Nginx, MariaDB, php, multi-sites dont Wordpress, plusieurs versions de php

- Objet : Un how to pour installer rapidement un serveur nginx avec une base de données (MariaDB), php pluri-versions et Wordpress.
- Niveau requis :
  débutant, avisé
- Commentaires : Un how to pour installer rapidement un serveur nginx avec une base de données (MariaDB), php pluri-versions et Wordpress.
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  à-placer
  - Création par cyrille 23/12/2021
  - Testé par cyrille le 23/12/2021
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Introduction

Un how to pour installer rapidement **un serveur nginx avec une base de données (MariaDB), php pluri-version et Wordpress**. Cette documentation expliquera aussi comment faire **du multi-sites avec nginx** et l'installation de **Wordpress**.

Attention, les configurations proposées le sont dans le cas **d'un serveur de développement**, pas de production. A noter que l'aspect sécurité ne sera ici pas abordé, la finalité de ce guide est de disposer d'un environnement de travail fonctionnel pour ceux qui veulent travailler le développement web dynamique.

A savoir, dans la configuration de nginx (**/etc/nginx/**), il y a deux dossiers importants : **sites-available** et **sites-enabled**.

sites-available: Ce dossier contient les fichiers de configurations de vos sites. Ce dossier est un dépôt ; les fichiers de configuration qui y sont ne sont pas pris en compte. **sites-enabled**: Dossier de liens symboliques vers les fichiers de site-available que vous souhaitez activer.

Pour éditer les fichiers de configuration, il est utilisé ici le paquet **micro** car ses raccourcis clavier sont plus instinctifs que ceux de **nano**, **vi**, **vim**...

Pour l'installer :

```
apt install micro
```

Installation des paquets

Installer les paquets suivants

```
apt install nginx php-cli php-fpm php-mysql php-json php-opcache php-
mbstring php-xml php-gd php-curl mariadb-server
```

Démarrer et activer le démarrage automatique des services suivants

```
systemctl start nginx.service
systemctl enable nginx.service

systemctl start mariadb.service
systemctl enable mariadb.service
```

A cette étape nginx est opérationnel

Sécuriser la base données MariaDB

```
mysq_secure_installation
```

Répondre de la manière suivante

```
Enter current password for root (enter for none):
--> saisie du mot de passe root
Switch to unix_socket authentication [Y/n]
--> n
Change the root password? [Y/n]
--> n
Remove anonymous users? [Y/n]
--> y
Disallow root login remotely? [Y/n]
--> y
Remove test database and access to it? [Y/n]
--> y
Reload privilege tables now? [Y/n]
--> y
```

Si tout se passe bien, vous obtenez ce message :

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

Thanks for using MariaDB!
```

Configurer nginx : créer son premier bloc de serveur

Nginx n'utilise pas des répertoires virtuels (**Virtual Hosts**, terme d'Apache) mais des “**server block**s”, ici traduit en **blocs de serveur**, qui utilisent les directives **server_name** et **listen** pour se lier aux sockets tcp **(2)**.

Créer votre premier répertoire de publication web

```
mkdir -p /var/www/html/site1
```

Adapter les droits Nginx utilise l'user et le groupe **www-data**.

Si vous voulez que votre utilisateur puisse écrire dans ce répertoire, il devra faire partie du groupe www-data et les fichiers et dossiers devront appartenir également à ce groupe

```
chown -R www-data:www-data /var/www/html/site1/
chmod -R 775 /var/www/html/site1/
```

Ajouter votre utilisateur au groupe www-data

```
adduser <utilisateur> www-data
```

(remplacer <utilisateur> par le login du compte qui doit faire parti du groupe www:data

Se déconnecter et se relogguer pour que la modification soit prise en compte Après reconnexion, on vérifie que l'utilisateur est bien dans le groupe www-data

```
groups
ragnarok cdrom floppy audio dip www-data video plugdev netdev bluetooth
lpadmin scanner
```

Configurer votre bloc de serveur

```
micro /etc/nginx/sites-available/site1.conf
```

/etc/nginx/sites-available/site1.conf

```
server {
        listen 80;
        root /var/www/html/site1;
        index  index.php index.html index.htm;
        server_name site1.local;

        error_log /var/log/nginx/site1.local_error.log;
        access_log /var/log/nginx/site1.local_access.log;

        client_max_body_size 100M;
        location / {
                try_files $uri $uri/ /index.php?$args;
```

```
        }
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/run/php/php7.4-fpm.sock;
                fastcgi_param   SCRIPT_FILENAME
$document_root$fastcgi_script_name;
        }
}
```

Attention, pour la ligne

```
                fastcgi_pass unix:/run/php/php7.4-fpm.sock;
```

Bien vérifier que la version php-fpm est la bonne par la commande suivante

```
/var/run/php/
php7.4-fpm.pid  php7.4-fpm.sock  php-fpm.sock
```

Supprimer l'ancienne config de nginx

```
rm /etc/nginx/sites-enabled/default /etc/nginx/sites-available/default
```

Activer le bloc de serveur

```
ln -s /etc/nginx/sites-available/site1.conf /etc/nginx/sites-enabled/
```

Vérifier la bonne configuration de votre premier bloc de serveur

```
nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Recharger la configuration du serveur

```
service nginx reload
```

Créer un fichier php à l'intérieur de votre répertoire de publication web.

Exemple de syntaxe

index.php

```
<?php
echo "<h1>Site 1</h1>";
phpinfo();
```

```
?>
```

Vérifier dans votre navigateur

Le fichier /etc/hosts

Si vous voulez accéder à votre site, via son entrée **server_name** de votre bloc de serveur, alors insérer la ligne suivante dans votre fichier **/etc/hosts**

/etc/hosts

```
127.0.0.1   site1.local

Ce qui donne un fichier de ce style
127.0.0.1   localhost
127.0.1.1   debian11Vbox
127.0.0.1   site1.local



# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Nginx et plusieurs blocs de serveur

1. Copier votre répertoire de publication web

```
cp -R /var/www/html/site1 /var/www/html/site2
```

2. Adapter les droits et permissions

```
chown -R www-data:www-data /var/www/html/site2
chmod -R 775 /var/www/html/site2/
```

2. Adapter le fichier index.php

index.php

```
<?php
echo "<h1>Site 2</h1>";
phpinfo();
?>
```

3. Copier le bloc de serveur site1.conf vers site2.conf et l'adapter

```
cp /etc/nginx/sites-available/site1.conf /etc/nginx/sites-
available/site2.conf
```

Modifier /etc/nginx/sites-available/site2.conf de la manière suivante

/etc/nginx/sites-available/site2.conf

```
server {
        listen 80;
        root /var/www/html/site2;
        index  index.php index.html index.htm;
        server_name site2.local;

        error_log /var/log/nginx/site2.local_error.log;
        access_log /var/log/nginx/site2.local_access.log;

        client_max_body_size 100M;
        location / {
                try_files $uri $uri/ /index.php?$args;
        }
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/run/php/php7.4-fpm.sock;
                fastcgi_param   SCRIPT_FILENAME
$document_root$fastcgi_script_name;
        }
}
```

4. L'activer, tester la configuration de nginx et la recharger

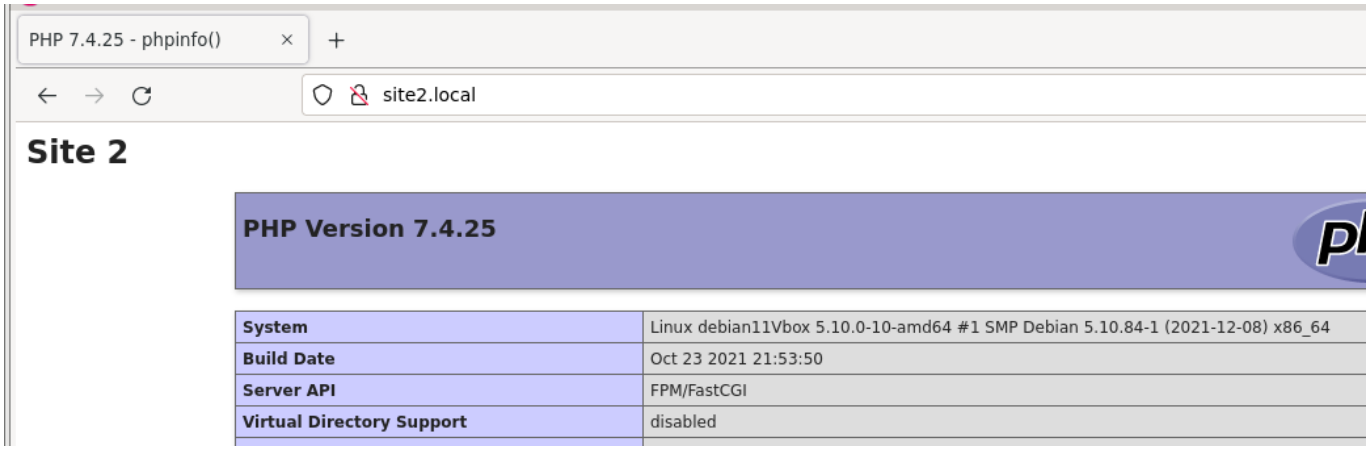
```
ln -s /etc/nginx/sites-available/site2.conf /etc/nginx/sites-enabled/
nginx -t
service nginx reload
```

5. Adpater le fichier /etc/hosts Ajouter la ligne

/etc/hosts

```
127.0.0.1   site2.local
```

6. Tester en saisissant dans votre navigateur site1.local puis site2.local



Installer Wordpress

MariaDB étant installé, il faut configurer la base de données pour Wordpress.

Se connecter au serveur de la base de données, créer une base de données et un utilisateur Wordpress et ajuster les privilèges

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

Saisir les commandes SQL suivantes

```
CREATE DATABASE BDDWPress;
GRANT ALL ON BDDWPress.* TO 'wp-admin'@'localhost' IDENTIFIED BY 'wp-mot-de-passe-1';
quit
```

Récupérer la dernière version de wordpress et l'installer

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

```
wget https://wordpress.org/latest.tar.gz
tar xzfv latest.tar.gz
rm latest.tar.gz
```

```
cd wordpress
mv wp-config-sample.php wp-config.php
```

```
micro wp-config.php
```

Modifier ce fichier en précisant le nom de la base de données, le nom de l'administrateur wordpress précédemment créé et son mot de passe.

wp-config.php

```
// ** MySQL settings - You can get this info from your web host ** //
```

```
/** The name of the database for WordPress */
define( 'DB_NAME', 'BDDWPress' );

/** MySQL database username */
define( 'DB_USER', 'wp-admin' );

/** MySQL database password */
define( 'DB_PASSWORD', 'wp-mot-de-passe-1' );
```

Adapter les droits et les permissions de ce nouveau répertoire de publication web

```
chown -R www-data:www-data /var/www/html/wordpress
chmod -R 755 /var/www/html/wordpress
```

(PS ici les droits sont en 755, le groupe www:data n'ayant pas besoin (sauf usage avancé de WP) d'écrire dans ce répertoire.)

Il ne reste pas qu'à créer le fichier du bloc de serveur de nginx. En voici un exemple :

```
micro /etc/nginx/sites-available/blog.conf
```

/etc/nginx/sites-available/blog.conf

```
server {
        listen 80;
        root /var/www/html/wordpress;
        index  index.php index.html index.htm;
        server_name blog.local ;

        error_log /var/log/nginx/blog.local_error.log;
        access_log /var/log/nginx/blog.local_access.log;

        client_max_body_size 100M;
        location / {
                try_files $uri $uri/ /index.php?$args;
        }
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/run/php/php7.4-fpm.sock;
                fastcgi_param   SCRIPT_FILENAME
$document_root$fastcgi_script_name;
        }
}
```

Tester la configuration de ce fichier, l'activer et recharger nginx.

```
nginx -t
```

```
ln -s /etc/nginx/sites-available/blog.conf /etc/nginx/sites-enabled/
service nginx reload
```

Reste à adapter le fichier /etc/hosts en ajoutant la ligne suivante

```
micro /etc/hosts
```

/etc/hosts

```
Et ajouter la ligne
127.0.0.1   blog.local
```

Faire tourner plusieurs versions de PHP

Installer les paquets suivants

```
apt install -y lsb-release ca-certificates apt-transport-https software-
properties-common gnupg2
```

Ajouter cette entrée aux dépôts APT de la machine. Cette entrée sera identifiée dans un fichier spécifique : sury-php.list

```
echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" | tee
/etc/apt/sources.list.d/sury-php.list
```

Vérifier la bonne création de ce fichier

```
cat /etc/apt/sources.list.d/sury-php.list
deb https://packages.sury.org/php/ bullseye main
```

Importer la clef de contrôle de ce dépôt

```
wget -qO - https://packages.sury.org/php/apt.gpg | apt-key add -
```

Mettre à jour la base de données APT

```
apt update
```

Vérifier que les paquets php8 sont bien présents

```
apt search php
```

Limiter aux paquets php-fpm

```
apt search php8 | grep fpm
```

Installer php8.1 fpm et quelques utilitaires

```
apt install php8.0-{mysql,cli,common,imap,ldap,xml,fpm,curl,mbstring,zip}
```

Vérifier que les 2 versions de php soient bien prises en compte

```
ls /run/php/
php7.4-fpm.pid  php7.4-fpm.sock  php8.1-fpm.pid  php8.1-fpm.sock  php-
fpm.sock
```

Activer une version de php 8.1 dans un bloc de serveur nginx. On utilisera ici le bloc de serveur précédemment créé : site2

/etc/nginx/sites-available/site2.conf

```
micro /etc/nginx/sites-available/site2.conf
```

Et modifier la ligne

```
                fastcgi_pass unix:/run/php/php7.4-fpm.sock;
```

doit devenir

```
                fastcgi_pass unix:/run/php/php8.1-fpm.sock;
```

Sauvegarder

Vérifier la syntaxe de nginx et recharger sa configuration

```
nginx -t
service nginx reload
```

Dans le navigateur internet, vérifier que le site2 renvoie bien la nouvelle configuration.

Dès lors vous avez donc

site1 –> interprète du php 7.4

site2 –> interprète du php 8.1

Conclusions

Voilà donc un premier aperçu de la prise en main de nginx afin d'un serveur web dynamique, pluri-sites et gérant différentes versions de php.

(2) Nginx Server Blocks

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !