CMDB中心化的运维平台建设

千量级互联网运维CMDB的实现

刘海阳
@海阳的阳
liuhaiyang@gmail.com

我们面对的问题

- 几千服务器如漆似胶
- 申请机器？先花一天找找看
- 空闲机器大隐于机房
- 业务关系混乱
- 核心数据保存在人脑中
- 一人请假全体歇菜
- 大量垃圾报警
- 依赖关系傻傻搞不清楚
- 一万IP手工管理，用时才知错
- 没人敢下线历史业务
- 业务挂了没人知道
- 沟通基本靠吼
- …
- …

我们的解决之道

- 建设CMDB
  - 建模
  - 流程
- CMDB中心化
  - 监控
  - 报警
  - 发布
  - …

建设CMDB

现状

- X千台服务器，由一个人维护
  - 一人请假全体歇菜
- 记录在Notes+Excel，二维数据
  - 没有领域模型
- 一切数据都是纯文本
  - 业务：(EPG(搜索)/XX项目/预留/Web3.0/EPG(已关机))
  - IP：201.45.13.1(2/4/6/14.7)
  - 磁盘：1.8T(300G*6)
- 你存在～我婶婶的脑海里
  - 所有流程依赖人脑确认数据，效率极低

目标

- 建立运维领域模型
  - 用OO方法抽象业务，整理核心流程
- 数据程序可读
  - 禁止非格式化的纯文本数据
- 吃掉大脑！
  - 挖出人脑里的信息
  - 存储，公开，透明

撸袖管，开干！


为革命敢超世界先进水平
毛澤東選集
WEI GE MING GAN CHAO SHI JIE XIAN JIN SHUI PING
华声论坛
bbs.voc.com.cn

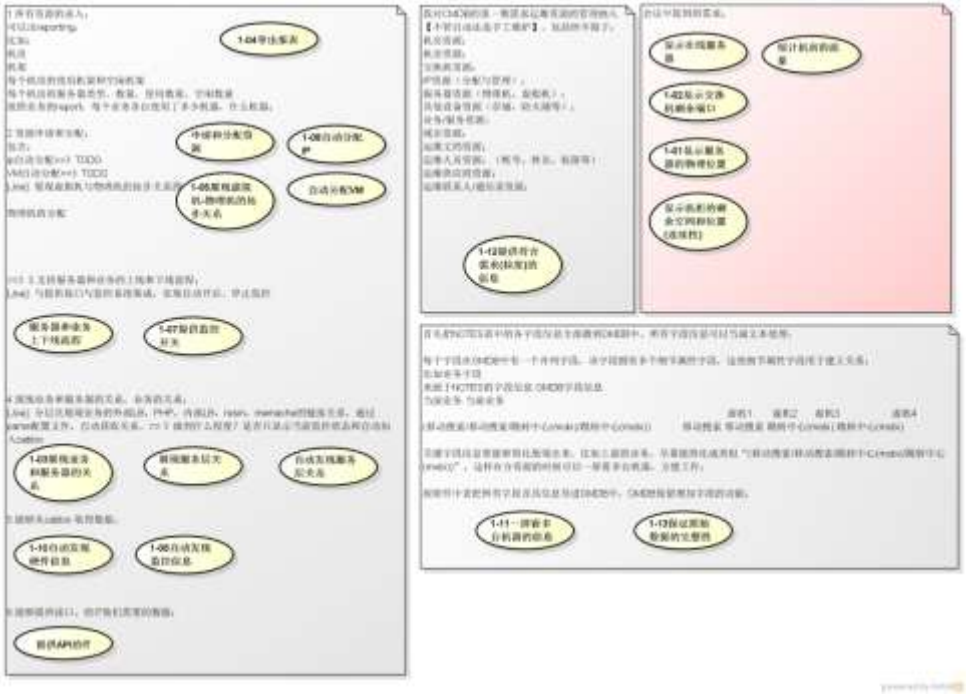
用户访谈

- 你是谁？
- 你是怎么工作的？
- 你关心什么数据？Input？Output？
- Pain Point？

定义CMDB用户角色

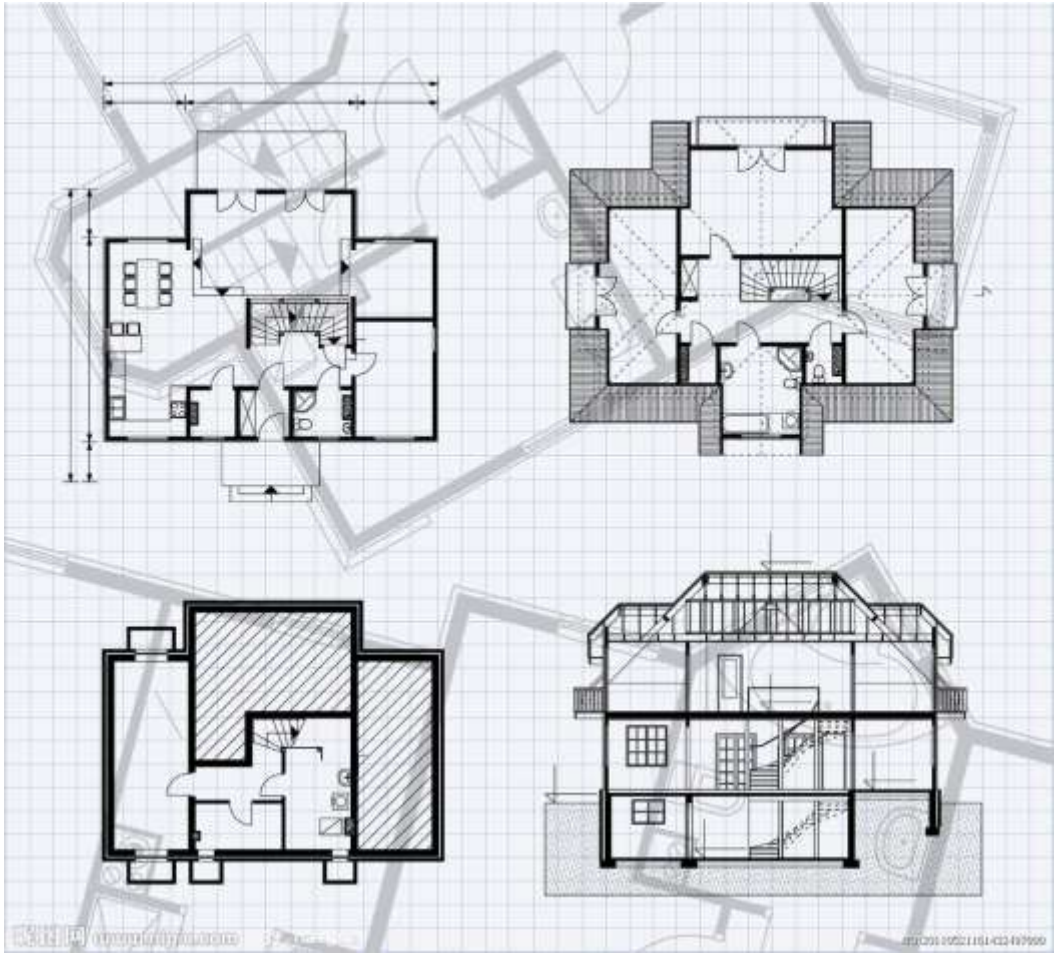
- 谁会关心CMDB里的数据？
  - IDC
    - 数据维护者，资源分配者
  - 业务负责人
    - 数据使用者，资源申请者
  - 一线运维(NOC)
    - 监控者
  - 财务/商务
    - 服务器/机房/带宽一切和钱有关的…
  - 老板
    - 你懂的

需求整理


1-04导出报表
1-08自动分配IP
1-05展现虚拟机-物理机的拓扑关系
自动分配VM
1-07提供监控开关
1-03展现业务和服务器的关系
展现服务层关系
自动发现服务层关系
1-10自动发现硬件信息
1-06自动发现监控信息
提供API给外
1-12提供符合需求(粒度)的信息
1-11一群要求台机器的信息
1-13保证原始数据的完整性


一阶段
二阶段
三阶段
拓扑结构
1-01显示服务器的物理位置
显示机柜的剩余空间和位置(连续性)
1-02显示交换机剩余端口
显示在线服务器
1-03展现业务和服务器的关系
1-04导出报表
1-05展现虚拟机-物理机的拓扑关系
VM-物理机的自动发现
监控系统相关
1-06自动发现监控信息
1-07提供监控开关
统计机房的流量
元数据发现
自动发现服务层关系
1-10自动发现硬件信息
运维自动化
服务器和业务上下线流程
自动分配VM
申请和分配管道
1-08自动分配IP
外部接口
提供API给外
1-09提供自用的API
1-11一群要求台机器的信息
展现服务层关系
1-12提供符合需求(粒度)的信息
1-13保证原始数据的完整性

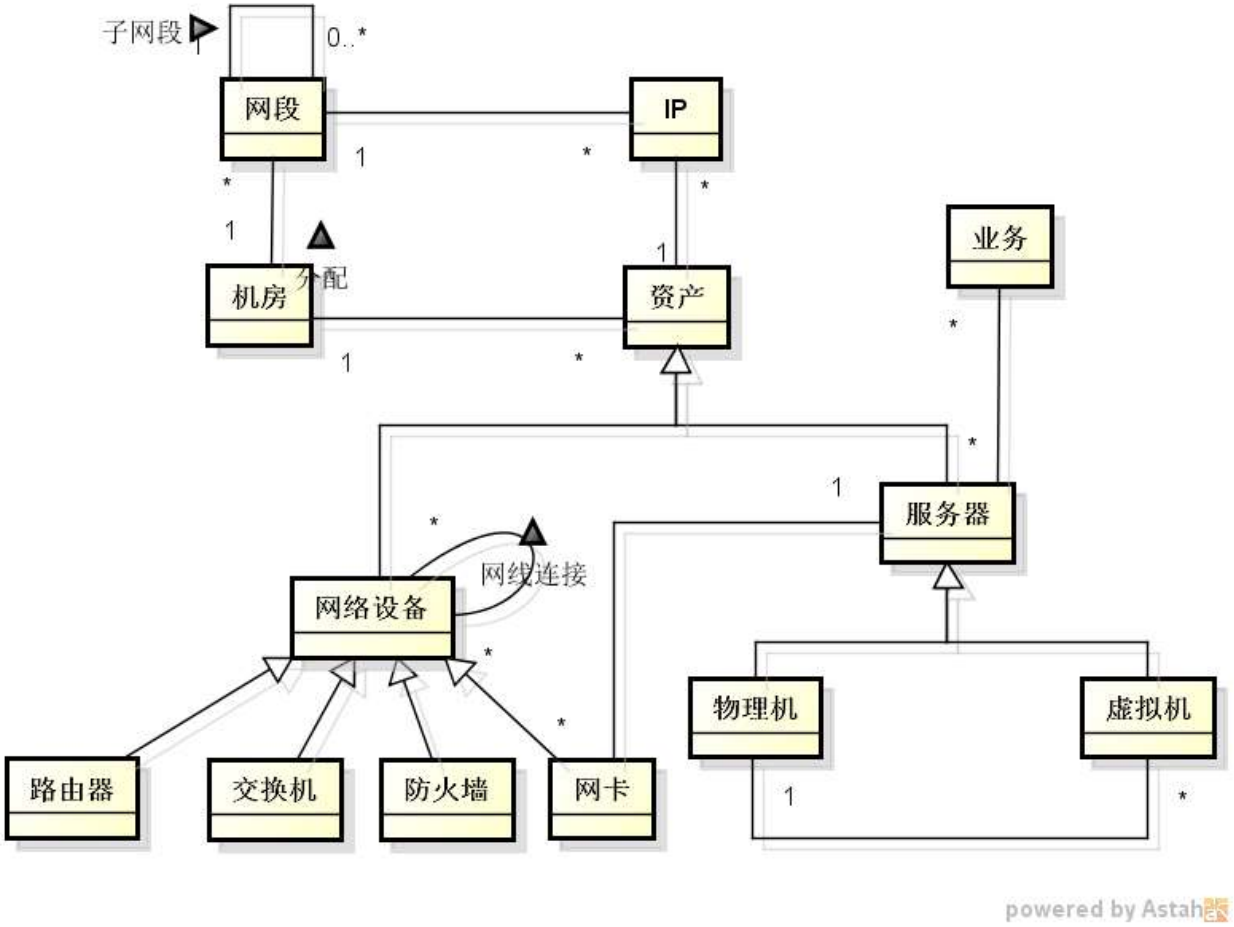
建模



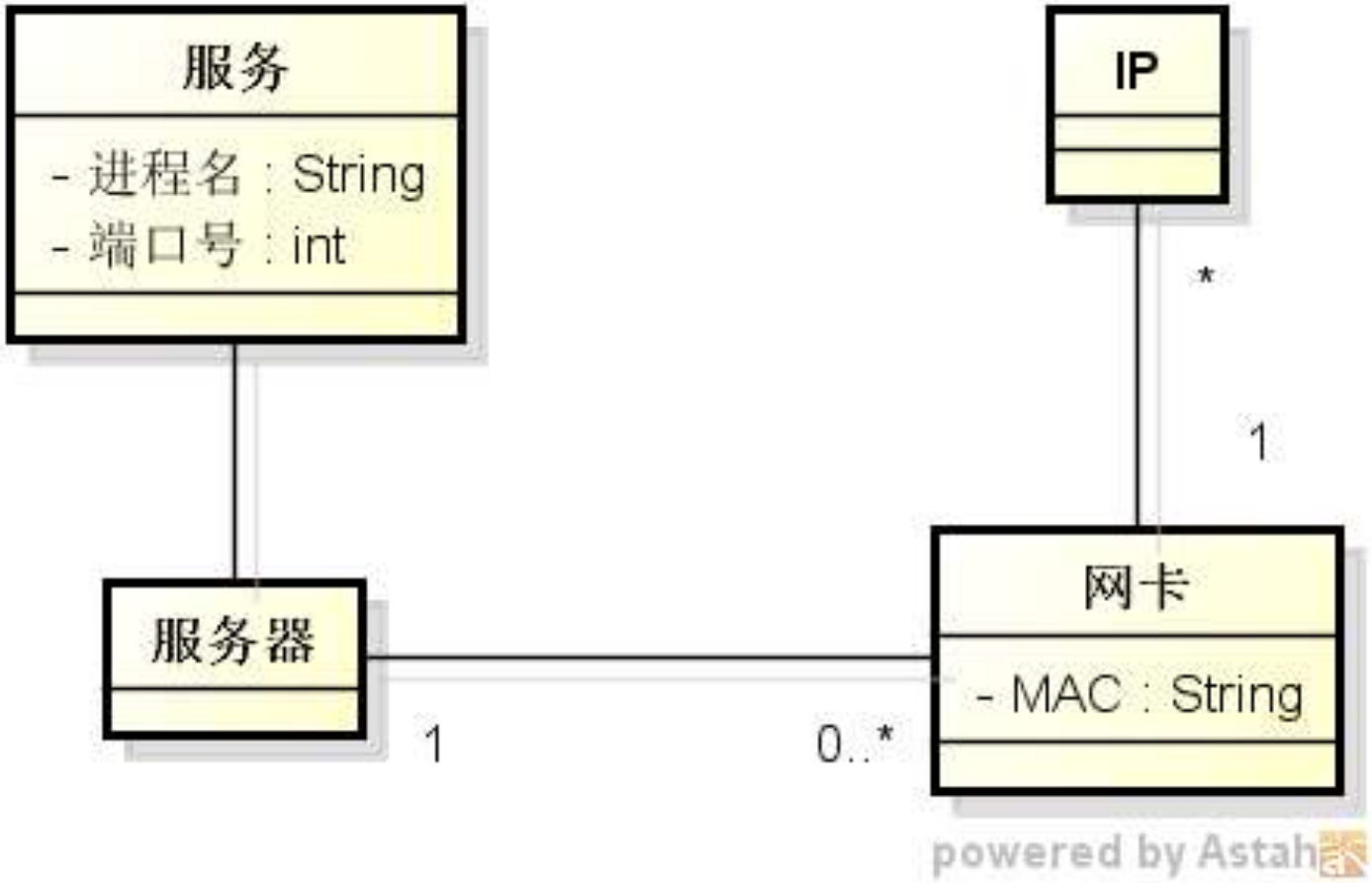
分层

- 模型定义为三个层次
  - 资产层🡐部分人工维护
    - 机房，IP，服务器，交换机，防火墙
    - 需要人工操作的部分
  - 服务层🡐全自动扫描
    - 面向In Service的服务器
    - 不区分物理机和虚拟机
  - 应用层🡐全自动扫描
    - 描述业务之间各Service的依赖关系
    - 面向业务，展现拓扑
- 为什么分层？
  - 隔离人工操作
  - 分层分阶段实现
  - 方便局部重构

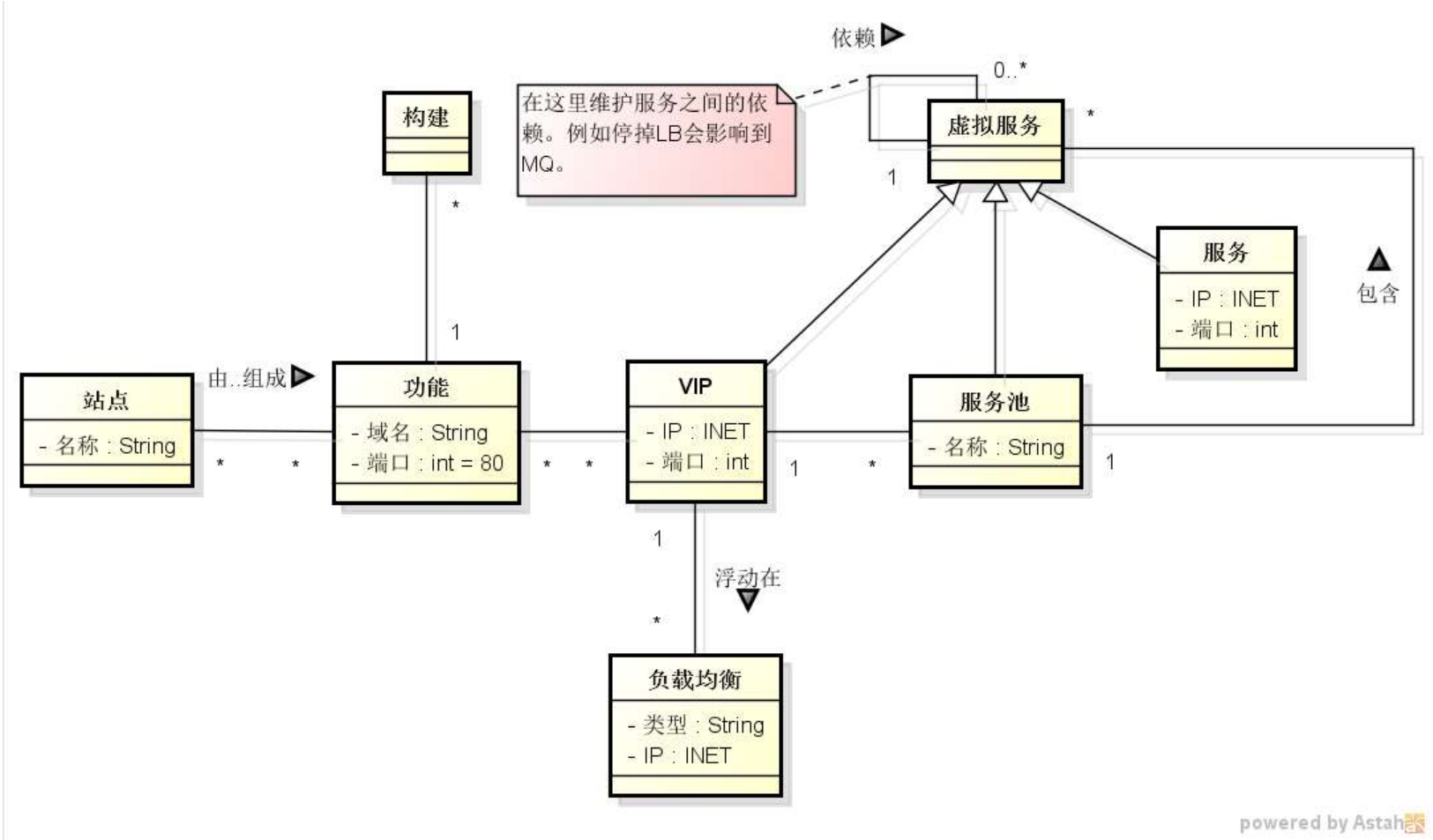
主要类图-资产层


子网段
0..*
网段
IP
1
*
*
*
1
1
机房
分配
资产
业务
*
1
*
*
*
1
服务器
网线连接
网络设备
*
*
物理机
虚拟机
路由器
交换机
防火墙
网卡
1
*
powered by Astah

服务层


服务
- 进程名 : String
- 端口号 : int
IP
*
1
服务器
网卡
- MAC : String
1
0..*
powered by Astah

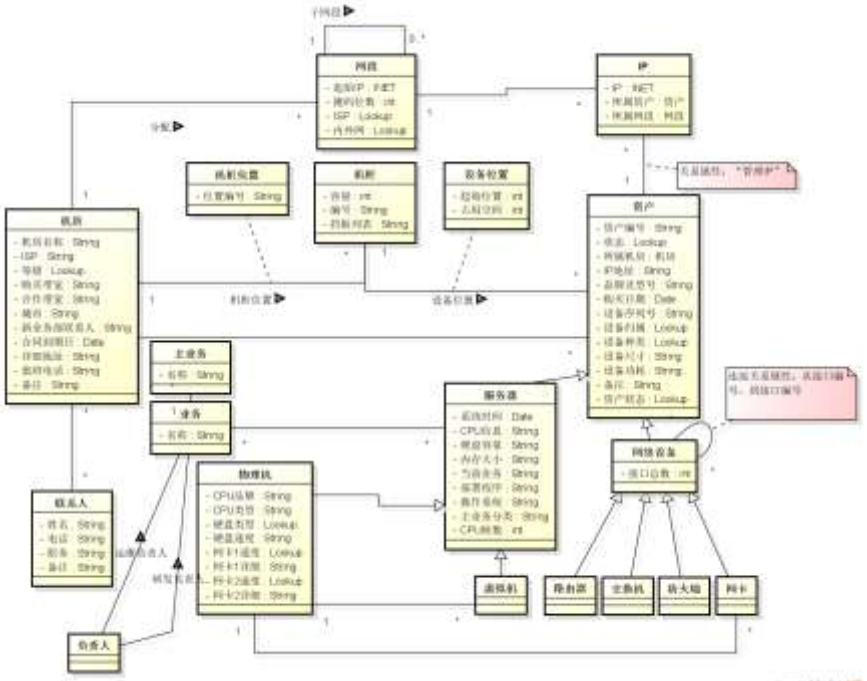
应用层

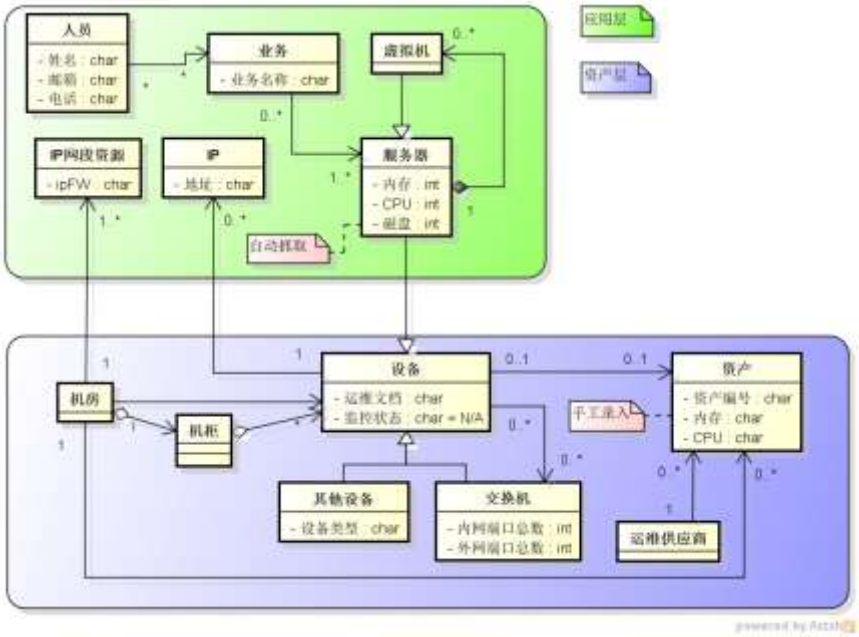

依赖
0..*
构建
在这里维护服务之间的依赖。例如停掉LB会影响到MQ。
虚拟服务
*
1
*
1
服务
- IP : INET
- 端口 : int
包含
站点
- 名称 : String
由..组成
功能
- 域名 : String
- 端口 : int = 80
VIP
- IP : INET
- 端口 : int
服务池
- 名称 : String
1
*
*
*
*
1
*
1
1
浮动在
*
负载均衡
- 类型 : String
- IP : INET
powered by Astah

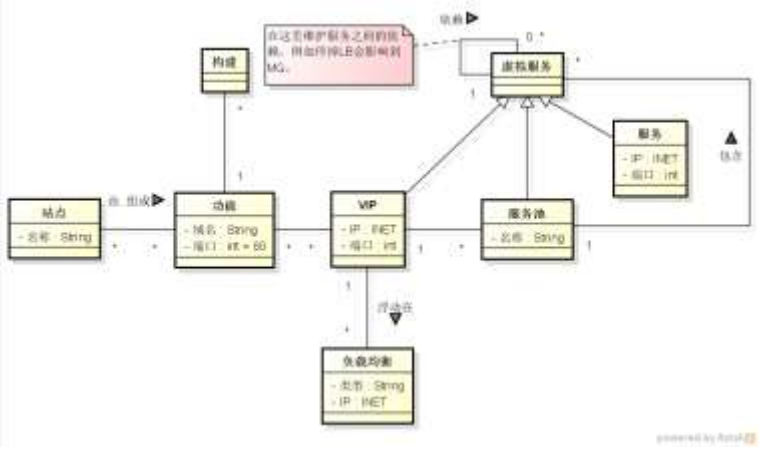
一些细节

- 保持层次的隔离
  - 不要使用强依赖
- 保持数据的纯净
  - 原则：有人工的地方就有错误!
  - 服务/应用层禁止人工修改
- 建模粒度问题
  - 没人关心的对象/属性不要引入
  - 用关系代替纯文本数据。
    - 201.45.13.1(2/4/6/14.7) ==》服务器 1:N IP

overview




人员
- 姓名 : char
- 邮箱 : char
- 电话 : char
业务
- 业务名称 : char
虚拟机
IP网段资源
- ipFW : char
IP
- 地址 : char
服务器
- 内存 : int
- CPU : int
- 磁盘 : int
自动抓取
应用层
资产层
机房
机柜
设备
- 运维文档 : char
- 监控状态 : char = N/A
手工录入
资产
- 资产编号 : char
- 内存 : char
- CPU : char
其他设备
- 设备类型 : char
交换机
- 内网端口总数 : int
- 外网端口总数 : int
运维供应商


虚拟服务
服务
站点
- 名称 : String
动态
- 域名 : String
- 端口 : int = 80
VIP
- IP : INET
- 端口 : int
服务池
- 名称 : String
负载均衡
- 类型 : String
- IP : INET

定义核心流程

- 静态模型定义完毕
- 开始考虑动态模型

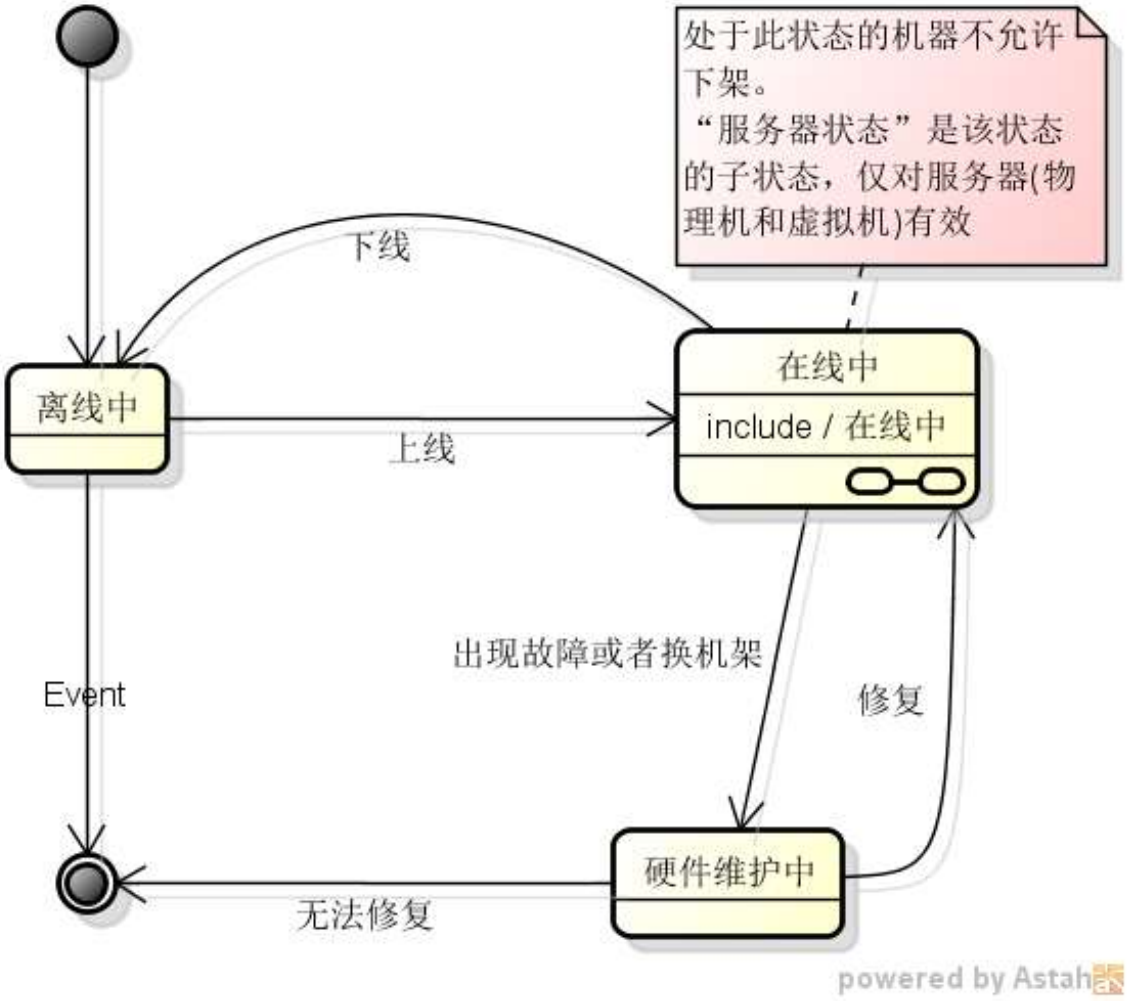
基本理念：无状态化

- 有状态：
  - 记录服务器的历史
    - 例子：
    - 下线的机器重新通电可启用历史业务
  - 弊端：懒得说了
- 无状态：
  - 下线就进资源池，无前生今世
  - 优点：池化，标准化，自动化

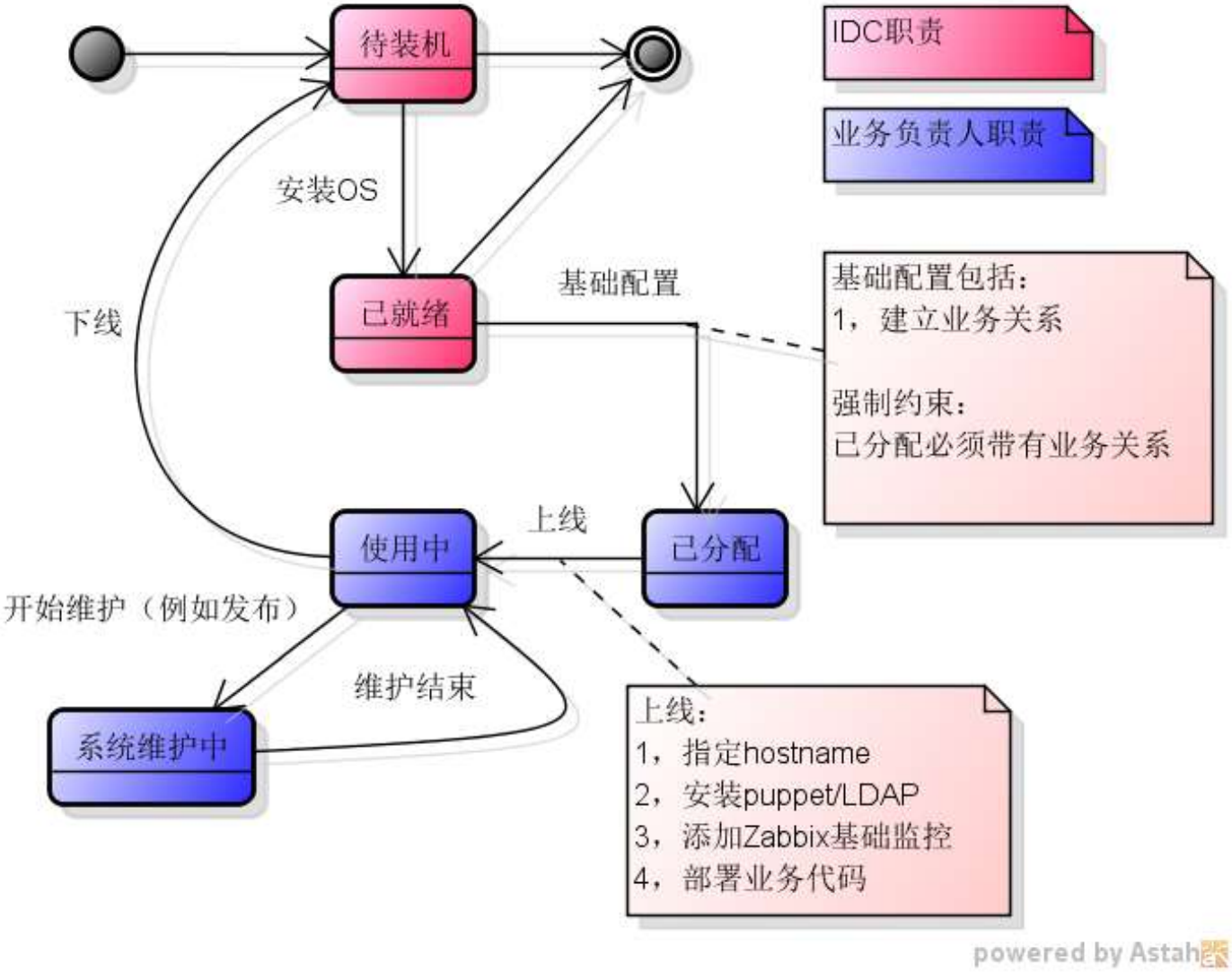
服务器生命期

- 两个状态
  - 资产状态 – IDC关心
    - 表述硬件（含交换机等）的生命期
    - 只有离线的资产才允许下架
  - 系统状态 – 业务负责人关心
    - 表述服务器从申请到回收的生命期

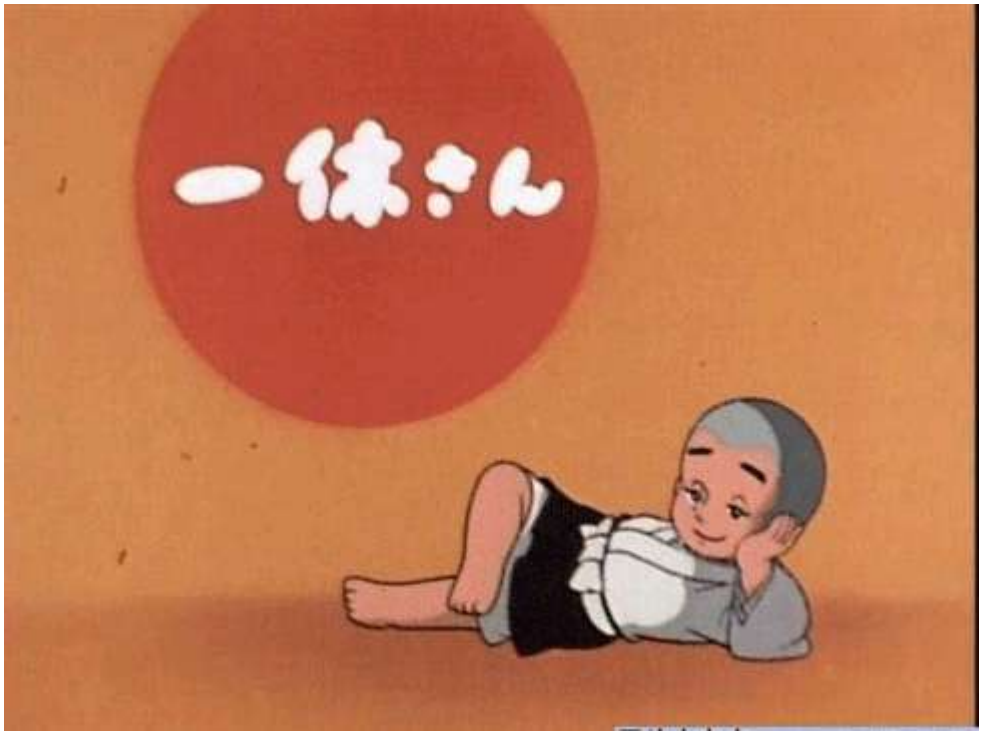
服务器的生命期-资产状态



服务器的生命期-系统状态



建模结束



开始实现

- 采用CMDBuild
- 优点
  - 版本持续更新，软件质量可靠
  - 对OO支持较好，能配置实现复杂模型
- 缺点
  - UI定制困难
  - 非开发背景的人使用感到费解

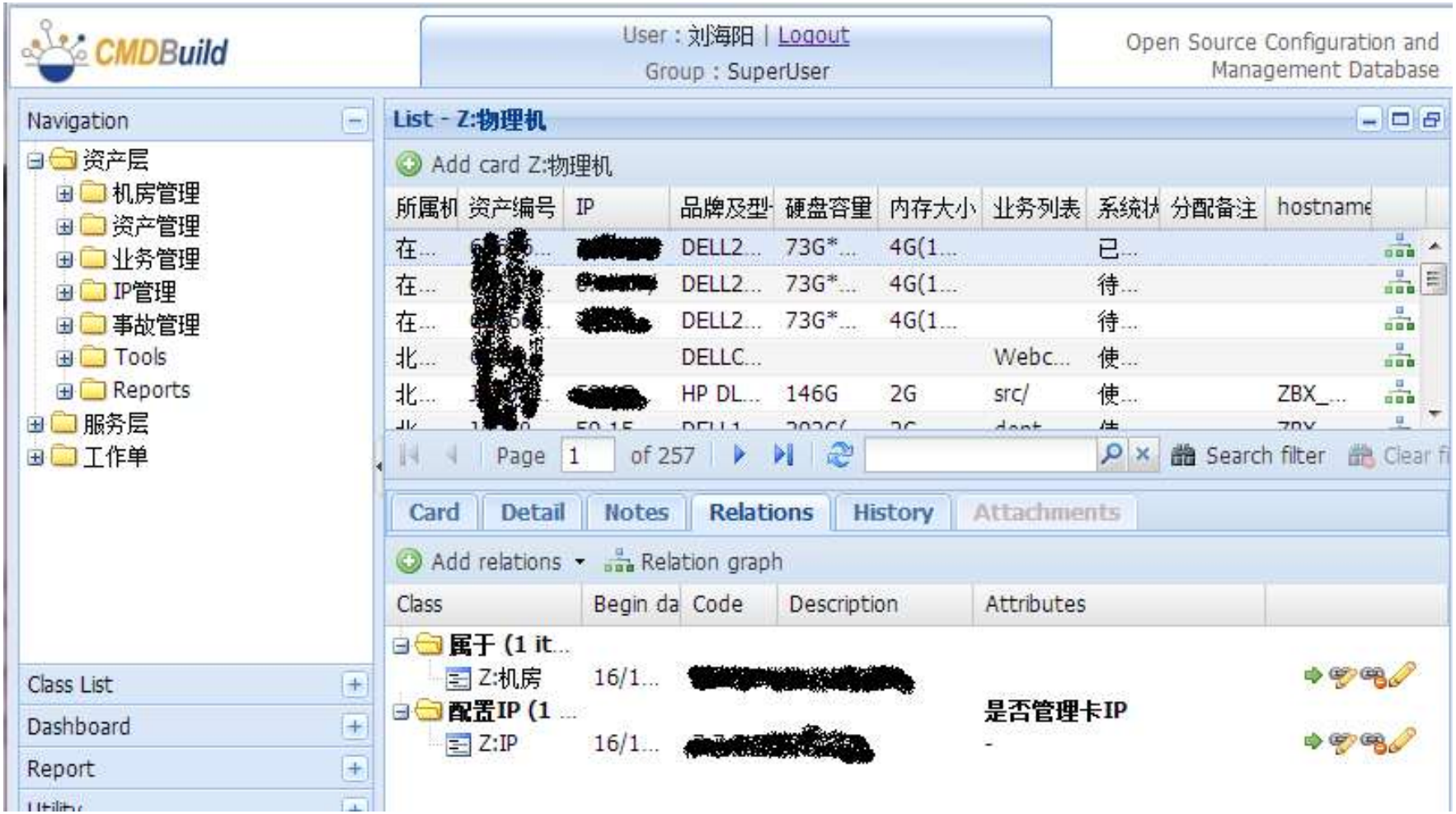
CMDBuild搭建

- 略，参阅官方文档

历史数据导入

- 坑爹啊，目前为止最让人沮丧的事情

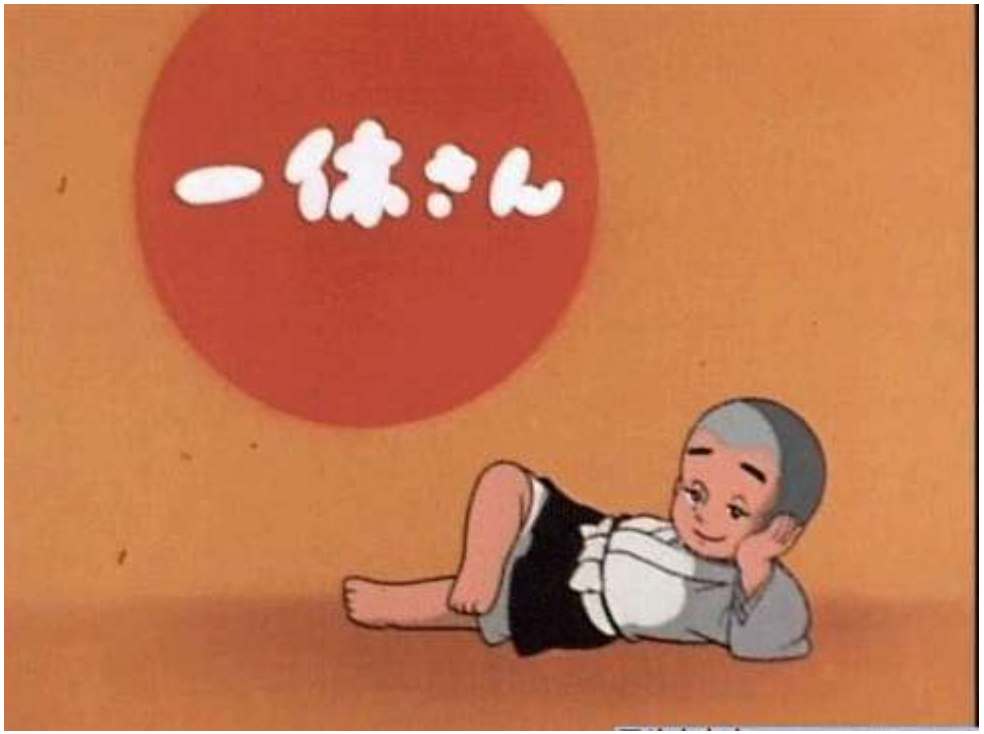
Hello CMDB


CMDBuild
User : 刘海阳 | Logout
Group : SuperUser
Open Source Configuration and Management Database
Navigation
资产层
机房管理
资产管理
业务管理
IP管理
事故管理
Tools
Reports
服务层
工作单
Class List
Dashboard
Report
List - Z:物理机
Add card Z:物理机
所属机
资产编号
IP
品牌及型
硬盘容量
内存大小
业务列表
系统状
分配备注
hostname
在...
DELL2...
73G*...
4G(1...
已...
待...
北...
DELLC...
Webc...
使...
HP DL...
146G
2G
src/
ZBX_...
Page 1 of 257
Search filter
Card
Detail
Notes
Relations
History
Attachments
Add relations
Relation graph
Class
Begin da
Code
Description
Attributes
属于 (1 it...
Z:机房
16/1...
配置IP (1 ...
是否管理卡IP
Z:IP
16/1...
-

CMDBuild二次开发

- API
  - Python API
  - Restful API
  - JSONP API
- UI
  - ExtJS+JSONP
- PGSQL Trigger
- Report
  - Jasper Report

CMDB建设结束



CMDB中心化

理念

- CMDB作为所有配置变更的入口
- 人脑-决策/电脑-执行
- 决策，执行，反馈

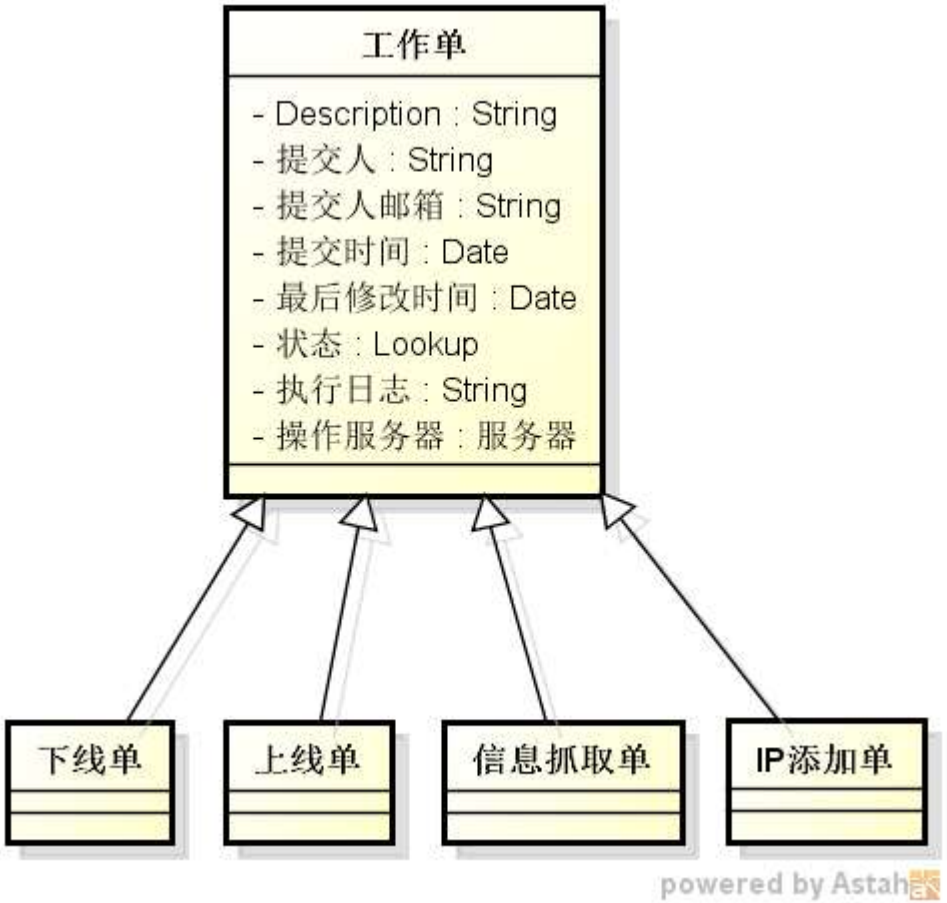
决策，执行，反馈

- 例子：
  - 在A服务器上配置IP 1.1.1.1
- CMDB：
  - 在CMDB中人工配置“决策”：A增加IP1.1.1.1
  - CMDB自动“执行”此操作
  - 自动脚本把此操作的结果“反馈”回CMDB
  - CMDB判断“执行”是否符合“决策”

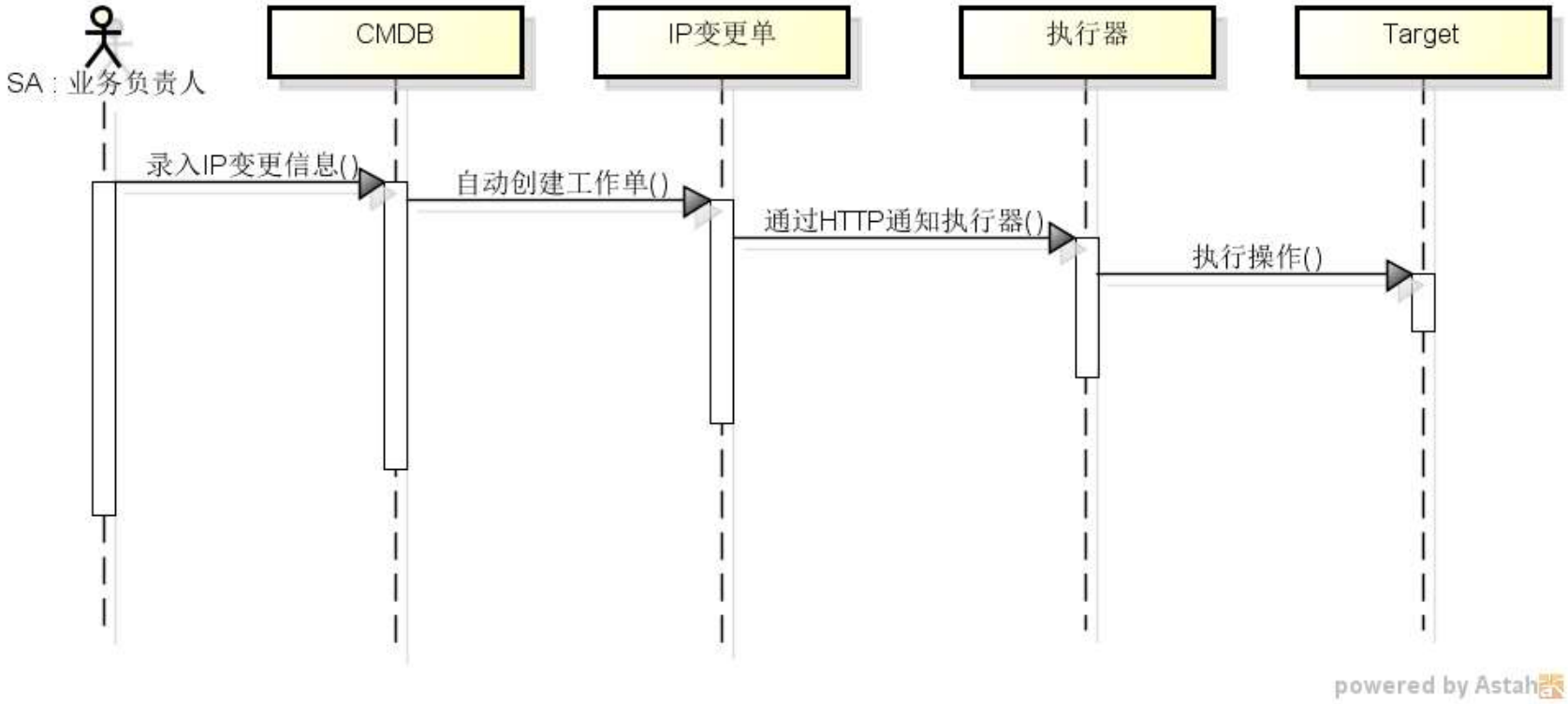
如何实现

- 工作单
  - 自定义的轻量级工作流
  - 跟踪/追溯操作

类图-工作单


工作单
- Description : String
- 提交人 : String
- 提交人邮箱 : String
- 提交时间 : Date
- 最后修改时间 : Date
- 状态 : Lookup
- 执行日志 : String
- 操作服务器 : 服务器
下线单
上线单
信息抓取单
IP添加单
powered by Astah

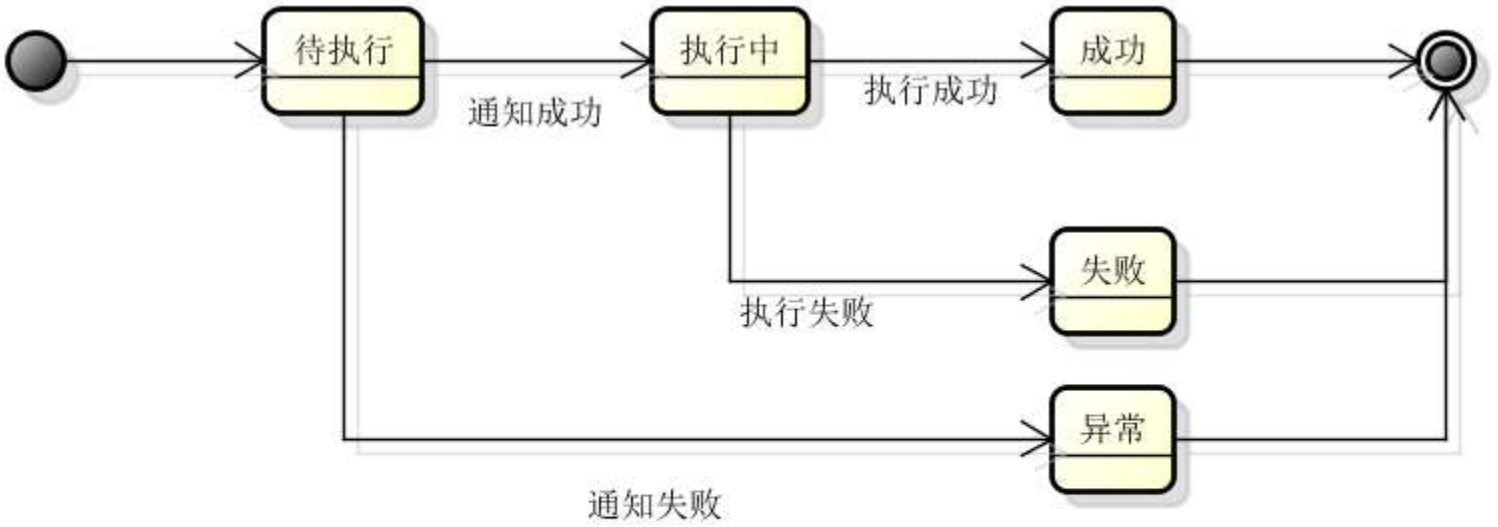
工作单时序图-例子


SA : 业务负责人
CMDB
IP变更单
执行器
Target
录入IP变更信息()
自动创建工作单()
通过HTTP通知执行器()
执行操作()
powered by Astah

描述

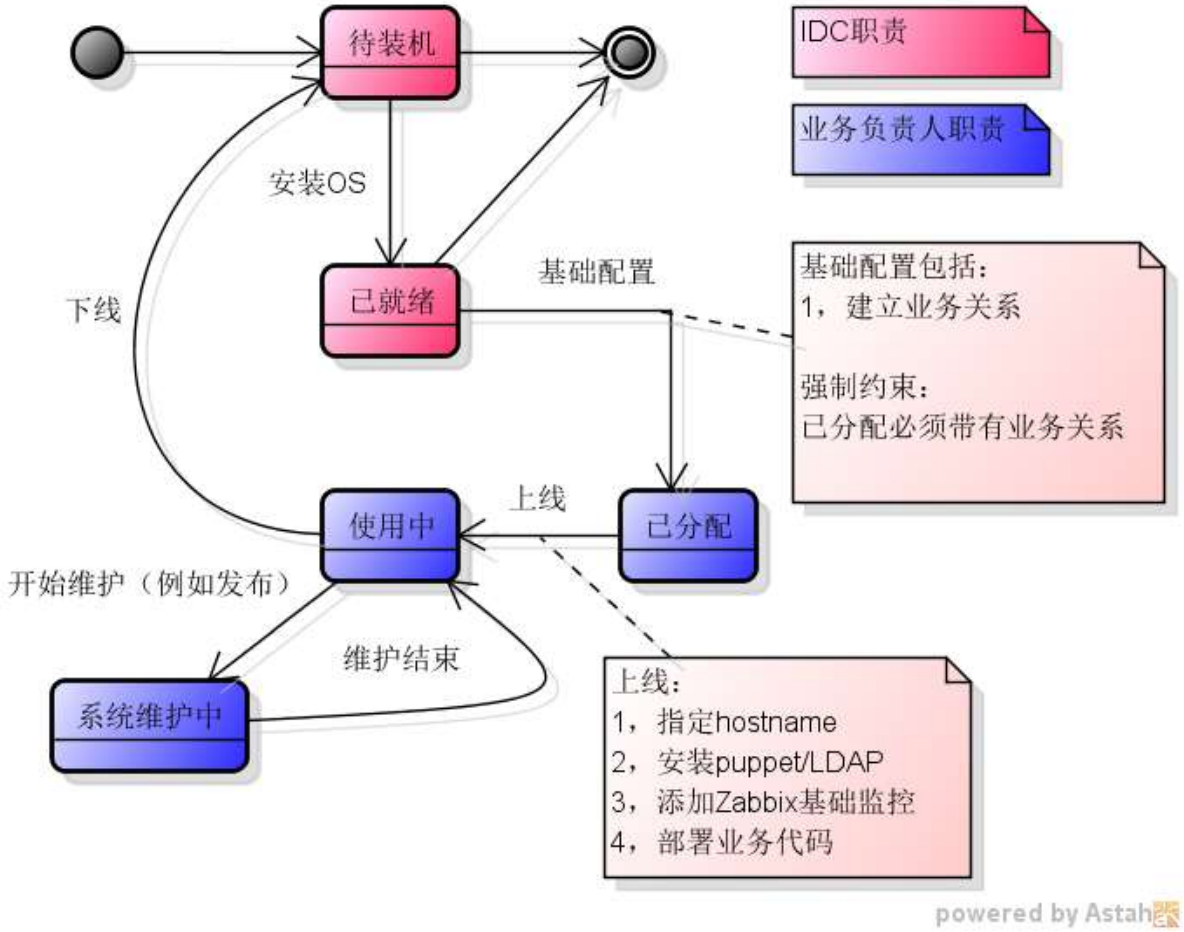
- 用户在CMDB UI上更新IP
- CMDB通过触发器创建工作单
- 工作单通知执行器(trigger)
- 执行器执行变更IP操作

工作单状态图



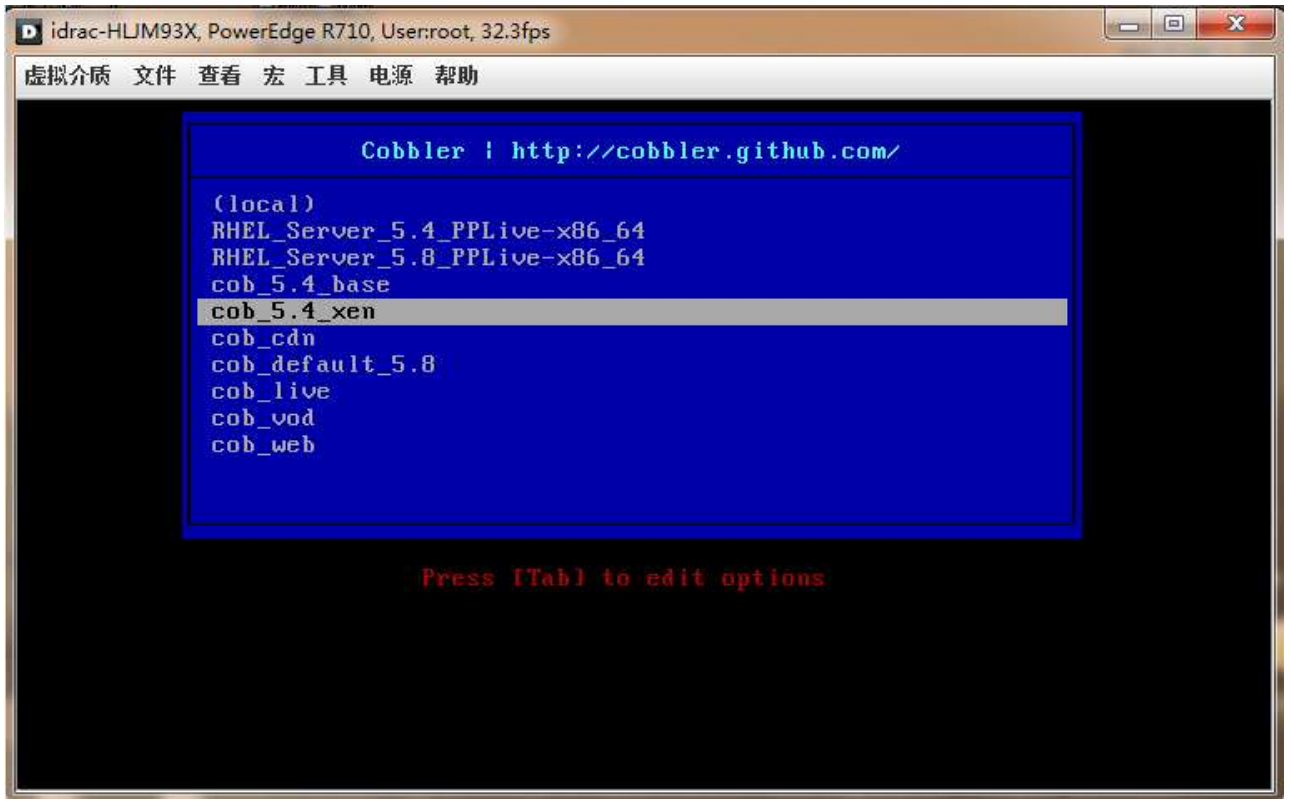
集成

- 回顾服务器生命期：


待装机
安装OS
已就绪
基础配置
下线
使用中
上线
已分配
开始维护（例如发布）
维护结束
系统维护中
IDC职责
业务负责人职责
基础配置包括：
1，建立业务关系
强制约束：
已分配必须带有业务关系
上线：
1，指定hostname
2，安装puppet/LDAP
3，添加Zabbix基础监控
4，部署业务代码
powered by Astah

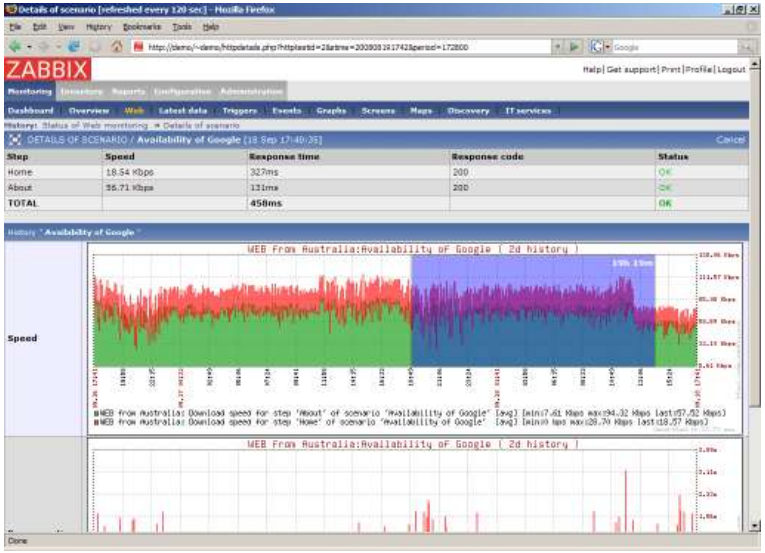
自动装机-Cobbler集成

- 使用工作单机制实现
  - “待装机”后通知自动装机系统执行装机



自动基础监控-Zabbix集成

- 采用工作单机制实现
  - 自动下线
    - 使用中==》待装机：通知Zabbix删除监控
  - 自动上线
    - 已分配==》使用中：通知Zabbix添加监控



自动业务监控-XXMon集成

- 我们有XXMon业务监控系统
  - 配置项来自CMDB Model中的“业务-监控项”关系
  - 业务负责人只需在CMDB中配置业务的“监控URL”则可自动添加业务监控

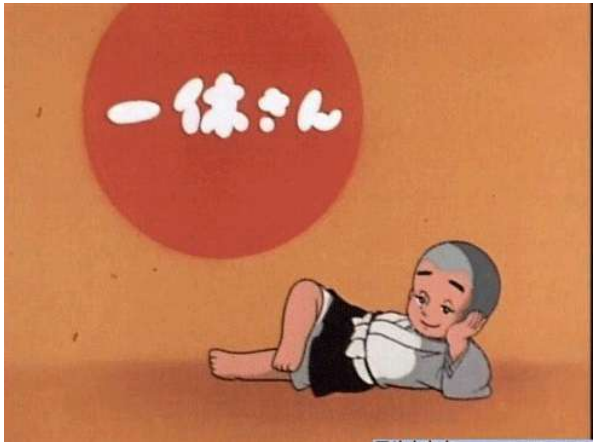
报警系统集成

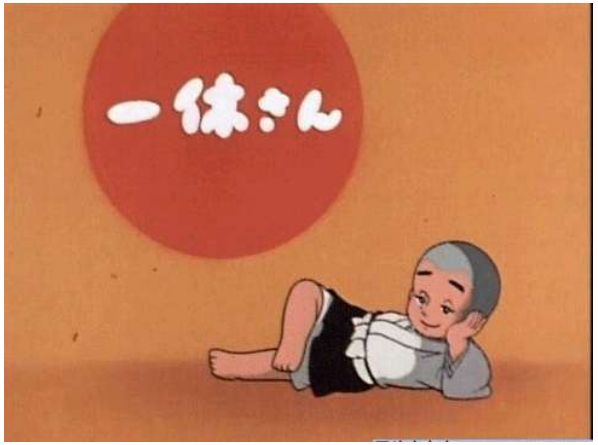
- 报警前检查服务器状态
  - “系统维护中”则停发报警

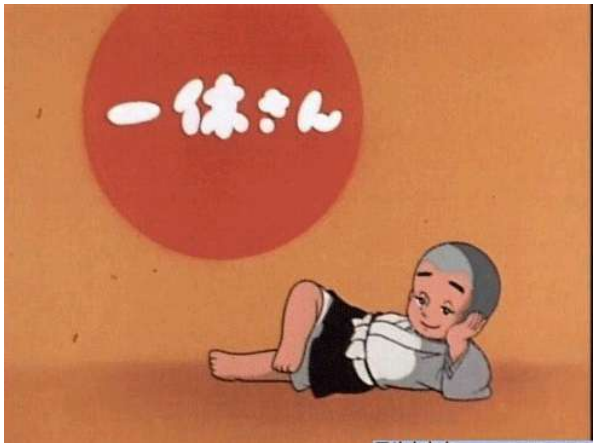
自动发布-CT集成

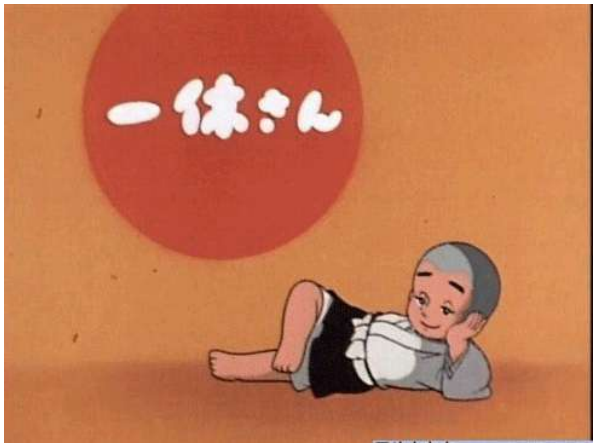
- 定制ControlTier
  - 通过业务-服务器关系获取发布列表
  - 实现灰度发布

CMDB中心化结束









值得一提的

- 领域模型是核心
  - 没有完美模型
  - 符合业务最重要
- 坚守理念
  - 用户访谈的同时推广理念
  - 改变旧流程/观念会有阻力 ←老板支持很重要
- Don’t make me think
  - 用模型固化隐式约定
    - 例：状态跳转约束
  - 提供工具辅助没有OO概念的同事使用CMDB
    - UI定制，隔离模型复杂度
    - 提供冗余(只读)字段：例如服务器的“业务列表”“IP列表”
  - 培训

References

- 本文重点关注CMDB建设，相关领域文档：
- 诸超：中型规模的网站架构运维
  - http://vdisk.weibo.com/s/crclO/1347622515
- 姚仁捷：Zabbix实践
  - http://vdisk.weibo.com/s/f29BA
- 朱宁：Cobbler实践
  - http://t.cn/zlWY1DF