Compte-rendu TP
Architecture client/serveur

I. Ressources utilisées pour le TP

Pour ce TP, nous avons utilisé les tutoriels mis à notre disposition dans les documents de références ainsi qu'Internet.
Pour les machines virtuelles nous avons utilisé *VMware Fusion 2024*. Sur notre VM, nous avons utilisé la distribution Linux *Ubuntu Server*. Dessus, nous avons installé *MariaDB*, *phpMyAdmin* et *Apache2*.

II. Installation d'Apache2

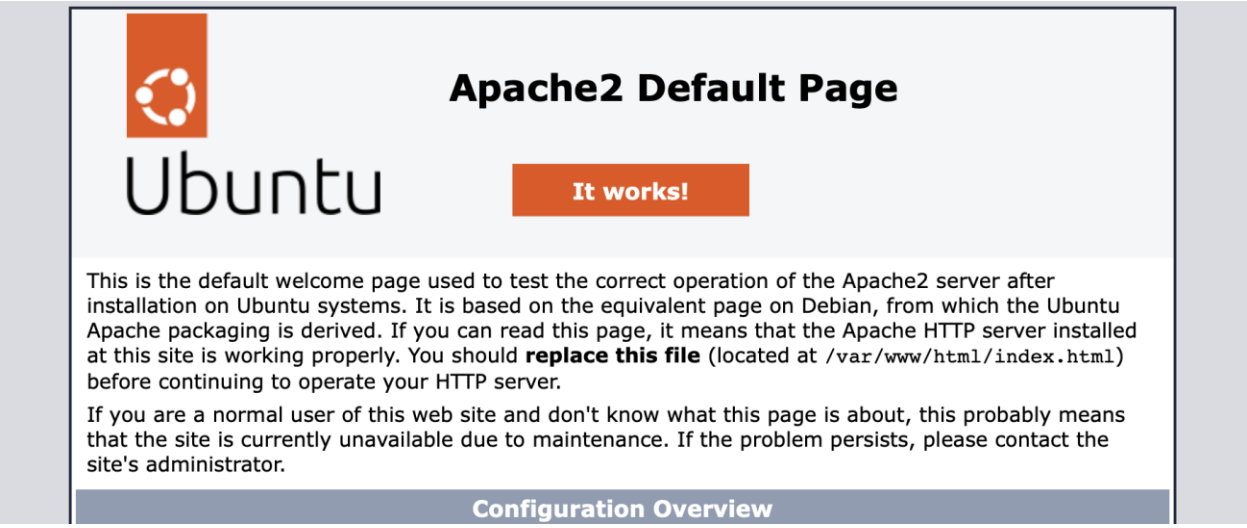
On commence par mettre à jour la liste des paquets disponibles sur notre machine avec la commande 'sudo apt update'.

```
alex@ubuntu:~$ sudo apt update
[sudo] password for alex:
Atteint :1 http://ports.ubuntu.com/ubuntu-ports noble InRelease
Atteint :2 http://ports.ubuntu.com/ubuntu-ports noble-updates InRelease
Atteint :3 http://ports.ubuntu.com/ubuntu-ports noble-backports InRelease
Atteint :4 http://ports.ubuntu.com/ubuntu-ports noble-security InRelease
Réception de :5 http://ports.ubuntu.com/ubuntu-ports noble/main Translation-fr [491 kB]
Réception de :6 http://ports.ubuntu.com/ubuntu-ports noble/restricted Translation-fr [3 292 B]
Réception de :7 http://ports.ubuntu.com/ubuntu-ports noble/universe Translation-fr [3 898 kB]
Réception de :8 http://ports.ubuntu.com/ubuntu-ports noble/multiverse Translation-fr [89,0 kB]
4 481 ko réceptionnés en 1s (6 463 ko/s)
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
35 paquets peuvent être mis à jour. Exécutez « apt list --upgradable » pour les voir.
```

On peut ensuite installer *Apache2* avec la commande 'sudo apt install apache2'. Ci-dessous un screenshot de l'installation du paquet.

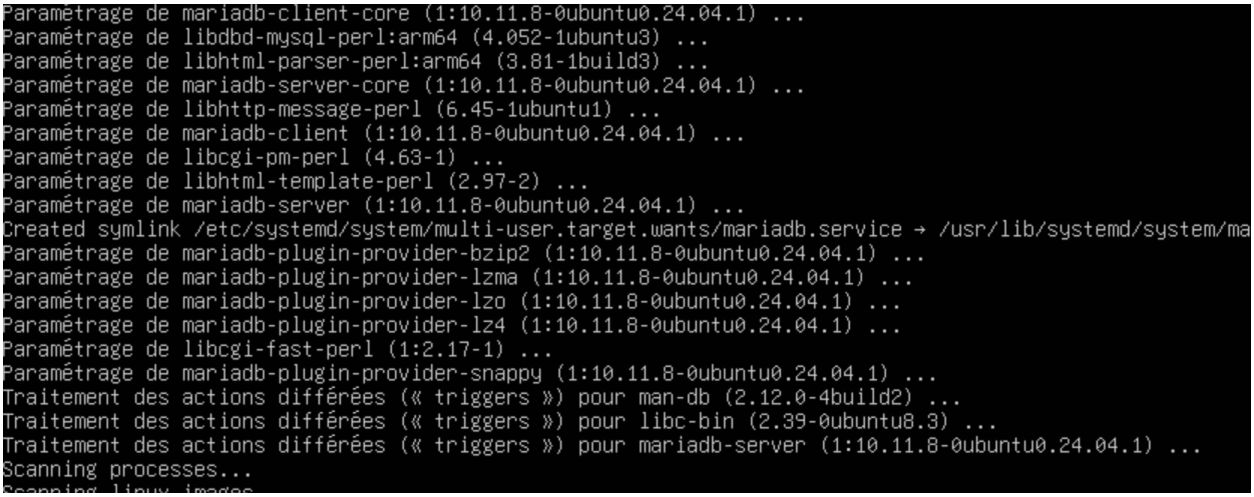
```
Paramétrage de apache2-data (2.4.58-1ubuntu8.4) ...
Paramétrage de libaprutil1t64:arm64 (1.6.3-1.1ubuntu7) ...
Paramétrage de libaprutil1-ldap:arm64 (1.6.3-1.1ubuntu7) ...
Paramétrage de libaprutil1-dbd-sqlite3:arm64 (1.6.3-1.1ubuntu7) ...
Paramétrage de apache2-utils (2.4.58-1ubuntu8.4) ...
Paramétrage de apache2-bin (2.4.58-1ubuntu8.4) ...
Paramétrage de apache2 (2.4.58-1ubuntu8.4) ...
Enabling module mpm_event.
Enabling module authz_core.
Enabling module authz_host.
Enabling module authn_core.
Enabling module auth_basic.
Enabling module access_compat.
Enabling module authn_file.
Enabling module authz_user.
Enabling module alias.
Enabling module dir.
Enabling module autoindex.
Enabling module env.
Enabling module mime.
```

On peut vérifier qu'*Apache2* fonctionne correctement en prenant l'adresse IP de notre machine et en la mettant dans la barre d'URL de notre moteur de recherche. Si tout fonctionne on obtient une page web *Apache2* avec écrit : 'it works!'.



III. Installation de MariaDB

On installe par la suite *MariaDB* avec la commande 'sudo apt install mariadb-server'. Screenshot de l'installation en dessous.



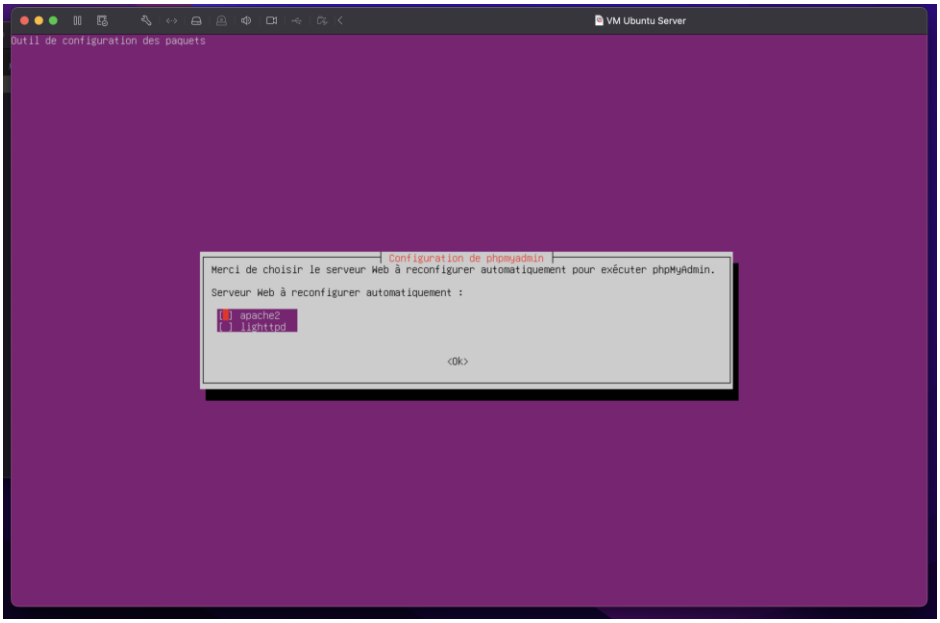
IV. Installation de php

Une fois fait on installe *php* dans la version qui convient à notre machine avec la commande ‘sudo apt install php libapache2-mod-php'. Pour nous c’est la version 8.3 de *php*. Screenshot de l’installation en dessous.

```
Paramétrage de mariadb-client-core (1:10.11.8-0ubuntu0.24.04.1) ...
Paramétrage de libdbd-mysql-perl:arm64 (4.052-1ubuntu3) ...
Paramétrage de libhtml-parser-perl:arm64 (3.81-1build3) ...
Paramétrage de mariadb-server-core (1:10.11.8-0ubuntu0.24.04.1) ...
Paramétrage de libhttp-message-perl (6.45-1ubuntu1) ...
Paramétrage de mariadb-client (1:10.11.8-0ubuntu0.24.04.1) ...
Paramétrage de libcgi-pm-perl (4.63-1) ...
Paramétrage de libhtml-template-perl (2.97-2) ...
Paramétrage de mariadb-server (1:10.11.8-0ubuntu0.24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /usr/lib/systemd/system/ma
Paramétrage de mariadb-plugin-provider-bzip2 (1:10.11.8-0ubuntu0.24.04.1) ...
Paramétrage de mariadb-plugin-provider-lzma (1:10.11.8-0ubuntu0.24.04.1) ...
Paramétrage de mariadb-plugin-provider-lzo (1:10.11.8-0ubuntu0.24.04.1) ...
Paramétrage de mariadb-plugin-provider-lz4 (1:10.11.8-0ubuntu0.24.04.1) ...
Paramétrage de libcgi-fast-perl (1:2.17-1) ...
Paramétrage de mariadb-plugin-provider-snappy (1:10.11.8-0ubuntu0.24.04.1) ...
Traitement des actions différées (« triggers ») pour man-db (2.12.0-4build2) ...
Traitement des actions différées (« triggers ») pour libc-bin (2.39-0ubuntu8.3) ...
Traitement des actions différées (« triggers ») pour mariadb-server (1:10.11.8-0ubuntu0.24.04.1) ...
Scanning processes...
Scanning linux images...
```

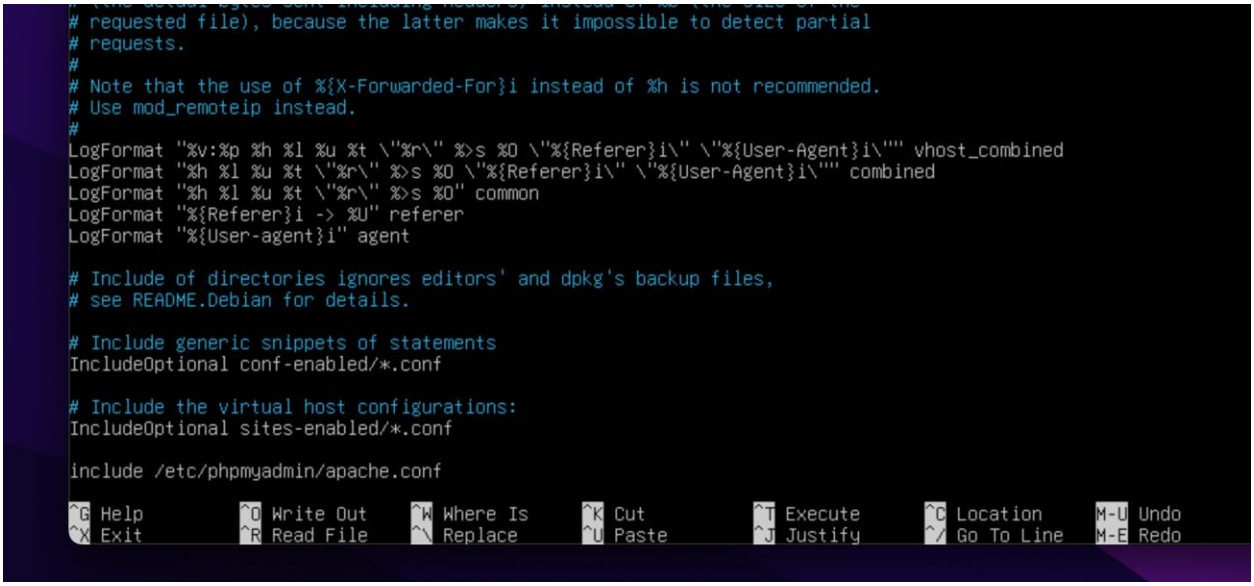
V. Installation de phpMyAdmin

On peut passer à l’installation du dernier paquet dont on a besoin, *phpMyAdmin*. On l’installe avec la commande ‘sudo apt install phpmyadmin’. Quand on effectue la commande, une fenêtre s’ouvre et nous demande quel serveur web doit être configuré pour *phpMyAdmin*, on sélectionne *Apache2*.



Après avoir installé *phpMyAdmin*, on peut redémarrer *Apache2* avec la commande ‘sudo service apache2 restart’.

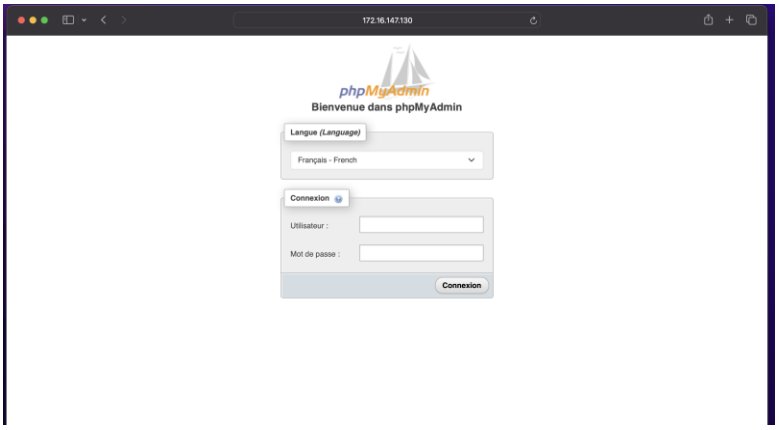
Par la suite, on doit configurer *Apache2* pour le faire fonctionner avec *phpMyAdmin*, pour ce faire, on doit éditer le fichier configuration d'*Apache2* pour donner l'accès à *phpMyAdmin*. On utilise la commande 'sudo nano /etc/apache2/apache2.conf'. Elle nous ouvre l'éditeur de texte nano et affiche ceci :



Comme affiché sur le screenshot, on ajoute à la fin du fichier la ligne 'include /etc/phpmyadmin/apache.conf'. On fait ^O (Ctrl + O) et entrer pour enregistrer, puis ^X (Ctrl + X) pour sortir de l'éditeur de texte.

Après ça, on redémarre encore une fois *Apache2* avec la commande 'sudo service apache2 restart'.

Pour vérifier si on a bien lié notre serveur *Apache2* et *phpMyAdmin*, on peut retourner sur l'URL de notre page Apache2 en ajoutant à la fin '/phpmyadmin'. Dans notre cas, on va donc sur la page '172.16.147.130/phpmyadmin' et on obtient cette page qui nous indique que tout est bien lié :



Maintenant, on va créer un nom d'utilisateur et un mot de passe pour accéder à notre base de données. Pour ce faire on doit entrer dans l'interface de commande de MariaDB ne tapant la commande 'sudo mariadb –u root –p'.

Une fois dans l'invite de commande *MariaDB*. On tape la commande "CREATE USER 'alex'@'localhost' IDENTIFIED BY 'alex';" pour créer notre nom d'utilisateur et notre mot de passe.

```
MariaDB [(none)]> CREATE USER 'alex'@'localhost' IDENTIFIED BY 'alex';
Query OK, 0 rows affected (0,005 sec)
```

On lui donne ensuite tous les droits sur la base de données avec la commande "GRANT ALL PRIVILEGES ON *.* TO 'alex'@'localhost' WITH GRANT OPTION;".

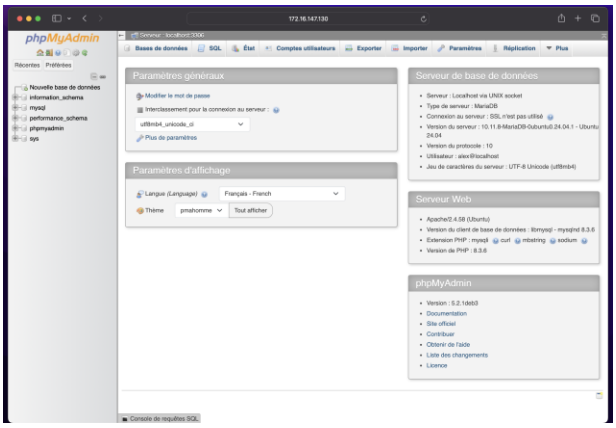
```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'alex'@'localhost' WITH GRANT OPTION;
Query OK, 0 rows affected (0,002 sec)
```

On peut ensuite partir de l'invite de commande *MariaDB*.

```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,002 sec)

MariaDB [(none)]> EXIT
Bye
```

On peut enfin tenter d'accéder à notre base de données, en se connectant. Si tout marche on obtient ceci :



VI. Installation de SSL

On commence par activer mod_ssl avec la commande 'sudo a2enmod ssl'.

```
alex@ubuntu:~$ sudo a2enmod ssl
[sudo] password for alex:
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

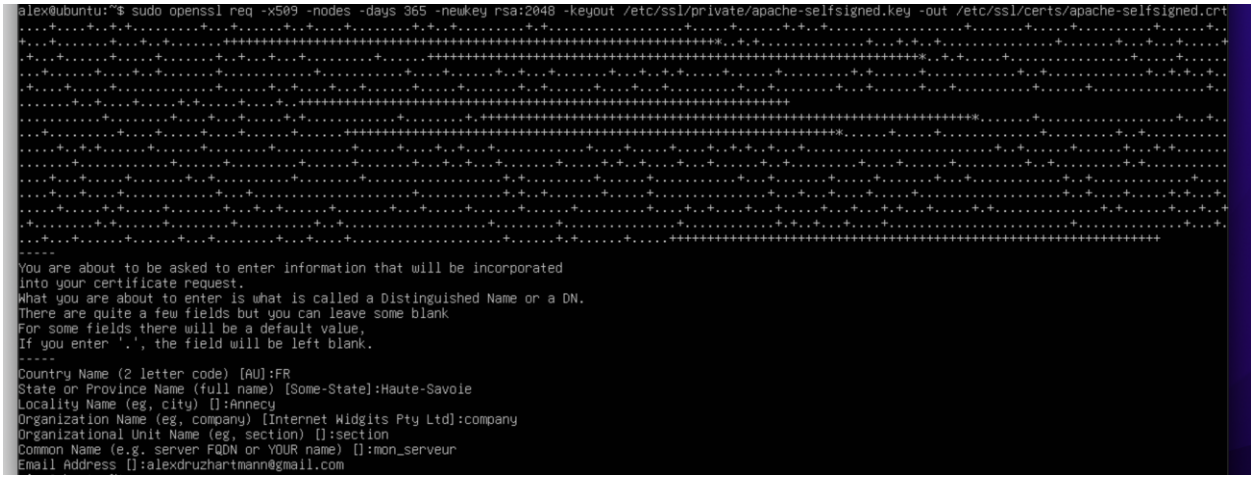
On peut ensuite redémarrer Apache2 avec la commande 'systemctl restart apache2'.

```
alex@ubuntu:~$ systemctl restart apache2
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to restart 'apache2.service'.
Authenticating as: alex
Password:
==== AUTHENTICATION COMPLETE ====
alex@ubuntu:~$ systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-09-30 16:16:38 UTC; 7s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 1540 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 1543 (apache2)
      Tasks: 6 (limit: 4550)
     Memory: 14.0M (peak: 14.4M)
        CPU: 23ms
     CGroup: /system.slice/apache2.service
             ├─1543 /usr/sbin/apache2 -k start
             ├─1545 /usr/sbin/apache2 -k start
             ├─1546 /usr/sbin/apache2 -k start
             ├─1547 /usr/sbin/apache2 -k start
             ├─1548 /usr/sbin/apache2 -k start
             └─1549 /usr/sbin/apache2 -k start

sept. 30 16:16:38 ubuntu systemd[1]: Starting apache2.service - The Apache HTTP Server...
sept. 30 16:16:38 ubuntu apachectl[1542]: [Mon Sep 30 16:16:38.747879 2024] [alias:warn] [pid 1542] AH00671: The Alias directive in /etc/phpmyadmin/apache.conf
sept. 30 16:16:38 ubuntu apachectl[1542]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'Se
sept. 30 16:16:38 ubuntu systemd[1]: Started apache2.service - The Apache HTTP Server.
```

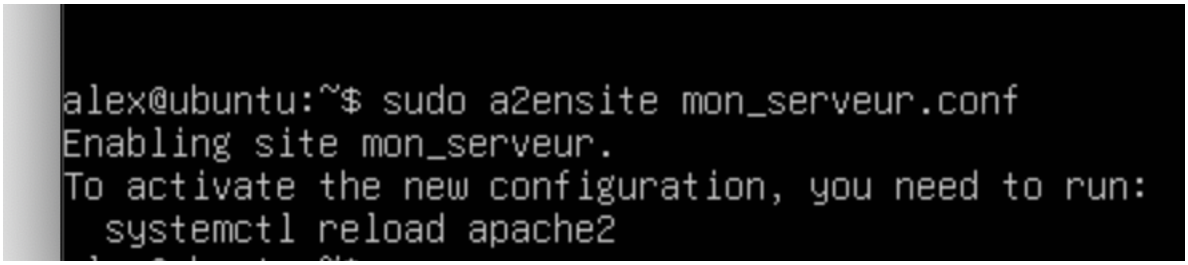
Ensuite, on doit générer notre certificat TLS, qui stocke les informations à propos de notre site. Et nous génère un fichier clé qui permet au serveur de manipuler les données encryptées. Pour ce faire, on utilise la longue commande 'sudo openssl req –x509 –nodes

–days 365 –newkey rsa:2048 –keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt'. On va obtenir une liste de sélection où on doit entrer les informations liées à notre serveur web.



On va ensuite configurer *Apache2* pour qu'il utilise notre certificat et notre clé. Pour faire ça, on utilise la commande 'sudo nano /etc/apache2/sites-available/mon_serveur.conf'. Cette commande va nous permettre d'éditer notre fichier de configuration *Apache2* et de le charger pour notre serveur web. Dans l'éditeur de texte, on écrit les lignes suivantes :

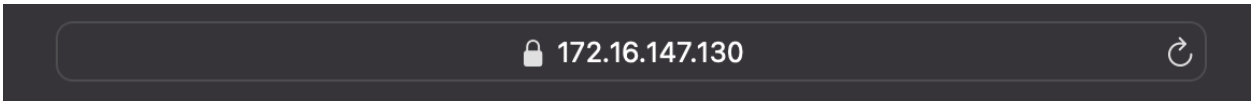
On peut ensuite activer le fichier de configuration avec la commande 'sudo a2ensite mon_serveur.conf'.



On fait ce que l'output nous demande de faire, c'est-à-dire reload *Apache2*. On le fait avec la commande 'sudo systemctl reload apache2'.

```
alex@ubuntu:~$ sudo systemctl reload apache2
```

Une fois tout ceci fait, on peut essayer de charger notre page web *Apache2* en mettant l'URL 'https://172.16.147.130'. Lorsqu'on fait ça, on reçoit une alerte du moteur de recherche qui nous prévient qu'il ne peut pas vérifier l'identité du serveur à cause de notre certificat que l'on a généré nous-même, cela veut dire que tout fonctionne. On peut décider de continuer et arriver sur notre page.



Pour finir, on va faire en sorte de rediriger les requêtes HTTP vers HTTPS. On va donc modifier le fichier mon_serveur.conf que l'on a déjà modifié plus haut avec la commande 'sudo nano /etc/apache2/sites-available/mon_serveur.conf'. Dedans, on y entre les lignes suivantes :

```
<VirtualHost *:80>
        ServerName mon_serveur
        Redirect / https://mon_serveur/
</VirtualHost>
```

On enregistre et on quitte l'éditeur de texte. Ensuite, on peut reload *Apache2* une dernière fois avec la commande 'sudo systemctl reload apache2'.