Let’s talk about **Spring Cloud LoadBalancer** — this is the **modern replacement** for Netflix Ribbon in Spring Cloud.  
It’s **lightweight**, **actively maintained**, and works seamlessly with **Eureka**, **Feign**, and **RestTemplate**.

**🧠 1️⃣ What is Spring Cloud LoadBalancer?**

Spring Cloud LoadBalancer is a **client-side load balancer** provided by Spring Cloud itself (not Netflix).  
When a microservice calls another service by **service name** instead of hostname, LoadBalancer chooses one of the available instances to forward the request to.

📝 Think of it as:

“Ribbon → ❌ Deprecated”  
“Spring Cloud LoadBalancer → ✅ Replacement”

**🏗 2️⃣ Typical Architecture**

currency-conversion-service

|

| (Spring Cloud LoadBalancer)

v

Eureka Service Registry ----> multiple instances of currency-exchange-service

* The **service name** is resolved using **Eureka**.
* The **LoadBalancer** then decides which instance to call (default = round robin).

**⚙️ 3️⃣ Add Dependencies**

If you’re using **Spring Cloud 2020+**, you probably already have it.  
But here’s the explicit dependency:

<dependency>

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

<artifactId>spring-cloud-starter-loadbalancer</artifactId>

</dependency>

If you were previously using Ribbon, **remove**:

<dependency>

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

<artifactId>spring-cloud-starter-netflix-ribbon</artifactId>

</dependency>

**📝 4️⃣ RestTemplate with LoadBalancer**

Create a **@LoadBalanced RestTemplate bean**:

@Configuration

public class RestTemplateConfig {

@Bean

@LoadBalanced // 👈 enables service name resolution + load balancing

public RestTemplate restTemplate() {

return new RestTemplate();

}

}

Then you can make calls like:

@Autowired

private RestTemplate restTemplate;

public String convertCurrency() {

String response = restTemplate.getForObject(

"http://currency-exchange-service/exchange/from/USD/to/INR",

String.class

);

return response;

}

👉 currency-exchange-service here is **not a hostname** — LoadBalancer uses **Eureka** to find instances.

**🧭 5️⃣ Feign + LoadBalancer**

Feign now **uses Spring Cloud LoadBalancer by default** (since Ribbon is gone):

@FeignClient(name = "currency-exchange-service")

public interface CurrencyExchangeProxy {

@GetMapping("/exchange/from/{from}/to/{to}")

ExchangeValue getExchangeValue(@PathVariable String from, @PathVariable String to);

}

* No need for url = "http://localhost:8000" in the annotation.
* LoadBalancer + Eureka will handle it.

**⚡ 6️⃣ Configuration Example (application.yml)**

spring:

application:

name: currency-conversion-service

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka/

**Currency Exchange Service** also registers with Eureka:

spring:

application:

name: currency-exchange-service

server:

port: 8000

Start multiple instances of exchange service (e.g., ports 8000 & 8001), both will be registered in Eureka.

When you call:

restTemplate.getForObject("http://currency-exchange-service/exchange/from/USD/to/INR", ...)

Spring Cloud LoadBalancer will **round robin** between 8000 and 8001 automatically ✅

**🧠 7️⃣ Customizing Load Balancing Strategy**

By default, LoadBalancer uses **RoundRobin**.  
You can plug in a custom strategy by creating a bean of type ReactorServiceInstanceLoadBalancer.

Example: **Random Strategy**

@Configuration

public class LoadBalancerConfiguration {

@Bean

public ReactorServiceInstanceLoadBalancer randomLoadBalancer(

Environment environment,

LoadBalancerClientFactory loadBalancerClientFactory) {

String name = environment.getProperty(LoadBalancerClientFactory.PROPERTY\_NAME);

return new RandomLoadBalancer(

loadBalancerClientFactory.getLazyProvider(name, ServiceInstanceListSupplier.class),

name

);

}

}

Then annotate your main app with:

@LoadBalancerClient(name = "currency-exchange-service", configuration = LoadBalancerConfiguration.class)

**✅ 8️⃣ Summary Table**

| **Feature** | **Ribbon 🧭** | **Spring Cloud LoadBalancer ⚡** |
| --- | --- | --- |
| Maintainer | Netflix (deprecated) | Spring Cloud (active) |
| Default Algorithm | Round Robin | Round Robin |
| Eureka Integration | ✅ | ✅ |
| Feign Integration | Implicit | Implicit |
| Custom Strategy | Via IRule | Via custom LoadBalancer bean |
| Status | ❌ Deprecated | ✅ Recommended |

**🚀 Quick Example**

**CurrencyConversionService → CurrencyExchangeService**

@RestController

public class CurrencyConversionController {

@Autowired

private RestTemplate restTemplate;

@GetMapping("/convert/from/{from}/to/{to}")

public String convert(@PathVariable String from, @PathVariable String to) {

return restTemplate.getForObject(

"http://currency-exchange-service/exchange/from/{from}/to/{to}",

String.class,

from,

to

);

}

}

Start:

* Eureka Server (8761)
* currency-exchange-service on 8000 and 8001
* currency-conversion-service on 8100

Access:

http://localhost:8100/convert/from/USD/to/INR

Each call will hit different exchange instances in a round robin way ✅