**Circuit Breaker :**

In microservices architecture there is a complex call chain.

A microservice can call another microservice. That microservice might be dependent

on another microservice and so on.

And what would happen if one of these services is down or is very slow?

Let's say microservice four is down or it's very, very slow.

What would happen?

There would be an impact on the entire chain. If the microservice four is down,

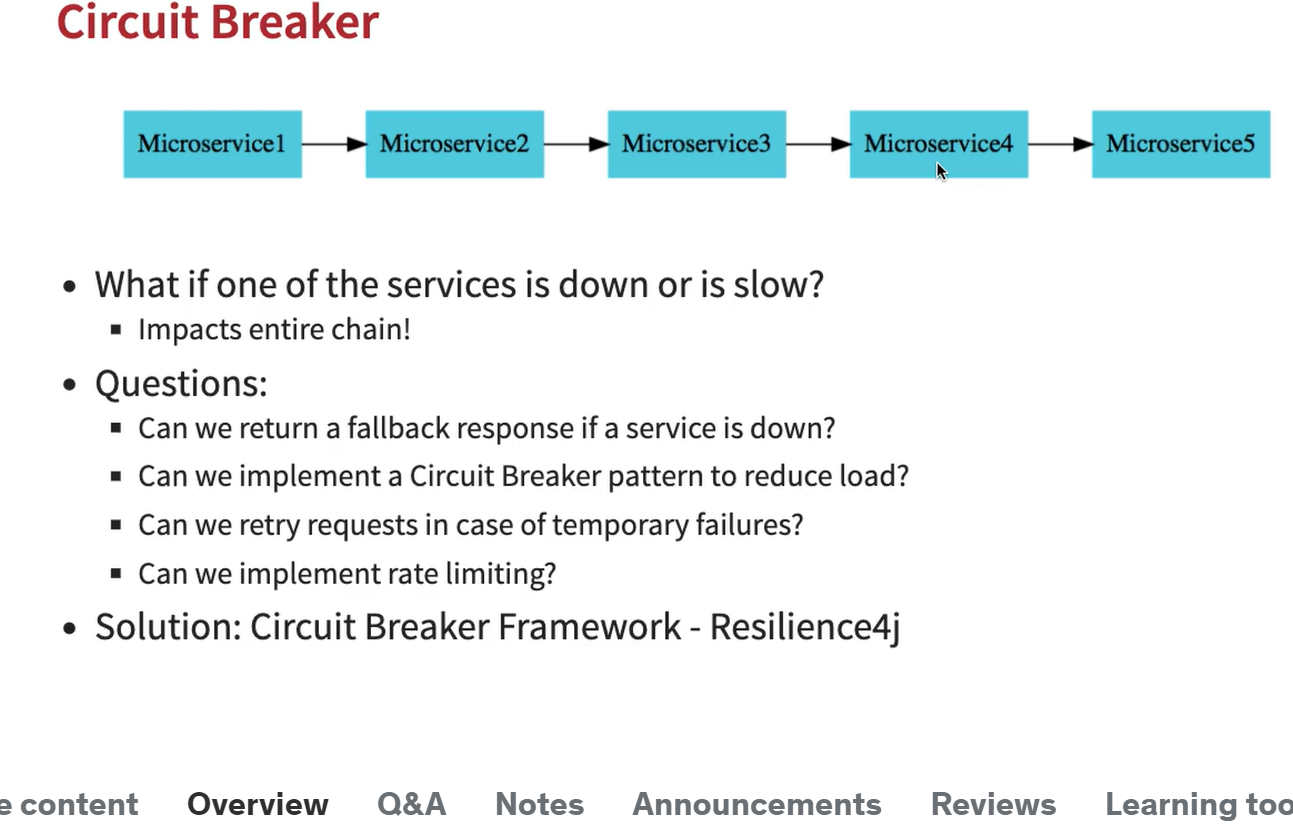
then Microservice three also will be down, Microservice Two also will be down,

because these are all depending on microservice 4.

Even if it's slow, then there is a corresponding impact on the other microservices too.

In these microservices, there will be a buildup of calls because this microservice is slow.

In the previous versions of Spring Boot and Spring Cloud, **Netflix Hystrix** was the recommended



If I see that microservice four is down, in microservice three, can I return a fallback response?

Can I configure a default response?

This might not always be possible. For example, in the case of a credit card transaction,

or something of that kind, you do not have any fallback responses possible.

But in the case of a shopping application, instead of returning a set of products you might return a default set of products.

The other question to consider is can we implement a circuit breaker pattern to reduce the load?

If I see that microservice four is down, instead of repeatedly hitting it and causing it to go down, can I actually return the default response back without even hitting the microservice?

If there is a temporary failure from a microservice four, can I retry it a few times

and only when it has failed multiple times, I return a default response back

1. add 3 dependence

<!--3 dependency need for curcuit breaker aop, actuator, resilience4j-->  
  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>  
<dependency>  
 <groupId>io.github.resilience4j</groupId>  
 <artifactId>resilience4j-spring-boot2</artifactId>  
</dependency>  
  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-aop</artifactId>  
</dependency>

@Retry(name = “sample-api”)

If we annotated @retry in the method when it's executing if there's an exception, then

what would happen is it would be retried thrice and if the Retry fails all the three times,

only then it would return a error back.

Addition in properties file (sample-api = its endpoint name)

1. resilience4j.retry.instances.sample-api.maxAttempts=5 #NEW

It will try for 5 times as mentioned in app.prop , while default will try 3 times.

So you can see that each subsequent Retry is taking longer

and longer and that's why is called exponential back off.

1. **@Retry(name = "retry-sampleService", fallbackMethod = "fallBackCall")**

**fallBackCall** is the method name that we have to create to send some default response In case of retry failure.

NOTE : that method need to pass with Exception param as shown below

private String fallBackCall(Exception ex){  
  
 return "fall-back-response";  
}

**3. resilience4j.retry.instances.retry-sampleService.waitDuration=5s**

Each retry will wait for 5s here

4. **Resilience4j.retry.instances.retry-sampleService.enableExponentialBackoff = true**

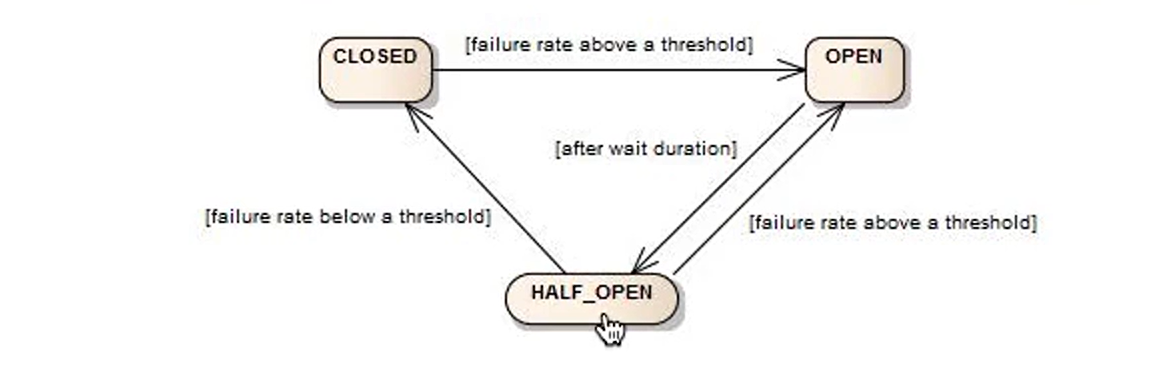
So each subsequent Retry is taking longer and longer and that's why is called exponential back off.

**@CircuitBreaker(name = "default", fallbackMethod = "fallBackCall")**

The circuit breaker framework allows us to do If a microservice is down ,Then why do I need to really call it and add load to it?

Why can't I return a default response back directly?

So that is what the circuit breaker does is It'll break this circuit and it'll directly return a response back.



So in a **closed state** I'll always be calling the dependent microservice.

In a **open state**, the circuit breaker will not call the dependent microservice. It'll directly return the fallback response.

And i**n a half open state**, a circuit secure breaker would be sending a percentage of requests to the dependent microservice, and for rest of the requests, it would return the hard coded response or the fallback response back.

For example, the circuit breaker is in the closed state.When you start the application up the circuit breaker is typically in a closed state.

Let's say I'm calling the dependent microservice 10,000 times and if all of them are failing

or 90% of them are failing. In that kind of scenario, the circuit breaker would switch to a open state.

Once it switches through an open state it waits for a little while. there's a wait duration that you can configure.

After that wait duration the circuit breaker would switch to a half open state.

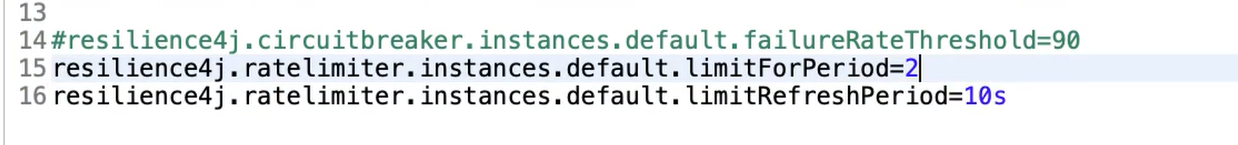
During the half open state. The circuit breaker would try and see if the dependent microservice is up. So it sends a percentage of the request.

You can configure how much percentage.

if it gets proper responses for that then it would go back to the closed state. If it does not get proper responses, then it would go back to the open state.

@RateLimiter(name = ‘default’)

We are setting up like in 10sec only 1000 call is allow to the specific end.



@bulkhead

 Allow concurrent calls eg 20 concurrent calls.

