MICROSERVICES
BEGINNING OF A NEW ERA

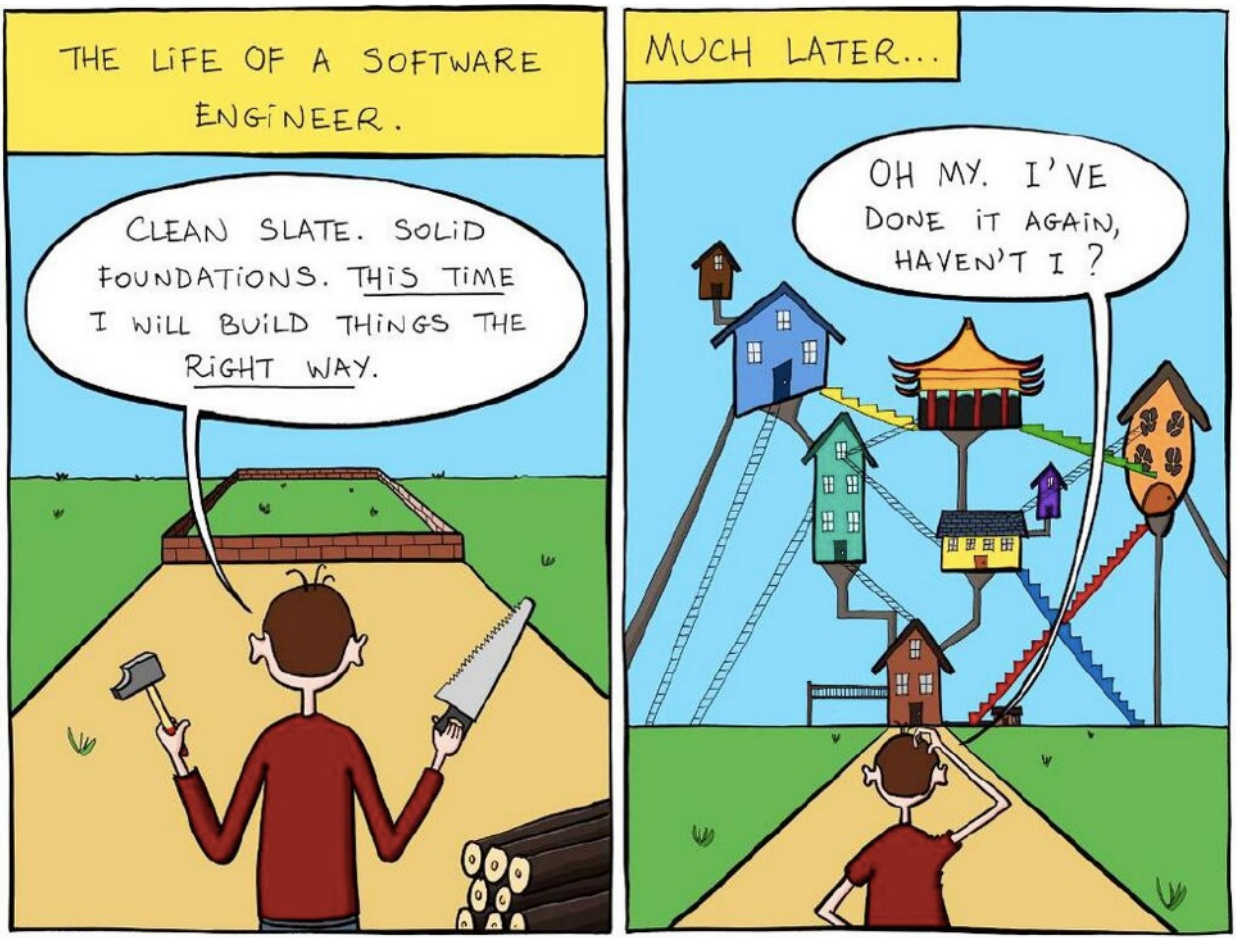
Early Ages!

Traditional application development has always followed a monolithic approach of dividing functionalities into methods and classes but inside the same single unit which is deployed as a process. Such an architecture allows for proper testing of the application as well as usage of the deployment pipeline.

However, this approach shows severe limitation when considering scaling of systems to multiple users and rapid deployment cycles. Each and every change requires the entire application to be re-deployed.

Furthermore, while scaling up, the entire application is in duplicated onto servers instead of just scaling the functions which are facing a higher load.

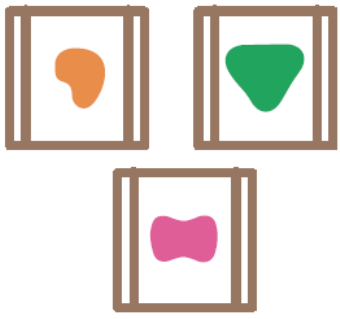
Monolithic Approach



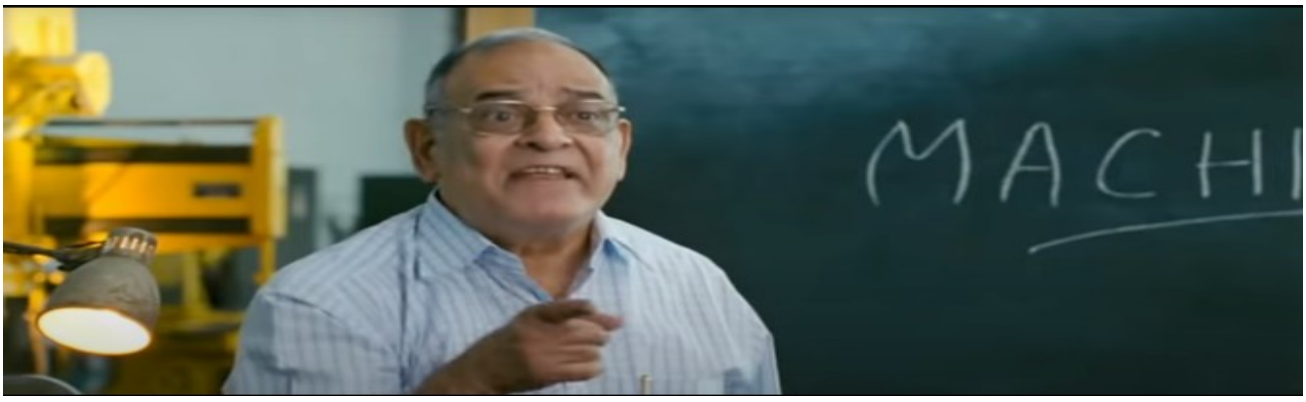
For this reason, companies like Netflix and Spotify popularized the use of new style of software development based on r

A monolithic application puts all its functionality into a single process...

A microservices architecture puts each element of functionality into a separate service...



Monolithic Architecture Definition

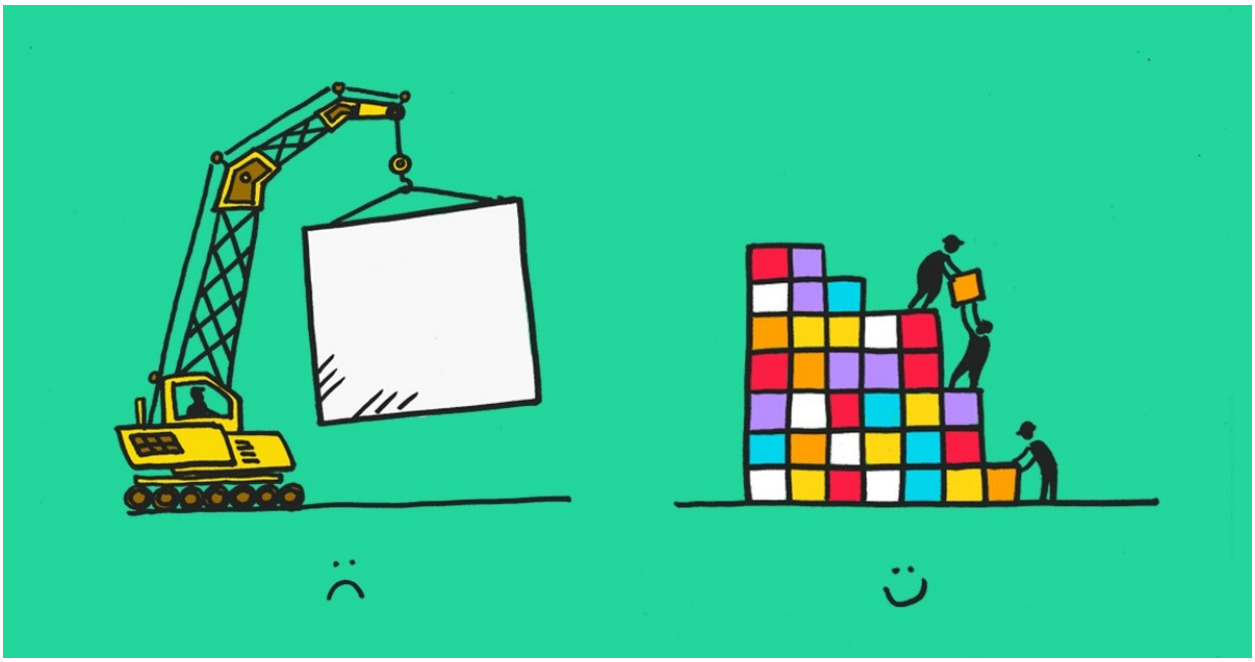


Monolithic architecture is something that build from single piece of material, historically from rock. Monolith term normally use for object made from single large piece of material.

A monolithic architecture is the traditional unified model for the design of a software program. Monolithic, in this context, means composed all in one piece. Monolithic software is designed to be self-contained; components of the program are interconnected and interdependent rather than loosely coupled as is the case with modular software programs. In a tightly-coupled architecture, each component and its associated components must be present in order for code to be executed or compiled

Era of Microservices.

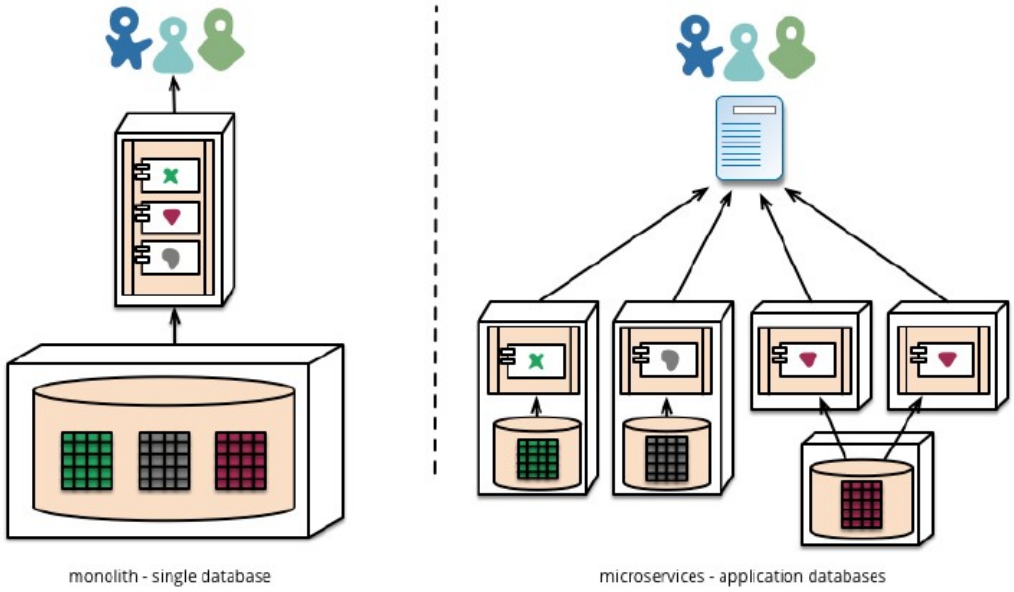
Recently monoliths designs were effectively abandoned by architects in favor of native microservices, bringing agility to the table by speeding up changes and lowering the risk of these changes. Microservices are nowadays the evolution of software development.



“Microservice architecture” is an approach of building large enterprise application with multiple small unit called service, each service develops, deploy and test individually. Each service intercommunicates with a common communication protocol. Each service run individually either in single machine or different machine but they execute its own separate process. Each service may have own database or storage system or they can share common database or storage system. Microservice is all about distribute or break application in small chunks.

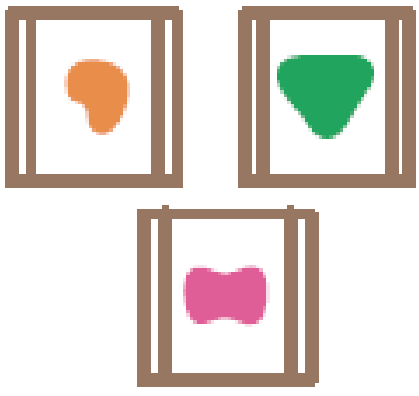
Independent Process

Microservices help build an application as a suite of small services, each running in its own process and are independently deployable.

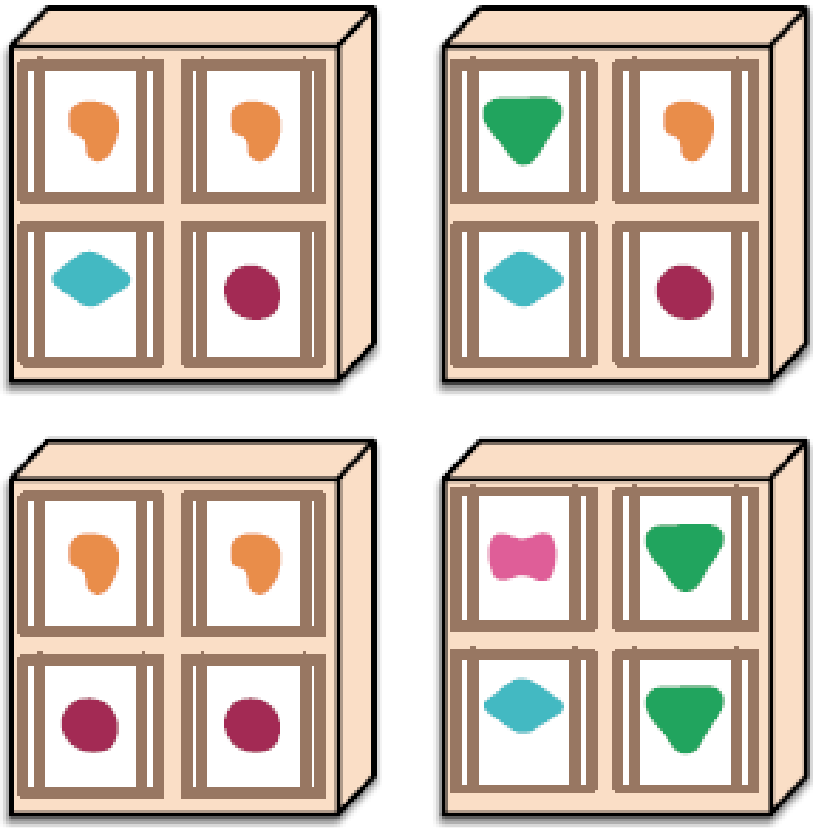


Scaling

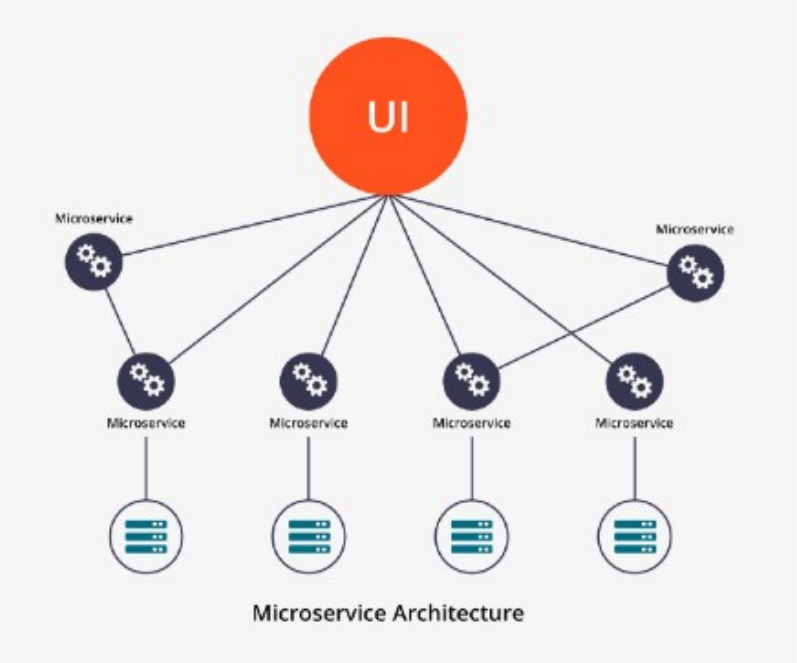
A microservices architecture puts each element of functionality into a sperate service.



and these can be scaled by distributing these services across servers, replicating as needed.



Microservice is more than code and structure. It’s way of working. Its culture in a way, every developer or team own some part of large application. Microservice is not ultimate solution for every application but it’s surely solution for large enterprise application. Microservice has some shortfalls like increase lot of operations overhead, DevOps skills required, complex to manage because of distributed system, bug tracking become challenging.



Microservice Architecture

Pros of Microservices

Microservices have become hugely popular in recent years. Mainly, because they come with a couple of benefits that are super useful in the era of containerization and cloud computing. You can develop and deploy each microservice on a different platform, using different programming languages and developer tools. Microservices use APIs and communication protocols to interact with each other, but they don't rely on each other otherwise.

The biggest pro of microservices architecture is that teams can develop, maintain, and deploy each microservice independently. This kind of single-responsibility leads to other benefits as well. Applications composed of microservices scale better, as you can scale them separately, whenever it's necessary.

Cons of microservices

As microservices heavily rely on messaging, they can face certain problems. Communication can be hard without using automation and advanced methodologies such as Agile. You need to introduce DevOps tools such, configuration management

platforms, and tools to manage the network. This is great for companies who already use these methods. However, the adoption of these extra requirements can be a challenge for smaller companies.

Communication between microservices can mean poorer performance, as sending messages back and forth comes with a certain overhead. And, while teams can choose which programming language and platform they want to use, they also need to collaborate much better. After all, they need to manage the whole lifecycle of the microservice, from start to end.

Conclusion

Microservices based design pattern is a relatively new concept. Other than Amazon and Microsoft’s platforms, others still are under active development. This indicates that the disadvantages of such a microservices based architecture, is still something that remains to be fully studied.