面向DEVOPS的方方面面

DEVOPS 、CMDB 、持续交付、应用管理/BIGOPS、NEW ITSM及组织/实施与度量

优维科技(深圳)有限公司
DevOps管理专家

写在前面的话

1、基于交付链(Dev/Test/Ops)的全局优化，而非局部(Ops)优化

2、运维的问题不是仅仅运维侧的问题，是一个IT问题

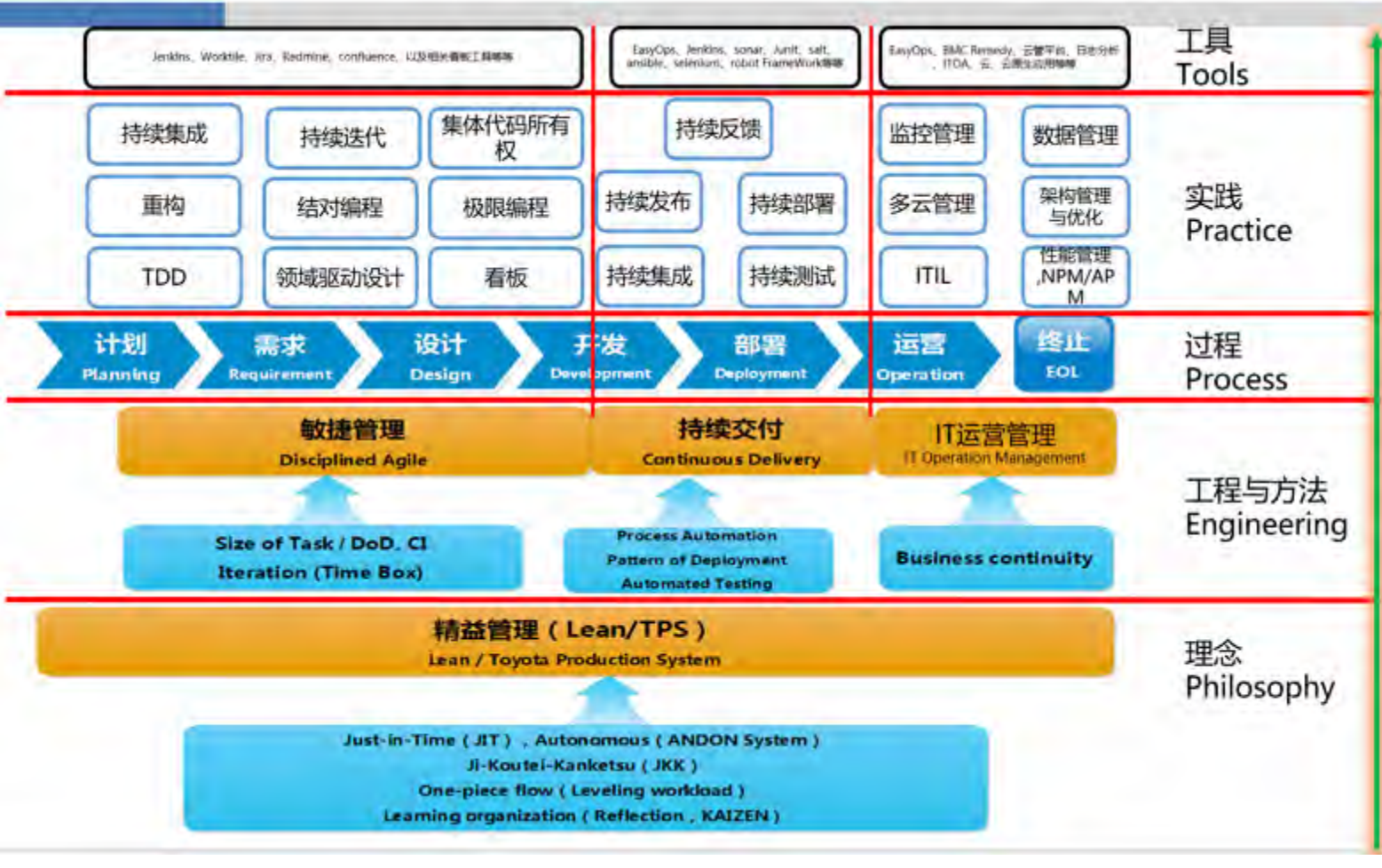
3、运维问题表面是复杂的(组织/人)，本质上是简单的(技术)

4、坚持技术是内部第一驱动力，业务是外部第一驱动力，很多问题便找到了答案

5、从运维到运营的蜕变，做好准备

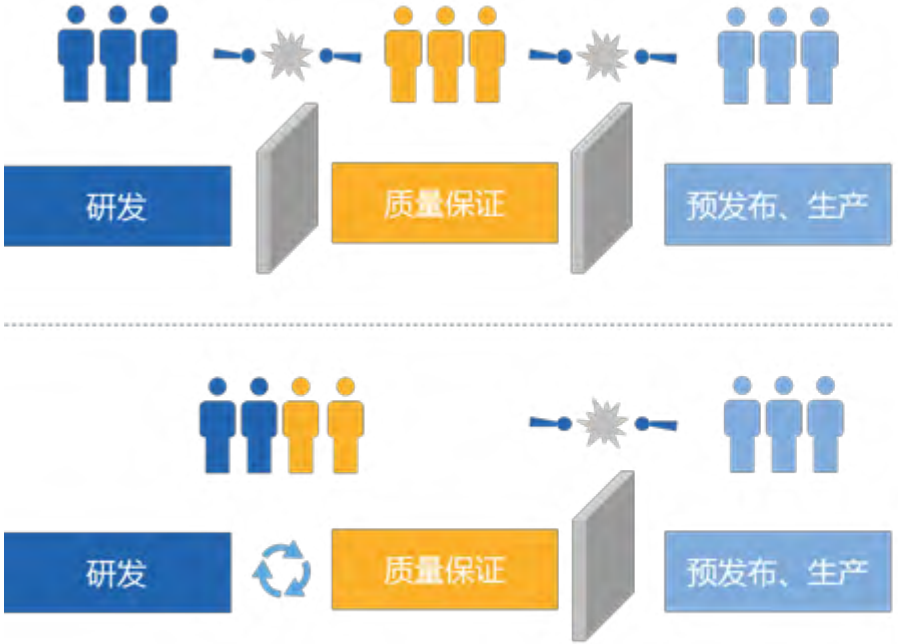
到底什么是DevOps

一张图彻底让你了解DEVOPS



- DevOps是一连串的工程实践的有机组合，其中包括敏捷管理、持续交付、IT服务管理等等。
- DevOps是关注整个业务/应用/服务生命周期的管理，把业务和IT的战略进行了对齐。
- DevOps以精益思想为基础，强调自动化、拉动式、“拒绝浪费、创造价值”等

DEVOPS是一种新型的IT交付模式



- DevOps让团队共享面向客户的价值、共享集成目标、共享质量责任
- DevOps让运维的作用变得更加凸显，此时需要全新的思维 / 平台 / 方法论来实现Dev的软件快速交付到Ops阶段，并且能够稳定地运营

DEVOPS整个模式

DevOps运维体系，分成横向和纵向两部分：

横向：DevOps运维6个维度（组织、过程、架构、工具、基础设施、度量）+文化

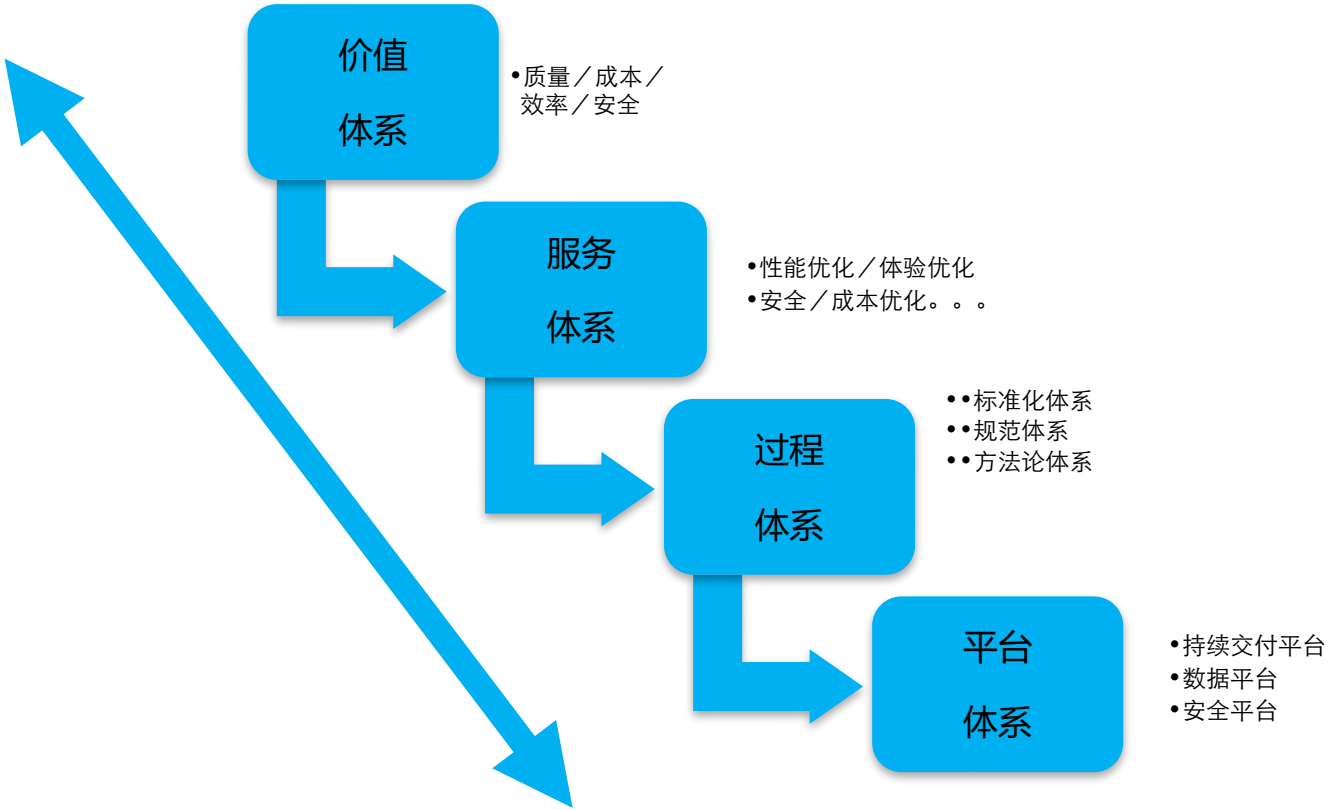
- 组织，DevOps首先必须打破组织之间的隔阂，其次DevOps团队建立面向产品而非项目的协作文化。
- 过程，轻量级流程和自动化工具的完美结合，确保企业的高度敏捷性；自动化为先，而后再流程。
- 架构，架构是DevOps成功的重要影响因素；巨石、单体架构是快速交付的最大障碍；架构与持续交付紧密联系。
- 工具，运维平台或者工具在提升运维效率的同时，也在影响着质量和成本；高效的应用化平台能力确保故障快速恢复
- 基础设施，从虚拟化到IaaS到容器化，敏捷的基础设施是高效运维的基础。
- 度量：建立全面的DevOps运维度量体系，正向驱动运维、研发、测试团队不断完成面向用户的交付。

纵向：DevOps能力评估模型（五个等级）

DEVOPS的业务价值

- 通过持续服务交付价值链打破孤岛
- 整合开发和运维的能力成为一个协作的团队
- 端到端持续服务交付流程的变革
- 对新的应用和服务，加快且缩短实现价值的时间
- 不影响安全性，兼容性，性能

DEVOPS运维平台在运维体系中的位置



DEVOPS运维平台的能力设计环

- 全栈的运维平台，是要考虑各种角色 / 场景 / 能力下的交付需求，面向能力域的交付能力是一种专业化的服务能力交付。

- 价值流分析法是基于场景的分析，端到端的交付实现。

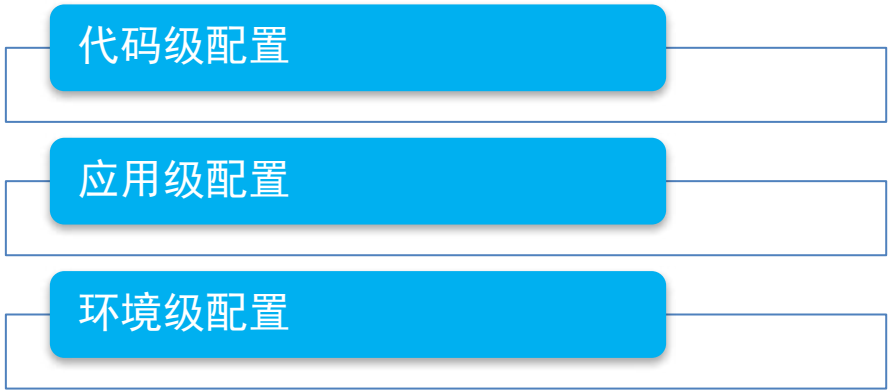
DEVOPS运维平台概念模型

Operation As Service，运营及服务，是以EasyOps PaaS平台能力为基础，实现了运维的IT能力和业务能力的对接

DEVOPS运维全平台系统概览(二级)

CMDB，从配置到IT资源的改变（问题篇）

真正的配置管理是什么？



- 代码的配置管理SCM。聚焦在源代码的整个管理过程。
- 应用的配置管理ACM。聚焦在应用相关的配置，一般由应用包产生。从管理模式上来说，该配置可以升级成环境的配置管理。
- 环境的配置管理ECM。系统级的环境配置管理，比如说系统的内核调优参数

你的CMDB=资产管理

资产管理	配置管理
目的:管理资产成本、合同和使用情况	目的:作为基础为 ITIL 提供基础的模型
价值:更低的资产总拥有成本,减少购买	价值:面向业务管理,保证业务服务的可靠 性、质量,通过关联的其他 ITIL 流程来保证
资产:可以被基于合同跟踪的 IT 组件	配置项:为了其运维角度而被管理的逻辑和 物理资源
区别:不可能作为配置项来管理的资产包括在仓库中的显示器,打印机、服务器等等	区别:不可能作为资产来管理的配置项包括 客户化的 java 组件、流程、服务等等.
关系:资产之间的关系被维护以用于回收流程 。	关系:配置项之间的关系被维护以评估变更 风险、分析原因、评估服务影响 。

你的组织对CMDB存在偏见

表现	描述
把CMDB当成一个数据库来看待	认为CMDB就是一个数据库，里面存放的就是一些配置信息，不考虑其上的管理流程
CMDB的管理方法论没有统一	CMDB的管理方法论，自动化是一方面，但也要关注流程的作用
认为CMDB的管理粒度越细越好	越细在人工、无场景的情况下，失败的可能性就越大
CMDB是一次建设就可以完善的	CMDB是数据，是一个持续完善、动态变化的过程
缺少领导的支持	因为CMDB的建设一个全员建设和参与的过程，必须要领导深度支持
你的配置项缺少Owner	缺少Owner的配置项维护是非常失败的，每个配置项必须落实到具体的责任人。

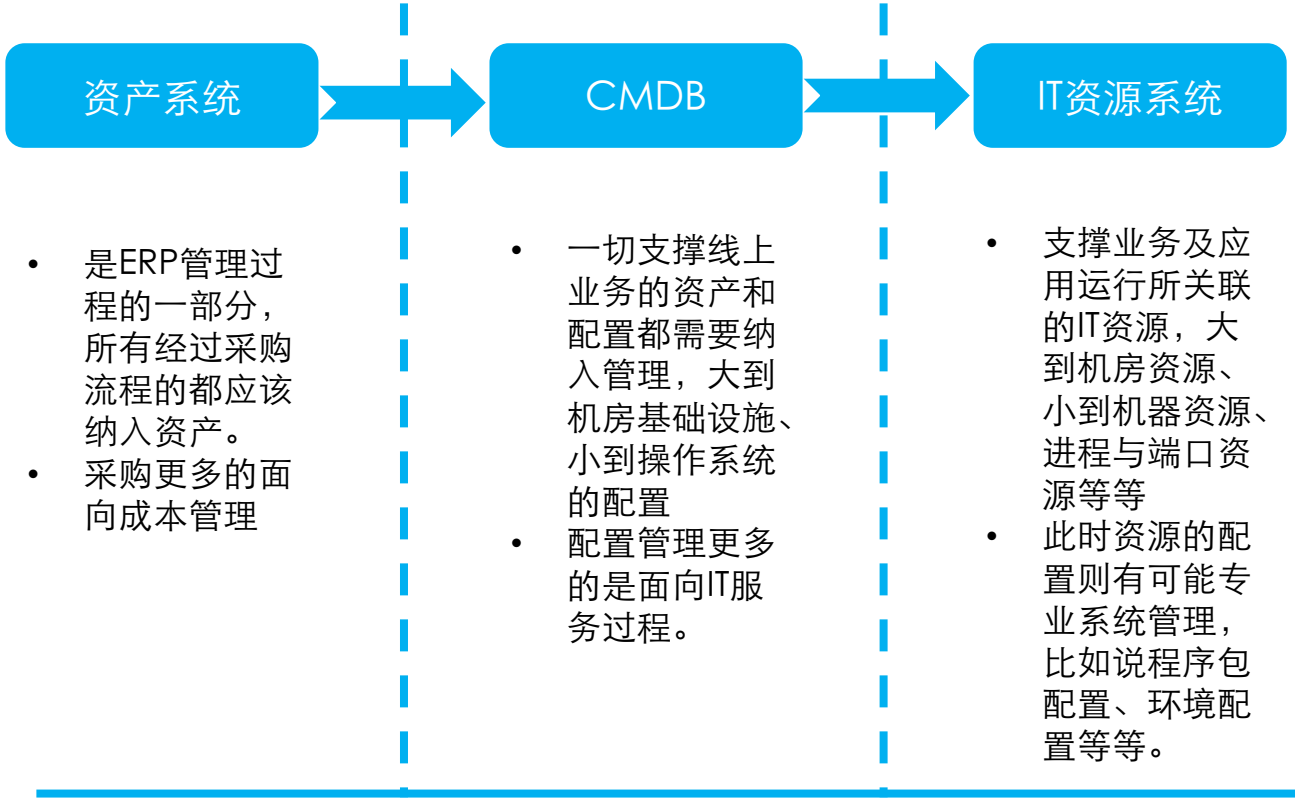
你的CMDB=数据中心的CMDB

表现	描述
数据中心基础资源组负责CMDB建设	这是CMDB失败的大部分原因所在，纯粹是为了上CMDB而上CMDB。
DBA负责CMDB建设	CMDB变成一个真正的数据库
一次建设，运动式维护	这是一种结果驱动式的CMDB建设，而非过程式的CMDB建设，让CMDB的作用逐渐流于形式。
CMDB没有Excel维护方便	从表格的能力管理来说，基于EXCEL信息管理模式绝对是远超所有系统，但管理能力的嵌入，才是CMDB的核心能力。
不重视配置标准化能力	配置越标准化管理成本就越低

你的CMDB缺少对场景化

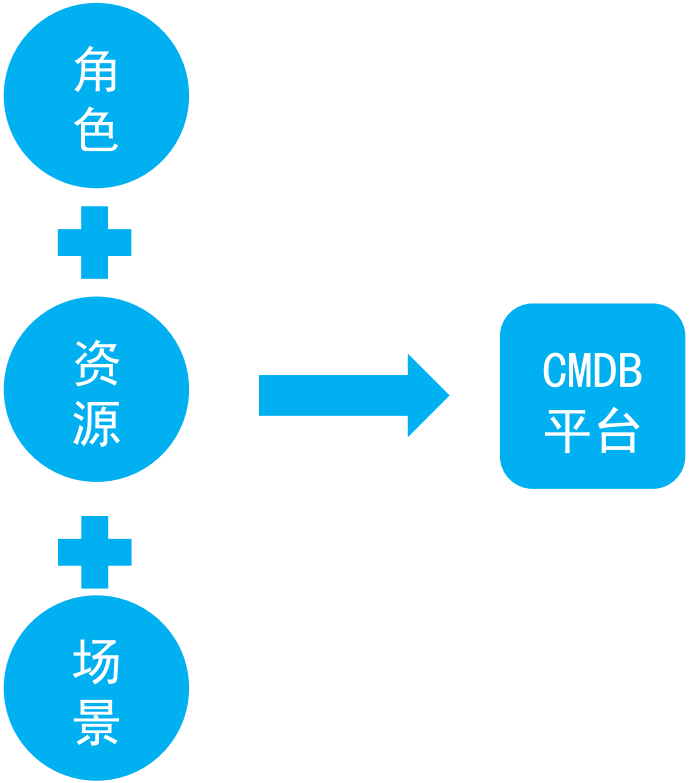
表现	描述
CMDB是元数据中心	CMDB是元数据中心，但不知道如何理解这个元数据？
缺少应用的维度关联CMDB	仅仅是为了管理配置项，管理资源，而不知道这些资源的应用者是谁。
缺少更多场景的支撑和驱动	CMDB数据的维护绝不是孤立维护的过程，而是要加入场景化的运维需要，驱动其不断维护的过程。这种维护分直接和间接的两种。
严格控制外围系统的数据自己建设	监控系统、事件系统、分析系统严格禁止其独立维护数据的过程，从而导致CMDB逐渐边缘化。
人工维护意味着规则隐藏	人工引入维护的数据越多，就以为这个规则越隐藏，未来被修改的可能性就越低。

CMDB的内涵需要扩展



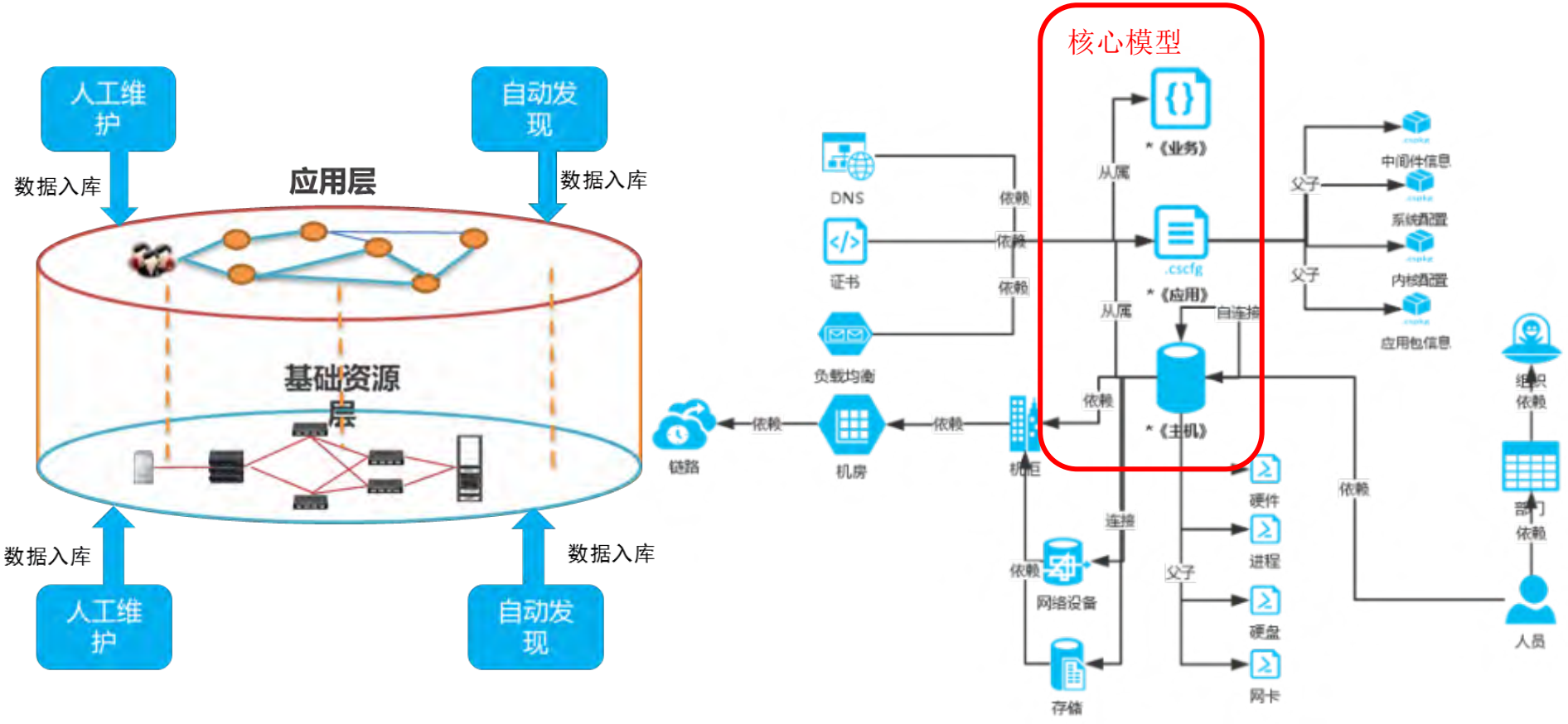
CMDB，从配置到IT资源的改变（实现篇）

CMDB构建的核心逻辑

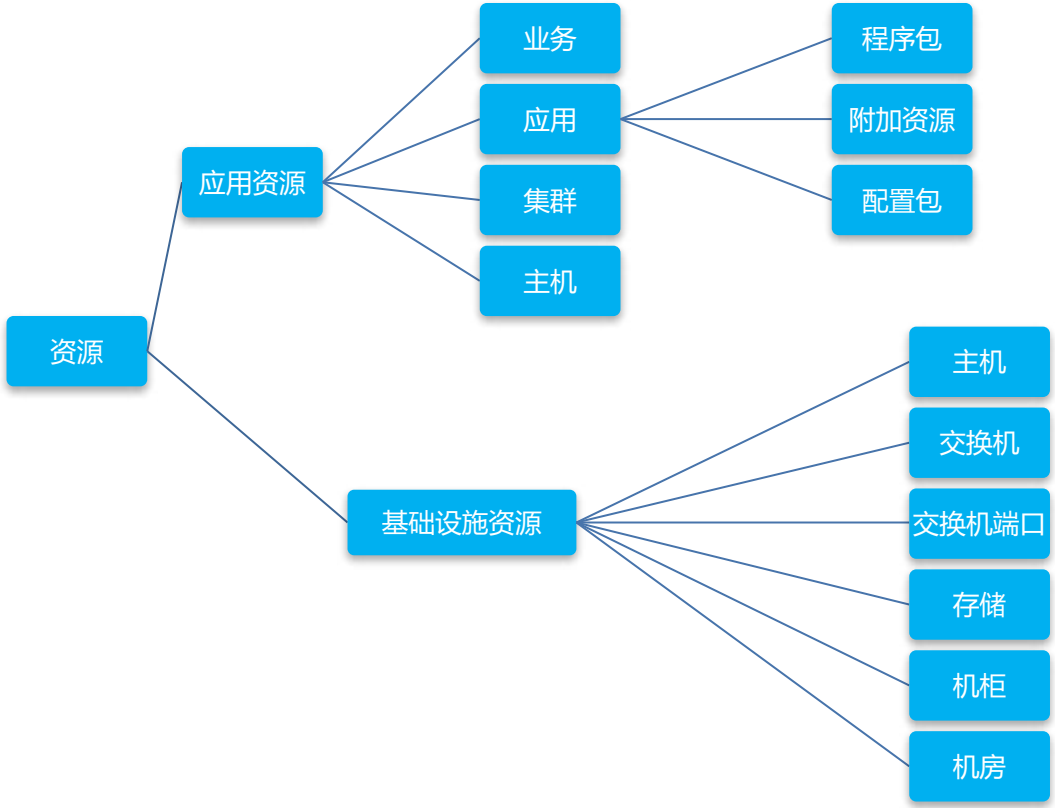


构成组件	描述
资源	在你的组织中到底存在多少的资源对象需要管理。就是Ci对象。
角色	这些资源对象到底是被谁管理的。对象的Owner是谁
场景	这些管理员平时的日常活动是什么。比如说流程，关联平台的场景

CMDB的两层逻辑架构及模型设计

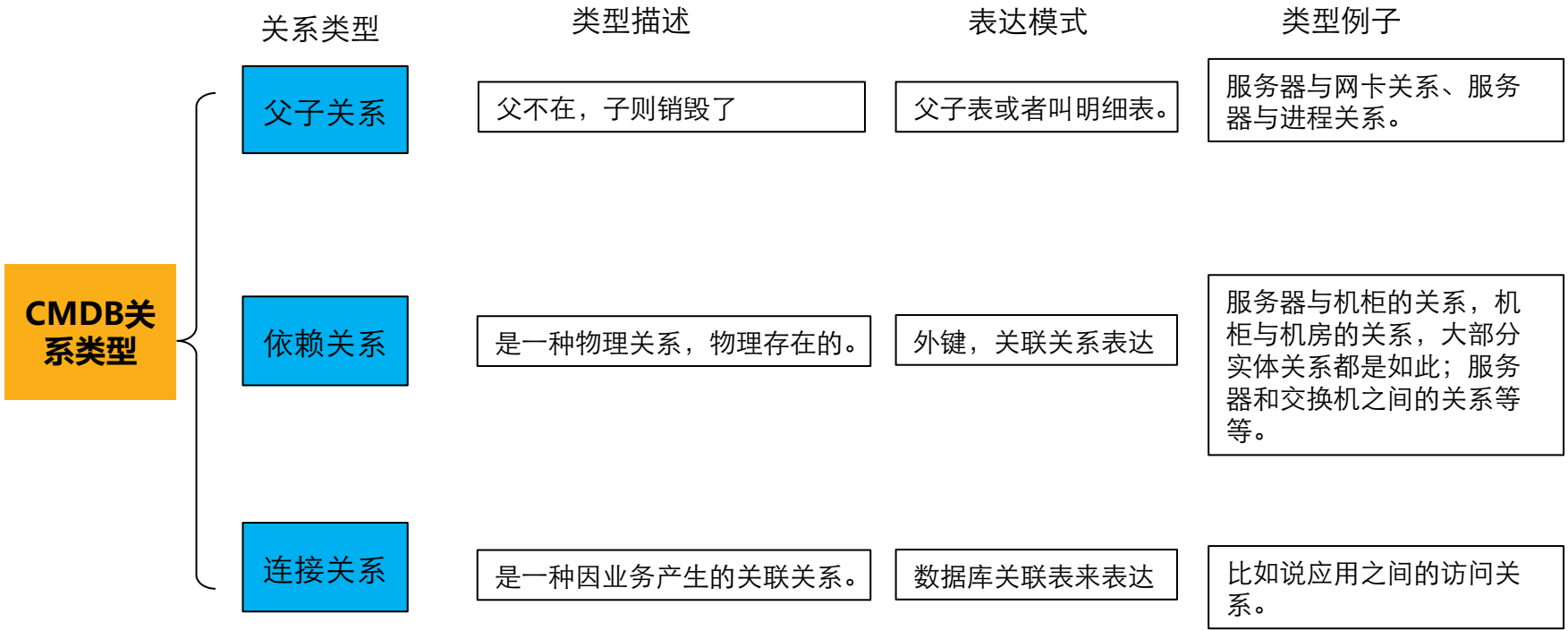


CMDB对象识别方法（层次分析法）

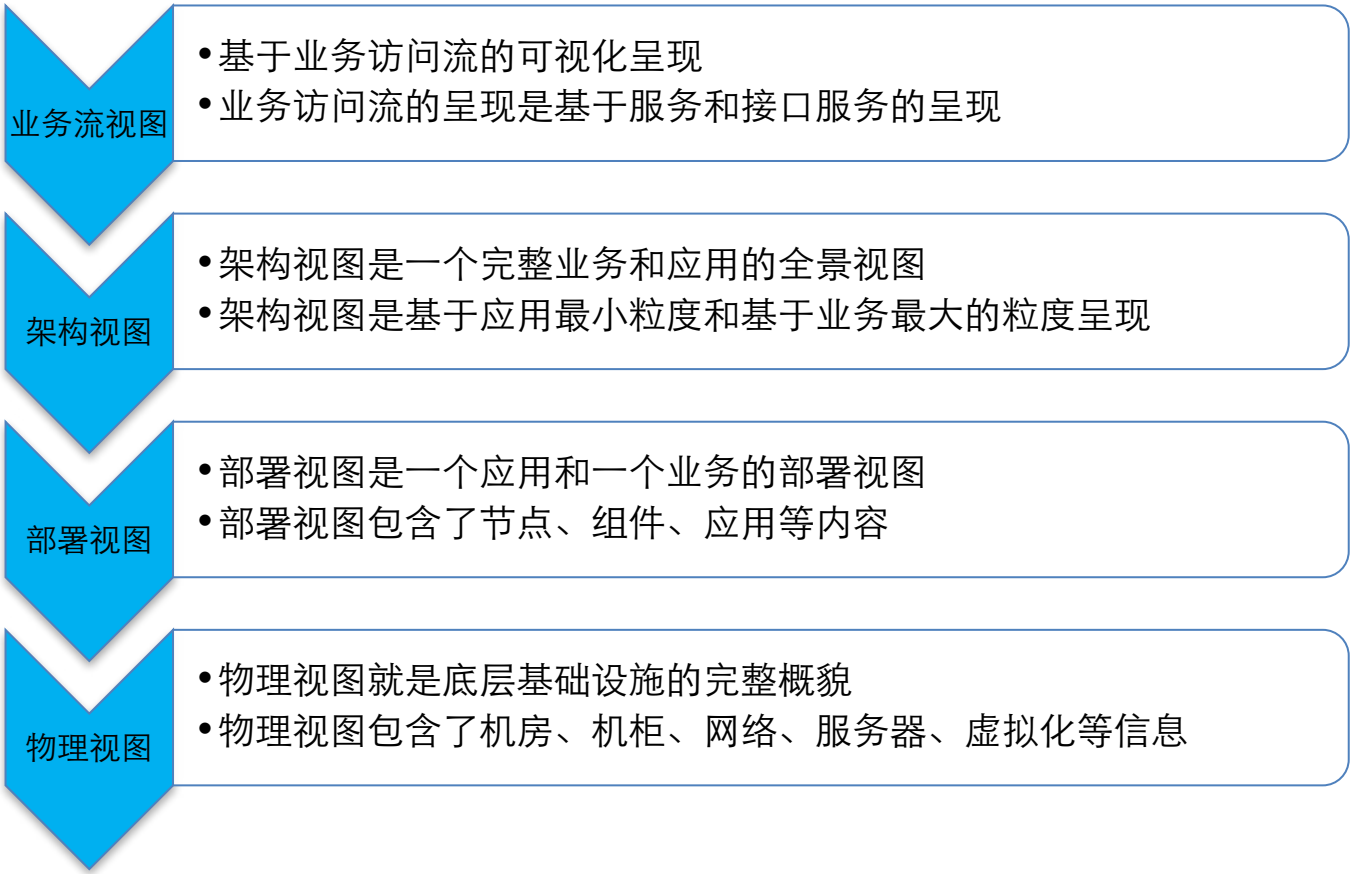


- 资源一分为二：基础设施资源和应用资源。
- 根据各自的资源域不断的向下划分，建立完整的资源管理谱系，这是系统分析的方法。

CMDB关系的表达


关系类型
类型描述
表达模式
类型例子
CMDB关系类型
父子关系
父不在，子则销毁了
父子表或者叫明细表。
服务器与网卡关系、服务器与进程关系。
依赖关系
是一种物理关系，物理存在的。
外键，关联关系表达
服务器与机柜的关系，机柜与机房的关系，大部分实体关系都是如此；服务器和交换机之间的关系等等。
连接关系
是一种因业务产生的关联关系。
数据库关联表来表达
比如说应用之间的访问关系。

CMDB关系的业务应用（四种视图）

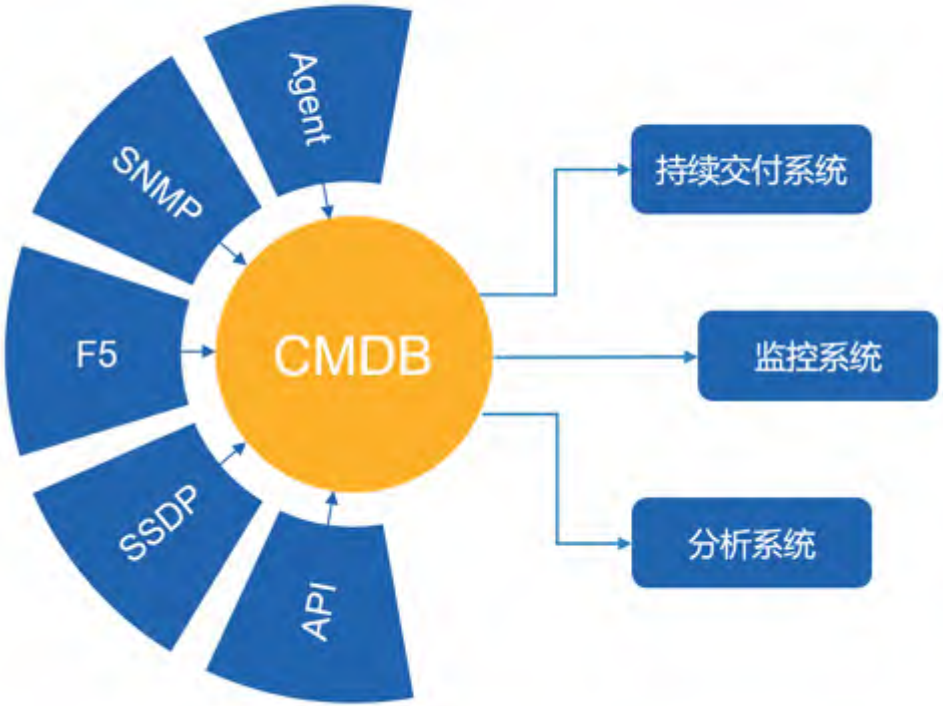


CMDB，从配置到IT资源的改变（自动化和人工流程的平衡）

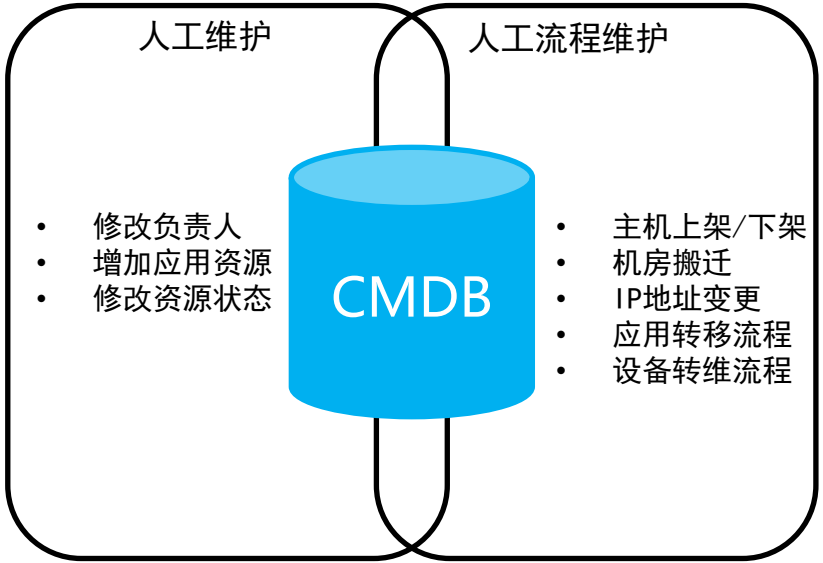
CMDB的自动发现

资源系统管理的难点——信息“准确性”的维护

- 服务器自动发现
  - Agent采集资源信息自动录入
  - 多协议支持
  - 外部API服务的自发现支持
  - 外部API服务的自发现支持
- 信息自动更新
  - 资源信息驱动系统变更
  - 变更后信息实时反馈更新
- 个性化信息采集
  - 支持采集插件的扩展
  - Agent实现统一插件调度

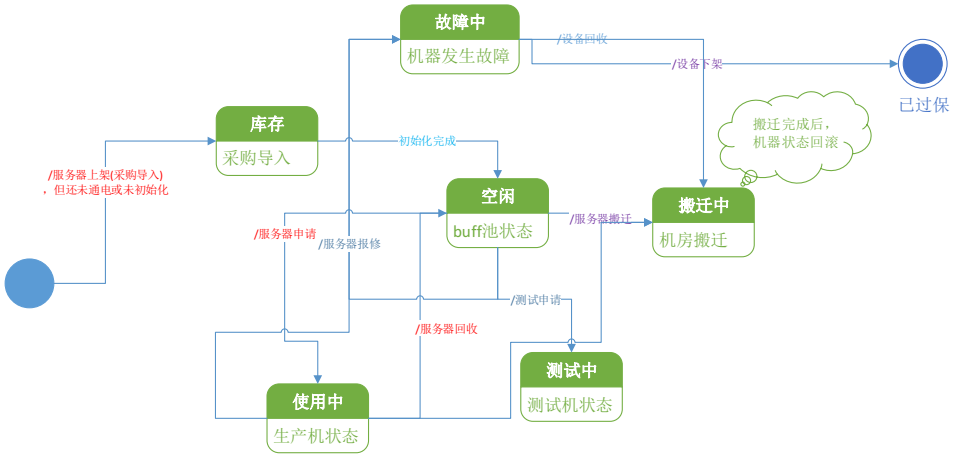


CMDB为什么还需要流程



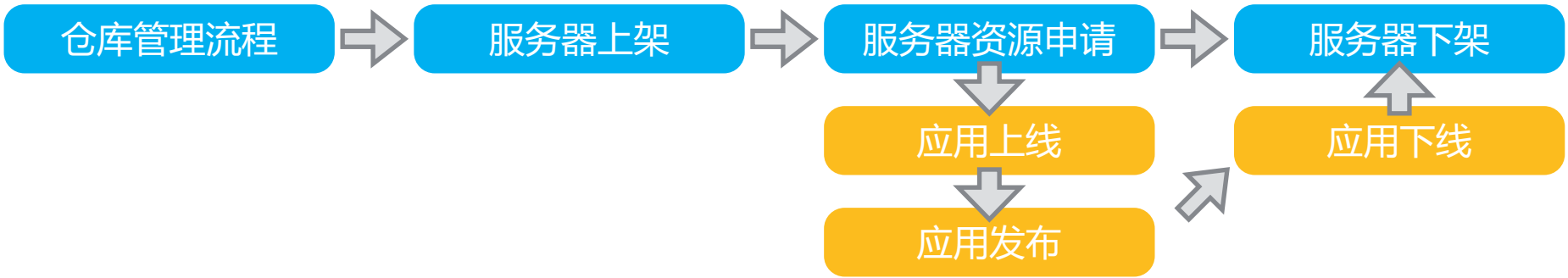
- CMDB流程是多角色协同控制的有效手段，流程一定是梳理某个场景下的多角色的职责关系。
- 资源的状态变化是需要人工触发或者流程触发
- 资源的状态的原子性保证需要通过流程来解决并发冲突问题
- 资源的状态变化是需要流程来控制过程（空闲-》申请中-》使用中）

资源对象生命周期状态图



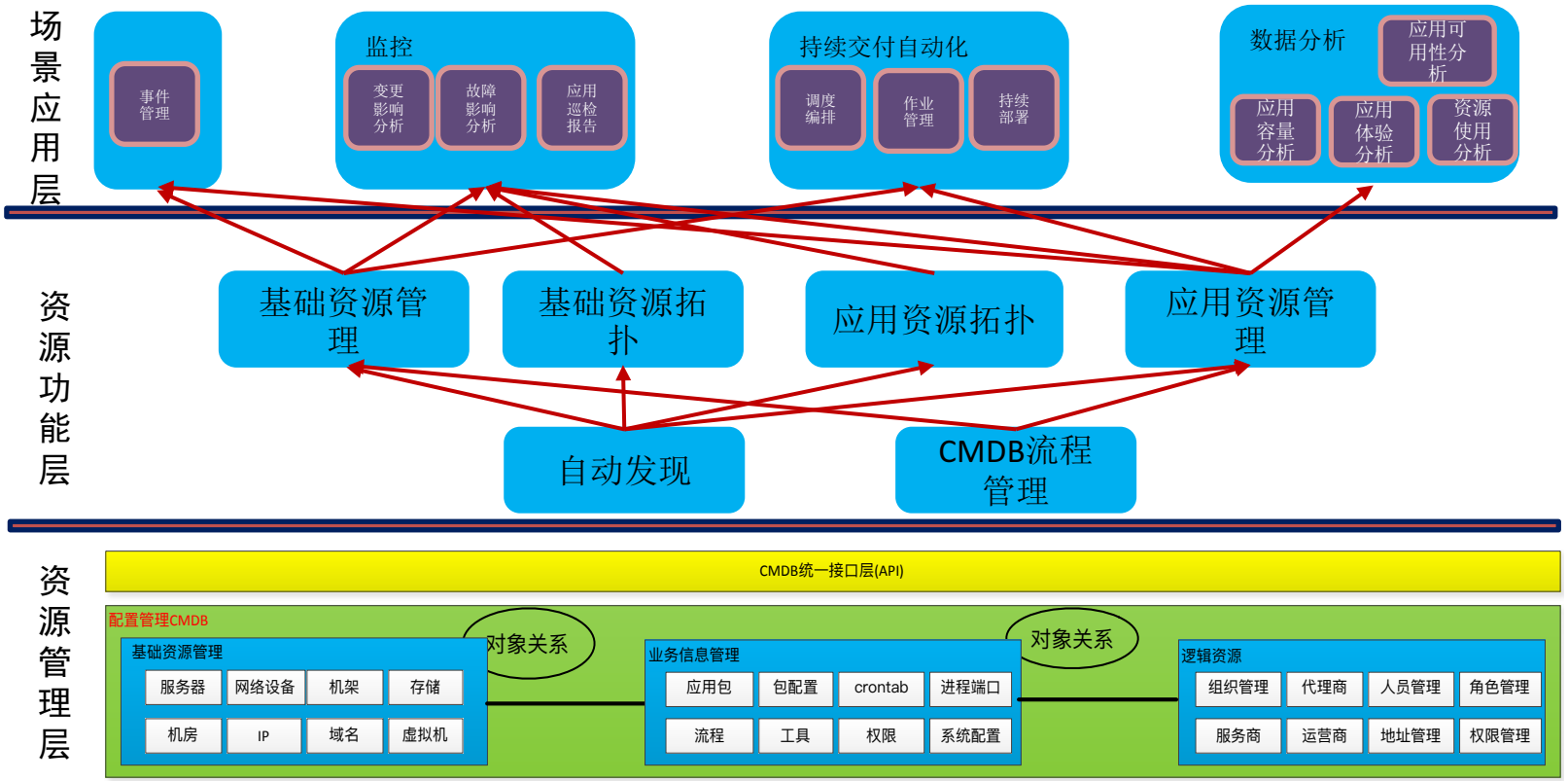
服务器状态转换图

- 物理状态
- 业务状态



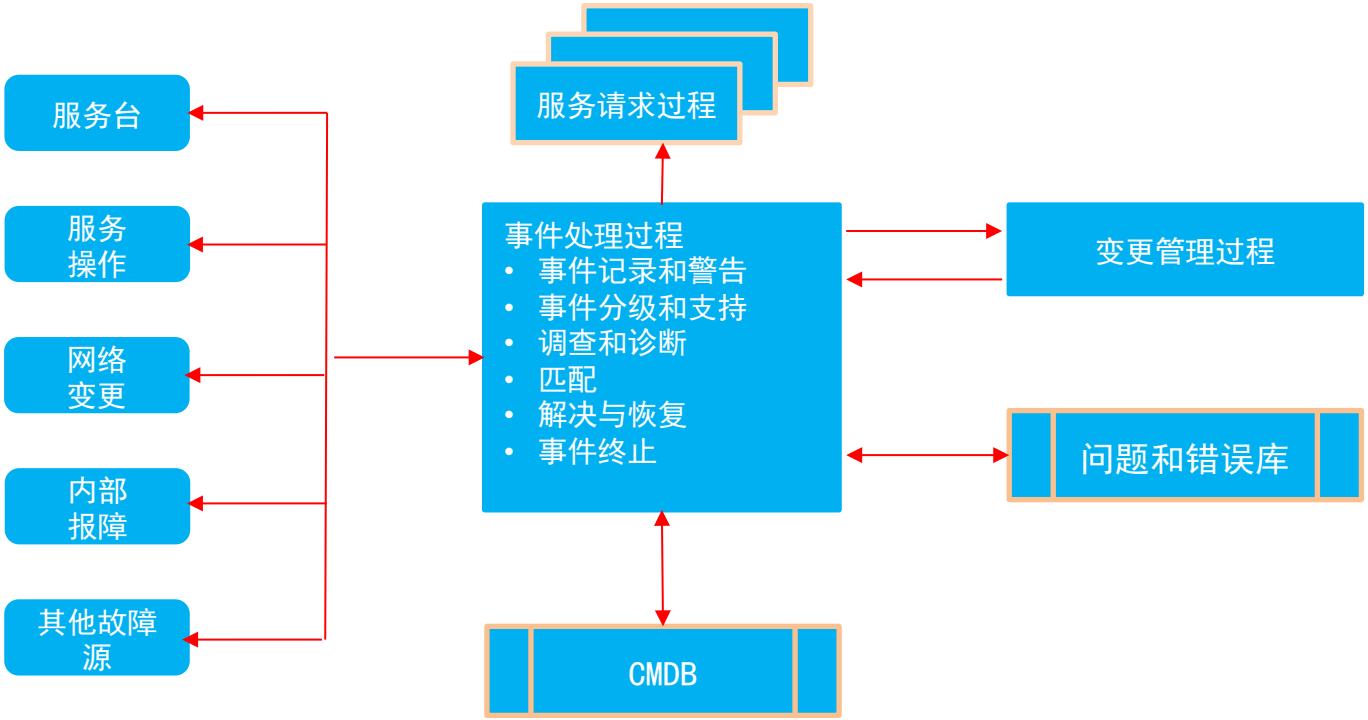
CMDB，从配置到IT资源的改变（场景化应用）

CMDB系统架构层次图



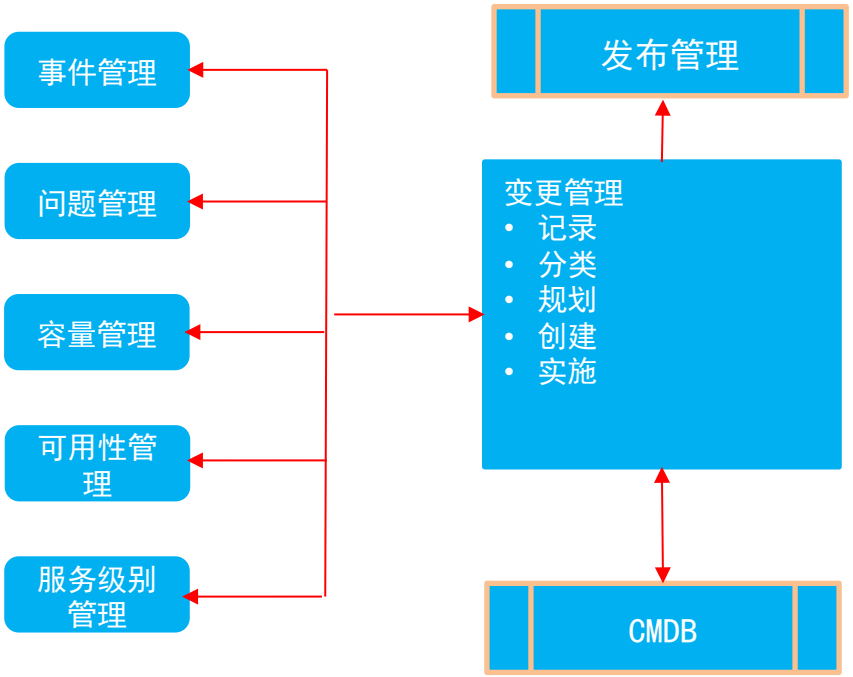
- 确定CMDB的管理层次，分期建设；领导参与；核心能力覆盖。注：不包含拓扑管理的功能

CMDB与事件管理



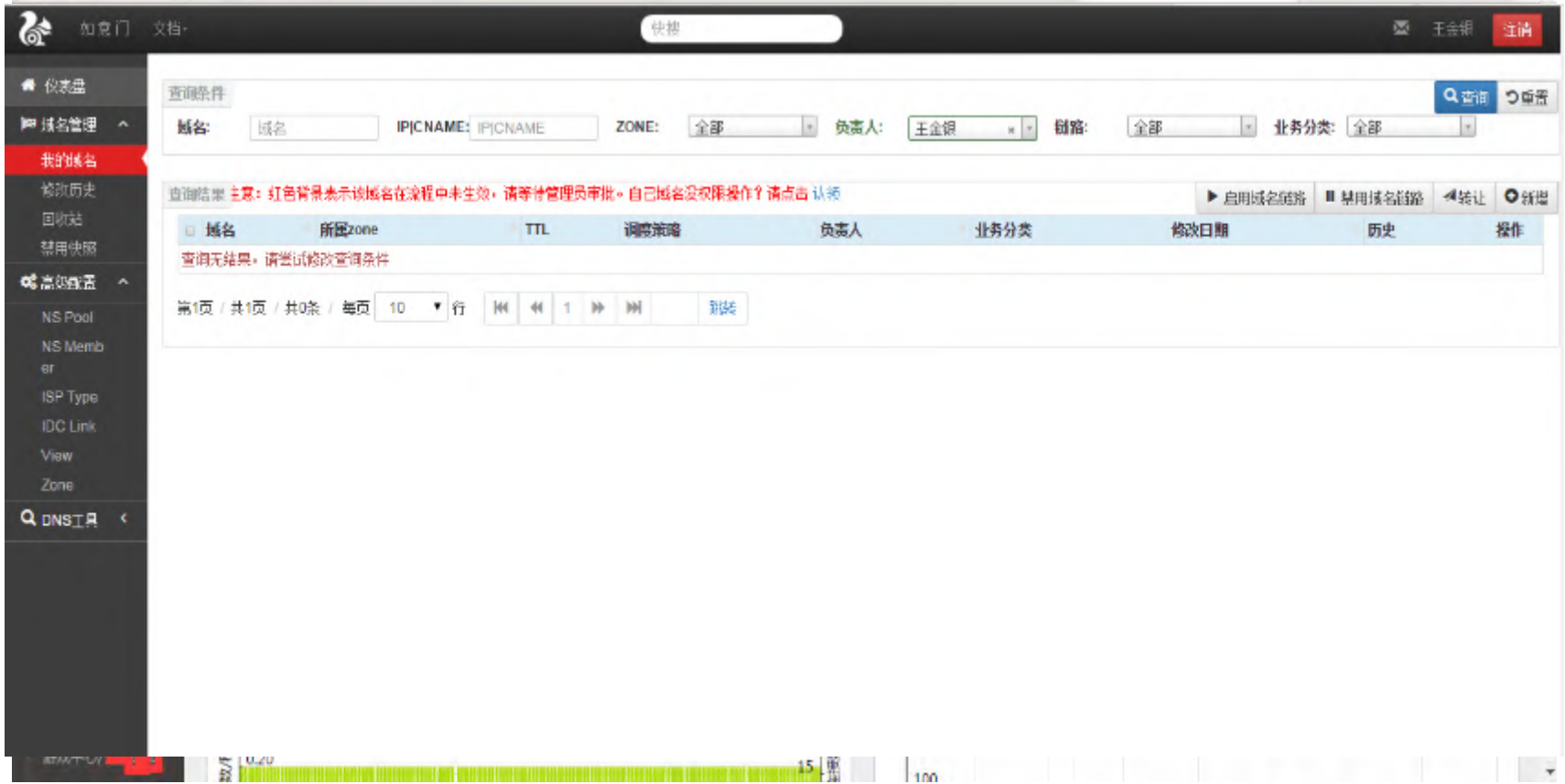
- 事件收敛
- 事件和资源对象状态关联
- 事件推送
- 事件升级

CMDB与变更管理



- 变更对象的资源管理
- 变更对象的状态管理
- 变更对象的权限管理
- 变更的影响评估

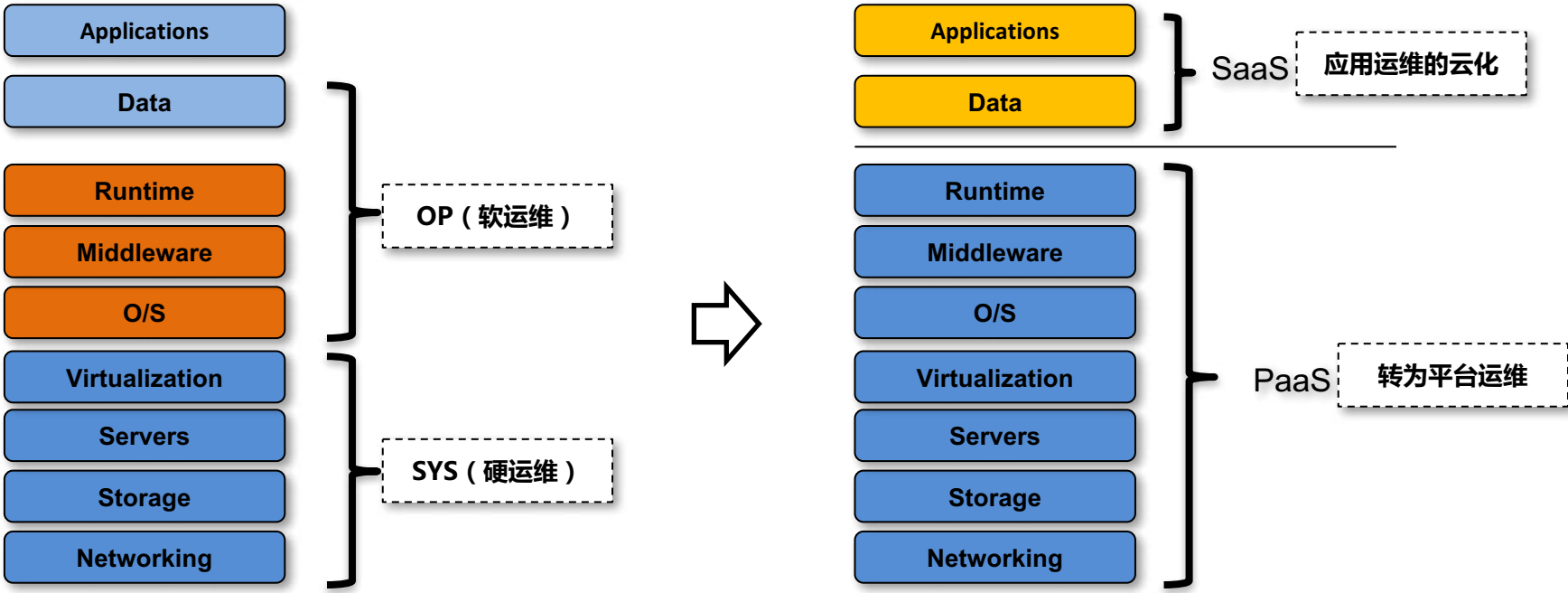
CMDB与周边平台集成



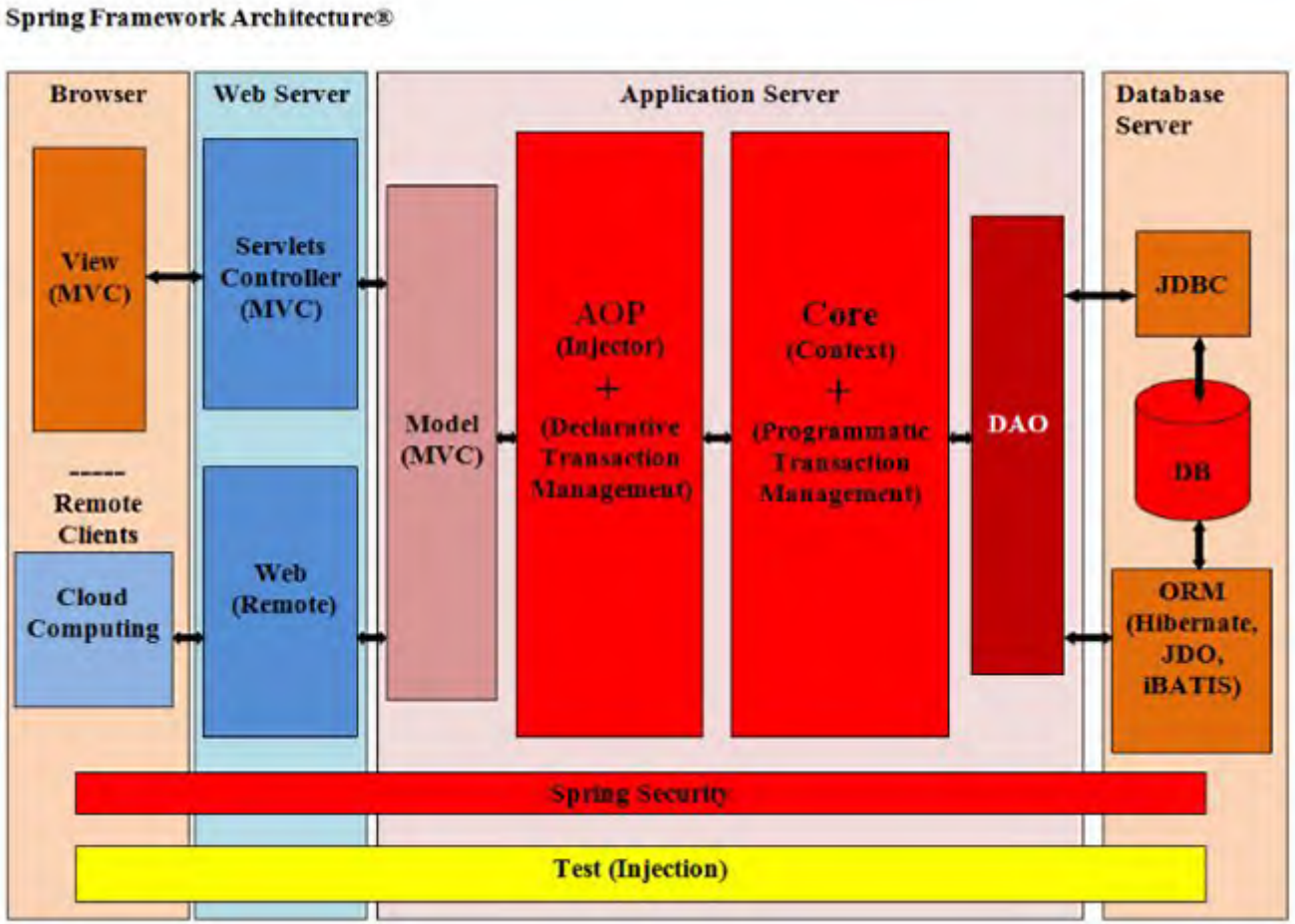
主张一：CMDB改名成IT资源管理
主张二：从应用的角度管理资源

BigOps，从基础设施到应用的跨越

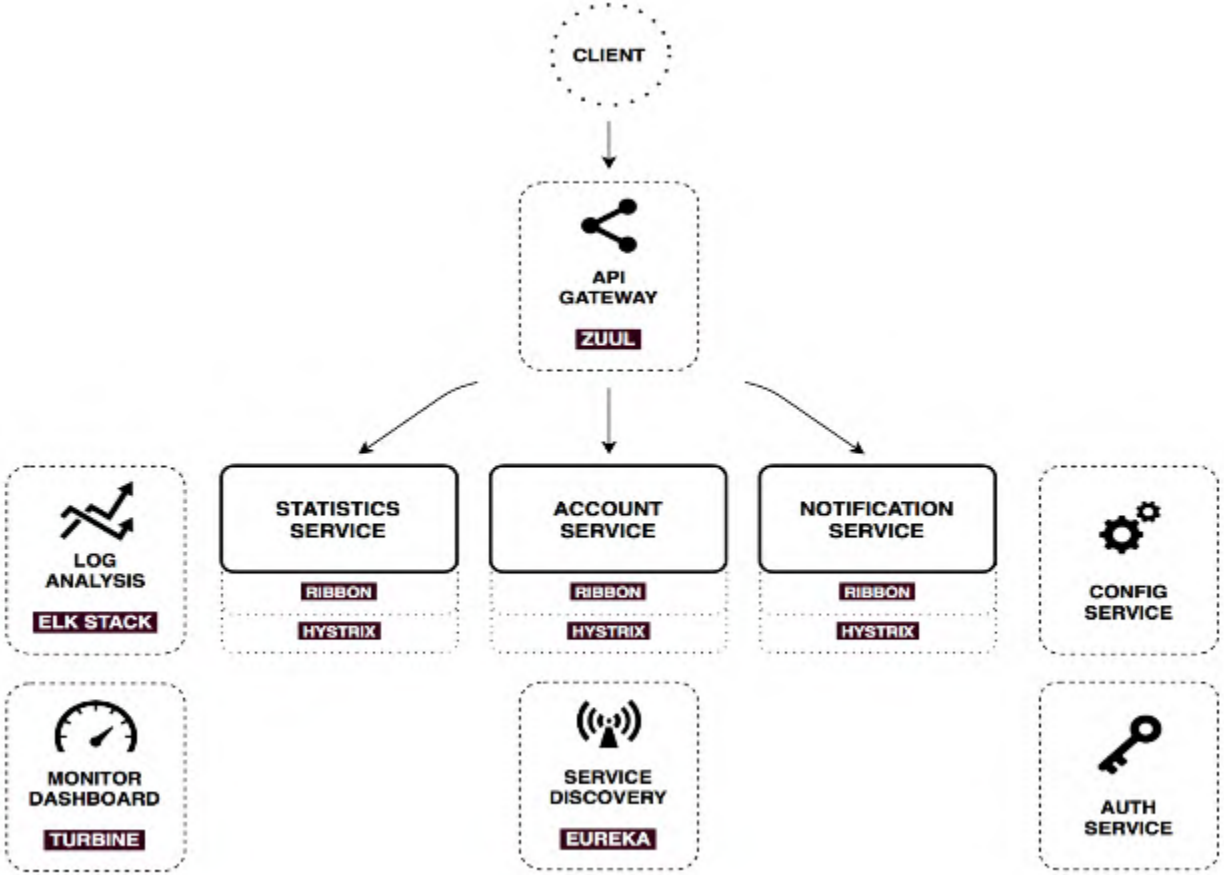
如何看待运维的未来


Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
OP（软运维）
SYS（硬运维）
Applications
Data
SaaS
应用运维的云化
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
PaaS
转为平台运维

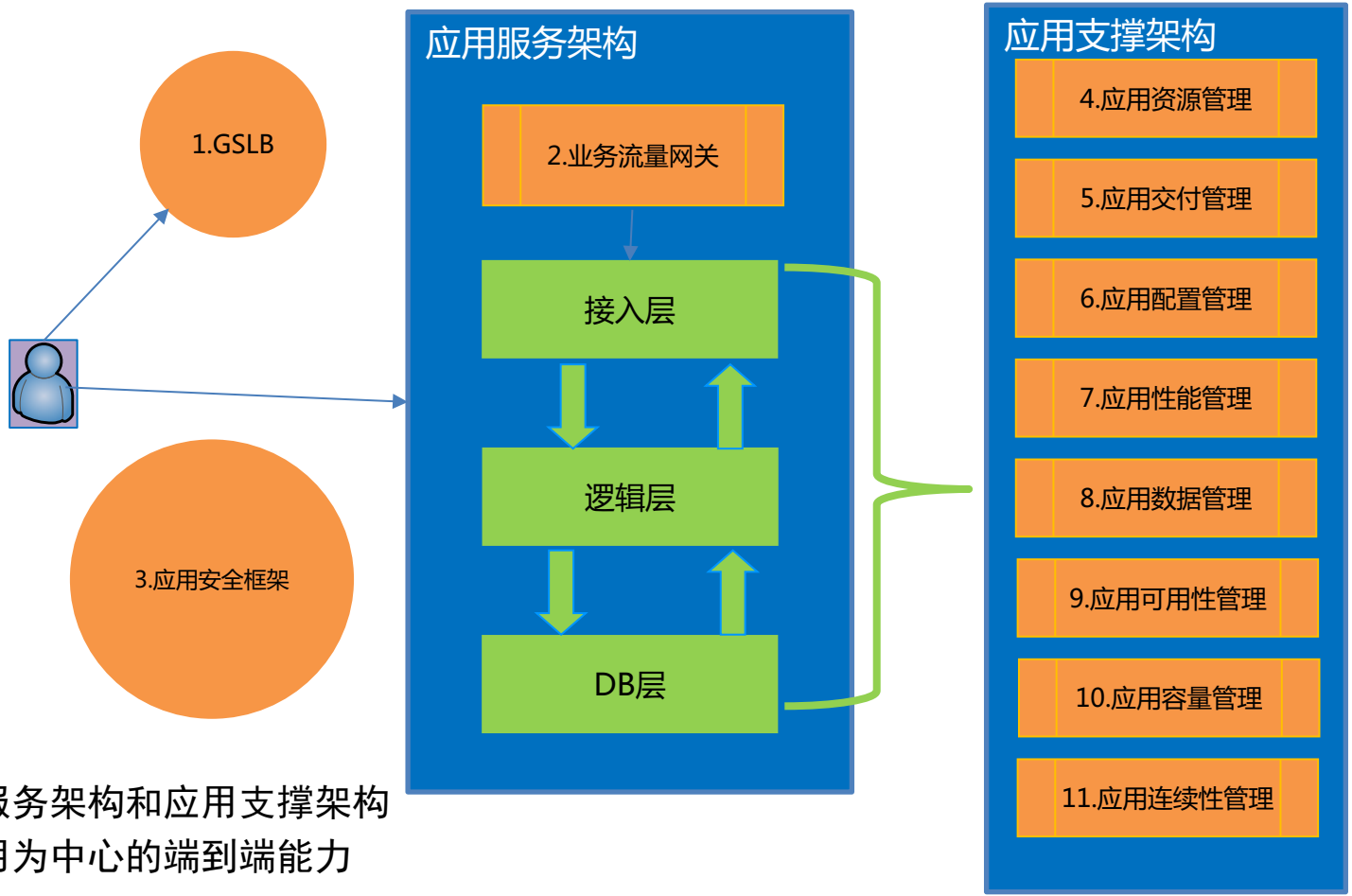
面向应用的传统架构—SPRING MVC



面向应用层的云架构—SRING CLOUD

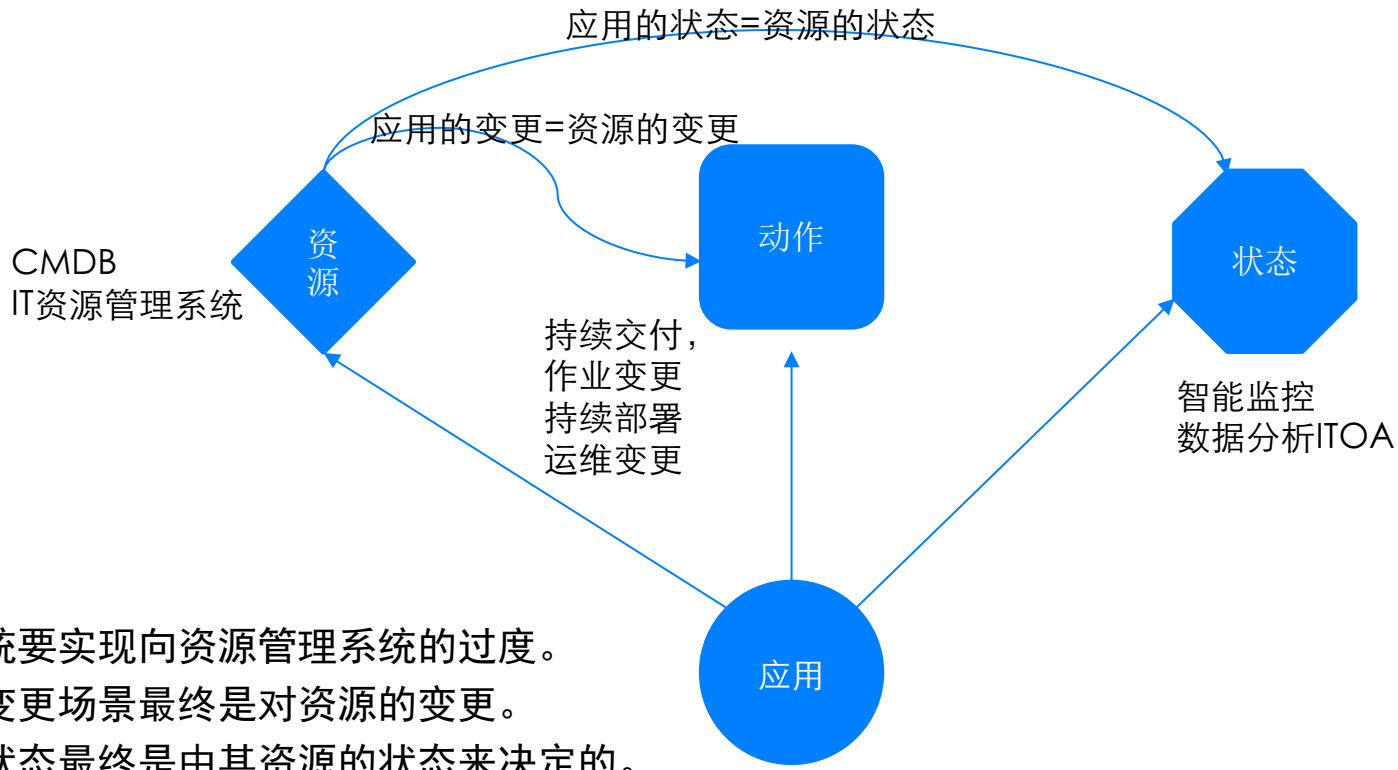


面向应用的运维管理——BIGOPS



- 应用服务架构和应用支撑架构
- 以应用为中心的端到端能力

基于应用的核心管理框架



- CMDB系统要实现向资源管理系统的过度。
- 应用的变更场景最终是对资源的变更。
- 应用的状态最终是由其资源的状态来决定的。

应用的资源管理

- 应用的环境是资源的一种，如开发、测试、生产环境等等
- 应用的环境管理包括环境的基本管理、环境的参数管理、环境的状态管理。
- 应用的环境管理一致性是核心
- 环境是有生命周期状态的
- 不可变基础设施是环境管理一致性终极手段

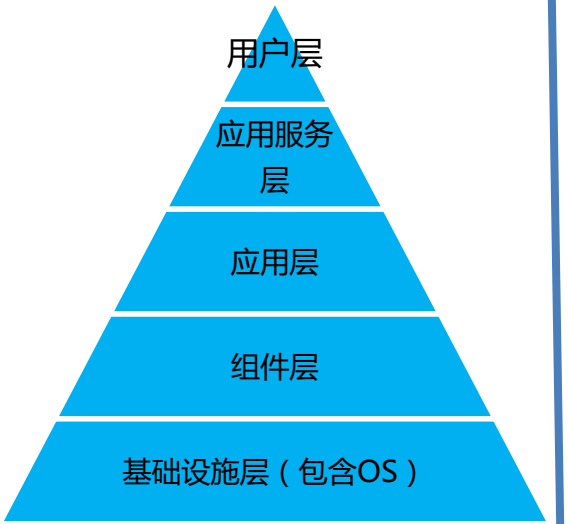
应用的动作管理

- 作业管理，一方面把运维的脚本能力可视化，另一方面也在提高运维的效率和质量

- 工具平台化，运维能力沉淀，并可以无缝转移

应用的状态管理

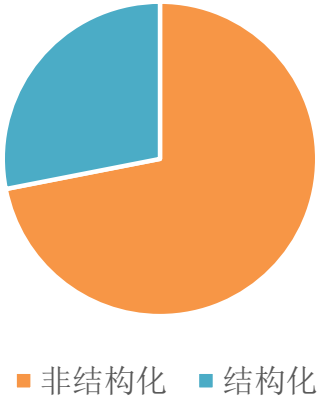
数据体系



数据来源

- RDS
- 日志
- 服务接口
- API
- NoSql

数据种类



处理手段

流式处理

多维分析

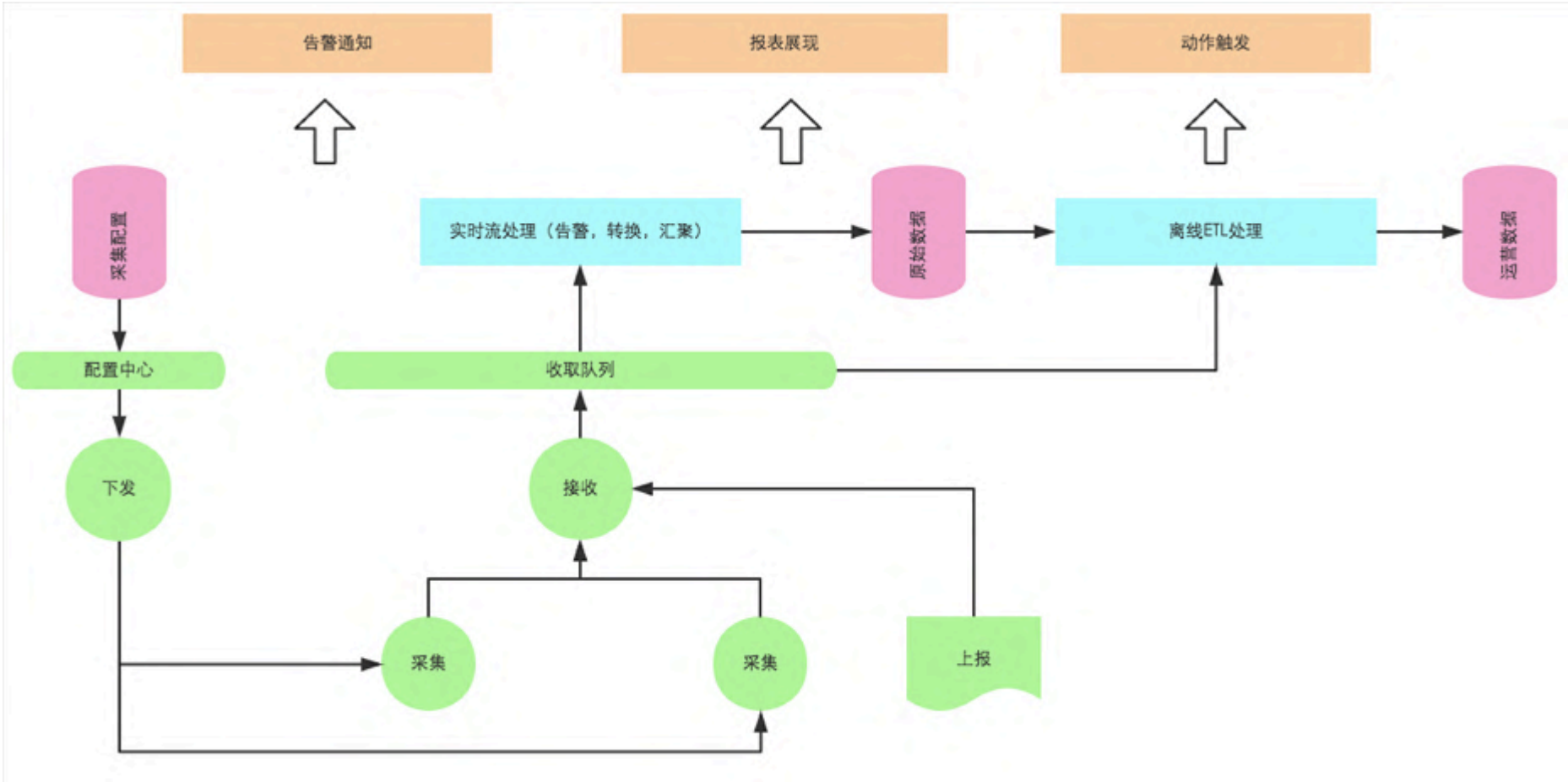
常规处理

数据应用

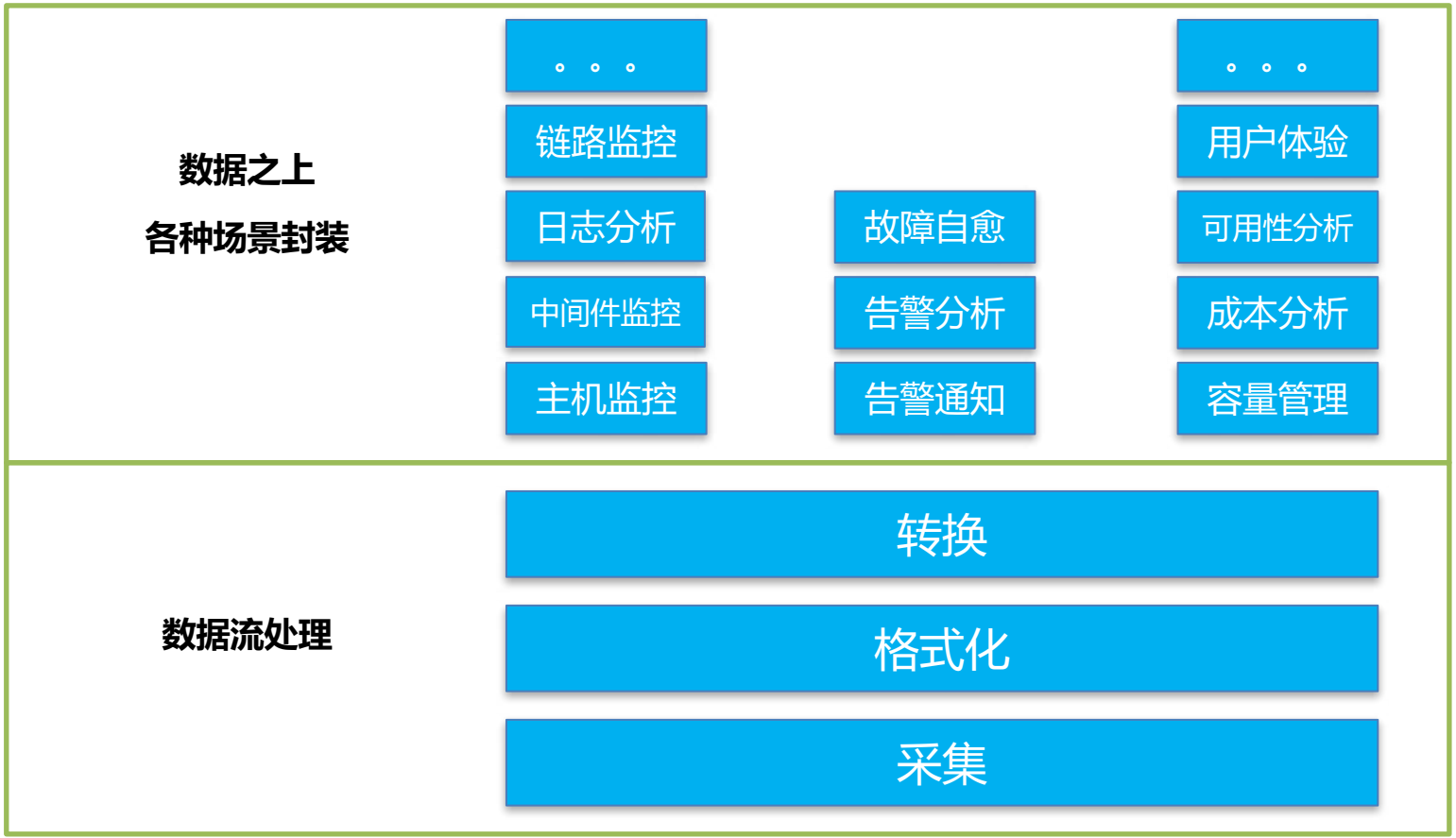
监控告警

运营分析

数据的统一处理架构


告警通知
报表展现
动作触发
采集配置
实时流处理（告警，转换，汇聚）
原始数据
离线ETL处理
运营数据
配置中心
收取队列
下发
接收
采集
采集
上报

数据的场景化应用


数据之上
各种场景封装
。。。
链路监控
日志分析
中间件监控
主机监控
故障自愈
告警分析
告警通知
。。。
用户体验
可用性分析
成本分析
容量管理
数据流处理
转换
格式化
采集

日志分析

日志分析引擎

关键字解析

Nginx日志

Mysql慢查询

日志模板引擎

告警黄金指标

- HTTP返回500占比
- 接口逻辑错误占比

- 程序接口调用延时

- HTTP每秒请求量
- 网络流量
- 网卡包量

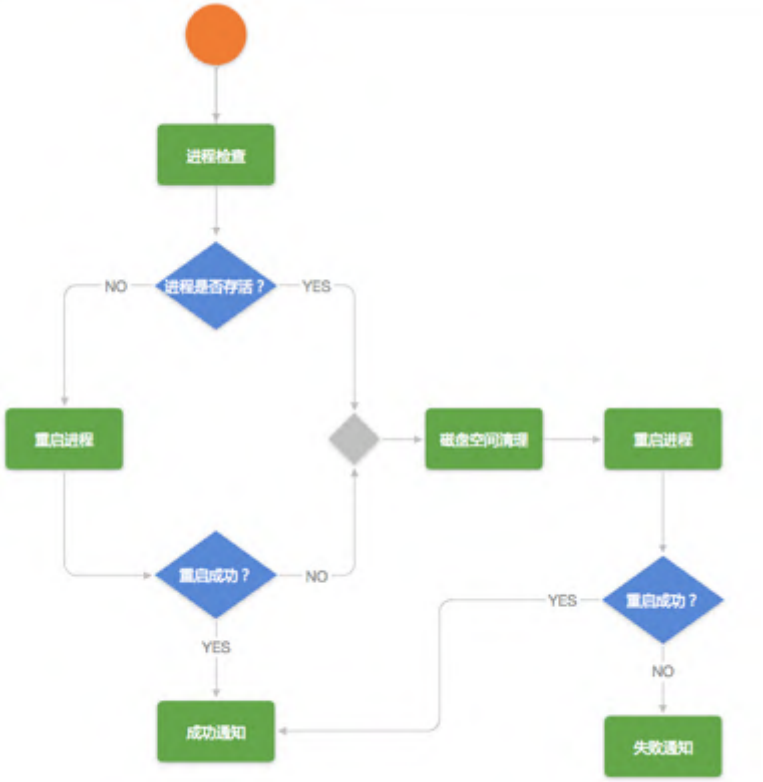
- CPU使用率
- IO使用率

数据的更高级应用

基于深度学习的人工智能异常探测

- 算法适应力强
- 预测准确
- 自动学习，无需给每个时间序列单独配置参数

基于事件的统一处理中心



- 经验沉淀后的解决方案模板
- 自定义处理流程
- 告警事件、容量事件、变更事件

DevOps精益工程实践——持续交付

运维的本质之一——交付

面向资源层的自动化能力，包括IaaS和PaaS层自动化

面向应用的自动化能力，包括持续交付的自动化和运营维护过程自动化

什么是持续交付

Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, safely and quickly in a sustainable way.

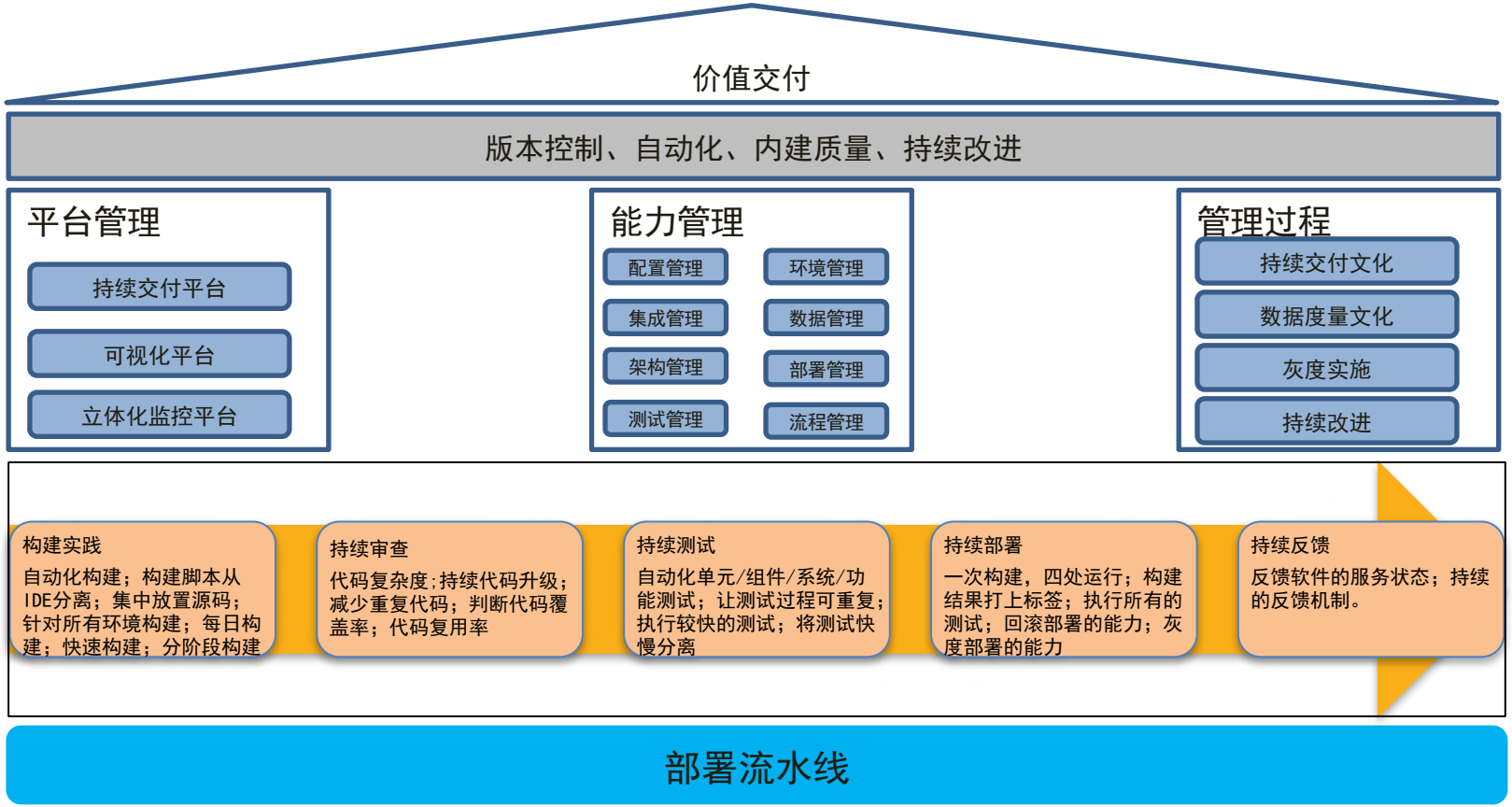
--Jez Humble

- 持续交付是DevOps的最佳工程实践
- 持续交付是精益企业的最佳工程实践

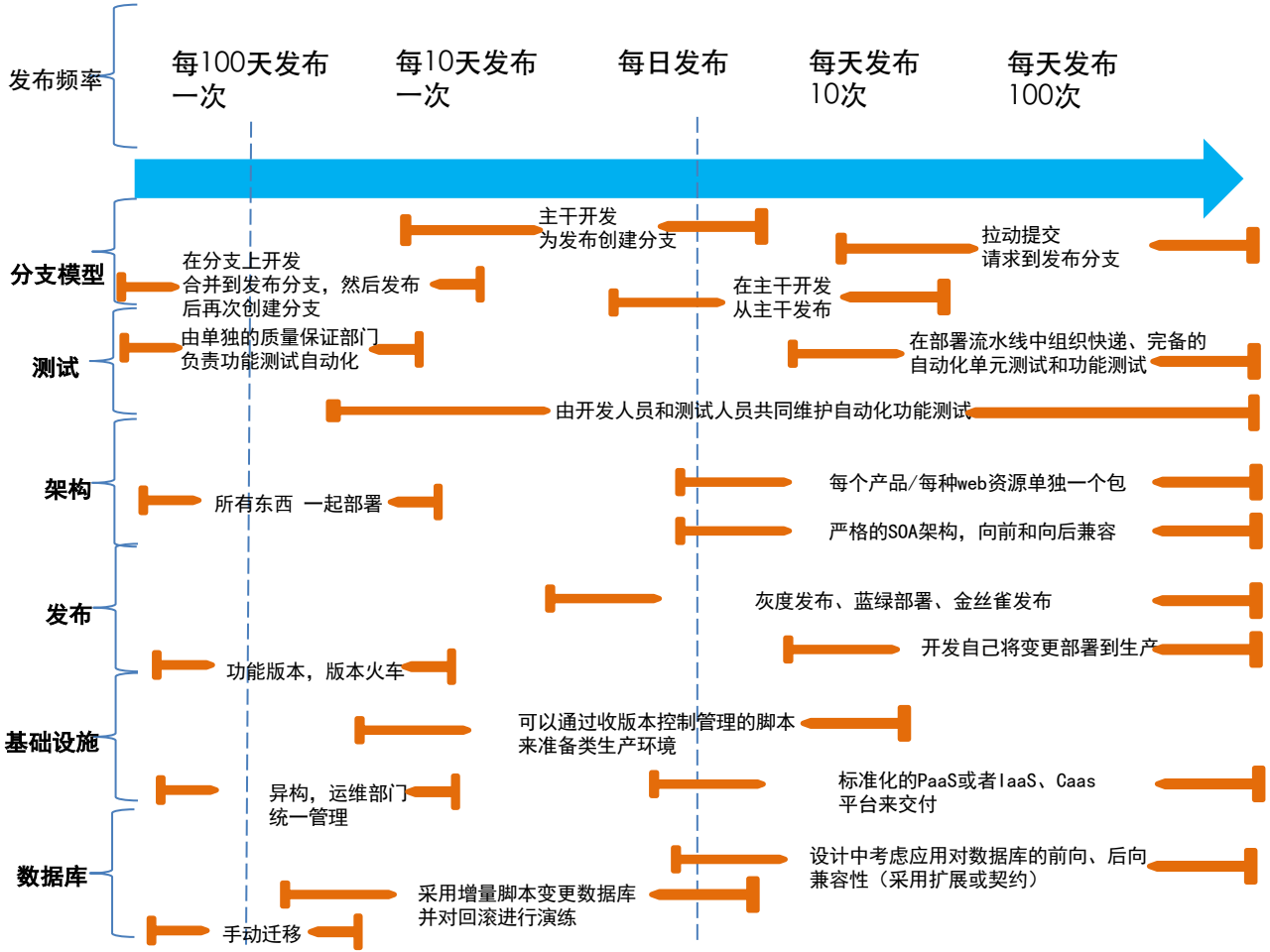
持续交付原则

- 为软件的发布创建一个可重复且可靠的过程
- 将几乎所有事情自动化
- 把所有的东西都纳入版本控制
- 提前并频繁地做让你感到痛苦的事情
- 内建质量
- “DONE”意味着“已发布”
- 交付过程是每个成员的责任
- 持续改进

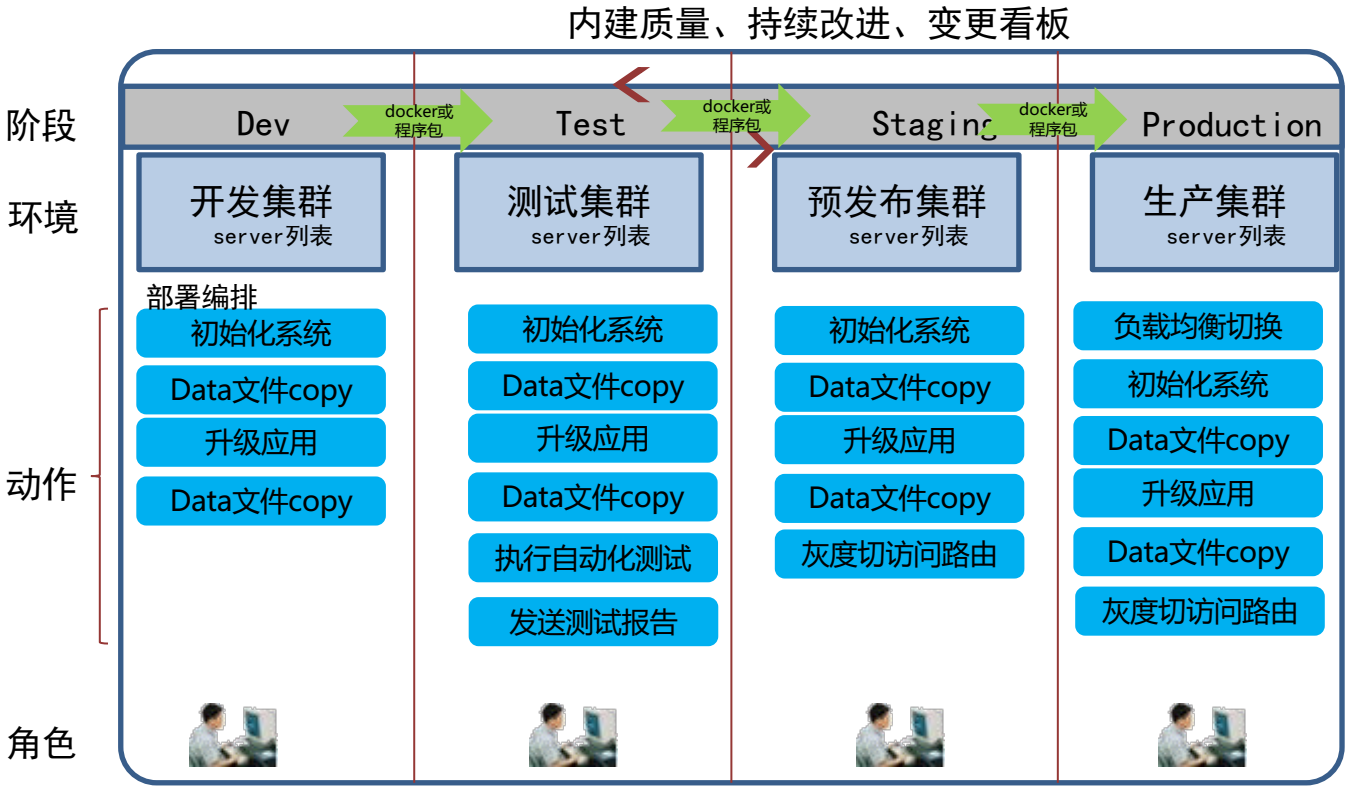
持续交付屋—交付阶段


价值交付
版本控制、自动化、内建质量、持续改进
平台管理
持续交付平台
可视化平台
立体化监控平台
能力管理
配置管理
环境管理
集成管理
数据管理
架构管理
部署管理
测试管理
流程管理
管理过程
持续交付文化
数据度量文化
灰度实施
持续改进
构建实践
自动化构建；构建脚本从IDE分离；集中放置源码；针对所有环境构建；每日构建；快速构建；分阶段构建
持续审查
代码复杂度;持续代码升级；减少重复代码；判断代码覆盖率；代码复用率
持续测试
自动化单元/组件/系统/功能测试；让测试过程可重复；执行较快的测试；将测试快慢分离
持续部署
一次构建，四处运行；构建结果打上标签；执行所有的测试；回滚部署的能力；灰度部署的能力
持续反馈
反馈软件的服务状态；持续的反馈机制。
部署流水线

持续交付频率与能力对照表（优化版）



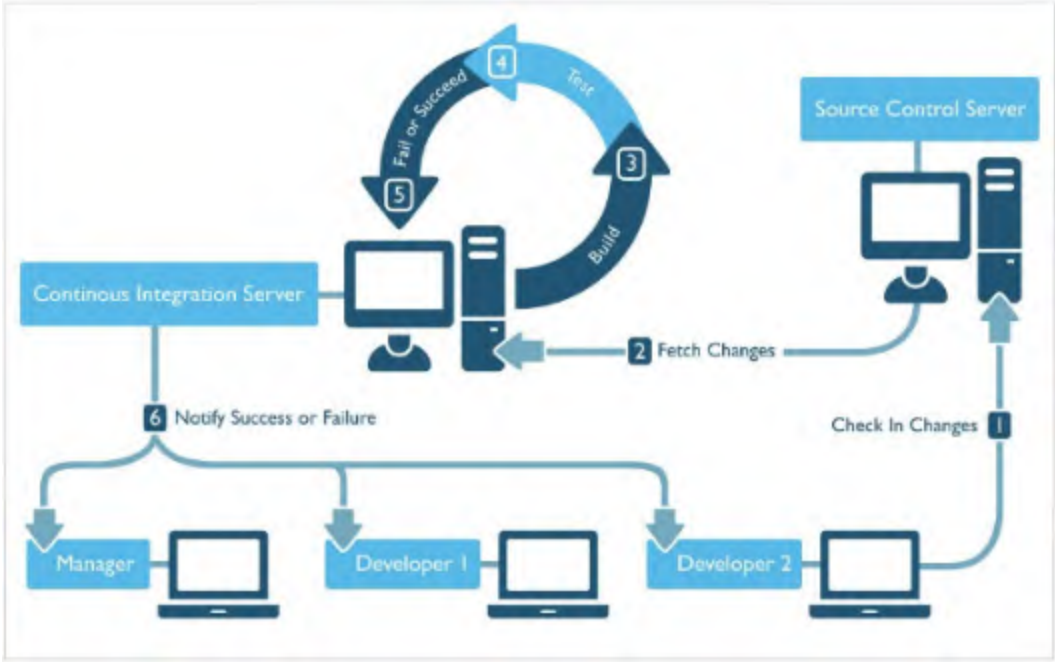
持续交付流水线的实现


内建质量、持续改进、变更看板
阶段
环境
动作
角色
Dev
docker或程序包
Test
docker或程序包
Staging
docker或程序包
Production
开发集群
server列表
测试集群
server列表
预发布集群
server列表
生产集群
server列表
部署编排
初始化系统
Data文件copy
升级应用
Data文件copy
初始化系统
Data文件copy
升级应用
Data文件copy
执行自动化测试
发送测试报告
初始化系统
Data文件copy
升级应用
Data文件copy
灰度切访问路由
负载均衡切换
初始化系统
Data文件copy
升级应用
Data文件copy
灰度切访问路由

持续集成与构建

不仅是技术，重点是原则的坚持

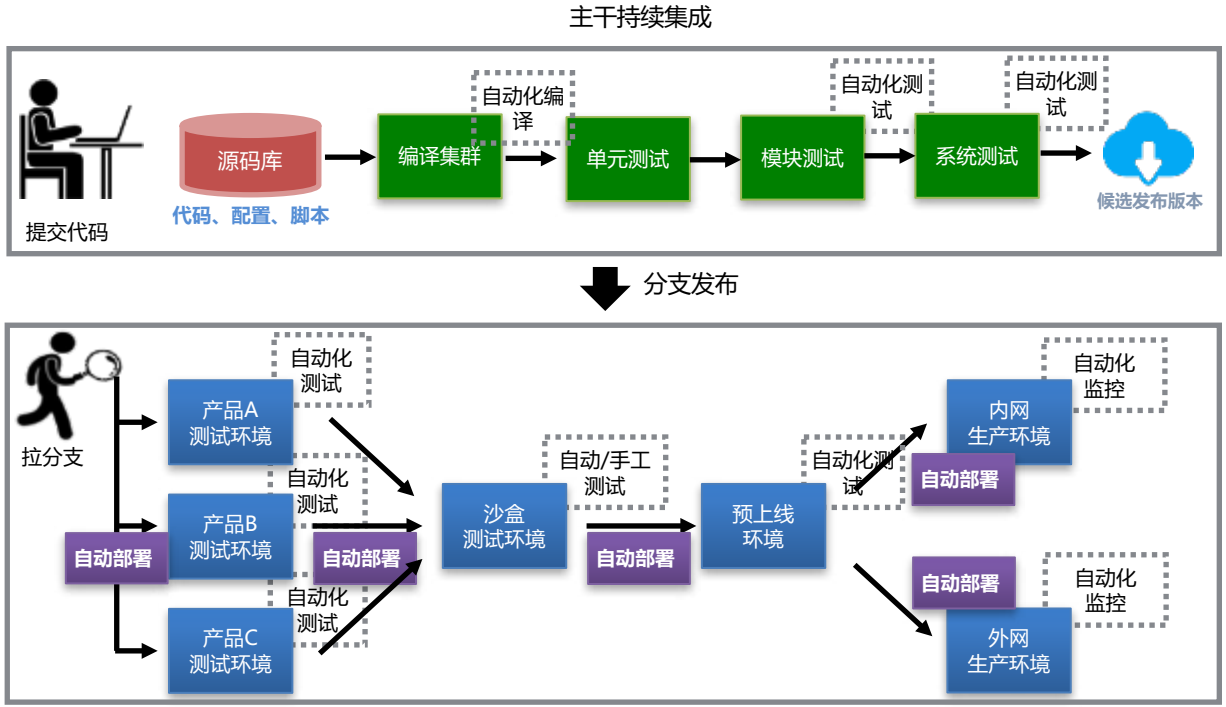
- 小的改动，逐步构建
- 每人、每天提交代码
- 在主干上做持续集成
- 至少要每天进行集成
- 使用好持续集成工具
- 自动化的构建和测试
- 分级测试并快速反馈
- 红灯需要立即被修复



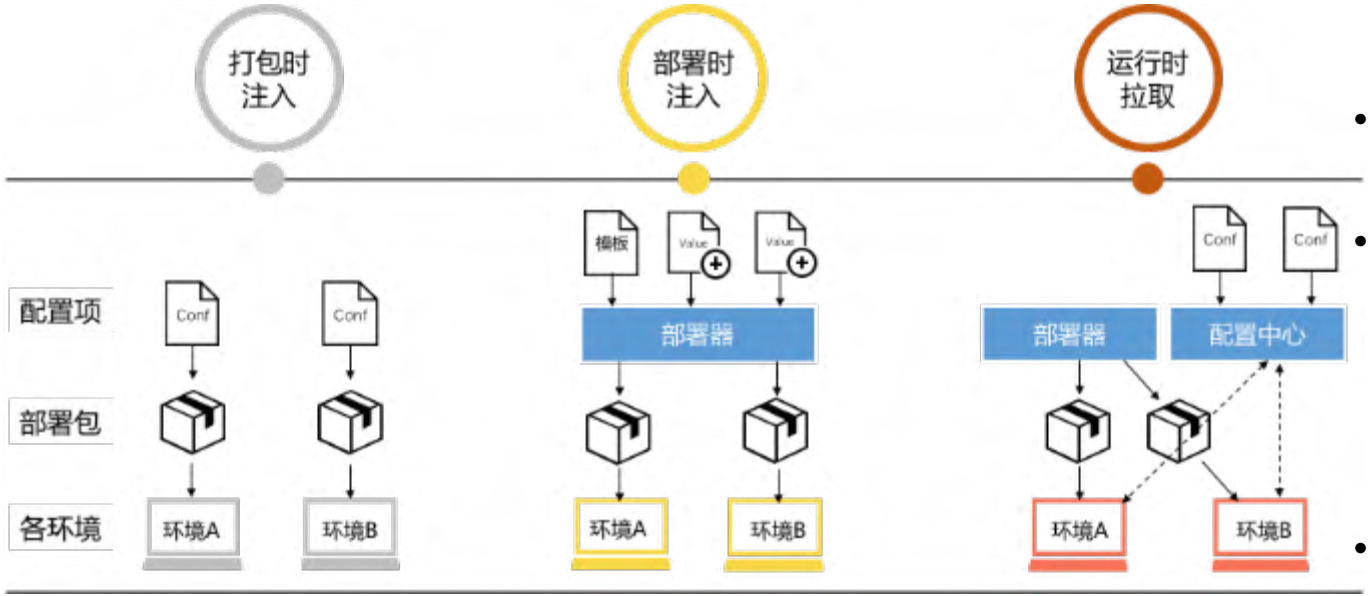
持续部署最佳实践

部署管理的最佳实践

- 部署包全部来自制品仓库
- 各环境使用相同部署方式
- 各环境使用相同部署脚本
- 部署流程编排，阶梯式晋级
- 运维人员参与部署过程创建
- 只有流水线才可变更生产环境，防止配置漂移
- 不可变（Immutable）服务器
- 热部署：蓝绿部署、金丝雀发布

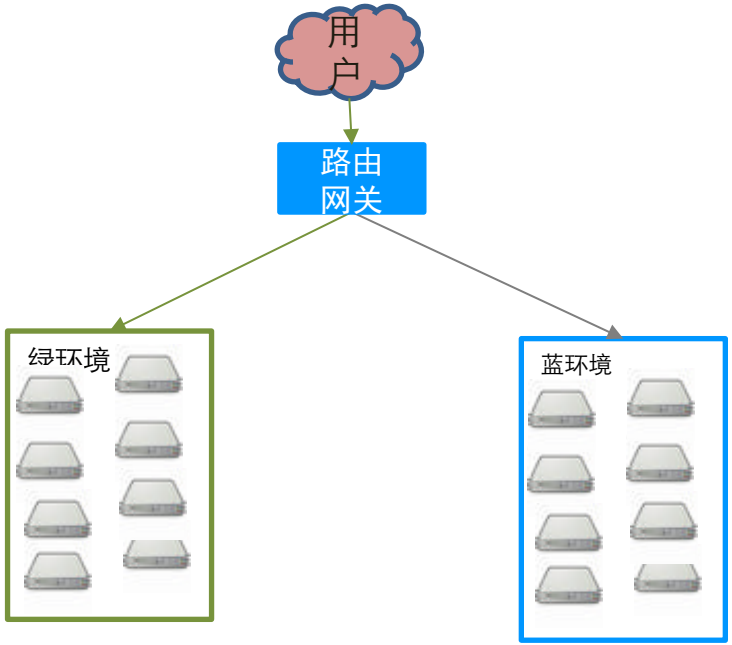


部署配置管理

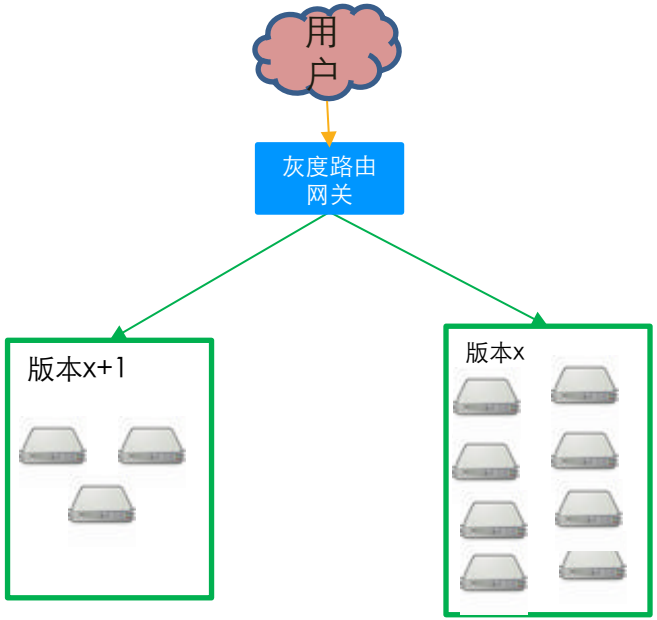


- 原始文件管理（File模式）
- 配置项管理（KV模式）
  - 环境变量
  - 数据库集中管理
  - 配置项文件统一管理
- 分布式配置中心管理

持续部署之蓝绿部署和金丝雀发布



- 蓝绿环境在切换的过程中，是通过路由网关的来控制的
- 蓝绿后端的数据也是分开两个环境的



- 同一批用户可能使用新旧版本
- 有利于回滚
- 逐步添加负载，增加新特性的使用

架构管理

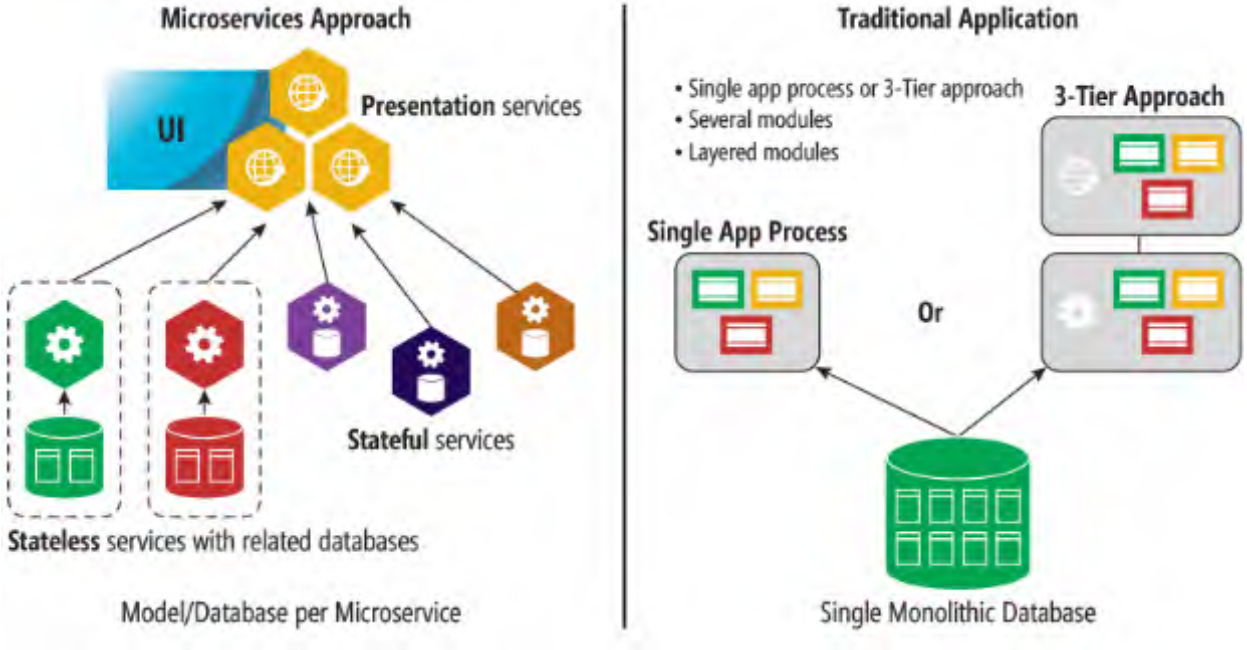
微服务开发模式强调功能特性端到端解耦，特性可以独立、并行地开发、集成、验证和发布，每个服务都有自己的生命周期。

微服务架构解决的问题

- 降低不同服务间开发相互影响
- 减少团队之间相互等待、集成和验证容易被阻塞的问题

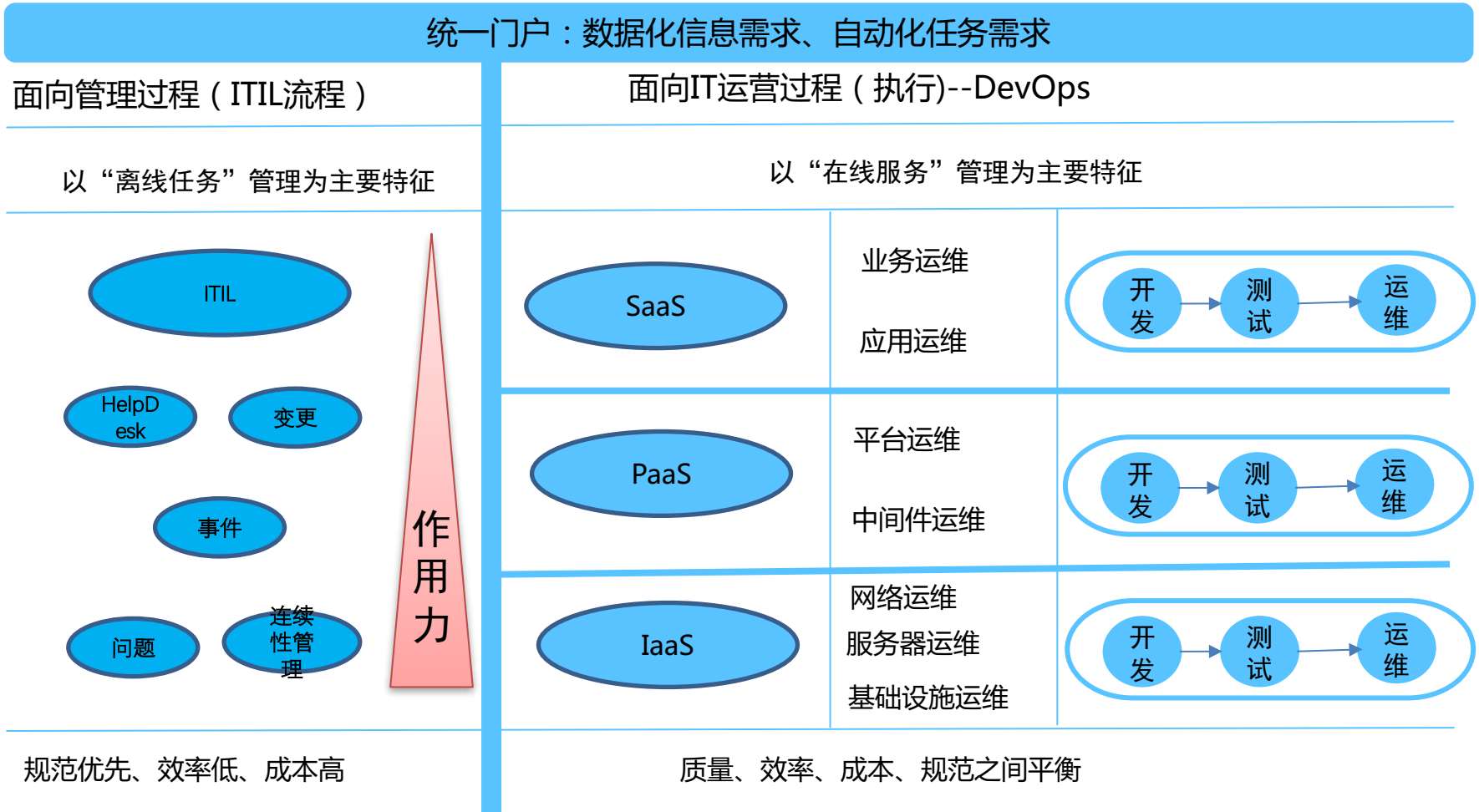
微服务架构实践

- 每个服务独立部署
- 每个服务独立构建流水线
- 在隔离的容器中运行服务
- 消费者驱动的测试（CDC ）



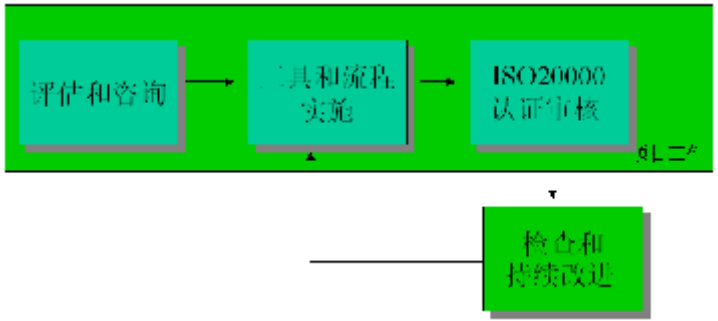
老树发新芽，New ITSM

ITIL因DEVOPS不同而不同



新IT需要NEW ITSM

传统的ITSM



- 强调过程标准、规范；
- 分阶段持续完善；
- 重质量，轻效率；
- 重建设，轻运营。

DevOps

设计，构建
测试
配置发行版
部署
反馈

- 自动化
- 高效
- 持续交付

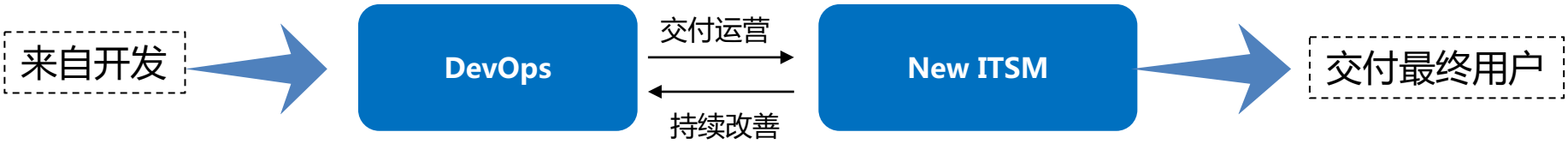
New ITSM

ITSM体系
建设
自动化运营
持续改善

- 重视运营大于建设；
- 实现运营自动化，提高运行效率；
- 加强持续改善，缩短周期。

什么是NEW ITSM？

新ITSM是对传统ITSM模式进行升级改造，使其能够充分适应和融合DevOps，共同实现业务价值的快速交付。

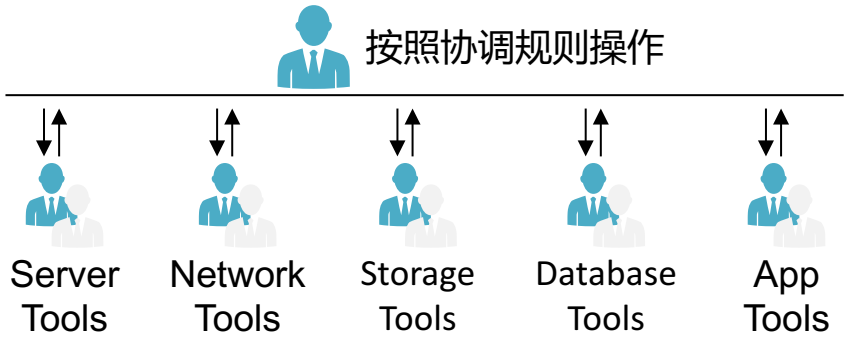


- 持续交付，频繁变更
- 在线部署，快速检测和修复缺陷

- 将DevOps的频繁变更进行细分，固定分类与操作步骤，形成标准变更，实现自动审批和实施；
- 将运维的经验，进行提炼总结，形成事件或问题诊断知识库，将诊断步骤标准化，通过DevOps的在线监测与修复，解决事件和问题。

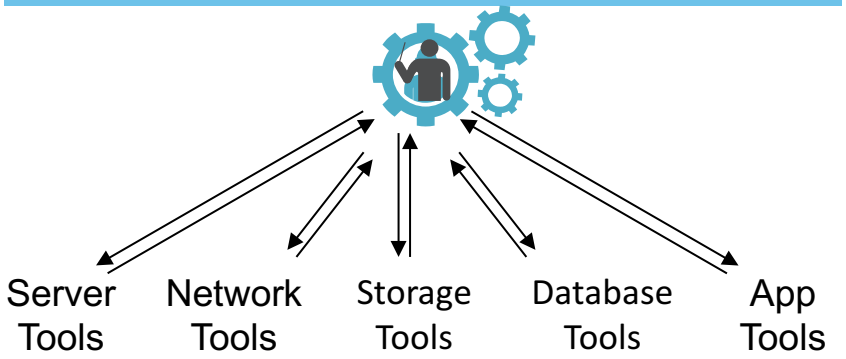
NEW ITSM结合DEVOPS实现自动化运维

开环任务协调



手工或半自动; 简单的流程
在每个组中有专门的任务执行工具
组到组的实现一系列操作
半自动的交换数据

闭环任务协调



完全自动化, 简单或复杂的流程
每项任务都有专门的执行工具
并行或串行组操作
自动交换数据

新ITSM在实际运营中的实例

1 **用户在系统中提交一个服务请求申请**

2 **通过web界面收集用户信息，并且触发服务请求流程。**

3 编制专门的脚本来自动化配置用户所需的基础设施

自动创建或分配一个虚拟机

自动连接和配置网络端口或Vlan

自动分配存储空间

自动分配其他基础设施

4 编制专门的脚本来自动安装相关软件或部署相关应用

- 自动安装配置操作系统
- 自动安装配置Tomcat或其他中间件
- 自动安装配置数据库
- 自动安装应用
- 自动加载数据
- 测试运行

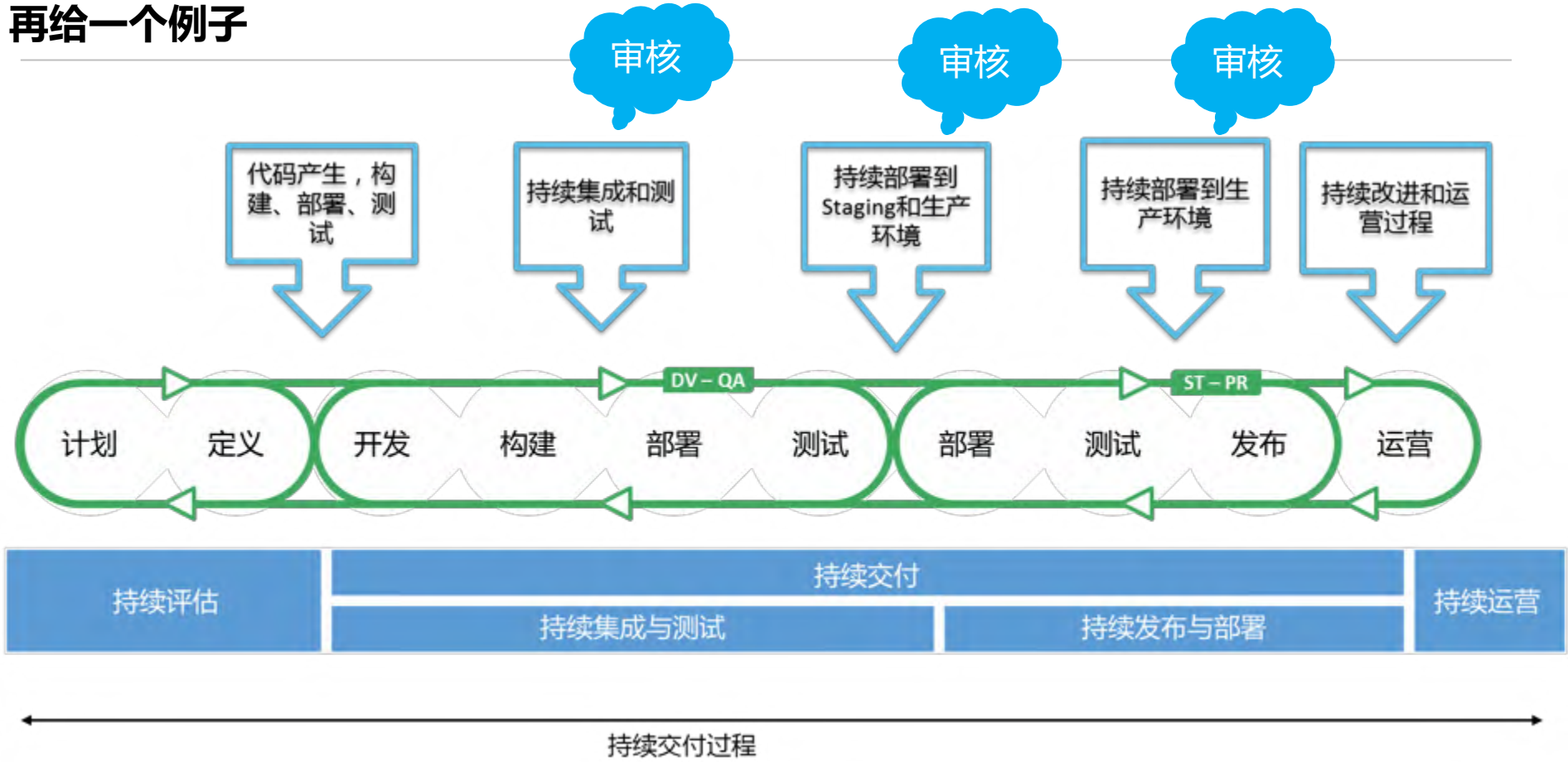
实时进度

5 **连接网络，启动应用，以便用户能够访问。**

一个小时，轻松搞定所有基础应用的按照配置，高效省事

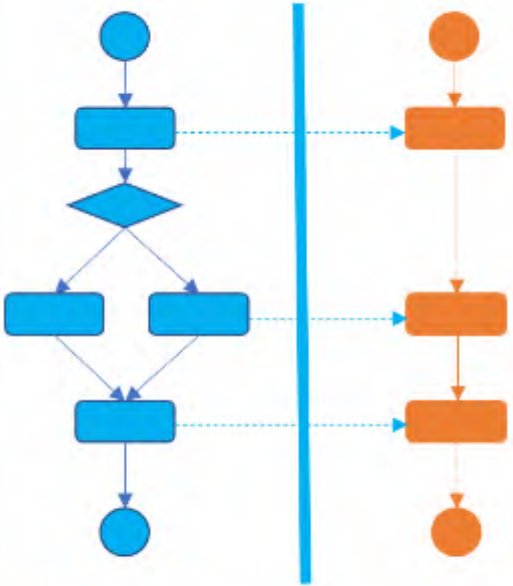
没有人工参与，没有协调延误，没有报错和问题

再给一个例子


审核
审核
审核
代码产生，构建、部署、测试
持续集成和测试
持续部署到Staging和生产环境
持续部署到生产环境
持续改进和运营过程
DV – QA
ST – PR
计划
定义
开发
构建
部署
测试
部署
测试
发布
运营
持续评估
持续交付
持续集成与测试
持续发布与部署
持续运营
持续交付过程

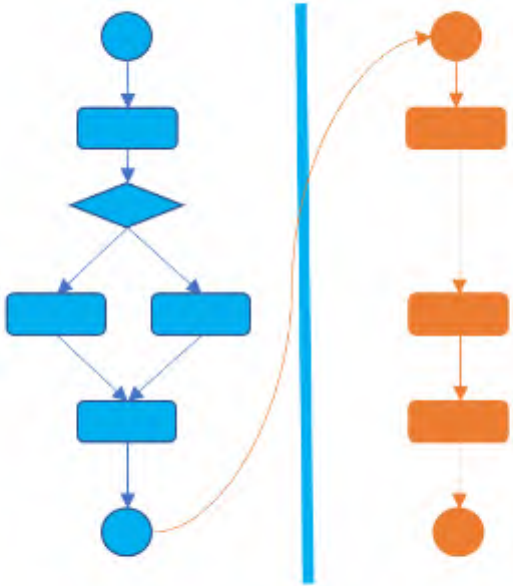
DEVOPS与ITIL融合模式

模式1：



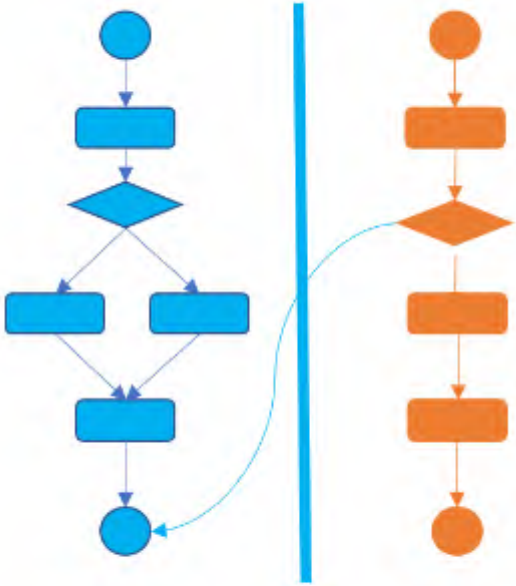
- 在线服务开通流程
- 资源管理流程

模式2：



- 重大变更流程
- 高稳定性服务保障流程、变更流程

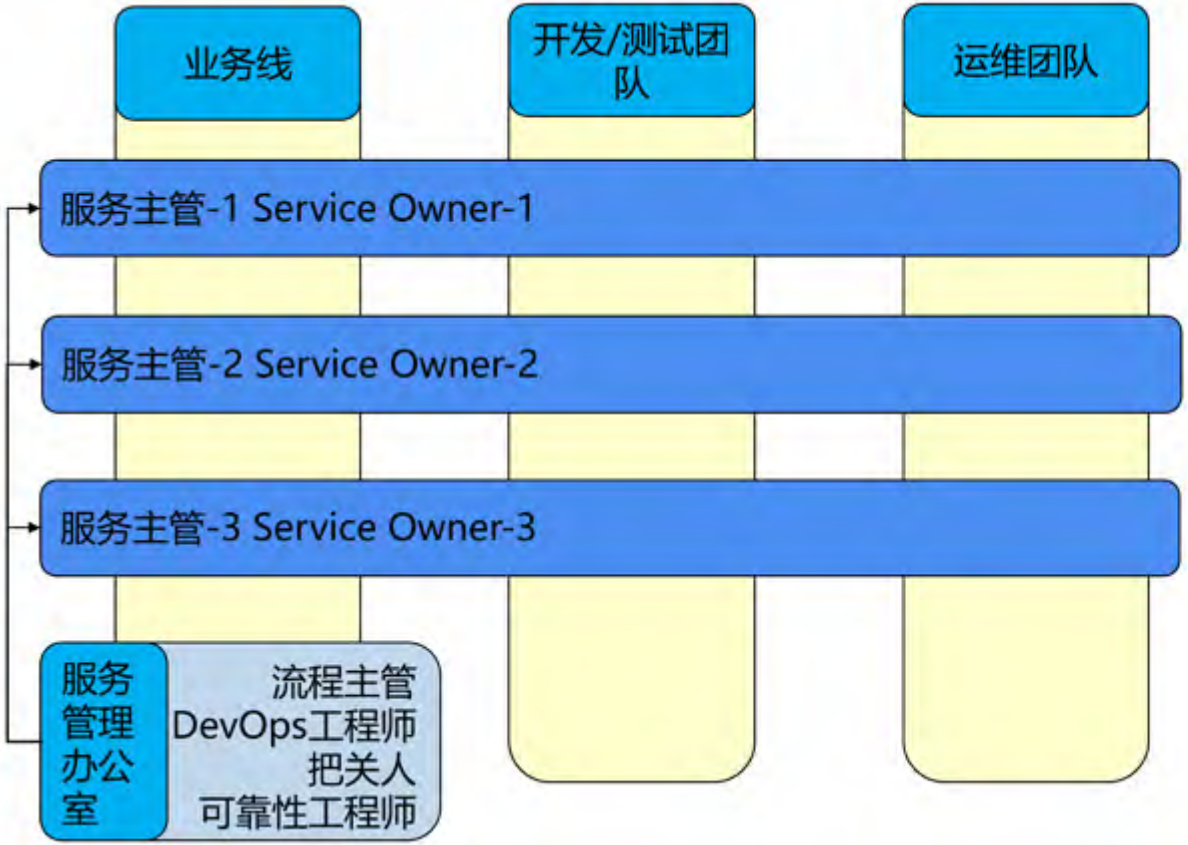
模式3：



- 敏捷发布流程
- DevOps变更流程

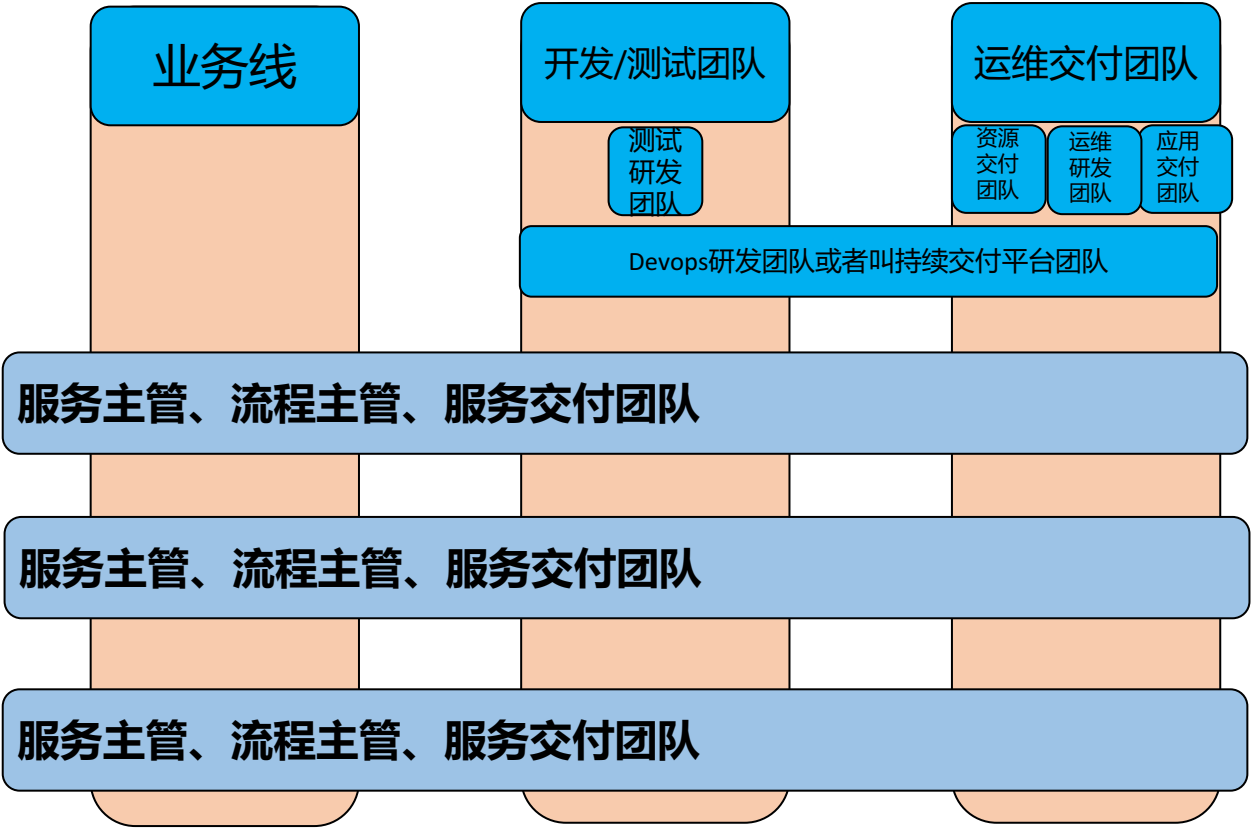
关于组织、实施与度量

DEVOPS的组织——二元组织(模式一)



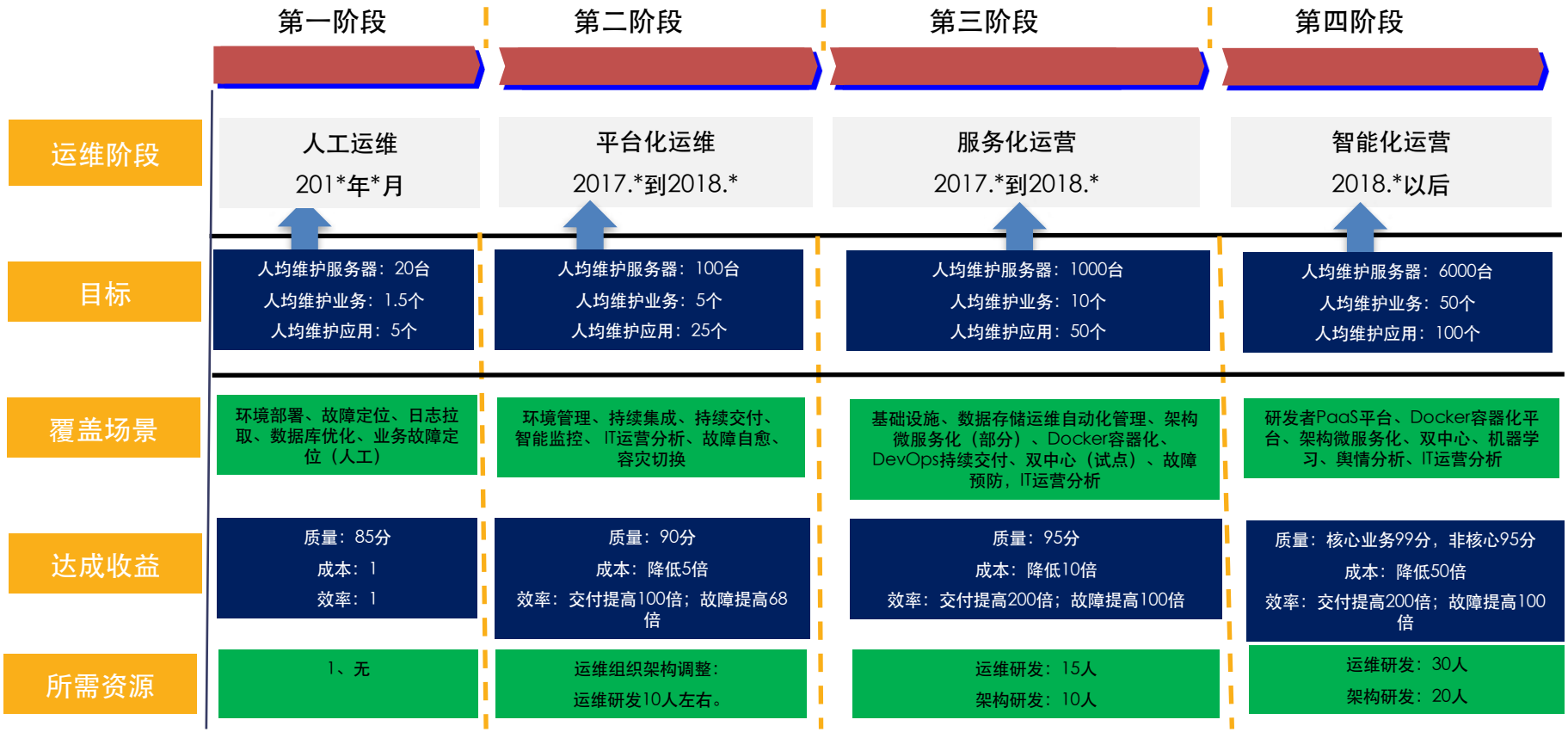
- 服务主管。对IT服务及时性相应负责，类似Scrum的PO。
- DevOps工程师。有义务提高和维护自动化流程，构建完整的自动化过程和工具，提升效率。
- 把关人。负责监控IT服务的运行状态和下一步发布的进展。
- 可靠性工程师（可选）。监控部署过程中的服务并处理正在服务执行中产生的问题
- 流程主管。领导并促进团队，这个角色类似于在Scrum中 的Scrum Master

DEVOPS的组织——二元组织（模式二）



- 运维交付团队。分资源交付团队、应用交付团队、运维研发团队。运维研发负责运维交付能力自动化。
- 开发测试团队。设立测试研发团队，负责测试能力自动化。
- DevOps研发团队。负责从持续交付的角度端到端的能力集成。
- 服务主管、流程主管角色不变。

IT运营管理实施步骤


第一阶段
第二阶段
第三阶段
第四阶段
运维阶段
人工运维
201*年*月
平台化运维
2017.*到2018.*
服务化运营
2017.*到2018.*
智能化运营
2018.*以后
目标
人均维护服务器：20台
人均维护业务：1.5个
人均维护应用：5个
人均维护服务器：100台
人均维护业务：5个
人均维护应用：25个
人均维护服务器：1000台
人均维护业务：10个
人均维护应用：50个
人均维护服务器：6000台
人均维护业务：50个
人均维护应用：100个
覆盖场景
环境部署、故障定位、日志拉取、数据库优化、业务故障定位（人工）
环境管理、持续集成、持续交付、智能监控、IT运营分析、故障自愈、容灾切换
基础设施、数据存储运维自动化管理、架构微服务化（部分）、Docker容器化、DevOps持续交付、双中心（试点）、故障预防，IT运营分析
研发者PaaS平台、Docker容器化平台、架构微服务化、双中心、机器学习、舆情分析、IT运营分析
达成收益
质量：85分
成本：1
效率：1
质量：90分
成本：降低5倍
效率：交付提高100倍；故障提高68倍
质量：95分
成本：降低10倍
效率：交付提高200倍；故障提高100倍
质量：核心业务99分，非核心95分
成本：降低50倍
效率：交付提高200倍；故障提高100倍
所需资源
1、无
运维组织架构调整：
运维研发10人左右。
运维研发：15人
架构研发：10人
运维研发：30人
架构研发：20人

IT运营管理平台导入路径

DEVOPS度量指标

	高效能组织	中等效能组织	低效能组织
发布频率 所负责的企业应用多长时间进行一次发布?	按需发布（每天进行多次发布）	每周至每月之间	每月至每半年之间
部署前置时间 距离代码提交到运行于生产系统，有多长时间?	少于一个小时	每周至每月之间	每月至每半年之间
平均故障修复时间 (MTTR) 应用从故障中恢复的平均时长？	少于一个小时	少于一天	少于一天
变更失败率 有多大比例的变更操作导致服务故障，或需要进行修复（包括回滚，开发Hotfix，补丁，甚至业务中断）？	0-15%	31-45%	16-30%

1 商业成功

- 利润率
- 市场占有率
- 连续现金流
- 连续付费率
- 生产效率

2 客户体验

- A/B测试结果
- 留存率
- 客户满意度度
- 核心交易的频率
- 用户的传播率
- 业务/app 用户平均花费时间
- 正常服务时间

3 成本

- 有效资源成本
- 有效服务成本
- 有效人力成本

4 速度

- 变更前置期
- 部署频率
- 故障恢复速度

5 质量

- 变更失败率
- Crash率
- 突发事件数
- 用户主动投诉数

案例：德邦物流案例（IT运营）

客户现状分析

传统IT运维管理基本仍然处在手工运维辅以少量工具状态，在这种状态下需要等到IT故障出现后再由运维人员采取相应的补救措施。这些传统式被动、孤立、手工劳作式的IT运维管理模式经常令IT运维部门疲惫不堪，也难以让IT系统使用部门满意。

救火式系统运维

- 简单重复的问题
- 事后处理
- 事倍功半
- IT运维质量低下
- 运维人员被动、效率低
- 业务部门满意度低

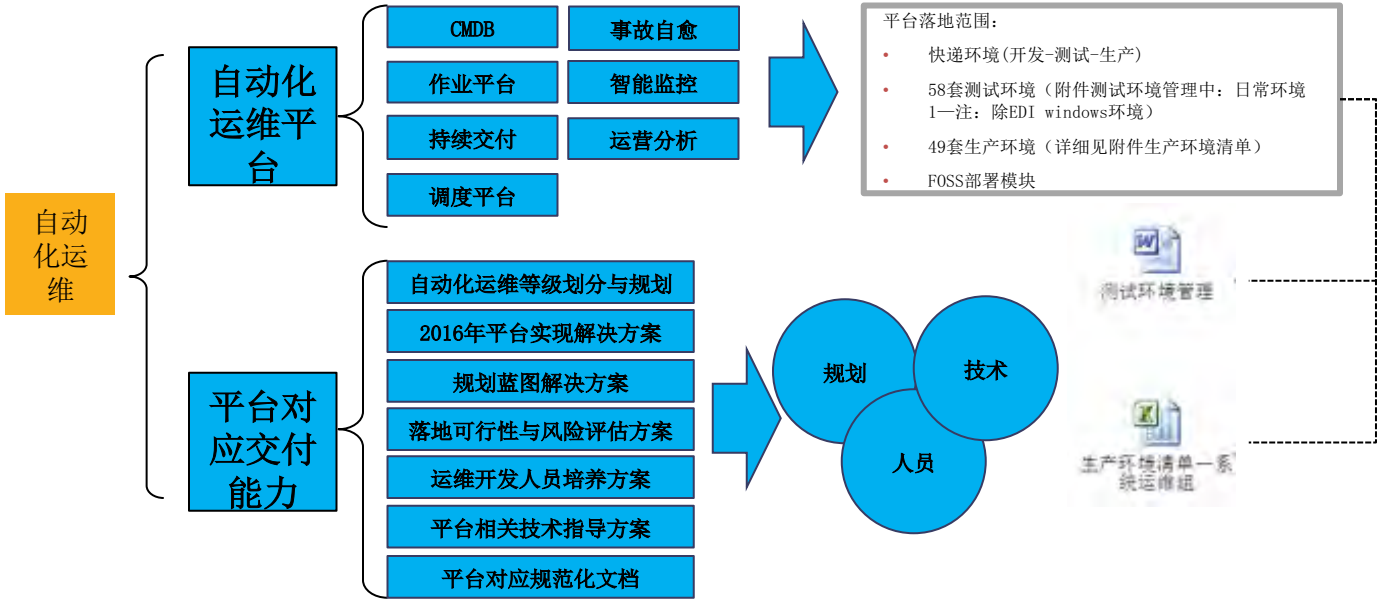
缺乏高效的IT运维机制

- 明确的角色定义和责任划分
- 很难快速、准确地找到故障原因
- 缺乏流程化的故障处理机制
- 欠缺规范化的解决方案
- 缺乏全面的跟踪记录
- 重监控，轻处理

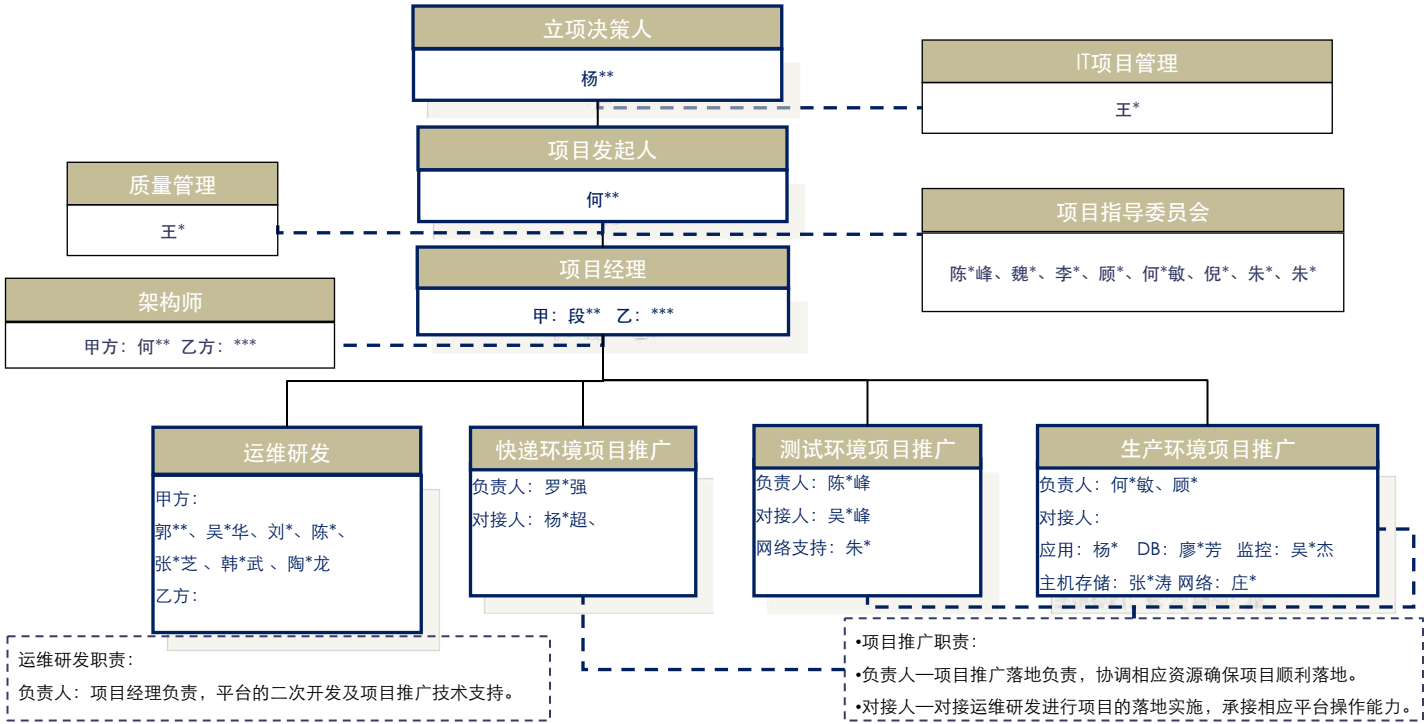
缺乏高效的IT运维工具

- 系统架构复杂
- 多种软硬件产品
- 设备数量众多
- 事件众多，难于应付
- 缺乏故障预警机制
- 故障难以得到主动、快速处理

IT运营管理平台解决方案



项目的组织结构


立项决策人
杨**
IT项目管理
王*
项目发起人
何**
质量管理
王*
项目指导委员会
陈*峰、魏*、李*、顾*、何*敏、倪*、朱*、朱*
项目经理
甲：段** 乙：***
架构师
甲方：何** 乙方：***
运维研发
甲方：
郭**、吴*华、刘*、陈*、
张*芝 、韩*武 、陶*龙
乙方：
快递环境项目推广
负责人：罗*强
对接人：杨*超、
测试环境项目推广
负责人：陈*峰
对接人：吴*峰
网络支持：朱*
生产环境项目推广
负责人：何*敏、顾*
对接人：
应用：杨* DB：廖*芳 监控：吴*杰
主机存储：张*涛 网络：庄*
运维研发职责：
负责人：项目经理负责，平台的二次开发及项目推广技术支持。
•项目推广职责：
•负责人—项目推广落地负责，协调相应资源确保项目顺利落地。
•对接人—对接运维研发进行项目的落地实施，承接相应平台操作能力。

案例：某证券CMDB实施效果

实施达到的效果（CMDB）

现有方式	弊端
通过excel方式人工统计资产	易出现误差，管理便捷性和合规性不高
固定资产盘点困难	易遗漏，无法准确记录资产从入库到出库全过程
相关资产与个人关联度较大	无法有效快速对用户提供服务
监控告警信息缺乏有效的配置依据	告警信息单一，无法快速定位事件源
变更时，人为经验判断风险。	有可能出现对变更预估不到，造成业务影响。

实施达到的效果

资源模型

三层简化结构：

- 应用层
- 服务层
- 基础层

资源项 43

属性 750

流程制度

5 维护动作

录入、修改、删除、人工变更、自动发现

2 关联流程

事件、主机上架流程

系统建设

即搜即得

100% 自动采集

标准化实时接口

与流程对应的维护权限控制

业务系统覆盖

业务系统 136个

组件 632个

实例信息

机房 8个

机柜 209个

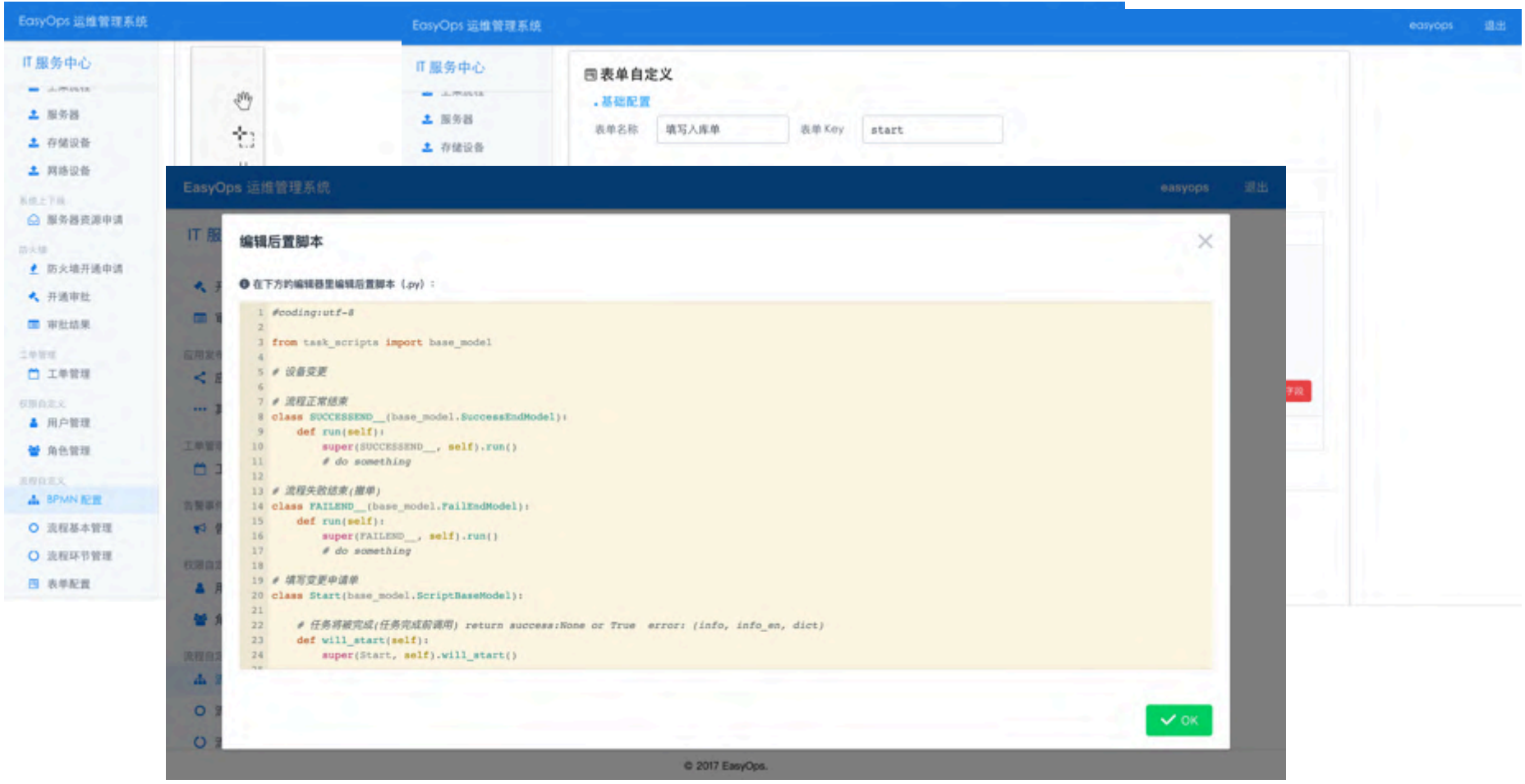
网络设备 424个

存储 16个

主机 4k个左右

在用IP 5k个左右

实施达到的效果（流程）


EasyOps 运维管理系统
IT 服务中心
服务器
存储设备
网络设备
服务器资源申请
防火墙开通申请
开通审批
审批结果
工单管理
用户管理
角色管理
BPMN 配置
流程基本管理
流程环节管理
表单配置
表单自定义
基础配置
表单名称
填写入库单
表单 Key
start
easyops
退出
编辑后置脚本
在下方的编辑器里编辑后置脚本（.py）：
#coding:utf-8
from task_scripts import base_model
# 设备变更
# 流程正常结束
class SUCCESSEND__(base_model.SuccessEndModel):
def run(self):
super(SUCCESSEND__, self).run()
# do something
# 流程失败结束(撤单)
class FAILEND__(base_model.FailEndModel):
def run(self):
super(FAILEND__, self).run()
# do something
# 填写变更申请单
class Start(base_model.ScriptBaseModel):
# 任务将被完成(任务完成前调用) return success:None or True error: (info, info_en, dict)
def will_start(self):
super(Start, self).will_start()
OK
© 2017 EasyOps.

关于优维、EasyOps及其他

关于优维

团队介绍

基本上来自一线BAT，有着丰富的海量互联网运维经验，海量运维平台建设经验，不同应用架构经验。

咨询

提供如下咨询服务：

1、DevOps 端到端咨询服务

2、互联网运维咨询服务

3、持续交付服务

优势

优势如下：

1、全面的互联网运维实践，如规范、流程、平台和架构经验。

2、多家传统企业项目成功实施经验，融合传统与互联网。

产品与服务

一站式DevOps运维平台

1、全面IT资源管理

2、持续交付

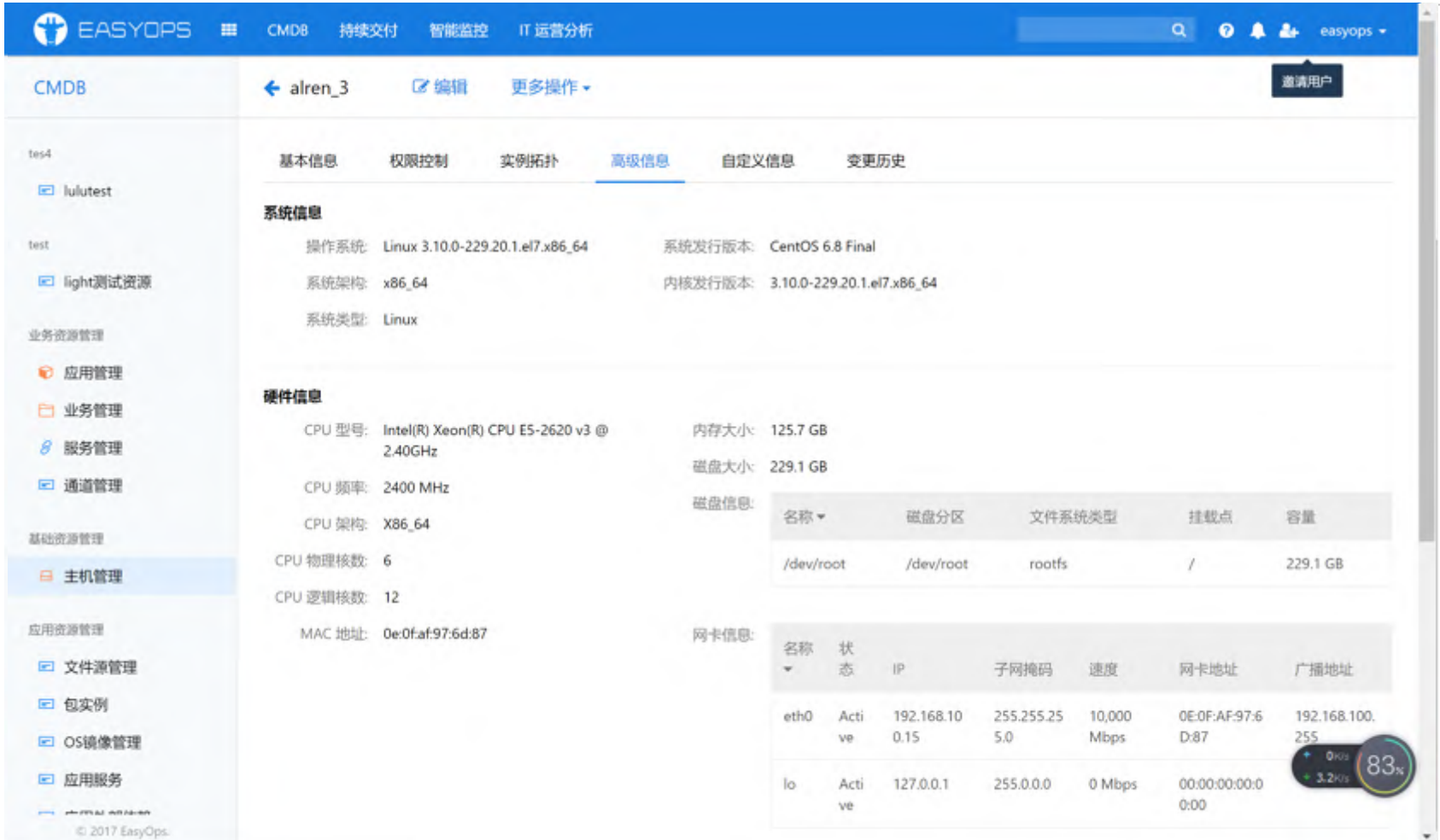
3、运维自动化

4、应用监控平台（大屏、监控、故障自愈）

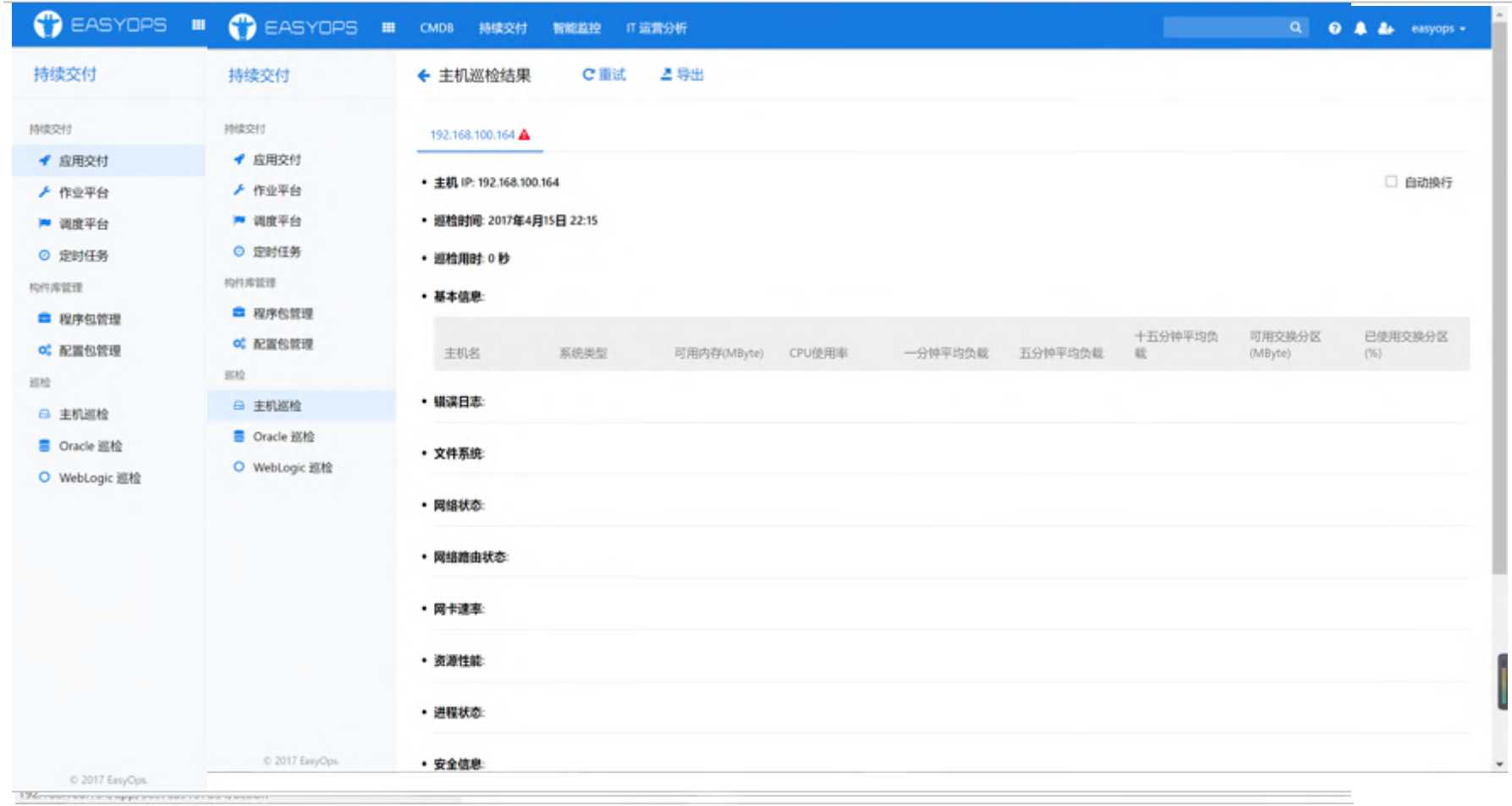
5、运营分析平台

EASYOPS，一站式DEVOPS运维平台

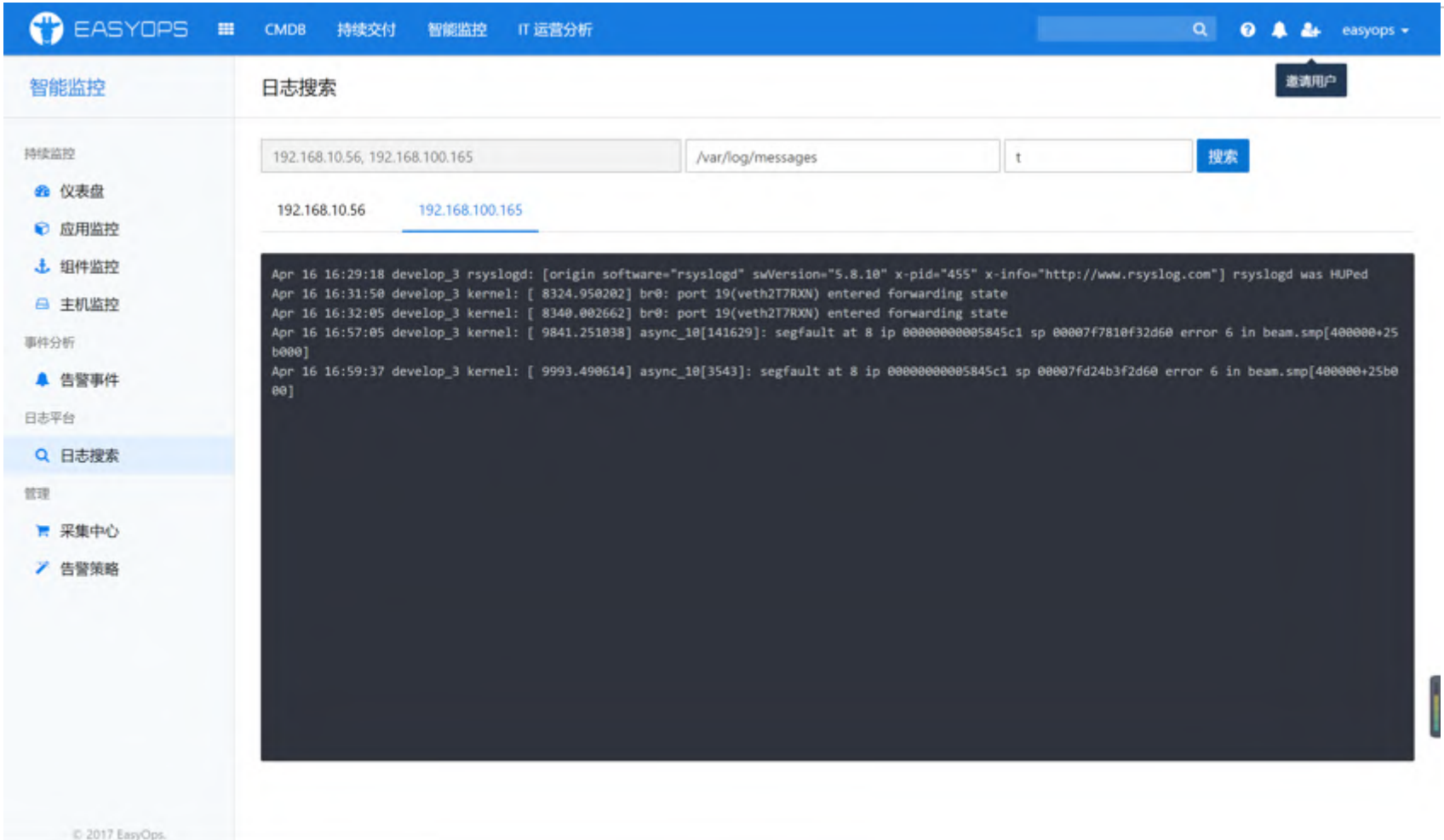
EASYOPS之CMDB



EASYOPS之应用交付



EASYOPS之智能监控



写在后面的话

1、基于交付链(Dev/Test/Ops)的全局优化，而非局部(Ops)优化。

2、建立以业务运维导向、应用运维支撑/驱动、基础运维服务的运维体系与架构。

3、实现运维向IT运营的跨越，在产生业务价值的同时，也需要关注工程效率。

4、 DevOps、SRE、PE都是在工程化/业务化运维能力，甚至IT的能力。

5、运维架构也有大中台、前后台的策略。

6、 DevOps是组织、流程、架构、交付、文化、平台等全面的能力提升，但Ops有最好的DevOps实施驱动力。

7、从应用出发，形成运维最强的驱动力。

高效运维社区
GreatOPS Community

会议

- 3月18日 DevOpsDays 北京
- 8月18日 DevOpsDays 上海
- 全年 DevOps China 巡回沙龙
- 4月21日 GOPS深圳
- 11月17日 DevOps金融上海

培训

- EXIN DevOps Master 认证培训
- DevOps 企业内训
- DevOps 公开课
- 互联网运维培训

咨询

- 企业DevOps 实践咨询
- 企业运维咨询

商务经理：刘静女士
电话 / 微信：13021082989
邮箱：liujing@greatops.com

Thanks

高效运维社区

开放运维联盟

荣誉出品

想第一时间看到
高效运维社区公众号
的好文章吗？

请打开高效运维社区公众号，点击右上角小人，如右侧所示设置就好