Table des matières :

Avant-Propos

Compétences :

- A1.1.1 Analyse du cahier des charges d'un service à produire
- A1.2.4 Déterminer des tests nécessaires à la validation d'un service (3)
- A4.1.9 Rédaction d'une documentation technique

Répartition de charge :

HAProxy → 192.168.1.144 : 10.22.100.210

Lab → 10.22.100.212

HDLAB→10.22.100.215

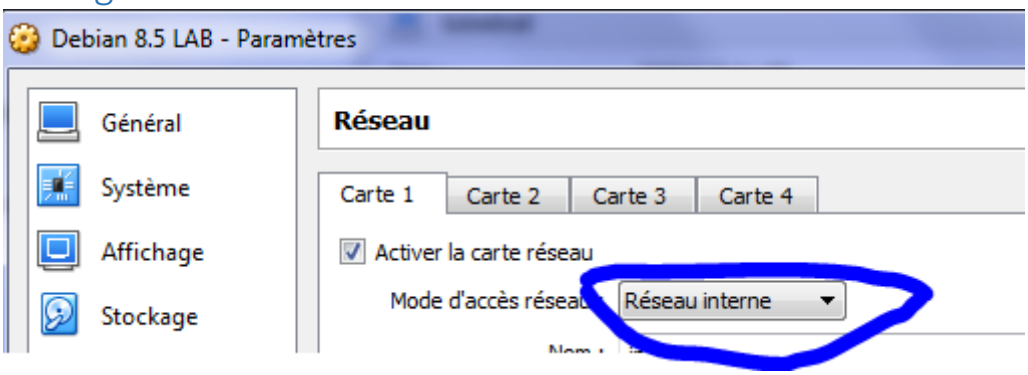
```
# The primary network interface
allow-hotplug eth0
auto eth0:0
iface eth0:0 inet static
        address 192.168.1.144
        netmask 255.255.255.0
        gateway 192.168.1.254

auto eth0:0
```

Objectif :

Dans cette procédure, nous allons montrer comment installer et configurer une répartition de charge sur une plate-forme web sous Debian.

OS	Distribution	Version
Debian	Linux	8.5

Configuration interfaces lab



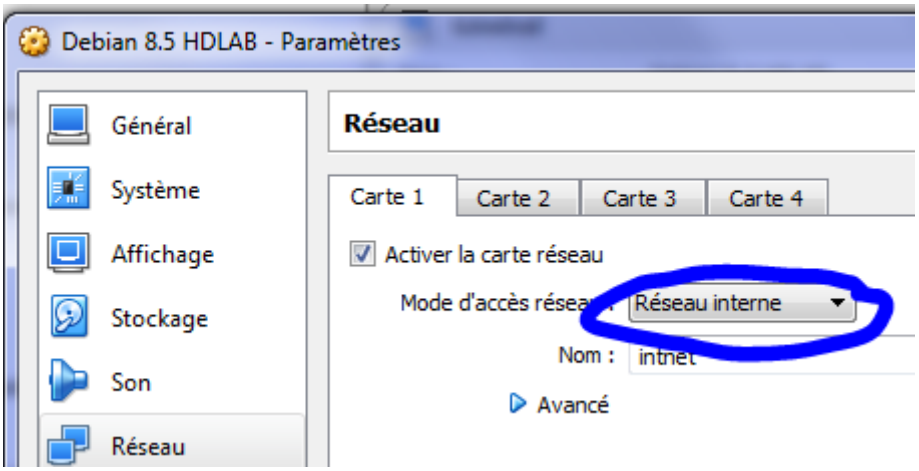

```
  GNU nano 2.2.6                Fichier : hosts

127.0.0.1       localhost
127.0.1.1       mariette

10.22.100.215   hdlab_
```

```
  GNU nano 2.2.6                Fichier : hostname

hdlab_
```

```
  GNU nano 2.2.6        Fichier : /etc/network/interfaces

# This file describes the network interfaces available on
# and how to activate them. For more information, see inte

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 10.22.100.212
netmask 255.255.255.0
```

```
root@lab:~# ifup eth0_
```

Configuration intefaces hdlab



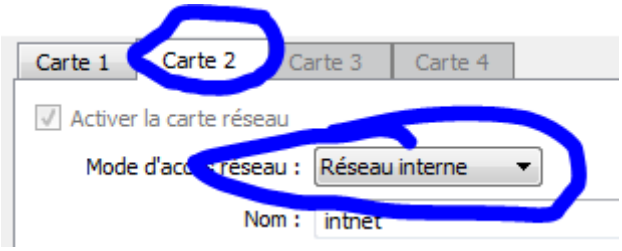

```
  GNU nano 2.2.6              Fichier : /etc/hosts

127.0.0.1       localhost
127.0.1.1       mariette
10.22.100.212   lab_
```

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

hdlab
```

```
  GNU nano 2.2.6        Fichier : /etc/network/interfaces

# This file describes the network interfaces available on
# and how to activate them. For more information, see inte

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
address 10.22.100.215_
netmask 255.255.255.0
```

```
root@hdlab:~# ifup eth0_
```

Configuration intefaces haproxy




```
  GNU nano 2.2.6        Fichier : /etc/network/interfaces

iface eth0 inet static
address 192.168.1.144
netmask 255.255.255.0
gateway 192.168.1.254
# This is an autoconfigured IPv6 interface
iface eth0 inet6 auto

iface eth1 inet static
address 10.22.100.210
netmask 255.255.255.0
_
```

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

127.0.0.1       localhost
127.0.1.1       mariette
192.168.1.144   haproxy_
```

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

haproxy
```

```
root@mariette:/etc/haproxy# ifup eth1_
```

Installation HAproxy

Il faut d'abord ajouter les dépôt de Haproxy pour l'installation qui sont dans source.list

```
 GNU nano 2.2.6          Fichier : /etc/apt/sources.list

#

# deb cdrom:[Debian GNU/Linux 8.5.0 _Jessie_ - Official amd64 CD

deb cdrom:[Debian GNU/Linux 8.5.0 _Jessie_ - Official amd64 CD B

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

Puis

```
root@mariette:~# apt-get update_
```

```
root@mariette:~# apt-get install haproxy -t jessie-backports_
```

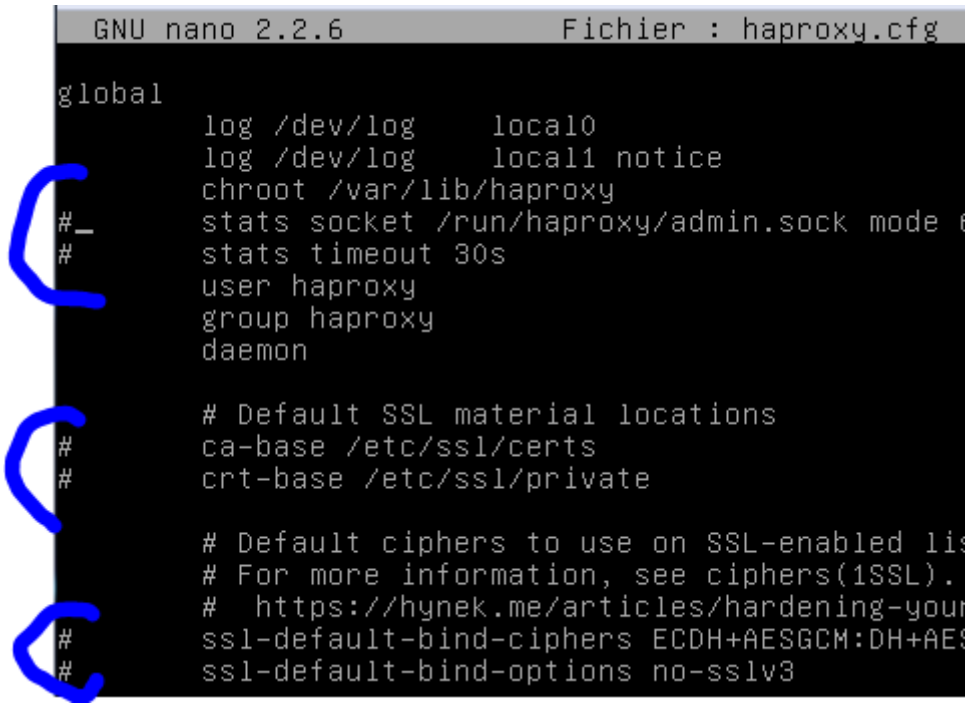
Pour connaitre la version de haproxy :

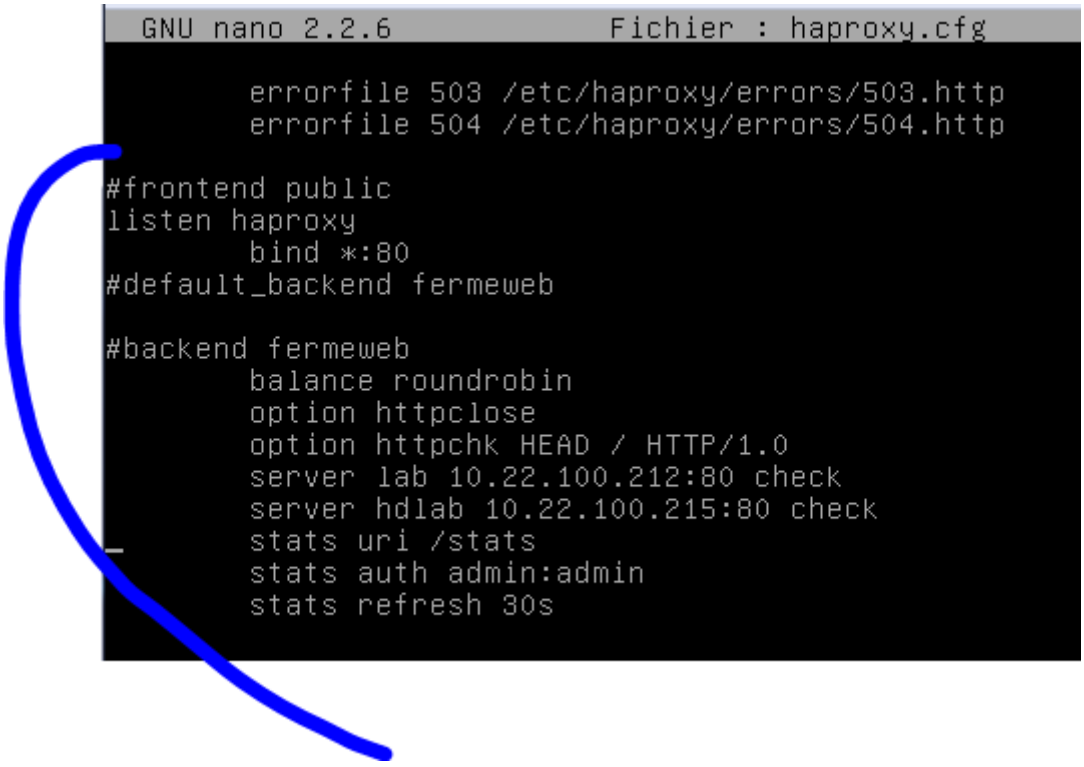
```
root@mariette:~# haproxy -v
HA-Proxy version 1.6.9 2016/08/30
Copyright 2000-2016 Willy Tarreau <willy@haproxy.org>
```

Configuration haproxy

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

Cela permet de verifier la syntaxe du fichier



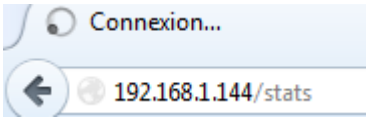


On reste la syntaxe et on redemarre le service

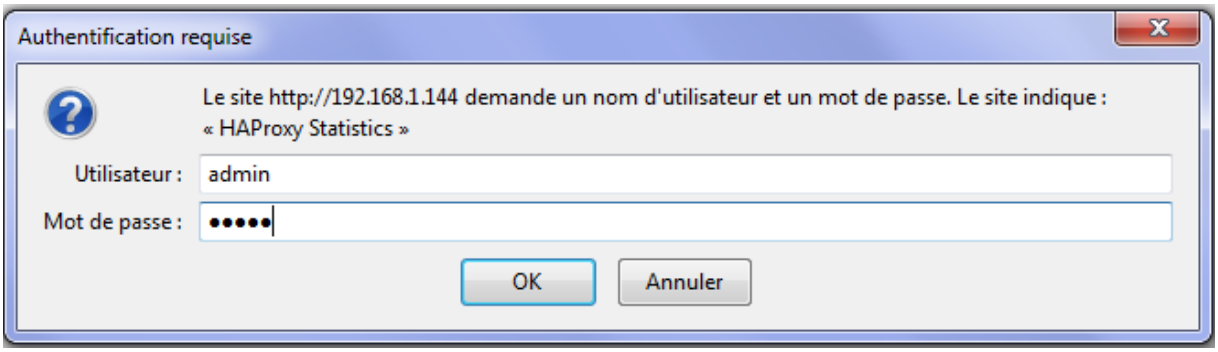
```
root@mariette:~# haproxy -c -f /etc/haproxy/haproxy.cfg
Configuration file is valid
```

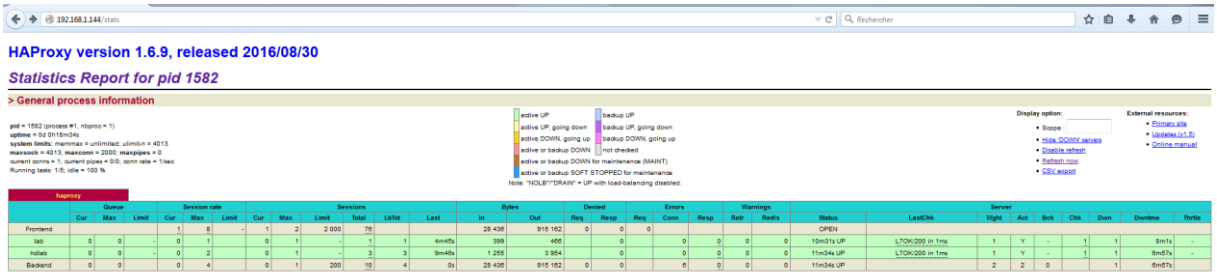
```
root@mariette:/etc/haproxy# service haproxy restart
```

On test ensuite dans un navigateur avec l'adresse de son ip /stats

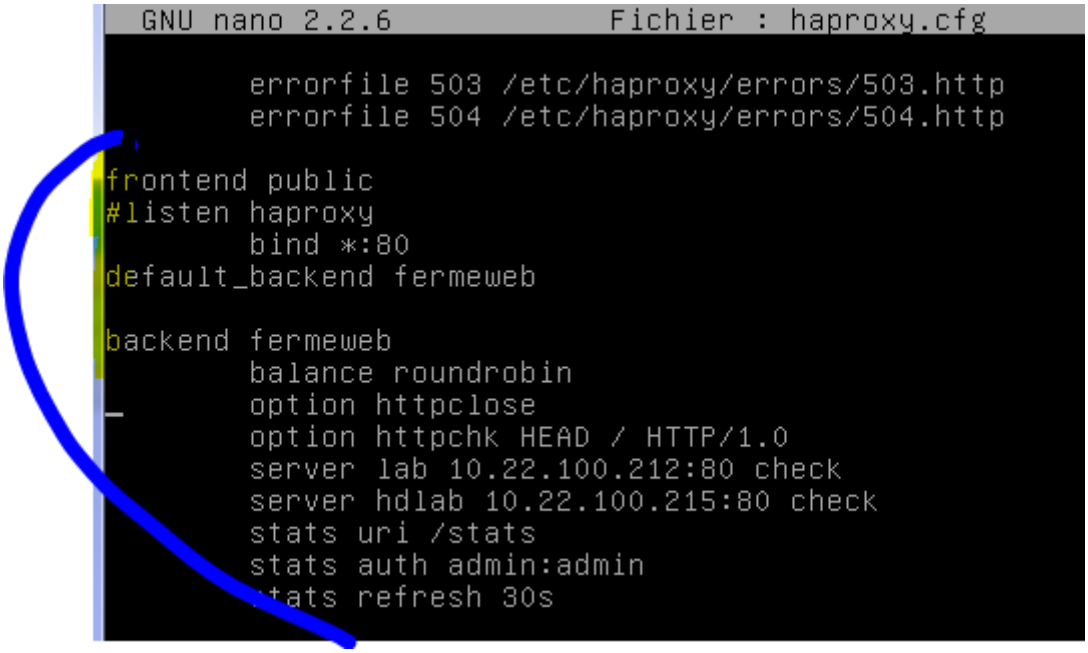


Une boite d'identification s'affiche, il faut utilisé les identifiants « admin » en id et mdp

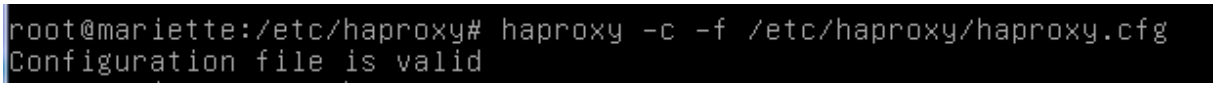




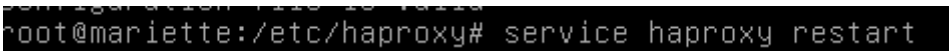
Configuration haproxy avancée



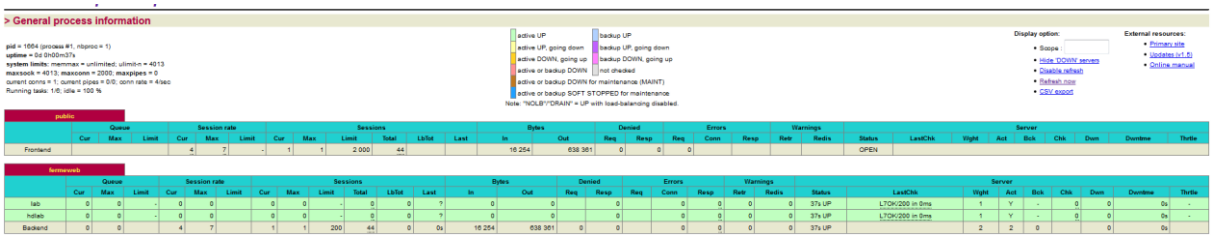
```
  GNU nano 2.2.6                Fichier : haproxy.cfg

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

frontend public
#listen haproxy
        bind *:80
default_backend fermeweb

backend fermeweb
        balance roundrobin
        option httpclose
        option httpchk HEAD / HTTP/1.0
        server lab 10.22.100.212:80 check
        server hdlab 10.22.100.215:80 check
        stats uri /stats
        stats auth admin:admin
        stats refresh 30s
```



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



```
root@mariette:/etc/haproxy# service haproxy restart
```



On regle ensuite la répartition de charge des serveurs en fonction de leur puissance « weight »

```
backend fermeweb
        balance roundrobin
        option httpclose
        option httpchk HEAD / HTTP/1.0
        server lab 10.22.100.212:80 cookie W1 weight 100 check
        server hdlab 10.22.100.215:80 cookie W2 weight_50 check
        stats uri /stats
        stats auth admin:admin
        stats refresh 30s
```

On ajoute aussi les cookies

```
backend fermeweb
        balance roundrobin
        option httpclose
        option httpchk HEAD / HTTP/1.0
        server lab 10.22.100.212:80 cookie W1 weight 100 check
        server hdlab 10.22.100.215:80 cookie W2 weight_50 check
        stats uri /stats
        stats auth admin:admin
        stats refresh 30s
```

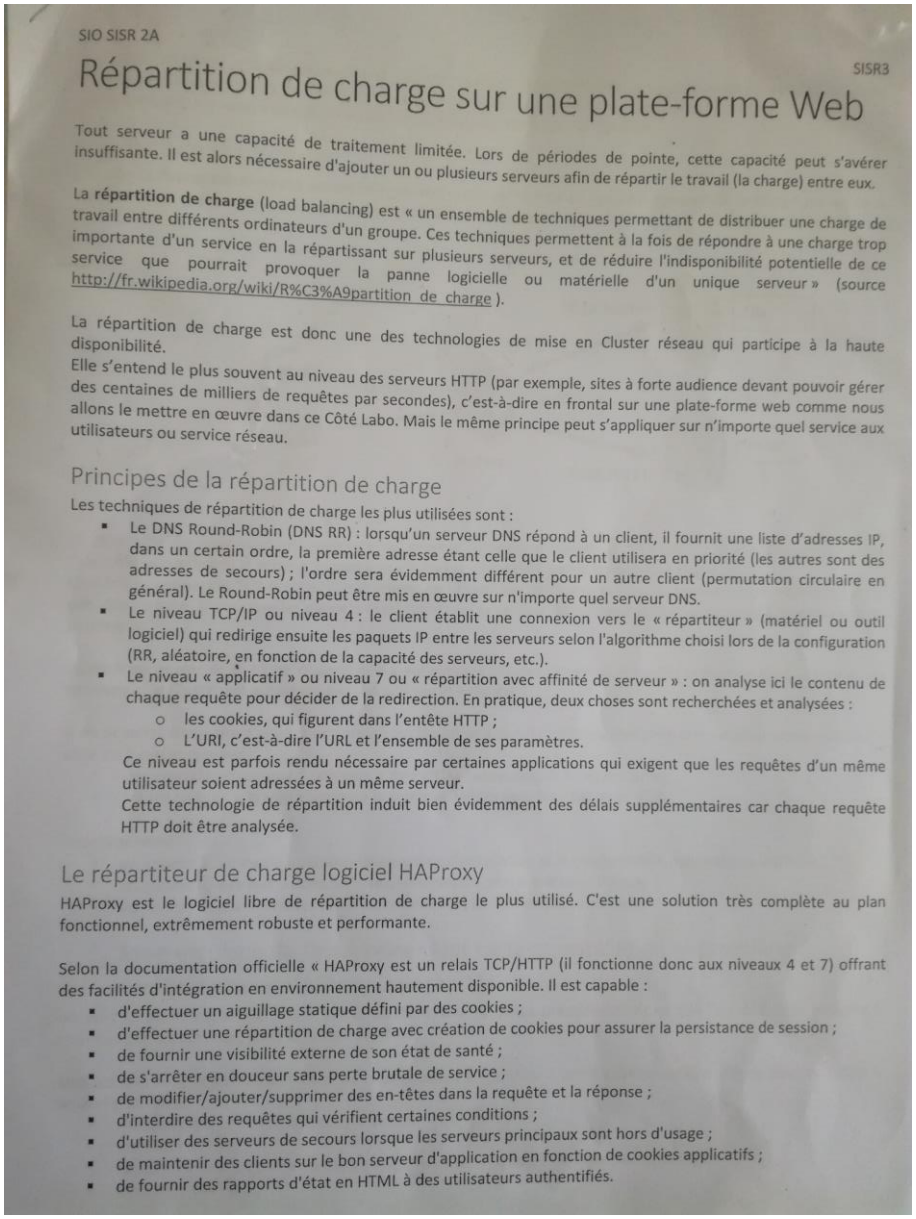
```
root@haproxy:~# haproxy -c -f /etc/haproxy/haproxy.cfg
Configuration file is valid
```

```
root@haproxy:~# service haproxy restart
```

Répartition de charge de niveau 7

On test maintenant de dédier un serveur a la gestion d'une mailing liste/

Annexes



SIO SISR 2A

SISR3

Répartition de charge sur une plate-forme Web

Tout serveur a une capacité de traitement limitée. Lors de périodes de pointe, cette capacité peut s'avérer insuffisante. Il est alors nécessaire d'ajouter un ou plusieurs serveurs afin de répartir le travail (la charge) entre eux.

La **répartition de charge** (load balancing) est « un ensemble de techniques permettant de distribuer une charge de travail entre différents ordinateurs d'un groupe. Ces techniques permettent à la fois de répondre à une charge trop importante d'un service en la répartissant sur plusieurs serveurs, et de réduire l'indisponibilité potentielle de ce service que pourrait provoquer la panne logicielle ou matérielle d'un unique serveur » (source http://fr.wikipedia.org/wiki/R%C3%A9partition_de_charge).

La répartition de charge est donc une des technologies de mise en Cluster réseau qui participe à la haute disponibilité.
Elle s'entend le plus souvent au niveau des serveurs HTTP (par exemple, sites à forte audience devant pouvoir gérer des centaines de milliers de requêtes par secondes), c'est-à-dire en frontal sur une plate-forme web comme nous allons le mettre en œuvre dans ce Côté Labo. Mais le même principe peut s'appliquer sur n'importe quel service aux utilisateurs ou service réseau.

Principes de la répartition de charge

Les techniques de répartition de charge les plus utilisées sont :

- Le DNS Round-Robin (DNS RR) : lorsqu'un serveur DNS répond à un client, il fournit une liste d'adresses IP, dans un certain ordre, la première adresse étant celle que le client utilisera en priorité (les autres sont des adresses de secours) ; l'ordre sera évidemment différent pour un autre client (permutation circulaire en général). Le Round-Robin peut être mis en œuvre sur n'importe quel serveur DNS.
- Le niveau TCP/IP ou niveau 4 : le client établit une connexion vers le « répartiteur » (matériel ou outil logiciel) qui redirige ensuite les paquets IP entre les serveurs selon l'algorithme choisi lors de la configuration (RR, aléatoire, en fonction de la capacité des serveurs, etc.).
- Le niveau « applicatif » ou niveau 7 ou « répartition avec affinité de serveur » : on analyse ici le contenu de chaque requête pour décider de la redirection. En pratique, deux choses sont recherchées et analysées :
  - les cookies, qui figurent dans l'entête HTTP ;
  - L'URI, c'est-à-dire l'URL et l'ensemble de ses paramètres.

  Ce niveau est parfois rendu nécessaire par certaines applications qui exigent que les requêtes d'un même utilisateur soient adressées à un même serveur.
  Cette technologie de répartition induit bien évidemment des délais supplémentaires car chaque requête HTTP doit être analysée.

Le répartiteur de charge logiciel HAProxy

HAProxy est le logiciel libre de répartition de charge le plus utilisé. C'est une solution très complète au plan fonctionnel, extrêmement robuste et performante.

Selon la documentation officielle « HAProxy est un relais TCP/HTTP (il fonctionne donc aux niveaux 4 et 7) offrant des facilités d'intégration en environnement hautement disponible. Il est capable :

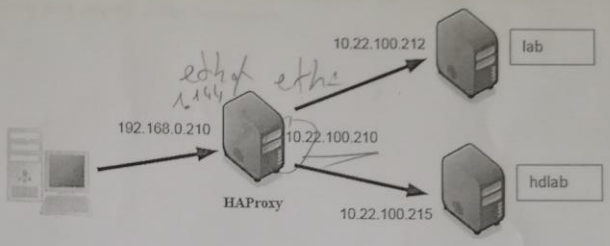
- d'effectuer un aiguillage statique défini par des cookies ;
- d'effectuer une répartition de charge avec création de cookies pour assurer la persistance de session ;
- de fournir une visibilité externe de son état de santé ;
- de s'arrêter en douceur sans perte brutale de service ;
- de modifier/ajouter/supprimer des en-têtes dans la requête et la réponse ;
- d'interdire des requêtes qui vérifient certaines conditions ;
- d'utiliser des serveurs de secours lorsque les serveurs principaux sont hors d'usage ;
- de maintenir des clients sur le bon serveur d'application en fonction de cookies applicatifs ;
- de fournir des rapports d'état en HTML à des utilisateurs authentifiés.

SIO SISR 2A SISR3

En outre, il requiert peu de ressources et son architecture événementielle mono-processus lui permet de gérer facilement plusieurs milliers de connexions simultanées sur plusieurs relais sans effondrer le système.
HAProxy peut aider aussi à se prémunir contre les attaques DOS (Deny Of Service).

Architecture et contexte

Une configuration relativement simple va nous permettre de mettre en œuvre la répartition de charge sur les deux serveurs Web préalablement installés avec réplication multi-maître des données (voit TP « Haute disponibilité du service Web »).



La demande de connexion est adressée au serveur HAProxy d'adresse IP 192.168.0.210 qui détermine, selon l'algorithme configuré, le serveur auquel il va affecter la connexion, parmi les serveurs disponibles.
Une fois la connexion TCP établie, l'équipement de répartition de charge devient pratiquement transparent : dans son rôle de base, il transfère les paquets IP du client vers le serveur sélectionné et vice versa jusqu'à fermeture de la connexion.

La mise en œuvre de la haute disponibilité a été simulée sur des machines virtuelles présentes dans la ferme des serveurs et le Cluster est accessible par son adresse IP virtuelle.

Il est ensuite nécessaire d'ajouter un serveur sur lequel sera installé HAProxy (qui peut être aussi un serveur virtuel) doté de 2 cartes réseaux :

- une considérée comme « publique » permettra l'accès depuis « l'extérieur » ;
- l'autre comme « privée » assurera la communication avec les 2 serveurs Web.

Vous trouverez :

- en annexe 1, un mode opératoire concernant l'installation et la configuration minimale de HAProxy ;
- en annexe 2, des indications sur une configuration plus avancée d'HAProxy ;
- en annexe 3, des exemples (avec commentaires) de statistiques obtenues.

1. Mise en place de l'environnement de test, installation et première configuration d'HAProxy (annexe 1)

On part ici du principe que l'on utilise la solution mise en place précédemment et que l'on doit donc intégrer le nouveau service HAProxy dans le Cluster existant.

Rédigez une note argumentée listant les modifications à apporter à votre environnement afin de pouvoir tester la nouvelle solution.

- Mettre en place cette plate-forme de test.
- Installer HAProxy.