



Programmes coopérants



Côté Serveur !

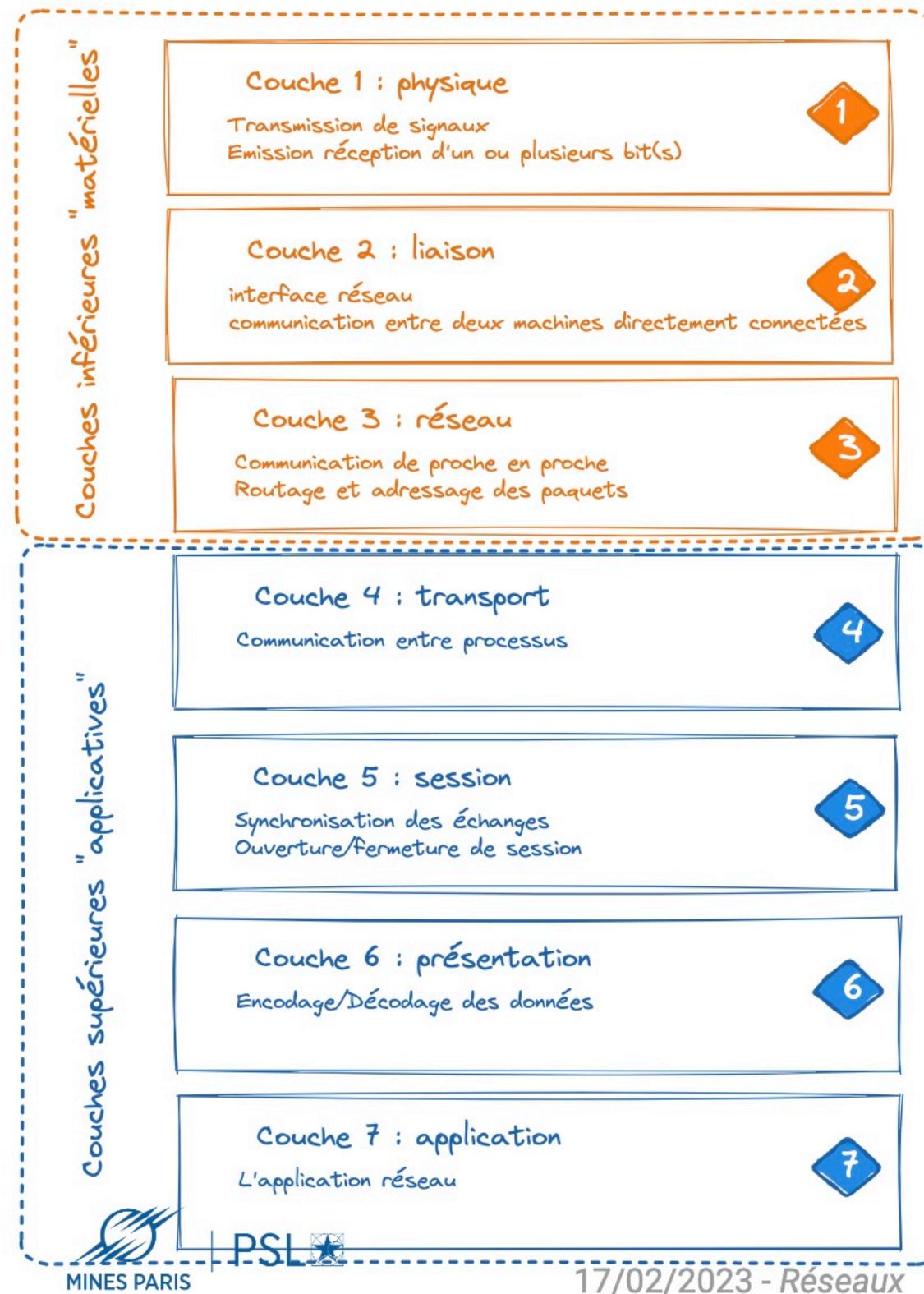
Basile Marchand¹



1 - Plateforme SISDev, Centre des Matériaux, MINES Paris - CNRS - Université PSL

Récap de la semaine dernière

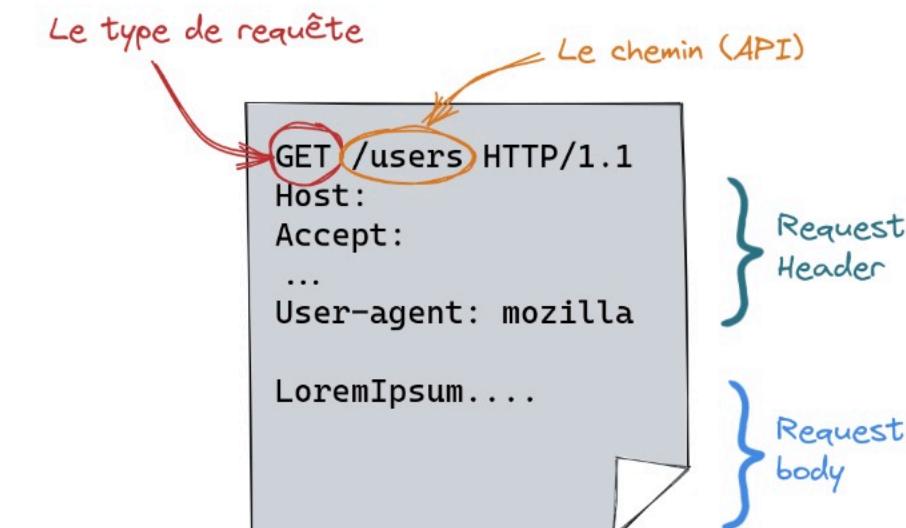
Architecture classique Client <-> Serveur avec des variations peer-to-peer, three-tier, ...



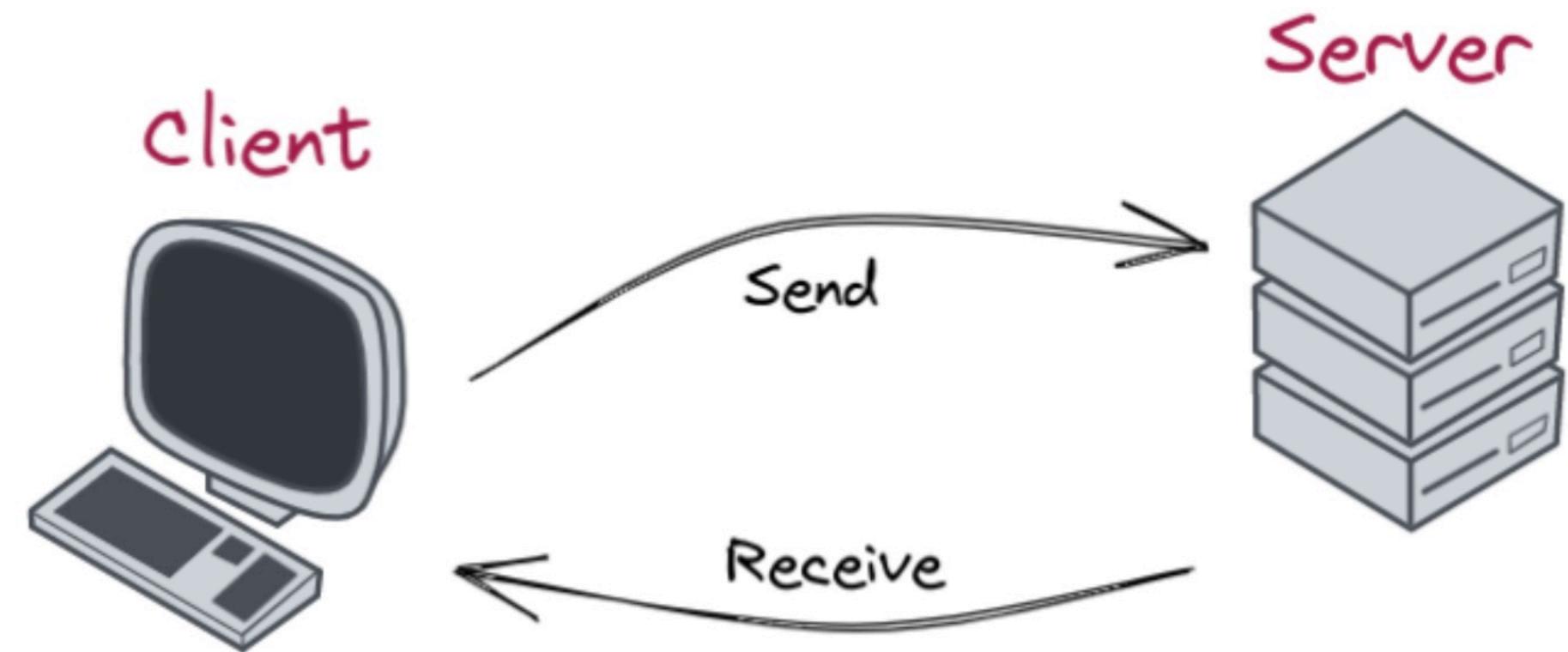
Un modèle OSI en 7 couches

172.16.254.42
Masque réseau Machine sur le réseau

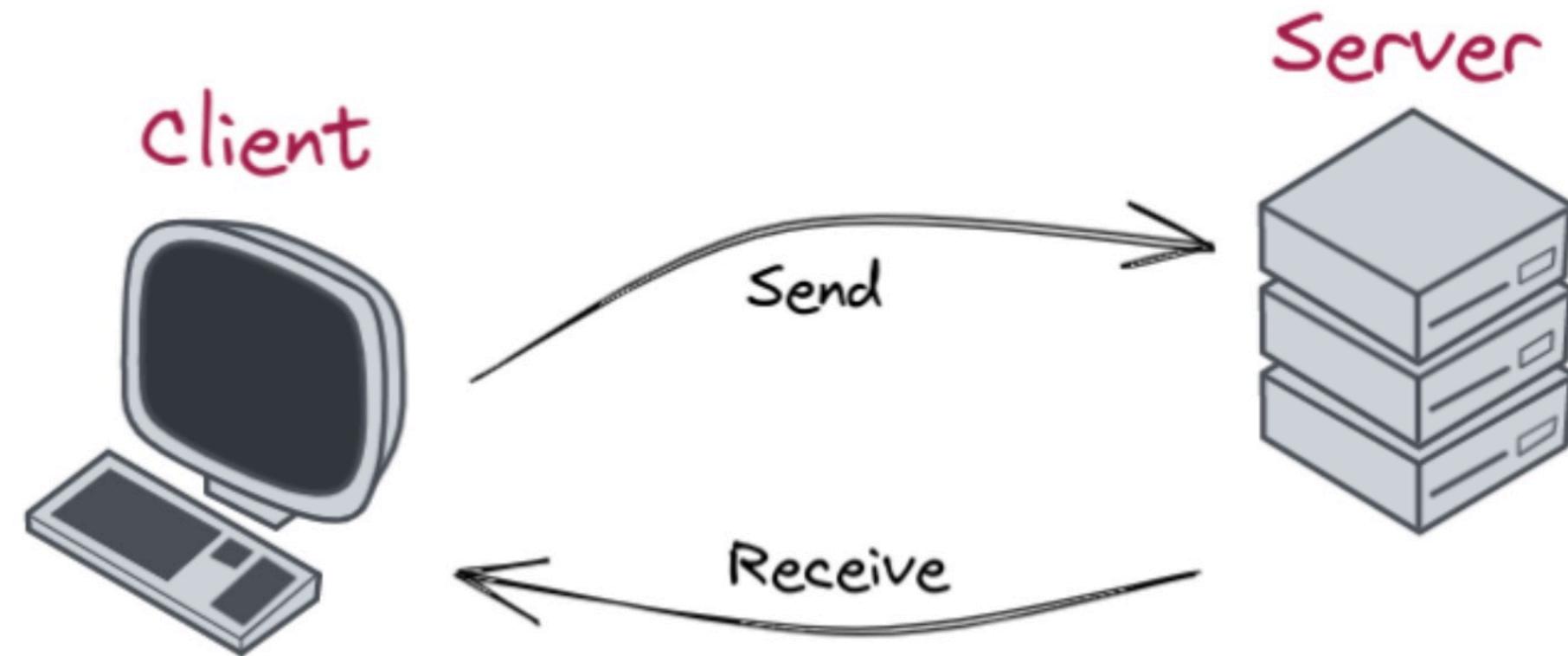
Un protocole HTTP(S) pour le web



Quel est le rôle du serveur ?

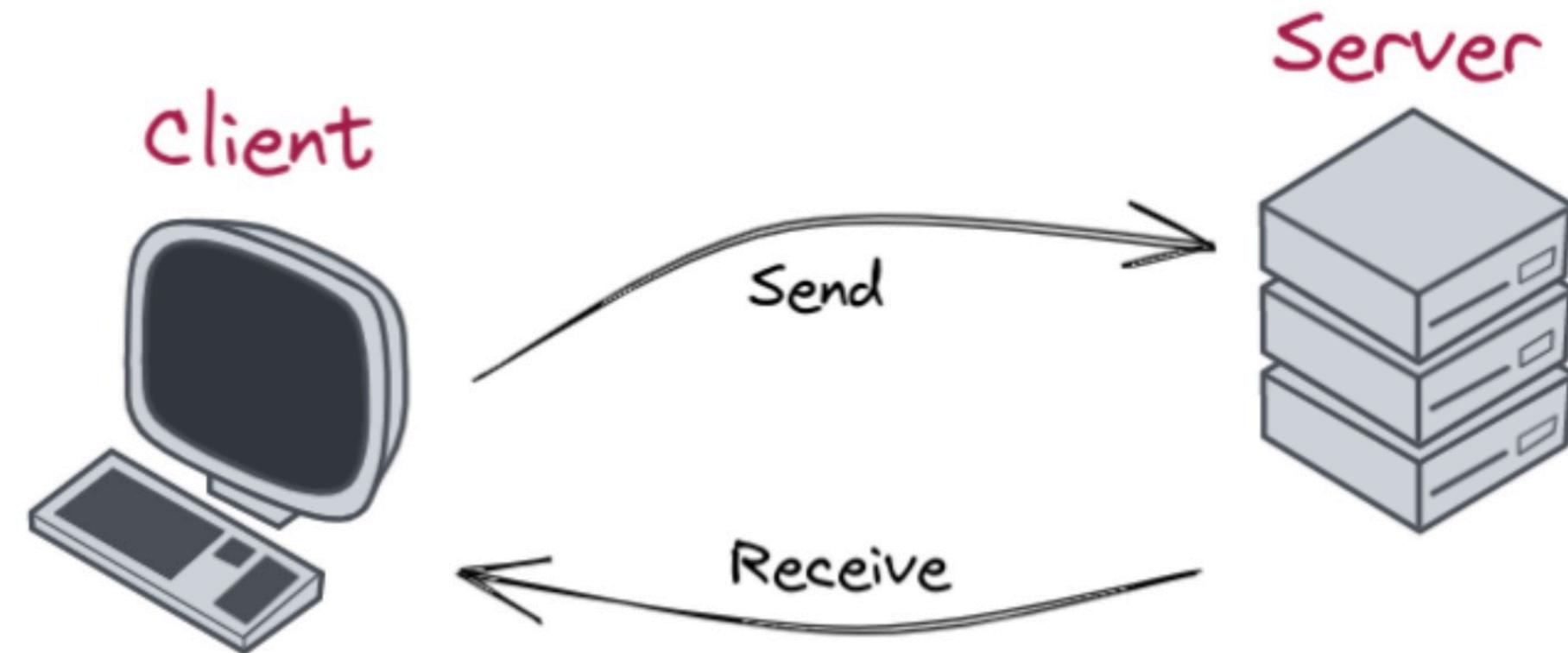


Quel est le rôle du serveur ?



😴 Attendre et attendre et attendre ... 😴

Quel est le rôle du serveur ?



😴 Attendre et attendre et attendre ... 😴

Et de temps en temps 😴 il doit traiter une requête !

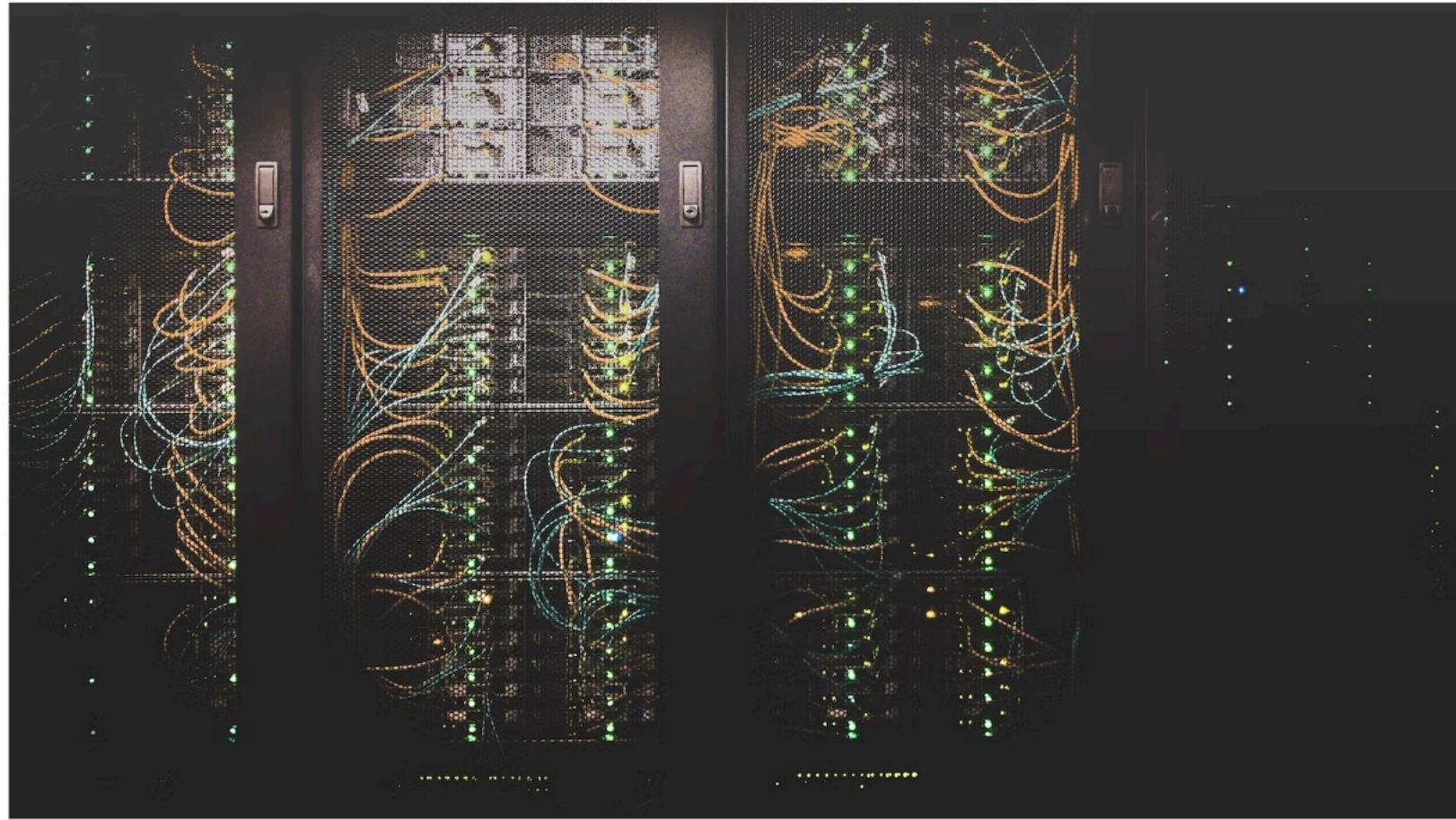
Serveur et serveur deux choses différentes

Attention il y a deux significations à serveur ...



Serveur et serveur deux choses différentes

Le serveur hardware

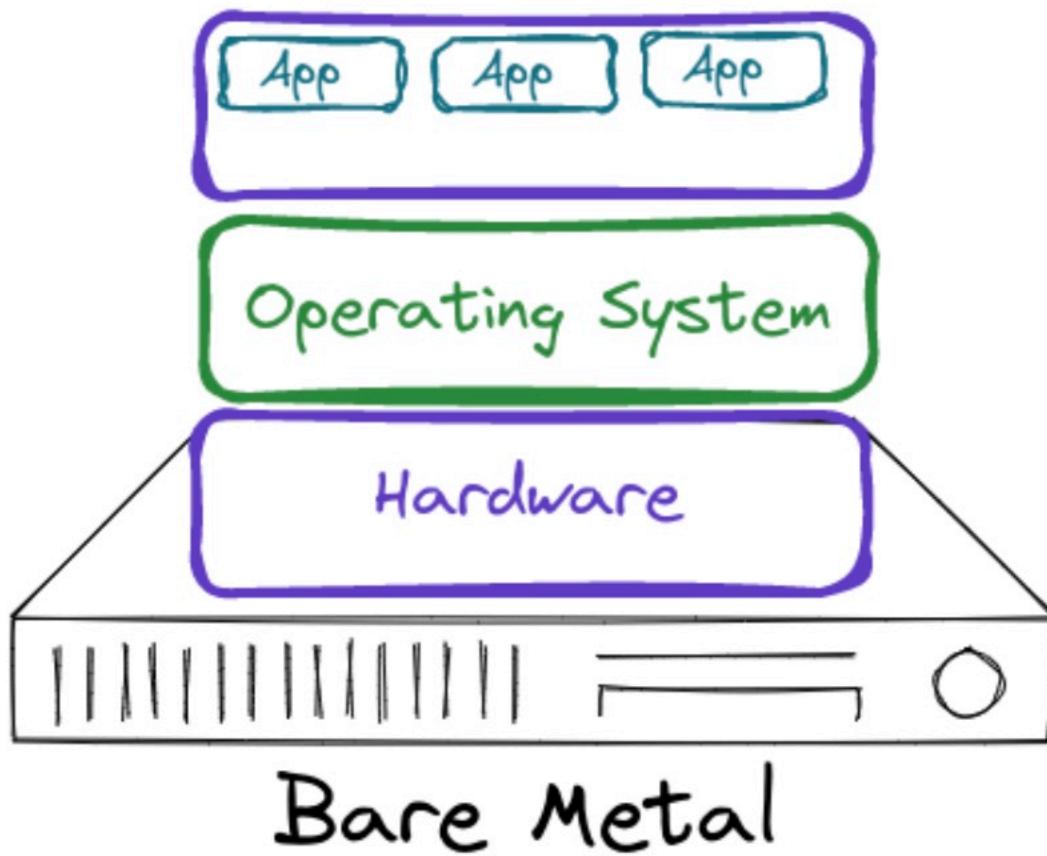


C'est la machine **physique ou virtuelle** connectée au réseau qui va recevoir des paquets de données mais en aucun cas ne s'occupera du traitement de ces données

Serveur et serveur deux choses différentes

Le serveur hardware : différents types

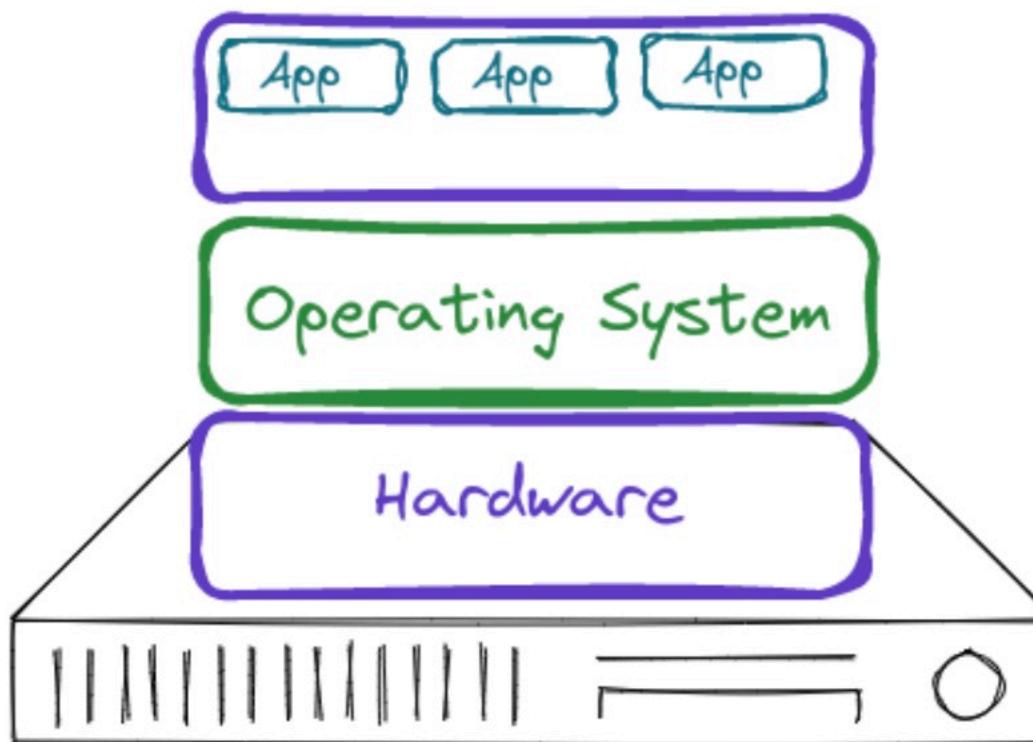
Serveur physique vs serveur virtuel (VPS)



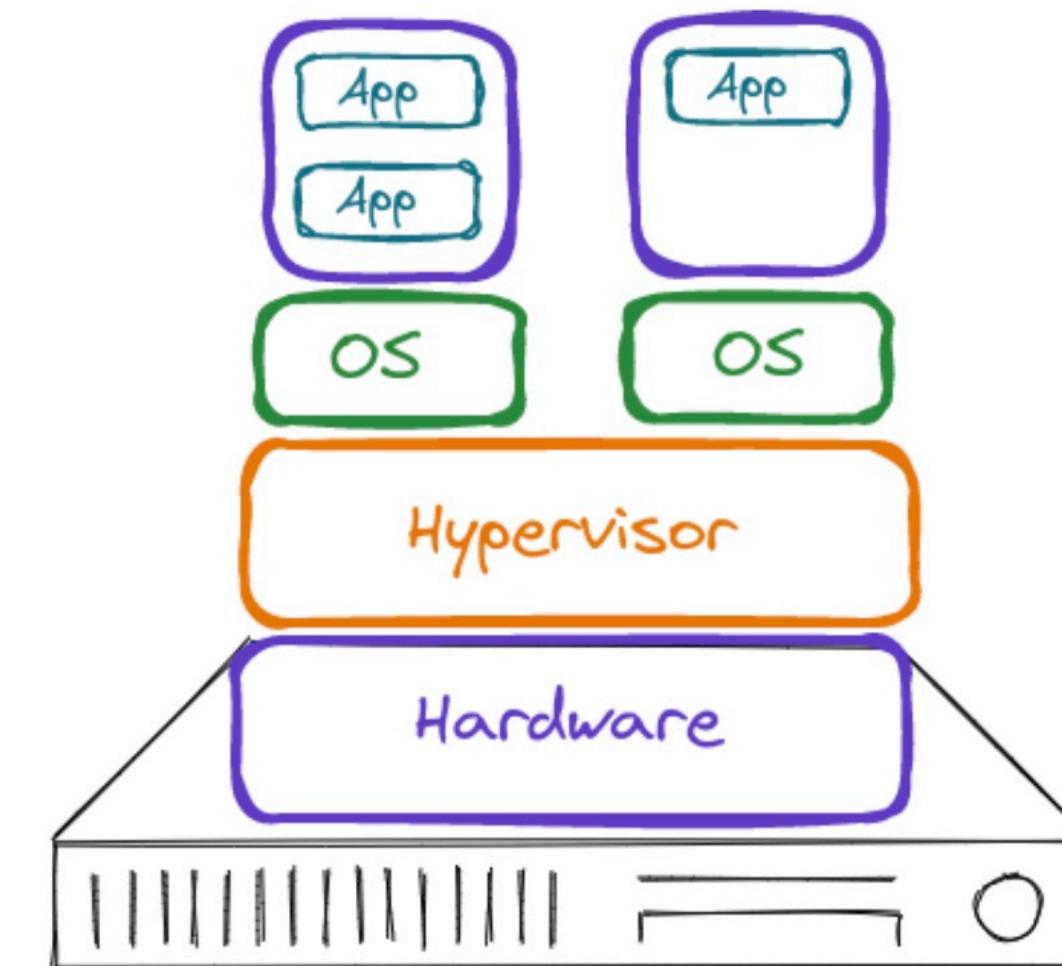
Serveur et serveur deux choses différentes

Le serveur hardware : différents types

Serveur physique vs serveur virtuel (VPS)



Bare Metal

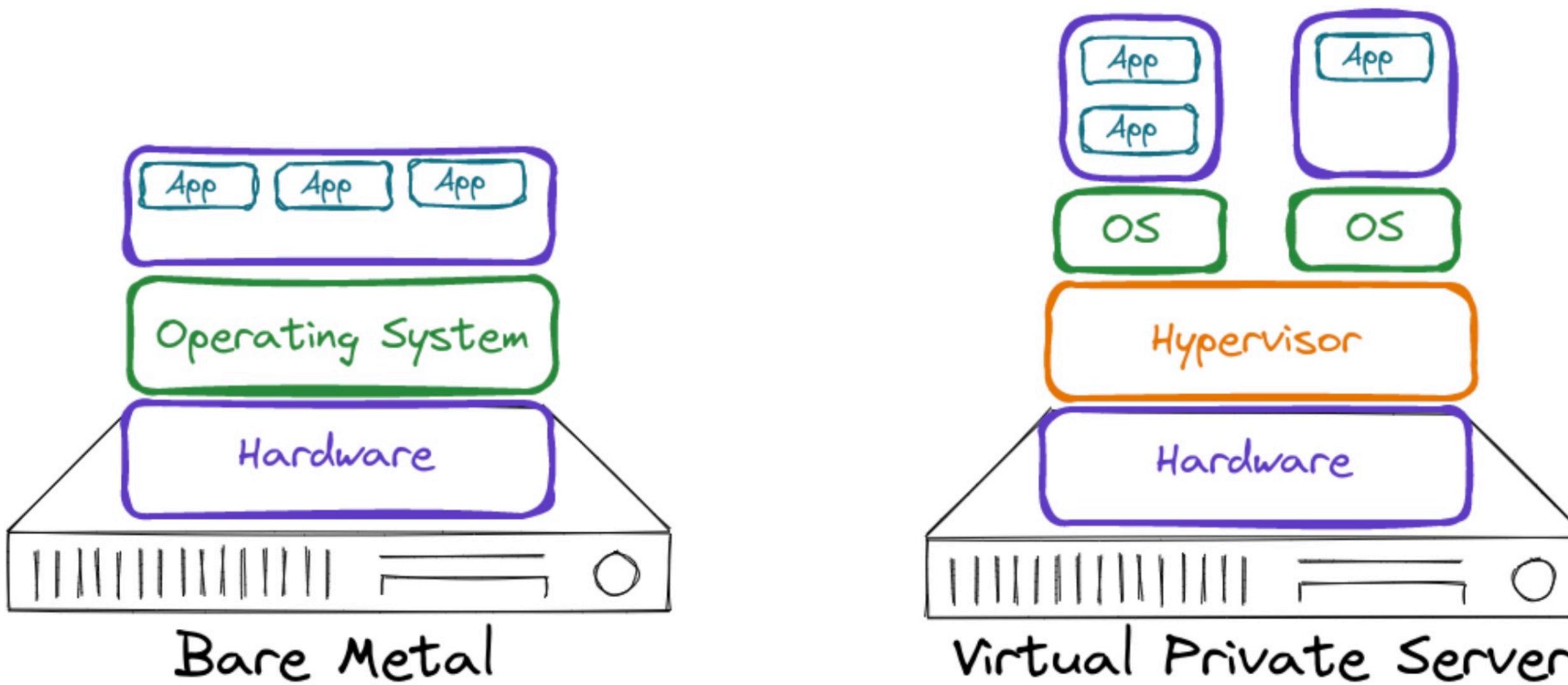


Virtual Private Server

Serveur et serveur deux choses différentes

Le serveur hardware : différents types

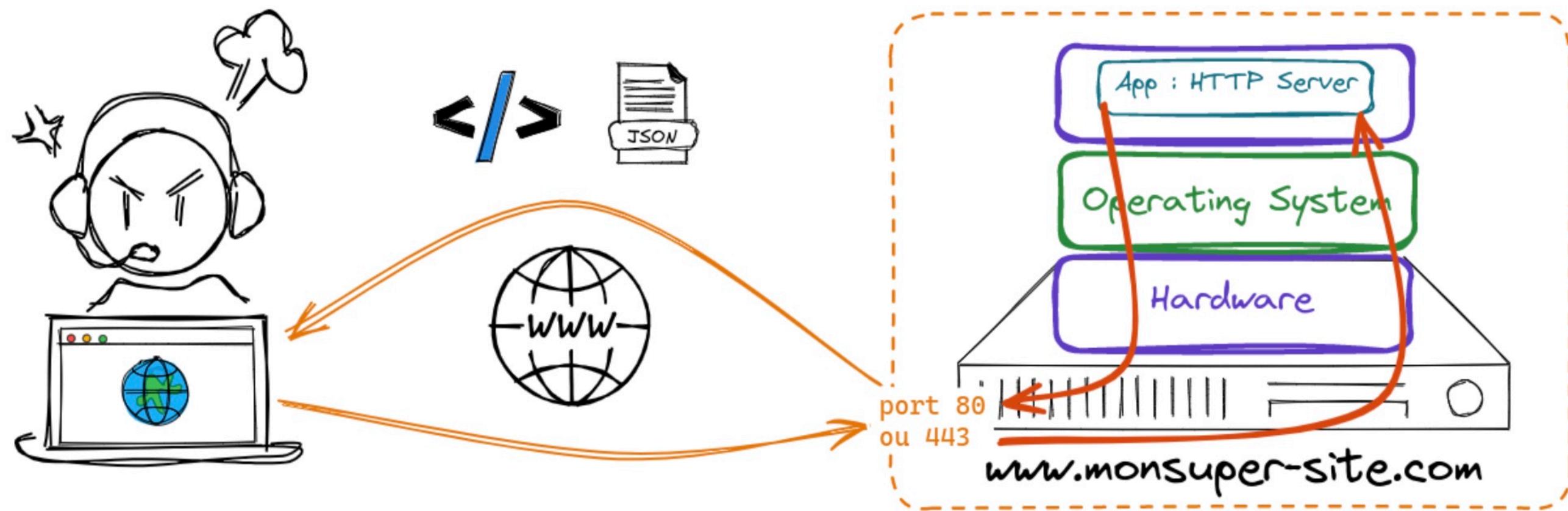
Serveur physique vs serveur virtuel (VPS)



Différentes solutions : On Premise vs Cloud (OVH, Azure, GCP, AWS, ...)

Serveur et serveur deux choses différentes

Le serveur "software"



C'est l'application (au sens logiciel) qui va s'occuper de

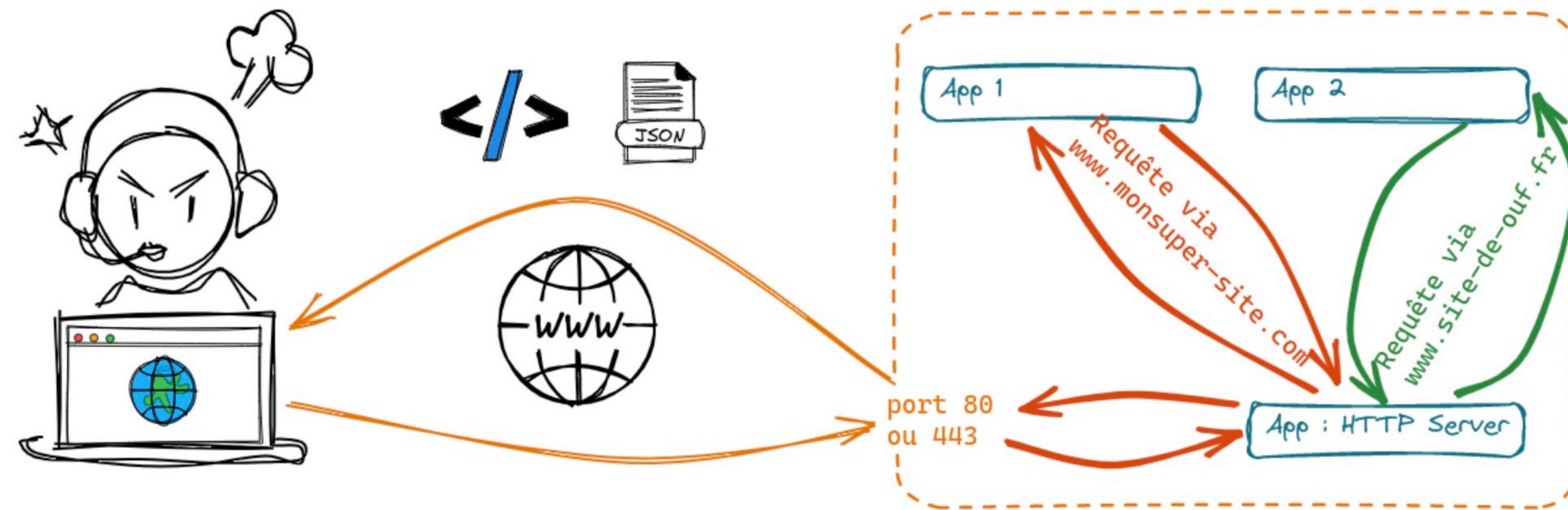
Recevoir, Traiter et Répondre aux requêtes HTTP

Différentes solutions : Apache (40%), Nginx (20%), IIS (10%), ...

Source : <https://fr.hostadvice.com/marketshare/server/>

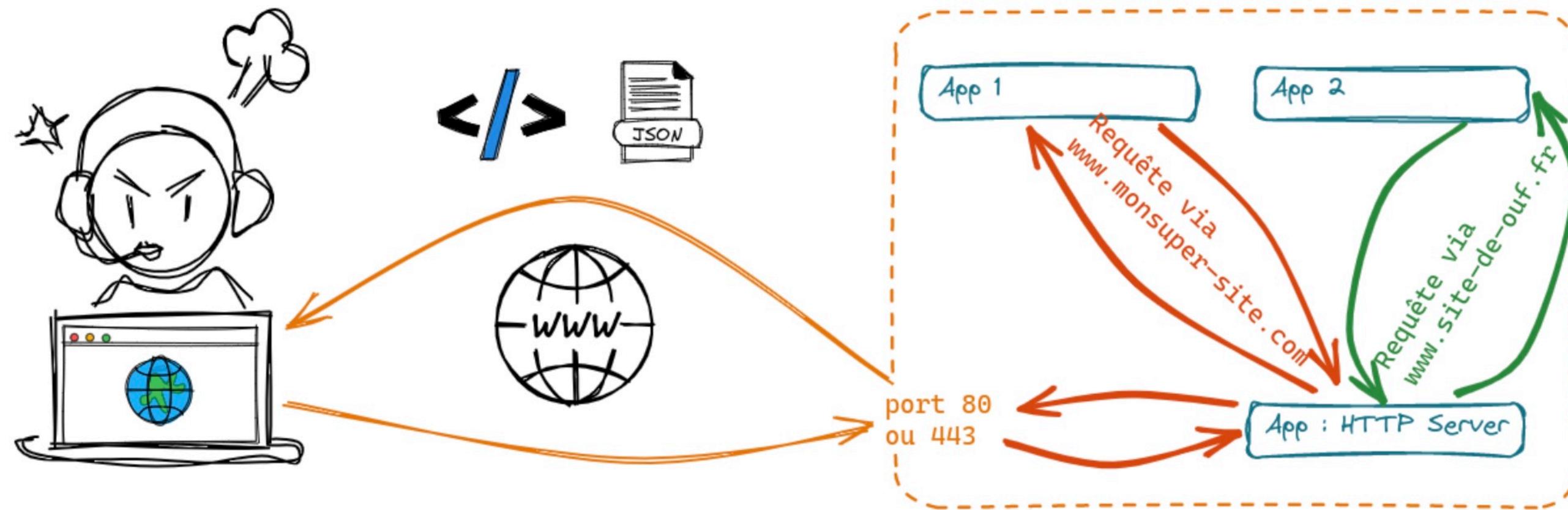
Héberger plusieurs serveurs HTTP(S) sur un même serveur physique ?

OUI 🎯 il suffit de se partager le port 80 🤝



Héberger plusieurs serveurs HTTP(S) sur un même serveur physique ?

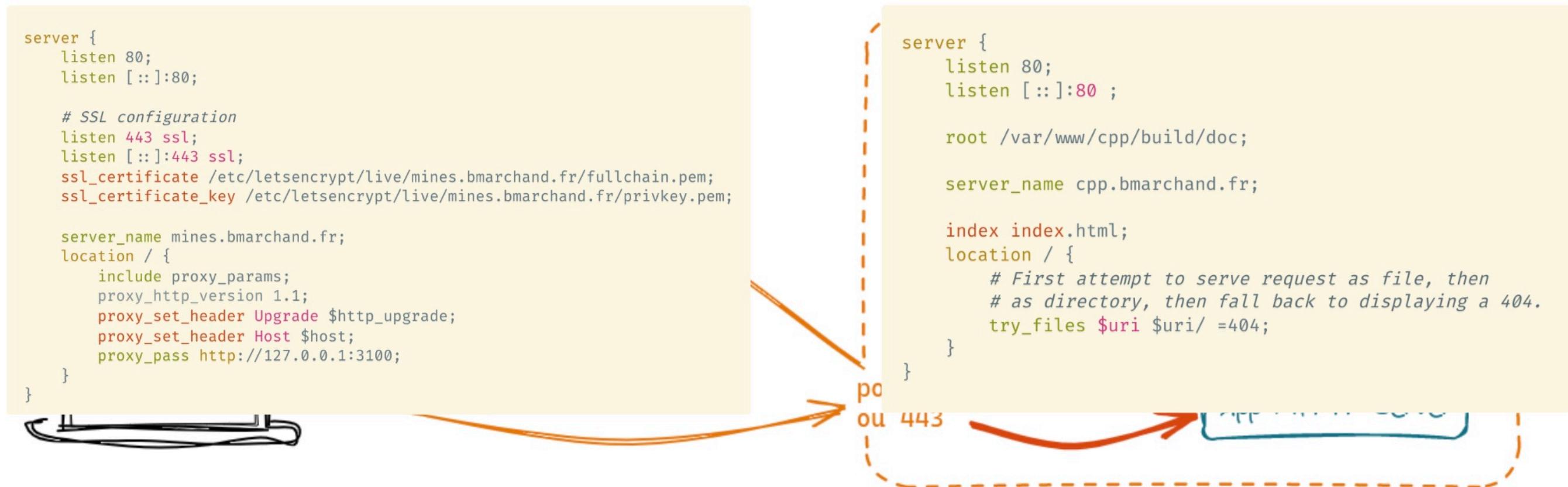
OUI 🎯 il suffit de se partager le port 80 🤝



Il suffit de configurer au niveau du serveur HTTP des **Virtual Host**

Héberger plusieurs serveurs HTTP(S) sur un même serveur physique ?

OUI 🚀 il suffit de se partager le port 80 🤝



Il suffit de configurer au niveau du serveur HTTP des **Virtual Host**

Tous les serveurs font la même chose ?

Deux applications

Sites statiques vs dynamiques

ApprentissageProgrammationC

Page principale

Recherche

Ce site utilise des cookies (google analytics) pour mesurer l'audience

Accepter

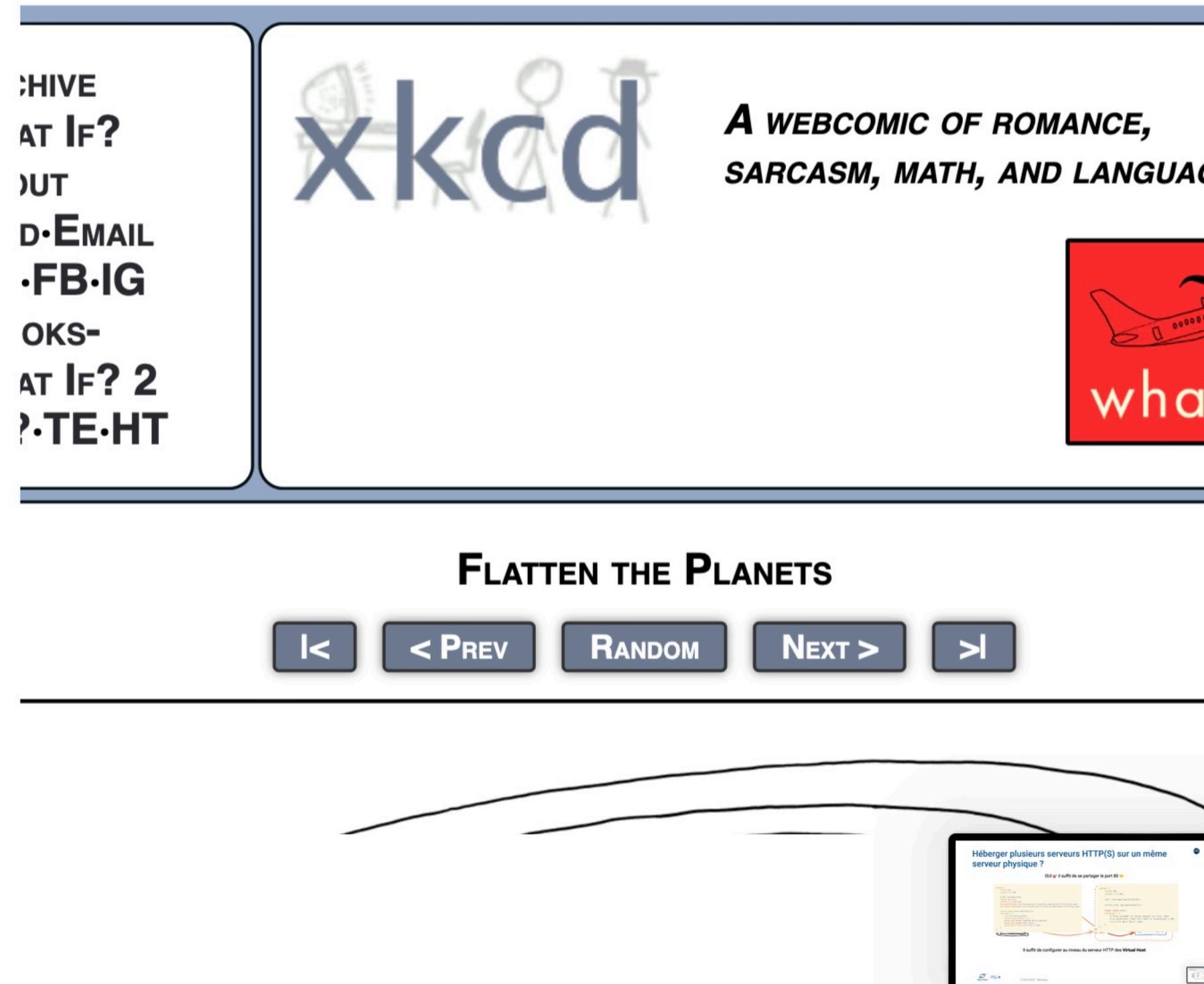
Continuer sans accepter

la notion de *bloc*

es de contrôle

Table des matières

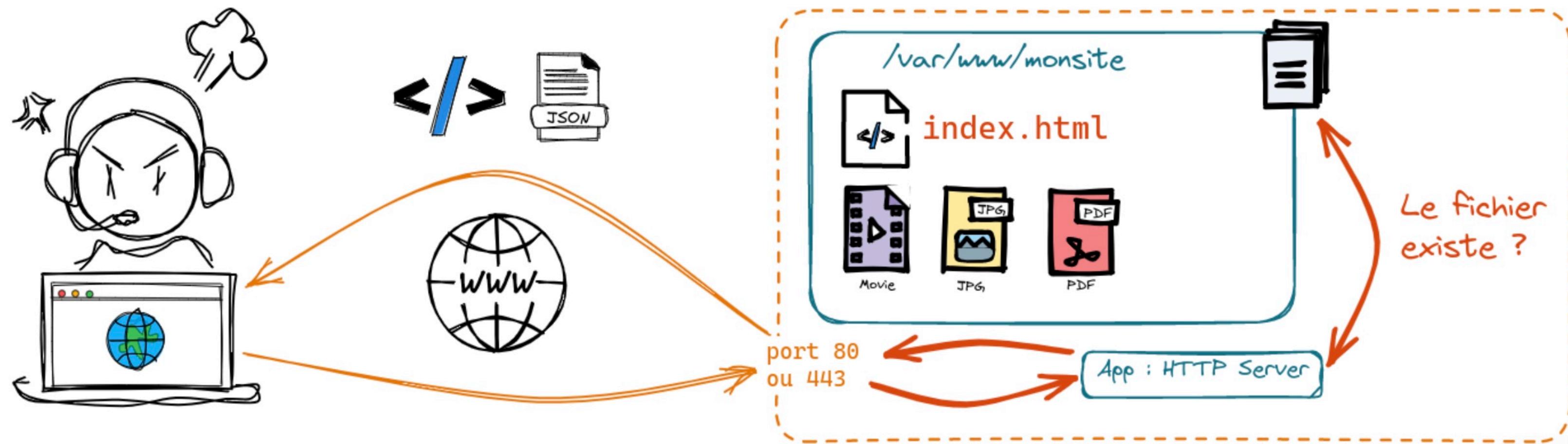
- ↓ Bloc d'instruction et portée des variables
- ↓ La variable a une durée de vie limitée
- ↓ Instructions conditionnelles



Site statique

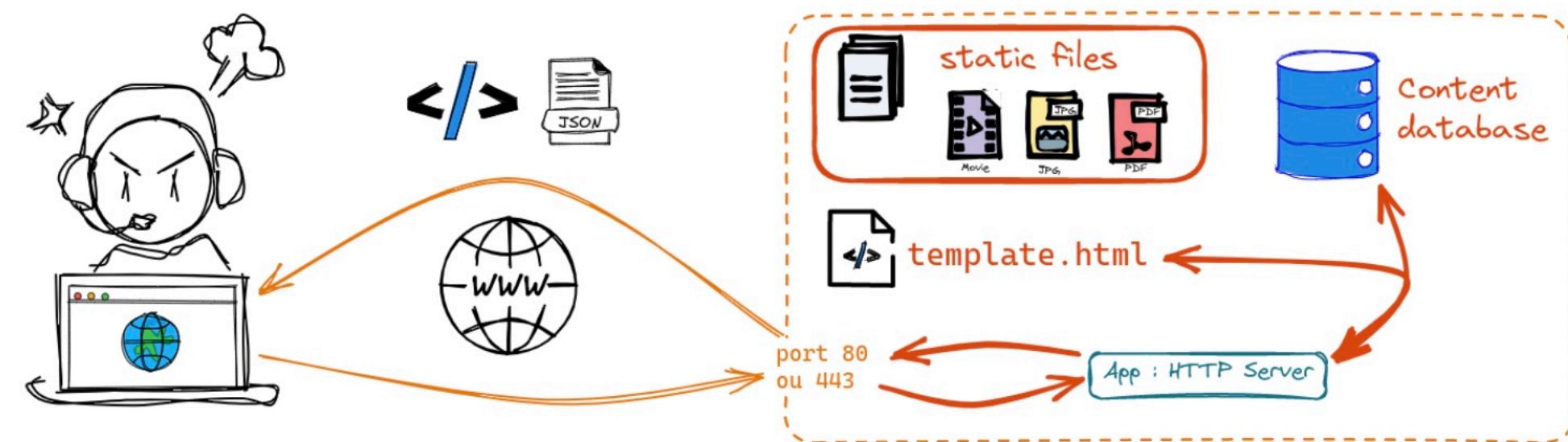
Le serveur http ne fait qu'une seule et unique chose

lire des fichiers html, png, jpg, pdf, et envoyer le contenu au client



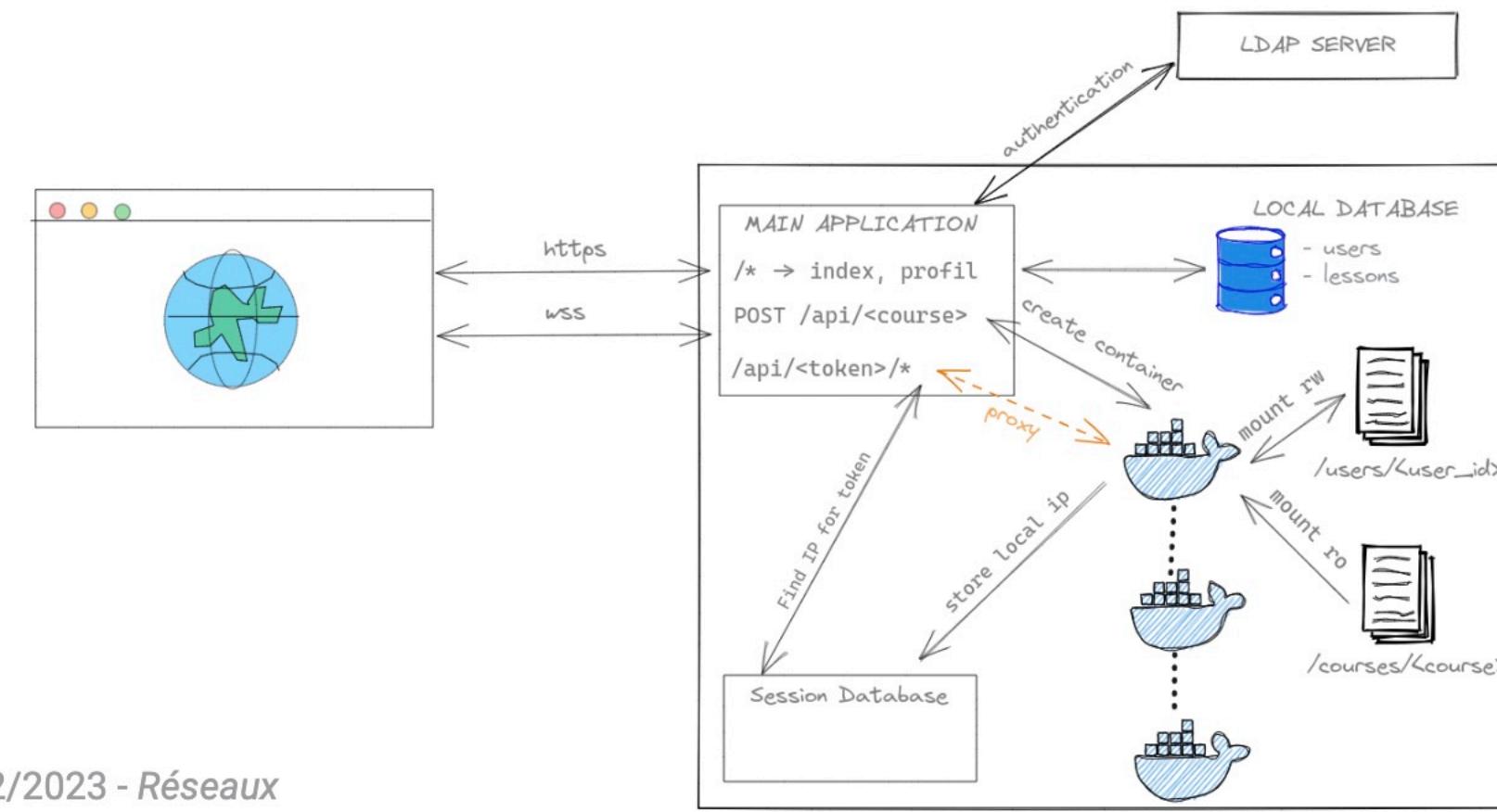
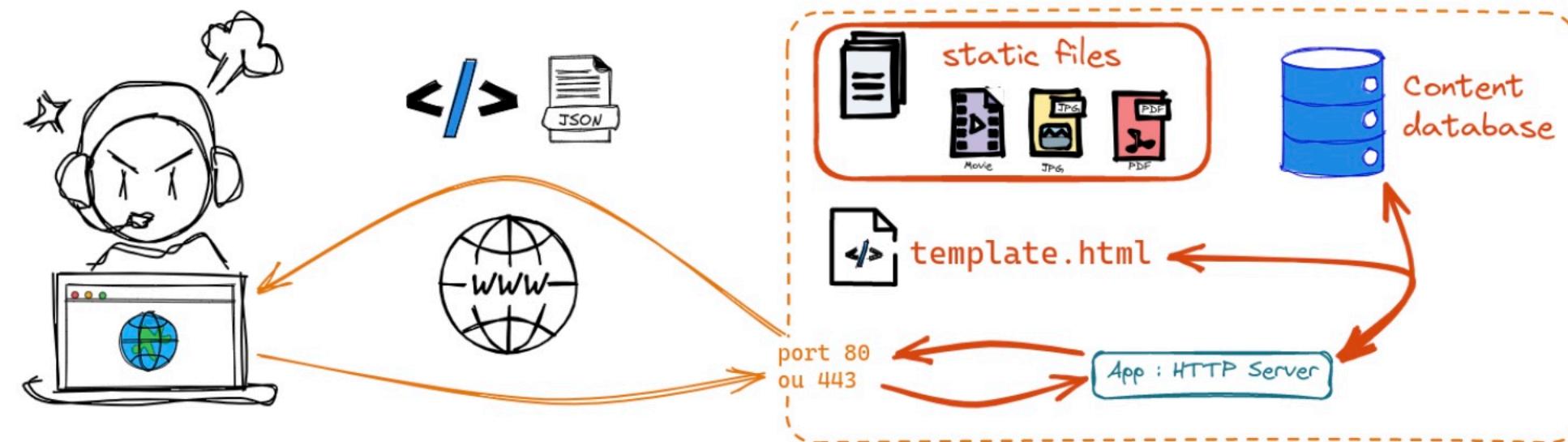
Site dynamique

Le serveur http va devoir travailler avec d'autre service afin de produire le résultat final pouvant être envoyé au client

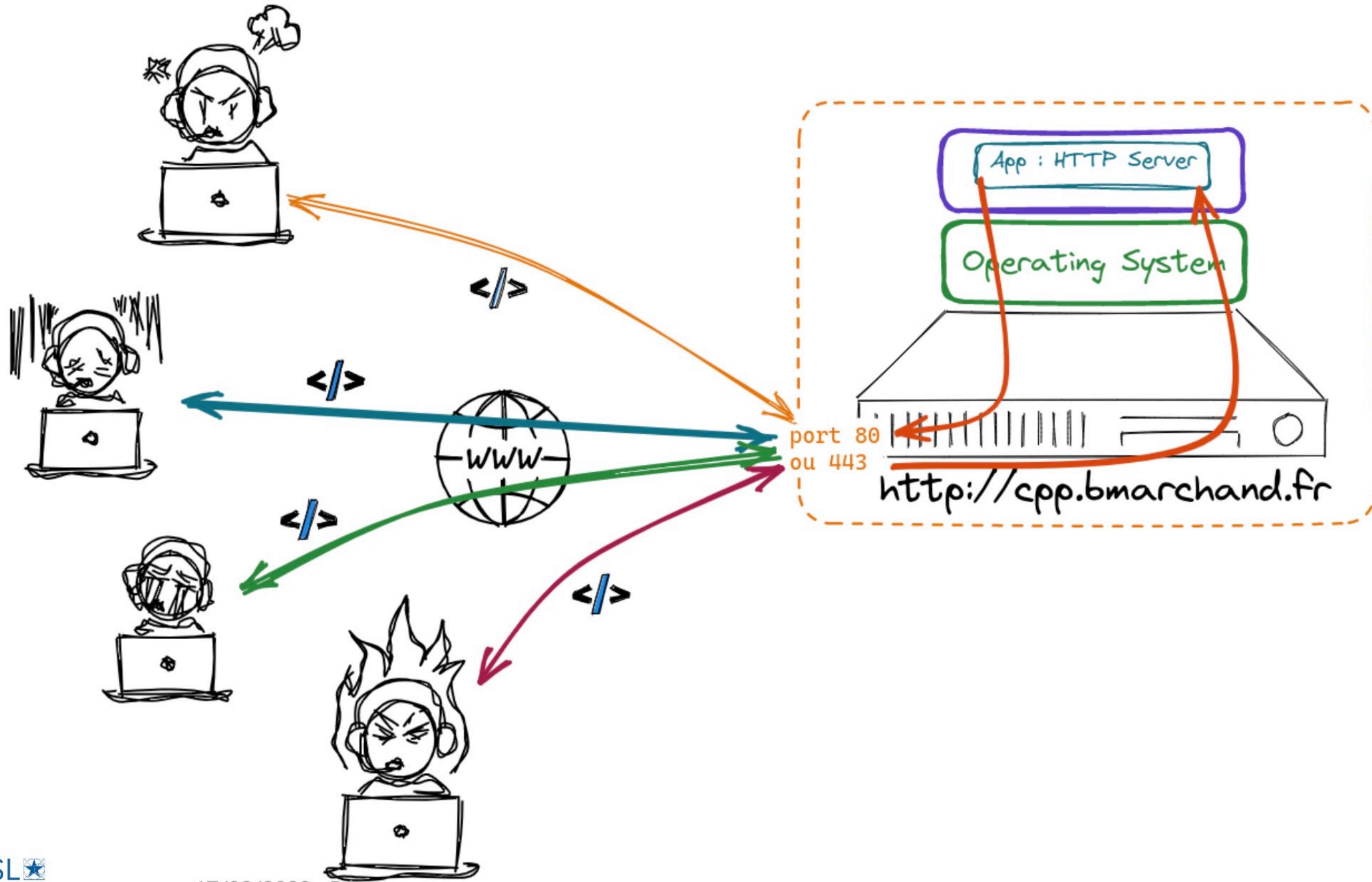


Site dynamique

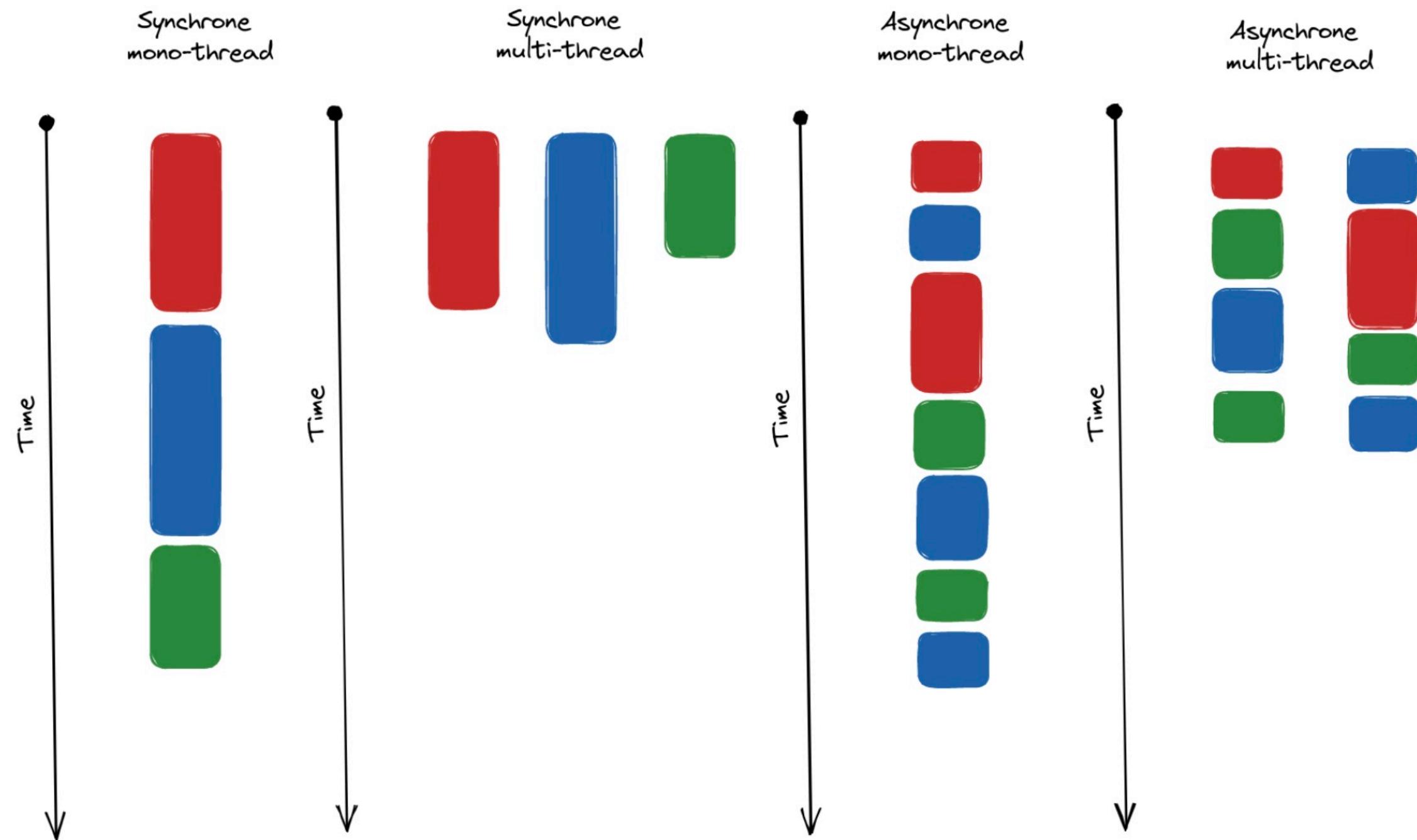
Le serveur http va devoir travailler avec d'autre service afin de produire le résultat final pouvant être envoyé au client



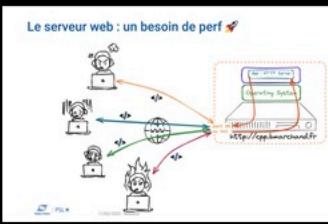
Le serveur web : un besoin de perf



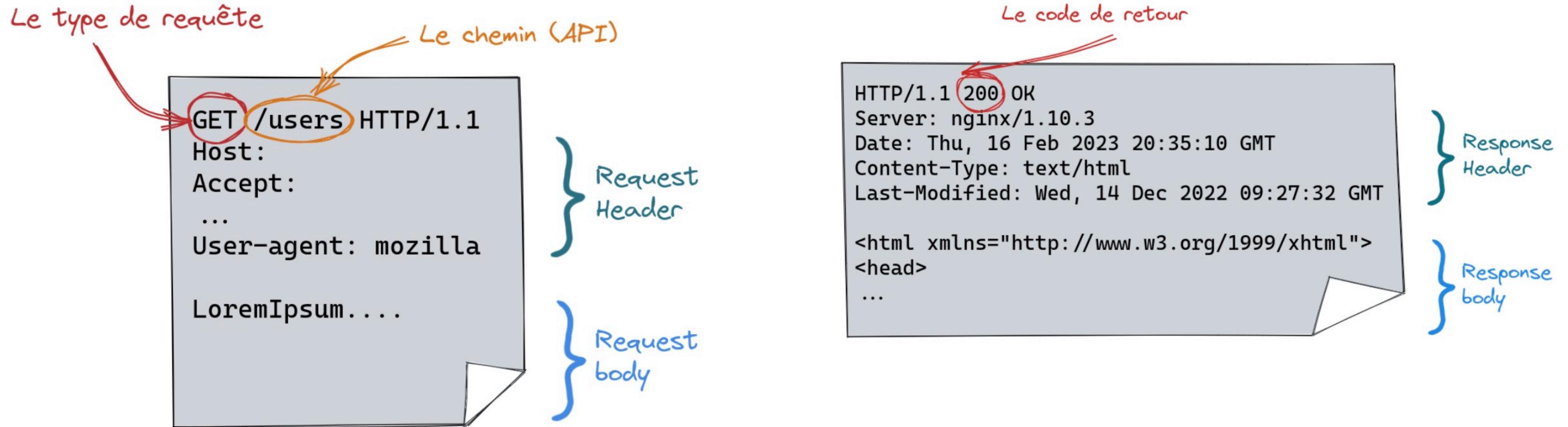
Solutions techniques



Utilisation du parallélisme de tâches processus/thread et/ou programmation asynchrone



Et au fait il répond quoi le serveur à GET ?



Possible de voir les requêtes et réponses dans votre navigateur via

Outils de développement > Network

Faisons un serveur http de base

<http://bit.ly/3EeuLLo>



⚠️ On regarde le fichier `minimal_server.py`



Au passage c'est quoi mon IP ?

Quand je suis un serveur comment je fais pour connaître mon IP ?

Il suffit de demander à TEST-NET-1,2,3 :

192.0.2.0 ou 198.51.100.0 ou 203.0.113.0

```
import socket
def find_my_ip():
    candidates = []
    for test_ip in ["192.0.2.0", "198.51.100.0", "203.0.113.0"]:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect((test_ip, 80))
        ip_addr = s.getsockname()[0]
        s.close()
        if ip_addr in candidates:
            return ip_addr
        candidates.append(ip_addr)

    return candidates[0]
```

⚠️ socket.SOCK_DGRAM connexion via UDP et pas TCP



Un truc un peu plus advance : définissons une API

<http://bit.ly/3EeuLLo>



⚠️ On regarde les fichiers `more_advance_server.py` et `more_more_advance_server.py`



On a fini ...

On a fini ...

... ou pas en fait



Un petit point sécurité



Quelle différence entre

HTTP et HTTPS

?



Un petit point sécurité

Quelle différence entre

HTTP et HTTPS

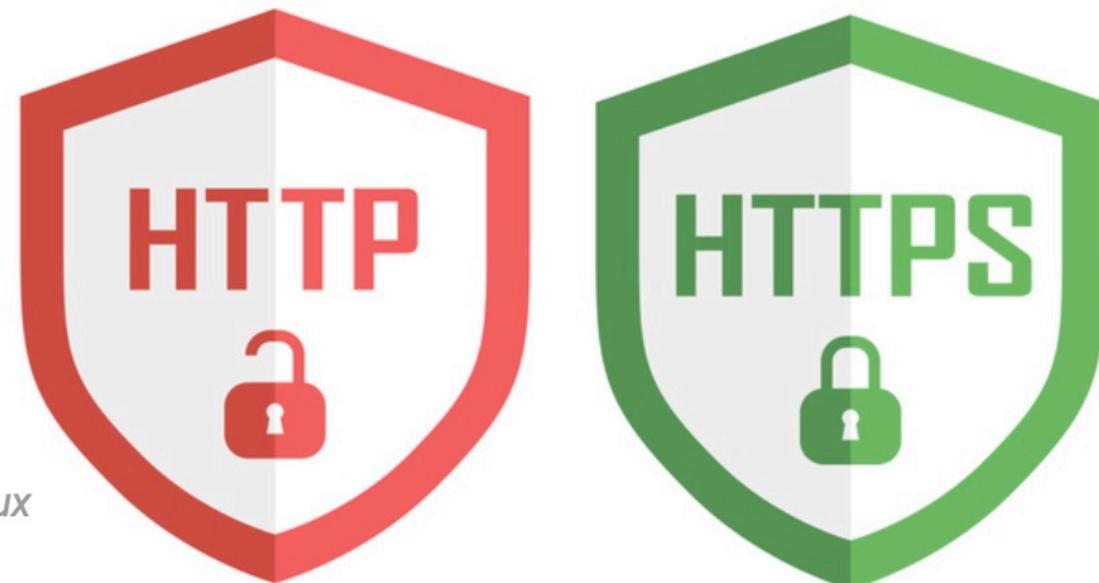
?

Oui oui c'est le **S** de Secure 😊

Grosso modo :

Enrobage du protocole HTTP dans une couche de chiffrement

pour garantir la sécurité de l'utilisateur



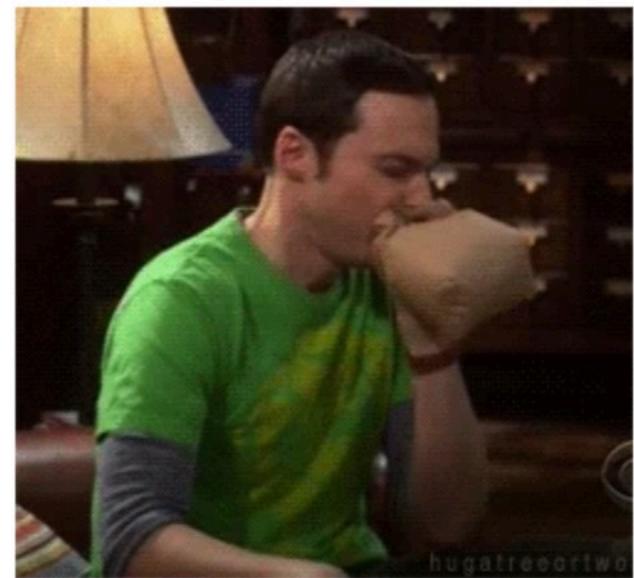
HTTP un truc pas safe ?

Alors oui le HTTP de base n'est pas sécurisé



HTTP un truc pas safe ?

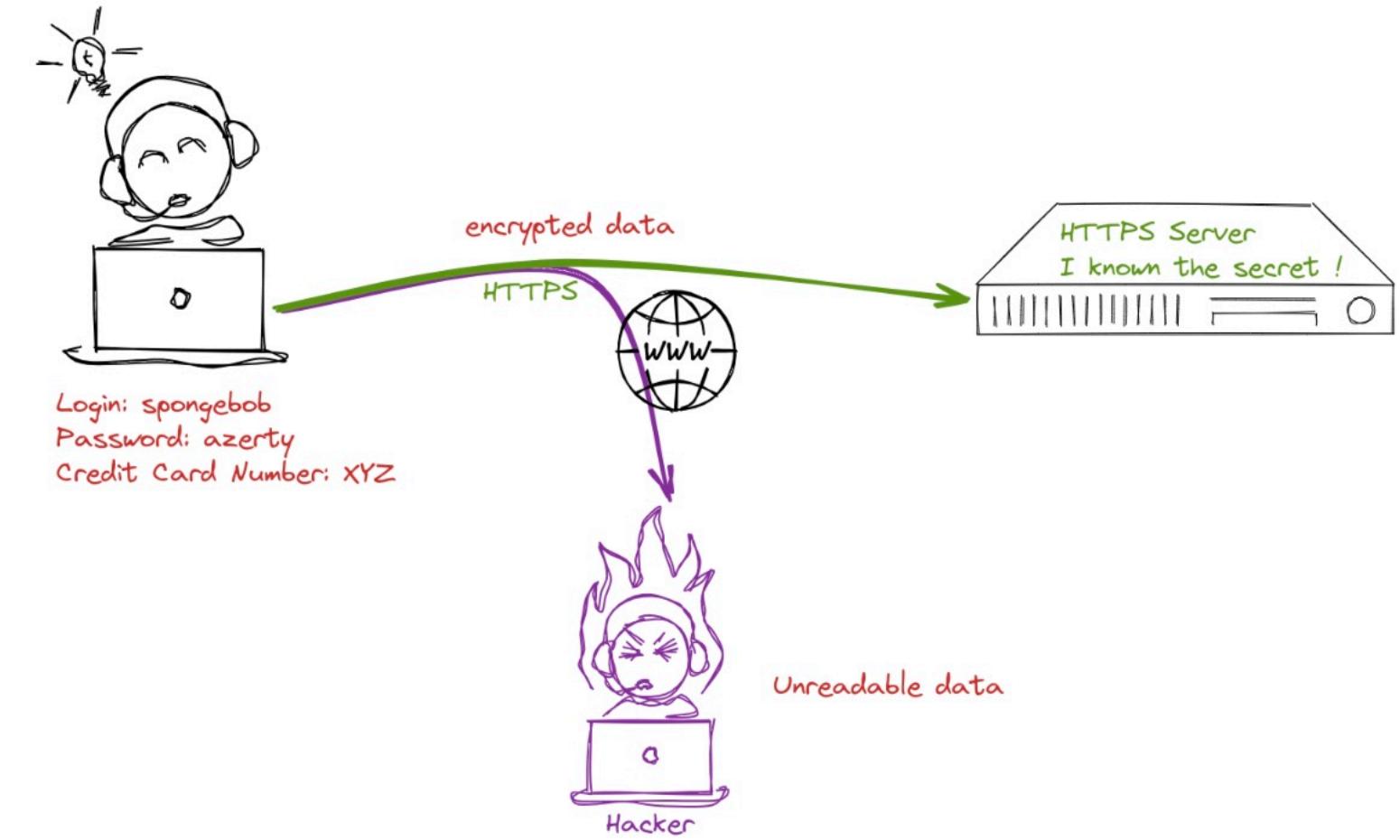
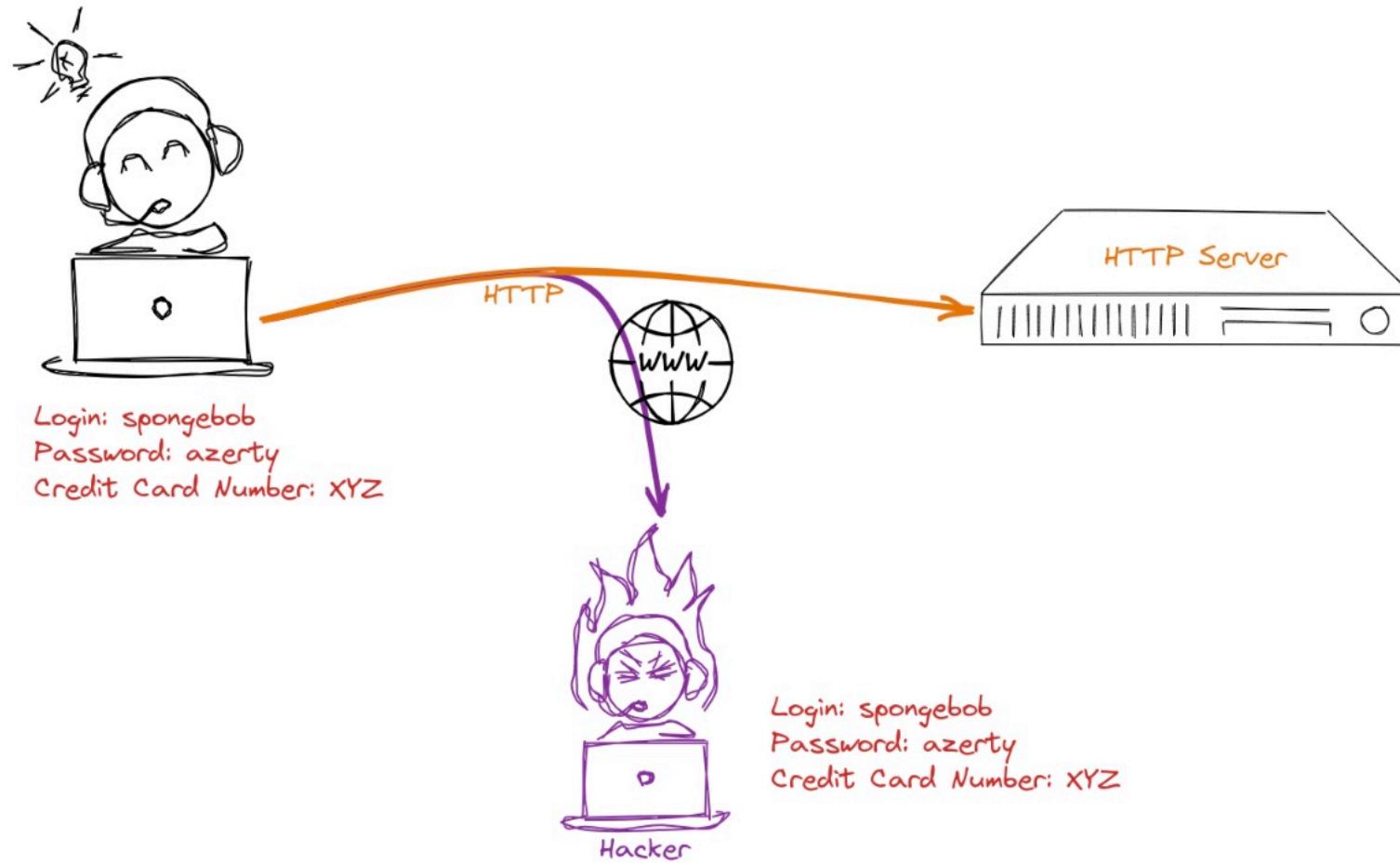
Alors oui le HTTP de base n'est pas sécurisé



Mais ce n'est pas très grave dans pleins de cas

A presentation slide with the title 'HTTP un truc pas safe ?' and a subtitle 'Alors oui le HTTP de base n'est pas sécurisé'. It features logos for MINES PARIS, PSL, and CNRS. A screenshot of the slide content is shown in the bottom right corner.

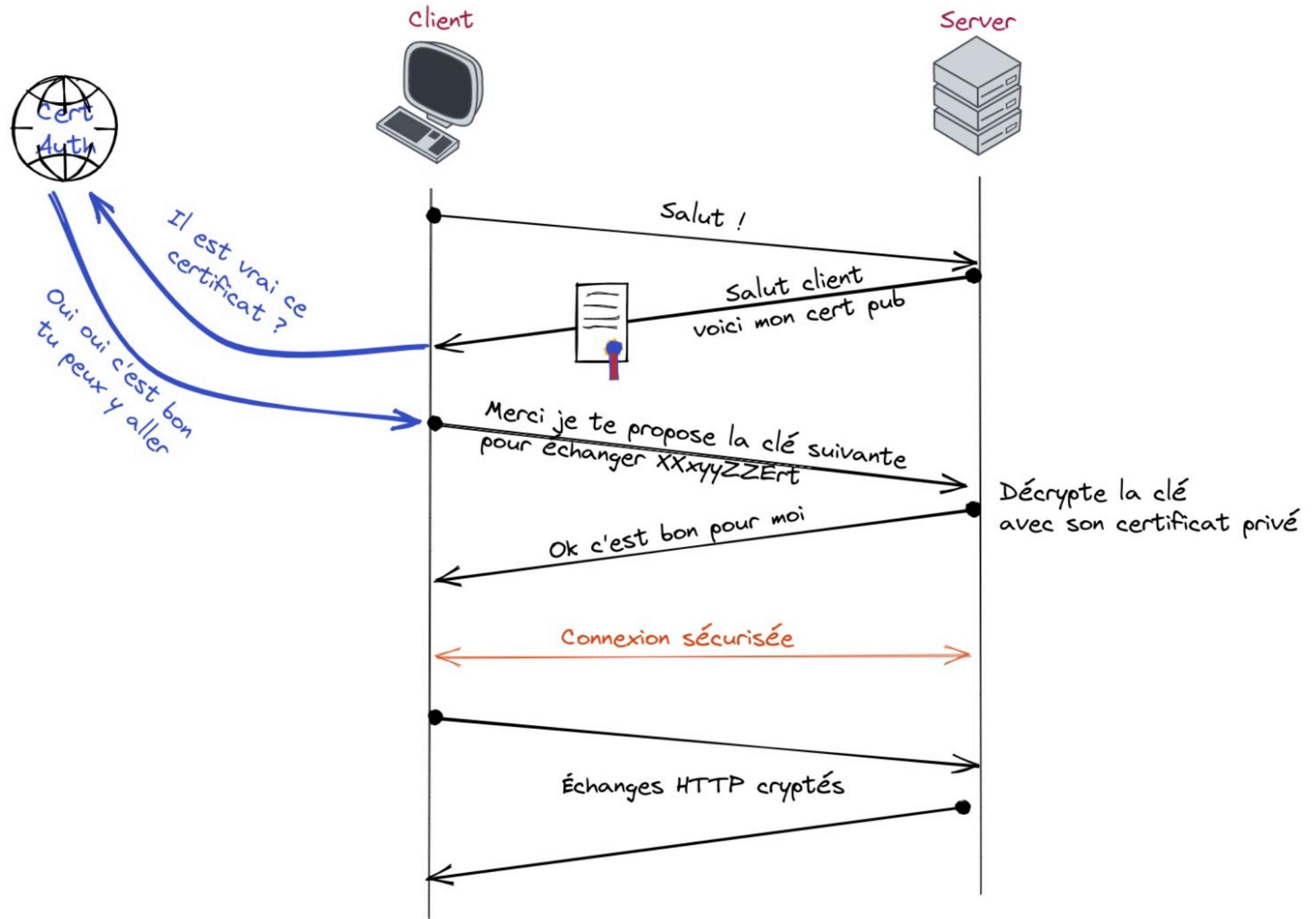
Le risque du HTTP



Le principe est donc de renfermer la requête HTTP et les informations qu'elle contient dans un message crypté

Principes de chiffrement

En pratique le chiffrement fonctionne avec un système clé publique/clé privée



Autorité de certification

Tiers de confiance

qui va générer les certificats permettant le chiffrement et l'authentification de l'identité des correspondants

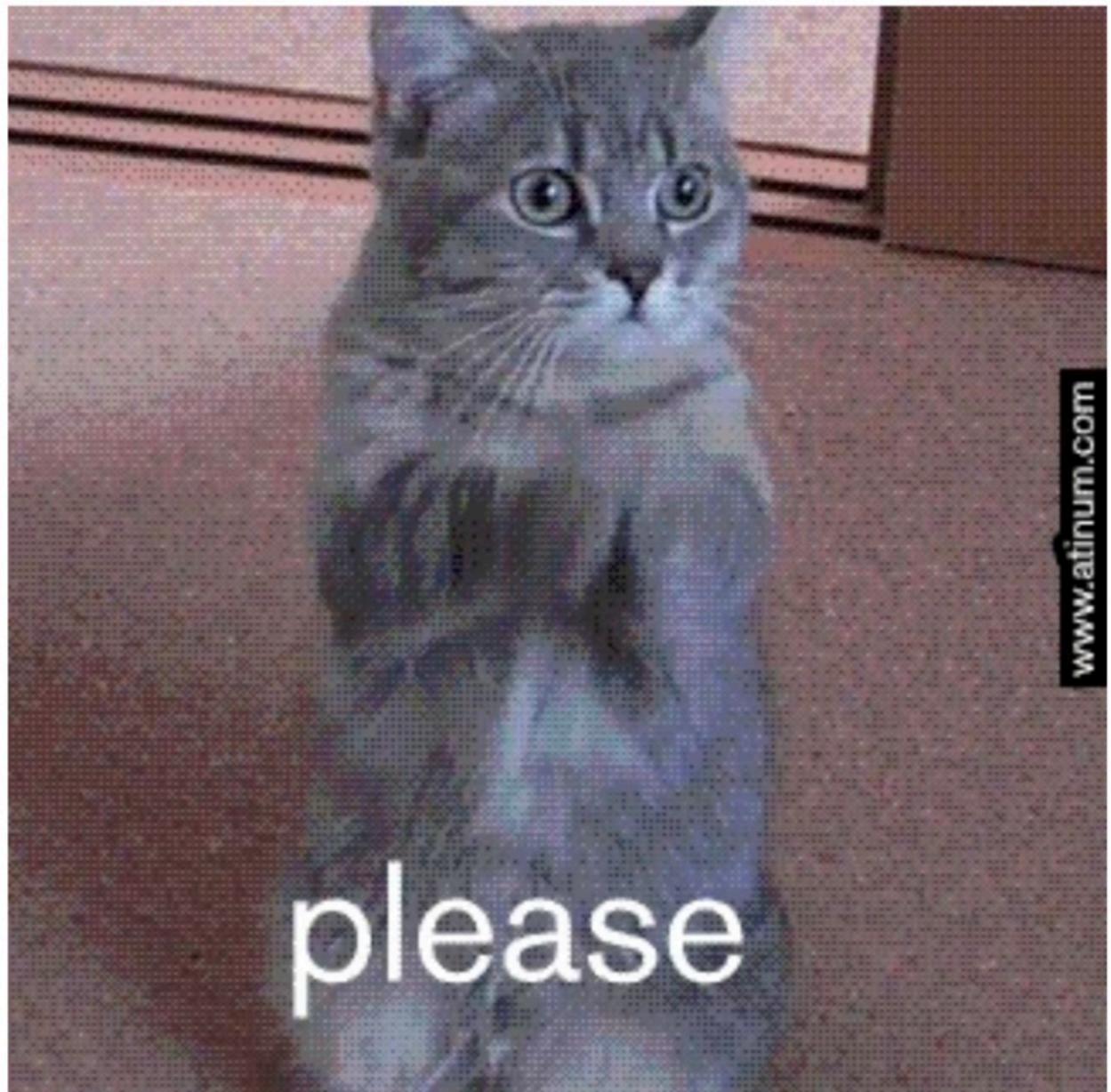
Possible de générer ses propres certificat soi-même mais ils ne sont pas considérés comme valide par les clients standards.



Pour générer des certificats gratuitement il existe l'initiative **Let's Encrypt**



Et maintenant c'est fini ?



Pause Exercice

Objectif :

Mettre en place un serveur HTTP qui sait gérer une requête POST et une requête GET

Use-case :

1. Je fais une requête POST sur votre-url/api/register dans laquelle je peux mettre un json de la forme {name: "Basile", secret: "A secret token"}
2. Votre serveur doit stocker l'information (à tout hasard dans un dictionnaire)
3. Je peux faire une requête GET sur votre-url/info/<name> qui doit me renvoyer un bout de html de la forme

```
<h1>Hello NameFromPOST your token is TokenFromPOST</h1>
<h2>We are Student X, Student Y, ...</h2>
```

Mettez vous par groupe de 2~3 et vous avez **30 minutes max.**

A la fin envoyez moi l'url de votre serveur et on test en réel !

```
import requests
import sys

url = "your url"
name = "Sponge Bob"
token = "un truc au pif"
### POST
resp = requests.post(f"{url}/api/register",
                     json={"name": name, "token": token})
if resp.status_code != 200:
    print("Y a un problème dans le post")
    sys.exit(1)
### GET
resp = requests.get(f"{url}/info/{name}")
print(resp.content.decode())
```



Les cookies



Faisons une pause goûter 😊

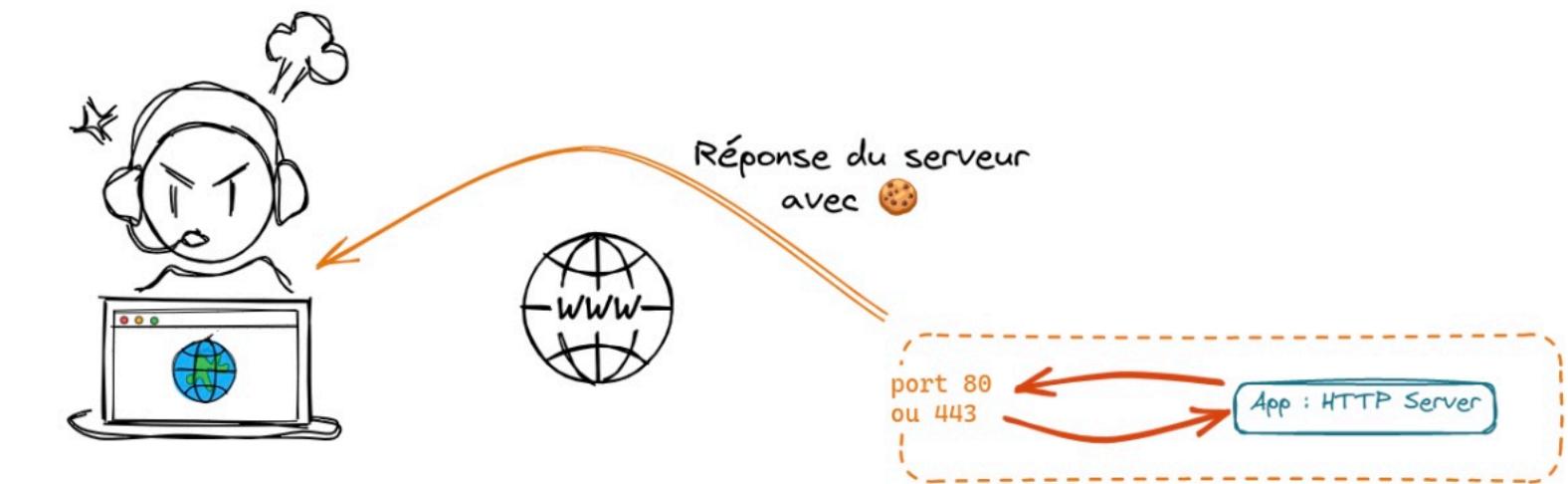
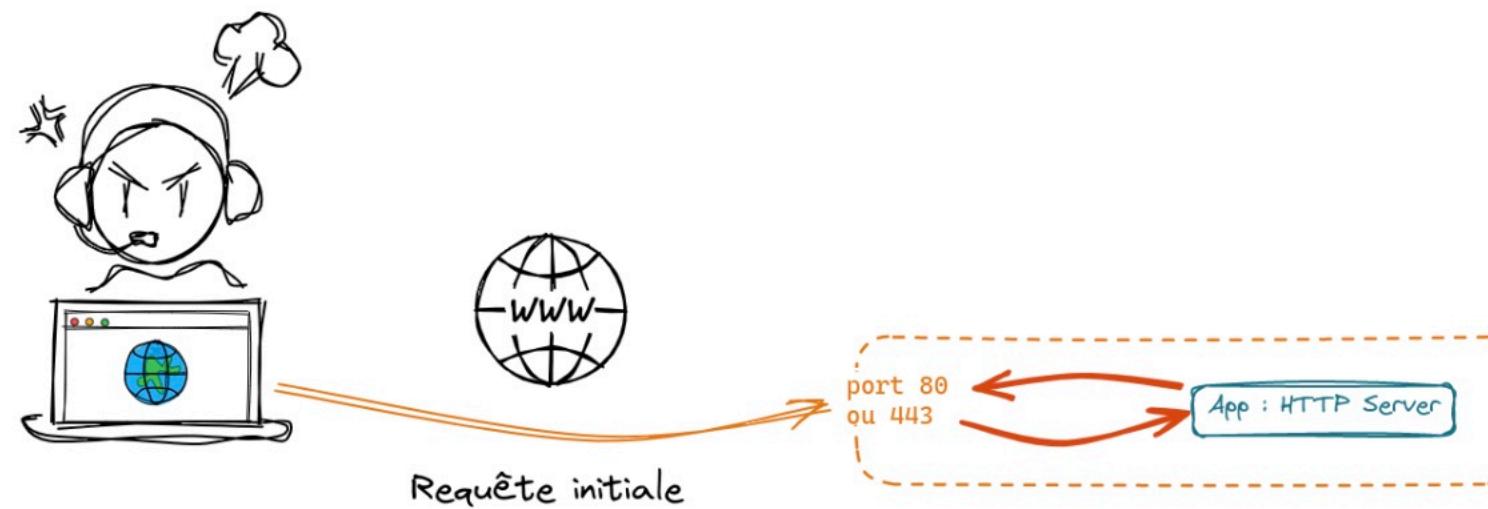


Ca fait parti de ces petites choses **cachées** dans le header des réponses http.

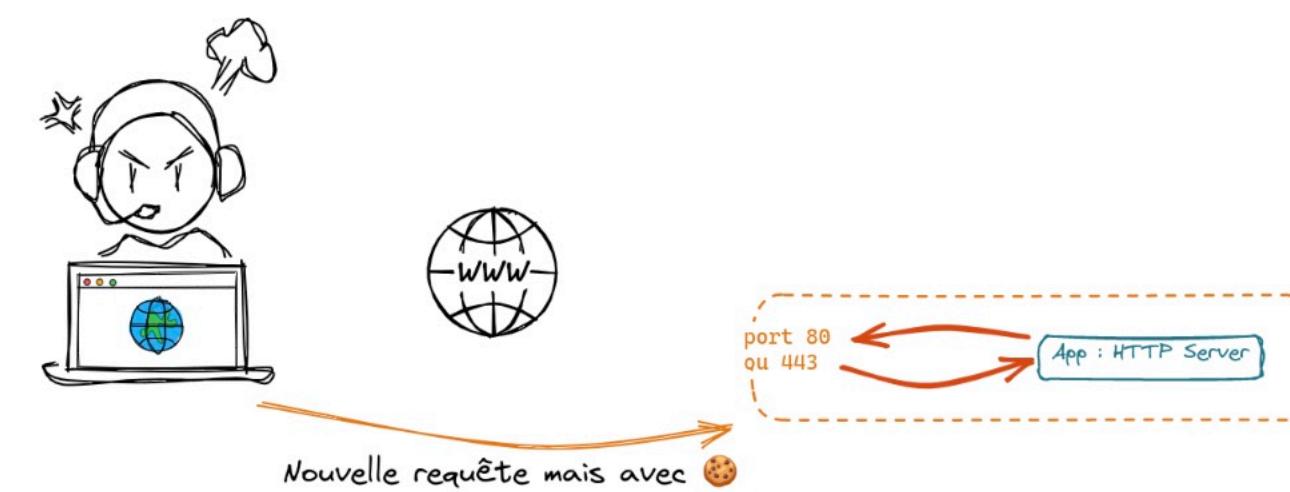
Concrètement c'est quoi ?

Un 🍪 HTTP c'est

données qu'un serveur envoie à un client



stockée sur le client (dans le navigateur)
renvoyée au serveur à chaque nouvelle requête



Quel intérêt ?

Les cookies sont là pour enrichir le HTTP.

HTTP = protocole sans état

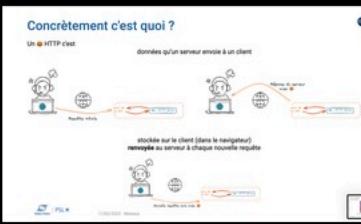
En gros impossible pour un serveur HTTP de savoir si deux requêtes viennent d'un même client ou pas 🤯
comment rester authentifier alors ?

La solution

Les cookies 🍪 parce que ça laisse des miettes

Concrètement on va pouvoir stocker :

Un session ID, des préférences utilisateur (light/dark theme, langue, ...)



Mettre des cookies

Rien de plus simple, dans l'en-tête de la réponse serveur à une requête client il suffit d'ajouter

Set-Cookie: <name>=<value>; <attributs...>

Attributs de Cookie

- Expires : durée de vie (date/heure)
- Max-Age : durée de vie (seconde)
- Domain : noms de domaine pour lesquels le cookie est renvoyé [par exemple](#)
- Path : chemin particulier pour lesquels le cookie est renvoyé
- Secure : si autorise ou pas l'envoie via HTTP et non HTTPS
- HttpOnly : si autorise ou pas l'accès via autre chose de du http(s)



Quelques règles à suivre



<https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies>
[\(https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies\)](https://www.cnil.fr/fr/cookies-et-autres-traceurs/regles/cookies)

- Internautes doivent être informé et donner leur consentement avant le dépôt de certains cookies
 - ✗ Traçage publicitaire / réseaux sociaux
 - ✓ Cookie pour dire qu'on refuse les cookies exemple, panier d'achat, authentification, ...
- Recueillir le consentement
 - Bouton refusé aussi visible que celui accepté
 - Possibilité de choisir les cookies
 - Facilité de retrait du consentement

Rajoutons un Cookie dans notre serveur

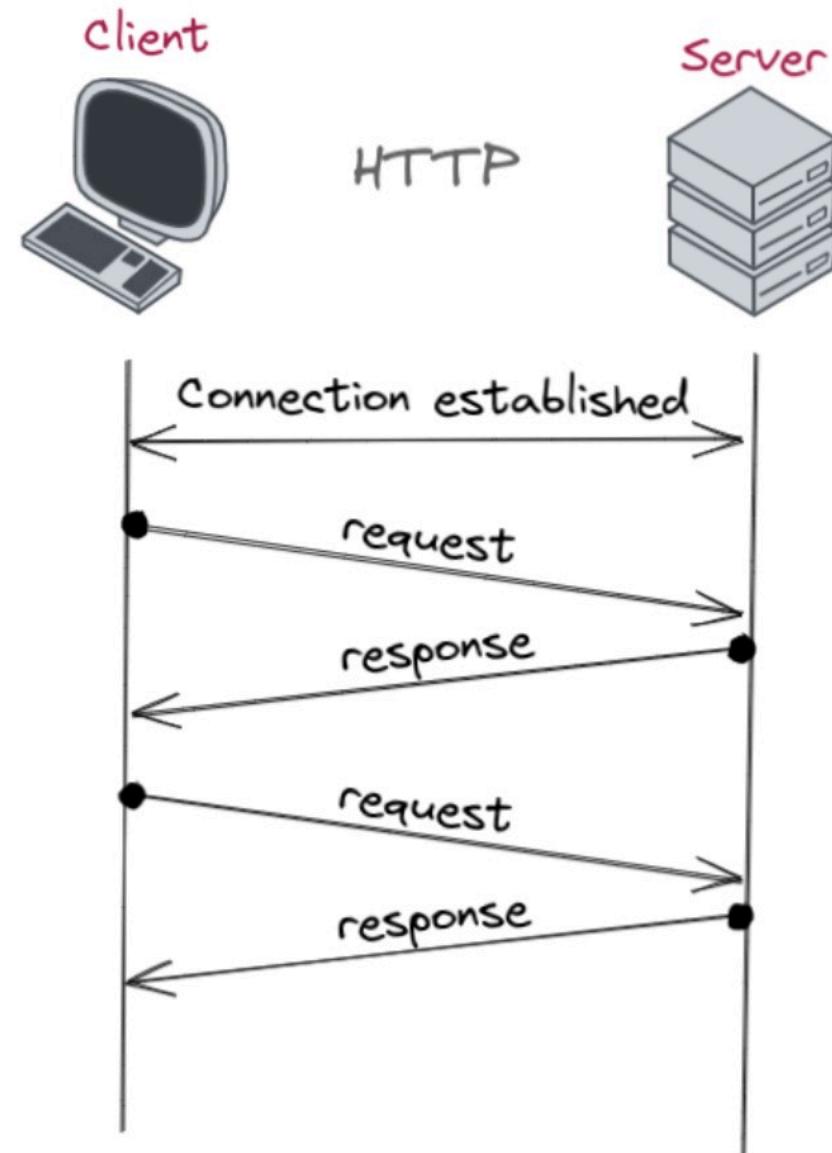
<http://bit.ly/410qbdD>



HTTP + 🍪 suffisant pour tout faire ?



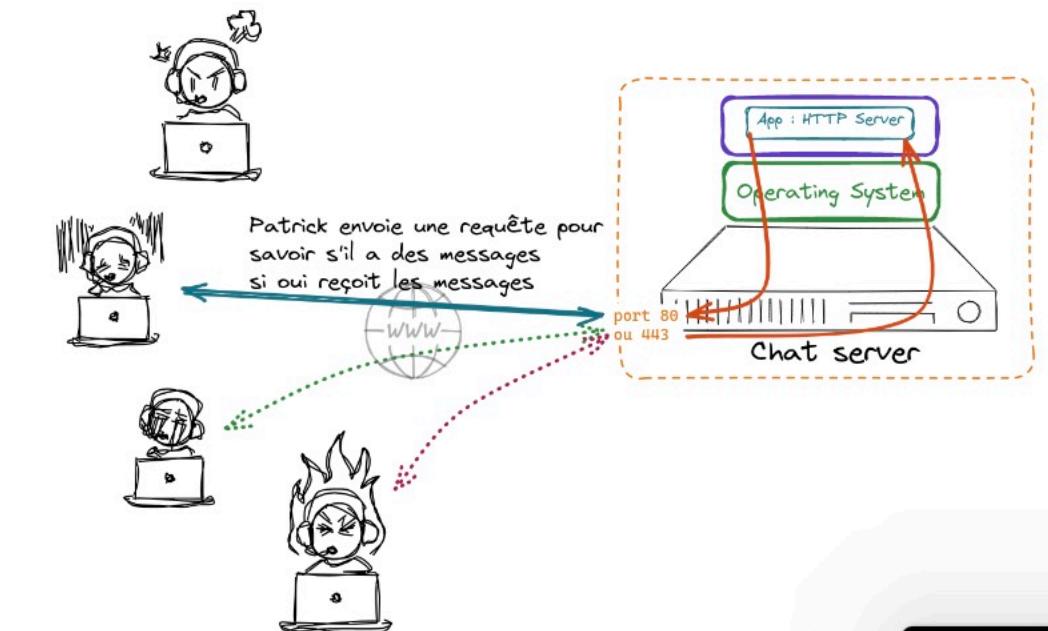
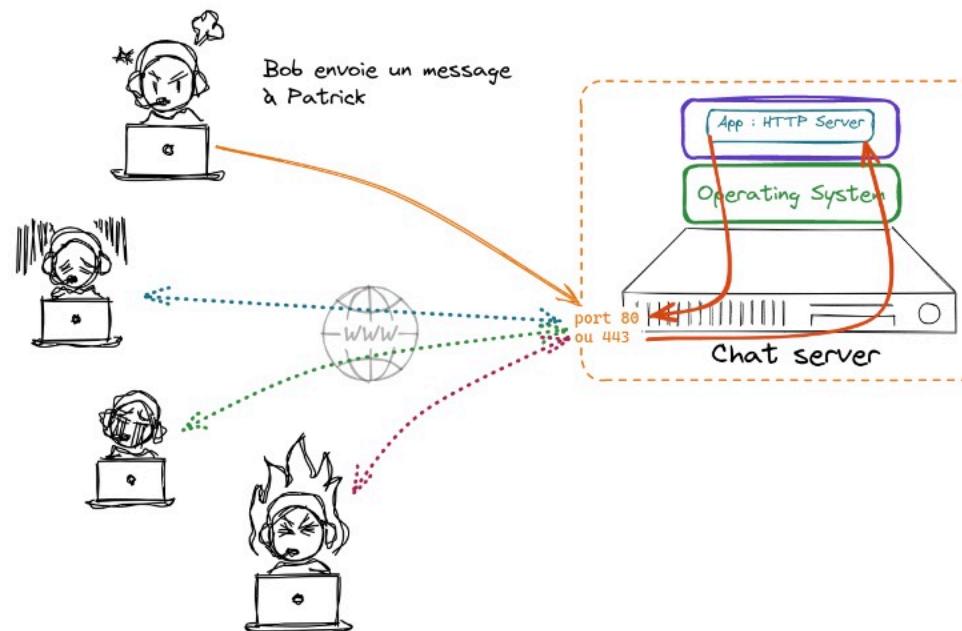
Mais pourquoi ?



Fonctionnement de HTTP très rigide

question/réponse

Impossible pour le serveur d'être à l'origine de l'échange
Assez limitant en fait 😞



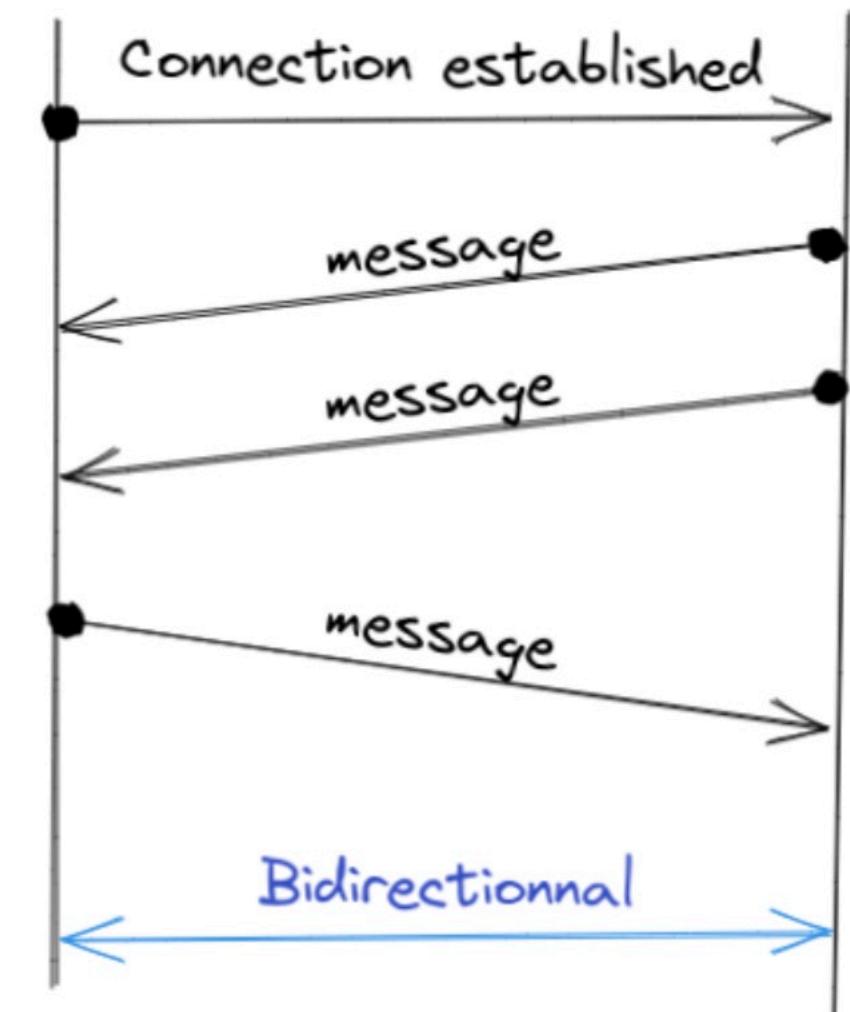
Websocket

En 2011 révolution arrivée de Websocket 😱

connexion **bidirectionnelle** entre un client et le serveur

on parle de connexion *full-duplex*

Permet au serveur de **pousser** des informations vers le client sans que ce dernier n'est rien demandé 😲



Comment ça marche

Très simplement en fait !

Première étape on établie une connexion vers un serveur WebSocket
via

ws://mon-super-server.com ou wss://mon-super-server.com

Une fois la connexion établie

on doit simplement se mettre en état d'écoute à des évènement particulier

Quatre types d'évènements

onopen , onclose , onerror , onmessage 

Et à chaque évènement on va venir associer une action



Par exemple :

```
var socket = new WebSocket("ws://localhost:3060/ws");

socket.onopen = function (e) {
    alert("[open] Connection established");
    alert("Sending to server");
    socket.send("My name is John");
};

socket.onmessage = function (event) {
    alert(`[message] Data received from server: ${event.data}`);
};

socket.onclose = function (event) {
    if (event.wasClean) {
        alert(`[close] Connection closed cleanly,
              code=${event.code} reason=${event.reason}`);
    } else {
        // e.g. server process killed or network down
        // event.code is usually 1006 in this case
        alert("[close] Connection died");
    }
};

socket.onerror = function (error) {
    alert(`[error] ${error.message}`);
};
```

```
from tornado.websocket import websocket_connect

def on_message( msg ):
    print(f"[In on message] {msg}")

ws = await websocket_connect("ws://localhost:3060/ws",
                            on_message_callback=on_message)

await ws.write_message("coucou")
await ws.write_message("byebye")
await ws.write_message("vive la MMC")
```

⚠️ Vous voyez apparaître le mot clé `await` que vous ne connaissez pas en Python 🐍

C'est lié à la programmation asynchrone. Pour plus de détail je vous encourage à faire un tour sur le Mooc

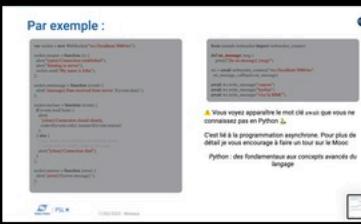
Python : des fondamentaux aux concepts avancés du langage



En pratique

Une messagerie instantannée !

<http://bit.ly/3xu599H>



Un mot sur les Framework



Flask
web development,
one drop at a time

django

 **Laravel**


node
express


RAILS


spring
Framework

In the next episode



Un tour d'horizon du **Framework Flask**
qui va vous simplifier la vie pour tous les développements Web