42 ft_server

Projet d'administration système (automatisation de tâches d'installation et de configuration) + introduction à Docker. Nous devons rédiger le Dockerfile (~ image) qui nous permettra de build notre container. **Une image est en quelque sorte le 'blueprint' ou la spécification du container résultant du build.** Nous devons/pouvons également rédiger un script à exécuter une fois que le container est "run". Certaines commandes peuvent être mentionnées dans le Dockerfile ou dans le script (comme la création d'une clé et d'un certificat pour la partie SSL (HTTPS), l'etudiant(e) est donc a priori libre de choisir dans quel fichier il/elle souhaite les mentionner.

On nous demande d'installer la "**LEMP Stack**" + Wordpress + ~Docker. **LEMP : Linux - Nginx ("Engine x") - MySQL - PHP.** *Il s'agit de mettre en place le setup nécessaire pour pouvoir travailler sur un projet web Wordpress (en local dans un premier temps) avec PHP + MySQL (système de gestion de base de données relationnelle) utilisés par Wordpress.*

Debian Buster

"Debian est une organisation communautaire et democratique" dont le but est le développement de systèmes d'exploitation basés exclusivement sur des logiciels libres. On assimile généralement ces systèmes d'exploitation au système Debian GNU/Linux car c'était jusqu'en 2009 la seule branche fonctionnelle (mais d'autres distributions sont en cours de développement). Debian est utilisée comme base de nombreuses autres distributions comme Ubuntu (qui rencontre un grand succès). Debian Buster (version 10) est sortie en juillet 2019 et contient de nombreux paquets dont Nginx, PHP, MariaDB, le noyau linux, apt et bien d'autres (environ 60 000 paquets).

Il existe une documentation fournie :
https://www.debian.org/doc/manuals/debian-reference/ch02.fr.html

Notions Linux / user www:data (a revoir).

https://www.youtube.com/watch?v=JuYfEq4VIWQ
Sur linux il y a un groupe nommé "www:data" qui doit avoir les droits suffisants (exécution) pour que Nginx puisse fonctionner. **Notion à creuser.**

Nginx

Tutoriel de Grafikart : https://www.youtube.com/watch?v=YD_exb9aPZU&t=205s
Information importante sites-enabled

Playlist :
https://www.youtube.com/watch?v=cfJh8vdKuQU&list=PLjwdMgw5TTLUnvhOKLcpCG8ORQsfE7uB4&index=1

Nginx est un serveur très performant, qu' on peut utiliser comme un proxy, un caching service... Je vous recommande de visionner les vidéos de turoriaLinux a ce sujet.
https://www.youtube.com/watch?v=ZyQRqa8I6bU
Cette suite de vidéo est parfaite pour comprendre l'ensemble du projet ft_server (excepté la partie sur docker).

```
→  nginx ls
conf.d           koi-win            nginx.conf       sites-enabled
fastcgi.conf     mime.types         proxy_params     snippets
fastcgi_params   modules-available  scgi_params      uwsgi_params
koi-utf          modules-enabled    sites-available  win-utf
→  nginx pwd
/etc/nginx
→  nginx
```

Le fichier de configuration le plus "high level" est le fichier nginx.conf.
On peut retrouver les différentes pages dans "sites-available". Les "sites-enabled" sont "simplement" un lien symbolique avec les sites-availables ce qui permet de les "unable" en supprimant le lien symbolique.

On retrouve un fichier de configuration "default" dans le dossier "sites-available" qui va permettre de spécifier le server_name, les ports en écoute, les fichiers index, l'éventuelle redirection en 443 (SSL) etc.

```
" Press ? for help

.. (up a dir)
/etc/nginx/sites-available/
  default [RO]
```

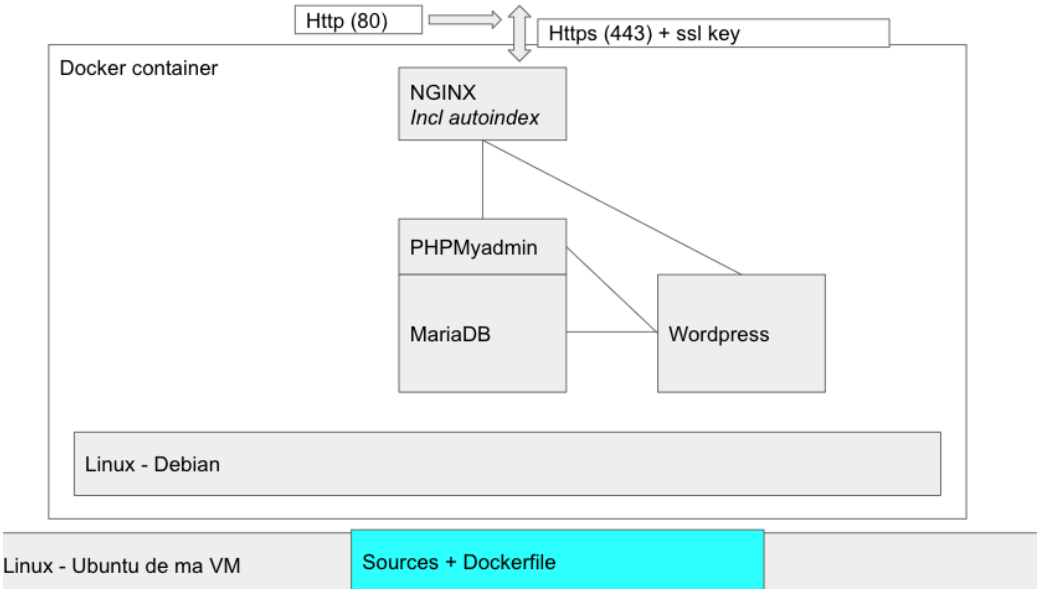
```
# applications, such as Drupal or Wordpress. These applications will
# available underneath a path with that package name, such as /drupal
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed exa
##

# Default server configuration
#
server {
	listen 80 default_server;
	listen [::]:80 default_server;

	# SSL configuration
	#
	# listen 443 ssl default_server;
	# listen [::]:443 ssl default_server;
	#
	# Note: You should disable gzip for SSL traffic.
	# See: https://bugs.debian.org/773332
	#
	# Read up on ssl_ciphers to ensure a secure configuration.
	# See: https://bugs.debian.org/765782
	#
	# Self signed certs generated by the ssl-cert package
	# Don't use them in a production server!
	#
	# include snippets/snakeoil.conf;

	root /var/www/html;

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

	server_name _;

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

	# pass PHP scripts to FastCGI server
	#
```

A noter que des configurations supplémentaires seront à réaliser pour les pages en PHP (puisqu'il s'agit d'un contenu dynamique).

Dans usr/share/nginx/html se trouve le fichier index.html que l'on peut voir depuis la page www://localhost lorsque nginx est running :

```
→  html pwd
/usr/share/nginx/html
→  html cat index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
```

Schéma de l'architecture à mettre en place pour le projet ft_server :



Remplacer MariaDB par **MySQL**

Le serveur http (**Nginx**) va être capable de comprendre les requêtes qui **sont faites par le navigateur** et va rendre/retourner les fichiers en fonction de l'url qui a été tapée (de la requête). Nginx est relativement récent par rapport aux autres outils du marché (2002). Il va pouvoir gérer la montée en charge (beaucoup d'utilisateurs simultanément = +10 000), beaucoup plus simplement que Apache. Cependant il est un petit peu plus compliqué à configurer. Il est un petit peu moins "modulable". Nginx semble être parfait pour les pages/du contenu "statique(s)". **Détails à creuser.**

Docker

Fun fact : en 2015 Docker fonctionnait uniquement sous Linux.
Avantages :

1. Scalable
2. Portable
3. Meilleures performances
4. Déploiement

Comment ca marche ?

1. Docker daemon
2. Docker client
3. Boot2docker (permettait de faire fonctionner Docker lorsqu'il n'était disponible que sur Linux).

Pour mieux comprendre Docker, je vous invite à regarder des vidéos didactiques à ce sujet. Dans la mesure où le docker-compose est interdit pour ft_server, vous n'avez pas "besoin" de vous renseigner dès maintenant sur ce point.

Cette vidéo est particulièrement claire :
https://www.youtube.com/watch?v=i7ABlHngi1Q&t=1144s

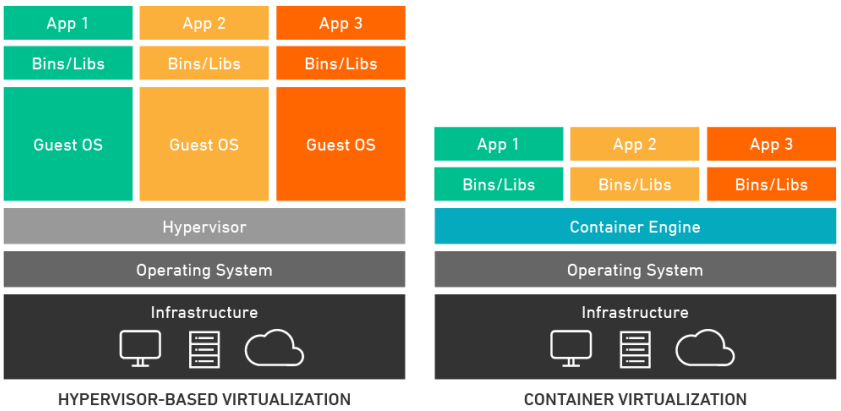
Nous aurons principalement besoin de connaître les notions d'images et de conteneur/conteneurisation + la rédaction d'un Dockerfile.

Containerisation

La containerisation va permettre d' installer php etc (toute la stack) sans être contraint par des problèmes d'OS (marcherait sur la machine d'un développeur mais pas sur la machine d'une autre personne y ayant accès). C' est préférable/recommandé pour le développement local d'un site web.

"Containers are lightweight and faster."

La containerisation se différencie de la virtualisation sur différents points notamment au niveau de l'architecture :



Le container doit tourner avec **Debian Buster. Il s'agit donc du Guest OS.**

Debian Buster est aussi appelé Debian 10 (dernière version stable).

Dockerfile

Le dockerfile servira *uniquement* à build le container. Pas le droit à docker-compose.
Pour plus d'information sur la rédaction d'un dockerfile je vous invite à regarder des vidéos didactiques sur youtube ou à lire la documentation. Il existe quelques commandes / directives telles que **FROM, RUN, WORKDIR, COPY, CMD**…

The `WORKDIR` **command** is used to define the *working directory* of a Docker container at any given time. The command is specified in the Dockerfile.

Any `RUN`, `CMD`, `ADD`, `COPY`, or `ENTRYPOINT` command will be executed in the specified working directory.

Wordpress

Wordpress est le **CMS (Content Management System)** le plus utilisé au monde. Il permet de créer des sites web plus ou moins en "no code". Il possède une interface graphique mais aussi un outil CLI. De nombreuses extensions existent. La majorité des sites dans le monde sont supportés par Wordpress. Wordpress est gratuit, libre et open-source.

Installation de wordpress

https://fr.wordpress.org/support/article/editing-wp-config-php/

Quand vous téléchargez WordPress le fichier `wp-config.php` n'existe pas. Le programme d'installation automatique se charge de le créer pour vous à partir des informations que vous saisissez.

Vous pouvez créer manuellement un fichier `wp-config.php` en éditant le fichier `wp-config-sample.php` qui est situé à la racine du répertoire d'installation puis en le sauvegardant sous le nom de `wp-config.php`.

Pour modifier le fichier `wp-config.php` de votre installation, vous aurez besoin des informations suivantes :

- **Nom de la base donnée**s utilisée par WordPress ;
- **Nom d'utilisateur** utilisé pour accéder à la base de données ;
- **Mot de passe** de l'utilisateur utilisé pour accéder à la base de données ;
- **Nom du serveur** de la base de données. Un numéro de port, un *socket* Unix ou un *pipe* peuvent également être nécessaires.

Si votre hébergeur a installé WordPress pour vous, vous pourrez récupérer ces informations auprès de ce dernier. Si vous gérez vous même votre serveur web ou compte d'hébergement, vous aurez ces informations lorsque vous créez une base de données et un utilisateur.

Wordpress et sa base de données

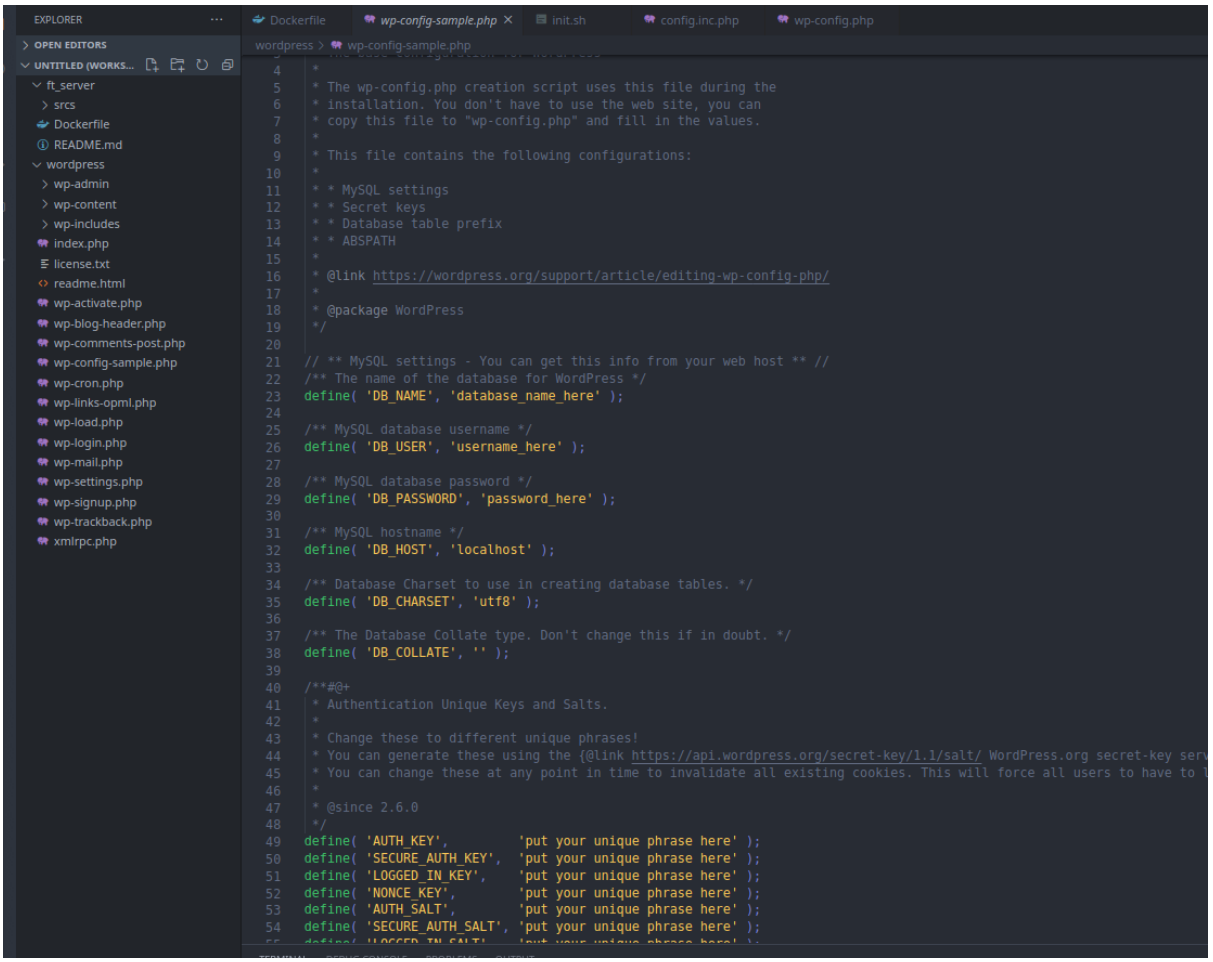
Reference : https://wpmarmite.com/base-donnees-wordpress/
Il semblerait que wordpress ait par défaut une base de données avec différentes tables et champs. Les tables et champs représentent des éléments "classiques" d'un site/blog (exemples : comments, users, links…).

Les informations de connexion à la base de données Wordpress se trouvent dans le fichier de configuration **wp-config.php**.

```
// ** Réglages MySQL - Votre hébergeur doit vous fournir ces informations. ** //
/** Nom de la base de données de WordPress. */
define( 'DB_NAME', 'bdd' );

/** Utilisateur de la base de données MySQL. */
define( 'DB_USER', 'root' );

/** Mot de passe de la base de données MySQL. */
define( 'DB_PASSWORD', 'root' );

/** Adresse de l'hébergement MySQL. */
define( 'DB_HOST', 'localhost' );

/** Jeu de caractères à utiliser par la base de données lors de la création des tables. */
define( 'DB_CHARSET', 'utf8mb4' );
```

Une fois Wordpress téléchargé, vous verrez que le dossier contient de nombreux fichiers et sous-dossiers. Pour pouvoir créer votre fichier wp-config.php, vous devez reprendre le fichier wp-config-sample et l'arranger à votre sauce.

```
total 208
-rw-r--r--  1 user42 user42   405 févr.  6  2020 index.php
-rw-r--r--  1 user42 user42 19915 janv.  1 01:19 license.txt
-rw-r--r--  1 user42 user42  7345 déc.  29 21:14 readme.html
-rw-r--r--  1 user42 user42  7165 janv. 21 02:37 wp-activate.php
drwxr-xr-x  9 user42 user42  4096 mars   9 21:19 wp-admin
-rw-r--r--  1 user42 user42   351 févr.  6  2020 wp-blog-header.php
-rw-r--r--  1 user42 user42  2328 févr. 17 14:08 wp-comments-post.php
-rw-r--r--  1 user42 user42  2913 févr.  6  2020 wp-config-sample.php
drwxr-xr-x  4 user42 user42  4096 mars   9 21:19 wp-content
-rw-r--r--  1 user42 user42  3939 juil. 30  2020 wp-cron.php
drwxr-xr-x 25 user42 user42 12288 mars   9 21:19 wp-includes
-rw-r--r--  1 user42 user42  2496 févr.  6  2020 wp-links-opml.php
-rw-r--r--  1 user42 user42  3313 janv. 10 20:28 wp-load.php
-rw-r--r--  1 user42 user42 44993 févr.  2 19:13 wp-login.php
-rw-r--r--  1 user42 user42  8509 avril 14  2020 wp-mail.php
-rw-r--r--  1 user42 user42 21125 févr.  2 01:10 wp-settings.php
-rw-r--r--  1 user42 user42 31328 janv. 27 22:03 wp-signup.php
-rw-r--r--  1 user42 user42  4747 oct.   8 23:15 wp-trackback.php
-rw-r--r--  1 user42 user42  3236 juin   8  2020 xmlrpc.php
➜  wordpress cat wp-config-sample.php
wp-comments-post.php  wp-config-sample.php  wp-content/
```



Pour “personnaliser” votre fichier de configuration, vous devez remplir les valeurs entre ‘’ dans les define(‘cle’, ‘valeur’).

Vous allez devoir remplir deux blocs de "define" : un premier avec les informations générales de la base de données Wordpress MySQL/MariaDB. Un deuxième qui va permettre l'authentification (a Wordpress) via des secret key cryptées. Pour la deuxième étape, vous allez devoir utiliser l'api mise à disposition qui est mentionnée dans l'url https://api.wordpress etc.

Vous pouvez laisser le reste du code de la page tel quel.

WP_CLI : voir tutoriel de Grafikart.
Installation locale ou "globale" possible (au niveau du dossier exécutable de notre système, la commande sera disponible partout).

MySQL

https://www.youtube.com/watch?v=noy5ZYKfOz8&list=PLjwdMgw5TTLUnvhOKLcpCG8ORQsfE7uB4&index=11

MySQL est un système de gestion de base de données relationnelle aussi **appelé serveur de base de données**. SQL seul (langage qui vous permet de requêter votre base) ne suffit pas, seul. C'est un logiciel libre et open-source. Il va vous permettre de créer, supprimer des bases de données ainsi **que des tables, des champs** et des enregistrements. Pour plus d'informations sur les bases de données je vous invite à faire les recherches nécessaires. MySQL est un serveur tandis que PhpMyAdmin est un client, une interface graphique.

Attention, le mysql-server ne semble pas vouloir s'installer, il faut installer "**default-mysql-server**" pour que ça marche sur debian buster. Pas beaucoup d'informations à ce sujet.

Debian a l' air de vouloir qu'on utilise MariaDB a tout prix. L'installation de MySQL sur Debian Buster est particulièrement complexe (génère plein d'erreurs) tandis que MariaDB fait partie des paquets les plus connus inclus dans Debian Buster. De plus, il convient tout à fait pour PhpMyAdmin et donc le projet wordpress.

PHP

PHP est un langage de programmation qui va vous permettre de dynamiser vos pages web (HTML/CSS). Il interagit avec le navigateur web *et permet également de transmettre des informations au "backend"/a la base de données*. PHP est un langage largement utilisé par les développeurs (web) avec ses 25/30 années d'ancienneté. Il est parfois critiqué notamment en raison de la mauvaise lisibilité du code et d'éventuelles failles de sécurité. C'est un langage qui reste cependant très populaire. La documentation est très fournie et souvent même écrite en français.

PHP vous permet d'interagir avec différents types de base de données et possède de nombreuses librairies/extensions qui contiennent leur spécificités.

Cependant, PHP DPO permet une **couche d' abstraction** qui peut vous permettre d'interagir avec n'importe quelle base de données.

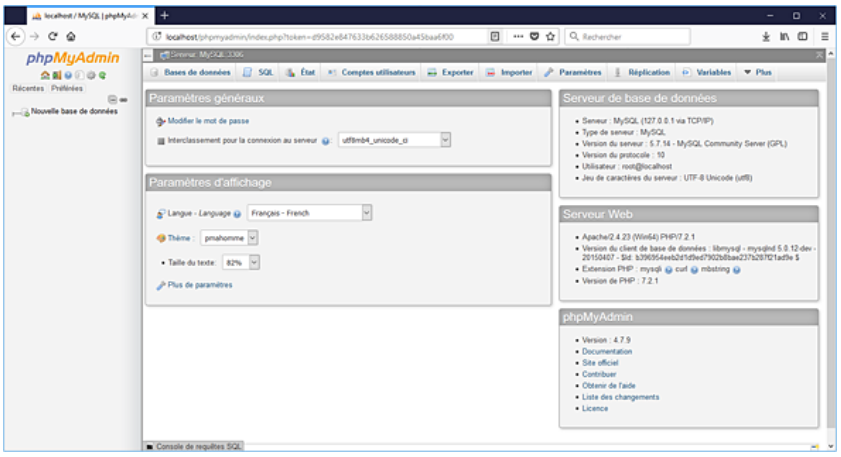
Pour des raisons de sécurité il était recommandé de "configurer" php. /etc/php5/fpm/php.ini.

Fichier phpinfo.php : Nginx va transmettre la requête faite par l'utilisation (principalement un GET sur l'url tapée) et va le transmettre via php fpm pour que cela soit interprété. Il est tres important de faire en sorte que phpinfo.php ne puisse pas être accessible car s'il l'est des pirates peuvent lire les informations relatives à l'environnement et potentiellement savoir comment vous attaquer.

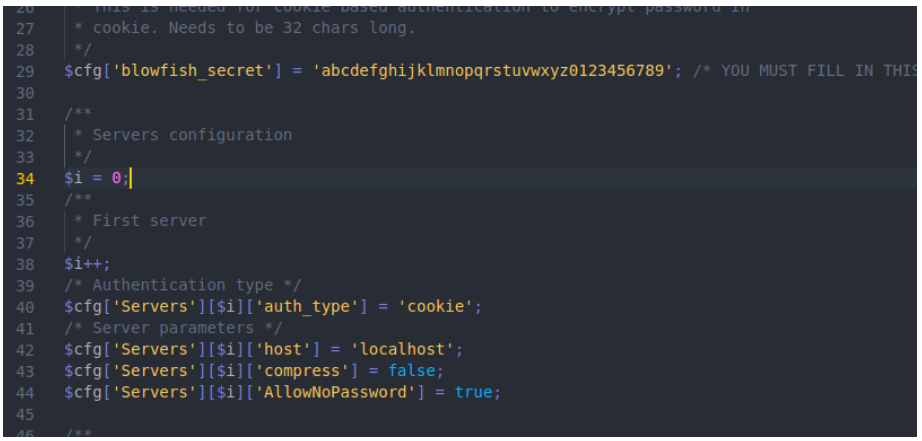
Phpmyadmin (PMA)

*PHPmyadmin est une **application web de gestion** pour les systèmes de gestion de base de données mySQL **et MariaDB**, réalisée principalement en PHP et distribuée sous licence GNU GPL.* C'est l'une des interfaces les plus célèbres pour gérer une base de données MySQL.

La configuration de PhpMyAdmin se fera grâce au fichier config.inc.php (écrit, bien sur, en php).



Il faut retenir que la connexion **par cookie** qui doit se faire pour accéder à l'interface depuis un navigateur nécessite un "blowfish_secret" c'est-à-dire un mot de passe / une clé d'une longueur de 32 chars.



Notez que plusieurs serveurs peuvent être mentionnés même si nous n'en aurons qu'un.
$cfg signifie configuration/config.

SSL

Pour pouvoir utiliser/naviguer via le protocole HTTPS (couche application), vous devez "respecter"/ être identifié en respectant le protocole SSL/TLS (couche transport). SSL signifie Secure Sockets Layer et il est le prédécesseur de TLS. Ce sont des protocoles de sécurisation des échanges par réseau informatique (notamment Internet). Ils fonctionnent suivant un mode "client serveur". Ils vous permettent de :

1. Identifier le serveur
2. Assurer la confidentialité des données échangées (session chiffrée).
3. Assurer l'intégrité de données échangées.
4. De manière optionnelle, l'authentification du client.

Vous aurez besoin de deux fichiers .cert : une clé et un certificat.

Certificate pour permettre l' authentification et pouvoir utiliser le protocole HTTPS.
We will create a self signed certificate (certificate authority and certificate applicant).

First step : Generate a private key for the CA (certificate applicant) and a self signed certificate (it will be used to sign the 'CSR' later.
Second : Generate a private key for the web server and a certificate signing request (CSR)
Third : Use the CA's private key to sign the web server CSR and get back the signed certificate.

La private key est encryptée (ca-cert.pem) mais le certificat ne l'est pas puisqu'il contient une clé publique.

openssl req -x509 -newkey rsa:4096 -days 365 -keyout ca-key.pem -out ca-cert.pem
pour voir toutes les informations sur le certificat :
openssl x509 -in ca-cert.pem -noout -text
En faisant un script, cela va nous permettre d' automatiser le processus de création d'une clé et d' un certificat. On peut utiliser des échos dans notre script pour indiquer ce qu' on est en train de faire.