

Morgan

Bissey

# Procédure :

# HAproxy

SOMMAIRE	
1.	INTRODUCTION
2.	Installation D'HAproxy
3.	Configuration d'HAproxy
4.	Problème rencontrer

## 1.Introduction

HAProxy est un logiciel open-source de répartition de charge et de proxy qui est utilisé pour améliorer la disponibilité et les performances des applications web. Il agit comme un équilibreur de charge en distribuant le trafic entrant entre plusieurs serveurs backend, ce qui permet de répartir la charge et d'éviter les points de défaillance uniques. HAProxy prend en charge différents algorithmes de répartition de charge, tels que round-robin, leastconn, et bien d'autres, et peut être configuré pour effectuer des contrôles de santé des serveurs backend afin de garantir la disponibilité des services. En outre, il offre des fonctionnalités avancées telles que la mise en cache, la compression de contenu et la possibilité de configurer des règles de routage et de redirection du trafic en fonction de divers critères. En bref, HAProxy est un outil puissant utilisé pour optimiser les performances, la fiabilité et la sécurité des applications web en gérant efficacement le trafic réseau.

## 2.Installation d'HAproxy

Pour commencer HAProxy est un outils de linux pour installer HAproxy nous devons taper la commande :

```
sudo apt install haproxy
```

Une fois HAproxy installer nous devons taper la commande suivante :

```
sudo apt update && sudo apt upgrade -y
```

Pour mettre à jour tous les packet / application

## 3. Configuration d'HAproxy.

Il faut aller dans le repertoire ou se situe haproxy avec la commande cd :

```
oot@test-virtual-machine:/home/test# cd /etc/haproxy/
```

Une fois dans le repertoire nous pouvons voir avec la commande ls les fichiers dans le dossier

```
root@test-virtual-machine:/etc/haproxy# ls
errors haproxy.cfg haproxy.cfg.save
root@test-virtual-machine:/etc/haproxy#
```

Nous allons modifier notre haproxy.cfg(les .cfg sont des dossier de configuration) et pour modifier un fichier nous pouvons utiliser plusieurs commande mais nous allons utiliser la commande nano.

```
oot@test-virtual-machine:/etc/haproxy# nano haproxy.cfg
```

Après avoir taper la commande il nous fait rentrer dans le dossier de configuration d'HAproxy

```
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
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

    # Default SSL material locations
    ca-base /etc/ssl/certs
    crt-base /etc/ssl/private

    # See: https://ssl-config.mozilla.org/#server=haproxy&server
    ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDE
    ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES
    ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets
```

Dans cette image ci-dessus nous pouvons voir nos certificats.

```
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
```

```
6LL0L1776 204 \67c\µ9bloxλ\6LL0L2\204.µ7cb
6LL0L1776 203 \67c\µ9bloxλ\6LL0L2\203.µ7cb
6LL0L1776 205 \67c\µ9bloxλ\6LL0L2\205.µ7cb
```

Dans l'image ci-dessus nous pouvons voir les fichiers d'erreur. Imaginons que nous tapons notre adresse IP de l'un de nos serveurs, si il ne réponds il nous affiche les erreurs que vous voyez ci-dessus comme erreur 400 /403/408/500/502/503/504.

Une fois nos messages d'erreur configurés nous allons configurer nos serveurs d'application web.

Pour ça nous devons déclarer nos serveurs avec backend nom\_du\_server/service le mode utilisé en l'occurrence http, roundrobin pour l'équilibrage de charge puis server suivi du nom puis après l'IP du serveur concerné. Il ne faut pas oublier le check à la fin de chaque IP.

```

backend glpi_backend
    mode http
    balance roundrobin
    server glpi_server 192.168.10.9:80 check

backend nagios_backend
    mode http
    balance roundrobin
    server nagios_server 192.168.10.50:80 check

backend HAproxy_backend
    mode http
    option forwardfor
    balance roundrobin
    server HAproxy_server 192.168.10.70:9999 check

backend xivo_backend
    mode http
    balance roundrobin
    server xivo_server 192.168.12.3:80 check

```

```

26LVL6L xfl0~26LVL6L 1A5*108*15*3:80 CMECK
p9f9uCG6 L0nuqLopfu
mode http
p9ck6uq xfl0~p9ck6uq

```

Une fois tous nos server déclarer tous nos server , nous allons configurer notre cluster web

Avec les options ci-dessous

```

listen cluster_web
    bind 192.168.10.70:1000
    mode http
    option httpclose
    option forwardfor
    balance roundrobin
    server web1 192.168.10.9:80 check
    server web2 192.168.10.50:80 check
    server web3 192.168.12.8:80 check

```

```

26LVL6L M6P3 1A5*108*15*8:80 CMECK
26LVL6L M6P5 1A5*108*10*20:80 CMECK

```

Une fois notre cluster web configurer, Il reste un dernière parti qui est pour accédez à notre pages HAproxy

```
listen stats
bind *:9999
stats enable
stats hide-version
stats refresh 30s
stats auth admin:admin
stats uri /stats
```

Bind\*= port utiliser

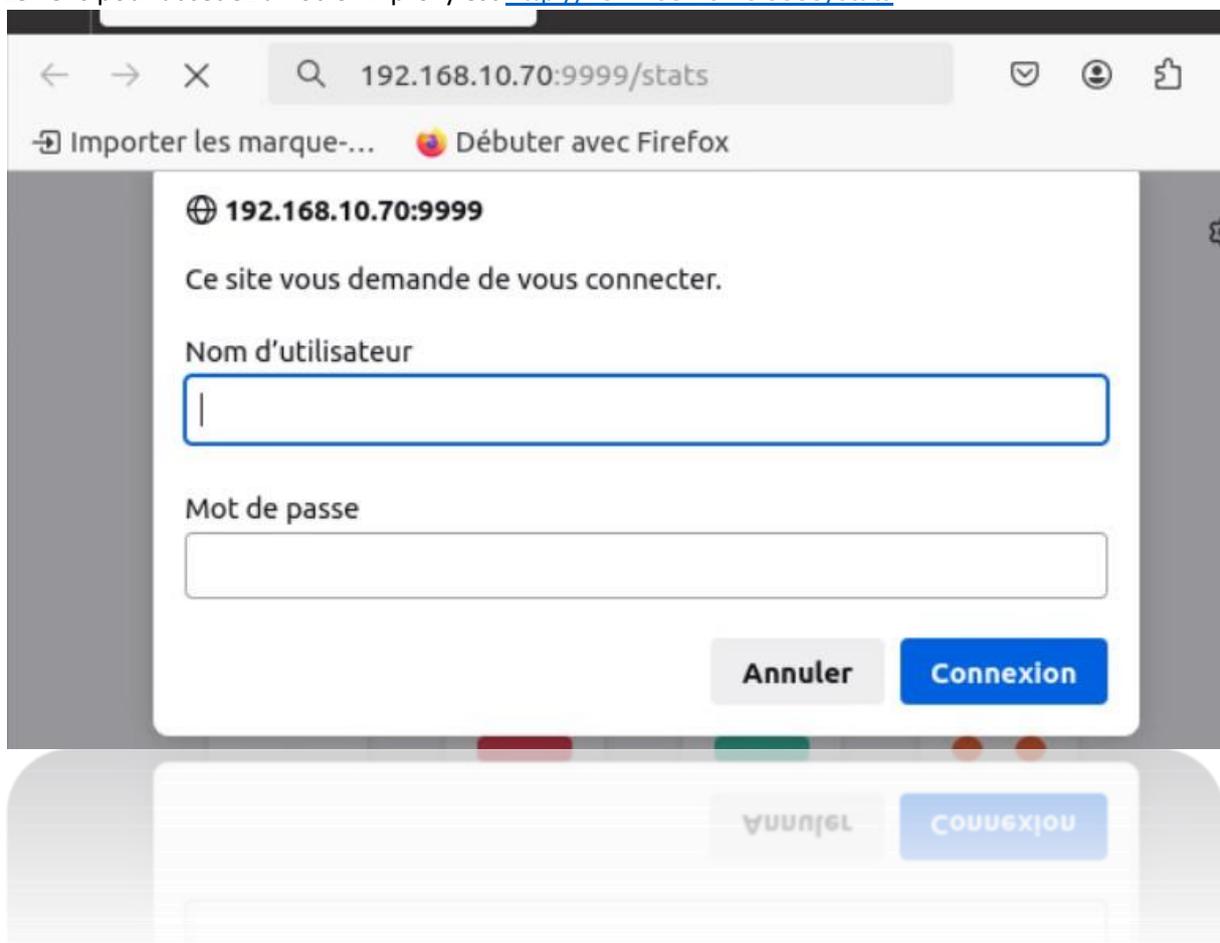
Stats enable = stats activer

Stats refresh 30s = toute les 30 sec il se mets a jour

Stats auth = user / password

Stats uri /stats = liens

Le liens pour accédez a notre HAproxy est <http://192.168.10.70:9999/stats>



Il nous demande le user et mot de passe en l'occurrence admin : admin

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server										
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrble	
<b>gpi_backend</b>																															
gpi_server	0	0	-	0	0		0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	37m30s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	1	0	0	?	0	0	0	0	0	0	0	0	0	0	37m30s UP		1	1	0		0	0s	
<b>nagios_backend</b>																															
nagios_server	0	0	-	0	0		0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	37m30s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	1	0	0	?	0	0	0	0	0	0	0	0	0	0	37m30s UP		1	1	0		0	0s	
<b>H4proxy_backend</b>																															
H4proxy_server	0	0	-	0	0		0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	37m30s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	1	0	0	?	0	0	0	0	0	0	0	0	0	0	37m30s UP		1	1	0		0	0s	
<b>xivo_backend</b>																															
xivo_server	0	0	-	0	0		0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	37m28s DOWN	L4TOUT in 2001ms	1	Y	-	1	1	37m28s	-
Backend	0	0		0	0		0	0	1	0	0	?	0	0	0	0	0	0	0	0	0	0	37m28s DOWN		0	0	0		1	37m28s	
<b>nas_backend</b>																															
nas_server	0	0	-	0	0		0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	37m28s DOWN	* L4TOUT in 2001ms	1	Y	-	1	1	37m28s	-
Backend	0	0		0	0		0	0	1	0	0	?	0	0	0	0	0	0	0	0	0	0	37m28s DOWN		0	0	0		1	37m28s	
<b>server_backend</b>																															
server_server	0	0	-	0	0		0	0	-	0	0	?	0	0	0	0	0	0	0	0	0	0	37m28s DOWN	* L4TOUT in 2001ms	1	Y	-	1	1	37m28s	-
Backend	0	0		0	0		0	0	1	0	0	?	0	0	0	0	0	0	0	0	0	0	37m28s DOWN		0	0	0		1	37m28s	

Une fois sur l'interface nous voyons le quel de nos server si le server est vert ca veut dire que le service / application web fonctionne L4OK signifie Layer 4 ok .