买单侠微服务的API网关演进之路

架构师
林丹

上海秦苍信息科技有限公司（Omni Prime）以“你的梦想，我来买单”为宗旨，是国内领先的FinTech（金融科技）公司，专注为中国年轻人群提供消费分期服务。旗下现有“买单侠”和“星计划”系列产品。“买单侠”面向中国年轻蓝领用户，提供移动端消费分期服务。“星计划”为医美商户和年轻用户提供消费金融场景下的一站式解决方案。

CATALOGUE

为什么需要API网关

- 路由
- 安全/鉴权
- 日志/监控/报警
- …

API网关类型

API网关	**网关类型**		**服务对象**
	接入层网关	入口网关	面向web应用
			面向mobile应用
			面向物联网设备
		出口网关	面向合作渠道
	应用层网关		面向内部服务

从无到有

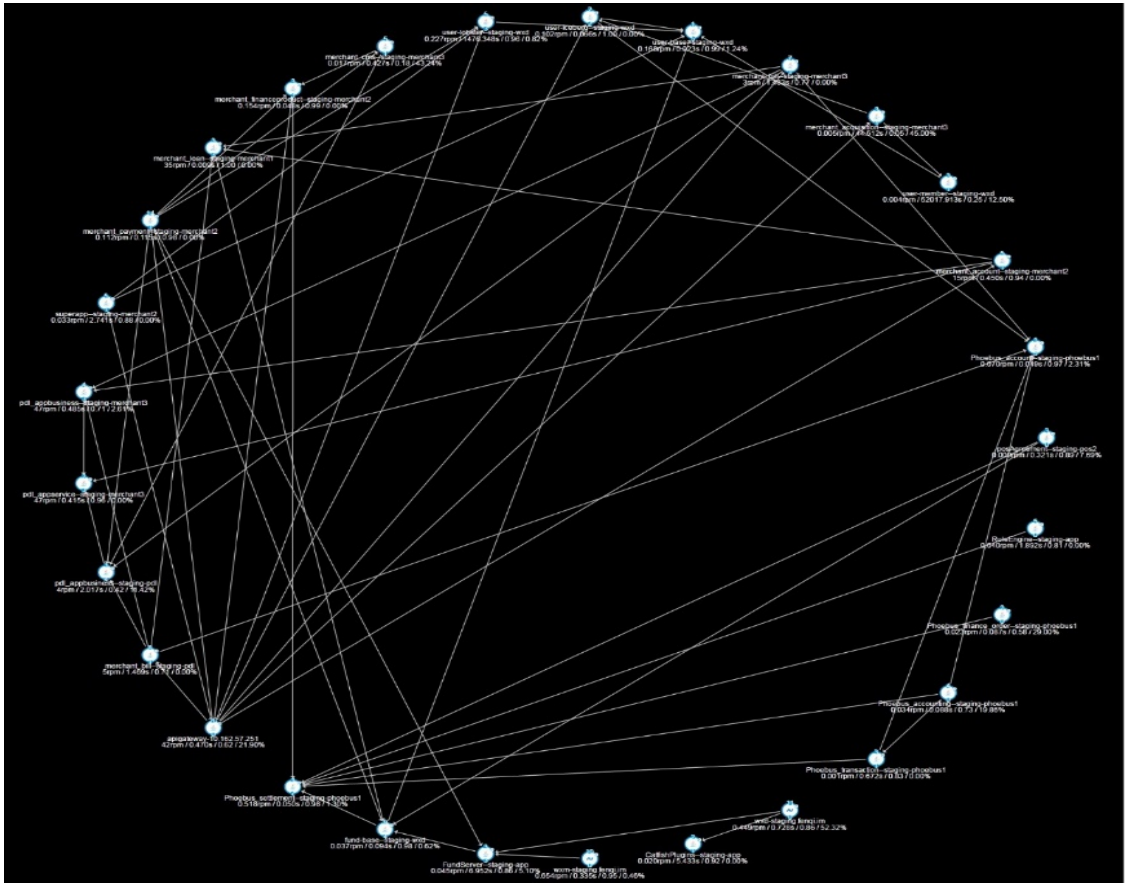
面向服务的API网关Spring Cloud Zuul

挑战

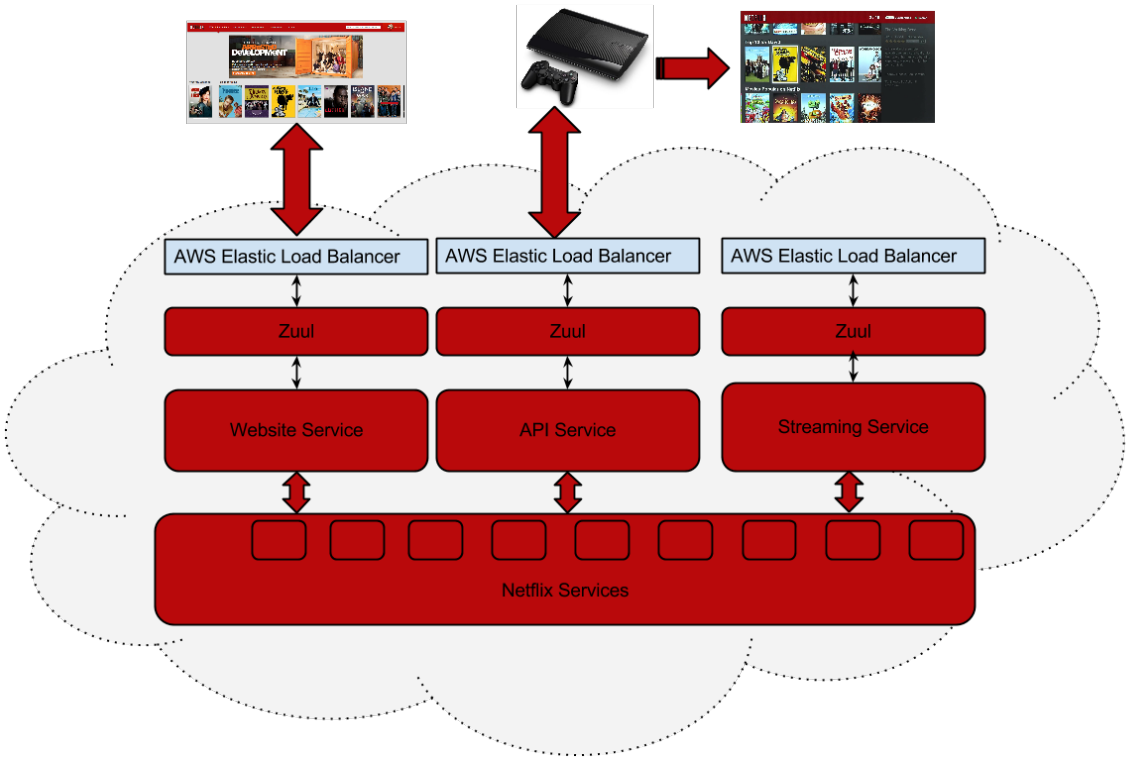
随着微服务增减添加发现路由

没有API网关的微服务

- 对外耦合强
- 对内关系乱
- 出错多
- 重构难



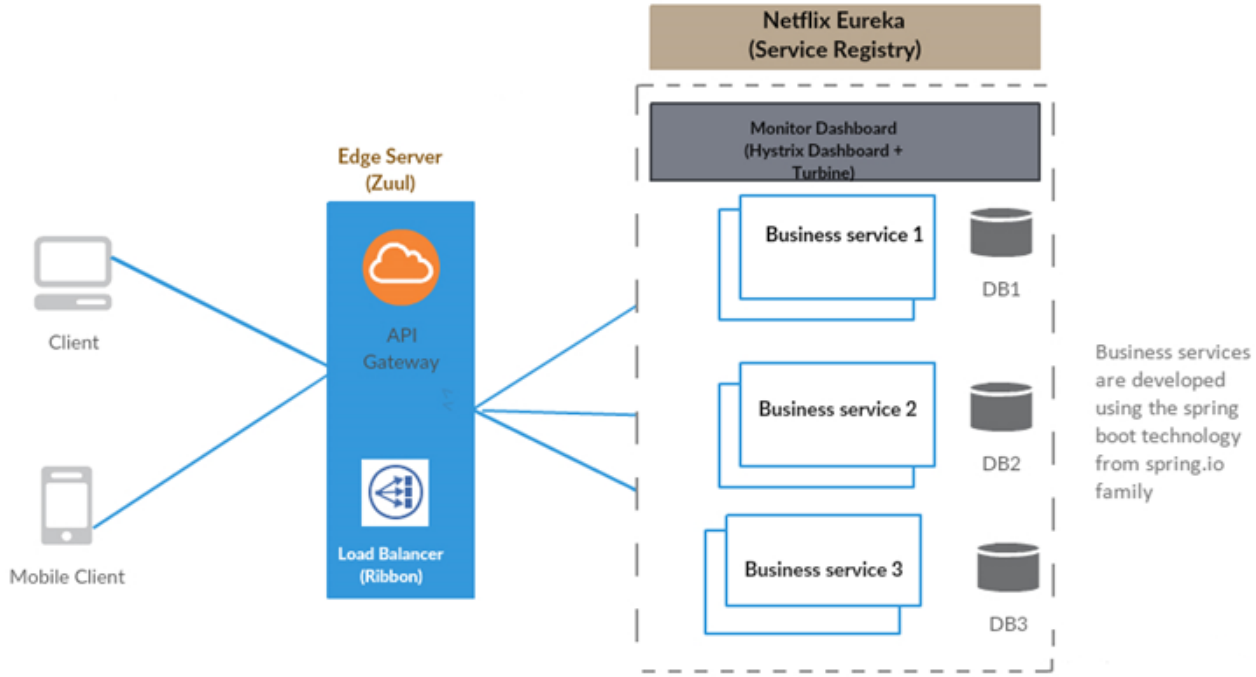
为什么选择Zuul



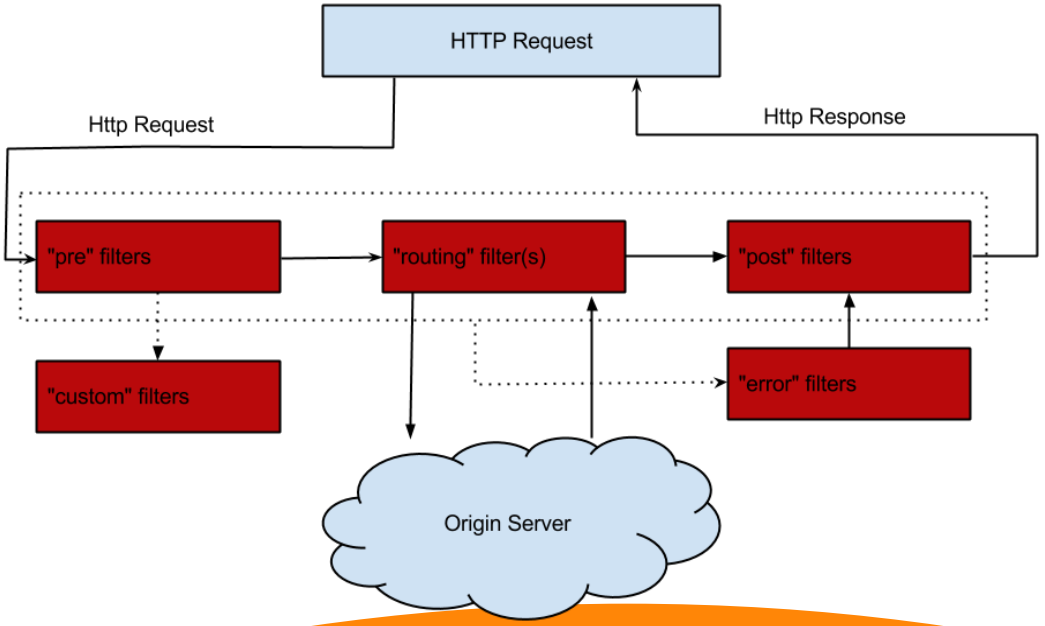
- 验证与安全保障
- 审查与监控
- 动态路由
- 压力测试
- 负载分配
- 静态响应处理
- 多区域弹性

服务发现和自动路由

http://[Zuul IP: PORT]/demo-service/user



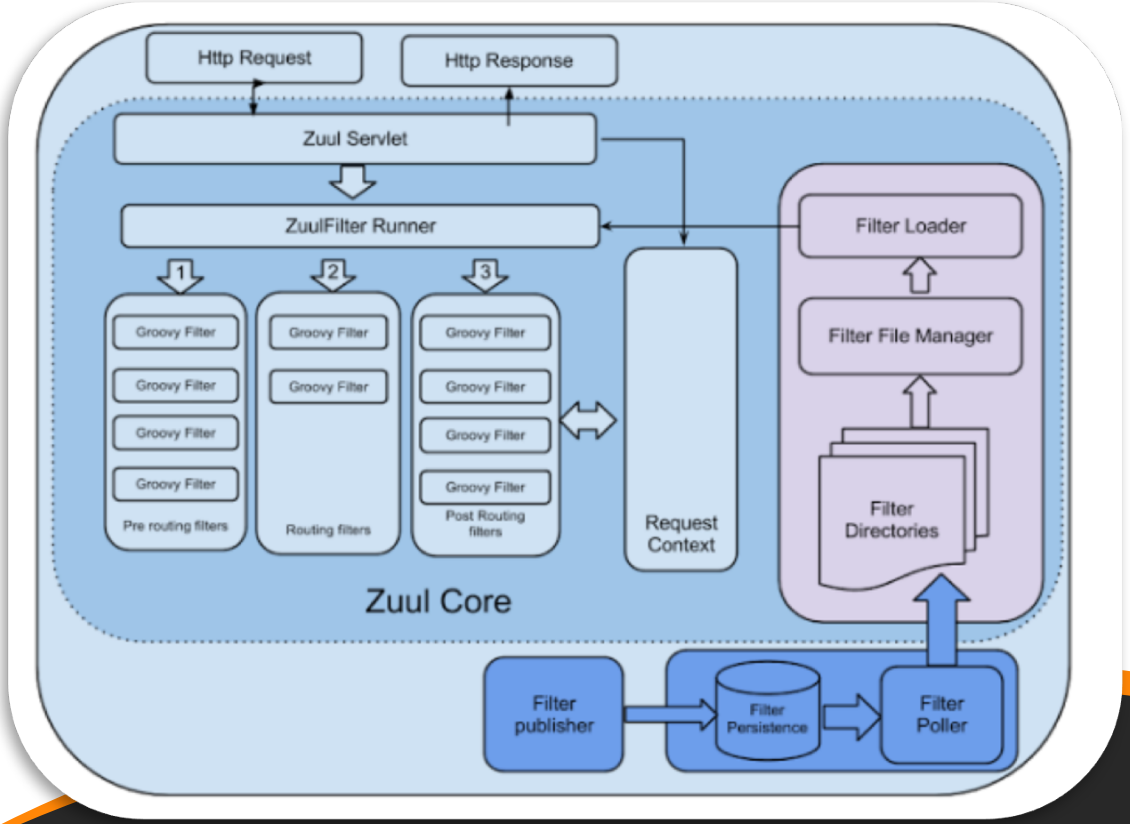
Zuul的过滤器框架



开发友好

请求处理
参数校验
参数转换

热加载


Http Request
Http Response
Zuul Servlet
ZuulFilter Runner
1
2
3
Groovy Filter
Groovy Filter
Groovy Filter
Groovy Filter
Pre routing filters
Groovy Filter
Groovy Filter
Routing filters
Groovy Filter
Groovy Filter
Groovy Filter
Groovy Filter
Post Routing filters
Request Context
Filter Loader
Filter File Manager
Filter Directories
Zuul Core
Filter publisher
Filter Persistence
Filter Poller

动态加载过滤器

热加载

```java
@EnableZuulProxy
@Import({ FilterConfig.class, UniconfConfig.class, FF4jConfig.class, RedisConfig.class })
public class ApiGatewayApplication implements CommandLineRunner {

    public static void main(String[] args) {
        try {
            //动态加载路由策略
            GroovyStrategyLoader.getInstance().setCompiler(new GroovyCompiler());
            GroovyFileManager.setFilenameFilter(new GroovyFileFilter());
            GroovyFileManager.init(30, "groovy/ff4j/strategy");
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        new SpringApplicationBuilder(ApiGatewayApplication.class).web(true).run(args);
    }
```

动态加载
灰度发布的
路由策略

从少到多

支持灰度发布的服务分组入口Zuul

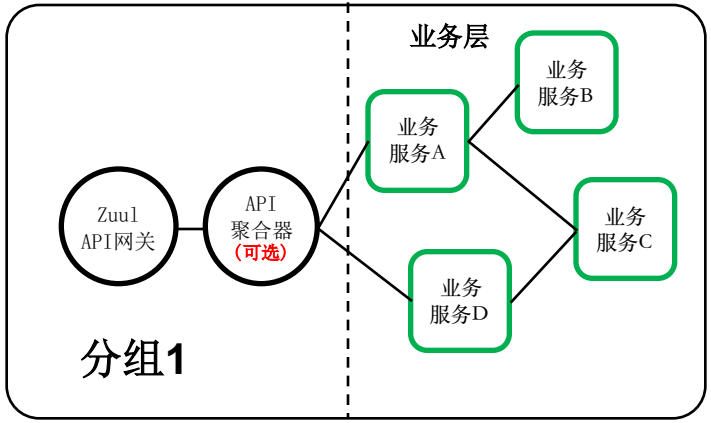
挑战

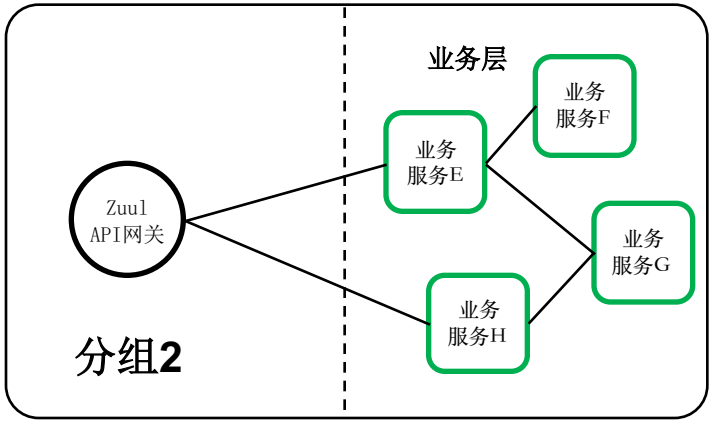
微服务数量多，管理难度大

服务分组

分组原则和单一入口原则

- 紧密相关的服务构成一个组（应用），组内所有服务通过一个API网关暴露服务。
- 外部服务只能通过API网关调用组内的服务。逻辑上，可以把整个组内的服务看做一个服务。



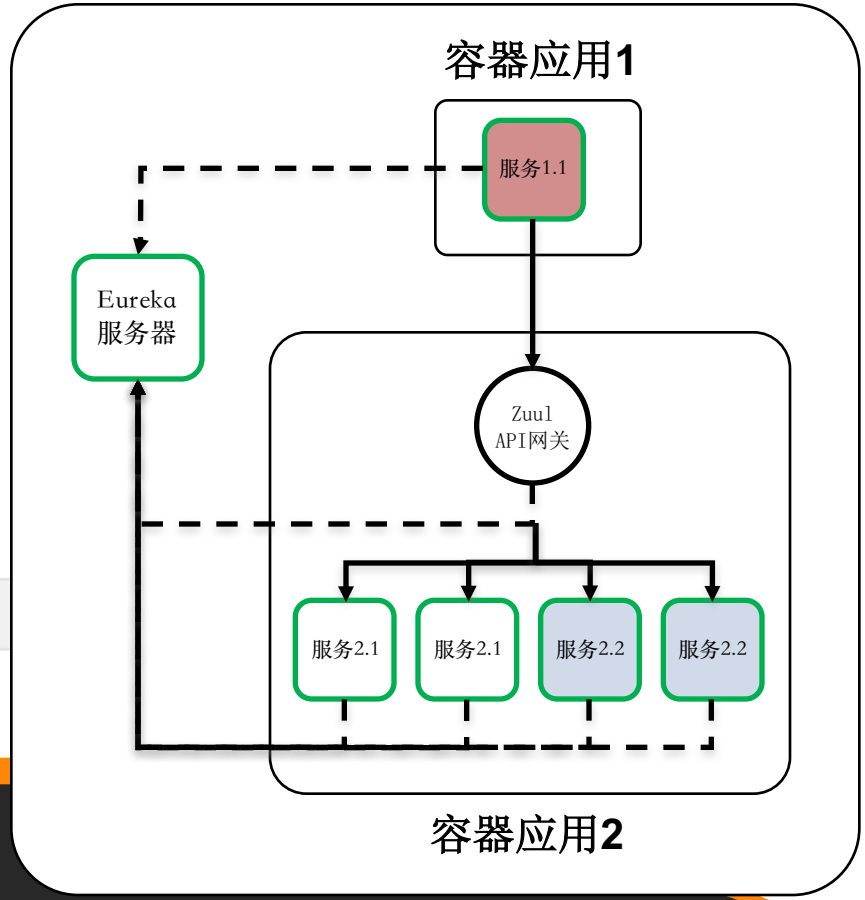


服务发现

- 应用内的服务直接调用
- 跨应用的调用都需要经过API网关路由到相应的服务

开发需要在application.properties添加下面设置，从而指定应用的入口

```
eureka.instance.metadataMap.groupName=【application-apigateway】
```



服务发现基于网关的实现

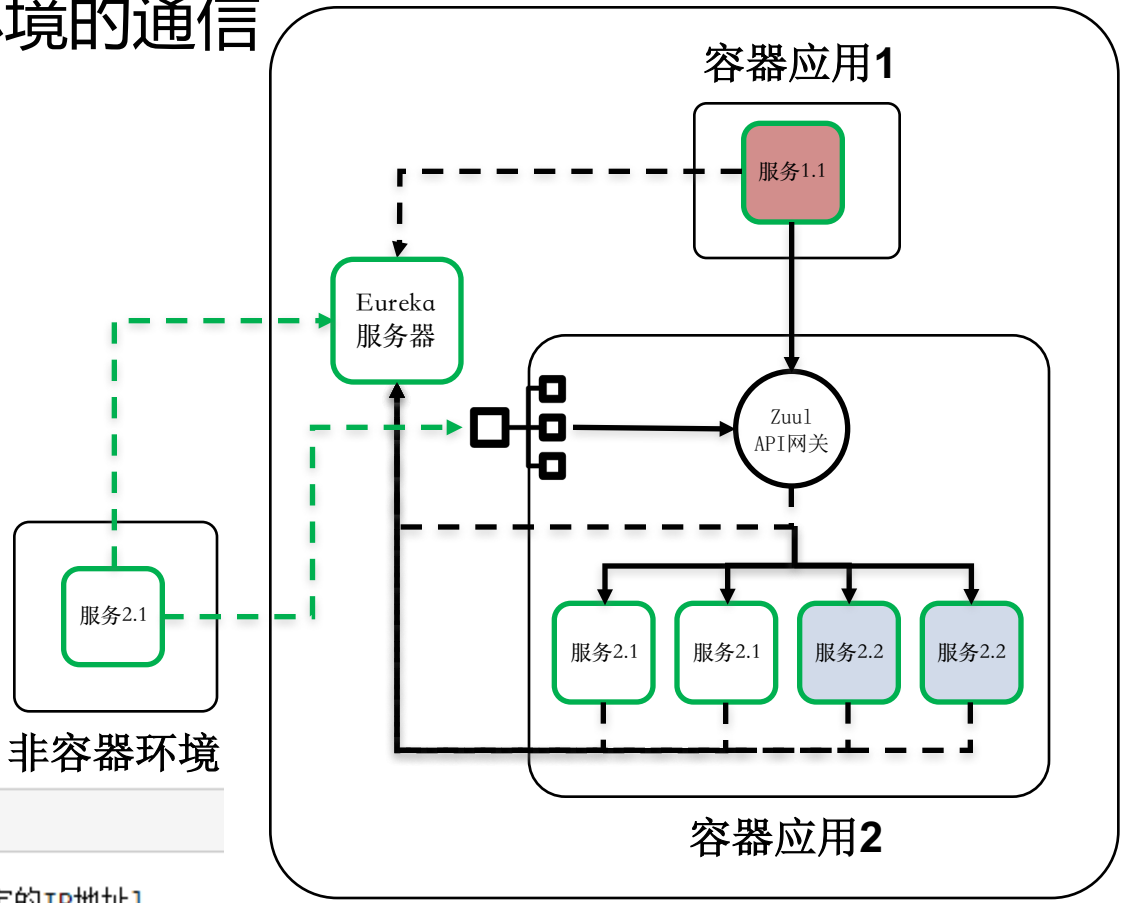
ClientLoadBalancer.java

ClientLoadBalancer execute()

```
04
05          URI newUri = null;
06          ServiceInstance serverInstance = loadBalancer.choose(serviceName);
07          if (serverInstance != null) {
08              if (!StringUtils.isEmpty(serverInstance.getHost())) {
09                  log.info("Service Information: " + serverInstance.getHost() + ":" + serverInstance.getPort());
10              }
11
12              String targetApiGatewayServiceId = serverInstance.getMetadata().get(METADATA_KEY_APIGATEWAY_SERVICEID);
13              if (!StringUtils.isEmpty(targetApiGatewayServiceId) &&
14                  !targetApiGatewayServiceId.equalsIgnoreCase(BalancedClientRequestFilter.getApiGatewayServiceId())) {
15                  ServiceInstance apiGatewayInstance = loadBalancer.choose(targetApiGatewayServiceId);
16                  newUri = reconstructURIWithAPIGateway(apiGatewayInstance, oldUri);
17                  log.info("Service API Gateway Information: " + apiGatewayInstance.getHost() + ":" + apiGatewayInstance.getPort());
18              }
19
20              if (newUri == null) {
21                  newUri = loadBalancer.reconstructURI(serverInstance, oldUri);
22              }
23          }
```

容器环境和非容器环境的通信

- 容器环境中，API网关将自己的地址注册为SLB的地址
- 非容器环境中，通过SLB访问容器环境中的服务



api-gateway注册到Eureka的额外配置

```
eureka.instance.hostname=[SLB或节点固定的IP地址]
#eureka.instance.preferIpAddress=false
```

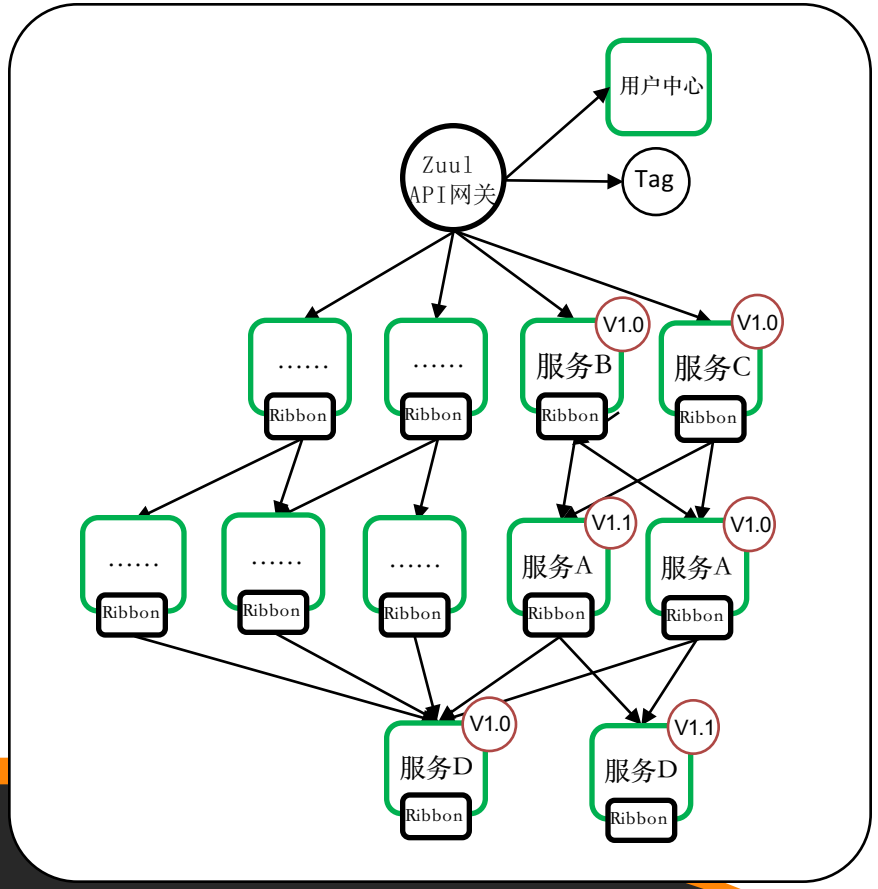
灰度发布

Tag	A	B	C	D
tag1	1.1	1.0	1.0	1.0
tag2	1.0	1.0	1.0	1.1
tag3	1.0	1.0	1.0	1.1
默认	1.0	1.0	1.0	1.0

权重策略

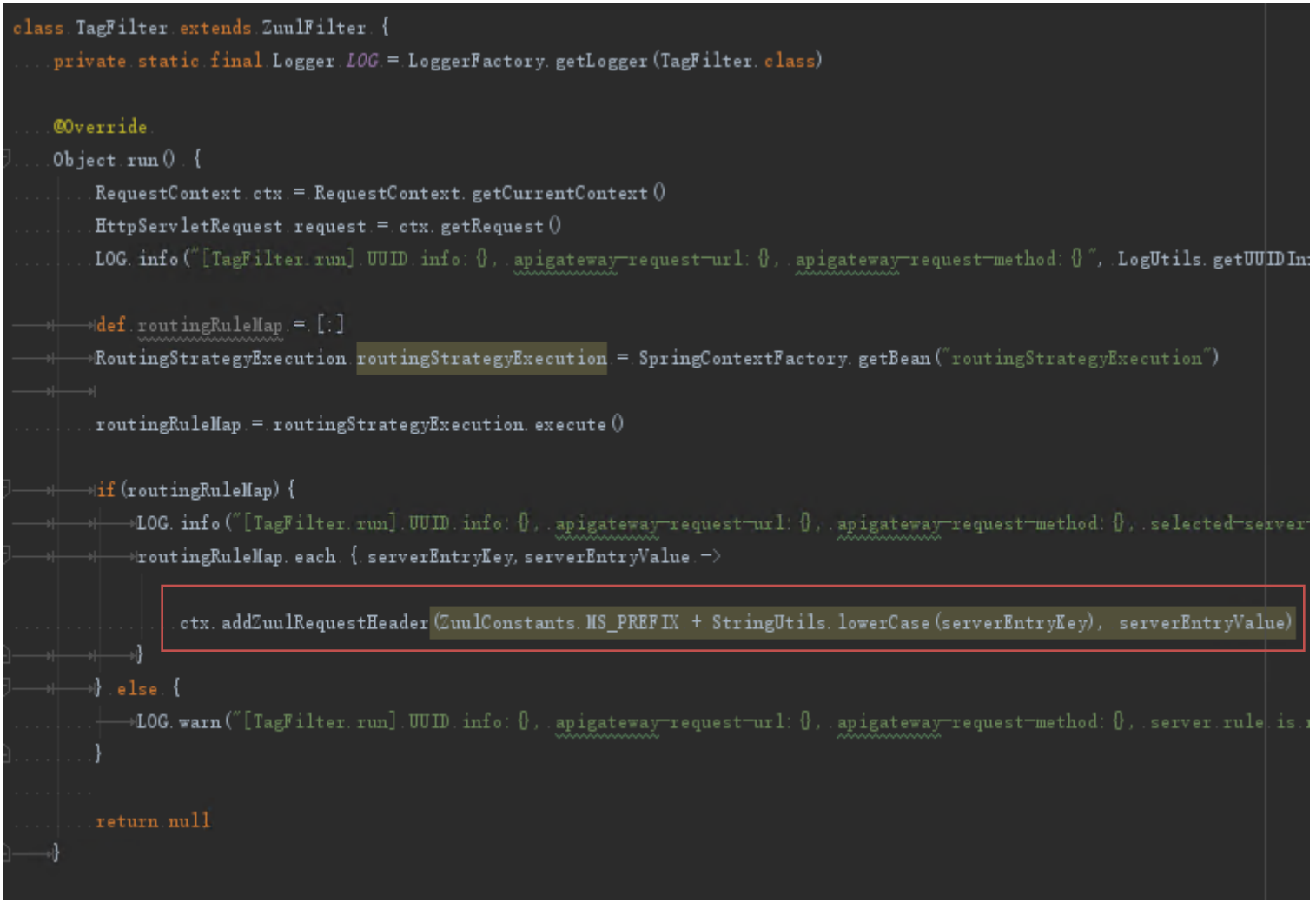
白名单策略

区域策略



容器应用

标签



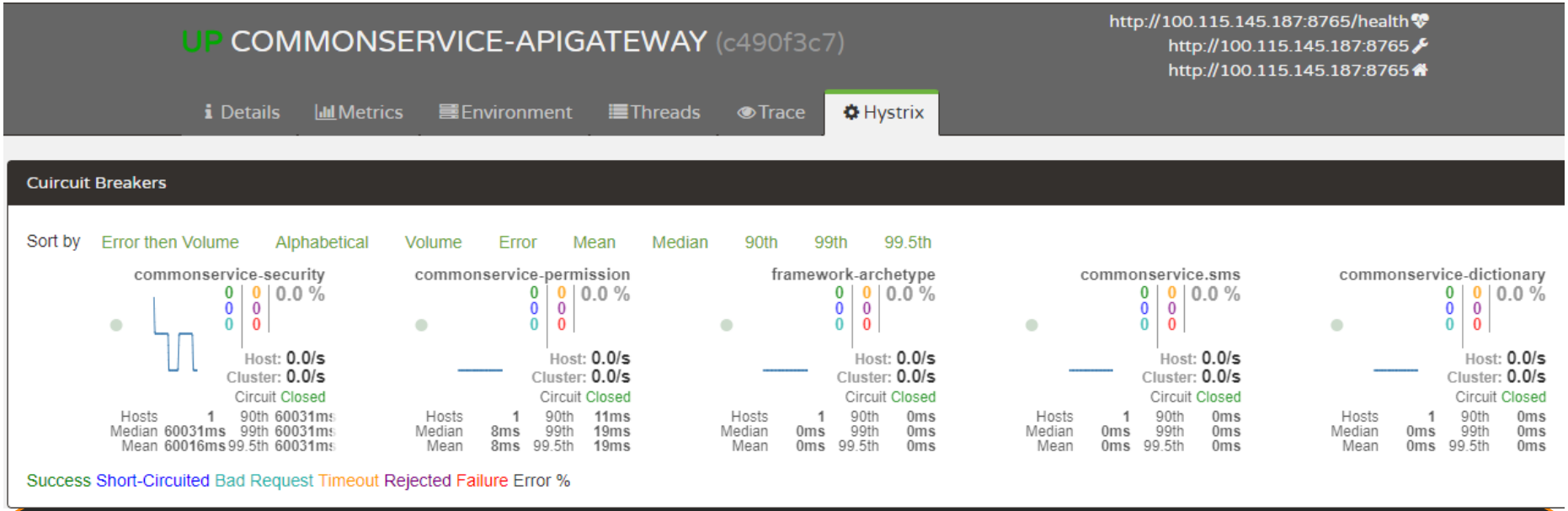
```
class TagFilter extends ZuulFilter {
    private static final Logger LOG = LoggerFactory.getLogger(TagFilter.class)

    @Override
    Object run() {
        RequestContext ctx = RequestContext.getCurrentContext()
        HttpServletRequest request = ctx.getRequest()
        LOG.info("[TagFilter.run] UUID info: {}, apigateway-request-url: {}, apigateway-request-method: {}", LogUtils.getUUIDIn

        def routingRuleMap = [:]
        RoutingStrategyExecution routingStrategyExecution = SpringContextFactory.getBean("routingStrategyExecution")

        routingRuleMap = routingStrategyExecution.execute()

        if (routingRuleMap) {
            LOG.info("[TagFilter.run] UUID info: {}, apigateway-request-url: {}, apigateway-request-method: {}, selected-server
            routingRuleMap.each { serverEntryKey, serverEntryValue ->

                ctx.addZuulRequestHeader(ZuulConstants.MS_PREFIX + StringUtils.lowerCase(serverEntryKey), serverEntryValue)
            }
        } else {
            LOG.warn("[TagFilter.run] UUID info: {}, apigateway-request-url: {}, apigateway-request-method: {}, server.rule.is.
        }

        return null
    }
```

路由

```java
public class VersionPredicate extends AbstractServerPredicate {
    private static final Logger LOG = LoggerFactory.getLogger(VersionPredicate.class);

    @Override
    public boolean apply(PredicateKey input) {
        RequestContext ctx = RequestContext.getCurrentContext();

        LOG.debug("[VersionPredicate.apply] UUID info: {}", LogUtils.getUUIDInfo());

        Server server = input.getServer();
        if (!(server instanceof DiscoveryEnabledServer)) {
            LOG.debug("[VersionPredicate.apply] UUID info: {}, server is not instanceof DiscoveryEnabledServer", LogUtils.getUUIDInfo());

            return false;
        }

        DiscoveryEnabledServer discoveryEnabledServer = (DiscoveryEnabledServer) server;
        String requetsedVersionKey = ZuulConstants.MS_PREFIX + StringUtils.lowerCase(discoveryEnabledServer.getInstanceInfo().getAppName());
        String requestedVersion = ctx.getZuulRequestHeaders().get(requetsedVersionKey);
        String serverVersion = discoveryEnabledServer.getInstanceInfo().getMetadata().get(ZuulConstants.VERSION);
        boolean result = (requestedVersion == null || requestedVersion.equalsIgnoreCase(serverVersion));

        LOG.debug("[VersionPredicate.apply] UUID info: {}, ZuulRequestHeaders: {}, request-zuulheader-key: {}, request-zuulheader-version: {}, req

        return result;
    }

}
```

监控和熔断

健康检查
调用链跟踪 Sleuth + Zipkin
熔断及实时监控



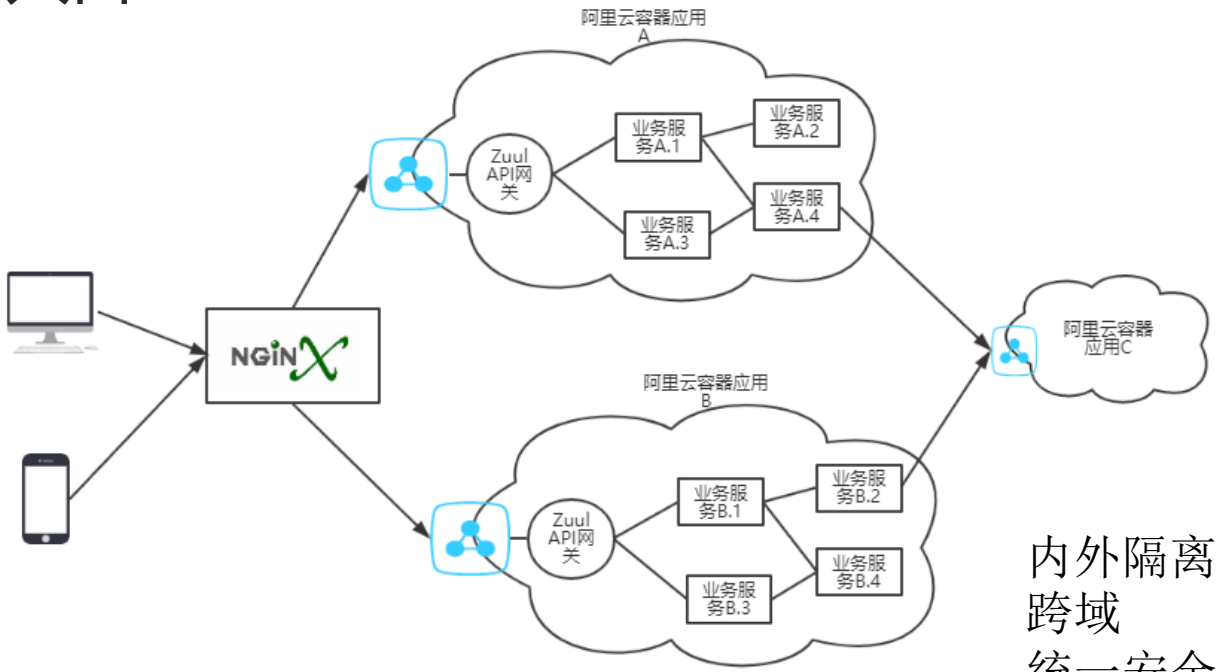
从单一到多元

接入层网关Nginx(Kong)

挑战

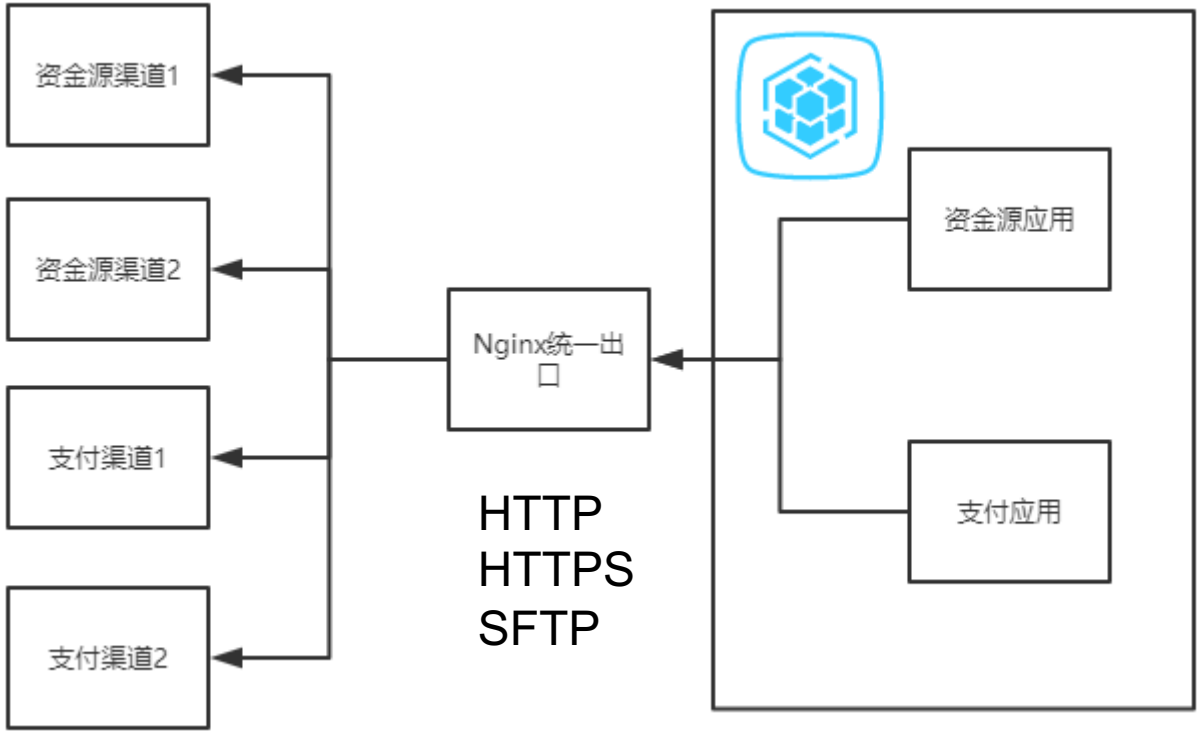
服务和外界直接接触，安全风险高，认证难

统一入口



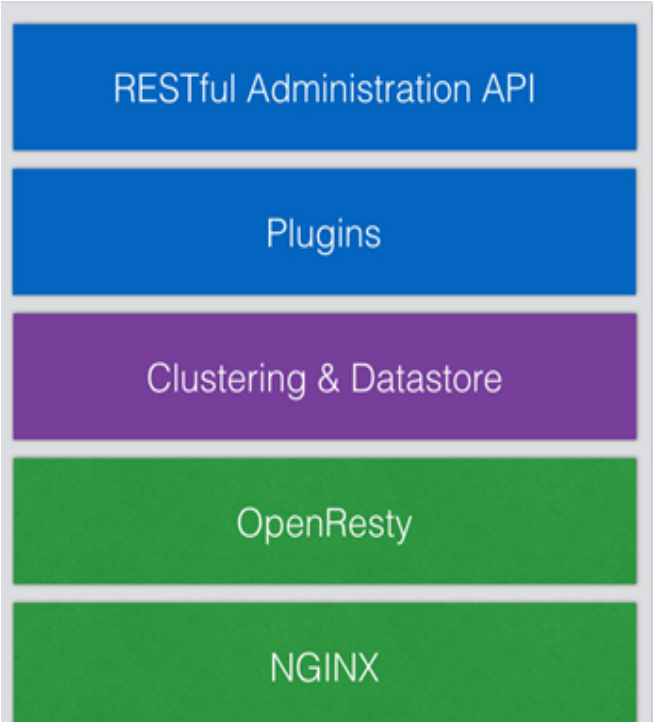
内外隔离
跨域
统一安全管控
支持多套环境的自动化测试

统一出口



统一授权
统一监控
统一审计

接入层网关探索从Nginx到Kong



KONG

Kong =OpenResty +NGINX

配置存储在 Cassandra or PostgreSQL

对Kong的配置操作都过REST API完成并即时生效

高扩展性

KONG Star

ABOUT DOCS PLUGINS COMMUNITY ENTERPRISE GITHUB INSTALLATION

Security

Protect your services with additional security layers:

ACL
Control which consumers can access APIs

CORS
Allow developers to make requests from the browser

Dynamic SSL
Add an SSL certificate for an underlying service

IP Restriction
Whitelist or blacklist IPs that can make requests

Bot Detection
Detects and blocks bots or custom clients

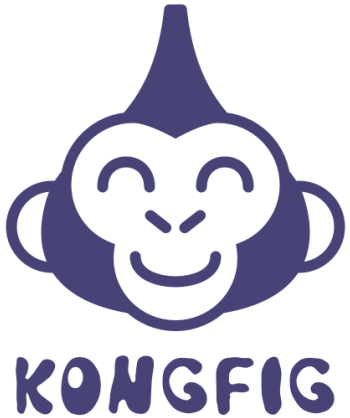
Traffic Control

Manage, throttle and restrict inbound and outbound API traffic:

Rate Limiting
Rate-limit how many HTTP requests a developer can make

Response Rate Limiting
Rate-Limiting based on a custom response header value

Request Size Limiting
Block requests with bodies greater than a specific size



Home APIs Consumers Plugins

APIs

Name	Host(s)	Uri(s)	Method(s)	Upstream url	Created
example-api	example.com			http://httpbin.org	Apr 21, 2017
test		/test	• GET • POST	http://httpbin.org	Apr 21, 2017
example		/zuul	• GET • POST	http://172.16.15.30:8765	Apr 20, 2017

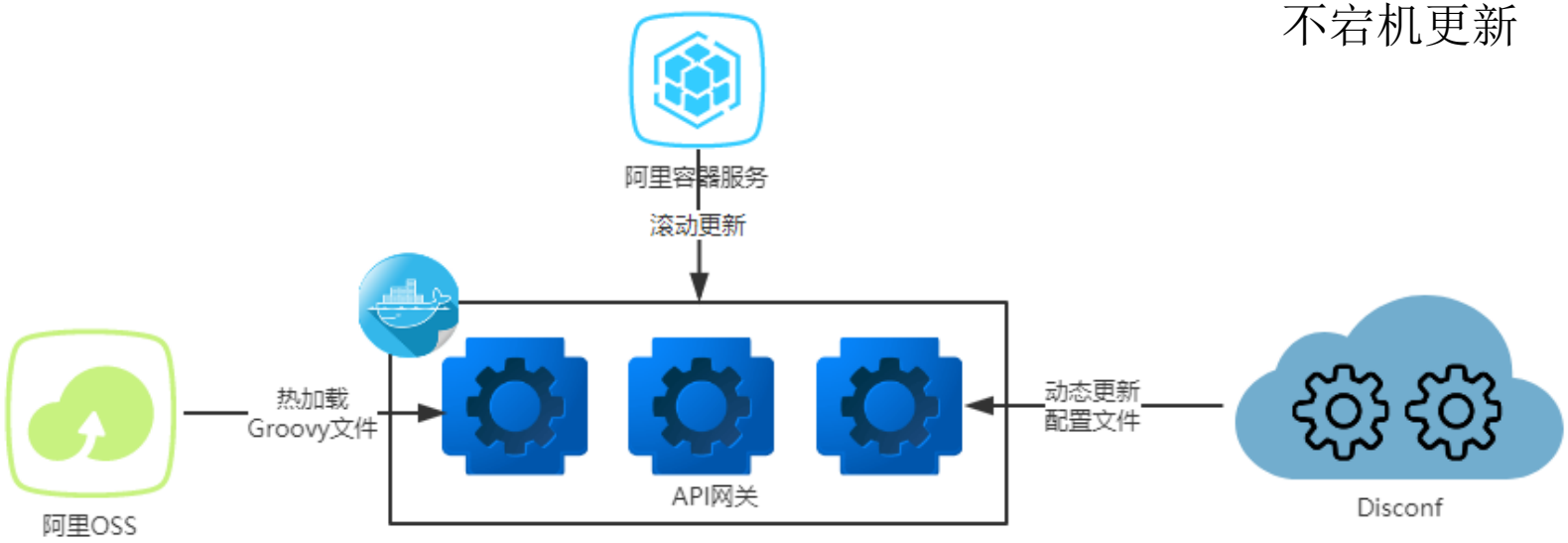
总结

API Gateway 成本收益

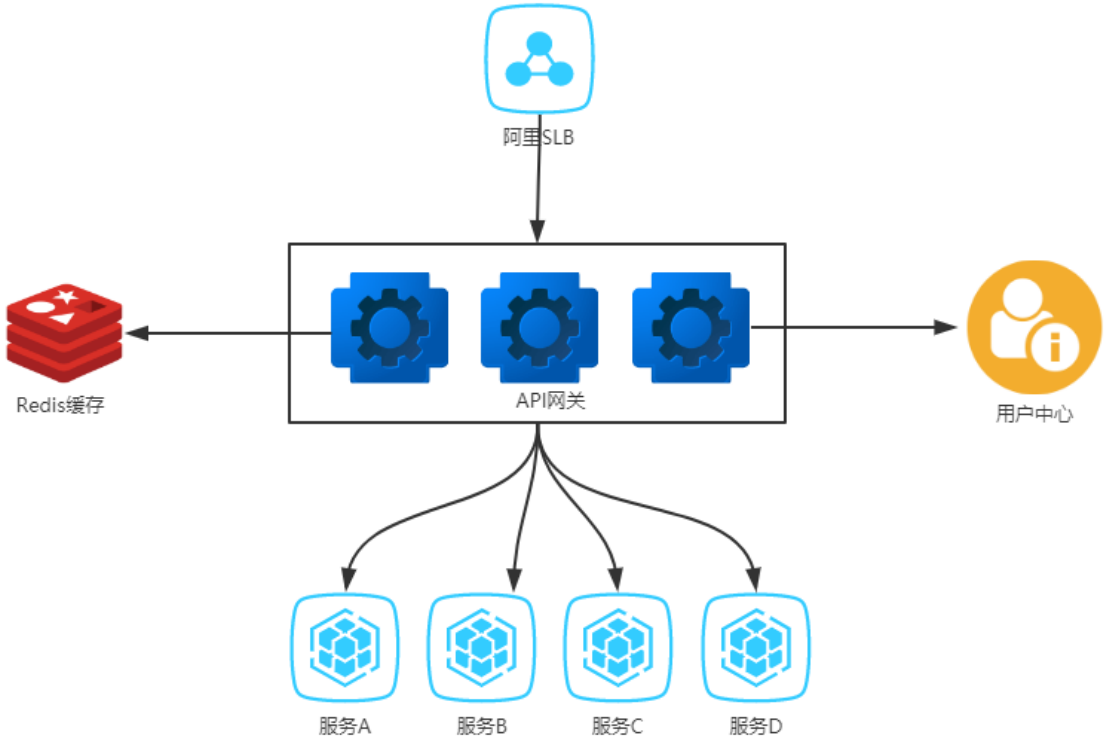
收益	成本
内外隔离，避免内部信息暴露给外部客户端	增加了调用层次和开销
降低耦合，降低微服务复杂性	防止成为单点故障
统一管控，统一非功能性扩展	防止成为性能瓶颈

可用性方案

高可用性
不宕机更新

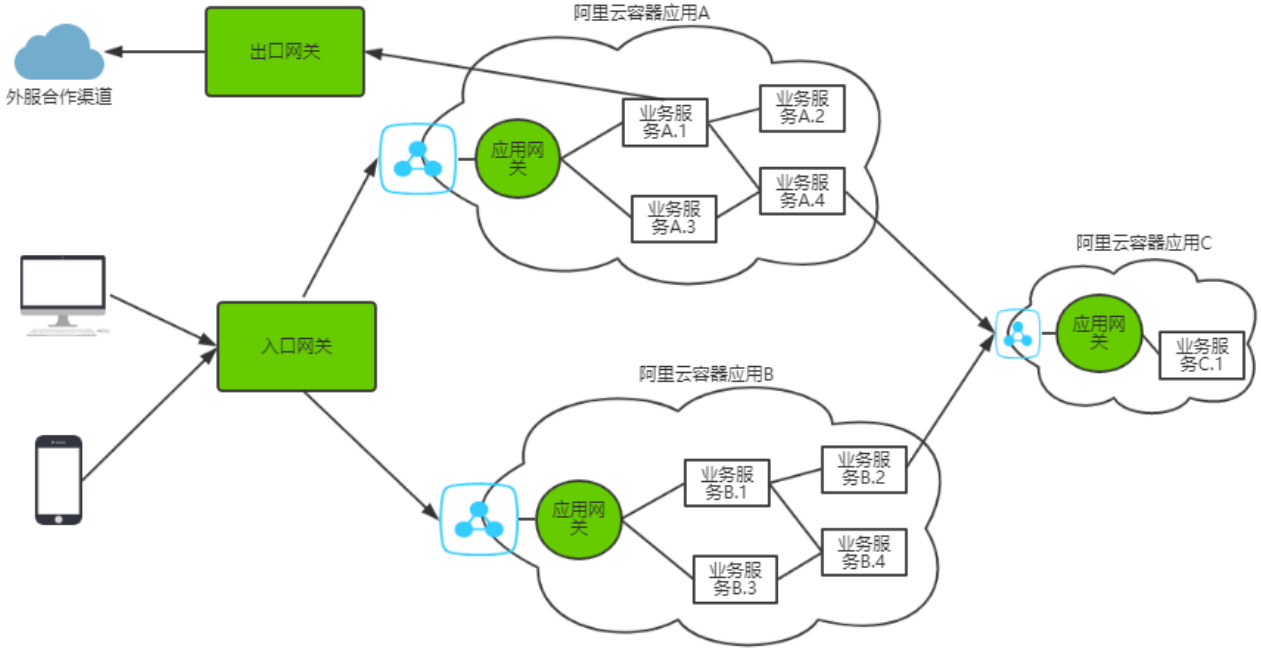


性能优化方案



- 应用缓存
- 动态控制日志级别
- 容器化水平扩展

买单侠API网关方案


出口网关
外服合作渠道
阿里云容器应用A
应用网关
业务服务A.1
业务服务A.2
业务服务A.3
业务服务A.4
入口网关
阿里云容器应用C
应用网关
业务服务C.1
阿里云容器应用B
应用网关
业务服务B.1
业务服务B.2
业务服务B.3
业务服务B.4

买单侠API网关方案

网关类型	网关方案	显著优势	具体应用
应用层网关	Spring Cloud Zuul	适合JAVA开发团队 Spring Cloud Netflix结合紧密	服务分组管控 灰度发布 熔断监控 容器化迁移等
入口网关	Nginx（Kong）	运维友好，适合于没有专门网关研发团队 但需要拥有高性能网关能力的团队	入口管理 跨域 安全管控等
出口网关			出口管理 身份认证 审计等

技术公众号

秦苍科技geek们的集散地

THANK YOU

林丹