# Microservices

## Introduction

An architectural style that can separate concerns by service domain, decreasing the coupling in code and increasing the cohesion. Principal difference between this architecture and monolithic architecture is the separation of concerns, older days monolithic architecture was the common use because it was simpler to develop, but the microservice came to enhance the application process time and the smatter use of hardware.

## Principles

Decentralization helps the release of features to be easier to do, once that we do not need to stop our entire application to release a new version, we have the option to make use of Kubernetes to release our changes by steps, per service instance.

Componentization gives the developers the ability to create the services using different technologies, for instance, creating the user service using java and product service using c# and each of these services using different databases, so we can have the best technology for each purpose in our application.

Autonomy enhances the way teams used to work on the application, bringing safety once that the developer can change the code for one service having the security that it will not break any other part of the application, because the code is divided.

Technology diversity is an important thing for microservice, because if the team are working with a big system being developed in java but they need to implement AI and the python is the best programming language for working with this technology, then the team could create a new service using python to supply this needed using the best technology for that purpose.

## Advantages

Scalability allows the developers to create new instances of the service as needed, for example, when the application is having a lot of requests from the users and it is going to use a lot of process, we can configure the Kubernetes to create new instances based on some rules, like if the CPU process is at some point or the quantity of request per some time.

Resilience is one of the principal key of microservice, because as we have our application broken into several pieces, if some of these pieces stop working, we still have the others pieces, so our users still have access to our application and with this technique we can create new instances of the broken service to keep supplying our users, so the service can still be up.

Technological Agility gives the developers the ability to use a new technology without affecting the entire system as we have all services in different code base, if we need to implement a new feature into some of these services, we will not affect another service.

## Challenges

Complexity is the principal challenge of microservice, because when we break our monolithic into many parts, we must implement a lot of technology to try to keep the consistency among the services and our code are spread across all the application making the development documentation hardest to keep updated.

Data integrity is the first concerning when we start working with microservice, because the basic structure of microservice is to use different databases for each service, so it brings us the challenge of keeping these databases with consistency data and for it we have different techniques like using SAGA pattern to let all interested services know when some data is changed or we can use a message broker to notify the interested service about the data change.

Network issues is something that we need to consider when we are thinking about using microservice in our application, because some people think microservice is faster than monolithic but when we consider the network latency specially among the services, we have a slower process than just using a monolithic architecture as we have all the code in the same application and we do not need to use network to communicate between services.

Skill set is a big challenge in microservice architecture, because the company will need to hire more developers and it can increase a lot the money invested in the system, specially if the different services are developed in different technologies, it is harder to find developers with all the required knowledge or developers with the specific knowledge.

## Conclusion

We need to consider microservice as the last option in our stack of architecture styles, because it is expensive for the company and a lot complex for the developers to keep maintainability of the application. Meanwhile, monolithic architecture can afford a higher number of problems, being cheaper and not so complex to maintain.