Guide installation serveurs 2014

Version pour les serveurs installés en 2014, basé sur l'architecture déjà en place.

1. Préambule

L'objectif de ce guide et de formuler de manière la plus détaillée possible la mise en œuvre de ma nouvelle architecture serveur. Ce guide s'inspire de nombreux autres guides qui seront repris en grande partie. L'idée est de disposer d'un manuel autonome qui pourra persister y compris après la disparition des sites de références.

2. Références en vrac

Cette partie liste tous les liens qui m'ont servi à construire ce guide. C'est à ordonner mieux que ça mais le vrac a le mérite de lister les choses.

- **Un peu de blabla perso pour introduire mon ancienne installation**
  - http://vivihome.net/2013/02/17/nouvelle-architecture-serveurs-partie-1-point-de-depart/
- **La suite du blabla perso avec quelques liens plus utiles**
  - http://vivihome.net/2013/02/23/nouvelle-architecture-serveurs-partie-2-preparation-des-machines/
- **L'excellent article de Victor Héry qui m'a servi de base principale pour la partie cluster proxmox / vpn**
  - http://blog.héry.com/article6/cluster-proxmox-distant-construction-du-cluster-openvpn
- **Une info utile sur les snapshot proxmox**
  - http://vivihome.net/2013/03/08/faire-des-snapshots-sous-proxmox-sans-que-les-vms-soient-suspendues-letitux/
  - www.letitux.com/proxmox/2013/02/02/faire-des-snapshots-sous-proxmox-sans-que-les-vms-soient-suspendues.html
- **Des notes importantes sur l'utilisation de DRDB / LVM**
  - http://vivihome.net/2013/03/28/nested-lvm-configuration-with-drbd/
  - http://vivihome.net/2013/09/28/cluster-proxmox-disques-drbd-shared/
  - http://vivihome.net/2013/03/28/drbd-proxmox-ve/
- **Une petite note sur le FTP passif et iptables (utile chez online.net depuis un temps)**
  - http://vivihome.net/2012/12/29/online-passe-sans-prevenir-ses-serveurs-backup-en-ftp-passif-only-et-aller-on-refait-iptables/
- **L'excellent article de Alsacréation sur la sécurisation d'un serveur**
  - http://vivihome.net/2011/08/15/securiser-son-serveur-dedie/
  - www.alsacreations.com/tuto/lire/622-Securite-firewall-iptables.html

3. Architecture du cluster

3.1. Introduction

Nombre d'étapes de ce guilde sont communes aux deux serveurs physiques. Il s'agit de les dérouler et d'apporter quelques personnalisations selon les cas.

A noter que nous aurons deux serveurs physiques nommés ainsi (pour l'exemple mais a adapter selon votre config) :

- **SERVEUR_PHYSIQUE_1**
- **SERVEUR_PHYSIQUE_2**

Note importante : que chacune des opérations abordées dans les sous chapitres de cette partie sont à faire sur chacun des serveurs physiques en adaptant bien sur la configuration lorsque c'est nécessaire.

3.2. Choix des serveurs

Comme depuis un moment maintenant, je suis chez online.net. J'ai donc choisi assez naturellement leurs petits serveurs dédiés, soit 2 LT 2014 de la gamme start LTS.

3.3. Choix de la distribution

Je voulais une distribution qui facilite la construction de cluster et permette la virtualisation. L'idée est de pouvoir monter plusieurs serveurs virtuels avec seulement deux machines physiques et pouvoir migrer les VM facilement d'un noeud à l'autre en cas de crash.

La distribution que j'ai choisie sera donc proxmox. Elle permet la virtualisation et a le mérite d'être un minimum ouverte (bien que cela change un peu ces temps derniers), d'être gratuite si l'on ne prend pas de support et de fonctionner de manière assez simple avec des outils linux connus mais packagés correctement. Je ne dis pas que ESXI est mauvais, mais juste que dès qu'il s'agit de bidouiller et ajouter de fonctionns ce n'est pas possible sur ESXI (sauf a sortir son carnet de chèques).

3.4. HA or not HA / nombre de noeuds

Pour cette architecture, nous allons fonctionner sur un cluster avec 2 noeuds. En théorie, un cluster permettant de faire du HA doit comporter un nombre impaire de noeuds donc au moins 3 (notamment pour des histoires de quorum). Je n'ai pas choisi de partir sur cette architecture pour plusieurs raisons :

- Mon budget, limité, m'empêche d'investir sur un 3ème serveur

- Mon temps, limité, m'empêche également de passer du temps a un plan B style le 3ème serveur chez moi
- Le HA pour bien fonctionner doit mettre en place du fencing ce qui n'est pas simple du tout et nécessite en théorie du matériel spécifique.

Nous partons donc sur **2 noeuds**, un principal et un secondaire. L'idée est que sur ce cluster les VM puissent migrer très facilement d'un noeud à l'autre en cas de défaillance d'un des noeuds.

3.5. Utilisation de DRBD

Comme je le disais, je souhaite mettre en place une solution me permettant une architecture robuste et tolérante aux pannes. Ce qui veut dire pouvoir démarrer mes VM sur n'importe quel noeud et ce même après le crash d'un noeud. Il existe pas mal de choses pour faire cela mais j'ai choisi de mettre en place une architecture basée sur DRBD.

DRBD me permet ainsi de mettre en place une **réplication bidirectionnelle** entre des disques de mes deux clusters. Ansi, toute l'activité d'un cluster est répliquée en temps réel sur l'autre serveur ce qui, en cas de crash me permet de démarrer sans perte la VM sur le noeud restant grâce aux données répliquées.

Attention, pour éviter les problèmes (comme moi lors de ma première tentative), vous devez mettre en place **2 disques en réplication bidirectionnelle sur chaque noeud du cluster**. Pourquoi ? Parce que si vous faites tourner en même temps les VM de chaque cluster sur le même disque DRBD (avec répli bidirectionnelle) vous aurez fatalement du split brain. Ce qui est assez dur à régler. En gros en cas de crash vous aurez le même disque DRBD (normalement synchro sur les deux noeuds du cluster) qui aura vécu sa vie de manière différente sur chacun des noeuds. Les données sont donc à resynchro des deux côtés et vous ne pouvez pas écraser un le disque DRBD d'un noeud avec celui de l'autre “plus récent” puisqu'il y a des données différentes des deux côtés. Dans ce cas (vécu chez moi) vous pourriez être obligés de perdre des données...

La manière simple d'éviter cela est donc d'avoir **deux disques DRBD** sur chaque noeud du cluster. Pour chaque noeud, n'utiliser qu'un seul disque DRBD différent de l'autre noeud pour faire tourner les VM. Ansi en cas de crash vous pourrez facilement restaurer les données d'un disque périmé avec celles de celui qui a tourné sans risquer de perdre des données.

3.6. Côté réseau

Je me base pas mal sur cet article de Victor Héry ainsi que mes expériences perso sur la précédente configuration, qui m'ont amenées à vouloir tester d'autres façons de faire.

Chaque serveur de cette gamme dispose de deux cartes réseau physique :

- Une connectée sur le réseau public
- Une connectée sur le réseau RPN online.net

Par défaut votre configuration proxmox aura une **interface bridgée vmbr0** connectée à l'interface physique qui sera sur le réseau public.

Pour notre architecture nous mettrons en place un autre bridge pour les VM : **vmbr1**. Nous détaillerons la configuration plus bas. Ce bridge assurera pour la communication entre les VM ainsi que la réplication DRBD. Les deux machines physiques seront mises en relation via ces bridges et un tunnel VPN.

On utilisera le réseau RPN pour les opérations cluster proxmox. Par défaut il faut du multicast, mais on sait le désactiver pour fonctionner en unicast. L'idée ici est de ne pas tout mettre sur le VPN et donc utiliser un peu le RPN pour les échanges interne proxmox. De plus cela augmente la fiabilité : pas tous les œufs dans le même panier. Par contre il faut noter que par défaut le réseau RPN est à une vitesse de 100Mbps ce qui est insuffisant pour une réplication DRBD : raison de plus pour la faire sur OpenVPN qui sera configuré via les interfaces publiques du serveur qui elles fonctionnent à 1Gpbs.

Note : avoir le RPN à 100Mbps est possible mais c'est une option qui va vous obliger a prendre un niveau de service online.net supérieur à “Basic” (qui lui est gratuit). Là en gros vous allez devoir sortir au minimum 20 euros HT en plus par serveur pour avoir ce niveau de service. Ensuite il faudra encore rajouter quelques euros pour avoir l'option RPN 1Gbps. BREF ! Quand on est pas super riche on ruse avec le VPN sur l'interface publique. Si on est riche alors je vous encourage a utiliser le RPN au maximum car cela évitera de charger les interfaces publiques par lesquelle tout le traffic internet arrive déjà. De plus niveau robustesse c'est mieux. Si l'interface publique est down, le réseau RPN est toujours potentiellement OK lui… Bref, la solution VPN est un compromis économique mais qui comporte des défauts de conception…

4. Premières étapes d'installation

Les premières étapes sont assez simple chez online, après un tour sur l'interface d'admin, on choisit sa distribution et on lance l'installation pour chacun des serveurs.

On retouchera plus tard les choses qui ne vont pas comme par exemple le partitionnement. De toutes manières, l'outil de partitionnement ne permet pas grand chose sur l'interface online.net. Si on demande une configuration vraiment spécifique, l'outil ne fonctionne pas et n'enregistre rien du tout.

Note : vous vous dites que vous pourriez faire une installation manuelle via Idrac pour résoudre les limitations de l'install web. C'est pas faux. Sauf que pour activer l'idrac (ou l'ilo si vous avez un serveur hp) il faut de toutes façons installer complètement le serveur au moins une fois avec que l'option idrac ou ilo aparaisse sur l'interface d'admin de votre dédié… Bref, ça n'évite pas une première install bateau via le web quoi qu'il arrive…

5. Une petite mise à jour ne fait pas de mal !

Si vous n'utilisez comme moi que la version gratuite, il faut désactiver les mises à jour venant de la version pro en commantant le deb dans ce fichier :

```
nano /etc/apt/sources.list.d/pve-enterprise.list
```

Puis ensuite lancer la mise à jour.

```
apt-get update
apt-get upgrade
```

6. Personnalisation shell Bash (couleurs / infos)

En grande partie repris (aux perso près) de cet article..

Afin de rendre le shell plus lisible avec quelques couleurs dont celle du prompt qui pourra être différente selon le noeud du cluster.

```
nano /etc/bash.bashrc
```

Et on ajoute cela à la fin du fichier :

```
#
# Colors and prompt configuration
#
C_RED="[e[1;31m\]"
C_BLUE="[e[1;34m\]"
C_GRAY="[e[1;30m\]"
C_WHITE="[e[1;37m\]"
C_YELLOW="[e[1;33m\]"
C_DEF="[033[0m\]"

mUID=`id -u`
MACHINE=`hostname -f`

#if [ "$mUID" = "0" ] ; then
PS1="${C_RED}\u${C_DEF}@${C_RED}${MACHINE}${C_DEF}:\w${C_RED}#${C_DEF} "
PS2="${C_RED}>${C_DEF} "
#else
#PS1="${C_BLUE}\u${C_DEF}@${MACHINE}:\w${C_BLUE}\$ ${C_DEF}"
#PS2="${C_BLUE}>${C_DEF} "
#fi

export PS2
export PS1

case $TERM in
xterm*)
PROMPT_COMMAND='echo -ne "\033]0;${USER}@${MACHINE}: ${PWD}\007"'
echo -ne "\033]0;${USER}@${MACHINE}: ${PWD}\007"
;;
*)
```

```
setterm -blength 0
;;
esac

#
# Alias configuration
#
alias l='ls -ahlF --color=yes --full-time --time-style=long-iso | more'
alias ll='ls -alhF --color=yes --full-time --time-style=long-iso'
alias cpav='cp -av'
```

Note : pour activer un prompt différent entre root et les autres users, vous n'avez qu'a décommenter le if.

A noter également que les autres users que root ont un .bashrc dans leur réportoire home très complet écrasant la configuration qu'on vient de faire. Donc deux options, soit on personnalise aussi le .bahsrc de tous les users soit on le simplifie pour que le paramétrage par défaut s'applique. Pour ma part je choisis l'option 2 en recopiant le .bashrc minimaliste du user root.

Donc dans le répertoire home du user (ex : /home/monuser)

```
cp /root/.bashrc .
```

Et on oublie pas de vérifier que les droits sont bien pour le user courant sur ce fichier, sinon changer user / group avec chown.

Pour le second serveur, on renouvelle l'opération de manière identique.

Je conseille de changer les couleurs du prompt avec autre chose pour bien différentier les serveurs. Pour cela dans les lignes “PS1=” changer C_RED par la couleur de votre choix.

7. Configurer ses DNS

Il est très utile (par exemple pour les mails) d'avoir une zone DNS correctement configurée pour accéder à ses serveurs. Cela veut dire au minimum, que vous allez vous acheter un nom de domaine et configurer ensuite des sous domaines pour vos différentes machines.

Evidemment les parties type *machine1.domaine.net* sont à personnaliser tout comme *IP_SERVEUR_PHYSIQUE_1*.

Exemple :

- machine1.domaine.net
- machine2.domaine.net

Avec une conf DNS de ce genre par machine :

- **Type :***A*

- **Nom :***machine1.domaine.net.*
- **TTL :***900*
- **Valeur :***IP_SERVEUR_PHYSIQUE_1*

Et une conf SPF de ce genre pour ne pas vous faire jeter par les analyses antispam (toujours pour chaque serveur) :

- **Type :***TXT*
- **Nom :***machine1.domaine.net.*
- **TTL :***3600*
- **Valeur :***“v=spf1 a mx ip4:IP_SERVEUR_PHYSIQUE_1 -all”*

Et bien sur ne pas oublier de **modifier les reverses** de vos serveurs pour qu'ils pointes sur les domaines que vous venez de définir. Pour online.net il suffit d'aller sur la console d'administration de votre serveur et cliquer sur “Modifier les reverses” et vous mettez le domaine correspondant (ex : machine1.domaine.net.).

Note importante : attention la propagation des DNS peut mettre un certain temps. Disons jusqu'à 24 à 48H pour être à jour partout. Donc il vaut mieux anticiper cette partie... Cela dit pour ce qu'on a à faire la mise à jour va assez vite (quelques minutes tout au plus).

8. Changer le hostname de ses machines

Il faut maintenant changer le nom des machines, conformément à ce qu'on a mis en DNS et en reverse. C'est un peu particulier sur proxmox et j'ai trouvé ce topic qui en parle pas mal. Comme toujours, à adapter en fonction de votre config.

Changer le fichier hostname :

```
nano /etc/hostname
```

```
machine1
```

Puis le fichier hosts :

```
nano /etc/hosts
```

```
# Localhost
127.0.0.1       localhost
IP_SERVEUR_PHYSIQUE_1 machine1
```

Evidemment il peut y avoir d'autres choses sur la ligne de IP_SERVEUR_PHYSIQUE_1.

Note 1 : nous verrons tout à leur que nous rechangeront cette configuration pour l'adapter aux besoins du cluster et des VM mais pour l'instant faisons propre :)

Et enfin changement du certificat proxmox (faites un ptit backup au cas ou avant et regardez aussi ce topic) :

```
rm /etc/pve/pve-root-ca.pem
```

```
pvecm updatecerts --force
```

Note 2 : mieux vaut faire cette opération de suite, car ce sera le bordel une fois le cluster monté… Changer le nom des machines en cours de route alors qu'elles sont utilisées pour identifier les noeuds, ça sent le sapin !

9. Configurer son serveur mail

Par défaut la configuration de postfix laisse à désirer. Il faut donc y redonner un coup de tournevis :

```
dpkg-reconfigure postfix
```

La conf que j'utilise (réponse aux écrans 1 par 1) :

- **Configuration type du serveur de messagerie :***Site Internet*
- **Nom de courrier :***machine.domaine.net* (a adapter avec votre domaine et sous domaines sur votre DNS).
- **Destinataire des courriels de « root » et de « postmaster » :***/etc/aliases*
- **Autres destinations pour lesquelles le courrier sera accepté (champ vide autorisé) :** *$myhostname, $myhostname.$mydomain, $mydomain*
- **Faut-il forcer des mises à jour synchronisées de la file d'attente des courriels ?***Non*
- **Réseaux internes :***127.0.0.0/8*
- **Faut-il utiliser procmail pour la distribution locale ?***Oui*
- **Taille maximale des boîtes aux lettres (en octets) :***51200000*
- **Caractère d'extension des adresses locales :**+
- **Protocoles internet à utiliser :***ipv4*

Et un petit tour dans le fichier de conf postfix (précédemment édité) pour vérifier que tout est OK et modifiifier myhostname, mydomain avec par exemple machine1.domaine.net et enfin /etc/postfix/generic pour forcer la réécriture des “from”.

```
nano /etc/postfix/main.cf
```

Ce qui vous donnera

```
myhostname = machine1.domaine.net
mydomain = mondomaine.net
smtp_generic_maps = hash:/etc/postfix/generic
```

On crée / modifie le fichier generic :

```
nano /etc/postfix/generic
```

Qui ressemble à ceci (voir la documentation postfix) :

```
@machine1        noreply@machine1.mondomaine.net
```

On active la configuration contenu dans le fichier generic et on reboot postfix :

```
postmap /etc/postfix/generic
service postfix restart
```

Note 1 : Pour la debian fournie avec promox 3 (squeeze je crois) pas besoin de configuration avec un fichier generic pour réécrire les adresses mail expéditeur. Il prend correctement myhostname. Par contre pour la dernière ubuntu (la trusty) impossible de lui faire prendre myhostname ou mydomain ou myorigin... Donc seule la configuration avec generic permet de forcer l'écriture d'une adresse expéditeur correcte.

Note 1.1 : Après enquête la différence (que je soupçonne) serait sur l'absence de procmail sur la dernière ubuntu contraitement à la debian...

Note 2 : Dans generic je force a une adresse norply@ pour éviter de montrer les users système qui envoient les mails dans les mails automatiques. A adapter a votre configuration.

Note 3 : C'est très important de bien configururer une bonne adresse from qui correspond à votre host.domaine.net car ça vous permettra d'éviter les échec (et donc refus ou spam) d'envoi de mail sur la négocation avec les serveurs distants via HELO (ce topic en parle pas mal).

Note 4 : Il est très très important d'avoir des dns configurés (host.domaine.net) correctement pour émettre vos mails. Car sinon tous les mails envoyés par votre serveur finiront dans la boite à spam (meilleur des cas) voire vos ip seront blacklistées. Pourquoi ? En gros parce que vos serveurs enverrons des mails avec des dns incohérents et les filtres antispam sont sans pitié !

Pour en savoir plus sur la configuration correcte de postfix pour éviter les spams, se référer à cet article, plus détaillé.

10. Sécurisation des serveurs

10.1 Introduction

En grande partie repris de cet article. Je reprends quasi tel que (défois que ça disparaisse), sauf peut être la partie firewall que j'ai adaptée.

10.2. Modifier le mot de passe root

N'hésitez pas à modifier le mot de passe surtout si celui-ci vous a été attribué par défaut. Identifiez-vous d'abord en root (voir ci-dessus) puis entrez la commande :

```
passwd root
```

10.3. Configuration SSH

Afin de sécuriser l'accès SSH au serveur, éditons le fichier /etc/ssh/sshd_config. Nous allons changer le port de connexion par défaut pour éviter quelques attaques par bruteforce sur le port 22, qui est bien connu pour héberger ce service. N'oubliez pas de préciser ce nouveau port (dans Putty ou en ligne de commande ssh sous Linux) à la prochaine connexion.

```
nano /etc/ssh/sshd_config
```

Pour rappel, le port et le users sont à changer en fonction de votre configuration perso bien sûr.

```
Port 2222                    # Changer le port par défaut (à retenir!)
PermitRootLogin yes          # Obligé a cause de proxmox
```

Dans l'idéal on devrait n'autoriser qu'un user a se connecter et surtout pas root :

```
PermitRootLogin no           # Ne pas permettre de login en root
AllowUsers dew               # N'autoriser qu'un utilisateur précis
```

Mais on ne le fera pas hélas car proxmox a besoin de se connecter en root notamment pour les histoires de gestion du cluster.

Pour la partie client (et surtout pour proxmox) on doit également changer la configuration de base du client ssh pour qu'il utilise le port qu'on a changé (qui n'est plus le port par défaut) :

```
nano /etc/ssh/ssh_config
```

Et modifier la ligne du port pour l'adapter à votre configuration :

```
Port 2222
```

La modification est à faire sur les deux noeuds. Sans cette modification, lorsque vous voudrez ajouter votre 2eme noeud au cluster vous aurez un beau

```
unable to copy ssh ID
```

Redémarrez le service SSH après ces modifications :

```
service ssh restart
```

10.5. Alerte login root ou autre user

Vous pouvez éditer le fichier **/root/.bashrc** (ou celui d'un utilisateur lambda /home/myuser/.bashrc) qui est exécuté au démarrage d'une sesion root (ou du user correspondant) pour envoyer un e-mail de notification. De cette façon, vous serez prévenu lorsqu'un login est effectué.

```
nano /root/.bashrc
```

Ajoutez la ligne (en modifiant l'adresse e-mail de destination) :

```
#
# Mail alert on root connection
#
echo "Accès shell root le `date` `who`" | mail -s "`hostname` - Shell root
de `who | cut -d"(" -f2 | cut -d")" -f1`" monitoring@test.com
```

10.6. Firewall et IPtables

Bon c'est un peu long et tout ne va pas servir de suite... Cependant il faut bien en parler à un moment ou un autre...

Pour désactiver l'IPv6, si vous n'en avez pas besoin (pas de troll :)) et ça évitera de devoir faire de la conf ipv6... (oui je suis flemmard mais ca n'est pas très utile pour moi a priori).

```
echo "options ipv6 disable=1" > /etc/modprobe.d/disable-ipv6.conf
```

On crée notre fichier de configuration

```
nano /etc/init.d/iptables
```

Et on le valorise dans ce gout là. A adapter en fonction de votre config et à faire pour chaque serveur physique. Bien sur quand on autorise le serveur 2 sur le serveur 1, on autorise le serveur 1 sur le serveur 2 (faut inverser).

Attention celui ci est très complet mais il faudra élaguer sans doute en fonction de vos besoins. Un principe de sécurité : on interdit tout et on ouvre que le stricte minimum.

Note : la configuration ftp passive est un peu délicate. Je me facilite donc la vie en ne l'utilisant que pour mon backup online en mode semi bourrin (voir cet article). Par ailleurs pour savoir si l'adresse est bien *dedibackup-dc2.online.net* Rendez vous dans la rubrique sauvegarde sur la page de votre serveur sur le site online.net.

```
#!/bin/bash

echo Setting iptables / firewall rules...
#
# Dedibox iptables / firewall configuration
#
# http://wiki.debian.org/LSBInitScripts/DependencyBasedBoot
# http://wiki.debian.org/LSBInitScripts
#
http://documentation.online.net/fr/serveur-dedie/tutoriel/iptables-netfilter
-configuration-firewall
# http://www.alsacreations.com/tuto/lire/622-Securite-firewall-iptables.html
#
```

```
# Modifications alban.montaigu
#

###### Start Initialisation ######

# Empty current tables
iptables -t filter -F
iptables -t filter -X
echo - Empty current tables : [OK]

# No input connection
iptables -t filter -P INPUT DROP
iptables -t filter -P FORWARD DROP
echo - No input connection : [OK]

# No output connection
iptables -t filter -P OUTPUT DROP
echo - No output connection : [OK]

# Authorize private hosts for all
# Physical public IP of physical server 2
iptables -A INPUT -s IP_SERVEUR_PHYSIQUE_2 -j ACCEPT
iptables -A FORWARD -s IP_SERVEUR_PHYSIQUE_2 -j ACCEPT
iptables -A OUTPUT -d IP_SERVEUR_PHYSIQUE_2 -j ACCEPT

# Physical RPN IP of physical server 2
iptables -A INPUT -s RPN_IP_SERVEUR_PHYSIQUE_2 -j ACCEPT
iptables -A FORWARD -s RPN_IP_SERVEUR_PHYSIQUE_2 -j ACCEPT
iptables -A OUTPUT -d RPN_IP_SERVEUR_PHYSIQUE_2 -j ACCEPT

# Authorize SSH
iptables -t filter -A INPUT -p tcp --dport 2222 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 2222 -j ACCEPT
echo - Authorize SSH : [OK]

# Authorize OpenVPN interfaces (enable multicast)
iptables -A INPUT -i tap+ -j ACCEPT
iptables -A FORWARD -i tap+ -j ACCEPT
iptables -A OUTPUT -o tap+ -j ACCEPT
echo - Authorize OpenVPN : [OK]

# Authorize interfaces on internal network (enable multicast)
iptables -A INPUT -i vmbr1 -j ACCEPT
iptables -A FORWARD -i vmbr1 -j ACCEPT
iptables -A OUTPUT -o vmbr1 -j ACCEPT
echo - Authorize vmbr1 [OK]

#  Do not break established connections
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
echo - Do not break established connections : [OK]

###### End Inialisation ######

##### Start Rules  ######

# Authorize DNS, FTP, HTTP, NTP
iptables -t filter -A OUTPUT -p tcp --dport 21 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 53 -j ACCEPT
iptables -t filter -A OUTPUT -p udp --dport 53 -j ACCEPT
iptables -t filter -A OUTPUT -p udp --dport 123 -j ACCEPT
echo - Authorize DNS, FTP, HTTP, NTP : [OK]

# Authorize loopback
iptables -t filter -A INPUT -i lo -j ACCEPT
iptables -t filter -A OUTPUT -o lo -j ACCEPT
echo - Authorize loopback : [OK]

# Authorize ping
iptables -t filter -A INPUT -p icmp -j ACCEPT
iptables -t filter -A OUTPUT -p icmp -j ACCEPT
echo - Authorize ping : [OK]

# HTTP
iptables -t filter -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 8443 -j ACCEPT
echo - Authorize HTTP : [OK]

# Monit
iptables -t filter -A INPUT -p tcp --dport 8080 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 8080 -j ACCEPT
echo - Authorize monit : [OK]

# Proxmox admin
iptables -t filter -A INPUT -p tcp --dport 8006 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 8006 -j ACCEPT
echo - Authorize proxmox admin : [OK]

# Proxmox VNC it's ok because vncserver is started only when requested in
the GUI
iptables -t filter -A INPUT -p tcp --dport 5900 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 5900 -j ACCEPT
echo - Authorize proxmox VNC : [OK]

# FTP (passive dedibackup-dc2.online.net and active)
modprobe ip_conntrack_ftp
iptables -t filter -A INPUT -p tcp --dport 20 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 21 -j ACCEPT
iptables -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -t filter -A OUTPUT -p tcp --sport 1024:65535 --dport 1024:65535 -m
state --state NEW -d dedibackup-dc2.online.net -j ACCEPT
echo - Authorize FTP : [OK]

# Mail
iptables -t filter -A INPUT -p tcp --dport 25 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 110 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 143 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 25 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 110 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 143 -j ACCEPT
echo - Authorize mail : [OK]

# LOGGING if necessary (must be at the end)
# From http://www.thegeekstuff.com/2012/08/iptables-log-packets/
#iptables -N LOGGING
#iptables -A INPUT -j LOGGING
#iptables -A OUTPUT -j LOGGING
#iptables -A LOGGING -j LOG --log-prefix "[IPTABLES-DROP]" --log-level debug
#iptables -A LOGGING -j DROP
#echo - Log enabled

###### End rules ######

echo Iptables / firewall update successfull !
```

On donne les droits d'execution à ce script :

```
chmod +x /etc/init.d/iptables
```

Et on le test **AVANT** de le démarrer automatiquement... Sinon vous êtes bon pour redémarrer votre serveur en mode secours pour retoucher le fichier... L'avantage de pas le mettre en mode boot automatique c'est que quand vous faites une connerie, suffit de reboot le serveur via l'interface d'admin online.net pour reprendre la main sur le serveur...

Pour activer le script à chaque démarrage,modifiez le script et ajouter ceci après la première ligne #!/bin/bash (plus de update rc, c'est LSBInitScripts maintenant) :

```
### BEGIN INIT INFO
# Provides:             iptables
# Required-Start:       $remote_fs $syslog
# Required-Stop:        $remote_fs $syslog
# Default-Start:        2 3 4 5
# Default-Stop:
# Short-Description:    Iptables / firewall configuration
### END INIT INFO
```

Et on active le boot automatique (voir cette page) :

```
insserv iptables
```

10.7. Fail2ban

Fail2ban est un script surveillant les accès réseau grâce aux logs des serveurs. Lorsqu'il détecte des erreurs d'authentification répétées, il prend des contre-mesures en bannissant l'adresse IP grâce à iptables. Cela permet d'éviter nombre d'attaques bruteforce et/ou par dictionnaire.

Installation :

```
apt-get install fail2ban
```

Configuration :

```
nano /etc/fail2ban/fail2ban.conf
```

- **loglevel** *Niveau de détail des logs (défaut 3)*
- **logtarget = /var/log/fail2ban.log** *Chemin vers le fichier de log (description des actions entreprises par fail2ban)*

Les services à monitorer sont stockés dans **jail.conf**. Il est recommandé d'en effectuer une copie nommée **jail.local** qui sera automatiquement utilisée à la place du fichier exemple.

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

```
nano /etc/fail2ban/jail.local
```

Quelques paramètres globaux :

- **ignoreip = 127.0.0.1** *Liste des adresses IP de confiance à ignorer par fail2ban*
- **bantime = 600** *Temps de ban en secondes*
- **maxretry = 3** *Nombre d'essais autorisés pour une connexion avant d'être banni*
- **destmail monitoring@test.com** *Adresse e-mail destinataire des notifications*
- **action** *Action à entreprendre en cas de détection positive (voir dans /etc/fail2ban/action.d/)*

Chaque section possède ses propres paramètres qui prennent le pas sur les globaux s'ils sont mentionnés :

- **enabled** *Monitoring activé (true) ou non (false)*
- **maxretry, bantime, ignoreip, destmail** *Voir ci-dessus*
- **port** *Port IP concerné*
- **logpath** *Fichier de log à analyser pour détecter des anomalies*
- **filter** *Filtre utilisé pour l'analyser du log*

Les filtres par défaut sont stockés dans **/etc/fail2ban/filter.d**. Ils contiennent en général une instruction failregex suivie d'une expression régulière matchant la détection d'une authentification erronée. Par exemple pour le service Courier :

```
failregex = LOGIN FAILED, ip=[<HOST>]$
```

Note : Celle-ci peut être précisée directement dans **jail.local** à la section appropriée pour prendre le pas sur la directive filter.

Modifiez les ports le cas échéant dans la section ssh si vous avez suivi la recommandation ci-dessus...

```
enabled = true
port    = 2222
```

Après modification de la configuration, n'oubliez pas de redémarrer fail2ban :

```
service fail2ban restart
```

10.8. Rkhunter

Rootkit Hunter est un programme de détection de rootkits. Vous pouvez l'installer grâce à :

```
apt-get install rkhunter
```

Il procédera à des détections journalières anti-rootkits et enverra des notifications par e-mail si nécessaire. Il est conseillé de l'installer très tôt car il calcule l'empreinte MD5 des programmes installés afin de détecter d'éventuels changements. Editez /etc/default/rkhunter pour indiquer l'adresse de notification et l'exécution journalière :

```
nano /etc/default/rkhunter
```

Modifier ainsi :

```
REPORT_EMAIL="monitoring@test.com"
```

En cas de fausses détections positives sur des répertoires ou fichiers existants et sains, éditez /etc/rkhunter.conf pour les ajouter à la liste des éléments autorisés.

```
nano /etc/rkhunter.conf
```

Modifier ainsi (dans la partie correspondante avec les autres du genre) :

```
SCRIPTWHITELIST=/usr/bin/unhide.rb
```

Pour les distributions ubuntu (utile pour les VM seulement dans la suite du guide), on a quelques warnings supplémentaires à enlever de cette manière :

```
ALLOWHIDDENDIR="/dev/.udev"
ALLOWHIDDENFILE="/dev/.initramfs"
ALLOWDEVFILE="/dev/.udev/rules.d/root.rules"
```

Vous pouvez également utiliser chkrootkit qui est un équivalent.

11. Surveillance des serveurs

11.1. Introduction

Dans le domaine de la surveillance / monitoring des serveurs il existe vraiment des tonnes de choses à faire. Sur cette partie, pour l'instant du moins, j'ai décider d'opter pour un service minimum dans le sens ou j'ai une architecture de relativement peu de serveurs et je veux avoir quelque chose qui reste relativement simple. A voir plus tard pour améliorer cette section. Dans tous les cas il faut bien commencer par quelque chose.

Ressources :

- Alsacréation avec cet article (monit + logwatch)
- Moi même (ha c'est bien de garder des notes) avec cet article.

11.2. Logwatch

Je trouve cette solution intéressante car elle permet d'avoir un moyen de synthétiser et suivre ce qui se passe dans les logs du serveur. Et cela on ne le fait jamais assez alors que pourtant c'est une mine d'or !

Logwatch est par un démon pouvant analyser et résumer les logs générés par les autres services durant la journée pour en détecter d'éventuelles anomalies ou en tirer des statistiques. Il permet d'envoyer un e-mail récapitulatif quotidien à l'administrateur. Son installation est elle aussi très simple grâce à APT et au paquet éponyme :

```
apt-get install logwatch
```

La configuration par défaut suffit amplement, il suffit de modifier le destinataire dans le fichier /usr/share/logwatch/default.conf/logwatch.conf. Celui-ci peut se trouver à un autre endroit du disque, n'hésitez pas à vous servir de la commande locate.

```
nano /usr/share/logwatch/default.conf/logwatch.conf
```

Modifiez l'option MailTo et tout ce que vous jugerez nécessaire. Par exemple je modifie le format aussi pour qu'il soit plus sympa a lire. Il peut être utile également de régler le niveau de détail.

```
MailTo = monitoring@test.com
Format = html
Detail = Med
```

Vérifiez au besoin la présence de logwatch dans le répertoire /etc/cron.daily/ dont les scripts sont exécutés quotidiennement.

Vous pouvez même lancer l'execution telle que prévue par cron pour voir si tout est OK :

```
/etc/cron.daily/00logwatch
```

11.3. Monit

Monit est une application permettant de surveiller l'état des services (notamment web, ftp, mail, mysql, ssh) par une interface web, et de notifier l'administrateur si nécessaire (trop grande charge cpu, redémarrage, indisponibilité...).

L'installation de Monit se réalise en quelques secondes grâce au paquet éponyme :

```
apt-get install monit
```

La configuration de Monit se fait en deux temps. Tout d'abord, autoriser le démarrage du service en éditant /etc/default/monit.

```
nano /etc/default/monit
```

L'option START doit être à yes. Il semble que ce soit la valeur par défaut désormais sur les versions récentes.

```
# You must set this variable to yes for monit to start
START=yes
```

Ensuite, éditer le fichier de configuration /etc/monit/monitrc contenant la description de tous les services à surveiller.

```
nano /etc/monit/monitrc
```

Voici un exemple complet. Celui-ci est relativement explicite et à adapter selon votre configuration notamment pour le port SSH (22 par défaut) qui est ici 1337 pour correspondre au précédent tutoriel de configuration.

- **with start delay 240***Va avec le set daemon et permet de différer le démarrage de monit. Utile pour éviter de surveiller des process de suite alors qu'ils n'ont potentiellement pas encore démarré lors du boot*
- **set mailserver***Indique le(s) serveur(s) de mail à utiliser pour l'envoi des notifications*
- **set alert***Indique les adresses destinataires*
- **set httpd port***Spécifie le port de connexion web. Vous pourrez ensuite vous connecter grâce à votre navigateur sur l'IP et le port correspondant (ex: http://test.alsacreations.com:8080/)*
- **allow login:password***Spécifie le couple login/password pour l'accès web (à renseigner)*
- **check device***Va permettre de surveiller l'espace disque restant : il faut ici indiquer le bon path vers /dev/XXX correspondant à la partition à monitorer (ex : /dev/sda, /dev/md1... selon votre configuration)*

Note 1 : ici je ne surveille pas des choses comme MySQL ou Apache2 puisque sur le cluster proxmox c'est inutile. De plus je ne surveille pas le serveur FTP puisque je ne souhaite pas en mettre.

Note 2 : Le fichier par défaut est assez fourni, vous pouvez donc prendre exemple dessus et

compléter les éléments indiqués ci dessous.

Eléments à modifier (ne pas reprendre juste cela, manquerait des choses sinon) :

```
# Config

  set daemon 120
    with start delay 240

  set mailserver localhost

  set mail-format {
      from: monit@$HOST.mondomaine.net
      subject: $HOST - Monit : $EVENT $SERVICE
  }

  set alert monitoring@test.com

  set httpd port 8080 and
     allow login:password
```

Puis dans **/etc/monit/conf.d/** vous pouvez ajouter les services qui vous intéressent. On peut aussi tout mettre dans monitrc mais je trouve ça peu lisible au final.

```
nano /etc/monit/conf.d/ssh.conf
```

```
# SSH
check process sshd with pidfile /var/run/sshd.pid
group ssh
start program "/etc/init.d/ssh start"
stop program "/etc/init.d/ssh stop"
if failed host 127.0.0.1 port 1337 protocol ssh then restart
if 5 restarts within 5 cycles then timeout
```

```
nano /etc/monit/conf.d/postfix.conf
```

```
# Postfix
check process postfix with pidfile /var/spool/postfix/pid/master.pid
group mail
start program = "/etc/init.d/postfix start"
stop  program = "/etc/init.d/postfix stop"
if failed port 25 protocol smtp then restart
if 5 restarts within 5 cycles then timeout
```

```
nano /etc/monit/conf.d/disk.conf
```

```
# Disk
check device system-root with path /dev/mapper/system-root
if space usage > 85% then alert
group system
```

Vous êtes libre d'ajouter tous les services à monitorer sur votre machine (pop3, imap, spamassassin, clamav, fail2ban...). La syntaxe est abordable et les exemples nombreux. Pour vérifier cette syntaxe, utilisez la commande :

```
/etc/init.d/monit syntax
```

Si aucun message d'erreur n'est indiqué, vous pourrez ensuite démarrer monit :

```
service monit start
```

Vérifiez une nouvelle fois la bonne interprétation de la configuration grâce à monit -v.

12. Sauvegarde des serveurs

J'aurais pu mettre cette partie dans sécurité mais bon la sauvegarde est tellement importante qu'elle mérite un chapitre à elle même. Cette rubrique sera à compléter car avec un serveur proxmox, on a aussi la question de backup des VM qui peut se poser. Ici je ne traiterai pour l'instant que du stricte minimum (vu la config et les options prises sur mes serveurs) à savoir la sauvegarde de la configuration de mes serveurs.

Pour cela je trouve l'utilitaire backup-manager très bien et le tutorial suivi est celui d'Alsacreation (encore :)).

Backup Manager est un outil en ligne de commande pour Linux conçu pour effectuer des sauvegardes quotidiennes de votre système. Il est écrit en bash et Perl, permet de constituer des archives dans plusieurs formats de compression (tar, gzip, bzip2, lzma, dar, zip) et de les exporter vers un serveur FTP ou de les graver automatiquement.

Dans notre cas, nous allons l'installer, et le configurer pour envoyer des archives compressées sur un serveur FTP externe de sauvegarde. Si vous utilisez une Dedibox, vous pourrez ainsi profiter du système dedibackup.

Consultez si nécessaire Configuration d'un serveur dédié de A à Z (Connexion SSH, Accès root, Edition des fichiers en ligne de commande) avant de poursuivre. Installation

Note importante : pour online.net n'oubliez pas d'aller sur la page console.online.net de votre serveur, cliquez sur **Sauvegarde** notez les paramètres du serveur FTP et enfin éditez les infos de connexion pour initialiser le mot de passe de votre choix.

Backup-manager peut s'installer via apt (pensez à passer en root) :

```
apt-get install backup-manager
```

Il sera lancé quotidiennement grâce au démon cron de Linux. Configuration

Editez le fichier /etc/backup-manager.conf

```
nano /etc/backup-manager.conf
```

Par défaut, les archives seront placées dans le répertoire /var/archives. Ceci est modifiable par l'instruction

```
export BM_REPOSITORY_ROOT="/var/archives"
```

Ce répertoire doit être détenu par root (afin que personne d'autre ne puisse en consulter le contenu).

La durée de conservation des archives est définie (en jours) par :

```
export BM_ARCHIVE_TTL="15"
```

Il faut la moduler en fonction de l'espace de sauvegarde disponible et la quantité de données à conserver.

Des fichiers tar.gz seront constitués :

```
export BM_TARBALL_FILETYPE="tar.gz"
```

Les répertoires à sauvegarder (vous pouvez avoir plusieurs dossiers séparés par des espaces) :

```
export BM_TARBALL_DIRECTORIES="/etc"
```

Dans ce cas, nous sauvegardons l'essentiel de la configuration du système qui se trouve dans /etc.

Il est possible de blacklister certains types de fichiers (ne pas les sauvegarder), ceci est pratique pour ne pas faire exploser la taille des archives :

```
export BM_TARBALL_BLACKLIST="/var/archives *.mp3 *.avi *.rar *.zip *.ogg
*.sql *.tgz *.mpg *.log *.7z"
```

Configuration de la sauvegarde par FTP

Dans notre cas, nous exportons les archives par FTP :

```
export BM_UPLOAD_METHOD="ftp"
export BM_UPLOAD_HOSTS="dedibackup-dc2.online.net"
export BM_UPLOAD_DESTINATION="/"
```

On précise certaines options FTP (BM_UPLOAD_FTP_DESTINATION est redondant avec la directive BM_UPLOAD_HOSTS par défaut) :

```
export BM_UPLOAD_FTP_PASSIVE="true"
```

Dont il faut renseigner l'accès avec login et mot de passe :

```
export BM_UPLOAD_FTP_USER="votre login"
export BM_UPLOAD_FTP_PASSWORD="votre mot de passe"
```

Nous précisons aussi qu'il faut purger les archives trop anciennes pour être conservées :

```
export BM_UPLOAD_FTP_PURGE="true"
```

Avec un BM_UPLOAD_FTP_TTL qui vient de la conf par défaut BM_UPLOAD_FTP_TTL tout comme BM_UPLOAD_FTP_DESTINATION qui vient de BM_UPLOAD_DESTINATION par défaut.

La destination d'upload BM_UPLOAD_FTP_DESTINATION est valorisée par défaut avec BM_UPLOAD_DESTINATION.

Enfin, on oublie pas de désactiver la gravure sur CDROM :

```
export BM_BURNING_METHOD="none"
```

Grâce à l'option BM_POST_BACKUP_COMMAND vous pourrez exécuter un script shell une fois la sauvegarde terminée. Ceci est pratique pour envoyer une notification par e-mail lorsque la sauvegarde est terminée ou pour calculer la quantité de données sauvegardées dans /var/archives/. Laissez libre cours à votre imagination ou prenez exemple sur ce script écrit en PHP.

Au préalable, une petite installation de PHP pour ligne de commande s'impose

```
apt-get install php5-cli
```

Nous utiliserons donc :

```
export BM_POST_BACKUP_COMMAND="/etc/backup-manager-post"
```

En créant le fichier correspondant :

```
nano /etc/backup-manager-post
```

Et en y plaçant le code suivant. N'oubliez pas de modifier les deux premières lignes relatives à l'e-mail de destination et au répertoire stockant les archives :

```
#!/usr/bin/php

<?php

$dest = array('monitoring@domain.net');
$archives = '/var/archives';

$host = trim(file_get_contents('/etc/hostname'));

clearstatcache();

$dir = opendir($archives);
if($dir) {
        $totalsize = "";
        $pagetext = "";
```

```
        while(false !== ($filename = readdir($dir))) {
                if($filename[0]!='.' &&
preg_match('/'.date('Ymd').'/',$filename)) {
                        $thefile = $archives.'/'.$filename;
                        $size = exec("ls -l '".$thefile."' | awk '{print
$5}'");
                        if($size>0) {
                                $pagetext.= $filename."
(".round($size/1000000000,2)." Go)\n";
                        } else {
                                $pagetext.= $filename." (? Go)\n";
                        }
                        $totalsize += $size;
                }
        }
        $pagetext .= "\nTotal : ".round($totalsize/1000000000,2)." Go\n";
}

foreach($dest as $d) {
        mail($d,'['.$host.'] Backup OK',$pagetext);
}

?>
```

Attribuez le droit d'exécution :

```
chmod +x /etc/backup-manager-post
```

Et on oublie pas de déclencher nos sauvegardes automatiquement par cron :

```
nano /etc/cron.daily/backup-manager
```

Avec ce contenu :

```
#!/bin/bash
# cron script for backup-manager
test -x /usr/sbin/backup-manager || exit 0
/usr/sbin/backup-manager
```

Et on le rend executable

```
chmod +x /etc/cron.daily/backup-manager
```

Pour des informations plus détaillées, consultez aussi la documentation officielle de Backup Manager

13. Partitionnement / LVM

13.1. Introduction

Pour l'installation dont on a besoin, il va falloir faire un peu de tuning sur le processus d'installation de base notamment le partitionnement. Pour rappel, l'installation de base via la console online.net nous oblige a faire quelque chose de très basique avec toute la place disque disponible affectée au système. Or, nous voulons mettre en place 2 **espaces disques pour mettre en place la réplication DRBD**.

Comme **DRBD fonctionne avec des volumes physiques**, soit nous utilisons des disques physiques (si on en a non utilisé) soit on en crée avec LVM sur le disque déjà utilisé. Il se trouve que beaucoup de systèmes linux dont proxmox utilisent déjà LVM sur l'installation de base. Il va donc falloir adapter la configuration existante:

- Réduire le système de fichier correspondant à la racine /
- Réduire le volume LVM correspondant à /
- Créer les volumes nécessaire pour nos 2 DRBD

13.2. Bilan du montage (mount)

On regarde le montage par défaut :

```
mount
```

Ce qui nous intéresse c'est ça

```
/dev/mapper/system-root on / type ext4 (rw,relatime,errors=remount-
ro,barrier=1,data=ordered)
```

C'est là ou le système est monté par défaut. On va regarder la place disponnible dessus

```
df -H /
```

Ce qui donne :

```
Sys. fich.              Taille Util. Dispo Uti% Monté sur
/dev/mapper/system-root   952G  1,4G  902G   1% /
```

C'est ladessus qu'on va devoir travailler.

13.3. Bilan du LVM

Cette page de la doc ubuntu est pas mal pour en savoir un peu plus sur LVM.

En gros ce qu'il faut noter c'est qu'il y a un enchainement de 3 couches :

- **PV :** volume physique
- **VG :** groupe de volumes logiques (contenus dans un PV)
- **LV :** volume logique (contenu dans un VG)

On regarde ce que donnent les volumes physiques :

```
pvdisplay
```

```
  --- Physical volume ---
  PV Name               /dev/sda5
  VG Name               system
  PV Size               900,29 GiB / not usable 0
  Allocatable           yes (but full)
  PE Size               4,00 MiB
  Total PE              230475
  Free PE               0
  Allocated PE          230475
  PV UUID               tG2TXB-uY12-oazV-j0un-AMhW-XNk8-gzvsUn
```

Puis on regarde les groupes de volumes

```
vgdisplay
```

```
  --- Volume group ---
  VG Name               system
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               900,29 GiB
  PE Size               4,00 MiB
  Total PE              230475
  Alloc PE / Size       230475 / 900,29 GiB
  Free  PE / Size       0 / 0
  VG UUID               pSBkN7-xpRm-aDYC-LWwH-M6eS-q1Hh-Gd6dod
```

Et enfin on regarde les volumes logiques

```
lvdisplay
```

```
  --- Logical volume ---
  LV Path                /dev/system/root
  LV Name                root
  VG Name                system
  LV UUID                48E9TR-LNqs-uP11-kMKY-3UeF-JsLS-Ko3glR
  LV Write Access        read/write
  LV Creation host, time 62-210-140-177, 2014-05-31 11:27:01 +0200
  LV Status              available
  # open                 1
  LV Size                900,29 GiB
  Current LE             230475
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

En résumé, nous avons :

- **Un volume physique :** /dev/sda5
- **Avec un groupe de volumes :** system
- **Avec un volume logique :** /dev/system/root

Et c'est /dev/system/root qui contient notre système de fichier monté sur / à la racine de notre système.

13.4. Réduction du système de base

Bon, maintenant qu'on y voit mieux sur ce qui se passe, on va passer aux choses sérieuses et c'est un peu casse pied puisqu'il va falloir retoucher l'installation de base...

Pour commencer il faut se connecter à la console web online.net de votre serveur **pour le redémarrer en mode secours**. Cela permettra de travailler sur le système de fichier sans qu'il soit monté. Sinon impossible de le redimensionner.

Je choisis de prendre environ 100Go pour le système de base, ce qui laisse largement de quoi jouer. Ensuite il me restera donc environ 400Go pour chacun de mes deux futurs volumes DRBD.

Puis en mode secours (on est sur un rescue ubuntu donc on passe avec sudo puisque sur proxmox on devient root avec su et il n'y a pas de sudo) :

```
sudo e2fsck -f /dev/mapper/system-root
sudo resize2fs /dev/mapper/system-root 100G
```

Et enfin on réduit la taille du LV (inutile de dire qu'il faut avoir fait un backup avant et bien croiser les doigts ensuite) :

```
sudo lvresize -L 100G /dev/system/root
```

Là retour sur l'interface online.net pour sortir du mode secours et démarrer votre serveur normalement. C'est le moment de s'éponger le front car si on a foiré un truc avant le système sera probablement irrécupérable...

Ah, le serveur répond, ça doit être pas si mal donc... On vérifie que le système est bien redimensionné.

```
df -H /
```

```
Sys. fich.              Taille Util. Dispo Uti% Monté sur
/dev/mapper/system-root   106G  1,4G   99G   2% /
```

Victoire ! Evidemment l'opération est à faire sur chacun des deux serveurs. L'étape suivante sera ensuite de créer des LV pour DRBD.

13.4. Préparation des volumes pour DRBD

Je conseille cet excellent article pour avoir un peu de documentation sur le sujet LVM et DRBD.

En gros l'idée ici va être de créer deux volumes logiques dans notre volume group par défaut pour les attribuer ensuite à notre système de réplication entre serveurs via DRBD.

N'oublions pas que nous avons réduit la taille du LV à **100GiB** qui est dans un VG de **900,29GiB**. Reste donc 800,29GiB à attribuer à nos deux volumes DRBD. Pour faire simple et ne pas tailler les cheveux en 4, on fera donc **400,14GiB** d'un côté et **400,15GiB** de l'autre.

On crée le premier volume logique dans le VG (qui porte le doux nom de system n'oublions pas) :

```
lvcreate --name drbd1 --size 400,14G system
```

Puis le deuxième

```
lvcreate --name drbd2 --size 400,15G system
```

Et on vérifie que tout a été créé

```
lvdisplay
```

```
  --- Logical volume ---
  LV Path                /dev/system/root
  LV Name                root
  VG Name                system
  LV UUID                48E9TR-LNqs-uP11-kMKY-3UeF-JsLS-Ko3glR
  LV Write Access        read/write
```

```
  LV Creation host, time ,
  LV Status              available
  # open                 1
  LV Size                100,00 GiB
  Current LE             25600
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0

  --- Logical volume ---
  LV Path                /dev/system/drbd1
  LV Name                drbd1
  VG Name                system
  LV UUID                5TN2B2-5Opz-knwM-K3B3-wZVS-iMqc-qB3toa
  LV Write Access        read/write
  LV Creation host, time psrv101, 2014-06-24 23:12:25 +0200
  LV Status              available
  # open                 0
  LV Size                400,14 GiB
  Current LE             102436
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:1

  --- Logical volume ---
  LV Path                /dev/system/drbd2
  LV Name                drbd2
  VG Name                system
  LV UUID                oiFDTJ-P2hg-z0wp-2vsd-JnCe-GEZ1-bQXScg
  LV Write Access        read/write
  LV Creation host, time psrv101, 2014-06-24 23:13:38 +0200
  LV Status              available
  # open                 0
  LV Size                400,15 GiB
  Current LE             102439
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:2
```

C'est tout bon.

Note importante : Ne pas oublier de créer ces deux volumes sur chacun des deux serveurs physiques. Ils doivent avoir la même taille chacun pour que la réplication soit OK.

14. Configuration réseau / VPN

14.1 Choix réseau & interfaces réseaux

Je revoie une nouvelle foi à cet excellent article de Victor Héry dont cette partie est inspirée. Je rabache aussi quelque trucs déjà vu dans les parties précédentes.

Les spécificités pour moi :

- J'utilise le **réseau RPN** Online.net pour les échanges du cluster proxmox (peu d'infos échangées donc le débit de 100Mbps de base est suffisant et ça fait une redondance intéressante).
- J'utilise un **réseau VPN** monté sur l'interface publique pour les réplications DRBD (c'est du 1Gbps et c'est pas de trop pour de la répli…) et le réseau interne des VM.

Avec les qualités et inconvénients déjà vus plus haut sur ce genre de config. Mais bon si quelqu'un veut me payer le RPN à 1Gbps n'hésitez pas :D

Après une installation de Proxmox, vous devez disposer d'un bridge **vmbr0**.

Pour utiliser openVPN et simplifier la gestion des adresses IP entre les serveurs, nous allons créer un bridge **vmbr1** qui sera utilisé par nos VM pour communiquer entres elles (et aussi pour la réplication DRBD comme je le disais plus haut). C'est une solution simplifiée par rapport a celle proposée par Victor que j'ai déjà utilisé auparavant.

Pour rappel à l'origine Victor propose deux bridges, un pour communiquer avec les passerelles only et un pour que les vm middle communiquent entre elles. A l'usage cependant, j'ai rencontré certains soucis à cette architecture.

En effet les domaines associés à mes serveurs web pointent sur une IP failover que je peux changer de serveur. Sauf que pour que cela fonctionne il faut que chacune de mes VM front puisse accepter cette ip failover. Ce qui sous entend une préparation des ip en double sur chaque vm front et n'est pas sans provoquer des soucis réseau puisque cela me fait monter des interfaces en double avec des ip identiques…

Du coup, je suis obligé de faire une configuration réseau dormante sur chaque VM front et l'activer au besoin ce qui est franchement naze selon moi…

Bref, la nouvelle solution que je me propose de mettre en place est de mettre en place des VM front joignables de n'importes quel seveur physique pour que mes VM middle puissent en profiter ou qu'elles soient également.

Au plus simple donc, il n'y a donc au final qu'un bridge a mettre en place : vmbr1…

Enfin, nous allons mettre en place **dummy1**, une interface “bidon” va permettre d'avoir une interface fixe pour “brancher” openvpn, qui a besoin d'une interface réseau physique pour se lancer correctement au démarrage du serveur.

14.2. Activer le RPN

Assez basique, il suffit d'aller dans la console web online.net et créer un groupe RPN entre vos 2 serveurs dédiés. Je ne m'attarderai pas dessus.

Partant de là vos serveurs pourront causer directement ensemble sur leur deuxième interface physique une fois qu'elle sera configurée.

Pour avoir les ip de ces interfaces, rendez vous sur la console web du serveur.

14.3. Configuration de l'interface réseau bidon dummy

Nous allons commencer par créer l'interface dummy (un peu particulière) nécessaire à OpenVPN. De base, Proxmox peut utiliser une interface dummy pour des usages internes, nous allons donc utiliser une deuxième par précaution :

```
echo "options dummy numdummies=2" > /etc/modprobe.d/dummy.conf
```

```
modprobe dummy
```

Pour vérifier que les interfaces ont bien été créées, tapez la commande suivante :

```
ifconfig -a | grep Link | grep -v inet6
```

Qui doit vous renvoyer quelque chose comme ça :

```
dummy0 : Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
dummy1 : Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
eth0 : Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
lo : Link encap:Local Loopback
vmbr0 : Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
```

14.4. Configuration des autres interfaces réseau

Nous allons aller toucher au fichier /etc/network/interfaces.

Avant ça, sauvegardez le fichier :

```
cpav /etc/network/interfaces /etc/network/interfaces.original
```

Puis éditez le :

```
nano /etc/network/interfaces
```

On fait un peu de ménage sur l'existant (commentaires pour s'y retrouver) et on active le RPN en

dhcp pour pouvoir l'utiliser. Normalement il est sur eth1. Voir la documentation online.net.

```
# The loopback network interface
auto lo
iface lo inet loopback

# Main network interface used with vmbr0
iface eth0 inet manual

# Bridge with the main interface for public network
auto vmbr0
iface vmbr0 inet static
    address IP_SERVEUR_PHYSIQUE_1
    netmask 255.255.255.0
    gateway GW_SERVEUR_PHYSIQUE_1
    bridge_ports eth0
    bridge_stp off
    bridge_fd 0

# RPN Interface
auto eth1
iface eth1 inet dhcp
```

Ajoutez vmbr1 qui servira pour consruire le réseau entre les VM :

```
# Bridge to communicate between VM
auto vmbr1
iface vmbr1 inet static
        address  192.168.100.1
        netmask  255.255.255.0
        bridge_ports dummy1
        bridge_stp off
        bridge_fd 0
        post-up route add -net 224.0.0.0 netmask 240.0.0.0 dev vmbr1
```

Je prends 192.168.100.1 pour le serveur principal et 192.168.100.2 pour le secondaire.

Pour appliquer les modifications, effectuez un :

```
service networking restart
```

Pour vérifier que les nouvelles interfaces fonctionnent correctement, vous pouvez les pinguer sur leurs serveurs respectifs :

```
ping 192.168.100.1
```

```
ping 192.168.120.2
```

Une fois le VPN monté (plus tard) on pourra joindre toutes les IP depuis n'importe quel serveur.

14.5 Configuration des services

Pour rappel, j'ai choisi de faire en sorte d'avoir les échanges d'infos entre les noeuds du cluster via le réseau RPN. Il faut pour cela faut configurer les hosts qui seront utilisés par proxmox pour qu'ils soient résolus avec l'IP dans le réseau qui nous intéresse (ici le RPN) :

```
nano /etc/hosts
```

Mettez bien le nom du host (ici machine1 et machine2) de votre serveur sur l'ip du RPN (si c'est ce réseau que vous voulez utiliser) :

```
# Localhost
127.0.0.1 localhost
IP_SERVEUR_PHYSIQUE_1 sd-yyyyy.dedibox.fr sd-yyyyy machine1.wan
machine1.mondomaine.net
IP_RPN_SERVEUR_PHYSIQUE_1 machine1 machine1.rpn
192.168.100.2 machine1.vpn

# Other proxmox physical node
IP_SERVEUR_PHYSIQUE_2 sd-yyyyy.dedibox.fr sd-yyyyy machine2.wan
machine2.mondomaine.net
IP_RPN_SERVEUR_PHYSIQUE_2 machine2 machine2.rpn
192.168.100.2 machine2.vpn
```

Pour bootlogd, plus besoin de configuration depuis Wheezy comme vu dans cet article.

15. Installation d'OpenVPN

15.1 Installation du paquet

On est parti ! Tout ce qui était avant va vous servir :)

Installons OpenVPN pour commencer :

```
apt-get install openvpn
```

Faites ceci sur tous vos serveurs.

15.2. Préparation des certificats

Sur le serveur PRA, le serveur principal d'OpenVPN, nous allons générer les certificats serveurs et clients.

```
cpav /usr/share/doc/openvpn/examples/easy-rsa/2.0/ /etc/openvpn/easy-rsa/
```

```
cd /etc/openvpn/easy-rsa/
```

```
cpav vars vars.original
```

Editez le fichier vars, tout à la fin du fichier, pour modifier les variables globales. Ainsi votre génération de clef se fera automatiquement. Sinon, vous devrez indiquer les informations à chaque génération.

```
nano /etc/openvpn/easy-rsa/vars
```

Version un peu bourrin :

```
export KEY_COUNTRY="FR"
export KEY_PROVINCE="FR"
export KEY_CITY="Paris"
export KEY_ORG="mondomaine.net"
export KEY_EMAIL="me@myhost.mydomain"
export KEY_CN=mondomaine.net
export KEY_NAME=mondomaine.net
export KEY_OU=mondomaine.net
export PKCS11_MODULE_PATH=mondomaine.net
export PKCS11_PIN=1234
```

Puis :

```
source ./vars
```

Vous verrez ceci :

```
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-
rsa/keys
```

C'est parti :

```
./clean-all
./build-dh
```

```
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.....................................................+......
..+............+.................................................
.....................................................................
...+.............................................++*++*++*
```

```
./pkitool --initca
```

```
Using CA Common Name: Mon organisation CA
Generating a 1024 bit RSA private key
```

```
...++++++
.................++++++
writing new private key to 'ca.key'
-----
```

15.3. Générez la clef serveur

La clé serveur sera utilisée sur la machine 1.

```
./pkitool --server server
```

```
openvpn --genkey --secret ./keys/ta.key
```

15.4. Générez la clef client

La clé client sera utilisée sur la machine 2.

Comme indiqué sur ce site nous allons créer deux certificats ayant le même sujet (même DN), passez la valeur **“unique_subject” à “no”** dans le fichier **“index.txt.attr”** (qui se trouve dans le dossier keys) de votre autorité de certification. C'est mal faudra améliorer ça mais on verra plus tard.

Sans cette modification votre fichier CRT sera vide suite a une erreur de ce type

```
failed to update database
TXT_DB error number 2
```

On génère la clé :

```
./pkitool client-machine2
```

Vous pouvez également en profiter pour générer d'autres clefs clients si vous souhaitez vous connecter vous-même au VPN, pour sécuriser à fond vos accès SSH par exemple.

15.5. Sauvegarde des clefs

```
cd /etc/openvpn/easy-rsa/keys/
```

```
tar cvzf /root/server-keys.tar.gz server.crt server.key ca.crt dh1024.pem
ta.key
```

```
tar cvzf /root/client-machine2-keys.tar.gz client-machine2.crt client-
machine2.key ca.crt ta.key
```

15.6. Configuration du serveurs OpenVPN

Comme nous ne mettons pas en place de triangle VPN comme Victor le suggère, pas faute de moyens et parceque nous ne faisons pas de HA faute de moyens et de fencing, on ne configurera qu'une seule machine physique en serveur openvpn. L'autre sera donc forcement configurée en client :)

Importez le fichier de configuration par défaut :

```
cd /etc/openvpn/
```

On crée les dossiers utiles :

```
mkdir logs
mkdir scripts
```

Et on s'occupe du fichier de configuration :

```
zcat /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz > server.conf
```

Le fichier doit ressemble à ça sans ses commentaires (et en changeant les valeurs style port à vote convenance) :

```
cat server.conf | grep -v -e '^;' | grep -v -e '^#' | grep -v -e '^$'
```

```
local IP_PUBLIQUE_SERVEUR
port 4450
proto tcp
dev tap1
ca ca.crt
cert server.crt
key server.key # This file should be kept secret
dh dh1024.pem
server-bridge XXX.XXX.XXX.XXX 255.255.255.0 YYY.YYY.YYY.YYY TTT.TTT.TTT.TTT
client-to-client
duplicate-cn
keepalive 10 120
tls-auth /etc/openvpn/ta.key 0 # This file is secret
comp-lzo
max-clients 4
user nobody
group nogroup
persist-key
persist-tun
status logs/openvpn-status.log
log-append logs/openvpn.log
verb 3
tls-server
script-security 2
```

```
up "/etc/openvpn/scripts/up.sh"
down "/etc/openvpn/scripts/down.sh"
```

Attention, ici on choisit de fonctionner en UDP ce qui représente un risque en fiabilité par rapport à TCP. Cependant, je considère qu'étant donné la topologie réseau entre mes serveurs (réseau local d'un hébergeur pro) c'est acceptable. D'autant plus que le passage à UDP réduit l'overhead réseau et surtout accélère la vitesse / augmente la bande passante dispo.

Les choses à modifier selon le serveur :

- **local IP_PUBLIQUE_SERVEUR :** remplacez par l'adresse IP publique de votre serveur, pour écouter en VPN sur cette adresse. Cette adresse est a priori différente selon le serveur ! wink
- **Le port d'écoute :** Par défaut 1194, mais il est conseillé de le changer pour éviter d'être reconnu direct comme un VPN
- **XXX.XXX.XXX.XXX :** remplacez par l'adresse privée du vmbr1 du serveur. Attention à ne pas vous tromper !
- **YYY.YYY.YYY.YYY :** cette IP est le début de la plage allouée pour OpenVPN. Pour ma part, je prend 4 clients possibles. Les 2 serveurs et 2 autres (pour vos connexions de gestion par exemple)
- **TTT.TTT.TTT.TTT :** Cette IP est la fin de la plage allouée commencée par YYY.YYY.YYY.YYY. Cette plage doit contenir les IP de vos serveurs, mais comme ceux-ci sont en IP fixe, pas de risque d'allocation se marchant dessus.

15.7. Scripts up.sh et down.sh (pour le serveur)

Ces scripts vont permettre d'ajouter l'interface OpenVPN (tap1) au bridge vmbr1 à chaque démarrage du serveur VPN. Et bien sûr de le retirer à l'arrêt du serveur VPN (faisons les choses bien)

Ces scripts seront sensiblement différents pour les clients, nous verrons ça plus bas.

```
nano /etc/openvpn/scripts/up.sh
```

```
#!/bin/bash
/sbin/ifconfig vmbr1 promisc
/sbin/ifconfig tap1 up promisc
/sbin/brctl addif vmbr1 tap1
```

```
nano /etc/openvpn/scripts/down.sh
```

```
#!/bin/bash
/sbin/brctl delif vmbr1 tap1
/sbin/ifconfig tap1 down -promisc
/sbin/ifconfig vmbr1 -promisc
```

Et on oublie pas de donner le droit d'execution :

```
chmod +x etc/openvpn/scripts/*
```

15.8. Installation des clefs serveurs

Le serveur devra disposer du set de clef serveur que nous avons généré plus haut (mais dans le dossier spécifique). Sans quoi, il n'acceptera pas les connexions du client.

```
cd /etc/openvpn
```

```
tar xvzf /root/server-keys.tar.gz
```

Une fois ceci fait, vous pouvez redémarrer OpenVPN :

```
service openvpn restart
```

Normalement, il doit démarrer sans erreurs des deux côtés. Les logs s'écrivent dans **/etc/openvpn/openvpn.log**

Un démarrage réussit doit se terminer par

```
Initialization sequence completed
```

Sinon, vérifiez bien les adresses IP que vous avez indiquée (problème de bind) et la synthaxe des paramètres dans le fichier de conf (problème de “unknown parameter”)

15.9. Configuration du client

A ce stade, nous avons uniquement le serveurs VPN qui tourne Il va vous falloir configurer le client qui s'y connecte c'est a dire machine2.

Importez le fichier de configuration par défaut :

```
cd /etc/openvpn/
```

On crée les dossiers utiles :

```
mkdir logs
mkdir scripts
```

Et on s'occupe du fichier de configuration :

```
cpav /usr/share/doc/openvpn/examples/sample-config-files/client.conf .
```

Pour une configuration idéale (d'après mes tests de robustesse personnels), le fichier doit ressemble à ça sans ses commentaires (et en changeant les valeurs style port à vote convenance) :

```
cat client.conf | grep -v -e '^;' | grep -v -e '^#' | grep -v -e '^$'
```

Voici le fichier de configuration :

```
client
dev tap1
proto udp
remote IP_PUBLIQUE_SERVEUR_1
port PORT_VPN
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client-machine2.crt
key client-machine2.key
ns-cert-type server
tls-auth ta.key 1
comp-lzo
verb 3
log-append logs/client.log
script-security 2
up "/etc/openvpn/scripts/up-client.sh"
down "/etc/openvpn/scripts/down-client.sh"
keepalive 30 120
float
```

Les paramètres à adapter :

- **dev tap1 :** c'est l'interface que créera openvpn pour sa connexion. Il est important de ne pas mettre tap0, qui est déjà utilisé pour le serveur VPN sur le serveur principal. Pour uniformiser, mieux vaut utiliser la même interface sur les 2 clients
- **remote …. :** c'est là que vous indiquez sur quel serveur se connecter. Indiquez l'adresse IP du PRA ainsi que le port que vous avez choisi. Sur l'arbitre, rajoutez une ligne avec l'adresse IP du serveur principal. OpenVPN essaiera toujours la première ligne, puis si elle échoue il passera à la deuxième.
- **cert, key :** c'est là que vous définirez les certificats et les clefs à utiliser pour la connexion client (clientPrincipale ou clientArbitre) que nous allons rapatrier via scp.
- **log-append :** indique le fichier dans lequel les logs de connexion seront écrits. Indiquez un nom de fichier différent de celui du serveur pour avoir des logs bien séparés
- **up, down :** les fichiers up.sh et down.sh sont différents entre les serveurs et les clients, donc indiquez là aussi des noms différents ce ceux dans le fichier de conf du serveur
- **keepalive 30 120 :** cette directive va indiquer aux clients qu'il doivent tenter de relancer leur connexion si jamais elle tombe (On ping toutes les 30s, on redémarre au bout de 120s si pas de réponse). Dans le cas du serveur principal, il essaiera en boucle de se reconnecter sur le PRA. Dans le cas de l'arbitre, il alternera entre le PRA et le serveur principal jusqu'à ce qu'une connexion réussisse

15.10. Scripts up-client et down-client

Les fichiers up.sh et down.sh sont différents pour les clients. Comme le serveur principal possède déjà up.sh et down.sh, on les nomme up-client.sh et down-client.sh pour qu'ils ne se marchent pas dessus

(quelle imagination !) et on les mets dans le dossier /etc/openvpn/scripts.

```
nano /etc/openvpn/scripts/up-client.sh
```

```
#!/bin/bash
/sbin/ifconfig vmbr1 promisc
/sbin/ifconfig tap1 up promisc
/sbin/brctl addif vmbr1 tap1
/sbin/ifconfig tap1 0
```

```
nano /etc/openvpn/scripts/down-client.sh
```

```
#!/bin/bash
/sbin/ifconfig vmbr1 -promisc
```

La principale différence est le nom des interfaces à bridger (tap1 dans mon cas). Le down client ne doit également pas sortir le tap1 du bridge, sinon il risque de générer une erreur qui va bloquer la tentative de connexion en boucle du client.

Et on oublie pas de donner les droits d'execution :

```
chmod +x /etc/openvpn/scripts/*
```

15.11. Récupération de la clef client

Bien, il ne reste plus qu'à récupérer la clef openvpn à qui de droit (en ayant au préalable bougé les clés de du dossier root vers le dossier user que vous utiliserez pour scp et sans oublier les droits + ménage ensuite) :

```
scp -P portssh user@IP_SERVEUR_PHYSIQUE_1:~/client-psrv102-keys.tar.gz .
```

Et on oublie pas de bouger les clés dans le dossier root et ensuite faire le ménage !

Puis extrayez les clefs dans /etc/openvpn :

```
cd /etc/openvpn
```

```
tar xvzf /root/client-machine2-keys.tar.gz
```

Et enfin, redémarrez openvpn.

La connexion va s'établir. Là encore, les logs s'écrivent dans /etc/openvpn, dans le fichier client.log spécifé dans le fichier de configuration.

Si tout se passe bien, vous pourrez y voir apparaitre un “Initialization sequence completed”.

C'est également dans ce fichier que vous pouvez vérifier si la reconnexion se fait bien.

Enfin vous pouvez vérifier que le ping se passe bien désormais sur les 2 machines :

```
ping 192.168.100.1
ping 192.168.100.2
```

16. Mise en place DRBD

16.1. Rappels

Nous avons donc un système de cluster avec 2 noeuds physiques. Nous avons mis en place deux volumes DRBD dans les parties précédentes. Chaque serveur contiendra deux volumes DRBD mais un seul à la fois sera utilisé en écriture. Ce système évite normalement les cas de split brain qui sont généralement d'une grande tristesse à résoudre vu que cela implique en général de la perte de données...

Pour cette partie, on s'appuiera en grande partie sur la documentation DRBD de proxmox. La partie nested LVM a été vue finalement plus haut avec le travail LVM qu'on à fait en amont.

16.2. Installation DRBD

Rien de plus simple et sur chacun des serveurs physiques :

```
apt-get install drbd8-utils
```

16.3. Configuration DRBD

On remplace **/etc/drbd.d/global_common.conf** par le contenu suivant :

```
global { usage-count no; }
common {
        syncer { rate 30M; verify-alg md5; }
        handlers { out-of-sync "/usr/lib/drbd/notify-out-of-sync.sh root"; }
}
```

Le rate c'est à dire la bande passante utilisée est très important à configurer correctement. Plus d'informations sur cette page. Ici le 30 est en Mbyte et non bit. La valeur ici par défaut correspond à une **liaison Gibabit !!** Avec un coef maxi de 30% de la bande passante utilisée.

La partie **disk** doit correspondre aux volumes LVM que nous avons créé spécialement pour DRBD plus haut.

Pour notre premier volume DRBD on crée le fichier de configuration et sur chacun des serveurs (adaptez la conf avec le nom de vos machines, éventuellement les IP etc...) **/etc/drbd.d/r1.res** :

```
# From http://pve.proxmox.com/wiki/DRBD#Start_DRBD
resource r1 {
        protocol C;
        startup {
                wfc-timeout  0;     # non-zero wfc-timeout can be dangerous
(http://forum.proxmox.com/threads/3465-Is-it-safe-to-use-wfc-timeout-in-DRBD
-configuration)
                degr-wfc-timeout 60;
                become-primary-on both;
        }
        net {
                cram-hmac-alg sha1;
                shared-secret "my-secret";
                allow-two-primaries;
                after-sb-0pri discard-zero-changes;
                after-sb-1pri discard-secondary;
                after-sb-2pri disconnect;
                #data-integrity-alg crc32c;     # has to be enabled only for
test and disabled for production use (check man drbd.conf, section "NOTES ON
DATA INTEGRITY")
        }
        on MACHINE1 {
                device /dev/drbd1;
                disk /dev/system/drbd1;
                address 192.168.100.1:7788;
                meta-disk internal;
        }
        on MACHINE2 {
                device /dev/drbd1;
                disk /dev/system/drbd1;
                address 192.168.100.2:7788;
                meta-disk internal;
        }
}
```

Pour notre second volume DRBD on crée le fichier de configuration et sur chacun des serveurs **/etc/drbd.d/r2.res** :

```
# From http://pve.proxmox.com/wiki/DRBD#Start_DRBD
resource r2 {
        protocol C;
        startup {
                wfc-timeout  0;     # non-zero wfc-timeout can be dangerous
(http://forum.proxmox.com/threads/3465-Is-it-safe-to-use-wfc-timeout-in-DRBD
-configuration)
                degr-wfc-timeout 60;
                become-primary-on both;
        }
        net {
                cram-hmac-alg sha1;
                shared-secret "my-secret";
```

```
                allow-two-primaries;
                after-sb-0pri discard-zero-changes;
                after-sb-1pri discard-secondary;
                after-sb-2pri disconnect;
                #data-integrity-alg crc32c;     # has to be enabled only for
test and disabled for production use (check man drbd.conf, section "NOTES ON
DATA INTEGRITY")
        }
        on MACHINE1 {
                device /dev/drbd2;
                disk /dev/system/drbd2;
                address 192.168.100.1:7789;
                meta-disk internal;
        }
        on MACHINE1 {
                device /dev/drbd2;
                disk /dev/system/drbd2;
                address 192.168.100.2:7789;
                meta-disk internal;
        }
}
```

16.4. Démarrage de DRBD

On démarre DRBD sur chaque serveur mais ce ne sera pas suffisant pour avoir la conf complète :

```
service drbd start
```

Note importante : bon pour moi ca a été le souk à ce stade. J'ai beau suivre le tuto proxmox il m'a insulté au démarrage. Finalement j'ai fait les étapes suivantes et j'ai bidouillé et cela a fini par fonctionner mais c'est super moche. A voir comment faire mieux

On crée nos ressources sur chaque serveur

```
drbdadm create-md r1
drbdadm create-md r2
```

On monte les volumes DRBD

```
drbdadm up r1
drbdadm up r2
```

On regarde l'état DRBD sur les différents noeud et on constate que c'est le souk (pas de synchro initiale donc tout est en vrac) :

```
cat /proc/drbd
```

On a un truc qui ressemble à ça pour chacun des noeuds :

```
 1: cs:Connected ro:Secondary/Secondary ds:Inconsistent/Inconsistent C r----
-
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b
oos:419565012
 2: cs:Connected ro:Secondary/Secondary ds:Inconsistent/Inconsistent C r----
-
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b
oos:419577300
```

Pour remédier à cela on va démarrer la synchro initiale en passant un noeud en primaine pour chacune des ressources (donc on ne fait pas ça sur les 2 !). Cela va lui permettre de prendre la main et la réplication va pouvoir démarrer) :

```
drbdadm -- --overwrite-data-of-peer primary r1
drbdadm -- --overwrite-data-of-peer primary r2
```

Selon la taille du disque cela va être assez long donc on suit cela jusqu'à ce que ce soit terminé

```
watch cat /proc/drbd
```

On devrait avoir une jauge de progression comme ceci :

```
 1: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:12539660 nr:0 dw:0 dr:12548504 al:0 bm:765 lo:1 pe:4 ua:64 ap:0 ep:1
wo:b oos:407025748
        [>....................] sync'ed:  3.0% (397484/409728)M
        finish: 3:33:23 speed: 31,784 (27,680) K/sec
 2: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:11591568 nr:0 dw:0 dr:11600408 al:0 bm:707 lo:1 pe:4 ua:64 ap:0 ep:1
wo:b oos:407986132
        [>....................] sync'ed:  2.8% (398420/409740)M
        finish: 4:44:19 speed: 23,892 (25,756) K/sec
```

Evidemment si vous avez de gros diques, la réplication initiale peut prendre *beaucoup de temps* (plusieurs heures) donc il vaut mieux prévoir à l'avance :). Une fois terminé on stope DRBD sur chaque noeud :

```
service drbd stop
```

Puis on les redémarre sur chaque noeud :

```
service drbd start
```

Et on devrait voir désormais que les deux noeuds sont en primaire / primaire et up to date.

```
cat /proc/drbd
```

Ce qui donne quelque chose du genre si tout va bien :

```
 1: cs:Connected ro:Primary/Primary ds:UpToDate/UpToDate C r-----
```

```
    ns:0 nr:0 dw:0 dr:664 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:0
 2: cs:Connected ro:Primary/Primary ds:UpToDate/UpToDate C r-----
    ns:0 nr:0 dw:0 dr:664 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:0
```

16.5. Configuration LVM pour proxmox sur les volumes DRBD

Nous avons donc maintenant deux volumes DRBD synchronisés entre eux sur les deux serveurs. En l'état ils ne servent à rien pour notre cluster proxmox. En effet, promox peut utiliser LVM pour monter des VM mais seulement des volumes group (VG). On va donc devoir procéder à la création de ces VG sur nos volumes DRBD pour pouvoir les donner à manger à promox.

Sur **UN SEUL** des deux systèmes (pas besoin de faire sur les deux puisque c'est répliqué) on crée un volume physique sur les ressources DRBD qu'on a configuré :

```
pvcreate /dev/drbd1
pvcreate /dev/drbd2
```

On vérifie que c'est bien créé :

```
pvscan
```

Ce qui donne ceci :

```
  PV /dev/sda5    VG system          lvm2 [900,29 GiB / 0    free]
  PV /dev/drbd1                      lvm2 [400,13 GiB]
  PV /dev/drbd2                      lvm2 [400,14 GiB]
  Total: 3 [1,66 TiB] / in use: 1 [900,29 GiB] / in no VG: 2 [800,27 GiB]
```

Et vous pouvez même lancer un petit pvscan sur l'autre système pour vérifier que vos volumes sont apparus aussi, comme par magie :)

Enfin, sur un seul des noeuds (toujours pareil l'autre ca se fera tout seul grace a la répli) on créé un volume group sur chaque volume physique. Ce sont ces VG qui seront utilisé par proxmox :

```
vgcreate pve-drbd1 /dev/drbd1
vgcreate pve-drbd2 /dev/drbd2
```

On vérifie (sur chacun des noeuds) que c'est bien créé :

```
vgscan
```

Ce qui doit donner quelque chose comme ça :

```
  Reading all physical volumes.  This may take a while...
  Found volume group "system" using metadata type lvm2
  Found volume group "pve-drbd2" using metadata type lvm2
```

```
  Found volume group "pve-drbd1" using metadata type lvm
```

Enfin, pour ajouter ces volumes groups dans proxmox, **rendez vous sur l'interface web**, puis dans la rubrique *Stockage* cliquez sur *Ajouter* et *LVM*. Vous devriez pouvoir voir vos volumes précédemment créés. A noter que puisque ce sont des volumes répliquez vous devez cocher l'option **shared** sur ces volumes pour évider que proxmox se plante lors des migrations de VM. Choisissez également pour chaque volume les 2 noeuds et répétez l'opération pour les deux DRBD que nous avons créé.

17. Mise en place du cluster proxmox

17.1. Multicast / Unicast

Pour notre configuration, nous allons faire passer la configuration cluster proxmox sur le réseau RPN. Cela permettra de soulager les interfaces primaires déjà utilisé pour le traffic public et le VPN qui contient lui même la partie DRBD ainsi que les communications entre nos VM.

Le problème est que le réseau RPN ne permet pas non plus le multicast. Notre VPN le permet mais comme dit plus haut il est bien assez chargé comme ça. Reste donc la **possibilité de monter un deuxième VPN sur le réseau RPN pour utiliser le multicast** (non abordé dans ce guide puisque ce n'est pas le choix retenu d'autant plus que le débit du réseau RPN de base est limité à 100Mbps).

Pour tester si le réseau dispose bien du multicast, installez l'outil suivant :

```
apt-get install ssmping
```

Il vous faut lancer sur un serveur pour le mettre en écoute :

```
ssmpingd
```

Puis depuis un autre, lancez le test :

```
asmping 224.0.2.1 IP_SERVEUR_TEST
```

Vous devez voir passer sur le serveur en écoute des messages de ce type :

```
asmping joined (S,G) = (*,224.0.2.234)
pinging IP_SERVEUR_TEST from IP_SERVEUR_SOURCE
unicast from IP_SERVEUR_TEST, seq=1 dist=0 time=38.699 ms
multicast from IP_SERVEUR_TEST, seq=1 dist=0 time=76.405 ms
unicast from IP_SERVEUR_TEST, seq=2 dist=0 time=38.802 ms
multicast from IP_SERVEUR_TEST, seq=2 dist=0 time=76.238 ms
```

Les lignes importantes à repérer sont celles du multicast, que vous devez impérativement constater pour le bon fonctionnement du cluster !

Dans notre cas, nous n'aurons pas de multicast mais du unicast pour proxmox. Cependant,

ce n'est pas un souci dans notre cas puisque nous n'avons que 2 noeuds. Il est donc tout à fait possible et sans surcoût de ne faire qu'un simple unicast. Et heureusement c'est possible en suivant cette documentation.

Par contre il faut bien comprendre qu'utiliser le mode unicast avec plus de deux noeuds peut être contre productif en terme d'utilisation réseau. En effet au lieu d'avoir des messages diffusés une fois à tout les noeuds il devra y avoir de la communication bidirectionnelle entre chaque noeud avec toute la répétition et non fiabilité d'un point de vue cluster HA que cela représente.

Nous verrons comment faire plus loin pour activer le mode unicast car il faut d'abord créer le cluster.

17.2. Création du cluster

C'est le bon moment pour vérifier le fichier /etc/hosts. Pour rappel, le nom de vos hosts utilisé pour le cluster doivent êtres associées aux ip du réseau que vous voulez utiliser. Ici si vous vous basez sur mes choix, vous devez avoir indiqué l'adresse IP du du host correspondant à l'IP du RPN.

Pour le cluster, nous allons effectuer sa création depuis le **premier serveur physique** et nous y ajouterons les autres noeuds.

Je vous rappelle qu'il ne doit y avoir aucune VM sur les serveurs ! Si votre système a déjà des VM en fonctionnement, migrez les temporairement sur le premier serveur (celui ou vous créez le cluster) le temps de créer le cluster. Cette limitation a pour but de s'assurer qu'aucun VMID ne sera en conflit au sein du cluster.

Le nom du cluster est important, dans le sens où une fois le cluster créé, vous ne pourrez plus changer ce nom. Choisissez le bien !

```
pvecm create NOM_DU_CLUSTER
```

qui doit vous renvoyer :

```
Restarting pve cluster filesystem: pve-cluster[dcdb] notice: wrote new
cluster config '/etc/cluster/cluster.conf'
.
Starting cluster:
   Checking if cluster has been disabled at boot... [  OK  ]
   Checking Network Manager... [  OK  ]
   Global setup... [  OK  ]
   Loading kernel modules... [  OK  ]
   Mounting configfs... [  OK  ]
   Starting cman... [  OK  ]
   Waiting for quorum... [  OK  ]
   Starting fenced... [  OK  ]
   Starting dlm_controld... [  OK  ]
   Tuning DLM kernel config... [  OK  ]
   Unfencing self... [  OK  ]
```

Puis on active le mode unicast, comme spécifié dans cette page

```
cp /etc/pve/cluster.conf /etc/pve/cluster.conf.new
```

On modifie la nouvelle configuration pour activer le mode unicast :

```
nano /etc/pve/cluster.conf.new
```

En ajoutant la nouvelle directive *transport="udpu"* :

```
<cman keyfile="/var/lib/pve-cluster/corosync.authkey" transport="udpu">
</cman>
```

Pensez également aussi a incrémenter **"config_version"** ce qui garantira l'application des modifications.

Vous pouvez valider la configuration avec la commande suivante :

```
ccs_config_validate -v -f /etc/pve/cluster.conf.new
```

La configuration sera à activer uniquement via l'interface web, dans la partie HA. Vous pouvez vérifier la syntaxe et l'appliquer. Elle sera ensuite prise en compte par tous les noeuds.

Enfin, **depuis le second serveur physique** on passe cette commande pour qu'il s'ajoute dans le cluster configuré sur le premier serveur physique :

```
pvecm add 192.168.100.1
```

Cette commande doit vous renvoyer à peu près la même chose qu'au dessus, lors de la création du cluster sur le premier noeud.

17.3. Etat du cluster

La commande pvecm vous permettra d'avoir plusieurs infos sur le cluster.

```
pvecm n
```

vous donnera un aperçu des nodes (date de connexion, état, etc)

```
pvecm s
```

vous donnera des informations sur le cluster lui-même (état du qorum, nombre de node, nombre de vote, etc)

Nous verrons plus tard des commandes plus spécifiques au HA.

18. Mise en place des VM

18.1. Récupération des iso

La première étape consiste à récupérer les images ISO de la distribution que vous choisirez pour votre installation.

Pour ma part je pars une ubuntu dernière version pour architecture 64 bits.

```
http://releases.ubuntu.com/14.04/ubuntu-14.04-server-amd64.iso
```

On va placer cette image (sur chaque serveur physique) dans le dossier dédié à accueillir les images iso pour que proxmox puisse les retrouver lorsqu'on va vouloir installer :

```
cd /var/lib/vz/template/iso
```

On récupère l'iso :

```
wget http://releases.ubuntu.com/14.04/ubuntu-14.04-server-amd64.iso
```

Pour vérifier votre ISO, allez sur l'IHM web proxmox puis cliquez sur le disque local de vos serveurs physiques et enfin choisissez l'onglet contenu. Vous devrez voir votre iso.

18.2. Architecture des VM en deux mots

En quelques mots, l'architecture de serveurs virtuels que j'adopte :

- Une petite VM 1 front sur SERVEUR_PHYSIQUE_1 et sur les volumes répliqués pve-drbd1
- Une bonne VM 1 middle / back sur SERVEUR et sur les volumes répliqués pve-drbd1
- Une petite VM 2 front sur SERVEUR_PHYSIQUE_1 et sur les volumes répliqués pve-drbd2
- Une bonne VM 2 middle / back sur SERVEUR et sur les volumes répliqués pve-drbd2

Je n'ai pas choisi de faire une architecture n-tiers du avec un back parce que vu ce que j'héberge cela m'a semblé un peu trop luxueux. De plus ça multiplie les install les administrations et les efforts. Par contre le garde le principe du front et du middle séparés comme ça je protège mes applications avec un front.

18.3. Installation des VM front

18.3.1. Création des VM

Rendez vous sur l'interface WEB proxmox et cliquez sur **“Créer VM”**. L'avantage du cluster c'est que

vous n'avez pas besoin d'aller sur chaque noeud pour faire l'opération. Vous pourrez créer les VM sur chaque serveur physique depuis la même interface. Il faudra simplement prendre garde de **bien choisir le bon noeud** pour chaque VM créée.

Général :

- Cliquez sur **Créer VM**
- Choisissez le noeud de votre VM (= le serveur physique)
- Choisissez l'id et le nom de la vm

OS :

- Linux 3.X/2.6 Kernel (à adapter selon votre conf mais j'ai pris plus haut la dernière ubuntu)

CD/DVD :

- Utilisé une image disque
- Choisissez votre iso sur le disque local

Disque Dur :

- Bus/Device : SATA
- Stockage : pve-drbd1 (pour les VM sur SERVEUR_PHYSIQUE_1 et pve-drbd2 pour les VM sur SERVEUR_PHYSIQUE_2)
- Taille : 15Go (c'est une VM front, on y aura presque rien et ce sera bien large pour un reverse proxy avec un cache)

CPU :

- 1 socket, 2 cores (à adapter en fonction de votre serveur physique mais il ne faut pas s'emballer et faire plus que ce qu'on a physiquement et réparti sur toutes les VM)
- Type KVM 64

Mémoire :

- 2048Mio (pour une VM front 2Gio est bien assez pour mon usage et idem que CPU ne pas dépasser ce que vous avez en physique distribué sur toutes vos VM)

Réseau :

- vmbr0 (pour l'accès au net) avec une MAC que vous aurez récupéré d'une IP failover avec une mac pour KVM online.net
- On ajoutera vmbr1 plus tard

On valide et la VM est créée. On renouvèle pour la seconde VM front sur le second serveur physique mais en adaptant la configuration. Les VM sont créées mais pas encore démarrées.

18.3.3. Ajout carte réseau interne des VM

Par défaut, on a configuré une carte réseau sur **vmbr0** qui permet d'accéder au réseau public.

Or nous avons configuré sur nos serveurs physiques un second bridge **vmbr1** pour le réseau interne

des VM.

Nous allons donc aller sur la VM fraichement créée puis onglet **matériel**, **Ajouter** et enfin **Carte réseau**. Nous ajoutons une nouvelle carte réseau configurée sur vmbr1.

18.3.4. Installation des VM

Avant toute choses, n'oubliez pas avant de régler dans les options de la VM la disposition du **clavier à français** sinon bonjour le qwerty dans la console VNC !

Selon ce que vous souhaitez, n'oubliez pas, toujours dans l'ihm web partie **Options**, de configurer votre VM pour qu'elle démarre automatiquement au démarrage du serveur. Vous pouvez également configurer (conseillé) l'ordre de démarrage avec un numéro d'ordre et un temps de délai. Pour plus d'informations vous pouvez consulter cette page

Vos VM ne sont pas encore dispo sur le réseau (et pas installées d'ailleurs). Mais proxmox fournit un accès console direct à vos VM. Il suffit d'aller sur l'IHM web de choisir votre VM et cliquer sur **console**. Attention il vous faudra un plugin java qui tourne.

Cliquez sur **démarrer** pour démarrez la VM et faites l'install comme si c'était chez vous.

Lors de l'installation, **pour la partie réseau**, choisissez l'option **configurer plus tard**. On le fera manuellement dans la partie suivante.

Enfin une fois que c'est fini, on oublie pas de retourner sur l'IHM web proxmox, choisir sa VM, **cliquer sur le CDROM puis éditer et enfin mettre sur none**. Plus besoin de l'iso d'install ubuntu puisqu'on vient de la terminer.

18.3.5. Configuration réseau des VM

On le fera manuellement notamment en nous basant sur la documentation de online.net dans la partie proxmox.

On commence par désactiver ipv6 qu'on utilise pas comme indiqué dans la doc unbuntu :

```
nano /etc/sysctl.conf
```

```
# Disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Et enfin :

```
echo "options ipv6 disable=1" > /etc/modprobe.d/disable-ipv6.conf
```

Ensuite on configure le réseau :

```
sudo nano /etc/network/interfaces
```

Avec de quoi atteindre le réseau public fourni par vmbr0 et le réseau interne fourni par vmbr1 :

```
# The loopback network interface
auto lo
iface lo inet loopback

# To access vmbr0 for public network
auto eth0
iface eth0 inet static
    address IP_FAILOVER_VM_1
    netmask 255.255.255.255
    broadcast IP_FAILOVER_VM_1
    dns-nameservers 62.210.16.6 62.210.16.7
    post-up route add IP_SERVEUR_PHYSIQUE_1_SANS_LA_FIN.1 dev eth0
    post-up route add default gw IP_SERVEUR_PHYSIQUE_1_SANS_LA_FIN.1
    post-down route del IP_SERVEUR_PHYSIQUE_1_SANS_LA_FIN.1 dev eth0
    post-down route del default gw IP_SERVEUR_PHYSIQUE_1_SANS_LA_FIN.1

# To access vmbr1 for internal network
auto eth1
iface eth1 inet static
    address 192.168.100.11
    netmask 255.255.255.0
```

Pour eth0, je laisse les DNS qui sont ceux de online.net (les nouveaux, vu sur cette page).

La gateway du réseau publique correspond à l'ip physique du dédié mais qu'on termine par .1.

La configuration est a adapter en fonction de vos spécificités.

On redémarre les VM et on vérifie que les accès réseau fonctionnent sur les interfaces publiques comme internes.

18.3.6. Préparation des VM

Sur chaque VM, afin d'éviter de passer par l'accès console via l'IHM proxmox, on install openssh et ses petits amis :

<sudo bash> sudo apt-get install ssh </code>

Et on oublie pas un bon passage par la case mise à jour :

```
sudo apt-get install update
sudo apt-get install upgrade
```

Enfin, le plus important **ON SECURISE ET ON BACKUP ses serveurs VM**. Je vous renvoie aux parties similaires traitées plutôt dans ce guide. On peut les reprendre en grande partie à part quelques adaptations que nous verrons plus tard.

Même combat également pour la configuration du **serveur mail, des DNS** etc etc. Voir les parties précédentes du guide.

18.3.6. Firewall iptables et NAT / routage sur les VM

Pour rappel nous sommes sur des VM front. Elles vont donc notamment assurer le NAT de tout le traffic réseau des VM middle. Il convient donc de construire une configuration iptables qui va bien.

Pour activer le mode routeur sur notre serveur middle, on doit activer l'ip forwarding.

```
sudo nano /etc/sysctl.conf
```

Et activer cette ligne

```
net.ipv4.ip_forward=1
```

Enfin on prépare la configuration firewall / iptables. Elle ressemble beaucoup à ce qu'on a vu précédemment mais version plus limitée et avec une commande supplémentaire pour l'ip forwarding.

```
sudo nano /etc/init.d/iptables
```

```
#!/bin/bash

echo Setting iptables / firewall rules...
#
# Dedibox iptables / firewall configuration
#
# http://wiki.debian.org/LSBInitScripts/DependencyBasedBoot
# http://wiki.debian.org/LSBInitScripts
#
http://documentation.online.net/fr/serveur-dedie/tutoriel/iptables-netfilter
-configuration-firewall
# http://www.alsacreations.com/tuto/lire/622-Securite-firewall-iptables.html
#
# Modifications alban.montaigu
#

###### Start Initialisation ######

# Empty current tables
iptables -t filter -F
iptables -t filter -X
echo - Empty current tables : [OK]

# No input connection
iptables -t filter -P INPUT DROP
iptables -t filter -P FORWARD DROP
echo - No input connection : [OK]
```

```
# No output connection
iptables -t filter -P OUTPUT DROP
echo - No output connection : [OK]

# Authorize SSH
iptables -t filter -A INPUT -p tcp --dport 2222 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 2222 -j ACCEPT
echo - Authorize SSH : [OK]

# Authorize interfaces on internal network (enable multicast)
iptables -A INPUT -i eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A OUTPUT -o eth1 -j ACCEPT
echo - Authorize eth1 [OK]

#  Do not break established connections
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
echo - Do not break established connections : [OK]

###### End Inialisation ######

##### Start Rules  ######

# Authorize DNS, FTP, HTTP, NTP
iptables -t filter -A OUTPUT -p tcp --dport 21 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 53 -j ACCEPT
iptables -t filter -A OUTPUT -p udp --dport 53 -j ACCEPT
iptables -t filter -A OUTPUT -p udp --dport 123 -j ACCEPT
echo - Authorize DNS, FTP, HTTP, NTP : [OK]

# Authorize loopback
iptables -t filter -A INPUT -i lo -j ACCEPT
iptables -t filter -A OUTPUT -o lo -j ACCEPT
echo - Authorize loopback : [OK]

# Authorize ping
iptables -t filter -A INPUT -p icmp -j ACCEPT
iptables -t filter -A OUTPUT -p icmp -j ACCEPT
echo - Authorize ping : [OK]

# HTTP
iptables -t filter -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 8443 -j ACCEPT
echo - Authorize HTTP : [OK]

# Monit
iptables -t filter -A INPUT -p tcp --dport 8080 -j ACCEPT
```

```
iptables -t filter -A OUTPUT -p tcp --dport 8080 -j ACCEPT
echo - Authorize monit : [OK]

# FTP (passive dedibackup-dc2.online.net and active)
modprobe ip_conntrack_ftp
iptables -t filter -A INPUT -p tcp --dport 20 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 21 -j ACCEPT
iptables -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --sport 1024:65535 --dport 1024:65535 -m
state --state NEW -d dedibackup-dc2.online.net -j ACCEPT
echo - Authorize FTP : [OK]

# Mail
iptables -t filter -A INPUT -p tcp --dport 25 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 110 -j ACCEPT
iptables -t filter -A INPUT -p tcp --dport 143 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 25 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 110 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp --dport 143 -j ACCEPT
echo - Authorize mail : [OK]

###### Start NAT ######

# NAT to run as a gateway
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
echo - Activate NAT : [OK]

###### End NAT ######

# LOGGING if necessary (must be at the end)
# From http://www.thegeekstuff.com/2012/08/iptables-log-packets/
#iptables -N LOGGING
#iptables -A INPUT -j LOGGING
#iptables -A OUTPUT -j LOGGING
#iptables -A LOGGING -j LOG --log-prefix "[IPTABLES-DROP]" --log-level debug
#iptables -A LOGGING -j DROP
#echo - Log enabled

###### End rules ######

echo Iptables / firewall update successfull !
```

On donne les droits d'execution à ce script :

```
sudo chmod +x /etc/init.d/iptables
```

Et seulement quand on est pret et qu'on a testé on le configure pour s'executer au démarrage :

```
sudo update-rc.d iptables defaults
```

Et on reboot.

18.4. Installation des VM Middle

18.4.1. Création, configuration, installation préparation des VM

Pour cette partie, je fais faire court, le principe reste le même que celui des VM front. Vous pouvez donc vous y référer.

Evidemment ce sont des VM middle, elles doivent être donc plus caustaud. C'est bien sur et encore une fois à adapter en fonction de vos besoins.

Pour ma part, pour un site à faible fréquentation, je prends :

- 9Go de mémoire
- 32Go de HDD

A noter que la configuration **réseau** sera différente. Nous prendrons qu'une seule interface qui sera connectée sur **vmbr1**. En effet, nos VM middle passeront systématiquement par les VM front pour tous les accès réseau.

18.4.2. Configuration réseau des VM

Ici nous avons une configuration vraiment simplifiée des VM middle. Il faut juste qu'elles puissent joindre les fronts via vmbr1.

On oublie pas de désactiver ipv6 aussi comme pour les fronts.

```
sudo nano /etc/network/interfaces
```

Avec de quoi atteindre le réseau public fourni par vmbr0 et le réseau interne fourni par vmbr1 :

```
# The loopback network interface
auto lo
iface lo inet loopback


# To access vmbr1 for internal network
auto eth0
iface eth0 inet static
    address 192.168.100.21
    netmask 255.255.255.0
    gateway 192.168.100.11
    dns-nameservers 62.210.16.6 62.210.16.7
```

Note : évidemment sur la 2ème vm middle on change la gateway pour utilisa la seconde vm front et non pas mettre tout sur la première. Cela fonctionnerait puisqu'on a construit un réseau qui permet à toutes les VM de parler ensemble.

Et on reboot (version j'ai la flemme) puis on test.

On oublie pas ensuite une mise à jour du système puisqu'on a le réseau public enfin acessible.

19. Conseils de conf en vrac pour tous les serveurs

C'est un peu de la répétition mais pour tous vos serveur il peut être utile de configurer les clients ssh pour utiliser votre port personnalisé. Sinon ce sera des commandes ou il faudra préciser toujours le port à la main.

Pour cela on modifie le fichier

```
sudo nano /etc/ssh/ssh_config
```

Et on active la ligne (à adapter en fonction de votre configuration) :

```
Port 2222
```

Egalement pour simplifier les commandes sur chaque host, je vous conseille de mettre leurs nom avec leur ip sur le réseau interne dans votre fichier hosts.

```
sudo nano /etc/hosts
```

A adapter par contre mais c'est pour l'exemple :

```
# Localhost
127.0.0.1 localhost
127.0.1.1 NOM_MACHINE_LOCALE


# Other servers on internal network
IP1 NOM_MACHINE_2
IP2 NOM_MACHINE_3
```

20. Installation / Configuration de

l'infrastructure applicative

20.1. Architecture

Pour la partie applicative sur nos différentes VM (je rappelle 2 front et deux middle / back) je choisis les éléments suivants :

- Sur les fronts, nginx configuré en proxy inverse avec du cache
- Sur les middles, nginx en serveur web
- PHP FPM pour la partie applicative
- MySQL en fidèle SGBD

Nginx en proxy inverse, car j'ai testé varnish mais il est proprio et surtout ne gère pas le SSL ce qui m'a beaucoup embêté (obligé de remettre un nginx devant pour le SSL !!). De plus il semble qu'il n'ait pas du tout à rougir de ses performances d'après ce comparatif.

Nginx en serveur web, car il est plus léger que le vénérable apache.

PHP FPM car je veux que PHP tourne en service standalone et maintenant que ça existe faut pas se priver ! Je n'ai jamais aimé le fait que PHP soit intégré aux serveurs webs.

MySQL rien à dire dessus c'est souvent ce serveur qui est utilisé pour les applications web que j'installe donc bon...

Evidemment cette architecture concerne mes besoins et ma réflexion, elle sera a adapter chez vous je suppose.

20.2. Installation / Configuration NGINX en proxy cache sur toutes les VM Front

Pour cette architecture, j'ai choisi de faire fonctionner nginx en reverse proxy et en serveur web. Il est donc temps de l'installer sur toutes les VM. Comme les packets de ubuntu sont plus que dépassés, on choisit d'installer nginx via les dépots officiels nginx.

Avant tout on récupère la clé gpg :

```
cd
wget http://nginx.org/keys/nginx_signing.key
```

Et on ajoute cette clé dans apt :

```
sudo apt-key add nginx_signing.key
```

Et enfin on ajoute le repo officiel nginx :

```
sudo nano /etc/apt/sources.list.d/nginx.list
```

Pour y ajouter (a adapter en fonction de votre distribution) :

```
# From http://wiki.nginx.org/Install
deb http://nginx.org/packages/ubuntu/ trusty nginx
deb-src http://nginx.org/packages/ubuntu/ trusty nginx
```

Et on met a jour les dépots :

```
sudo apt-get update
```

Et on installe le paquet (les autres noms de paquets viennent des dépots d'ubuntu et comportent des versions plus vieilles) :

```
sudo apt-get install nginx
```

Comme on veut configurer notre **NGINX en proxy reverse avec du cache** nous retouchons à la configuration. Je me suis basé en partie sur la doc nginx et la doc ubuntu mais attention sur la doc ubuntu : mettre la conf proxy dans un fichier conf à part charge la conf trop tard dans l'ordre des conf et provoque une erreur. La solution est donc de mettre cette partie dans le nginx.conf.

```
sudo nano /etc/nginx/nginx.conf
```

Qui devrait ressembler à quelque chose comme ça :

```
user  www-data www-data;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;


    # Log configuration
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
'
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;
```

```
    # Misc configuration
    sendfile        on;
    keepalive_timeout  65;
    gzip  on;


    # Proxy and cache configuration
    proxy_redirect          off;
    proxy_set_header        Host            $host;
    proxy_set_header        X-Real-IP       $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_hide_header       X-Powered-By;
    proxy_intercept_errors on;
    proxy_buffering on;


    proxy_cache_key "$scheme://$host$request_uri";
    proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=cache:10m
inactive=7d max_size=700m;


    # Include configuration files for differents sites
    include /etc/nginx/conf.d/*.conf;
}
```

Et enfin nous configurons nos sites (ici je fais celui par defaut mais le travail est le même pour chaque site et à adapter) :

```
sudo nano /etc/nginx/conf.d/default.conf
```

Qui va ressembler à quelque chose comme ceci :

```
server {
    listen       80;
    server_name  localhost;


    # Proxy to the backend
    location / {
        proxy_pass            http://192.168.100.21;
        proxy_cache cache;
        proxy_cache_valid 12h;
        expires 12h;
        proxy_cache_use_stale error timeout invalid_header updating;
    }

}
```

Evidemment ce sont des **conf minimales à perfectionner et a adapter** en fonction de ce que vous avez sur vos backends (i.e. sur les middles).

20.3. Installation / Configuration PHP 5 FPM sur les VM middle / back

Comme vu plus haut, on part sur une installation de PHP5 FPM sur les middle pour faire tourner nos applis dans un service PHP indépendant. Pour cela on fait simple (enfin ça c'est pour mes besoins, c'est à adapter pour vous) :

```
sudo apt-get install php5-fpm
sudo apt-get install php5-mysql
sudo apt-get install php5-cgi
sudo apt-get install php5-cli
sudo apt-get install php5-curl
sudo apt-get install php5-gd
sudo apt-get install php5-mcrypt
sudo apt-get install phpmyadmin
```

Voilà les packages installés chez moi :

```
sudo dpkg --get-selections | grep php
```

```
php-gettext                                     install
php5-cgi                                        install
php5-cli                                        install
php5-common                                     install
php5-curl                                       install
php5-fpm                                        install
php5-gd                                         install
php5-json                                       install
php5-mcrypt                                     install
php5-mysql                                      install
php5-readline                                   install
phpmyadmin                                      install
```

Un peu de configuration de PHP FPM

```
sudo nano /etc/php5/fpm/php.ini
```

Voici les choses à modifier principalement pour la sécurité :

```
open_basedir = /var/www:/tmp:/usr/share/phpmyadmin

disable_functions =
pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,exec,system,popen,proc_open,shell_exec
```

A noter que par défaut php fpm écoutera sur un socket

```
/var/run/php5-fpm.sock
```

Mais qu'on peut le faire écouter sur un port particulier. Ici on choisit le sock puisque notre serveur web est sur le middle aussi donc pas besoin de passer par le réseau.

Le détail de cette partie de la conf se situe dans le fichier

```
sudo nano /etc/php5/fpm/pool.d/www.conf
```

Enfin, avec la dernière version de PHP, APC semble ne plus fonctionner... C'est remplacé par un module built in comme expliqué sur cette page et cette page

```
sudo nano /etc/php5/fpm/php.ini
```

Et modifier ainsi

```
opcache.enable=1
opcache.memory_consumption=128
opcache.max_accelerated_files=4000
opcache.revalidate_freq=60
```

Et bien sur redémarrer le service :

```
sudo service php5-fpm restart
```

20.4. Installation / Configuration NGINX sur les VM middle / back

Ici il s'agit d'installer NGINX sur les VM middle. La procédure est la même que pour les front. Par contre concernant la configuration on n'est plus sur de serveur proxy mais du serveur web classique.

Il suffira de le faire communiquer avec PHP FPM via le sock mis en place précédemment.

Par exemple :

```
sudo nano /etc/nginx/conf.d/default.conf
```

Donne :

```
server {
    listen       80;
    server_name  localhost;

    #access_log  /var/log/nginx/log/localhost.access.log  main;

    root   /var/www/default;
```

```
    index  index.html index.htm index.php;

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # pass the PHP scripts to FastCGI server
    location ~ ^(.+\.php)(/.*)?$ {
        fastcgi_pass   unix:/var/run/php5-fpm.sock;
        include        fastcgi_params;
    }
}
```

20.5. Installation / Configuration MySQL sur les VM middle / back

Installation MySQL, rien de particulier à ajouter.

```
sudo apt-get install mysql-server
```

Et une petite correction à faire en se basant sur cet article si vous voyez ce message :

```
[Warning] Using unique option prefix key_buffer instead of key_buffer_size
is deprecated and will be removed in a future release. Please use the full
name instead.
```

On corrige donc :

```
sudo nano /etc/mysql/my.cnf
```

Et on modifie ceci :

```
    key_buffer = 16M
    myisam-recover = BACKUP
```

Comme ceci (et c'est le premier buffer size si vous en avez plusieurs) :

```
    key_buffer_size = 16M
    myisam-recover-options = BACKUP
```

20.6. Installation / Configuration des mails sur toutes les VM

Sur ubuntu de base il faut passer par une petite phase d'installation.

```
sudo apt-get install mailutils
```

Puis pour la configuration voir comme évoqué plus haut.

```
sudo dpkg-reconfigure postfix
```

Spécificité pour les middles, comme leurs mails passeront forcément par le front, **le nom de courrier doit correspondre au nom de la vm front**. Exemple monserveur1.domaine.net si la vm front s'appelle monserveur1 et que vous lui avez bien affecté un dns monserveur1.domaine.net comme vu dans les parties précédentes.

20.7. Installation / Configuration rkhunter, monit, logwatch backup-manager

Pas grand chose à ajouter, simplement il ne faut pas oublier. Vous pouvez vous référer aux précédentes parties où j'ai déjà abordé ces sujets.

From:
https://wiki.montaigu.io/ - **Alban's Wiki**

Permanent link:
https://wiki.montaigu.io/doku.php?id=guide:installation_serveurs_2014&rev=1405420016

Last update: **2021/04/18 20:24**