**Spring CLOUD**

When we implement micro services architectures we can create any number of micro services by using the modules provided by Spring Boot.For example if we are working on a hospital management software we can create a patient registration micro service a patient clinical micro service patient claim management and patient bed management micro services. But there will be several non-functional requirements for these micro services starting with service registration and discovery. That is each macro service will have to register itself with a centralized server and the other micro services will be able to discover that particular micro service and communicate with it dynamically without that each macro service will be tightly coupled to another micro service. So that is number one non-functional requirement.

Load balancing as the Load to our micro services increase will have to have multiple instances of the same micro service running on different servers and the load should be balanced.

Thirdly if something goes wrong in one of these micro services then entire systems should not come down or collapse. So these Microservice results should be fault tolerant and they should handle the faults gracefully. Easy integration they should be able to communicate with each other easily through restful client API. As for the API in easy steps.

Next will also how cross cutting concerns which are common across these macro services like security could be authentication authorization logging etc. which are common requirements across these micro services. So instead of repeating those cross cutting concerns across these micro services we should have one place where we can address them.

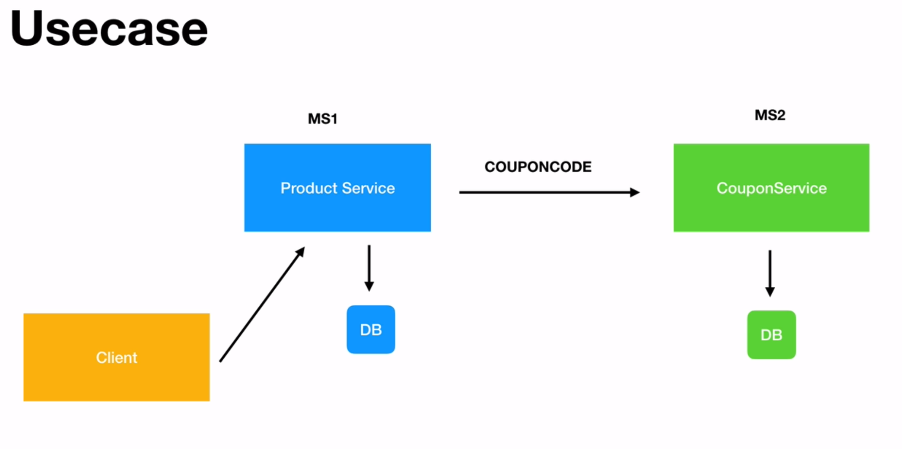
Last but not the least distributed tracing as the requests go from one micro service to another for example clinical to registration or registration too clinical. Too bad management. We should be able to trace how the requests are going and how the responses are coming back. When something goes wrong we can exactly pinpoint where it went wrong. Using distributed tracing so all these are not available in Spring Boot.

That is where ***spring cloud*** comes in spring cloud is a collection of open source components that help us implementing all these and more.

**Ports**

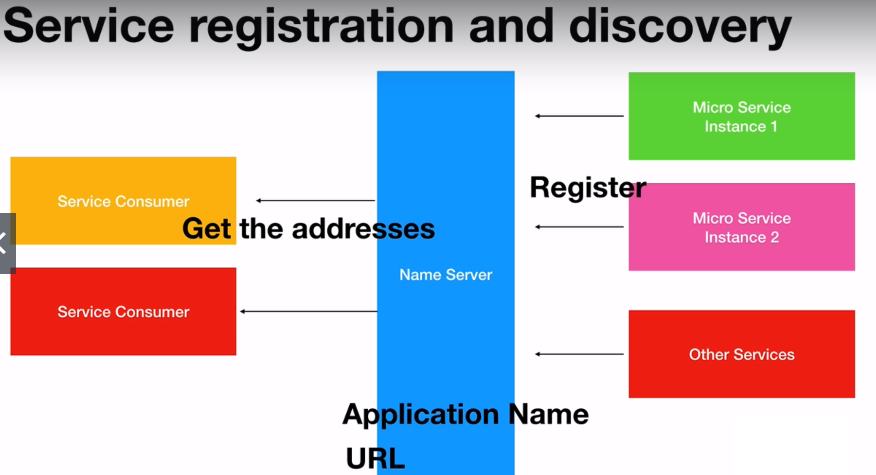
|  |  |
| --- | --- |
| ***Microservice Component*** | ***Ports Used*** |
| ***Coupon Service*** | ***8080 & 881*** |
| ***Product Service*** | ***9090*** |
| ***Eureka Server*** | ***8761*** |
| ***Zuul API Gateway Server*** | ***8765*** |
| ***Config Server (Spring Cloud Config)*** | ***8888*** |
| ***Zipkin Distributed Tracing Server*** | ***9411*** |

**Use case –**



**Eureka**

When we have multiple Microservices running or multiple instances of the same Microservices running. There will be multiple Microservices that require or communicate with these micro services through restful calls and to do that the consumers will have to know the URL details the details the port number etc. in order to communicate with the appropriate micro service. This will be very hard to maintain because there could be multiple instances of the same Microservices is running one of the instances could be down at a given point in time. So, this dynamism is hard to maintain. ***That is where spring cloud offers naming services or naming Server called Eureka.***

******

Micro services will register themselves as soon as they start or come up they will register themselves with this naming server through an application name or application ID which is a unique ID for each application and also that URL that is required to communicate with that. Server will automatically be fetch by this name server and it will store all that information once all that information is stored in Eureka Code naming Server the service consumers when they come up can communicate with the naming server fetch those details based on just the application name as long as the consumers know the application ID the unique application ID they can fetch the URL port number. All that communication details will be maintained by the Eureka server so Eureka server will decouple their micro services that are running and they can communicate with any other micro services easily through registration and discovery without maintaining any URL etc. That are required otherwise.

