Servicios de almacenamiento

NAS con NFS · SAN con iSCSI

📧 José Domingo Muñoz

🏫 IES Gonzalo Nazareno · Dos Hermanas

📚 SRI · Servicios de Red e Internet

01

Introducción al almacenamiento

DAS, NAS, SAN y Cloud

¿Por qué necesitamos servicios de almacenamiento?

Los servidores y aplicaciones generan y consumen volúmenes de datos cada vez mayores. El **almacenamiento local** se queda corto en cuanto necesitamos:

- **Compartir datos** entre varios equipos o servicios
- **Centralizar copias de seguridad** y políticas de retención
- **Ampliar capacidad** sin tocar el hardware del servidor
- **Tolerar fallos** y caídas de un nodo individual
- **Separar cómputo y datos** para escalar de forma independiente

> ℹ️ De ahí surgen distintas arquitecturas: cada una resuelve un problema diferente.

Las cuatro grandes categorías

DAS — *DIRECT ATTACHED STORAGE*

Disco conectado **directamente** al servidor (SATA, SAS, USB).

NAS — *NETWORK ATTACHED STORAGE*

Almacenamiento compartido **por red** a nivel de **sistema de ficheros**.

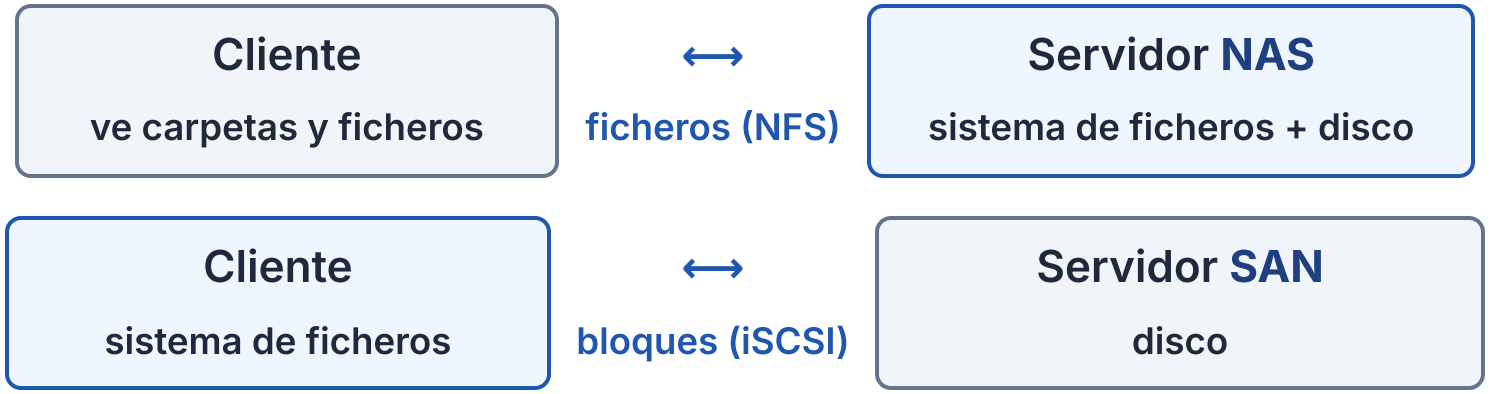
SAN — *STORAGE AREA NETWORK*

Red **dedicada** que ofrece a los servidores **dispositivos de bloques**.

CLOUD — *OBJECT STORAGE*

Almacenamiento en la **nube** orientado a objetos (S3, Swift...).

¿Ficheros o bloques?



- En una **NAS**, el sistema de ficheros está en el **servidor**: el cliente pide ficheros
- En una **SAN**, el servidor solo ofrece un disco: el **cliente lo formatea** y lo gestiona

DAS — Direct Attached Storage

> *El disco está **físicamente conectado** al servidor que lo usa.*

- Conexión **directa**: SATA, SAS, NVMe, USB...
- **Máximo rendimiento** y la menor latencia posible
- **Económico** y simple de instalar
- Sólo accesible desde **un único servidor**
- No es trivial **compartir** los datos con otros equipos
- Si el servidor cae, los datos quedan **inaccesibles**

ℹ️ Es lo que tiene un PC o un servidor "estándar": discos internos formateados con un sistema de ficheros local.

NAS — Network Attached Storage

> *Un servidor comparte **sistemas de ficheros completos** por la red.*

- Trabaja a nivel de **archivo**: el cliente ve carpetas y ficheros
- Habitualmente sobre **TCP/IP**, en redes de uso general
- Protocolos típicos: **NFS** (Unix/Linux), **SMB/CIFS** (Windows)
- Sencillo de **administrar** y de compartir entre varios clientes
- Ideal para **copias de seguridad**, perfiles de usuario, contenidos web

ℹ️ El cliente **no** formatea el almacenamiento: lo monta como una carpeta y trabaja con ficheros.

SAN — Storage Area Network

> *Una **red dedicada** ofrece a los servidores **dispositivos de bloques**.*

- Trabaja a nivel de **bloque**: el cliente ve un disco "como si fuera local"
- Red **dedicada**, normalmente de **alta velocidad** (10 Gbps, fibra...)
- El cliente **formatea** el disco con su sistema de ficheros preferido
- Protocolos típicos: **iSCSI** (sobre TCP/IP) y **Fibre Channel**
- Habitual en virtualización, bases de datos y entornos exigentes

⚠️ Un dispositivo SAN **no** debe montarse simultáneamente en dos clientes con un sistema de ficheros tradicional: corrompería los datos.

Cloud Storage

> *Almacenamiento ofrecido como servicio en la nube, generalmente **orientado a objetos**.*

- **API HTTP** (REST) para subir, descargar y gestionar objetos
- Estándar de facto: **S3** de Amazon (y compatibles: MinIO, Ceph RGW...)
- **Escalado** prácticamente ilimitado
- Modelo de pago **por uso**
- Pensado para datos **inmutables** o de poca modificación: imágenes, vídeos, *backups*

ℹ️ No reemplaza a NAS o SAN: cubre **otro caso de uso**, donde la latencia importa menos que la escala y la durabilidad.

Comparativa rápida

	DAS	NAS	SAN	CLOUD
Acceso	Local	Red TCP/IP	Red dedicada	Internet (HTTP)
Granularidad	Bloque	Archivo	Bloque	Objeto
Compartir entre clientes	❌	✅	⚠️ con cuidado	✅
Rendimiento	Muy alto	Medio	Alto	Variable
Coste	Bajo	Medio	Alto	Pago por uso
Caso típico	Servidor único	Compartir ficheros	Virtualización, BBDD	Backups, *media*

Escenario de los ejemplos

MÁQUINA	PAPEL	IP (RED DE DATOS)
almacenamiento	Servidor NAS (NFS) y SAN (iSCSI), con RAID 5 y LVM	`192.168.200.10`
servidorweb	Cliente iSCSI	`192.168.200.20`
backend1 · **backend2**	Clientes NFS	`192.168.200.21` · `.22`

- El servidor de almacenamiento tiene **tres discos** adicionales: `/dev/vdb`, `/dev/vdc` y `/dev/vdd`
- Lo que se comparte son **volúmenes lógicos** creados sobre el RAID 5

> ℹ️ Es el escenario de la práctica: en el tuyo, cambia las IP por las de tu red.

02

Preparar el almacenamiento

RAID 5 con mdadm y LVM

RAID 5 con `mdadm`

RAID 5 reparte los datos y la **paridad** entre todos los discos: si falla **un** disco, no se pierde nada.

```
sudo apt install mdadm
sudo mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/vdb /dev/vdc /dev/vdd

cat /proc/mdstat              # estado y progreso de la sincronización
sudo mdadm --detail /dev/md0
lsblk /dev/md0
```

- Tamaño útil: **(n – 1) × tamaño del disco**: la paridad, repartida entre todos, ocupa el equivalente a un disco
- Necesita al menos **3 discos**

Que el RAID sobreviva al reinicio

Hay que guardar la definición del RAID y actualizar el *initramfs*:

```
sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf
sudo update-initramfs -u
```

Ejecútalo **una sola vez**: `tee -a` añade al final y, si lo repites, la línea `ARRAY` queda duplicada.

> ⚠️ Sin estos pasos, tras reiniciar el RAID puede aparecer como `/dev/md127` y fallar lo que dependa de `/dev/md0`.

Comprobar tras reiniciar: `cat /proc/mdstat`.

LVM sobre el RAID

LVM divide el RAID en **volúmenes lógicos** que se pueden crear, ampliar o borrar sin reparticionar.

```
sudo apt install lvm2
sudo pvcreate /dev/md0                      # volumen físico
sudo vgcreate vgalmacen /dev/md0            # grupo de volúmenes
sudo lvcreate -L 512M -n lun1 vgalmacen     # volúmenes lógicos
sudo lvcreate -L 512M -n lun2 vgalmacen
sudo lvcreate -L 1G   -n nfs  vgalmacen
sudo lvs
```

- Cada volumen es un dispositivo de bloques: `/dev/vgalmacen/lun1`, `/dev/vgalmacen/nfs`…
- Los de la SAN **no se formatean** en el servidor (lo hace el cliente); el de la NAS **sí**

SRI · UNIDAD 2 — SERVICIOS DE ALMACENAMIENTO

03

NAS con NFS

Compartir directorios entre máquinas Linux

¿Qué es NFS?

> *"**Network File System** es un protocolo para **compartir ficheros y directorios** por red. Originalmente desarrollado por **Sun Microsystems**, es el estándar de NAS en entornos Unix y Linux.*

- El servidor **exporta** uno o varios directorios
- El cliente los **monta** como si fueran locales
- Las operaciones de lectura y escritura se realizan **remotamente** sobre TCP/IP
- **NFSv4** (la versión que se monta por defecto) usa solo el puerto **2049/tcp**

ℹ️ Es transparente para las aplicaciones: leen y escriben en una ruta como cualquier otra.

Instalación del servidor NFS

```
sudo apt install nfs-kernel-server
systemctl status nfs-server
```

- El servicio queda activo, pero aún **no exporta nada**: hay que configurar `/etc/exports`
- En Debian 13, NFS sobre **UDP** está desactivado por defecto

ℹ️ **NFSv3** necesita además **RPC** (`rpcbind`, `mountd`): lo usa, por ejemplo, `showmount`. NFSv4 solo necesita el puerto 2049.

Montaje permanente en el servidor

El volumen que se va a compartir se **formatea** y se monta en el servidor de forma **permanente**:

```
sudo mkfs.ext4 /dev/vgalmacen/nfs
sudo mkdir -p /srv/nfs
```

Línea en `/etc/fstab` (o una unidad `.mount`, que veremos más adelante):

```
/dev/vgalmacen/nfs   /srv/nfs   ext4   defaults   0   2
```

```
sudo systemctl daemon-reload
sudo mount -a
```

Configuración: `/etc/exports`

Cada línea declara un **directorio exportado** y la lista de clientes con sus opciones:

```
/srv/nfs       192.168.200.0/24(rw,sync,no_subtree_check)
/srv/publico   192.168.200.20(ro,sync,no_subtree_check)
```

- El primero, en **lectura y escritura** para toda la red de datos
- El segundo, **solo lectura** y para un único cliente

APLICAR LOS CAMBIOS

```
sudo exportfs -ra        # recarga la configuración
sudo exportfs -v         # muestra lo exportado actualmente
```

Opciones más usadas en exports

OPCIÓN	PARA QUÉ SIRVE
`rw` / `ro`	Lectura y escritura / sólo lectura (por defecto)
`sync`	Confirma la escritura en disco antes de responder (por defecto)
`async`	Más rápido, pero menos seguro ante caídas
`no_subtree_check`	Desactiva la verificación de subdirectorio (por defecto)
`root_squash`	El `root` del cliente se mapea a `nobody` (por defecto)
`no_root_squash`	Permite que el `root` remoto sea `root` real (peligroso)
`all_squash`	Todos los usuarios remotos se mapean a `nobody`

Usuarios y permisos en NFS

- El cliente envía el **UID y el GID** (números, no nombres) y el servidor comprueba los permisos con ellos
- Con `root_squash`, el `root` del cliente es `nobody` en el servidor: si el directorio es de `root`, **no puede escribir** (*Permission denied*)

SOLUCIONES

- Dar permisos en el **servidor**: `chown` del directorio al usuario que va a escribir
- `no_root_squash`, solo si se confía plenamente en el cliente

Cliente NFS: montaje manual

```
sudo apt install nfs-common

# Lo que exporta el servidor (usa NFSv3)
showmount -e 192.168.200.10

sudo mkdir -p /var/www/nfs
sudo mount 192.168.200.10:/srv/nfs /var/www/nfs
```

A partir de ahí, `/var/www/nfs` se usa como cualquier directorio local.

Línea equivalente en `/etc/fstab`:

```
192.168.200.10:/srv/nfs   /var/www/nfs   nfs   defaults,_netdev   0   0
```

Unidades de montaje de systemd

Un `.mount` describe un montaje como una unidad más de systemd (`/etc/systemd/system/`).

El nombre del fichero **tiene que coincidir** con la ruta de montaje, y se activa como cualquier servicio:

```
$ systemd-escape -p --suffix=mount /var/www/nfs
var-www-nfs.mount
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now var-www-nfs.mount
```

ℹ️ systemd ya convierte cada línea de `/etc/fstab` en una unidad `.mount`: las dos formas son equivalentes.

Unidad de montaje para NFS

`/etc/systemd/system/var-www-nfs.mount` en **backend1** y **backend2**:

```
[Unit]
Wants=network-online.target
After=network-online.target

[Mount]
What=192.168.200.10:/srv/nfs
Where=/var/www/nfs
Type=nfs
Options=_netdev

[Install]
WantedBy=remote-fs.target
```

`After=network-online.target` y **`_netdev`**: esperar a que haya red antes de montar.

Comprobaciones útiles

```
# En el cliente: montajes NFS activos y versión usada
findmnt -t nfs,nfs4
nfsstat -m

# En el servidor: lo que se exporta y a quién
sudo exportfs -v

# Errores
journalctl -u nfs-server
journalctl -u var-www-nfs.mount
```

04

SAN con iSCSI

Compartir dispositivos de bloques sobre TCP/IP

¿Qué es iSCSI?

> *"**Internet Small Computer Systems Interface** transporta **comandos SCSI** sobre **TCP/IP**, permitiendo acceder remotamente a discos como si fueran **dispositivos de bloques locales**.*

- Implementa **SAN** sobre redes Ethernet **estándar**, sin hardware específico
- Alternativa **económica** a Fibre Channel
- Habitual en redes de **1 Gbps** y **10 Gbps**
- El cliente ve un **disco nuevo** que formatea y monta a su gusto
- Puerto **3260/tcp** entre *initiator* y *target*

Elementos de iSCSI

LADO SERVIDOR

- **Target** — recurso publicado por el servidor; agrupa una o varias LUN
- **LUN** (*Logical Unit Number*) — cada disco que ofrece el target (un disco, una partición, un volumen lógico...)
- **Portal** — IP y puerto donde escucha (`192.168.200.10:3260`)

LADO CLIENTE

- **Initiator** — el cliente iSCSI; descubre targets y se conecta a ellos

LOS DOS

- **IQN** (*iSCSI Qualified Name*) — nombre único de cada target y de cada initiator: `iqn.2026-10.org.example:almacen` (año-mes, dominio al revés y nombre)

Implementaciones en Linux

SERVIDORES (TARGET)

- `tgt` — sencillo, configuración por archivos
- **Linux-IO (LIO)** — actual, gestionado con `targetcli`
- `scst`, `istgt` — alternativas menos extendidas

CLIENTE (INITIATOR)

- `open-iscsi` — implementación estándar en Linux
- Gestionado con la herramienta `iscsiadm`

ℹ️ En estos ejemplos usaremos **tgt** en el servidor y **open-iscsi** en el cliente.

Servidor: configuración del target

Tras `sudo apt install tgt`, fichero `/etc/tgt/conf.d/almacen.conf`:

```
<target iqn.2026-10.org.example:almacen>
    backing-store /dev/vgalmacen/lun1
    backing-store /dev/vgalmacen/lun2
    initiator-address 192.168.200.20
    incominguser usuario secreto123456
</target>
```

- **`backing-store`** — cada uno es una LUN · **`initiator-address`** — IP que puede conectarse
- **`incominguser`** — usuario y contraseña **CHAP** (la contraseña, de 12 caracteres o más)

Servidor: aplicar y comprobar

```
sudo systemctl restart tgt
sudo tgtadm --lld iscsi --op show --mode target
```

En la salida se ve el target, sus **LUN**, la cuenta CHAP y las IP permitidas:

- **LUN 0** es el controlador del target; los discos son la **LUN 1** y la **LUN 2**
- Al reiniciar, `tgt` vuelve a leer los ficheros de `/etc/tgt/conf.d/`

ℹ️ También se puede crear un target en caliente con `tgtadm --op new`, pero se pierde al reiniciar: para algo permanente, el fichero.

Cliente: instalación y descubrimiento

```
sudo apt install open-iscsi
cat /etc/iscsi/initiatorname.iscsi        # IQN de este initiator
```

Descubrir los targets que publica el portal:

```
$ sudo iscsiadm -m discovery -t sendtargets -p 192.168.200.10
192.168.200.10:3260,1 iqn.2026-10.org.example:almacen
```

El descubrimiento guarda el target en la base de datos del initiator (los **nodos**), donde se configura antes de conectarse.

⚠️ Si **repites** el descubrimiento, el nodo se actualiza con los valores por defecto de `/etc/iscsi/iscsid.conf` y se pierde lo configurado (CHAP, `node.startup`): tendrías que volver a configurarlo.

Cliente: autenticación CHAP

Antes de conectarse, hay que guardar en el nodo el método, el usuario y la contraseña:

```
T="-m node -T iqn.2026-10.org.example:almacen -p 192.168.200.10"

sudo iscsiadm $T -o update -n node.session.auth.authmethod -v CHAP
sudo iscsiadm $T -o update -n node.session.auth.username   -v usuario
sudo iscsiadm $T -o update -n node.session.auth.password   -v secreto123456
```

⚠️ Sin `authmethod CHAP` el cliente no envía las credenciales y el login falla (`authorization failure`).

Cliente: conectar al target

```
sudo iscsiadm -m node -T iqn.2026-10.org.example:almacen \
              -p 192.168.200.10 --login
```

El kernel detecta **un disco nuevo por cada LUN**:

```
$ lsblk -S
NAME HCTL     TYPE VENDOR   MODEL          REV  TRAN
sda  2:0:0:1  disk IET      VIRTUAL-DISK   0001 iscsi
sdb  2:0:0:2  disk IET      VIRTUAL-DISK   0001 iscsi
$ sudo mkfs.ext4 /dev/sda           # comprueba antes con lsblk cuál es
$ sudo mkdir -p /srv/iscsi && sudo mount /dev/sda /srv/iscsi
```

Sesiones y desconexión

```
# Sesiones activas (con -P 3, también los discos de cada sesión)
sudo iscsiadm -m session
sudo iscsiadm -m session -P 3 | grep "Attached scsi disk"

# Desconectar de un target concreto
sudo iscsiadm -m node -T iqn.2026-10.org.example:almacen \
              -p 192.168.200.10 --logout
```

⚠️ Antes de cerrar la sesión, **desmonta** el dispositivo: si hay datos en uso, el sistema puede quedar en un estado inconsistente.

Reconexión automática al arrancar

```
sudo iscsiadm -m node -T iqn.2026-10.org.example:almacen \
              -p 192.168.200.10 --op update \
              -n node.startup -v automatic
```

A partir de ese momento, el cliente **vuelve a conectarse** al target en cada arranque (lo hace el servicio `open-iscsi`).

> ℹ️ La sesión no monta el disco: para eso, una **unidad** `.mount` que espere a que haya sesión.

Unidad de montaje para iSCSI

El nombre `/dev/sda` puede **cambiar** al reiniciar: se monta por **UUID** (`sudo blkid /dev/sda`).

`/etc/systemd/system/srv-iscsi.mount` en **servidorweb**:

```
[Unit]
Requires=open-iscsi.service
Wants=network-online.target
After=open-iscsi.service network-online.target

[Mount]
What=/dev/disk/by-uuid/7c1e0d2a-5b3f-4e8a-9d61-2f0c4a8b9e15
Where=/srv/iscsi
Type=ext4
Options=_netdev

[Install]
WantedBy=remote-fs.target
```

NFS vs iSCSI — cuándo elegir cada uno

	NFS (NAS)	iSCSI (SAN)
Granularidad	Archivos y directorios	Dispositivo de bloques
Quién formatea	El servidor	El cliente
Acceso simultáneo	✅ pensado para varios clientes	⚠️ un cliente a la vez (sin FS clusterizado)
Rendimiento	Bueno para ficheros	Mejor para bases de datos y máquinas virtuales
Caso típico	Compartir documentos, *home* de usuarios, backups	Discos para VMs, BBDD, almacenamiento dedicado

Para profundizar

- Documentación oficial de NFS (kernel.org)
- Wiki de Open-iSCSI
- Manual de tgt — Linux SCSI target framework
- systemd.mount

SRI · UNIDAD 2 — SERVICIOS DE ALMACENAMIENTO

¡Gracias!

Almacenamiento

📧 José Domingo Muñoz 🏫 IES Gonzalo Nazareno · Dos Hermanas 📚 SRI