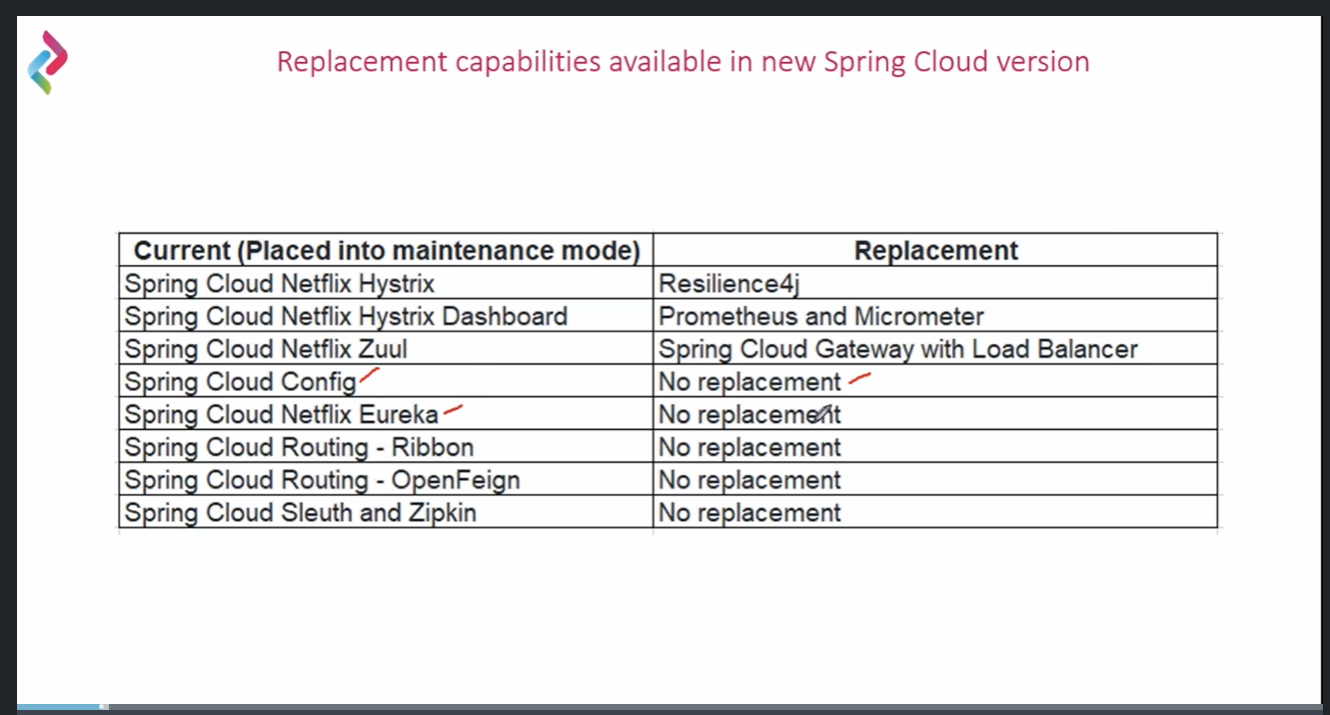
**Note : Spring Cloud new version capabilities:**



Spring config (Distributed Configuration)

=============================

Implementation :

spring cloud config

- config client get properties and start of the application

- need to make actuator/refresh api for each service

Spring Cloud Bus

- no need to call actuator/refresh for each service instead of need to config service

actuator/busrefresh it will update the all serivce with the help of distibuted cofig using

Advance Message Queue protocal (AMQP)

publisher and receiver event broadcasting

problem:

- update the config need to re-deploy the application (we have down time)

config-server

-------------

1) <dependency>

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

<artifactId>spring-cloud-config-server</artifactId>

</dependency>

2) create git repository for store property files

3) Added the git properties in config server - application.properties

spring.cloud.config.server.git.uri=https://github.com/karthick678/talentnext

#spring.cloud.config.server.git.username=

#spring.cloud.config.server.git.password=

4) add @EnableConfigServer in spring boot started application

config-client

------------

1) <dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

2) spring.cloud.import=optional:configserver:http://localhost:8888 (config-server url)

spring.cloud.config.label=main // git branch name

spring.application.name=product-service // properties file name

3)client app get the properties from the server while do below any one step:

- On start of the application get the properties from config server

- on while do manual refresh of config client - POST - http://localhost:8080/actuator/refresh

will receive the updated properites from server

4) use @Refreshscope for custom application property update

Spring Cloud Bus

=============================

Config-server

------------

1) <dependency>

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

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

</dependency>

2) Enble endpoint in propertties file

management.endpoint.web.exposure.include=busrefresh

# message broker

spring.rabbitmq.host=localhost

spring.rabbitmq.port=15672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

config-client

-------------

1) <dependency>

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

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

</dependency>

# message broker

spring.rabbitmq.host=localhost

spring.rabbitmq.port=15672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

testing

-------

1) update the properties file in github

2) http://localhost:8888 - config server

hit post api through postman (http://localhost:8888/actuator/busrefresh)

Service Registory and Sevice Discovery (Netflix Eureka or Consul or Netflix zookeeper)

=================================================

Aws EC2 - ip will change after stop and start

docker - port will change dynamic

problem:

dynamic port no or dynamic ip will generate while creating multiple instane for a service

while do service to service call - service will be unreachable (where the required service is located)

Self Registory

--------------

service registory must be registor when startup and unregistor on shutdown

service registory unregistor while craushing

service registory unregistory not able to handle the request

Eureka server (Service Registry)

----------------

1) create project for service register

<dependency>

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

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

</dependency>

2) create a application.properties/bootstrap.properties

server.port=8761

# no need to register this application with eureka

eureka.client.register-with-eureka=false

# no need to fetch the registred service information

eureka.client.fetch-registry=false

3) Add @EnableEurekaServer in application.java

application became service registry

4) open in browser http://localhost:8761

Eureka client (Service discovery)

--------------------------------

1) add below dependeny in the service app

<dependency>

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

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

</dependency>

2) eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

eureka.instane.preferIpAddress=true

3) spring.application.name=product-service

4) @EnableDiscoveryClient

5) hit http://localhost:8761/ we can see the client service is present in eureka server

Client Side Discovery pattern or Client Side Load balanceing (Implementation Nexflix Ribbon - one service call another service)

===================================================================================

https://microservices.io/patterns/client-side-discovery.html

- Registry awre HttpClient

- Round Rabbin Algoritham

Spring Cloud Ribbon

- @LoadBalanced

1) need to add dependency

<dependency>

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

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

</dependency>

client side load balanceing with ribbon

----------------------------------------

1) create project with -

- Eureka discovery client

- spring web

- spring boot actuator

2) in application.properties

server.port=9997

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

eureka.client.register-with-eureka=false

3) consume

http://product-service/product/1

4) RestTemplate - @LoadBalancec

5) create config class

@Configuration

public class config {

@LoadBalanced

@Bean

public RestTemplate getRestTemplate() {

return new RestTemplate();

}

}

6) Create product service controller

@RestController

@Scope("request")

public class ProductClientController {

@Autowired

private RestTemplate restTemplate;

@GetMapping(value = "/get-products/{id}", produces = {MediaType.APPLICATION\_JSON\_VALUE}, consumes = {MediaType.APPLICATION\_JSON\_VALUE})

public Product getProductById(@PathVariable("id") Integer id) {

Product product = restTemplate.getForObject("http://product-service/products" + id, Product.class);

return product;

}

}

7) test http://localhost:9997/get-products/1

problem:

---------

need to repeat the controller code again and again for all api rest point repeat with rest template

Spring Feign client (will give dynamic implementation)

==========================================================

- It is used for service to service call in synchronous way

- need to write a interface no need to worry about the implementation

dependency>

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

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

</dependency>

1) create project with

- openfeign

- Eureka discovery client

- spring web

- spring boot actuator

2) in application.properties

server.port=9996

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

eureka.client.register-with-eureka=false

3) consume

http://product-service/product/1

http://product-service/products

4) create ProductServiceProxyInterface

@FeignClient("product-service")

public interface ProductServiceProxy {

@GetMapping(value = "/products", produces = {MediaType.APPLICATION\_JSON\_VALUE})

public List<Product> getAllProducts();

@GetMapping(value = "/product/{id}", produces = {MediaType.APPLICATION\_JSON\_VALUE})

public Optional<Product> getProductById(@PathVariable("id") Integer id);

}

5) create controller

@RestController

@Scope("request")

public class ProductClientController {

@Autowired

private ProductServiceProxy productServiceProxy;

@GetMapping(value = "/get-products/{id}", produces = {MediaType.APPLICATION\_JSON\_VALUE})

public Optional<Product> getProductById(@PathVariable("id") Integer id) {

return this.productServiceProxy.getProductById(id);

}

@GetMapping(value = "get-products", produces = {MediaType.APPLICATION\_JSON\_VALUE})

public List<Product> getAllProducts() {

return this.productServiceProxy.getAllProducts();

}

}

6) http://localhost:9996/get-products/2

service to service call

---------------------------

1) Ribbon (RestTemplate - register aware httpclient)

- @LoadBalanced

2) OpenFeign

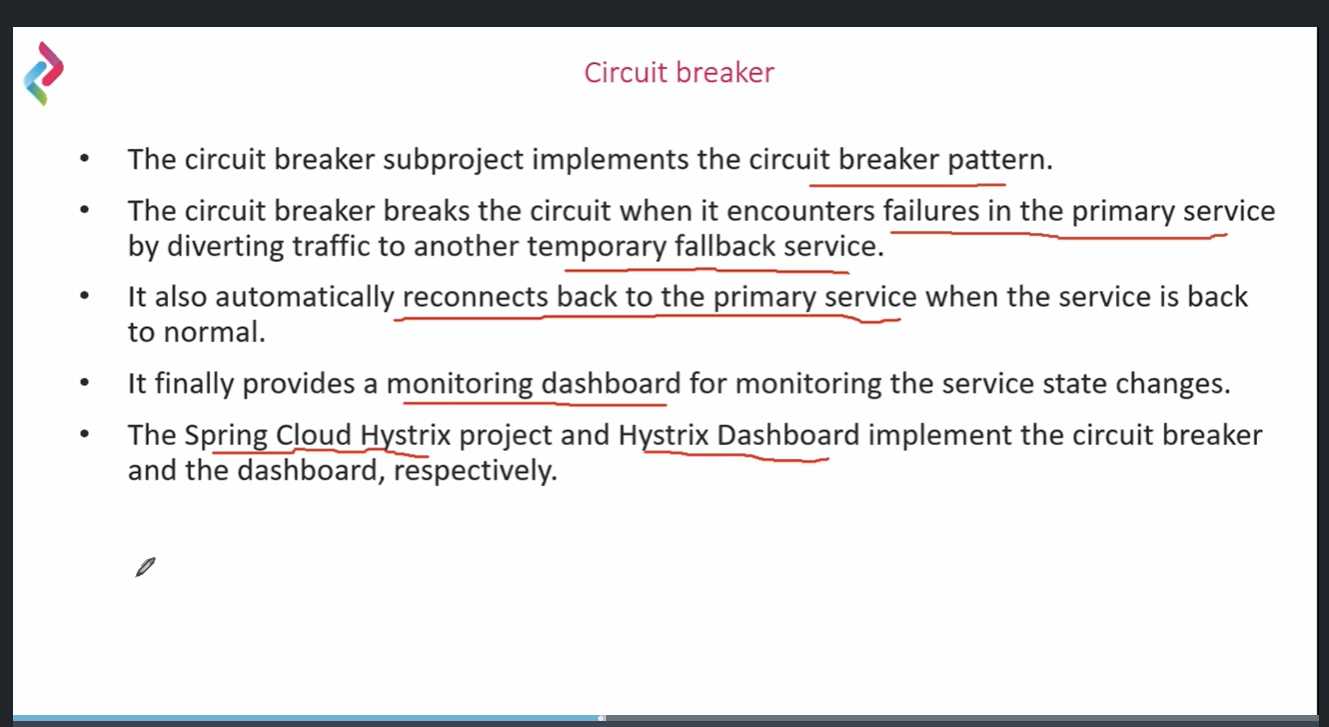
- @EnableOpenFeign

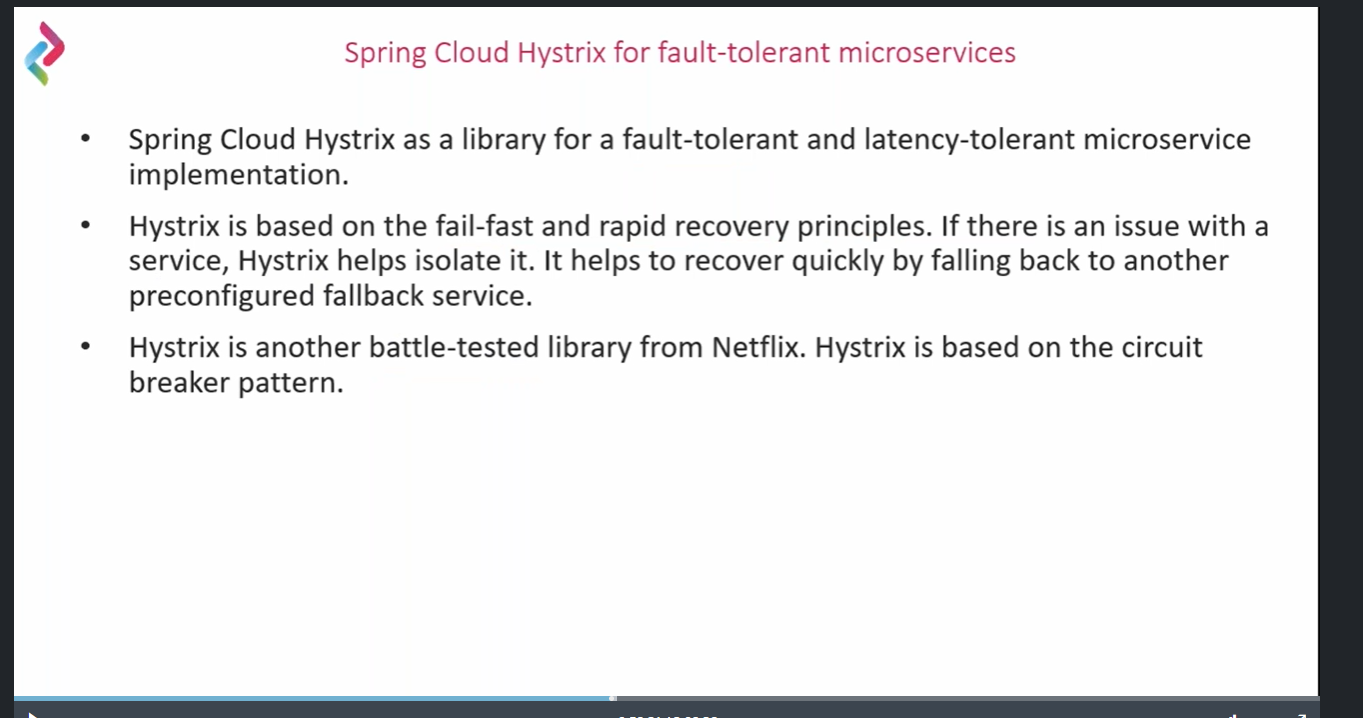
- @FeignClient("product-service")

Circuit Breaker pattern

======================================================================

https://microservices.io/patterns/reliability/circuit-breaker.html





way of implementation

---------------------

1) Hystrix

Spring Boot < 2.4.x

2) Resilience4j

Spring Boot >= 2.4.x

Netflix Hystix: (Downgrade)

-----------------

Spring boot: 2.3.10.RELEASE

Spring Cloud: Hoxtonz.SR11

Spring Cloud Hystrix

Service to service call

1. Ribbon (REST template)

-@HystrixCommand(fallbackMethod=” ”)

1. OpenFeign

-@FeignClient(name=”product-service”, fallback=ProductServiceProxyImpl.class)

-----------------------------------------------------------------------------------

Note:

-@HystrixCommand(fallbackMethod=” ”) /@FeignClient(name=”product-service”, fallback=ProductServiceProxyImpl.class) will works in a class marked with @Service/@Component

-------------------------------------------------------------------------------------------------------

Steps:

1. Create a spring boot project: (storeApp-Consumer-Eureka-Ribbon-Hystrix)
2. Starter dep – Eureka Discovery Client, web, lombok, actuator, Netflix Historix
3. Change the version spring boot 2.3.10.RELEASE and spring cloud: Hoxton.SR11
4. Pom.xml

<**dependency**>

<**groupId**>org.springframework.cloud</**groupId**> <**artifactId**>spring-cloud-starter-hystrix</**artifactId**> </**dependency**>

5) In application.properties

` spring.application.name=storeapp-consumer-eureka-ribbon-hystrix  
  
server.port=9995  
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka  
eureka.client.register-with-eureka=false  
  
#Enable Actuator  
management.endpoints.web.exposure.include=\*

6) create a product service class

-@HystrixCommand(fallbackMethod=” ”) /@FeignClient(name=”product-service”, fallback=ProductServiceProxyImpl.class) will works in a class marked with @Service/@Component

7) Write call back method when there is a exception occurring while do the RESTTemplate service call

@HystrixCommand(fallbackMethod = "fallbackMethodForGetProductById")

8) need to add in Application class

@EnableCircuitBreaker

----------------------------------------------------------

Circuit break will check the service available or not For update the status of the service

Will take some default timeout 800. We can update using below flag

execution.isolation.thread.timeoutInMilliseconds

------------------------------------------------------------------------------

**OpenFeign with circuit break using Hystrix**

1. Create a spring boot project: (storeApp-Consumer-Eureka-feign-Hystrix)
2. Starter dep – OpenFeign, Eureka Discovery, web, lombok, actuator, Netflix Historix
3. Change the version spring boot 2.3.10.RELEASE and spring cloud: Hoxton.SR11
4. Pom.xml
5. <dependency>  
    <groupId>org.springframework.cloud</groupId>  
    <artifactId>spring-cloud-starter-hystrix</artifactId>  
    <version>1.4.7.RELEASE</version>  
   </dependency>

6) In application.properties

` spring.application.name=storeapp-consumer-eureka-feign-hystrix  
  
server.port=9994  
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka  
eureka.client.register-with-eureka=false  
  
#Enable Actuator  
management.endpoints.web.exposure.include=\*

1. # for handle the callback

feign.hystrix.enabled=true

1. Create a ProductServiceFallback class

* Should be annotated with @Component/@Service

1. @FeginClient(fallback= ProductServiceFallback.class)
2. Add @EnableCircuitBreaker in the applicationMain class

**hystrix dashboard:**

1. <dependency>

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

<artifactId>spring-cloud-netflix-hystrix-dashboard</artifactId>

<version>2.2.10.RELEASE</version>

</dependency>

1. @EnableHystrixDashboard in Application.class file
2. Application.properties

hystrix.dashboard.proxy-stream-allow-list=\*

1. Hystrix stream

<http://localhost:9994/actuator/hystrix.stream>

1. Hystrix dashboard

* Hit <http://localhost:9994/hystrix>
* Pass url: http://localhost:9994/actuator/hystrix.stream

API Gateway

======================================================

<https://microservices.io/patterns/apigateway.html>

**Netflix zuul:**

Spring Cloud Routing : (zuul)

A screenshot of a computer

Description automatically generated

A diagram of a software application

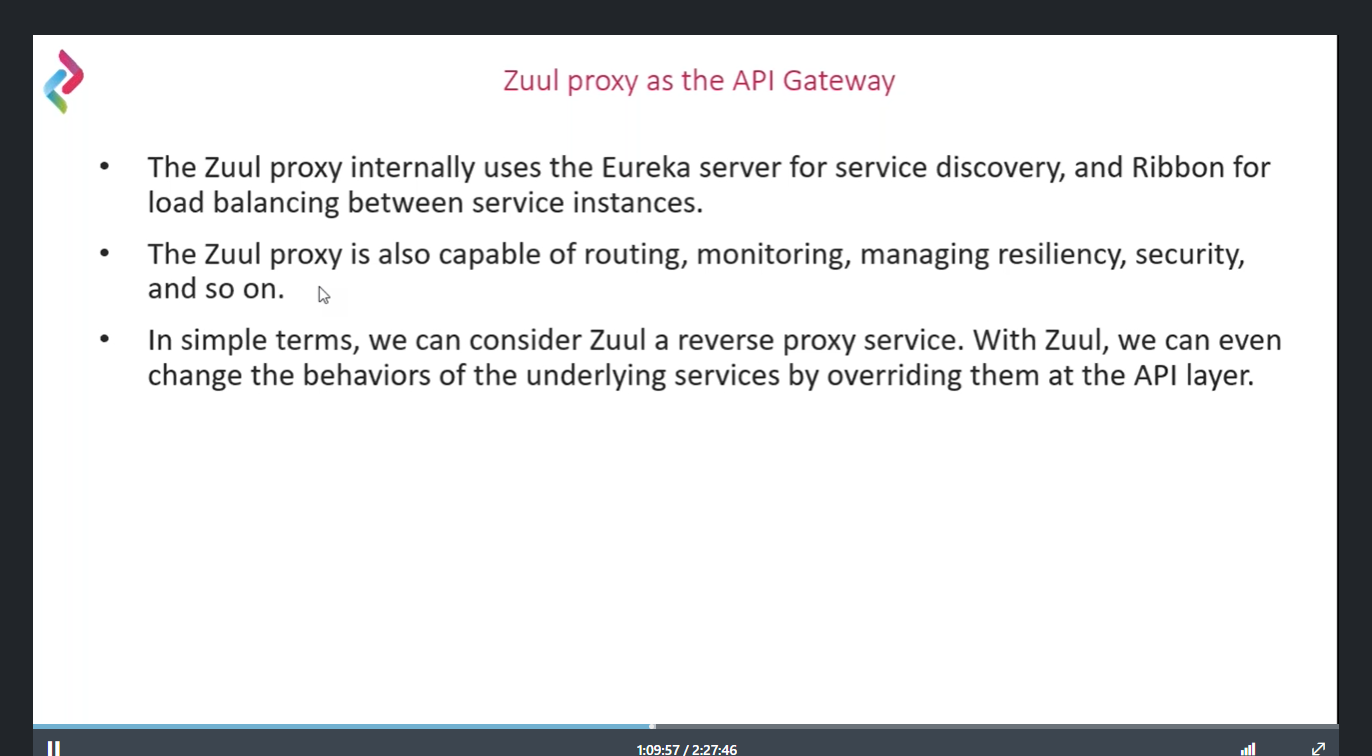
Description automatically generated

A screenshot of a computer

Description automatically generated

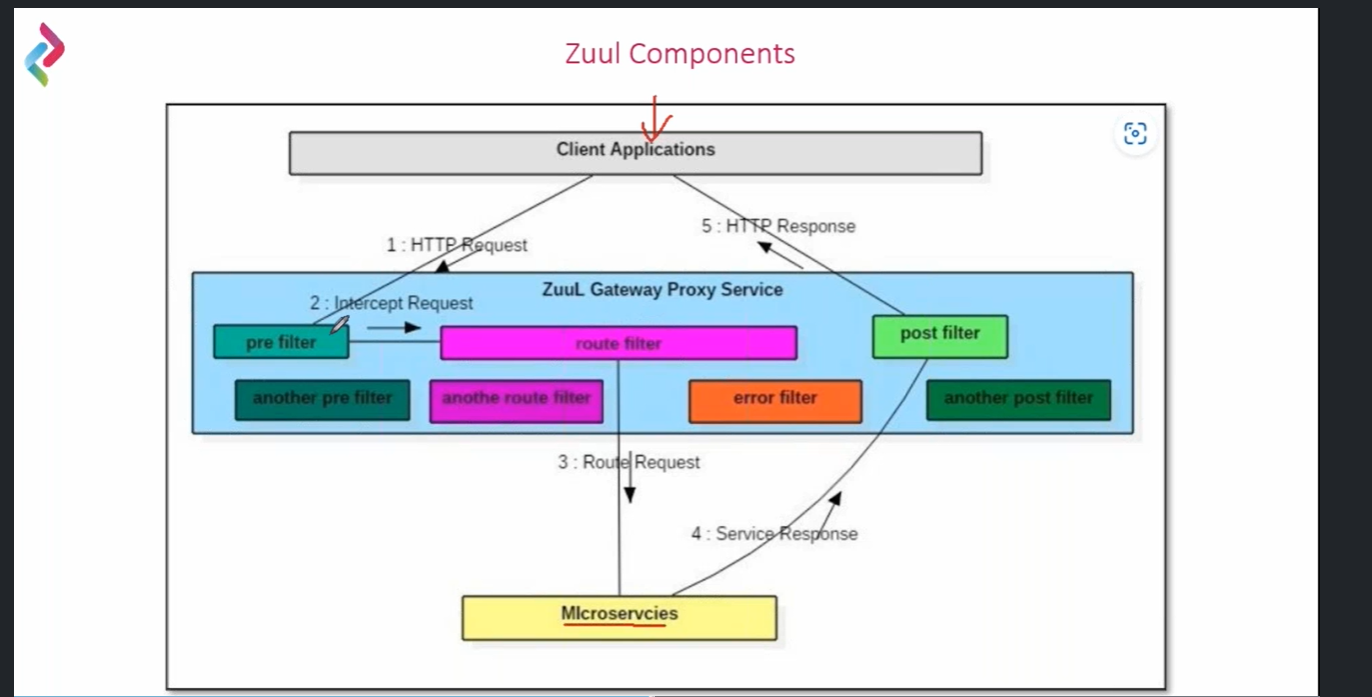
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



**Steps**:

Create new project (storeapp-proxy-api)

Add Eureka discovery client – help for lookup from service registry, web, Actuator

1. <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-zuul -->

<dependency>

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

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

<version>2.2.10.RELEASE</version>

</dependency>

1. Spring Boot: 2.3.10.RELEASE

Spring Cloud: Hoxton.SR11

1. @EnableZuulProxy in Spring boot application class
2. Create the different types of custom filters extending zuulFilter

* PreFilter
* PostFilter
* RouteFilter
* ErrorFilter

1. Register the Filters

6)

A screenshot of a computer

Description automatically generated

**Resilence4j (Circuit break)**

====================================

* It contains functional programming features like functional Interface, lambda expressions, method references.
* It contains retry mechanisms, cache cells results, bulkheads, ratelimter.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

