



Documentation  
technique

Travail  
Pratique  
Individuel



Lucas Dousse  
I.IN-D4a  
Informaticien CFC  
2017



## Table des matières

---

1	Introduction.....	5
1.1	Pourquoi ce sujet.....	5
1.2	En quoi consiste-t-il ? .....	5
2	Installation.....	6
2.1	Hardware .....	6
2.2	Serveur 1.....	6
2.3	Serveur 2.....	6
2.4	Plan d'adressage IP .....	6
2.4.1	Publique 213.162.11.176/28.....	6
2.4.2	Privé 192.168.42.0/28.....	7
2.5	Software.....	7
2.5.1	Installation de base .....	7
2.5.2	Configuration de base des hôtes.....	10
3	Installation et configuration de ZFS .....	11
3.1	Qu'est-ce que c'est ? .....	11
3.2	Vocabulaire à connaître.....	11
3.3	Installations des paquets manquants.....	12
3.4	Initialisation des disques .....	12
3.5	Formatage des disques.....	12
3.6	Création du pool zfs.....	13
3.6.1	Explication des paramètres .....	13
3.7	Test des 2 stockages .....	14
4	Installation et config de KVM .....	15
4.1	Qu'est-ce que c'est ? .....	15
4.2	Installations des paquets manquants.....	15
4.3	Configuration de base.....	16
4.4	Vérification de l'installation.....	16
4.5	Définitions du stockage .....	16
4.6	Configuration du réseau .....	18
4.6.1	Configuration de l'hôte .....	18
4.6.2	Explication de la configuration.....	18
4.6.3	Configuration de KVM .....	19

5	Déploiement de l'environnement de test .....	21
5.1	Schéma de l'environnement.....	21
5.2	Déploiement de la vm « Firewall » .....	21
5.2.1	Créations de la VM dans kvm .....	22
5.2.2	Installation de la vm .....	22
5.2.3	Configuration des vm guest agent .....	23
5.2.4	Création du réseau NAT .....	23
5.3	Déploiement de la vm « Linux » .....	25
5.3.1	Déploiement de la vm .....	25
5.3.2	Installation de la vm .....	25
5.3.3	Configuration particulière .....	25
5.4	Déploiement de la vm « Windows » .....	25
5.4.1	Déploiement de la vm .....	25
5.4.2	Installation de la vm .....	26
5.4.3	Config particulière .....	30
6	Mise en place de la réPLICATION des données .....	31
6.1	Généralité .....	31
6.2	Fonctions de chaque script.....	31
6.3	Mise en place des scripts.....	32
6.4	Retour après le failover .....	32
6.5	Amélioration possible.....	32
7	Test et validation du processus.....	33
7.1	Liste des tests .....	33
7.2	Failover manuel simple.....	33
7.2.1	Contexte .....	33
7.2.2	Action à réaliser.....	33
7.2.3	Résultats attendus.....	33
7.2.4	Résultats obtenus.....	34
7.3	Failover manuel à l'aide d'un script.....	35
7.3.1	Contexte .....	35
7.3.2	Action à réaliser.....	35
7.3.3	Résultats attendus.....	35
7.3.4	Résultats obtenus.....	35

7.4	Failover manuel avec de l'écriture sur les vm .....	37
7.4.1	Contexte .....	37
7.4.2	Action à réaliser.....	37
7.4.3	Résultats attendus.....	37
7.4.4	Résultats obtenus.....	37
7.5	Test de perte physique d'un serveur.....	38
7.5.1	Contexte .....	38
7.5.2	Action à réaliser.....	38
7.5.3	Résultats attendus.....	38
7.5.4	Résultats obtenus.....	38
8	Complément.....	39
8.1	zRam .....	39
8.2	Backup de mon système.....	39
8.3	Monitoring de mon système .....	39
9	Conclusion .....	40
9.1	Connaissance .....	40
9.2	Regard critique .....	40
10	Sources .....	41
10.1	Articles en lignes .....	41
10.2	Magazine.....	41
11	Annexes .....	42



# 1 Introduction

## 1.1 Pourquoi ce sujet

Vtx m'a proposé ce sujet car je travaille dans la team Unix. J'adore aussi découvrir de nouvelles technologies ainsi que travailler avec. Quand on m'a présenté ce projet, j'ai tout de suite été ravi de ce travail. Je me réjouis de mettre en pratique mes connaissances avec ZFS et KVM dans un cas concret. Surtout que ce projet pourrait peut-être être utilisé dans un cas de production au sein de notre plate-forme cloudstack. Cette plate-forme est une plate-forme de VPS destiné à nos clients.

Ce travail marque aussi l'accomplissement de mon apprentissage au sein de VTX. 4 années ou plutôt 3 devrais-je dire car la première année de mon apprentissage s'est déroulée chez infologo. En première année, j'ai fait beaucoup d'installation sur site avec des techniciens qui m'ont apporté beaucoup d'un point de vue technique : mise en place de serveurs ainsi que de PBX Avaya. Ensuite mes 3 autres années se passèrent chez VTX. Ces 3 années, furent enrichissantes d'un point de vue de la gestion de projet ainsi que de la maintenance de serveur UNIX.

Durant cette documentation nous ferons abstraction de tout ce qui est configuration réseaux de base et paramétrage de base, le public cible étant des ingénieurs système ou administrateurs système cela va de soi que c'est un prérequis.

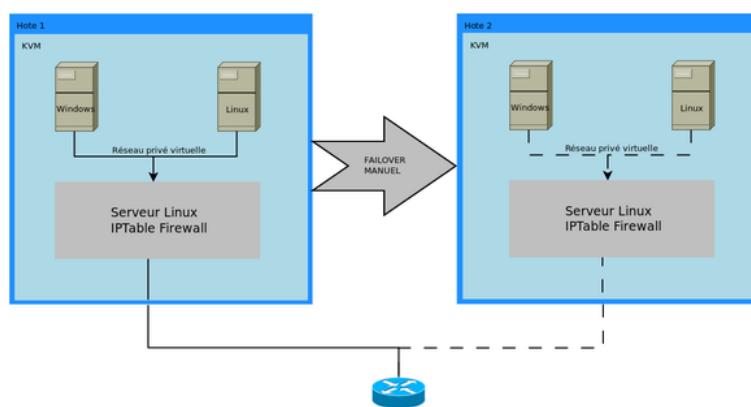
Ce projet a eu pour durée 120 heures (Pratique + Documentation).

## 1.2 En quoi consiste-t-il ?

Mon projet est un projet de virtualisation sur des serveurs Ubuntu avec un plan de Failover manuel sur un deuxième serveur. Avec un plan de Disaster Recovery Plan.

Les 2 serveurs physiques feront tourner KVM comme hyperviseur. Il y aura aussi localement à chaque serveur un pool de stockage ZFS. Ce pool de stockage accueillera les données des machines virtuelles. Le deuxième serveur sera uniquement une copie du premier. Ce deuxième serveur servira au cas où le premier subit un dommage.

A la fin cela devrait ressembler au schéma ci-dessous :



## 2 Installation

### 2.1 Hardware

2 Serveurs DELL POWEREDGE R410

### 2.2 Serveur 1

Nom	Srv-ldo-tpi-01
Domaine	dsprod.ch
Fonctions	Premier serveurs KVM
RAM	16Gb
CPU	Intel Xeon E5520 @ 2.27GHz x16
DISQUES	2x250Gb en RAID 1 hardware (/dev/sdc) 2x500Gb (/dev/sda & /dev/sdb)
IP eno1	213.162.11.178 (srv-ldo-tpi-01.dsprod.ch)
IP iDrac	213.162.11.179 (idrac.srv-tpi-ldo-01.dsprod.ch)

### 2.3 Serveur 2

Nom	Srv-ldo-tpi-02
Domaine	dsprod.ch
Fonctions	Deuxième serveurs KVM
RAM	16Gb
CPU	Intel Xeon E5504 @ 2.00GHz x4
DISQUES	2x500Gb en RAID 1 hardware (/dev/sda) 2x500Gb (/dev/sdb & /dev/sdc)
IP eno1	213.162.11.178 (srv-tpi-ldo-02.dsprod.ch)
IP iDrac	213.162.11.179 (idrac.srv-tpi-ldo-02.dsprod.ch)

### 2.4 Plan d'adressage IP

#### 2.4.1 Publique 213.162.11.176/28

IP address	Description	Hostname
213.162.11.177	Switch 48 port - uplink	gve-dp7-er-35-01-40.vtxnet.net
213.162.11.178	Server dell r410	srv-tpi-ldo-01.dsprod.ch
213.162.11.179	iDrac - Server dell r410	idrac.srv-tpi-ldo-01.dsprod.ch
213.162.11.180	Server dell r410	srv-tpi-ldo-02.dsprod.ch
213.162.11.181	iDrac - Server dell r410	idrac.srv-tpi-ldo-02.dsprod.ch
213.162.11.182	Firewall Machine (VM)	fw-tpi-ldo-01.dsprod.ch
213.162.11.190	Switch 10 port - uplink	SW-TPI-LDO-01.dsprod.ch

## 2.4.2 Privé 192.168.42.0/28

IP address	Description	Hostname
192.168.42.1	VM - Firewall translation NAT	fw-tpi-lde-01.dsprod.ch
192.168.42.10	VM - Windows	vm-tpi-lde-01.dsprod.ch
192.168.42.11	VM - Linux	vm-tpi-lde-02.dsprod.ch

## 2.5 Software

### 2.5.1 Installation de base

Chaque Hôte aura cette configuration :

1. Télécharger l'ISO de Ubuntu serveur 16.04 et la graver sur une clé USB
2. Démarrer le serveur sur la clé USB
3. Sur le premier écran sélectionné « Install Ubuntu Server »

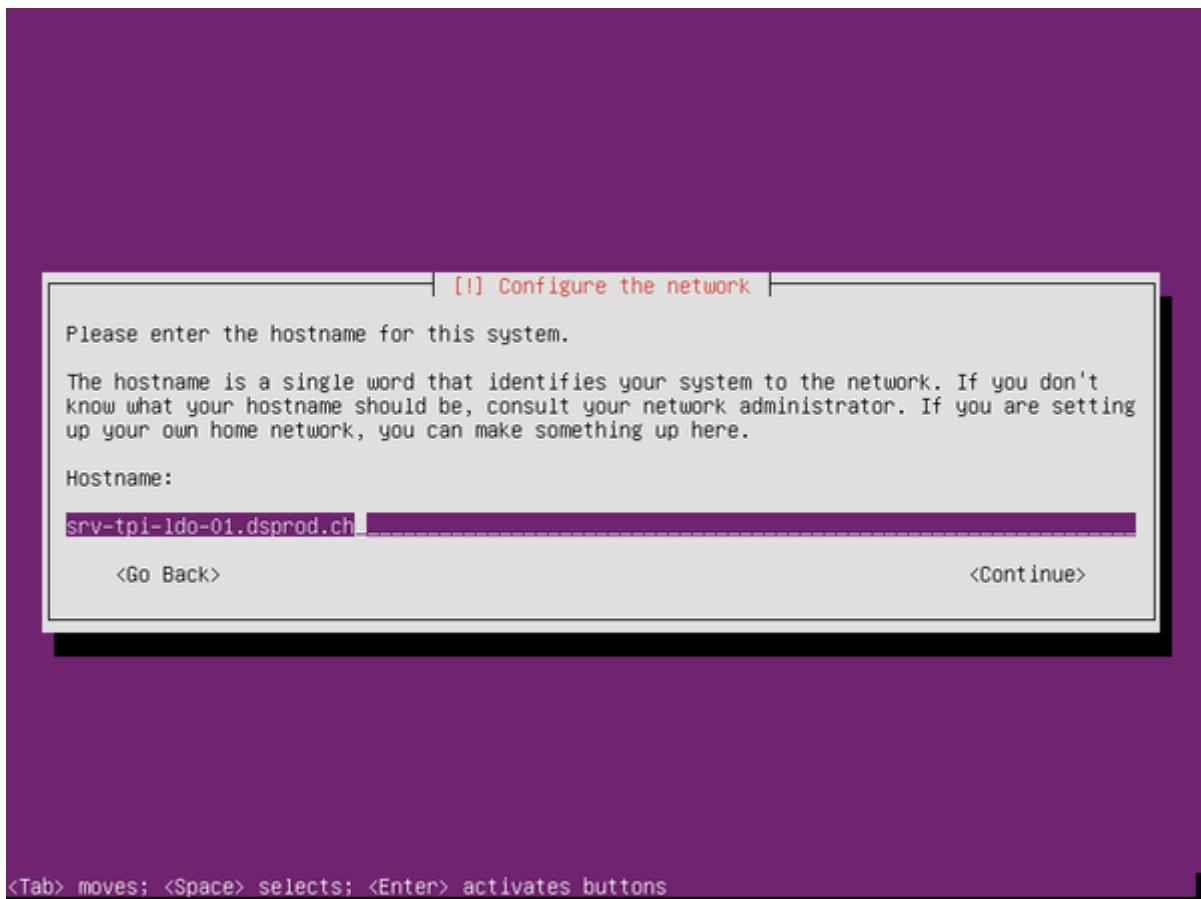


4. Sur l'écran suivant, il faut sélectionner la langue par défaut du système, ici je vais l'installer en anglais.
5. Ensuite, je vais sélectionner le pays dans lequel nous nous trouvons, ici je vais choisir « autre » puis « Europe » puis « Switzerland »
6. Je vais laisser tel quel avec « United States – en\_US.UTF-8 »
7. Je vais configurer le choix du clavier manuellement en choisissant « Switzerland » puis « Switzerland – French (Switzerland) »

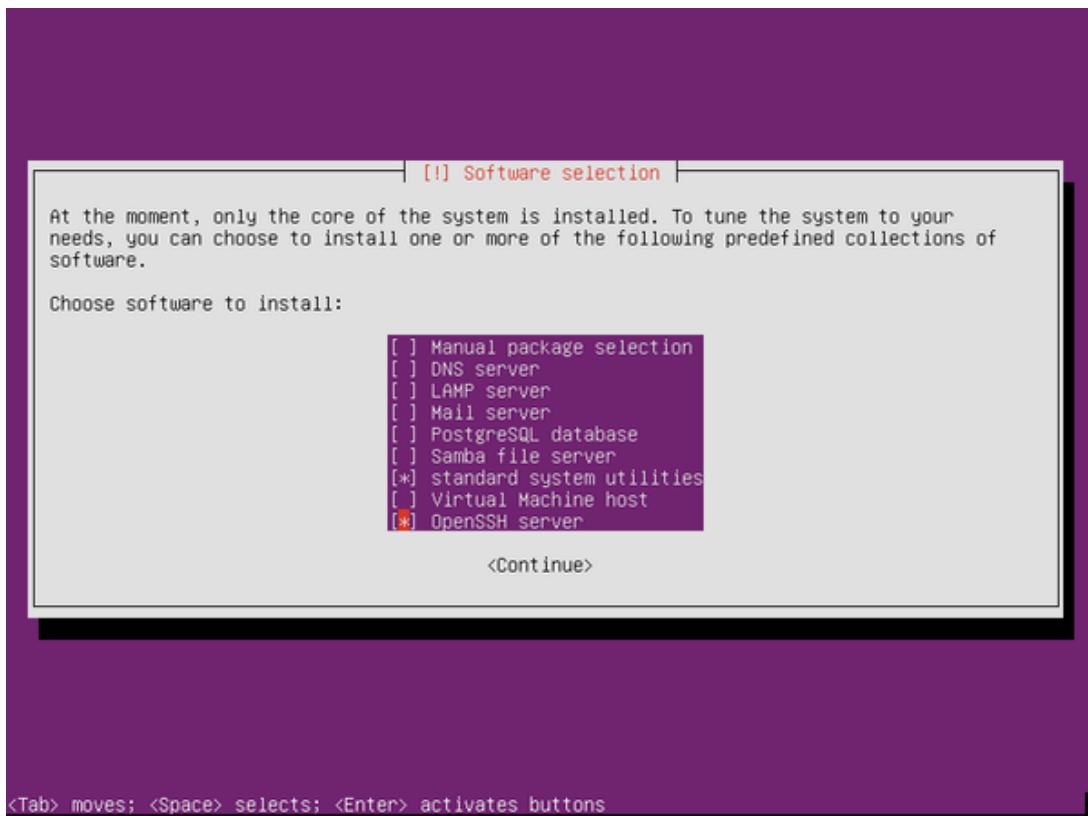
8. A ce stade, nous devons configurer le réseau car il va essayer de se configurer en DHCP. Vu que nous sommes sur un réseau « Public » j'ai pas de DHCP. Je vais donc choisir « Configure network manually »
9. J'ai comme spécifié ci-dessus un bloc de 16 ip en 213.162.11.176/28, je vais donc les utiliser dans l'ordre comme le plan d'adressage ci-dessus.



10. Vu que j'ai un /28 je vais mettre comme Netmask 255.255.255.240
11. Comme Gateway : 213.162.11.177
12. Je vais aussi utiliser nos serveurs DNS : 212.147.10.10
13. Nous devons maintenant leurs donner un nom. Je vais le spécifier suivant les infos citées plus haut.



14. Sur l'écran, suivant il va falloir définir le Nom de l'utilisateur. Suivant les habitudes internes de VTX, nous avons un utilisateur qui s'appelle VTX SSH qui fera très bien son affaire ici.
15. Je vais choisir un mot passe complexe par exemple avec 24 caractères car ces machines seront quand même directement en public.
16. Je vais donc sélectionner « Guided – Use entire disk and setup LVM »
17. Je vais sélectionner notre volumes RAID créé auparavant avec les 2 plus petits disques
18. Je vais écrire les changements sur le disque
19. Je vais choisir d'installer la mise à jour de sécurité automatiquement vu que les machines sont quand même vulnérables.
20. Je vais choisir « Standard system utilities » et « OpenSSH server » afin d'avoir les outils de base type git, dig, ... et openssh pour se connecter à distance.



<Tab> moves; <Space> selects; <Enter> activates buttons

21. Je vais installer le Grub sur le même volume que celui du système
22. Je vais redémarrer la machine et l'Installation est terminée

### 2.5.2 Configuration de base des hôtes

Chaque Hôte aura cette configuration :

1. J'ai commencé par enlever le cd dans le /etc/apt/sources.list
2. Ajouter les repo non-free dans le même fichier
3. Faire les updates avec :

apt update && apt upgrade

4. Installation des outils de base manquants :

apt install iotop iftop htop sudo subversion zsh pwgen screen tcpdump vim ntp ntpdate iptables-persistent ethtool git rsync tree curl fail2ban -y

5. Configuration de mon ZSH ainsi que les plugins Vim ainsi que git : Les scripts pour ces étapes sont disponibles sur mon GitHub : <https://git.io/v94W8>
6. Ajout de mes clés publique dans le ~/.ssh/authorized\_keys de ROOT afin de pouvoir se connecter sans mot de passe
7. Configuration de iptables-persistent dans /etc/iptables/rules.v4
  - o Par défaut je bloque tout en entrée et ensuite j'autorise les range ip vtx sur tous les ports

## 3 Installation et configuration de ZFS

### 3.1 Qu'est-ce que c'est ?

ZFS (System Files Zettabyte) est un système de fichiers distribué de Sun. Avec un support de 128 bits, contre 64 pour NTFS et d'autres systèmes de fichiers contemporains, la capacité de stockage est illimitée à tout fins pratiques. Qui est la raison du nom "zettabyte". ZFS prend en charge jusqu'à 18 pools de stockage virtuels de 1 zêta. En raison de cette capacité titanique, son stockage est considéré comme illimité. ZFS comprend la vérification de l'intégrité des données, la protection contre la corruption de données, la prise en charge de capacités de stockage élevées, de bonnes performances, de la réPLICATION, des instantanés et des clones de copie sur écriture, et l'auto reconstruction qui en font un choix naturel pour le stockage de données. Les possibilités sont infinies avec ZFS, par exemple mettre la ram en zfs et ainsi gagner beaucoup de performance sur la RAM. Si j'ai le temps à la fin de mon tpi, je le ferai.

### 3.2 Vocabulaire à connaître

**Raidz** : Le système de fichiers ZFS fournit RAID-Z, un schéma de distribution de données / parité similaire à RAID 5, mais en utilisant une largeur de bande dynamique : chaque bloc à sa propre bande RAID, quelle que soit la taille des blocs, ce qui permet à chaque écriture RAID-Z d'être une écriture complète. RAID-Z est également plus rapide que le RAID 5 traditionnelle car il n'a pas besoin d'effectuer la séquence habituelle de lecture-modification-écriture. Il existe plusieurs RAID-Z, cité ci-dessous :

MIRROR (raid1) utilisé avec deux à quatre disques ou plus.

RAIDZ-1 (raid5) utilisé avec cinq disques ou plus.

RAIDZ-2 (raid6) utilisé avec six disques ou plus.

RAIDZ-3 (raid7) utilisé avec onze disques ou plus.

Dans mon cas ce que je vais devoir choisir du mirror vu que j'ai 2 disques.

**Zpool** : Contrairement aux systèmes de fichiers traditionnels qui résident sur des périphériques uniques et exigent qu'un gestionnaire de volume utilise plus d'un périphérique. Les systèmes de fichiers ZFS sont construits sur des pools de stockage virtuels appelés zpools. Un zpool est constitué de périphériques virtuels (vdevs), eux-mêmes construits à partir de périphériques de blocs : fichiers, partitions de disque dur ou lecteurs entiers, ce dernier étant l'utilisation recommandée.

### 3.3 Installations des paquets manquants

Avant de commencer l'installation et la configuration de zfs il faut installer les packages nécessaires.

Avec la commande suivante :

```
apt install zfs-initramfs zfsutils-linux
```

**zfsutils-linux** : contient tous les outils pour pouvoir gérer et manager les différents zpool

**zfs-initramfs** : contient le core de zfs

### 3.4 Initialisation des disques

Le mappage des disques est tel que représenté ci-dessous :

```
srv-tpi-ldo-01
/dev/sda zfs1
/dev/sdb zfs2
/dev/sdc System

srv-tpi-ldo-02
/dev/sda System
/dev/sdb zfs1
/dev/sdc zfs2
```

### 3.5 Formatage des disques

Maintenant, il va falloir formater les disques pour cela je vais avoir besoin de sgdisk qui fait parti du package ubuntu gdisk. Je vais installer gdisk avec : apt install gdisk

Je vais maintenant formater les disques avec la commande :

```
sgdisk -n9:-8M:0 -t9:BF07 /dev/sdb
```

Le -n9 c'est le numéro de partition que l'on donne au secteur d'amorçage qui doit correspondre avec le t9. Ce qui suit le -n9 est la taille que l'on donne à ce secteur à savoir dans notre cas. Il commence 8méga avant la fin du dernier block disponible du disque. Se termine à 0. Je vais maintenant donner le type de secteur que je veux à ma partition. Le -t9 :BF07 donne que le numéro du secteur précédemment donné qui est le 9 aura le type BF07. BF07 représente le type ZFS.

## 3.6 Cration du pool zfs

Je vais crer un zpool contenant un VDEV de 2 disques dans un miroir :

```
zpool create -f \
-o ashift=12 \
-o failmode=continue \
-o atime=off \
-o mountpoint=/zroot \
-o compression=lz4 \
-o checksum=sha256 \
-R /zroot \
zroot mirror /dev/disk/by-id/ata-WDC_WD5003ABYX-01WERA0_WD-WMAYP0616826
/dev/disk/by-id/ata-WDC_WD5003ABYX-01WERA0_WD-WMAYP0800491
```

### 3.6.1 Explication des paramtres

- ashift : Meilleure performance en criture, mais limitation suivant le raidz (Performance, capacit)
- failmode : Controle le comportement du systme si une panne de pool catastrophique se produit. Cette condition rsulte habituellement de la perte de connectivit du priphrique de stockage ou des priphriques sous-jacents ou d'une dfaillance de tous les priphriques du pool. Le comportement d'un tel vnement est dtermin par la valeur suivante : Continue - Renvoie une erreur d'EIO `a toutes les nouvelles requtes d'E / S d'criture, mais permet de lire l'un des priphriques sain restant. Toutes les requtes d'criture qui n'ont pas encore te engages sur le disque sont bloques. Une fois l'appareil reconnect ou remplac, les erreurs doivent tre effaces avec la commande zpool clear.
- atime : Le kernel ne doit pas enregistrer chaque access time s'il est sur off
- mountpoint : C'est le point de montage automatique du pool
- compression : En interne, ZFS alloue des donnes `a l'aide de multiples de la taille du secteur de l'appareil, gneralement 512 octets ou 4KB (voir ci-dessus). Lorsque la compression est active, un nombre plus petit de secteurs peut tre attribu pour chaque bloc. La taille du bloc non compress est dfinie par la proprit recordsize (par dfaut pour 128KB) ou volblocksize (par dfaut pour 8KB) (pour les systmes de fichiers et les volumes).
- checksum : fait la somme de contrle
- -R : C'est le dossier de montage immdiat

A la fin de ces paramtres, il reste une ligne. Cette ligne contient les dernires informations ncessaires `a la cration du zpool. Je vais donner un nom `a notre zpool, qui est reprsent par « zroot ». Ensuite le « mirror » donne le type de RaidZ que je veux faire. Maintenant, il va falloir dfinir les disques sur lesquels nous voulons notre zraid. Nous allons dfinir les disk avec leur id et non leur nom. C'est mieux pour les petits systmes avec un seul contrleur de disque. Parce que les noms sont persistants et du coup garantit de ne pas changer, peu importe la faon dont les disques sont attachs au systme.

Vous pouvez les enlever tous, les mélanger au hasard sur le bureau, les remettre partout dans le système et votre pool sera automatiquement reconstitué.

Une dernière chose que j'ai dû faire c'est ajouter « zfs mount -a » dans le /etc/rc.local. Afin de monter le stockage zfs à chaque démarrage

### 3.7 Test des 2 stockages

J'ai utilisé un outil de benchmark qui s'appelle bonnie ++ afin de tester la différence entre les 2 serveurs vu qu'ils n'ont pas tout à fait les mêmes disques. La commande utilisée fut :

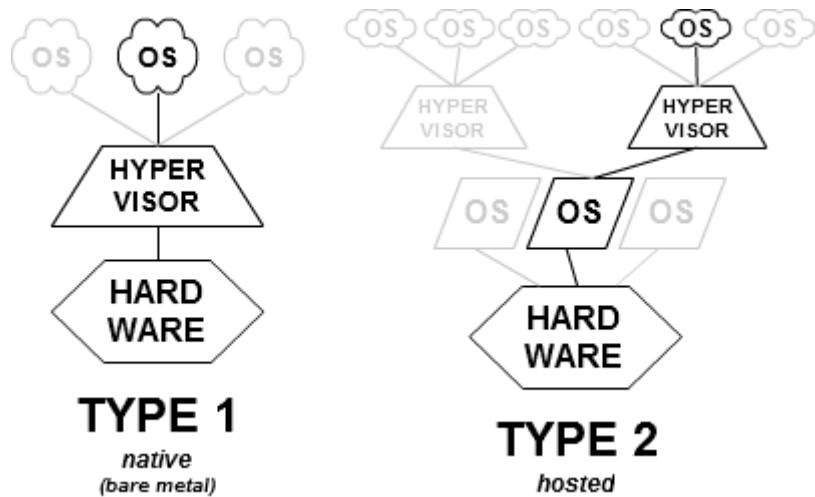
```
bonnie++ -u root -r 1024 -s 16384 -d /zroot -f -b -n 1 -c 4 | bon_csv2html | tee /root/bonnie_benchmark1.html
```

Version 1.97		Sequential Output				Sequential Input				Random Seek				Sequential Create				Random Create						
Concurrency	Size	Per Char	Block	Rewrite	Per Char	Block			sec	% CPU			sec	% CPU			sec	% CPU			sec	% CPU		
		K/sec % CPU	K/sec % CPU	K/sec % CPU	K/sec % CPU	K/sec % CPU			sec	% CPU			sec	% CPU			sec	% CPU			sec	% CPU		
stv-tpi-ldo-01	4 16G	509920 91	255399 69		917965 83	345.5 17	1	111 2	+++++ +++	114 2	111 2	+++++ +++	109 2											
	Latency	7093us	118ms		95633us	234ms	Latency	74270us	29us	46566us	49165us	20us	77919us											
srv-tpi-ldo-02	4 16G	472748 90	331858 87		827585 71	874.4 36	1	+++++ +++	+++++ +++	+++++ +++	941us	1199us	11us	12664us										
	Latency	14813us	52124us		540ms	70902us	Latency	20504us	8us															

## 4 Installation et config de KVM

### 4.1 Qu'est-ce que c'est ?

KVM (Kernel-based Virtual Machine) est un hyperviseur libre et open source de type I pour Linux. KVM est intégré dans le noyau Linux depuis la version 2.6.201.



Il fonctionne originellement sur les processeurs à architectures x86 disposant des instructions de Virtualisation Intel VT ou AMD-V 2. Depuis, KVM a été porté pour les architectures Power PC3, IA-64 ainsi que ARM depuis le noyau Linux 3.9 4.

### 4.2 Installations des paquets manquants

Avant de commencer l'installation et la configuration de KVM, il faut installer les packages nécessaires.

Avec la commande suivante :

```
apt install qemu-kvm libvirt-bin virtinst bridge-utils cpu-checker virt-manager
```

**Qemu-kvm** : KVM (Kernel Virtual Machine) est un module kernel Linux qui permet d'utiliser les fonctionnalités de virtualisation matérielle de différents processeurs.

**Libvirt-bin** : Libvirt est une API, trousse à outils pour gérer les hôtes de virtualisation.

**Virtinst** : L'outil "Virt Install" est un outil de ligne de commande qui fournit un moyen simple de déployer des vm.

**Bridge-utils** : Le paquet bridge-utils est un utilitaire pour créer et gérer des périphériques en bridge.

**Cpu-checker** : Est un module qui permet à kvm d'afficher le % de cpu utilisé par chaque vm

**Virt-manager** : L'application virt-manager est une interface utilisateur de bureau pour gérer des vm via libvirt. Il cible principalement les VM KVM, mais gère également Xen et LXC (container linux).

## 4.3 Configuration de base

Après cette installation, nous allons encore rajouter notre utilisateur donc « root » dans le groupe « KVM » afin de ne pas être bloqué avec des droits. Avec la commande « adduser root kvm », je vais également ajouter le fait que kvm interroge les I/O du disk, la bande passante et la mémoire de chaque vm. Pour ce faire, je vais aller dans le virt-manager. Menu « Edit » puis « preferences » puis l'onglet « Polling ». Là, je vais cocher toutes les cases.

## 4.4 Vérification de l'installation

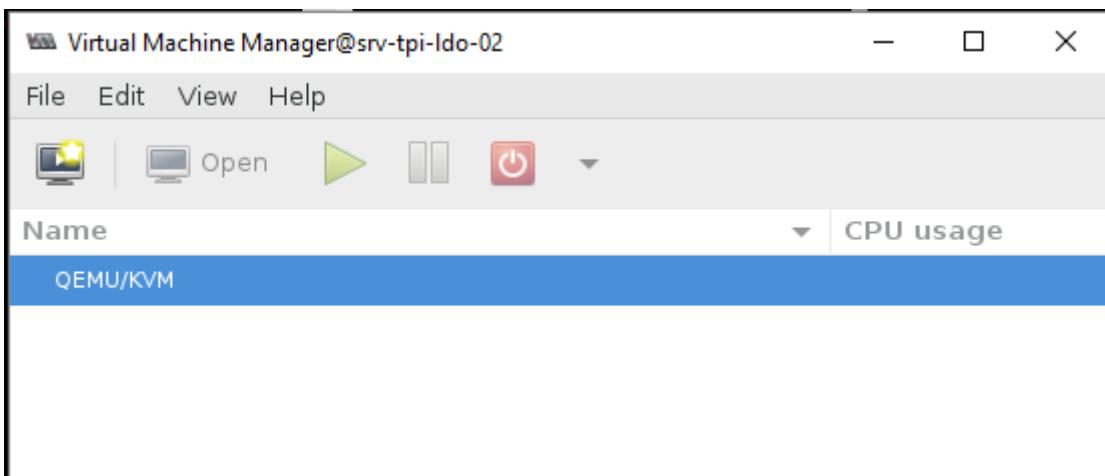
On peut vérifier que tout fonctionne correctement avec kvm. Pour cela, il suffit de taper la commande suivante : kvm-ok. S'il retourne :

```
INFO: /dev/kvm exists
KVM acceleration can be used
```

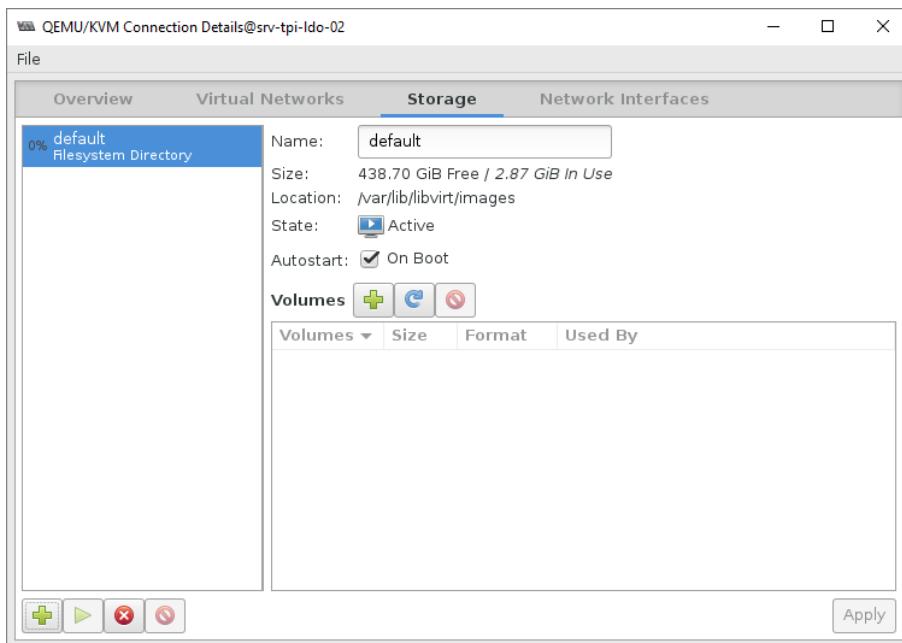
Tout va bien. Dans mon cas, un des deux serveurs m'a retourné que le vmx n'était pas activé. Pour l'activer j'ai dû redémarrer le serveur aller dans le bios. Puis activer la fonction de virtualisation dans le menu du processeur.

## 4.5 Définitions du stockage

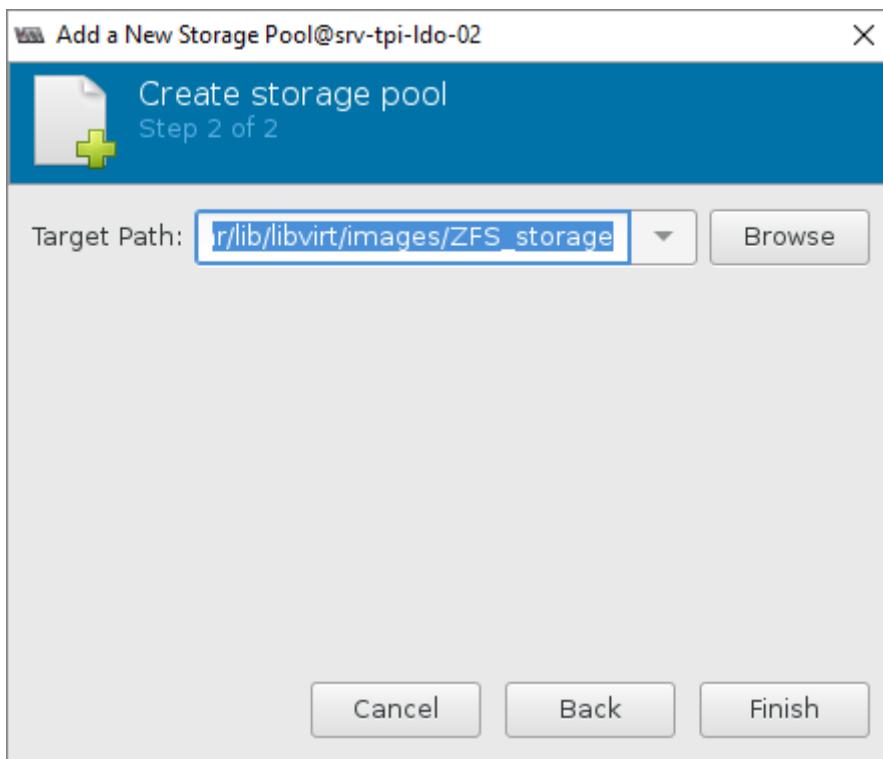
Je vais définir que le stockage ZFS dans notre KVM. Pour cela nous allons utiliser le virt-manager. Je vais simplement lancer le virt-manager avec la commande du même nom. J'ai cette fenêtre qui s'affiche.



Suite à cela, je vais cliquer sur « Edit » puis « Connection Details ». Je vais maintenant cliquer sur storage.



Je vais maintenant cliquer sur le « + » situé en bas à gauche. Je vais maintenant donner un nom à mon « storage pool ». Dans ce cas je vais l'appeler « ZFS\_storage ». La fenêtre suivante ressemble à cela :



Je vais donc maintenant changer le Path. Comme je l'ai défini à la configuration de zfs. Je vais mettre "/zroot".

## 4.6 Configuration du réseau

### 4.6.1 Configuration de l'hôte

C'est à cette étape que j'ai pris du retard car, pour configurer ensuite ma machine virtuelle en mode bridge il fallait que je passe ma carte réseau en mode bridge. Pour cela finalement, j'ai trouvé la syntaxe exacte du fichier de configuration `/etc/network/interface`

```
# The primary network interface
auto br0
iface br0 inet static
    address 213.162.11.180
    netmask 255.255.255.240
    network 213.162.11.176
    broadcast 213.162.11.191
    gateway 213.162.11.177
    bridge_ports en0
    bridge_stp off
    bridge_maxwait 5
    # dns-* options are implemented by the resolvconf package, if installed
    dns-nameservers 212.147.10.10
    dns-search dsprod.ch
```

Cette configuration doit remplacer la configuration de base de l'interface

### 4.6.2 Explication de la configuration

Les 5 premières lignes n'ont pas besoin d'explication je pense. Car suffisamment parlante.

**Bridge\_ports** : Ici je dois donner le nom de l'interface physique avec laquelle il va faire le bridge

**Bridge\_stp** : Ici je décideras de le mettre sur off. Car ce paramètre est utile pour le spanning-tree. Donc je n'en ai pas besoin sur cette interface.

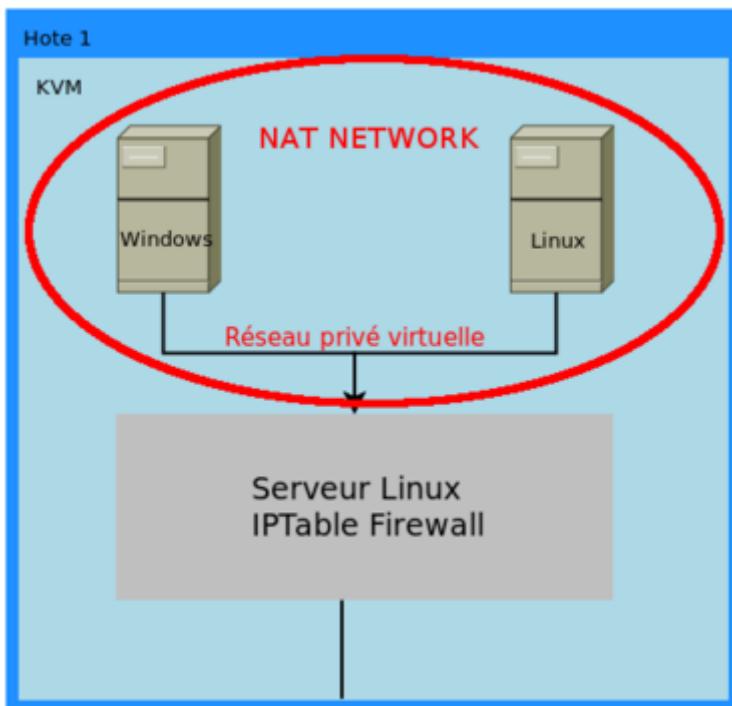
**Bridge\_maxwait** : Cela donne un temps d'attente au script des interfaces.

**Dns-nameserver** : Cela définit le serveur dns

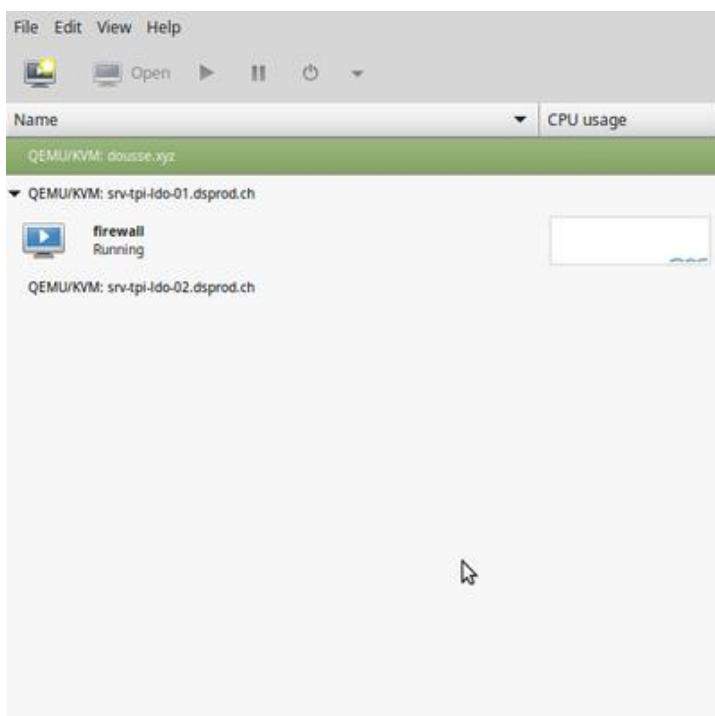
**Dns-search** : Cela définit le domaine de recherche

### 4.6.3 Configuration de KVM

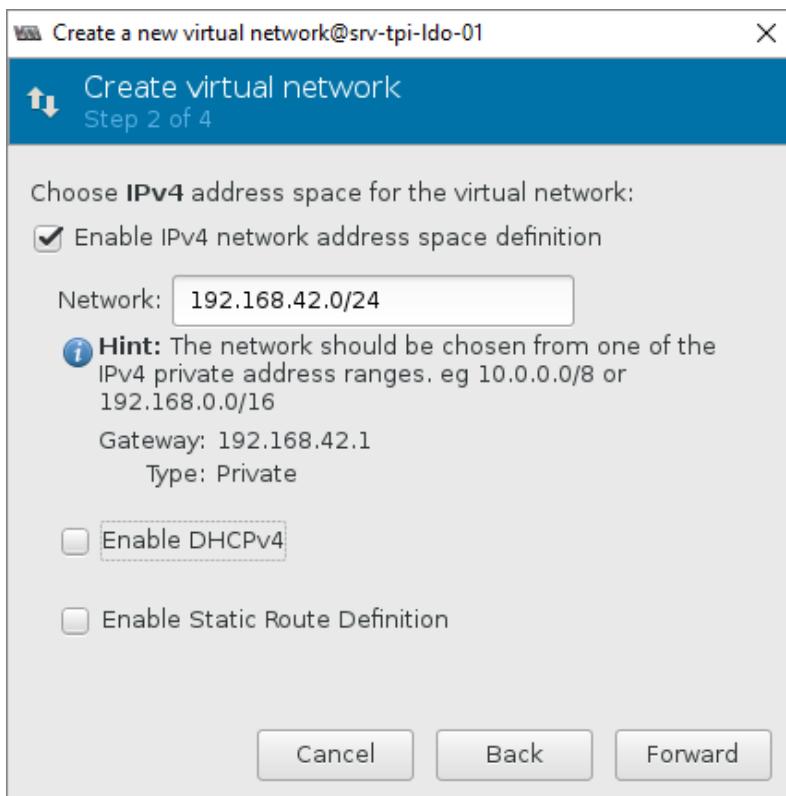
Je vais maintenant configurer le réseau que je vais utiliser pour la partie NAT de mon réseau comme expliqué ci-dessous.



Pour cela nous allons commencer par se connecter avec le « virt Manager » qui est la console de gestion de kvm.



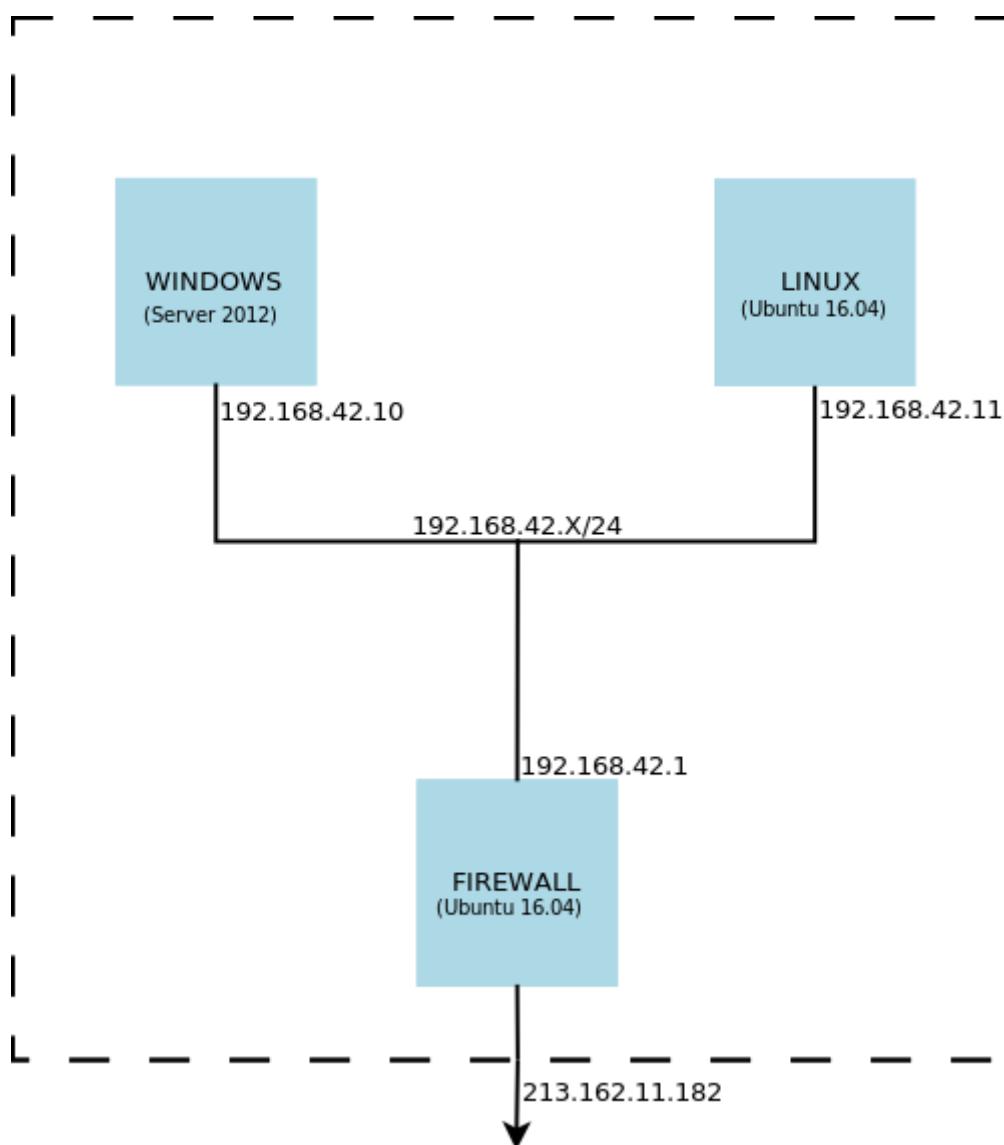
Ensuite je vais cliquer sur « edit » puis « Connection details ». Je vais ensuite cliquer sur l'onglet « Virtual Networks ». Je vais ensuite ajouter un réseau avec le bouton « + ». Je vais donner un nom à mon réseau.



Je définis ici mon réseau qui sera en 192.168.42.0/24. Je désactive le DHCP car je n'en ai pas besoin et il y en aura pas. Je n'active pas non plus ipv6 car pas besoin également. Suite à cette fenêtre je choisis de créer un réseau isolé.

## 5 Déploiement de l'environnement de test

### 5.1 Schéma de l'environnement



Dans chaque serveur j'ai 16Gb de RAM au total. Je vais distribuer 10Gb pour l'ensemble de mes vm afin de garder de la marge pour l'hôte. Le partitionnement sera fait ainsi : Firewall : 3Gb Linux : 2Gb Windows : 5Gb

### 5.2 Déploiement de la vm « Firewall »

Je vais commencer par l'installation et la configuration de cette vm car suite à cela je pourrai avoir du réseaux NAT derrière cette machine afin d'installer les suivantes.

### 5.2.1 Créations de la VM dans kvm

Je vais commencer par me connecter au srv-tpi-lde-01 car sur le 2 pas besoin vu que toutes les vms seront répliquées. Pour la créer, nous allons utiliser la commande virt-install qui fait partie du package virtinst. La commande entière se présente comme suit :

```
virt-install \
--virt-type=kvm \
--name=firewall \
--ram=3056 \
--vcpus=2 \
--cdrom=/zroot/boot/ubuntu-16.04.2-server-amd64.iso \
--os-variant ubuntu16.04 \
--hvm \
--network=bridge=br0,model=virtio \
--graphics vnc \
--disk path=/zroot/images/firewall.qcow2,size=40,bus=virtio,format=qcow2
```

Explications de chaque paramètre :

- Virt-type : L'hyperviseur choisis. Les exemples de choix sont kvm, qemu ou xen. Dans mon cas j'ai dû installer Qemu-kvm donc l'option à choisir est KVM.
- Name : Définit le nom de la machine virtuelle. Dans mon cas firewall car ce sera sa fonction
- Ram : Définit la quantité de RAM en Mo
- Vcpus : Définit le nombre de cpu virtuel pour cette machine
- Cdrom : Définit le Path du CD que nous allons utiliser pour l'installation
- Os-variant : Définit la version de l'os que nous allons utiliser. Dans mon cas un ubuntu 16.04
- Hvm : Demande l'utilisation de la virtualisation complète, si les possibilités de virtualisation intégrales et paramétrables sont disponibles sur l'hôte.
- Network : Définit le type de réseau que nous voulons ainsi que le modèle
- Graphics : Définit le type d'interface graphique que nous voulons connecter à notre virt-manager.
- Disk-path : Définit le Path du disk ainsi que sa taille et son format

### 5.2.2 Installation de la vm

L'installation et la configuration de base sera la même que pour les serveurs. Se référer au chapitre 2.5.1 et 2.5.2.

### 5.2.3 Configuration des vm guest agent

Nous allons nous connecter sur chaque hyperviseur afin de mettre en place la configuration pour les qemu-guest-agent. Pour ce faire, je vais commencer par créer les dossiers nécessaires avec :

```
mkdir -p /var/lib/libvirt/qemu/channel/target  
chown -R libvirt-qemu:kvm /var/lib/libvirt/qemu/channel
```

Ensuite je vais éditer le fichier */etc/apparmor.d/abstraction/libvirt-qemu* et y ajouter cette ligne à la fin du document :

```
/var/lib/libvirt/qemu/channel/target/* rw,
```

Ensuite, je vais me connecter à ma vm firewall. Je vais installer les qemu-guest-agent avec :

```
apt install qemu-guest-agent
```

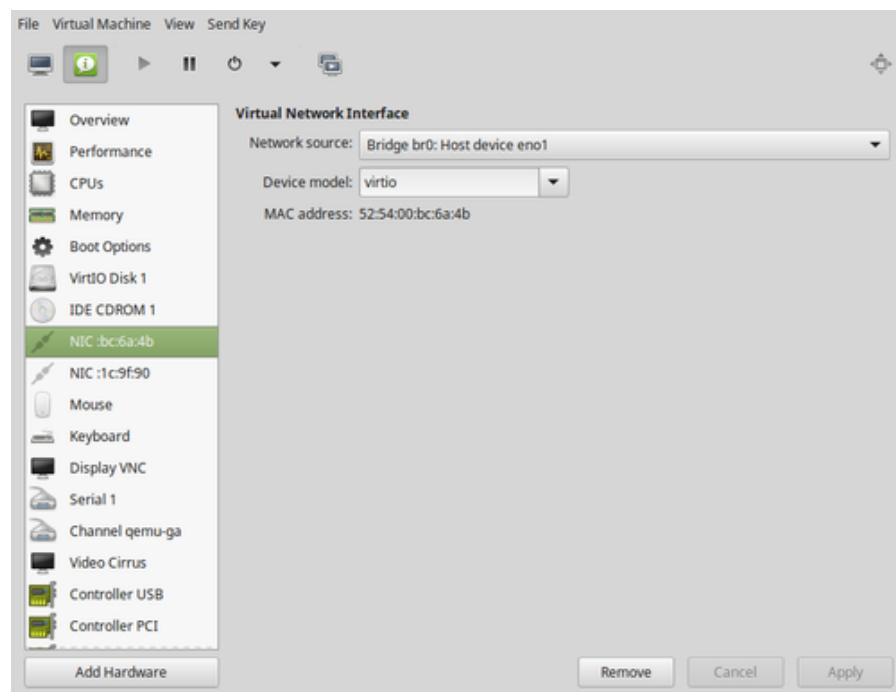
Maintenant, je vais éteindre ma vm. Edit ma vm avec : virsh edit firewall. Ajouter dans la partie "device":

```
<channel type="unix">  
  <source mode="bind"/>  
  <target type="virtio" name="org.qemu.guest_agent.0"/>  
</channel>
```

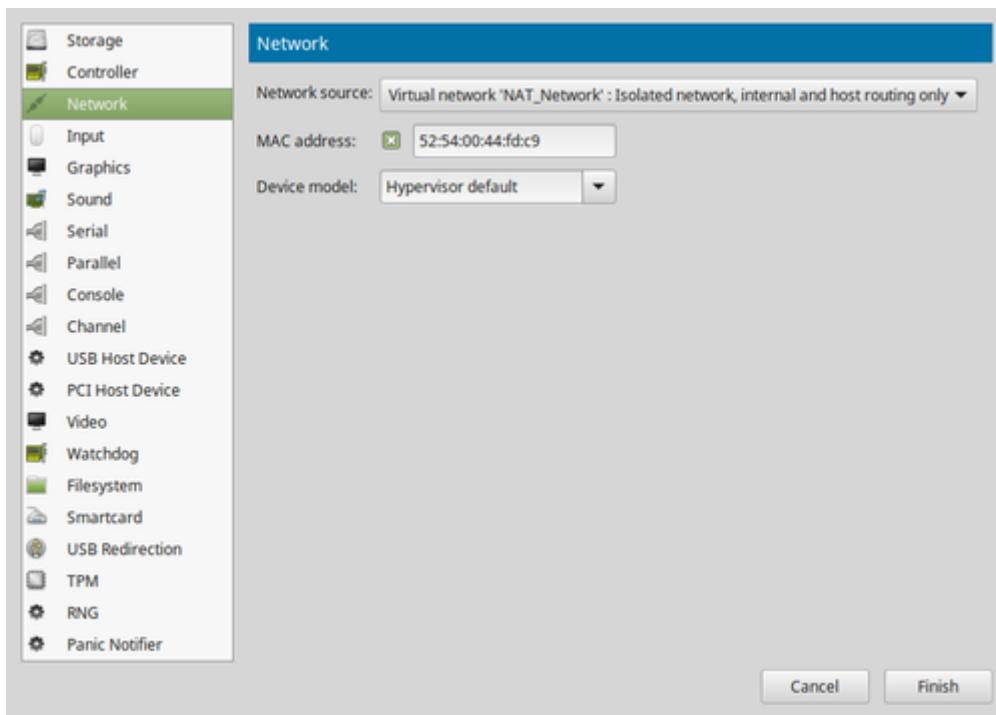
Finalement, je vais redémarrer ma vm

### 5.2.4 Crédation du réseau NAT

Nous allons maintenant créer le réseau NAT afin de connecter nos vm à ce réseau. Nous allons aller dans les paramètres de la vm afin de rajouter une carte réseau.



Je vais maintenant rajouter la carte réseau en appuyant sur « Add Hardware »



Maintenant, nous allons nous connecter à la vm afin de configurer la carte réseau Nous allons assigner l'adresse ip 192.168.42.1 vu que ce sera la gateway. La configuration de la carte réseau de la machine "Firewall" ressemble à cela :

```
auto ens8
iface ens8 inet static
    address 192.168.42.1
    netmask 255.255.255.0
    network 192.168.42.0
    broadcast 192.168.42.255
```

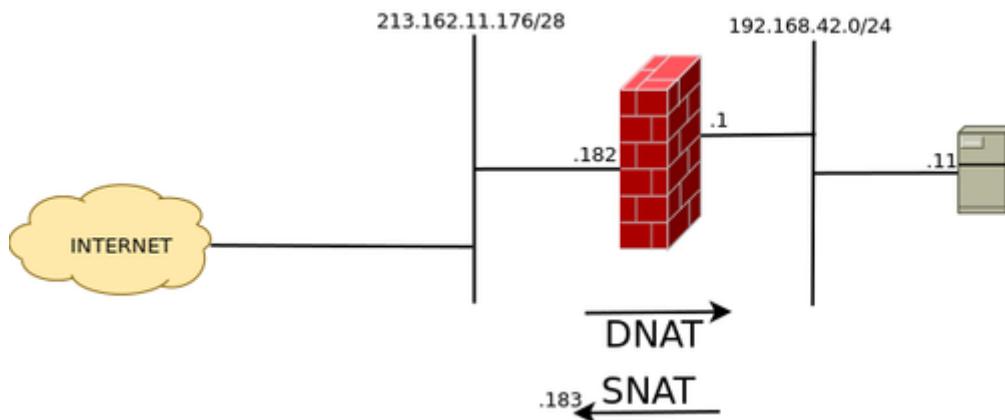
La configuration de l'interface n'a pas de Gateway vu que nous allons après routeur ce trafic avec l'aide de iptable. Je vais maintenant créer 2 interfaces virtuelles dans /etc/network/interfaces afin de les assigner plus tard au vm qui seront derrière le firewall. La configuration est comme ça :

```
# Config for machine Windows
auto ens3:0
iface ens3:0 inet static
    address 213.162.11.183
    netmask 255.255.255.240

# Config for machine Linux
auto ens3:1
iface ens3:1 inet static
    address 213.162.11.184
    netmask 255.255.255.240
```

Quand on crée une interface virtuelle sur une machine unix, on donne un numéro après l'interface physique. On ne met pas de Gateway car il utilise la même que la physique. Suite à

ça, je vais aller modifier mon iptable afin de router le traffic de chaque ip. Cf. schéma ci-dessous :



Dans `/etc/iptables/rules.v4` je vais rajouter les règles ci-dessous dans le module nat de iptables afin de définir le snat et dnat. Il faudra le faire pour chaque ip supplémentaire rajouter.

```

-A PREROUTING -d 213.162.11.183 -j DNAT --to-destination 192.168.42.11
-A POSTROUTING -s 192.168.42.11 -j SNAT --to 213.162.11.183
    
```

## 5.3 Déploiement de la vm « Linux »

### 5.3.1 Déploiement de la vm

Le déploiement sera le même que pour la vm précédente. Se référer au chapitre 5.2.1.

### 5.3.2 Installation de la vm

L'installation et la configuration de base sera la même que pour les serveurs. Se référer au chapitre 2.5.1 et 2.5.2.

### 5.3.3 Configuration particulière

La seule chose qu'il y à faire sur cette vm est le fait de lui attribuer l'adresse ip fixe 192.168.42.11. En sachant que là, j'ai un Windows 2012 R2 et les agents qemu ne sont pas compatible avec cette version. Ils sont uniquement compatibles jusqu'à Windows server 2008.

## 5.4 Déploiement de la vm « Windows »

### 5.4.1 Déploiement de la vm

Le déploiement sera la même que pour la vm précédente. Se référer au chapitre 5.2.1. A la différence de remplacer l'iso par celle de Windows.

### 5.4.2 Installation de la vm

Sélectionnez la langue désirée, et cliquez sur suivant :



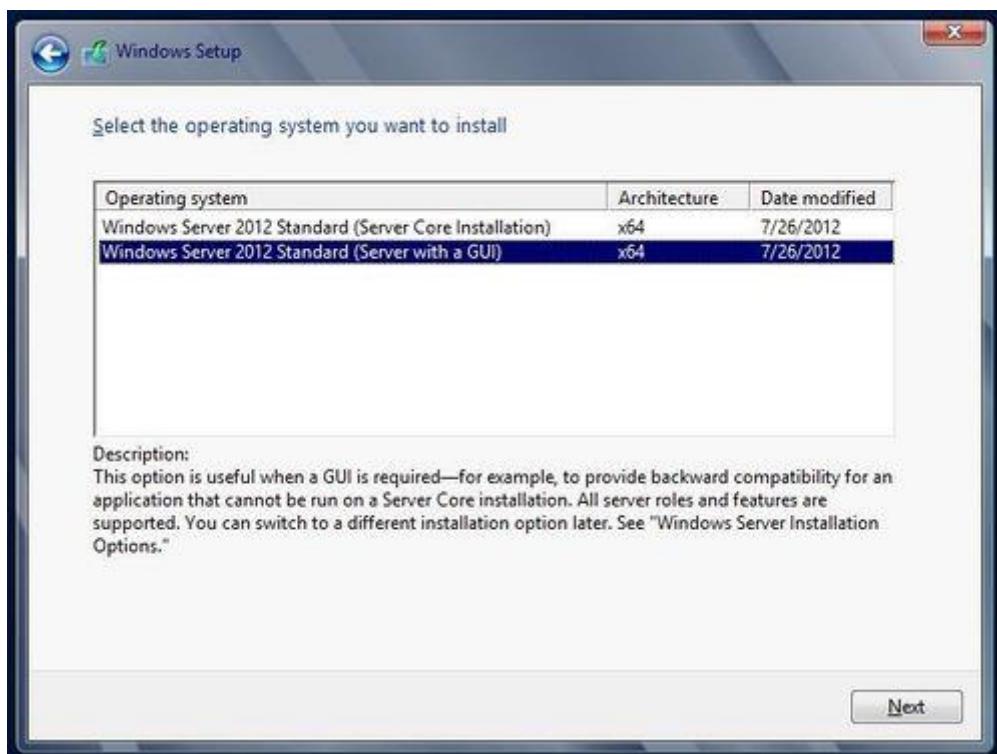
Cliquez sur Install now :



Entrez le numéro de série :



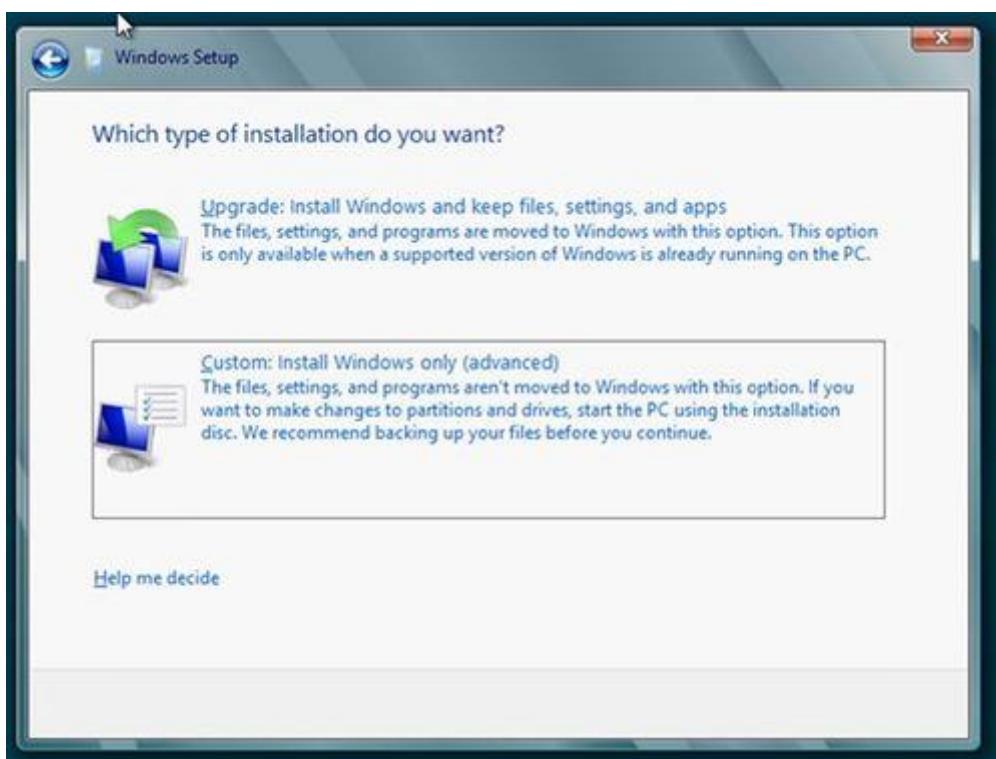
Sélectionnez Windows Server 2012 Server with a GUI et cliquez sur suivant :



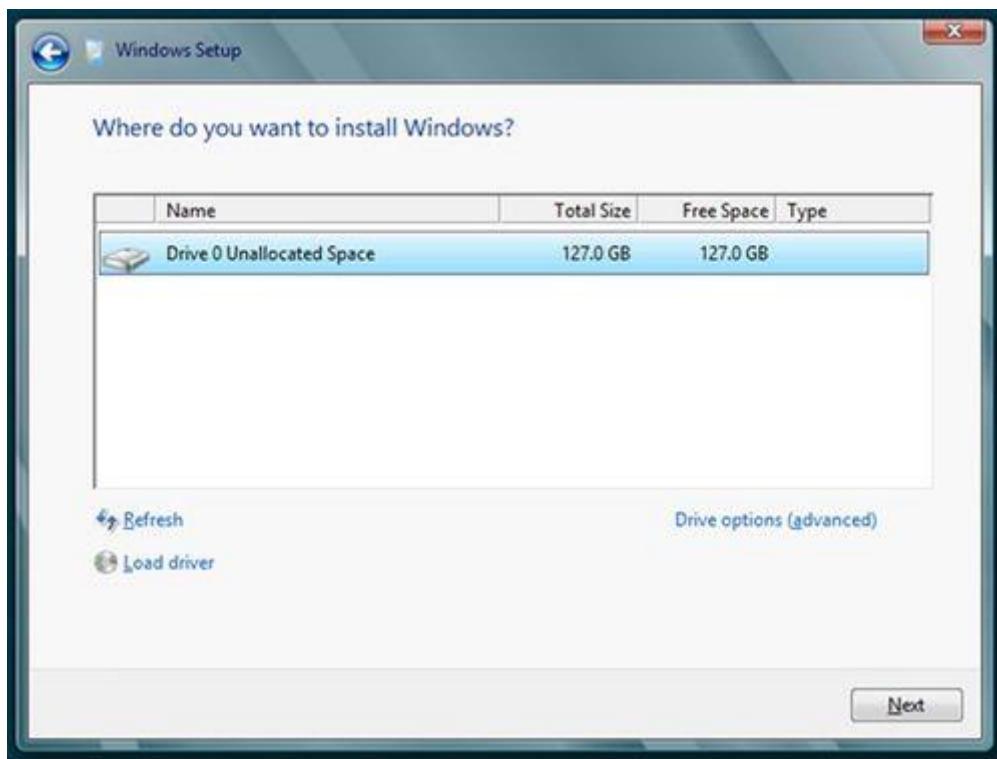
Acceptez la licence et cliquez sur suivant :



Sélectionnez custom Install :



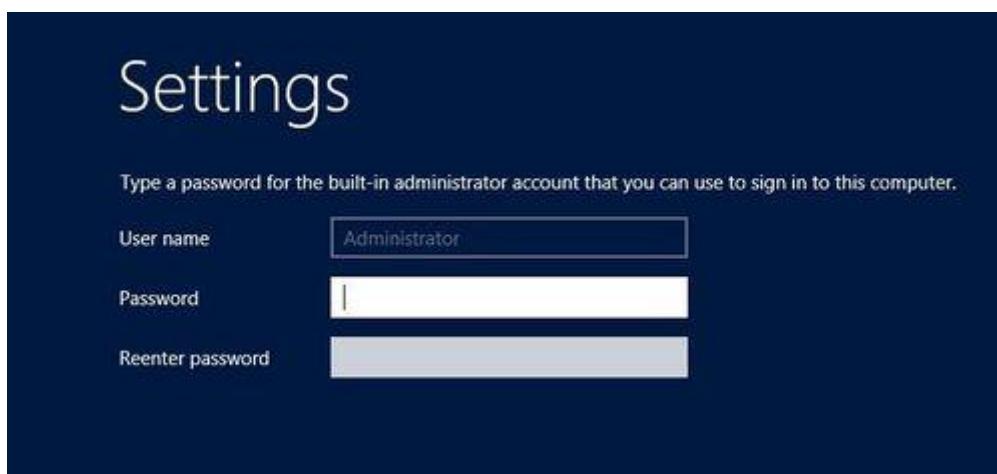
Sélectionnez le disque et la partition sur lesquels vous souhaitez installer le système (minimum 32 Go) puis cliquez sur Suivant :



L'installation commence :



A la fin de l'installation, vous devez définir le mot de passe du compte :

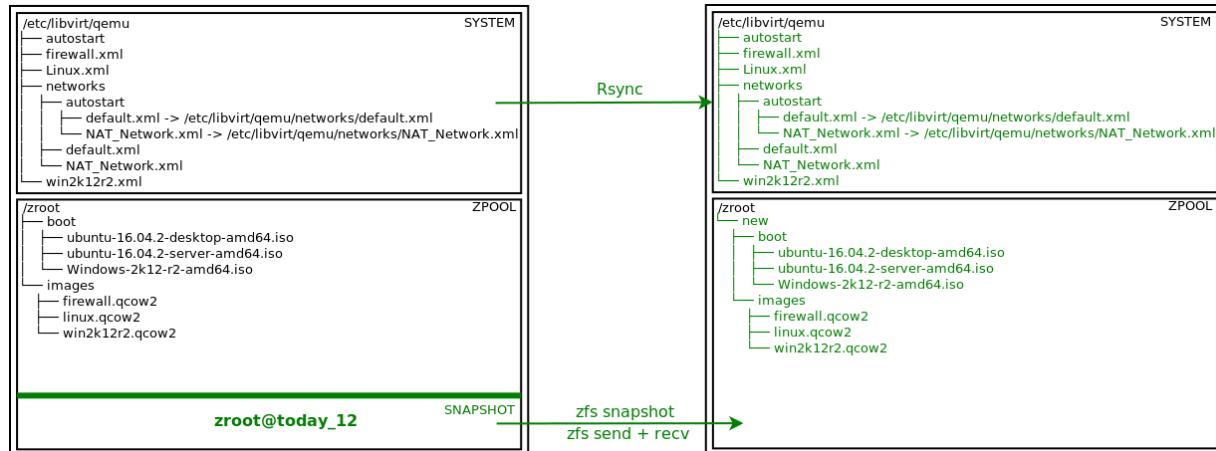


### 5.4.3 Config particulière

La seule chose qu'il y à faire sur cette vm est, le fait de lui attribuer l'adresse ip fixe 192.168.42.10.

## 6 Mise en place de la réPLICATION des données

La réPLICATION entre les 2 serveurs se fera comme le schéma ci-dessous :



J'ai dû utiliser du rsync en parallèle de zfs car la partie de configuration des vm comme vous le voyez ci-dessus se trouve dans la partition système. Il n'y pas de moyens de bouger cette partie à un autre endroit. J'ai également dû faire de l'incrémentiel pour la partie zfs pour éviter chaque heure de rebalancer toute les données d'un côté à l'autre. Cette partie sont les lignes 49 à 83. Du script « replicationZFS ».

### 6.1 Généralité

Le script qui fera cette réPLICATION se trouve sur le srv-tpi-lde-01.dsprod.ch. Dans le répertoire /usr/bin avec le nom replicationKVM. Ce script est également rentré dans le crontab du server srv-tpi-lde-01 de la façon suivante :

```
0 */1 * * * PATH=$PATH:/sbin && export PATH && /usr/bin/replicationZFS
```

J'ai dû rajouter le path /sbin car zfs se trouve là-dedans. Ce script et en annexe de mon rapport.

### 6.2 Fonctions de chaque script

- replicationZFS : C'est le script principal celui qui est dans le crontab du serveur.
- shutdownKVM : Ce script est utile dans le cas où je voudrais éteindre toutes les vm d'un coup
- startKVM : Ce script est utile dans le cas où je voudrais démarrer toutes les vm d'un coup
- delOldSnap : Ce script est utile dans le cas où je veux forcer à effacer les anciens snapshot
- failoverNow : Ce script est utile dans le cas où je voudrais basculer manuellement d'un host à l'autre

Tous les scripts cités ci-dessus sont en annexe de mon dossier ou sur mon repo github [ICI](#).

## 6.3 Mise en place des scripts

Les scripts énumérés ci-dessus, se trouve sur mon repo Github comme mentionné ci-dessus. Pour mettre en place ces scripts sur le serveur nous, allons les placer dans /usr/bin car ce chemin se trouve dans le PATH. Comme ça nous pouvons exécuter les commandes depuis n'importe où sur le serveur. Pour ce faire nous allons exécuter cette commande :

```
git clone git@github.com:Cormoran96/TPI-KVM-ZFS-DRP.git /usr/bin/
```

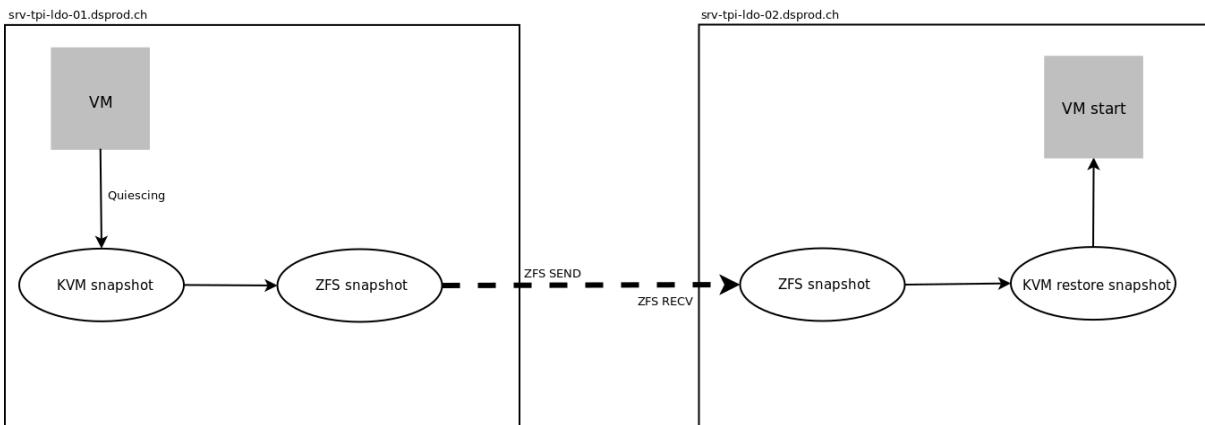
## 6.4 Retour après le failover

Suite à un failover manuelle le serveur master change. Le slave devient master. Afin de garder la réPLICATION. Il faut :

1. Réparer le premier serveur
2. Mettre dans le cron du deuxième serveur cette ligne  
`0 */1 * * * PATH=$PATH:/sbin && export PATH &&  
/usr/bin/replicationZFS`
3. La réPLICATION reprend mais a l'inverse du serveur 2 au 1

## 6.5 Amélioration possible

S'il me reste du temps à la fin de mon travail, je planifierai différemment les tâches de snapshot zfs. Faire un snapshot KVM de chaque vm afin de d'améliorer l'intégrité des données.



## 7 Test et validation du processus

### 7.1 Liste des tests

- Failover manuel en éteignant les machines à la main sur le premier serveur puis rallumer les vm sur le nouveau serveur
- Failover manuel en utilisant le script failoverNow. Vérifier suite à ça si les vm ont bien redémarré ainsi que toutes les vm soient de nouveau joignables.
- Failover manuel en utilisant le script failoverNow. En ajoutant le fait d'écrire des fichiers à ce moment-là afin de tester l'intégrité de la chose.
- Test de perte physique du premier serveur

### 7.2 Failover manuel simple

#### 7.2.1 Contexte

Je veux valider le fait que de façon simple le failover entre les 2 machines fonctionne.

#### 7.2.2 Action à réaliser

Éteindre les vm :

- Se connecter sur le serveur srv-tpi-ldo-01
- Utiliser le virt-manager
- Éteindre toutes les vm avec le virt-manager
- Se déconnecter du serveur srv-tpi-ldo-01

Allumer les vm :

- Se connecter sur le serveur sr-tpi-ldo-02
- Utiliser le virt-manager
- Allumer toutes les vm avec le virt-manager
- Se déconnecter du serveur

Contrôle des vm :

- Pinger toute les ip de toutes les vm voir si elles répondent
- Se connecter sur chaque vm avec soit SSH soit Remote Desktop en fonction de l'os

#### 7.2.3 Résultats attendus

Toutes les vm sont censés s'éteindre comme il le faut sans erreur. Ensuite sur le 2<sup>ième</sup> serveur, les vm dans le virt-manager doivent être identiques au premier. Les vm doivent s'allumer parfaitement sans erreur. Je dois pouvoir pinger les vm quasiment instantanément après leur boot.

## 7.2.4 Résultats obtenus

Aucun problème rencontré pour se connecter sur le premier serveur. Utilisation du virt-manager sans problème. Toutes les vm ont démarré correctement. Toutes les vm répondent aux ping.

```
Pinging srv-tpi-lde-02.dsprod.ch [213.162.11.180] with 32 bytes of data:  
Reply from 213.162.11.180: bytes=32 time=3ms TTL=53  
Reply from 213.162.11.180: bytes=32 time=3ms TTL=53
```

```
Pinging fw-tpi-lde-01.dsprod.ch [213.162.11.182] with 32 bytes of data:  
Reply from 213.162.11.182: bytes=32 time=3ms TTL=53  
Reply from 213.162.11.182: bytes=32 time=3ms TTL=53
```

```
Pinging win.tpi.dsprod.ch [213.162.11.183] with 32 bytes of data:  
Reply from 213.162.11.183: bytes=32 time=4ms TTL=116  
Reply from 213.162.11.183: bytes=32 time=4ms TTL=116
```

```
Pinging linux.tpi.dsprod.ch [213.162.11.184] with 32 bytes of data:  
Reply from 213.162.11.184: bytes=32 time=4ms TTL=52  
Reply from 213.162.11.184: bytes=32 time=4ms TTL=52
```

**FONCTIONNALITE VALIDE : OUI**

## 7.3 Failover manuel à l'aide d'un script

### 7.3.1 Contexte

Je veux valider le fait qu'avec le script FailoverNow le failover entre les 2 machines fonctionne.

### 7.3.2 Action à réaliser

Srv-tpi-lde-01 :

- Se connecter sur le serveur srv-tpi-lde-01
- Lancer la commande faileoverNow
- Se déconnecter du serveur srv-tpi-lde-01

Contrôle des vm :

- Pinger toute les ip de toutes les vm voir si elles répondent
- Se connecter sur chaque vm avec soit SSH soit Remote Desktop en fonction de l'os

### 7.3.3 Résultats attendus

Toutes les vm sont censées s'éteindre comme il le faut sans erreur. Ensuite sur le 2<sup>ième</sup> serveur, les vm dans le virt-manager doivent être identique au premier. Les vm doivent s'allumer parfaitement sans erreur. Je dois pouvoir pinger les vm quasiment instantanément après leur boot.

### 7.3.4 Résultats obtenus

Aucun problème pour se connecter sur le serveur srv-tpi-lde-01. La commande est bien passée. Les logs sont ci-dessous :

```
May 18 07:31:01 srv-tpi-lde-01 CRON[8954]: (root) CMD (PATH=$PATH:/sbin &&
export PATH && /usr/bin/replicationZFS)
May 18 07:31:01 srv-tpi-lde-01 replicationZFS: [ OK ] New snapshot exist !
May 18 07:31:01 srv-tpi-lde-01 replicationZFS: [ OK ] Older 1 hour snapshot
exist !
May 18 07:31:01 srv-tpi-lde-01 replicationZFS: [ INFO ] Starting sending
incremental snapshot...
May 18 07:31:03 srv-tpi-lde-01 replicationZFS: [ OK ] destination have good
receive snapshot !
May 18 07:31:04 srv-tpi-lde-01 replicationZFS: [ INFO ] Sending xml files
for KVM...
May 18 07:31:04 srv-tpi-lde-01 replicationZFS: [ INFO ] Modification of xml
for new zroot/new ...
May 18 07:31:04 srv-tpi-lde-01 replicationZFS: [ INFO ] Restarting kvm for
apply change...
May 18 07:31:05 srv-tpi-lde-01 replicationZFS: [ INFO ] Delete:
zroot/new@today_04
May 18 07:31:05 srv-tpi-lde-01 replicationZFS: [ INFO ] Delete:
zroot@today_05
```

Toutes les vm répondent aux ping.

```
Pinging srv-tpi-lde-02.dsprod.ch [213.162.11.180] with 32 bytes of data:  
Reply from 213.162.11.180: bytes=32 time=2ms TTL=53  
Reply from 213.162.11.180: bytes=32 time=2ms TTL=53
```

```
Pinging fw-tpi-lde-01.dsprod.ch [213.162.11.182] with 32 bytes of data:  
Reply from 213.162.11.182: bytes=32 time=4ms TTL=53  
Reply from 213.162.11.182: bytes=32 time=4ms TTL=53
```

```
Pinging win.tpi.dsprod.ch [213.162.11.183] with 32 bytes of data:  
Reply from 213.162.11.183: bytes=32 time=2ms TTL=116  
Reply from 213.162.11.183: bytes=32 time=2ms TTL=116
```

```
Pinging linux.tpi.dsprod.ch [213.162.11.184] with 32 bytes of data:  
Reply from 213.162.11.184: bytes=32 time=3ms TTL=52  
Reply from 213.162.11.184: bytes=32 time=3ms TTL=52
```

Il n'y a eu aucun problème pour se connecter au vm

**FONCTIONNALITE VALIDE : OUI**

## 7.4 Failover manuel avec de l'écriture sur les vm

### 7.4.1 Contexte

Je veux valider le fait que de façon générale le failover entre les 2 machines fonctionne alors qu'il y a de l'écriture sur les machines à ce moment-là. Refaire le test sur 5 itération afin de vraiment vérifier le résultat.

### 7.4.2 Action à réaliser

Initialiser l'écriture :

- Se connecter sur le serveurs Windows lancer dummy en créant un fichier de 5GB
- Se connecter sur les 2 serveur linux lancer « dd if=/dev/urandom of=newfile bs=1000000 count=5000 » afin de créer un fichier de 5 GB

Éteindre les vm :

- Se connecter sur le serveur srv-tpi-ldo-01
- Lancer la commande failoverNow
- Se déconnecter du serveur srv-tpi-ldo-01

Contrôle des vm :

- Pinger toutes les ip de toutes les vm voir si elles répondent
- Se connecter sur chaque vm avec soit SSH soit Remote Desktop en fonction de l'os
- Lancer un fsck de chaque disque pour les serveurs linux
- Lancer un CHKDSK du disque de la vm de windows

### 7.4.3 Résultats attendus

Toutes les vm sont censées s'éteindre comme il le faut sans erreur. Ensuite sur le 2<sup>ième</sup> serveur les vm dans le virt-manager doivent être identiques au premier. Les vm doivent s'allumer parfaitement sans erreur. Je dois pouvoir pinger les vm quasiment instantanément après leur boot. Les checkdisk ne doivent donner aucune corruption.

### 7.4.4 Résultats obtenus

Aucun problème rencontré sur le fait de lancer les commandes de génération de fichiers. J'ai pu me connecter sans problème sur le serveur. Le script c'est déroulé sans erreur : la même chose qu'au test précédent. Toutes les vm ping. Les check disque sont sans aucun problème. Sur les 5 itération les résultats sont les mêmes.

**FONCTIONNALITE VALIDE : OUI**

## 7.5 Test de perte physique d'un serveur

### 7.5.1 Contexte

Je veux valider le fait que si un serveur physique s'éteint ou s'il y a plus de réseau le second peut reprendre la main.

### 7.5.2 Action à réaliser

Srv-tpi-lde-01 :

- Tirer la prise réseau

Allumer les vm :

- Se connecter sur le serveur sr-tpi-lde-02
- Utiliser le virt-manager
- Allumer toutes les vm avec le virt-manager
- Se déconnecter du serveur

Contrôle des vm :

- Pinger toutes les ip de toutes les vm voir si elles répondent
- Se connecter sur chaque vm avec soit SSH soit Remote Desktop en fonction de l'os

### 7.5.3 Résultats attendus

Le serveur 1 ne doit plus être joignable. Ensuite sur le 2<sup>ième</sup> serveur les vm dans le virt-manager doivent être identiques au premier. Les vm doivent s'allumer parfaitement sans erreur. Je dois pouvoir pinger les vm quasiment instantanément après leur boot.

### 7.5.4 Résultats obtenus

Il y a eu aucun problème sur le fait de se connecter au 2<sup>ième</sup> serveur. Le serveur 1 n'est plus joignable. Les vm se sont allumées sans erreur et les ping fonctionnent.

**FONCTIONNALITE VALIDE : OUI**

## 8 Complément

### 8.1 zRam

J'ai découvert cette fonctionnalité géniale de Linux par hasard. ZRAM est un périphérique de bloc (c'est-à-dire un « disque ») où le contenu est compressé et stocké en mémoire, ce qui le rend plutôt banal et peu intéressant. La manière dont il est habituellement configuré est comme espace de swap ... alors en effet, le zRAM sert à compresser la mémoire normale, à utiliser du processeur plutôt que la mémoire.

Pour le mettre en place, je vais devoir charger le module zRAM et choisir combien de cœur processeur je veux utiliser en même temps. Je vais décider d'utiliser tous les cœurs en même temps pour gagner du temps de compression. Pour faire cela, je vais ajouter la ligne suivante dans mon /etc/rc.local

```
/sbin/modprobe zram zram_num_devices=$(cat /proc/cpuinfo | grep processor | wc -l)
```

### 8.2 Backup de mon système

Si à l'avenir mon système venait à être mis en production. Il faudrait prévoir un plan de backup car ni le RAID, ni le HA n'est un backup. Le RAID protège contre la faille matérielle mais ne protège pas en cas de :

- Corruption de fichier
- Erreur humaine (suppression de fichier par erreur)
- Dommage catastrophique (De l'eau qui tombe sur les serveurs)
- Virus ou malware

Pour le HA, les arguments ci-dessus sont valables. Du coup, pour le plan de backup il faudrait backup, prioritairement 2 dossiers. Ces 2 dossier sont :

- /etc/libvirt/qemu (Car c'est là que sont stockés les xml de chaque vm)
- /zroot (Car c'est là que sont stockés les disques durs de chaque vm)

### 8.3 Monitoring de mon système

Si à l'avenir mon système venait à être mis en mis en production. Il faudrait prévoir un plan afin de monitorer mon système. Nous pouvons réaliser les check suivants afin de tester les valeurs du système :

- Check en faisant un zfs list -t snapshot chaque heure afin de tester si les snapshot les plus récents sont bien sur le nouveau serveur
- Check de disque afin de tester la place restante sur le zpool
- Check l'état du mirror du Raidz afin de voir si un disque crash
- Check de la vm Firewall car tout le trafic sort par cette vm
- Check du temps des snapshot

## 9 Conclusion

---

Pour conclure mon TPI, je vais expliquer ce qui m'a plus et les points sur lesquels j'ai eu plus de problème. Au début, j'étais super motivé à commencer ce travail qui résume mes 4 ans de formations. Je connaissais toute les technologies que j'allais utilisé durant ce projet, mais je ne les avais jamais combinés ensemble sur un seul et même projet. Cette partie-là du coup me faisait un peu peur. J'apprécie les nouveaux challenges donc j'ai adoré rechercher les infos manquantes afin de faire cohabiter les différentes technologies ensemble.

Une fois cette partie de recherche effectuée, le projet s'est déroulé sans accroche. Ce fut très plaisant de ne pas rester bloquer sur un des aspects.

### 9.1 Connaissance

Ce travail m'a permis d'acquérir des nouvelles compétences sur Ubuntu ainsi qu'avec KVM. Comme par exemple la gestion du bridge réseau qui pour moi était une notion un peu vague.

### 9.2 Regard critique

Je suis dans l'ensemble content de mon travail. J'aurais aimé plus approfondir certains aspects, mais par manque de temps je n'ai pas pu le faire. Quand on y pense ont peu toujours faire mieux mais comme disait Lord Baden-Powell, "L'ambition de faire le bien est la seule qui compte.". Je me suis donner à 100% dans ce projet et je suis fier du travail que j'ai sorti.

## 10 Sources

### 10.1 Articles en lignes

- StoneyCloud (en ligne), Qemu Guest Agent Integration. 12 Mars 2015. Disponible à l'adresse : [http://wiki.stoney-cloud.org/wiki/Qemu\\_Guest\\_Agent\\_Integration](http://wiki.stoney-cloud.org/wiki/Qemu_Guest_Agent_Integration)
- RedHat (en ligne), Freeze Windows. 9 Juin 2015. Disponible à l'adresse : <https://www.redhat.com/archives/libvirt-users/2015-June/msg00036.html>
- Github (en ligne), FAQ zfs. 27 Avril 2017. Disponible à l'adresse : <https://github.com/zfsonlinux/zfs/wiki/faq#performance-considerations>
- Blog O'Matty (en ligne), Disabling access time (atime) updates on ZFS file systems. 25 Juillet 2006. Disponible à l'adresse : <http://prefetch.net/blog/index.php/2006/07/25/disabling-access-time-atime-updates-on-zfs-file-system/>
- Open-zfs (en ligne), Performance tuning. 22 Juin 2016. Disponible à l'adresse : [http://open-zfs.org/wiki/Performance\\_tuning#Compression](http://open-zfs.org/wiki/Performance_tuning#Compression)
- Ubuntu (en ligne), KVM/Networking. 15 Avril 2016. Disponible à l'adresse : <https://help.ubuntu.com/community/KVM/Networking>
- Libvirt (en ligne), Networking. 15 Mai 2017. Disponible à l'adresse : <https://wiki.libvirt.org/page/Networking>
- Oracle (en ligne), ZFS administration Guide. 2010. Disponible à l'adresse : <http://docs.oracle.com/cd/E19253-01/819-5461/gaztn/>

### 10.2 Magazine

- Linux Magazine, 2017. Kernel le guide pour plonger au cœur de votre système GNU/LINUX. Hors-série N°87. Disponible à l'adresse : <https://boutique.ed-diamond.com/de-retour-en-kiosque/1088-gnulinux-magazine-hs-87.html>

# 11 Annexes

---

```

1 #!/bin/bash -
2 #title      :replicationzfs
3 #description :This script copy all data with ZFS send and rsync
4 #author     :Lucas Dousse
5 #date       :20170512
6 #version    :1.1.4
7 #usage      :./replicationzfs
8 #notes      :
9 #bash_version :4.3.46(1)-release
10 =====
11
12
13 # Logger
14 exec 1> >(logger -s -t $(basename $0)) 2>&1
15
16 ######
17 #      VARIABLES      #
18 #####
19 srv_dest="srv-tpi-ldo-02.dsprod.ch" # Serveur de destination
20
21 src_pool="zroot"                   # Pool ZFS de départ
22 dest_pool="zroot/new"              # Pool ZFS de réception sur le serveur distant
23
24 snap_prefix="today_"
25 snap_time=$(date '+%H')
26 snap_time_old=0
27 let "snap_time_old=${snap_time#0}-1"
28 snap_new="$src_pool@$snap_prefix$nap_time"           # Forge du nom du nouveau
snapshot
29
30 #####
31 #      TEST      #
32 #####
33 if [ $snap_time_old -lt 10 ]          # Test si l'heure est inférieure a 10 heure alors 0X
then
34     snap_time_old=0$nap_time_old"
35 fi
36
37
38 if [ $snap_time == 00 ]                # Test si l'heure et inférieur a minuit alors 23
then
39     snap_time_old=23
40 fi
41
42
43 snap_old="$src_pool@$snap_prefix$nap_time_old"        # Forge du nom de l'ancien
snapshot
44
45
46 ######
47 #      FUNCTIONS      #
48 #####
49 function zfs_snap() {                  # Fonction de départ qui crée le
snapshot et vérifie que il existe un plus vieux pour faire de l'incrémental
50     zfs snapshot -r "$snap_new"          # Create the snapshot
51     zfs list -t snapshot > /tmp/snapshot.txt      # Export de la liste des
snapshot
52
53     if grep -Fq "$snap_new" /tmp/snapshot.txt      # Si | test si le nouveau
snapshot a été fait
54         then
55             echo "[ OK ] New snapshot exist !"
56             if grep -Fq "$snap_old" /tmp/snapshot.txt      # Si | test si le snapshot
vieux de 1 heure existe
57                 then
58                     echo "[ OK ] Older 1 hour snapshot exist !"
59                     rm -rf /tmp/snapshot.txt
60                     zfs_send                                # Si oui | Il va aller dans
la fonction zfs_send, afin de faire de l'incrementiel
61             else
62                 echo "[ WARNING ] Older 1 hour snapshot don't exist !"
63                 rm -rf /tmp/snapshot.txt
64                 zfs_send_first                         # Si non | Il va créer le
premier backup full
65         fi

```

```

66     else
67         echo "[ ERROR ] unknown"
68         rm -rf /tmp/snapshot.txt
69         exit -1;
70     fi
71 }
72
73 function zfs_send_first() {                                     # Fonction de backup full avec zfs
74     send
75     echo "[ INFO ] Starting sending first snapshot..."
76     zfs send "$snap_new" | ssh "$srv_dest" zfs recv "$dest_pool"      # Envoie tout le
77     contenu sur le serveur distant en un bloc
78     zfs_control_recv
79 }
80
81 function zfs_send() {                                         # Fonction de backup incrementiel
82     avec zfs send
83     echo "[ INFO ] Starting sending incremental snapshot..."
84     zfs send -R -i "$snap_old" "$snap_new" | ssh "$srv_dest" zfs recv "$dest_pool"
85     # Envoie le diff du vieux snapshot sur le nouveaux serveur
86     zfs_control_recv
87 }
88
89 function zfs_control_recv() {                                # Fonction de control que le
90     snapshot envoyer aie bien été reçu de l'autre coté
91     ssh "$srv_dest" 'zfs list -t snapshot > /tmp/snapshot.txt'
92     ssh "$srv_dest" 'if grep -Fq "$snap_new" /tmp/snapshot.txt; then echo "[ OK ]"
93     destination have good receive snapshot !; else exit -1; fi'
94     ssh "$srv_dest" 'rm -rf /tmp/snapshot.txt'
95     send_xml
96 }
97
98 function send_xml() {                                       # Fonction d'envoie des fichier de
99     config des vm
100    echo "[ INFO ] Sending xml files for KVM..."
101    rsync -ra /etc/libvirt/qemu/ "$srv_dest":/etc/libvirt/qemu/
102    modifiaction_xml
103 }
104
105 function modifiaction_xml() {                             # Modification des xml, car nouveau
106     chemin n'est pas le même que l'ancien
107     echo "[ INFO ] Modification of xml for new $dest_pool ..."
108     ssh "$srv_dest" "sed -i 's/zroot/zroot\\new/g' /etc/libvirt/qemu/*.xml"
109     restart_kvm
110 }
111
112 function restart_kvm() {                                 # Fonction de restart de kvm afin de
113     prendre en compte les modifs
114     echo "[ INFO ] Restarting kvm for apply change..."
115     ssh "$srv_dest" service libvirt-bin restart
116     del_old_snap
117 }
118
119 function del_old_snap() {                               # Fonction de suppression des snapshots
120     zfs list -H -t snapshot -o name -S creation | sed '1d' | sed '1d' >
121     /tmp/snap_srv1.txt # Liste les snapshot et supprime les 2 derniers du fichiers
122     ssh "$srv_dest" "/usr/bin/delOldSnap"                  # Execution du
123     script sur le server distant
124
125     if [[ -s /tmp/snap_srv1.txt ]]; then                   # Test si le fichier
126         n'est pas vide
127         while IFS='`' read -r line || [[ -n "$line" ]]; do   # Boucle pour lire
128             et supprimer les vieux snapshot
129             echo "[ INFO ] Delete: $line"
130             zfs destroy "$line"
131             done < "/tmp/snap_srv1.txt"
132             rm -rf /tmp/snap_srv1.txt
133         else
134             echo "[ OK ] snapshot is already delete"
135             rm -rf /tmp/snap_srv1.txt
136             exit 0;
137         fi

```

```
126 }
127
128 ######
129 #      EXECUTION      #
130 ######
131 zfs_snap
132
133 #####
134 #      UNSET      #
135 #####
136 unset snap_time_old
137 unset srv_dest
138 unset src_pool
139 unset dest_pool
140 unset snap_prefix
141 unset snap_time
142 unset snap_new
143 unset snap_old
144
```

```

1 #!/bin/bash -
2 #title :failoverNow
3 #description :This script copy all data with ZFS send and rsync for now
4 #author :Lucas Dousse
5 #date :20170516
6 #version :0.1
7 #usage :./failoverNow
8 #notes :
9 #bash_version :4.3.46(1)-release
10 #=====
11
12
13 # Logger
14 exec 1>>(logger -s -t $(basename $0)) 2>&1
15
16 ######
17 #      VARIABLES      #
18 #####
19 srv_dest="srv-tpi-ldo-02.dsprod.ch" # Serveur de destination
20
21 src_pool="zroot"                   # Pool ZFS de départ
22 dest_pool="zroot/new"              # Pool ZFS de réception sur le serveur distant
23
24 snap_prefix="today_"
25 snap_time=$(date '+%H%M')
26 snap_time_full=$(date '+%H')
27 snap_new="$src_pool@$snap_prefix$nap_time"           # Forge du nom du nouveau
snapshot
28
29 #####
30 #      TEST      #
31 #####
32 if [ $snap_time_full -lt 10 ]          # Test si l'heure est inférieure à 10 heure alors 0X
33 then
34     snap_time_full=0$nap_time_full
35 fi
36
37 snap_old="$src_pool@$snap_prefix$nap_time_full"    # Forge du nom de l'ancien
snapshot
38
39
40 ######
41 #      FUNCTIONS      #
42 #####
43 function zfs_snap() {                  # Fonction de départ qui crée le
snapshot et vérifie que il existe un plus vieux pour faire de l'incrémental
44     zfs snapshot -r "$snap_new"          # Create the snapshot
45     zfs list -t snapshot > /tmp/snapshot.txt        # Export de la liste des snapshot
46
47     if grep -Fq "$snap_new" /tmp/snapshot.txt      # Si | test si le nouveau
snapshot a été fait
48         then
49             echo "[ OK ] New snapshot exist !"
50             if grep -Fq "$snap_old" /tmp/snapshot.txt      # Si | test si le snapshot
vieux de 1 heure existe
51                 then
52                     echo "[ OK ] Older 1 hour snapshot exist !"
53                     rm -rf /tmp/snapshot.txt
54                     zfs_send                                # Si oui | Il va aller dans la
fonction zfs_send, afin de faire de l'incrementiel
55                 else
56                     echo "[ WARNING ] Older 1 hour snapshot don't exist !"
57                     rm -rf /tmp/snapshot.txt
58                     zfs_send_first                         # Si non | Il va créer le
premier backup full
59                 fi
60             else
61                 echo "[ ERROR ] unknown"
62                 rm -rf /tmp/snapshot.txt

```

```

63         exit -1;
64     fi
65 }
66
67 function zfs_send_first() {                                # Fonction de backup full avec zfs send
68     echo "[ INFO ] Starting sending first snapshot...""
69     zfs send "$snap_new" | ssh "$srv_dest" zfs recv "$dest_pool"      # Envoie tout le
70     contenu sur le serveur distant en un bloc
71     zfs_control_recv
72 }
73
74 function zfs_send() {                                     # Fonction de backup incrementiel avec
75     zfs send
76     echo "[ INFO ] Starting sending incremental snapshot..."
77     zfs send -R -i "$snap_old" "$snap_new" | ssh "$srv_dest" zfs recv "$dest_pool" #
78     Envoie le diff du vieux snapshot sur le nouveaux serveur
79     zfs_control_recv
80
81 function zfs_control_recv() {                            # Fonction de control que le snapshot
82     envoyer aie bien été reçu de l'autre coté
83     ssh "$srv_dest" 'zfs list -t snapshot > /tmp/snapshot.txt'
84     ssh "$srv_dest" 'if grep -Fq "$snap_new" /tmp/snapshot.txt; then echo "[ OK ]"
85     destination have good receive snapshot !"; else exit -1; fi'
86     ssh "$srv_dest" 'rm -rf /tmp/snapshot.txt'
87     send_xml
88 }
89
90 function send_xml() {                                    # Fonction d'envoie des fichier de
91     config des vm
92     echo "[ INFO ] Sending xml files for KVM..."
93     rsync -ra /etc/libvirt/qemu/ "$srv_dest":/etc/libvirt/qemu/
94     modifiaction_xml
95 }
96
97 function modifiaction_xml() {                           # Modification des xml, car nouveau
98     chemin n'est pas le meme que l'ancien
99     echo "[ INFO ] Modification of xml for new $dest_pool ..."
100    ssh "$srv_dest" "sed -i 's/zroot/zroot\\new/g' /etc/libvirt/qemu/*.xml"
101    restart_kvm
102 }
103
104 function restart_kvm() {                             # Fonction de restart de kvm afin de
105     prendre en compte les modifs
106     echo "[ INFO ] Restarting kvm for apply change..."
107     ssh "$srv_dest" service libvirt-bin restart
108     shutdown_restart
109 }
110
111 function shutdown_restart() {                         # Si il y a des vm allumée
112     /usr/bin/shutdownKVM
113     virsh list --uuid > /tmp/shutdownKVM
114     sed -i '/^$/d' /tmp/shutdownKVM
115     if [[ -s /tmp/shutdownKVM ]]; then
116         /usr/bin/shutdownKVM
117         alors go a la fonction pour les éteindre
118     else
119         echo "[ OK ] All vm is already down"
120         rm -rf /tmp/shutdownKVM
121     fi
122
123     ssh "$srv_dest" "/usr/bin/startKVM"
124 }
125 #####
126 #      EXECUTION      #
127 #####
128 zfs_snap

```

```
123
124 ######
125 #      UNSET      #
126 ######
127 unset snap_time_old
128 unset srv_dest
129 unset src_pool
130 unset dest_pool
131 unset snap_prefix
132 unset snap_time
133 unset snap_time_full
134 unset snap_new
135 unset snap_old
136
```

```

1 #!/bin/bash -
2 #title      :delOldSnap
3 #description :This script delete old Snapshot
4 #author     :Lucas Dousse
5 #date       :20170512
6 #version    :0.1
7 #usage      :./delOldSnap.sh
8 #notes      :
9 #bash_version :4.3.46(1)-release
10 =====
11
12 # Logger
13 exec 1>>(logger -s -t ${basename $0}) 2>&1
14
15 zfs list -H -t snapshot -o name -S creation | sed '1d' | sed '1d' | sed '1d' >
16 /tmp/snap_srv.txt # Liste les snapshot et supprime les 3 derniers du fichiers
17
18 if [[ -s /tmp/snap_srv.txt ]]; then                                # Test si le fichier n'est
pas vide
19   while IFS='' read -r line || [[ -n "$line" ]]; do                  # Boucle pour lire et
supprimer les vieux snapshot
20     echo "[ INFO ] Delete: $line"
21     zfs destroy "$line"
22   done < "/tmp/snap_srv.txt"
23   rm -rf /tmp/snap_srv.txt
24 else
25   echo "[ OK ] snapshot is already delete"                            # Sinon ne fait rien
26   rm -rf /tmp/snap_srv.txt
27 exit 0;
28 fi

```

```

1 #!/bin/bash -
2 #title          :startKVM
3 #description    :This script shutdown all vm of the kvm
4 #author         :Lucas Dousse
5 #date          :20170512
6 #version        :0.1
7 #usage          :./startKVM
8 #notes          :
9 #bash_version   :4.3.46(1)-release
10 =====
11
12 # Logger
13 exec 1>>(logger -s -t ${basename $0}) 2>&1
14
15 ######
16 #      FUNCTIONS      #
17 ######
18 function get_uuid() {                                # Fonction récupérant les uuid des vm
allumée
    virsh list --uuid --all > /tmp/startKVM
    sed -i '/^$/d' /tmp/startKVM
    if [[ -s /tmp/startKVM ]]; then
        go_start                                # Si il y a des vm éteinte alors go a
        la fonction pour les start
    else
        echo "[ OK ] All vm is already Up"
        rm -rf /tmp/startKVM
        exit 0;
    fi
}
28
29
30 function go_start() {                                # fonction pour start les vm
31     while IFS= read -r line                         # Boucle démarrent les vm
32 do
33 # display $line or do somthing with $line
34     printf "$line" Shutdown
35     virsh start "$line"
36 done < "/tmp/startKVM"
37 rm -rf /tmp/startKVM
38 exit 0;
39 }
40
41
42 ######
43 #      EXECUTION      #
44 ######
45 echo "Starting all vm"
46 get_uuid
47 echo "End of start"
48

```

```

1 #!/bin/bash -
2 #title           :shutdownKVM
3 #description     :This script shutdown all vm of the kvm
4 #author          :Lucas Dousse
5 #date            :20170512
6 #version         :0.1
7 #usage           :./shutdownKVM
8 #notes           :
9 #bash_version   :4.3.46(1)-release
10 #####
11
12 # Logger
13 exec 1>>(logger -s -t ${basename $0}) 2>&1
14
15 ######
16 #      FUNCTIONS      #
17 ######
18 function get_uuid() {                                # Fonction récupérant les uuid des vm
allumée
    virsh list --uuid > /tmp/shutdownKVM
    sed -i '/^$/d' /tmp/shutdownKVM
    if [[ -s /tmp/shutdownKVM ]]; then
        go_shut
        la fonction pour les éteindre
    else
        echo "[ OK ] All vm is already down"
        rm -rf /tmp/shutdownKVM
        exit 0;
    fi
}
28
29
30 function go_shut() {                                # fonction pour éteindre les vm
31     while IFS= read -r line                         # Boucle éteignant les vm
32 do
33 # display $line or do somthing with $line
34     printf "$line" Shutdown
35     virsh shutdown "$line"
36 done < "/tmp/shutdownKVM"
37 rm -rf /tmp/shutdownKVM
38 exit 0;
39 }
40
41
42 ######
43 #      EXECUTION      #
44 ######
45 echo "[ INFO ] Starting going down"
46 get_uuid
47 echo "[ INFO ] End of shutdown"
48

```