Serveur Linux : Haproxy

Mise en place d'un service haproxy sous Linux

Bouron Dimitri

09/11/2013

Ce document sert de démonstration concise pour l'installation, la configuration, d'un serveur haproxy sous Linux.

Table des matières

I. Machine virtuelle par défaut

Ce document sert de base pour la mise en place d'un service haproxy, mais aussi la configuration de base faite sur la machine virtuelle complète du serveur.

A. Identifiants de connexion

Les identifiants de connexions de la machine virtuelle par défaut sont les suivants :

Login : root / Password : P@ssword

Login : dimitri / Password : P@ssword

B. Hostname

L'hostname de la machine virtuelle par défaut est : serveur-HAPROXY-base.

C. Interfaces

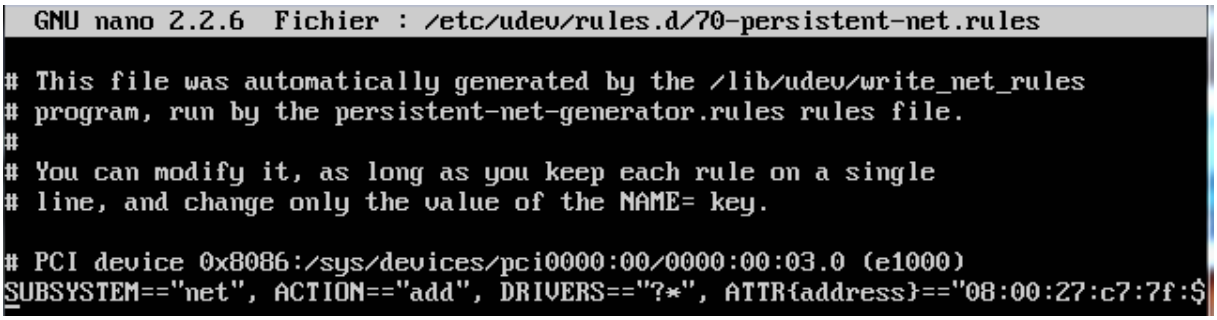
La configuration réseau est la suivante :

Adresse IP statique : 192.168.1.170 /24 (pour l'actif)
Réseau : 192.168.1.0
Adresse de diffusion : 192.168.1.255
Passerelle : 192.168.1.254

Si on crée une nouvelle machine virtuelle à partir de celle-ci, un problème d'interface surviendra. C'est-à-dire que l'interface réseau va s'ajouter et devenir eth1 (si vous utilisez la machine par défaut, dans un autre cas il se peut que vous ayez plusieurs autres interfaces réseau). Le problème c'est que l'on rend la machine accessible uniquement par eth0, donc problème de connexion réseau. Pour y remédier deux solutions, soit on fait ça proprement en supprimant dans un premier temps l'interface eth0 actuelle et on change le nom de l'interface eth1 en eth0, soit on le fait salement en remplaçant la valeur eth0 par eth1 dans le fichier de conf. Voir ci-dessous les deux solutions :

1) Première solution : /etc/udev/rules.d/70-persistent-net.rules

Bon, pour le faire proprement, on va donc faire ce qui est expliqué plus haut, c'est-à-dire supprimer notre eth0 actuelle pour la remplacer par notre eth1. Pour ce faire on va aller dans le fichier avec la commande **# nano /etc/udev/rules.d/70-persistent-net.rules**. Voir ci-dessous le résultat :



Voilà ce que nous avons actuellement, naturellement je ne peux pas montrer les deux interfaces (eth0 et eth1) vu que je n'ai pas déplacé ma machine. Mais ce qui arrivera dans ce cas c'est que les deux dernières lignes visibles sur l'image ci-dessus seront en double exemplaire, ou plutôt 2 autres lignes semblables (les valeurs spécifiques seront différentes) seront visibles. Pour voir le nom de

l'interface il faut aller à la fin de la ligne (qui n'apparaît pas entièrement) qui commence par « SUBSYSTEM=="net" ». Voir ci-dessous :

```
# PCI device 0x8086:/sys/devices/pci0000:00/0000:00:03.0 (e1000)
$c7:7f:b5", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"_
```

Comme on peut le voir, pour cette ligne on peut voir à la fin le critère « NAME="eth0" » qui correspond bien au nom de l'interface réseau, si j'avais une seconde ligne ce serait : « NAME="eth1" ». Donc ce qu'il nous reste à faire est tout simplement de supprimer les deux lignes correspondants à eth0, pour aller plus vite il nous suffit de se mettre en début de ligne et de faire la combinaison de touche **[CTRL + K]** (qui correspond à un couper, cela coupera la ligne entièrement), il faut donc faire cette manipulation pour les deux lignes. Une fois qu'il ne reste que deux lignes (celles pour notre eth1), il suffit juste de modifier la valeur de « NAME="eth1" » en remplaçant eth1 par eth0. Une fois que cela est fait, on utilise la commande **# /etc/init.d/networking restart** pour prendre en compte les modifications.

Pour vérifier si oui ou non la manipulation a bien fonctionné, il suffit d'essayer de ping (même une machine locale). Si ça marche tant mieux, sinon il faut essayer un **# reboot** de la machine. Et si ça ne marche toujours pas, il faut essayer de faire en partie la seconde solution (qui sera sale mais fonctionnera peut-être mieux au final).

2) Deuxième solution : /etc/network/interfaces

Comme dit précédemment, cette solution est « sale ». La raison est simple, cela s'explique par la manière de résoudre le problème. Explication : Précédemment, la solution était de supprimer l'actuelle interface eth0 et de la remplacer par l'interface eth1 pour obtenir le même résultat que si nous n'avions pas pris une machine virtuelle déjà existante. Pour notre seconde solution, nous n'allons pas remplacer l'interface eth1 par eth0 mais définir dans notre fichier de configuration réseau que l'interface réseau à utiliser est l'interface eth1 à la place de l'interface eth0 (soit on va remplacer eth0 par eth1). Le fait d'agir ainsi fait que nos interfaces réseaux vont continuer à s'empiler au fur et à mesure qu'on les déplacera ou les réutilisera.

Voilà comment il faut procéder. On utilise la commande **# nano /etc/network/interfaces**, puis on va remplacer la valeur eth0 par eth1, une fois fait et sauvegardé on utilisera la commande **# /etc/init.d/networking restart**. Voir ci-dessous :

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.150
        netmask 255.255.255.0

iface eth0 inet dhcp
```

On remplace eth0 par eth1 pour obtenir :

```
# The primary network interface
auto eth1
iface eth1 inet static
        address 192.168.1.150
        netmask 255.255.255.0

iface eth1_inet dhcp
```

Une fois fait, pour vérifier le bon fonctionnement on peut toujours essayer de ping et la commande **# ifconfig** (je précise que les captures d'écran sont faites à partir d'un serveur web et non du serveur haproxy, donc c'est normal si les valeurs ne correspondent pas).

D. Accès ssh

La configuration d'openssh-server est faite de telle sorte que :

- Le port d'écoute est 12543.
- Le PermitRootLogin n'est pas autorisé (valeur à no). On ne peut pas se connecter directement par le biais de l'utilisateur root.

E. Configuration haproxy

La configuration de haproxy est faite de telle sorte que le fichier de configuration /etc/haproxy/haproxy.cfg soit créé et rempli mais il ne manque plus qu'à ajouter les valeurs pour les serveurs web de notre cluster. Trois lignes sont déjà créées, il suffit de reprendre sur ce modèle.

F. Configuration pour heartbeat

La configuration de heartbeart est faite de sorte que :

Le serveur passif se nomme serveur-HAPROXY-passif-base, son adresse IP est 192.168.1.171 /24.

L'adresse IP virtuelle écoutée par les 2 serveurs est 192.168.1.160.

II. Installation et configuration de base

Une fois l'installation de Ubuntu 12.04 faite, il faut se connecter avec l'identifiant de connexion crée lors de l'installation de l'OS. Dans notre exemple nous avons créé l'utilisateur « dimitri:P@ssword ». Une fois que nous avons réussi à nous connecter nous allons devoir activer l'utilisateur root pour les configurations à venir en utilisant la commande **$ sudo passwd root**. Le mot de passe que nous donnerons à root sera : P@ssword (faites ce que vous voulez). Puis on se connectera avec les identifiants de ce nouvel utilisateur avec la commande **$ su root**. Voir ci-dessous :

```
mangetsu@serveur-WEB-base:~$ sudo passwd root
[sudo] password for mangetsu:
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd: password updated successfully
mangetsu@serveur-WEB-base:~$ su root
Mot de passe :
root@serveur-WEB-base:/home/mangetsu#
```

(L'image ci-dessus a été prise à partir d'un serveur web, donc il est normal que le nom d'utilisateur et le nom d'hôte ne correspondent pas à notre situation).

À partir de maintenant, nous considérerons que toutes les commandes qui suivront sont faites à partir du compte root, même si l'utilisateur simple peut faire la modification pour des raisons de facilités.

A. Apt-get update

On met rapidement la base à jour avec la commande **# apt-get update**.

B. Hostname

L'hostname correspond au nom d'hôte de la machine, que l'on renseigne une première fois lors de l'installation de l'OS. La commande qui permet de visualiser le nom d'hôte est **# hostname**. Pour modifier le nom d'hôte de la machine, on utilise la commande **# nano /etc/hostname**, puis nous modifierons la première ligne en la remplaçant par le nom d'hôte souhaité (attention, il faut éviter de mettre un ou plusieurs underscore, il se peut que cela ne fonctionne pas correctement). Une fois la ligne modifiée et le fichier hostname sauvegardé, nous allons devoir faire relire le fichier hostname par le serveur, le plus bourrin serait d'utiliser **# reboot** pour redémarrer le serveur, mais une solution simple est de forcer la relecture du fichier hostname en utilisant la commande suivante : **# /etc/init.d/hostname restart**. Pour vérifier si la modification a bien été lu après le redémarrage, il suffit de réutiliser la commande **# hostname**. Voir ci-dessous :

```
 GNU nano 2.2.6          Fichier : /etc/hostname

serveur-HAPROXY-base_
```

```
root@ubuntu-haproxy:~# /etc/init.d/hostname restart
Rather than invoking init scripts through /etc/init.d, use the service(8)
utility, e.g. service hostname restart

Since the script you are attempting to invoke has been converted to an
Upstart job, you may also use the stop(8) and then start(8) utilities,
e.g. stop hostname ; start hostname. The restart(8) utility is also available.
hostname stop/waiting
root@ubuntu-haproxy:~# _
```

```
root@ubuntu-haproxy:~# hostname
serveur-HAPROXY-base
```

Attention, on peut remarquer que la modification est bien prise en compte actuellement en utilisant la commande pour afficher l'hostname mais la valeur affichée dans « **root@ubuntu-haproxy:~#** » est toujours la même, elle n'a pas été modifié malgré le changement prit en compte. Toutefois, il suffit de taper la commande **# exit** pour revenir à l'écran d'identification et nous pouvons constater que la modification apparaît bien désormais. Voir ci-dessous :

```
Ubuntu 12.04 LTS serveur-HAPROXY-base tty1

Hint: Num Lock on

serveur-HAPROXY-base login: _
```

C. Interfaces

Nous utilisons un serveur, haproxy pour être exacte, ce qui implique de lui attribuer une adresse IP fixe, nous allons donc attribuer l'adresse fixe directement sur notre serveur haproxy. Pour cela nous allons utiliser la commande **# nano /etc/network/interfaces** et nous modifierons le fichier en remplaçant la ligne « iface eth0 inet dhcp » comme suit :

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

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.170
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
        dns-nameservers 192.168.1.1_
```

Les adresses IP sont à modifier selon la situation naturellement.

Renseigner toutes les valeurs peut rapidement devenir fastidieux. Du coup on peut utiliser cette méthode tout en conservant l'utilisation du DHCP du réseau (encore faut-il en avoir un). Comme je suis feignant je n'ai pas envie d'avoir à renseigner toutes les valeurs, qui risque d'être faux si je me trompe alors je vais utiliser la seconde méthode qui est de rendre statique les valeurs souhaitées et

de faire appel au DHCP pour renseigner le reste automatiquement et donc s'adapter au mieux à ma situation. Voir ci-dessous la configuration à faire :

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

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.170
        netmask 255.255.255.0

iface eth0 inet dhcp_
```

Se contenter de ça ne nous suffira pas, il nous faut faire comme pour l'hostname forcer la reconfiguration réseau avec la commande **# /etc/init.d/networking restart** et utiliser la commande **# ifconfig** pour vérifier que le changement de valeur est pris en compte puis on peut toujours vérifier l'accès à internet avec un ping. Voir ci-dessous :

```
root@serveur-HAPROXY-base:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:a0:b1:e2
          inet adr:192.168.1.170  Bcast:192.168.1.255  Masque:255.255.255.0
          adr inet6: fe80::a00:27ff:fea0:b1e2/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Packets reçus:2687 erreurs:0 :0 overruns:0 frame:0
          TX packets:414 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:840449 (840.4 KB) Octets transmis:49673 (49.6 KB)

lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          Packets reçus:88 erreurs:0 :0 overruns:0 frame:0
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          Octets reçus:6876 (6.8 KB) Octets transmis:6876 (6.8 KB)
```

Si on utilise une machine virtuelle qui existait déjà mais que l'on a déplacé d'un poste à l'autre, ou même lorsque l'on en fait une copie directement depuis VirtualBox (ne pas oublier de réinitialiser l'adresse MAC en cochant l'option lors de la copie de la machine). Il se peut qu'il y ait des problèmes de réseau. Certainement qu'il y a une interface ethx de trop et la bonne n'est pas prise en compte. Pour remédier à ce problème il y a deux solutions possibles, une propre et une sale. Ces 2 solutions sont expliquées plus haut dans le document.

D. Mise en place de openssh-server

1) Installation du paquet

Maintenant que notre serveur de base est prêt nous allons mettre en place les différents paquets dont nous avons besoin pour mettre en place le service haproxy et avant ça encore activer l'accès ssh. Installons notre paquet openssh-server avec la commande **# apt-get install openssh-server**.

```
root@serveur-HAPROXY-base:~# apt-get install openssh-server
```

On peut désormais vérifier le fonctionnement du ssh avec la commande **# service ssh status** et en essayant de se connecter avec putty au serveur.

```
root@serveur-HAPROXY-base:~# service ssh status
ssh start/running, process 2850
```

S'il on voit start/running, on peut essayer putty. Sinon il faut essayer de le lancer avec la commande **# /etc/init.d/ssh start**.

2) Configuration de openssh-server

Nous allons maintenant configurer openssh-server correctement, ou plutôt sécuriser l'accès ssh un minimum pour l'instant en modifiant le port d'écoute (ce qui implique le port d'écoute pour le sftp car ceux sont les mêmes) ainsi que la restriction de l'utilisation de root pour se connecter en ssh (et sftp). Pour cela on utilise la commande **# nano /etc/ssh/sshd_config**, puis on va remplacer le numéro de port (22) à la ligne 5 par le port souhaité mais en faisant attention à utiliser un port correcte et non utilisé. Il faut donc un port inférieur à 65535 et éviter les ports utilisés (pour en savoir plus sur les ports utilisés le plus souvent par les autres services et donc les éviter il faut suivre ce lien). Ce n'est pas tout, on va modifier la valeur du PermitRootLogin à la ligne 27 en remplaçant la valeur actuelle (yes) par la valeur no. Voir ci-dessous :

```
# What ports, IPs and protocols we listen for
Port 12543_
```

```
# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
```

Une fois fait et sauvegardé, on utilisera la commande **# /etc/init.d/ssh restart** pour forcer la prise en compte des modifications.

Maintenant pour vérifier que cela fonctionne, il suffit simplement d'utiliser putty et d'essayer de se connecter sur le port 22, normalement aucune réponse possible. Ensuite on se connecte au bon port (12543 pour l'exemple), cette fois on va essayer de se connecter en tant que root, si on rentre les bons identifiants le serveur répondra d'un « access denied ». Donc si on est là tout va bien, on va encore se connecter au serveur mais avec l'utilisateur simple cette fois ci (dimitri pour l'exemple), et normalement on est censé pouvoir se connecter. Bien entendu il est possible ensuite de se connecter au root avec la commande **$ su root**.

E. Serveur passif

Pour préparer la mise en place du service Heartbeat. Si on est sur machine virtuelle, on peut gagner du temps en clonant le serveur actuel pour créer une machine passive pour heartbeat. Si on clone notre machine après avoir installé haproxy et heartbeat, on aura des défauts de fonctionnement.

Ce qui va suivre sera donc à répéter pour chaque serveur, mais il faudra adapter les fichiers de configuration pour chaque serveur. Je préciserais ce qui doit être adapté et en fonction de quoi. On

va donc cloner notre machine, l'éteindre et configurer notre clone. Il ne faut pas oublier de réinitialiser l'adresse MAC.

a. Network

Même si on a réinitialisé les adresses MAC, on a conservé la même configuration ce qui inclus eth0, donc on veut modifier eth0 pour le remplacer par notre eth1 (voir en début de document la méthode). Il est préférable de ne pas faire la configuration avec la machine originelle en marche. Une fois fait on fait un **reboot**, puis on vérifie avec **# ifconfig** que l'adresse IP est bonne, si ce n'est pas le cas on fait **# /etc/init.d/networking restart** puis on revérifie avec **# ifconfig** et normalement le changement d'adresse est bon.

Donc nous avons désormais une seconde machine sur l'interface eth0 qui utilise l'adresse IP 192.168.1.171 /24 et récupère le reste des informations via le DHCP.

b. Hostname

Nous allons maintenant modifier le nom d'hôte du serveur passif pour qu'il diffère de la machine originelle. On utilise la commande **# nano /etc/hostname** puis on remplace « serveur-HAPROXY-base » par « serveur-HAPROXY-passif-base ».

```
 GNU nano 2.2.6          Fichier : /etc/hostname                  Modifié

serveur-HAPROXY-passif-base
```

Et on réinitialise le nom avec **# /etc/init.d/hostname restart** puis **# exit** et on se reconnecte.

```
root@serveur-HAPROXY-base:~# /etc/init.d/hostname restart
Rather than invoking init scripts through /etc/init.d, use the service(8)
utility, e.g. service hostname restart

Since the script you are attempting to invoke has been converted to an
Upstart job, you may also use the stop(8) and then start(8) utilities,
e.g. stop hostname ; start hostname. The restart(8) utility is also available.
hostname stop/waiting
```

```
root@serveur-HAPROXY-passif-base:~# _
```

Après s'être reconnecté, on constate que le nom a bel et bien été modifié.

c. /etc/hosts

Il faut modifier le fichier /etc/hosts pour l'adapter au passif avec la commande **# nano /etc/hosts**.

On peut démarrer notre serveur actif.

F. Mise en place de haproxy

Il est possible de faire de la tolérance aux pannes avec heartbeat (principe où un serveur web prend le relai d'un autre lorsque celui-ci tombe en panne). Mais nous ne gérons pas la répartition de charge avec ce système. Il est donc possible de faire un cluster de serveurs web (au moins 2) qui fonctionneront ensemble, mais cette fois la charge se répartira entre ces deux serveurs. Lorsque l'un des serveurs tombe en panne alors toute la charge sera systématiquement répartie sur le reste du cluster, donc on gère aussi la tolérance aux pannes.

On satisfait donc nos besoins sauf que nous avons un problème, c'est que notre serveur haproxy (qui est la solution pour gérer notre cluster) va recevoir toutes les requêtes et les répartir ensuite. Par conséquent s'il tombe en panne alors il n'y plus d'accès possible (c'est donc là que nous utiliserons heartbeat pour assurer la tolérance au panne sur le serveur haproxy et éviter d'avoir un Single Point Of Failure (zone du réseau sensible à la panne, ce qui mettrait le réseau en panne ou en grande partie).

Attention, ce service ne fonctionnera pas avec les certificats SSL.

1) Installation du paquet

La base même du serveur haproxy est d'installer le paquet gérant ce service. Pour installer le paquet nous utiliserons la commande **# apt-get install haproxy**.

```
root@serveur-HAPROXY-base:~# apt-get install haproxy
```

2) Configuration de haproxy

Nous allons maintenant configurer haproxy correctement. Mais avant cela, nous allons voir ce qui doit être fait sur nos serveurs web comme modifications pour que cela fonctionne.

a) Modifications à apporter sur les serveurs web

Nous allons créer un fichier spécial qui permettra à haproxy de savoir si le service apache2 fonctionne. On utilisera la commande **# touch /var/www/haproxytest.txt**, attention ce fichier doit se trouver dans le DocumentRoot de l'adresse IP du serveur.

```
root@serveur-WEB-base:~# touch /var/www/haproxytest.txt
root@serveur-WEB-base:~# _
```

Il faudra aussi modifier la configuration du fichier des VirtualHost pour éviter que les requêtes du serveur haproxy ne soient comprises dans les fichiers de log (sinon il sera très vite trop fastidieux à lire). On utilisera la commande **# nano /etc/apache2/sites-available/default** puis on ajoutera une ligne et on en modifiera une. Voir ci-dessous :

```
root@serveur-WEB-base:~# nano /etc/apache2/sites-available/default
```

On avait :

```
        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
_
```

Après les modifications on obtient :

```
        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        SetEnvIf Request_URI "^/haproxytest\.txt$" dontlog
        CustomLog /var/log/apache2/access.log combined env=!dontlog
```

Il suffira ensuite de redémarrer apache2 avec la commande **# /etc/init.d/apache2 restart**.

```
root@serveur-WEB-base:~# /etc/init.d/apache2 restart
```

b) Modifications à apporter sur le serveur haproxy

On va maintenant sur le serveur haproxy supprimer le fichier de configuration par défaut et en créer un nouveau. On utilise la commande **# rm –f /etc/haproxy/haproxy.cfg** puis la commande **# nano /etc/haproxy/haproxy.cfg** et on y mettra le contenu comme suit :

```
root@serveur-HAPROXY-base:~# rm -f /etc/haproxy/haproxy.cfg
root@serveur-HAPROXY-base:~# nano_/etc/haproxy/haproxy.cfg
```

```
  GNU nano 2.2.6        Fichier : /etc/haproxy/haproxy.cfg

global
        log 127.0.0.1 local0
        log 127.0.0.1 local1 notice
        #log loghost local0 info
        maxconn 4096
        #debug
        #quiet
        user haproxy
        group haproxy

defaults
        log global
        mode http
        option httplog
        option dontlognull
        retries 3
        option redispatch
        maxconn 2000
        contimeout 5000
        clitimeout 50000
        srvtimeout 50000

listen srv-web 192.168.1.170:80
        mode http
        stats enable
```

La suite :

```
        srvtimeout 50000

listen srv-web 192.168.1.170:80
        mode http
        stats enable
        stats auth user:password
        balance roundrobin
        cookie SERVID prefix
        option httpclose
        option forwardfor
        option httpchk HEAD /haproxytest.txt HTTP/1.0
        #server srv-web1 192.168.1.XXX:80 cookie A check weight 200
        #server srv-web2 192.168.1.YYY:80 cookie B check weight 100
        #server srv-web3 192.168.1.ZZZ:80 cookie C check weight 100
```

Dans « listen » : On définit que l'on écoute sur l'adresse IP du haproxy (192.168.1.170 sur le port http « 80 »). Mais il faut adapter l'adresse IP en fonction de l'adresse IP du serveur, on verra plus tard que l'on modifiera cette adresse IP pour écouter sur l'adresse IP virtuelle fournie par heartbeat.

- Stats enable : Permet d'accéder à l'aide des identifiants de stats auth à une interface web présentant les statistiques d'utilisation de haproxy et des différents serveurs apache2 grâce à l'URL : http://IP-haproxy/haproxy?stats (IP-haproxy correspond à 192.168.1.170 dans notre cas).
- Stats auth : Définit que les identifiants pour se connecter à l'interface web des statistiques.
- Option httpchk : Définit que l'on vérifie si oui ou non on arrive à trouver le fichier haproxytest.txt dans le DocumentRoot du serveur web que haproxy vérifie (pour savoir si oui ou non apache2 fonctionne sur le serveur ou même si le service est accessible tout simplement).
- Server : On définit ici nos différents serveurs qui appartiennent à notre cluster.
    - Check : définit que l'on souhaite vérifier l'état du serveur par l'existence du fichier haproxytest.txt.
    - Weight : définit un poids au serveur, plus le poids est important et plus la répartition de charge sera ciblé sur celui-ci. Si l'on suit par exemple ce qui est donné comme base, on a notre premier serveur qui recevra autant de requêtes que les 2 autres serveurs réunis (si on peut résumer ainsi). Cela va permettre de définir les machines puissantes comme pouvant prendre plus de requêtes, et les machines moins puissantes en prendront moins.

On va maintenant modifier un fichier qui va permettre de faire fonctionner les scripts de démarrage du service (start/stop/restart) avec la commande **# nano /etc/default/haproxy** et mettre ENABLED à 1 et non 0 :

```
root@serveur-HAPROXY-base:~# nano /etc/default/haproxy
```

```
  GNU nano 2.2.6         Fichier : /etc/default/haproxy                Modifié

# Set ENABLED to 1 if you want the init script to start haproxy.
ENABLED=1
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
```

On va maintenant démarrer le service avec la commande **# /etc/init.d/haproxy start**.

```
root@serveur-HAPROXY-base:~# /etc/init.d/haproxy start
```

Vous pouvez désormais tenter de voir si le serveur haproxy détecte bien tous vos serveurs web avec pour notre exemple l'adresse URL http://192.168.1.170/haproxy?stats.

On a réussi à faire notre cluster, mais ce n'est pas parfait. Oui vous pouvez toujours accéder directement à vos serveurs web en utilisant leur adresse IP lors de la requête http.

En plus de cela, si votre haproxy tombe en panne, les serveurs web seront inaccessibles (si on ne peut pas accéder par leur propre adresse IP). Je vais seulement résoudre ce dernier problème et montrer comment se règle le premier par un schéma réseau.

G. Tolérance aux pannes avec Heartbeat

Nous allons désormais mettre en place un service de tolérance aux pannes, qui va permettre lorsque notre serveur haproxy tombe en panne ou n'est plus accessible d'être remplacé temporairement par un autre serveur haproxy reprenant la même configuration. Normalement il faudrait reconfigurer un nouveau serveur mais on a la chance d'être en virtuel donc on va faire un clone de la machine (sans les instantanés) et sans oublier de réinitialiser les adresses MAC. Ce nouveau serveur s'appellera serveur-HAPROXY-passif-base.

1. Installation de Heartbeat

On va maintenant installer heatbeat avec la commande **# apt-get install heartbeat**.

```
root@serveur-HAPROXY-base:~# apt-get install heartbeat_
```

L'installation est assez longue.

2. Configuration de Heartbeat

On va devoir configurer heartbeat, pour cela il faut créer 3 fichiers et c'est ce que nous allons voir maintenant.

a. /etc/heartbeat/ha.cf

Ce fichier détermine la liste des machines à utiliser et la manière de dialoguer entre elles.

On utilise la commande **# nano /etc/heartbeat/ha.cf** pour le créer et l'éditer.

Voilà ce qu'il faut mettre dans ce fichier :

```
  GNU nano 2.2.6         Fichier : /etc/heartbeat/ha.cf                  Modifié

logfile /var/log/ha-log
logfacility     local0
keepalive       2
deadtime        10
bcast   eth0
node    serveur-HAPROXY-base serveur-HAPROXY-passif-base
auto_failback   on
respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster
```

Attention, la différence pour ce fichier entre le serveur actif et le passif est pour la carte réseau de broadcast qui doit être adaptée en fonction de la carte réseau utilisée par le serveur. Dans mon cas, les deux serveurs utilisent tous les deux eth0 comme carte réseau.

b. /etc/heartbeat/haresources

Ce fichier indique les opérations à effectuer au démarrage de la haute disponibilité sur une machine.

La syntaxe est la suivante : NOM_1er_inode action1 action2…actionN.

Bon ce n'est pas très parlant alors notre cas voilà ce que ça va donner :

serveur-HAPROXY-base IPaddr::192.168.1.170/24/eth0 haproxy

On utilise donc la commande **# nano /etc/heartbeat/haresources** pour créer et éditer ce fichier et on y ajoute cette ligne telle quelle ce qui nous donne :

```
 GNU nano 2.2.6       Fichier : /etc/heartbeat/haresources

serveur-HAPROXY-base IPaddr::192.168.1.169/24/eth0 haproxy
```

C'est cette adresse IP qui servira d'adresse commune pour nos deux serveurs. Le nom d'hôte que l'on donne au début est celui qui définit quel serveur sera le serveur actif pour heartbeat, donc on aura la même chose pour nos deux serveurs.

c. /etc/heartbeat/authkeys

Ce fichier détermine la clé et le protocole de protection utilisé. Voilà un exemple, on utilise la commande **# nano /etc/heartbeat/authkeys** pour le créer et l'éditer :

```
 GNU nano 2.2.6       Fichier : /etc/heartbeat/authkeys              Modifié

auth3
3 md5 passeHeartbeat
```

On va modifier les droits sur le fichier pour le protéger avec la commande **# chmod 600 /etc/heartbeat/authkeys**, on le rend donc modifiable et lisible par root et aucun droit pour tout le reste.

```
root@serveur-HAPROXY-base:~# chmod 600 /etc/heartbeat/authkeys
```

Ceci est à faire sur les deux serveurs il ne faut pas que le fichier soit différent.

d. Arrêt du service (haproxy)

Avant de poursuivre, il faut s'assurer qu'on a bien arrêté le service concerné par heartbeat sur notre serveur. La commande est **# /etc/init.d/nom_service stop** puis **# update-rc.d –f nom_service remove**. Cette dernière permet d'empêcher le service de se démarrer automatiquement au démarrage de la machine. Donc on va faire :

```
root@serveur-HAPROXY-base:~# /etc/init.d/haproxy stop
 * Stopping haproxy haproxy                                        [ OK ]
root@serveur-HAPROXY-base:~# update-rc.d -f haproxy remove
 Removing any system startup links for /etc/init.d/haproxy ...
   /etc/rc0.d/K20haproxy
   /etc/rc1.d/K20haproxy
   /etc/rc2.d/S20haproxy
   /etc/rc3.d/S20haproxy
   /etc/rc4.d/S20haproxy
   /etc/rc5.d/S20haproxy
   /etc/rc6.d/K20haproxy
```

Puis on va faire un **# reboot** de la machine et lancer la commande **# service haproxy status** n'a pas démarré au démarrage :

```
root@serveur-HAPROXY-base:~# service haproxy status
haproxy not running.
```

e. /etc/hosts

Nous allons ajouter les noms de nos serveurs dans leur liste de connaissances avec la commande **# nano /etc/hosts** :

```
 GNU nano 2.2.6              Fichier : /etc/hosts

127.0.0.1       localhost
127.0.1.1       serveur-HAPROXY-base
192.168.1.171   serveur-HAPROXY-passif-base_
```

Ce qui donne pour le second serveur :

```
 GNU nano 2.2.6              Fichier : /etc/hosts

127.0.0.1       localhost
127.0.1.1       serveur-HAPROXY-passif-base
192.168.1.170   serveur-HAPROXY-base
```

f. Modification de haproxy

Maintenant que nous avons mis en place heartbeat, nous allons devoir modifier la configuration de haproxy pour qu'elle convienne à l'arrivée de heartbeat. C'est-à-dire plus exactement qu'il faut que haproxy écoute désormais l'adresse virtuelle de heartbeat. On va donc utiliser la commande **# nano /etc/haproxy/haproxy.cfg** et modifier l'adresse IP de notre Listen, on remplace 192.168.1.170 par 192.168.1.169.

```
listen srv-web 192.168.1.169:80
        mode http
```

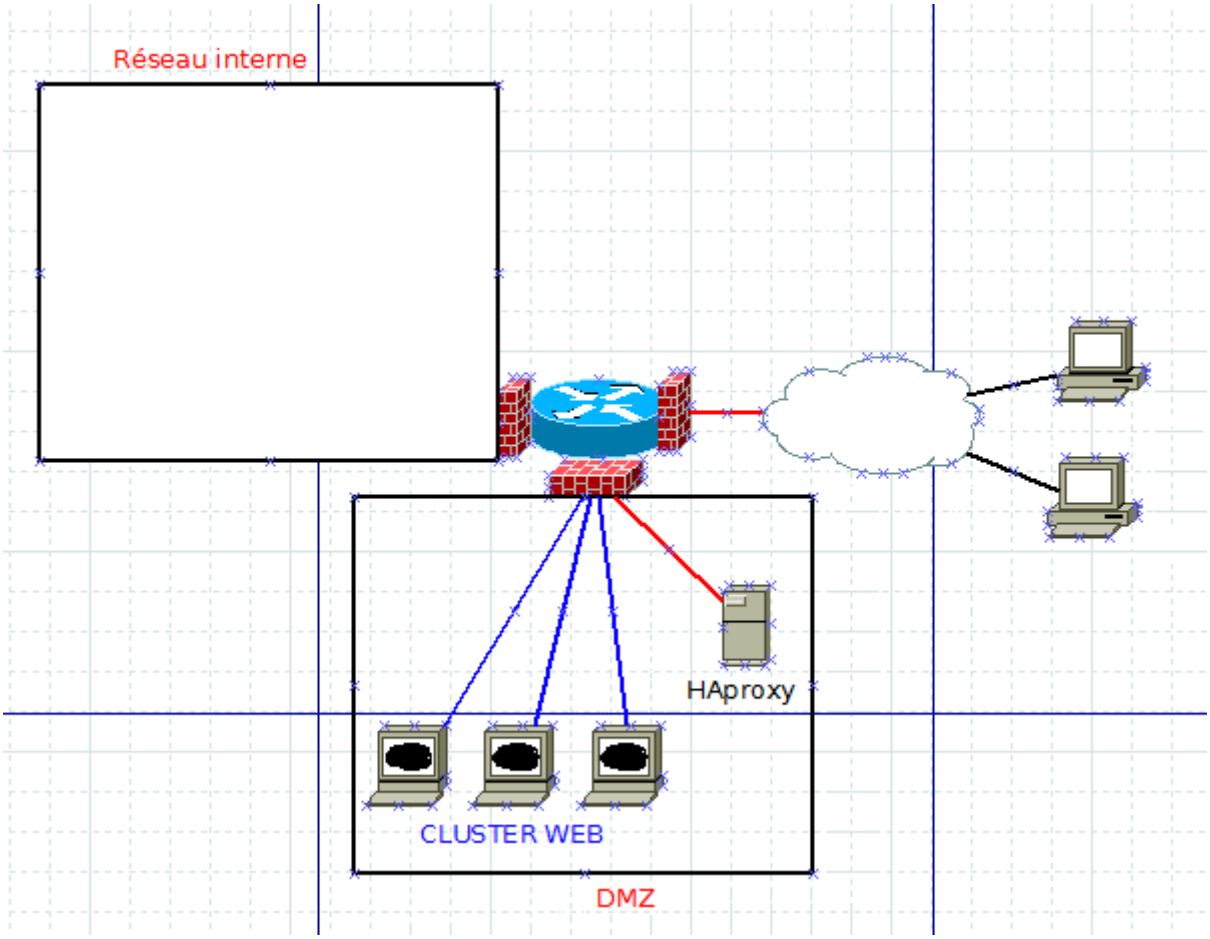
Il est important de comprendre que si on se contente de modifier l'adresse de cette manière, alors le contournement de heartbeat sera impossible. La configuration fera que l'on écoute l'adresse IP virtuelle fournie par heartbeat, sinon haproxy ne répondra pas quand on utilisera l'adresse IP virtuelle de heartbeat.

g. Démarrage de heartbeat

On va pouvoir essayer de lancer heartbeat et vérifier son bon fonctionnement. On va donc utiliser la commande **# /etc/init.d/heartbeat start**.

3. Non au contournement

Pour empêcher le contournement du haproxy, c'est-à-dire empêcher les internautes d'accéder aux serveurs web par leur adresse IP propre. Il faut contraindre les internautes à passer par le haproxy. Ce sera très simple :



On peut par exemple, définir une règle sur le pare-feu de notre DMZ pour interdire toutes les requêtes http venant de l'extérieur de la DMZ sauf si l'adresse IP de destination est celle du HAproxy (dans notre exemple pratique on aurait pris l'adresse IP 192.168.1.169 car on écoute cette adresse avec nos deux serveurs HAproxy). D'ailleurs cette même règle empêche par la même occasion le contournement de Heartbeat.