Redondance et continuité de service

Deux serveurs partagent une ressource de stockage commune grâce à du RAID 1 en réseau (DRBD). Un logiciel (Heartbeat) installé sur chaque noeud vérifie que l'autre noeud est vivant. Un autre logiciel (Monit) interroge les services et les redémarre s'ils ne répondent pas.
Si un des services ne peut pas être redémarré, malgré plusieurs essais, le noeud primaire transfert tous les services sur le noeud secondaire. Les services seront de nouveau interrogés et redémarrés, s'il le faut. Par contre, les services ne seront pas transférés vers le noeud primaire s'ils ne peuvent être redémarrés. Nous fonctionnerons en mode dégradé. Maintenant, si le noeud secondaire où se trouve les services meurt, le noeud primaire récupérera tous les services et les surveillera à nouveau ...

Ces logiciels seront utilisés dans ce tutoriel :

DRBD (Distributed Replicated Block Device) est un module du kernel Linux qui offre un système de stockage distribué.
Il réplique un périphérique de type bloc à travers un réseau, comme un RAID 1 entre des disques durs.

Avantage : Synchronisation en temps réel.
Inconvénient : Vitesse de transfert des données limitée par la connexion réseau dans notre cas.

Note : DRBD est inutile si vous utilisez un système de stockage SAN.

DRBDlinks permet de faire des liens (symboliques ou montages) depuis des données partagées vers une destination locale. Exemples : /home, fichiers de configuration d'un service.

Avantage : Les liens sont réalisés ou enlevés avec une simple commande.
Inconvénient : Si le fichier de destination n'existe pas, le lien sera réalisé mais une erreur sera retournée quand il sera enlevé.

Heartbeat existe en deux version :
- les séries 1.x, fonctionnent avec deux noeuds seulement.
  Les noeuds se surveillent mutuellement et chaque noeud prend des décisions selon l'état de l'autre noeud.
  Il faut ajouter un logiciel (mon, monit, nagios, ...) pour surveiller les services et lancer des actions si des services ne répondent plus.

- les séries 2.x, fonctionnent avec plusieurs noeuds.
  Les noeuds se surveillent mutuellement comme pour les séries 1.x.
  Une fonction de surveillance des services est intégrée.

Avantage : Permet de connaître l'état de chaque noeuds en cas de kernel panic ou de coupure électrique et de réaliser ou non des actions.
Inconvénient : Si le nombre de liaisons de surveillance est insuffisant, les décisions prises par Heartbeat sont catastrophiques. Le script de gestion des ressources DRBD fournit dans le package Heartbeat, n'est pas très fiable.

Monit permet d'interroger des services ou de vérifier des ressources, permissions, ... et de lancer des actions.

Avantage : Configuration de la surveillance des services très simple. Faible consommation de mémoire par rapport à d'autres logiciels de supervision.
Inconvénient : Limitation de l'exécution de scripts pour certains contrôles (socket).

Bonding (channel bonding) est un module du kernel Linux qui permet de faire une agrégation de plusieurs interfaces réseau physiques vers une seule interface logique. Le but étant de sécuriser la liaison réseau et d'augmenter la bande passante selon le mode choisi.

Avantage : Selon le type de mode choisi (sept modes sont disponibles) on peut faire de la répartition de charge tout en conservant une liaison sécurisée.
Inconvénient : Vos cartes réseau doivent être compatibles ethtools et miitools. Vos switch doivent supporter la configuration du mode "port truking" sur les ports utilisés ainsi que la norme IEEE 802.3ad.

Voici les étapes que nous suivrons :

- Configuration du channel bonding
- Synchronisation de l'heure sur les deux noeuds par NTP
- Configuration de DRBD sur les deux noeuds
- Configuration de NFS sur le noeud primaire
- Configuration de l'écoute des services sur une adresse IP virtuelle
- Installation des services redondants sur le noeud secondaire (sans les configurer)
- Copie des répertoires et fichiers de configuration des services vers la ressource commune des deux noeuds
- Configuration de DRBDlinks sur les deux noeuds
- Configuration de Heartbeat sur les deux noeuds
- Configuration de Monit sur les deux noeuds

Configuration des machines nas1 et nas2

Configuration du channel bonding

Installation des packages

Sur nas1.alex.dmz et nas2.alex.dmz :
ifenslave-2.6 ethtool

Les machines auront trois cartes réseau :

- eth0 => Connecté sur un switch.
- eth1 et eth2 => Connectés entre chaque serveur par câbles croisés. Utilisés pour le channel bonding.

ATTENTION : Les câbles croisés GigaBit Ethernet doivent avoir la paire 4 et 5 sur la 7 et 8

et inversement.

Pour vérifier que la connexion par câbles croisés est bien en GigaBit :

```
nas1:~# ethtool eth1
Settings for eth1:
    Supported ports: [ TP ]
    Supported link modes: 10baseT/Half 10baseT/Full
                          100baseT/Half 100baseT/Full
                          1000baseT/Full
    Supports auto-negotiation: Yes
    Advertised link modes: 10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Advertised auto-negotiation: Yes
    Speed: 1000Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 0
    Transceiver: internal
    Auto-negotiation: on
    Supports Wake-on: umbg
    Wake-on: g
    Current message level: 0x00000007 (7)
    Link detected: yes

nas1:~# ethtool eth2 | grep Speed
    Speed: 1000Mb/s
```

Il est possible que l'utilitaire mii-tool ne connaisse pas votre carte réseau. Pareil pour ethtool. Employez les deux utilitaires pour obtenir les bonnes informations.

Créez un fichier /etc/modprobe.d/bonding sur les deux serveurs, qui permettra de charger le module bonding :

```
# On associe l'interface bond0 au module bonding.
# On passe ces options au module :
# miimon => Fréquence de surveillance des interfaces par Mii ou ethtool.
# La valeur conseillée est 100.
# mode => Le type de mode de fonctionnement associé au bonding.
alias bond0 bonding
options bonding miimon=100 mode=balance-rr
```

Modifiez le fichier /etc/network/interfaces sur **nas1** :

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

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 172.16.0.17
```

```
netmask 255.255.255.0
up route add -net 172.17.0.0/16 gw 172.16.0.254

# Bonding interface
auto eth1 eth2 bond0
allow-hotplug eth1 eth2 bond0
iface bond0 inet static
address 192.168.0.1
netmask 255.255.255.252
up ifenslave bond0 eth1 eth2
down ifenslave -d bond0 eth1 eth2
```

Modifiez le fichier /etc/network/interfaces sur **nas2** :

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

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 172.16.0.18
netmask 255.255.255.0
up route add -net 172.17.0.0/16 gw 172.16.0.254

# Bonding interface
auto eth1 eth2 bond0
allow-hotplug eth1 eth2 bond0
iface bond0 inet static
address 192.168.0.2
netmask 255.255.255.252
up ifenslave bond0 eth1 eth2
down ifenslave -d bond0 eth1 eth2
```

Reboutez les machines après ces modifications.

La commande "ifconfig" vous affichera les informations sur vos interfaces.

Lancez des pings depuis un des serveurs pour vérifier qu'il n'y a pas de pertes de paquets :

```
nas1:~# ping -c 5 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.023 ms
...
64 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=0.022 ms

--- 192.168.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.018/0.020/0.023/0.006 ms
```

Synchronisation de l'heure sur les deux noeuds par NTP

Installation des packages

Sur nas1.alex.dmz et nas2.alex.dmz :
ntpdate

Opérations à réaliser sur les deux noeuds.

Modifiez le fichier /etc/default/ntpdate :

```
# The settings in this file are used by the program ntpdate-debian, but
# not by the upstream program ntpdate.

# Set to "yes" to take the server list from /etc/ntp.conf, from package
# ntp, so you only have to keep it in one place.
NTPDATE_USE_NTP_CONF=no

# List of NTP servers to use  (Separate multiple servers with spaces.)
# Not used if NTPDATE_USE_NTP_CONF is yes.
#NTPSERVERS="0.debian.pool.ntp.org 1.debian.pool.ntp.org
2.debian.pool.ntp.org 3.debian.pool.ntp.org"
NTPSERVERS="ntp.alex.dmz"

# Additional options to pass to ntpdate
NTPOPTIONS="-u"
```

Créez le fichier /etc/cron.hourly/ntpdate-sync-cron et rendez-le exécutable :

```
#!/bin/bash

/usr/sbin/ntpdate-debian > /dev/null 2>&1
```

Modifiez le fichier /etc/hosts :

Sur nas1 :

```
127.0.0.1       localhost
172.16.0.17     nas1.alex.dmz
192.168.0.1     nas1
```

Sur nas2 :

```
127.0.0.1       localhost
172.16.0.18     nas2.alex.dmz
192.168.0.2     nas2
```

Configuration de DRBD sur les deux noeuds

Les deux noeuds doivent avoir le même kernel, donc la même architecture, pour

simplifier la gestion.

Installation des packages

Sur nas1.alex.dmz :
drbd0.7-utils

Sur nas2.alex.dmz :
linux-headers-2.6-686* build-essential kernel-package ncurses-dev dpkg-dev drbd0.7-module-source drbd0.7-utils

Note* : Selon votre architecture.

Compilation du module DRBD pour votre kernel (2.6.18-5-686 dans cet exemple)

```
nas2:~# cd /usr/src
nas2:/usr/src# ls
drbd0.7.tar.gz linux-headers-2.6.18-5 linux-headers-2.6.18-5-686 linux-kbuild-2.6.18

nas2:/usr/src# tar xzf drbd0.7.tar.gz

nas2:/usr/src# cd linux-headers-$(uname -r)
nas2:/usr/src/linux-headers-2.6.18-5-686# module-assistant
```

Une fois module-assistant lancé :

choisissez **SELECT**, puis sélectionnez le module drbd0.7 (drbd0.7-module) et validez.
choisissez **BUILD** et répondez par <Oui> au deux questions posées.
choisissez **LIST** pour afficher les paquets binaires installés. Vous devez voir les informations suivantes :

```
drbd0.7-module-source (source) installé (V : 0.7.21-4) :
-- Paquet(s) binaires pour le(s) noyau(x) :
...
```

Sortez de l'assistant de module en faisant <Annuler> puis **EXIT**.

Chargez le module drbd fraîchement compilé :

```
nas2:/usr/src/linux-headers-2.6.18-5-686# modprobe drbd
nas2:/usr/src/linux-headers-2.6.18-5-686# lsmod | grep drbd
```

Dans cet exemple, un package *drbd0.7-module-2.6.18-5-686_0.7.21-4+2.6.18.dfsg.1-13etch4_i386.deb* a était créé sur mesure pour votre kernel par l'assistant de module.

```
nas2:/usr/src# ls
drbd0.7-module-2.6.18-5-686_0.7.21-4+2.6.18.dfsg.1-13etch4_i386.deb
drbd0.7.tar.gz linux-headers-2.6.18-5 linux-headers-2.6.18-5-686 linux-
kbuild-2.6.18 modules
```

Copiez ce package sur le noeud primaire nas1 est installez-le (dpkg -i ...)
Chargez le module drbd :

```
nas1:~# modprobe drbd
nas1:~# lsmod | grep drbd
```

Opérations de maintenance pour DRBD

Si vous devez réaliser une mise à jour de votre kernel sur des machines en production, il faudra recompiler le module DRBD pour le nouveau kernel.

Voici la procédure :

- Arrêtez Monit sur le noeud primaire (*/etc/init.d/monit stop*)
- Mettez Heartbeat en attente sur le noeud secondaire (*/usr/lib/heartbeat/hb_standby local*)
- Arrêtez Heartbeat sur le noeud secondaire (*/etc/init.d/heartbeat stop*)
- Arrêtez DRBD sur le noeud secondaire (*/etc/init.d/drbd stop*)
- Désinstallez le package drbd0.7-module-2.6.18-*-* (*apt-get remove --purge drbd0.7-module-2.6.18-5-686* dans mon cas)
- Mettez à jour le kernel sur le noeud secondaire
- Modifiez le script /etc/init.d/heartbeat pour que Heartbeat ne démarre pas au boot de la machine :

```
#!/bin/sh
exit 0
...
```

- Reboutez le noeud secondaire sur le nouveau kernel
- Compilez le module DRBD avec l'assistant de module
- Commentez dans le script /etc/init.d/heartbeat la ligne ajoutée précédemment :

```
#!/bin/sh
#exit 0
...
```

- Démarrez DRBD sur le noeud secondaire
- Démarrez Heartbeat sur le noeud secondaire
- Migrez les ressources actuellement sur le noeud primaire, vers le noeud secondaire depuis le noeud primaire (*/usr/lib/heartbeat/hb_standby all*)
- Arrêtez Monit sur le noeud secondaire
- Arrêtez DRBD sur le noeud primaire
- Désinstallez le package drbd0.7-module-2.6.18-*-*

- Mettez à jour le kernel sur le noeud primaire
- Modifiez le script /etc/init.d/heartbeat pour que Heartbeat ne démarre pas au boot de la machine :

```
#!/bin/sh
exit 0
...
```

- Reboutez le noeud primaire sur le nouveau kernel
- Copiez le nouveau package drbd0.7-module-2.6.18... depuis le noeud secondaire vers le noeud primaire
- Installez le package drbd0.7-module-2.6.18...
- Commentez dans le script /etc/init.d/heartbeat la ligne ajoutée précédemment :

```
#!/bin/sh
#exit 0
...
```

- Démarrez DRBD sur le noeud primaire
- Démarrez Heartbeat sur le noeud primaire
- Démarrez Monit sur le noeud secondaire

Création de la partition qui sera utilisée par DRBD

Il est préférable de sécuriser vos données sur cette partition à l'aide d'un RAID matériel plutôt que logiciel. Ce tutoriel décrira l'utilisation de DRDB avec un RAID matériel uniquement.

Dans cet exemple, /dev/sdb est constitué d'un ensemble de disques en RAID 5.
Je crée une seule partition qui sera nommée /dev/sdb1.

```
nas1:~# fdisk /dev/sdb

Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel Building a new DOS disklabel. Changes will remain in memory
only, until you decide to write them. After that, of course, the previous
content won't be recoverable.

The number of cylinders for this disk is set to 1044.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs (e.g., DOS FDISK, OS/
2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will be corrected by
w(rite)

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1044, default 1): (réponse par défaut)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1044, default 1044):
Using default value 1044

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1044     8385898+  83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
nas1:~#
```

Réalisez les mêmes commandes sur le noeud secondaire nas2.

Configuration de DRBD

Modifiez le fichier /etc/drbd.conf :

```
# Paramètres pour la ressource nommé nas.
resource nas {

# Protocole de transfert à utiliser :
#
# protocole A = asynchrone
# protocole B = quasi-synchrone
# protocole C = synchrone
protocol C;

# what should be done in case the cluster starts up in
# degraded mode, but knows it has inconsistent data.
incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep 60 ; halt
-f";

startup {
# Wait for connection timeout.
```

```
# The init script blocks the boot process until the resources
# are connected. This is so when the cluster manager starts later,
# it does not see a resource with internal split-brain.
# In case you want to limit the wait time, do it here.
# Default is 0, which means unlimited. Unit is seconds.
#
wfc-timeout 90;

# Wait for connection timeout if this node was a degraded cluster.
# In case a degraded cluster (= cluster with only one node left)
# is rebooted, this timeout value is used.
#
degr-wfc-timeout 120;    # 2 minutes.
}

disk {
# if the lower level device reports io-error you have the choice of
# "pass_on"  -> Report the io-error to the upper layers.
#               Primary   -> report it to the mounted file system.
#               Secondary -> ignore it.
# "panic"    -> The node leaves the cluster by doing a kernel panic.
# "detach"   -> The node drops its backing storage device, and
#               continues in disk less mode.
#
on-io-error detach;

# In case you only want to use a fraction of the available space
# you might use the "size" option here.
#
# size 10G;
}

net {
# this is the size of the tcp socket send buffer
# increase it _carefully_ if you want to use protocol A over a
# high latency network with reasonable write throughput.
# defaults to 2*65535; you might try even 1M, but if your kernel or
# network driver chokes on that, you have been warned.
# sndbuf-size 512k;

# timeout       60;    #  6 seconds  (unit = 0.1 seconds)
# connect-int   10;    # 10 seconds  (unit = 1 second)
# ping-int      10;    # 10 seconds  (unit = 1 second)

# Maximal number of requests (4K) to be allocated by DRBD.
# The minimum is hardcoded to 32 (=128 kByte).
# For high performance installations it might help if you
# increase that number. These buffers are used to hold
# datablocks while they are written to disk.
#
# max-buffers     2048;

# When the number of outstanding requests on a standby (secondary)
# node exceeds unplug-watermark, we start to kick the backing device
# to start its request processing. This is an advanced tuning
# parameter to get more performance out of capable storage controlers.
# Some controlers like to be kicked often, other controlers
# deliver better performance when they are kicked less frequently.
```

```
# Set it to the value of max-buffers to get the least possible
# number of run_task_queue_disk() / q->unplug_fn(q) calls.
#
# unplug-watermark   128;

# The highest number of data blocks between two write barriers.
# If you set this < 10 you might decrease your performance.
# max-epoch-size  2048;

# if some block send times out this many times, the peer is
# considered dead, even if it still answers ping requests.
# ko-count 4;

# if the connection to the peer is lost you have the choice of
#  "reconnect"   -> Try to reconnect (AKA WFConnection state)
#  "stand_alone" -> Do not reconnect (AKA StandAlone state)
#  "freeze_io"   -> Try to reconnect but freeze all IO until
#                   the connection is established again.
#  [ lge: oops. freeze_io is not implemented yet...       ]
#  [ at least not in drbd 0.7.x; but nobody wanted to use ]
#  [ that anyways, otherwise we had noticed earlier :-)   ]
# on-disconnect reconnect;
}

syncer {
# Limit the bandwith used by the resynchronisation process.
# default unit is kByte/sec; optional suffixes K,M,G are allowed.
#
# Even though this is a network setting, the units are based
# on _byte_ (octet for our french friends) not bit.
# We are storage guys.
#
# Note that on 100Mbit ethernet, you cannot expect more than
# 12.5 MByte total transfer rate.
# Consider using GigaBit Ethernet.
# Bande passante pour le transfert de resynchronisation.
# La connexion passe par câble croisé en GigaBit dans mon cas.
#
rate 100M;

# All devices in one group are resynchronized parallel.
# Resychronisation of groups is serialized in ascending order.
# Put DRBD resources which are on different physical disks in one group.
# Put DRBD resources on one physical disk in different groups.
#
group 1;

# Configures the size of the active set. Each extent is 4M,
# 257 Extents ~> 1GB active set size. In case your syncer
# runs @ 10MB/sec, all resync after a primary's crash will last
# 1GB / ( 10MB/sec ) ~ 102 seconds ~ One Minute and 42 Seconds.
# BTW, the hash algorithm works best if the number of al-extents
# is prime. (To test the worst case performace use a power of 2)
al-extents 257;
}

# Paramètres pour le noeud nas1.
# Nom du premier noeud (résolvable par la commande hostname).
```

```
on nas1 {
# Utilisation du périphérique /dev/drbd0.
device    /dev/drbd0;
# Notre partition en RAID sur nas1 associée au périphérique /dev/drbd0.
disk      /dev/sdb1;
# Adresse IP réelle du serveur nas1 et port de communication.
address   192.168.0.1:7788;
# Les méta-données DRBD seront stockées sur le périphérique /dev/sdb1.
meta-disk internal;

# meta-disk is either 'internal' or '/dev/ice/name [idx]'
#
# You can use a single block device to store meta-data
# of multiple DRBD's.
# E.g. use meta-disk /dev/hde6[0]; and meta-disk /dev/hde6[1];
# for two different resources. In this case the meta-disk
# would need to be at least 256 MB in size.
#
# 'internal' means, that the last 128 MB of the lower device
# are used to store the meta-data.
# You must not give an index with 'internal'.
}

# Paramètres pour le noeud nas2.
# Nom du deuxième noeud.
on nas2 {
device    /dev/drbd0;
# Notre partition en RAID sur nas2 associée au périphérique /dev/drbd0.
disk      /dev/sdb1;
# Adresse IP réelle du serveur nas2 et port de communication.
address   192.168.0.2:7788;
meta-disk internal;
}

}
```

Le fichier /etc/drbd.conf doit être identique sur les deux serveurs, copiez-le sur nas2 :

```
nas1:~# scp /etc/drbd.conf root@172.16.0.18:/etc/
```

Démarrez le service DRBD sur les deux serveurs en même temps :

```
nas1:~# /etc/init.d/drbd start
Starting DRBD resources:    [ d0 s0 n0 ].
nas1:~#
```

```
nas2:~# /etc/init.d/drbd start
Starting DRBD resources:    [ d0 s0 n0 ].
nas2:~#
```

Les périphériques DRBD n'ont jamais étaient synchronisés, ils sont dans un état "Incohérent" (Inconsistent).

```
nas1:~# cat /proc/drbd
version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@nas1, 2007-11-20 14:00:00
0: cs:Connected st:Secondary/Secondary ld:Inconsistent
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
nas1:~#
```

```
nas2:~# cat /proc/drbd
version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@nas1, 2007-11-20 14:00:00
0: cs:Connected st:Secondary/Secondary ld:Inconsistent
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
nas2:~#
```

Initialisation des périphériques DRBD depuis nas1 :

```
nas1:~# drbdadm -- --do-what-I-say primary all
```

Attendre que la synchronisation soit terminée (quelques heures selon la taille de votre partition) :

```
nas1:~# cat /proc/drbd
version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@nas1, 2007-11-20 14:00:00
0: cs:SyncSource st:Primary/Secondary ld:Consistent
    ns:1322976 nr:0 dw:0 dr:1330572 al:0 bm:80 lo:0 pe:37 ua:1899 ap:0
        [===>................] sync'ed: 16.1% (6769/8061)M
        finish: 0:02:18 speed: 49,996 (47,244) K/sec
nas1:~#
```

```
nas2:~# cat /proc/drbd
version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@nas1, 2007-11-20 14:00:00
0: cs:SyncTarget st:Secondary/Primary ld:Inconsistent
    ns:0 nr:1442568 dw:1442568 dr:0 al:0 bm:88 lo:5 pe:2024 ua:5 ap:0
        [===>................] sync'ed: 17.6% (6652/8061)M
        finish: 0:02:37 speed: 43,084 (35,184) K/sec
nas2:~#
```

Une fois la synchronisation terminée, leur état devient "Cohérent" (Consistent).
nas1 est maintenant Primaire et nas2 Secondaire.

```
nas1:~# cat /proc/drbd
version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@nas1, 2007-11-20 14:00:00
0: cs:Connected st:Primary/Secondary ld:Consistent
    ns:8254824 nr:0 dw:0 dr:8254824 al:0 bm:504 lo:0 pe:0 ua:0 ap:0
nas1:~#
```

```
nas2:~# cat /proc/drbd
version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@nas1, 2007-11-20 14:00:00
0: cs:Connected st:Secondary/Primary ld:Consistent
    ns:0 nr:8254824 dw:8254824 dr:0 al:0 bm:504 lo:0 pe:0 ua:0 ap:0
nas2:~#
```

Signification des informations DRBD
cs (connection state) :

```
Unconfigured    = Le périphérique attend pour être configuré.
StandAlone      = N'essayant pas de se connecter à l'autre noeud, les
requettes IO sont seulement passées localement.
Unconnected     = État de transition.
WFConnection    = Le périphérique attend que l'autre noeud soit
configuré.
WFReportParams  = État de transition, en attente de réception de
connexion.
Connected       = Connecté.
Timeout, BrokenPipe, NetworkFailure = État de transition quand la
connexion est perdue.
WFBitMap{S,T}   = État de transition quand la synchronisation commence.
SyncSource      = Synchronisation en cours, ce noeud possède des données
à jour.
SyncTarget      = Synchronisation en cours, ce noeud ne possède pas des
données à jour.
PausedSync{S,T} = La synchronisation de ce périphérique est mis en pause,
un périphérique prioritaire est en cours de synchronisation.
```

st (Local/Remote) :

```
Primary      = Noeud actif.
Secondary    = Noeud passif, aucun accès au périphérique ne doit être
fait.
Unconfigured = Noeud non configuré.
```

ld (local data) :

```
ns = network send
nr = network receive
dw = disk write
dr = disk read
al = activity log updates
bm = bitmap updates
lo = reference count on local device
pe = pending (waiting for ack)
ua = unack'd (still need to send ack)
ap = application requests expecting io-completion
```

Pour connaître simplement l'état de DRBD :

```
nas1:~# drbdsetup /dev/drbd0 state
```

```
Primary/Secondary
nas1:~#
```

```
nas2:~# drbdsetup /dev/drbd0 state
Secondary/Primary
nas2:~#
```

Pour changer d'état (Primaire/Secondaire) :

```
nas1:~# drbdsetup /dev/drbd0 secondary
nas1:~# drbdsetup /dev/drbd0 state
Secondary/Secondary
nas1:~#

nas1:~# drbdsetup /dev/drbd0 primary
nas1:~# drbdsetup /dev/drbd0 state
Primary/Secondary
nas1:~#
```

Création du système de fichier sur /dev/drbd0

Parmi les différents systèmes de fichier existant sous Linux, j'ai choisi Ext3 pour sa fiabilité (relative ... voir cette étude et mon bench) à la place de ReiserFS.

```
nas1:~# mkfs.ext3 -j /dev/drbd0

mke2fs 1.40-WIP (14-Nov-2006)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
1032192 i-noeuds, 2063706 blocs
103185 blocs (5.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=2113929216
63 groupes de blocs
32768 blocs par groupe, 32768 fragments par groupe
16384 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

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

Le système de fichiers sera automatiquement vérifié tous les 37 montages
ou après 180 jours, selon la première éventualité. Utiliser tune2fs -c ou
-i pour écraser la valeur.
nas1:~#
```

Pour désactiver l'intervalle de vérification automatique du système de fichier ext3 sur /dev/drbd0, lancez cette commande :

```
nas1:~# tune2fs -c 0 -i 0 /dev/drbd0
tune2fs 1.40-WIP (14-Nov-2006)
Initialisation du nombre maximal de montages à -1
Initialisation de l'intervalle de vérification à 0 secondes
nas1:~#
```

Si vous souhaitez vérifier manuellement l'intégrité du système de fichier ext3 sur le périphérique /dev/drbd0. Sur le noeud qui est dans un état primaire (Primary/Secondary) uniquement, vérifiez que le périphérique /dev/drbd0 n'est pas monté et lancez cette commande :

```
nas1:~# fsck.ext3 -p -y /dev/drbd0
```

Ne surtout pas lancer cette commande sur la partition (/dev/sdb1), associée au périphérique /dev/drbd0 ! car DRBD a inscrit des méta-données dessus.

Montage de la ressource /dev/drbd0

Créez un répertoire /export sur les deux serveurs nas1 et nas2. Il contiendra tous les fichiers de configurations des services et les données des utilisateurs partagés.

Sur nas1 testez le montage de /dev/drbd0 dans /export :

```
nas1:~# mount /dev/drbd0 /export
```

Vérification :

```
nas1:~# mount | grep drbd0
/dev/drbd0 on /export type ext3 (rw)
nas1:~#
```

Démontez /export :

```
nas1:~# umount /export
nas1:~# mount | grep drbd0
nas1:~#
```

Configuration de NFS sur le noeud primaire

Installation des packages

Sur nas1.alex.dmz et nas2.alex.dmz :
nfs-kernel-server acl

Je souhaite utiliser les droits étendus (acl) et les options noatime, nodiratime (voir man mount), je monte donc /dev/drbd0 avec l'option acl,noatime,nodiratime :

```
nas1:~# mount -o acl,noatime,nodiratime /dev/drbd0 /export
nas1:~# mount | grep drbd0
/dev/drbd0 on /export type ext3 (rw,acl,noatime,nodiratime)
nas1:~#
```

Modifier le fichier /etc/default/nfs-kernel-server :

```
# Number of servers to start up
# Vous pouvez augmenter le nombre de serveurs NFS selon la charge.
RPCNFSDCOUNT=8

# Runtime priority of server (see nice(1))
RPCNFSDPRIORITY=0

# Options for rpc.mountd.
# If you have a port-based firewall, you might want to set up
# a fixed port here using the --port option. For more information,
# see rpc.mountd(8) or http://wiki.debian.org/?SecuringNFS
# Comme nous allons surveiller ce service, il faut définir statiquement
# le port d'écoute.
RPCMOUNTDOPTS="-p 32000"

# Do you want to start the svcgssd daemon? It is only required for
# Kerberos exports.
NEED_SVCGSSD=

# Options for rpc.svcgssd.
RPCSVCGSSDOPTS=
```

Modifier le fichier /etc/default/nfs-common :

```
# If you do not set values for the NEED_ options, they will be attempted
# autodetected; this should be sufficient for most people. Valid
# alternatives for the NEED_ options are "yes" and "no".

# Options for rpc.statd.
# Should rpc.statd listen on a specific port? This is especially useful
# when you have a port-based firewall. To use a fixed port, set this
# this variable to a statd argument like: "--port 4000 --outgoing-port
# 4001". For more information, see rpc.statd(8) or
# http://wiki.debian.org/?SecuringNFS
# Nous définissons statiquement les ports d'écoute.
STATDOPTS="--name nas --port 31000 --outgoing-port 31001"

# Some kernels need a separate lockd daemon; most don't. Set this if you
# want to force an explicit choice for some reason.
NEED_LOCKD=
```

```
# Do you want to start the idmapd daemon? It is only needed for NFSv4.
# Nécessaire pour NFSv4.
NEED_IDMAPD=yes

# Do you want to start the gssd daemon? It is required for Kerberos
mounts.
NEED_GSSD=
```

Création des répertoires à partager dans le montage /export (/dev/drbd0) :

```
nas1:~# mkdir -p /export/www /export/home
```

Copiez les homedirectories existantes vers le partage /export/home :

```
nas1:~# cp -a /home/* /export/home/
```

Configuration du partage NFS v4 en modifiant le fichier /etc/exports :

```
# /etc/exports: the access control list for filesystems which may be
# exported to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync) hostname2(ro,sync)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt)
# /srv/nfs4/homes  gss/krb5i(rw,sync)
#
# Répertoire à partager.
# IP du client autorisé.
# Options :
#   rw = lecture/écriture.
#   sync = Ne répondre aux requêtes qu'après l'exécution de tous les
# changements sur le support réel.
#   fsid = Utilisé avec NFSv4, la valeur 0 signifie que le point de
# partage exporté sera une racine globale.
#   acl = Prise en compte des ALCs.
#   no_subtree_check = Cette option neutralise la vérification de
# sous-répertoires, ce qui a des subtiles implications au niveau de la
# sécurité, mais peut améliorer la fiabilité dans certains cas.
#   root_squash = Transformer les requêtes d'UID/GID 0 en l'UID/GID
# anonyme.
/export
172.16.0.8(rw,sync,fsid=0,acl,no_subtree_check,root_squash)
```

Configuration de l'écoute des services sur une adresse IP virtuelle

Arrêtez tous les services sur nas1 :

```
nas1:~# /etc/init.d/nscd stop
nas1:~# /etc/init.d/slapd stop
nas1:~# /etc/init.d/postfix stop
nas1:~# /etc/init.d/postgrey stop
nas1:~# /etc/init.d/policyd-weight stop
nas1:~# /etc/init.d/spamassassin stop
nas1:~# /etc/init.d/clamav-daemon stop
nas1:~# /etc/init.d/clamav-freshclam stop
nas1:~# /etc/init.d/nfs-common stop
nas1:~# /etc/init.d/nfs-kernel-server stop
nas1:~# /etc/init.d/portmap stop
```

Modifiez le fichier /etc/default/slapd en décommentant/commentant ces 2 lignes :

```
...
SLAPD_SERVICES="ldap://127.0.0.1:389/ ldap://172.16.0.19/
ldaps://172.16.0.19/"
#SLAPD_SERVICES="ldap://127.0.0.1:389/ ldap://172.16.0.17/
ldaps://172.16.0.17/"
...
```

Modifiez le fichier /etc/default/postgrey en décommentant/commentant ces 2 lignes :

```
...
#POSTGREY_OPTS="--inet=172.16.0.17:60000"
POSTGREY_OPTS="--inet=172.16.0.19:60000"
...
```

Modifiez le fichier /etc/policyd-weight.conf en décommentant/commentant ces 2 lignes :

```
$BIND_ADDRESS = '172.16.0.19';
#$BIND_ADDRESS = '172.16.0.17';
```

Installation des services redondants sur le noeud secondaire (sans les configurer)

Installation des packages

Sur nas2.alex.dmz :
slapd ldap-utils postfix postfix-ldap libnss-ldap nscd postgrey policyd-weight procmail spamassassin spamc clamav clamav-daemon arj unzoo unace unalz unzip unrar

Vous pouvez ajouter le dépôt « debian-volatile » pour obtenir les dernières versions de certains logiciels de filtrage et de détection.

Ajoutez cet ligne au fichier /etc/apt/sources.list :

```
...
```

```
deb http://volatile.debian.org/debian-volatile etch/volatile main
```

Répondre dans l'interface de configuration de Postfix :
Type de configuration : Pas de configuration

Installez aussi le package clamassassin (Debian Lenny) :

```
nas2:~# dpkg -i clamassassin_1.2.4-1_all.deb
```

Arrétez tous les services sur nas2 :

```
nas2:~# /etc/init.d/nscd stop
nas2:~# /etc/init.d/slapd stop
nas2:~# /etc/init.d/postfix stop
nas2:~# /etc/init.d/postgrey stop
nas2:~# /etc/init.d/policyd-weight stop
nas2:~# /etc/init.d/spamassassin stop
nas2:~# /etc/init.d/clamav-daemon stop
nas2:~# /etc/init.d/clamav-freshclam stop
nas2:~# /etc/init.d/nfs-common stop
nas2:~# /etc/init.d/nfs-kernel-server stop
nas2:~# /etc/init.d/portmap stop
```

Il faut créer ces fichiers, sinon il y aura des erreurs plus loin :

```
nas2:~# rm -f /etc/aliases.db
nas2:~# touch /etc/aliases.db
nas2:~# rm -f /etc/mailname
nas2:~# touch /etc/mailname
nas2:~# rm -f /etc/procmailrc
nas2:~# touch /etc/procmailrc
nas2:~# rm -f /etc/policyd-weight.conf
nas2:~# touch /etc/policyd-weight.conf
```

Supprimez le démarrage des services au boot des serveurs

Heartbeat se chargera de démarrer les services redondants, il faut donc que tous ces services ne démarrent pas au boot de la machine.
Créez un répertoire /root/RC afin de sauvegarder les paramètres de démarrage des services :

```
nas2:~# update-rc.d -f nscd remove > /root/RC/nscd
nas2:~# update-rc.d -f slapd remove > /root/RC/slapd
nas2:~# update-rc.d -f postfix remove > /root/RC/postfix
nas2:~# update-rc.d -f postgrey remove > /root/RC/postgrey
nas2:~# update-rc.d -f policyd-weight remove > /root/RC/policyd-weight
nas2:~# update-rc.d -f spamassassin remove > /root/RC/spamassassin
nas2:~# update-rc.d -f clamav-daemon  remove > /root/RC/clamav-daemon
nas2:~# update-rc.d -f clamav-freshclam  remove > /root/RC/clamav-
```

```
freshclam
nas2:~# update-rc.d -f nfs-common remove > /root/RC/nfs-common
nas2:~# update-rc.d -f nfs-kernel-server remove > /root/RC/nfs-kernel-
server
nas2:~# update-rc.d -f portmap remove > /root/RC/portmap
```

Réalisez la même opération sur nas1.

Copie des répertoires et fichiers de configuration des services vers la ressource commune des deux noeuds

Sur nas1, créez les répertoires suivants (avec /dev/drbd0 toujours monté dans /export) :

```
nas1:~# mkdir -p /export/bin /export/var/lib /export/var/log /export/var/
spool /export/etc/default /export/etc/init.d /export/usr/share
/export/etc/clamav
```

Copiez la configuration de chaque service vers /export :

```
nas1:~# cp -a /etc/nscd.conf /export/etc/
nas1:~# cp -a /var/db /export/var/

nas1:~# cp -a /etc/ldap /export/etc/
nas1:~# cp -a /var/lib/ldap /export/var/lib/
nas1:~# cp -a /var/spool/slurpd /export/var/spool/
nas1:~# cp -a /etc/default/slapd /export/etc/default/
nas1:~# cp -a /etc/libnss-ldap.conf /export/etc/
nas1:~# cp -a /etc/nsswitch.conf /export/etc/

nas1:~# cp -a /etc/postfix /export/etc/
nas1:~# cp -a /etc/mailname /export/etc/
nas1:~# cp -a /etc/aliases* /export/etc/
nas1:~# cp -a /var/spool/postfix /export/var/spool/
nas1:~# rm -r /export/var/spool/postfix/etc/*
/export/var/spool/postfix/pid/*

nas1:~# cp -a /etc/procmailrc /export/etc/

nas1:~# cp -a /etc/default/postgrey /export/etc/default/
nas1:~# cp -a /etc/postgrey /export/etc/
nas1:~# cp -a /var/lib/postgrey /export/var/lib/

nas1:~# cp -a /etc/policyd-weight.conf /export/etc/

nas1:~# cp -a /etc/default/spamassassin /export/etc/default/
nas1:~# cp -a /etc/spamassassin /export/etc/
nas1:~# cp -a /usr/share/spamassassin /export/usr/share/

nas1:~# cp -a /etc/clamav /export/etc/
nas1:~# cp -a /etc/default/clamassassin /export/etc/default/
nas1:~# cp -a /var/lib/clamav /export/var/lib/
nas1:~# cp -a /var/log/clamav /export/var/log/
```

```
nas1:~# cp -a /etc/exports /export/etc/
nas1:~# cp -a /etc/default/nfs-* /export/etc/default/
nas1:~# cp -a /var/lib/nfs /export/var/lib/
```

Note : Postfix écrit ses logs dans /var/log/mail.* en plus de syslog. Mais étrangement, il ne suit pas les liens symboliques et écrit dans /var/log/mail.* au lieu de /export/...
C'est pour cette raison que je ne les copie pas, ils resteront locaux.

Modifiez le fichier /etc/nsswitch.conf sur nas1 :

```
passwd:     compat
group:      compat
shadow:     compat

hosts:      files dns
networks:   files

protocols:  db files
services:   db files
ethers:     db files
rpc:        db files

netgroup:   nis
```

Configuration de DRBDlinks sur les deux noeuds

Installation des packages

Sur nas1.alex.dmz et nas2.alex.dmz :
drbdlinks

Modifiez le fichier /etc/drbdlinks.conf :

```
# If passed an option of 1, SELinux features will be used.  If 0, they
# will not.  The default is to auto-detect if SELinux is enabled.  If
# enabled, created links will be added to the SELinux context using
# chcon -h -u <USER> -r <ROLE> -t <TYPE>, where the values plugged
# in this command are pulled from the original file.
#selinux(1)
selinux(0)

# If passed an option of 1, syslog will be restarted if any links are
# changed.  If 0, syslog is not restarted after links are changed.
#restartsyslog(1)
restartsyslog(0)

# If passed an option of 1, "mount -o bind" is used instead of symbolic
# links. If 0, symbolic links are used.  This only applies to link()
# calls made after this call, and until usebindmount() is called with a
```

```
# different value. In other words, usebindmount can be called with 1 and
# then several link()s created that use bind mounts, then with 0 and
# other link()s which will be symlinks.
#usebindmount(1)
usebindmount(0)

# One mountpoint must be listed.  This is the location where the DRBD
# drive is mounted.
#mountpoint('/shared')
mountpoint('/export')

# Multiple "link" lines may be listed, one for each link that needs to be
# set up into the above shared mountpoint. If "link()" is passed one
# argument, it is assumed that it is linked into that name under the
# mountpoint above. Otherwise, you can specify a second argument which is
# the location of the file on the shared partition.
#
# For example, if mountpoint is "/shared" and you call
# "link('/etc/httpd')", it is equivalent to calling
# "link('/etc/httpd', '/shared/etc/httpd')".
#link('/etc/httpd')
#link('/var/lib/pgsql/')
#
# /home
link('/home/')
# NSS
link('/etc/nscd.conf')
link('/var/db/')
# LDAP
link('/etc/ldap/')
link('/var/lib/ldap/')
link('/var/spool/slurpd/')
link('/etc/default/slapd')
link('/etc/libnss-ldap.conf')
link('/etc/nsswitch.conf')
# Mail
link('/etc/postfix/')
link('/etc/mailname')
link('/etc/aliases')
link('/etc/aliases.db')
link('/var/spool/postfix/')
#
link('/etc/procmailrc')
#
link('/etc/default/postgrey')
link('/etc/postgrey/')
link('/var/lib/postgrey/')
#
link('/etc/policyd-weight.conf')
#
link('/etc/default/spamassassin')
link('/etc/spamassassin/')
link('/usr/share/spamassassin/')
#
link('/etc/clamav/')
link('/etc/default/clamassassin')
link('/var/lib/clamav/')
link('/var/log/clamav/')
```

```
# NFS
link('/etc/exports')
link('/etc/default/nfs-common')
link('/etc/default/nfs-kernel-server')
link('/var/lib/nfs/')
#link('/etc/init.d/nfs-kernel-server')
#link('/etc/init.d/portmap')
```

Copiez le fichier /etc/drbdlinks.conf sur nas2 :

```
nas1:~# scp /etc/drbdlinks.conf root@172.16.0.18:/etc/
```

La commande drbdlinks :

```
nas1:~# drbdlinks status
info: DRBD Links stopped (not set up)
nas1:~#
```

Vous pouvez vérifier la création des liens par DRBDlinks :

```
nas1:~# drbdlinks start

nas1:~# drbdlinks status
info: DRBD Links OK (present)
nas1:~#
```

Note :Familiarisez-vous avec les commandes DRBD et DRBDlinks avant de passer à Heartbeat.

Configuration de Heartbeat sur les deux noeuds

Installation des packages

Sur nas1.alex.dmz et nas2.alex.dmz :
heartbeat mailx

Sur nas1.

Créez le fichier /etc/ha.d/ha.cf :

```
# Fichier de configuration pour le noeud : nas1
#
#debugfile /var/log/ha-debug
#logfile /var/log/ha-log
logfacility daemon
```

```
# Intervalle de vérification de la santé de l'autre noeud.
keepalive 2

# Le délai avant d'annoncer un avertissement si l'autre noeud ne répond
pas.
warntime 10

# Au bout de combien de temps considérer l'autre noeud comme mort.
deadtime 30

# Délai à attendre pour la phase de démarrage d'un noeud, avant de le
# considérer comme mort.
# Doit être au minimum deux fois plus important que deadtime.
initdead 120

# Noms des deux noeuds Heartbeat (résolvable par la commande hostname).
node nas1
node nas2

# Numéro de port udp pour la communication entre les noeuds.
# Vérification de la santé de l'autre noeud Heartbeat.
udpport 694

# Méthode de communication IP entre les noeuds Heartbeat sur une ou
# plusieurs liaisons dédiées (1 câble croisé au minimum) :
# - par broadcast (bcast interfaces)
# - par unicast (ucast interface adresse_IP_de_l'autre_noeud)
# - par multicast (mcast interface groupe_mcast port_udp ttl 0)
#
# J'ai choisi la méthode par unicast.
# Déclaration des adresses IP pour communiquer avec le noeud nas2.
ucast bond0 192.168.0.2
ucast eth0 172.16.0.18
#
# Ce fichier peut être identique sur les deux noeuds si vous utilisez
# la méthode par broadcast à la place de l'unicast.
#bcast eth0 bond0

# Il est conseillé de connecter un ou plusieurs ports série entre les
# deux noeuds Heartbeat, pour renforcer les moyens de communication.
# Si vous ne pouvez pas, parce que vos machines sont dans des salles
# différentes, ajoutez une deuxième carte réseau connectée en câble
# croisé.
#
# baud 19200
# serial /dev/ttyS0 /dev/ttyS1

# Si le noeud primaire ne répond plus ou est arrêté volontairement, les
# services sont démarrés sur le noeud secondaire.
# Si le noeud primaire répond ou est démarré, les services resteront sur
# le noeud secondaire.
auto_failback off

# Mécanisme de détection de pannes réseau, afin de prendre de meilleures
# décisions sur les noeuds Heartbeat.
respawn hacluster /usr/lib/heartbeat/ipfail
```

```
# Adresses IP des machines de votre réseau censées répondre au ping
# (ici un DNS et SMTP). Il est conseillé d'ajouter une autre carte réseau
# connectée sur un autre switch. Dans mon schéma d'implantation, je n'ai
# qu'un seul switch principal pour mon réseau interne DMZ.
ping 172.16.0.1 172.16.0.7
```

Copiez le fichier /etc/ha.d/ha.cf sur nas2 et changez les adresses IP unicast :

```
...
# Déclaration des adresses IP pour communiquer avec le noeud nas1.
ucast bond0 192.168.0.1
ucast eth0 172.16.0.17
...
```

Créez le fichier /etc/ha.d/authkeys :

```
# - Numéro de l'authentification.
# - Méthode d'authentification :
# crc = authentification pas sécurisé, dans un réseau de confiance
# uniquement (liaison série), faible utilisation CPU.
# md5 = authentification sécurisé, dans un réseau non sécurisé, utilise
# plus de ressources CPU, réclame une clef.
# sha1 = authentification très sécurisé, dans un réseau non sécurisé,
# utilise beaucoup de ressources CPU, réclame une clef.
#
# Création d'une clef md5 :
# dd if=/dev/urandom count=4 2>/dev/null | md5sum | cut -c1-32
# Création d'une clef sha1 :
# dd if=/dev/urandom count=4 2>/dev/null | openssl dgst -sha1
#
auth 1
1 md5 2f0afe8a9b9f191a0a829f76fc2ef3a4
```

Changez les droits sur ce fichier :

```
nas1:~# chmod 600 /etc/ha.d/authkeys
```

Créez le fichier /etc/ha.d/haresources :

```
# - Nom du noeud préférentiel pour cette ressource.
# - Modification du statut DRBD (Primary/Secondary) pour la ressource
nas.
# - montage/démontage du périphérique /dev/drbd0 dans /export.
# - création/suppression de l'allias IP.
# - démarrage/arrêt des scripts/services se trouvant dans
# /etc/init.d/ ou /etc/ha.d/resource.d/
# - Envoit de mails d'information pour ce groupe de ressource.
nas1 \
drbddisk::nas \
Filesystem::/dev/drbd0::/export::ext3::acl,noatime,nodiratime \
```

```
IPaddr2::172.16.0.19/24/eth0 \
setGoodRights drbdlinks Delay::30::10 killAllServices slapd nscd clamav-
daemon clamav-freshclam spamassassin postgrey policyd-weight portmap nfs-
kernel-server nfs-common postfix \
MailTo::superviseur@alex.lan,superviseur@unfai.com,superviseur@unautrefai
.com::Groupe-NFS_MAIL_LDAP
```

Lors de l'acquisition d'une ressource, Hearbeat passe l'argument "**start**" aux scripts, selon l'ordre ou ils sont écrit dans le fichier /etc/ha.d/haresources.
Lors du relâchement d'une ressource, Hearbeat passe l'argument "**stop**" aux scripts, dans l'ordre inverse ou ils sont écrit dans le fichier /etc/ha.d/haresources.

J'ai créé des scripts sur mesure :

setGoodRights, applique les droits sur les fichiers/répertoires de configuration des services lors du basculement d'un noeud à l'autre.

killAllServices, tue tous les services restés "coincés" malgré l'arrêt standard, afin de libérer le périphérique DRBD pour le démonter.

Créez le fichier /etc/ha.d/resource.d/setGoodRights et rendez-le exécutable :

```
#!/bin/bash
#
# Applique les bons droits pour les services suivants.
# Openldap, Postfix, Postgrey, Clamav et NFS.

path='/export'

if [ "$1" = 'start' ]
then
    /bin/chown -R openldap: $path/etc/ldap/tls/*
    /bin/chown -R openldap: $path/var/lib/ldap
    /bin/chown -R openldap: $path/var/spool/slurpd

    /bin/chown -R postfix:postdrop $path/var/spool/postfix/*
    /bin/chown -R postfix: $path/var/spool/postfix/private
    /bin/chown -R root: $path/var/spool/postfix/pid/*
    /bin/chmod +rw $path/var/spool/postfix/public/*

    /bin/chown -R postgrey: $path/var/lib/postgrey

    /bin/chown clamav:adm $path/etc/clamav/freshclam.conf
    /bin/chown -R clamav:adm $path/var/log/clamav
    /bin/chown -R clamav: $path/var/lib/clamav

    /bin/chown -R statd: $path/var/lib/nfs
fi

exit 0
```

Créez le fichier /etc/ha.d/resource.d/killAllServices et rendez-le exécutable :

```
#!/bin/bash
#
# Permet de tuer les services récalcitrants.
# NFS, Postfix, Policyd-weight, Postgrey, SpamAssassin, Clamav, Openldap
et Monit.

PIPEFS_MOUNTPOINT=/var/lib/nfs/rpc_pipefs

function killall_services(){
    /usr/bin/killall -q -9 procmail rpc.gssd rpc.idmapd rpc.lockd
rpc.statd rpc.mountd rpc.svcgssd nfsd nfsd4 portmap qmgr pickup master
policyd-weight postgrey spamd freshclam clamd nscd slapd monit

    /bin/umount $PIPEFS_MOUNTPOINT >/dev/null 2>&1
    /usr/sbin/exportfs -au
    /usr/sbin/exportfs -f
}

if [ "$1" = 'stop' ]
then
    # Lance 2 fois la fonction "killall_services".
    killall_services
    sleep 2
    killall_services

    # Efface les fichiers de pid.
    rm -f /var/run/slapd/slapd.pid /var/spool/postfix/pid/master.pid
/var/run/clamav/*.pid /var/run/spamd.pid /var/run/postgrey.pid
/var/run/policyd-weight.pid /var/run/rpc.mountd.pid
/var/run/rpc.statd.pid /var/run/portmap.pid /var/run/nscd/nscd.pid
fi

exit 0
```

Copiez les fichiers suivant sur nas2 :

/etc/ha.d/authkeys
/etc/ha.d/haresources
/etc/ha.d/resource.d/setGoodRights
/etc/ha.d/resource.d/killAllServices

Arrêtez drbdlinks :

```
nas1:~# drbdlinks stop
```

Démontez /export :

```
nas1:~# umount /export
```

Passez l'état de /dev/drbd0 à Secondary/Secondary :

```
nas1:~# drbdsetup /dev/drbd0 secondary

nas1:~# drbdsetup /dev/drbd0 state
Secondary/Secondary
nas1:~#
```

Vérifiez que les deux adresses que Heartbeat va pinger sont joignable et démarrez Heartbeat sur les deux noeuds :

```
nas1:~# /etc/init.d/heartbeat start
Starting High-Availability services:
Done.
nas1:~#
```

```
nas2:~# /etc/init.d/heartbeat start
Starting High-Availability services:
Done.
nas2:~#
```

Contrôlez sur d'autres consoles les logs de Heartbeat en temps réel :

```
nas1:~# tail -f /var/log/syslog
...
```

```
nas2:~# tail -f /var/log/syslog
...
```

Si tout se passe bien, sur le noeud préféré (nas1), Heartbeat doit logger ces étapes :

```
- Vérification de l'existence de l'autre noeud.
- Vérification de l'existence des adresses tierces (les machines
répondant aux pings).
- Passage au statut actif.
- Vérification que l'état de l'autre noeud est stable.
- Acquisition des ressources locales, en exécutant dans l'ordre les
commandes contenues dans le fichier /etc/ha.d/haresources (avec argument
start).
- Vérification que chaque commandes se passent bien avant d'exécuter la
suivante.
```

Sur le noeud secondaire (nas2), Heartbeat doit logger ces étapes :

```
- Vérification de l'existence de l'autre noeud.
- Vérification de l'existence des adresses tierces (les machines
répondant aux pings).
- Passage au statut actif.
- Vérification que l'état de l'autre noeud est stable.
```

Pour vérifier que l'allias IP est activé :

```
nas1:~# ip addr list eth0 | grep -w inet
    inet 172.16.0.17/24 brd 172.16.0.255 scope global eth0
    inet 172.16.0.19/24 brd 172.16.0.255 scope global secondary eth0
nas1:~#
```

```
nas2:~# ip addr list eth0 | grep -w inet
    inet 172.16.0.18/24 brd 172.16.0.255 scope global eth0
nas2:~#
```

Les commandes de Heartbeat

Les commandes **hb_standby** et **hb_takeover** sont utilisées pour déplacer les services d'un serveur à l'autre si les deux serveurs sont disponibles.
Ils permettent de contrôler manuellement quel noeud contrôle quelle ressource à un instant donné.

hb_standby :
/usr/lib/heartbeat/hb_standby [all|foreign|local|failback]

Pour obtenir la liste des options de hb_standby, tapez la commande : "*/usr/lib/heartbeat/hb_standby --help*"
ATTENTION : pour des raisons de compatibilité, la commande hb_standby sans option est équivalente à "*hb_standby all*" !

Si vous lancez la commande "*/usr/lib/heartbeat/hb_standby local*" sur nas1 :
Toutes les ressources pour lesquelles nas1 **EST** le noeud préférentiel seront migrées vers le noeud secondaire nas2.

Sur le noeud préféré (nas1), Heartbeat doit loguer ces étapes :

```
- Vérification que l'état de l'autre noeud est stable.
- Informe le noeud secondaire (nas2) qu'il va libérer les ressources pour
se mettre en veille.
- Libère les ressources locales, en exécutant dans l'ordre inverse les
commandes contenues dans le fichier /etc/ha.d/haresources (avec argument
stop).
- Vérification que la transition est complète.
- Vérification que l'état de l'autre noeud est stable.
```

Sur le noeud secondaire (nas2), Heartbeat doit loguer ces étapes :

```
- Réception de l'information que le noeud préféré (nas1) veut libérer les
ressources pour se mettre en veille.
- Déclare le noeud préféré (nas1) comme instable.
- Acquisition des ressources locales, en exécutant dans l'ordre les
commandes contenues dans le fichier /etc/ha.d/haresources (avec argument
start).
- Vérification que chaque commandes se passent bien avant d'exécuter la
```

```
suivante.
- Vérification que la transition est complète.
- Vérification que l'état de l'autre noeud est stable.
```

Si vous lancez la commande "*/usr/lib/heartbeat/hb_standby foreign* ou *failback*" sur nas1 :
Toutes les ressources pour lesquelles nas1 **N'EST PAS** le noeud préférentiel seront migrées vers le noeud secondaire nas2.

Dans notre exemple, il n'y a qu'une ressource sur le noeud préférentiel nas1.
nas2 est en veille si nas1 meurt.
Il est possible de répartir les ressources, c'est à dire de mettre une partie des services sur nas1 (ex: ldap) et une autre sur nas2 (ex: mail, nfs).
Il faut dans ce cas qu'il y ait deux ressources DRBD, de cette manière, nas1 et nas2 travail tous les deux. Par contre la gestion en cas de panne devient plus complexe.

Si vous lancez la commande "*/usr/lib/heartbeat/hb_standby all*" :
Quel que soit le noeud sur lequel cette commande est exécutée, toutes les ressources sont abandonnées et l'autre noeud en prend le contrôle.

hb_takeover :
/usr/lib/heartbeat/hb_takeover [all|foreign|local|failback]

hb_takeover fait l'inverse de la commande hb_standby.
Si vous lancez la commande "*/usr/lib/heartbeat/hb_takeover all*" sur un des noeuds, celui-ci prendra le contrôle des ressources de l'autre noeud.

Exécuter hb_takeover sur un noeud est équivalent à exécuter hb_standby sur l'autre noeud.

Note :Familiarisez-vous avec les commandes de Heartbeat afin de valider vos configurations et comprendre les mécanismes de chaque éléments présentés jusqu'ici.

Configuration de Monit sur les deux noeuds

Installation des packages

Sur nas1.alex.dmz et nas2.alex.dmz :
monit

Supprimez le démarrage du service Monit au boot des serveurs :

```
nas1:~# update-rc.d -f monit  remove > /root/RC/monit
```

Réalisez la même opération sur nas2.

Ajoutez au fichier /etc/ha.d/haresources le service "monit" et copiez-le sur nas2 :

```
...
setGoodRights drbdlinks Delay::30::10 killAllServices slapd nscd clamav-
daemon clamav-freshclam spamassassin postgrey policyd-weight portmap nfs-
kernel-server nfs-common postfix monit \
...
```

Pour que le logiciel Monit puisse surveiller les services liés à NFS sur les deux noeuds, il faut modifier les scripts d'init suivants.

Ajoutez à la fin de la section "**start**" et "**stop**" du fichier /etc/init.d/nfs-kernel-server ces commandes :

```
...
# See how we were called.
case "$1" in
  start)

...

else
          log_warning_msg "Not starting $DESC: no exports."
    fi

    # arnofear.free.fr
    /bin/pidof rpc.mountd > /var/run/rpc.mountd.pid
    ;;

  stop)
    log_daemon_msg "Stopping $DESC"

...

    if mountpoint -q /proc/nfs/nfsd
    then
        $PREFIX/sbin/exportfs -f
    fi

    # arnofear.free.fr
    rm -f /var/run/rpc.mountd.pid
    ;;

  status)
...
```

Ajoutez à la fin de la section "**start**" et "**stop**" du fichier /etc/init.d/portmap ces commandes :

```
...
case "$1" in
    start)
```

```
...

    else
      if [ -f /var/run/portmap.state ]; then
        pmap_set </var/run/portmap.state
        rm -f /var/run/portmap.state
      fi
    fi

    # arnofear.free.fr
    /bin/pidof portmap > /var/run/portmap.pid
    ;;
    stop)
    log_begin_msg "Stopping portmap daemon..."
    pmap_dump >/var/run/portmap.state
    start-stop-daemon --stop --quiet --oknodo --exec /sbin/portmap
    log_end_msg $?

    # arnofear.free.fr
    rm -f /var/run/portmap.pid
    ;;
    force-reload)
...
```

Copiez ces scripts vers /export :

```
nas1:~# cp -a /etc/init.d/nfs-kernel-server /export/etc/init.d/
nas1:~# cp -a /etc/init.d/portmap /export/etc/init.d/
```

Décommentez dans le fichier /etc/drbdlinks.conf les deux dernières lignes :

```
...
link('/etc/init.d/nfs-kernel-server')
link('/etc/init.d/portmap')
```

Copiez le fichier /etc/drbdlinks.conf sur nas2.

Modifiez le fichier /etc/default/monit :

```
# Defaults for monit initscript
# sourced by /etc/init.d/monit
# installed at /etc/default/monit by maintainer scripts
# Fredrik Steen <stone@debian.org>

# You must set this variable to for monit to start
startup=1

# To change the intervals which monit should run uncomment
# and change this variable.
# Pour que Monit vérifie les services toutes les 60 secondes.
CHECK_INTERVALS=60
```

Copiez le fichier /etc/default/monit sur nas2.

Modifiez le fichier /etc/monit/monitrc :

ATTENTION les tabulations font partie de la syntaxe.

```
# Fichier de configuration pour le noeud : nas1
#
## Global section
#
set logfile syslog facility log_daemon
# Adresse du serveur mail.
set mailserver localhost
# File d'attente des mails s'ils ne peuvent pas être envoyés.
set eventqueue basedir /var/spool/monit slots 100
# Adresses mail du superviseur qui recevront tous les types d'alerte.
set alert superviseur@alex.lan
set alert superviseur@unfai.com
set alert superviseur@unautrefai.com
# Vous pouvez choisir quels types d'informations Monit enverra pour
# chaque adresse mail.
#set alert uncompte@fai only on { timeout }
#
# Activation de l'interface Web de Monit pour visualiser l'état des
# services surveillés sur le port 2812.
# Connexion autorisée pour le réseau 172.17.0.0/16.
# Authentification avec le login "monit" et le mot de passe "mypassword".
# Connexion en https uniquement, avec utilisation d'un certificat/clef.
# Génération du fichier SSL détaillé plus bas.
set httpd port 2812
    allow 172.17.0.0/16
    allow monit:mypassword
    ssl enable
    pemfile /etc/monit/ssl/monit.pem


## Services
#
# Liste des ressources à surveiller.
#
# Charge serveur
# Si la charge est supérieur à 30 durant 8 cycles => envoyer une alerte.
check system localhost
    if loadavg (1min) > 30 for 8 cycles then alert
    group system

# Espace disque pour le périphérique /dev/drbd0
check device nas with path /dev/drbd0
    if space usage > 90% then alert
    if inode usage > 90% then alert
    group system

# Openldap
# - Surveille le processus slapd et le redémarre s'il est absent.
# - S'il y a plus de 50 processus slapd => redémarrage de slapd.
```

```
# - Si la connexion à l'IP virtuelle au port 389 en protocole LDAPv3
# avec un timeout de 10s échoue => redémarrage de slapd.
# - Si la connexion à l'IP virtuelle au port 389 en protocole LDAPv3
# avec un timeout de 10s échoue 3 fois en 3 cycles => exécution de la
# commande Heartbeat permettant de relâcher tous les services préférés,
# pour qu'ils soit repris par l'autre noeud.
check process openldap with pidfile /var/run/slapd/slapd.pid
    start program = "/etc/init.d/slapd start"
    stop program  = "/etc/init.d/slapd stop"
    if children > 50 then restart
    if failed host 172.16.0.19 port 389 protocol LDAP3 with timeout 10
seconds then restart
    if failed host 172.16.0.19 port 389 protocol LDAP3 with timeout 10
seconds for 3 times within 3 cycles then exec
"/usr/lib/heartbeat/hb_standby all"
    group server

# Postfix
check process postfix with pidfile /var/spool/postfix/pid/master.pid
    start program = "/etc/init.d/postfix start"
    stop  program = "/etc/init.d/postfix stop"
    if failed host 172.16.0.19 port 25 protocol SMTP with timeout 10
seconds then restart
    if failed host 172.16.0.19 port 25 protocol SMTP with timeout 10
seconds for 3 times within 3 cycles then exec
"/usr/lib/heartbeat/hb_standby all"
    group mail

# Clamav-daemon
# - Surveille le processus clamd et le redémarre s'il est absent.
# - Interroge le démon clamd par son socket et le redémarre s'il ne
# répond pas.
# - Si le service n'a pas redémarré 6 fois en 6 cycles => on ne surveille
# plus le service.
check process clamav-daemon with pidfile /var/run/clamav/clamd.pid
    start program = "/etc/init.d/clamav-daemon start"
    stop  program = "/etc/init.d/clamav-daemon stop"
    if failed unixsocket /var/run/clamav/clamd.ctl then restart
    if 6 restarts within 6 cycles then timeout
    group mail

# Clamav-freshclam
check process clamav-freshclam with pidfile /var/run/clamav/freshclam.pid
    start program = "/etc/init.d/clamav-freshclam start"
    stop  program = "/etc/init.d/clamav-freshclam stop"
    if 6 restarts within 6 cycles then timeout
    group mail

# SpamAssassin
check process spamassassin with pidfile /var/run/spamd.pid
    start program = "/etc/init.d/spamassassin start"
    stop  program = "/etc/init.d/spamassassin stop"
    if failed unixsocket /var/run/spamd.socket then restart
    if 6 restarts within 6 cycles then timeout
    group mail

# Postgrey
check process postgrey with pidfile /var/run/postgrey.pid
```

```
    start program = "/etc/init.d/postgrey start"
    stop  program = "/etc/init.d/postgrey stop"
    if failed host 172.16.0.19 port 60000 type TCP with timeout 10 seconds
then restart
    if failed host 172.16.0.19 port 60000 type TCP with timeout 10 seconds
for 3 times within 3 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group mail

# Policyd-weight
check process policyd-weight with pidfile /var/run/policyd-weight.pid
    start program = "/etc/init.d/policyd-weight start"
    stop  program = "/etc/init.d/policyd-weight stop"
    if failed host 172.16.0.19 port 12525 type TCP with timeout 10 seconds
then restart
    if failed host 172.16.0.19 port 12525 type TCP with timeout 10 seconds
for 3 times within 3 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group mail

# nfs-server
# Avec le fichier /etc/init.d/nfs-kernel-server modifié.
check process nfs-server with pidfile /var/run/rpc.mountd.pid
    start program = "/etc/init.d/nfs-kernel-server start"
    stop  program = "/etc/init.d/nfs-kernel-server stop"
    if failed host 172.16.0.19 port 32000 type TCP with timeout 10 seconds
then restart
    if failed host 172.16.0.19 port 32000 type TCP with timeout 10 seconds
for 3 times within 3 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server

# nfs-common
check process nfs-common with pidfile /var/run/rpc.statd.pid
    start program = "/etc/init.d/nfs-common start"
    stop  program = "/etc/init.d/nfs-common stop"
    if failed host 172.16.0.19 port 31000 type TCP with timeout 10 seconds
then restart
    if failed host 172.16.0.19 port 31000 type TCP with timeout 10 seconds
for 3 times within 3 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server

# nfs-portmap
# Avec le fichier /etc/init.d/portmap modifié.
check process nfs-portmap with pidfile /var/run/portmap.pid
    start program = "/etc/init.d/portmap start"
    stop  program = "/etc/init.d/portmap stop"
    if failed host 172.16.0.19 port 111 type TCP with timeout 10 seconds
then restart
    if failed host 172.16.0.19 port 111 type TCP with timeout 10 seconds
for 3 times within 3 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server

# Nscd
check process nscd with pidfile /var/run/nscd/nscd.pid
    start program = "/etc/init.d/nscd start"
    stop  program = "/etc/init.d/nscd stop"
    if 6 restarts within 6 cycles then timeout
    group server
```

Créez un répertoire pour stocker les informations SSL :

```
nas1:~# mkdir /etc/monit/ssl
```

Créez un fichier /etc/monit/ssl/monit.cnf :

```
# create RSA certs - Server

RANDFILE = ./openssl.rnd

[ req ]
default_bits = 1024
encrypt_key = yes
distinguished_name = req_dn
x509_extensions = cert_type

[ req_dn ]
countryName         = Country Name (2 letter code)
countryName_default = FR

stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = Haute-Savoie

localityName         = Locality Name (eg, city)
localityName_default = Alex

organizationName         = Organization Name (eg, company)
organizationName_default = Alex

organizationalUnitName         = Organizational Unit Name (eg, section)
organizationalUnitName_default = Alex

commonName         = Common Name (FQDN of your server)
commonName_default = localhost

emailAddress         = Email Address
emailAddress_default = root@localhost

[ cert_type ]
nsCertType = server
```

Lancez ces commandes pour générer le certificat/clef :

```
nas1:~# openssl req -new -x509 -days 3650 -nodes -config
/etc/monit/ssl/monit.cnf -out /etc/monit/ssl/monit.pem -keyout
/etc/monit/ssl/monit.pem

Generating a 1024 bit RSA private key
.....................++++++
......++++++
writing new private key to '/etc/monit/ssl/monit.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [Haute-Savoie]:
Locality Name (eg, city) [Alex]:
Organization Name (eg, company) [Alex]:
Organizational Unit Name (eg, section) [Alex]:
Common Name (FQDN of your server) [localhost]:
Email Address [root@localhost]:
nas1:~#
```

```
nas1:~# openssl gendh 512 >> /etc/monit/ssl/monit.pem

Generating DH parameters, 512 bit long safe prime, generator 2
This is going to take a long time
..+.........+..+...
nas1:~#
```

```
nas1:~# openssl x509 -subject -dates -fingerprint -noout -in
/etc/monit/ssl/monit.pem

subject= /C=FR/ST=Haute-
Savoie/L=Alex/O=Alex/OU=Alex/CN=localhost/emailAddress=root@localhost
notBefore=Nov 26 15:38:03 2007 GMT
notAfter=Nov 23 15:38:03 2017 GMT
SHA1
Fingerprint=1F:44:FD:3A:1E:FB:75:49:78:02:22:D4:27:65:0A:17:27:C3:49:FF
nas1:~#
```

Changez les droits :

```
nas1:~# chmod 600 /etc/monit/ssl/monit.pem
```

Copiez le répertoire /etc/monit/ssl sur nas2 :

```
nas1:~# scp -r /etc/monit/ssl root@172.16.0.18:/etc/monit/
```

Vous pouvez vérifier la syntaxe de cette manière :

```
nas1:~# /etc/init.d/monit syntax
Control file syntax OK
nas1:~#
```

Sur nas2.

Modifiez le fichier /etc/monit/monitrc :

```
# Fichier de configuration pour le noeud : nas2
#
## Global section
#
set logfile syslog facility log_daemon
set mailserver localhost
set eventqueue basedir /var/spool/monit slots 100
set alert superviseur@alex.lan
set alert superviseur@unfai.com
set alert superviseur@unautrefai.com
#set alert uncompte@fai only on { timeout }
set httpd port 2812
    allow 172.17.0.0/16
    allow monit:mypassword
    ssl enable
    pemfile /etc/monit/ssl/monit.pem

check system localhost
    if loadavg (1min) > 30 for 8 cycles then alert
    group system


## Services
#
# Espace disque pour le périphérique /dev/drbd0
check device nas with path /dev/drbd0
    if space usage > 90% then alert
    if inode usage > 90% then alert
    group system

# Openldap
# - Surveille le processus slapd et le redémarre s'il est absent.
# - S'il y a plus de 50 processus slapd => redémarrage de slapd.
# - Si la connexion à l'IP virtuelle au port 389 en protocole LDAPv3
# avec un timeout de 10s échoue => redémarrage de slapd.
# - Si le service n'a pas redémarré 6 fois en 6 cycles => on ne surveille
# plus le service.
# - On ne transfert pas les services vers le noeud primaire puisqu'il a
# dû rencontrer des problèmes si nous sommes maintenant sur le noeud
# secondaire.
check process openldap with pidfile /var/run/slapd/slapd.pid
    start program = "/etc/init.d/slapd start"
    stop program  = "/etc/init.d/slapd stop"
    if children > 50 then restart
    if failed host 172.16.0.19 port 389 protocol LDAP3 with timeout 10
seconds then restart
    if 6 restarts within 6 cycles then timeout
    group server

# Postfix
check process postfix with pidfile /var/spool/postfix/pid/master.pid
    start program = "/etc/init.d/postfix start"
    stop  program = "/etc/init.d/postfix stop"
```

```
    if failed host 172.16.0.19 port 25 protocol SMTP with timeout 10
seconds then restart
    if 6 restarts within 6 cycles then timeout
    group mail

# Clamav-daemon
check process clamav-daemon with pidfile /var/run/clamav/clamd.pid
    start program = "/etc/init.d/clamav-daemon start"
    stop  program = "/etc/init.d/clamav-daemon stop"
    if failed unixsocket /var/run/clamav/clamd.ctl then restart
    if 6 restarts within 6 cycles then timeout
    group mail

# Clamav-freshclam
check process clamav-freshclam with pidfile /var/run/clamav/freshclam.pid
    start program = "/etc/init.d/clamav-freshclam start"
    stop  program = "/etc/init.d/clamav-freshclam stop"
    if 6 restarts within 6 cycles then timeout
    group mail

# SpamAssassin
check process spamassassin with pidfile /var/run/spamd.pid
    start program = "/etc/init.d/spamassassin start"
    stop  program = "/etc/init.d/spamassassin stop"
    if failed unixsocket /var/run/spamd.socket then restart
    if 6 restarts within 6 cycles then timeout
    group mail

# Postgrey
check process postgrey with pidfile /var/run/postgrey.pid
    start program = "/etc/init.d/postgrey start"
    stop  program = "/etc/init.d/postgrey stop"
    if failed host 172.16.0.19 port 60000 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group mail

# Policyd-weight
check process policyd-weight with pidfile /var/run/policyd-weight.pid
    start program = "/etc/init.d/policyd-weight start"
    stop  program = "/etc/init.d/policyd-weight stop"
    if failed host 172.16.0.19 port 12525 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group mail

# nfs-server
check process nfs-server with pidfile /var/run/rpc.mountd.pid
    start program = "/etc/init.d/nfs-kernel-server start"
    stop  program = "/etc/init.d/nfs-kernel-server stop"
    if failed host 172.16.0.19 port 32000 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group server

# nfs-common
check process nfs-common with pidfile /var/run/rpc.statd.pid
    start program = "/etc/init.d/nfs-common start"
```

```
    stop  program = "/etc/init.d/nfs-common stop"
    if failed host 172.16.0.19 port 31000 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group server

# nfs-portmap
check process nfs-portmap with pidfile /var/run/portmap.pid
    start program = "/etc/init.d/portmap start"
    stop  program = "/etc/init.d/portmap stop"
    if failed host 172.16.0.19 port 111 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group server

# Nscd
check process nscd with pidfile /var/run/nscd/nscd.pid
    start program = "/etc/init.d/nscd start"
    stop  program = "/etc/init.d/nscd stop"
    if 6 restarts within 6 cycles then timeout
    group server
```

Configuration des machines fs1 et fs2

Configuration du channel bonding

Installation des packages

Sur fs1.alex.lan et fs2.alex.lan :
ifenslave-2.6 ethtool

Les machines auront trois cartes réseau :

- eth0 => Connecté sur un switch.
- eth1 et eth2 => Connectés entre chaque serveurs par câbles croisés. Utilisés pour le channel bonding.

Créez un fichier /etc/modprobe.d/bonding sur les deux serveurs, qui permettra de charger le module bonding :

```
alias bond0 bonding
options bonding miimon=100 mode=balance-rr
```

Modifiez le fichier /etc/network/interfaces sur **fs1** :

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

```
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 172.17.0.7
netmask 255.255.0.0
gateway 172.17.0.254
up route add -net 172.18.0.0/16 gw 172.17.0.253

# Bonding interface
auto eth1 eth2 bond0
allow-hotplug eth1 eth2 bond0
iface bond0 inet static
address 192.168.0.1
netmask 255.255.255.252
up ifenslave bond0 eth1 eth2
down ifenslave -d bond0 eth1 eth2
```

Modifiez le fichier /etc/network/interfaces sur **fs2** :

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

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 172.17.0.8
netmask 255.255.0.0
gateway 172.17.0.254
up route add -net 172.18.0.0/16 gw 172.17.0.253

# Bonding interface
auto eth1 eth2 bond0
allow-hotplug eth1 eth2 bond0
iface bond0 inet static
address 192.168.0.2
netmask 255.255.255.252
up ifenslave bond0 eth1 eth2
down ifenslave -d bond0 eth1 eth2
```

Reboutez les machines après ces modifications.

Lancez des pings depuis un des serveurs pour vérifier qu'il n'y a pas de pertes de paquets.

Synchronisation de l'heure sur les deux noeuds par NTP

Suivez la même démarche pour les serveurs fs1 et fs2 en adaptant les valeurs.

Configuration de DRBD sur les deux noeuds

Suivez la même démarche pour les serveurs fs1 et fs2 en adaptant les valeurs.

Configuration de NFS sur le noeud primaire

Installation des packages

Sur fs1.alex.lan et fs2.alex.lan :
nfs-kernel-server acl

Depuis fs1, réalisez le montage en activant le support des acls et modifiez le fichier /etc/default/nfs-kernel-server et /etc/default/nfs-common comme pour nas1.
Copiez les homedirectories existantes vers le répertoire /export/home (avec /dev/drbd0 toujours monté dans /export).

Modifiez le fichier /etc/exports :

```
# Partage pour la machine BackupPC et deux réseaux cette fois.
/export   172.17.0.2/32(rw,sync,fsid=0,acl,no_subtree_check,no_root_squash
) 172.17.0.0/16(rw,sync,fsid=0,acl,no_subtree_check,root_squash)
172.18.0.0/16(rw,sync,fsid=0,acl,no_subtree_check,root_squash)
```

Configuration de l'écoute des services sur une adresse IP virtuelle

Arrêtez le service Samba sur la machine pdc.alex.lan et bdc.alex.lan et modifiez leur fichier /etc/samba/smb.conf en décommentant/commentant ces 2 lignes :

```
...
wins server = 172.17.0.9
#wins server = 172.17.0.7
...
```

Arrêtez tous les services sur fs1, sauf Postfix qui sera configuré sur chaque noeuds :

```
fs1:~# /etc/init.d/samba stop
fs1:~# /etc/init.d/nfs-common stop
fs1:~# /etc/init.d/nfs-kernel-server stop
fs1:~# /etc/init.d/portmap stop
```

Installation des services redondants sur le noeud secondaire (sans les configurer)

Installation des packages

Sur fs2.alex.lan :

ldap-utils libnss-ldap samba smbclient postfix mailutils

Répondre dans l'interface de configuration de Postfix :
Type de configuration : Système satellite
Nom de courrier : fs2.alex.lan
Serveur relais SMTP : smtp.alex.dmz

Arrêtez tous les services sur fs2 :

```
fs2:~# /etc/init.d/samba stop
fs2:~# /etc/init.d/nfs-common stop
fs2:~# /etc/init.d/nfs-kernel-server stop
fs2:~# /etc/init.d/portmap stop
```

Supprimez le démarrage des services au boot des serveurs

Créez un répertoire /root/RC afin de sauvegarder les paramètres de démarrage des services :

```
fs2:~# update-rc.d -f samba remove > /root/RC/samba
fs2:~# update-rc.d -f nfs-common remove > /root/RC/nfs-common
fs2:~# update-rc.d -f nfs-kernel-server remove > /root/RC/nfs-kernel-
server
fs2:~# update-rc.d -f portmap remove > /root/RC/portmap
```

Réalisez la même opération sur fs1.

Copie des répertoires et fichiers de configuration des services vers la ressource commune des deux noeuds

Sur fs1, créez les répertoires suivants (avec /dev/drbd0 toujours monté dans /export) :

```
fs1:~# mkdir -p /export/var/lib /export/var/run /export/etc/default
/export/etc/init.d
```

Copiez la configuration de chaque services vers /export :

```
fs1:~# cp -a /etc/ldap /export/etc/
fs1:~# cp -a /etc/libnss-ldap.conf /export/etc/
fs1:~# cp -a /etc/nsswitch.conf /export/etc/

fs1:~# cp -a /etc/samba /export/etc/
fs1:~# cp -a /var/lib/samba /export/var/lib/
fs1:~# cp -a /var/run/samba /export/var/run/

fs1:~# cp -a /etc/exports /export/etc/
fs1:~# cp -a /etc/default/nfs-* /export/etc/default/
```

```
fs1:~# cp -a /var/lib/nfs /export/var/lib/
```

Modifiez le fichier /etc/nsswitch.conf sur fs1 :

```
passwd:     compat
group:      compat
shadow:     compat

hosts:      files dns
networks:   files

protocols:  db files
services:   db files
ethers:     db files
rpc:        db files

netgroup:   nis
```

Configuration de DRBDlinks sur les deux noeuds

Installation des packages

Sur fs1.alex.lan et fs2.alex.lan :
drbdlinks

Modifiez le fichier /etc/drbdlinks.conf :

```
selinux(0)

restartsyslog(0)

usebindmount(0)

mountpoint('/export')

# /home
link('/home/')
# LDAP
link('/etc/ldap/')
link('/etc/libnss-ldap.conf')
link('/etc/nsswitch.conf')
# Samba
link('/etc/samba/')
link('/var/lib/samba/')
link('/var/run/samba/')
# NFS
link('/etc/exports')
link('/etc/default/nfs-common')
link('/etc/default/nfs-kernel-server')
link('/var/lib/nfs/')
#link('/etc/init.d/nfs-kernel-server')
#link('/etc/init.d/portmap')
```

Copiez le fichier /etc/drbdlinks.conf sur fs2 :

```
fs1:~# scp /etc/drbdlinks.conf root@172.17.0.8:/etc/
```

Configuration de Heartbeat sur les deux noeuds

Installation des packages

Sur fs1.alex.lan et fs2.alex.lan :
heartbeat

Sur fs1.

Créez le fichier /etc/ha.d/ha.cf :

```
# Fichier de configuration pour le noeud : fs1
#
logfacility daemon
keepalive 2
warntime 10
deadtime 30
initdead 120

# Noms des deux noeuds Heartbeat.
node fs1
node fs2

udpport 694

# Déclaration des adresses IP pour communiquer avec le noeud fs2.
ucast bond0 192.168.0.2
ucast eth0 172.17.0.8

# baud 19200
# serial /dev/ttyS0

auto_failback off

respawn hacluster /usr/lib/heartbeat/ipfail

# Adresses IP des machines de votre réseau censées répondre au ping
# (ici un DNS et un routeur).
ping 172.17.0.1 172.17.0.254
```

Copiez le fichier /etc/ha.d/ha.cf sur fs2 et changez les adresses IP unicast :

```
...
# Déclaration des adresses IP pour communiquer avec le noeud fs1.
ucast bond0 192.168.0.1
```

```
ucast eth0 172.17.0.7
...
```

Créez le fichier /etc/ha.d/authkeys :

```
auth 1
1 md5 d34eea0e1bf8a9e4a7b80ab3e7c41faf
```

Changez les droits sur ce fichier :

```
fs1:~# chmod 600 /etc/ha.d/authkeys
```

Créez le fichier /etc/ha.d/haresources :

```
fs1 \
drbddisk::fs \
Filesystem::/dev/drbd0::/export::ext3::acl,noatime,nodiratime \
IPaddr2::172.17.0.9/16/eth0 \
setGoodRights drbdlinks Delay::10::10 killAllServices portmap nfs-kernel-
server nfs-common samba \
MailTo::superviseur@alex.lan,superviseur@unfai.com,superviseur@unautrefai
.com::Groupe-NFS_SAMBA
```

Créez un fichier /etc/ha.d/resource.d/setGoodRights sur mesure et rendez-le exécutable :

```
#!/bin/bash
#
# Applique les bons droits pour les services suivants.
# NFS.

path='/export'

if [ "$1" = 'start' ]
then
    /bin/chown -R statd: $path/var/lib/nfs
fi

exit 0
```

Créez un fichier /etc/ha.d/resource.d/killAllServices sur mesure et rendez-le exécutable :

```
#!/bin/bash
#
# Permet de tuer les services récalcitrants.
# NFS, Samba et Monit.

PIPEFS_MOUNTPOINT=/var/lib/nfs/rpc_pipefs
```

```
function killall_services(){
    /usr/bin/killall -q -9 rpc.gssd rpc.idmapd rpc.lockd rpc.statd
rpc.mountd rpc.svcgssd nfsd nfsd4 portmap smbd nmbd monit

    /bin/umount $PIPEFS_MOUNTPOINT >/dev/null 2>&1
    /usr/sbin/exportfs -au
    /usr/sbin/exportfs -f
}

if [ "$1" = 'stop' ]
then
    # Lance 2 fois la fonction "killall_services".
    killall_services
    sleep 2
    killall_services

    # Efface les fichiers de pid.
    rm -f /var/run/rpc.mountd.pid /var/run/rpc.statd.pid
/var/run/portmap.pid /var/run/samba/*.pid
fi

exit 0
```

Copiez les fichiers suivant sur fs2 :

/etc/ha.d/authkeys
/etc/ha.d/haresources
/etc/ha.d/resource.d/setGoodRights
/etc/ha.d/resource.d/killAllServices

Arrêtez drbdlinks :

```
fs1:~# drbdlinks stop
```

Démontez /export :

```
fs1:~# umount /export
```

Passez l'état de /dev/drbd0 à Secondary/Secondary :

```
fs1:~# drbdsetup /dev/drbd0 secondary

fs1:~# drbdsetup /dev/drbd0 state
Secondary/Secondary
fs1:~#
```

Vérifiez que les deux adresses que Heartbeat va pinger sont joignable et démarrez Heartbeat sur les deux noeuds :

```
fs1:~# /etc/init.d/heartbeat start
Starting High-Availability services:
Done.
fs1:~#
```

```
fs2:~# /etc/init.d/heartbeat start
Starting High-Availability services:
Done.
fs2:~#
```

Contrôlez sur d'autres consoles les logs de Heartbeat en temps réel.

Configuration de Monit sur les deux noeuds

Installation des packages

Sur fs1.alex.lan et fs2.alex.lan :
monit

Supprimez le démarrage du service Monit au boot des serveurs :

```
fs1:~# update-rc.d -f monit  remove > /root/RC/monit
```

Réalisez la même opération sur fs2.

Ajoutez au fichier /etc/ha.d/haresources le service "monit" et copiez-le sur fs2 :

```
...
setGoodRights drbdlinks Delay::10::10 killAllServices portmap nfs-kernel-
server nfs-common samba monit \
...
```

Pour que le logiciel Monit puisse surveiller les services liés à NFS, modifiez les scripts /etc/init.d/nfs-kernel-server et /etc/init.d/portmap.

Copiez ces scripts vers /export :

```
fs1:~# cp -a /etc/init.d/nfs-kernel-server /export/etc/init.d/
fs1:~# cp -a /etc/init.d/portmap /export/etc/init.d/
```

Décommentez dans le fichier /etc/drbdlinks.conf les deux dernière lignes :

```
...
link('/etc/init.d/nfs-kernel-server')
```

```
link('/etc/init.d/portmap')
```

Copiez le fichier /etc/drbdlinks.conf sur fs2.

Modifiez le fichier /etc/default/monit :

```
startup=1

CHECK_INTERVALS=60
```

Copiez le fichier /etc/default/monit sur fs2.

Modifiez le fichier /etc/monit/monitrc :

```
# Fichier de configuration pour le noeud : fs1
#
## Global section
#
set logfile syslog facility log_daemon
set mailserver localhost
set eventqueue basedir /var/spool/monit slots 100
set alert superviseur@alex.lan
set alert superviseur@unfai.com
set alert superviseur@unautrefai.com
#set alert uncompte@fai only on { timeout }

set httpd port 2812
    allow 172.17.0.0/16
    allow monit:mypassword
    ssl enable
    pemfile /etc/monit/ssl/monit.pem


## Services
#
# Liste des ressources à surveiller.
#
# Charge serveur
check system localhost
    if loadavg (1min) > 30 for 8 cycles then alert
    group system

# Espace disque pour le périphérique /dev/drbd0
check device nas with path /dev/drbd0
    if space usage > 90% then alert
    if inode usage > 90% then alert
    group system

# Samba (résolution NetBIOS)
# Il faut interroger les ports UDP 137 et 138.
check process samba-nmbd with pidfile /var/run/samba/nmbd.pid
    start program = "/etc/init.d/samba start"
```

```
    stop program  = "/etc/init.d/samba stop"
    if failed host 172.17.0.9 port 137 type UDP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 138 type UDP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 137 type UDP with timeout 10 seconds
for 6 times within 6 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    if failed host 172.17.0.9 port 138 type UDP with timeout 10 seconds
for 6 times within 6 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server

# Samba (session NetBIOS et CIFS)
# Il faut interroger les ports TCP 139 et 445.
# Ce service s'appuie sur des serveurs OpenLDAP pour fonctionner.
# smbd est dépendant de bdc.alex.lan et pdc.alex.lan, on ajoute l'option
# "depends on [service]". Si les serveurs LDAP ne sont pas joignable ce
# service ne sera pas arrêté pour autant puisque nous interrogeons des
# serveurs LDAP distant et non locaux.
check process samba-smbd with pidfile /var/run/samba/smbd.pid
    start program = "/etc/init.d/samba start"
    stop program  = "/etc/init.d/samba stop"
    if failed host 172.17.0.9 port 139 type TCP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 445 type TCP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 139 type TCP with timeout 10 seconds
for 6 times within 6 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    if failed host 172.17.0.9 port 445 type TCP with timeout 10 seconds
for 6 times within 6 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server
    depends on bdc.alex.lan
    depends on pdc.alex.lan

# Openldap sur le serveur distant pdc.alex.lan
# Interroge cette adresse en LDAPS.
check host pdc.alex.lan with address 172.17.0.3
    if failed port 636 type TCPSSL protocol LDAP3 with timeout 10 seconds
then alert
    group server

# Openldap sur le serveur distant bdc.alex.lan
check host bdc.alex.lan with address 172.17.0.4
    if failed port 636 type TCPSSL protocol LDAP3 with timeout 10 seconds
then alert
    group server

# nfs-server
# Avec le fichier /etc/init.d/nfs-kernel-server modifié.
check process nfs-server with pidfile /var/run/rpc.mountd.pid
    start program = "/etc/init.d/nfs-kernel-server start"
    stop  program = "/etc/init.d/nfs-kernel-server stop"
    if failed host 172.17.0.9 port 32000 type TCP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 32000 type TCP with timeout 10 seconds
for 3 times within 5 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server

# nfs-common
```

```
check process nfs-common with pidfile /var/run/rpc.statd.pid
    start program = "/etc/init.d/nfs-common start"
    stop  program = "/etc/init.d/nfs-common stop"
    if failed host 172.17.0.9 port 31000 type TCP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 31000 type TCP with timeout 10 seconds
for 3 times within 5 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server

# nfs-portmap
# Avec le fichier /etc/init.d/portmap modifié.
check process nfs-portmap with pidfile /var/run/portmap.pid
    start program = "/etc/init.d/portmap start"
    stop  program = "/etc/init.d/portmap stop"
    if failed host 172.17.0.9 port 111 type TCP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 111 type TCP with timeout 10 seconds
for 3 times within 5 cycles then exec "/usr/lib/heartbeat/hb_standby all"
    group server

# Postfix
check process postfix with pidfile /var/spool/postfix/pid/master.pid
    start program = "/etc/init.d/postfix start"
    stop  program = "/etc/init.d/postfix stop"
    if failed host 127.0.0.1 port 25 protocol SMTP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group mail
```

Copiez le répertoire /etc/monit/ssl du serveur nas1 vers fs1 et fs2.

Sur fs2.

Modifiez le fichier /etc/monit/monitrc :

```
# Fichier de configuration pour le noeud : fs2
#
## Global section
#
set logfile syslog facility log_daemon
set mailserver localhost
set eventqueue basedir /var/spool/monit slots 100
set alert superviseur@alex.lan
set alert superviseur@unfai.com
set alert superviseur@unautrefai.com
#set alert uncompte@fai only on { timeout }

set httpd port 2812
    allow 172.17.0.0/16
    allow monit:mypassword
    ssl enable
    pemfile /etc/monit/ssl/monit.pem
```

```
## Services
#
# Liste des ressources à surveiller.
#
# Charge serveur
check system localhost
    if loadavg (1min) > 30 for 8 cycles then alert
    group system

# Espace disque pour le périphérique /dev/drbd0
check device nas with path /dev/drbd0
    if space usage > 90% then alert
    if inode usage > 90% then alert
    group system

# Samba (résolution NetBIOS)
check process samba-nmbd with pidfile /var/run/samba/nmbd.pid
    start program = "/etc/init.d/samba start"
    stop program  = "/etc/init.d/samba stop"
    if failed host 172.17.0.9 port 137 type UDP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 138 type UDP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group server

# Samba (session NetBIOS et CIFS)
check process samba-smbd with pidfile /var/run/samba/smbd.pid
    start program = "/etc/init.d/samba start"
    stop program  = "/etc/init.d/samba stop"
    if failed host 172.17.0.9 port 139 type TCP with timeout 10 seconds
then restart
    if failed host 172.17.0.9 port 445 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group server
    depends on bdc.alex.lan
    depends on pdc.alex.lan

# Openldap sur le serveur distant pdc.alex.lan
check host pdc.alex.lan with address 172.17.0.3
    if failed port 636 type TCPSSL protocol LDAP3 with timeout 10 seconds
then alert
    group server

# Openldap sur le serveur distant bdc.alex.lan
check host bdc.alex.lan with address 172.17.0.4
    if failed port 636 type TCPSSL protocol LDAP3 with timeout 10 seconds
then alert
    group server

# nfs-server
check process nfs-server with pidfile /var/run/rpc.mountd.pid
    start program = "/etc/init.d/nfs-kernel-server start"
    stop  program = "/etc/init.d/nfs-kernel-server stop"
    if failed host 172.17.0.9 port 32000 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
```

```
    group server

# nfs-common
check process nfs-common with pidfile /var/run/rpc.statd.pid
    start program = "/etc/init.d/nfs-common start"
    stop  program = "/etc/init.d/nfs-common stop"
    if failed host 172.17.0.9 port 31000 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group server

# nfs-portmap
check process nfs-portmap with pidfile /var/run/portmap.pid
    start program = "/etc/init.d/portmap start"
    stop  program = "/etc/init.d/portmap stop"
    if failed host 172.17.0.9 port 111 type TCP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group server

# Postfix
check process postfix with pidfile /var/spool/postfix/pid/master.pid
    start program = "/etc/init.d/postfix start"
    stop  program = "/etc/init.d/postfix stop"
    if failed host 127.0.0.1 port 25 protocol SMTP with timeout 10 seconds
then restart
    if 6 restarts within 6 cycles then timeout
    group mail
```

Exemples de configuration Monit pour d'autres services

Nous pouvons installer Monit sur pratiquement tous nos serveurs afin de redémarrer les services présents en cas de problèmes.

Configuration partielle du fichier /etc/monit/monitrc

Exemple pour un serveur DNS/DHCP :

```
# bind9
check process bind9 with pidfile /var/run/bind/run/named.pid
    start program = "/etc/init.d/bind9 start"
    stop program  = "/etc/init.d/bind9 stop"
    if failed host 172.17.0.1 port 53 type UDP protocol DNS with timeout
10 seconds then restart
    if failed host 172.17.0.1 port 53 type TCP protocol DNS with timeout
10 seconds then restart
    if 5 restarts within 5 cycles then timeout
    group server

# dhcp3-server
check process dhcp3-server with pidfile /var/run/dhcpd.pid
    start program = "/etc/init.d/dhcp3-server start"
    stop program  = "/etc/init.d/dhcp3-server stop"
```

```
    if failed host 172.17.0.1 port 67 type UDP with timeout 10 seconds
then restart
    if 5 restarts within 5 cycles then timeout
    group server
    depends on bind9
```

Exemple pour un serveur BackupPC :

```
# backuppc
check process backuppc with pidfile /var/run/backuppc/BackupPC.pid
    start program = "/etc/init.d/backuppc start"
    stop  program = "/etc/init.d/backuppc stop"
    if 5 restarts within 5 cycles then timeout
```

Exemple pour un serveur CUPS :

```
# cups
check process cups with pidfile /var/run/cups/cupsd.pid
    start program = "/etc/init.d/cupsys start"
    stop program  = "/etc/init.d/cupsys stop"
    if failed host 172.17.0.5 port 631 type TCP with timeout 10 seconds
then restart
    if 5 restarts within 5 cycles then timeout
```

Exemple pour un serveur NTP :

```
# ntp
check process ntp with pidfile /var/run/ntpd.pid
    start program = "/etc/init.d/ntp start"
    stop  program = "/etc/init.d/ntp stop"
    if failed host 172.16.0.3 port 123 type UDP with timeout 10 seconds
then restart
    if 5 restarts within 5 cycles then timeout
```

Exemple pour un serveur Squid :

```
# squid
check process squid with pidfile /var/run/squid.pid
    start program = "/etc/init.d/squid start"
    stop program  = "/etc/init.d/squid stop"
    if children > 50 then restart
    if failed host 172.16.0.4 port 3128 type TCP with timeout 10 seconds
then restart
    if 5 restarts within 5 cycles then timeout
```

Exemple pour un serveur Courier/client NFS/LDAP :

```
# pop
```

```
check process courier-pop with pidfile /var/run/courier/pop3d.pid
    start program = "/etc/init.d/courier-pop start"
    stop  program = "/etc/init.d/courier-pop stop"
    if failed host 172.16.0.8 port 110 protocol POP with timeout 10
seconds then restart
    if 5 restarts within 5 cycles then timeout
    group mail

# pops
check process courier-pop-ssl with pidfile /var/run/courier/pop3d-ssl.pid
    start program = "/etc/init.d/courier-pop-ssl start"
    stop  program = "/etc/init.d/courier-pop-ssl stop"
    if failed host 172.16.0.8 port 995 type TCPSSL with timeout 10 seconds
then restart
    if 5 restarts within 5 cycles then timeout
    group mail

# imap
check process courier-imap with pidfile /var/run/courier/imapd.pid
    start program = "/etc/init.d/courier-imap start"
    stop  program = "/etc/init.d/courier-imap stop"
    if failed host 172.16.0.8 port 143 protocol IMAP with timeout 10
seconds then restart
    if 5 restarts within 5 cycles then timeout
    group mail

# imaps
check process courier-imap-ssl with pidfile /var/run/courier/imapd-
ssl.pid
    start program = "/etc/init.d/courier-imap-ssl start"
    stop  program = "/etc/init.d/courier-imap-ssl stop"
    if failed host 172.16.0.8 port 993 type TCPSSL with timeout 10 seconds
then restart
    if 5 restarts within 5 cycles then timeout
    group mail

# courier-authdaemon
check process courier-authdaemon with pidfile
/var/run/courier/authdaemon/pid
    start program = "/etc/init.d/courier-authdaemon start"
    stop  program = "/etc/init.d/courier-authdaemon stop"
    if failed unixsocket /var/run/courier/authdaemon/socket with timeout
10 seconds then restart
    if 5 restarts within 5 cycles then timeout
    depends on nfs-mountpoint-home
    depends on ldap2.alex.dmz
    depends on ldap1.alex.dmz
    group mail

# nfs-mountpoint-home
# Vérification des droits sur un répertoire dans ce montage NFS.
check directory nfs-mountpoint-home with path /home/superviseur
    start program = "/bin/mount /home"
    stop program  = "/bin/umount -f /home"
    if failed permission 0700 then restart
    group mail

# Openldap sur le serveur distant ldap1.alex.dmz
```

```
check host ldap1.alex.dmz with address 172.16.0.19
    if failed port 636 type TCPSSL protocol LDAP3 with timeout 10 seconds
then alert
    group mail

# Openldap sur le serveur distant ldap2.alex.dmz
check host ldap2.alex.dmz with address 1172.16.0.20
    if failed port 636 type TCPSSL protocol LDAP3 with timeout 10 seconds
then alert
    group mail
```

Exemple pour un serveur MySQL :

```
# mysql
check process mysql with pidfile /var/run/mysqld/mysqld.pid
    start program = "/etc/init.d/mysql start"
    stop program  = "/etc/init.d/mysql stop"
    if failed unixsocket /var/run/mysqld/mysqld.sock with timeout 10
seconds then restart
    if failed host 172.16.0.16 port 3306 protocol MYSQL with timeout 10
seconds then restart
    if 5 restarts within 5 cycles then timeout
```

Exemple pour un serveur/client NUT :

```
# nut-upsd
check process nut-upsd with pidfile /var/run/nut/upsd.pid
    start program = "/etc/init.d/nut start"
    stop  program = "/etc/init.d/nut stop"
    if 5 restarts within 5 cycles then timeout
    group system

# nut-upsmon
check process nut-upsmon with pidfile /var/run/nut/upsmon.pid
    start program = "/etc/init.d/nut start"
    stop  program = "/etc/init.d/nut stop"
    if 5 restarts within 5 cycles then timeout
    depends on nut-upsd
    group system
```

Sources :
http://www.linux-ha.org/DRBD
http://www.linux-ha.org/ConfiguringHeartbeat
http://www.tildeslash.com/monit/doc/

Document mis à jour :19/12/08