ETTORI Bastien	**BTS SIO 2ème année**
19 Mai 2017	**Année scolaire : 2016/2017**
Option : SISR	**Version 2**

PROCEDURE E4 : SERVEUR HAPROXY DEBIAN

SOMMAIRE :

ETTORI Bastien	BTS SIO 2ème année
19 Mai 2017	Année scolaire : 2016/2017
Option : SISR	Version 2

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	Linux	8.5	**S**	HAProxyGSB	192.168.1.132 /24

IP virtuelle du serveur HAProxy	Nom du premier serveur web	Adresse IP du serveur web 1	Nom du second serveur web	Adresse IP du serveur web 2
10.0.0.132 /24	Web1GSB	10.0.0.133 /24	Web2GSB	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 et l'installation du service « **apache2** » ne fonctionnent pas, nous laissons la configuration **TCP/IP** principal en **DHCP** et ensuite, nous configurons les machines sur le réseau IP interne.

- 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 mettons à jour les paquets à nouveau :

```
root@HAProxyGSB:~# apt-get update
```

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

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

- Pour visualiser la version d'**HAProxy**, nous tapons la commande :

```
root@HAProxyGSB:~# haproxy -v
HA-Proxy version 1.7.3-1~bpo8+1 2017/03/02
Copyright 2000-2017 Willy Tarreau <willy@haproxy.org>
```

Nous constatons que la version d'**HAProxy** est « **1.7.3** ».

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

```
root@HAProxyGSB:~# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 08:00:27:5d:d5:ac
          inet adr:10.0.0.132  Bcast:10.0.0.255  Masque:255.255.255.0
          adr inet6: fe80::a00:27ff:fe5d:d5ac/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:707 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:0 (0.0 B)  TX bytes:42588 (41.5 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   HAProxyGSB
10.0.0.133      Web1GSB
10.0.0.134      Web2GSB
```

V) Configuration et test du service « haproxy »

- Nous ouvrons le fichier « **/etc/haproxy/haproxy.cfg** » pour configurer le service **HAProxy** en ajoutant les lignes suivantes à la fin du fichier :

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 « **Web1GSB** » 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 « **Web2GSB** » 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.

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

      errorfile 502 /etc/haproxy/errors/502.http
      errorfile 503 /etc/haproxy/errors/503.http
      errorfile 504 /etc/haproxy/errors/504.http

      listen HAProxyGSB
      bind *:80
      balance roundrobin
      option httpclose
      server Web1GSB 10.0.0.133:80 check
      server Web2GSB 10.0.0.134:80 check

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

- Nous testons si la configuration d'**HAProxy** est valide et constatons que c'est le cas :

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

- Nous redémarrons le service « **haproxy** » pour prendre en compte les modifications :

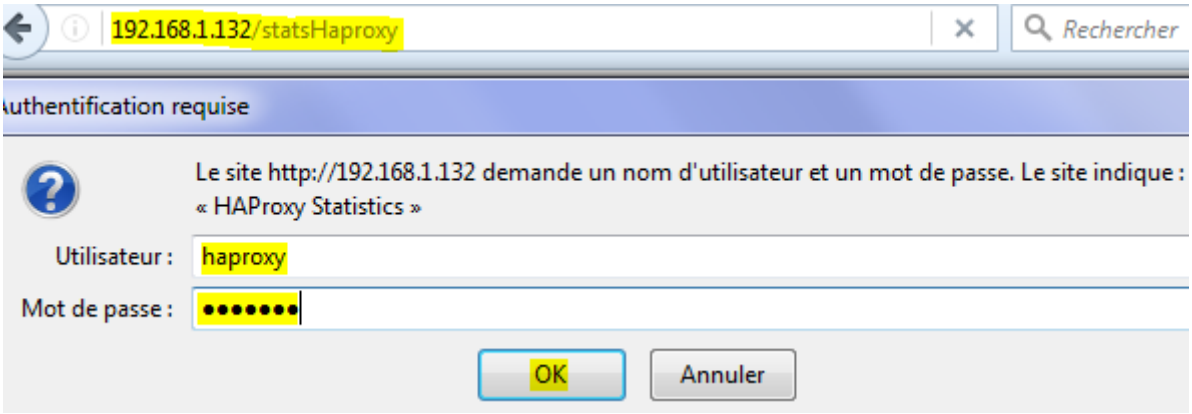
```
root@HAProxyGSB:~# systemctl restart haproxy.service
root@HAProxyGSB:~#
```

- Nous vérifions si le service « **haproxy** » est bien démarré et constatons que c'est le cas :

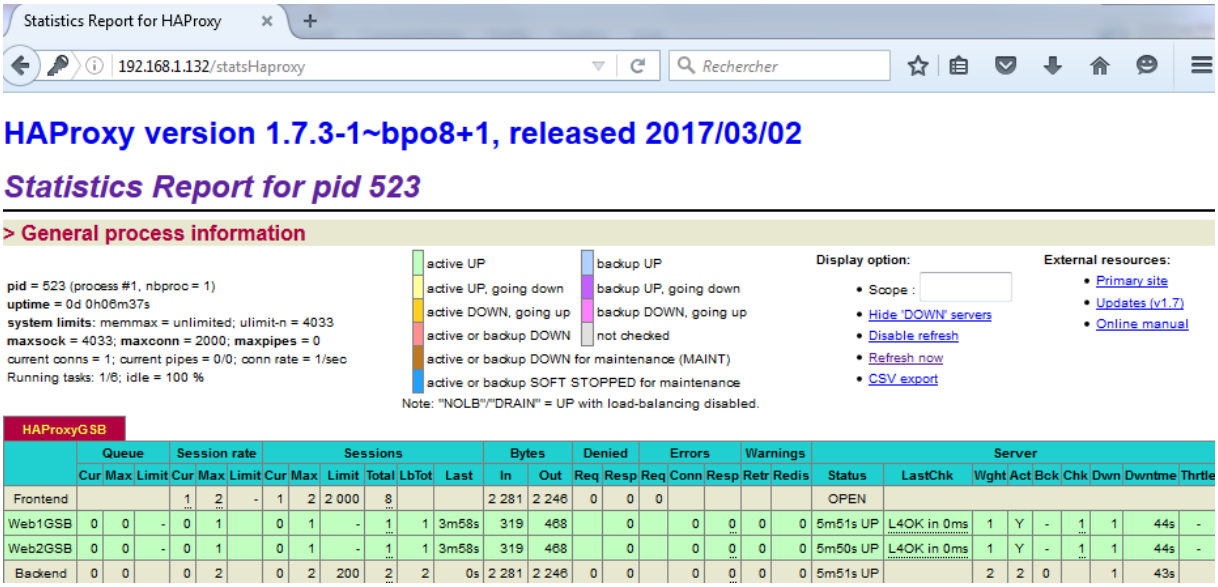
```
root@HAProxyGSB:~# systemctl status haproxy.service
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled)
   Active: active (running) since lun. 2017-04-03 08:36:12 CEST; 18s
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
  Process: 868 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -c -q $EXTR
exited, status=0/SUCCESS)
 Main PID: 869 (haproxy-systemd)
   CGroup: /system.slice/haproxy.service
           ├─869 /usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/h
           ├─872 /usr/sbin/haproxy-master
           └─873 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /r

avril 03 08:36:12 HAProxyGSB systemd[1]: Started HAProxy Load Balanc
avril 03 08:36:12 HAProxyGSB haproxy-systemd-wrapper[869]: haproxy-s
avril 03 08:36:12 HAProxyGSB haproxy[872]: Proxy HAProxyGSB started.
avril 03 08:36:12 HAProxyGSB haproxy[872]: Proxy HAProxyGSB started.
Hint: Some lines were ellipsized, use -l to show in full.
root@HAProxyGSB:~#
```

- Maintenant, nous vérifions qu'**HAProxy** fonctionne en tapant l'URL et nous connectons via les identifiants que nous avons définis dans le fichier « **/etc/haproxy/haproxy.cfg** » (« **stats auth haproxy:haproxy** ») :

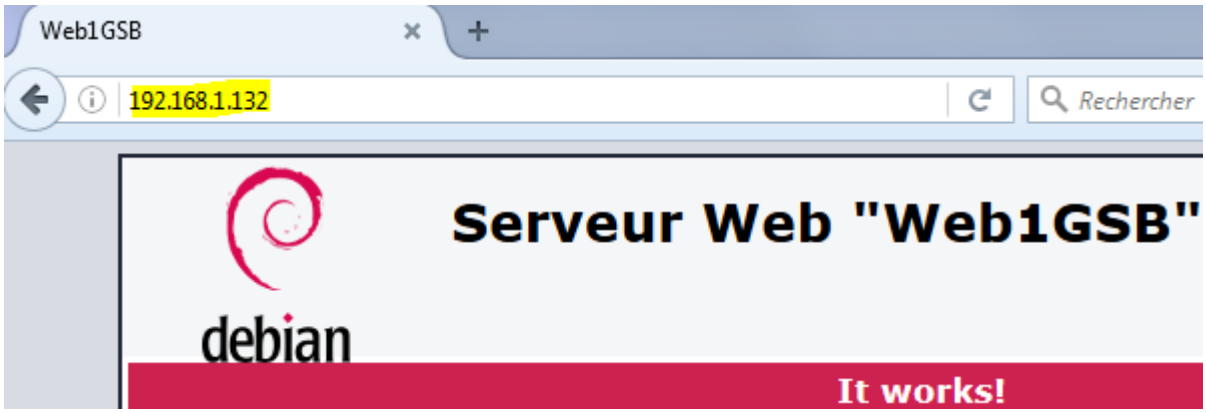


- Et, nous accédons à l'interface Web d'**HAProxy** et constatons la présence des 2 serveurs Web en exécution :



VI) Test d'exécution des serveurs Web

- Tout d'abord, nous tapons l'adresse IP du serveur **HAProxy** et constatons que le serveur Web **principal** répond :



- Ensuite, nous éditons une partie des contenus des fichiers **HTML** respectifs des 2 serveurs Web de **GSB** :

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

        <span class="floating_element">
          Serveur Web "Web2GSB"
        </span>
      </div>
```

- Ensuite, pour tester la tolérance de panne entre les 2 serveurs Web, nous éteignons le service « **apache2** » du serveur Web **principal** « **Web1GSB** » :

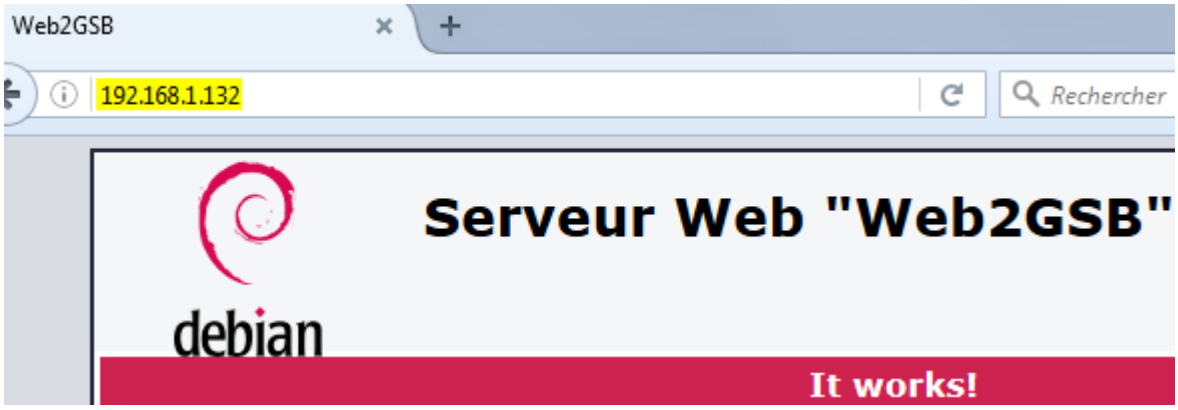
```
root@Web1GSB:~# systemctl stop apache2.service
root@Web1GSB:~# _
```

- Nous constatons que sur l'interface d'**HAProxy**, le serveur « **Web1GSB** » est bien éteint et le serveur « **Web2GSB** » reste disponible :

HAProxyGSB

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk
Frontend				1	2	-	1	2	2 000	26			9 850	96 846	0	0	0					OPEN	
Web1GSB	0	0	-	0	1		0	1	-	10	7	18s	2 908	14 541		0		1	0	3	0	17s DOWN	L4CON in 0ms
Web2GSB	0	0	-	0	2		0	1	-	8	8	15s	3 380	10 868		0		0	0	0	0	8m21s UP	L4OK in 0ms
Backend	0	0		0	2		0	2	200	15	15	0s	9 850	96 846	0	0		1	0	3	0	8m22s UP	

- Nous faisons un test de connexion sur le serveur **HAProxy** en tapant son adresse IP et constatons que le serveur Web **secondaire** répond :



- Nous faisons de même pour le **second** serveur Web « **Web2GSB** » :

```
root@Web2GSB:~# systemctl stop apache2.service
root@Web2GSB:~# _
```

- Nous démarrons à nouveau le **premier** serveur Web « **Web1GSB** » :

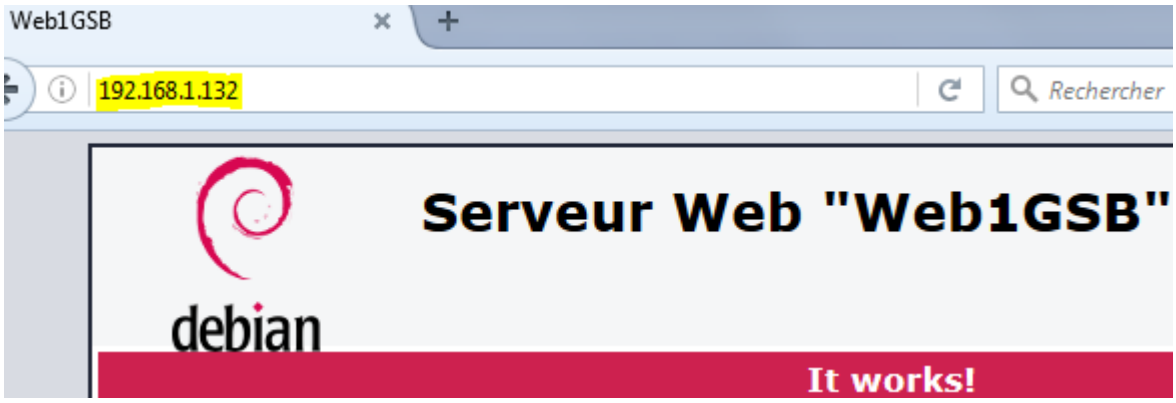
```
root@Web1GSB:~# systemctl start apache2.service
root@Web1GSB:~# _
```

- Nous constatons que le serveur « **Web2GSB** » est bien éteint et que le serveur « **Web1GSB** » est de nouveau en exécution :

HAProxyGSB

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk
Frontend				1	2	-	1	2	2 000	29			11 017	139 186	0	0	0					OPEN	
Web1GSB	0	0	-	0	1		0	1	-	10	7	1m36s	2 908	14 541		0		1	0	3	0	5s UP	L4OK in 0ms
Web2GSB	0	0	-	0	2		0	1	-	8	8	1m33s	3 380	10 868		0		0	0	0	0	8s DOWN	L4CON in 0ms
Backend	0	0		0	2		0	2	200	15	15	0s	11 017	139 186	0	0		1	0	3	0	5s UP	

- 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.