

# 云原生训练营

## 模块十五： 微服务项目的开发和部署案例

孟凡杰

前 eBay 资深架构师



# 目录

1. 应用管理的实践分享
2. 基于 Bookinfo 的服务治理
3. 全课总结

# 1. 应用管理的实践分享

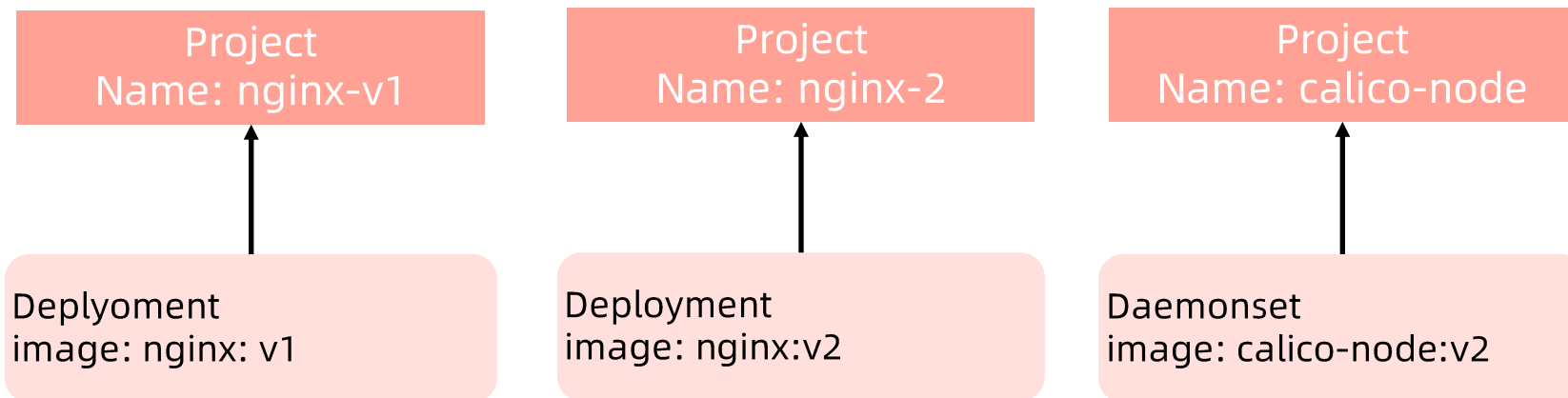
# 多租户系统

基于 Kubernetes 构建多租户平台

- 基于 Tenant 的租户治理
- 基于 Application 的应用治理

# Project

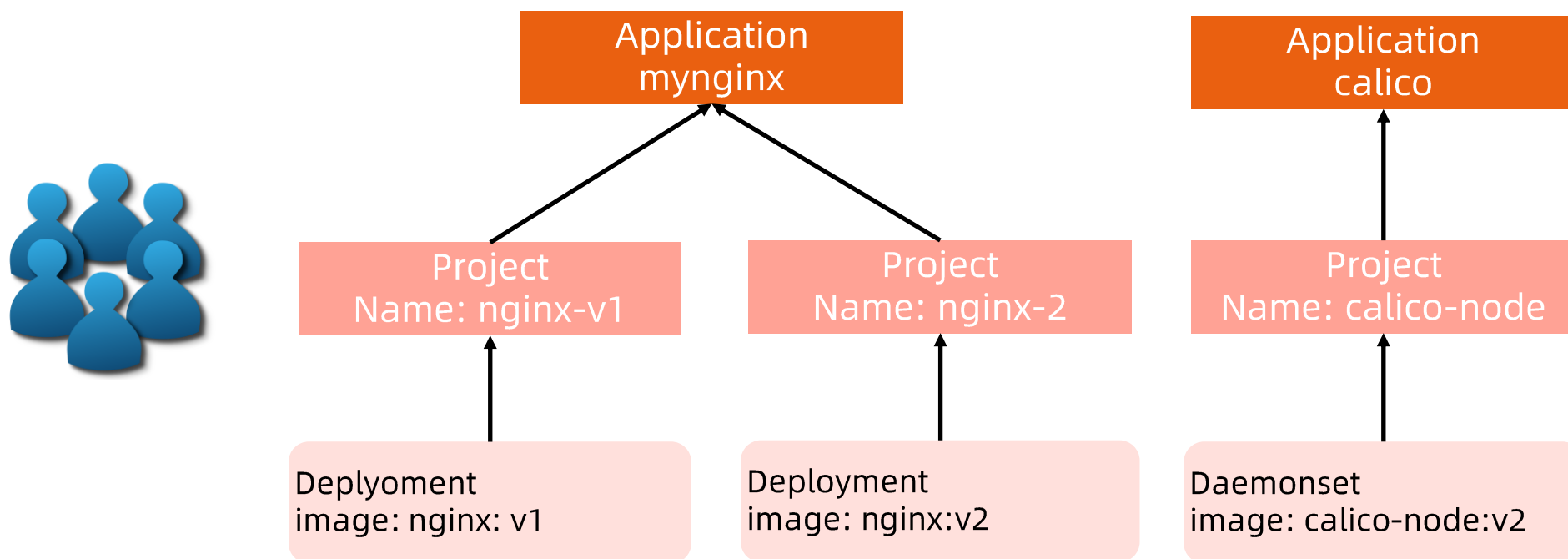
Project 描述应用实例，每一组有特定业务目的独立的部署（deployment, statefulset, daemonset）可定义为一个 Project。



# Application

Application 是对应用的抽象

Project 是应用的部署实例



# Application

Application 是一个或多个具有相同业务目的的应用实例的抽象。包含应用配置，应用代码定义，责任人，联系方式等等。

Application 被 Tenant 管理。  
Application 定义如下属性：

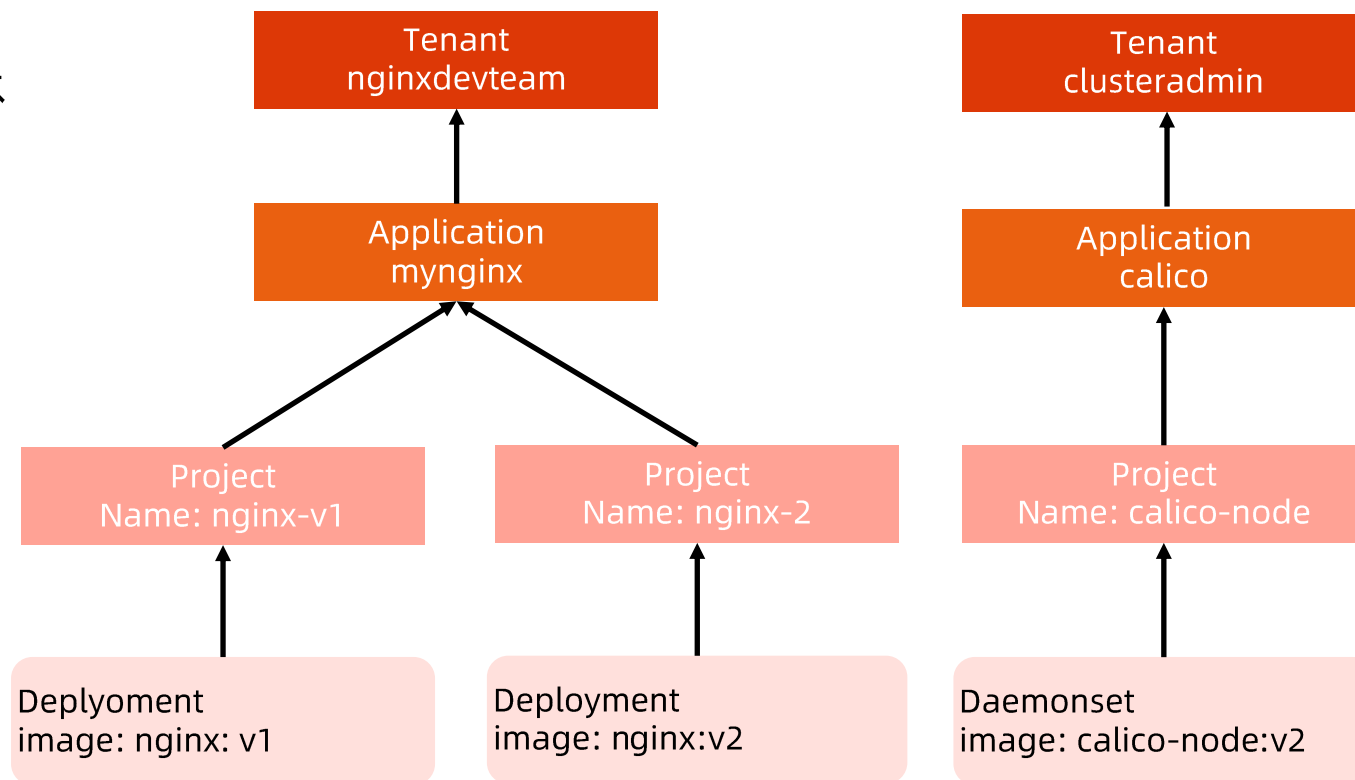
- Application type: web, bigdata 等
- Data protection level: 数据安全需求，是否要加密
- Owner: 责任人
- Admin Tenant: 管理账号
- GitHub: 源代码地址
- ...

# Tenant

Tenant 是管理应用的账号

Tenant 也是集群费用分摊实体

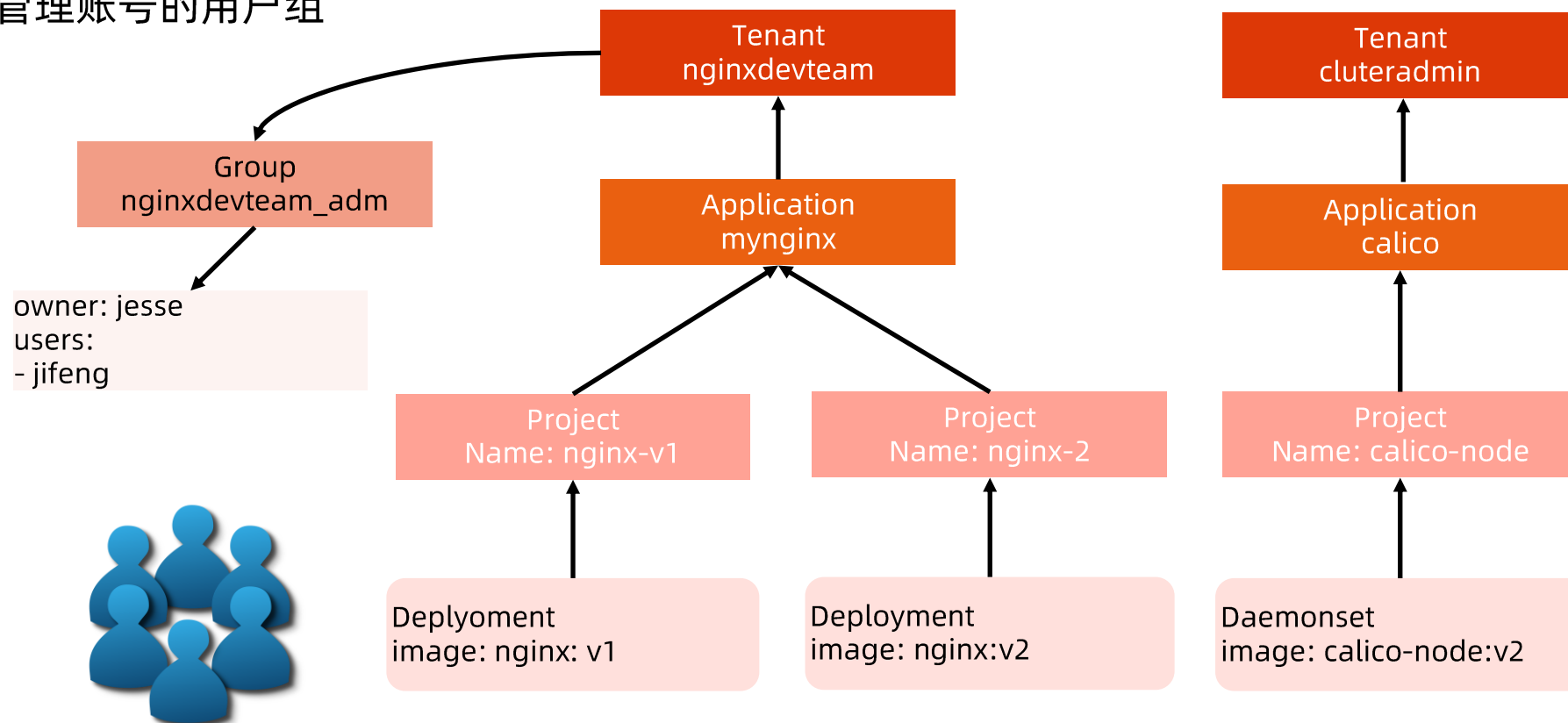
Tenant 是多租户集群中的租户



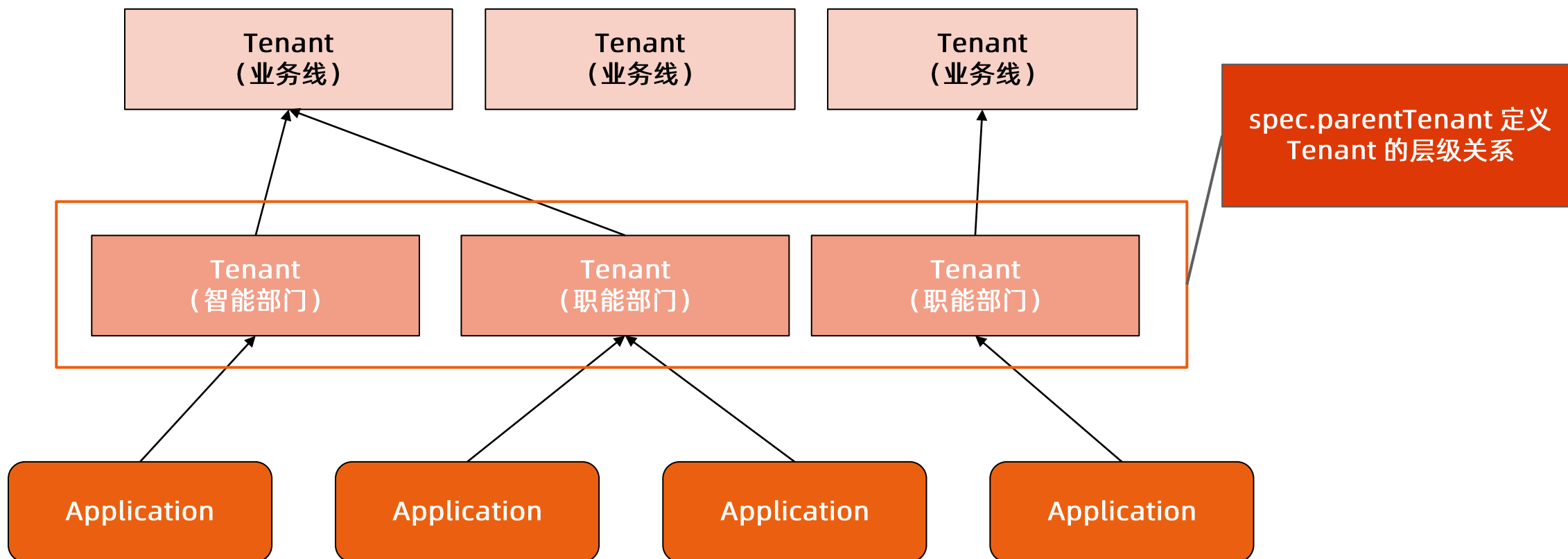


# Group

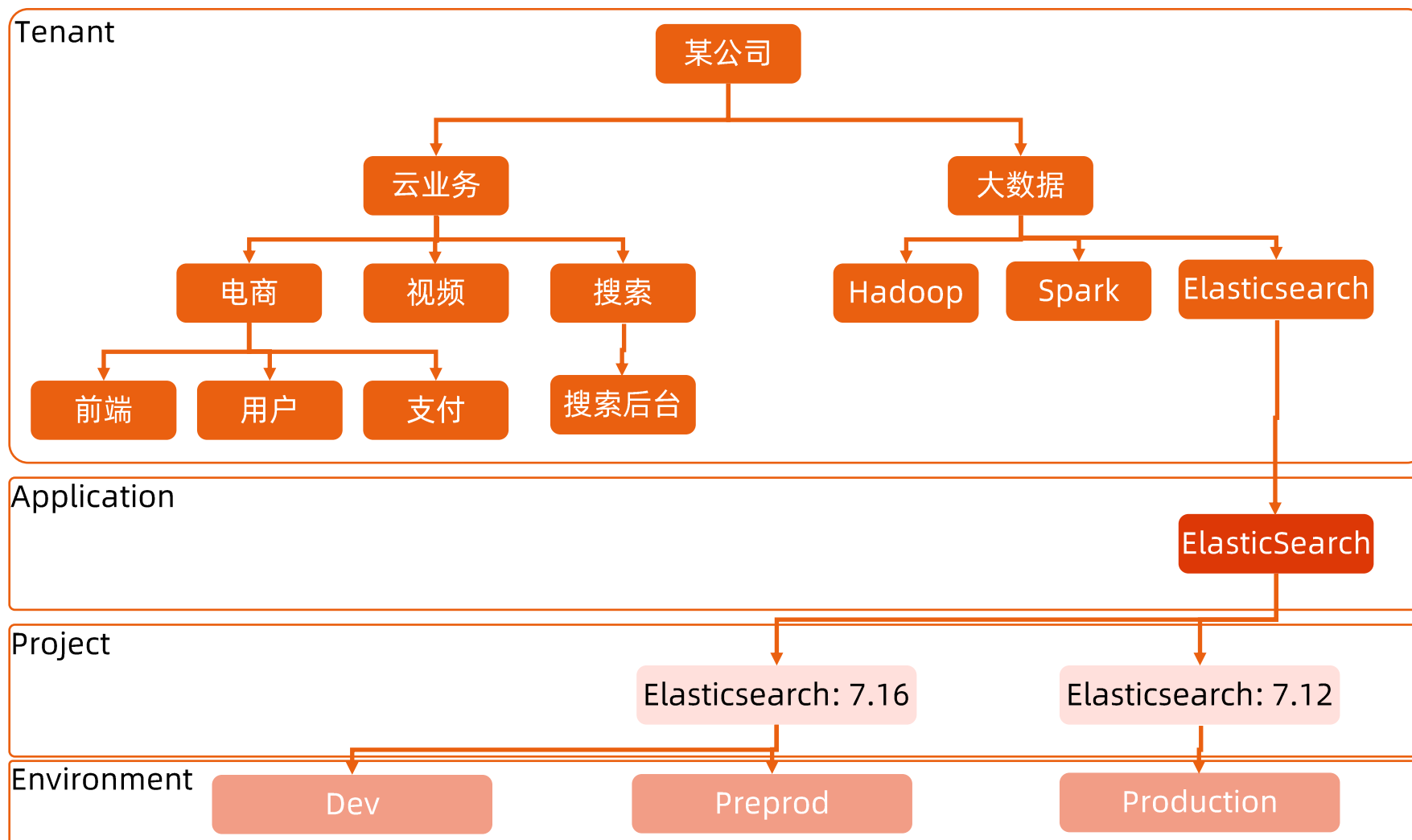
Group 是管理账号的用户组



# Tenant 树状分级




# 基于 Tenant 管理的企业组织架构



# 新模型与 Kubernetes 原生对象的关联

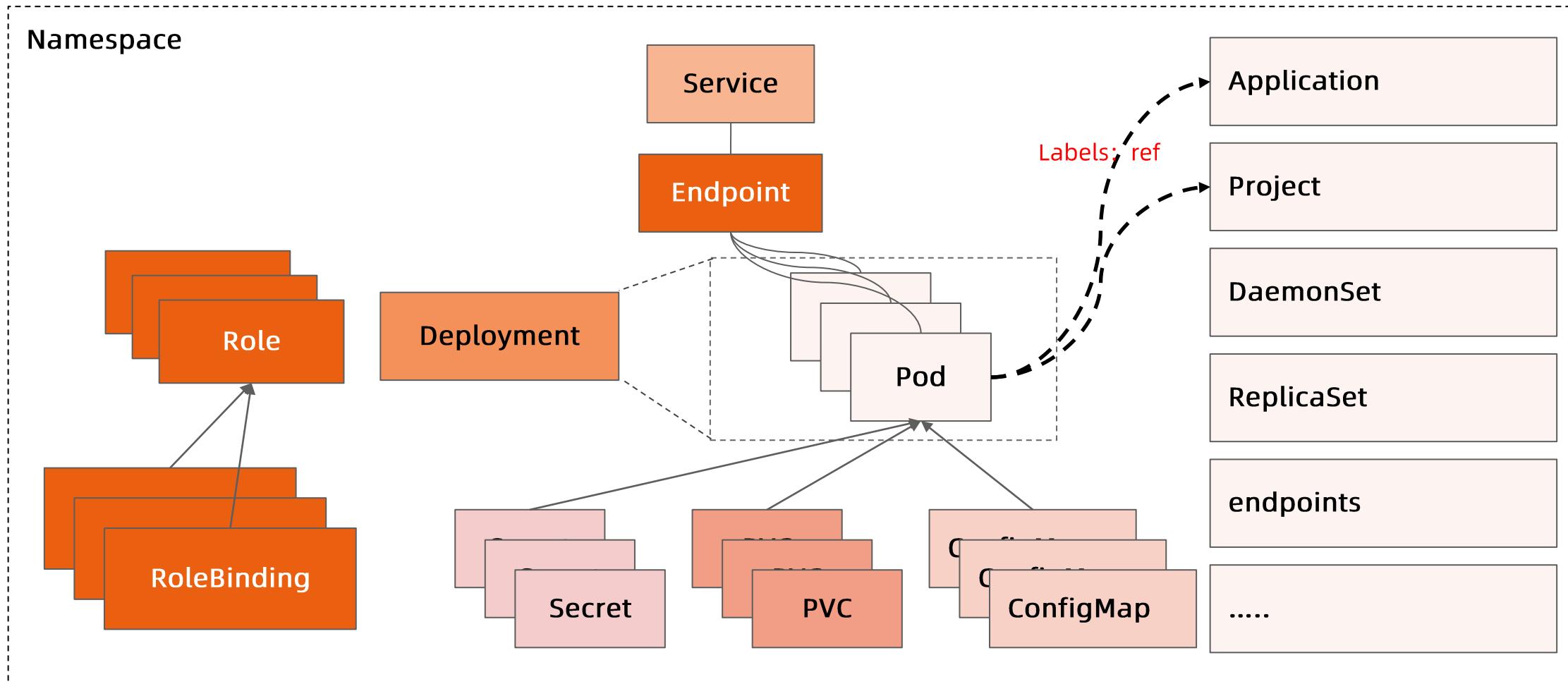
将 Namespace 定义为应用运行的隔离环境

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    application.k8s.io/name: mynginx
    tenant.k8s.io/name: jesse
name: mynginx
```

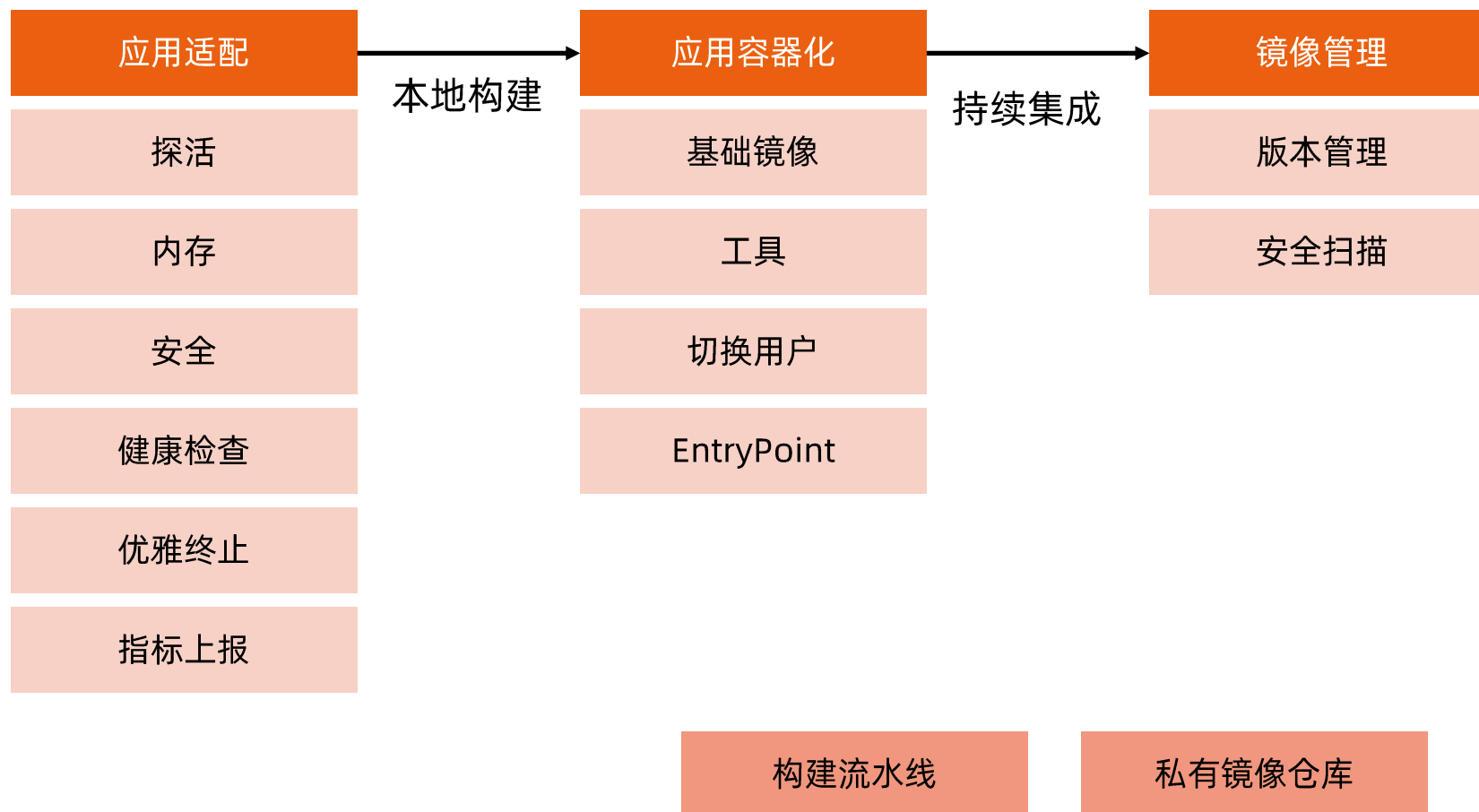


该 Namespace 的默认 application

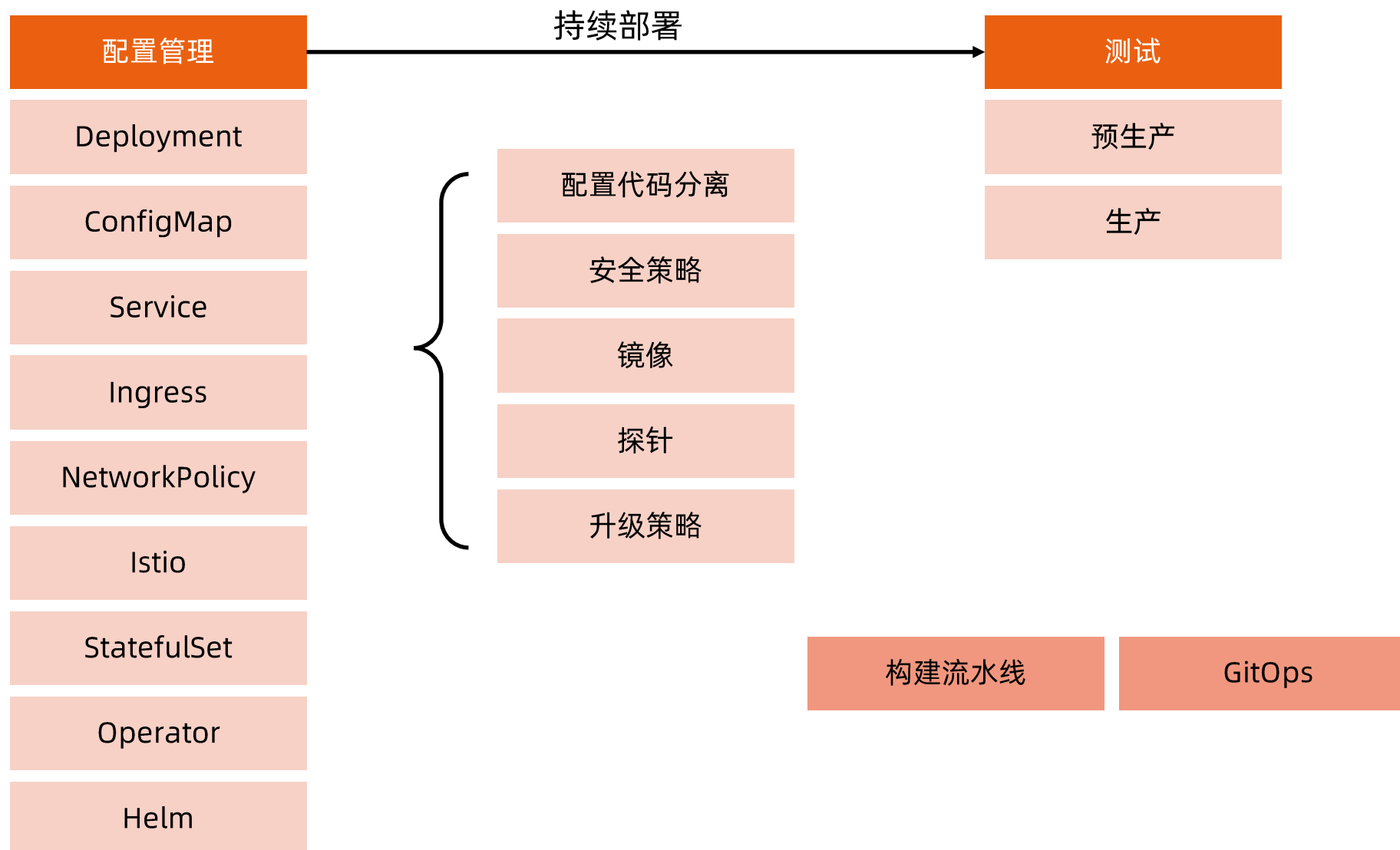
# 新模型与 Kubernetes 原生对象的关联



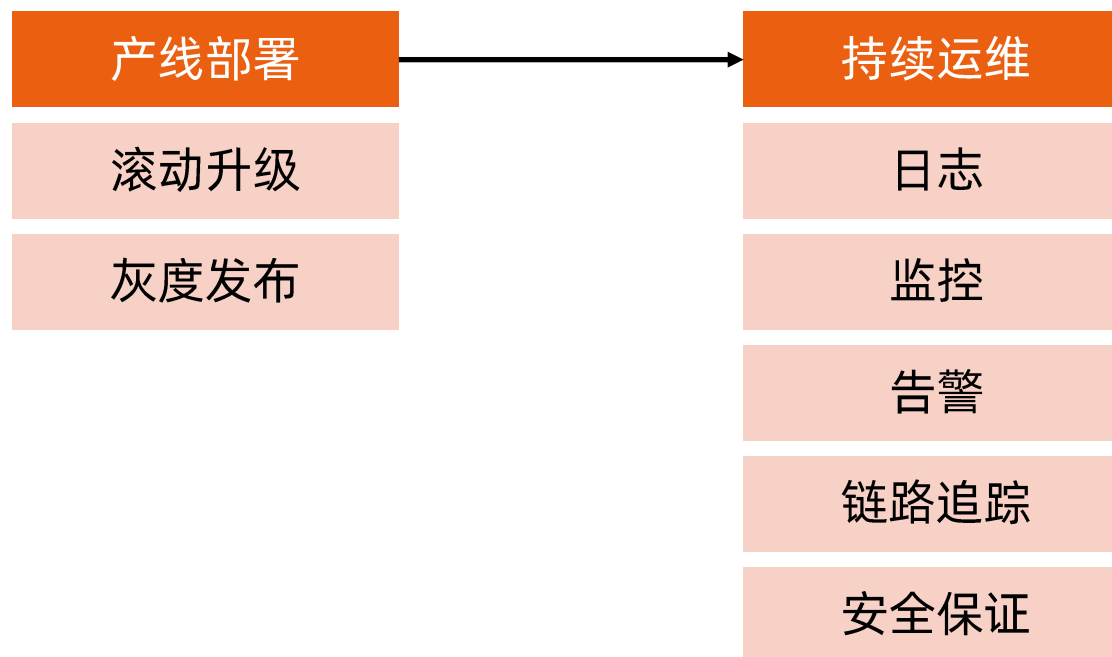
# Kubernetes 应用实战回顾



# Kubernetes 应用实战回顾



# Kubernetes 应用实战回顾





## 2. 基于 Bookinfo 的服务治理

# Bookinfo 简介

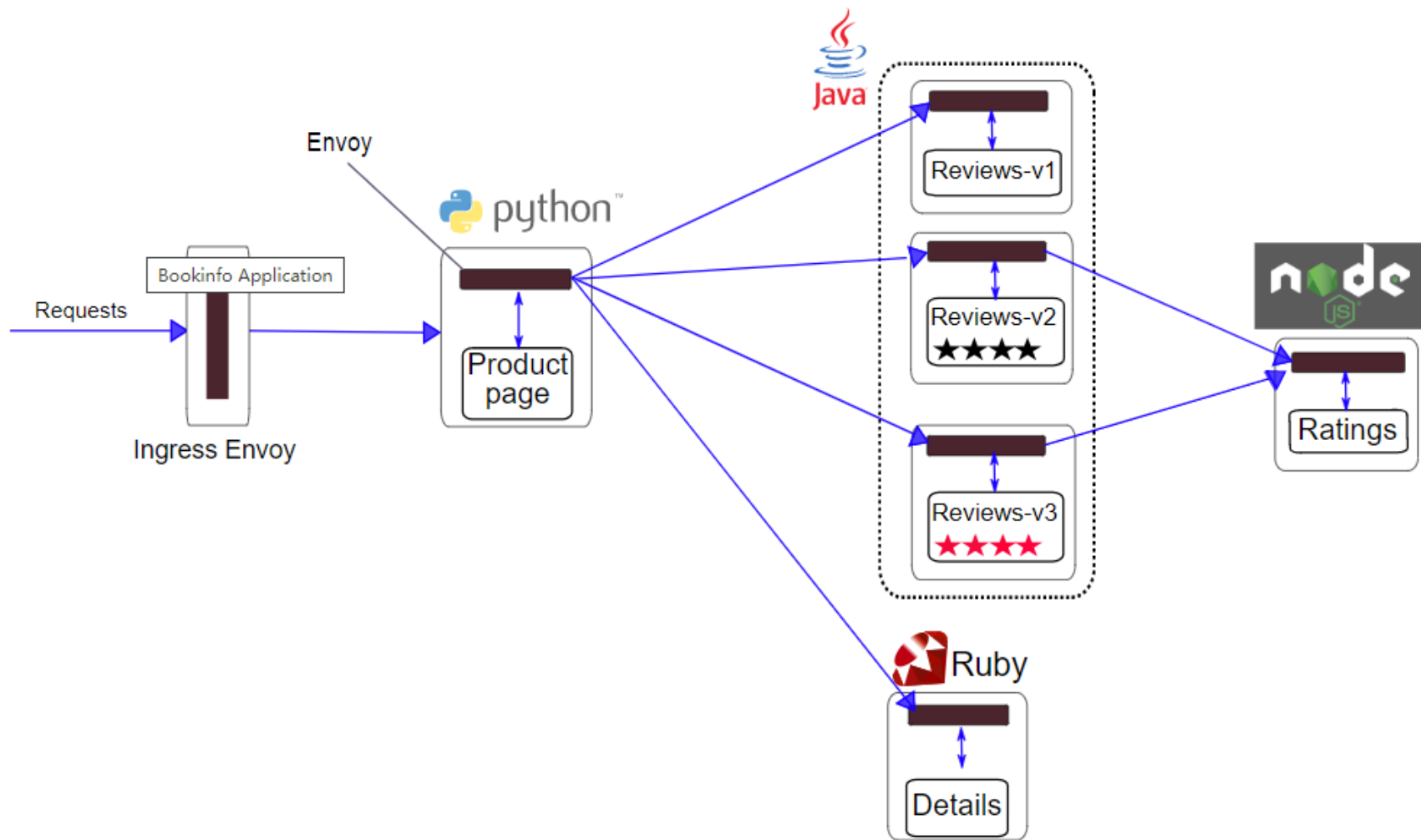
Bookinfo 应用分为四个单独的微服务：

- productpage 会调用 details 和 reviews 两个微服务，用来生成页面。
- details 中包含了书籍的信息。
- reviews 中包含了书籍相关的评论。它还会调用 ratings 微服务。
- ratings 中包含了由书籍评价组成的评级信息。

reviews 微服务有 3 个版本，可用来展示各服务之间的不同的调用链路：

- v1 版本不会调用 ratings 服务。
- v2 版本会调用 ratings 服务，并使用 1 到 5 个黑色星形图标来显示评分信息。
- v3 版本会调用 ratings 服务，并使用 1 到 5 个红色星形图标来显示评分信息。

# Bookinfo 应用架构



# 安装

## 下载

```
curl -L https://istio.io/downloadIstio | sh -
```

## 安装应用

```
cd /root/istio-1.12.0  
kubectl label namespace default istio-injection=enabled  
kubectl apply -f <(istioctl kube-inject -f  
samples/bookinfo/platform/kube/bookinfo.yaml)
```

# 第一步：将应用发布至 istio ingress 网关

- 创建 Istio 配置对象

```
kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
```

- 查看 Istio 网关入口

```
kubectl get svc -n istio-system
```

```
istio-ingressgateway LoadBalancer 10.109.127.136 <pending>  
15021:30784/TCP,80:31783/TCP,443:31106/TCP,31400:31463/TCP,15443:31663/TCP  
13d
```

- 通过 istio ingress 网关访问 Bookinfo

```
http://192.168.34.2:31783/productpage
```

## 第二步：安全保障

- 将 http 网关转换为 https 网关

```
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -subj '/O=cncamp Inc./CN=192.168.34.2' -keyout bookinfo.key -out bookinfo.crt
```

```
kubectl create -n istio-system secret tls bookinfo-credential --key=bookinfo.key --cert=bookinfo.crt
```

```
kubectl apply -f https-gateway.yaml
```

- 通过 https 端口访问 Bookinfo

```
https://192.168.34.2:31106/productpage
```

## 第三步：开启全链路 mTLS

- 在开启全链路 mTLS 之前，直接访问 productpage 返回成功；
- 创建全局默认 PeerAuthentication；

```
kubectl apply -f mtls.yaml -n istio-system
```

- 在开启全链路 mTLS 之后，直接访问 productpage 返回不成功。

## 第四步：开启服务认证授权

- 创建 RequestAuthentication, 解密 JWT token;

```
kubectl apply -f requestauthentication.yaml
```

- 创建 AuthorizationPolicy, 开启 details 服务基于 JWT 授权;

```
kubectl apply -f authorizationpolicy.yaml
```

- 访问 productpage 页面, 会看到 details 已经无法显示;

Sorry, product details are currently unavailable for this book.

- 改造 productpage 应用;

```
cat productpage-v2/productpage.py
```

- 部署 productpage v2。

```
kubectl apply -f productpage-v2/productpage.yaml
```

```
scale down productpage v1
```



## 第五步：开启 http 到 https 的跳转

- 更新 gateway，增加 80 端口并设置 TLS 转发规则；

```
kubectl apply -f https-gateway.yaml
```

- 访问 gateway（基于浏览器的跳转不工作，因为 http redirect 跳转至默认端口）。

```
curl 10.109.127.136/productpage -v -L -k
```

## 第六步：开启 Kiali

- 安装 Prometheus;

```
kubectl apply -f prometheus.yaml
```

- 安装 Kiali;

```
kubectl apply -f samples/addons/kiali.yaml
```

- 更新 Kiali service 为 NodePort 类型;

```
k edit svc kiali -n istio-system
```

- 登录 Kiali, 进入 Graph 页面查看调用视图。

```
http://192.168.34.2:31816/
```

# 第七步：灰度发布

灰度，就是存在于黑与白之间的一个平滑过渡的区域。对于互联网产品来说，上线和未上线就是黑与白之分，而实现未上线功能平稳过渡的一种方式就叫做灰度发布。

## 1. 按流量百分比

先到先得的方式，比如：限制 10% 的用户体验的是新版本，90% 的用户体验的是老版本。先访问网站的用户就优先命中新版本，直到流量用完为止。

## 2. 按人群划分

按用户 ID、用户 IP、设备类型划分，比如：可通过平时的埋点上报数据得知用户的 PV、UV、页面平均访问时长等数据，根据用户活跃度来让用户优先体验新版本，进而快速观察使用效果。

按地域、性别、年龄等用户画像划分，比如：可通过用户的性别、年龄等做下新老版本的对比效果来看看目标用户在新版本的使用年龄段，性别范围是多少。

## 第七步：灰度发布

- 为每个服务创建 destinationrule，该步骤将所有服务按版本切分为不同子集；

```
kubectl apply -f samples/bookinfo/networking/destination-rule-all.yaml
```

- 将所有流量转发至 v1；

```
kubectl apply -f samples/bookinfo/networking/virtual-service-all-v1.yaml
```

- 访问 Bookinfo，可看到 ratings 从三个版本轮番切换变成了不显示，因为 reviews v1 版本不调用 ratings；

- 定义灰度流量规则，将 Jason 用户切换至 reviews v2；

```
kubectl apply -f samples/bookinfo/networking/virtual-service-reviews-test-v2.yaml
```

- 访问 Bookinfo，可看到 ratings 变为固定版本。

## 第八步：故障注入

- 故障注入通常用于混乱测试场景，确保单个服务出现故障不会引起联动效应，导致局部错误引起全局系统性故障；

为 ratings 服务 v2 添加固定延迟时间：

```
kubectl apply -f samples/bookinfo/networking/virtual-service-ratings-test-delay.yaml
```

- 更新后，延迟超出了客户端的等待时间，当登录用户为 Jason 时，productpage 页面出错，其他用户无错误；

Sorry, product reviews are currently unavailable for this book.

- 为 ratings 服务添加错误返回值；

```
kubectl apply -f samples/bookinfo/networking/virtual-service-ratings-test-abort.yaml
```

- 更新后 ratings 服务不可用。

## 第九步：最佳实践

- 记得为客户端添加超时时间

```
http:  
- route:  
  - destination:  
    host: reviews  
    subset: v2  
timeout: 0.5s
```

- 记得加断路器规则

```
trafficPolicy:  
  connectionPool:  
    tcp:  
      maxConnections: 1  
    http:  
      http1MaxPendingRequests: 1  
      maxRequestsPerConnection: 1  
  outlierDetection:  
    consecutive5xxErrors: 1  
    interval: 1s  
    baseEjectionTime: 3m  
    maxEjectionPercent: 100
```

### 3. 全课总结

# 回顾一下内容

## Go语言

- 核心知识点和原理

## Kubernetes

- 架构基础
- 核心组件
- 生产集群管理
- 集群运维
- 应用上云

## 容器技术

- Namespace
- Cgroup
- OverlayFS
- Network
- 最佳实践

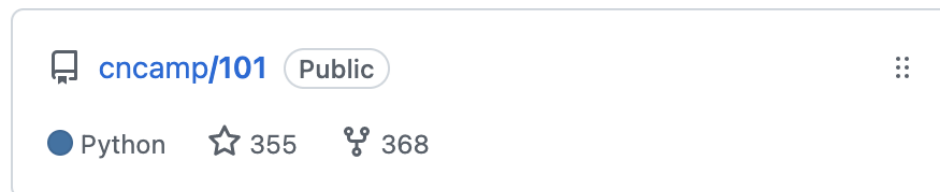
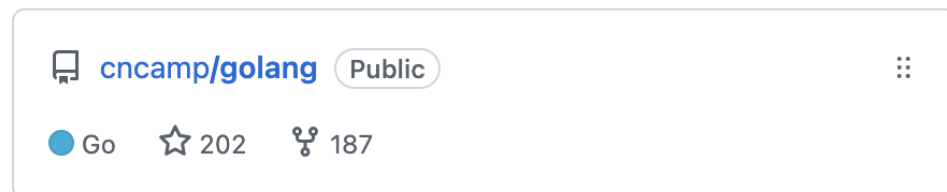
## 服务网格

- 流量管理
- 安全



# 这门课于我是一个史诗级的任务

- 这门课我从2022年 6 月开始规划;
- 期间我经历了 eBay 的一个月长假, 新旧工作过度的过度, 一直到 12 月底, 所有非工作时间;
- cncamp github 11 个 repo 跟本课程有关, 101 和 golang 是大家关注度最多的两个:
  - golang: 201 颗星, 187 个 fork, 32 个 commit;
  - 101: 352 颗星 368 个 fork, 206 个 commit。
- PPT 1300+ 页;
- 直播 80+ 小时, 录播120+小时



# 感谢大家的积极参与

年轻人有动力，未来有希望

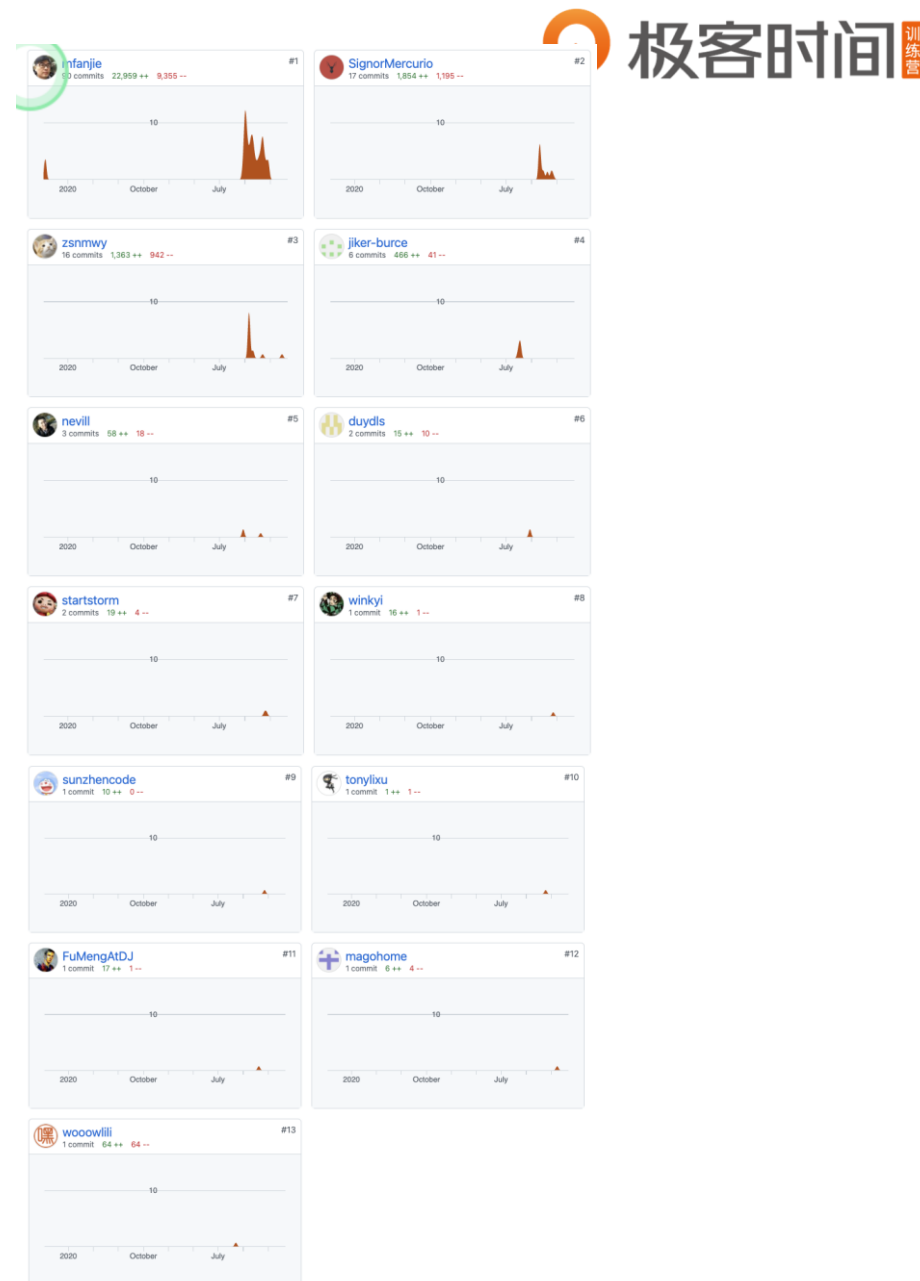
我们每天喊着不要卷了，但从没停下过脚步

微信群里凌晨还在讨论技术问题

感谢大家对项目本身的回馈，感谢所有的contributor

感谢我的兄弟们，助教老师，班主任老师们和平台

这是一段值得记住的历程。



# 课程的结束是新人生开始

没有按照我准备的练习做过一遍的同学，我依然建议结课后继续

真心希望大家都有光明的未来

希望我的分享能帮到大家

我也还在，大家还是很容易找到我

- 微信号：finops,
- Email: [mengfj@gmail.com](mailto:mengfj@gmail.com), 很少看
- 我还在继续新课程
- 我会经常出现在极客时间平台直播
- GitHub会一直活跃
- 来开源共建啊，年轻人<https://github.com/gocrane/crane/>

# THANKS

 极客时间 | 训练营