Microservices architecture is a software development approach that structures an application as a collection of small, independent, and loosely coupled services.

* Each service is focused on a specific business capability and can be developed, deployed, and scaled independently of other services.
* In a microservices architecture, each service typically runs in its own process and communicates with other services through a lightweight protocol such as HTTP or messaging.
* Microservices architecture can be a powerful approach to building complex software systems, but it requires careful planning, design, and management to be successful.
* Uses http protocol.
* - Microservices are language, platform, and data base independent.
* - They are loosely coupled and independently deployed.
* - Communicate with each other over the combination of rest apis, event streaming and message broker.
* - Microservices are design by dealing failures and try to manage errors with a proper action.
* - Any new feature can be easily added without touching the entire application as it is light weight and independent.
* - They are scalable independently (whenever we want to scale the application, it will be costly and take a lot of time. So microservices solve this issue).
* - Highly maintainable and testable as it is loosely coupled.
* - Develop and deploy rapidly
* - It is based on agile development
* - Help us to manage the network and latency problem
* - High data integrity

**Key components of Microservice architecture**

Five key components make up microservices architecture. They include the following:

* Microservices
* Containers
* Service mesh
* Service discovery- A service registry that helps services locate and communicate with each other.
* API gateway- An entry point for client requests that routes requests to the appropriate [microservices](https://microsoft.optisolbusiness.com/), performs authentication and authorization, and handles other tasks such as caching and request-response mapping.

#### **Microservice vs Service-Oriented architecture**

* While microservices use loose connections and APIs, SOA uses protocols to create an application and create functionality and compatibility for services.
* This system lacks when it comes to operating within the cloud and cloud computing processes. It can't compete due to latency and lack of scalable service.

**Monolithic applications**   
If all the functionalities of a project exist in a single codebase, then that application is known as a monolithic application.

We design our application in various layers like presentation, service, and persistence and then deploy that codebase as a single jar/war file. This is nothing but a monolithic application, where **“mono”**represents the single codebase containing all the required functionalities.

**Disadvantages of Monolithic applications:**

* It becomes too large with time and hence, difficult to manage.
* We need to redeploy the whole application, even for a small change.

**Advantages :**

1. Simplicity: Monolithic applications are relatively easy to develop, deploy, and maintain compared to distributed or microservices architectures, as there is only one codebase to manage.
2. Debugging: Debugging monolithic applications is easier than distributed or microservices architectures, as it is easier to locate and isolate issues within a single codebase.

**principles of micro services (five principles) –**

1.Single responsibility 2. Modelled around business domain 3. Isolated Failure 4. Infrastructure Automation 5. Deploy independently

# **Components of Microservices**

There are the following components of microservices:

* Spring Cloud Config Server
* Netflix Eureka Naming Server
* Hystrix Server
* Netflix ZuulAPI Gateway Server
* Netflix Ribbon
* Zipkin Distributed Tracing Server

# **Creating a Simple Microservice**

**Step 1**: Create a Maven project using Spring Initializr <https://start.spring.io/>

**Step 2**: Choose the Spring Boot version **2.2.0 M6** or higher version. Do not choose the snapshot version.

**Step 3**: Provide the **Group** name. In our case c**om.customer**

**Step 4**: Provide the **Artifact id**. We have provided **customer-service**.

**Step 5**: Add the following dependencies: **Spring Web, Spring Boot DevTools, Spring Boot Actuator, Config Client**.

**Step 6**: Click **on Generate the project** button. A **zip** file will download, extract it into the hard disk.

**Step 7**: Now, open the **eclipse**. Import the created maven project. It takes some time to download the required files.