HAPROXY

Documentation Technique

Installation & Configuration

HAProxy + Keepalived

Cluster Haute Disponibilité – Load Balancing

Projet BTS SIO – Option SISR

1. Qu'est-ce que HAProxy ?

HAProxy est un logiciel de load balancing et de reverse-proxy. En termes simples, il répartit le trafic réseau entre plusieurs serveurs.

À quoi ça sert ?

- Distribuer les requêtes vers plusieurs serveurs web
- Éviter qu'un seul serveur soit surchargé
- Assurer de meilleures performances globales
- Vérifier si un serveur est en panne (health checks)
- Rediriger automatiquement vers les serveurs encore en ligne

Exemple simple

Avec deux serveurs web disponibles :

- 192.168.1.101
- 192.168.1.102

HAProxy envoie une requête sur le premier serveur, la suivante sur le second, et ainsi de suite. Le site reste rapide et disponible même si l'un des serveurs tombe.

2. Qu'est-ce que Keepalived ?

Keepalived est un outil de haute disponibilité (HA). Il utilise un protocole appelé VRRP qui permet de partager une IP virtuelle (VIP) entre deux serveurs.

À quoi ça sert ?

- Avoir une IP virtuelle qui bascule automatiquement d'un serveur à l'autre
- Assurer qu'un service reste disponible même si un nœud tombe
- Surveiller l'état d'un service (par exemple HAProxy)

Exemple simple

Configuration typique avec deux serveurs :

- node1 (MASTER) – priorité la plus haute
- node2 (BACKUP) – prend le relais si le MASTER tombe

Les deux nœuds se partagent une IP virtuelle : 192.168.1.50. Si le serveur principal tombe, l'IP 192.168.1.50 passe automatiquement au serveur secondaire et le service reste disponible pour les utilisateurs.

3. Comment HAProxy et Keepalived fonctionnent ensemble ?

Les deux outils forment un cluster complémentaire :

- HAProxy distribue le trafic vers plusieurs serveurs web (load balancing)
- Keepalived assure que HAProxy reste toujours disponible grâce à la VIP

Exemple de fonctionnement

1. Les utilisateurs se connectent à 192.168.1.50 (IP virtuelle)
2. Keepalived garantit que cette IP est toujours associée à un serveur en bon état
3. HAProxy répartit le trafic vers les serveurs web derrière lui

> 💡 Même si un nœud tombe, le service reste accessible grâce à la bascule automatique de l'IP virtuelle.

4. Procédure d'installation d'un cluster Keepalived + HAProxy

Installation HAProxy (Debian/Ubuntu)

Installez HAProxy avec les commandes suivantes :

```
sudo apt update
sudo apt install haproxy
```

Exemple de configuration /etc/haproxy/haproxy.cfg

```
frontend web_front
    bind *:80
    default_backend web_servers

backend web_servers
    balance roundrobin
    server web1 192.168.1.11:80 check
    server web2 192.168.1.12:80 check
```

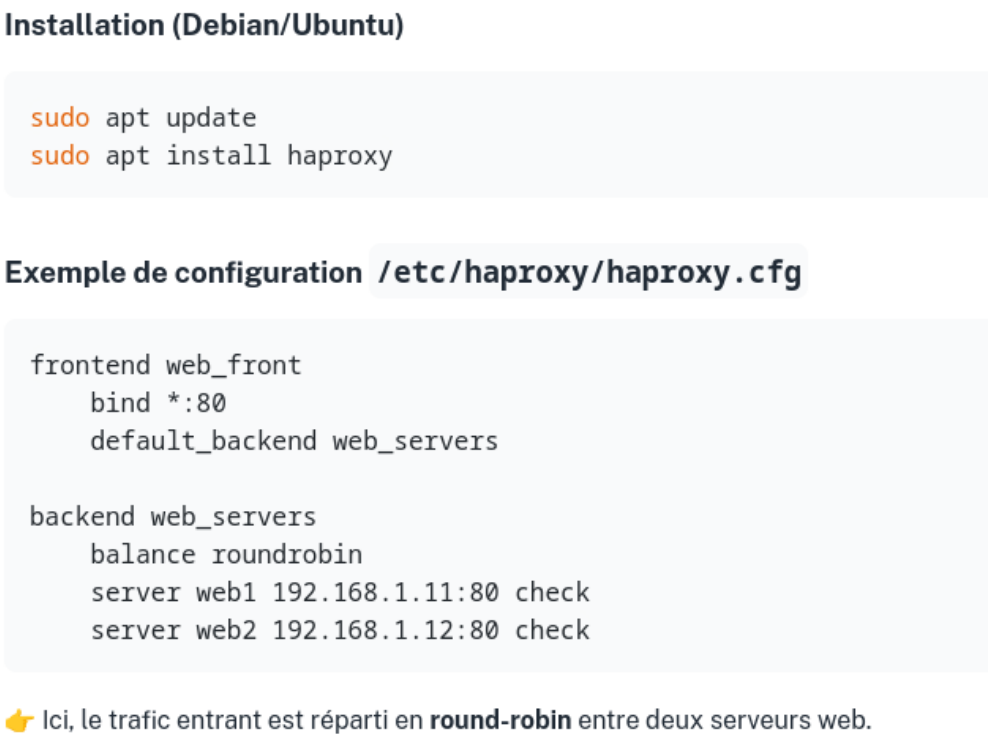

Figure 1 – Exemple de configuration HAProxy (round-robin entre deux serveurs web)

> 💡 Ici, le trafic entrant est réparti en round-robin entre deux serveurs web.

5. Mise en place d'un cluster avec Keepalived

Étape 1 : Installer Keepalived

```
sudo apt install keepalived
```

Étape 2 : Exemple de configuration /etc/keepalived/keepalived.conf

```
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1234
    }
    virtual_ipaddress {
        192.168.1.100
    }
}
```

1. Installer keepalived :

```
sudo apt install keepalived
```

2. Exemple de configuration `/etc/keepalived/keepalived.conf` :

```
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1234
    }
    virtual_ipaddress {
        192.168.1.100
    }
}
```

👉 Si le nœud principal tombe, le secondaire prend automatiquement la main.

Figure 2 – Configuration keepalived.conf (nœud MASTER)

💡 Si le nœud principal tombe, le secondaire prend automatiquement la main.

6. Configuration IP Statique (interfaces)

Configuration de l'interface réseau avec une adresse IP statique :

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
address 192.168.1.30
netmask 255.255.255.0
gateway 192.168.1.254
dns-nameservers 8.8.8.8
# This is an autoconfigured IPv6 interface
iface enp0s3 inet6 auto
```

Figure 3 – Fichier /etc/network/interfaces avec IP statique

Explication des paramètres

`enp0s3`	*Nom de l'interface réseau physique (peut varier selon le matériel : eth0, ens33, etc.)*
`allow-hotplug enp0s3`	*Active l'interface automatiquement si un câble réseau est branché (détection à chaud)*
`iface enp0s3 inet static`	*Configure l'interface pour utiliser une adresse IP statique*
`dns-nameservers 8.8.8.8`	*Adresse IP du serveur DNS (ici, le serveur DNS public de Google)*

7. Configuration VRRP et Script de vérification

Exemple de configuration avancée avec surveillance du processus HAProxy :

```
vrrp_script reload_haproxy {
        script "killall -0 haproxy"
        interval 1
}

vrrp_instance VIH {
   virtual_router_id 7
   state MASTER
   priority 100
   # Check inter-load balancer toutes les 1 secondes
   advert_int 1
   # Synchro de l'état des connexions entre les LB sur l'interface enp0s3
   lvs_sync_daemon_interface enp0s3
   interface enp0s3
   # Authentification mutuelle entre les LB, identique sur les deux membres
   authentication {
        auth_type PASS
        auth_pass secret
   }
# Interface réseau commune aux deux LB
   virtual_ipaddress {
        192.168.1.35/32 brd 192.168.1.255 scope global
   }

   track_script {
       reload_haproxy
   }

}
```

Figure 4 – Configuration vrrp_script et vrrp_instance (nœud MASTER)

Explication des directives

`vrrp_script`	*Définit un script utilisé par VRRP pour vérifier l'état d'un service*
`reload_haproxy`	*Nom du script de vérification*
`script "killall -0 haproxy"`	*Vérifie si le processus HAProxy est en cours d'exécution*
`killall -0 haproxy`	*Envoie un signal nul à HAProxy. Si HAProxy répond, le script retourne un succès*

8. Rôles des nœuds : MASTER et BACKUP

8.1 Routeur MASTER

- Rôle : Le routeur MASTER est le routeur actif qui gère le trafic réseau et répond aux requêtes pour l'adresse IP virtuelle.
- Adresse IP virtuelle : Le MASTER « possède » l'adresse IP virtuelle partagée et répond aux requêtes ARP pour cette IP.
- Priorité : Le routeur avec la priorité la plus élevée (ex : priority 100) devient généralement le MASTER.

8.2 Routeur BACKUP (ou SLAVE)

- Rôle : Le routeur BACKUP est en veille. Il ne gère pas le trafic tant que le MASTER est opérationnel.
- Priorité : Un routeur BACKUP a une priorité inférieure à celle du MASTER (ex : priority 90).
- Prêt à prendre le relais : Si le MASTER tombe en panne, le BACKUP prend automatiquement le relais et devient le nouveau MASTER.

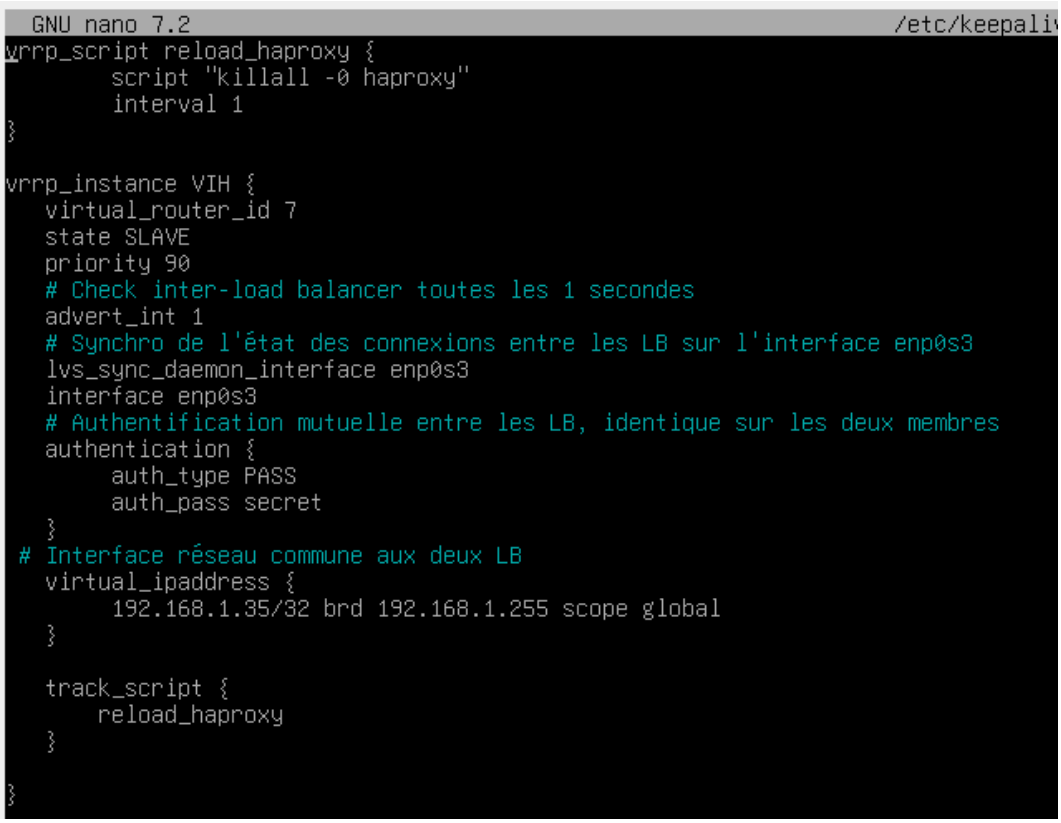

Figure 5 – Configuration keepalived.conf du nœud SLAVE (BACKUP)

💡 La différence principale entre MASTER et BACKUP est la valeur de priority. MASTER = 100, BACKUP = 90. La bascule est automatique.

9. Configuration HAProxy complète

Configuration complète avec section defaults, frontend, backend et stats :

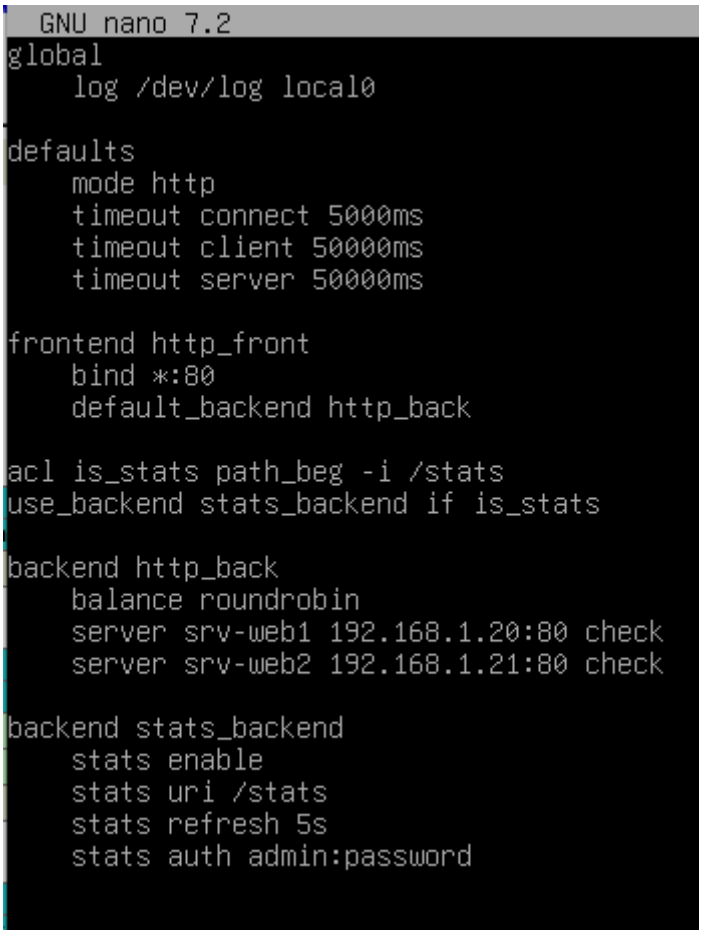

Figure 6 – Configuration HAProxy complète (defaults, frontend, backend, stats)

Explication des paramètres defaults

`defaults`	*Définit les paramètres par défaut pour les sections frontend, backend et listen*
`mode http`	*HAProxy fonctionne en mode HTTP (peut aussi fonctionner en mode TCP)*
`timeout connect 5000ms`	*Temps maximum pour établir une connexion avec un serveur backend (5 secondes)*
`timeout client 50000ms`	*Temps maximum d'inactivité côté client (50 secondes)*

`timeout server 50000ms`	*Temps maximum d'inactivité côté serveur (50 secondes)*

Explication des paramètres frontend / backend

`frontend http_front`	*Définit un point d'entrée pour les requêtes clients*
`bind *:80`	*HAProxy écoute sur le port 80 de toutes les interfaces réseau*
`default_backend http_back`	*Par défaut, les requêtes sont envoyées au backend nommé http_back*
`acl is_stats path_beg -i /stats`	*Définit une ACL qui vérifie si le chemin commence par /stats (insensible à la casse)*
`use_backend stats_backend if is_stats`	*Si la requête correspond à l'ACL is_stats, elle est redirigée vers stats_backend*
`balance roundrobin`	*Les requêtes sont réparties en round-robin (tourniquet) entre les serveurs backend*
`server srv-web1 192.168.1.20:80 check`	*Définit un serveur backend avec health check activé (vérifie régulièrement la disponibilité) même chose pour le srv-web2*

💡 L'option check dans la définition des serveurs backend permet à HAProxy de vérifier automatiquement si chaque serveur est disponible et de l'exclure en cas de panne.