In this essay, I'm going to discuss the difference between Microservices and Monolithic Architecture. They look similar from the perspective that there is one grand application that does a lot of multiple things, but it's too general, let's go deeper, let's find out what is the difference between those two approaches.

Let's start with the deployment. It's a necessary step for any kind of system. If deployment went incorrectly, in the best-case scenario, the company has a backup system and you'll be able to start all over again. Monolith apps allow you to set your deployment once and then simply adjust it based on ongoing changes. Microservices require much more work because each of the microservices deployed independently. The nice part about it is that if something goes wrong you will only break one piece, which is a lot less problematic to fix than the entire project.

Maintenance is another big piece of any project, and from the perspective of maintenance, the biggest feature of microservices architecture is a huge disadvantage because it's a lot harder to track all changes if they're located in multiple different places.

Obviously, the reliability of the Monolith architecture won't be as good as the reliability of the Microservices architecture. Let's think about the app as a logical tree for a second. In the case of the Monolith, If we're going to implement something on the top of the tree, it's going to affect all of the aspects of it, and if something breaks, it's a lot harder to track it, because even if we're entirely sure where exactly the error has occurred, we may not know what exactly caused the object to break. In the case of the Microservices, we're implementing new features, not at the top of the tree, but to each aspect separately, so if something went wrong, it'll break only this exact entity.

From a scalability perspective, the Microservices architecture is a winner again. When we're using the microservices application we're a step ahead from the beginning because the application itself already a collection of loosely coupled services, so it's a lot easier to divide between different workers. In the case of the monolith applications sometimes it's just impossible to add extra workers at all.

As we can see Microservices and Monolith architectures are very different from each other and they have a completely different idea behind. In my opinion, it's better to start the application as the Monolith ones and then switch to the Microservices. Microservices architecture brings a lot of cool features that aren't as important at the beginning, but they do sound very juicy later on. It gives advantages such as the ability to: keep changing the product constantly without need of going through the entire project every time, decompose the complex project into a set of manageable services. Also, it significantly improves the scalability and reliability of the project.