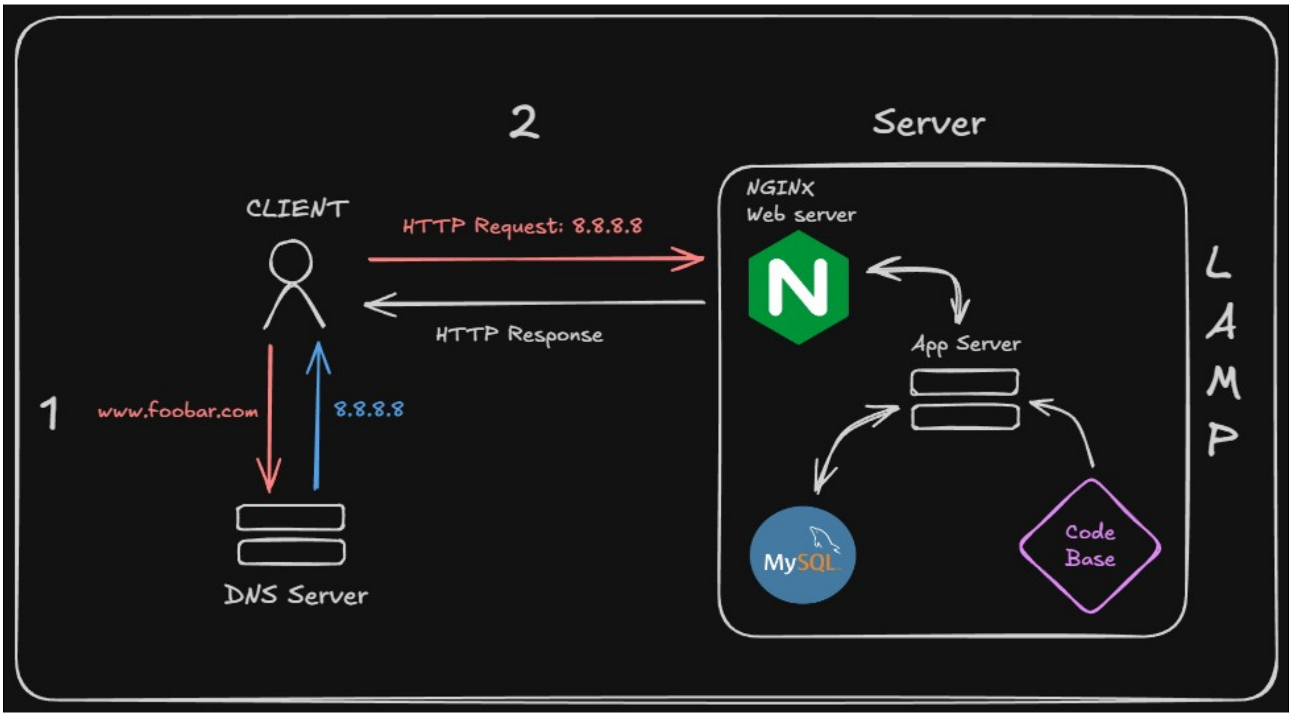
# Web Infrastructure

A web infrastructure is essential for websites to operate. It refers to the arrangement of the components and technologies required to support operations of web applications, websites and online services in order for them to work. Each technologie have a certain rôle : security, availability, performance, scalability… The web infrastructure determined what software/hardware are used and how they interact with each other, to deliver data to the client.

## Single Web Infrastructure



Here you have a single web infrastructure which is composed of one single server with a LAMP stack, wich provide everything an application needs to be hosted :
- L : Linux → run the environnement
- A : Apache → The web server
- M : MySQL → the database systeme
-P : Python/PHP → the Back-End languages

We will detail how the infrastructure works and what are the terms used on this schema :

- First of all, our client send a request in order to access our Website by typing his Domain Name.

The Domain Name is a human-readable adress of our Website, since devices find other machines with their IP adress, which is a sequence of decimal numbers and not very convenient to memorize for human.

In ‘www.foobar.com’ the ‘www’ is a ‘’hostname’’ type subdomain, meaning a standard subdomain without any special category. It has no particular sttatus in the DNS : it’s simply a name like any other subdomain.

So to find the IP adress our computer will request the IP adress associated with the domain name sent to the server of our internet service provider, wich, if it does not find it in the cache, will send the request back to the DNS servers, who are made to find IP adresses. They will return a response which will come back to the client.

- Then when it got the IP adress of the server, our client will send an HTTP Request to it and the web server will analyse it and send it to our application server that will use our code base to manage the data and will communicate with the database if it’s necessary.

The Web Server (Nginx): Nginx serves as a key component in managing user requests. It acts as a gateway, receiving and interpreting incoming HTTP requests from users’ browsers. From there, Nginx forwards these requests to the application server, making sure users get the web content they’re looking for.

The Application Server: The application server runs the server-side code that powers the website’s logic and dynamic features. It interacts with the database to read or write data, then returns the appropriate response to the web server.

The Database (MySQL): MySQL is the system where all the data needed by the website is stored and managed. It can hold a wide range of information, including user accounts, product details, and much more. The application server interacts with the database to read or write data whenever necessary, helping the website run reliably and efficiently.

- Finally the application server will send a response to the web server that will return all the way back to the client.

All of this operations happend under several protocoles as TCP/IP in the network ensuring that the client can acces our website.

## **<u>ISSUES of the infrastructure</u>**

The infrastructure has however some limitations :

- SPOF Single Point Of Failure : The infrastructure relies on a single server which makes it susceptible to downtime if hardware fails, the network experiences issues, or other problems occur. Adding redundancy or implementing load balancing would significantly improve resilience and help ensure the website remains accessible.

- Downtime During Maintenance: Updating the website or deploying new code may require restarting the web server, which can temporarily take the site offline. Introducing redundancy or using load balancing can help avoid interruptions and keep the website available during maintenance operations.

- Limited Scalability: With this setup, the website may have difficulty handling a large surge in incoming traffic. To manage traffic spikes effectively, it would be necessary to scale horizontally by adding more servers or vertically by increasing the resources of the existing server.