Administración de Sistemas

Práctica 4

23/11/2021

Marcos Eladio Somoza Corral

21711787

ÍNDICE

1. Para comenzar, se recomienda que por seguridad cada servicio utilice su propio usuario, por lo que deberemos crear un usuario llamado postgres sin acceso a través de SSH. ¿Qué comando has utilizado para crear este usuario?

Con el comando ***sudo adduser*** nombre creamos un nuevo usuario, y para darle acceso de *sudoer*, se deberá modificar el archivo ***/etc/sudoers*** como super usuario y añadir **postgres ALL=(ALL:ALL) ALL.**

```
somo@somo:~$ sudo adduser  postgres
Adding user `postgres' ...
Adding new group `postgres' (1001) ...
Adding new user `postgres' (1001) with group `postgres' ...
```

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
postgres        ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

Finalmente, para deshabilitar las conexiones **SSH** para el usuario **postgres**, tendremos que modificar el fichero **/etc/ssh/sshd_config** y añadir **DenyUsers postgres**. Cabe destacar que debe haber una tabulación entre *DenyUsers* y el usuario para que reconozca la directiva.

```
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
DenyUsers       postgres
```

2. Descarga el software desde la página oficial, indicando los pasos necesarios para descargarlo y descomprimirlo.

Para descargarlo, se hará uso del comando **wget** con el link del archivo **.tar** tal que **wget https://ftp.postgresql.org/pub/source/v14.1/postgresql-14.1.tar.gz.**

```
postgres@somo:~/download$ wget https://ftp.postgresql.org/pub/source/v14.1/postgresql-14.1.tar.gz
--2021-11-22 02:46:00--  https://ftp.postgresql.org/pub/source/v14.1/postgresql-14.1.tar.gz
Resolving ftp.postgresql.org (ftp.postgresql.org)... 87.238.57.227, 72.32.157.246, 217.196.149.55, ...
Connecting to ftp.postgresql.org (ftp.postgresql.org)|87.238.57.227|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28666442 (27M) [application/octet-stream]
Saving to: 'postgresql-14.1.tar.gz'

postgresql-14.1.tar.gz              100%[=============================================================================>]  27.34M  7.95MB/s    in 4.3s

2021-11-22 02:46:05 (6.34 MB/s) - 'postgresql-14.1.tar.gz' saved [28666442/28666442]
```

Y podremos descomprimir el archivo con **tar -xzvf postgresql-14.1.tar.gz**.

```
postgres@somo:~/download$ tar -xzvf postgresql-14.1.tar.gz
```

```
postgres@somo:~/download$ ls
postgresql-14.1  postgresql-14.1.tar.gz
postgres@somo:~/download$ ls postgresql-14.1
aclocal.m4  config  configure  configure.ac  contrib  COPYRIGHT  doc  GNUmakefile.in  HISTORY  INSTALL  Makefile  README  src
```

3. Ejecuta la compilación teniendo en cuenta que nuestro directorio de instalación será /usr/local/pgsql. Describe el proceso que has seguido.

Primero, moveremos la carpeta descomprimida **postgresql-14.1** a el directorio **/usr/local** y cambiaremos el nombre de la carpeta a **pgsql**.

```
postgres@somo:~/download$ sudo mv postgresql-14.1 /usr/local/
[sudo] password for postgres:
postgres@somo:~/download$ ls
postgresql-14.1.tar.gz
postgres@somo:~/download$ ls /usr/local
bin  etc  games  include  lib  man  postgresql-14.1  sbin  share  src
```

```
postgres@somo:/usr/local$ sudo mv postgresql-14.1/ pgsql/
postgres@somo:/usr/local$ ls
bin  etc  games  include  lib  man  pgsql  sbin  share  src
```

Para facilitar la compilación, se ha creado un script **runconfigure.sh** que actuará de *wrapper* para ayudar a la posterior configuración.

```
 GNU nano 4.8
#!/bin/bash
./configure \
    --prefix=/usr/local/pgsql \
    --with-ossp-uuid
```

Para poder compliar el script, se deben instalar primero algunos paquetes para que sea posible:

```
postgres@somo:~$ sudo apt-get install flex
postgres@somo:~$ sudo apt-get install bison build-essential
postgres@somo:~$ sudo apt-get install libreadline6-dev
postgres@somo:~$ sudo apt-get install zlib1g-dev
postgres@somo:~$ sudo apt-get install libossp-uuid-dev
```

Y ya se podría ejecutar el archivo **runconfig.sh** con **./runconfig.sh**.

```
postgres@somo:/usr/local/pgsql$ ./runconfigure.sh
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking which template to use... linux
```

Ahora se deberá ejecutar el comando **make** en el mismo directorio (de *cmake*, para el compilador **gcc**) así como **make install**. Una vez terminados los procesos, se deberá acceder al directorio **/contrib/uuid-ossp** y realizar los dos mismos comandos (**make** y **make install**).

```
postgres@somo:/usr/local/pgsql/contrib/uuid-ossp$ make install
make -C ../../src/backend generated-headers
make[1]: Entering directory '/usr/local/pgsql/src/backend'
```

```
postgres@somo:/usr/local/pgsql/contrib$ cd uuid-ossp/
postgres@somo:/usr/local/pgsql/contrib/uuid-ossp$ make
make -C ../../src/backend generated-headers
make[1]: Entering directory '/usr/local/pgsql/src/backend'
```

Y asignamos el directorio de **/usr/local/pgsql** al usuario postgres con el tag **-R** para asignar recursivamente cada archivo del directorio mediante el comando **sudo chown -R postgres:postgres /usr/local/pgsql**.

```
postgres@somo:/usr/local/pgsql/contrib/uuid-ossp$ sudo chown -R postgres:postgres /usr/local/pgsql
```

4.¿Qué acciones complementarias has tenido que realizar para poder realizar la instalación?

Como se ha mencionado anteriormente, se ha debido de crear el script **runconfig.sh** para facilitar el proceso de configuración así como instalar los paquetes **flex**, **bison**, **build-essential**, **libreadline6-dev**, **zlib1g-dev** y **libossp-uuid-dev** para que el proceso de compilación funcione correctamente. Por último, ejecutar los comandos **make** y **make install** en los directorios **/usr/local/pgsql** y **/usr/local/pgsql/contrib/uuid-ossp** y asignar todos los ficheros al usuario *postgres*. Ahora ya se encuentra instalado postgresql en el equipo, pero se deben configurar la cantidad de memoria que el *kernel* permite usar a *postgresql*:

Se creará un script en el directorio home postgresql-kernel-params.sh tal que:

```
#!/bin/bash
SYSCTL=/sbin/sysctl
echo "# add the output of this script to /etc/sysctl.conf,"
echo "# and then, as root, run"
```

```
echo
echo "# sysctl -p /etc/sysctl.conf"
echo
echo "# to load change the kernel settings for these parameters."
echo
PAGE_SIZE=`getconf PAGE_SIZE`
echo "# page size is: $PAGE_SIZE"
NUM_PHYS_PAGES=`getconf _PHYS_PAGES`
echo "# number of physical pages on this box: $NUM_PHYS_PAGES"
CURR_SHMALL=`$SYSCTL -n kernel.shmall`
PREF_SHMALL=`expr $NUM_PHYS_PAGES / 2`
echo "# kernel.shmall should be half of the number of pages. Current
kernel.shmall, in pages, is: $CURR_SHMALL"
echo "# kernel.shmall should be:"
echo
echo "kernel.shmall = $PREF_SHMALL"
echo
CURR_SHMMAX=`$SYSCTL -n kernel.shmmax`
PREF_SHMMAX=`expr $PREF_SHMALL \* $PAGE_SIZE`
echo "# kernel.shmmax should be half of available RAM, in kB. Current
kernel.shmmax, in kB, is: $CURR_SHMMAX"
echo "# kernel.shmmax should be:"
echo
echo "kernel.shmmax = $PREF_SHMMAX"
echo
# CURR_SHMMIN=`$SYSCTL -n kernel.shmmin`  # XXX: does not exist on linux
# CURR_SHMSEG=`$SYSCTL -n kernel.shmseg`  # XXX: does not exist on linux
CURR_SHMMNI=`$SYSCTL -n kernel.shmmni`
echo "# kernel.shmmni is usually set to a sane amount on Linux. Currently, it is:
$CURR_SHMMNI"
# CURR_SEMMNI=`$SYSCTL -n kernel.semmni`  # XXX: does not exist on linux
# CURR_SHMMNI=`$SYSCTL -n kernel.semmns`  # XXX: does not exist on linux
# CURR_SHMMSL=`$SYSCTL -n kernel.semmsl`  # XXX: does not exist on linux
# CURR_SHMMSL=`$SYSCTL -n kernel.semmap`  # XXX: does not exist on linux
# CURR_SHMMSL=`$SYSCTL -n kernel.semmvx`  # XXX: does not exist on linux
CURR_SEM=`$SYSCTL -n kernel.sem`
echo "# kernel.sem usually has sane defauls. They are currently: $CURR_SEM"
```

que al ejecutar mostrará:

```
postgres@somo:~$ ./postgresql-kernel-params.sh
# add the output of this script to /etc/sysctl.conf,
# and then, as root, run

# sysctl -p /etc/sysctl.conf

# to load change the kernel settings for these parameters.

# page size is: 4096
# number of physical pages on this box: 1005150
# kernel.shmall should be half of the number of pages. Current kernel.shmall, in pages, is: 18446744073692774399
# kernel.shmall should be:

kernel.shmall = 502575

# kernel.shmmax should be half of available RAM, in kB. Current kernel.shmmax, in kB, is: 18446744073692774399
# kernel.shmmax should be:

kernel.shmmax = 2058547200

# kernel.shmmni is usually set to a sane amount on Linux. Currently, it is: 4096
# kernel.sem usually has sane defauls. They are currently: 32000        1024000000      500     32000
```

Lo que indica que se deberá modificar el archivo **/etc/sysctl.conf** para añadir las líneas **kernel.shmall = 502575** y **kernel.shmax = 2058547200**.

```
###############################################################
# Magic system request Key
# 0=disable, 1=enable all, >1 bitmask of sysrq functions
# See https://www.kernel.org/doc/html/latest/admin-guide/sysrq.html
# for what other values do
#kernel.sysrq=438
kernel.shmall = 502575
kernel.shmmax = 2058547200
```

Y con el comando **sysctl -p /etc/sysctl.conf** aplicamos los cambios :

```
postgres@somo:~$ sudo sysctl -p /etc/sysctl.conf
kernel.shmall = 502575
kernel.shmmax = 2058547200
```

5. Crea la carpeta de datos en /var/pgsql/data y la carpeta de logs en /var/pgsql/data/pg_log , asignándole como propietario el usuario creado en el punto 1. Adicionalmente inicializa la BD con codificación es_ES.utf8 ¿Qué comando has utilizado para inicializarla?

Crearemos las carpetas correspondientes y con el mismo comando anteriormente utilizado le daremos privilegios al usuario *postgres* para todas ellas (**sudo chown -R postgres:postgres /var/pgsql/**).

```
postgres@somo:/usr/local/pgsql/bin$ sudo mkdir /var/pgsql/data
postgres@somo:/usr/local/pgsql/bin$ sudo mkdir /var/pgsql/data/pg_log
```

```
postgres@somo:/usr/local/pgsql/bin$ sudo chown -R postgres:postgres /var/pgsql/
```

Para inicializar la BD, se deberá ir al directorio **/usr/local/pgsql/bin** y llamar a **initdb** , indicando el directorio de data y la codificación UTF8. El comando completo sería : **./initdb --pgdata=/var/pgsql/data --encoding=UTF8 --no-locale.**

```
postgres@somo:/usr/local/pgsql/bin$ ./initdb --pgdata=/var/pgsql/data --encoding=UTF8 --no-locale
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "C".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/pgsql/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    ./pg_ctl -D /var/pgsql/data -l logfile start
```

Cabe destacar que se ha debido eliminar la carpeta **/var/pgsql /data/pg_log/** dado que la inicialización de la BD no permitía que el directorio **/var/pgsql/data/** no estuviera vacío. Sin embargo, se ha vuelto a crear una vez inicializada la BD:

```
postgres@somo:/usr/local/pgsql/bin$ ls /var/pgsql/data/
base     pg_commit_ts  pg_hba.conf    pg_logical    pg_notify     pg_serial     pg_stat      pg_subtrans  pg_twophase  pg_wal   postgresql.auto.conf
global   pg_dynshmem   pg_ident.conf  pg_multixact  pg_replslot   pg_snapshots  pg_stat_tmp  pg_tblspc    PG_VERSION   pg_xact  postgresql.conf
postgres@somo:/usr/local/pgsql/bin$ sudo mkdir /var/pgsql/data/pg_log
postgres@somo:/usr/local/pgsql/bin$ ls /var/pgsql/data/
base     pg_commit_ts  pg_hba.conf    pg_log        pg_multixact  pg_replslot   pg_snapshots  pg_stat_tmp  pg_tblspc    PG_VERSION   pg_xact              postgresql.conf
global   pg_dynshmem   pg_ident.conf  pg_logical    pg_notify     pg_serial     pg_stat       pg_subtrans  pg_twophase  pg_wal       postgresql.auto.conf
postgres@somo:/usr/local/pgsql/bin$
```

6. Lanza la ejecución del servicio teniendo en cuenta los parámetros configurados en el punto anterior. ¿qué comando has utilizado?

Para lanzar la ejecución del servicio se ha utilizado el fichero **pg_ctl** donde se le ha dado el directorio de configuración **data** así como el de salida de **log**, sumado al comando **start**: **./pg_ctl -D /var/pgsql/data -l /var/pgsql/data/pg_log/logfile start**.

```
postgres@somo:/usr/local/pgsql/bin$ ./pg_ctl -D /var/pgsql/data -l /var/pgsql/data/pg_log/logfile st
art
waiting for server to start.... done
server started
```

7. Implementa un archivo para que se inicie como un servicio automáticamente al arrancar el servidor. Adjunta el código del script agregando los comentarios oportunos.

Se deberá ir al directorio **/usr/local/pgsql/contrib/start-sctrips** y dar permisos al fichero **Linux**. Luego lo copiaremos al subdirectorio **postgresql** de **init.d** en **/etc/init.d/postrgesql**.

```
postgres@somo:/usr/local/pgsql/contrib/start-scripts$ ls
freebsd  linux  macos
postgres@somo:/usr/local/pgsql/contrib/start-scripts$ sudo chmod +x linux
```

```
postgres@somo:/usr/local/pgsql/contrib/start-scripts$ sudo cp linux /etc/init.d/postgresql
```

Y si con nano editaremos el fichero Linux, asignando correctamente los directorios de data y log de postgresql, **/var/pgsql/data** y **/var/pgsql/data/pg_log/logfile** respectivamente.

```
  GNU nano 4.8                                postgresql                                Mo
# Original author:  Ryan Kirkpatrick <pgsql@rkirkpat.net>

# contrib/start-scripts/linux

## EDIT FROM HERE

# Installation prefix
prefix=/usr/local/pgsql

# Data directory
PGDATA="/var/pgsql/data"

# Who to run the postmaster as, usually "postgres".  (NOT "root")
PGUSER=postgres

# Where to keep a log file
PGLOG="/var/pgsql/data/pg_log/logfile"

# It's often a good idea to protect the postmaster from being killed by the
# OOM killer (which will tend to preferentially kill the postmaster because
# of the way it accounts for shared memory).  To do that, uncomment these
# three lines:
#PG_OOM_ADJUST_FILE=/proc/self/oom_score_adj
#PG_MASTER_OOM_SCORE_ADJ=-1000
#PG_CHILD_OOM_SCORE_ADJ=0
# Older Linux kernels may not have /proc/self/oom_score_adj, but instead
# /proc/self/oom_adj, which works similarly except for having a different
# range of scores.  For such a system, uncomment these three lines instead:
#PG_OOM_ADJUST_FILE=/proc/self/oom_adj
#PG_MASTER_OOM_SCORE_ADJ=-17
#PG_CHILD_OOM_SCORE_ADJ=0
```

Como se explicará en el siguiente apartado, se debe crear enlaces entre el directorio recientemente creado **/etc/init.d/postgres** y cada uno de los fichers postgres (**K02postgresql** en rc0.d,rc1.d y rc2.d, y **S98postgresql** en rc3.d,rc4.d y rc5.d) para que funcione correctamente la ejecución en inicio, mediante el comando **sudo ln -s /etc/init.d/postgresql /etc/rcX.d/XXXpostgresql**, donde X cambia tal que:

```
cryptdisks          kmod                open-vm-tools    screen-cleanup
postgres@somo:/etc/init.d$ sudo ln -s /etc/init.d/postgresql /etc/rc0.d/K02postgresql
postgres@somo:/etc/init.d$ sudo ln -s /etc/init.d/postgresql /etc/rc1.d/K02postgresql
postgres@somo:/etc/init.d$ sudo ln -s /etc/init.d/postgresql /etc/rc2.d/K02postgresql
postgres@somo:/etc/init.d$ sudo ln -s /etc/init.d/postgresql /etc/rc3.d/S98postgresql
postgres@somo:/etc/init.d$ sudo ln -s /etc/init.d/postgresql /etc/rc4.d/S98postgresql
postgres@somo:/etc/init.d$ sudo ln -s /etc/init.d/postgresql /etc/rc5.d/S98postgresql
postgres@somo:/etc/init.d$
```

Finalmente, guardaremos los cambios (autentificándonos) con el comando **update-rc.d postgresql defaults**.

```
postgres@somo:~$ update-rc.d postgresql defaults_
```

```
==== AUTHENTICATING FOR org.freedesktop.systemd1.reload-daemon ===
Authentication is required to reload the systemd state.
Authenticating as: somo
Password:
==== AUTHENTICATION COMPLETE ===
postgres@somo:~$
```

Para comprobar que todo funciona correctamente, si realizamos un sudo reboot y cuando cargue usamos el comando **sudo /etc/init.d/postgresql status** veremos que todo ha funcionado correctamente.

Ubuntu Server 20.04LTS [Running] - Oracle VM VirtualBox

```
postgres@somo:/home/somo$ sudo /etc/init.d/postgresql status
pg_ctl: server is running (PID: 1413)
/usr/local/pgsql/bin/postgres "-D" "/var/pgsql/data"
postgres@somo:/home/somo$
```

8. ¿Qué es el init.d?¿Qué otras carpetas podemos encontrar similares?

El **init.d** es el directorio donde se encuentran todos los scripts que se ejecutarán al arranque del sistema Linux. Se encuentra en **/etc/init.d** y es donde se especifican los comandos de arranque y parada de los servicios que contiene.
Existen también una serie de directorios en **/etc/rcX.d** (siendo X uno carácter determinado que actúa como nivel de ejecución, véase la imagen siguiente) llamados también directorios de enlace.

```
postgres@somo:/etc/init.d$ ls /etc/rc
rc0.d/ rc1.d/ rc2.d/ rc3.d/ rc4.d/ rc5.d/ rc6.d/ rcS.d/
```

Su función es determinar el orden de ejecución de los scripts en **/etc/init.d** basándose en los niveles de prioridad ´X´. Si se analiza uno de ellos, **rc0.d** por ejemplo, podremos ver cómo se deben estructurar los ficheros; el primer carácter debe se una **K** o una **S**, referenciando a **killed** o **started** en el nivel de ejecución actual. Seguido del número de orden (en este caso, 01) y finalizando con el nombre del script que, aunque sea irrelevante, donde apunta es el script que ejecutará.

```
postgres@somo:/etc/init.d$ ls /etc/rc0.d/
K01apache2              K01irqbalance       K01nscd             K01rsyslog
K01apache-htcacheclean  K01iscsid           K01open-iscsi       K01slapd
K01atd                  K01libnss-ldap      K01open-vm-tools    K01udev
K01cryptdisks           K01lvm2-lvmpolld    K01plymouth         K01unattended-upgrades
K01cryptdisks-early     K01multipath-tools  K01postgresql       K01uuidd
```

9. Configura el autovacuum de postgreSQL para que se ejecute automáticamente. ¿Qué archivo has tenido que modificar?

Para configurar **autovacuum** se deberá abrir el archivo **postgresql.conf** del directorio **/var/pgsql/data** y buscar el apartado de **autovacuum**. Ahí se podrá de comentar la línea **autovacuum = on** así como los otros parámetros de configuración si se deseara. De esta forma el demonio **autovacuum** se ejecutará automáticamente.

```
#------------------------------------------------------------------------------
# AUTOVACUUM
#------------------------------------------------------------------------------

autovacuum = on                         # Enable autovacuum subprocess?  'on'
                                        # requires track_counts to also be on.
autovacuum_max_workers = 3              # max number of autovacuum subprocesses
                                        # (change requires restart)
#autovacuum_naptime = 1min              # time between autovacuum runs
#autovacuum_vacuum_threshold = 50       # min number of row updates before
                                        # vacuum
#autovacuum_vacuum_insert_threshold = 1000      # min number of row inserts
                                        # before vacuum; -1 disables insert
                                        # vacuums
#autovacuum_analyze_threshold = 50      # min number of row updates before
                                        # analyze
#autovacuum_vacuum_scale_factor = 0.2   # fraction of table size before vacuum
#autovacuum_vacuum_insert_scale_factor = 0.2    # fraction of inserts over table
                                        # size before insert vacuum
#autovacuum_analyze_scale_factor = 0.1  # fraction of table size before analyze
#autovacuum_freeze_max_age = 200000000  # maximum XID age before forced vacuum
                                        # (change requires restart)
#autovacuum_multixact_freeze_max_age = 400000000        # maximum multixact age
                                        # before forced vacuum
                                        # (change requires restart)
#autovacuum_vacuum_cost_delay = 2ms     # default vacuum cost delay for
                                        # autovacuum, in milliseconds;
                                        # -1 means use vacuum_cost_delay
#autovacuum_vacuum_cost_limit = -1      # default vacuum cost limit for
                                        # autovacuum, -1 means use
                                        # vacuum_cost_limit
```

10. Indica los principales parámetros y su utilidad del archivo anterior.

autovacuum=on : Controla si el servidor debe ejecutar el demonio autovacuum.
autovacuum_max_workers=3 : Especifica el número máximo de de procesos autovacuum que pueden ejecutarse a la vez.
autovacuum_naptime=1min : Especifica el tiempo mínimo de espera entre procesos de autovacuum ejecutados en una base de datos.
autovacuum_vacuum_threshold =50 : Especifica el mínimo número de tuplas actualizadas o eliminadas necesarias para que se lance un demonio vacuum en una tabla.
autovacuum_vacuum_insert_threshold=1000 : Especifica el mínimo número de tuplas insertadas necesarias para que se lance un demonio vacuum en una tabla.
autovacuum_analyze_threshold =50 : Especifica el mínimo número de tuplas insertadas, actualizadas o eliminadas necesarias para que se lance un demonio analyze en una tabla.
autovacuum_ vacuum_scale_factor =0.2 : Especifica la fracción del tamaño de la tabla para añadir a autovacuum_vacuum_threshold para decidir lanzar un demonio vacuum.

autovacuum_ vacuum_insert_scale_factor =0.2 : Especifica la fracción del tamaño de la tabla para añadir a autovacuum_vacuum_insert_threshold para decidir lanzar un demonio vacuum.
autovacuum_ analyze_scale_factor =0.1 : Especifica la fracción del tamaño de la tabla para añadir a autovacuum_analyze_threshold para decidir lanzar un demonio analyze.
autovacuum_ freeze_max_age =200000000 : Determina la edad máxima (en transacciones) que un pg_class.relfrozenxid de una tabla puede alcanzar antes de que se lanze un vacuum.
autovacuum_multixact_freeze_max_age =0.2 : Determina la edad máxima (en multixact) que un pg_class.relminmxid de una tabla puede alcanzar antes de que se lanze un vacuum. También permite eliminar archivos antiguos de los subdirectorios pg_multixast/members y pg_multixact/offsets.
autovacuum_vacuum_cost_delay =2ms: Especifica el tiempo de espera que se usará en operaciones vacuum automáticas. Si se delimita a -1 se usará el valor de vacuum_cost_delay.
autovacuum_vacuum_cost_limit= -1: Determina el tiempo de espera límite usado en operaciones vacuum automáticas. Si se delimita a -1 se usará el valor de vacuum_cost_limit.

11. Crea una nueva BD llamada test. Indica los pasos que has seguido para su creación.

Para crear una nueva base de datos, bastaría con escribir el comando **createbd test** (siendo test el nombre de la bd) desde nuestro usuario **postgres**. Para comprobar que se ha creado correctamente, podemos acceder a **postgres** con **pgsql** desde el directorio **/usr/local/pgsql** y escribir el comando **\list**. Podremos conectarnos a la base de datos test con el comando **\c test.**

```
postgres@somo:/usr/local/pgsql$ createdb test
```

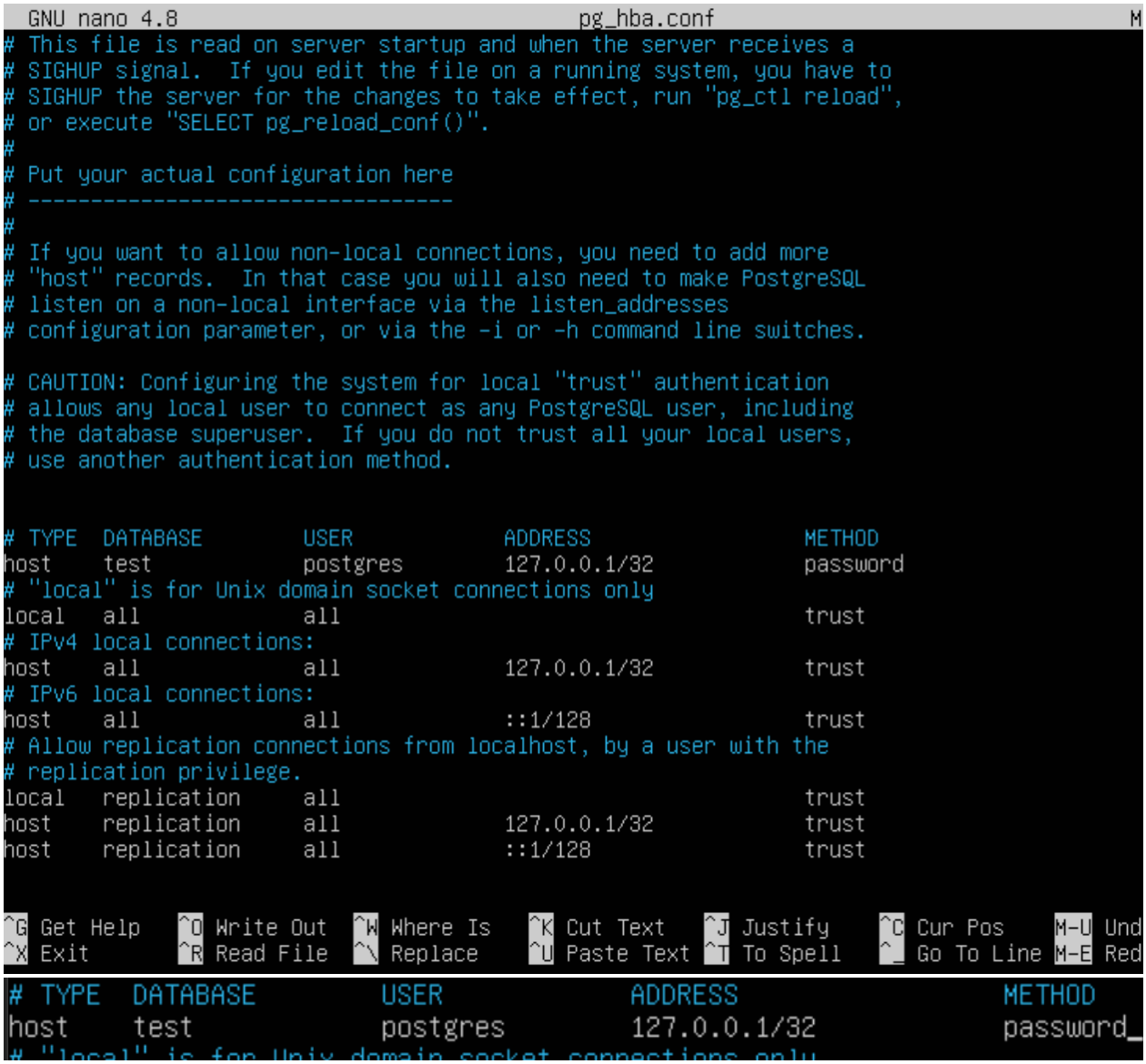
```
postgres@somo:/usr/local/pgsql$ psql
psql (12.9 (Ubuntu 12.9-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# /list
postgres-# \list
                                 List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 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
 test      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(4 rows)
```

```
postgres-# \c test
You are now connected to database "test" as user "postgres".
test-#
```

12. Configura para que la BD creada en el punto anterior sea accesible sólo desde localhost utilizando las credenciales del usuario que creamos en el punto 1. Describe la configuración utilizada y el archivo que has modificado.

El archivo para modificar es **pg_hba.conf**, en **/var/pgsql/data**, donde se ha añadido una línea extra con tipo **host** (para conexiones ipv4 locales) para la base de datos **test**, donde sólo puede acceder el usuario **postgres** desde la dirección de localhost (**127.0.0.1/32**) con el método de certificación de acceso **password**.



13. ¿Qué otros parámetros de configuración se pueden utilizar en el archivo pg_hba.conf?

Los parámetros disponibles se dividen en las cinco categorías:

- **TYPE**: El tipo de conexión.
  - local
  - host
  - hostssl
  - hostnossl

- **DATABASE**: El nombre de la base de datos.
- **USER**: El nombre de usuario con acceso, all serían todos los usuarios.
- **ADDRESS**: La ip entrante al servidor, puede ser ipv4 e ipv6 (incluyendo localhost).
- **METHOD**: El tipo de método de acceso; el proceso de autentificación.
  - trust
  - reject
  - md5
  - crypt
  - password
  - krb5
  - ident
  - pam
  - ldap

14. Describe las ventajas e inconvenientes principales entre MariaDB y PostgreSQL.

MariaDB cuenta con varias ventajas claramente marcadas, su simplicidad a la hora de utilizarla e implementarla seguido del hecho de ser de código abierto son las más llamativas. Da soporte a la mayor parte de casos de uso para una base de datos, también, es multiplataforma y corre sin problemas en cualquier sistema operativo.
En cuanto a sus desventajas, tiene muchos problemas en cuanto a escalados horizontales se refiere dado su alto coste. Su sistema de *backups* es flojo y suele dar problemas o no funcionar con grandes bases de datos. Además, no cuenta con *logfiles* en sus procesos de inicio lo que lleva a hacer *debugging* a mano si surge algún error.

Sobre las ventajas de **PostgreSQL**, ofrece mucha estabilidad con una velocidad de respuesta y una gestión de recursos excelente (aún en entornos de bases de datos complejas) con un coste de la máquina bajo. Ofrece una enorme customización que se amolda a todas las necesidades que un consumidor pueda tener, así como una adaptabilidad excelente para entornos distribuidos. También tiene un excelente gestor de *backups* y una muy buena seguridad.
Por otro lado, en cuanto a sus desventajas, es necesario destacar que su instalación es complicada y poco documentada, además, varía dependiendo del sistema operativo donde se instale. No hay documentación de las herramientas ni *plugins* compatibles existente. Además, en cuanto a la administración en sí misma de PostgreSQL, puede llegar a ser muy compleja y poco detallada, sobre todo en los fallos de seguridad y estabilidad.