Eureka

SpringCloud的注册中心服务

注册中心服务应当满足的功能

1:服务注册
2:服务发现
3:动态跟踪服务注册信息

注册中心的好处

1:实现客户端和服务端IP地址的接偶，避免对服务提供者IP硬编码
2:实现服务端的动态扩容
3:多模块部署下,可监控各个服务运行状况

启动eureka服务

1:添加eureka依赖包

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-eureka-server</artifactId>
    </dependency>
```

2:声明为eureka注册中心服务

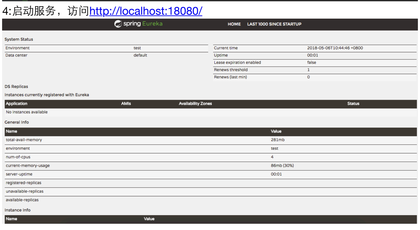
```
@SpringBootApplication
@EnableEurekaServer
public class CloudEurekaApplication {

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

3:配置文件

```
eureka:
  instance:
    hostname: localhost
  client:
    register-with-eureka: false
    fetch-registry: false
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

4:启动服务，访问http://localhost:18080/



注册服务

1:启动类声明为eureka客户端

```
import xxx

@SpringBootApplication
@EnableDiscoveryClient
@RestController
@EnableFeignClients
public class SpringcloudFeignApplication {

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

    @Autowired
    private FeignHello feignHello;

    @GetMapping("hello")
    public String helloFeign(){
        return feignHello.hello();
    }

}
```

2:配置注册中心地址

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:18080/eureka/
server:
```

原理解析

1:eureka提供的EndPoint

1:注册服务： POST /eureka/apps/appId
2:删除服务： DELETE /eureka/apps/appId/instanceId
3:更新服务： PUT /eureka/apps/appId/instanceId
4:服务发现： GET /eureka/apps/appId/instanceId
5:服务发现： GET /eureka/apps/appId
.....

客户端和注册中心所有的交互都是根据endpoint来的

2:注册服务

服务启动时，向注册中心发起注册请求。申请向注册中心注册，最终由注册中心维护一个服务注册表

3:服务发现

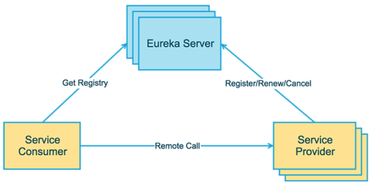
消费者根据服务名称，向注册中心查找对应的IP:PORT列表

4:服务跟踪

有定时任务向eureka发心跳。默认90秒之后收不到心跳，则认为服务已经关闭了

5:服务请求流程

1:客户端、服务端均向注册中心注册
2:客户端根据服务名到注册中心查找对应的IP:PORT
3:客户端向该IP发起RPC调用



6:注册中心集群

注册中心集群会共享彼此的注册表，即使某服务没有明确注册在该注册中心下。从而保证了注册中心的高可用性。推荐服务明确注册到各个注册中心上。

7:自我保护策略

当注册中心，某一时段内收不到大部分服务的心跳。会默认自身出了问题，并不会将这些服务踢出注册表

8:认证策略

为了防治其他服务恶意注册到注册中心上，eureka结合security完成了服务注册的认证流程

1:引入security依赖
2:声明注册用户名密码
3:客户端指明用户名密码

Eureka的高可用性

1:注册中心集群，某一节点宕机，其他节点继续对外服务。
2:自我保护策略，

Eureka的缓存

1:客户端拉去服务列表时，注册中心会从缓存中拿，并不是每次都去服务注册表中获取。默认30秒
2:客户端会缓存服务列表。默认30秒
3:负载均衡器Ribbon会缓存服务列表

思考

1:注册中心宕机,服务间能否正常通信?

可以,消费端会缓存服务端的IP:PORT,注册中心宕机,只是不能更新已经缓存的IP集合。所以还是可以通信的。但此时如果客户端宕机,客户端感知不到。由此也可以看出,注册中心只是服务注册和发现的作用,并不会干预服务间的调用。

https://blog.csdn.net/neosmith/article/details/53131023