Logiciel

Nginx - le serveur Web hautes performances (LEMP)

Nginx (« engine X ») est un serveur Web, similaire à Apache, qui peut être utilisé :

- comme serveur web autonome
- ou comme proxy pour réduire la charge de serveurs HTTP ou de courrier électronique en arrière-plan.

Le paquet a trois versions :

- **nginx-full** (par défaut), avec l'ensemble de modules standard sauf ceux qui sont dans nginx-extra
- **nginx-light**
- et **nginx-extras**.

Fonctionnement de **nginx** : un processus maître gère plusieurs processus de travail qui font le traitement réel des demandes.

Le fichier de configuration est **/etc/nginx/nginx.conf**.

Voir le guide du débutant : **Nginx - Guide du débutant**

Pré-requis

Installation

1. **Installez nginx**

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

L'installation a créé les arborescences :

```
...@...:~$ tree /var -d
/var
...
└── www
    └── html
```

et

```
...@...:~$ tree /etc/nginx -d
/etc/nginx
├── conf.d
├── modules-available
├── modules-enabled
├── sites-available
├── sites-enabled
└── snippets
```

2. **Testez l'installation** (deux possibilités) :
   1. en testant l'état du service :

```
...@...:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse
proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled;
vendor preset: en
   Active: active (running) since Mon 2020-10-26 16:22:02 CET;
14min ago
<...>Le statut **active** indique que le service fonctionne
correctement.
```

   2. ou en ouvrant **http://localhost** ou http://IP_du_serveur (par ex. http://framboise.local), il s'affiche :

3. **Déplacer la racine (répertoire de base) d'un serveur HTTP**

Installez le paquet **php-xml** ou en ligne de commande :

```
$ sudo apt install php-xml
```

L'utilisateur **$USER** (qui fait partie du groupe **www-data**) a lui aussi accès à ce répertoire.

N'oubliez pas de recharger la page du navigateur pour vider le cache, sinon c'est l'ancienne page qui s'affiche.

Configuration

Le fichier de configuration de **nginx** est **/etc/nginx/nginx.conf**.

Nous n'y toucherons pas pour ne pas perdre les réglages lors des mises à jour.

Au lieu de toucher au fichier /etc/nginx/nginx.conf, nous utiliserons certains dossiers :

- Pour la **configuration**, des fichiers dans le répertoire **/etc/nginx/conf.d**.
- Pour les **hôtes virtuels**, des fichiers dans le répertoire **/etc/nginx/sites-available** (le fichier default peut servir de modèle)
- Le dossier **/etc/nginx/sites-enabled** permet de lancer les hôtes virtuels en production

Pour en savoir plus sur le fichier de configuration, voir Structure du fichier de configuration nginx.conf

Changer le port de fonctionnement de Nginx

Nginx fonctionne par défaut sur le port 80. Pour changer ce port, par exemple en 8080

1. **Éditez avec les droits d'administration le fichier /etc/nginx/sites-enabled/default** [1] pour définir le port souhaité (ici, 8080) :

   /etc/nginx/sites-enabled/default

   ```
   server {
       listen 8080;
   }
   ```

2. **Démarrez le serveur** :

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

3. Vous pouvez maintenant accéder à votre site sur le port 8080 (http://monsite.tld:8080).

Configuration de Nginx en serveur Web

La configuration de Nginx en serveur Web définit les URL qu'il gère et comment il traite les requêtes HTTP pour ces URL.

Gestion d'erreurs

La directive **error_page** demande à Nginx de renvoyer une page personnalisée ou un autre URI pour un code d'erreur donné.

Par exemple, pour le code 404 :

```
error_page 404 /404.html;
```

Autre exemple : si Nginx ne trouve pas une page, il remplace le code 401 par le code 301 et redirige le client vers http:/example.com/new/path.html.

Cette configuration est utile lorsque les clients tentent toujours d'accéder à une page par son ancien URI.

Le code 301 informe le navigateur que la page a été déplacée de façon permanente, et il doit remplacer automatiquement l'ancienne adresse par la nouvelle au retour.

```
location /old/path.html {
    error_page 404 =301 http:/example.com/new/path.html;
}
```

Voici un exemple de transmission d'une requête au back end lorsqu'un fichier est introuvable.

Comme aucun code d'état n'est spécifié après le signe égal dans la directive **error_page**, la réponse au client a le code d'état renvoyé par le serveur proxy (pas nécessairement 404).

1. **error_page** indique à Nginx d'effectuer une redirection interne lorsqu'un fichier n'est pas trouvé.
2. La variable $uri dans le paramètre final de la directive **error_page** contient l'URI de la requête en cours, qui est transmise dans la redirection.
3. Par exemple, si /images/some/file n'est pas trouvé, il est remplacé par /fetch/images/some/fil et une nouvelle recherche d'emplacement commence.
   Par conséquent, la requête se retrouve dans le deuxième contexte d'emplacement et est envoyée par proxy à http: http://backend/.
4. **open_file_cache_errors** empêche d'écrire un message d'erreur si un fichier n'est pas trouvé.
   Ce n'est pas nécessaire ici car les fichiers manquants sont correctement gérés.

/etc/nginx/conf.d/

```
server {
    ...
```

```
    location /images/ {
        # Set the root directory to search for the file
        root /data/www;

        # Disable logging of errors related to file existence
        open_file_cache_errors off;

        # Make an internal redirect if the file is not found
        error_page 404 = /fetch$uri;
    }

    location /fetch/ {
        proxy_pass http://backend/;
    }
}
```

Serveur de contenu statique

Le fichier de configuration peut inclure plusieurs blocs server se distinguant par les ports sur lesquels ils écoutent et par les noms de serveurs.

location

Une fois que nginx décide quel serveur traite une requête, il teste l'URI spécifié dans l'en-tête de la requête par rapport aux paramètres des directives location définies dans le bloc server.

Nous allons implémenter un exemple où les fichiers seront servis à partir des répertoires locaux: **/data/www** (qui peut contenir des fichiers HTML) et **/data/images** (contenant des images).

1. **Créez le répertoire /data/www/html** et placez-y un fichier **index.html** avec un contenu textuel quelconque
2. **Créez le répertoire /data/images** et placez-y quelques images.
3. **Créez un fichier maconf.conf** et écrivez un nouveau bloc server :

   /etc/nginx/conf.d/maconf.conf

```
    server {
        location / {
            root /data/www;
        }

        location /images/ {
            root /data;
        }
    }
```

Ce serveur écoute sur le port standard 80 et est accessible sur la machine locale à l'adresse http://localhost/.

Par exemple, en réponse à la requête http://localhost/some/example.html, nginx renverra le fichier /data/www/some/example.html.

1. Le bloc **location /** spécifie la racine du site. Pour les requêtes qui correspondent, l'URI sera ajouté au chemin spécifié dans la directive root, c'est-à-dire **/data/www**, pour former le chemin vers le fichier demandé sur le système de fichiers local. Si plusieurs blocs location correspondent, nginx sélectionne celui avec le préfixe le plus long. Le bloc location ci-dessus fournit le préfixe le plus court, de longueur un, et donc, seulement si tous les autres blocs d'emplacement ne parviennent pas à correspondre, c'est ce bloc qui sera utilisé.
2. Pour le second bloc **location /images/**, il y aura correspondance pour les requêtes commençant par /images/. En réponse à des requêtes dont les URI commencent par /images/, le serveur renverra des fichiers du répertoire /data/images. Par exemple, en réponse à la requête http://localhost/images/example.png, nginx renverra le fichier **/data/images/example.png**. Si ce fichier n'existe pas, nginx renverra l'erreur 404. Les demandes avec des URI ne commençant pas par /images/ seront mappées dans le répertoire /data/www.

Pour appliquer la nouvelle configuration,

1. si nginx n'est pas encore démarré, lancez-le :

```
$ sudo nginx
```

2. ou envoyez le signal reload :

```
$ sudo nginx -s reload
```

En cas d'erreur, recherchez la cause dans les fichiers /var/log/nginx/access.log et /var/log/nginx/error.log

root

La directive **root** spécifie le répertoire racine qui sera utilisé pour rechercher un fichier. Pour obtenir le chemin d'un fichier demandé, Nginx ajoute l'URI de la requête au chemin spécifié par la directive root.

Elle peut être placée à n'importe quel niveau dans les contextes **http**, **server** ou **location**.

Dans l'exemple ci-dessous, la directive **root** est définie pour un serveur virtuel.

Elle s'applique à tous les blocs location où la directive root n'est pas incluse pour redéfinir explicitement la racine :

```
server {
    root /www/data;

    location / {
    }

    location /images/ {
    }

    location ~ \.(mp3|mp4) {
        root /www/media;
    }
}
```

Ici, pour un URI qui commence par **/images/**, Nginx recherche dans le répertoire **/www/data/images/** du système de fichiers.

Mais si l'URI se termine par l'extension **.mp3** ou **.mp4**, Nginx recherche plutôt le fichier dans le répertoire /www/media/ comme défini dans le bloc d'emplacement correspondant.

Si une requête se termine par une barre oblique, Nginx la traite comme une demande de répertoire et essaie de trouver un fichier d'index dans le répertoire.

index

La directive **index** définit le nom du fichier d'index (par défaut index.html).

Pour continuer avec l'exemple, si l'URI de la requête est **/images/some/path/**,

1. Nginx délivre le fichier /www/data/images/some/path/index.html s'il existe.
2. Si ce n'est pas le cas, Nginx renvoie le code HTTP 404 (non trouvé) par défaut.

Pour configurer Nginx pour qu'il renvoie une liste de répertoires générée automatiquement, incluez le paramètre **on** dans la directive **autoindex** :

```
location /images/ {
    autoindex on;
}
```

Vous pouvez lister plus d'un nom de fichier dans la directive index. Nginx recherche les fichiers dans l'ordre spécifié et renvoie le premier qu'il trouve :

```
location / {
```

```
    index index.htm index.html;
}
```

Pour retourner le fichier d'index, Nginx vérifie son existence puis effectue une redirection interne vers l'URI obtenue en ajoutant le nom du fichier d'index à l'URI de base.

La redirection interne entraîne une nouvelle recherche d'un emplacement et peut aboutir à un autre emplacement, comme dans l'exemple suivant :

```
location / {
    root /data;
    index index.html index.php;
}

location ~ \.php {
    fastcgi_pass localhost:8000;
    ...
}
```

Si l'URI d'une requête est /path/, et que /data/path/index.html n'existe pas mais que /data/path/index.php existe, la redirection interne vers /path/index.php est mappée au deuxième emplacement.

Par conséquent, la demande est envoyée par proxy.

Essayer plusieurs options

La directive **try_files** vérifie si le fichier ou le répertoire spécifié existe ; Nginx effectue une redirection interne si c'est le cas, ou renvoie un code d'état si ce n'est pas le cas.

Par exemple, pour vérifier l'existence d'un fichier correspondant à l'URI de la requête, utilisez la directive **try_files** et la variable $uri comme suit :

```
server {
    root /www/data;

    location /images/ {
        try_files $uri /images/default.gif;
    }
}
```

Le fichier est spécifié sous forme de l'URI, qui est traité à l'aide des directives **root** ou **alias** définies dans le contexte de l'emplacement actuel ou du serveur virtuel.

Dans ce cas, si le fichier correspondant à l'URI d'origine n'existe pas, Nginx effectue une redirection interne vers l'URI spécifié par le dernier paramètre, en renvoyant /www/data/images/default.gif.

Le dernier paramètre peut également être un code d'état (directement précédé du signe égal) ou le nom d'un emplacement.

Dans l'exemple suivant, une erreur 404 est renvoyée si aucun des paramètres de la directive **try_files** ne se résout en un fichier ou un répertoire existant.

```
location / {
    try_files $uri $uri/ $uri.html =404;
}
```

Dans l'exemple suivant, si ni l'URI d'origine ni l'URI avec la barre oblique finale ajoutée ne se résolvent en un fichier ou répertoire existant, la requête est redirigée vers l'emplacement nommé qui le transmet à un serveur proxy.

```
location / {
    try_files $uri $uri/ @backend;
}

location @backend {
    proxy_pass http://backend.example.com;
}
```

Pour plus d'informations, visionnez le webinaire à la demande Content Caching pour apprendre à améliorer considérablement les performances d'un site Web et approfondir les capacités de mise en cache de Nginx.

Configuration d'un serveur proxy simple

Une utilisation fréquente de nginx consiste à le configurer en serveur proxy : un serveur reçoit les demandes, les transmet aux serveurs proxy, récupère les réponses et les envoie aux clients.

Nous allons configurer un serveur proxy de base, qui sert les demandes d'images avec des fichiers du répertoire local et envoie toutes les autres demandes à un serveur proxy.

Dans cet exemple, les deux serveurs seront définis sur une seule instance de nginx.

Tout d'abord, définissez le serveur proxy en ajoutant un bloc server supplémentaire au fichier de configuration de nginx :

```
    server {
        listen 8080;
        root /data/up1;

        location / {
        }
    }
```

Ce sera un serveur simple qui écoute sur le port 8080 (précédemment, la directive listen n'a pas été spécifiée puisque le port 80 standard était utilisé) et mappe toutes les demandes vers le répertoire /data/up1 sur le système de fichiers local.

Créez ce répertoire et placez-y le fichier index.html.

Notez que la directive root est placée dans le contexte server.

Cette directive root est utilisée lorsque le bloc location sélectionné pour traiter une requête n'a pas sa propre directive racine.

Ensuite, utilisez la configuration du serveur de la section précédente et modifiez-la pour en faire une configuration de serveur proxy.

Dans le premier bloc location, mettez la directive proxy_pass en spécifiant le protocole, le nom et le port du serveur proxy dans le paramètre (dans notre cas, c'est http://localhost:8080) :

```
server {
    location / {
        proxy_pass http://localhost:8080;
    }

    location /images/ {
        root /data;
    }
}
```

Nous allons modifier le second bloc location, qui pour l'instant mappe les demandes avec le préfixe /images/ vers les fichiers du répertoire /data/images, pour qu'il corresponde aux demandes d'images avec des extensions de fichier typiques. Le bloc location modifié ressemble à ceci :

```
location ~ \.(gif|jpg|png)$ {
    root /data/images;
}
```

Le paramètre est une expression régulière qui correspond à tous les URI se terminant par .gif, .jpg ou .png.

Une expression régulière devrait être précédée par ~.

Les requêtes correspondantes seront mappées dans le répertoire /data/images.

Quand nginx choisit un bloc location pour servir une demande, il vérifie d'abord des directives location qui spécifient des préfixes, en se souvenant de l'endroit avec le plus long préfixe et vérifie ensuite les expressions régulières.

S'il y a correspondance avec une expression régulière, nginx choisit cet endroit ; sinon, il choisit celui retenu plus tôt.

La configuration résultante d'un serveur proxy ressemblera à ceci :

```
server {
    location / {
        proxy_pass http://localhost:8080/;
    }
```

```
        location ~ \.(gif|jpg|png)$ {
            root /data/images;
        }
    }
```

Ce serveur filtre les requêtes se terminant par .gif, .jpg, ou .png et les mappe dans le répertoire /data/images (en ajoutant l'URI au paramètre de la directive root) et transmet toutes les autres requêtes au serveur mandaté configuré ci-dessus.

Pour appliquer la nouvelle configuration, envoyez le signal de rechargement à nginx comme décrit plus haut.

De nombreuses autres directives peuvent être utilisées pour configurer davantage une connexion proxy.

Configuration de la fonction proxy de FastCGI

nginx peut être utilisé pour router des requêtes vers des serveurs FastCGI qui exécutent des applications construites avec différents frameworks et langages de programmation tels que PHP.

La configuration nginx la plus basique pour travailler avec un serveur FastCGI comprend l'utilisation de la directive fastcgi_pass au lieu de la directive proxy_pass, et des directives fastcgi_param pour définir les paramètres transmis à un serveur FastCGI.

Supposons que le serveur FastCGI soit accessible sur localhost:9000.

En prenant pour base la configuration du proxy de la section précédente, remplacez la directive proxy_pass par la directive fastcgi_pass et remplacez le paramètre par localhost:9000.

En PHP, le paramètre SCRIPT_FILENAME est utilisé pour déterminer le nom du script et le paramètre QUERY_STRING est utilisé pour transmettre les paramètres de la requête.

La configuration résultante serait :

```
    server {
        location / {
            fastcgi_pass  localhost:9000;
            fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
            fastcgi_param QUERY_STRING    $query_string;
        }

        location ~ \.(gif|jpg|png)$ {
            root /data/images;
        }
    }
```

Cela va mettre en place un serveur qui acheminera toutes les demandes à l'exception des demandes d'images statiques au serveur proxy opérant sur localhost:9000 via le protocole FastCGI.

Serveurs Virtuels

Voir **Nginx : Comment configurer des hôtes virtuels sur Ubuntu 16.04**

Les blocs server de Nginx permettent d'héberger plusieurs sites sur un seul serveur.

Nous allons utiliser le domaine example.com.

Par défaut, Nginx traite toujours le contenu du répertoire /var/www/html/.

Cette configuration ne convient que si vous hébergez un seul site. Ici, nous allons créer un autre répertoire pour servir le domaine exemple.com.

Nous laisserons le répertoire /var/www/html pour le cas où aucun domaine de la configuration nginx ne correspond.

Créez le répertoire :

```
$ sudo mkdir -p /var/www/example.com/html
```

Le -p crée le ou les répertoires parents si nécessaire.

Après cela, nous assignerons le propriétaire du répertoire en utilisant la variable $USER :

```
$ sudo chown -R $USER:$USER /var/www/example.com/html
```

Définissez les autorisations pour la racine Web :

```
$ sudo chmod -R 755 /var/www/example.com
```

Ensuite, vous devez créer la page index.html ; éditez avec les droits d'administration le fichier **/var/www/example.com/html/index.html** pour y écrire :

/var/www/example.com/html/index.html

```
<html>
    <head>
        <title>Welcome to Example.com!</title>
    </head>
    <body>
        <h1>Success!  The example.com server block is
working!</h1>
    </body>
</html>

Pour que le contenu ci-dessus soit servi, nous devons créer un
bloc server Nginx dans /etc/nginx/sites-available/example.com.
```

```
éditez avec les droits d'administration le fichier
**/etc/nginx/sites-available/example.com** pour y écrire
:<code nginx /etc/nginx/sites-available/example.com>
server {
        listen 80;
        listen [::]:80;

        root /var/www/example.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name example.com www.example.com;

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

Activez en créant le lien :

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

Nous avons configuré deux blocs pour les requêtes du serveur:

1. **example.com** qui servira le contenu pour exemple.com et www.example.com
2. **default** qui servira le contenu de la demande sur le port 80.

Nous devons faire une petite modification dans le fichier de configuration Nginx pour éviter un problème de mémoire. le problème se pose lorsque vous ajoutez plusieurs noms de serveur. Éditez avec les droits d'administration le fichier **/etc/nginx/nginx.conf** pour dé-commenter la ligne :

```
...
http {
    ...
    server_names_hash_bucket_size 64;
    ...
}
```

Une fois terminées les étapes ci-dessus, testez la syntaxe du fichier Nginx :

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

Puis redémarrez le serveur Nginx :

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

si vous allez à votre adresse IP sur votre navigateur, vous devriez voir «Succès! Le bloc de

serveur example.com fonctionne !»

Sécurisation avec OpenSSL

- **SSL pour Nginx : mettre en place un certificat SSL auto-signé**
  - **SSL pour Nginx sur Raspberry Pi : mettre en place un certificat SSL auto-signé**

2. **SSL pour Nginx : mettre en place un certificat SSL Let's Encrypt avec Certbot**

Utilisation

Commandes de gestion de Nginx

Le serveur est en cours d'exécution.

1. **Pour l'arrêter** :

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

2. **Pour le démarrer** :

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

3. **Pour le redémarrer** :

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

4. **Si vous avez apporté des modifications au serveur** :

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

ne coupera pas les connexions existantes et ne fera que mettre à jour les modifications

5. **Pour empêcher Nginx de démarrer avec le système** :

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

6. **Pour démarrer Nginx avec le système** :

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

Autre méthode : démarrer, arrêter et recharger la configuration

1. **arrêt rapide** :

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

2. **arrêt propre** (à exécuter sous l' utilisateur qui a démarré nginx), arrête nginx en attendant la fin des processus en cours :

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

3. **recharger le fichier de configuration** :

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

4. **rouvrir les fichiers journaux** :

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

On peut aussi tuer les processus nginx avec l'utilitaire kill.

Pour obtenir la liste de tous les processus nginx en cours d'exécution, vous pouvez lancer :

```
USER@MACHINE:~$ sudo ps -ax | grep nginx
```

Désinstallation

Comme d'habitude,

```
USER@MACHINE:~$ sudo apt remove nginx
```

Problèmes connus

- Si on installe Nginx et qu'on essaie de le démarrer, on obtient le message suivant :

Si on installe Nginx et qu'on essaie de le démarrer, on obtient le message suivant :

```
USER@MACHINE:~$ sudo nginx

Starting nginx: the configuration file /etc/nginx/nginx.conf
```

```
syntax is ok
configuration file /etc/nginx/nginx.conf test is successful
[emerg]: bind() to 0.0.0.0:80 failed (98: Address already in
use)
```

C'est logique car Apache utilise le port 80.

Il faut changer le port de fonctionnement de Nginx, par exemple en 8080 :

1. Éditez avec les droits d'administration le fichier **/etc/nginx/sites-enabled/default** [2] pour définir le port souhaité (ici, 8080) :

   /etc/nginx/sites-enabled/default

   ```
   server {
       listen 8080;
   }
   ```

2. Démarrez le serveur :

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

3. Vous pouvez maintenant accéder à votre site sur le port 8080 (http://monsite.tld:8080).

Voir aussi

- **(en)** https://nginx.org/en/docs/beginners_guide.html
- **(en)** documentation :https://docs.nginx.com/nginx/admin-guide/
- **(fr)** Créer un serveur Web Nginx + PHP7 + Maria DB (Mysql) + PhpMyAdmin sous Debian 9 Stretch
- **(fr)** https://prograide.com/pregunta/32900/comment-demarrer-nginx-via-un-autre-port-autre-que-80

Basé sur « Article » par Auteur.

[1], [2]
c'est la configuration par défaut

From:
https://doc.wikis.frapp.fr/ - **doc**

Permanent link:
https://doc.wikis.frapp.fr/doku.php?id=logiciel:reseau:http:serveur:nginx:start

Last update: **2024/09/21 10:43**