Nginx: configuration pour un revers_proxy

Comment mettre ne place un revers_proxy

1) NGINX comme revers_proxy_SSL

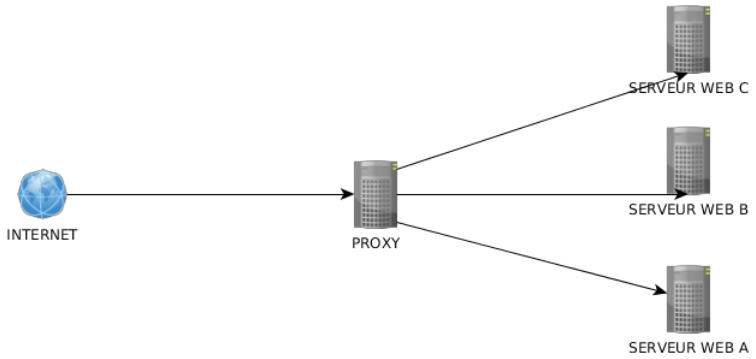
Logo Nginx

Lien vers le site Nginx

Le **revers_proxy** permet de présenter un serveur en *frontal* qui receptionnera les requêtes *HTTP* à destination des serveurs *web* cachés derrières.
Definition wikipedia Cette technique permet * Mémoire cache : le proxy inverse peut décharger les serveurs Web de la charge de pages/objets statiques (pages HTML, images) par la gestion d'un cache web local. La charge des serveurs Web est ainsi généralement diminuée, On parle alors d'« accélérateur web » ou d'« accélérateur HTTP ». * Intermédiaire de sécurité : le proxy inverse protège un serveur Web des attaques provenant de l'extérieur. En effet, la couche supplémentaire apportée par les proxys inverses peut apporter une sécurité supplémentaire. La ré-écriture programmable des URL permet de masquer et de contrôler, par exemple, l'architecture d'un site web interne. Mais cette architecture permet surtout le filtrage en un point unique des accès aux ressources Web. * Chiffrement SSL : le proxy inverse peut être utilisé en tant que « terminateur SSL », par exemple par l'entremise de matériel dédié, * Répartition de charge : le proxy inverse peut distribuer la charge d'un site unique sur plusieurs serveurs Web applicatifs. Selon sa configuration, un travail de ré-écriture d'URL sera donc nécessaire, * Compression : le proxy inverse peut optimiser la compression du contenu des sites.

1.1) Contexte

Le *revers_proxy_SSL* va me servir à rediriger les requêtes sur des *dockers* , en utilisant des certificats



situés en local sur le serveur.

1.2) Chiffrement SSL letsencrypt

Let's Encrypt

Letsencrypt nous permet d'utiliser le protocole **https** en signant et chiffrant nos connexions. Pour que cela fonctionne il faut que les enregistrements *DNS* existent car une des méthodes de vérification de letsencrypt consiste à vérifier l'enregistrement *DNS*, c'est très pratique.
Suivre la procédure ici ### Installation Nginx

```
sudo apt update
sudo apt install nginx-full
```

1.2.1) Configuration

Je modifie seulement le fichier `/etc/nginx/default.conf`, et y place l'ensemble des redirections à cet endroit.

1.2.1.1) Template

copier-coller le bloc autant de fois que vous avez de *dockers*.
`server_tokens off` placé à off, ne renverra pas la version du serveur.
Le code :

```
if ($scheme = http){

   return 301 https://$server_name$request_uri;
}
```

redirigera automatiquement sur le `https` en indiquant l'erreur **301 moved permanently**.
Pour rappel , les page web ne disparraissent pas dans des couches obscure type web profond, cela n'éxiste pas.
La directive *location* indique l'emplacement des fichiers `/application/index.html`, ici ce n'est pas utile, j'indique l'IP du serveur ainsi que son port, le docker peut très bien être sur une machine diférente que le serveur `nginx`.

```
location / {
    proxy_pass http://IP_du_serveur:port;
}
```

Voici le fichier que j'utilise, à adapter en fonction de son besoin.

```
server {
    listen       80;
    listen  443 ssl;
    server_name  sous.domaine.com;
    server_tokens off;

 ## Certificates
 ssl_certificate /etc/letsencrypt/live/sous.domaine.com/fullchain.pem;
 ssl_certificate_key /etc/letsencrypt/live/sous.domaine.com/privkey.pem;
 ssl_trusted_certificate /etc/letsencrypt/live/sous.domaine.com/chain.pem;
 if ($scheme = http){

    return 301 https://$server_name$request_uri;
}
location / {
    proxy_pass http://IP_du_serveur:port;
}

 ## Protocol
 ssl_protocols TLSv1.2;

 ## Diffie-Hellman
 ssl_ecdh_curve secp384r1;

 ## Ciphers
 ssl_ciphers EECDH+CHACHA20:EECDH+AESGCM:EECDH+AES;
 ssl_prefer_server_ciphers on;

 # OCSP Stapling
 ssl_stapling on;
 ssl_stapling_verify on;

 ## TLS parameters
 ssl_session_cache shared:SSL:10m;
 ssl_session_timeout 5m;
 ssl_session_tickets off;

 ## HSTS
 add_header Strict-Transport-Security "max-age=15552000; includeSubdomains; preload";
}
```

1.2.2) Test

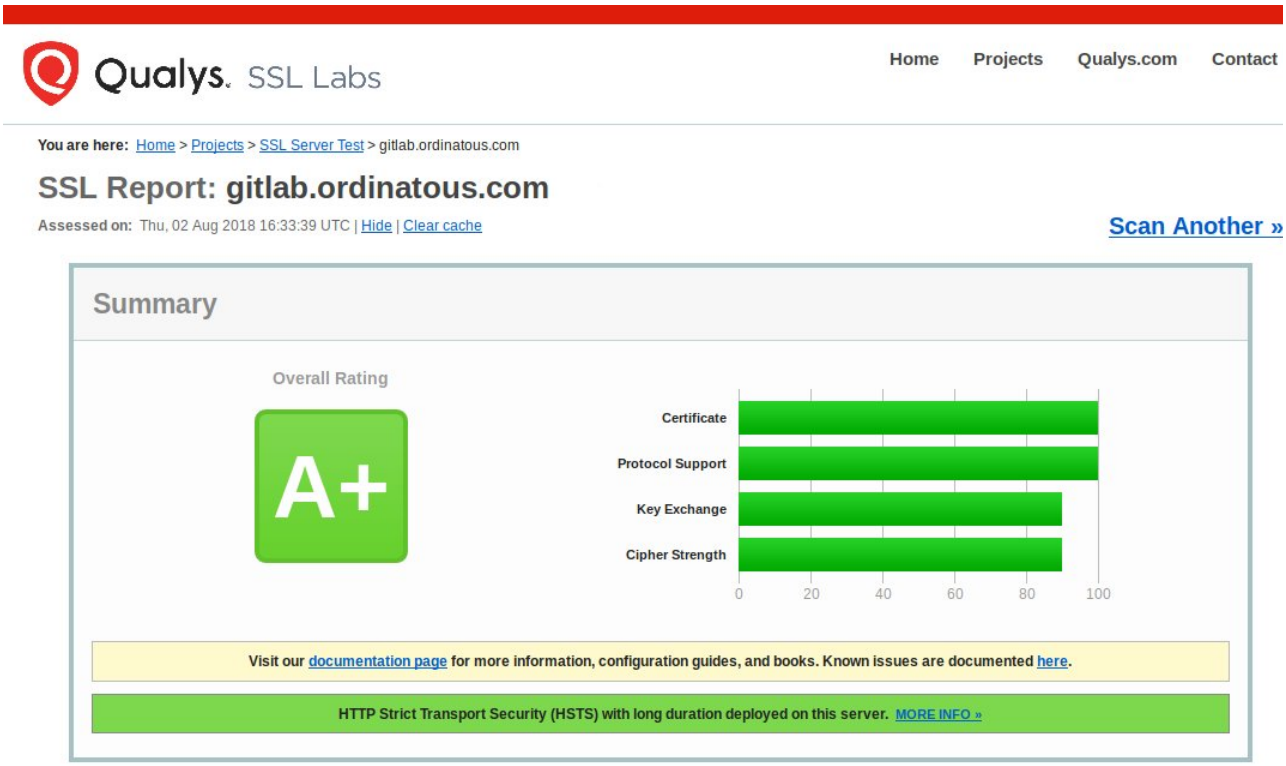
Avant de lancer *nginx* , vérifiez la configuration avec :

```
sudo nginx -t
[sudo] Mot de passe de ordinatous :
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

En cas d'erreur, le test indique précisément qoui, et a quelle ligne, difficile de faire plus simple et efficace.
On pourra tester la configuration du *SSL* (en réalité **TLS**), avec une commande `openSSL`.

```
openssl s_client -showcerts -connect gitlab.ordinatous.com:443
CONNECTED(00000003)
depth=2 O = Digital Signature Trust Co., CN = DST Root CA X3
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
verify return:1
depth=0 CN = gitlab.ordinatous.com
verify return:1
---
Certificate chain
 0 s:/CN=gitlab.ordinatous.com
   i:/C=US/O=Let's Encrypt/CN=Let's Encrypt Authority X3
Sortie tronquée
```

Ou encore sur le service SSLlabs:



test_ssl

Cool un **A+**.
Si vous n'arrivez pas à joindre votre site pour une raison ou un autre, utilisez le service : downforeveryoneorjustme.