Documentation Reverse Proxy

Ce document explique ce qu'est un reverse proxy, comment l'installer, le configurer et l'utiliser pour sécuriser et optimiser la distribution des requêtes web.

1. Présentation

Un reverse proxy est un serveur intermédiaire placé devant un ou plusieurs serveurs web. Il reçoit les requêtes des clients et les redirige vers le serveur approprié, tout en offrant :

- Sécurité : filtrage, protection DDoS, restriction d'accès.
- Performance : mise en cache, compression, répartition de charge (load balancing).
- SSL/TLS : terminaison des connexions chiffrées.

2. Choix de la solution

Les solutions courantes sont :

- NGINX : léger, haute performance, support natif du load balancing et SSL.
- Apache HTTP Server (mod_proxy) : modulable, intégré aux environnements Apache existants.
- HAProxy : spécialisé en load balancing, très haute fiabilité.

3. Installation

3.1. NGINX (Ubuntu/Debian)

- sudo apt update
- sudo apt install nginx

3.2. Apache (Ubuntu/Debian)

- sudo apt update
- sudo apt install apache2
- sudo a2enmod proxy proxy_http proxy_balancer proxy_wstunnel

3.3. HAProxy (Ubuntu/Debian)

- sudo apt update
- sudo apt install haproxy

4. Configuration de base

4.1. Exemple NGINX

```
server {
  listen 80;
   server_name example.com;

   location / {
      proxy_pass http://backend_pool;
     proxy_set_header Host $host;
     proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
   }
```

```
}

upstream backend_pool {
    server 192.168.1.10:80;
    server 192.168.1.11:80;
}
```

4.2. Exemple Apache

```
<VirtualHost *:80>
   ServerName example.com

ProxyPreserveHost On
     ProxyPass / http://backend_pool/
    ProxyPassReverse / http://backend_pool/

    <Proxy backend_pool>
        Require all granted
        BalancerMember http://192.168.1.10:80
        BalancerMember http://192.168.1.11:80
    </Proxy>
</VirtualHost>
```

4.3. Exemple HAProxy

```
global
    log /dev/log local0
    maxconn 4096
```

```
defaults
    log     global
    mode    http
    option  httplog
    timeout connect 5000ms
    timeout client  50000ms
    timeout server  50000ms

frontend http_front
    bind *:80
    default_backend servers_back

backend servers_back
    balance roundrobin
    server web1 192.168.1.10:80 check
    server web2 192.168.1.11:80 check
```

5. SSL/TLS et sécurité

5.1. Terminaison SSL (NGINX)

```
server {
    listen 443 ssl;
    server_name example.com;

    ssl_certificate /etc/ssl/certs/example.crt;
    ssl_certificate_key /etc/ssl/private/example.key;

    location / {
        proxy_pass http://backend_pool;
```

```
    }
}
```

5.2. Headers de sécurité

```
add_header X-Frame-Options "SAMEORIGIN";
add_header X-Content-Type-Options "nosniff";
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";
```

6. Mise en cache et compression

Cache (NGINX) : proxy_cache_path et proxy_cache pour stocker les réponses.

Compression : gzip on; gzip_types text/plain application/json ...;

7. Monitoring et logs

NGINX : voir /var/log/nginx/access.log et error.log.

HAProxy : utiliser la socket UNIX et stats socket pour dashboard.

8. Bonnes pratiques

Garder le logiciel à jour.

Limiter la surface d'exposition (ports, IP).

Utiliser des certificats via Let's Encrypt et renouvellement automatique.

Surveiller les performances et erreurs via un outil (Grafana, Kibana...).