

Installation Keepalived



HAPROXY



Sommaire

Sommaire.....	2
1. Qu'est-ce que HAProxy ?.....	3
À quoi ça sert ?.....	3
Exemple simple.....	3
Qu'est-ce que Keepalived ?.....	4
À quoi ça sert ?.....	4
Exemple simple.....	4
Comment HAProxy et Keepalived fonctionnent ensemble ?.....	5
Procédure d'installation d'un cluster Keepalived + HAProxy.....	6
Mise en place d'un cluster avec keepalived.....	7

1. Qu'est-ce que HAProxy ?

HAProxy est un logiciel de *load balancing* et de *reverse-proxy*.

En français : **il répartit le trafic 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

Vérifier si un serveur est en panne (health checks)

Rediriger automatiquement vers les serveurs encore en ligne

Exemple simple

Tu as 2 serveurs web :

192.168.1.101

192.168.1.102

HAProxy va envoyer 1 requête sur le premier, 1 sur le second, etc.

Le site est plus rapide et ne tombe pas si un serveur tombe

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 qui bouge automatiquement** d'un serveur à l'autre

Assurer qu'un service reste disponible même si un nœud tombe

Surveiller l'état du service (par ex. HAProxy)

Exemple simple

Tu as 2 serveurs :

node1 (MASTER)

node2 (BACKUP)

Ils 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.

Le service reste disponible pour les utilisateurs.

Comment HAProxy et Keepalived fonctionnent ensemble ?

Les deux forment un cluster :

HAProxy = distribue le trafic vers plusieurs serveurs web

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.

Procédure d'installation d'un cluster Keepalived + HAProxy

Installation (Debian/Ubuntu)

```
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
```

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

Mise en place d'un cluster avec keepalived

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.

```
# 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
```

enp0s3 : Nom de l'interface réseau physique (peut varier selon le matériel, par exemple *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).

```

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
    }
}

```

***vrrp_script** : Définit un script utilisé par VRRP pour vérifier l'état d'un service.*

***reload_haproxy** : Nom du script.*

***script "killall -0 haproxy"** : Cette commande 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.*

1.1. 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 (par exemple, **priority 100**) devient généralement le MASTER.*

1.2. 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 (par exemple, **priority 90**).*

Prêt à prendre le relais : Si le MASTER tombe en panne, le BACKUP prend automatiquement le relais et devient le nouveau MASTER.

```
GNU nano 7.2 /etc/keepalived/keepalived.conf
vrrp_script reload_haproxy {
    script "killall -0 haproxy"
    interval 1
}

vrrp_instance VIH {
    virtual_router_id 7
    state SLAVE
    priority 90
    # 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
    }
}
```

```

GNU nano 7.2
global
    log /dev/log local0

defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend http_front
    bind *:80
    default_backend http_back

acl is_stats path_beg -i /stats
use_backend stats_backend if is_stats

backend http_back
    balance roundrobin
    server srv-web1 192.168.1.20:80 check
    server srv-web2 192.168.1.21:80 check

backend stats_backend
    stats enable
    stats uri /stats
    stats refresh 5s
    stats auth admin:password

```

defaults : Définit les paramètres par défaut pour les sections *frontend*, *backend*, et *listen*.

mode http : HAProxy fonctionne en mode HTTP (il 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).

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 (Access Control List) qui vérifie si le chemin de la requête 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 le backend **stats_backend**.

backend http_back : Définit un groupe de serveurs backend.

balance roundrobin : Les requêtes sont réparties entre les serveurs backend en utilisant l'algorithme **round-robin** (tourniquet).

server srv-web1 192.168.1.20:80 check : Définit un serveur backend nommé **srv-web1** avec l'adresse IP **192.168.1.20** et le port **80**. L'option **check** permet à HAProxy de vérifier régulièrement si le serveur est disponible.

server srv-web2 192.168.1.21:80 check : Même chose pour le serveur **srv-web2**.