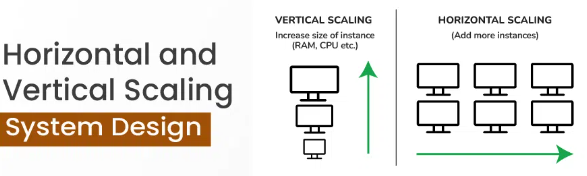
**System Design**

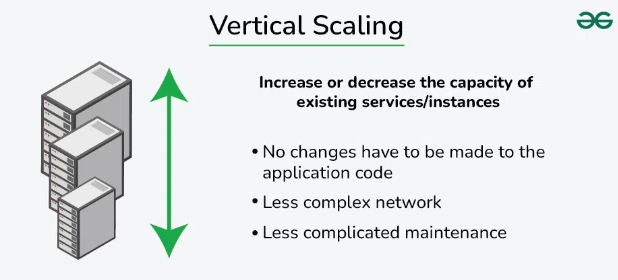
# Horizontal and Vertical Scaling



## What is Vertical Scaling?

Imagine you have a computer at home that you use for various tasks such as web browsing, word processing, and light gaming. Over time, you find that your computer struggles to keep up with more resource-intensive tasks like video editing or running advanced software.

* To address this issue, you decide to upgrade your existing computer by adding more powerful components such as a faster processor, additional RAM, or a larger hard drive.
* By doing so, you’re vertically scaling your system, increasing its capacity to handle more demanding tasks.



**Characteristics of the Vertical Scaling**

* This approach is also known as the ‘**scale-up**‘ approach.
* It doesn’t require any partitioning of data and all the traffic resides on a **single node with more resources**.
* Its implementation is easy.
* Less administrative effort as you need to manage just one system.
* Application compatibility is maintained.
* Mostly used in small and mid-sized companies.
* MySQL and Amazon RDS is a good examples of vertical scaling.

### Advantages of vertical scaling:

### Increased capacity: Upgrading the hardware of a server can improve its performance and increase its capacity to handle incoming requests.

### Easier management: Vertical scaling typically involves upgrading a single node, which can be less complex than managing multiple nodes.

### Disadvantages of vertical scaling:

### Limited scalability: Vertical scaling is limited by the physical constraints of the hardware, whereas horizontal scaling can be easily expanded by adding more nodes.

### Increased cost: Upgrading the hardware of a server can be more expensive than adding more nodes.

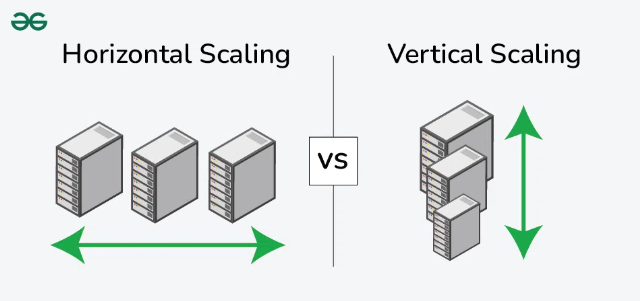
* **Single point of failure:** All incoming requests are still directed to a single server, which increases the risk of downtime if the server fails.

## What is Horizontal Scaling?

Imagine you own a small delivery service company that initially operates with a single delivery van. As your business grows, you start receiving more orders and delivering to a larger area. However, you quickly realize that the single van is not sufficient to handle the increasing demand efficiently.

* To address this issue, you decide to horizontally scale your delivery operation by adding more delivery vehicles to your fleet. Instead of relying on just one van, you purchase several additional vans and hire more drivers.
* By doing so, you’re able to distribute the workload across multiple vehicles, enabling your company to handle a greater volume of packages and serve a larger customer base.

**Note:**Most organizations choose this approach because it includes increasing I/O concurrency, reducing the load on existing nodes, and increasing disk capacity. This approach is the best solution for projects which have requirements for high availability or failover.



### Characteristics of the horizontal scaling

* This approach is also known as the ‘**scale-out**’ approach.
* Horizontal scalability can be achieved with the help of a distributed file system, clustering, and load–balancing.
* Traffic can be managed effectively.
* Easier to run fault tolerance.
* Easy to upgrade.
* Instant and continuous availability.
* Easy to size and resize according to your needs.
* Implementation cost is less expensive compared to scaling up.
* Google with its Gmail and YouTube, Yahoo, Facebook, eBay, Amazon, etc. are heavily utilizing horizontal scaling.
* Cassandra and MongoDB are good examples of horizontal scaling.

### Advantages of horizontal scaling

* **Increased capacity:**More nodes or instances can handle a larger number of incoming requests.
* **Improved performance:**Load can be balanced across multiple nodes or instances, reducing the likelihood of any one server becoming overwhelmed.
* **Increased fault tolerance:** If one node fails, incoming requests can be redirected to another node, reducing the risk of downtime.

### Disadvantages of horizontal scaling

* **Increased complexity:** Managing multiple nodes or instances can be more complex than managing a single node.
* **Increased cost:**Adding more nodes or instances will typically increase the cost of the system.

**Resiliency**: Recover from failure quickly.

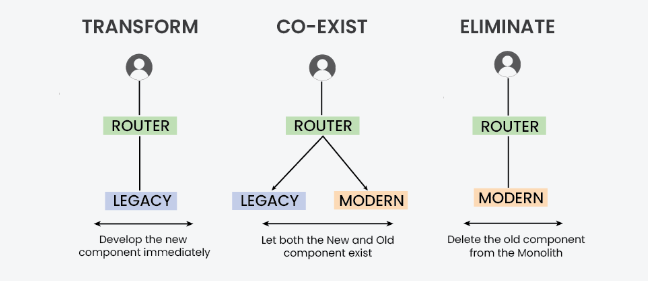
**Fault tolerance:** It is the ability of a system to continue operating normally even when a component fails.

# Strangler Pattern in Micro-services

The Strangler pattern is an architectural approach employed during the migration from a monolithic application to a microservices-based architecture. It derives its name from the way a vine slowly strangles a tree, gradually replacing its growth. Similarly, the Strangler pattern involves replacing parts of a monolithic application with microservices over time.

In order to implement strangler pattern, we need to follow 3 steps that are as follows:

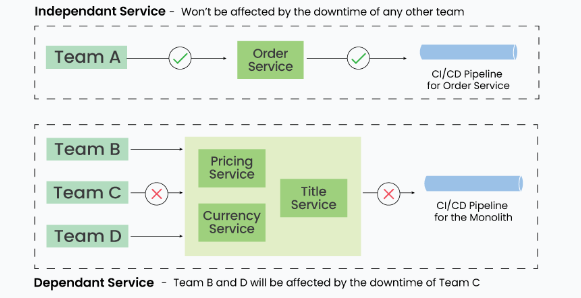
* Transform
* Co-exists
* Eliminate



## Implementation of Strangler Pattern:

Consider an e-commerce application with a monolithic architecture. To migrate the order management functionality to microservices using the Strangler pattern, follow these implementation steps:

* Identify the order management functionality within the monolithic application.
* Create an order management microservice.
* Configure the API gateway to route order management requests to the microservice.
* Migrate specific functionalities from the monolithic application to the microservice.
* Repeat steps 1-4 until the monolithic application is fully replaced.



**Scale from ZERO to MILLION Users:**

**https://medium.com/geekculture/system-design-scaling-from-zero-to-millions-of-users-deca270ef784**