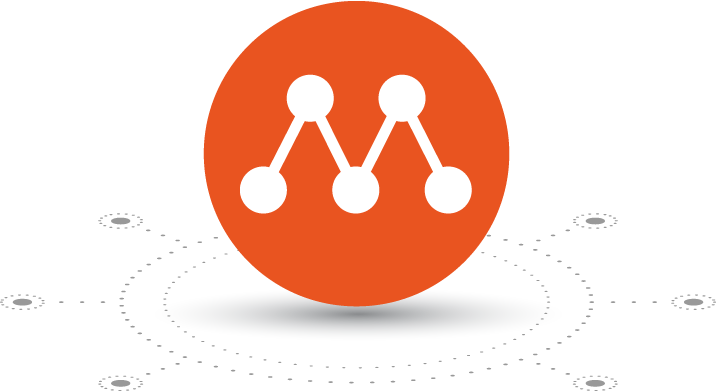


**CCI - SIO Atelier 2 - SACCHETTO Vladimir le 21/05/2022**

Configuration de l’environnement de travail

Rédaction de documentation utilisateur



SOMMAIRE

1. Introduction. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. Environnement des Instances. . . . . . . . . . . . . . . . . . . . . . . . . . 3
3. Télécharger Multipass. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
4. Installation des instances. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
5. Configuration des instances. . . . . . . . . . . . . . . . . . . . . . . . . . . 5
   1. Mise à jour des librairies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2. Installation et Configuration de Apache 2 – PHP – Maria DB. . . . . . 6
   3. Installation et Configuration de Xdebug - PhpMyAdmin. . . . . . . . . . 8
   4. Création de l’user pour PhpMyAdmin. . . . . . . . . . . . . . . . . . . . . . . . . . 9
   5. Installation composer. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
   6. Monter le projet git et configuration du Virtual host . . . . . . . . . . . 10
   7. Afficher la page web et le nom de domaine. . . . . . . . . . . . . . . . . . . . . 11
6. Scripts pour passage de dev a test. . . . . . . . . . . . . . . . . . . . . . . 13
7. Scripts pour passage de test a prod. . . . . . . . . . . . . . . . . . . . . . 15
8. Introduction

A l’aide de ce document nous allons démontrer comment un jeune développeur Web peut mettre en place une stratégie pour suivre ses projets et tester son code avant de le rendre effectif.

Pour ce faire nous allons devoir créer des instances *(machines virtuelles)* qui nous donneront la possibilité de suivre nos projets.

Les instances nécessaires sont 3 :

* Machine de **DEVELOPPEMENT**
* Machine de **TEST**
* Machine de **PRODUCTION**

Afin de créer ces instances nous avons besoin de télécharger l’outil de création d’instances Ubuntu **« Multipass v 1.8.0 ».**

Multipass est l’outil qui permet de créer des instances Ubuntu.  Il est conçu pour les développeurs qui souhaitent un nouvel environnement Ubuntu avec une seule commande et fonctionne sous Linux, Windows et macOS.

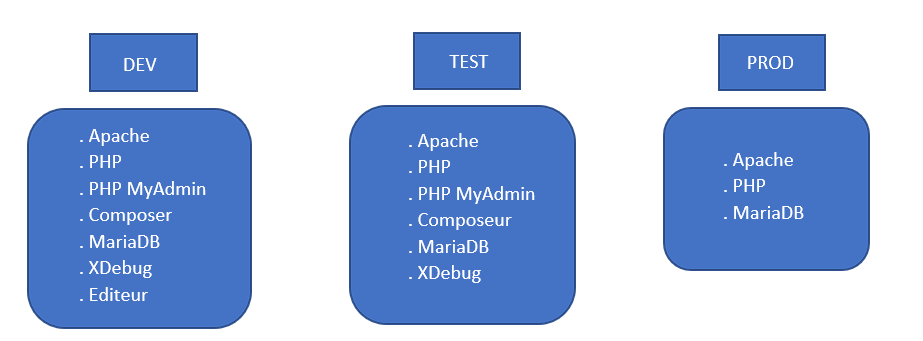
Multipass est une méthode recommandée pour créer des machines virtuelles.

[Créer une instance](https://multipass.run/docs/working-with-instances) [Informations sur Multipass](https://ubuntu.com/server/docs/virtualization-multipass#:~:text=Multipass%20is%20the%20recommended%20method,multipass%20%2D%2Dbeta%20%2D%2Dclassic)

1. Environnement des Instances

Nos machines virtuelles devront respectivement présenter des outils adaptés à l’environnement d’un développeur. Ainsi nous devons nous assurer que chaque instance contienne les outils de travail nécessaires et les installer soit à l’aide d’un script **bash** ou avec les commandes saisies l’une après l’autre.

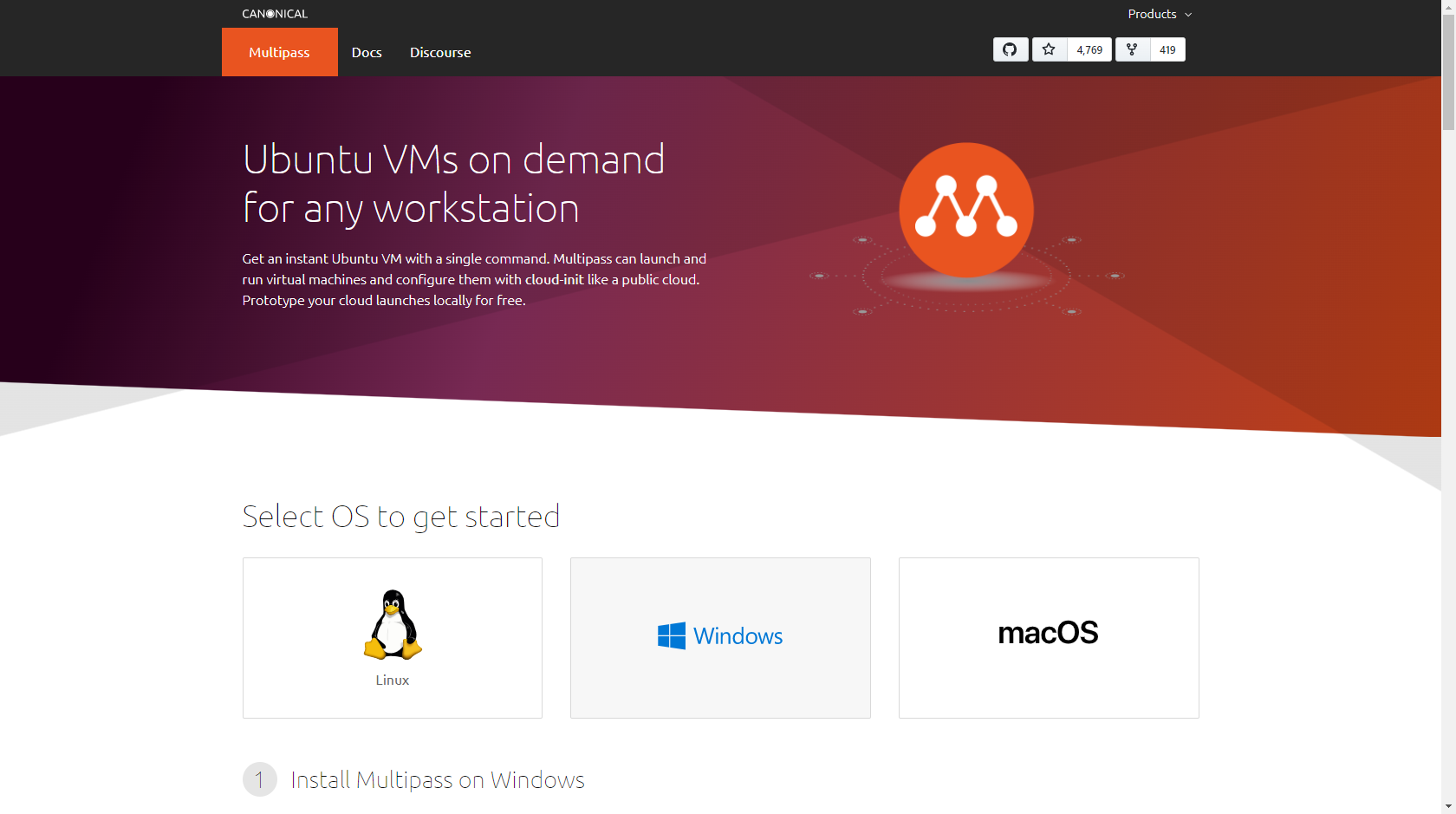
L’environnement pour chaque machine est le suivant :

**

Afin de bien développer et de mettre à jour notre projet au fur et à mesure du temps, nous devons faire communiquer les 3 machines entre elles. Nous devons ainsi utiliser deux scripts que nous allons nommer **Maj.sh** et **Synch.sh**.

1. Télécharger Multipass

Avant tout nous avons besoin de [télécharger](https://multipass.run/) Multipass.



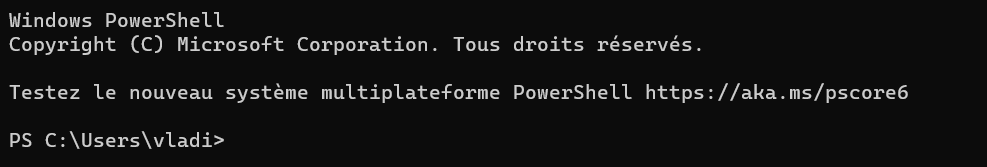
Ceci est la page officielle de Multipass Ubuntu. Ici vous pouvez choisir votre environnement et télécharger la version de Multipass lié à votre OS.

Après avoir téléchargé Multipass vous pouvez procéder à l’installation.

Ce document se base sur un système d’exploitation **Windows 10 Professional**.

1. Installation des instances

Nous allons alors ouvrir le terminal pour l’invite de commande (CMD) ou PowerShell de Windows.



Afin de créer notre première machine virtuelle nous rentrons la commande multipass launch --name foo.

foo est un nom standard de l’instance que nous allons remplacer par **développeur, test** ou **production** comme cela a été défini initialement dans notre document.

**





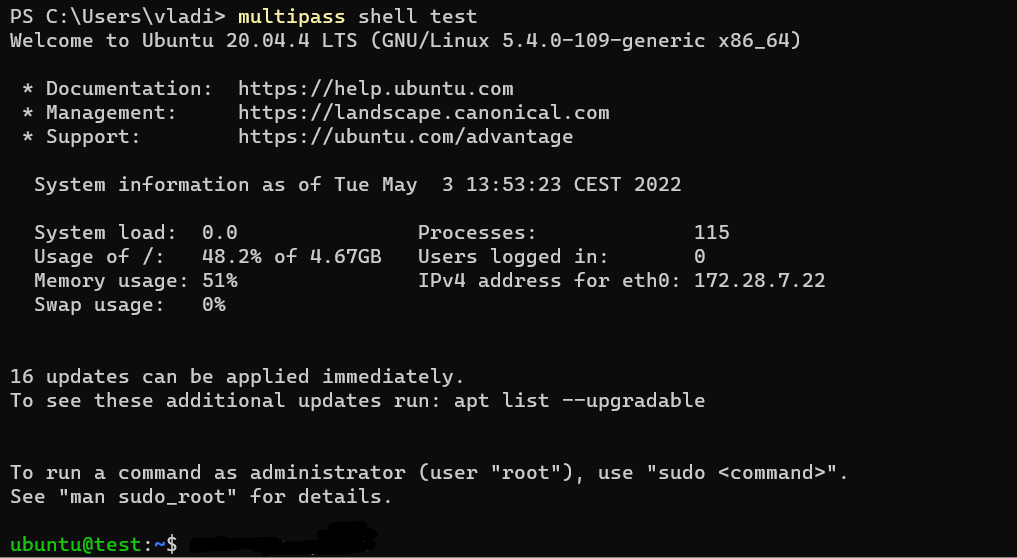
Une fois que l’installation des 3 machines virtuelles a été effectuée, nous pouvons voir l’état des instances avec la commande multipass list

**

Nous pouvons stopper et redémarrer nos machines à tout moment avec les commandes multipass stop *(nom instance)* et multipass start *(nom instance)*

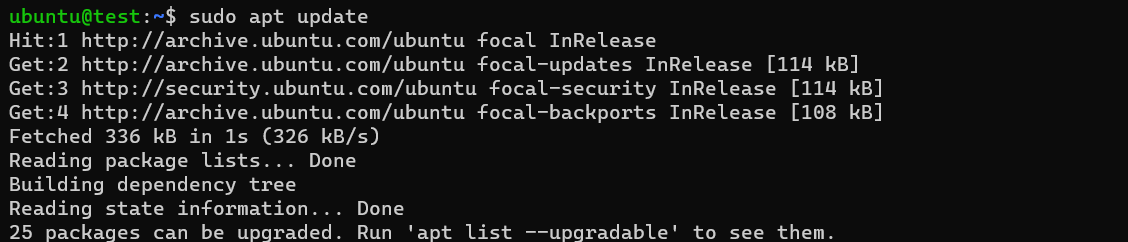
1. Configuration des instances

Afin de pouvoir installer notre environnement de travail dans une des instances, nous allons accéder à notre instance avec la commande multipass shell test

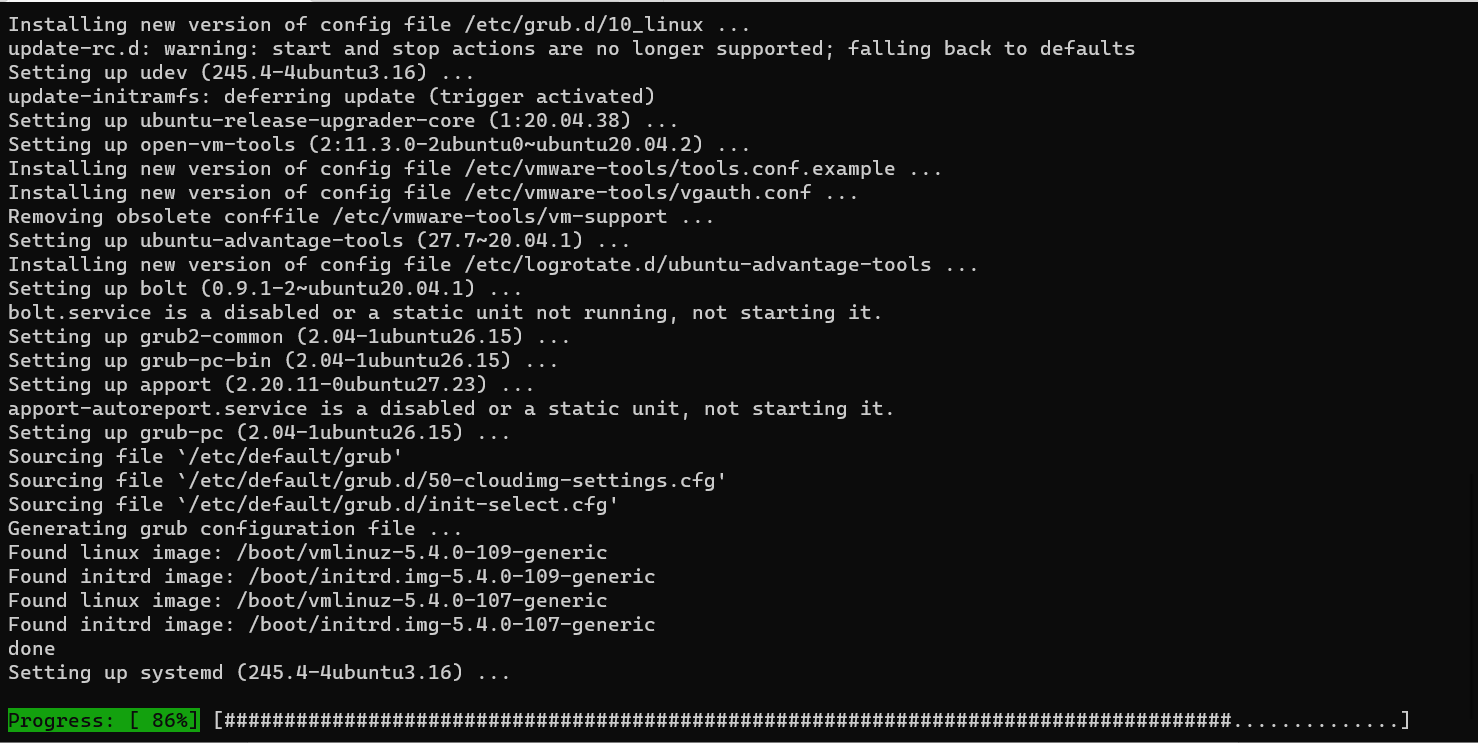
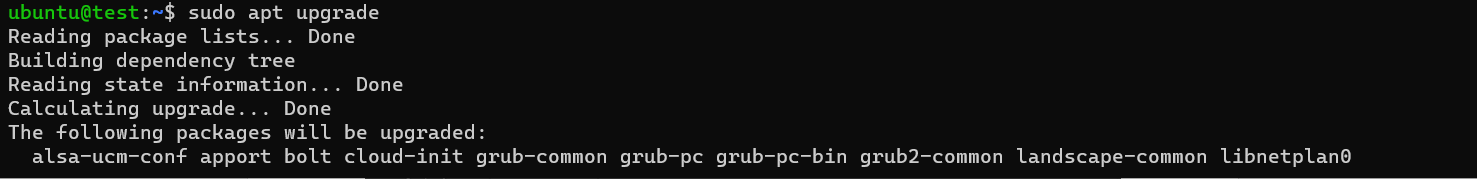


* 1. Mise à jour des librairies

Lorsque nous sommes dans notre machine virtuelle vide, nous allons mettre à jour les sources avec la commande sudo apt update

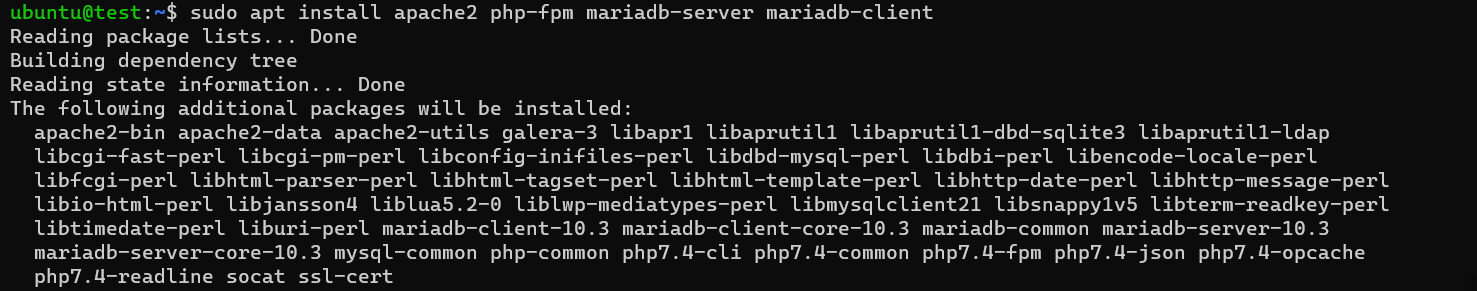


Ensuite nous allons les installer dans la VM avec la commande sudo apt upgrade

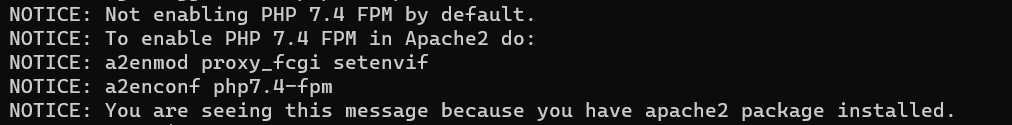


5.2 Installation et Configuration de Apache 2 – PHP – Maria DB

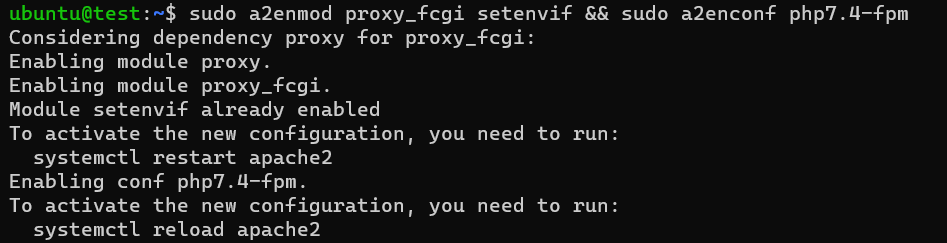
Après avoir effectué la mise à jour des sources, nous allons installer les environnements **Apache 2 – PHP – MariaDB** avec la commande sudo apt install apache2 php-fpm mariadb-server mariadb-client -y *(elle regroupe les trois installations d’un seul coup)*



Après l’installation des environnements un message s’affiche

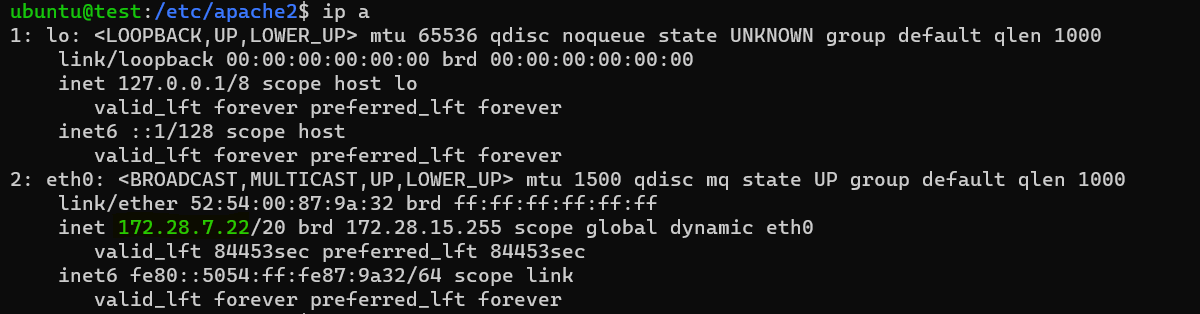


Ce message nous sert à configurer PHP dans le serveur Apache avec les commandes sudo a2enmod proxy\_fcgi setenvif && sudo a2enconf php7.4-fpm *(a2 = apache2, en = enable)*

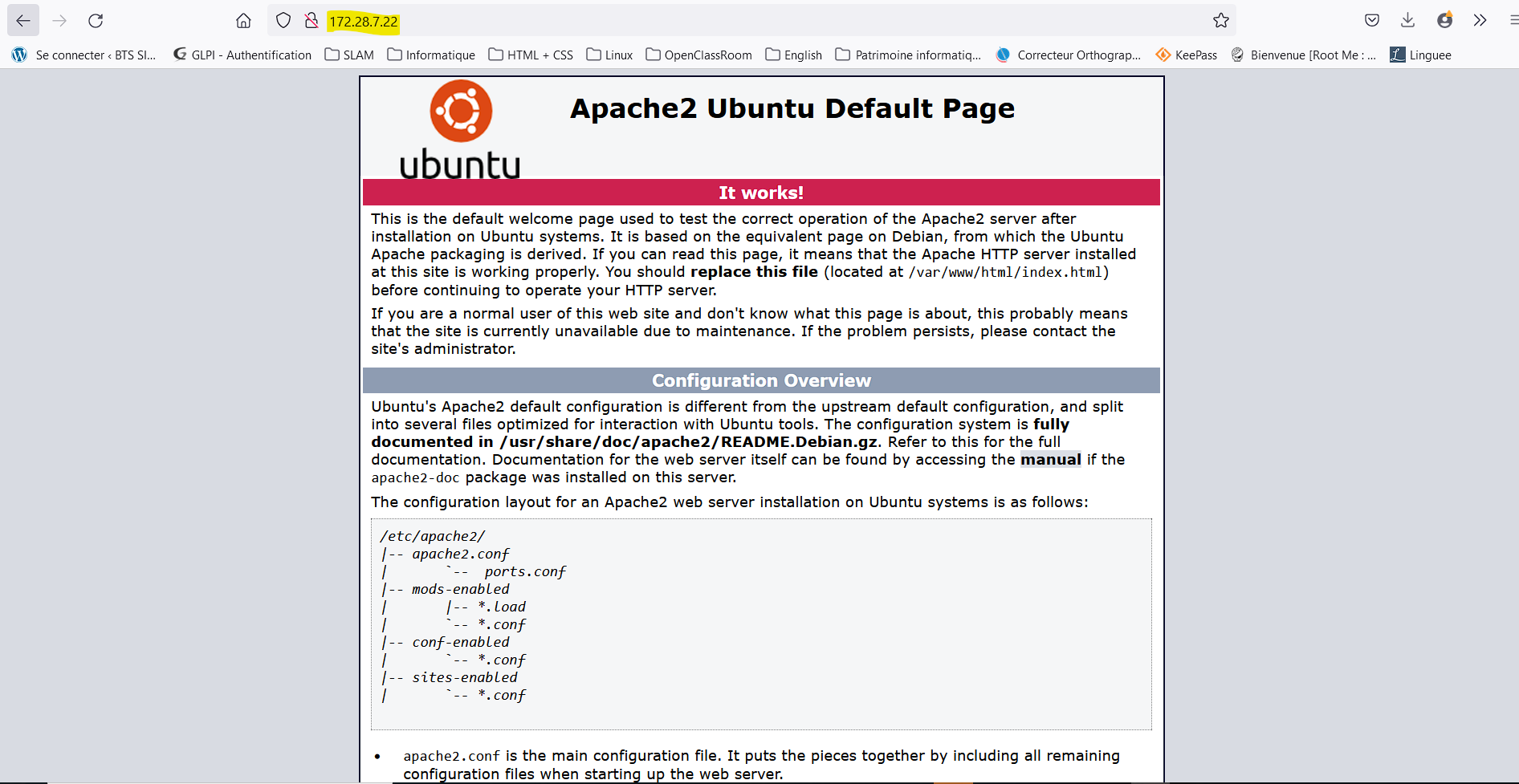


Nous allons ensuite relancer apache2 sudo systemctl restart apache2

Nous pouvons voir le statut d’Apache en allant dans l’adresse ip du serveur Apache. Nous allons donc retrouver l’adresse ip avec la commande ip a

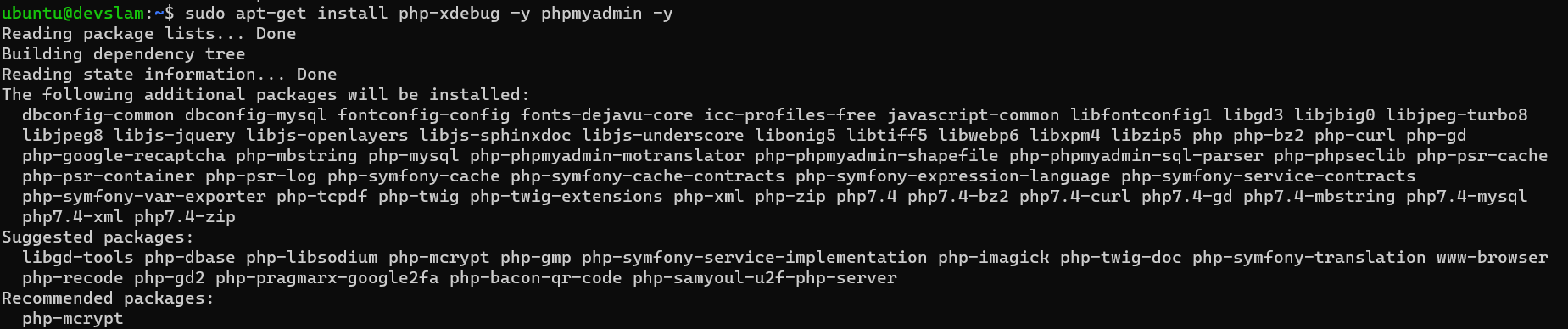


Puis nous prenons l’adresse ip et la mettant dans le navigateur



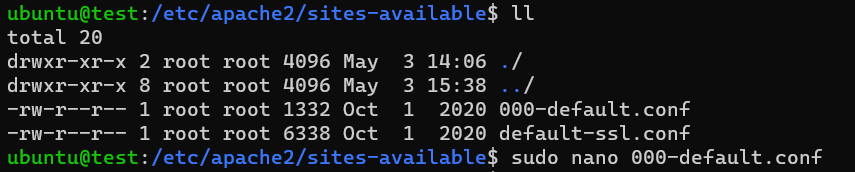
5.3 Installation et Configuration de Xdebug - PhpMyAdmin

Nous allons ensuite installer **Xdebug** et **PHPMyAdmin** avec la commande sudo apt-get install php-xdebug -y phpmyadmin -y

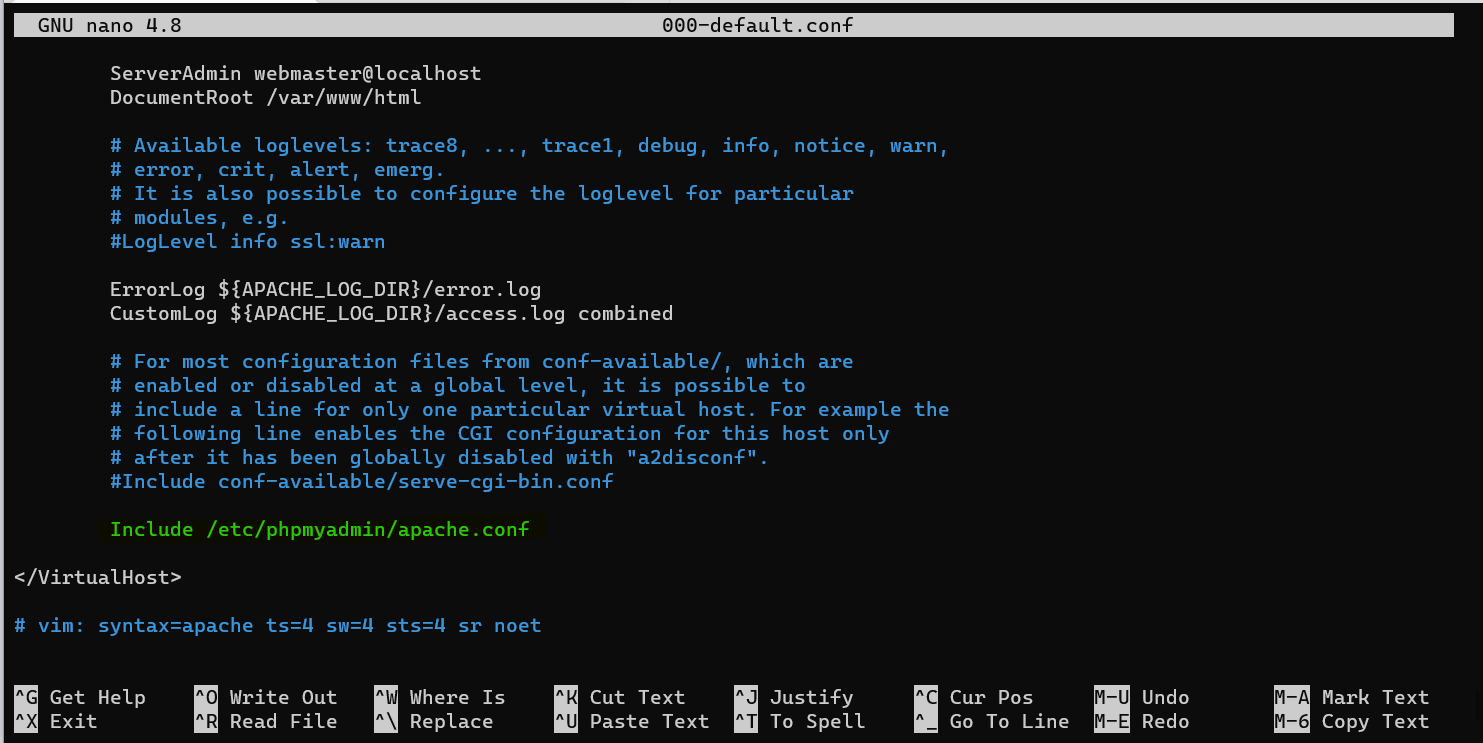


Après l’installation des deux outils nous allons modifier la configuration Apache et rajouter le fichier **apache.conf** contenu dans **phpmyadmin** dans le Virtual Host de Apache.

Pour ce faire nous allons nous rendre dans le dossier */etc/apache2/sites-available*. Ici nous allons modifier le fichier **000-default.conf** avec la commande sudo nano **000-default.conf.**

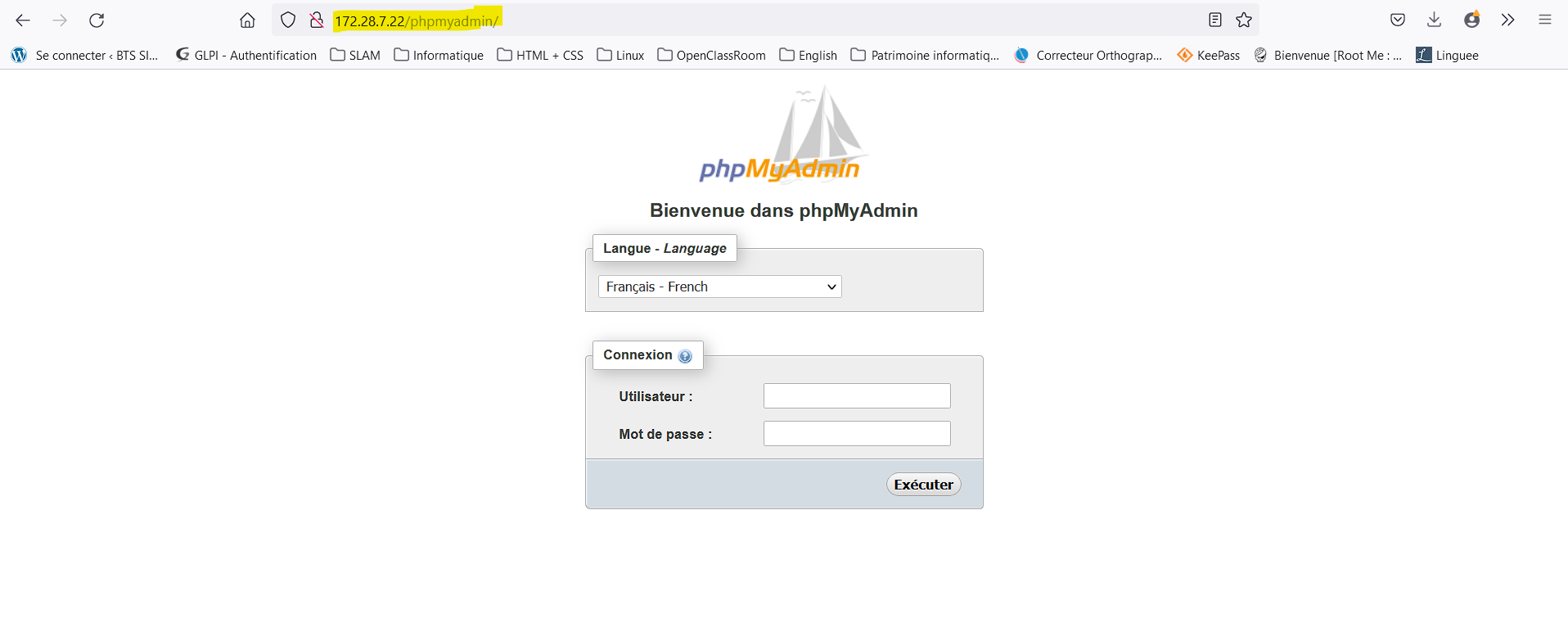


Ici nous allons indiquer le chemin pour mettre à jour le fichierde Virtual Host et allons écrire à l’avant dernière ligne : Include /etc/phpmyadmin/apache.conf



Nous allons ensuite relancer Apache2 afin que la modification soit prise en compte

Pour afficher la page PHPMyAdmin dans le navigateur nous allons prendre l’adresse ip de Apache2 et mettre un */phpmyadmin* : *http://172.28.7.22/phpmyadmin/* afin d’afficher la page



5.4 Création de l’user pour PhpMyAdmin

Afin de créer les Users nous allons modifier le SQL avec la commande sudo mysql

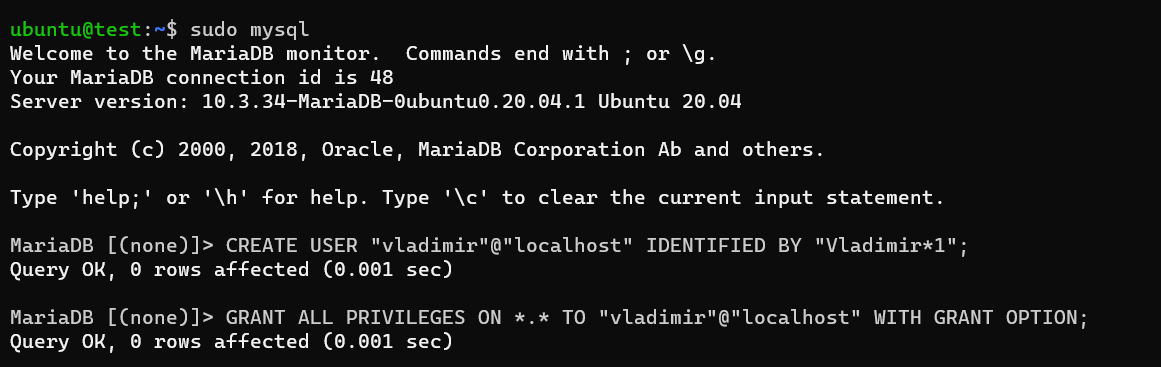
Dans le SQL nous allons créer l’User et lui attribuer un mot de passe ainsi que lui donner tous les droits.

CREATE USER “xxx”@”xxx” IDENTIFIED BY ”yyy”; *(creation User + MDP)*

GRANT ALL PRIVILEGES ON \*.\* TO "xxx"@"xxx" WITH GRANT OPTION; *(tous les droits à l’User créé)*

FLUSH PRIVILEGES;

Ensuite nous allons relancer mysql sudo systemctl restart mysql



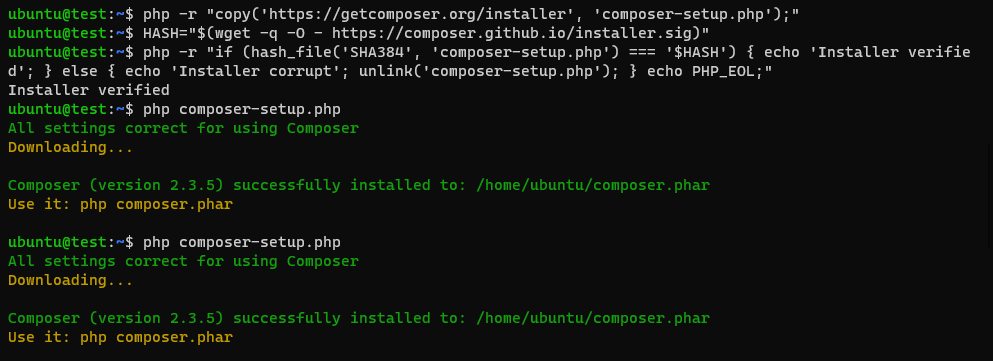
5.5 Installation composer

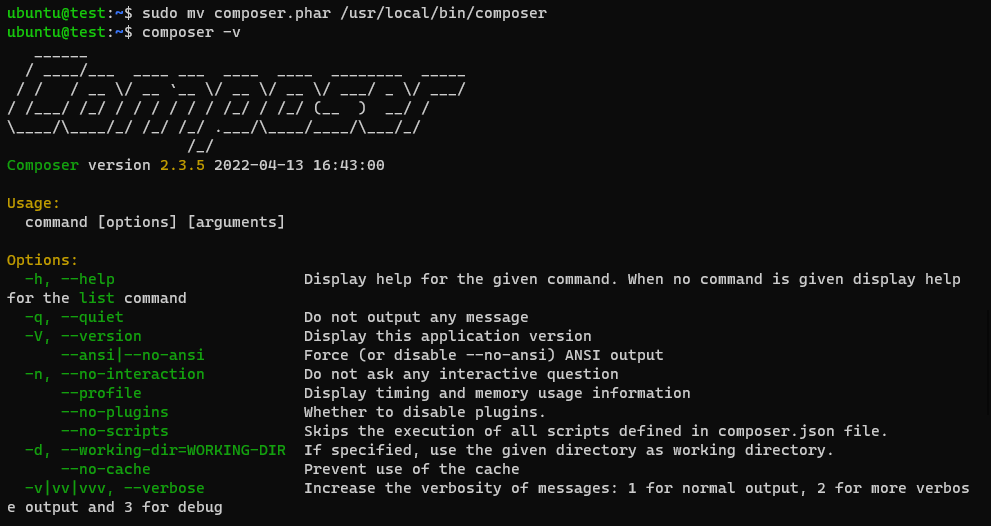
Nous allons installer composer avec les commandes :

* php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
* HASH="$(wget -q -O - https://composer.github.io/installer.sig)"
* php -r "if (hash\_file('SHA384', 'composer-setup.php') === '$HASH') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;"
* php composer-setup.php
* php -r "unlink('composer-setup.php');"

Ensuite nous le mettons dans le dossier */usr/local/bin/composer*

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





5.6 Monter le projet git et configuration du Virtual host

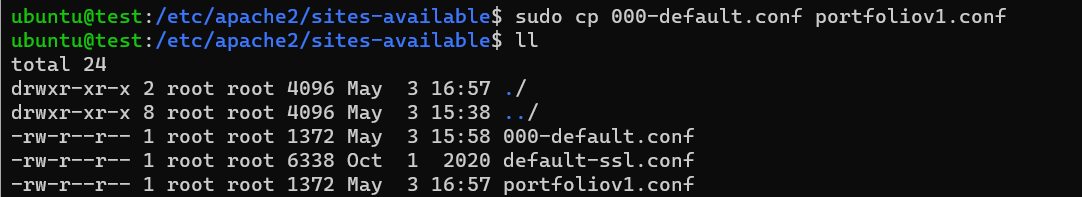
Nous allons lancer une commande dans Windows pour monter notre projet présent dans la machine physique vers la VM de **développement**. Nous allons lancer la commande multipass mount "D:\Cours de la CCI\Atelier 1\portfolio\CCI-SIO21-Portfolio" devslam:/var/www/CCI-SIO21-Portfolio.



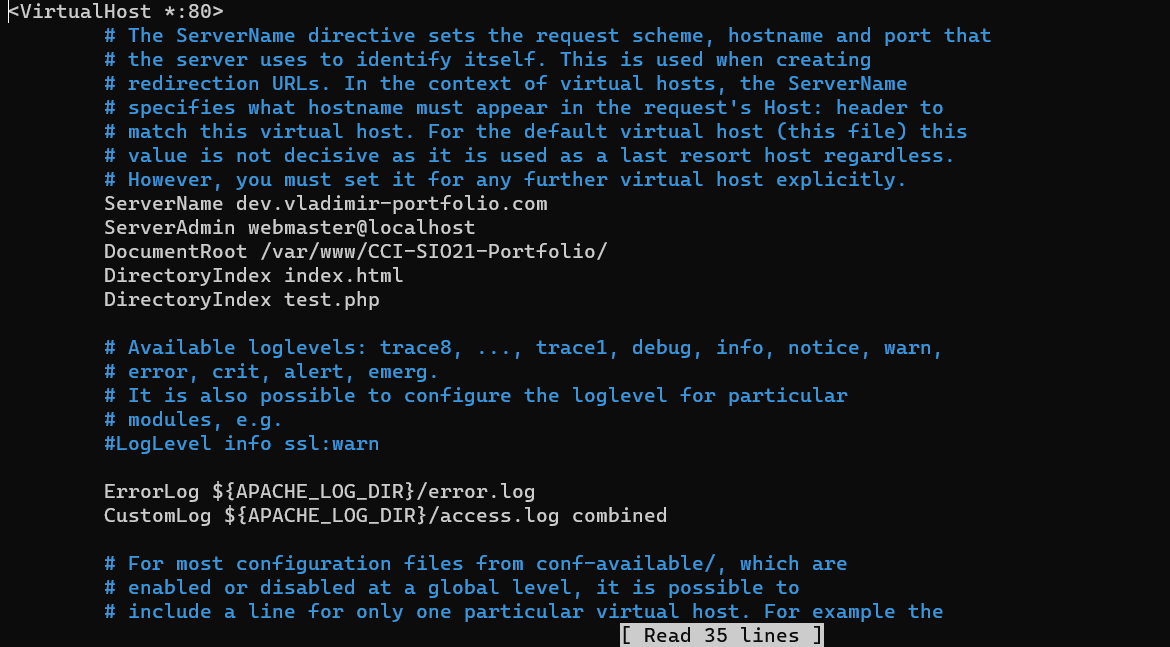
Ainsi nous pouvons apporter des modifications aux fichiers de notre projet directement sur notre machine Windows et seront automatiquement modifiés dans la VM.

Afin d’afficher notre site nous devons pointer le dossier monté dans le Virtual Host. Nous allons copier le fichier **000-default.conf** présent dans le dossier **/etc/apache2/sites-available** et le renommeren **portfoliov1.conf.**

Cela avec la commandesudo cp 000-default.conf portfoliov1.conf



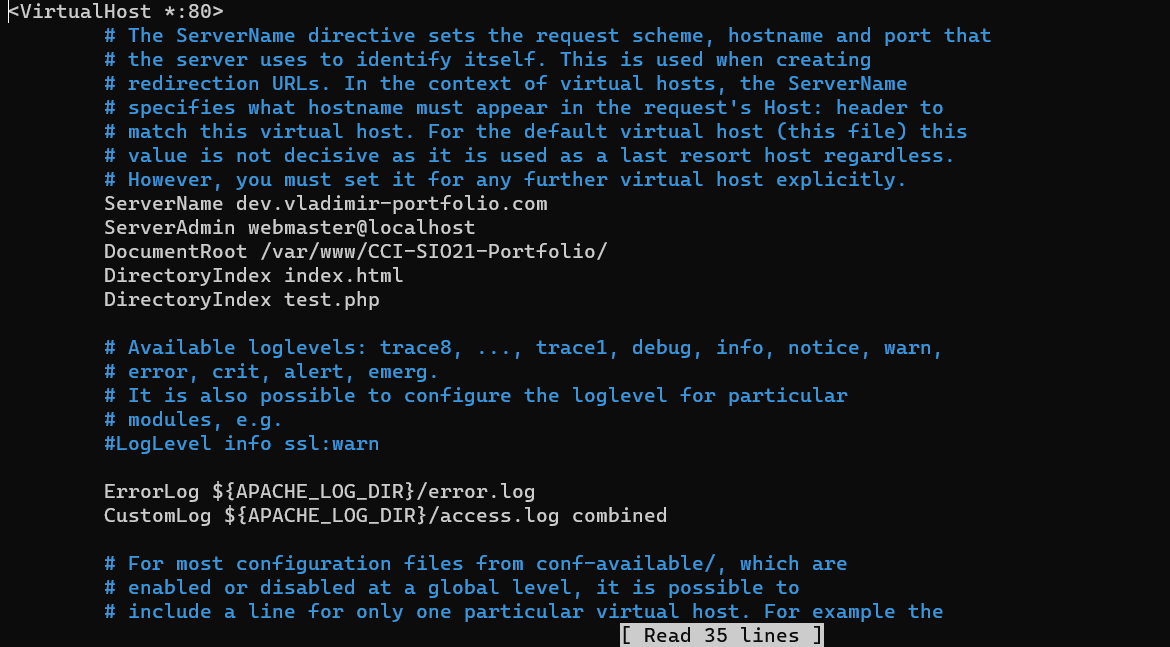
Dans le nouveau fichier de configuration du Virtual Host nous allons indiquer depuis quel répertoire **apache2** doit afficher la page web ***(DocumentRoot /var/www/CCI-SIO21-Portfolio/)*** et quel fichier prendre en compte ***(DirectoryIndex index.html* et/ou *DirectoryIndex test.php)***



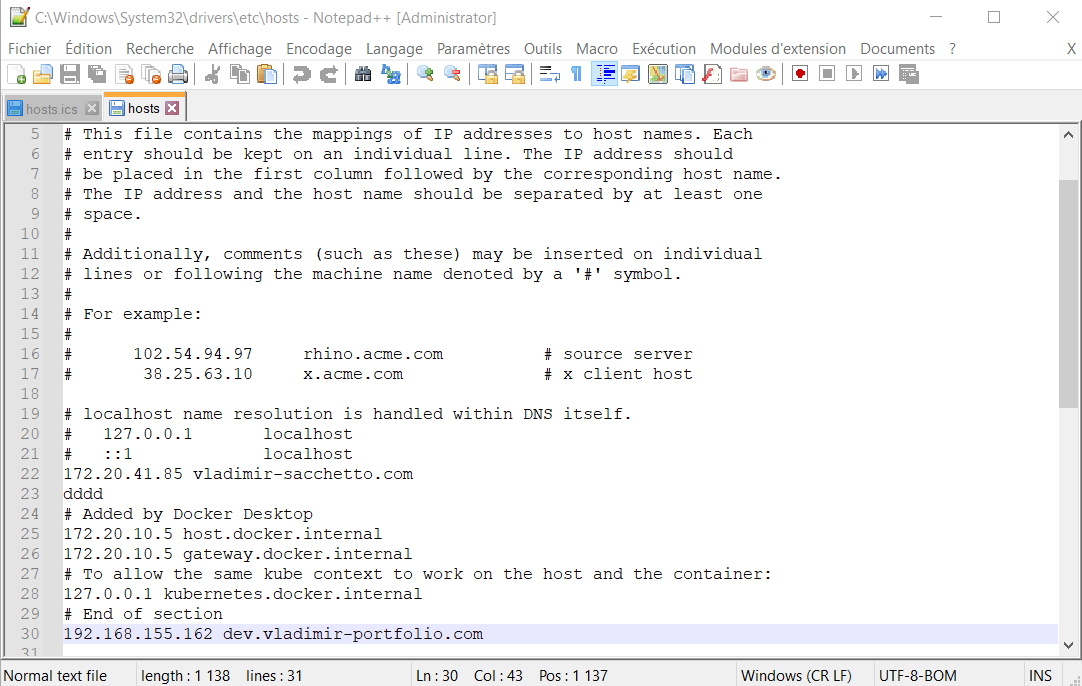
Ensuite nous allons activer le fichier créé avec la commande sudo a2ensite portfoliov1.conf et désactiver le fichier standard **000-default.conf** avec la commandesudo a2dissite 000-default.conf

5.7 Afficher la page web et le nom de domaine

Nous allons modifier le fichier crée avec la commande sudo nano portfoliov1.conf. Dans le fichier de config nous allons indiquer le nom de domaine de notre index.html. Cela avec le **ServerName** *(dev.vladimir-portfolio.com)*



Ensuite nous allons rajouter l’adresse ip de notre VM suivie du nom de domaine dans le fichier **hosts** qui se trouve dans *C:\Windows\System32\drivers\etc*





1. Scripts pour passage de dev a test

Nous allons maintenant voir comment faire communiquer les instances afin de mettre à jour nos modifications au fur et à mesure du temps.

Avant tout nous allons faire un clone de notre projet git à l’aide du code HTTPS dans le dossier */var/www* de la machine de **test** avec la commande git clone https://github.com/Vladimir9595/CCI-SIO21-Portfolio.git /var/www/CCI-SIO21-Portfolio.

Ensuite la communication entre les deux VM se fera à l’aide d’un script que nous allons créer à la racine de la VM de **test**.

Ici nous créons le fichier **maj.sh** avec la commande touch maj.sh.



Ce fichier va contenir des commandes qui tout d’abord vont donner les droits à l’user **ubuntu** de rechercher les dernières modifications du projet sur Github avec un git pull :

* sudo chown -R ubuntu:ubuntu /var/www/CCI-SIO21-Portfolio
* git pull

Ensuite donne les droits à **apache** de sauvegarder le contenu et l’afficher :

* sudo chown -R www-data:www-data /var/www/CCI-SIO21-Portfolio

Puis les droits sur le dossier *.git* à l’user **ubuntu** :

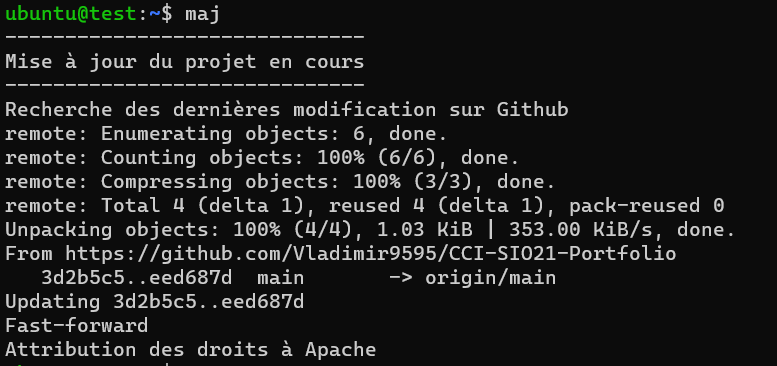
* sudo chown -R ubuntu:ubuntu /var/www/CCI-SIO21-Portfolio/.git

Nous allons insérer ces commandes dans le script ***maj.sh*** avec l’éditeur nano sudo nano maj.sh



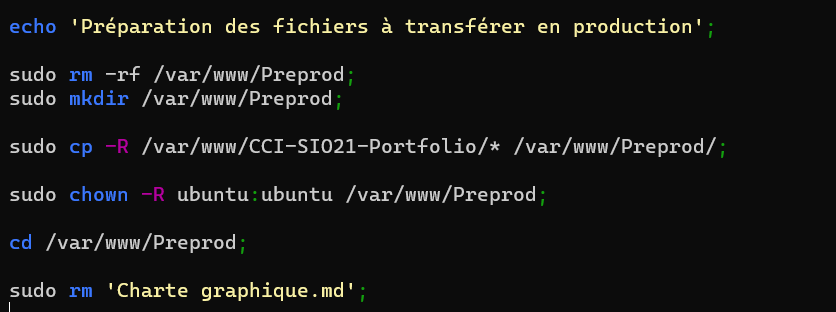


Nous pouvons lancer à tout moment notre script en saisissant le nom du fichier **maj**

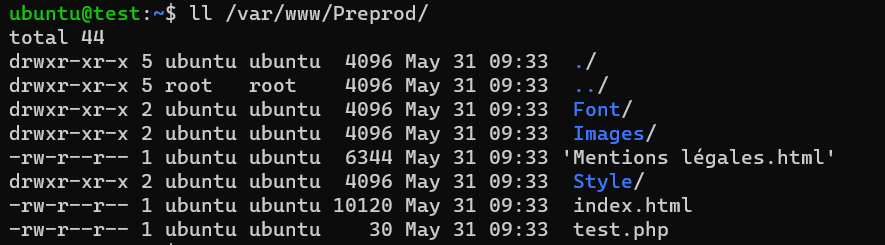


Avant de terminer nous devons pouvoir envoyer certains éléments vers la production en excluant d’autres. Pour ce faire, dans la machine de **test**, nous pouvons copier tout ce qui est présent dans le dossier cloné *(/var/www/CCI-SIO21-Portfolio)* vers un dossier que nous allons nommer **Preprod** *(/var/www/Preprod).* Puis nous allons supprimer ce que nous ne voulons pas envoyer en production.

Nous allons insérer les commandes dans notre script *maj.sh*



Puis nous allons vérifier la présence des éléments dans le dossier *Preprod*



1. Scripts pour passage de test a prod

Nous allons ensuite créer un script **synch.sh** qui va pouvoir synchroniser les éléments présents dans le dossier de **Preprod** de la machine de **test** *(/var/www/Preprod)* vers un dossier qui est créé dans la machine de **production** */var/www/CCI-SIO21-Portfolio*.



Cette action s’effectue avec la commande **rsync.** Cette commande permet de transférer des fichiers d’une machine à l’autre ainsi faire une copie et mettre à jour les dernières modifications.

Ce script est lancé dans la machine de **test.** Nous allons donner les droits à l’utilisateur sur le dossier qui contient le clone de notre projet git :

* sudo chown -R ubuntu:ubuntu /var/www/CCI-SIO21-Portfolio;

Ensuite nous allons nous connecter en SSH à notre serveur de production et allons créer un répertoire dans le */var/www/* puis donnons les droits à l’utilisateur sur le dossier créé :

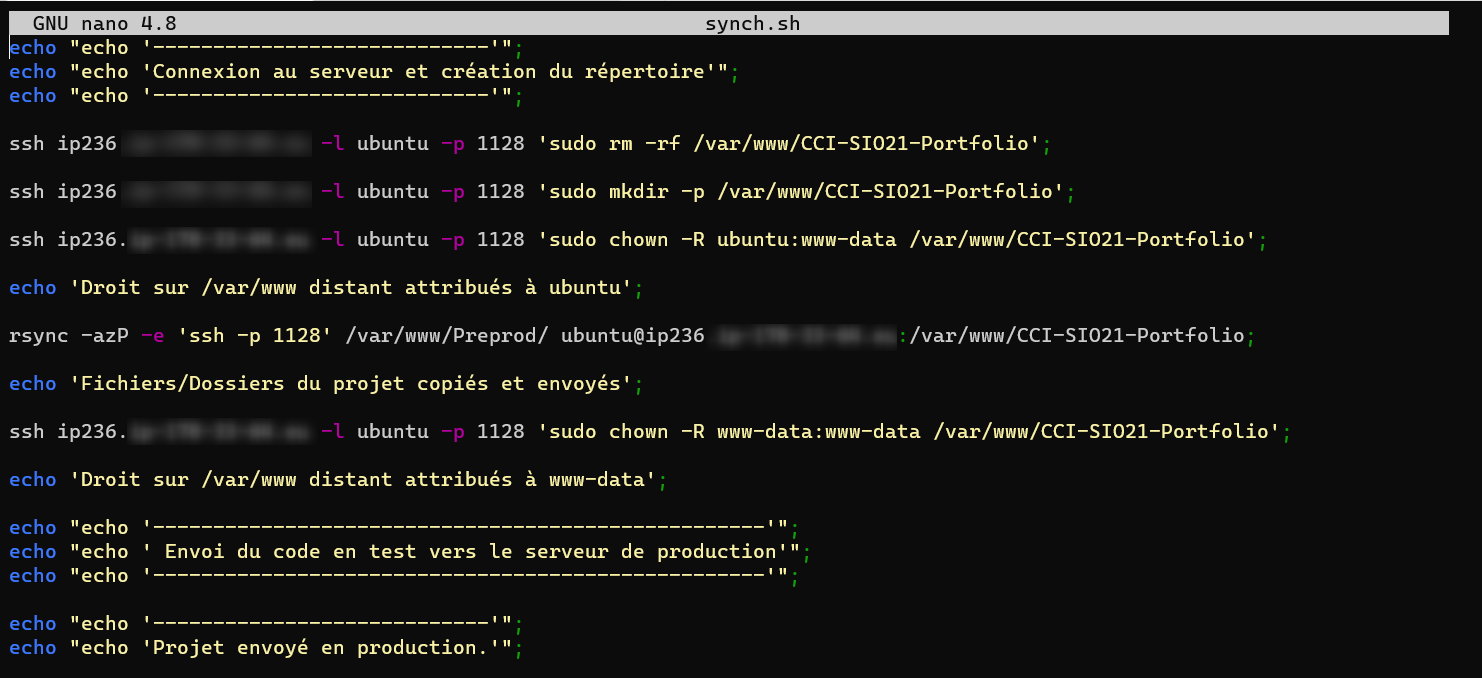
* ssh *(adresse ip du serveur)* -l ubuntu -p 1128 'sudo mkdir -p /var/www/CCI-SIO21-Portfolio';
* ssh *(adresse ip du serveur)* -l ubuntu -p 1128 'sudo chown -R ubuntu:www-data /var/www/CCI-SIO21-Portfolio';

Après avoir effectué ces actions nous allons utiliser notre méthode **rsync** pour transférer nos fichiers présents dans le dossier de Preprod de la machine de **test** au serveur de **production :**

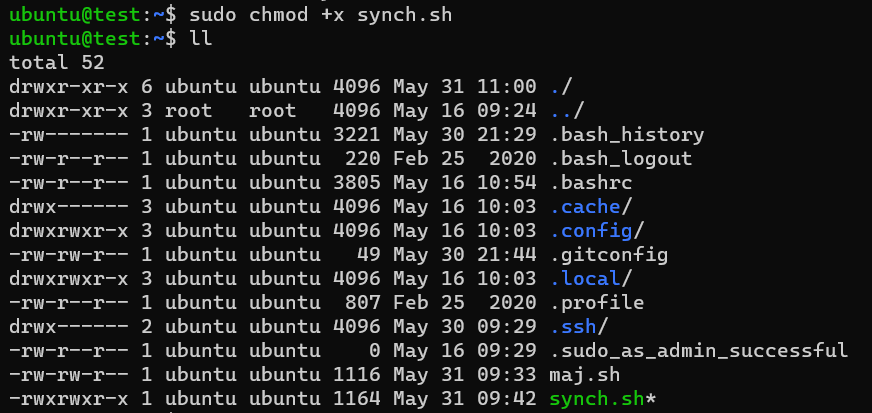
* rsync -azP -e 'ssh -p 1128' /var/www/Preprod/ ubuntu@*(adresse ip du serveur**)*:/var/www/CCI-SIO21-Portfolio;

Nous allons enfin donner les droits à Apache sur le dossier CCI-SIO21-Portfolio :

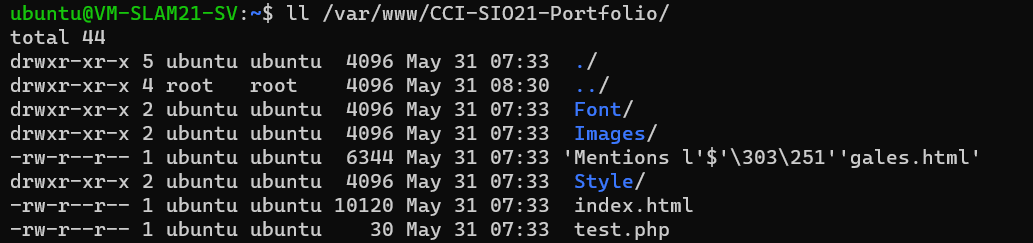
* ssh *(adresse ip du serveur)* -l ubuntu -p 1128 'sudo chown -R www-data:www-data /var/www/CCI-SIO21-Portfolio';



Nous allons enregistrer nos modifications puis on rend le script exécutable avec la commande sudo chmod +x synch.sh



Après avoir exécuté notre script (sudo ./synch.sh) nous allons nous connecter à notre machine de production et vérifions si les informations ont bien été prises en compte.



Nous allons modifier le fichier de Virtual Host *(comme indiqué aux pages 10 et 11)* et allons activer le fichier .conf qui pointe notre projet.

