**SERVICE DISCOVERY**

In a typical microservice architecture we have many small applications deployed separately and they often need to communicate with each other. Specifically, when we say client service, we mean a service that needs to make REST calls to some other end service.

The problem in this type of architecture is how the client service finds all of its end services. We could hardcode the hostname/port in some property file, but this isn't always practical or feasible in a cloud environment. There could be any number of microservices, and it's time and resource-consuming to hard-code when there's an uncertain amount of them, and when their locations may change.

**Netflix Eureka**

**Netflix Eureka** is a lookup server (also called a registry). All the microservices in the cluster register themselves to this server.

When making a REST call to another service, instead of providing a hostname and port, they just provide the service name.

The actual routing is done at runtime along with equally distributing the load among the end services

To make a Eureka server, all we need to do is add the @EnableEurekaServer annotation to our main class:

<http://localhost:8761/>

To make this a client, all we need to do is add the @EnableEurekaClient annotation on the class level:

**After creating Client side controller that returns a list , we again register a controller that behaves and registers as a Eureka Client to get discovered**

Client-side load balancing decides which instance (in case of multiple end service running in the cluster that client can call) to call.

Above, we @Autowired the Ribbon-enabled RestTemplate and use it to call the movie-service. Note that we do not have to specify the hostname or the port anywhere.

With that, we've registered our service to the server, and any other registered service can call it using spring.application.name.

**Basically , in Microservices Service discovery concept , we consume a microservice from a set of given**

**Microservices via the “Service Name” and not hardcoded host – name or port number nor even any kind of base url strategy for the same .**

**Service name in Spring Boot is given using following annotation :**

spring.application.name.

We also need to create a RestTemplate bean and mark it as @LoadBalanced. This tells Spring that we want to take advantage of client-side load balancing, which is in this case done by [Ribbon](https://spring.io/guides/gs/client-side-load-balancing/).