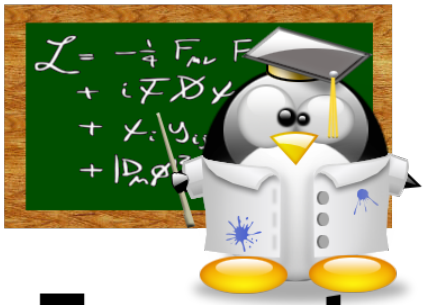

Formatux

Serveur web Nginx

Table des matières

Nginx est un serveur web **HTTP libre sous licence BSD**. Son développement commence en Russie en 2002 par Igor Sysoev. Nginx dispose, en plus des fonctionnalités standards d'un serveur web, celles de **proxy inverse** (Reverse Proxy) pour le protocole **HTTP** mais aussi de **proxy** pour les protocoles de messageries **POP et IMAP**.

Le développement du serveur nginx est une réponse au problème **C10K** : supporter 10 000 connexions concurrentes (chose courante sur le web moderne) est un vrai challenge pour des serveurs web.

Un support commercial est possible par Nginx Inc.

Chapitre 1. Généralités

L'architecture interne du serveur permet des **performances très élevées** avec une **faible consommation de charge mémoire** en comparaison avec ce que peut faire le serveur web Apache notamment.

Les modules venant compléter les fonctions de base du noyau nginx sont liés à la compilation : ils ne peuvent pas être activés/désactivés à chaud.

Les processus serveurs sont contrôlés par un processus maître, rendant la **modification de la configuration ou la mise à jour du logiciel possible sans arrêt du service**.

Nginx détient une part de marché non négligeable de 28% sur les sites les plus chargés du marché, juste derrière Apache (41%).

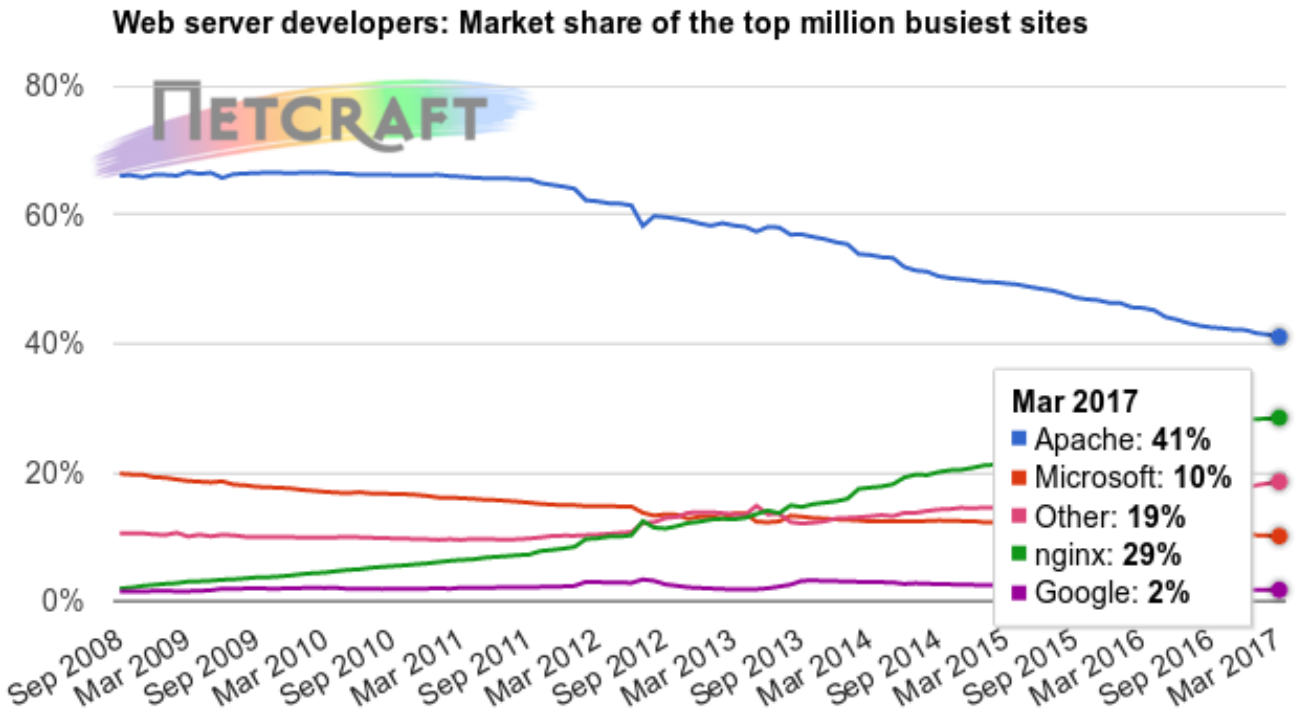

Figure 1. Statistiques NetCraft : top million busiest sites

1.1. Fonctionnalités

Nginx offre les fonctionnalités basiques suivantes :

- Hébergement de pages web statiques ;
- Génération de pages d'index automatique ;
- Proxy inverse accéléré avec cache ;
- Répartition de charge ;

- Tolérance de panne ;
- Support avec cache du FastCGI, uWSGI, SCGI et serveur de cache memcached ;
- Filtres divers pour gzip, xslt, ssi, transformation d'images, …
- Support pour SSL/TLS et SNI ;
- Support du HTTP/2.

Autres fonctionnalités :

- Hébergement par nom ou par adresse IP ;
- Gestion du keepalive des connexions clientes ;
- Gestion des logs : syslog, rotation, buffer ;
- Ré-écriture d'URI ;
- Contrôle d'accès : par IP, mot de passe…
- Streaming FLV et MP4.

Chapitre 2. Installation du service

2.1. Nginx sous debian

Depuis Debian Wheezy, l'installation de Nginx est proposée par 3 paquets : nginx-light (le moins de modules), nginx-full (installé par le méta-paquet nginx - paquet par défaut), nginx-extras (le plus de modules).

Installation du metapaquet nginx

```
sudo apt-get install nginx
...
Les paquets supplémentaires suivants seront installés :
  libgd3 libvpx1 libxpm4 nginx-common nginx-full
Paquets suggérés :
  libgd-tools fcgiwrap nginx-doc ssl-cert
Les NOUVEAUX paquets suivants seront installés :
  libgd3 libvpx1 libxpm4 nginx nginx-common nginx-full
...
```

2.2. Nginx sous RedHat 7

Le dépôt de nginx pour RHEL/CentOS doit être ajouté aux dépôts existants. Créer le fichier /etc/yum.repos.d/nginx.repo:

```
$ sudo vi /etc/yum.repos.d/nginx.repo
```

Et ajouter le contenu suivant :

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=0
enabled=1
```

L'installation peut être lancée :

```
$ sudo yum update
$ sudo yum install nginx
```

2.3. Configuration de Nginx

La configuration de Nginx se situe sous ***/etc/nginx*** :

- Le fichier **/etc/nginx/nginx.conf** : fichier de configuration globale du serveur. Les paramètres impactent l'ensemble du serveur.
- le répertoire **sites-available** : contient les fichiers de configuration des sites.
- le répertoire **sites-enabled** : contient des liens symboliques vers les fichiers de **sites-available**, ce qui permet d'activer ou de désactiver les sites.
- le répertoire **conf.d** : répertoire contenant les paramètres communs à tous les sites.

La fonctionnalité de fichier .htaccess connues des administrateurs Apache, n'existe pas sous nginx !

Le fichier nginx.conf, épuré de tous ses commentaires, et fourni ci-dessous à titre indicatif :

Fichier nginx.conf par défaut

```
user www-data;
worker_processes 4;
pid /run/nginx.pid;

events {
    worker_connections 768;
}

http {
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    gzip on;
    gzip_disable "msie6";

    application/javascript text/xml application/xml application/xml+rss
text/javascript;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

Table 1. Directives de la configuration par défaut

Directives	**Observations**
user	Définit l'**utilisateur** et le **groupe** propriétaires du processus. Si le groupe n'est pas spécifié, le groupe du même nom que l'utilisateur est utilisé.
worker_processes	Définit le **nombre de processus**. La valeur optimale dépend de nombreux facteurs comme le nombre de coeurs CPU, les spécificités des disques durs, etc. En cas de doute, la documentation de nginx propose comme valeur de départ le nombre équivalent au nombre de coeurs CPU disponibles (la valeur **auto** essaiera de le déterminer).

Directives	Observations
pid	Définit un fichier pour stocker la valeur du **pid**.
worker_connections	Fixe le **nombre maximum de connexions simultanées** qu'un processus worker peut ouvrir (vers le client et vers les serveurs mandatés).
tcp_nopush	tcp_nopush est indissociable de l'option sendfile. Elle permet d'**optimiser la quantité d'information envoyée en une seule fois**. Les paquet ne sont envoyés que lorsque ils ont atteints leur taille maximale.
tcp_nodelay	Activer tcp_nodelay force l'**envoi immédiat des données** contenues dans la socket, quelle que soit la taille du paquet, ce qui est le contraire de ce que fait tcp_nopush.
sendfile	Optimiser l'envoi de **fichiers statiques** (option inutile dans le cadre d'une configuration en proxy-inverse). Si **sendfile** est activée, nginx s'assure que tous les paquets soient bien remplis avant d'être envoyés au client (grâce à tcp_nopush), puis, quand arrive le dernier paquet, nginx désactive tcp_nopush, et force l'envoi des données avec tcp_nodelay.
keepalive_timeout	**temps maximum avant fermeture** d'une connexion inactive.
types_hash_max_size	Nginx entretient des tables de hashage contenant des informations statiques. Permet de définir **la taille maximale de la table de hachage**.
include	Inclure un autre fichier ou d'autres fichiers qui correspondent au modèle fourni dans la configuration.
default_type	Type MIME par défaut d'une requête.
ssl_protocols	Versions du protocole TLS acceptés.
ssl_prefer_server_ciphers	Préférer l'utilisation de la cipher suite du serveur plutôt que celle du client.
access_log	Configurer les **journaux d'accès** (voir paragraphe "gestion des logs").
error_log	Configurer les **journaux d'erreurs** (voir paragraphe "gestion des logs").
gzip	Le module **ngx_http_gzip_module** est un filtre compressant les données transmises au format gzip.
gzip_disable	Désactiver gzip en fonction d'une expression régulière.

La configuration de nginx est articulée de la manière suivante :

```
# directives globales

events {
    # configuration du worker
}

http {
    # configuration du service http

    # Configuration du premier serveur en écoute sur le port 80
    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;
        }
    }
}

mail {
    # configuration du service mail

    # directives globales du service mail

   server {
        # Un premier serveur en écoute sur le protocole pop
        listen     localhost:110;
        protocol   pop3;
        proxy      on;
   }


   server {
        # Un second serveur en écoute sur le protocole imap
       listen     localhost:143;
       protocol   imap;
       proxy      on;
   }
}
```

La configuration du premier serveur en écoute sur le port 80 se situe sous **/etc/nginx/sites-available/default**. Ce fichier est inclue au fichier nginx.conf grâce à la ligne **include /etc/nginx/sites-enabled/*;**

2.4. Configuration https

Pour configurer un service https, il faut ajouter un bloc serveur, ou modifier le bloc server existant (un bloc server peut à la fois écouter sur le port 443 et sur le port 80).

Ce bloc peut, par exemple, être ajouté au nouveau fichier *sites-available/default_https* :

```
server {
    listen              443 ssl default_server;
    ssl_protocols       TLSv1.2 TLSv1.1
    ssl_certificate     /chemin/vers/cert.pem;
    ssl_certificate_key /chemin/vers/key.key;
    root                /var/www/html;
    index               index.html index.htm index.nginx-debian.html;
    server_name         _;
    location / {
        try_files       $uri $uri/ =404;
    }
}
```

ou le server par défaut peut être modifié pour prendre en compte le https :

```
server {
    listen              80;
    listen              443 ssl;
    server_name         _;
    ssl_protocols       TLSv1.2 TLSv1.1
    ssl_certificate     /chemin/vers/cert.pem;
    ssl_certificate_key /chemin/vers/key.key;
    ...
}
```

2.5. La gestion des logs

La directive **error_log** permet de configurer les journaux d'erreurs.

Syntaxe de la directive error_log

```
error_log fichier [niveau];
```

Le premier paramètre définit un fichier qui va recevoir les logs.

Le second paramètre détermine le niveau des logs : debug, info, notice, warn, error, crit, alert ou emerg (voir le cours syslog).

L'envoi des enregistrements vers syslog peut être effectué en employant le préfixe "syslog:".

```
access_log syslog:server=192.168.1.100:5514,tag=nginx debug;
```

2.6. Nginx en proxy inverse

La fonctionnalité de proxy inverse est fourni par le module **ngx_http_upstream_module**. Il permet de définir des groupes de serveurs qui sont ensuite appelés par les directives proxy_pass ou fastcgi_pass, memcached_pass, etc.

Exemple de configuration basique, qui réparti la charge de 2/3 vers le premier serveur et d'1/3 vers le second serveur applicatif :

```
    upstream svrmetiers {
        server metiers1.formatux.fr:8080       weight=2;
        server metiers2.formatux.fr:8080       weight=1;
    }

    server {
        location / {
            proxy_pass http://svrmetiers;
        }
    }
```

Des serveurs peuvent être déclarés en secours :

```
    upstream svrmetiers {
        ...
        server secours1.formatux.fr:8080   backup;
        server secours2.formatux.fr:8080   backup;

    }
```

La directive serveur accepte de nombreux arguments :

- **max_fails=nombredetentative** : fixe le nombre de tentatives de connexion devant être en echec durant le laps de temps défini par la paramètre **fail_timeout** pour que le serveur soit considéré comme indisponible. La valeur par défaut est fixée à 1, la valeur à 0 désactive la fonctionnalité.
- **fail_timeout=time**: fixe la durée durant laquelle un nombre de connexion défini bascule le serveur comme indisponible et fixe la période de temps durant laquelle le serveur sera considéré comme indisponible. La valeur par défaut est de 10 secondes.

Chapitre 3. Sources

- https://t37.net/optimisations-nginx-bien-comprendre-sendfile-tcp-nodelay-et-tcp-nopush.html
- http://nginx.org/en/docs/