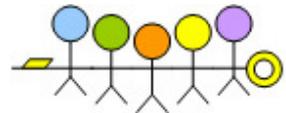


火龙果•整理

uml.org.cn

面向微服务的企业云计算架构转型



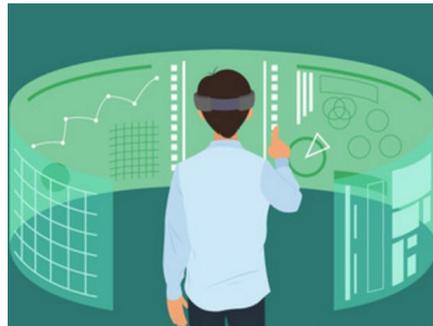
虚拟时空交易与现实时空交付的数字化时代



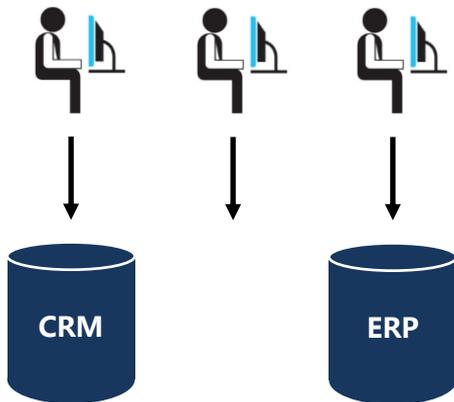
互联网
信息互通



移动互联网
时间自由
空间自由

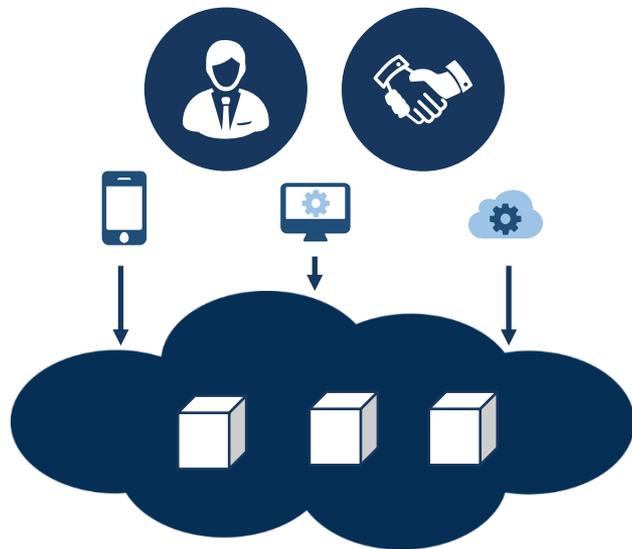


互联网+
虚拟时空&现实时空
交织一体



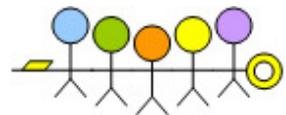
传统应用

- 服务内部用户为主
- 需求明确、功能全，覆盖广，大集成，中央控制，适合稳定发展阶段
- 刚性强，难以快速变化，维护成本高，快速变革的新业态无法支持

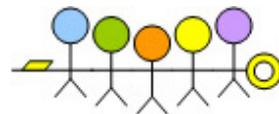


新兴互联网应用

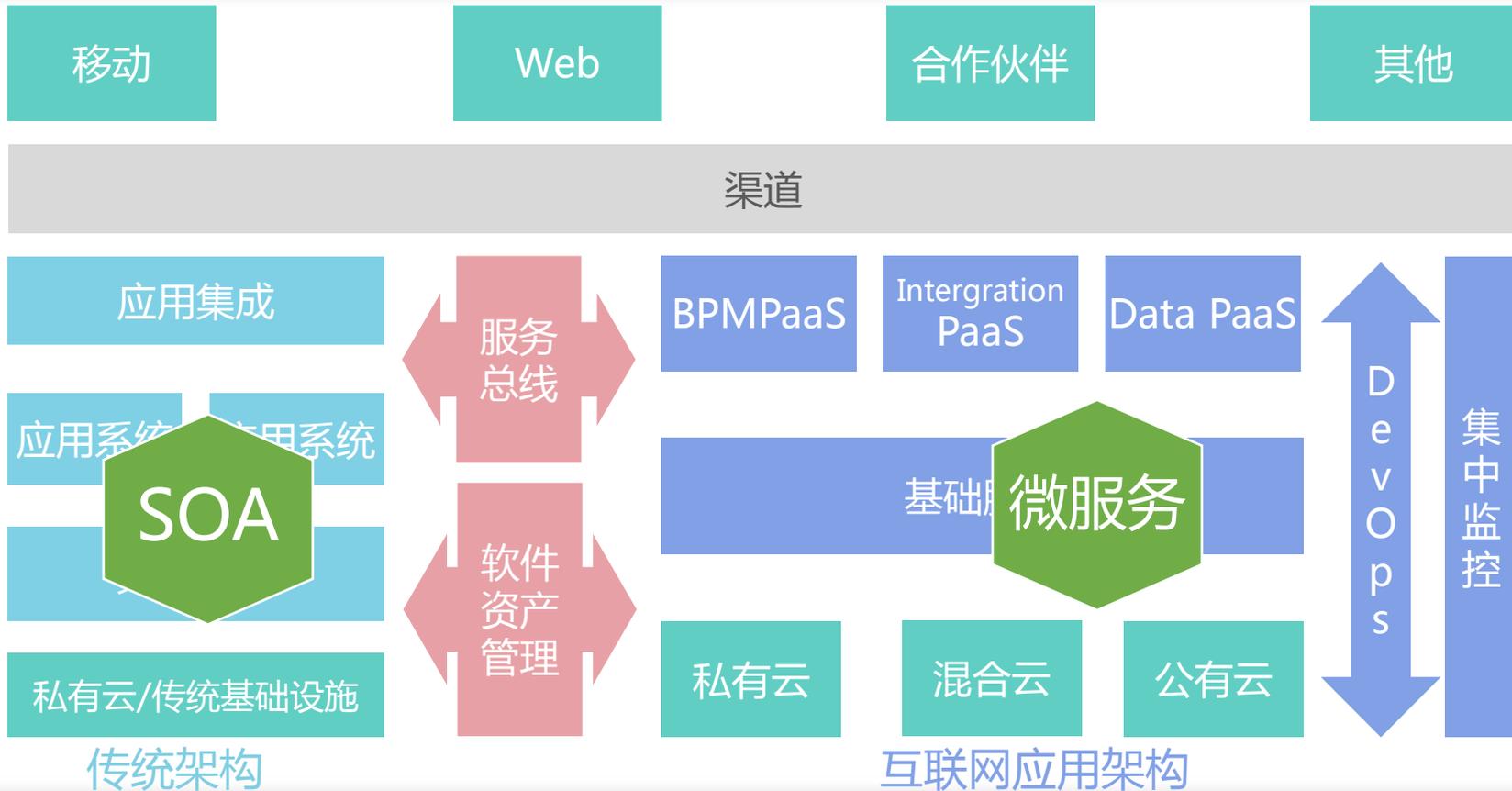
- 服务外部客户和合作伙伴
- 需求变动快，功能简单，独立和分散，分布式进化，一切都从零开始，业务与IT无法分开，需要快速创新
- 运用规模变化大，大范围广泛的尝试，易失败（淘汰），对业务弹性、快速发布要求高



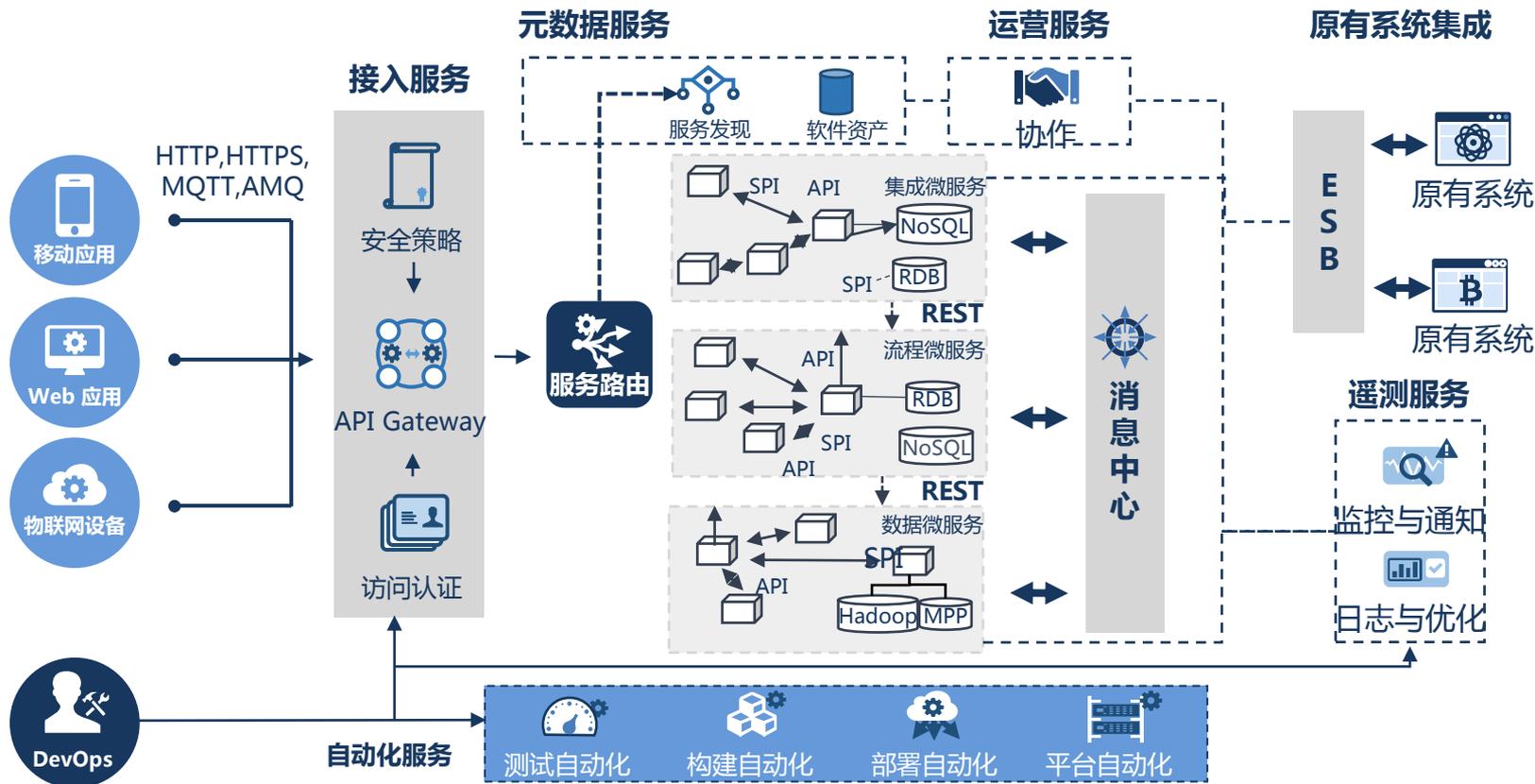
支撑企业互联网转型的混合架构



火龙果•整理
uml.org.cn

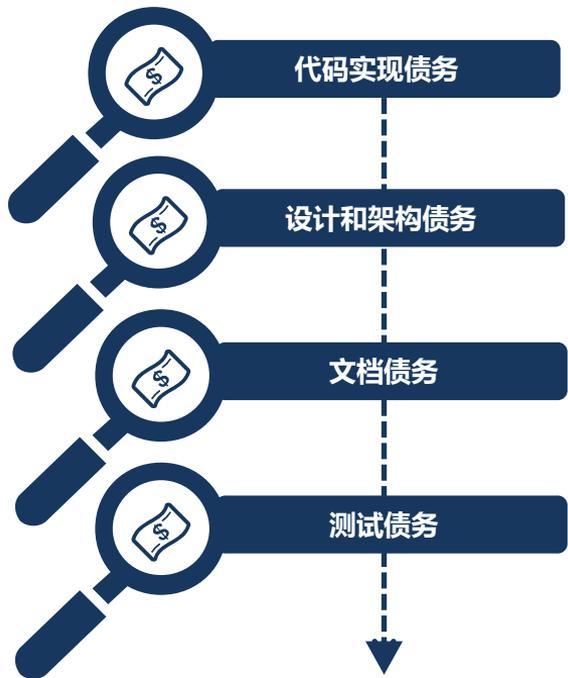


基于微服务的企业云计算架构





原有信息系统中存在未知的技术债务不适应微服务模式



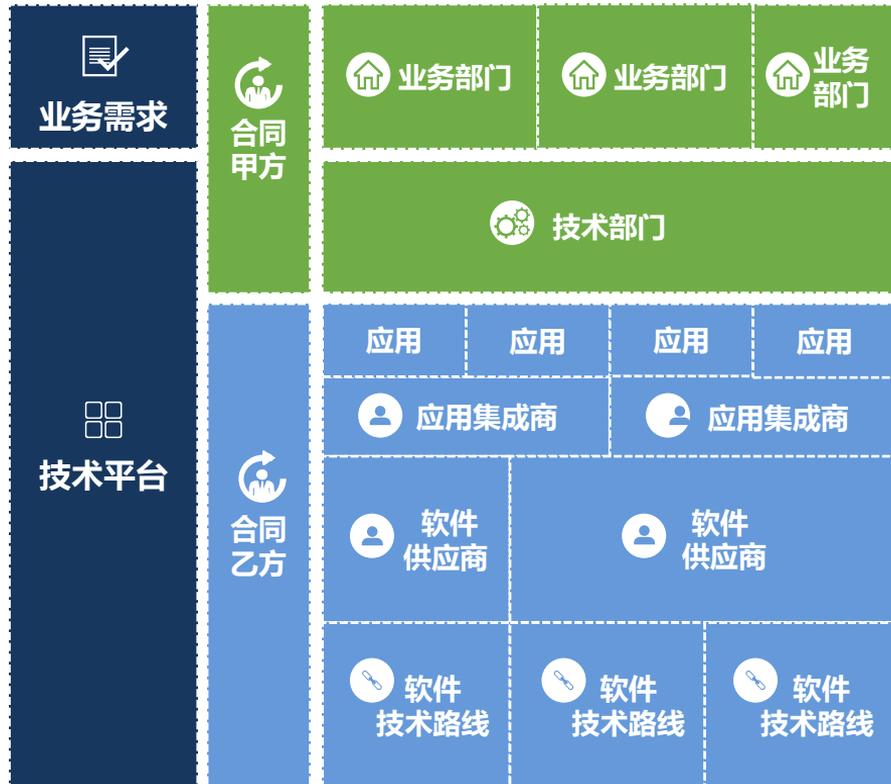
产生原因：技术维度



产生原因：业务或管理维度



传统信息系统建设中过高的隐性成本阻碍业务快速推出



业务成本

- 需求沟通
- 业务耦合

技术成本

- 学习成本
- 使用成本
- 运维成本
- 迭代成本

供应商竞争成本

- 多方曲线沟通
- 融合多个技术
- 重复单个技术

管理成本

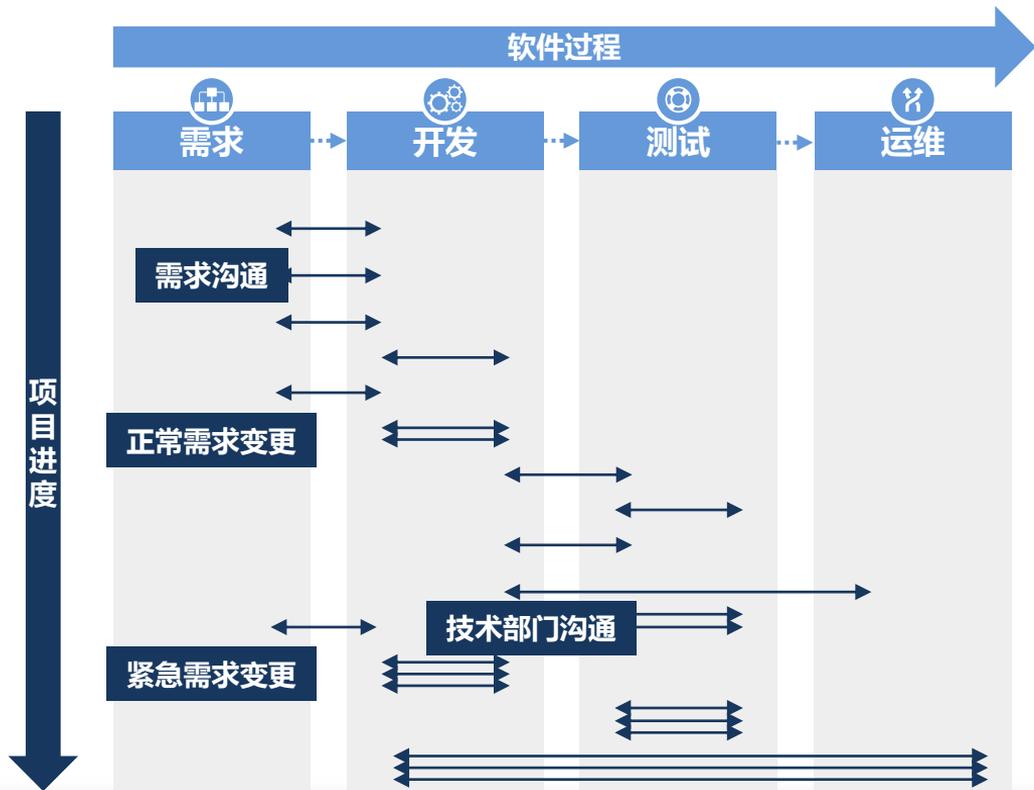
- 技术债务
- 流程环节

安全成本

- 效率下降
- 隐私保护
- 政策法规
- 应急响应
- 软件缺陷
- 网络隔离



失真的知识传递导致整体交付速度严重降低



- 市场变化、个性化用户需求增加快求变更速度，需求描述不明确
- 系统架构复杂，设计、开发、测试、运维理解不清晰。
- 跨部门，跨团队沟通效率不足
- 「开发」在交付过程中占比不高，但却承担主要责任



微服务架构升级对可用性提出更高的要求



依赖关系
变复杂

前后台调试

App
调试和适配

RPC
网络延迟

系统设计技术栈变厚

系统间调试

环境配置增多

调用链路变长

人员知识面系统性要求变高



部署内容
增多

物理部署包个数 = 应用版本数 * 目标环境数 * 系统组件数 * 冗余实例数

文档过期黑洞

程序不友好

自动化程度低

配置项增多：包含环境相关配置项 + 依赖相关配置项 + 安全相关配置项



终端
移动化

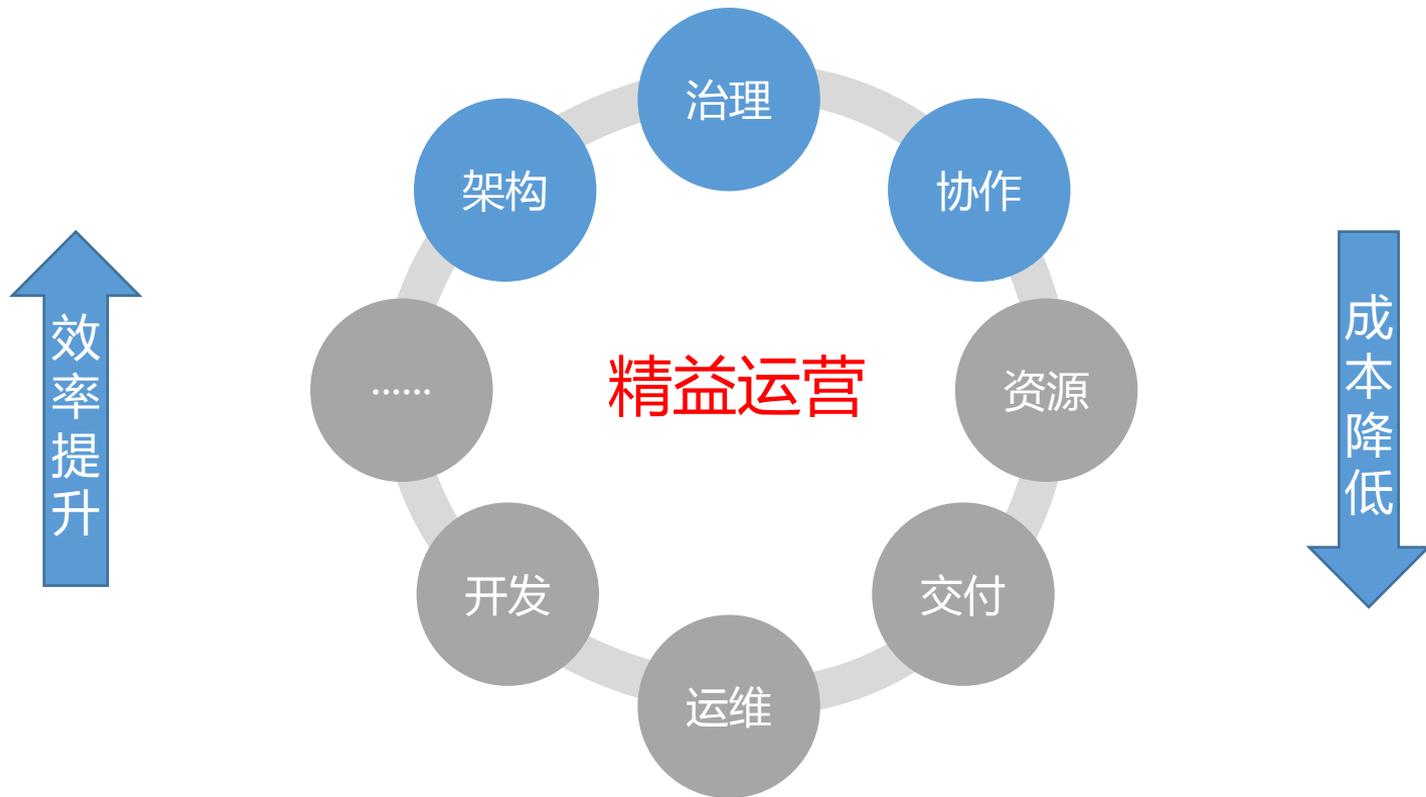
移动App上架周期长节奏慢

最终用户直达率高

个性化需求进一步提高

Cloud+Client系统架构对于终端的管理

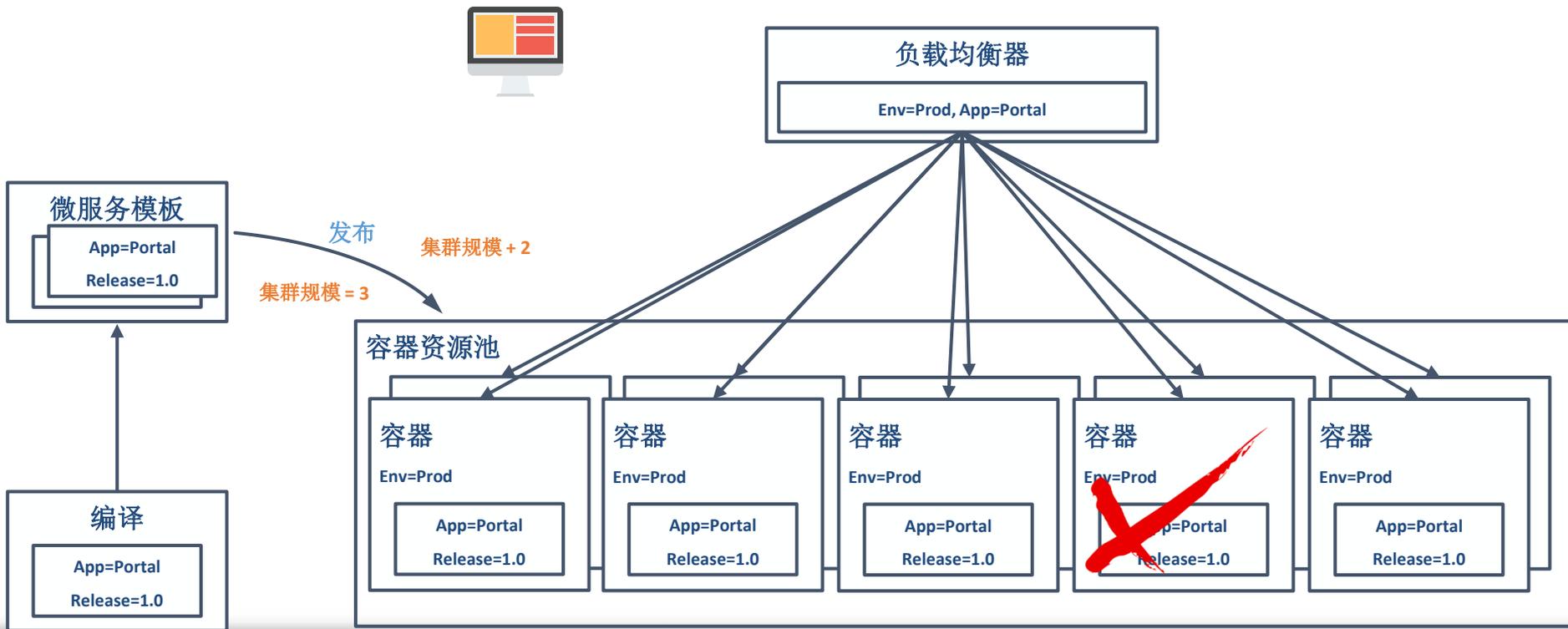
加剧系统离散化



采用Kubernetes提升微服务的架构能力



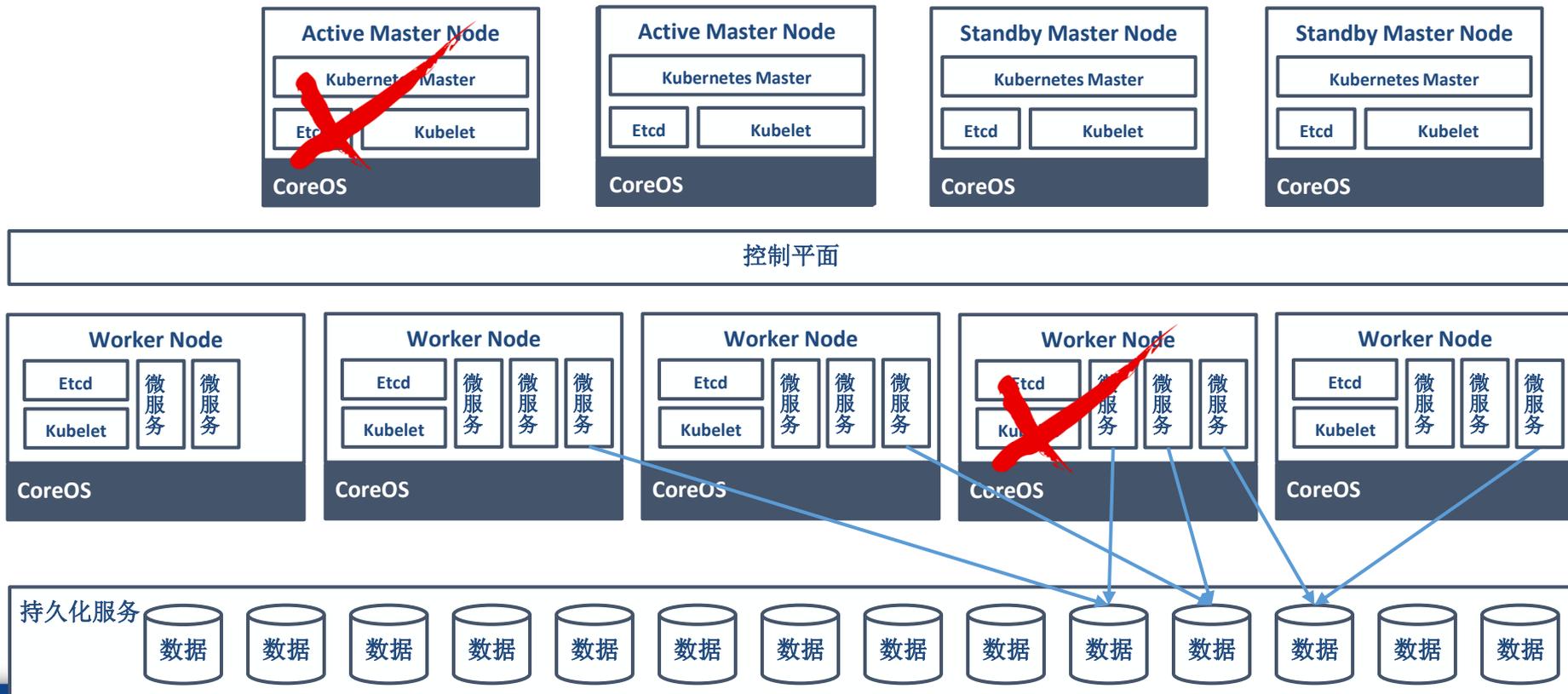
基于 Kubernetes 实现微服务发布、高可用和集群伸缩，优于OpenSatck

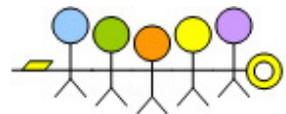


采用Kubernetes提升微服务的架构能力

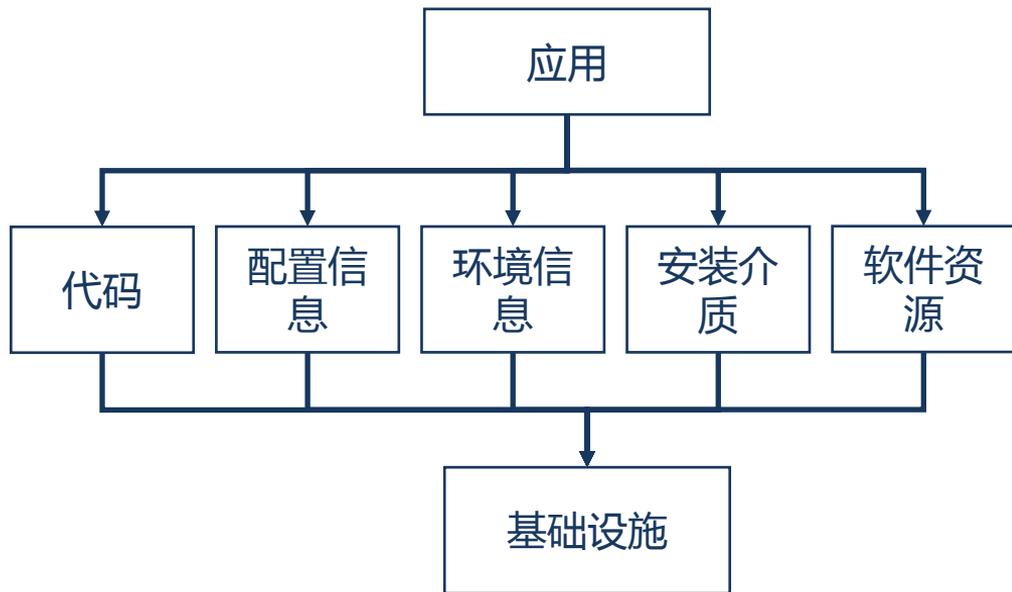
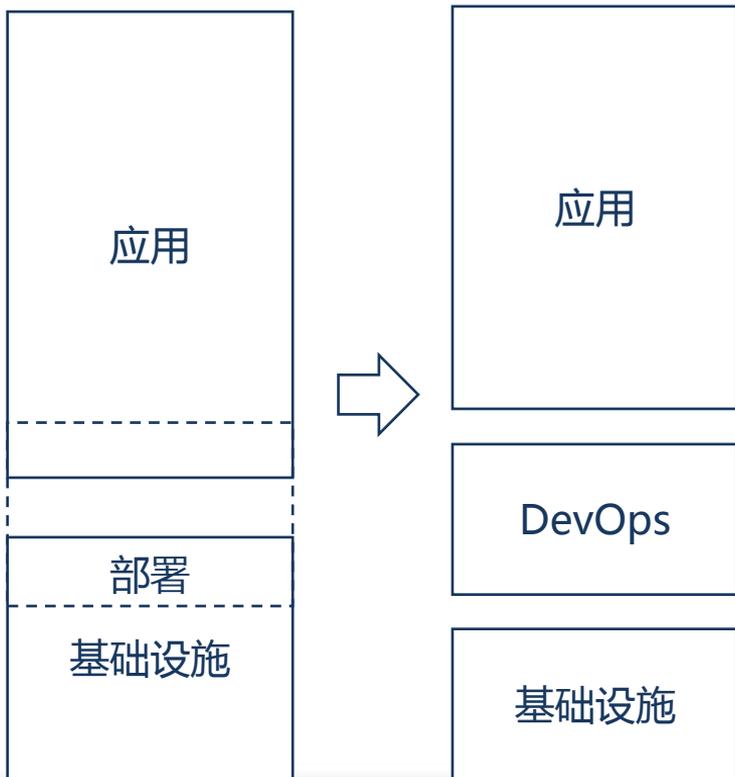


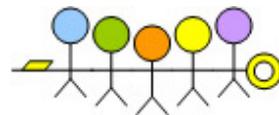
基于 Kubernetes 实现微服务高可用，优于OpenSatck



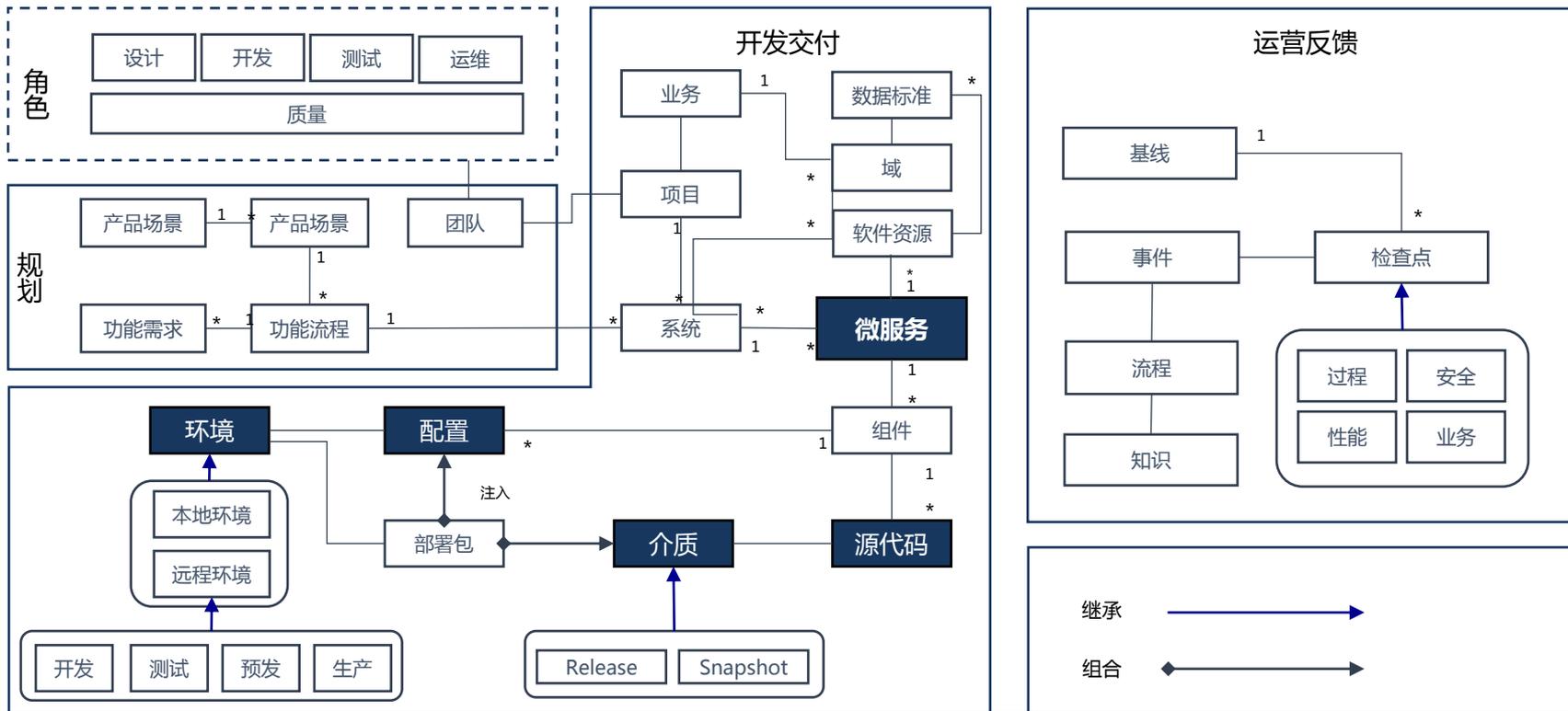


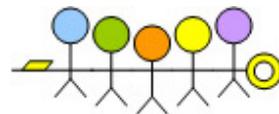
实现代码、配置、运行环境的分离



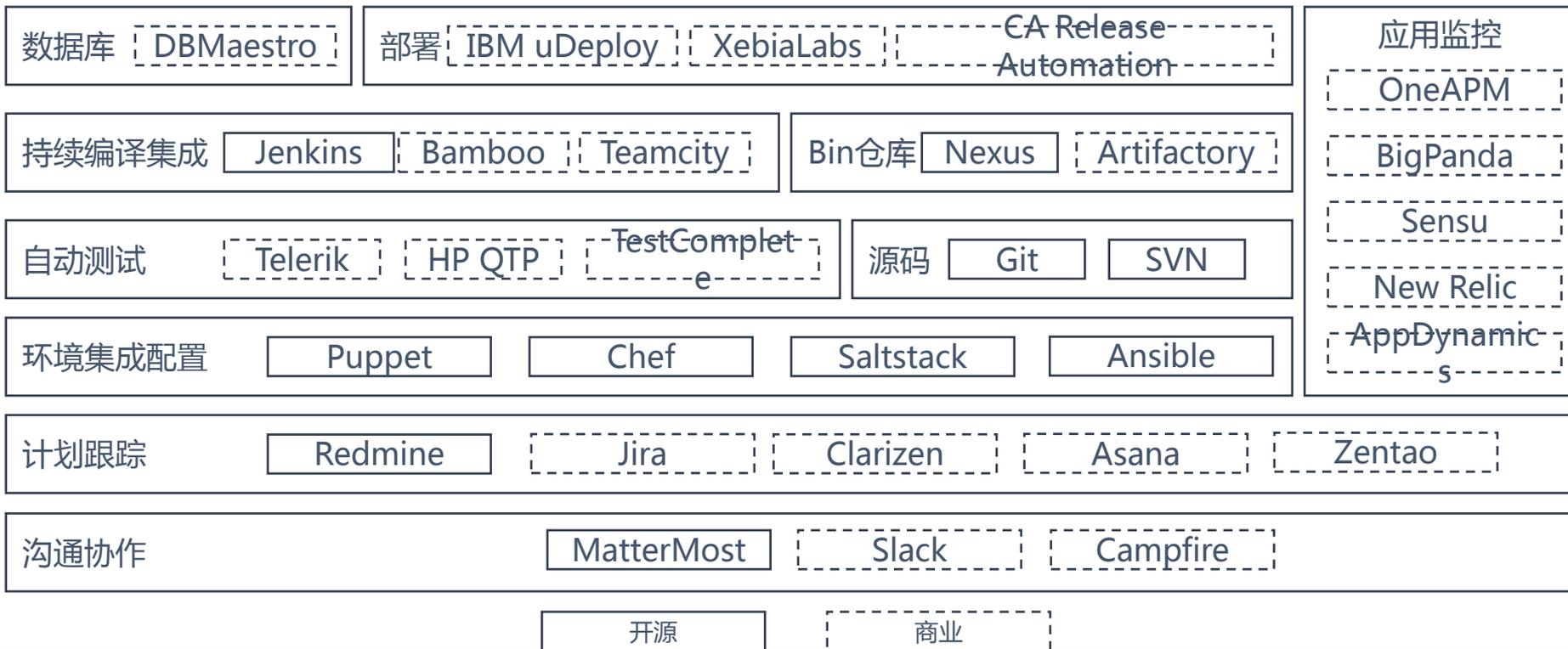


微服务 DevOps 的概念模型，实现代码、配置、运行环境的分离

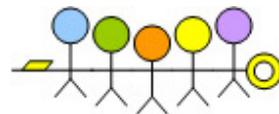




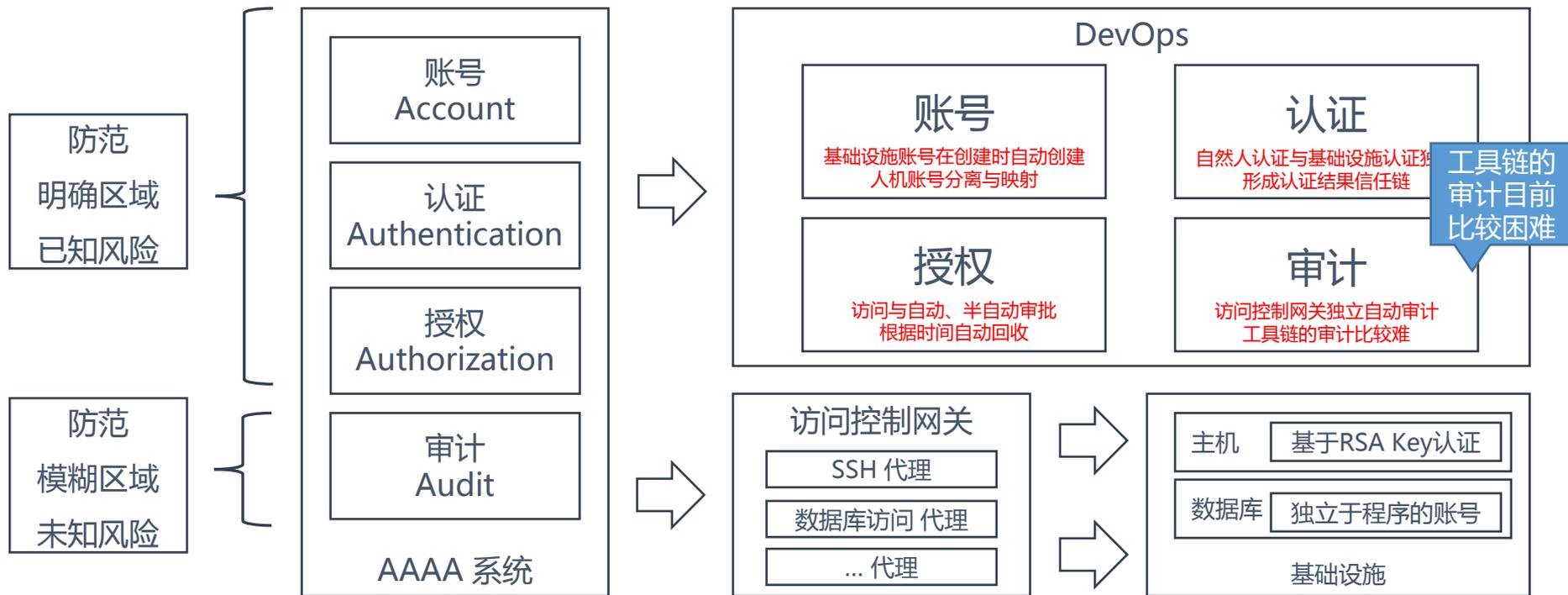
基于概念模型，集成现有的 DevOps 领域工具链

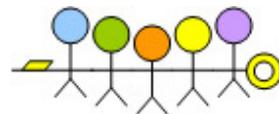


提高协作效率：微服务的DevOps

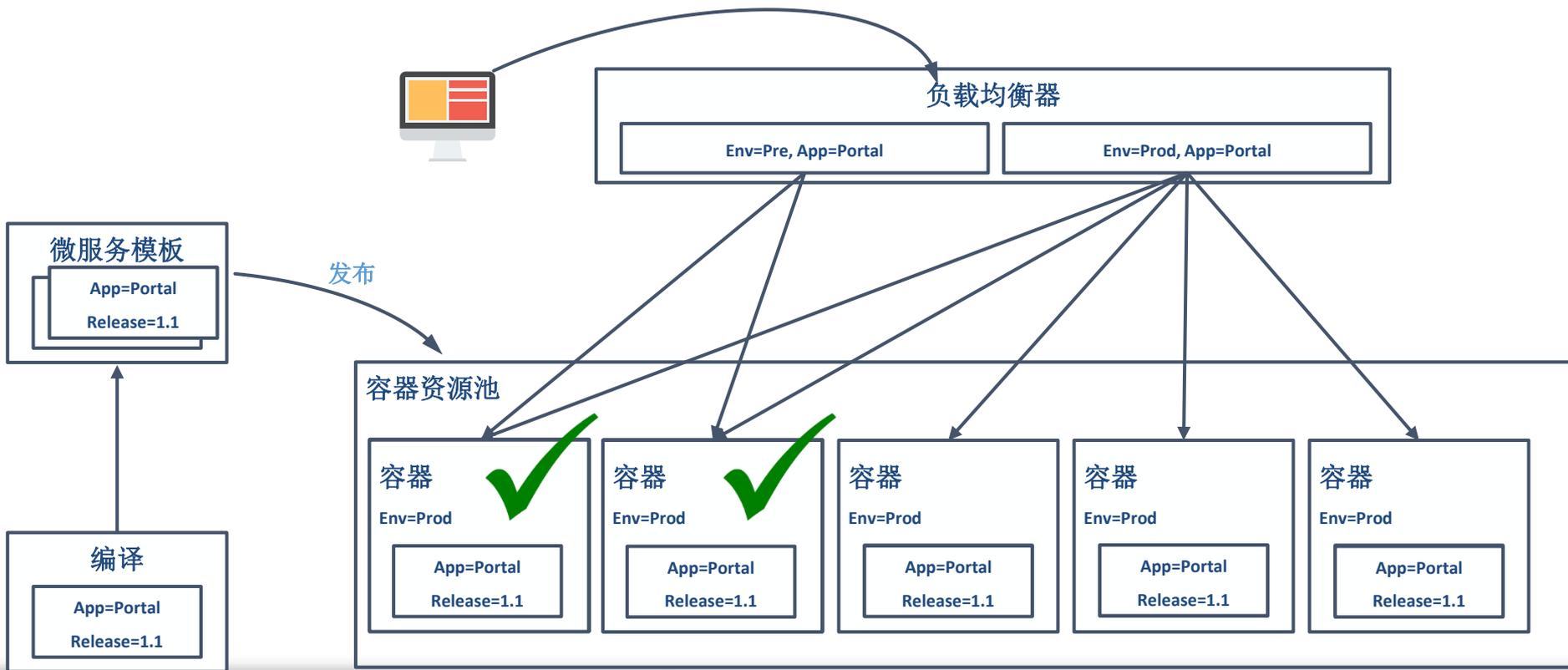


基于概念模型，结合 AAAA，集成 DevOps 领域工具链



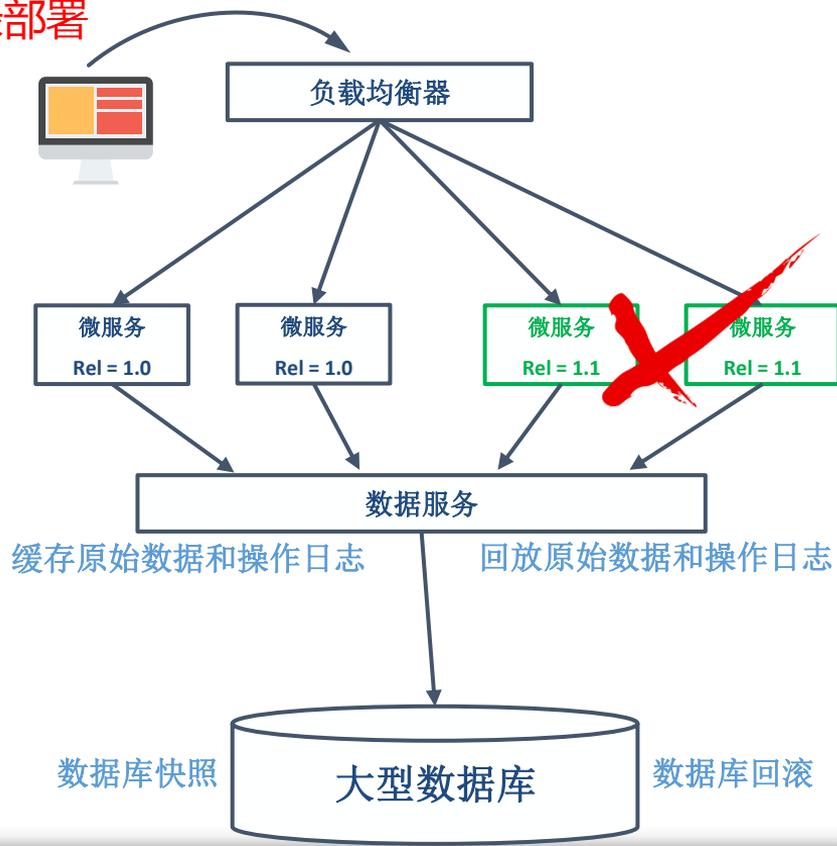
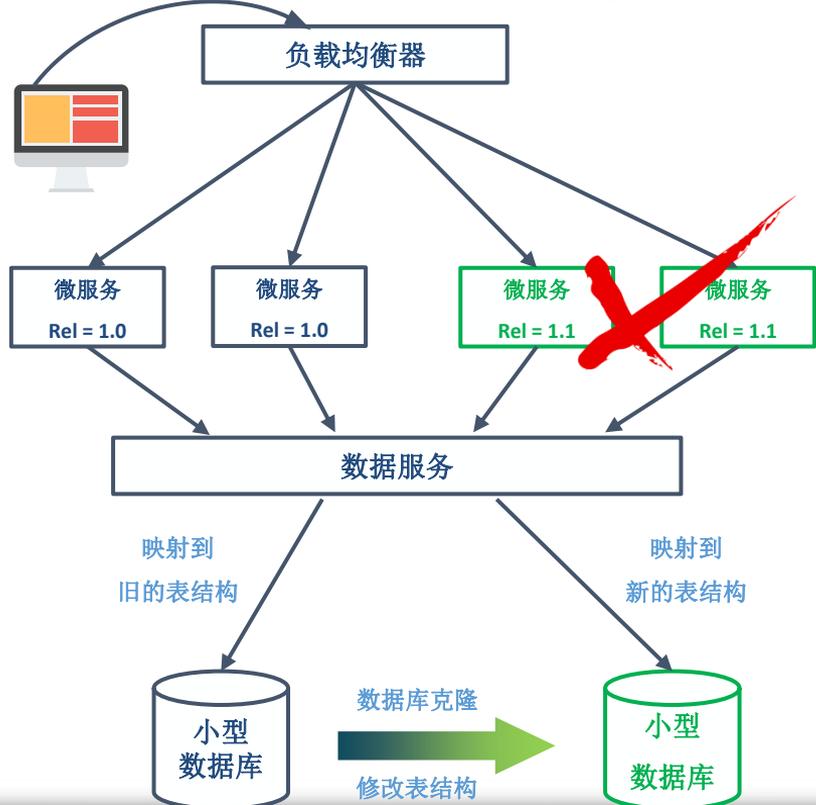


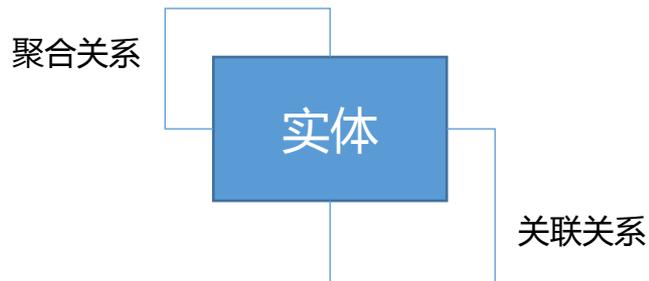
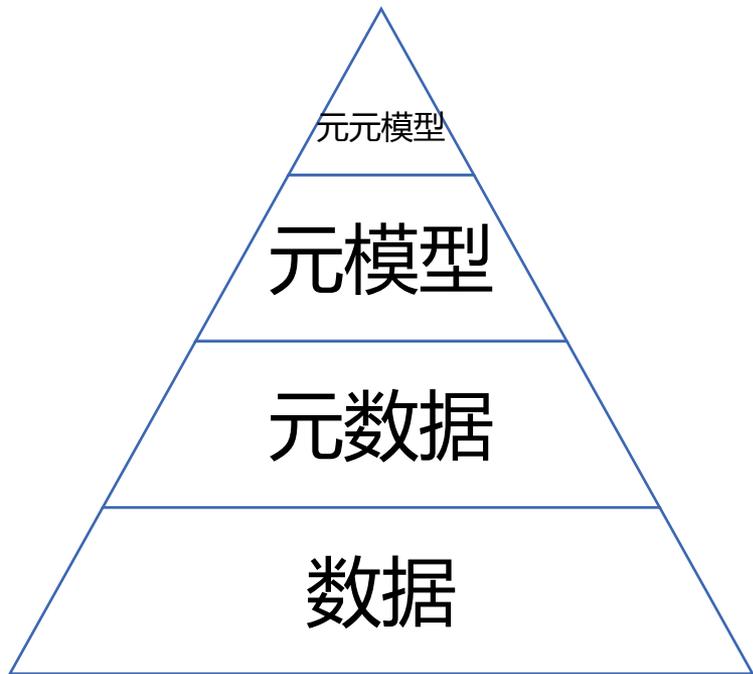
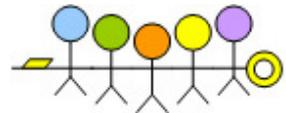
基于 Kubernetes 实现微服务新版本发布、金丝雀测试、预发和滚动更新

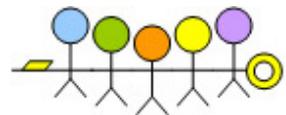




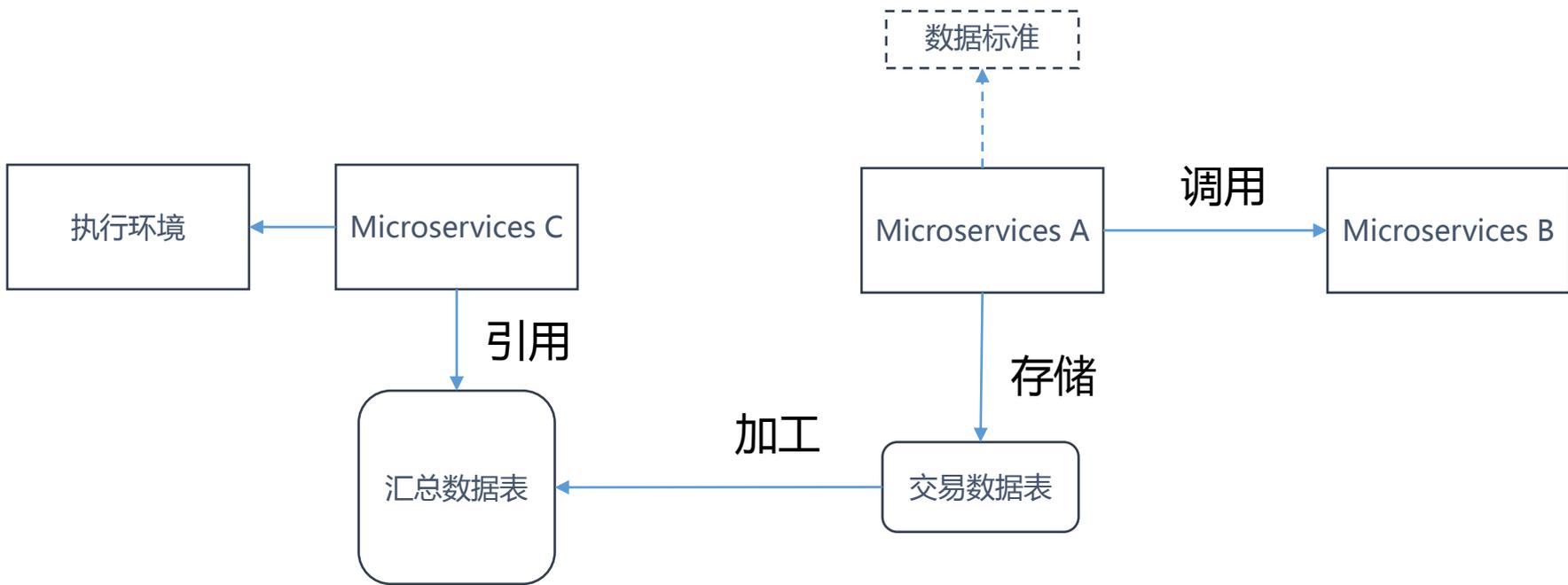
基于 Kubernetes 实现微服务和数据库的蓝绿部署





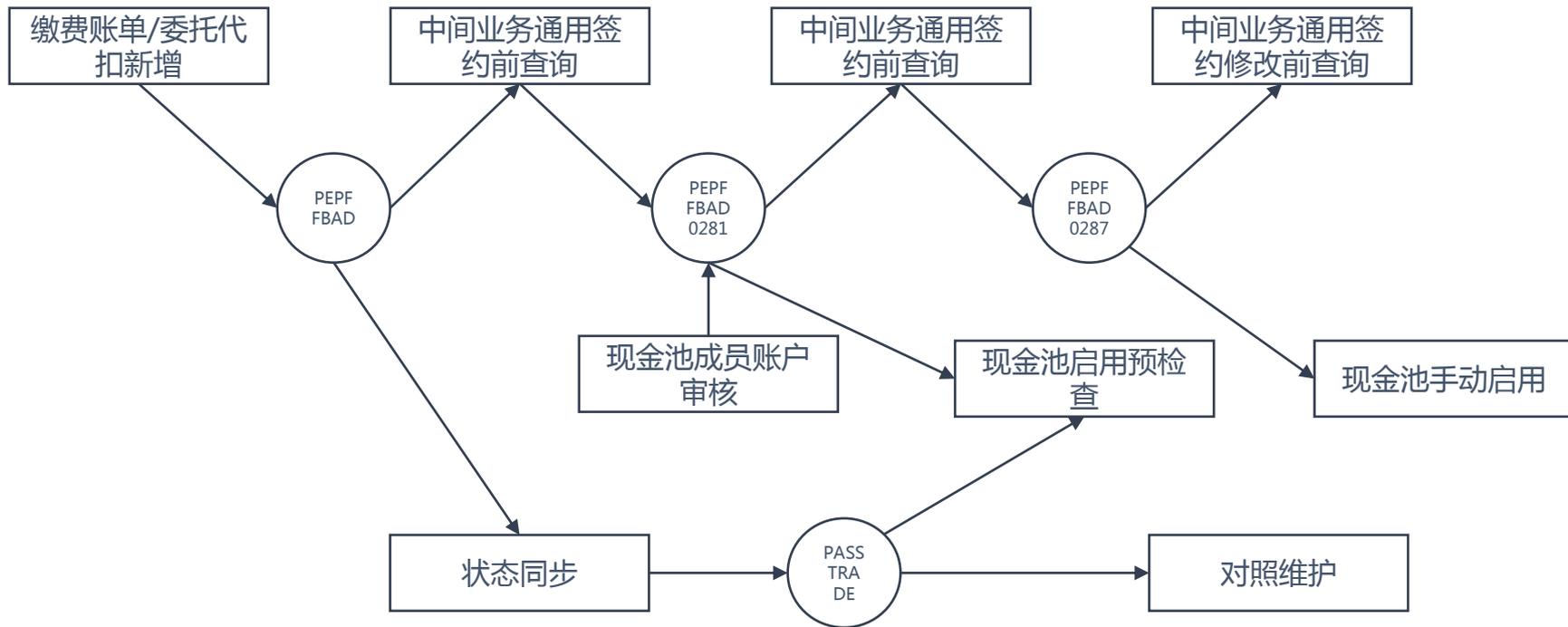


表述服务与服务、服务与资源间关系，明确版本变化



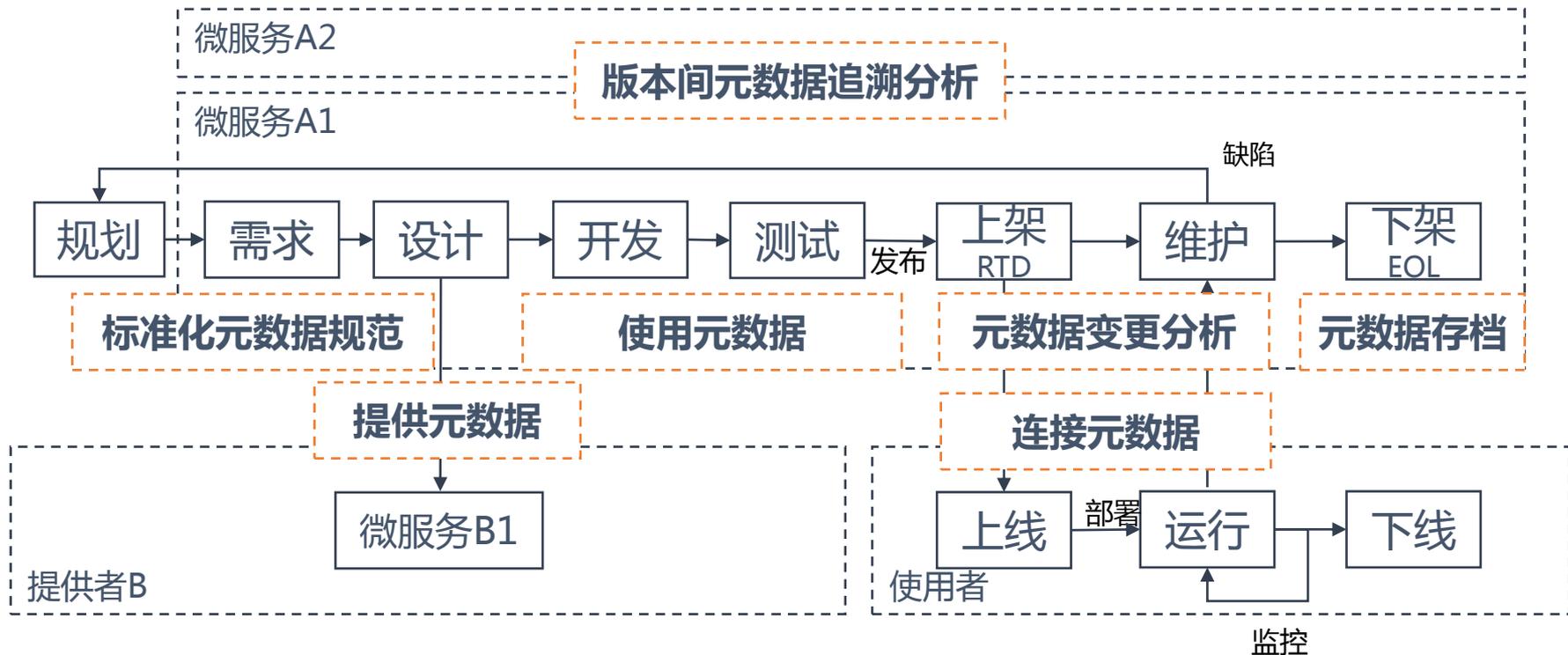


通过元数据，表述服务/资源之间关系，明确版本变化对其他服务的影响





通过元数据，实现版本溯源，加强对微服务生命周期管理的能力



关键技术提升



火龙果•整理
uml.org.cn

现有技术栈

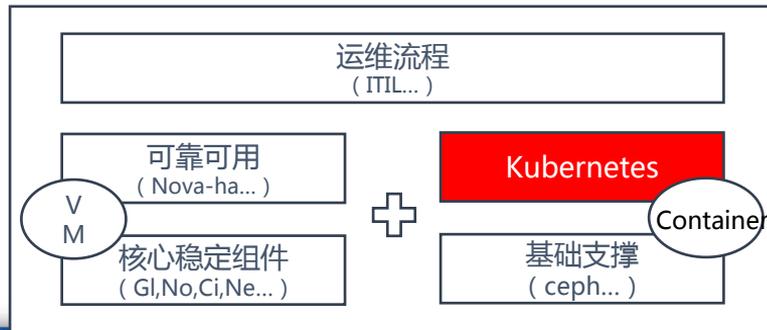
提升

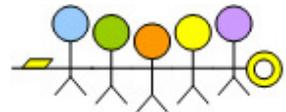


Mesos

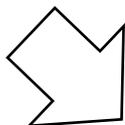
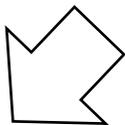
广义PaaS

做薄+做厚





产品开放式研发



开放内容

开放形式

研发过程
向内外公开



技术组件
开放研发



研发成果
向社会公开



微信群

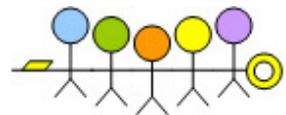
技术文章

在线培训

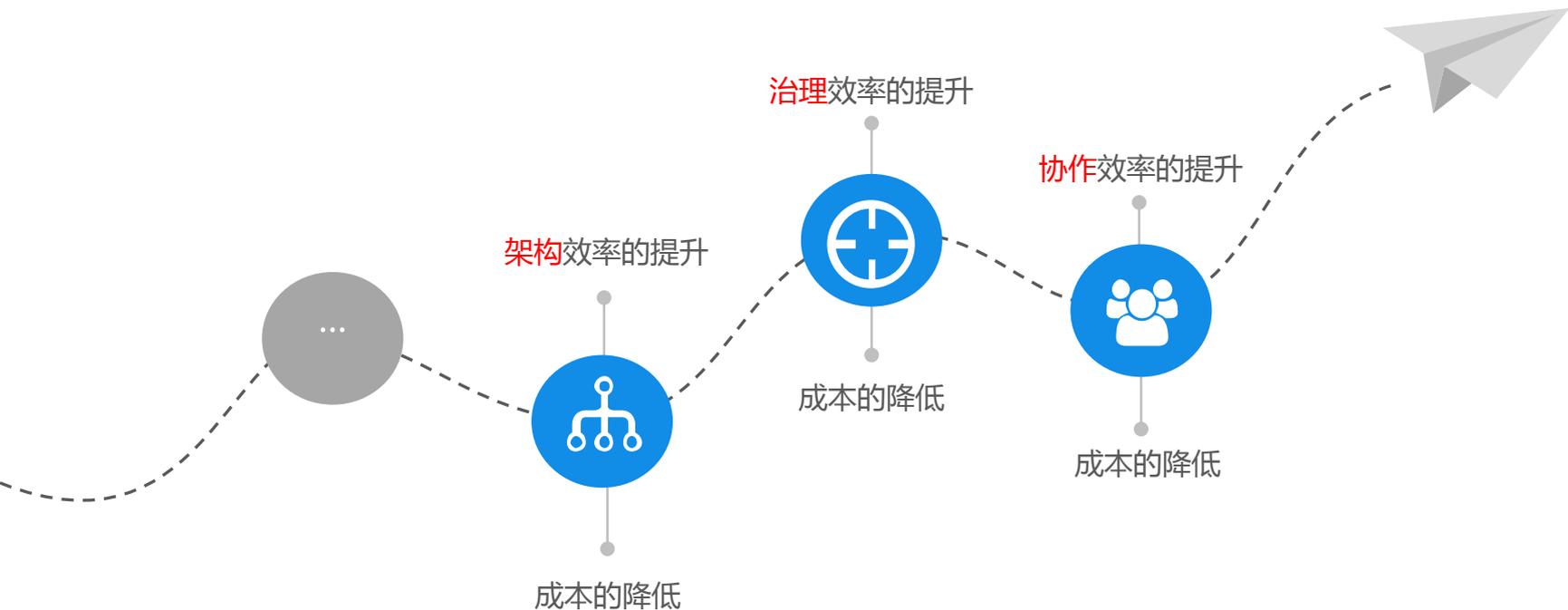
线下沙龙

微课堂

技术视频



从 IT 精益运营入手逐步实施



架构提升：提高微服务的可用性



 传统方式 Scale up & Scale out

纵向弹性扩展(8C)

当前负载($\leq 4C$)



资源突发高峰怎么办



超配？预置？

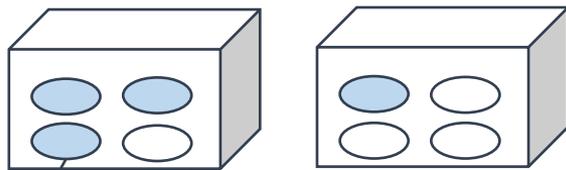
横向弹性扩展

当前负载



窗口怎么办？

 微服务容器模式



一个容器：0.125C-2C，1G

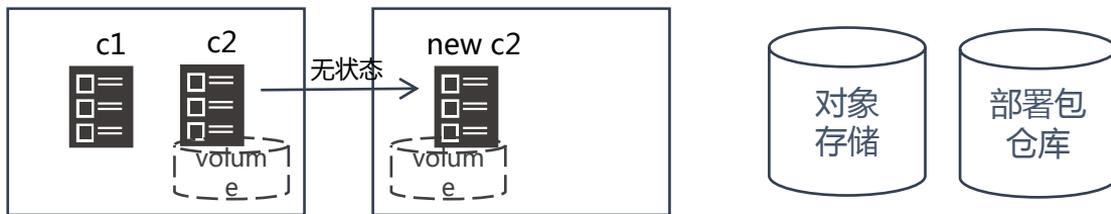
- 1 外部访问：DNS方式，nginx+lua
- 2 内部通信：直调，rest+etcd
- 3 关联服务，本地寻址优先

架构提升：提高微服务的可用性

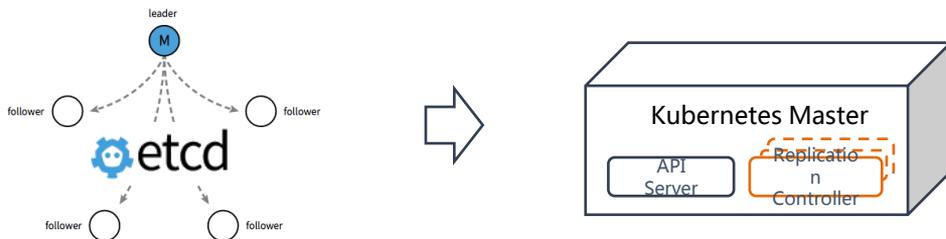


火龙果•整理
uml.org.cn

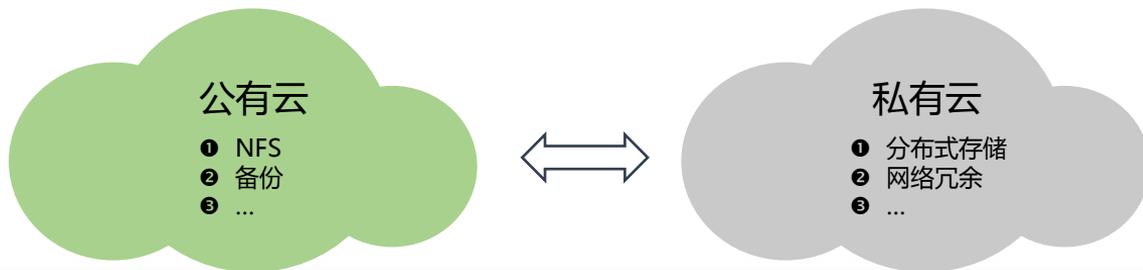
应用层

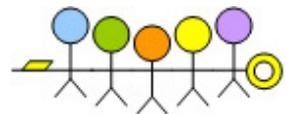


容器层



基础设施





从 IT 精益运营入手逐步实施

