

Installation du serveur

Plan

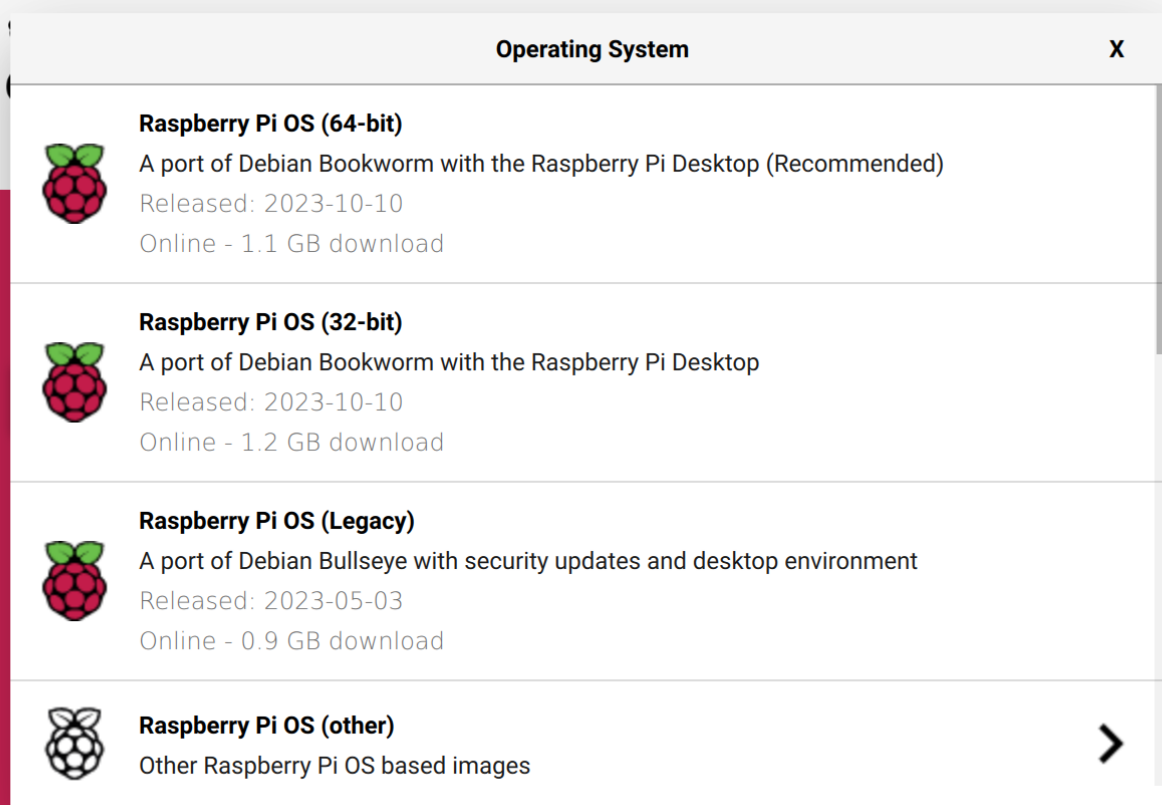
1. Installation du système d'exploitation
2. Installation du serveur en lui même
 - 2.1. Service SSH
 - 2.2. Service HTTP
 - 2.3. Installation de PHP
 - 2.4. Service MySQL
 - 2.5. Installation de PhpMyAdmin
 - 2.6. Service fail2ban
3. Comment mettre le site sur le serveur

1. Installation du système d'exploitation

Il faut d'abord installer **Raspberry Pi Imager**, étant sur un environnemen linux (Arch Linux) j'ai effectué une installation sur **Flatpak** :

```
flatpak install flathub org.raspberrypi.rpi-imager
```

une fois l'application installé j'ai flashé la carte SD avec une une version de Debian **Raspberry Pi OS (64-bit)**



2. Installation du serveur en lui même

Une fois le système d'exploitation j'ai renommé le nom de la machine en accord avec les membres de mon groupe dans les fichiers `/etc/hostname` et `/etc/hosts`

2.1. Service ssh

L'installation d'un service SSH nous permettra d'accéder au serveur à distance afin d'effectuer des maintenances ou de pouvoir mettre sur le serveur les nouvelles version du site web grâce au protocole SFTP

```
sudo apt install ssh
```

une fois installé :

```
sudo systemctl start ssh  
sudo systemctl enable ssh
```

cela aura pour effet d'exécuter le service grâce au paramètre `start` mais aussi de l'exécuter à chaque redémarrage grâce au paramètre `enable`.

Une fois sur un autre poste il suffit d'entrer la commande suivante pour accéder au serveur à distance :

```
ssh pisae@192.168.1.57
```

2.2 Service HTTP

Il suffit d'installer le service `apache2` qui sera le serveur HTTP :

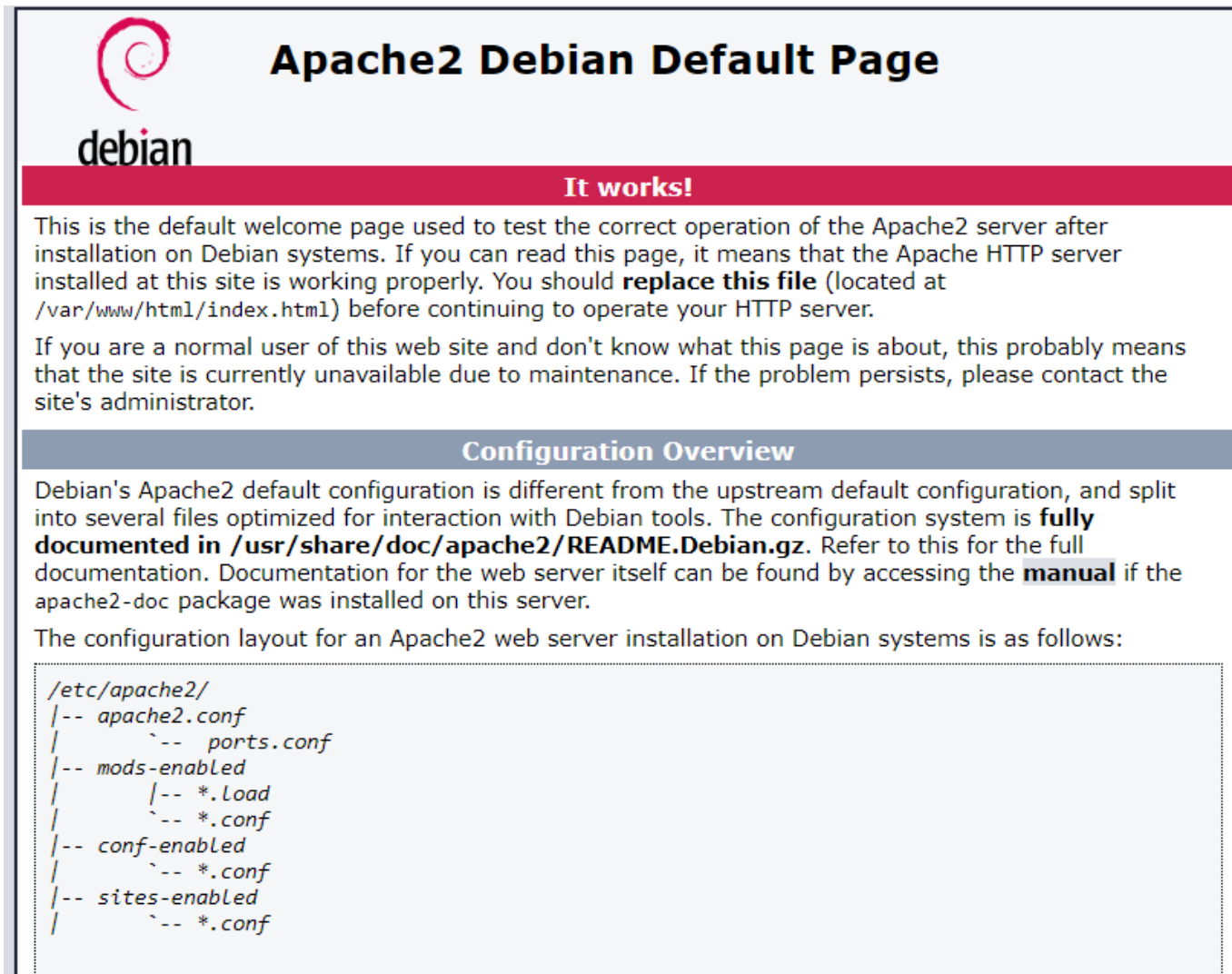
```
sudo apt install apache2
```

Une fois installé :

```
sudo systemctl start http  
sudo systemctl enable http
```

cela aura pour effet d'exécuter le service grâce au paramètre `start` mais aussi de l'exécuter à chaque redémarrage grâce au paramètre `enable`.

Pour vérifier si l'installation s'est bien faite il faut aller sur `localhost` ou `127.0.0.1` et cette page est censée être affichée :



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

Une fois le service installé avec succès il faut mettre les pages web qui nous intéressent dans le répertoire `/var/www/html` mais dans notre cas nous avons des restrictions à mettre en place car des données sensibles seront manipulées dans le cadre de notre site de ticketing comme les logins pour accéder à la base de données ou encore la clé de cryptage (nous utilisons un cryptage symétrique ce qui implique qu'avec la clé de cryptage n'importe quelle donnée peut être décryptée comme cryptée avec la même clé). Pour faire cela nous avons créé un fichier de configuration externe `/etc/apache2/sites-enabled/myconf.conf` qui sera inclus dans le fichier de configuration principal `/etc/apache2/apache2.conf`.

Nous avons donc rajouté la ligne suivante à la fin de `apache2.conf` :

```
IncludeOptional sites-enabled/*.conf
```

ce qui aura pour effet d'inclure tous les fichiers de configuration dans le répertoire `/etc/apache2/sites-enabled/`.

Et voici le contenu de notre fichier `myconf.conf` :

```
Define APACHE_LOG_DIR /var/log/apache2

<Directory "/var/www/html/src">
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
    IndexIgnore *
</Directory>

<Directory "/var/www/html/src/Vues">
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory "/var/www/html/src/json">
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory "/var/www/html/src/controllers">
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory "/var/www/html/src/log">
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory "/var/www/html/src/_medias">
    Options FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

<Directory "/var/www/html/src/Modeles">
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory "/var/www/html/src/script">
```

```
Options FollowSymLinks
AllowOverride None
Require all denied
</Directory>

<files "index.php">
    require all granted
</files>

<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/src

    <Directory /var/www/html/src>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

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

Pour comprendre plus en détail notre configuration voici l'arborescence de notre site qui nous le rappelle respecte l'architecture MVC (Modèle Vue Contrôleur):

```
/var/www/html/src/
|-- controllers
|   |-- dashboardController.php
|   |-- profileController.php
|   `-- userController.php
|-- index.php
|-- json
|   |-- key.json
|   `-- logins.json
|-- log
|   `-- history.log
|-- _medias
|   `-- LOGO_TicketOpia.svg
|-- Modeles
|   |-- Connexion.php
|   |-- LabelFunc.php
|   |-- logsFunc.php
|   |-- rc4.php
|   |-- Ticket.php
|   `-- User.php
|-- style.css
`-- Vues
    |-- accueil.php
```

```
|-- footer.php
|-- header.php
|-- Label
|   |-- dashboardLabel.php
|   |-- formAjouterLabel.php
|   |-- formModifierLabel.php
|-- logs.php
|-- Profile
|   |-- formPwd.php
|   |-- profile.php
|-- stats.php
|-- Ticket
|   |-- dashboard.php
|   |-- formModifierEtat.php
|   |-- formTicket.php
|   |-- myTickets.php
|-- User
|   |-- formAssignerTec.php
|   |-- formConnexion.php
|   |-- formCreationTec.php
|   |-- formInscription.php
|-- video.php
```

2.3 Installation de PHP

Une fois le service http installé il faut aussi installer **php** grâce à la commande :

```
sudo apt install php libapache2-mod-php php-mysql
```

Pour vérifier que php est bien installé :

```
php -v
```

2.4 Service MySQL

La commande suivante permet l'installation du service MySQL :

```
sudo apt install mariadb-server
```

une fois installé :

```
sudo systemctl start mysql
sudo systemctl enable mysql
```

cela aura pour effet d'exécuter le service grâce au paramètre **start** mais aussi de l'exécuter à chaque redémarrage grâce au paramètre **enable**.

Puis pour finaliser l'installation de la base de donnée il faut exécuter le programme **mysql_secure_installation**. Une fois l'installation terminée on peut utiliser la commande **mysql** :

```
mysql -u root
```

2.5 Installation de PhpMyAdmin

Le paquet **wget** qui permettra de récupérer les archives nécessaire à la mise en place des pages PhpMyAdmin

```
sudo apt install wget
```

J'ai choisi de télécharger la dernière version en date de PhpMyAdmin (la **5.2.1**) :

```
wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.tar.gz
```

dans le répertoire de téléchargement, extraire l'archive :

```
tar xvf phpMyAdmin-5.2.1-all-languages.tar.gz
```

puis déplacer cette extraction dans le dossier **/usr/share/phpmyadmin** :

```
sudo mv phpMyAdmin-*/ /usr/share/phpmyadmin
```

créer un dossier temporaire qui nous servira plus tard :

```
sudo mkdir -p /var/lib/phpmyadmin/tmp  
sudo chown -R www-data:www-data /var/lib/phpmyadmin
```

créer un répertoire pour les fichiers de configuration de phpMyAdmin tels que le fichier htpass :

```
sudo mkdir /etc/phpmyadmin/  
sudo cp /usr/share/phpmyadmin/config.sample.inc.php  
/usr/share/phpmyadmin/config.inc.php
```

editer le fichier `/usr/share/phpmyadmin/config.inc.php` :

```
sudo nano /usr/share/phpmyadmin/config.inc.php
```

et il rajouter la ligne qui indique notre répertoire temporaire créer précédement :

```
$cfg['TempDir'] = '/var/lib/phpmyadmin/tmp';
```

puis il faut créer un fichier configuration pour qu'elle soit répertoriée sur le serveur apache à l'adresse `localhost/phpmyadmin` :

```
sudo nano /etc/apache2/conf-enabled/phpmyadmin.conf
```

pour cela j'ai choisis un script trouvé sur internet, je l'ai évidemment modifié pour qu'il convienne à mon serveur :

```
Alias /phpmyadmin /usr/share/phpmyadmin

<Directory /usr/share/phpmyadmin>
    Options SymLinksIfOwnerMatch
    DirectoryIndex index.php

    <IfModule mod_php5.c>
        <IfModule mod_mime.c>
            AddType application/x-httpd-php .php
        </IfModule>
        <FilesMatch ".+\.php$">
            SetHandler application/x-httpd-php
        </FilesMatch>

        php_value include_path .
        php_admin_value upload_tmp_dir /var/lib/phpmyadmin/tmp
        php_admin_value open_basedir
/usr/share/phpmyadmin/:/etc/phpmyadmin/:/var/lib/phpmyadmin/:/usr/share/php
/php-gettext/:/usr/share/php/php-php-
gettext/:/usr/share/javascript/:/usr/share/php/tcpdf/:/usr/share/doc/phpmya
dmin/:/usr/share/php/phpseclib/
        php_admin_value mbstring.func_overload 0
    </IfModule>
    <IfModule mod_php.c>
        <IfModule mod_mime.c>
            AddType application/x-httpd-php .php
        </IfModule>
        <FilesMatch ".+\.php$">
            SetHandler application/x-httpd-php
        </FilesMatch>
```



```

        php_value include_path .
        php_admin_value upload_tmp_dir /var/lib/phpmyadmin/tmp
        php_admin_value open_basedir
/usr/share/phpmyadmin/:/etc/phpmyadmin/:/var/lib/phpmyadmin/:/usr/share/php
/php-gettext/:/usr/share/php/php-php-
gettext/:/usr/share/javascript/:/usr/share/php/tcpdf/:/usr/share/doc/phpmya
dmin/:/usr/share/php/phpseclib/
        php_admin_value mbstring.func_overload 0
    </IfModule>

</Directory>

# Authorize for setup
<Directory /usr/share/phpmyadmin/setup>
    <IfModule mod_authz_core.c>
        <IfModule mod_authn_file.c>
            AuthType Basic
            AuthName "phpMyAdmin Setup"
            AuthUserFile /etc/phpmyadmin/htpasswd.setup
        </IfModule>
        Require valid-user
    </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>
<Directory /usr/share/phpmyadmin/setup/lib>
    Require all denied
</Directory>
Alias /phpmyadmin /usr/share/phpmyadmin

<Directory /usr/share/phpmyadmin>
    Options SymLinksIfOwnerMatch
    DirectoryIndex index.php

    <IfModule mod_php5.c>
        <IfModule mod_mime.c>
            AddType application/x-httpd-php .php
        </IfModule>
        <FilesMatch ".+\.php$">
            SetHandler application/x-httpd-php
        </FilesMatch>

        php_value include_path .
        php_admin_value upload_tmp_dir /var/lib/phpmyadmin/tmp
        php_admin_value open_basedir
/usr/share/phpmyadmin/:/etc/phpmyadmin/:/var/lib/phpmyadmin/:/usr/share/php
/php-gettext/:/usr/share/php/php-php-

```

```

gettext:/usr/share/javascript:/usr/share/php/tcpdf:/usr/share/doc/phpmya
dmin:/usr/share/php/phpseclib/
    php_admin_value mbstring.func_overload 0
</IfModule>
<IfModule mod_php.c>
    <IfModule mod_mime.c>
        AddType application/x-httpd-php .php
    </IfModule>
    <FilesMatch ".+\.php$">
        SetHandler application/x-httpd-php
    </FilesMatch>

    php_value include_path .
    php_admin_value upload_tmp_dir /var/lib/phpmyadmin/tmp
    php_admin_value open_basedir
/usr/share/phpmyadmin:/etc/phpmyadmin:/var/lib/phpmyadmin:/usr/share/php
/php-gettext:/usr/share/php/php-php-
gettext:/usr/share/javascript:/usr/share/php/tcpdf:/usr/share/doc/phpmya
dmin:/usr/share/php/phpseclib/
    php_admin_value mbstring.func_overload 0
</IfModule>

</Directory>

# Authorize for setup
<Directory /usr/share/phpmyadmin/setup>
    <IfModule mod_authz_core.c>
        <IfModule mod_authn_file.c>
            AuthType Basic
            AuthName "phpMyAdmin Setup"
            AuthUserFile /etc/phpmyadmin/htpasswd.setup
        </IfModule>
        Require valid-user
    </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>
<Directory /usr/share/phpmyadmin/setup/lib>
    Require all denied
</Directory>

```

enfin il suffit de redémarrer le service **apache2** :

```
sudo systemctl restart apache2
```

Maintenant que **PhpMyAdmin** est fonctionnel il suffit de s'y connecter à `localhost/phpmyadmin` avec les login de la base de donnée qui a été installée précédement.



2.6. Service fail2ban

Le service **fail2ban** est disponible sur apt :

```
sudo apt install fail2ban
```

Il faut démarrer le service et faire en sorte qu'il se lance à chaque redémarrage :

```
sudo systemctl start fail2ban  
sudo systemctl enable fail2ban
```

Normalement le service devrait fonctionner par défaut :

```
systemctl status fail2ban
```

Maintenant le service est certes installé mais doit être configuré dans un fichier **myconfig.conf** que j'ai créé dans `/etc/fail2ban/jail.d/jail.d/` :

```
sudo touch /etc/fail2ban/jail.d/jail.d/myconfig.conf
```

Dans ce fichier configuration j'ai mis les lignes suivantes :

```
[DEFAULT]
ignoreip = 127.0.0.1
findtime = 600
bantime = 1800
maxretry = 3

[sshd]
enabled = true
port = 22
logpath = /var/log/auth.log
maxretry = 3
findtime = 600
bantime = 1800
```

Dans la configuration ci dessus j'ai mis un temps de ban de 30 minutes avec un temps entre chaque "maxretry" de 10 minutes avec un maxretry de 3 (essai maximum de 3)

J'ai aussi inclus ce fichier config dans le fichier configuration par défaut :

```
[INCLUDES]

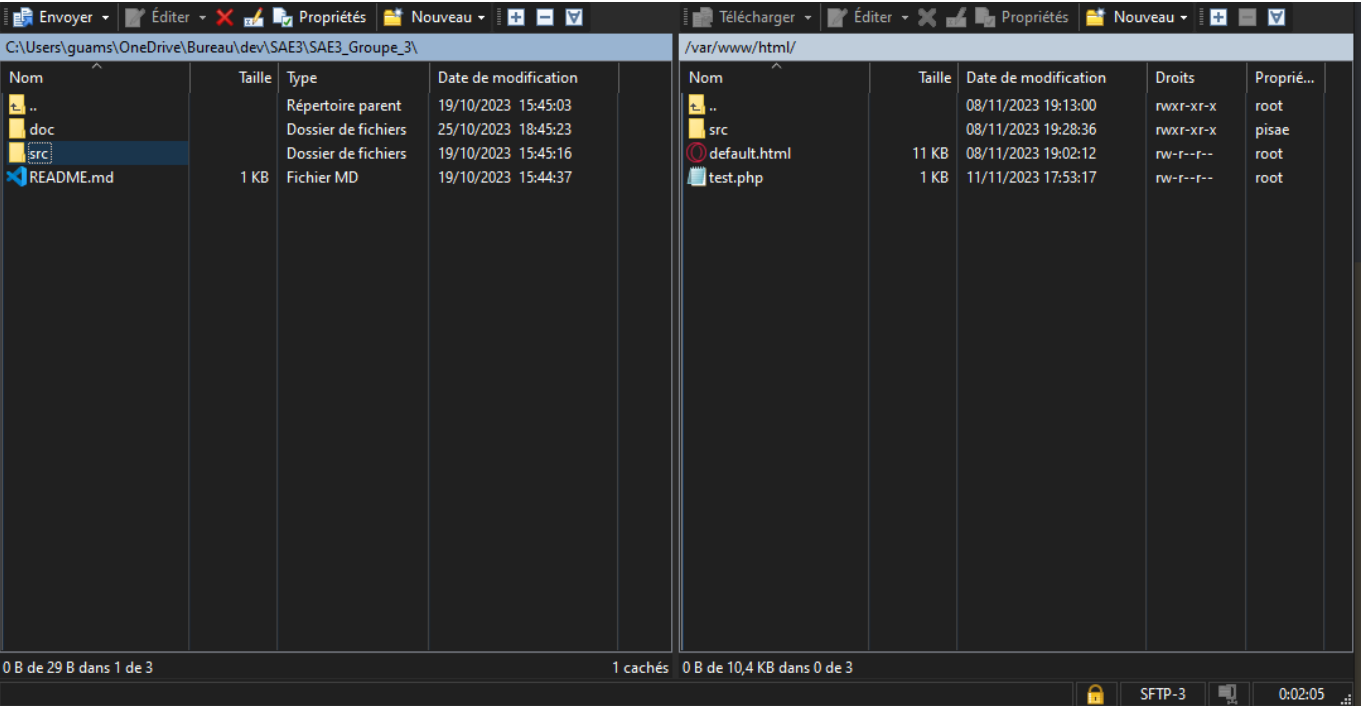
#before = paths-distro.conf
before = paths-debian.conf /etc/fail2ban/myconf.conf
```

maintenant il suffit de redémarrer le service pour tester :

```
sudo systemctl restart fail2ban
```

3. Comment mettre le site sur le serveur

Il suffit d'utiliser le protocole SFTP grâce à un logiciel comme FileZilla ou WinSCP. Une fois la connexion avec le serveur établi avec le protocole SFTP le répertoire du serveur se situe dans `/var/www/html/src`. Il suffit de glisser/déposer les fichiers nécessaire :



Il ne suffit plus qu'à rentrer les scripts SQL donnés sur le repo github et le site est prêt à l'usage !

