网易大规模容器化与Service Mesh实践

刘超 网易轻舟首席架构师

网易轻舟微服务
NetEase QingZhou

第一阶段：应用层和基础设施层各自为战

开发写代码

运维管理资源负责部署

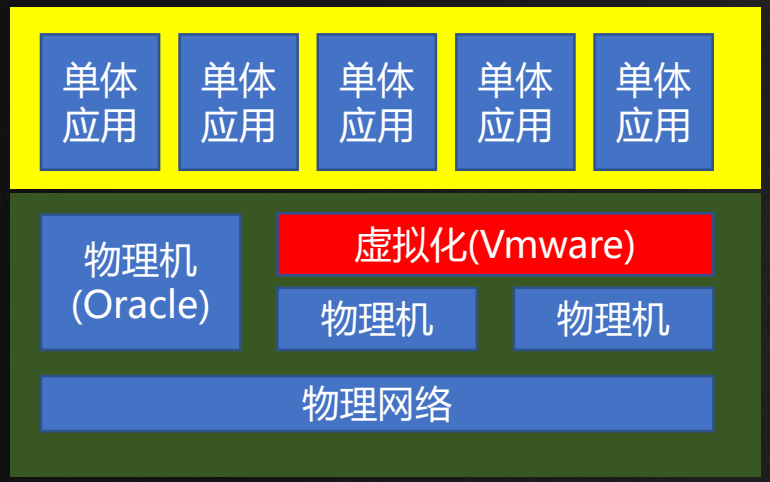
应用层：
- 多为单体应用，应用数目少

资源层：
- 应用和数据库多为物理机部署
- 应用直接使用物理网络
- 数据库使用Oracle

发布模式：
- 开发运维隔离严重
- 发布处于手工化阶段

存在问题：
- 资源申请慢，粒度不灵活
- 资源复用难，进程相互影响
- 上线依赖于人，部署风险高

第一阶段：应用层和基础设施层各自为战



开发写代码

运维管理资源
负责部署

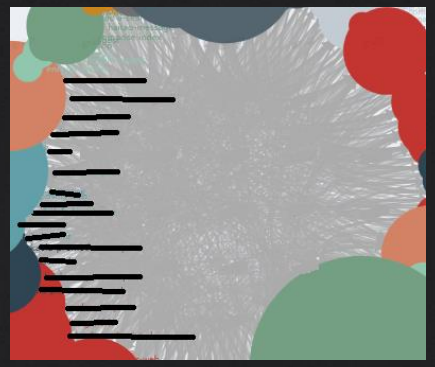
应用层：
- 单体应用增多，成为烟囱

资源层：
- 数据库继续使用Oracle，物理机部署
- 应用使用虚拟化Vwmare进行部署
- 直接使用物理网络

发布模式：
- 虚拟化使得资源点即可得，粒度灵活，复用灵活，隔离性好
- 运维开始感受到压力，通过批量脚本解放人力
- 发布处于脚本化阶段

存在问题：
- 发布脚本复杂逻辑不好实现
- 发布脚本多样，不成体系，难以维护
- 虚拟机依赖人工调度(Excel)，共享存储限制规模
- 高可用依赖底层FT，HA，DR机制，无法区分业务优先级
- 网络，虚拟化，存储等基础设施无抽象化概念，复杂度高，全面依赖运维，大量审批

第二阶段：应用层服务化，自研微服务体系



业务日渐复杂，版本更新迭代遭遇瓶颈

服务治理平台——密密麻麻的调用关系

引入服务化架构：

- 工程拆分 --> 服务拆分
- 服务统一注册、发现
- RPC透明封装

网易轻舟微服务
NetEase QingZhou

第二阶段：应用层服务化，自研微服务体系

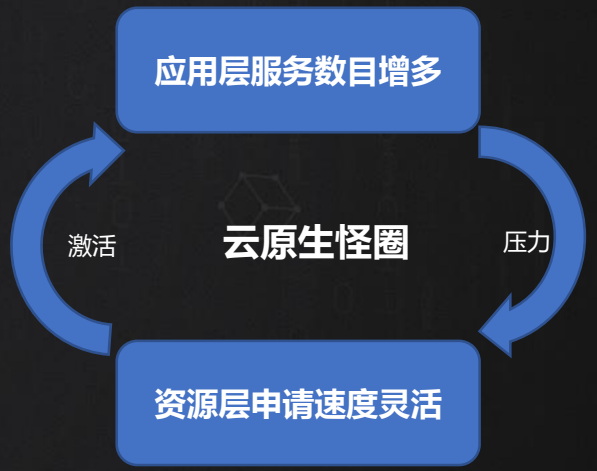
引入服务化架构后直接产生的一些问题与对策：

问题	对策
服务雪崩 (Failure Cascading)	Bulkheading (Thread Pool、Queue)/Fail Fast
大量请求堆积、故障恢复慢	引入熔断机制
内网负载均衡维护代价较大，引起较多抖动问题	引入软负载均衡

第二阶段：应用层服务化对基础设施层产生压力

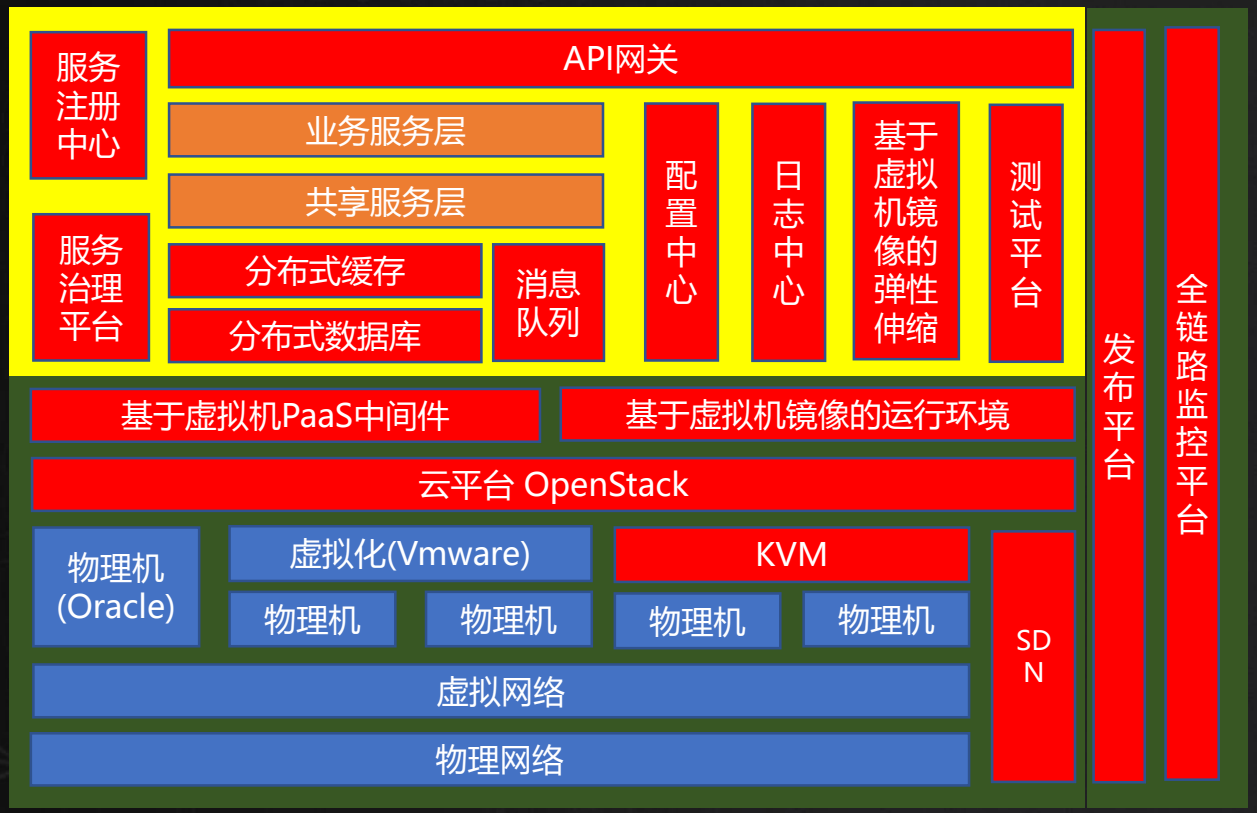
前面问题开始暴露，并引入一些新问题：

- 服务器资源分配困难，服务器机型碎片化
- 一台服务器上多个进程互相影响、QoS难以保障
- 测试、开发环境数量大增，环境管理、部署更新困难



网易轻舟微服务
NetEase QingZhou

第二阶段：基础设施层云化，平台自助化



云计算带来的改变，统一接口，抽象概念，租户自助

- OpenStack实现接口统一，大部分部署工具支持其接口，可基于开源工具实现发布的工具化和平台化
- Flavor抽象资源配比（4G 8G 计算优化型，网络优化型，存储优化型），统一硬件配置，提升利用率，硬件上线效率提升
- 自动调度代替人工调度，区域可用区抽象对机房机架交换机的感知
- 云提供租户概念，有账号子账号体系，有quota，可以让租户在管理员许可的范围内自助操作，加快环境部署速度
- VPC屏蔽物理网络复杂性，冲突问题和安全问题，使得租户可自行配置网络
- 基于虚拟机分层镜像发布和回滚机制，构建发布平台，可实现大规模批量部署和弹性伸缩
- 基于虚拟机的PaaS托管中间件，简化租户创建，运维，调优中间件的难度
- 发布平台提供基于虚拟机镜像+PaaS中间件的统一编排
- 要求业务对于高可用性设计要在应用层完成

第三阶段：基础设施层提供统一应用层框架(服务治理)

微服务框架与开源技术栈统一

将服务治理逻辑抽离、以无侵入方式实现、支持Spring Cloud、Dubbo等开源技术栈

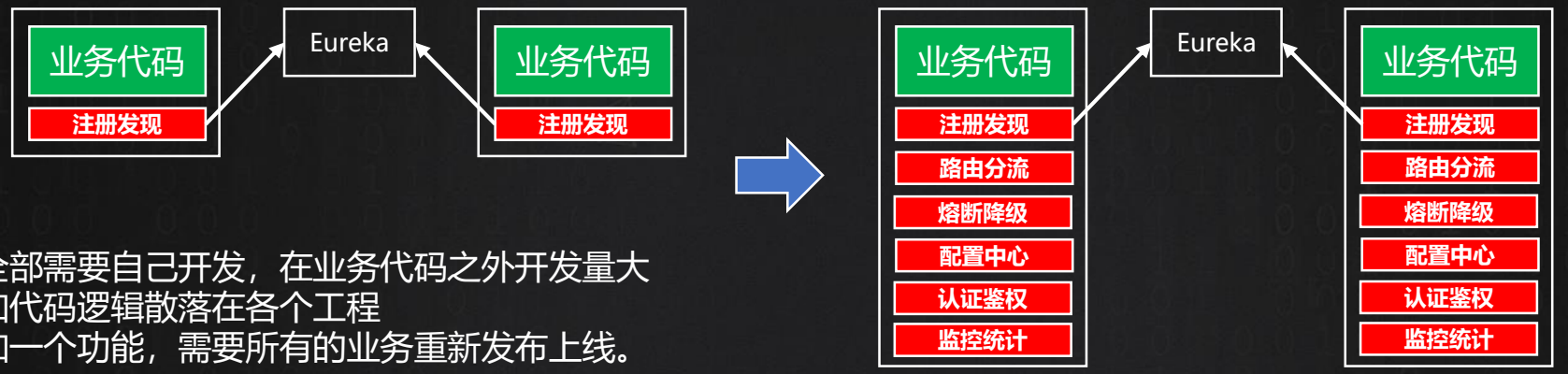
打造CI/CD服务

基于Jenkins，衔接云计算环境、支持通过Agent部署软件包

抽象出产品、环境等多级概念，匹配实际研发情境

网易轻舟微服务
NetEase QingZhou

统一微服务框架之前



- 功能全部需要自己开发，在业务代码之外开发量大
- 配置和代码逻辑散落在各个工程
- 每增加一个功能，需要所有的业务重新发布上线。

服务发现(Eureka)

1. 添加 eureka 依赖

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

2. 添加 @EnableEurekaClient 注解

```
@EnableEurekaClient
public class Application
```

3. 在 application.yml 或者 application.properties 中添加配置

```
eureka:
  instance:
    leaseRenewalIntervalInSeconds: 1
    leaseExpirationDurationInSeconds: 2
  client:
    serviceUrl:
      defaultZone: http://127.0.0.1:8761/eureka/
```

错误容忍(Hystrix)

```
@Configuration
public class HystrixConfiguration {

  @Bean
  public HystrixCommandAspect hystrixAspect() {
    return new HystrixCommandAspect();
  }

}

public class UserService {
...
    @HystrixCommand
    public User getUserById(String id) {
        return userResource.getUserById(id);
    }
}
...
```

负载均衡(Ribbon)增强示例

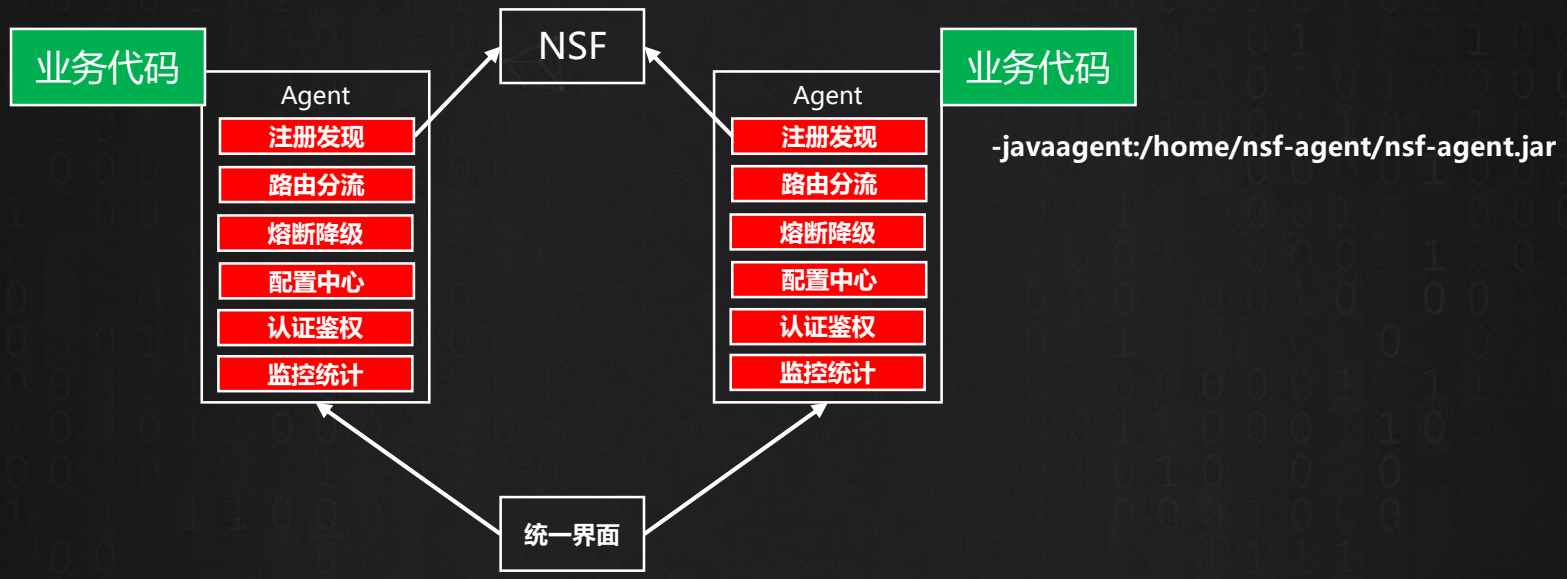
```
@Configuration
public class Config {
    @LoadBalanced
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

```
@Autowired
private RestTemplate restTemplate;

public MessageWrapper<Customer> getCustomer(int id) {
    Customer customer = restTemplate.exchange( "http://customer-service/customer/{id}", H
    return new MessageWrapper<>(customer, "server called using eureka with rest template"
}
```

网易轻舟微服务
NetEase QingZhou

统一微服务框架之后



- 功能全部集中在agent中，开发仅需关注业务代码
- 配置在界面统一操作
- 增加或者删除功能，可通过界面配置实时更新

网易轻舟微服务
NetEase QingZhou

第三阶段：基础设施层提供统一应用层框架(APM，API网关)

新问题与对策：

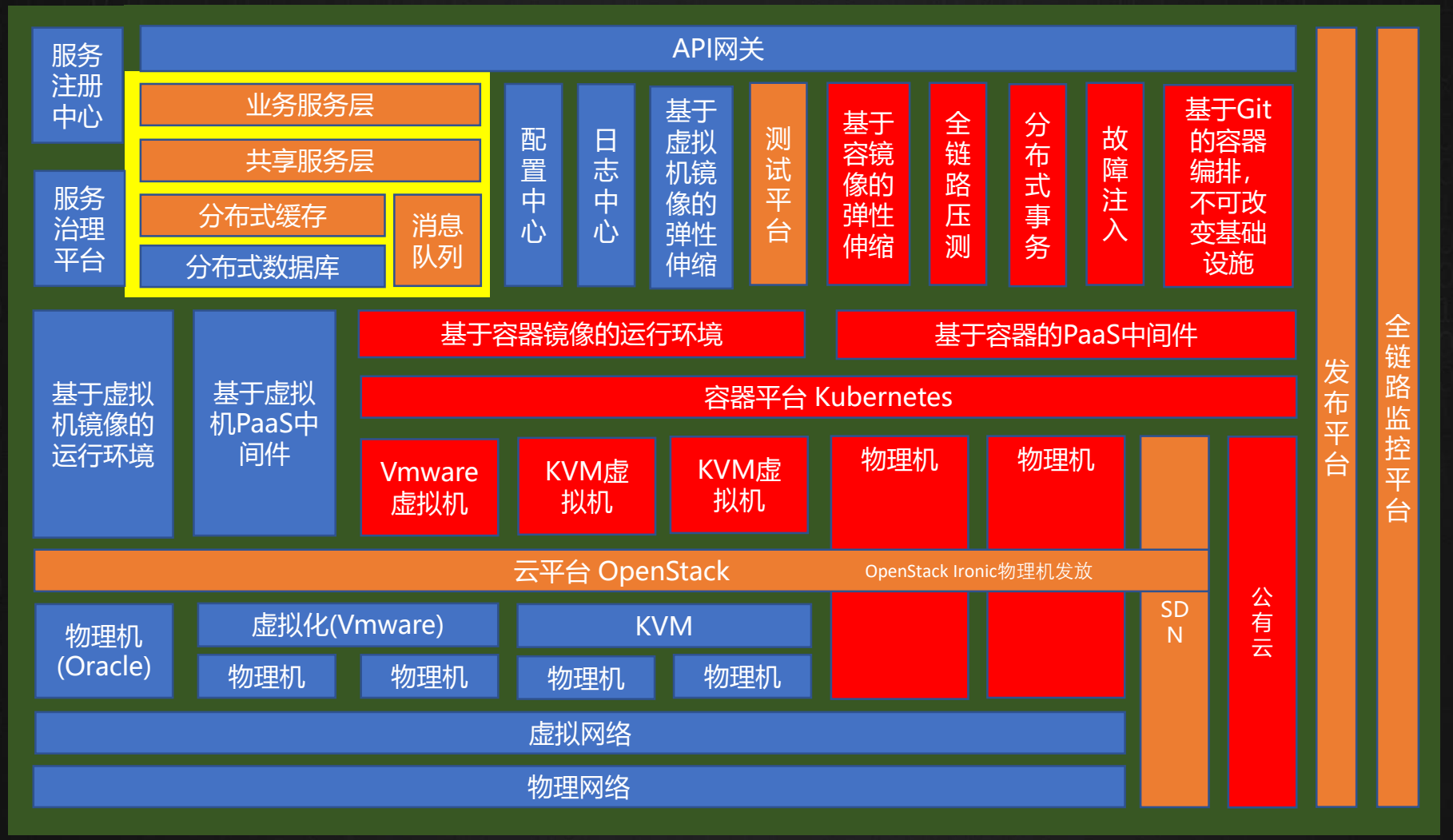
问题	对策
排障复杂度高，传统监控服务无法满足需求	引入全链路跟踪服务 全链路跟踪服务与日志服务依据ID进行联系，以发现故障点上下文 近期支持OpenTracing 标准
难以进行演练、故障频出	引入故障注入服务
API版本混乱	一组服务通过API网关暴露服务 引入API管理、测试平台 自动Client SDK生成

第三阶段：基础设施层提供统一应用层框架(容器，分布式事务)

新问题与对策：

问题	对策
CD服务模版管理混乱， 模版适用的镜像环境基准难以强制， 出现上千个无法重用的模版	2015年起拥抱容器标准
服务器生命周期管理复杂， 服务器下线或转移流程冗长	零散实现Auto Scaling 2015年拥抱Kubernetes标准
分布式事务支持方式多样	实现TCC中间件、事务消息队列等设施

第三阶段：基础设施层提供统一应用层框架

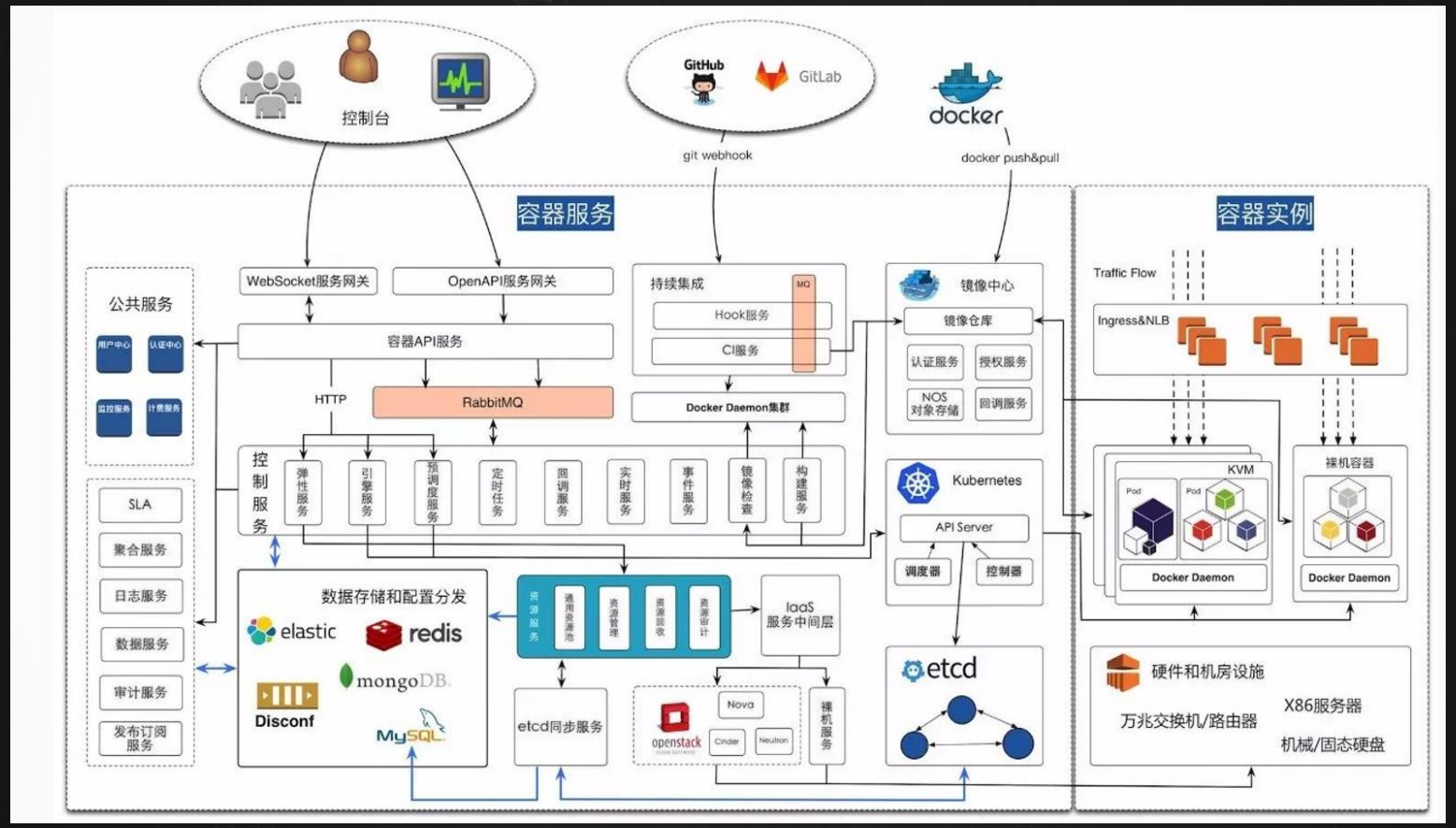


网易轻舟微服务
NetEase QingZhou

第三阶段：基础设施层提供统一应用层框架

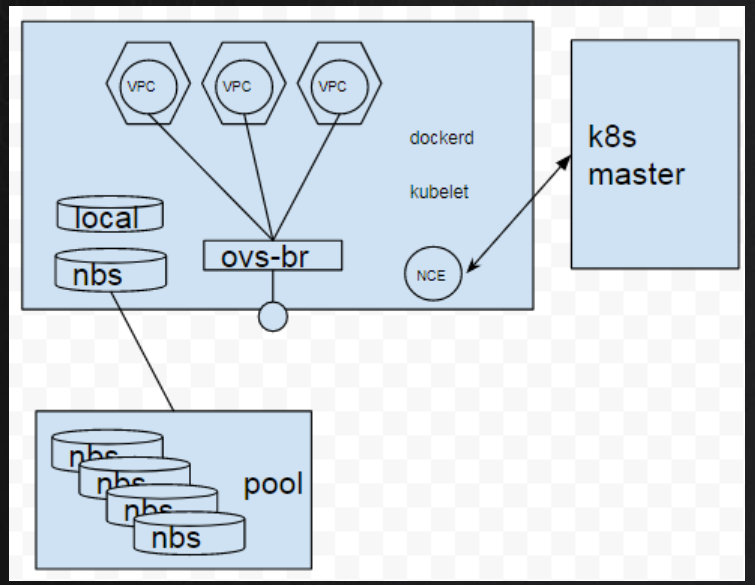
- 容器化要兼容虚拟机部署模式，逐渐迁移
- Kubernetes服务治理能力差，不适合治理大规模微服务
- SpringCloud和应用层耦合，老业务迁移有困难

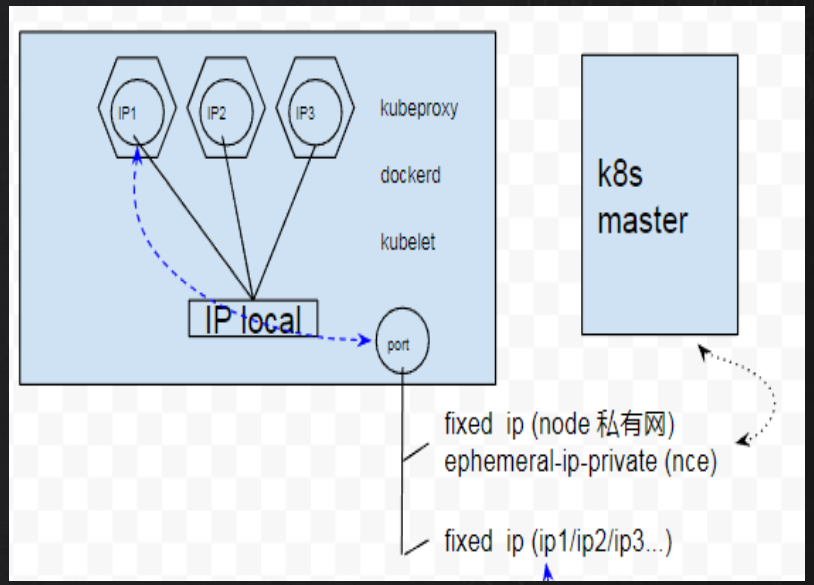
和IaaS深度融合的容器平台



网易轻舟微服务
NetEase QingZhou

网络和存储的深度融合





资源的动态创建

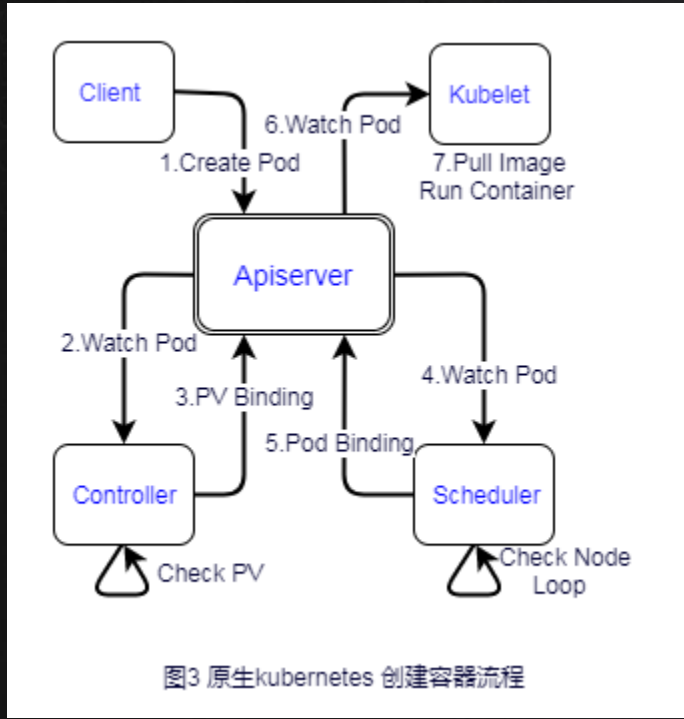


图3 原生kubernetes 创建容器流程

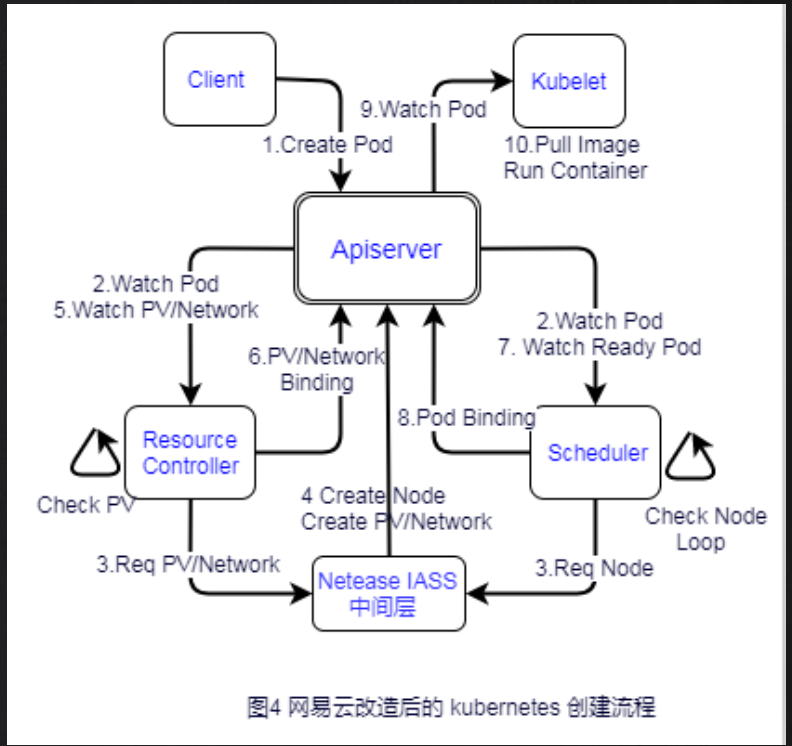


图4 网易云改造后的 kubernetes 创建流程

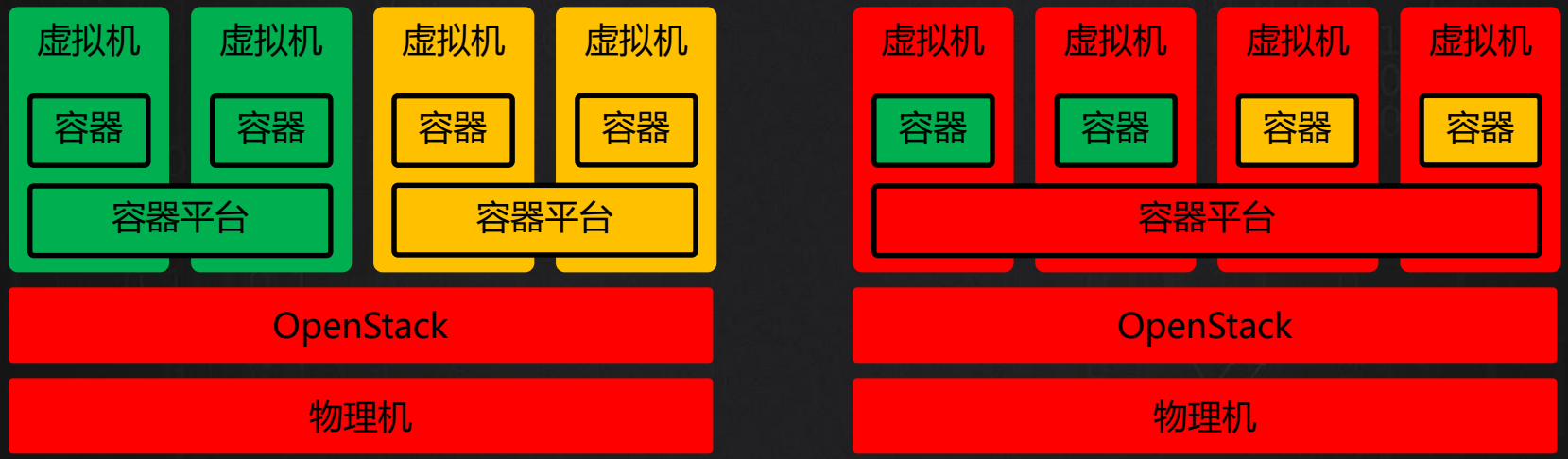
网易轻舟微服务
NetEase QingZhou

Fargate多租户模式

大规模单容器集群，统一由云平台运维，用户仅仅需要关注应用

租户A应用运维人员负责

租户B应用运维人员负责



云计算运维人员负责

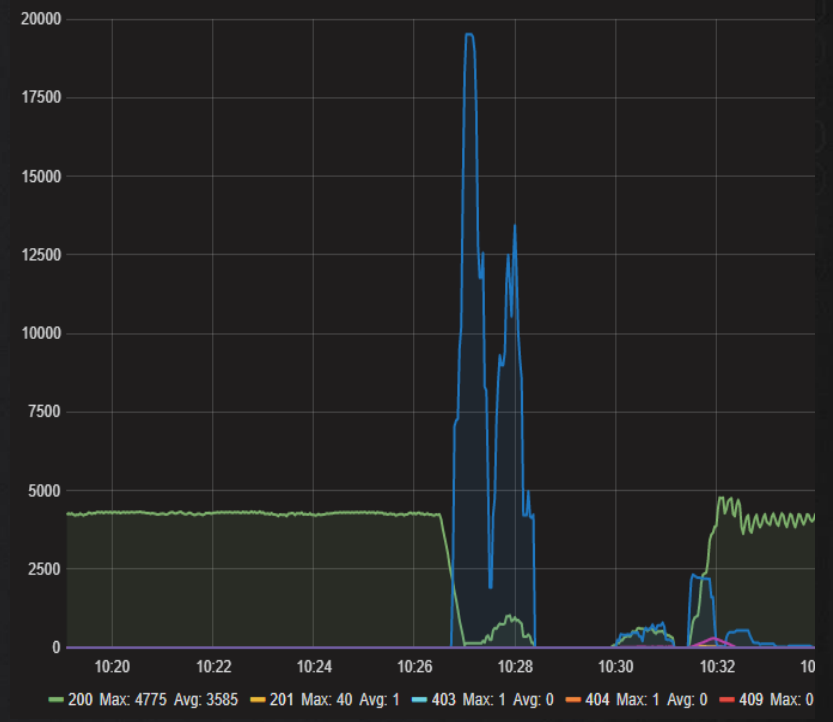
容器层

网易轻舟微服务
NetEase QingZhou

规模问题：大规模容器云性能优化——一个例子

- 例如：APIServer 响应码统计
- 背景：Node – 2.5w / Pod – 15w
- 操作：重启 APIServer
- 现象：
- -平时主要是200(绿色)，4600个每秒
- - 重启429(蓝色)异常增高，最多近2w个
- - 有段时间没有 200 的响应
- - 重启时间长达近 7 分钟

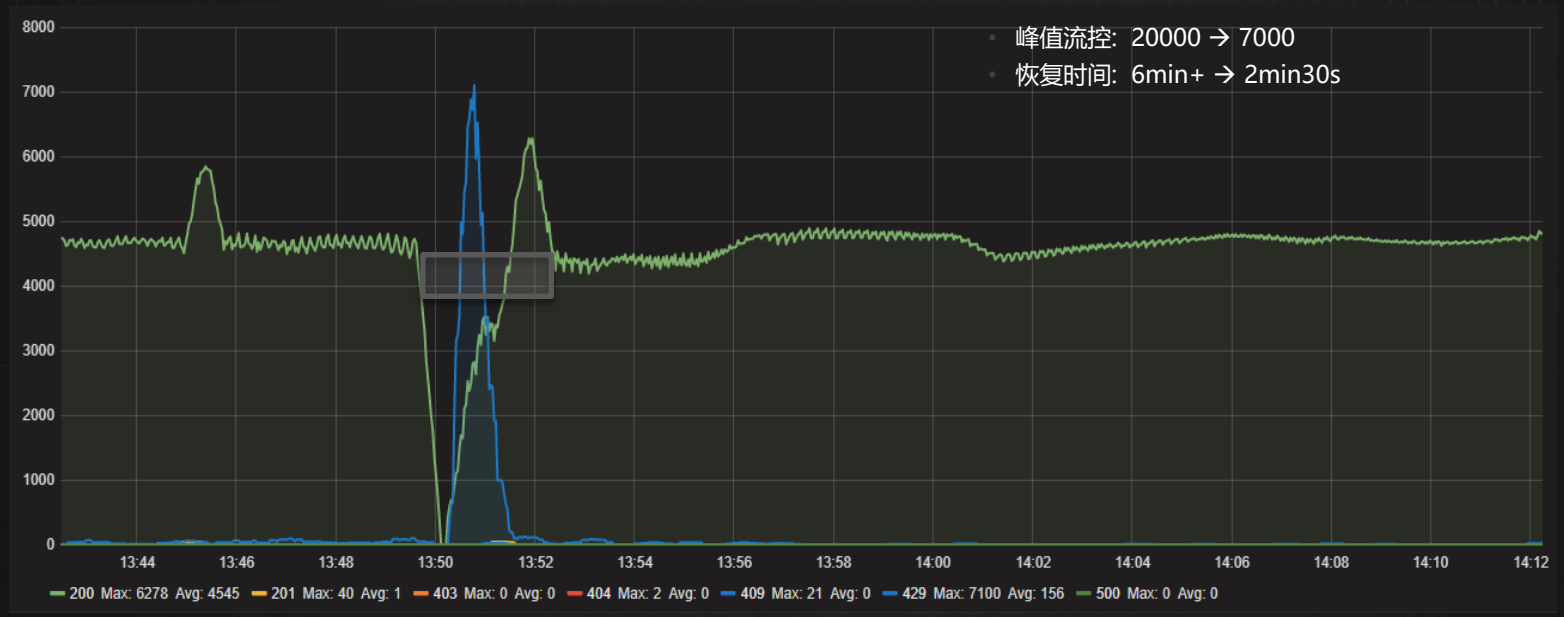
不符合我们的性能通过标准
(重启/升级停服时间<3min)



网易轻舟微服务
NetEase QingZhou

规模问题：大规模容器云性能优化——一个例子

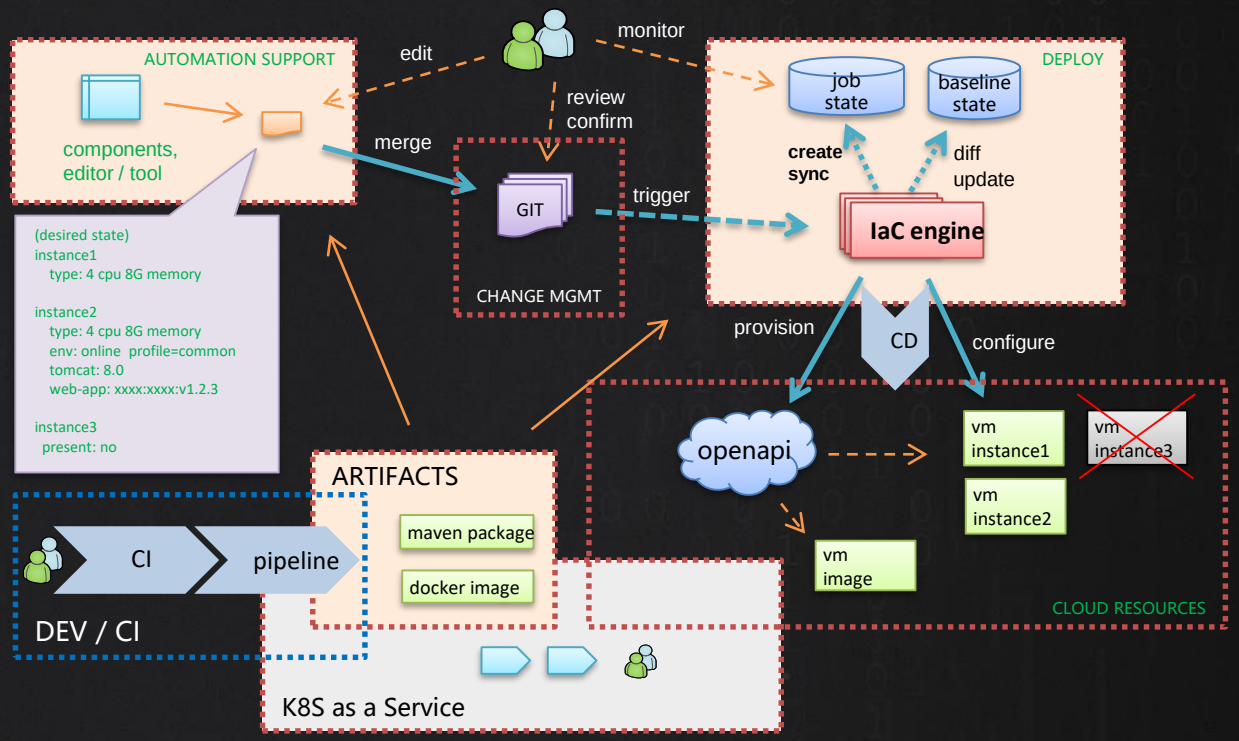
- 多级流控：在读写流控的基础上，根据 UserAgent、Resource、Verb 进行细粒度流控
- 拥塞控制：静态 Retry-After 改为动态值，根据系统当前繁忙程度调节1到8s（token bucket）



网易轻舟微服务
NetEase QingZhou

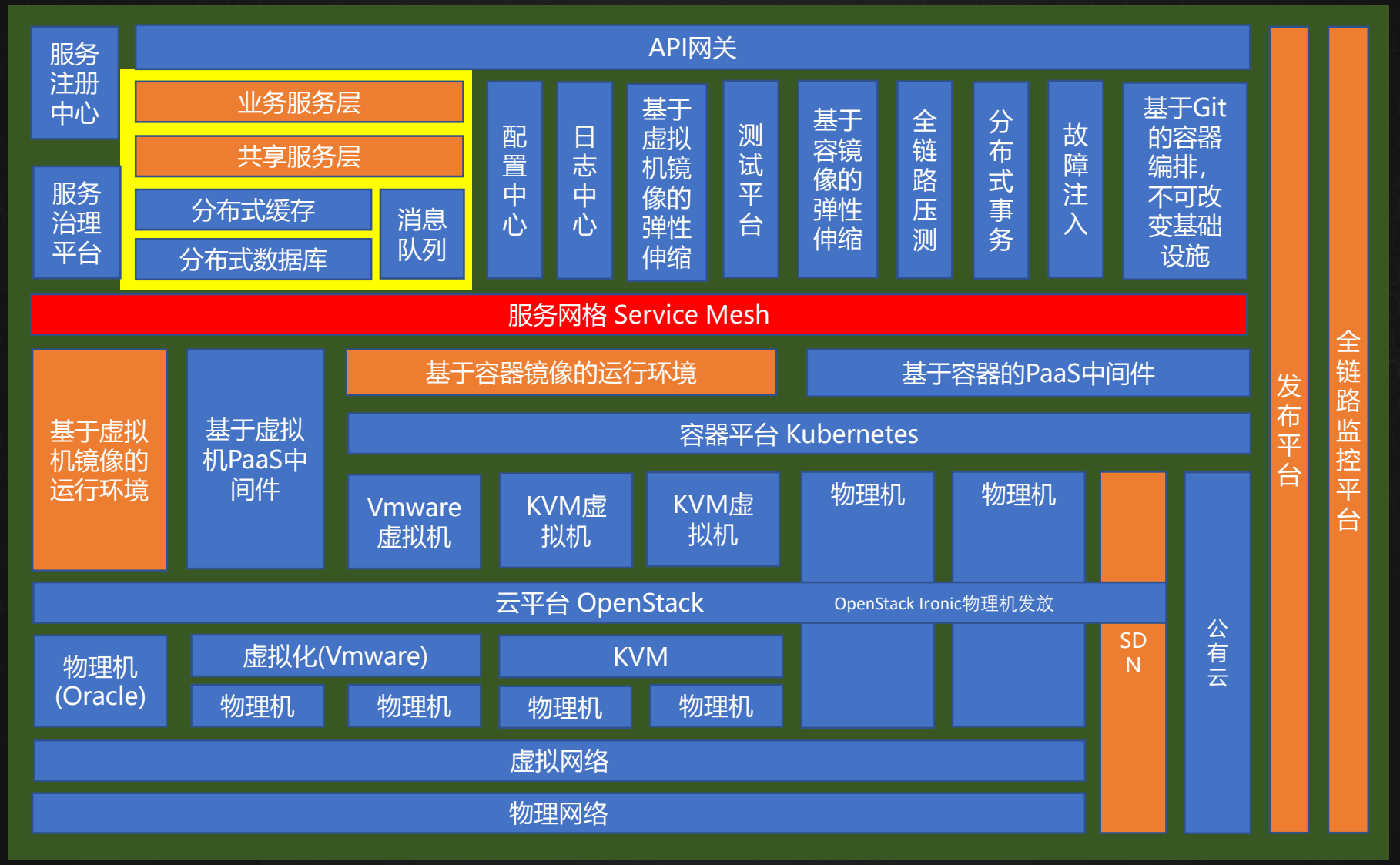
持续集成平台兼容虚拟机和容器

- 代码是代码，配置是代码，单实例运行环境Dockerfile是代码，多实例运行环境编排文件是代码
- 提供标准化的构建
- 支持动态资源申请
- 支持资源编排部署
  - 采用声明式风格
  - 追踪变更，支持计划和审查
  - 支持应用编排
  - 支持模块化开发，可重用
  - 避免繁琐的手工操作



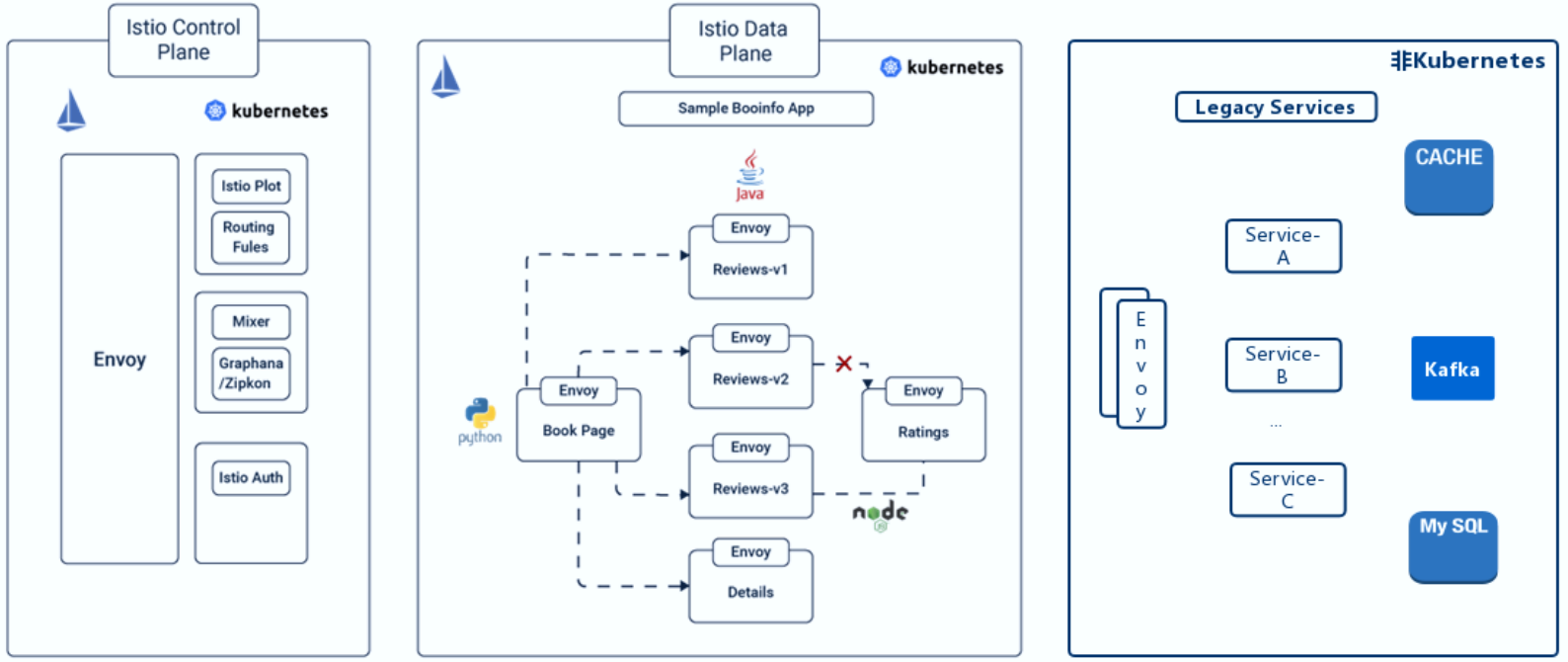
网易轻舟微服务
NetEase QingZhou

第四阶段：基础设施层提供Service Mesh框架

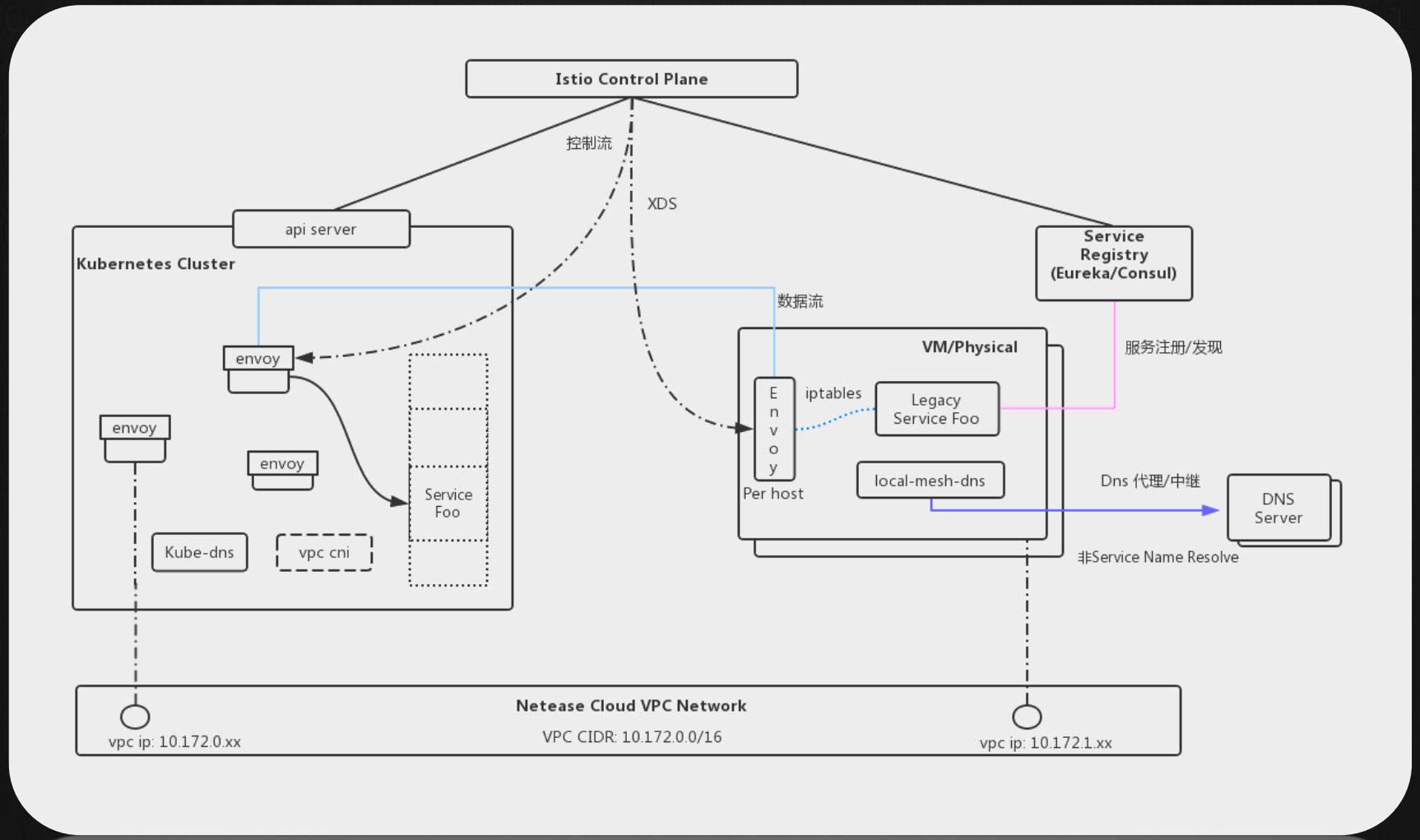


网易轻舟微服务
NetEase QingZhou

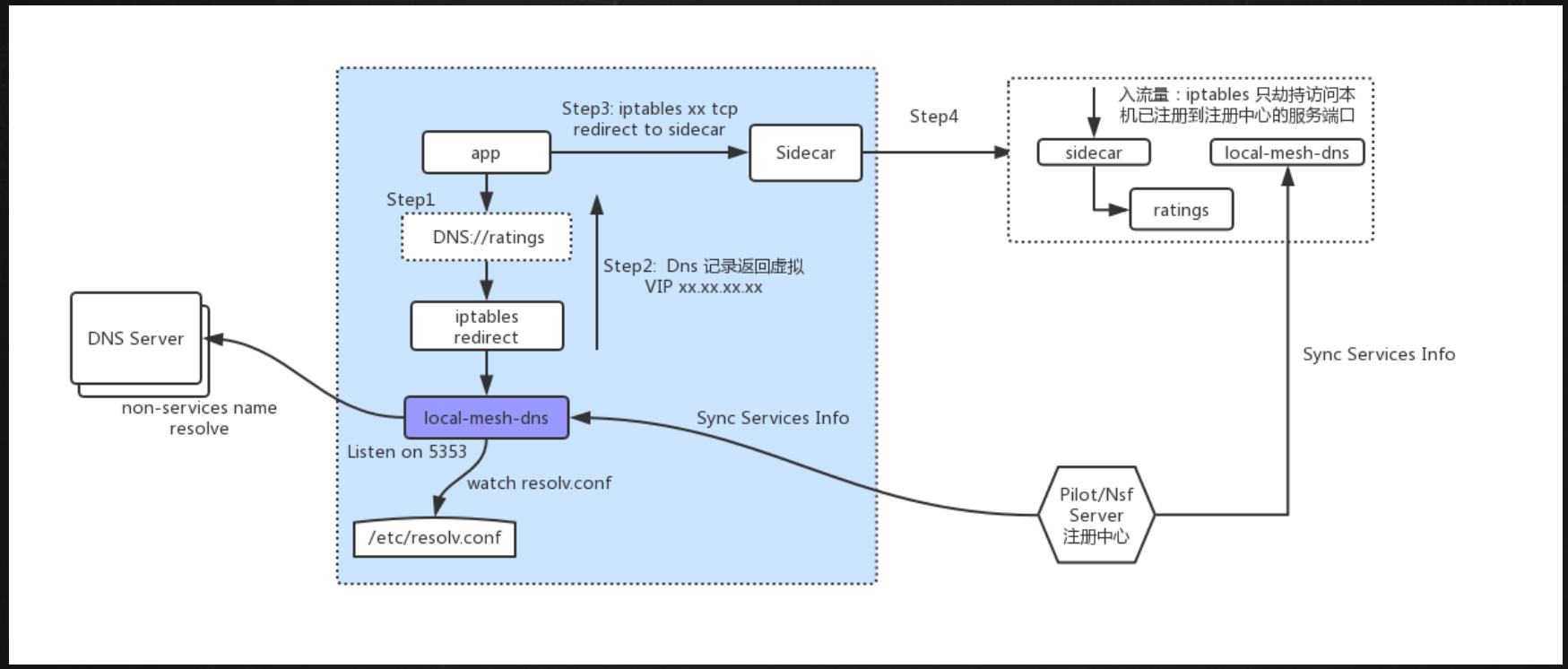
服务网格混部技术



服务网格混部技术

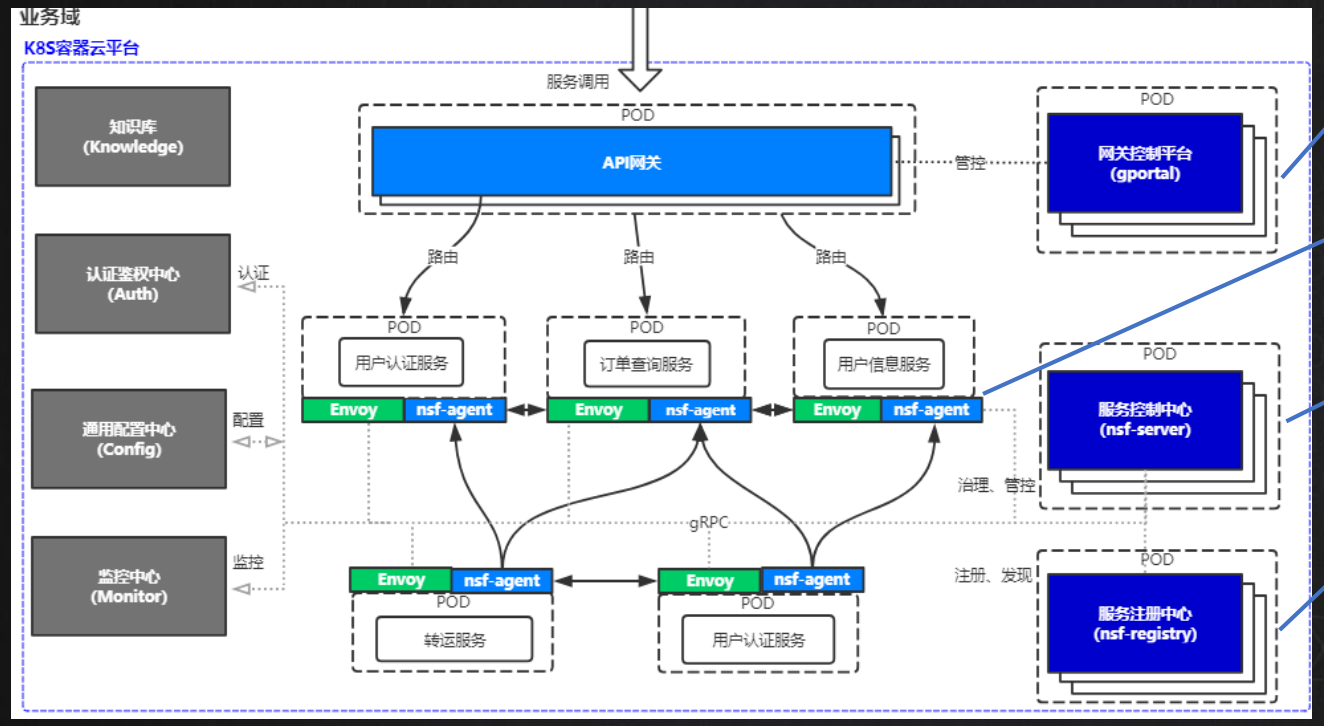


服务网格混部技术



网易轻舟微服务
NetEase QingZhou

微服务治理整体架构



多环境支持
后端服务发现
认证鉴权

大幅减少重复框架代码
统一组件版本配置
多语言支持

服务治理 熔断、降级
流量控制
多数据面接入

服务注册
服务发现
租户隔离性

网易轻舟微服务
NetEase QingZhou

解决了哪些问题?

- 应用减负：通过Agent 和Sidecar 技术，对应用无成本增强
- 开发减负：以微服务治理框架为设计目标，大幅减少重复框架代码，避免重复造轮子;
- 版本控制：统一组件版本配置，避免隐性问题
- 兼容性：兼容的HTTP、RPC调用。兼容非java应用
- 服务治理：根据业务线场景选择治理支持方法级别治理粒度

高性能：更低的性能损耗，并提供更细粒度的服务治理;

网易轻舟微服务
NetEase QingZhou

网易某大型电商产品ServiceMesh迁移实践

现状

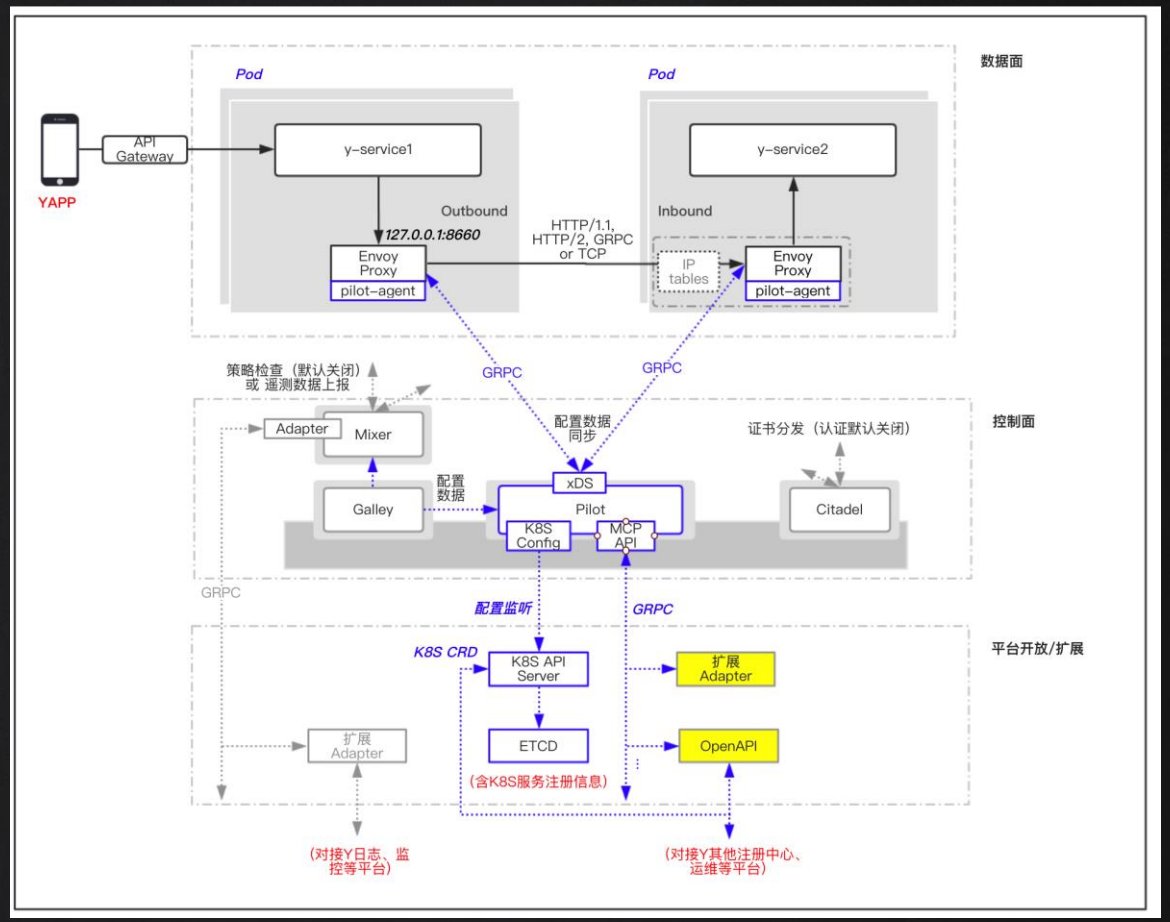
- 以Nginx作为流量出口代理
- 与Consul紧密结合
- 服务发现基本实现业务无感知

挑战

- 数据面能力弱，无控制面
- 维护成本高
- 开源社区偏离
- 遗留服务迁移困难
- ServiceMesh性能优化

总体方案

- 整体基于**Envoy+Istio**方案
- 数据面
  - Outbound指向Sidecar
  - Inbound可配置流量拦截
- **控制面**以Pilot为核心
- **注册中心**以K8S原生方式
- 通过**MCP机制**扩展



网易轻舟微服务
NetEase QingZhou

迁移 —— 跨环境混合部署互访

- **云外 -> 云内**

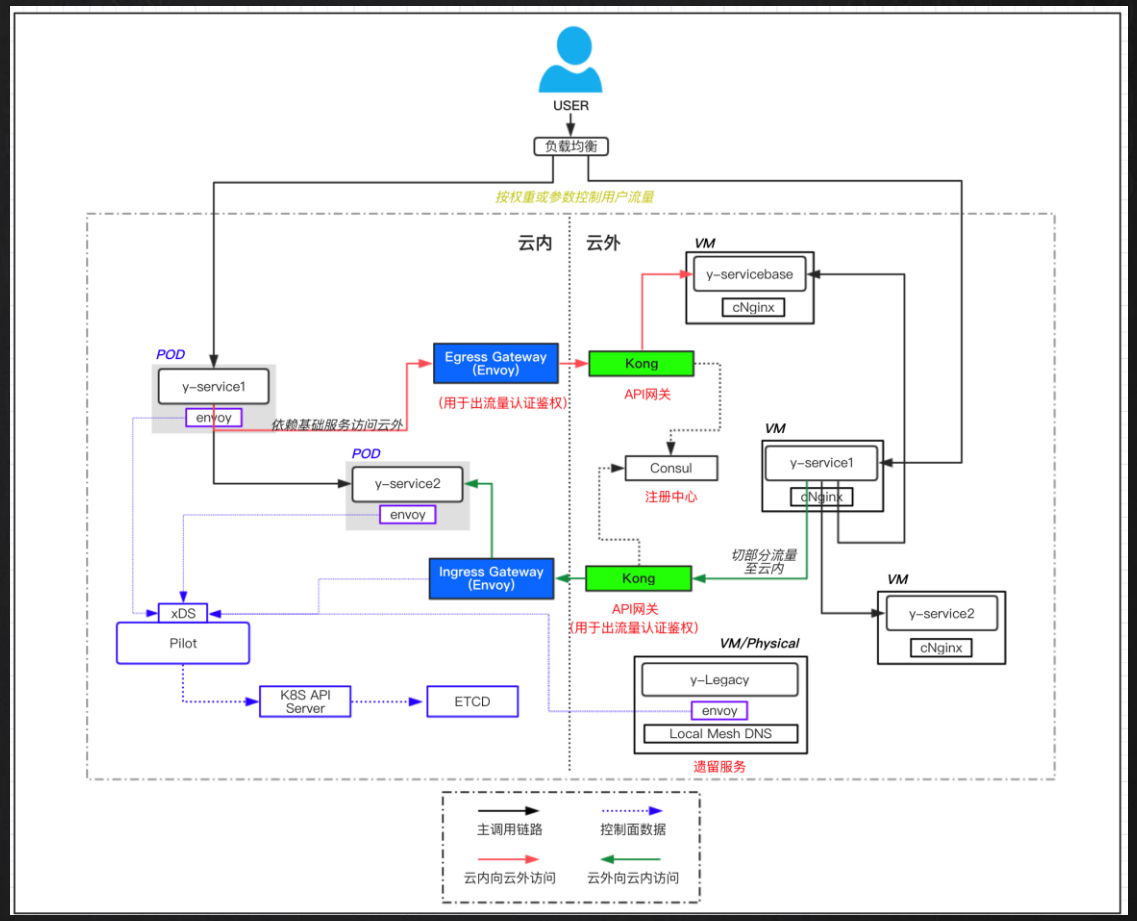
  使用**Istio Ingress Gateway**

- **云内 -> 云外**

  沿用**原有API Gateway**

- **基本迁移原则**
  - 服务间调用灰度引流
  - 用户调用灰度引流

    (基于权重 or 参数)
- **遗留服务处理**
  - 统一纳入Istio管控
  - Local Mesh DNS



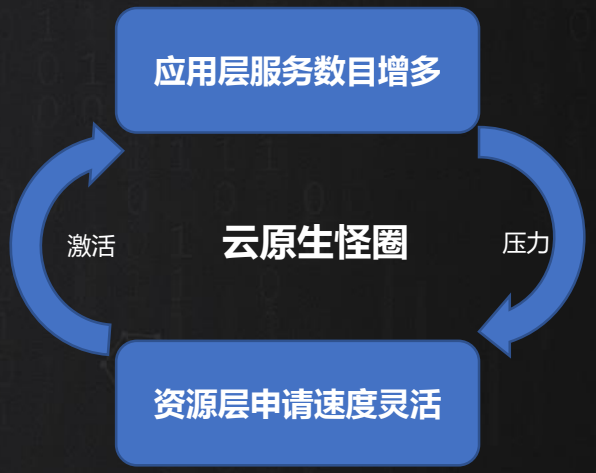
网易轻舟微服务
NetEase QingZhou

容器化带来的问题

服务规模越来越多，增加速度越来越快。

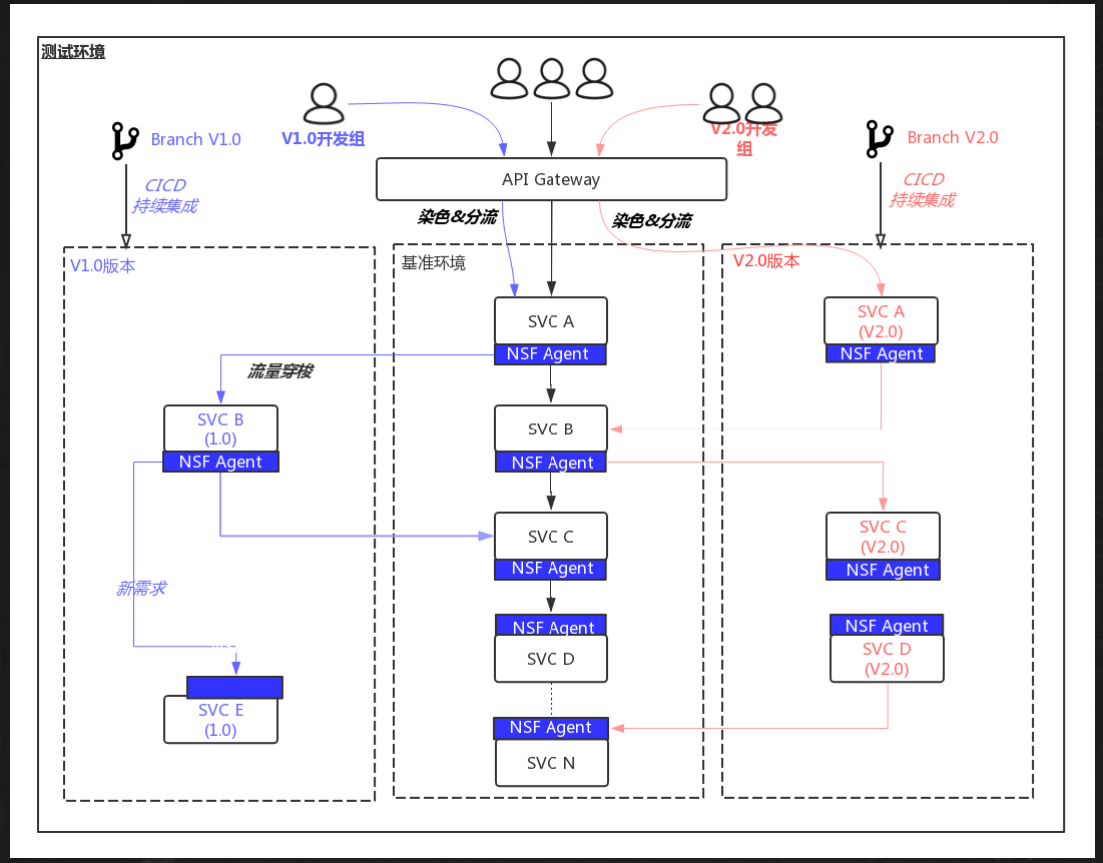
业务线展开后，需求迭代的速度越来越快。

对测试环境的需求指数性增加



网易轻舟微服务
NetEase QingZhou

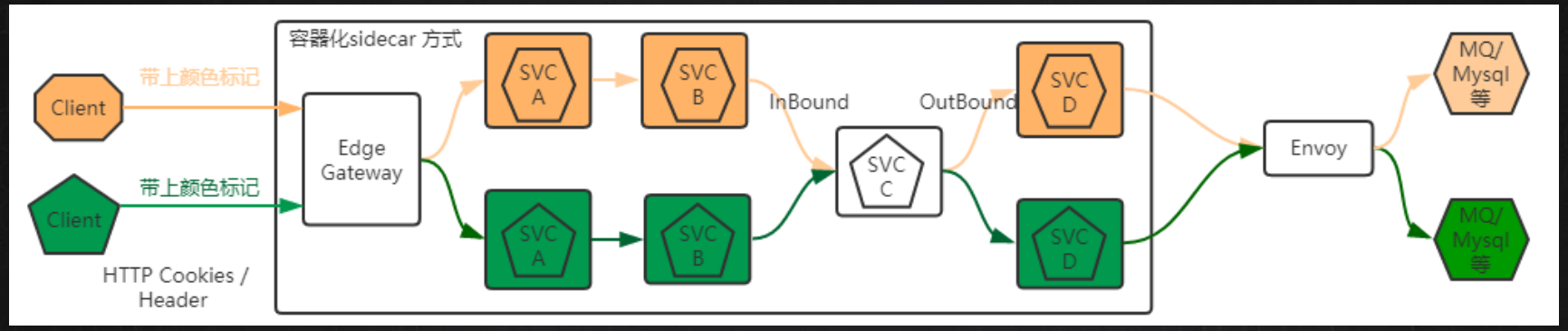
基于流量染色的测试环境管理，仅需部署增量服务

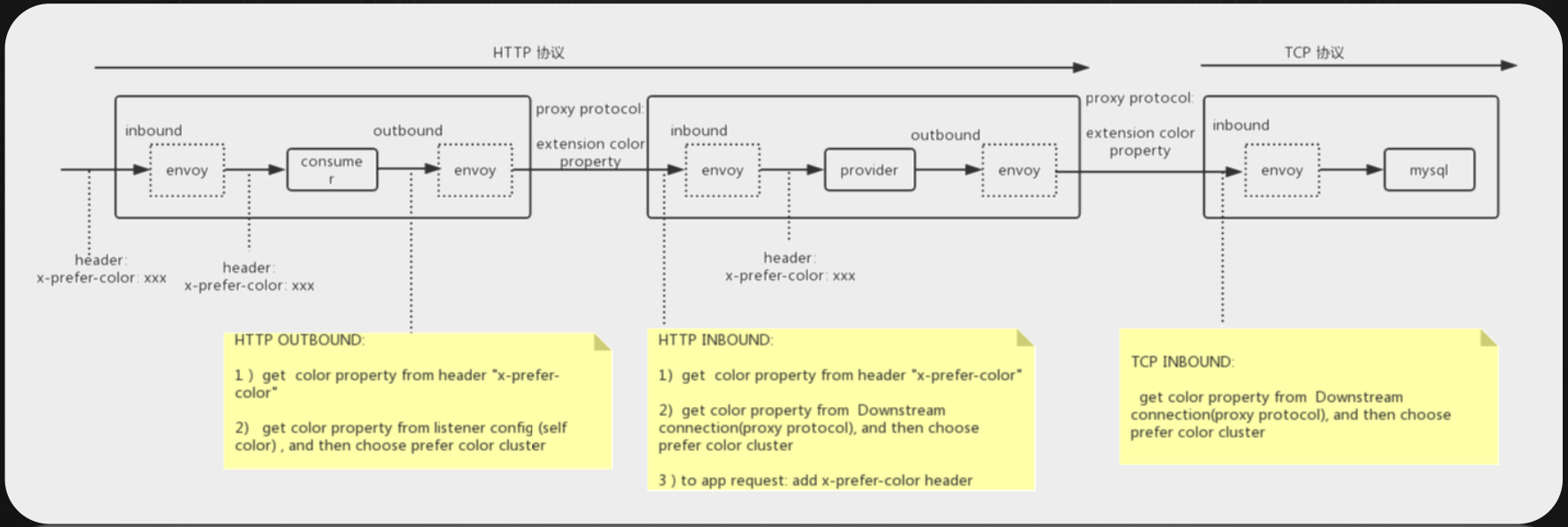


1、整个测试环境共享一套基准环境，部署所有应用。
2、API网关层进行流量染色。
3、NSF Agent携带染色消息，并且染色在调用链上持续传递（小环境调用到基准环境后，还能路由回到小环境），按照同环境优先的策略进行路由和消费。
4、若分支环境缺失相关应用，则路由到基准环境或择由基准换消费。

网易轻舟微服务
NetEase QingZhou

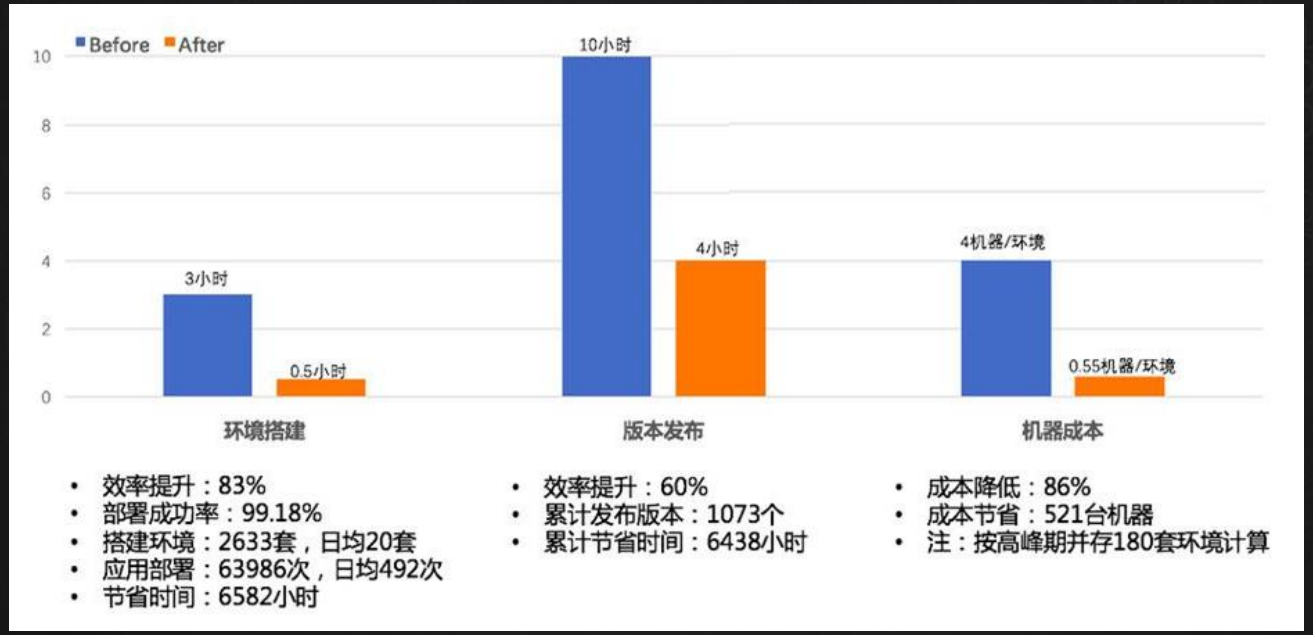
多环境治理实现原理





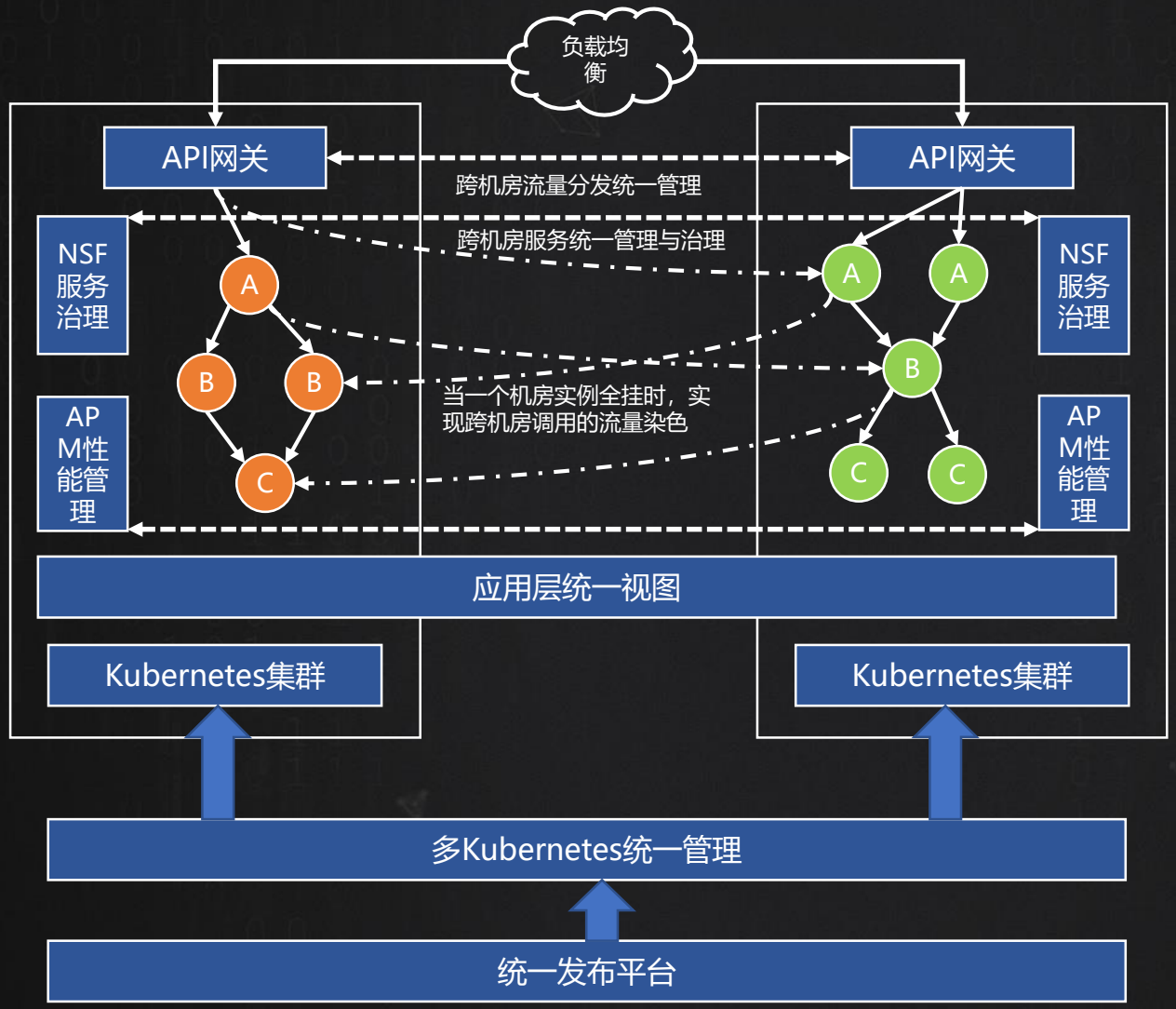
流量染色效果

- 每个环境只需要部署修改了代码的应用（显著降低部署成本）
- 从客户端看来，每个小环境都能提供完整的功能（避免功能不全的干扰）
- 修改的逻辑在小环境中可以得到验证（便于验证,联调）
- 小环境之间彼此独立（隔离）。
- 环境合并和代码合并逻辑一致，统一在发布平台管理，谁后合并谁负责Merge



网易轻舟微服务
NetEase QingZhou

全栈多环境多机房流量染色方案



总-分-总模型

第一个总：统一发布平台对接统一的多K8s管理平台，有统一的灰度发布策略。

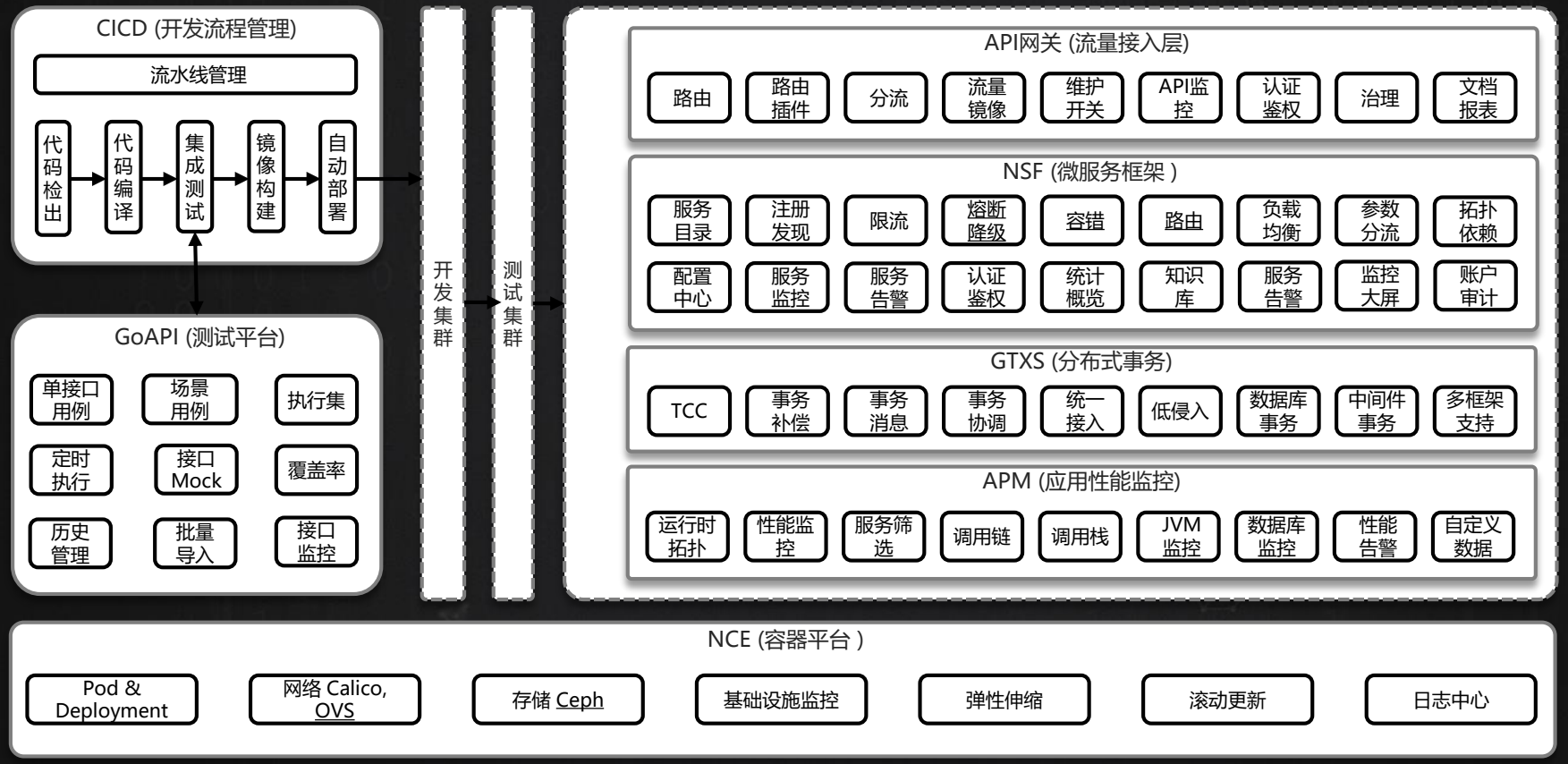
第二个分，多数据中心kubernetes应分开部署的，不建议一套Kubernetes跨多个机房，但要保持网络连通性。

第三个总，应用虽部署在多数据中心的多个Kubernetes中，但应使用统一服务治理中心，形成统一的视图，可通过染色识别当前的kubernetes即可

同机房可部署多套Kubernetes实现A/B测试，也可应对Kubernetes的升级风险问题

网易轻舟微服务
NetEase QingZhou

云原生架构的一栈式工具链



网易轻舟微服务
NetEase QingZhou

本PPT来自2019携程技术峰会

更多信息请关注“携程技术中心”微信公众号~

网易轻舟微服务

NetEase QingZhou