



BTS SIO 2 JV SISR

Réalisation de l'infrastructure HAProxy, NGINX sous haute disponibilité

Sur Debian 12



Antoine SPITERI

Sommaire

I)	Infrastructure.....	2
1.	Introduction	2
2.	Topologie	2
2.1	Les interfaces réseaux.....	4
2.1.1	Configuration des interfaces réseaux.....	5
II)	NGINX (Serveur Web)	8
1.	Présentation	8
1.1	Utilisation.....	9
1.2	Topologie	9
2.	Installation	10
3.	Tests de fonctionnement	13
III)	Heartbeat (Surveillance)	14
1.	Introduction	14
2.	Explications.....	14
3.	Installation	14
4.	Vérification et tests	16
IV)	HAProxy	17
1.	Définition	17
2.	Explications.....	17
3.	Topologie	17
4.	Installation	18
5.	Vérification et tests	20
V)	Conclusion	22

I) Infrastructure

1. Introduction

Ce livrable va passer en revue la mise en place de serveurs HAproxy afin de mettre en place l'équilibrage de charges pour des serveurs Web (HTTP) sous NGINX. Afin de garantir la haute disponibilité et l'accès perpétuel au site internet hébergé, nous utiliserons Heartbeat, un service de surveillance et de création de cluster.

Pour utiliser plusieurs machines virtuelles, nous utiliserons le logiciel **VMware Workstation Pro**. Cependant, si vous utilisez un logiciel différent comme VirtualBox ou bien posséder des machines (ordinateurs) physiques, cela n'impacte en aucun cas le suivi de ce livrable.

A la fin de ce livrable, vous serez capable de :

- Comprendre ce qu'est Haproxy
- Installer et configurer Haproxy dans un environnement Linux
- Comprendre ce qu'est NGINX
- Installer et configurer NGINX pour nos serveurs Web
- Comprendre ce qu'est Heartbeat et la haute disponibilité
- Installer et configurer Heartbeat pour garantir l'accessibilité de nos services

2. Topologie

Afin de réaliser notre infrastructure, nous aurons besoin de cinq machines :

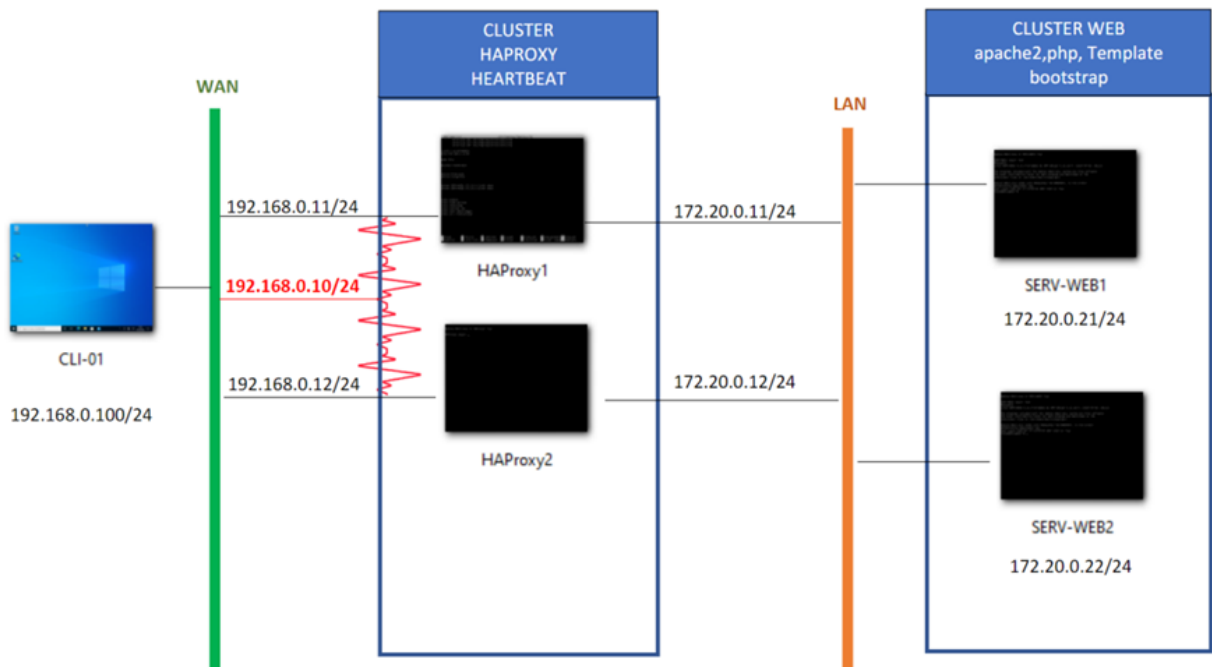
- Deux machines joueront le rôle de serveurs HAProxy (Equilibrage de charges) afin de gérer le trafic sur le réseau.
Elles seront liées au segment LAN « *LAN* » avec les serveurs Web et au segment LAN « *WAN* » avec les machines clientes.
Ces machines s'appelleront : HAProxy1 et HAProxy2
- Par la suite, deux machines joueront le rôle de serveurs Web afin de gérer les requêtes HTTP envoyés par les machines clientes.
Pour communiquer avec les serveurs HAProxy, elles seront également connectées sur le segment LAN « *LAN* »
Ces machines s'appelleront : SRV-WEB1 et SRV-WEB2
- Enfin, une machine jouera le rôle du client pour vérifier le bon fonctionnement de l'ensemble de nos services et l'affichage du site internet local.
Elle sera connecté au segment LAN « *WAN* »

L'intégralité des machines seront sous Debian 12, à l'exception de la machine cliente qui elle sera sous une distribution Windows cliente.

Les caractéristiques des interfaces réseaux de chaque machine seront détaillés dans le prochain chapitre (Chapitre 2.1)

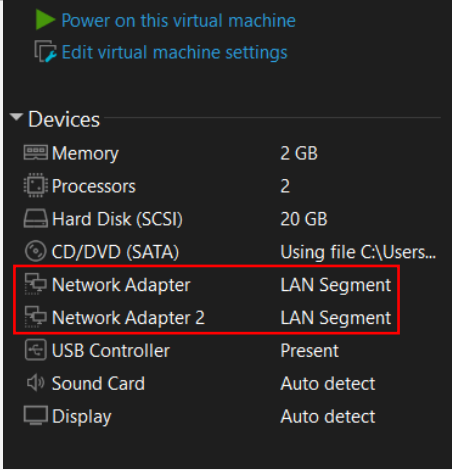
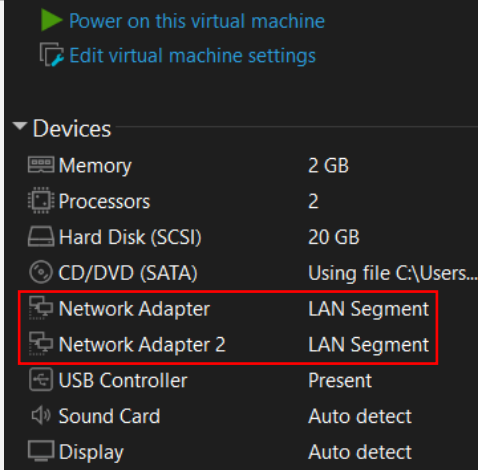
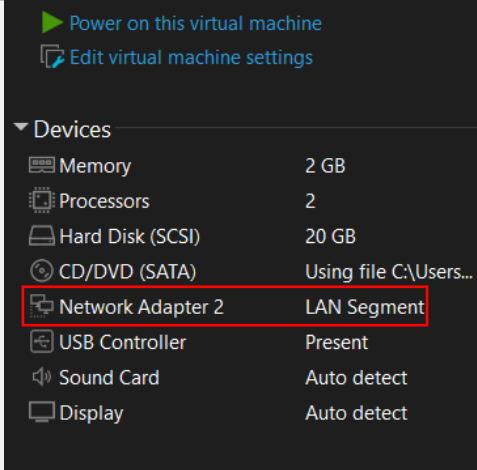
La topologie ci-dessous résume la disposition évoquée :

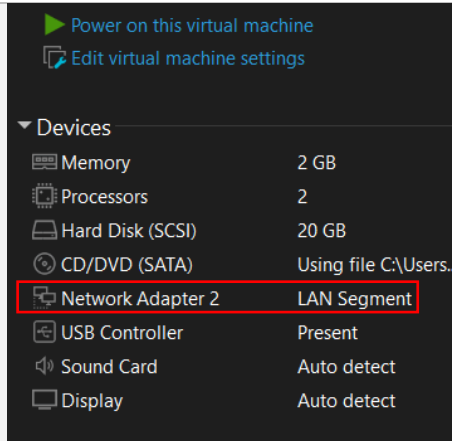
L'adresse IP 192.168.0.10 représente l'adresse IP virtuelle gérée et utilisé par le service Heartbeat, nous y reviendrons plus en détail lors de l'évocation de ce service.



2.1 Les interfaces réseaux

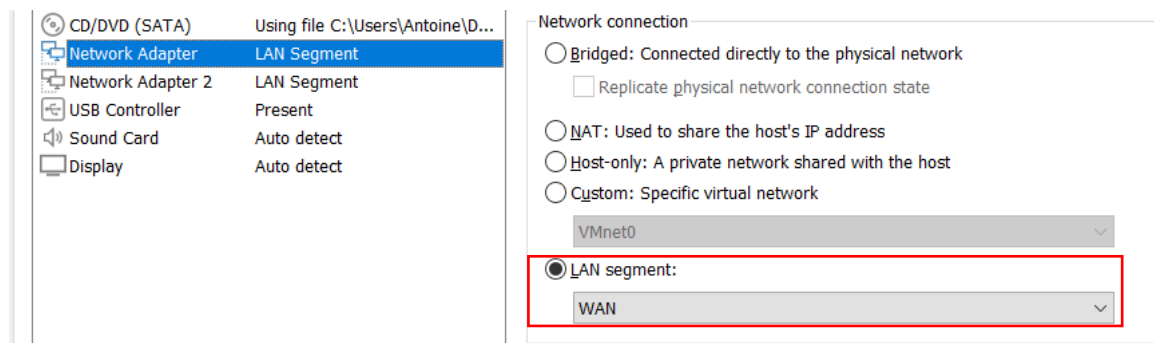
Si vous utilisez un autre logiciel ou bien même, posséder des machines physiques, les pratiques en lien avec le réseau sont les mêmes.

HAProxy1	HAProxy2	SRV-WEB1
		
LAN (NA) : Permet à l'ordinateur de se situer sur le réseau local et Switch virtuel « WAN » LAN (NA 2) : Permet à l'ordinateur de se situer sur le réseau local et Switch virtuel « LAN »	LAN (NA) : Permet à l'ordinateur de se situer sur le réseau local et Switch virtuel « WAN » LAN (NA 2) : Permet à l'ordinateur de se situer sur le réseau local et Switch virtuel « LAN »	LAN : Permet à l'ordinateur de se situer sur le réseau local et Switch virtuel « LAN »

SRV-WEB2

LAN : Permet à l'ordinateur de se situer sur le réseau local et Switch virtuel « LAN »

Pour que toutes nos machines puissent communiquer entre eux et également avec internet, nous allons leur attribuer différentes cartes réseaux (également appelé interfaces réseaux) :

Un LAN pour Local Area Network représente comme son nom l'indique en anglais, un réseau local virtuel (on peut même le décrire comme un switch virtuel). Dans les logiciels de virtualisation, on assigne une machine à ce que l'on appelle un segment. Chaque segment possède un nom. De ce fait, les machines ayant le même nom dans la rubrique segment LAN font parties du même réseau local.



Dans notre cas, nos deux machines clientes font parties de deux réseaux différents. Notre machine Windows fait partie du segment LAN « Windows », et celle avec Debian dans le segment LAN « Linux ».

2.1.1 Configuration des interfaces réseaux

Une fois les interfaces assignées, il est temps d'allumer les machines de notre infrastructure pour les configurer.

Etant donné que nos machines HAProxy et Web jouent le rôle de serveur, il est important de leur donner une configuration IP fixe.

2.1.2 Réseau HAProxy

Une fois sur la machine, assurez-vous d'être connecté en tant que root pour avoir les permissions nécessaires pour écrire dans le fichier interfaces (fichier de configuration des interfaces), sinon, pour passer de votre utilisateur à root, taper la commande :

```
> sudo su -
```

Ouvrir le fichier « interfaces » situé dans /etc/network, pour cela :

```
> nano /etc/network/interfaces
```

Voici la configuration à écrire dans le fichier pour le serveur **HAProxy1** :

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens33 ← # Passerelle statique dirigé vers le lan « WAN »
iface ens33 inet static
address 192.168.0.11/24

# The second network interface
auto ens37 ← # Passerelle statique dirigé vers le lan « LAN »
iface ens37 inet static
address 172.20.0.11/24
```

La configuration à écrire dans le fichier pour le serveur **HAProxy2** (seul l'adresse IP des interfaces changent. La disposition étant la même pour nos serveurs HAProxy) :

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens33
iface ens33 inet static
address 192.168.0.12/24

# The second network interface
auto ens37
iface ens37 inet static
address 172.20.0.12/24
```

Les « # » dans le fichier sont des commentaires, le fichier ne les prend pas en compte lors de son exécution

Enregistrer le fichier grâce à la combinaison CTRL + O puis entrer.

Pour appliquer les modifications et en fonction du nom des interfaces, faites les commandes (ou la commande en fonction des machines) suivantes :

```
> ifdown ens33 && ifup ens33
> ifdown ens37 && ifup ens37
```

ens33, ens37 représente l'identifiant des interfaces, ceci dit attention ! Chaque machine peut se retrouver avec des identifiants différents, prenez donc le temps de réadapter les exemples de ce livrable avec le nom de vos interfaces.

Les « # » dans le fichier sont des commentaires, le fichier ne les prends pas en compte lors de son exécution

Pour les connaître, faite la commande :

```
> ip a
```

Après l'ordre de vos interfaces (2, 3, ...) vous retrouverez juste après leurs identifiants. Dans ce cas précis, les interfaces commencent toutes par **ens** mais cela peut aussi être « eth0 » en fonction du type interface (port Ethernet par exemple) que vous utilisez. **ens** est courant lorsqu'il s'agit de machine virtuelle.

Exemple pour l'interface ens33 :

```
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
    link/ether 00:0c:29:f5:08:fc brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.168.132/24 brd 192.168.168.255 scope global dynamic ens33
        valid_lft 1096sec preferred_lft 1096sec
    inet6 fe80::20c:29ff:fef5:8fc/64 scope link
        valid_lft forever preferred_lft forever
```

2.1.3 Réseau NGINX (Web)

Il n'y a plus qu'à répéter les mêmes étapes mais cette-fois ci, pour nos serveurs Web sous NGINX en respectant l'adressage spécifié dans la topologie au début du chapitre 2 de ce livrable.

Voici la configuration à écrire dans le fichier pour le serveur **SRV-WEB1** :

```
# This file describes the network interfaces available on your system
```



```
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens33
iface ens33 inet static
address 172.20.0.21/24
```

Et enfin pour **SRV-WEB2** :

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens33
iface ens33 inet static
address 172.20.0.22/24
```

N'oubliez pas d'enregistrer le fichier de configuration pour chaque machine et de redémarrer vos interfaces réseaux à l'aide des commandes : « ifdown » et « ifup »

Vos interfaces sont maintenant configurées.

II) NGINX (Serveur Web)

1. Présentation

NGINX est un logiciel (paquet) permettant de la mise en place de serveur HTTP (serveur Web) au sein de notre infrastructure. Connu pour cette fonctionnalité principale, il est également répandu dans la configuration de Proxy inversé et de répartition des charges. Il est surtout apprécié pour ses grandes performances et sa faible consommation sur les ressources matérielles, ce qui fait de lui aujourd'hui, l'un des logiciels de serveur Web le plus utilisé.

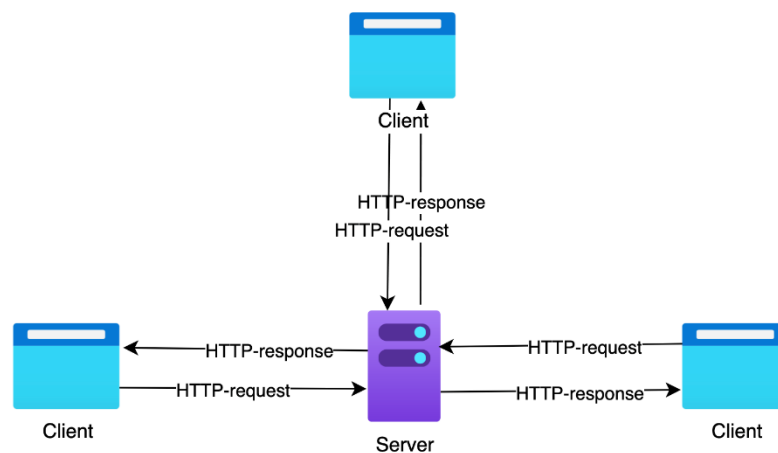
1.1 Utilisation

En configurant un serveur HTTP, que l'on appelle aussi par sa forme moins précise serveur Web, notre server sera capable de répondre à des requêtes et d'envoyer des requêtes en destination ou provenant d'Internet en utilisant (comme son nom l'indique) le protocole de communication HTTP et HTTPS.

Dans notre cas, nous allons installer le paquet NGINX mais également les dépendances nécessaires au fonctionnement de ce dernier, notamment PHP et Wget pour la récupération du code HTML et CSS de notre futur site.

A la fin de la configuration, nous serons capable d'afficher et de visiter un site de restaurant.

1.2 Topologie



2. Installation

Pour l'ensemble du chapitre sur NGINX, chaque étape et chaque commande doit être réalisé sur les deux machines qui serviront de serveurs Web. Les deux doivent être identiques

On va commencer la configuration de nos serveurs NGINX ! Avant toutes choses, on s'assure que les paquets sont bien mis à jour sur toutes nos machines (clientes et serveur), pour cela, on exécute la commande :

```
> apt update && apt upgrade -y
```

Les paquets représente des fichiers d'archives contenant les fichiers informatiques d'un logiciel.

On peut désormais lancer la commande d'installation de NGINX ainsi que PHP et son module FPM sur notre serveur SRV-WEB1 et 2 :

```
> apt install nginx php php-fpm wget unzip -y
```

Une fois le téléchargement fini, rendez-vous dans le répertoire de configuration de site de NGINX. Pour cela :

```
> cd /etc/nginx/sites-availables
```

Effacer le contenu du fichier default, nous allons réécrire la configuration pour qu'elle soit adapté à notre site web.

```
> echo > default
```

Ouvrez le fichier avec votre éditeur sous Debian (nano dans ce cas-ci), et entrez ici la configuration suivante (vous trouverez les explications juste en-dessous) :

```

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/thegrill;

    # Add index.php to the list if you are using PHP
    index index.php index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    # pass PHP scripts to FastCGI server
    #
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        #
        # With php-fpm (or other unix sockets):
        fastcgi_pass unix:/run/php/php8.2-fpm.sock;
        # With php-cgi (or other tcp sockets):
        fastcgi_pass 127.0.0.1:9000;
    }
}

```

Une fois rédigé, enregistrer ce fichier

Les deux premières lignes (listen 80) spécifie sur quel port doit écouter le serveur, étant donné qu'il devra traiter les requêtes menant vers le web (http), nous spécifions le protocole de ce dernier.

```
root /var/www/thegrill;
```

Nous spécifions ensuite à notre serveur où trouver le contenu de notre site internet pour qu'il puisse aller chercher et charger les fichiers quand un client de notre réseau local en fera la demande.

```
index index.php index.html index.htm index.nginx-debian.html;
```

On indique à notre serveur le nom du fichier index (le fichier qu'il charge quand on tape l'adresse IP dans notre navigateur) par ordre de priorité. En l'occurrence, il commencera par chercher le fichier index.php avant de chercher index.html

⚠ Si vous comptez ajouter du PHP au sein de votre site (récupérer le nom de la machine par exemple), n'oubliez pas de renommer votre fichier *index.html* en *index.php*.

Le paragraphe *location* / ne possède pas beaucoup d'options dans notre configuration. Il sert simplement à rediriger vers une page erreur 404 si il ne trouve pas la page demandée.

Vous pouvez ignorer le dernier paragraphe si vous ne comptez pas utiliser et inclure PHP dans votre site internet.

Enfin le dernier paragraphe *location \.php\$* nous sert à spécifier à notre site que l'on utilise le module FPM de PHP et d'inclure le snippets. En quelque sorte, ce paragraphe nous permet d'activer et d'attribuer PHP à notre site. Sans ce dernier renseigné, une erreur sera générée quand nous souhaiterons y accéder.

Il est maintenant temps d'ajouter le contenu de notre site au répertoire indiqué dans la configuration précédemment enregistrer.

```
> cd /var/www/
```

Il ne nous reste plus qu'à importer le fichier zip comprenant l'ensemble des ressources constituant notre beau site internet. Nous pouvons importer un repertoire ou des fichiers présents sur internet grâce à la commande et l'outil wget.

```
> wget https://github.com/technext/thegrill/archive/master.zip
```

Il faut maintenant décompresser le fichier zip, pour cela, nous devons faire la commande suivante :

```
> unzip thegrill-master
```

Nous avons maintenant un répertoire sous le nom de « thegrill-master » nous allons le renommer en « thegrill » pour qu'il corresponde avec notre configuration NGINX

```
> mv thegrill-master thegrill
```

Désormais, lorsque vous vous rendrez dans le répertoire « thegrill » vous devriez apercevoir le fameux fichier index.

Si vous avez écrit du code PHP à l'intérieur de ce dernier, n'oubliez pas de le renommer en *index.php*

3. Tests de fonctionnement

Notre configuration et l'importation de notre site terminé, il est maintenant temps de redémarrer NGINX :

```
> systemctl restart nginx
```

Puis on vérifie le status avec :

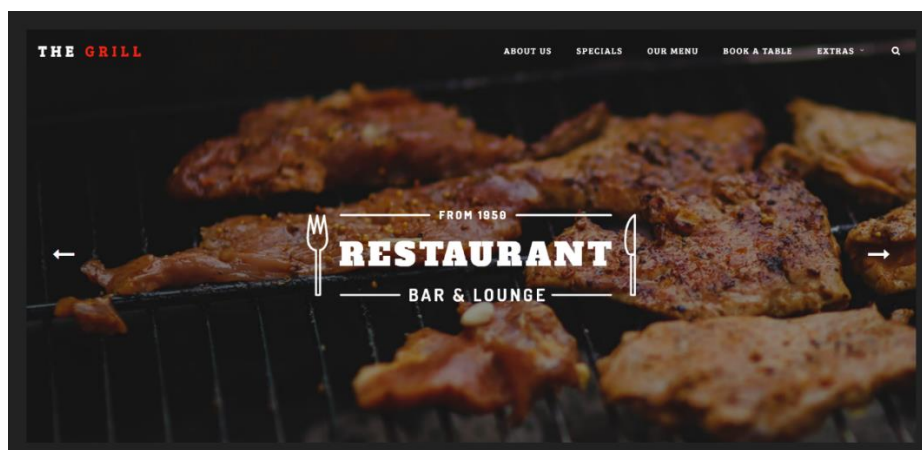
```
> systemctl status nginx
```

```
• nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sat 2025-01-25 16:55:59 CET; 1h 49min ago
     Docs: man:nginx(8)
   Process: 945 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 984 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 985 (nginx)
    Tasks: 3 (limit: 2264)
   Memory: 5.0M
      CPU: 26ms
   CGroup: /system.slice/nginx.service
           └─985 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─986 "nginx: worker process"
               └─987 "nginx: worker process"

janv. 25 16:55:59 SRV-WEB2 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
janv. 25 16:55:59 SRV-WEB2 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
root@SRV-WEB2:~#
```

Le **active (running)** nous indique que le service est opérationnel. Il est maintenant temps de faire un test sur la machine cliente Windows (vous pouvez temporairement mettre la carte réseau de votre machine cliente dans le segment lan « LAN » pour qu'il puisse communiquer avec nos deux serveurs Web)

Si tout va bien, une fois connecté sur votre machine cliente windows (graphique), en tapant l'adresse IP de votre serveur web, vous devriez tomber sur le site du restaurant « thegrill » qui mets même l'eau à la bouche !



Nos serveurs Web sont désormais opérationnels et sont prêts à être intégré à nos prochains serveurs HAProxy pour la configuration de la répartition des charges et garantir la haute disponibilité de notre infrastructure.

III) Heartbeat (Surveillance)

1. Introduction

Le chapitre IV expliquera et mettra en pratique nos serveurs HAProxy. Ce dernier nous permet de répartir les requêtes vers un serveur Web d'un même groupement en fonction du trafic. Cependant, il faut s'assurer que ces serveurs tournent continuellement en s'assurant qu'au moins un serveur soit capable de traiter les requêtes pour le balancement. Une panne de serveur, sans système de relais ou de haute disponibilité, empêchera les machines clientes toute communication et ne pourra plus recevoir et envoyer de paquet. Nous allons donc utiliser le service Heartbeat

2. Explications

Heartbeat est un service de surveillance des serveurs. Il permet de mettre en clusters plusieurs serveurs afin de garantir la haute disponibilité. En d'autres termes, un serveur faisant parti du même cluster qu'un autre peut prendre le relais s'il vient à tomber en panne ou présenter un problème avec un autre service (Apache par exemple).

Ce service représente les clusters par une adresse IP virtuelle. Lorsqu'un client souhaitera solliciter les serveurs d'un même cluster, ce dernier devra taper cette fameuse adresse IP pour que Heartbeat, en fonction de la disponibilité des serveurs, nous redirige vers l'un d'entre eux.

3. Installation

Comme pour NGINX, chaque étape et chaque commande doit être réalisé sur les deux machines qui serviront de serveurs Load-Balancing (HAProxy). Les deux doivent être identiques.

On commence tranquillement par installer le paquet Heartbeat :

```
> apt install heartbeat -y
```

Une fois l'installation terminée, on va créer le fichier de configuration, le répertoire `/etc/ha.d/`

```
> nano /etc/ha.d/ha.cf
```

Entrer-y le contenu suivant (*Les commentaires vous expliquent le principe de chaque instruction*) :

```
# Les fichiers logs de heartbeat (événements relatifs à heartbeat)
logfile    /var/log/heartbeat
logfacility local0

# Intervalle entre deux battements de cœur en seconde
keepalive  5

# Temps nécessaire avant de considérer qu'un serveur (nœud) est mort (en seconde)
deadtime   30

# Interface d'écoute
bcast      ens33

# liste des nœuds utilisées pour la HD
node       HAProxy1 HAProxy2

# comportement si le nœud revient dans le réseau
auto_failback on
```

Enregistrer ce fichier (CTRL + O)

Nous allons maintenant créer un second fichier dans ce même répertoire pour y renseigner et activer l'adresse IP virtuelle qui servira à nous rediriger vers l'un des deux serveurs HAProxy en fonction de leurs états.

```
> nano /etc/ha.d/haresources
```

Entrez-y, cette configuration :

```
# Active l'interface IP Virtuelle avec comme noeud principal srvWeb1
# syntaxe :hostname IPaddr::IPvirtuelle/CIDR/interface service

HAProxy1 IPaddr::192.168.0.10/24/ens33 haproxy
```

Nous spécifions le « nœud maitre », c'est-à-dire le serveur principal que heartbeat priorise, suivi de l'adresse IP virtuelle avec son CIDR (255.255.255.0 pour /24) et l'interface d'écoute. Enfin, on spécifie sur quel service il doit se baser.

Enregistrer ce fichier (CTRL + O)

Il nous reste plus qu'à créer un dernier fichier, toujours dans le même répertoire :

```
> nano /etc/ha.d/authkeys
```

```
auth 1
1 md5 greta
```

Ce fichier de configuration est nécessaire à l'authentification.

Enregistrer ce fichier (CTRL + O)

4. Vérification et tests

Notre configuration Heartbeat est maintenant terminée. Nous devons redémarrer le service pour appliquer le changement

```
> systemctl restart heartbeat
```

On vérifie qu'aucune erreur n'a été générée

```
> systemctl status heartbeat
```

À la suite de cela, l'adresse IP virtuelle spécifiée dans le fichier *haresources* devrait figurer en dessous du nom de notre interface et de son adresse IP principale. On peut vérifier cela en faisant la commande :

```
> ip a
```

```
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:fc:9c:84 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.0.11/24 brd 192.168.0.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet 192.168.0.10/24 brd 192.168.0.255 scope global secondary ens33:0
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe9c:84/64 scope link
        valid_lft forever preferred_lft forever
```

Cette dernière doit être présente sur nos deux serveurs HAProxy.

Elle s'est mise correctement ! Il est donc temps de configurer le service HAProxy pour que l'on puisse réaliser le test d'accès à notre site web et notre page des statistiques de notre infrastructure via cette même adresse.

IV) HAProxy

1. Définition

HAProxy est un logiciel (paquet) Open-source, permettant l'équilibrage de charges pour maintenir la haute disponibilité sur son infrastructure réseau. Utilisé pour les applications TCP et HTTP, son but est d'intercepter les requêtes clientes afin de les rediriger vers l'un des serveurs d'un cluster en fonction de la fréquentation et de l'utilisation de ces derniers. Les performances, l'optimisation et la rapidité sont ainsi garantie.

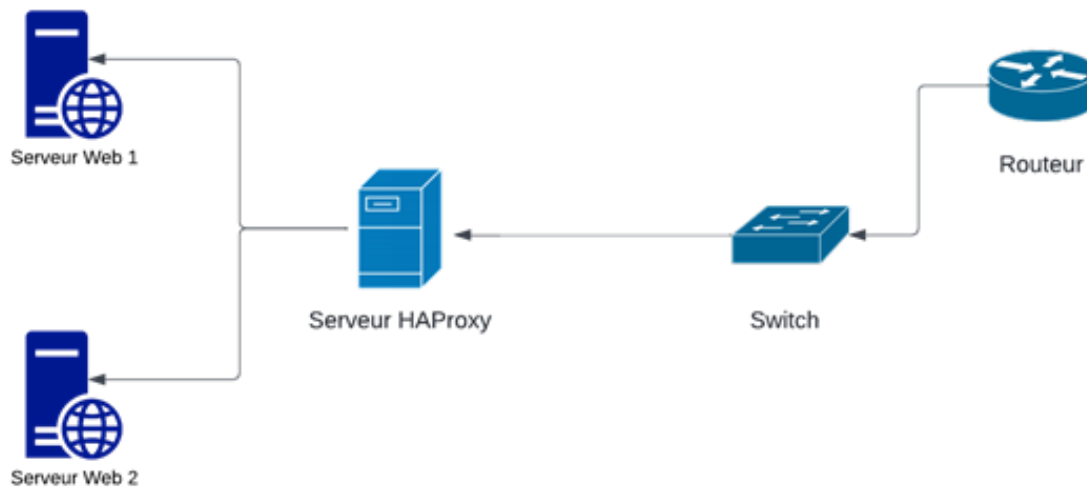
2. Explications

Par exemple, lorsqu'une application ou un site internet commence à connaître une popularité remarquable, augmentant fortement son taux de fréquentation, les ressources d'un seul serveur Web pourra difficilement tenir la cadence (même avec du matériel puissant). Dans ce type de situations, plusieurs serveurs Web sont mis en place pour répondre aux requêtes HTTP et envoyer le site. Cependant, avant que ces derniers ne puissent traiter les requêtes, un intermédiaire intervient pour qu'un serveur Web ne soit pas surcharger de demandes et ralentisse. HAProxy entre en jeu. Lorsqu'un client souhaite accéder à une page Web, sa requête sera envoyée vers un serveur HAProxy qui, en fonction du trafic, l'enverra vers un des multiples serveurs Web (qui ont par ailleurs, le même contenu) présent pour que leurs charges soit tous équitables entre eux.

Dans le cas de ce livrable, nous allons mettre en place deux serveurs HAProxy identiques qui pointeront vers le LAN où se situe nos serveurs Web Topologie et fonctionnement

3. Topologie

En résumé de l'explication et de l'introduction, voici une topologie démontrant ses principes (cette dernière ne reflète pas le cas pratique de ce livrable mais le principe reste le même) :



4. Installation

Même chose, chaque étape et chaque commande doit être réalisé sur les deux machines qui serviront de serveurs Load-Balancing (HAProxy). Les deux doivent être identiques.

Pour installer le service HAProxy sur notre serveur, il faut installer le paquet du même nom

```
> apt install haproxy -y
```

La première chose consiste à renseigner les informations de notre répartition des charges sur le fichier de configuration d'HAProxy. Ce dernier se trouve dans le répertoire `/etc/haproxy/haproxy.cfg`

```
> cd /etc/haproxy
> nano haproxy.cfg
```

Une fois à l'intérieur, on y entre cette configuration :

Le tableau ci-dessous donne les informations relatives aux termes employés dans le fichier de configuration d'HAProxy (`haproxy.cfg`) :

```
# Configuration du balancement
listen  HAProxy1
bind    192.168.0.11:80

# mode d'écoute
mode    http

# mode du balancement (roundrobin (50%-50%))
balance roundrobin

# Option
option  httpclose
option  forwardfor

# Liste des serveurs impliqués pas le balancement
server  SRV-WEB1  172.20.0.21:80  check
server  SRV-WEB2  172.20.0.22:80  check

# Pour les statistiques
stats   enable
stats   hide-version
stats   refresh 30s
stats   show-node
stats   auth    admin:password
stats   uri     /stats
```

Listen HAProxy1 bind 192.168.0.10 :80	Ce premier paragraphe indique à HAProxy la source IP et le nom du serveur par lequel il va fonctionner (en l'occurrence cette même machine) ainsi que son port (80 pour HTTP)
mode http	Quel protocole doit-il écouter pour le balancement
Balance roundrobin	Plusieurs types d'options algorithmiques existent pour HAProxy. Ici, « roundrobin » signifie que le trafic sera distribué aux serveurs à tour de rôle. Vu que nous n'avons que deux serveurs Web, chaque serveur récupérera une moitié du trafic
option	Comprend les options d'interprétation des requêtes
server SRV-WEB1 172.20.0.21 :80 check server SRV-WEB2 172.20.0.22 :80 check	Nous indiquons quels serveurs seront impliqués par le balancement avec leur adresse IP et le type de requête qui seront pris en charge (80 -> HTTP) <i>Check</i> signifie qu'un contre-rendu sera fait sur l'état des serveurs spécifiés.
stats enable stats hide-version stats refresh 30s stats show-mode stats auth admin :password stats uri /stats	<ul style="list-style-type: none"> - On active la page statistique - On cache la version - Rafraichissement automatique toutes les 30s de la page des statistiques - Affichage d'informations supplémentaires sur divers modules - Identifiants de connexion au panel admin - Alias personnalisé de l'URL pour la page des statistiques

Enregistrer votre fichier (CTRL + O).

Toutefois, pour que nos serveurs puissent traduire les noms d'hôtes en adresses IP, nous allons éditer le fichier `hosts`. Ce dernier se trouve dans le répertoire `/etc`

La configuration d'HAProxy est d'or est déjà terminée, c'était plutôt rapide ! 😊

Il est donc temps de redémarrer le service sur nos deux serveurs pour que les changements soient pris en compte.

```
> systemctl restart nginx
```

On vérifie que tout est correct en vérifiant son status (Active Running doit s'afficher) :

```
> systemctl status nginx
```

```

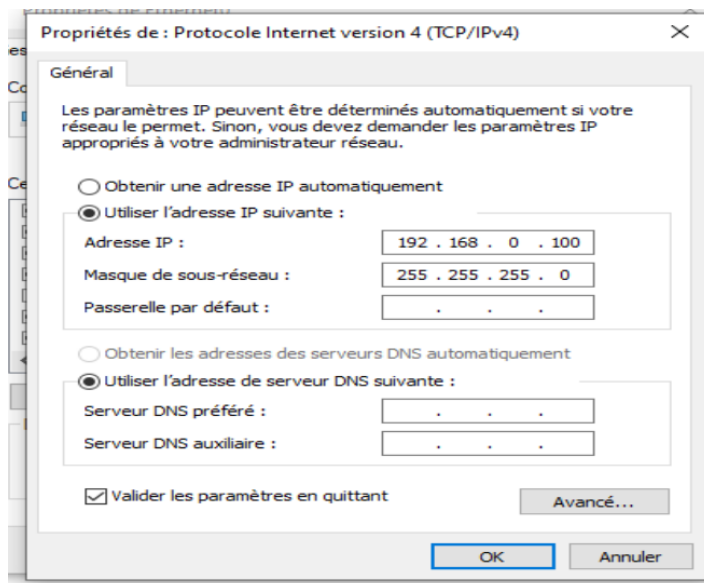
• haproxy.service - HAProxy Load Balancer
  Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; preset: enabled)
  Active: active (running) since Mon 2025-01-27 12:20:59 CET; 2h 47min ago
    Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
  Main PID: 878 (haproxy)
    Tasks: 3 (limit: 2264)
  Memory: 49.1M
    CPU: 2.174s
  CGroup: /system.slice/haproxy.service
          └─878 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock
            └─939 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock

```

5. Vérification et tests

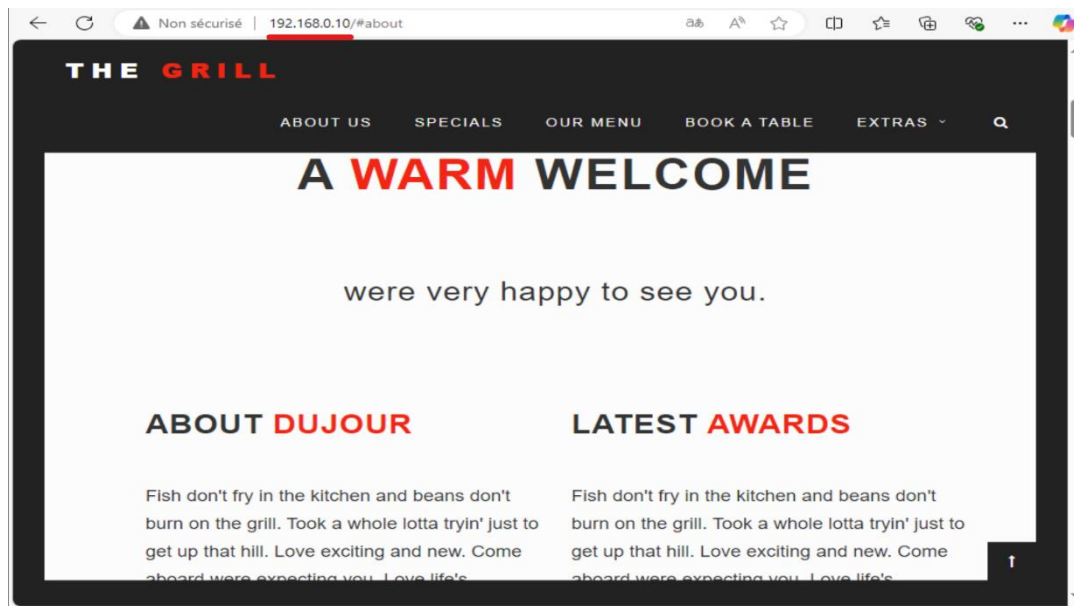
Parfait ! Il est temps de réaliser des tests avec notre machine cliente pour s'assurer que le balancement fonctionne bien. Allumez votre machine cliente graphique (Windows, Debian, peu importe) et assurez-vous (dans les paramètres de vos interfaces réseaux de votre logiciel de virtualisation) que sa carte réseau est bien dans le LAN « WAN ».

Rendez-vous dans le centre réseau de Windows et sélectionner-y votre interface puis cliquer sur l'option *TCP/IPv4*, entrer ces informations IP :



Cocher « Valider les paramètres en quittant » puis OK.

Ouvrez un navigateur Internet et taper dans la barre d'URL, l'adresse IP virtuelle Heartbeat (soit 192.168.0.10), si vous n'avez pas fait d'erreur dans votre configuration, surprise ! Votre site apparaît et l'un de vos deux serveurs HAProxy vous a bien redirigé vers l'un des deux serveurs Web.



Pour rappel, les deux serveurs HAProxy doivent avoir la même configuration. En d'autres termes, être identiques.

Cette fois-ci, en ajoutant /stats après l'adresse IP, vous devriez tomber sur la page des statistiques, avec l'état des différentes machines, des informations sur leurs performances, etc... Un vrai tableau de bord !

HAProxy1		Queue		Session rate		Sessions		Bytes		Denied		Errors		Warnings		Status		Server															
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Reb	Redis	LastChk	Wght	Act	Bck	Chk	Down	Downtime	Thrtle				
Frontend	0	0	0	24	-	1	6	262	123	30			16 077	1 972 418	0	0	1	0	0	0	0	0	37m26s	UP	L4CHK	in 0ms	1/1	Y	-	1	1	1m48s	-
SRV-WEB1	0	0	0	13	0	3	-	15		18	2m54s	6 567	1 065 645	0	0	0	0	0	0	0	0	37m25s	UP	L4CHK	in 0ms	1/1	Y	-	1	1	1m48s	-	
SRV-WEB2	0	0	0	12	0	3	-	15	15	31m20s	0	209	819 459	0	0	0	0	0	0	0	0	37m25s	UP	L4CHK	in 0ms	1/1	Y	-	1	1	1m48s	-	
Backend	0	0	0	25	0	6	26	213	31	31		0s	16 077	1 972 418	0	0	0	0	0	0	0	37m26s	UP			2/2	2	0		1	1m47s		

Comme l'indique la flèche rouge, Heartbeat nous a bien redirigé vers notre serveur prioritaire (actif) HAProxy1 comme spécifié dans notre configuration.

Maintenant, éteignez un de vos serveurs Web. Dans cet exemple, nous allons éteindre SRV-WEB1. Une fois éteint, rafraichissez la page des statistiques et observez les changements :

HAProxy1		Queue		Session rate		Sessions				Bytes		Denied		Errors		Warnings		Server										
		Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bok	Chk	Dwn	Dwntme	Thrtle
Frontend		0	0	25	-	1	6	262 123	138		67 466	3 682 321	0	0	5					OPEN								
SRV-WEB1	0	0	-	0	13	0	3	-	15	15	1h31m	6 417	1 033 759	0	0	0	0	0	0	3m36s DOWN	* L4TOUT in 2002ms	1/1	Y	-	4	2	8m53s	-
SRV-WEB2	0	0	-	0	12	0	3	-	14	14	1h31m	5 743	806 071	0	0	0	0	0	0	3h56m UP	L4OK in 0ms	1/1	Y	-	1	1	5m22s	-
Backend	0	0	0	25	0	6	26	213	29	29	0s	67 466	3 682 321	0	0	0	0	0	0	3h56m UP		1/1	1	0	1	5m16s		

HAProxy a bien pris en compte le changement d'état de la machine (vous pouvez aussi attendre 30 secondes que l'auto-refresh fasse effet) et a passé SRV-WEB1 en rouge. Désormais, lorsque vous taperez l'adresse IP d'HAProxy 1 ou 2 dans la barre d'URL d'un navigateur, ce dernier vous redigera forcément SRV-WEB2 étant donné que SRV-WEB1 n'a plus la capacité de servir les requêtes HTTP.

Nos serveurs de Load-Balancing sont désormais opérationnels et prêt à être utiliser pour traiter de futures requêtes.

V) Conclusion

Ce livrable vous a permis de mettre en place une infrastructure système qui assure la haute disponibilité avec Heartbeat de vos serveurs Web tout en anticipant les charges de trafic élevés grâce à HAProxy.

Vous l'aurez compris, le but de cette configuration est de maximiser la fluidité de navigation et accès aux différents sites présents dans un cluster Web. Cela permet d'augmenter les performances de ces derniers et de maintenir une sérénité importante au sein du service informatique.

Vous pouvez donc être sûr, que vos différents services pourront tourner sans pépin au sein d'un réseau local, métropolitain ou bien même, tout autour du globe terrestre avec une installation de ce type.