

Configuración redes NAT

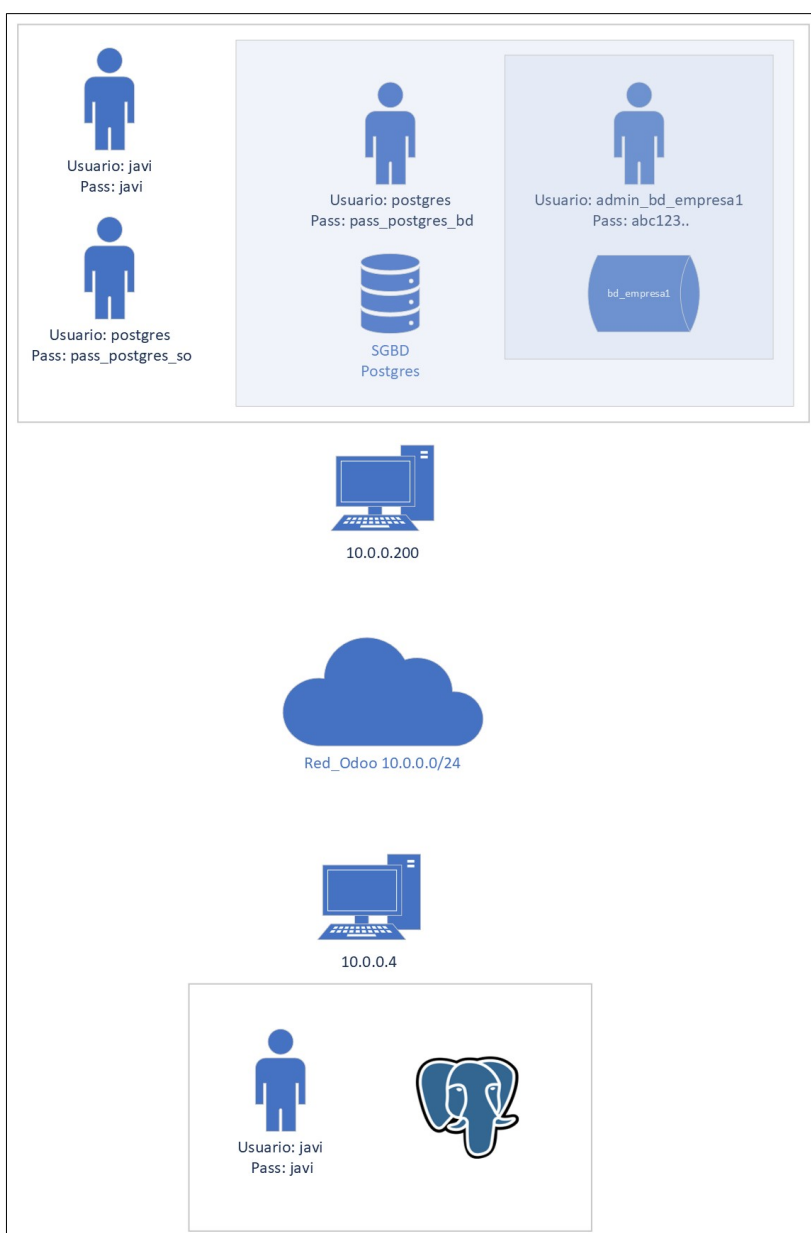
0. Escenario

Este tutorial creará el escenario con un servidor de bases de datos Postgres y un equipo cliente que se conectará al servidor.

Para ello usaremos dos máquinas virtuales Ubuntu y para aislar este escenario crearemos una red virtual NAT en VirtualBox llamada Red_Odoo configurada con la red 10.0.0.0/24 y un servidor DHCP.

Para la máquina virtual que hará de servidor asignaremos la IP manual 10.0.0.200/24, instalaremos el SGBD Postgres y configuraremos usuarios y servidor.

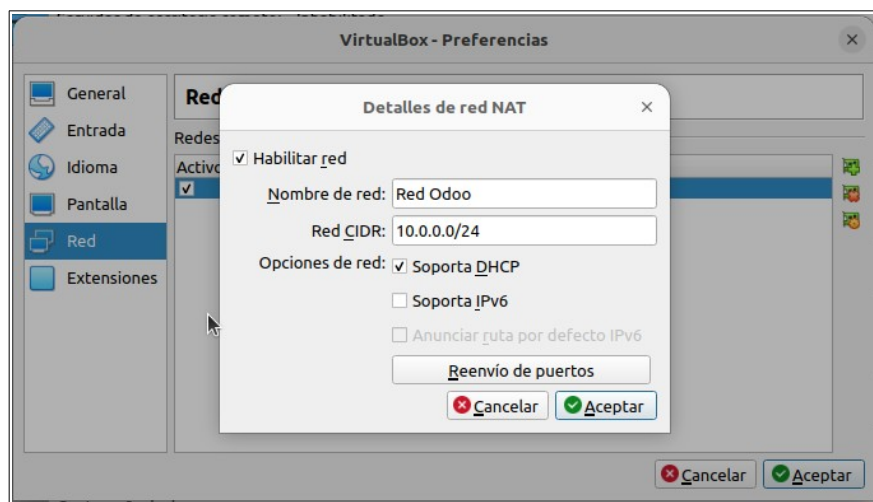
Para la máquina virtual del equipo cliente instalaremos pgAdmin4 que nos permitirá administrar la base de datos en remoto.



1.- Configuración VirtualBox

En este escenario necesitamos tener varias máquinas virtuales que se vean entre si. Para ello vamos a configurar una red NAT de manera que todas las máquinas se vean entre ellas, tengan conexión a internet pero que a la vez permanezcan aisladas de nuestra red real.

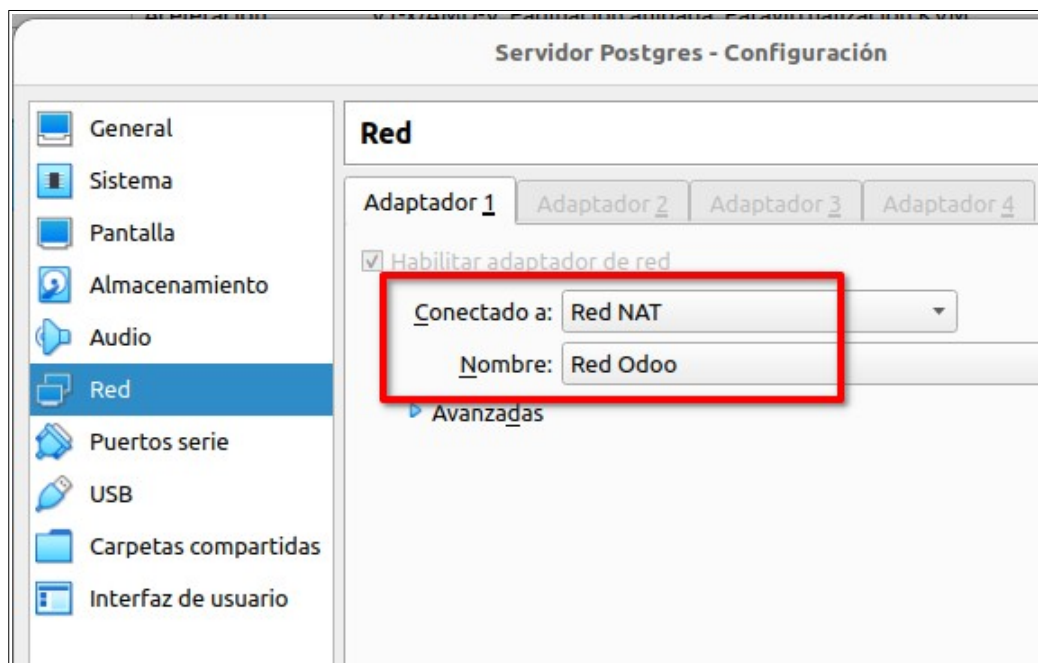
Para ello desde el menú Archivo / Preferencias seleccionamos el apartado Red y dentro de este creamos una nueva red con los siguientes datos:



Con esto hemos creado la red 10.0.0.0 con máscara 255.255.255.0 lo que nos permite disponer del último octeto para asignar hasta 254 IPs. En esta red algunas IPs están reservadas por VirtualBox como por ejemplo la 10.0.0.1 que hace de gateway y la 10.0.0.2 que hace de servidor DHCP.

Empecemos con el que será nuestro servidor de bases de datos independiente. Crea una nueva máquina virtual con solamente un Ubuntu 22.04 instalado en ella. Recuerda que puedes clonar alguna otra máquina virtual que tengas preparada.

En la configuración del servidor de bases de datos debes modificar el apartado de Red para incluir esta máquina en la red NAT que creamos antes.



Es necesario aplicar este cambio en la configuración a todas las máquinas que participen en este escenario.

2.- Configuración IP del servidor de bases de datos

Con la máquina virtual arrancada ejecutamos los siguientes comandos:

```
route1
```

Este comando muestra la lista de rutas de la máquina virtual. En mi caso puedo ver que la ruta por defecto, el gateway, es la 10.0.0.1 y que mi dirección ip local asignada es la 10.0.0.4.

Esta máquina funcionará como servidor así que debería tener una IP fija para que todo el resto de equipo sepa donde dirigirse. Lo propio sería manipular el servidor DHCP pero esto queda fuera del alcance de este módulo así que nos limitaremos a configurar una IP manual preferiblemente alta para tener menos probabilidades de que el DHCP la asigne a otra máquina.

```
sudo nano /etc/netplan/01-network-manager-all.yaml
```

Con este comando editamos el archivo de configuración del gestor de redes.

```
# Let NetworkManager manage all devices on this system
```

```
network:
  version: 2
  renderer: NetworkManager
  ethernet:
    enp0s3:
      addresses:
        - 10.0.0.200/24
      nameservers:
        addresses:
          - 8.8.8.8
          - 8.8.4.4
      routes:
        - to: default
          via: 10.0.0.1
```

Este es el contenido que debes dejar en el archivo de configuración que has abierto. Fíjate que hemos añadido líneas que tienen un indentado variable usando espacios, yo he usado 2 espacios.

En esas líneas configuramos la interfaz de la tarjeta enp0s3 asignándoles una dirección manual, un par de servidores DNS y una ruta por defecto.

Recuerda que en nano, con Ctrl+X guardas los cambios.

```
sudo netplan apply
```

Este comando recarga la configuración de red para que los cambios que has hecho surjan efecto inmediato.

```
route
```

Volvemos a usar este comando para confirmar que los cambios están bien hechos. Si la ruta por defecto sigue siendo 10.0.0.1 y la dirección local 10.0.0.200 será señal de que todo está bien.

3. Instalación servidor de bases de datos Postgres

```
sudo apt update
```

Comprobamos los repositorios en busca de nuevas versiones de los paquetes instalados.

```
sudo apt upgrade
```

Actualizamos a su última versión los paquetes que tengamos instalados.

```
sudo apt install postgresql postgresql-contrib
```

Este comando instalará el sistema gestor de bases de datos Postgres. Recuerda que con la instalación también se creará un usuario "postgres" en el sistema operativo y otro distinto pero de mismo nombre "postgres" en el sistema gestor de bases de datos. Como medida de seguridad (y didáctica) vamos a poner contraseñas a estos dos usuarios.

```
sudo passwd postgres
```

Este comando cambia la contraseña al usuario del sistema operativo postgres. Te propongo que le pongas "pass_postgres_so" en referencia al tipo de usuario que es. Probablemente te aparezca un warning diciendo que la contraseña no es apropiada por contener el nombre del usuario en la misma contraseña. Es cierto, pero en este ejemplo didáctico nos lo podemos permitir.

```
sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD  
'pass_postgres_bd';"
```

Con este comando modificamos el usuario postgres del sistema gestor de bases de datos Postgres para cambiarle la contraseña que, hasta ahora no tenía. Te propongo "pass_postgres_bd" para que la diferencias del usuario local del sistema operativo.

4. Configuración del servidor Postgres

El sistema gestor de bases de datos Postgres tiene dos archivos de configuración interesantes

- /etc/postgresql/14/main/postgresql.conf que trata aspectos genéricos técnicos del servidor.
- /etc/postgresql/14/main/pg_hba.conf que trata configuraciones referentes a la autenticación desde distintos hosts.

Para nuestro escenario debemos ver estos archivos como las IPs a las que escuchamos para el primer archivo de configuración y el tipo de login para el segundo archivo de configuración.

En postgresql.conf podemos configurar las IPs desde las que se admiten conexiones, el puerto de escucha, número máximo de conexiones, aspectos relacionados con la seguridad, la memoria, escritura, archivos log, consumo de recursos, etc...

En nuestro caso debemos habilitar las escuchas remotas de clientes desde otras IPs distintas a localhost. Podemos insertar, separadas por comas, todas las IPs desde las que permitimos clientes. También pode-

mos poner un asterisco (*) para permitir desde todas las interfaces. La IP 0.0.0.0 permitirá conexiones desde cualquier IPv4, dos dos puntos (::) permitirá aceptar cualquier conexión IPv6.

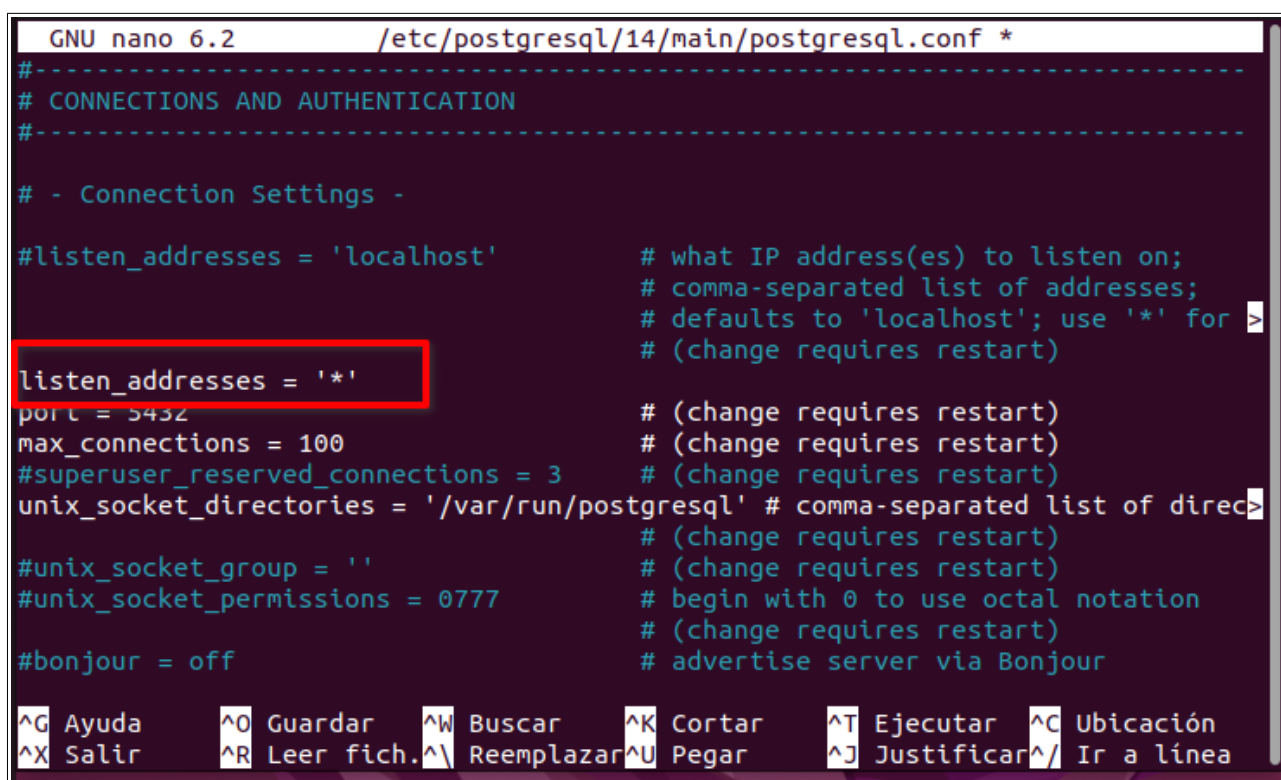
```
sudo nano /etc/postgresql/14/main/postgresql.conf
```

Con este comando abrimos el archivo de configuración del servidor en un editor de texto de terminal.

Para que el servidor escuche a cualquier dirección IPv4 hay que añadir la siguiente línea preferentemente en el apartado de "CONNECTIONS AND AUTHENTICATION"

```
listen_addresses = '*'
```

Guardamos con Ctrl+X.



```
GNU nano 6.2 /etc/postgresql/14/main/postgresql.conf *
#-----
# CONNECTIONS AND AUTHENTICATION
#-----
# - Connection Settings -
#listen_addresses = 'localhost'      # what IP address(es) to listen on;
#                                     # comma-separated list of addresses;
#                                     # defaults to 'localhost'; use '*' for
#                                     # (change requires restart)
listen_addresses = '*'
port = 5432                          # (change requires restart)
max_connections = 100                # (change requires restart)
#superuser_reserved_connections = 3  # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of direc
#                                     # (change requires restart)
#unix_socket_group = ''              # (change requires restart)
#unix_socket_permissions = 0777     # begin with 0 to use octal notation
#                                     # (change requires restart)
#bonjour = off                       # advertise server via Bonjour
```

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
 ^X Salir ^R Leer fich. ^_ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea

```
sudo nano /etc/postgresql/14/main/pg_hba.conf
```

Con este comando abrimos el archivo de configuración del servidor en un editor de texto de terminal.

Para que el servidor permita autenticaciones a cualquier dirección IPv4 hay que añadir la siguiente línea al final del archivo y guardar.

```
host all all 0.0.0.0/0 password
```

```
# Database administrative login by Unix domain socket
local all postgres peer

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256

#IPv4 remotas permitidas
host all all 0.0.0.0/0 password
```

```
sudo service postgresql restart
```

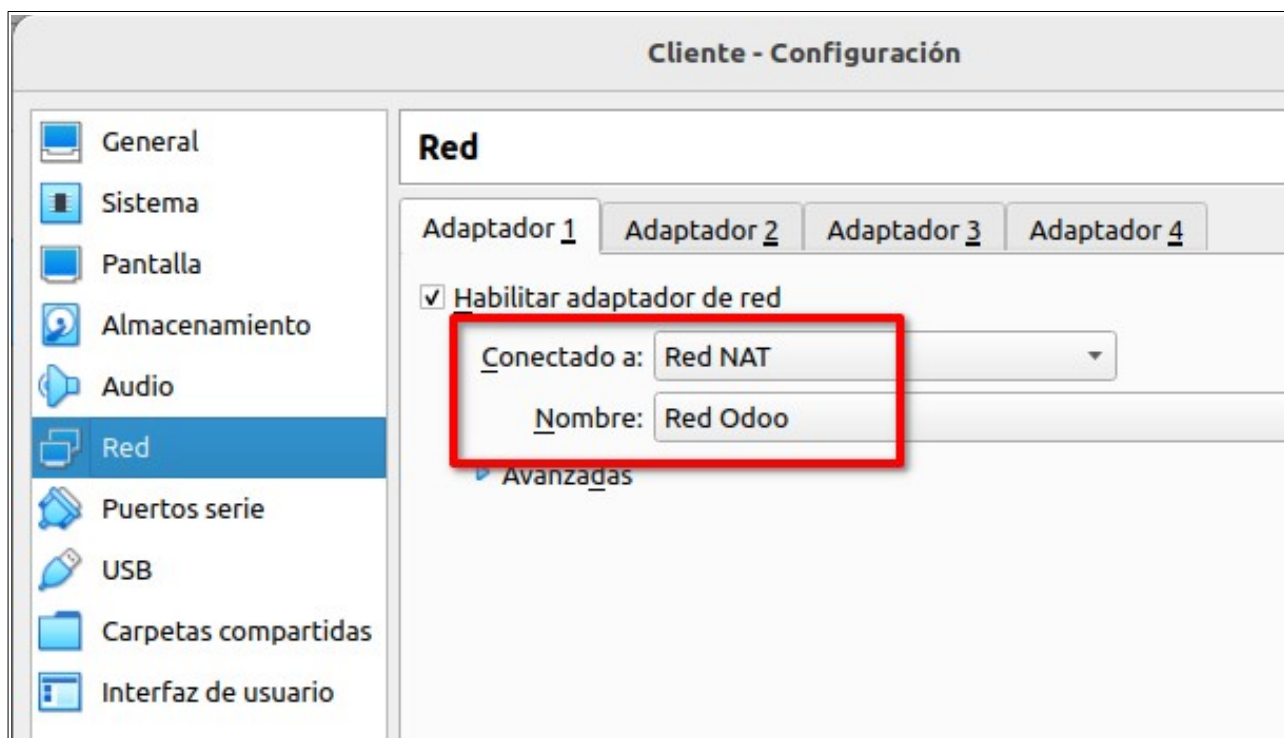
Al hacer cambios en los archivos de configuración es necesario reiniciar el sistema gestor de bases de datos Postgres. Como está instalado como un servicio se puede realizar mediante el comando anterior.

El puerto de escucha por defecto que usa Postgres es el 5432 pero podemos cambiarlo si es necesario. La manera de proceder sería igual que el anterior caso. Se cambia en el archivo de configuración, se guardan los cambios y se reinicia el servicio.

5. Configuración del equipo cliente

En primer lugar clonamos o volvemos a instalar un Ubuntu 22.04 en una máquina virtual. Para el cliente podemos reducir drásticamente los requisitos del memoria y así no sobrecargar innecesariamente el ordenador anfitrión de VirtualBox.

En la configuración de la máquina virtual dentro de VirtualBox será necesario modificar el apartado de red para que se conecte a la red NAT que creamos en pasos anteriores.



Con la máquina arrancada vamos a instalar la aplicación pgAdmin4, una aplicación python que permite administrar una base de datos Postgres de manera gráfica que usaremos para conectar en remoto contra el servidor.

```
sudo apt install curl
```

Con este comando instalamos la aplicación curl

```
curl -fsSL https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo gpg  
--dearmor -o /etc/apt/trusted.gpg.d/pgadmin.gpg
```

Con el comando anterior descargamos la llave que permitirá confiar en las descargas del servidor de pgadmin

```
sudo sh -c 'echo "deb  
https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/jammy pgadmin4 main" >  
/etc/apt/sources.list.d/pgadmin4.list'
```

Con este comando añadimos el repositorio oficial de pgAdmin a la lista de repositorios de nuestro sistema.


```
sudo apt update
```

Con este comando refrescamos los repositorios. En este caso necesario por haber añadido uno nuevo.

```
sudo apt upgrade
```

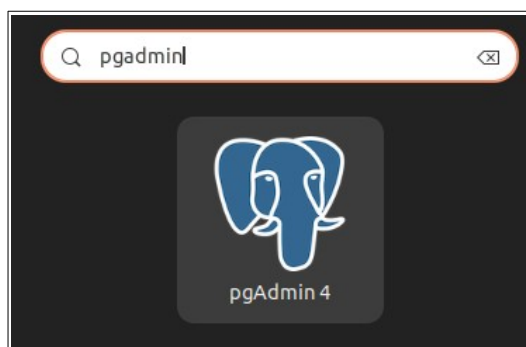
Con este comando actualizamos las versiones antiguas de los paquetes que tenemos instalados a las más recientes en los repositorios.

```
sudo apt install pgadmin4-desktop
```

Finalmente con este comando instalamos la aplicación pgAdmin en su versión de escritorio para ejercer de cliente.

6. Configuración de pgAdmin4 en el cliente

Desde el lanzador de aplicaciones buscamos pgAdmin4 y lo ejecutamos como una aplicación normal.



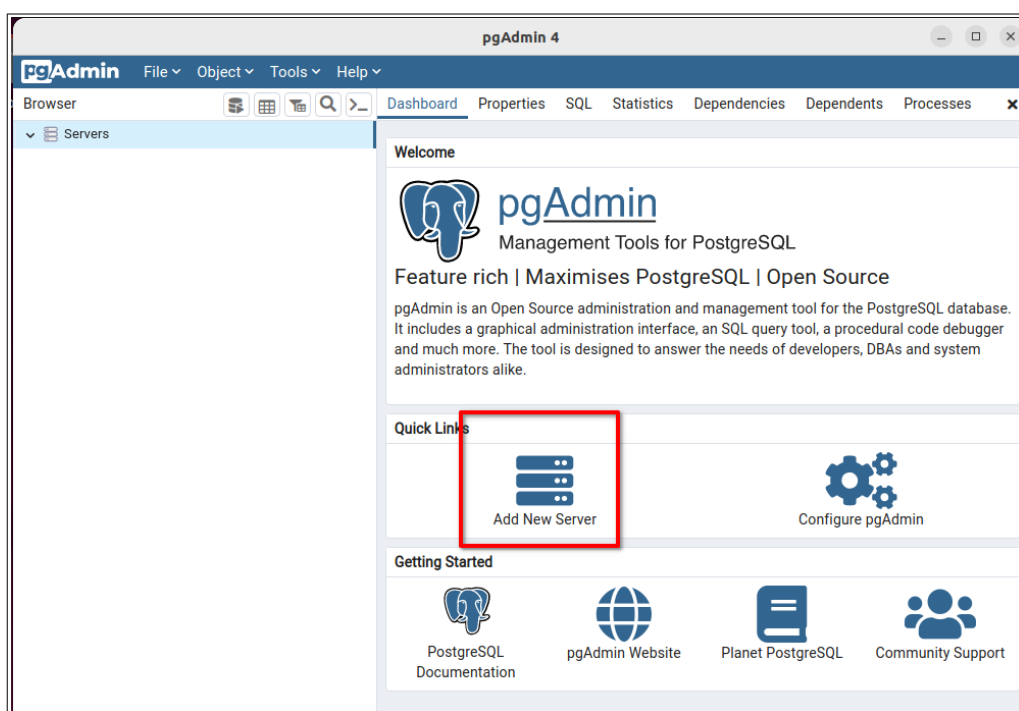
En la primera ventana que aparece al cargar pgAdmin4 nos pregunta por una Master Password que servirá para mantener cifradas las contraseñas de los servidores a los que nos conectemos. Posteriormente, en cada nuevo arranque nos la preguntará para usar las contraseñas almacenadas. Si no recordamos la master password podemos cambiarla pero perderemos las contraseñas de los servidores y tendremos que volver a configurarlas.

Set Master Password

Please set a master password for pgAdmin.
This will be used to secure and later unlock saved passwords and other credentials.

?
Cancel
OK

Desde el menú Object / Register / Server o desde el icono de la ventana principal podemos crear una nueva conexión a un servidor de bases de datos Postgres

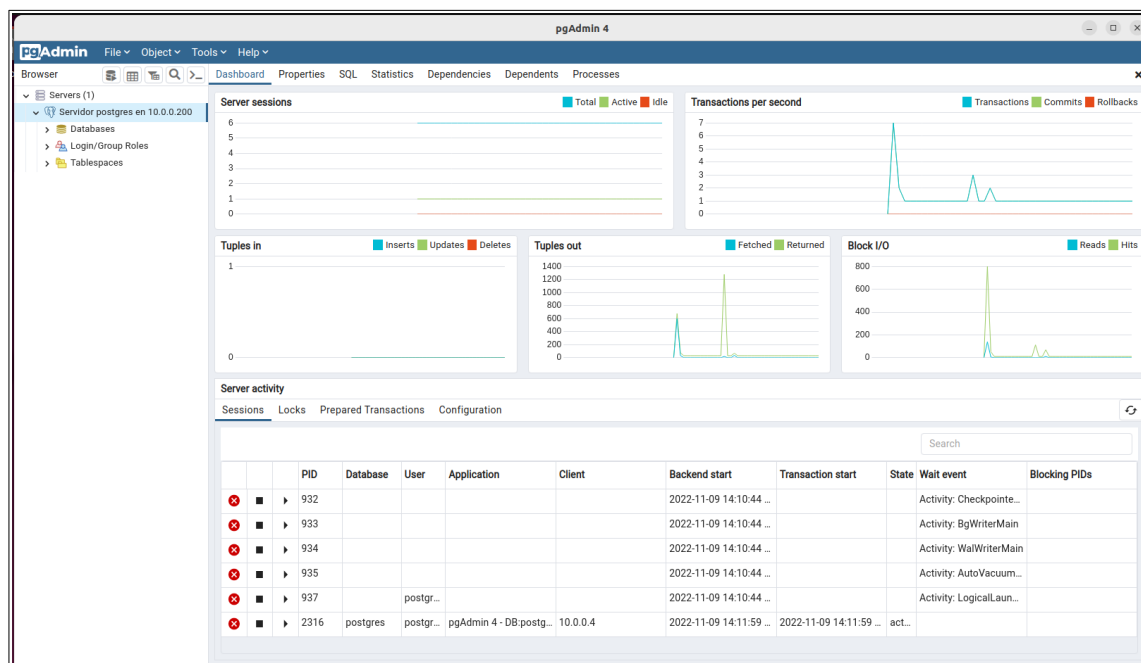


En la pestaña General le pondremos un nombre a la conexión, algo significativo que nos ayude a identificarla claramente incluso si tuviéramos muchas conexiones a diferentes bases de datos configuradas.

En la pestaña Connection tendremos que poner la IP del servidor, el puerto de escucha, el nombre de la base de datos, el usuario y su contraseña.

La IP será la 10.0.0.200, el puerto seguirá siendo el por defecto 5432, la base de datos postgres, usuario postgres y contraseña pass_postgres_bd.

Guardamos e intentará automáticamente una conexión con la base de datos. Si hubiera algún problema nos lo indicará en esa misma ventana.



Si todo ha ido bien aparecerá un panel como el de la imagen con todas las bases de datos a las que tenemos acceso con ese usuario.