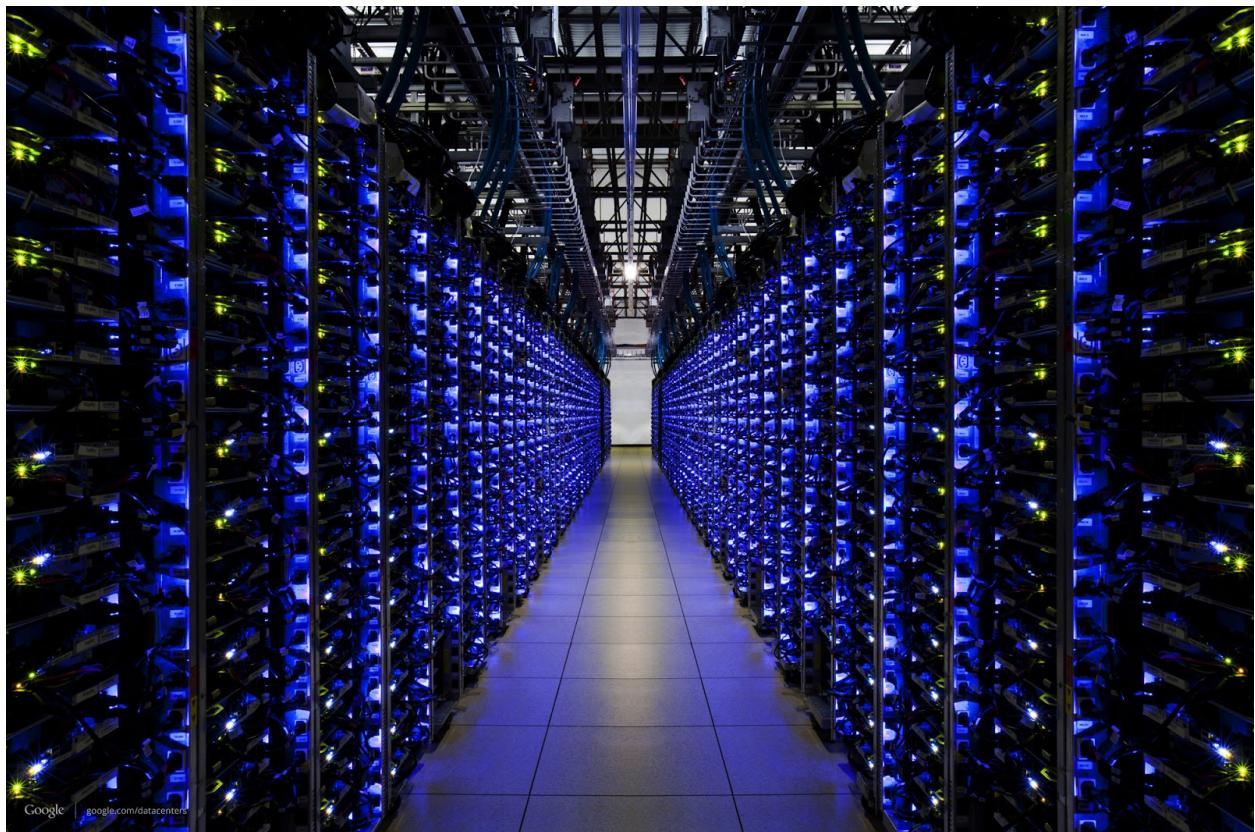

Servidores Web de Altas Prestaciones

3º Grado en Ingeniería Informática (Tecnologías de la información)
Adrián Acosa Sánchez

Práctica 6: Servidor de disco NFS

23 de mayo del 2022



Índice

Índice	2
Crear una máquina virtual y configurarla como servidor NFS	3
Configurar los clientes M1 y M2	4
Seguridad en el servidor NFS	7
Demostración de funcionamiento	11
Bibliografía	12

Crear una máquina virtual y configurarla como servidor NFS

Para comenzar con esta práctica crearemos una nueva máquina virtual que será nuestro servidor NFS. La instalación será la misma que para las otras máquinas virtuales ya vistas anteriormente. El nombre que he asignado a la máquina ha sido *nfs-adrianacosa* y el usuario el mismo que en las demás máquinas (mi usuario UGR, que es *adrianacosa*). La IP asignada a esta máquina por DHCP ha sido 192.168.122.95.

Lo primero que tenemos que hacer para convertir esta máquina en un servidor NFS es instalar las herramientas necesarias para ello, que son las siguientes:

```
adrianacosa@nfs-adrianacosa:~$ sudo apt-get install nfs-kernel-server nfs-common rpcbind
[sudo] password for adrianacosa:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  keyutils libevent-core-2.1-7 libnfsidmap1
Suggested packages:
  watchdog
The following NEW packages will be installed:
  keyutils libevent-core-2.1-7 libnfsidmap1 nfs-common nfs-kernel-server rpcbind
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 616 kB of archives.
After this operation, 2235 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Lo siguiente que haremos será configurar la carpeta que compartiremos entre los servidores M1 y M2. Vamos a crear la carpeta siguiendo las recomendaciones del guión, es decir, crearemos la carpeta en */datos/compartido* y le quitaremos cualquier tipo de pertenencia a la carpeta mediante el uso del comando *chown* y le daremos todos los permisos posibles de la siguiente manera:

```
adrianacosa@nfs-adrianacosa:~$ sudo mkdir /datos
adrianacosa@nfs-adrianacosa:~$ sudo mkdir /datos/compartido
```

```
adrianacosa@nfs-adrianacosa:~$ sudo chown nobody:nogroup /datos/compartido
adrianacosa@nfs-adrianacosa:~$ sudo chmod -R 777 /datos/compartido/
```

Una vez hemos creado la carpeta y hemos editado sus permisos correctamente, procedemos a darle permiso de acceso a ambas máquinas (M1 y M2). Para ello debemos añadir sus IP en el archivo de configuración */etc/exports*:

```

# /etc/exports: the access control list for filesystems which may be exported
#           to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4      gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/datos/compartido/ 192.168.122.43(rw) 192.168.122.36(rw)

```

Tras esto debemos reiniciar el servicio y comprobamos que todo ha salido correctamente:

```

adrianacosa@nfs-adrianacosa: $ sudo service nfs-kernel-server restart
adrianacosa@nfs-adrianacosa: $ sudo service nfs-kernel-server status
● nfs-server.service - NFS server and services
  Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset: enabled)
  Active: active (exited) since Mon 2022-05-23 21:06:18 UTC; 9s ago
    Process: 2341 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
    Process: 2342 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
      Main PID: 2342 (code=exited, status=0/SUCCESS)
        CPU: 5ms

May 23 21:06:18 nfs-adrianacosa systemd[1]: Starting NFS server and services...
May 23 21:06:18 nfs-adrianacosa exportfs[2341]: exports: /etc/exports [2]: Neither 'subtree_check' or 'no_subtree_check' specified for export "192.168.122.43:/datos/compartido/".
May 23 21:06:18 nfs-adrianacosa exportfs[2341]: Assuming default behaviour ('no_subtree_check').
May 23 21:06:18 nfs-adrianacosa exportfs[2341]: NOTE: this default has changed since nfs-utils version 1.0.x
May 23 21:06:18 nfs-adrianacosa exportfs[2341]: exports: /etc/exports [2]: Neither 'subtree_check' or 'no_subtree_check' specified for export "192.168.122.36:/datos/compartido/".
May 23 21:06:18 nfs-adrianacosa exportfs[2341]: Assuming default behaviour ('no_subtree_check').
May 23 21:06:18 nfs-adrianacosa exportfs[2341]: NOTE: this default has changed since nfs-utils version 1.0.x
May 23 21:06:18 nfs-adrianacosa systemd[1]: Finished NFS server and services.

```

Y con esto terminamos la configuración de la carpeta compartida en el servidor NFS.

Configurar los clientes M1 y M2

En las máquinas clientes, en este caso M1 y M2, debemos instalar los paquetes necesarios para poder recibir la información del servidor NFS y crear el punto de montaje para los datos que recibiremos o enviaremos a éste. Para ello ejecutamos los siguientes comandos en ambos servidores:

```

adrianacosa@m1-adrianacosa:~$ sudo apt-get install nfs-common rpcbind
[sudo] password for adrianacosa:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  keyutils libnfsidmap2 libtirpc-common libtirpc3
Suggested packages:
  watchdog
The following NEW packages will be installed:
  keyutils libnfsidmap2 libtirpc-common libtirpc3 nfs-common rpcbind
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 404 kB of archives.
After this operation, 1,517 kB of additional disk space will be used.
Do you want to continue? [Y/n]

```

```
adrianacosa@m1-adrianacosa:~$ mkdir datos
adrianacosa@m1-adrianacosa:~$ chmod -R 777 datos
adrianacosa@m1-adrianacosa:~$ ls -l | grep datos
drwxrwxrwx 2 adrianacosa adrianacosa 4096 May 23 21:15 datos
```

Ahora que ya tenemos un punto de montaje con todos los permisos disponibles, podemos montar la carpeta remota del servidor NFS en el directorio creado específicamente para ello:

```
adrianacosa@m1-adrianacosa:~$ sudo mount 192.168.122.95:/datos/compartido datos
```

Y tras esto, vamos a probar que se pueden leer y escribir archivos dentro de la carpeta datos:

```
adrianacosa@m1-adrianacosa:~$ ls -la datos/
total 8
drwxrwxrwx 2 nobody      nogroup      4096 May 23 21:21 .
drwxr-xr-x 6 adrianacosa adrianacosa 4096 May 23 21:15 ..
adrianacosa@m1-adrianacosa:~$ touch datos/archivo.txt
adrianacosa@m1-adrianacosa:~$ ls -la datos/
total 8
drwxrwxrwx 2 nobody      nogroup      4096 May 23 21:22 .
drwxr-xr-x 6 adrianacosa adrianacosa 4096 May 23 21:15 ..
-rw-rw-r-- 1 adrianacosa adrianacosa     0 May 23 21:22 archivo.txt
adrianacosa@m1-adrianacosa:~$
```

Y podemos comprobar que existe una sincronización instantánea de la carpeta en ambas máquinas haciendo un `ls` en la otra máquina:

```
adrianacosa@m2-adrianacosa:~$ ls -la datos/
total 8
drwxrwxrwx 2 nobody      nogroup      4096 May 23 21:22 .
drwxr-xr-x 6 adrianacosa adrianacosa 4096 May 23 21:16 ..
-rw-rw-r-- 1 adrianacosa adrianacosa     0 May 23 21:22 archivo.txt
adrianacosa@m2-adrianacosa:~$
```

Hay que tener en cuenta que esta configuración al reiniciar habría que volver a hacerla cada vez que queramos acceder a la carpeta. Para no tener que realizar ésta configuración cada vez que reiniciamos la máquina, podemos hacerla permanente modificando el archivo de inicialización de montajes de sistemas de archivo, que es `/etc/fstab`, añadiendo la siguiente línea:

```

# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/ubuntu-vg/ubuntu-lv during curtin installation
/dev/disk/by-id/dm-uuid-LVM-TwoZFSaZIJvTS831sQ4hQqmKcpWYTtqeIYfRjAWLDwkarzvrGLMkEoY8X0ustMb8 / ext4 defaults 0 1
# /boot was on /dev/vda2 during curtin installation
/dev/disk/by-uuid/925b9cc7-e068-46e1-803b-64bd07a62542 /boot ext4 defaults 0 1
/swap.img none swap sw 0 0
192.168.122.95:/datos/compartido /home/adrianacosa/datos/ nfs auto,noatime,nolock,bg,nfsvers=3,intr,tcp,actimeo=1800 0 0

```

En esta línea indicamos, en este orden:

- Qué queremos montar. En nuestro caso, la carpeta del servidor NFS.
- Dónde lo queremos montar. Nosotros queremos montarlo en la carpeta datos de nuestro home.
- Tipo de sistema de archivo. El tipo de sistema de archivos que tiene la carpeta que montamos en *datos* es nfs.
- Opciones de montaje:
 - *auto*: el sistema de archivo se montará automáticamente cuando iniciemos el sistema o cuando el comando ‘mount -a’ se ejecute.
 - *noatime*: no actualiza los tiempos de acceso de los inodos en el sistema de archivos, cosa que puede aumentar el rendimiento de éste.
 - *nolock*: indica que las aplicaciones pueden bloquear archivos, pero al bloquear éstos sólo proporcionan el bloqueo de cara al cliente que bloquea el archivo. El resto de clientes no se ven afectados por dicho bloqueo.
 - *bg*: permite que si hay algún fallo en la conexión con el sistema de archivos remoto se cree un fork que continúa intentando montar éste. El padre devuelve un *exit code* con valor 0.
 - *nfsvers=3*: indica el número de servidores que se encuentran conectados mediante NFS.
 - *intr*: indica que si una señal interrumpe una operación NFS en curso, las llamadas al sistema devuelven una señal EINTR.
 - *tcp*: indica el protocolo que usará el transporte de los datos de un servidor a otro.
 - *actimeo=1800*: establece los valores de *acregmin* (mínimo tiempo que guarda en caché los atributos de archivo regular el servidor NFS) , *acermax* (máximo tiempo que guarda en caché los atributos de un archivo regular el servidor NFS), *acdirmmin* (lo mismo que *acregmin* pero con directorios) y *acdirmmax* al mismo valor indicado (lo mismo que *acermax* pero con directorios).
- Flag de *dump*. Establece cuándo hacer un *backup*. Cuando se establece a 0, *dump* ignora el sistema de archivos, y si está a 1 hace un *backup*.
- Flag de *pass*. Establece si hay que registrar el sistema de archivo en busca de errores con la utilidad *fsck*. En este caso, le hemos indicado que no.

Una vez reiniciemos el sistema, tendremos la carpeta montada en el mismo sitio y disponible.

Seguridad en el servidor NFS

Para comenzar con la seguridad en nuestro servidor NFS, partimos de la configuración de denegación implícita en IPTABLES para el tráfico entrante:

```
#!/bin/bash
#
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Una vez hecho esto, abrimos los puertos que se encuentran asociados a los distintos servicios NFS. Para ello tenemos que establecer unos puertos a *mountd* y *nlockmgr* ya que utilizan puertos dinámicos por defecto. Para poder configurar un puerto estático para estos servicios, vamos a configurar el archivo */etc/default/nfs-kernel-server* añadiendo *-p 2000* (tanto tcp como udp):

```
RPCMOUNTDOPTS="--manage-gids -p 2000 --no-nfs-version 4"
```

He añadido la opción de *--no-nfs-version 4* debido a que me ha dado fallos con una versión más reciente de ubuntu server y no me asignaba bien el puerto. Investigando he intuido que puede ser porque la versión más reciente de nfs funciona de una manera distinta a la explicada en el guión, entonces he decidido crear de nuevo la máquina, pero esta vez en *debian* para evitar usar la versión más reciente de nfs, y he preferido añadir esta opción por si acaso.

Tras esto configuramos el puerto para el servicio *nlockmgr*. Esta tarea es un poco más complicada porque es parte de un módulo del kernel. Pero podemos reconfigurar el módulo a través de *systemd* sin tener que reiniciar el sistema completo. Para ello vamos a crear un archivo nuevo de configuración que se llamará *swap-nfs-ports.conf* en la carpeta */etc/sysctl.d/* y a la que añadiremos la siguiente información:

```
fs.nfs.nlm_tcpport = 2001
fs.nfs.nlm_udpport = 2002
~
~
~
~
~
~
```

/etc/sysctl.d/swap-nfs-ports.conf |

Donde le indicamos que use el puerto 2001 para comunicaciones tcp y el 2002 para comunicaciones udp.

Tras esto vamos a ejecutar el nuevo archivo de configuración que hemos creado con la orden siguiente, donde se puede ver al final que se aplican las configuraciones que acabamos de escribir.

```
adrianacosa@nfs-adrianacosa:~$ sudo sysctl --system /etc/init.d/nfs-kernel-server restart
[sudo] password for adrianacosa:
* Applying /usr/lib/sysctl.d/30-tracker.conf ...
fs.inotify.max_user_watches = 65536
* Applying /usr/lib/sysctl.d/50-bubblewrap.conf ...
kernel.unprivileged_userns_clone = 1
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
kernel.pid_max = 4194304
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /usr/lib/sysctl.d/protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/swap-nfs-ports.conf ...
fs.nfs.nlm_tcpport = 2001
fs.nfs.nlm_udpport = 2002
* Applying /etc/sysctl.conf ...
adrianacosa@nfs-adrianacosa:~$
```

Por último, para comprobar que las configuraciones de los puertos han salido correctamente, usamos la herramienta *rpcinfo -p server*:

```

adrianacosa@nfs-adrianacosa:~$ sudo rpcinfo -p localhost
    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
adrianacosa@nfs-adrianacosa:~$ 

```

Y como podemos ver, se han asignado todos los puertos correctamente. Lo último que nos falta es crearnos un script donde sólo se permita el tráfico NFS en esta máquina, y específicamente de las máquinas M1 y M2. Para ello, he creado el siguiente script:

```

#!/bin/bash

iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
#iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -s 192.168.122.43,192.168.122.36 -m state --state NEW,ESTABLISHED -m tcp -p tcp -m multiport --dports 2049,111,2000,2001 -j ACCEPT
iptables -A INPUT -s 192.168.122.43,192.168.122.43 -m state --state NEW,ESTABLISHED,RELATED -m udp -p udp -m multiport --dports 2049,111,2000,2002 -j ACCEPT

# (6) Permitir comunicación SSH con el anfitrión
iptables -A INPUT -p tcp -s 192.168.122.1 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

```

Donde estoy indicando que acepte todo el tráfico saliente sólamente, además que indicar que para las comunicaciones *tcp* acepte sólo comunicaciones entrantes en los puertos 2049(nfs), 111(portmapper), 2000(mountd) y 2001(nlockmgr), y que también acepte las comunicaciones *udp* en los mismos puertos excepto el 2001 (que es sólo *tcp*) que lo he cambiado al 2002 (sólo *udp*). Además he indicado que éstas comunicaciones sólo se pueden realizar si la fuente de las peticiones entrantes están en las máquinas M1 ó M2. La última regla es para poder usar SSH desde la máquina anfitrión.

Éste sería el script necesario en la parte del servidor NFS. Ahora vamos a pasar a configurar cada uno de los scripts en la parte de los clientes para que a parte de todo lo permitido en las

anteriores prácticas, podamos también permitir la comunicación saliente y entrante del protocolo NFS. Por lo tanto, para llevar ésta tarea a cabo, he elaborado el siguiente script:

```
#!/bin/bash
# (1) Eliminamos todas las reglas que hubiera para
# realizar una configuracion limpia
iptables -F
iptables -X

# (2) establecemos las politicas por defecto, es decir, la denegacion
# de cualquier tipo de trafico
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# (3) Permitir cualquier acceso desde localhost (interface lo):
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# (4) Permitir solo los paquetes de entrada y salida solamente
# si provienen de y salen hacia la maquina 3
iptables -A INPUT -s 192.168.122.68 -p tcp --dports 80,443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -d 192.168.122.68 -p tcp -m multiport --sports 80,443 -m state --state ESTABLISHED -j ACCEPT

# (5) Permitir comunicacion entre las bases de datos de ambas maquinas
iptables -A INPUT -p tcp -s 192.168.122.36 --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT

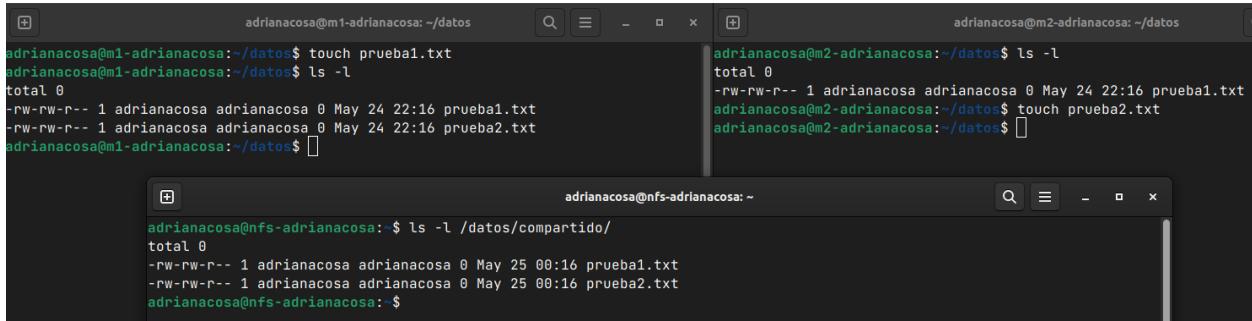
# (6) Permitir comunicacion SSH con el anfitrion
iptables -A INPUT -p tcp -s 192.168.122.1 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT

# (7) Permitir comunicacion NFS con el servidor NFS
iptables -A INPUT -s 192.168.122.67 -p tcp -m multiport --sports 111,2049,2000,2001 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -d 192.168.122.67 -p tcp -m multiport --dports 111,2049,2000,2001 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A INPUT -s 192.168.122.67 -p udp -m multiport --sports 111,2049,2000,2002 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -d 192.168.122.67 -p udp -m multiport --dports 111,2049,2000,2002 -m state --state NEW,ESTABLISHED -j ACCEPT
```

Así es como ha quedado el script del cortafuegos finalmente en ambas máquinas. Las 4 últimas líneas son las nuevas en ésta práctica, donde indico que acepte las comunicaciones entrantes que tengan origen en la máquina del servidor NFS, ya sea udp o tcp, con puertos de origen 111, 2049, 2000 y 2001 para tcp o 2002 para udp, y que para las comunicaciones salientes de igual manera, con la única diferencia de que los puertos son de destino y no de origen.

Demostración de funcionamiento



The image shows three terminal windows demonstrating file replication across three servers (m1, m2, and nfs).
Terminal 1 (m1-adrianacosa):
adrianacosa@m1-adrianacosa:~/datos\$ touch prueba1.txt
adrianacosa@m1-adrianacosa:~/datos\$ ls -l
total 0
-rw-rw-r-- 1 adrianacosa adrianacosa 0 May 24 22:16 prueba1.txt
-rw-rw-r-- 1 adrianacosa adrianacosa 0 May 24 22:16 prueba2.txt
adrianacosa@m1-adrianacosa:~/datos\$
Terminal 2 (m2-adrianacosa):
adrianacosa@m2-adrianacosa:~/datos\$ ls -l
total 0
-rw-rw-r-- 1 adrianacosa adrianacosa 0 May 24 22:16 prueba1.txt
adrianacosa@m2-adrianacosa:~/datos\$ touch prueba2.txt
adrianacosa@m2-adrianacosa:~/datos\$
Terminal 3 (nfs-adrianacosa):
adrianacosa@nfs-adrianacosa: \$ ls -l /datos/compartido/
total 0
-rw-rw-r-- 1 adrianacosa adrianacosa 0 May 25 00:16 prueba1.txt
-rw-rw-r-- 1 adrianacosa adrianacosa 0 May 25 00:16 prueba2.txt
adrianacosa@nfs-adrianacosa: \$

Como se puede apreciar (sin ejecutar nada anteriormente), cuando creamos cualquier archivo en la carpeta desde cualquier servidor, se replica inmediatamente el contenido de esta en el resto de servidores.

Indicar que las horas están mal debido a las distintas configuraciones horarias que tienen las máquinas, todo ha sido de forma simultánea.

Bibliografía

- 1.- <https://wiki.debian.org/es/fstab>
- 2.- <https://linux.die.net/man/5/nfs>
- 3.-<https://stackoverflow.com/questions/26187345/iptables-rules-for-nfs-server-and-nfs-client>