Project 1

Microservices Architecture in Industry

Introduction

Microservices architectures are a relatively new and popular way to design and deploy systems. In this paper, we'll describe microservices architectures and analyze their benefits and drawbacks.

Microservices Definition

Microservices architecture (MSA) is the modern way of building and delivering applications. Unlike the monolithic method, which centralized all code in a single area as a single-tiered application from a single platform, MSA divides the application into smaller components. Each of which is responsible for a specific activity or functionality, and each service runs its process. They communicate to each other with lightweight mechanisms, usually an HTTP resource API (e.g., REST APIs). From there, all these microservices will work together to give complete functional applications to end-users.

Traditionally, the applications are developed using a monolithic model. That means a whole team of developers works on a centralized code repository together. Once a change is made, they make changes to the whole application stack. Monolith would be known as easy to deploy the application, manage services, and develop components. However, when the application is enhanced and becomes to scale, monolith meets its limitation with respect to the scalability and continuous integration.

Since all of the code is centralized in one repository, developers may find problems in testing and fixing bug because they might need to go through the entire source code for troubleshooting and running extensive testing. At this point, the source code already become very large, complex and hard to manage [3]. Therefore, it leads to two major issues in monolith that are complicated scalability and fault tolerance capability. It would not be possible to scale up one or several components and not make the huge impact on the whole application. Consequently, if one component is down or malfunctioning, the whole application is considered an operational failure from the end user's perspective.

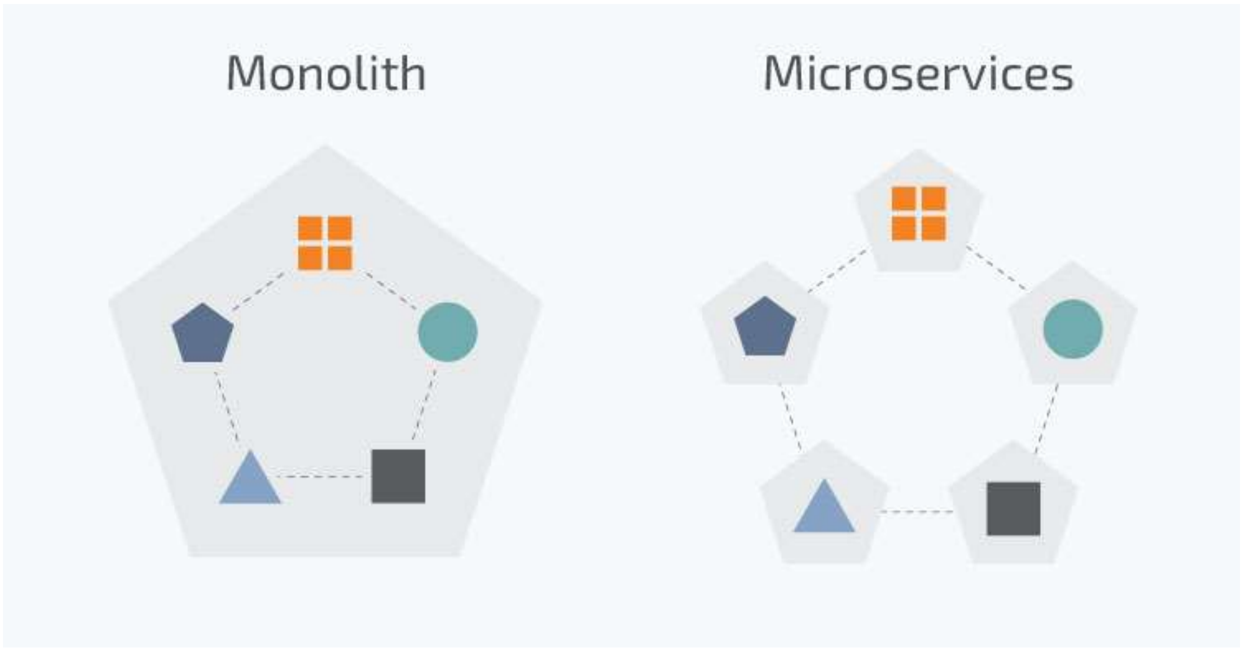


Figure 1: Monolith and Microservices architecture. [3]

As mentioned, MSA, on the other hands, breaks the application into smaller separate autonomous components. They work independently to each other with their own functional logic and database. It not only solves these challenges described above but also contributes significant enhancements while developing an enterprise application system.

Bug fixing or troubleshooting gets easier since the source code's functionality is split and deployed into each individual service. If the developers are looking for a specific problem, they would need to look at the relevant microservice, not the entire application system. It means MCS isolates the failure. As a result, automation testing and continuous integration are also simplified because only one microservice is modified and not impact to the whole system [4].

With the decentralization in microservice, each component can be scaled up or down at ease independently. The developer team and the company would be beneficial of wide range of technology alternatives instead of stick in a stack compatibility in monolith [4]. Database or software would be selected with the best suit for the company situation, budget, and functionality's requirements. Finally, the fault tolerance issue is solved as any issue in a specific microservice will only break that service rather than the entire application.

Apart of the strong features offered from MSA, it has several drawbacks that are considered whether MSA is the best suit for the application. The potential downsides are:

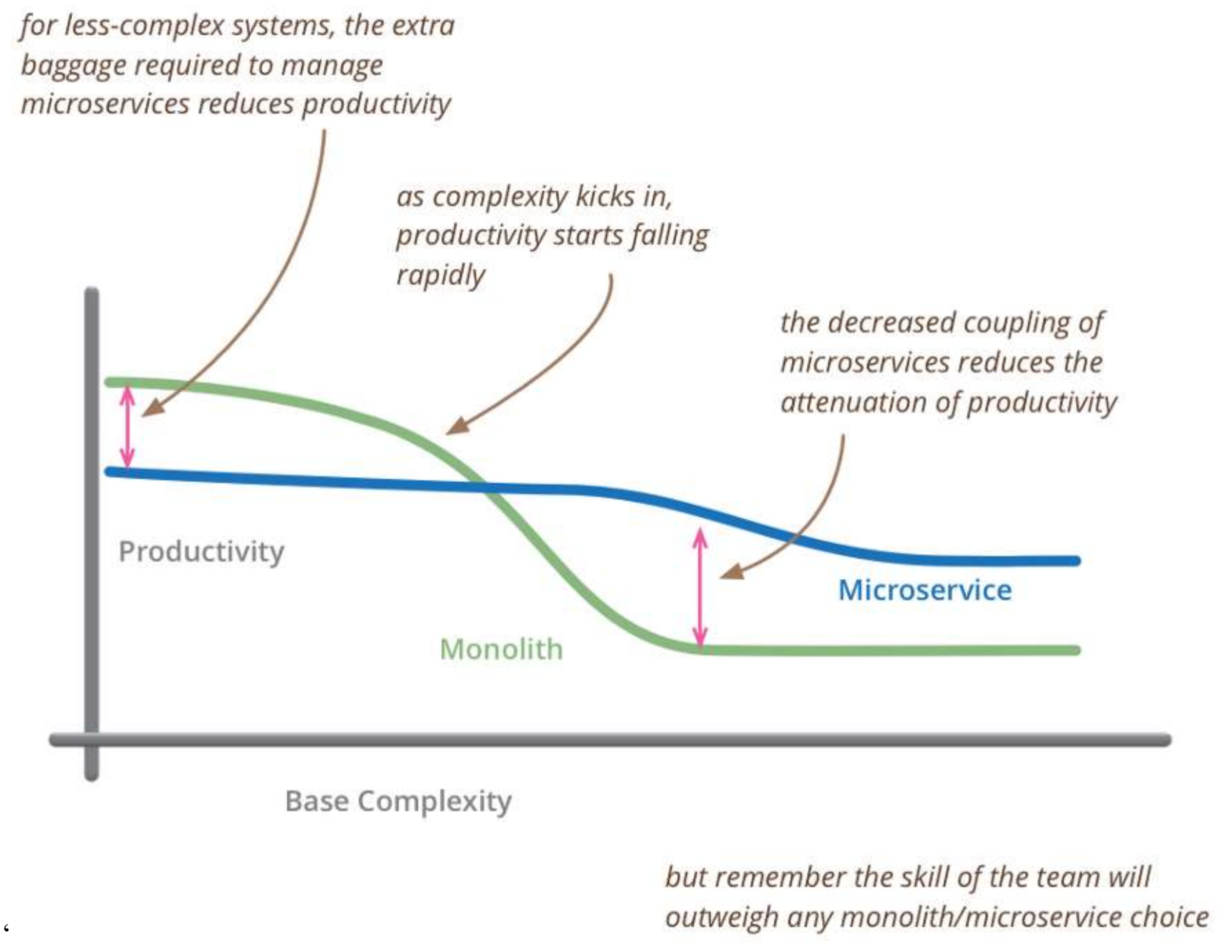
- **Extra complexity**: MSA is a decentralized system, it takes extra effort to set up the communication connections among services and among the service with its dependencies. The process must be deployed independently.
- **System distribution**: With the complexity of MSA, all modules and their databases, components have to be managed carefully in various procedure.
- **Cross-cutting concerns**: Monitoring such as "configuration, logging, metrics, health checks" becomes harder to implement with time and effort.
- **Testing:** Each module or software may require different testing component suites, which makes workload for testing phase heavier. [3]

When Microservice Architecture

Any organization wishes to shift to a new framework has to conduct thoroughly due diligence to ensure its suitability. Exploring and adapting with microservices, there are several situations and factors contribute to the decision whether the microservices architecture is the right choice for the application.

- **Microservices expertise:** the teams have essential skills and knowledge with MSA. It is not only about the developer team but also the DevOps and Containers expert team to make sure all components are connected and deployed efficiently. [2]
- **Enhance application:** the company grows bigger and is in high demand of "[accommodating] scalability, agility, manageability and delivery speed" [5] for their application. MSA would be complicated at the beginning, if the company has a clear vision of developing a large enterprise application with "multiple modules and user journeys", [2] MSA is the good fit.
- **Engineering resources:** A microservice project requires the combination of difference service technologies. There are multiple teams and each of them is responsible for one or several services handling. Then, if the company needs to make sure that they have enough resources to mange those components, processes. [2]

When NOT Microservice Architecture



‘

Figure 2: Productivity and Complexity between Microservice and Monolith [1]

The figure above shows the productivity benefits of MSA but also specifies a higher complexity compared with monolith. MSA would bring higher availability and beneficial for developing an application. However, it would add extra complexity that might overwhelm the team. Therefore, it worth to consider when a company should use the monolith approach:

- **Small team**: The company is a startup or a small size organization. The company does not have enough engineering resources, and it is the ideal for handling MSA. Instead, starting with monolith can meet the current goal and situation.
- **A simple application:** If the company plans to produce a small application for serving a certain business need, it will not require a superior scalability, high demand, or flexibility. Then, there is no need of MSA at this point.
- **No microservices expertise:** As mentioned, “Microservices requires profound expertise to work well and bring business value”, even though the company has large resources, it would not be an appropriate solution when there is no microservice technical experts support.
- **Quick launch:** If application can be developed and it does not need to be split into microservices, monolithic would be an ideal model. The application can be deployed and launch faster [3].

References

[1] Fowler, M. (2015, May 13). *Bliki: Microservicepremium*. martinfowler.com. Retrieved October 22, 2021, from https://martinfowler.com/bliki/MicroservicePremium.html.

[2] Curry, D. (2021, August 30). *Uber eats revenue and Usage Statistics (2021)*. Business of Apps. Retrieved October 23, 2021, from https://www.businessofapps.com/data/uber-eats-statistics/.

[3] Gnatyk, R. (2018, October 3). *Microservices vs Monolith: Which architecture is the best choice for your business?* Microservices vs Monolith: which architecture is the best choice? Retrieved October 22, 2021, from https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/.

[4] *How microservices architecture helps businesses*. Cleo. (n.d.). Retrieved October 22, 2021, from https://www.cleo.com/blog/microservices-architecture.

[5] Shah, H. (2020, December 23). *Best of 2020: When to use - and not to use - microservices*. Container Journal. Retrieved October 22, 2021, from https://containerjournal.com/topics/container-ecosystems/when-to-use-and-not-to-use-microservices/.