API Introduction

Auteur : Cyril Vimard

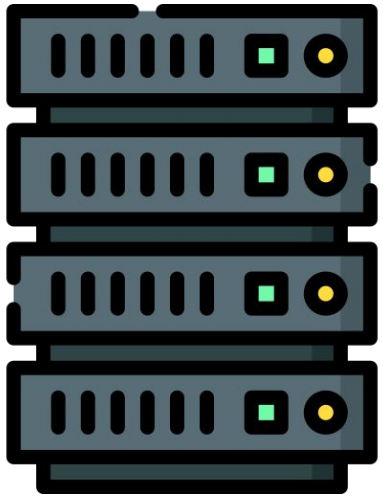
Introduction
serveur web

Qu’est-ce qu’un serveur web ?

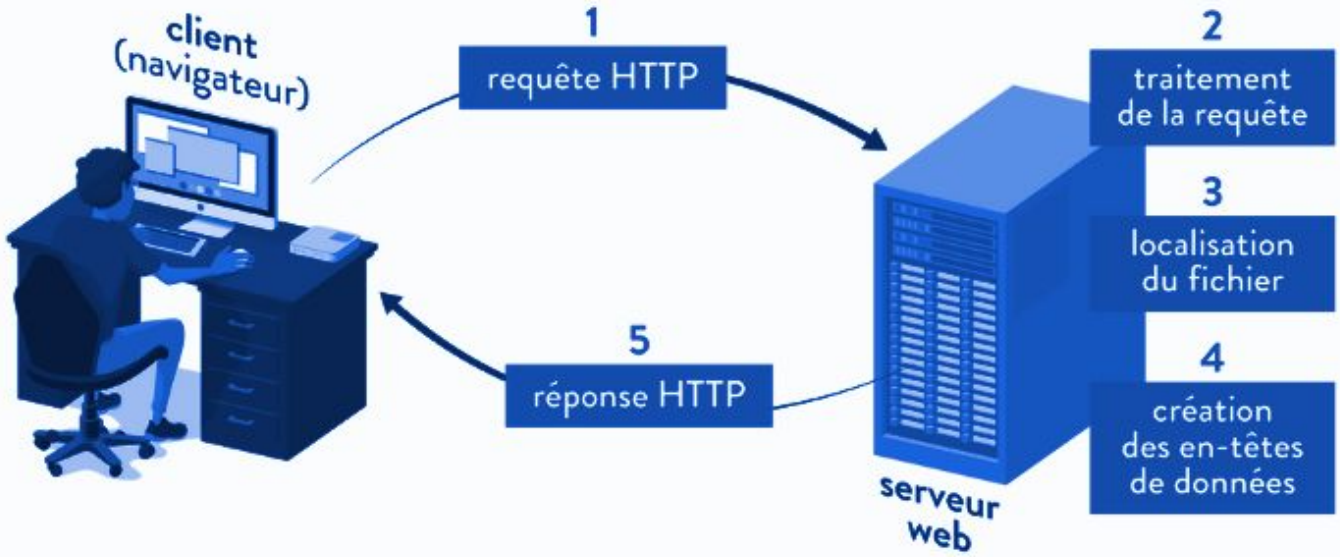
Un serveur web est un ordinateur (ou un logiciel) conçu pour stocker, traiter et délivrer des pages web aux utilisateurs via un réseau (généralement Internet). Le terme "serveur web" peut faire référence à deux choses :

Le matériel (l'ordinateur) : Il s'agit de la machine physique qui héberge le site web et ses fichiers (HTML, images, scripts, etc.) et qui les envoie aux navigateurs des utilisateurs lorsqu'ils en font la demande.

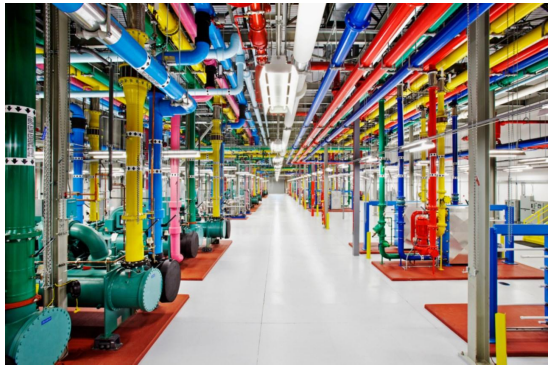
Le logiciel : Un serveur web peut aussi désigner le programme qui gère la communication entre le client (le navigateur de l'utilisateur) et le serveur. Les logiciels de serveurs web populaires incluent Apache, Nginx ou Microsoft IIS. Ils utilisent des protocoles comme HTTP (Hypertext Transfer Protocol) ou HTTPS (HTTP sécurisé) pour échanger des données avec les utilisateurs.

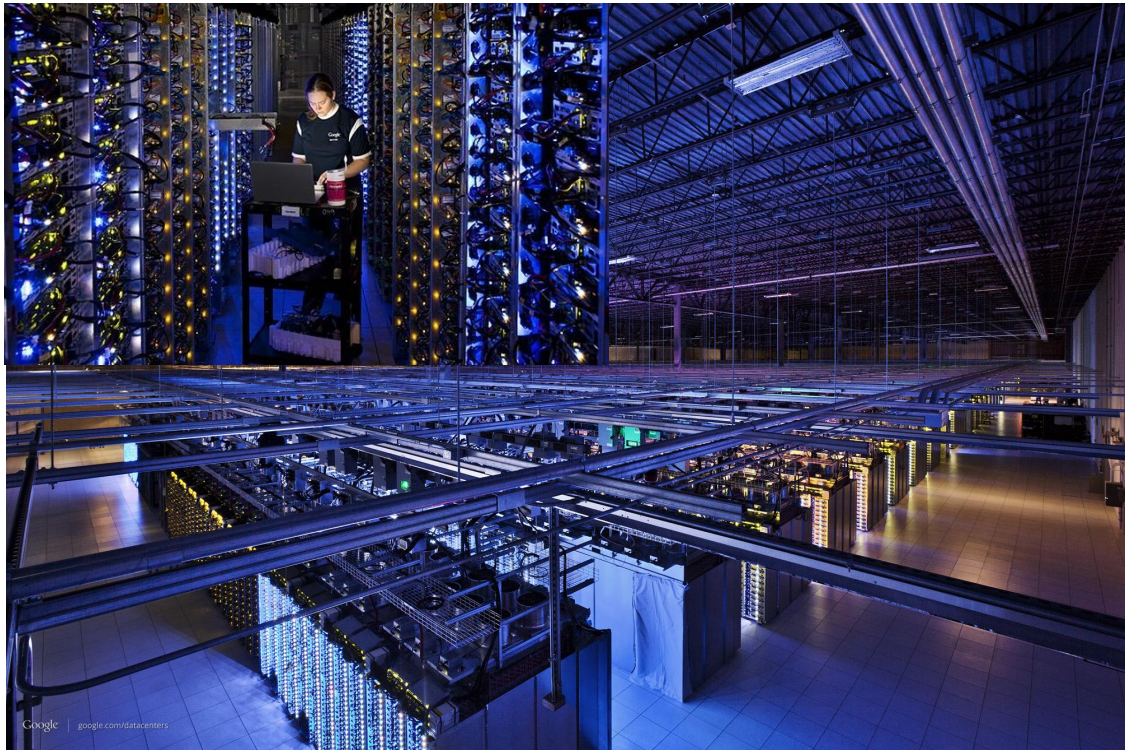


Fonctionnement du serveur web

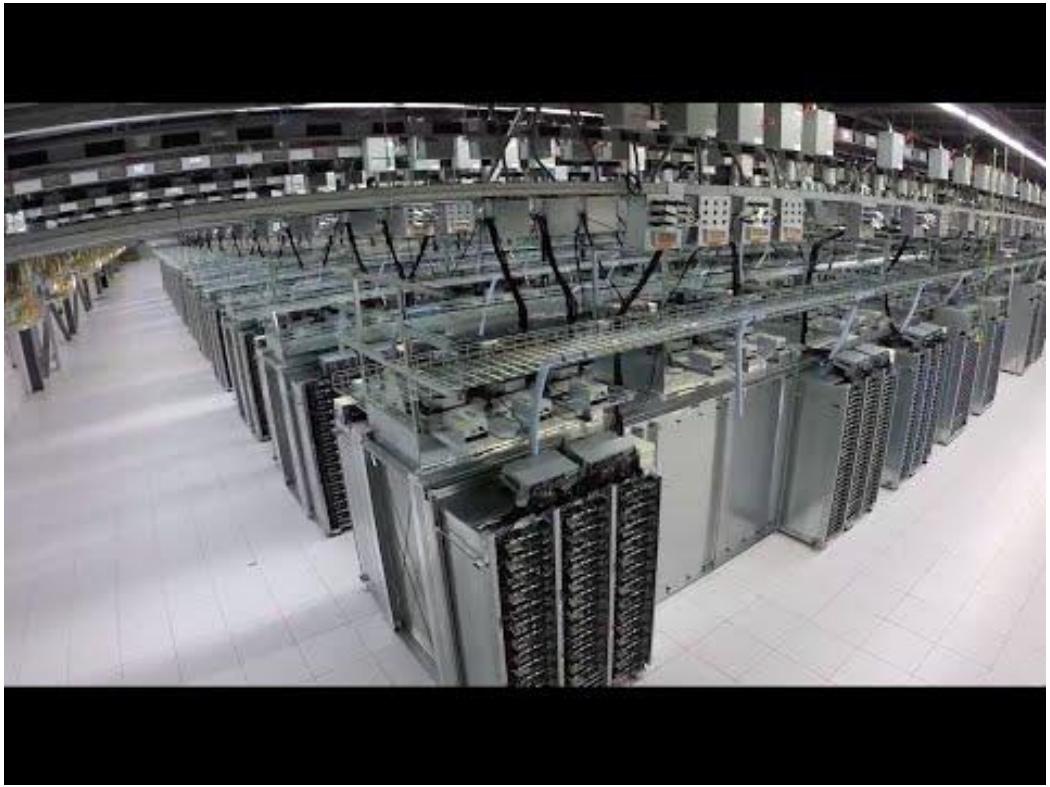


Les centres de données





Les centres de données



Différence entre **client** et **serveur**

Client **VS** Serveur

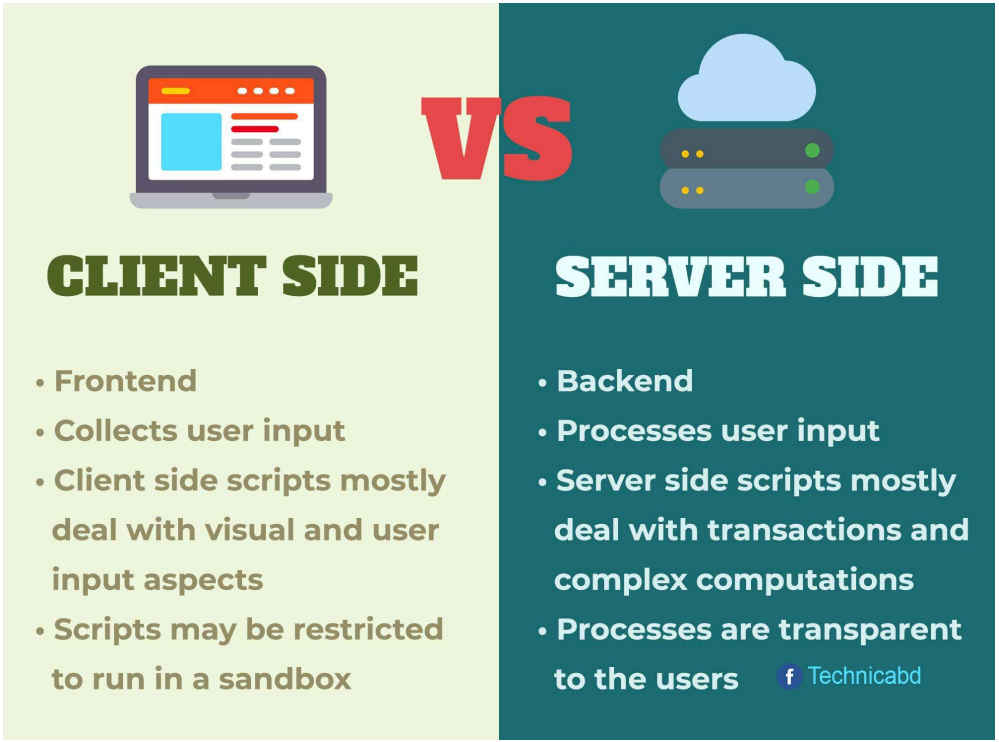
Il est important de noter la différence entre un serveur web et un serveur d'applications.

Serveur web : Son rôle principal est de fournir des pages web statiques ou dynamiques aux utilisateurs.

Serveur d'applications : Il va plus loin en gérant la logique d'entreprise (business logic).

Il fournit des services à des applications plus complexes qui interagissent avec des bases de données, des services web, et des utilisateurs.

Le serveur d'applications peut intégrer un serveur web, mais il a un rôle plus large que simplement servir des pages.



Fonctionnement du serveur web

1/ Requête du client (navigateur) :
L'utilisateur entre une URL dans la barre d'adresse de son navigateur ou clique sur un lien. Le navigateur envoie une requête HTTP au serveur web associé à cette URL.

Par exemple, lorsqu'on tape "**www.example.com**", une requête est envoyée vers le serveur qui héberge le site "**example.com**".

Fonctionnement du serveur web

2/ Traitement de la requête : Le serveur web reçoit la requête et identifie quel contenu est demandé. Si la requête est pour une page HTML statique, le serveur peut simplement envoyer cette page au client.

Si des données dynamiques (générées en temps réel) sont requises, le serveur peut exécuter des scripts (comme PHP ou Python) pour générer le contenu avant de le renvoyer.

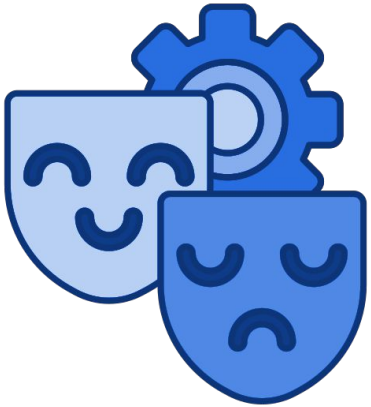
Fonctionnement du serveur web

3/ Réponse du serveur : Une fois que le serveur a traité la requête, il renvoie une réponse HTTP au navigateur.

Cette réponse contient généralement le fichier HTML, mais peut également inclure des images, des feuilles de style (CSS), des scripts JavaScript, etc.

Fonctionnement du serveur web

4/ Affichage pour l'utilisateur : Le navigateur web reçoit la réponse et affiche le contenu (page web) à l'utilisateur.



Rôle d’un serveur Web

Rôle d’un serveur web

1/ Hébergement de contenu : Le serveur web est l'endroit où les fichiers du site web (HTML, CSS, JavaScript, images, etc.) sont stockés. Chaque fois qu'un utilisateur demande une page, le serveur web doit la fournir.

2/ Exécution de scripts : Les serveurs web peuvent également exécuter des scripts côté serveur pour générer des pages dynamiques. Par exemple, lorsqu'un utilisateur se connecte à un site, le serveur peut exécuter un script (souvent en PHP ou Python) pour authentifier l'utilisateur, générer un tableau de bord personnalisé et renvoyer la page.

3/ Sécurité : En plus de simplement fournir des fichiers, un serveur web doit également gérer la sécurité. Il peut être configuré pour restreindre l'accès à certaines ressources (par exemple, en utilisant l'authentification), chiffrer les communications via HTTPS, et surveiller et bloquer les attaques telles que les injections SQL ou les attaques par déni de service (DDoS).

4/ Optimisation des performances : Les serveurs web sont aussi souvent responsables de l'optimisation des temps de réponse. Cela peut inclure la mise en cache des pages pour répondre plus rapidement aux requêtes répétées, la compression des fichiers avant leur envoi pour réduire la bande passante, ou encore la distribution des fichiers via un réseau de diffusion de contenu (CDN).

5/ Gestion des erreurs : Un autre rôle du serveur web est de gérer les erreurs. Si une page n'existe pas, le serveur renverra une erreur 404 ("page non trouvée"). Si une requête est mal formée ou que l'utilisateur n'a pas les permissions nécessaires, le serveur peut renvoyer une autre erreur, comme une erreur 403 (accès interdit) ou 500 (erreur serveur).

Exemples de serveurs web populaires

Apache HTTP Server : C'est l'un des serveurs web les plus utilisés dans le monde. Il est open source, flexible et supporte de nombreux modules pour ajouter des fonctionnalités supplémentaires (ex. PHP, SSL, etc.).

Nginx : Très populaire pour sa légèreté et sa capacité à gérer un grand nombre de connexions simultanées avec une faible consommation de ressources. Il est souvent utilisé en tant que serveur web ou proxy inverse.

Microsoft IIS (Internet Information Services) : Serveur web développé par Microsoft, utilisé principalement dans les environnements Windows. Il est souvent associé avec des technologies Microsoft telles que ASP.NET.

Les différentes technologies utilisées par les serveurs web

Les différentes technologies utilisées par les serveurs web

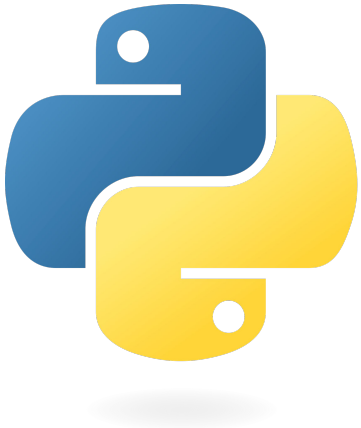
HTML/CSS/JavaScript : Le trident de base pour afficher des pages web.

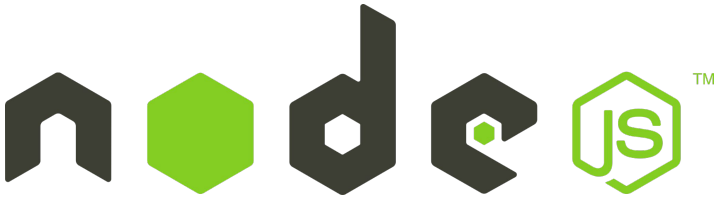
Les fichiers HTML définissent la structure des pages, CSS les styles, et JavaScript le comportement côté client.

Les différentes technologies utilisées par les serveurs web

PHP, Python, Ruby, Node.js ... : Ces langages sont utilisés côté serveur pour générer dynamiquement des pages en fonction des requêtes des utilisateurs.

Par exemple, un script PHP pourrait extraire des informations d'une base de données et les insérer dans une page HTML avant de l'envoyer à l'utilisateur.





Les cas d'utilisation d'un serveur web

Hébergement de sites web

Le cas d'utilisation le plus courant d'un serveur web est l'hébergement de sites web.

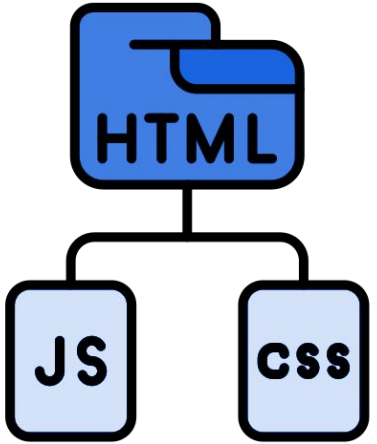
Un serveur web stocke les fichiers d'un site (HTML, CSS, JavaScript, images, etc.) et les envoie aux utilisateurs lorsqu'ils en font la demande via leur navigateur web.

Sites web statiques : Un serveur web renvoie simplement les fichiers HTML et les ressources associées (images, CSS, etc.) à l'utilisateur. Ces sites ne changent pas en fonction de l'utilisateur.

Sites web dynamiques : Le contenu est généré dynamiquement à la demande. Cela peut inclure des interactions avec des bases de données, des utilisateurs ou des API pour afficher un contenu personnalisé (ex : un site e-commerce ou un réseau social).

Exemple :

- **Hébergement statique :** Une entreprise peut héberger son site vitrine (page de présentation).
- **Hébergement dynamique :** Une plateforme comme Facebook génère du contenu pour chaque utilisateur en fonction de ses interactions.



API et services web

Les serveurs web peuvent également être utilisés pour fournir des API (Application Programming Interfaces), qui permettent à différentes applications ou systèmes de communiquer entre eux via des requêtes HTTP.

API RESTful : Utilisent le protocole HTTP pour permettre la communication entre les services. Elles permettent aux applications d'accéder aux données ou aux fonctionnalités d'autres applications (par exemple, une application météo qui récupère les prévisions depuis une API tierce).

API GraphQL : Fournissent une approche plus flexible que REST, permettant aux clients de spécifier exactement quelles données ils veulent recevoir.

Exemple :

- **API d'un service bancaire :** Une banque peut fournir une API permettant à des applications mobiles ou à des systèmes internes de vérifier le solde des comptes ou d'effectuer des transactions.
- **API d'un service de paiement :** PayPal ou Stripe offrent des API pour effectuer des paiements en ligne via des serveurs web.

Hébergement de contenu multimédia

Un serveur web peut être utilisé pour héberger et diffuser des contenus multimédias tels que des images, des vidéos, et de l'audio. Cela inclut les services de streaming.

Vidéos en streaming : Les plateformes comme YouTube ou Netflix utilisent des serveurs web pour diffuser des vidéos aux utilisateurs via un réseau mondial de serveurs.

Stockage de fichiers : Des services comme Dropbox ou Google Drive utilisent des serveurs web pour permettre aux utilisateurs de télécharger ou de consulter des fichiers.

Exemple :

- **Diffusion en continu :** Netflix utilise des serveurs web pour diffuser ses séries et films via une technologie de streaming qui adapte la qualité en fonction de la connexion de l'utilisateur.
- **Stockage de fichiers personnels :** Google Photos héberge les photos et vidéos personnelles des utilisateurs, accessibles via le web.

Services de gestion de base de données

Certains serveurs web sont utilisés pour fournir une interface utilisateur ou des services permettant d'interagir avec une base de données. Cela peut inclure la gestion des données ou l'accès à des informations stockées dans des bases de données relationnelles (comme MySQL) ou non relationnelles (comme MongoDB).

Applications de gestion des données : Des interfaces web qui permettent aux utilisateurs de manipuler des bases de données directement à partir d'un navigateur.

Applications web transactionnelles : Comme les systèmes de réservation de vols ou d'hôtels qui interagissent en temps réel avec des bases de données pour fournir des informations actualisées.

Exemple :

- **Systèmes de gestion de contenu (CMS) :** Un CMS comme WordPress utilise un serveur web pour gérer et afficher le contenu stocké dans une base de données MySQL.
- **Plateformes e-commerce :** Les sites comme Amazon gèrent leurs catalogues de produits et informations de clients via des bases de données dynamiques.

Serveur de fichiers (FTP, WebDAV)

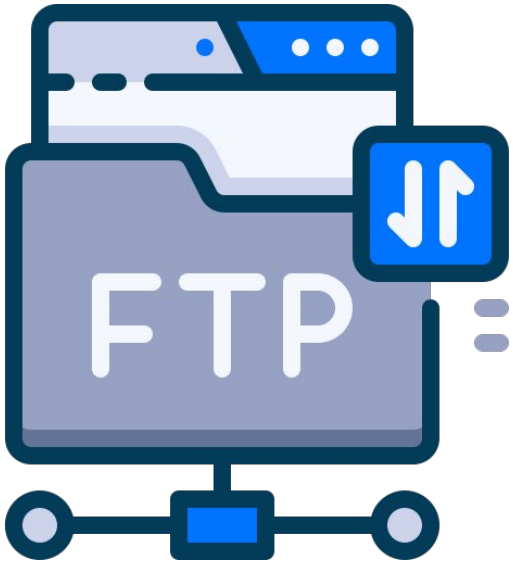
Un serveur web peut être configuré pour servir de serveur de fichiers, permettant aux utilisateurs d'uploader, télécharger ou partager des fichiers à distance. Deux protocoles souvent utilisés dans ce contexte sont FTP (File Transfer Protocol) et WebDAV (Web Distributed Authoring and Versioning).

Hébergement de fichiers : Permettre à des utilisateurs distants d'accéder à des fichiers, de les télécharger ou les mettre à jour.

Collaboration en ligne : Des systèmes comme WebDAV permettent la modification simultanée de documents en ligne par plusieurs utilisateurs.

Exemple :

- **Serveur FTP personnel :** Une entreprise peut utiliser un serveur web pour permettre à ses employés de télécharger et partager des fichiers de travail depuis n'importe où.
- **Cloud personnel :** Des solutions comme Nextcloud utilisent un serveur web pour offrir une plateforme privée de gestion de fichiers.



Proxy Web et Caches

Un serveur web peut agir comme un proxy pour d'autres serveurs ou ressources en ligne. Il intercepte et redirige les requêtes des utilisateurs vers d'autres serveurs, souvent pour améliorer les performances ou la sécurité.

Proxy inverse : Utilisé pour équilibrer la charge et protéger les serveurs en amont en filtrant les requêtes entrantes (ex : NGINX, HAProxy).

Cache HTTP : Un serveur web comme Varnish peut stocker en cache les pages web les plus demandées pour améliorer les performances et réduire la charge sur le serveur principal.

Exemple :

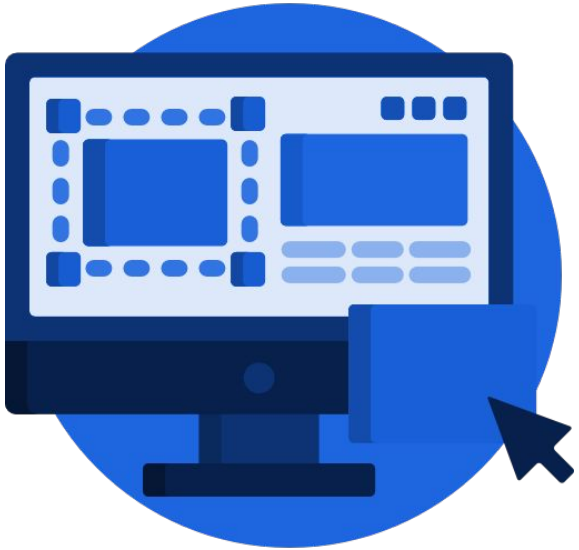
- **Répartition de charge (load balancing) :** Un site de grande envergure comme Amazon ou Google utilise des proxys pour distribuer les requêtes vers différents serveurs afin de gérer des millions d'utilisateurs simultanément.
- **Amélioration des performances avec un cache :** YouTube utilise des serveurs de cache pour stocker temporairement des vidéos populaires et réduire les temps de latence.

Serveurs d’applications Web

Un serveur web peut être intégré avec un serveur d'application pour héberger des applications web complexes. Cela inclut des applications nécessitant des traitements back-end sophistiqués, comme des calculs complexes, des interconnexions avec des bases de données ou des systèmes externes.

Types d'applications hébergées : Applications d’entreprise : ERP, CRM, et autres logiciels d’entreprise souvent déployés via des serveurs web.

Jeux en ligne : Certains serveurs web permettent d'héberger des jeux qui nécessitent des interactions complexes entre les utilisateurs.



Exemple :

- **Application d’entreprise :** Salesforce, un CRM populaire, repose sur des serveurs web pour fournir ses services à des millions d’utilisateurs professionnels.
- **Jeux vidéo en ligne :** Fortnite utilise des serveurs web pour gérer les sessions de jeu, les mises à jour de contenu et les interactions des joueur

Accès à distance et administration

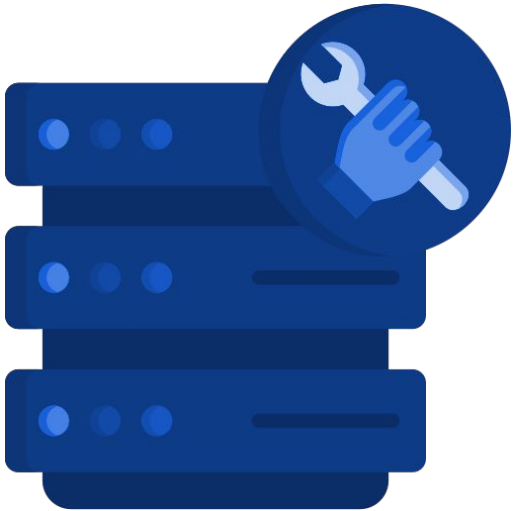
Les serveurs web peuvent être utilisés pour offrir des services d'accès à distance ou d'administration à travers une interface web. Cela inclut des solutions comme les interfaces de gestion de serveurs ou les bureaux à distance.

Panneaux de contrôle de serveur : Comme cPanel ou Plesk, qui permettent aux administrateurs de gérer un serveur web via une interface web.

Bureaux distants : Accès à distance aux machines via des interfaces web sécurisées, comme dans les solutions de bureaux à distance (RDP, VNC).

Exemple :

- **Gestion de serveur :** Un administrateur peut utiliser cPanel pour configurer et gérer des sites web, des bases de données et des utilisateurs sur un serveur.
- **Accès à distance sécurisé :** Des outils comme TeamViewer permettent d'accéder et de contrôler des ordinateurs à distance via une interface web.



http://

Protocole HTTP

HTTP/HTTPS

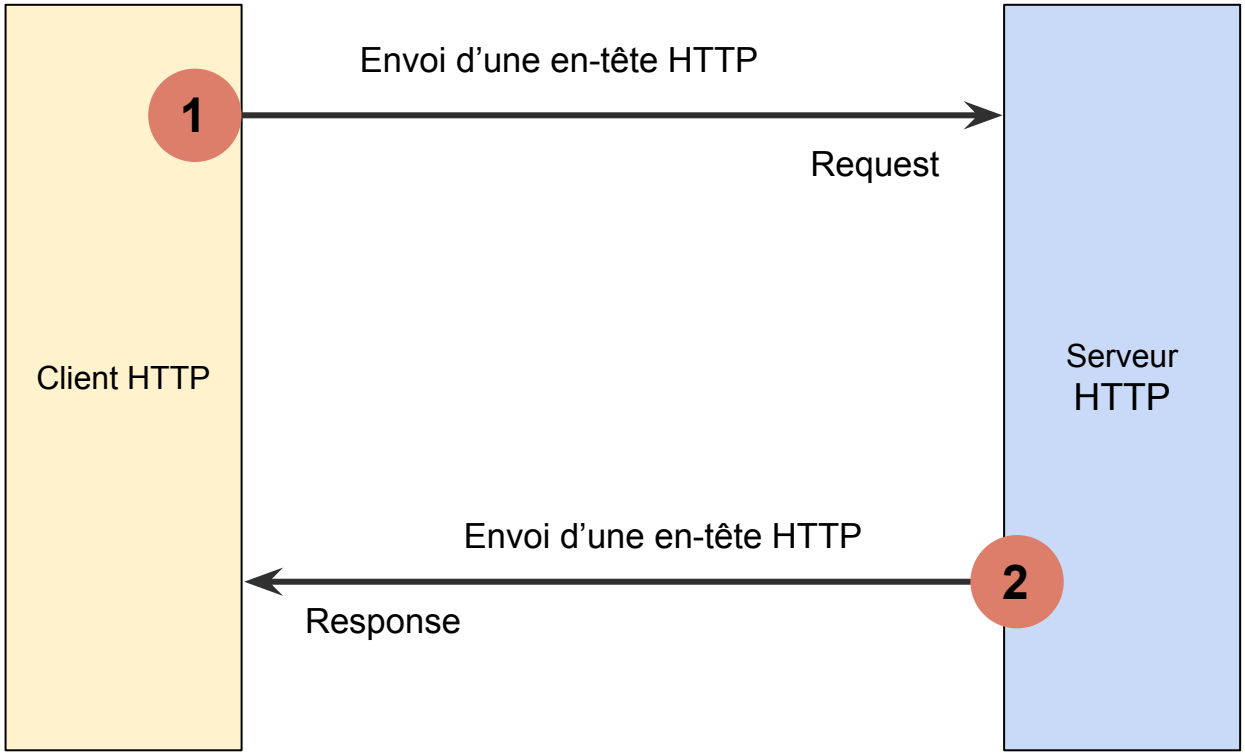
L'HyperText Transfer Protocol, plus connu sous l'abréviation HTTP littéralement (protocole de transfert hypertexte) est un protocole de communication client-serveur développé pour le World Wide Web.

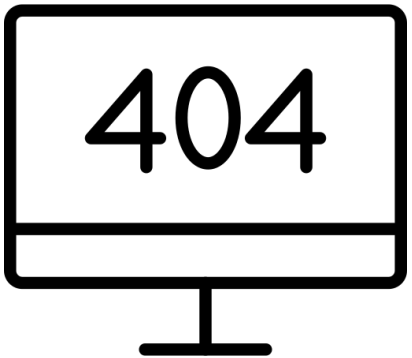
HTTPS (avec S pour secured, soit « sécurisé ») est la variante du HTTP sécurisée par l'usage des protocoles SSL ou TLS.

Dans le protocole HTTP, une méthode est une commande spécifiant un type de requête, c'est-à-dire qu'elle demande au serveur d'effectuer une action.

SSL (Secure Sockets Layer) :, un protocole de sécurisation des échanges sur Internet, devenu TLS (Transport Layer Security) en 2001.

HTTP


Envoi d’une en-tête HTTP
1
Request
Client HTTP
Serveur
HTTP
Envoi d’une en-tête HTTP
2
Response



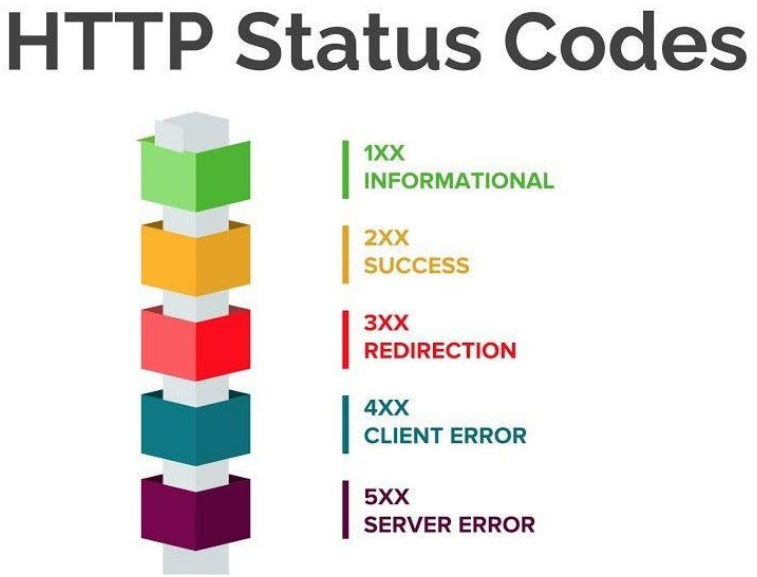
Codes HTTP

Les catégories des codes HTTP

Les codes HTTP permettent de définir le résultat d'une requête ou une erreur du navigateur. Les codes sont principalements adressés pour permettre le traitement automatiques par les clients HTTP. Effectivement le client qui utilise les URI et le protocol HTTP suis la la norme RFC7231 qui régit les codes HTTP pour tout les navigateur utilisant le protocole.

Les principales catégories de code permettent de déterminer : **succès**, **l'informations**, **redirection**, **erreur client** et **erreur serveur**.

2xx : Succès (la requête a été traitée avec succès).
3xx : Redirections (une action supplémentaire est nécessaire pour finaliser la requête).
4xx : Erreurs côté client (problèmes dans la requête envoyée par le client).
5xx : Erreurs côté serveur (le serveur a échoué à traiter une requête valide).



Les sous catégorie des codes HTTP

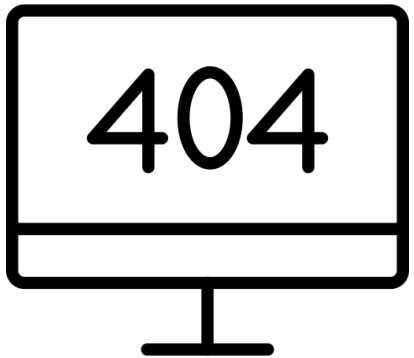
Chaque **code HTTP** donne une indication du résultat de la requête et est important pour diagnostiquer les problèmes ou comprendre comment interagir correctement avec un serveur.

Il est donc important de bien spécifier les codes retour HTTP que le serveur devrait retourner dans sa configuration ou dans l'application qui sera déposé sur le serveur.

Les codes les plus utilisés :

- **200** : succès de la requête
- **301** et **302** : redirection, respectivement permanente et temporaire
- **401** : utilisateur non authentifié
- **403** : accès refusé
- **404** : page non trouvée
- **500** : Erreur interne du serveur
- **504** : Gateway Time-out

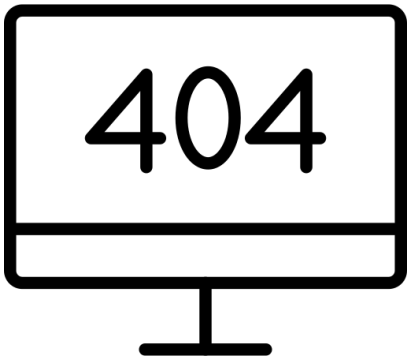
Documentation officiel RFC7231. ici



Codes HTTP : 1XX

Code de catégorie 1XX : Information

Code	Message	Apparition	Signification
100	*Continue*		Attente de la suite de la requête.
101	*Switching Protocols*		Acceptation du changement de protocole.
102	*Processing*	WebDAV RFC 25185,6	Traitement en cours (évite que le client dépasse le temps d'attente limite).
103	*Early Hints*	RFC 82977	(Expérimental) Dans l'attente de la réponse définitive, le serveur renvoie des liens que le client peut commencer à télécharger



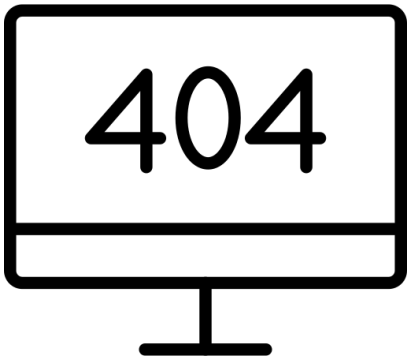
Codes HTTP : 2XX

Code de catégorie 2XX : succès

Code	Message	Apparition	Signification
200	*OK*	RFC 19458	Requête traitée avec succès. La réponse dépendra de la méthode de requête utilisée.
201	*Created*	RFC 19458	Requête traitée avec succès et création d'un document.
202	*Accepted*	RFC 19458	Requête traitée, mais sans garantie de résultat.
203	*Non-Authoritative Information*		Information renvoyée, mais générée par une source non certifiée.
204	*No Content*	RFC 19458	Requête traitée avec succès mais pas d'information à renvoyer.
205	*Reset Content*	RFC 20689	Requête traitée avec succès, la page courante peut être effacée.

Code de catégorie 2XX : succès

206	*Partial Content*	RFC 20689	Une partie seulement de la ressource a été transmise.
207	*Multi-Status*	WebDAV	Réponse multiple.
208	*Already Reported*	WebDAV	Le document a été envoyé précédemment dans cette collection.
210	*Content Different*	WebDAV	La copie de la ressource côté client diffère de celle du serveur (contenu ou propriétés).
226	*IM Used*	RFC 322910	Le serveur a accompli la requête pour la ressource, et la réponse est une représentation du résultat d'une ou plusieurs manipulations d'instances appliquées à l'instance actuelle.



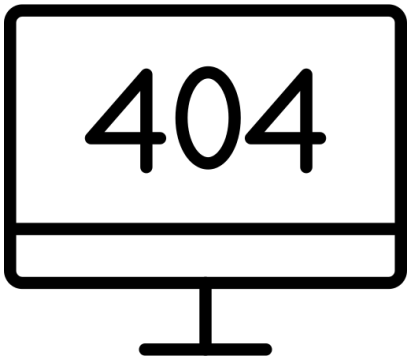
Codes HTTP : 3XX

Code de catégorie 3XX : Redirection

300	*Multiple Choices*	RFC 19458	L'URI demandée se rapporte à plusieurs ressources.
301	*Moved Permanently*	RFC 19458	Document déplacé de façon permanente.
302	*Found*	RFC 19458	Document déplacé de façon temporaire.
303	*See Other*	RFC 20689	La réponse à cette requête est ailleurs.
304	*Not Modified*	RFC 19458	Document non modifié depuis la dernière requête.
305	*Use Proxy* (depuis HTTP/1.1)	RFC 20689	La requête doit être ré-adressée au proxy.

Code de catégorie 3XX : Redirection

306	(inutilisé)	RFC 261611	La RFC 261611 indique que ce code inutilisé est réservé, car il était utilisé dans une ancienne version de la spécification. Il signifiait « Les requêtes suivantes doivent utiliser le proxy spécifié » 12.
307	*Temporary Redirect*		La requête doit être redirigée temporairement vers l'URI spécifiée sans changement de méthode13.
308	*Permanent Redirect*		La requête doit être redirigée définitivement vers l'URI spécifiée sans changement de méthode14.
310	*Too many Redirects*		La requête doit être redirigée de trop nombreuses fois, ou est victime d'une boucle de redirection.



Codes HTTP : 4XX

Code de catégorie 4XX : Erreur du client HTTP

400	*Bad Request*	RFC 19458	La syntaxe de la requête est erronée.
401	*Unauthorized*	RFC 19458	Une authentification est nécessaire pour accéder à la ressource.
402	*Payment Required*	RFC 20689	Paiement requis pour accéder à la ressource.
403	*Forbidden*	RFC 19458	Le serveur a compris la requête, mais refuse de l'exécuter. Contrairement à l'erreur 401, s'authentifier ne fera aucune différence. Sur les serveurs où l'authentification est requise, cela signifie généralement que l'authentification a été acceptée mais que les droits d'accès ne permettent pas au client d'accéder à la ressource.
404	*Not Found*	RFC 19458	Ressource non trouvée.

Code de catégorie 4XX : Erreur du client HTTP

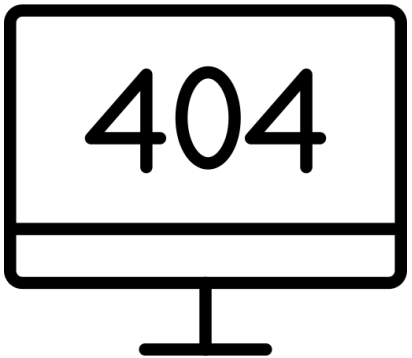
405	*Method Not Allowed*	RFC 20689	Méthode de requête non autorisée.
406	*Not Acceptable*	RFC 20689	La ressource demandée n'est pas disponible dans un format qui respecterait les en-têtes « Accept » de la requête.
407	*Proxy Authentication Required*	RFC 20689	Accès à la ressource autorisé par identification avec le proxy.
408	*Request Time-out*	RFC 20689	Temps d'attente d'une requête du client, écoulé côté serveur. D'après les spécifications HTTP : « Le client n'a pas produit de requête dans le délai que le serveur était prêt à attendre. Le client PEUT répéter la demande sans modifications à tout moment ultérieur »15.
409	*Conflict*	RFC 20689	La requête ne peut être traitée à la suite d'un conflit avec l'état actuel du serveur.

Code de catégorie 4XX : Erreur du client HTTP

410	*Gone*	RFC 20689	La ressource n'est plus disponible et aucune adresse de redirection n'est connue.
411	*Length Required*	RFC 20689	La longueur de la requête n'a pas été précisée.
412	*Precondition Failed*	RFC 20689	Préconditions envoyées par la requête non vérifiées.
413	*Request Entity Too Large*	RFC 20689	Traitement abandonné dû à une requête trop importante.
414	*Request-URI Too Long*	RFC 20689	URI trop longue.
415	*Unsupported Media Type*	RFC 20689	Format de requête non supporté pour une méthode et une ressource données.
416	*Requested range unsatisfiable*		Champs d'en-tête de requête « *range* » incorrect.
417	*Expectation failed*		Comportement attendu et défini dans l'en-tête de la requête insatisfaisante.

Code de catégorie 4XX : Erreur étendu du serveur Ngnix

Code	Message	Apparition	Signification
444	*No Response*	Nginx	Indique que le serveur n'a retourné aucune information vers le client et a fermé la connexion.
495	*SSL Certificate Error*	Nginx	Une extension de l'erreur 400 Bad Request, certificat invalide.
496	*SSL Certificate Required*	Nginx	Une extension de l'erreur 400 Bad Request, certificat client requis.
497	*HTTP Request Sent to HTTPS Port*	Nginx	Une extension de l'erreur 400 Bad Request, utilisée lorsque le client envoie une requête HTTP vers le port 443 normalement destiné aux requêtes HTTPS.
498	*Token expired/invalid*	Nginx	Le jeton a expiré ou est invalide.
499	*Client Closed Request*	Nginx	Le client a fermé la connexion avant de recevoir la réponse.



Codes HTTP : 5XX

Code de catégorie 5XX : Erreur côté serveur

500	*Internal Server Error*	RFC 19458	Erreur interne du serveur.
501	*Not Implemented*	RFC 19458	Fonctionnalité réclamée non supportée par le serveur.
502	*Bad Gateway* ou *Proxy Error*	RFC 19458	En agissant en tant que serveur proxy ou passerelle, le serveur a reçu une réponse invalide depuis le serveur distant.
503	*Service Unavailable*	RFC 19458	Service temporairement indisponible ou en maintenance.
504	*Gateway Time-out*	RFC 20689	Temps d'attente d'une réponse d'un serveur à un serveur intermédiaire écoulé.
505	*HTTP Version not supported*	RFC 20689	Version HTTP non gérée par le serveur.

Code de catégorie 5XX : Erreur côté serveur

506	*Variant Also Negotiates*	RFC 229523	Erreur de négociation. *Transparent content negotiation*.
507	*Insufficient storage*	WebDAV	Espace insuffisant pour modifier les propriétés ou construire la collection.
508	*Loop detected*	WebDAV	Boucle dans une mise en relation de ressources (RFC 584224).
509	*Bandwidth Limit Exceeded*		Utilisé par de nombreux serveurs pour indiquer un dépassement de quota.
510	*Not extended*	RFC 277425	La requête ne respecte pas la politique d'accès aux ressources HTTP étendues.
511	*Network authentication required*	RFC 658520	Le client doit s'authentifier pour accéder au réseau. Utilisé par les portails captifs pour rediriger les clients vers la page d'authentification.

http://

Verbes HTTP (CRUD)

Verbe HTTP & CRUD

Les requêtes HTTP sont souvent basé sur ce qu'on appel **les verbes HTTP**.

Effectivements ces requêtes et réponses HTTP sont basées sur un ensemble de verbes (ou méthodes) HTTP, qui spécifient l'action à réaliser par le serveur.

Le CRUD est une norme que l'on applique en général à une base de données pour permettre d'en manipuler son contenu.

Le CRUDS doit donc s'appliquer à chaque ROUTES de l'API pour les opération suivants : Create, Read, Update, Delete et Search.

Verbes HTTP : **GET**

Verbe HTTP : **GET**

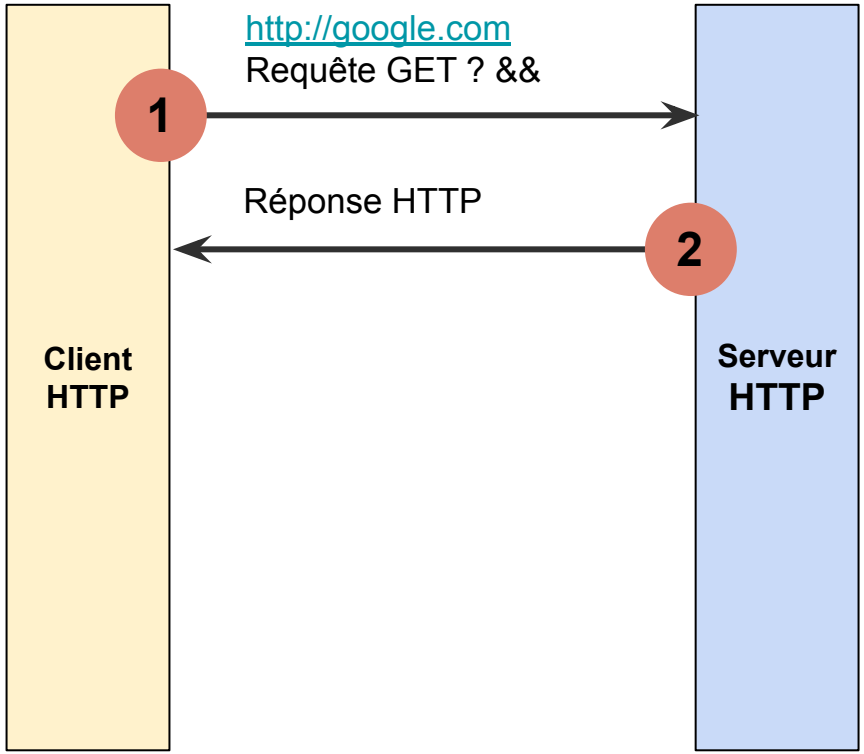
GET Définition : Le verbe GET est utilisé pour demander une ressource spécifique depuis le serveur.

Cette méthode ne modifie pas l'état du serveur ; elle récupère simplement des données.

Exemple d'utilisation : Si vous tapez une URL dans un navigateur, celui-ci envoie une requête GET pour récupérer la page web demandée.

Caractéristiques :

- **Idempotent** : Une requête GET répétée donnera toujours le même résultat (tant que la ressource demandée n'a pas été modifiée).
- **Sécurisée** : Elle ne change pas les données du serveur.



http://

Verbes HTTP :
POST

Verbe HTTP : **POST**

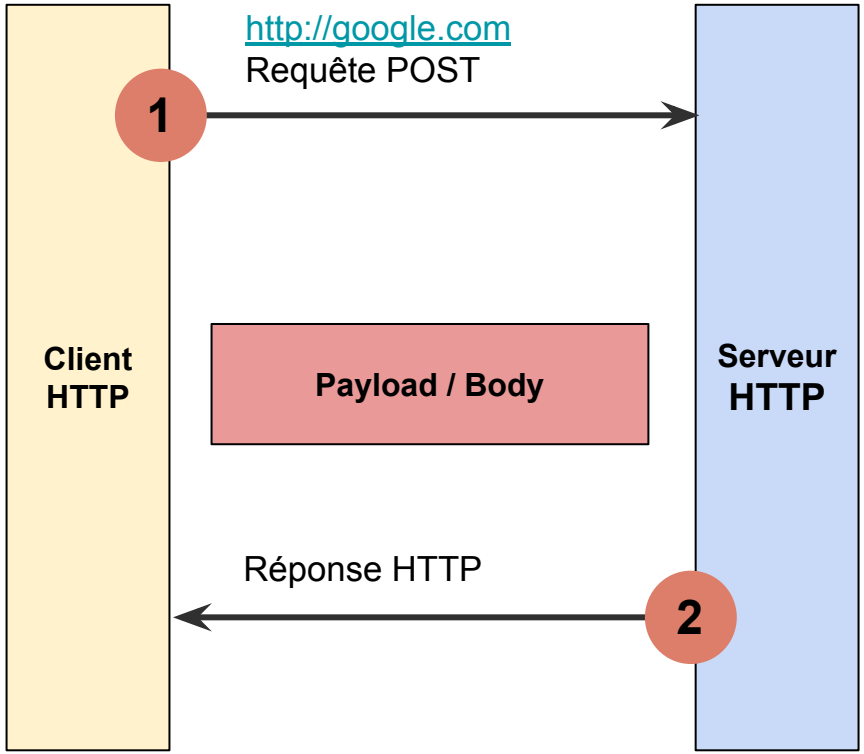
POST Définition : Le verbe **POST** est utilisé pour envoyer des données au serveur, souvent dans le but de créer une ressource.

Contrairement à GET, cette méthode modifie l'état du serveur.

Exemple d'utilisation : Lorsque vous remplissez un formulaire d'inscription sur un site web et que vous cliquez sur "Soumettre", une requête POST est envoyée au serveur avec les informations du formulaire.

Caractéristiques :

- **Non-idempotent :** Refaire la même requête POST peut produire des effets différents (par exemple, soumettre deux fois un formulaire peut créer deux entrées).
- **Utilisé pour :** L'envoi de données complexes, comme des fichiers ou des formulaires.



http://

Verbes HTTP : **PUT / PATCH**

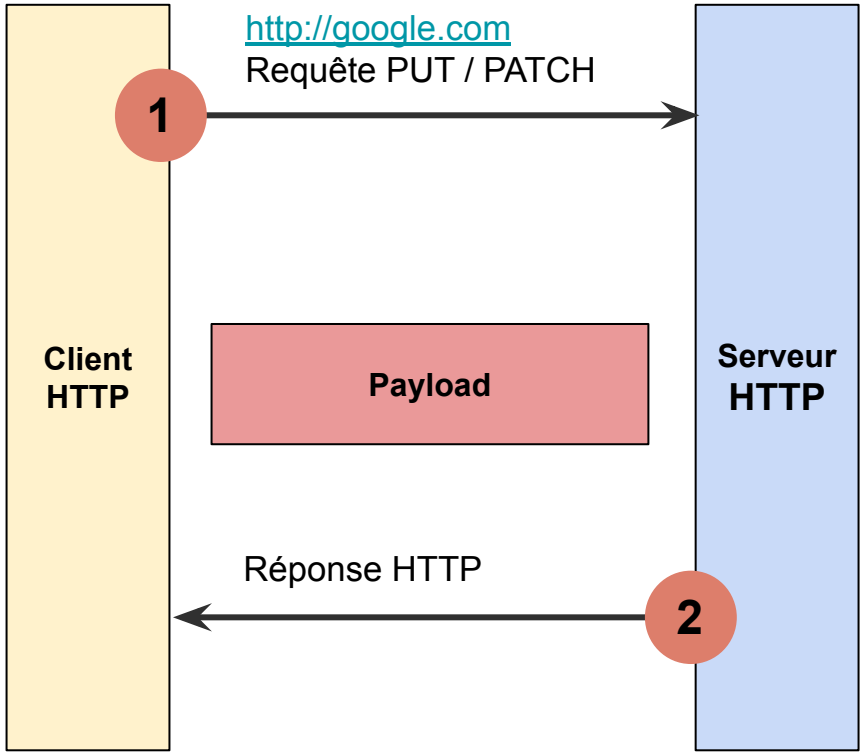
Verbe HTTP : **PUT / PATCH**

PUT / PATCH Définition : La méthode **PUT / PATCH** est utilisée pour envoyer une ressource spécifique au serveur. Elle remplace ou crée la ressource sous l'URI indiqué. Si la ressource existe, elle est mise à jour ; sinon, elle est créée.

Exemple d'utilisation : Si vous mettez à jour votre profil utilisateur sur un site web, une requête PUT peut être utilisée pour remplacer les anciennes informations par les nouvelles.

Caractéristiques :

- **Idempotent** : Faire plusieurs fois la même requête PUT aura le même effet.
- **Utilisé pour** : La mise à jour ou la modification d'une ressource spécifique.



Verbes HTTP : **DELETE**

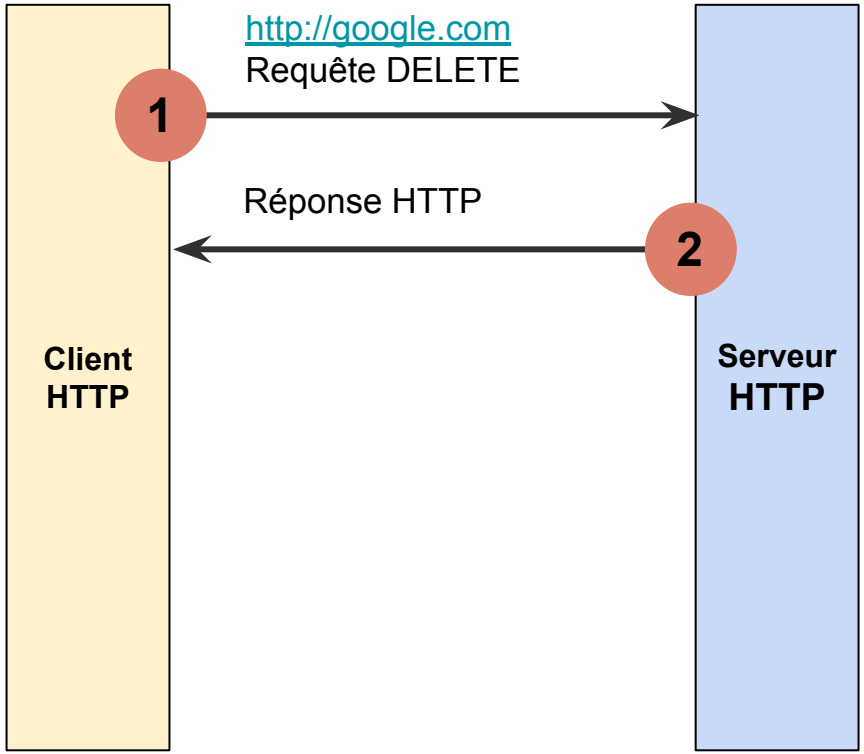
Verbe HTTP : **DELETE**

Définition : Le verbe DELETE est utilisé pour demander la suppression d'une ressource spécifique sur le serveur.

Exemple d'utilisation : Lorsque vous supprimez un compte ou un article de blog, une requête DELETE est généralement envoyée au serveur.

Caractéristiques :

- **Idempotent** : Si vous envoyez plusieurs requêtes DELETE pour la même ressource, elle sera supprimée (ou déjà supprimée), et le résultat restera le même.
- **Utilisé pour :** Supprimer des ressources.



http://

Verbes HTTP : **HEAD**

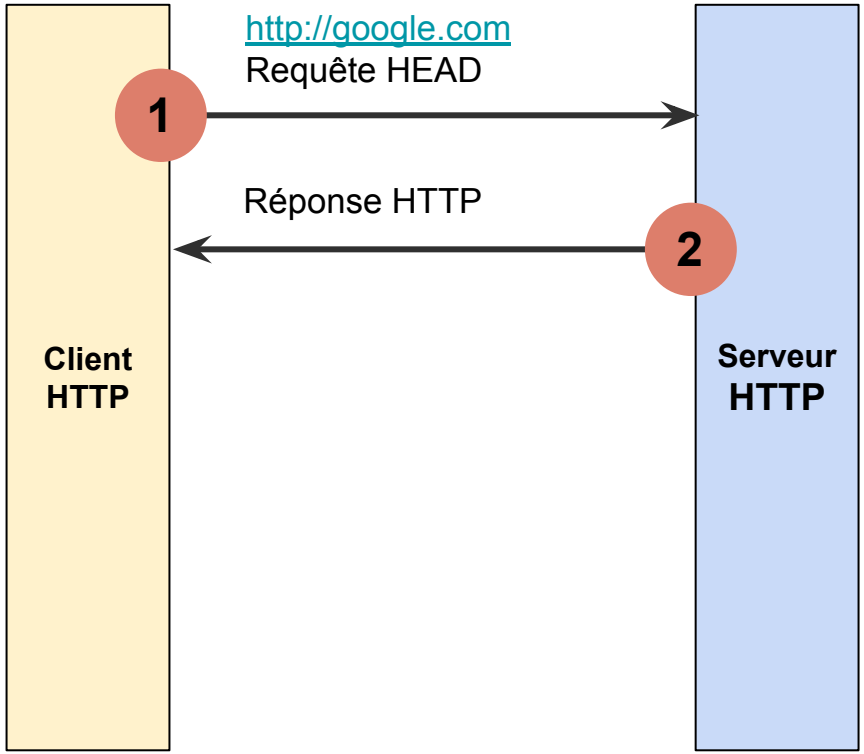
Verbe HTTP : **HEAD**

Définition : HEAD fonctionne de manière similaire à GET, mais il ne renvoie que les en-têtes de la réponse, sans le corps. Il est souvent utilisé pour vérifier si une ressource existe ou obtenir des informations comme la taille du fichier ou la date de dernière modification, sans télécharger la ressource entière.

Exemple d'utilisation : Vérifier si une page existe sans télécharger son contenu.

Caractéristiques :

- **Idempotent** : Comme GET, elle est idempotente.
- **Sécurisée** : Ne change pas les données sur le serveur.



Verbes HTTP : **OPTION**

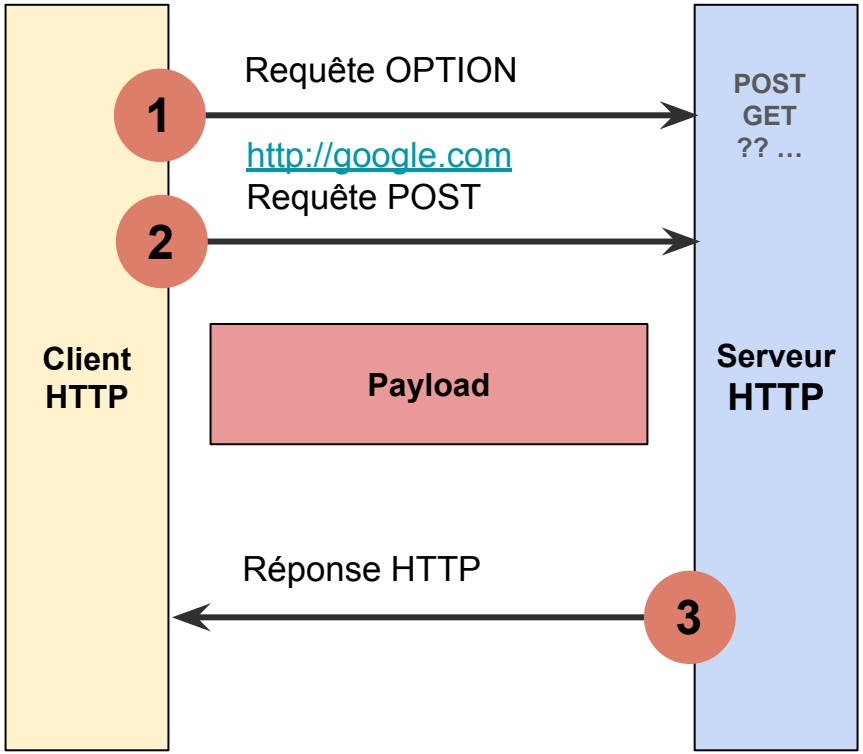
Verbe HTTP : **OPTION**

Définition : Le verbe OPTIONS est utilisé pour demander au serveur quelles méthodes HTTP sont supportées pour une ressource spécifique. Cette requête est souvent utilisée dans le cadre des **CORS (Cross-Origin Resource Sharing)**.

Exemple d'utilisation : Lorsque le navigateur d'un utilisateur envoie une requête **OPTIONS** pour vérifier quelles méthodes sont autorisées avant d'envoyer une requête **POST** provenant d'un domaine différent.

Caractéristiques :

- **Idempotent** : Faire la requête plusieurs fois ne change rien.



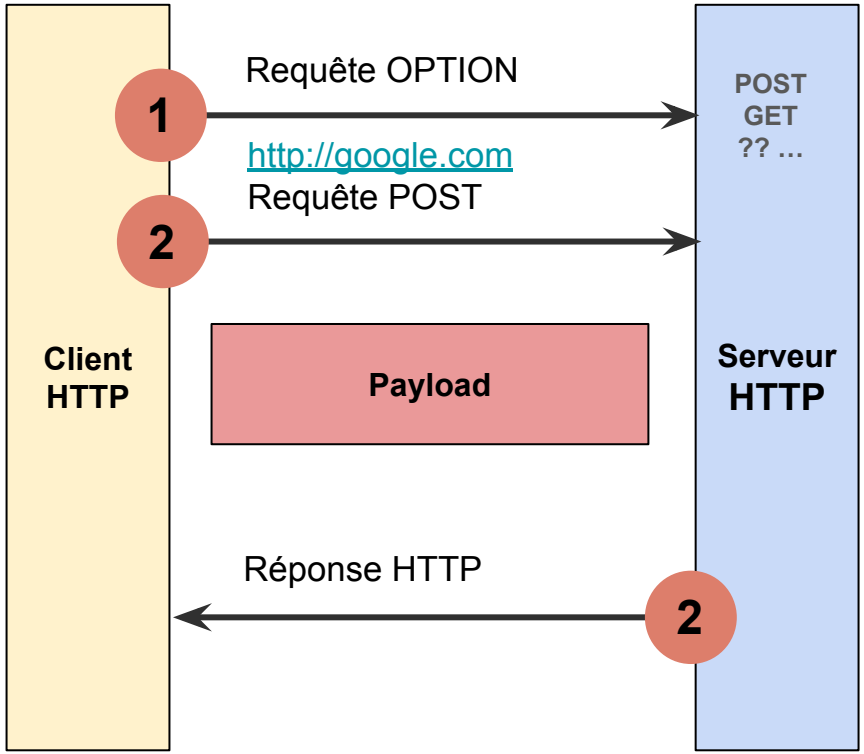
Autres verbes HTTP
moins courants
TRACE / CONNECT

Autres verbes HTTP moins courants TRACE / CONNECT

Bien que les verbes mentionnés ci-dessus soient les plus couramment utilisés, il existe d'autres méthodes HTTP moins connues mais toujours utiles dans des cas spécifiques :

TRACE : Permet de suivre le chemin exact emprunté par une requête jusqu'au serveur. Il est souvent utilisé pour le débogage.

CONNECT : Utilisé pour établir un tunnel pour la communication bidirectionnelle, notamment dans les connexions HTTPS sécurisées. Il est plus utilisé pour les proxys que pour les interactions web classiques.



Comparaison des méthodes HTTP

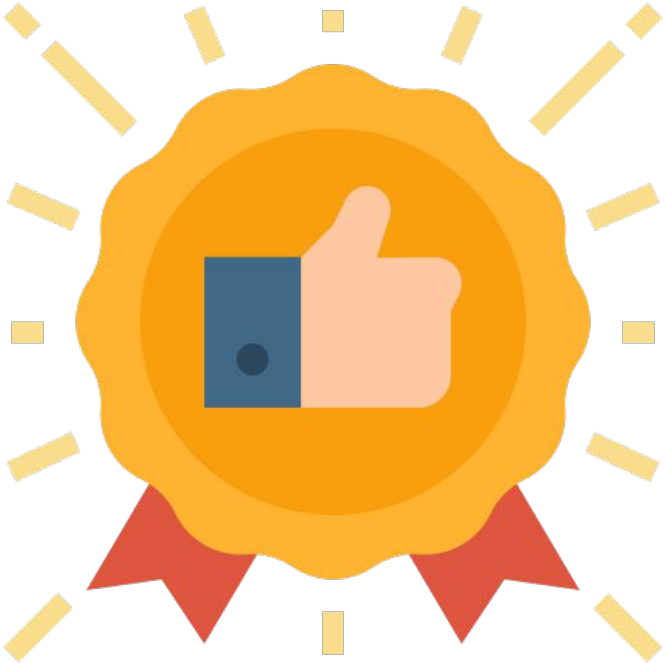
Méthode	Action	Idempotente	Sécure	Corps de requête
GET	Récupérer une ressource	Oui	Oui	Non
POST	Créer ou envoyer des données	Non	Non	Oui
PUT	Remplacer une ressource	Oui	Non	Oui
DELETE	Supprimer une ressource	Oui	Non	Non (en général)
PATCH	Mise à jour partielle	Non	Non	Oui
HEAD	Récupérer les en-têtes	Oui	Oui	Non
OPTIONS	Vérifier les méthodes disponibles	Oui	Oui	Non

Bonnes pratiques dans l'utilisation des verbes HTTP

Utiliser les verbes correctement : Chaque verbe HTTP a un rôle bien défini. Par exemple, n'utilisez pas POST pour récupérer des données (ce rôle appartient à GET).

Idempotence : Les méthodes idempotentes (GET, PUT, DELETE) doivent être utilisées lorsque vous voulez vous assurer que plusieurs appels n'auront pas d'effets supplémentaires indésirables.

Sécuriser les méthodes sensibles : Les requêtes POST, PUT, DELETE, et PATCH doivent être correctement sécurisées, notamment avec des systèmes d'authentification et d'autorisation, car elles peuvent modifier les ressources du serveur.



DNS : Domain **N**ame **S**ystem

DNS (Domain Name System)

Le DNS (Domain Name System) est un système fondamental dans le fonctionnement d'Internet, qui permet de traduire les noms de domaine lisibles par les humains (comme www.google.com) en adresses IP numériques que les machines utilisent pour communiquer (comme 216.58.214.14). Sans DNS, les utilisateurs devraient saisir des adresses IP pour accéder aux sites web, ce qui serait beaucoup moins pratique.

Le Domain Name System (DNS) est un système distribué qui permet d**e faire correspondre un nom de domaine à une adresse IP**.

Il est souvent comparé à un **"annuaire téléphonique"** d'Internet, dans lequel les noms de domaine sont les noms des personnes et les adresses IP sont leurs numéros de téléphone.

Le rôle du DNS est d'aider les ordinateurs à trouver et à se connecter aux sites web, serveurs, ou autres ressources sur Internet en transformant les noms de domaine en adresses IP.

- **Nom de domaine** : Un nom de domaine est un identifiant facile à mémoriser (comme example.com) qui est utilisé pour représenter une ressource web.
- **Adresse IP** : Une adresse IP est un numéro unique attribué à chaque appareil connecté à Internet. Par exemple, une adresse IPv4 ressemble à 192.168.1.1.

Structure du DNS : Root

Le DNS a une structure hiérarchique et distribuée. Cette hiérarchie est divisée en plusieurs niveaux :

A/ Le domaine racine (Root)

Le sommet de la hiérarchie DNS est la racine. C'est un ensemble de serveurs de noms (appelés serveurs racine) qui connaissent les adresses des serveurs DNS pour tous les domaines de premier niveau (comme .com, .org, .fr). Il est représenté par un simple point "." dans la structure DNS, mais il est généralement omis lors de l'écriture des noms de domaine.

Exemple : pour www.example.com, la racine est représentée par le "." final, mais il est généralement omis : www.example.com.

Structure du DNS : Niveau TLD

B/ Domaines de premier niveau (TLD)

Le niveau suivant dans la hiérarchie DNS est celui des TLD (Top-Level Domains) ou domaines de premier niveau. Ce sont des extensions comme .com, .org, .fr, .gov, etc.

Les TLD sont gérés par des organisations spécifiques (ex : ICANN gère une partie des TLD génériques comme .com, .net, .org). Il existe plusieurs types de TLD :

gTLD (Generic TLD) : Ce sont des domaines génériques comme .com, .org, .net.
ccTLD (Country Code TLD) : Ce sont des domaines spécifiques à un pays, comme .fr pour la France, .uk pour le Royaume-Uni, ou .jp pour le Japon.

C/ Domaines de second niveau

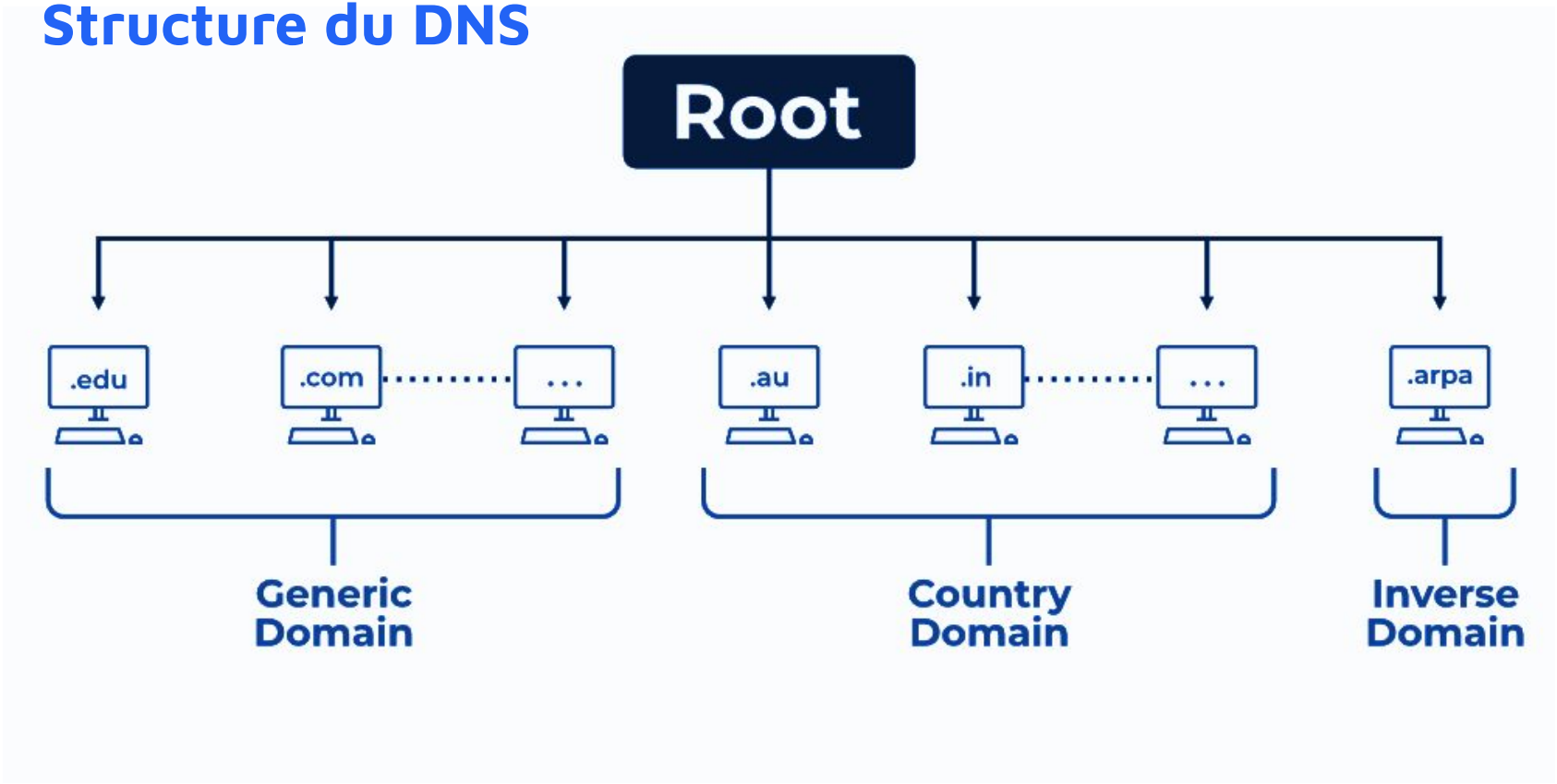
Les domaines de second niveau (SLD) sont les noms qui apparaissent immédiatement avant le TLD. Par exemple, dans example.com, "example" est le domaine de second niveau.

Les propriétaires de domaines enregistrent généralement des domaines de second niveau sous un TLD de leur choix. Ainsi, ils peuvent créer des sous-domaines à leur convenance.

D/ Sous-domaines

Les sous-domaines sont des subdivisions d'un domaine de second niveau. Par exemple, dans www.example.com, "www" est un sous-domaine de "example.com". Les sous-domaines peuvent être utilisés pour organiser un site web ou pour gérer des services spécifiques (ex : mail.example.com pour le courrier électronique).

Structure du DNS



DNS : Fonctionnement

Fonctionnement du DNS

Le processus de résolution DNS se produit lorsque vous entrez un nom de domaine dans votre navigateur. Voici les étapes générales :

Requête DNS (DNS Query) Lorsque vous saisissez www.example.com dans votre navigateur :

- **Requête au résolveur local :** Le navigateur envoie une requête DNS à un résolveur DNS local (souvent fourni par votre fournisseur d'accès Internet ou configuré dans vos paramètres réseau). Ce résolveur DNS local va chercher à résoudre le nom de domaine.

- **Vérification du cache DNS :** Avant de contacter les serveurs DNS externes, le résolveur local vérifie d'abord s'il a déjà la réponse en mémoire (dans son cache). Si c'est le cas, il renvoie immédiatement l'adresse IP au navigateur.

Requête au serveur racine

Si la réponse n'est pas trouvée dans le cache, le résolveur DNS local envoie une requête au serveur racine. Ce serveur racine ne connaît pas directement l'adresse IP de www.example.com, mais il peut indiquer au résolveur quels serveurs gèrent le domaine de premier niveau .com.

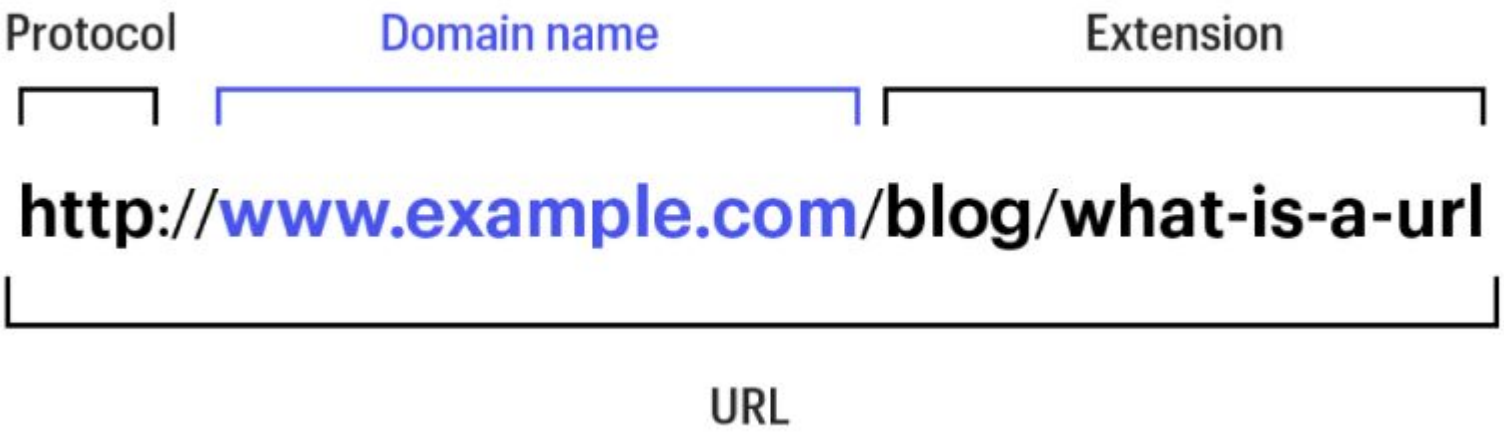
Requête au serveur TLD

Le résolveur contacte alors le serveur DNS du TLD correspondant (.com dans notre exemple). Ce serveur TLD connaît l'adresse des serveurs DNS qui gèrent le domaine example.com.

Fonctionnement du DNS

Requête au serveur de noms autoritaire : Le résolveur local contacte ensuite le serveur de noms autoritaire pour example.com. Ce serveur connaît l'adresse IP associée à www.example.com et la renvoie au résolveur.

Réponse au client : Le résolveur DNS local envoie l'adresse IP trouvée (par exemple, 93.184.216.34) au navigateur web. Le navigateur peut alors se connecter au serveur web à cette adresse IP et télécharger la page demandée.



DNS : Les types des serveurs DNS

Les types des serveurs DNS

Le système DNS fait appel à différents types de serveurs pour fonctionner correctement. Voici les principaux types de serveurs DNS :

Serveur résolveur DNS (DNS Resolver) : Le résolveur DNS est un serveur qui reçoit les requêtes DNS des utilisateurs et qui prend en charge le processus de résolution, en passant par les serveurs racine, TLD, et autoritaires. C'est généralement le premier point de contact pour un client lorsqu'il souhaite résoudre un nom de domaine.

Serveur de noms racine (Root Name Server) : Les serveurs racine sont au sommet de la hiérarchie DNS. Ils redirigent les requêtes vers les serveurs DNS des domaines de premier niveau (TLD), mais ne stockent pas directement les informations d'adresses IP.

Serveur de noms TLD (Top-Level Domain Name Server) : Les serveurs TLD sont responsables des domaines de premier niveau, comme .com, .org, .fr, etc. Ils connaissent les serveurs autoritaires pour les domaines situés sous leur extension.

Serveur de noms autoritaire (Authoritative Name Server) : Le serveur de noms autoritaire contient les informations finales pour un domaine spécifique. Il fournit les adresses IP pour les noms de domaine et est la source ultime d'information pour la résolution DNS.

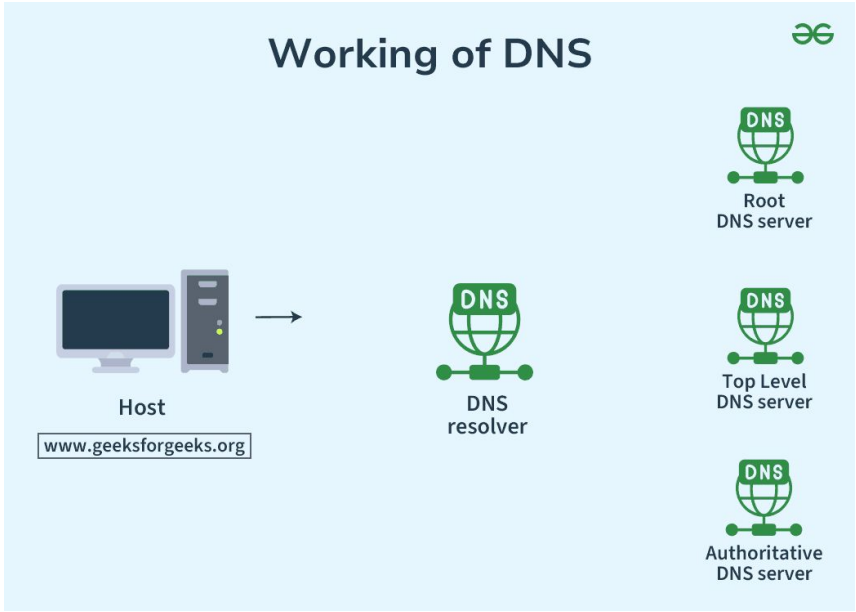
DNS : Quelles sont les étapes d’une recherche ?

Quelles sont les étapes d’une recherche ?

Souvent, les informations de recherche DNS sont stockées temporairement sur votre propre ordinateur ou dans le système DNS lui-même.

Une recherche DNS comprend généralement 8 étapes. Si les informations sont déjà stockées (mises en cache), certaines de ces étapes peuvent être ignorées, ce qui accélère le processus. Voici un exemple des 8 étapes lorsque rien n'est mis en cache :

1/ Un utilisateur tape « exemple.com » dans un navigateur Web.
2/ La requête est envoyée à un résolveur DNS.
3/ Le résolveur demande à un serveur racine où trouver le serveur de domaine de premier niveau (TLD) pour .com.
4/ Le serveur racine indique au résolveur de contacter le serveur TLD .com.
5/ Le résolveur demande ensuite au serveur TLD .com l’adresse IP de « exemple.com ».
6/ Le serveur TLD .com donne au résolveur l'adresse IP du serveur de noms du domaine.
7/ Le résolveur demande ensuite au serveur de noms du domaine l'adresse IP de « exemple.com ».
8/ Le serveur de noms du domaine renvoie l'adresse IP au résolveur.



Quelles sont les étapes d'une recherche ?