MAO Pauline
OUDAR Nicolas
SIO2 SISR

Année 2023/2024

TP : Mise en place d'une haute disponibilité

Sommaire :

Contexte :

Notre objectif est de créer de la haute disponibilité sur Linux pour un service web. Ce service web sera installé sur une machine physique Debian qui sera le serveur principal de même que sur un serveur secondaire qui sera sur une VM, Debian également.

Mise en place des services web et des paquets :

Avant tout, il est nécessaire d'avoir de la connexion donc nous allons configurer les IPs sur nos deux serveurs :
Sur le serveur physique : 10.0.0.133/19
Sur la VM : 10.0.0.131/19
Passerelle par défaut : 10.0.0.253

Pour héberger notre serveur web, nous allons utiliser **Nginx** et prendre **Pacemaker**, **Corosync** et **PCS** *(remplacés par Keepalived plus bas !)* pour la gestion de cluster et la supervision de haute disponibilité.
En installant Nginx (*"apt install nginx"*), nous allons désinstaller et supprimer les fichiers Apache2 comme l'index.html pour éviter que sa page s'affiche sur le navigateur au lieu de Nginx (*seulement si dans l'installation de Debian on coche "installation serveur web" comme nous l'avons fait, sinon Apache2 n'est pas installé par défaut*).

Nous ferons ce procédé sur les deux serveurs. Sur un navigateur, pour nous, c'est Firefox, il faut voir si les pages s'affichent en tapant leur IP.

Par la suite, nous avons voulu mettre en place Pacemaker, Corosync et PCS ("*apt install corosync pacemaker pcs*")
Pour savoir si le daemon PCS est en exécution, il faut inscrire dans l'invite de commande : "*systemctl enable/start/status pcsd*"**.**
Cependant, nous nous sommes rendu compte que la manière dont nous faisions le projet n'était plus à jour avec Debian 12 et que les commandes avec **PCS ne fonctionnaient plus** donc nous avons choisi d'utiliser **Keepalived** qui a les mêmes principes que Pacemaker, Corosync et PCS mais regroupés en un.

Gestion par cluster :

Un cluster est un agrégat, une liaison de ressources comme des serveurs pour agir comme un seul et même système.
Pour commencer, nous avons installé Keepalived :

```
root@debian:~# apt install keepalived
```

Nous avons créé un dossier "*keepalived.conf*" dans le répertoire "*/etc/keepalived/*" avec la commande *nano* :

```
root@debian:~# nano /etc/keepalived/keepalived.conf
```

Pour la configuration du fichier, nous avons dû mettre l'adresse virtuelle (nous avons choisi le **10.0.0.233**) il ne faut pas oublier de mettre l'interface réseau adéquat sinon cela ne fonctionne pas. Et nous avons fait la même configuration pour les deux machines, sauf que l'interface réseau changeait

pour chacune.

```
  GNU nano 7.2
! Configuration File for keepalived
global_defs {
...
}
vrrp_script check_nginx {
    script "/etc/keepalived/check_nginx.sh"
    interval 2
    weight 50
}
vrrp_instance VI_1 {
    state BACKUP
    interface ens33
    virtual_router_id 50
    priority 100
    advert_int 1
    virtual_ipaddress {
        10.0.0.233
    }
    track_script {
        check_nginx
    }
}
```

```
  GNU nano 7.2
! Configuration File for keepalived
global_defs {
...
}
vrrp_script check_nginx {
    script "/etc/keepalived/check_nginx.sh"
    interval 2
    weight 50
}
vrrp_instance VI_1 {
    state MASTER
    interface eno1
    virtual_router_id 50
    priority 110
    advert_int 1
    virtual_ipaddress {
        10.0.0.233
    }
    track_script {
        check_nginx
    }
}
```

Puis, sur les deux machines nous avons créé un script en bash qui dit de lancer Nginx avant Keepalived dans le même répertoire que le fichier de configuration.

```
  GNU nano 7.2
#!/bin/sh
if [ -z "`/bin/pidof nginx`" ]; then
        systemctl stop keepalived.service
        exit 1
fi
# Change mode of the script
sudo chmod +x /etc/keepalived/check_nginx.sh
```

Ensuite nous avons redémarré Keepalived :

```
root@debian:/etc/keepalived# systemctl restart keepalived
```

Pour la suite, nous avons créé sur les deux machines un fichier *“index.nginx-debian.html”* (la page web par défaut qui s’affiche) dans le répertoire *“/var/www/html/”* mais avec une petite différence pour pouvoir voir la différence dans les tests donc dans la machine n°1 nous avons mis **web01** et pour la machine n°2, **web02**.

```
  GNU nano 7.2                                                  index.nginx-debian.html
$ curl http://10.0.0.233
<!DOCTYPE html>
<html>
<head>
<title>Bienvenue dans nginx !</title>
<style>
    body {
        width: 35em;
        marge : 0 automatique ;
        famille de polices : Tahoma, Verdana, Arial, sans-serif ;
    }
</style>
</head>
<body>
<h1>Bienvenue sur nginx web01 !</h1>
<p>Si vous voyez cette page, le serveur Web nginx est installé et
fonctionne avec succès. Une configuration supplémentaire est requise.</p>
<p>Pour obtenir de la documentation et une assistance en ligne, veuillez vous référer à
<a href="http://nginx.org/">nginx.org</a>.<br/>
Une assistance commerciale est disponible sur
<a href="http : //nginx.com/">nginx.com</a>.</p>
<p><em>Merci d'utiliser nginx.</em></p>
</body>
</html>
```

Démonstration du fonctionnement des serveurs :

Par défaut quand les deux serveurs sont allumés :

$ curl http://10.0.0.233

Bienvenue sur nginx web01 !

Si vous voyez cette page, le serveur Web nginx est installÃ© et fonctionne avec succÃ¨s. Une configuration supplÃ©mentaire est requise.

Pour obtenir de la documentation et une assistance en ligne, veuillez vous rÃ©fÃ©rer Ã nginx.org.
Une assistance commerciale est disponible sur nginx.com.

Merci d'utiliser nginx.

Quand le serveur physique/primaire est le seul fonctionnel :

$ curl http://10.0.0.233

Bienvenue sur nginx web01 !

Si vous voyez cette page, le serveur Web nginx est installÃ© et fonctionne avec succÃ¨s. Une configuration supplÃ©mentaire est requise.

Pour obtenir de la documentation et une assistance en ligne, veuillez vous rÃ©fÃ©rer Ã nginx.org.
Une assistance commerciale est disponible sur nginx.com.

Merci d'utiliser nginx.

Quand le serveur secondaire est le seul fonctionnel :

Réplication des données :

La réplication des données est le partage d'information pour s'assurer que l'un des serveurs ne soit pas surchargé de mises à jour ou d'autres données.
Pour cela, nous avons utilisé la commande rsync sur Linux.

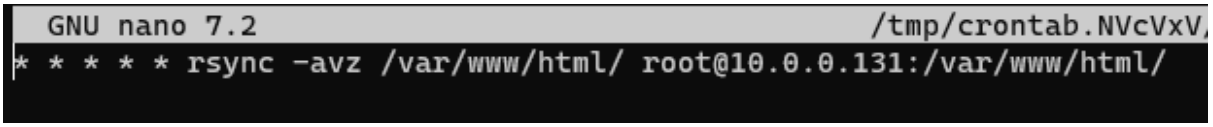
```
root@debian:~# apt install rsync
```

Serveur source : 10.0.0.133
Serveur de destination : 10.0.0.131
Nous avons voulu faire la réplication des données du site web du serveur source à la source de destination, donc nous faisons *"rsync -avz /var/www/html/ root@10.0.0.131:/var/www/html/"*
PS : Nous avions oublié de mettre la permission root dans les configurations ssh dans le fichier sshd_config PermiRootLogin no en yes.
Si le répertoire *"/var/www/html"* n'existe pas sur le serveur de destination, nous allons le créer :

```
root@debian:~# mkdir -p /var/www/html/
```

Pour automatiser la synchronisation des données à intervalles réguliers, nous allons créer une tâche cron sur le serveur source.

```
root@debian:~# crontab -e
```



Au niveau sécurité, nous allons générer une clé SSH du serveur source pour le serveur de destination pour permettre une connexion sans mot de passe. Il serait plus fiable d'utiliser une clé SSH qui est cryptée plutôt que de s'authentifier avec un mot de passe (que l'on peut oublier, qui peut être piraté…).

```
root@debian:~# ssh-keygen -t rsa -b 2048
```

Commande pour l'ajout de la clé publique à la liste des clés autorisés sur le serveur de destination :

```
root@debian:~# ssh-copy-id root@10.0.0.131
```

Ensuite on peut se connecter en SSH sur le serveur de destination sans mot de passe :

```
root@debian:~# ssh root@10.0.0.131
```

Nous pouvons tester la réplication de données en allant sur le fichier de la page web du serveur source et écrire "test". Cela se répercute sur le serveur de destination.

```
root@debian:~# nano /var/www/html/index.nginx-debian.html
```

Répartition des charges :

Nous allons sur le répertoire des sites web 1 et 2 :

```
root@debian:~# cd /etc/nginx/sites-available/
```

Afin de configurer le fichier par défaut.

```
root@debian:/etc/nginx/sites-available# nano default
```

Nous avons la configuration pour le serveur de base avec le port 81 pour l'accès et nous allons rajouter la configuration d'un serveur virtuel avec la localisation du proxy_pass en mettant le port 80 et la fonction "upstream_backend" présente les serveurs qui feront la répartition des charges.

```
 GNU nano 7.2
server {
        listen 81 default_server;
        listen [::]:81 default_server;

        root /var/www/html;

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

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }

}

server {
  listen 80 default_server;
  listen [::]:80 default_server;
  root /var/www/html;
  server_name _;

  location / {
    proxy_pass http://backend;
  }
}


upstream backend {
   server 10.0.0.133:81;
   server 10.0.0.131;
}
```

Partie juridique :

Quelles sanctions encourent un prestataire s'il ne respecte pas la haute disponibilité du site de son client ?

La haute disponibilité d'un service est importante pour la résilience et la fiabilité des systèmes. Elle est utile en cas de piratage, d'erreur humaine ou de pannes matérielles. Cette mise en œuvre se fait par un prestataire de service pour un contrat établi avec un client (comme une entreprise par exemple). Plus précisément, cela rentre dans le **Service Level Agreement (soit SLA)** qui est une partie de contrat de service entre un prestataire informatique et son client qui définit le niveau de service attendu et les garanties associées.

Si le prestataire ne respecte pas le seuil de performance, il peut encourir à des sanctions pécuniaires, soit financières. Cela mène aussi d'une obligation de moyens à une obligation de résultat, c'est-à-dire qu'il doit mettre tous les moyens possibles pour atteindre un objectif sans forcément le réussir ou finir mais si l'obligation de résultat est demandée, il devra nécessairement aboutir à un bon résultat.

Problèmes rencontrés :

Nous n'avons pas pu faire un serveur secondaire physique sur Linux car l'autre poste n'arrivait pas à booter sur Linux, il se dirigeait directement sur Windows et avait un problème avec l'écran donc nous avions remplacé le PC pour un autre mais second problème, nous n'avions pas le mot de passe du root lorsqu'on voulait installer les dépendances alors nous avons opté pour une VM Linux sur VMWare.

Liens utiles :

Haute disponibilité sur Linux :
https://fr.linux-console.net/?p=120#gsc.tab=0
https://fr.linux-console.net/?p=4724#gsc.tab=0
ces deux-là sont obsolètes ^
nous avons continué avec celui-ci avec Keepalived v
https://j3ffyang.medium.com/nginx-high-availability-and-load-balancing-with-keepalived-521d44798bff
Définition de Pacemaker :
https://doc.ubuntu-fr.org/pacemaker
Définition de Nginx :
https://doc.ubuntu-fr.org/nginx
Contrat en service informatique :
https://www.wifirst.com/blog/reseau-haute-disponibilite-pour-les-entreprises#:~:text=La%20disponibilit%C3%A9%2C%20facteur%20cl%C3%A9%20du%20SLA&text=Une%20indisponibilit%C3%A9%20au%20niveau%20du,interrompre%20sa%20cha%C3%AEne%20de%20production.

https://www.wifirst.com/blog/service-level-agreement-informatique

https://www.redhat.com/fr/topics/linux/what-is-high-availability#:~:text=La%20haute%20disponibilit%C3%A9%20va%20plus,et%20la%20fiabilit%C3%A9%20des%20syst%C3%A8mes.

https://www.group-dis.com/blog/infogerance/sla-le-contrat-de-maintenance-de-votre-informatique

https://www.prd-avocats.com/wp-content/uploads/2014/04/La-clause-de-SLA.pdf

Réplication des données :
https://www.hostinger.fr/tutoriels/commande-rsync-linux

Répartition des charges :
https://upcloud.com/resources/tutorials/configure-load-balancing-nginx

https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-setup-an-Nginx-load-balancer-example

Bloc-note :
Réseau de la S4 : 10.0.0.0/19 (255.255.224.0)
Serveur DNS : 10.10.10.10
Plage d'adresse disponible : 10.0.0.100 à .200
Pauline : .131 Nicolas : .133