Logiciel

# Structure du fichier de configuration nginx.conf

## Syntaxe

### Directives

Le fichier de configuration est constitué de directives avec leurs paramètres.

- Chaque directive simple (en une ligne, un nom et des paramètres séparés par des espaces) se termine par un point-virgule **;** en bout de ligne.
- Un **bloc** de directives a la même structure qu'une directive simple, mais le point-virgule est remplacé par un ensemble d'instructions supplémentaires entourées d'accolades (**{** et **}**).

Voici quelques exemples de directives simples.

/etc/nginx/nginx.conf

```
...
user             nobody;
error_log        logs/error.log notice;
worker_processes 1;
...
```

### Contextes

Un bloc de directives qui peut avoir d'autres directives entre accolades s'appelle un **contexte** (exemples: **events**, **http**, **server** et **location**).

Quelques blocs de directives de premier niveau, appelées **contextes**, regroupent les directives qui s'appliquent à différents types de trafic :

1. **events** : Processus général de connexion
2. **http** : flux HTTP
3. **mail** : flux Mail
4. **stream** : flux TCP et UDP

Les directives placées en dehors de tout contexte sont considérées comme étant dans le **contexte principal**.

Les évènements et les directives **http** sont dans le **contexte principal**, **server** dans **http** et **location** dans **server**.

La configuration suivante illustre l'utilisation des contextes :

/etc/nginx/nginx.conf

```
user nobody; # une directive dans le contexte "principal"

events {
    # configuration du traitement de la connexion
}

http {
    # Configuration spécifique à HTTP et affectant tous les
serveurs virtuels

    server {
        # configuration du serveur HTTP virtuel 1
        location /one {
            # configuration pour le traitement des URI commençant
par '/ one'        }
        location /two {
            # configuration pour le traitement des URI commençant
par '/two'
        }
    }

    server {
        # configuration du serveur HTTP virtuel 2
    }
}

stream {
    # Configuration spécifique à TCP / UDP et affectant tous les
serveurs virtuels
    server {
        # configuration du serveur TCP virtuel 1
    }
}
```

## Commentaires

Dans une ligne, ce qui suit un **#** est un commentaire.

## Utilisation de variables

On peut utiliser des variables, dont le nom commence par le caractère **$** (dollar).

Des variables sont prédéfinies, vous pouvez définir des variables personnalisées à l'aide des directives **set**, **map** et **geo**.

Par exemple,

- **$remote_addr** contient l'adresse IP du client
- **$uri** contient la valeur réelle de l'URI.

## Directive location : configuration des emplacements

Un bloc **location** peut contenir d'autres directives **location**.

Une directive **location** a deux types de paramètres :

1. les chaînes de préfixe (chemins)
2. et les expressions régulières.

Dans l'exemple suivant, le paramètre **pathname** correspond aux requêtes qui commencent par **/some/path/**, comme **/some/path/document.html**, mais pas à /my-site/some/path car /some/path n'est pas au début de cet URI :

/etc/nginx/nginx.conf

```
location /some/path/ {
    ...
}
```

Une **expression régulière** est précédée :

- du tilde (**~**) pour une correspondance sensible à la casse,
- ou du tilde-astérisque (**~***) pour une correspondance insensible à la casse.

L'exemple suivant correspond aux URI qui incluent la chaîne .html ou .htm à n'importe quelle position.

/etc/nginx/nginx.conf

```
location ~ \.html? {
    ...
}
```

Un contexte **location** peut contenir des directives qui définissent la façon de résoudre une requête :

servir un fichier statique ou transmettre la requête à un serveur proxy.

Dans l'exemple suivant, les demandes qui correspondent au premier contexte **location** sont des fichiers servis depuis le répertoire /**data** et les requêtes correspondant au second sont transmises au serveur proxy qui héberge le contenu du domaine **www.example.com**

/etc/nginx/nginx.conf

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

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

La directive **root** spécifie le chemin du système de fichiers où rechercher les fichiers statiques à servir.

L'URI de la demande associée à l'emplacement est ajoutée au chemin pour obtenir le nom complet du fichier statique à servir.

Dans l'exemple ci-dessus, en réponse à une demande de **/images/example.png**, NGginx fournit le fichier **/data/images/example.png**.

La directive **proxy_pass** transmet la requête au serveur proxy accédé avec l'URL configurée.

La réponse du serveur proxy est ensuite renvoyée au client.

Dans l'exemple ci-dessus, toutes les demandes avec des URI qui ne commencent pas par /images/ sont transmises au serveur proxy.

# Héritage

Un contexte fils (contenu dans un autre contexte, son parent) hérite des paramètres des directives du parent.

Si une directive apparaît dans plusieurs contextes, le paramètre hérité du parent est remplacé par la directive du contexte enfant.

Par exemple, la directive **proxy_set_header**.

# Renvoyer des codes d'état spécifiques

Certains URI de site Web exigent le retour immédiat d'une réponse avec une erreur spécifique ou un code de redirection, par exemple lorsqu'une page a été déplacée temporairement ou définitivement.

La méthode la plus simple consiste à utiliser la directive **return**. Par exemple :

```
location /wrong/url {
    return 404;
}
```

1. Le premier paramètre de retour est un code de réponse.
2. Le deuxième paramètre facultatif peut être l'URL d'une redirection (pour les codes 301, 302, 303 et 307) ou le texte à renvoyer dans le corps de la réponse. Par exemple :

   ```
   location /permanently/moved/url {
       return 301 http://www.example.com/moved/here;
   }
   ```

   La directive **return** peut être incluse dans les contextes **location** et **server**.

# Rewrite : réécriture d'URI dans les demandes

la directive **rewrite** permet de modifier une URI de requête plusieurs fois lors du traitement. Elle a un paramètre facultatif et deux paramètres obligatoires :

1. expression régulière à laquelle l'URI de la requête doit correspondre (obligatoire)
2. URI à substituer à l'URI correspondant
3. un indicateur qui peut interrompre le traitement d'autres directives de réécriture ou envoyer une redirection (code 301 ou 302).

Par exemple :

```
location /users/ {
    rewrite ^/users/(.*)$ /show?user=$1 break;
}
```

Comme le montre cet exemple, le deuxième paramètre users capture en faisant correspondre les expressions régulières.

Vous pouvez inclure plusieurs directives **rewrite** dans les contextes **server** et **location**.

Nginx exécute les directives une par une dans l'ordre où elles apparaissent.

Les directives **rewrite** dans un contexte server sont exécutées une fois lorsque ce contexte est sélectionné.

Une fois que Nginx a traité un ensemble d'instructions de réécriture, il sélectionne un contexte d'emplacement en fonction du nouvel URI.

Si l'emplacement sélectionné contient des directives de réécriture, elles sont exécutées à tour de rôle.

Si l'URI correspond à l'un de ceux-ci, une recherche du nouvel emplacement démarre après le traitement de toutes les directives de réécriture définies.

L'exemple suivant montre les directives de réécriture en combinaison avec une directive **return**.

```
server {
    ...
    rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 last;
    rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra  last;
    return  403;
    ...
}
```

Cet exemple de configuration distingue deux ensembles d'URI.

Les URI tels que **/download/some/media/file** sont remplacés par **/download/some/mp3/file.mp3**.

En raison de l'indicateur **last**, les directives suivantes (la seconde réécriture et la directive return) sont ignorées mais Nginx continue à traiter la demande, qui a maintenant un URI différent.

De même, les URI tels que /download/some/audio/file sont remplacés par /download/some/mp3/file.ra.

Si un URI ne correspond à aucune directive rewrite, Nginx renvoie le code d'erreur 403 au client.

Deux paramètres interrompent le traitement des directives de réécriture :

1. **last** : Arrête l'exécution des directives de réécriture dans le contexte actuel server ou location, mais Nginx recherche les emplacements correspondant à l'URI réécri et toutes les directives de réécriture dans le nouvel emplacement sont appliquées (ce qui signifie que l'URI peut être modifiée).
2. **break** : Comme la directive break, arrête le traitement des directives de réécriture dans le contexte actuel et annule la recherche des emplacements correspondant au nouvel URI. Les directives de réécriture dans le nouvel emplacement ne sont pas exécutées.

# sub_filter : réécriture des réponses HTTP

La directive **sub_filter** permet de réécrire ou modifier le contenu d'une réponse HTTP, en substituant une chaine à une autre.

Elle permet des changements plus complexes, avec des variables et des chaines de substitution.

Par exemple, pour modifier les liens absolus qui renvoient à un autre serveur que le proxy :

/etc/nginx/nginx.conf

```
location / {
    sub_filter      /blog/ /blog-staging/;
    sub_filter_once off;
}
```

Voici un autre exemple qui modifie http: *en http_s_:* et remplace l'adresse localhost par le nom d'hôte dans le champ d'en-tête de la requête.

La directive **sub_filter_once** demande à Nginx d'appliquer successivement les directives sub_filter dans un emplacement :

/etc/nginx/nginx.conf

```
location / {
    sub_filter     'href="http://127.0.0.1:8080/'
'href="https://$host/';
    sub_filter     'img src="http://127.0.0.1:8080/' 'img
src="https://$host/';
    sub_filter_once on;
}
```

La partie de la réponse déjà modifiée par sub_filter ne sera pas remplacée à nouveau si une autre correspondance sub_filter se produit.

# Voir aussi

- **(fr)** Article
- **(en)** Article

---

*Basé sur « Article » par Auteur.*