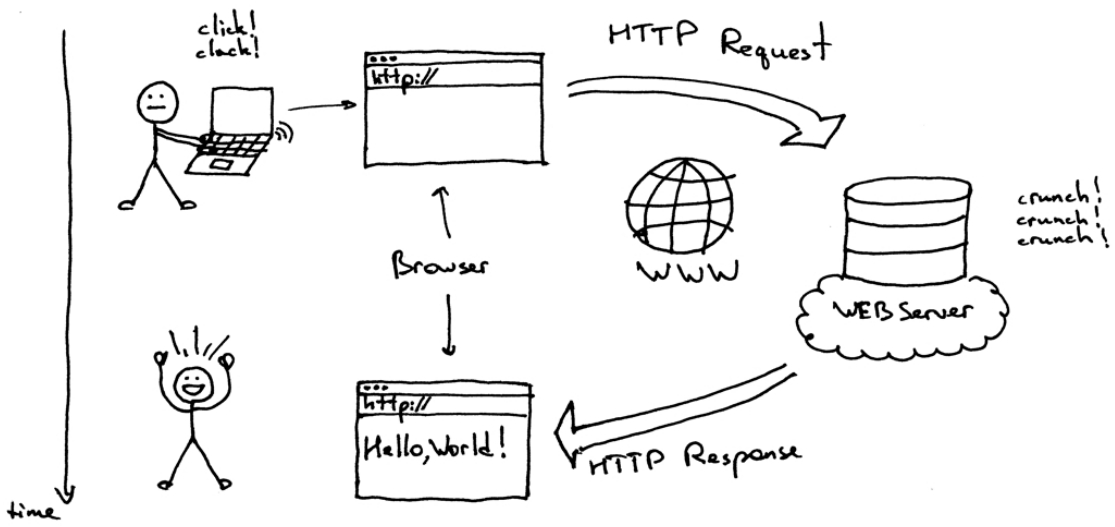
CLIENTS, SERVEURS ET HTTPS

Tu as lu dans l'introduction sur le développement web que le World Wide Web repose entièrement sur une architecture client-serveur et le protocole HTTP.

Rappelle-toi : un **navigateur est un client web** qui **fait appel à des serveurs web** pour recevoir (ou transmettre) des informations.

Le **protocole** utilisé pour la communication entre un client et un serveur sur le web s'appelle **HTTP**(s).



1 (c) Hackernoon

Dans cette partie du cours, nous allons étudier **ce qu'est un navigateur web** au-delà des apparences et découvrir des outils intégrés que monsieur et madame Toulemonde ne connaissent probablement pas. Surtout, tu vas aussi apprendre qu'**il existe d'autres types de clients web** et que client web n'est peut-être pas le terme technique le plus approprié.

Nous étudierons aussi les **différents serveurs web disponibles**, à quoi ils servent et comment effectuer une **installation de base** sur une machine de développement. Pour ce faire, nous utiliserons un package prêt à l'emploi pour Windows. La configuration en sera ainsi facilitée et tu pourras rapidement commencer à **héberger tes premières pages web** et à développer ton premier **site dynamique** sur ta machine de développement (en local).

Pour pouvoir comprendre ces deux mondes, il faut **comprendre ce qui les unit**, à savoir le protocole **HTTP**. Je te propose donc d'en **découvrir les bases**, qui te permettront de **comprendre les règles et conventions qui régissent les échanges** entre navigateurs/clients et serveurs web.

LE MODÈLE CLIENT-SERVEUR

Comme répété plusieurs fois déjà, **HTTP (et donc le web) sont basés sur un modèle client-serveur**.

Pratiquement, cela veut dire que d'une part, il y a un logiciel/programme qui peut **effectuer des requêtes** et que, d'autre part, il y a un logiciel/programme qui **attend une requête** et essaie d'y répondre.

Un serveur est capable de **servir plusieurs clients**. Souvent, il est même capable d'en servir plusieurs **en même temps**.

Bien sûr, la plupart du temps, **le serveur va effectuer un traitement** pour pouvoir répondre à la requête d'un client. Ce traitement effectué par le serveur peut consister simplement à lire le contenu d'un fichier sur son disque et à l'envoyer tel quel au client, tout comme il peut consister à créer un document complexe de toute pièce en interrogeant d'autres serveurs…

Ce qui nous amène à une **notion importante du modèle client-serveur** : une fois qu'ils sont d'accord sur la façon de communiquer, **le client n'a pas besoin de savoir comment le serveur fonctionne** et, inversement, **le serveur n'a pas besoin de savoir comment le client fonctionne**. Chacun sa « popote interne ».

> Pour reprendre l'analogie avec le **restaurant** évoquée dans l'introduction, quand tu commandes un cocktail/plat à un serveur, tu n'as pas besoin de savoir comment celui-ci va préparer ta commande. Tout ce que tu as besoin de savoir, c'est qu'il va t'apporter ta commande… après un certain temps. De la même façon, une fois qu'il t'a apporté ton plat/verre, le serveur n'a pas vraiment besoin de savoir comment tu vas déguster (avec ou sans la paille, cul-sec ou en sirotant, avec les doigts ou des couverts…).

Enfin, avec le protocole HTTP (et pour faire simple), comme **le serveur attend de recevoir une requête** pour renvoyer une réponse, c'est **le client** qui **est à l'initiative**

des échanges. C'est **le client** qui **initie l'échange** en envoyant sa requête à un serveur[1].

LES CLIENTS

Quand on fait référence à un client, on parle en réalité d'**un programme**, que ce soit un **logiciel** avec une **interface graphique** ou en **ligne de commande**, ou encore d'un **processus** qui tourne en tâche de fond, sans aucune interface visible, tels qu'un **service** sous Windows ou un *daemon* sous Linux.

Peu importe la forme qu'il prend, **un client est défini par le fait que c'est lui qui émet les requêtes**. C'est le client qui **demande** des informations ou **initie** des traitements. Dans le cadre du web, par exemple, c'est **le navigateur** qui **fait appel à un serveur pour afficher une page ou une image** quand un utilisateur tape une adresse dans la barre d'adresse ou clique sur un hyperlien.

Le plus souvent, quand on parle de **client web**, on fait référence à un navigateur web. En tant que technicien, il est bien de savoir **qu'il existe en réalité d'autres types de clients web**. En fait, tout type de **programme capable d'effectuer une requête HTTP et d'interpréter une réponse HTTP** peut être considéré comme un client web.

Tant qu'on y est, en réalité, **pour être techniquement correct, on devrait même parler de client HTTP** :

- les programmes de **Google** qui arpentent et indexent les pages web sont en réalité des clients web.
- les logiciels qui utilisent des requêtes HTTP pour récupérer des données en ligne utilisent aussi un client web.
- Il existe des programmes qui permettent d'aspirer des sites web. Ce sont aussi des clients web
- Sous Linux, wget et curl sont des clients web en **ligne de commande**.
- Le programme youtube-dl, qui permet de télécharger des vidéos Youtube, est un client web en ligne de commande.

[1] Il existe aujourd'hui d'autres modèles d'interaction entre un client et un serveur web. Nous les aborderons plus tard dans le cours (pour les curieux, il s'agit entre-autres des web-sockets)

- En PHP, il y a des librairies de fonctions qui permettent d'interagir avec d'autres pages web. Ces librairies proposent des clients http intégrables dans ton code.
- …

LE SERVEUR

Le raisonnement est le même pour un serveur web que pour les clients web. Premièrement, pour être techniquement correct, on devrait parler de **serveur HTTP**.

Ensuite, il faut voir le serveur ~~web~~ http comme un **logiciel**, un **programme** ou un **processus** capable de comprendre une requête http, de réaliser un traitement permettant d'y répondre et, comme on l'a dit, de renvoyer la réponse au client.

Sur une même machine (physique ou virtuelle), il peut logiquement y avoir plusieurs **programmes** qui sont prêts à répondre à une multitude de requêtes de types différents ou, si tu préfères, des programmes **qui implémentent des services et des protocoles complètements différents**. Le http n'est donc pas toujours le seul protocole à être implémenté/disponible sur un serveur.

Comment le client fait-il donc pour **se connecter** non seulement **au bon serveur** mais aussi **au bon programme** qui est capable de répondre à sa demande ?

La réponse réside dans l'URL. Si tu ne te rappelles pas de tous les fragments qui peuvent composer une URL, retourne lire l'introduction du cours. Tu pourras y lire que l'adresse consiste en un protocole, un nom de domaine et un… **port** (facultatif). C'est ce port qui permet de déterminer quel programme va répondre à la requête d'un client.

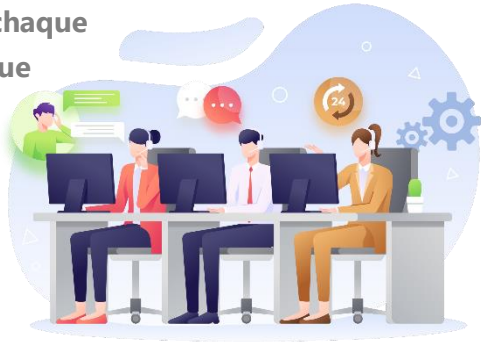
ARRIVÉE À BON PORT

Quand on dit qu'**un serveur** web/**http** « attend une requête d'un client », cela veut dire techniquement qu'il y a **un processus/logiciel qui écoute sur un port TCP spécifique** sur une machine spécifique.

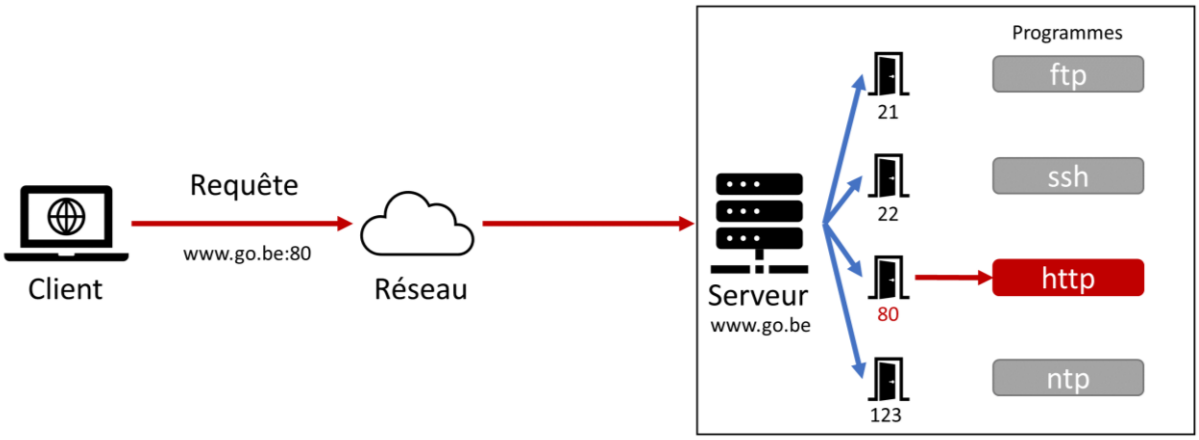
C'est technique et lié au protocole TCP qui est abordé au cours de télématique, mais voici une explication simplifiée pour te permettre de comprendre ce qui se passe.

Il faut voir le **port comme une porte d'entrée** ouverte sur une machine et qui donne accès à un programme spécifique qui, de son côté, est configuré pour « **écouter** » spécifiquement ce qui se passe sur cette porte.

En réalité, même si on utilise globalement le terme serveur pour désigner une machine physique complète (ou une machine virtuelle), rappelle-toi que le terme **serveur fait aussi souvent référence au processus/logiciel capable de prendre en charge des requêtes**. De façon non surprenante, plusieurs programmes peuvent tourner en même temps sur une machine, et faire des choses totalement différentes les uns des autres.

> C'est un peu comme le standard téléphonique d'une grande entreprise. Le numéro de **téléphone est unique et permet d'identifier le central téléphonique de l'entreprise à coup sûr sur le réseau** (téléphonique), Former le numéro de téléphone permet de contacter le serveur téléphonique. [Tu fais le lien avec le web ?] Derrière ce numéro de téléphone se cachent de nombreux services. Dans le cas d'un garage par exemple : la compta, le service mécanique, la prise de rendez-vous, le magasin… bref, pleins de services différents. Et, généralement, **chaque service utilise son propre vocabulaire technique spécialisé**. Inutile de parler de soupapes et d'arbre à cames à la comptabilité. De même, le mécanicien du service mécanique n'y pourra rien si la tva de votre facture n'a pas été imputée correctement en respect de l'article 44.
>
> 
>
> Pour éviter, que les clients ne tombent sur des services qui ne comprennent pas à leurs demandes, les centraux téléphoniques ont mis en place un système de menu. Vous savez, le fameux « tapez 1 », « tapez 2 » ? Avant même de pouvoir formuler votre demande, **vous devez spécifier le numéro qui correspond au service que vous voulez joindre**.

Eh bien, sur Internet[2], c'est pareil. **Il faut toujours spécifier** « le numéro du service » ou, autrement dit, **le numéro de port**. Le port standard défini par le protocole HTTP est **le port 80**. Les clients HTTP sont assez malins pour le savoir et il est donc inutile la plupart du temps d'ajouter ce numéro de port dans les URLs, puisque c'est le port standard, le port **par défaut**. Sans rien dire, **le client http va envoyer ta requête sur le port 80 du serveur identifié par l'adresse** que tu as fournie.



Tu pourras te rendre compte, au fur et à mesure de tes apprentissages, que de nombreux protocoles ont un port standard qui leur a été attribué. D'autres, sans être des standards, peuvent utiliser un des milliers de ports disponibles restants.

Voici quelques ports standards officiellement reconnus (et réservés) :

Number	Assignment
20	File Transfer Protocol (FTP) Data Transfer
21	File Transfer Protocol (FTP) Command Control
22	Secure Shell (SSH) Secure Login
23	Telnet remote login service, unencrypted text messages
25	Simple Mail Transfer Protocol (SMTP) email delivery
53	Domain Name System (DNS) service
67, 68	Dynamic Host Configuration Protocol (DHCP)
80	Hypertext Transfer Protocol (HTTP) used in the World Wide Web
110	Post Office Protocol (POP3)
119	Network News Transfer Protocol (NNTP)
123	Network Time Protocol (NTP)
143	Internet Message Access Protocol (IMAP) Management of digital mail
161	Simple Network Management Protocol (SNMP)
194	Internet Relay Chat (IRC)
443	HTTP Secure (HTTPS) HTTP over TLS/SSL

[2] En fait, sur tout réseau basé sur les protocoles **TCP**/IP

Pour terminer, tu pourras te rappeler que si un programme peut écouter sur plusieurs ports en revanche, **un port ne peut être écouté que par un seul programme** (en théorie, sans tricher).

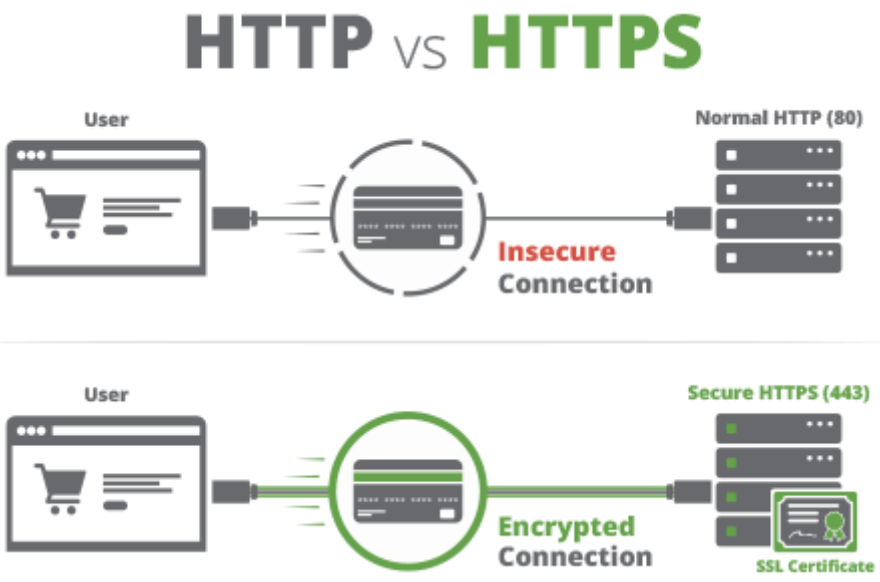
HTTP.S

Tu ne seras pas étonné de lire que HTTP est un **protocole de type requête-réponse (request-response) basé sur le modèle client-serveur** qui standardise et codifie les échanges de données sur le Web.

Il consiste, à la base, en un ensemble de requêtes effectués par des clients HTTP et de réponses renvoyées par un serveur HTTP.

ET HTTPS ?

Le protocole **HTTPS** est, quant à lui, la version sécurisée (**secured**) du protocole HTTP et permet d'utiliser **des certificats digitaux** pour **crypter** les données échangées entre les clients et un serveur et de s'assurer de l'identité / l'authenticité d'un serveur.



Techniquement, HTTPS utilise un autre protocole, **TLS**[3] (Transport Layer Security) pour crypter les communications et ainsi **créer un canal de communication sécurisé** sur un réseau qui est, de par sa conception, non-sécurisé.

[3] Il y a quelques années, c'est le protocole SSL qui était utilisé pour sécuriser HTTPS

On peut dire, de façon simplifiée, que **HTTPS ajoute une couche de sécurité au protocole HTTP** et que, donc, ce qui est valide pour le HTTP reste valide pour le HTTPS. En effet, en dehors des aspects sécurité, HTTPS fonctionne quasiment à l'identique du HTTP.

Alors que HTTP utilise généralement le port 80, **HTTPS utilise le port 443**.

HTTPS a reçu un énorme coup de boost de la part des géants du web ces dernières années, notamment de la part de Google qui a commencé à favoriser les sites en HTTPS dans ses résultats de recherche.

Du coup, **la maîtrise du HTTPS est une chose importante** dans le cadre du développement Web. Dans un premier temps, on se contentera de savoir que tout ce qui est valide pour le HTTP l'est aussi pour le HTTPS. Dans un second temps, au moment de travailler sur le déploiement et l'hébergement d'un site sur un serveur, tu verras **comment configurer gratuitement ton site** pour qu'il soit accessible aussi en HTTPS, en utilisant des certificats fournis par un service appelé **lets's encrypt**.