**Steps to setup Streaming Replication for High Availability with EDB**

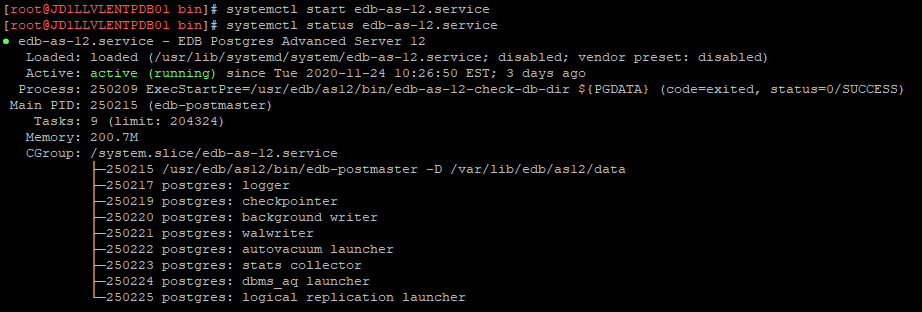
In the following steps, the Master server is: **10.40.116.52** and the Standby server is: **10.40.117.16**

**Step 1 :**

Install and start EDB services on both Master and Slave nodes :

## As root, initialize and start EDB12 on the Master and Slave

* **PGSETUP\_INITDB\_OPTIONS="-D <data directory> -X <Log Directory>" /usr/edb/as12/bin/edb-as-12-setup initdb**
* **systemctl start edb-as-12.service**
* **systemctl status edb-as-12.service**

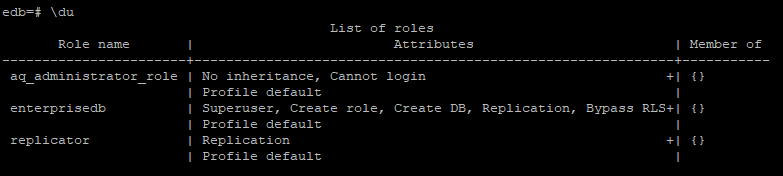


**Step 2 :**

Create a User for replication in the Master. It is discouraged to use superuser enterprisedb in order to setup replication, though it works.

edb=# CREATE USER replicator WITH REPLICATION ENCRYPTED PASSWORD '<password>';

CREATE ROLE



**Step 3 :**

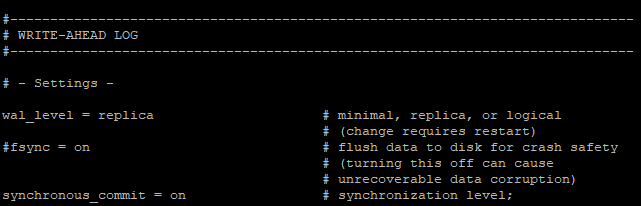
As the root user stop the EDB services and we need to add the following entry to pg\_hba.conf file and set certain parameters in postgresql.conf on the master and reload PostgreSQL.

Following are the parameters to be set in postgresql.conf on the master

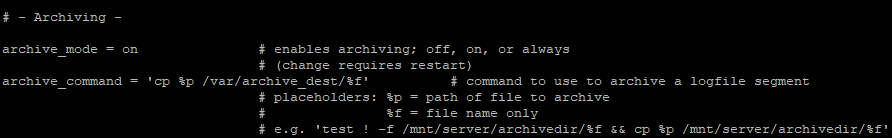
* max\_wal\_senders - This parameter specifies the maximum number of concurrent connections allowed on the sender from standby(s) or tools like pg\_basebackup clients for streaming. The default value is 10. Set this parameter based on number standby(s) and pg\_basebackup like tool going to stream the data from a sender. Based on our diagram, we don’t need to make any changes because the default value is giving us enough room for our two standby(s)
* max\_replication\_slots - This parameter specifies the maximum number of replication slots that a sender can support, and the number of standby(s)/tools can be allowed to use replication slots to stream data. The default value is 10. Set this parameter based on the number of standby(s)/tools is going to use for streaming.
* wal\_sender\_timeout - With this parameter, a sender/master decides on terminating replication connections that are inactive for longer than the amount of time mentioned for the parameter. This parameter is useful to detect a standby crash or network outage. The default value is 60 seconds. A value of zero disables the timeout mechanism.
* wal\_level = replica - With this parameter settings, the master server decides on how much information is written to the WAL. The default value of this parameter is replica. With this setting, the sender writes enough data to enable streaming replication.
* wal\_log\_hints = on - This parameter makes a master server write each disk page to WAL with the first modification with hint bits. With this setting, you can use pg\_rewind to make a master be standby of a new master after the failover.
* synchronous\_standby\_names - This parameter specifies standby(s) that can support synchronous replication for very high availability and protection against data loss. Synchronous replication in Postgres gives the ability to confirm that all changes made on a master have been transferred to standby(s) servers.
* archive\_mode = on enables archiving (change requires restart)
* archive\_command = command to use to archive a logfile segment

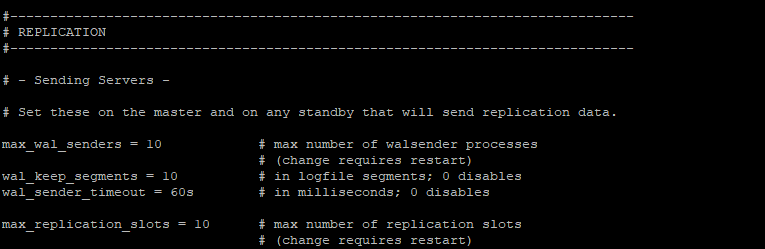
'cp %p<archive\_destination\_path> %f'

placeholders: %p = path of file to archive



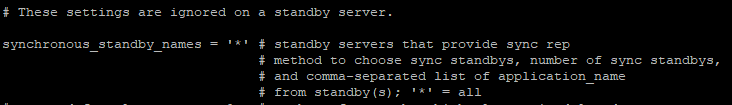






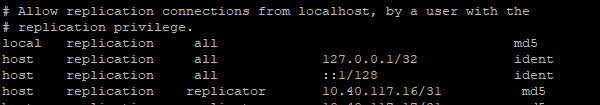






following entry to pg\_hba.conf file :

*# TYPE DATABASE USER ADDRESS<* Standby server ip: **10.40.117.16***> METHOD*



**md5**

The method md5 uses a custom less secure challenge-response mechanism. It prevents password sniffing and avoids storing passwords on the server in plain text but provides no protection if an attacker manages to steal the password hash from the server.

**Step 4 :**

As the root user reload systemd, updating the modified service scripts and Start the Services on the master node :

* **systemctl daemon-reload**
* **systemctl start edb-as-12.service**

**Step 5 :**

As the root user on the Slave node **10.40.117.16** stop the EDB services and remove the data and log directory and create new ones . Also give them correct the ownership:

cd /data

rm -rf pg\_data

mkdir pg\_data

chown enterprisedb:enterprisedb pg\_data

cd /var

rm -rf pg\_log

rm -rf archive\_dest

mkdir pg\_log

mkdir archive\_dest

chown enterprisedb:enterprisedb pg\_log

chown enterprisedb:enterprisedb archive\_dest

**Step 6 :**

As enterprisedb user use **pg\_basebackup**  to backup the data directory of the Master from the Standby. While creating the backup, you may also tell pg\_basebackup  to create the replication specific files.

**sudo su - enterprisedb**

**pg\_basebackup -h 10.40.116.52 -D /data/pg\_data -U replicator -P -v -R -X stream -C -S standby**

-h, --host=HOSTNAME database server host or socket directory

-D, --pgdata=DIRECTORY receive base backup into directory

-U, --username=NAME connect as specified database user

-P, --progress show progress information

-v, --verbose output verbose messages

-R, --write-recovery-conf

-X, --wal-method=none|fetch|stream

-C, --create-slot create replication slot

-S, --slot=SLOTNAME replication slot to use

You may use multiple approaches such as rsync or any other disk backup methods to copy the master’s data directory to the standby. But, there is an important file (standby.signal) that must exist in a standby data directory to help postgres determine its state as a standby. It is automatically created when you use the "-R" option while taking pg\_basebackup.

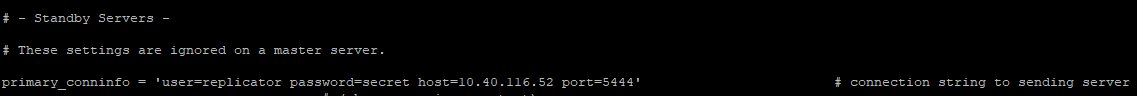
**Step 7 :**

Set following parameters in postgresql.conf on the Slave side :

Set the primary\_conninfo details

primary\_conninfo = 'host=<primary servers ip address> port=5444' user=enterprisedb password=<password>" >> postgresql.conf

recovery\_target\_timeline = latest (This parameter makes Postgres/standby(s) follow the new master in case of the failure of an old master. )





**Step 8:**

Prepare a **recovery.conf** file which will let our slave connect to the master. A file with the following content has to be created in /<DATA\_DIRECTORY>/main/recovery.conf and give all the privileges :

* mkdir main
* cd main
* vi recovery.conf

add below details in recover.conf file :

standby\_mode = 'on'

primary\_conninfo = 'user=replicator password=secret host=10.40.116.52 port=5444'

promote\_trigger\_file = '/tmp/postgresql-repl-5444.trigger'

* chmod -R g-rwx,o-rwx main
* chown -R enterprisedb:enterprisedb main

promote\_trigger\_file :

This parameter specifies a trigger file whose presence will promote a standby to become a master.

**Step 9 :**

Create the trigger file in the above mentioned location and give them pirilege.

* **touch /tmp/postgresql-repl-5444.trigger**
* **chown enterprisedb:enterprisedb postgresql-repl-5444.trigger**

**Step 10 :**

As the root user reload systemd, updating the modified service scripts and Start the Services on the master node :

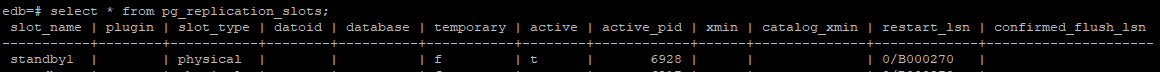
* **systemctl daemon-reload**
* **systemctl start edb-as-12.service**

**Step 11 :**

Monitoring your replication slots is something that you don't want to miss. Just collect the information from view **[pg\_replication\_slots](https://www.postgresql.org/docs/9.4/catalog-pg-replication-slots.html" \t "_blank)** in the primary/master node just like below:

**On Master :**

**edb=# select \* from pg\_replication\_slots;**



**Step 12 :**

Verify the replication between the Master and the Standby. In order to verify, run this command on the Master. In the following log, you see a lot of details of the standby and the lag between the Master and Standby.

edb=# select \* from pg\_stat\_replication;

