LÉONARD DE VINCI LYCÉE POLYVALENT MELUN

BTS SIO option SISR

AP10 REPLICATION HAPROXY HEARTBEAT

Société menuimetal

Présenté par :

BAYERE Abdoul Fatahou
GUIHARD Mathieu

Sommaire

Liens vers les ressources partagées :

Gestion VMs et VLANS :

(GUIHARD_BAYERE) Gestion VLAN et VMs du contexte "Menui…

Gantt :

(GUIHARD_BAYERE) Diagramme de Gantt AP10

I. Introduction

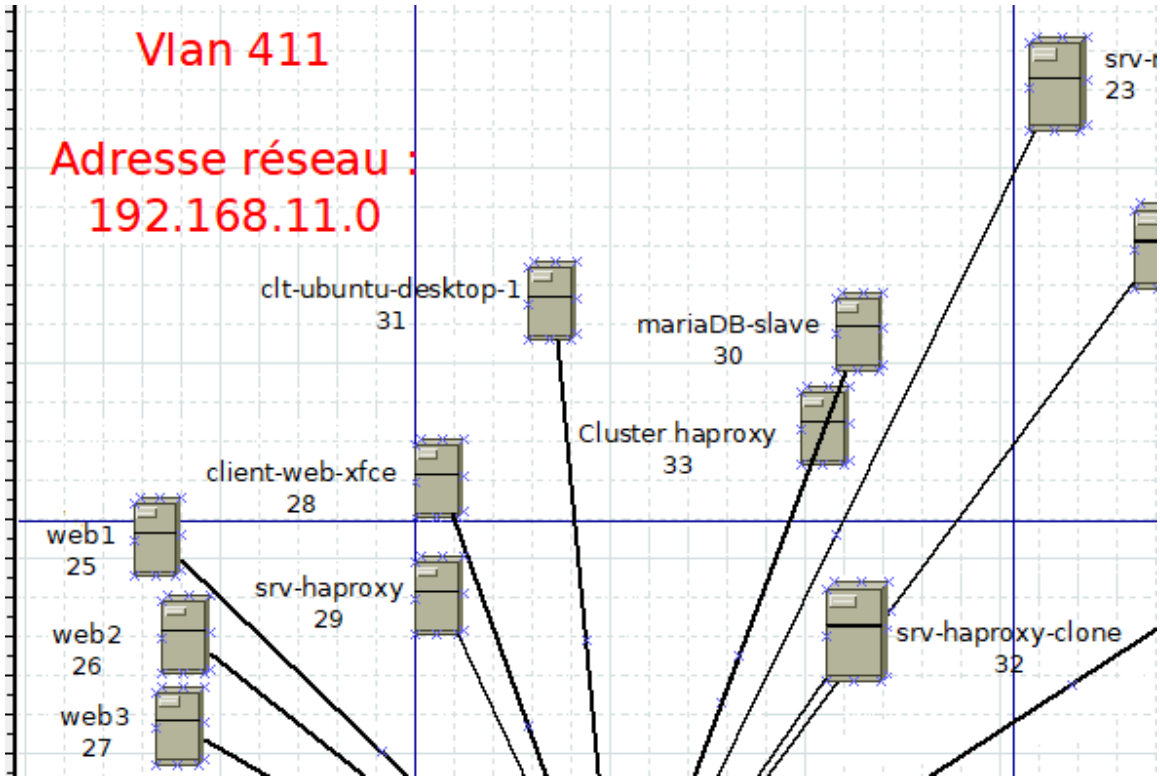
En 1980, Jean Morin crée Menuimetal.SA à Lens, une entreprise spécialisée dans la conception et la fabrication de structures en métal et en verre. Tout en se concentrant sur la production de huisseries et d'éléments de façade, Menuimetal délègue la pose à des partenaires externes. Avec un bureau d'étude capable de répondre aux besoins spécifiques de ses clients, l'entreprise propose des solutions sur mesure et poursuit sa croissance en cherchant à optimiser ses services informatiques.

II. Mise en place du Gantt

Tâches ou WBS					
Lettre	Titre	Jour et heure de début	Antécédent(s)	Durée en heure	Affectée à
A	Clonage et configuration de 3 serveurs de web Linux	19/11/2024 14:00:00		1	
B	Clonage et configuration de 1 cliente pour les tests	19/11/2024 14:15:00	A	1	
C	Vérification de l'accès SSH pour les VMs clonées.	19/11/2024 14:30:00	B	0,5	
D	Référencement des VMs clonées (serveurs et cliente) sur le DNS	19/11/2024 15:00:00	C	0,5	
E	Remonter les VMs sur GLPI et Nagios	19/11/2024 15:30:00	D	1	
F	Configuration de la VM Haproxy	21/11/2024 08:30:00	E	1,5	
G	Sauvegarde et Configuration de la VM MariaDB (maitre)	21/11/2024 10:00:00	F	2	
H	Sauvegarde et Configuration de la VM MariaDB (slave)	21/11/2024 12:00:00	G	2	
I	Tester la réplication et intégrer les logs dans rsyslog.	21/11/2024 13:30:00	H	0,5	
J	Configurer les sections frontend/backend dans haproxy.cfg.	21/11/2024 14:00:00	I	1	
K	Tester la continuité en arrêtant un serveur web.	21/11/2024 14:30:00	J	0,5	
L	Activer les statistiques et commenter les résultats.	21/11/2024 15:00:00	K	0,5	
M	Cloner et configurer le second serveur HaProxy	22/11/2024 13:30:00	L	1	
N	Installer et configurer Heartbeat	22/11/2024 14:30:00	M	1	
O	Tester la tolérance de panne en arrêtant un serveur HaProxy.	22/11/2024 15:30:00	N	1	
P	Documenter les configurations effectuées et les erreurs rencontrées	22/11/2024 16:00:00	O	0,5	
Q	Valider tous les tests	22/11/2024 16:30:00	P	0,5	

Visualisation du Gantt

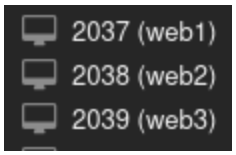
III. Mise à jour du schéma réseau



Visualisation du schéma réseau mis à jour

IV. Mission 0 : Prérequis et contraintes

Clonage de 3 VMs Serveurs Web Linux sur le VLAN LAN et Modification des Hostnames :



Création des VM web

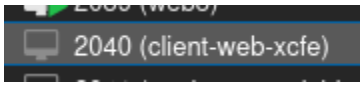
Vérification de l'Attribution de l'Adresse IP sur le Serveur :

```
sio21@web1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKN
OWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo
_fast state UP group default qlen 1000
    link/ether bc:24:11:05:2e:e3 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.11.25/24 brd 192.168.11.255 scope global ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:fe05:2ee3/64 scope link
       valid_lft forever preferred_lft forever
```

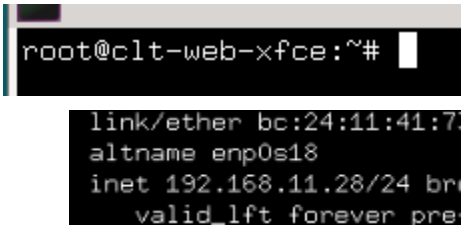
```
sio21@web2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKN
OWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo
_fast state UP group default qlen 1000
    link/ether bc:24:11:59:5c:04 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.11.26/24 brd 192.168.11.255 scope global ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:fe59:5c04/64 scope link
       valid_lft forever preferred_lft forever
```

```
sio21@web3:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKN
OWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo
_fast state UP group default qlen 1000
    link/ether bc:24:11:b0:fb:ce brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.11.27/24 brd 192.168.11.255 scope global ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:feb0:fbce/64 scope link
       valid_lft forever preferred_lft forever
```

Création d'une VM Cliente Clonée pour Tester les Serveurs Web :

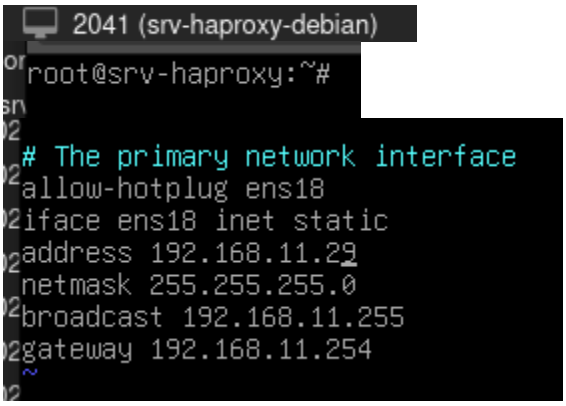


Création d'une VM cliente



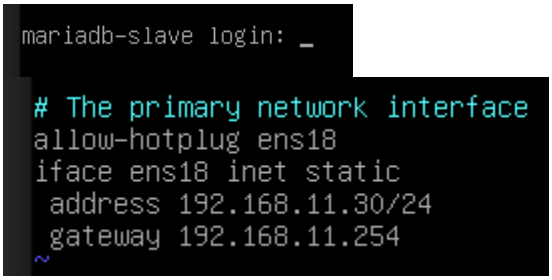
Attribution de l'Adresse IP sur le client

Clonage d'une VM Serveur pour Haproxy et Configuration du VLAN LAN (Hostname et IP) :



Attribution de l'Adresse IP sur le client

Clonage de la VM MariaDB avec les Bases de Données GLPI, Nagios et Autres :



Attribution de l'Adresse IP sur le client

Configuration et Attribution d'un DNS pour les Serveur :

```
search menuimetal.fr
nameserver 192.168.12.1
```

Référencement des VMs Serveurs et de la VM Cliente sur le DNS :

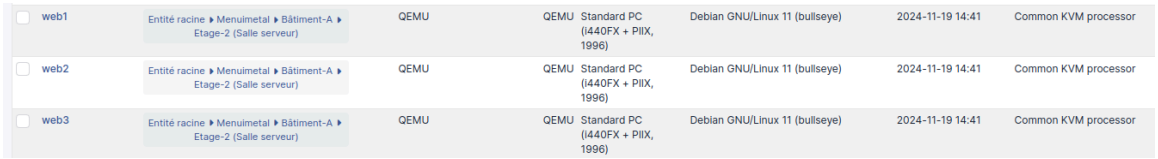
```
web1 IN A 192.168.11.25
web2 IN A 192.168.11.26
web3 IN A 192.168.11.27
client-web-xfce IN A 192.168.11.28
srv-haproxy IN  A 192.168.11.29
mariadb-slave IN A 192.168.11.30
```

Test de connectivité avec nslookup :

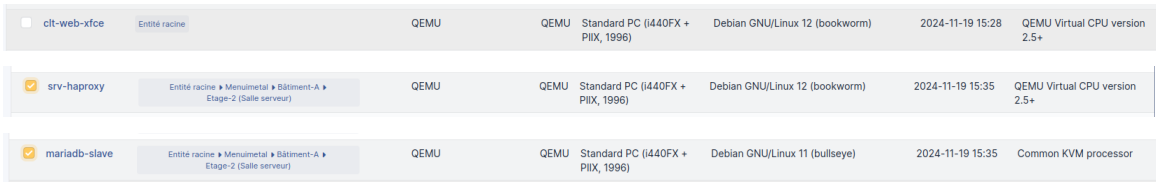
```
root@dns:~# nslookup
> web1
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   web1.menuimetal.fr
Address: 192.168.11.25
> web2
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   web2.menuimetal.fr
Address: 192.168.11.26
> web3
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   web3.menuimetal.fr
Address: 192.168.11.27
>
```

```
root@dns:~# nslookup
> client-web-xfce
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   client-web-xfce.menuimetal.fr
Address: 192.168.11.28
> srv-haproxy
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   srv-haproxy.menuimetal.fr
Address: 192.168.11.29
> mariadb-slave
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   mariadb-slave.menuimetal.fr
Address: 192.168.11.30
>
```

Intégration de toutes les VMs sur le Serveur GLPI pour Suivi et Administration :

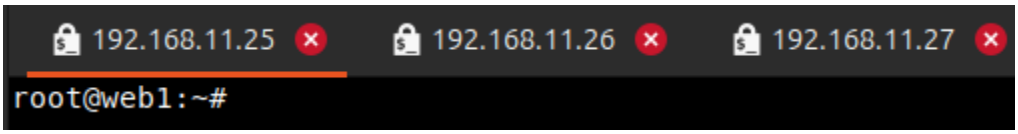


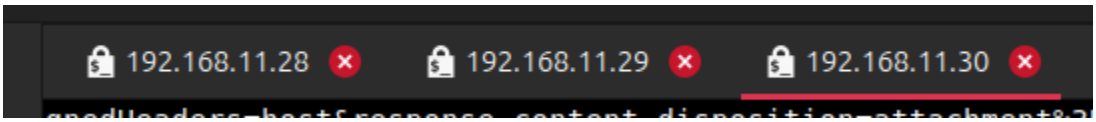
web1	Entité racine ▸ Menuimetal ▸ Bâtiment-A ▸ Etage-2 (Salle serveur)	QEMU	QEMU	Standard PC (i440FX + PIIX, 1996)	Debian GNU/Linux 11 (bullseye)	2024-11-19 14:41	Common KVM processor
web2	Entité racine ▸ Menuimetal ▸ Bâtiment-A ▸ Etage-2 (Salle serveur)	QEMU	QEMU	Standard PC (i440FX + PIIX, 1996)	Debian GNU/Linux 11 (bullseye)	2024-11-19 14:41	Common KVM processor
web3	Entité racine ▸ Menuimetal ▸ Bâtiment-A ▸ Etage-2 (Salle serveur)	QEMU	QEMU	Standard PC (i440FX + PIIX, 1996)	Debian GNU/Linux 11 (bullseye)	2024-11-19 14:41	Common KVM processor



clt-web-xfce	Entité racine	QEMU	QEMU	Standard PC (i440FX + PIIX, 1996)	Debian GNU/Linux 12 (bookworm)	2024-11-19 15:28	QEMU Virtual CPU version 2.5+
srv-haproxy	Entité racine ▸ Menuimetal ▸ Bâtiment-A ▸ Etage-2 (Salle serveur)	QEMU	QEMU	Standard PC (i440FX + PIIX, 1996)	Debian GNU/Linux 12 (bookworm)	2024-11-19 15:35	QEMU Virtual CPU version 2.5+
mariadb-slave	Entité racine ▸ Menuimetal ▸ Bâtiment-A ▸ Etage-2 (Salle serveur)	QEMU	QEMU	Standard PC (i440FX + PIIX, 1996)	Debian GNU/Linux 11 (bullseye)	2024-11-19 15:35	Common KVM processor

Accéder à la console des VMs via SSH :





V. MISSION 1 : REPLICATION DU SERVEUR MYSQL

Pourquoi monsieur Lepage a intérêt d'après vous de mettre en place la réplication ?

Afin d'améliorer la tolérance aux pannes pour garantir une disponibilité maximale de son service Web, M. Lepage a tout intérêt à mettre en place la réplication pour permettre un accès continu aux BDD.

Qu'est-ce qu'une réplication ?

Une réplication est un mécanisme permettant de copier et synchroniser les données entre plusieurs serveurs. Cela permet de garantir une haute disponibilité et de distribuer la charge de travail.

Quel résultat doit-être attendu ?

Le résultat attendu doit être une augmentation de la vitesse de traitement des informations ainsi qu'une disponibilité assurée même en cas de panne.
En cas de modification sur le serveur master, les modifications sont également appliquées sur le serveur slave.

A. Sauvegarder toutes les bases de données via mysqldump

Les étapes pour mettre en place la réplication Master/Slave :

- Sur le serveur maître (celui qui contient toutes les bases de données) :

  On configure le serveur en maître (fichier **/etc/mysql/mariadb.conf.d/50-server.cnf**) :

```
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
```

  On crée un utilisateur pour la réplication :

```
_repl'; [(none)]> CREATE USER 'repl'@'%' IDENTIFIED BY 'password
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)
```

```
MariaDB [(none)]> GRANT REPLICATION CLIENT ON *.* TO 'repl'@'%';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)
```

On verrouille les tables :

```
es tables(none)]> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.039 sec)
```

On récupère l'ID du log binaire :

```
MariaDB [(none)]> SHOW MASTER STATUS;
Empty set (0.000 sec)
```

Au début, on ne peut pas récupérer l'ID du log binaire car la commande **SHOW MASTER STATUS** ne renvoie rien.

On ajoute cette ligne dans le fichier **/etc/mysql/mariadb.conf.d/50-server.cnf** :

```
binlog_format = ROW
```

```
root@mariadb:/var/log/mysql# ls /var/log/mysql/
error.log  error.log.1.gz  error.log.2.gz  error.log.3.gz  error.log.4.gz  error.log.5.gz  error.log.6.gz  error.log.7.gz  mysql-bin.000001  mysql-bin.index
```

Le fichier **/var/log/mysql/mysql-bin.000001** est bien généré, on peut récupérer l'ID du log binaire :

```
MariaDB [(none)]> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 |      328 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.000 sec)
```

Sauvegarde :

mysqldump -u root -p --all-databases > save_bdd_19_11_2024.sql

```
root@mariadb:~# mysqldump -u root -p --all-databases > save_bdd_19_11_2024.sql
Enter password:
root@mariadb:~# ls
glpi-agent-1.7.1-linux-installer.pl  save_bdd_19_11_2024.sql
```

Envoi de la sauvegarde depuis le serveur maître **mariadb** :

scp save_bdd_19_11_2024.sql sio21@192.168.11.30:/home/sio21

```
root@mariadb:~# scp save_bdd_19_11_2024.sql sio21@192.168.11.30:
/home/sio21
sio21@192.168.11.30's password:
save_bdd_19_11_2024.sql       100% 7435KB 103.4MB/s   00:00
```

```
sio21@mariadb-slave:~$ ls
save_bdd_19_11_2024.sql
```

On déverrouille les tables :

```
MariaDB [(none)]> UNLOCK TABLES;
Query OK, 0 rows affected (0.000 sec)
```

B. Sur le serveur slave

On configure le serveur en slave (**fichier /etc/mysql/mariadb.conf.d/50-server.cnf**) :

```
server-id               = 2
log_bin                 = /var/log/mysql/mysql-bin.log
relay_log               = /var/log/mysql/mysql-relay-bin.log
expire_logs_days        = 10
```

Importation de la sauvegarde dans **mariadb-slave** : **mysql -u root -p < /home/sio21/save_bdd_19_11_2024.sql :**

```
root@mariadb-slave:~# mysql -u root -p < /home/sio21/save_bdd_19_11_2024.sql
Enter password:
root@mariadb-slave:~# echo $?
0
```

On peut noter que l'importation est un succès

Puis, on configure la connexion au master :

CHANGE MASTER TO
MASTER_HOST='192.168.11.1',
MASTER_USER='repl',
MASTER_PASSWORD='password_repl',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=328;

```
MariaDB [(none)]> CHANGE MASTER TO
    -> MASTER_HOST='192.168.11.1',
    -> MASTER_USER='repl',
    -> MASTER_PASSWORD='password_repl',
    -> MASTER_LOG_FILE='mysql-bin.000001',    MASTER_LOG_POS=328;
Query OK, 0 rows affected (0.028 sec)
```

```
MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.001 sec)
```

```
MariaDB [(none)]> SHOW SLAVE STATUS\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.11.1
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000001
          Read_Master_Log_Pos: 328
               Relay_Log_File: mysql-relay-bin.000002
                Relay_Log_Pos: 555
        Relay_Master_Log_File: mysql-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

Les champs **Slave_IO_Running** et **Slave_SQL_Running** affichent “Yes” donc le serveur SLAVE est bien lié au serveur MASTER.

Réaliser les tests en créant une base de données avec deux tables sur le serveur maître et vérifier si elle se retrouve sur le slave :

Création de la table sur le serveur MASTER :

```
MariaDB [(none)]> USE dokuwiki;
Database changed
MariaDB [dokuwiki]> CREATE TABLE test_slave (colonne1 VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.168 sec)
```

Vérification sur le serveur SLAVE :

```
MariaDB [(none)]> USE dokuwiki;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [dokuwiki]> SHOW TABLES;
+--------------------+
| Tables_in_dokuwiki |
+--------------------+
| test_slave         |
+--------------------+
1 row in set (0.000 sec)
```

C. Remonter les Logs des Serveurs vers le Serveur rsyslog

Les logs sont déjà remontés pour le serveur **mariadb** dans l'AP9. Notre serveur **mariadb-slave** étant un clone du serveur mariadb, il est également référencé.

On vérifie dans le fichier **/var/log/syslog** de notre serveur rsyslog :

```
root@srv-rsyslog-debian:/var/log# tail syslog
2024-11-21T15:25:46+01:00 mariadb-slave systemd[1]: Mounting Arbitrary Executable File Formats File System...
2024-11-21T15:25:46+01:00 mariadb-slave systemd[1]: Mounted Arbitrary Executable File Formats File System.
2024-11-21T15:25:55+01:00 mariadb-slave glpi-agent[407]: [info] target server0: next run: Fri Nov 22 15:21:01 2024 - http://192.168.13.1/
2024-11-21T15:34:01+01:00 mariadb-slave CRON[8602]: (daemon) CMD (test -x /usr/bin/debsecan && /usr/bin/debsecan --cron)
2024-11-21T16:13:01+01:00 mariadb CRON[8547]: (daemon) CMD (test -x /usr/bin/debsecan && /usr/bin/debsecan --cron)
2024-11-21T16:17:01.128812+01:00 srv-rsyslog-debian CRON[1310]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)
2024-11-21T16:17:01+01:00 mariadb CRON[8554]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
2024-11-21T16:17:01+01:00 mariadb-slave CRON[8619]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
2024-11-21T16:24:36.272093+01:00 srv-rsyslog-debian qemu-ga: info: guest-ping called
2024-11-21T16:24:47.091969+01:00 srv-rsyslog-debian qemu-ga: info: guest-ping called
root@srv-rsyslog-debian:/var/log#
```

On voit bien la communication entre le serveur **mariadb** (MASTER) et le serveur **mariadb-slave** (SLAVE).

VI. Mission 2 : Mise en place d'une répartition de charges sur un serveur Web avec HaProxy

L'intérêt de la répartition de charges :

- La répartition de charges optimise l'utilisation des ressources serveurs en dirigeant les requêtes des utilisateurs vers plusieurs serveurs Web.

A. Configuration de la page d'accueil pour les 3 serveur web

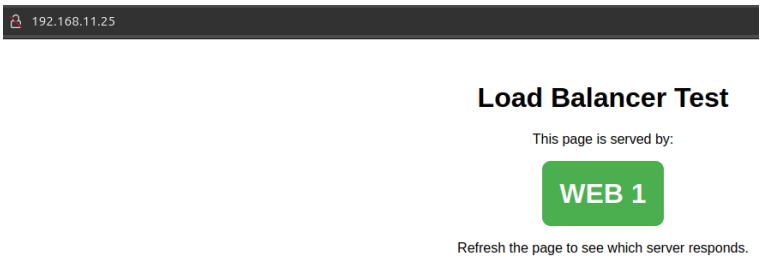
Ici, nous allons créer ou modifier le fichier index.html de chaque serveur et les différencier en référençant le nom du serveur associé :

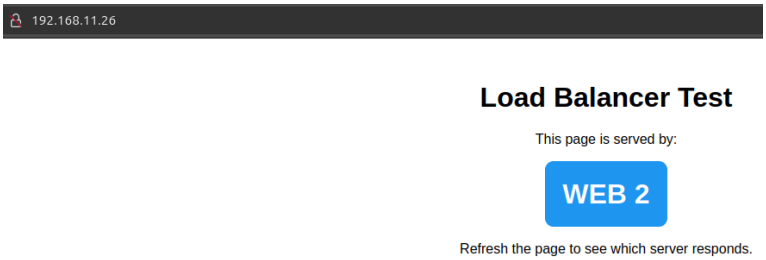
Nous allons éditer le fichier index.html dans le répertoire **`/var/www/html/index.html`** sur les serveurs web 1, 2 et 3 :

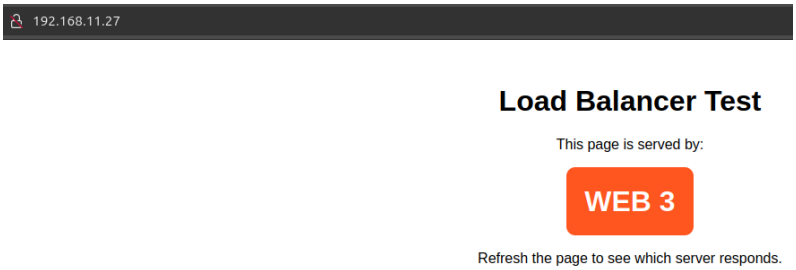
```
root@web1:~# vim /var/www/html/index.html
```

```
<h1>Load Balancer Test</h1>
<p>This page is served by:</p>
<div class="server server2">WEB 2</div>
<!-- Change "server3" to "server1" or "se
```

Visualisation de la Page Web dans le Navigateur :







B. Installation et configuration de haproxy

```
apt update && sudo apt upgrade
```

```
apt install haproxy -y
```

Configuration d'Haproxy :

```
vim /etc/haproxy/haproxy.cfg
```

Configuration du Fronted :

```
frontend front_webservers
        bind *:80
        default_backend backend_webservers
        option forwardfor
```

Cette configuration permettra de gérer les **requêtes HTTP des clients** avec HAProxy, qui fonctionnera comme un proxy inversé. HAProxy recevra **les requêtes sur le port 80**, le port standard pour HTTP. Grâce à la directive **`bind *:80`**, il acceptera les connexions sur toutes les adresses IP configurées sur le serveur. Ensuite, les requêtes seront envoyées à un groupe de serveurs backend, grâce à la directive **`default_backend backend_webservers`,** pour être traitées par les serveurs qui hébergent l'application ou le site web.

Configuration du backend :

```
backend backend_webservers
        balance roundrobin
        server  web1 192.168.11.25:80 check
        server  web2 192.168.11.26:80 check
        server  web3 192.168.11.27:80 check
~
```

Cette configuration permettra de répartir les requêtes entrantes de manière équilibrée entre les serveurs **srv-web-1** et **srv-web-2** grâce à la directive **`balance roundrobin`** de HAProxy. Cela signifie que la première requête sera envoyée à **srv-web-1**, la suivante à **srv-web-2**, et ainsi de suite, en alternance. Ce processus se répétera en boucle. Cette répartition des charges permet d'éviter que l'un des serveurs soit trop sollicité, ce qui améliore les performances globales du système en équilibrant la charge de travail entre les différents serveurs.

```
root@srv-haproxy:~# systemctl restart haproxy
```

Vérification du Statut du Service Haproxy :

```
root@srv-haproxy:~# systemctl status haproxy*
● haproxy.service - HAProxy Load Balancer
     Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-11-21 09:01:35 CET; 1min 17s ago
       Docs: man:haproxy(1)
             file:/usr/share/doc/haproxy/configuration.txt.gz
   Main PID: 5902 (haproxy)
      Tasks: 2 (limit: 2306)
     Memory: 43.6M
        CPU: 152ms
     CGroup: /system.slice/haproxy.service
             ├─5902 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/h>
             └─5904 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/h>

nov. 21 09:01:35 srv-haproxy systemd[1]: Starting haproxy.service - HAProxy Load Balancer...
nov. 21 09:01:35 srv-haproxy haproxy[5902]: [NOTICE]   (5902) : New worker (5904) forked
nov. 21 09:01:35 srv-haproxy haproxy[5902]: [NOTICE]   (5902) : Loading success.
nov. 21 09:01:35 srv-haproxy systemd[1]: Started haproxy.service - HAProxy Load Balancer.
lines 1-17/17 (END)
```

Test du fonctionnement du service HAProxy en accédant à l'adresse IP :

```
root@srv-haproxy:~# ip a | grep "192.168.11.29"
    inet 192.168.11.29/24 brd 192.168.11.255 scope global ens18
```

IP du serveur Haproxy

Load Balancer Test

This page is served by:

WEB 1

Refresh the page to see which server responds.

Load Balancer Test

This page is served by:

WEB 2

Refresh the page to see which server responds.

Load Balancer Test

This page is served by:

WEB 3

Refresh the page to see which server responds.

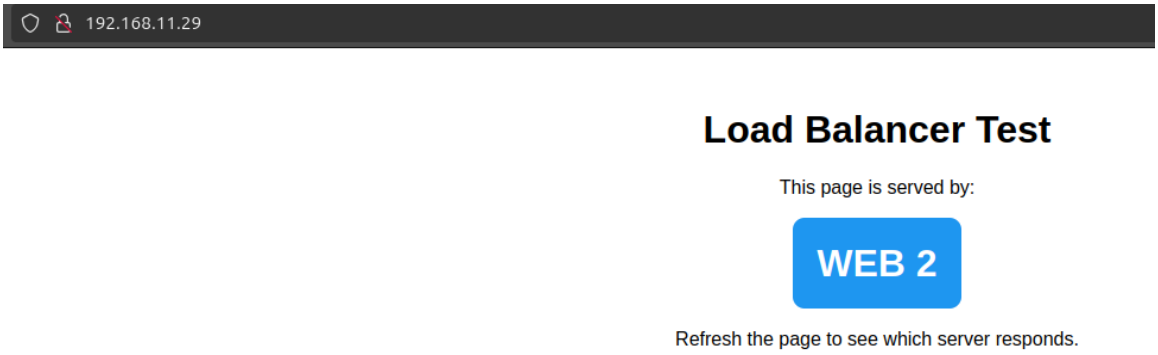
1. Vérification du Fonctionnement du Balancer Haproxy

Nous allons désactiver le service Apache2 sur le serveur web 1 :

```
root@web1:~# systemctl stop apache2.service
```

```
root@web1:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Thu 2024-11-21 11:25:50 CET; 2min 23s ago
```

Test du balancer :



Ici, on constate que l'IP ou l'URL renvoie directement à la page web du serveur 2, donc le balancement fonctionne correctement.

2. Configuration de l'Interface Web pour Visualiser les Statistiques Haproxy

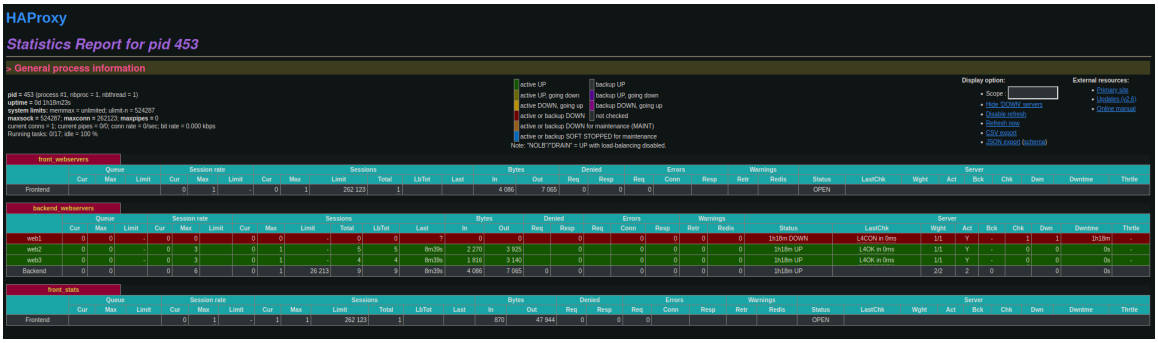
```
frontend front_stats
        bind *:1480
        stats enable
        stats uri /stats
        stats hide-version
        stats refresh 10s
        stats realm Haproxy\ Statistics
        stats auth haproxy:P@ssword!
```

Dans la configuration effectuée dans l'image, les logins vont être :

- haproxy
- P@ssword!

Accès à l'Interface de Statistiques Haproxy :

http://192.168.11.29:1480/stats



a) Commentaires sur les Statistiques d'Haproxy via l'Interface Web

General process information :

Cette section donne un aperçu général du processus HAProxy :

- **PID** : Le numéro d'identification du processus HAProxy.
- **Temps d'activité (uptime)** : Montre depuis combien de temps HAProxy fonctionne.
- **Sessions actives** : Indique combien de connexions sont actuellement gérées. Cela inclut à la fois les connexions aux frontends et celles redirigées vers les backends.
- **Mémoire utilisée** : Permet de suivre l'impact de HAProxy sur les ressources système.

Frontend_webservers :

Les frontends définissent les points d'entrée pour les requêtes des clients.

- **Colonne Sessions** :
    - **Actives (courantes)** : Nombre de connexions en cours.
    - **Maximales** : Pic de connexions simultanées enregistré depuis le démarrage.
    - **Cumulatives** : Total des connexions ayant transité par le frontend.
- **Colonne Req_rate** :
    - Indique le taux de requêtes par seconde sur ce frontend.
- **Colonne Status** :
    - L'état "OPEN" indique que le frontend accepte activement des connexions.
    - Si un frontend est "FULL" ou "CLOSED", cela signifie qu'il atteint sa capacité maximale ou qu'il est désactivé.

Backend_webservers :

Les backends regroupent les serveurs où HAProxy redirige les requêtes après traitement initial.

Informations importantes à analyser :

- **Colonne L7OK** :
    - Nombre de réponses HTTP 2xx (succès) et 3xx (redirections). Une valeur élevée est un bon indicateur de fonctionnement.
- **Colonnes Warnings/Errors** :

- **Warnings** : Les avertissements peuvent indiquer des anomalies (ex. : dépassements de délais ou connexions instables).
- **Errors** : Les erreurs comme les HTTP 5xx signalent des problèmes côté application sur les serveurs backend.

Front_stats :

État : OPEN
Le frontend **`front_stats`** est en état **OPEN**, ce qui signifie qu'il accepte actuellement des connexions. Cela indique que l'interface des statistiques HAProxy est accessible pour la surveillance et l'administration.

Colonne Sessions :

- **Actives (courantes)** : Nombre de connexions en cours. Ce nombre est généralement faible, car ce frontend est utilisé principalement pour des accès administratifs.
- **Maximales** : Pic de connexions simultanées enregistré depuis le démarrage.
- **Cumulatives** : Total des connexions ayant transité par ce frontend depuis le démarrage.

Colonne Req_rate :

- Indique le taux de requêtes par seconde sur ce frontend. Ce taux est habituellement faible en raison du peu de connexions nécessaires pour accéder à l'interface des statistiques.

<u>Serveurs individuels (dans les backends) :</u>

Serveur 1 (web server 1) :

- État : DOWN
  Ce serveur est hors service. HAProxy ne parvient pas à établir de

connexion avec lui ou les tests de santé (*health checks*) ont échoué. Cela peut être causé par une panne, une surcharge ou un problème de réseau.

Serveur 2 (web server 2) :

- État : UP
  Le serveur fonctionne correctement et gère des connexions sans problème apparent. Aucun avertissement, erreur ou surcharge n'est signalé, ce qui indique qu'il est pleinement opérationnel.

Serveur 3 (web server 3) :

- État : UP
  Ce serveur est également en ligne et opérationnel.

Actuellement, **webserver2** et **webserver3** gèrent correctement le trafic, mais l'indisponibilité de **webserver1** pourrait entraîner une augmentation de la charge sur les deux autres serveurs. Il est donc essentiel de rétablir **webserver1** pour maintenir un bon équilibre de charge et éviter toute saturation des serveurs restants.

3. Mise en Place d'une ACL sur Haproxy

Ici, l'objectif est d'utiliser un nom de domaine afin de ne pas avoir à utiliser l'IP du serveur.

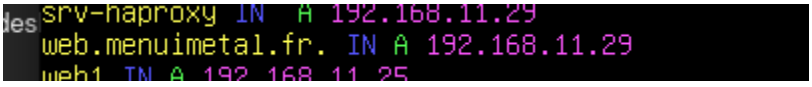
```
frontend front_webservers
        bind *:80
        default_backend backend_webservers
        option forwardfor
        # ACL pour app.it-connect.local"
        acl ACL_app.it-connect.local hdr(host) -i app.it-connect.local
        use_backend backend_webservers if ACL_app.it-connect.local
        option forwardfor
```

Configuration de l 'ACL

L'image montre un message d'erreur lorsque nous tentons d'accéder à la page web **`web.meniumetal.fr`**. Ce message indique un problème lié à la configuration du DNS ou à la résolution de l'adresse IP. Cela signifie que le serveur DNS n'a pas encore l'enregistrement **A** pour ce nom de domaine, ce qui empêche la résolution correcte du domaine vers l'adresse IP correspondante.

Dans le fichier **db.menuimetal** on va rajouter ceci dans le DNS :
web.menuimetal.fr IN A 192.168.11.29

Puis nous allons effectué un test de connectivité avec nslookup :



```
root@dns:~# nslookup web.menuimetal.fr
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   web.menuimetal.fr
Address: 192.168.11.29
Name:   web.menuimetal.fr
Address: 192.168.12.2

root@dns:~# _
```

Ici, on peut constater que le nom de domaine possède deux adresses IP. En effet , on constate que dans le DNS, **web** et **web.menuimetal.fr** sont considérés comme le même nom. J'ai donc attribué deux adresses IP différentes à un même nom de machine.

```
web     IN      A       192.168.12.2
```

```
srv-haproxy IN  A 192.168.11.29
web.menuimetal.fr. IN A 192.168.11.29
web1 IN A 192.168.11.25
```

Correction :

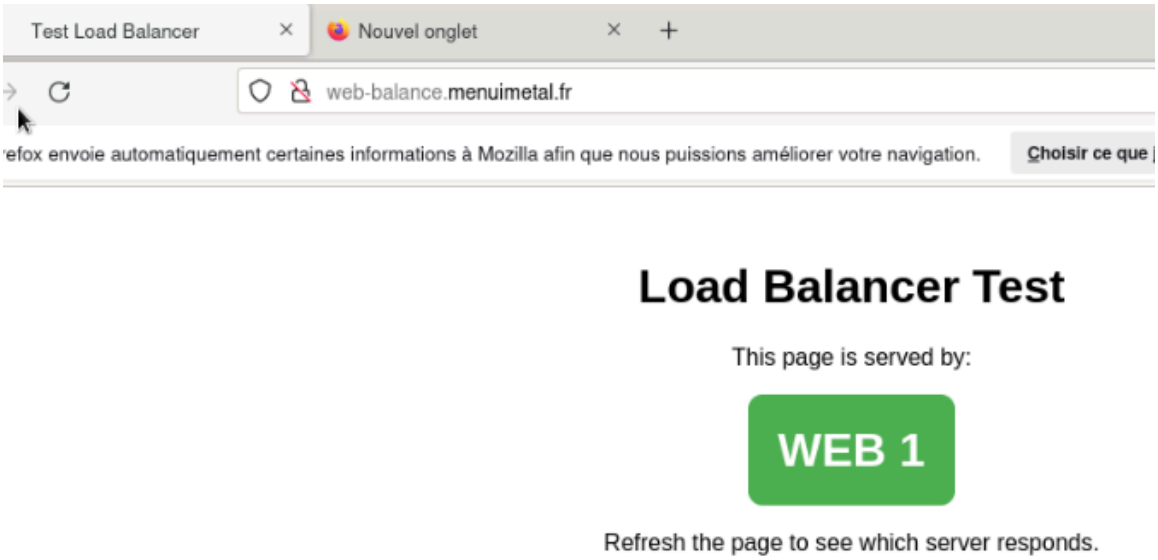
```
srv-haproxy IN  A 192.168.11.29
web-balance.menuimetal.fr. IN A 192.168.11.29
web1 IN A 192.168.11.25
```

```
root@dns:~# nslookup web-balance.menuimetal.fr
Server:         192.168.12.1
Address:        192.168.12.1#53

Name:   web-balance.menuimetal.fr
Address: 192.168.11.29

root@dns:~# _
```

Nous allons configurer le fichier **`haproxy.cfg`** pour attribuer une URL correcte :

```
frontend front_webservers
        bind *:80
        default_backend backend_webservers
        # ACL pour web-balance.menuimetal.fr
        acl acl_web_balance hdr(host) -i web-balance.menuimetal.fr
        use_backend backend_webservers if acl_web_balance
        option forwardfor
```

Test de l'URL sur le client debian :



4. Supervision Complète de Haproxy sur Nagios : État, Charge, Réseau et Disque

a) Configuration de Nagios pour Intégrer la Machine et Surveiller ses Services

```
root@srv-nagios:~# vim /usr/local/nagios/etc/servers/haproxy.cfg
```

```
# Nagios Host configuration file template
   define host {
       use                             linux-server
       host_name                       srv-haproxy
       alias                           srv-haproxy
       address                         192.168.11.29
       register                        1
   }
~
```

Configuration du coté du serveur haproxy :

```
root@srv-haproxy:~# apt install nagios-nrpe-server nagios-plugins
```

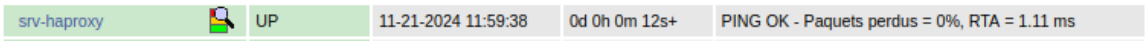
```
root@srv-haproxy:~# vim /etc/nagios/nrpe.cfg
```

```
allowed_hosts=127.0.0.1,192.168.13.2
```

Test de connectivité depuis le serveur nagios :

```
root@srv-nagios:~# /usr/local/nagios/libexec/check_nrpe -H 192.168.11.29
NRPE v4.1.0
root@srv-nagios:~#
```

Vérification de la présence de la machine sur le serveur nagios :



Configuration des services à surveiller :

Définition du service dans :

```
vim /usr/local/nagios/etc/servers/haproxy.cfg
```

Contrôle de l'État (Allumé/Éteint) du Serveur Haproxy sur Nagios :

```
# État de fonctionnement (allumé/éteint)
            define service {
    host_name                       srv-haproxy
    service_description             PING
    check_command                   check_ping!100.0,20%!500.0,60%
    max_check_attempts              2
    check_interval                  2
    retry_interval                  2
    check_period                    24x7
    check_freshness                 1
    contact_groups                  admins
    notification_interval           2
    notification_period             24x7
    notifications_enabled           1
    register                        1
}
```

Contrôle de la Charge Système du Serveur Haproxy via Nagios :

```
# charge du système

define service {
    host_name                       srv-haproxy
    service_description             Load Average
    check_command                   check_local_load!5,3,1!10,5,3
    max_check_attempts              3
    check_interval                  5
    retry_interval                  1
    check_period                    24x7
    contact_groups                  admins
    notification_interval           30
    notification_period             24x7
    notifications_enabled           1
    register                        1
}
```

Surveillance de l'État des Interfaces Réseau (Up/Down) du Serveur Haproxy dans Nagios :

On se deplace dans vim /usr/local/nagios/etc/objects/commands.cfg :

```
root@srv-nagios:~# vim /usr/local/nagios/etc/objects/commands.cfg
```

```
# etat des interfaces réseau

define command {
    command_name    check_ifoperstatus
    command_line    $USER1$/check_ifoperstatus -i $ARG1$
}
```

```
# État des interfaces réseau (up/down)
define service {
    host_name                       srv-haproxy
    service_description             Network_Interface_ens18
    check_command                   check_ifoperstatus!ens18
    max_check_attempts              2
    check_interval                  2
    retry_interval                  2
    check_period                    24x7
    check_freshness                 1
    contact_groups                  admins
    notification_interval           30
    notification_period             24x7
    notifications_enabled           1
    register                        1
}
```

On a constaté un problème avec l'interface. En effet, nous avons mis **eth0** par défaut, donc nous avons modifié le code pour qu'il regarde au niveau de **ens18.**

Surveillance du Taux de Remplissage du Disque Dur du Serveur Haproxy dans Nagios :

```
define service {
    host_name                       srv-graylog
    service_description             Local Disk
    check_command                   check_local_disk!20%!10%!/
    max_check_attempts              2
    check_interval                  2
    retry_interval                  2
    check_period                    24x7
    check_freshness                 1
    contact_groups                  admins
    notification_interval           2
    notification_period             24x7
    notifications_enabled           1
    register                        1
}
~
```

Surveillance de l'État du Service Haproxy dans Nagios :

Installation du check spécifique à haproxy :
https://raw.github.com/benprew/nagios-checks/master/check_haproxy.rb

```
root@srv-nagios:~# apt install ruby-full
```

```
root@srv-nagios:~# wget https://raw.github.com/benprew/nagios-checks/master/check_haproxy.rb -O /usr
/local/nagios/libexec/check_haproxy.rb
--2024-11-21 15:59:38--  https://raw.github.com/benprew/nagios-checks/master/check_haproxy.rb
Connexion à 172.16.0.51:8080… connecté.
requête Proxy transmise, en attente de la réponse… 301 Moved Permanently
Emplacement : https://raw.githubusercontent.com/benprew/nagios-checks/master/check_haproxy.rb [suiva
nt]
--2024-11-21 15:59:39--  https://raw.githubusercontent.com/benprew/nagios-checks/master/check_haprox
y.rb
Connexion à 172.16.0.51:8080… connecté.
requête Proxy transmise, en attente de la réponse… 200 OK
Taille : 6583 (6,4K) [text/plain]
Sauvegarde en : « /usr/local/nagios/libexec/check_haproxy.rb »

/usr/local/nagios/libexe 100%[=================================>]   6,43K  --.-KB/s    ds 0s

2024-11-21 15:59:40 (29,4 MB/s) – « /usr/local/nagios/libexec/check_haproxy.rb » sauvegardé [6583/65
83]
```

Vérification de la Présence du Plugin Haproxy dans Nagios :

```
root@srv-nagios:~# ls /usr/local/nagios/libexec | grep "haproxy"
check_haproxy.rb
```

Intégration des Commandes dans `commands.cfg` pour Suivre l'État du Service Haproxy :

```
# haproxy service

define command {
    command_name    check_haproxy_status
    command_line    /usr/bin/ruby /usr/local/nagios/libexec/check_haproxy.rb --url http://192.168.11
.29:1480/stats --user haproxy --password P@ssword!
}
```

Ajout de la Surveillance du Statut du Service Haproxy :

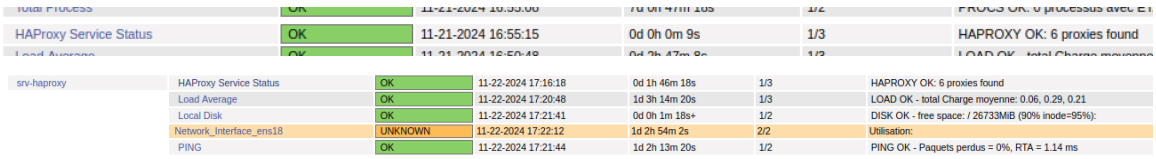
```
# statut haproxy service

define service {
    host_name                       srv-haproxy
    service_description             HAProxy Service Status
    check_command                   check_haproxy_status
    max_check_attempts              3
    check_interval                  5
    retry_interval                  1
    check_period                    24x7
    check_freshness                 1
    contact_groups                  admins
    notification_interval           30
    notification_period             24x7
    notifications_enabled           1
    register                        1
}
```

Test de la Commande en Ligne pour Vérifier le Statut du Service Haproxy :

root@srv-nagios:~# /usr/bin/ruby /usr/local/nagios/libexec/check_haproxy.rb --url http://192.168.11.29:1480/stats --user haproxy --password P@ssword!

```
HAPROXY OK: 6 proxies found
backend_webservers BACKEND (act) UP     sess=0/26213(0%) smax=1
backend_webservers web1 (act) UP        sess=0/-1(0%) smax=1
backend_webservers web2 (act) UP        sess=0/-1(0%) smax=1
backend_webservers web3 (act) UP        sess=0/-1(0%) smax=1
front_stats FRONTEND (act) OPEN sess=1/262123(0%) smax=2
front_webservers FRONTEND (act) OPEN    sess=1/262123(0%) smax=1
root@srv-nagios:~#
```

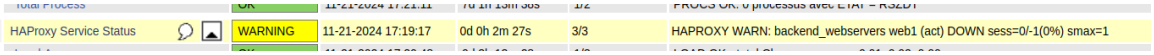
Vérification de la Présence des Service surveillant Haproxy sur le Site Nagios :



HAProxy Service Status	OK	11-21-2024 16:55:15	0d 0h 0m 9s	1/3	HAPROXY OK: 6 proxies found	
srv-haproxy	HAProxy Service Status	OK	11-22-2024 17:16:18	0d 1h 46m 18s	1/3	HAPROXY OK: 6 proxies found
	Load Average	OK	11-22-2024 17:20:48	1d 3h 14m 20s	1/3	LOAD OK - total Charge moyenne: 0.06, 0.29, 0.21
	Local Disk	OK	11-22-2024 17:21:41	0d 0h 1m 18s+	1/2	DISK OK - free space: / 26733MiB (90% inode=95%):
	Network_Interface_ens18	UNKNOWN	11-22-2024 17:22:12	1d 2h 54m 2s	2/2	Utilisation:
	PING	OK	11-22-2024 17:21:44	1d 2h 13m 20s	1/2	PING OK - Paquets perdus = 0%, RTA = 1.14 ms

5. Simulation de Panne des Serveurs Web pour Tester le Service Haproxy

Simulation de Panne du Serveur Web1 :
Observation des Résultats en Interface Graphique et Console:



HAProxy Service Status	WARNING	11-21-2024 17:19:17	0d 0h 2m 27s	3/3	HAPROXY WARN: backend_webservers web1 (act) DOWN sess=0/-1(0%) smax=1

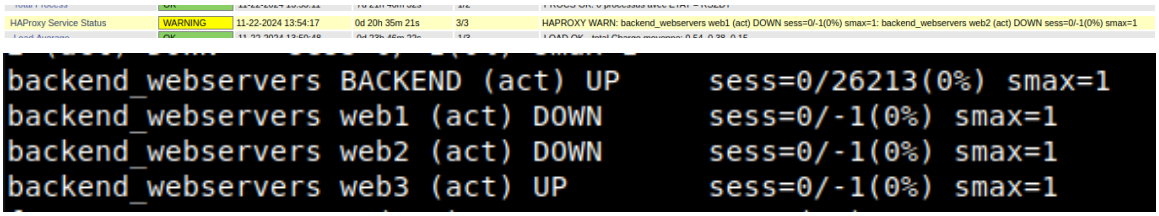
```
root@srv-nagios:~# /usr/bin/ruby /usr/local/nagios/libexec/check_haproxy.rb -
29:1480/stats --user haproxy --password P@ssword!
HAPROXY WARN: backend_webservers web1 (act) DOWN        sess=0/-1(0%) smax=1
backend_webservers BACKEND (act) UP     sess=0/26213(0%) smax=1
backend_webservers web1 (act) DOWN      sess=0/-1(0%) smax=1
backend_webservers web2 (act) UP        sess=0/-1(0%) smax=1
backend_webservers web3 (act) UP        sess=0/-1(0%) smax=1
front_stats FRONTEND (act) OPEN sess=1/262123(0%) smax=2
front_webservers FRONTEND (act) OPEN    sess=0/262123(0%) smax=1
```

Remarque :
Dans l'interface graphique, un avertissement apparaît, indiquant que le serveur web 1 est hors ligne. En ligne de commande, cette information est également visible, car le statut du serveur est clairement mentionné.

Simulation de Panne du Serveur Web2 :

```
root@web2:~# systemctl stop apache2.service
root@web2:~#
```

Observation des Résultats en Interface Graphique et Console:

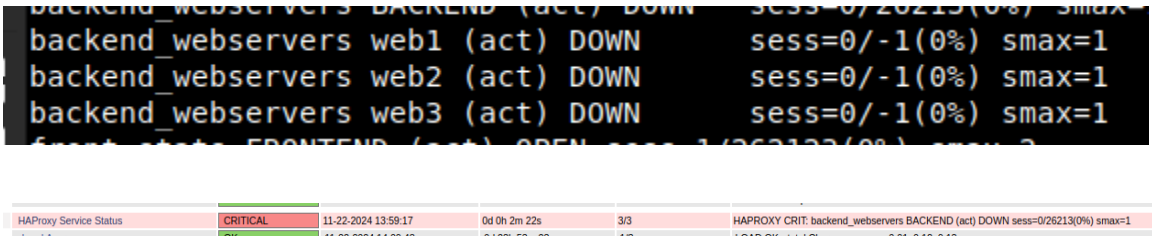


Remarque :
Dans l'interface graphique, un avertissement apparaît, indiquant que le serveur web 1 et web 2 sont hors ligne. En ligne de commande, cette information est également visible, car le statut des serveurs est mentionné.

Simulation de Panne du Serveur Web3 :

```
root@web3:~# systemctl stop apache2.service
root@web3:~#
```

Observation des Résultats en Interface Graphique et Console:



Ici on constate que lorsque les 3 services web sont hors ligne, on reçoit un message critique.

VII. Mission 3 : Mise en place d'un outil de haute disponibilité avec Heartbeat

Cloner le serveur Haproxy en modifiant son hostname et son IP dans le vlan LAN :

```
sio@srv-haproxy-clone:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
N group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_code
l state UP group default qlen 1000
    link/ether bc:24:11:8e:12:c4 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.11.32/24 brd 192.168.11.255 scope global ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:fe8e:12c4/64 scope link
       valid_lft forever preferred_lft forever
```

Remonter le deuxième serveur sur Nagios :

```
define host {
    use                     linux-server
    host_name               srv-haproxy-clone
    alias                   Haproxy Server Clone
    address                 192.168.11.32
}
```

srv-haproxy-clone	UP	11-21-2024 15:06:43	0d 0h 0m 19s+	PING OK - Paquets perdus = 0%, RTA = 0.84 ms

A. Installer le service Heartbeat sur les deux serveurs

Haproxy :

```
root@srv-haproxy:~# apt install haproxy
```

Configurer les fichiers ha.cf et haresources et expliquer leur rôle :

Le fichier **ha.cf** contient les paramètres de base pour la communication entre les nœuds :

```
logfile /var/log/heartbeat.log
logfacility daemon
node srv-haproxy
node srv-haproxy-clone
keepalive 1
deadtime 10
bcast ens18
#ping 192.168.1.1
auto_failback yes
```

Configuration de la clé d'authentification :

```
root@srv-haproxy:~# vim /etc/heartbeat/authkeys
```

```
auth 1
1 sha1 clef
```

Le fichier **/etc/heartbeat/haresources** définit les ressources qui seront gérées par Heartbeat (comme l'adresse IP virtuelle et le service HAProxy) :

```
root@srv-haproxy:~# vim /etc/heartbeat/haresources
```

```
srv-haproxy 192.168.11.33
```

Haproxy-Clone :

```
root@srv-haproxy-clone:~# apt install heartbeat
```

Configurer les fichiers ha.cf et haresources et expliquer leur rôle :

Le fichier **ha.cf** contient les paramètres de base pour la communication entre les nœuds :

```
logfile /var/log/heartbeat.log
logfacility daemon
node srv-haproxy
node srv-haproxy-clone
keepalive 1
deadtime 10
bcast ens18
#ping 192.168.1.1
auto_failback yes
```

Configuration de la clé d'authentification :

```
root@srv-haproxy-clone:~# vim /etc/heartbeat/authkeys
```

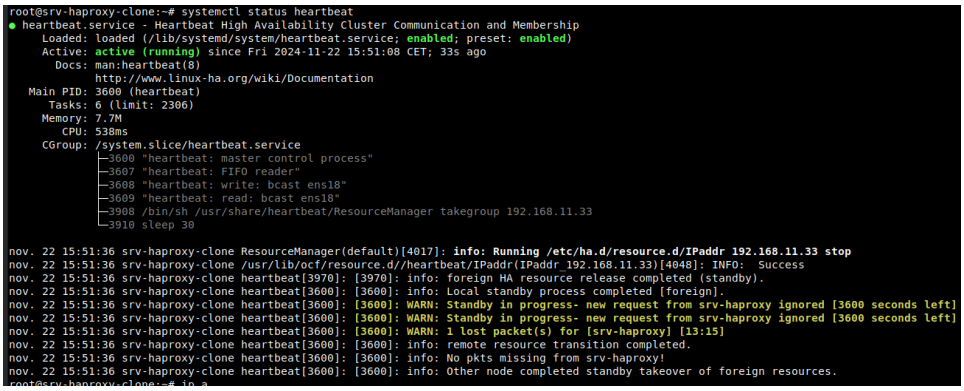
```
auth 1
1 sha1 clef
```

Le fichier **/etc/heartbeat/haresources** définit les ressources qui seront gérées par Heartbeat (comme l'adresse IP virtuelle et le service HAProxy) :

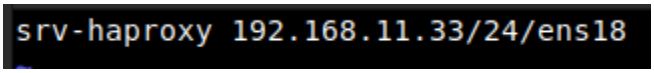
```
root@srv-haproxy-clone:~# vim /etc/heartbeat/haresources
```

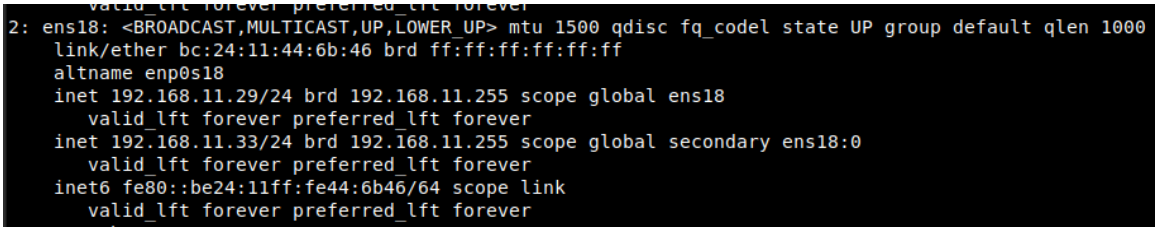
```
srv-haproxy 192.168.11.33
```

Pour les tests de fonctionnement, arrêter un des serveurs haproxy :

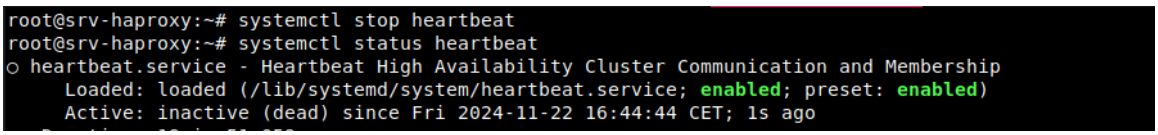


Ici, nous avons rencontré un problème car nous n'arrivions pas à obtenir l'IP virtuelle. Nous avons donc modifié le fichier de configuration sur les deux serveurs HAProxy.

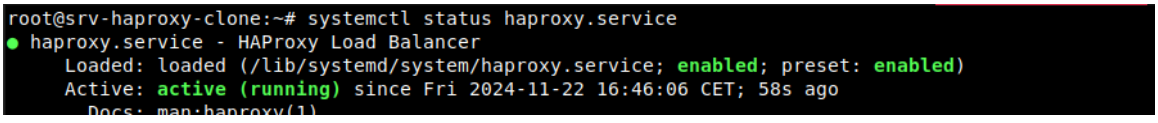




Par la suite, nous vérifions le bon fonctionnement du balancement en arrêtant le premier service HAProxy :

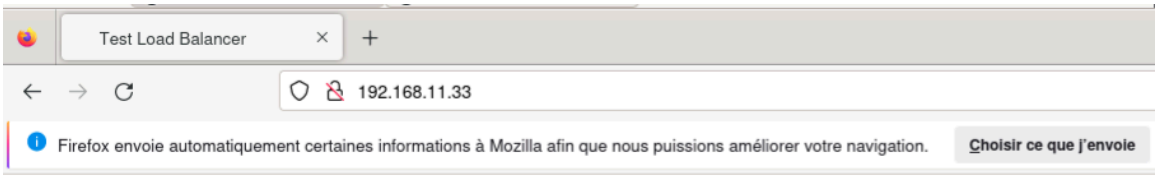


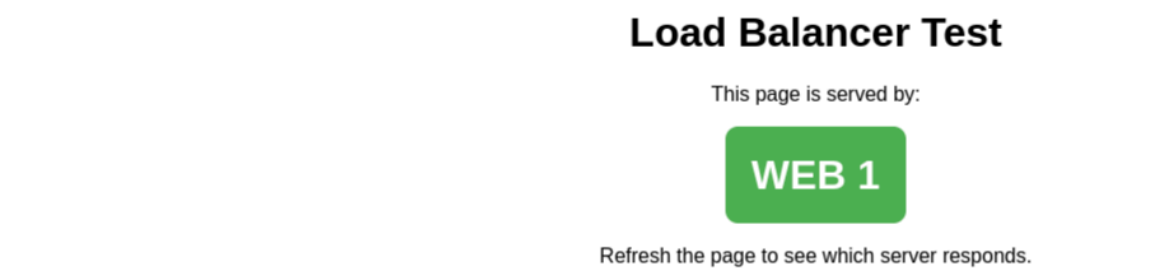
On voit bien que l’ip virtuelle bascule vers le 2eme serveur haproxy :

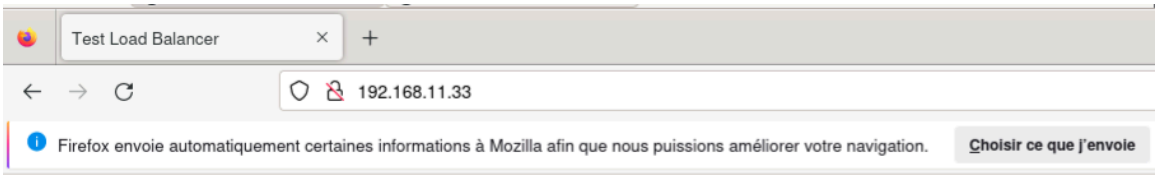


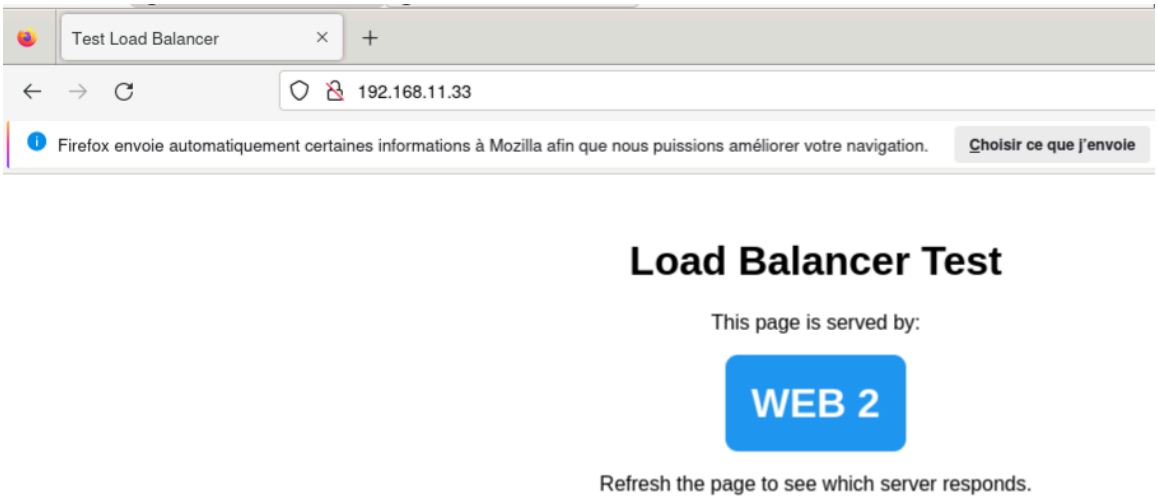
```
root@srv-haproxy-clone:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether bc:24:11:8e:12:c4 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.11.32/24 brd 192.168.11.255 scope global ens18
       valid_lft forever preferred_lft forever
    inet 192.168.11.33/24 brd 192.168.11.255 scope global secondary ens18:0
       valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:fe8e:12c4/64 scope link
       valid_lft forever preferred_lft forever
root@srv-haproxy-clone:~#
```

Vérification de l'Accessibilité du Service Web via l'IP Virtuelle :









B. Remonter les logs du service Heartbeat sur rsyslog et commenter les résultats en fonction des tests réalisés :

Serveur HaProxy Principal :

```
root@srv-haproxy:~# apt install rsyslog
```

On édite le fichier de configuration de Heartbeat **/etc/ha.d/ha.cf** pour associer les journaux de Heartbeat à la facility local5 :

```
logfacility local5
```

Sur le serveur Heartbeat, on crée un fichier pour envoyer les logs sur le serveur distant rsyslog:

```
root@srv-haproxy:~# vim /etc/rsyslog.d/heartbeat.conf
```

```
local5.* @192.168.11.23:514
```

Sur le serveur rsyslog on crée un fichier de configuration pour lui demander de rediriger les logs :

```
root@srv-rsyslog-debian:/var/log# vim /etc/rsyslog.d/heartbeat.conf
```

```
local5.* /var/log/heartbeat.conf
```

On vérifie dans le fichier **/var/log/heartbeat.conf** :

```
root@srv-rsyslog-debian:/var/log# tail /var/log/heartbeat.conf
2024-11-22T16:24:51+01:00 srv-haproxy heartbeat: [8298]: info: Comm_now_up(): updating status to active
2024-11-22T16:24:51+01:00 srv-haproxy heartbeat: [8298]: info: Local status now set to: 'active'
```

Serveur HaProxy Secondaire (Clone) :

```
root@srv-haproxy-clone:~# apt install rsyslog
```

On édite le fichier de configuration de Heartbeat **/etc/ha.d/ha.cf** pour associer les logs de Heartbeat à la facility local5 :

```
logfacility local5
```

Sur le serveur Heartbeat, on crée un fichier pour envoyer les logs sur le serveur distant rsyslog :

```
root@srv-haproxy-clone:~# vim /etc/rsyslog.d/heartbeat.conf
```

```
local5.* @192.168.11.23:514
```

Sur le serveur rsyslog on a crée avant un fichier de configuration pour lui demander de rediriger les logs :

```
root@srv-rsyslog-debian:/var/log# vim /etc/rsyslog.d/heartbeat.conf
```

```
local5.* /var/log/heartbeat.conf
```

On vérifie dans le fichier **/var/log/heartbeat.conf** :

```
root@srv-rsyslog-debian:/var/log# tail /var/log/heartbeat.conf
2024-11-22T16:14:28+01:00 srv-haproxy-clone heartbeat: [4554]: info: remote resource transition completed.
2024-11-22T16:14:28+01:00 srv-haproxy-clone heartbeat: [4554]: info: remote resource transition completed.
2024-11-22T16:14:28+01:00 srv-haproxy-clone heartbeat: [4554]: info: Local Resource acquisition completed. (none)
2024-11-22T16:14:28+01:00 srv-haproxy-clone heartbeat: [4554]: info: srv-haproxy wants to go standby [foreign]
2024-11-22T16:14:29+01:00 srv-haproxy-clone heartbeat: [4554]: info: standby: acquire [foreign] resources from srv-haproxy
2024-11-22T16:14:29+01:00 srv-haproxy-clone heartbeat: [4583]: info: acquire local HA resources (standby).
2024-11-22T16:14:29+01:00 srv-haproxy-clone heartbeat: [4583]: info: local HA resource acquisition completed (standby).
2024-11-22T16:14:29+01:00 srv-haproxy-clone heartbeat: [4554]: info: Standby resource acquisition done [foreign].
2024-11-22T16:14:29+01:00 srv-haproxy-clone heartbeat: [4554]: info: Initial resource acquisition complete (auto_failback)
2024-11-22T16:14:29+01:00 srv-haproxy-clone heartbeat: [4554]: info: remote resource transition completed.
```

Dans les logs de rsyslog, on voit bien que les 2 serveurs communiquent entre eux et que le cluster (**192.168.11.33**) prend le relais.

```
root@srv-haproxy:~# tail /var/log/heartbeat.log
```

```
nov. 22 16:59:44 srv-haproxy /usr/lib/ocf/resource.d//heartbeat/IPaddr(IPaddr_192.168.11.33)[8814]: INFO:  Resource is stopped
nov. 22 16:59:44 srv-haproxy ResourceManager(default)[8835]: info: Running /etc/ha.d/resource.d/IPaddr 192.168.11.33/24/ens18 start
```

On peut également voir dans les logs la génération de l’ip virtuelle .