wavelabs

Service Registry and Discovery

with Eureka and Spring Cloud

-Tharun kumar Bairoju

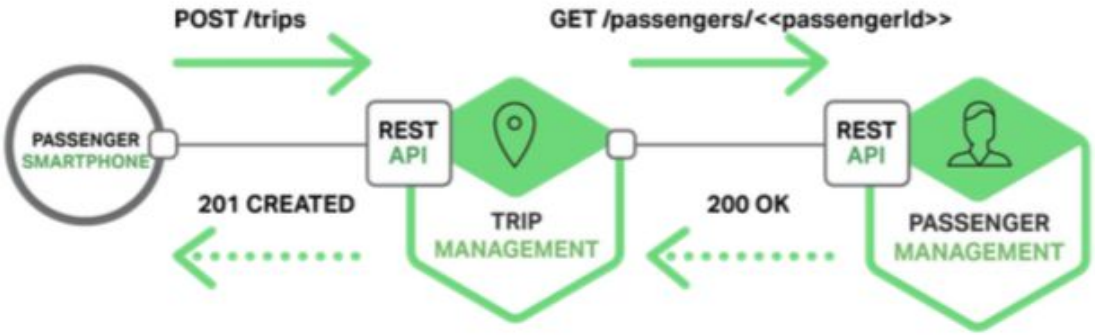
Agenda

- Traditional vs Modern application
- Communication between services
- Service discovery
- Service registry
- Eureka and Spring Cloud
- Sample snippets

wavelabs

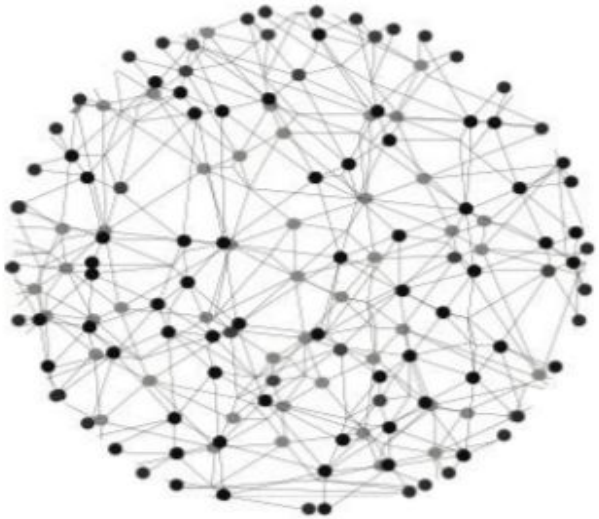
Traditional application

To perform communication between services we need know the location of the service(port, host). In traditional applications it's a simple task because services run in a fixed and known location.



Modern application

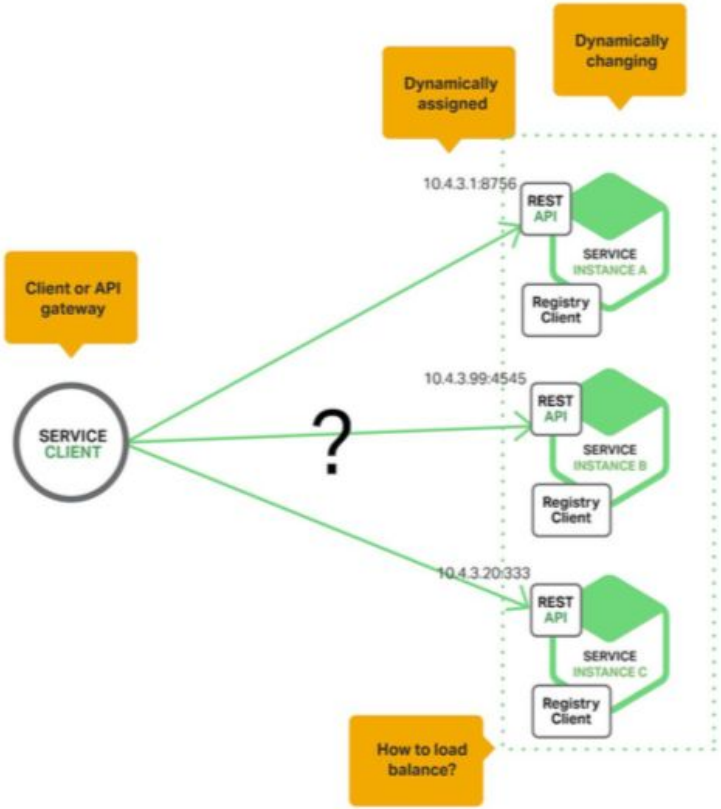
In modern applications the services are running in a dynamic environment. A service can have N instances running in N different machines. In this case, to know host and port of each service is very painful.



wavelabs

Discovery problem:

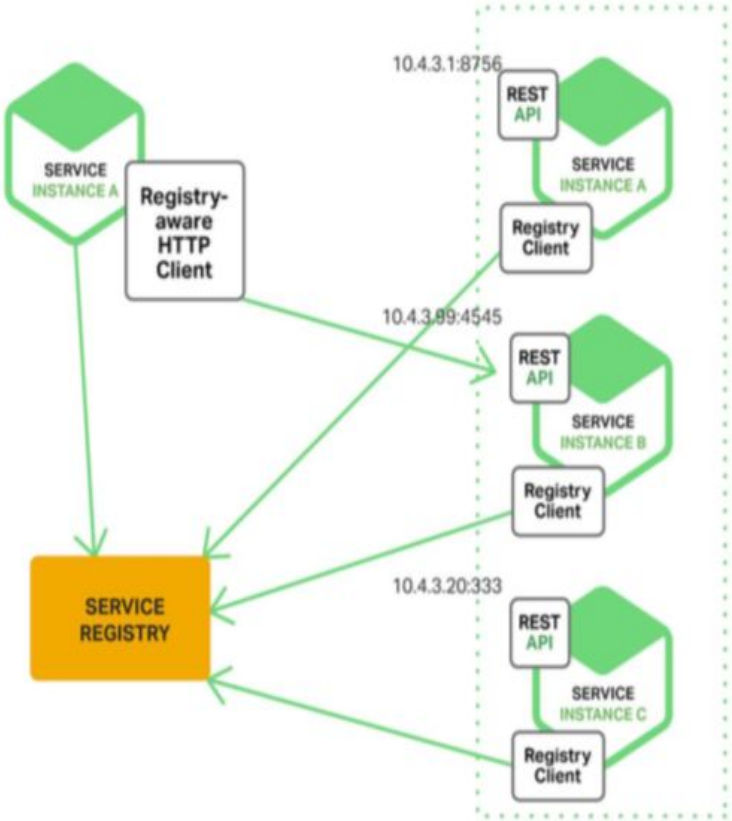
Maintain individual Microservices addresses. This task can be hugely complex – depending on number of services and their dynamic nature. If whole infrastructure is distributed and there is some replication as well, then maintaining this service addresses becomes harder.



wavelabs

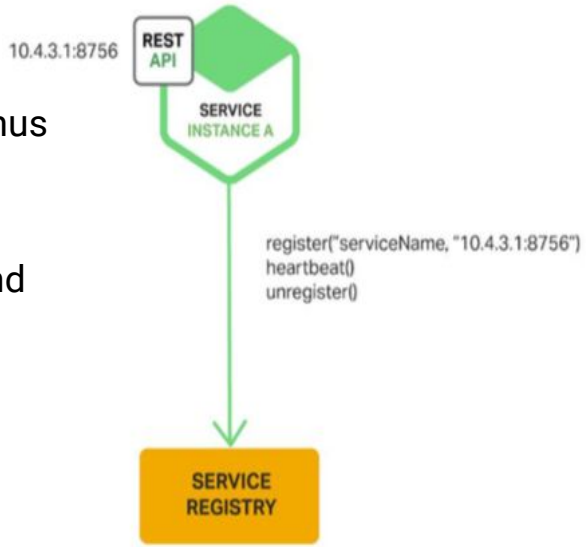
Service discovery

“**Service registration and discovery**” where one dedicated server is responsible to maintain the registry of all the Microservice that has been deployed and removed. This will act like a phone book of all other applications/microservices.

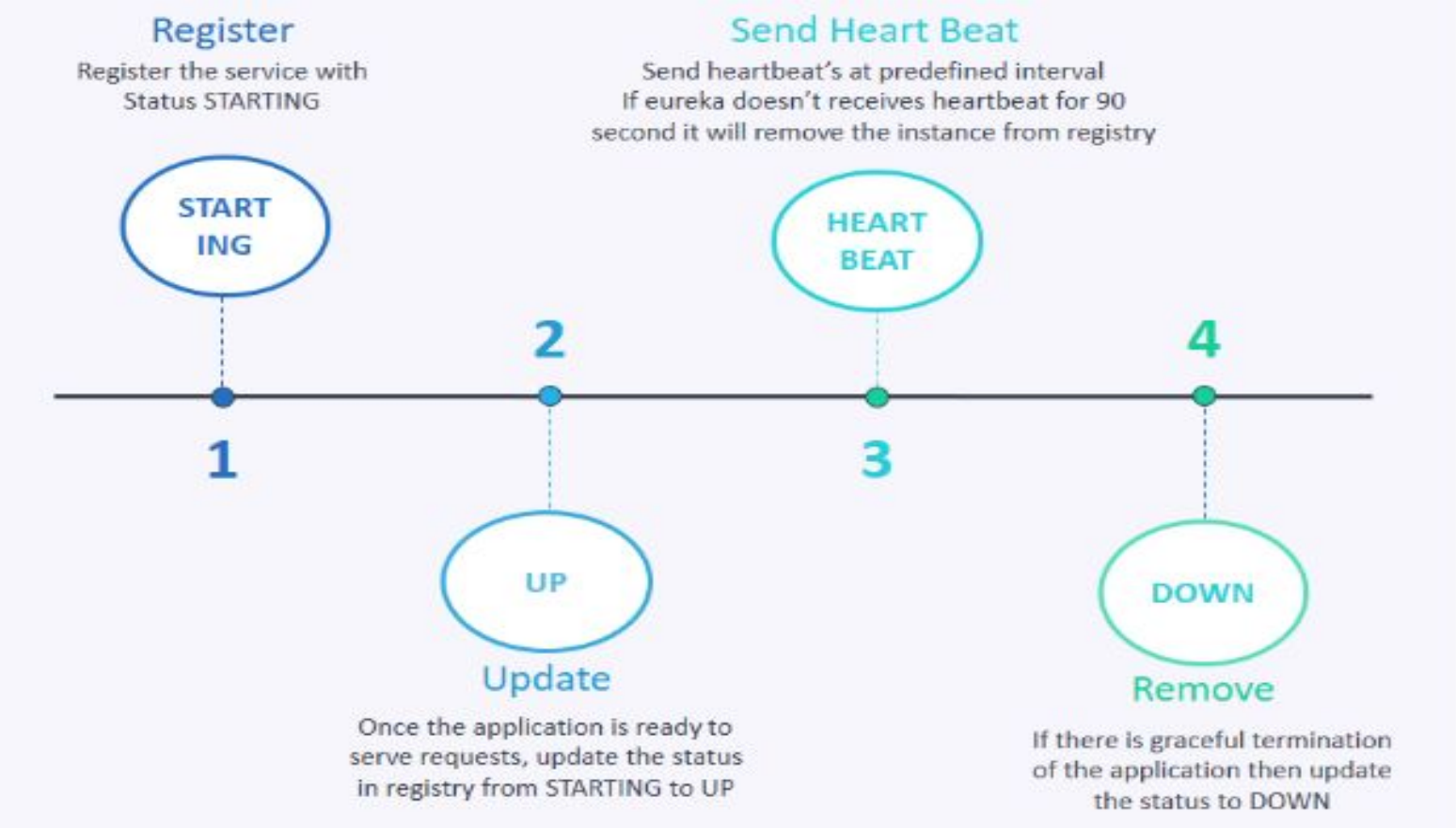


Service registry

- Microservices (clients) can register themselves and discover other registered microservices. When a client microservice registers with Eureka it provides metadata such as host, port, and health indicator thus allowing for other microservices to discover it.
- The discovery server expects a regular heartbeat message from each microservice instance. If an instance begins to consistently fail to send a heartbeat, the discovery server will remove the instance from his registry.
- This way we will have a very stable ecosystem of Microservices collaborating among each other, and on top of it we don't have to manually maintain address of other Microservice, which is a next to impossible task if the scale up/down is very frequent, on demand and we use virtual host to host the services specially in the cloud environment.



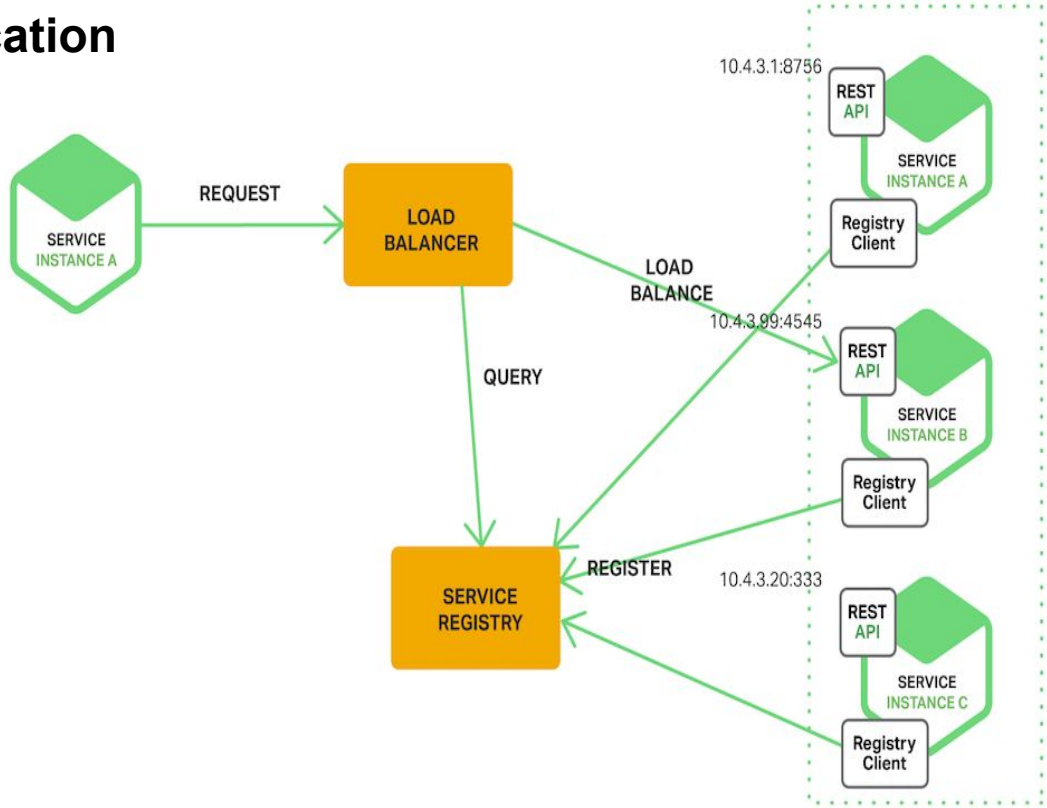
What registry will do?



wavelabs

Register and Discovery communication

1. Microservice will register on the service registry.
2. When request goes to loadbalancer, and it will talk to registry regarding address of the microservices.
3. Registry will give the addresses and port details, and load balancer is responsible to decide which instance need to call based on the load on the microservices

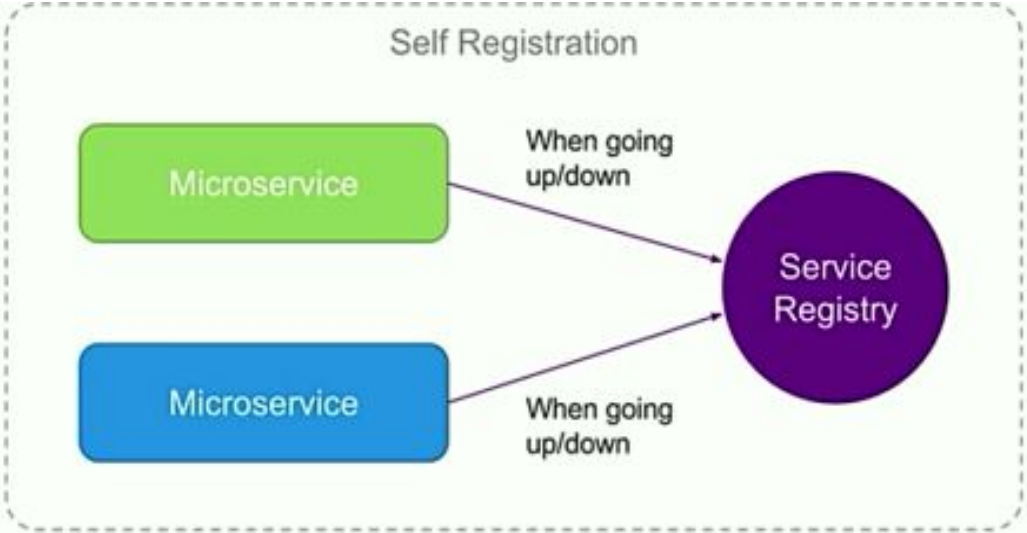


Service registry patterns

- Self registration
- Third party registration
- Client side registration
- Server side registration

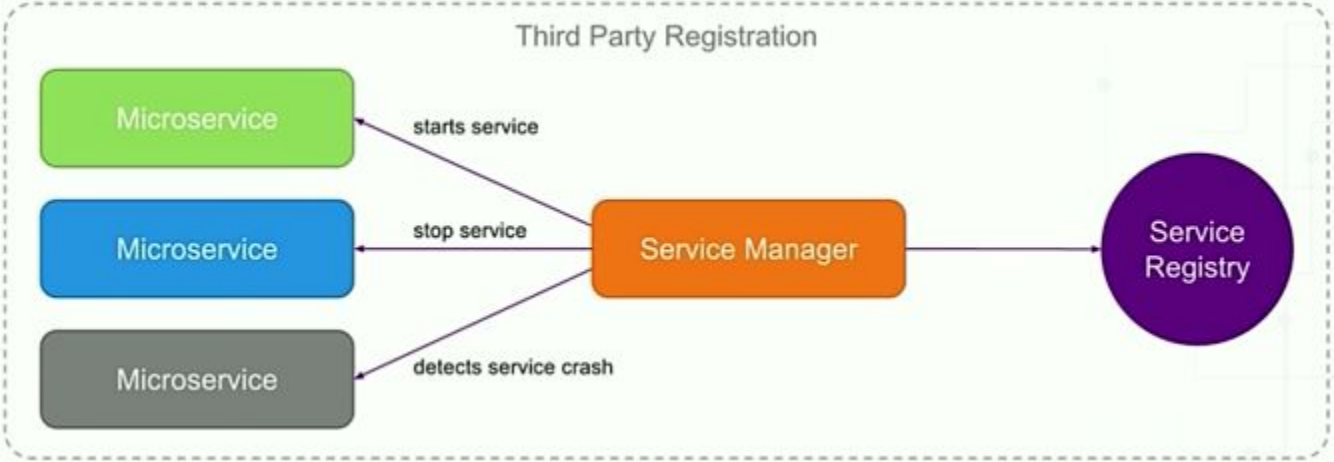
Self registration

- Microservice is responsible for registering the service details on the service registry.
- It needs to intimate registry whenever going up/down



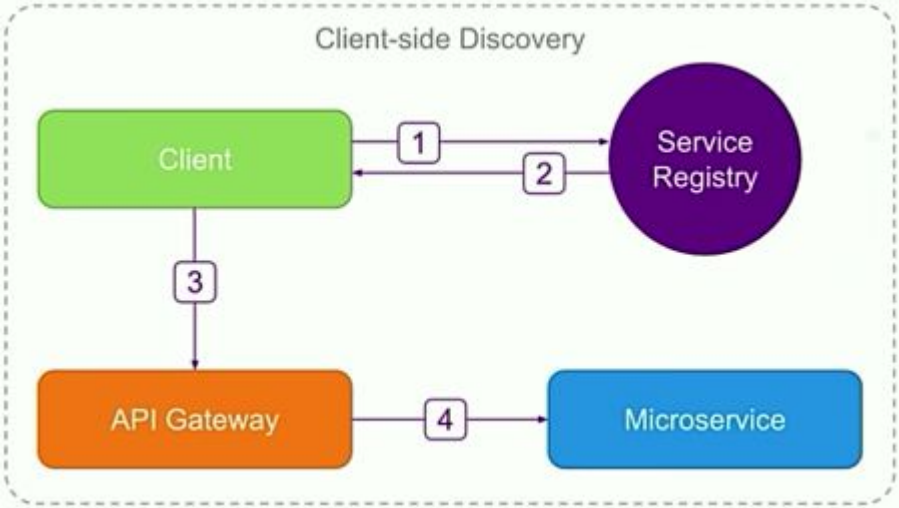
Third party registration

There should be a third party service manager, which is responsible for start service, stop service and detects service crash.



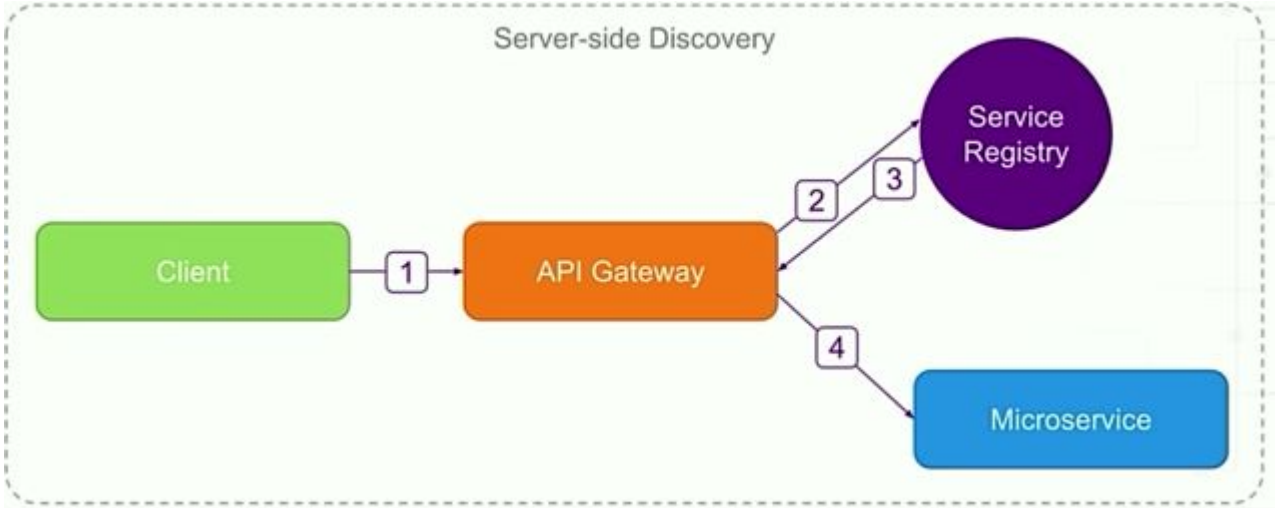
Client side discovery

**Client is responsible to check the service status



Server side discovery

Gateway is responsible to call the service registry and make the call



Eureka…

"Eureka is a REST (Representational State Transfer) based service that is primarily used in the AWS cloud for locating services for the purpose of load balancing and failover of middle-tier servers."

Eureka server with Spring Cloud configuration

```java
@SpringBootApplication
@EnableEurekaServer
public class ServiceDiscoveryApp {

    public static void main(String[] args){
        SpringApplication.run(ServiceDiscoveryApp.class, args);
    }

}
```

```yaml
server:
  port: 8761

eureka:
  instance:
    hostname: localhost
  client:
    registerWithEureka: false
    fetchRegistry: false
    serviceUrl:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

wavelabs

Eureka server dashboard

localhost:8761

Search

System Status

Environment	test
Data center	default

Current time	2016-08-23T23:21:11 -0300
Uptime	00:01
Lease expiration enabled	false
Renews threshold	1
Renews (last min)	0

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

Name	Value
total-avail-memory	278mb
environment	test

wavelabs

Eureka client configuration

```java
@SpringBootApplication
@EnableDiscoveryClient
public class Application {

    public static void main(String[] args){
        SpringApplication.run(Application.class, args);
    }

}
```

```yaml
server:
  port: 9001

spring:
  application:
    name: book-service

eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
```

When you run the client application...

```
com.netflix.discovery.DiscoveryClient    : Application version is -1: true
com.netflix.discovery.DiscoveryClient    : Getting all instance registry info from the eureka server
com.netflix.discovery.DiscoveryClient    : The response status is 200
com.netflix.discovery.DiscoveryClient    : Starting heartbeat executor: renew interval is: 30
c.n.discovery.InstanceInfoReplicator     : InstanceInfoReplicator onDemand update allowed rate per min is 4
com.netflix.discovery.DiscoveryClient    : Discovery Client initialized at timestamp 1472005974288 with initial instances count: 1
c.n.e.EurekaDiscoveryClientConfiguration : Registering application book-service with eureka with status UP
com.netflix.discovery.DiscoveryClient    : Saw local status change event StatusChangeEvent [timestamp=1472005974367, current=UP, previous=STARTIN
com.netflix.discovery.DiscoveryClient    : DiscoveryClient_BOOK-SERVICE/192.168.0.101:book-service:9001: registering service...
com.netflix.discovery.DiscoveryClient    : DiscoveryClient_BOOK-SERVICE/192.168.0.101:book-service:9001 - registration status: 204
s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 9001 (http)
c.n.e.EurekaDiscoveryClientConfiguration : Updating port to 9001
com.ms.bookservice.Application           : Started Application in 16.186 seconds (JVM running for 17.81)
c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
```

wavelabs

Show all instances registered

localhost:8761

Search

System Status

Environment	test
Data center	default

Current time	2016-08-24T00:09:13 -0300
Uptime	00:03
Lease expiration enabled	false
Renews threshold	6
Renews (last min)	1

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BOOK-SERVICE	n/a (2)	(2)	UP (2) - 192.168.0.101:book-service:9020 , 192.168.0.101:book-service:9001
USER-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.101:user-service:9002

wavelabs

Consuming a service registered on Eureka

```
@RestController
public class SchoolServiceController {
    @Autowired
    RestTemplate restTemplate;

    @RequestMapping(value = "/getSchoolDetails/{schoolname}", method = RequestMethod.GET)
    public String getStudents(@PathVariable String schoolname) {
        System.out.println("Getting School details for " + schoolname);
        String response = restTemplate.exchange("http://student-service/getStudentDetailsForSchool/{schoolname}", HttpM
        }, schoolname).getBody();

        System.out.println("Response Received as " + response);

        return "School Name -  " + schoolname + " \n Student Details " + response;
    }
```

wavelabs

Thank you…

wavelabs