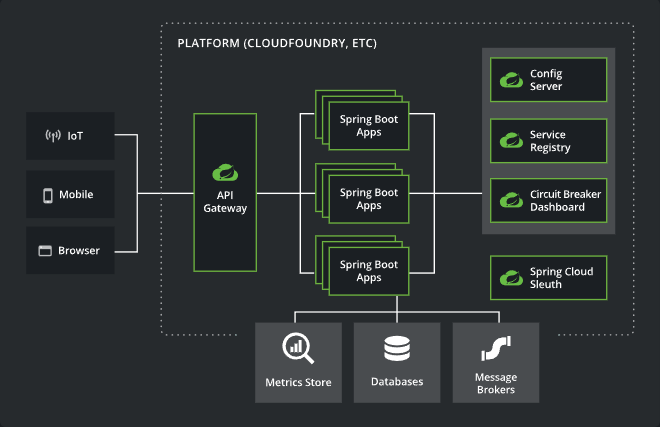
**Microservice**

1. Set of loosely coupled multiple services which will inter communicate with each other.
2. Highly maintainable and tastable
3. Independent deployable



**Service Registry**

****

**@LoadBalanced**

* When more than one instances is present in service then it will distribute this load.
* It helps to enable us to call directly name of the services rather than host of the services.

**Feign Client**

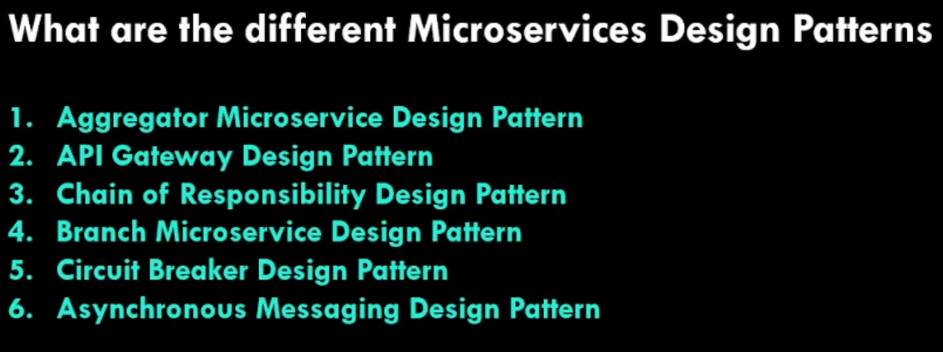
* It is Declarative HTTP web Client developed by Netflix.
* It fallows Declarative Approach
* Now Declarative approach means we need to declare the things in Advance, we have to declare interface and inside that we need to add Annotations in that then it will call methods automatically.

1. First, we need to add Dependency in our microservice project – **OpenFeign**
2. Now we can Create a package name externalservices and in that we create Interface.
3. In this interface Add the annotation on Class level **@FeignClient(name = “NAME-SERVICE”).**
4. After that we need to create a method in Interface Same as we Controller.
5. Give the respective annotation on method.
6. Now call those method where we want call RestApis in the service class.

**API Gateway**

**Config Server**

Add Config Server dependency



Circuit Breakers

Spring Cloud Config Server

Open Feign

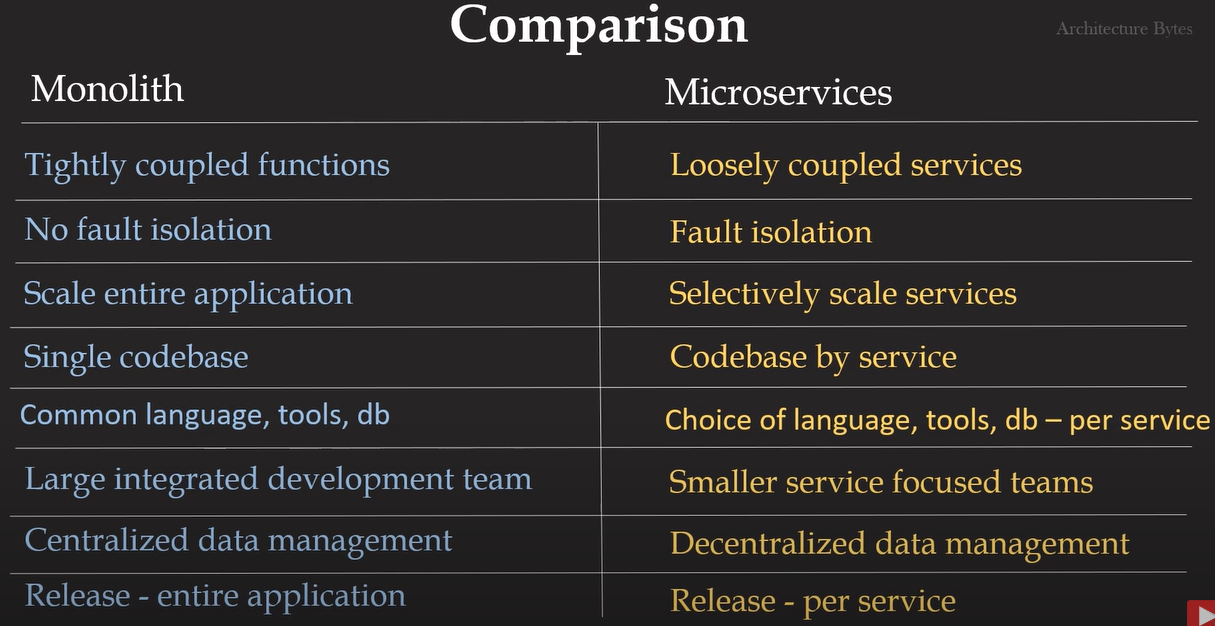
Eureka Discovery Server

Actuator

API gateway Server

Zipkin Distributed Tracing

Git Repository



Microservice Project Example.

1. Create User Service Spring Boot Project

* Add Dependancy Spring Web, MySql Driver, Spring Data JPA, Lombok
* Write all business logics and make API

1. Create Hotel Service Spring Boot Project

* Add Dependancy Spring Web, MySql Driver, Spring Data JPA, Lombok
* Write all business logics and make API

1. Create Rating Service Spring Boot Project

* Add Dependancy Spring Web, MySql Driver, Spring Data JPA, Lombok
* Write all business logics and make API

1. Create Service Registry / Discovery
2. Create Eureka Server Spring Boot Project

* Add dependency Cloud Bootstrap, Eureka Server
* Add annotation @EnableEurekaServer
* Add properties for server setup

1. Create Eureka Client in All services projects

* Add Cloud Bootstrap, Eureka Discovery Client dependency in User, Hotel and Rating services
* Add annotation @EnableEurekaClient in main class
* Add configuration properties in properties file
* Add spring.application.name=”specific application service name”

1. Now we have to set host and port dynamically

* We should use @LoadBalanced annotation on rest templet bean
* @LoadBalanced annotation will create Rest template which will call url with registered services name
* This will help to distribute load if there more than on instances is present
* Now we will call other API with Eureka client registred service Application name
* For example <http://APPLICATION-SERVICE-NAME/getUsers>

1. We can also use Feign Client

**Feign Client**

* It is Declarative HTTP web Client developed by Netflix.
* It fallows Declarative Approach
* Now Declarative approach means we need to declare the things in Advance, we have to declare interface and inside that we need to add Annotations in that then it will call methods automatically.

1. First, we need to add Dependency in our microservice project – **OpenFeign**
2. Now we can Create a package name externalservices and in that we create Interface.
3. In this interface Add the annotation on Class level **@FeignClient(name = “NAME-SERVICE”).**
4. After that we need to create a method in Interface Same as we Controller.
5. Give the respective annotation on method.
6. Now call those method where we want call RestApis in the service class.
7. Create Gateway Spring boot Project

* Add dependency spring-cloud-starter, spring-cloud-starter-gateway, Lombok, spring-boot-starter-webflux, Eureka Discovery Client
* Add properties in properties file to register gateway application in Eureka server
* Add properties in properties file which route for all services to call URL’s through this gateway

1. Create Config Server Spring Project

* Add Dependency Config Server, Eureka Discovery Client
* Add @EnableConfigServer Annotation in main class
* Add properties in properties file “spring.cloud.config.server.git.uri:=git repo URL having config files”

1. Fetching Configuration data from Config Server

* Add spring-cloud-starter-config dependency in all services
* Add property in properties file spring.config.import.optional.configserver=url of config server

1. Now implement Circuit breaker

* Add Spring Actuator dependency for Health monitoring of spring application and spring-boot-starter-aop and resilience4j-spring-boot2 in those services which is depend on another services
* Create a method which will call by fallback for circuit breaker
* Add @CircuitBreaker(name=”write any name which is related to call other services”, fallbackMethod=”method name which will call while fall back for circuit breaker”) annotation on those particular methods in controller class which will call another Api from this method
* Add properties in properties file management.health.circuitbreakers.enabled=true
* management.endpoints.web.exposure.include=health
* management.endpoints.health.show-details=always