Serveur Web

- Apache2
  - Installer un Reverse Proxy Apache2 sur Debian 11
  - Sécuriser Apache2 sur un serveur
  - Debian 11 utiliser Certbot avec Apache2
  - Certificat SSL Apache2 Debian 11
  - Installer CrowdSec pour Apache sur Debian 11
  - Créer une Autorité de certification
- Nginx
  - Setting up an NGINX Reverse Proxy with SSL
- Nginx Proxy-Manager
  - Securing NGinX Proxy Manager
  - Using NginX Proxy Manager for proper Website Routing
- Wordpress
  - Installer WordPress sur Debian 11
  - WordPress Désactiver wp_xmlrpc
  - WordPress référencement naturel (SEO)
  - WordPress Ajouter un Sitemap
  - WordPress et Google Analytics
  - WordPress Thème enfant
- Caddy
  - Introduction
  - Set Up a CrowdSec Using OPNsense LAPI on Caddy

Apache2

Apache2

Installer un Reverse Proxy Apache2 sur Debian 11

Dans cette procédure je vais vous expliquer comment installer un serveur **reverse proxy** sur Debian 11 avec **Apache2**. Un serveur Reverse Proxy est un serveur qui se situe entre un accès internet et les serveurs internes afin de gérer une mémoire cache des applications.

Prérequis pour installer un Reverse Proxy Apache2 :

- Une machine sous Debian 11
- Un DNS Local ou public
- Un serveur web

Installer un Reverse Proxy Apache2 sur Debian 11 :

Depuis une machine Debian 11 qui ne va servir qu'a ça (recommandation mais pas obligé), nous allons saisir les instructions suivantes :

Mise à jour du système :

```
apt update && apt full-upgrade -y
```

Installation des **Apache2** :

```
apt-get install apache2
```

Afin de bénéficier du mode reverse sur Apache2, il faut activer les module : proxy et proxy_http. Pour activer ces 2 modules saisissez la commande :

```
a2enmod proxy proxy_http
```

Pour que l'activation de ces modules soient pris en compte redémarrer le service d'apache2 :

```
systemctl restart apache2
```

Créer un fichier de configuration Apache :

```
vim /etc/apache2/conf-available/votre-conf.conf

# ou
nano /etc/apache2/conf-available/votre-conf.conf
```

Voici un fichier de configuration d'exemple :

```
<VirtualHost *:80>
    ServerName votre-domaine.fr
    ServerAdmin postmaster@domaine.fr

    ProxyPass / http://127.0.0.1/
    ProxyPassReverse / http://127.0.0.1/
    ProxyRequests Off
</VirtualHost>
```

- ServerName correspond à votre domaine
- ProxyPass et ProxyPassReverse correspondent au serveur de destination.
- ProxyRequests est en off pour des raison de sécurité.

Activer la configuration :

```
a2ensite votre-conf.conf
```

Puis redémarrer apache2 :

```
systemctl restart apache2
```

> Source :
>
> *https://httpd.apache.org/docs/2.4/fr/*

Recommandations :

Je vous recommande de sécuriser ce serveur à l’aide d’un firewall sur vos deux machines, comme UFW qui est facile à prendre en mains ou Iptables. Sur votre machine qui sert de serveur Web, vous pouvez autoriser uniquement le serveur reverse proxy à se connecter à l’aide du port 80 ou 443 (ports part défaut) et n’oubliez pas de garder le port SSH (port 22 par défaut) ouvert si vous l’utilisez. Vous pouvez également utiliser l’outil CrowdSec afin de limiter les tentatives de cyberattaques si votre site web est exposé sur internet.

Je vous recommande aussi d’utiliser faill2ban pour limiter les attaques de force brute sur vos serveurs web Apache2.

Apache2

Sécuriser Apache2 sur un serveur

Dans cette procédure, je vais vous montrer quelques bases pour sécuriser vos serveurs Apache2. Je vous recommande de faire une sauvegarde de votre serveur avant toutes modifications, quelques opérations peuvent être sensibles pour serveur Apache2.

Prérequis pour sécuriser Apache2 sur un serveur :

- Un serveur Apache2
- Avoir fait une sauvegarde de ce serveur

Modifications sur la configuration Apache2 :

Pour sécuriser un serveur Apache2, la première étape à faire sur un serveur Apache2. C’est de cacher la version du service.

Not Found

The requested URL /asdf was not found on this server.

Apache/2.4.7 (Ubuntu) Server at 127.0.0.1 Port 80

Not Found

The requested URL was not found on this server.

Le but va être de passer de la première image à la deuxième. Pour cela il faut modifier la configuration de Apache2 :

```
vim /etc/apache2/conf-enabled/security.conf
```

Remplacer : ServerSignature On
Par : ServerSignature Off

Il est également possible de faire cette étape avec la commande suivante :

```
sed -i 's/ServerSignature On/ServerSignature Off/g' /etc/apache2/conf-enabled/security.conf
```

La seconde étape de sécurisation du serveur Apache2 sera de désactiver la lisibilité des fichiers présents dans les dossiers :

```
vim /etc/apache2/conf-enabled/security.conf
```

Ajouter ceci à la fin du fichier :

```
<Directory /var/www>
□□Options -Indexes
</Directory>
```

Index of /assets/css

Name	Last modified	Size	Description
Parent Directory		-	
bootstrap-grid.css	2019-02-13 14:47	63K	
bootstrap-grid.css.map	2019-02-13 14:47	148K	
bootstrap-grid.min.css	2019-02-13 14:47	47K	

Forbidden

You don't have permission to access this resource.

Maintenant, il faut redémarrer le service Apache2 pour que la configuration soit prise en compte :

```
sudo systemctl restart apache2
```

Ensuite avoir un système à jour permet de limiter les possibles intrusions.

```
sudo apt update && sudo apt full-upgrade -y
```

Il est aussi recommandé d'installer un certificat et de forcer la communication avec le protocole HTTPS afin de chiffrer chaque communication.

Je recommande l'utilisation de firewall comme Iptables ou UFW. Attention à ne pas bloquer les ports que vous utilisez. Comme le 80/443 pour l'utilisation web et le 22 pour l'utilisation du protocole SSH. Si vous n'avez pas changé les ports par défaut.

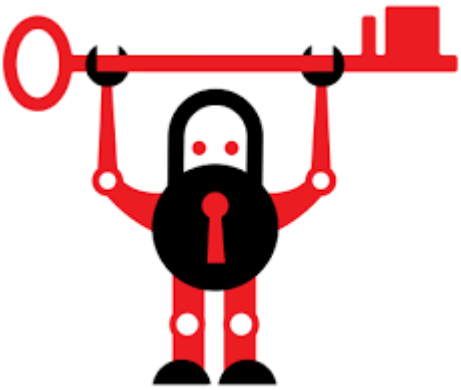
Je conseille aussi d'utiliser une machine en tant que reverse proxy dans le but de limiter les possibles accès à vos données qui sont accessibles depuis les serveurs web que vous possédez. De plus, en cas de Cyber attaque il s'agira du serveur qui contiendra ce reverse proxy qui sera le premier à subir l'attaque. Il vous offrira aussi un cache permettant d'obtenir de meilleur temps de réponses lors des chargements des pages de votre site Web. Attention, il ne faut pas abuser de ce cache et le nettoyer régulièrement sinon il pourrait être source d'intrusion.

Il est également recommandé d'utiliser un parseur de logs (comme CrowdSec) afin de détécter les comportements annormaux : **Installer CrowdSec pour Apache sur Debian 11**

Apache2

Debian 11 utiliser Certbot avec Apache2

Dans cet article, je vais vous montrer comment utiliser Certbot avec Debian 11 pour mettre en place un certificat SSL (Let’s encrypt) pour votre site web. Dans cette procédure, je vais utiliser Apache2 comme serveur web, si vous souhaitez utiliser un autre type de serveur il faudra adapter les commandes. Avoir un certificat SSL sur votre site web vous permet de chiffrer la communication en votre serveur Web et le client, en revanche le certificat SSL sur un site garantie pas que le site est sécurisé.



Prérequis :

- Une machine Debian avec un accès root

Utiliser Certbot sur Debian 11 pour mettre en place un certificat SSL:

Pour utiliser Cerbot sur une machine Debian 11 (avec Apache2), il faut d’abord l’installer :

On va d’abord mettre à jour la liste des paquets :

```
apt update
```

Puis nous allons télécharger Certbot :

```
apt install certbot python3-certbot-apache -y
```

Ensuite vous allez avoir accès à la commande Certbot pour générer les certificats et Certbot s’occupe de mettre en place les certificats :

```
certbot --apache
```

Puis répondez aux questions demandées par Certbot, ensuite redémarrer Apache si vous utilisez Apache ou sinon le service web que vous utilisez :

```
systemctl restart apache2
```

Une fois le service redémarré, le certificat SSL sera installé pour votre site (le virtualhost sera configuré par Certbot).

Si vous souhaitez utiliser Certbot pour un sous-domaine vous pouvez utiliser la commande suivante :

```
certbot --apache --expand -d domaine.com  -d sous.domaine.com
```

> Sources :
>
> *https://certbot.eff.org/*

Apache2

Certificat SSL Apache2 Debian 11

Le certificat SSL est important pour un serveur Web car il va crypté chaque communication avec le serveur Web et le Client. Pour avoir le HTTPS sur son serveur Web Apache2 qui se trouve sur la distribution Debian 11, il faut installer un certificat SSL. Dans cette procédure, je vais le réaliser à l'aide d'un VPS, un DNS et un certficat SSL de l'hébergeur 1&1 Ionos.

Prérequis pour un certificat SSL dans Apache2:

- Un VPS sous Debian 11
- Un DNS
- Un Certificat SSL (certficat.cert, certificat_intermédiaire.cert, cle_privee.key)
- Avoir installer Apache2

Installer un certificat SSL dans Apache2 :

Pour installer le certificat, utiliser un protocole de transfert de fichiers pour envoyer sur votre serveur les fichiers.

Nous allons créer plusieurs répertoire, chacun devra contenir un fichier.

```
mkdir -p /etc/ssl/cles
mkdir -p /etc/ssl/certificats
```

Dans le dossier « cles » on va y mettre le fichier de clé privées (.key)

Dans le dossier « certificats », il faut mettre les fichiers certificats (.cert), le certificat et le certificat intermédiaire.

Puis rendez-vous dans le fichier de configuration du VirtualHost de votre site à l'emplacement suivant /etc/apache2/sites-available/

Modifier votre VirtualHost :

```
<VirtualHost *:80>
        ServerName votre-domaine.fr

        ServerAdmin postmaster@votre-domaine.fr
        DocumentRoot /var/www/votre-domaine

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf

        Redirect permanent / https://votre-domaine.fr
</VirtualHost>

<VirtualHost *:443>
        ServerName votre-domaine.fr
        ServerAdmin postmaster@votre-domaine.fr
        DocumentRoot /var/www/votre-domaine

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
        SSLEngine on
        SSLCertificateFile /etc/ssl/certificats/votre-domaine.fr_ssl_certificate.cer
        SSLCertificateKeyFile /etc/ssl/cles/votre-domaine.fr_private_key.key
        SSLCertificateChainFile /etc/ssl/certificats/votre-domaine.fr_ssl_certificate_INTERMEDIA
</VirtualHost>
```

Une fois la configuration du VirtualHost faite et enregistrer, vous pouvez vérifier que votre configuration apache2 est correct :

Si vous n'aviez pas de VirtualHost avant, n'oubliez pas de l'activée ! Avec la commande a2ensite votre-domaine.fr

```
apachectl configtest
```

Si le message suivant vous est retourner, c'est que votre configuration est correct sinon il faut regarder votre fichier de configuration, il doit y avoir une erreur.

```
root@localhost:/etc/apache2/sites-available# apachectl configtest
Syntax OK
```

Une fois que vous êtes sûr que votre configuration est correct, alors vous pouvez redémarrer le service Apache2.

```
systemctl restart apache2
```

Après avoir fait cette opération si vous essayé de vous connecter sur votre domaine depuis un navigateur. Vous aller être automatiquement rediriger sur le protocole HTTPS, même si vous précisé que vous utilisé le protocole HTTP.

> Sources :
>
> *https://wiki.debian.org/Apache*

Apache2

Installer CrowdSec pour Apache sur Debian 11

Dans cette procédure, je vais vous montrer comment installer CrowdSec sur Debian 11 pour protéger son site Web Apache. CrowdSec est un outil qui va permettre de bloquer les attaques qu'il détecte sur le serveur, nous allons utilisé un serveur Debian 11 dans notre démonstration. CrowdSec fonctionne avec des modules appelés des Bouncers. Les bouncers permettent de bloqué les trafics réseau qui ne sont pas détecter comme « normaux ». Ils peuvent soit bloqué complètement la requête ou afficher à Captcha dans le but de vérifier qu'il ne s'agit pas d'un robot.

Prérequis :

- Une machine Debian avec Apache2 d'installé.
- Avoir Composer d'installer
- Avoir les permissions root ou avoir sudo

Installer CrowdSec pour Apache 2 sur Debian 11 :

Avant de commencer l'installation de CrowdSec, commencer par mettre à jour la liste des paquets votre machine Debian 11

```
sudo apt update
```

Ensuite télécharger le CrowdSec sur votre machine Debian :

```
sudo apt install -y crowdsec
```

Après avoir installer CrowdSec sur votre machine, la CLI de CrowdSec (CSCLI) vous permet de voir les services pouvant être protéger par CrowdSec présents sur votre machine.

```
cscli collections list
```

```
root@localhost:~# cscli collections list
--------------------------------------------------------------------------------------------------------------------
 NAME                                  STATUS    VERSION  LOCAL PATH
--------------------------------------------------------------------------------------------------------------------
 crowdsecurity/modsecurity             ✔ enabled  0.1      /etc/crowdsec/collections/modsecurity.yaml
 crowdsecurity/vsftpd                  ✔ enabled  0.1      /etc/crowdsec/collections/vsftpd.yaml
 crowdsecurity/sshd                    ✔ enabled  0.1      /etc/crowdsec/collections/sshd.yaml
 crowdsecurity/base-http-scenarios     ✔ enabled  0.4      /etc/crowdsec/collections/base-http-scenarios.yaml
 crowdsecurity/mysql                   ✔ enabled  0.1      /etc/crowdsec/collections/mysql.yaml
 crowdsecurity/linux                   ✔ enabled  0.2      /etc/crowdsec/collections/linux.yaml
 crowdsecurity/postfix                 ✔ enabled  0.2      /etc/crowdsec/collections/postfix.yaml
 crowdsecurity/whitelist-good-actors   ✔ enabled  0.1      /etc/crowdsec/collections/whitelist-good-actors.yaml
 crowdsecurity/apache2                 ✔ enabled  0.1      /etc/crowdsec/collections/apache2.yaml
 crowdsecurity/dovecot                 ✔ enabled  0.1      /etc/crowdsec/collections/dovecot.yaml
 crowdsecurity/nginx                   ✔ enabled  0.1      /etc/crowdsec/collections/nginx.yaml
 crowdsecurity/wordpress               ✔ enabled  0.1      /etc/crowdsec/collections/wordpress.yaml
 crowdsecurity/iptables                ✔ enabled  0.1      /etc/crowdsec/collections/iptables.yaml
 crowdsecurity/naxsi                   ✔ enabled  0.1      /etc/crowdsec/collections/naxsi.yaml
--------------------------------------------------------------------------------------------------------------------
```

Vous pouvez également lister les bouncers installés et disponibles sur votre machine :

```
cscli bouncers list
```

Si vous exécutez cette commande, vous allez voir un tableau vide. C’est normal car nous n’avons pas encore installer de bouncer.

Installation du Bouncer Apache 2 sur CrowdSec (Debian 11) :

Avant de commencer l’installation du Bouncer, assurez-vous d’avoir composer d’installé. Si vous ne l’avez pas installer, voici une procédure pour l’installer :

Installer Composer sur Debian 11

Nous allons installer git sur la machine pour pouvoir cloner le Repo du bouncer :

```
sudo apt install git -y
```

Puis nous clonons le Repo GitHub :

```
git clone https://github.com/crowdsecurity/cs-php-bouncer.git
```

On se rend dans le dossier du projet :

```
cd cs-php-bouncer/
```

Enfin on lance l'installation avec un compte utilisateur (pas avec le compte root ni avec sudo mais le compte utilisateur doit être dans le groupe 'sudo')

```
./install.sh --apache
```

Puis on donne les droit en tant que propriétaire www-data (apache) sur le répertoire de CrowdSec

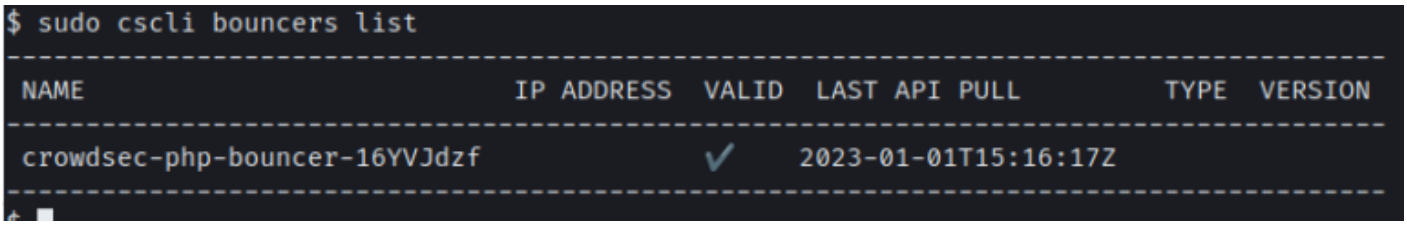
```
sudo chown www-data /usr/local/php/crowdsec/
```

Puis on redémarre le service de Apache

```
sudo systemctl reload apache2
```

Enfin le Bouncer de Apache va apparaître dans la liste des bouncers :

```
sudo cscli bouncers list
```



Ensuite on va afficher un Captcha pour les utilisateurs qui consulterons notre site web avec un mauvais User-agent (Généralement les outils de pentest utilisent des User-Agent différent des navigateurs Web) et les crawler non static.

On va modifier le fichier de configuration :

```
sudo nano /etc/crowdsec/profiles.yaml
```

```
# Mauvais User-Agent + Crawler
name: crawler_captcha_remediation
filters:
 - Alert.Remediation == true && Alert.GetScenario() in ["crowdsecurity/http-crawl-non_statics",
decisions:
 - type: captcha
   duration: 4h
on_success: break
---
name: default_ip_remediation
filters:
 - Alert.Remediation == true && Alert.GetScope() == "Ip"
decisions:
 - type: ban
   duration: 4h
on_success: break
```

```
---
```

Puis on redémarre CrowdSec pour qu'il utilise la nouvelle configuration :

```
sudo systemctl restart crowdsec
```

Il est possible d'afficher toutes les actions que CrowdSec aura pris avec cette commande :

```
sudo cscli decisions list
```

C'est également possible de débanir une adresse IP :

```
sudo cscli decisions delete --ip adresse-ip-a-deban
```

La CLI de CrowdSec est pratique mais CrowSec propose également une interface graphique Web avec Docker.

> Sources :
>
> *https://doc.crowdsec.net/*

Apache2

Créer une Autorité de certification

Dans cette procédure je vais vous expliquer comment créer une autorité de certification et un certificat hôte pour Apache2. Pour réaliser l'autorité de certification nous allons utiliser Openssl et une machine Debian.

Prérequis :

- Une machine sous Debian
- Openssl

Créer une Autorité de certification :

D'abord, nous allons créer un répertoire pour notre autorité de certification :

```
mkdir -p /etc/ssl/certificats/CA
```

Ensuite nous allons créer la clé privée de l'autorité de certification :

```
openssl genrsa -des3 -out /etc/ssl/certificats/CA/CA.key 2048
```

Saisir une passphrase qui sera utilisée pour signer les certificats (Donc à ne pas perdre).

Puis nous allons générer le certificat *root* (racine) de l'autorité de certification au format .pem :

```
 openssl req -x509 -new -nodes -key /etc/ssl/certificats/CA/CA.key -sha256 -days 10000 -out /etc/ssl/certificats/CA/CA.pem
```

Une liste de questions va vous être demandée.

Ensuite nous allons générer le certificat root (racine) au format .crt :

```
openssl x509 -in /etc/ssl/certificats/CA/CA.pem -inform PEM -out
/etc/ssl/certificats/CA/CA.crt
```

Installer l’autorité de certification sur une machine :

Nous avons créer 3 fichiers dans le répertoire /etc/ssl/certificats

Intallation version web :

Le fichier CA.crt est a importé dans votre navigateur Web :

- Firefox => saisir dans l’url : about:preferences#privacy => Se rendre dans **Certificats** => **Afficher les certificats** => **Importer** => Choisir le fichier .crt
- Chrome => saisir dans l’url : chrome://settings/security => Se rendre dans **Gérer les certificats** => **Importer** => Choisir le fichier .crt

Installation au niveau de l’OS :

Windows :

Récupérer le fichier CA.crt et double cliquer dessus => Choisir l’emplacement : « Autorité de certification racine de confiance »

Ou en ligne de commande :

certutil.exe -addstore root CA.crt

Pour Firefox, il faut autoriser l’utilisation des autorités de certifications de confiances de Windows. Donc il faut créer un fichier dans le répertoire suivant C:\Program Files (x86)\Mozilla Firefox\Defaults\Pref\defaults\pref\ ou C:\Program Files\Mozilla Firefox\Defaults\Pref\defaults\pref\ :

Créer un fichier enableroot.js
Ajoutez y le contenu suivant :

```
pref("security.enterprise_roots.enabled", true);
```

Linux :

D’abord, on va copier le fichier CA.crt dans le répertoire des autorités (**/usr/local/share/ca-certificates/**) :

```
cp CA.crt /usr/local/share/ca-certificates/
```

Ensuite mettre à jour les autorités :

```
update-ca-certificates
```

Créer un certificat hôte :

D’abord nous allons créer un répertoire pour notre hôte :

```
mkdir -p /etc/ssl/certificats/hote
```

Ensuite nous allons créer la clé privée de l’hôte :

```
openssl genrsa -out /etc/ssl/certificats/hote/hote.key 2048
```

Puis on génère la demande de signature de certificat (fichier au format .csr) :

```
openssl req -new -key /etc/ssl/certificats/hote/hote.key -out
/etc/ssl/certificats/hote/hote.csr
```

Dans les questions qui sont demandées, il faut mettre le DNS du serveur pour le Common Name.

Signer le certificat hôte par l’autorité de certification :

D’abord on va créer un fichier de configuration :

```
nano /etc/ssl/certificats/hote/hote.ext
```

Avec le contenu suivant :

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = hote
```

Pour signer le certificat de l'hôte par l'autorité de certification précédemment créée nous allons exécuter la commande suivante :

```
openssl x509 -req -in /etc/ssl/certificats/hote/hote.csr -CA /etc/ssl/certificats/CA/CA.pem -CAk
```

Enfin on renseigne la passphrase de l'autorité de certification.

Intégré le certificat signé avec Apache2 :

Pour intégré le certificat dans Apache2, on doit éditer le Virtual Host qui est utilisé dans le répertoire **/etc/apache2/sites-available** :

```
nano /etc/apache2/sites-available/000-default.conf
```

Voici un exemple de configuration :

```
<VirtualHost *:80>
        ServerName hote

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        Redirect permanent / https://hote/
</VirtualHost>
<VirtualHost *:443>
        ServerName hote

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        SSLEngine on
        SSLCertificateFile /etc/ssl/certificats/hote/hote.crt
        SSLCertificateKeyFile /etc/ssl/certificats/hote/hote.key
</VirtualHost>
```

Cette configuration permet d'automatiquement rediriger les connexions http en https et d'utiliser les fichiers du certificat et de la clé de l'hôte.

Ensuite on va activer le SSL sur Apache2 :

```
a2enmod ssl
```

Puis on redémarrer le service Apache2 :

```
systemctl restart apache2
```

Nginx

Setting up an NGINX Reverse Proxy with SSL

A reverse proxy is a function of a web server that allows it to forward requests onto other web servers. Typically they are set up in combination with a wildcard SSL certificate, allowing each subdomain to go to a different server. Apache2 and NGINX can both function as reverse proxies, but NGINX is much easier to set up!

Installing NGINX

Install using apt

First, install nginx using apt:

```
sudo apt install nginx
```

Testing with telnet

Telnet is great for testing TCP ports. NGINX listens on TCP port 80 by default, so we can test it with telnet:

```
telnet 127.0.0.1 80
```

If the server is responding correctly, you should see an output similar to below:

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

Configuring NGINX

> NGINX configuration files are stored in /etc/nginx/ - the site configuration files are stored in sites-available/, then these are enabled by creating a symbolic link to them in sites-enabled/

Disabling the default site configuration file

Now that we know the server is running, we are going to disable the default NGINX configuration file. This is so that we can create a new one that is more focused around the reverse proxy abilities of NGINX.

To remove the default configuration file, run this command:

```
sudo rm /etc/nginx/sites-enabled/default
```

Creating a new site configuration file

We will now create a new site configuration file, using our domain name as the file name:

```
sudo nano /etc/nginx/sites-available/YOUR.DOMAIN
```

Here is an example template, which retains NGINX's default site configuration for the root domain and does reverse proxy for a subdomain to a Proxmox server:

```
# We are defining our Let's Encrypt certificate here
# If your NGINX server will serve multiple domains, you will instead need to place it within
the 443 definition
ssl_certificate /etc/letsencrypt/live/YOUR.DOMAIN/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/YOUR.DOMAIN/privkey.pem;

# Required when reverse proxying to a Proxmox server
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

# This will forward all HTTP requests to HTTPS
server {
listen 80;
listen [::]:80;
server_name *.YOUR.DOMAIN;
rewrite ^ https://$host$request_uri? permanent;
}

# This will serve NGINX's default web page, via HTTPS
server {
listen 443 ssl;
listen [::]:443 ssl;
server_name YOUR.DOMAIN;

# Define the web root
root /var/www/html;
```

```
	# Add index.php to the list if you are using PHP
	index index.html index.htm index.nginx-debian.html;

	location / {
		# First attempt to serve request as file, then
		# as directory, then fall back to displaying a 404.
		try_files $uri $uri/ =404;
	}
}

# This subdomain will reverse proxy requests to a Proxmox web interface
server {
	listen 443 ssl;
	listen [::]:443 ssl;
	server_name proxmox.YOUR.DOMAIN;

        location / {
                proxy_pass              https://YOUR.PROXMOX.IP.ADDRESS:8006;
                proxy_set_header        Upgrade $http_upgrade;
                proxy_set_header        Connection "upgrade";
        }
}
```

Once modified to suit your setup, use Control-S to save and Control-C to quit nano.

Enabling the new configuration file

We now have a new configuration file, and we are going to enable it by creating a symbolic link:

```
sudo ln -s /etc/nginx/sites-available/YOUR.DOMAIN /etc/nginx/sites-enabled/YOUR.DOMAIN
```

Testing the configuration

We can now run nginx -t in order to test the new configuration file:

```
sudo nginx -t
```

Assuming all is well, you should see an output similar to below:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Restarting NGINX

Once we've confirmed that the configuration file tests successfully, we can apply it by restarting the NGINX service:

```
sudo systemctl restart nginx
```

Testing port 443 with telnet

We already know that NGINX was listening on TCP port 80, but since we have just enabled SSL, we want to make sure that it's also listening on port 443:

```
telnet 127.0.0.1 443
```

The output should be similar to before:

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

Nginx Proxy-Manager

Securing NGinX Proxy Manager

Installing NGinX Proxy Manager

To install NPM you need to install docker and docker-compose, and create a new folder on the server you want to run it in. Next, you'll create two files inside that folder:

- config.json
- docker-compose.yml

Inside the config.json file, you'll put the following:

```
{
  "database": {
    "engine": "mysql",
    "host": "db",
    "name": "npm",
    "user": "<your desired username>",
    "password": "<a strong password>",
    "port": 3306
  }
}
```

And inside the docker-compose.yml file you'll put:

```
version: '3'
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    volumes:
      - ./config.json:/app/config/production.json
      - ./data:/data
```

```
      - ./letsencrypt:/etc/letsencrypt
  db:
    image: 'jc21/mariadb-aria:10.4'
    environment:
      MYSQL_ROOT_PASSWORD: 'npm'
      MYSQL_DATABASE: 'npm'
      MYSQL_USER: '<username from config.json>'
      MYSQL_PASSWORD: '<strong password from config.json>'
    volumes:
      - ./data/mysql:/var/lib/mysql
```

Make sure to replace the items with < and > around it in each file, and that the username and passwords in each file match.

Now run the command:

`docker-compose up -d`

Give it a minute to pull down everything, and get started, and then in your browser go to the IP address of your server. You should get a Congratulations screen.

if you go tot he IP address at port 81 (http://192.168.1.x:81), you'll be prompted to login to NPM.

Default credentials are:

username: admin@example.com
passwrod: changeme

Make sure to update the email and password, from the default values, then log out, and back in usign the new values you entered.

Now, you're ready to start proxying traffic.

Securing NGinX Proxy Manger Admin Console

The simples and most direct way is to secure NPM to itself. Yep, you just make a loop so that when you ask for a specific URL that you'll have created an A Record for, you get your NGinX Proxy Manager install will proxy the traffic to it's port 81 admin console.

Let's add a new Host entry, and on the dtails page enter the URL you want to use to access the admin console.

In my case, I called it "manage" and created an A Record to point to my public IP address, which is port forwarded 80, 443 to my NPM server.

In my Details tab, I'll enter "manage.example.com" (replace example.com with your domain of course).

Next, enter the IP address of your docker0 interface. You can find this with either:

```
ifconfig
```

or

```
ip addr show docker0
```

Next, enter 81 for the port number.

Turn on the "Block Common Exploits" option, and Save.

We Save now so we can test it and make sure we are routed properly to our Admin login page. This is still unencrypted, so don't log in, but make suer you get to the Admin login page by visiting your URL.

If you get to the page successfully, you can go back to NPM via the IP nad Port, and click the 3 dot icon at the right end of your 'manage' row. Select 'Edit' andn move to the 'SSL' tab.

Choose "Request a New Certificate" from the first dropdown.

Turn on "Force SSL".

Fill in your email address. LetsEnctrypt uses this to let you know if your Certificates have issues or will expire soon and haven't been renewed.

Select to 'Accept the Terms of Service'.

Click 'Save'.

Now, attempt again, to reach your URL. You should be routed to an SSL encrypted site, and you can now login to your Admin console.

What About Keeping the Rest of the Internet Out?

Yep, you just created a publicly accessible URL for your NPM admin console. Not to fear. You can still secure it further. Go to the "Access Lists" tab, and create a new Access List.

Give your new List a name, and then move to the Authorization tab. Enter as many emails and passwords for users you want to have access to the site. If you want to restrict to http basic auth, then save, and close your browser (Note: you may have to clear your cache). Then, re-open, and visit your site, and you should get a prompt for Credentials before you get to the main login screen of NPM itself.

Want more than just basic authentication?

You can also add public IP addresses that will be allowed to access the site. Edit your Access List, and move to the Access tab. Enter yoru public IP address, then try to access the site from a machine not on your LAN, and you'll find you won't be able to.

Now, let's say you want to access the site with one or the other, Username and passsword, or Public IP. Then on the Details tab of your Access List, and enable the option for 'Satisfy Any'. This makes it so either User or IP will be allowed. When this is disabled, then you must have both User credentials and be on an allowed Public IP.

Conclusion

You have amazing open source tools at your fingertips, and making them more secure is highly recommended. Please use the tools and capabilities to run securely.

Using NginX Proxy Manager for proper Website Routing

What's NginX?

NginX (pronounced Engine-X) is a web-server, and reverse proxy. Basically, it can serve up web pages, and can provide a proxy service for incoming web requests. It all sounds a bit generic, but that's because it is. NginX can serve web pages, but can also direct requests for Web pages, Web Services, and Web Applications to the right place. Essentially, it's a hub and router. It receives a request for any number of web services, and routes the requests to the proper location.

NginX in and of itself, if configured through the terminal in configuration files. It's not pretty, it's not overly difficult, but you definitely have to have a good feel for all of the options that can be set.

NginX Proxy Manager is a nice Graphical User Interface (GUI) for the user to utilize vs. having to edit and create a bunch of configurations.

Workflow of NginX Proxy Manager

When you want to route a user to a specific web page or site, NginX is a great tool for the job. NginX will listen on your server, and once it receives a request, will route the request to the appropriate service, server, page, or application.

Clarification of the term "Router"

I need to be clear about something here. NginX is not an application router. Many applications use routers to move you from page, or portion of a page or application, to another. These are application level routers. Also, NginX is not a hardware router, like the one on your home network that routes all of your network traffic to various machines, smart devices, etc.

NginX is a Proxy Router. It acts as a proxy for the requested web page or site, and forwards that request on to the appropriate site on your server, then returns the response information to the browser.

Workflow Continued

Requests for websites come into a server on a standard port (generaly 80 or 443). When those requests are received, NginX will parse the request by name, and look through it's configuration files to see if any of them match for the request being made.

If I request `fixitdelrio.com`, NginX will look for a configuration file that tells it what to do with requests for that site. If it finds a match it will then use the other information in that confiuration to push that request along to the appropriate server or service.

The configuration might tell NginX, "Hey, when you see `fixitdelrio.com`, send it to the IP 10.20.30.40 please." So, since we asked so nicely, NginX does as requested.

For another site being run on the same server, like `opensourceisawesome.com`, NginX will send the request along to the same IP, but a different port. While we make the request on the standard port 80, NginX knows that really that site is running on port 24356, and has in it's confguration file to push our request along to 10.20.30.40:24356. Thus, we ask for `opensourceisawesome.com`, and don't have to know it's running on a special port.

What about SSL and Encryption?

NginX can also deal with SSL and Encryption, and can be quite helpful with it as well.

If I want to run `lubbocklug.org` on https instead of http, I can use NginX to help me do that. I can use NginX-Proxy-Manager to tell the request for http://lubbocklug.org to always force the requestor over to https://lubbocklug.org, thus they never go to my site without encryption.

This is huge in today's world of unethical hackers. Protecting our users is one of the most important steps we can take as self-hosters.

Okay, I'm tired of the Intro...tell me how to do it.

Information sourced from https://nginxproxymanager.com/setup/

First, you want to install Docker. Docker CE (Community Edition) is a wonderful tool that uses a very lightweight virtualisation engine to run applications, web sites, and services. You can hose many containers (a virtualized application server) on a single Docker install.

Once you have Docker installed, you will want to install NginX Proxy Manager. This part is fairly straight-forward, so let's look at how it's done.

docker-compose.yml

Next, we need a docker-copose.yml (pronounced yamuhl) file. This file tells docker what images to pull, what containers to start, what to call them, how they connect to each other if there is more than one, and all kinds of other information. It's, again, pretty straight-forward, so let's jump into it.

Still in our `nginx_proxy_manager` folder, we now want to create a file called `docker-compose.yml`. So enter the command

`nano docker-compose.yml`

Copy the code below using CTRL+C (Win, Linux, Unix) or CMD+C (MacOS).

```
version: '3.8'
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
```

Now paste the text into the blank text editor window. Use CTRL+Shift+V for Linux, Unix, Win; and usse CMD+V for MacOS.

Again, we need to edit come values in this file. We want a couple of these values to match the values we changed in our config.json. Using your arrow keys move down to the section titled `# environment:`. Under that section you need to either remove the `#` symbol in front of the second line, or remove this entire section (3 lines).

I suggest if you are not going to use IPv6, then make that section look like this.

```
environment:
      # Uncomment this if IPv6 is not enabled on your host
      DISABLE_IPV6: 'true'
```

If you intend to use IPv6, then change `true` to `false`, and ensure it's still surrounded by single quotes '.

Now use your arrow keys to move down to the section titled `db:`. Below it, we want to chagne three values.

First let's change the value for MYSQL_ROOT_PASSWORD. We, again, want to make this a strong, but different password from the config file earlier.

After that, change the MYSQL_USER value to match what we entered in our config.json file for "user", and our MYSQL_PASSWORD value to match what we entered in our config.json file for "password".

Once those changes are made, save the file with CTRL+O, then press Enter / Return. Next press CTRL+X to exit the nano editor.

Start the NginX Proxy Manager

Finally, we will use our docker-compose.yml file to fetch the docker images, and start our containers (yep, there are 2 conttainers - 1 for NginX Proxy Manager, and 1 for the MySql database for configs).

In the same terminal window, enter the command:

```
docker-compose up -d
```

if your user is not part of the `docker ` group, you may have to use

```
sudo docker-compose up -d
```

Then, enter your sudo password when prompted.

If all goes well, you should be able to browse to your server URL or IP address on port 81 to see the NginX-Proxy-Manager admin portal.

something like `http://opensourceisawesome.com:81` <– of course using your own domain or IP.

If you see the admin portal, congratulations! You've got it setup! Now NginX is listening on port 80 and port 443 for web-requests.

The next part is setting up various sites for NginX to proxy.

Proxying Site Traffic with NGinX Proxy Manager

Now that NGinX Proxy Manager is up and running, let's setup a site. Click on 'Proxy Hosts' on the dashboard. The card will likely have a 0, and the view will be empty, or should, so we need to add a new host.

Dashboard View

Now click on the 'Add Proxy Host' button on the upper right of the Proxy Hosts view.

You should see a modal (pop-up) window like the one below.

Add Proxy Host Modal Window

Enter the domain name you want NGinX to listen for in the "Domain Name" field. Domain names should be entered wtihout http or https on the front. so only enter something like `billybobsbassboatsandboots.com` or if you are listening for a subdomain `inventory.billybobsbassboatsandboots.com`

Next, enter the hostname or IP address of the server where the site you entered in the previous step is running. Finally, enter the port number on which that site is listening / hosting it's traffic. If you're using Docker to host these sites, then you can see any port mapping using the `docker ps` or `sudo docker ps` command.

Click 'Save'.

If all goes well, the modal (pop-up) window will close, and you should see an entry in your Proxy Hosts view.

Now you can click on the domain name to have it open in a new tab. If everything is setup properly, you should see your web site.

But what about SSL?

SSL is absolutely an option, and pretty easy to get setup with NginX Proxy Manager. You do have to make sure that you've setup your domain to be reached on port 80. Don't get confused. The port you entered in the last step above, does not need to be 80, but the Domain Name you entered should not have a port added on the end of it...that's all it means.

Essentially, you need to be able to get to `billybobsbassboatandboots.com` without having to add a specific port number. So we don't want to have to do `billybobsbassboatandboots.com:11232` or anything.

Once you are sure that your site comes up on port 80, you'll want to click on the three vertical dot icon at the right end of the line with your domain on it.

Click the 3-dot Menu Option

Select 'Edit' from the menu that is shown, and we'll edit our NginX entry. In the modal (pop-up) window, we want to move to the second tab "Custom Locations". In this section we just want to re-type the same domain name we entered on the first tab, but in the 'Location' field.

Next, click the drop-down menu under 'Scheme', and select "https". Now enter the IP or Hostname address of the server we are proxying the traffic to (usually the same as what we entered on the first tab as well. Finally, enter the port you mapped to 443 in the Port field.

Now we want to move to the third tab, "SSL". Here you want to click where it says "None", and select "Request a new SSL Certificate". If you want to force users to always go to the secure version of your site (which you almost always do), turn on the switch next to the "Force SSL" option.

Now enter your email address into the email field, and turn on the option next to "I agree to the LetsEncrypt Terms of Service".

Options to Request a LetsEncrypt SSL Certificate for your site.

Now click "Save". It may take a minute or so, but if you are returned to the Proxy Hosts view, and no errors are displayed, then your site should now be available on https.

Conclusion

This process of proxying traffic through a single endpoint, is useful for controlling not only the traffic to and from your home or server, but also for allowing you to run multiple web-sites / hosts on a single server install.

Repeat the above steps for each site you are hosting, and over time you'll get a full list of sites being proxied by NginX. You can add custom options as you become more familiar with NginX right inside the Edit modal as well (tab 4).

Wordpress

Wordpress

Installer WordPress sur Debian 11

Dans cette procédure, je vais vous expliquer pas à pas comment installer WordPress sur une machine Debian 11. Après avoir suivi cette procédure vous aurez une machine Debian avec WordPress d'installer. En cas de soucis, je vous conseil de consulter le site officiel :

https://wordpress.com/

Prérequis :

- Une machine Debian 11
- Un serveur Lamp
- Un accès root sur la machine

Installation de WordPress sur Debian 11

WordPress est un CMS qui utilise une base de données pour stocker tous les articles, utilisateurs et autres... Pour notre utilisation nous allons installer Mariadb, qui est très similaire avec Mysql.

MariaDB :

Connexion sur MariaDb :

```
mysql -u root -p
```

Création d’une nouvelle base de données :

```
CREATE DATABASE wordpress_db;
```

Création d’un nouvel utilisateur MariaDB et attribution des droits sur la nouvelle base de données :

```
GRANT ALL ON wordpress_db.* TO 'wordpress_user'@'localhost' IDENTIFIED BY 'password';
```

Appliquer les privilèges :

```
FLUSH PRIVILEGES;
```

Quitter MariaDB :

```
EXIT;
```

Se diriger dans le répertoire /var/www/html :

```
cd /var/www/html/
```

Installer WordPress sur Debian 11 :

Télécharger WordPress :

```
curl -O https://wordpress.org/latest.tar.gz
```

Décompression de l’archive de WordPress

```
tar -xvf latest.tar.gz
```

Suppression de l’archive :

```
rm latest.tar.gz
```

Attribution des droits au serveur Web :

```
chown -R www-data:www-data /var/www/html/wordpress
```

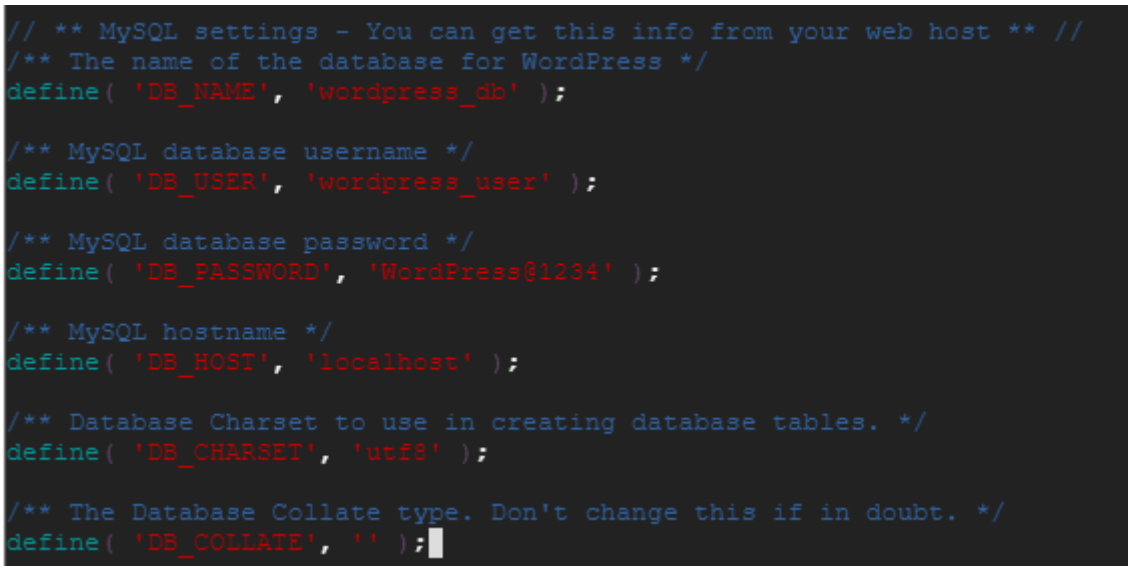
Rendez-vous dans le dossier WordPress :

```
cd wordpress
```

Copie du fichier de configuration :

```
cp wp-config-sample.php wp-config.php
```

Édition du fichier de configuration :



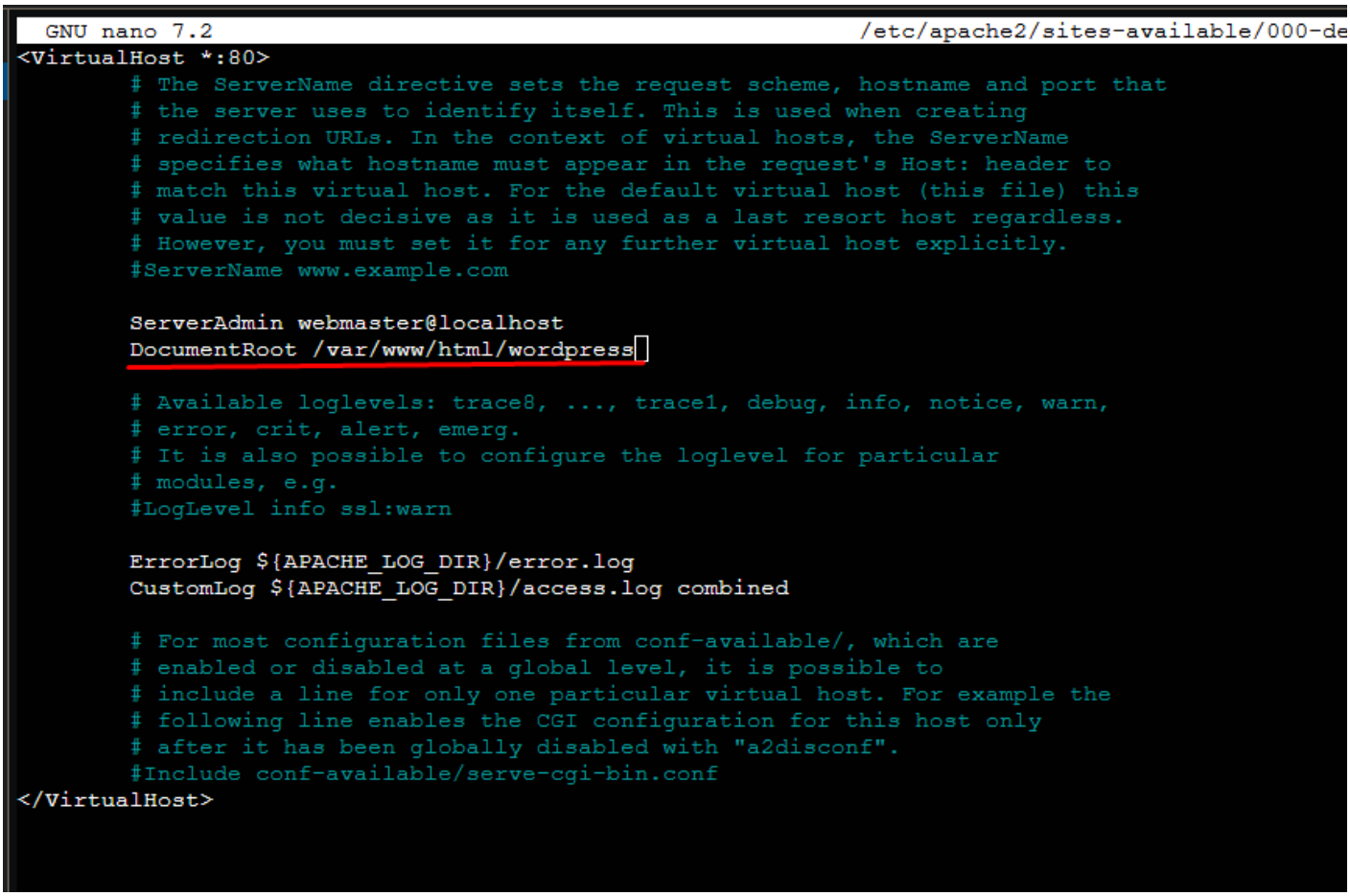
Suppression du fichier readme.html

```
rm /var/www/html/wordpress/readme.html
```

Installation de modules php pour WordPress :

```
apt install php-gd php-intl php-mbstring php-imagick
```

Aller modifier le fichier de config Apache et bouger la racine d'apache dans le dossier wordpress



Puis redémarrer Apache :

```
systemctl restart apache2
```

Depuis votre navigateur vous devriez accéder à la configuration de WordPress :

```
http://votreip/wordpress
```

Avant la publication de WordPress sur le Web, il est conseillé de faire quelques opérations en plus d'un serveur Web. La première chose à faire est d'activer une extension qui empêchera les tentatives d'attaque par brute force. La seconde sera de modifier l'accès admin par un autre emplacement, des extensions permettent de le faire. La troisième sera de ne pas avoir de comptes qui contiennent « admin » ou des mots de passe faibles.

Je déconseille l'hébergement d'un site WordPress sur votre réseau, le moindre oublie de sécurité peut laisser la possibilité à un attaquant de contrôler votre machine / réseau.

Wordpress

WordPress Désactiver wp_xmlrpc

Le XML-RPC permet à WordPress de pouvoir communiquer avec d'autres applications, le soucis de XML-RPC est qu'il est également utiliser pour les attaques sur les sites WordPress. Dans cette procédures, je vais vous montrer comment désactiver wp_xmlrpc dans WordPress. Si wp_xmlrpc n'est pas utilisé, il est impératif à désactiver car il représente une menace de sécurité.

Prérequis :

- Avoir un site WordPress
- Avoir un accès de modification sur les fichiers de WordPress

Désactiver wp_xmlrpc dans WordPress avec Apache :

Pour désactiver wp_xmlrpc dans WordPress, il faut interagir avec le fichier .htaccess qui se trouve à la racine de votre site, alors avant de modifier quoi que ce soit faites une copie de se fichier avant de le modifier. En cas de mauvaise manipulation vous aurez juste à restaurer le fichier de base pour revenir à la normale.

Commande pour copier le fichier :

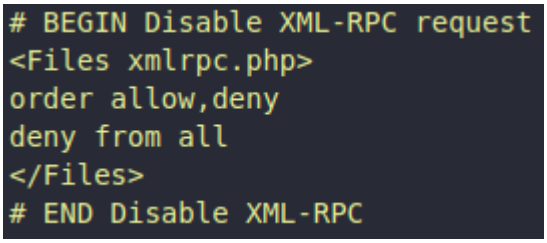
```
cp .htaccess .htaccess.sauveguarde
```

Ensuite ouvrer le fichier .htaccess avec un éditeur de texte

```
nano .htaccess
```

Puis coller cette configuration à la fin de votre fichier :

```
# BEGIN Disable XML-RPC request
<Files xmlrpc.php>
   order allow,deny
   deny from all
</Files>
# END Disable XML-RPC
```



Une fois que vous avez changé cette configuration, wp_xmlrpc de votre site Web sera désactiver.

> Sources :
>
> *https://ws.apache.org/xmlrpc/*

Après avoir fait ces modifications dans le fichier .htaccess, il se peut que des services que vous utiliser ne fonctionnent plus.
Si vous souhaitez revenir en arrière, soit vous devez retirer la configuration qui a été ajouter dans le fichier .htaccess ou sinon il faudra restauré le fichier .htaccess.sauveguarde.
Ou vous pouvez également modifier votre structure de la manière suivant :

```
# BEGIN Disable XML-RPC request
<Files xmlrpc.php>
   order allow,deny
   deny from all
   allow from xxx.xxx.xxx.xxx
</Files>
# END Disable XML-RPC
```

En remplacent xxx.xxx.xxx.xxx par l’ip de votre service

Wordpress

WordPress référencement naturel (SEO)

Dans cette procédure, je vais vous expliquer comment améliorer son référencement naturel avec WordPress. Posséder un site WordPress n'est pas quelque chose de compliqué à obtenir. Mais par contre, avoir des visiteurs à l'aide de référencement naturel n'est aussi simple à mettre en œuvre. Le Référencement naturel est une pratique qui consiste à améliorer son positionnement dans les moteurs de recherche sans financer de la publicité (ex : Google Ads).

Prérequis pour améliorer son référencement naturel avec WordPress :

- Avoir un site WordPress qui est indexé sur plusieurs moteurs de recherche (ex: Google, Bing, etc.)

Optimiser son référencement naturel avec WordPress :

Le référencement naturel est aussi appelé SEO (**Search Engine Optimisation**), avoir un bon SEO nécessite quelques bonnes pratiques à mettre en œuvre.

Voici ma liste des pratiques à utiliser pour obtenir un meilleur positionnement :

- Un chargement rapide du site web
- Du contenu inédit et régulier
- Ciblé des mots-clés
- Effectuer des maillages interne et externe
- Utiliser correctement les Images

Un chargement rapide du site Web :

Afin d'avoir un site rapide avec WordPress, il faut un hébergeur de qualité, mais pas seulement et compresser son contenu à l'aide d'outils tel que Gzip.
Il faut également se protéger contre les cyberattaques, je vous conseille d'utiliser les outils suivants : Crowdsec, Change wp-admin login et Limit Login Attempts Reloaded.

Ensuite, il faut utiliser des CDN pour délivrer son site plus rapidement.

Il est possible de mesurer la vitesse de son site à l'aide des outils suivants :

- PageSpeed Insights
- GTmetrix
- Vérificateur de compression Gzip (Websiteplanet)

Du contenu inédit et régulier :

Je vous conseille de publier au moins une fois par semaine un article, que vous avez écrit vous-même (sans faire de copier/coller).

Il est préférable de rien poster plutôt que de faire un copier/coller d'un article de quelqu'un d'autre.

Cibler des mots-clés :

Utiliser les bons mots-clés permet d'obtenir plus d'impression/clics sur son article.

Certains outils mesurent les performances comme Ubersuggest ou Google Analytics / Matomo.

Effectuer des maillages interne et externe :

Dans vos pages et articles, n'hésitez pas à renvoyer le visiteur sur d'autres pages de votre site.

Vous pouvez par ailleurs renvoyer des visiteurs vers les sites de vos contacts afin de rediriger vos visiteurs.

Utiliser correctement les Images :

Pour utiliser correctement les images, il faut utiliser un nom de fichier qui correspond à l’image et au mot-clé de l’article.

Renseigner un Texte alternatif qui, comme le nom de l’image, correspond à l’image et au mot-clé de l’article.

> Sources :
>
> *https://www.seo.fr/definition/seo-definition*

Wordpress

WordPress Ajouter un Sitemap

Un Sitemap et une page qui permet de trouver l'ensemble des pages d'un site web. Il existe les Sitemap Xml utiliser pour le référencement et les Sitemap Html utilisé par les utilisateurs du site web. Dans cette procédure nous allons voir comment créer ces deux types de Sitemap sur un site WordPress à l'aide de plugins. Ajouter un Sitemap sur son site WordPress et utile pour le référencement SEO et faciliter l'utilisation du site pour les utilisateurs. Le Sitemap est aussi important pour indexer toutes les pages de son site web sans en oublier.

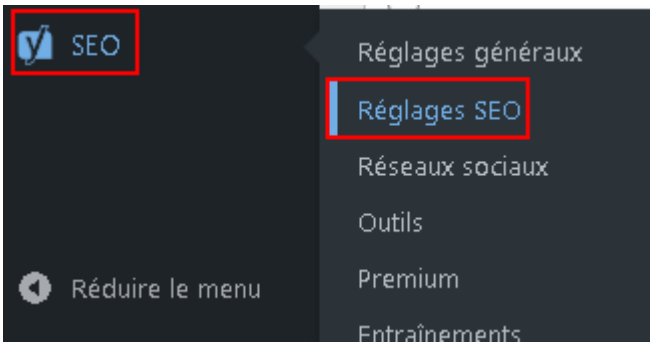
Prérequis :

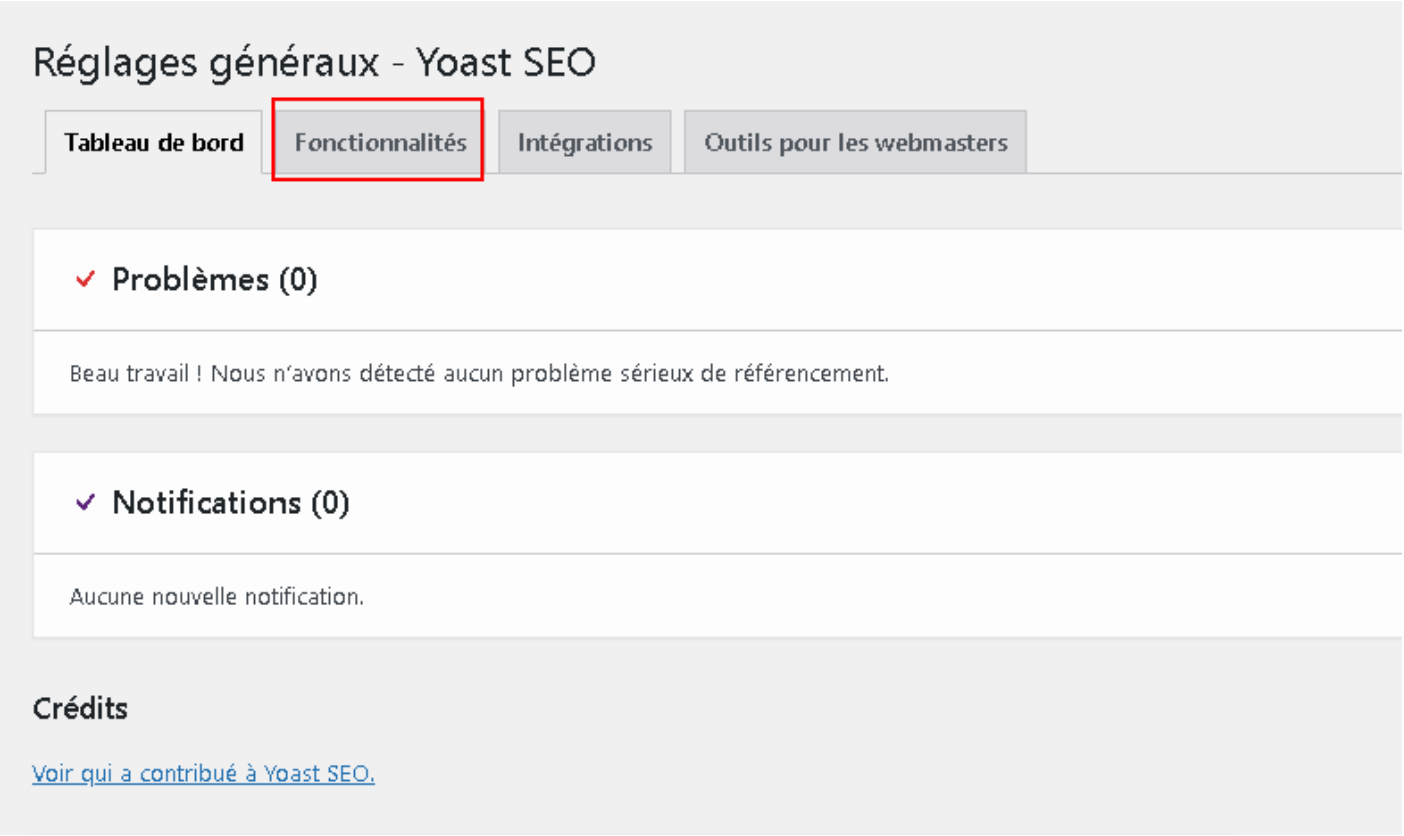
- Une site WordPress

WordPress Ajouter un Sitemap XML :

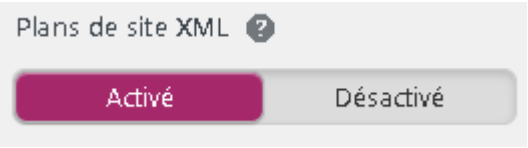
Pour ajouter un Sitemap XML sur son site, il faut installer et activer l'extension **Yoast SEO**

Ensuite, il faudra se rendre dans la catégorie **SEO -> Réglages SEO -> Fonctionnalités**

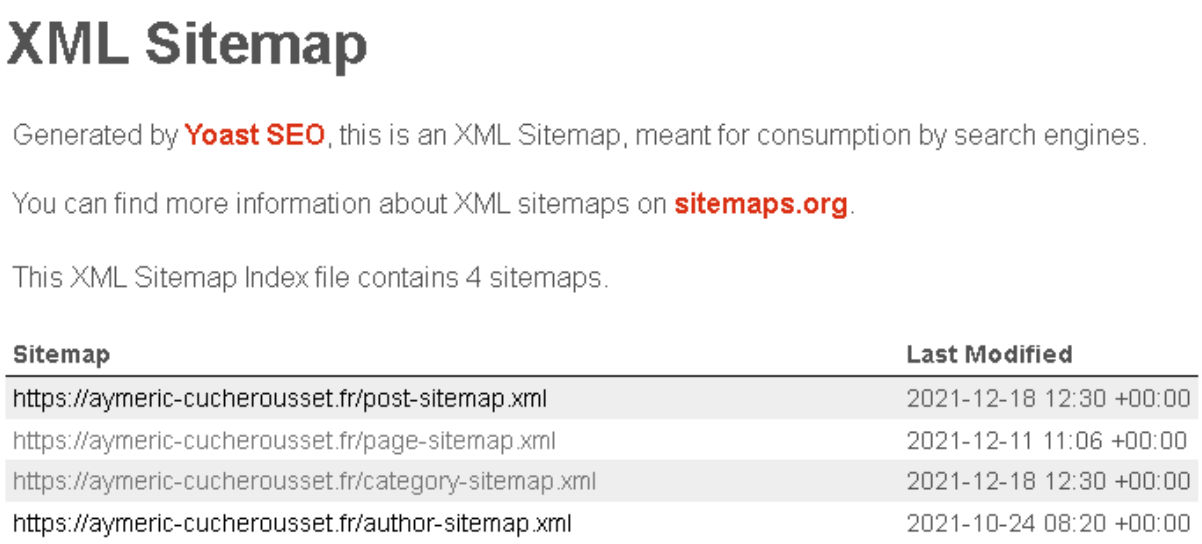




Puis ensuite activé la fonctionnalité : **Plans de site XML**



Ensuite le sitemap sera présent à l'emplacement : https://votresite.com/sitemap_index.xml

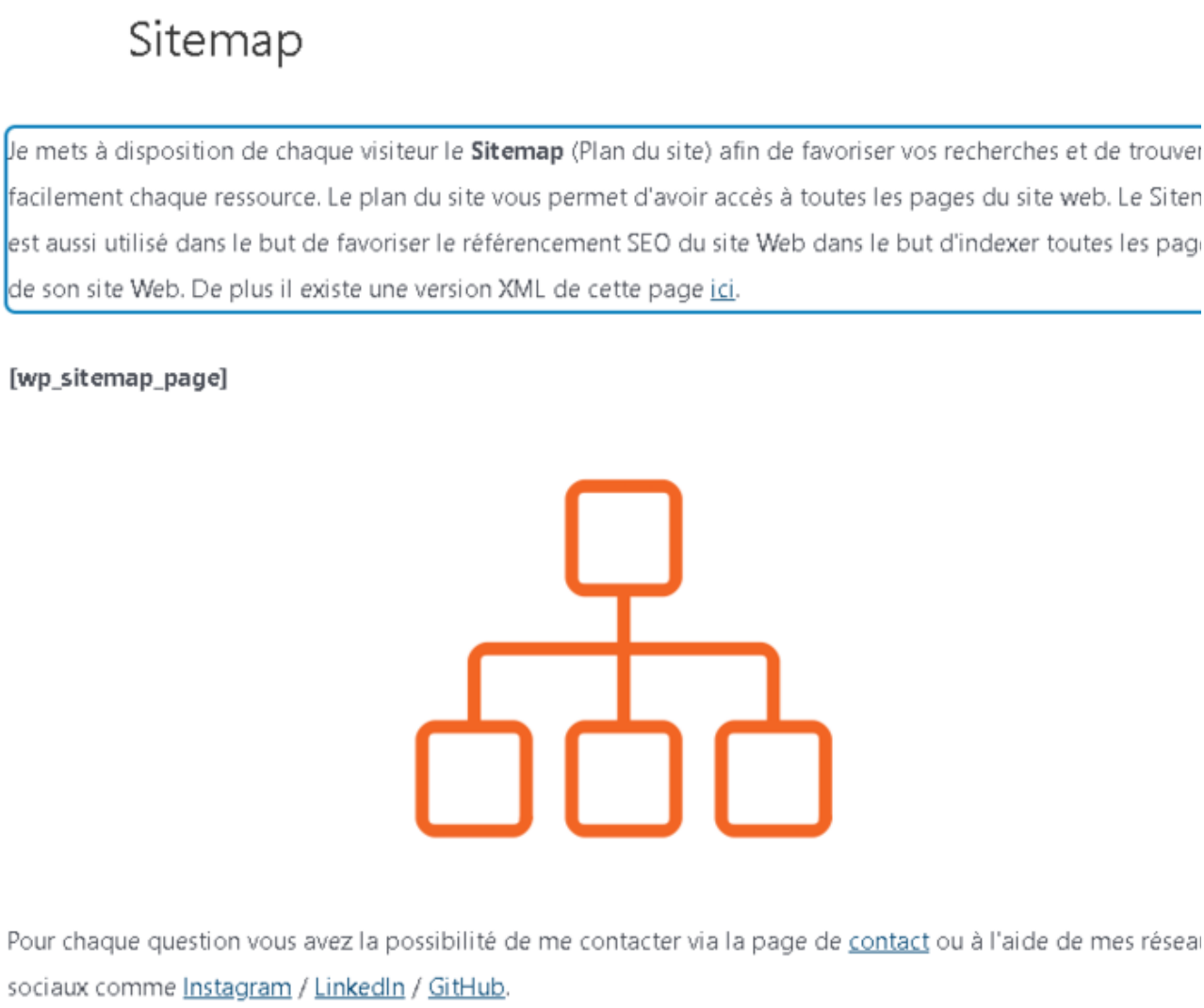


WordPress Ajouter un Sitemap HTML :

Pour ajouter un Sitemap HTML, il faudra installer et activer l'extension : **WP Sitemap Page**

Ensuite il faudra créer une nouvelle page dans cette page il faudra ajouter sur cette page le short code suivant :

```
[wp_sitemap_page]
```



Ensuite le sitemap va être automatiquement générer sur la nouvelle page. Puis vous avez la possibilité de mettre dans votre footer un lien vers votre Sitemap.

Avoir un SiteMap Xml ou Html pour son site est important car il permet moteur de recherches d'indexer toutes les pages du site.
Un fois vos Sitesmaps générés vous pouvez les envoyer aux moteurs de recherches pour qu'ils puissent indexer toutes les pages de votre site web.
Il est possible de mettre un lien du Sitemap Html dans votre footer pour le rendre accessible à proximité des CGU.

> Sources :
>
> *https://fr.wordpress.org/*

Wordpress

WordPress et Google Analytics

Dans cette procédure, je vais vous montrer comment lier son site web **WordPress** et **Google Analytics**. La réalisation de cette procédure sera utiliser sans utiliser de plugins WordPress. Installer Google Analytics permet de suivre les actions des utilisateurs de son site Web.

Prérequis :

- Un site WordPress
- Un thème enfant
- Les accès administrateurs du tableau de bord

Installation de Google Analytics sur WordPress

Pour commencer il faut se connecter un compte avec un compte Google sur le site de Google Analytics.

Une fois connecter sur Google Analytics il faut aller dans les paramètres en bas à gauche de la fenêtre.

Puis il faudra créer une propriété et répondre aux questions que Google Analytics va vous demander.

Une fois la propriété créer, rendez-vous sur l’assistant de configuration -> installation de la balise -> Ajouter un flux -> Web

Saisissez la configuration de votre site Web :

Récupérer le contenue du Global site tag

Puis ensuite il faudra ajouter une fonction dans votre thème enfant :

```
/**
* Ajout du code de suivi de Google Analytics dans le Header avant la fermeture de la balise Head
*/
add_action('wp_head', 'add_google_analytics');
function add_google_analytics() {
?>
<!-- Global site tag (gtag.js) - Google Analytics -->
```

```
<script async src="https://www.googletagmanager.com/gtag/js?id=G-5P696FEXTB"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'G-XXXXXXXXXX');
</script>
<?php }
/**
* Fin Google Analitics
*/
```

Puis remplacer ce script gtag.js par celui que Google Analytics vous a donné.

Les modifications seront effectives quelques minutes après la modification.

> Sources :
>
> *https://fr.wordpress.org/*

Wordpress

WordPress Thème enfant

Dans cette procédure je vais vous expliquer comment personnaliser un thème **WordPress** à l'aide d'un **thème enfant**. Après avoir suivit la procédure vous serez en mesure de personnaliser votre thème principale **WordPress** grâce a un **thème enfant.** L'utilisation d'un thème enfant permet de réaliser des modifications et de les conserver après les mises à jour thème.

Prérequis :

- Un site web WordPress
- Les accès SSH ou FTP de l'hébergeur
- Les accès Administrateur de WordPress

Création du Thème Enfant WordPress :

A l'aide des accès SSH ou FTP connectez vous à l'emplacement de votre site sur la machine qui héberge le site.

Puis rendez-vous dans l'emplacement : wp-content/themes

Ensuite créer un dossier qui s'appellera le-nom-de-votre-theme**-child**

Après la création de ce dossier, il faut créer deux fichiers dans ce dossier :

- style.css
- functions.php

Puis dans le fichier « style.css », il faudra l'adapter à votre thème avec :

- **Theme Name :** Le nom de votre thème enfant
- **Description :** La description du thème
- **Author :** Le nom de l'auteur du thème enfant
- **Author URI :** Un lien de site du thème enfant
- **Template :** Le thème parent
- **Version :** La version du thème enfant

```
/*
Theme Name: astra-child
Description: Modification du theme parent.
Author: Cucherousset Aymeric
Author URI: https://aymeric-cucherousset.fr
Template: astra
Version: 1.0
*/
```

Après ce commentaire vous pouvez y ajouter vos modifications de CSS.

Ensuite il faut modifier le fichier functions.php pour activer le thème enfant :

Le code pour activer le thème peut être différent d'un thème parent à un autre.

```
<?php

/**
** activation theme
**/

add_action( 'wp_enqueue_scripts', 'theme_enqueue_styles' );
function theme_enqueue_styles() {
wp_enqueue_style( 'parent-style', get_template_directory_uri() . '/style.css' );
wp_enqueue_style( 'astra-child', get_stylesheet_uri() );
}

?>
```

Pour rendre effectif le thème il ne reste plus qu'a se rendre dans le pannel admin de WordPress.

Il faut ensuite aller dans Apparences -> Thèmes.

Puis activer le thème enfant. Désormais les modifications faites sur le fichier (Accessible depuis Apparences -> Editeur de thèmes) du thème enfant serons prises en compte sur le thème et seront conservés au mises à jour du thème.

> Sources :
>
> *https://fr.wordpress.org/*

Caddy

Introduction

Dans le monde de l’hébergement de sites web, choisir le bon serveur web peut faire toute la différence en termes de performance, de sécurité et de facilité de gestion. Caddy est un serveur web moderne et performant qui se distingue par sa simplicité d’utilisation et ses fonctionnalités avancées. Conçu pour automatiser les tâches fastidieuses comme la gestion des certificats SSL/TLS, Caddy permet aux administrateurs systèmes de se concentrer sur des aspects plus importants de leur infrastructure. Dans cet article, je vais vous présenter les caractéristiques principales de Caddy, son historique, les concepts de base, et quelques exemples de configurations pour que vous puissiez tirer le meilleur parti de ce serveur web innovant.

Historique de Caddy

Caddy a vu le jour en 2015, créé par **Matt Holt** avec une vision claire : simplifier la configuration et la gestion des serveurs web tout en intégrant des fonctionnalités modernes dès le départ. À l’époque, la gestion des certificats SSL/TLS était une tâche compliquée et chronophage pour les administrateurs systèmes. **Matt Holt** a voulu changer cela en intégrant le support automatique de Let’s Encrypt directement dans **Caddy**, rendant ainsi la sécurité web accessible à tous sans effort supplémentaire.

Dès sa sortie, **Caddy** s’est distingué par sa philosophie axée sur la simplicité et l’automatisation. En plus de la gestion automatisée des certificats SSL/TLS, **Caddy** a introduit une syntaxe de configuration claire et lisible, appelée le **Caddyfile**, facilitant ainsi la vie des développeurs et des administrateurs systèmes.

Au fil des années, **Caddy** a continué d’évoluer et d’innover, ajoutant des fonctionnalités comme le support des plugins pour étendre ses capacités, une meilleure gestion des performances et des améliorations constantes en termes de sécurité. La communauté autour de **Caddy** a également grandi, contribuant à son développement et à son adoption dans divers environnements, des petits sites web personnels aux grandes infrastructures d’entreprise.

Aujourd’hui, **Caddy** est reconnu non seulement pour sa simplicité et ses fonctionnalités intégrées, mais aussi pour sa robustesse et sa flexibilité, faisant de lui un choix populaire parmi les administrateurs systèmes cherchant une solution moderne et efficace pour leurs besoins en serveur web.

Fonctionnalités principales de Caddy

Caddy se distingue par un ensemble de fonctionnalités qui le rendent unique parmi les serveurs web. Voici quelques-unes des fonctionnalités principales qui font de **Caddy** un choix populaire pour les administrateurs systèmes et les développeurs :

L'une des forces de Caddy réside dans sa facilité de configuration. Il utilise un fichier de configuration appelé **Caddyfile**, qui est conçu pour être lisible et facile à écrire. Voici un exemple de **Caddyfile** minimaliste :

```
example.com {
    root * /var/www/html
    file_server
}
```

Dans cet exemple, **Caddy** sert le contenu du répertoire `/var/www/html` lorsque le domaine `example.com` est accédé.

HTTPS automatique

Caddy automatise entièrement la gestion des certificats SSL/TLS via Let's Encrypt. Cela signifie que dès que vous configurez un domaine dans **Caddy**, il obtient automatiquement les certificats nécessaires et les renouvelle périodiquement sans intervention manuelle. Cette fonctionnalité simplifie grandement la sécurisation des sites web.

Extensibilité

Caddy est conçu pour être extensible grâce à un système de plugins. Ces plugins peuvent ajouter des fonctionnalités supplémentaires comme des authentifications spécifiques, des redirections avancées ou des intégrations avec d'autres services. La communauté développe activement des plugins pour répondre à divers besoins.

Performances élevées

Caddy est conçu pour être rapide et efficient en termes de ressources. Il utilise des techniques modernes pour gérer les connexions et les requêtes, ce qui permet de servir un grand nombre de requêtes simultanées sans sacrifier la performance.

Support natif des HTTP/2 et HTTP/3

Caddy prend en charge les protocoles HTTP/2 et HTTP/3 nativement, ce qui améliore la vitesse de chargement des pages et la sécurité. Ces protocoles sont conçus pour offrir une meilleure performance en termes de latence et de débit.

Gestion des fichiers statiques

Avec la directive `file_server`, **Caddy** peut facilement servir des fichiers statiques. Il peut également gérer les répertoires et fournir des listings de répertoires automatiquement.

Reverse proxy

Caddy peut agir en tant que revers-proxy, redirigeant les requêtes vers d'autres serveurs backend. Cette fonctionnalité est particulièrement utile pour les applications réparties sur plusieurs serveurs ou pour les microservices.

Redirections et réécritures d'URL

Caddy offre des fonctionnalités puissantes pour la redirection et la réécriture d'URL. Vous pouvez facilement configurer des redirections permanentes ou temporaires, ainsi que des réécritures d'URL complexes.

Journaux et surveillance

Caddy fournit des journaux détaillés et des outils de surveillance intégrés qui permettent de suivre l'activité du serveur et de diagnostiquer les problèmes. Ces outils sont essentiels pour maintenir une infrastructure stable et performante.

Sécurité renforcée

En plus de la gestion automatisée des certificats, **Caddy** intègre des fonctionnalités de sécurité telles que la prévention des attaques DDoS, la protection contre les scripts intersites (XSS) et bien d'autres.

Avec toutes ces fonctionnalités, **Caddy** se positionne comme un serveur web moderne et performant, idéal pour une variété de cas d'utilisation, des sites web simples aux infrastructures complexes nécessitant une gestion avancée des requêtes et des ressources.

Concepts de base

Pour tirer pleinement parti de **Caddy**, il est essentiel de comprendre quelques concepts de base. Ces concepts vous aideront à configurer et à gérer efficacement votre serveur web avec Caddy.

Caddyfile

Le **Caddyfile** est le fichier de configuration principal de Caddy. Il est conçu pour être simple et lisible, ce qui facilite la configuration des sites et des services. Un Caddyfile typique est structuré en blocs de configuration, chacun représentant un site ou un service distinct.

Voici un exemple de Caddyfile de base :

```
example.com {
    root * /var/www/html
    file_server
}
```

Dans cet exemple, le bloc `example.com` configure le domaine `example.com` pour servir le contenu du répertoire `/var/www/html` en utilisant la directive `file_server`.

Directives

Les **directives** sont des commandes spécifiques utilisées dans le Caddyfile pour configurer le comportement du serveur. Chaque directive a une fonction particulière, comme servir des fichiers statiques, rediriger des requêtes, ou configurer des paramètres de sécurité. Par exemple :

```
root * /var/www/html
file_server
```

Ces directives configurent le serveur pour servir des fichiers statiques à partir du répertoire `/var/www/html`.

Plugins

Caddy est conçu pour être extensible grâce à son support pour les **plugins**. Ces plugins permettent d'ajouter des fonctionnalités supplémentaires à Caddy, telles que l'authentification, la gestion des URL courtes ou des intégrations avec d'autres services. Les plugins peuvent être installés et configurés via le Caddyfile.

Routes et gestion des chemins

Les **routes** dans Caddy permettent de gérer comment les requêtes sont dirigées et traitées. Vous pouvez configurer des routes pour rediriger certaines requêtes vers des services backend, pour appliquer des réécritures d'URL, ou pour servir différents contenus selon le chemin de la requête. Voici un exemple de configuration de route pour un proxy inversé :

```
example.com {
reverse_proxy /api/* http://backend:8080
}
```

Dans cet exemple, toutes les requêtes vers `example.com/api/*` sont redirigées vers un serveur backend fonctionnant sur `http://backend:8080`.

Installation de Caddy

L'installation de **Caddy** est un processus simple et direct, que vous soyez sur un système d'exploitation Linux, macOS ou Windows. Dans cette section, je vais vous guider à travers les étapes d'installation de Caddy sur différents systèmes.

Installation sur Linux

Pour installer **Caddy** sur un système basé sur Debian (comme Ubuntu), vous pouvez suivre ces étapes :

Mettez à jour la liste des paquets et installez les prérequis :

```
sudo apt update
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https
```

Ajoutez la clé GPG de Caddy et le dépôt APT :

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo tee
/etc/apt/trusted.gpg.d/caddy-stable.asc
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee
/etc/apt/sources.list.d/caddy-stable.list
```

Mettez à jour la liste des paquets et installez Caddy :

```
sudo apt update
sudo apt install caddy
```

Installation sur macOS

Pour installer Caddy sur macOS, vous pouvez utiliser Homebrew, un gestionnaire de paquets populaire pour macOS :

Installez Homebrew si ce n'est pas déjà fait. Vous pouvez le faire en suivant les instructions sur brew.sh ↗.

Installez Caddy en utilisant Homebrew :

```
brew install caddy
```

Installation sur Windows

Pour installer Caddy sur Windows, vous pouvez télécharger l'exécutable directement depuis le site officiel ou utiliser un gestionnaire de paquets comme Scoop ou Chocolatey.

Utilisation de Scoop

Installez Scoop si ce n'est pas déjà fait. Vous pouvez le faire en suivant les instructions sur scoop.sh ↗.

Installez Caddy en utilisant Scoop :

```
scoop install caddy
```

Utilisation de Chocolatey

Installez Chocolatey si ce n’est pas déjà fait. Vous pouvez le faire en suivant les instructions sur chocolatey.org ↗.

Installez Caddy en utilisant Chocolatey :

```
choco install caddy
```

Vérification de l’installation

Après avoir installé Caddy, vous pouvez vérifier que l’installation a réussi en exécutant la commande suivante dans votre terminal ou votre invite de commande :

```
caddy version
```

Cette commande doit afficher la version de Caddy installée sur votre système.

Démarrage et arrêt de Caddy

Pour démarrer Caddy en utilisant un fichier de configuration, utilisez la commande suivante :

```
caddy run --config /path/to/Caddyfile
```

Pour arrêter Caddy, vous pouvez utiliser `Ctrl+C` dans le terminal où Caddy est en cours d’exécution.

Exécution en tant que service

Sur les systèmes Linux, vous pouvez configurer Caddy pour qu’il s’exécute en tant que service système, ce qui permet de démarrer automatiquement Caddy au démarrage du système. Voici comment faire sur un système utilisant `systemd` :

Créez un fichier de service `systemd` pour Caddy :

```
sudo nano /etc/systemd/system/caddy.service
```

Ajoutez les lignes suivantes dans ce fichier :

```
[Unit]
Description=Caddy web server
After=network.target


[Service]
ExecStart=/usr/bin/caddy run --config /etc/caddy/Caddyfile
Restart=always
User=caddy
Group=caddy


[Install]
WantedBy=multi-user.target
```

Rechargez `systemd` et démarrez le service Caddy :

```
sudo systemctl daemon-reload
sudo systemctl start cadd
sudo systemctl enable caddy
```

Avec ces instructions, vous êtes prêt à installer et à exécuter Caddy sur différentes plateformes. Dans la prochaine section, je vais vous montrer comment configurer Caddy pour servir des sites web et des applications.

Configuration de base

Une fois **Caddy** installé, la prochaine étape consiste à le configurer pour servir des sites web et des applications. Dans cette section, je vais vous montrer comment créer un fichier de configuration Caddyfile et expliquer les directives de base pour démarrer rapidement.

Création du Caddyfile

Le **Caddyfile** est le fichier de configuration principal de Caddy. Il est conçu pour être simple et lisible. Voici un exemple de Caddyfile minimaliste pour servir un site web statique :

```
example.com {
root * /var/www/html
file_server
}
```

Dans cet exemple :

- `example.com` est le domaine pour lequel Caddy doit servir les fichiers.
- `root * /var/www/html` spécifie le répertoire racine contenant les fichiers du site web.
- `file_server` indique à Caddy de servir les fichiers statiques à partir du répertoire racine.

Servir des fichiers statiques

Pour servir des fichiers statiques à partir d'un répertoire, utilisez la directive `file_server`. Voici un exemple de configuration complète :

```
example.com {
root * /var/www/html
file_server browse
}
```

Avec `browse`, Caddy générera automatiquement une liste des fichiers et des répertoires si l'index du répertoire n'est pas trouvé.

Utilisation de variables d'environnement

Caddy permet l'utilisation de variables d'environnement pour rendre la configuration plus flexible. Voici comment définir et utiliser une variable d'environnement dans le Caddyfile :

```
{$SITE_DOMAIN} {
root * /var/www/html
file_server
tls {$EMAIL}
}
```

Avant de démarrer Caddy, vous pouvez définir les variables d’environnement :

```
export SITE_DOMAIN=example.com
export EMAIL=email@example.com
caddy run --config /path/to/Caddyfile
```

Gérer les erreurs personnalisées

Caddy permet de définir des pages d’erreur personnalisées pour différents codes d’erreur HTTP. Voici comment configurer une page d’erreur personnalisée pour les erreurs 404 :

```
example.com {
root * /var/www/html
file_server
handle_errors {
@404 {
expression {http.error.status_code} == 404
}
rewrite @404 /html
file_server
}
}
```

Activer la compression

Pour améliorer les performances, vous pouvez activer la compression des réponses HTTP. Caddy prend en charge la compression gzip et zstd :

```
example.com {
root * /var/www/html
file_server
encode gzip zstd
}
```

Utilisation des directives de réécriture

Les directives de réécriture permettent de modifier les URL des requêtes avant qu'elles ne soient traitées. Voici un exemple de réécriture d'URL :

```
example.com {
root * /var/www/html
file_server
rewrite /old-path /new-path
}
```

Configuration des en-têtes HTTP

Caddy permet de définir des en-têtes HTTP personnalisés pour les réponses. Voici un exemple de configuration des en-têtes :

```
example.com {
root * /var/www/html
file_server
header {
Strict-Transport-Security "max-age=31536000;"
X-Content-Type-Options "nosniff"
}
}
```

Rechargement de la configuration

Pour appliquer les modifications apportées au Caddyfile, vous devez recharger la configuration de Caddy. Cela peut être fait sans redémarrer le serveur en utilisant la commande suivante :

```
caddy reload --config /path/to/Caddyfile
```

Avec ces configurations de base, vous pouvez commencer à utiliser Caddy pour servir des sites web et des applications. La prochaine section abordera la gestion des certificats SSL/TLS de manière plus détaillée.

Gestion des certificats SSL/TLS

Une des fonctionnalités les plus appréciées de **Caddy** est sa capacité à gérer automatiquement les certificats SSL/TLS. Cette fonctionnalité simplifie grandement la sécurisation des sites web, éliminant la nécessité de gérer manuellement les certificats.

Obtention automatique des certificats

Caddy obtient automatiquement les certificats SSL/TLS pour vos domaines via Let’s Encrypt. Dès que vous configurez un domaine dans le Caddyfile, Caddy se charge d’obtenir et de renouveler les certificats sans intervention supplémentaire. Voici un exemple de configuration de base pour un site sécurisé :

```
example.com {
root * /var/www/html
file_server
tls email@example.com
}
```

Dans cet exemple, la directive `tls` avec une adresse email permet à Caddy de gérer les certificats SSL/TLS pour `example.com`.

Configuration avancée des certificats

Caddy offre des options avancées pour la gestion des certificats, comme l’utilisation de certificats personnalisés ou le contrôle des paramètres de TLS. Voici comment configurer des certificats personnalisés :

```
example.com {
root * /var/www/html
file_server
```

```
tls /path/to/cert.pem /path/to/key.pem
}
```

Dans cet exemple, Caddy utilise les fichiers `cert.pem` et `key.pem` pour le certificat et la clé privée.

Utilisation de DNS Challenge

Pour les domaines où le HTTP Challenge ne fonctionne pas (par exemple, les domaines wildcard), vous pouvez utiliser le DNS Challenge. Cela nécessite des configurations spécifiques pour le DNS provider. Voici un exemple pour Cloudflare :

Installez le plugin DNS correspondant :

```
xcaddy build --with github.com/caddy-dns/cloudflare
```

Configurez le DNS Challenge dans le Caddyfile :

```
example.com {
root * /var/www/html
file_server
tls {
dns cloudflare {env.CLOUDFLARE_API_TOKEN}
}
}
```

Définissez la variable d’environnement avec votre token API Cloudflare :

```
export CLOUDFLARE_API_TOKEN=your-cloudflare-api-token
caddy run --config /path/to/Caddyfile
```

Renouvellement des certificats

Caddy gère le renouvellement automatique des certificats avant leur expiration. Vous n’avez donc pas besoin de vous inquiéter de l’expiration des certificats. Caddy surveille les certificats et les renouvelle au besoin.

Gestion des erreurs liées aux certificats

Il est important de surveiller les logs de Caddy pour détecter toute erreur liée aux certificats, comme l'échec de l'obtention ou du renouvellement des certificats. Voici comment configurer Caddy pour journaliser les erreurs liées aux certificats :

```
example.com {
root * /var/www/html
file_server
tls email@example.com
log {
output file /var/log/caddy/caddy.log
level ERROR
}
}
```

Paramètres TLS avancés

Pour les utilisateurs avancés, Caddy permet de configurer des paramètres TLS spécifiques, comme les suites de chiffrement ou les protocoles à utiliser. Voici un exemple de configuration TLS avancée :

```
example.com {
root * /var/www/html
file_server
tls email@example.com {
protocols tlstls3
ciphers TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
}
}
```

Dans cet exemple, Caddy est configuré pour utiliser uniquement TLS et avec des suites de chiffrement spécifiques.

Monitoring et gestion des certificats

Pour une gestion proactive, vous pouvez utiliser des outils de monitoring pour surveiller l'état des certificats SSL/TLS et recevoir des alertes en cas de problème. Des solutions comme Let's Monitor ou des scripts personnalisés peuvent être intégrés à votre infrastructure de surveillance.

Avec ces fonctionnalités de gestion des certificats SSL/TLS, Caddy simplifie la sécurisation de vos sites web, en s'assurant que les certificats sont toujours à jour et configurés correctement. Dans la prochaine section, nous allons explorer l'utilisation des plugins pour étendre les capacités de Caddy.

Utilisation des plugins Caddy

Caddy est conçu pour être extensible et une des manières principales d'étendre ses fonctionnalités est par l'utilisation de **plugins**. Ces plugins permettent d'ajouter des fonctionnalités supplémentaires telles que l'authentification, la gestion des URL courtes, ou des intégrations avec d'autres services.

Les plugins pour Caddy sont disponibles sous forme de modules que vous pouvez inclure lors de la compilation de Caddy. Ils peuvent ajouter des fonctionnalités spécifiques ou modifier le comportement par défaut de Caddy. Vous pouvez trouver une liste des plugins disponibles sur le site officiel de Caddy.

Installation de plugins avec xcaddy

Pour installer des plugins, vous pouvez utiliser l'outil `xcaddy`, qui permet de compiler Caddy avec les plugins souhaités. Voici comment installer `xcaddy` et compiler Caddy avec un plugin :

Installez `xcaddy` :

```
go install github.com/caddyserver/xcaddy/cmd/xcaddy@latest
```

Compilez Caddy avec un plugin, par exemple le plugin `caddy-l4` pour le support de niveau 4 (TCP/UDP) :

```
xcaddy build --with github.com/mholt/caddy-l4
```

Une fois la compilation terminée, un nouveau binaire `caddy` est créé avec le plugin intégré.

Authentification

Le plugin `http.authz` permet d'ajouter des mécanismes d'authentification à vos sites web. Voici un exemple de configuration pour utiliser HTTP Basic Auth :

```
example.com {
root * /var/www/html
file_server



basicauth {
user hashed-password
}
}
```

Vous pouvez générer le mot de passe haché en utilisant une commande comme `htpasswd` :

```
htpasswd -nB user
```

Gestion des URL courtes

Le plugin `http.urlshort` permet de créer des redirections courtes. Voici un exemple de configuration :

```
example.com {
route /short {
redir /long-url
}
}
```

WebSocket

Pour ajouter le support WebSocket, vous pouvez utiliser le plugin `caddy.websocket`. Voici un exemple de configuration pour un serveur WebSocket simple :

```
example.com {
root * /var/www/html
file_server
reverse_proxy /ws localhost:8080 {
transport http {
versions h2c 2
}
}
}
```

Cloudflare DNS

Le plugin `caddy-dns/cloudflare` permet d'utiliser le DNS Challenge avec Cloudflare pour les certificats SSL/TLS. Voici un exemple de configuration :

```
example.com {
root * /var/www/html
file_server
tls {
dns cloudflare {env.CLOUDFLARE_API_TOKEN}
}
}
```

Configuration des plugins dans le Caddyfile

Les plugins sont configurés dans le Caddyfile en utilisant des directives spécifiques à chaque plugin. Chaque plugin dispose de sa propre documentation pour les options de configuration disponibles. Voici un exemple de Caddyfile avec plusieurs plugins configurés :

```
example.com {
root * /var/www/html
file_server



# Basic Auth plugin
```

```
basicauth {
user hashed-password
}


# URL Shortener plugin
route /short {
redir /long-url
}


# WebSocket plugin
reverse_proxy /ws localhost:8080 {
transport http {
versions h2c 2
}
}


# Cloudflare DNS plugin
tls {
dns cloudflare {env.CLOUDFLARE_API_TOKEN}
}
}
```

Mise à jour et gestion des plugins

Pour mettre à jour Caddy avec les plugins, vous pouvez simplement recompiler Caddy avec `xcaddy` en spécifiant les versions mises à jour des plugins. Il est important de vérifier régulièrement les mises à jour de sécurité et les nouvelles fonctionnalités des plugins utilisés.

Dépannage des plugins

En cas de problèmes avec les plugins, consultez les logs de Caddy pour obtenir des informations sur les erreurs. Vous pouvez également consulter la documentation spécifique du plugin et les forums de support pour obtenir de l’aide.

Avec cette flexibilité offerte par les plugins, vous pouvez adapter Caddy à une multitude de scénarios et besoins spécifiques. La prochaine section traitera des meilleures pratiques pour la gestion et l'optimisation des performances de Caddy.

Gestion et optimisation des performances

Caddy est conçu pour être performant dès la sortie de la boîte, mais il existe des pratiques et des configurations supplémentaires qui peuvent améliorer encore plus ses performances. Dans cette section, je vais aborder les techniques de gestion et d'optimisation des performances pour tirer le meilleur parti de votre serveur Caddy.

Gestion de la mémoire et des ressources

Pour surveiller et gérer l'utilisation de la mémoire et des ressources CPU par Caddy, vous pouvez utiliser des outils de monitoring comme **Prometheus** et **Grafana**. Voici comment configurer Caddy pour exporter des métriques compatibles avec Prometheus :

Ajoutez le module Prometheus lors de la compilation de Caddy :

```
xcaddy build --with github.com/prometheus/client_golang/prometheus
```

Configurez le Caddyfile pour exporter les métriques :

```
{
    metrics {
        prometheus
    }
}

example.com {
    root * /var/www/html
    file_server
}
```

Configurez Prometheus pour scraper les métriques exportées par Caddy en ajoutant cette section dans le fichier de configuration de Prometheus (`prometheus.yml`) :

```
scrape_configs:
- job_name: "caddy"
static_configs:
- targets: ["localhost:2019"]
```

Compression et mise en cache

La compression et la mise en cache des réponses peuvent considérablement améliorer les performances de votre site web.

Compression

Pour activer la compression des réponses HTTP, utilisez la directive `encode` :

```
example.com {
root * /var/www/html
file_serve
encode gzip zstd
}
```

Mise en cache

Bien que Caddy ne propose pas de solution de mise en cache intégrée, vous pouvez utiliser des solutions externes comme **Varnish** ou des modules tiers pour gérer la mise en cache des réponses. Voici un exemple de configuration de Varnish avec Caddy :

Installez Varnish sur votre serveur.

Configurez Varnish pour écouter sur le port 80 et configurer Caddy pour écouter sur un port différent (par exemple, 8080).

Modifiez le fichier de configuration de Varnish (`default.vcl`) pour rediriger les requêtes vers Caddy :

```
vcl 0;



backend default {
.host = "1";
.port = "8080";
}



sub vcl_recv {
if (req.method == "PURGE") {
if (!client.ip ~ purgers) {
return (synth(405, "Not allowed."));
}
return (purge);
}
}



sub vcl_backend_response {
if (beresp.status == 200) {
set beresp.ttl = 10m;
}
}
```

Configurez le Caddyfile pour écouter sur le port 8080 :

```
example.com {
root * /var/www/html
file_server
encode gzip zstd
reverse_proxy localhost:8080
}
```

Optimisation des en-têtes HTTP

L’optimisation des en-têtes HTTP peut améliorer la sécurité et les performances de votre site web.

En-têtes de sécurité

Ajoutez des en-têtes de sécurité pour protéger votre site contre les attaques courantes :

```
example.com {
root * /var/www/html
file_server
header {
Strict-Transport-Security "max-age=31536000;"
X-Content-Type-Options "nosniff"
X-Frame-Options "DENY"
Referrer-Policy "no-referrer"
Content-Security-Policy "default-src 'self'"
}
}
```

En-têtes de mise en cache

Configurez des en-têtes de mise en cache pour les ressources statiques :

```
example.com {
    root * /var/www/html
    file_server
    header /static/* {
        Cache-Control "public, max-age=31536000, immutable"
    }
}
```

Gestion des connexions

Configurer le nombre maximal de connexions et de workers peut aider à gérer la charge sur votre serveur :

```
example.com {
    root * /var/www/html
    file_server
    tls email@example.com
    max_conns 100
    max_header_bytes 1048576
}
```

Utilisation de HTTP/2 et HTTP/3

Caddy supporte HTTP/2 par défaut et peut également être configuré pour utiliser HTTP/3 pour des performances encore meilleures :

Activez HTTP/3 dans le Caddyfile :

```
example.com {
    root * /var/www/html
    file_server
    tls {
        alpn h3
    }
}
```

Vérifiez que votre serveur supporte HTTP/3 et que les ports UDP nécessaires sont ouverts.

Load Balancing

Pour gérer un trafic important, vous pouvez configurer Caddy en tant que load balancer :

```
example.com {
    reverse_proxy {
        to backend1:8080 backend2:8080 backend3:8080
        lb_policy round_robin
    }
}
```

Optimisation des logs

```
example.com {
    root * /var/www/html
    file_server
    log {
        output file /var/log/caddy/access.log {
            roll_size 50mb
            roll_keep 5
            roll_keep_for 48h
        }
        level INFO
    }
}
```

En appliquant ces techniques de gestion et d'optimisation des performances, vous pouvez garantir que votre serveur Caddy fonctionne efficacement et reste performant même sous des charges élevées. Dans la prochaine section, je conclurai notre exploration de Caddy et résumerai les points clés abordés.

Conclusion

Au fil de ce billet, j'ai détaillé les nombreuses fonctionnalités et avantages qu'offre **Caddy** en tant que serveur web moderne et performant. Nous avons exploré son historique, ses concepts clés et ses fonctionnalités principales telles que la gestion automatique des certificats SSL/TLS, la flexibilité du fichier de configuration Caddyfile et l'utilisation de plugins pour étendre ses capacités.

En conclusion, **Caddy** se positionne comme un choix robuste et flexible pour la gestion des serveurs web. Grâce à ses fonctionnalités intégrées et à son extensibilité, il offre une solution complète pour déployer et gérer des sites web sécurisés et performants. J'espère que ce billet vous a aidé à mieux comprendre les avantages de **Caddy** et comment l'utiliser efficacement dans vos projets.

Set Up a CrowdSec Using OPNsense LAPI on Caddy

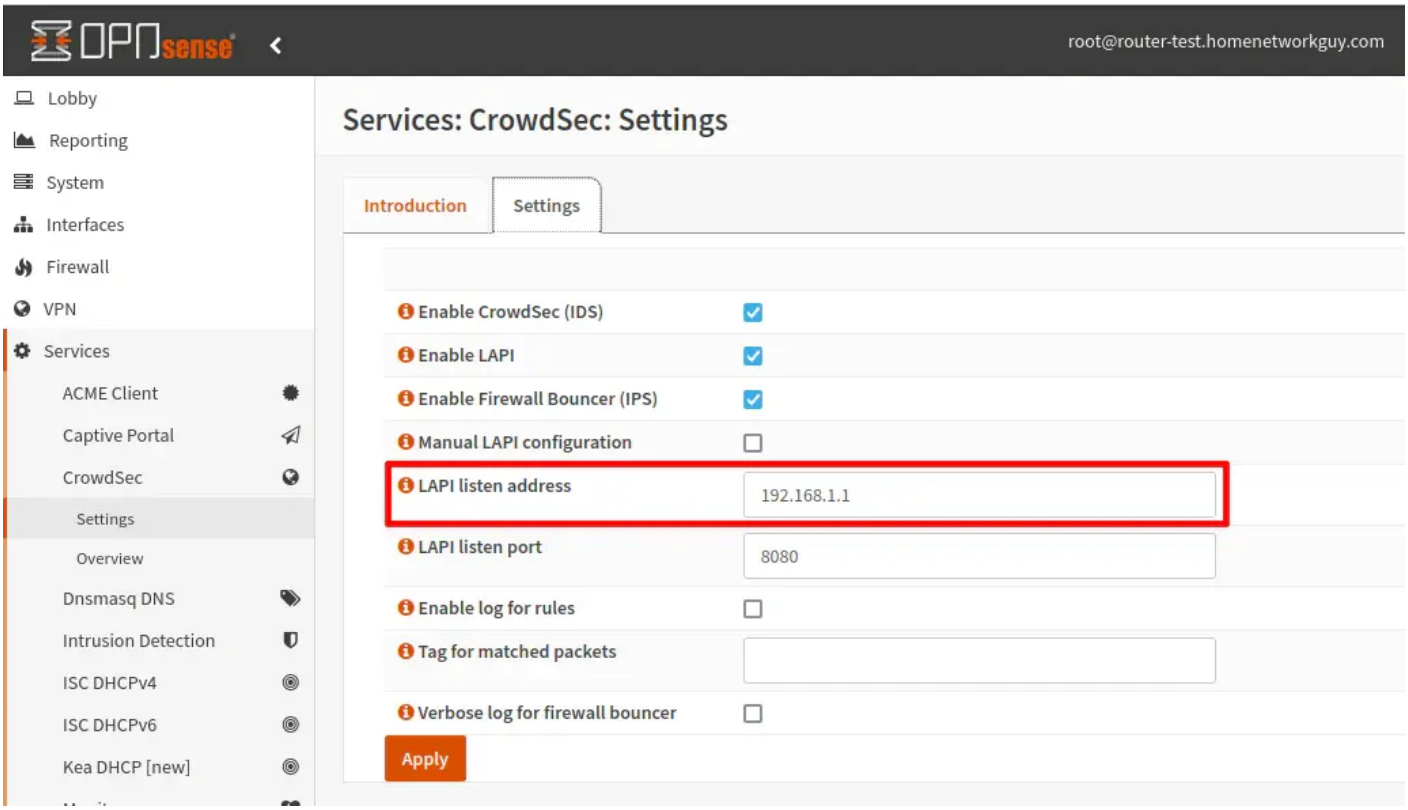
Prepare the OPNsense CrowdSec Configuration

Before setting up the Caddy reverse proxy, some settings for CrowdSec and firewall rules can be configured in OPNsense to prepare for a CrowdSec multi-server environment.

Update the Existing CrowdSec Plugin Configuration

The first thing you can do is change your CrowdSec plugin settings in OPNsense to allow other CrowdSec agents/bouncers to use the LAPI (Local API) on OPNsense. By default, the LAPI on OPNsense only listens on `localhost` (`127.0.0.1`).

In this example, I am setting the IP address to be on the LAN interface of `192.168.1.1`. You may wish to put it on a different interface.



You will need to start/stop the CrowdSec plugin for changes to take effect.

Create API Key for Caddy CrowdSec Bouncer

To prepare for setting up the CrowdSec bouncer for Caddy in a later step, you will need an API key generated for the bouncer.

Log into your OPNsense system via SSH or the console and issue the following command to create an API key. You may use any name you wish in place of `caddyDmz`. I used that name since this will be for a Caddy instance on the DMZ network.

```
sudo cscli bouncers add caddy
```

You should see the API key in the console output. Copy/paste this key until it is needed later.

```
API key for 'caddy':

   xxxxxxxxxxxxxxxxxxxxxx

Please keep this key since you will not be able to retrieve it!
```

Build Caddy with the Desired Modules

In order to use DNS challenges with Let's Encrypt and to use a CrowdSec bouncer, you will need to a build a custom Caddy executable to extend the base functionality. Fortunately, the build process is easy with `xcaddy` since you can build a Caddy executable with a single command.

I will be using the Cloudflare module for Let's Encrypt but you may use a different provider. Replace `github.com/caddy-dns/cloudflare` with a provider from the list found on GitHub.

```
xcaddy build \
    --with github.com/caddy-dns/cloudflare \
    --with github.com/hslatman/caddy-crowdsec-bouncer/crowdsec
```

Delete the old file and move the file to the `/usr/bin/` folder:

```
sudo rm /usr/bin/caddy
sudo mv caddy /usr/bin/
```

You should be able to run the `caddy` executable to ensure it can be found on the path.

```
caddy version
```

Modify Caddyfile

In the configuration file, you will need to enter the Cloudflare API key used for editing DNS zones for the `acme_dns cloudflare` option.

In addition to the DNS API key, newer versions of Caddy (v2.8.0+) require you to enter an email address for ZeroSSL (Caddy uses both Let's Encrypt and ZeroSSL for issuing certificates).

Then in a `crowdsec` block, you will need to enter the API key what was generated from OPNsense earlier. The URL is for the CrowdSec LAPI on OPNsense which is `192.168.1.1:8080`.

These first two settings should be contained in the global settings block as shown below.

```
{
        email nicolas.lespinasse@vainsta.fr
        acme_dns cloudflare xxxxxxxxxxxxxxxxxxxxxxxx
        admin :2019
        metrics
        crowdsec {
                api_key xxxxxxxxxxxxxxx
                api_url http://192.168.1.1:8080/
        }
```

```
}

panel.vainsta.fr {
        log {
                output file /var/log/caddy/panel-access.log {
                        roll_size 100mb
                        roll_keep 20
                        roll_keep_for 720h
                }
                format json
                level INFO
        }
        reverse_proxy https://192.168.1.4:444 {
                transport http {
                        tls_insecure_skip_verify
                }
        }
}
```

There are two additional CrowdSec bouncer options you may include.

If you wish for the bouncer to check the LAPI each time instead of caching the decisions in memory and polling the LAPI every 10 seconds by default, you can disable streaming by adding the `disable_streaming` option to the `crowdsec` block. Streaming decision information is more efficient if you have a lot of requests, but it is possible there will be a slight increase in delay when decisions on the LAPI have changed.

```
...
        crowdsec {
            ...
            disable_streaming
...
```

There is an option to enable “hard fails” if there is an issue connecting to the CrowdSec LAPI. This feature might be nice if you wish to prevent your services from being accessed if something is wrong with the LAPI since they will be unprotected by CrowdSec. Of course, that would negatively affect uptime, but it would make it very apparent something bad has happened.

I noticed it takes about 30 seconds to fail after the CrowdSec service is stopped in OPNsense. I was starting to wonder if this option was working properly, but I simply was not patient enough during my testing.

```
...
        crowdsec {
            ...
            enable_hard_fails
...
```

Press “Ctrl + O”, “Enter”, and “Ctrl + X” to save and close the `Caddyfile`.

Install CrowdSec Agent on Caddy Server

The CrowdSec bouncer is what will block connections to services behind the reverse proxy, but the Caddy server will need a CrowdSec agent installed so it can run the parsers and scenarios on the server. The agent sends the information to the LAPI on OPNsense to make decisions on blocking content. For a single CrowdSec instance, this usually occurs on the same system, but in a multi-server configuration, the CrowdSec agent and bouncer communicates with the LAPI on a different server (in this case, OPNsense).

Install CrowdSec using the following commands. Basically the next step is following the CrowdSec installation guide. It is very simple to install.

```
curl -s https://packagecloud.io/install/repositories/crowdsec/crowdsec/script.deb.sh | sudo bash
sudo apt install crowdsec
```

The CrowdSec agent needs to be registered with OPNsense CrowdSec LAPI.

```
sudo cscli lapi register -u http://192.168.1.1:8080
```

Copy the default CrowdSec `systemd` file from `/lib/systemd/system` to `/etc/systemd/system` so customizations can be made to the service file.

```
sudo cp /lib/systemd/system/crowdsec.service /etc/systemd/system/crowdsec.service
```

Edit the `/etc/systemd/system/crowdsec.service` to add the `-no-api` flag to the end of the `ExecStart` command. This disables the LAPI on the Caddy server since it is not needed because the LAPI on OPNsense will be used instead (see CrowdSec’s multi-server configuration example).

```
[Unit]
Description=Crowdsec agent
After=syslog.target network.target remote-fs.target nss-lookup.target
```

```
[Service]
Type=notify
Environment=LC_ALL=C LANG=C
ExecStartPre=/usr/bin/crowdsec -c /etc/crowdsec/config.yaml -t -error
ExecStart=/usr/bin/crowdsec -c /etc/crowdsec/config.yaml -no-api
#ExecStartPost=/bin/sleep 0.1
ExecReload=/bin/kill -HUP $MAINPID
Restart=always
RestartSec=60

[Install]
WantedBy=multi-user.target
```

You will need to reload the `systemd` service since changes to the service was made.

```
sudo systemctl daemon-reload
```

Do not reload the CrowdSec service just yet because with the LAPI on the Caddy server disabled, CrowdSec will error on startup until you have validated the Caddy machine on OPNsense. CrowdSec will be unable to connect to the LAPI on OPNsense until validation occurs. When there is no reachable LAPI, the CrowdSec agent will fail to load.

Validate the Caddy Machine in OPNsense

While logged into OPNsense via SSH or the console, list the machines which have been registered or requesting to be registered:

```
sudo cscli machines list
```

You should see similar output to below. Notice the status of the machine `02a3nfadce4ez4b19zh582e0f68f72a4CX4EzFJ2Th4PNkj1` shows the “No” symbol under “Status”.

```
────────────────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
 Name                                                IP Address       Last Update
Status   Version                   Auth Type   Last Heartbeat
────────────────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
 localhost                                           192.168.1.1      2024-03-11T14:11:50Z   ✔️
       v1.6.0-freebsd-4b8e6cd7   password    21s
 02a3nfadce4ez4b19zh582e0f68f72a4CX4EzFJ2Th4PNkj1   192.168.1.2    2024-03-11T13:55:29Z   🚫
```

```
                                password     ⚠ 16m42s
──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────
```

To validate the machine, you can enter the following command.

```
sudo cscli machines validate 02a3nfadce4ez4b19zh582e0f68f72a4CX4EzFJ2Th4PNkj1
```

Add Collection(s) to CrowdSec Agent on the Caddy Server

To add extra Caddy-specific parsers, you can add the following collection to your CrowdSec installation on your Caddy server. The Caddy collection includes a Caddy log parser and basic HTTP protections.

While the Caddy log parser may only be beneficial if you are using Caddy as a web server instead of a reverse proxy, the included basic HTTP protections should be helpful to protect web apps that are behind the reverse proxy.

```
sudo cscli collections install crowdsecurity/caddy
```

If you are hosting a public service (which great care must be taken to address security), it may not be a bad idea to also include the HTTP DoS collection to help detect denial of service attacks.

Of course, you should test this does not interfere with the normal operation of your app/service. I also do not know if this would be beneficial if you are using a Cloudflare proxy which includes DDoS protection.

```
sudo cscli collections install crowdsecurity/http-dos
```

You may now finally restart the CrowdSec agent on the Caddy server.

```
sudo systemctl reload crowdsec
```

After reloading CrowdSec, you should check if it is running properly.

```
sudo systemctl status crowdsec
```