Secured and Monitored Web Infrastructure

3 new elements have been added to our web infrastructure.

- **Firewalls:** This will protect the servers from been attacked and exploited by hackers

- **SSL Certificate:** SSL stands for Secure Sockets Layer, and its purpose is to provide a secure and encrypted connection between a user's web browser and a web server. The SSL certificate will serve www.foobar.com over the HTTPS protocol.

- **Monitoring Clients:** will serve to collect logs and send them to our data collector sumologic.

What are Firewalls used for?
They are network security devices or software applications designed to monitor, filter, and control incoming and outgoing network traffic based on predetermined security rules. Firewalls play a crucial role in protecting computer systems and networks from unauthorized access, cyberattacks, and other security threats.

Why is the traffic served over HTTPS?
Traffic served over HTTPS (Hypertext Transfer Protocol Secure) is encrypted and secured, providing several important benefits for both website owners and users.

What monitoring is used for
They are used to collect and transmit information about the performance, status, and health of a system or network to a monitoring server.

How the monitoring tool is collecting data
Several methods listed below can be adopted by a monitoring tool to collect data.

- Agent-Based Monitoring:
- Agentless monitoring
- SNMP (Simple Network Management Protocol):
- API Integration

- Logs and Log File Monitoring
- Packet Sniffing
- Scripting and Custom Plugins
- JMX (Java Management Extensions)
- Performance Counters (Windows)
- Syslog and Events
- Database Queries

Each of the methods listed above can be applied to collect relevant data from our server (web server, application server, and database).

Explain what to do if you want to monitor your web server QPS
Monitoring QPS provides insights into the server's performance, traffic patterns, and helps identify potential issues. The following steps can be taken to monitor the Query Per Second (QPS) of our web server.

- **Choose a Monitoring Tool:** New Relic, Datadog, Uptime Robot, Nagios, Wavefront
- **Instrument Your Web Server:** Depending on the monitoring tool chosen, you may need to instrument your web server to collect QPS metrics
- **Configure QPS Monitoring:** This may involve setting up specific queries, counters, or metrics that measure the number of queries or requests processed per second.
- **Set Up Alerts:** Establish alerting thresholds for QPS to be notified when the QPS exceeds normal operating levels.
- **Create Dashboards:** Use the monitoring tool to create dashboards that visualize QPS data over time.
- **Analyze and Optimize:** Regularly review the QPS data and analyze patterns. Identify peak usage times, assess the impact of changes, and optimize your web server configuration accordingly.
- **Consider Load Testing:** Perform load testing on your web server to simulate high traffic conditions and observe how it handles increased QPS.
- **Documentation and Documentation:** Document the monitoring setup, including the metrics being collected, alerting thresholds, and any custom configurations.

Issues

Why terminating SSL at the load balancer level is an issue
When SSL termination occurs at the load balancer, the encrypted connection between the client and the load balancer is decrypted, and the subsequent communication between the load balancer and backend servers is typically unencrypted.
This means that the traffic between the load balancer and the backend servers is no longer encrypted. If the communication between the load balancer and backend servers traverses an untrusted network (e.g., the internet), it could potentially be intercepted.

Why having only one MySQL server capable of accepting writes is an issue
Having only one MySQL server capable of accepting writes can be problematic for several reasons, mainly related to reliability, scalability, and high availability. When there is only one MySQL server accepting writes, it becomes a single point of failure. If this server experiences downtime or failure, write operations are disrupted, and the entire system becomes unavailable. This lack of redundancy can result in significant downtime for applications relying on the database.

Why having servers with all the same components (database, web server and application server) might be a problem
Having servers with identical components across all layers is known as a monolithic architecture. While this approach has its advantages, it also comes with challenges

- **Scalability Challenges:** In a monolithic architecture, scaling typically involves increasing the resources (CPU, RAM) of individual servers. This is known as vertical scaling.
- **Limited Flexibility and Technology Stack:** Using the same technology stack for all components may limit flexibility in choosing the best tools or frameworks for specific tasks. Different components might have different requirements that are better served by specialized technologies.
- **Deployment Challenges:** Deploying changes to a monolithic system can be complex and may require taking the entire system offline during updates. This can lead to downtime, and it becomes challenging to roll out updates or new features incrementally.