What Is Microservices – Introduction To Microservice Architecture

Have you ever wondered, ***What is Microservices*** *and how the scaling industries integrate with them while building applications to keep up with their client expectations?*

To get an idea, you have to understand how a monolithic application is decomposed into small tiny micro applications which are packaged and deployed independently.

Why are Microservices used?

Microservices were used to overcome the challenges of monolithic architecture that prevailed initially in the market and also enables you to deploy independent services.

So, before we move on to what is Microservices in simple words, let's see the architecture that prevailed initially in the market i.e. the **Monolithic Architecture.**

In layman terms, you can say that its similar to a big container wherein all the software components of an application are assembled together and tightly packaged.

Listed below are the challenges of Monolithic Architecture:

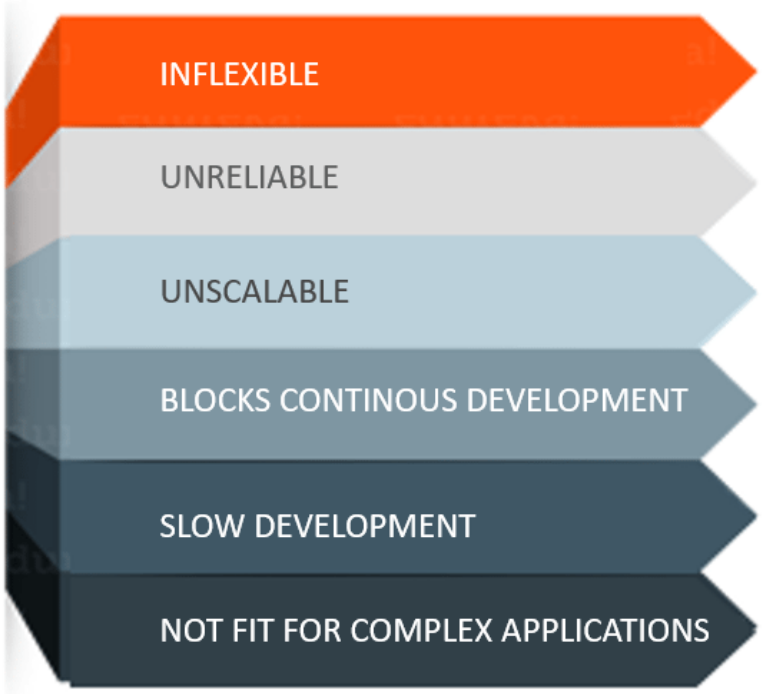

Figure 1: Challenges of Monolithic Architecture

- **Inflexible –** Monolithic applications cannot be built using different technologies
- **Unreliable –** Even if one feature of the system does not work, then the entire system does not work
- **Unscalable –** Applications cannot be scaled easily since each time the application needs to be updated, the complete system has to be rebuilt
- **Blocks Continous Development –** Many features of the applications cannot be built and deployed at the same time
- **Slow Development –** Development in monolithic applications take lot of time to be built since each and every feature has to be built one after the other
- **Not Fit For Complex Applications –** Features of complex applications have tightly coupled dependencies

The above challenges were the main reasons that led to the evolution of microservices. So, now let us understand what is Microservices in simple words?

What exactly is Microservices?

Microservices is an architectural style that structures an application as a collection of small autonomous services, modeled around a **business domain.**

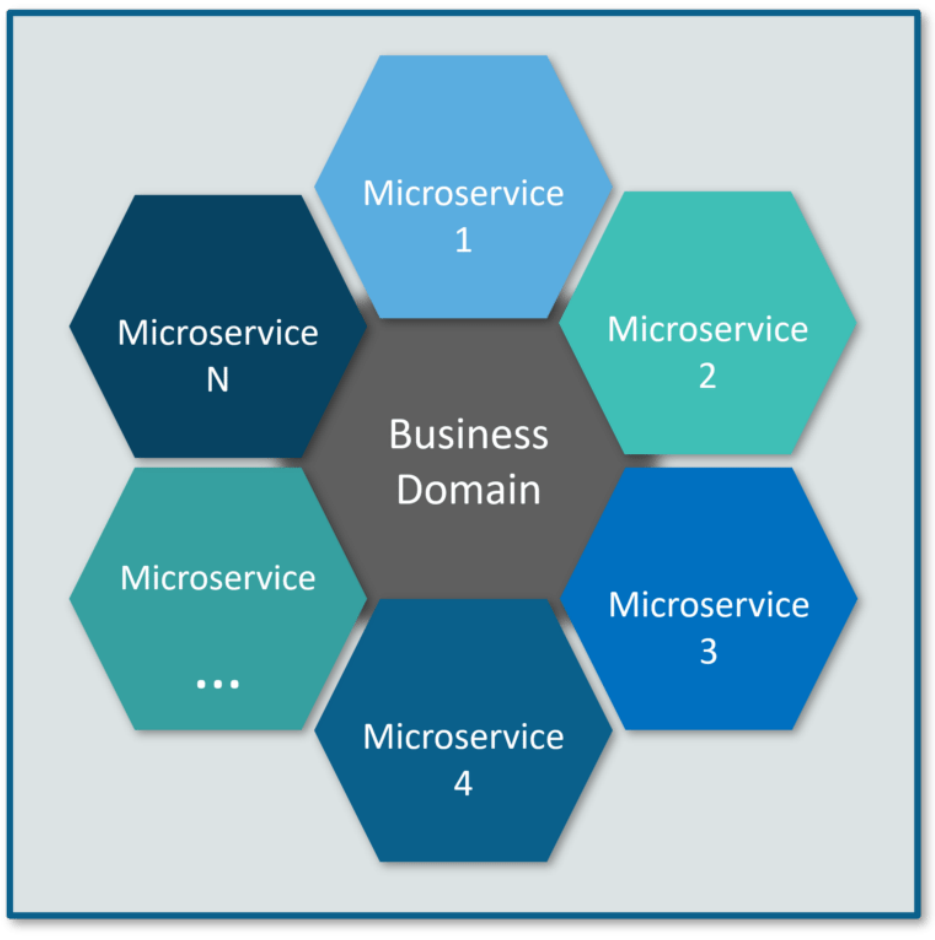

Figure 2: Microservices Representation

In the given Architecture, each service is **self-contained** and implements a **single business capability.**

Differences Between Traditional Architecture and Microservices

Consider an E-commerce application as a use-case to understand the difference between both of them.

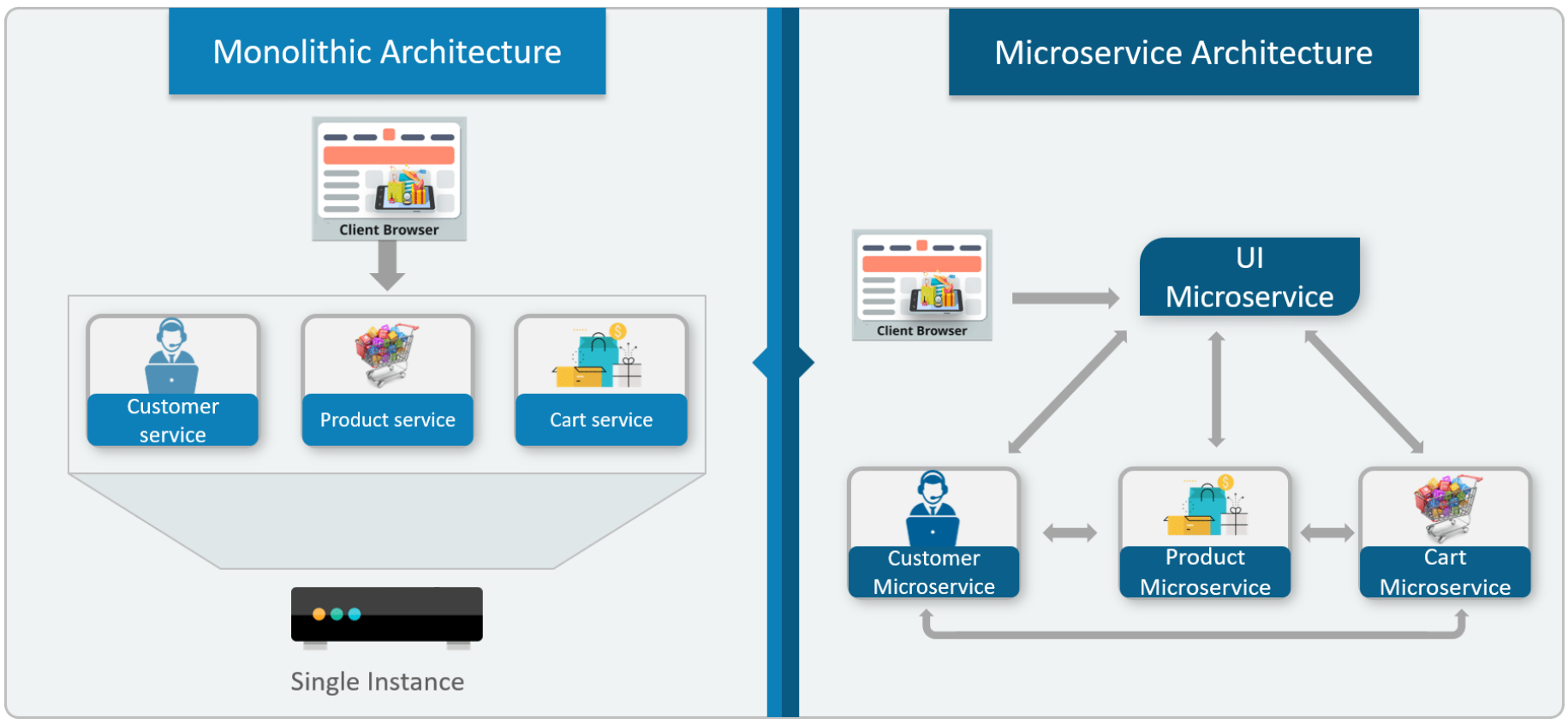

Figure 3: Differences Between Monolithic Architecture and Microservices

The main difference we observe in the above diagram is that all the features initially were under a single instance sharing a single database. But then, with microservices, each feature was allotted a different microservice, handling their own data, and performing different functionalities.

Now, let us understand more about its architecture. Refer to the diagram below:

Architecture of Microservices

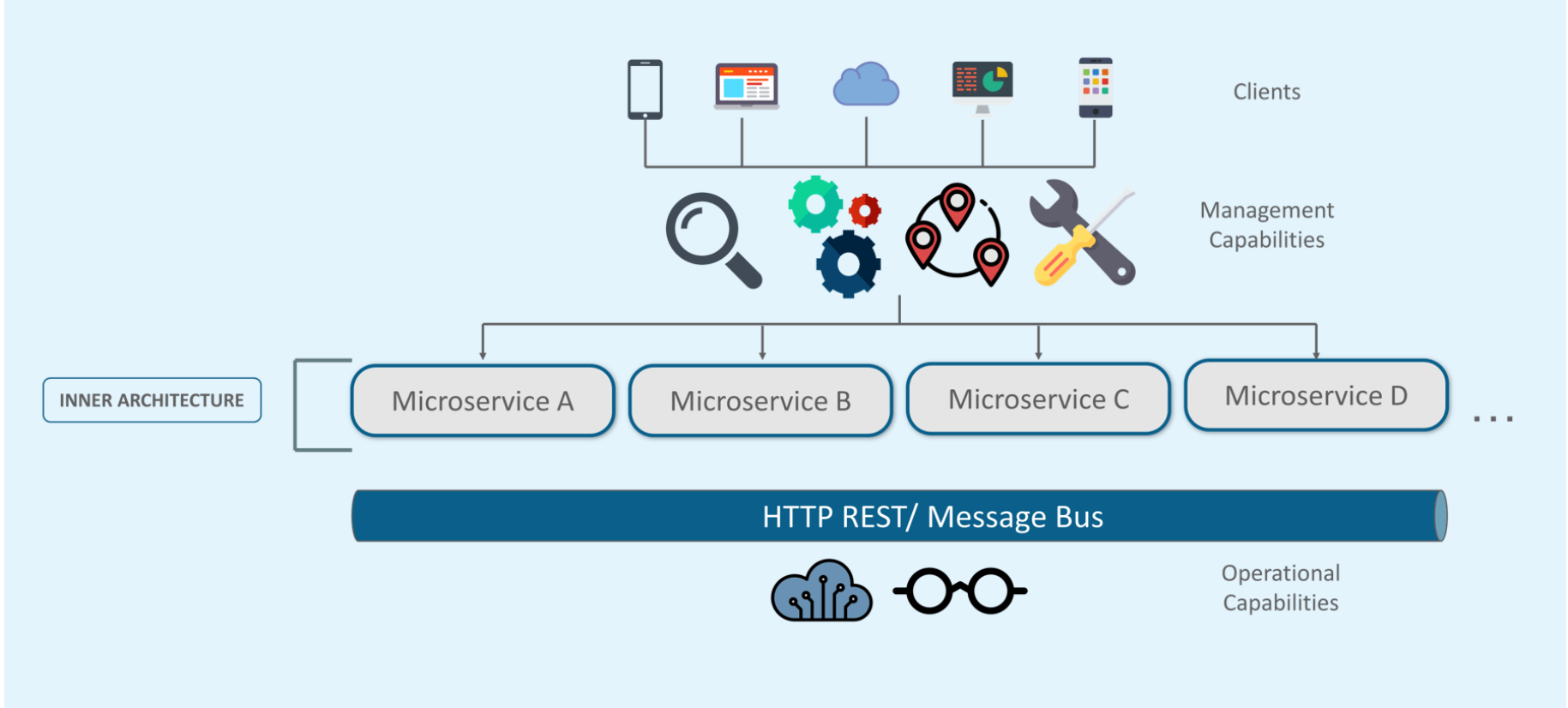

Figure 4: Architecture

- Different clients from different devices try to use different services like search, build, configure and other management capabilities
- All the services are separated based on their domains and functionalities and are further allotted to individual microservices
- These microservices have their own **load balancer** and **execution environment** to execute their functionalities & at the same time captures data in their own databases
- All the microservices communicate with each other through a stateless server which is either **REST** or **Message Bus**
- Microservices know their path of communication with the help of **Service Discovery** and perform operational capabilities such as automation, monitoring
- Then all the functionalities performed by microservices are communicated to clients via **API Gateway**
- All the internal points are connected from the API Gateway. So, anybody who connects to the API Gateway automatically gets connected to the complete system

Now, let us learn more about microservices by looking at its features.

Microservices Features

Figure 5: Features

- **Decoupling** – Services within a system are largely decoupled. So the application as a whole can be easily built, altered, and scaled
- **Componentization** – Microservices are treated as independent components that can be easily replaced and upgraded
- **Business Capabilities** – Microservices are very simple and focus on a single capability
- **Autonomy** – Developers and teams can work independently of each other, thus increasing speed
- **Continous Delivery** – Allows frequent releases of software, through systematic automation of software creation, testing, and approval
- **Responsibility** – Microservices do not focus on applications as projects. Instead, they treat applications as products for which they are responsible

- **Decentralized Governance** – The focus is on using the right tool for the right job. That means there is no standardized pattern or any technology pattern. Developers have the freedom to choose the best useful tools to solve their problems
- **Agility** – Any new feature can be quickly developed and discarded again

Advantages Of Microservices

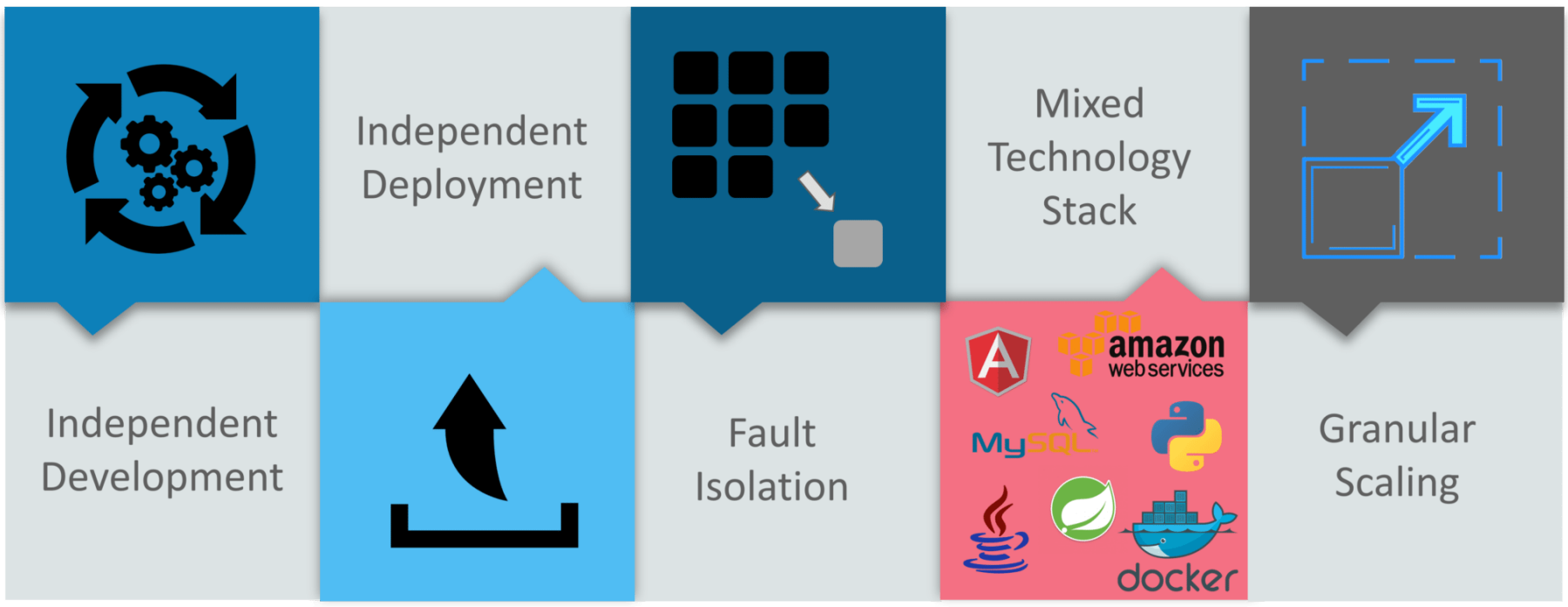

Figure 6: Advantages

- **Independent Development** – All microservices can be easily developed based on their individual functionality
- **Independent Deployment** – Based on their services, they can be individually deployed in any application
- **Fault Isolation** – Even if one service of the application does not work, the system still continues to function
- **Mixed Technology Stack** – Different languages and technologies can be used to build different services of the same application
- **Granular Scaling** – Individual components can scale as per need, there is no need to scale all components together

What are the Best Practices to Design Microservices?

In today's world, complexity has managed to creep into products. Microservice architecture promises to keep teams scaling and function better.

The following are the best practices to design them:

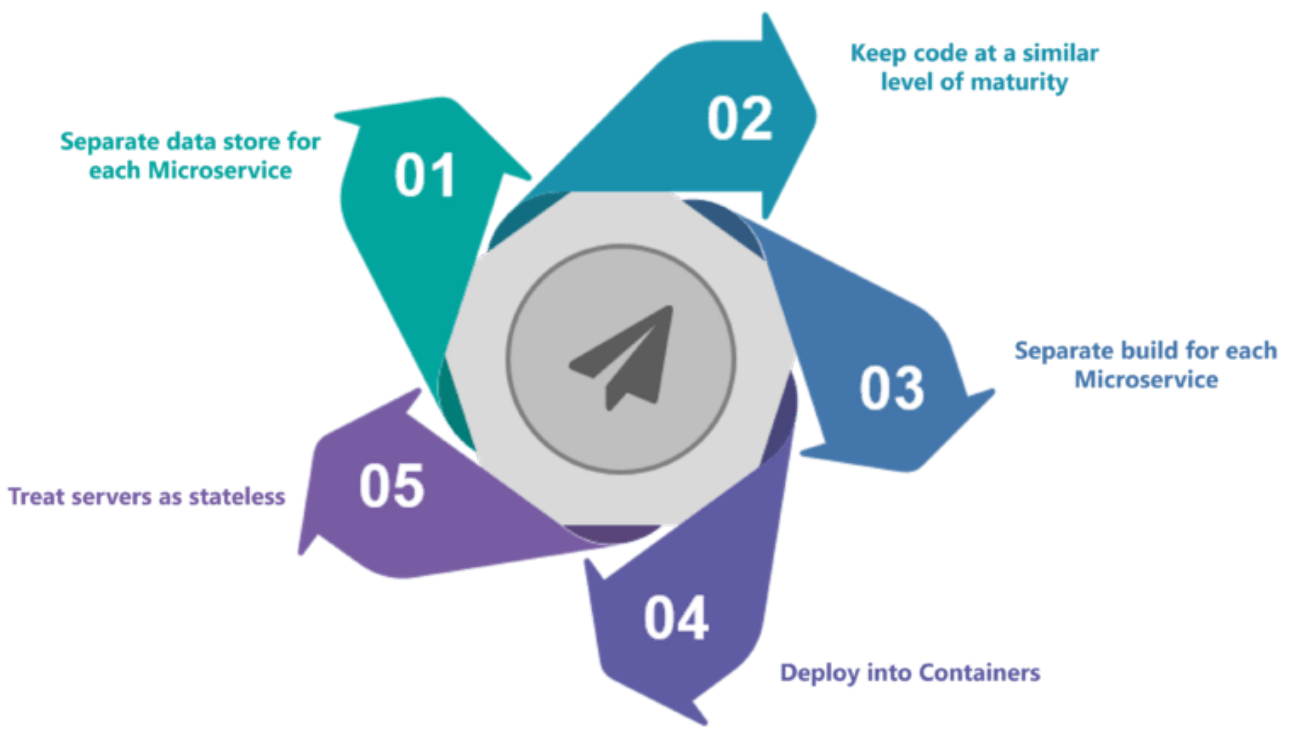

Figure 7: Best Practices

Microservice Examples: Shopping Cart Application

Let's take a classic use case of a shopping cart application.

When you open a shopping cart application, all you see is just a website. But, behind the scenes, the shopping cart application has a service for accepting payments, a service for customer services and so on.

Assume that developers of this application have created it in a monolithic framework. Refer to the diagram below:

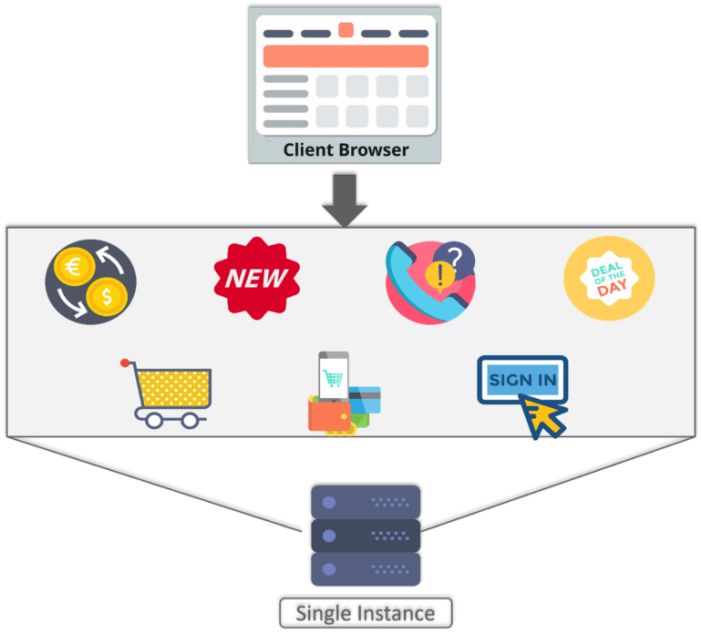

Figure 8: Monolithic Framework Of Shopping Cart Application

So, all the features are put together in a single code base and are under a single underlying database.

Now, let's suppose that there is a new brand coming up in the market and developers want to put all the details of the upcoming brand in this application.

Then, they not only have to rework on the service for new labels, but they also have to reframe the complete system and deploy it accordingly.

To avoid such challenges developers of this application decided to shift their application from a monolithic architecture to a newer architecture. Refer to the diagram below to understand the architecture of shopping cart application

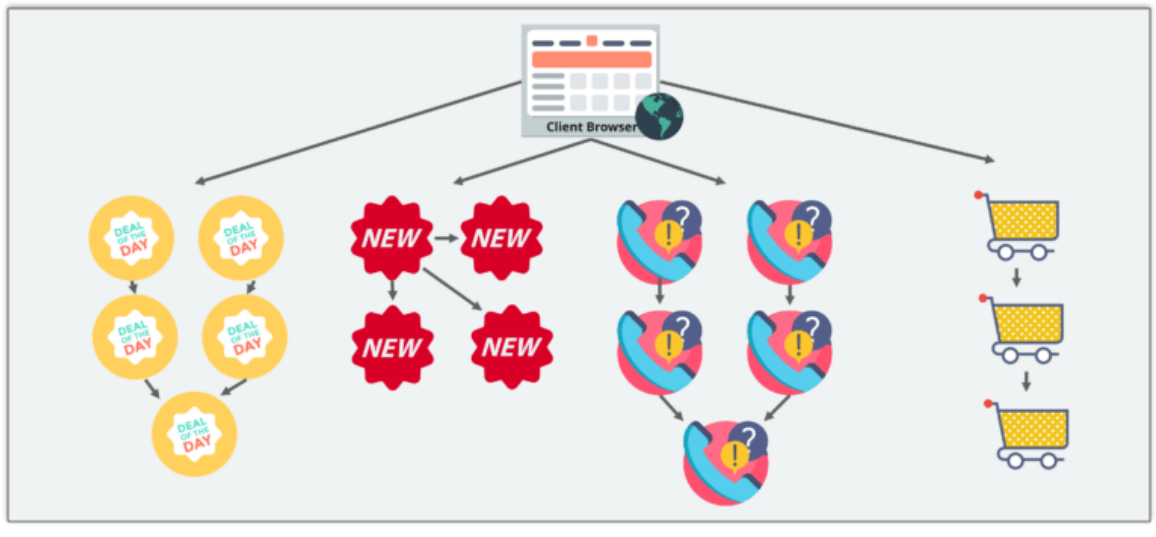

Figure 9: Architecture Of Shopping Cart Application

This means that developers don't create a web microservice, a logic microservice, or a database microservice. Instead, they create separate microservices for search, recommendations, customer services and so on.

This type of architecture for the application not only helps the developers to overcome all the challenges faced with the previous architecture but also helps the shopping cart application to be built, deployed, and scale up easily.

Companies using Microservices

There is a long list of companies using Microservices to build applications, these are just to name a few: