd /dbdata/pgsql/data/log

Shut down the standby server.

Check the standby server settings and correct them if needed.

On the standby server, rename the main folder to something new, such as main\_old\_2:

$ mv ../../var/lib/postgresql/9.3/main ../../var/lib/postgresql/9.3/main\_old\_2

On the standby server, run pgbasebackup again to synchronize the data. Substitute [primary-IP] with your primary server's external IP address:

$ sudo -u postgres pg\_basebackup -h [primary-IP] -D /var/lib/postgresql/9.3/main -U replication -v -P --xlog-method=stream

The main folder now needs a copy of recovery.conf. You can simply copy it from the folder that you renamed to main\_old\_2:

$ cp ../../var/lib/postgresql/9.3/main\_old\_2/recovery.conf ../../var/lib/postgresql/9.3/main/recovery.conf

Start the standby server.

TESTING

===========

-- create a sequence that will serve as the PK of the employees table

create sequence employees\_seq start with 1 increment by 1 no maxvalue minvalue 1 cache 1;

-- create the employees table

create table employees (

emp\_id numeric primary key default nextval('employees\_seq'::regclass),

first\_name text not null,

last\_name text not null,

birth\_year numeric not null,

birth\_month numeric not null,

birth\_dayofmonth numeric not null

);

-- insert some data into the table

insert into employees (first\_name, last\_name, birth\_year, birth\_month, birth\_dayofmonth) values ('Emily','James',1983,03,20);

insert into employees (first\_name, last\_name, birth\_year, birth\_month, birth\_dayofmonth) values ('John','Smith',1990,08,12);

===

SELECT slot\_name, database, active, pg\_xlog\_location\_diff(pg\_current\_xlog\_insert\_location(), restart\_lsn) AS retained\_bytes FROM pg\_replication\_slots;

slot\_name | database | active | retained\_bytes

--------------+----------+--------+----------------

slave01\_slot | | f | 201326688

(1 row)

and after i started the slave :

SELECT slot\_name, database, active,pg\_xlog\_location\_diff(pg\_current\_xlog\_insert\_location(), restart\_lsn) AS retained\_bytes FROM pg\_replication\_slots;

slot\_name | database | active | retained\_bytes

--------------+----------+--------+----------------

slave01\_slot | | t | 0

(1 row)

the second monitoring statement is :

postgres=# SELECT pg\_xlog\_location\_diff(pg\_current\_xlog\_insert\_location(), flush\_location) AS lag\_bytes, pid, application\_name

FROM pg\_stat\_replication;

lag\_bytes | pid | application\_name

-----------+-----+------------------

0 | 855 | walreceiver

This will show you how many bytes your slave is behind.

(both statements are to be run on Master unless cascading replicas are in place)

select \* from pg\_stat\_replication;

select \* from guestbook;

INSERT INTO guestbook (visitor\_email, date, message) VALUES ('jim@gmail.com', current\_date, 'Now we are replicating.');

INSERT INTO guestbook (visitor\_email, date, message) VALUES ('notherdummy.com', current\_date, 'Now we are replicating.');

Switch back to the standby server terminal and repeat the query for all rows of the guestbook:

ALTER TABLE public.myname\_record OWNER TO rndb;

ALTER TABLE public.myname\_record OWNER TO rndb;

for tbl in `psql -qAt -c "select tablename from pg\_tables where schemaname = 'public';" YOUR\_DB` ; do psql -c "alter table \"$tbl\" owner to NEW\_OWNER" YOUR\_DB ; done

for tbl in `psql -qAt -c "select sequence\_name from information\_schema.sequences where sequence\_schema = 'public';" YOUR\_DB` ; do psql -c "alter sequence \"$tbl\" owner to NEW\_OWNER" YOUR\_DB ; done

for tbl in `psql -qAt -c "select table\_name from information\_schema.views where table\_schema = 'public';" YOUR\_DB` ; do psql -c "alter view \"$tbl\" owner to NEW\_OWNER" YOUR\_DB ; done

SELECT 'ALTER TABLE '|| schemaname || '."' || tablename ||'" OWNER TO my\_new\_owner;'

FROM pg\_tables WHERE NOT schemaname IN ('pg\_catalog', 'information\_schema')

ORDER BY schemaname, tablename;

SELECT 'ALTER SEQUENCE '|| sequence\_schema || '."' || sequence\_name ||'" OWNER TO my\_new\_owner;'

FROM information\_schema.sequences WHERE NOT sequence\_schema IN ('pg\_catalog', 'information\_schema')

ORDER BY sequence\_schema, sequence\_name;

SELECT 'ALTER VIEW '|| table\_schema || '."' || table\_name ||'" OWNER TO my\_new\_owner;'

FROM information\_schema.views WHERE NOT table\_schema IN ('pg\_catalog', 'information\_schema')

ORDER BY table\_schema, table\_name;

SELECT 'ALTER TABLE '|| oid::regclass::text ||' OWNER TO my\_new\_owner;'

FROM pg\_class WHERE relkind = 'm'

ORDER BY oid;