5. Configurer un serveur Web

- Apache2
  - Mise en place d'Apache2
  - Les hôtes virtuels
  - Créer un hôte virtuel
  - Activer le HTTPS
  - Mettre un site en HTTPS
- Certbot
  - Les commandes Certbot
- Nginx
  - Installation de Nginx
  - Protéger un site par mot de passe avec l'authentification de base
  - Rediriger un nom domaine sur un container

Apache2

Mise en place d'Apache2

Installation

```
sudo apt install apache2
```

Lancer le service

```
sudo service apache2 start
```

Pour vérifier que Apache fonctionne, entrer l'adresse IP du serveur dans un navigateur http://YOUR_IP_OR_DOMAIN/ et la page par défaut d'Apache doit être visible :

page-defaut-ubuntu2.png

Le dossier Apache2

Il est important de comprendre comment Apache2 fonctionne une fois installé.

Tout d'abord toute la configuration d'Apache2 se trouve à l'arborescence suivante : `/etc/apache2`

Dans cette arborescence il y a 3 fichiers de configuration standards d'Apache2 et des dossiers de conf.

- `apache2.conf`
- `envvars`
- `ports.conf`

> les fichiers de configuration globale `apache2.conf`, `envars` et `ports.conf` n'ont pas à être modifiés. Toute la configuration devrait se faire dans les sous dossiers `xxx-available`

- `sites-available` contient les fichiers de configuration des **sites disponibles**
- `sites-enabled` contient des liens symboliques vers les configurations, dans `site-available` . Chaque configuration présente dans ce dossier correspondant aux **sites activés**.
- `conf-available` contient les fichiers de configuration des **autres services** disponibles

- `conf-enabled` contient des liens symboliques vers les configurations, dans `conf-available`, des **services activés**
- `mods-available` contient les fichiers de configuration des **modules d'Apache disponibles**
- `mods-enabled` contient des liens symboliques vers les configurations, dans `mods-available`, des **modules activés**

Pour activer ou désactiver les différentes configurations présentent dans les dossiers `xxx-available` ou `xxx-enabled` il existe une commande permettant de créer ou enlever le lien symbolique directement dans le bon dossier.

```
sudo a2ensite <filename> # [configuration d'un site à activer]
sudo a2dissite <filename> # [configuration d'un site à désactiver]

sudo a2enconf <confname> # [configuration d'un service à activer]
sudo a2disconf <confname> # [configuration d'un service à désactiver]

sudo a2enmod <modname> # [configuration d'un module à activer]
sudo a2dismod <modname> # [configuration d'un module à désactiver]
```

Les hôtes virtuels

Avec Apache, chaque site ou application web correspond en principe à un hôte virtuel (***VirtualHost*** en anglais).

> Chaque hôte virtuel est défini par un fichier de configuration indépendant, qu'on trouve ou qu'on créé dans le répertoire `/etc/apache2/sites-available/`.

Par défaut

Dans le dossier `sites-available/` il existe 2 configurations par défaut.

1) Le premier *VirtualHost* est défini dans le fichier `/etc/apache2/sites-available/000-default.conf`. Voici son contenu sans les commentaires :

000-default.conf => Il correspond a une configuration par défaut pour un site en HTTP (port 80).

```
<VirtualHost *:80>
	ServerAdmin webmaster@localhost
	DocumentRoot /var/www/html
	ErrorLog ${APACHE_LOG_DIR}/error.log
	CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

2) Le deuxième est défini dans le fichier `/etc/apache2/sites-available/default-ssl.conf`. Voici son contenu sans les commentaires :

default-ssl.conf => Correspond a une configuration par défaut pour un site en HTTPS (port 443).

```
<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerAdmin webmaster@localhost


                DocumentRoot /var/www/html


                ErrorLog ${APACHE_LOG_DIR}/error.log
```

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

                #   SSL Engine Switch:
                #   Enable/Disable SSL for this virtual host.
                SSLEngine on
                SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
                SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

                <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                SSLOptions +StdEnvVars
                </FilesMatch>
                <Directory /usr/lib/cgi-bin>
                                SSLOptions +StdEnvVars
                </Directory>

        </VirtualHost>
</IfModule>
```

Créer un hôte virtuel

Création d'un hôte virtuel

> Apache recommande de créer un fichier de configuration dans lequel est défini un hôte virtuel pour chaque site ou application web dans le répertoire `/etc/apache2/sites-available/`.

Chaque hôte virtuel peut être appelé en fonction d'un nom de domaine ou sous-domaine, c'est la configuration la plus courante. Mais on peut également définir un numéro de port particulier, ou une adresse IP particulière (si le serveur en possède plusieurs) pour laquelle on affichera tel ou tel contenu web.

Chaque hôte virtuel ayant son fichier de configuration dédié, pour s'y repérer on peut le nommer par le nom de domaine auquel il correspond, suivi de l'extension `.conf`. Pour un nom de domaine *example.com* on créera donc un fichier `/etc/apache2/sites-available/example.com.conf`.

```
sudo nano /etc/apache2/sites-available/exemple.com.conf
```

Voici un exemple de contenu pour ce fichier :

example.com.conf

```
<VirtualHost *:80>
	ServerName example.com
	ServerAlias www.example.com
	DocumentRoot "/var/www/example"
	<Directory "/var/www/example">
		Options +FollowSymLinks
		AllowOverride all
		Require all granted
	</Directory>
	ErrorLog /var/log/apache2/error.example.com.log
	CustomLog /var/log/apache2/access.example.com.log combined
</VirtualHost>
```

Explication du fichier de configuration :

directive	description

`<VirtualHost *:80>`	On accepte les connexions sur n'importe quelle IP du serveur (`*`) sur le port 80.
`ServerName example.com`	Cet hôte virtuel sera seulement appelé pour le nom de domaine *example.com*…
`ServerAlias www.example.com`	…ainsi que pour le sous-domaine *www.example.com*. On peut spécifier ici d'autres noms de domaine en les séparant par un espace. On peut aussi utiliser **.example.com* pour inclure tous les sous-domaines.
`DocumentRoot "/var/www/example"`	On placera les fichiers du site dans le répertoire `/var/www/example`.
`<Directory "/var/www/example">`	On spécifie dans cette section des règles pour le répertoire `/var/www/example` sous cet hôte virtuel.
`Options +FollowSymLinks`	Apache suivra les liens symboliques qu'il trouvera dans ce répertoire (et ses descendants).
`AllowOverride all`	On pourra inclure une configuration personnalisée via un fichier .htaccess.
`Require all granted`	Tous les visiteurs pourront accéder au contenu de ce répertoire. Voir la documentation officielle pour modifier ce comportement. Pour des raisons de sécurité ou de privacité on peut par exemple limiter l'accès au serveur à seulement une ou certaines adresses IP avec une directive du type `Require ip 192.168.1.10`.
`ErrorLog /var/log/apache2/error.example.com.log` `CustomLog /var/log/apache2/access.example.com.log combined`	Il est pratique d'avoir des logs séparés pour chaque hôte virtuel, afin de ne pas mélanger toutes les informations.

Vérifier la configuration

Avec Apache2 il est possible de vérifier si la configuration des hôtes virtuels sont correctement écrite. Il suffit d’exécuter la commande suivante :

```
sudo apachectl configtest
```

Dans le cas ou tout est bon l'invite de commande devra vous retourner le message suivant : `Syntax OK`

Déployer le nouveau site

Après avoir l'avoir créée l'hôte virtuel, il faut activer cette configuration avec la commande `sudo a2ensite <hostname>`.

```
sudo a2ensite example.com
```

On recharge ensuite la configuration d'Apache :

```
sudo systemctl reload apache2
```

Activer le HTTPS

Le Https

HTTPS permet de chiffrer les communications entre le navigateur et Apache au moyen du protocole SSL/TLS, et de garantir l'authenticité de votre serveur (au moyen d'un certificat). Cela empêche qui que ce soit de récupérer ("sniffer") des informations sensibles (tels que des mots de passes) en particulier lorsqu'on se connecte depuis un réseau public, ou empêche un autre serveur de se faire passer pour le vôtre.

Il n'est ni nécessaire, ni faisable de mettre en place HTTPS avec un certificat valide sur un serveur de développement local.

Pour rendre disponible les sites de manière sécurisée via HTTPS avec des certificats valides, **la solution la plus simple est d'utiliser l'outil Certbot de Let's Encrypt.**

Activation du module

Pour que le **protocole TLS** (successeur du **SSL**) puisse fonctionner avec Apache2, il faut activer le module *ssl* avec la commande :

```
sudo a2enmod ssl
```

puis recharger la configuration d'Apache 2 :

```
sudo systemctl reload apache2
```

Installation de Certbot

Pour installer Certbot :

```
sudo apt install python3-certbot-apache
```

Mettre un site en HTTPS

Pour mettre un site en HTTPS il faut avoir préalablement effectuer les étapes dans la page précédente.

Utilisation de Certbot

Certbot permet de générer tous les certificats et d'adapter les configurations d'Apache pour tous les noms de domaine associés aux hôtes virtuels au moyen d'une seule commande :

```
sudo certbot --apache
```

Lors de l'opération le script nous invite à cocher les domaines pour lesquels on souhaite obtenir les certificats et à choisir de forcer l'usage de HTTPS ou pas. Pour des raisons de sécurité, c'est généralement une très bonne idée de forcer HTTPS.

Après cette opération les sites devraient être accessibles en HTTPS de manière sécurisée, sans que les navigateurs affichent de message d'alerte.

> Grâce à l'option `–apache`, Certbot s'occupe automatiquement de créer des fichiers de configuration de la forme `/etc/apache2/sites-available/example.com-le-ssl.conf` pour les hôtes virtuels en HTTPS sur le port 443 et de les activer (`-le-ssl` pour *Let's Encrypt*).

Renouvellement automatique

Pour information c'est la commande `certbot renew` qui permet de renouveler les certificats très simplement, mais Certbot créé automatiquement une tâche cron à cet effet dans `/etc/cron.d/certbot`. Il est également créé un « *timer* » systemd qui fait la même chose (/lib/systemd/system/certbot.timer et certbot.service).

Le script est lancé automatiquement toutes les 12 heures, mais les certificats ne seront renouvelés que si nécessaire. En principe il n'y a donc rien à faire.

Vérifier le HTTPS

Il existe des sites tel que Qualys pour vérifier la sécurité SSL du site.

Lien : https://www.ssllabs.com/ssltest/

On remplis le “hostname” avec notre nom de domaine : exemple.com

Certbot

Les commandes Certbot

Créer un certificat

```
sudo certbot --apache
```

Renouveler les certificats

```
sudo certbot renew
```

Supprimer un ancien certificat

```
sudo certbot delete
```

Afficher les certificats

```
sudo certbot certificates
```

Nginx

Installation de Nginx

Installation

```
sudo apt install nginx
```

Lancer le service

```
sudo service nginx start
```

Pour vérifier que Nginx fonctionne, entrer l’adresse IP du serveur dans un navigateur http://YOUR_IP_OR_DOMAIN/ et la page par défaut de Nginx doit être visible :

image.png

Le dossier Nginx

Il est important de comprendre comment Nginx fonctionne une fois installé.

Tout d'abord toute la configuration de Nginx se trouve à l'arborescence suivante : `/etc/nginx`

Dans cette arborescence il y a les dossiers de configuration des hosts :

- `sites-available` contient les fichiers de configuration des **sites disponibles**
- `sites-enabled` contient des liens symboliques vers les configurations, dans `site-available` . Chaque configuration présente dans ce dossier correspondant aux **sites activés**.

Protéger un site par mot de passe avec l'authentification de base

Prérequis

Tout d'abord pour utiliser la "basic authentification" il faut installer un package appartenant à Apache :

```
sudo apt install apache2-utils
```

Il est nécessaire d'installer ce package car nous avons besoin de la commande `htpasswd` qui permet de créer un fichier `.htpasswd` qui contiendra tous les utilisateurs et leurs mot de passe.

Création du fichier .htpasswd et d'un utilisateur

```
sudo htpasswd -c /path/to/file/.htpasswd <user>


# Exemple
sudo htpasswd -c /etc/nginx/.htpasswd marc
New password:
Re-type new password:
Adding password for user marc
```

L'option `-c` permet de créé le *fichier-mots-de-passe*. Si *fichier-mots-de-passe* existe déjà, il est réécrit et tronqué.

Nginx

Rediriger un nom domaine sur un container

Se placer dans le répertoire :

`cd /etc/nginx/conf.d`

Puis créer un fichier : <nom-de-mon-service>.conf

exemple :

`nano msn.conf`

Puis insérer le contenu suivant en modifiant les variables suivantes :

```
server {
    server_name msn.chiken.ovh;

    location / {
        proxy_pass http://localhost:3001; # Le port exposé par ton Docker
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    listen 80;
    listen [::]:80;
}
```

Puis redémarrer nginx :

`sudo systemctl restart nginx`

Pour mettre le service en HTTPS il faut exécuter la commande suivante :

`certbot --nginx`

Puis entrer le numéro du service correspondant.