ETTORI Bastien	**BTS SIO 2ème année**
20 Septembre 2016	**Année scolaire : 2016/2017**
Option : SISR	**Version 1**

SERVEUR HAPROXY DEBIAN

SOMMAIRE :

I) Objectif……2

II) Prérequis……2

III) Définition……2

IV) Installation du service « haproxy »……2-3

V) Configuration et tests du service « haproxy »……3-6

VI) Tests d'exécution des serveurs Web……6-7

VII) Conclusion……7

ETTORI Bastien	**BTS SIO 2ème année**
20 Septembre 2016	**Année scolaire : 2016/2017**
Option : SISR	**Version 1**

I) Objectif

Dans cette procédure, nous allons montrer comment installer et configurer un serveur de répartition de charges **HAProxy** sous Debian.

II) Prérequis

Pour réaliser cette procédure, nous avons besoin des éléments suivants :

- Réseau IP **principal** : **192.168.1.0** /24
- Réseau IP **privé** (interne) : **10.0.0.0** /24

OS	Distribution	Version	C/S	Nom du serveur HAProxy	Adresse IP du serveur HAProxy
Debian Jessie	Linux	8.5	**S**	HAProxy	192.168.1.132 /24

IP virtuelle du serveur HAProxy	Nom du serveur web 1	Adresse IP serveur web 1	Nom du serveur web 2	Adresse IP du serveur web 2
10.0.0.132 /24	lab	10.0.0.133 /24	hdlab	10.0.0.134 /24

Caractéristiques des cartes réseau des différents serveurs :

Serveur HAProxy	Serveur Web 1	Serveur Web 2
2 cartes réseau : - 1 en accès par pont - 1 en réseau interne	1 carte réseau en mode « **Réseau interne** »	1 carte réseau en mode « **Réseau interne** »

III) Définition

Le service **HAProxy** permet de faire de la répartition des charges (**Load Balancing**) entres différents serveurs (notamment ici, ce sera pour des serveurs Web). La répartition de charges représente un ensemble de techniques qui distribue une charge de travail sur plusieurs serveurs. Celle-ci assure une haute disponibilité entre eux et donc diminue l'indisponibilité d'un ou plusieurs services.

IV) Installation du service « haproxy »

- Tout d'abord, nous mettons à jour les paquets sur les 3 serveurs (**HAProxy** et les 2 serveurs Web) :

« **apt-get update** ».

Remarque : Si la mise à jour des paquets ne fonctionne pas, nous laissons la configuration **TCP/IP** principal en **DHCP** et ensuite, nous configurons les machines sur le réseau IP privé.

- Ensuite, nous installons le service « **apache2** » sur les 2 serveurs Web (ne pas installer « **apache2** » sur le serveur **HAProxy**) :

« **apt-get install apache2** ».

- Nous éditons le fichier « **/etc/apt/sources.list** » contenant les miroirs et ajoutons cette ligne permettant le téléchargement du service **HAProxy** :

```
deb http://ftp.fr.debian.org/debian/ jessie-backports main
```

- Nous installons le service « **haproxy** » :

```
root@HAProxy:~# apt-get install haproxy -t jessie-backports
```

- Pour vérifier la version d'**HAProxy**, nous tapons la commande suivante (Ici, sa version est « **1.6.9** » :

```
root@HAProxy:~# haproxy -v
HA-Proxy version 1.6.9 2016/08/30
Copyright 2000-2016 Willy Tarreau <willy@haproxy.org>
```

- Nous configurons la nouvelle interface nommée « **eth1** » du serveur **HAProxy** :

```
eth1      Link encap:Ethernet  HWaddr 08:00:27:a6:10:fd
          inet adr:10.0.0.132  Bcast:10.0.0.255  Masque:255.255.255.0
          adr inet6: fe80::a00:27ff:fea6:10fd/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:30 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:5944 (5.8 KiB)  TX bytes:2272 (2.2 KiB)
```

- Nous éditons le fichier « **/etc/hosts** » et ajoutons les 2 serveurs Web :

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

127.0.0.1       localhost
192.168.1.132   HAProxy
10.0.0.133      lab
10.0.0.134      hdllab
```

V) Configuration et tests du service « haproxy »

- Nous ouvrons le fichier « **/etc/haproxy/haproxy.cfg** » pour configurer **HAProxy** :

```
root@HAProxy:~# nano /etc/haproxy/haproxy.cfg
```

- Nous saisissons le contenu en jaune à la fin du fichier :

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

      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

      listen HAProxy
      bind *:80
      balance roundrobin
      option httpclose
      server lab 10.0.0.133:80 check
      server hdlab 10.0.0.134:80 check

      stats uri /statsHaproxy
      stats auth haproxy:haproxy
      stats refresh 30s
```

Descriptions des lignes saisies :

- ⇨ « **Listen HAProxyGSB** » : Ecoute sur le serveur **HAProxy**.
- ⇨ « **bind *:80** » : Port d'écoute d'un ou plusieurs serveurs Web via un navigateur Web.
- ⇨ « **balance roundrobin** » : Spécification de l'équilibrage de charges.
- ⇨ « **option httpclose** » : Option permettant la déconnexion au serveur Web après la réception d'une réponse du client.
- ⇨ « **server Web1GSB 10.0.0.133:80 check** » : Prise en compte et vérification de l'exécution et de la présence du serveur Web « **lab** » visible sur l'interface Web **HAProxy**.
- ⇨ « **server Web2GSB 10.0.0.134:80 check** » : Prise en compte et vérification de l'exécution et de la présence du serveur Web « **hdlab** » visible sur l'interface Web **HAProxy**.
- ⇨ « **stats uri /statsHaproxy** » : Visualisation de l'état des serveurs.
- ⇨ « **stats auth haproxy:haproxy** » : Les identifiants de connexions à l'interface Web **HAProxy** dont le **premier** « **haproxy** » représente le nom d'utilisateur et le **second** « **haproxy** » représente le mot de passe.
- ⇨ « **stats refresh 30s** » : Activation de l'actualisation automatique des statistiques des serveurs à partir de 30 secondes.

- Nous testons si la configuration d'**HAProxy** est valide :

```
root@HAProxy:~# haproxy -c -f /etc/haproxy/haproxy.cfg
Configuration file is valid
root@HAProxy:~# _
```

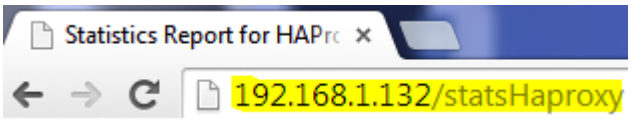
- Nous redémarrons le service « **haproxy** » :

```
root@HAProxy:~# systemctl restart haproxy.service
root@HAProxy:~# _
```

- Nous vérifions si le service « **haproxy** » est bien démarré :

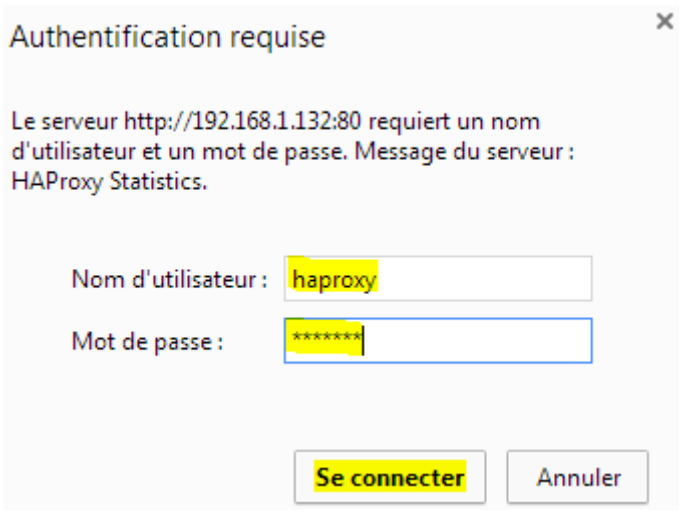
```
root@HAProxy:~# systemctl status haproxy.service
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled)
   Active: active (running) since lun. 2016-09-19 11:30:13 CEST; 7min ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
  Process: 17168 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -c -q $EXTRAOPTS (cod
e=exited, status=0/SUCCESS)
 Main PID: 17169 (haproxy-systemd)
   CGroup: /system.slice/haproxy.service
           ├─17169 /usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy....
           ├─17172 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/hapr...
           └─17173 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/hapr...

sept. 19 11:30:13 HAProxy haproxy-systemd-wrapper[17169]: haproxy-systemd-wra...
sept. 19 11:30:13 HAProxy haproxy[17172]: Proxy HAProxy started.
sept. 19 11:30:13 HAProxy haproxy[17172]: Proxy HAProxy started.
Hint: Some lines were ellipsized, use -l to show in full.
root@HAProxy:~# _
```

- Maintenant, nous ouvrons un navigateur et vérifions qu'**HAProxy** fonctionne en tapant l'URL « **IP_serveurHAProxy/statsHaproxy** » :

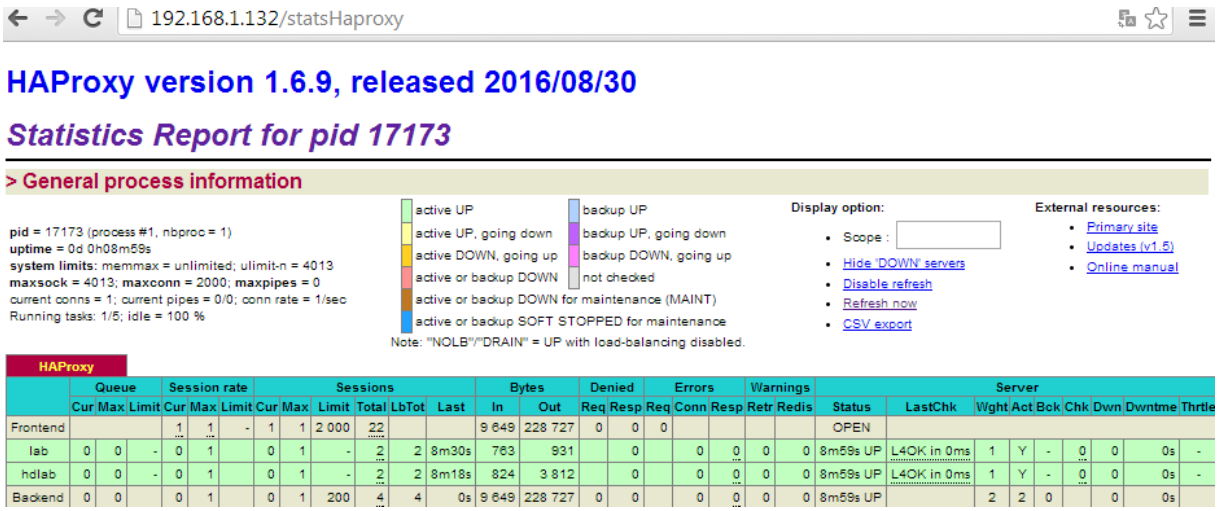


- Nous nous connectons via les identifiants que nous avons définis dans le fichier « **/etc/haproxy/haproxy.cfg** » (« **stats auth haproxy:haproxy** ») :

  ⇨ Le **premier** « **haproxy** » représente le nom d'utilisateur.
  ⇨ Le **second** « **haproxy** » représente le mot de passe.



- Et, nous accédons à l'interface d'**HAProxy** :



Nous pouvons constater que sur l'interface Web, nous voyons les 2 serveurs Web intégrés au service **HAProxy** qui sont en exécution.

VI) Tests d'exécution des serveurs Web

- Tout d'abord, nous éditons les fichiers **HTML** des serveurs Web pour savoir quel serveur répond :

  ⇨ Voici le contenu du fichier **HTML** du serveur Web **principal** :

```
  GNU nano 2.2.6        Fichier : /var/www/html/index.html

<h1>Serveur Web 1 "lab"</h1>
```

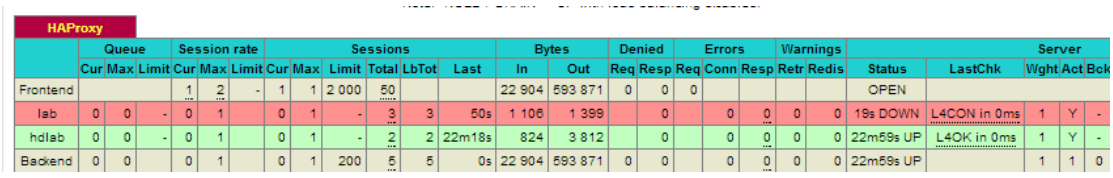
  ⇨ Voici son contenu pour le serveur Web **secondaire** :

```
  GNU nano 2.2.6        Fichier : /var/www/html/index.html

<h1>Serveur Web 2 "hdlab"</h1>
```

- Nous éteignons le service « **apache2** » du serveur Web **principal** « **lab** » :

```
root@lab:~# systemctl stop apache2.service
root@lab:~# _
```

- Nous constatons que sur l'interface d'**HAProxy**, le serveur « **lab** » (principal) est bien éteint et le serveur « **hdlab** » (secondaire) reste disponible :



- Nous faisons un test de connexion sur le serveur **HAProxy** et constatons que le serveur Web **secondaire** répond :

- Nous faisons de même pour le **second** serveur Web « **hdlab** » :

```
root@hdlab:~# systemctl stop apache2.service
root@hdlab:~# _
```

- Nous démarrons à nouveau le **premier** serveur Web « **lab** » :

```
root@lab:~# systemctl start apache2.service
root@lab:~# _
```

- Nous constatons que le serveur « **hdlab** » est bien éteint et que le serveur « **lab** » est de nouveau en exécution :

HAProxy

	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
Frontend				1	2	-	1	1	2 000	17			7 034	155 202	0	0	0					OPEN				
lab	0	0	-	0	1		0	1	-	1	1	14m58s	343	468		0		0	0	0	0	2m4s UP	L4OK in 0ms	1	Y	-
hdlab	0	0	-	0	2		0	1	-	2	2	4m25s	752	745		0		0	0	0	0	5s DOWN	L4CON in 0ms	1	Y	-
Backend	0	0		0	2		0	1	200	3	3	0s	7 034	155 202	0	0		0	0	0	0	15m57s UP		1	1	0

- Nous faisons un test de connexion sur le serveur **HAProxy** et constatons que le serveur Web **principal** répond de nouveau :

VII) Conclusion

En conclusion, nous pouvons dire que le service **HAProxy** est fonctionnel car celui-ci permet de répartir la charge de travail entre les serveurs Web et de faire de la tolérance de pannes entre ces serveurs.