tutoriel

Nginx : Comment configurer des hôtes virtuels sur Ubuntu 16.04

Les blocs **server** (similaires aux hôtes virtuels dans Apache) peuvent encapsuler les détails de la configuration et héberger plusieurs domaines sur un seul serveur.

Les serveurs virtuels contrôlent le traitement des demandes pour des domaines ou des adresses IP particuliers.

Chaque serveur virtuel définit des emplacements («**location**») qui contrôlent le traitement d'ensembles spécifiques d'URI.

Chaque **location** peut transmettre la demande par proxy ou renvoyer un fichier.

L'URI peut être modifié, en redirigeant la demande vers un autre emplacement ou un serveur virtuel.

Un code d'erreur spécifique peut être renvoyé et vous pouvez configurer une page spécifique pour chaque code d'erreur.

Dans chacun des contextes de gestion du trafic, vous incluez un ou plusieurs blocs **server** pour définir les serveurs virtuels qui contrôlent le traitement des demandes.

Les directives que vous pouvez inclure dans un contexte **server** varient en fonction du type de flux :

- Pour le flux HTTP (contexte **http**), chaque directive **server** contrôle le traitement des demandes de ressources sur des domaines ou des adresses IP particuliers.
- Dans un contexte **server**, un ou plusieurs contextes **location** définissent comment traiter des groupes spécifiques d'URI.
- Pour le flux mail et TCP/UDP (contextes **mail** et **stream**) les directives **server** contrôlent chacune le traitement du trafic arrivant sur un port TCP ou un socket UNIX particulier.

Vous pouvez partir du fichier exemple fourni :

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

On peut ajouter plusieurs directives **server** pour définir plusieurs serveurs virtuels.

La directive **listen** spécifie l'adresse IP:port sur lesquels le serveur écoute les demandes.

Les adresses IPv4 et IPv6 sont acceptées. Placez les adresses IPv6 entre crochets.

Voici un exemple de configuration d'un serveur qui écoute l'adresse IP 127.0.0.1 sur le port 8080 :

/etc/nginx/sites-available/monserveur

```
server {
```

```
    listen 127.0.0.1:8080;
    # Le reste de la configuration du serveur
}
```

- Si le port est omis, c'est le port standard qui est utilisé.
- Si l'adresse est omise, le serveur écoute toutes les adresses.
- Sans directive listen, le port standard est 80/tcp et le port par défaut est 8000/tcp, selon les privilèges du super-utilisateur.
- Si plusieurs serveurs correspondent à l'IP:port de la requête, Nginx teste les directives **server_name**. **server_name** peut être un nom complet, un nom avec un joker (une astérisque qui correspond à n'importe quelle séquence de caractères au début, à la fin ou les deux) ou une expression régulière (selon la syntaxe Perl ; faites-les précéder du tilde (~)).

Voici un exemple avec un nom exact :

```
server {
    listen      80;
    server_name example.org www.example.org;
    ...
}
```

Si plusieurs noms correspondent, Nginx sélectionne dans l'ordre suivant la première correspondance trouvée :

1. Nom exact
2. Le joker le plus long commençant par un astérisque, tel que ***.example.org**
3. Le joker le plus long se terminant par un astérisque, tel que **mail.***
4. Première expression régulière correspondante (par ordre d'apparition dans le fichier de configuration)

Si aucun nom ne correspond, Nginx prend le au serveur par défaut pour le port sur lequel la requête est arrivée.

Le serveur par défaut est le premier répertorié dans le fichier **nginx.conf**, sauf si vous incluez le paramètre **default_server** dans la directive **listen** pour désigner explicitement un serveur comme serveur par défaut :

```
server {
    listen      80 default_server;
```

```
    ...
}
```

Pré-requis

- **un utilisateur avec les privilèges sudo** tout au long de ce tutoriel.
- **Nginx** installé sur votre serveur. Voir en particulier :
  - How To Install Nginx on Ubuntu 16.04 : pour configurer Nginx lui-même.
  - How To Install Linux, Nginx, MySQL, PHP (LEMP stack) in Ubuntu 16.04: si vous utilisez Nginx avec MySQL et PHP.

Exemple de configuration

Dans ce document, nous allons configurer deux domaines avec notre serveur Nginx. Nous les appellerons **example.com** et **test.com**.

Première étape : Configurer de nouveaux répertoires racine de document

Par défaut, Nginx a un bloc **server** activé par défaut, configuré pour servir des documents à partir du répertoire **/var/www/html**.

Bien que cela fonctionne bien pour un seul site, pour desservir plusieurs sites, il faut des répertoires supplémentaires.

Le répertoire /var/www/html peut être vu comme le répertoire par défaut, servi si la demande du client ne correspond à aucun de nos autres sites.

Nous allons créer une structure de répertoires dans /var/www pour chacun de nos sites.

Le contenu Web sera placé dans un répertoire spécifique.

Cela nous donne plus de flexibilité pour créer d'autres répertoires associés à nos sites dans le répertoire html si nécessaire.

Nous devons créer ces répertoires pour chacun de nos sites (L'option -p indique à mkdir de créer les répertoires parents nécessaires en cours de route) :

```
USER@MACHINE:~$ sudo mkdir -p /var/www/example.com
USER@MACHINE:~$ sudo mkdir -p /var/www/test.com
```

Maintenant que nous avons nos répertoires, nous allons rendre notre compte d'utilisateur normal propriétaire de ces répertoires, ce qui nous permettra d'y écrire sans sudo.

Selon vos besoins, vous devrez peut-être ajuster les

autorisations pour autoriser l'accès à l'utilisateur **www-data**.

Nous utilisons la variable d'environnement $USER pour attribuer la propriété au compte sur lequel nous sommes actuellement connectés. Cela nous permettra de créer ou d'éditer facilement le contenu de ce répertoire :

```
USER@MACHINE:~$ sudo chown -R $USER:$USER /var/www/example.com
USER@MACHINE:~$ sudo chown -R $USER:$USER /var/www/test.com
```

Les autorisations de nos racines Web devraient déjà être correctes si vous n'avez pas modifié votre valeur umask, mais nous pouvons nous en assurer en saisissant :

```
USER@MACHINE:~$ sudo chmod -R 755 /var/www
```

Notre structure de répertoire est maintenant configurée et nous pouvons continuer.

Autres étapes

Créer des pages exemples pour chaque site

Maintenant que notre structure de répertoires est en place, créons une page par défaut pour chacun de nos sites pour avoir quelque chose à afficher.

Créez avec les droits d'administration le fichier **/var/www/example.com/index.html** :

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

```
<html>
    <head>
        <title>Bienvenue sur Example.com !</title>
    </head>
    <body>
        <h1>Succès ! Le serveur example.com fonctionne !</h1>
    </body>
</html>
```

Comme le fichier de notre deuxième site est semblable, copiez-le sur notre deuxième racine de document comme ceci :

```
USER@MACHINE:~$ sudo cp /var/www/example.com/html/index.html
/var/www/test.com/html/
```

Éditez avec les droits d'administration le fichier **/var/www/test.com/html/index.html** pour le modifier comme ceci :

/var/www/test.com/html/index.html

```
<html>
    <head>
        <title>Bienvenue sur Test.com !</title>
    </head>
    <body>
        <h1>Succès ! Le serveur test.com fonctionne !</h1>
    </body>
</html>
```

Nous avons maintenant quelques pages à afficher aux visiteurs de nos deux domaines.

Créer des blocs server pour chaque domaine

Maintenant que nous avons le contenu à diffuser, nous devons créer les blocs **server** qui indiqueront à Nginx comment procéder.

Par défaut, Nginx contient un bloc **server** nommé default que nous pouvons utiliser comme modèle pour nos propres configurations.

Nous allons commencer par le bloc **server** de notre premier domaine, que nous copierons ensuite pour notre deuxième domaine et modifierons.

Créer le premier fichier bloc server

Comme mentionné ci-dessus, nous allons créer notre premier fichier de configuration de bloc server en copiant le fichier default :

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

Éditez avec les droits d'administration le fichier **/etc/nginx/sites-available/example.com** pour le modifier comme ceci : (en ignorant les lignes commentées) :

/etc/nginx/sites-available/example.com

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

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

    server_name _;

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

```
    }
}
```

Examinons d'abord les directives **listen**.

Sur le serveur, l'option default_server ne doit être activée que sur un seul bloc server.

Ceci spécifie quel bloc doit servir une demande si le nom de serveur demandé ne correspond à aucun des blocs server disponibles.

Cela ne devrait pas arriver très souvent dans des scénarios réels, car les visiteurs accéderont à votre site par votre nom de domaine.

Vous pouvez choisir de désigner l'un de vos sites comme **default** en incluant l'option **default_server** dans la directive listen, ou vous pouvez laisser activé le bloc server par défaut, qui servira le contenu du répertoire /var/www/html si l'hôte demandé ne peut pas être trouvé.

Dans ce guide, nous allons laisser le bloc server par défaut en place pour traiter les demandes qui ne correspondant pas.

Nous allons donc supprimer le serveur default_server de ce bloc serveur et du prochain serveur.

Vous pouvez choisir d'ajouter cette option à l'un des blocs de votre serveur qui vous convient.

/etc/nginx/sites-available/example.com

```
server {
    listen 80;
    listen [::]:80;

    . . .
}
```

Vous pouvez vérifier que l'option default_server n'est activée que dans un seul fichier actif en tapant :

```
USER@MACHINE:~$ grep -R default_server
/etc/nginx/sites-enabled/
```

Si des correspondances non commentées sont trouvées dans plus d'un fichier (indiqué dans la colonne la plus à gauche), Nginx signalera une configuration invalide.

La prochaine chose que nous devrons ajuster est la racine du document, spécifiée par la directive root.

Faites-la pointer sur la racine du site que vous avez créée :

/etc/nginx/sites-available/example.com

```
server {
    listen 80;
    listen [::]:80;

    root /var/www/example.com;
}
```

Ensuite, nous devons modifier le nom du serveur pour qu'il corresponde aux demandes de notre premier domaine.

Nous pouvons en outre ajouter tous les alias que nous voulons faire correspondre.

Nous allons ajouter un alias www.example.com pour démonstration.

Lorsque vous avez terminé, votre fichier ressemblera à ceci :

/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;
    }
}
```

C'est tout ce dont nous avons besoin pour une configuration de base.

Créer le deuxième fichier bloc server

Maintenant que nous avons notre configuration de bloc server initiale, nous pouvons l'utiliser comme base pour notre deuxième fichier.

Recopiez-le pour créer un nouveau fichier :

```
USER@MACHINE:~$ sudo cp /etc/nginx/sites-available/example.com
/etc/nginx/sites-available/test.com
```

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

Encore une fois, assurez-vous de ne pas utiliser l'option default_server pour la directive listen de ce fichier si vous l'avez déjà utilisée ailleurs.

Ajustez la directive root pour qu'elle pointe vers la racine du deuxième domaine et ajustez server_name pour qu'il corresponde au nom de domaine de votre deuxième site (assurez-vous d'inclure tous les alias).

Lorsque vous aurez terminé, votre fichier ressemblera probablement à ceci :

/etc/nginx/sites-available/test.com

```
server {
    listen 80;
    listen [::]:80;

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

    server_name test.com www.test.com;

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

Activation des blocs servers et redémarrage de Nginx

Maintenant que nous avons nos fichiers de blocs server, nous devons les activer.

Pour les activer, créez des liens symboliques en tapant :

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

Ces fichiers sont maintenant dans le répertoire activé.

Nous avons maintenant trois blocs server activés, configurés pour répondre en fonction de leur port d'écoute et du nom du serveur.

1. **example.com** : Répondra aux demandes de example.com et www.example.com
2. **test.com** : Répondra aux demandes de test.com et www.test.com
3. **default** : Répondra à toutes les demandes sur le port 80 qui ne correspondent pas aux deux autres blocs.

Pour éviter tout problème de mémoire pouvant résulter de l'ajout de noms de serveurs, nous allons ajuster une valeur unique dans notre fichier /etc/nginx/nginx.conf.

Éditez avec les droits d'administration le fichier **/etc/nginx/nginx.conf** pour le modifier comme ceci :

1. Dans le fichier, recherchez la directive server_names_hash_bucket_size.
2. Supprimez le symbole # pour supprimer la mise en commentaire de la ligne :

/etc/nginx/nginx.conf

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

Ensuite, testez pour vous assurer qu'il n'y a pas d'erreur de syntaxe dans vos fichiers Nginx :

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

Si aucun problème n'a été détecté, redémarrez Nginx pour activer vos modifications :

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

Nginx devrait maintenant servir vos deux noms de domaine.

Modifier votre fichier Hosts local pour le test (facultatif)

Si vous n'avez pas utilisé de noms de domaine que vous possédez mais plutôt des valeurs factices, vous pouvez modifier la configuration de votre ordinateur local pour vous permettre de tester temporairement la configuration des blocs server de Nginx.

Cela ne permettra pas aux autres visiteurs de voir votre site correctement, mais cela vous permettra d'accéder à chaque site indépendamment et de tester votre configuration.

Cela fonctionne essentiellement en interceptant des demandes qui iraient généralement au DNS pour résoudre les noms de domaine.

Au lieu de cela, nous pouvons définir les adresses IP auxquelles nous voulons que notre ordinateur local accède lorsque nous demandons les noms de domaine.

Assurez-vous que vous opérez sur votre ordinateur local
au cours de ces étapes et non sur votre serveur VPS.

Vous devez avoir un accès root, être membre du groupe

d'administration ou pouvoir modifier les fichiers système
à cette fin.

Éditez avec les droits d'administration le fichier **/etc/hosts** pour le modifier comme ceci :

Vous devez connaître l'adresse IP publique de votre serveur et les domaines que vous souhaitez router vers le serveur.

En supposant que l'adresse IP publique de mon serveur soit 203.0.113.5, les lignes que j'ajouterais à mon fichier ressembleraient à ceci :

/etc/hosts

```
127.0.0.1   localhost
. . .
203.0.113.5 example.com www.example.com
203.0.113.5 test.com www.test.com
```

Cela interceptera toutes les requêtes de example.com et test.com et les enverra à votre serveur, ce que nous souhaitons si nous ne possédons pas les domaines que nous utilisons.

Testez vos résultats

- Ouvrez http://example.com ; vous devriez voir une page qui ressemble à ceci :

  Success! The example.com server block is working!

- En ouvrant http://test.com, vous devriez voir une page qui ressemble à ceci :

  Success! The test.com server block is working!

Si ces deux sites fonctionnent, vous avez correctement configuré deux blocs server indépendants avec Nginx.

À ce stade, si vous avez ajusté votre fichier hosts sur votre ordinateur local afin de le tester, vous voudrez probablement supprimer les lignes que vous avez ajoutées.

Conclusion

Vous pouvez créer des blocs server pour chaque domaine que vous souhaitez héberger à partir du même serveur.

Il n'existe aucune limite réelle quant au nombre de blocs server que vous pouvez créer, tant que votre matériel peut gérer le trafic.

Problèmes connus

Voir aussi

- **(en)** How To Set Up Nginx Server Blocks (Virtual Hosts) on Ubuntu 14.04 LTS

Basé sur « How To Set Up Nginx Server Blocks (Virtual Hosts) on Ubuntu 14.04 LTS » par Justin Ellingwood.

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

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

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