PROJET PROFESSIONNELLE N°1

CONTEXTE :

Les Médecins ruraux se plaignent de la communication parfois inexistante entre eux et les hôpitaux quant aux patients suivis pour une greffe de rein. C'est pour cela que le CHU de Bordeaux a contacté l'entreprise GOAT-IT afin de mettre en place un logiciel facilitant la communication entre les deux parties.
Ils nous ont demandé de mettre en place le développement total du logiciel ainsi que son fonctionnement et hébergement.
Étant Administrateur réseaux, je suis chargé de mettre en place une infrastructure permettant la haute disponibilité et la sécurité du logiciel.

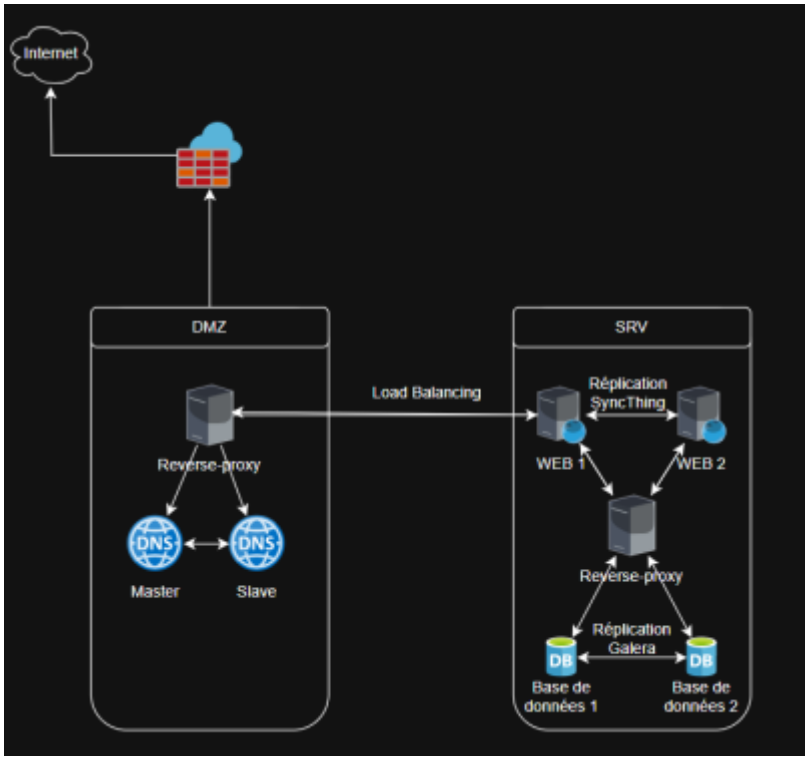
Sommaire des modifications apportées à l'infrastructure :

REFLEXION :

1.1 Analyse des besoins

Pour ce faire je compte mettre en place un reverse-proxy dans une DMZ qui aura pour but de protéger et faire l'équilibrage de charges entre les deux serveurs WEB qui hébergeront le logiciel et qui eux seront sur un VLAN nommé SRV dans le réseau interne et non accessible depuis internet.
Pour renforcer le bon fonctionnement du logiciel je vais mettre en place 2 Serveurs de Base données répliqué a l'aide de l'outil galera qui les répliquera en temps réels. Ces Bases de données seront elle mêmes relié a un autre reverse-proxy qui fera la liaison avec les deux serveur Web et qui permettra ainsi l'équilibrage de charges vers les bases de données ce qui renforcera d'autant plus la pérennité du logiciel.
En cas de problème un serveur de backup fera les backupe de tout les serveurs ainsi que reverse-proxy grâce a l'outils Backuppc.
Les machines seront aussi superviser a l'aide de l'outils zabbix. La connexion a internet et le nom de domaine se feront grâce a nos DNS.

1.2 Choix de l’architecture :



VLAN	MACHINE	ADRESSE IP	MASQUE	PORT
DMZ	DNS MASTER	10.31.248.53	255.255.252.0/22	53
DMZ	DNS SLAVE	10.31.248.54	255.255.252.0/22	53
DMZ	REVERSE-PROXY	10.31.248.10	255.255.252.0/22	22000
SRV	WEB 1	10.31.252.81	255.255.254.0/23	80 443
SRV	WEB 2	10.31.252.82	255.255.254.0/23	80 443
SRV	REVERSE-PROXY	10.31.252.5	255.255.254.0/23	22000
SRV	BDD 1	10.31.252.10	255.255.254.0/23	3306
SRV	BDD 2	10.31.252.11	255.255.254.0/23	3306

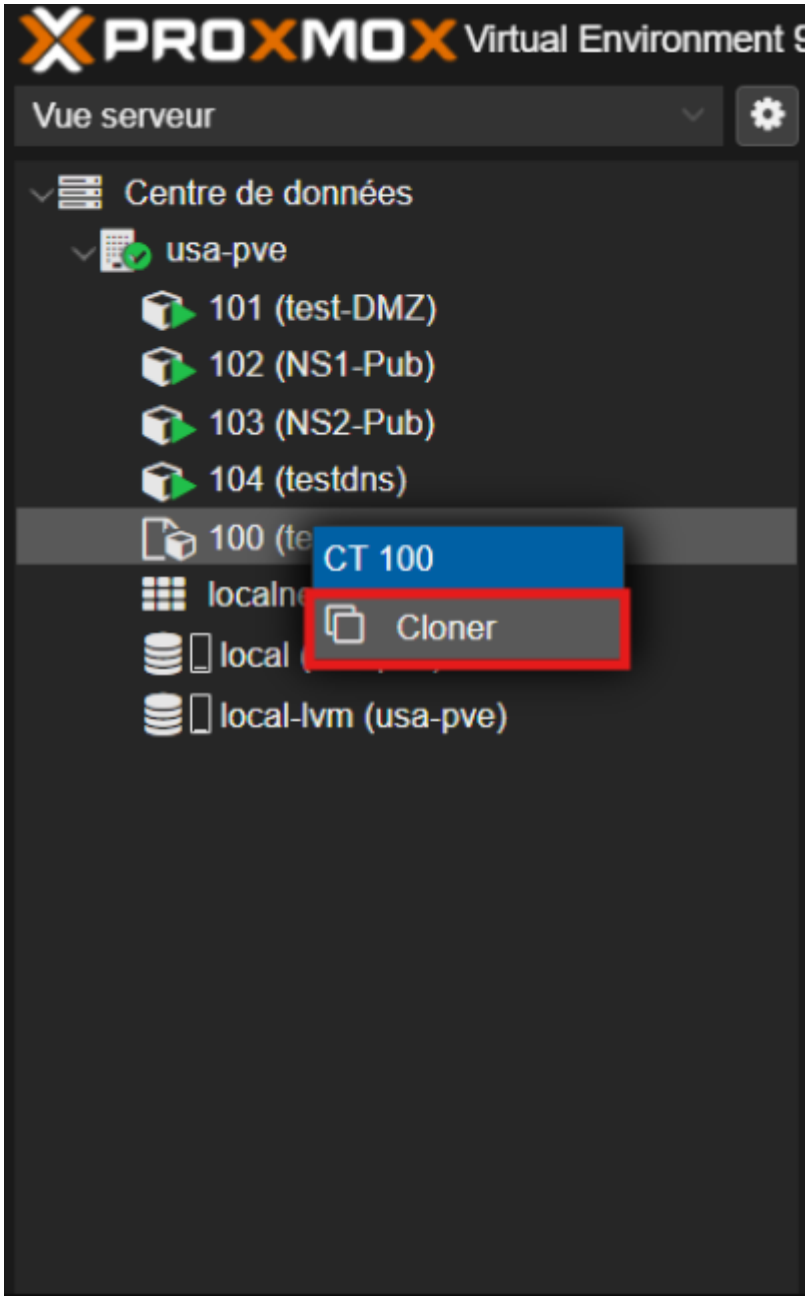
2. Création du reverse-proxy dans la DMZ

Pour commencer nous allons créer un reverse-proxy pour assurer la haute disponibilité de nos serveurs WEB dans le LAN depuis la DMZ

> **Avantage du reverse-proxy :**
>
> * Permet d'assurer la haute disponibilité de nos serveur WEB.
> * Permet de résister aux attaques DDOS.
> * Filtrer les entré et les sorties des requêtes .

2.1 Installation du reverse-proxy

Nous allons commencer par faire un clone de notre template pour ensuite le trensformer en un reverse-proxy



Ensuite nous allons configurer ses paramètre réseau, étant dans la DMZ son adresse sera en 248 ce qui va nous donner 10.31.248.10

une fois fait il ne reste plus qu'a installer HA proxy qui est le logiciel qui va nous servir de reverse-proxy grâce a cette ligne de commande :

```
curl https://haproxy.debian.net/haproxy-archive-keyring.gpg --create-dirs --output /etc/apt/keyrings/haproxy-archive-keyring.gpg
```

Ensuite nous allons ajouter le dépôt HAProxy
La commande suivante permet d'ajouter un dépôt APT dédié à **HAProxy** dans le système :

```
echo 'deb [signed-by=/etc/apt/keyrings/haproxy-archive-keyring.gpg] https://haproxy.debian.net trixie-backports-3.2 main' > /etc/apt/sources.list.d/haproxy.list
```

Cette ligne crée le fichier `/etc/apt/sources.list.d/haproxy.list` et y ajoute un dépôt externe.
Ce dépôt fournit les paquets de **HAProxy** pour **Debian** (version *Trixie*) via le serveur `https://haproxy.debian.net`.
Le paramètre `signed-by=/etc/apt/keyrings/haproxy-archive-keyring.gpg` indique à APT quelle clé GPG utiliser pour vérifier l'authenticité des paquets téléchargés depuis ce dépôt.\ L'objectif est de permettre au gestionnaire de paquets `apt` de récupérer et installer la version d'HAProxy fournie par ce dépôt, généralement plus récente que celle disponible dans les dépôts standards de Debian.

Après ca nous allons donc l'installer

```
apt-get install haproxy=3.0.\*
```

Et c'est sur ca que la partie installation ce termine.

2.2 Configuration réseau et sécurité DMZ

Pour que notre reverse proxy puisse communiquer avec notre futur infrastructure il va falloir autoriser la communication de la DMZ avec le VLAN SRV. Pour ce faire on va devoir faire des règles de pare-feu sur notre routeur OPNSense

Les règles sont :

Pour le DMZ

Type	Protocole	Source IP	Port Src	Destination	Port Dest
IPv4 UDP	UDP	10.31.248.53/32	*	10.31.252.10/32	53 (DNS)
IPv4 UDP	UDP	10.31.248.53/32	*	10.31.252.11/32	53 (DNS)
IPv4 UDP	UDP	10.31.248.53/32	*	8.8.8.8/32	53 (DNS)
IPv4 UDP	UDP	10.31.248.53/32	*	8.8.4.4/32	53 (DNS)
IPv4 UDP	UDP	10.31.248.54/32	*	8.8.8.8/32	53 (DNS)
IPv4 UDP	UDP	10.31.248.54/32	*	8.8.4.4/32	53 (DNS)
IPv4 UDP	UDP	10.31.248.53/32	*	Vlansrv	53 (DNS)
IPv4 UDP	UDP	10.31.248.54/32	*	Vlansrv	53 (DNS)
IPv4 TCP	TCP	*	*	*	443 (HTTPS)
IPv4 TCP	TCP	*	*	*	80 (HTTP)

Pour le Vlan SRV :

Type	Protocole	Source IP	Port Src	Destination	Port Dest
IPv4 TCP	TCP	10.31.252.81/32	*	10.31.248.10/32	80 (HTTP)
IPv4 TCP	TCP	10.31.252.82/32	*	10.31.248.10/32	80 (HTTP)
IPv4 TCP	TCP	10.31.252.81/32	*	10.31.248.10/32	443 (HTTPS)
IPv4 TCP	TCP	10.31.252.82/32	*	10.31.248.10/32	443 (HTTPS)

Le premier tableau regroupe principalement des règles de type **IPv4 UDP** destinées au trafic DNS (port 53). On y voit que les machines `10.31.248.53` et `10.31.248.54` sont autorisées à interroger plusieurs serveurs DNS, à la fois internes (`10.31.252.10`, `10.31.252.11`, et `Vlansrv`) et externes (les serveurs publics `8.8.8.8` et `8.8.4.4`). Cela permet d'assurer la résolution de noms de domaine même en cas de défaillance d'un serveur interne. En complément, deux règles en **IPv4 TCP** autorisent le trafic sortant vers les ports 443 (HTTPS) et 80 (HTTP), ce qui permet un accès web classique pour tous les hôtes.

Le second tableau présente des règles plus ciblées en **IPv4 TCP** sur le port 80 (HTTP). Ici, seules les adresses `10.31.252.81` et `10.31.252.82` sont autorisées à communiquer avec le serveur `10.31.248.10`. Cela correspond typiquement à un contrôle d'accès vers un serveur web interne, en limitant les connexions aux machines autorisées uniquement. Cette configuration renforce la sécurité en restreignant l'accès au service HTTP à un périmètre précis plutôt qu'à l'ensemble du réseau.

3. Création d'un serveur Web dans le VLAN SRV

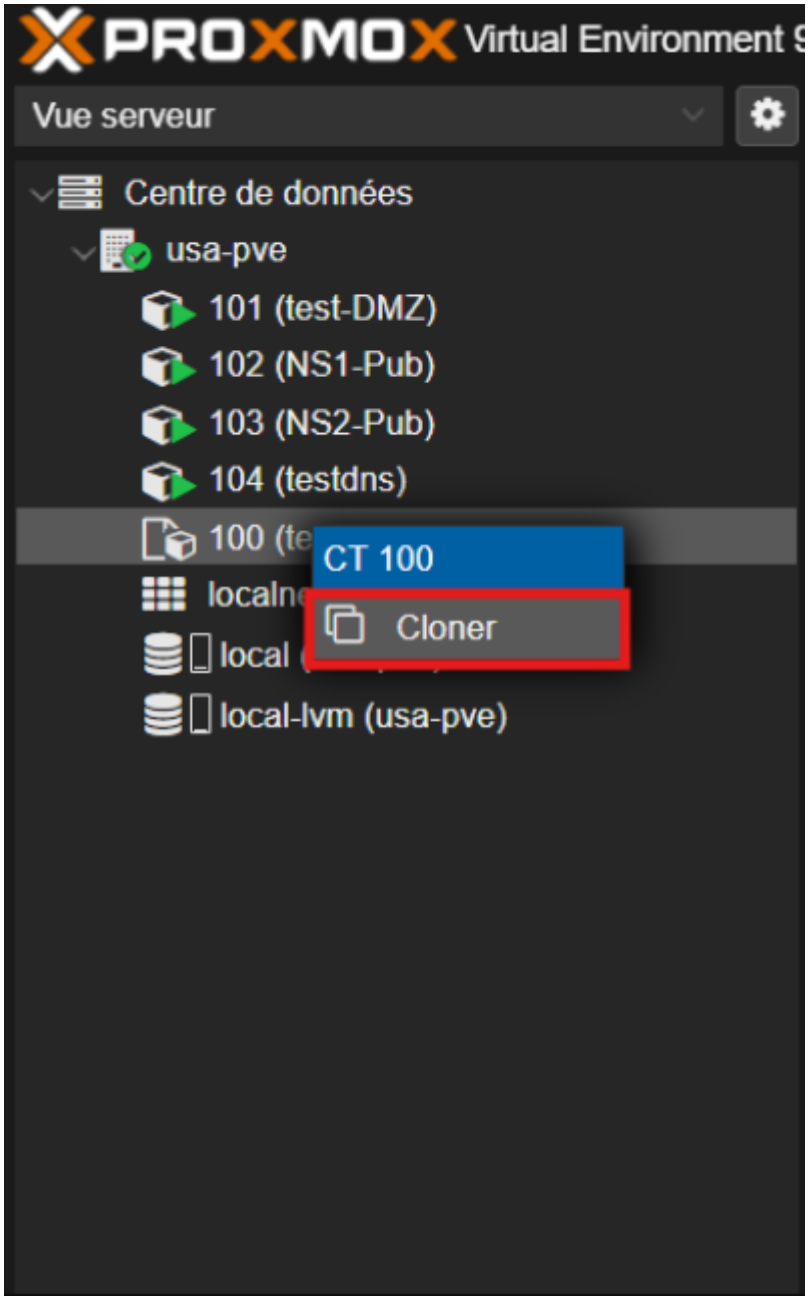
Pourquoi mettre les serveur dans le VLAN SRV ?

Le Vlan SRV est un vlan dédié au serveur au sein de notre système d'information. Le reverse proxy étant dans la DMZ il va nous permettre de rajouter une couche de sécurité en plus car notre site qui lui sera héberger sur le serveur WEB sera contractable que depuis la DMZ depuis le reverse proxy ce qui rajoute un couche de sécurité en plus car si un pirate réussi a avoir accès a notre reverse proxy il sera bloquer dans la DMZ et non pas dans notre system d'information ou ce trouve tout les autres serveur.

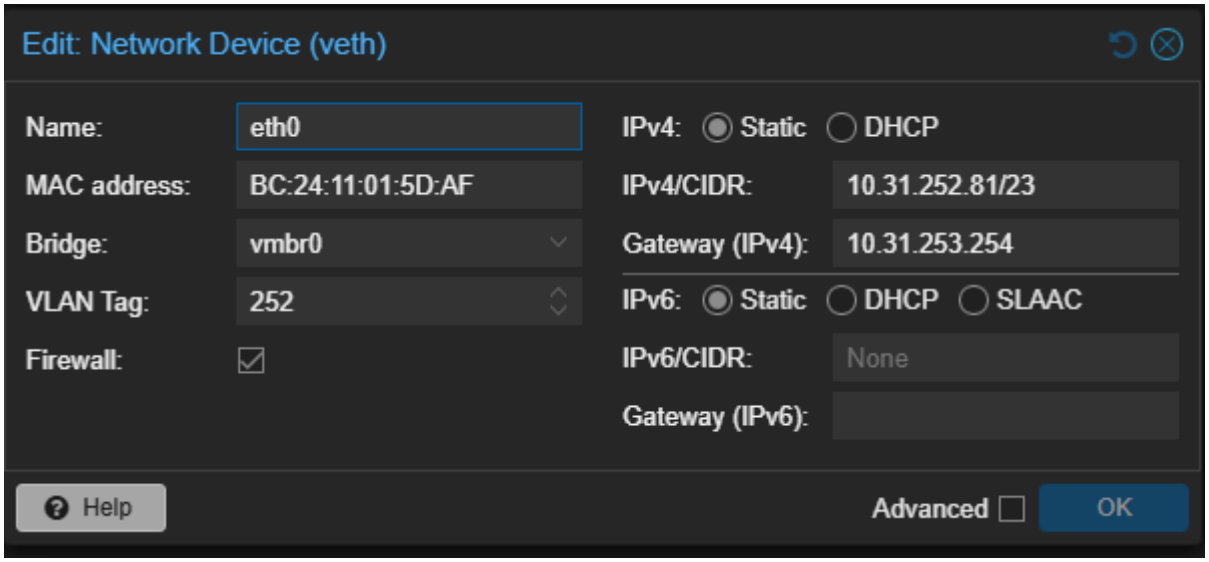
En conclusion seul le reverse proxy ce trouve en danger depuis internet.

3.1 Installation du serveur Web

Pour la création de notre serveur web nous allons cloner notre Template :



Une fois ca fait nous allons attaquer la configuration réseau de notre machine :



Une fois la machine configurer on va pouvoir ce connecter en SSH

```
ssh root@10.31.252.81
```

une fois connecter on va pouvoir commencer l'installation d'apache 2

> **Avantage d'Apache2 :**
> * Gratuit et open source
> * Compatible avec plusieurs systèmes
> * Supporte de nombreux langages (PHP, Python, Perl...)
> * Très flexible grâce aux modules
> * Bonne sécurité (HTTPS, gestion des accès) * Stable et fiable (utilisé depuis longtemps) * Fichier .htaccess pratique pour configurer facilement * Grande communauté et beaucoup de documentation * Facile à prendre en main * Polyvalent pour différents types de projets web

Pour commencer nous allons l'installer

```
sudo apt update
sudo apt install -y apache2 php php-cli php-xml php-mbstring php-intl php-curl php-zip unzip curl
```

Une fois apache2 installer nous allons installer un composer pour Symfony car les développeur travail a l'aide de cette solution

```
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
composer --version
```

Après avoir fait ca je créer le dossier symfony dans var www

```
mkdir /var/www/symfony
```

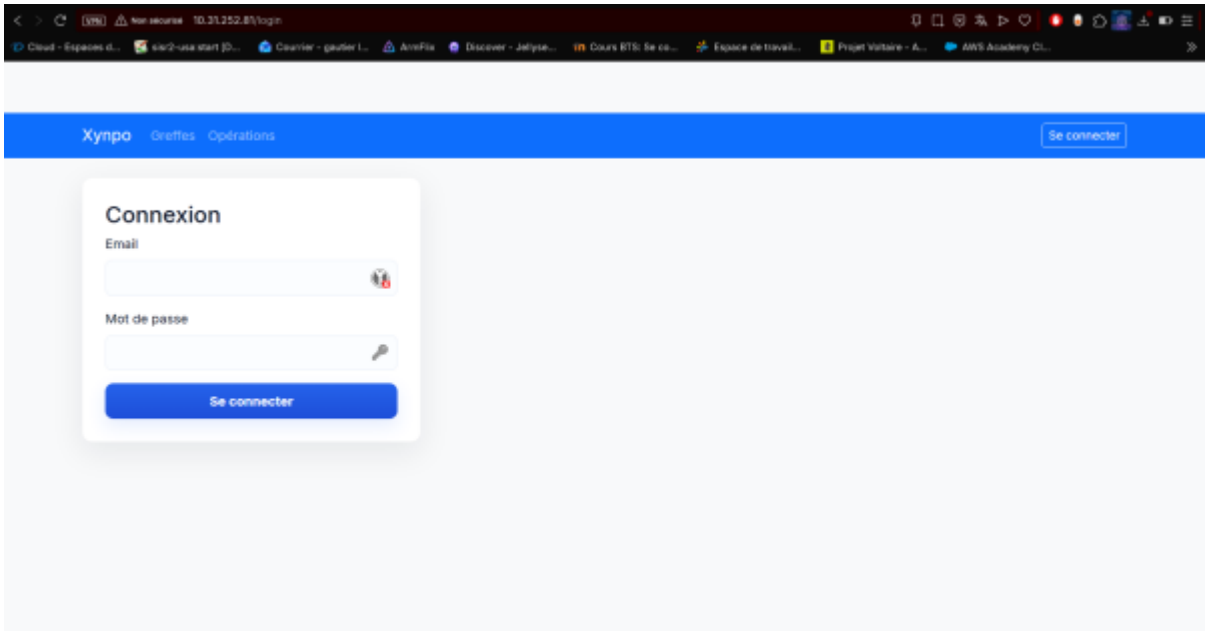
C'est d'ici que le développeur va faire toutes son infrastructure

```
root@Web-N:/var/www/symfony# ls
Dockerfile  bolt.db              composer.lock  migrations         public        translations
LICENSE     compose.override.yaml  config       node_modules       src           var
README.md   compose.prod.yaml      docs         package-lock.json  symfony.lock  vendor
assets      compose.yaml           frankenphp   package.json       templates     webpack.config.js
bin         composer.json          importmap.php  phpunit.dist.xml  tests
```

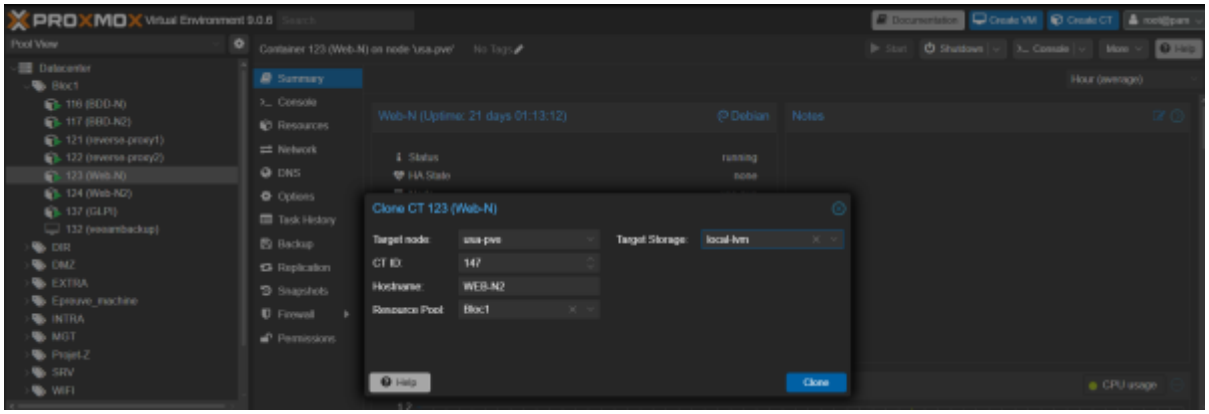
Evidemment pour que le site soit bien héberger il nous faut un Virtual host :

```
</VirtualHost>
root@Web-N:/var/www/symfony# cat /etc/apache2/sites-available/symfony.conf
<VirtualHost *:80>
    ServerName symfony.local
    ServerAlias 10.31.248.10

    DocumentRoot /var/www/symfony/public

    <Directory /var/www/symfony/public>
        AllowOverride All
        Require all granted
        ProxyPreserveHost On
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/symfony_error.log
    CustomLog ${APACHE_LOG_DIR}/symfony_access.log combined
</VirtualHost>
```

Une fois ca fait on va pouvoir tester notre configuration

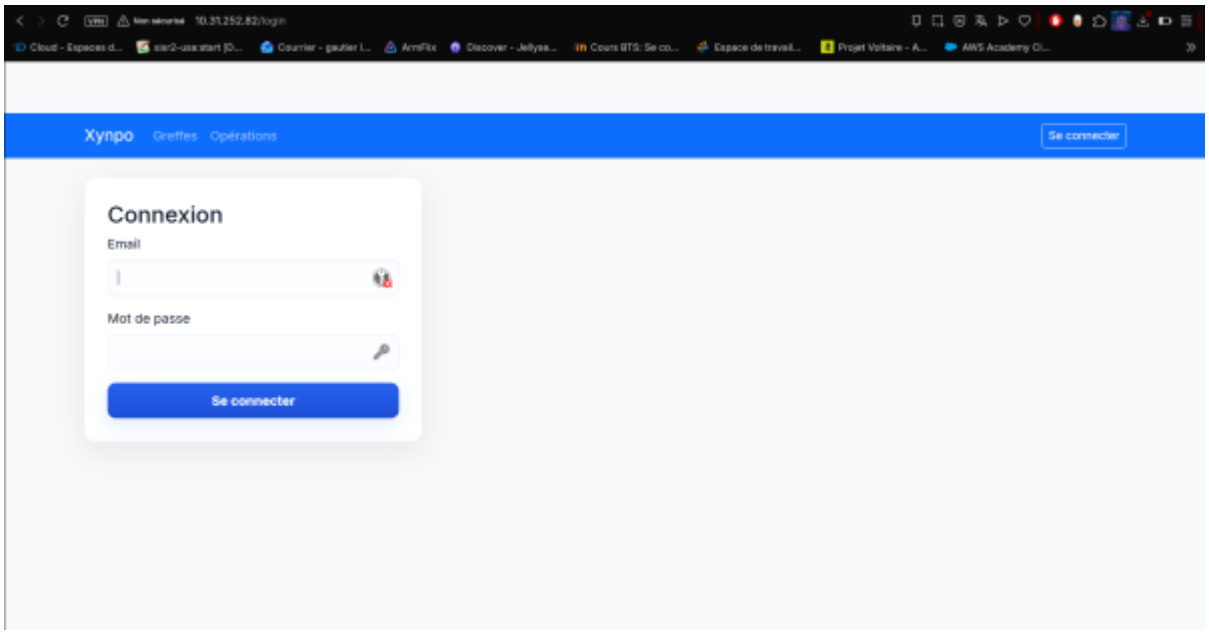


on peut voir que notre site est bien héberger sur notre serveur web
Une fois le premier crée il nous reste plus qu'a le cloner dans proxmox pour avoir les deux serveur web similaire



on change ses paramètres réseau de sorte a ce que les deux machines n'ais pas les mêmes adresses

Une fois le clone effectuer on peut aussi tester de contacter notre site depuis l'adresse IP de notre serveur



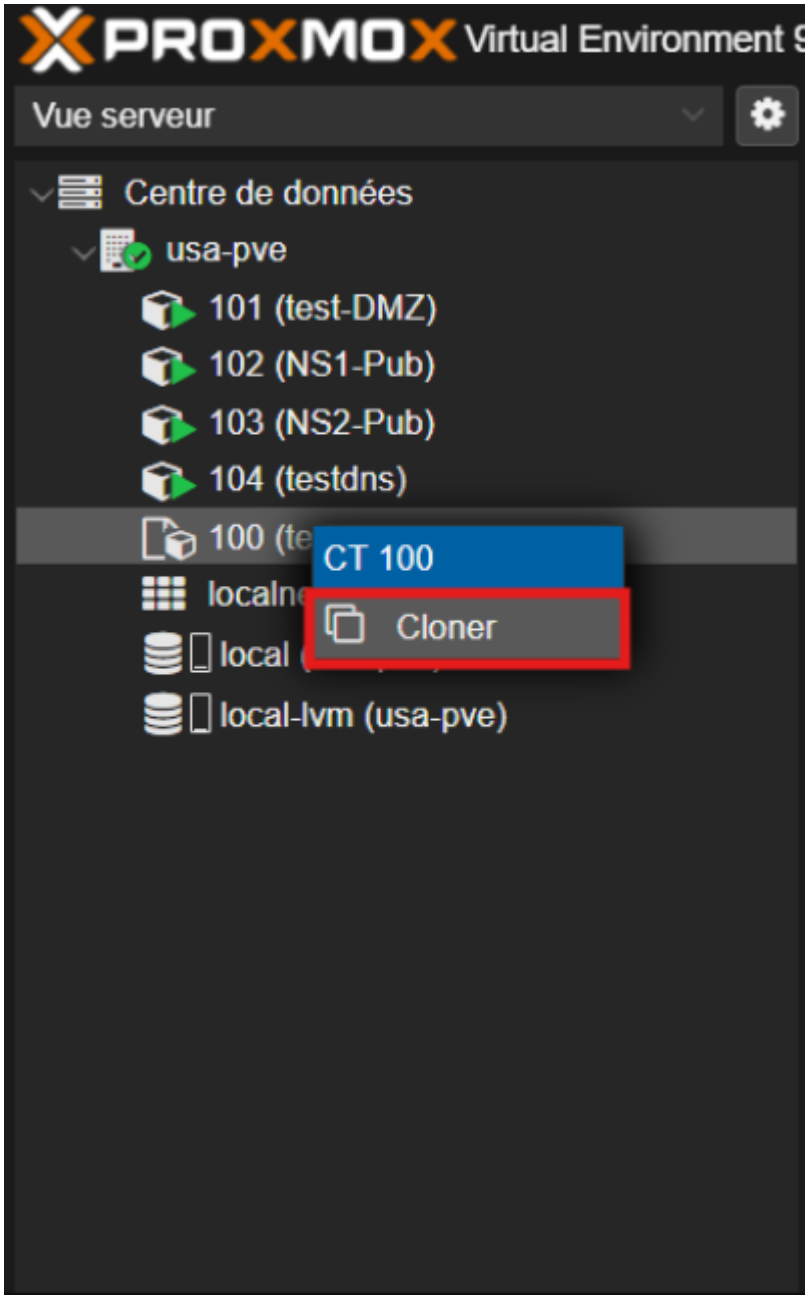
4. Mise en place du load balancing entre les serveurs WEB

4.1 Principe de fonctionnement

Pour assurer la haute disponibilité de nos serveur web il va falloir mettre en place un load balancing entre le deux serveurs de sorte a premièrement soulager le nombre de requetés par 2 sur chaque serveurs web et ca va permettre qu'un des serveurs web prenne le relais si un des serveurs tombe. Haproxy qui est la solution que j'ai choisi pour mettre en place ce système de load balancing vas regarder grâce a des requêtes ICMP, si il se rend compte qu'un des serveurs n'est pas contractable il va automatiquement redirigé toutes les connexions sur le serveurs web encore fonctionnel

4.2 Configuration du load balancing

On va commencer par créer le reverse-proxy :



Ensuite on va configurer son réseau de sorte a ce qu'il soit dans la DMZ :

Une fois sur notre serveur on va commencer l'installation de HAproxy

```
sudo apt install curl -y
curl https://haproxy.debian.net/haproxy-archive-keyring.gpg --create-dirs --
output /etc/apt/keyrings/haproxy-archive-keyring.gpg
echo deb "[signed-by=/etc/apt/keyrings/haproxy-archive-keyring.gpg]"
https://haproxy.debian.net trixie-backports-3.2 main
/etc/apt/sources.list.d/haproxy.list
apt-get install haproxy=3.0.\*
```

Une fois HAproxy installer on va pouvoir le configurer

```
nano /etc/haproxy/haproxy.cfg
```

Fichier de configuration haproxy :

```
global
    log /dev/log local0
    log /dev/log local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

    maxconn 4000    # Capacité max

defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    option  forwardfor       # Ajoute X-Forwarded-For
    option  http-server-close
    timeout http-request 10s
```

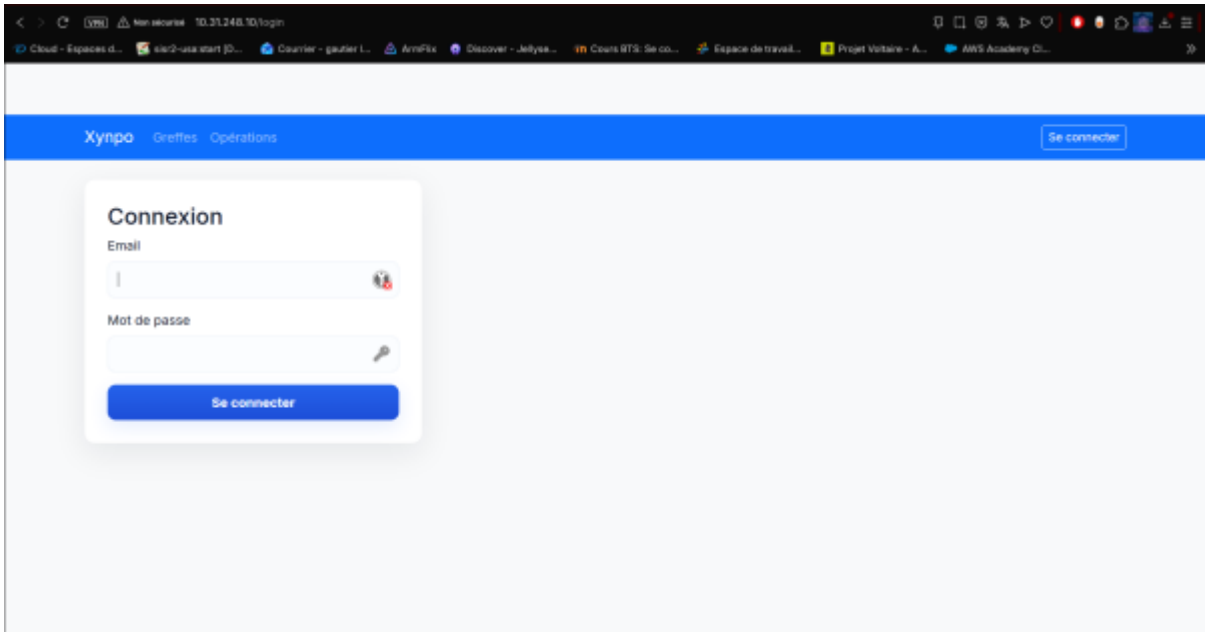
```
    timeout queue        1m
    timeout connect      10s
    timeout client       1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check        10s
    maxconn 3000

# ============================
#     FRONTEND (Reverse Proxy)
# ============================
frontend http_frontend
    bind 10.31.248.10:80
    mode http

    # Redirection vers le backend
    default_backend web_backend

# ====================================
#     BACKEND (Load Balancing Web)
# ====================================
backend web_backend
    mode http
    balance roundrobin

    # Vérifie que les serveurs répondent
    option httpchk GET /

    # Serveurs web derrière HAProxy
    server web1 10.31.252.81:80 check
    server web2 10.31.252.82:80 check

# ====================================
#      INTERFACE DE STATISTIQUES
# ====================================
listen stats
    bind 10.31.248.10:9000
    mode http
    stats enable
    stats uri /stats
    stats refresh 10s
    stats realm HAProxy\ Stats
    stats auth admin:admin
```

Une fois la configuration faites on redémarre HAproxy

```
systemctl restart haproxy
```

Une fois avoir fait ca on test en tapant l'adresse IP du reverse proxy dans un navigateur si jamais on tombe sur le site alors c'est qu'il marche
Par contre pour savoir si le load balancing marche on doit modifier quelque chose sur le site du

serveur 1 par exemple et si quand on refresh la page plusieurs fois un des deux sites est diffèrent



On peut voir que j'ai bien rentré l'adresse de mon reverse proxy et ca me ressors ma page de login de mon site donc la communication fonctionne bien

5. Création et mise en place de la réplication entre les deux serveurs Web

5.1 But d'une réplication entre les deux serveurs web

La réplication entre mes deux serveurs web va permettre que les développeurs travaille uniquement sur un des serveurs web de sort a ce que quand ils apportent une modification elles soient effective sur tout les serveurs en même temps.
Dans mon cas ca n'est pas obligatoire car je n'ai que deux serveurs mais dans une entreprise qui a 40 serveurs WEB qui héberge la même chose ca en devient obligatoire

5.2 Configuration de la réplication

Pour la réplication des serveurs je me suis tourné sur un script rsync avec une table cron de sorte a ce que je puisse décider du délai qu'il y a entre le déploiement de nouveauté et mise en production de plus ca va me permettre de décider de quand est-ce que les serveurs se réplique.
Ce script va prendre l'entièreté du dossier :

```
/var/www/symfony
```

Car c'est dans se dossier que toutes les modification vont avoir lieu, c'est aussi dans ce répertoire que se trouve l'entièreté du site. C'est la raison pour la quelle on réplique que ce répertoire

```
#!/bin/bash
# Script de réplication Symfony vers le serveur secondaire

rsync -avz --delete \
```

```
    --exclude 'var/cache/' \
    --exclude 'var/log/' \
    --exclude 'vendor/' \
    --exclude 'node_modules/' \
    /var/www/symfony/ root@10.31.252.82:/var/www/symfony/
```

Pour accompagner notre script j'ai mit en place une crontab de sorte a ce que le script soit exécuté toutes les minutes

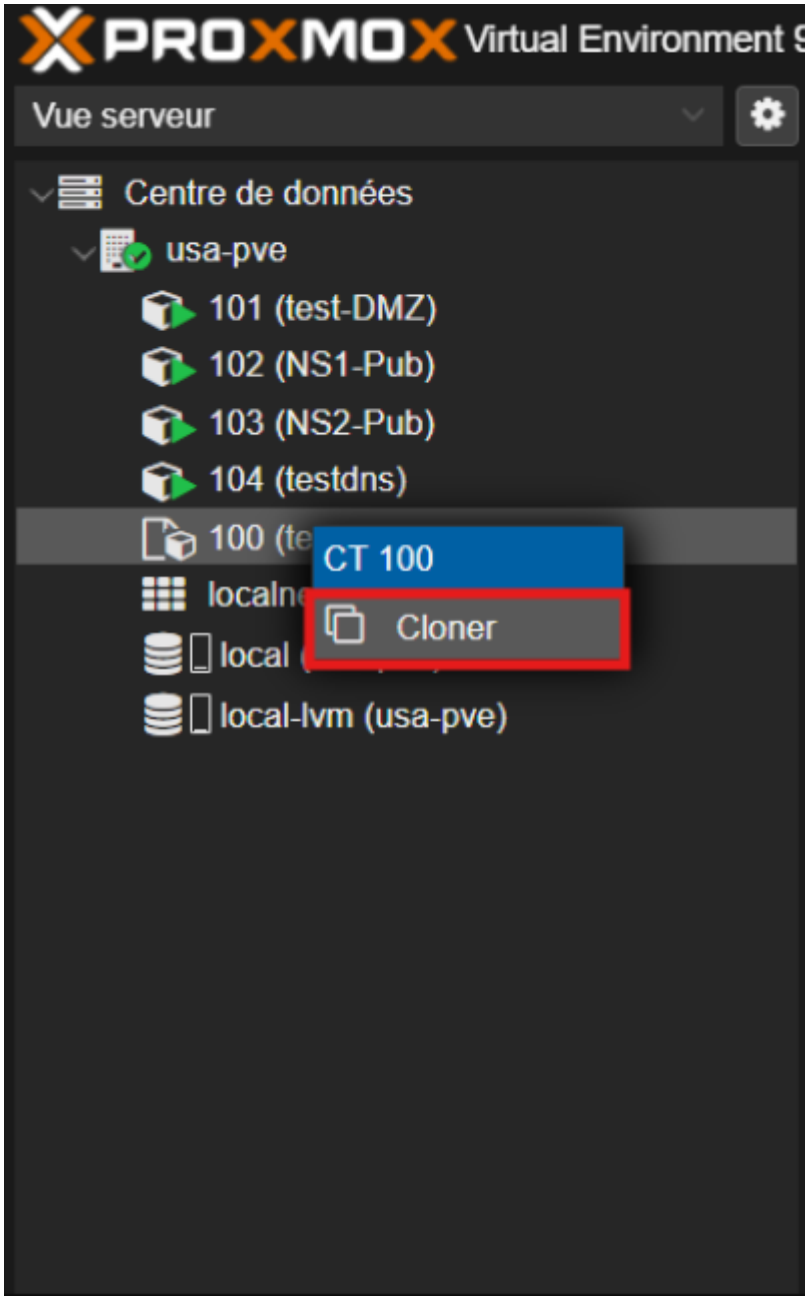
```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:

* * * * * /root/rsync_symfony.sh
* * * * * rsync -az --delete --exclude 'var/cache/' /var/www/html/symfony/
root@10.31.252.82:/var/www/html/symfony/

* * * * * rsync -az --delete /etc/apache2/sites-available/
root@10.31.252.82:/etc/apache2/sites-available/

* * * * * rsync -az --delete /etc/apache2/sites-enabled/
root@10.31.252.82:/etc/apache2/sites-enabled/
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

Si après modification du serveurs web 1, les deux site sont les mêmes alors la réplication fonctionne

6. Création d'une base de données

6.1 Création d'un serveur de base de données

Pour créer la base de données on va faire un nouveau conteneur clone depuis notre Template



Ensuite on va configurer ses paramètre réseau de sorte qu'elle soit dans le même réseau que mes serveurs web

Une fois connecté dessus on va installer notre mariadb

```
 apt update
apt install mariadb-server -y
systemctl start mariadb
systemctl enable mariadb
```

On va ensuite créer l'utilisateur de la base

```
CREATE USER 'nolan'@'%' IDENTIFIED BY 'mot_de_passe_solide';
GRANT ALL PRIVILEGES ON *.* TO 'nolan'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

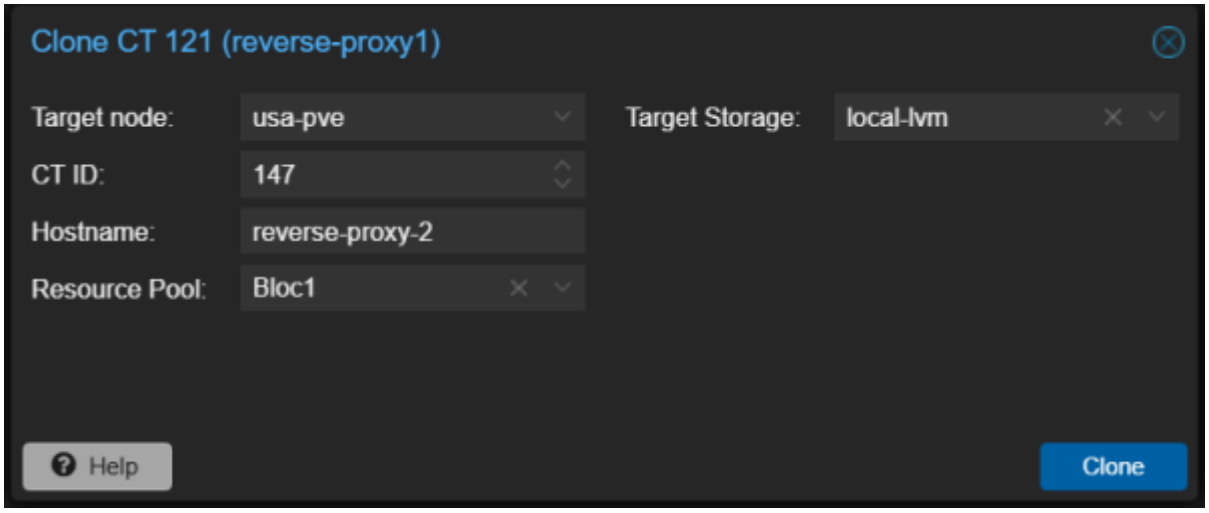
une fois avoir créé son utilisateur on va permettre la connexion a distance car on a pas fait un compte en localhost

```
[mysqld]

bind-address=0.0.0.0

default_storage_engine=InnoDB

innodb_autoinc_lock_mode=2

binlog_format=row
```

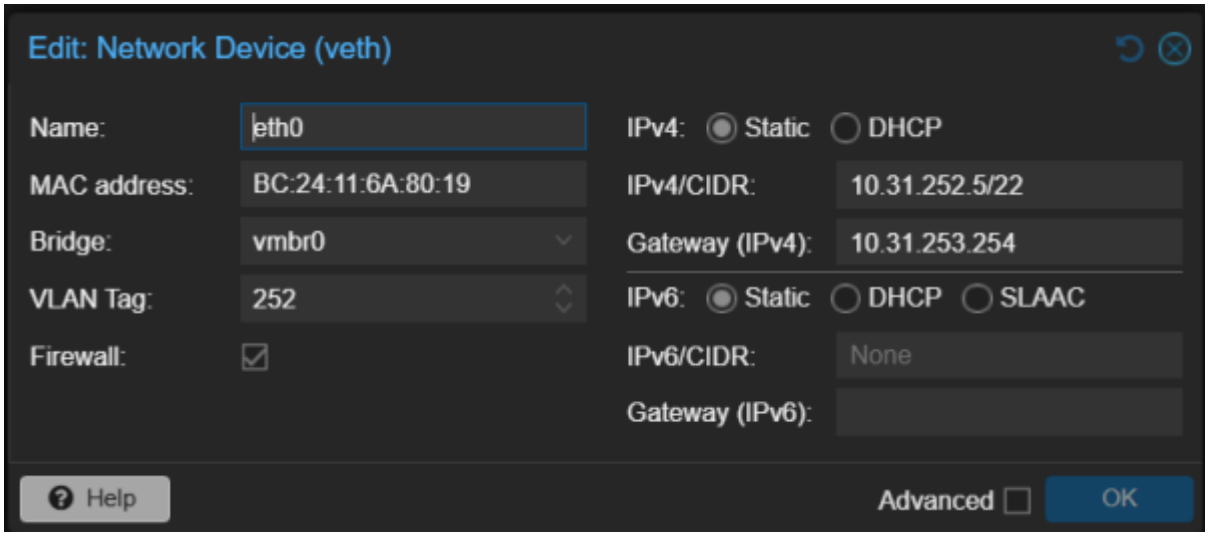
7. Mise en place du load balancing entre la base de données et le nouveau reverse proxy

7.1 Mise en place et Configuration du load balancing base de données

Pour ce faire il va falloir créer un nouveau reverse proxy qui lui sera dans le même vlan que nos serveurs web ainsi que base de données, ce reverse proxy va nous permettre de soulager la charge qu'encaisse les deux base de données par 2 ainsi que ca haute disponibilité puisque si une base de données tombe l'autre prendra le dessus.
Pour créer ce reverse proxy on va cloner celui qu'on a déjà crée auparavant ce qui va nous faire gagner du temps puisque tout est deja installé on aura plus qu'a modifié les configuration :



Puis nous allons changer sa configuration réseau de sorte a ce qu'elle soit placé dans le bon vlan



Une fois paramétré on peut se connecter dessus, ayant déjà tout d'installer du au clone nous devons juste changer la configuration

```
nano /etc/haproxy/haproxy.cfg
```

```
# ==============================
#  FRONTEND SQL (Reverse Proxy)
# ==============================
frontend mysql_frontend
    bind 10.31.252.5:3306
    mode tcp
    option tcplog
    default_backend mysql_backend

# ==============================
#  BACKEND SQL (Load Balancing)
# ==============================
backend mysql_backend
    mode tcp
    balance roundrobin
```

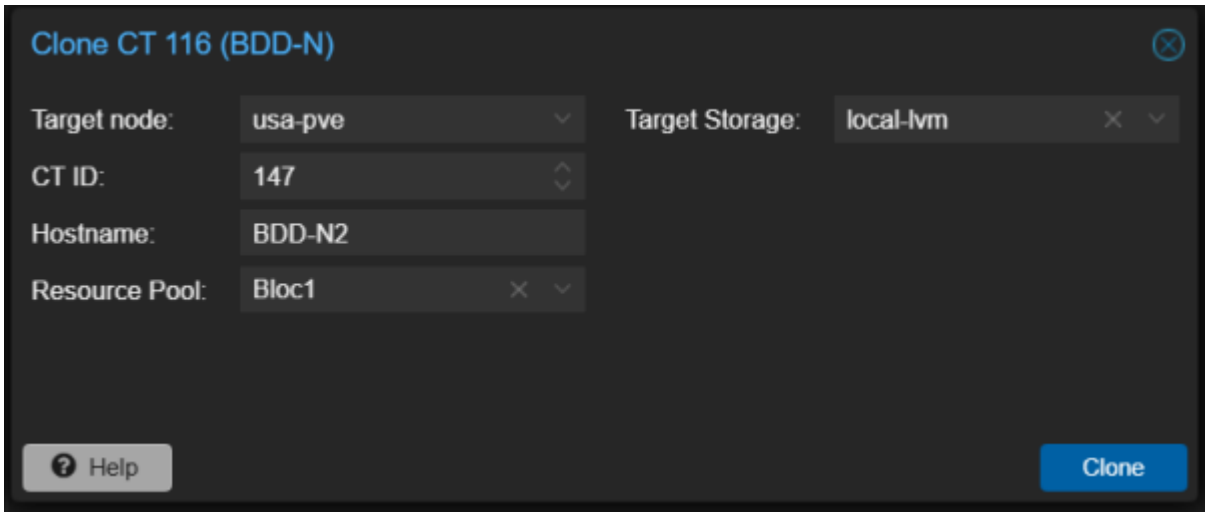
```
    # Health check MySQL
    option mysql-check user haproxycheck

    # Serveurs SQL répliqués
    server db1 10.31.252.10:3306 check
    server db2 10.31.252.11:3306 check
```

Une fois cela fait votre load balancing est fonctionnelle

8. Création d'une seconde base de données répliquée sur la première à l'aide de Galera

8.1 Création de la seconde base de données

Pour effectuer une réplication il faut forcement une base de données pour ce faire on va se faciliter la tache en clonant celle déjà existante ce qui va nous permettre d'avoir tout de déjà installer avec les utilisateurs etc..



Puis ensuite on va modifier son adressage IP

Une fois ca fait notre seconde base de données est crée et prête a l'emploie

8.2 Mise en place de la réplication

Pour la réplication j'ai choisie la solution galera car elle est faite nativement pour mariadb donc c'était une évidence
La configuration s'est faite premièrement sur la BDD 1
On va commencer par installer galera sur notre serveurs de base de données

```
sudo apt update && sudo apt install mariadb-server galera-4 -y
```

En suite sur les deux serveur il y a ce fichier a modifier

```
nano /etc/mysql/mariadb.conf.d/60-galera.cnf
```

BDD 1

```
[mysqld]
# ---- Options essentielles ----
server-id=1
binlog_format=ROW
default_storage_engine=InnoDB
innodb_flush_log_at_trx_commit=2
innodb_autoinc_lock_mode=2

# ---- Activation de Galera ----
#wsrep_on=ON
#wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Nom du cluster
wsrep_cluster_name="mariadb_cluster"

# Liste des noeuds du cluster
#wsrep_cluster_address="gcomm://10.31.252.10,10.31.252.11"

# Configuration du noeud
wsrep_node_address="10.31.252.10"
wsrep_node_name="node1"

# Méthode de réplication
wsrep_sst_method=rsync
```

BDD 2

```
[mysqld]
# ---- Options essentielles ----
server-id=2
binlog_format=ROW
default_storage_engine=InnoDB
innodb_flush_log_at_trx_commit=2
innodb_autoinc_lock_mode=2

# ---- Activation de Galera ----
```

```
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Nom du cluster
wsrep_cluster_name="mariadb_cluster"

# Liste des noeuds du cluster
wsrep_cluster_address="gcomm://10.31.252.10,10.31.252.11"

# Configuration du noeud
wsrep_node_address="10.31.252.11"
wsrep_node_name="node2"

# M  thode de r  plication
wsrep_sst_method=rsync
```

De plus il va falloir modifier le fichier :

```
/etc/mysql/mariadb.conf.d/50-server.cnf
```

BDD 1

```
[mysqld]

bind-address=0.0.0.0

default_storage_engine=InnoDB

innodb_autoinc_lock_mode=2

binlog_format=row



wsrep_on=ON

wsrep_provider=/usr/lib/galera/libgalera_smm.so

wsrep_cluster_name="galera_cluster"

wsrep_cluster_address="gcomm://10.31.252.10,10.31.252.11"

wsrep_node_address="10.31.252.10"  # Changer sur le second serveur

wsrep_sst_method=rsync
```

BDD2

```
[mysqld]
bind-address=0.0.0.0
default_storage_engine=InnoDB
```

```
innodb_autoinc_lock_mode=2
binlog_format=row


wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://10.31.252.10,10.31.252.11"
wsrep_node_address="10.31.252.11"
wsrep_sst_method=rsync
```

Une fois avoir fait ca on va ouvrir les ports sur les 2 serveurs

```
sudo ufw allow 3306
sudo ufw allow 4567
sudo ufw allow 4568
sudo ufw allow 4444
```

Ensuite on va initialiser le nouveau cluster

```
sudo galera_new_cluster
```

On verifie avec :

```
sudo systemctl status mariadb
```

Ensuite sur la 2eme base de données on va démarrer notre mariadb

```
sudo systemctl start mariadb
```

ensuite on teste grâce a :

```
mysql -u root -p
SHOW STATUS LIKE 'wsrep_cluster_size';
```

Résultat attendu