Readme: Web Server vs. Application Server

We have implemented a scalable and resilient web application in this infrastructure configuration by utilising distinct servers for the web server, application server, and database. In order to optimally disperse traffic, we have also created a high-availability architecture using two load balancers (HAproxy) configured as a cluster.

Elements:

Web server:1.

The web server, such as Apache or Nginx, is in charge of responding to HTTP requests from users' web browsers. It sends dynamic queries to the application server and serves static information immediately. To enhance overall performance and optimise web page delivery, we employ a dedicated web server. Reducing the attack surface through web server isolation further improves security.

. Application Server2: This server runs programs and server-side scripts that are dynamic in nature. It runs the required code, creates dynamic web pages, and handles requests that have been forwarded by the web server. Scalability, ease of maintenance, and effective resource use are all made possible by the use of a distinct application server. Additionally, it offers separation of the web server setup from the application logic.

3. Server for Database
The data on the website is managed and stored by the database server (such as MySQL or PostgreSQL). It is in charge of managing database functions, guaranteeing data consistency, and offering trustworthy data storage. Relocating the database to a separate server improves scalability, security, and eases backup and recovery processes.

4. Load Balancer (HAproxy) Cluster: To provide high availability and load distribution, we have set up two HAproxy load balancers as a cluster. By distributing incoming traffic among several application servers, load balancers reduce the risk of server overload and speed up response times. Fault tolerance is provided by the cluster configuration; in the event that one load balancer fails, the other can take over with ease, guaranteeing the application's continuous availability.

Reasons for Adding Each Element:
Divided Servers: Putting components on separate servers guarantees better security, scalability, and effective use of resources. It is possible to optimise and secure each component separately, which simplifies maintenance and troubleshooting.
Implementing a load balancer cluster guarantees uniform traffic distribution and excellent availability. It offers fault tolerance, improves performance, and avoids bottlenecks. Service continuity is ensured by the cluster configuration even in the case of a load balancer failure.

In conclusion, this infrastructure delivers optimal performance, scalability, and high availability through the deployment of a load balancer cluster and the segregation of components onto dedicated servers. This configuration is made to manage fluctuating traffic volumes, guaranteeing a dependable and smooth user experience.