京东公有云容器服务的实践与探索

京东云高级产品经理

目录页

Contents Page

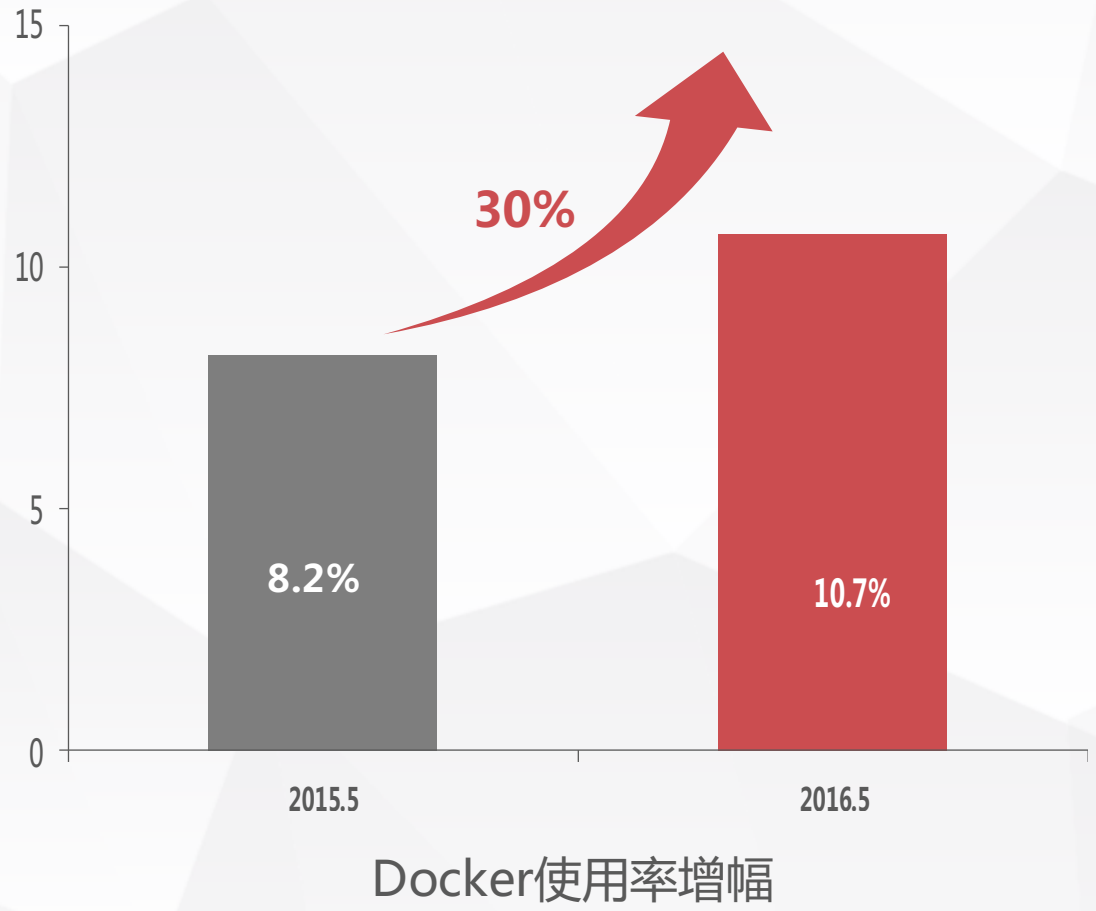
容器趋势分析——Docker关键数据

Docker主机数量
超过
1400万

Docker Hub镜像
pull 次数
120亿

Docker化应用程
序数量
90万

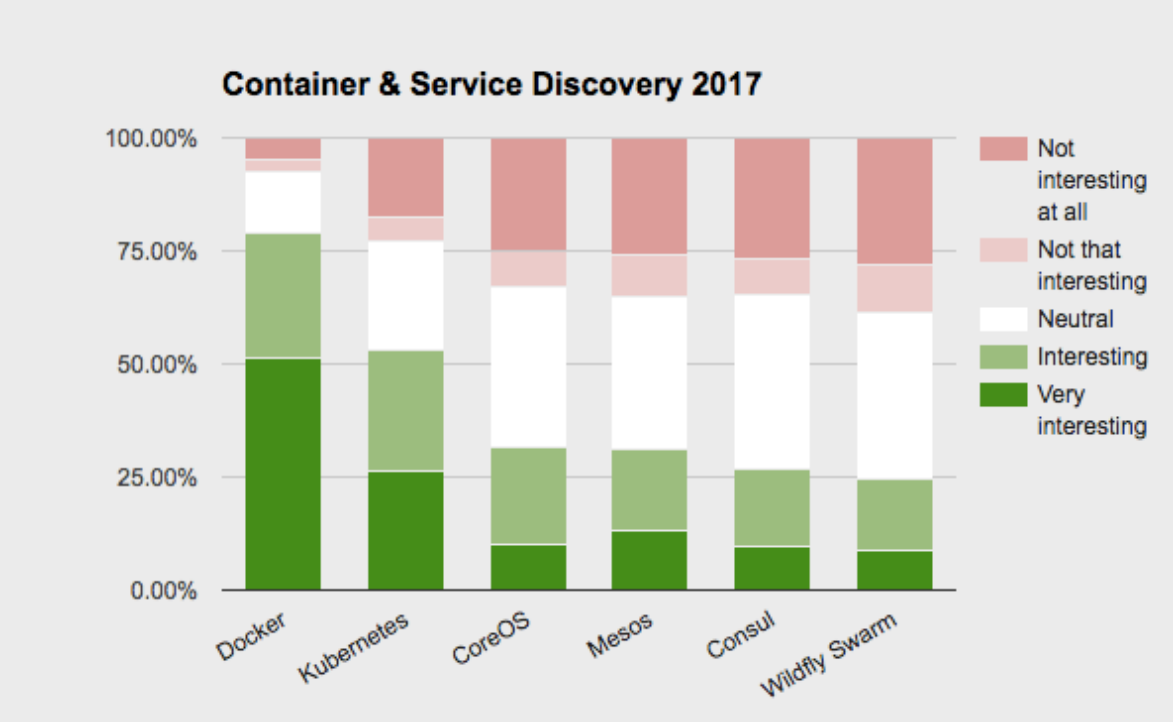
Docker项目贡献
人数超过
3300

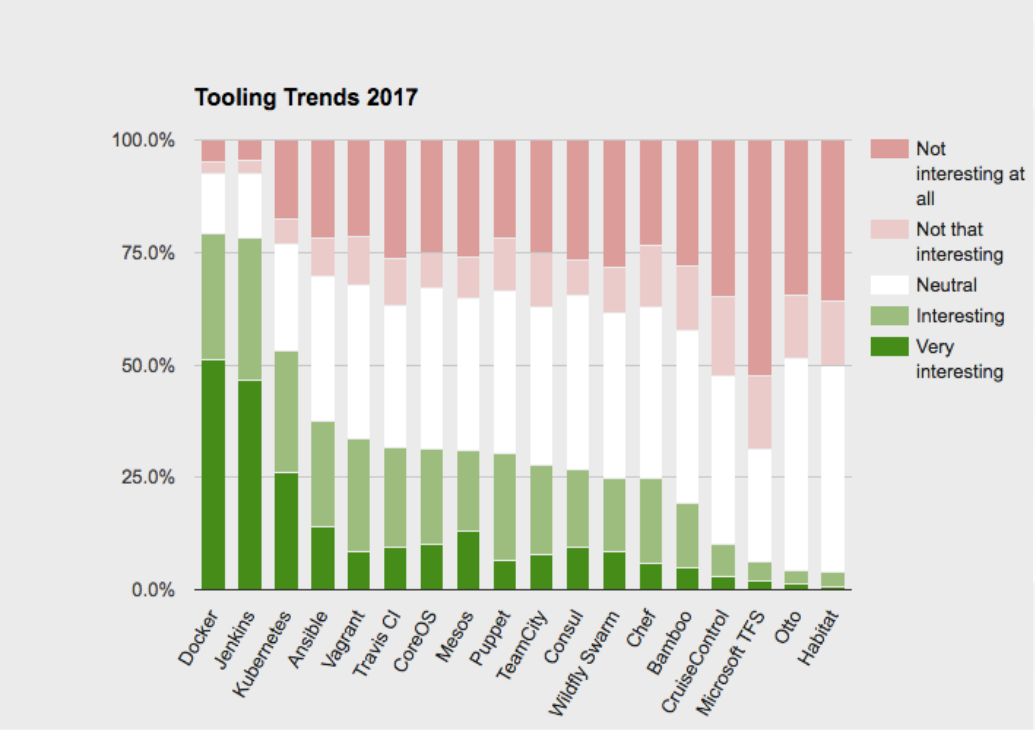


Docker使用率增幅

容器趋势分析

- 随着容器技术的发展，50%以上用户的关注点主要聚焦在docker和Kubernetes上
- 比较受用户关注的工具是docker、Jenkins和Kubernetes



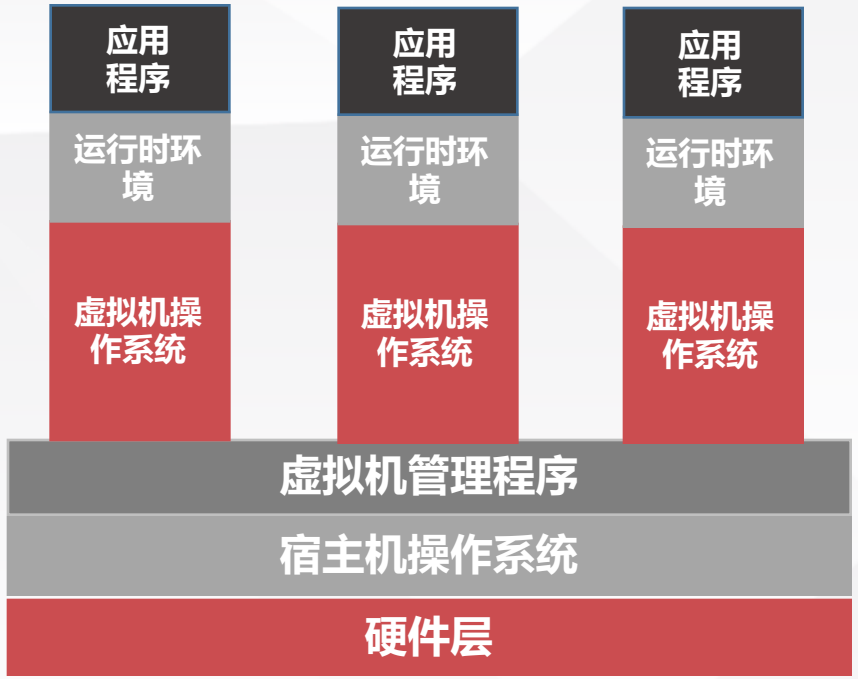


目录页

Contents Page

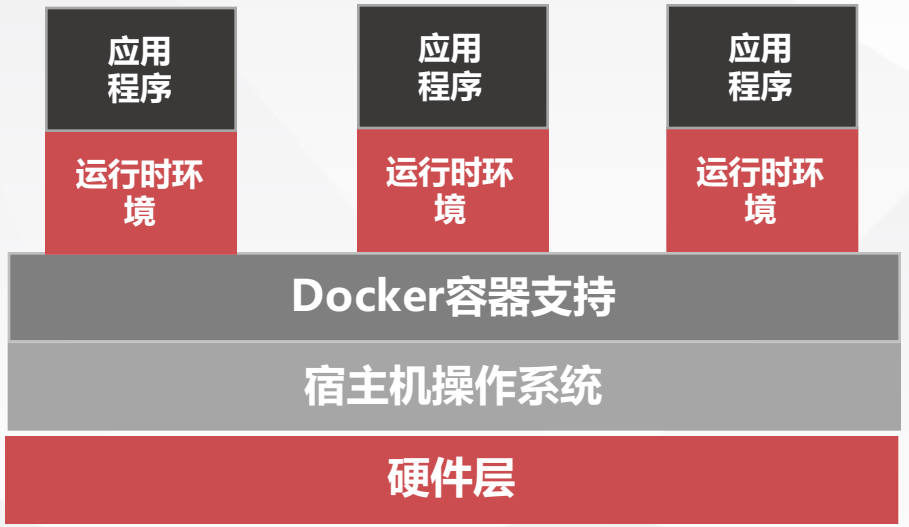
从虚拟机到容器

传统的虚拟化方式



- **安全** 强隔离
- **生态** 是一台“机器”，很多已有工具——网络、存储、在线迁移
- **成熟度** 可以使用自己的Kernel

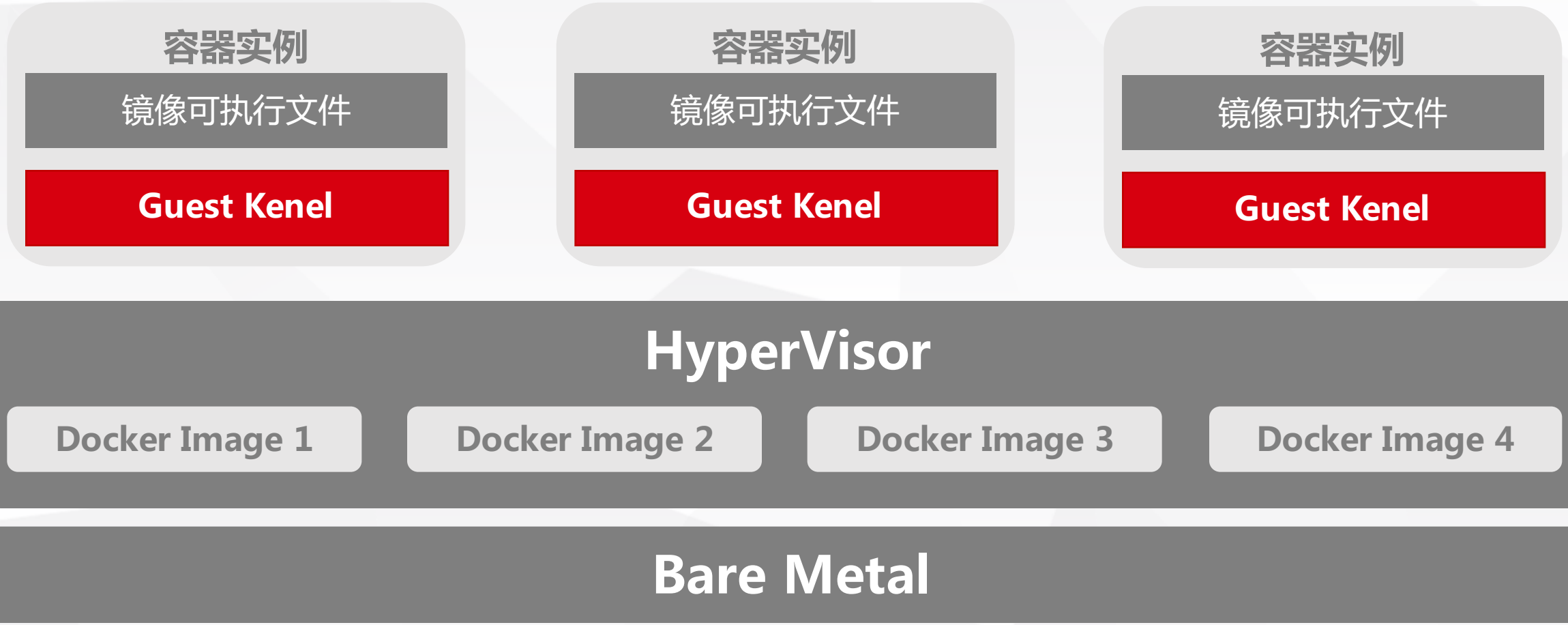
容器的虚拟化方式



- **启动** 启动快，亚秒级启动时间
- **体积** 轻巧，几十到上百MB
- **应用** 发布方便，封装应用用于发布

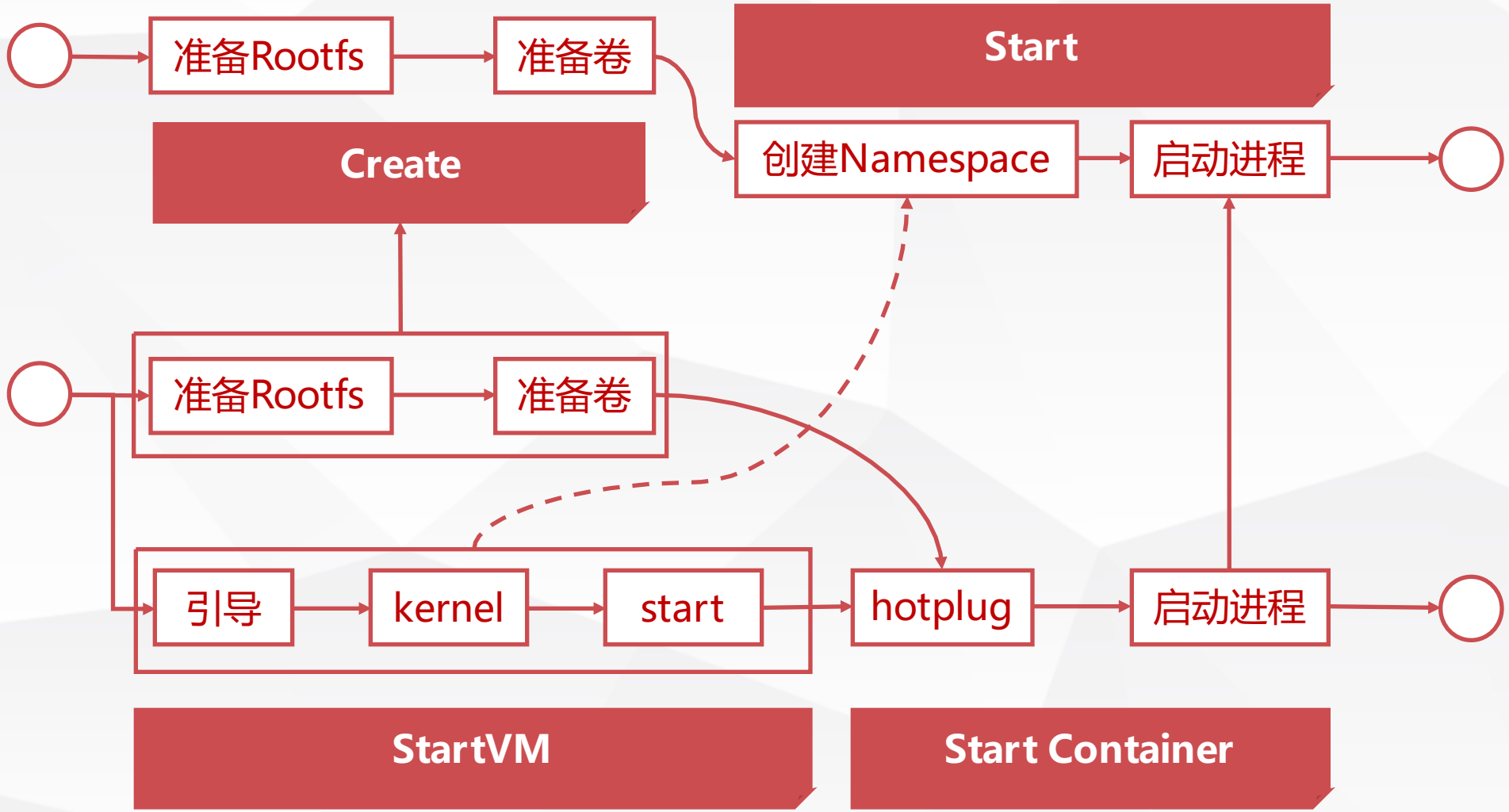
京东云“蜂鸟”容器

- 每个容器使用各自独立的内核



从虚拟机到容器

- 让虚机直接运行Docker镜像



京东“蜂鸟”容器服务

强隔离 **安全**

完整 **生态**

独立的Kernel **成熟度**

启动 快，亚秒级启动

体积 轻巧，仅MB级

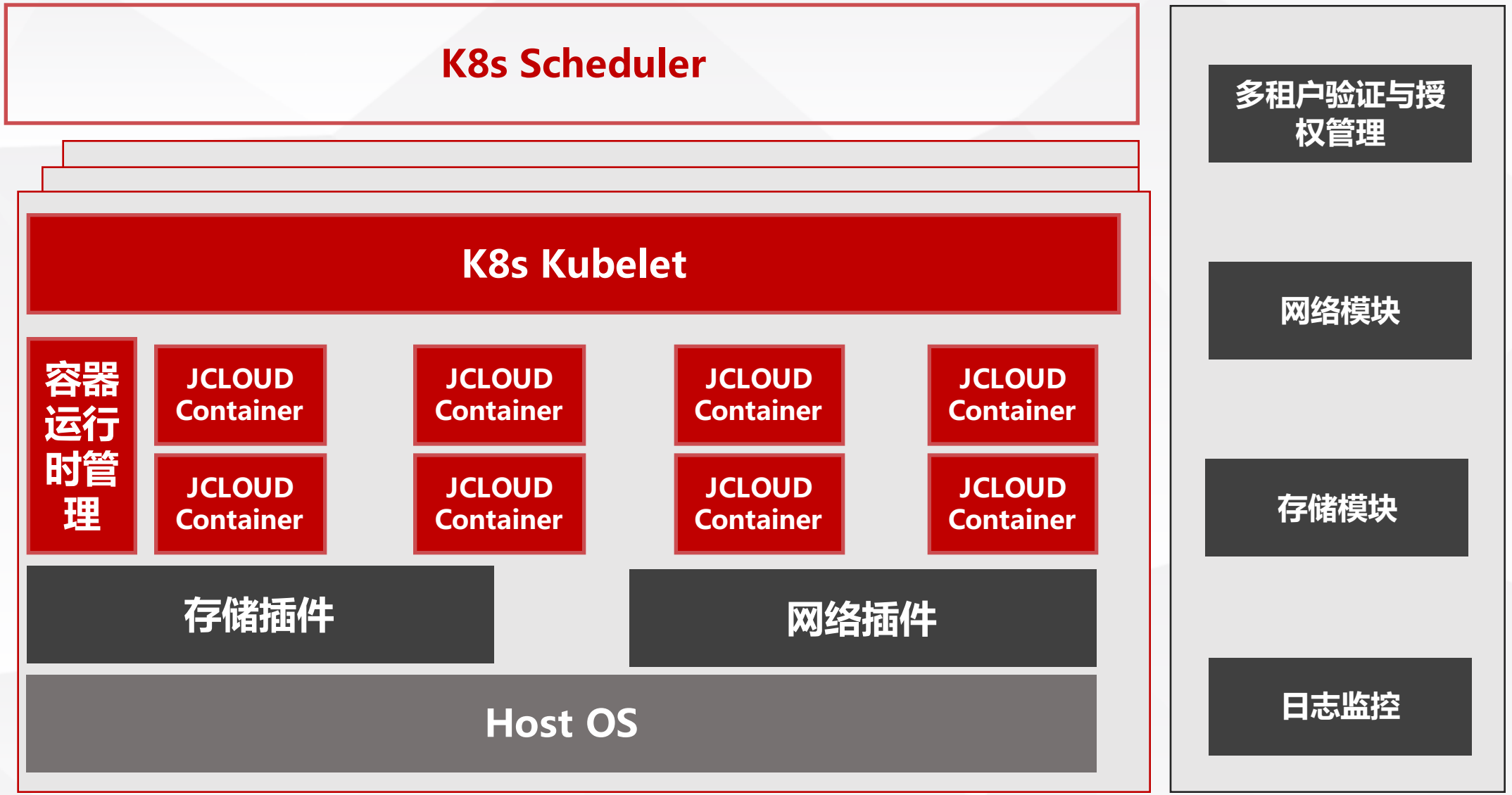
应用 封装应用，便捷发布

京东云容器——充分融合了虚拟化和容器的优点

目录页

Contents Page

“蜂鸟”容器的K8s应用


K8s Scheduler
K8s Kubelet
容器运行时管理
JCLOUD Container
JCLOUD Container
JCLOUD Container
JCLOUD Container
JCLOUD Container
JCLOUD Container
JCLOUD Container
JCLOUD Container
存储插件
网络插件
Host OS
多租户验证与授权管理
网络模块
存储模块
日志监控

“蜂鸟”容器的K8s应用解决方案

身份验证与授权

- Keystone管理不同租户并在管理操作流程中执行身份验证任务
- 兼容SDN网络与存储插件

多租户网络模型

- 每位租户都拥有与其他租户完全隔离的自有网络环境
- 用户创建的任意Network对象，都将由我们所新增的Kubernetes Network控制器（Network Controller）来负责其生命周期管理工作
- 每一个Network被分配至一个或多个Kubernetes Namespaces，而任意归属于同一Network的容器都能够直接通过IP地址彼此通信

“蜂鸟”容器的K8s应用解决方案

多租户Service代理

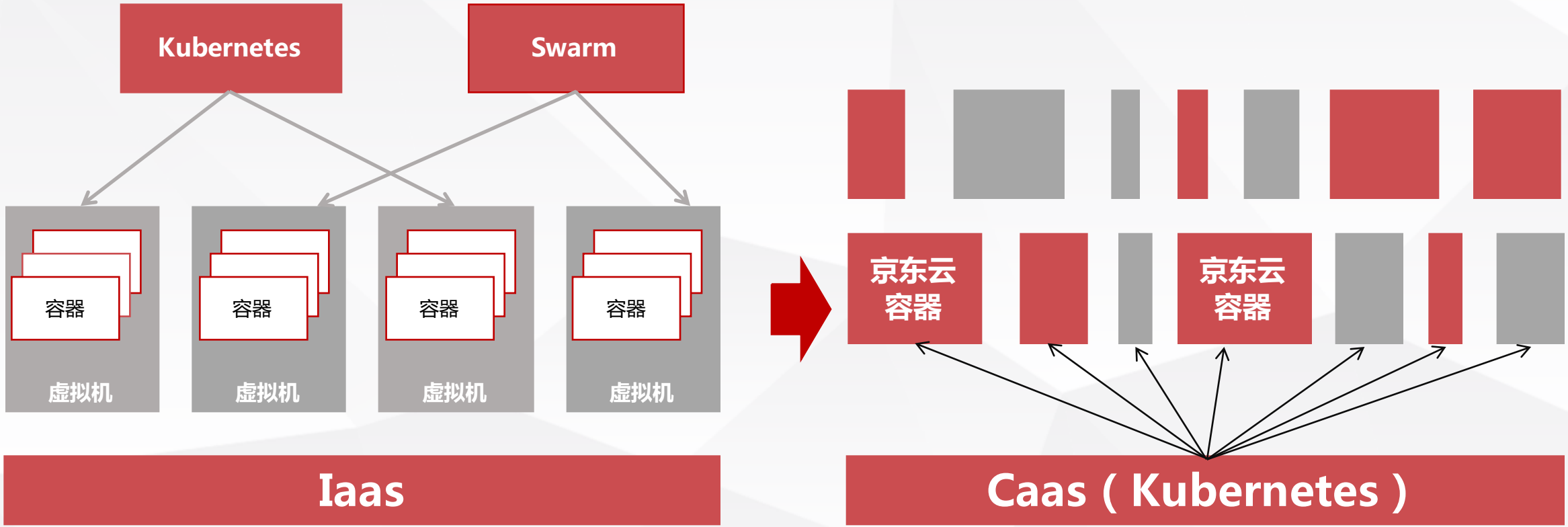
- 多租户环境下，基于iptables的默认kube-proxy无法访问到所有Pod，因为这些Pod会被隔离在不同网络当中
- HAproxy将负责代理该Pod namespace中的全部Service实例，Kube-proxy则根据标准的OnServiceUpdate 和 OnEndpointsUpdate流程对这些后端服务器进行更新

持久化存储

- 接数据卷作为块存储设备直接添加至Pod当中
- kubelet使用京东云容器运行时，用户新声明的数据卷可直接与Pod进行挂载

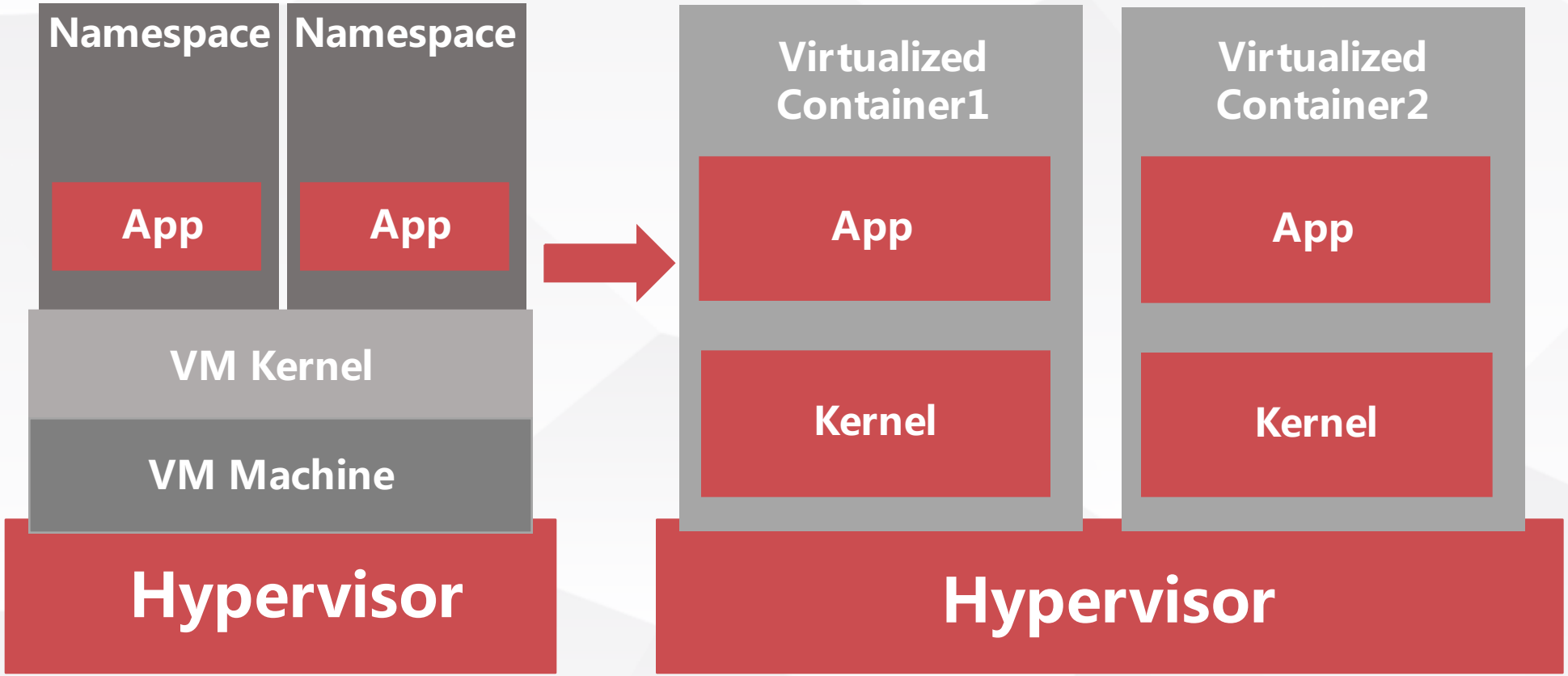
“蜂鸟”容器K8s应用

- 利用隔离性，减少间接层



“蜂鸟”特性——安全隔离、快速启动

- 利用隔离性，减少间接层



“蜂鸟”特性二——生态兼容

标准化

1 接口标准化

兼容标准Docker API 管理工具

2 模块标准化

兼容原生Docker Image

兼容性

1 平台兼容性

兼容现有虚拟化平台（分布式存储、SDN、在线迁移）

2 应用兼容性

1）提供原生的云上的Docker 使用体验
2）让容器应用更加通用化

“蜂鸟”特性三——多样化配置


灵活的计费模式
1、秒级计费，节省资源成本；
2、运行状态计费，停止状态不计费，成本投入最优；
微小型业务
最小配置：1C64M
超大型业务
最大配置：32C128G
可选配置：25种

目录页

Contents Page

容器服务典型应用场景——微服务

容器够小

- 解决微服务对机器数量的诉求

容器独立

- 解决多语言问题

开发环境与生产环境相同

- 单机开发、提升效率

代码/image一体化

- 可复用管理系统

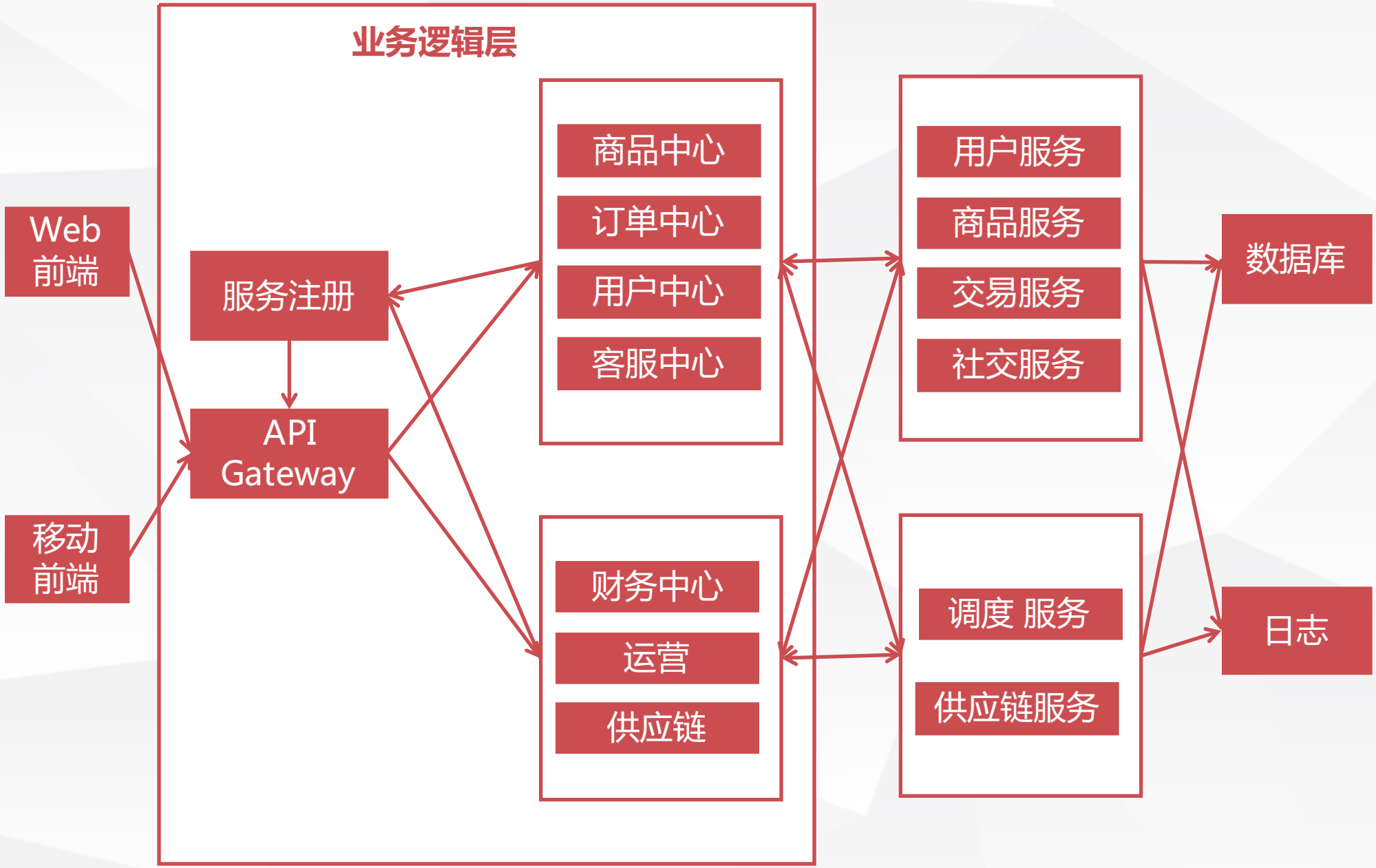
容器的横向与纵向扩容

- 可复制
- 可动态调节CPU与内存

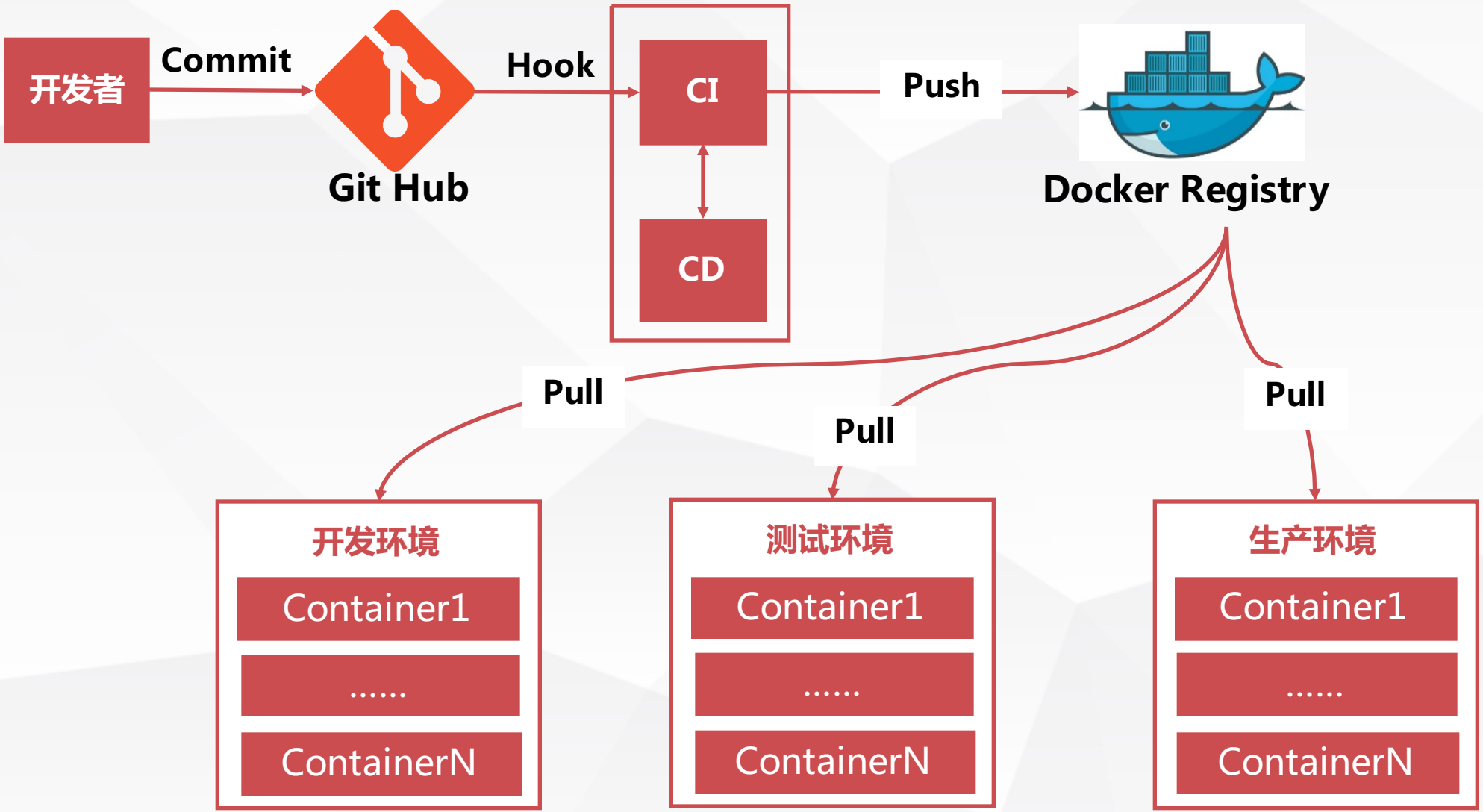
容器效率高

- 省钱

容器服务典型应用场景——微服务



容器服务典型应用场景——DevOps



谢谢！