tutoriel

LEMP - un serveur avec Linux, Nginx, MariaDB, PHP

LEMP [1] est un serveur HTTP composé de :

- **Nginx - le serveur Web hautes performances (LEMP)**
- **PHP**, ou plutôt le packet **php-fpm**, interface pour communiquer avec le serveur NGINX et PHP.
- **Mariadb : une base de données (fork de MySQL)**
- **Adminer - une interface web pour gérer les BDD SQL**

Pré-requis

Pour le cas d'un Raspberry, procéder en ligne de commande via SSH.

- un **PC** sous Linux (ubuntu) (ou un Raspberry Pi accédé via **SSH**)
- mis à jour :

```
USER@MACHINE:~$ sudo apt update
USER@MACHINE:~$ sudo apt upgrade
USER@MACHINE:~$ sudo apt dist-upgrade
```

Première étape

1. **Installez nginx,php-fpm,php-gd,php-curl,php-intl,php-xmlrpc,php-mbstring,php-mysql,php-xml,php-zip**

   ```
   USER@MACHINE:~$ sudo apt install {nginx php-
   {fpm,gd,curl,intl,xmlrpc,mbstring,mysql,xml,zip}}
   ```

   Pour gérer les fichiers PHP avec Nginx, nous utilisons **php-fpm** (une version plus rapide de PHP) plutôt que PHP. Cette installation a créé le répertoire **/var/www/html** pour la racine des sites.

2. **Vérifiez que NGINX est bien démarré** : sur un PC du réseau, ouvrez dans un navigateur l'adresse IP du serveur http://<AdresseIpDeVotreServeur>. Si tout va bien, il s'affiche :

Vous pouvez aussi vérifier que le service NGINX est bien démarré :

```
USER@MACHINE:~$ sudo systemctl is-active nginx
active
```

3. **Sauvegardez les fichiers de configuration** :

```
USER@MACHINE:~$ sudo cp /etc/nginx/sites-available/default
/etc/nginx/sites-available/default.dist
```

Autres étapes

1. **Déplacer la racine (répertoire de base) d'un serveur HTTP**
2. **Sauvegardez /etc/nginx/sites-available/default** :

```
USER@MACHINE:~$ sudo cp /etc/nginx/sites-available/default
/etc/nginx/sites-available/default.dist
```

3. **Éditez avec les droits d'administration le fichier /etc/nginx/sites-available/default** :
    1. **Remplacez la ligne**

    ```
    index index.html index.htm index.nginx-debian.html;
    ```

    par

    ```
    index index.html index.htm index.php;
    ```

    Cela rajoute une redirection automatique vers les fichiers **index.php** pour les dossiers du site.

    2. **Activez php-fpm** pour Nginx, modifiez les lignes

    ```
    #location ~ \.php$ {
    ```

```
# include snippets/fastcgi-php.conf;
#
# # With php5-cgi alone:
# fastcgi_pass 127.0.0.1:9000;
# # With php5-fpm:
# fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
#}
```

pour obtenir :

```
location ~ \.php$ {
include snippets/fastcgi-php.conf;
fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
}
```

4. **Modifiez les droits du dossier /var/www/html** pour gérer plus facilement les sites :

```
USER@MACHINE:~$ sudo chown -R www-data:pi /var/www/html/
USER@MACHINE:~$ sudo chmod -R 2770 /var/www/html/
```

5. **Ajoutez un fichier index.php** pour vérifier que PHP fonctionne :

```
USER@MACHINE:~$ echo "<?php phpinfo(); ?>" > /var/www/html/index.php
```

6. **Redémarrez Nginx** pour appliquer les modifications :

```
USER@MACHINE:~$ sudo nginx -s reload
```

7. **Vérifiez l'installation de PHP** :
    1. **Version de PHP installée** :

```
USER@MACHINE:~$ php -v
PHP 7.3.19-1~deb10u1 (cli) (built: Jul  5 2020 06:46:45) ( NTS )
...
```

→ Ici, la version de php installée est 7.3

    2. **État de PHP** :

```
USER@MACHINE:~$ sudo systemctl status php7*
● php7.3-fpm.service - The PHP 7.3 FastCGI Process Manager
...
   Active: active (running) since Tue 2020-10-27 14:14:12 CET;
5min ago
...
```

3. **Sur un PC du réseau, ouvrez en http l'adresse IP du serveur** http://<AdresseIpDeVotreServeur>. Si tout va bien, une page affiche les informations de PHP.

8. **Installez Mariadb : une base de données (fork de MySQL)**
9. **Installez VSFTPd - un serveur FTP sécurisé avec des utilisateurs virtuels**
10. **Installez Adminer - une interface web pour gérer les BDD SQL**
11. **Sécurisez Nginx** :
    - **SSL pour Nginx sur Raspberry Pi : mettre en place un certificat SSL auto-signé**
    - **SSL pour Nginx : mettre en place un certificat SSL Let's Encrypt avec Certbot**

Désinstallation

1. **Arrêtez nginx** :

```
USER@MACHINE:~$ sudo systemctl stop nginx.service
```

2. **Démontez /var/www** et **/srv** :

```
USER@MACHINE:~$ sudo umount /var/www
USER@MACHINE:~$ sudo umount /srv
```

3. **Retirez (commentez) le montage de /var/www et /srv** :

```
USER@MACHINE:~$ sudo nano /etc/fstab
```

4. **Supprimez php-fpm** :

```
USER@MACHINE:~$ sudo apt purge php-fpm
```

5. **Supprimez nginx** :

```
USER@MACHINE:~$ sudo apt purge nginx*
```

6. **Terminez l'effacement** :

```
USER@MACHINE:~$ sudo apt autoremove
```

7. **Effacez /var/www/** et ses sous-répertoires :

```
USER@MACHINE:~$ sudo rm -R /var/www/
```

Installation du serveur MariaDB et du client en ligne de commande

1. Installez mariadb-server

```
USER@MACHINE:~$ sudo apt install {mariadb-server,}
```

2. Vérifiez votre installation en lançant le client MariaDB en ligne de commande :

```
USER@MACHINE:~$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 49
Server version: 10.3.25-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and
others.

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.
MariaDB [(none)]> exit
Bye
```

Sortez de la session en tapant **exit**. C'est cet outil que vous utiliserez pour configurer l'instance de base de données pour votre application PHP.

Configuration par défaut de Nginx pour qu'il traite les requêtes PHP

1. Faites une copie du fichier **/etc/nginx/sites-available/default** :

```
USER@MACHINE:~$ sudo cp /etc/nginx/sites-available/default
/etc/nginx/sites-available/default.dist
```

2. Répérez le **socket d'écoute de php-fpm** :

```
USER@MACHINE:~$ ls -l /var/run/php/
total 4
-rw-r--r-- 1 root     root     5 oct.  27 08:15 php7.3-fpm.pid
srw-rw---- 1 www-data www-data 0 oct.  27 08:15 php7.3-fpm.sock
```

→ Ici, il faudra utiliser /var/run/php/php7.3-fpm.sock dans les fichiers de configuration de nginx.

3. **Éditez avec les droits d'administration le fichier etc/nginx/sites-available/default** pour le modifier comme ceci :
    1. remplacez

       etc/nginx/sites-available/default

       ```
       # Add index.php to the list if you are using PHP
       index index.html index.htm index.nginx-debian.html;
       ```

       par

       etc/nginx/sites-available/default

       ```
       # Add index.php to the list if you are using PHP
       index index.html index.htm index.php index.nginx-
       debian.html;
       ```

    2. remplacez

       ```
       # pass PHP scripts to FastCGI server
       #
       #location ~ \.php$ {
       #   include snippets/fastcgi-php.conf;
       #
       #   # With php-fpm (or other unix sockets):
       #   fastcgi_pass unix:/run/php/php7.3-fpm.sock;
       #   # With php-cgi (or other tcp sockets):
       #   fastcgi_pass 127.0.0.1:9000;
       #}
       ```

       par

       ```
       # pass PHP scripts to FastCGI server
       #
           location ~ \.php$ {
               include snippets/fastcgi-php.conf;
               # With php-fpm (or other unix sockets):
               fastcgi_pass unix:/run/php/php7.3-fpm.sock;
           }
       ```

4. Redémarrez Nginx

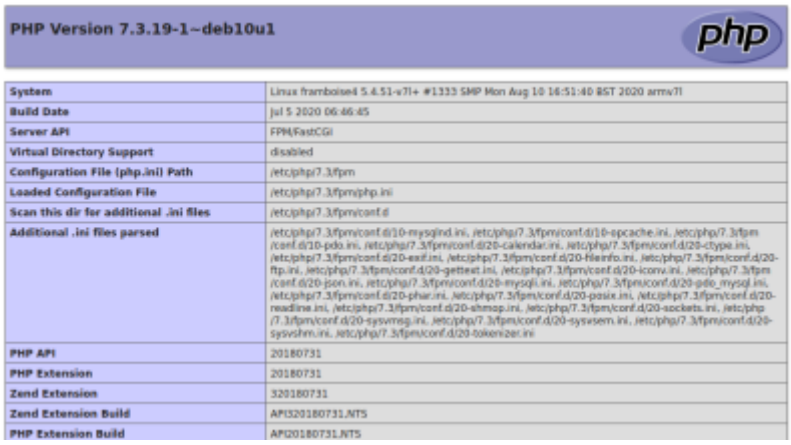
   ```
   USER@MACHINE:~$ sudo systemctl restart nginx
   ```

5. Pour vérifier que Nginx exécute PHP, créez le fichier **/var/www/html/index.php** :

   ```
   <?php
   ```

```
phpinfo();
```

Dans votre navigateur, allez à http://<adresse_ip_de_votre_serveur> :



Configurer Nginx pour qu'il traite les requêtes PHP pour un nom de domaine

Supposons que nous souhaitons répondre aux requêtes PHP pour **abcd1234.com**.

Créez avec les droits d'administration le fichier **/etc/nginx/sites-enabled/abcd1234.com.conf** :

/etc/nginx/sites-enabled/abcd1234.com.conf

```
server {
    listen   80;
    ## Your website name goes here.
    server_name abcd1234.com  www.abcd1234.com;
    root /var/www/abcd1234.com;
    ## This should be in your http block and if it is,
it's not needed here.
    index index.php;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include fastcgi.conf;
        fastcgi_intercept_errors on;
        fastcgi_pass unix:/run/php/php7.3-fpm.sock;

        fastcgi_buffers 16 16k;
        fastcgi_buffer_size 32k;
    }
```

```
}
```

Mise en place de PHP

1. Éditez avec les droits d'administration le fichier **/etc/nginx/sites-available** pour y écrire ceci :

/etc/nginx/sites-available

```
# Site framboise4

server {
    listen 80;

    server_name  framboise4.local;
    root /var/www/html;
    location / {
        index index.php index.html index.htm
index.nginx-debian.html;
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.3-
fpm.sock;
    }

    location ~ /\.ht {
            deny all;
    }
}

server {
    listen 80;

    server_name  ~^(?P<sub>.+)\.framboise4\.local$;
    root /var/www/html/$sub;
    location / {
        index index.php index.html index.htm
index.nginx-debian.html;
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.3-
fpm.sock;
```

```
    }

    location ~ /\.ht {
            deny all;
    }
}
```

2. **Installez les modules de php** (remplacez **php7.3** par la version trouvée ci-dessus) :

```
USER@MACHINE:~$ sudo apt install php7.3-{fpm}
```

1. Pour les installer :

```
USER@MACHINE:~$ sudo apt install php7.4-
{fpm,cli,opcache,mbstring,curl,xml,gd,mysql,common,json,b
cmath,bz2,intl,zip,pdo,imagick,tidy,xmlrpc,dev,imap,soap}
```

🔧 Fix Me! php7.4-apc n'est pas retrouvé

3. **Installez PHP : un langage de programmation libre**
4. **Nginx - le serveur Web hautes performances (LEMP)**
    1. **Installez Nginx sur RaspBerry Pi : le serveur Web hautes performances (LEMP)**
5. **Installez la base de données SQL**, selon le cas :
    1. **MariaDB** : définissez le mot de passe de l'utilisateur root de MariaDb :

```
USER@MACHINE:~$ sudo mysql_secure_installation
```

Au début, répondre ↵ Entrée car il n'y a pas de mot de passe, puis en donner un). MariaDB est installé et prêt à l'emploi.

    2. **SQLite** :
        1. Repérez la version de php installée :

```
USER@MACHINE:~$ php -v
PHP 7.4.3 (cli) (built: May 26 2020 12:24:22) ( NTS )
...
```

        2. et utilisez-la pour installer sqlite :

```
USER@MACHINE:~$ sudo apt install sqlite sqlite-doc
php7.4-sqlite3
...
Souhaitez-vous continuer ? [O/n]
...
```

6. Configurer MariaDB : vérifiez que MariaDb est bien démarré :

```
USER@MACHINE:~$ sudo systemctl is-active nginx
active
```

Démarrer le gestionnaire de processus FastCGI

1. **Affichez la version de php** :

```
USER@MACHINE:~$  php -v
PHP 7.2.19-0ubuntu0.18.04.1 (cli) (built: Jun  4 2019
14:48:12) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend
Technologies
    with Zend OPcache v7.2.19-0ubuntu0.18.04.1,
Copyright (c) 1999-2018, by Zend Technologies
```

2. **Démarrez le gestionnaire de processus FastCGI** (aidez-vous de l'auto-complétion avec la version trouvée ci-dessus) :

```
USER@MACHINE:~$ sudo systemctl enable php7.2-fpm
Synchronizing state of php7.2-fpm.service with SysV
service script with /lib/systemd/systemd-sysv-
install.
Executing: /lib/systemd/systemd-sysv-install enable
php7.2-fpm
```

PHP-FPM est une alternative à PHP FastCGI.

Configurer et démarrer le serveur NGINX

Une fois PHP installé, il faut indiquer à NGINX d'exécuter PHP en utilisant PHP-FPM.

Fichier /etc/nginx/sites-available/default d'origine

/etc/nginx/sites-available/default

```
##
# You should look at the following URL's in
order to grasp a solid understanding
# of Nginx configuration files in order to fully
unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
#
https://www.nginx.com/resources/wiki/start/topic
s/tutorials/config_pitfalls/
#
https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this
file from sites-enabled/ and
# leave it as reference inside of sites-
available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load
configuration files provided by other
# applications, such as Drupal or Wordpress.
These applications will be made
# available underneath a path with that package
name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/
for more detailed examples.
##

# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL
traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure
configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-
cert package
    # Don't use them in a production server!
```

```
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using
PHP
    index index.html index.htm index.nginx-
debian.html;

    server_name _;

    location / {
        # First attempt to serve request as
file, then
        # as directory, then fall back to
displaying a 404.
        try_files $uri $uri/ =404;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #    include snippets/fastcgi-php.conf;
    #
    #    # With php-fpm (or other unix sockets):
    #    fastcgi_pass unix:/var/run/php/php7.0-
fpm.sock;
    #    # With php-cgi (or other tcp sockets):
    #    fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if
Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny all;
    #}
}


# Virtual Host configuration for example.com
#
# You can move that to a different file under
sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
#   listen 80;
```

```
#   listen [::]:80;
#
#   server_name example.com;
#
#   root /var/www/example.com;
#   index index.html;
#
#   location / {
#       try_files $uri $uri/ =404;
#   }
#}
```

1. **Dupliquez le fichier /etc/nginx/sites-available/default** :

```
USER@MACHINE:~$ sudo cp /etc/nginx/sites-
available/default /etc/nginx/sites-
available/default.dist
```

2. **Éditez avec les droits d'administration le fichier /etc/nginx/sites-available/default** pour le remplacer par ceci :

/etc/nginx/sites-available/default

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    root /var/www/html;
    index index.php index.html index.htm
index.nginx-debian.html;

    server_name _;
    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass
unix:/var/run/php/php7.2-fpm.sock;
    }
    location ~ /\.ht {
        deny all;
    }
}
```

3. Testez la configuration :

```
USER@MACHINE:~$ sudo nginx -t
```

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

```
USER@MACHINE:~$ sudo nginx -s stop
USER@MACHINE:~$ sudo nginx
```

4. Créez un fichier /var/www/html/test/test.php dans le répertoire web racine de NGINX :

   /var/www/html/test/test.php

   ```
   <?php phpinfo()?>
   ```

5. Vérification : http://localhost/test

Serveurs virtuels

Créez avec les droits d'administration le fichier **/etc/nginx/sites-available/monsite.tld** pour y écrire :

/etc/nginx/sites-available/monsite.tld

```
server {
        listen 80;
        root /var/www/html;
        index index.php index.html index.htm
index.nginx-debian.html;
        server_name monsite.tld;

        location / {
                try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
                include snippets/fastcgi-
php.conf;
                fastcgi_pass
unix:/var/run/php/php7.2-fpm.sock;
        }

        location ~ /\.ht {
                deny all;
        }
```

```
}
```

Explications

Voici ce que font ces directives et ces blocs location :

1. **listen** : Port sur lequel Nginx écoute. 80 = port par défaut pour HTTP
2. **root** : emplacement de la racine du site Web.
3. **index** : fichiers index.php prioritaires
4. **server_name** : nom de domaine ou adresse IP de votre serveur.
5. **location /** : La directive try_files vérifie l'existence de fichiers demandés. Si Nginx ne le trouve pas, il retourne une erreur 404.
6. **location ~ \.php$** : gère le traitement PHP en pointant Nginx vers le fichier de configuration fastcgi-php.conf et le fichier php7.2-fpm.sock, qui indique le type de socket associé à php-fpm..
7. **location ~ /\.ht** : bloc location pour les fichiers .htaccess, que Nginx ne traite pas. La directive deny all ne transmet pas aux visiteurs des éventuels fichiers .htaccess

Activez votre site en créant un lien symbolique dans le dossier **/etc/nginx/sites-enable** :

```
USER@MACHINE:~$ sudo ln -s /etc/nginx/sites-available/monsite.tld /etc/nginx/sites-enabled/monsite.tld
```

Redémarrez nginx :

```
USER@MACHINE:~$ sudo nginx -s reload
```

Configuration d'hôtes virtuels sur NGinx avec support automatique

des sous-domaines, du SSL et de l'authentification

Configurer Nginx pour gérer automatiquement les sous-domaines locaux

1. **Création d'un nouveau site nginx** :
    1. **Éditez avec les droits d'administration le fichier /etc/nginx/sites-available/monsite.local** pour y écrire :

    /etc/nginx/sites-available/monsite.local

```
server {
    # On écoute le port 80.
    listen 80;

    # expression régulière pour
récupérer
    # le sous-domaine dans une variable
nommée "sub".
    server_name
~^(?P<sub>.+)\.monsite\.local$;

    location / {
         # On définit le chemin local
         # en utilisant la variable
"sub" récupérée précédemment.
         root /var/www/html/$sub;
    }
}
```

    2. **Activez votre site** en créant un lien symbolique dans le dossier **/etc/nginx/sites-enable** :

```
USER@MACHINE:~$ sudo ln -s /etc/nginx/sites-
available/local.dev /etc/nginx/sites-
enable/monsite.local
```

    3. **Redémarrez le serveur** :

```
USER@MACHINE:~$ sudo service nginx restart
```

    4. /etc/nginx/sites-available/chateau.parc

```
# Le nom du sous-domaine est mis dans
une variable sub,
```

```
#   utilisée ensuite pour définir la
racine

# Sites dokuwiki
server {
    listen 80;

    server_name
~^(?P<sub>doc|perso|site2)\.chateau\.pa
rc$;
    root /var/www/html/$sub;
    location / {
        index index.php index.html
index.htm;
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-
php.conf;
        fastcgi_pass
unix:/var/run/php/php7.2-fpm.sock;
    }

    location ~ /(data|conf|bin|inc)/ {
        deny all;
    }

    location ~ /\.ht {
            deny all;
    }
}

# Autres sous-domaines
server {
    listen 80;

    server_name
~^(?P<sub>.+)\.chateau\.parc$;
    root /var/www/html/$sub;
    location / {
        index index.php index.html
index.htm index.nginx-debian.html;
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-
php.conf;
        fastcgi_pass
unix:/var/run/php/php7.2-fpm.sock;
```

```
    }

    location ~ /\.ht {
            deny all;
    }
}
```

Création des domaines génériques avec dnsmasq

1. Installez dnsmasq

```
USER@MACHINE:~$ sudo apt install {dnsmasq,}
```

2. Éditez avec les droits d'administration le fichier **/etc/dnsmasq.d/local.conf** et ajoutez-lui la ligne :

   /etc/dnsmasq.d/local.conf

```
address=/mondomaine.tld/127.0.0.1
```

   Dans notre exemple, nous avons donc ajouté :

   /etc/dnsmasq.d/local.conf

```
address=/monsite.local/127.0.0.1
```

3. redémarrez dnsmasq :

```
USER@MACHINE:~$ sudo systemctl restart dnsmasq
```

   : nos domaines sont accessibles.

Désormais, tous les domaines ***.mondomaine.tld** existent.

Vous pouvez le vérifier :

1. **Installez dnsutils**

```
USER@MACHINE:~$ sudo apt install {dnsutils,}
```

2. Lancez :

```
USER@MACHINE:~$ dig test.localhost

; <<>> DiG 9.11.3-1ubuntu1.7-Ubuntu <<>>
test.localhost
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR,
id: 47480
;; flags: qr rd ra; QUERY: 1, ANSWER: 1,
AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;test.localhost.           IN    A

;; ANSWER SECTION:
test.localhost.        0    IN    A    127.0.0.1

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sat May 04 23:35:51 CEST 2019
;; MSG SIZE  rcvd: 59
```

Conclusion

Problèmes connus

- Erreur 403 forbidden, un classique de Nginx

Erreur 403 forbidden, un classique de Nginx

L'erreur 403 est fréquente avec Nginx, mais simple à corriger.

1. Cela peut signifier que Nginx ne trouve pas de fichier à afficher. Pour tester cela, créez un fichier index.html dans le répertoire **/var/www** :

```
echo 'Nginx marche !' >
/var/www/html/index.html
```

et ré-essayez d'accéder à votre site. Si le problème venait de là, vous devriez obtenir une page marquée **Nginx marche !**

2. Sinon, c'est peut-etre que Nginx ne peut accéder au répertoire **/var/www/html**. Dans ce cas, donne au répertoire **/var/www** les autorisations suffisantes.
3. Dernière possibilité : vous avez mal activé php.

Voir aussi

- **(fr)** Créer un serveur Web Nginx + PHP7 + Maria DB (Mysql) + PhpMyAdmin sous Debian 9 Stretch
- **(fr)** https://raspberry-pi.fr/installer-nginx-raspbian-raspberry/
- **(en)** How to setup a Raspberry Pi LEMP server with Raspbian Buster Lite for running PHP applications
- **(en)** https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-on-ubuntu-14-04

Basé sur « Créer un serveur Web Nginx + PHP7 + Maria DB (Mysql) + PhpMyAdmin sous Debian 9 Stretch » par Quentin Beauvais.

1)

Linux, NGINX (**e**ngine X), **M**ySQL/MariaDB, **P**HP/**P**erl/**P**ython

From:
http://doc.frapp.fr/ - **doc**

Permanent link:
http://doc.frapp.fr/doku.php?id=tutoriel:reseau:web:serveur:lemp:start

Last update: **2023/05/15 22:13**