14 AVRIL 2025

DEPLOIEMENT D’UN SITE WEB PUBLIQUE AVEC UNE BASE POSTGRESQL.

PERIER--PICARD VALENTIN

Table des matières

Infrastructure

Les serveurs :

L'entreprise SuperInfo possède 3 serveurs, 1 switch et 1 OPNsense.

- **Premier serveur** :
  CPU: 2x Intel(R) Xeon(R) CPU E5620 8 cœurs
  RAM: 16 Go DDR3
  Stockage: 4 HDD de 130Go en RAID 5 physique pour un total de 400Go utilisable
- **Deuxième serveur** :
  CPU: 13th Gen Intel(R) Core(TM) i7-13700 24 cœurs
  RAM: 32 Go DDR5
  Stockage: 1 SSD Nvme 500Go utilisable
- **Troisième serveur** :
  CPU: 13th Gen Intel(R) Core(TM) i7-13700 24 cœurs
  RAM: 32 Go DDR5
  Stockage: 1 SSD Nvme 500Go utilisable
- **OPNsense :**
  CPU: Intel® Core™ i5-6600 4 coeurs
  RAM: 16 Go DDR3
  Stockage: 1 HDD de 500Go utilisable

L'entreprise SuperInfo a décidé de déployer Proxmox en cluster sur le serveur 1, 2 et 3.
Elle a aussi choisi d'utiliser OPNsense comme routeur installé sur le serveur OPNsense.

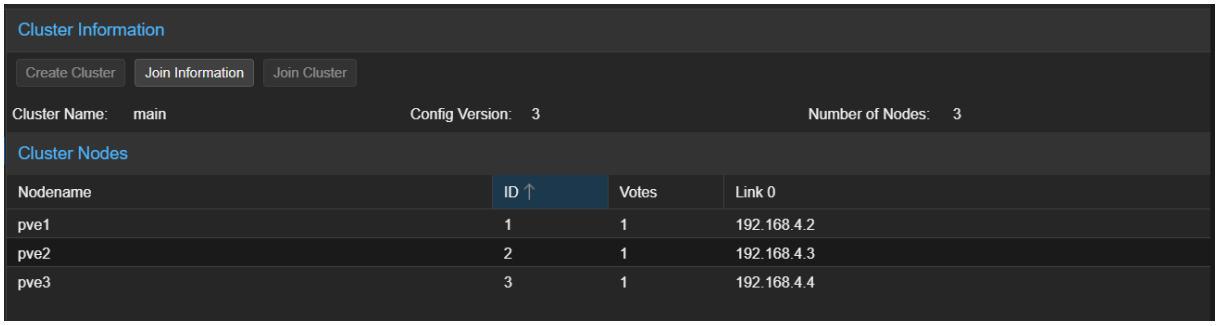
Déploiement de Proxmox.

On effectue une installation de Proxmox basique.
Les serveurs étant sur le Vlan 4 « serveur » on modifie durant l'installation leurs ips :

- 192.168.4.2
- 192.168.4.3
- 192.168.4.4

Les serveurs 2 et 3 on deux interfaces, on connecte donc la deuxième interface sur le Vlan 10 « DMZ » sans leur attribuer d'ips.

Nous pouvons ensuite les faire rejoindre un cluster.



Nous pouvons ensuite exécuter les script Proxmox VE Post Install de Proxmox VE Helper Scripts afin de terminer l'installation de Proxmox et désactiver les options inutiles.

Déploiement de OPNsense

OPNsense sera installé sur le serveur 4.

Le serveur OPNsense possède deux interfaces :

- Interface 1 WAN
- Interface 2 LAN

On créer dessus 3 Vlans afin de séparer les différents réseaux :

- Vlan 4 Serveurs
  192.168.4.1/24
- Vlan 10 DMZ
  192.168.10.1/24
- Vlan 50 Clients
  192.168.50.1/24

Mise en place de la base de données

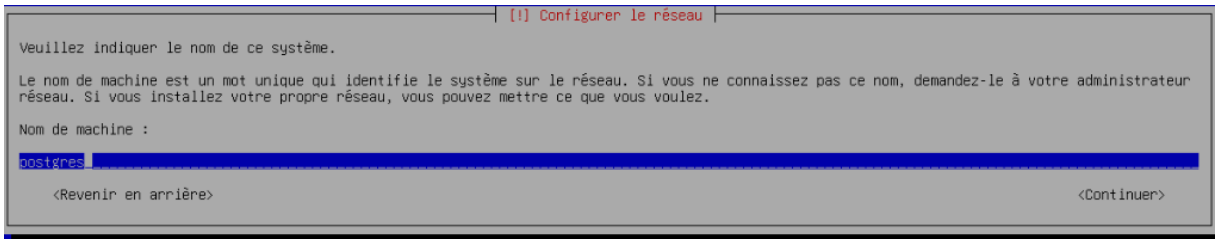
Installation des VMs postgres

SuperInfo a fait le choix d'utiliser Debian 12.

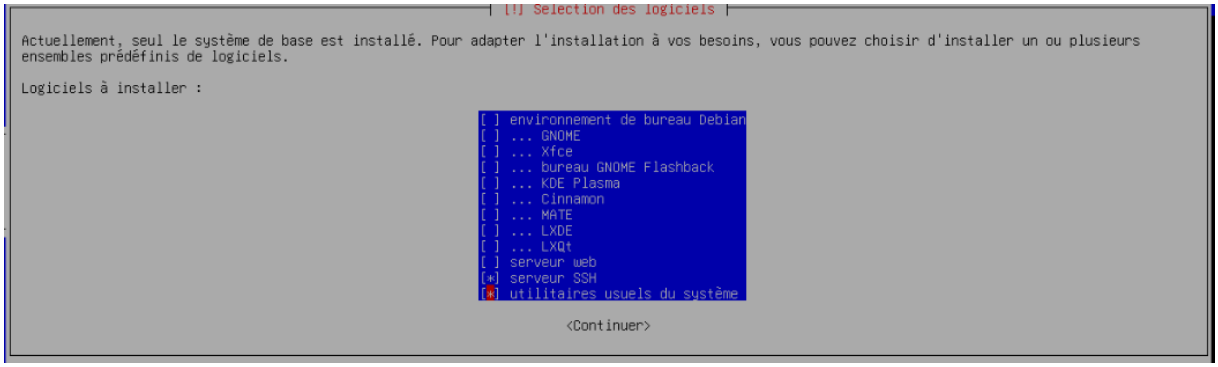
Les VMs postgres ont 2 CPU et 4 Go de RAM et un disque de 20Go chacune.
Elles n'ont que accès a l'interface sur le Vlan 4.

On procède à une installation Debian classique :

Installer en Français.
On a nommé les VMs « postgres » et « postgresRep »



On a effectué une installation sous LVM non chiffré avec le dossier /home séparé.
Seul un serveur SSH et les utilitaires systèmes sont installés sur la VM.



Une fois la vm installé on vérifi que qemu-guest-agent est bien installé sur la vm.

```
root@isntall:~# apt install qemu-guest-agent
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
qemu-guest-agent est déjà la version la plus récente (1:7.2+dfsg-7+deb12u12).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

Installation de Postgres

Nous pouvons installer sur les deux machine postgresql :

apt install postgresql

systemctl status postgresql

```
root@isntall:~# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; preset: enabled)
     Active: active (exited) since Sat 2025-04-12 14:09:10 CEST; 3min 10s ago
   Main PID: 2379 (code=exited, status=0/SUCCESS)
        CPU: 453us

avril 12 14:09:10 isntall systemd[1]: Starting postgresql.service - PostgreSQL RDBMS...
avril 12 14:09:10 isntall systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.
```

Les deux instances fonctionnent bien.

Mise en place de la réplication

Tout d'abord se connecter à la console PostgreSQL pour créer un utilisateur dédier à la réplication.
sudo -u postgres psql
CREATE ROLE replica_user WITH REPLICATION LOGIN PASSWORD 'Azerty10';
On modifie ensuite l'adress d'acoute dans /etc/postgresql/15/main/postgresql.conf.

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '192.168.4.25'               # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for all
```

On modifie aussi les WAL afin que les serveurs puissent fusionner leurs log.

```
#------------------------------------------------------------------------------
# WRITE-AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

wal_level = logical                     # minimal, replica, or logical
                                        # (change requires restart)
#fsync = on                             # flush data to disk for crash safety
```

On peux ensuite autoriser les autre host sur le network a se connecter au serveur postgresql et au serveur de réplication a se connecter avec le compte replica_user.

```
  GNU nano 7.2                          /etc/postgresql/15/main/pg_hba.conf
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            scram-sha-256
host    all             all             192.168.4.1/24          md5
# IPv6 local connections:
host    all             all             ::1/128                 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            scram-sha-256
host    replication     all             ::1/128                 scram-sha-256
host    replication     replica_user    192.168.4.29/32         md5
```

On restart ensuite postgresql sur la machine.
systemctl restart postgresql

On passe ensuite sur la deuxième VM.
Pour effectuer la replication la deuxième instance de postgresql doit être à l'arrêt on stop donc le service.
systemctl stop postgresql
On supprime aussi tous les config sur la deuxième instance :
rm -rv /var/lib/postgresql/15/main/
Et on peut lancer l'utilitaire pour actier la replication :
pg_basebackup -h 192.168.4.25 -U replica_user -X stream -C -S replica_1 -v -R -W -D /var/lib/postgresql/15/main/
On réattribue le dossier à l'utilisateur postgres :
chown postgres -R /var/lib/postgresql/15/main/
Et on démarre le service :
systemctl start postgresql

On peut ensuite tester la réplication en effectuant cette commande dans la console de postgreSQL.

SELECT client_addr, state FROM pg_stat_replication;

```
root@postgres:~# sudo -u postgres psql
could not change directory to "/root": Permission non accordée
psql (15.12 (Debian 15.12-0+deb12u2))
Type "help" for help.

postgres=# SELECT client_addr, state
FROM pg_stat_replication;
 client_addr  |   state
--------------+-----------
 192.168.4.29 | streaming
(1 row)
```

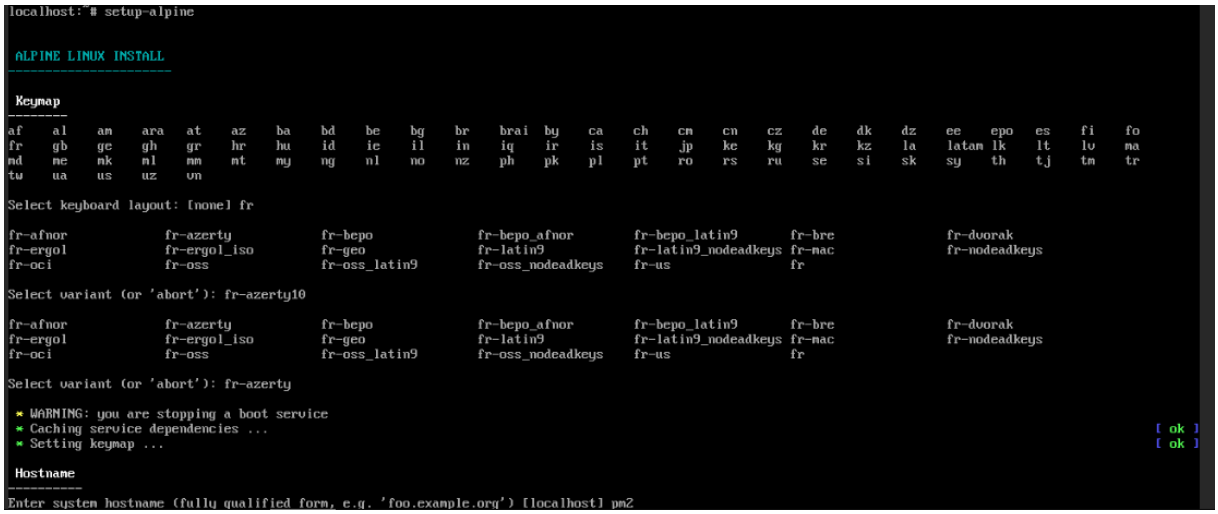
Mise en place du backend

Installation de la VM

Cette fois SuperInfo a décidé d'utiliser Alpine Linux comme système d'exploitation pour son serveur.

La VM a 1 CPU et 2 Go de RAM et un disque de 5 Go.

On procède à une installation d'Alpine classique avec la commande setup-alpine :



On nomme la VM pm2.

Durant l'installation il ne faut pas oublier d'activer les dépôts communautaires.

```
APK Mirror
------------
 (f)    Find and use fastest mirror
 (s)    Show mirrorlist
 (r)    Use random mirror
 (e)    Edit /etc/apk/repositories with text editor
 (c)    Community repo enable
 (skip) Skip setting up apk repositories

Enter mirror number or URL: [1] c

Community repository enabled

Community repository enabled
 (f)    Find and use fastest mirror
 (s)    Show mirrorlist
 (r)    Use random mirror
 (e)    Edit /etc/apk/repositories with text editor
 (c)    Community repo disable
 (skip) Skip setting up apk repositories

Enter mirror number or URL: [1]
```

Une fois l'installation terminé on peut redémarrer le système en enlevant le disque d'installation de la VM.

On installe ensuite qemu-guest-agent et on ajoute au démarrage :

```
alpine:~# apk update
fetch http://alpinelinux.mirrors.ovh.net/v3.21/main/x86_64/APKINDEX.tar.gz
fetch http://alpinelinux.mirrors.ovh.net/v3.21/community/x86_64/APKINDEX.tar.gz
v3.21.3-305-gf53b3a2c730 [http://alpinelinux.mirrors.ovh.net/v3.21/main]
v3.21.3-306-g591e6485dc1 [http://alpinelinux.mirrors.ovh.net/v3.21/community]
OK: 25395 distinct packages available
alpine:~# apk add qemu-guest-agent
(1/9) Installing libffi (3.4.7-r0)
(2/9) Installing libintl (0.22.5-r0)
(3/9) Installing libmount (2.40.4-r0)
(4/9) Installing pcre2 (10.43-r0)
(5/9) Installing glib (2.82.5-r0)
(6/9) Installing numactl (2.0.18-r0)
(7/9) Installing liburing (2.8-r0)
(8/9) Installing qemu-guest-agent (9.1.2-r1)
(9/9) Installing qemu-guest-agent-openrc (9.1.2-r1)
Executing busybox-1.37.0-r12.trigger
Executing glib-2.82.5-r0.trigger
OK: 140 MiB in 65 packages
alpine:~# rc-update add qemu-guest-agent
 * service qemu-guest-agent added to runlevel default
```

Installation de nodejs, npm et npm

On installe nodejs et npm via le gestionnaire de paquet apk.

```
alpine:~# apk add nodejs npm
(1/11) Installing ca-certificates (20241121-r1)
(2/11) Installing libgcc (14.2.0-r4)
(3/11) Installing libstdc++ (14.2.0-r4)
(4/11) Installing ada-libs (2.9.2-r1)
(5/11) Installing icu-data-en (74.2-r0)
Executing icu-data-en-74.2-r0.post-install
*
* If you need ICU with non-English locales and legacy charset support, install
* package icu-data-full.
*
(6/11) Installing icu-libs (74.2-r0)
(7/11) Installing simdjson (3.10.1-r0)
(8/11) Installing simdutf (5.6.3-r0)
(9/11) Installing sqlite-libs (3.48.0-r0)
(10/11) Installing nodejs (22.13.1-r0)
(11/11) Installing npm (10.9.1-r0)
Executing busybox-1.37.0-r12.trigger
Executing ca-certificates-20241121-r1.trigger
OK: 212 MiB in 86 packages
```

On installe ensuite pm2 via npm avec l'option -g pour qu'il s'installe globalement.
On l'ajoute aux scripts de startup avec la commande : pm2 startup

```
Target path
/etc/init.d/pm2
Command list
[ 'chmod +x /etc/init.d/pm2', 'rc-update add pm2 default' ]
[PM2] Writing init configuration in /etc/init.d/pm2
[PM2] Making script booting at startup...
[PM2] [-] Executing: chmod +x /etc/init.d/pm2...
[PM2] [v] Command successfully executed.
[PM2] [-] Executing: rc-update add pm2 default...
 * service pm2 added to runlevel default
[PM2] [v] Command successfully executed.
+---------------------------------------+
[PM2] Freeze a process list on reboot via:
$ pm2 save

[PM2] Remove init script via:
$ pm2 unstartup openrc
```

Démarrage du backend

On peut copier le serveur de backend dans le dossier /scripts/preseed
puis installer les dépendance npm :

```
pm2:/script# npm install

added 105 packages, and audited 106 packages in 5s

21 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
pm2:/script#
```

On lance le serveur de backend avec la commande :

```
pm2:~#  cd /scripts/preseed/ && pm2 start server.js --name "preseed" --watch --time
```

on peux vérifier qu'il est bien lancé avec la commande : pm2 list

```
pm2:/script# pm2 list
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	preseed	default	1.0.0	fork	2992	6s	31	online	0%	66.7mb	root	enabled

On fait ensuite pm2 save pour qu'il démarre automatiquement :

```
pm2:~# pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /root/.pm2/dump.pm2
```

Mise en place de Nginx

Installation de la VM Nginx

Pour le serveur Nginx, SuperInfo a aussi décidé d'utiliser Alpine Linux.
Nous allons donc suivre le même processus d'installation que le serveur de backend.
Le serveur Nginx a accès aux deux interfaces Vlan 4 et Vlan 10.

Installation de Nginx

On installe Nginx sur la VM grâce au gestionnaire de paquet apk :

```
alpine:~# apk add nginx
(1/3) Installing pcre (8.45-r3)
(2/3) Installing nginx (1.26.3-r0)
Executing nginx-1.26.3-r0.pre-install
Executing nginx-1.26.3-r0.post-install
(3/3) Installing nginx-openrc (1.26.3-r0)
Executing busybox-1.37.0-r12.trigger
OK: 214 MiB in 89 packages
```

On ajoute un fichier proxy_params qui contient la configuration du reverse proxy.

```
nginx:~# ls /etc/nginx/
dhparam.pem     fastcgi_params  mime.types      nginx.conf      scgi_params
fastcgi.conf    http.d          modules         proxy_params    uwsgi_params
```

```
proxy_set_header  Upgrade           $http_upgrade;
proxy_set_header Connection         "upgrade";
proxy_pass_request_headers          on;
proxy_buffering                     off;
client_max_body_size                0;
proxy_connect_timeout               60s;
proxy_read_timeout                  60s;
proxy_send_timeout                  60s;
send_timeout                        60s;
proxy_buffer_size                   4k;
proxy_buffers                       4 32k;
proxy_busy_buffers_size             64k;
proxy_temp_file_write_size          64k;
proxy_set_header   Host                 $host;
#proxy_set_header   Host                 $host:$server_port;
proxy_set_header   X-Real-IP            $remote_addr;
#proxy_set_header  X-Forwarded-For      $proxy_add_x_forwarded_for;
```

On ajoute aux scripts de démarrage :
rc-update add nginx

Mise en place du serveur web

On créer un fichier preseed.conf dans le dossier http.d qui contient la config du frontend et backend pour le site web.

```
server {
        listen 443 ssl;
        server_name preseed.lan;
        http2  on;

        ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;

        location / {
                root /www/preseed;
        }
}


server {
        listen 443 ssl;
        server_name seed.lan;

        ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;


        location / {
                include proxy_params;
                proxy_pass http://192.168.4.51:39843;
        }
}
```

On copie ensuite les fichiers du site dans le dossier /www/preseed

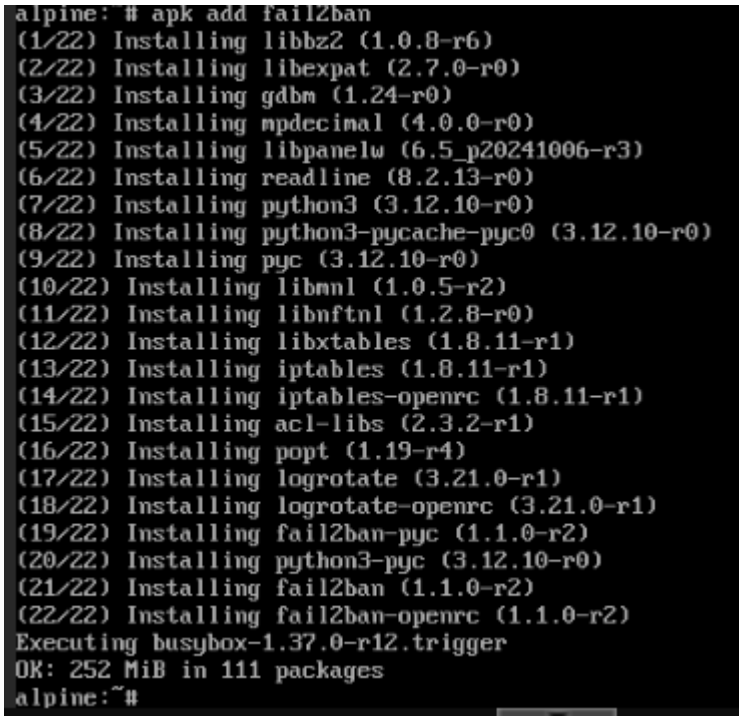
```
nginx:~# ls /www
preseed
nginx:~# ls /www/preseed/
about.html   css          img          index.html   js           preseed.js   robots.txt   sitemap.xml
```

On créer aussi un certificat SSL afin que le site soit accessible en SSL.

```
nginx:~# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
nginx:~# openssl dhparam -out /etc/nginx/dhparam.pem 4096
```

Installation de Fail2ban

On installe Fail2ban sur la machine avec le gestionnaire de paquet apk.



On l'ajoute aux scripts de démarrage :

rc-update add fail2ban

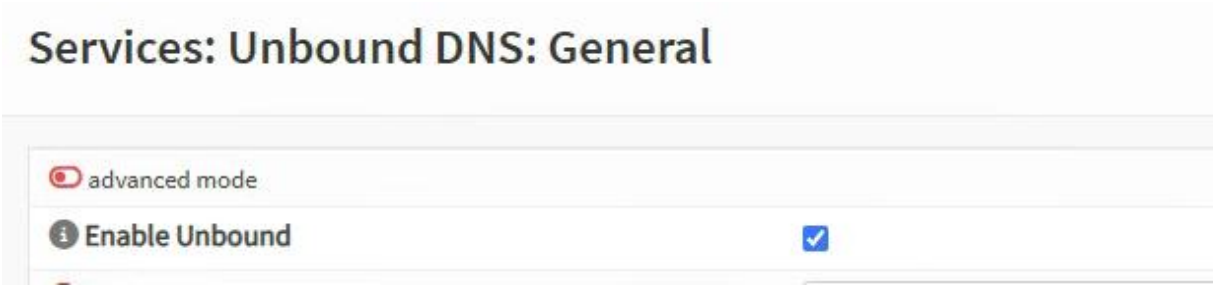
Pour activer fail2ban on ajoute enabled = true aux jail que l'on veut activer dans le fichier jail.local de la configuration de fail2ban

```
# To use more aggressive http-auth modes set filter parameter "mode" in jail.local:
# normal (default), aggressive (combines all), auth or fallback
# See "tests/files/logs/nginx-http-auth" or "filter.d/nginx-http-auth.conf" for usage example and details.
[nginx-http-auth]
# mode = normal
enabled = true
port    = http,https
logpath = %(nginx_error_log)s

# To use 'nginx-limit-req' jail you should have `ngx_http_limit_req_module`
# and define `limit_req` and `limit_req_zone` as described in nginx documentation
# http://nginx.org/en/docs/http/ngx_http_limit_req_module.html
# or for example see in 'config/filter.d/nginx-limit-req.conf'
[nginx-limit-req]
port    = http,https
logpath = %(nginx_error_log)s

[nginx-botsearch]
enabled = true
port     = http,https
logpath  = %(nginx_error_log)s

[nginx-bad-request]
enabled = true
port    = http,https
logpath = %(nginx_access_log)s

[nginx-forbidden]
enabled = true
port     = http,https
logpath  = %(nginx_error_log)s
```

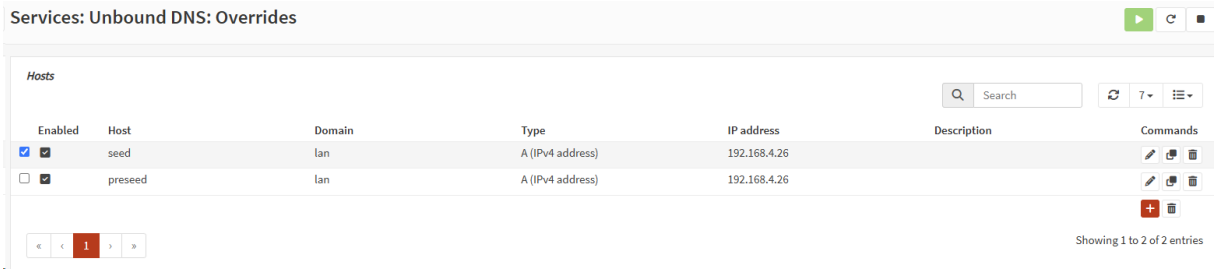
Parametrage de Unbound DNS

Activation du service dans OPNsense

Sur OPNsense il suffit d'aller dans Services->Unbound DNS et d'activer le service :



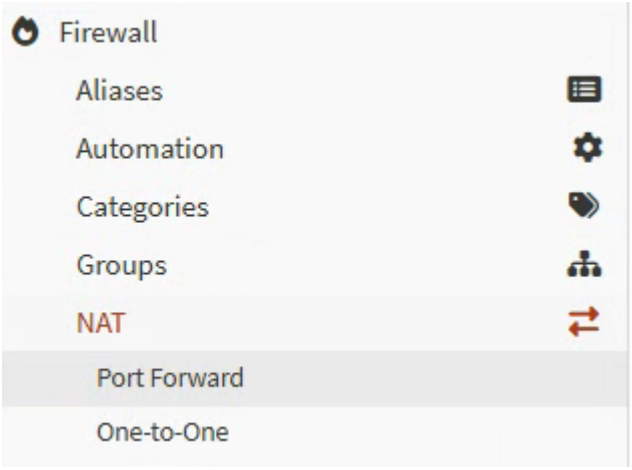
On ajoute ensuite les Overrides DNS pour notre serveur web



Mise en place du Port Forwarding

Paramétrage dans OPNsense

Dans Firewall->NAT->Port Forwrd

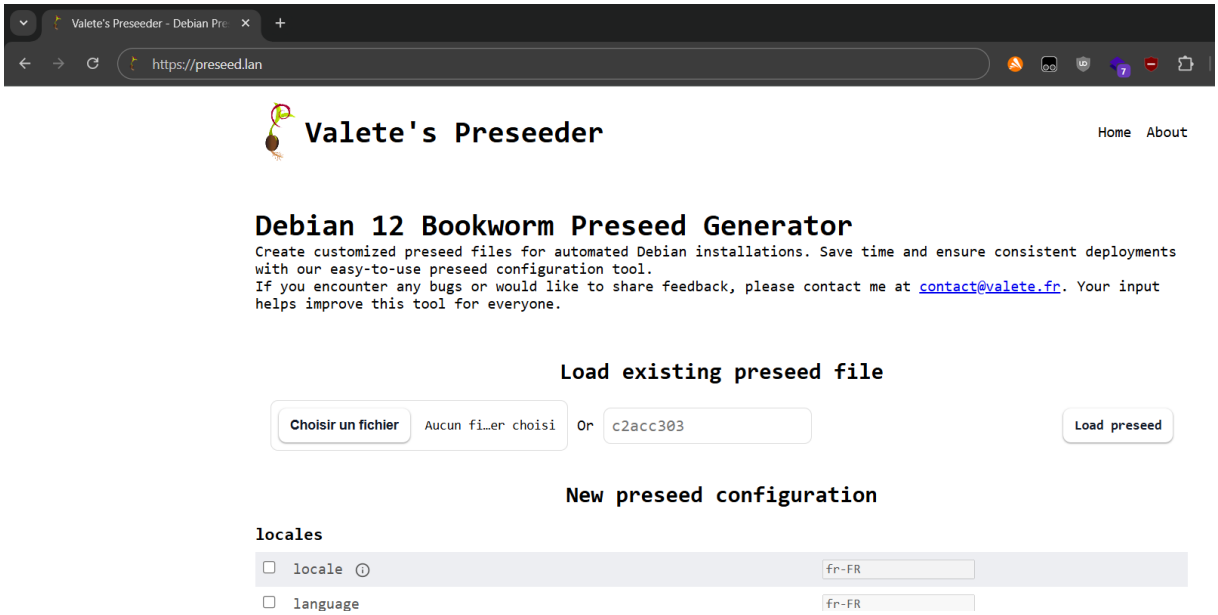


On ajoute une règle qui redirige toutes les requêtes sur l'adresse WAN et le port 443 vers le serveur Nginx.

Test du service

Test avec un ordinateur client connecté sur le serveur

Il essaie d'accéder à preseed.lan via un navigateur internet



L'utilisateur accède bien au service en local.