Cluster Haute Disponibilité/Équilibrage de charge avec des OpenLDAP multi-maître

Nous abordons ici la mise en oeuvre d'un cluster haut-disponiblité (H-A) avec équilibrage de charge sous une debian ETCH. Pour cela, nous allons nous appuyer sur un serveur d'annuaire Openldap, de Linux Virtual Server et de Heartbeat.

CONFIGURATION DE DEUX OPENLDAP EN MULTIMAITRE

Cette partie a pour but d'expliquer comment mettre en oeuvre la fonctionnalité de replication multi-maître qu'offre la version 2.4.8 de OpenLdap.

Elle sera certainement à adapter ou améliorer selon les besoins et les prochaines versions. La méthode de réplication qui est utilisée ici est le MIRRORMODE qui est relativement facile à mettre en oeuvre. Il existe aussi la méthode “N-Way Multi-Master replication” non testé pour le moment, pour plus d'information http://www.openldap.org/doc/admin24/replication.html

---

Attention, cette partie n'explique pas toute la procédure détaillée d'installation ou d'exploitation d'un serveur LDAP. Pour plus d'informations, ici

---

L'installation décrite, sera faite depuis les sources tar.gz. Car la version actuelle n'est pas encore disponible sous Debian Etch, à l'heure où j'écris cette documentation. Mais elle ne serait tardé. La version testée est la 2.4.8, rendez-vous sur http://www.openldap.org si une nouvelle version n'est disponible.

Ce qui va suivre juste après, devra être effectué sur les 2 machines de préférence identiques Nous allons les nommer N1 et N2 et comme adresse IP respectivement @N1 et @N2.

Par exemple @N1=192.168.0.130 et @N2=192.168.0.131 Pour que la réplication fonctionne correctement N1 et N2 devront être synchronisé avec un serveur NTP. Par exemple :

```
N1:# ntpdate pool.ntp.org
N2:# ntpdate pool.ntp.org

N1:# wget ftp://sunsite.cnlab-switch.ch/mirror/OpenLDAP/openldap-release/openldap-
2.4.8.tgz
N1:# tar xzf openldap-2.4.8.tgz
N1:# cd openldap-2.4.8/
```

Compilation et Installation

Pré-requis pour la compilation

```
N1:# apt-get install gcc ldap-utils db4.2-util ...
```

```
N1:# ./configure
N1:# make depend
N1:# make
N1:# make test
N1:# make install
N1:# cd /usr/local/
```

Configuration du serveur

On va choisir comme DN principal : dc=gezen,dc=fr Editez le fichier etc/openldap/slapd.conf

```
include /usr/local/etc/openldap/schema/core.schema
include /usr/local/etc/openldap/schema/cosine.schema
include /usr/local/etc/openldap/schema/nis.schema
include /usr/local/etc/openldap/schema/inetorgperson.schema

pidfile  /var/run/slapd/slapd.pid
argsfile /var/run/slapd/slapd.args

loglevel    0
#128 1 4 16 32

allow bind_v2

modulepath  /usr/lib/ldap
moduleload  back_bdb
moduleload  syncprov

sizelimit unlimited
tool-threads 1

access to attrs=userPassword
  by dn="uid=mirrormode,dc=gezen,dc=fr" read
  by anonymous auth
  by * none

access to dn.base="" by * read
access to dn.base="cn=Subschema" by * read

access to *
  by dn="uid=mirrormode,dc=gezen,dc=fr" read
  by * read

defaultsearchbase dc=gezen,dc=fr

sasl-host localhost
sasl-secprops none
```

```
########################
# Database definition
########################
database  bdb
suffix    "dc=gezen,dc=fr"
rootdn    "cn=Manager,dc=gezen,dc=fr"
rootpw    secret

directory   "/var/lib/ldap/gezen.fr"
checkpoint  512 30

dbconfig set_cachesize 0 128000000 0
dbconfig set_lk_max_objects 1500
dbconfig set_lk_max_locks 1500
dbconfig set_lk_max_lockers 1500

index    objectClass,entryCSN,entryUUID eq
lastmod  on

overlay syncprov
syncprov-checkpoint 100 10
syncprov-sessionlog 100
```

Vous remarquez dejà qu'il y a des ACL de positionner. Elles nous seront utiles plus tard pour la réplication.

Préparation et démarrage du serveur

```
N1:# mkdir -p /usr/local/openldap/var/openldap-data
N1:# /usr/local/libexec/slapd -f /usr/local/etc/openldap/slapd.conf
```

Alimentation du serveur

```
N1:# vi /tmp/fichier.ldif

dn: dc=gezen,dc=fr
dc: gezen
objectClass: dcObject
objectClass: organization
o: GEZEN
```

Création de la racine de l'annuaire.

N1:# ldapadd -f /tmp/fichier.ldif -x -D "cn=Manager,dc=gezen,dc=fr" -w secret

Création de l'utilisateur mirrormode

Création d'un mot de passe “secret”

```
# slappasswd
New password: secret
Re-enter new password: secret
{SSHA}Obsd5PgAoUoakwn+xoNEUmLT3zfmRQ3G
# echo "{SSHA}Obsd5PgAoUoakwn+xoNEUmLT3zfmRQ3G" | openssl base64 -e
e1NTSEF9T2JzZDVQZ0FvVW9ha3duK3hvTkVVbUxUM3pmbVJRM0cK
```

Créer le fichier /tmp/mirrormode.ldif

```
N1:# vi /tmp/mirrormode.ldif

dn: uid=mirrormode,dc=gezen,dc=fr
uid: mirrormode
objectClass: simpleSecurityObject
objectClass: account
userPassword:: e1NTSEF9T2JzZDVQZ0FvVW9ha3duK3hvTkVVbUxUM3pmbVJRM0cK

N1:# ldapadd -f /tmp/mirrormode.ldif -x -D "cn=Manager,dc=gezen,dc=fr" -w secret
```

Vérifier les données

```
N1:# ldapsearch -h localhost -x -b "dc=gezen,dc=fr"
```

Arrêter ensuite le serveur OpenLdap.

Configuration de la réplication

Sur N1, rajouter les lignes à la fin du fichier

```
syncrepl rid=001
    provider=ldap://192.168.1.123
    binddn="uid=mirrormode,dc=gezen,dc=fr"
    credentials=mirrormode
    bindmethod=simple
    searchbase="dc=gezen,dc=fr"
    schemachecking=on
    type=refreshAndPersist
    retry="60 +"

syncrepl rid=002
    provider=ldap://192.168.0.242
    binddn="uid=mirrormode,dc=gezen,dc=fr"
    credentials=mirrormode
    bindmethod=simple
    searchbase="dc=gezen,dc=fr"
    schemachecking=on
    type=refreshAndPersist
    retry="60 +"
```

```
    retry="60 +"

mirrormode on
serverID    1
```

Et pour N2

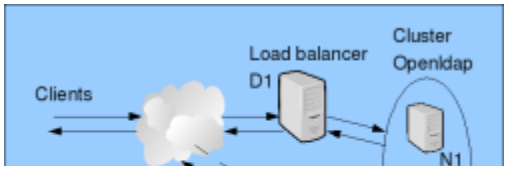
```
syncrepl rid=001
    provider=ldap://192.168.1.123
    binddn="uid=mirrormode,dc=gezen,dc=fr"
    credentials=mirrormode
    bindmethod=simple
    searchbase="dc=gezen,dc=fr"
    schemachecking=on
    type=refreshAndPersist
    retry="60 +"

syncrepl rid=002
    provider=ldap://192.168.0.242
    binddn="uid=mirrormode,dc=gezen,dc=fr"
    credentials=mirrormode
    bindmethod=simple
    searchbase="dc=gezen,dc=fr"
    schemachecking=on
    type=refreshAndPersist
    retry="60 +"

mirrormode on
serverID    2
```

Démarrer le serveur Openldap sur N1 et sur N2. Effectuez des ldapadd sur N1 et N2 et vérifiez que les données soumises ont bien été répliquées.

UN PEU DE THÉORIE

Vous remarquez dejà qu'il y a des ACL de positionner. Elles nous seront utiles plus tard pour la réplication.

Notre cluster est un groupe de machines proposant un service d'annuaire de manière transparente pour les clients. Sa particularité est que les clients ne savent pas qu'ils s'adressent à un cluster. Nous aurons besoin d'un serveur que l'on va nommer "D1" qui sera le load-balancer appelé aussi sous le nom de "director". Son rôle sera de dispatcher les requêtes clientes vers un noeud ; un des nos deux openldap. Si le load-balancer tombe en panne, le cluster est indisponible, il faut penser à redonder ce serveur. Voici un schéma qui illustre dans l'infrastructure cible:



D2 notre deuxième load-balancer reste en stanby tant que D1 fonctionne. Lors D1 rencontrera un disfonctionnement ça sera D2 qui prendra le relais.

Nous allons utiliser pour cela 5 machines :

- le directeur + un deuxième en standby
- deux noeuds N1 et N2 proposant un serveur Openldap
- un poste client pour effectuer des tests

N1 et N2 sont deux serveurs openldap que nous avons configuré en multi-maître. Au niveau des adressages :

D1 : notre directeur Maitre

- IP : @D1 (par ex 192.168.0.121)
- VIP : @VIP (par ex 192.168.0.195)

D2 : notre directeur Esclave

- IP : @D2 (par ex 192.168.0.122)

N1 : premier noeud openldap

- IP : @N1 (par ex 192.168.0.130)

N2 : deuxième noeud openldap

- IP : @N2 (par ex 192.168.0.131)

D1 et D2 sont deux serveurs qui ont été installés avec la même distribution Debian Etch. Les serveurs doivent être synchronisés via NTP.

CONFIGURATION DU LOAD-BALANCER

Activation des modules.

Avant tout chose, il est préférable de vérifier que les modules IPVS soient activer dans le noyau linux. Si ce n'est pas le cas, voici les commandes nécessaires.

```
D1:# modprobe ip_vs
D1:# modprobe ip_vs_dh
D1:# modprobe ip_vs_ftp
```

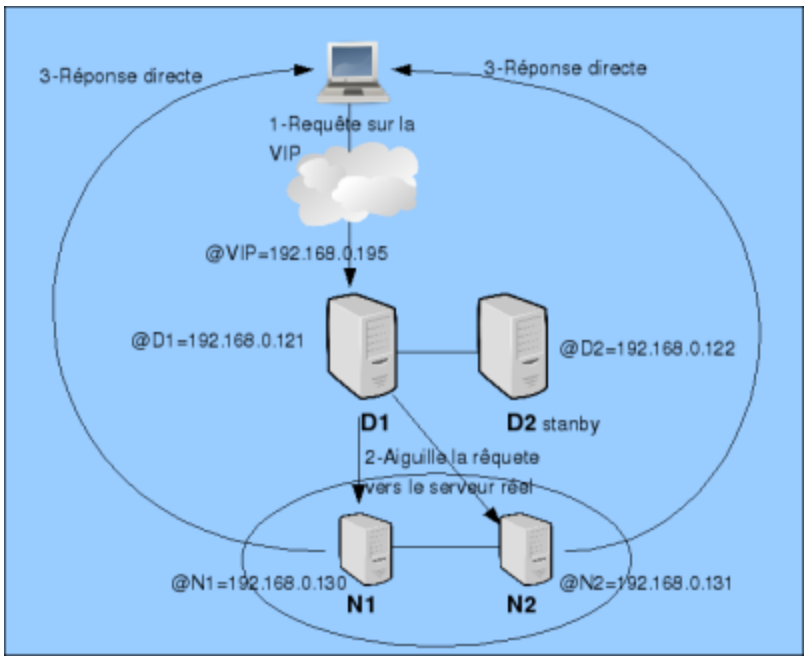
```
D1:# modprobe ip_vs_rrp
D1:# ...
```

Ajout d'un alias reseau pour l'adresse IP virtuelle.

Notre cluster a besoin d'une adresse IP virtuelle. Si dans votre infrastructure vous utiliser un serveur DHCP, vérifier que l'adresse @VIP est bien disponible et qu'elle ne sera pas attribuée à une machine du réseau. Ce qui peut poser quelques problème par la suite.

```
D1:# ip addr add 192.168.0.195 dev eth0
D1:# apt-get install ipvsadm
```

Ici notre load-balancer est sur le même sous-réseau que les serveurs réels, nous allons utilisé comme méthode le routage directe (DR ou Direct Return). Cette solution s'avère performante, car elle décharge le load-balancer de la gestion des réponses.



Configuration de ipvsadm

On met en place un équilibrage de charge pour les paquets à la destination du port 389 sur la @VIP. Nous devons choisir un algorithme de répartition. IPVS en propose 10 dont les principaux sont :

- rr (Round-Robin) : les requêtes sont transmises à tour de rôle sur chaque nœud du cluster.
- wrr (Weighted Round Robin) : identique au précédent, mais peut gérer une pondération associée à chaque nœud, l'objectif étant de charger les machines proportionnellement à leur puissance.
- lc (Least Connection) : IPVS transmet la requête au serveur sur lequel il y a le moins de connexions actives.
- wlc (Weighted Least-Connection) : sa variante pondérée…
- lblc, lblcr, dh et sh : déterminent le nœud en fonction de l'adresse IP source ou destination.

Ici l'algorithme de répartition choisi est rr (Round-Robin). Il nous permettra, dans un premier temps, de bien vérifier que tous les serveurs sont actifs. La commande de création d'un nouveau service à équilibrer est :

```
D1:# ipvsadm -A -t 192.168.0.195:389 -s rr
D1:# ipvsadm -a -t 192.168.0.195:389 -r 192.168.1.130:389
D1:# ipvsadm -a -t 192.168.0.195:389 -r 192.168.1.131:389
```

Les règles gérées par IPVS deviennent alors :

```
D1:# ipvsadm -L
```

CONFIGURATION DES NOEUDS N1 et N2

Configuration réseau

Comment le load-balancer va-t-til effectuer le relayage ? Dès réception du paquet à la destination de la VIP, le load-balancer va modifier l'adresse MAC destination en y plaçant celle du serveur réel. Le serveur réel va donc recevoir le paquet, mais encore faut-il qu'il l'accepte ! Ce n'est possible que s'il dispose aussi de l'adresse VIP définie sur une interface. Pour pas que les serveurs réels ne rentrent pas en conflit avec l'adresse $VIP du load-balancer, nous allons placer la @VIP comme adresse secondaire sur l'interface loopback.

```
N1:# ip addr add 192.168.0.195 dev lo
N2:# ip addr add 192.168.0.195 dev lo
```

Pour empêcher les serveurs réels de répondre aux requêtes ARP concernant la @VIP, nous allons modifier les paramètres arp_announce et arp_ignore de la pile TCP/IP.

```
N1:# echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
N1:# echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce
N2:# echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
N2:# echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce
```

TESTS DE FONCTIONNEMENT DE IPVS

Depuis un poste client, lancer la commande ldapsearch sur l'adresse IP virtuelle Sur le load-balancer, vous pouvez lancer la commande

```
D1:# ipvsadm -L
```

HEARBEAT

Avant de commencer cette partie, réinitialisons la configuration de ipvsadm sur le load-balancer

```
D1:# ipvsadm -C
```

L'installation et la configuration de Hearbeat se fera depuis les sources de la 2.1.3.

Télécharger la dernière version

```
D1:# cd /tmp
D1:# wget http://www.ultramonkey.org/download/heartbeat/2.1.3/heartbeat-2.1.3.tar.gz
```

Prérequis à la compilation de heartbeat

```
D1:# apt-get install bison debhelper flex iproute iputils-ping libbz2-dev libcurl3-
openssl-dev
libglib2.0-dev libgnutls-dev libltdl3-dev libncurses5-dev libnet1-dev libopenhpi-dev
libpam0g-dev libsensors-dev libsnmp9-dev libtool libxml2-dev lynx net-tools
openssh-client perl psmisc python python-central python-dev swig uuid-dev
zlib1g-dev libmailtools-perl libwww-perl libnet-ldap-perl
```

Compilation

```
D1:# tar xjf heartbeat-2.1.3.tar.gz
D1:# cd heartbeat-2.1.3
D1:# ./configure
D1:# make
D1:# make install
```

Voici quelques commandes à lancer après l'installation : il faut créer l'utilisateur et le groupe associé au démon heartbeat.

```
D1:# adduser hacluster
D1:# addgroup haclient
D1:# adduser hacluster haclient
```

Configuration de ldirectord

Tel que nous l'avons mis en oeuvre, notre load-balancer fonctionne tant que tous les noeuds sont fonctionnels ! Que se passe-t-il si le noeud N2 est arrêté ou même si tout simplement le service slapd sur N2 est interrompu ? Le load-balancer va rediriger une requête sur deux vers N2 et retournera donc, pour ces requêtes, une erreur au client.

Pour résoudre ce problème, nous allons mettre en oeuvre sur le load-balancer un processus spécialisé, nommé ldirectord et que nous appellerons le " directeur ", en français. Écrit en Perl, ce programme a pour rôle la surveillance applicative des nœuds et modifie, en temps réel, les règles d'équilibrage de charge, à l'aide de la commande ipvsadm.

d'équilibrage de charge, à l'aide de la commande ipvsadm.

Heartbeat stocke ses fichiers de configuration dans /usr/local/etc/ha.d

```
D1:# cd /usr/local
D1:/usr/local# cat etc/ha.d/ldirectord.cf

checktimeout=10
checkinterval=2
autoreload=no
logfile="/var/log/ldirectord.log"
logfile="local0"
quiescent=yes

## Virtual Service for LDAP
virtual=@VIP:389
        real=@N1:389 gate
        real=@N2:389 gate
        service=ldap
        scheduler=rr
        #persistent=600
        checkport=389
        protocol=tcp
        checktype=negotiate
```

N'oublier pas de remplacer les valeurs de @VIP @N1 et @N2

Pour démarrer le service, lancer :

```
D1:# ldirectord -d etc/ha.d/ldirectord.cf start
```

Cette commande vous permettra de voir les messages de debug. Vous pouvez aussi utiliser cette commande :

```
D1:/usr/local# /etc/init.d/ldirectord start
```

Vérifier que le service ldirectord a bien lancer ipvsadm.

```
D1:# ipvsadm -L
```

TEST DE LDIRECTORD

- Effectuer des tests ldapsearch avec un poste client pour voir que tout fonctionne
- Arrêter par exemple du N1, le service slapd. On peut vérifier ce qui se passe via

```
D1:# ipvsadm -L
```

- Lancer de nouveau des commandes ldapsearch.

CONFIGURATION de HEARTBEAT

Voici la dernière étape dans la mise en place de notre cluster, nous allons expliquer comment mettre en place de la haute dispo sur notre load-balancer. Cela semble logique, il nous faut installer Heartbeat sur D2 et recopier le fichier “ldirectord.cf” vu au §7

Réiniatialiser la config réseau

```
D1:# ip addr del 192.168.0.195 dev eth0
```

Les fichiers de configuration

La procédure ci-dessous devra être réalisée sur D1 et D2.
Heartbeat utilise trois fichiers de configuration:

- ha.cf qui définit le cluster et les méthodes de surveillance des nœuds
- haresources qui liste les ressources à gérer (migrer)
- authkeys qui indique la méthode d’authentification/chiffrement à utiliser pour la communication entre D1 et D2

Fichier ha.cf

Ce fichier décrit les paramètres de la ” mécanique ” heartbeat : les valeurs des différents timeouts, les méthodes de surveillance de l’état des nœuds, les informations de login, la liste des nœuds du cluster.

```
D1:# cat /usr/local/etc/ha.d/ha.cf

# mcast <interface> <groupe multicast> <udp port> <ttl> 0
# Le groupe multicast est une adresse que vous choisissez comprise entre 224.0.0.0
et
# 239.255.255.255
# dans laquelle D1 et D2 vont communiquer. Prenez garde à ne pas avoir d'autres
# systèmes utilisant ce groupe multicast.
# Le ttl indique le nombre de saut de réseau que le paquet peut effectuer (de 1 à
255).
# D1 et D2 sont le même réseau et une valeur de 1 est parfaite.
mcast eth0 225.0.0.7 694 1 0

# logfacility undique que l'on utilise la facilité syslog en plus.
logfacility  local0

# logfile indique le log d'activité à utiliser.
logfile /var/log/ha.log

# udpport indique le fichier de détection à utili
```

```
# debugfile indique le fichier de debogage a utiliser.
debugfile /var/log/ha-debug.log

# keepalive indique le délai entre deux battements de pouls. Ce délai est donné par
# défaut en seconde;
# pour utiliser les millisecondes, on ajoute le suffixe ms.
keepalive  2
# deadtime indique le temps nécessaire avant de déclarer un noeud comme mort
deadtime  10

# warntime indique le temps avant d'avertir dans le log
warntime  6

# initdead : valeur utilisée pour le démarrage (au moins 2 fois le deadtime)
initdead  60

# auto_failback indique le comportement à adopter si le node maître revient dans le
# cluster.
# Si on met la valeur on, lorsque le node maître revient dans le cluster, tout est
transféré
# sur ce dernier.
# Si on met la valeur off, les services continuent à tourner sur l'esclave même
lorsque le
# maître revient dans le cluster.
# Il est préférable de mettre comme valeur à off pour pouvoir faire un retour à la
normale
# manuellement lorsque la charge de production est moins importante.
# De plus, si le D1 souffre d'un grave problème, on pourrait avoir une boucle de
# démarrage/extinction qui entraînerait un relais des services de manière
incessante.
auto_failback  off

# le nom des noeuds DOIT être celui retourné par `uname -n`
node    D1
node    D2
```

Fichier authkeys

Ce fichier décrit le mécanisme d'authentification utilisé par les nœuds du cluster. Plusieurs méthodes sont disponibles :

- le calcul d'un crc, idéal pour ses performances en cas de liaison série
- le chiffrement avec MD5 ou SHA-1, à l'aide d'un secret partagé.

Le fichier indique plusieurs méthodes, et, parmi celles-ci, la méthode à utiliser :

```
D1:# cat /usr/local/etc/ha.d/authkeys

auth 2
```

```
auth 2
1 md5 "secret"
2 crc
3 sha1 "autre secret"
```

Fichier haresources

Notre fichier haresources contient une seule ligne :

```
D1:# cat /usr/local/etc/ha.d/haresources

D1 IPaddr2::@VIP ldirectord
```

Cette ligne indique qu'au démarrage la machine D1 (le maître) doit acquérir les ressources IPaddr2 et ldirectord. Ces “ressources” sont en fait des scripts situés sous $HA_DIR/resource.d et ressemblent à des scripts d’initialisation “System V”, c’est-à-dire qu’ils acceptent des paramètres, ainsi que les arguments start, stop, status, restart. Certains de ces scripts peuvent en outre être conformes à la norme OCF (Open Cluster Framework) qui définit les paramètres en entrée, les comportements attendus (codes retours) et des règles de nommage de variables pour échanger des informations [OCF].

Dans notre exemple, le script IPaddr2 reçoit un paramètre supplémentaire : la @VIP. Ce script fait partie de la distribution de heartbeat. Il gère une adresse virtuelle passée en paramètre. Il existe aussi un script nommé IPaddr qui fait la même chose, à ceci près que, dans notre cas, IPaddr2 supporte la gestion des interfaces cachée.

Mise en route

Sur D1 et N2, on surveille, dans un terminal, les logs de heartbeat par la commande

```
D?:# tail -f /var/log/ha-debug.log
```

Puis, on lance heartbeat sur les deux machines :

```
D1:# /etc/init.d/heartbeat start
```

Quand le message “local resource transition completed” apparaît, le D1 a pris la @VIP et le directeur est en place.

```
D1:# ip addr show
D1:# ipvsadm -L
```

Vous pouvez accéder au serveur slapd par la @VIP.

Vérifiez que D2 est resté en standby.

```
D2:# ip addr show
D2:# ipvsadm -L
```

Simulons une panne sur D1 et voyons comment se porte D2

```
D1:# /etc/init.d/heartbeat stop
```

Sur D2, vous devriez voir apparaître dans les logs

```
WARN: node debian1: is dead
[..]
Resources being acquired from debian1.
[..]
local resource transition completed.
```

Vérifiez que D2 a bien pris le relais sur D1 et que sur un poste client le serveur ldap répond toujours

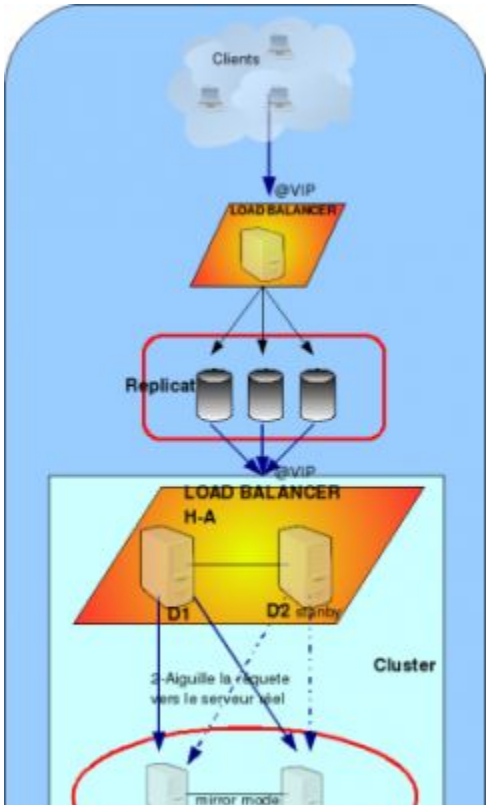
```
D2:# ip addr show
D2:# ipvsadm -L
```

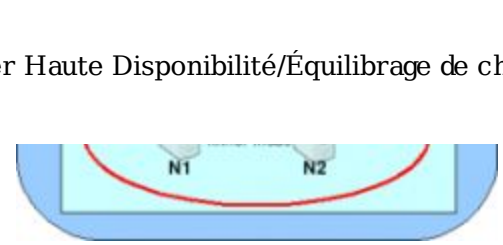
Rédémarrons Hearbeat sur D1, D2 a détecté la présence de D1 et reste toujours actif. Ce que confirme la commande ipvsadm sur D2

Pour que D1 reprenne le relais il va falloir le faire manuellement en arrêtant D2 et redémarrant D1. Action de maintenance à faire la nuit par exemple :-p

Pour aller plus loin

On pourrait imaginer l'infrastructure suivante si les besoins seraient énormes pour des requêtes clientes, en utilisant des simples réplicats avec équilibrage de charge.




N1
N2