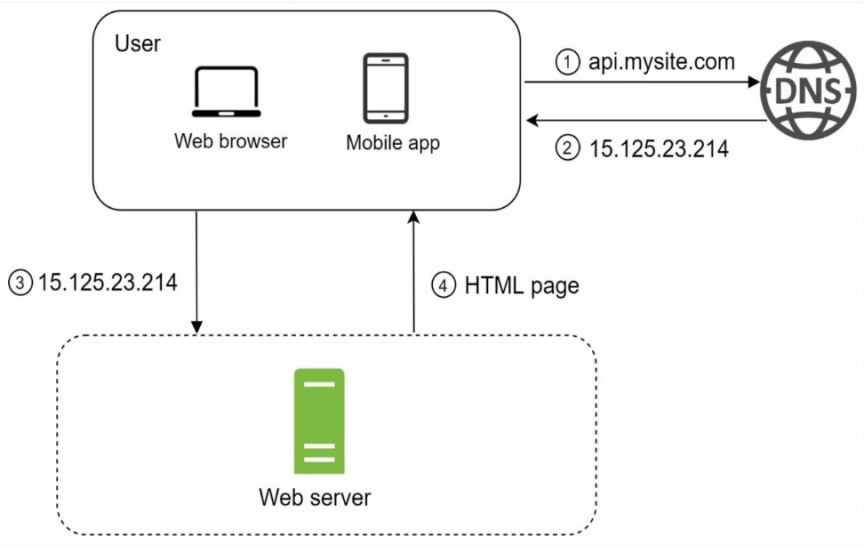
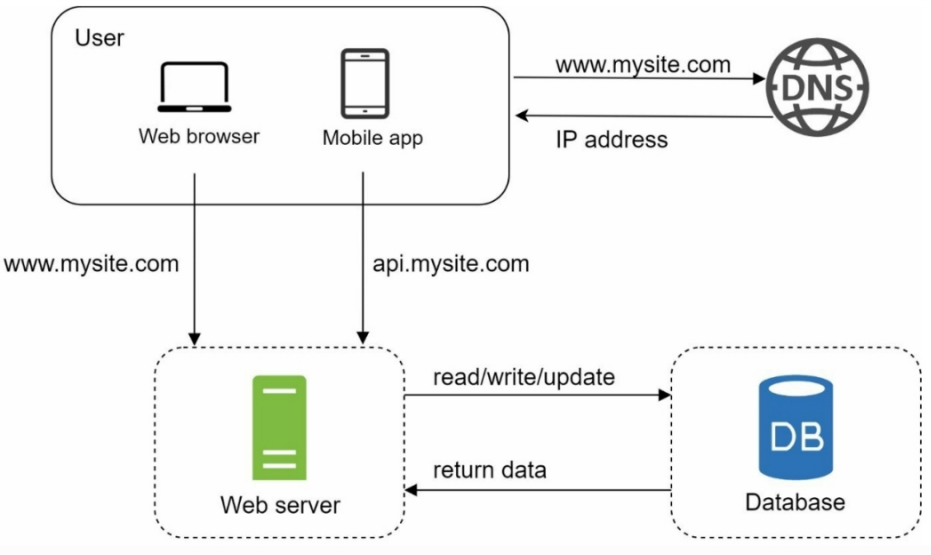
Scale from zero to million of users

- Single Server Setup -



- Users access websites through domain names.
- IP address is returned to the browser or mobile app.
- Once IP address is obtained, HTTP request are sent directly to the web server.
- The web server returns HTML / Json for rendering

- Database -



- Seperating web traffic and database servers allows them to be scaled independently.

Which database to use?

- Relational Database represent and store data in tables and rows. Join operations using SQL across different database tables.
- Non Relational Database (NoSQL) are grouped in 4 categories -
  - Key-value stores
  - Graph stores
  - Column stores
  - Document stores

  Join operations are generally not supported.

Non Relational Databases might be right choice if -

- Application requires super low latency.
- Data is unstructured or their is no relational data.
- Data needs to be serialized or deserialized.
- Massive amount of data needs to be stored.

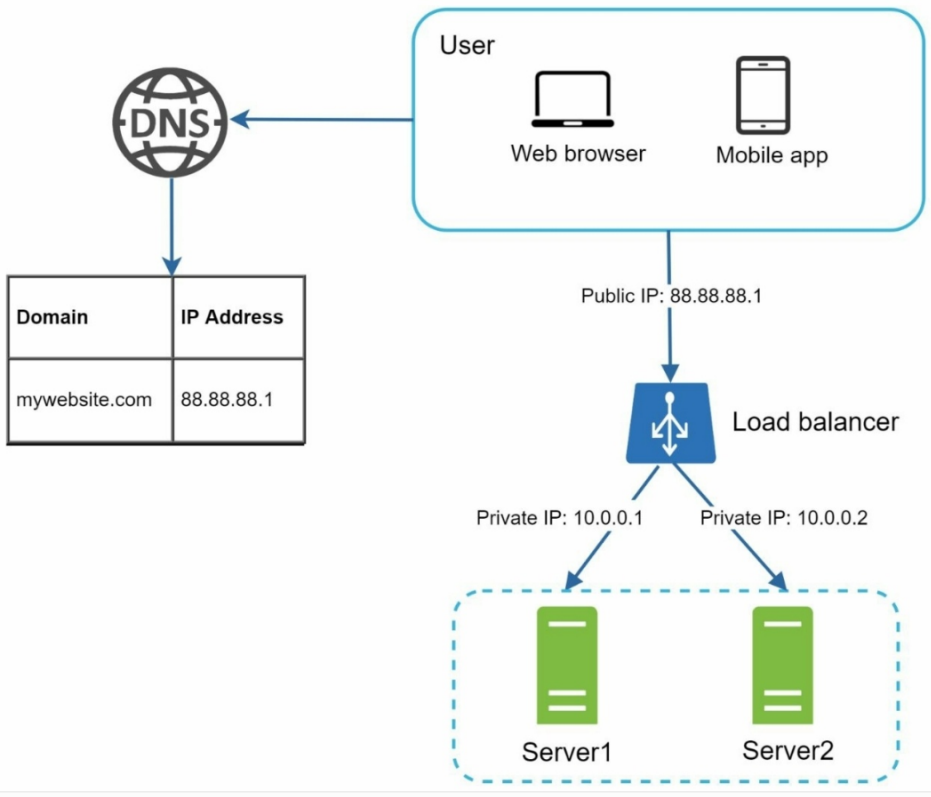
- Vertical scaling v/s Horizontal scaling

  - Vertical Scaling referred to as 'scale up', means the process of adding more power (CPU, RAM etc)
  - Horizontal Scaling also known as 'scale out', allows you to scale by adding more servers into the pool of resources.

- Vertical scaling is a great option when traffic is low and simplicity of vertical scaling.
- Limitations of vertical scaling -
  - Vertical scaling has a hard limit. It is impossible to add unlimited CPU and memory to a single server.
  - Vertical scaling doesn't have failover and redundancy.
- Horizontal scaling is more desirable for large scale applications due to limitations of vertical scaling.

- Load Balancer

- Load Balancer evenly distributes incoming traffic web servers.

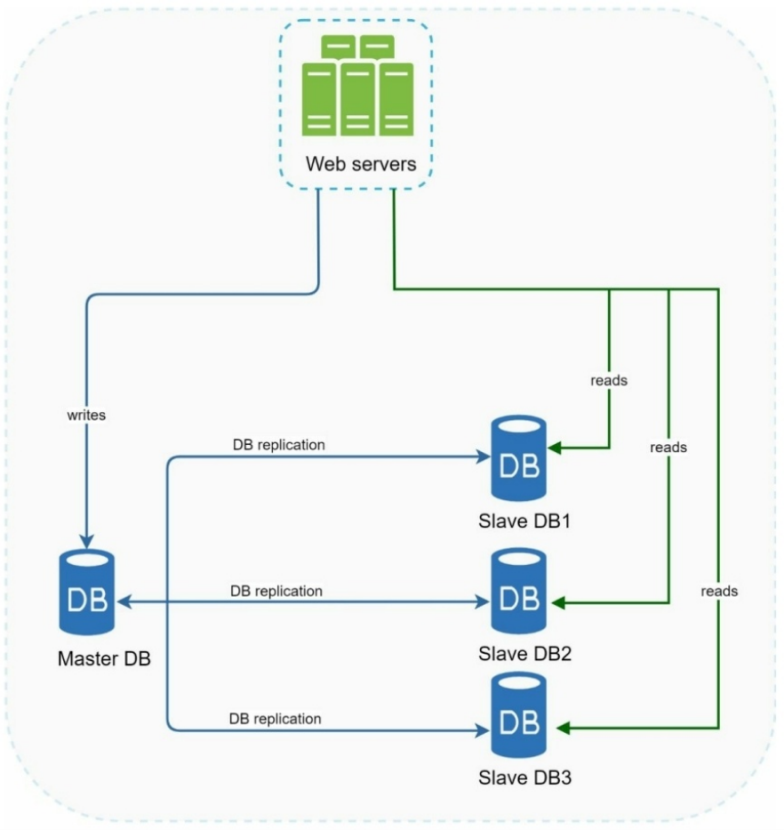


- With this setup, web servers are unreachable directly by clients anymore.

- For better security, private IPs are used for communication between servers
- The load balancer communicates with web servers through private IPs.

- If server 1 goes offline, all the traffic is routed to server 2 preventing the website from going offline.
- If website traffic grows rapidly, a third server can be added to the sowes pool and load balancer will automatically start sending requests to it.

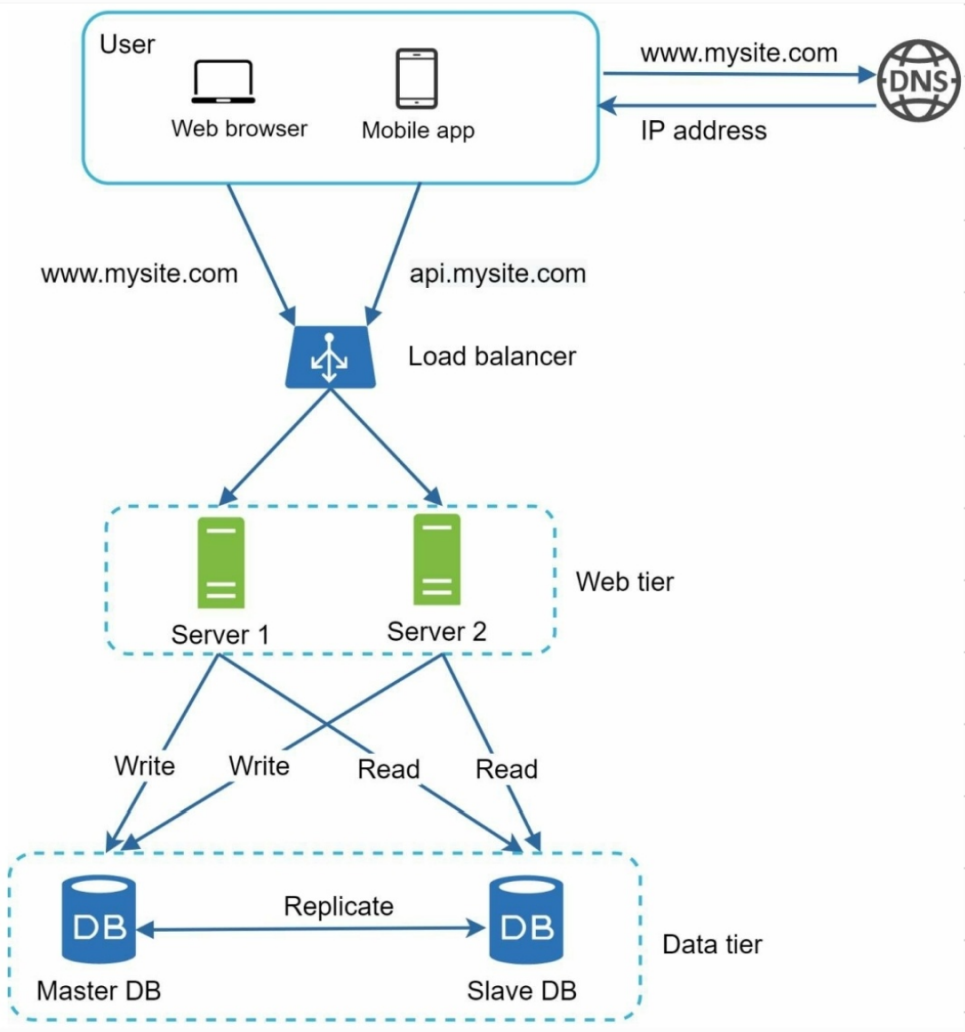
- Database Replication
  - Usually master / slave relationship b/w the original (master) and the copies (slave).
  - Master database only write operations. A slave database gets copies of data from master database and only supports read operation.
    - All commands like insert, delete or update must be sent to master database.
    - Most application requires a much higher ratio of reads to writes, number of slave database is usually larger than master database.



- Advantages of Database Replication -
  - Better Performance - All writes happen in master nodes whereas read operations are distributed across slave nodes, improving performance as more queries are processed in parallel.
  - Reliability - If one database server is destroyed, data is still preserved as it is replicated across multiple locations.
  - High Availability - By replicating data across different locations, data can be accessed from another server.

- If all slave database goes offline, read operations are redirected to master database until issue is fixed.
- If master database goes offline, a slave database is promoted to master database.

- Overall Design -

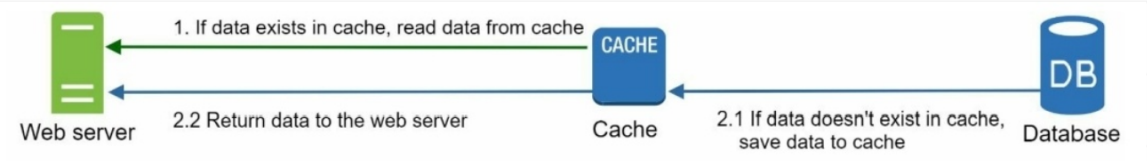


- User gets IP of Load Balancer from DNS.
- User connects to Load Balancer using IP.
- HTTP request is routed to either Server 1 or 2.
- Web server reads data from slave database.
- Web server routes data-modifying operations to master database.

- Cache
  - Cache is temporary storage area that stores the result of expensive responses or frequently accessed data in memory so that subsequent requests are served more quickly.

- Cache Tier -

- The cache tier is a temporary data store layer, much faster than database.
- Benefits of having a seperate cache tier include better system performance, ability to reduce database workloads and ability to scale the cache tier independently.



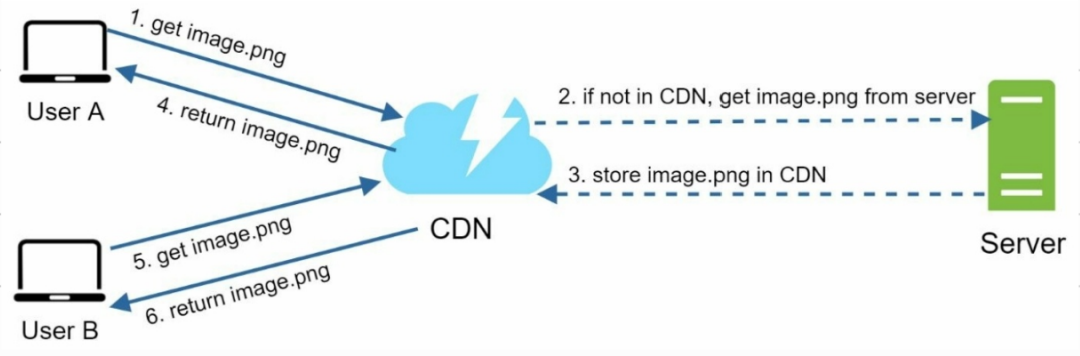
- After receiving a request, a web server first checks if the cache has the available response. If it has, it sends data back to the client. If not, it queries the database, stores the response in cache, and send it back to the client.
- Interacting with cache servers is simple because most cache servers provide APIs for common programming languages.

- Consideration for using cache

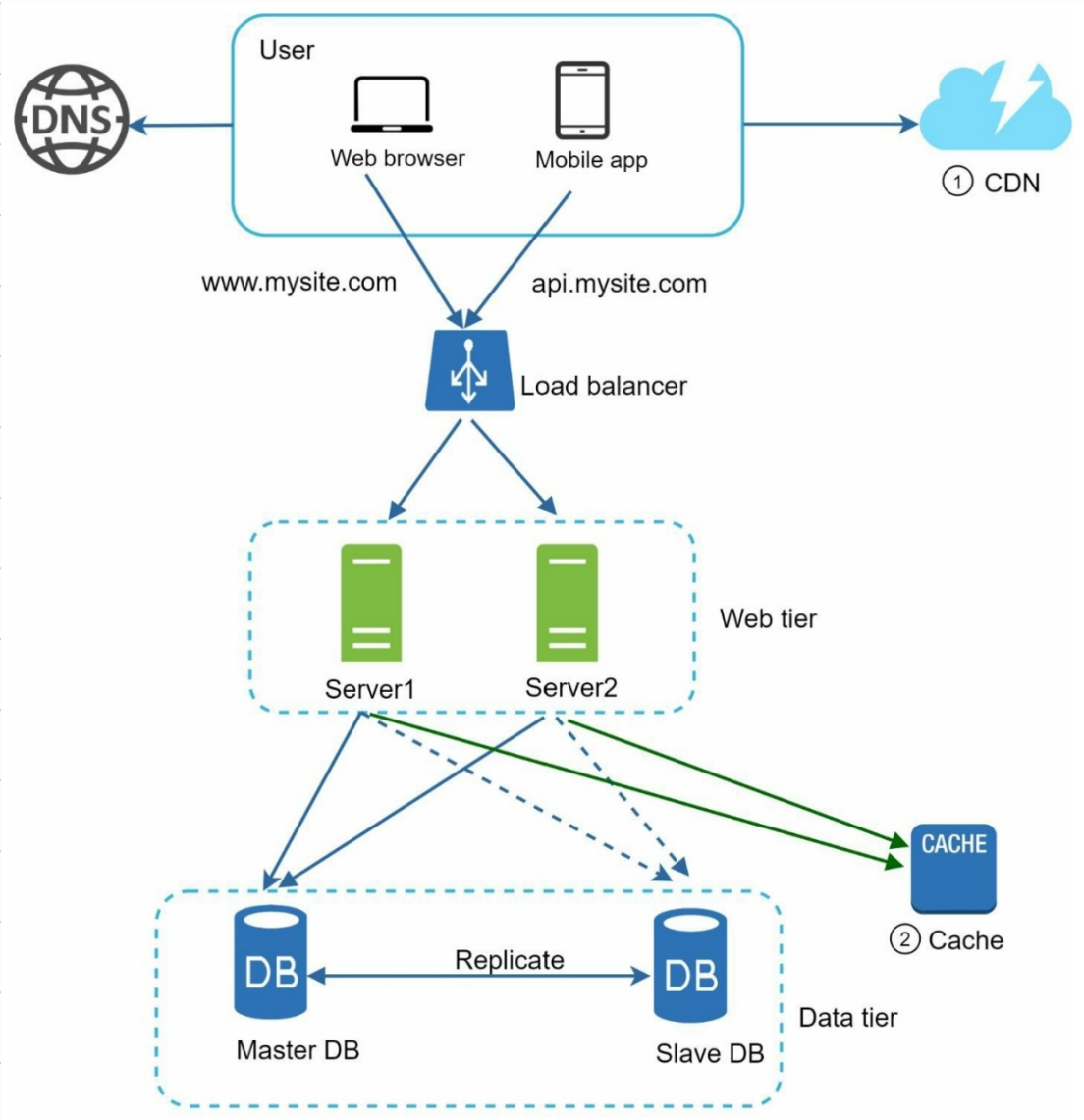
- Cache is used when data is read frequently but modified infrequently.
- Expiration Policy helps prevent cache from becoming too large or old.
- Consistency involves keeping stored data and cache in sync. Inconsistency can happen if data modifying operation on data store but not cache.
- Mitigating failures - Multiple cache servers are recommended to avoid SPOF (Single point of failure)

- Eviction Policy - Policy for removing items when cache is full and an item needs to be added. Eg. - LRU (Most Popular), LFU, FIFO

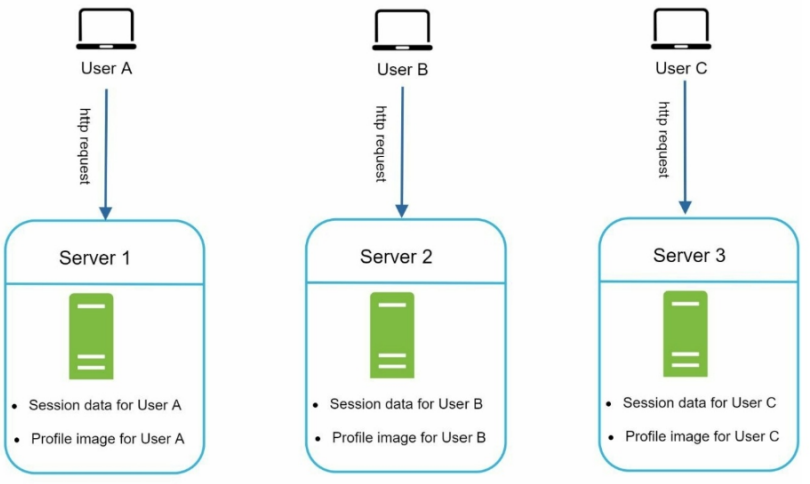
- Content Delivery Network (CDN)
  - CDN is a network of geographically dispersed servers used to deliver static content.
  - CDN servers cache static content like images, videos, CSS, Javascript etc.
  - When user visits a website, a CDN server closest to the user will deliver static content.



  - Considerations of using CDN
    - Cost - CDN charges for data transfer in and out of the CDN. Caching infrequently used assets provides no significant benefits and can be moved out of CDN.
    - Setting cache expiry - For time sensitive content, setting a cache expiry time is important.
    - CDN fallback - Clients should be able to detect CDN outage and request resources from origin.
    - Invalidating files - Files from CDN before it expires by either using CDN API or using object versioning.

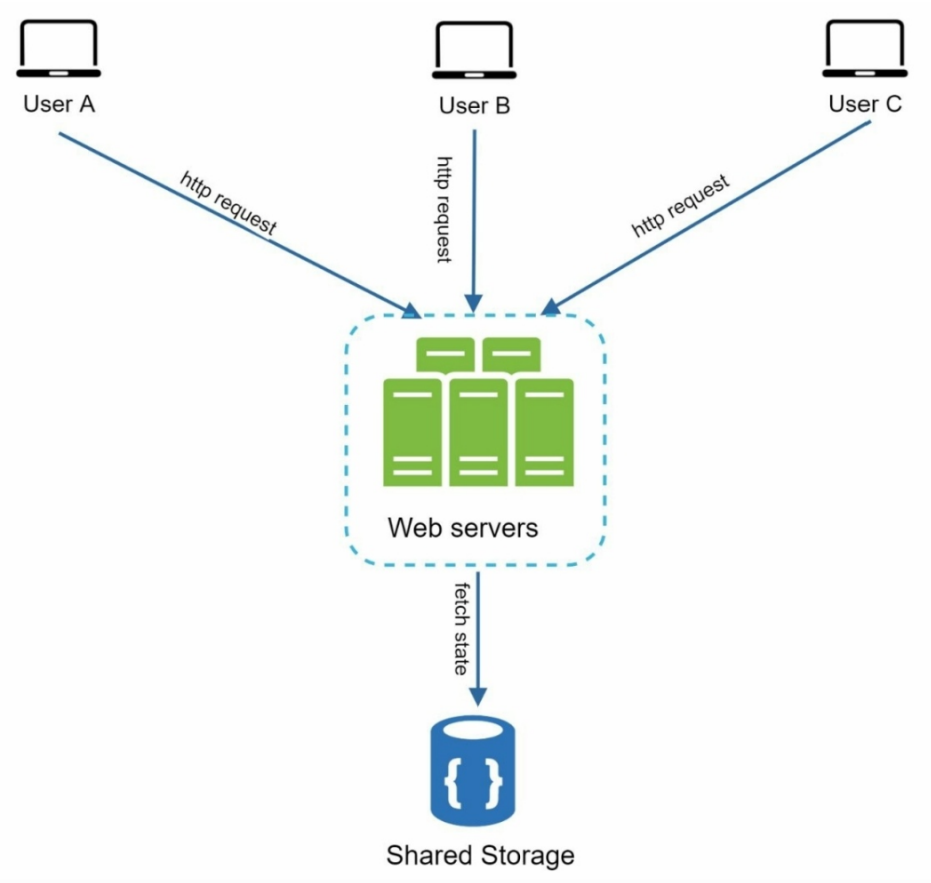


- Stateless Web Tier
  - Stateful Architecture
  - A stateful server remembers client data from one request to next.
  - A stateless server keeps no state information.



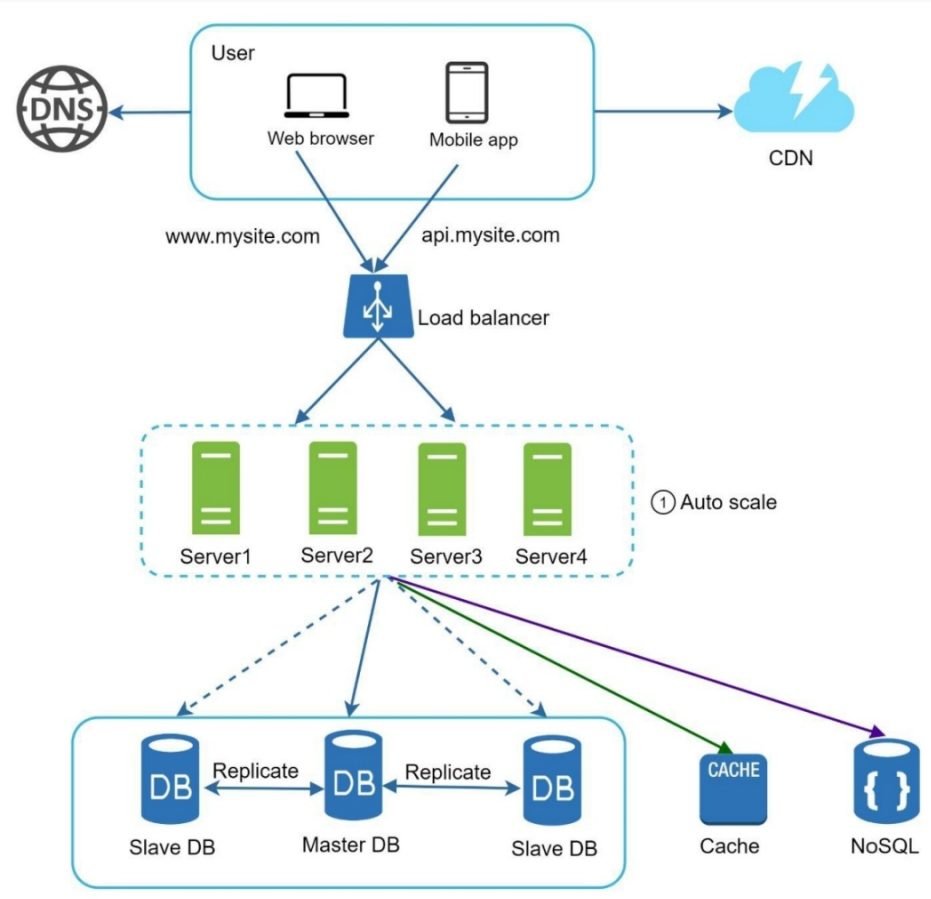
- The issue is that every request from the same client must be routed to the same servers.
- This can be done with sticky sessions in load balancers, however this adds the overhead..
- Adding or removing servers is much more difficult with this approach. It is also challenging to handle server failures.

- Stateless Architecture

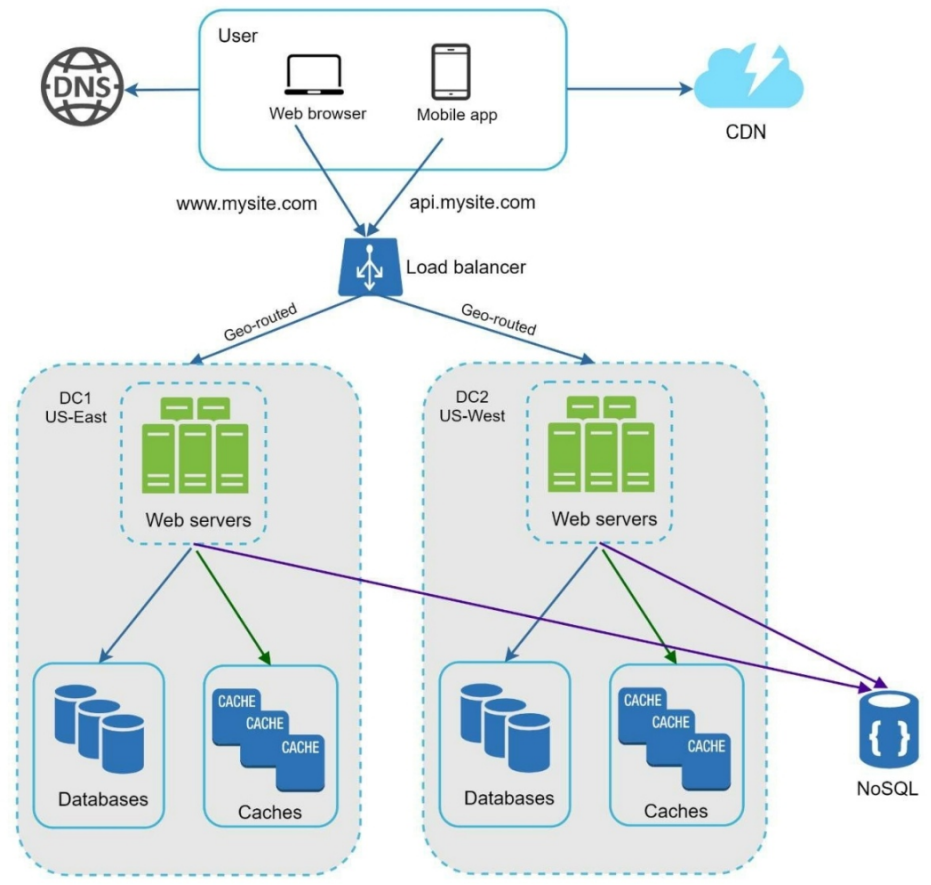


- HTTP requests from users can be sent to any web servers, which fetch data from a stored data storage.
- State data is stored in a shared data store and kept out of web servers.
- A stateless system is simpler, more robust and scalable.

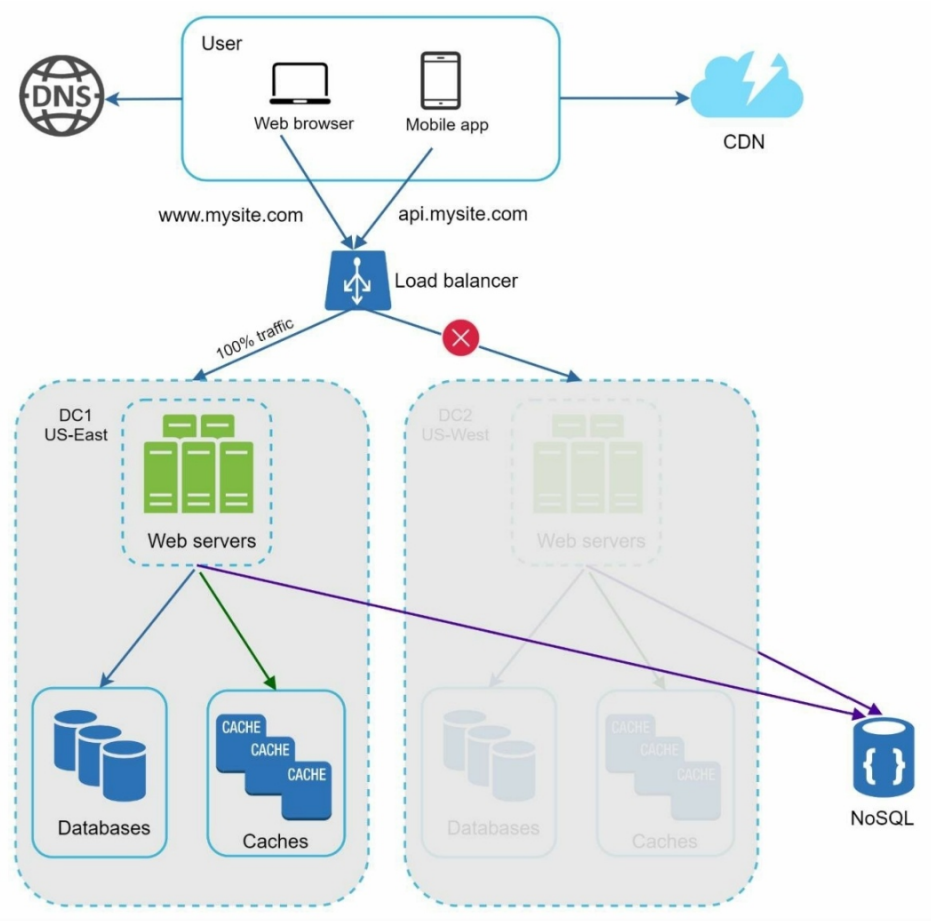
- Updated Design with stateless web tier



- Data Centers



- In normal operation, users are geoDNS-routed, also known as geo-routed to the closest data center.
- GeoDNS is a DNS service that allows domain names to be resolved to IP addresses based on the location of a user.



- In the event of any significant data center outage, all traffic is routed to healthy data center.

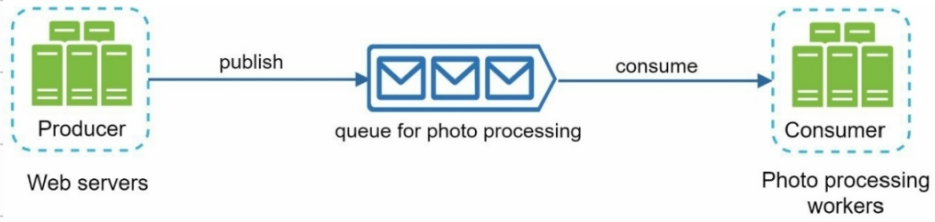
- Technical Challenges in multi-data center setups -
  - Traffic Redirection - Effective tools are needed to direct traffic to the correct data center. GeoDNS can be used to direct traffic to nearest data center depending on location.

- Data Synchronization - Users from different regions could use different local database or cache. For failover cases, a common strategy to replicate data across multiple data centers.
- Test and Deployment - It is important to test your website at different locations. Automated deployment tools are vital to keep services consistent through all data centers.

- Message Queue

- Message queue is a durable component, stored in memory that supports asynchronous communication.
- It serves as a buffer and distributes asynchronous requests.
- Basic Architecture involves Input services, called producer / publisher, create messages and publish them to a message queue.
- Other services or servers, called consumers / subscribers, connect to the queue, and performs actions defined by the messages.

- With message queue, producers can post a message to the queue when consumer is unavailable. The consumers can read messages even when the producer is unavailable.



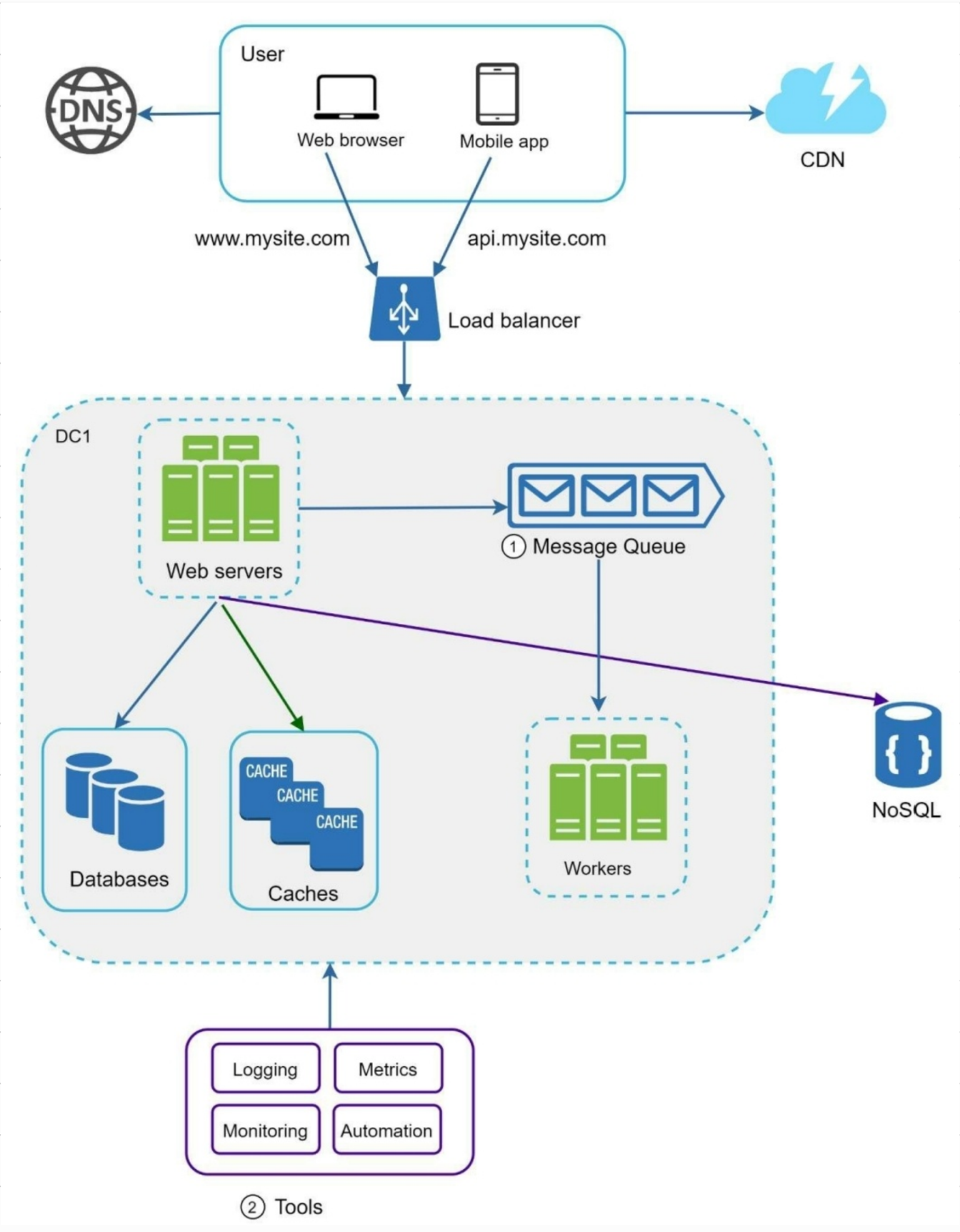
- Logging, Metrics and Automation

  - Logging - Monitoring error log is important to identify errors and problem in the system. A tool can be used to aggregate them to a centralized service for easy search and viewing.

  - Metrics - Collecting metrics help us gain business insights and understand health status of a system. Following Metrics are useful -
    - Host Level Metrics: CPU, Memory, Disk I/O etc.
    - Aggregated Level Metrics: Performance of entire database tier, cache tier etc.
    - Business Metrics: Daily active users, retention, revenue etc.

  - Automation - When system gets big and complex, automation tools can improve productivity.

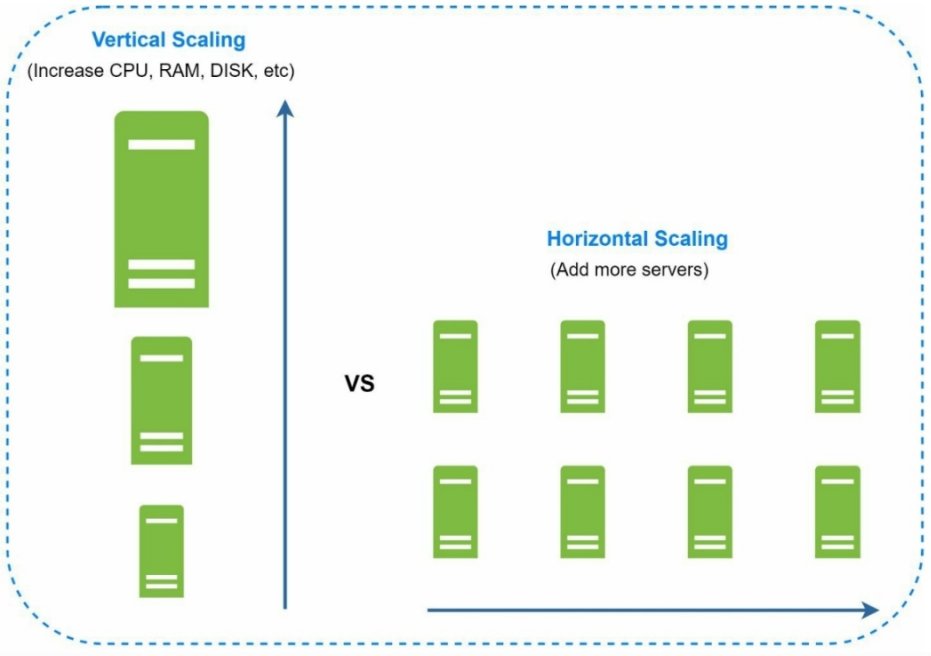
- Updated Design



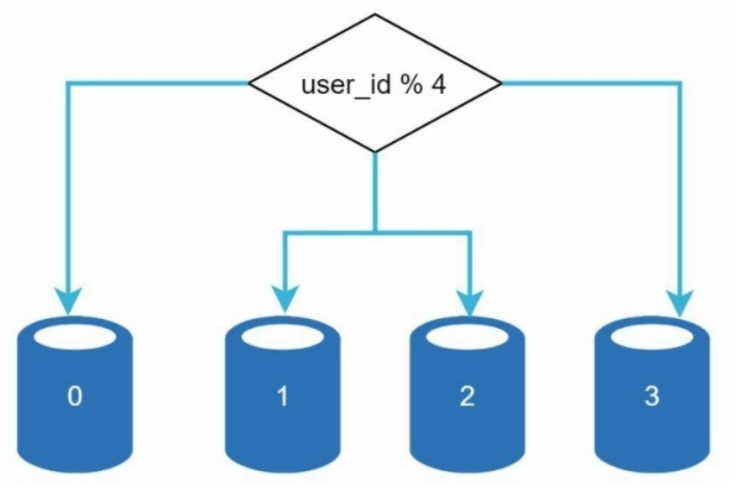
- Database Scaling

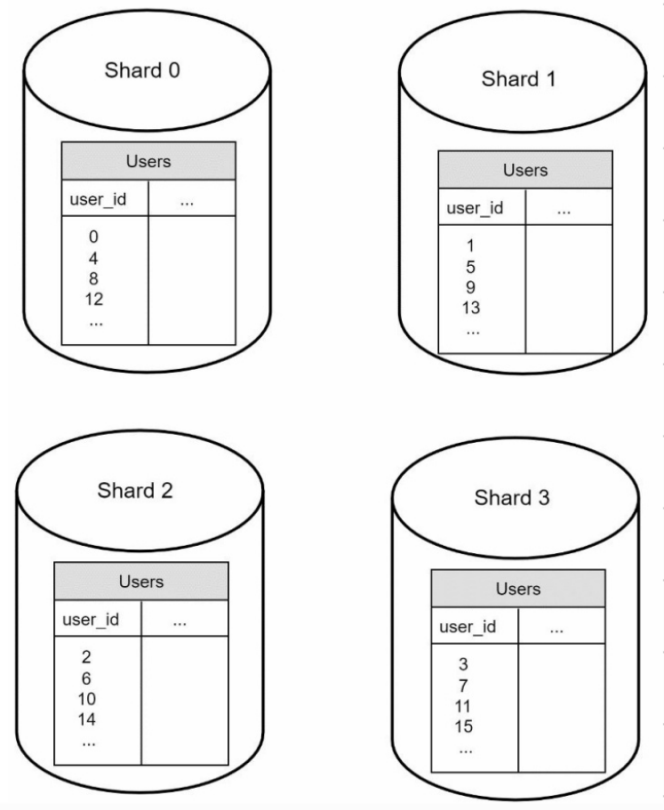
  - Vertical Scaling -
    - Adding more resources to an existing machine. (CPU, RAM, DISK etc).
    - Drawbacks -
      - Hitting hardware limits.
      - Greater risk of SPOF.
      - High Cost.

  - Horizontal Scaling -
    - Process of adding more servers.
    - Also known as sharding.



- Sharding separates larger database into smaller more easily managed parts called shards.





- Sharding Key (Partition Key) consists of one or more columns that determine how data is distributed.
- Complexity and challenges
  - Resharding Data: It is needed when a single shard could no longer hold more data. or certain shards exhausts due to uneven data distribution. Sharding function needs to be updated for moving data around.

- Celebrity Problem: Hotspot key problem. Excessive

access to a specific shard causing server overload.

- Join and de-normalization: Join operations are hard to perform due to sharding. Workaround is to de-normalize the database so that queries can be performed in a single table.

- Updated Design

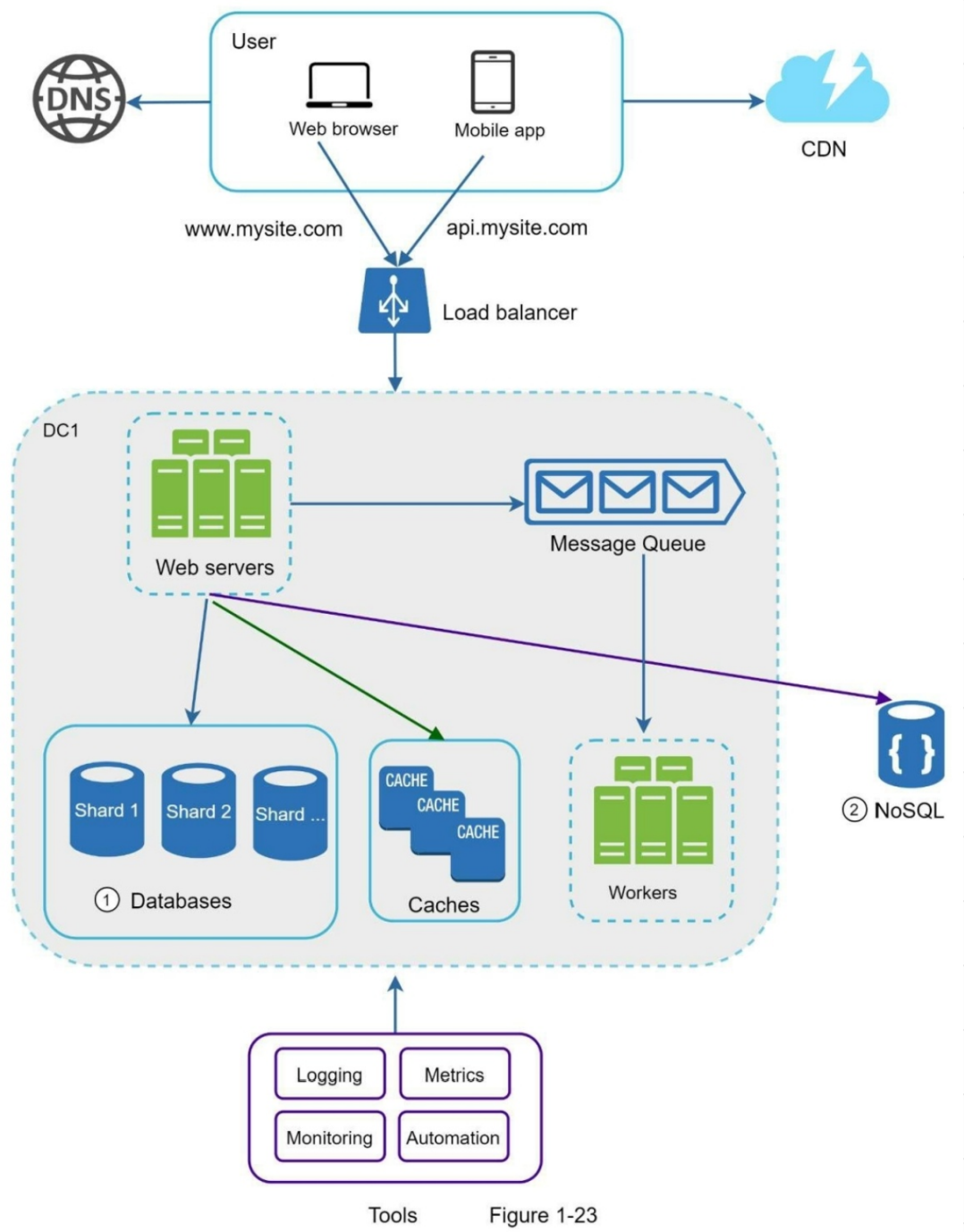


Figure 1-23

- Key Summary

  - Keep web tier stateless
  - Build redundancy at every tier.
  - Cache data as much as possible.
  - Support multiple data centers.
  - Host static assets in CDN.
  - Scale data tier by sharding.
  - Split tiers into individual services.
  - Monitor system and use automation tools.