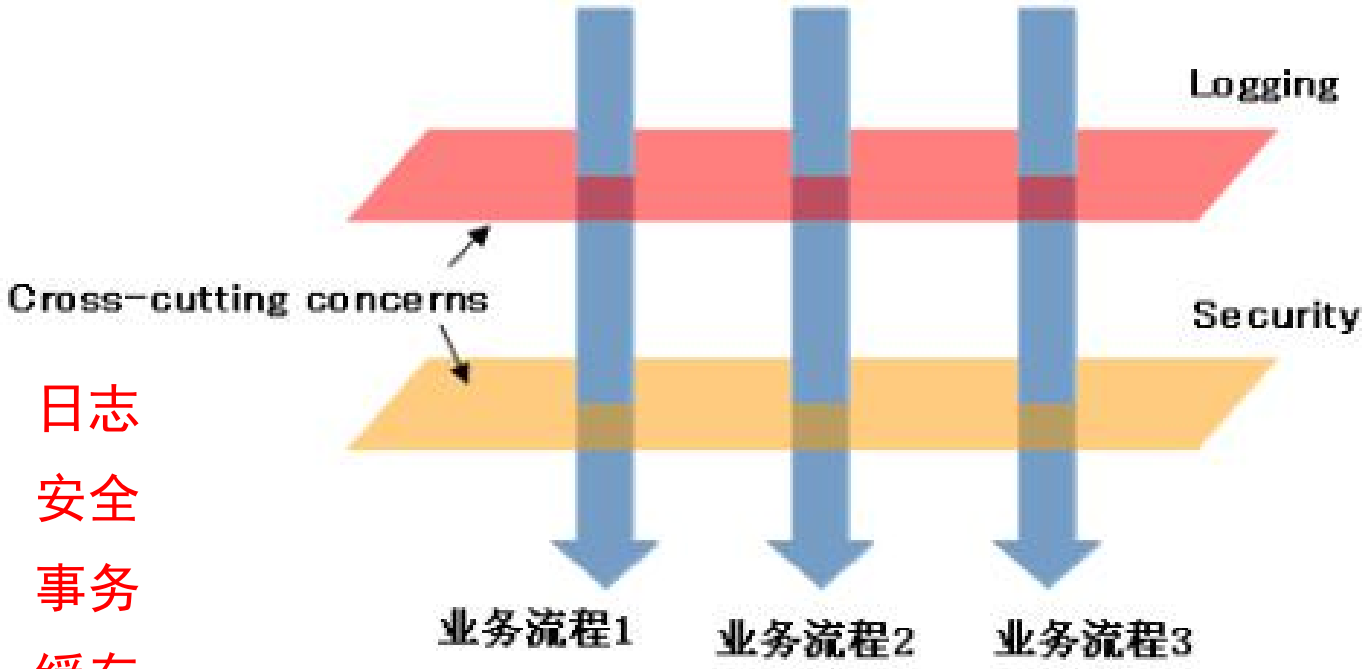
2021-服务端开发

第15节　　服务网关和路由

回忆AOP的横切关注点（cross-cutting concern）

- 日志
- 安全
- 事务
- 缓存



AOP图解

分布式系统的横切关注点

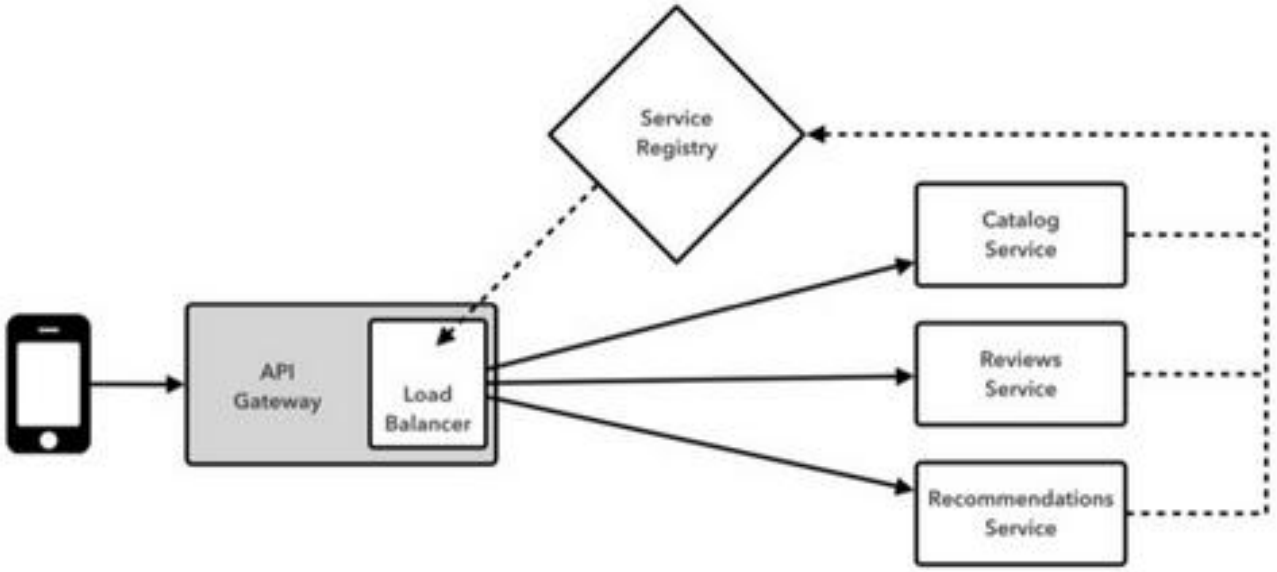
- cross-cutting concern
- 安全
- 日志记录
- 用户跟踪
- 服务网关（service gateway）

服务网关

- 服务网关位于服务客户端和相应的服务实例之间
- 所有服务调用（内部和外部）都应流经服务网关
- 服务网关提供的能力
  - 静态路由
  - 动态路由
  - 验证和授权
  - 度量数据收集和日志记录

Zuul

- Zuul is a gateway service that provides dynamic routing, monitoring, resiliency, security, and more.
- 将应用程序中的所有服务的路由映射到一个URL
- 过滤器



使用Zuul

- pom文件添加依赖

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

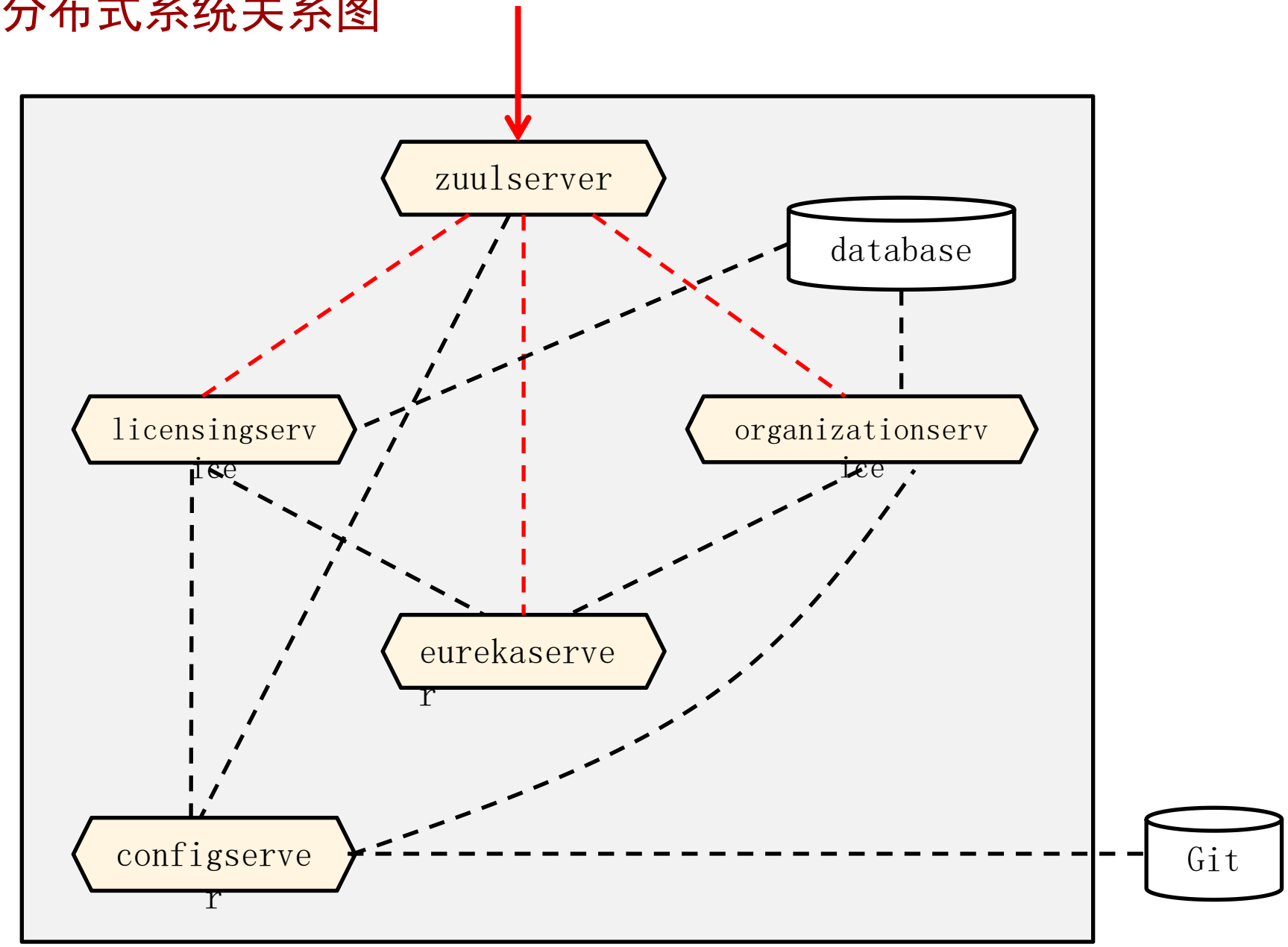
- 启动类添加注解

```
@EnableZuulProxy
```

- 配置zuul与Eureka通信

Eureka、Ribbon

分布式系统关系图


zuulserver
database
licensingservice
organizationservice
eurekaserver
configserver
Git

通过服务发现自动映射路由

- 服务ID
- 需要访问Eureka，有服务才会创建路由

使用服务发现手动映射路由

- application.yml
- ignored-services
- prefix

使用静态URL手动映射路由

- 静态URL是指向未通过Eureka服务发现引擎注册的服务的URL
- 禁用Ribbon与Eureka集成，手动指定负载均衡的服务实例

动态重新加载路由配置

- Git-更新zuulservice配置
- zuul POST:http://localhost:5555/refresh

设置超时

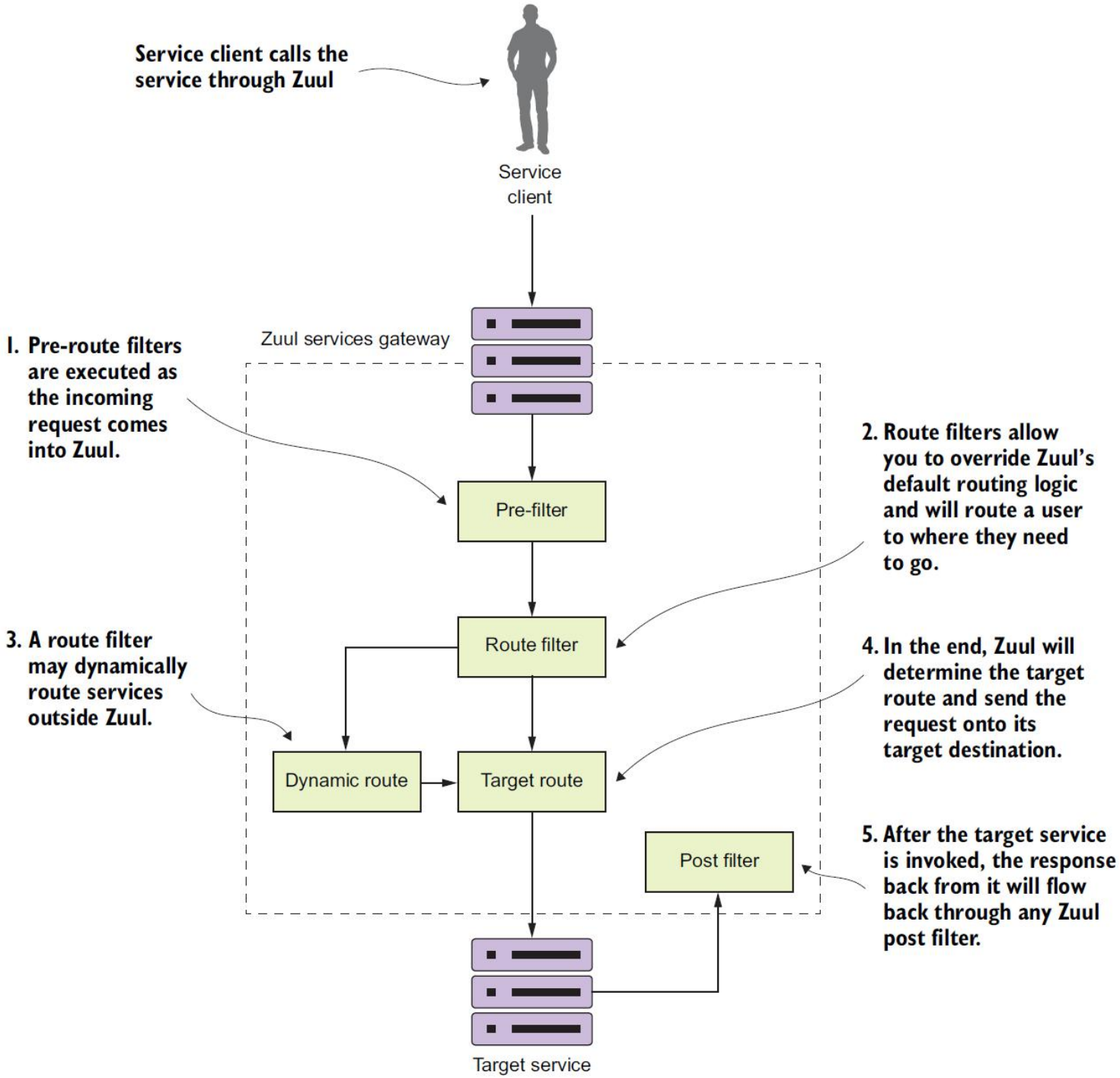
- Hystrix,1S
- Ribbon,5S
- Ribbon的懒加载导致第一次调用慢，引起失败

测试网关侧的负载均衡

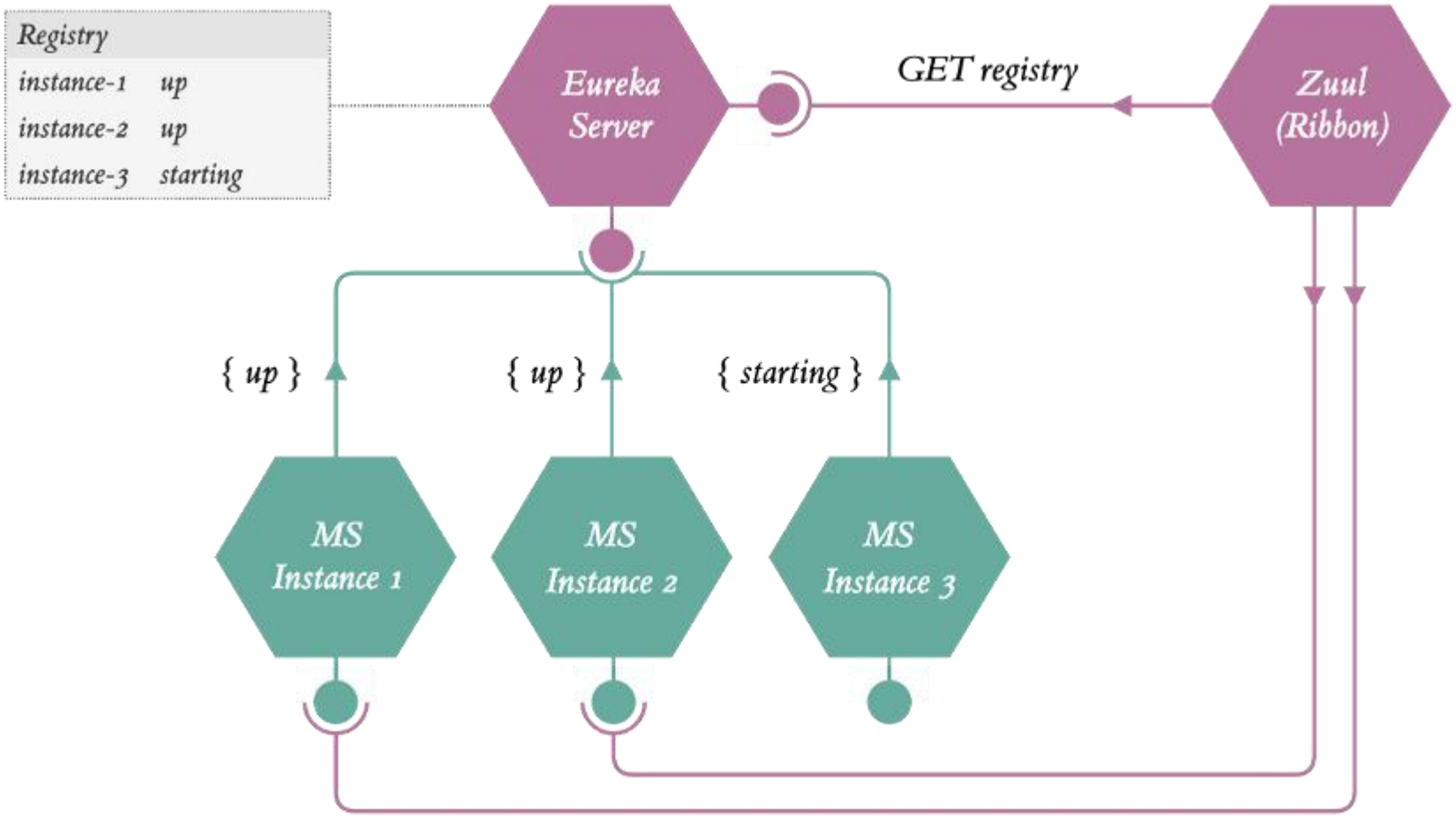
- `docker-compose up  --scale organizationservice=2`

过滤器

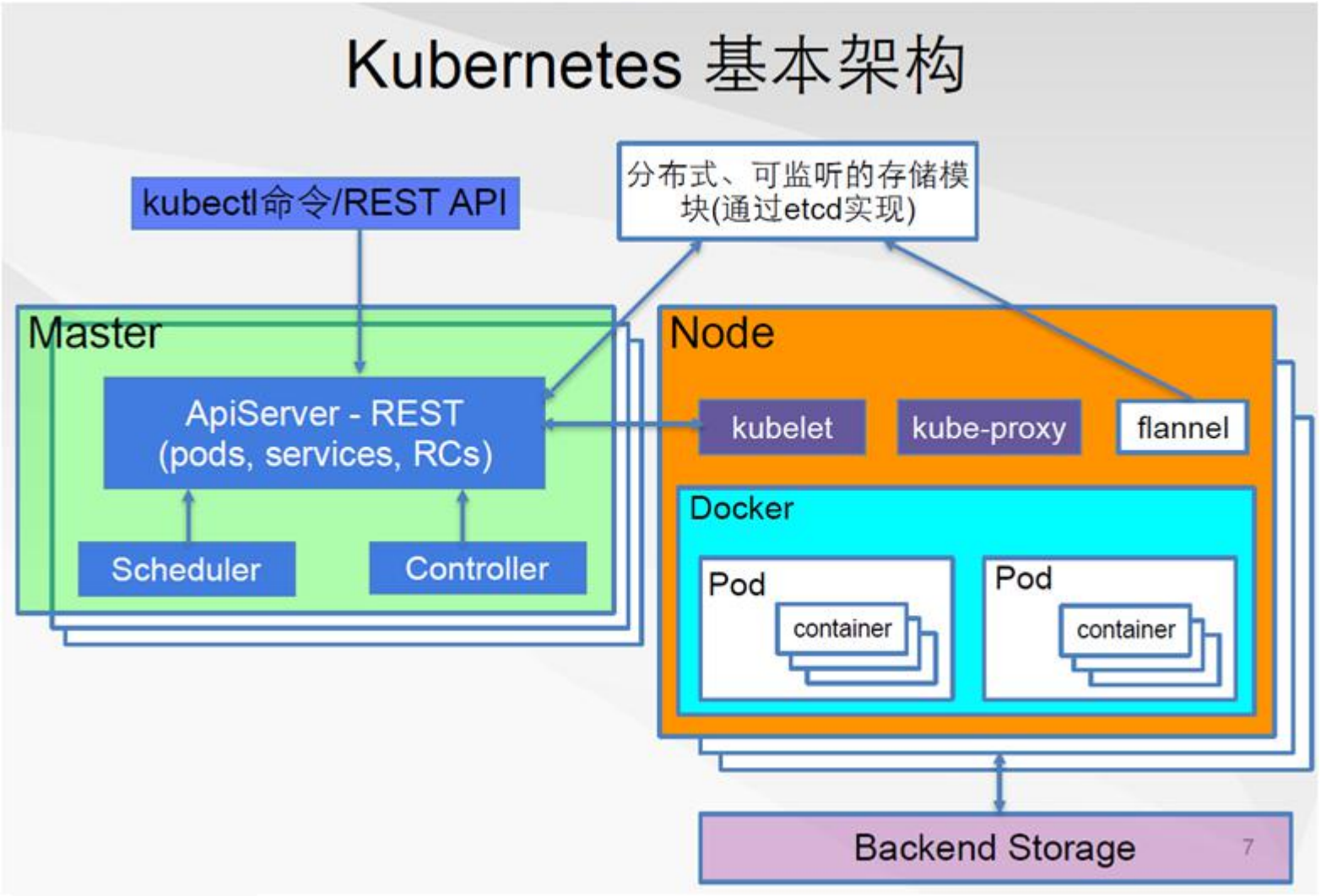
- 使用Zuul和Zuul过滤器允许开发人员为通过Zuul路由的所有服务实现横切关注点
- ZuulFilter
  - 前置过滤器，在Zuul将实际请求发送到目的地之前被调用
  - 后置过滤器，在目标服务被调用并将响应发送回客户端后被调用
  - 路由过滤器，用于在调用目标服务之前拦截调用


Service client calls the service through Zuul
Service client
Zuul services gateway
1. Pre-route filters are executed as the incoming request comes into Zuul.
Pre-filter
2. Route filters allow you to override Zuul's default routing logic and will route a user to where they need to go.
Route filter
3. A route filter may dynamically route services outside Zuul.
Dynamic route
Target route
4. In the end, Zuul will determine the target route and send the request onto its target destination.
Post filter
5. After the target service is invoked, the response back from it will flow back through any Zuul post filter.
Target service

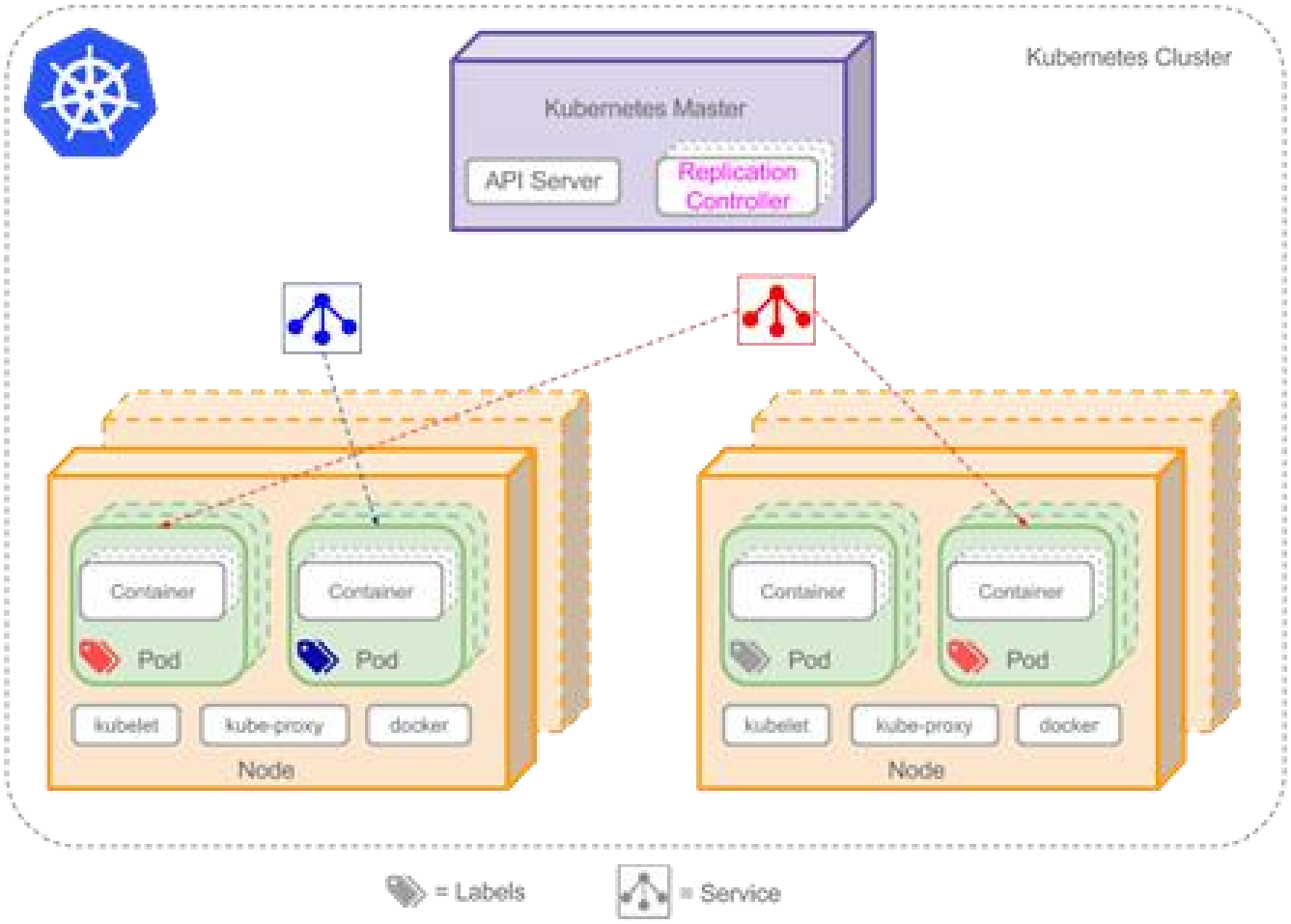
总结


Registry
instance-1 up
instance-2 up
instance-3 starting
Eureka Server
GET registry
Zuul (Ribbon)
{ up }
{ up }
{ starting }
MS Instance 1
MS Instance 2
MS Instance 3

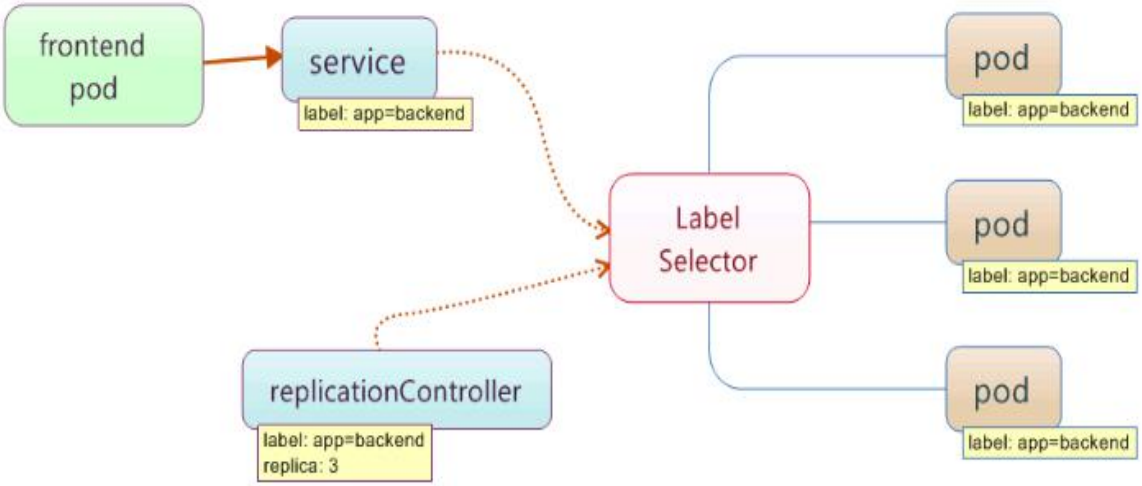
Kubernetes


Kubernetes 基本架构
kubectl命令/REST API
分布式、可监听的存储模块(通过etcd实现)
Master
ApiServer - REST
(pods, services, RCs)
Scheduler
Controller
Node
kubelet
kube-proxy
flannel
Docker
Pod
container
Pod
container
Backend Storage
7

核心概念

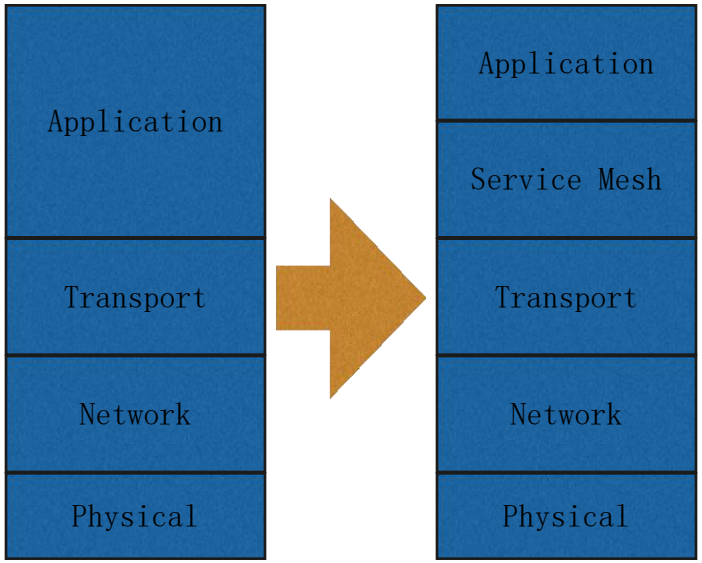

Kubernetes Cluster
Kubernetes Master
API Server
Replication Controller
Container
Container
Pod
Pod
kubelet
kube-proxy
docker
Node
Container
Container
Pod
Pod
kubelet
kube-proxy
docker
Node
= Labels
= Service

Label

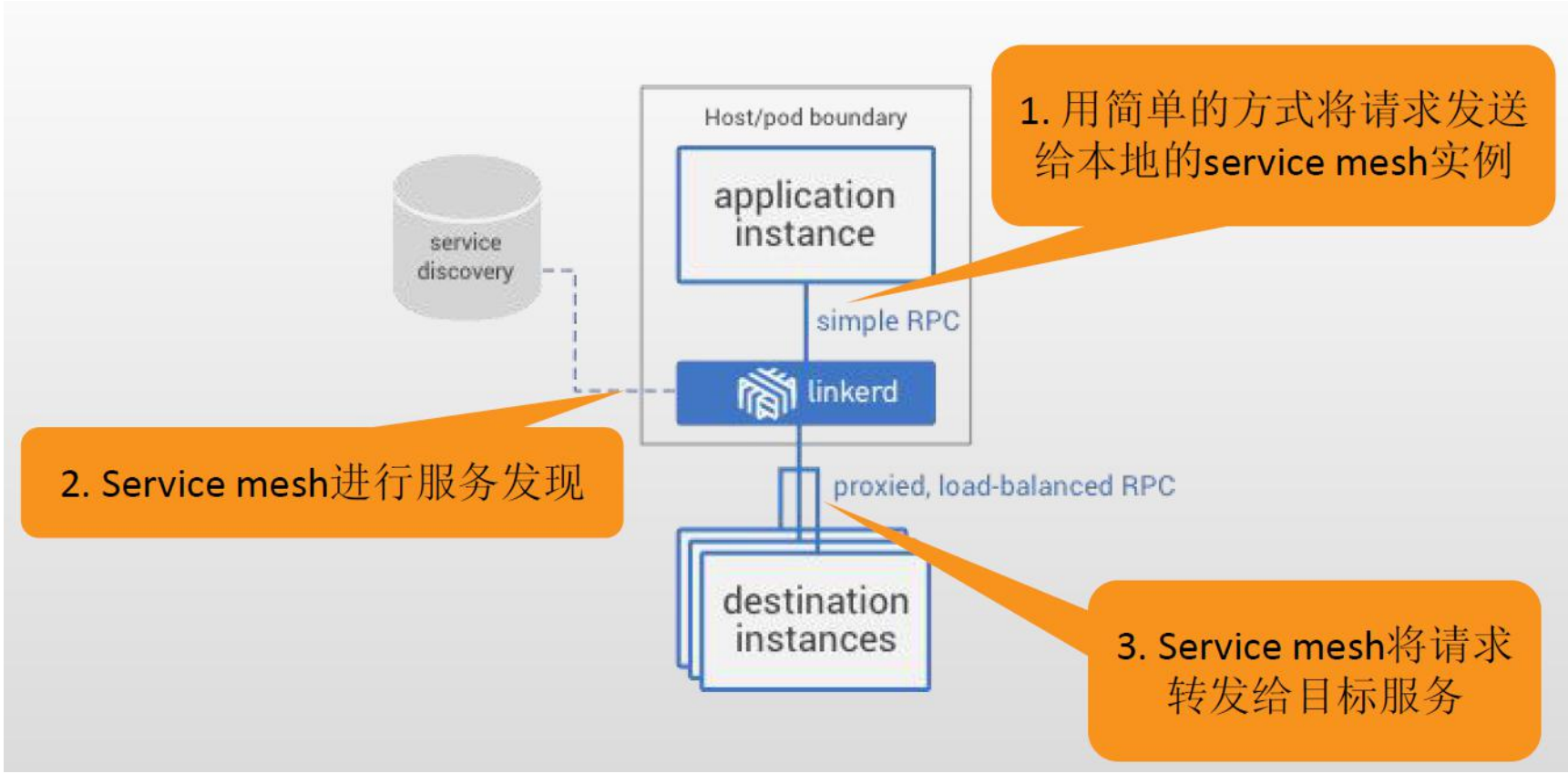

frontend pod
service
label: app=backend
Label Selector
pod
label: app=backend
pod
label: app=backend
pod
label: app=backend
replicationController
label: app=backend
replica: 3

下一代微服务： Service Mesh

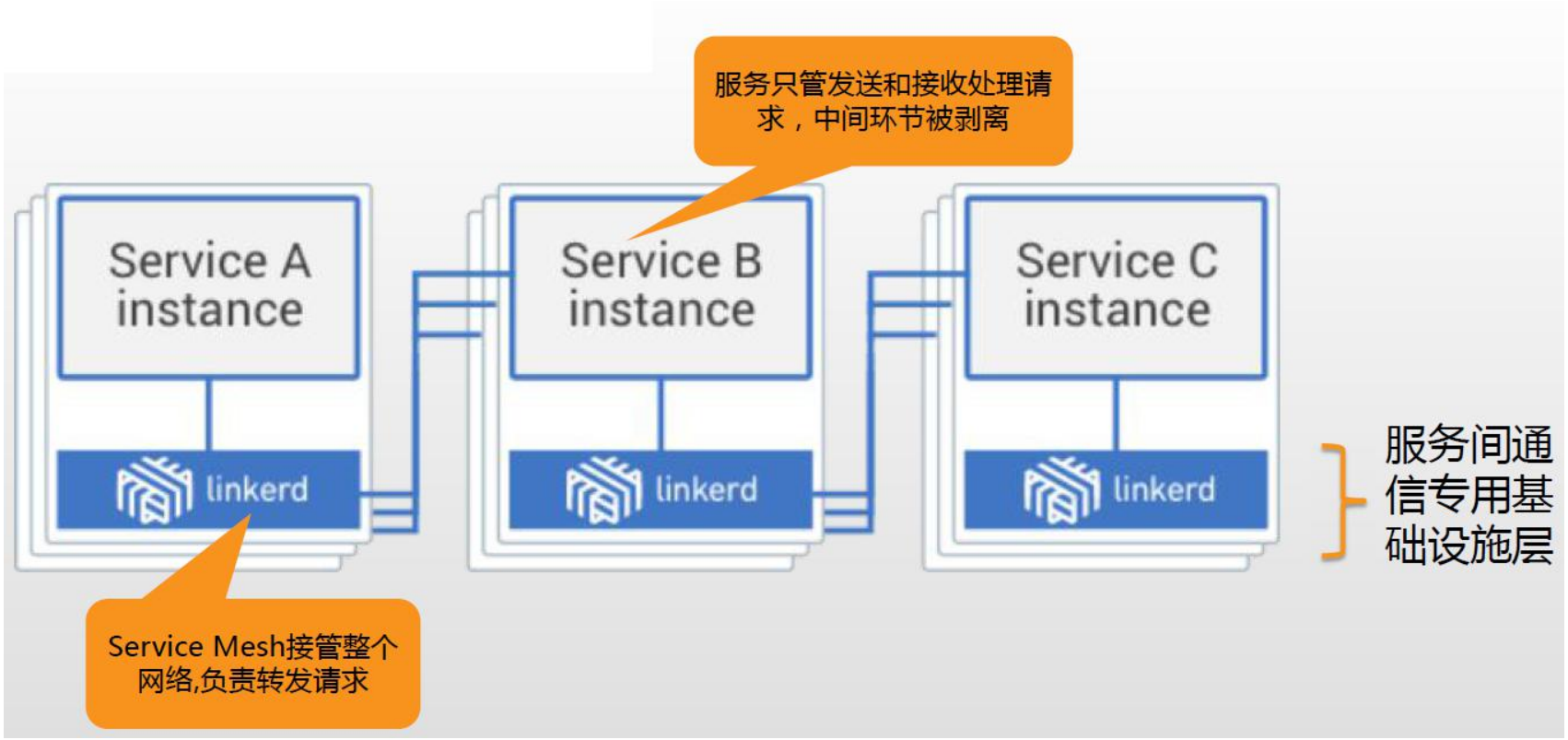
- 一种基础设施层，服务间的通信通过Service Mesh进行。
- 可靠地传输复杂网络拓扑中服务的请求，将服务变成现代的云原生服务。
- 一种网络代理的实现，通常与业务服务部署在一起，业务服务不感知。
- 一种TCP/IP之上的网络模型。



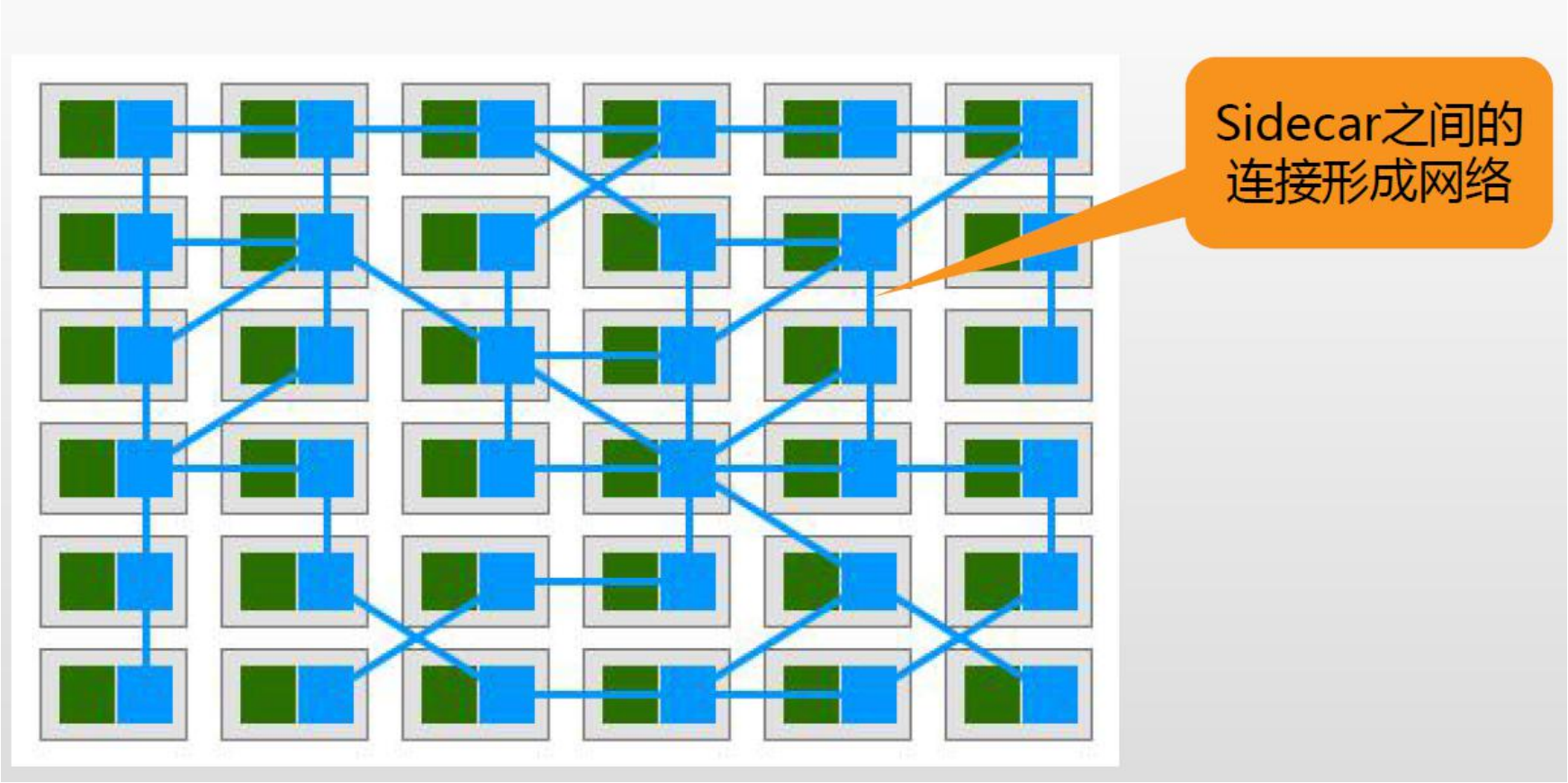
部署模型：单个服务调用，表现为sidecar



部署模型：多个服务调用，表现为通讯层



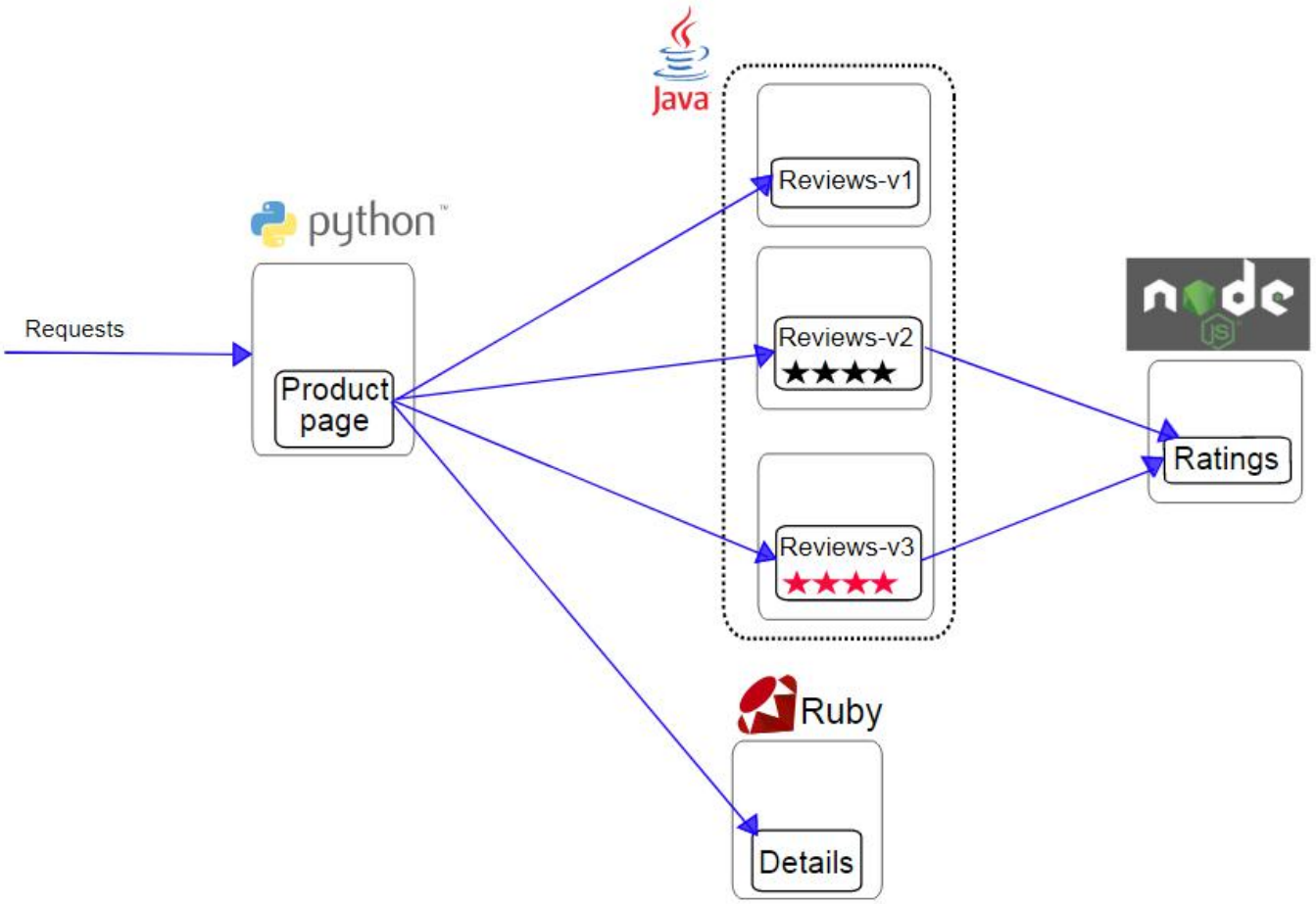
部署模型：有大量服务，表现为网格



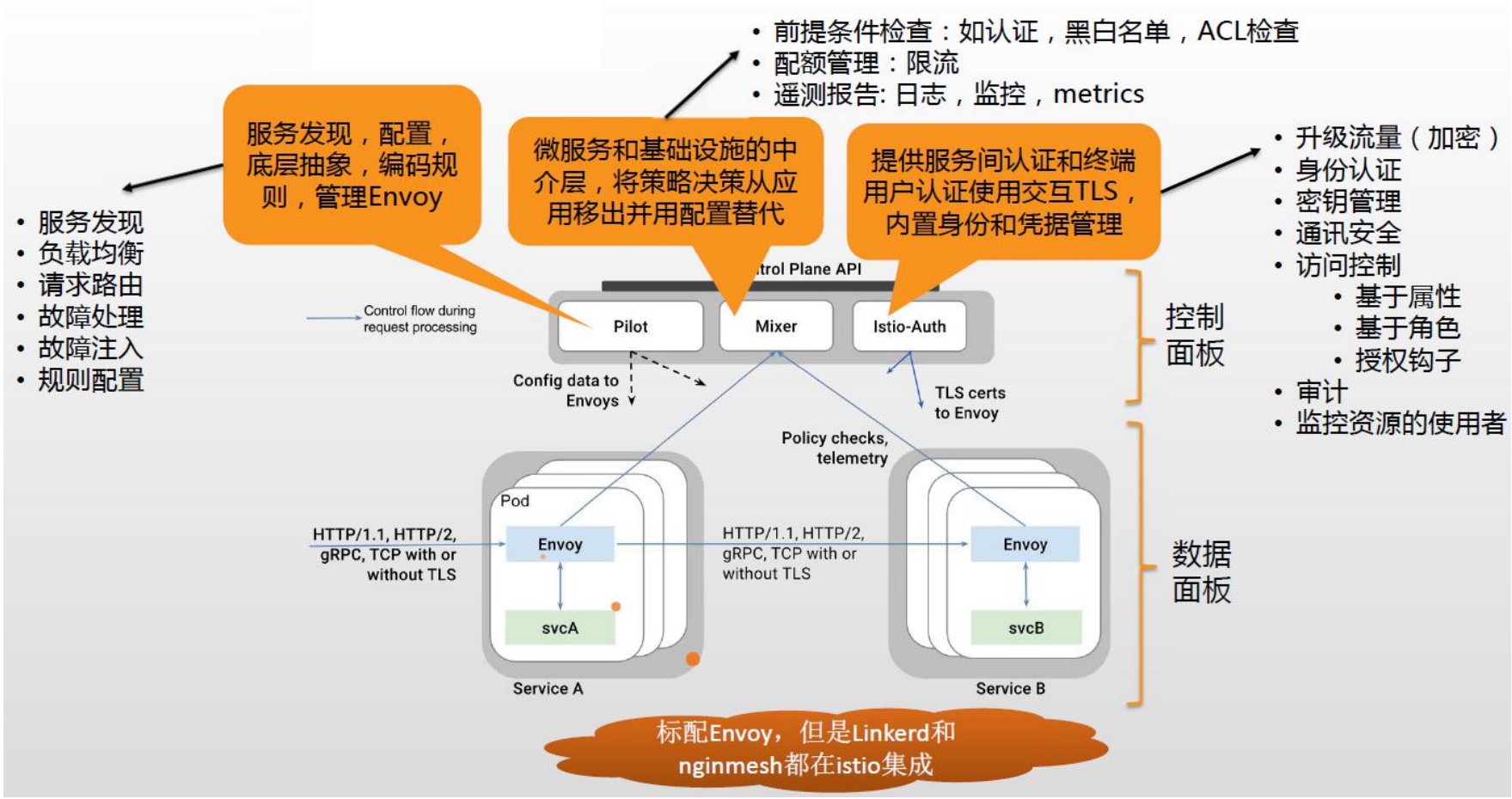
为什么使用Service Mesh

- 无需多种语言的微服务框架开发
- 对业务代码0侵入
- 不适合改造的单体应用
- 开发出开的应用既是云原生的又具有独立性

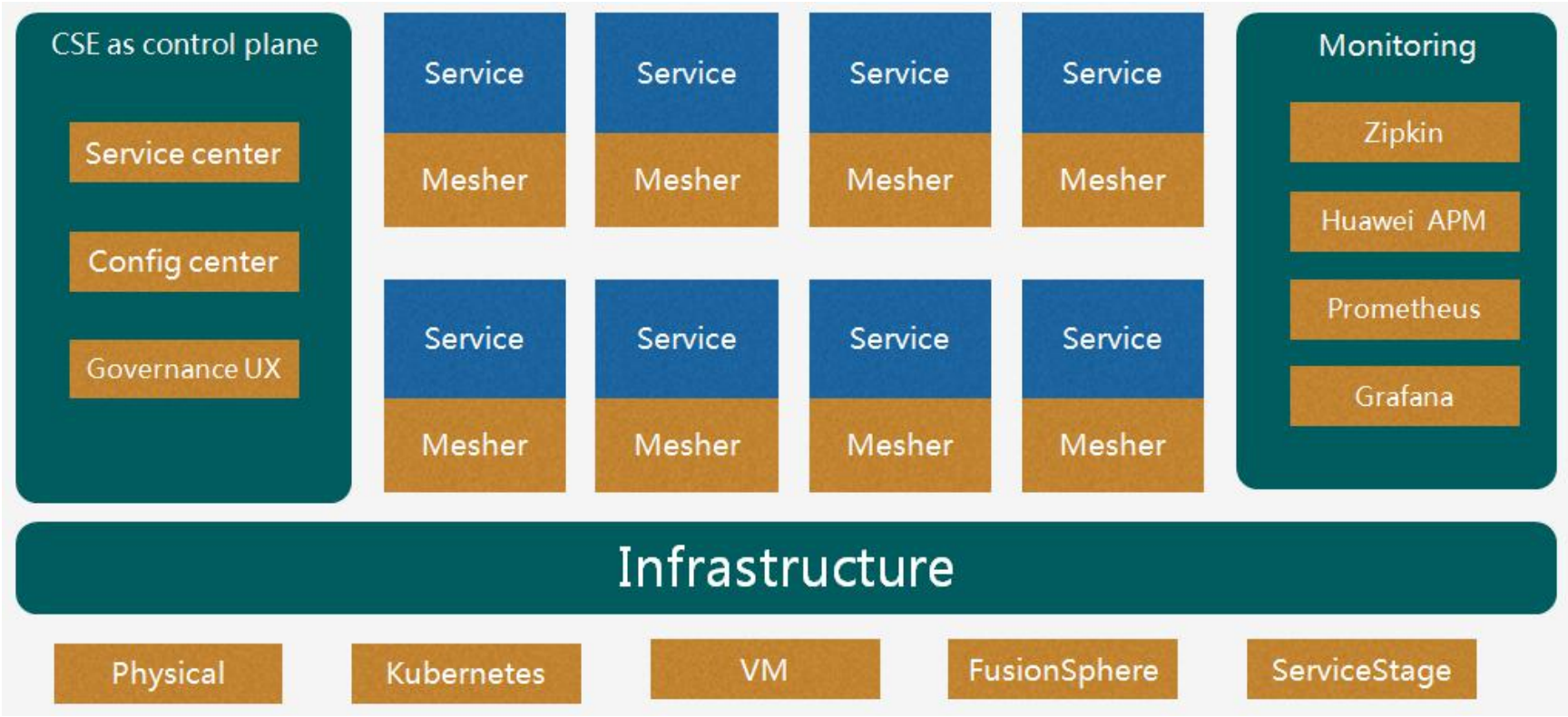
一个例子


Java
python
Requests
Product page
Reviews-v1
Reviews-v2
Reviews-v3
Ratings
Ruby
Details

Google的istio架构



华为的Mesher架构



谢谢！