

NFA083

Réseaux et administration web

Projet final

Etienne LARROUMETS

Table des matières

Introduction	3
Le matériel	3
Topologie.....	3
Déroulement du projet	4
Configuration du serveur de développement	5
Installation de Xampp	5
Création de la base de données	6
Configuration de PHP	6
Installation de CodeIgniter.....	8
Création du VirtualHost	9
Installation et configuration du serveur de production	10
Installation de Ubuntu Server	10
Tests de fonctionnement du réseau	12
Test de la connexion SSH	12
Installation du serveur Apache.....	13
Configuration de PHP	14
Installation de phpMyAdmin	14
Création de la base de données	16
Sécurisation de l'accès à phpMyAdmin.....	18
Création du VirtualHost	19
Sécurisation de Apache	22

Ouverture du serveur vers le réseau public.....	23
Acquisition du nom de domaine	23
Configuration du nom de domaine/serveur.....	23
Ouverture des ports de la Livebox	27
Obtention et installation des certificats SSL/TLS.....	27
Activation du pare-feu UFW.....	29
Création de l'adresse email	29
Configuration d'un serveur externe	30
Configuration de la sauvegarde des fichiers.....	31
Montage de la clé USB de stockage.....	31
Installation et configuration de rsnapshot	32
Automatisation des sauvegardes	33
Conclusion.....	34
Récapitulatif du projet :.....	34

Introduction

Le but de ce projet est de mettre en place un environnement web permettant la mise en ligne d'un site internet.

Pour ce faire, et dans un but pédagogique, j'ai fait le choix de créer moi-même le serveur d'hébergement en partant de zéro. Cela nous permettra de naviguer à travers toutes les étapes de construction du serveur.

Le matériel

Pour héberger le serveur web, j'ai choisi d'utiliser un nano ordinateur Raspberry Pi 4. Cet ordinateur de la taille d'une carte bleue a l'avantage de ne consommer que très peu de ressources, qui plus est quand on y installe un système d'exploitation headless (sans interface graphique). Il est très facilement configurable, on peut y installer un grand nombre de systèmes d'exploitation et notamment des systèmes dédiés à l'hébergement web.



Raspberry Pi 4

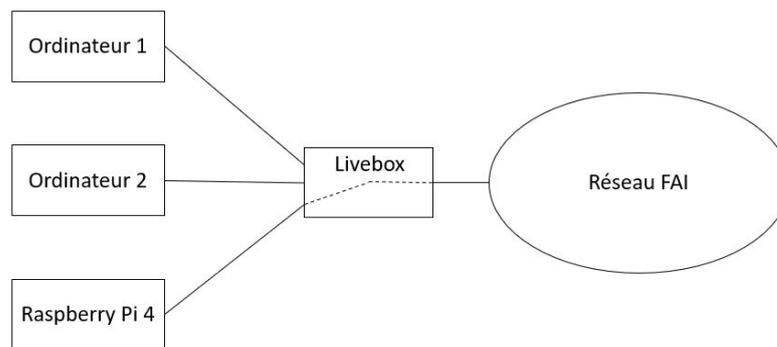
Pour mon projet je vais y installer Ubuntu Server 22.04 LTS ARM qui est un système d'exploitation dans lequel on retrouve toutes les fonctionnalités liées à l'hébergement de services web, bases de données, et bien d'autres encore.

Pour effectuer tous mes tests, j'utiliserai une instance de Ubuntu Server 22.04 tournant dans une machine virtuelle et hébergée sur mon ordinateur de bureau Windows 10. Cela me permettra de ne pas corrompre ou endommager mon serveur de production en cas de mauvaise manipulation ou de test infructueux.

Enfin, pour mon environnement de développement, j'utiliserai Windows 10 car c'est la version qui est installée sur mon ordinateur de bureau.

Topologie

Ce Raspberry Pi sera relié en réseau filaire Ethernet à la Livebox (Oraneg) de mon domicile. C'est elle qui fait office de routeur entre mon réseau local et le réseau extérieur de l'opérateur. Une ouverture des ports sera nécessaire afin que le serveur soit accessible depuis l'extérieur de mon réseau local.



Topologie du réseau utilisé

Je fais le choix de relier le Raspberry à mon routeur via une liaison Ethernet ce qui facilitera la gestion de l'interface réseau et sécurisera le transfert des données à travers le réseau local.

Déroulement du projet

Pour mener à bien ce projet, je commencerai par installer Ubuntu Server sur mon nano-ordinateur. Je ferai une pré-installation de OpenSSH dans le but d'administrer à distance mon serveur.

Ensuite je mettrai en place un serveur web Apache qui sera chargé de répondre aux requêtes HTTP. Sur ce serveur sera installé PHP, ce qui permettra de générer des pages web dynamiques. J'installerai également une base de données MySQL afin de stocker les données du site

Afin de renforcer la sécurité et faciliter le développement du site hébergé, j'utiliserai un framework PHP. Étant donné que le serveur hébergera un site simple de type blog, j'ai fait le choix du framework CodeIgniter4 qui, une fois déployé en production, ne pèse que 1,5 Mo. L'utilisation d'un framework facilite aussi l'échange entre développeurs ou bien encore la transmission de l'administration d'un site.

Pour que mon site soit facilement accessible depuis l'extérieur, j'achèterai un nom de domaine auprès d'un registrar et j'utiliserai un système de DNS Dynamique qui adaptera continuellement mon nom de domaine à l'adresse IP publique de mon réseau en cas de changement de cette dernière (dorénavant, avec un abonnement fibre, Orange ne modifie l'adresse IP d'un particulier que dans certaines situations exceptionnelles mais il est toujours intéressant de se prémunir en amont).

J'installerai ensuite les certificats SSL/TLS sur le serveur afin d'assurer le chiffrement des données entre le serveur et le client. Pour cela je ferai la demande de certificat auprès de l'organisme Let's Encrypt via le programme Certbot de Ubuntu Server.

Puis je créerai une adresse mail liée à mon nom de domaine via un serveur mail externe. Une redirection DNS permettra de lier ce serveur à mon domaine.

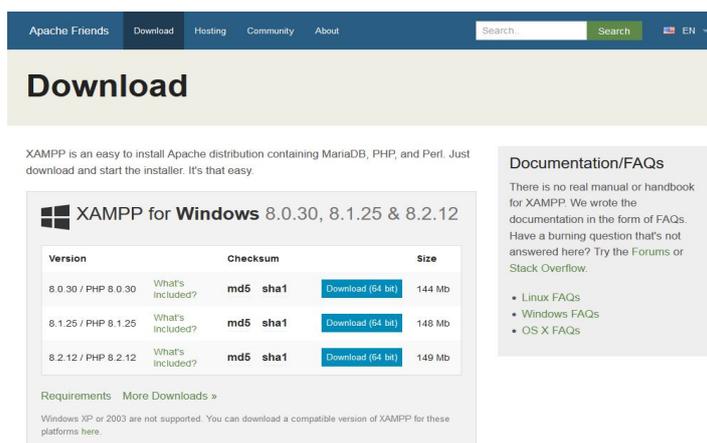
Pour terminer, je mettrai en place un système de sauvegardes automatiques des fichiers importants et de la base de données de mon serveur.

Configuration du serveur de développement

Afin de développer un site internet de manière efficace, il est conseillé de séparer l'environnement de développement de celui de production. Je vais donc créer un serveur web sur ma machine de bureau Windows 10. Cet environnement devra être le plus ressemblant possible à l'environnement de production, ainsi, lors du transfert des fichiers vers celui-ci, on évitera au maximum les mauvaises surprises.

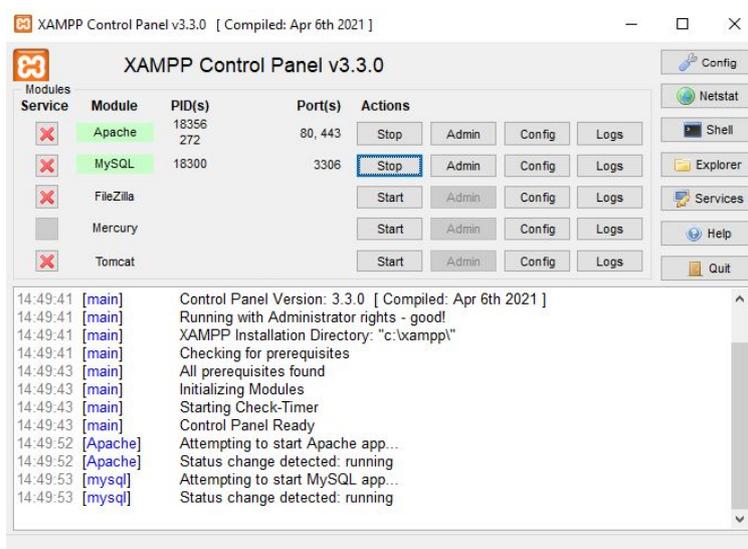
Installation de Xampp

Pour faire tourner un serveur Apache fonctionnel en local je vais installer Xampp sur ma machine. Xampp est une distribution permettant d'installer un serveur Apache, une base de données (MariaDB), PHP et Perl sur une machine Windows, Linux ou bien Mac. Cette version de Xampp contient Apache en version 2.4.58 et PHP en version 8.2.12 :



Page de téléchargement de Xampp

Et voici le panneau de contrôle de Xampp après avoir lancé le serveur Apache et MySQL :



Panneau de contrôle de Xampp

J'en profite pour modifier le fichier hosts de mon ordinateur de bureau (« C:\Windows\System32\drivers\etc\hosts »), afin de faire pointer différents noms de domaines vers des adresses IP locales :

```
15 192.168.1.41 larroumets.dev.ubuntu
16 192.168.1.47 larroumets.dev.local
17 127.0.0.1 larroumets.dev.win
18
```

Configuration du fichier hosts Windows

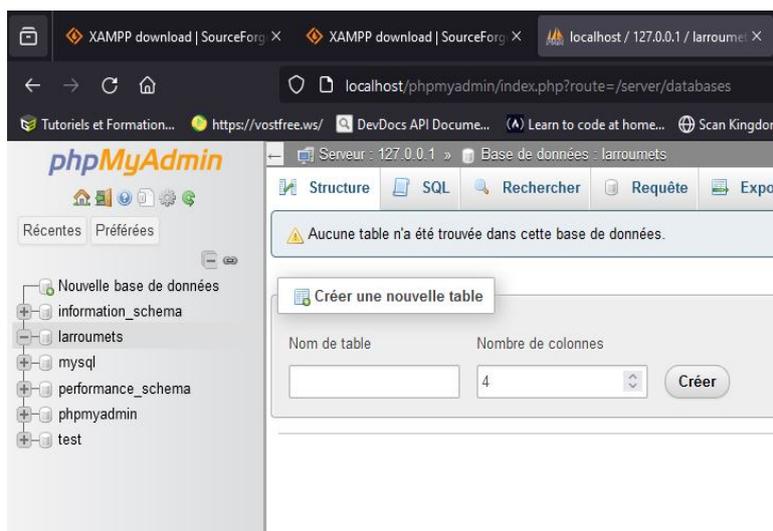
Le nom de domaine **larroumets.dev.local** pointe vers le serveur de production (192.168.1.47) fonctionnant sur mon Raspberry Pi.

Le nom de domaine **larroumets.dev.ubuntu** pointe vers une instance de Ubuntu Server (192.168.1.41) fonctionnant sur une machine virtuelle hébergée sur ma machine de bureau Windows qui me sert pour tous mes tests.

Enfin, **larroumets.dev.win** pointera sur l'adresse de loopback pour accéder au serveur développement sur ma machine Windows 10 (Xampp).

Création de la base de données

Xampp nous facilite grandement la tâche pour créer une base de données, un simple clic sur le bouton « Admin » du panneau de contrôle nous renvoie vers phpMyAdmin dans lequel je vais pouvoir créer ma base que je nommerai « *larroumets* » :

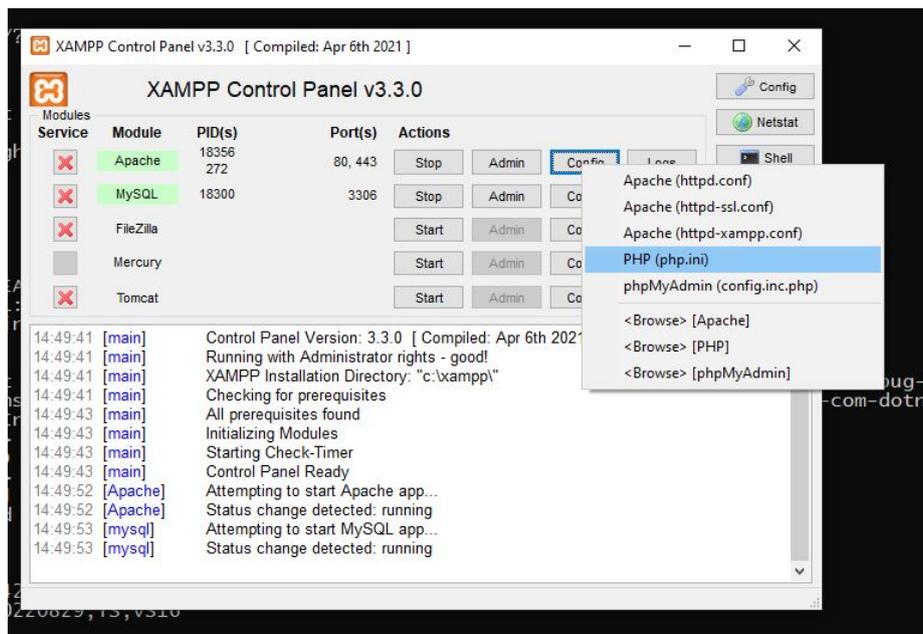


Création de la base dans phpMyAdmin

Configuration de PHP

Pour fonctionner correctement, le framework CodeIgniter a besoin d'une version PHP au moins équivalente à la version 7.4 avec les extensions intl, mbstring et json activées. L'extension json fait partie intégrante de PHP depuis sa version 8.0, la version étant installée sur mon système étant la 8.2.12, je n'aurai donc pas à l'activer manuellement. Accessoirement, j'aurai besoin de l'extension mysqlnd pour utiliser MySQL.

La commande « **php -i** » dans la console de Xampp me permet de connaître la liste des extensions activées. Elles le sont toutes à l'exception de *intl*. Je vais donc devoir l'activer manuellement en décommentant la ligne dans le fichier *php.ini*. Encore une fois, Xampp nous aide en nous proposant d'accéder directement au fichier de configuration depuis le panneau de contrôle :



Accès au fichier de configuration de PHP

Activation de l'extension *intl* dans *php.ini* :

```
extension=mysqli
;extension=gd
extension=gettext
;extension=gmp
extension=intl
;extension=imap
extension=mbstring
extension=exif ; Must be after mbstring as it depends on it
extension=mysqli
;extension=oci8_12c ; Use with Oracle Database 12c Instant Client
;extension=oci8_19 ; Use with Oracle Database 19 Instant Client
```

Activation de l'extension intl

Un nouvel appel à la commande **php -i** nous confirme l'activation de *intl* :

```
intl
Internationalization support => enabled
ICU version => 71.1
ICU Data version => 71.1
ICU TZData version => 2022a
ICU Unicode version => 14.0
```

intl activé

Installation de CodeIgniter

L'installation de CodeIgniter se fait à l'aide de Composer (<https://getcomposer.org/>) un gestionnaire de dépendances PHP qui facilite grandement la tâche des développeurs. La plupart des framework PHP peuvent s'installer via Composer (Laravel, Symfony, etc...).

Après téléchargement et installation, tout se passe en ligne de commandes :

```
Invite de commandes
Microsoft Windows [version 10.0.19045.3930]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Maison>composer

Composer version 2.6.6 2023-12-08 18:32:26

Usage:
  command [options] [arguments]
```

Page d'accueil de Composer

Je ne m'attarderai pas trop sur l'installation et la configuration de CodeIgniter car cela déborderait du cadre de NFA083 mais voici les étapes effectuées pour installer le framework :

Je commence par créer un dossier qui contiendra les fichiers CodeIgniter puis je démarre son installation avec la commande :

```
composer create-project codeigniter4/appstarter larroumets_dev
```

Je configure ensuite l'URL de base dans **app/Config/App.php** :

```
12 * -----
13 *
14 * URL to your CodeIgniter root. Typically, this will be your base URL,
15 * WITH a trailing slash:
16 *
17 *   http://example.com/
18 */
19 public string $baseUrl = 'http://larroumets.dev.win';
20
21 /**
22 * Allowed Hostnames in the Site URL other than the hostname in the baseUrl.
23 * If you want to accept multiple Hostnames, set this.
```

Configuration de l'URL de base dans CodeIgniter

Je configure les paramètres de connexion à la base de données dans **app/Config/Database.php** :

```
27 public array $default = [
28     'DSN'           => '',
29     'hostname'     => 'localhost',
30     'username'     => 'root',
31     'password'     => '',
32     'database'     => 'larroumets',
33     'DBDriver'     => 'MySQLi',
34     'DBPrefix'     => '',
35     'pConnect'     => false,
```

Configuration de l'accès à la base de données dans CodeIgniter

J'effectue une copie du fichier `env` que je renomme en `.env`. Ce fichier contiendra la configuration de l'environnement. Dans ce fichier, je modifie la ligne `CI_ENVIRONMENT` pour indiquer au framework que je passe en mode développement :

```
13 #-----
14 # ENVIRONMENT
15 #-----
16
17 CI_ENVIRONMENT = development
18
19 #-----
20 # APP
21 #-----
22
```

Passage en mode développement de CodeIgniter

Je termine en testant l'installation du framework. Pour ce faire, j'utilise le serveur interne de ce dernier ce qui me permet de faire un test rapide sans avoir encore configuré le VirtualHost :

```
PS E:\DEUST IOSI Etienne\Cours\NFA083 - Réseaux Administration web\Projet\larroumets_dev> php spark serve
CodeIgniter v4.4.4 Command Line Tool - Server Time: 2024-01-20 15:09:40 UTC+00:00
CodeIgniter development server started on http://localhost:8080
Press Control-C to stop.
[Sat Jan 20 16:09:40 2024] PHP 8.1.2 Development Server (http://localhost:8080) started
[Sat Jan 20 16:09:48 2024] [::1]:56525 Accepted
[Sat Jan 20 16:09:48 2024] [::1]:56525 Closing
```

Démarrage du serveur CodeIgniter

Création du VirtualHost

Le système de VirtualHosts (serveurs virtuels) est très pratique car il permet de faire fonctionner un ou plusieurs serveurs sur une même machine physique. Chaque serveur virtuel peut avoir sa propre configuration ce qui rend l'administration des serveurs plus simple et plus sûre.

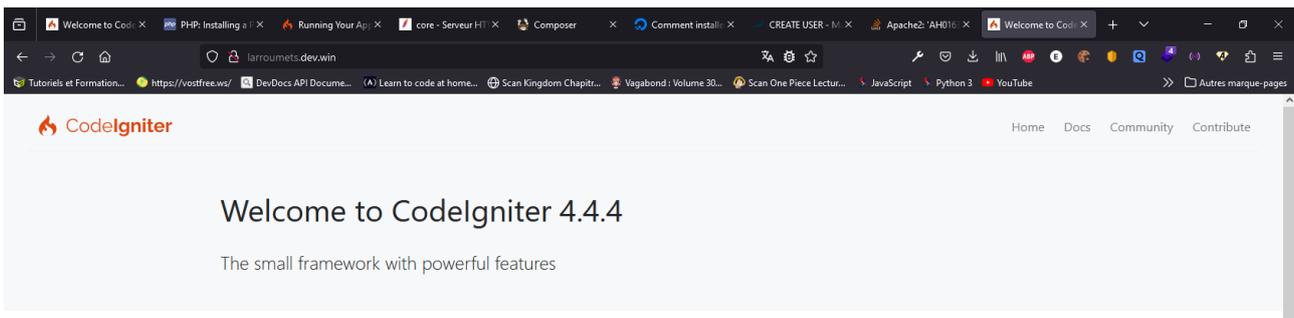
Pour créer un VirtualHost je dois le déclarer et le configurer dans le fichier `C:\xampp\apache\conf\extra\httpd-vhosts.conf` :

```
44 <VirtualHost *:80>
45     DocumentRoot "E:\DEUST IOSI Etienne\Cours\NFA083 - Réseaux Administration web\Projet\larroumets_dev\public"
46     ServerName larroumets.dev.win
47     ServerAlias www.larroumets.dev.win
48     ErrorLog "logs/larroumets-dev-error.log"
49     CustomLog "logs/larroumets-dev-access.log" common
50     <Directory "E:\DEUST IOSI Etienne\Cours\NFA083 - Réseaux Administration web\Projet\larroumets_dev\public">
51         AllowOverride All
52         Require all granted
53     </Directory>
54 </VirtualHost>
```

Configuration du VirtualHost de développement

- DocumentRoot : chemin vers les fichiers de la partie publique du site
- ServerName : permet d'identifier le serveur virtuel de manière unique
- ServerAlias : permet de définir des noms alternatifs pour atteindre le serveur virtuel
- ErrorLog : permet d'enregistrer les erreurs liées au serveur
- CustomLog : permet d'enregistrer les accès au serveur
- Directory : permet d'indiquer des directives liées à un répertoire en particulier
- Allowoverride All : permet d'outrepasser la configuration si besoin, avec un fichier `.htaccess` par exemple
- Require all granted : autorise l'accès à ce répertoire à tout le monde.

Une fois le fichier enregistré, je redémarre Apache et je tente de me connecter à mon site via mon navigateur :



Le site fonctionne en local sur ma machine

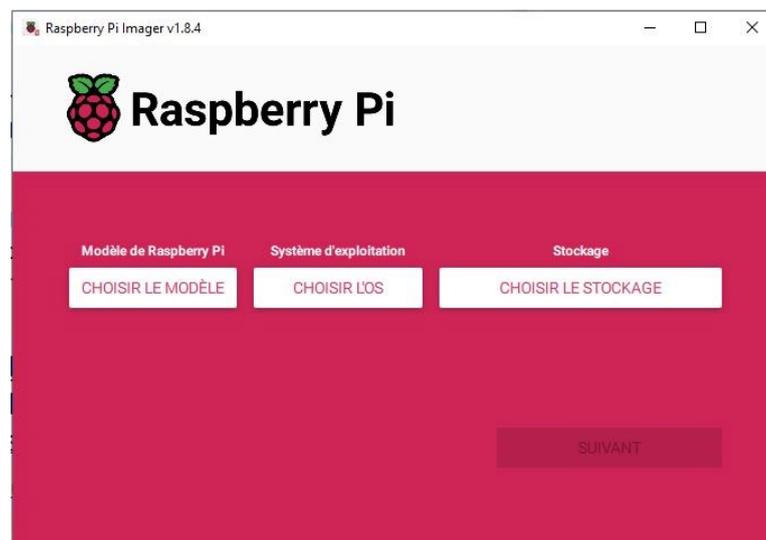
Le serveur virtuel est correctement configuré, la page de mon site s'affiche correctement.

Mon environnement de développement est prêt, je peux maintenant m'attaquer à l'installation et la configuration du serveur de production.

Installation et configuration du serveur de production

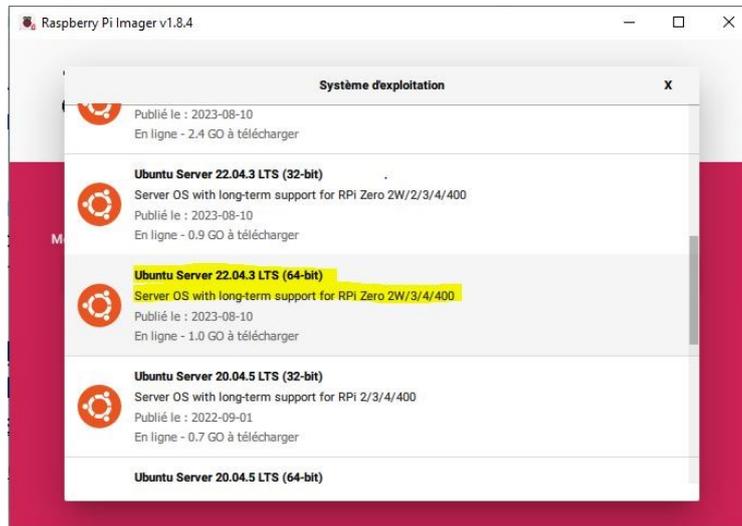
Installation de Ubuntu Server

Pour installer Ubuntu Server sur un Raspberry Pi 4, la solution la plus simple consiste à utiliser l'utilitaire Raspberry Pi Imager fourni par Raspberry.



Démarrage de Raspberry Pi Imager

Cet utilitaire tout-en-un permet de sélectionner le système d'exploitation de son choix, le modèle de Raspberry Pi sur lequel l'installer et la carte SD pour le stockage.



Choix de l'OS dans Raspberry Pi Imager

Je choisis donc Ubuntu Server 22.04.3 LTS en version 64bits car cette version bénéficie d'un support à long terme (LTS).

Je configure ensuite mon installation (nom d'hôte, nom d'utilisateur, activation de SSH, mot de passe SSH, etc.) et je lance l'installation sur la carte SD.

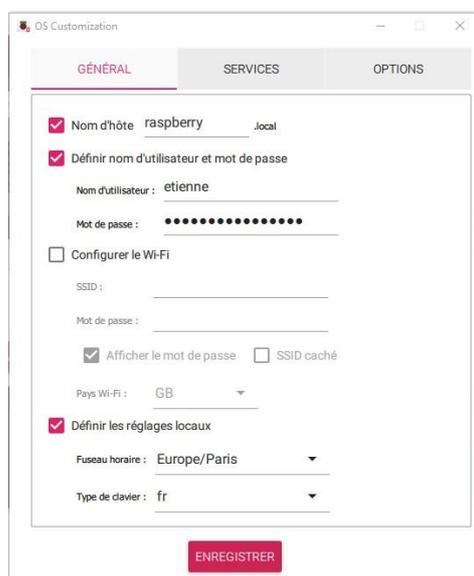


Figure 19 : pré-configuration de Ubuntu

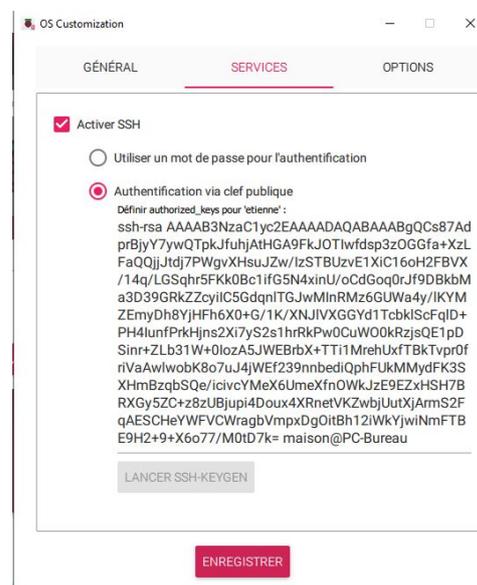


Figure 20 : Activation de SSH

Une fois l'installation terminée, j'insère la carte SD dans le Raspberry Pi, je branche l'alimentation, je le relie à mon réseau local via le connecteur RJ45 et je peux me lancer dans les premiers tests.



Connexion au réseau local

Tests de fonctionnement du réseau

Je teste la liaison locale en effectuant un ping vers le Raspberry via la console Windows depuis mon ordinateur de bureau :

```
Sélection Invite de commandes
Microsoft Windows [version 10.0.19045.3803]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Maison>ping 192.168.1.47

Envoi d'une requête 'Ping' 192.168.1.47 avec 32 octets de données :
Réponse de 192.168.1.47 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.1.47 :
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 0ms, Maximum = 0ms, Moyenne = 0ms

C:\Users\Maison>
```

Test de ping vers le serveur en local

Tous les paquets sont reçus, la liaison est correcte, le système d'exploitation coté Raspberry fonctionne et l'interface réseau est correctement configurée.

Test de la connexion SSH

Afin d'administrer mon serveur à distance, je dois ouvrir un shell sécurisé SSH (Secure Shell). Plusieurs possibilités sont offertes pour ouvrir un shell à distance. Pour ma part j'utiliserai la console Windows qui fonctionne très bien pour cela mais j'aurais pu utiliser PuTTY qui est un utilitaire dédié à cette fonctionnalité

J'ouvre donc un invite de commande Windows et je rentre la commande suivante :

```
ssh etienne@192.168.1.47
```

La machine m'informe que l'empreinte numérique du serveur n'est pas connue et me demande si je suis sûr de vouloir m'y connecter. Je valide par « yes », je rentre mon mot de passe préalablement configuré et le shell de la machine distante s'ouvre.

```

C:\Users\Maison>ssh etienne@192.168.1.47
The authenticity of host '192.168.1.47 (192.168.1.47)' can't be established.
ECDSA key fingerprint is SHA256:He1Fkd1eFxpNlKTUnH0f3PQyEOfOUZGQD+RsHyHJV4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.47' (ECDSA) to the list of known hosts.
etienne@192.168.1.47's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-1034-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Dec 25 19:28:26 CET 2023

System load:          0.080078125
Usage of /:           8.5% of 28.93GB
Memory usage:        5%
Swap usage:          0%
Temperature:         49.7 C
Processes:           135
Users logged in:     0
IPv4 address for eth0: 192.168.1.47
IPv6 address for eth0: 2a01:cb19:8f27:5f00:da3a:ddff:fe83:2240

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

etienne@raspberry:~$

```

Accès distant au serveur via SSH

Maintenant la première chose à effectuer est la mise à jour du système. En effet il est très important de maintenir le système à jour afin de pallier aux failles de sécurité qui peuvent être présentes. Les deux commandes suivantes vont respectivement mettre à jour les informations des paquets logiciels et appliquer les mises à jour :

```

sudo apt update
sudo apt upgrade

```

Après la mise à jour, il faudra redémarrer la machine avec : `sudo reboot` et se reconnecter à nouveau au shell sécurisé. Nous avons donc maintenant un système opérationnel et à jour.

Installation du serveur Apache

Pour installer Apache sur notre serveur allons utiliser la pile logicielle LAMP (Linux Apache MySQL PHP). Je suis les recommandations indiquées par la documentation d'Ubuntu pour installer les paquets nécessaires :

```

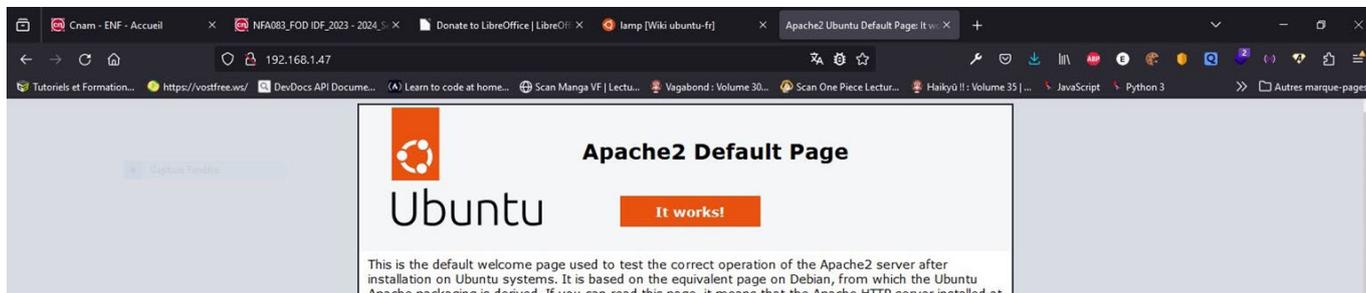
sudo apt install apache2 php libapache2-mod-php mysql-server php-
mysql

```

Afin de pouvoir faire fonctionner mon framework PHP correctement, j'installe également les paquets les plus courants :

```
sudo apt install php-curl php-gd php-intl php-json php-mbstring php-xml php-zip
```

Une fois l'installation terminée, il ne reste plus qu'à tester le serveur en rentrant son adresse IP dans la barre d'adresse du navigateur de mon ordinateur de bureau :



Le serveur Apache est opérationnel

La page d'accueil d'Apache avec le fameux slogan « It works ! » nous indique que le serveur est opérationnel sur le réseau local.

Configuration de PHP

Pour fonctionner correctement, CodeIgniter a besoin d'une version PHP au moins équivalente à la version 7.4 avec les extensions intl, mbstring et json activées. L'extension json fait partie intégrante de PHP depuis sa version 8.0, la version étant installée sur mon système étant la 8.1.2, je n'aurai donc pas à l'activer manuellement. Accessoirement, j'aurai besoin de l'extension mysqlnd pour utiliser MySQL.

La commande `php -i` me permet de connaître la configuration de PHP sur mon système. Je vois que les extensions intl, mbstring, json et mysqlnd sont activées par défaut.

En annexe de ce dossier vous trouverez le fichier de sortie de la config PHP obtenue avec `php -i > config_php` et récupérée à l'aide de WinSCP via une connexion SFTP.

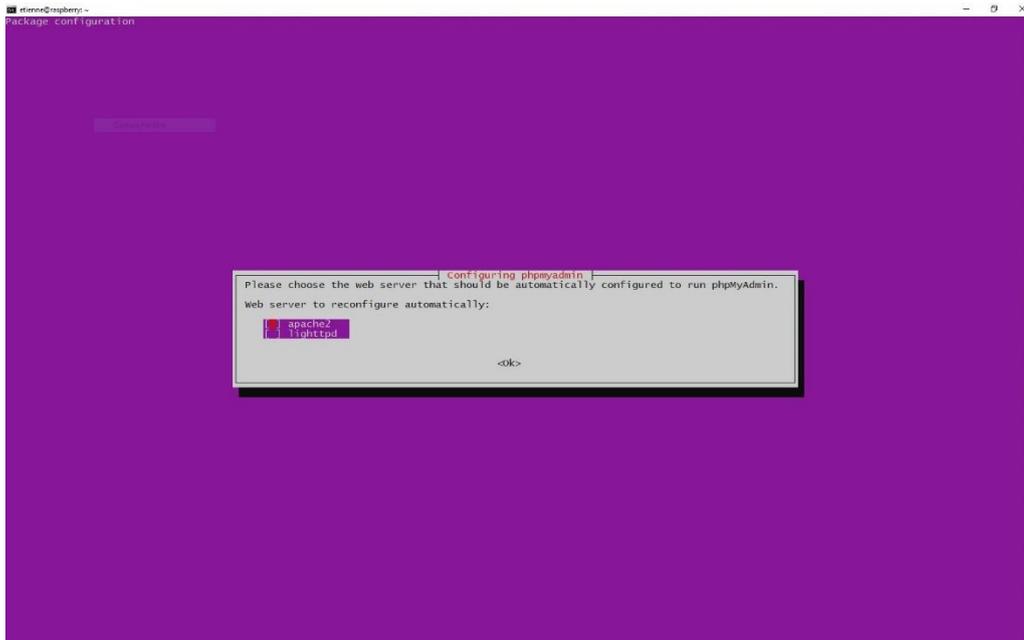
Installation de phpMyAdmin

Afin de faire fonctionner le framework, j'aurai besoin d'utiliser une base de données. Nous avons déjà installé MySQL avec la pile logicielle LAMP.

Pour faciliter la création et l'administration de ma base de données, je vais installer phpMyAdmin sur mon serveur. Pour cela, j'utilise la commande :

```
sudo apt install phpmyadmin
```

L'utilitaire d'installation est lancé :



Écran d'installation de phpMyAdmin

L'utilitaire me demande de choisir le serveur à reconfigurer (Apache2) puis il me demande si je veux qu'il s'occupe de la création de la base. Je confirme par « Oui ». Ensuite il me demande d'enregistrer un mot de passe de connexion. Enfin il me demande de confirmer le redémarrage des services nécessaires au bon fonctionnement du serveur.

Une fois l'exécution terminée, je me connecte à mon serveur dans mon navigateur à l'adresse 192.168.1.47/phpmyadmin afin de tester le bon fonctionnement du service :



Écran d'authentification de phpMyAdmin

La page d'accueil de phpMyAdmin s'affiche, le service est correctement installé !

Pour accéder à la liste des utilisateurs de MySQL, je démarre MySQL à l'aide de la commande :

```
sudo mysql
```

puis je rentre la requête SQL suivante :

```
mysql> SELECT user FROM mysql.user ;
```

```
mysql> SELECT user
-> ;
ERROR 1054 (42S22): Unknown column 'user' in 'field list'
mysql> SELECT user FROM mysql.user;
+-----+
| user |
+-----+
| debian-sys-maint |
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| phpmyadmin       |
| root             |
+-----+
6 rows in set (0.00 sec)

mysql>
```

Liste des utilisateurs de MySql

Lors de l'installation phpMyAdmin a créé un utilisateur *phpmyadmin*, c'est l'utilisateur par défaut du service, celui pour lequel il m'a été demandé de créer le mot de passe.

Pour simplifier les choses, je vais créer un utilisateur dédié pour me connecter au service, toujours dans l'utilitaire mysql :

```
mysql> CREATE USER 'etienne'@'localhost' IDENTIFIED WITH caching_sha2_password BY '*****';
```

```
mysql> CREATE USER 'etienne'@'localhost' IDENTIFIED WITH caching_sha2_password BY '*****';
Query OK, 0 rows affected (0,07 sec)

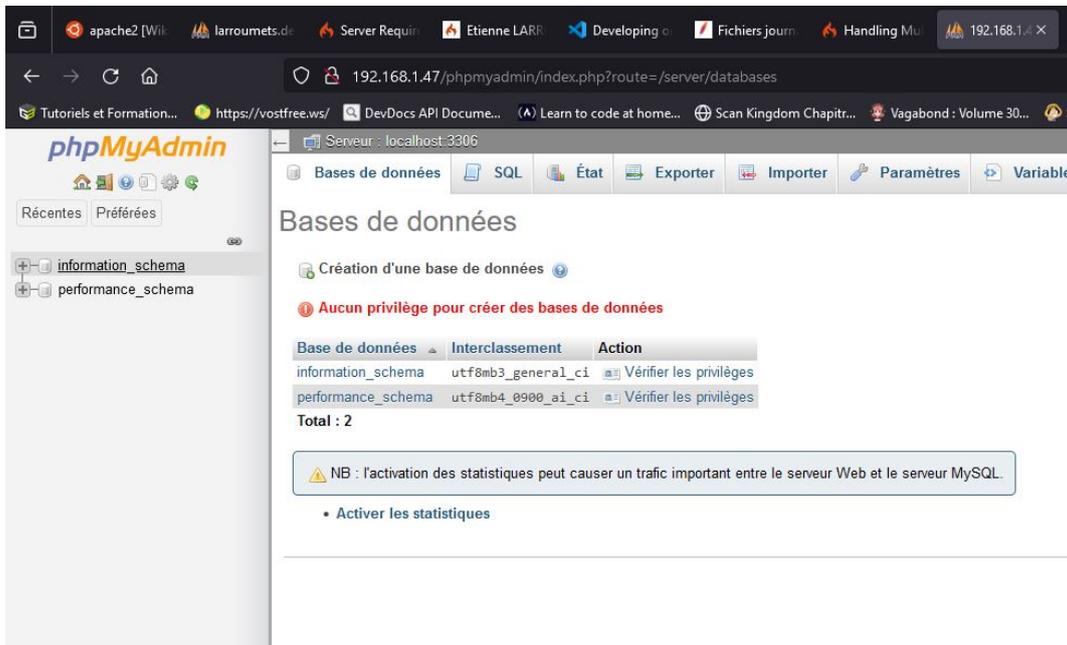
mysql> SELECT user FROM mysql.user;
+-----+
| user |
+-----+
| debian-sys-maint |
| etienne          |
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| phpmyadmin       |
| root             |
+-----+
7 rows in set (0,00 sec)
```

Ajout de l'utilisateur etienne

Création de la base de données

Maintenant que MySQL et phpMyAdmin sont opérationnels, je peux créer la base de données dont se servira mon site internet.

Pour cela, je me connecte à phpMyAdmin mais je me rends compte que je n'ai pas les privilèges pour créer une base de données :

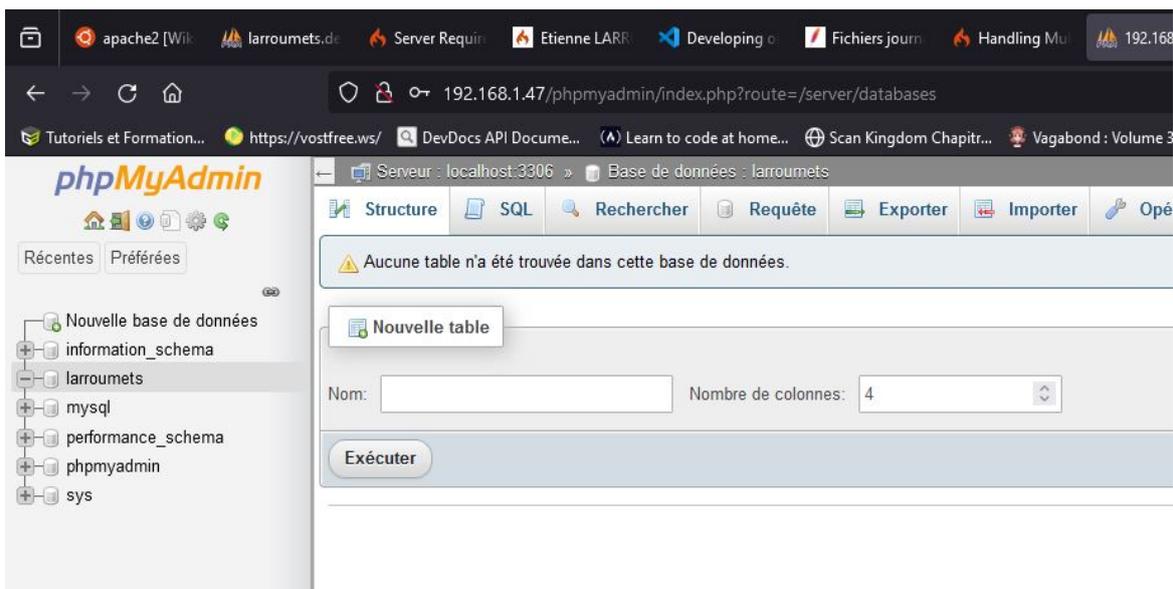


Création de la base de données refusée

Je dois donc retourner dans l'utilitaire mysql pour donner les droits nécessaires à l'utilisateur « etienne » :

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'etienne'@'localhost' WITH GRANT OPTION;
```

Je quitte phpMyAdmin, recharge la page, et me reconnecte. Je tente de créer ma base de données en cliquant sur « Créer une nouvelle base » et je la nomme « larroumets » :



Création de la base "larroumets"

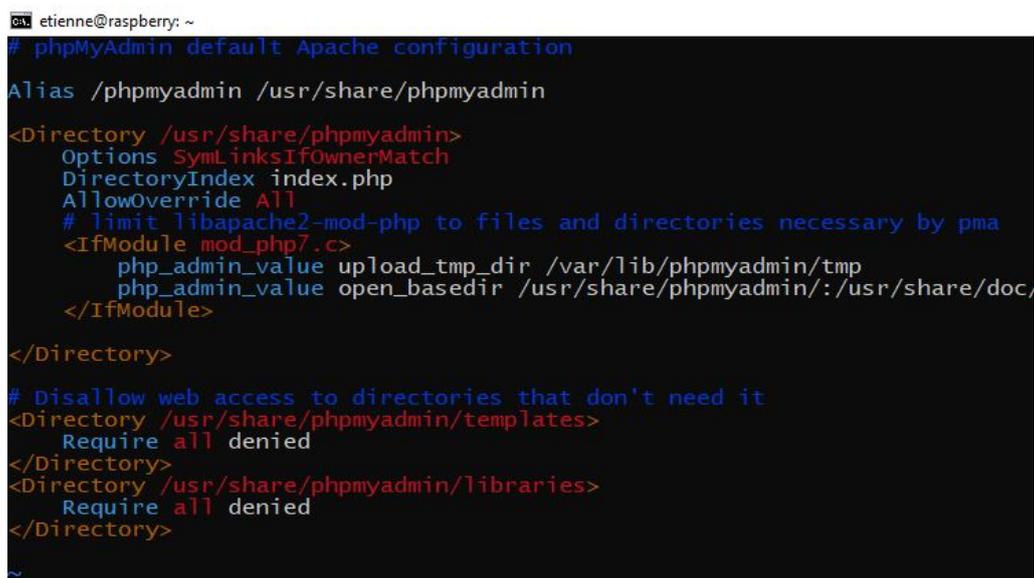
Je ne crée pas de table pour le moment, celles-ci seront créées via CodeIgniter (Database Migrations) quand le besoin s'en fera sentir au fur et à mesure du développement du site.

Sécurisation de l'accès à phpMyAdmin

Par défaut, l'accès à l'interface de connexion de phpMyAdmin est libre pour tout utilisateur, ce qui pose de sévères problèmes de sécurité. Pour remédier à cela, je vais verrouiller l'accès à cette page via un fichier .htaccess.

Je commence par autoriser l'utilisation d'un fichier .htaccess dans la configuration de phpMyAdmin. Le fichier de configuration de phpMyAdmin se trouve dans `/etc/apache2/conf-available/`. Je rajoute la ligne `AllowOverride All` à la directive `Directory` :

```
sudo vim /etc/apache2/conf-available/phpmyadmin.conf
```



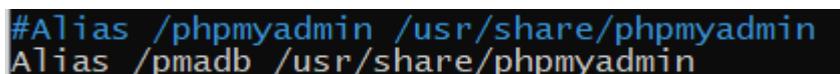
```
etienne@raspberrypi: ~
# phpMyAdmin default Apache configuration
Alias /phpmyadmin /usr/share/phpmyadmin

<Directory /usr/share/phpmyadmin>
  Options SymLinksIfOwnerMatch
  DirectoryIndex index.php
  AllowOverride All
  # limit libapache2-mod-php to files and directories necessary by pma
  <IfModule mod_php7.c>
    php_admin_value upload_tmp_dir /var/lib/phpmyadmin/tmp
    php_admin_value open_basedir /usr/share/phpmyadmin:/usr/share/doc
  </IfModule>
</Directory>

# Disallow web access to directories that don't need it
<Directory /usr/share/phpmyadmin/templates>
  Require all denied
</Directory>
<Directory /usr/share/phpmyadmin/libraries>
  Require all denied
</Directory>
```

Configuration de phpMyAdmin

Edit : En faisant la relecture de mon dossier, je m'aperçois qu'il n'est pas très pertinent en termes de sécurité de laisser le chemin d'accès à phpmyadmin par défaut (<https://larroumets.dev/phpmyadmin>). J'ai donc modifié celui-ci par (<https://larroumets.dev/pmadb>) dans phpmyadmin.conf :



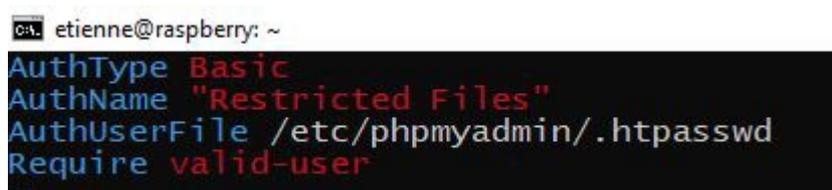
```
#Alias /phpmyadmin /usr/share/phpmyadmin
Alias /pmadb /usr/share/phpmyadmin
```

Figure 32 : Modification du chemin d'accès à phpMyAdmin

Je crée maintenant mon fichier .htaccess dans le répertoire de phpMyAdmin :

```
sudo vim /usr/share/phpmyadmin/.htaccess
```

Et je rajoute les lignes suivantes :



```
etienne@raspberrypi: ~
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /etc/phpmyadmin/.htpasswd
Require valid-user
```

Configuration du .htaccess

Auth-Type : indique le type d'authentification utilisé

AuthName : indique le message affiché lors de la demande d'authentification

AuthUserFile : indique le chemin vers le fichier contenant les mots de passe

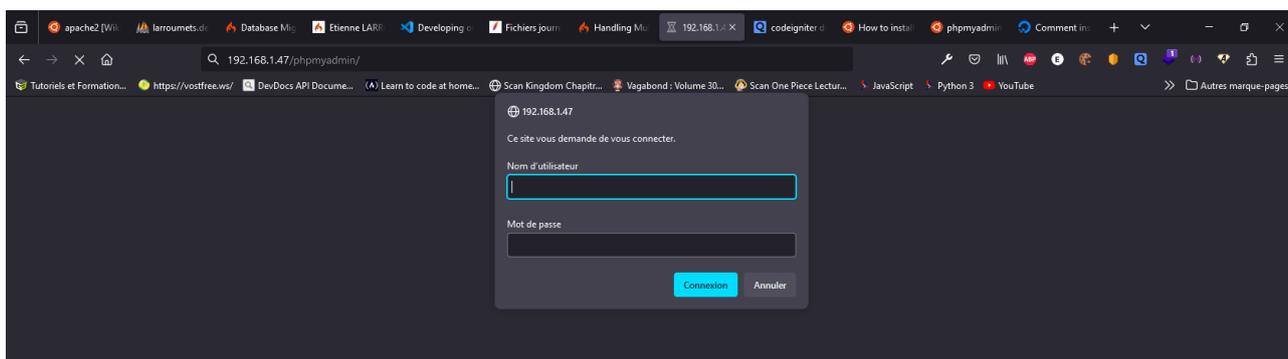
Require valid-user : indique que seuls les utilisateurs authentifiés peuvent accéder à la ressource

Je peux maintenant créer le fichier `htpasswd` grâce à une commande que nous fournit nativement Ubuntu, la commande « `htpasswd` » :

```
etienne@raspberrypi:~$ sudo htpasswd -c /etc/phpmyadmin/.htpasswd etienne
New password:
Re-type new password:
Adding password for user etienne
etienne@raspberrypi:~$ sudo systemctl restart apache2
etienne@raspberrypi:~$
```

Création du `.htpasswd`

Après redémarrage d'Apache, je teste à nouveau la connexion à phpMyAdmin et cette fois-ci le serveur me demande de m'authentifier pour accéder à la ressource :



Demande d'authentification `.htaccess`

Pour des raisons évidentes de sécurité, il est déconseillé d'utiliser le même mot de passe pour le `.htaccess` et l'authentification à phpMyAdmin. Dans le cas contraire tout l'intérêt de ce dispositif tombe à l'eau !

Me voilà donc avec une base de données opérationnelle avec un accès sécurisé, je peux passer à l'installation du VirtualHost.

Création du VirtualHost

Pour créer un VirtualHost, je vais reprendre la configuration de mon environnement de développement dans laquelle je vais modifier le chemin vers le dossier contenant les fichiers du framework

```
cd /etc/apache2/sites-available/
```

J'édite le fichier créé avec Vim afin de configurer le VirtualHost :

```
sudo vim larroumets-dev.conf
```

Je reprends la même configuration que celle de mon environnement de développement en adaptant le nom du serveur et le chemin vers les fichiers :

```
etienne@raspberrypi: /var/www
<VirtualHost *:80>
    ServerName larroumets.dev.local
    ServerAlias www.larroumets.dev.local
    DocumentRoot "/var/www/larroumets.dev/public"

    <Directory "/var/www/larroumets.dev/public">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Création du VirtualHost

Une fois créé, je dois activer ce VirtualHost avec la commande :

```
sudo a2ensite larroumets-dev.conf
```

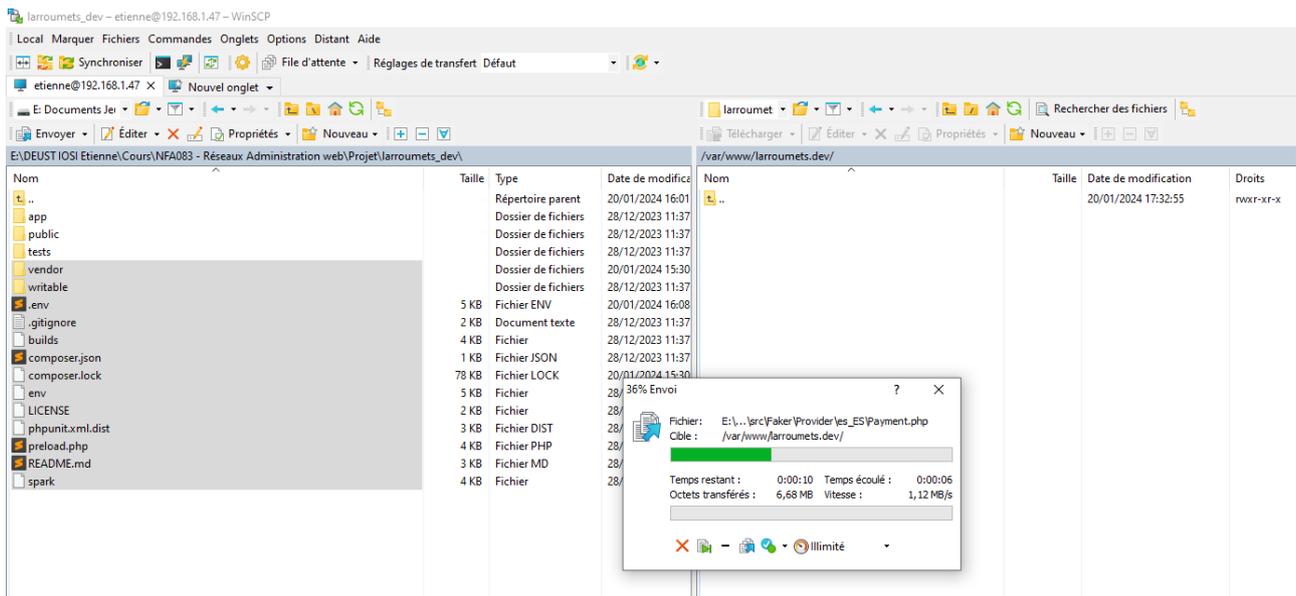
Je tente alors de redémarrer Apache mais je fais face à une erreur. Je mets un moment à comprendre que l'erreur vient de ma directive VirtualHost. En effet, j'ai voulu utiliser des fichiers logs personnalisés afin d'enregistrer les erreurs et accès vers le serveur (non présent sur la capture d'écran ci-dessus), mais Apache ne les trouvant pas refuse de redémarrer. Je supprime ces lignes de la directive pour palier au problème et décide d'utiliser les fichiers logs par défaut. J'y reviendrai plus tard.

Ensuite, je crée le dossier qui contiendra les fichiers du site et j'en modifie le propriétaire :

```
etienne@raspberrypi:~$ sudo mkdir /var/www/larroumets.dev
etienne@raspberrypi:~$ ll /var/www/
total 16
drwxr-xr-x  4 root root 4096 Jan 20 17:32 ./
drwxr-xr-x 14 root root 4096 Dec 25 19:46 ../
drwxr-xr-x  2 root root 4096 Dec 25 19:48 html/
drwxr-xr-x  2 root root 4096 Jan 20 17:32 larroumets.dev/
etienne@raspberrypi:~$ cd /var/www/
etienne@raspberrypi:/var/www$ sudo chown -R etienne larroumets.dev/
```

Création du dossier pour accueillir les fichiers du site

Puis je transfère les fichiers de mon environnement de développement vers celui de production, en les plaçant dans le dossier indiqué lors de la configuration du VirtualHost. Le transfert est effectué à l'aide de WinSCP en me connectant en SFPT à mon serveur :

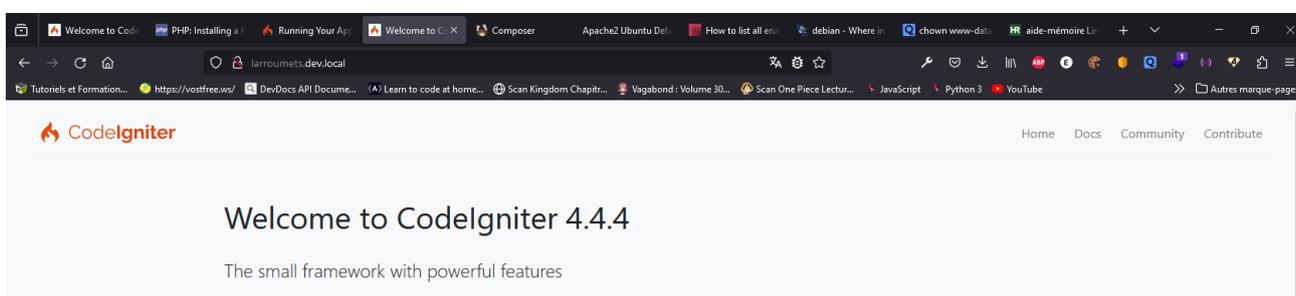


Transfert des fichiers de l'environnement de développement vers le serveur de production

Afin que le framework fonctionne correctement, je dois également apporter quelques petits changements à la configuration de Apache et de CodeIgniter semblables à ce que j'avais fait lors de l'installation de mon environnement de développement :

- activer le module vhost_alias grâce à la commande : `sudo a2enmod vhost_alias`
- activer le module rewrite grâce à la commande : `sudo a2enmod rewrite`
- permettre l'accès en écriture au groupe pour le dossier `writable` avec : `sudo chmod g+x larroumets.dev/writable`
- modifier l'URL de base dans le fichier `app/Config/App.php` du framework
- modifier la configuration de la base de données dans `app/Config/Database.php`

Une fois que tout cela est fait, je relance le serveur Apache avec : `sudo systemctl restart apache2`
Et je tente de me connecter au site dans mon navigateur à l'adresse `larroumets.dev.local` :



Test de connexion au site en local

Le serveur et le framework sont tous les deux fonctionnels !

Sécurisation de Apache

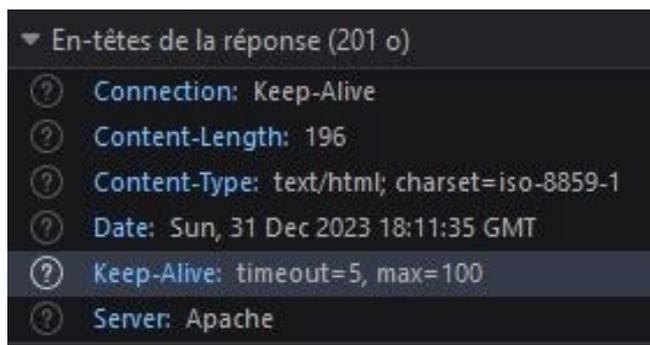
Je vais effectuer quelques petites modifications de la configuration de Apache afin d'apporter un niveau de sécurité supplémentaire au serveur.

Masquer la version de PHP et Apache :

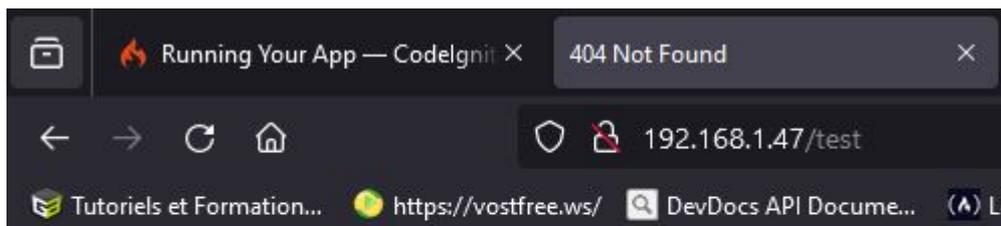
Pour commencer, je vais faire en sorte que les versions de Apache et PHP n'apparaissent pas sur la partie publique du site. Pour cela j'insère à l'aide de Vim les deux lignes suivantes dans le fichier de configuration Apache qui se trouve dans `/etc/apache2/apache2.conf`

```
ServerTokens Prod
ServerSignature Off
```

Après redémarrage de Apache, je teste le résultat :



La version de Apache n'apparaît pas



Not Found

The requested URL was not found on this server.

Aucun information sur la version de Apache

Grâce à ces deux directives, les versions Apache et PHP n'apparaissent ni dans les outils réseau du navigateur ni dans la page 404.

Prévention des attaques DOS :

Je paramètre le nombre de connexions simultanées et le nombre de connexions persistantes afin de limiter les attaques de type DOS (Denial Of Service) avec les directives suivantes

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
```

Vérifications des modules Apache

Je termine en vérifiant qu'aucun module Apache compromettant ne soit activé par défaut tels **mod_include** ou encore **mod_perl**. J'utilise la commande suivante pour afficher la liste des modules Apache :

```
apachectl -M
```

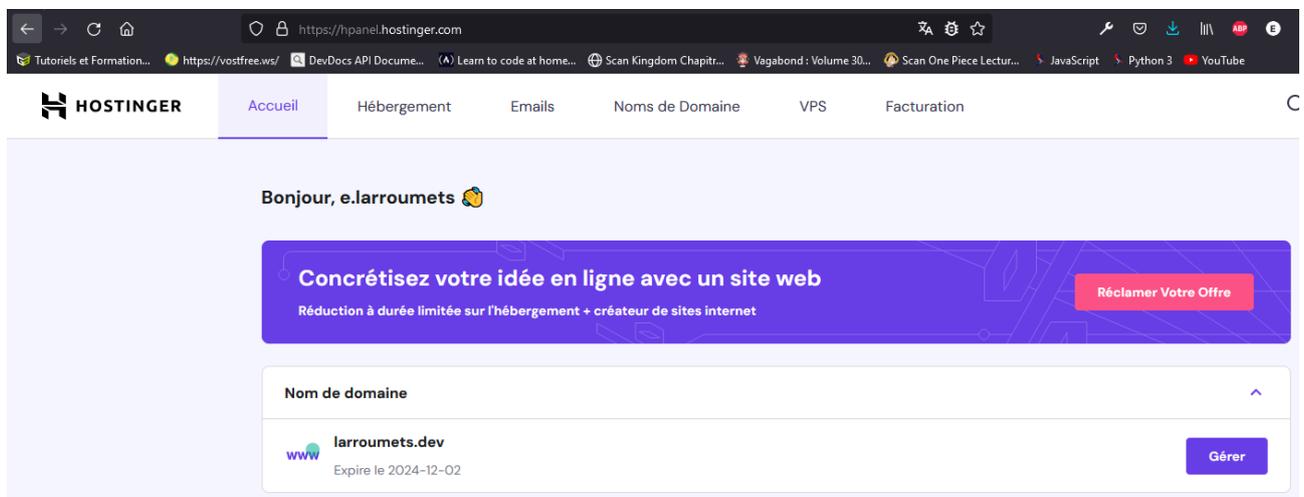
Vous pouvez retrouver la sortie de cette commande dans le fichier « modules_apache.txt » dans le dossier « annexes ».

Ouverture du serveur vers le réseau public

Acquisition du nom de domaine

La première étape est d'acquérir un nom de domaine pour que les visiteurs puissent facilement se connecter à mon site. Pour cela, je fais appel au registrar Hostinger (<https://www.hostinger.fr/>) qui propose des tarifs assez compétitifs vis-à-vis de la concurrence.

Pour les besoins du projet, j'ai déjà acquis le nom de domaine **larroumets.dev** :



Interface de gestion chez Hostinger

Configuration du nom de domaine/serveur

Je dois maintenant faire pointer mon nom de domaine vers l'adresse IP publique de mon serveur. Un problème peut alors se poser : mon FAI est Orange et ce dernier peut, si besoin, modifier mon adresse publique (déconnexion de Livebox durant un certain temps, problème sur la ligne, etc...). Il me faut donc une solution pour que mon nom de domaine s'adapte aux changements d'adresse IP publique.

Pour remédier à ce problème, j'utilise un service de DNS dynamique. C'est un serveur de nom de domaines intermédiaire auquel ma LiveBox se connecte pour transmettre à intervalles réguliers mon adresse IP publique. Ainsi je suis assuré d'avoir une adresse toujours à jour pour mon nom de domaine.

Pour ce faire j'utilise les services de Dynu.com (<https://www.dynu.com>) car ma Livebox est compatible avec ce service.

Après avoir créé mon compte sur le site de Dynu, je me connecte à l'interface de ma Livebox (192.168.1.1) afin d'y enregistrer mon login/mot de passe de Dynu et mon nom de domaine. Ainsi, elle sera capable de transmettre ses informations :

Retour Réseau

DHCP NAT/PAT DNS UPnP DynDNS DMZ NTP IPv6

Le service DynDNS permet d'attribuer un nom de domaine et d'hôte fixe, facile à mémoriser, à une adresse IP statique ou dynamique ou à une longue URL.

Utile, par exemple, si vous hébergez un site web ou un serveur FTP derrière votre Livebox pour le retrouver facilement (nom de type monserveur.dydns.org).

dyndns [Enregistrer](#)

Service	Nom d'hôte/de domaine	Email utilisateur	Mot de passe	Mise à jour	
GnuDIP	larroumets.dev	e.larroumets@hotm...	*****	20/01/2024 21:55	

Configuration du DNS dynamique dans la Livebox

Au passage, je vois que le service de DNS dynamique chez Dynu.com a été renommé en GnuDIP.

Pour connaître mon adresse IP publique, je me connecte au site : <http://ip.lafibre.info/> :

Quelle est mon adresse IPv4 / IPv6 publique ?

Connectivité IP : **Bienvenue dans l'internet du futur !**

- Connectivité IPv4 (via requête DNS) **OK** : IPv4 publique = 90.2.48.184
- Connectivité IPv4 (via IPv4 littérale) **OK** : IPv4 publique = 90.2.48.184
- Connectivité IPv6 (via requête DNS) **OK** : IPv6 publique = 2a01:cb19:8f27:5f00:657a:e098:ed2b:cc4
- La version du protocole IP utilisée par défaut est **IPv6**

Vérification de l'adresse IP publique du routeur Livebox

Mon adresse IP publique est bien à jour sur les serveurs de Dynu.com :

Dynamic DNS Service Logged in as elarroumets from IP address 90.2.48.184

Home / Control Panel / Dynamic DNS Service

Show 25 entries [+ Add](#)

Domain	IPv4	IPv6	Last Update	Actions
larroumets.dev	90.2.48.184	2a01:cb19:8f27:5f00:a6ce:daff:fe16:7232	2024/01/18 14:53:41	

Showing 1 to 1 of 1 entries ← Previous 1 Next →

Vérification de l'adresse IP du serveur chez Dynu.com

Maintenant que mon service DNS dynamique est configuré, je dois indiquer à mon registrar Hostinger ne plus chercher l'adresse IP de mon serveur dans ses serveurs DNS mais d'aller la chercher sur les serveurs de Dynu.com.

Dans un premier temps je récupère les adresses des serveurs DNS chez Dynu.com :

Sous-domaine ?

Enregistrements de serveur de noms ?

- NS1.DYNU.COM [Los Angeles, États-Unis]
- NS2.DYNU.COM [Phoenix, États-Unis]
- NS3.DYNU.COM [Atlanta, États-Unis]
- NS4.DYNU.COM [Dallas, États-Unis]
- NS5.DYNU.COM [Seattle, États-Unis]
- NS6.DYNU.COM [New Jersey, États-Unis]

- NS7.DYNU.COM [Tokyo, JP]
- NS8.DYNU.COM [Sydney, AU]
- NS9.DYNU.COM [Singapour, SG]
- NS10.DYNU.COM [Amsterdam, NL]
- NS11.DYNU.COM [Francfort, DE]
- NS12.DYNU.COM [London, UK]

Nom d'hôte ? **TTL** ?

+ Ajouter
✖ Annuler
↶ Utiliser les défauts

Enregistrements de serveur de noms

Nom d'hôte ?	Serveur de nom de domaine ?	TTL ?	Actions ?
larroumets.dev	ns1.dynu.com	3600	
larroumets.dev	ns2.dynu.com	3600	
larroumets.dev	ns3.dynu.com	3600	
larroumets.dev	ns4.dynu.com	3600	
larroumets.dev	ns5.dynu.com	3600	
larroumets.dev	ns6.dynu.com	3600	

Serveurs DNS de Dynu.com

Puis dans un second temps je me connecte à l'interface de Hostinger pour configurer les nouveaux serveurs DNS :

larroumets.dev |
 [🏠 - Noms de Domaine - larroumets.dev](#) -
 [DNS / Serveurs de noms](#)

Enregistrements DNS
Serveurs de noms enfants
DNSSEC
Transfert

Serveurs de noms

Les serveurs de noms traitent les demandes Internet pour votre domaine. Vous pouvez utiliser les serveurs de noms de Hostinger ou utiliser des serveurs de noms personnalisés pour pointer vers un autre fournisseur d'hébergement.

ns1.dns-parking.com

ns2.dns-parking.com

Sélectionnez les serveurs de noms

Utiliser les serveurs de noms Hostinger (recommandé)

Changer les serveurs de noms

Sauvegarder
Annuler

Inscription des serveurs DNS Dynu.com chez Hostinger

À la suite de ce changement, il peut se passer jusqu'à 24h pour que la prise en compte des nouveaux serveurs soit effective.

J'avais anticipé ce délai et paramétré manuellement mon adresse IP publique chez Hostinger car il y a peu de chances pour que mon adresse IP soit modifiée par mon FAI dans les prochaines 24h :

CAA	@	0	0 issuewild "globalsign.com"	14400	Supprimer	Modifier
CAA	@	0	0 issuewild "letsencrypt.org"	14400	Supprimer	Modifier
A	@	0	90.2.48.184	14400	Supprimer	Modifier

Enregistrement A (IPV4) chez Hostinger

Pour m'assurer que tout fonctionne bien, j'utilise les commandes nslookup et ping dans un invite de commande :

```
C:\Users\Maison>nslookup larroumets.dev
Serveur : livebox.home
Address: 2a01:cb19:8f27:5f00:a6ce:daff:fe16:7232

Réponse ne faisant pas autorité :
Nom : larroumets.dev
Address: 90.2.48.184
```

Test des DNS avec la commande nslookup

```
C:\Users\Maison>ping larroumets.dev

Envoi d'une requête 'ping' sur larroumets.dev [90.2.48.184] avec 32 octets de données :
Réponse de 90.2.48.184 : octets=32 temps=2 ms TTL=64
Réponse de 90.2.48.184 : octets=32 temps<1ms TTL=64
Réponse de 90.2.48.184 : octets=32 temps<1ms TTL=64
Réponse de 90.2.48.184 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 90.2.48.184:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 0ms, Maximum = 2ms, Moyenne = 0ms
```

Test de ping vers le domaine larroumets.dev

La résolution DNS fonctionne parfaitement, je vais maintenant pouvoir ouvrir les ports de ma Livebox pour rediriger les connexions entrantes sur les ports 80 et 443 vers mon serveur.

Ouverture des ports de la Livebox

Pour effectuer une redirection de ports, la première chose à faire est d'attribuer une adresse IP statique à l'interface réseau de mon Raspberry. Sans cela le serveur DHCP pourrait attribuer une adresse IP différente en cas de déconnexion/reconnexion, ce que l'on veut absolument éviter :

Baux DHCP statiques

Attribuez vous-même une adresse IP à votre équipement.

Galaxy-S7-1 ▼	192.168.1.13	4C:66:41:4D:45:EB	Ajouter
---------------	--------------	-------------------	---------

Équipement	Adresse IP statique	Adresse MAC	
PC-Bureau	192.168.1.65	2C:4D:54:58:30:7C	
ubuntuserver	192.168.1.41	08:00:27:26:8D:66	
raspberry	192.168.1.47	D8:3A:DD:83:22:40	

Attribution d'adresses IP statiques sur mon réseau local

Je peux maintenant configurer la redirection de ports :

FTP Server ▼	21 <small>ex. : 1000</small>	21 <small>ex. : 1000-2000</small>	TCP ▼	raspberry ▼	Toutes <small>IP externes autorisées</small>	Créer
--------------	---------------------------------	--------------------------------------	-------	-------------	---	-------

Activer	Application/Service	Port interne	Port externe	Protocole	Équipement	IP externe	
<input checked="" type="checkbox"/>	Web Server (HTTP)	80	80	TCP	raspberry	Toutes	
<input checked="" type="checkbox"/>	Secure Web Server (HTTPS)	443	443	TCP	raspberry	Toutes	
<input checked="" type="checkbox"/>	Secure Shell Server (SSH)	22	22	TCP	raspberry	Toutes	

Redirection des ports 22, 80, 443 de la Livebox

En paramétrant ainsi mon routeur, toutes les requêtes entrantes sur ses ports 22, 80 et 443 côté réseau public seront redirigées vers les ports 22, 80 et 443 de mon Raspberry. L'ouverture du port 22 me servira à administrer mon serveur via SSH en dehors de mon réseau local.

En testant l'accès à mon site depuis l'extérieur, l'accès par adresse IP fonctionne parfaitement mais pas l'accès par nom de domaine. Cela est dû au fait que l'accès HTTP est automatiquement transformé en accès HTTPS mais je n'ai, pour le moment, aucun VirtualHost qui écoute sur le port 443 de mon serveur. Pour remédier à cela, je vais directement installer mes certificats SSL/TLS via **Certbot** et enregistrer mon domaine auprès de **Let's Encrypt**.

Obtention et installation des certificats SSL/TLS

Je vais installer mes certificats SSL/TLS via une solution fournie par le logiciel Certbot (<https://certbot.eff.org/>) qui utilise les certificats Let's Encrypt. Certbot est disponible sur Linux. Premièrement je retire toute trace d'une éventuelle installation antérieure de Certbot :

```
sudo apt-get remove certbot
```

J'installe ensuite Certbot via Snap :

```
sudo snap install --classic certbot
```

Je crée le lien symbolique vers le lanceur de Certbot :

```
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Et je lance enfin la commande `sudo certbot --apache` pour obtenir et installer les certificats :

```
etienne@raspberrypi:~$ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log

Which names would you like to activate HTTPS for?
We recommend selecting either all domains, or all domains in a VirtualHost/server block.
-----
1: larroumets.dev
2: www.larroumets.dev
-----
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel):
Requesting a certificate for larroumets.dev and www.larroumets.dev

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/larroumets.dev/fullchain.pem
Key is saved at: /etc/letsencrypt/live/larroumets.dev/privkey.pem
This certificate expires on 2024-04-20.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

Deploying certificate
Successfully deployed certificate for larroumets.dev to /etc/apache2/sites-available/larroumets-dev-le-ssl.conf
Successfully deployed certificate for www.larroumets.dev to /etc/apache2/sites-available/larroumets-dev-le-ssl.conf
Congratulations! You have successfully enabled HTTPS on https://larroumets.dev and https://www.larroumets.dev

-----
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
-----
```

Installation des certificats SSL/TLS avec Certbot

Comme me l'indique Certbot, mes certificats sont validés et déployés. Certbot a automatiquement créé un nouveau VirtualHost : `/etc/apache2/sites-available/larroumets-dev-le-ssl.conf` écoutant sur le port 443 et incluant les certificats SSL/TLS:

```
etienne@raspberrypi: ~
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName larroumets.dev
    ServerAlias www.larroumets.dev
    DocumentRoot "/var/www/larroumets.dev/public"

    <Directory "/var/www/larroumets.dev/public">
        AllowOverride All
        Require all granted
        Allow from all
    </Directory>
    ErrorLog /var/log/apache2/errorlarroumets.log
    TransferLog /var/log/apache2/accesslarroumets.log

    Include /etc/letsencrypt/options-ssl-apache.conf
    SSLCertificateFile /etc/letsencrypt/live/larroumets.dev/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/larroumets.dev/privkey.pem
</VirtualHost>
</IfModule>
```

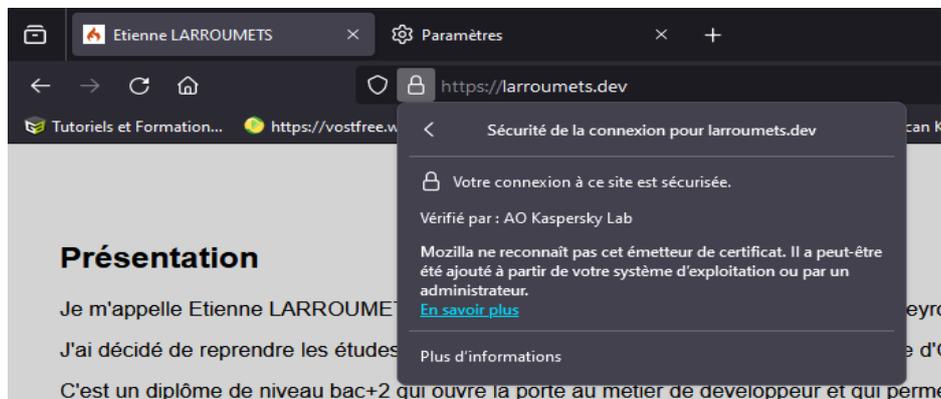
VirtualHost pour le port 443 incluant les certificats SSL/TLS

Je m'aperçois au passage qu'il a automatiquement activé le module Apache `mod_ssl`.

En retournant sur mon navigateur, je retente une connexion sur `larroumets.dev` et cette fois ci, la page d'accueil de mon site s'affiche correctement et la connexion est bien sécurisée :



Connexion réussie au site en https



Vérification de la connexion sécurisée

Activation du pare-feu UFW

À la suite de cela, je m'aperçois que je n'ai oublié d'activer le pare-feu sur le serveur. Je remédie à cela en ajoutant les règles suivantes pour le trafic entrant :

```
sudo ufw allow ssh/tcp
sudo ufw allow http/tcp
sudo ufw allow https/tcp
sudo ufw enable
```

Grace à cette configuration et à l'activation du pare-feu, les connexions entrantes sur les ports autres que les ports 22, 80 et 443 seront automatiquement bloquées.

Création de l'adresse email

Mon idée première était d'installer un serveur de messagerie Postfix sur Ubuntu Server pour pouvoir tout gérer en interne. Mais je n'ai pas réussi à faire fonctionner correctement ce serveur. En effet, la première difficulté à laquelle j'ai dû faire face est la restriction du port 25 sur ma Livebox.

Je suis parvenu à faire fonctionner le serveur mais c'est au moment de se connecter au serveur destinataire que le problème se posait (connection timed out) :

```
213 Jan 21 13:19:27 larroumets postfix/qmgr[10841]: E2DEA69C0B: from=<etienne@larroumets.dev>, size=347, nrcpt=1 (queue active)
214 Jan 21 13:19:27 larroumets postfix/qmgr[10841]: 53D9469C07: from=<etienne@larroumets.dev>, size=347, nrcpt=1 (queue active)
215 Jan 21 13:19:27 larroumets postfix/qmgr[10841]: 048DE69C01: from=<etienne@larroumets.dev>, size=347, nrcpt=1 (queue active)
216 Jan 21 13:19:57 larroumets postfix/smtp[10845]: connect to smtp2.mailfence.com[212.3.242.79]:25: Connection timed out
217 Jan 21 13:19:57 larroumets postfix/smtp[10847]: connect to smtp2.mailfence.com[212.3.242.79]:25: Connection timed out
218 Jan 21 13:19:57 larroumets postfix/smtp[10848]: connect to smtp2.mailfence.com[212.3.242.79]:25: Connection timed out
219 Jan 21 13:20:27 larroumets postfix/smtp[10845]: connect to smtp1.mailfence.com[212.3.242.78]:25: Connection timed out
```

Fichier log de la commande mail

Pour l'anecdote, depuis 2006, l'AFA (Association des fournisseurs d'accès et de services internet) recommande aux FAI de bloquer le port 25 en connexion sortante des box internet afin d'éviter le relai de spam via des botnets (ordinateurs de particuliers infectés).

J'ai donc tenté d'utiliser le protocole SMTP sécurisé avec une authentification SASL (via le port 587) mais là encore ce fut un échec. J'ai également tenté de configurer Postfix comme serveur relais pour que le courrier soit acheminé via le serveur SMTP de Orange comme j'avais pu le lire sur certains forums mais là aussi, Orange semble avoir mis en place des restrictions...

J'ai donc décidé d'utiliser un serveur de messagerie externe et d'y rattacher mon nom de domaine. Cette solution est plus adaptée, sécurisée et peu onéreuse pour peu qu'on mette un peu « les mains dans le cambouis » !

Configuration d'un serveur externe

Pour ce serveur, j'aurais pu utiliser les services de mon registrar (Hostinger) qui propose un service d'emails. Mais le problème chez Hostinger est qu'ils obligent à prendre un abonnement pour quatre ans, ce qui ne m'intéressait pas d'un point de vue financier.

J'ai donc utilisé les services de Zoho Mail (<https://www.zoho.com/fr/mail/>). Après avoir créé mon compte, j'ai dû acheter un abonnement annuel (12€ TTC/utilisateur/an).

Une fois l'abonnement confirmé, Zoho nous fournit la configuration complète du serveur que je vais pouvoir utiliser pour configurer mon serveur DNS chez Dynu.com :

Type d'enregistrement	Hôte ?	Valeur	Priorité	Statut
MX ?	@	mx.zoho.eu	10	!
	@	mx2.zoho.eu	20	!
	@	mx3.zoho.eu	50	!
SPF ?	@	v=spf1 include:zoho.eu -all	-	!
DKIM ?	zmail_domainkey	v=DKIM1; k=rsa; p=MIGfMA0GCsqGS1b3DQEBAQUAA4GNADCBiQKBgQCM9H1ILKY6UOWSJLKR68SF238tHerw3Usuq5sp7n9XegJdR1Ge+Gb3nZcXGqhsR9Y8I7Nfi5pEcRXmIXUjNOuGyiwYvLYVxY9dhQITackFfoDPq/kFA3MRpTAx1qLhVxj554nkm+KrEYINCmvOHYH4EmcfnxaWsfKgvblLaUBqJwIDAQAB	-	!

Informations de configuration des serveurs mails chez Zoho

Sur l'image ci-dessus, les statuts sont encore en rouge car j'ai fait la capture d'écran avant de configurer mes enregistrements DNS chez Dynu.com.

Le principe de fonctionnement est le suivant : quand une personne m'enverra un mail (via le protocole SMTP), son agent de transfert mail (MTA) enverra une requête vers les serveurs de Hostinger puisque mon nom de domaine est enregistré chez eux. De là, et puisque j'ai reconfiguré les serveurs DNS, Hostinger renverra la requête chez Dynu.com. Il faudra donc que j'indique à Dynu.com de rediriger cette requête vers les serveurs de Zoho pour enfin acheminer l'email.

Je peux maintenant configurer mes enregistrements DNS chez Dynu.com pour que le courrier soit acheminé vers le MTA (Mail Transfert Agent) de Zoho. Voici ce que cela donne une fois les enregistrements effectués :

Nom d'hôte	Type	Données	TTL	Actions
*.larroumets.dev	A	90.2.48.184	120	 
*.larroumets.dev	AAAA	2a01:cb19:8f27:5f00:a00:27ff:fe26:8d66	120	 
larroumets.dev	A	90.2.48.184	120	
larroumets.dev	AAAA	2a01:cb19:8f27:5f00:a6ce:daff:fe16:7232	120	
larroumets.dev	MX	mx.zoho.eu [Priority: 10]	120	  
larroumets.dev	MX	mx2.zoho.eu [Priority: 20]	120	  
larroumets.dev	SPF	v=spf1 include:zoho.eu ~all	120	  
zmail_domainkey.larroumets.dev	TXT	v=DKIM1; k=rsa; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCM9H1ILKY6UOWSJLKR G85F238tHerw3Uusuq5sp7n9XegjdR1Ge+Gb3nZcXGqhsR9Y8I7Nf5pEcRXmIXUj N0uGyiwYvLYVxY9dhQITackFfoDPq/kFA3MRpTAx1qLhVxj554nkm+KrEYINCmv OHyH4EmcfnxaWsfKgvbLaUBqjwIDAQAB	120	  

Configuration des enregistrements MX, SPF et DKIM chez Dynu.com

Dans sa version gratuite, Dynu ne propose que quatre enregistrements au maximum pour les mails (dans mon cas : deux MX + un SPF + un DKIM), je n'ai donc pas pu enregistrer le troisième serveur de messagerie **mx3.zoho.eu**. Mais cela fonctionne quand même parfaitement bien.

Après un délai de quelques minutes mes enregistrements sont pris en compte et validés. Les enregistrements MX permettent d'indiquer l'adresse des serveurs vers lesquels le courrier doit être acheminé. L'enregistrement SPF indique les serveurs autorisés à envoyer un mail pour un nom de domaine donné. Enfin l'enregistrement DKIM permet d'identifier les emails envoyés depuis mon domaine, ce qui limite l'usurpation de nom de domaine pour l'envoi de mails frauduleux (phishing).

Après avoir testé l'envoi et la réception de mails, tout est OK et fonctionne parfaitement. En complément je génère une clé PGP avec Mozilla Thunderbird afin de pouvoir échanger des mails chiffrés avec des destinataires ayant eux aussi leur clé PGP (<https://keys.openpgp.org/>).

Configuration de la sauvegarde des fichiers

Pour terminer ce projet, je vais mettre en place un système de sauvegarde automatique. Le but sera de créer des sauvegardes des dossiers contenant les fichiers du site et les fichiers de configuration de Apache. Pour effectuer cette tâche, je vais utiliser **rsnapshot**.

Rsnapshot est un utilitaire basé sur **rsync**. Il permet de faire de la sauvegarde incrémentielle tout en évitant la duplication de fichiers lorsque ceux-ci sont inchangés. Cela est rendu possible grâce aux liens physiques. En effet, lors d'une sauvegarde si les fichiers sont similaires à la sauvegarde précédente un lien physique est créé vers cette sauvegarde évitant ainsi de surcharger le support de stockage des sauvegardes.

La force de **rsnapshot** réside également dans sa simplicité de configuration. Un seul fichier est à configurer pour avoir un système opérationnel. Viens ensuite la configuration de l'automatisation des sauvegardes grâce à **crontab**.

Montage de la clé USB de stockage

Je vais stocker mes sauvegardes sur une clé USB branchée en permanence sur mon Raspberry Pi. De cette manière, je pourrai les récupérer à la fois de manière physique mais aussi via le réseau en SFTP avec par exemple WinSCP.

Je commence par monter ma clé USB préalablement formatée en ext4 dans le dossier `/media/usb/` :

Après avoir repéré l'identifiant (`sda1`) de ma clé USB grâce à la commande `lsblk`, je monte cette dernière dans un dossier :

```
cd /media
sudo mkdir usb
sudo mount -t ext4 /dev/sda1 /media/usb
```

Il est important d'avoir une clé USB formatée en système de fichier ext2, ext3 ou ext4 car seuls ces systèmes de fichiers permettent la création de liens physiques ! Ce détail m'a causé pas mal de soucis car au départ j'utilisais une clé formatée en FAT32. Il m'a fallu un moment pour comprendre mon erreur.

Installation et configuration de rsnapshot

Rsnapshot s'installe de manière classique :

```
sudo apt install rsnapshot
```

Une fois installé, il faut éditer la configuration selon mes besoins. Une des forces de rsnapshot est sa facilité de configuration. En effet tout se passe dans le fichier `rsnapshot.conf` que j'édite de la façon suivante :

```
sudo nano /etc/rsnapshot.conf
```

Dans ce fichier, je configure les dossiers dans lesquels enregistrer les sauvegardes et le nombre de sauvegardes à conserver pour chaque intervalle de sauvegarde choisi :

Je joins les captures d'écran des directives les plus importantes mais le **fichier complet est disponible dans le dossier « Annexes »**.

```
#####
# SNAPSHOT ROOT DIRECTORY #
#####
# All snapshots will be stored under this root directory.
#
snapshot_root    /media/usb/backup_rsnapshot/
```

Configuration du dossier racine des sauvegardes

```
#####
#      BACKUP LEVELS / INTERVALS      #
# Must be unique and in ascending order #
# e.g. alpha, beta, gamma, etc.      #
#####
retain hourly 6
retain daily 7
retain weekly 4
retain monthly 3
```

Configuration du nombre de sauvegardes à conserver pour chaque intervalle de sauvegarde

Pour la configuration ci-dessus, rsnapshot conservera six sauvegardes de type « hourly », sept de type « daily », etc... Puis effectuera un roulement à chaque nouvelle sauvegarde, écrasant la plus ancienne.

```
#####
### BACKUP POINTS / SCRIPTS ###
#####
# LOCALHOST
backup /var/www/larroumets.dev/ localhost/
backup /etc/apache2/ localhost/
#backup /usr/local/ localhost/
#backup /var/log/rsnapshot localhost/
```

Dossiers à sauvegarder

Après avoir enregistré les changements, la commande `rsnapshot configtest` nous permet de confirmer la bonne syntaxe du fichier :

```
etienne@larroumets:~$ rsnapshot configtest
Syntax OK
```

Vérification de la bonne syntaxe du fichier

La commande `rsnapshot hourly` nous permet de tester rapidement le fonctionnement de la sauvegarde :

```
etienne@larroumets:~$ sudo rsnapshot hourly
[sudo] password for etienne:
Sorry, try again.
[sudo] password for etienne:
etienne@larroumets:~$ sudo rsnapshot hourly
etienne@larroumets:~$ date
Thu Jan 25 20:47:47 CET 2024
etienne@larroumets:~$ ll /media/usb/backup_rsnapshot/
total 32
drwx----- 8 etienne etienne 4096 Jan 25 20:47 ./
drwxr-xr-x 4 etienne root 4096 Jan 24 13:38 ../
drwxr-xr-x 3 etienne etienne 4096 Jan 25 20:47 hourly.0/
drwxr-xr-x 3 etienne etienne 4096 Jan 25 20:47 hourly.1/
drwxr-xr-x 3 etienne etienne 4096 Jan 25 20:00 hourly.2/
drwxr-xr-x 3 etienne etienne 4096 Jan 25 16:00 hourly.3/
drwxr-xr-x 3 etienne etienne 4096 Jan 25 12:00 hourly.4/
drwxr-xr-x 3 etienne etienne 4096 Jan 25 08:00 hourly.5/
etienne@larroumets:~$
```

Test d'une sauvegarde hourly

Pour les besoins du projet, le système de sauvegardes est déjà opérationnel à l'heure à laquelle j'écris ces lignes. C'est pour cela que l'on voit des sauvegardes déjà présentes.

Sur la capture ci-dessus, on voit que j'exécute deux fois une sauvegarde « hourly » à 20h47. Ces deux sauvegardes prennent la place de « hourly.0 » et « hourly.1 ».

Automatisation des sauvegardes

Pour rendre les sauvegardes automatiques, j'utiliserai l'utilitaire bien connu de linux : **cron**. Cet utilitaire permet de lancer automatiquement, à des intervalles définis par l'utilisateur, des programmes, des scripts, des commandes, etc...

Les tâches à exécuter sont définies dans le fichier **crontab**. Chaque utilisateur possède son fichier crontab. Pour éditer le fichier j'utilise la commande : `sudo crontab -e`. J'utilise `sudo` car les tâches crontab seront exécutées par le système et non pas par un utilisateur .

Cette commande ouvre le fichier dans lequel je vais paramétrer les intervalles de sauvegarde. J'en profite au passage pour paramétrer l'automatisation de sauvegarde de la base de données « larroumets » :

```
# m h dom mon dow command
0 */4 * * * /usr/bin/rsnapshot hourly
30 23 * * * /usr/bin/rsnapshot daily
0 23 * * 1 /usr/bin/rsnapshot weekly
30 22 1 * * /usr/bin/rsnapshot monthly
20 */4 * * * /usr/bin/mysqldump larroumets > /media/usb/backup_db/larroumets.sql
40 1 * * * /usr/bin/mysqldump larroumets > /media/usb/backup_db/larroumets_daily.sql
```

Configuration des tâches cron pour l'automatisation

Mes sauvegardes seront effectuées aux intervalles suivants :

- Une toutes les quatre heures (hourly)
- Une tous les jours à 23h30 (daily)
- Une tous les lundis à 23h (weekly)
- Une tous les premiers jours du mois à 22h30 (monthly)

Pour ma base de données crontab exécutera une sauvegarde :

- Toutes les quatre heures
- Tous les jours à 1h40

Conclusion

Récapitulatif du projet :

- Installation d'un environnement de développement sous Windows
- Installation de Ubuntu Server sur un nano ordinateur à processeur ARM
- Installation d'un serveur Apache + PHP
- Installation d'une base de données MySQL
- Sécurisation de Apache et MySQL
- Installation du framework CodeIgniter 4
- Création du nom de domaine
- Configuration des DNS
- Configuration de la LiveBox pour accès depuis l'extérieur
- Création d'une adresse email
- Configuration des enregistrements MX, SPF et DKIM pour les DNS mails
- Configuration de la sauvegarde des fichiers et de la base de données

Ce projet et cette UE ont été pour moi l'occasion de découvrir de nouveaux aspects de l'administration web. J'ai énormément appris, notamment sur le fonctionnement des services mails. J'ai pu approfondir mes connaissances en ce qui concerne Apache et les services DNS.

Cela faisait longtemps que je souhaitais créer un serveur domestique et cette UE a été l'occasion de concrétiser ce projet. Dans un avenir proche ce serveur sera doté d'un service de diffusion multimédia pour les films et séries. Il me servira également à des projets Arduino liés à la domotique.