

Serveurs web

André Abramé

`andre.abrame@logike.fr`

Plan I

1. Introduction

2. Le protocole HTTP

3. Apache

3.1 Introduction

3.2 Installation

3.3 Démarrer, arreter et voir le status d'apache

3.4 Déployer un site web

3.5 Configuration du serveur

3.6 Virtual hosts

3.7 Contrôle d'accès

4. Nginx

4.1 Introduction

4.2 Installation

4.3 Démarrer, arreter et recharger nginx

4.4 Configuration du serveur

Introduction

1. Introduction

2. Le protocole HTTP

3. Apache

4. Nginx

Introduction

Qu'est ce qu'un serveur HTTP ?

- un logiciel
- qui héberge des ressources
- qui met ces ressources à disposition par le protocole HTTP

Quelles ressources ?

- des ressources **statiques** : pages HTML, images, vidéos, ...
- des ressources **dynamiques** : des programmes qui génèrent la réponse à envoyer à l'utilisateur (CGI, PHP, Java, ...)

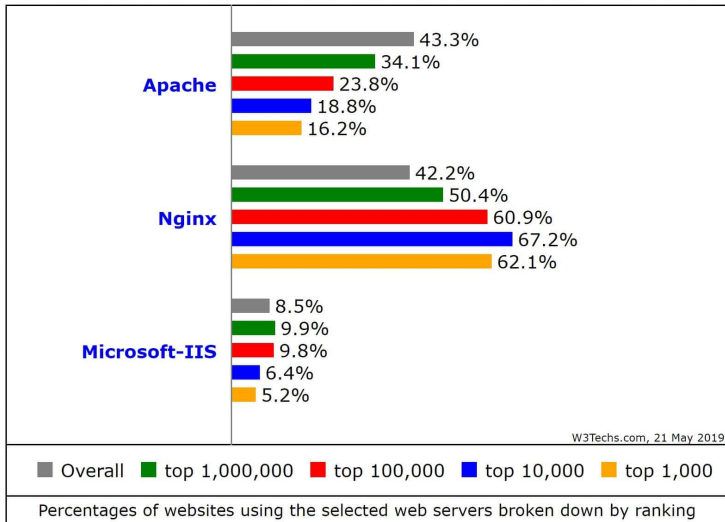
Introduction

Les principaux serveurs webs

- Apache (Fondation Apache)
- nginx (NGINX, Inc.)
- IIS (Microsoft)
- LiteSpeed Web Server (LiteSpeed Technologies)
- GWS (Google)

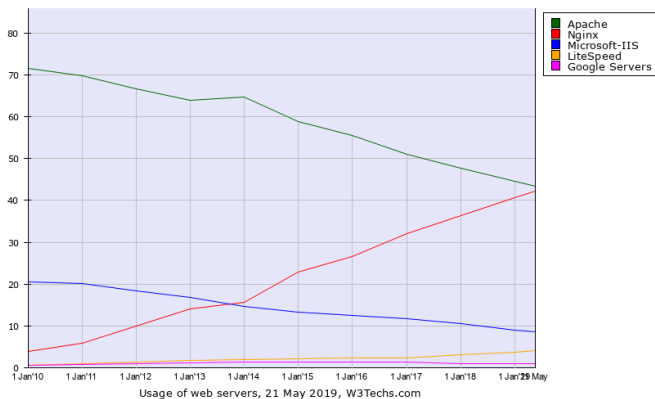
Introduction

Parts de marché des serveurs webs



Introduction

Parts de marché des serveurs webs



Le protocole HTTP

1. Introduction

2. Le protocole HTTP

3. Apache

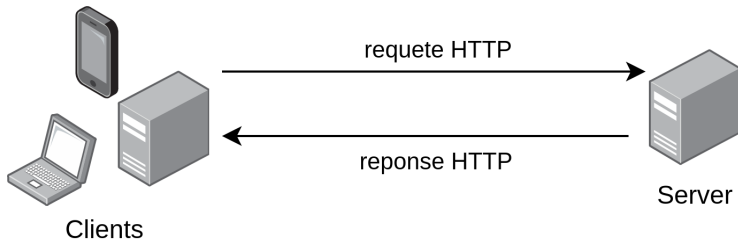
4. Nginx

Le protocole HTTP

HTTP (HyperText Transfer Protocol)

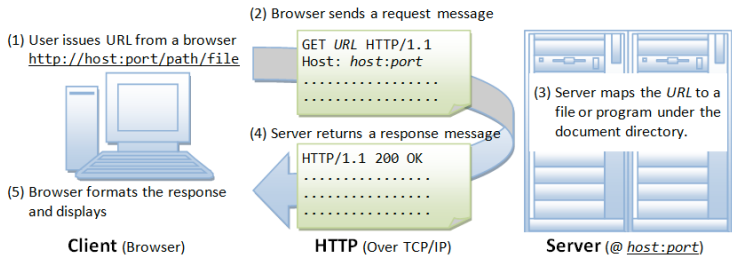
Protocole d'échange d'informations sur le web

- basé sur TCP/IP
- les clients envoient des requêtes
- les serveur envoient des réponses



Le protocole HTTP

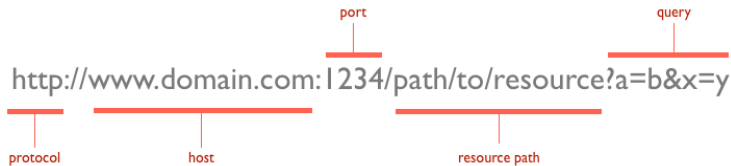
Exemple d'échange client / serveur



Le protocole HTTP

Les URL (Unique Resource Location)

Identifie les ressources de manière unique.



Le protocole HTTP

Les méthodes HTTP

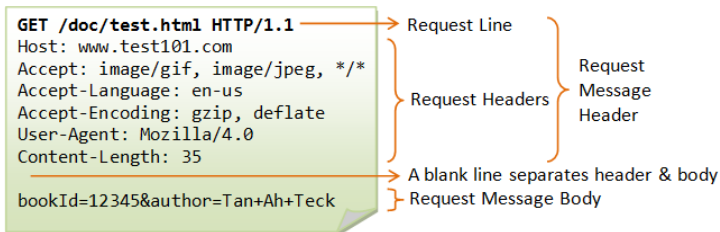
Spécifie l'action effectuer sur la ressource. Les principales :

- **GET** : requests a representation of the specified resource.
- **POST** : submit an entity to the specified resource.
- **PUT** : replaces all current representations of the target resource.
- **DELETE** : deletes the specified resource.

Et les autres : HEAD, CONNECT, OPTIONS, TRACE, PATCH.

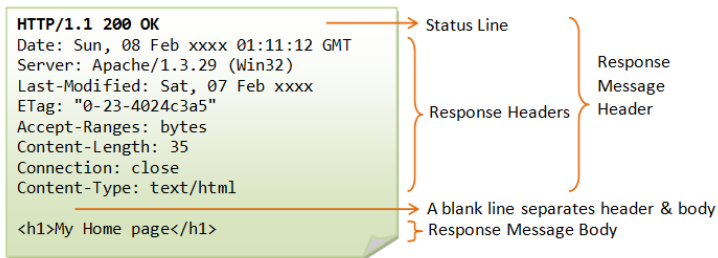
Le protocole HTTP

Requête



Le protocole HTTP

Réponse



Apache

1. Introduction

2. Le protocole HTTP

3. Apache

3.1 Introduction

3.2 Installation

3.3 Démarrer, arreter et voir le status d'apache

3.4 Déployer un site web

3.5 Configuration du serveur

3.6 Virtual hosts

3.7 Contrôle d'accès

4. Nginx

Apache

Introduction

Caractéristiques

- serveur web le plus utilisé
- gratuit et open source
- maintenu par la fondation Apache
- fonctionne sur la plupart des OS

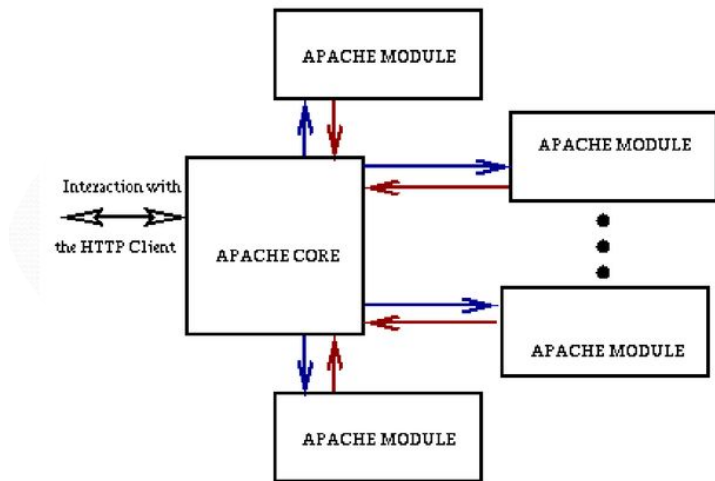
Fonctionnalités

- ressources statiques
- ressources dynamiques (Perl, Python, Tcl, PHP, ...)
- authentication / authorization
- virtual hosts (plusieurs sites sur le même serveur)
- configuration des accès par répertoire (.htaccess)
- TLS/SSL
- reverse proxy

Apache

Introduction

Architecture



Apache

Installation

Debian/Ubuntu

```
sudo apt install apache2
```

Autres plateformes

voir la documentation

Apache

Démarrer, arreter et voir le status d'apache

avec apache2ctl

```
$ apache2ctl --help
Usage: /usr/sbin/apache2ctl
↪ start|stop|restart|graceful|graceful-stop|configtest|status|fullstatus|help
   /usr/sbin/apache2ctl <apache2 args>
   /usr/sbin/apache2ctl -h           (for help on <apache2 args>)
```

- `apache2ctl start` : démarrer le serveur
- `apache2ctl stop` : arrêter le serveur
- `apache2ctl restart` : redémarrer le serveur
- `apache2ctl status` : information sur l'état du serveur

avec systemctl

- `systemctl start apache2.service` : démarrer le serveur
- `systemctl stop apache2.service` : arrêter le serveur
- `systemctl restart apache2.service` : redémarrer le serveur
- `systemctl status apache2.service` : information sur l'état du serveur

Apache

Déployer un site web

Par défaut, les sites sont situés dans `/var/www/`. Pour tester, exécuter la commande :

```
$ echo "<html><body>Hello world</body><html>" > /var/www/index.html
```

Puis, dans votre navigateur préféré, connectez-vous sur `localhost:80`.

Apache

Configuration du serveur

L'emplacement des fichiers de configuration varie selon les OS. Sur debian/ubuntu, ils sont situés dans `/etc/apache2/`.

```
$ tree -L 1 /etc/apache2/  
/etc/apache2/  
├── apache2.conf  
├── conf-available  
├── conf-enabled  
├── envvars  
├── magic  
├── mods-available  
├── mods-enabled  
├── ports.conf  
├── sites-available  
└── sites-enabled  
  
6 directories, 4 files
```

Apache

Configuration du serveur

`/etc/apache2/apache2.conf`

Fichier de configuration principal, contenant :

- comportement basique du serveur (processus, timeouts, ...)
- liens vers d'autres fichiers de configuration
- nom de l'utilisateur qui exécute apache
- règles d'accès par défaut aux répertoires
- ...

`/etc/apache2/envvars`

Les variables d'environnement d'apache.

`/etc/apache2/ports.conf`

Les ports écoutés par apache.

Les modules d'apache (1/2)

Les modules d'apache étendent les fonctionnalités du logiciel.
De nombreux modules sont disponibles (538 au 8 décembre 2011).
Certain n'ont pas été mis à jour depuis très longtemps.

Modules disponibles

Les modules disponibles sont placés dans le répertoire `/etc/apache2/mods-available/`

Deux types de fichiers existent :

- Les fichiers `.load`, qui contiennent des directives sur les bibliothèques (des `.so`) à charger (`LoadModule`) et sont très similaires les uns aux autres.
- Les fichiers `.conf`, qui contiennent des directives de configuration du module et qui sont très spécifiques à chaque module.

Les modules d'apache (2/2)

Modules activés

Les modules activés se trouvent dans le répertoire

`/etc/apache2/mods-enabled/`

En pratique, ce ne sont pas des fichiers mais des liens vers les fichiers

`.load` et `.conf` situés dans le répertoire

`/etc/apache2/mods-available/`

Pour activer un module, il suffit de créer un lien dans `mods-enabled` vers les fichiers `.load` et `.conf` situés dans `mods-available` (l'utilitaire `a2enmod` est là pour ça).

Chargement des modules activés

Le chargement des modules activés se fait lors du (re)démarrage de

apache, et plus particulièrement lors de l'évaluation de `apache2.conf` :

```
Include /etc/apache2/mods-enabled/*.load
```


Installation de modules supplémentaires

apt-get

Les modules apache sont des bibliothèques et s'installent comme telles.

```
root@server$ apt-get install libapache2-mod-php5
```

The following NEW packages will be installed:

libapache2-mod-php5 php5-common

```
root@server$ ls /etc/apache2/mods-available/*php*
```

```
/etc/apache2/mods-available/php5.conf
```

```
/etc/apache2/mods-available/php5.load
```

```
root@server$ ls /etc/apache2/mods-enabled/*php*
```

```
/etc/apache2/mods-enabled/php5.conf
```

```
/etc/apache2/mods-enabled/php5.load
```

Les modules installés apparaissent dans mods-available et sont activés automatiquement.

Il faut relancer apache pour que les modifications soient effectives !

```
root@server$ apache2ctl restart
```


Tester apache+php

Pour tester l'interaction le module php5 de apache, le mieux est de créer un fichier appelant phpinfo() et de le demander au serveur.

```
root@server$ cat /var/www/index.php
<?php phpinfo() ?>
```

Bavard...

Attention, phpinfo() est particulièrement bavard et il vaut mieux, pour des raisons de sécurité, ne pas le laisser trainer sur le serveur.

PHP Version 5.2.6-1+lenny13	
	
System	Linux mc-188-165-56-174.ovh.net 2.6.34.6-xxxx-grs-ipv6-64 #3 SMP Fri Sep 17 16:06:38 UTC 2010 x86_64
Build Date	Jul 1 2011 15:54:30
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
additional .ini files parsed	/etc/php5/apache2/conf.d/ldap.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220080519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, phar, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.*, bzcomp*, zlib*
This server is protected with the Suhosin Patch 0.9.6.2 Copyright (c) 2006 "Suhosin-PHP Project"	
수호신	

Fonctionnement des *Virtual Hosts*

Distinguer les *Virtual Hosts*

Apache distingue les différents *Virtual Hosts* grâce au paramètre `Host` de la requête `http` du client.

En pratique, chaque site web situé d'un hébergement mutualisé donné est nommé par une entrée différente dans le DNS.

En pratique

Les hôtes virtuels déployables sont décrits dans les fichiers situés dans `/etc/apache2/sites-available/`.

Les hôtes virtuels déployés sont décrits par les liens sur ces mêmes fichiers situés dans `/etc/apache2/sites-enabled`.

L'outil `a2ensite` permet de créer facilement le lien dans `sites-enabled` à partir du nom du fichier dans `sites-available`.

Les fichiers de description des *Virtual Hosts*

Les descriptions de *Virtual Hosts* dans les fichiers sont placées entre des balises :

```
<VirtualHost *:80>
```

```
...
```

```
</VirtualHost>
```

Le *virtual host* décrit ici sera proposé sur toutes les adresses IP, et sur le port 80.

Informations primordiales

Les informations primordiales dans un *virtual host* sont :

- **ServerName** : valeur du paramètre `host` : pour lequel le *virtual host* sera choisi.
- **DocumentRoot** : emplacement des fichiers du site du *virtual host*.

Modules et *virtual hosts* : outils pratiques

Les quatre outils suivants sont très pratiques pour l'activation et la désactivation de modules et de *virtual hosts* :

a2enmod

Active un module en créant le lien dans `mods-enabled`.

a2dismod

Désactive un module en supprimant le lien dans `mods-enabled`.

a2ensite

Active un *virtual host* en créant le lien dans `sites-enabled`.

a2dissite

Désactive un *virtual host* en supprimant le lien dans `sites-enabled`.

Contrôle d'accès

Pourquoi faire ?

Par défaut, le serveur apache fournira toutes les pages présentes dans le répertoire du site.

Néanmoins, il pourrait être souhaitable dans certaines circonstances de contrôler l'accès à certains fichiers/répertoires (autorisé à tous, interdit à tous, réservé à certains utilisateurs authentifiés).

Comment le faire ?

La configuration du contrôle d'accès peut se faire :

- Au niveau de la configuration globale du site.
- Au niveau de chaque répertoire si la configuration globale du site le permet. C'est le rôle des fichiers `.htaccess`.

Portée des directives : Directory, File, Location

Directory

Les directives portant sur des répertoires sont placées dans des balises `<Directory nom_rep> ... </Directory>`

File

Les directives portant sur des fichiers sont placées dans des balises `<Files nom_fich> ... </Files>`

Location

Les directives portant sur des *URL* sont placées dans des balises `<Location url> ... </Location>`

Expressions régulières

Les balises <DirectoryMatch>, <FilesMatch> et <LocationMatch> permettent l'utilisation d'expressions régulières.

Authentification des utilisateurs

Fichier de mots de passes

L'outil htpasswd permet de créer un fichier de mots de passes :

```
root@server$ htpasswd -c /var/www/.htpasswd neeko
```

New password:

Re-type new password:

Adding password for user neeko

```
root@server$ less /var/www/.htpasswd
```

```
neeko:j.qH.WnsISR4g
```

Protection du fichier .htpasswd

Il ne faut évidemment pas que le fichier de mots de passes soit accessible de l'extérieur. /etc/apache2/apache2.conf contient une ligne de configuration qui interdit explicitement l'accès aux fichiers .htpasswd.

Contrôle d'accès dans les fichiers de configuration

Le fichier de configuration `/etc/apache2/apache2.conf` et chaque fichier de configuration de site peuvent contenir des politiques de contrôle d'accès.

Protection du dossier `/var/www/name` pour l'utilisateur neeko

```
<Directory /var/www/name>
    AuthType Basic
    AuthName "Mot de passe necessaire"
    AuthUserFile /var/www/.htpasswd
    Require user neeko
</Directory>
```

Les contrôles s'exercent sur le répertoire indiqué et sur les sous-répertoires et fichiers qu'il contient.

Utilisation de l'authentification Digest

Création du fichier des mots de passes

```
htdigest crée les fichiers de mots de passe pour Digest :  
root@server$ htdigest -c .htdigest myrealm neeko  
New password:  
Re-type new password:  
root@server$ less .htdigest  
neeko:myrealm:7c90f03dc61192989cfcfc92f2a8e259
```

Exemple de configuration

```
<Directory /var/www/name/truc>  
    AuthType Digest  
    AuthName "myrealm"  
    AuthDigestDomain /var/www/name/truc  
    AuthDigestProvider file  
    AuthUserFile /var/www/.htdigest  
    Require valid-user  
</Directory>
```


Utiliser les groupes pour l'authentification

Exemple d'utilisation

```
root@server$ less .htdigest
neeko:myrealm:7c90f03dc61192989cfcfc92f2a8e259
bilbon:myrealm:dd7246d00183cbb58c95c12ec2819b50
root@server$ less .htgroup
users : bilbon neeko
admins : neeko
root@server$ less /etc/apache2/sites-enabled/name
...
<Directory /var/www/name/truc>
    AuthType Digest
    AuthName "myrealm"
    AuthDigestDomain /var/www/name/truc
    AuthDigestProvider file
    AuthUserFile /var/www/.htdigest
    AuthGroupFile /var/www/.htgroup
    Require group admins
</Directory>
```


Authentification du serveur et confidentialité : https

Contrôle d'accès n'est pas confidentialité

Avec les mécanismes que nous avons présenté jusqu'à présent, rien n'empêche un attaquant d'écouter les communications et donc d'accéder en lecture à toutes les pages consultées par le client légitime.

Authentification unidirectionnelle et usurpation d'identité

De plus, aucune authentification du serveur n'est réalisée. En d'autres termes, rien n'empêche un attaquant d'usurper l'identité du serveur pour faire révéler son mot de passe au client ou pour s'en servir comme d'un oracle.

https

Pour ces raisons, le besoin d'une solution cryptographique de protection des communications s'est fait sentir, et https a été proposé. Il consiste en l'application de ssl à http.

Fonctionnalités d'https

Authentification du serveur

https permet l'authentification du serveur de manière cryptographique via l'utilisation de certificats émis par une autorité de confiance portant sur la clé publique du serveur.

Confidentialité et authenticité des communications

Les sessions https entre le client et le serveur sont protégées cryptographiquement en confidentialité et en authenticité grâce à une clé générée par le client pour chaque session.

Authentification mutuelle (optionnel)

Optionnellement, `https` permet, en plus de l'authentification du serveur, l'authentification forte du client basée sur la possession par celui-ci d'un certificat signé pour sa clé publique.

Apache et https

Activation du module ssl

Pour qu'https fonctionne correctement, le module ssl doit être activé :

```
root@server$ a2enmod ssl
```

Configuration du site/*virtual host* pour le faire fonctionner en https

Pour qu'un site/*virtual host* fonctionne en https, les directives suivantes doivent lui être fournies :

```
SSLEngine on # Activation du moteur SSL
```

```
SSLCertificateFile /etc/ssl/certs/moncert.pem # Certificat
```

```
SSLCertificateKeyFile /etc/ssl/private/macle.pem # Clé privée
```

Si un site doit fonctionner à la fois en http et en https, deux VirtualHosts doivent être créés.

Clé et certificat par défaut

apache dispose d'une clé par défaut et du certificat associé

Générations de la paire de clés et du certificat du serveur

Certificats auto-signés

Lorsqu'on veut juste assurer la confidentialité des communications, il peut être suffisant d'utiliser un certificat auto-signé, c-à-d un certificat signé avec la clé privée correspondant à la clé publique sur laquelle il porte.

Attention

Un certificat auto-signé n'apporte aucune garantie d'authenticité car n'importe qui peut le générer.

make-ssl-cert

make-ssl-cert permet de générer une clé privée et le certificat associé.
\$ make-ssl-cert /usr/share/ssl-cert/ssleay.cnf ./mycert.pem
où /usr/share/ssl-cert/ssleay.cnf est un fichier contenant la conf. pour la clé et mycert.pem le nom du fichier contenant le certificat.
La clé sera écrite dans un fichier nommé par un nombre du même répertoire.

Autorités de certifications

Les garants de l'authenticité des certificats

Les autorités de certifications jouent le rôle de tiers de confiance.
Leur business est de certifier les clés publiques de leurs clients après avoir vérifié leur identité.

Les certificats (auto-signés) des autorités de certification étant incluses par défaut dans les navigateurs et étant considérés comme de confiance, les navigateurs peuvent vérifier les certificats des sites et donc leur légitimité.

Limites des autorités de certification

La confiance qu'on prête aux autorités de certification est parfois trahie. Récemment, par exemple, Diginotar (mais ce n'est pas le seul) a été compromis et des certificats frauduleux ont été émis.

Nginx

1. Introduction

2. Le protocole HTTP

3. Apache

4. Nginx

4.1 Introduction

4.2 Installation

4.3 Démarrer, arreter et recharger nginx

4.4 Configuration du serveur

Nginx

Caractéristiques

- un des serveurs webs les plus utilisés
- licence BSD
- maintenu par la Nginx Inc. (qui fournit aussi un support commercial)

Fonctionnalités

- ressources statiques
- ressources dynamiques (Perl, Python, Tcl, PHP, ...)
- bonne performance, faible occupation mémoire (vs. apache)
- TLS/SSL
- reverse proxy pour HTTP, POP et IMAP

Nginx

Installation

Debian/Ubuntu

```
sudo apt install nginx
```

Autres plateformes

voir la documentation

Nginx

Démarrer, arreter et recharger nginx

avec `nginx -s <signal>`

On contrôle nginx en lui envoyant des signaux. Cela peut être fait en appelant :

```
$ nginx -s <signal>
```

Où `<signal>` peut être :

- `stop` : arrêter rapidement
- `quit` : arrêter proprement
- `reload` : recharger la configuration
- `reopen` : rouvrir les fichiers de logs

avec `systemctl`

On peut toujours contrôler le service via `systemctl` :

- `systemctl start nginx.service` : démarrer le serveur
- `systemctl stop nginx.service` : arrêter le serveur
- `systemctl restart nginx.service` : redémarrer le serveur
- `systemctl status nginx.service` : information sur l'état du serveur

Nginx

Configuration du serveur - structure du répertoire

L'emplacement des fichiers de configuration varie selon les OS. Sur debian/ubuntu, ils sont situés dans `/etc/nginx/`.

```
$ tree -L 1 /etc/nginx/  
/etc/nginx/  
├── conf.d  
├── fastcgi.conf  
├── fastcgi_params  
├── koi-utf  
├── koi-win  
├── mime.types  
├── modules-available  
├── modules-enabled  
├── nginx.conf  
├── proxy_params  
├── scgi_params  
├── sites-available  
├── sites-enabled  
├── snippets  
├── uwsgi_params  
└── win-utf
```

```
6 directories, 10 file
```

Nginx

Configuration du serveur - les fichiers de configuration

`/etc/nginx/nginx.conf`

Fichier de configuration principal, contenant :

- comportement global du serveur (utilisateur, processus, timeouts, ...)
- nom de l'utilisateur qui exécute apache
- ...

`/etc/nginx/sites-available`

Les fichiers de configuration des sites.

`/etc/nginx/sites-enabled`

Des liens symboliques vers les fichiers de `sites-available` pour les sites actifs.

Nginx

Configuration du serveur - /etc/nginx/nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {

    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    # server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

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

Nginx

Configuration du serveur - /etc/nginx/nginx.conf

```
##
# SSL Settings
##

ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
ssl_prefer_server_ciphers on;

##
# Logging Settings
##

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

##
# Gzip Settings
##

gzip on;

# gzip_vary on;
# gzip_proxied any;
# gzip_comp_level 6;
# gzip_buffers 16 8k;
# gzip_http_version 1.1;
# gzip_types text/plain text/css application/json application/javascript text/xml
↪ application/xml application/xml+rss text/javascript;

##
```

Nginx

Configuration du serveur - /etc/nginx/nginx.conf

```
# Virtual Host Configs
##

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

#mail {
# # See sample authentication script at:
# # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
# # auth_http localhost/auth.php;
# # pop3_capabilities "TOP" "USER";
# # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
# server {
#     listen     localhost:110;
#     protocol   pop3;
#     proxy      on;
# }
#
# server {
#     listen     localhost:143;
#     protocol   imap;
#     proxy      on;
# }
#}
```

Nginx

Configuration du serveur - /etc/nginx/sites-available/default

```
$ cat /etc/nginx/sites-available/default
##
# 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;
```

Nginx

Configuration du serveur - /etc/nginx/sites-available/default

```
#
# 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;
```

Nginx

Configuration du serveur - /etc/nginx/sites-available/default

```
#
# # With php-fpm (or other unix sockets):
# fastcgi_pass unix:/var/run/php/php7.4-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;
```


Nginx

Configuration du serveur - `/etc/nginx/sites-available/default`

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

Nginx

Configuration du serveur - activer / désactiver un virtual host

Activation

Il suffit de créer un lien symbolique vers sites-enabled :

```
$ ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/
```

Desactivation

On supprime le lien symbolique de sites-enabled :

```
$ rm /etc/nginx/sites-enabled/default
```