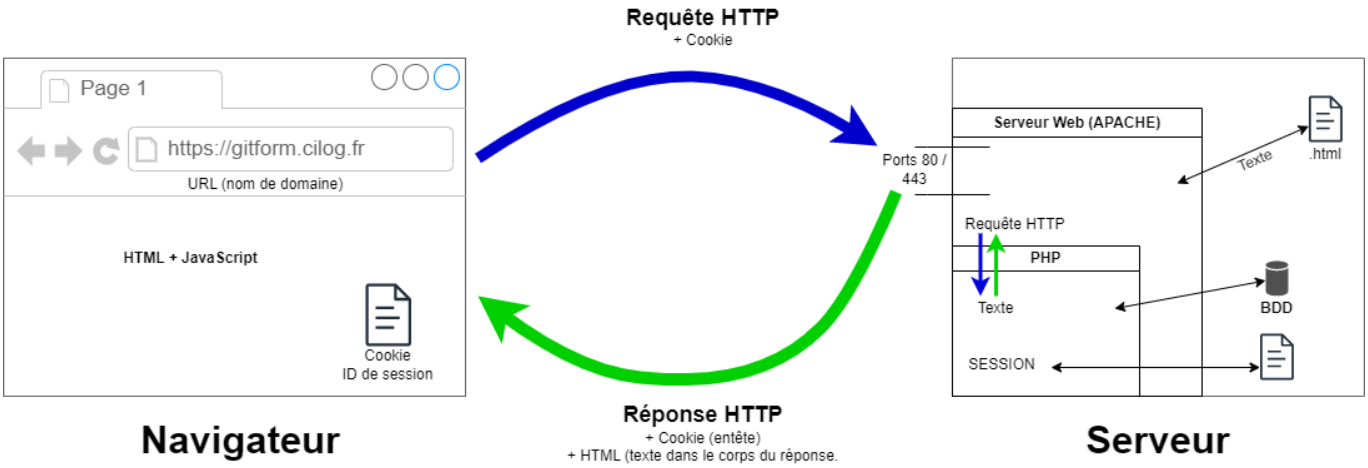
Web Application

Application Web en PHP

Schéma de functionnement :

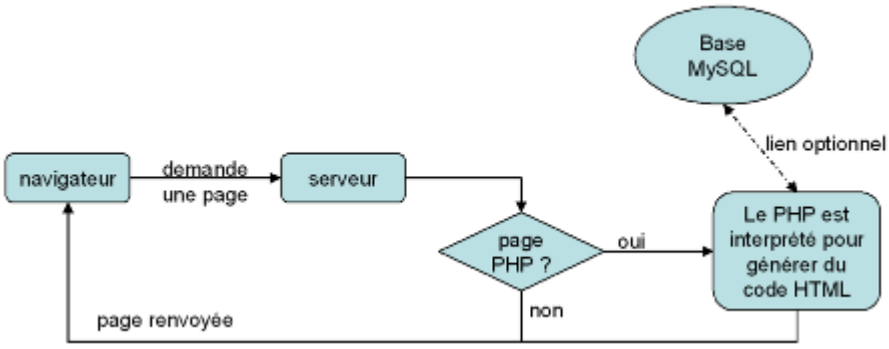


1. Le serveur Apache reçoit les requêtes HTTP des navigateurs des utilisateurs.
2. Apache analyse les requêtes et extrait les informations telles que l'URL demandée, les en-têtes de requête et les paramètres.
3. Si la requête nécessite un traitement PHP, Apache la transmet au module PHP correspondant.
4. Le module PHP exécute le code PHP correspondant à la requête, qui peut interagir avec des bases de données, effectuer des calculs, accéder à des fichiers, etc.
5. Une fois le code PHP exécuté, il renvoie le résultat au module PHP.
6. Le module PHP renvoie ensuite la réponse générée à Apache.
7. Apache envoie la réponse au navigateur de l'utilisateur, généralement sous forme de texte HTML.
8. Le navigateur interprète le texte HTML reçu et l'affiche en tant que page Web dans sa fenêtre de navigation.

Les cookies sont échangés entre le navigateur et le serveur dans les en-têtes de requête et de réponse, et l'HTML en corps de requête et de réponse. Les cookies sont des petits fichiers de texte stockés localement par le navigateur et sont utilisés pour stocker des informations temporaires ou de session lors des interactions avec le serveur.

PHP

PHP ('Hypertext Preprocessor' *ancien* '**P**ersonal **H**ome **P**age') est un langage de programmation utilisé pour la création de sites Web dynamiques. Il permet de générer du contenu HTML en utilisant des instructions PHP imbriquées dans le code HTML. PHP génère donc du texte et le renvoie.



Tableaux associatifs :

En PHP il existe deux tableaux associatifs **$_GET** et **$_POST** qui contient les elements envoyés au serveur Web par le formulaire HTML.

Un tableau associatif est un tableau qui a une clé associée à une valeur.

Les principales différences entre ces deux méthodes :

- La méthode **$_GET** envoie les données dans l'URL de la requête, alors que la méthode **$_POST** envoie les données dans le corps de la requête HTTP. Ainsi, les données envoyées via la méthode **$_POST** sont invisibles dans l'URL.
- La méthode **$_GET** est généralement utilisée pour récupérer des données, tandis que la méthode **$_POST** est généralement utilisée pour envoyer des données qui doivent être modifiées ou stockées sur le serveur.
- Les données envoyées via la méthode **$_GET** sont limitées en taille par la longueur maximale de l'URL, alors que les données envoyées via la méthode **$_POST** n'ont pas de limite de taille.
- Les données envoyées via la méthode **$_GET** peuvent être mises en cache par le navigateur ou les serveurs intermédiaires, tandis que les données envoyées via la méthode **$_POST** ne peuvent pas être mises en cache.
- Les données envoyées via la méthode **$_GET** peuvent être bookmarkées, alors que les données envoyées via la méthode **$_POST** ne peuvent pas être bookmarkées.
- Les données envoyées via la méthode **$_GET** peuvent être vues dans les journaux du serveur, les données envoyées via la méthode **$_POST** ne peuvent pas être vues dans les journaux du serveur.

En résumé, la méthode **$_GET** est plus appropriée pour les requêtes qui ne modifient pas l'état du serveur (par exemple, la récupération de données), tandis que la méthode **$_POST** est plus appropriée pour les requêtes qui modifient l'état du serveur (par exemple, l'ajout de données dans une base de données).

Voici un exemple de formulaire HTML simple pour envoyer les données via la méthode **$_POST** :

```
<form method="post" action="traitement.php">
    <label for="nom">Nom :</label>
    <input type="text" name="nom" id="nom">
    <br>
```

```
    <label for="email">E-mail :</label>
    <input type="email" name="email" id="email">
    <br>
    <input type="submit" value="Envoyer">
</form>
```

Dans ce formulaire, deux champs sont créés pour saisir le nom et l'adresse e-mail de l'utilisateur. Lorsque le formulaire est soumis, les données sont envoyées à un fichier nommé traitement.php, où elles peuvent être récupérées à l'aide du tableau associatif **$_POST**, comme suit :

```
$nom = $_POST['nom'];
$email = $_POST['email'];
echo "Nom : " . $nom . "<br>";
echo "E-mail : " . $email;
```

Dans cet exemple, les valeurs entrées par l'utilisateur sont stockées dans les variables $nom et $email à l'aide du tableau associatif **$_POST**, puis affichées à l'écran à l'aide de la fonction **echo()**.

Concaténation en PHP :

En PHP, la concaténation se fait avec un point :

```
$nom = "Jean";
$prenom = "Dupont";
$message = "Bonjour, je m'appelle " . $prenom . " " . $nom . ".";
echo $message;
```

Dans cet exemple, la variable **$message** est créée en utilisant l'opérateur de concaténation pour joindre la chaîne de caractères "Bonjour, je m'appelle ", la variable **$prenom**, un espace, la variable $nom, et un point final. Le résultat est stocké dans la variable **$message**. Ensuite, le contenu de **$message** est affiché à l'écran à l'aide de la fonction **echo**.

Session :

Une session PHP est un mécanisme qui permet de stocker des données côté serveur pour un utilisateur donné pendant une durée limitée. Contrairement aux cookies, qui sont stockés côté client (dans le navigateur de l'utilisateur), les sessions sont stockées côté serveur.

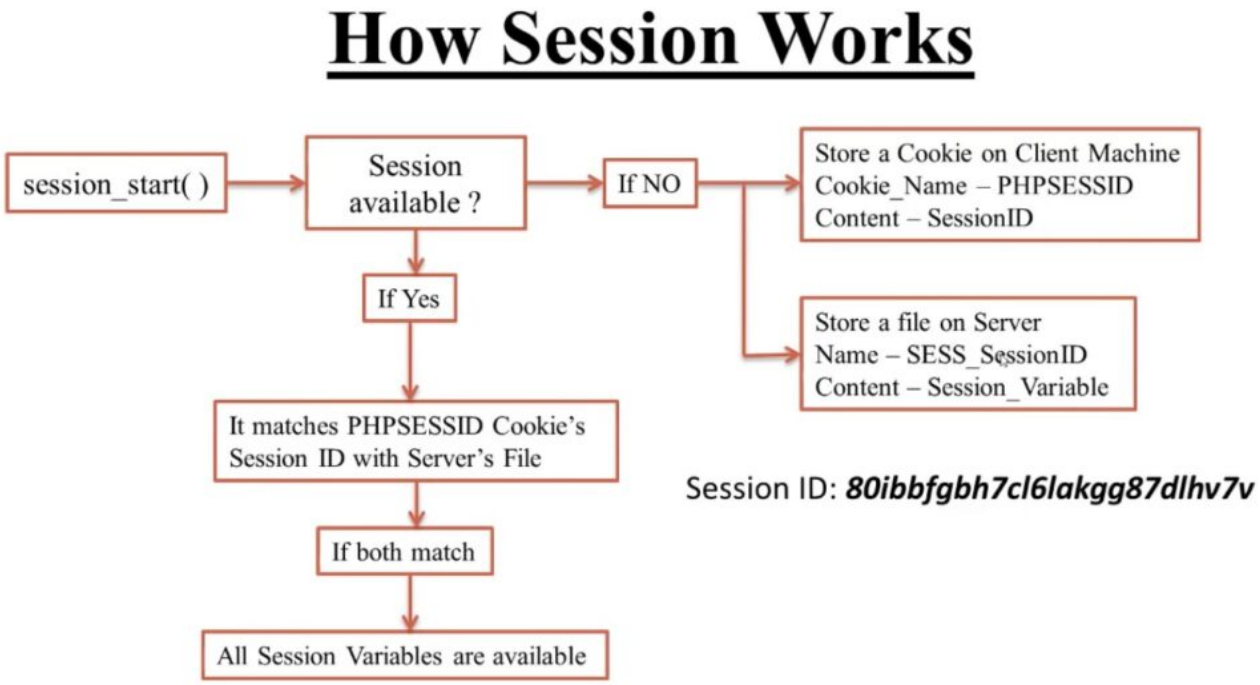
Lorsqu'un utilisateur accède à un site Web qui utilise des sessions, le serveur crée une session pour cet utilisateur et lui attribue un identifiant unique appelé ID de session. Cet ID de session est généralement stocké dans un cookie côté client, afin que le serveur puisse identifier l'utilisateur lors de ses visites ultérieures.

Les données de la session sont stockées dans un fichier texte ou dans une base de données côté serveur, et sont accessibles via un tableau associatif en PHP appelé **$_SESSION**. Les données stockées dans **$_SESSION** peuvent être des variables de toutes sortes (chaînes de caractères, nombres, tableaux, etc.) et sont

généralement utilisées pour stocker des informations d'état sur l'utilisateur (par exemple, son nom d'utilisateur, ses préférences, etc.).

La durée de vie d'une session est généralement déterminée par la durée de vie du cookie qui contient l'ID de session, bien que cela puisse être configuré par le développeur. Lorsque l'utilisateur ferme son navigateur ou que la durée de vie de la session expire, les données de session sont automatiquement supprimées côté serveur.

Les appels à **session_start()** et **session_destroy()** doivent être effectués avant tout autre code HTML ou texte de votre script PHP, les données stockées dans la session sont accessibles sur toutes les pages du site Web qui utilisent la même session.



Exemple de code de l'utilisation de sessions en PHP :

1. Tout d'abord, vous devez démarrer une session en appelant la fonction **session_start()**. Cette fonction doit être appelée avant tout code HTML ou texte de votre script PHP.

```
// Démarrer la session
session_start();
```

```
// Démarrer la session si il n'y a pas une autre session ouverte
<?php
if (session_status() == PHP_SESSION_NONE) {
  session_start();
}
?>
```

2. Vous pouvez ensuite stocker des données dans la session en utilisant le tableau associatif **$_SESSION**. Par exemple, pour stocker le nom d'utilisateur d'un utilisateur connecté :

```
$_SESSION['username'] = 'JohnDoe';
```

3. Vous pouvez récupérer les données stockées dans la session à tout moment en accédant à l'élément correspondant du tableau **$_SESSION**. Par exemple, pour afficher le nom d'utilisateur stocké dans l'exemple ci-dessus :

```
echo 'Bienvenue, ' . $_SESSION['username'];
```

4. Enfin, vous pouvez détruire la session et toutes les données qu'elle contient en appelant la fonction **session_destroy()**. Cela est souvent utile lorsque l'utilisateur se déconnecte de votre application.

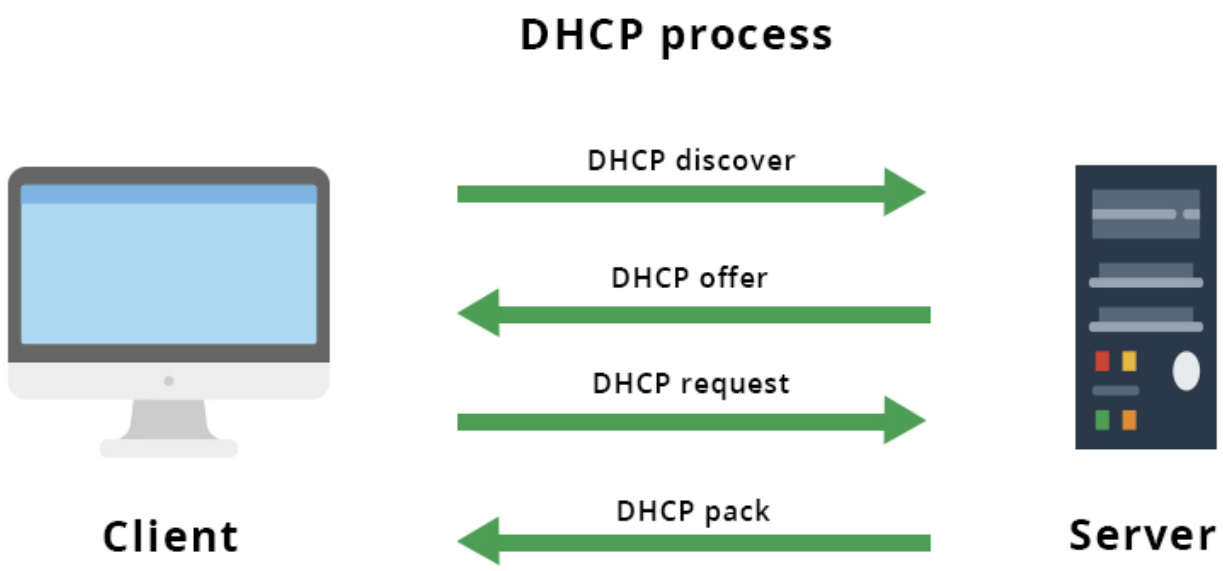
```
// Détruire la session
session_destroy();
```

Le code complet :

```
// Démarrer la session si il n'y a pas une autre session ouverte
<?php
if (session_status() == PHP_SESSION_NONE) {
  session_start();
}

// Stocker le nom d'utilisateur dans la session
$_SESSION['username'] = 'JohnDoe';

// Afficher le nom d'utilisateur
echo 'Bienvenue, ' . $_SESSION['username'];

// Détruire la session
session_destroy();
?>
```

DHCP :

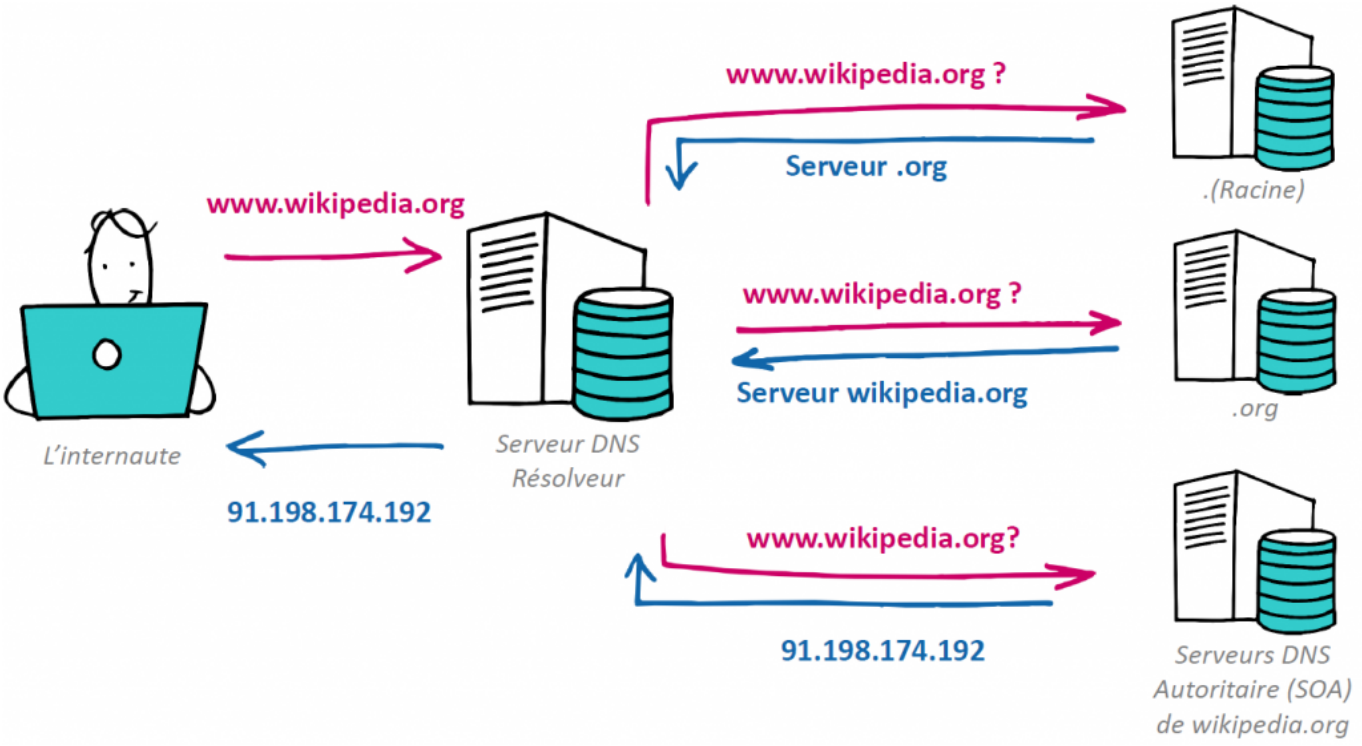
DHCP (**D**ynamic **H**ost **C**onfiguration **P**rotocol) est un protocole réseau qui permet aux ordinateurs et autres dispositifs réseau de recevoir automatiquement une configuration réseau de la part d'un serveur DHCP. Cette configuration inclut généralement une adresse IP, un masque de sous-réseau, une passerelle par défaut et une ou plusieurs adresses de serveurs DNS.



DNS :

Le DNS (**D**omain **N**ame **S**ystem) est un système informatique qui permet de traduire les noms de domaine en adresses IP. Les noms de domaine, tels que "google.com" ou "wikipedia.org", sont des noms faciles à retenir, mais les ordinateurs utilisent plutôt des adresses IP numériques pour identifier les ressources sur le réseau. Le DNS permet de faire la correspondance entre les noms de domaine et les adresses IP correspondantes.

Lorsqu'un utilisateur saisit une URL dans son navigateur Web, le navigateur envoie une requête DNS au serveur DNS pour récupérer l'adresse IP correspondant au nom de domaine. Si le serveur DNS est en mesure de résoudre le nom de domaine, il renvoie l'adresse IP au navigateur, qui peut alors établir une connexion avec le serveur Web correspondant.



Buffer :

En informatique, un buffer (ou tampon en français) est une zone de la mémoire qui sert d'espace de stockage temporaire pour les données en transit entre deux processus ou deux périphériques. Les données sont généralement stockées dans le buffer avant d'être traitées ou transférées, ce qui permet de les manipuler de manière plus efficace et de mieux gérer les flux de données.

Cookie :

Un cookie est un petit fichier texte qui est stocké sur l'ordinateur d'un utilisateur par son navigateur Web. Les cookies sont généralement utilisés pour stocker des informations qui peuvent être récupérées par un site Web lors de visites ultérieures, afin de personnaliser l'expérience de l'utilisateur ou de maintenir sa session ouverte.

Les cookies peuvent contenir une variété d'informations, telles que des préférences utilisateur, des identifiants de session, des informations de connexion, etc. Lorsqu'un utilisateur visite un site Web, le serveur Web envoie un cookie au navigateur de l'utilisateur, qui stocke alors le cookie sur son ordinateur. Lorsque l'utilisateur visite à nouveau le site Web, le navigateur envoie le cookie au serveur Web, qui peut alors utiliser les informations stockées dans le cookie pour personnaliser l'expérience utilisateur.

Il existe deux types de cookies : les cookies de session et les cookies persistants. Les cookies de session sont temporaires et sont supprimés lorsque l'utilisateur ferme son navigateur. Les cookies persistants, quant à eux, sont stockés sur l'ordinateur de l'utilisateur pour une durée déterminée (par exemple, plusieurs jours, semaines ou mois) et sont utilisés pour stocker des informations qui doivent être conservées entre les sessions.

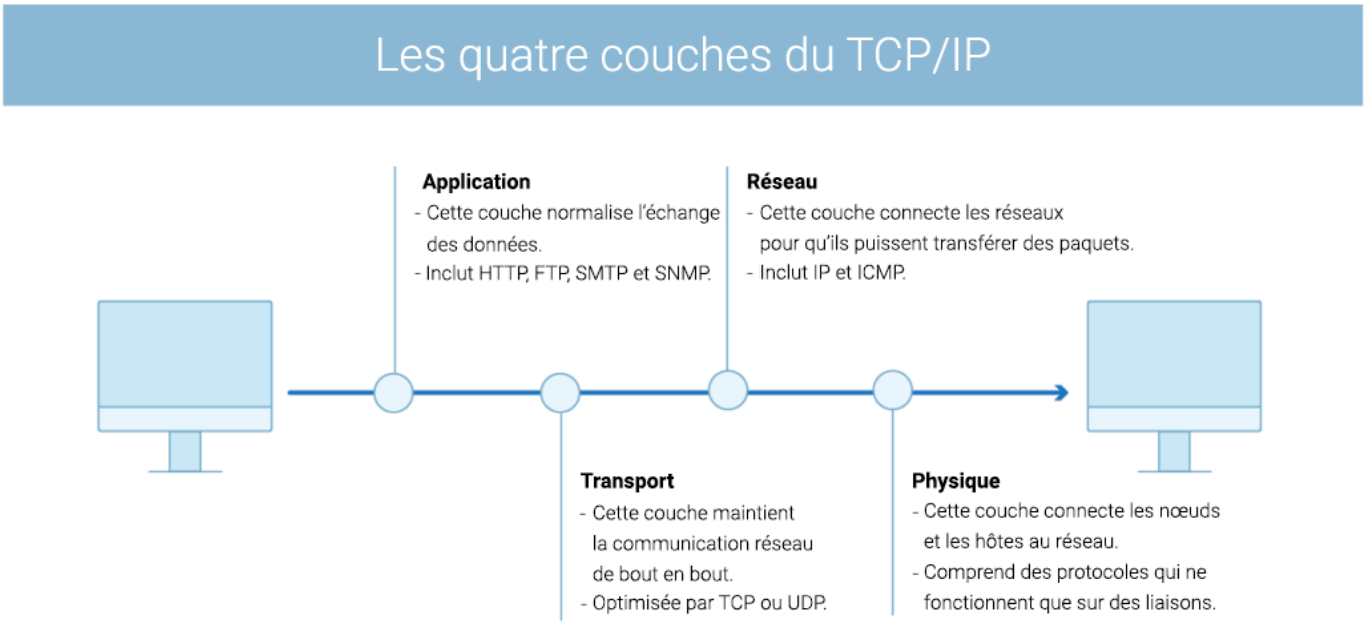
TCP/IP :

TCP/IP (**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol) est une suite de protocoles de communication utilisée pour la transmission de données sur les réseaux informatiques, y compris Internet. Elle est constituée de deux protocoles principaux : le protocole de contrôle de transmission (**TCP**) et le protocole Internet (**IP**).

Le protocole de contrôle de transmission (**TCP**) est responsable de la division des données en paquets, de la gestion des connexions et du contrôle de la qualité de service. Il garantit que les données sont transmises de manière fiable, sans perte ni altération, en fournissant un mécanisme de vérification de l'intégrité des données.

Le protocole Internet (**IP**) est responsable de l'adressage et du routage des paquets de données à travers le réseau. Il fournit une méthode pour identifier les périphériques sur le réseau en utilisant des adresses IP uniques et permet de transférer les paquets de données de manière transparente entre les réseaux.

TCP/IP est utilisé pour la communication entre tous les types de périphériques connectés à Internet, y compris les ordinateurs, les téléphones mobiles, les tablettes et les objets connectés. Il est également utilisé pour la communication entre les différents réseaux, y compris les réseaux locaux (**LAN**), les réseaux étendus (**WAN**) et Internet.



HTTP / HTTPS :

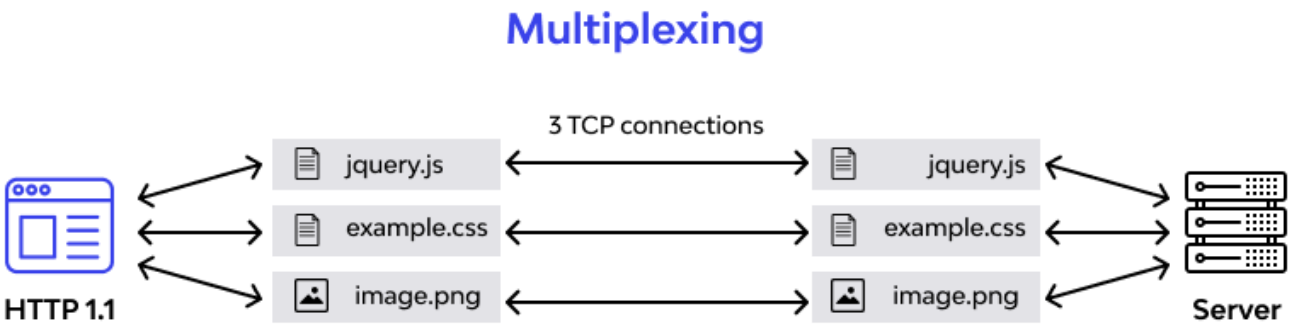
HTTP (**H**yper**t**ext **T**ransfer **P**rotocol) est un protocole utilisé pour transférer des données sur Internet. Il est la fondation de la communication de données pour le World Wide Web. HTTP est un protocole de demande/réponse, ce qui signifie qu'un client envoie une demande à un serveur et que le serveur répond avec un message contenant les informations demandées.

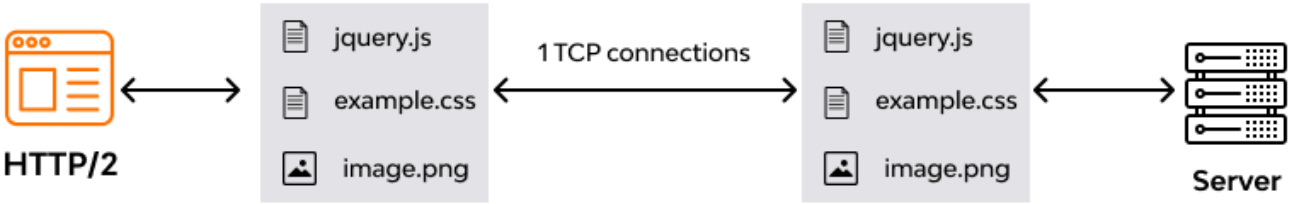
HTTP utilise un modèle client-serveur, où le client (généralement un navigateur Web) envoie des demandes au serveur, et le serveur répond avec les données demandées. Le client envoie une demande sous la forme d'un message qui inclut une méthode (comme **GET** ou **POST**) et une **URL** (**U**niform **R**esource **L**ocator) qui identifie la ressource à accéder. Le serveur répond ensuite avec un message contenant les données demandées, ainsi qu'un code d'état qui indique si la demande a été réussie ou non.

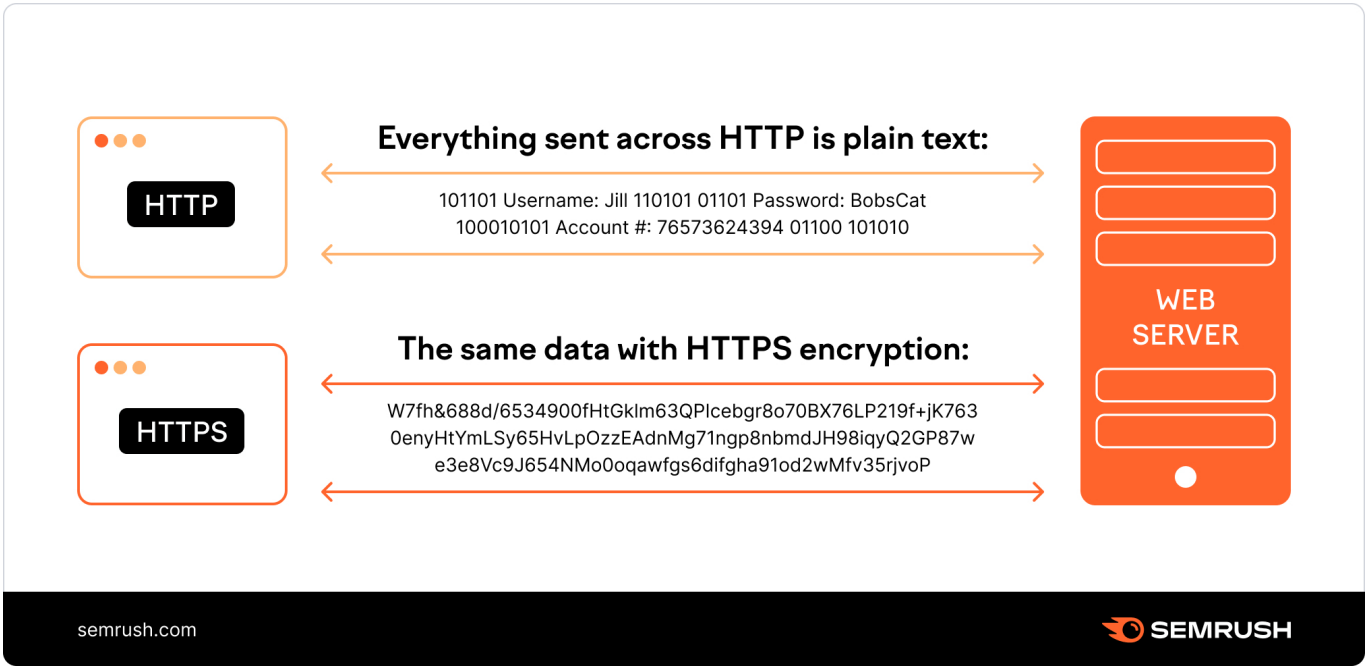
HTTP est un protocole sans état, ce qui signifie que chaque demande et réponse est indépendante de toutes les demandes ou réponses précédentes. Pour maintenir des informations d'état entre les demandes, des cookies sont souvent utilisés pour stocker des informations de session.

HTTP a évolué au fil des ans, avec de nouvelles versions telles que HTTP/2 et HTTP/3 développées pour améliorer les performances et la sécurité. HTTPS (HTTP sécurisé) est une version de HTTP qui utilise le chiffrement SSL/TLS pour sécuriser la communication entre le client et le serveur.

HTTP : Port 80
HTTPS : Port 443







URL :

Une **URL** (**U**niform **R**esource **L**ocator) est une adresse unique qui identifie une ressource sur Internet, telle qu'une page Web, une image, une vidéo ou un fichier. Elle est utilisée pour localiser et accéder à des ressources sur le World Wide Web.

Une URL se compose généralement de plusieurs parties :

- **Le protocole :** indique le protocole utilisé pour accéder à la ressource, tel que HTTP ou HTTPS.
- **Le nom de domaine :** l'adresse du serveur où se trouve la ressource, telle que <www.example.com>.
- **Le chemin :** le chemin d'accès à la ressource sur le serveur, tel que /page/index.html.

- **Les paramètres :** des paramètres supplémentaires pouvant être ajoutés à l'URL pour spécifier des options ou des informations supplémentaires.

Par exemple, voici une URL typique pour une page Web :

https://www.example.com/page/index.html?lang=fr

Dans cet exemple, "https://" indique le protocole de sécurité utilisé, "www.example.com" est le nom de domaine, "/page/index.html" est le chemin d'accès à la page, et "?lang=fr" sont les paramètres supplémentaires indiquant la langue de la page.

Les URL sont utilisées dans les navigateurs Web pour accéder à des pages Web et dans de nombreuses autres applications pour accéder à des ressources sur Internet.