Eureka 简介

Eureka 是 Spring Cloud Netflix 的一个子模块，也是核心模块之一，用于云端服务发现，是一个基于 REST 的服务，用于定位服务，以实现云端中间层服务发现和故障转移。

为什么要用 Eureka

在传统架构中，当调用 API 或者发起网络通信的时候，我们需要知道被调用方的 IP、端口号，大部分情况是通过域名和服务端口，事实上基于 DNS 的服务发现，而这种 DNS 方式，通常都是通过 nginx 或者其他代理软件实现的，因为物理机器的 ip 和端口号通常都是固定的，所以 nginx 中的配置信息也是固定的，服务列表需要通过手动去刷新，随着后台复杂度的增加，后台服务也越来越多，手动刷新的方式效率低而且容易出错，当后台发生故障时，处理也会变得很麻烦。

在微服务架构中，尤其是使用了 Docker 等虚拟化技术的微服务，其 IP 和 port 都是动态分配的，服务实例数也是动态变化的，那么就需要精细而准确的服务发现机制。当你有一个新的服务运行后，服务中心可以感知到，然后把它加添加到服务列表里，然后当它死掉后，会从服务中心移除，而作为一个服务，对其它服务公开的只是服务名称，而不是最终的服务地址 URL，这对于云平台容器化架构来说是非常重要的。

Eureka 原理

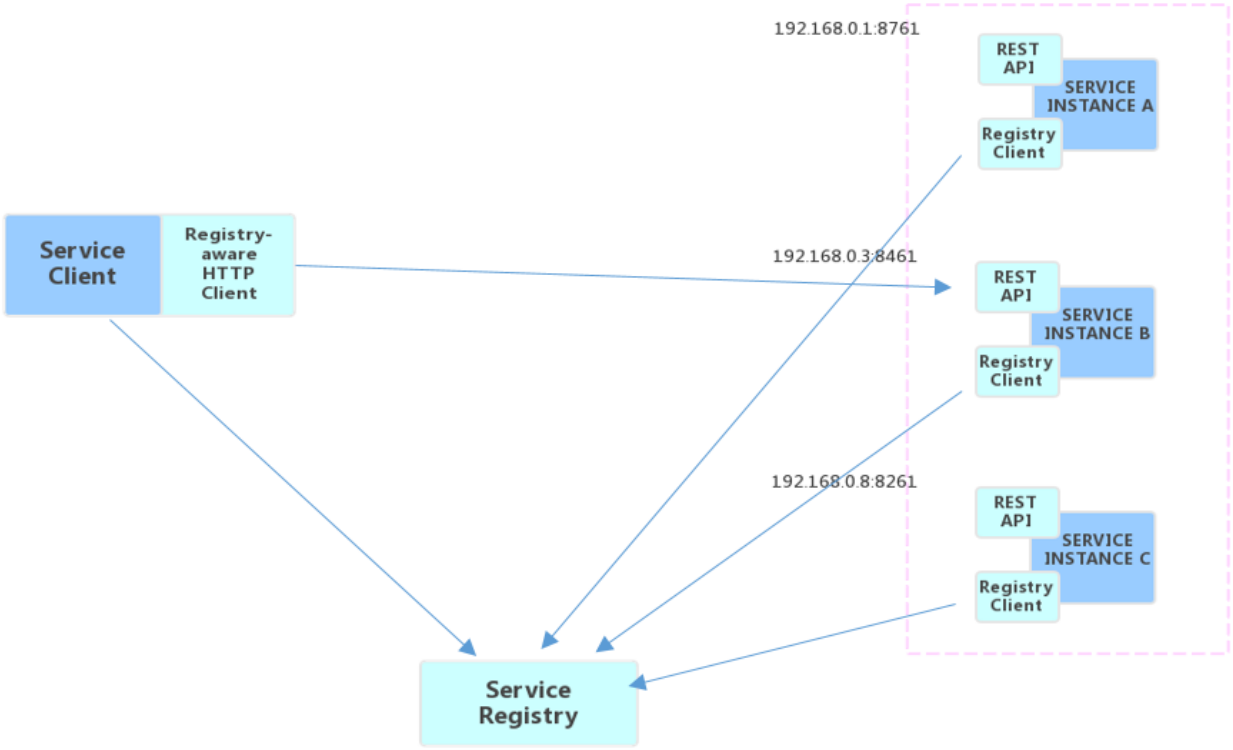

图-客户端发现模式

一个服务实例被启动时，它的网络地址会被写到注册表上；当服务实例终止时，再从注册表中删除。这个服务实例的注册表通过心跳机制动态刷新。

Netflix OSS 提供了一个客户端服务发现的好例子。Netflix Eureka 是一个服务注册表，提供了 REST API 用来管理服务实例的注册和查询可用的实例。Netflix Ribbon 是一个 IPC 客户端，和 Eureka 一起处理可用服务实例的负载均衡。

Eureka 架构

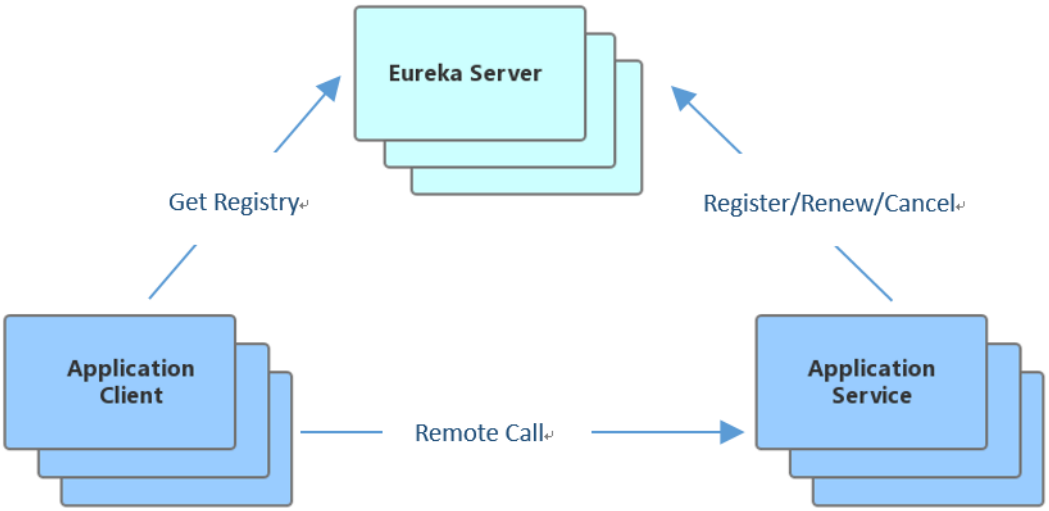

图-单节点

- Eureka Server: 注册中心

  服务都注册在哪里，哪里就可称为 Eureka Server

- Eureka Client

  注册在注册中心可以是服务的提供者（Applicaton Service）或者服务的消费者（Application Client），这两类都被称为 Eureka client

- Application Service：服务提供者

- Application Client：服务消费者

- Register：服务注册

  当 Eureka 客户端向 Eureka Server 注册时，会向 Eurekaserver 注册自己的信息（比如 IP，端口，微服务名等），Eureka Server 会存储这些信息。

- Renew：服务续约

  微服务启动后，会周期性（默认 30 秒）地向 Eureka Server 发送心跳。如果 Eureka Server

在一定时间内没有接收到某个微服务实例的心跳，Eureka Server 将会注销该实例（默认 90 秒）

• Cancel：服务下线

Eureka Client 在程序关闭时向 Eureka Server 发送取消请求。 发送请求后，该客户端实例信息将从服务器的实例注册表中删除。该下线请求不会自动完成，它需要调用以下内容：

```
DiscoveryManager.getInstance().shutdownComponent()
```

Eureka 特点

1. Eureka Client 会缓存服务注册表中的信息，并将其缓存在本地，该注册列表信息定期（默认 30 秒）更新一次，因此微服务无须每次请求都查询 Eureka Server，从而降低了 Eureka Server 的压力；其次，当注册中心挂了，客户端之间依然可以通过缓存中的信息进行调用；注册中心重启后，客户端会继续注册进来。

2.当服务提供者挂了，在关闭自我保护的情况下，注册中心在规定时间（默认是 60s）内移除客户端。

3.自我保护，当一个新的 Eureka Server 出现时，它尝试从相邻节点获取所有实例注册表信息。如果从 Peer 节点获取信息时出现问题，Eureka Serve 会尝试其他的 Peer 节点。如果服务器能够成功获取所有实例，则根据该信息设置应该接收的更新阈值。如果有任何时间，Eureka Serve 接收到的续约低于为该值配置的百分比（默认为 15 分钟内低于 85%），则服务器开启自我保护模式，即不再剔除注册列表的信息。