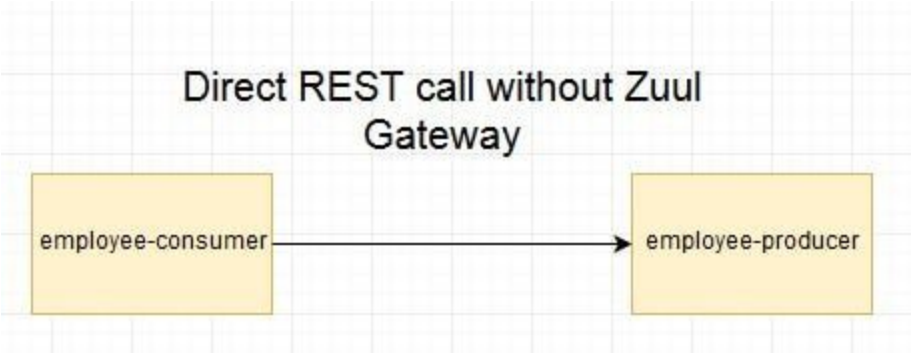
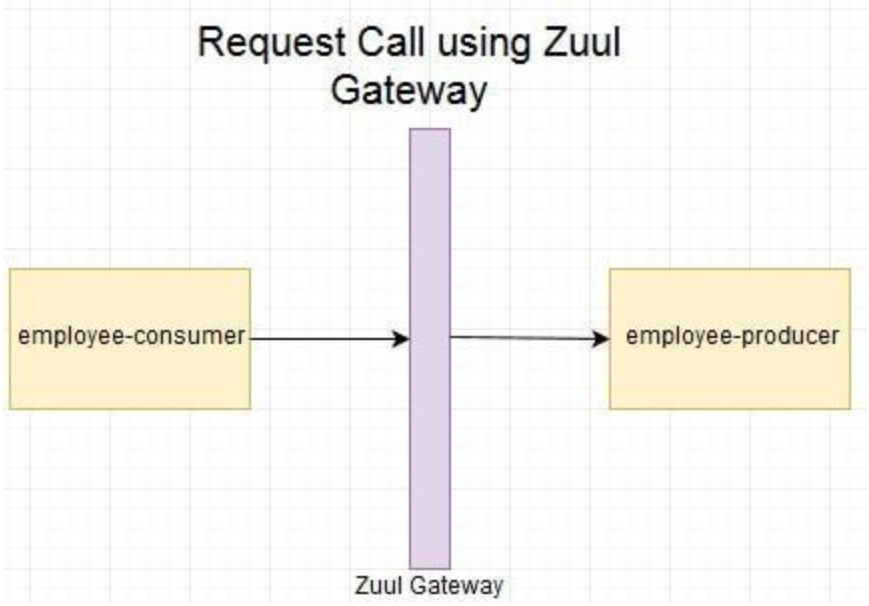
What is the Netflix Zuul? Need for it?

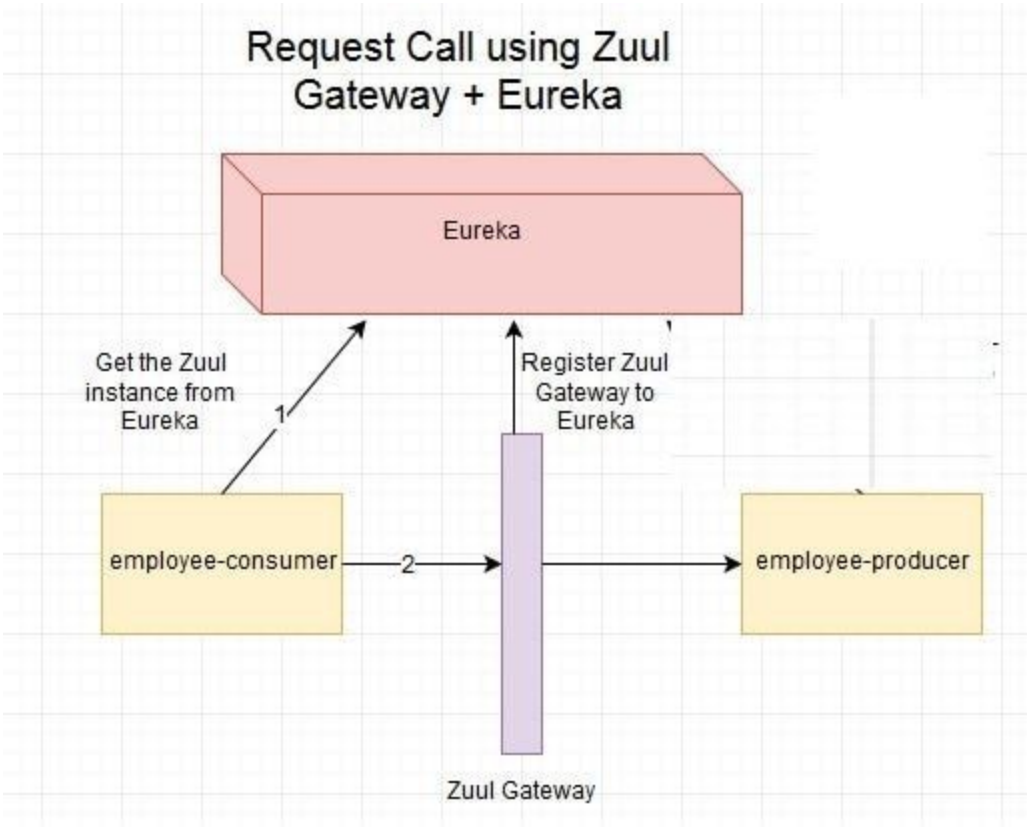
Zuul is a JVM based router and server side load balancer by Netflix. It provides a single entry to our system, which allows a browser, mobile app, or other user interface to consume services from multiple hosts without managing cross-origin resource sharing (CORS) and authentication for each one. We can integrate Zuul with other Netflix projects like Hystrix for fault tolerance and Eureka for service discovery, or use it to manage routing rules, filters, and load balancing across your system.
Microservice call without Netflix Zuul.



- Microservice call with Netflix Zuul



- Microservice call with Netflix Zuul + Netflix Eureka



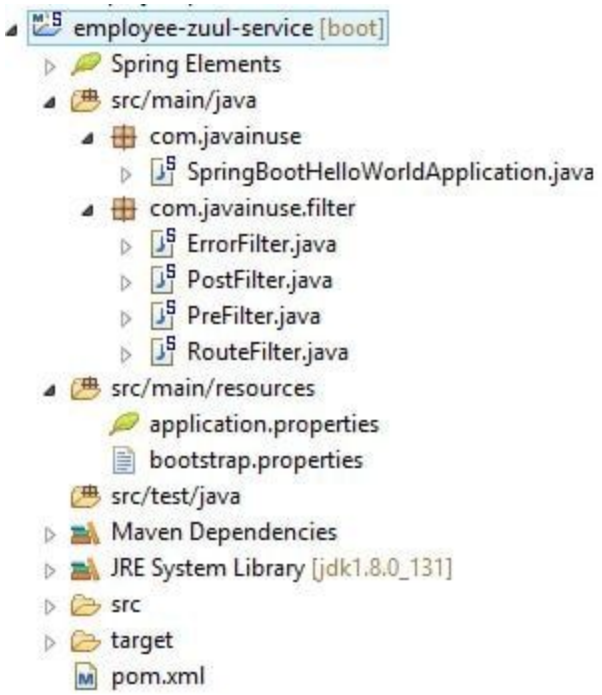
Lets Begin-

- ➢ We will be creating four modules as shown in the above diagram.
- ➢ employee consumer
- ➢ employee producer
- ➢ Eureka Server
- ➢ employee-zuul-service

Of these modules there will be no change in the employee-producer and Eureka Server code we had developed in the Netflix Eureka Tutorial. We will be creating a new module employee-zuul-service and modifying the employee-consumer module code developed in the Netflix Eureka Tutorial.

Zuul Gateway

The project structure for this module will be as follows-



The pom.xml will be as follows with the zuul dependency.

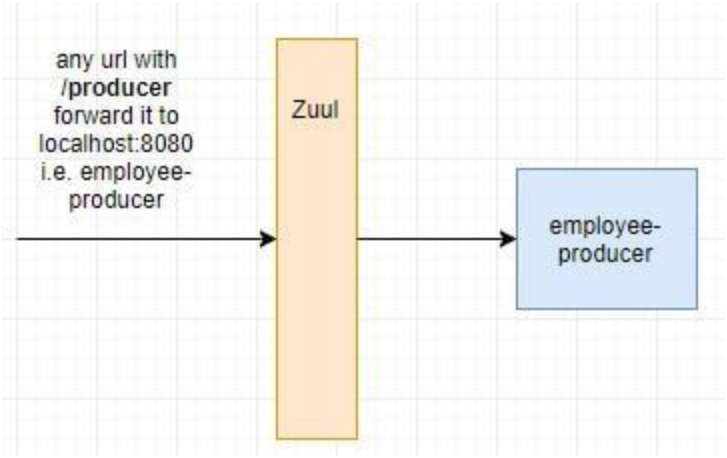
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.javainuse</groupId>
    <artifactId>employee-zuul-service</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>SpringBootHelloWorld</name>
    <description>Demo project for Spring
Boot</description>
```

```
	<parent>
		<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>
		<version>1.4.1.RELEASE</version>
		<relativePath /> <!-- lookup parent from
repository -->
	</parent>

	<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourc
eEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporti
ng.outputEncoding>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>

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

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

		<dependency>
```

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

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

     </dependencies>

     <dependencyManagement>
          <dependencies>
               <dependency>

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

<artifactId>spring-cloud-dependencies</artifactId>
                    <version>Camden.SR6</version>
                    <type>pom</type>
                    <scope>import</scope>
               </dependency>
          </dependencies>
     </dependencyManagement>

     <build>
          <plugins>
               <plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>
               </plugin>
          </plugins>
     </build>


</project>
```

Next define the following properties in application.properties-

zuul.routes.producer.url=http://localhost:8080
eureka.client.serviceUrl.defaultZone=http://localhost:8090/eureka
server.port=8079



Here zuul.routes.producer.url will route incoming traffic to request for /producer to the employee-producer microservice. Similar routes can be added for other microservices as well.

Next name the application module in the bootstrap.properties file
spring.application.name=employee-zuul-service

Next we define the 4 types of filters supported by Zuul-
pre
post
route
error
Define the ErrorFilter as follows-

```
package com.chandra.filter;

import com.netflix.zuul.ZuulFilter;

public class ErrorFilter extends ZuulFilter {
```

```
	@Override
	public String filterType() {
		return "error";
	}

	@Override
	public int filterOrder() {
		return 0;
	}

	@Override
	public boolean shouldFilter() {
		return true;
	}

	@Override
	public Object run() {
		System.out.println("Using Route Filter");

		return null;
	}

}
```

Ad by Value impression

Define the PostFilter as follows-

```
package com.chandra.filter;

import com.netflix.zuul.ZuulFilter;

public class PostFilter extends ZuulFilter {

	@Override
	public String filterType() {
		return "post";
	}
```

```
	@Override
	public int filterOrder() {
		return 0;
	}

	@Override
	public boolean shouldFilter() {
		return true;
	}

	@Override
	public Object run() {
		System.out.println("Using Post Filter");

		return null;
	}

}
```

Define the PreFilter as follows-

```
package com.chandra.filter;

import javax.servlet.http.HttpServletRequest;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;

public class PreFilter extends ZuulFilter {

	@Override
	public String filterType() {
		return "pre";
	}

	@Override
	public int filterOrder() {
```

```
		return 0;
	}

	@Override
	public boolean shouldFilter() {
		return true;
	}

	@Override
	public Object run() {
		RequestContext ctx = RequestContext.getCurrentContext();
		HttpServletRequest request = ctx.getRequest();

		System.out.println(
				"Request Method : " + request.getMethod() + "
Request URL : " + request.getRequestURL().toString());

		return null;
	}

}
```

Ad by Value impression

Define the RouteFilter as follows-

```
package com.chandra.filter;

import com.netflix.zuul.ZuulFilter;

public class RouteFilter extends ZuulFilter {

	@Override
	public String filterType() {
		return "route";
	}

	@Override
	public int filterOrder() {
```

```
		return 0;
	}

	@Override
	public boolean shouldFilter() {
		return true;
	}

	@Override
	public Object run() {
		System.out.println("Using Route Filter");

		return null;
	}

}
```

Finally we annotate the Spring Boot Main class with @EnableZuulProxy.With this the module will act as a service proxy or gateway.
Also we create the beans for the filters defined above.

```
package com.chandra;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;
import org.springframework.context.annotation.Bean;

import com.chandra.filter.ErrorFilter;
import com.chandra.filter.PostFilter;
import com.chandra.filter.PreFilter;
import com.chandra.filter.RouteFilter;

@SpringBootApplication
@EnableDiscoveryClient
@EnableZuulProxy
```

```
public class SpringBootHelloWorldApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringBootHelloWorldApplication.class,
args);
	}

	@Bean
	public PreFilter preFilter() {
		return new PreFilter();
	}

	@Bean
	public PostFilter postFilter() {
		return new PostFilter();
	}

	@Bean
	public ErrorFilter errorFilter() {
		return new ErrorFilter();
	}

	@Bean
	public RouteFilter routeFilter() {
		return new RouteFilter();
	}
}
```


Code changes for employee-consumer

The changes we make for the consumer module are
We fetch the Zuul Service instance instead of the Employee Producer service we were doing earlier.
So in code we have
discoveryClient.getInstances("EMPLOYEE-ZUUL-SERVICE") instead of discoveryClient.getInstances("EMPLOYEE-PRODUCER")

Append the URL to be hit with /producer since we have defined so in the applicatio.properties above.
baseUrl = baseUrl + "/producer/employee"

```
package com.chandra.controllers;

import java.io.IOException;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;

@Controller
public class ConsumerControllerClient {

	@Autowired
	private DiscoveryClient discoveryClient;

	public void getEmployee() throws RestClientException, IOException
{

		List<ServiceInstance> instances =
discoveryClient.getInstances("EMPLOYEE-ZUUL-SERVICE");
		ServiceInstance serviceInstance = instances.get(0);

		String baseUrl = serviceInstance.getUri().toString();

		baseUrl = baseUrl + "/producer/employee";
```

```
		RestTemplate restTemplate = new RestTemplate();
		ResponseEntity<String> response = null;
		try {
			response = restTemplate.exchange(baseUrl,
HttpMethod.GET, getHeaders(), String.class);
		} catch (Exception ex) {
			System.out.println(ex);
		}
		System.out.println(response.getBody());
	}

	private static HttpEntity<?> getHeaders() throws IOException {
		HttpHeaders headers = new HttpHeaders();
		headers.set("Accept",
MediaType.APPLICATION_JSON_VALUE);
		return new HttpEntity<>(headers);
	}
}
```

Ad by Value impression

As we had done in previous posts- Start the following Spring Boot Applications-

eureka-server

employee-producer

employee-zuul-service

Employee-consumer

On running the employee-consumer we get the output as follows-

```
2017-09-08 01:16:24.079  INFO 8316 --- [           main] c.j.SpringBootHelloWorldApplication      : Started SpringBootHelloWorldApplication in 8.574 seconds (JVM running for
com.javainuse.controllers.ConsumerControllerClient@153cd6bb
2017-09-08 01:16:24.096  INFO 8316 --- [           main] o.s.c.n.eureka.InstanceInfoFactory       : Setting initial instance status as: STARTING
2017-09-08 01:16:24.815  INFO 8316 --- [           main] c.n.d.provider.DiscoveryJerseyProvider   : Using JSON encoding codec LegacyJacksonJson
2017-09-08 01:16:24.815  INFO 8316 --- [           main] c.n.d.provider.DiscoveryJerseyProvider   : Using JSON decoding codec LegacyJacksonJson
2017-09-08 01:16:41.125  INFO 8316 --- [           main] c.n.d.provider.DiscoveryJerseyProvider   : Using XML encoding codec XStreamXml
2017-09-08 01:16:41.125  INFO 8316 --- [           main] c.n.d.provider.DiscoveryJerseyProvider   : Using XML decoding codec XStreamXml
2017-09-08 01:16:47.920  INFO 8316 --- [           main] c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
2017-09-08 01:16:56.598  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Disable delta property : false
2017-09-08 01:16:56.598  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Single vip registry refresh property : null
2017-09-08 01:16:56.598  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Force full registry fetch : false
2017-09-08 01:16:56.598  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Application is null : false
2017-09-08 01:16:56.598  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Registered Applications size is zero : true
2017-09-08 01:16:56.598  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Application version is -1: true
2017-09-08 01:16:56.598  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Getting all instance registry info from the eureka server
2017-09-08 01:17:03.369  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : The response status is 200
2017-09-08 01:17:03.369  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Starting heartbeat executor: renew interval is: 30
2017-09-08 01:17:03.369  INFO 8316 --- [           main] c.n.discovery.InstanceInfoReplicator     : InstanceInfoReplicator onDemand update allowed rate per min is 4
2017-09-08 01:17:03.919  INFO 8316 --- [           main] com.netflix.discovery.DiscoveryClient    : Discovery Client initialized at timestamp 1504813623385 with initial inst
{"empId":"1","name":"emp1","designation":"manager","salary":3000.0}
2017-09-08 01:17:33.633  INFO 8316 --- [tbeatExecutor-0] com.netflix.discovery.DiscoveryClient    : DiscoveryClient_EMPLOYEE-CONSUMER/192.168.1.140:employee-consumer:8091 -
2017-09-08 01:17:33.633  INFO 8316 --- [tbeatExecutor-0] com.netflix.discovery.DiscoveryClient    : DiscoveryClient_EMPLOYEE-CONSUMER/192.168.1.140:employee-consumer:8091: r
2017-09-08 01:17:33.681  INFO 8316 --- [tbeatExecutor-0] com.netflix.discovery.DiscoveryClient    : DiscoveryClient_EMPLOYEE-CONSUMER/192.168.1.140:employee-consumer:8091 -
```

The Zuul console output is as follows-

```
2017-08-09 00:26:39.343  INFO 6316 --- [nio-8079-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2017-08-09 00:26:39.344  INFO 6316 --- [nio-8079-exec-1] o.s.web.servlet.DispatcherServlet
2017-08-09 00:26:39.403  INFO 6316 --- [nio-8079-exec-1] o.s.web.servlet.DispatcherServlet
2017-08-09 00:26:39.475  INFO 6316 --- [nio-8079-exec-1] o.s.c.n.zuul.web.ZuulHandlerMapping
2017-08-09 00:26:39.476  INFO 6316 --- [nio-8079-exec-1] o.s.c.n.zuul.web.ZuulHandlerMapping
Request Method : GET Request URL : http://VAIO:8079/producer/employee
Inside Route Filter
Inside Response Filter
Request Method : GET Request URL : http://localhost:8079/producer/employee
Inside Route Filter
Inside Response Filter
2017-08-09 00:29:34.483  INFO 6316 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver
```