SAN con iSCSI y sistemas de ficheros OCFS2

CDA 2018/19

25 de septiembre de 2018

Índice

1. Descripción

Simular un escenario de despliegue de una solución SAN (*Storage Area Network*) combinando:

- SAN iSCSI (*Internet Small Computer Systems Interface*)

- sistema de ficheros de cluster OCFS2
- array de discos RAID5 y LVM

Recursos complementarios

- Linux RAID Wiki
- iSCSI
- Documentación de Oracle Linux sobre OCFS2 (uso ligeramente distinto que sobre Debian)

2. Entorno de prácticas

2.1. Software de virtualización VIRTUALBOX

En estas prácticas se empleará el software de virtualización VIRTUALBOX para simular los equipos GNU/Linux sobre los que se realizarán las pruebas.

- Página principal: `http://virtualbox.org`
- Más información: `http://es.wikipedia.org/wiki/Virtualbox`

2.2. Imágenes a utilizar

1. Scripts de instalación
   - para GNU/Linux: ejercicio-iscsi.sh
     `alumno@pc: $ sh ejercicio-iscsi.sh`
   - para MS windows: ejercicio-iscsi.ps1
     `Powershell.exe -executionpolicy bypass -file ejercicio-iscsi.ps1`

   Notas:
   - Se pedirá un identificador (sin espacios) para poder reutilizar las versiones personalizadas de las imágenes creadas (usad por ejemplo el nombre del grupo de prácticas o el login LDAP)
   - En ambos scripts la variable `$DIR_BASE` especifica donde se descargarán las imágenes y se crearán las MVs. Por defecto en GNU/Linux será en `$HOME/CDA1819` y en Windows en `C:/CDA1819`.
     Puede modificarse antes de lanzar los scripts para hacer la instalación en otro directorio más conveniente (disco externo, etc)
   - Es posible descargar las imágenes comprimidas manualmente (o intercambiarlas con USB), basta descargar los archivos con extensión `.vdi.zip` de `http://ccia.esei.uvigo.es/docencia/CDA/1819/practicas/` y copiarlos en el directorio anterior (`$DIR_BASE`) para que el script haga el resto.
   - Si no lo hacen desde el script anterior, se pueden arrancar las instancias VIRTUALBOX desde el interfaz gráfico de VirtualBOX o desde la línea de comandos con `VBoxManage startvm <nombre MV>_<id>`
2. Imágenes descargadas
   - **base_cda.vdi** (0,65 GB comprimida, 2,9 GB descomprimida): Imagen genérica (común a todas las MVs) que contiene las herramientas a utilizar
     Contiene un sistema Debian 9 con herramientas gráficas y un entorno gráfico ligero LXDE (*Lighweight X11 Desktop Environment*) [LXDE].
   - **swap1GB.vdi**: Disco de 1 GB formateado como espacio de intercambio (SWAP)
3. Usuarios configurados e inicio en el sistema

- Usuarios disponibles

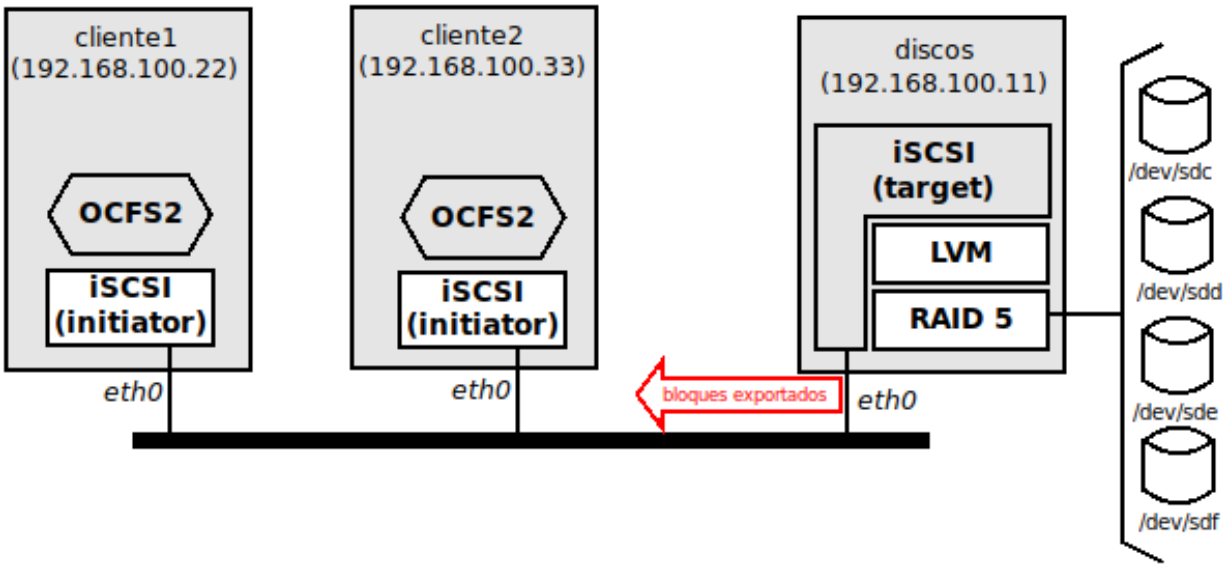
login	password
root	purple
usuario	usuario

- Acceso al entorno gráfico una vez logueado (necesario para poder copiar y pegar desde/hacia el anfitrión)

```
root@datos:~# startx
```

- Habilitar copiar y pegar desde/hacia el anfitrión en el menú `Dispositivos -> Portapapeles compartido -> bidir` de la ventana de la máquina virtual.

2.3. Máquinas virtuales y redes creadas



- `DISCOS` con la dirección 192.168.100.11
  - Jugará el papel de "cabina de discos", cuyos dispositivos de almacenamiento se combinarán con RAID5, el array resultante se gestionará con LVM y se expondrán sus volúmenes lógicos mediante iSCSI [Se corresponde con un *target* en terminología iSCSI]
  - Discos disponibles en la máquina virtual `DISCOS`
    - `/dev/sda`: disco principal, 16 GB (con la partición *ext4* `/dev/sda1` montada en `/`)
    - `/dev/sdb`: disco para swap, 1 GB (swap la en partición `/dev/sdb1`)
    - `/dev/sdc`: disco para ejercicios, 100 MB (no particionado)
    - `/dev/sdd`: disco para ejercicios, 100 MB (no particionado)
    - `/dev/sde`: disco para ejercicios, 100 MB (no particionado)
    - `/dev/sdf`: disco para ejercicios, 100 MB (no particionado)
- `CLIENTE1` con la dirección 192.168.100.22, cliente (*initiator*) del almacenamiento expuesto por `DISCOS`
- `CLIENTE2` con la dirección 192.168.100.33, cliente (*initiator*) del almacenamiento expuesto por `DISCOS`

3. PREVIO 1: Uso básico de iSCSI en GNU/Linux [no entregable]

3.1. Elementos iSCSI

- **Targets**: componentes responsable de hacer accesibles los dispositivos de bloques expuestos mediante iSCSI (≈ servidor)

- **LUNs** (*Logical Unit Number*): identificador único de cada uno de los dispositivos de bloques expuesto por un *target*
- **iqn**: identificador único (≈ URI) del *target* (sigue la convención del RFC-3720)
  `iqn:[fecha: yyyy-mm].[nombre del equipo invertido]:[identificador local del target]`
  ejemplos: `iqn:2018-09.net.cda.discos:prueba`, `iqn:2018-09.net.cda.discos:raid5.completo`

En GNU/Linux: proyecto TGT (*Linux SCSI target framework*) http://stgt.sourceforge.net/

- Instalación: `apt-get install tgt` (ya hecho en las MV de prácticas)
- Comando `tgtadm` (más detalles con `man tgtadm`)

- **Initiators**: componentes encargados gestionar la conexión con el *target* y hacer disponibles en los equipos cliente los dispositivos de bloques expuestos mediante iSCSI (≈ cliente)

  Los dispositivos de bloques que hace accesible el *initiator* (cada uno identificado por si propio LUN) aparecerán en el sistema "cliente" como discos SCSI convencionales

  Cuando están implementados en hardware (≈ tarjeta) se denominan HBA(*Host Bus Adapter*).

  En GNU/Linux: Linux Open-iSCSI Initiator https://github.com/open-iscsi/open-iscsi

  - Instalación: `apt-get install open-iscsi` (ya hecho en las MV de prácticas)
  - Comando `iscsiadm` (más detalles con `man iscsiadm`)

3.2. Configuración del Target iSCSI y exposición el dispositivo /dev/sdf

1. Crear target (servidor que expone los dispositivos de bloque) asignando un `iqn` y un identificador interno (`tid=1`)

```
root@discos:~# tgtadm --lld iscsi --mode target --op new --tid=1 --targetname iqn.2018-09.net.cda.discos:pruebas

root@discos:~# tgtadm --lld iscsi --mode target --op show
Target 1: iqn.2018-09.net.cda.discos:pruebas
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET     00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
  Account information:
  ACL information:
```

2. Añadir un LUN (dispositivo de bloque a exponer) dentro del *target* creado (`tid=1`) vinculado al dispositivo de bloques local `/dev/sdf` (parámetro `--backing-store`)

   Nota: El LUN 0 está reservado, identifica al controlador del propio *target* ⇒ LUN se empiezan a asignar desde 1

```
root@discos:~# tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 --backing-store /dev/sdf

root@discos:~# tgtadm --lld iscsi --mode target --op show
Target 1: iqn.2018-09.net.cda.discos:pruebas
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      ...
    LUN: 1
      Type: disk
      SCSI ID: IET     00010001
      SCSI SN: beaf11
      Size: 105 MB, Block size: 512
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: rdwr
      Backing store path: /dev/sdf
      Backing store flags:
  Account information:
  ACL information:
```

3. Control de accesos

   a) Restringir acceso al *target* a las direcciones IP de los *initiators* de CLIENTE1 (192.168.100.22) y CLIENTE2 (192.168.100.33)

   ```
   root@discos:~#  tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.100.22
   root@discos:~#  tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.100.33
   ```

   Nota 1: con la opción `--op unbind` se deshabilita el acceso desde una dirección IP dada **Nota 2:** puede habilitarse/deshabilitarse el acceso a cualquier dirección IP usando el valor `ALL` (no aplicable en nuestro caso)

   ```
   tgtadm --lld iscsi --mode target --op bind   --tid 1 --initiator-address ALL
   tgtadm --lld iscsi --mode target --op unbind --tid 1 --initiator-address ALL
   ```

   b) Definir usuario y contraseñas para control de acceso mediante CHAP
   En iSCSI se puede usar un mecanismo de autenticación mutua (*initiator* ante *target* [*initiator authentication*] y, opcionalmente, de *target* ante *initiator* [*target authentication*]) basado en el protocolo CHAP *Challenge-Handshake Authentication Protocol*.

   - Es necesario crear el par usuario-contraseña y vincular el usuario al *target* (y opcionalmente al *initiator* en autenticación inversa).

   ***Initiator authetication*:** el *initiator* es autneticado por el *target*

   1) Crear un par usuario-contraseña (`cda/cdapass`)
      ```
      root@discos:~# tgtadm --lld iscsi --mode account --op new --user cda --password cdapass
      ```
   2) Vincular usuario `cda` con el *target* al que tendrá acceso (`tid=1`)
      ```
      root@discos:~# tgtadm --lld iscsi --mode account --op bind --tid 1 --user cda
      ```

   Nota: En los *initiators* `open-iscsi` debe habilitarse la autenticación CHAP e incluirse el par `cda/cdapass` en el fichero de configuración `/etc/iscsi/iscsid.conf` (parámetros `node.session.auth.{username,password}`)

   c) Verificar resultado

   ```
   root@discos:~# tgtadm --lld iscsi --mode target --op show
   Target 1: iqn.2018-09.net.cda.discos:pruebas
     System information:
   ```

```
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      ...
    LUN: 1
      Type: disk
      SCSI ID: IET     00010001
      SCSI SN: beaf11
      Size: 105 MB, Block size: 512
      Online: Yes
      ...
      Backing store path: /dev/sdf
      ...
  Account information:
    cda
  ACL information:
    192.168.100.22
    192.168.100.33
```

4. (Opcional) Eliminar LUNs y *target* (por ese orden)

```
root@discos:~# tgtadm --lld iscsi --mode logicalunit --op delete --tid 1 --lun 1

root@discos:~# tgtadm --lld iscsi --mode target --op delete --tid 1
```

3.2.1. EXTRA: Configuración permanente de targets

<pendiente>

3.3. Configuración y uso de los initiators

1. Configurar los *initiators* `!open-iscsi!` de CLIENTE1 y CLIENTE2

   a) Establecer valores del fichero `/etc/iscsi/initiatorname.iscsi` (en ambos *initiators*)
   Nota: este paso no es estrictamente necesario, en una instalación normal del paquete `open-iscsi` se genera este fichero con un nombre de *initiator* aleatorio. En nuestro caso, como ambas máquinas comparten la misma imagen base es necesario forzar que sean diferentes, volviendo a generar esos nombres.

   ```
   root@cliente1:~# echo "InitiatorName="`iscsi-iname` > /etc/iscsi/initiatorname.iscsi

   root@cliente2:~# echo "InitiatorName="`iscsi-iname` > /etc/iscsi/initiatorname.iscsi
   ```

   b) Habilitar la autenticación mediante CHAP y establecer las credenciales de acceso en `/etc/iscsi/iscsid.conf` (en ambos *initiators*)

   ```
   root@cliente1:~# nano /etc/iscsi/iscsid.conf

   root@cliente2:~# nano /etc/iscsi/iscsid.conf


   --- CONTENIDO A EDITAR ----

   ...
   # *************
   # CHAP Settings
   # *************
   node.session.auth.authmethod = CHAP
   ```

```
node.session.auth.username = cda
node.session.auth.password = cdapass
...

--- CONTENIDO A EDITAR ----
```

c) Reiniciar el demonio `iscsid` para que cargue la nueva configuración (en ambos *initiators*)

```
root@cliente1:~# systemctl restart iscsid.service

root@cliente2:~# systemctl restart iscsid.service
```

2. Consultar la lista de `iqn` de los *targets* disponibles en la máquina `discos.cda.net` (192.168.100.11) (comando `sendtargets` del modo `discover` desde ambos `initiators`)

```
root@cliente1:~# iscsiadm --mode discovery --type sendtargets --portal 192.168.100.11
192.168.100.11:3260,1 iqn.2018-09.net.cda.discos:pruebas

root@cliente2:~# iscsiadm --mode discovery --type sendtargets --portal 192.168.100.11
192.168.100.11:3260,1 iqn.2018-09.net.cda.discos:pruebas
```

Como resultado de este comando `sendtarget` se actualiza la información de los *target* conocidos que se mantiene en el directorio `/etc/iscsi/nodes/[iqn]/[port]/defult`.

- En cualquier momento se puede consultar la lista de *target*/nodos conocidos

```
root@cliente1:~# iscsiadm --mode node
192.168.100.11:3260,1 iqn.2018-09.net.cda.discos:pruebas
```

- La configuración para el acceso al *target* `iqn.2018-09.net.cda.discos:pruebas` está en el fichero `/etc/iscsi/nodes`

```
root@cliente1:~# cat "/etc/iscsi/nodes/iqn.2018-09.net.cda.discos:pruebas/192.168.100.11,3260,1/default"
# BEGIN RECORD 2.0-874
node.name = iqn.2018-09.net.cda.discos:pruebas
node.tpgt = 1
node.startup = manual
node.leading_login = No
iface.iscsi_ifacename = default
iface.transport_name = tcp
...
node.discovery_address = 192.168.100.11
node.discovery_port = 3260
node.discovery_type = send_targets
node.session.initial_cmdsn = 0
node.session.initial_login_retry_max = 8
node.session.xmit_thread_priority = -20
node.session.cmds_max = 128
node.session.queue_depth = 32
node.session.nr_sessions = 1
node.session.auth.authmethod = CHAP
node.session.auth.username = cda
node.session.auth.password = cdapass
...
node.conn[0].address = 192.168.100.11
node.conn[0].port = 3260
node.conn[0].startup = manual
...
# END RECORD
```

- Es posible actualizar esa información con el comando `--op=update` del modo `node`.

  Por ejemplo, para ajustar las credenciales de acceso en caso de no corresponderse con los valores globales de `/etc/iscsi/iscsid.conf` (no necesario en nuestro ejemplo)

```
root@cliente1:~# iscsiadm --mode node --targetname "iqn.2018-09.net.cda.discos:pruebas" \
                          --portal "192.168.100.11:3260" \
                          --op=update --name node.session.auth.authmethod --value=CHAP

root@cliente1:~# iscsiadm --mode node --targetname "iqn.2018-09.net.cda.discos:pruebas" \
                          --portal "192.168.100.11:3260" \
                          --op=update --name node.session.auth.username --value=someuser

root@cliente1:~# iscsiadm --mode node --targetname "iqn.2018-09.net.cda.discos:pruebas" \
                          --portal "192.168.100.11:3260" \
                          --op=update --name node.session.auth.password --value=secret
```

3. Conectar con el *target* seleccionado de la máquina `discos.cda.net` (192.168.100.11) indicando su `iqn` (comando `login` del modo `node` desde ambos `initiators`)

```
root@cliente1:~# iscsiadm  --m node  --targetname iqn.2018-09.net.cda.discos:pruebas --portal 192.168.100.11 --l
Logging in to [iface: default, target: iqn.2018-09.net.cda.discos:pruebas, portal: 192.168.100.11,3260] (multipl
Login to [iface: default, target: iqn.2018-09.net.cda.discos:pruebas, portal: 192.168.100.11,3260] successful.


root@cliente2:~# iscsiadm  --m node  --targetname iqn.2018-09.net.cda.discos:pruebas --portal 192.168.100.11 --l
Logging in to [iface: default, target: iqn.2018-09.net.cda.discos:pruebas, portal: 192.168.100.11,3260] (multipl
Login to [iface: default, target: iqn.2018-09.net.cda.discos:pruebas, portal: 192.168.100.11,3260] successful.
```

4. Si la conexión ha tenido éxito en ambas máquinas (CLIENTE1 y CLIENTE2) deberá aparecer un nuevo dispositivo de bloques de tipo SCSI, vinculado a `/dev/sdc` y con 100 MB de capacidad.

```
root@cliente1:~# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   16G  0 disk
|--sda1   8:1    0   16G  0 part /
sdb      8:16   0    1G  0 disk
|--sdb1   8:17   0 1022M  0 part [SWAP]
sdc      8:32   0  100M  0 disk

root@cliente1:~# parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 17,2GB
...

Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
...

Error: /dev/sdc: unrecognised disk label
Model: IET VIRTUAL-DISK (scsi)
Disk /dev/sdc: 105MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:

root@cliente1:~# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   16G  0 disk
|--sda1   8:1    0   16G  0 part /
sdb      8:16   0    1G  0 disk
|--sdb1   8:17   0 1022M  0 part [SWAP]
sdc      8:32   0  100M  0 disk

root@cliente2:~# parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 17,2GB
...
```

```
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
...

Error: /dev/sdc: unrecognised disk label
Model: IET VIRTUAL-DISK (scsi)
Disk /dev/sdc: 105MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

También puede verificarse desde el *target* en la máquina DISCOS la existencia de dos *initiators* accediendo al LUN 1 mediante iSCSI.

```
root@discos:~# tgtadm --lld iscsi --mode target --op show
Target 1: iqn.2018-09.net.cda.discos:pruebas
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
    I_T nexus: 65
      Initiator: iqn.2005-03.org.open-iscsi:b8876af7eaa alias: cliente1.cda.net
      Connection: 0
        IP Address: 192.168.100.22
    I_T nexus: 66
      Initiator: iqn.2005-03.org.open-iscsi:202d6339674d alias: cliente2.cda.net
      Connection: 0
        IP Address: 192.168.100.33
  LUN information:
    LUN: 0
      Type: controller
      ...
    LUN: 1
      Type: disk
      ...
  Account information:
    cda
  ACL information:
    192.168.100.22
    192.168.100.33
```

5. Desconexión de un *target* (comando `logout` del modo `node`) [**no necesario** de momento en nuestro ejemplo]

```
root@cliente1:~# iscsiadm  --m node  --targetname iqn.2018-09.net.cda.discos:pruebas --portal 192.168.100.11 --l

root@cliente2:~# iscsiadm  --m node  --targetname iqn.2018-09.net.cda.discos:pruebas --portal 192.168.100.11 --l
```

3.3.1. Uso incorrecto de dispositivos iSCSI compartidos

En este punto del ejemplo ambos *initiator* (CLIENTE1 y CLIENTE2) cuentan con un nuevo dispositivo de bloques SCSI (con el nombre `/dev/sdc`), que se corresponde con el dispositivo `/dev/sdf` expuesto mediante iSCSI por el *target* DISCOS.

- Cada equipo (CLIENTE1 ó CLIENTE2) "ve" ese dispositivo como un dispositivo local propio y, en principio, de uso exclusivo.
- En teoría, es posible que cada uno de esos equipos (CLIENTE1 ó CLIENTE2) puede particionarlo, formatearlo y montarlo como un dispositivo de bloques normal.

- No obstante, como veremos, si ambos equipos tratan ese dispositivo de bloques compartido como un dispositivo local, el comportamiento no es predecible y con el tiempo, las estructuras del sistema de ficheros usado acabaran corruptas (al ser usadas a la vez por dos sistemas diferentes)

1. Formatear el dispositivo iSCSI compartido con un sistema de ficheros `ext3` desde la máquina `CLIENTE1`, crear un punto de montaje, montarlo como `ext3`, crear uno o más ficheros y desmontar el dispositivo

```
root@cliente1:~# mkfs.ext3 /dev/sdc

root@cliente1:~# mkdir /mnt/prueba_en_uno

root@cliente1:~# mount -t ext3  /dev/sdc /mnt/prueba_en_uno

root@cliente1:~# touch /mnt/prueba_en_uno/creado_por_uno_{0,1,2}.txt

root@cliente1:~# umount /dev/sdc
```

2. En la máquina `CLIENTE2`, crear un punto de montaje, montar el dispositivo iSCSI compartido como `ext3` y verificar que los archivos creados por `CLIENTE1` están en el disco.

   Nota: no es necesario formatearlo de nuevo, ya se hizo en `CLIENTE1`

```
root@cliente2:~# mkdir /mnt/prueba_en_dos

root@cliente2:~# mount -t ext3  /dev/sdc /mnt/prueba_en_dos

root@cliente2:~# ls -l /mnt/prueba_en_dos/

root@cliente2:~# touch /mnt/prueba_en_dos/creado_por_dos_{0,1,2}.txt

root@cliente2:~# umount /dev/sdc
```

3. Montar en ambas máquinas el dispositivo iSCSI compartido como `ext3` y probar varias operaciones (crear, modificar, borrar ficheros) sobre el sistema de ficheros alternando ambas máquinas.

4. Si se desmonta el dispositivo en ambas máquinas y se vuelve a montar, comprobar qué cambios se han mantenido y cuáles se han perdido

   Conclusión: No es posible usar sistemas de ficheros convencionales (ext3, por ejemplo) simultáneamente en dos o más equipos que compartan un mismo dispositivo iSCSI.

   - Son necesarios sistemas de ficheros *cluster aware* como OCFS2, donde los nodos que hagan uso del dispositivo compartido se coordinen entre ellos.

3.3.2. EXTRA: Configuración permanente de initiators

`<pendiente>`

4. PREVIO 2: Uso del sistema de ficheros en cluster OCFS2 [no entregable]

4.1. Funcionamiento OCFS2 y comandos básicos

Como se ha comprobado, en un escenario como el utilizado en este ejemplo (un dispositivo de bloques expuesto en la red mediante iSCSI) u otros similares donde varios equipos puedan acceder simultáneamente a un dispositivo de bloques compartido, los sistema de ficheros convencionales (`ext3, ext4, ntfs` y similares) no pueden ser utilizados,

dado que por su diseño asumen que el dispositivo de bloques sólo es accedido por una única máquina. Sólo en el caso de garantizar el montaje en modo sólo escritura del dispositivo compartido por parte de todos los participantes, se podría usar esa opción de modo fiable.

Para garantizar la consistencia de los datos y evitar la corrupción de las estructuras de datos propias del sistema de ficheros ante ese uso concurrente del dispositivo de almacenamiento, es necesario hacer uso de sistemas de ficheros *"de cluster"*. Se trata de sistemas de ficheros que añaden una capa de coordinación adicional sobre el almacenamiento compartido que coordina el acceso de los equipos finales a los bloques de datos mediante mecanismos de bloqueo (*lock*) distribuidos.

Ejemplos de este tipo de sistemas de ficheros serían:

- OCFS2 (*Oracle Cluster File System*), desarrollado y mantenido por Oracle.
- GFS/GFS2 (*Global File System*), desarrollado y mantenido por Red Hat.
- CSV (*Cluster Shared Volumes*), de Microsoft, usado en soluciones de alta disponibilidad sobre entornos de virtualización en MS Windows.

4.1.1. Paquetes necesarios (ya instalados)

```
root@cliente1:~# apt-get install ocfs2-tools

root@cliente2:~# apt-get install ocfs2-tools
```

4.1.2. Comandos básicos

Como resultado de la instalación del paquete `ocsf2-tools` estarán disponibles, entre otros, los siguientes comandos/herramientas

- `mkfs.ocfs2`: formatea un dispositivo con un sistema de ficheros OCFS2
- `mount.ocfs2`: permite montar un dispositivo con un sistema de ficheros OCFS2 (usando mediante `mount -t ocfs2`)
- `tunefs.ocfs2`: permite configurar los parámetros de un sistema de ficheros OCFS2 (tamaño, journaling, etc)
- `debugfs.ocfs2`: muestra información sobre un sistema de ficheros OCFS2
- `fsck.ocfs2`: comprueba y corrige defectos de un sistema de ficheros OCFS2
- `/etc/init.d/o2cb`: script de inicio y control del cluster OCFS2
- `o2cb_ctl`: herrramienta de consulta y control del estado del cluster OCFS2 (añadir/eliminar nodos, etc)

En el caso de OCFS2 los nodos que forman parte del cluster se comunican y coordinan a través del puerto TCP 7777 (puerto por defecto).

- Dicho puerto (o el que se use en su lugar) debe estar disponible en las máquinas del cluster y los posibles contafuegos que estén en uso deben permitir ese tipo de tráfico en ambos sentidos.

4.2. Configuración y uso del cluster OSCF2

1. Declarar el *cluster* formado por las máquinas `CLIENTE1` y `CLIENTE2`

   El script `o2cb`, instalado como parte del paquete `ocf2-tools`, se encarga de la gestión del cluster

- Funciona como un script de arranque (está ubicado en `/etc/init.d`) pero también es posible invocarlo desde línea de comandos con `service o2cb`
  - `service o2cb load`: Fuerza la carga de los módulos del kernel necesarios
  - `service o2cb online [nombre cluster]`: Pone en marcha el cluster indicado (o todos los que estén definidos en `/etc/ocfs2/cluster.conf`, si no se especifica un nombre), incluyendo la incorporación del nodo donde se invoca a dicho cluster/es.
  - `service o2cb start`: Combina las dos acciones anteriores
  - `service o2cb offline [nombre cluster]`: Finalizar el cluster indicado (o todos los que estén definidos en `/etc/ocfs2/cluster.conf`, si no se especifica un nombre)
  - `service o2cb unload`: Descarga del kernel los módulos necesarios para dar soporte a OCSFS2.
  - `service o2cb stop`: Combina las dos acciones anteriores
  - `service o2cb configure`: En sistemas Debian está desahilitado y delega en una configuración "manual" o mediante `dpkg-reconfigure ocsfs2-tools`
  - `service o2cb status`: Muestra el estado del servicio O2CB.
- La definición del cluster está en el fichero `/etc/ocfs2/cluster.conf` y debe ser la misma en todos los nodos del cluster (de modo que estos puedan comunicarse entre ellos)
  - Puede editarse manualmente o mediante las herramientas `o2cb_ctl` (línea de comandos) o `ocfs2console` (interfaz gráfico)
- En los sistemas basados en Debian la configuración del script `o2cb` con los parámetros a usar por el cluster (timeouts, nombre de cluster a arrancar) está en `/etc/default/o2cb`

a) Editar el fichero de definición del cluster `/etc/ocfs2/cluster.conf` en `CLIENTE1` (creándolo si no existe)

```
root@cliente1:~# nano /etc/ocfs2/cluster.conf

cluster:
heartbeat_mode = local
node_count = 2
name = clustercda

node:
number = 0
cluster = clustercda
ip_port = 7777
ip_address = 192.168.100.22
name = cliente1

node:
number = 1
cluster = clustercda
ip_port = 7777
ip_address = 192.168.100.33
name = cliente2
```

Nota: Otra alternativa es configurar el cluster OCSFS2 empleando el comando `o2cb`

```
root@cliente1:~# o2cb add-cluster clustercda
root@cliente1:~# o2cb add-node clustercda cliente1 --ip 192.168.100.22
root@cliente1:~# o2cb add-node clustercda cliente2 --ip 192.168.100.33
```

Importante: Las tabulaciones de las diferentes secciones del documento son relevantes.

b) Copiar el fichero `/etc/ocfs2/cluster.conf` a `CLIENTE2` (o volver a crearlo desde cero)

```
root@cliente1:~# scp /etc/ocfs2/cluster.conf root@192.168.100.33:/etc/ocfs2/
```

Importante: Para que esta copia remota con `scp` pueda hacerla el usuario `root`, debe habilitarse en `CLIENTE2` la opción de *login* SSH como `root` (deshabilitada por defecto como medida de seguridad) y reiniciar el servicio.

```
root@cliente2:~# nano /etc/ssh/sshd_config

...
# Authentication:
#PermitRootLogin without-password
PermitRootLogin yes
...

root@cliente2:~# service sshd restart
```

c) Editar en ambas máquinas (`CLIENTE1` y `CLIENTE2`) el fichero `/etc/default/o2cb` con la configuración de inicio del servicio `o2cb`

```
root@cliente1:~# nano /etc/default/o2cb
root@cliente2:~# nano /etc/default/o2cb
```

Asegurar, en ambos casos, que el parámetro `O2CB_ENABLED` está habilitado (indica que se inicie OCFS2 en el arranque) y que el parámetro `O2CB_BOOTCLUSTER` se corresponde con el nombre de nuestro cluster (indica el cluster a iniciar/unirse).

```
...
# O2CB_ENABLED: 'true' means to load the driver on boot.
O2CB_ENABLED=true

# O2CB_BOOTCLUSTER: If not empty, the name of a cluster to start.
O2CB_BOOTCLUSTER=clustercda
...
```

- **Extra:** También es posible configurar los parámetros del servicio `o2cb` reconfigurando el paquete `ocfs2-tools` (no necesario en nuestro caso)

```
root@cliente1:~# dpkg-reconfigure ocfs2-tools

root@cliente2:~# dpkg-reconfigure ocfs2-tools
```

d) Arrancar/reiniciar el servicio `o2cb` **en ambas máquinas** para cargar la nueva configuración

```
root@cliente1:~# service o2cb stop
root@cliente1:~# service o2cb start
root@cliente1:~# service o2cb status
root@cliente1:\~{}\# service o2cb status
* o2cb.service - LSB: Load O2CB cluster services at system boot.
   Loaded: loaded (/etc/init.d/o2cb; generated; vendor preset: enabled)
   Active: active (running) since Tue 2018-09-25 09:37:50 CEST; 20s ago
     Docs: man:systemd-sysv-generator(8)
   Process: 1823 ExecStop=/etc/init.d/o2cb stop (code=exited, status=0/SUCCESS)
   Process: 1873 ExecStart=/etc/init.d/o2cb start (code=exited, status=0/SUCCESS)
     Tasks: 1 (limit: 4915)
     CGroup: /system.slice/o2cb.service
             |-- 1922 o2hbmonitor

sep 25 09:37:49 cliente1.cda.net systemd[1]: Starting LSB: Load O2CB cluster services at system boot....
sep 25 09:37:49 cliente1.cda.net o2cb[1873]: Loading stack plugin "o2cb": OK
sep 25 09:37:49 cliente1.cda.net o2cb[1873]: Loading filesystem "ocfs2\_dlmfs": OK
sep 25 09:37:49 cliente1.cda.net o2cb[1873]: Creating directory '/dlm': OK
sep 25 09:37:49 cliente1.cda.net o2cb[1873]: Mounting ocfs2\_dlmfs filesystem at /dlm: OK
sep 25 09:37:49 cliente1.cda.net o2cb[1873]: Setting cluster stack "o2cb": OK
sep 25 09:37:50 cliente1.cda.net o2cb[1873]: Registering O2CB cluster "clustercda": OK
sep 25 09:37:50 cliente1.cda.net o2cb[1873]: Setting O2CB cluster timeouts : OK
sep 25 09:37:50 cliente1.cda.net systemd[1]: Started LSB: Load O2CB cluster services at system boot..
sep 25 09:37:50 cliente1.cda.net o2hbmonitor[1922]: Starting


root@cliente2:~# service o2cb stop
root@cliente2:~# service o2cb start
root@cliente2:~# service o2cb status
* o2cb.service - LSB: Load O2CB cluster services at system boot.
```

```
   Loaded: loaded (/etc/init.d/o2cb; generated; vendor preset: enabled)
   Active: active (running) since Tue 2018-09-25 09:40:16 CEST; 1s ago
     Docs: man:systemd-sysv-generator(8)
   Process: 2036 ExecStop=/etc/init.d/o2cb stop (code=exited, status=0/SUCCESS)
   Process: 2086 ExecStart=/etc/init.d/o2cb start (code=exited, status=0/SUCCESS)
     Tasks: 1 (limit: 4915)
     CGroup: /system.slice/o2cb.service
             |-2134 o2hbmonitor

sep 25 09:40:15 cliente2.cda.net systemd[1]: Starting LSB: Load O2CB cluster services at system boot....
sep 25 09:40:16 cliente2.cda.net o2cb[2086]: Loading stack plugin "o2cb": OK
sep 25 09:40:16 cliente2.cda.net o2cb[2086]: Loading filesystem "ocfs2_dlmfs": OK
sep 25 09:40:16 cliente2.cda.net o2cb[2086]: Creating directory '/dlm': OK
sep 25 09:40:16 cliente2.cda.net o2cb[2086]: Mounting ocfs2_dlmfs filesystem at /dlm: OK
sep 25 09:40:16 cliente2.cda.net o2cb[2086]: Setting cluster stack "o2cb": OK
sep 25 09:40:16 cliente2.cda.net o2cb[2086]: Registering O2CB cluster "clustercda": OK
sep 25 09:40:16 cliente2.cda.net o2cb[2086]: Setting O2CB cluster timeouts : OK
sep 25 09:40:16 cliente2.cda.net systemd[1]: Started LSB: Load O2CB cluster services at system boot..
sep 25 09:40:16 cliente2.cda.net o2hbmonitor[2134]: Starting
```

- **Nota 1:** en lugar de `service o2cb start` podría haberse usado la combinación `service o2cb load`, seguido de `service o2cb online cluster_cda` (en este caso los resultados son idénticos, esta alternativa tendría sentido en un escenario con varios clusters OCFS2 definidos)
- **Nota 2:** En este momento, desde cualquiera de los nodos del cluster se puede ejecutar la herramienta gráfica `ocfs2console`, para ver y gestionar el cluster (añadir/liminar nodos, formatear dispositivos, montar el sistema de ficheros).

2. Formatear en una de las máquinas el dispositivo AoE expuesto en el apartado anterior con un sistema de ficheros OCFS2

   - Se deberá usar el comando `mkfs.ocfs2` ó `mkfs -t ocfs2`
   - **Importante:** sólo es necesario hacerlo en una de las máquinas
     La situación de los sistemas de ficheros OCFS2 usados por los nodos pertenecientes al cluster es sincronizada por el servicio de gestión del cluster `o2cb`.
   - Puede incluirse la opción `-L [etiqueta]` para vincular una etiqueta al volumen OCFS2.
     - De este modo será posible montar el sistema de ficheros OCFS2 indicando la etiqueta en lugar del nombre del dispositivo a montar
     - Este uso de etiquetas puede ser conveniente con tecnologías de almacenamiento donde los nombres de dispositivos no sean predecibles y/o conocidos de antemano

```
root@cliente1:~# mkfs -t ocfs2 /dev/sdc
mkfs.ocfs2 1.8.4
Cluster stack: classic o2cb
Label:
Features: sparse extended-slotmap backup-super unwritten inline-data strict-journal-super xattr indexed-dirs ref
Block size: 1024 (10 bits)
Cluster size: 4096 (12 bits)
Volume size: 104857600 (25600 clusters) (102400 blocks)
Cluster groups: 4 (tail covers 2560 clusters, rest cover 7680 clusters)
Extent allocator size: 2097152 (1 groups)
Journal size: 4194304
Node slots: 2
Creating bitmaps: done
Initializing superblock: done
Writing system files: done
Writing superblock: done
Writing backup superblock: 0 block(s)
Formatting Journals: done
Growing extent allocator: done
Formatting slot map: done
Formatting quota files: done
```

```
Writing lost+found: done
mkfs.ocfs2 successful
```

3. Montar en ambos "clientes" el sistema de ficheros OCFS2 sobre el punto de montaje del ejemplo anterior (`/mnt/prueba_en_uno`, `/mnt/prueba_en_dos`) y comprobar su funcionamiento (crear archivos en una máquina, comprobar que son visibles desde la otra, eliminar y/o modificar archivos en máquinas distintas, etc)

```
root@cliente1:~# mkdir /mnt/prueba_en_uno
root@cliente1:~# mount -t ocfs2 /dev/sdc /mnt/prueba_en_uno
root@cliente1:~# touch /mnt/prueba_en_uno/ocfs_en_uno.{a,b,c}

root@cliente2:~# mkdir /mnt/prueba_en_dos
root@cliente2:~# mount -t ocfs2 /dev/sdc /mnt/prueba_en_dos
root@cliente1:~# touch /mnt/prueba_en_dos/ocfs_en_dos.{a,b,c}

root@cliente1:~# ls /mnt/prueba_en_uno

root@cliente2:~# ls /mnt/prueba_en_dos
```

5. Tareas entregables

5.1. Eliminar trazas del ejemplo anterior (no entregable)

1. Desmotar el sistema de ficheros

```
root@cliente1:~# umount /mnt/prueba_en_uno

root@cliente2:~# umount /mnt/prueba_en_dos
```

2. Desconectar ambos *initiator* del *target*

```
root@cliente1:~# iscsiadm  --m node  --targetname iqn.2018-09.net.cda.discos:pruebas --portal 192.168.100.11 --l

root@cliente2:~# iscsiadm  --m node  --targetname iqn.2018-09.net.cda.discos:pruebas --portal 192.168.100.11 --l
```

3. Eliminar la definición actual de LUNs y *target* en la máquina `DISCOS`

```
root@discos:~# tgtadm --lld iscsi --mode logicalunit --op delete --tid 1 --lun 1

root@discos:~# tgtadm --lld iscsi --mode target --op delete --tid 1
```

5.2. Tareas a realizar (entregable)

1. Definir un array de discos RAID5 con los 4 dispositivos disponibles en la máquina `DISCOS` (`/dev/sdc`,`/dev/sdd`,`/dev/sde`,`/d`
2. Crear un grupo de volúmenes LVM empleando como volumen físico el array RAID5 anterior y definir **tres volúmenes lógicos** con 50 MB de capacidad cada uno (con los nombres `UNO`,`DOS`,`COMPARTIDO`)
3. Definir el/los *target*/*targets* para hacer disponibles mediante iSCSI estos tres volúmenes lógicos
   a) sólo `CLIENTE1` tendrá acceso al volumen lógico `UNO`
   b) sólo `CLIENTE2` tendrá acceso al volumen lógico `DOS`
   c) `CLIENTE1` y `CLIENTE2` tendrán ambos acceso al volumen lógico `COMPARTIDO`

4. Configurar y conectar los *initiator* de `CLIENTE1` y `CLIENTE2` conforme a las restricciones anteriores
5. Formatear y montar los dispositivos iSCSI en las máquinas `CLIENTE1` y `CLIENTE2`
   a) En `CLIENTE1` se formateará como `ext3` el LUN correspondiente al volumen lógico `UNO` y se montará en `/mnt/uno`
   b) En `CLIENTE2` se formateará como `ext3` el LUN correspondiente al volumen lógico `DOS` y se montará en `/mnt/dos`
   c) En `CLIENTE1` se formateará como `ocfs2` (configurando previamente el cluster OCFS2 si fuera necesario) el LUN correspondiente al volumen lógico `COMPARTIDO` y tanto en `CLIENTE1` como en `CLIENTE2` se montará en `/mnt/compartido`
6. Comprobar el correcto montaje y funcionamiento de los dispositivos montados.

6. Entrega

Práctica **individual**

Documentación entregable (un único archivo PDF [sin tildes ni espacios en el nombre])

- Documentar los pasos realizados en las tareas de la sección 5.2
  - Detallar qué se pretende hacer en cada caso (justificar la definición de *targets* y LUNs realizada)
  - Indicar los comandos y opciones utilizados en cada caso
  - Detallar los resultados obtenidos (salidas por consola, etc)

Entrega: hasta Domingo 21/10/2018 en FATIIC