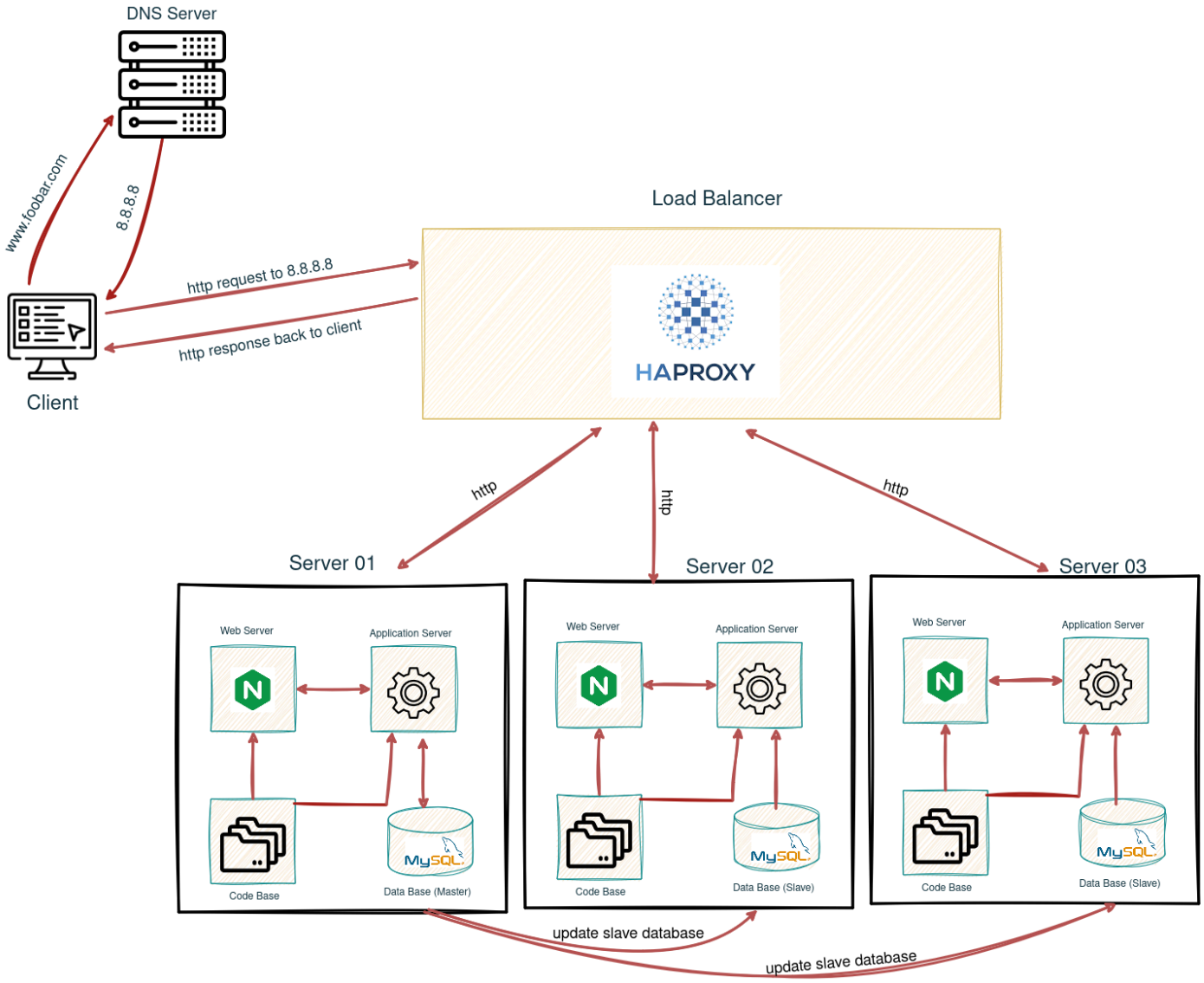

Fig 1: Three-Server Web Infrastructure for www.foobar.com

- For every additional element, why are you adding it?

Load Balancer (HAproxy): Manages incoming traffic and distributes it efficiently across multiple web servers. This provides scalability (handling increased traffic), high availability (minimizing downtime during server failures), and potentially improved performance (preventing overload on any single server).

Database replica:

- ➔ If the master database fails, the replica can take over to ensure continued access to the data,
- ➔ distributing read queries among replicas can reduce the load on the master database,
- ➔ the replica can be used to recover data in case of a disaster affecting the master database.

- What distribution algorithm your load balancer is configured with and how it works?

The load balancer (HAproxy) can be configured to use different algorithms such as Round Robin, Weighted Round Robin, Least Connections, Weighted Least Connections, Random…

Using **Round Robin** might be a good option, since it distributes requests equally among servers in a circular manner, ensuring each server receives a fair share of traffic.

- Is your load-balancer enabling an Active-Active or Active-Passive setup, explain the difference between both?

The load balancer (HAproxy) is allowing multiple servers to run at the same time, enabling an **Active-Active** setup.

In active-active load balancing, all web servers actively participate in handling incoming traffic for www.foobar.com. The load balancer distributes requests between them according to its configured algorithm (round-robin in this case).

In active-passive load balancing, only one web server (active) is responsible for handling incoming traffic. The other server (passive) acts as a hot standby, ready to take over if the active server fails.

- How does a database Primary-Replica (Master-Slave) cluster work?

In a Primary-Replica (Master-Slave) database cluster, the primary database (master) handles write operations and replicates these changes to one or more replica databases (slaves). The replication process typically involves the master sending a stream of data changes (such as insertions, updates, and deletions) to the replicas, which then apply these changes to their own copies of the data.

This setup provides several benefits, including improved read scalability (as read queries can be distributed among the replicas), fault tolerance (if the master fails, one of the replicas can be promoted to master), and offloading of read operations from the master, allowing it to focus on write operations.

- What is the difference between the Primary node and the Replica node in regard to the application server?

The application server directs all **write** queries (e.g., INSERT, UPDATE, DELETE) to the **primary node**, ensuring that data modifications are made to the authoritative database. On the other hand, replica nodes are often configured as **read** replicas, and the application server can direct read queries (e.g., SELECT) to these replica nodes.

Issues with the above infrastructure:

- **SPOFs:** If the load balancer fails, it can cause a disruption in the availability of the application, as incoming traffic would no longer be properly distributed among the servers.
- **Security**: No firewall exposes servers to attacks, and no HTTPS means data can be intercepted.
- **No Monitoring:** It's difficult to identify issues like server overload or application errors without monitoring.