Microservice

# Why we use Microservice

Suppose we have monolithic application and contains 3 modules let’s say chatms,userms,orderms

here our all three modules will be dependent on each other by compile time dependency that means suppose we want to release the new version of every module and our chatms is ready to release but others are not ready to release then we can’t release chatms separately because it has compile time dependency on other modules , similary if want to use different languages, databases, packages version etc for different modules but we can’t do in monolithic application

So now in microservice application every module will be complete application and will dependent on each other by runtime and we can compile independently and release and we can use different languages, databases, packages version etc for different modules

they all having runtime dependency so each module can use other module by http request

**Payment**

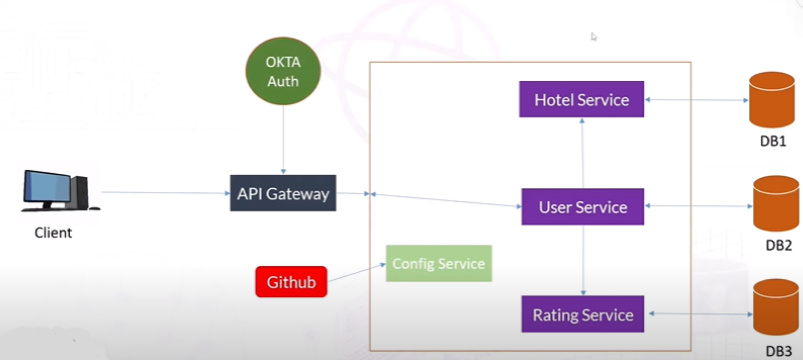
### Product

### Orders

## Users

communicate Using Rest APIs

Microservices



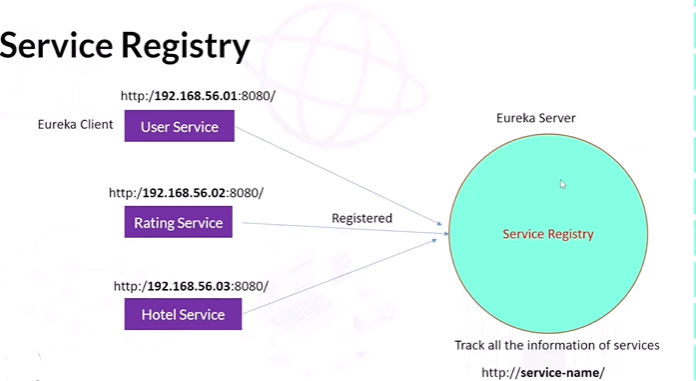
Request will come to api gateway then it will call different services, we will use OKTA auth for authentication and if the services have some common configurations then we will keep that configurations in config server for that we will use github then we will call them in different services

Service Registry

Suppose if our one service is running on one Ip, port and calling another service using http which obviously running on different Ip, port or different machine but due to some technical issue port or any other things like ip, path are not working so our one service can’t use other service because two services are calling each other on the basis of physical properties

to get rid of it we will use Eureka server which obviously is one service and we will register our different services on the eureka server by name so now our different services are not dependent on each other by physical things like ip, port, now one service can call other service by http request using name

Eureka server/Discovery server will keep track of each registered service , we can easily see which services are up or down or the complete information of the services



#### Implementing Service Registry using Eureka SERVER

As our eureka server/discovery server is also service so we will create spring boot project

in the pom.xml file we will add two dependency

**Cloud Bootstrap SPRING CLOUD**

**and**

**Eureka Server SPRING CLOUD DISCOVERY**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

Now we for enabling the Eureka/Discovery server we will go inside main class and top of that we will mention @EnableEurekaServer

And now will go inside application.properties file or application.yml file and add some configuration to make this spring project as Eureka server instead of normal spring boot service

server.port=9000(any port you can mention)  
  
#configure discovery server  
eureka.instance.hostname=localhost  
eureka.client.registerWithEureka=false  
eureka.client.fetchRegistry=false

Last two properties we are adding because we want to prevent this service to be registered on eureka server or behave as eureka client because it is itself eureka server and other service will be registered here and will be eureka client for this service

Now we will run this service and will check on browser then Eureka UI will be rendered

##### Implementing Service Discovery Client

Now we will register our services on eureka server as eureka/discovery client

So for this first we will add dependencies inside our existing service pom.xml file

**Eureka Discovery Client SPRING CLOUD DISCOVERY**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

And where dependencies tag is ending after that we will add

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

Now inside the service main class we will add @EnableEurekaClient on the top of the class

Next thing we will do is to add the configuration in the application. properties file or application.yml file

eureka:  
 instance:  
 prefer-ip-address: true  
 client:  
 fetch-registry: true  
 register-with-eureka: true  
 serviceUrl:  
 defaultZone: http://localhost:9000/eureka/

and also

spring:  
 application:  
 name: serviceName

This service will be registered on the Eureka Server with this name

Till now we are using our services on the basis of IP