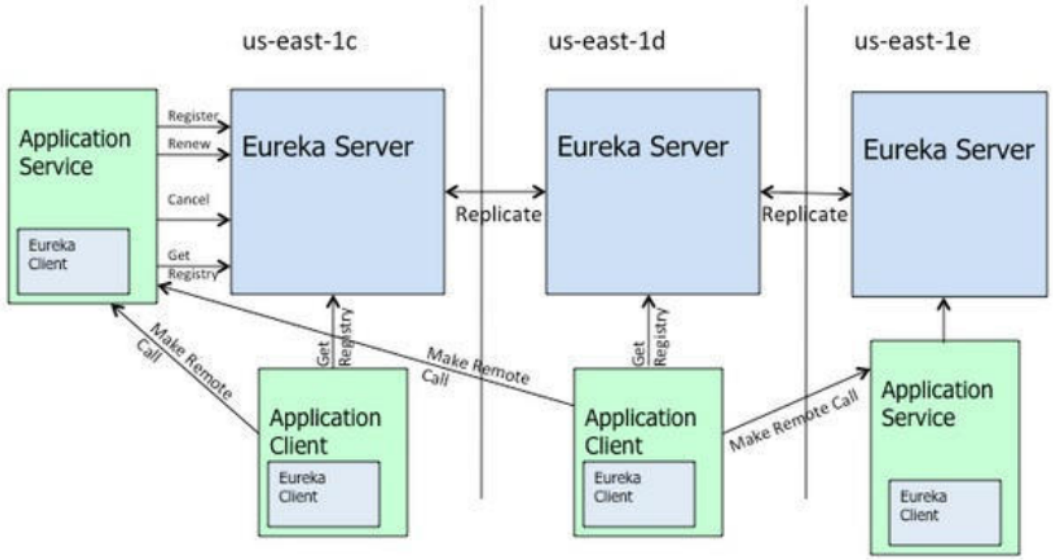
eureka集群



上图是来自eureka的官方架构图，这是基于集群配置的eureka；

- 处于不同节点的eureka通过Replicate进行数据同步
- Application Service为服务提供者
- Application Client为服务消费者
- Make Remote Call完成一次服务调用

服务启动后向Eureka注册，Eureka Server会将注册信息向其他Eureka Server进行同步，当服务消费者要调用服务提供者，则向服务注册中心获取服务提供者地址，然后会将服务提供者地址缓存在本地，下次再调用时，则直接从本地缓存中取，完成一次调用。

见示例：08-ms-eureka-server-ha

引入依赖

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

application.yml配置如下(见配置文件application-ha.yml)

```yaml
spring:
  application:
    name: microservice-eureka-server-ha
---
spring:
  profiles: peer1                                 # 指定profile=peer1
server:
  port: 8761
eureka:
  instance:
    hostname: peer1                               # 指定当profile=peer1时，主机名是peer1
  client:
    serviceUrl:
      defaultZone: http://peer2:8762/eureka/      # 将自己注册到peer2这个Eureka上面去

---
spring:
  profiles: peer2
server:
  port: 8762
eureka:
  instance:
    hostname: peer2
  client:
    serviceUrl:
      defaultZone: http://peer1:8761/eureka/
```

配置host：127.0.0.1 peer1 peer2

用jar包方式启动：

java -jar 项目的jar包 --spring.profiles.active=peer1

springboot-actuator监控

springboot-actuator提供了很多系统的监控端点，如下所示：

ID	描述	敏感（Sensi
autoconfig	显示一个auto-configuration的报告，该报告展示所有auto-configuration候选者及它们被应用或未被应用的原因	true
beans	显示一个应用中所有Spring Beans的完整列表	true
configprops	显示一个所有@ConfigurationProperties的整理列表	true
dump	执行一个线程转储	true
env	暴露来自Spring ConfigurableEnvironment的属性	true
health	展示应用的健康信息（当使用一个未认证连接访问时显示一个简单的’status’，使用认证连接访问则显示全部信息详情）	false
info	显示任意的应用信息	false
metrics	展示当前应用的’指标’信息	true
mappings	显示一个所有@RequestMapping路径的整理列表	true

shutdown	允许应用以优雅的方式关闭（默认情况下不启用）	true
trace	显示trace信息（默认为最新的一些HTTP请求）	true

见示例：08-ms-provider-user

在项目中添加依赖：

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

增加配置：

```
management:
  security:
    enabled: false  #关掉安全认证
  port: 8888  #管理端口调整成8888,独立的端口可以做安全控制
  context-path: /monitor  #actuator的访问路径
```

Spring Boot Admin监控

Spring Boot Admin 是一个管理和监控Spring Boot 应用程序的开源软件，它针对spring-boot的actuator接口进行UI美化封装

搭建spring boot admin的服务端，见示例：08-ms-spring-boot-admin

添加依赖并在启动类上增加注解@EnableAdminServer

```
<dependency>
    <groupId>de.codecentric</groupId>
    <artifactId>spring-boot-admin-server</artifactId>
    <version>1.5.6</version>
</dependency>
<dependency>
    <groupId>de.codecentric</groupId>
    <artifactId>spring-boot-admin-server-ui</artifactId>
    <version>1.5.6</version>
</dependency>
```

```
@Configuration
@EnableAutoConfiguration
@EnableAdminServer
public class SpringBootAdminApplication {

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

搭建spring boot admin的客户端，见示例：08-ms-provider-user

添加依赖

```xml
<!-- spring boot admin监控客户端依赖 -->
<dependency>
  <groupId>de.codecentric</groupId>
  <artifactId>spring-boot-admin-starter-client</artifactId>
  <version>1.5.6</version>
</dependency>
```

增加application.yml配置：

spring.boot.admin.url=http://localhost:9999 # spring boot admin服务端地址，搜集客户端监控数据

微服务网关背景及简介

不同的微服务一般有不同的网络地址，而外部的客户端可能需要调用多个服务的接口才能完成一个业务需求。比如一个电影购票的收集APP,可能回调用电影分类微服务，用户微服务，支付微服务等。如果客户端直接和微服务进行通信，会存在一下问题：

客户端会多次请求不同微服务，增加客户端的复杂性

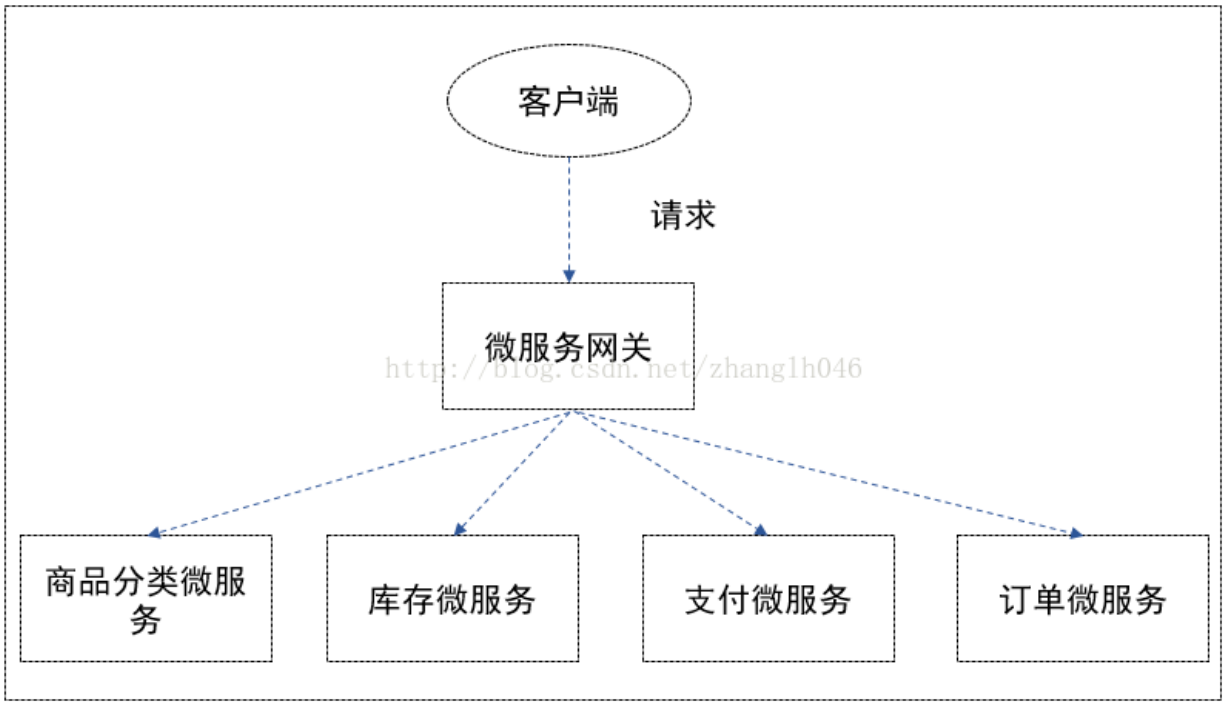
存在跨域请求，在一定场景下处理相对复杂

认证复杂，每一个服务都需要独立认证

难以重构，随着项目的迭代，可能需要重新划分微服务，如果客户端直接和微服务通信，那么重构会难以实施

某些微服务可能使用了其他协议，直接访问有一定困难

上述问题，都可以借助微服务网管解决。微服务网管是介于客户端和服务器端之间的中间层，所有的外部请求都会先经过微服务网关，架构演变成：



这样客户端只需要和网关交互，而无需直接调用特定微服务的接口，而且方便监控，易于认证，减少客户端和各个微服务之间的交互次数

zuul微服务网关

Zuul是Netflix开源的微服务网关，他可以和Eureka,Ribbon,Hystrix等组件配合使用。Zuul组件的核心是一系列的过滤器，这些过滤器可以完成以下功能：

动态路由：动态将请求路由到不同后端集群

压力测试：逐渐增加指向集群的流量，以了解性能

负载分配：为每一种负载类型分配对应容量，并弃用超出限定值的请求

静态响应处理：边缘位置进行响应，避免转发到内部集群

身份认证和安全: 识别每一个资源的验证要求，并拒绝那些不符的请求

Spring Cloud对Zuul进行了整合和增强。目前，Zuul使用的默认是Apache的HTTP Client，也可以使用Rest Client，可以设置ribbon.restclient.enabled=true.

编写一个简单微服务网关，**见示例：08-ms-gateway-zuul**

添加依赖并在启动类上增加注解@EnableZuulProxy，声明一个zuul代理。该代理使用Ribbon来定位注册在 Eureka server中的微服务；同时，该代理还整合了Hystrix，从而实现了容错，所有经过 zuul的请求都会在 Hystrix命令中执行。

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

配置文件application.yml如下：

```
server:
  port: 8040
spring:
  application:
    name: microservice-gateway-zuul
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    prefer-ip-address: true
```

运行项目(**需启动两个用户微服务和一个订单微服务**，参看视频)，访问地址：

http://localhost:8040/microservice-provider-user/getIpAndPort

发现zuul会代理所有注册到eureka server的微服务，并使用ribbon做负载均衡，zuul的路由规则如下，可以访问地址：http://localhost:8040/routes

http://ZUUL_HOST:ZUUL_PORT/微服务在Eureka上的serviceId/**会被转发到serviceId对应的微服务

zuul还自动整合了hystrix，访问地址：http://localhost:8040/hystrix.stream