Distributed Web Infrastructure Design for www.foobar.com

Overview

This document outlines the design of a three-server web infrastructure for hosting the website www.foobar.com. The design includes adding servers and key components to enhance load distribution, reliability, and performance.

Components

1. **Load Balancer (HAProxy)**
2. **Web Server (Nginx)**
3. **Application Server**
4. **Database Server (MySQL)**

Diagram

```
User
  |
  v
DNS (www.foobar.com -> Load Balancer IP)
  |
  v
Firewall
  |
  v
Load Balancer (HAProxy)
  |
  v
  -------------------
  |                 |
  v                 v
Web Server (Nginx)  Application Server
  |                 |
  v                 v
Database Server (MySQL)
```

PROF

Components and Their Roles

Load Balancer (HAProxy)

- **Role**: Distributes incoming traffic to the web server and application server to balance the load and increase availability.

- **Why Add It**: Ensures that traffic is evenly distributed, preventing any one server from becoming a bottleneck.

Web Server (Nginx)

- **Role**: Handles HTTP requests, serves static content, and forwards dynamic content requests to the application server.
- **Why Add It**: Specialized in serving static files quickly and efficiently and acts as a reverse proxy for dynamic content.

Application Server

- **Role**: Runs the application logic and processes dynamic content.
- **Why Add It**: Separates dynamic content processing from static content serving, optimizing resource usage and performance.

Database Server (MySQL)

- **Role**: Manages data storage and retrieval.
- **Why Add It**: Provides a central location for data management, ensuring data integrity and availability.

Load Balancer Configuration

Distribution Algorithm

- **Round-Robin**: Distributes requests sequentially across all servers. This method is simple and ensures a fairly even distribution of traffic.
    - **How It Works**: Each incoming request is passed to the next server in the list. If there are three servers, the first request goes to Server 1, the second to Server 2, the third to Server 3, and the fourth request starts again with Server 1.

Active-Active vs. Active-Passive Setup

PROF

- **Active-Active**:
    - Both servers are actively handling traffic.
    - Provides better resource utilization and redundancy.
    - Both servers share the load and can take over if one fails.
- **Active-Passive**:
    - One server actively handles traffic while the other is on standby.
    - Standby server takes over if the active server fails.
    - Simpler setup but can lead to underutilization of the standby server.

Database Primary-Replica (Master-Slave) Cluster

How It Works

- **Primary (Master) Node**:
    - Handles all write operations and updates.

    - For the application: The application writes data to the primary node.
- **Replica (Slave) Node**:
    - Copies data from the primary node and handles read operations.
    - For the application: The application reads data from the replica node, reducing the load on the primary node.

Differences Between Primary and Replica Nodes

- **Primary Node**:
    - Handles write operations.
    - Directs data updates and modifications.
- **Replica Node**:
    - Handles read operations.
    - Mirrors the data from the primary node to ensure consistency.

Issues with This Infrastructure

Single Points of Failure (SPOF)

- **Load Balancer**: If the load balancer fails, the entire system goes down.
- **Database**: If the primary database server fails, write operations are halted until a failover occurs.

Security Issues

- **No Firewall**: Lack of firewalls exposes the system to unauthorized access and attacks.
- **No HTTPS**: Without HTTPS, data transmission between the user and the web server is not encrypted, leading to potential data breaches.

No Monitoring

- **Issue**: Without monitoring, it's difficult to detect performance issues, track server health, and receive alerts for problems.

PROF

Recommendations

Addressing SPOF

- **Load Balancer**: Implement a second load balancer in a cluster configuration for failover capabilities.
- **Database**: Use database clustering with automatic failover to ensure high availability.

Enhancing Security

- **Firewalls**: Deploy firewalls to protect each server from unauthorized access.
- **HTTPS**: Implement SSL/TLS certificates to encrypt data transmission.

Implementing Monitoring

- **Monitoring Tools**: Use monitoring tools (e.g., Sumologic, Prometheus) to track performance metrics, server health, and set up alerts for any issues.

Conclusion

This distributed web infrastructure design provides a robust and scalable setup for hosting www.foobar.com, with clearly defined roles for each component and considerations for load balancing, security, and high availability. Implementing additional measures for security and monitoring will further enhance the reliability and performance of the system.