Introduction au réseau de stockage iSCSI

Philippe Latu
philippe.latu(at)inetdoc.net

https://www.inetdoc.net

Résumé

Ce support de travaux pratiques est consacré à l'étude d'une infrastructure de stockage illustrant les technologies DAS (Direct Attached Storage), SAN (Storage Area Network) et la redondance de niveau 1 (RAID1). Le protocole iSCSI est utilisé pour la partie SAN comme exemple d'accès «en mode bloc» aux unités de stockage réseau. La redondance de niveau 1 utilise les fonctions intégrées au noyau Linux. L'infrastructure proposée montre comment les différentes technologies élémentaires peuvent être combinées pour atteindre les objectifs de haute disponibilité et de sauvegarde.

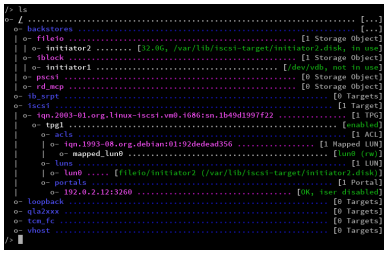


Table des matières

1. Copyright et Licence

1.1. Méta-information

Ce document est écrit avec *DocBook* XML sur un système *Debian GNU/Linux*. Il est disponible en version imprimable au format PDF : `sysadm-net.iscsi.qa.pdf`.

1.2. Conventions typographiques

Tous les exemples d'exécution des commandes sont précédés d'une invite utilisateur ou *prompt* spécifique au niveau des droits utilisateurs nécessaires sur le système.

- Toute commande précédée de l'invite `$` ne nécessite aucun privilège particulier et peut être utilisée au niveau utilisateur simple.
- Toute commande précédée de l'invite `#` nécessite les privilèges du super-utilisateur.

2. Adressage IP des postes de travail

À partir de l'infrastructure proposée dans la section suivante, on constitue des couples de postes de travail qui vont partager le même domaine de diffusion durant la séance de travaux pratiques.

Ces opérations de réaffectation du plan d'adressage IP sont répétées à chaque début de séance de travaux pratiques. Elles s'appuient sur les indications données dans le document *Architecture réseau des travaux pratiques*.

Tableau 1. Affectation des adresses et des réseaux IP

Poste 1	Poste 2	Passerelle par défaut
alderaan	bespin	`10.4.4.1/23`
centares	coruscant	`192.168.109.1/25`
dagobah	endor	`10.0.117.1/27`
felucia	geonosis	`10.7.10.1/23`
hoth	mustafar	`172.19.112.1/26`
naboo	tatooine	`192.168.111.1/25`

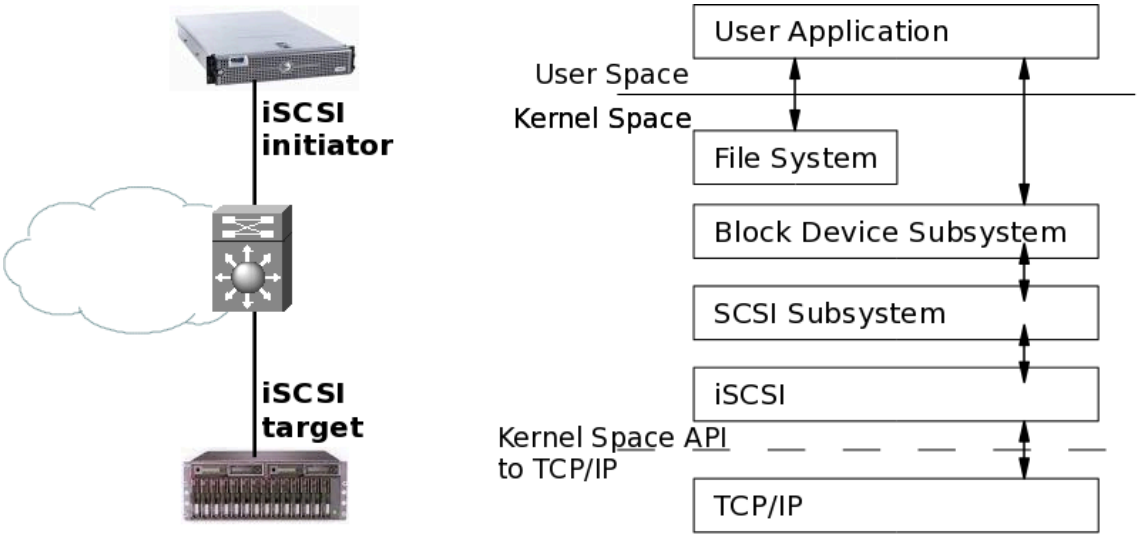
Une fois la passerelle du réseau IP affecté à chaque paire de postes de travaux pratiques, il faut rechercher dans le document *Architecture réseau des travaux pratiques* les éléments nécessaires à la connexion physique de ces postes. Les étapes usuelles sont les suivantes :

1. Attribuer une adresse IP à chacun des postes en fonction de l'espace d'adressage du réseau défini.
2. Rechercher le numéro de VLAN correspondant au réseau IP attribué.
3. Repérer le commutateur sur lequel des ports ont été affectés au VLAN recherché. Connecter les deux postes de travaux pratiques sur les ports identifiés.
4. Configurer les interfaces réseau de chaque poste : adresse, masque et passerelle par défaut. Valider la connectivité IP entre les deux postes puis avec les autres réseaux de l'infrastructure de travaux pratiques.

3. Technologie iSCSI et topologie de travaux pratiques

Cette section présente sommairement le protocole iSCSI puis attribue les rôles et les tâches de chacun des postes de travaux pratiques en fonction de la topologie mise en œuvre. Ce support fait suite à la présentation sur le *Stockage Réseau* utilisée en cours.

3.1. Bases de la technologie iSCSI



Topologie iSCSI basique - vue complète

La technologie iSCSI dont l'acronyme reprend la définition historique *Internet Small Computer System Interface* est un protocole réseau de stockage basé sur le modèle TCP/IP. Le principe de base consiste à encapsuler des commandes SCSI dans des paquets IP transmis entre un hôte et une unité de disque. Comme les paquets IP peuvent être perdus (et|ou) retransmis, ils peuvent très bien ne pas arriver dans l'ordre d'émission. Le protocole iSCSI doit donc conserver une trace de la séquence de transmission de commandes SCSI. Les commandes sont placées dans une file d'attente dans l'ordre d'émission.

Le protocole iSCSI a initialement été développée par *IBM* et a ensuite été soumise à l'IETF (*Internet Engineering Task Force*). Le standard a été publié par le comité *IP Storage Working Group* en août 2002.

On peut identifier deux fonctions principales dans la technologie iSCSI. La première est la fonction *target*. C'est un système simple qui possède le volume de stockage à publier sur le réseau IP. Ce système peut être matériel ou logiciel. Dans le cas de ces travaux pratiques, il s'agit d'un poste de travail utilisant son second disque dur ou bien un fichier comme unité de stockage SAN. La seconde fonction est baptisée *initiator*. Elle correspond au «client» qui utilise le volume de stockage réseau. Dans le contexte de ce document, l'autre poste de travaux pratiques joue ce rôle de client.

Fondamentalement, iSCSI est un protocole de la famille *Storage Area Network* (SAN). Le client ou *initiator* accède à une unité de stockage en *mode bloc*. Ce mode de fonctionnement est quasi identique à la technologie *Fibre Channel*. Le type de réseau constitue la principale différence entre ces deux technologies. La technologie iSCSI s'appuie sur TCP/IP alors que *Fibre Channel* comprend une définition de réseau propre (FC) qui nécessite des équipements spécifiques.

Ces dernières années, la technologie iSCSI gagne en popularité relativement à son ainée pour plusieurs raisons.

- Le prix des configurations iSCSI peut être bien meilleur marché qu'avec la technologie *Fibre Channel*. Si l'architecture du réseau de de stockage est adaptée, iSCSI devient très attractif.

  Il est important de bien identifier les fonctionnalités réseau que l'on associe à iSCSI pour accroître les performances du stockage. Dans ces fonctions complémentaires on trouve l'agrégation de canaux qui recouvre plusieurs dénominations et plusieurs standards de l'IEEE. Par exemple, elle est baptisée *bonding* sur les systèmes GNU/Linux et *etherchannel* sur les équipements *Cisco*. Côté standard, le *Link Aggregation Control Protocol* (LACP) pour Ethernet est couvert par les versions *IEEE 802.3ad*, *IEEE 802.1aq* et *IEEE 802.1AX*. L'utilisation de ces techniques est totalement transparente entre équipements hétérogènes. En appliquant ce principe d'agrégation de canaux, on peut pratiquement assimiler les performances de quatre liens

Gigabit Ethernet à celles d'un lien *Fibre Channel*. Une autre technique consiste à utiliser aussi plusieurs liens dans une optique et redondance et de balance de charge. Elle est appelée *multipath*.

- L'utilisation d'une technologie réseau unique est nettement moins complexe à administrer. En effet, on optimise les coûts, les temps de formation et d'exploitation en utilisant une architecture de commutation homogène. C'est un des avantages majeurs de la technologie Ethernet sur ses concurrentes.

- Au début de son exploitation, le coût d'un réseau 10 Gigabit Ethernet est prohibitif relativement à toutes les autres solutions. Du point de vue hôte, le point déterminant est l'uniformisation de l'interface réseau. En effet, avec une interface 10GigE on ne devrait plus avoir de distinction entre NIC et HBA.

Aujourd'hui la technologie iSCSI est supportée par tous les systèmes d'exploitation communs. Côté GNU/Linux, plusieurs projets ont vu le jour dans les années qui ont suivi la publication du standard en 2002. Pour la partie *initiator* les développements des deux projets phares ont fusionné pour ne plus fournir qu'un seul code source ; celui disponible à l'adresse *Open-iSCSI*. La partie *target* a suivi un processus analogue et le code source est disponible à l'adresse *Linux-IO : the Linux SCSI Target wiki*. La partie *Kernelspace* est maintenant intégrée dans l'arborescence principale du noyau Linux. La mise en œuvre du rôle *target* ne nécessite donc plus de manipulations spécifiques pour compiler les modules dédiés.

À partir des informations fournies à l'adresse *Linux-IO : the Linux SCSI Target wiki*, on recherche le paquet utile de la distribution pour la configuration du rôle *target* :

```
$ aptitude search targetcli
p   targetcli          - administration tool for managing LIO core target
```

En revanche, le choix du paquet pour le rôle *initiator* à l'aide de la liste ci-dessous est plus délicat à priori. Cependant, si on élimine les mentions relatives au rôle *target* et les bibliothèques, il ne reste plus que le paquet open-iscsi.

```
$ aptitude search iscsi
p   iscsitarget        - iSCSI Enterprise Target userland tools
p   iscsitarget-dkms   - iSCSI Enterprise Target kernel module source - dkms version
p   libiscsi-bin       - iSCSI client shared library - utilities
p   libiscsi-dev       - iSCSI client shared library
p   libiscsi1          - iSCSI client shared library
p   open-iscsi         - High performance, transport independent iSCSI implementation
```

3.2. Infrastructure de stockage étudiée

Le séquencement des opérations à réaliser lors de la séance de travaux pratiques est décrit dans le tableau ci-dessous. Les deux postes occupent chacun un rôle distinct. Comme le rôle *initiator* demande moins de travail de préparation, c'est à ce poste que l'on confie les essais des outils de *micro-benchmark*.

Tableau 2. Attribution des rôles

Rôle *initiator*	**Rôle *target***
Préparation d'une unité de stockage locale pour évaluer les différences entre les accès DAS et SAN	Préparation d'une unité de stockage iSCSI
Recherche et installation du ou des paquet(s) pour le rôle *initiator*	Recherche et installation du ou des paquet(s) pour le rôle *target*
Étude des outils de configuration du service	Étude des outils de configuration du service
Validation manuelle de la configuration SAN iSCSI	
Validation de la configuration système	
Validation de l'authentification mutuelle entre les rôles *initiator* et *target*	
Mise en place de la réplication synchrone avec un tableau RAID1 entre unité de disque locale et le volume iSCSI	Mise en place de la réplication asynchrone avec un volume logique de type *snapshot* de sauvegarde des fichiers images de volume de stockage
Étude comparative des performances d'accès	

4. Préparation d'une unité de stockage

Dans cette section on présente les manipulations à effectuer pour préparer une unité de stockage à son utilisation dans une configuration DAS (et|ou) SAN.

> **Avertissement**
>
> Les copies d'écran utilisées dans les réponses correspondent à l'utilisation de machines virtuelles. Les unités de disques apparaissent donc sous le nom `/dev/vd[a-z]`. Les unités de disques physiques d'un système réel apparaissent sous le nom `/dev/sd[a-z]`.

4.1. Destruction de la table des partitions

Sachant que les disques des postes de travaux pratiques sont utilisés régulièrement, il est préférable de se placer dans le contexte d'utilisation d'une unité de disque vierge de tout système de fichiers.

Q1. Quelle est la syntaxe d'appel de l'outil parted qui permet de visualiser la table de partition d'une unité de disque ?

Consulter la documentation de parted à l'adresse *Using Parted*.

```
# parted /dev/vda print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 34,4GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  256MB   255MB   primary   ext2         boot
 2      257MB   34,4GB  34,1GB  extended
 5      257MB   34,4GB  34,1GB  logical               lvm
```

Q2. Quelle est la syntaxe de la commande **dd** qui permet d'effacer complètement la table des partitions d'une unité de disque ?

Utiliser l'aide en ligne de la commande : **`dd --help`**.

La commande suivante écrit des 0 dans les 4 premiers blocs de 512 octets de l'unité de disque.

```
# dd if=/dev/zero of=/dev/vdb bs=512 count=4
4+0 enregistrements lus
4+0 enregistrements écrits
2048 octets (2,0 kB) copiés, 0,00135867 s, 1,5 MB/s

# parted  /dev/vdb print
Error: /dev/vdb: unrecognised disk label
```

4.2. Création de la table des partitions et formatage

Une fois que l'on dispose d'une unité de disque vierge, on peut passer à l'étape de création de la table des partitions. Dans le contexte de ces travaux pratiques, cette opération doit être effectuée deux fois sur les postes de travail pour les deux types d'unité de stockage utilisées.

1. Le second disque physique des postes de travail est destiné à intégrer l'unité logique RAID1.

2. Le disque réseau iSCSI est disponible une fois que la configuration du rôle *initiator* est active.

Cette manipulation est l'opération de plus bas niveau qui caractérise un accès réseau au stockage en *mode bloc* et non en mode fichier.

Q3. Quelles sont les instructions de l'outil parted qui permettent de créer une partition primaire unique couvrant la totalité de l'espace de stockage de l'unité de disque ?

Consulter la documentation de parted à l'adresse *Using Parted*.

```
# parted /dev/vdb
GNU Parted 2.3
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Error: /dev/vdb: unrecognised disk label
(parted) mklabel gpt
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 85,9GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start  End  Size  File system  Name  Flags

(parted) mkpart ext4 1 -1
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 85,9GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  85,9GB  85,9GB  ext4         ext4

(parted) quit
```

Q4. Quelle est la commande à utiliser pour les opérations de formatage ? Quel est le rôle de l'option `-T` de cette commande ?

Les informations utiles sont disponibles à la page *Ext4 Howto*. Les pages de manuels détaillent les fonctions des options.

La commande utilisée pour le formatage d'un système de fichiers `ext4`.

```
# dpkg -S `which mkfs.ext4`
e2fsprogs: /sbin/mkfs.ext4
```

L'option `-T` définit le type d'utilisation du système de fichiers à formater suivant sa taille. Les paramètres par défaut sont les suivants :

- `floppy` : 0 < taille < 3Mo
- `small` : 3Mo < taille < 512Mo
- `default` : 512Mo < taille < 4To
- `big` : 4To < taille < 16To
- `huge` : 16To < taille

Q5. Quelle est la syntaxe de la commande de formatage de la partition créée lors de l'étape précédente ?

Des exemples de syntaxe sont disponibles à la page *Ext4 Howto*.

```
# mkfs.ext4 /dev/vdb1
mke2fs 1.42.5 (29-Jul-2012)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
« Stride » = 0 blocs, « Stripe width » = 0 blocs
5242880 i-noeuds, 20971008 blocs
1048550 blocs (5.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=4294967296
640 groupes de blocs
32768 blocs par groupe, 32768 fragments par groupe
8192 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000

Allocation des tables de groupe : complété
Écriture des tables d'i-noeuds : complété
Création du journal (32768 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété
```

Q6. Quelle est la syntaxe de la commande de visualisation des attributs du système de fichiers créé lors du formatage ?

Les informations utiles sur les attributs sont fournies à la page *Ext4 Howto*.

```
# tune2fs -l /dev/vdb1
tune2fs 1.42.13 (17-May-2015)
Filesystem volume name:   <none>
Last mounted on:          /var/lib/iscsi-target
Filesystem UUID:          3ccc9115-f206-4a05-86f3-5902ac49b82d
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index \
                          filetype needs_recovery extent flex_bg sparse_super \
     large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              4718592
Block count:              18873856
Reserved block count:     943692
Free blocks:              18531693
Free inodes:              4718581
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      1019
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Sun Aug 23 17:20:15 2015
Last mount time:          Sun Aug 23 17:36:50 2015
Last write time:          Sun Aug 23 17:36:50 2015
Mount count:              2
Maximum mount count:      -1
Last checked:             Sun Aug 23 17:20:15 2015
Check interval:           0 (<none>)
Lifetime writes:          1284 MB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     28
Desired extra isize:      28
Journal inode:            8
Default directory hash:   half_md4
Directory Hash Seed:      8edb7b6a-5586-4d32-a02a-de231242f353
Journal backup:           inode blocks
```

4.3. Montage manuel d'un volume de stockage

Une fois qu'un volume de stockage a été partitionné et formaté, il faut le *monter* dans l'arborescence du système de fichiers de la machine de façon à lire et écrire des données dedans.

Q7. Comment obtenir l'identifiant du volume de stockage à ajouter au système de fichiers ?

Consulter la liste des utilitaires fournis avec le paquet util-linux. Il faut se rappeler que la représentation fichier d'un périphérique de stockage se distingue par son mode d'accès : le mode bloc.

La commande à utiliser est **blkid**. Dans l'exemple de la partition `/dev/vdb1`, on obtient le résultat suivant.

```
# blkid /dev/vdb1
/dev/vdb1: UUID="224f6aad-16c0-4923-8949-0628a8e10228" TYPE="ext4"
```

C'est cet identifiant que l'on doit utiliser pour compléter la configuration système et rendre le montage du périphérique de stockage permanent.

Q8. Dans quel fichier de configuration trouve-t-on la liste des périphériques montés lors de l'initialisation du système ?

Consulter la liste des fichiers du paquet util-linux.

Le fichier recherché est `/etc/fstab`. Il contient bien la liste des points de montage. Dans l'exemple ci-dessous, la racine et la partition d'échange utilisée en cas de saturation des ressources RAM du système.

```
# # grep -v '^#' /etc/fstab
UUID=78cc7982-fa2e-4498-af98-45272c7726c9 /    ext4    errors=remount-ro 0       1
UUID=56c21189-6cb3-4a6e-a6f6-ccf3e28db8b0 none swap    sw              0       0
```

Q9. Quelle est la commande qui donne la liste des montages en cours d'utilisation sur le système ? Quelle est l'option qui permet de scruter les entrées du fichier recherché dans la question précédente et de monter tous les points non encore utilisés ?

La commande est fournie par le paquet du même nom.

Le paquet mount fournit la commande du même nom. Cette commande liste tous les montages actifs du système. La liste comprend les systèmes de fichiers virtuels qui représentent l'état courant des paramètres du noyau ainsi que les systèmes de fichiers physiques qui correspondent aux volumes de stockage effectifs. En reprenant l'exemple utilisé auparavant et en filtrant les systèmes de fichiers virtuels, on obtient :

```
# mount | grep "/dev/vd"
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
```

L'option de montage des entrées inutilisées du fichier `/etc/fstab` est `-a`. Elle doit être utilisée dans la question suivante.

Q10. Comment compléter la configuration système pour assurer le montage du nouveau périphérique ?

Il faut utiliser les réponses aux questions précédentes pour valider le nouveau point de montage.

- Création du point de montage dans l'arborescence du système

```
# mkdir /var/lib/iscsi-target
```

- Ajout d'une ligne dans le fichier `/etc/fstab` avec l'identifiant du périphérique à ajouter

```
# echo UUID=3ccc9115-f206-4a05-86f3-5902ac49b82d \
  /var/lib/iscsi-target/ \
  ext4 \
  defaults \
  0      2 >>/etc/fstab
```

- Montage du nouveau périphérique

```
# mount -a
```

- Nouvelle liste des montages actifs

```
# mount | grep "/dev/vd"
/dev/vda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/vdb1 on /var/lib/iscsi-target type ext4 (rw,relatime,data=ordered)
```

5. Configuration du système initiator

Dans cette partie, on prépare le système auquel on a attribué le rôle *initiator*. Ce système est celui qui utilise le volume de stockage mis à disposition sur le réseau par le rôle *target*.

5.1. Sélection du paquet et lancement du service

Q11. Comment identifier et installer le paquet correspondant au rôle *initiator* ?

En effectuant une recherche simple dans le catalogue des paquets disponibles, on obtient la liste des paquets dont le nom contient la chaîne de caractères `iscsi`.

```
# aptitude search iscsi
p   iscsitarget         - iSCSI Enterprise Target userland tools
p   iscsitarget-dkms    - iSCSI Enterprise Target kernel module source - dkms version
p   iscsitarget-source  - iSCSI Enterprise Target kernel module source
p   open-iscsi          - High performance, transport independent iSCSI implementation
```

On remarque que le paquet `open-iscsi` est le seul qui ne soit pas identifié comme appartenant à la catégorie *target*.

```
# aptitude install open-iscsi
```

Q12. Comment lancer le service *initiator* et valider son fonctionnement ?

À partir de la liste des fichiers du paquet on peut identifier les éléments de démarrage et de configuration du service.

```
# dpkg -L open-iscsi
/.
/etc
/etc/init.d
/etc/init.d/open-iscsi
/etc/init.d/umountiscsi.sh
/etc/network
/etc/network/if-up.d
/etc/default
/etc/default/open-iscsi
/etc/iscsi
/etc/iscsi/iscsid.conf
/etc/iscsi/initiatorname.iscsi
/usr
/usr/share
/usr/share/man
/usr/share/man/man8
/usr/share/man/man8/iscsiadm.8.gz
/usr/share/man/man8/iscsid.8.gz
/usr/share/man/man8/iscsi_discovery.8.gz
/usr/share/man/man8/iscsistart.8.gz
/usr/share/man/man8/iscsi-iname.8.gz
/usr/share/initramfs-tools
/usr/share/initramfs-tools/hooks
/usr/share/initramfs-tools/hooks/iscsi
/usr/share/initramfs-tools/scripts
/usr/share/initramfs-tools/scripts/local-top
/usr/share/initramfs-tools/scripts/local-top/iscsi
/usr/share/doc
/usr/share/doc/open-iscsi
/usr/share/doc/open-iscsi/copyright
/usr/share/doc/open-iscsi/README.Debian.gz
/usr/share/doc/open-iscsi/README.gz
/usr/share/doc/open-iscsi/changelog.Debian.gz
/usr/share/doc/open-iscsi/changelog.gz
/usr/sbin
/usr/sbin/iscsi-iname
/usr/sbin/iscsid
/usr/sbin/iscsi_discovery
/usr/sbin/iscsistart
/usr/bin
/usr/bin/iscsiadm
/var
/var/lib
/var/lib/open-iscsi
```

Le lancement du service se fait de façon classique à partir de l'arborescence des scripts des niveaux de démarrage (*runlevels*).

```
# /etc/init.d/open-iscsi restart
```

La même opération avec systemd utilise la syntaxe suivante :

```
# systemctl restart open-iscsi
```

On peut ensuite consulter les journaux système pour valider l'initialisation du ou des démons.

```
# grep -i iscsi /var/log/syslog
Loading iSCSI transport class v2.0-870.
iscsi: registered transport (tcp)
iscsi: registered transport (iser)
```

On retrouve les informations correspondantes aux messages ci-dessus dans la liste des processus actifs.

```
# ps aux | grep -i iscsi
root      3479  0.0  0.0      0     0 ?        S<   16:28   0:00 [iscsi_eh]
root      3487  0.0  0.0   4980   472 ?        Ss   16:28   0:00 /usr/sbin/iscsid
root      3488  0.0  0.6   5480  3280 ?        S<Ls 16:28   0:00 /usr/sbin/iscsid
```

5.2. Tests de fonctionnement du service

Q13. Quelle est la commande principale du rôle *initiator* qui permet de tester la connectivité iSCSI ?

Consulter la liste des fichiers du paquet `open-iscsi`.

En consultant la liste donnée ci-dessus, on ne relève qu'un seul outil exécutable : la commande **iscsiadm**.

Q14. Quelles sont les options de découverte proposées avec cette commande ? Donner un exemple fournissant l'identifiant de l'unité de stockage réseau visible.

Consulter les pages de manuels de la commande identifiée dans la question précédente.

À partir du poste *initator* numéro 1, le seul volume de stockage visible est :

```
# iscsiadm -m discovery --type sendtargets --portal=192.0.2.12:3260
192.0.2.12:3260,1 iqn.2003-01.org.linux-iscsi.vm0.i686:sn.1b49d1997f22
[2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.vm0.i686:sn.1b49d1997f22
192.0.2.12:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566
[2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.63701809056
```

Q15. Quel est l'identifiant à communiquer ou à paramétrer pour le système *initiator* soit reconnu côté système *target* ?

Rechercher les informations relatives au nommage iSCSI dans les outils et les fichiers fournis avec le paquet de gestion du rôle *initiator*.

Le filtrage de la liste des fichiers fournis avec le paquet open-iscsi donne le résultat suivant.

```
# dpkg -L open-iscsi | grep name
/etc/iscsi/initiatorname.iscsi
/usr/share/man/man8/iscsi-iname.8.gz
/usr/sbin/iscsi-iname
```

- Le fichier `/etc/iscsi/initiatorname.iscsi` contient l'identifiant du système à communiquer au système *target* pour que celui-ci l'associe dans la rubrique des listes de contrôle d'accès : `acls`.

  Dans le contexte de ces travaux pratiques, on se contente de relever l'identifiant généré automatiquement lors de l'installation du paquet et de l'implanter dans la liste de contrôle d'accès créée avec targetcli sur le système *target*.

  Côté *initiator*, on lit l'identifiant iqn

  ```
  # grep -v ^# /etc/iscsi/initiatorname.iscsi
  InitiatorName=iqn.1993-08.org.debian:01:9d11913c78ac
  ```

  Côté *target*, on créé la liste de contrôle d'accès qui associe l'unité logique SCSI au système *initiator*.

  ```
  /iscsi/iqn.20...18090566/tpg1> ls
  o- tpg1 ..................................................................... [enabled]
    o- acls ....................................................................... [1 ACL]
    | o- iqn.1993-08.org.debian:01:9d11913c78ac ..................... [1 Mapped LUN]
    |   o- mapped_lun0 ............................................... [lun0 (rw)]
    o- luns ....................................................................... [1 LUN]
    | o- lun0 ..................................... [iblock/initiator1 (/dev/vdb)]
    o- portals ................................................... [2 Portals]
      o- 192.0.2.12:3260 ..................................... [OK, iser disabled]
      o- 2001:db8:feb2:2:b8ad:ff:feca:fe00:3260 ............... [OK, iser disabled]
  ```

- La commande **iscsi-iname** sert à générer un nouvel identifiant conforme au forat iqn. Elle permet de fournir un nouvel identifiant compatible avec la nomenclature de l'infrastructure de stockage d'un opérateur.

Q16. Quelles sont les options de connexion proposées avec cette même commande ? Donner un exemple illustrant l'établissement d'une connexion.

Consulter les pages de manuels de la commande identifiée précédemment.

```
# iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.target.i686:sn.1b49d1997f22 -p 192.0.2.12 -l
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.1b49d1997f22, p
iscsiadm: Could not login to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization failure)
iscsiadm: Could not log into all portals
```

Dans l'exemple ci-dessus, la connexion sans authentification a échoué faute d'autorisations côté rôle *target*. Comme nous sommes dans un contexte de travaux pratiques, il faut paramétrer deux attributs spécifiques : `authentication=0` et `demo_mode_write_protect=0`.

```
# iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566 -p 192.0.2.12 -l
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566, p
Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566, portal
```

La connexion est maintenant établie et le volume de stockage réseau est disponible sous forme d'unité logique SCSI.

Q17. Comment obtenir les caractéristiques de l'unité de stockage iSCSI utilisée ?

Consulter les journaux système.

Voici un extrait du fichier `/var/log/syslog`.

```
# egrep '(sd|scsi)' /var/log/syslog
initiator1 kernel: [18614.778860] scsi host8: iSCSI Initiator over TCP/IP
initiator1 kernel: [18615.035238] scsi 8:0:0:0: Direct-Access     LIO-ORG  IBLOCK          4.0
initiator1 kernel: [18615.038358] scsi 8:0:0:0: Attached scsi generic sg1 type 0
initiator1 kernel: [18615.067546] sd 8:0:0:0: [sda] 150994944 512-byte logical blocks: (77.3 GB/7
initiator1 kernel: [18615.070885] sd 8:0:0:0: [sda] Write Protect is off
initiator1 kernel: [18615.070893] sd 8:0:0:0: [sda] Mode Sense: 43 00 10 08
initiator1 kernel: [18615.071854] sd 8:0:0:0: [sda] Write cache: enabled, read cache: enabled, su
initiator1 kernel: [18615.117019]  sda:
initiator1 kernel: [18615.120840] sd 8:0:0:0: [sda] Attached SCSI disk
initiator1 iscsid: Connection3:0 to [target: iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090
                                 portal: 192.0.2.12,3260] through [iface: default] is operational
```

Q18. Donner la liste des entrées de périphériques de stockage créées par le démon `udev` ?

Lister les entrées de périphériques mode bloc de l'arborescence système.

Les fichiers de description des périphériques mode bloc sont tous situés dans le répertoire `/dev/`. En reprenant l'exemple ci-dessus, on obtient :

```
# ls -lA /dev/[v,s]d*
brw-rw---- 1 root disk   8,  0 sept.  1 14:38 /dev/sda
brw-rw---- 1 root disk 254,  0 sept.  1 11:28 /dev/vda
brw-rw---- 1 root disk 254,  1 sept.  1 11:28 /dev/vda1
brw-rw---- 1 root disk 254,  2 sept.  1 11:28 /dev/vda2
brw-rw---- 1 root disk 254,  5 sept.  1 11:28 /dev/vda5
brw-rw---- 1 root disk 254, 16 sept.  1 11:28 /dev/vdb
```

L'entrée `/dev/sda` correspond à l'unité de disque iSCSI. Le volume de stockage est donc bien vu de façon transparente comme un périphérique local au système accessible en mode bloc. Il entre bien dans la catégorie SAN ou *Storage Area Network*.

5.3. Réinitialisation de session iSCSI

Dans le cas d'une reconfiguration avec un autre hôte *target* ou dans le cas d'un dépannage, il est utile de pouvoir reprendre les paramètres du rôle *initiator*.

Q19. Comment obtenir la liste des sessions actives avec le système *target* ?

Consulter les pages de manuels de la commande de configuration du rôle *initiator* : **iscsiadm**.

C'est le mode `session`, documenté dans les pages de manuels de la commande **iscsiadm**, qui permet de répondre à la question.

```
# iscsiadm -m session
tcp: [3] 192.0.2.12:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566 (non-flash)
tcp: [4] [2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.63
```

Q20. Comment libérer toutes les sessions actives depuis le système *initiator* ?

Consulter les pages de manuels de la commande de configuration du rôle *initiator* : **iscsiadm**.

Pour cette question, c'est le mode `node` qui nous intéresse.

```
# iscsiadm -m node -U all
Logging out of session [sid: 3, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566,
Logout of [sid: 3, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566, portal: 2001:
```

Q21. Comment effacer les informations sur les systèmes *target* déjà découverts en cas de problème de configuration ?

Consulter les pages de manuels de la commande de configuration du rôle *initiator* : **iscsiadm**.

Toutes les manipulations sur les systèmes *target* découverts dépendent du mode `discovery` et l'opération à utiliser est `delete`.

```
# iscsiadm -m discovery -p 192.0.2.12 -o delete
```

Il suffit ensuite de reprendre la découverte décrite à la question Q : Q14.

5.4. Configuration système permanente

Une fois la connexion à la ressource iSCSI testée, on peut passer à la configuration système de façon à retrouver le volume de stockage après une réinitialisation du système *initiator*.

Q22. Comment rendre la connexion à l'unité de stockage automatique lors de l'initialisation du système *initiator* ?

Rechercher dans la liste des fichiers du paquet open-iscsi les éléments relatifs à la configuration système. Éditer le fichier de configuration principal de façon à rendre automatique le lancement du service.

Au niveau système, les fichiers de configuration sont nécessairement dans le répertoire `/etc/`.

```
# dpkg -L open-iscsi | grep '/etc/'
/etc/default
/etc/default/open-iscsi
/etc/init.d
/etc/init.d/open-iscsi
/etc/init.d/umountiscsi.sh
/etc/iscsi
/etc/iscsi/iscsid.conf
/etc/iscsi/initiatorname.iscsi
/etc/network
/etc/network/if-up.d
```

Le fichier `/etc/iscsi/iscsid.conf` contient une directive dans la section *Startup settings* qui rend automatique l'accès à une ressource déjà enregistrée. Voici le contenu de cette section extraite du fichier de configuration.

```
#*****************
# Startup settings
#*****************

# To request that the iscsi initd scripts startup a session set to "automatic".
node.startup = automatic
```

Q23. Comment connaître l'état et la liste d'une session iSCSI active ?

Consulter les pages de manuels de la commande de configuration du rôle *initiator* : **iscsiadm**.

Il existe un mode `session` dédié aux manipulations sur les sessions. La commande de test la plus simple est la suivante.

```
# iscsiadm -m session
tcp: [3] 192.0.2.12:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.637018090566 (non-flash)
tcp: [4] [2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.63
```

La copie d'écran ci-dessus indique deux sessions actives. Si la liste est vide, il n'y a pas de session iSCSI en cours.

Il est possible d'obtenir davantage d'informations sur les paramètres de session en cours à l'aide de l'option `-P` suivie d'un numéro désignant le niveau de détail attendu. Par exemple, la commande `iscsiadm -m session -P 3` affiche les paramètres sur les interfaces réseau utilisées, etc.

Q24. Comment retrouver un point de montage unique du volume de stockage iSCSI après réinitialisation du système *initiator* ?

Créer un répertoire de montage et rechercher les options utiles dans les pages de manuels des commandes **mount**, **systemd.mount** et **blkid**. Éditer le fichier `/etc/fstab` en utilisant les options sélectionnées. Noter que le fichier `fstab` possède ses propres pages de manuels.

La création du répertoire destiné au montage du volume de stockage iSCSI ne pose pas de problème.

```
# mkdir mkdir /var/cache/iscsi-storage
```

La commande **blkid** permet d'obtenir l'identifiant unique du volume de stockage. Dans la copie d'écran ci-dessous, la partition `/dev/sda1` correspond au résultat de l'établissement de la session iSCSI et l'identification du système de fichiers utilisé (`btrfs`) correspond au résultat du formatage de la partition. Ce ne sont que des exemples particuliers au contexte de la maquette utilisée.

```
# blkid /dev/sda1
/dev/sda1: UUID="11924824-00f1-4735-bd30-4bacaa3cbde0" UUID_SUB="09505fb1-d90c-4d05-b9e7-b4a0454a
           TYPE="btrfs" PARTLABEL="iSCSI-LUN0" PARTUUID="0da2be8f-ff7b-40d1-a720-15f08c456351"
```

Le choix des options à utiliser lors de l'édition du fichier `/etc/fstab` constitue le point délicat de cette question.

```
UUID=11924824-00f1-4735-bd30-4bacaa3cbde0       /var/cache/iscsi-storage        btrfs   noauto,x-
```

- Le choix de la valeur `UUID` se fait à partir du résultat de la commande **blkid** donné ci-dessus.
- Le point de montage `/var/cache/iscsi-storage` a lui aussi été défini ci-dessus.
- Le système de fichiers utilisé est, là encore, connu : `btrfs`.

Les trois paramètres suivants sont spécifiques au contexte iSCSI.

- L'option `noauto` empêche le déclenchement du montage lors de la scrutation du fichier `/etc/fstab`. Les entrées présentes dans ce fichier doivent être disponibles très tôt dans le processus d'initialisation des services du système. Or, un volume de stockage réseau iSCSI n'est pas nécessairement disponible au moment du parcours des entrées en question.
- L'option `x-systemd.automount` provoque la création d'une unité d'automontage (au sens systemd). Le principe de l'automontage veut que l'opération de montage soit effective au moment du parcours de l'arborescence `/var/cache/iscsi-storage` par un utilisateur ou une application. Autrement dit, tant que l'arborescence n'est pas utilisée, le montage n'est pas réalisé et l'initialisation du système *initiator* se poursuit normalement.
- L'option `_netdev` spécifie que le système de fichiers réside sur un périphérique nécessitant des accès réseau. Il est donc inutile d'y accéder tant qu'aucune interface réseau n'est active.

6. Configuration du système target

Dans cette partie, on prépare le système auquel on a attribué le rôle *target* à l'aide de l'outil targetcli.

6.1. Installation de l'outil de paramétrage du rôle target

Q25. Quel est le paquet qui contient les éléments de configuration du service dans l'espace utilisateur ?

On consulte le site de référence à l'adresse *Linux-IO : the Linux SCSI Target wiki* pour identifier l'outil principal et on effectue ensuite une recherche dans la liste des paquets.

```
# aptitude search targetcli
i   targetcli          - administration tool for managing LIO core target
```

Q26. Comment installer le paquet identifié à la question précédente avec le minimum de dépendances (ou sans les paquets recommandés associés) ?

Consulter les pages de manuels de la commande **aptitude** et rechercher l'option qui évite l'installation des paquets recommandés.

L'option recherchée est `-R` ou `--without-recommends`.

```
# aptitude install -R targetcli
Les NOUVEAUX paquets suivants vont être installés :
  python-configobj{a} python-configshell{a} python-epydoc{a} python-ipaddr{a}
  python-netifaces{a} python-prettytable{a} python-pyparsing{a}
  python-rtslib{a} python-simpleparse{a} python-simpleparse-mxtexttools{a}
  python-six{a} python-urwid{a} targetcli
Les paquets suivants sont RECOMMANDÉS mais ne seront pas installés :
  graphviz python-docutils python-tk
0 paquets mis à jour, 13 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de télécharger 1 964 ko d'archives. Après dépaquetage,
8 051 ko seront utilisés.
Voulez-vous continuer ? [Y/n/?]
```

6.2. Configuration du rôle target

La technologie iSCSI dispose d'un schéma de nommage propre défini dans le document standard RFC3721 *Internet Small Computer Systems Interface (iSCSI) Naming and Discovery*. Le format retenu ici est baptisé iqn (*iSCSI Qualified Name*). Il s'agit d'une chaîne qui débute par "iqn." suivie d'une date au format AAAA-MM, du nom de l'autorité qui a attribué le nom (le nom de domaine à l'envers), puis une autre chaîne unique qui identifie le nœud de stockage.

On a choisi de n'utiliser aucun mécanisme d'authentification sachant que la configuration se fait dans un contexte de travaux pratiques et non d'exploitation sur un réseau réel.

Q27. Quelles sont les étapes à suivre pour publier un volume de stockage sur le réseau à partir de l'interface de l'outil targetcli ?

Ici aussi, il faut consulter le site de référence à l'adresse *Linux-IO : the Linux SCSI Target wiki* pour identifier les différentes étapes.

La copie d'écran ci-dessous liste les opérations disponibles.

```
# targetcli
targetcli 3.0.pre4.1~ga55d018 (rtslib 3.0.pre4.1~g1b33ceb)
Copyright (c) 2011-2014 by Datera, Inc.
All rights reserved.

/> ls
o- / ..................................................................... [...]
  o- backstores ......................................................... [...]
  | o- fileio ..................................... [0 Storage Object]
  | o- iblock ..................................... [0 Storage Object]
  | o- pscsi ...................................... [0 Storage Object]
  | o- rd_mcp ..................................... [0 Storage Object]
  o- ib_srpt ........................................................... [0 Targets]
  o- iscsi ............................................................. [0 Targets]
  o- loopback .......................................................... [0 Targets]
  o- qla2xxx ........................................................... [0 Targets]
  o- tcm_fc ............................................................ [0 Targets]
  o- vhost ............................................................. [0 Targets]
```

- La section *backstores* désigne les volumes de stockage à publier sur le réseau. Ici, les deux items intéressants sont *fileio* et *iblock*. Le premier fait correspondre un fichier du système local au volume à publier. Le second fait correspondre une unité de disque physique au volume à publier.
- La section *iscsi* sert à définir une «cible» (*target*) qui comprend au moins une unité logique (LUN en vocabulaire SCSI) et un point de contact réseau.

Partie stockage local ***backstores***

Q28. Quelles sont les opérations à effectuer définir un disque physique comme volume de stockage ?

Il faut consulter le site de référence et repérer les options du menu `iblock`.

La création du volume se fait à l'aide de la commande ci-dessous.

```
/backstores/iblock> create initiator1 /dev/vdb
Created iblock storage object initiator1 using /dev/vdb.
/backstores/iblock> ls /
o- / ..................................................... [...]
  o- backstores .......................................... [...]
  | o- fileio ............................. [0 Storage Object]
  | o- iblock ............................. [1 Storage Object]
  | | o- initiator1 ................... [/dev/vdb, not in use]
  | o- pscsi .............................. [0 Storage Object]
  | o- rd_mcp ............................. [0 Storage Object]
  o- ib_srpt ..................................... [0 Targets]
  o- iscsi ....................................... [0 Targets]
  o- loopback .................................... [0 Targets]
  o- qla2xxx ..................................... [0 Targets]
  o- tcm_fc ...................................... [0 Targets]
  o- vhost ....................................... [0 Targets]
```

Q29. Quelles sont les opérations à effectuer pour définir un fichier comme volume de stockage ?

Il faut consulter le site de référence et repérer les options du menu `fileio`.

La création du volume se fait à l'aide de la commande ci-dessous.

```
/backstores/fileio> create storage_file /var/lib/iscsi-target/storage-for-myinitiator 32G
Using buffered mode.
Created fileio storage_file.
/backstores/fileio> ls
o- fileio ..................................................................... [2 Storage Objects]
  o- initiator2 ................ [32.0G, /var/lib/iscsi-target/initiator2.disk, not in use]
  o- storage_file ....... [32.0G, /var/lib/iscsi-target/storage-for-myinitiator, not in use]
```

Dans l'exemple ci-dessus on a créé un nouvel objet dans le dépôt des volumes de stockage appelé `storage_file`. Dans la même commande on lui a attribué une capacité de 32Go. On a aussi précisé le chemin d'accès à ce fichier.

Il faut noter que l'espace effectivement occupé par le fichier `/var/lib/iscsi-target/storage-for-myinitiator` correspond à celui utilisé côté *initiator*. La commande de l'exemple ci-dessus a provoqué la création d'un fichier vide.

Partie iSCSI

Q30. Quelles sont les opérations à effectuer pour définir une nouvelle cible iSCSI ?

Il faut consulter le site de référence et repérer les options du menu `iscsi`. Attention ! Une cible iSCSI comprend plusieurs attributs.

1. **Nommage de la cible au format iqn.**
   Si le nom de la cible n'est pas fourni avec la commande **create**, il est généré automatiquement.

   ```
   /> cd iscsi/
   /iscsi> create
   Created target iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2e.
   Selected TPG Tag 1.
   Created TPG 1.
   ```

C'est après cette première opération que les attributs apparaissent pour la nouvelle cible.

```
/iscsi> ls
o- iscsi ..................................................... [1 Target]
  o- iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2e ...... [1 TPG]
    o- tpg1 ....................................................... [enabled]
      o- acls ..................................................... [0 ACLs]
      o- luns ..................................................... [0 LUNs]
      o- portals ................................................. [0 Portals]
```

2. **Affectation de l'unité logique à la cible iSCSI.**
Les numéros d'unités logiques SCSI ou LUNs sont affectés automatiquement. Ici, l'unité `lun0` correspond à la première association faite depuis le dépôt des volumes de stockage.

```
/iscsi> cd iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2e/tpg1/
/iscsi/iqn.20...6efd0f2e/tpg1> luns/ create /backstores/iblock/initiator1
Selected LUN 0.
Created LUN 0.
/iscsi/iqn.20...6efd0f2e/tpg1> ls
o- tpg1 ..................................... [enabled]
  o- acls ................................... [0 ACLs]
  o- luns ................................... [1 LUN]
  | o- lun0 .................. [iblock/initiator1 (/dev/vdb)]
  o- portals ................................. [0 Portals]
```

3. **Ouverture du point de contact réseau pour cette cible iSCSI.**
Un même point de contact peut être en écoute sur plusieurs adresses IP. Dans l'exemple ci-dessous on ouvre une configuration double pile IPv4 et IPv6.

```
/iscsi/iqn.20...6efd0f2e/tpg1> portals/ create 192.0.2.12 3260
Created network portal 192.0.2.12:3260.
/iscsi/iqn.20...6efd0f2e/tpg1> portals/ create 2001:db8:feb2:2:b8ad:ff:feca:fe00 3260
Created network portal 2001:db8:feb2:2:b8ad:ff:feca:fe00:3260.
/iscsi/iqn.20.../tpg1/portals> ls
o- portals ............................................... [2 Portals]
  o- 192.0.2.12:3260 ............................... [OK, iser disabled]
  o- 2001:db8:feb2:2:b8ad:ff:feca:fe00:3260 ........ [OK, iser disabled]
```

On peut sortir de l'outil targetcli pour vérifier que le service réseau est bien accessible.

```
# ss -tan
State       Recv-Q Send-Q                          Local Address:Port  Peer Address:Port
LISTEN      0      128                                         *:22               *:*
LISTEN      0      20                                      127.0.0.1:25               *:*
LISTEN      0      256                                  192.0.2.12:3260            *:*
LISTEN      0      128                                         :::22              :::*
LISTEN      0      20                                        ::1:25               :::*
LISTEN      0      256       2001:db8:feb2:2:b8ad:ff:feca:fe00:3260              :::*
```

Enfin, on peut aussi vérifier que le service est ouvert *côté initiator* à l'aide de la fonction de découverte.

```
root@initiator1:~# iscsiadm -m discovery --type sendtargets --portal=192.0.2.12:3260
192.0.2.12:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2e
[2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156ef
```

4. **Création d'une liste de contrôle d'accès.**
Même si le service réseau et la fonction découverte sont ouverts, le volume de stockage réseau n'est pas encore accessible. La connexion depuis l'hôte *initiator* échoue et on obtient le message suivant.

```
root@initiator1:~# iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2
iscsiadm: Could not login to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization failure)
iscsiadm: Could not log into all portals
```

Côté hôte *target*, les journaux système font apparaître un message du type suivant.

```
iSCSI Initiator Node: iqn.1993-08.org.debian:01:9d11913c78ac is not authorized to access iSCS
iSCSI Login negotiation failed.
```

Il est donc necessaire d'autoriser l'accès depuis l'hôte *initiator*. Dans l'outil targetcli, on configure l'attribut `acls` de la cible iSCSI.

```
/iscsi/iqn.20...6efd0f2e/tpg1> acls/ create iqn.1993-08.org.debian:01:9d11913c78ac
Created Node ACL for iqn.1993-08.org.debian:01:9d11913c78ac
Created mapped LUN 0.
/iscsi/iqn.20...6efd0f2e/tpg1> ls
o- tpg1 ..................................................... [enabled]
  o- acls ....................................................... [1 ACL]
  | o- iqn.1993-08.org.debian:01:9d11913c78ac ........... [1 Mapped LUN]
  |   o- mapped_lun0 ..................................... [lun0 (rw)]
  o- luns ....................................................... [1 LUN]
  | o- lun0 ............................ [iblock/initiator1 (/dev/vdb)]
  o- portals ................................................. [2 Portals]
    o- 192.0.2.12:3260 ............................. [OK, iser disabled]
    o- 2001:db8:feb2:2:b8ad:ff:feca:fe00:3260 ...... [OK, iser disabled]
```

Ce n'est pas terminé ! Par défaut, une cible iSCSI n'est accessible qu'après authentification. Il est donc nécessaire de désactiver cette authentification pour tester l'accès depuis l'hôte *initiator*.

```
/iscsi/iqn.20...6efd0f2e/tpg1> set attribute authentication=0 demo_mode_write_protect=0
Parameter authentication is now '0'.
Parameter demo_mode_write_protect is now '0'.
```

Finalement, le volume de stockage est mis à disposition de l'hôte *initiator*.

```
root@initiator1:~# iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2
Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.bf156efd0f2e, po
```

7. Configuration de l'authentification CHAP

Dans cette partie, on suppose que tous les tests précédents ont été effectués avec succès et que les échanges entre les systèmes *target* et *initiator* sont validés.

On s'intéresse maintenant à l'authentification entre ces mêmes systèmes. Pour traiter les questions suivantes, une nouvelle entrée a été utilisée pour le rôle *target*.

Le mécanisme d'authentification le plus communément utilisé dans le déploiement des connexions iSCSI s'appuie sur CHAP (*Challenge-Handshake Authentication Protocol*). Il s'agit d'une méthode d'authentification entre deux hôtes pairs sans échange de mot de passe en clair sur le réseau. Cette méthode suppose que les deux hôtes utilisent le même mot de passe.

Q31. Comment régler les paramètres d'authentification CHAP sur le système *target* ?

Comme pour les étapes précédentes, toutes les manipulations se font à partir de l'outil targetcli. Il faut donc consulter la documentation de cet outil à l'adresse *Linux-IO : the Linux SCSI Target wiki*. Il existe une section *Mutual CHAP authentication*.

Partant d'une nouvelle configuration, on obtient la liste de paramètres suivante dans laquelle aucun contrôle d'accès n'a été défini.

```
/iscsi/iqn.20...57c35b07/tpg1> ls
o- tpg1 ..................................................... [enabled]
  o- acls ....................................................... [0 ACLs]
  o- luns ....................................................... [1 LUN]
  | o- lun0 ..................................... [iblock/initiator1 (/dev/vdb)]
  o- portals ................................................. [1 Portal]
    o- 2001:db8:feb2:2:b8ad:ff:feca:fe00:3260 ............... [OK, iser disabled]
```

On passe à la création d'une entrée de contrôle d'accès basée sur l'identifiant `iqn` unique du système *initiator*.

```
/iscsi/iqn.20...57c35b07/tpg1> acls/ create iqn.2015-09.org.debian:01:9d11913c78ac
Created Node ACL for iqn.2015-09.org.debian:01:9d11913c78ac
Created mapped LUN 0.
/iscsi/iqn.20...57c35b07/tpg1> ls
o- tpg1 ..................................................... [enabled]
  o- acls ..................................................... [1 ACL]
  | o- iqn.2015-09.org.debian:01:9d11913c78ac ................ [1 Mapped LUN]
  |   o- mapped_lun0 ........................................ [lun0 (rw)]
  o- luns ..................................................... [1 LUN]
  | o- lun0 ................................ [iblock/initiator1 (/dev/vdb)]
  o- portals ................................................... [1 Portal]
    o- 2001:db8:feb2:2:b8ad:ff:feca:fe00:3260 ............. [OK, iser disabled]
```

On définit ensuite les paramètres d'authentification pour cette entrée. Comme la méthode CHAP est symétrique, on doit déposer de part et d'autre le secret. On fixe ici les paramètres `userid` et `password`.

```
/iscsi/iqn.20...57c35b07/tpg1> acls/iqn.2015-09.org.debian:01:9d11913c78ac/ set auth userid=initi
Parameter userid is now 'initiator-username'.
/iscsi/iqn.20...57c35b07/tpg1> acls/iqn.2015-09.org.debian:01:9d11913c78ac/ set auth password=ini
Parameter password is now 'initiator-53cr3t-p455w0rd'.
```

Q32. Comment régler les paramètres d'authentification CHAP sur le système *initiator* ?

Rechercher dans le fichier de configuration principal du rôle *initiator* les paramètres relatifs à l'authentification.

Le nom d'utilisateur et le mot de passe sont définis dans le fichier `/etc/iscsi/iscsid.conf` du système *initiator*.

```
# *************
# CHAP Settings
# *************

# To enable CHAP authentication set node.session.auth.authmethod
# to CHAP. The default is None.
node.session.auth.authmethod = CHAP

# To set a CHAP username and password for initiator
# authentication by the target(s), uncomment the following lines:
node.session.auth.username = SAN-lab-1stInitiator
node.session.auth.password = MyS4N-1stInitiator-53cr3t
```

Le même principe peut être appliqué au mécanisme de découverte en appliquant un couple *login/password* identique ou non à la suite de ce fichier de configuration.

Une fois la configuration en place, on obtient les résultats suivants lors de la validation.

- Découverte du nouveau volume réseau :

```
# iscsiadm -m discovery --type sendtargets --portal=[2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260
[2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.f58f71d5ba26
192.0.2.12:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.f58f71d5ba26
[2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.8b7457c35b07
```

- Connexion avec authentification CHAP :

```
# iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.target.i686:sn.8b7457c35b07 -p 2001:db8:feb2:2:b8ad
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.8b7457c35b07, porta
Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.i686:sn.8b7457c35b07, portal: 20
```

- Affichage de la session active :

```
# iscsiadm -m session
tcp: [4] [2001:db8:feb2:2:b8ad:ff:feca:fe00]:3260,1 iqn.2003-01.org.linux-iscsi.target.i686:sn.8b7457
```

8. Configuration d'une unité logique RAID1

Dans cette partie, on crée une unité logique RAID1 composée d'une unité de disque locale et d'une unité de disque iSCSI dans le but d'illustrer une solution de réplication synchrone. En effet, dans un volume RAID1

chaque disque contient à tout moment exactement les mêmes données. Ici, le contenu de l'unité de disque locale est identique à celui de l'unité de disque réseau. La réplication ainsi réalisée est dite synchrone puisque toute écriture locale est dupliquée sur le réseau de stockage iSCSI.

8.1. Sélection du paquet et création de l'unité de stockage

Q33. Quel est le paquet qui contient les outils de configuration et de gestion des différents types d'unités RAID logicielles ? Installer ce paquet et identifier l'outil d'administration de tableau RAID logiciel.

Effectuer une recherche dans les descriptions de paquets avec l'acronyme clé RAID.

```
# # aptitude search ~draid | grep administration
p   mdadm                           - outil d'administration d'ensembles RAID
```

Une fois le paquet identifié et installé, on peut lister son contenu et isoler les commandes utilisateur.

```
# dpkg -L mdadm | grep bin
/sbin
/sbin/mdmon
/sbin/mdadm-startall
/sbin/mdadm
```

Q34. Rechercher la syntaxe d'appel à l'outil identifié dans la question précédente pour créer l'unité logique RAID1 ? Exécuter cette commande.

Après s'être assuré qu'aucune table de partition n'existe sur les deux unités constituant le tableau, on obtient le résultat suivant.

```
# mdadm --create /dev/md0 --level=raid1 --raid-devices=2 /dev/sda /dev/vdb
mdadm: Note: this array has metadata at the start and
    may not be suitable as a boot device.  If you plan to
    store '/boot' on this device please ensure that
    your boot-loader understands md/v1.x metadata, or use
    --metadata=0.90
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

8.2. Manipulations sur l'unité de stockage RAID1

Q35. Comment connaître l'état de l'unité logique RAID1 ?

Effectuer une recherche dans le système de fichiers virtuel `/proc/`.

Exemple du tableau créé lors l'exécution de la commande de la question précédente.

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 vdb[1] sda[0]
      33537920 blocks super 1.2 [2/2] [UU]

unused devices: <none>
```

Q36. Comment afficher la liste des propriétés de l'unité logique RAID1 ?

Effectuer une recherche dans les options de la commande d'administration.

```
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sun Sep  9 17:06:34 2012
     Raid Level : raid1
     Array Size : 33537920 (31.98 GiB 34.34 GB)
  Used Dev Size : 33537920 (31.98 GiB 34.34 GB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent

    Update Time : Sun Sep  9 17:13:18 2012
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           Name : iSCSI-1StInitiator:0  (local to host iSCSI-1StInitiator)
           UUID : 01749969:3a764b9f:2749b4c4:5953b282
         Events : 17

    Number   Major   Minor   RaidDevice State
       0       8        0        0      active sync   /dev/sda
       1     254       16        1      active sync   /dev/vdb
```

Q37. Comment rendre la configuration du tableau RAID1 permanente au niveau système ?

Effectuer une recherche dans les options de la commande d'administration.

C'est le fichier `/etc/mdadm/mdadm.conf` qui contient les directives de configuration. On ajoute en fin de ce fichier la définition du tableau créé plus haut.

```
# mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

9. Configuration d'un volume logique de sauvegarde

L'objectif de cette partie est de créer un mécanisme de sauvegarde réseau automatisé en s'appuyant sur la notion de «prise de vue» ou *snapshot* proposée par le gestionnaire de volume logique LVM. Dans une prise de vue, on ne stocke que les différences relativement au volume logique original.

```
# pvcreate /dev/md0
  Writing physical volume data to disk "/dev/md0"
  Physical volume "/dev/md0" successfully created
```

```
# pvdisplay
  --- Physical volume ---
  PV Name               /dev/vda5
  VG Name               vm0
  PV Size               31,76 GiB / not usable 2,00 MiB
  Allocatable           yes (but full)
  PE Size               4,00 MiB
  Total PE              8130
  Free PE               0
  Allocated PE          8130
  PV UUID               CpaZ5D-vbVS-32w3-QLnk-GVAd-06pB-y2Iw8Y

  "/dev/md0" is a new physical volume of "31,98 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/md0
  VG Name
  PV Size               31,98 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               KAmRl0-ugMa-0eE3-ZJCc-Q2t0-lqeM-RB8Qxn
```

```
# vgextend vm0 /dev/md0
  Volume group "vm0" successfully extended
```

```
# vgdisplay
  --- Volume group ---
  VG Name               vm0
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               2
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               63,74 GiB
  PE Size               4,00 MiB
  Total PE              16317
  Alloc PE / Size       8130 / 31,76 GiB
  Free  PE / Size       8187 / 31,98 GiB
  VG UUID               dnw5zr-hPPU-L1FZ-P6Be-HL7E-FUNu-0OuosE
```

```
# lvcreate --name backup -L12G vm0
```

```
# lvcreate --snapshot --name LVM-snapshot-lab --extents +100%FREE /dev/vm0/root
  Logical volume "LVM-snapshot-lab" created
```

```
# lvdisplay /dev/vm0/LVM-snapshot-lab
  --- Logical volume ---
  LV Path                /dev/vm0/LVM-snapshot-lab
  LV Name                LVM-snapshot-lab
  VG Name                vm0
  LV UUID                md1QF6-NI2p-tmxB-9IeO-mlBi-Xbi6-IUB3xE
  LV Write Access        read/write
  LV Creation host, time iSCSI-1StInitiator, 2012-09-09 21:49:31 +0200
  LV snapshot status     active destination for root
  LV Status              available
  # open                 0
  LV Size                30,41 GiB
  Current LE             7784
  COW-table size         19,98 GiB
  COW-table LE           5115
  Allocated to snapshot  0,00%
  Snapshot chunk size    4,00 KiB
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:3
```

```
# mkdir /mnt/LVM-snapshot-lab
# mount /dev/vm0/LVM-snapshot-lab /mnt/LVM-snapshot-lab/
```

```
# ll /mnt/LVM-snapshot-lab/
total 112K
drwxr-xr-x  2 root root 4,0K sept.  5 11:36 bin
drwxr-xr-x  2 root root 4,0K oct.  25  2010 boot
drwxr-xr-x  5 root root 4,0K oct.  25  2010 dev
drwxr-xr-x 79 root root 4,0K sept.  9 18:17 etc
drwxr-xr-x  3 root root 4,0K oct.  25  2010 home
lrwxrwxrwx  1 root root   30 sept.  5 11:36 initrd.img -> /boot/initrd.img-3.2.0-3-amd64
drwxr-xr-x 14 root root  12K sept.  5 11:36 lib
drwxr-xr-x  2 root root 4,0K sept.  5 11:33 lib64
drwx------  2 root root  16K oct.  25  2010 lost+found
drwxr-xr-x  3 root root 4,0K oct.  25  2010 media
drwxr-xr-x  2 root root 4,0K août   6  2010 mnt
drwxr-xr-x  2 root root 4,0K oct.  25  2010 opt
drwxr-xr-x  2 root root 4,0K août   6  2010 proc
drwx------  4 root root 4,0K sept.  7 17:18 root
drwxr-xr-x  2 root root 4,0K déc.  23  2011 run
drwxr-xr-x  2 root root  12K sept.  9 17:05 sbin
drwxr-xr-x  2 root root 4,0K juil. 21  2010 selinux
drwxr-xr-x  2 root root 4,0K oct.  25  2010 srv
drwxr-xr-x  2 root root 4,0K août  15  2010 sys
drwxrwxrwt  2 root root 4,0K sept.  9 18:17 tmp
drwxr-xr-x 10 root root 4,0K janv. 29  2012 usr
drwxr-xr-x 11 root root 4,0K janv. 29  2012 var
```

```
# mkfs.ext4 /dev/vm0/backup
mke2fs 1.42.5 (29-Jul-2012)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
« Stride » = 0 blocs, « Stripe width » = 0 blocs
786432 i-noeuds, 3145728 blocs
157286 blocs (5.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=3221225472
96 groupes de blocs
32768 blocs par groupe, 32768 fragments par groupe
8192 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocation des tables de groupe : complété
Écriture des tables d'i-noeuds : complété
Création du journal (32768 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété

# mkdir /backup
# mount /dev/vm0/backup /backup/
```

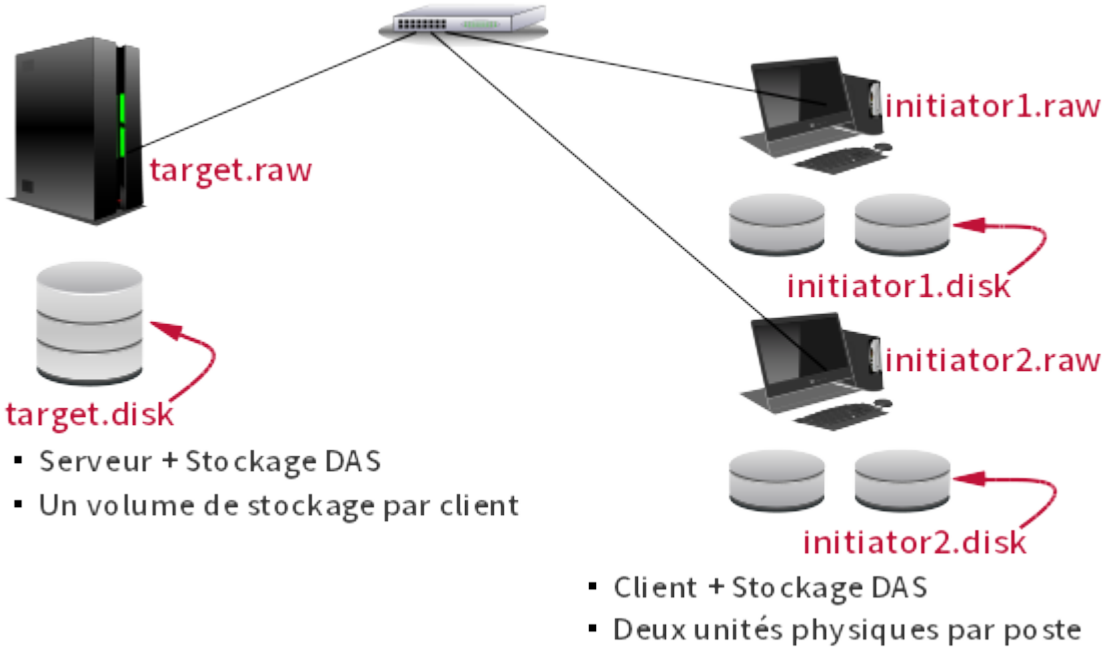
```
# tar --exclude-from backup-exclude.list -cvjf /backup/actually.tar.bz2 /

# /backup
/lib/init/rw
/proc
/sys
/dev
/run
/mnt
/selinux
/media
/var/lib/nfs
/etc/lvm
```

10. Manipulations sur machines virtuelles

Il est possible de réaliser l'ensemble des manipulations de ce support à l'aide de deux ou trois instances de machines virtuelles appartenant un même réseau de diffusion (LAN).

L'infrastructure à implanter sur le système hôte est la suivante.



Topologie virtualisation iSCSI - vue complète

On débute avec la création des fichiers image des trois systèmes virtuels. Les fichiers de type `.qed` sont des images compressées faciles à transférer.

```
$ mkdir -p ~/vm/iscsi
$ cd ~/vm/iscsi
$ ionice -c 3 cp ../vm0-debian-testing-i386-base.raw target.raw
$ ionice -c 3 cp ../vm0-debian-testing-i386-base.raw initiator1.raw
$ ionice -c 3 cp ../vm0-debian-testing-i386-base.raw initiator2.raw
```

On crée ensuite les fichiers correspondant aux unités de stockage supplémentaires.

```
$ dd if=/dev/null of=target.disk bs=1 seek=72G
$ dd if=/dev/null of=initiator1.disk bs=1 seek=32G
$ dd if=/dev/null of=initiator2.disk bs=1 seek=32G
```

Enfin, il ne reste qu'à mettre en place le script de lancement de ces trois systèmes avec leurs unités de stockages respectives.

```
#!/bin/bash

../scripts/ovs-startup.sh target.raw 4096 0 \
        -drive if=none,id=storagevol0,aio=native,cache.direct=on,format=raw,media=disk,file=target.disk
        -device virtio-blk,drive=storagevol0,scsi=off,config-wce=off,x-data-plane=on

../scripts/ovs-startup.sh initiator1.raw 1024 1 \
        -drive if=none,id=initiator1addon,aio=native,cache.direct=on,format=raw,media=disk,file=initiat
        -device virtio-blk,drive=initiator1addon,scsi=off,config-wce=off,x-data-plane=on

../scripts/ovs-startup.sh initiator2.raw 1024 2 \
        -drive if=none,id=initiator2addon,aio=native,cache.direct=on,format=raw,media=disk,file=initiat
        -device virtio-blk,drive=initiator2addon,scsi=off,config-wce=off,x-data-plane=on
```

Ce script fait lui-même appel au script commun `ovs-startup.sh` qui sert à initialiser une instance de machine virtuelle en utilisant comme paramètres le nom du fichier image, la quantité de RAM et le cordon de brassage virtuel tap. Le guide *Virtualisation système et enseignement* fournit le code source du script de lancement d'une machine virtuelle raccordée à un commutateur Open vSwitch.

11. Évaluation des performances

Voici quelques exemples de mesures de performances d'accès aux volumes de stockage. L'objectif est de présenter quelques outils qui produisent des résultats dans un laps de temps relativement court.

Note

La pertinence ou la validité des résultats dépendent énormément du facteur temps. Une mesure valide suppose un temps d'exécution de quelques heures au moins pour chaque outil. Les résultats donnés ici ne peuvent donc pas être considérés comme représentatif des performances de chaque technologie de stockage.

Il convient de décrire de façon très détaillée les conditions dans lesquelles ces tests sont réalisés. En effet, d'une plateforme matérielle à l'autre la distorsion des performances est considérable.

Tous les résultats ci-dessous sont obtenus avec l'outil bonnie++ et une taille de fichier de 8Go.

Unité de disque locale

Système de fichiers `ext3` avec gestion de volume logique LVM.

```
# time bonnie++ -u 1000 -s 8000 -d /var/tmp >result.txt
<snipped>
# cat result.txt
Version  1.96       ------Sequential Output------ --Sequential Input- --Random-
Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
iSCSI-1StInit 8000M   511  99 234868  55 180260  30  2985  99 615617  49 15925 260
Latency             26238us     535ms     545ms    4181us    8362us   63959us
Version  1.96       ------Sequential Create------ --------Random Create--------
iSCSI-1StInitiator  -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
              files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                 16 +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++
Latency               411us     779us    1413us     265us      28us     699us
```

Unité de disque iSCSI

Système de fichiers `ext4`.

```
# time bonnie++ -u 1000 -s 8000 -d /mnt/tmp >result.txt
<snipped>
# cat result.txt
Version  1.96       ------Sequential Output------ --Sequential Input- --Random-
Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
iSCSI-1StInit 8000M   534  99 96128  15 44584  11  2761  98 109216  16  3112  96
Latency             17770us     961ms     333ms    6060us    7910us   76502us
Version  1.96       ------Sequential Create------ --------Random Create--------
iSCSI-1StInitiator  -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
              files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                 16 27168  50 +++++ +++ +++++ +++ +++++ +++ +++++ +++ +++++ +++
Latency              1228us     762us     820us     262us      34us     749us
```

Tableau RAID1 constitué d'une unité de disque locale et d'une unité de disque iSCSI

Système de fichiers `ext4`.

```
# time bonnie++ -u 1000 -s 8000 -d /mnt/tmp >result.txt
<snipped>
# cat result.txt
Version  1.96       ------Sequential Output------ --Sequential Input- --Random-
Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
iSCSI-1StInit 8000M   525  99 93851  15 60117  12  2795  95 177757  19  3707  99
Latency             25078us     729ms     194ms   45986us     343ms    1055ms
Version  1.96       ------Sequential Create------ --------Random Create--------
iSCSI-1StInitiator  -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
              files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                 16 26606  51 +++++ +++ +++++ +++ +++++ +++ +++++ +++ 30648  41
Latency               195us     791us     823us     351us      47us     745us
```

12. Documents de référence

Architecture réseau des travaux pratiques

Architecture réseau des travaux pratiques : présentation de l'implantation des équipements d'interconnexion réseau dans l'armoire de brassage et du plan d'adressage IP prédéfini pour l'ensemble

des séances de travaux pratiques. Ce document est utilisé dans la Section 2, « Adressage IP des postes de travail ».

Configuration d'une interface réseau

Configuration d'une interface de réseau local : tout sur la configuration des interfaces réseau de réseau local. Comme dans le cas précédent, ce document est utile pour effectuer les opérations demandées dans la Section 2, « Adressage IP des postes de travail ».

Introduction to iSCSI

L'article intitulé *Introduction to iSCSI* du site Linux Magazine présente les points clés de la technologie iSCSI. Il complète la Section 3, « Technologie iSCSI et topologie de travaux pratiques ».

iSCSI - Debian Wiki

La page *iSCSI and Debian* contient deux sous-rubriques sur les rôles *initiator* et *target*. Pour le rôle *target*, la section relative à l'utilisation du sous système *Linux-IO : the Linux SCSI Target wiki* n'a pas encore été documentée.