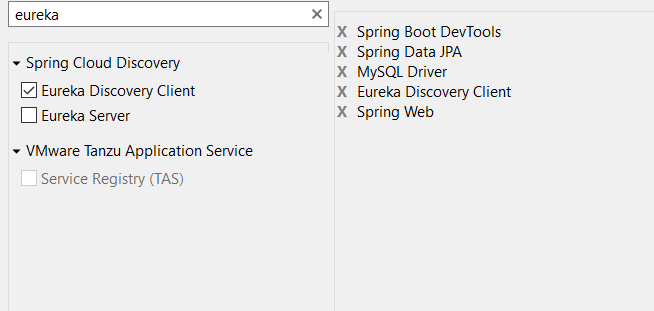


Microservice Project :

1. Citizen Service :



1. Eureka Server

Eureka Server is **an application that holds the information about all client-service applications**. Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

If the IP address of a server/container is fixed, then you can use this approach to easily call your service, but what happens when your IP address and hostname are unpredictable?

Nowadays, on a cloud platform, it is obvious that all servers or containers use dynamic IPs for autoscaling. And the interesting thing is that in microservices architecture, the key principle is that your service can autoscale as per load, so cloud platforms are ideal for microservices.

Therefore as a result, putting dependent service’s IP address in the config file is not a solution. We need a more sophisticated technique to identify the service, and Eureka server steps in here.



registerWithEureka – false: It means you are the service registery so do not use othere eureka server

fetchRegistry – false

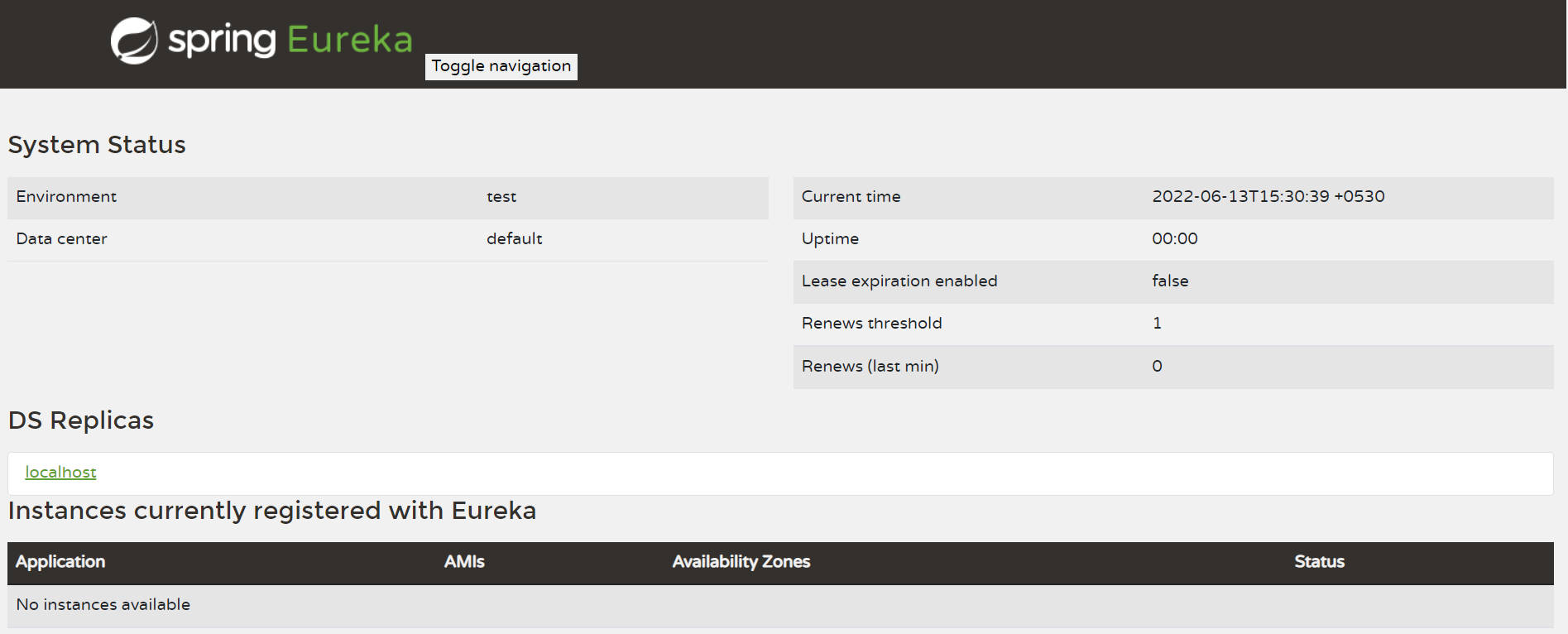
<dependency>

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

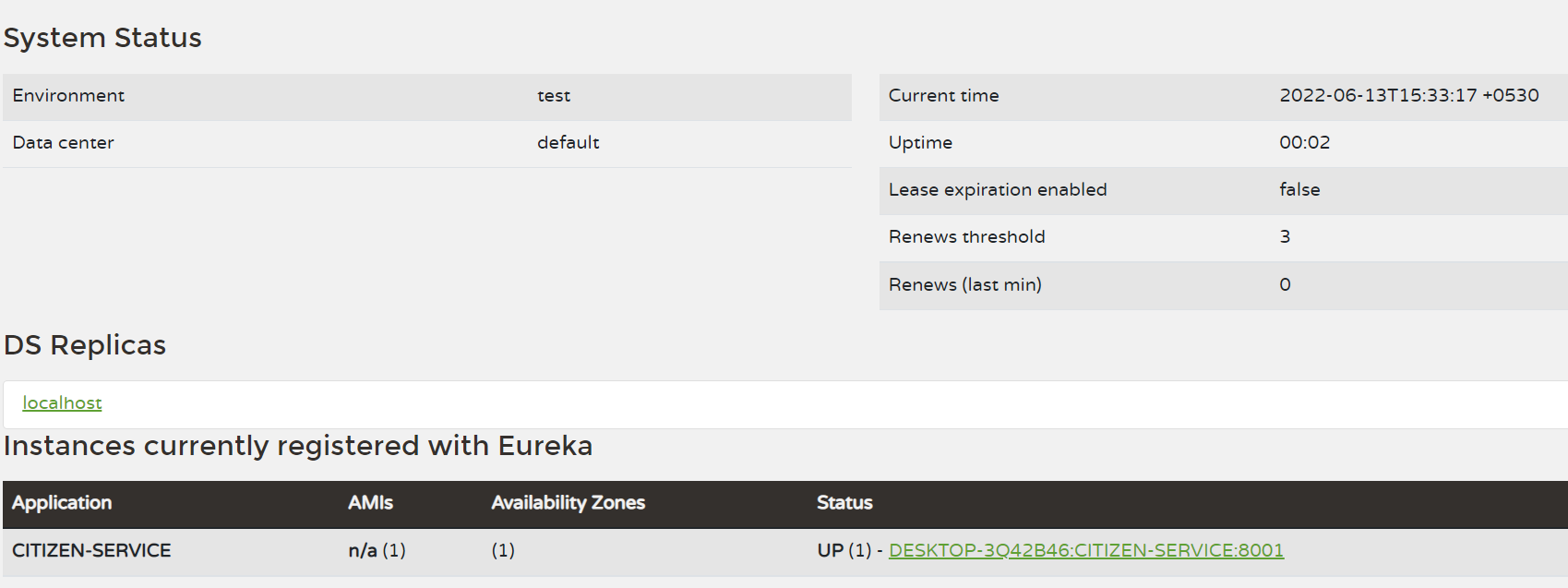
<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

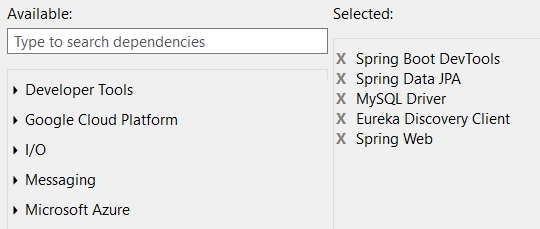
@EnableEurekaServer – enable eureka server to spring boot app



IF you run citizen also then your project register with eureka



1. Vaccination server



To get connect with vaccination module to citizen module we need rest template.

The [RestTemplate](http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/web/client/RestTemplate.html) is the central Spring class for client-side HTTP access.

#### **What is Load Balancing?**

Load balancing means efficient distribution of incoming network traffic across a backend servers. It distributes client requests or network traffic efficiently across multiple servers.

**Spring client side load balancing**

Spring Netflix Eureka has a built-in client side load balancer called Ribbon.

Ribbon can automatically be configured by registering RestTemplate as a bean and annotating it with @LoadBalanced.

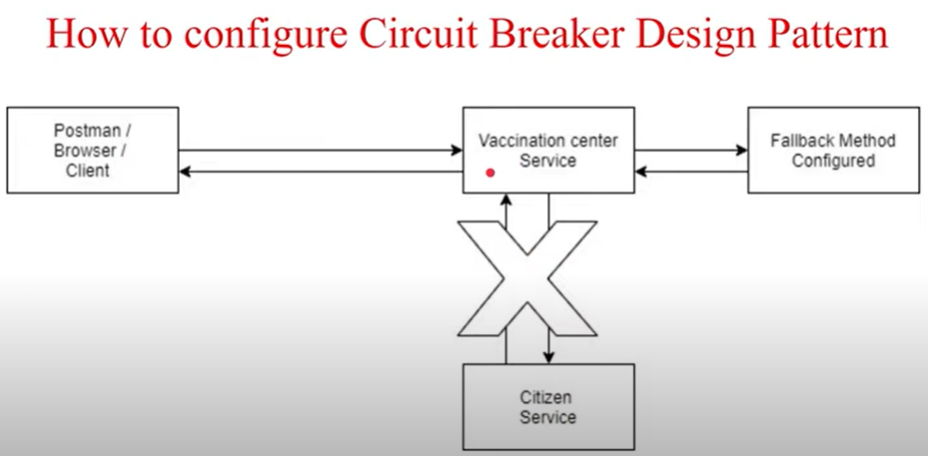
@Loadbalanced is basically used when you don’t need to hard card your localhost URL every time so use eureka service register name instead of localhost:8000

**Fault in your app :**

If your citizen service is down somehow then your vaccination service is worked but your main end point response will not get there is internal server error. This called fault in your tolerance in your app.

You should get answer even if your citizen service down because vaccination service is working properly and at least you should get vaccination details in response.

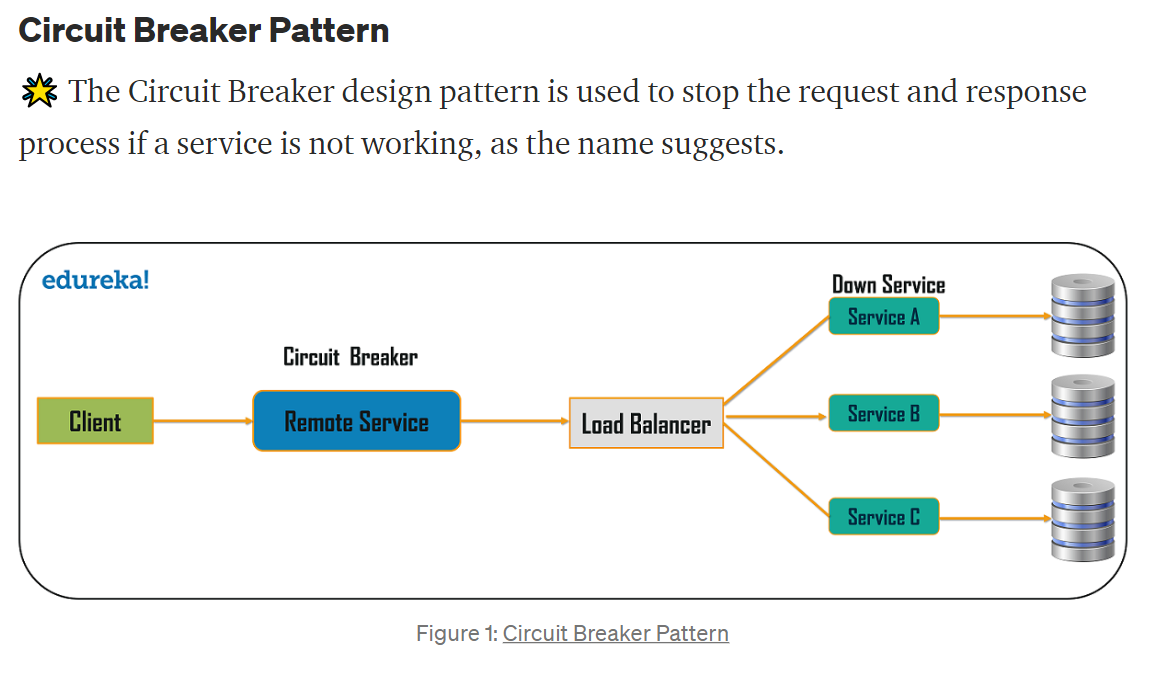
We have circuit breaker design pattern follow.



To implement this us : Hystrix

@EnableCircuitBreaker

@HystrixCommand => it has many method one of method is fallbackmethod is method if one service is down in that case give you other service response atleast.



Why this type of issue come :

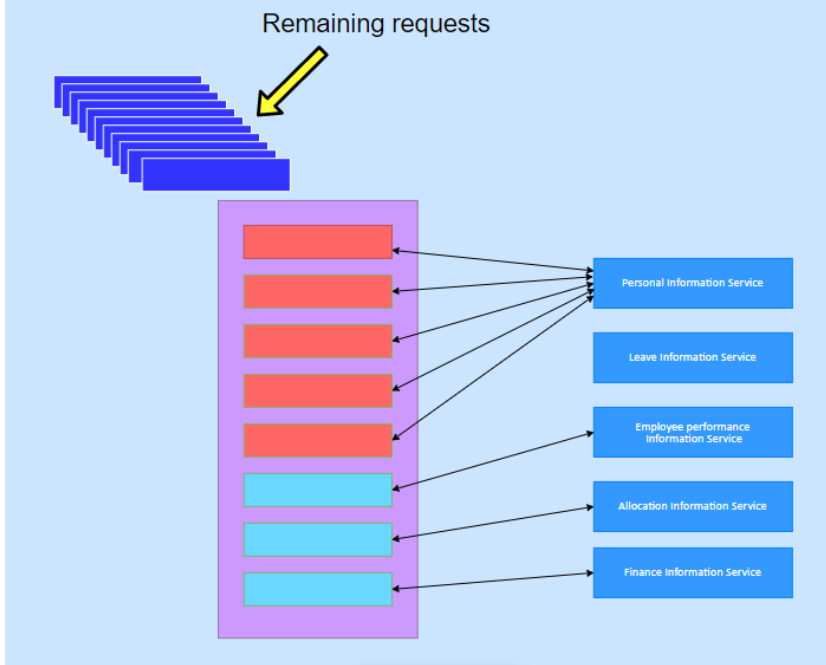
* First, because the consumer will be unaware that a particular service is unavailable (failed), so the requests will be sent to that service continuously.
* The second issue is that network resources will be exhausted with low performance and user experience.

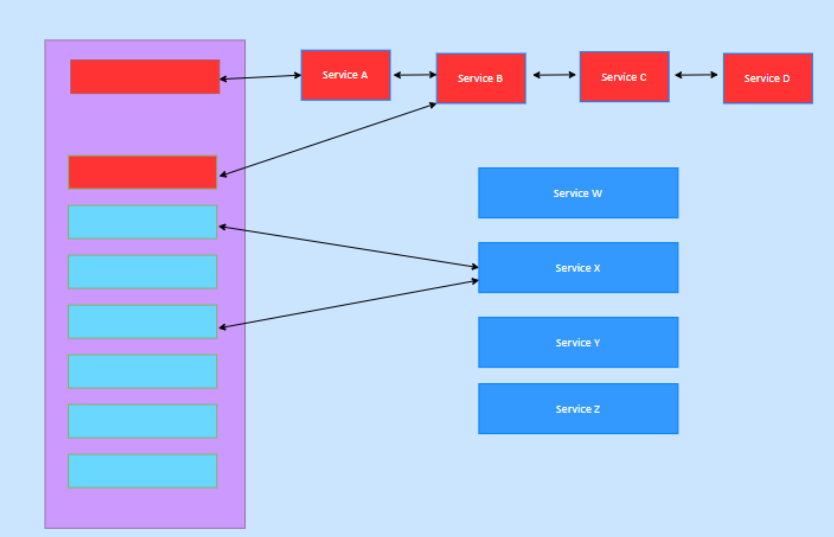
# What are the causes to break the services?

Use case 1 : When Multiple request coming from client to service that time service will not give proper response each and every request so I may delay or service gets so load so it kill or down.

Use Case 2 : When let say you services which is dependent on themselves that time let say if last service is not working then second last service or all previous service will not get proper response

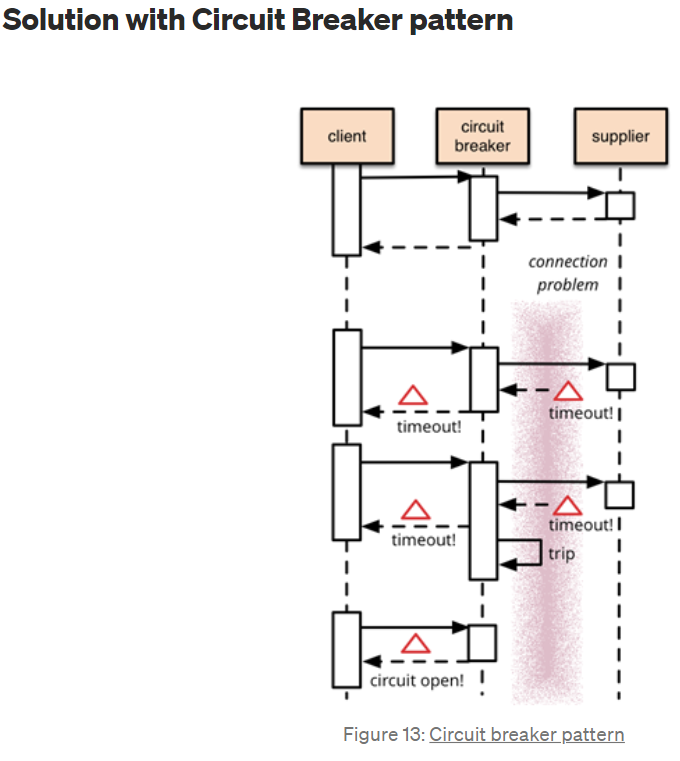
So we need circuit breaker pattern to avoid such senerio.

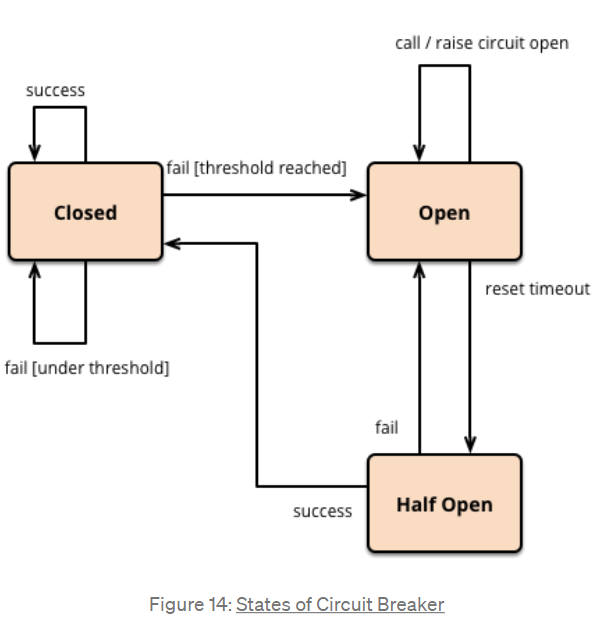




Working:

<https://medium.com/geekculture/design-patterns-for-microservices-circuit-breaker-pattern-276249ffab33>





The Circuit Breaker pattern has 3 states.

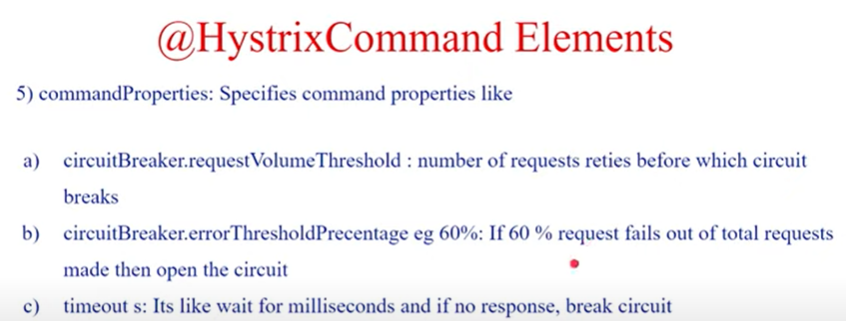
1. **Open.**
2. **Closed.**
3. **Half-Open.**

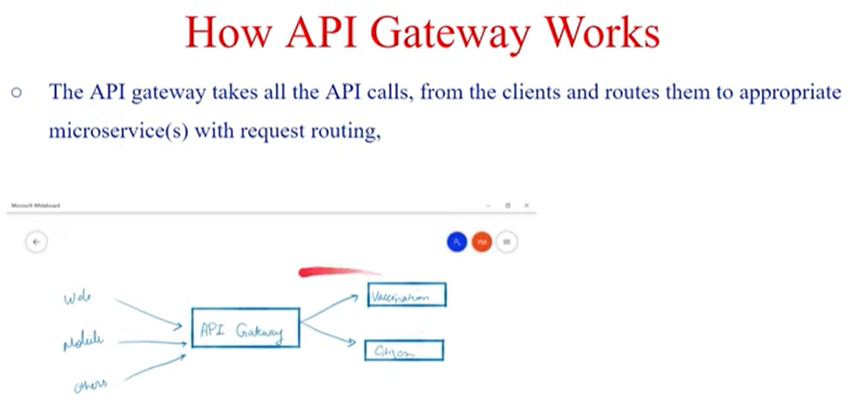
First time your circuit breaker is in close state. If you have multiple request and your service is giving response in delay in that case it reaches to threshold time delay so your circuit breaker is opened.

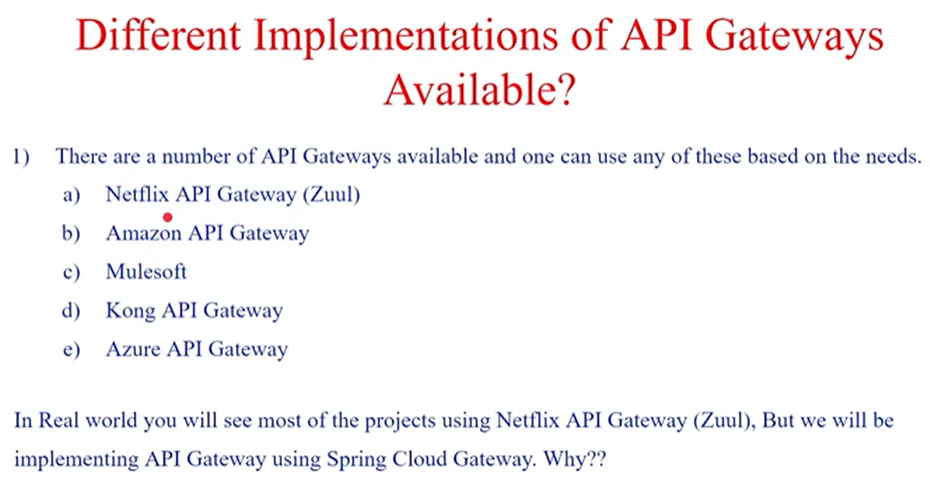
 That’s the principle behind Circuit Breaker Pattern.

*From the user’s perspective, having to wait a long time for a response is not a good user experience. Rather than keeping the consumer waiting for a longer duration, it is better to respond quickly. It doesn’t matter if it’s a success or a failure; what counts is that the user isn’t kept waiting.*

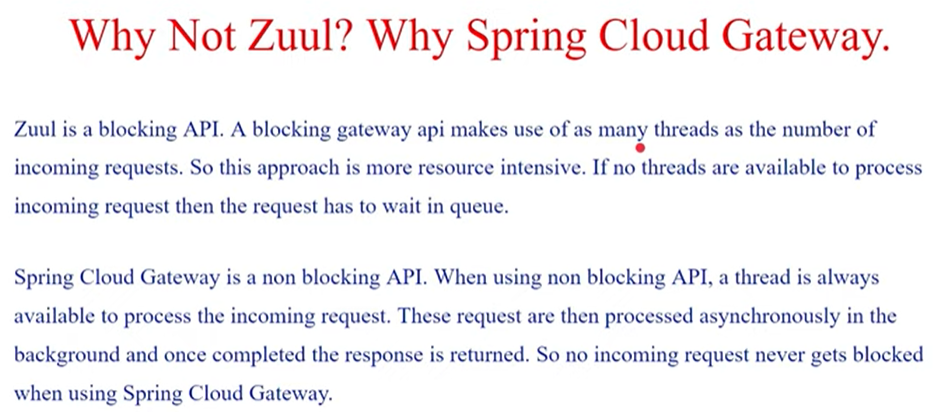


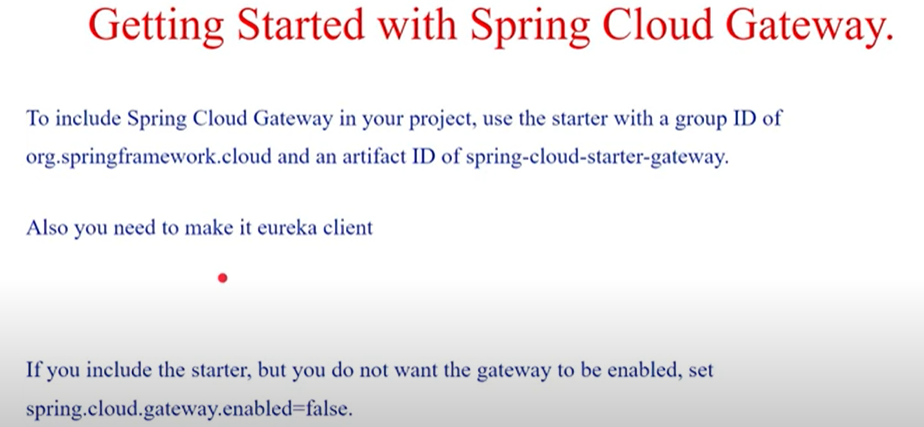












API-Gateway :

