Distributed web infrastructure

Added features:

1. 2 more servers.
2. Load Balancer with High availability proxy (HAproxy).

Why add 2 more servers?

- Having multiple servers offers numerous benefits in terms of performance, scalability, reliability, and fault tolerance, making it a common practice in building robust and scalable web applications.

Why having a Load Balancer?

- The primary function of a load balancer is to evenly distribute incoming network traffic across multiple servers. By spreading the workload across servers, a load balancer helps prevent any single server from becoming overwhelmed, ensuring optimal performance and responsiveness for users accessing the application or website.
- Load balancers play a critical role in optimizing performance, scalability, reliability, and availability in server infrastructures by efficiently distributing traffic, managing server resources, and ensuring seamless user experiences.

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

- I have chosen to configure the Load Balancer with the Roubin Round algorithm. In this algorithm, incoming requests are distributed sequentially to each server in a circular manner. Each server is assigned a turn to handle requests, and subsequent requests are passed to the next server in the list.

Round robin is simple to implement and ensures a roughly equal distribution of traffic among servers.

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

- Yes the Load Balancer is working on an active-active setup. The difference between the two setups is that an active-passive configuration for HAproxy involves a standby instance that takes over in the event of a failure, while an active-active configuration utilizes multiple instances simultaneously to handle traffic, providing better scalability and reliability.

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

- A database Primary-Replica (Master-Slave) cluster is a configuration where you have one primary (master) database server and one or more replica (slave) database servers. This setup is commonly used to achieve high availability, fault tolerance, and scalability in database systems.
- The primary database server, also known as the master server, serves as the primary source of truth for the database. All write operations (such as inserts, updates, and deletes) are performed on the primary database. It manages the data modifications and ensures consistency and integrity across the database.
- Replica database servers, also known as slave servers, replicate data from the primary database server. Read-only copies of the data from the primary database are continuously replicated to the replica databases. Replica databases can serve read queries (such as selects) from clients, providing read scalability and offloading read operations from the primary database.
- In the event of a failure or outage of the primary database server, one of the replica databases can be promoted to become the new primary database. This process, known as failover, ensures continuous availability of the database system and prevents data loss.

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

- The primary node is responsible for handling write operations and serving as the authoritative source of data, while replica nodes replicate data from the primary node and serve as read replicas for read operations. Applications interact with the primary node for write operations and can distribute read operations across replica nodes to improve performance and scalability.

Issues:

While a distributed web infrastructure such as this one is considered an upgrade over a simple web stack like the LAMP stack, we still as developers and organizations may encounter issues regarding this infrastructure and its applications. Following are some of these issues:

1. **Where are the (SPOF)**:

   Potential single points of failure (SPOF) could include having the Load Balancer failing to distribute the traffic to the servers, leading to service downtime or degraded performance.

   If the primary database server fails, the entire database cluster could become unavailable until a failover to a replica node occurs.

   Lack of redundancy in critical components such as having only one application server that fails leading to rendering the entire application unavailable until the server is restored.

2. **Security Issues**:

   Lack of firewall as without a firewall, the infrastructure is more easily affected by unauthorized access, attacks, and malicious activities.

No HTTPS, Transmitting data over HTTP instead of HTTPS poses security risks, as sensitive information can be intercepted, tampered with, or stolen by attackers.

3. **No Monitoring**:

   Without monitoring, it's challenging to detect and respond to issues promptly, leading to potential service disruptions, performance degradation, or security incidents going unnoticed.

   Monitoring helps track the health, performance, and availability of infrastructure components, identify bottlenecks or failures, and proactively address issues before they impact users.