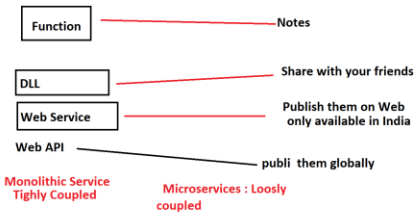
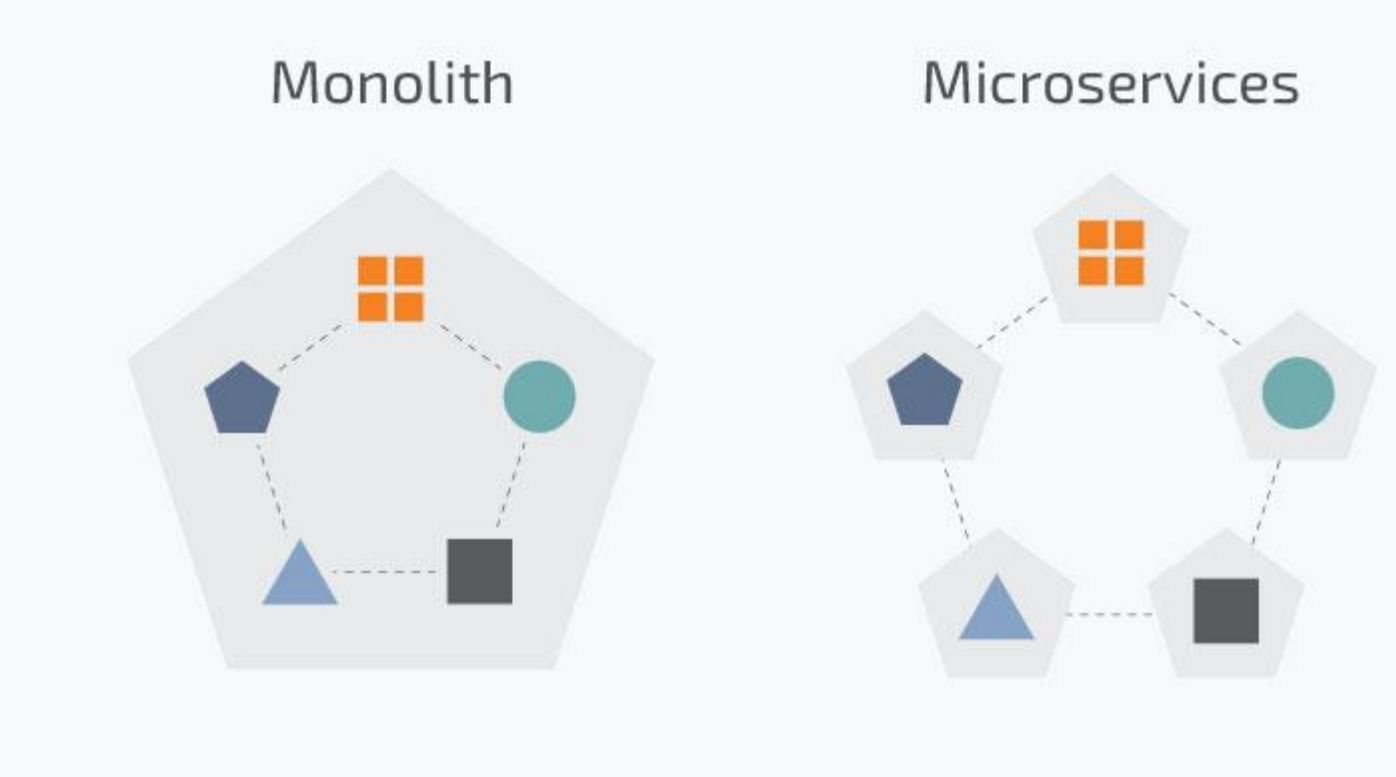
Microservices vs Monolith: which architecture is the best choice for your business?



Having come into light just a few years ago, microservices are an accelerating trend these days. Indeed, microservices approach offers tangible benefits including an increase in scalability, flexibility, agility, and other significant advantages. Netflix, Google, Amazon, and other tech leaders have successfully switched from monolithic architecture to microservices. Meanwhile, many companies consider following this example as the most efficient way for business growth.

On the contrary, the monolithic approach is a default model for creating a software application. Still, its trend is going down because building a monolithic application poses a number of challenges associated with handling a huge code base, adopting a new technology, scaling, deployment, implementing new changes and others.

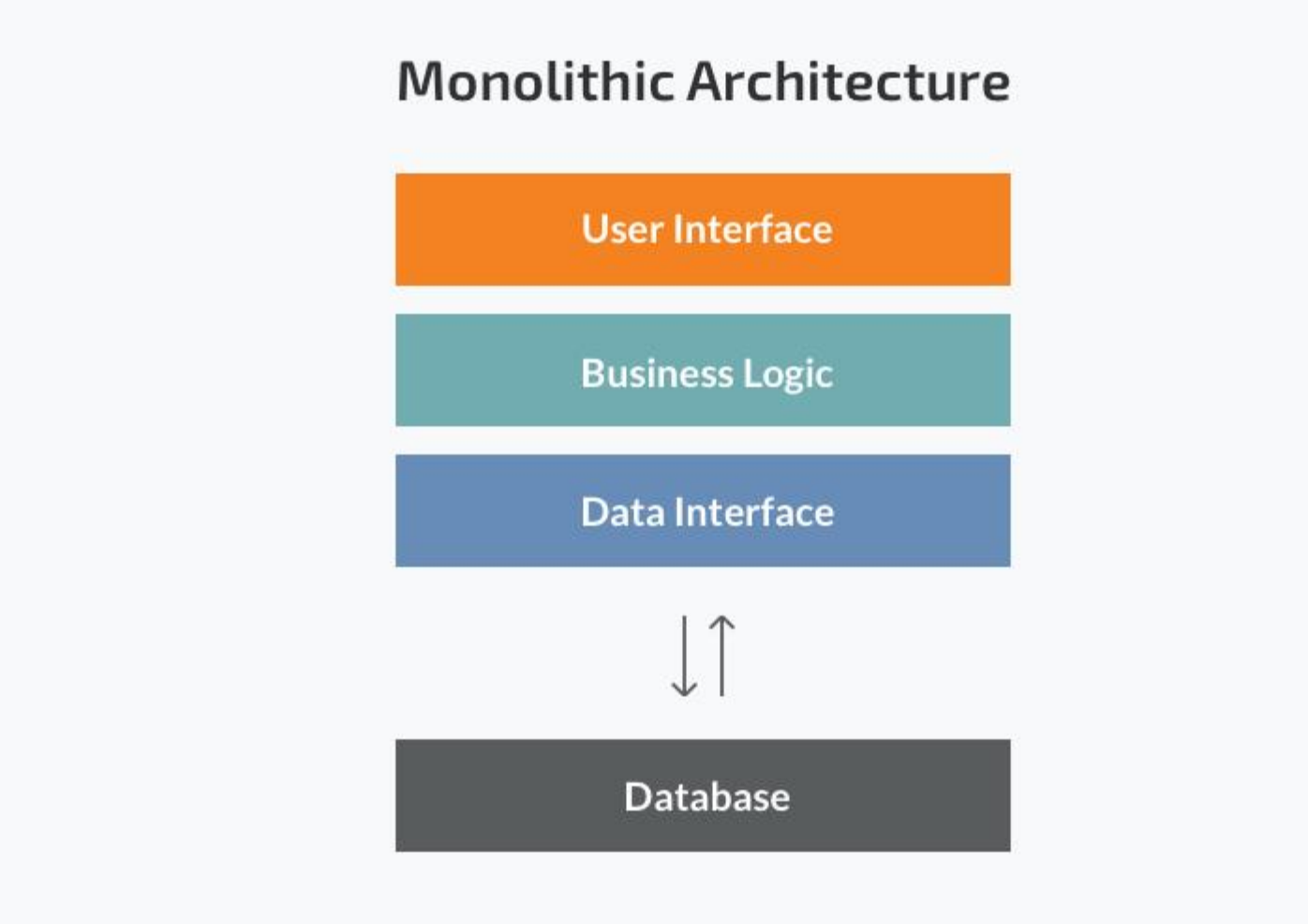


We are going to compare microservices with the monolithic architecture, outline the strengths and weaknesses of both approaches, and find out which software architecture style will be the best fit for your business.

MONOLITHIC ARCHITECTURE

The monolithic architecture is considered to be a traditional way of building applications. A monolithic application is built as a single and indivisible unit. Usually, such a solution comprises a client-side user interface, a server side-application, and a database. It is unified and all the functions are managed and served in one place.

Normally, monolithic applications have one large code base and lack modularity. If developers want to update or change something, they access the same code base. So, they make changes in the whole stack at once.



Strengths of the Monolithic Architecture

- **Less cross-cutting concerns.** Cross-cutting concerns are the concerns that affect the whole application such as logging, handling, caching, and performance monitoring. In a monolithic application, this area of functionality concerns only one application so it is easier to handle it.
- **Easier debugging and testing.** In contrast to the microservices architecture, monolithic applications are much easier to debug and test. Since a monolithic app is a single indivisible unit, you can run end-to-end testing much faster.
- **Simple to deploy. Another advantage associated with the simplicity of monolithic apps is easier deployment. When it comes to monolithic applications, you do not have to handle many deployments – just one file or directory.**
- **Simple to develop. As long as the monolithic approach is a standard way of building applications, any engineering team has the right knowledge and capabilities to develop a monolithic application.**

Weaknesses of the Monolithic Architecture

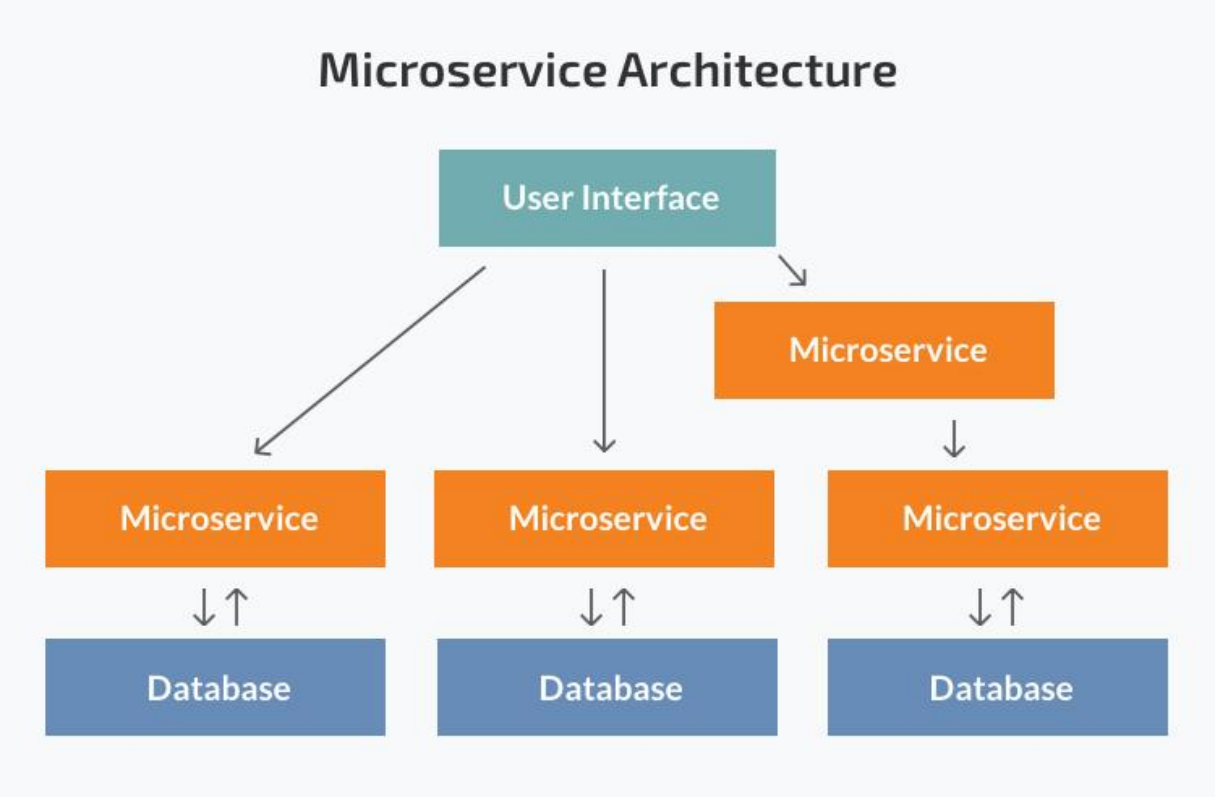
- **Understanding. When a monolithic application scales up, it becomes too complicated to understand. Also, a complex system of code within one application is hard to manage.**
- **Making changes. It is harder to implement changes in such a large and complex application with highly tight coupling. Any code change affects the whole system so it has to be thoroughly coordinated. This makes the overall development process much longer.**
- **Scalability.** You cannot scale components independently, only the whole application.
- **New technology barriers.** It is extremely problematic to apply a new technology in a monolithic application because then the entire application has to be rewritten.

MICROSERVICES ARCHITECTURE

While a monolithic application is a single unified unit, a **microservices architecture** breaks it down into a collection of smaller independent units. These units carry out every application process as a separate service. So all the services have their own logic and the database as well as perform the specific functions.

In short, 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.
Martin Fowler

Within a microservices architecture, the entire functionality is split up into independently deployable modules which communicate with each other through defined methods called APIs (Application Programming Interfaces). Each service covers its own scope and can be updated, deployed, and scaled independently.



Strengths of the Microservice Architecture

- **Independent components.** Firstly, all the services can be deployed and updated independently, which gives more flexibility. Secondly, a bug in one microservice has an impact only on a particular service and does not influence the entire application. Also, it is much easier to add new features to a microservice application than a monolithic one.
- **Easier understanding.** Split up into smaller and simpler components, a microservice application is easier to understand and manage. You just concentrate on a specific service that is related to a business goal you have.
- **Better scalability.** Another advantage of the microservices approach is that each element can be scaled independently. So the entire process is more cost- and time-effective than with monoliths when the whole application has to be scaled even if there is no need in it. In addition, every monolith has limits in terms of scalability, so the more users you acquire, the more problems you have with your monolith. Therefore, many companies, end up rebuilding their monolithic architectures.

- **Flexibility in choosing the technology.** The engineering teams are not limited by the technology chosen from the start. They are free to apply various technologies and frameworks for each microservice.
- **The higher level of agility.** Any fault in a microservices application affects only a particular service and not the whole solution. So all the changes and experiments are implemented with lower risks and fewer errors.

Weaknesses of the Microservice Architecture

- **Extra complexity.** Since a microservices architecture is a distributed system, you have to choose and set up the connections between all the modules and databases. Also, as long as such an application includes independent services, all of them have to be deployed independently.
- **System distribution.** A microservices architecture is a complex system of multiple modules and databases so all the connections have to be handled carefully.
- **Cross-cutting concerns.** When creating a microservices application, you will have to deal with a number of cross-cutting concerns. They include externalized configuration, logging, metrics, health checks, and others.
- **Testing.** A multitude of independently deployable components makes testing a microservices-based solution much harder.

SO WHICH SOFTWARE ARCHITECTURE SUITS YOUR SOLUTION AND YOUR BUSINESS BEST?

Choosing a monolithic architecture

- **Small team.** If you are a startup and your team is small, you may not need to deal with the complexity of the microservices architecture. A monolith can meet all your business needs so there is no emergency to follow the hype and start with microservices.
- **A simple application.** Small applications which do not demand much business logic, superior scalability, and flexibility work better with monolithic architectures.
- **No microservices expertise. Microservices require profound expertise to work well and bring business value. If you want to start a microservices application from scratch with no technical expertise in it, most probably, it will not pay off.**
- **Quick launch. If you want to develop your application and launch it as soon as possible, a monolithic model is the best choice. It works well when you aim to spend less initially and validate your business idea.**

Choosing a microservices architecture

- **Microservices expertise. Without proper skills and knowledge, building a microservice application is extremely risky. Still, just having the architecture knowledge is not enough. You need to have DevOps and Containers experts since the concepts are tightly coupled with microservices. Also, domain modelling expertise is a must. Dealing with microservices means splitting the system into separate functionalities and dividing responsibilities.**
- **A complex and scalable application. The microservices architecture will make scaling and adding new capabilities to your application much easier. So if you plan to develop**

a large application with multiple modules and user journeys, a microservice pattern would be the best way to handle it.

- **Enough engineering skills.** Since a microservice project comprises multiple teams responsible for multiple services, you need to have enough resources to handle all the processes.

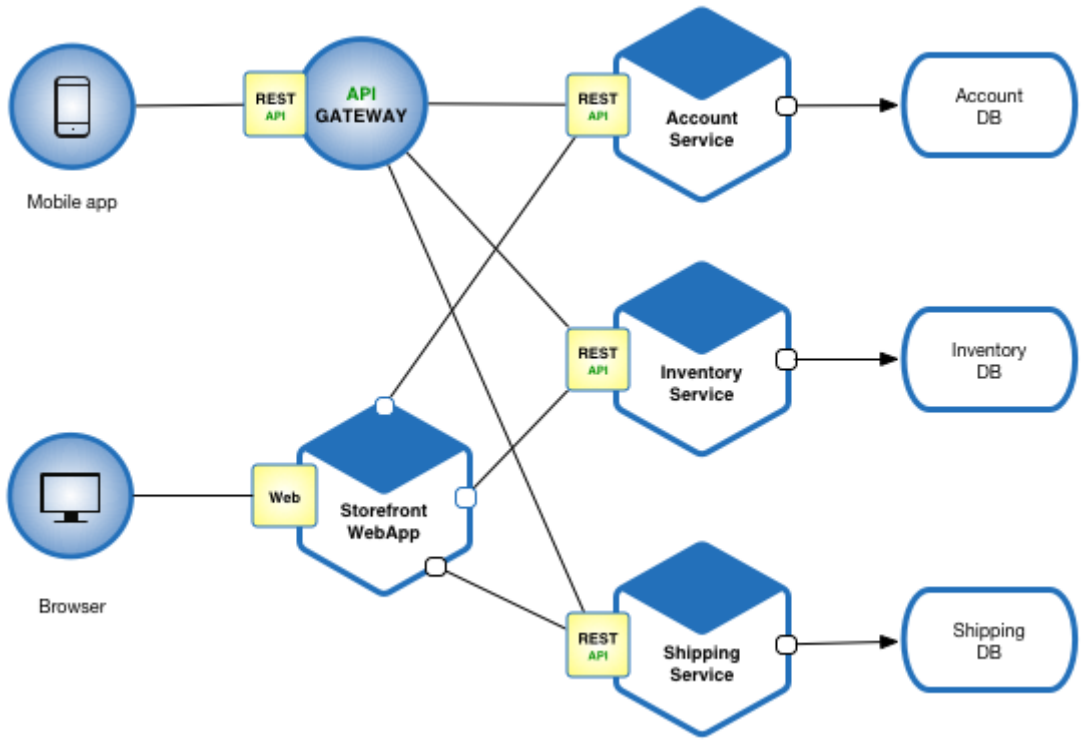
What are Microservices?

Microservices are a design pattern in which applications are composed of small, independent modules that communicate with each other using well-defined contracts. Each microservice focuses on a single concept.

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.



“**Microservices** are a method of developing software applications which are made up of independently deployable, modular services. Each **microservice** runs a unique process and communicates through a well-defined, lightweight mechanism, such as a container, to serve a business goal.”

With **Docker**, you can make your application independent of the host environment. Since you have **microservices** architecture, you can now encapsulate each of them in **Docker** containers. **Docker** containers are lightweight, resource isolated environments through which you can build, maintain, ship and deploy your application.

https://searchapparchitecture.techtarget.com/feature/5-core-components-of-microservices-architecture

Why Microservices?

Microservices make it easier to develop, test, and deploy isolated parts of your application. Once deployed, each microservice can be independently scaled as needed.

Microservices and containers

Containers combine an app plus its configuration and dependencies into a single, independently deployable unit. Containers are an excellent fit for bundling and deploying independent microservices.