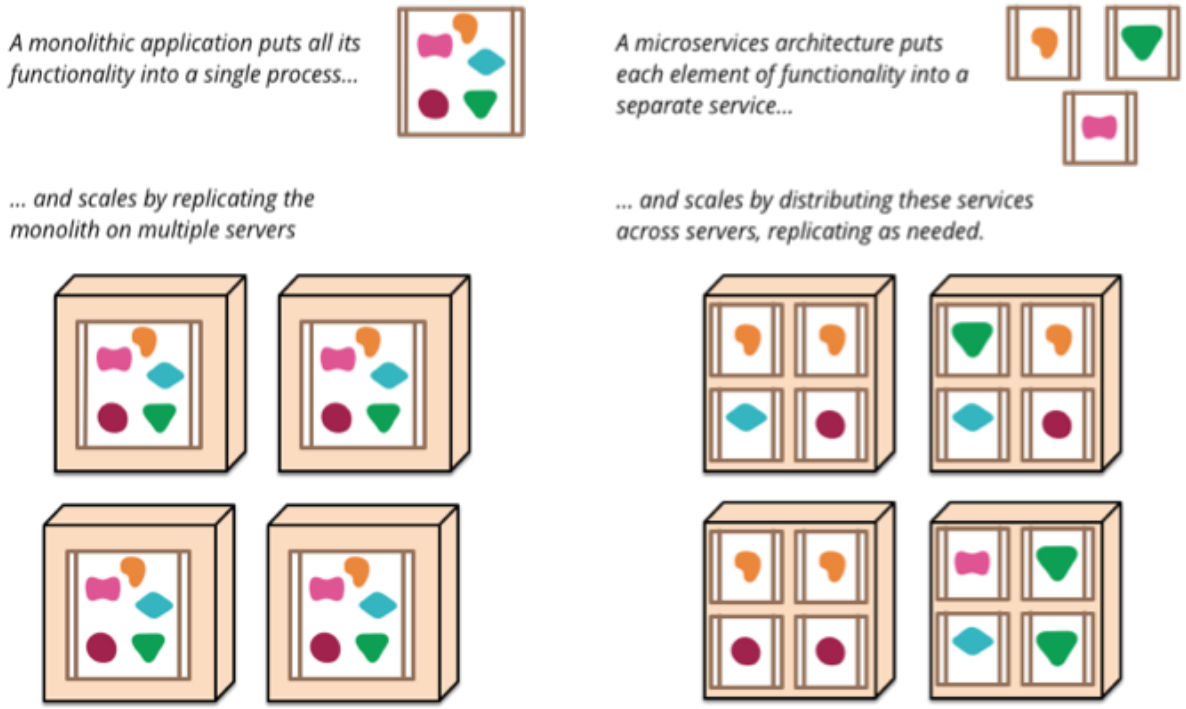
SOA - Microservices

MICROSERVICE DEFINITION

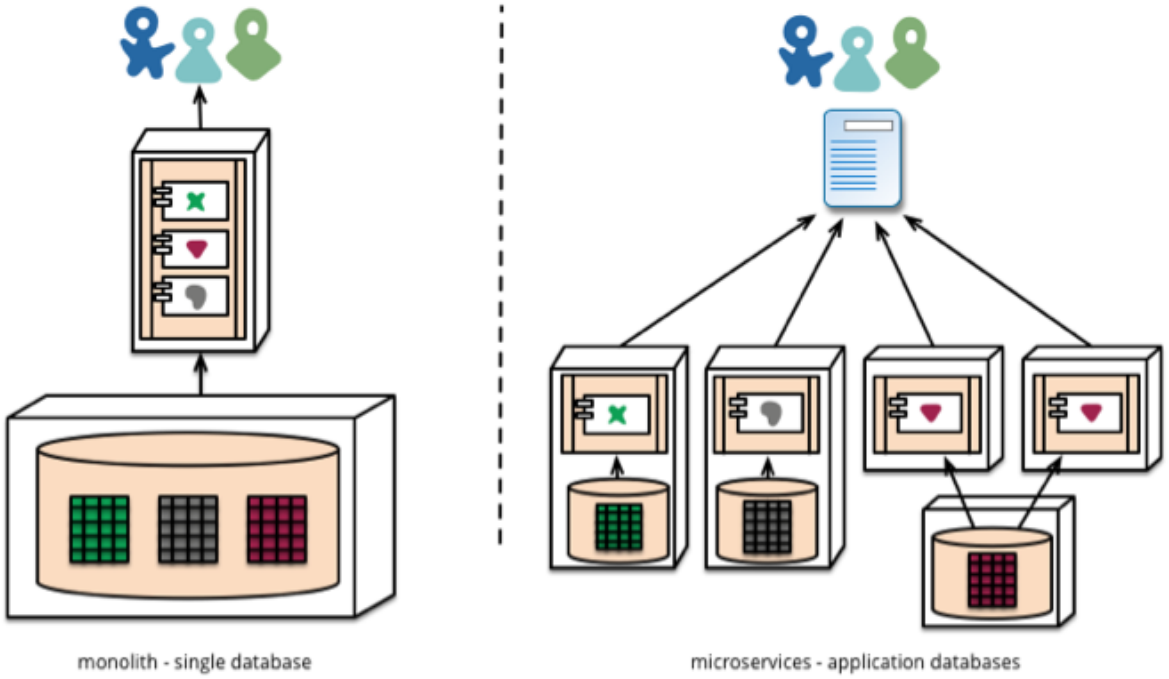
- *The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services*. The trend has grown popular in recent years as Enterprises look to become more Agile and move towards a DevOps and continuous testing.



https://www.martinfowler.com/articles/microservices.html

DECENTRALIZED DATA MANAGEMENT

- Microservices prefer letting each service manage its own database, either different instances of the same database technology, or entirely different database systems.



CHARACTERISTICS OF MICROSERVICES

- **Multiple Components**
  - Software built as microservices can, by definition, be broken down into multiple component services. Why? So that each of these services can be deployed, tweaked, and then redeployed independently without compromising the integrity of an application.
- **Built For Business**
  - The microservices style is usually organized around business capabilities and priorities. Unlike a traditional monolithic development approach—where different teams each have a specific focus on, say, UIs, databases, technology layers, or server-side logic
- **Simple Routing**
  - Microservices act somewhat like the classical UNIX system: they receive requests, process them, and generate a response accordingly.

CHARACTERISTICS OF MICROSERVICES

- **Decentralized**
  - Since microservices involve a variety of technologies and platforms, old-school methods of centralized governance aren't optimal. Decentralized governance is favored by the microservices community because its developers strive to produce useful tools that can then be used by others to solve the same problems.
- **Failure Resistant**
  - Like a well-rounded child, microservices are designed to cope with failure. Since several unique and diverse services are communicating together, it's quite possible that a service could fail, for one reason or another (e.g., when the supplier isn't available).
- **Evolutionary**
  - Microservices architecture is an evolutionary design and, again, is ideal for evolutionary systems where you can't fully anticipate the types of devices that may one day be accessing your application.

Source: https://smartbear.com/solutions/microservices/

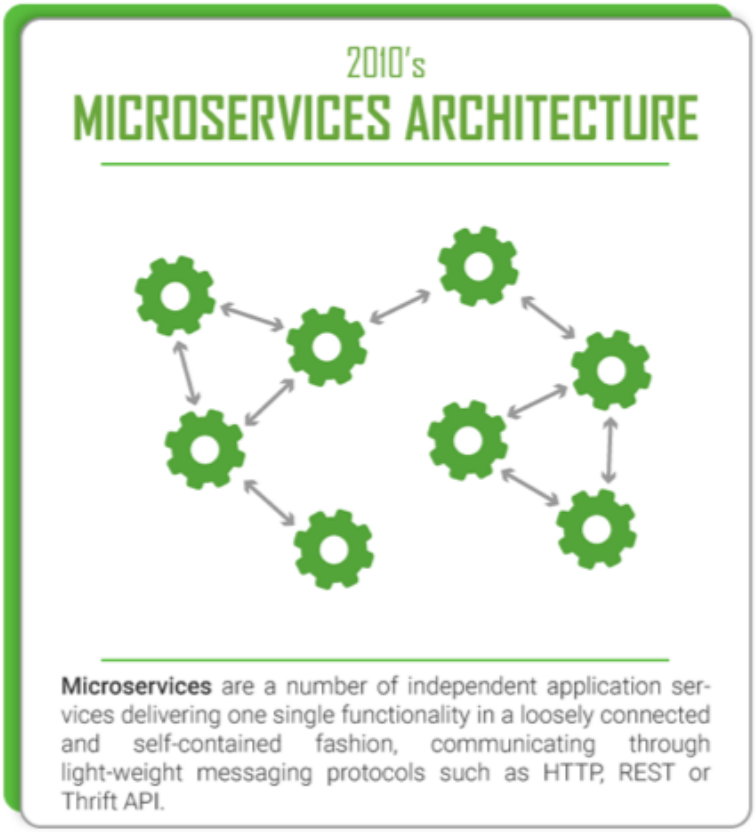
MICROSERVICE PROS AND CONS

- Pros
  - Microservice architecture gives developers the freedom to independently develop and deploy services
  - A microservice can be developed by a fairly small team
  - Code for different services can be written in different languages (though many practitioners discourage it)
  - Easy integration and automatic deployment (using open-source continuous integration tools such as Jenkins, Hudson, etc.)
  - Easy to understand and modify for developers, thus can help a new team member become productive quickly
  - The developers can make use of the latest technologies
  - The code is organized around business capabilities
  - Starts the web container more quickly, so the deployment is also faster
  - When change is required in a certain part of the application, only the related service can be modified and redeployed—no need to modify and redeploy the entire application
  - Better fault isolation: if one microservice fails, the other will continue to work (although one problematic area of a monolith application can jeopardize the entire system)
  - Easy to scale and integrate with third-party services
  - No long-term commitment to technology stack

MICROSERVICE PROS AND CONS

- Cons
  - Due to distributed deployment, testing can become complicated and tedious
  - Increasing number of services can result in information barriers
  - The architecture brings additional complexity as the developers have to mitigate fault tolerance, network latency, and deal with a variety of message formats as well as load balancing
  - Being a distributed system, it can result in duplication of effort
  - When number of services increases, integration and managing whole products can become complicated
  - In addition to several complexities of monolithic architecture, the developers have to deal with the additional complexity of a distributed system
  - Developers have to put additional effort into implementing the mechanism of communication between the services
  - Handling use cases that span more than one service without using distributed transactions is not only tough but also requires communication and cooperation between different teams

MICROSERVICES VS. SOA



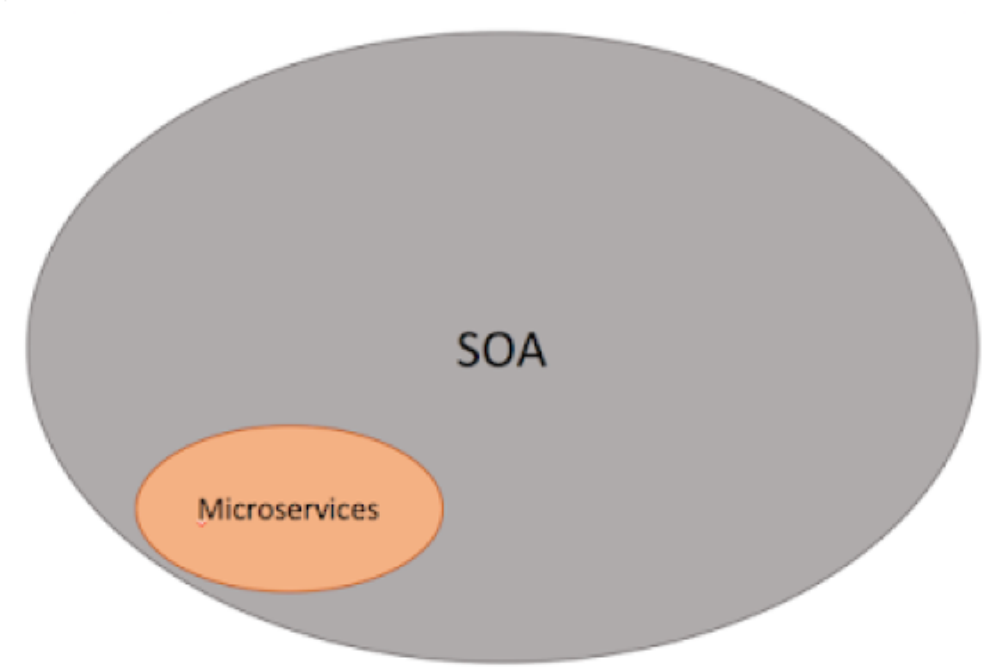
Source: https://dzone.com/articles/microservices-vs-soa-is-there-any-difference-at-al

MICROSERVICES VS. SOA

SERVICE-ORIENTED ARCHITECTURE	MICROSERVICES ARCHITECTURE
Maximizes application service reusability	Focused on decoupling
A systematic change requires modifying the monolith	A systematic change is to create a new service
DevOps and Continuous Delivery are becoming popular, but are not mainstream	Strong focus on DevOps and Continuous Delivery
Focused on business functionality reuse	More importance on the concept of “bounded context”
For communication it uses Enterprise Service Bus (ESB)	For communication uses less elaborate and simple messaging systems
Supports multiple message protocols	Uses lightweight protocols such as HTTP, REST or Thrift APIs
Use of a common platform for all services deployed to it	Application Servers are not really used, it’s common to use cloud platforms
Use of containers (such as Docker) is less popular	Containers work very well with microservices
SOA services share the data storage	Each microservice can have an independent data storage
Common governance and standards	Relaxed governance, with greater focus on teams collaboration and freedom of choice

MICROSERVICES VS. SOA

- The main difference between SOA and microservices lies in the size and scope. Microservice has to be significantly smaller than what SOA tends to be and mainly is a small(er) independently deployable service. On the other hand, an SOA can be either a monolith or it can be comprised of multiple microservices.



RECOMMENDED READING

- https://www.martinfowler.com/articles/microservices.html
- https://smartbear.com/solutions/microservices/
- https://dzone.com/articles/api-management-for-heroku-applications
- https://dzone.com/articles/the-future-of-spring-cloud-microservices-after-net
- https://www.youtube.com/watch?v=PY9xSykods4