我们运维的 CMDB 模型是不是都做错了？

点击关注👉 DevOps技术栈 2025年07月24日 11:40 河南

> 原文链接：https://cloud.tencent.com/developer/article/1540181

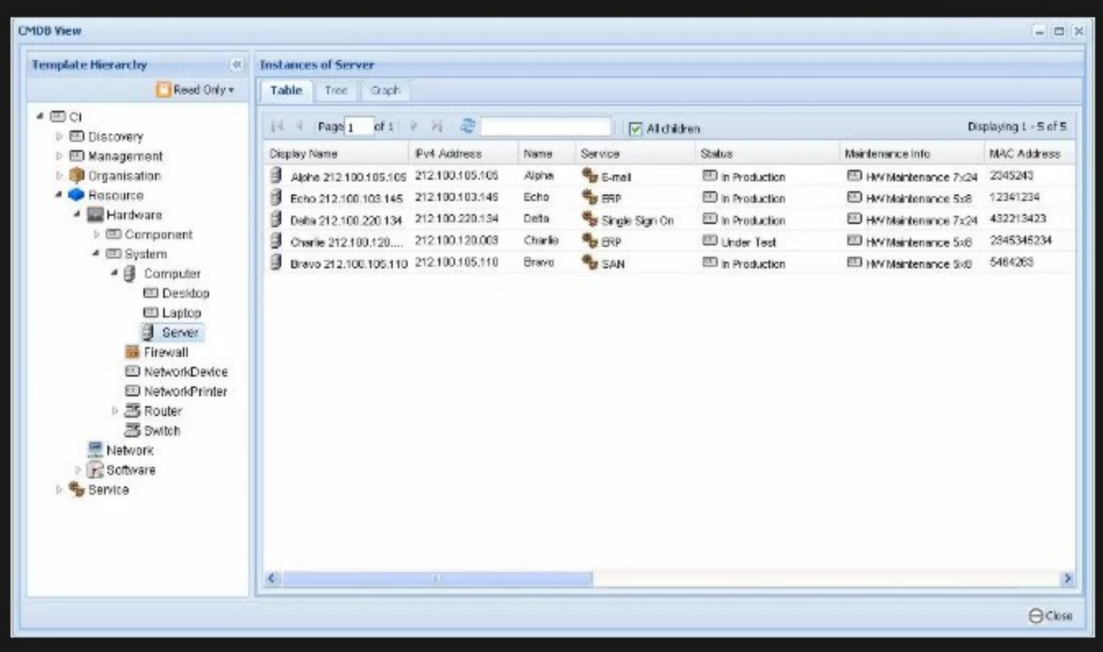
大家有没有想过，我们过去做的CMDB模型都是错的？也许真的错了，可以往下看看。

当前CMDB模型面临的问题

当前CMDB的模型问题：

- 首先是思考的深度不够，当今很多**CMDB的模型还是聚焦在底层资源**。这个底层资源指的一部分是IaaS层的资源管理，另一部分是PaaS层中间件的资源管理。到上层应用这块，其实它的模型表述特别简单，只有一些应用的基本信息。
- 第二个要讲的问题是**无应用视角**。今天我们创建管理了这么多资源对象，但不知道是给谁用的，其实真正的着力点是应用。这个我将其总结为无应用层的理解力。
- **模型的动态性不强**。每个模型对象调整它的属性或者关系的时候，在传统数据库里技术端的特点带来的代价特别高。我把模型的动态性抽象成两个维度，第一是模型对象之间在CI级别的动态性，第二个就是实例级。
- 第四个问题是**场景的过度设计**。我认为场景是可以预设的，但是细粒度的模型会带来很大的管理负担。有时候会把场景考虑得过于复杂，导致这里面的模型管理后续负担特别重。从简到繁很容易，但是从繁到简很难。
- **技术限制想象力**。受CMDB平台技术本身的能力限制，导致无法扩展这个模型。
- **欠缺IT架构思考力**。我要讲的是从业务架构到应用架构再基础架构。业务架构中还包含了基础设施架构和数据架构。弄清楚这三者的关系后，就能表达出在每一层架构上所带来的本质上的关系连接到底是什么。

CMDB系统截图：



构建CMDB模型的正确思路

新一代CMDB到底新在哪儿？

新思维：突破配置管理的认知，导致边界不清。配置往IT资源方向转变。

新方法：自上而下的推动CMDB落地，而不是自下而上。

新模型：模型重构，传统的关系模型无法满足。

新技术：使用新的技术，新的功能架构，重新定义功能边界。

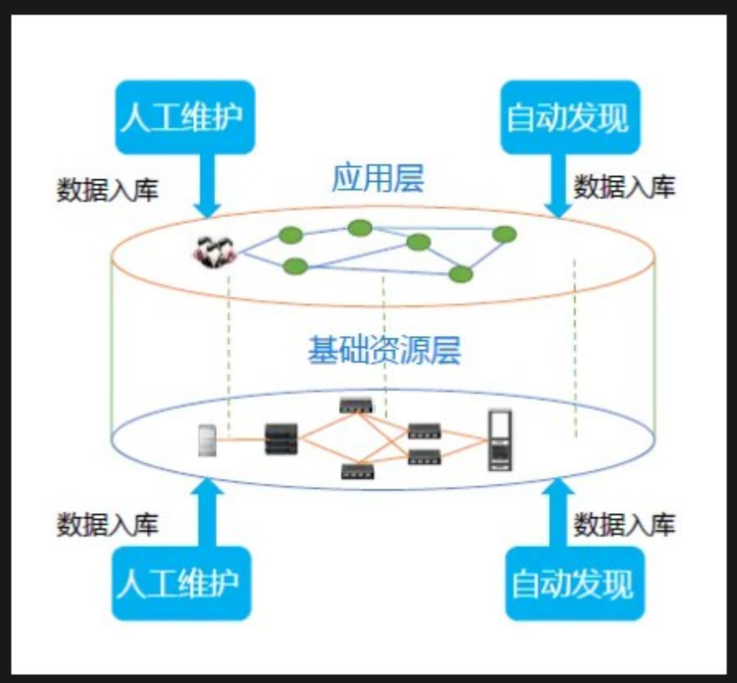
CMDB元数据的两类用途：

CMDB模型最终是要实例化数据和关系的，正确的模型构建可以为多变的场景提供数据基础。

1. 面向管理层的ITSM流程。在很多传统企业里面，CMDB还是要为ITSM的流程做好数据支撑服务。
2. 面向执行层的DevOps过程。端到端整个IT交付过程需要完整的元数据，特别是应用层面的元数据。

总的来说，新一代CMDB应该能支撑整个IT过程管理（ITPM）,所以CMDB可以成为：基础元数据平台、数据总线分享平台、共享实例数据平台、统一数据规则平台等等。

两层CMDB，构建不同管理视角：



CMDB架构分基础资源层架构和应用资源层架构。应用层资源架构把相关的资源以应用为中心实现资源整合。资源及其资源的关系称之为拓扑（应用拓扑、物理拓扑），资源管理方式有人工维护和自动发现两种方式，从详细的事前、事中和事后来看，可以分成详细的四中模式：人工、IT对象生命周期流程、场景化变更管理、自动发现等等。

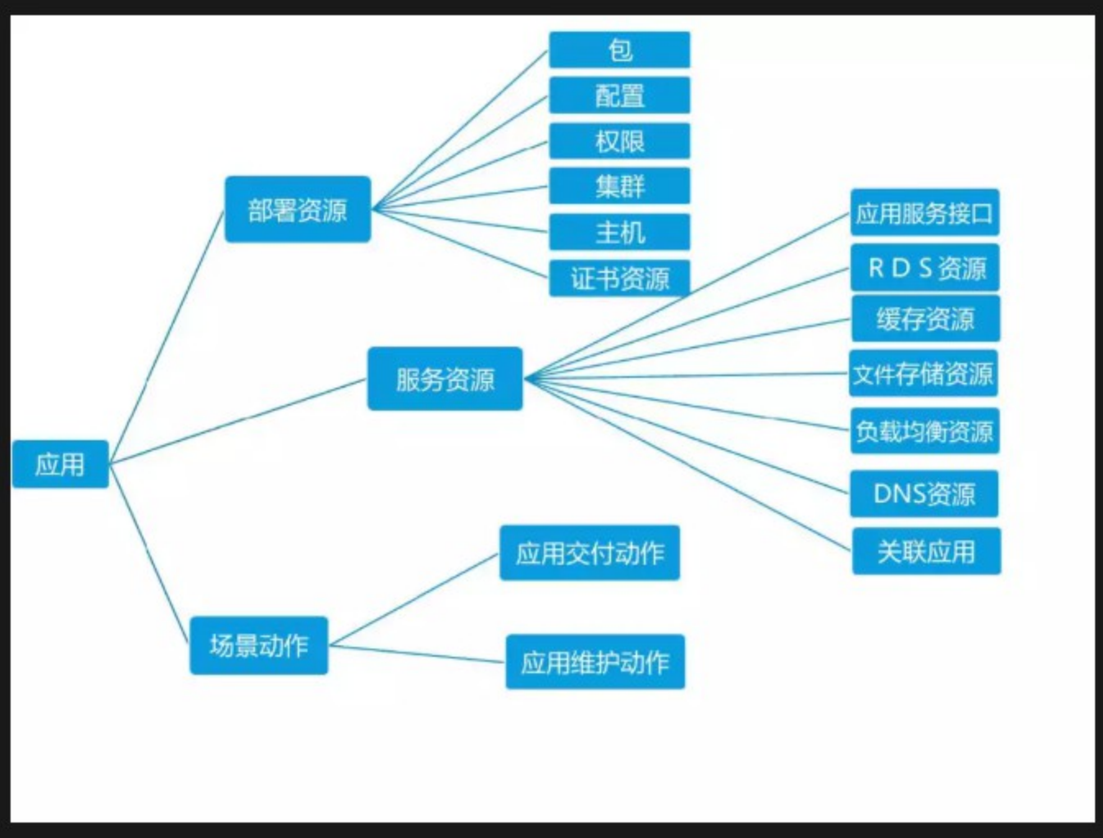
基础CMDB建设五原则：

1. 面向IaaS和PaaS设计，能够管理底层的一切资源。
2. 状态控制借助运维流程自动化完成。
3. CI的维护要深度使用自动发现，而不是人工维护。
4. 资源信息必须能为上层应用提供服务。
5. 必须满足基础资源的CI管理需要。

应用CMDB建设七原则：

1. 提供统一的应用元数据管理能力，和应用类型无关。
2. 核心诉求是应用生命周期管理。
3. 以应用为中心，而非基础资源为中心。
4. 从应用资源的角度构建起与IT资源的弹性关系。
5. 为应用资源、动作、状态的统一管理提供支撑。
6. 以统一的基础资源层CMDB作为基础。
7. 核心场景就是持续交付。

应用CMDB模型层次化理解：

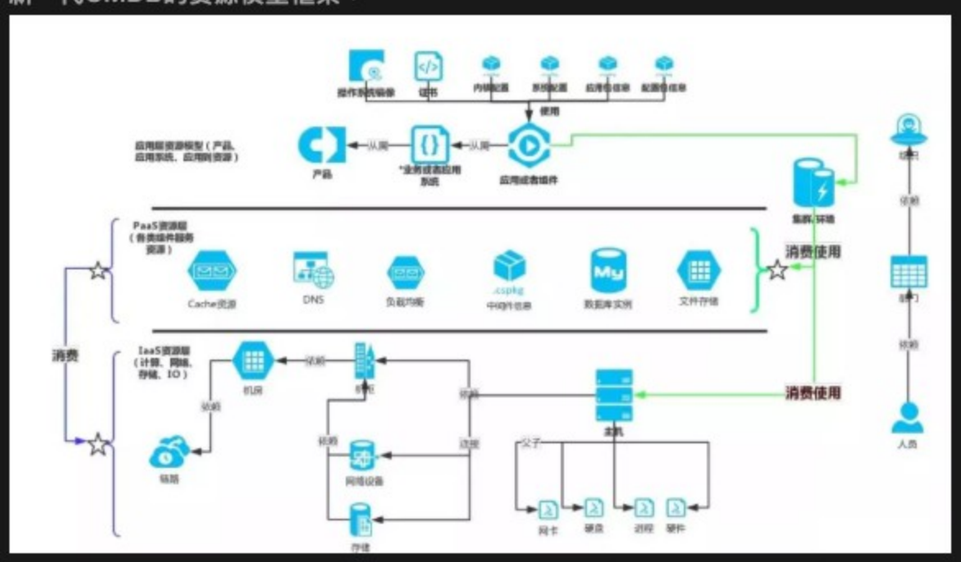


应用CMDB是面向资源的完整描述，**应用的资源分成应用的部署资源、服务资源和动作资源。**

- **部署资源**是一次应用部署所依赖的资源，一般又称本地资源，比如说主机Host、程序包等等。
- **服务资源**是应用运行依赖的资源，一般有称之为附加资源（来自于12factor），比如说应用的服务接口、应用依赖的PaaS资源、应用依赖的应用资源等等。
- **场景动作**是资源其上附加的动作描述，是资源的管理方法。

新一代模型的标准化框架

新一代CMDB的资源模型框架：

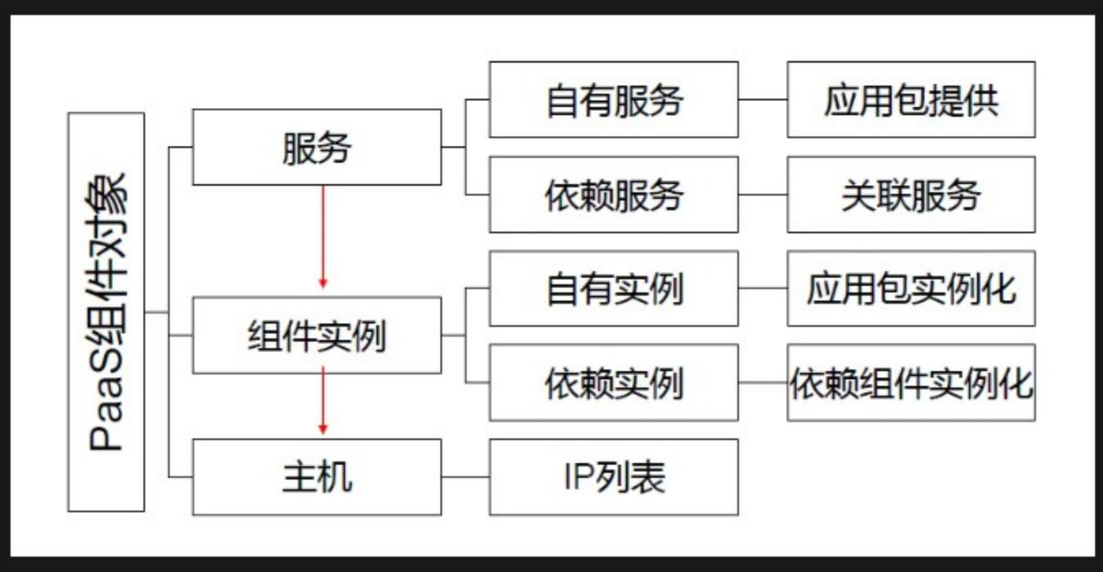


核心准则：一个资源能够提供服务，还要看它关联的资源，因此必须采用结构化模型方案。纷繁复杂的IT对象模型，其实只有两种：一种是硬件对象模型，一种是软件对象模型。这两种模型都要用新的模型表达方法来做——结构化模型定义方法，而非关系型平面表达模式。

IaaS层硬件对象模型：

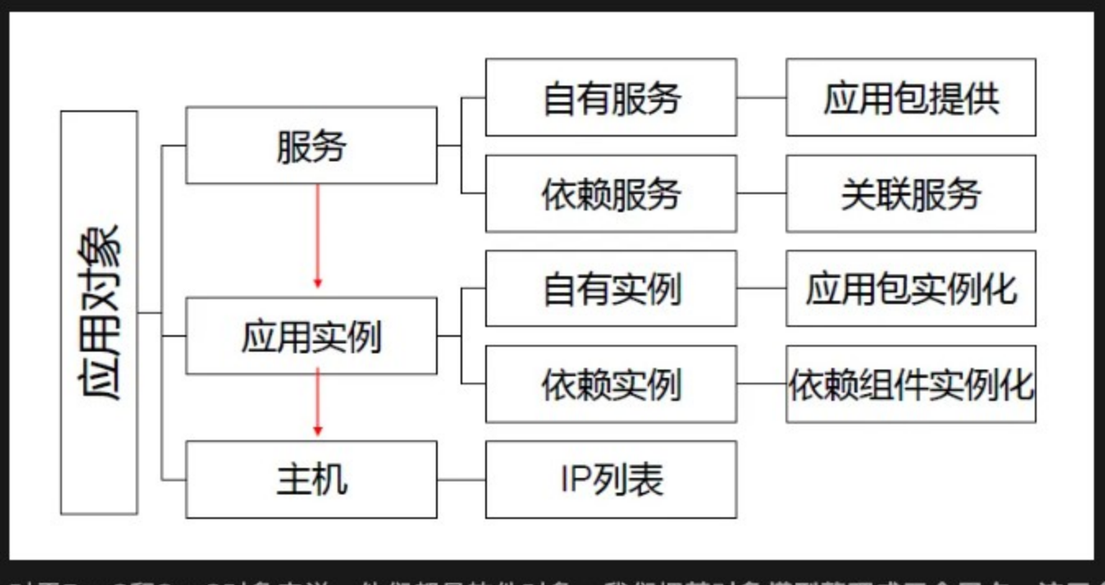
针对每一个象实例化描述它们就可以了，无非就是属性和关系。

PaaS层对象模型核心概念：



一定要深刻理解服务、实例和主机之间的层次关系，并且要精确表达注意组件和集群的区别，例如Mysql组件和Mysql集群。待会儿和SaaS层对象一起详解。

应用层对象模型核心概念



对于PaaS和SaaS对象来说，他们都是软件对象。我们把其对象模型整理成三个层次，这三个层次中必须要包含服务、组件服务节点和主机三层，而他们又分别对应三层、四层和七层模型。

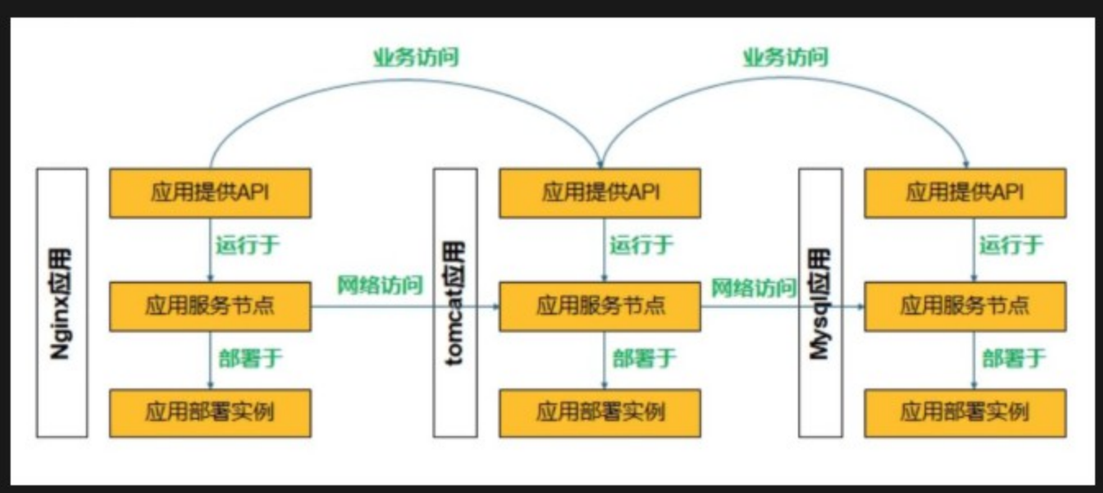
- 最底下的层次其实应该叫做部署资源，主机只是其中的一种。
- 服务运行过程中起了哪些运行实例、这些实例进程在哪些主机上，主机延伸出来就是IP列表。组件实例有两种，一种是自有实例，当程序运行的时候需要的应用包实例化。一种是依赖实例，就是将依赖组件实例化。
- 自有服务是自己启动的服务对外提供的时候以怎样的方式暴露出去。依赖服务则是运行时还关联了哪些服务。

应用层和PaaS对象都要深刻理解服务、实例和主机之间的层次关系，并且要精确表达。

一个应用架构的示例

接下来，我们以一个真实的应用系统架构为例子，如下：

通过我们讲的模型，最终表达如下：



在CMDB模型中，必须要表达这些元素的横向和纵向关系，才能构建一个真正的应用系统完整的视图，其中包括应用架构视图、应用访问视图、应用部署视图等等。

这是我要讲的新一代CMDB模型的全部，为什么我说以前的CMDB模型可能都是错的，大家也应该看出构建模型的思路有所不同。这个地方还有一个关键的技术问题没讲：是否应该关系数据库来实现CMDB？我的答案是否定的。但如果选择一个非关系型数据库，如何选型？考虑什么要素？这些数据库的坑怎么填？这里面有涉及到实践经验了。

往期推荐

- 双机热备神器 Keepalived，实现零故障！
- Linux Shell 奇技淫巧：sed
- 云上自动化运维利器：Terraform
- 线上Linux服务器中挖矿病毒，如何处理？
- Jenkins 搭建、权限管理、参数化、流水线等，太详细了！
- 线上Linux服务器每天1w多条暴力破解，如何防范？
- 企业级 MySQL 高可用集群搭建与监控，太详细了！
- Jenkins Pipeline 流水线详解，建议收藏！
- 终于搞懂了Nginx反向代理！
- Linux 网络丢包怎么排查？终于搞懂了！
- K8s 如何实现金丝雀/灰度发布？
- 20多个Linux命令分析日志，太全了！
- K8s 部署一套 MySQL 集群，稳！
- 10个 Linux运维操作禁忌！
- 40道 Nginx 常见面试题（带答案）
- Jenkins 生产环境笔记，很详细！
- 19 个 K8s 集群常见问题与，建议收藏
- Nginx 工作原理和优化总结 （超详细）
- 六个高频Linux运维故障排查笔记！
- 17 个运维常用指标，90%人都不知道！
- 应用程序容器化后，为什么性能下降这么多？
- Nginx限流详解，应对流量突发和恶意攻击
- Linux 磁盘打爆了，如何排查?
- 面试官问你：CPU狂飙900%，该怎么处理？
- Nginx 性能优化有这篇就够了！
- K8s Pod“OOM Killer”，原因找到了
- K8s Pod 故障排查，一个不为人知的技巧！
- 大厂总结 Nginx 高并发性能优化笔记
- 基于 Jenkins 搭建一套 CI/CD 系统
- Nginx 七大应用场景（附配置）
- 史上最全 Jenkins Pipeline流水线详解
- 主流监控系统 Prometheus 学习指南
- 40个 Nginx 常问面试题
- 常见Linux运维面试题，找工作的必看！
- 25个 MySQL 常见面试题及答案

点亮，服务器三年不宕机