
Servidor de disco NFS

Andrés Millán Muñoz (amilmun@correo.ugr.es)

May 26, 2022

Contents

1 Creación de la máquina NFS	2
2 M1 y M2 como clientes	4
3 Seguridad del servidor NFS	5
Bibliografía	8

Para esta última práctica vamos a configurar un servidor NFS con el fin de proporcionar espacio adicional a las máquinas del backend; las cuales actuarán como clientes de NFS.

Las IPs de las máquinas son:

- **M1:** 192.168.49.128.
- **M2:** 192.168.49.129.
- **M3:** 192.168.49.130.
- **NFS:** 192.168.49.131.

1 Creación de la máquina NFS

Nuestro primer objetivo será crear una nueva máquina virtual que haga las veces de NFS. Para ello, utilizamos la misma ISO de Ubuntu Server de las anteriores máquinas.

El proceso de instalación es análogo, así que no lo detallaremos. Lo único que merece la pena destacar es que el usuario será `amilmun` y su contraseña `Swap1234`, al igual que en los casos anteriores. Además, editaremos el netplan para fijar en la segunda tarjeta la IP a `192.168.49.131/24` y dhcp a `false`.

```
amilmun@nfs-amilmun:~$ cat /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    ens33:
      dhcp4: true
    ens34:
      dhcp4: false
      addresses: [192.168.49.131/24]
  version: 2
```

Figure 1.1: Netplan de NFS

Tras esto, instalaremos los programas necesarios para tener en marcha NFS. Podemos hacerlo con `sudo apt-get install nfs-kernel-server nfs-common rpcbind`. Ahora creamos la carpeta compartida con el resto de máquinas. Para ello:

```
1 sudo mkdir /datos
2 sudo mkdir /datos/compartido
3 sudo chown nobody:nogroup /datos/compartido
4 sudo chmod -R 777 /datos/compartido
```

Para que M1 y M2 sean capaces de acceder, debemos añadir las siguientes líneas al fichero `/etc/exports`:

```
1 /datos/compartido/ 192.168.49.128(rw) 192.168.49.129(rw)
```

Reiniciando el servicio de NFS, podemos comprobar que está funcionando:

```
1 sudo systemctl restart nfs-kernel-server
```

```
● nfs-server.service - NFS server and services
   Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset
   Active: active (exited) since Wed 2022-05-25 16:21:46 UTC; 6s ago
   Process: 3363 ExecStopPost=/usr/sbin/exportfs -f (code=exited, status=0/SUCCESS
   Process: 3359 ExecStopPost=/usr/sbin/exportfs -au (code=exited, status=0/SUCCESS
   Process: 3358 ExecStop=/usr/sbin/rpc.nfsd 0 (code=exited, status=0/SUCCESS)
   Process: 3386 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/
   Process: 3385 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS
   Main PID: 3386 (code=exited, status=0/SUCCESS)

May 25 16:21:46 nfs-amilmun systemd[1]: Starting NFS server and services...
May 25 16:21:46 nfs-amilmun exportfs[3385]: exportfs: /etc/exports [1]: Neither '
May 25 16:21:46 nfs-amilmun exportfs[3385]: Assuming default behaviour ('no_sub
May 25 16:21:46 nfs-amilmun exportfs[3385]: NOTE: this default has changed sinc
May 25 16:21:46 nfs-amilmun exportfs[3385]: exportfs: /etc/exports [1]: Neither '
May 25 16:21:46 nfs-amilmun exportfs[3385]: Assuming default behaviour ('no_sub
May 25 16:21:46 nfs-amilmun exportfs[3385]: NOTE: this default has changed sinc
May 25 16:21:46 nfs-amilmun systemd[1]: Started NFS server and services.
```

Figure 1.2: Estado del servicio de NFS tras reiniciarlo.

2 M1 y M2 como clientes

Para que M1 y M2 puedan acceder a la carpeta, necesitamos proveerles de algunos paquetes necesarios. Los podemos conseguir con `sudo apt-get install nfs-common rpcbind`.

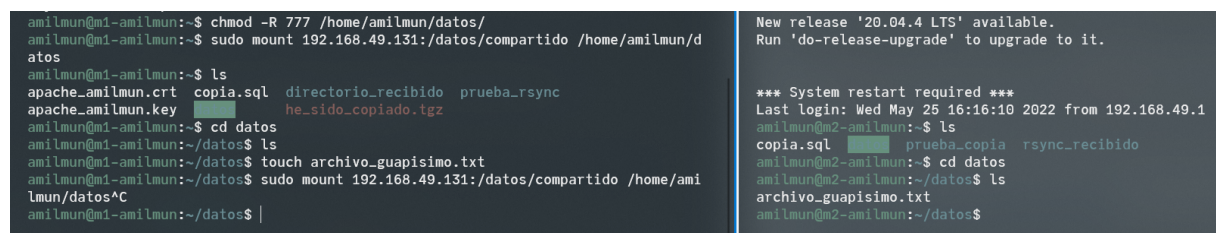
Creamos una carpeta donde montar los datos compartidos:

```
1 # M1 y M2
2 mkdir /home/amilmun/datos
3 chmod -R 777 /home/amilmun/datos
```

Montar el servidor NFS es tan sencillo como hacer

```
1 sudo mount 192.168.49.131:/datos/compartido /home/amilmun/datos
```

Si ponemos un archivo en la carpeta, veremos que se sincroniza en todas las máquinas:



The screenshot shows a terminal session on machine M1. The user runs `chmod -R 777 /home/amilmun/datos/`, then `sudo mount 192.168.49.131:/datos/compartido /home/amilmun/datos`. They then create a directory `datos` and list its contents, showing files like `apache_amilmun.crt`, `copia.sql`, `directorio_recibido`, `prueba_rsync`, and `he_sido_copiado.tgz`. They then create a file `archivo_guapisimo.txt` in the `datos` directory. On the right side of the screenshot, the terminal output from machine M2 is shown, indicating that it has received the file `archivo_guapisimo.txt` and that the directory `datos` now contains `copia.sql`, `prueba_copia`, and `rsync_recibido`.

Figure 2.1: Sincronización de un archivo por NFS.

En este momento existe un problema con la configuración: cada vez que reiniciemos las máquinas hará falta montar de nuevo la carpeta de `datos`. Podemos solucionarlo añadiendo la siguiente línea en el fichero `/etc/fstab`:

```
1 # M1 y M2
2 192.168.49.131:/datos/compartido /home/amilmun/datos/ nfs auto,noatime,
   noexec,bg,nfsvers=3,intr,tcp,actimeo=1800 0 0
```

3 Seguridad del servidor NFS

Para reforzar la seguridad del servidor, bloquearemos mediante iptables todo el tráfico a excepción del que esté relacionado con NFS.

Antes de crear el script, debemos cambiar la configuración de mountd y nlockmgr. Por defecto, utilizan puertos dinámicos. Para la configuración que sabemos hacer nosotros con iptables esto no nos beneficia. Aunque no es el método óptimo, podemos cambiarlo a puertos estáticos:

- Para **mountd**, añadimos al archivo `/etc/default/nfs-kernel-server` cambiamos la línea `RPCMOUNTDOPTS="--manage-gids"` por `RPCMOUNTDOPTS="--manage-gids -p 2000"`
- Para **nlockmgr**, crearemos un archivo llamado `nfs-ports.conf` en la carpeta `/etc/sysctl.d`, que contenga los parámetros `fs.nfs.nlm_tcpport = 2001` y `fs.nfs.nlm_udpport = 2002`. Una vez esté listo, reiniciamos el servicio con dicha configuración (`sudo sysctl --system /etc/init.d/nfs-kernel-server restart`).

Podemos comprobar que está funcionando con el comando

```
1 sudo rpcinfo -p localhost
```

En mi caso fue necesario reiniciar la máquina para la configuración de nlockmgr hiciera efecto.

```
amilmun@nfs-amilmun:~$ sudo rpcinfo -p
      program vers  proto  port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp   2000  mountd
    100005    1   tcp   2000  mountd
    100005    2   udp   2000  mountd
    100005    2   tcp   2000  mountd
    100005    3   udp   2000  mountd
    100005    3   tcp   2000  mountd
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    3   tcp   2049
    100003    3   udp   2049  nfs
    100227    3   udp   2049
    100021    1   udp   2002  nlockmgr
    100021    3   udp   2002  nlockmgr
    100021    4   udp   2002  nlockmgr
    100021    1   tcp   2001  nlockmgr
    100021    3   tcp   2001  nlockmgr
    100021    4   tcp   2001  nlockmgr
```

Figure 3.1: Configuración de los puertos.

Ahora estamos en condiciones de diseñar las reglas. Creamos un script similar a los de las anteriores prácticas, especificando las IPs de las máquinas M1 y M2 ([Vivek Gite 2017](#)). En mi caso, añadiré SSH puesto que lo estoy utilizando para realizar esta práctica más fácilmente:

```
1  #!/bin/bash
2
3  # Eliminar config anterior
4  iptables -F
5  iptables -X
6
7  # Denegar todo el tráfico
8  iptables -P INPUT DROP
9  iptables -P FORWARD DROP
10 iptables -P OUTPUT DROP
11
12 # Permitir SSH
13 iptables -A INPUT -p tcp --dport 22 -j ACCEPT
14 iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
```



```
15
16 # NFS
17 iptables -A INPUT -p tcp --dport 2049 -s 192.168.49.128,192.168.49.129
   -j ACCEPT
18 iptables -A OUTPUT -p tcp --sport 2049 -d 192.168.49.128,192.168.49.129
   -j ACCEPT
19
20 # Portmapper
21 iptables -A INPUT -p tcp --dport 111 -s 192.168.49.128,192.168.49.129 -
   j ACCEPT
22 iptables -A OUTPUT -p tcp --sport 111 -d 192.168.49.128,192.168.49.129
   -j ACCEPT
23
24 # Mountd, nlockmgr
25 iptables -A INPUT -p tcp --dport 2000 -s 192.168.49.128,192.168.49.129
   -j ACCEPT
26 iptables -A INPUT -p tcp --dport 2001 -s 192.168.49.128,192.168.49.129
   -j ACCEPT
27 iptables -A INPUT -p udp --dport 2002 -s 192.168.49.128,192.168.49.129
   -j ACCEPT
28 iptables -A OUTPUT -p tcp --sport 2000 -d 192.168.49.128,192.168.49.129
   -j ACCEPT
29 iptables -A OUTPUT -p tcp --sport 2001 -d 192.168.49.128,192.168.49.129
   -j ACCEPT
30 iptables -A OUTPUT -p udp --sport 2002 -d 192.168.49.128,192.168.49.129
   -j ACCEPT
```

Podemos ver que funciona perfectamente:

```
amilmun@m1-amilmun:~/datos$ ls
archivo_guapisimo.txt
tras_iptables.txt
amilmun@m1-amilmun:~/datos$ |
```

```
amilmun@m2-amilmun:~/datos$ ls
archivo_guapisimo.txt
amilmun@m2-amilmun:~/datos$ touch tra
s_iptables.txt
amilmun@m2-amilmun:~/datos$
```

Figure 3.2: Tras aplicar las reglas de iptables el servidor de NFS sigue funcionando

Bibliografía

Vivek Gite. 2017. “How to Use or Specify Multiple IP Addresses in Iptables Source or Destination on Linux.” June 29, 2017. <https://www.cyberciti.biz/faq/how-to-use-iptables-with-multiple-source-destination-ips-addresses/>.