

**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? 4](#_Toc88595882)

[2. Descarga el software desde la página oficial, indicando los pasos necesarios para descargarlo y descomprimirlo. 5](#_Toc88595883)

[3. Ejecuta la compilación teniendo en cuenta que nuestro directorio de instalación será /usr/local/pgsql. Describe el proceso que has seguido. 5](#_Toc88595884)

[4.¿Qué acciones complementarias has tenido que realizar para poder realizar la instalación? 6](#_Toc88595885)

[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? 8](#_Toc88595886)

[6. Lanza la ejecución del servicio teniendo en cuenta los parámetros configurados en el punto anterior. ¿qué comando has utilizado? 9](#_Toc88595887)

[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. 10](#_Toc88595888)

[8. ¿Qué es el init.d?¿Qué otras carpetas podemos encontrar similares? 11](#_Toc88595889)

[9. Configura el autovacuum de postgreSQL para que se ejecute automáticamente. ¿Qué archivo has tenido que modificar? 12](#_Toc88595890)

[10. Indica los principales parámetros y su utilidad del archivo anterior. 12](#_Toc88595891)

[11. Crea una nueva BD llamada test. Indica los pasos que has seguido para su creación. 13](#_Toc88595892)

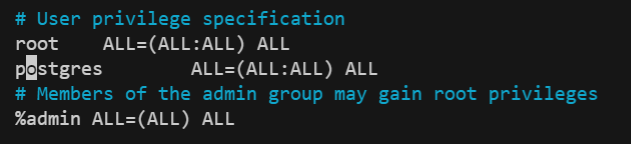
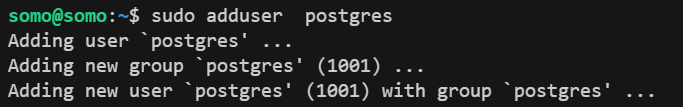
[12. Configura para que la BD creada en el punto anterior sea accesible sólo desde localhost utilizando las credenciales del usuario que creamos en el punto 1. Describe la configuración utilizada y el archivo que has modificado. 14](#_Toc88595893)

[13. ¿Qué otros parámetros de configuración se pueden utilizar en el archivo pg\_hba.conf? 15](#_Toc88595894)

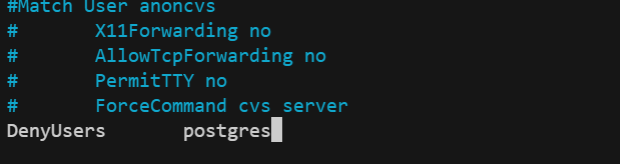
[14. Describe las ventajas e inconvenientes principales entre MariaDB y PostgreSQL. 15](#_Toc88595895)

# 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?

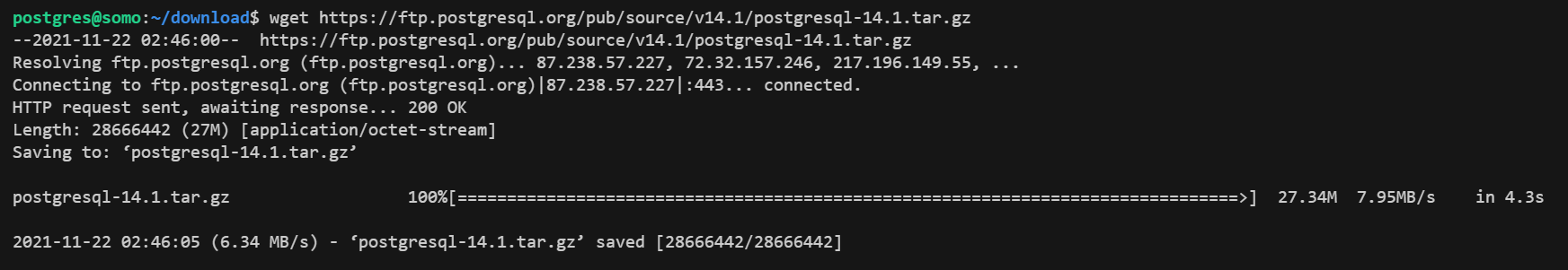
Con el comando ***sudo adduser*** nombre creamos un nuevo usuario, y para darle acceso de *sudoer*, se deberá modificar el archivo ***/etc/sudoers*** como super usuario y añadir **postgres** **ALL=(ALL:ALL) ALL.**

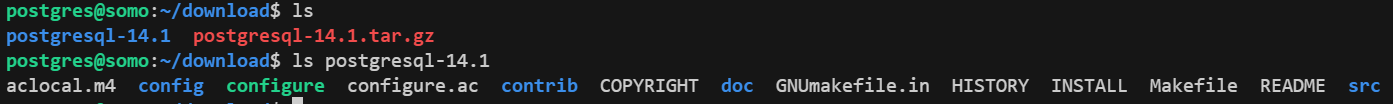


Finalmente, para deshabilitar las conexiones **SSH** para el usuario **postgres**, tendremos que modificar el fichero **/etc/ssh/sshd\_config** y añadir **DenyUsers postgres**. Cabe destacar que debe haber una tabulación entre *DenyUsers* y el usuario para que reconozca la directiva.



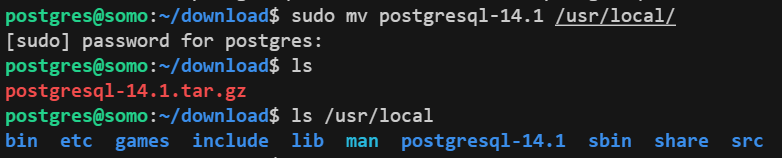
# 2. Descarga el software desde la página oficial, indicando los pasos necesarios para descargarlo y descomprimirlo.

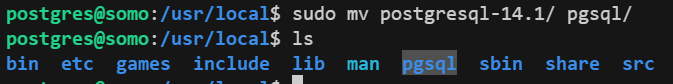
Para descargarlo, se hará uso del comando **wget** con el link del archivo **.tar** tal que **wget https://ftp.postgresql.org/pub/source/v14.1/postgresql-14.1.tar.gz.**

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

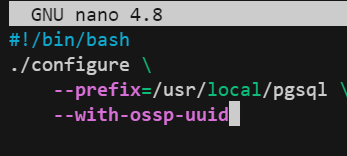
# 3. Ejecuta la compilación teniendo en cuenta que nuestro directorio de instalación será /usr/local/pgsql. Describe el proceso que has seguido.

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





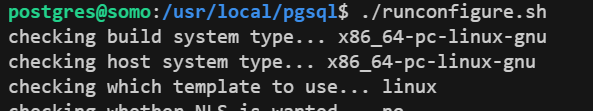
Para facilitar la compilación, se ha creado un script **runconfigure.sh** que actuará de *wrapper* para ayudar a la posterior configuración.

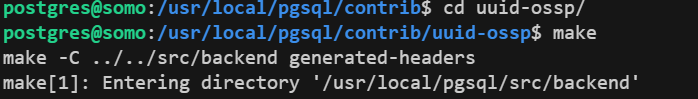
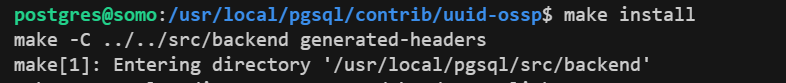


Para poder compliar el script, se deben instalar primero algunos paquetes para que sea posible:



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



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**). 

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**. 

# 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

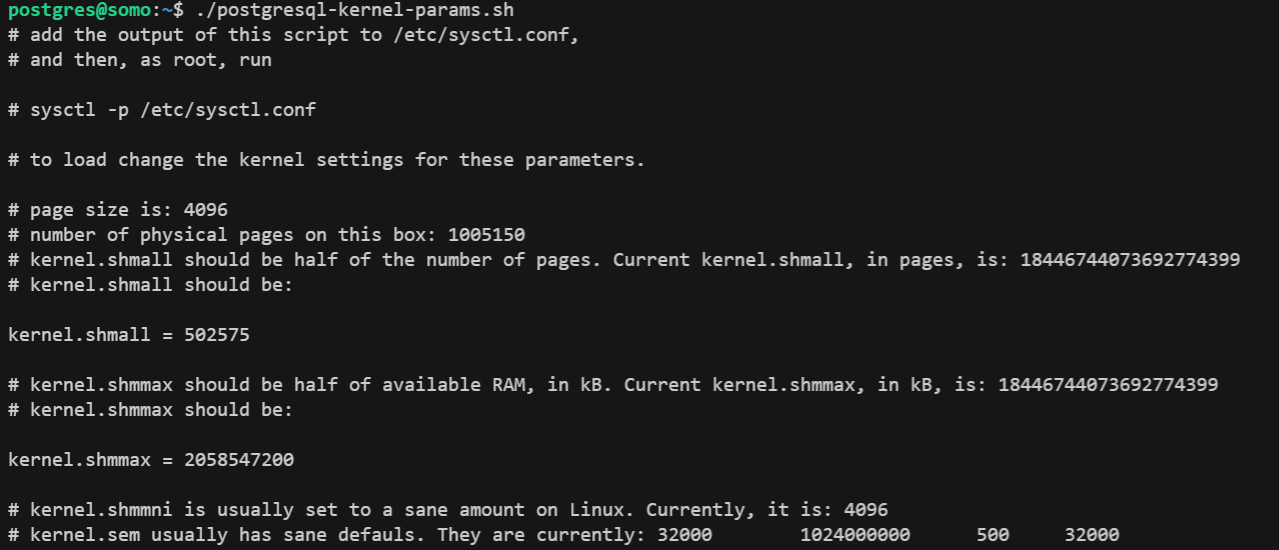
# CURR\_SHMMSL=`$SYSCTL -n kernel.semmap`  # XXX: does not exist on linux

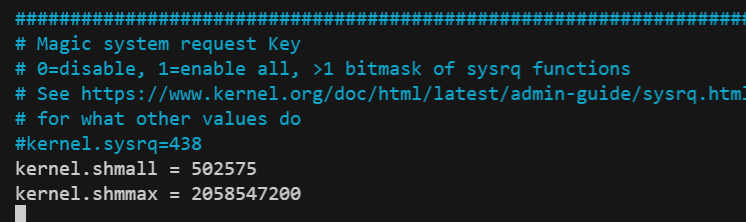
# CURR\_SHMMSL=`$SYSCTL -n kernel.semmvx`  # XXX: does not exist on linux

CURR\_SEM=`$SYSCTL -n kernel.sem`

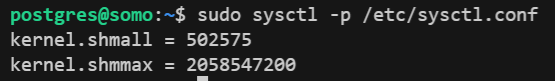
echo "# kernel.sem usually has sane defauls. They are currently: $CURR\_SEM"

que al ejecutar mostrará:



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

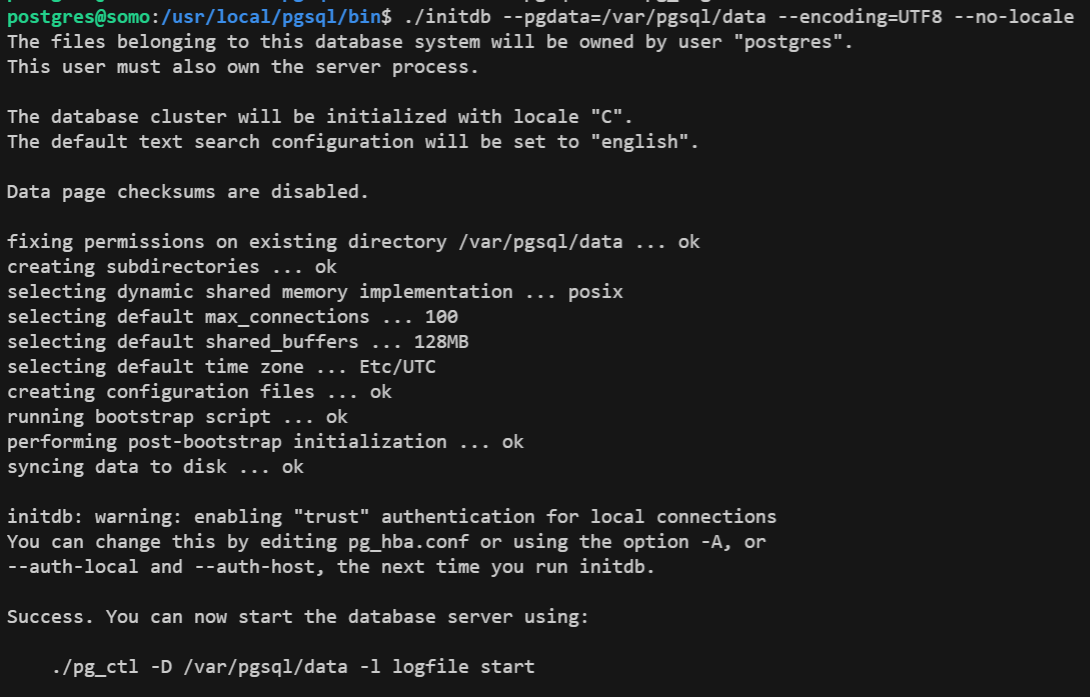
Y con el comando **sysctl -p /etc/sysctl.conf** aplicamos los cambios :



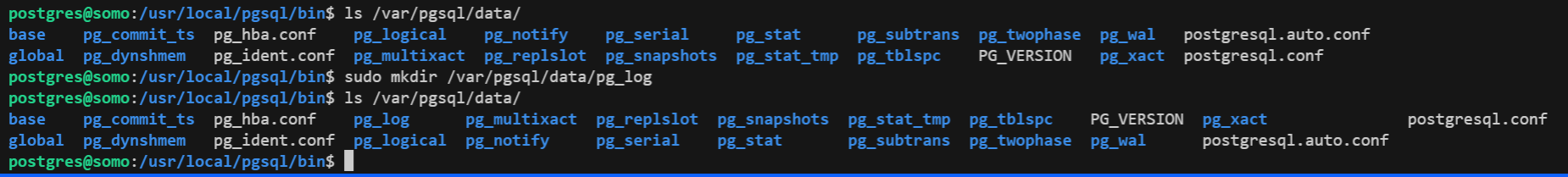
# 5. Crea la carpeta de datos en /var/pgsql/data y la carpeta de logs en /var/pgsql/data/pg\_log , asignándole como propietario el usuario creado en el punto 1. Adicionalmente inicializa la BD con codificación es\_ES.utf8 ¿Qué comando has utilizado para inicializarla?

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/**).



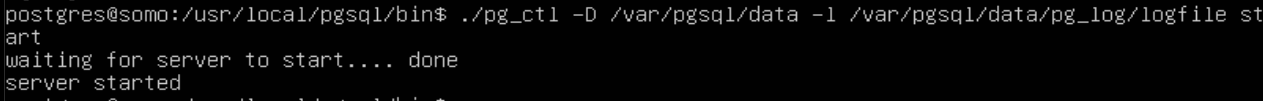
Para inicializar la BD, se deberá ir al directorio **/usr/local/pgsql/bin** y llamar a **initdb** , indicando el directorio de data y la codificación UTF8. El comando completo sería : **./initdb --pgdata=/var/pgsql/data --encoding=UTF8 --no-locale.**

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:



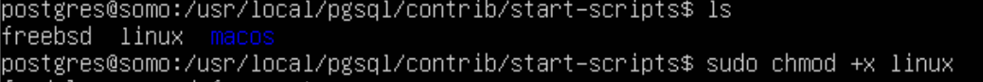
# 6. Lanza la ejecución del servicio teniendo en cuenta los parámetros configurados en el punto anterior. ¿qué comando has utilizado?

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**.



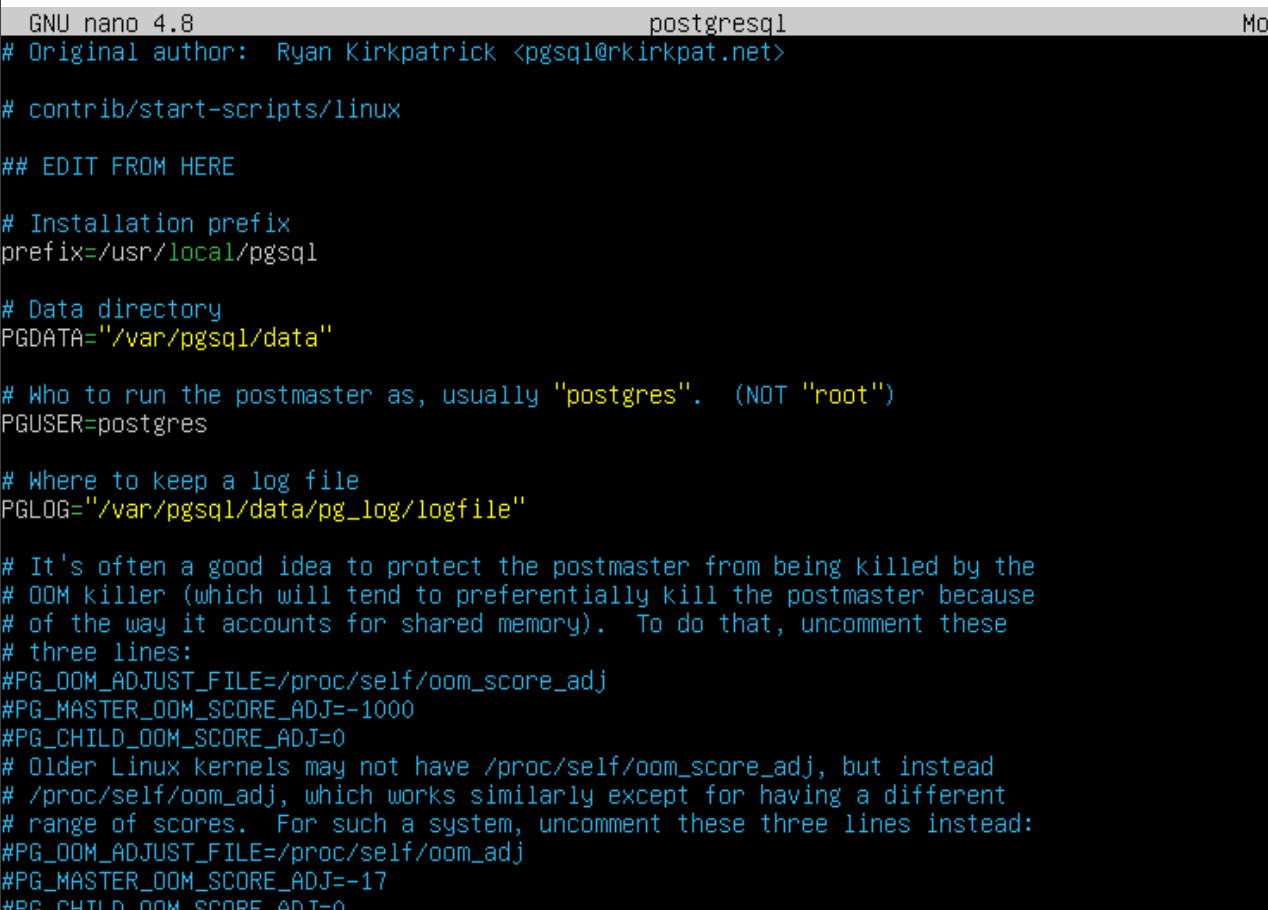
# 7. Implementa un archivo para que se inicie como un servicio automáticamente al arrancar el servidor. Adjunta el código del script agregando los comentarios oportunos.

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**.

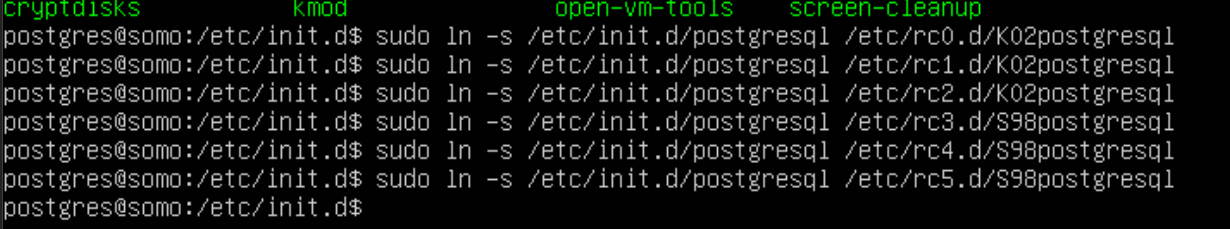




Y si con nano editaremos el fichero Linux, asignando correctamente los directorios de data y log de postgresql, **/var/pgsql/data** y **/var/pgsql/data/pg\_log/logfile** respectivamente.

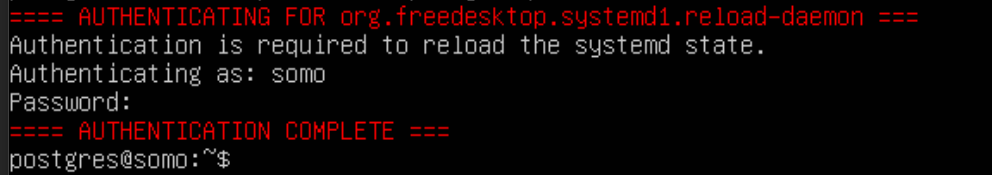


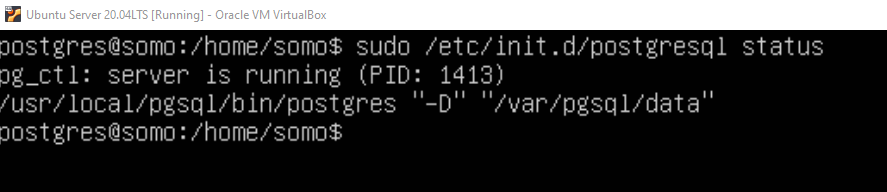
Como se explicará en el siguiente apartado, se debe crear enlaces entre el directorio recientemente creado **/etc/init.d/postgres** y cada uno de los fichers postgres (**K02postgresql** en rc0.d,rc1.d y rc2.d, y **S98postgresql** en rc3.d,rc4.d y rc5.d) para que funcione correctamente la ejecución en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



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





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. 

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

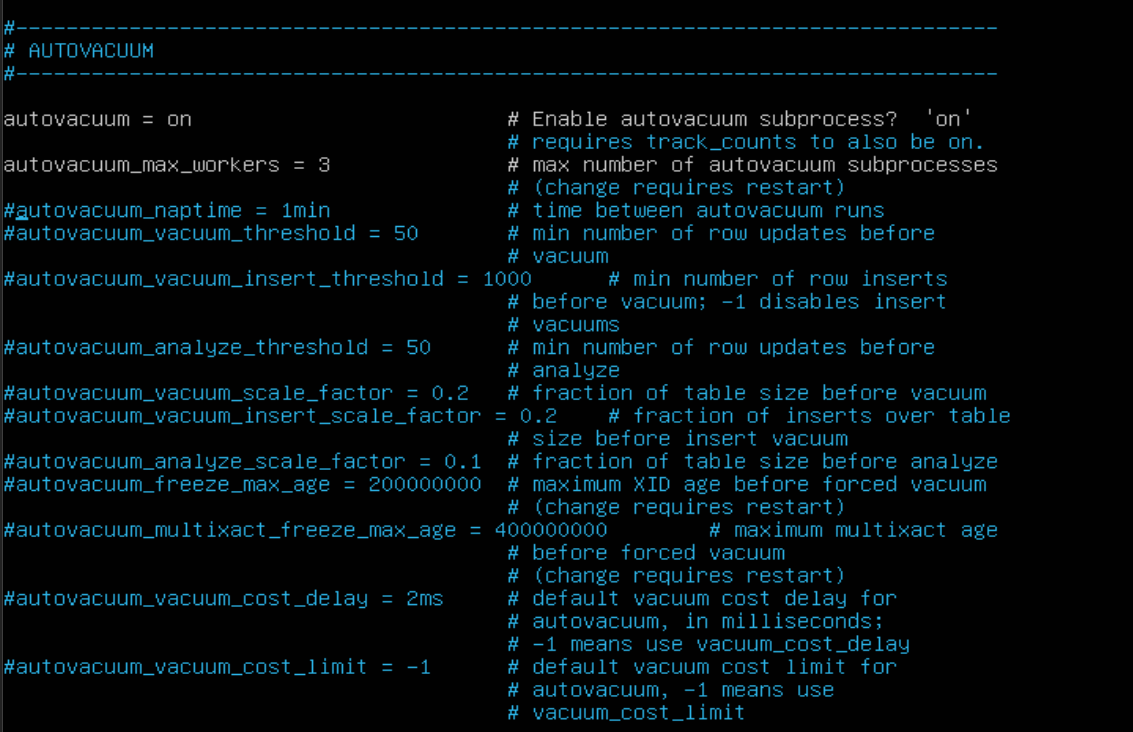
El **init.d** es el directorio donde se encuentran todos los scripts que se ejecutarán al arranque del sistema Linux. Se encuentra en **/etc/init.d** y es donde se especifican los comandos de arranque y parada de los servicios que contiene.

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. 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.



# 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.



# 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.

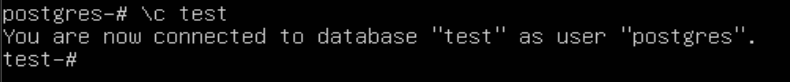
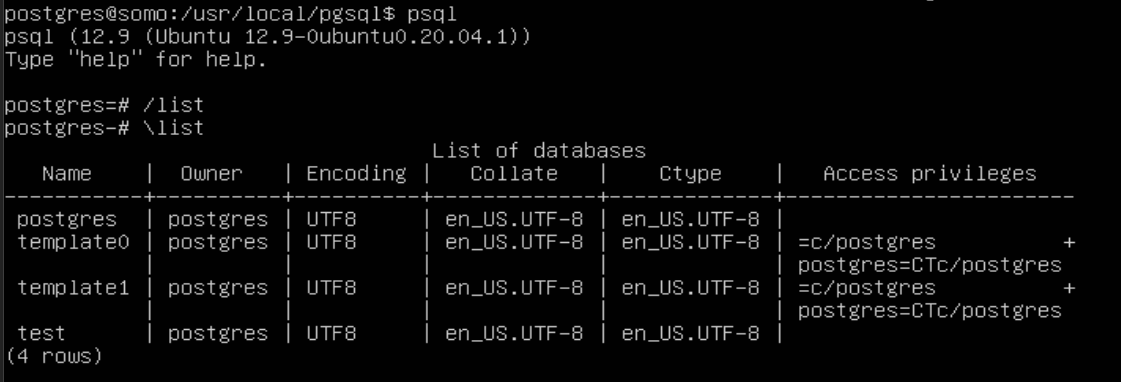
**autovacuum\_vacuum\_cost\_delay =2ms:** Especifica el tiempo de espera que se usará en operaciones vacuum automáticas. Si se delimita a -1 se usará el valor de vacuum\_cost\_delay.

**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.

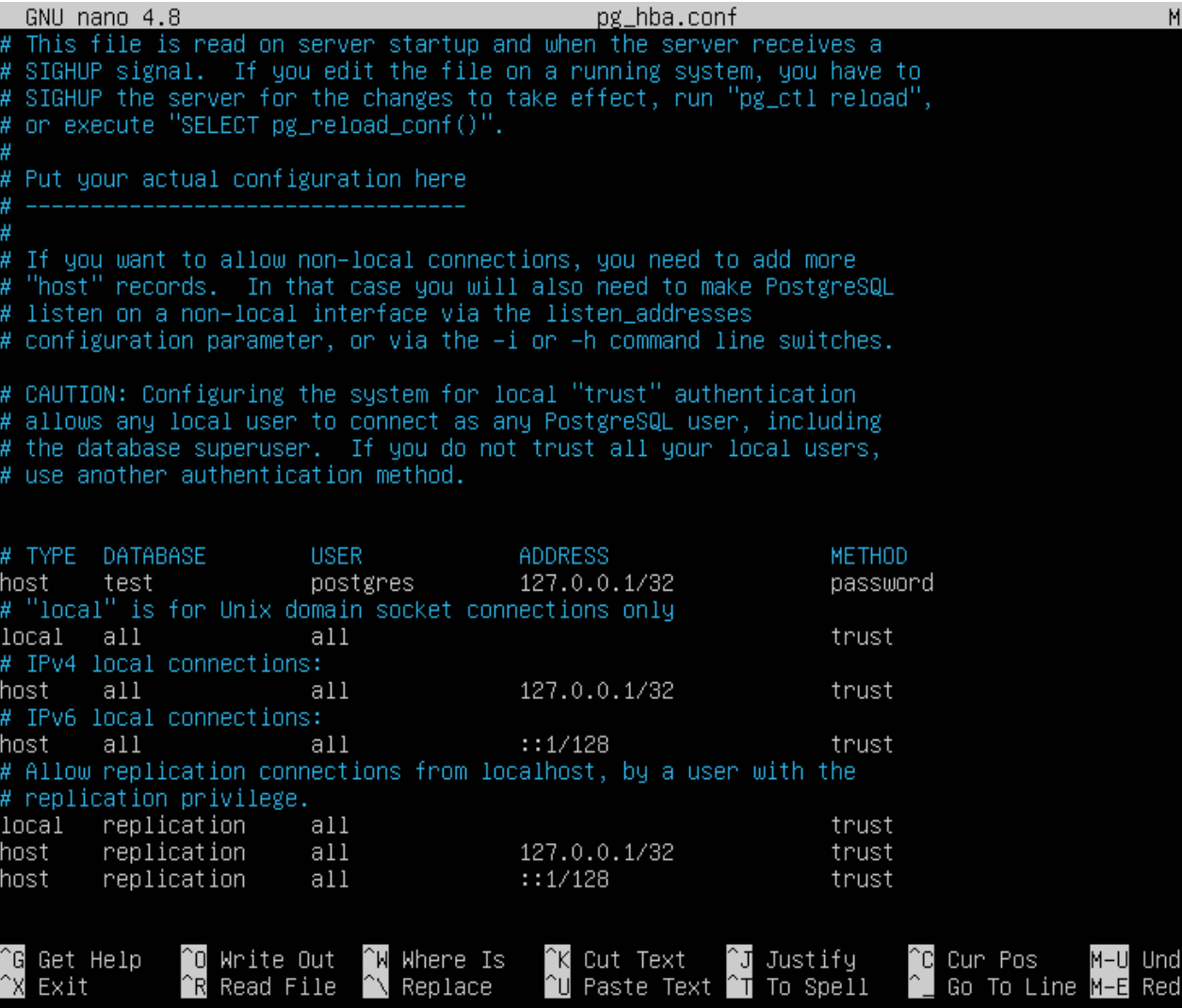
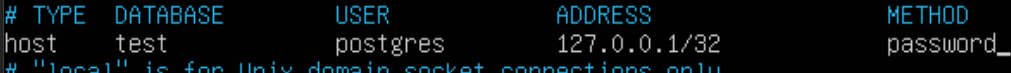
Para crear una nueva base de datos, bastaría con escribir el comando **createbd test** (siendo test el nombre de la bd) desde nuestro usuario **postgres**. Para comprobar que se ha creado correctamente, podemos acceder a **postgres** con **pgsql** desde el directorio **/usr/local/pgsql** y escribir el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





# 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.