Raspberry Pi

Mise en place d’un serveur web pour le développement

Table des matières

Introduction

Le but de cette documentation est de mettre en place l'infrastructure pour faire du développement web, typiquement, la pile LAMP : Linux Apache, PHP, MySQL

On va donc monter un serveur web sur un Raspberry, avec tout le nécessaire.

On verra ensuite les principaux outils nécessaires, comme Git ou Visual Studio Code.

Créer un nouvel utilisateur

On va créer un nouvel utilisateur, c’est mieux pour la sécurité.

```
sudo adduser patrice
```

Et on l’ajoute au groupe www-data, le groupe d’Apache2 :

```
sudo usermod -a -G www-data patrice
```

On l’ajoute au groupe sudoer :

```
sudo adduser patrice sudo
```

On redémarre le système et on se connecte avec le nouvel utilisateur. Puis on supprime l’utilisateur pi :

```
sudo deluser -remove-home pi
```

Définir le hostname

```
sudo hostnamectl set-hostname hostname
```

Vous pouvez vérifier ensuite, après redémarrage, en tapant *hostname* dans la console.

Sécurité du serveur

Un serveur web est une pièce importante d'un système informatique et puisqu'il est directement (souvent) relié à internet, il se doit d'être protégé comme il faut. La première des choses à faire est d'avoir des mots de passe fort et de le maintenir à jour.

Configuration du serveur SSH

On commence par vérifier que le serveur fonctionne :

```
sudo service ssh status
```

On configure un peu le serveur SSH, via le fichier de configuration *sshd_config* dans */etc/ssh* pour interdire la connexion en root, en ajoutant à la fin du fichier :

```
PermitRootLogin no
```

On vérifie que seul le protocole en version 2 est accepté :

```
Protocol 2
```

On se connecte via clé SSH. On crée sa clé sur son système (*ssh-keygen -t rsa -b 4096*) et on l'envoie sur le serveur avec *ssh-copy-id -i ~/.ssh/rsa_key.pub user@IP*

Il faut interdire la connexion par mot de passe :

```
PasswordAuthentication no
```

On se connecte via une clé SSH :

```
PubkeyAuthentication yes
```

On change le port par défaut :

```
Port 4422
```

On écoute une seule adresse :

```
ListenAdrress 192.168.1.1
```

Avec `AllowUsers` et `DenyUsers` on spécifie qui a le droit ou non de se connecter.

On modifie le nombre de tentative de connexions autorisées avec :

```
MaxAuthTries 3
```

On enregistre, puis on redémarre le service SSH :

```
sudo service ssh restart
```

Pour afficher un message de bienvenue en connexion SSH, on modifie `/etc/ssh/sshd_config` pour mettre le chemin du fichier qui contient le message.

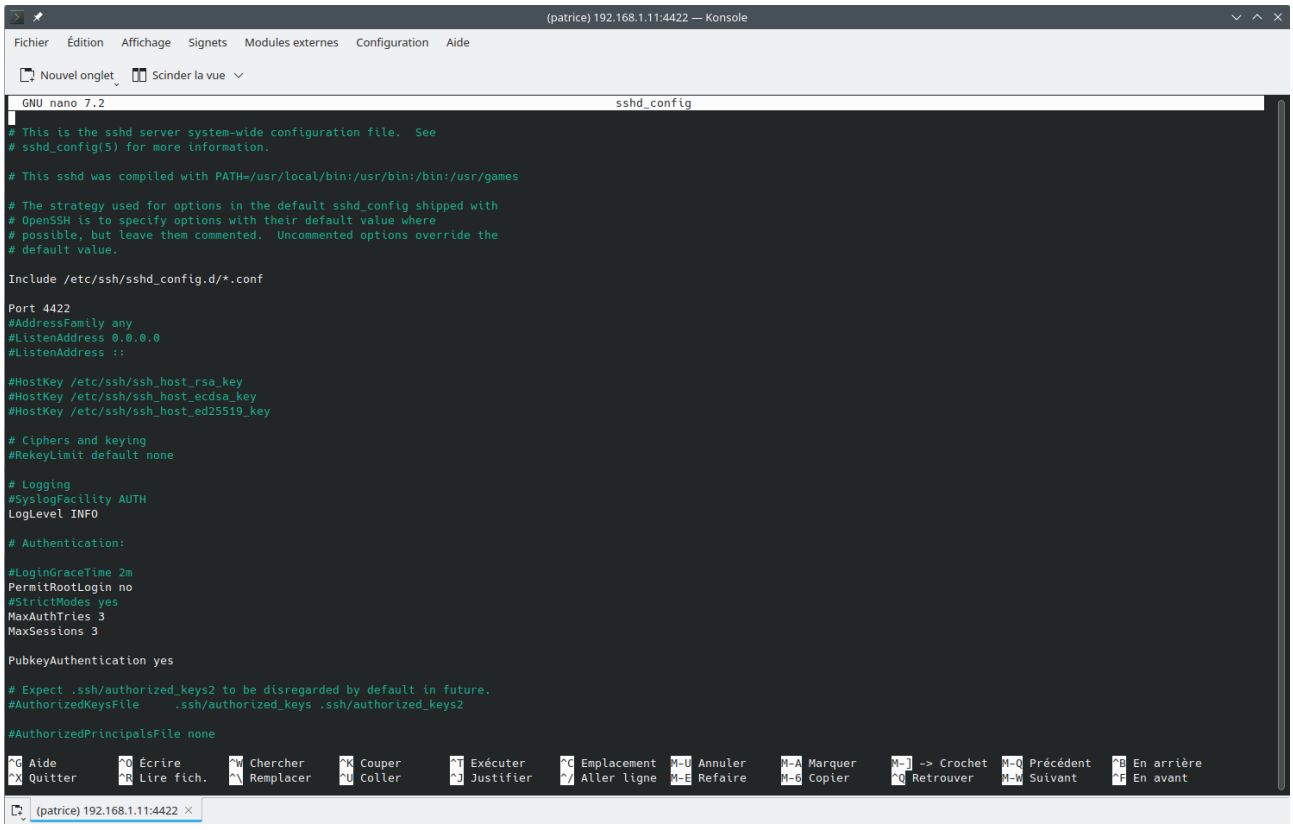
```
Banner /etc/ssh/message
```

Et on crée le fichier, on met le message de bienvenue.

On peut aussi modifier MOTD (message of the day) via les scripts dans `/etc/update-motd.d`

Voir ici :

https://github.com/ar51an/raspberrypi-motd



Mise en place d’un firewall

Il faut mettre en place un firewall, afin de protéger les ports de votre serveur. Vous pouvez vérifier les ports de votre machine à l’aide de Nmap par exemple : `nmap IP`

Ensuite, il faut installer ufw et on l’active :

```
sudo apt-get install ufw
sudo ufw status
sudo ufw enable
```

Pour voir les règles actuelles :

```
sudo ufw status verbose
```

Pour activer la journalisation :

```
sudo ufw logging on
```

Pour ajouter de règles (et donc autoriser un certain type de trafic sur certains ports) voici des exemples. Il ne faut bien sûr autoriser que le strict minimum (ports 443 et le ssh).

```
sudo ufw  allow 4422/tcp
sudo ufw allow  smtp
sudo ufw allow proto tcp from 1.2.3.4 to any port 514
```

On peut lister les règles par numéro :

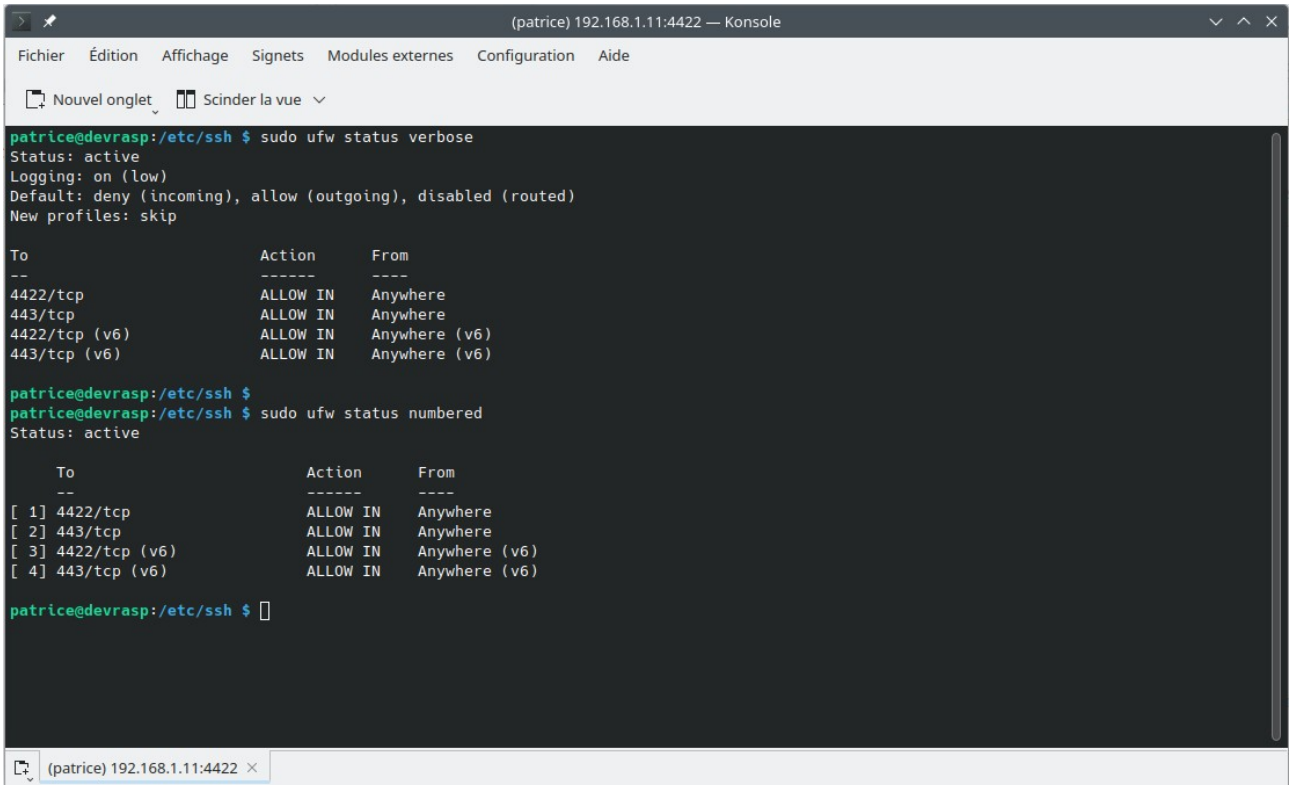
```
sudo ufw status numbered
```

Pour en supprimer une :

```
sudo ufw delete numero_regle
```

Pour blocker des IP par pays :

https://www.ustoopia.nl/technical/block-all-traffic-from-a-geo-located-country-with-ufw-firewall-on-ubuntu/


(patrice) 192.168.1.11:4422 — Konsole
Fichier Édition Affichage Signets Modules externes Configuration Aide
Nouvel onglet Scinder la vue
patrice@devrasp:/etc/ssh $ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
To Action From
-- ------ ----
4422/tcp ALLOW IN Anywhere
443/tcp ALLOW IN Anywhere
4422/tcp (v6) ALLOW IN Anywhere (v6)
443/tcp (v6) ALLOW IN Anywhere (v6)
patrice@devrasp:/etc/ssh $
patrice@devrasp:/etc/ssh $ sudo ufw status numbered
Status: active
To Action From
-- ------ ----
[1] 4422/tcp ALLOW IN Anywhere
[2] 443/tcp ALLOW IN Anywhere
[3] 4422/tcp (v6) ALLOW IN Anywhere (v6)
[4] 443/tcp (v6) ALLOW IN Anywhere (v6)
patrice@devrasp:/etc/ssh $
(patrice) 192.168.1.11:4422

Installer les paquets nécessaires

Installation d'Apache

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install apache2
```

Pour tester, rendez-vous dans votre navigateur à l'adresse :

http://127.0.0.1

Vous devez avoir la page par défaut du serveur web Apache.

On va modifier les droits et propriétaires du dossier par défaut où sont stockés les sites web.

```
sudo chown -R user:www-data /var/www/html/
sudo chmod -R 770 /var/www/html
```

On active quelques modules d'Apache :

```
sudo a2enmod rewrite
sudo a2enmod ssl
sudo a2enmod headers
```

Mesure de base pour la sécurité, on cache les informations du serveur, qui apparaissent en cas d'erreur (nom du serveur, version, adresse). Moins l'attaquant à d'informations, plus c'est compliqué.

Dans `/etc/apache2/conf-available/security.conf` on modifie ces 2 variables :

```
servertokens prod
serversignature off
```

Editer ce fichier pour ajouter à la fin le serveur name :

```
sudo nano /etc/apache2/apache2.conf
```

```
ServerName 127.0.0.1
```

Utilisez cette commande pour vérifier la configuration d'Apache :

```
sudo apache2ctl -t
```

Installation de PHP

```
sudo apt-get install php
```

On peut aussi installer ceux-ci :

```
sudo apt-get install php-zip php-gd php-curl php-intl php-xml php-mbstring php-mysql php-xdebug
```

Puis on les active :

```
phpenmod zip gd curl intl
```

Et on redémarre Apache :

```
sudo systemctl restart apache2
```

Pour vérifier le fonctionnement, on va créer un fichier php info.

On se rend dans le répertoire du site par défaut :

```
cd /var/www/html
```

```
nano info.php
```

On remplit le fichier avec ceci :

```
<?php
```

```
phpinfo();
```

```
?>
```

Crtl + o pour enregistrer le fichier, y pour confirmer, puis Ctrl + x pour fermer l'éditeur et revenir dans le terminal.

Dans le navigateur, on se rend à l'adresse :

http://127.0.0.1/info.php

Une page avec toute la configuration de PHP doit s'afficher.

Configuration de PHP pour la taille des fichiers à envoyer par exemple, ou la mémoire, dans :

```
/etc/php/8.2/apache2/php.ini
post_max_size
upload_max_filesize
memory_limit
max_execution_time
max_input_time
MaxClients 50
KeepAlive On
MaxKeepAliveRequests 50
KeepAliveTimeout 10
```

Pour la sécurité, il faut aussi modifier cela dans le *php.ini* :

```
expose_php = off
display_errors = off
```

Installation de MySQL

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

Pour sécuriser le serveur de base de données, on lance ce script et on répond à quelques questions :

```
sudo mysql_secure_installation
```

On redémarre Apache :

```
sudo systemctl restart apache2
```

On se connecte sur MySQL, on crée un utilisateur auquel on donne tous les droits et on crée les bases de données :

```
sudo mysql -u root -p
```

```
CREATE DATABASE bdd;
```

```
CREATE USER 'patrice'@'localhost' IDENTIFIED BY 'adminadmin';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'patrice'@'localhost' WITH GRANT OPTION;
```

```
FLUSH PRIVILEGES;
```

Mise en place des sites

On va créer le répertoire pour héberger le site sur le serveur :

```
cd /var/www
mkdir mon-site
```

On modifie les droits et propriétaires pour le répertoire du site :

```
sudo chown -R user:www-data /var/www/html/
sudo chmod -R 770 /var/www/html
```

On peut placer les fichiers du site dans ce répertoire et on crée ensuite les virtualhost dans `/etc/apache2/sites-available`

On va faire une copie du fichier de configuration par défaut (bonne pratique, on garde le fichier par défaut, en cas de soucis) :

```
sudo cp 000-default.conf mon-site.conf
```

Puis on l'édite avec Nano :

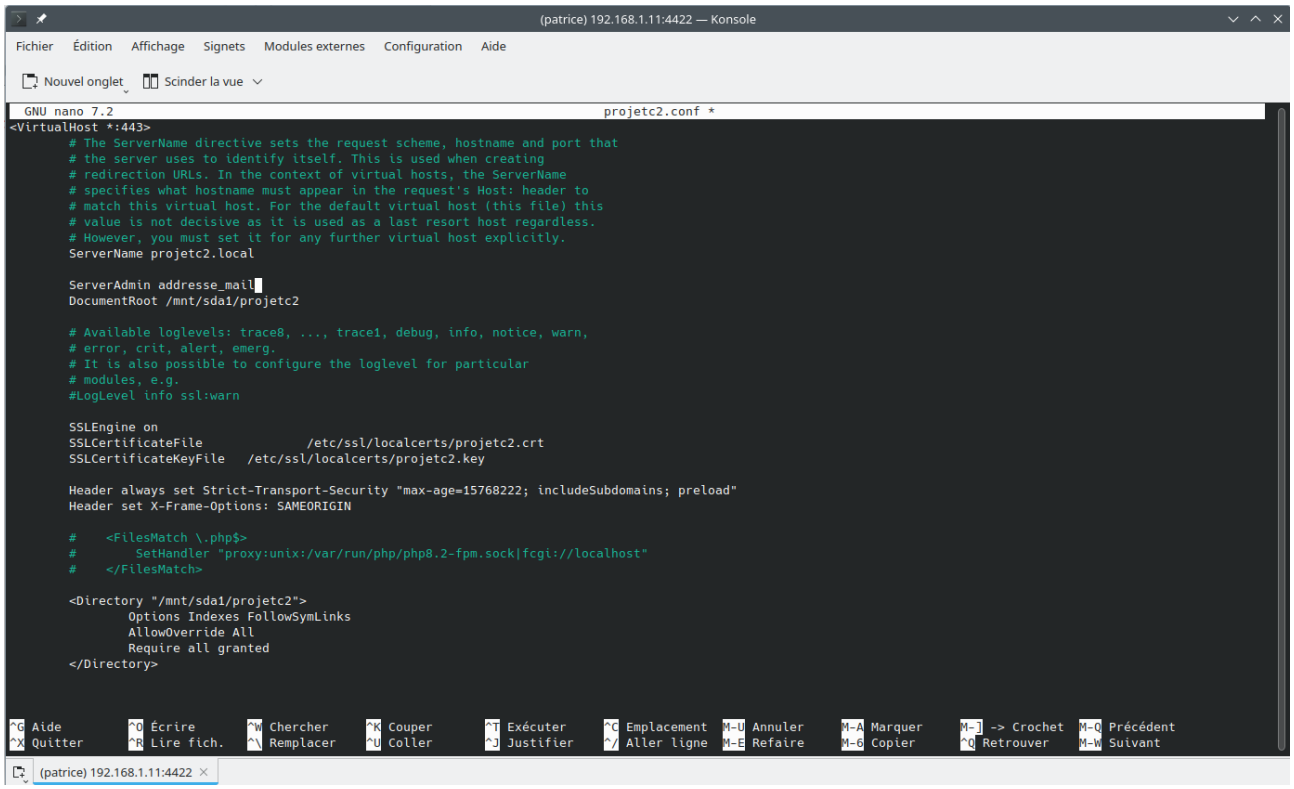
```
sudo nano mon-site.conf
<VirtualHost 192.168.1.11:80>
     ServerName mon-site.local
     ServerAdmin andreani.patrice@net-c.fr
     DocumentRoot /var/www/mon-site
<Directory /var/www/mon-site/>
        Options -Indexes +FollowSymlinks
        AllowOverride All
        Require all granted
     </Directory>
     ErrorLog ${APACHE_LOG_DIR}/mon-site-error.log
     CustomLog ${APACHE_LOG_DIR}/mon-site-access.log combined
</VirtualHost>
```

On peut ajouter quelques directives afin d'améliorer la sécurité :

```
Header always append X-Frame-Options SAMEORIGIN

Header set X-XSS-Protection "1; mode=block"

Header unset X-Powered-By

Header always set X-Content-Type-Options nosniff

Header always set Strict-Transport-Security "max-age=15552000; includeSubDomains"

FileETag None

TraceEnable off

Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

On active le virtualhost qu'on vient de mettre en place, on désactive celui par défaut, puis on redémarre Apache

```
sudo a2dissite 000-default

sudo a2ensite mon-site

sudo systemctl reload apache2
```



Une fois le site en place, ne pas oublier de modifier le fichier hosts des postes clients pour ajouter IP/domaine pour pouvoir accéder au site.

Aller plus loin

Mise en place de msmtp pour les mails

Installer les paquets msmtp et msmtp-mta ainsi que s-nail.

Fichier de configuration `/etc/msmtprc` :

```
# Valeurs par défaut pour tous les comptes.
defaults
auth           on
tls            on
tls_starttls   off
tls_trust_file /etc/ssl/certs/ca-certificates.crt
logfile        /var/log/msmtp
# Exemple pour un compte
account        patrice
auth           on
host           smtp_hébergeur_mail
port           465
from           mail
#auto_from on
#add_missing_from_header on
user           user
password       mdp
# Définir le compte par défaut
account default : patrice
aliases /etc/aliases
```

Mettre les droits sur le fichier de log :

```
sudo touch /var/log/msmtp
```

```
sudo chown msmtp:msmtp /var/log/msmtp
sudo chmod 660 /var/log/msmtp
```

Ensuite, dans le fichier `/etc/aliases` ajoutez une adresse mail comme ceci :

```
postmaster : mail
default : mail
root : mail
```

Dans `/etc/mail.rc`, ajouter ces lignes :

```
set mta=/usr/bin/msmtp
set sendmail="usr/bin/msmtp
```

Vérifier que le système utilise bien msmtp :

```
ls -la /usr/sbin/sendmail
```

Pour tester l'envoie de mail :

```
echo -e "Subject: Un mail de test\r\n\r\nEnvoyé via msmtp" | msmtp -t moi@example.com
```

Vérifier les logs

Il faut régulièrement vérifier les logs dans `/var/log` et notamment `auth` pour les tentatives de connexion. Les logs pour apache et vos virtualhosts se trouvent dans (suivant la configuration dans le fichier du virtual host) :

```
/var/log/apache/nom_virtualhost
```

Vous pouvez installer Logwatch afin de surveiller vos logs et recevoir un mail d'information.

Copier ce fichier :

```
sudo cp /usr/share/logwatch/default.conf/logwatch.conf /etc/logwatch/conf/
```

Regarder les valeurs de MailTo ainsi que Detail et ajoutez une tâche cron.

Création d’un certificat avec Let’s Encrypt

Pour que le site soit accessible en HTTPS, il faut un certificat. On peut utiliser Let’s Encrypt qui propose des certificats gratuits. Il y a un outil simple à utiliser pour mettre cela en place, Certbot.

Commençons par l’installer :

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

Vérifiez que vous avez bien renseigné la directive ServerName dans la configuration d’Apache car Certbot se servira de cela, même si vous pouvez aussi nommer le domaine manuellement. Lançons la commande :

```
sudo certbot –apache
```

Voilà, une fois répondu à quelques questions, certbot vous affichera le résultat et votre configuration, vous aurez votre certificat. Redémarrez Apache et rechargez votre page afin de vérifier qu’elle est bien en HTTPS et que les informations du certificat sont correctes dans le navigateur.

Création d’un certificat auto-signé avec OpenSSL

Le certificat va permettre d’accéder au site via https, ce qui est mieux pour la sécurité, même si ici tout reste en local.

On installe openssl :

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install openssl
```

On va créer un dossier spécifique pour nos clés, dans `/etc/ssl` :

```
cd /etc/ssl
sudo mkdir localcerts
cd localcerts
```

On crée la clé privée

```
sudo openssl genrsa -out mon-site.key 2048
```

On crée le fichier de demande de signature de certificat :

```
sudo openssl req -new -key mon-site.key -out mon-site.csr
```

On génère enfin le certificat signé :

```
sudo openssl x509 -req -days 365 -in mon-site.csr -signkey mon-site.key -out mon-site.crt
```

On édite le fichier de configuration du virtualhost pour ajouter les informations du certificat :

```
SSLEngine on
SSLCertificateFile      /etc/ssl/localcerts/mon-site.crt
SSLCertificateKeyFile   /etc/ssl/localcerts/mon-site.key
```

On peut aussi activer HTTP2 :

```
sudo a2enmod http2
```

Et ajouter ceci dans le fichier de configuration du virtualhost :

```
<IfModule mod_http2.c>
     Protocols h2 http/1.1
</IfModule>
```

On peut aussi activer HSTS, en ajoutant cette ligne dans le fichier de configuration du virtualhost :

```
Header always set Strict-Transport-Security "max-age=63072000;
includeSubdomains; preload"
```

On redémarre le serveur Apache :

```
sudo systemctl restart apache2
```

Le debug

On a installé au début, dans les paquets de php, le paquet php-xdebug. Pour pouvoir l'utiliser, il faut le configurer mais également installer le plugin dans Visual Studio Code et dans Firefox :

https://addons.mozilla.org/fr/firefox/addon/xdebug-helper-for-firefox/

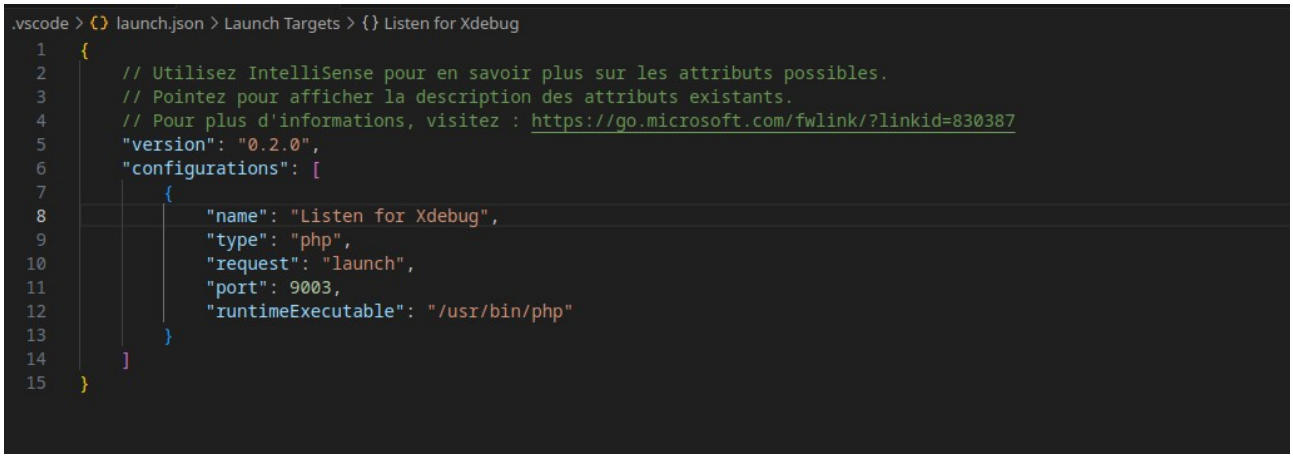
https://marketplace.visualstudio.com/items?itemName=xdebug.php-debug

Voici quelques liens pour la configuration :

https://www.cloudways.com/blog/php-debug/

https://www.tutorials24x7.com/php/how-to-debug-php-using-xdebug-and-visual-studio-code-on-ubuntu

Voici sous Visual Studio Code les fichiers de configurations pour xdebug :



Visual Studio Code

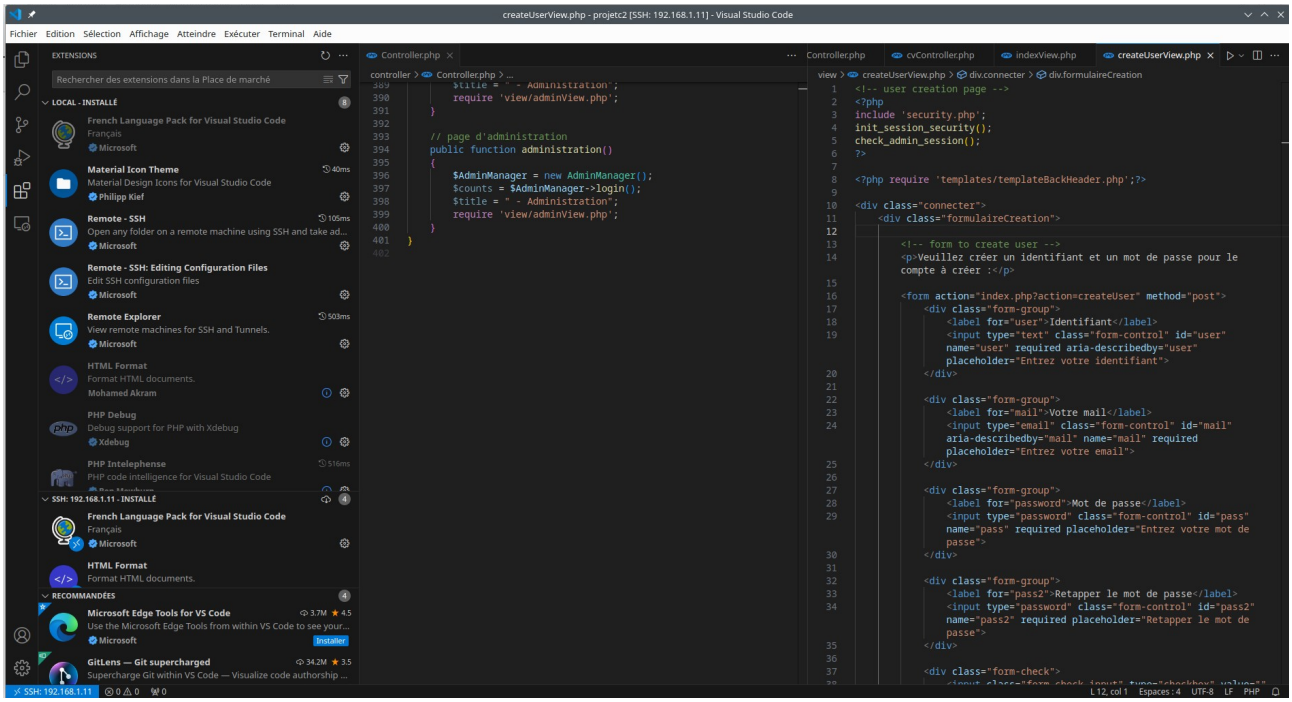
Visual Studio Code est un éditeur de code très complet. On peut le personnaliser, au niveau de l'apparence mais aussi au niveau des fonctionnalités, grâce à de nombreux plugins. Il gère git par défaut et dispose d'une console.

Quelques plugins utiles :

Remote – SSH : permet de se connecter sur un hôte distant pour travailler directement sur le serveur depuis votre ordinateur.

PHP Intelephense : donne de nombreuses informations sur le PHP, les classes, etc...

PHP Debug : pour utiliser le mode debug, très pratique. Peut faire du pas à pas, surveiller, les valeurs des variables, etc...



Mettre en place Git

Une fois votre compte créé sur Gitlab et une clé SSH mise en place dessus (pour la connexion à distance), il faut configurer git sur votre système, après l'avoir installé avec :

```
sudo apt install git
```

Configuration globale de Git sur votre système :

```
git config --global user.name "nom_du_compte"
git config --global user.email "adresse_email"
```

Pour mettre en place git sur un projet existant, rendez-vous dans le dossier de celui-ci puis suivez ces commandes. Cela initie un projet git dans le répertoire courant, le configure pour le projet créé sur Gitlab et envoie vos fichiers sur Gitlab.

```
git init --initial-branch=main
git remote add origin git@gitlab.com:adresse_de_votre_projet_git
git add .
git commit -m "Initial commit"
git push --set-upstream origin main
```

Une fois votre projet mis en place sur git, il y a quelques commandes à connaître (dans le répertoire de votre projet bien sûr) :

`git status` → pour savoir s'il y a des modifications à synchroniser.

`git clone` → pour récupérer et synchroniser un projet.

`git pull origin main` → pour récupérer les dernières modifications d'un projet depuis sa branch main.

`git push origin main` → pour envoyer vos dernières modifications sur la branche main.

`git reset --hard` → annule vos dernières modifications.

`git branch` → connaître la branche actuelle.

`git checkout -b` créer une nouvelle branche.

`git checkout nom_branch` → permet de changer de branche.

Utiliser plusieurs versions de PHP

Il est possible d'installer plusieurs versions de PHP sur votre système. Ainsi, vous pouvez tester vos applications sur diverses versions, c'est très utile.

Il va falloir mettre en place un nouveau dépôt, qui permettra ensuite d'installer les différentes versions de php que vous souhaitez. On utilisera ensuite php-fpm.

Dans le fichier de configuration de vos virtual host, il faudra ajouter ces lignes, qui permettent d'indiquer quelle evrsion de PHP utiliser pour celui-ci :

```
<FilesMatch \.php$>
    SetHandler
"proxy:unix:/var/run/php/php8.2-fpm.sock|fcgi://localhost"
</FilesMatch>
```

Voici quelques liens utiles pour réaliser tout cela :

https://techvblogs.com/blog/install-multiple-php-versions-on-ubuntu-22-04

https://www.phoca.cz/blog/1302-running-multiple-php-instances-with-lamp-stack-on-ubuntu-kubuntu-linux

https://www.linuxbabe.com/ubuntu/php-multiple-versions-ubuntu

Bibliographie

LAMP : https://raspberry-pi.fr/installer-serveur-web-raspberry-lamp/

LAMP : https://simple-duino.com/installer-serveur-web-raspberry/

Certificat SSL auto-signé : https://www.linuxtricks.fr/wiki/certificat-auto-signe-creation-et-parametrage-dans-apache2-ou-nginx

ufw : https://doc.ubuntu-fr.org/ufw

Les services : https://linuxhint.com/disable-a-service-in-ubuntu/

SSH : https://www.it-connect.fr/chapitres/bonnes-pratiques-de-configuration-ssh/

SSH : https://www.it-connect.fr/chapitres/openssh-configuration-du-serveur-ssh/

SSH limit : https://www.cyberciti.biz/faq/howto-limiting-ssh-connections-with-ufw-on-ubuntu-debian/

User pi : https://programmation.surleweb-france.fr/raspberry-supprimer-le-compte-pi/

Sécurité : https://websetnet.net/fr/how-to-secure-your-lamp-server/

Sécurité : https://www.linuxtricks.fr/wiki/apache-installation-configuration-securisation

Sécurité : https://www.rosehosting.com/blog/how-to-secure-your-lamp-server/

Hardening Linux : https://www.cyberciti.biz/tips/linux-security.html

Let’s Encrypt : https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-20-04-fr

Ce document est publié sous les termes de la licence Attribution-NonCommercial-ShareAlike 4.0 International. Pour voir une copie de cette licence, rendez-vous sur :

http://creativecommons.org/licenses/by-nc-sa/4.0/