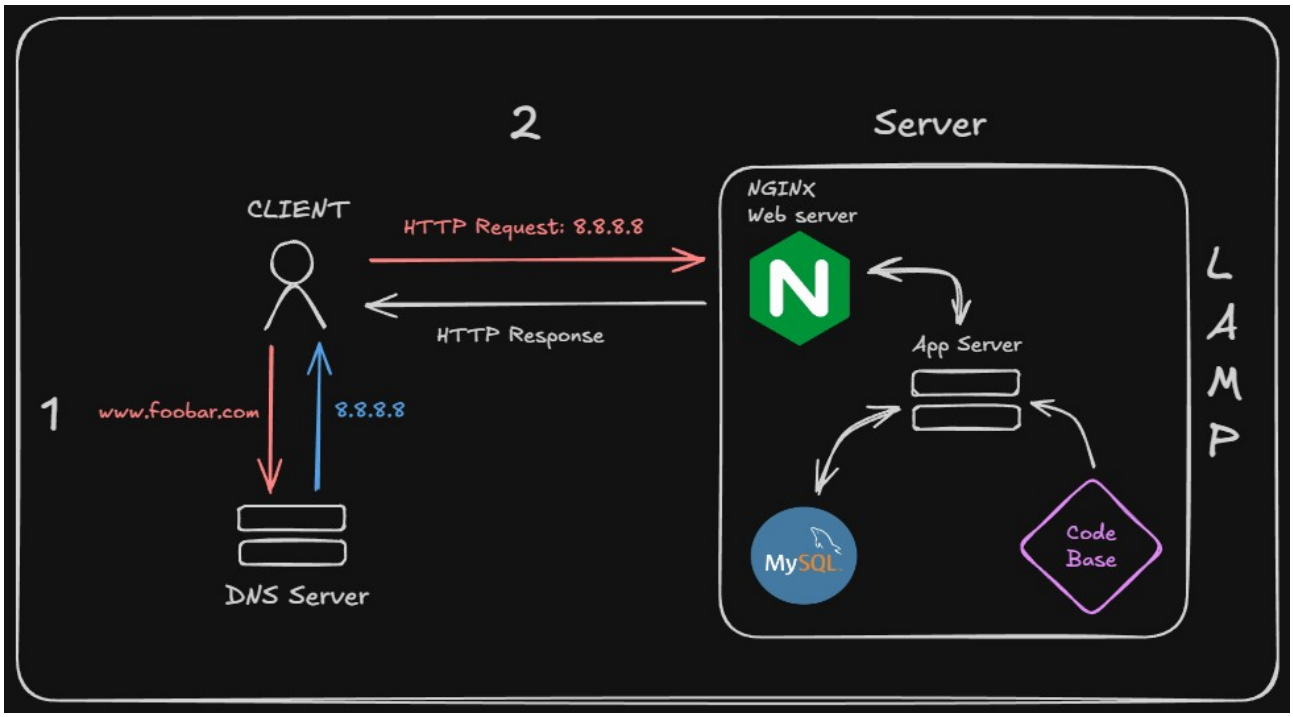
Web Infrastructure

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.

ISSUES of the infrastructure

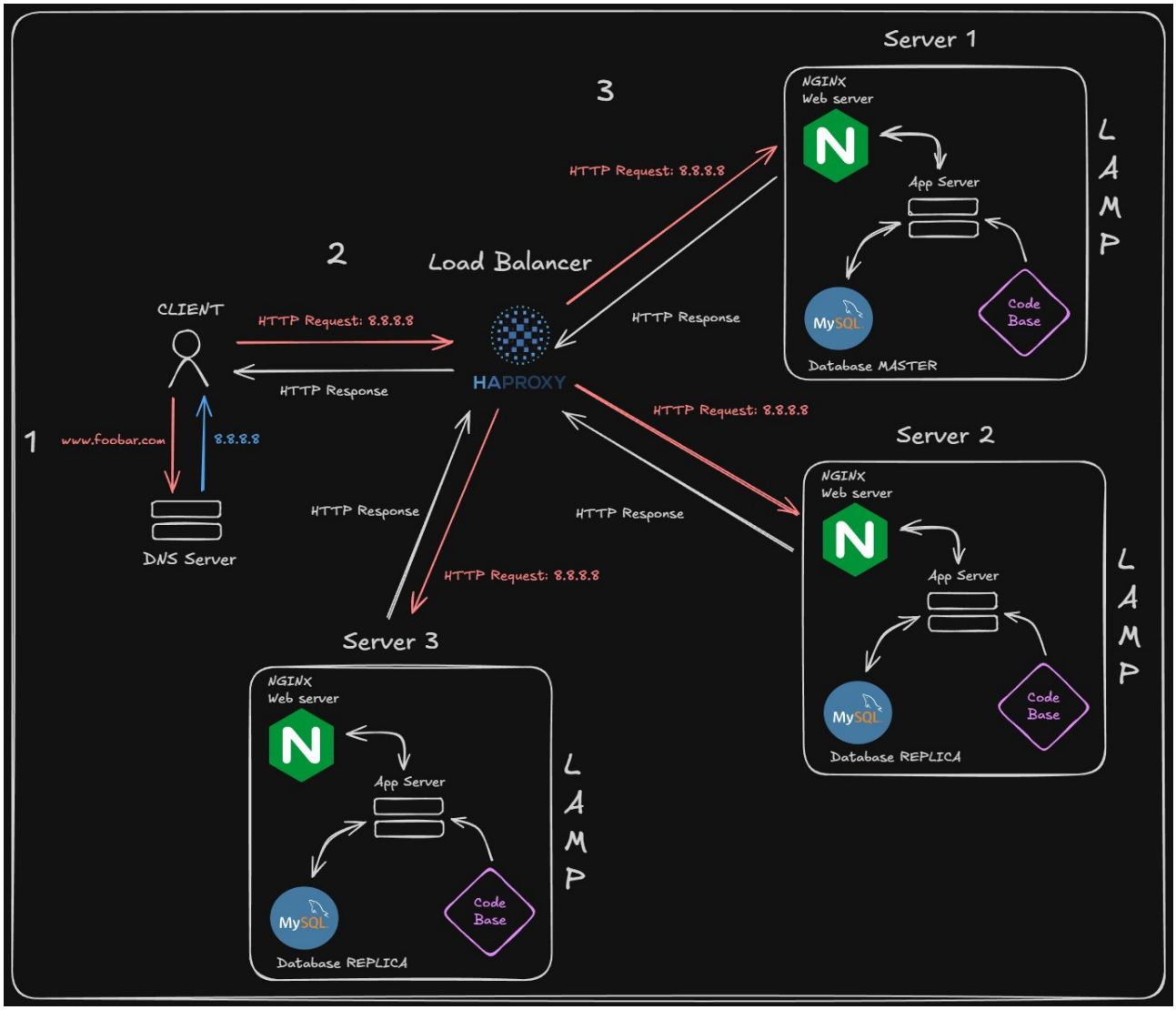
The infrastructure has however some limitations :

- 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.

Distributed web infrastructure



To adress the drawbacks of a single-server infrastructure, we added elements to our infrastructure that we will details on the why, what and how :

Server 1:
This server hosts the web server (Nginx), the application server, the MySQL database, and the code base. It functions as the primary machine responsible for delivering web content and handling the application’s logic and data operations.

Server 2 and 3:
These servers are an exact replica of Server 1, mirroring its configuration and data. It acts as a backup and provides redundancy in case Server 1 encounters any failures.

Load Balancer (HAProxy)
The load balancer—HAProxy in this setup—distributes incoming traffic across both servers. It helps balance the workload and provides failover capabilities by redirecting traffic to the remaining server if one becomes unavailable.

MySQL Master-Replica Cluster

A MySQL Master-Replica setup ensures data synchronization between the two servers. The Master handles all write operations, while the Replica continuously syncs with the Master and is used for read-only queries.

Application Files (Code Base)

The application's code and files are deployed on Server 1 and Server 2, 3. This redundancy ensures that the application can continue running smoothly even if one of the servers goes down.

Why ?

Using three servers increases redundancy and improves high availability. If the primary server fails, one of the replica servers can take over, significantly reducing downtime and ensuring continuous service.

The load balancer distributes incoming traffic across all three servers. It prevents any single server from becoming overloaded and provides seamless failover by redirecting traffic to healthy nodes if one becomes unavailable.

For the databases with MySQL, this configuration uses one Master and two Replica nodes to maintain data consistency across the infrastructure. The Master handles all write operations, while both Replicas continuously synchronize with it and can serve read-only queries. This setup helps balance database load and improves overall performance.

This architecture remains closer to an Active-Passive model. The Master server actively manages both read and write operations, while the two Replica servers act as passive nodes, handling only read operations and standing by to take over if the Master fails.

ISSUES :

Single Point of Failure (SPOF):

The load balancer remains a potential single point of failure. If it goes down, traffic can no longer be routed to the servers. Introducing a redundant load balancer would help eliminate this risk and maintain continuous traffic distribution.

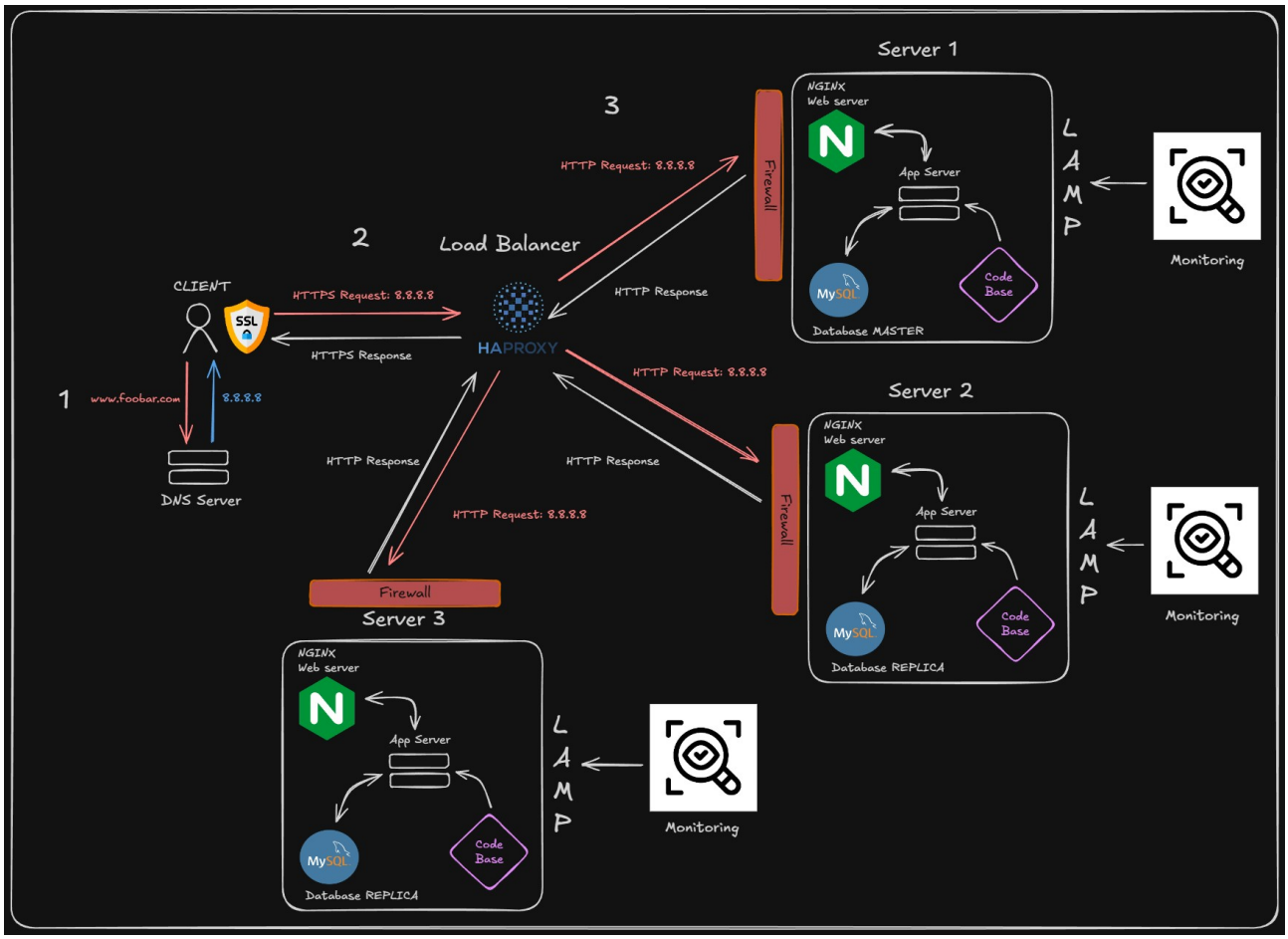
Security Issues:

The current setup lacks essential security measures such as a firewall and HTTPS. Adding a firewall would help filter and protect incoming traffic, while enabling HTTPS is critical for encrypting data in transit and ensuring secure communication.

Monitoring:

There are no monitoring tools in place, making it difficult to detect issues early or maintain visibility over system performance. Implementing monitoring solutions is key to proactively identifying problems and ensuring the overall health of the infrastructure.

Secured and monitored web infrastructure



What we add and why :

Firewalls for Enhanced Security:
Security is a critical aspect of any web infrastructure. To strengthen protection against threats and unauthorized access, three firewalls are deployed—one for each server. These firewalls filter and control both incoming and outgoing traffic, acting as gatekeepers that inspect data packets, block malicious activity, and enforce security policies.

SSL Certificate for HTTPS:
To secure data in transit and protect user privacy, an SSL certificate is essential. It is configured to serve **www.foobar.com** over HTTPS, ensuring that all communication between users' browsers and the web server is encrypted and confidential. HTTPS is fundamental for safeguarding sensitive information and preventing interception or tampering.

Monitoring for Infrastructure Health:
Monitoring plays a vital role in maintaining the stability and performance of the infrastructure. To support this, three monitoring clients are installed. These agents continuously gather information about server performance, traffic behavior, security events, and other key metrics.

Monitoring Tool Data Collection :

Each monitoring client collects logs, metrics, and relevant operational data from its respective server and sends it to a central monitoring system. This centralized platform aggregates and analyzes the data, giving administrators visibility into the infrastructure's health and enabling proactive detection of anomalies or issues.

Monitoring Web Server requests:
If you need to track your web server's Queries Per Second (QPS), the monitoring system can be configured to measure HTTP request rates, response times, and error counts. By setting up alerts for unusual QPS patterns, the monitoring platform can quickly notify administrators of unexpected traffic spikes or irregularities, allowing timely intervention.

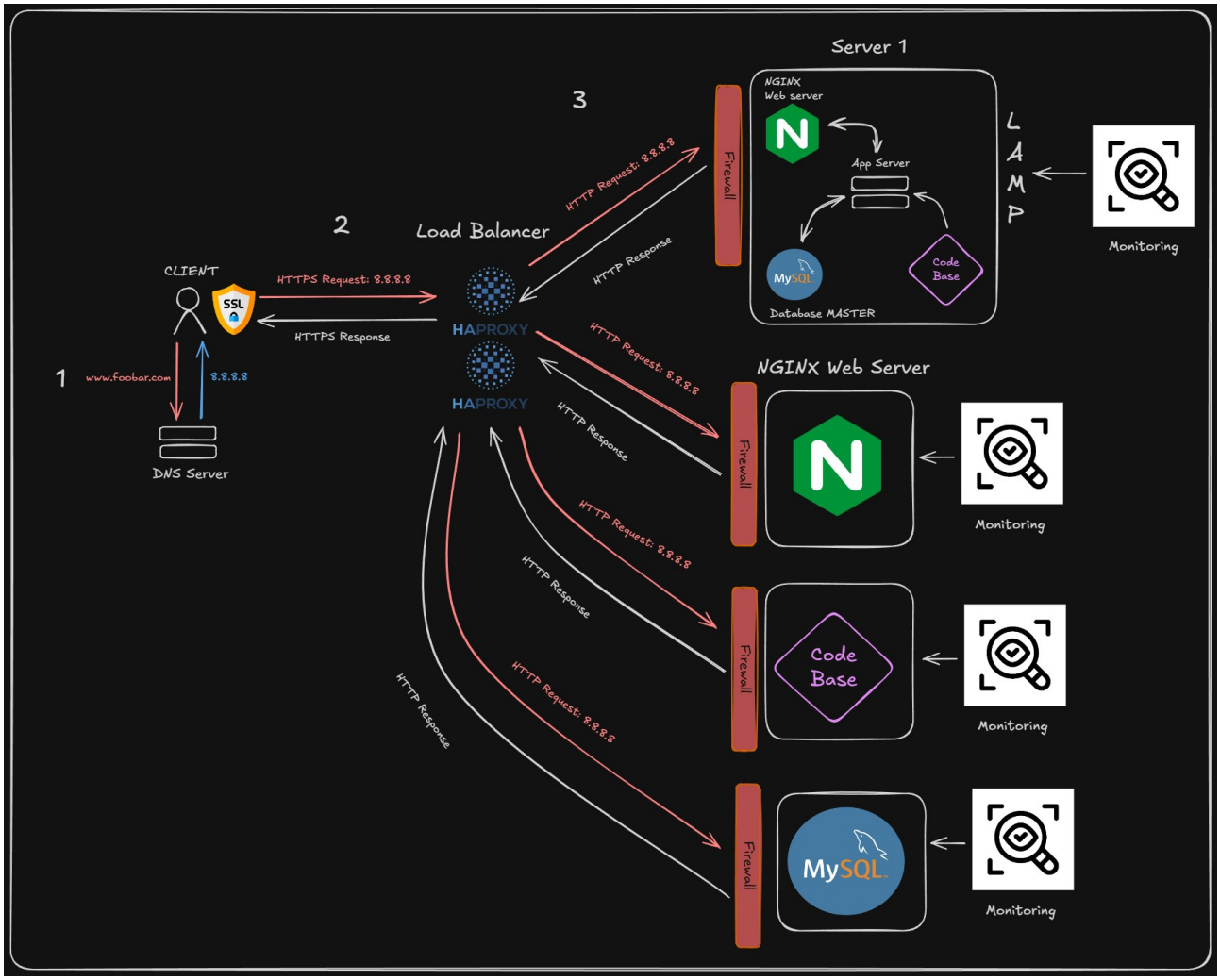
ISSUES :

Terminating SSL at the Load Balancer Level:
Handling SSL termination at the load balancer can introduce risks, as traffic between the load balancer and the backend servers may remain unencrypted. To maintain full data integrity and confidentiality, encryption should be applied across the entire communication path.

Identical Server Components:
Using servers that all run the same set of components—web, application, and database services—can lead to inefficient resource usage. Each layer of the stack has different performance requirements, so allocating resources based on the specific role of each server would help optimize overall performance. Then, as they are identical if one point of the code base is wrong all the servers will be impacted.

Single MySQL Server for Write Operations:
Relying on a single MySQL node for write operations creates a potential single point of failure. Implementing a primary-replica MySQL cluster is recommended to ensure high availability, maintain data redundancy, and prevent service disruption if the primary node fails.

Resilient and Secure Web Infrastructure



Explaination of the components we add :

Additional Server for Redundancy:
Adding an extra server increases overall redundancy and improves load distribution. If one server becomes unavailable or experiences heavy traffic, the additional server helps maintain continuous website availability.

Clustered Load Balancers (HAProxy):
To efficiently distribute incoming traffic and avoid overloading a single load balancer—which would otherwise become a single point of failure—two load balancers are configured to operate as a cluster. This setup ensures high availability and reliable load-balancing capabilities.

The principle:

- One load balancer is active (it handles the traffic).
- The other is passive or on standby (it waits).
- If the active one fails, the standby automatically takes over.
  This setup is known as an active/passive cluster.

Dedicated Servers for Each Component:
For better performance and resource management, each major component—such as the web server, application server, and database—is deployed on its own dedicated server. This separation allows each component to scale independently and prevents resource contention or bottlenecks.