# Migrating a Postgres cluster using the Streaming Replication

Dear readers,

First of all, thank you for your time.

In this document, I'm explaining a test case in which I have been asked to migrate a Postgres 12.3 cluster.

- The cluster was running on RHEL 7 with 1 application database.
- It was hosted on Google Cloud but I have recreated an exact scenario where both, the source (primary) and the target (standby) clusters will be running on a VM.
- The source cluster was not in an archive mode as it was a non-production environment,
- To reduce the downtime, I decided to use the streaming replication and it resulted in less than a minute of downtime (just for some parameter changes).
- Steps 1 to 7 are kind of preparation steps where the connectivity will be made between both of the VMs where the primary and the secondary servers are running. On the cloud, there will be some extra steps like opening a virtual cloud level firewall to facilitate both VMs to communicate via the cluster port 5432.
- I'm using some alias like ls, lk which are shorthands for ls -ltrh, ps -ef | grep, etc.
- I have a few scripts to install supported packages of the Postgres, install the Postgres and initialize the cluster (pg_main.sh).
- The green color text is for the primary cluster while the yellow color cluster specifies the standby cluster terminal.

So let's begin.

1. I have a CentOS 7 VirtualBox VM which I imported and changed it's IP address, hostname

```
[root@pgserver1 ~]# hostname
pgserver1.perfdba.info

[root@pgserver1 ~]# nmcli general hostname
pgserver1.perfdba.info

[root@pgserver1 ~]# nmcli general hostname pgserver2.perfdba.info

[root@pgserver1 ~]# service systemd-hostnamed restart
Redirecting to /bin/systemctl restart systemd-hostnamed.service

[root@pgserver1 ~]# hostname
pgserver2.perfdba.info
```

   Once done just re-open another SSH session for server 2.

2. Open the port 5432 at firewall level for **both the servers**

```
[cent@pgserver2 pg]$ sudo firewall-cmd --permanent --add-port=5432/tcp
success
[cent@pgserver2 pg]$ sudo firewall-cmd --reload
success
[cent@pgserver2 pg]$ sudo firewall-cmd --list-ports
5432/tcp
```

```
[cent@pgserver1 ~]$ sudo firewall-cmd --permanent --add-port=5432/tcp
success
[cent@pgserver1 ~]$ sudo firewall-cmd --reload
success
[cent@pgserver1 ~]$ sudo firewall-cmd --list-ports
5432/tcp
```

```
[cent@pgserver2 pg]$ sudo firewall-cmd --permanent --add-port=5432/tcp
success
[cent@pgserver2 pg]$ sudo firewall-cmd --reload
success
[cent@pgserver2 pg]$ sudo firewall-cmd --list-ports
5432/tcp
```

3. Add host entries of each other to **both server's** /etc/hosts and verify

```
[cent@pgserver2 pg]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

# Primary and standby hosts
192.168.0.145 pgserver1.perfdba.info pgserver1
192.168.0.146 pgserver2.perfdba.info pgserver2
```

```
[root@pgserver1 ~]# ping -c 2 pgserver1
PING pgserver1.perfdba.info (192.168.0.145) 56(84) bytes of data.
64 bytes from pgserver1.perfdba.info (192.168.0.145): icmp_seq=1 ttl=64 time=0.036 ms
64 bytes from pgserver1.perfdba.info (192.168.0.145): icmp_seq=2 ttl=64 time=0.046 ms

--- pgserver1.perfdba.info ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.036/0.041/0.046/0.005 ms
[root@pgserver1 ~]# ping -c 2 pgserver2
PING pgserver2.perfdba.info (192.168.0.146) 56(84) bytes of data.
64 bytes from pgserver2.perfdba.info (192.168.0.146): icmp_seq=1 ttl=64 time=0.768 ms
64 bytes from pgserver2.perfdba.info (192.168.0.146): icmp_seq=2 ttl=64 time=0.332 ms

--- pgserver2.perfdba.info ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.332/0.550/0.768/0.218 ms
```

```
[cent@pgserver2 pg]$ ping -c 2 pgserver1
PING pgserver1.perfdba.info (192.168.0.145) 56(84) bytes of data.
64 bytes from pgserver1.perfdba.info (192.168.0.145): icmp_seq=1 ttl=64 time=0.532 ms
64 bytes from pgserver1.perfdba.info (192.168.0.145): icmp_seq=2 ttl=64 time=0.368 ms

--- pgserver1.perfdba.info ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.368/0.450/0.532/0.082 ms
[cent@pgserver2 pg]$ ping -c 2 pgserver2
PING pgserver2.perfdba.info (192.168.0.146) 56(84) bytes of data.
64 bytes from pgserver2.perfdba.info (192.168.0.146): icmp_seq=1 ttl=64 time=0.045 ms
64 bytes from pgserver2.perfdba.info (192.168.0.146): icmp_seq=2 ttl=64 time=0.035 ms

--- pgserver2.perfdba.info ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.035/0.040/0.045/0.005 ms
```

4. Transfer installation files of pg12 to the **standby server**

```
[cent@pgserver2 ~]$ mv pg mv_9.4

[cent@pgserver1 ~]$ scp -r pg cent@pgserver2:/home/cent
```

5. Transfer /etc/resolve.conf from primary to the **standby server** and verify

```
[cent@pgserver2 pg]$ sudo cp /etc/resolv.conf /etc/resolv.conf_03102020

[root@pgserver1 ~]# scp /etc/resolv.conf root@pgserver2:/etc/
```

```
[cent@pgserver2 pg]$ ping -c 2 www.google.com
PING www.google.com (142.250.67.196) 56(84) bytes of data.
64 bytes from bom12s08-in-f4.1e100.net (142.250.67.196): icmp_seq=1 ttl=119 time=5.55 ms
64 bytes from bom12s08-in-f4.1e100.net (142.250.67.196): icmp_seq=2 ttl=119 time=6.46 ms

--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 5.551/6.008/6.466/0.464 ms
```

6. Install, initialize Postgres 12 cluster on the **standby server**

```
[cent@pgserver2 pg]$ sudo sh pg_main.sh
```

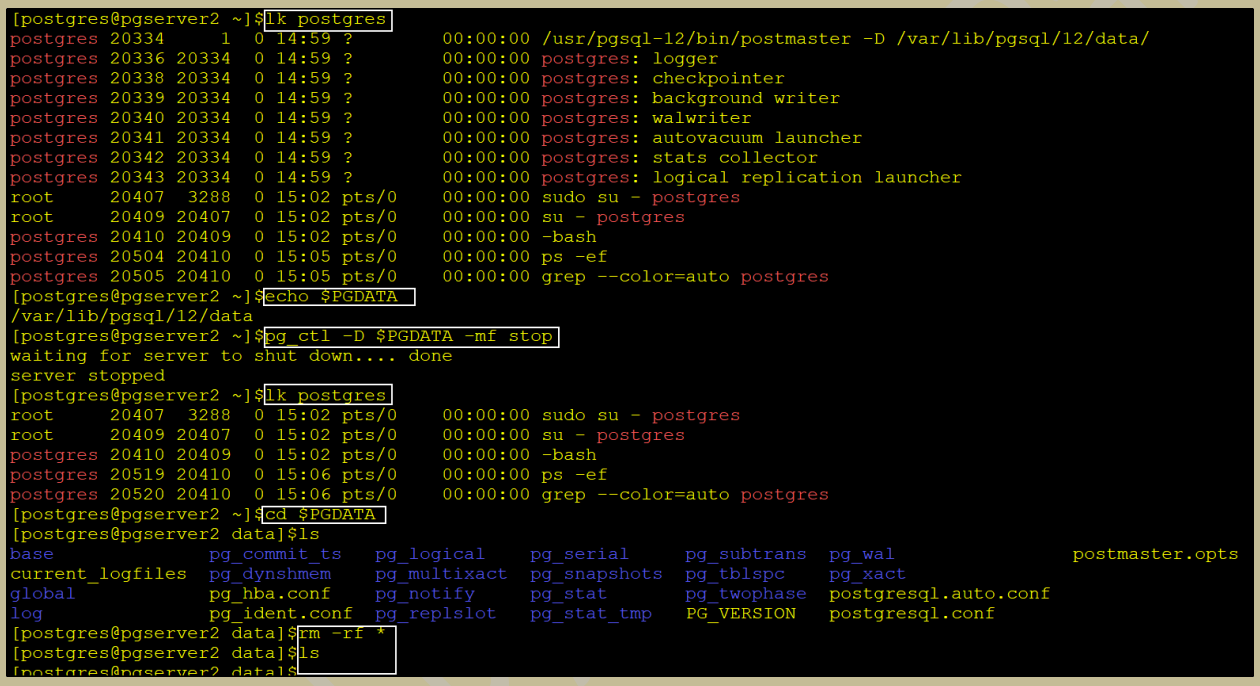
7. Transfer /var/lib/pgsql/env.profile and /var/lib/pgsql/.bash_profile **from pgserver1 to pgserver2** at location /var/lib/pgsql

```
[root@pgserver1 ~]# scp /var/lib/pgsql/env.profile /var/lib/pgsql/.bash_profile
root@pgserver2:/var/lib/pgsql
```

8. Login to the **standby server**, stop the running postgres cluster and remove the contents of the $PGDATA directory. During the cloning process, this $PGDATA directory will be populated with the

primary cluster's data.

```
[postgres@pgserver2 ~]$lk postgres
[postgres@pgserver2 ~]$echo $PGDATA
[postgres@pgserver2 ~]$pg_ctl -D $PGDATA -mf stop
[postgres@pgserver2 ~]$cd $PGDATA
[postgres@pgserver2 data]$ls
[postgres@pgserver2 data]$rm -rf *
[postgres@pgserver2 data]$ls
```




9. On the **primary server**, allow the server to accept connections from any servers, create a replication slot and then restart the cluster.

   Creating a replication slot is not mandatory but I'm creating it as a best practice. Once we open the standby cluster in read-only mode, it will help in the case where the primary cluster is having too many transactions and many WALs are getting generated, In such cases, the standby cluster may fall behind. Also, in my example, the primary cluster is not in archive mode so I don't want the primary cluster to remove/recycle/overwrite WALs before they have been received by the standby cluster.

```
[postgres@pgserver1 data]$cp postgresql.auto.conf postgresql.auto.conf_03102020
[postgres@pgserver1 data]$psql -c "ALTER SYSTEM SET listen_addresses TO '*'";
ALTER SYSTEM
[postgres@pgserver1 data]$psql -c "select * from
pg_create_physical_replication_slot('replica');"
[postgres@pgserver1 data]$pg_ctl -D $PGDATA -mf restart
[postgres@pgserver1 data]$psql -c "select * from pg_replication_slots;"
```

```
[postgres@pgserver1 data]$cp postgresql.auto.conf postgresql.auto.conf_03102020
[postgres@pgserver1 data]$psql -c "ALTER SYSTEM SET listen_addresses TO '*'";
ALTER SYSTEM
[postgres@pgserver1 data]$psql -c "select * from pg_create_physical_replication_slot('replica');"
 slot_name | lsn
-----------+-----
 replica   |
(1 row)

[postgres@pgserver1 data]$pg_ctl -D $PGDATA -mf restart
waiting for server to shut down.... done
server stopped
waiting for server to start....2020-10-03 15:30:44.685 IST [4195] LOG:  starting PostgreSQL 12.4 on x86_64-pc-linux-gnu, compiled by gc
c (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39), 64-bit
2020-10-03 15:30:44.690 IST [4195] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2020-10-03 15:30:44.690 IST [4195] LOG:  listening on IPv6 address "::", port 5432
2020-10-03 15:30:44.694 IST [4195] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2020-10-03 15:30:44.700 IST [4195] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2020-10-03 15:30:44.715 IST [4195] LOG:  redirecting log output to logging collector process
2020-10-03 15:30:44.715 IST [4195] HINT:  Future log output will appear in directory "log".
 done
server started
[postgres@pgserver1 data]$psql -c "select * from pg_replication_slots;"
 slot_name | plugin | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_f
lush_lsn
-----------+--------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+------------
---------
 replica   |        | physical  |        |          | f         | f      |            |      |              |             |
(1 row)

[postgres@pgserver1 data]$
```

We can see that the slot is not active as there's no standby which is connected to this slot.

10. On the **primary server** create a user for the replication and add replication rule in the pg_hba.conf

```
[postgres@pgserver1 data]$cp $PGDATA/pg_hba.conf $PGDATA/pg_hba.conf_03102020
[postgres@pgserver1 data]$echo "host replication replicator 192.168.0.146/32 md5" >>
$PGDATA/pg_hba.conf
[postgres@pgserver1 data]$psql -c "CREATE USER replicator WITH REPLICATION
ENCRYPTED PASSWORD 'Oracle123';"
CREATE ROLE
[postgres@pgserver1 data]$pg_ctl -D $PGDATA reload
server signaled
```

```
[postgres@pgserver1 data]$cp $PGDATA/pg_hba.conf $PGDATA/pg_hba.conf_03102020
[postgres@pgserver1 data]$echo "host replication replicator 192.168.0.146/32 md5" >> $PGDATA/pg_hba.conf
[postgres@pgserver1 data]$psql -c "CREATE USER replicator WITH REPLICATION ENCRYPTED PASSWORD 'Oracle123';"
CREATE ROLE
[postgres@pgserver1 data]$pg_ctl -D $PGDATA reload
server signaled
```

11. Using pg_basebackup restore the primary server's $PGDATA directory at standby server. At the **standby server**, run the below command:

```
pg_basebackup -h pgserver1 -U replicator -p 5432 -D $PGDATA -Fp -Xs -P -R
```

- pg_basebackup utility will copy the primary cluster to the -D directory path of the standby server.
- -Fp suggests that $PGDATA of the primary cluster will be copied in plain/non-compressed format at the standby cluster.
- Xs is for the Streaming type of WAL transfer. With this flag, pg_asebackup will open 2 connections, 1 for copying $PGDATA and the other is for copying WALs of the primary. With the

streaming option, as soon as the new WAL gets generated, it will be transferred to the standby site. It's the default option but to highlight it to the readers, I have put it in the command.

- -P will show the progress like how many KBs have been transferred so far out of total KBs
- -R will create a standby.signal file. Which is a 0 byte file but it's existence suggests that the current cluster will be started as a standby cluster.
- Sometimes if the tablespaces are residing outside the $PGDATA at the primary site and in the standby site we want them to reside at the different path (e.g. $PGDATA) then we can do so with the mapping flag:
  pg_basebackup -h pgserver1 -U replicator -p 5432 -D $PGDATA --tablespace-mapping=/old_path/focus/data=/new_path/data -Fp -Xs -P -R

```
[postgres@pgserver2 ~]$ls $PGDATA
[postgres@pgserver2 ~]$pg_basebackup -h pgserver1 -U replicator -p 5432 -D $PGDATA -Fp -Xs -P -R
Password:
25325/25325 kB (100%), 1/1 tablespace
[postgres@pgserver2 ~]$ls $PGDATA
backup_label      pg_commit_ts            pg_ident.conf  pg_serial     pg_tblspc    postgresql.auto.conf
base              pg_dynshmem             pg_logical     pg_snapshots  pg_twophase  postgresql.auto.conf_03102020
current_logfiles  pg_hba.conf             pg_multixact   pg_stat       PG_VERSION   postgresql.conf
global            pg_hba.conf_03102020    pg_notify      pg_stat_tmp   pg_wal       standby.signal
log               pg_hba.conf_03102020_1  pg_replslot    pg_subtrans   pg_xact
[postgres@pgserver2 ~]$_
```

12. In the postgresql.conf of the **standby** add slot info:

```
[postgres@pgserver2 data]$grep primary_slot_name postgresql.conf
primary_slot_name = 'replica'          # replication slot on sending server
```

13. Start the **Standby** cluster

```
[postgres@pgserver2 data]$pg_ctl -D $PGDATA start
```

```
[postgres@pgserver2 data]$pg_ctl -D $PGDATA start
waiting for server to start....2020-10-03 16:29:55.168 IST [21580] LOG:  starting PostgreSQL 12.4 on x86_64-pc-linux-gnu, compiled by g
cc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39), 64-bit
2020-10-03 16:29:55.172 IST [21580] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2020-10-03 16:29:55.172 IST [21580] LOG:  listening on IPv6 address "::", port 5432
2020-10-03 16:29:55.185 IST [21580] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2020-10-03 16:29:55.201 IST [21580] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2020-10-03 16:29:55.241 IST [21580] LOG:  redirecting log output to logging collector process
2020-10-03 16:29:55.241 IST [21580] HINT:  Future log output will appear in directory "log".
 done
server started
[postgres@pgserver2 data]$
```

In the log of standby below entries were seen:

```
[postgres@pgserver2 log]$tail -10f postgresql-Sat.log
2020-10-03 15:30:44.720 IST [4197] LOG:  database system was shut down at 2020-10-03 15:30:44 IST
2020-10-03 15:30:44.726 IST [4195] LOG:  database system is ready to accept connections
2020-10-03 15:43:43.302 IST [4195] LOG:  received SIGHUP, reloading configuration files
2020-10-03 16:15:32.334 IST [4791] LOG:  invalid length of startup packet
2020-10-03 16:29:55.248 IST [21582] LOG:  database system was interrupted; last known up at 2020-10-03 16:17:24 IST
2020-10-03 16:29:55.628 IST [21582] LOG:  entering standby mode
2020-10-03 16:29:55.631 IST [21582] LOG:  redo starts at 0/2000028
2020-10-03 16:29:55.634 IST [21582] LOG:  consistent recovery state reached at 0/2000138
2020-10-03 16:29:55.634 IST [21580] LOG:  database system is ready to accept read only connections
2020-10-03 16:29:55.647 IST [21586] LOG:  started streaming WAL from primary at 0/3000000 on timeline 1
```

After the standby has been started, the slot was occupied at the primary cluster:

```
[postgres@pgserver1 ~]$psql -c "select * from pg_replication_slots;"
 slot_name | plugin | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_f
lush_lsn
-----------+--------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+------------
---------
 replica   |        | physical  |        |          | f         | t      |       4943 |      |              | 0/3000148   |
(1 row)
```

14. Verification:
    At the **primary** cluster, create a DB dvdrental and import the data in it. Once done, connect to the standby cluster and verify if we can see the new DB and it's tables.

```
[postgres@pgserver1 tmp]$chmod +x dvdrental.tar
[postgres@pgserver1 tmp]$lt dvdrental.tar
-rwxr-xr-x. 1 postgres postgres 2.8M May 12  2019 dvdrental.tar
[postgres@pgserver1 tmp]$createdb -O postgres -U postgres dvdrental
[postgres@pgserver1 tmp]$pg_restore -d dvdrental -Ft dvdrental.tar
```

```
[postgres@pgserver2 data]$psql -c "\l"
[postgres@pgserver2 data]$psql -d dvdrental -c "select count(*) from film;"
```

```
[postgres@pgserver1 tmp]$chmod +x dvdrental.tar
[postgres@pgserver1 tmp]$lt dvdrental.tar
-rwxr-xr-x. 1 postgres postgres 2.8M May 12  2019 dvdrental.tar
[postgres@pgserver1 tmp]$createdb -O postgres -U postgres dvdrental
[postgres@pgserver1 tmp]$pg_restore -d dvdrental -Ft dvdrental.tar
[postgres@pgserver1 tmp]$
```

```
[postgres@pgserver2 data]$psql -c "\l"
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 dvdrental | 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)

[postgres@pgserver2 data]$psql -d dvdrental -c "select count(*) from film;"
 count
-------
  1000
(1 row)
```

However, only read-only (SELECT) queries can be supported at the time by the **standby** cluster.

```
[postgres@pgserver2 data]$psql -d dvdrental
dvdrental=# insert into actor (first_name,last_name) values ('Amol', 'Palav');
```

```
[postgres@pgserver2 data]$psql -d dvdrental
psql (12.4)
Type "help" for help.

dvdrental=# insert into actor (first name,last name) values ('Amol', 'Palav');
ERROR:  cannot execute INSERT in a read-only transaction
dvdrental=#
```

15. Switchover
    Stop the **primary** cluster, promote the **standby** cluster, verify the logs and check if we can perform transactions in the standby.

```
[postgres@pgserver1 tmp]$pg_ctl -D $PGDATA -mf stop
```

```
[postgres@pgserver2 data]$pg_ctl promote -D $PGDATA
[postgres@pgserver2 data]$psql -d dvdrental -c "insert into actor (first_name,last_name)
values ('Amol', 'Palav');"
```

```
[postgres@pgserver1 tmp]$pg_ctl -D $PGDATA -mf stop
waiting for server to shut down.... done
server stopped
[postgres@pgserver1 tmp]$
```

```
[postgres@pgserver2 data]$pg_ctl promote -D $PGDATA
waiting for server to promote.... done
server promoted
[postgres@pgserver2 data]$
```

```
[postgres@pgserver2 data]$psql -d dvdrental -c "insert into actor (first_name,last_name) values ('Amol', 'Palav');"
INSERT 0 1
[postgres@pgserver2 data]$psql -d dvdrental -c "select * from actor where first_name='Amol';"
 actor_id | first_name | last_name |        last_update
----------+------------+-----------+----------------------------
      201 | Amol       | Palav     | 2020-10-03 17:00:04.567954
(1 row)
```

```
2020-10-03 16:58:53.615 IST [21586] LOG:  replication terminated by primary server
2020-10-03 16:58:53.615 IST [21586] DETAIL:  End of WAL reached on timeline 1 at 0/360C578.
2020-10-03 16:58:53.615 IST [21586] FATAL:  could not send end-of-streaming message to primary: no COPY in progress
2020-10-03 16:58:53.617 IST [21582] LOG:  invalid record length at 0/360C578: wanted 24, got 0
2020-10-03 16:58:53.627 IST [21878] FATAL:  could not connect to the primary server: could not connect to server: Connection refused
                Is the server running on host "pgserver1" (192.168.0.145) and accepting
                TCP/IP connections on port 5432?
2020-10-03 16:58:58.642 IST [21879] FATAL:  could not connect to the primary server: could not connect to server: Connection refused
                Is the server running on host "pgserver1" (192.168.0.145) and accepting
                TCP/IP connections on port 5432?
2020-10-03 16:59:03.645 IST [21880] FATAL:  could not connect to the primary server: could not connect to server: Connection refused
                Is the server running on host "pgserver1" (192.168.0.145) and accepting
                TCP/IP connections on port 5432?
2020-10-03 16:59:08.646 IST [21881] FATAL:  could not connect to the primary server: could not connect to server: Connection refused
                Is the server running on host "pgserver1" (192.168.0.145) and accepting
                TCP/IP connections on port 5432?
2020-10-03 16:59:10.819 IST [21582] LOG:  received promote request
2020-10-03 16:59:10.819 IST [21582] LOG:  redo done at 0/360C500
2020-10-03 16:59:10.819 IST [21582] LOG:  last completed transaction was at log time 2020-10-03 16:43:45.760545+05:30
2020-10-03 16:59:10.824 IST [21582] LOG:  selected new timeline ID: 2
2020-10-03 16:59:11.323 IST [21582] LOG:  archive recovery complete
2020-10-03 16:59:11.360 IST [21580] LOG:  database system is ready to accept connections
```

- In the logs, we can see that a new timeline id is generated for the WALs
- No replication slots get created for the promoted standby (i.e. the new primary)

```
[postgres@pgserver2 data]$psql -c "select * from pg_replication_slots;"
 slot_name | plugin | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_f
lush_lsn
-----------+--------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+------------
---------
(0 rows)
```