Introduction

Allons-y

Le dixième docteur

Votre ordinateur est un serveur. En tout cas, il peut le devenir pour peu qu'on y installe et configure les logiciels idoines.

Les deux serveurs web les plus utilisés [1] sont les logiciels libres Nginx [2] et Apache [3], objets de ce cours.

D'autres solutions existent :

— libres : lighttpd, thttpd, Mathopd, Boa…
— commerciales : Stronghold (Red Hat), Oracle iPlanet Web Server (Oracle), IIS (Microsoft)…

Certains de ces serveurs présentent des avantages sur Apache et Nginx, dans des contextes particuliers d'utilisation ou simplement parce qu'ils sont associés à une assistance technique.

Le choix de tel ou tel logiciel dépend généralement de plusieurs critères :

— familiarité des administratrices systèmes avec le logiciel ;
— pré-requis du site à déployer ;
— support commercial possible ;
— choix historique : on arrive sur des serveurs où le choix a déjà été fait des années auparavant et une migration prendrait énormément de temps et/ou ne serait pas rentable.

1 Comment fonctionne le web ?

1.1 Principe du Web

Le Web est fondé sur l'utilisation de trois composantes principales :

— les URI (*Uniform Resource Identifier*), permettant d'identifier une ressource accessible via Internet ;
— le langage HTML (*HyperText Markup Language*), langage de description de données ;
— le protocole HTTP (*HyperText Transfer Protocol*), permettant de formaliser et de normaliser des échanges entre les clients et les serveurs Web.

1.2 Rôles d'un serveur Web

Rôle principal : attendre les requêtes des clients sur un port précis d'une machine donnée, afin de leur transmettre des documents. Les échanges entre client et serveur sont normalisés via le protocole HTTP.

Rôles secondaires : donner des méta-informations sur les documents transmis, entretenir des communications avec des applications tierces (base de données, réseau) généralement dans le but de transmettre des documents qui sont générés de façon *dynamique*, c'est-à-dire à la demande.

1.3 Le protocole HTTP

Avant d'attaquer la configuration des logiciels, attardons-nous un peu sur le protocole de communication qu'ils mettent en œuvre

1. source : https://news.netcraft.com/archives/2021/10/15/october-2021-web-server-survey.html
2. https://nginx.org/
3. https://httpd.apache.org/

Les différentes normes

Le protocole a peu évolué depuis sa création : seulement 4 normes se sont succédées.

— La première version, appelée 0.9, était très simple. Cette norme se contentait de fournir les documents qui étaient demandés.
— La norme 0.9 a été remplacée en mai 1996 par la norme 1.0, clairement définie par la RFC 1945 [4]. L'ajout le plus important a été l'utilisation d'en-têtes (les méta-informations évoquées plus haut) décrivant le type des données transmises, indiquant ainsi aux clients la façon de traiter les données qu'ils reçoivent.
— La norme 1.1, date de juin 1999 et est décrite par la RFC 2616 [5] bien que sa première définition date de 1997, dans la RFC 2068 [6]. Les apports de cette norme sont des méthodes supplémentaires de communication entre les clients et le serveur, ainsi que le support des hôtes virtuels (méthode permettant à un serveur d'accueillir plus d'un nom de domaine. Nous y reviendrons plus tard).
— Une nouvelle norme, HTTP/2 a été déposée en mai 2015 dans la RFC 7540 [7]. Elle est plus une surcouche à HTTP/1.1 qu'une évolution de celle-ci, contrairement aux précédentes évolutions. Apache et Nginx permettent son utilisation, respectivement depuis les versions 2.4.17 et 1.9.5 [8].
— HTTP/3, encore à l'état de brouillon Internet [9] (donc encore en travail, bien que déjà utilisable sur certains sites et avec certains navigateurs), est le successeur annoncé d'HTTP/1.1 et HTTP/2

Les méthodes de requête

Toutes les requêtes du protocole HTTP, quelle que soit la version utilisée, commencent par un en-tête indiquant la *méthode* de requête.

Ces méthodes, au nombre de 9, sont répertoriées dans le tableau suivant :

Méthode	Action
`GET`	Permet de récupérer le document correspondant à l'URL joint à la requête
`HEAD`	Identique à la requête `GET`, mais le serveur ne renvoie alors que l'en-tête de la réponse
`POST`	Permet de récupérer le document correspondant à l'URL joint à la requête, tout en envoyant des données qui sont requises par cette URL
`OPTIONS`	Demande les options de communication du serveur, permettant ainsi une négociation pour déterminer la communication qui sera la plus adaptée
`TRACE`	Demande à ce que le message de requête revienne au client, ce qui lui permet de voir exactement ce que reçoit le serveur
`PUT`	Permet d'envoyer un document au serveur, qui sera enregistré dans l'URL passée en paramètre, ou modifié s'il existe déjà
`PATCH`	Permet d'envoyer des modifications partielles à un document
`DELETE`	Demande au serveur de supprimer la ressource identifiée dans l'URL de la requête
`CONNECT`	Demande à un mandataire Web, typiquement un proxy, de tunneliser une connexion du client vers le serveur, plutôt que de simplement relayer la demande. Peut être typiquement utilisé pour demander à un proxy d'établir une connexion SSL vers un serveur

Ces différentes méthodes permettent, par exemple, de recevoir des résultats différents pour une même URL.

4. https://tools.ietf.org/html/rfc1945
5. https://tools.ietf.org/html/rfc2616
6. https://tools.ietf.org/html/rfc2068
7. https://tools.ietf.org/html/7540
8. consulter https://en.wikipedia.org/wiki/HTTP/2 pour plus de détails
9. https://en.wikipedia.org/wiki/Internet_Draft

Les en-têtes des clients

Une fois la méthode de requête précisée, les clients peuvent également envoyer des données au serveur, soit en utilisant des *en-têtes* HTTP si ce sont des informations complémentaires sur la requête, soit en utilisant le corps du message si ce sont des documents.

Les informations complémentaires sont envoyées grâce à une composition des en-têtes présentés dans le tableau suivant (un en-tête par ligne, liste non exhaustive) :

En-tête	Description
`Accept`	Précise les types MIME acceptés par le client (voir plus bas))
`Accept-Encoding`	Précise les encodages de caractères acceptés par le client
`Accept-Language`	Précise les langues acceptées par le client
`Host`	Nom de l'hôte (et éventuellement numéro de port) concerné par la requête
`User-Agent`	Informe le serveur sur le client (généralement le navigateur) à l'origine de la requête

L'en-tête `Accept-Language` va, par exemple, être utilisé par le serveur web pour rediriger la visiteuse vers la version du site correspondant à son langage préféré, ou pour afficher directement le site dans sa langue.

NB : on peut tout à fait envoyer des en-têtes forgés ne correspondant à rien comme `X-Sisterhood-of-Karn` (on les préfixe généralement avec `X`, mais cela n'a rien d'obligatoire). Normalement, ces en-têtes ne seront pas traités par le serveur ou le web, mais on peut tout à fait décider de les utiliser (dans les logs, pour une authentification, etc).

Les résultats renvoyés par le serveur Web

Si la requête se conforme à HTTP/0.9, le serveur se contente de renvoyer le document qui a été demandé.

À partir de la version 1.0, il renvoie également d'autres informations avant de renvoyer finalement le document demandé (si celui-ci peut être transmis bien sûr).

La première ligne de la réponse indique la version du protocole utilisé par le serveur Web, ainsi que la valeur de retour. Cette valeur est en 2 parties : un code et un commentaire, correspondant au libellé du code. Ces codes sont rangés par catégories, comme le montre le tableau suivant :

Codes	Catégorie de la réponse
100-199	Information
200-299	Succès de la requête du client
300-399	Redirection de la requête du client
400-499	Requête du client incomplète ou non autorisée
500-599	Erreurs du serveur

Le code 200 représente en particulier un succès de la requête. Les tranches de code décrites comportent environ 60 codes différents [10].

La suite des en-têtes renvoyés par le serveur est une composition des en-têtes suivants, suivant le type de demande et la configuration du serveur (liste non exhaustive) :

En-tête	Description
`Connection`	Précise les options de cette connexion réseau
`Content-Encoding`	Indique le schéma d'encodage (compression généralement) associée au contenu
`Content-Length`	Taille du corps du message

10. pour plus de détails, voir https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

En-tête	Description
`Content-Type`	Décrit le type MIME du contenu (voir plus bas))
`Date`	Date d'émission de la réponse
`Expires`	Précise une date et l'heure après lesquelles le document fourni sera obsolète
`Last-Modified`	Précise la date et l'heure de la dernière modification du document fourni
`Location`	Est associé à une URL vers laquelle le client est redirigé
`Server`	Informations sur le serveur ayant renvoyé la requête

Finalement, le document demandé est renvoyé, si cela est possible bien sûr.

Les types MIME

Les types MIME permettent de préciser le type de documents transmis lors d'une communication.

MIME, signifiant *Multipurpose Internet Mail Extensions*, a tout d'abord été conçu pour décrire le type des pièces jointes à des mails.

Ces types ont par la suite été repris dans HTTP/1.0 pour que les clients aient connaissance du type de document qui leur est renvoyé.

Les types MIME sont décrits par les RFC 1521 [11] et 1522 [12].

Quelques exemples de types MIME, catégorie par catégorie :

- texte : `text/plain`, `text/html`, `text/xml`
- image : `image/gif`, `image/jpg`, `image/png`
- audio : `audio/aiff`, `audio/mp3`, `audio/basic`
- vidéo : `video/mpeg`, `video/x-ms-asf`, `video/mpeg`
- application : `application/pdf`, `application/zip`, `application/x-latex`

C'est ce qui permet à votre navigateur d'afficher directement une image ou de vous proposer de télécharger une archive zip : le navigateur sait interprêter (afficher) un certain nombre de types de document (donc de types MIME), et vous propose de télécharger les autres

Résumé

Un client web va faire une requête à un serveur web en précisant la méthode (`GET`, `POST`...), l'adresse de la ressource souhaitée, la version du protocole HTTP utilisée si la requête est en HTTP/1.0 ou ultérieure et éventuellement des en-têtes relatifs au client (`User-Agent`, par exemple) ou au site voulu (`Host`).

Le serveur web va répondre avec le document dans le cas du protocole HTTP/0.9, précédé de la version du protocole HTTP, un code de retour, un commentaire et éventuellement des en-têtes dans les versions ultérieures du protocole.

Exemple de communication HTTP :

Méthode `GET`, protocole HTTP/0.9, ressource `/` :

```
GET /
```

La réponse sera semblable à :

```
<!DOCTYPE html>
<html lang="en">
    ...
</html>
```

11. https://tools.ietf.org/html/rfc1521
12. https://tools.ietf.org/html/rfc1522

Méthode `GET`, protocole HTTP/1.1, ressource `/`, en-tête `Host` :

```
GET / HTTP/1.1
host: example.org
```

La réponse sera semblable à :

```
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Mon, 08 Nov 2021 20:41:46 GMT
Content-Type: text/html
Content-Length: 35420
Last-Modified: Mon, 08 Nov 2021 11:00:04 GMT
Connection: keep-alive
ETag: "61890334-8a5c"
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="en">
    ...
</html>
```

2 Présentation générale d'Apache

Apache est actuellement second des serveurs Web, derrière Nginx, avec un déploiement d'un peu moins d'un quart de ce type de serveurs.

2.1 Principales caractéristiques d'Apache

Les caractéristiques suivantes sont le minimum qu'on attend de nos jours d'un serveur web :

- Conformité aux standards : Apache est complètement conforme à la norme HTTP/1.1 (le support d'HTTP/2 est apporté par le module `http2`[13]) ;
- Hébergement virtuel : Apache est capable de gérer plusieurs sites Web (relatifs à plusieurs domaines distincts) sur une même machine ;
- Sécurité : par la prise en compte de protocoles de sécurité tels que SSL/TLS.

Ces caractéristiques sont facultatives pour servir des sites web mais appréciables :

- Objets dynamiques partagés : Apache peut charger des modules d'extension de ses fonctionnalités en cours d'exécution et ces modules peuvent être intégrés sous Apache sans avoir à le recompiler ;
- Personnalisation et extensibilité : Apache peut être étendu par des modules programmé en C ou en Perl utilisant l'API Apache, ce qui permet d'étendre ses fonctionnalités pour un besoin particulier ;
- La communauté du libre et en particulier celle de l'*Apache Software Foundation*.

2.2 Installation et exécution d'Apache

Plusieurs solutions existent pour installer Apache :

- l'installer à partir de ses sources, ce qui permet d'obtenir une version optimisée pour une architecture et un besoin particulier ;
- l'installer à partir d'un paquet, dépendant de la distribution utilisée (ou de la plate-forme). Pour l'installer sur une Debian, par exemple : `apt install apache2`.

Le démarrage d'Apache correspond à l'exécution du programme `apache2`[14], soit en ligne de commande,

13. https://httpd.apache.org/docs/current/mod/mod_http2.html
14. selon la distribution ou la méthode d'installation, cela peut être le programme `httpd`

soit via un script de démarrage ou un service systemd, ce qui est le cas le plus fréquent.

Quelques options du programme `apache2` (parmi les nombreuses disponibles) sont très utiles :

- `-v` : obtenir la version d'Apache
- `-V` : identique à l'option `-v`, en ajoutant d'autres informations comme les options de compilation utilisées ou les répertoires par défaut...
- `-l` : liste des modules compilés de façon statique (non désactivables)
- `-M` : liste des modules chargés par Apache
- `-t` : vérifie la syntaxe des fichiers de configuration sans lancer le serveur
- `-f` : utilise le fichier de configuration passé en paramètre plutôt que le fichier de configuration par défaut.

Le fichier principal de configuration d'Apache s'appelle `apache2.conf` sur Debian et les distributions qui en dérivent comme Ubuntu, `httpd.conf` sur Red Hat et les distributions qui en dérivent comme CentOs. Il sera souvent modifié dans la suite de ce cours. Nous garderons le nom d'`apache2.conf` tout au long de ce cours.

3 Présentation générale de Nginx

3.1 Principales caractéristiques de Nginx

En plus des caractéristiques minimales pour un serveur web évoquées plus haut, voici quelques points intéressants à propos de Nginx :

- Pour répondre aux requêtes, Nginx utilise une approche par évènement, tout comme Apache lorsqu'il utilise le module `mpm_event`[15] (Apache peut avoir une approche par *processus*[16] en utilisant le module `mpm_prefork`[17] ou par *processus* et *threads*[18] avec le module `mpm_worker`[19]). Imaginons qu'on vous demande de faire du café et tout de suite après d'aller chercher le courrier : vous pouvez préparer la cafetière, la mettre en marche, attendre devant que le café soit prêt et ensuite aller chercher le courrier, ou alors vous pouvez, une fois la cafetière allumée, aller chercher le courrier et revenir chercher la cafetière une fois que c'est prêt. La deuxième approche est celle de Nginx, ce qui lui permet supporter un grand nombre de connections simultanées avec une empreinte mémoire réduite.
- Avant la version 1.9.11 de Nginx, il n'était pas possible d'activer et de désactiver dynamiquement les modules : les différents modules de Nginx devaient être inclus lors de sa compilation. C'est pourquoi il existait plusieurs paquets fournissant Nginx dans Debian (`nginx`, `nginx-light`, `nginx-full` et `nginx-extras`) : chacun d'eux embarquait plus ou moins de modules. Depuis Debian Stretch (sortie en 2017), ces différents paquets proposent toujours les mêmes modules mais ceux-ci sont fournis par des paquets dédiés dont dépendent les différents paquets de Nginx. Certains modules sont toutefois toujours compilés statiquement et ne sont pas désactivables.
- Support des scripts FastCGI[20], SCGI[21], WSGI[22] mais pas CGI[23] (contrairement à Apache).
- Nginx peut servir de répartiteur de charge (*load balancer*).
- Son système de cache est très appréciable.
- Nginx est aussi un excellent proxy[24] inverse ainsi qu'un proxy mail.

Il est à noter qu'Apache peut faire presque tout ce que Nginx fait (répartiteur de charge, cache, FastCGI...) pour peu qu'on installe et active les modules nécessaires.

15. https://httpd.apache.org/docs/current/mod/event.html
16. https://fr.wikipedia.org/wiki/Processus_(informatique)
17. https://httpd.apache.org/docs/current/mod/prefork.html
18. https://fr.wikipedia.org/wiki/Thread_(informatique)
19. https://httpd.apache.org/docs/current/mod/worker.html
20. https://en.wikipedia.org/wiki/FastCGI
21. https://en.wikipedia.org/wiki/Simple_Common_Gateway_Interface
22. https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface
23. https://fr.wikipedia.org/wiki/Common_Gateway_Interface
24. https://fr.wikipedia.org/wiki/Proxy

Apache étant plus ancien que Nginx (1995 versus 2002) et ayant eu une plus longue histoire de domination du marché des serveurs web, celui-ci dispose de bien plus de modules que Nginx, donc de plus de fonctionnalités.

3.2 Installation et exécution de Nginx

Plusieurs solutions existent pour installer Nginx :

— l'installer à partir de ses sources, ce qui permet d'obtenir une version optimisée pour une architecture et un besoin particulier ;
— l'installer à partir d'un paquet, dépendant de la distribution utilisée. Pour l'installer sur une Debian, par exemple : `apt install nginx`.

Le démarrage de Nginx correspond à l'exécution du programme `nginx`, soit en ligne de commande, soit via un script de démarrage ou un service systemd, ce qui est le cas le plus fréquent.

Quelques options du programme `nginx` sont très utiles :

— `-v` : obtenir la version de Nginx.
— `-V` : identique à l'option `-v`, en ajoutant d'autres informations comme les options de compilation utilisées. Vous pourrez y voir la liste des modules inclus dans votre version de Nginx.
— `-t` :vérifie la syntaxe des fichiers de configuration
— `-s SIGNAL` : permet d'envoyer un signal au serveur nginx lancé. `SIGNAL` peut être :
  — `stop` : ferme brutalement (sans attendre d'avoir servi les requêtes en cours)
  — `quit` : ferme une fois toutes les requêtes traitées
  — `reload` : recharge la configuration
  — `reopen` : rouvre les journaux systèmes

4 Quelques informations utiles

4.1 Les ports d'une machine

Deux services ne peuvent pas écouter en même temps sur la même adresse IP et le même port d'une machine. Dans leur configuration de base sur Debian (et sur la plupart des distributions), il n'est pas possible de faire tourner Apache *et* Nginx en même temps, ceux-ci souhaitant écouter par défaut sur le port 80 de toutes les adresses IP disponibles.

Pour pouvoir tester le contenu de ce cours avec Apache et Nginx, vous pouvez :

— couper l'un avant d'allumer l'autre ;
— les faire tourner dans des machines virtuelles différentes ;
— faire écouter l'un des deux sur un autre port que le port 80 (le port 8080 est généralement un bon choix de port alternatif pour un serveur web).

4.2 Les hôtes virtuels

Wikipédia offre une très bonne explication de ce qu'est l'hébergement virtuel [25] :

> En informatique, l'hébergement virtuel (de l'anglais *virtual hosting* abrégé *vhost*) est une méthode que les serveurs tels que serveurs Web utilisent pour accueillir plus d'un nom de domaine sur le même ordinateur, parfois sur la même adresse IP, tout en maintenant une gestion séparée de chacun de ces noms. Cela permet de partager les ressources du serveur, comme la mémoire et le processeur, sans nécessiter que tous les services fournis utilisent le même nom d'hôte. Le terme hébergement virtuel (*virtual hosting*) est utilisé habituellement

25. https://fr.wikipedia.org/wiki/H%C3%A9bergement_virtuel, consulté le 08 nov. 2020

en référence aux serveurs Web, mais les principes s'appliquent également à d'autres services internet.

Dans notre cas, les hôtes virtuels sont les différents sites web servis par Apache ou Nginx.

4.3 Prise en compte des modifications de configuration

Pour prendre en compte des modifications de configuration, on redémarre le logiciel (ce qui le coupe complètement et le rallume) ou on le recharge (les processus fils ou les *threades* sont coupés et rallumé mais le processus principal reste actif).

À noter : certaines modifications de configuration nécessitent obligatoirement un redémarrage, d'autres ne nécessitent qu'un rechargement. On privilégie le rechargement lorsque cela est possible afin d'éviter une période d'indisponibilité du service.

On doit s'assurer de la validité de la configuration avant de redémarrer ou recharger les services pour éviter une coupure du service (une configuration erronée fera planter le service au redémarrage ou au rechargement). Fort heureusement Apache et Nginx permettent de tester la syntaxe de leurs fichiers de configuration. Si cela ne permet pas de s'assurer qu'on aura bien le comportement souhaité, cela permet de s'assurer que l'on peut redémarrer ou recharger le service sans que celui-ci ne se mette en carafe.

Pour tester la configuration d'Apache :

```
apachectl -t
```

Pour tester la configuration de Nginx :

```
nginx -t
```

Il est nécessaire de prendre dès à présent l'habitude de tester sa configuration avant de redémarrer ou recharger un service. Le plus simple est de chaîner les opérations afin de n'effectuer le redémarrage / le rechargement qu'en cas de validité de la configuration :

Pour **recharger** Apache :

```
apachectl -t && apachectl graceful
```

Pour **redémarrer** Apache :

```
apachectl -t && systemctl restart apache2
```

Pour **recharger** Nginx :

```
nginx -t && nginx -s reload
```

Pour **recharger** Nginx :

```
nginx -t && systemctl restart nginx
```