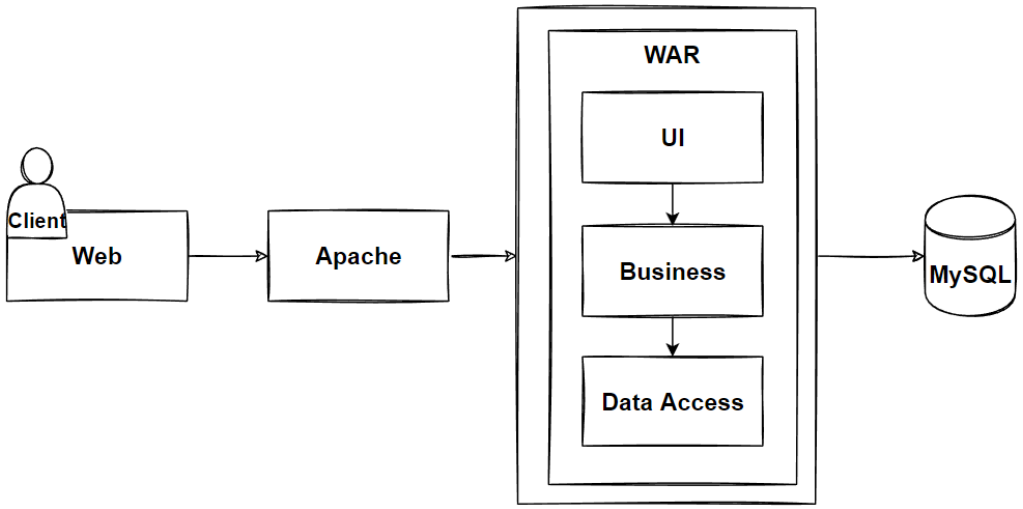
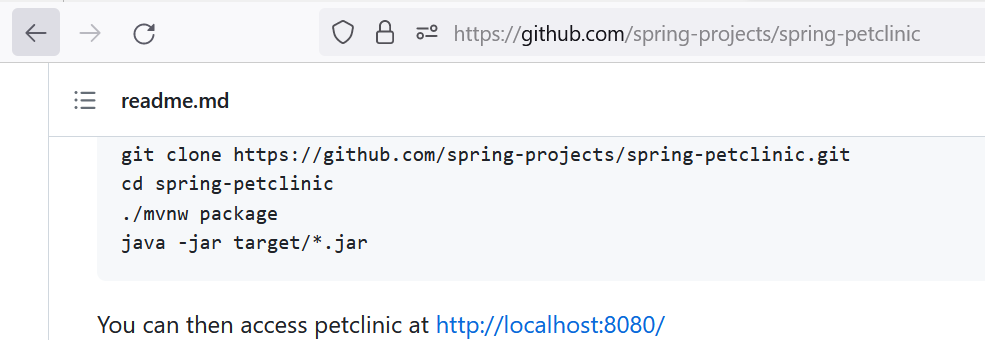
**Monolith Application**

In this category application is developed

* + to run on a single server
  + in single code base
  + in single technology

Architecture:  




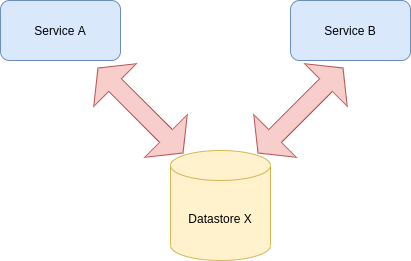
Problems >> Deployment: To make the smallest possible change on business logic server, we need to take the application down and then deploy

Advantages:

Simple to configure Simple to deploy

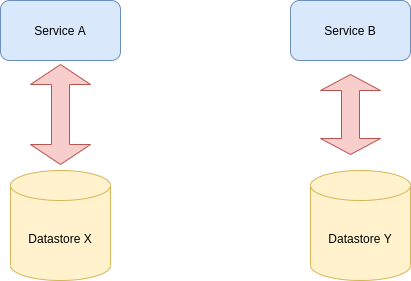
**Distributed Monolith Application**

Application is broken into individually runnable services which share the common database.



**Microservices:**

Application is broken down into smaller individually deployable services with each service having its own database/datastore



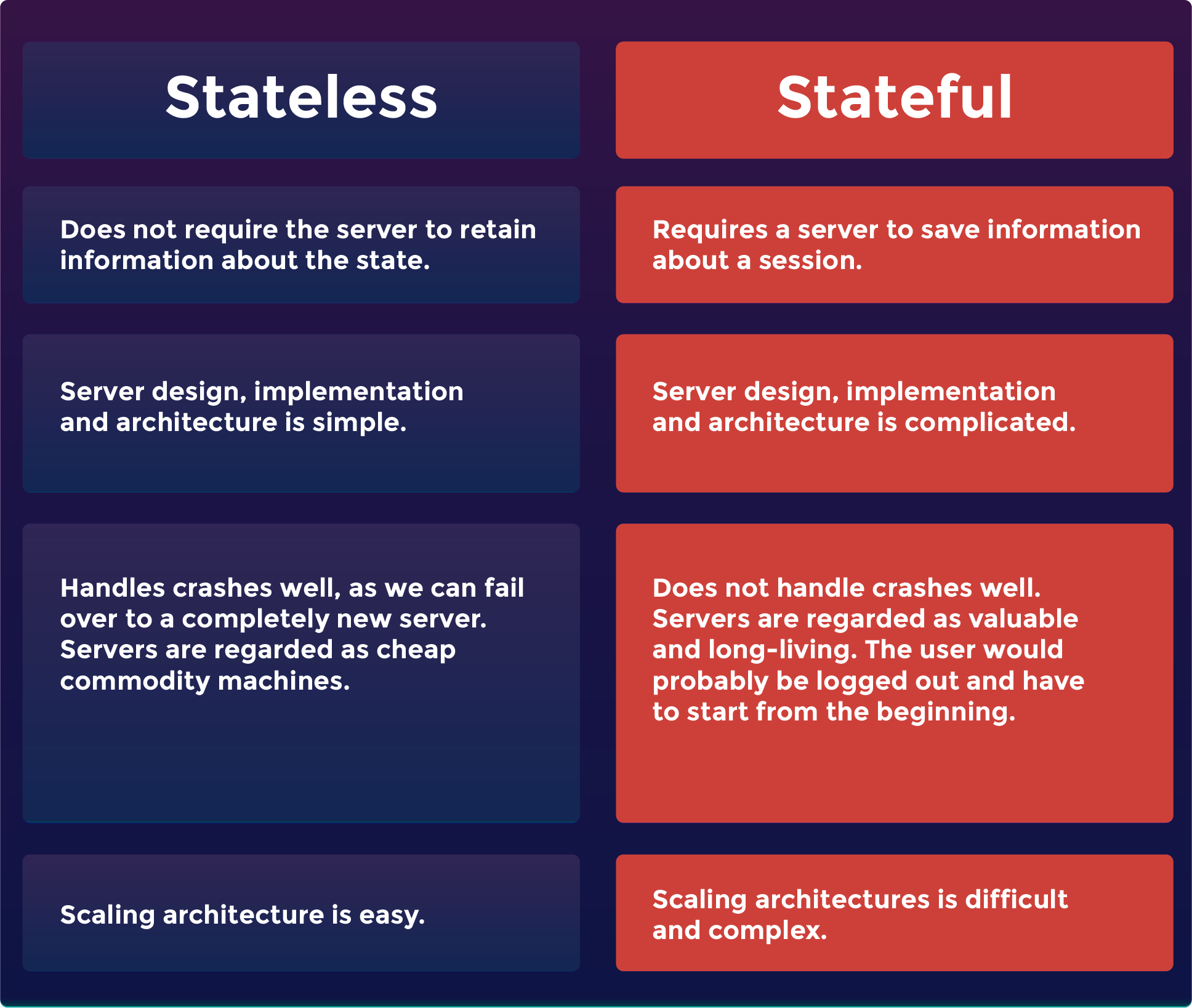
* Generally, domain/problem-based decomposition is popular phenomenon.
* Eventual consistency is important term
* Advantages:
  + Scaling: Individually scale the microservices
  + Technology Changes: each service can be implemented in technology suitable for it.
* Disadvantages:
  + Orchestration:
  + Distributed system challanges

**Terms of Distributed system:**

**Application**: Peace of code that satisfy end user requirement. Code: <https://github.com/2020ht66049/E-learning-portal>

**Stateful vs Stateless application**

The key difference between stateful and stateless applications is that **stateless applications don't “store” data whereas stateful applications require backing storage**. Stateful applications like the Cassandra, MongoDB and mySQL databases all require some type of persistent storage that will survive service restarts.



**Example for Stateful Application:** ELK, Mongodb

**Example of Stateless Application**

* **Performing an online search.** Each query is treated as a unique activity. If the session request is broken, it must be restarted from scratch.
* **Vending machines.** Each transaction is unique and is not linked to prior or future transactions.

**CPU and memory calculation**

The CPU percentage for every process is calculated using the formula given below:

ProcessCPU% = (((val2 - val1) / (time2 - time1)) \* 100) / NumberOfLogicalProcessors

where,

val1 is the CPU percentage of the process taken at time 1

val2 is the CPU percentage of the process taken at time 2

The memory percentage for every process is calculated using the below formula:

ProcessMemory% = ((WorkingSetSize of the process) \* 100) / (TotalVisibleMemorySize of the entire machine \* 1024)

Support Link: <https://support.site24x7.com/portal/en/kb/articles/how-is-the-memory-and-cpu-percentage-for-every-process-calculated-for-a-linux-server-monitor>

**3-tier architecture**

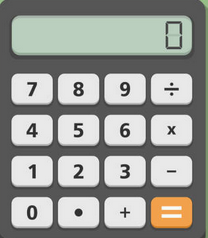


Architectural pattern for a three-tier application

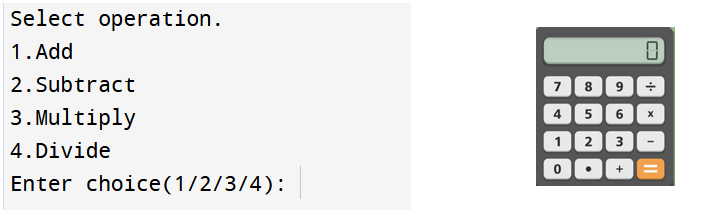


**Business logic vs presentation logic**

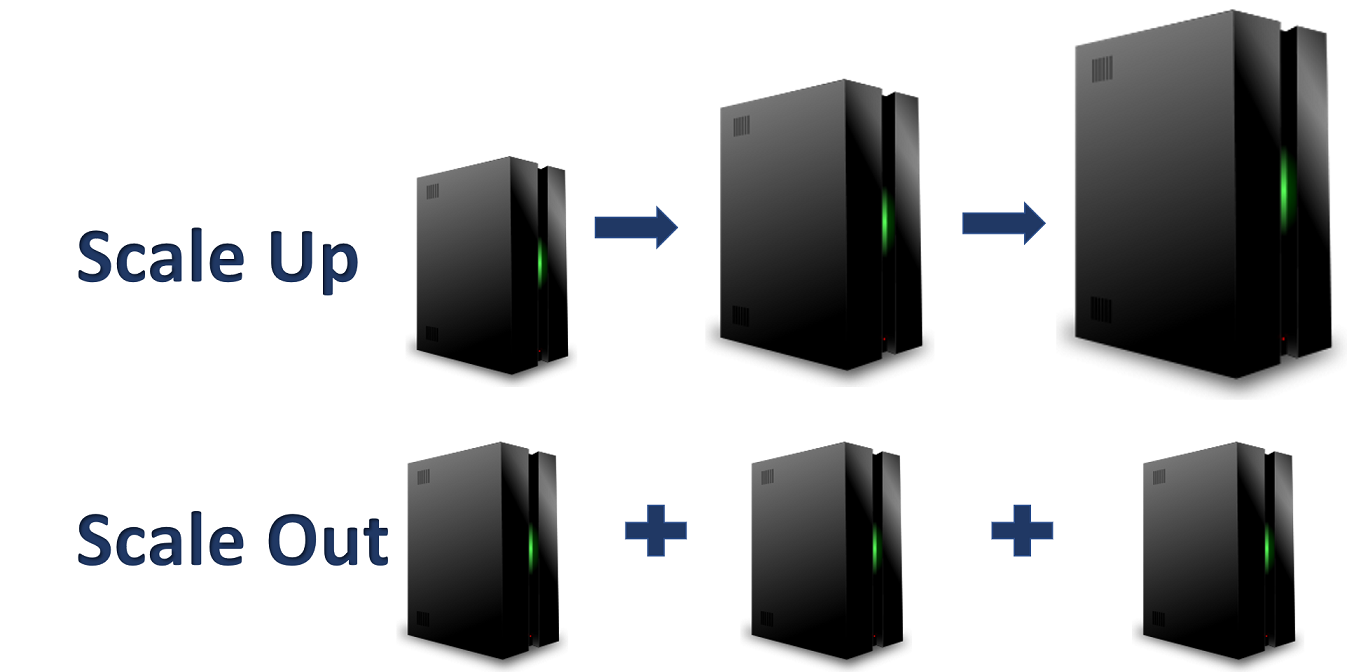
* Business logic: <https://www.programiz.com/python-programming/examples/calculator>
* Presentation logic:



**GUI/CLI**

****

**Scaling up/down [Autos calling]**



**Capacity planning**

Capacity planning is **the process of determining the potential needs of your project**. The goal of capacity planning is to have the right resources available when you'll need them. Resources could mean individuals with the right skills, time available to add another project, or the necessary budget

**Sustainable deployment**

"Sustainable development is **development that meets the needs of the present, without compromising the ability of future generations to meet their own needs**."

**Function**

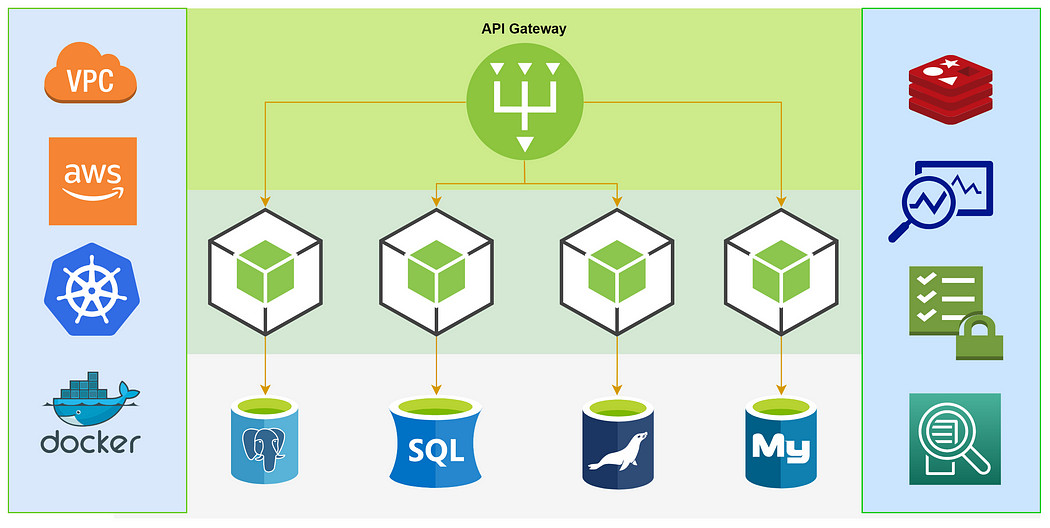
* A function is a block of code which only runs when it is called.
* You can pass data, known as parameters, into a function.
* A function can return data as a result.

def my\_function():  
  print("Hello from a function")

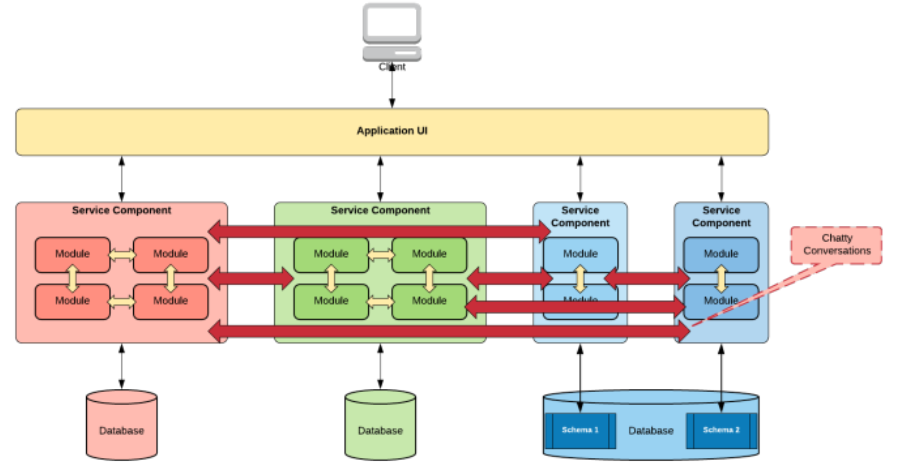
**AWS Lambda**: Function as service.

**AWS Fargate:** Pricing is based on container we used.

**Microservices**



**Inter Service communication**



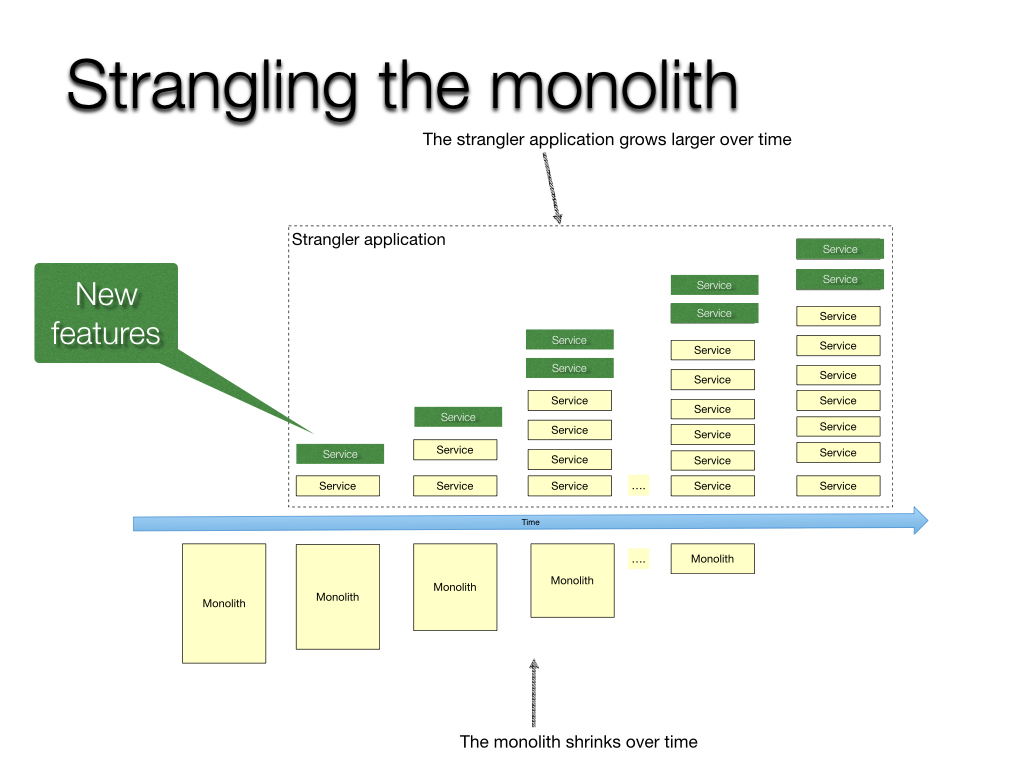
Synchronous vs Asynchronous communication

* Synchronous: Call (or) Face to face call.
* Asynchronous: Email

Reliable communication [based on acknowledgement]

****

**Strangler pattern**



**Authentication and authorisation:** Who are you? and What you have done?

**Sync Operation**: Taking everyone in same page. [Example> Production and DR sync]

**Tightly couple vs loosely couple**

