3 Things about Microservices

*I will be writing this blog post with an intention that people like myself with no or little background knowledge of software development, or people who may be new learners in the field, be able to get a thing or two out of it. So no advanced technical jargon or anything like it.*

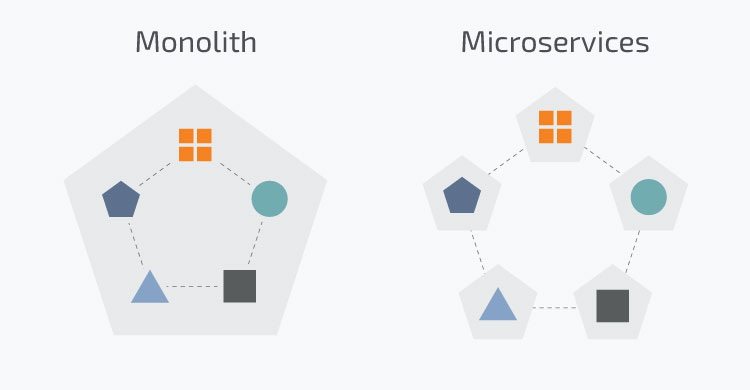
Source: <https://itnext.io/how-to-use-php-to-implement-microservice-94957206abc6>

1. What is Microservices?

Just like everything else in life there are different ways of doing things, so when we talk about how software applications are thought-out, designed, built, tested and implemented we have to determine how exactly we are going to structure our Software Architecture.

Microservices is an architecture where various functions of an application are divided into different modules, each responsible for its own specific job. Yeah it’s that's simple.

According to Renowned author, software consultant and speaker Martin Fowler, “The Microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.” (openxcell.com - Priyanka Garg, 2017)

Source: <https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/>

2. Comparison with Monolithic Architecture.

Microservices are definitely becoming popular more than ever, but still traditional approach – Monolithic Architecture and Modular Monolithic Architecture both have their tried and tested benefits which might suit to some cases. Below is a generic comparison between the two:

|  |  |
| --- | --- |
| **Microservices** | **Monolithic** |
| Easier understanding – Separate modules. | One Unit - easy to build, can be initiated and managed by a single engineer or a small team. |
| Modern approach for large enterprises where more complex software application are the norm. | Easier debugging and testing for lean projects. |
| Varies from case to case, there are advantages and disadvantage of this. | One size fits all makes it adaptable. |
| Scalability is one of the key benefits of deploying separate independent modules. | Scalability is an issue, as it becomes a ‘big ball of mud’. |
| One software/application can have different modules written in different languages. | New technology cannot be at all or in some cases difficult to integrate. |
| Security is one thing that will take both time and money. | Tight-coupling makes it really difficult to manage changes and improvements. |
| Microservices can easily be replaced. | With time it becomes really complicated, time consuming and costly. |

3. Final verdict on its adaptation.

Adopting a Microservices Architecture is not a one-size-fits-all approach. Despite being less and less popular, a monolith has its strong and durable advantages which work better for many use cases.

If your business idea is fresh and you want to validate it, you should start with a monolith. With a small engineering team aiming to develop a simple and lightweight application, there is no need to implement microservices. This way, a monolithic application will be much easier to build, make changes, deploy, and provide testing.

The microservices architecture is more beneficial for complex and evolving applications. It offers effective solutions for handling a complicated system of different functions and services within one application. Microservices are ideal when it comes to the platforms covering many user journeys and workflows. But without proper microservices expertise, applying this model would be impossible. (n-ix.com - Romana Gnatyk, 2018)