

Haute disponibilité et répartition des charges d'un serveur web avec HAProxy

Savoir-faire

- ❖ Installer et configurer des éléments d'infrastructure
- ❖ Déployer une solution d'infrastructure
- ❖ Administrer un système
- ❖ Automatiser des tâches d'administration
- ❖ Tester l'intégration et l'acceptation d'une solution d'infrastructure
- ❖ Rédiger ou mettre à jour la documentation technique et utilisateur d'une solution d'infrastructure

Contexte professionnel :

M. LAURENT, de la mairie de la ville des Abymes, a fait migrer le site web de la ville vers un système d'exploitation libre (opensource), et l'a hébergé sur l'un des serveurs Linux Debian Bookworm disponible au sein de l'infrastructure réseau et système actuelle.

Ce projet a été mené par un Développeur de C2I Caraïbes, spécialiste dans la création d'applications métiers basées sur les technologies web traditionnelles et 3.0.

Dans un premier temps, M. LAURENT désirait disposer d'un serveur web de secours prêt à prendre le relais si le serveur web principal, hébergeant le site de la ville venait à ne plus être opérationnel.

Finalemment, M. LAURENT souhaite que chacun des serveurs web hébergeant le site web soit utilisé de manière optimale, qu'il n'y en ait pas un surchargé tandis que l'autre serait sous-utilisé.

Après avoir effectué quelques recherches, il a décidé de réaliser la nouvelle infrastructure réseau et système présentée en annexes **document 1**.

Vous disposez, dans la ferme des serveurs, d'une machine virtuelle (et vous êtes en mesure d'en créer d'autres) sur laquelle se trouve le site web opérationnelle en HTTP.

La résolution des noms est prise en charge par un serveur DNS déjà configuré qui a (en interne) autorité sur la zone « **ville-abymes.fr** ».

Votre serveur web du site de la ville des Abymes fait partie de la zone « **ville-abymes.fr** », il est accessible par son nom pleinement qualifié déclaré sur le serveur DNS « **www.ville-abymes.fr** ».

Le site web est quant à lui accessible via l'URL : « **http://www.ville-abymes.fr/index.html** ».

MISSION : Vous êtes stagiaire au sein du service informatique de la ville des Abymes et vous avez pour objectif de construire, étape par étape, une solution totalement opérationnelle répondant au nouveau besoin exprimé par M. LAURENT.

SOMMAIRE

Étape 1 : Clonage et configuration IP du cluster de serveurs web.

Srvweb1	Srvweb2
127.0.0.1 localhost	127.0.0.1 localhost
127.0.1.1 SRVWEB1	127.0.1.1 SRVWEB2
10.0.0.3 SRVWEB2	10.0.0.2 SRVWEB1

Étape 2 : Mise à jour des fichiers de configuration d'Apache 2.0 des deux serveurs web.

- SRVWEB1 : « Bienvenue sur le site n°1 de la ville des Abymes »
- SRVWEB2 : « Bienvenue sur le site n°2 de la ville des Abymes »

Étape 3 : Installation et configuration de HAproxy sur un serveur basé sous Linux Debian nommé « SRVHAPROXY ».

HAProxy, (High Availability Proxy), est une solution de proxy open source et d'équilibrage de charge populaire qui peut être exécutée sur Linux, macOS et FreeBSD. Son utilisation la plus courante consiste à améliorer les performances et la fiabilité d'un environnement serveur en répartissant la charge de travail sur plusieurs serveurs.

HAProxy est une application qui peut agir comme proxy inverse ou proxy direct pour les applications basées sur TCP (couche 4 – modèle OSI) et HTTP (couche 7 – modèle OSI). HAProxy gère efficacement le trafic Web, réduit la charge du serveur et améliore les performances générales des applications en répartissant le trafic entre plusieurs hôtes ou services back-end.

HAProxy agit comme passerelle principale qui gère toutes les demandes réseau externes adressées à vos serveurs principaux qui exécutent des serveurs Web, des bases de données ou des applications de gestion de fichiers.

Étape 4 : Testez la répartition des charges du site web de la ville des Abymes.

Lancez le navigateur web de votre poste de travail et saisissez l'URL suivante :
« <http://192.168.X.4:8404/statshaproxy> ».

Début du PROJET en individuel

Étape 1 : Clonage et configuration IP du cluster de serveurs web

Premièrement pour avoir nos « **srvweb1** » et « **srvweb2** », on clone à partir de notre « **Clone de srvweb** » avec une nouvelle génération d'adresse mac. Puis nous lançons nos deux machines afin de modifier leur nom d'hôte et leur plan d'adressage IP.

Grâce aux commandes suivantes :

« **nano /etc/network/interfaces** » puis « **systemctl restart networking** » pour relancer le service réseau, & « **nano /etc/hostname** » pour vérifier le nom après le redémarrage on entre la commande « **hostname** » :

```
root@srvweb1:~# hostname
srvweb1
root@srvweb1:~#
```

```
root@srvweb2:~# hostname
srvweb2
root@srvweb2:~#
```

```
auto enp0s3
iface enp0s3 inet static
address 10.0.0.2
network 255.255.255.0
```

```
auto enp0s3
iface enp0s3 inet static
address 10.0.0.3
network 255.255.255.0
```

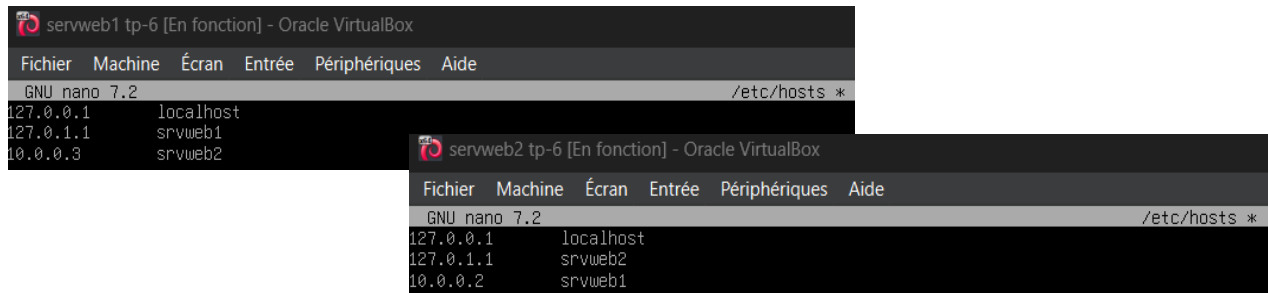
Ensuite pour vérifier qu'ils communiquent via le réseau nous allons les interroger entre-elles la commande « **ping** » :

```
root@srvweb2:~# ping srvweb1
PING srvweb1 (10.0.0.2) 56(84) bytes of data.
64 bytes from srvweb1 (10.0.0.2): icmp_seq=1 ttl=64 time=92.3 ms
64 bytes from srvweb1 (10.0.0.2): icmp_seq=2 ttl=64 time=1.37 ms
64 bytes from srvweb1 (10.0.0.2): icmp_seq=3 ttl=64 time=1.01 ms
64 bytes from srvweb1 (10.0.0.2): icmp_seq=4 ttl=64 time=0.510 ms
^C
--- srvweb1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 0.510/23.793/92.289/39.547 ms
```

```
root@srvweb1:~# ping srvweb2
PING srvweb2 (10.0.0.3) 56(84) bytes of data.
64 bytes from srvweb2 (10.0.0.3): icmp_seq=1 ttl=64 time=46.9 ms
64 bytes from srvweb2 (10.0.0.3): icmp_seq=2 ttl=64 time=0.708 ms
64 bytes from srvweb2 (10.0.0.3): icmp_seq=3 ttl=64 time=0.748 ms
64 bytes from srvweb2 (10.0.0.3): icmp_seq=4 ttl=64 time=0.912 ms
^C
--- srvweb2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.708/12.322/46.920/19.975 ms
```

HAProxy s'appuie sur le nom d'hôte des serveurs participant au cluster afin de garantir la communication entre eux. Alors nous allons configurer le fichier « **/etc/hosts** » pour les deux serveurs comme ceci :

Srvweb1	Srvweb2
127.0.0.1 localhost	127.0.0.1 localhost
127.0.1.1 SRVWEB1	127.0.1.1 SRVWEB2
10.0.0.3 SRVWEB2	10.0.0.2 SRVWEB1



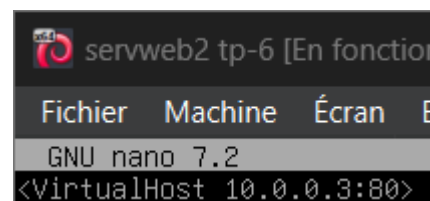
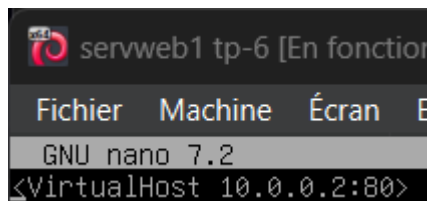
Le fichier « **/etc/hosts** » permet de résoudre localement les noms d'hôtes sans passer par un DNS. Maintenant chaque machine connaît l'IP de l'autre.

```
root@srweb1:~# ping srweb2
PING srweb2 (10.0.0.3) 56(84) bytes of data:
64 bytes from srweb2 (10.0.0.3): icmp_seq=1 ttl=64 time=46.9 ms
64 bytes from srweb2 (10.0.0.3): icmp_seq=2 ttl=64 time=0.708 ms
64 bytes from srweb2 (10.0.0.3): icmp_seq=3 ttl=64 time=0.748 ms
64 bytes from srweb2 (10.0.0.3): icmp_seq=4 ttl=64 time=0.912 ms
^C
--- srweb2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.708/12.322/46.920/19.975 ms
```

```
root@srweb2:~# ping srweb1
PING srweb1 (10.0.0.2) 56(84) bytes of data:
64 bytes from srweb1 (10.0.0.2): icmp_seq=1 ttl=64 time=92.3 ms
64 bytes from srweb1 (10.0.0.2): icmp_seq=2 ttl=64 time=1.37 ms
64 bytes from srweb1 (10.0.0.2): icmp_seq=3 ttl=64 time=1.01 ms
64 bytes from srweb1 (10.0.0.2): icmp_seq=4 ttl=64 time=0.510 ms
^C
--- srweb1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 0.510/23.793/92.289/39.547 ms
```

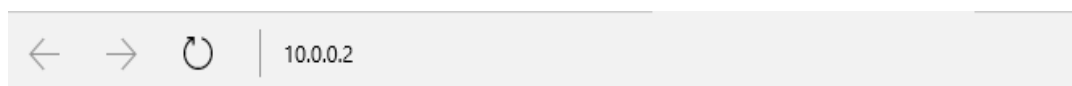
Étape 2 : Mise à jour des fichiers de configuration d'Apache 2.0 des deux serveurs web.

Dans cette étape, il s'agira de revoir les configurations du fichier « **nano /etc/apache2/sites-available/villeabymes.conf** » et modifier l'adresse IP pour chaque serveur web.



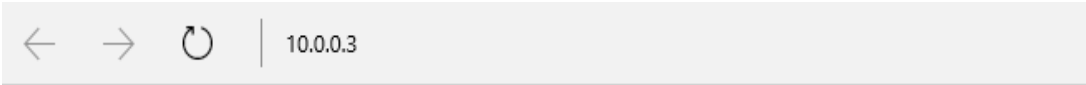
Sur chaque serveur web, on modifie la page d'accueil avec la commande « **nano /var/www/villeabymes/index.html** » du site de la ville des Abymes, comme suit :

- SRVWEB1 : « Bienvenue sur le site n°1 de la ville des Abymes »



Bienvenue sur le site n°1 de la ville des Abymes

- SRVWEB2 : « Bienvenue sur le site n°2 de la ville des Abymes »



Bienvenue sur le site n°2 de la ville des Abymes

Étape 3 : Installation et configuration de HAproxy sur un serveur basé sous Linux Debian nommé « SRVHAPROXY ».

HAProxy est une solution de proxy open source et d'équilibrage de charge populaire qui peut être exécutée sur Linux, macOS et FreeBSD. Son utilisation la plus courante consiste à améliorer les performances et la fiabilité d'un environnement serveur en répartissant la charge de travail sur plusieurs serveurs.

Pour cette étape nous allons à nouveau cloner une VM à partir de notre « **Clone de srvweb** » avec une nouvelle génération d'adresse MAC et le renommer « **srvhaproxy** ».

Puis au lancement on modifie son nom d'hôte et son plan d'adressage IP, grâce aux commandes : « **nano /etc/network/interfaces** » puis « **systemctl restart networking** » pour relancer le service réseau, & « **nano /etc/hostname** » pour vérifier le nom après le redémarrage on entre la commande « **hostname** » :

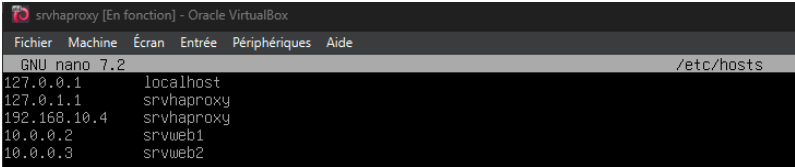
```
root@srvhaproxy:~# hostname
srvhaproxy
root@srvhaproxy:~#
```

```
# The primary network interface
auto enp0s3
iface enp0s3 inet static
address 192.168.10.4
network 255.255.255.0
gateway 192.168.10.1

#The second network interface
auto enp0s8
iface enp0s8 inet static
address 10.0.0.1
network 255.255.255.0
```

HAProxy s'appuie sur le nom d'hôte des serveurs participant au cluster afin de garantir la communication entre eux. Alors nous allons configurer le fichier « **/etc/hosts** » pour les deux serveurs comme ceci :

SRVHAPROXY	
127.0.0.1	localhost
127.0.1.1	SRVHAPROXY
192.168.10.4	SRVHAPROXY
10.0.0.2	SRVWEB1
10.0.0.3	SRVWEB2



Sur ce serveur on installe **HAProxy** avec la commande « **apt install haproxy -y** ». Ensuite on active le service avec « **systemctl enable haproxy** ». Enfin pour vérifier l'état du service on saisit « **systemctl status haproxy** ».

On constate que le service n'est pas activé car le fichier permettant de configurer **HAProxy** n'est pas encore paramétré. Mais avant d'accéder au fichier « **/etc/haproxy/haproxy.cfg** », on va créer une copie de ce fichier du fichier « **haproxy.cfg** » en « **haproxy.cfg.save** », à l'aide de la commande :

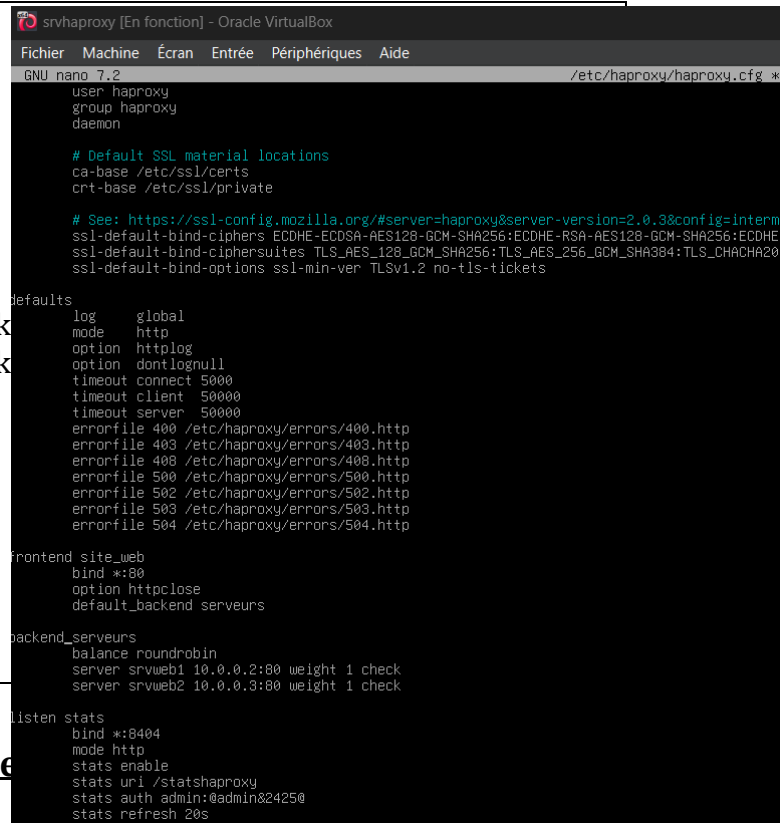
« **cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.save** »

Par la suite on accède au fichier de configuration « **haproxy.cfg** » et ajouter les sections et paramètres à la fin de ce dernier :

```
...
frontend site_web
    bind *:80
    option httpclose
    default_backend serveurs

backend serveurs
    balance roundrobin
    server srvweb1 10.0.0.2:80 weight 1 check
    server srvweb2 10.0.0.3:80 weight 1 check

listen stats
    bind *:8404
    mode http
    stats enable
    stats uri /statshaproxy
    stats auth admin:@admin2425@
    stats refresh 20s
```



```
srvhaproxy [En fonction] - Oracle VirtualBox
Fichier  Machine  Écran  Entrée  Périphériques  Aide
GNU nano 7.2 /etc/haproxy/haproxy.cfg
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-version=2.0.3&config=interm
ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20
ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

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

frontend site_web
    bind *:80
    option httpclose
    default_backend serveurs

backend serveurs
    balance roundrobin
    server srvweb1 10.0.0.2:80 weight 1 check
    server srvweb2 10.0.0.3:80 weight 1 check

listen stats
    bind *:8404
    mode http
    stats enable
    stats uri /statshaproxy
    stats auth admin:@admin&2425@
    stats refresh 20s
```

Rôle de chaque section / paramètre

► frontend site_web

- **frontend** : section qui gère les connexions entrantes.
- ***bind :80** : écoute sur **toutes les interfaces réseau**, sur le **port 80** (HTTP).
- **option httpclose** : ferme la connexion HTTP après chaque réponse (mode connexion courte, utile pour certaines compatibilités).
- **default_backend servers** : redirige le trafic vers la section backend appelée **serveurs**.

► backend servers

- **backend** : section qui gère les connexions vers les **serveurs web réels**.
- **balance roundrobin** : répartition du trafic en mode **round-robin** (un serveur après l'autre, tour par tour).
- **server srvweb1 10.0.0.2:80 weight 1 check** :
 - **srvweb1** : nom du serveur.
 - **10.0.0.2:80** : adresse IP et port du serveur web.
 - **10.0.0.3:80** : adresse IP et port du serveur web.
 - **weight 1** : poids du serveur (1 = égalité avec les autres).
 - **check** : active la vérification de l'état du serveur (ping/HTTP check).

► listen stats

- **listen** : mixe frontend + backend pour un service dédié.
- ***bind :8404** : écoute sur le port **8404** (interface de stats).
- **mode http** : communication HTTP.
- **stats enable** : active la page de stats.
- **stats uri /statshaproxy** : URL pour accéder à la page
→ "`http://<IP>:8404/statshaproxy`".
- **stats auth admin:@admin2425@** : identifiants pour se connecter.
- **stats refresh 20s** : rafraîchit la page toutes les 20 secondes.

Après avoir fait ceci, nous allons vérifier la validité du paramétrage à l'aide de la commande « **sudo haproxy -c -f haproxy.cfg** ». On constate que la configuration est bonne :

Configuration file is valid

On relance le service « **haproxy** » avec « **systemctl restart haproxy** » pour la prise en compte des nouveaux paramètres de configuration.

Rôle des options :

- c : **check** → teste la syntaxe de la config sans lancer HAProxy.
- f /etc/haproxy/haproxy.cfg : spécifie le **fichier de config à utiliser**.

Étape 4 : Testez la répartition des charges du site web de la ville des Abymes.

Pour cette dernière étape, nous allons depuis un poste de travail accéder à l'interface web fournissant des informations statistiques sur l'état de fonctionnement du service « **HAProxy** ».

- Sur un navigateur web du poste de travail on saisit l'URL :
`http://192.168.X.4:8404/statshaproxy`.
- À l'aide des éléments d'authentification comme :
 - L'utilisateur : admin
 - Le mot de passe : "`@admin2425@`".

poste1 [En fonction] - Oracle VirtualBox

Fichier Machine Écran Entrée Périphériques Aide

Nous ne pouvons pas atteindre 192.168.10.4:8404/statshaproxy

HAProxy version 2.6.12-1+deb12u1, released 2023/12/16

Statistics Report for pid 875

> General process information

pid = 875 (process #1, nbproc = 1, nbthread = 1)
 uptime = 0d 4h33m09s
 system limits: memmax = unlimited; ulimit-n = 524288
 maxsock = 524288; maxconn = 262124; maxpipes = 0
 current conns = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps
 Running tasks: 0/16; idle = 100 %

active UP backup UP
 active UP, going down backup UP, going down
 active DOWN, going up backup DOWN, going up
 active or backup DOWN not checked
 active or backup DOWN for maintenance (MAINT)
 active or backup SOFT STOPPED for maintenance
 Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
 • Scope:
 • Hide 'DOWN' servers
 • Disable refresh
 • Refresh now
 • CSV export
 • JSON export (schema)

External resources:
 • Primary site
 • Updates (v2.6)
 • Online manual

site_web															
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
0	0	-	0	0	0	0	0	0	262 124	0		0	0	0	0
Frontend															

servers															
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
0	0	-	0	0	0	0	0	0	-	0	?	0	0	0	0
snrweb1															
0	0	-	0	0	0	0	0	0	-	0	?	0	0	0	0
snrweb2															
0	0	-	0	0	0	0	0	0	-	0	?	0	0	0	0
Backend									26 213	0	?	0	0	0	0

stats															
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
1	2	-	1	1	1	262 124	7					54 667	3 626 889	0	0
Frontend															
0	0	-	0	0	0	0	0	0	26 213	0	0s	54 667	3 626 889	0	0
Backend															

Maintenant pour tester le bon fonctionnement de la répartition des charges des requêtes http vers le site web de la ville des Aymes depuis le poste de travail à partir de répartiteur des charges **HAProxy**.

Scénario 1 : Nous allons vérifier que HAProxy répartie les **requêtes HTTP** entre chaque serveur web.

- On accède au site web de la ville des Aymes depuis le poste client via l'URL : **http://192.168.5.4:8080**
- Puis en rafraichissant la page plusieurs fois on constate que les réponses alternent entre le **srvweb1** & **2**.
- Sur le site de la ville on remarque qu'à chaque rafraichissement HAProxy redirige la requête vers **srvweb2**, ce qui affiche "Bienvenue sur le site n°2 de la ville des Aymes". Si on rafraichit à nouveau HAProxy redirige la requête vers **srvweb1**, et affiche "Bienvenue sur le site n°1 de la ville des Aymes".

← → ↻ 192.168.10.4

← → ↻ 192.168.10.4

Bienvenue sur le site n°2 de la ville des Aymes

Bienvenue sur le site n°1 de la ville des Aymes

Scénario 2 : Nous allons simuler une panne sur un des serveurs web soit l'éteignant ou en le désactivant, le but est de vérifier qu'en cas de panne HAProxy redirige bien tout le trafic vers le deuxième serveur web afin de prendre le relais.

1. Pour simuler une panne on va sur un des **serveurs web** puis on l'éteint ou entre la commande « **shutdown -h now** »
2. Puis on accède au site web de la ville des Aymes depuis le poste client via l'URL : <http://192.168.5.4:8080>
3. On observe dans un premier temps via l'URL : <http://192.168.X.4:8404/statshaproxy>
 - Que le **srvweb1** n'est plus fonctionnel et qu'il ne reste que le **srvweb2** donc HAProxy devrait diriger toutes les requêtes vers le **srvweb2**.

servers																														
		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	Own	Downtime
srvweb1	0	0	-	0	3	0	1	-	15	15	19m34s	6 120	8 030	0	0	0	0	0	0	0	0	0	3m DOWN	* L4TOUT in 2002ms	1/1	Y	-	5	1	3m
srvweb2	0	0	-	0	3	0	1	-	18	18	2m45s	7 440	9 408	0	0	0	0	0	0	0	0	0	4h56m UP	L4OK in 1ms	1/1	Y	-	3	1	16s
Backend	0	0	0	6	0	1	28 213	33	33	2m45s	13 560	17 438	0	0	0	0	0	0	0	0	0	0	5h48m UP		1/1	1	0	0	0	0s

4. Dans un deuxième temps via l'URL : <http://192.168.5.4:8080>
 - Que le **srvweb2** a bien pris le relais car le site de la ville des Aymes affiche apprésent "Bienvenue sur le site n°2 de la ville des Aymes".



5. En redémarrant le **srvweb1** les requêtes sont à nouveau réparties entre les deux serveurs web.

Conclusion : Ce PROJET nous a permis de mettre en place une infrastructure de répartition de charge avec **HAProxy**, assurant ainsi une meilleure **disponibilité** et **répartition du trafic** entre deux serveurs web (**srvweb1** et **srvweb2**).

Les tests réalisés ont confirmé le bon fonctionnement de la répartition de charge en mode **round-robin**, avec une distribution alternée des requêtes entre les serveurs. De plus, en simulant une panne d'un serveur, nous avons constaté que HAProxy redirigeait automatiquement les requêtes vers le serveur disponible, garantissant ainsi la **continuité du service**.

FIN !