Explanation:

1. **Load Balancer (HAProxy):** The load balancer distributes incoming traffic from users across the three backend servers to achieve load balancing and ensure high availability. It helps distribute the workload and prevents a single server from being overwhelmed.
2. **Web Servers (Nginx):** Nginx serves as the web server and handles incoming HTTP/HTTPS requests. It can also handle SSL termination, freeing the application server from handling SSL encryption/decryption. It efficiently manages static content and forwards dynamic requests to the application server
3. **Application Server:** The application server handles the dynamic processing of user requests, such as executing code, accessing databases, and generating responses. It works in conjunction with the web server to provide the necessary content to users.
4. **MySQL Database:** The database stores and manages the website's data. It can be a primary-replica (master-slave) setup for data redundancy and fault tolerance.
5. **Load Balancer Distribution Algorithm:** The load balancer is configured with a round-robin distribution algorithm, meaning it evenly distributes incoming requests to each back-end server in a sequential manner.

   Our load-balancer is using a Round Robin algorithm distribution. Meaning the queries requested are distributed to every server sequentially one after another. And after sending the request to the last server, the algorithm startarts from the first server. This will bring on average and approximately, to a server load distribution of 50% on each of the two servers configuration.

   The Round Robin algorithm sends the requests to each server in turn in a loop every time it finishes with all of them the servers in the list and according to the idle capacity of each server
6. **Active-Active Setup:** The load-balancer setup is active-active, meaning all servers are actively serving user requests at all times. This provides better distribution of incoming traffic and load sharing.

- The **Active-Active cluster** is typically made up of at least two nodes, both activaley running the same type of services at the same time. Their purpose is to achieve load balancing by distributing tasks to different servers in order to prevent overload. As there are more than one servers (nodes) available to severe, the service time and process throughput can have improvements.
- On the other hand the **Active-Passive setup**, also made up of at least two nodes (servers), however not all nodes are going to be active

simultaneously. In this configuration, while one node is active, the other nodes (fail-over servers) are passively waiting to be active as backup in case the primary server (the one being in use actively) is disconnected or unable to serve. Under this configuration, and as in the Active-Active set up, it is important that primary and fail-over nodes have the exact server configuration, so clients won't be able to tell the difference when the fail-over server takes over the operation

7. **Primary-Replica (Master-Slave) Database Cluster:** The MySQL database operates as a primary-replica cluster. The primary node handles write operations and replicates data to replica nodes. The replica nodes handle read operations, offloading the primary node and providing redundancy.

   A database Primary-Replica (Master-Replica) is a mechanism which enables data of one database server (the master) to be replicated or to be copied to one or more computers or database servers (the slaves), in order all users share the same level of information. This process leads to a distributed database in which users can quickly access data without interfering with each other.

8. **Primary Node vs. Replica Node:** The primary node is responsible for handling write operations and maintaining the most up-to-date data. The replica nodes handle read operations and provide data redundancy. The application typically connects to the primary node for write operations and can connect to replica nodes for read operations, improving read scalability.

   Note:

   One of the main differences between the primary node and the replica node, regarding the application, is that the primary database is regarded as the authoritative source, while the replica database is synchronized to it. The primary node serves as the keeper of information, here the "real" data is kept, then writing only happens here. On the other hand, reading only occurs in the replica or slave node. This architecture purpose is due to safeguard site reliability. In case a site receives a lot of traffic, a replica node prevents overloading of the master node with reading and writing requests. This eases the load of the entire system preventing it to collapse

Issues with the Infrastructure:

1. **Single Point of Failure (SPOF):** The load balancer itself is a potential SPOF. If it fails, incoming traffic won't be distributed properly and the this will cause the entire system to stop working.

2. **Security Issues:** Lack of firewall configuration poses security risks. Also, using only the HTTP protocol leaving user data vulnerable to interception due to the absence of encryption of data.

   The data transmitted over the network isn't encrypted using an SSL certificate so hackers can spy on the network. There is no way of blocking unauthorized IPs since there's no firewall installed on any server

3. **No Monitoring:** Without monitoring clients, we won't have visibility into the health, performance, and issues within our infrastructure.

To address these issues, we should consider implementing redundancy for the load balancer, setting up firewalls, enabling HTTPS with SSL certificates, and implementing monitoring solutions to ensure the health and security of your infrastructure.