PostgreSQL ON LINUX

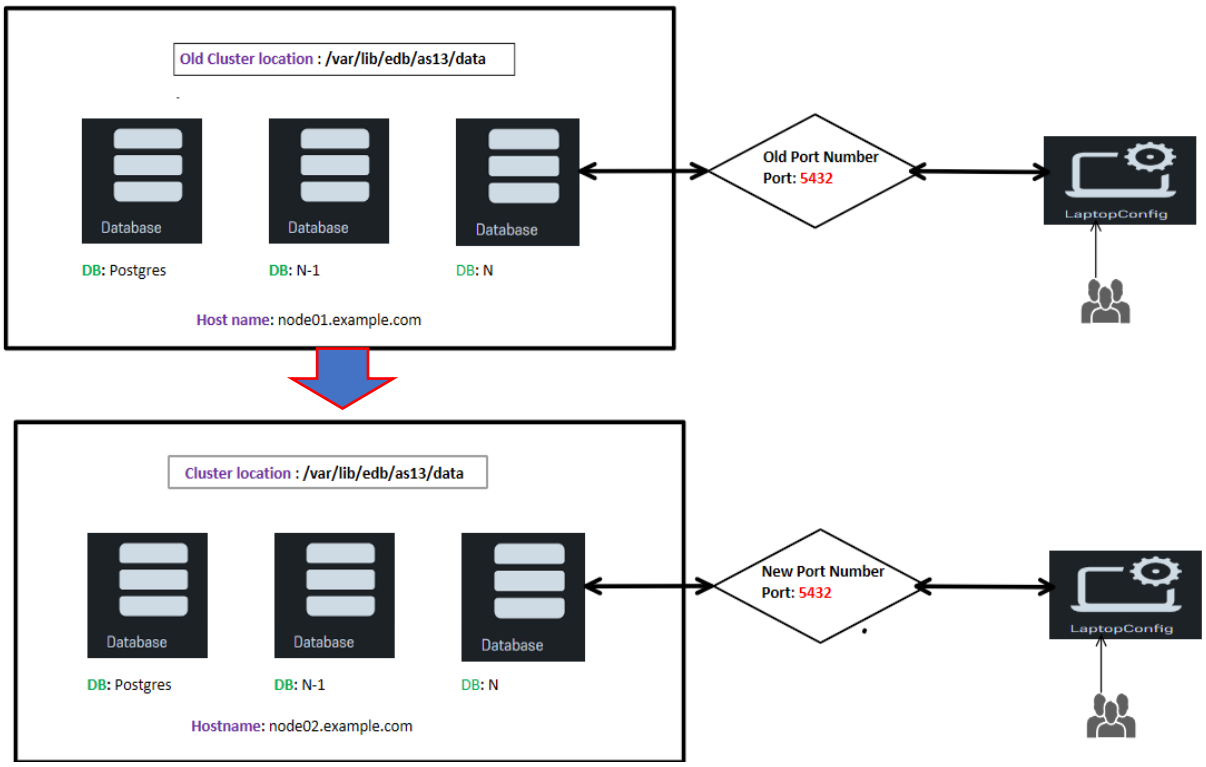
FIRST EDITION

RAVINDRA MALWAL

Topic: Clone the database using pg_basebackup in V13

PostgreSQL Database Cluster Cloning means creating a duplicate cluster of your current PostgreSQL cluster. Once you clone your cluster the duplicate cluster will be completely independent. Why do we need the cloning of the database/cluster? Sometimes we have requirements or some deployments we have to check before going to the production cluster in that case we need a cluster like the current production cluster. In this paper, I am showing how we can do the cloning of the current cluster using pg_basebackup.

Deployment diagram:



Infrastructure Details:

Primary Cluster Details	
OS version	Red Hat Enterprise Linux release 8.6 (Ootpa)
Hostname	Node01.example.com
PG Version	PostgreSQL 13
Port number	5432
Cluster location	/var/lib/edb/as13/data

Clone Cluster Details	
OS version	Red Hat Enterprise Linux release 8.6 (Ootpa)
Hostname	Node02.example.com
PG Version	PostgreSQL 13
Port number	5432
Cluster location	/var/lib/edb/as13/data

In this QuickStart, we learn:

- Validate the primary cluster details
- Change the **pg_hba.conf** and add the secondary server details to it
- Execute the pg_basebackup from secondary to create a clone
- Validation

Step-1 Validate the primary cluster details

Connect the **World** database and check the test table

```
[enterprisedb@pgedb01 ~]$ psql edb
psql (13.16.22)
Type "help" for help.

edb=# \dt
Did not find any relations.
edb=# \l
                                    List of databases
   Name    |    Owner     | Encoding |   Collate   |    Ctype    | ICU |       Access privileges
-----------+--------------+----------+-------------+-------------+-----+-------------------------------
 edb       | enterprisedb | UTF8     | en_US.UTF-8 | en_US.UTF-8 |     |
 postgres  | enterprisedb | UTF8     | en_US.UTF-8 | en_US.UTF-8 |     |
 template0 | enterprisedb | UTF8     | en_US.UTF-8 | en_US.UTF-8 |     | =c/enterprisedb              +
           |              |          |             |             |     | enterprisedb=CTc/enterprisedb
 template1 | enterprisedb | UTF8     | en_US.UTF-8 | en_US.UTF-8 |     | =c/enterprisedb              +
           |              |          |             |             |     | enterprisedb=CTc/enterprisedb
 world     | enterprisedb | UTF8     | en_US.UTF-8 | en_US.UTF-8 |     |
(5 rows)

edb=# \c world
You are now connected to database "world" as user "enterprisedb".
world=# \dt
              List of relations
 Schema |      Name       | Type  |    Owner
--------+-----------------+-------+--------------
 public | city            | table | enterprisedb
 public | country         | table | enterprisedb
 public | countrylanguage | table | enterprisedb
 public | test            | table | enterprisedb
(4 rows)

world=# select * from test;
 id | name
----+------
  1 | AAA
  2 | BBB
```

Step-2 Change the **pg_hba.conf** and add the secondary server details to it

- Add the below entry to pg_hba.conf

host replication all <Secondary host IP > trust

INFORMATION: The recommendation is not to use **trust** in production you can use **md5** and specify the password during pg_basebackup command.

Step-3 Execute the pg_basebackup from secondary to create a clone

- Check the connectivity from secondary to primary

```
[enterprisedb@pgedb02 ~]$
[enterprisedb@pgedb02 ~]$ psql -h pgedb01.example.com -d edb -U enterprisedb -p 5444
psql (13.16.22)
Type "help" for help.

edb=# select current_database();
 current_database
------------------
 edb
(1 row)

edb=# \! hostname
pgedb02
edb=#
```

- Execute the **pg_basebackup** command

```
[enterprisedb@pgedb02 ~]$ pg_basebackup -h pgedb01.example.com -U
enterprisedb -D /var/lib/edb/as13/data -p 5444 -vvv -P
```

```
[enterprisedb@pgedb02 ~]$ pg_basebackup -h pgedb01.example.com -U enterprisedb -D /var/lib/edb/as13/data -p 5444 -vvv -P
pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/9000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: created temporary replication slot "pg_basebackup_2665"
65593/65593 kB (100%), 1/1 tablespace
pg_basebackup: write-ahead log end point: 0/9000100
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: renaming backup_manifest.tmp to backup_manifest
pg_basebackup: base backup completed
[enterprisedb@pgedb02 ~]$
```

PARAMETER	DESCRIPTION
-h	Hostname name of the Source machine
-U	User name of Source database having Superuser privilege
-D	Data Directory location where you want to configure the PGDATA
-p	port Number of target database
-v	verbose
-P	show the progress

Step-4 validation

Connect to the database **(clone Cluster)**

```
[enterprisedb@pgedb02 ~]$
[enterprisedb@pgedb02 ~]$ psql -h pgedb02.example.com -d edb -U enterprisedb -p 5444
psql (13.16.22)
Type "help" for help.

edb=# select current_database();
 current_database
------------------
 edb
(1 row)

edb=# \c world
You are now connected to database "world" as user "enterprisedb".
world=# select * from test;
 id | name
----+------
  1 | AAA
  2 | BBB
(2 rows)

world=#
```

- Now let's add the database details to the PostgreSQL service so the database will start automatically After the database server reboot also it will start and stop the database using the system service. If you are using the **EDB** Advanced server the change PGDATA in the below file

vi /lib/systemd/system/edb-as-13.service

PGDATA=/var/lib/edb/as13/data

- Start the service using below command

```
[root@pgedb02 system]# systemctl start edb-as-16

[root@pgedb02 system]# chkconfig edb-as-13 on
```

- To validate it, restart the machine and check database will be UP after reboot.

```
[root@pgedb02 system]#
[root@pgedb02 system]# ps -ef | grep -i edb
enterpr+    1405       1  0 17:08 ?        00:00:00 /usr/edb/as13/bin/edb-postmaster -D /var/lib/edb/as13/data
root        1510    1352  0 17:16 pts/0    00:00:00 grep --color=auto -i edb
[root@pgedb02 system]#
```