**Detailed MicroService Steps (Spring boot 1.5)**

I: Create Two Service: student + course

* Spring boot + Actuator/web/Rest Repositories/Eureka Discovery
* Write services “allstudents”, “allcourses”
* Test two services, COMMENT OUT “Netflix-eureka” dependency to prevent spring boot error

II. Register Services in Eureka

Question: Why do we need Eureka?

Answer:

1) To manage service addresses, give every service a name -not using server address

2) To check service status, heartbeat of all services in one place!

3) To auto register updated/restarted/new services

* Eureka needs a server and a client
* Server setup: create server - a spring boot application with “Actuator” and “Eureka Server”
* Server setup: add “@EnableEnurekaServer” on spring boot application
* Server setup: Add config to application.properties to setup port and disable registry for itself

server.port=8761  
  
# Eureka register all applications, here we disable registration for server itself  
eureka.client.register-with-eureka=**false**# Do not fetch registry for itself  
eureka.client.fetch-registry=**false**

* Client setup: add back the dependency in pom.xml
* Client setup: add @EnableEurekaClient to spring boot application
* Client setup: add properties to application.properties

server.port=8098  
  
# tell the client where is the location of the eureka server  
# defaultZone is the default location and default server name is /eureka/  
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/  
  
eureka.client.healthcheck.enabled=**true**

eureka.instance.lease-renewal-interval-in-seconds=1  
eureka.instance.lease-expiration-duration-in-seconds=2  
  
eureka.client.lease.duration=5

spring.application.name=course-service

* Refresh 8761 server to see if client is registered
* Stop one registered client, and wait a while and refresh eureka server, it should be gone

III. Hystrix Circuit Breaker – a Pattern

Question: When one service call another service, but the other service is down.

Answer: Hystrix circuit breaker – it catches all problems with underlying services and process a backup plan for it.

* Create method in course to register students with restTemplate ()

@GetMapping(value = "/register/{courseName}/{studentId}")  
public Course registerStudent(@PathVariable String courseName, @PathVariable int studentId){

* Make sure it works
* There are 4 steps to add Hystrix Circuit Breaker pattern
* Step1: add Hystrix Dependency

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

* Step2: add @EnableCircuitBreaker & @EnableHystrixDashboard annotation on course application class
* Step3: create a deletegate package, move restTemplate and registerStudents method there
* Step4: add @HystrixCommand(fallback=”fallbackmethod”)

IV. Zuul API Gateway

Question: every since service is exposed to user. We need to 1) open one root to customer 2)filter request

Answer: Use Zuul API Gateway – Zuul can filter/intercept request url and also redirect url to each service

* Create spring boot application : mspoc-zuul-gateway with “Zuul” + “Eureka Discovery”
* Import into workspace as module
* Annotate application with “@EnableZuulProxy” and “@EnableDiscoveryClient”
* Add to application.properties with path configuration

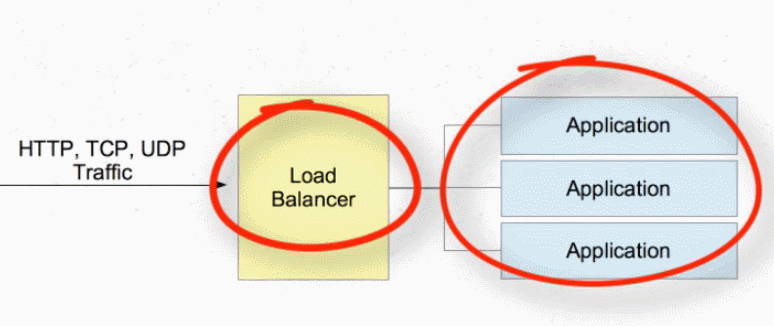
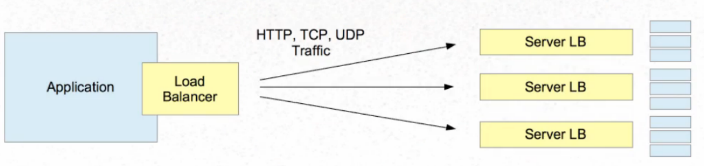
server.port=2020  
  
# donot need to register zuul proxy itself, just want services registered in eureka  
eureka.client.register-with-eureka=**false**# we need to fetch registry so that we know all services registered in eureka  
eureka.client.fetch-registry=**true**# for student-service  
# if below "path" if not defined, the service-id will auto pathed to "student-service"  
# i.e. student-service is the default path  
zuul.routes.student-service.path=/student-api/\*\*  
zuul.routes.student-service.service-id=student-service  
  
# for course-service  
zuul.routes.course-service.path=/course-api/\*\*  
zuul.routes.course-service.service-id=course-service  
  
# eureka server  
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

* Test with only port 2020

V. Ribbon Load Balancer

Question: When there are too many requests, we need to setup replica and we need to distribute the requests to different replica.

V1) Client-side load balancer vs Server-side load balancer

 vs 

V2) Load balancer algorithms

(**round robin** – next available in round order

**availability filtering** – round robin but skip failed service and heavy-connected service

**weighted response time** – given every service a weight based its response time, and pick up a service based its weight)

* In student service controller, write a getmapping to return port
* In course service pom.xml, add ribbon dependency

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>  
</dependency>

* Annotate course service with below (name must equal to serviceId of service to be called):

@RibbonClient(name="student-service", configuration = RibbonConfiguration.class)

* In application.properties, add below config:

# if not using service discovery, we have to set below (start with "student-service")  
student-service.ribbon.eureka.enabled=true  
student-service.ribbon.ServerListRefreshInterval=1000  
student-service.ribbon.listOfServers=localhost:9090,localhost:9091

* In student controller, add a new mapping for ribbon call which will return current server port
* In course controller, call delegate and then to student ribbon service
* Create a config.RibbonConfig class and setup the config rule (load balancer rule)
* Now add <packaging>jar</packaging> in student pom.xml and run maven install to build jar
* Start different instance of student service by

Java -jar \target\\*jar –server.port=9001,9002 ect

* Make sure they all started in eureka and in browser, call /ribbonClient