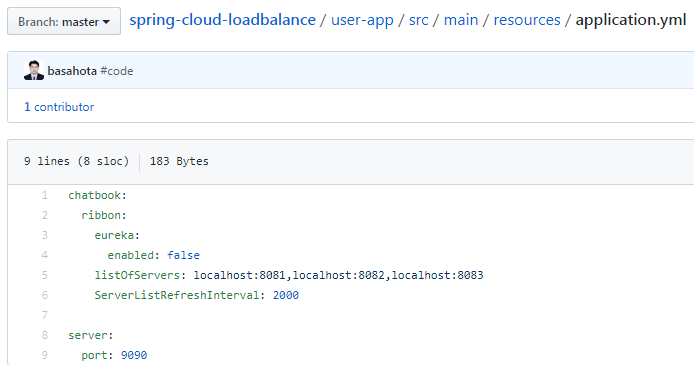
**Client side Load Balancing in Microservice :**

**Method 1 Using Ribbon :**

<https://github.com/Java-Techie-jt/spring-cloud-loadbalance>

We have two services chatbook (which has 3 instances deployed on **8081, 8082, 8083**) and user-app deployed on 9090

So in application.yml file of user-app service will be like :



And the main file will be like

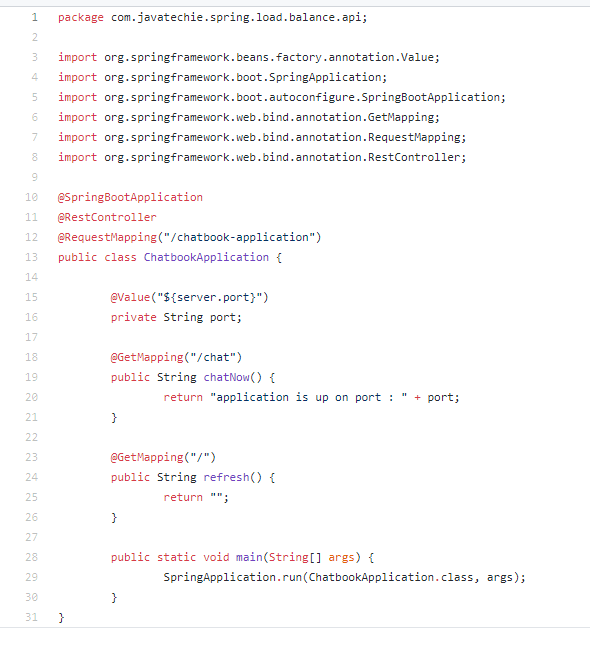
Put @RibbonCilent(name = “chatbook”, configuration = RibbonConfiguration.class)



In RibbonConfiguration.java -> we create the bean Iping and IRule



In Chatbook Service :



Start this service three time with changing the server.port value as 8081 ,8082, and 8083.

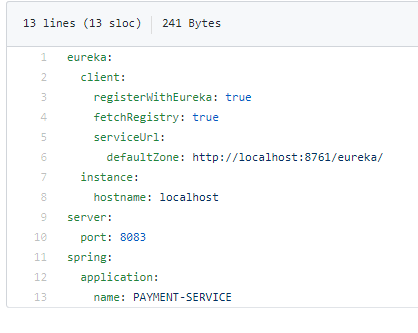
Once we hit the url from client side(user-app) this url will hit one by one instance of chatbook service

Method 2 Load Balancing Using Eureka :

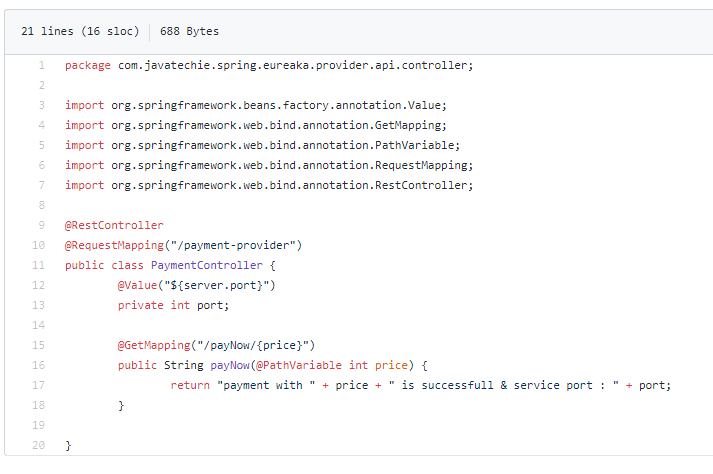
<https://github.com/Java-Techie-jt/spring-cloud-loadbalance-eureka>

Step 1: create an Eureka server to register all other services

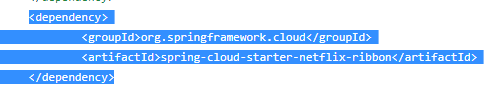
Step 2: Create a service named Payment-service as yml



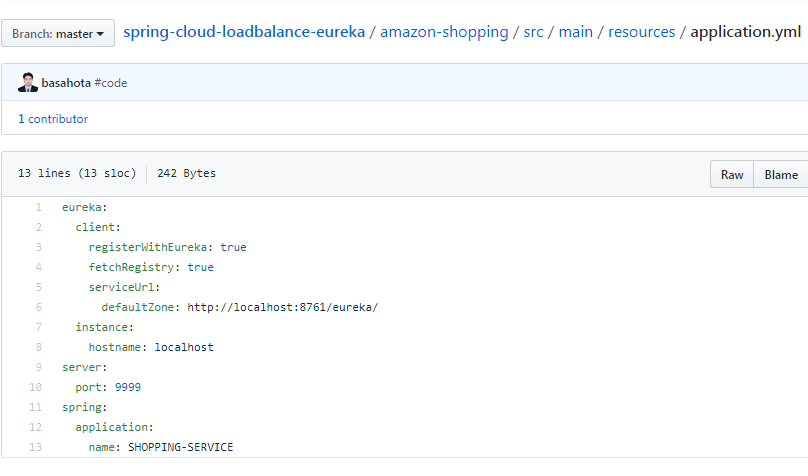
Step 3: Controller class will be



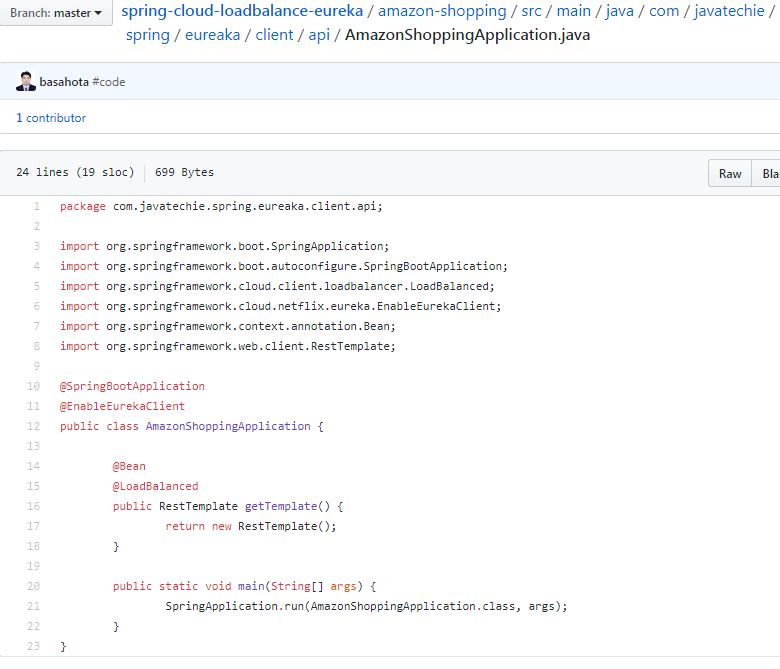
Step 4: We start this service on three port 8081,8082,8083

Step 5 : Create a service for Client as amazon-shoping with providing the dependency of Netflix-ribbon

Step 6 : yml of client will be like



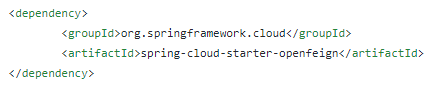
Step 7 : Main file of Client service : we have no need to configure IPing and IRule bean eureka handles these all things automatically

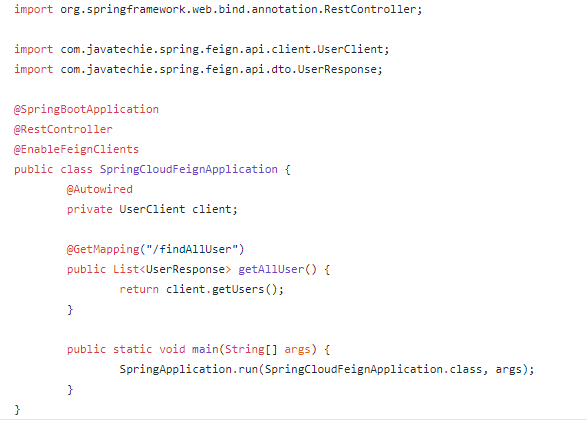


Step 8 : Client Controller class



[**spring-cloud-feign**](https://github.com/Java-Techie-jt/spring-cloud-feign): This approach is mainly used to work on data existing on cloud. Here client is used with Feign client and Server will be the service having data existing on cloud.

Step 1 : Put dependency of feign client

Step 2 : Main class will be annoted with Enabling Feign client 

Step 3 : Create a class UserResponse having dependent object as DTO

And its getters and setters

Step 3 : we create an interface with @FeignClient

