Microservices Development

=> Software Applications we can develop in 2 ways

1) Monolithic Architecture

2) Micro services Architecture

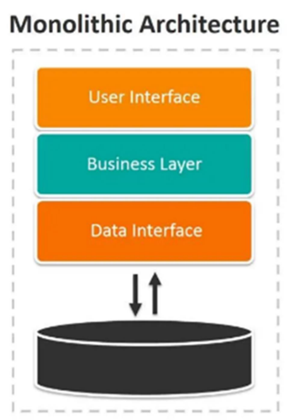
What is Monolithic Architecture

=> If we develop all the functionalities in single application then it is called as Monolithic Application.

1) Presentation Layer

2) Business Layer

3) Data Access Layer



Drawbacks

1) Tightly coupling

2) Maintenence is difficult

3) Re-Deploy entire application

4) Technology Dependent

5) Burden on Server (Server Crash)

6) Single Point of failure

=> To overcome problems of Monolith Architecture, we will use Micro services Architecture.

Microservices Introduction

=> It is not a technology

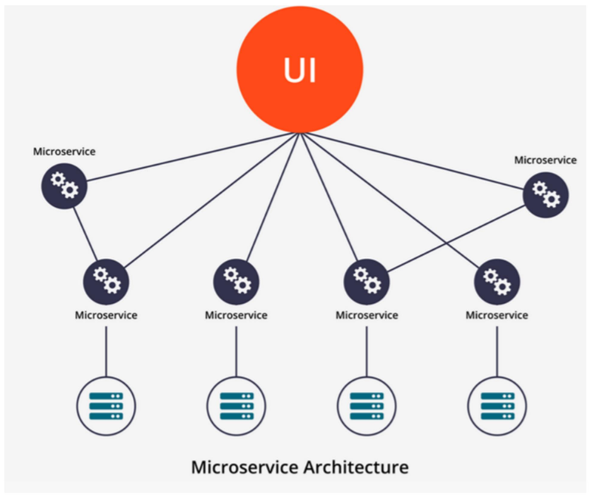
=> It is not programming language

=> It is not a framework

=> It is not an API

=> It is an architectural design pattern

=> It is universal pattern and anyone can use this architecture to develop applications using any technology.



Advantages

1) Loosely Coupled

2) Burden Reduced on Server

3) Easy Maintence

4) No Single point of failure

5) Technology Independent

6) Quick Deliveries

Challenges with Microservices

1) Bounded Context

2) Repeated configurations

3) Visibility

=> Bounded context means identifying how many micro services we need to develop for one application and deciding which functionality we need to add in which micro service.

=> In Several micro services we need to write same configurations like data source, smtp, kafka, redis etc.

=> In micro service architecture we might not get chance to work with all apis in the application.

Microservices Architecture

=> There is no fixed architecture for micro services development.

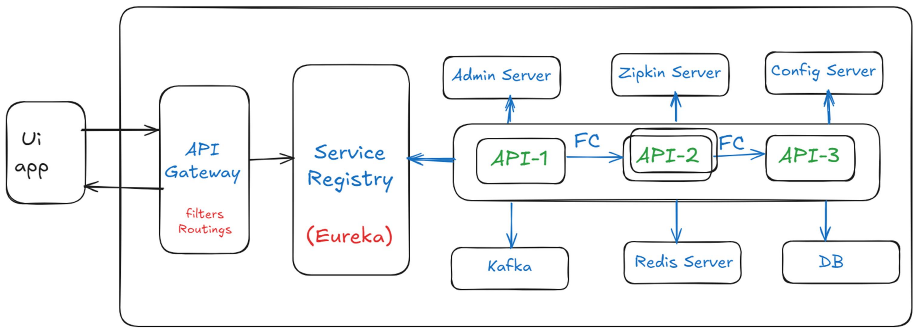
=> We can customize micro services architecture according to our project requirement.

=> As part of Micro services architecture we are going to use below components.

(all the below components are not mandatory, we can use them based on demand)

1) Service Registry (Eureka)	**5) Kafka Server**
2) Admin Server	**6) Redis Server**
3) Zipkin Server	**7) FeignClient**
4) Config Server	**8) API Gateway**

Microservices Architecture



Service Registry

=> Service Registry is used to maintain all apis information like name, status, url and health at once place.

=> It is also called as Service Discovery.

=> We can use Eureka Server as service registry.

=> It will provide user interface to get apis info.

Admin Server

=> It is used to monitor and manage all the apis at one place.

=> It provides beautiful user interface to access all apis actuator endpoints at one place.

Zipkin Server

=> It is used for distributed tracing of our requests

=> It provides beautiful user interface to access apis execution details.

Config Server

=> It is used to separate application code and application properties.

=> It is used to externalize config props of our application.

=> It makes our application loosely coupled with properties file or yml file.

Feign Client

=> It is used for interservice communication

=> If one api communicate with another api with in the same application then it is called as Inter service communication.

Kafka Server

=> It is used as message broker

=> Distributed streaming platform

=> It works based on pub-sub model

Redis Server

=> Redis is a cache server

=> Redis represents data in key-value format

=> Redis is used to reduce no.of db calls

API Gateway

=> It acts as Entry point for all backend apis

=> It acts mediator between frontend app and backend apis.

=> In API Gateway we will write filters + Routings

Filter: We can perform pre-process & post-process

Routings: To forward request to particular backend-api.