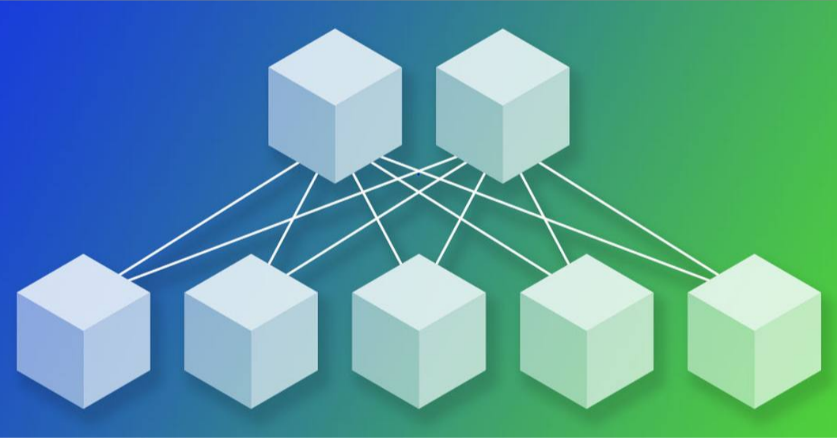
# HAproxy



HAproxy, qui signifie *High Availability Proxy*, est une application de répartition de charge (*Load* *Balancer*) open source TCP/HTTP. 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 (Web, applicatif, base de données, ...).

Il est utilisé dans de nombreux environnements de haut niveau, notamment ... GitHub, Instagram et Twitter.

## Terminologie HAProxy

De nombreux termes et concepts sont importants à connaître pour la mise en place de l'équilibrage de charge et du proxy.

**Liste de contrôle d'accès** (ACL)  
En ce qui concerne l'équilibrage de charge, les ACL sont utilisées pour tester certaines conditions et effectuer une action en fonction du résultat du test (par exemple, sélectionner un serveur ou bloquer une demande).   
L'utilisation d'ACL permet un transfert flexible du trafic réseau en fonction de divers facteurs tels que la correspondance de modèles et le nombre de connexions à un serveur dorsal (*backend*).

Exemple d'ACL ...  
**acl url\_blog path\_beg / blog**Cette ACL est mise en correspondance si le chemin de la demande d'un utilisateur commence par /blog.   
Cela correspondrait à une demande de http://<Nom DNS>/blog/blog01.

**Dorsale** (*backend*)  
Une dorsale est un ensemble de serveurs qui reçoivent des demandes du serveur frontal.   
Les dorsaux sont définis dans la **section backend** de la configuration HAProxy.

Dans sa forme la plus simple, un backend peut être défini par ...

* l’algorithme d'équilibrage de charge utiliser ;
* une liste de serveurs et de ports.

Une dorsale peut contenir un ou plusieurs serveurs - en règle générale - l'ajout de serveurs supplémentaires à la dorsale augmentera la capacité de charge potentielle en répartissant cette charge sur plusieurs serveurs. Une fiabilité accrue est également obtenue de cette manière, au cas où certains serveurs ne seraient plus disponibles.

Voici un exemple de configuration HAproxy à deux serveurs dorsaux identifiés comme web-backend et blog-backend proposant deux services web écoutant sur le port 80 …

**backend web-backend  
 balance roundrobin  
 server web1 web1.exemple.info:80 check  
 server web2 web2.exemple.info:80 check  
   
backend blog-backend  
 balance roundrobin  
 mode http  
 server blog1 blog1.exemple.info:80 check  
 server blog1 blog1.exemple.info:80 check**

Pour cet exemple ...

* la ligne **balance roundrobin** spécifie l'algorithme d'équilibrage de charge ;
* la **ligne mode http** spécifie que le proxy de couche 7 sera utilisé ;
* l'**option check** à la fin des directives serveur spécifie que les vérifications de l'état doivent être effectuées sur ces serveurs .

**Frontale** (*frontend*)

Un frontale définit comment les demandes doivent être transmises aux serveurs dorsaux.   
Les frontales sont définies dans la **section frontend** de la configuration du service HAProxy.

Leurs définitions sont composées des éléments suivants …

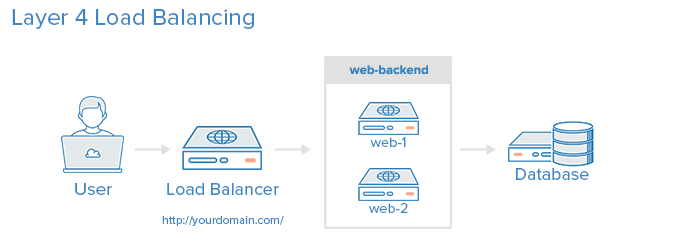
* un ensemble d'adresses IP et un port (par exemple 192.168.1.100:80, \*:443, ...) ;
* d’ACL ;
* des règles use\_backend , qui définissent les serveurs dorsaux à utiliser en fonction des règles définies dans les ACL qui correspondent, et/ou une règle default\_backend qui gère tous les autres cas.

Une interface peut être configurée pour différents types de trafic réseau.

## Types d'équilibrage de charge

Il existe plusieurs manière dont un équilibreur de tâches eut travailler.

**Aucune équilibrage de charge**  
Un environnement d'application Web simple sans équilibrage de charge.

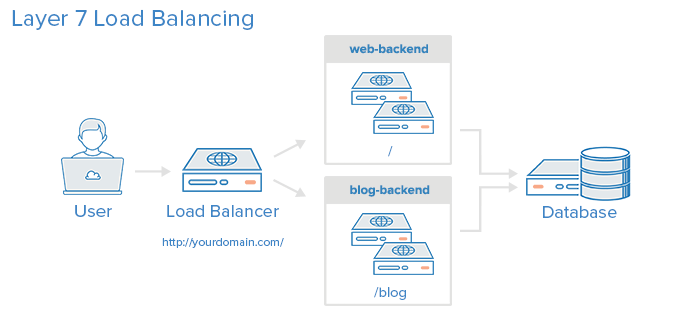


*Image DigitalOcean*

L'utilisateur accède à l'équilibreur de charge, qui transmet la demande de l'utilisateur au groupe de serveurs Web dorsaux. Quel que soit le serveur dorsal sélectionné, il répondra directement à la demande de l'utilisateur.   
En règle générale, tous les serveurs de la dorsale Web doivent diffuser un contenu identique, sinon l'utilisateur risque de recevoir un contenu incohérent.   
Dans l’exemple précédent, les deux serveurs Web se connectent au même serveur de base de données.

**Équilibrage de charge de couche 7**

Un autre moyen plus complexe d'équilibrer la charge du trafic réseau consiste à utiliser l'équilibrage de charge de couche 7 (couche application).   
L'utilisation de la couche 7 permet à l'équilibreur de charge de transférer les demandes vers différents serveurs principaux en fonction du contenu de la demande de l'utilisateur.   
Ce mode d'équilibrage de charge permet d'exécuter plusieurs serveurs d'applications Web sous le même domaine et le même port.



*Image DigitalOcean*

Dans cet exemple, si un utilisateur demande votredomaine.com/blog, il est redirigé vers la dorsale du blog , qui est un ensemble de serveurs qui exécutent une application de blog.   
Les autres demandes sont transmises à la dorsale Web , qui peut exécuter une autre application.   
Les deux dorsales peuvent utilise le même serveur de base de données.

Un extrait de l'exemple de configuration de la dorsale …

**frontend http  
 bind \*:80  
 mode http  
  
 acl url\_blog path\_beg /blog  
 use\_backend blog-backend if url\_blog  
  
 default\_backend web-backend**

Cette configuration propose une interface nommée http , qui gère tout le trafic entrant sur le port 80.  
**acl url\_blog path\_beg /blog** correspond à une requête si le chemin de la requête de l'utilisateur commence par /blog  
**use\_backend blog-backend if url\_blog** utilise l'ACL pour proxy le trafic vers le backend du blog  
**default\_backend web-backend** spécifie que tout le reste du trafic sera transféré vers la dorsaleWeb .

## Algorithmes d'équilibrage de charge

L'algorithme d'équilibrage de charge utilisé détermine quel serveur, pour une dorsale, sera sélectionné lors de l'équilibrage de charge. HAProxy propose plusieurs options pour les algorithmes.

En plus de l'algorithme d'équilibrage de charge, les serveurs peuvent se voir attribuer un paramètre de pondération afin de modifier la fréquence de sélection du serveur par rapport aux autres serveurs.

HAProxy fournit plusieurs algorithmes d'équilibrage de charge.  
Voici quelques-uns des algorithmes couramment utilisés …

* **round robin**  
  L’algorithme Round Robin sélectionne les serveurs à tour de rôle.  
  Il s’agit de l’algorithme le plus fluide et le plus juste car le temps de traitement des serveurs est également réparti.   
  Cet algorithme est dynamique, ce qui permet d'ajuster les pondérations des serveurs à la volée.   
  roundrobin est l'algorithme par défaut ;
* **lessconn**  
  Cet algorithme sélectionne le serveur avec le moins de connexions - il est recommandé pour les sessions plus longues.   
  Tout comme l’algorithme round robin, les serveurs de la même dorsale sont utilisés en alternance.  
  L'utilisation de cet algorithme est recommandée pour de longues sessions, telles que LDAP, SQL, TSE, etc., mais il n'est pas très bien adapté aux sessions courtes telles que HTTP. ;
* **source**  
  Cet algorithme sélectionne le serveur à utiliser en fonction d'un hachage de l'adresse IP source du client.   
  Il s'agit d'une méthode permettant de garantir qu'un utilisateur se connectera toujours au même serveur.  
  Cela s’avère utile si on utilise la persistance des connexions ;
* **first**  
  Cet algorithme envoie la demande au premier serveur disponibles.   
  Les serveurs sont choisis de l'identificateur numérique du plus bas au plus élevé, qui correspond par défaut à la position du serveur de la dorsale.   
  Une fois qu'un serveur atteint sa valeur maxconn définie dans le fichier /etc/haproxy/proxy.cfg, le serveur suivant est utilisé.

## Séances collantes (*sticky sessions*)

Certaines applications exigent qu'un utilisateur continue de se connecter au même serveur principal.   
Cette persistance est obtenue grâce à des sessions persistantes, en utilisant le **paramètre appsession** dans la dorsale qui le nécessite.

## Vérification d’état (*Health Check*)

HAProxy utilise des vérifications d'état afin de déterminer si un serveur principal (un serveur de la dorsale) est disponible pour traiter les demandes.   
Cette vérification évite d'avoir à supprimer manuellement un serveur de la dorsale s'il devient indisponible.

La vérification de l'état par défaut consiste à essayer d'établir une connexion TCP avec le serveur, c'est-à-dire qu'elle vérifie si le serveur principal écoute sur l'adresse IP et le port configurés.

Si un serveur échoue à une vérification de l'état et ne peut donc pas répondre aux demandes, il est automatiquement désactivé au niveau de la dorsale, c'est-à-dire que le trafic ne lui sera pas transmis tant qu'il ne redeviendra actif.   
Si tous les serveurs d’une dorsale échouent, le service deviendra indisponible jusqu'à ce qu'au moins l'un de ces serveurs backend redevienne actif.

Pour certains types de dorsales, comme les serveurs de base de données dans certaines situations, la vérification de l'état par défaut est insuffisante pour déterminer si un serveur est toujours actif.

## Autres solutions

Si HAProxy est peut-être trop complexe pour ses besoins, les solutions suivantes peuvent être mieux adaptées …

**Serveurs virtuels Linux (LVS)**Le LVS est un équilibreur de charge de couche 4 simple et rapide qui est disponible dans de nombreuses distributions Linux.

**Nginx**Nginx est un serveur Web rapide et fiable qui peut également être utilisé à des fins de proxy et d'équilibrage de charge. Nginx est souvent utilisé en conjonction avec HAProxy pour ses capacités de mise en cache et de compression.

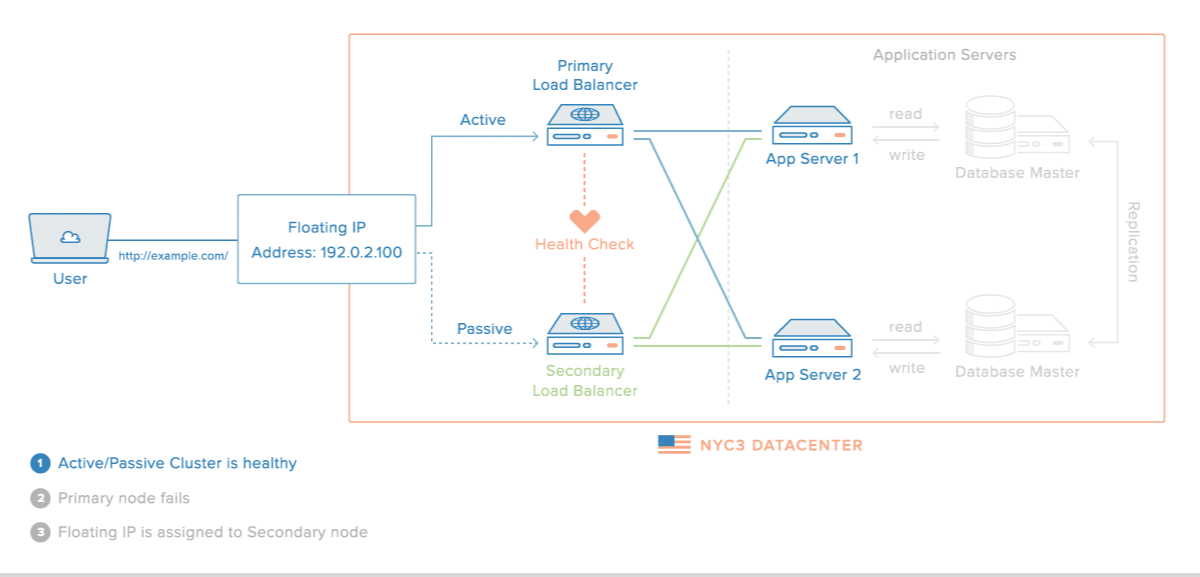
## Haute disponibilité

Les configurations d'équilibrage de charge de couche 4 et 7 utilisent un équilibreur de charge pour diriger le trafic vers l'un des nombreux serveurs principaux. Cependant, un équilibreur de charge est un point de défaillance unique dans ces configurations; s'il tombe en panne ou est submergé de demandes, cela peut entraîner une latence élevée ou des temps d'arrêt pour le service.

Une configuration haute disponibilité (HA) est une infrastructure sans point de défaillance unique.   
Elle évite qu'une panne de serveur unique soit un événement de temps d'arrêt en ajoutant de la redondance à chaque couche de votre architecture.

Un équilibreur de charge facilite la redondance pour la dorsale (serveurs Web/d'applications), mais pour une véritable configuration à haute disponibilité, on doit également disposer d'équilibreurs de charge redondants.

Voici un schéma d'une configuration de base de haute disponibilité …



*Image DigitalOcean*

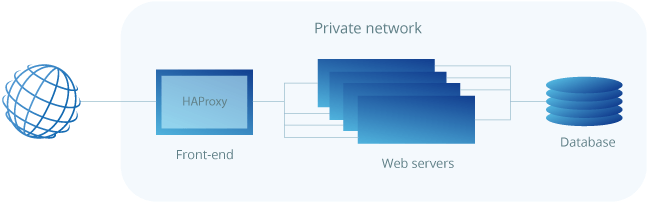
Dans cet exemple, on propose deux équilibreurs de charge (un actif et un ou plusieurs passifs) derrière une adresse IP statique qui peut être liée d'un serveur à un autre.   
Lorsqu'un utilisateur accède au site Web, la demande passe par l'adresse IP externe vers l'équilibreur de charge actif.   
Si cet équilibreur de charge échoue, un mécanisme de basculement le détecte et réaffecte automatiquement l'adresse IP à l'un des serveurs passifs.   
Il existe plusieurs façons d'implémenter une configuration HA active/passive.

# Mise en place du service HAProxy

Pour la mise en place du service, on va travailler avec trois serveurs …

* **Serveur 01** - **HAProxy** -- 192.168.1.12.
* **Serveur 02** - **nginx** -- 192.168.1.10 ;
* **Serveur 03** - **nginx** -- 192.168.1.11 ;

La mise en place devrait ressembler à …



**Installation des serveurs web**

**Installation du serveur HAproxy (frontend)  
>> sudo apt install haproxy**  
**>> sudo /usr/sbin/haproxy -v**

**Serveur 01 (backend)**  
**>> sudo apt install nginx**  
**>> sudo /usr/sbin/nginx -v**

**Serveur 02 (backend)**  
**>> sudo apt install nginx**  
**>> sudo /usr/sbin/nginx -v**

## Configuration du service HAProxy.

La structure du **fichier /etc/haproxy/haproxy.cfg** contient quatre sections …

* **global**  
  Cette section permet de spécifier les **paramètres globaux pour HAProxy**   
  Elle peut inclure le nombre maximum de connexions, le fichier pid, l'emplacement du fichier journal, l'utilisateur/groupe par lequel le démon haproxy sera démarré, … ;
* **defaults**  
  Cette section permet de spécifier **certaines options par défaut** qui peuvent être utilisées pour toutes les **configurations frontends et backends**
* **frontend**  
  Cette section permet de **configurer le serveur frontal**.  
  Elle inclut, entre autres, l’adresse IP et le port sur lequel HAProxy écoutera et d'autres options comme la définition de la frontale.   
  IL est possible de spécifier dans cette section plus d'un frontal au cas où on veut transférer divers trafics comme HTTP / HTTPS / SMTP, … ;
* **backend**   
  Cette section spécifie les **systèmes de la dorsale qui seront le serveur d'application réel vers lequel HAProxy transférera tout le trafic**.  
  Si HAProxy est configuré pour équilibrer la charge du trafic HTTP, les dorsaux seront des serveurs Web ou des serveurs proxy inverses.

Le fichier de configuration du service HAproxy se trouve dans le **répertoire /etc/haproxy** …  
**>> sudo nano /etc/haproxy/haproxy.cfg**

**Remarque** …  
Les **sections global et defaults du fichier /etc/haproxy/haproxy.cfg** peuvent demeurer inchangées.   
Après ces sections, on doit configurer les sections frontend et backend.

On ajoute à la fin du ficher la configuration du proxy …

**# Définition du serveur frontal (frontend)**

**frontend web\_frontal**

**# Port d'écoute frontal - 80**

**bind \*:80**

**# Définition du dorsal (backend) par défaut**

**default\_backend serveurs\_dorsaux**

**# Activation de l'envoi de l'en-tête X-Forwarded-For**

**option forwardfor**

**# Définition des dorsaux (backend)**

**backend serveurs\_dorsaux**

**# Utilisation de Roundrobin pour équilibrer la trafic charge**

**balance roundrobin**

**# Définition des serveurs dorsaux**

**server backend01 192.168.10.11:80 check**

**server backend02 192.168.10.12:80 check**

Avant de redémarrer le service, on **vérifie la cohérence du fichier de configuration** ...  
**> sudo** **haproxy -f /etc/haproxy/haproxy.cfg -c  
Configuration file is valid**

On redémarre le service haproxy …  
**>> sudo systemctl restart haproxy.service**

## Vérification du bon fonctionnement

On saisit l’adresse du serveur HAproxy afin de vérifier que les configurations fonctionnent normalement pour accéder au serveur frontal.  
On recharge la page et on devrait obtenir une réponse de second serveur.

## Aide

On peut facilement obtenir de l’aide sur la commande haproxy …  
**>> sudo haproxy --help**

## Configuration SSL

Dans un premier temps, on combine le certificat et la clé privée …  
**>> cat <Nom du certificat>.pem <Nom de la clé privée>.pem > /etc/haproxy/haproxy.pem**

On configure ensuite HAProxy pour utiliser le certificat SSL sur le serveur frontal ...  
**>> sudo nano /etc/haproxy/haproxy.cfg**

On ajoute à la fin du ficher la configuration du proxy …  
**frontend web\_frontl**  
 **bind \*:80**  
 **...**  
 **bind \*:443 ssl crt /etc/haproxy/haproxy.pem**  
 **...**

## Références

* <https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts>
* <https://computingforgeeks.com/how-to-install-and-configure-haproxy-on-debian/>