**MicroServices:**

Microservices are a modern approach to software whereby application code is delivered in small, manageable pieces, independent of others.

**Eureka Server:**

Eureka Service will register every microservice’s services and then microservice client will search for Eureka Server to get a dependent microservice to get the job done. Eureka Server is owned to Netflix and, Spring Cloud provides a declaration to register and call service by Java annotation.

**API Gateway:**

Imagine we are building a sales website and use micro service architecture, and we need to implement product detail page, so you need to build many UI, and we need to develop APIs for product for clients.

So we choose micro service to divide those web’s component into many small services.

*So the question is: How the clients can get access to each dependent services?*

Solution: we have to implement an API Gateway, create an entry point for all incoming request. API Gateway will handle all incoming request, redirect them to exact service which user desire.

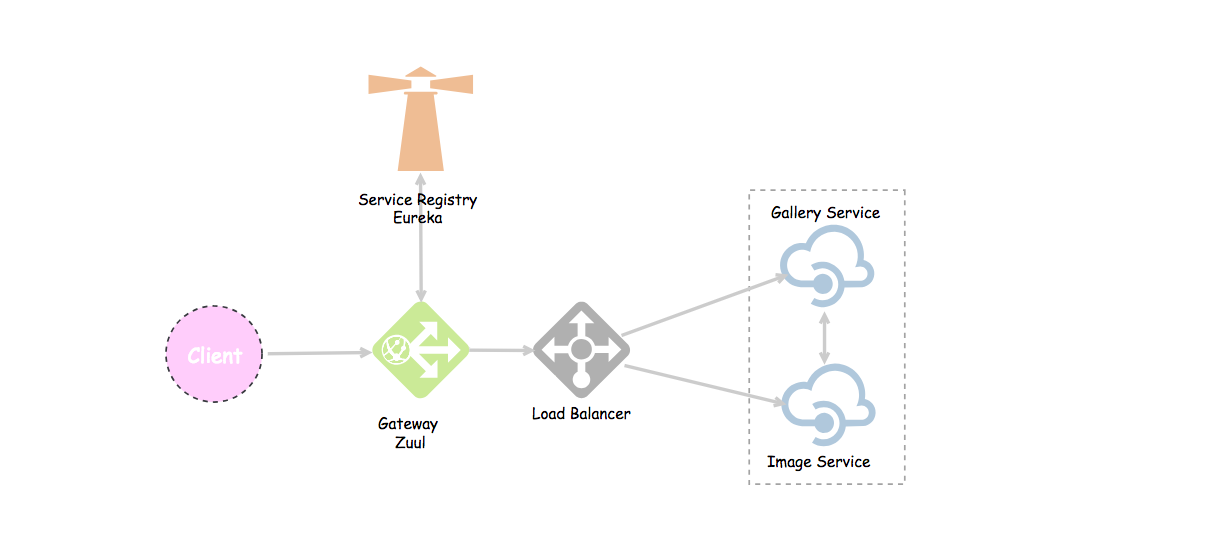
**Main features of API Gateway:**

* Routing: Using API Gateway similar to pack your application into whole, separate between client and application, all request from clients will go through only 1 entry point.
* Offloading: API Gateway provides features like Authentication, Authorization, rate limiting, load balancing, logging, tracing…

**Netflix API Gateway: Zuul**

Zuul is a proxy, gateway, a class between user and yours service. Eureka server has solved naming for each service instead of using their IP problem. But, a service still has many instance and they can run on different port so Zuul responsibility is:

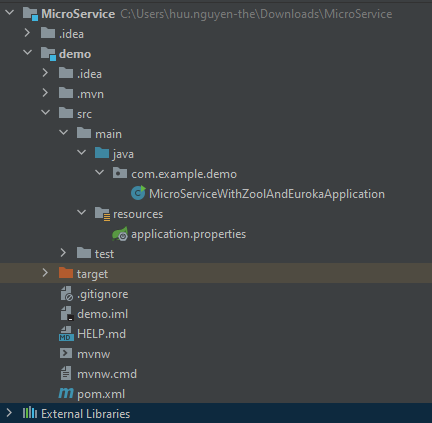
* Map a prefix path and a service. It uses Eureka to route requested services.
* It helps load balancer between instances of a service.
* It helps us filter request, authentication…



**Let jump to the demo:**

**1.Create Eureka Server:**

First, we create a spring boot project for Eureka Server.





Set up dependencies in *pom.xml*:

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  
 <version>3.1.2</version>  
 </dependency>

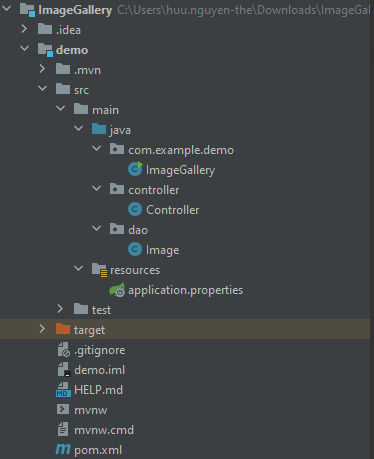
In Main class:

We will declare @EnableEurekaServer annotation to define it’s a EurekaServer.

@SpringBootApplication  
@EnableEurekaServer  
public class MicroServiceWithZoolAndEurokaApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(MicroServiceWithZoolAndEurokaApplication.class, args);  
 }  
  
}

**2.Create Image Service:**

The Eureka client service is an independent service in a microservice architecture. It could be for payment, account, notification, auth, config, etc.



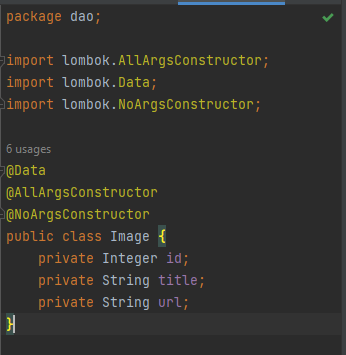
So, for the *pom.xml* file, instead of Eureka Server, use Eureka Client

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 <version>3.1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
</dependencies>

In *application.properties*:

# serivce name  
spring.application.name=image-service  
# port  
server.port=8200  
# eureka server url  
eureka.client.service-url.default-zone=http://localhost:8761/eureka

Now create the Image entity class with three fields; id, title, and url.



So, we need to create a controller, and define the action methods for our image service exposing some data through endpoints

import dao.Image;  
import java.util.Arrays;  
import java.util.List;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.core.env.Environment;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping("/")  
public class Controller {  
@Autowired  
 private Environment env;  
@RequestMapping("/")  
 public String home(){  
 return "Hello :"+env.getProperty("local.server.port");  
}  
@RequestMapping("/images")  
 public List<Image> getImages(){  
 List<Image> images= Arrays.*asList*(new Image(1, "Treehouse of Horror V", "https://www.imdb.com/title/tt0096697/mediaviewer/rm3842005760"),  
 new Image(2, "The Town", "https://www.imdb.com/title/tt0096697/mediaviewer/rm3698134272"),  
 new Image(3, "The Last Traction Hero", "https://www.imdb.com/title/tt0096697/mediaviewer/rm1445594112"));  
 return images;  
}  
}

Then, enable eureka client using @EnableEurekaClient annotation.

import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;  
  
@SpringBootApplication  
@EnableEurekaClient  
public class ImageGallery {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ImageGallery.class, args);  
 }  
  
}

**3. Create Gallery Service**

The Eureka Client service can be also a REST client that calls other services in our microservice application

The Gallery Service calls Image Service to get a list of all image, the calls from this REST client to other services can be done using RestTemplate or FeignClient.

Plus we use load balance to request across the services.

Load balancing is when more than one instance of a servie running on different ports , so we need to balance the requests among all the instances of a service.

First we set up pom.xml file:

<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 <version>3.1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
</dependencies>

Next is the application.properties file

spring.application.name=gallery-service  
server.port=8100  
eureka.client.service-url.default-zone=http://localhost:8761/eureka

Then in main class, we enable the eureka client and create a bean for RestTemplate co call the Image Service.

import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.loadbalancer.LoadBalanced;  
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.client.RestTemplate;  
  
@SpringBootApplication  
@EnableEurekaClient  
public class GalleryServiceApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(GalleryServiceApplication.class, args);  
 }  
  
}  
@Configuration  
class RestTemplateConfig {  
  
 // Create a bean for restTemplate to call services  
 @Bean  
 @LoadBalanced // Load balance between service instances running at different ports.  
 public RestTemplate restTemplate() {  
  
 return new RestTemplate();  
 }  
}

In the controller, call image service using *RestTemplate* and return the result.

import com.example.demo.dao.Gallery;  
import java.util.List;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.core.env.Environment;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
  
@RestController  
@RequestMapping("/")  
public class Controller {  
 @Autowired  
 private RestTemplate restTemplate;  
  
 @Autowired  
 private Environment env;  
  
 @RequestMapping("/")  
 public String home() {  
 return "Hello from Gallery Service running at port: " + env.getProperty("local.server.port");  
 }  
  
 @RequestMapping("/{id}")  
 public Gallery getGallery(@PathVariable final int id) {  
 // create gallery object  
 Gallery gallery = new Gallery();  
 gallery.setId(id);  
  
 // get list of available images  
 List<Object> images = restTemplate.getForObject("http://image-service/images/", List.class);  
 gallery.setImages(images);  
  
 return gallery;  
 }  
   
}

Since we are using *restTemplate* — which in turn uses Eureka Server for naming of services, and Ribbon for load balancing. So, we can use the service name (like image-service) instead of localhost:port

**4. Create Zuul API Gateway.**

In the *pom.xml*, we add dependencies: Web, Eureka Client, Zuul

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 <version>3.1.2</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-zuul</artifactId>  
 <version>2.2.10.RELEASE</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <optional>true</optional>  
</dependency>

We declare Zuul acts as a Eureka Client, so we give it a name, port and link to Eureka Server.

server.port=8762  
  
spring.application.name=zuul-server  
  
eureka.client.service-url.default-zone=http://localhost:8761/eureka/  
  
# A prefix that can added to beginning of all requests.  
  
#zuul.prefix=/api  
  
# Disable accessing services using service name (i.e. gallery-service).  
  
# They should be only accessed through the path defined below.  
  
zuul.ignored-services=\*  
  
# Map paths to services  
  
zuul.routes.gallery-service.path=/gallery/\*\*  
  
zuul.routes.gallery-service.service-id=gallery-service

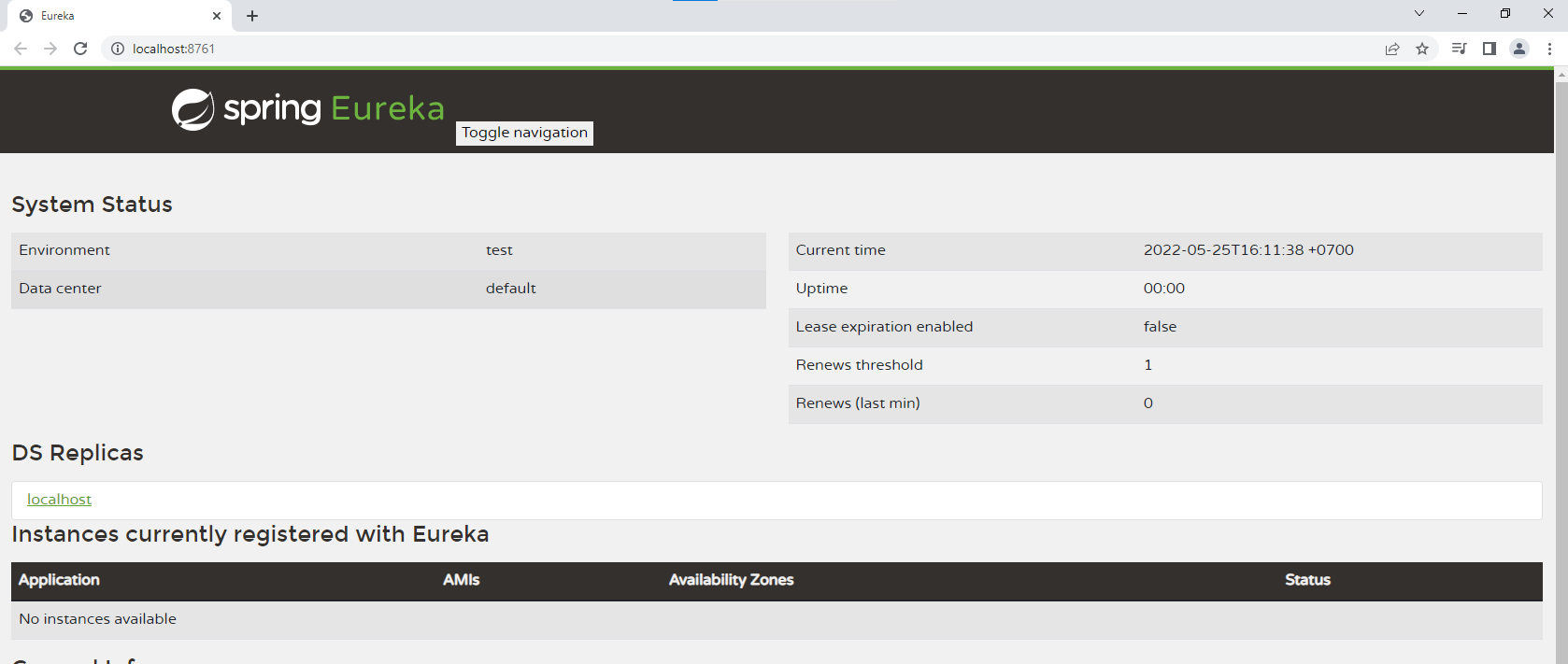
Finally, enable Zuul and Eureka Client in main class.

import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;  
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;  
  
@SpringBootApplication  
@EnableEurekaClient  
@EnableZuulProxy  
public class ZuulApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ZuulApplication.class, args);  
 }  
  
}

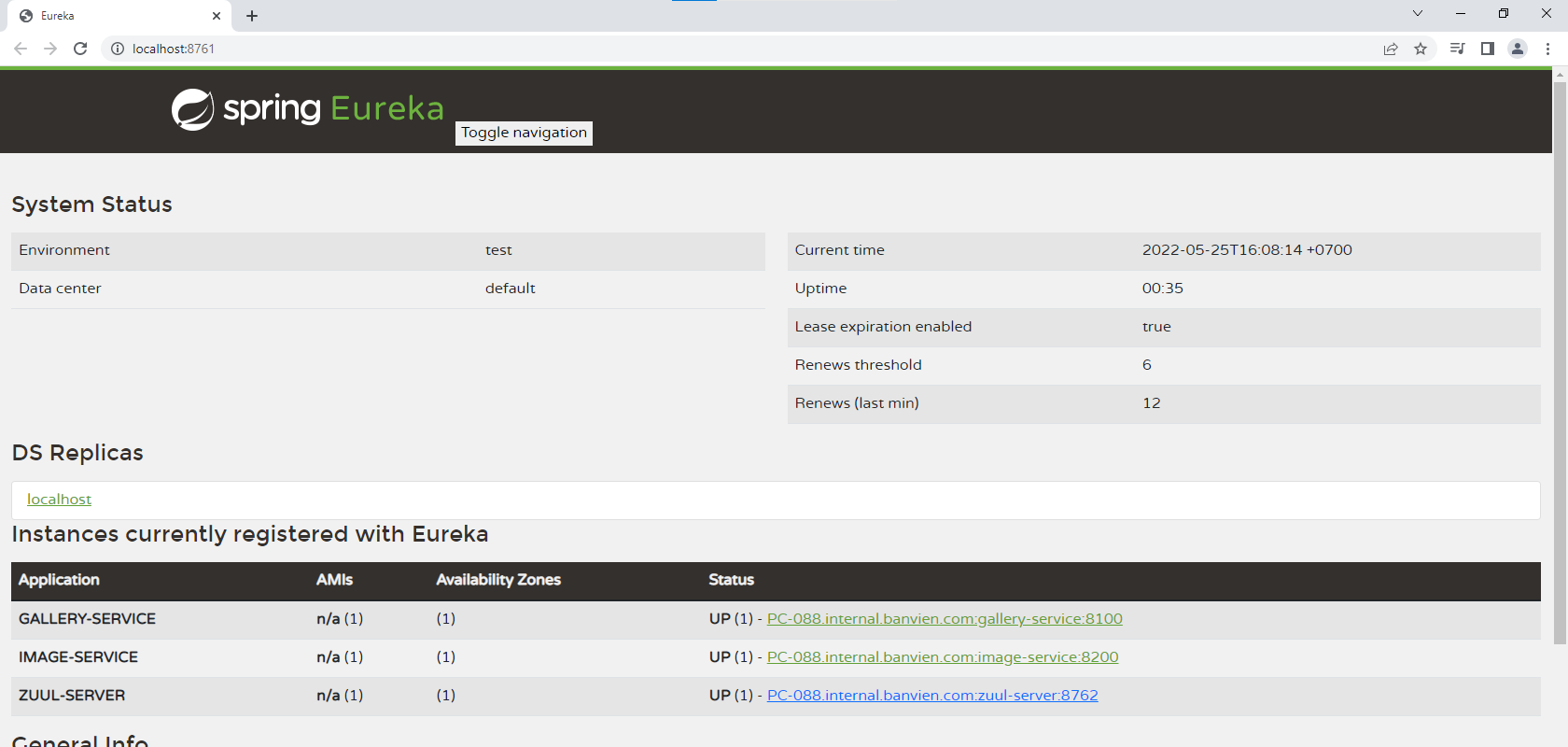
**Run our project:**

First we run Eureka Server,

Then go to localhost:8761



Then RZuul then two services (Image and Gallery).



Reference:

https://tubean.github.io/2018/12/microservice-springboot-eureka/