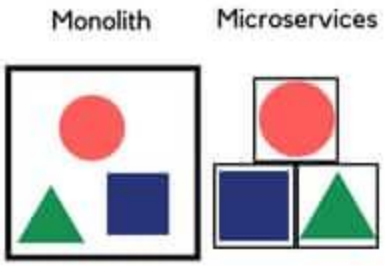
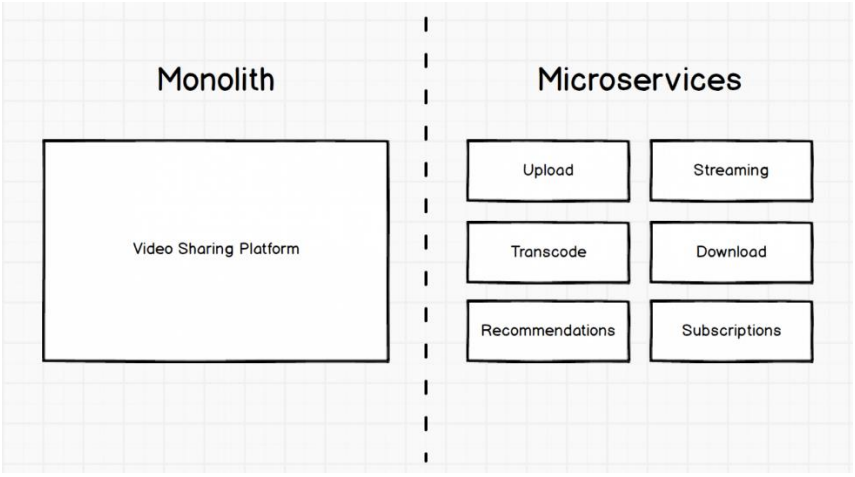
Microservice and Monolithic Architecture

First of all, I want to explain Monolithic Architecture. Understanding Monolithic Architecture helps to understand Microservices more efficently. Monolithic means composed all in one piece, different components combined into a single program from a single platform. Monolithic applications are designed to handle multiple related tasks. For example, think about a monolithic e-commerce application. It should contain a web server, a load balancer, product service, an ordering system and a payment system. Each of these sytems or services can change and that change of a system or service causes to compile and test all system again. Here is a basic example that helps to understand what is monolithic and microservice.



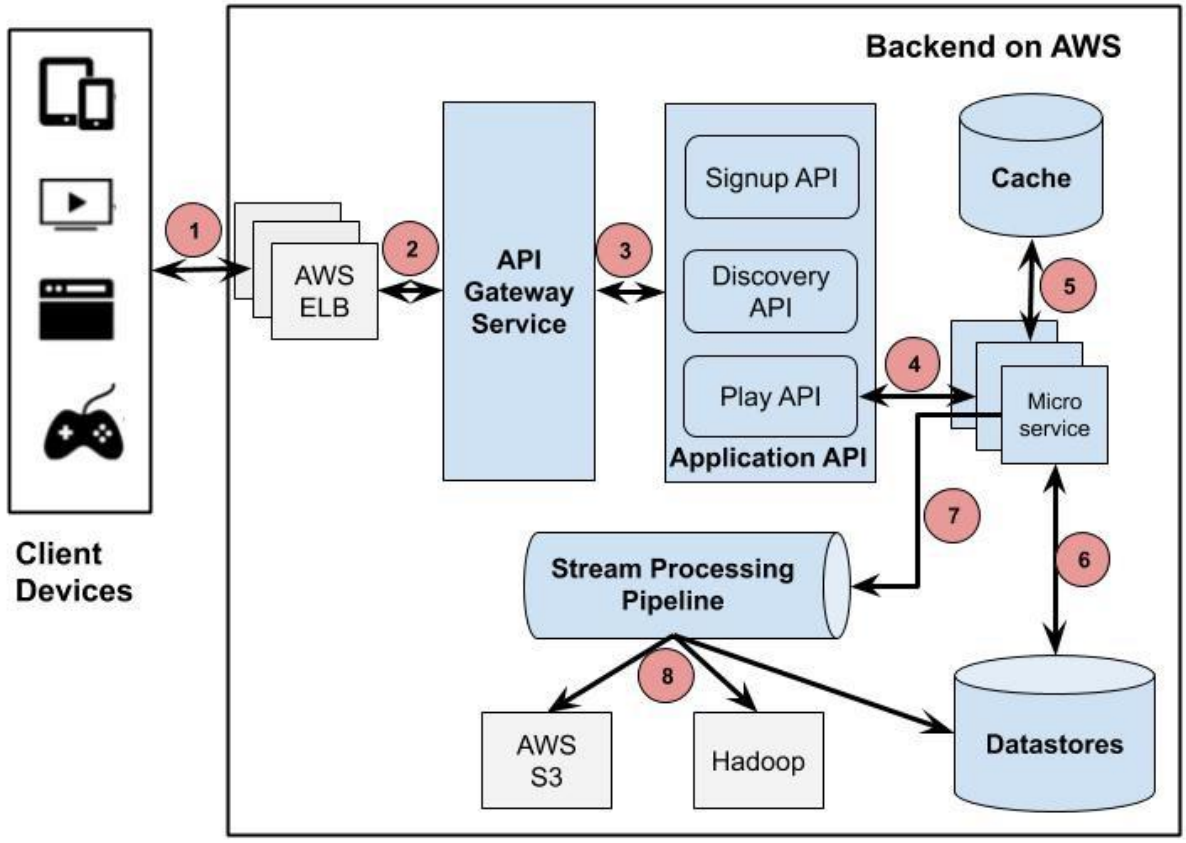


Microservice Architecture is an architectural design that each microservice interact with where it belongs and almost each microservice communicate with each other. They work together but actually that is not a dependency because, updates of any service does not effect another service. Besides, other services hold to run. Microservices are mostly stateless small programs and can call each

other as well. In systems that have intense traffic, they need to be scalable, highly available under high request volume.

From Martin Fowler' s definition, "microservices are a suite of small services, each running in its own process and communicate with lightweight mechanisms." These small services are independently deployable and upgradable.

I want to give an example for understanding easily. The described architectures help us get a general understanding of how different components organize and work together. Let' s look at that architecture:



That design shows us Cloud-based Microservices Architecture at Netflix. From the software architecture point of view, Netflix comprises three main parts: Client, Backend and Content Delivery Network. Client is any supported browsers on a laptop or desktop or a Netflix app on smartphones or smart TVs.

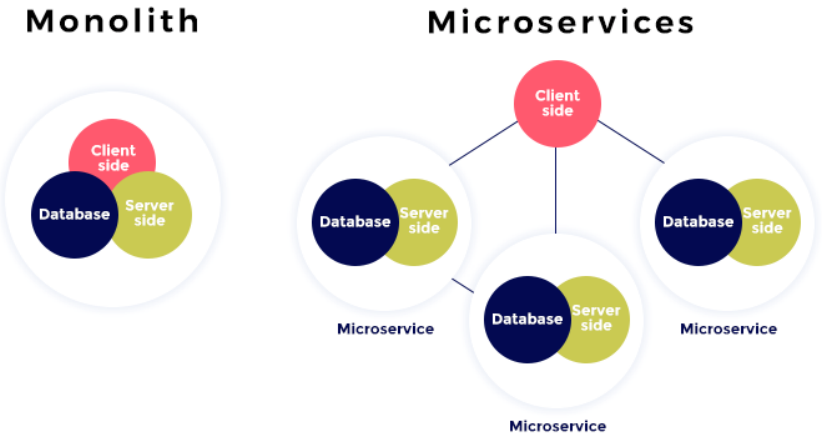
Backend includes services, databases and storages.

According to that architecture, the client sends a Play request to Backend. That request is handled by AWS.

AWS will forward that request to API Gateway Service and forwarded request is handled by Play API.

And this point, Play API connects with microservice that fulfill the request. Microservices can save or get data from a data store during the process.

Monolith vs Microservices



We can use any of these architectures but once, we need to compare them and decide what is more powerful for our software application.

In Monolithic Architecture, when your updates of any service in your applicaton increases and your architecture' s complexity increases, it is hard to manage. In services, they have dependency to each other so one change can effect whole system. Also, it is hard to manage version system in Monolithic Architecture and you need to use same frameworks or programming languages for all of the modules in whole project.

In Microservice Architecture, whatever your application big or small, adding new functionalities or updating the services is easy to manage. Because it is enough to change just that service that changed. Each service are independent and code base is basic. It is easy to manage version control system. There is no need to redeploy for services that did not change and also you can use different programming languages or frameworks together whole project.

By the way, for sure there are disadvantages of Microservice Architecture. You have multiple databases and multiple service so it is hard to manage but possible and if systems are independent needlessly, again it is hard to manage.

Consequently, each of architecture has advantages and disadvantages. It is important to know and decide what your system needs.

Şeyma Tezcan