tutoriel

De Apache à Nginx : remplacer Apache par Nginx sans perdre les données du site Web et avec le minimum de temps d'arrêt

Dans cet article, nous illustrerons la migration d'Apache vers Nginx avec l'application PHP Drupal.

Pour rendre l'exemple plus réaliste, nous utiliserons le CMS Drupal comme modèle d'application PHP.

En effet, Drupal 7, entièrement stockée dans la racine du document, est un bon exemple d'application PHP dépendant de la configuration du serveur HTTP.

Vous trouverez ici des idées précieuses même pour d'autres applications PHP.

C'est aussi une analyse détaillée de Drupal sur Apache.

Ce tutoriel explique comment migrer un site Web de **Apache** vers **Nginx** sur un VPS Ubuntu 12.04.

Il suppose que vous avez installé un serveur **LAMP** (Linux, Apache, MySQL et PHP).

L'idée est d'installer Nginx sur le port 8080 et en parallèle avec Apache qui reste accessible sur le port 80, pour pouvoir configurer le nouveau serveur Nginx tout en consultant les fichiers de configuration de Apache.

Principales différences entre Apache et Nginx :

	Apache	NGINX
emplacement des fichiers de configuration	/etc/apache2/apache2.conf	/etc/nginx/nginx.conf
fonctionnement	VirtualHost	bloc server
répertoire racine par défaut	/var/www/html	/var/www/html
système de cache intégré	non	oui

Pré-requis

- Un serveur **LAMP** installé

Première étape : Installer Nginx

Nous installons **nginx** sur le port **8080** (ou **8008**), pour pouvoir configurer le nouveau serveur en consultant les anciens fichiers de configuration Apache.

1. **Mettez à jour le système** :

```
...@...:~ $ sudo apt clean all && sudo apt update && sudo apt dist-upgrade
```

2. **Installez nginx** :

```
...@...:~$ sudo apt install nginx
```

```
Un message d'erreur signale qu'il ne peut pas démarrer ; c'est normal car le port 80 est déjà utilisé par Apache, nous résoudrons ce problème plus loin.
```

3. **Installez php-fpm** (implémentation FastCGI de PHP qui est compatible nginx) :

```
...@...:~$ sudo apt install php-fpm
```

4. **Configuration provisoire de nginx** pour qu'il fonctionne comme le faisait Apache [1] :
    1. **sauvegardez le fichier du site par défaut** :

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

/etc/nginx/sites-available/default.dist

```
##
# 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/topics/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:/run/php/php7.3-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;
# }
#}
```

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

1. **listen** : remplacez le port d'écoute 80 par le port **8080** ou **8008** [2] :

   /etc/nginx/sites-available/default

```
...
server {
#    listen 80 default_server;
#    listen [::]:80 default_server;
    listen 8080 default_server;
    listen [::]:8080 default_server;
...
```

2. **root** : même racine qu'apache
3. **index** : ajoutez index.php
4. **juste après index, ajoutez** :

   /etc/nginx/sites-available/default

```
location ~ \.php$ {
    try_files $uri =404;
    fastcgi_pass unix:/run/php/php7.3-fpm.sock;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
    include fastcgi_params;
}
```

5. **Démarrez Nginx** :

```
...@...:~ $ sudo systemctl start nginx
```

6. **Créez un vhost pour votre site**, par exemple :

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

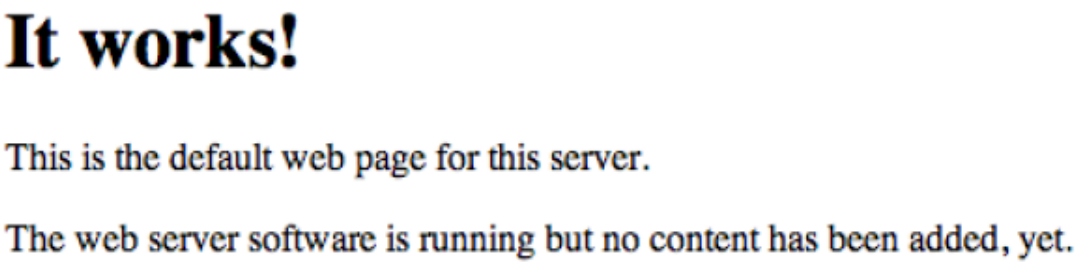
```
server {
    listen 80;
    server_name monsite.tld;
    location '/.well-known/acme-challenge' {
        default_type "text/plain";
        root /tmp/letsencrypt-auto/;
    }
    location / {
        return 301 https://domain.tld$request_uri;
    }
}
```

Autres étapes

Vérification

1. **Nginx fonctionne sur le port 8080 (ou 8008)** : visitez votre site avec le port 8080 (ou 8008) http://monsite.tld:8080

2. **Apache fonctionne toujours sur le port par défaut (80)** : visitez votre site sans numéro de port http://monsite.tld ; votre site s'affiche comme d'habitude, ou affiche la page d'Apache si le site est encore vierge



Transcription de la configuration Apache en configuration Nginx

Les deux serveurs étant maintenant opérationnels, nous allons traduire votre configuration Apache pour une utilisation avec Nginx.

Pour cela, il faut, manuellement, écrire des blocs **server** pour Nginx équivalents aux VirtualHosts d'Apache. Pour chaque déclaration de VirtualHost, vous créerez un bloc server.

Sous Ubuntu,

- Apache conserve ses VirtualHosts dans **/etc/apache2/sites-available/**
- Nginx conserve ses déclarations de bloc **server** dans **/etc/nginx/sites-available/**.

Les VirtualHosts d'Apache ont la forme suivante [3] :

/etc/apache2/sites-available/monsite.tld.conf

```
<VirtualHost *:80>
    ServerAdmin webmaster@monsite.tld
    ServerName monsite.tld
    ServerAlias www.monsite.tld

    DocumentRoot /var/www/html/

    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>

    <Directory /var/www/html/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Require all granted
    </Directory>

    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews
+SymLinksIfOwnerMatch
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Require all denied
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>
</VirtualHost>
```

1. **Éditez avec les droits d'administration le fichier /etc/nginx/sites-available/default** (précédemment édité pour déclarer le port 8080 ou 8008) pour le modifier comme ceci [4] :

/etc/nginx/sites-available/default

```
server {
    listen 8080;

    root /usr/share/nginx/www;
    index index.html index.htm;
```

```
    server_name localhost;

    location / {
        try_files $uri $uri/ /index.html;
    }

    location /doc/ {
        alias /usr/share/doc/;
        autoindex on;
        allow 127.0.0.1;
        deny all;
    }
}
```

Transcription des principales directives entre Apache et Nginx

Apache	Nginx
<VirtualHost *:80>	server { listen 80;
ServerName yoursite.com ServerAlias www.yoursite.com	server_name yoursite.com www.yoursite.com;
DocumentRoot /path/to/root	root /path/to/root;
AllowOverride All	(Aucune alternative disponible)
DirectoryIndex index.php	index index.php;
ErrorLog /path/to/log	error_log /path/to/log error;
CustomLog /path/to/log combined	access_log /path/to/log main;
Alias /url/ “/path/to/files” <Directory “/path/to/files”>	location /url/ { alias /path/to/files;
</VirtualHost>	}

2. La transcription du fichier de VirtualHost ci-dessus en bloc server pourrait ressembler à ceci :

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

```
server {
    listen 8080;   # Nous laissons cela tel quel
pour éviter tout conflit pour le moment

    root /var/www/html;
    server_name monsite.tld www.monsite.tld;

    location / {
        try_files $uri $uri/ /index.html;
    }

    location ~ \.php$ {
```

```
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass unix:/var/run/php5-fpm.sock;
        fastcgi_index index.php;
        include fastcgi_params;
    }

    location /doc/ {
        alias /usr/share/doc/;
        autoindex on;
        allow 127.0.0.1;
        deny all;
    }

    location ~/\.ht {
        deny all;
    }

}
```

Apache	Nginx
<VirtualHost *:80>	server { listen 8080; # Nous laissons cela tel quel pour éviter tout conflit pour le moment
ServerAdmin webmaster@monsite.tld	
ServerName monsite.tld ServerAlias www.monsite.tld	server_name monsite.tld www.monsite.tld;
DocumentRoot /var/www/html/	root /var/www/html;
<Directory /> Options FollowSymLinks AllowOverride None </Directory>	

	location / {
	try_files $uri $uri/ /index.html;
	}

<Directory /var/www/html/>	
Options Indexes FollowSymLinks MultiViews	
AllowOverride None	
Require all granted	
</Directory>	
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/	
<Directory “/usr/lib/cgi-bin”>	
AllowOverride None	
Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch	
Require all granted	
</Directory>	

Apache	Nginx
ErrorLog ${APACHE_LOG_DIR}/error.log	
LogLevel warn	
CustomLog ${APACHE_LOG_DIR}/access.log combined	
Alias /doc/ “/usr/share/doc/“ <Directory ”/usr/share/doc/”>	location /doc/ { alias /usr/share/doc/;
Options Indexes MultiViews FollowSymLinks AllowOverride None Require all denied Allow from 127.0.0.0/255.0.0.0 ::1/128	autoindex on; allow 127.0.0.1; deny all;
</Directory>	}
</VirtualHost>	}

Nous avons éliminé certains éléments et ajouté quelques lignes supplémentaires que nous devrions expliquer.

1. **Les lignes du journal des erreurs ont été supprimées** car ils sont déjà définis dans le fichier /etc/nginx/nginx.conf qui fournit une valeur par défaut que nous utiliserons.
2. **La directive ServerAdmin a été supprimée** car Nginx n'intègre pas ces informations dans ses pages d'erreur.
3. **La gestion de PHP a également un peu changé**. Étant donné que PHP est géré séparément dans Nginx, nous transmettons ces fichiers au programme php-fpm que nous avons installé précédemment. Ceci est implémenté via un socket (que nous devrons configurer momentanément).

La section documentation est modifiée pour refléter la documentation Nginx. Elle fonctionne par ailleurs de manière assez similaire.

Enfin, nous configurons Nginx pour refuser l'accès à tout fichier .htaccess ou autres fichiers commençant par .ht dans notre répertoire. Ce sont des fichiers de configuration spécifiques à Apache et ils ne fonctionnent pas avec Nginx. Il est plus sûr de ne pas exposer ces fichiers de configuration.

Basculement de Apache vers Nginx quasiment sans coupure

1. **Effectuez le basculement** quasiment sans coupure entre Apache et Nginx :

1. **Remettez le port d'écoute sur 80** (placé tout à l'heure à 8080)
2. **Arrêtez Apache** :

```
...@...:~ $ sudo systemctl stop apache2
```

3. **Redémarrez Nginx** (ce qui applique la modification de port) :

```
...@...:~ $ sudo systemctl restart nginx
```

4. **Désinstallez Apache** :

```
...@...:~ $ sudo apt remove apache2
```

ou en effaçant la configuration d'Apache :

```
...@...:~ $ sudo apt purge apache2
```

Transcrivez votre configuration Apache

Enregistrez et fermez le fichier lorsque vous avez terminé.

Nous devons redémarrer notre serveur Nginx pour que ces modifications soient reconnues :

```
...@...:~ $ sudo service nginx restart
```

Configurer PHP-FPM

Maintenant que nous avons préparé la plupart de la configuration Nginx, nous devons modifier la configuration php-fpm pour communiquer en utilisant les canaux que nous avons spécifiés.

Pour commencer, nous devons modifier le fichier php.ini afin qu'il ne serve pas les fichiers de manière non sécurisée.

Éditez avec les droits d'administration le fichier **/etc/php5/fpm/php.ini** pour modifier la directive **cgi.fix_pathinfo** comme ceci :

/etc/php5/fpm/php.ini

```
cgi.fix_pathinfo=0
```

Cela demandera à PHP de servir le fichier exact demandé au lieu de deviner s'il y a

une correspondance incomplète, ce qui empêche PHP de servir ou d'exposer des données sensibles à quelqu'un qui sonde les faiblesses du gestionnaire PHP.

Ensuite, nous allons changer la façon dont php-fpm se connecte à notre serveur.

Éditez avec les droits d'administration le fichier **/etc/php5/fpm/pool.d/www.conf** pour faire correspondre la directive listen à la valeur mise dans le fichier de configuration du bloc server :

/etc/php5/fpm/pool.d/www.conf

```
listen = /var/run/php5-fpm.sock
```

Si vous rencontrez des problèmes avec la gestion d'un grand nombre de requêtes PHP, vous voudrez peut-être revenir ici et augmenter le nombre de processus enfants pouvant être générés en même temps. La ligne à modifier est :

```
pm.max_children = Num_of_children
```

Maintenant, notre programme php-fpm devrait être configuré correctement. Nous devons le redémarrer pour que nos modifications se propagent :

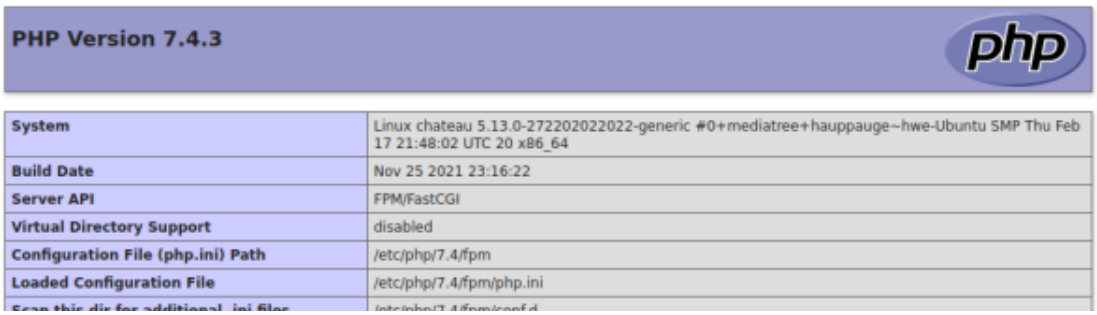
```
...@...:~ $ sudo service php5-fpm restart
```

Redémarrer Nginx ne fera pas de mal :

```
...@...:~ $ sudo service nginx restart
```

Vérifiez que tous les fichiers PHP que vous avez dans votre répertoire racine fonctionnent correctement. Vous devriez pouvoir exécuter les fichiers PHP comme dans Apache.

Si nous accédons au fichier info.php que nous avons créé dans le tutoriel Ubuntu LAMP, il devrait s'afficher comme ceci :http://your_ip_or_domain:8000/info.php



Dans la section Variables PHP, vous devriez voir Nginx répertorié comme la variable « SERVER_SOFTWARE » :

PHP Variables

Variable	Value
$_SERVER['USER']	www-data
$_SERVER['HOME']	/var/www
$_SERVER['HTTP_UPGRADE_INSECURE_REQUESTS']	1
$_SERVER['HTTP_CONNECTION']	keep-alive
$_SERVER['HTTP_ACCEPT_ENCODING']	gzip, deflate
$_SERVER['HTTP_ACCEPT_LANGUAGE']	fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
$_SERVER['HTTP_ACCEPT']	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
$_SERVER['HTTP_USER_AGENT']	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:97.0) Gecko/20100101 Firefox/97.0
$_SERVER['HTTP_HOST']	php.
$_SERVER['REDIRECT_STATUS']	200
$_SERVER['SERVER_NAME']	~^(?P<sub>.+
$_SERVER['SERVER_PORT']	80
$_SERVER['SERVER_ADDR']	192.168.0.1
$_SERVER['REMOTE_PORT']	48126
$_SERVER['REMOTE_ADDR']	192.168.0.1
$_SERVER['SERVER_SOFTWARE']	nginx/1.18.0
$_SERVER['GATEWAY_INTERFACE']	CGI/1.1
$_SERVER['REQUEST_SCHEME']	http
$_SERVER['SERVER_PROTOCOL']	HTTP/1.1

Transition à chaud de votre site Nginx

Après avoir effectué des tests approfondis, vous pouvez essayer de passer de manière transparente votre site d'Apache à Nginx.

Cela est possible car aucun de ces serveurs n'implémente de modifications tant qu'ils ne sont pas redémarrés. Cela nous permet de tout configurer, puis de basculer l'interrupteur en un instant.

La seule chose à faire est de modifier le port dans le bloc serveur Nginx.

Remettez le port par défaut à 80 pour accepter le trafic HTTP normal dès qu'il sera redémarré. Éditez avec les droits d'administration le fichier **/etc/nginx/sites-available/default** pour le modifier comme ceci :

/etc/nginx/sites-available/default

```
server {
    # listen 8000;
    listen 80;
    . . .
```

Si vous ne transférez que certains de vos sites vers Nginx et que vous continuez à diffuser du contenu à partir d'Apache, vous devez désactiver les VirtualHosts Apache qui traitent les requêtes sur le port 80.

C'est indispensable pour éviter les conflits.

Si cela n'est pas fait correctement, Nginx ne démarrera pas car le port sera déjà pris.

Si vous prévoyez de continuer à exécuter Apache, vérifiez ces fichiers et emplacements pour l'utilisation du port 80 :

- /etc/apache2/ports.conf
- /etc/apache2/apache2.conf
- /etc/apache2/httpd.conf
- /etc/apache2/sites-enabled/ ## Search all sites in this directory

Une fois satisfait des modifications de tous les ports nécessaires, vous pouvez redémarrer les deux services comme ceci :

```
...@...:~ $ sudo service apache2 reload && sudo service nginx
reload
```

Apache devrait se recharger, libérant le port 80. Immédiatement après, Nginx devrait se recharger et commencer à accepter les connexions sur ce port. Si tout s'est bien passé, votre site devrait maintenant être servi par Nginx.

Si vous n'utilisez plus Apache pour servir une partie de vos sites, vous pouvez à la place arrêter complètement son processus Web :

```
...@...:~ $ sudo service apache2 stop && sudo service nginx
reload
```

Si vous n'utilisez plus Apache, vous pouvez désinstaller les fichiers Apache à ce stade. Vous pouvez facilement trouver les fichiers liés à Apache en tapant :

```
...@...:~ $ dpkg --get-selections | grep apache
apache2                         install
apache2-bin                     install
apache2-data                    install
apache2-utils                   install
libapache-poi-java              install
libapache-pom-java              install
libapache2-mod-php              install
libapache2-mod-php7.3               install
python-certbot-apache               install
python3-certbot-apache              install
```

Vous pouvez ensuite les désinstaller avec apt. Par exemple:

```
...@...:~ $ sudo apt remove {apache2,apache2-
{bin,data,utils},libapache-{poi-java,pom-java},libapache2-mod-
{php,php7.3},python{-certbot-apache,3-certbot-apache}}
```

Vous pouvez également supprimer tous les packages de dépendance qui ne sont plus nécessaires :

```
...@...:~ $ sudo apt autoremove
```

Complications liées à la migration

Certaines choses courantes dans le monde Apache peuvent vous prêter à confusion lorsque vous essayez de passer à Nginx.

Réécrire les traductions et les fichiers .htaccess

One of the most fundamental differences is that Nginx does not respect directory overrides.

Apache utilise des fichiers **.htaccess**, ainsi qu'une directive **AllowOverride All** dans un bloc **location**.

Cela vous permet de placer des configurations spécifiques à un répertoire dans le répertoire qui héberge les fichiers.

Nginx n'autorise pas ces fichiers. L'hébergement de la configuration avec les fichiers servis est potentiellement un problème de sécurité s'il est mal configuré, et il est facile de regarder les fichiers de configuration centralisés et de ne pas se rendre compte qu'un paramètre est écrasé via un fichier .htaccess.

Par conséquent, toutes les configurations que vous avez répertoriées dans un fichier .htaccess actif doivent être placées dans le bloc d'emplacement dans la configuration du bloc de serveur pour cet hôte. Ce n'est généralement pas plus compliqué, mais vous devez traduire ces règles comme vous l'avez fait avec les définitions de VirtualHosts.

Une chose courante à conserver dans les fichiers .htaccess sont les règles du module mod_rewrite d'Apache, qui modifie les URL d'accès au contenu pour les rendre plus conviviales. Nginx a un module de réécriture similaire, mais utilise une syntaxe différente. Malheureusement, la réécriture des URL dans Nginx n'entre pas dans le cadre de ce guide.

Module and Outside Configuration Complications

Conclusion

Problèmes connus

Voir aussi

- **(fr)** https://lelibreauquotidien.fr/2020/01/28/remplacer-apache-par-nginx-sans-coupure-sur-debian/ (lien mort)
- **(en)**

https://www.digitalocean.com/community/tutorials/how-to-migrate-from-an-apache-web-server-to-nginx-on-an-ubuntu-vps

- **(en)** https://www.airpair.com/nginx/posts/ultimate-guide-migrating-apache-to-nginx-1
- **(en)** https://www.airpair.com/nginx/posts/ultimate-guide-migrating-apache-to-nginx-2

Basé sur « How To Migrate from an Apache Web Server to Nginx on an Ubuntu VPS » par digitalocean.

1)

Pour faire fonctionner les deux serveurs simultanément, nous allons tester Nginx sur un autre port en gardant Apache en exécution. Ainsi, notre site sera toujours opérationnel pendant la transition.

2)

les lignes d'origine ont été commentées

3)

notez le .conf désormais nécessaire

4)

sans les commentaires