Upgrading Postgres from 12.4 to 13.0 (major upgrade).docx

Dear Reader,
In this post, we will see how can we upgrade the major version upgrade of postgres.

- Before PG 10, if upgrade changes any of the first 2 sets of digits then it will get considered as a major version upgrade (e.g. from 9.5.8 to 9.6.4 OR 9.5.8 to 10.1) but if 3rd digit changes then it's a minor version upgrade (e.g. 9.6.2 to 9.6.3)
- After PG 10, if 1st digit changes after the upgrade (e.g. pg 12.4 to 13.0) then it's a major upgrade but if the 2nd set of digits are changing after the upgrade then it's a minor upgrade (e.g. from 12.1 to 12.4).
- In this document, we will see how to perform the major version upgrade. We will use pg_upgrade utility
- For the minor version, mostly I will create a separate post (but trust me it's very simple).

1. A freshly installed Postgres 12.4 cluster is running

```
[postgres@pgserver1 ~]$lk postgres
postgres  1404     1  0 14:38 ?        00:00:00 /usr/pgsql-12/bin/postmaster -D /var/lib/pgsql/12/data/
postgres  1406  1404  0 14:38 ?        00:00:00 postgres: logger
postgres  1408  1404  0 14:38 ?        00:00:00 postgres: checkpointer
postgres  1409  1404  0 14:38 ?        00:00:00 postgres: background writer
postgres  1410  1404  0 14:38 ?        00:00:00 postgres: walwriter
postgres  1411  1404  0 14:38 ?        00:00:00 postgres: autovacuum launcher
postgres  1412  1404  0 14:38 ?        00:00:00 postgres: stats collector
postgres  1413  1404  0 14:38 ?        00:00:00 postgres: logical replication launcher
root      1579  2806  0 14:45 pts/0    00:00:00 sudo su - postgres
root      1581  1579  0 14:45 pts/0    00:00:00 su - postgres
postgres  1582  1581  0 14:45 pts/0    00:00:00 -bash
postgres  1706  1582  0 14:50 pts/0    00:00:00 ps -ef
postgres  1707  1582  0 14:50 pts/0    00:00:00 grep --color=auto postgres
```

2. Let's create a new database and add some data to it.
   As we can see, I have created a database dvdrental and it's film table has 1000 rows in it.

```
[postgres@pgserver1 tmp]$lt dvdrental.zip
-rw-rw-r--. 1 postgres postgres 538K Sep 13 13:17 dvdrental.zip
[postgres@pgserver1 tmp]$chmod 755 dvdrental.zip
[postgres@pgserver1 tmp]$unzip dvdrental.zip
[postgres@pgserver1 tmp]$lt dvdrental.tar
-rw-r--r--. 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]$psql -d dvdrental -c "select count(*) from film;"
```

```
[postgres@pgserver1 tmp]$lt dvdrental.zip
-rw-rw-r--. 1 postgres postgres 538K Sep 13 13:17 dvdrental.zip
[postgres@pgserver1 tmp]$chmod 755 dvdrental.zip
[postgres@pgserver1 tmp]$unzip dvdrental.zip
Archive:  dvdrental.zip
  inflating: dvdrental.tar
[postgres@pgserver1 tmp]$lt dvdrental.tar
-rw-r--r--. 1 postgres postgres 2.8M May 12  2019 dvdrental.tar
[postgres@pgserver1 tmp]$
[postgres@pgserver1 tmp]$createdb -O postgres -U postgres dvdrental
[postgres@pgserver1 tmp]$pg_restore -d dvdrental -Ft dvdrental.tar
[postgres@pgserver1 tmp]$
[postgres@pgserver1 tmp]$psql -d dvdrental -c "select count(*) from film;"
 count
-------
  1000
(1 row)
```

3. As a little twist, I'm also installing 1 postgres extension pg_buffercache. This extension we use to understand which table pages are occupying space in the shared buffers.
   The purpose of adding an extension is to show how we can upgrade those extensions also to their higher versions. However, we must first read the documentation of the extension to make sure that the extension is supported in the newer version of the Postgres.
   pg_buffercache gets shipped as a utility with contrib so I'm installing it first and then enabling pg_buffercache in the dvdrental database. Please note that for installing contrib I'm using yum so either **root or sudo access** is needed.

```
[root@pgserver1 postgres-master]# yum install postgresql12-contrib
```

```
[postgres@pgserver1 ~]$psql -d dvdrental
dvdrental=# CREATE EXTENSION pg_buffercache;
dvdrental=# \dx
```

```
[root@pgserver1 postgres-master]# yum install postgresql12-contrib
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.hbcse.tifr.res.in
 * extras: centos.hbcse.tifr.res.in
 * updates: centos.hbcse.tifr.res.in
Resolving Dependencies
--> Running transaction check
---> Package postgresql12-contrib.x86_64 0:12.4-1PGDG.rhel7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch              Version                 Repository  Size
================================================================================
Installing:
 postgresql12-contrib   x86_64            12.4-1PGDG.rhel7        pgdg12     610 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 610 k
```

```
[postgres@pgserver1 ~]$psql -d dvdrental
psql (12.4)
Type "help" for help.

dvdrental=# \dx
                List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(1 row)

dvdrental=# CREATE EXTENSION pg_buffercache;
CREATE EXTENSION
dvdrental=#
dvdrental=# \dx
                     List of installed extensions
      Name       | Version |   Schema   |           Description
-----------------+---------+------------+--------------------------------
 pg_buffercache | 1.3     | public     | examine the shared buffer cache
 plpgsql        | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)

dvdrental=# \q
```

4. Install postgres 13 as a **root** user
   Here, we are just installing and initializing a cluster but we are not starting it.
   I copied the code from the official postgres site and removed the startup command.

```
[root@pgserver1 ~]# cat install_pg13.sh
# Install the repository RPM:
yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm

# Install PostgreSQL:
yum install -y postgresql13-server

# Optionally initialize the database and enable automatic start:
/usr/pgsql-13/bin/postgresql-13-setup initdb
systemctl enable postgresql-13



[root@pgserver1 ~]# sh install_pg13.sh
```

   After the installation, we can see that new base directory 13 is created which will be used like a new $PGDATA after the upgrade.

```
[postgres@pgserver1 ~]$ls -ld 12 13
drwx------. 4 postgres postgres 51 Oct 11 12:20 12
drwx------. 4 postgres postgres 51 Oct 11 15:45 13
[postgres@pgserver1 ~]$ls -l 13
total 8
drwx------.  2 postgres postgres    6 Sep 23 02:55 backups
drwx------. 20 postgres postgres 4096 Oct 11 15:45 data
-rw-------.  1 postgres postgres  920 Oct 11 15:45 initdb.log
[postgres@pgserver1 ~]$ls -l 13/data/
total 52
drwx------. 5 postgres postgres    41 Oct 11 15:45 base
drwx------. 2 postgres postgres  4096 Oct 11 15:45 global
drwx------. 2 postgres postgres     6 Oct 11 15:45 log
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_commit_ts
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_dynshmem
-rw-------. 1 postgres postgres  4548 Oct 11 15:45 pg_hba.conf
-rw-------. 1 postgres postgres  1636 Oct 11 15:45 pg_ident.conf
drwx------. 4 postgres postgres    68 Oct 11 15:45 pg_logical
drwx------. 4 postgres postgres    36 Oct 11 15:45 pg_multixact
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_notify
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_replslot
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_serial
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_snapshots
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_stat
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_stat_tmp
drwx------. 2 postgres postgres    18 Oct 11 15:45 pg_subtrans
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_tblspc
drwx------. 2 postgres postgres     6 Oct 11 15:45 pg_twophase
-rw-------. 1 postgres postgres     3 Oct 11 15:45 PG_VERSION
drwx------. 3 postgres postgres    60 Oct 11 15:45 pg_wal
drwx------. 2 postgres postgres    18 Oct 11 15:45 pg_xact
-rw-------. 1 postgres postgres    88 Oct 11 15:45 postgresql.auto.conf
-rw-------. 1 postgres postgres 27985 Oct 11 15:45 postgresql.conf
```

5. Let's install postgres contrib 13 for PG 13 database cluster as we have 1 extension in the PG 12 database which is supposed to be migrated to PG 13 compatible version of it's own.
   Please note that we are just installing PG 13 and PG contrib 13, we are not running any "CREATE EXTENSION….." command. It will automatically get created under the appropriate database during the upgrade process.
   As a **root** user run:

```
[root@pgserver1 ~]# yum install -y postgresql13-contrib
```

```
[root@pgserver1 ~]# yum install -y postgresql13-contrib
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.hbcse.tifr.res.in
 * extras: centos.hbcse.tifr.res.in
 * updates: centos.hbcse.tifr.res.in
Resolving Dependencies
--> Running transaction check
---> Package postgresql13-contrib.x86_64 0:13.0-1PGDG.rhel7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch          Version                 Repository    Size
================================================================================
Installing:
 postgresql13-contrib   x86_64        13.0-1PGDG.rhel7        pgdg13       607 k

Transaction Summary
================================================================================
Install  1 Package
```

6. For the upgrade process, pg_upgrade utility will be used. During the upgrade process, this utility will connect old and new clusters several times so we should adjust pg_hba.conf or use ~/.pgpass file. For me, I'm having local + peer rule for both 12 and 13 clusters which is sufficient.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
```

7. I created 2 simple separate environment files for PG 12 and 13 clusters. Those will come handy at the time of upgrade.

```
[postgres@pgserver1 ~]$lt *.env
-rw-r--r--. 1 postgres postgres 94 Oct 11 12:24 pg12.env
-rw-r--r--. 1 postgres postgres 94 Oct 11 16:24 pg13.env

[postgres@pgserver1 ~]$cat pg12.env
export PATH=/usr/pgsql-12/bin/:$PATH
export PGDATA=/var/lib/pgsql/12/data/
export PGPORT=5432

[postgres@pgserver1 ~]$cat pg13.env
export PATH=/usr/pgsql-13/bin/:$PATH
export PGDATA=/var/lib/pgsql/13/data/
export PGPORT=5432
```

```
[postgres@pgserver1 ~]$lt *.env
-rw-r--r--. 1 postgres postgres 94 Oct 11 12:24 pg12.env
-rw-r--r--. 1 postgres postgres 94 Oct 11 16:24 pg13.env
[postgres@pgserver1 ~]$cat pg12.env
export PATH=/usr/pgsql-12/bin/:$PATH
export PGDATA=/var/lib/pgsql/12/data/
export PGPORT=5432
[postgres@pgserver1 ~]$cat pg13.env
export PATH=/usr/pgsql-13/bin/:$PATH
export PGDATA=/var/lib/pgsql/13/data/
export PGPORT=5432
```

8. Take a proper backup of the existing cluster in case we need to undo the upgrade process. This is a very important step and must be followed to be double sure.
   I'm using pg_basebackup but you can use anything good for your environment.
   As a **postgres** user run:

```
[postgres@pgserver1 ~]$source pg12.env
[postgres@pgserver1 ~]$pg_basebackup -D /backup -Fp
[postgres@pgserver1 ~]$lt /backup
```

```
[postgres@pgserver1 ~]$source pg12.env
[postgres@pgserver1 ~]$pg_basebackup -D /backup -Fp
[postgres@pgserver1 ~]$lt /backup
total 88K
-rw-------. 1 postgres postgres  224 Oct 11 16:34 backup_label
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_twophase
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_subtrans
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_snapshots
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_serial
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_notify
drwx------. 4 postgres postgres   36 Oct 11 16:34 pg_multixact
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_dynshmem
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_commit_ts
drwx------. 3 postgres postgres   60 Oct 11 16:34 pg_wal
drwx------. 6 postgres postgres   54 Oct 11 16:34 base
-rw-------. 1 postgres postgres   88 Oct 11 16:34 postgresql.auto.conf
drwx------. 2 postgres postgres   18 Oct 11 16:34 pg_xact
-rw-------. 1 postgres postgres    3 Oct 11 16:34 PG_VERSION
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_tblspc
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_stat_tmp
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_stat
drwx------. 2 postgres postgres    6 Oct 11 16:34 pg_replslot
drwx------. 4 postgres postgres   68 Oct 11 16:34 pg_logical
-rw-------. 1 postgres postgres 4.5K Oct 11 16:34 pg_hba.conf
-rw-------. 1 postgres postgres  27K Oct 11 16:34 postgresql.conf_11102020_1
-rw-------. 1 postgres postgres  27K Oct 11 16:34 postgresql.conf
-rw-------. 1 postgres postgres 1.6K Oct 11 16:34 pg_ident.conf
drwx------. 2 postgres postgres   32 Oct 11 16:34 log
drwx------. 2 postgres postgres 4.0K Oct 11 16:34 global
-rw-------. 1 postgres postgres   30 Oct 11 16:34 current_logfiles
```

9. Stop the PG 12 cluster as a **postgres** user.

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

```
[postgres@pgserver1 ~]$lk postgres
root      1231 30742  0 16:33 pts/1    00:00:00 sudo su - postgres
root      1233  1231  0 16:33 pts/1    00:00:00 su - postgres
postgres  1234  1233  0 16:33 pts/1    00:00:00 -bash
postgres  1448  1234  0 16:35 pts/1    00:00:00 ps -ef
postgres  1449  1234  0 16:35 pts/1    00:00:00 grep --color=auto postgres
postgres 23327     1  0 13:47 ?        00:00:00 /usr/pgsql-12/bin/postgres -D /var/lib/pgsql/12/data
postgres 23328 23327  0 13:47 ?        00:00:00 postgres: logger
postgres 23330 23327  0 13:47 ?        00:00:00 postgres: checkpointer
postgres 23331 23327  0 13:47 ?        00:00:00 postgres: background writer
postgres 23332 23327  0 13:47 ?        00:00:00 postgres: walwriter
postgres 23333 23327  0 13:47 ?        00:00:00 postgres: autovacuum launcher
postgres 23334 23327  0 13:47 ?        00:00:00 postgres: stats collector
postgres 23335 23327  0 13:47 ?        00:00:00 postgres: logical replication launcher
[postgres@pgserver1 ~]$
[postgres@pgserver1 ~]$pg_ctl -D $PGDATA -mf stop
waiting for server to shut down.... done
server stopped
[postgres@pgserver1 ~]$
[postgres@pgserver1 ~]$lk postgres
root      1231 30742  0 16:33 pts/1    00:00:00 sudo su - postgres
root      1233  1231  0 16:33 pts/1    00:00:00 su - postgres
postgres  1234  1233  0 16:33 pts/1    00:00:00 -bash
postgres  1484  1234  0 16:36 pts/1    00:00:00 ps -ef
postgres  1485  1234  0 16:36 pts/1    00:00:00 grep --color=auto postgres
[postgres@pgserver1 ~]$
```

10. Source the environment file of PG 13 and run pg_upgrade with check mode.
    While doing upgrade we must use the target version's pg_uprade utility.
    Using a check flag only verifies the clusters based on several criteria but it doesn't change anything or never runs upgrade.
    Every time we run pg_upgrade, it creates 3 files which we can read for more information:
    a. pg_upgrade_utility.log
    b. pg_upgrade_internal.log
    c. pg_upgrade_server.log

```
[postgres@pgserver1 ~]$source pg13.env
[postgres@pgserver1 ~]$pg_upgrade -V
pg_upgrade (PostgreSQL) 13.0
[postgres@pgserver1 ~]$pg_upgrade \
>    --old-datadir=/var/lib/pgsql/12/data/ \
>    --new-datadir=/var/lib/pgsql/13/data/ \
>    --old-bindir=/usr/pgsql-12/bin \
>    --new-bindir=/usr/pgsql-13/bin \
>    --old-options '-c config_file=/var/lib/pgsql/12/data/postgresql.conf' \
>    --new-options '-c config_file=/var/lib/pgsql/13/data/postgresql.conf' \
>    --check
Performing Consistency Checks
-----------------------------
Checking cluster versions                                   ok
Checking database user is the install user                  ok
Checking database connection settings                       ok
Checking for prepared transactions                          ok
Checking for reg* data types in user tables                 ok
Checking for contrib/isn with bigint-passing mismatch       ok
Checking for presence of required libraries                 ok
Checking database user is the install user                  ok
Checking for prepared transactions                          ok

*Clusters are compatible*
```

```
[postgres@pgserver1 ~]$source pg13.env
[postgres@pgserver1 ~]$pg_upgrade -V
pg_upgrade (PostgreSQL) 13.0
[postgres@pgserver1 ~]$pg_upgrade \
>     --old-datadir=/var/lib/pgsql/12/data/ \
>     --new-datadir=/var/lib/pgsql/13/data/ \
>     --old-bindir=/usr/pgsql-12/bin \
>     --new-bindir=/usr/pgsql-13/bin \
>     --old-options '-c config_file=/var/lib/pgsql/12/data/postgresql.conf' \
>     --new-options '-c config_file=/var/lib/pgsql/13/data/postgresql.conf' \
>     --check
Performing Consistency Checks
-----------------------------
Checking cluster versions                                   ok
Checking database user is the install user                  ok
Checking database connection settings                       ok
Checking for prepared transactions                          ok
Checking for reg* data types in user tables                 ok
Checking for contrib/isn with bigint-passing mismatch       ok
Checking for presence of required libraries                 ok
Checking database user is the install user                  ok
Checking for prepared transactions                          ok

*Clusters are compatible*
```

11. Run the pg_upgrade without a check flag i.e. running the actual upgrade process.

```
[postgres@pgserver1 ~]$pg_upgrade \
>   --old-datadir=/var/lib/pgsql/12/data/ \
>   --new-datadir=/var/lib/pgsql/13/data/ \
>   --old-bindir=/usr/pgsql-12/bin \
>   --new-bindir=/usr/pgsql-13/bin \
>   --old-options '-c config_file=/var/lib/pgsql/12/data/postgresql.conf' \
>   --new-options '-c config_file=/var/lib/pgsql/13/data/postgresql.conf'
Performing Consistency Checks
-----------------------------
Checking cluster versions                           ok
Checking database user is the install user            ok
Checking database connection settings                  ok
Checking for prepared transactions                    ok
Checking for reg* data types in user tables           ok
Checking for contrib/isn with bigint-passing mismatch      ok
Creating dump of global objects                       ok
Creating dump of database schemas
                                            ok
Checking for presence of required libraries           ok
Checking database user is the install user            ok
Checking for prepared transactions                    ok

If pg_upgrade fails after this point, you must re-initdb the
new cluster before continuing.

Performing Upgrade
```

```
------------------
Analyzing all rows in the new cluster                       ok
Freezing all rows in the new cluster                        ok
Deleting files from new pg_xact                             ok
Copying old pg_xact to new server                           ok
Setting next transaction ID and epoch for new cluster       ok
Deleting files from new pg_multixact/offsets                ok
Copying old pg_multixact/offsets to new server              ok
Deleting files from new pg_multixact/members                ok
Copying old pg_multixact/members to new server              ok
Setting next multixact ID and offset for new cluster        ok
Resetting WAL archives                                      ok
Setting frozenxid and minmxid counters in new cluster       ok
Restoring global objects in the new cluster                 ok
Restoring database schemas in the new cluster
                                                            ok
Copying user relation files
                                                            ok
Setting next OID for new cluster                            ok
Sync data directory to disk                                 ok
Creating script to analyze new cluster                      ok
Creating script to delete old cluster                       ok

Upgrade Complete
----------------
Optimizer statistics are not transferred by pg_upgrade so,
once you start the new server, consider running:
    ./analyze_new_cluster.sh

Running this script will delete the old cluster's data files:
    ./delete_old_cluster.sh
```

```
[postgres@pgserver1 ~]$pg_upgrade \
>   --old-datadir=/var/lib/pgsql/12/data/ \
>   --new-datadir=/var/lib/pgsql/13/data/ \
>   --old-bindir=/usr/pgsql-12/bin \
>   --new-bindir=/usr/pgsql-13/bin \
>   --old-options '-c config_file=/var/lib/pgsql/12/data/postgresql.conf' \
>   --new-options '-c config_file=/var/lib/pgsql/13/data/postgresql.conf'
Performing Consistency Checks
-----------------------------
Checking cluster versions                                   ok
Checking database user is the install user                  ok
Checking database connection settings                       ok
Checking for prepared transactions                          ok
Checking for reg* data types in user tables                 ok
Checking for contrib/isn with bigint-passing mismatch       ok
Creating dump of global objects                             ok
Creating dump of database schemas
                                                            ok
Checking for presence of required libraries                 ok
Checking database user is the install user                  ok
Checking for prepared transactions                          ok

If pg_upgrade fails after this point, you must re-initdb the
new cluster before continuing.
```

12. After the upgrade completes successfully 2 files will be created in the same directory from where we run pg_upgrade:
    a. **analyze_new_cluster.sh**: During the upgrade, optimizer stats won't get transferred so it's important to gather the stats. We can use this script or we can manually gather the stats. This script internally runs below command:
    "/usr/pgsql-13/bin/vacuumdb" --all --analyze-in-stages
    b. **delete_old_cluster.sh**: This script will delete old cluster's datafiles i.e. old cluster's $PGDATA

13. Start the upgraded cluster, verify the data:

```
[postgres@pgserver1 ~]$pg_ctl -D $PGDATA -mf start
[postgres@pgserver1 ~]$psql


postgres=# \l
postgres=# \c dvdrental
dvdrental=# select count(*) from film;
postgres=# \q

[postgres@pgserver1 ~]$psql -c "select version();"
```

```
[postgres@pgserver1 ~]$pg_ctl -D $PGDATA -mf start
waiting for server to start....2020-10-11 17:05:07.987 IST [3687] LOG:  redirecting log output to logging collector process
2020-10-11 17:05:07.987 IST [3687] HINT:  Future log output will appear in directory "log".
 done
server started
[postgres@pgserver1 ~]$psql
psql (13.0)
Type "help" for help.

postgres=# \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 | postgres=CTc/postgres+
           |          |          |             |             | =c/postgres
(4 rows)

postgres=# \c dvdrental
You are now connected to database "dvdrental" as user "postgres".
dvdrental=# \dx
                        List of installed extensions
      Name      | Version |   Schema   |          Description
----------------+---------+------------+------------------------------
 pg_buffercache | 1.3     | public     | examine the shared buffer cache
 plpgsql        | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)

dvdrental=# select count(*) from film;
 count
-------
  1000
(1 row)

dvdrental=# \q
[postgres@pgserver1 ~]$psql -c "select version();"
                                                  version
-----------------------------------------------------------------------------------------------------------
 PostgreSQL 13.0 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39), 64-bit
(1 row)
```

14. If everything is good then we can run analyze_new_cluster.sh

```
[postgres@pgserver1 ~]$sh analyze_new_cluster.sh
```

```
[postgres@pgserver1 ~]$sh analyze_new_cluster.sh
This script will generate minimal optimizer statistics rapidly
so your system is usable, and then gather statistics twice more
with increasing accuracy.  When it is done, your system will
have the default level of optimizer statistics.

If you have used ALTER TABLE to modify the statistics target for
any tables, you might want to remove them and restore them after
running this script because they will delay fast statistics generation.

If you would like default statistics as quickly as possible, cancel
this script and run:
    "/usr/pgsql-13/bin/vacuumdb" --all --analyze-only

vacuumdb: processing database "dvdrental": Generating minimal optimizer statistics (1 target)
vacuumdb: processing database "postgres": Generating minimal optimizer statistics (1 target)
vacuumdb: processing database "template1": Generating minimal optimizer statistics (1 target)
vacuumdb: processing database "dvdrental": Generating medium optimizer statistics (10 targets)
vacuumdb: processing database "postgres": Generating medium optimizer statistics (10 targets)
vacuumdb: processing database "template1": Generating medium optimizer statistics (10 targets)
vacuumdb: processing database "dvdrental": Generating default (full) optimizer statistics
vacuumdb: processing database "postgres": Generating default (full) optimizer statistics
vacuumdb: processing database "template1": Generating default (full) optimizer statistics

Done
```

15. As a final step, we can run delete_old_cluster.sh and delete the old $PGDATA directory.

```
[postgres@pgserver1 ~]$sh delete_old_cluster.sh
```

```
[postgres@pgserver1 ~]$sh delete_old_cluster.sh
```