Création de containers Nginx avec Docker et équilibrage de charge avec HAProxy

Version de PfSense : 2.7.0
Version de Debian : 12
Version de Haproxy : 2.8.3

J'utiliserai une machine virtuelle Debian en guise d'hôte. Je n'aborderai pas l'installation de Debian dans cette procédure.

1 – Installation de Docker

Afin d'installer Docker, il vous suffira de tapez les commandes suivantes dans l'ordre. Si vous souhaitez comprendre ce que font ces commandes n'hésitez pas à chercher sur Internet leur fonctionnement.

```
> sudo apt install ca-certificates curl gnupg lsb-release
> sudo mkdir -m 0755 -p /etc/apt/keyrings
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
> echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

> sudo apt update
> sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-compose

Tous les composants nécessaires au bon fonctionnement de Docker son maintenant installés, Docker est opérationnel.

2 – Création du fichier Docker compose

Docker-compose permet d'orchestrer la création de plusieurs containers, il s'agit d'un fichier au format **.yaml** dans lequel nous allons spécifier différentes options nécessaires au fonctionnement de nos containers. Il faudra nommer votre fichier **docker-compose.yaml**, je vous conseille de le mettre dans un dossier spécifique.

Voici le fichier, les commentaires donnent un peu de contexte :

/!\ Il est important de respecter les tabulations, autrement vous aurez une erreur lors du lancement du docker compose /!

```yaml
version: "3" # La version à utilisé, généralement on laisse la 3.

services: # Nous allons spécifier tous nos containers dans services
  web-server01: # On spécifie le container que nous souhaitons créer
    image: nginx:latest # On spécifie l'image à utiliser pour le container
    container_name: web-server01 # On spécifie le nom du container
    hostname: web-server01 # On spécifie le hostname pour le container
    restart: always # On spécifie que l'on souhaite redémarrer le serveur si celui-ci s'arrête pas inadvertance
    ports: # On spécifie le port à mapper sur notre hôte (hôte:container)
      - "8081:80"
    volumes: # On spécifie le volume Docker à utiliser
      - web_data:/usr/share/nginx/html
    networks: # On spécifie le réseau à utiliser
      - web_servers

  web-server02:
    image: nginx:latest
    container_name: web-server02
    hostname: web-server02
    restart: always
    ports:
      - "8082:80"
    volumes:
      - web_data:/usr/share/nginx/html
    networks:
      - web_servers

networks: # Dans cette partie nous allons spécifier des options pour notre réseau
  web_servers: # On spécifie le nom de notre réseau
    driver: bridge # On spécifie le type de réseau
    ipam:
      config:
        - subnet: 10.0.0.0/24 # On spécifie le subnet de notre réseau
    driver_opts:
      com.docker.network.bridge.name: web_servers # On spécifie le nom de notre réseau à afficher lors d'une commande de type "ip a"

volumes: # Dans cette partie on va spéifier le/les volumes
  web_data: # On spécifie le nom de notre volume.
```

Vous aurez maintenant besoin de connaitre 4 commandes, même si je vous recommande d'étudier en détail le fonctionnement de docker et de docker-compose.

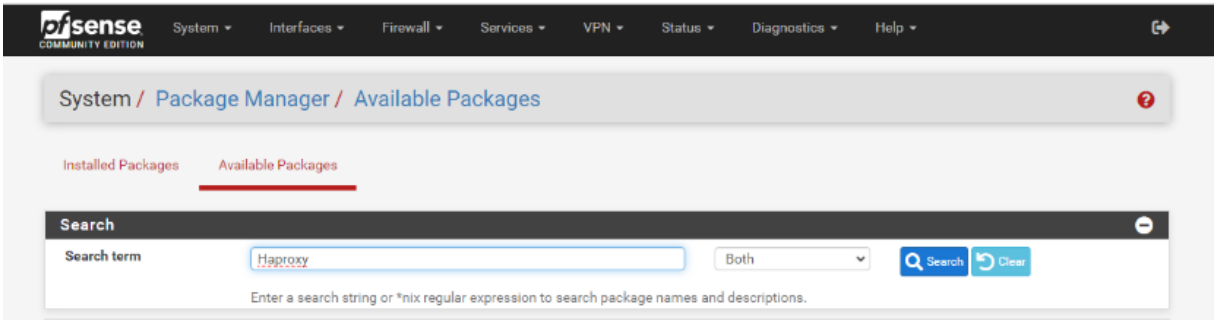
La première commande va permettre de lancer les containers :
> docker-compose -f docker-compose.yaml up -d

La seconde commande va permettre de supprimer les containers :
> docker-compose -f docker-compose.yaml down

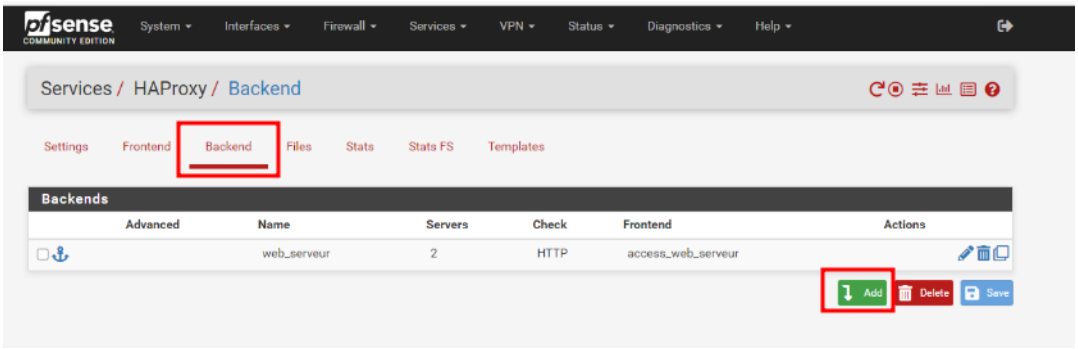
Ces deux commandes vont vous permettre d'arrêter et de démarrer les containers :
> docker-compose -f docker-compose.yaml stop
> docker-compose -f docker-compose.yaml start

3 – Installation de Haproxy et configuration du Backend

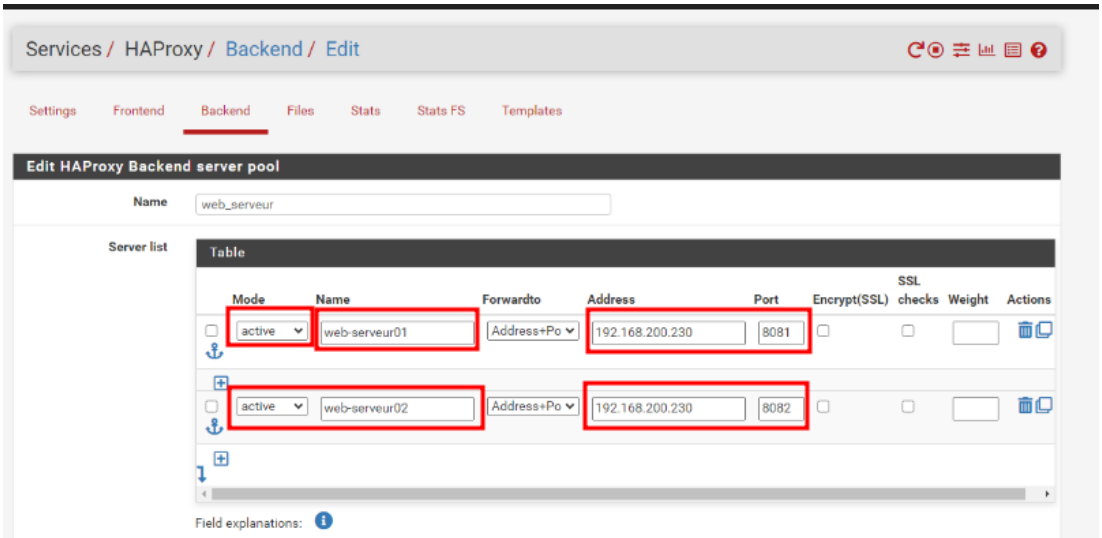
Afin de permettre l'équilibrage de charge, nous allons installer Haproxy directement en tant que package sur PfSense.
Rendez-vous dans **System -> Package Manager -> Available Packages** et installer le.



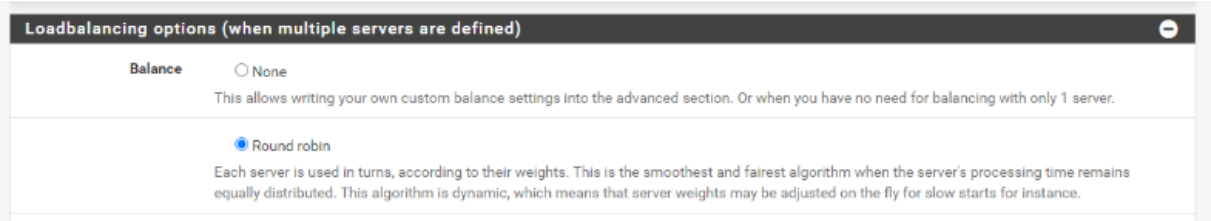
Maintenant que HAProxy est installé, nous allons passer à sa configuration.
Rendez-vous dans **Services -> HAProxy -> Backend.**



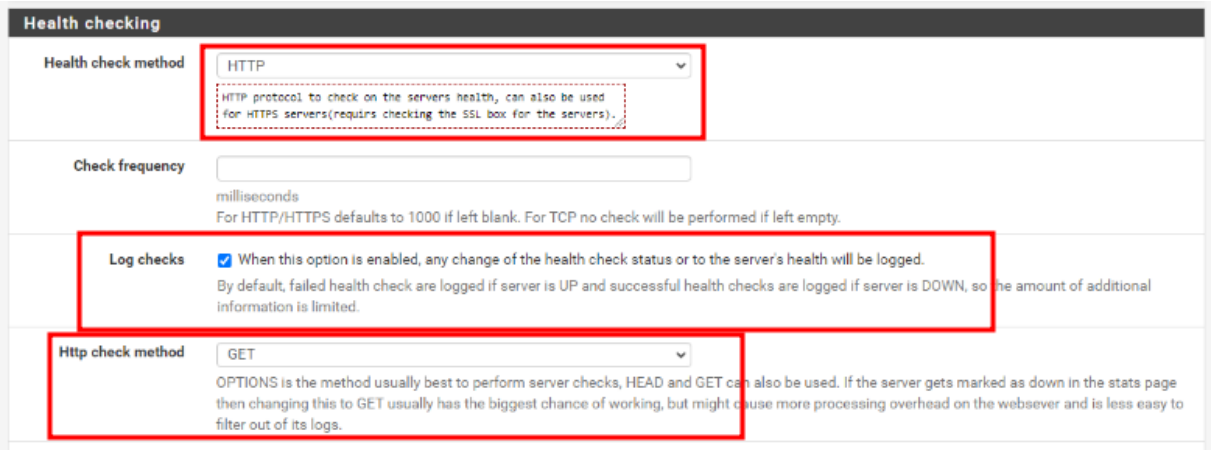
Donnez ensuite un nom à votre pool de serveur et ajoutez vos deux serveurs. Comme les containers sont hébergés sur la machine hôte Debian, il faudra mettre l'IP de cette machine et mettre le port que nous avons mis dans le Docker-compose (8081 et 8082)



Dans la partie **Loadbalancing options**, sélectionnez **Round robin**. Cela définit la méthode d'équilibrage de charge



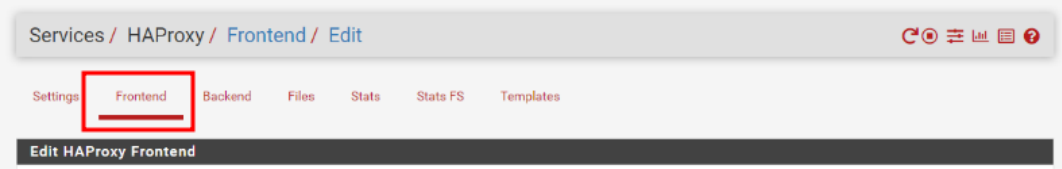
Dans la partie **Health checking**, sélectionnez **HTTP** et **GET** comme méthodes puis activez les logs. Cela va permettre à Haproxy d'envoyer des requêtes sur nos deux containers afin de savoir si ceux-ci sont up ou down.



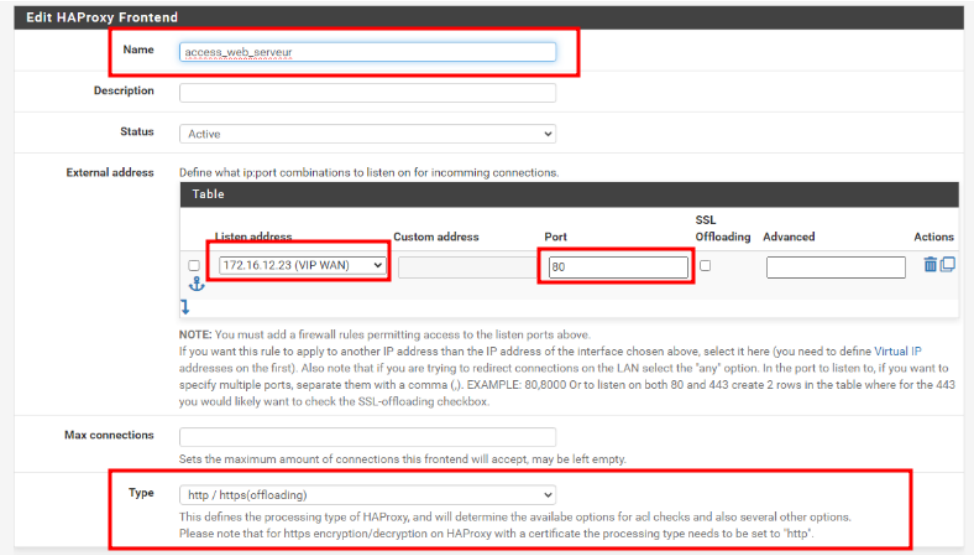
Une fois tout cela configuré, vous pouvez sauvegarder

4 – Configuration du frontend

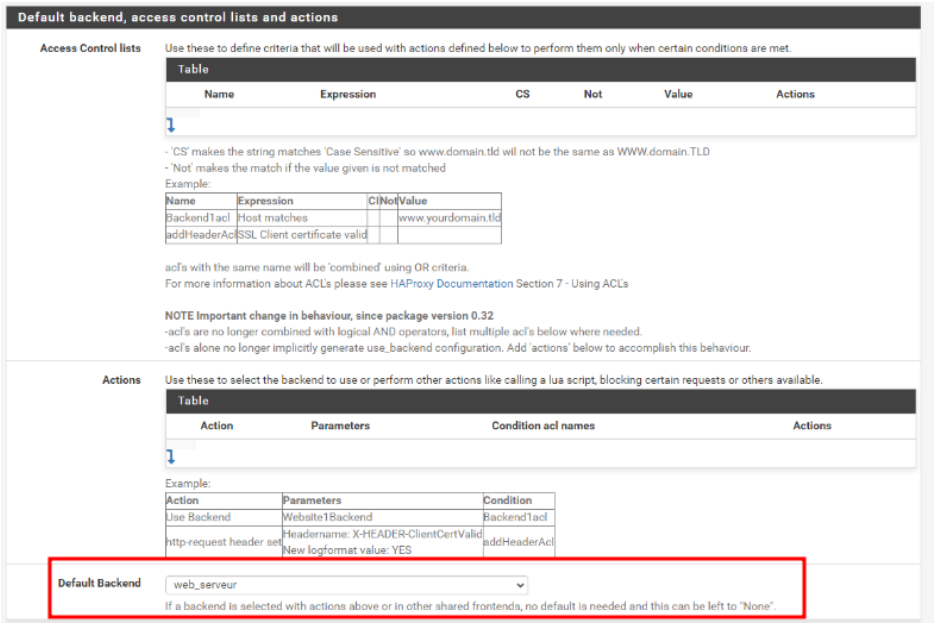
Rendez-vous dans la partie Frontend de HAProxy.



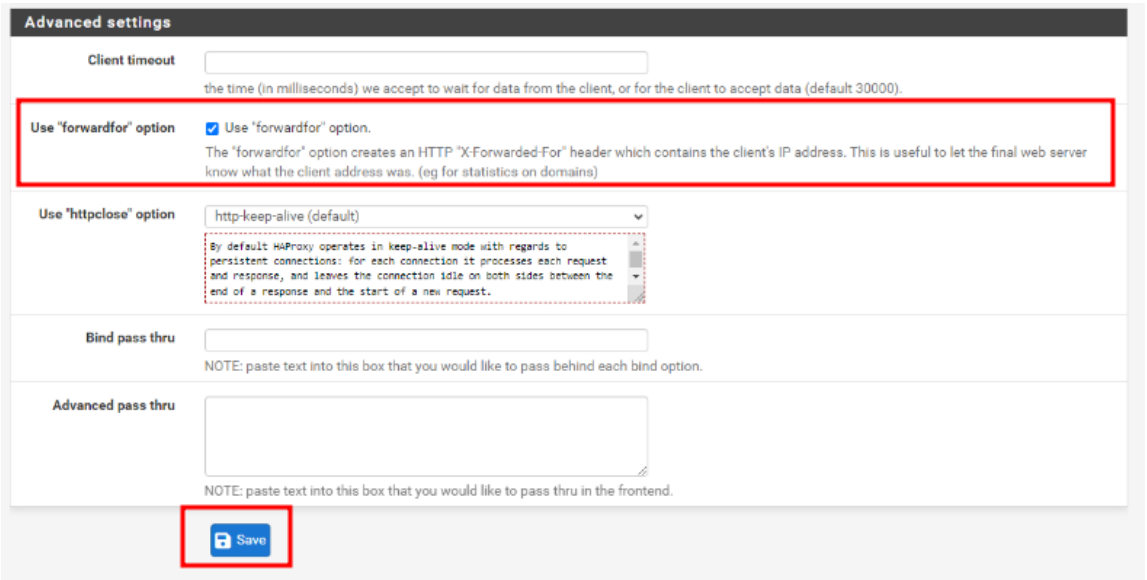
Dans cette partie, il va falloir définir l'adresse en écoute pour recevoir les connexions. Dans mon cas j'ai sélectionné l'adresse virtuelle de mon WAN car j'ai un cluster de 2 PfSense. Cependant, dans votre cas, sélectionnez votre interface WAN et définissez le port sur 80 (HTTP).



Dans la partie **Default backend**, sélectionnez le Backend que nous avons précédemment créé

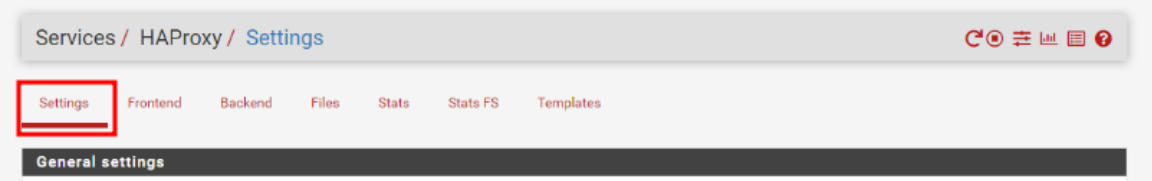


Enfin, dans la partie **Advanced settings**, sélectionnez **Use forwardfor options**. Cela va permettre à HAProxy de savoir l'adresse IP ayant fait la requête.
Une fois cela fait, sauvegardez.

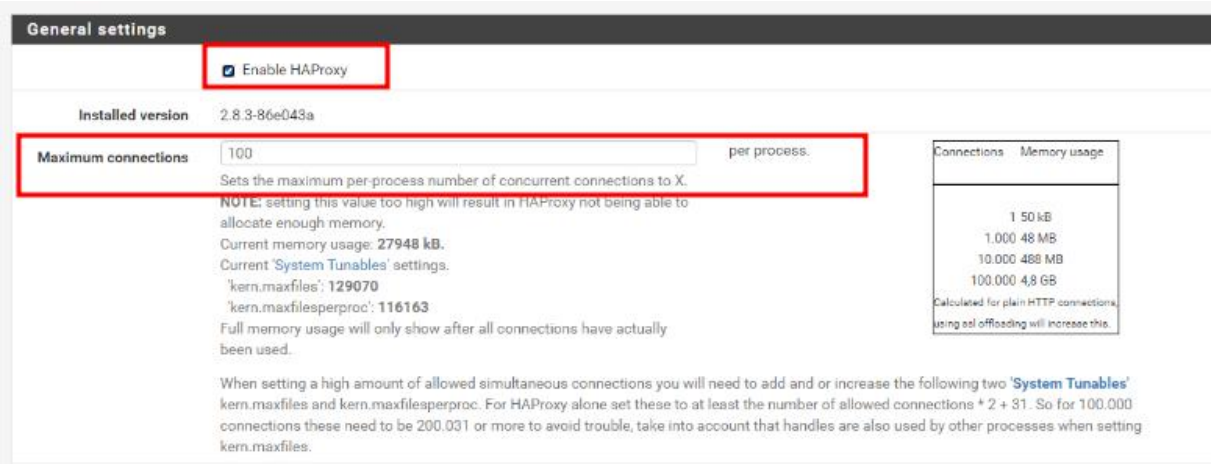


5 – Configuration générale de HAProxy

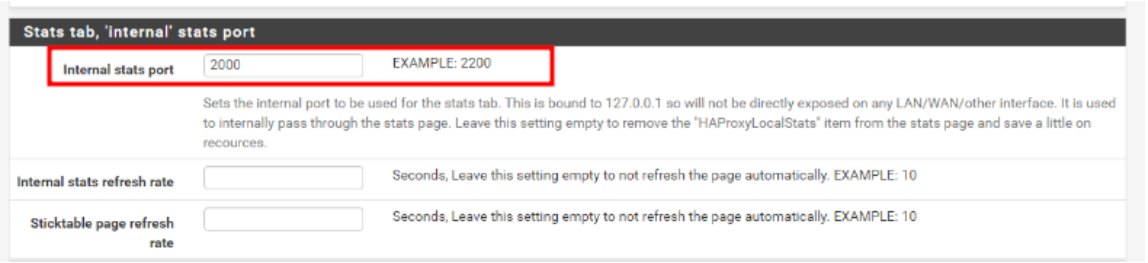
Rendez-vous dans la partie **Settings** de HAProxy.



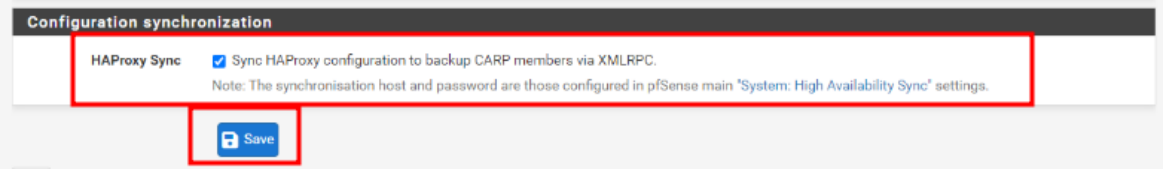
Sélectionnez un nombre maximum de connexions, pour ma part j'ai mis 100 mais vous pouvez mettre la valeur que vous souhaitez.



Définissez ensuite le port interne pour accéder aux statistiques de HAProxy.



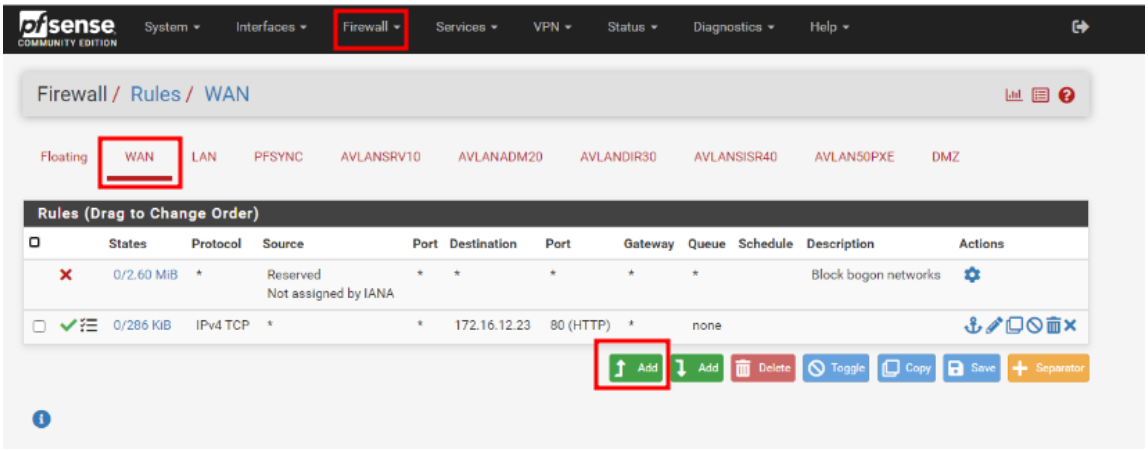
Si vous avez un cluster de PfSense comme c'est mon cas, n'hésitez pas à cocher la case permettant la synchronisation de HAProxy entre les deux PfSense



6 – Création de la règle pare-feu WAN

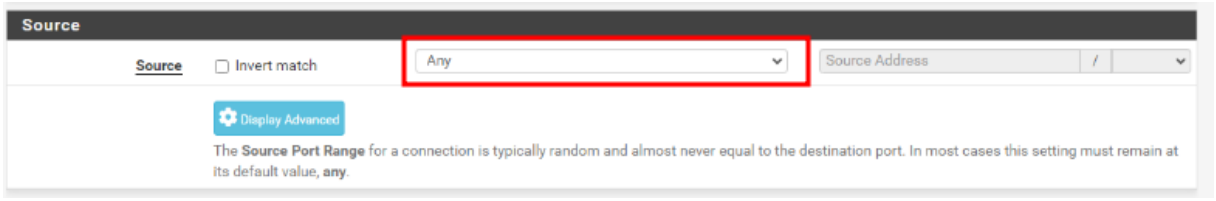
Maintenant, afin que les requêtes puissent être rediriger depuis le WAN, il va falloir créer une règle autorisant cela. Je tiens à préciser que dans mon cas le serveur hébergeant les containers Docker est dans une DMZ.

Pour créer la règle WAN, rendez-vous dans **Firewall -> Rules -> WAN**



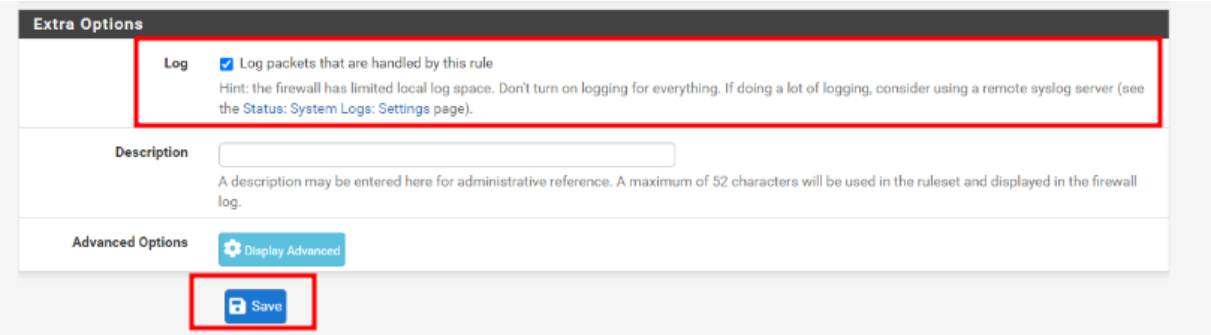
Laissez toutes les première options par défaut, nous allons nous intéresser à la partie **Source** et **Destination**.

Dans la partie **Source**, étant donné que mon serveur est isolé dans une DMZ, je vais mettre **any** afin de permettre à n'importe qui de pouvoir accéder à mon serveur web.



Dans la partie **Destination**, je vais préciser l'adresse IP virtuelle de mon WAN. Si vous n'avez pas de cluster sélectionnez alors l'interface WAN.
Concernant le port, étant donné que nous sommes en HTTP nous allons sélectionner le port 80.

Une fois cela fait, vous pouvez sauvegarder .



7 – Modification de index.html

Dans la partie Docker-compose, nous avons spécifier un volume docker qui est le même pour les deux serveurs. Cela signifie que si nous modifions le contenu présent dans ce volume, alors la modification sera effective sur les deux serveurs.

Afin de voir l'emplacement de ce volume sur notre machine, nous pouvons utiliser les commandes suivantes :

> docker volume ls (permet d'afficher les volumes)

```
root@SRVA-DOCKER:~/docker_web# docker volume ls
DRIVER    VOLUME NAME
local     docker_web_web_data
```

> docker volume inspect <nom_volume> (permet de voir les metadas du volume)

```
root@SRVA-DOCKER:~/docker_web# docker volume inspect docker_web_web_data
[
    {
        "CreatedAt": "2024-03-10T17:39:07+01:00",
        "Driver": "local",
        "Labels": {
            "com.docker.compose.project": "docker_web",
            "com.docker.compose.version": "1.29.2",
            "com.docker.compose.volume": "web_data"
        },
        "Mountpoint": "/var/lib/docker/volumes/docker_web_web_data/_data",
        "Name": "docker_web_web_data",
        "Options": null,
        "Scope": "local"
    }
]
```

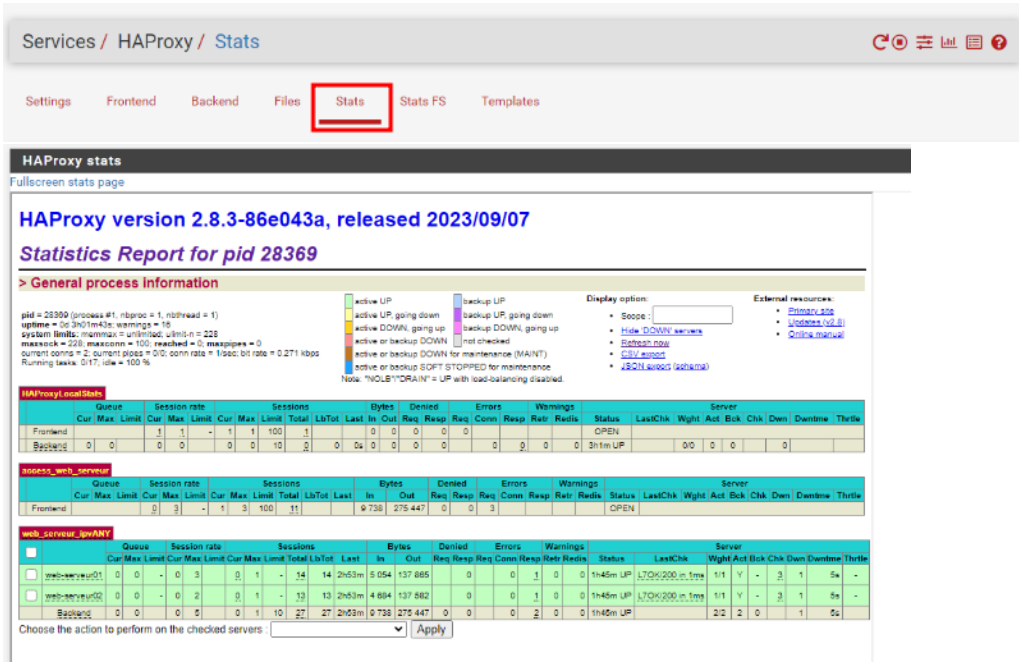
On trouve ici notre chemin **Mountpoint.**
Il va ensuite falloir se rendre dans le dossier **_data** et modifier les fichiers afin de personnaliser votre site web.

```
root@SRVA-DOCKER:~/docker_web# cd /var/lib/docker/volumes/docker_web_web_data/_data
root@SRVA-DOCKER:/var/lib/docker/volumes/docker_web_web_data/_data# ls
 50x.html   index.html   style.css  'undraw_dev_productivity_umsq 1 .svg'
```

8 - Test du bon fonctionnement

Afin de vérifier que notre configuration est fonctionnelle, il suffit de vous rendre sur l'IP WAN que vous avez configurez lors de la partie frontend.

Vous pouvez également vérifier dans la partie « Stats » les requêtes sur le backend et si les containers sont disponibles



Fin de la procédure.