

MPT03. DESPLIEGUE DE SERVICIOS EN RED

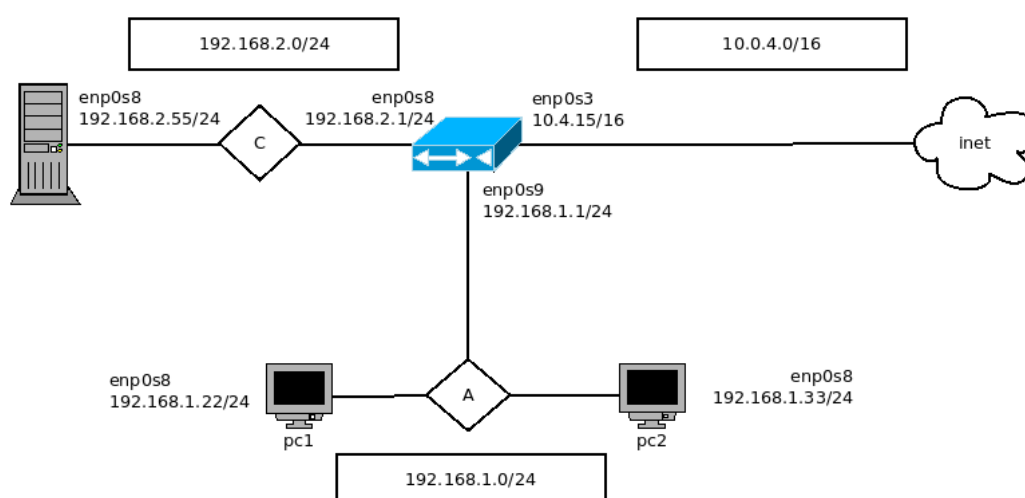
1. ESPECIFICACIÓN

En este trabajo pondremos en práctica el despliegue de servicios en una red privada aislada de distintas aplicaciones como las que hemos usado ya con anterioridad. La red propuesta es la siguiente:

- Estamos creando la red para dos socios autónomos que han decidido montar una tienda de equipación deportiva y han decidido implantar un servidor con unos servicios determinados.
- Esta red está formada por dos únicos **pcs** (a los cuales se podrían añadir más nodos), un **router** principal y un servidor. El **router** está conectado a internet mediante distintos routers (los cuales vamos a obviar ya que la idea es minimizar la demostración)
- Los **pcs** están conectados al mismo dominio de colisión ya que van a realizar la misma función. Ambos pueden conectar con el servidor con total libertad.
- El servidor tiene desplegado un **apache2**, un **postgresql**, un **opendldap** y un **stfp** (gestionado por **phpldapadmin**), de forma que se pueda iniciar sesión con los usuarios en el openldap en postgresql. El **apache2** tiene una página web la cual dispone de un catálogo de artículos de equipación deportiva y se pueden hacer pedidos **online**. Para ello, desde un cliente pueden acceder única y exclusivamente a los puertos correspondiente únicamente para poder mirar la tienda y hacer los pedidos correspondientes con unos usuarios que pueden poseer por hacerse socios de la tienda.

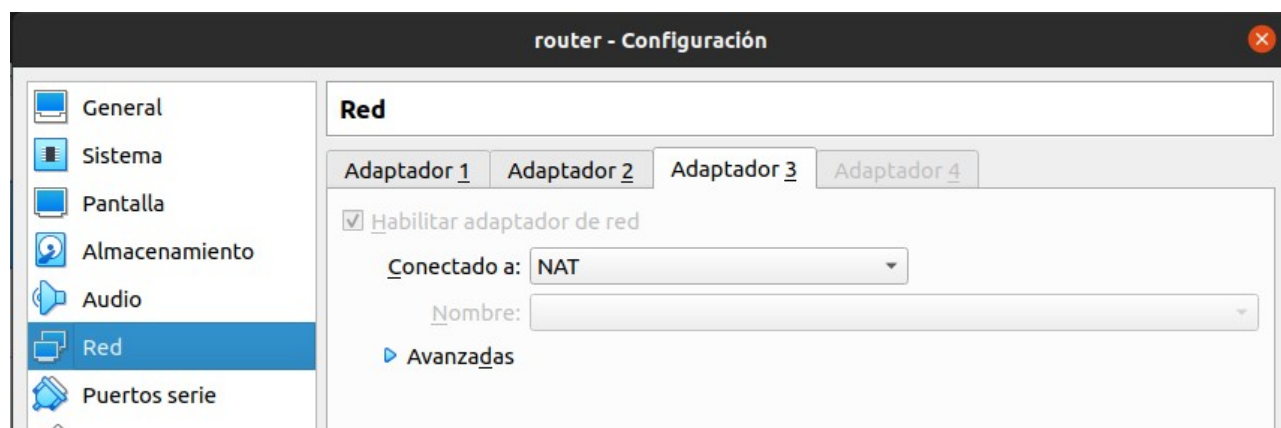
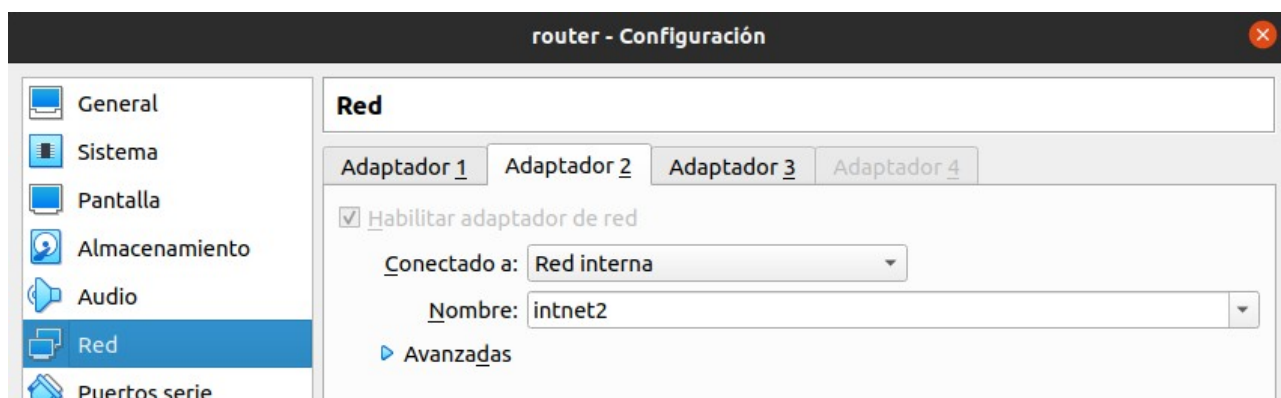
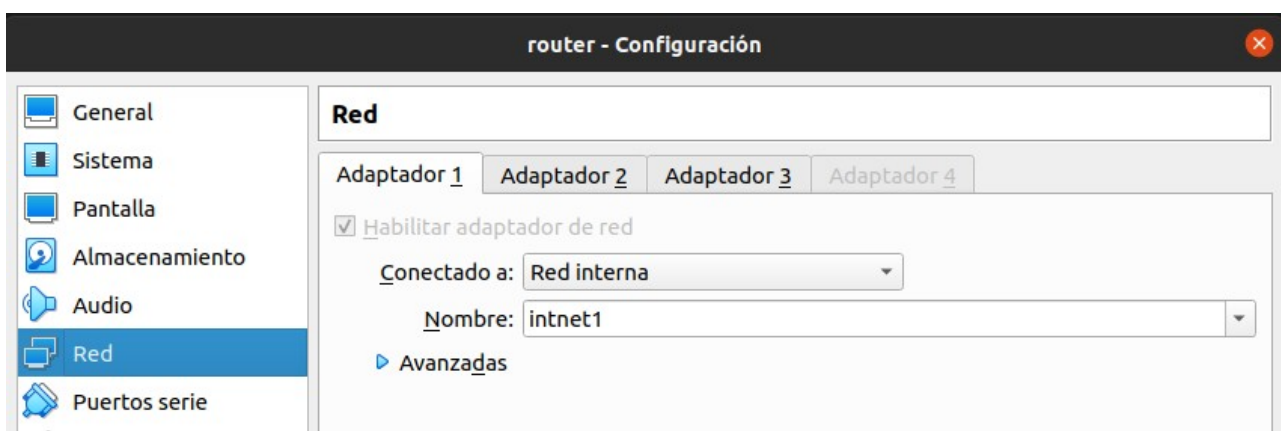
2. TOPOLOGÍA DE LA RED

La topología de la red física es la siguiente:



Y la topología lógica para simular en la medida de lo posible es la siguiente (EN VIRTUALBOX):

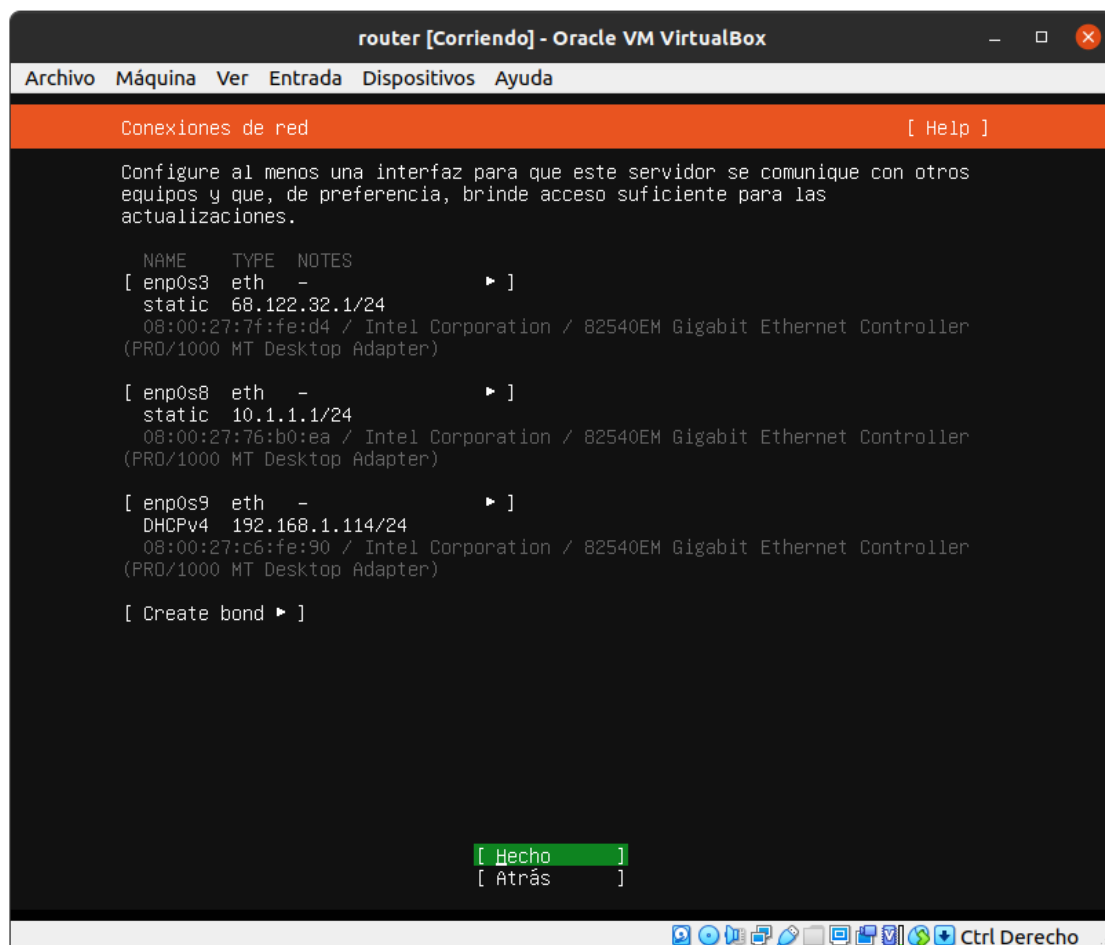
- Vamos a crear 4 máquinas virtuales para esta demostración
- El router tendrá un Ubuntu 20.04 Server y el resto Ubuntu 20.04 Desktop (El servidor puede ser Server, pero usaré el modo Escritorio para poder visualizar que se cumplen ciertas modificaciones de manera gráfica)
- Cada nodo tiene que tener un adaptador **NAT** (que le dará conexión a internet), y un adaptador con red INTERNA. Cada red **INTERNA** representa los dominios de difusión, por lo que serán dos redes internas. El router tendrá las dos redes internas **intnet1** e **intnet2**, server tendrá la primera y los pcs tendrán la segunda.



Con todo esto listo, podemos empezar instalando los sistemas operativos y preparamos los nodos.

3. PREPARACIÓN DE LOS NODOS

En primer lugar empezaremos con la instalación del router, el cual será de Ubuntu 20.04 Server. La instalación se hará con total normalidad, pero habrá que tener en cuenta el siguiente paso:



Dos interfaces corresponderán a los adaptadores de las redes internas. Cada una de ellas ha de ser configurada de manera estática y tener los adaptadores con las direcciones IP que vayamos a usar (en esta captura tiene otras IPs debido a ciertas pruebas, en este caso serán **192.68.1.1** y **192.168.2.1** respectivamente). La otra interfaz seguirá por DHCP ya que es de la NAT.

Una vez instalado nuestro sistema operativo, debemos activar el `ip_forward` para poder usar esta máquina como **router**. Se activa primero, quitando el comentario de la siguiente línea de `sysctl.conf` en la carpeta etc:

```
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

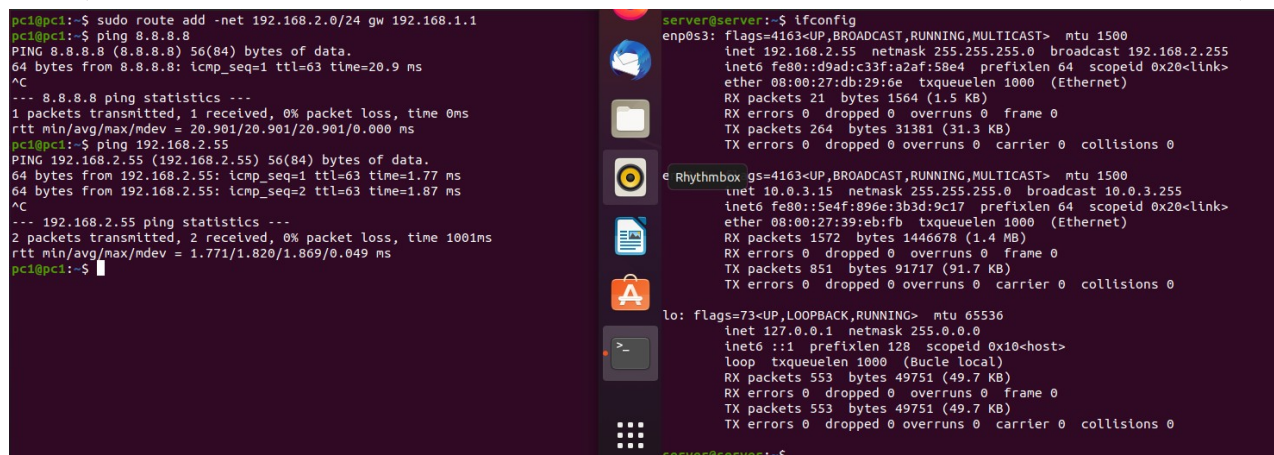
Y luego, como **root**, ejecutar el siguiente comandos:

```
root@router:/home/router# echo 1 > /proc/sys/net/ipv4/ip_forward
root@router:/home/router# sysctl -p
net.ipv4.ip_forward = 1
```

Con todo esto, el router está listo. Posteriormente añadiremos las reglas de iptables para el cortafuegos.

A continuación seguiremos con los pcs. Se instalarán con completa normalidad y, una vez iniciadas las máquinas, **cambiaremos las direcciones IPs manualmente** en la configuración de Ubuntu con sus máscaras y puerta de enlace:

Si añadimos el enrutamiento entre el servidor y el pc1 podemos ver que se puede hacer ping (no trabajo con el pc2 ya que es exactamente igual que pc1 pero cambiando la IP):



The screenshot shows two terminal windows. The left window is on 'pc1' and shows the command 'sudo route add -net 192.168.2.0/24 gw 192.168.1.1' being executed. It then shows two successful ping tests: one to 8.8.8.8 and another to 192.168.2.55. The right window is on 'server' and shows the output of 'ifconfig' for three interfaces: 'enp0s3', 'e Rhythmbox', and 'lo'. Each interface shows its IP address, netmask, broadcast address, and other network statistics. The 'enp0s3' interface has IP 192.168.2.55, 'e Rhythmbox' has IP 10.0.3.15, and 'lo' has IP 127.0.0.1.

Las máquinas se comunican perfectamente.

4. INSTALACIÓN DE LOS SERVICIOS

Como mencioné anteriormente, vamos a instalar ciertos servicios que vamos a usar en nuestra red. Primer vamos a empezar con el **openldap**. Para ello instalamos lo siguiente:

```
server@server:~$ sudo apt install slapd ldap-utils
Leyendo lista de paquetes... Hecho
C
  Cliente de correo Thunderbird cias
L
  estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libldap-2.4-2 libodbc1
```

Nos pedirá la contraseña del usuario **root**, pondremos la que queramos...

Luego tendremos que configurar el archivo hosts y añadiremos el servidor que vamos a usar:

```
GNU nano 4.8 /etc/hosts
127.0.0.1    localhost
127.0.1.1    server
192.168.2.55 server.local  ldapserver
# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

E instalamos la librería libnss-ldap y completamos los campos con la dirección IP del servidor, el dominio y contraseña del usuario administrador del servidor.

Configuración de slapd

El nombre de dominio DNS se utiliza para construir el DN base del directorio LDAP. Por ejemplo, si introduce «foo.example.org» el directorio se creará con un DN base de «dc=foo, dc=example, dc=org».

Introduzca el nombre de dominio DNS:

server.local

<Aceptar>

Después de rellenar todos los campos modificamos el archivo de **ldap.conf** para añadir los parámetros de nuestro servidor:

```
GNU nano 4.8 /etc/ldap/ldap.conf Modificado
#
# LDAP Defaults
#
# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE      dc=server,dc=local
URI        ldap://192.168.2.55:389

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never

# TLS certificates (needed for GnuTLS)
TLS_CACERT      /etc/ssl/certs/ca-certificates.crt
```

Activamos los puertos (los cuales desactivaremos luego y solamente dejaremos activados por iptables):

```
server@server:~$ sudo ufw allow 80
Reglas actualizadas
Reglas actualizadas (v6)
server@server:~$ sudo ufw allow 389
Reglas actualizadas
Reglas actualizadas (v6)
server@server:~$
```

Y comprobamos que openldap se ha instalado con éxito:

```
#
# LDAPv3
# base <dc=server,dc=local> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# server.local
dn: dc=server,dc=local
objectClass: top
objectClass: dcObject
objectClass: organization
o: server
dc: server

# admin, server.local
dn: cn=admin,dc=server,dc=local
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
server@server:~$
```

Ahora instalaremos **phpldapadmin** para administrar nuestros usuarios de ldap:

```
server@server:~$ sudo apt install phpldapadmin -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libapache2-mod-php7.4 php php-common php-ldap php-xml php7.4 php7.4-cli
  php7.4-common php7.4-json php7.4-ldap php7.4-opcache php7.4-readline
  php7.4-xml
Paquetes sugeridos:
```

Añadimos el **enlace simbólico**:

```
server@server:~$ sudo ln -s /usr/share/phpldapadmin/ /var/www/html/phpldapadmin
server@server:~$
```

Y cambiamos ciertos parámetros en el archivo de configuración para que funcione:


```

    assume UTC if you have not set PHP date.timezone. */
// $config->custom->appearance['timezone'] = null;
# $config->custom->appearance['timezone'] = 'Europe/Madrid';

/*****
 * Servers
 */

```

```

$servers->setValue('login','bind_id','cn=admin,dc=server,dc=local');
# $servers->setValue('login','bind_id','cn=Manager,dc=example,dc=com');

```

```

$servers->setValue('server','host','192.168.2.55');

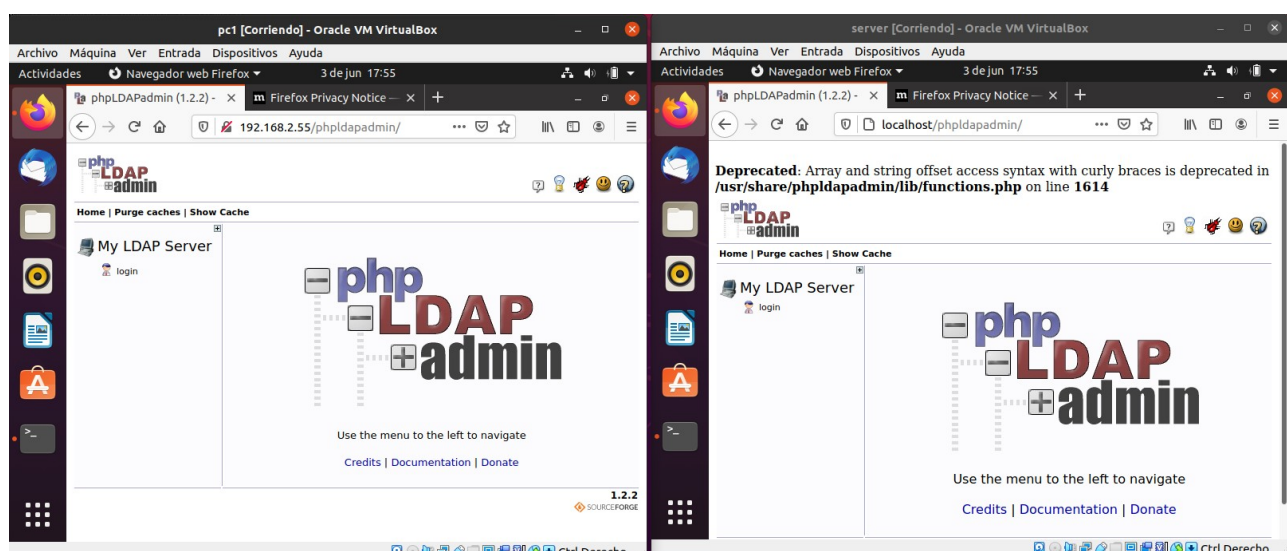
/* The port your LDAP server listens on (no quotes). 389 is standard. */
// $servers->setValue('server','port',389);

/* Array of base DN's of your LDAP server. Leave this blank to have phpLDAPadmin
   auto-detect it for you. */
$servers->setValue('server','base',array('dc=server,dc=local'));

```

(He aligerado la instalación ya que es algo con lo que hemos trabajado bastante)

Con esto listo, si entramos en nuestro navegador y escribimos la dirección IP seguido de /phpldapadmin desde cualquier nodo de la red podremos meternos:



Iniciamos sesión, creamos nuestra organización, grupo posix y usuario:



Finalmente, terminamos con la instalación de **postgresql**...

```
server@server:~$ sudo apt install postgresql-12
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libllvm10 libpq5 postgresql-client-12 postgresql-client-common
  postgresql-common sysstat
Paquetes sugeridos:
  postgresql-doc-12 libjson-perl isag
Se instalarán los siguientes paquetes NUEVOS:
  libllvm10 libpq5 postgresql-12 postgresql-client-12
```

Y modificamos el archivo de **pg_hba.conf** con los datos de nuestro servidor ldap:

```
# Database administrative login by Unix domain socket
local  all             postgres                                peer
host  all all 0.0.0.0/0 ldap ldapserver=server.local ldapbasedn="dc=server, dc=>
# TYPE  DATABASE  USER  ADDRESS  METHOD
```

Terminamos creando el **rol de login** para nuestro usuario 'miguelgulu':

```
server@server:~$ sudo -u postgres psql
psql (12.7 (Ubuntu 12.7-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# create role miguelgulu login;
CREATE ROLE
postgres=#
```

Si quisiéramos entrar con el usuario, nos saldría un error ya que no puede localizar el **listener** de postgresql. Esto se soluciona bastante fácil entrando en el archivo **/etc/postgresql/12/main/postgresql.conf**. Tendremos que descomentar esta línea y añadir **''** en el **listener** para elegir el que viene por defecto:

```
listen_addresses = '*'
# comma-separated list of addresses;
# defaults to 'localhost'; use '*' for
```

Y ahora sí, podremos entrar con nuestro nuevo usuario en postgresql!

Si queremos añadirle un sistema de guardado de archivos en el server podemos meterle un sftp (no es lo recomendable si lo llevamos a un caso real, lo suyo es que estuviera levantado en la nube). Para añadirlo es tan simple como tener instalado ssh con:

```
$ sudo apt install ssh
```

Añadimos las siguientes líneas en el archivo `/etc/ssh/sshd_config` al final:

```
Match group sftp
ChrootDirectory /home
X11Forwarding no
AllowTcpForwarding no
ForceCommand internal-sftp
```

Y creamos un grupo y un usuario al que llamaremos **sftp** y **sftpuser** respectivamente, además de su contraseña.

```
$ sudo addgroup sftp
$ sudo useradd -m sftpuser -g sftp
$ sudo passwd sftpuser
```

Y damos solamente **permisos** dentro del directorio al propio usuario:

```
$ sudo chmod 700 /home/sftpuser/
```

Y si queremos meternos mediante el explorador de archivos nautilus, basta con que en “Otras ubicaciones” pongamos en la conexión “**sftp://192.168.2.55/**”, nos pedirá el usuario y contraseña que hemos creado y... Listo!

5. CORTAFUEGOS

Para este caso no requeriremos de muchas reglas de **iptables** para darle cierta seguridad a nuestra red:

```
router@router:~$ sudo iptables -A FORWARD -p tcp --dport 80 -s 192.168.1.0/24 -d 192.168.2.55 -j ACCEPT
router@router:~$ sudo iptables -A FORWARD -p tcp --dport 389 -s 192.168.1.0/24 -d 192.168.2.55 -j ACCEPT
router@router:~$ sudo iptables -A FORWARD -p tcp --dport 5432 -s 192.168.1.0/24 -d 192.168.2.55 -j ACCEPT
router@router:~$ sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -i enp0s9 -j DNAT --to 192.168.2.55
```

Los tres primeros **iptables** sirven para darle permisos a los pcs de la red para que puedan acceder a los puertos del postgresql en el puerto **5432**, phpldapadmin con su openldap en el puerto **389** y a la página de apache2 que contendría el catálogo en el puerto **80** (donde también está el phpldapadmin).

Los otros dos **iptables** son reglas que permiten crear una **DNAT** para transformar las direcciones de red que vengan públicas en privadas y puedan comunicarse por protocolo **TCP** con el servidor sólo y exclusivamente en el **80**, para que puedan comprobar el catálogo de la tienda y hacer un pedido para que sea almacenado en la base de datos.