NETFLIX
RIBBON

Ribbon is a client-side load balancer that gives you a lot of control over the behaviour of HTTP and TCP clients. Feign already uses Ribbon, so, if you use @FeignClient, this section also applies.

A central concept in Ribbon is that of the named client. Each load balancer is part of an ensemble of components that work together to contact a remote server on demand, and the ensemble has a name that you give it as an application developer (for example, by using the @FeignClient annotation). On demand, Spring Cloud creates a new ensemble as an ApplicationContext for each named client by using RibbonClientConfiguration. This contains (amongst other things) an ILoadBalancer, a RestClient, and a ServerListFilter.

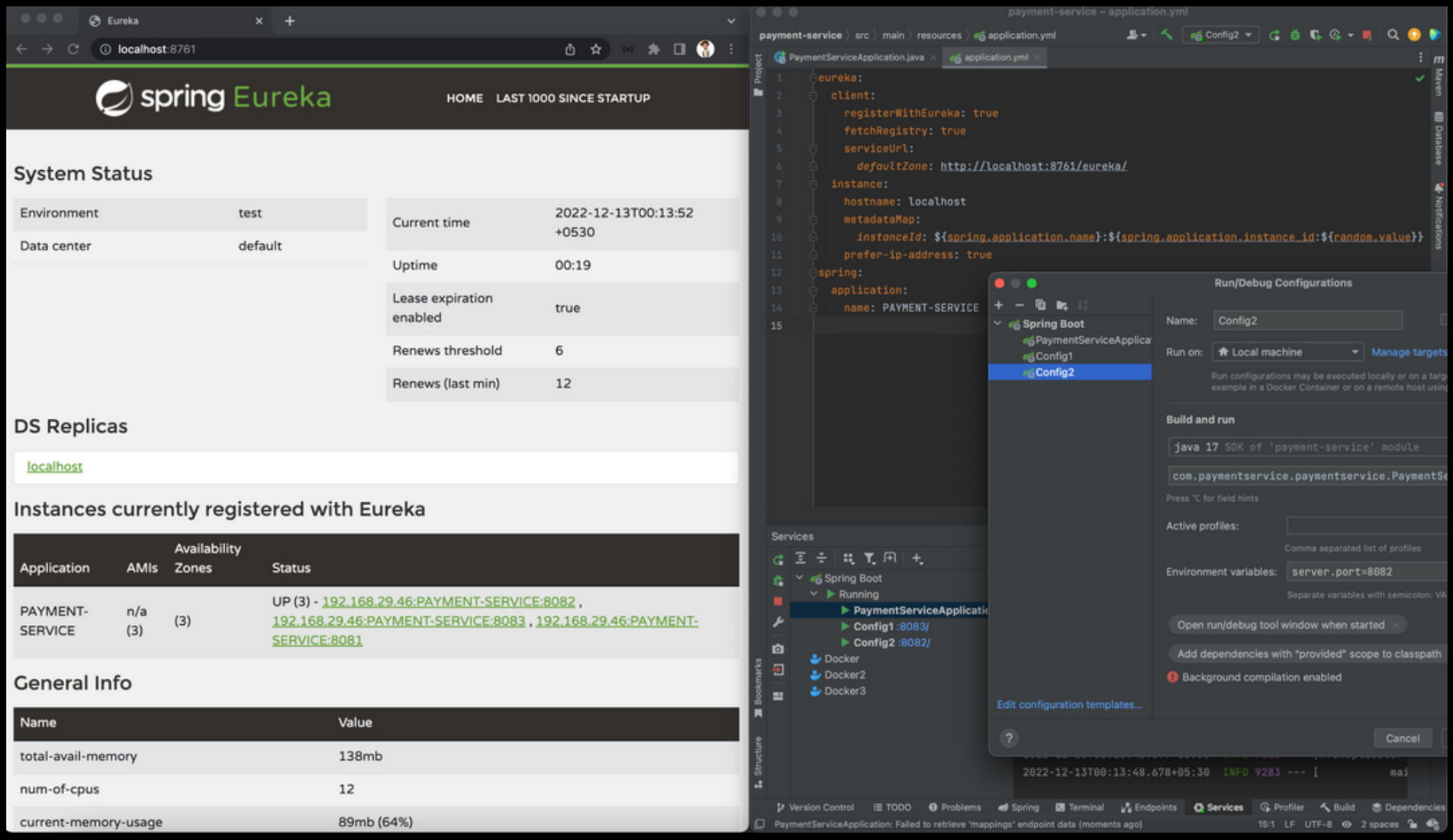
To include Ribbon in your project, use the starter with a group ID of **org.springframework.cloud** and an artifact ID of **spring-cloud-starter-netflix-ribbon**.

Note :

```
@LoadBalanced
@Bean
public RestTemplate getTemplate() { return new RestTemplate(); }
```

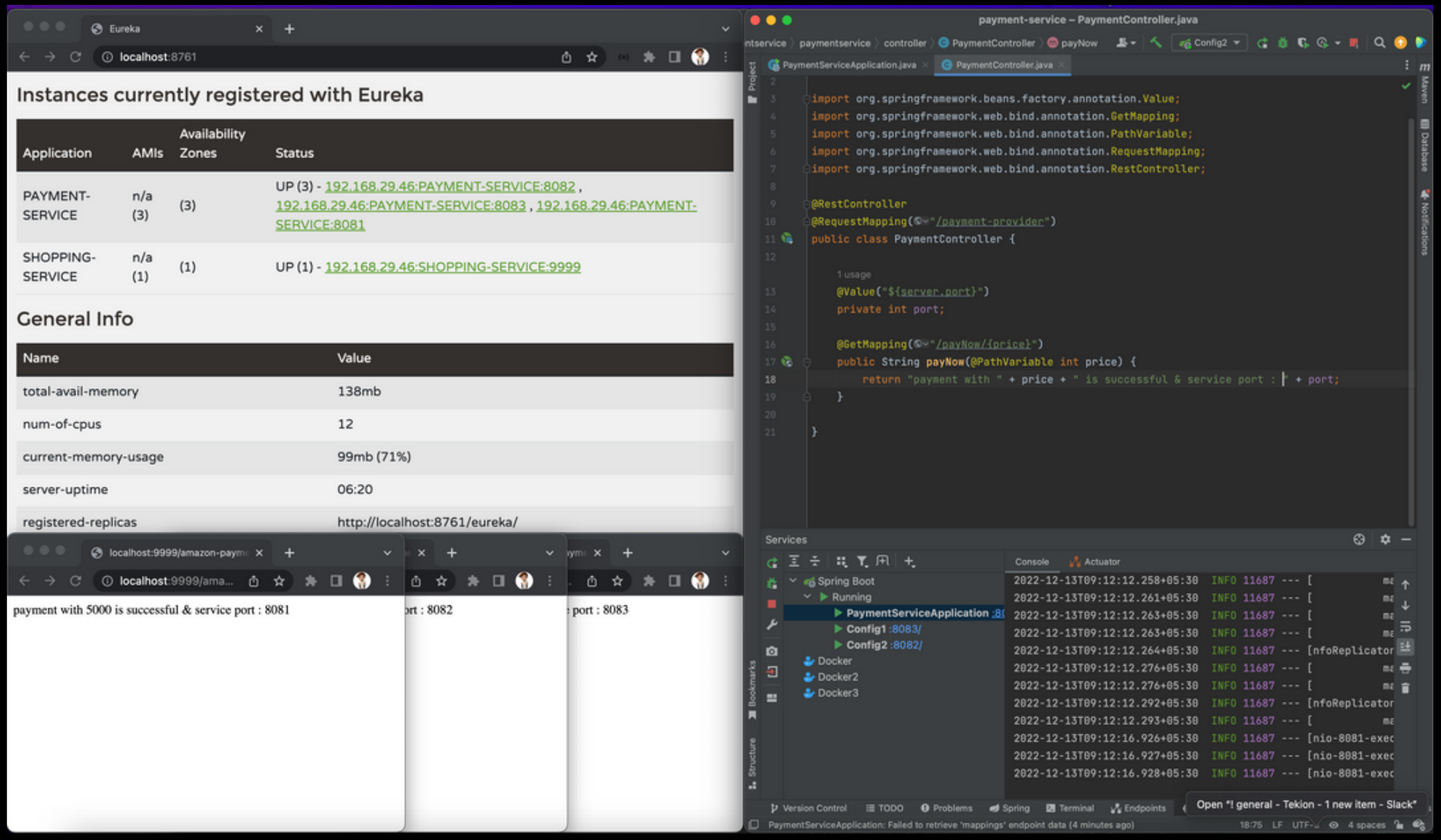
A load balanced RestTemplate can be configured to retry failed requests. By default this logic is disabled, you can enable it by adding Spring Retry to your application's class-path. The load balanced RestTemplate will honour some of the Ribbon configuration values related to retrying failed requests. Multiple resources can access our micro-services so better to annotate with **@LoadBalanced**

Running Multiple Instances Of Eureka Client Simultaneously



Step 1 : Select **Run/Debug Configuration**
Step 2 : Using the plus icon , select **Spring Boot** option
Step 3 : Using **Modify Options** , select **Environment variables** and add **server.port=8081** there
Step 4 : Repeat above steps for port(s) **8082** and **8083**

Load Balancing Using Spring Cloud Ribbon



We can observe here that Ribbon internally handles the load for 3 instances of payment service and one instance of shopping service.

Shopping service can make use of various instances of **Payment service** randomly.

Note:
Spring Cloud also lets us take full control of the ribbon client by declaring additional configuration (on top of the **RibbonClientConfiguration**) using **@RibbonClient**