

Mise en production d'une application Symfony

Ibrahim ALAME

14/02/2023

1 Les différentes étapes pour la mise en production

1.1 Rappel sur l'environnement de développement

Pour développer notre application nous avons cet environnement :

PHP : son interpréteur (qui compile et exécute le code PHP) et son serveur Web de développement (utilisé lorsque nous faisons symfony serve).

MySQL : nous avons installé une base de données locale MySQL que nous utilisons comme un service pour qu'elle s'exécute en continue.

Symfony : le CLI nous a permis d'obtenir des certificats TLS auto-signés. Ces certificats permettent d'utiliser le protocole HTTPS en local et seuls les navigateurs installés localement les acceptent. Ils sont donc utiles uniquement en développement.

Webpack Encore : Webpack nous permet principalement en développement de compiler le Sass en CSS, le TypeScript en JavaScript, de créer des bundles et les bons imports dans les templates Twig. Nous l'avons configuré pour utiliser le hot reload (détection des changements dans les assets, compilation et rechargement automatique).

Mailtrap : pour les emails en développement, nous utilisons le service externe Mailtrap qui permettait d'envoyer et de recevoir des emails et de les tester instantanément.

1.2 Environnement de production

OVH Cloud : nous utiliserons OVH à la fois pour le nom de domaine (serveurs DNS OVH), l'hébergement (Virtual Private Server, un serveur privé virtuel dans le cloud) et pour le serveur de messagerie (MTA). Ce sont trois services différents que nous devrons configurer.

MySQL : nous installerons directement la base de données sur le VPS.

Symfony : nous installerons bien sûr Symfony pour exécuter notre application serveur.

NGINX : est le serveur Web qui sera le point d'entrée de l'application pour les requêtes HTTP. NGINX transmettra toutes les requêtes visant des fichiers PHP à PHP-FPM.

PHP-FPM : PHP - FastCGI Process Manager permet d'utiliser le protocole FastCGI pour la communication entre un serveur Web (ici NGINX) et PHP. PHP-FPM est un service, c'est-à-dire un processus exécuté en arrière plan, qui va gérer des processus enfants. Ces processus enfants vont traiter les requêtes passées par le serveur Web. Avec PHP-FPM, une seule instance de l'interpréteur PHP, avec tous les modules et les configurations est chargée.

Vous pouvez le voir comme une sorte de serveur. Elle ne reçoit que les fichiers PHP à traiter suivant la requête. Ainsi, les ressources (RAM, disque, CPU) peuvent être réutilisées pour toutes les requêtes HTTP, elles sont partagées et elles ne sont plus dédiées à chaque requête.

Certbot : nous utiliserons l'autorité de certification Let's Encrypt et le client certbot qui permet d'automatiser l'obtention de certificats TLS pour utiliser le protocole HTTPS en production.

Sendgrid : nous utiliserons le service tiers Sendgrid pour gérer l'envoi des emails en production.

2 Création du VPS et configuration SSH

2.1 Création d'un VPS chez OVH

Nous allons utiliser OVH, le service français d'hébergement leader en Europe. Vous pouvez ouvrir un VPS (Virtual Private Server) chez n'importe quel hébergeur connu : AWS, Azure, Google Cloud etc.

1. rendez vous sur le site OVH et créez un compte.
2. cliquez sur l'onglet Bare Metal Cloud.
3. cliquez sur Commander > Serveurs privés virtuels. (Ce sont vraiment des petites configurations, mais parfaites pour démarrer).
4. choisissez un des abonnements suivant vos besoins.
5. choisissez Linux.
6. choisissez Système d'exploitation seul.
7. prenez la dernière version d'Ubuntu Server. Tous nos chapitres seront sur Ubuntu. Mais si vous êtes habitué au système Linux nous recommandons CentOS qui est réputé pour sa sécurité et sa stabilité.

Il faut compter environ 10 à 15 minutes pour que votre serveur VPS apparaisse. Pendant ce temps là vous pouvez acheter votre nom de domaine.

2.2 Achat du nom de domaine

1. Allez dans Web Cloud.
2. Noms de domaine.
3. Ne prenez pas d'hébergement car nous allons utiliser le VPS que nous venons de prendre.

Normalement vous n'aurez besoin de prendre aucune option.

2.3 Connexion en SSH au serveur VPS

Nous allons maintenant nous connecter à notre serveur en ssh afin notamment de pouvoir pull facilement notre projet depuis Github ou Gitlab.

2.3.1 Qu'est-ce que SSH ?

Secure SHell (SSH) est un protocole de communication sécurisé permettant d'ouvrir un shell sur une machine distante.

Par défaut, SSH utilise le port 22.

Avec SSH il est possible de se connecter avec utilisateur / mot de passe ou avec une clé privée (il faut au préalable avoir enregistré la clé publique correspondante sur la machine pour qu'elle accepte la connexion).

Un client ssh est normalement installé sur votre machine, par exemple sur Linux c'est souvent OpenSSH.

Vous pouvez regarder si vous avez un client en faisant `ssh -V`.

Si vous n'en avez pas, téléchargez par exemple putty.

2.3.2 Connexion au serveur

Pour vous connecter il suffit de faire

```
ssh UTILISATEUR@IP_DU_SERVEUR
```

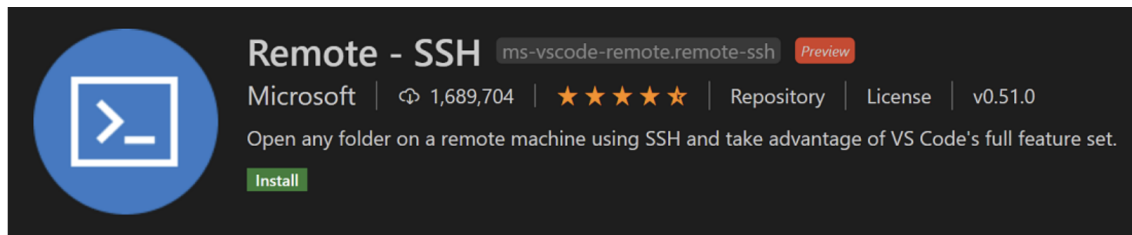
Remplacez par l'utilisateur (souvent root ou ubuntu) et l'ip de votre serveur VPS.

Il vous sera demandé le mot de passe qui vous a été envoyé par email par OVH.

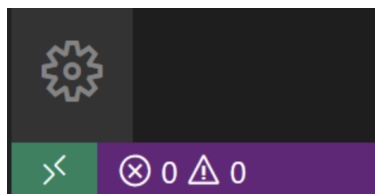
Vous êtes maintenant connecté sur le serveur !

2.4 Installer l'extension VS Code Remote - SSH

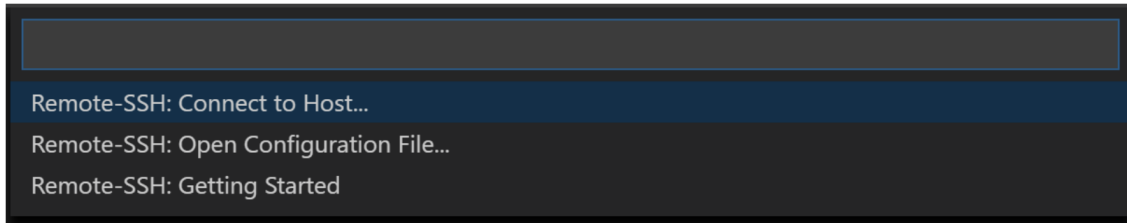
L'extension VS Code Remote SSH permet de se connecter sur un hôte distant en SSH et de pouvoir développer avec VS Code directement sur cet hôte distant !



Une fois que vous aurez installé l'extension, vous aurez une nouvelle icône en bas à gauche :



Si vous cliquez sur ce bouton vous aurez le menu de l'extension qui s'ouvrira :



2.5 Se connecter à un hôte distant

Après avoir sélectionné Remote-SSH : Connect to Host. . . , entrez l'hôte sur lequel vous souhaitez vous connecter en SSH :

```
utilisateur@hôte
```

Une fois connecté au serveur en SSH, le nom de celui-ci apparaîtra en bas à gauche.

Vous pouvez ensuite ouvrir n'importe quel dossier sur le serveur distant en allant dans File > Open. . . , exactement comme en local !

Si vous ouvrez un nouveau terminal (Terminal > New Terminal), celui-ci sera connecté au shell de votre hôte distant automatiquement en SSH !

Pour vous déconnecter de l'hôte il suffit de faire File > Close Remote Connection.

3 Configuration de la zone DNS

3.1 Configuration DNS

Nous avons maintenant un serveur virtuel privé qui possède une IPv4 et une IPv6. Notre objectif va tout simplement de configurer les serveurs DNS de OVH pour faire pointer notre nom de domaine vers l'IP de notre serveur. Vous pouvez donc aller dans Web Cloud puis cliquez sur votre serveur.

- Ensuite, copiez l'IPv4 ou l'IPv6 de votre serveur.
- Allez ensuite dans Web Cloud puis cliquez sur votre domaine dans la colonne de gauche.
- Cliquez une nouvelle fois sur le nom de votre domaine.
- Allez ensuite dans ZONE DNS.
- Si vous avez copié l'IPv4 cherchez les deux entrées de Type A et copiez l'IP.
- Si vous avez copié l'IPv6 cliquez sur ajouter une entrée puis choisissez Type AAAA et copiez l'IP.
- Ensuite confirmez le changement d'IP.

3.2 Les entrées A et AAAA dans la zone DNS

Les entrées DNS de type A permettent de relier un nom de domaine ou un sous-domaine à l'adresse IPv4 d'un serveur.

Les entrées DNS de type AAAA permettent de relier un nom de domaine ou un sous-domaine à l'adresse IPv6 d'un serveur.

4 Installation de MySQL sur le serveur et accès root en ssh

4.1 Mise à jour du serveur

Une fois connecté en ssh sur le serveur VPS, mettez le à jour en tapant les commandes suivantes :

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get autoremove
sudo reboot
```

Nous vérifions les paquets à mettre à jour, puis nous les mettons tous à jour, nous enlevons les paquets obsolètes et enfin nous redémarrons.

Reconnectez-vous en ssh après le redémarrage du serveur au bout d'une minute.

4.2 Installation de MySQL

Ouvrez un nouveau terminal sur votre distribution famille Debian (Ubuntu, Mint etc). Entrez :

```
sudo apt install mysql-server
```

4.3 Configuration du serveur MySQL

Lancez la commande de configuration :

```
sudo mysql_secure_installation
```

Si vous avez l'erreur Failed! Error : SET PASSWORD has no significance for user 'root'@'localhost'..., définissez un mot de passe pour l'utilisateur root en faisant d'abord `sudo mysql` puis faites `ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password by 'UN_MOT_DE_PASSE_SECURISE'`;

Tapez Y puis entrée à la première question et 1 à la seconde.

Choisissez un mot de passe pour l'utilisateur root (administrateur de la base de données).

Tapez Y puis entrée à toutes les questions. Cela supprimera les utilisateurs anonymes et la base de données de test et cela désactivera les connexions root à distance.

Vérifiez la connexion en faisant :

```
mysql -u utilisateur -p motdepasse
```

4.4 Création d'un utilisateur sur MySQL

Sur le VPS, connectez-vous au serveur MySQL en entrant :

```
sudo mysql
```

Créez un utilisateur :

```
CREATE USER 'utilisateur'@'%' IDENTIFIED BY 'motdepasse';
```

Remplacez bien sûr utilisateur et motdepasse.
Accordez-lui tous les droits :

```
GRANT ALL PRIVILEGES ON *.* TO 'utilisateur'@'%' WITH GRANT OPTION;
```

Quittez en tapant exit puis entrée.
Vérifiez la connexion avec le nouvel utilisateur en faisant :

```
mysql -u utilisateur -p motdepasse
```

4.5 Connexion root en ssh au VPS

Par défaut sur OVH on ne peut pas se connecter en root en ssh. Il faut l'autoriser manuellement.
Pour ce faire, connectez-vous en ssh avec l'utilisateur ubuntu.
Editez avec par exemple nano le fichier de configuration ssh du serveur :

```
sudo nano /etc/ssh/sshd_config
```

Trouvez la ligne #PermitRootLogin et remplacez simplement par :

```
PermitRootLogin yes
```

Sauvegardez (Ctrl + s) et quittez (Ctrl + x).
Redémarrez le service ssh :

```
sudo service ssh restart
```

Configurez le mot de passe root (choisissez un mot de passe très sécurisé) :

```
sudo passwd root
```

Vous pouvez maintenant vous connecter en root en ssh.

4.6 Modifiez la configuration de MySQL

Sur le serveur VPS avec une connexion root, modifiez le fichier /etc/mysql/mysql.conf.d/mysqld.cnf :

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Mettez la configuration suivante qui permet d'écouter sur toutes les interfaces IPv4 :

```
bind-address          = 0.0.0.0
mysqlx-bind-address   = 0.0.0.0
```

Redémarrez MySQL :

```
service mysql restart
```

4.7 Connexion Workbench

Créez une nouvelle connexion dans Workbench.
Dans connection name mettez par exemple production.
Dans Hostname mettez l'adresse IPv4 de votre serveur.
Dans username mettez le nom de votre utilisateur.
Cliquez sur store in key chain et entrez votre mot de passe.
Cliquez sur Test Connection. Si vous avez Successfully... cliquez sur Ok.
Cliquez enfin sur la connexion.

5 Installation de PHP, Composer et Symfony

5.1 Installation de PHP-FPM

Ajoutez d'abord le répertoire ppa :

```
sudo apt install software-properties-common  
sudo add-apt-repository ppa:ondrej/php  
sudo apt update
```

Installez ensuite PHP-FPM :

```
sudo apt install php8.2-fpm
```

Vérifiez l'installation :

```
sudo service php8.2-fpm status
```

5.2 Installation de Composer

Pour installer Composer, rendez-vous sur cette page et copiez les lignes :

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === '756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d37e9') {  
php composer-setup.php  
php -r "unlink('composer-setup.php');"
}
```

Déplacez le fichier de Composer dans le dossier des exécutables :

```
sudo mv composer.phar /usr/bin/composer
```

5.3 Installation du CLI Symfony

Un CLI permet d'interagir avec un programme en ligne de commande.
Cela permet ici d'interagir avec Symfony en utilisant un terminal.
Tapez simplement dans un terminal :

```
wget https://get.symfony.com/cli/installer -O - | bash
```

Nous vous recommandons de l'installer en global sur le système en faisant ensuite la commande proposée en dernier (Or install it globally on your system) :

```
sudo mv /home/AMODIFIER/.symfony/bin/symfony /usr/local/bin/symfony
```

Utilisez la commande proposée et pas celle-ci car le chemin sera différent sur votre système.
Relancez un nouveau terminal.

5.4 Installation des dépendances de Symfony

Dans un nouveau terminal tapez :

```
symfony check:requirements
```

Vous pourriez avoir des erreurs pour des dépendances obligatoires manquantes, par exemple :

```
14:57:53 ✖ erwan:~$ symfony check:requirements

Symfony Requirements Checker
~~~~~

> PHP is using the following php.ini file:
/etc/php/8.0/cli/php.ini

> Checking Symfony requirements:

.....E.....W.....

[ERROR]
Your system is not ready to run Symfony projects

Fix the following mandatory requirements
~~~~~

* simplexml_import_dom() must be available
  > Install and enable the SimpleXML extension.
```

Dans ce cas installez les dépendances requises, ici par exemple :


```
sudo apt-get install php-xml php-intl php-mysql
```

Installez également en plus mbstring :

```
sudo apt install php-mbstring
```

Refaites :

```
symfony check:requirements
```

Il faut que vous ayez :

```
15:01:57 ✓ erwan:~/Bureau$ symfony check:requirements

Symfony Requirements Checker
~~~~~

> PHP is using the following php.ini file:
/etc/php/8.0/cli/php.ini

> Checking Symfony requirements:

.....

[OK]
Your system is ready to run Symfony projects

Note The command console can use a different php.ini file
~~~~~ than the one used by your web server.
Please check that both the console and the web server
are using the same PHP version and configuration.
```

6 Préparation du projet et Github

6.1 Création d'un compte Github ou Gitlab et configuration Git

Créez-vous un compte Github ou Gitlab et créez un répertoire public ou privé.

Sur Github : bouton vert New dans la colonne Repositories.

Sur Gitlab : Projects puis New Project.

6.1.1 Utilisation d'une clé ssh sur Github

Commencez par générer une paire de clés sur votre machine en local :

```
$ ssh-keygen -t ed25519 -C "votre-adresse-email@example.com"
```

Pressez entrée à toutes les questions.
Affichez et copiez la clé publique générée :

```
$ cat ~/.ssh/id_ed25519.pub
```

Allez sur Github et connectez-vous.
Cliquez en haut à droite sur votre avatar, puis cliquez sur settings puis SSH and GPG keys.
Cliquez ensuite sur Add Ssh key.
Collez la clé publique que nous avons affichée avec la commande ci-dessous.

6.1.2 Initialiser Git dans votre projet

Allez sur Github et créez un répertoire.
Sélectionnez SSH en dessous de Quick Setup.
Vous aurez alors toutes les commandes à taper dans un terminal au niveau de la racine de votre projet. Par exemple :

```
cd VOTRE_PROJET
git init
git remote add origin REPERTOIRE_DISTANT
git add .
git commit -m "Initial commit"
git push -u origin master
```

Attention de bien supprimer le mot de passe de votre base de données locale si vous utilisez un "vrai" mot de passe et non de test.

git init : va initialiser un répertoire git. Ce qui vous permettra de push et de pull sur un répertoire distant.

git remote add : permet de définir le répertoire distant (qui est sur Gitlab ou sur Github).

git add . : permet d'ajouter tous les fichiers du répertoire sauf ceux dans .gitignore.

git commit -m "Premier commit" : permet de créer un commit et de mettre le message "Premier commit".

git push -u origin master : permet de push la branche locale master sur le répertoire distant, origin. L'option u permet de définir le répertoire central, vous pourrez ensuite git push et git pull sans argument.

6.2 Récupération du répertoire sur le serveur

Sur le VPS, en vous connectant en ssh sur l'utilisateur ubuntu, récupérez le répertoire depuis Github ou Gitlab :

```
cd /home/ubuntu
git clone https://github.com/dymafr/symfony-chapitre21-mise-en-production
```

Bien sûr utilisez le lien de votre répertoire.

6.3 Création des variables d'environnement sur le serveur

Sur le serveur, en utilisant l'extension Remote Host de VS Code, naviguez dans votre projet dans /home/ubuntu et créez un fichier .env.local :

```
APP_ENV=prod
DATABASE_URL="mysql://VOTRE_UTILISATEUR:VOTRE_MOT_DE_PASSE@127.0.0.1:3306/quora?serverVersion=8.0.26"
APP_SECRET=UN_SECRET
```

Remplacez l'utilisateur et le mot de passe.

Remplacez UN_SECRET par une chaîne de caractères aléatoires. Voici comment en générer une de manière sécurisée :

```
php -r 'echo(bin2hex(random_bytes(20)));'
```

6.4 Installation des dépendances Symfony

Toujours sur votre serveur, toujours en connexion ssh et toujours dans le dossier /home/ubuntu/VOTRE_PROJET, commencez par installer php-zip :

```
apt install php-zip unzip
```

Retournez sur l'utilisateur ubuntu :

```
su ubuntu
```

Puis installez les dépendances Symfony de production :

```
symfony composer install --no-dev --optimize-autoloader
```

Lancez la commande :

```
composer dump-env prod
```

Cela va générer le fichier des variables d'environnement pour la production en PHP ce qui est plus rapide en lecture.

7 Installation de Node.js, Yarn et NGINX

7.1 Installation de Node.js

7.1.1 Installer nvm :

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
```

Collez les commandes suivantes :

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && . "$NVM_DIR/bash_completion"
```

Installer Node.js LTS :

```
nvm install --lts
```

7.2 Installation de yarn

Installer en global yarn :

```
npm i -g yarn
```

Installer ensuite les dépendances du projet :

```
yarn
```

Lancez un build des assets :

```
yarn build
```

7.3 Installation de NGINX

Nous allons commencer par installer NGINX :

```
sudo apt install nginx
```

8 Configuration de NGINX

8.1 Création de la configuration NGINX

Dans le répertoire du projet, sur votre machine locale, créez un dossier nginx.

Dans le dossier nginx créez le fichier de configuration wonder.conf.

Installez l'extension NGINX Configuration sur VS Code.

Configuration du serveur NGINX Nous configurons le serveur :

```
server {
    listen 80;
    listen [::]:80;

    # Remplacez dymawonder.fr par votre nom de domaine :
    server_name dymawonder.fr www.dymawonder.fr;
    # Remplacez symfony-chapitre21-mise-en-production par le nom du répertoire de votre projet :
    root /home/ubuntu/symfony-chapitre21-mise-en-production/public;
```

```

location / {
    try_files $uri /index.php$is_args$args;
}

location ~ ^/index.php(/|$) {
    fastcgi_pass unix:/run/php/php8.2-fpm.sock;
    fastcgi_split_path_info ^(.+\.php)(/.*)$;
    include fastcgi_params;

    fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
    fastcgi_param DOCUMENT_ROOT $realpath_root;
    internal;
}

location ~ \.php$ {
    return 404;
}

error_log /var/log/nginx/project_error.log;
access_log /var/log/nginx/project_access.log;
}

```

Effectuez les modifications indiquées en commentaire.
Ajoutez le fichier au suivi de Git :

```

git add nginx
git commit -m "nginx conf"
git push origin master

```

8.2 Utiliser notre configuration NGINX

Sur le VPS, depuis l'extension Remote host de VS Code, modifiez le fichier `/etc/nginx/nginx.conf` :
Modifiez les lignes suivantes à la fin du bloc `http` :

```

include /etc/nginx/conf.d/*.conf;
include /home/ubuntu/VOTRE_PROJET/nginx/*;

```

Remplacez `VOTRE_PROJET` par le nom de votre répertoire pour le projet sur le serveur.

8.3 Pull de la configuration sur le serveur

Sur le VPS, dans le dossier du projet, récupérez les changements :

```

git pull

```

Redémarrez nginx :

```
sudo service nginx reload
```

8.4 Permissions pour NGINX

Commencez par installer acl (Access Control List) :

```
sudo apt install acl
```

Trouvez ensuite l'utilisateur qui exécute nginx :

```
ps -aef |grep nginx
```

Si vous voulez en apprendre plus sur toutes les commandes Unix suivez le cours GNU/Linux.

Vous verrez normalement l'utilisateur www-data pour le worker process.

Nous devons donner les permissions à cet utilisateur d'écrire notamment dans var/log et var/cache. Ces dossiers sont nécessaires pour le cache de symfony.

Donnez les permissions sur le dossier /home/ubuntu :

```
sudo setfacl -R -m u:"www-data":rwX /home/ubuntu
sudo setfacl -dR -m u:"www-data":rwX /home/ubuntu
```

8.5 Création de la base de données

Toujours sur le serveur, créez la base de données de production :

```
symfony console doctrine:database:create
```

Faites une migration :

```
symfony console doctrine:migration:migrate
```

9 Mise en place du protocole HTTPS

9.1 Qu'est-ce que letsencrypt ?

Let's Encrypt est une autorité de certification en 2015. Elle fournit gratuitement des certificats X509 pour pouvoir utiliser le protocole TLS. Les principaux sponsors de cette initiative sont Cisco, Mozilla, EFF, OVH, Google, l'Internet Society et Facebook. Ils permettent à 150 millions de sites web de pouvoir servir gratuitement leur application en HTTPS. Ils ont développé un protocole ACME permettant de gérer les certificats de manière totalement automatisée (attribution, révocation et mise à jour) qui est standardisé par l'IETF depuis début 2019.

9.1.1 Validation d'un domaine par letsencrypt

Pour valider un domaine, il faut créer une paire de clés et prouver la possession du domaine.

Etape 1 : le serveur envoie une demande de certificat pour quelques domaines à letsencrypt et envoie la clé publique d'une paire de clés.

Etape 2 : letsencrypt demande au serveur un challenge permettant de prouver qu'il possède bien ces domaines. Par exemple, il va fournir un token et demander au serveur de le servir à une URL donnée. Il va également demander de signer ce fichier avec sa clé privée pour prouver qu'il a la clé privée correspondant à la clé publique envoyée à l'étape 1.

Etape 3 : letsencrypt vérifie que le token est bien placé sur le serveur. Il vérifie ensuite la signature avec la clé publique envoyée à l'étape 1.

Etape 4 : letsencrypt valide la paire de clés qui devient une paire de clés autorisés (authorized key pair) pour les domaines vérifiés.

9.1.2 Obtention d'un certificat

Une fois une paire de clés obtenus pour un ou plusieurs domaines, il est possible de les utiliser pour gérer les certificats.

Avec une paire de clés autorisés vous pouvez demander un certificat, le renouveler ou le révoquer.

Etape 1 - le serveur génère une nouvelle paire de clés.

Etape 2 - le serveur construit une CSR pour un ou plusieurs domaines qui sont inclus dans les domaines validés pour la paire de clés autorisée. Il inclut la nouvelle clé publique dans la CSR (c'est la clé publique qui sera incluse dans le certificat).

Etape 3 - le serveur signe la CSR avec la nouvelle clé privée correspondant à la clé publique. Il signe ensuite la CSR + la signature avec la clé privée de la paire de clés autorisée. (Il y a bien deux signatures du serveur avec deux clés privées différentes).

Etape 4 - letsencrypt va vérifier la première signature avec la clé publique autorisée, et la seconde signature avec la clé publique dans la CSR.

Etape 5 - letsencrypt retourne un certificat pour le ou les domaines demandés.

9.1.3 Les différents challenges possibles

HTTP-01 challenge : c'est le challenge que nous avons pris en exemple. Un token est placé sur votre serveur et servi à l'adresse `http://<YOUR_DOMAIN>/.well-known/acme-challenge/<TOKEN>`. Ce fichier est signé par une clé privée générée par le serveur, comme nous l'avons étudié. Ce challenge utilise forcément le port 80 pour la vérification du token signé.

DNS-01 challenge : pour ce challenge il faut modifier la zone DNS et ajouter un token dans une entrée TXT. L'avantage de ce challenge est qu'il permet l'obtention d'un certificat wildcard. C'est-à-dire pour tous les domaines `*.votre_domaine.fr`. Mais pour l'utiliser il faut pouvoir accéder à l'API de votre hébergeur afin de mettre à jour l'entrée TXT à chaque renouvellement du certificat. Cette méthode est plus complexe à mettre en œuvre.

Que diriez-vous si toutes les étapes vues précédemment étaient complètement automatisées ?

Génial, non ?

Grâce à letsencrypt et à son agent certbot c'est possible !

9.2 Qu'est-ce que certbot ?

Certbot est le client officiel de Let's Encrypt permettant d'utiliser le protocole ACME de manière automatisée.

Il permet ainsi de gérer ses certificats de manière vraiment rapide et efficace.

9.2.1 Installation de certbot

Connectez-vous en ssh à votre serveur.

Puis lancez l'installation de certbot :

```
sudo apt-get install certbot
```

9.2.2 Création d'un certificat avec certbot

Pour créer un certificat avec certbot, il suffit de faire :

```
sudo certbot certonly -d DOMAINE1 -d DOMAINE2 -d DOMAINE3
```

N'oubliez pas de couper NGINX avant :

```
service nginx stop
```

Il faut que le domaine passé match exactement l'URL racine de votre site.

Par exemple si l'utilisateur tape `www.votresite.fr` ou `votresite.fr` il faudra ajouter les deux domaines à la demande de certificat.

Ensuite sélectionnez l'option 1 : standalone. Il faut que votre application soit coupée car le port 80 doit être disponible pour le mini-serveur que va lancer certbot.

Ensuite, vous allez devoir créer un compte auprès de letsencrypt, vous devrez fournir l'email de l'administrateur du domaine (le vôtre probablement), vous devez ensuite accepter les conditions d'utilisation avec A et enfin indiquer si vous voulez partager votre email à une ONG pour la liberté de l'Internet (qui défend la neutralité du net entre autre).

9.3 Utilisation du certificat

certbot vous a automatiquement généré une clé privée et obtenu un certificat valide auprès de letsencrypt.

Les certificats se trouvent dans :

```
/etc/letsencrypt/live/NOM_DOMAINE
```

Vous avez besoin que de deux fichiers : `fullchain.pem` et `privkey.pem`.

Nous allons modifier le fichier `/nginx/wonder.conf` pour utiliser le certificat et pouvoir mettre en place HTTPS :


```

server {
    listen 80 default_server;
    listen [::]:80 default_server;
    # Remplacez dymawonder.fr par votre domaine :
    return 301 https://dymawonder.fr$request_uri;
}

server {
    listen 443 ssl default_server http2;
    listen [::]:443 ssl default_server http2;

    # Remplacez dymawonder.fr par votre domaine :
    ssl_certificate /etc/letsencrypt/live/dymawonder.fr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/dymawonder.fr/privkey.pem;

    # Remplacez dymawonder.fr par votre nom de domaine :
    server_name dymawonder.fr www.dymawonder.fr;
    # Remplacez symfony-chapitre21-mise-en-production par le nom du répertoire de votre projet :
    root /home/ubuntu/symfony-chapitre21-mise-en-production/public;

    location / {
        try_files $uri /index.php$is_args$args;
    }

    location ~ ^/index.php(/|$) {
        fastcgi_pass unix:/run/php/php8.2-fpm.sock;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;

        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        fastcgi_param DOCUMENT_ROOT $realpath_root;
        internal;
    }

    location ~ \.php$ {
        return 404;
    }

    error_log /var/log/nginx/project_error.log;
    access_log /var/log/nginx/project_access.log;
}

```

Veillez à bien modifier les valeurs passées aux directives `return`, `ssl_certificate`, `ssl_certificate_key`, `root` et `server_name` en fonction du nom de votre répertoire distant et du nom de votre domaine.

Enregistrez les modifications dans Git :

```
git add .
git commit -m "nginx https"
git push origin master
```

Sur le VPS, dans le dossier de votre projet, vous n'avez plus qu'à git pull :

```
git pull
```

Relancez ensuite NGINX :

```
service nginx start
```

9.4 Qu'est-ce qu'une tâche cron ?

cron est un programme disponible sur les systèmes Unix (comme Linux ou MacOS) et qui permet d'exécuter automatiquement des scripts à une date et une heure spécifiques.

cron est le diminutif de crontab qui veut dire chrono table. Il vient du grec ancien chronos qui signifie le temps.

9.5 Créer une tâche cron

Pour créer une tâche il suffit d'ouvrir la crontab qui est la liste de toutes les tâches cron et de l'éditer.

Pour ce faire il suffit de taper :

```
crontab -e
```

Vous devrez ensuite choisir un éditeur de texte. Choisissez par exemple nano qui est très bien pour ce que vous devrez faire.

La syntaxe est la suivante :

```
mm hh jj MMM JJJ tâche
```

mm : sont les minutes (0 à 59).

hh : sont les heures (0 à 23).

jj : sont les jours (1 à 31).

MMM : sont les mois (1 à 12).

JJJ : est le numéro du jour dans la semaine, avec 0 = dimanche (0 à 6). Il existe également des syntaxes particulières :

***** : signifie tous les. Par exemple :

```
* * * * * tâche
```

Permet d'exécuter la tâche toutes les minutes.

```
0 0 * * * tâche
```

Permet d'exécuter une tâche tous les jours à 00h00.

9.6 Lister les tâches cron

Chaque tâche cron doit être écrite sur une ligne.
Vous pouvez faire autant de tâches que vous voulez.
Pour lister les tâches définies :

```
crontab -l
```

9.7 Créer un cron pour le renouvellement automatique du certificat

Il est possible de renouveler automatiquement le certificat, qui n'est valide que 90 jours, avec une tâche cron :

```
crontab -e
```

Créez une ligne :

```
0 0 * * * certbot --webroot -w /home/ubuntu/VOTRE_PROJET/public renew --deploy-hook "service nginx reload"
```

Remplacez bien sûr VOTRE_PROJET par le nom du répertoire que vous avez choisi.

Cette tâche va essayer tous les jours à minuit de lancer certbot pour renouveler le certificat.

-webroot -w : permet de spécifier le chemin où placer le fichier pour le challenge. Pour rappel, certbot a besoin du port 80 pour le challenge HTTP. Il y a donc deux solutions : soit vous coupez votre serveur, vous faites le challenge puis vous le relancez, soit vous indiquez un chemin où votre serveur pourra servir le token signé. Ici, nous utilisons la deuxième solution qui est plus propre.

certbot décidera ensuite si le certificat est due for renewal (c'est-à-dire qu'il reste suffisamment peu de jours), et si c'est le cas, il effectuera la requête ACME pour renouveler le certificat et vous en aurez un nouveau !

-deploy-hook : est une hook, c'est-à-dire un déclencheur, qui lancera la commande uniquement lorsque le certificat sera renouvelé avec succès. Dans le cas contraire, la commande n'est pas lancée.

9.8 Ajouter la fonction Twig asset()

Dans le projet, en local, recherchez src=" dans VS Code.

Partout où il n'y a pas la fonction asset() ajoutez la.

Ensuite ajoutez les modifications dans Git :

```
git add .
git commit -m "fix asset images"
git push origin master
```

Sur le VPS, dans le dossier de votre projet, vous n'avez plus qu'à git pull :

```
git pull
```