https://github.com/shabbir1903/SpringRibbonExample/

1.Start Eureka Server - <http://localhost:8761/>

2.Start ribbon Server - <http://localhost:9090/backend>

 3.Start ribbon client - <http://localhost:8888/client/frontend>

To Test Ribbon Client start multiple instance of ribbon server by changing the port number in the application properties and then run ribbon client You will see port number is changing in the UI of backend server by hitting <http://localhost:8888/client/frontend>

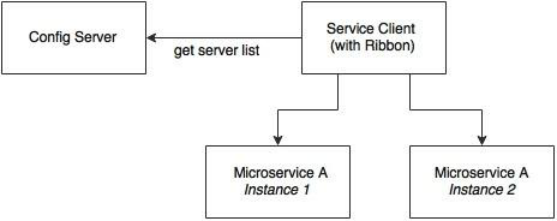
**Ribbon For Load Balancing**

So far, we were always running with a single instance of the microservice.The URL is hard coded both in client as well as in the service-to-service calls.Since there could be more than one service instance in the real world, this is not a recommended approach.

If there are multiple instances, then ideally, we should use a load balancer or a local DNS server to abstract the actual instance locations, and configure an alias name or the load balancer address in the clients.

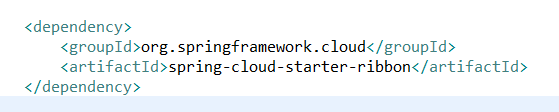
The load balancer then receives the alias name, and resolves it with one of the available instances. With this approach, we can configure as many instances behind a load balancer. It also helps us to handle server failures transparent to the client.

This is achievable with **Spring Cloud Netflix Ribbon**. **Ribbon is a client-side load balancer which can do round-robin load balancing across a set of servers**. There could be other load balancing algorithms possible with the Ribbon library. Spring Cloud offers a declarative way to configure and use the Ribbon client.



As shown in the preceding diagram, the Ribbon client looks for the Config server to get the list of available microservice instances, and, by default, applies a round-robin load balancing algorithm.

**Step-1:** In order to use the Ribbon client, we will have to add the following dependency to the pom.xml file:



In case of development from ground up, this can be selected from the Spring Starter libraries, or from **http://start.spring.io/**. Ribbon is available under Cloud Routing:

**@RibbonClient** annotation to enable client side load balancing.

Now Make sure to add the **listOfServers** property with this name I.e.**<name of the feign client>.ribbon.lostOfServers**. Hence In our case it should be like **patient-proxy.ribbon.listOfServers=localhost:9091,localhost:9090**