## Architecture 3-Tiers (Debian 12)

# Sur Parallels Desktop

Ce projet d'architecture 3-Tiers a été réalisé sur Parallels Desktop avec des machines virtuelles sous Debian 12. Toutes les commandes ont été réalisées avec l'utilisateur « **root** ». Pour la réalisation de ce projet nous aurons besoin de 3 machines virtuelles:

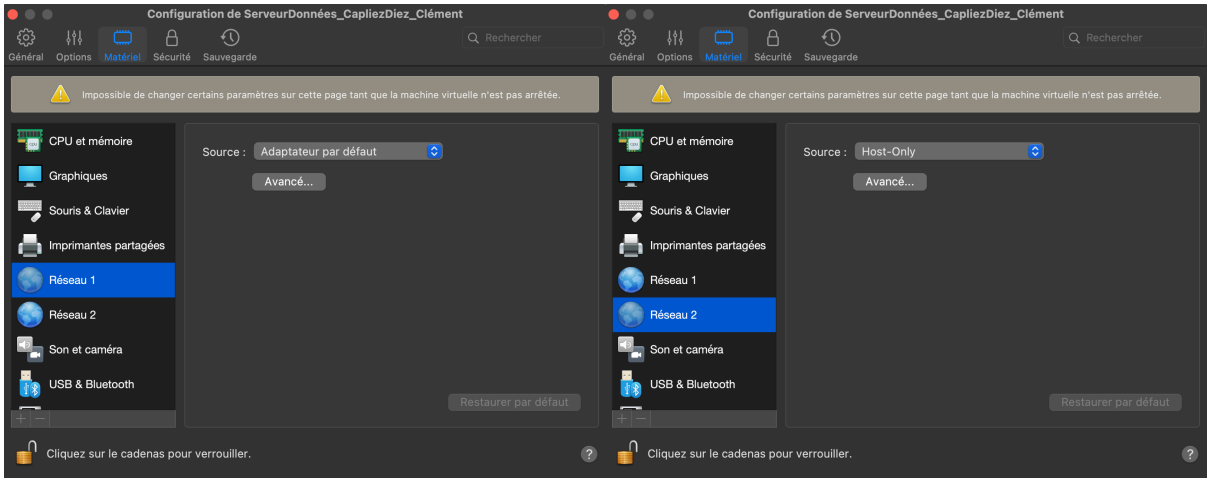
- Un client (interface graphique)
- Un serveur web (interface graphique)
- Un serveur de données (interface non graphique)

Cette documentation sera divisée en 5 parties:

# Configuration réseau des machines virtuelles

## Configuration réseau:

Pour pouvoir accéder à internet tout en attribuant une adresse IP fixe sans passer par un routeur, chaque machine virtuelle possède deux cartes réseau, la première configurée en mode « **Adaptateur par défaut** » (par pont) et l'autre en « **Host-Only** » :




**Les manipulations suivantes sont à effectuer sur chacune de vos machines virtuelles.**

- Pour connaître le nom de vos cartes réseau, il faut réaliser la commande « **ip a** » dans votre terminal:

```
root@serveurweb:/home/administrateur# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:1c:42:08:55:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.121/24 brd 192.168.1.255 scope global dynamic enp0s5
       valid_lft 81993sec preferred_lft 81993sec
    inet6 2001:861:3b84:1530:21c:42ff:fe08:5561/64 scope global dynamic mngtmpaddr
       valid_lft 86295sec preferred_lft 14295sec
    inet6 fe80::21c:42ff:fe08:5561/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:1c:42:7f:2c:ae brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.1/24 brd 192.168.0.255 scope global enp0s6
       valid_lft forever preferred_lft forever
    inet6 fdb2:2c26:f4e4:1:21c:42ff:fe7f:2cae/64 scope global dynamic mngtmpaddr
       valid_lft 2591679sec preferred_lft 604479sec
    inet6 fe80::21c:42ff:fe7f:2cae/64 scope link
       valid_lft forever preferred_lft forever
```

- Ici, le nom de mes cartes réseau sont « **enp0s5** » et « **enp0s6** »

- Pour modifier l'adresse IP de nos machines virtuelles, nous utiliserons la commande « **nano /etc/network/interfaces** » pour accéder à l'interface de configuration de nos cartes réseau et y écrire les lignes suivantes en veillant bien à remplacer « **enp0s5** » et « **enp0s6** » par le nom de vos cartes réseau:

```
allow-hotplug enp0s5
iface enp0s5 inet dhcp

allow-hotplug enp0s6
iface enp0s6 inet static
        address 192.168.0.1/24
```

address: l'adresse IP souhaitée avec un masque en /24

Personnellement, j'ai décidé d'attribuer les adresses IP suivantes à mes machines virtuelles:

- Client: 192.168.0.10
- Serveur Web: 192.168.0.1
- Serveur de données: 192.168.0.2

Pour enregistrer vos modifications, il faut exécuter la commande « **systemctl restart networking** »

## Test de communication:

Pour tester la communication entre vos machines virtuelles, ça se passe avec la commande « **ping x.x.x.x** », les « x » représentant l'adresse IP avec laquelle vous souhaitez communiquer.

En prenant l'exemple avec ma machine client (192.168.0.10), j'arrive à pinger correctement sans erreur mon serveur web (192.168.0.1) ainsi que mon serveur de données (192.168.0.2):

```
administrateur@client:~$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.583 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.599 ms
^C
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1015ms
rtt min/avg/max/mdev = 0.583/0.591/0.599/0.008 ms
administrateur@client:~$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.937 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.343 ms
^C
--- 192.168.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.343/0.640/0.937/0.297 ms
```

Nous pouvons également voir que mon serveur web (192.168.0.1) arrive à communiquer avec le serveur de données (192.168.0.2):

```
root@serveurweb:/home/administrateur# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.530 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.836 ms
^C
--- 192.168.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1017ms
rtt min/avg/max/mdev = 0.530/0.683/0.836/0.153 ms
```

Maintenant que la configuration réseau de nos machines est terminée, nous allons pouvoir passer au vif du sujet de ce projet.

# Mise en place du serveur de données

## Installation LAMP:

Nous allons commencer par installer Apache, MariaDB, libapache2-mod-php et phpMyAdmin.

Afin de réaliser cela, il faut réaliser les commandes suivantes:

- « **apt install apache2** »
- « **apt install mariadb-server** »
- « **apt install phpmyadmin** » (choisir apache2 lors de l'installation)
- « **apt install libapache2-mod-php** » <— Important pour phpMyAdmin
- « **Systemctl reload apache2** »

Maintenant nous pouvons voir que nous avons accès à notre serveur de données et à phpMyAdmin via notre serveur web en indiquant dans la barre de recherche de notre navigateur l'adresse IP de notre serveur de données suivi de /phpmyadmin « **192.168.0.2/phpmyadmin** » :

## Modification de l'utilisateur root:

Nous allons ensuite ajouter un mot de passe à l'utilisateur « **root** » de MySQL ainsi qu'autoriser sa connexion depuis notre serveur web pour pouvoir nous connecter à phpMyAdmin avec les commandes suivantes:

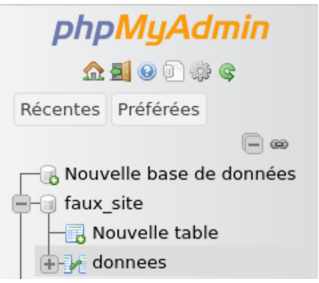
- « **mysql -u root** » <—— Connexion à MySQL
- « **ALTER USER 'root'@'localhost' IDENTIFIED BY 'votre_mot_de_passe';** » <—— Attribution du mot de passe à root
- « **GRANT ALL PRIVILEGES ON *.* TO 'root'@'192.168.0.1' IDENTIFIED BY 'votre_mot_de_passe' WITH GRANT OPTION;** » <— — Autorisation donnée au serveur web de se connecter à l'utilisateur root
- « **FLUSH PRIVILEGES;** » <—— Enregistrement des modifications

## Création de la base de données:

Enfin, je vais créer la base de données qui va servir à stocker les données du futur site de notre serveur web que je vais importer d'un ancien TP avec les commandes suivantes:

- « **CREATE DATABASE faux_site;** » <—— Création de la base de données
- « **USE faux_site;** » <—— Connexion à la base de données
- « **CREATE TABLE donnees (email_tel VARCHAR(100), mdp VARCHAR(100));** » <—— Création de la table avec 2 colonnes
- « **exit** » <—— Déconnexion de MySQL

Nous pouvons maintenant voir via notre serveur web que la base de données s'est correctement créee via l'interface phpMyAdmin:




Notre serveur de données est presque prêt à recevoir les données que nous allons lui envoyer via notre serveur web, cependant, il reste un petit détail, il faut que notre serveur de données accepte les connexions distantes.

Pour cela, nous utiliserons la commande « **nano /etc/mysql/mariadb.conf.d/50-server.cnf** ». Une fois que vous êtes dans le fichier, cherchez la ligne « **bind-address** » et remplacez la valeur « **127.0.0.1** » par « **0.0.0.0** » :

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
```

Et voilà ! Notre serveur de données est prêt.

Passons maintenant au serveur web.

# Mise en place du serveur web

## Installation des paquets d'Apache et des paquets nécessaires:

Tout d'abord, installons apache2, PHP et php-mysqli:

- « **apt install apache2** »
- « **apt install php** » <— Important pour que votre script PHP fonctionne
- « **apt install php-mysqli** » <— Important pour la connexion à la BDD

## Importation du site:

J'ai remplacé le dossier « **html** » situé dans le répertoire « **/var/www/** » par mon propre dossier **html** contenant les fichiers de mon site.

Veillez bien à ce que dans votre fichier php contenant le script de connexion à la base de données, la variable « **$servername** » soit bien définie avec l'adresse IP du serveur de données:

```
function connectToDatabase() {
    $servername = "192.168.0.2";
    $username = "root";
    $password = "root"; // Remplacez root par votre mot de passe MySQL
    $dbname = "faux_site"; // Remplacez faux_site par le nom de votre base de données
```

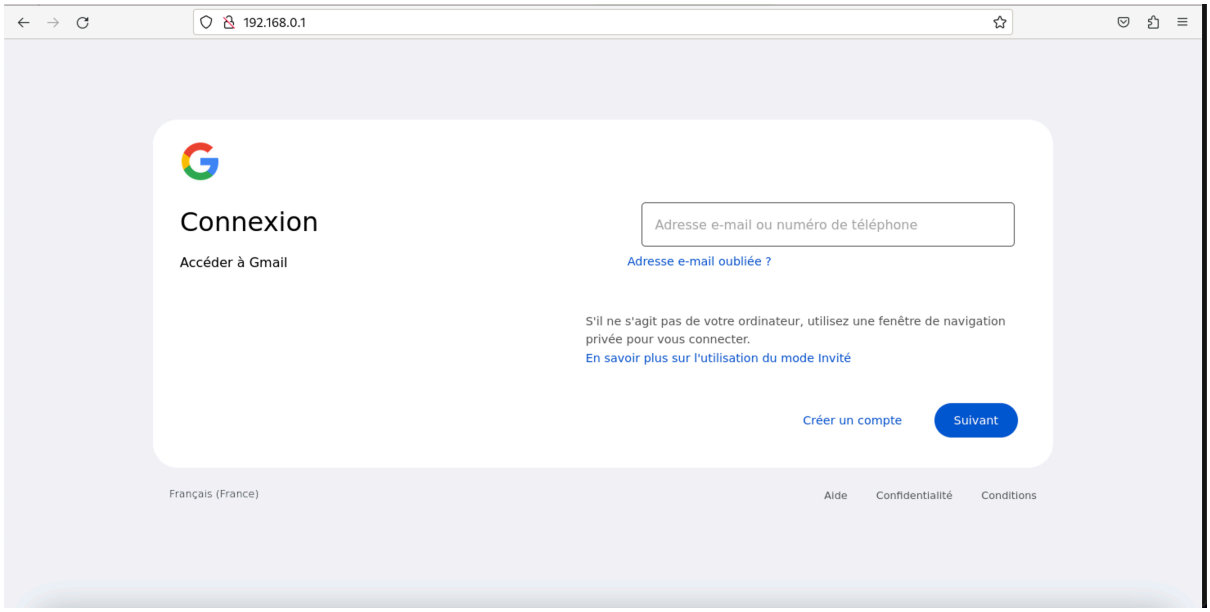
Nous avons maintenant une petite modification à faire dans le fichier de configuration du VirtualHost d'apache pour modifier la page d'accueil d'apache qui est affichée lorsque nous entrons l'adresse IP de notre serveur web dans la barre de recherche.

Pour se faire, il faut réaliser la commande « **nano /etc/apache2/sites-available/000-default.conf** ». Une fois dans le fichier, rajoutez la directive « **DirectoryIndex** » sous le « DocumentRoot » suivi du nom de votre fichier que vous souhaitez afficher en guise de page d'accueil de votre site:

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
DirectoryIndex identifier.php
```

Pour ma part, je souhaite afficher le fichier « **identifier.php** »

Maintenant, en mettant l'adresse IP de votre serveur web depuis votre client dans la barre de recherche de votre navigateur, le fichier que vous avez choisi va s'afficher:




Félicitations ! Maintenant que nos deux serveurs sont prêts, nous allons pouvoir restreindre l'accès au serveur de données depuis le client étant donné qu'il ne doit pas y avoir accès.

# Restriction d'accès du client au serveur de données

## Création de la règle de pare-feu:

Pour restreindre l'accès de notre machine client au serveur de données, nous allons créer une règle de pare-feu via notre serveur de données empêchant le client de pouvoir accéder à ce dernier en lui bloquant l'accès au port **80**.

Pour se faire, procédons par étape en suivant les commandes ci-dessous à réaliser sur le serveur de données:

- « **apt install sudo** » <—— Installation du paquet sudo nécessaire
- « **apt instal iptables** » <—— Installation d'iptables
- « **sudo iptables -A INPUT -s 192.168.0.10 -d 192.168.0.2 -p tcp —dport 80 -j DROP** » <—— Restriction client au port 80 du serveur de données
- « **apt install iptables-persistent** » <—— Sauvegarder les règles créees

## Test de la restriction:

Maintenant, installons « **nmap** » sur notre client pour pouvoir voir quels ports sont ouverts sur le serveur de données avec la commande « **apt Install nmap** ».

Une fois installé, il faut exécuter la commande « **nmap x.x.x.x** », les « x » représentant l'adresse IP de votre serveur de données.

Vous devriez avoir ceci qui apparaît:

```
root@client:/home/clem# nmap 192.168.0.2
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-20 20:08 CEST
Nmap scan report for 192.168.0.2
Host is up (0.00043s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE    SERVICE
80/tcp   filtered http
3306/tcp open     mysql
MAC Address: 00:1C:42:B1:88:0D (Parallels)

Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds
```

Nous pouvons voir que l'état du port 80 n'est plus « **open** » mais « **filtered** », ce qui veut dire que notre règle de pare-feu a correctement été appliquée.

En revanche, lorsque nous utilisons la commande nmap en destination de notre serveur web cette fois, nous remarquons que nous avons bien accès au port 80 de ce dernier:

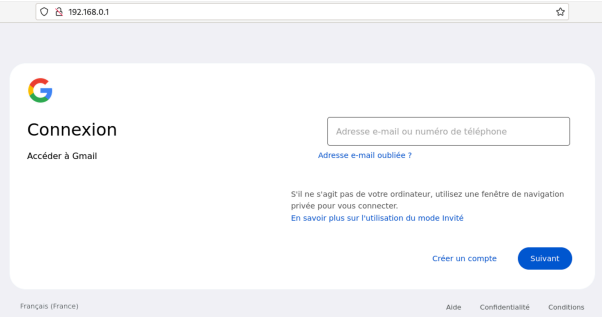
```
root@client:/home/clem# nmap 192.168.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-20 20:14 CEST
Nmap scan report for 192.168.0.1
Host is up (0.00013s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:1C:42:7F:2C:AE (Parallels)

Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
```

Ici, l'état du port 80 est bien « **open** ».

Testons maintenant la connexion au serveur de données via le navigateur de notre client en mettant l'adresse IP du serveur de données dans la barre de recherche:


Nous remarquons qu'un message d'erreur indiquant que le délai d'attente est dépassé apparaît, signifiant bien que notre client n'a pas accès au serveur de données.

Testons maintenant la connexion à notre serveur web via le navigateur:



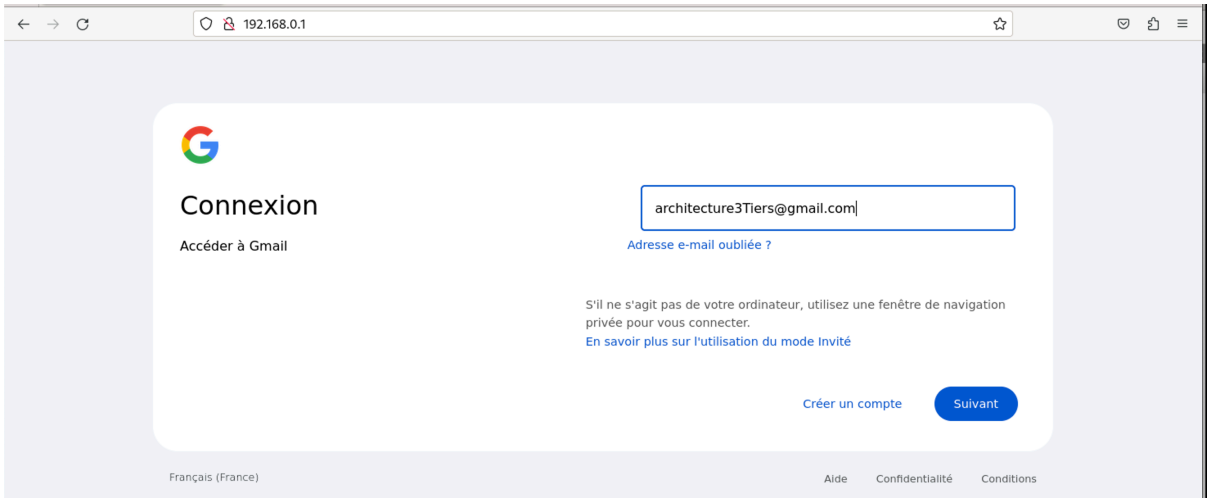

La connexion fonctionne toujours parfaitement.

Maintenant que nous avons bien empêché le client de se connecter au serveur de données, nous pouvons passer au test final de ce projet pour voir si notre architecture 3-Tiers est fonctionnelle.

# Test final de l'architecture 3-Tiers

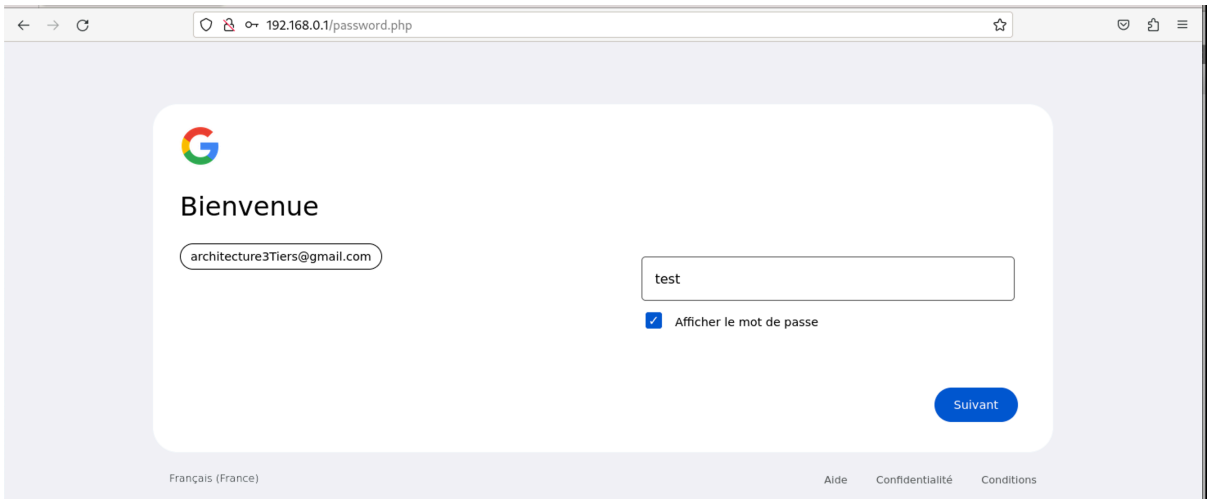
Il est maintenant temps de passer au test final de notre architecture.

Via notre client, accédons à notre serveur web via le navigateur pour accéder au site que nous avons installé et rentrons-y des données dans le formulaire:




J'ai renseigné « **architecture3Tiers@gmail.com** » dans le formulaire en guise d'e-mail.

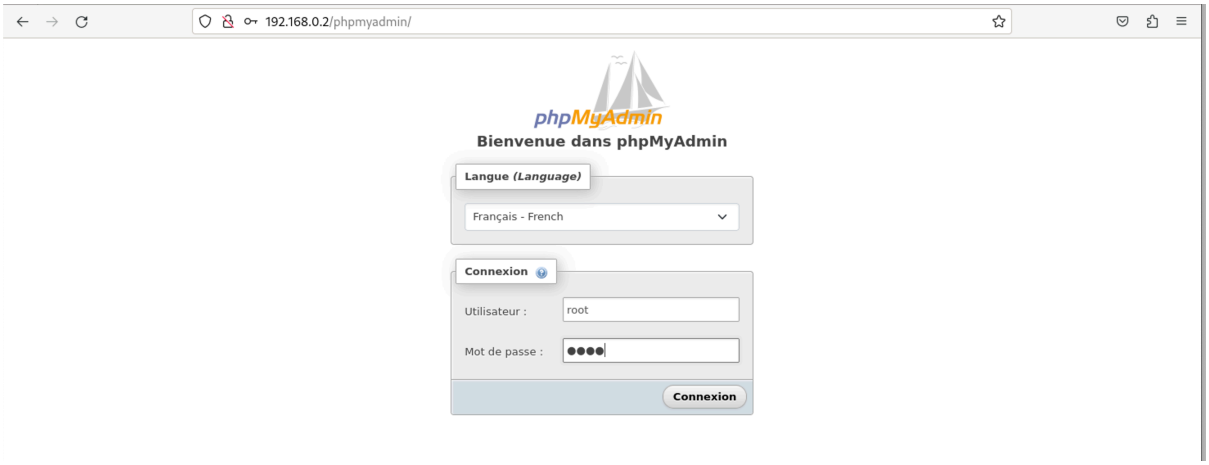
Après, cliquons sur « **suivant** » pour renseigner notre mot de passe:




En guise de mot de passe, j'ai choisi « **test** ».

Cliquons une dernière fois sur « **suivant** » pour envoyer nos données à la BDD.

Maintenant que nous avons envoyés nos données à notre serveur de données, connectons nous à phpMyAdmin via le navigateur de notre serveur web:




Je rappelle que pour se connecter à phpMyAdmin, nous utilisons le mot de passe que nous avons donné à l'utilisateur « **root** » précédemment lors de la mise en place de notre serveur de données. Dans mon cas, le mot de passe que je lui ai donné était « **root** ».

Accédons maintenant à la table de notre base de données pour vérifier si les données ont bien été collectées:


Nous remarquons que les données « **architecture3Tiers@gmail.com** » et « **test** » sont bien arrivées dans la table de notre base de donnée.

Je vous félicite d'être arrivés jusqu'au bout de ce tutoriel ! Vous êtes maintenant en capacité de pouvoir réaliser une architecture 3-Tiers fonctionnelle avec un client, un serveur web et un serveur de données.