

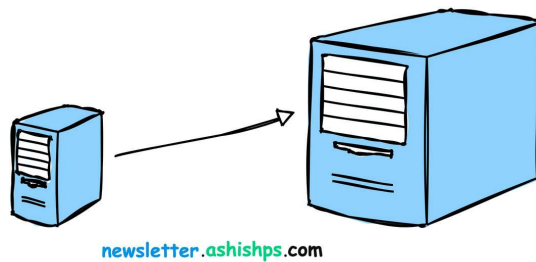
Scalability:

Scalability is the property of a system to handle a growing amount of load by **adding resources** to the system.

How to scale a system ?

1. Vertical Scaling (scale up):

This means adding more power to your existing machines by upgrading server with more RAM, faster CPUs, or additional storage.



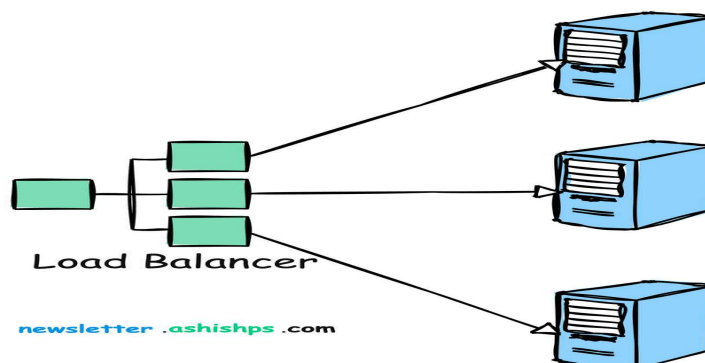
2. Horizontal Scaling (scale out):

This means adding more machines to your system to spread the workload across multiple servers. It's often considered the most effective way to scale for large systems.



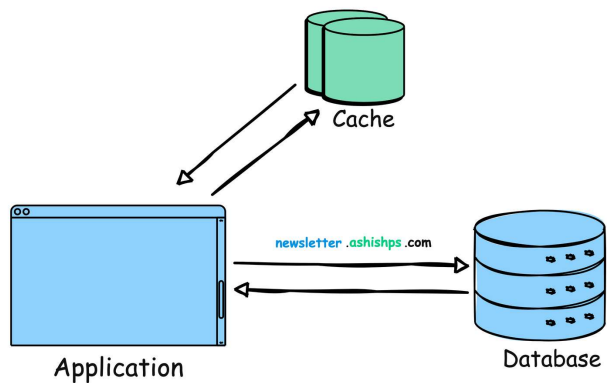
3. Load Balancing:

Load balancing is the process of distributing traffic across multiple servers to ensure no single server becomes overwhelmed.



4. Caching:

Store frequently accessed data in-memory (like RAM) to reduce the load on the server or database. Implementing caching can dramatically improve response times.



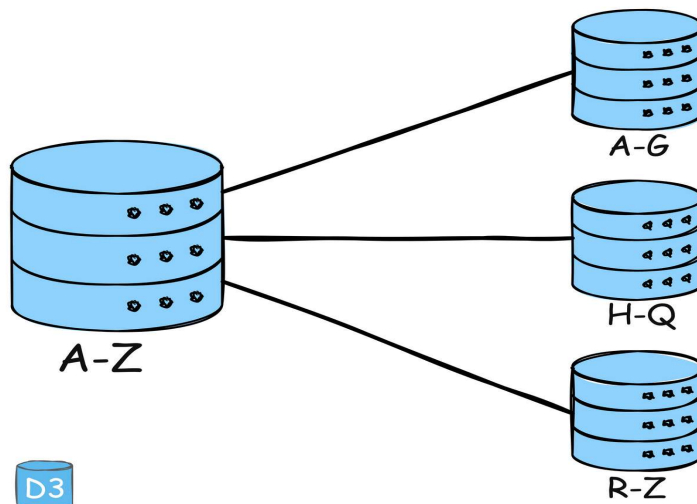
5. Content Delivery Networks (CDNs):

Distribute static assets (images, videos, etc.) closer to users. This can reduce latency and result in faster load times.



6. Partitioning:

Split data or functionality across multiple nodes/servers to distribute workload and avoid bottlenecks.



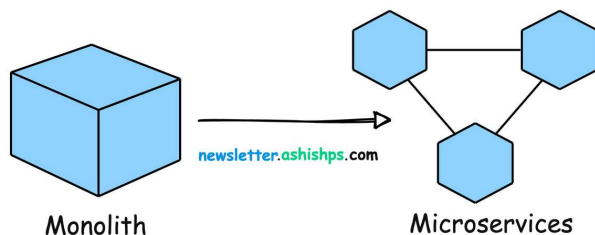
7. Asynchronous Communication:

Defer long-running or non-critical tasks to background queues or message brokers. This ensures your main application remains responsive to user.

Example: Slack uses asynchronous communication for messaging. When a message is sent, the sender's interface doesn't freeze; it continues to be responsive while the message is processed and delivered in the background.

8. Microservices Architecture:

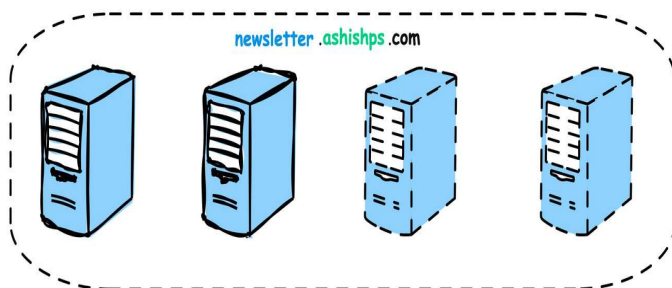
Break down your application into smaller, independent services that can be scaled independently. This improves resilience and allows teams to work on specific components in parallel.



Example: Uber has evolved its architecture into microservices to handle different functions like billing, notifications, and ride matching independently, allowing for efficient scaling and rapid development.

9. Auto-Scaling:

Automatically adjust the number of active servers based on the current load. This ensures that the system can handle spikes in traffic without manual intervention.



10. Multi-region Deployment:

Deploy the application in multiple data centres or cloud regions to reduce latency and improve redundancy.

Example: Spotify uses multi-region deployments to ensure their music streaming service remains highly available and responsive to users all over the world, regardless of where they are located.