Touiller

Solutions d'impression & informatiques

Groupe TOUILLER

Installation d'un serveur LAMP

Rédacteur : Rudy

Date : 31/01/2024

Version : 1.0

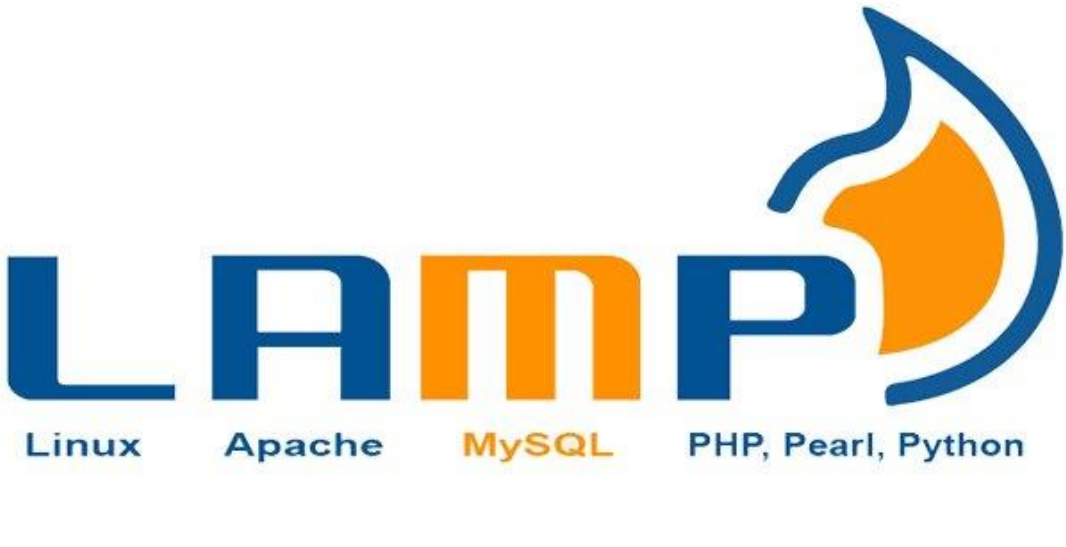


Table des matières

Les prérequis

Dans cette documentation, nous allons voir comment déployer un serveur web LAMP. Que veut dire L'AMP ?

- L pour Linux
- A pour Apache
- M pour MySQL/MariaDB
- P pour PHP

Notons qu'il sera nécessaire au préalable d'avoir une machine virtuelle unbuntu ou debian administrable via putty en SSH. Si ce n'est pas le cas, je vous invite à aller voir la procédure afin de configurer une machine virtuelle linux et administrable en ssh via PuTTY.

Avant de commencer l'installation de votre premier serveur WEB. Veuillez mettre à jour votre machine linux.

Pour ce faire, connecter vous en ssh sur votre machine virtuelle puis entrer les commandes suivantes.

Notons qu'il vous sera demandé le mot de passe du root. Il est donc nécessaire de le connaître.

- Sudo apt-get update
- Sudo apt-get install

Installation de APACHE sous UNBUNTU

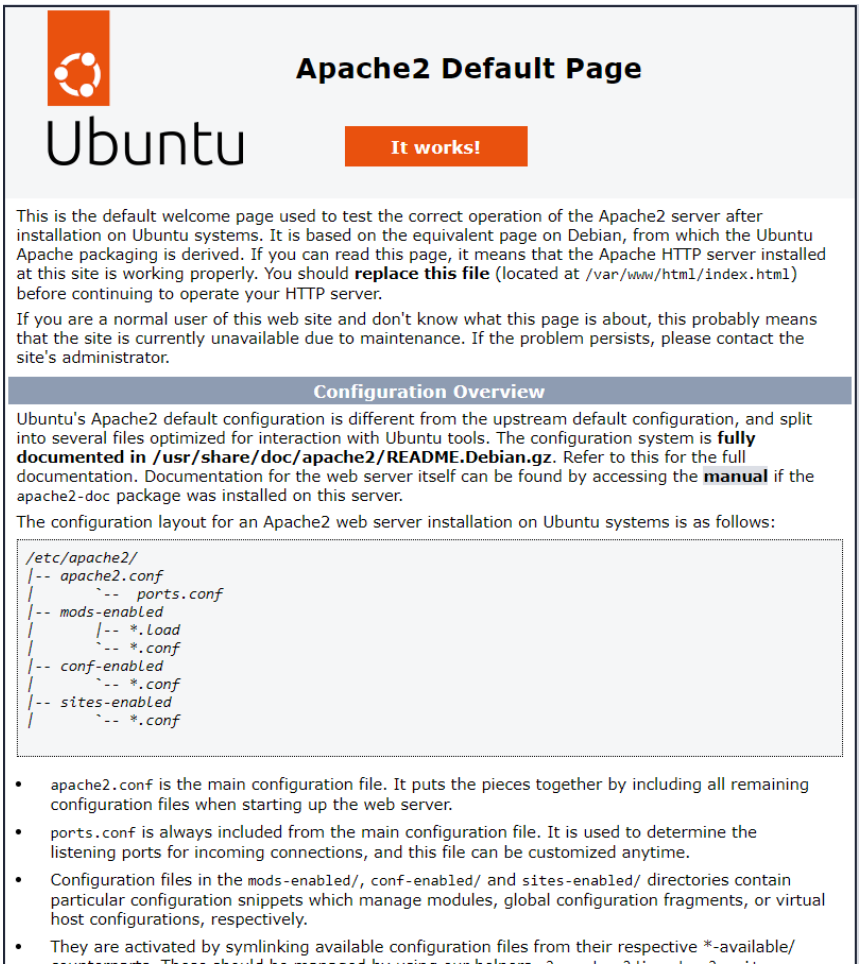
Nous allons donc installer maintenant le paquet APACHE 2.

- **Entrer la commande suivante :**
- **sudo apt-get install Apache2**. Il vous sera demander de confirmer l'installation par **Y**.
- Si vous voulez qu'APACHE démarre automatiquement : entrer la commande suivante :
- **sudo systemctl enable apache2**

À la suite de l'installation du paquet, le serveur Apache démarre directement. On devrait pouvoir accéder à sa page par défaut Pour cela, il suffit de récupérer l'adresse IP du serveur.

Entrer la commande suivante pour récupérer l'adresse ip de la machine : **ip address** dans le terminal de commande.

Ouvrez un navigateur web puis entrer dans la barre de recherche l'adresse ip récupérer au préalable.



Vous devriez arriver sur cette page.

Avant d'aller plus, Il est nécessaire d'activer quelque module d'Apache qui sont indispensables, notamment pour faire tourner un site Internet. Commençons par le module utilisé pour la réécriture d'URL.

Entrer la commande suivante :

- **sudo a2enmod rewrite.**

Nous allons maintenant installer deux autres modules essentiels. Le module SSL qui permettra de générer un certificat pour passer en HTTPS puis le module HEADERS pour modifier les en têtes HTTP.

Entrer les deux commandes suivantes :

- **sudo a2enmod ssl**
- **sudo a2enmod headers**

Il faut maintenant redémarrer le service Apache pour que les modifications ou installation de module soient prises en compte.

Entrer la commande suivante : **sudo systemctl restart Apache2**

Voilà, votre service Apache est maintenant prêt.

Installation de PHP sous UNBUNTU

PHP va être nécessaire afin de pouvoir traiter les scripts liés aux pages .php

Nous allons donc l'installer. Depuis le terminal entrer la commande suivante.

- **sudo apgt-get install php**
- **TAPER Y lorsque vous y êtes demandés**

Nous voyons qu'il installe une multitude de paquet. Nous allons devoir installer des paquets supplémentaires pour permettre les interactions entre PHP et notre instance MariaDB (base de données)

Entré la commande suivante :

sudo apt-get install -y php-pdo php-mysql php-zip php-gd php-mbstring php-curl php-xml php-pear php-bcmath

Vous pouvez maintenant vérifier sur quelle version de php vous êtes en entrant la commande suivante :

- **Php –v**

Installation de MariaDB sous UNBUNTU

Installons donc maintenant notre base de données. (Ceci pourrait-être nécessaire dans le cas par exemple d'une installation de GLPI qui nécessite une base de données pour exister.)

Toujours sur le terminal entrer la commande suivante :

- sudo apt-get install mariadb-server

Lorsque vous y êtes invités taper Y

Nous allons maintenant modifier le fichier de configuration de la base de données afin de la sécurisée.

Entrer la commande suivante dans le terminal :

- sudo mariadb-secure-installation

Avant d'entamer le questionnaire : vous serez invité à changer le mot de passe root pour la connexion à la base de données. **CE MOT DE PASSE dans la BONNE PRATIQUE DOIT CONTENIR 12 CARACTERES.**

Suivez l'interrogatoire comme dessous

```
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.
```

Avant de passer à la suite : vérifier que vous arrivez bien à vous connecter à votre base de données.

Entrer la commande suivante :

- **sudo mysql : si besoin entrer le mot de passe root.**

Vous pouvez vérifier votre base de donnée avec la commande suivante :

- **Show database;**

Pour sortir de Mariadb vous pouvez exécuter la touche ctlr + c

Voilà BRAVO !! Vous venez de déployer votre premier serveur WEB !

Configurer un VirutalHost pour Apache

Dans allons donc maintenant voire comment déployer un Virtual HOST. L'objectif des hôtes virtuels nous permettrons d'héberger plusieurs sites sur le même serveur que nous venons de déployer. Dans chaque hôte virtuel, nous pouvons configurer indépendamment les options et notamment le nom de domaine associé au site, ainsi que le dossier sur le serveur qui correspond aux données du site. Le nom de domaine est très important dans la configuration car lorsqu'Apache va recevoir une requête il va savoir à quel site la redirection est destinée.

Notons que dans notre cas, le domaine n'est pas déclaré auprès des services de google et ceci aura son importance par la suite mais nous y reviendrons ensuite.

TOUJOURS DEPUIS l'INVITE DE COMMANDE

1 – La première étape, consiste à créer une structure de répertoire qui contiendra les données du site que nous allons servir aux visiteurs. Nous allons créer ici un répertoire pour l'hôte virtuel que nous prévoyons de développer.

sudo mkdir -p /var/www/Nomdomaine.local/public_html

Remplacer "Nomdomaine.local" par le nom de domaine que vous voulez créer se terminant par.local

2 – Accorder les autorisations nécessaires :

Nous avons maintenant la structure de répertoire pour nos fichiers, mais ils sont la propriété de notre utilisateur root. Si nous voulons que notre utilisateur habituel puisse modifier des fichiers dans nos répertoire web, nous pouvons changer la propriété en faisant ceci :

sudo chown -R $USER:$USER /var/www/Nomdomaine.local/public_html

N'OUBLIER PAS DE REMPLACER Nomdomaine par le vôtre !

La variable $USER prendra la valeur de l'utilisateur sous lequel vous êtes actuellement connecté lorsque vous appuyé sur ENTER. En faisant cela, notre utilisateur habituel possède maintenant les sous répertoires publics_html ou nous allons stocker notre contenu.

Nous devons également modifier nos autorisations afin de garantir que l'accès en lecture est autorisé au répertoire général du web et à tous les fichiers et dossiers qu'il contient, afin que les pages puissent être servie correctement :

Taper la commande suivante :

sudo chmod -R 755 /var/www

3 – Créer une page de démonstration :

Nous allons faire une page index.html :

Taper la commande suivante : nano /var/www/Nomdomaine.local/public_hmtl/index.html

N'OUBLIER PAS DE REMPLACER Nomdomaine par le vôtre !

```
<html>
  <head>
    <title>Bienvenu sur votre premier site !</title>
  </head>
  <body>
    <h1>Félicitation TOUT FONCTIONNE ! ! </h1>
  </body>
</html>
```

Enregistrer et fermer le fichier (dans nano , appuyer sur CTRL + O puis ENTER puis CTRL + X)

4 – Créez de nouveaux fichiers d'hôte virtuel :

Apache est livré avec un fichier d'hôte virtuel par défaut appelé 000-defaut.conf. Nous allons l'utiliser comme point de départ.

sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/Nomdomaine.local.conf

N'OUBLIER PAS DE REMPLACER Nomdomaine par le vôtre !

Ouvrez le nouveau fichier dans votre éditeur avec les privilèges root :

Sudo nano /etc/apache2/sites-available/Nomdomaine.local.conf/

```
<VirtualHost *:80>
    ServerAdmin admin@Nomdomaine.local
    ServerName Nomdomaine.local
    ServerAlias Indiquer l'IP DE VOTRE SERVEUR
    DocumentRoot /var/www/Nomdomaine.local/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Enregistrez et fermez le fichier lorsque vous avez terminé.

5 – Activez les nouveaux fichiers de l'hôte virtuel.

Taper dans l'invite de commande :

sudo a2ensite Nomdomaine.local.conf

Ensuite, désactivez le site par défaut défini dans 000-defaut.conf

Taper la commande suivante :

```
sudo a2dissite 000-default.conf
```

Lorsque vous avez terminé, vous devez redémarrer apache2 pour que ces changements prennent effet.

Entrer la commande suivante :

```
Sudo systemctl reload apache2
```

```
sudo systemctl restart apache2
```

Vous pouvez vérifier l'état et vérifier qu'apache2 est bien démarré en tapant la commande suivante :

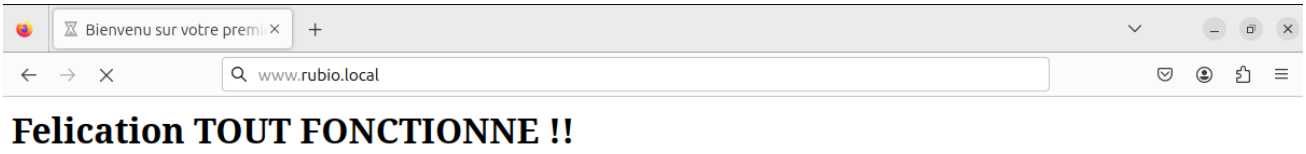
```
sudo systemctl status apache2
```

Si une erreur apparaît pour le démarrage d'apache, vous avez surement une erreur dans la configuration de votre Virtual host.

Vous pouvez maintenant essayer de vous connecter sur votre site en tapant l'adresse ip de votre serveur dans le navigateur web

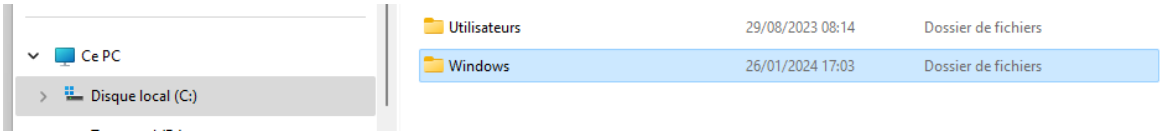
Vous devriez avoir une page de ce type avec les informations que vous avez entrés dans votre site html !

Vous pouvez maintenant essayer avec le nom de domaine que vous avez entrés à savoir www.Nomdomaine.local

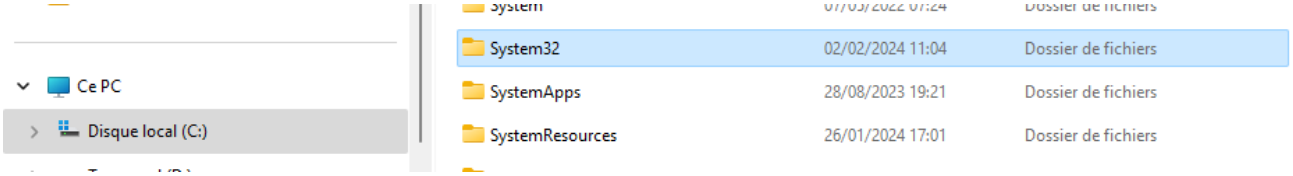


ATTENTION : Il est possible que cela ne fonctionne pas lorsque vous taper votre nom de domaine dans la barre url de votre navigateur. En effet, comme nous l'expliquions plus, Votre site n'est pas renseigné auprès du DNS de google. Il va être donc être nécessaire de modifier le fichier HOST sur votre PC.

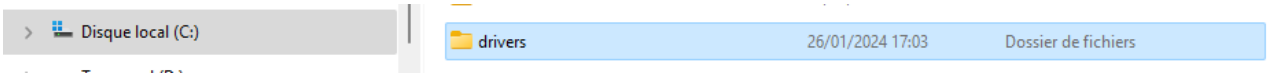
MODIFIER LE FICHIER HOST :



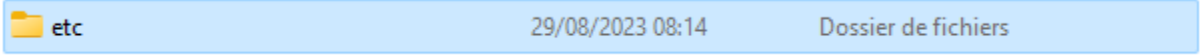
Aller sur c : / puis windows



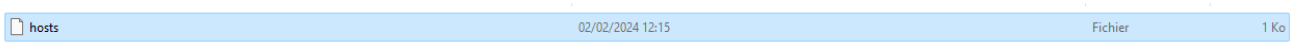
Ensuite entrer dans le dossier system32



Ensuite dans drivers



Ensuite dans etc



Puis copier-coller le fichier HOST sur votre bureau.

Ouvrez le fichier avec un éditeur de texte.

```
192.168.1.18 rubio.local
```

CECI EST UN EXEMPLE ! Remplacez par l'adresse ip de votre serveur ainsi que son Nomdomaine.local

Enregistrez votre fichier.

Supprimer l'extension du fichier .TXT

Pour ce faire faites un clic droit sur le fichier puis renommer le fichier et supprimer manuellement l'extension .TXT et faites ENTRER.

Vous aller maintenant devoir remplacer le fichier HOST dans le dossier ETC (CF CHEMIN précédemment expliqué en photo) et coller le fichier que vous venez de créer. Des droits d'administrateurs vous seront demandés pour pouvoir remplacer ce fichier.

Voilà ! Maintenant vous pouvez essayer de vous connecter sur votre site web avec son nom de domaine !

Configuration du site en HTTPS :

```
<VirtualHost *:443>
    ServerName your_domain_or_ip
    DocumentRoot /var/www/your_domain_or_ip

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
    SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
</VirtualHost>
```

Vous pouvez copier la configuration dans votre fichier virutal HOST.

A cette étape vous devriez être capable de retourner dans le fichier virutal HOST.

Si ce n'est pas le cas, le chemin est le suivant : sudo nano /etc/apache2/sites-available/Domaine_OR_IP.conf

Penser à redémarrer le service apache pour que les changements soient pris en compte !

Sudo systemctl reload apache2

Sudo systemctl restart apache2

Voilà, votre site est déployé en HTTPS ! Notons qu'il est toujours accessible en http !

Afin d'y remédier, nous pouvons effectuer une redirection des requêtes http vers https.

sudo nano /etc/apache2/sites-available/your_domain_or_ip.conf

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

CTLR + O pour enregister les modifications apportées et CTRL + X pour quitter

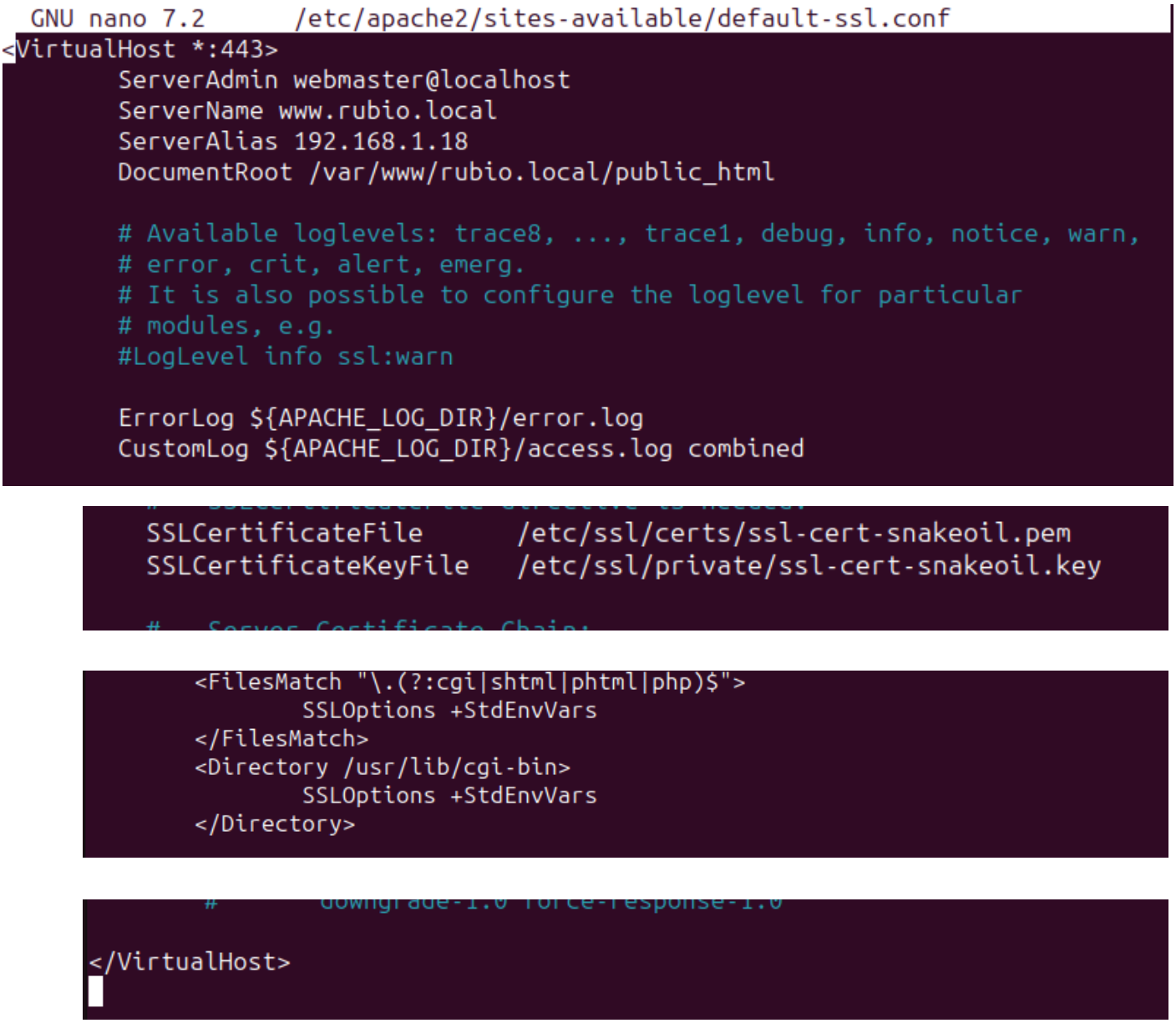
Modification du fichier default-ssl.conf

Entré la commande suivante : sudo nano /etc/apache2/sites-available/default-ssl.conf en adaptant les champs suivants :

ServerName www.Votrenomdedomaine.local

ServerAlias VOTRE IP

DocumentRoot /var/www/Votredomaine.local/public_html



Vous pourrez désormais à coup sur vous connecter en https://votredomaine.local

Siège social · Laval
9 Rue Robert Vauxion
53000 Laval
02 43 69 29 26

Agence Angers
22 Rue Michael Faraday
44070 Beaucouzé
02 41 86 83 01

Agence Le Mans
Rue Georges Auric
72700 Rouillon
02 43 43 00 16

touiller@touiller.fr
www.touiller.fr