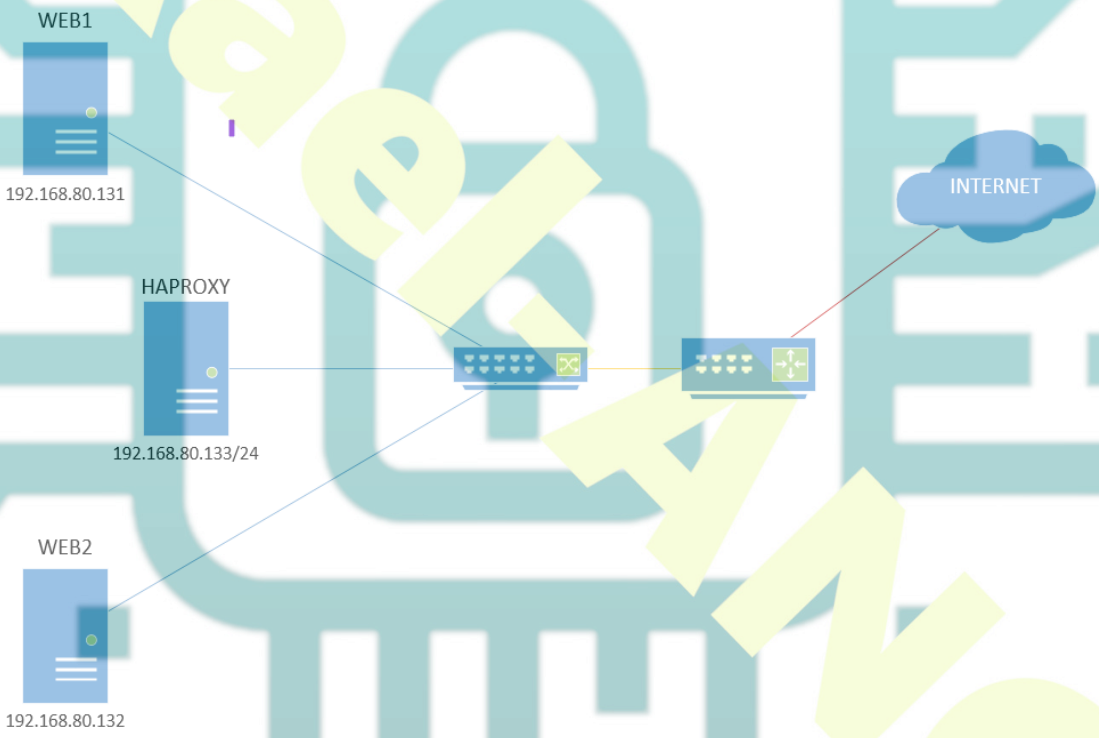
Tuto – HaProxy/Nginx

Prérequis communs

Mise en place de trois serveurs

- Un serveur de répartition de charge avec HaProxy/Nginx
- Deux serveurs web avec Apache

Topologie



Installation de DEBIANx

Pour tous les serveurs

- mot de passe **Pa$$word** pour root et user
- nom de domaine **test.lan**
- nom de machine **WEB1**, **WEB2** et **HAPROXY**
- choix d'un miroir réseau en France

- service **Apache** à installer

Contenu fichier /etc/hosts (tous les serveurs)

127.0.0.1 localhost
192.168.1.133 HAPROXY.test.lan HAPROXY
192.168.1.131 WEB1.test.lan WEB1
192.168.1.132 WEB2.test.lan WEB2

Serveur WEB1

hostname: WEB1
adresse IP: 192.168.80.131/24
Domaine: test.lan

Configurer la page d'accueil
/var/www/html/index.html

```
<html>
<head>
<title>SITE 1</title>
</head>
<body>
        <H1>WEB1 192.168.80.131</H1>
</body>

</html>
```

Serveur WEB2

hostname: WEB2
adresse IP: 192.168.80.132/24
Domaine: test.lan

Configurer la page d'accueil
/var/www/html/index.html

```
<html>
<head>
<title>SITE 2</title>
</head>
<body>
        <H1>WEB2 192.168.80.132</H1>
</body>
</html>
```

Partie Haproxy

Serveur HaProxy

hostname: HAPROXY
adresse IP: 192.168.80.133/24
Domaine: test.lan

Changer le port d'écoute du serveur Apache de HAPROXY (sinon le service HaProxy refusera de démarrer.

nano c/etc/apache2/ports.conf

Il suffit donc de remplacer **listen 80** par **listen 8080** pour faire écouter cet hôte sur le port **8080**

Il est à noter que pour modifier le port d'écoute sur les sites HTTPS et qui s'appuie sur le module SSL d'Apache, vous devez modifier les deux lignes "Listen 443" avec le port de votre choix (différent de l'autre port).

Il faut ensuite configurer les hôtes virtuels pour écouter également sur ce port. En effet, les virtuals host d'un serveur Apache peuvent écouter chacun sur un port différent, il faut donc dans Apache spécifier pour chaque hôte virtuel sur quel port il doit écouter.

Le site actif par défaut dans Apache2 est **“Default”,** on trouve son fichier de configuration dans “**/etc/apache/sites-enabled/000-default.conf**“, qu’il va falloir éditer :

Il suffit donc de remplacer encore une fois **80** par **8080** pour faire écouter cet hôte sur le port **8080** et de relancer le service Apache.

```
systemctl reload apache2
```

Installation de Haproxy

```
apt update
apt install -y haproxy
```

- Après installation on active et on redémarre le service

```
sudo systemctl enable haproxy
sudo systemctl restart haproxy
```

- Après installation on vérifie le service

```
sudo systemctl status haproxy
```

Configuration de Haproxy

- **Copier le fichier /etc/haproxy/haproxy.cfg dans haproxy.bak**
- **Puis ouvrir le fichier haproxy.cfg** et ajouter les lignes en gras à la fin du fichier.

```
Global
log /dev/log local0
log /dev/log local1 notice
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin expose-
fd listeners
stats timeout 30s
user haproxy
group haproxy
daemon

# See: https://ssl-config.mozilla.org/#server=haproxy&server-
version=2.0.3&config=intermediate
ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-
AES128-GCM-SHA256:ECDHE-ECDSA-A>
ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY13
0>
ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
log global
mode http
option httplog
option dontlognull
timeout connect 5000
timeout client 50000
timeout server 50000
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http
```

frontend HAPROXY
bind *:80

```
mode http
default_backend WEBSERVER

backend WEBSERVER
balance roundrobin
server WEB1 192.168.80.131:80 check
server WEB2 192.168.80.132:80 check
option httpchk
```

- Enregistrer le fichier et vérifier le fichier de conf

Tester la configuration

```
haproxy -f haproxy.cfg -c
```

Les sections

mode
Permet de spécifier le type de protocole visé par HaProxy. On utilisera le mode tcp pour mysql par exemple.
balance
Spécifie le choix concernant le fonctionnement de la répartition de charge.
RoundRobin : permet une répartition équitable entre les serveurs d'un cluster.
Least connection : le HaProxy renvoie les clients vers le serveur le moins chargé (peut poser problème car ce n'est pas toujours bien compris par HaProxy)
First Response : Les requêtes clientes sont envoyées simultanément à tous les serveurs et le premier qui répond a gagné.
Weight : Le paramètre weight permet de jouer sur l'équité de la répartition.

Poids

server WEB1 192.168.1.131:80 check weight 30

```
server WEB2 192.168.1.132:80 check weight 120
```

3. **server check**
Spécifie les serveurs web cibles lors de la répartition de charge

4. **server check** avec connexion persistante via Cookies
HAProxy va insérer un cookie (BACKENDID) qui sera utilisé lors des communications ultérieures entre le navigateur et le load balancer pour aiguiller les communications proprement.

Cookies

```
balance source
hash-type consistent
cookie BACKENDID insert indirect nocache
server WEB1 192.168.80.131:80 check cookie w1
server WEB2 192.168.80.132:80 check cookie w2
```

Stats

Permet de configurer la page de statistiques de HaProxy
#mettre en place les stats à la fin de la section default

```
stats enable
stats hide-version
stats refresh 30s
stats show-node
stats auth admin:Pa$$word
stats uri /haproxy?stats
```

- **Accéder aux statistiques**

URL: http://192.168.10.10/haproxy?stats

Login **admin** password **Pa$$word**

SSL pour Haproxy

Il existe deux stratégies principales.

- La terminaison SSL consistant à terminer/déchiffrer une connexion SSL au niveau du Haproxy et à envoyer des connexions non chiffrées aux serveurs principaux.
- Le SSL Pass-Through, qui envoie les connexions SSL directement aux serveurs mandatés. Avec SSL-Pass-Through, la connexion SSL est interrompue sur chaque serveur proxy.

Générer un certificat

- Création de la private key **(KEY)**

```
openssl genrsa -out testlan.key 2048
```

- Génération d'une demande de signature de certificat **(CSR)**

```
openssl req -new -key testlan.key -out testlan.csr
```

- Création du certificat autosigné **(CRT)**

```
openssl x509 -req -days 365 -in testlan.csr -signkey testlan.key -out testlan.crt
```

- Création du PEM avec le .KEY et le .CRT

```
bash -c 'cat testlan.key testlan.crt > /etc/ssl/certs/testlan.pem'
```

En cas de problème de longueur de clé saisir :
openssl dhparam – out /etc/haproxy/dhparams.pem 2048
puis dans le fichier haproxy.cfg ajouter après les lignes ssl
ssl-dh-param-file /etc/haproxy/dhparams.pem

Exemple de fichier SSL terminaison

```
frontend HAPROXY
bind *:80
bind *:443 ssl crt /etc/ssl/certs/testlan.pem
mode http
default_backend WEBSERVER

backend WEBSERVER
mode http
balance roundrobin
option forwardfor
option httpchk
server web1 192.168.80.131:80 check
server web2 192.168.80.132:80 check
```

- Forcer la redirection https

```
bind *:80
bind *:443 ssl crt /etc/ssl/certs/testlan.pem
redirect scheme https if !{ ssl_fc }
mode http
default_backend WEBSERVER
```

Sécuriser l'accès au Haproxy via ACL

Restrictions par User agent

Cette option permet de tester la plateforme (OS, navigateur…) qui se connecte au HaProxy. On peut interdire, autoriser ou rediriger les clients via un fichier contenant la liste des user agent sur lesquels on souhaite effectuer le filtrage.

```
frontend HAPROXY
bind *:80
acl is-blockedagent hdr_sub(user-agent) -i -f /etc/haproxy/blacklist-agent.txt
http-request deny if is-blockedagent
mode http
default_backend WEBSERVER
```

Exemple de contenu du fichier blacklist-agent.txt

```
#smartphone
iphone
android
#Navigateur
Edg
#OS
Windows
```

Restrictions par adresse IP

Ajouter une condition acl is-blocked-ip src -f /etc/haproxy/blocklisted.ips

```
frontend HAPROXY
bind *:80
acl is-blocked-ip src -f /etc/haproxy/blocklisted.ips
http-request deny if is-blocked-ip
```

```
mode http
default_backend WEBSERVER
```

Exemple de contenu du fichier blocklisted.ips

```
1.1.1.1
192.168.1.0/16
80.3.9.0/28
```

Redirection

Rediriger la demande du client vers un autre domaine

HAPROXY permet de rediriger sur des ports, des noms, en fonction du nom de domaine ou de l'adresse IP saisie dans l'ACL.

```
frontend HAPROXY
bind *:80
acl redirect hdr_dom(host) -i www.gp.com
acl redirect hdr_dom(host) -i 192.168.1.133
http-request redirect location https://www.google.com code 301 if redirect
```

Par l'intermédiaire d'un fichier contenant les différentes règles

```
frontend HAPROXY
bind *:80
acl redirect hdr_dom(host) -i -f /etc/haproxy/sites.txt
http-request redirect location https://www.google.fr code 301 if redirect
```

Contenu du fichier sites.txt

www.gp.com
www.gp.fr
gp.com
gp.fr
192.168.1.133

Rediriger le domaine vers un backend spécifique

frontend HAPROXY
mode http
bind *:80
use_backend SITECOM if { req.hdr(host) -i gp.com }
use_backend SITEFR if { req.hdr(host) -i gp.fr }

backend SITECOM
mode http
web1 192.168.80.131:80
web2 192.168.80.132:80

backend SITEFR
mode http
web1 192.168.1.83:80
web2 192.168.1.84:80

Partie Nginx

Serveur Nginx

hostname: NGINX
adresse IP: 192.168.80.133/24
Domaine: lan

Installation de Nginx

```
apt update
apt install -y nginx
```

- Après installation on active et on redémarre le service

```
sudo systemctl enable nginx
sudo systemctl restart nginx
```

- Après installation on vérifie le service

```
sudo systemctl status nginx
```

Configuration de Nginx

- Créer un fichier pour gérer le reverse proxy

```
sudo nano /etc/nginx/sites-available/sites.lan.conf
ln -s /etc/nginx/sites-available/sites.lan.conf /etc/nginx/sites-enabled/sites.lan.conf
```

Dans NGINX, l'ensemble de serveurs vers lequel vous acheminez est appelé en amont et est configuré comme une liste d'adresses énumérées :

```
upstream siteweb {
least_conn;
server 192.168.80.131;
server 192.168.80.132;
}
server {
server_name 192.168.80.133;
```

```
listen 80;
location / {
proxy_pass http://siteweb;
}
}
```

- Enregistrer le fichier

Les sections

upstream
Permet de définir un groupe de serveur pouvant répondre à la requête.
least_conn permet de sélectionner le serveur qui a le moins de connexions.
server
server_name définit le nom auquel Nginx devra réagir.
location
Indique où les requêtes seront redirigées et dans quelle condition. le "/" après location indique la racine du serveur web cible ou le site se trouve.
proxy_pass
Permet d'affecter un ensemble de serveur à cette configuration, le nom à indiqué ici est celui du bloc "upstream".
proxy_set_header
Permet d'ajouter un header à la requête qui va être passée au serveur web par le proxy. Le header "Host" par exemple, est le nom de domaine visé par la requête initiale
proxy_connect_timeout et proxy_send_timeout
ces directives permettent de gérer le temps (en seconde) au delà duquel le serveur web (backend) sera considéré comme injoignable/down.

Poids

Le poids (**weight**) donne la priorité à un serveur. Vous pouvez également définir les connexions maximales et divers délais d'attente.

```
upstream siteweb {
server 192.168.80.131 weight=5;

server 192.168.80.132;
}
```

Connexion persistante

La forme la plus élémentaire de persistance de session consiste à utiliser un hachage IP. NGINX utilisera l'adresse IP pour identifier les utilisateurs, puis s'assurera que ces utilisateurs ne changent pas de serveur en cours de session

```
upstream backend {
ip_hash;
server backend1.example.com;
server backend2.example.com;
}
```

SSL pour Nginx

Générer un certificat

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/certs/server.key -out /etc/ssl/certs/server.crt
```

Exemple de fichier SSL terminaison

```
upstream siteweb {
server 192.168.80.131;
server 192.168.80.132;
}
server {
listen 443 ssl;
server_name 192.168.80.133;
ssl_certificate /etc/ssl/certs/server.crt;
ssl_certificate_key /etc/ssl/certs/server.key;
```

```
location / {
proxy_pass http://siteweb;
}
}
```

- Forcer la redirection https

```
server {
listen 80;
server_name 192.168.80.133;
return 301 https://192.168.80.133$request_uri;
}
```

Redirection

Rediriger la demande du client vers un autre domaine

Nginx permet de rediriger sur des ports, des noms, en fonction du nom de domaine ou de l'adresse IP saisie dans l'ACL.

```
server {
listen 80;
server_name 192.168.80.133;
return 301 $scheme://www.shooga.ovh$request_uri;
}
```

- Cliquer sur l'icone pour en savoir plus

https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/