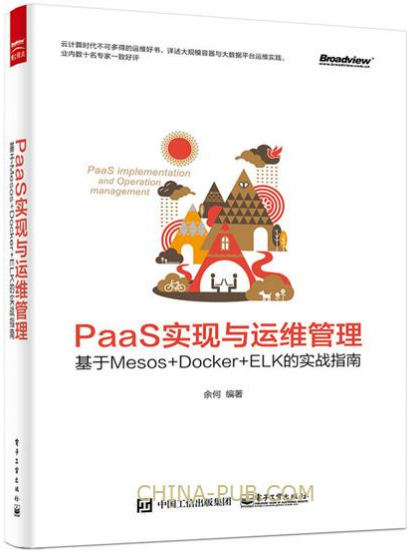
沉默的大多数

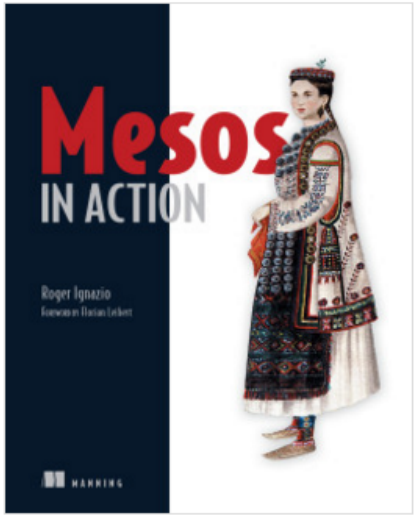
——中小企业运维互联网化之路

@众神的大师兄

About me

余何@众神的大师兄
运维者，重大故障现场
技术作者
《PaaS实现与运维管理》国内第一本PaaS著作
《Mesos实战》译者 敬请期待...
篮球爱好者
其他 ……





没有一把标准的尺子来衡量运维水平

很大一部分运维人员来自于中小企业

因此，全面的探讨下**运维，以及运维的“互联网化”**

议程

一、运维成长的道、术、法
二、运维环境的 A、B、C、D
三、传统运维 To 互联网
四、平安运维的**一些**实践经验

一、运维成长的道、术、法

运维&不完全定义

- 运维对象宽泛

数据中心
存储、计算、网络
操作系统
分布式中间件(平台中间件)
网络出口CDN

- 运维职能宽泛

管理 or 技术
运维 or 研发

运维&不完全定义

运维环境复杂：
与产品息息相关（vmvare、kvm）
与厂商息息相关（各种厂商、各种运营商）
与开发息息相关（面对开发，你有态度？）
与组织息息相关

运维&不完全定义

Who r u ?

系统工程师、网络工程师、运营商工程师、云工程师、

What r u doing ?

技术专家、管理专员

运维的道、术、法

道

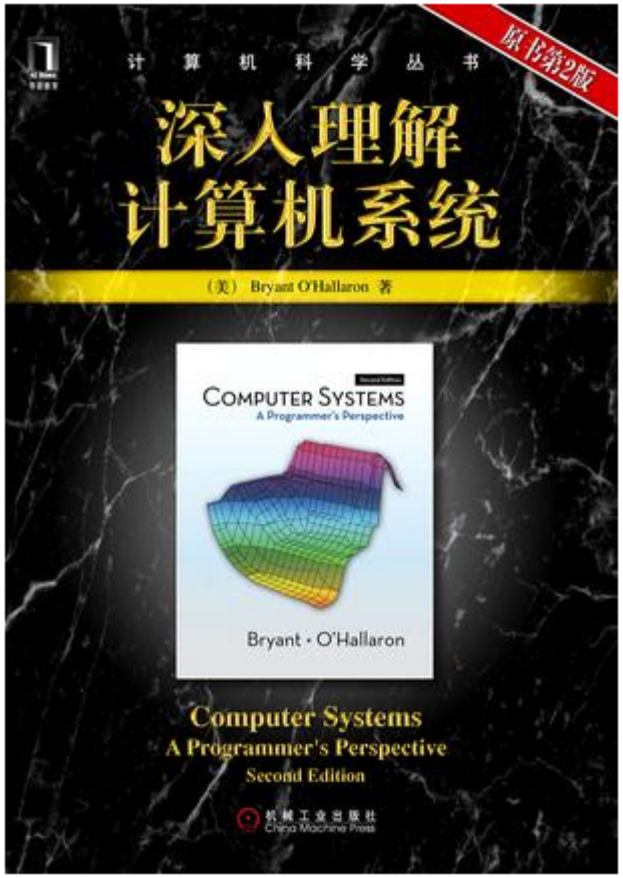
掌握基础原理，不变应万变

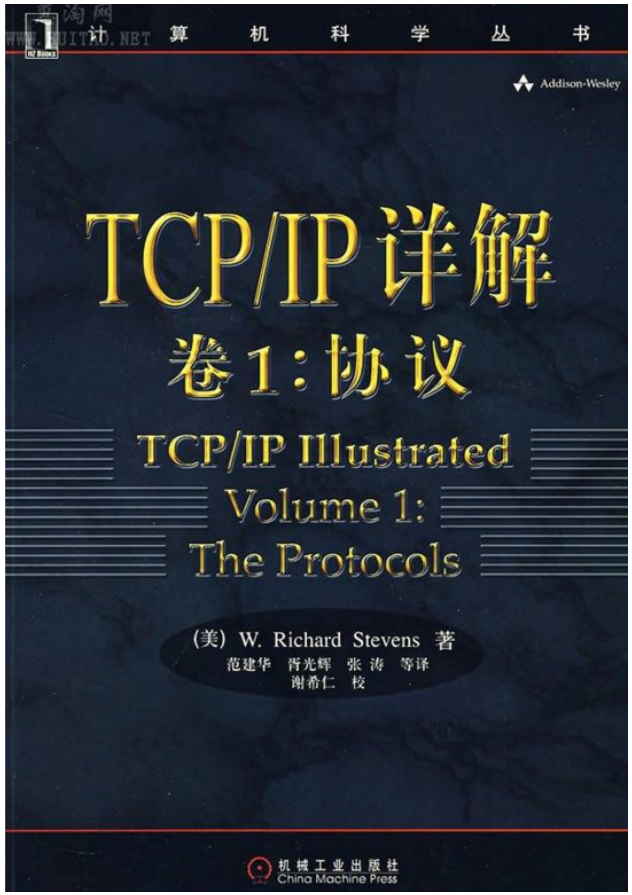
基础原理在哪里？不变的，很少变的就是基础原理

操作系统发行版在变，操作系统内核却稳定

网络设备型号层出不尽，以太网/tcp/ip/http协议却很稳定

道 ——大道隐于市而烂熟于心





术

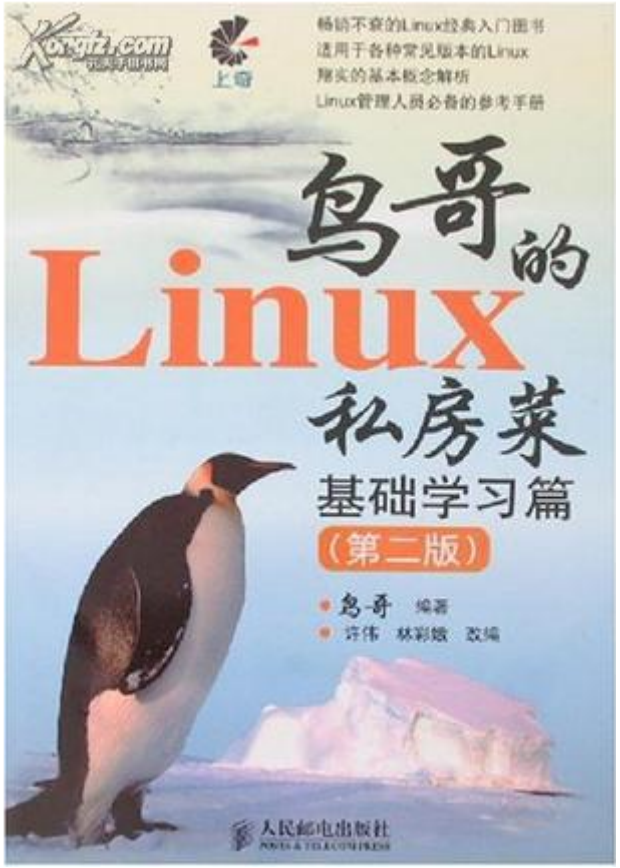
不别随意跟风，专注于一类

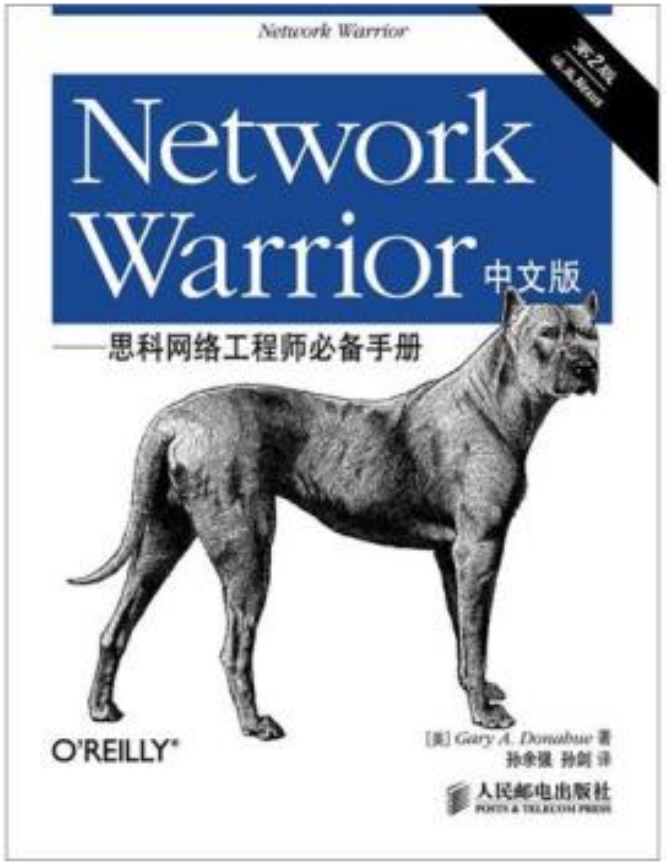
redhat、centos、ubuntu

vmvare、kvm

cisco、juniper

术 ——术者多变而不滥取之

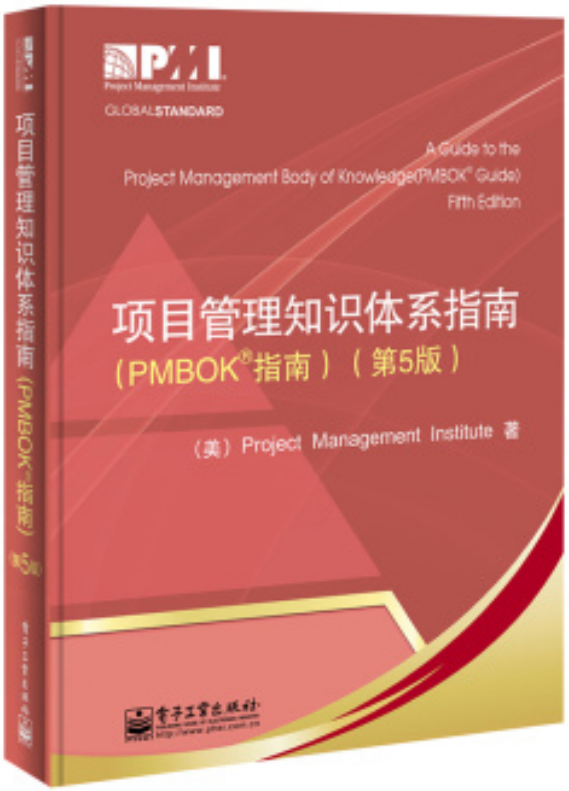

畅销不衰的Linux经典入门图书
适用于各种常见版本的Linux
翔实的基本概念解析
Linux管理人员必备的参考手册
鸟哥的
Linux
私房菜
基础学习篇
（第二版）
鸟哥 编著
许伟 林彩娥 改编
人民邮电出版社

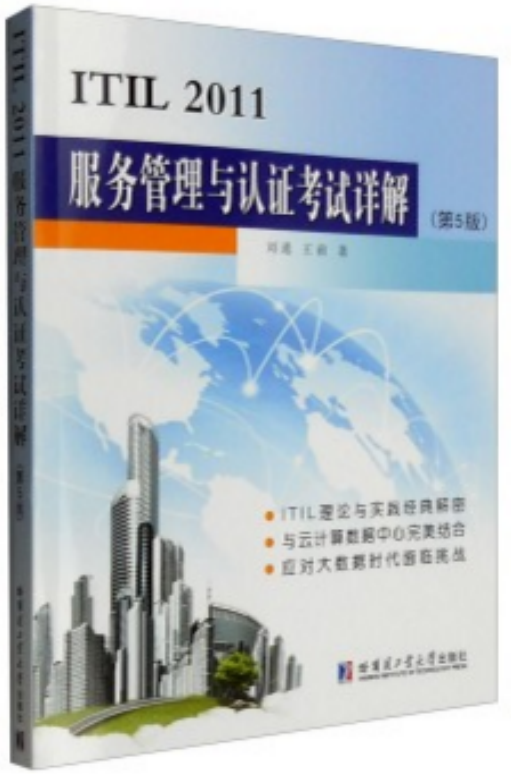

Network Warrior
Network Warrior
中文版
——思科网络工程师必备手册
O'REILLY
[美] Gary A. Donahue 著
人民邮电出版社

法

这个世界不仅仅只有道与术，还有法~

法——枯燥乏味而不得其解


PMI
Project Management Institute
GLOBAL STANDARD
A Guide to the
Project Management Body of Knowledge(PMBOK® Guide)
Fifth Edition
项目管理知识体系指南
(PMBOK®指南)(第5版)
(美) Project Management Institute 著
项目管理知识体系指南(PMBOK®指南)(第5版)


ITIL 2011
服务管理与认证考试详解 (第5版)
ITIL理论与实践经典解密
与云计算数据中心完美结合
应对大数据时代面临挑战

道、术、法，明白之后，下海

二、运维环境的A、B、C、D

A，运维&运营

运营：业务相关，内容运营、活动运营、用户运营，维持产品、服务生命力。互联网的运营属于IT，传统的运营属于业务。

运维：偏重技术，服务交付、事件处理、监控巡检，保障系统本身稳定性。

B，组织环境

小微组织：保姆式运维。所有与电脑相关的都与你有关。乱术而无道法，上升空间狭窄。作为一种业余兴趣是的聊以慰藉。

中小组织：创意式运维。抱团业务，承上启下，背锅之最多，受屈之最冤。业务驱动-道术法，管理重于技术。平台级工具解决不了或者说无需解决这些问题。

大型组织：平台级运维。业务解耦，平台服务，专注研发，术精道明而法严。

C，开发标准

无统一标准：各种OS，各种中间件，各种配置信息。
例如：中小企业传统应用，互联网游戏行业
自动化运维难以一步到位
抽象自动化运维平台，动态模块交由小团队定义。

标准架构组件：良好标准规范，从开发语言到架构组件。
例如：大型金融公司，互联网标准
自动化运维较容易实现
平台级服务与组件（分布式数据库、文件系统、负载均衡）

D，传统vs互联网业务

传统：ERP，柜面、**交易**系统，面向客户

特点：遗留系统、功能稳定、监管要求、影响面大，

互联网：2O（out of site），流量入口，**渠道**系统，面向用户

如何将用户转化为客户。

特点：渠道类，新系统、需求多变、版本频繁、故障影响可控（分散）

在截然不同的业务场景下产生了对运维不同的要求，传统、互联网运维

ITIL vs Automation

运维，共性

系统上线鼓掌的是开发，应用宕机背锅的是运维
不出问题的运维是好运维，好到老板不知道你存在

业务运维
平台运维
领域专家

互联网业务给运维带来的变化

传统运维方法已无法满足互联网业务飞速变化要求

互联网应用在传统环境中快不起来

传统应用的稳定性被互联网打破

三、传统运维 To 互联网

Step1：识别渠道

互联网化的是业务渠道变化

让渠道飞起来，让核心固然金汤

So.. 互联网化的应用是没包袱的

http://www.jianshu.com/p/34c0d4ea0072

民主共和与君主立宪@余何

Step2：主数据

主数据是新启渠道系统的关键
用来描述企业核心业务实体的数据。
客户、合作伙伴、员工、产品、物料单...
具有高业务价值的、跨越各个业务部门
被重复使用的，存在于多个异构系统中

Step3：隔离核心

1. Restful API 提供原子服务 **隔离**
2. Message Queue 实现异步解耦 **防护**

核心限速是受控的

Step4：Cloud-Native App

应用是否适合在云上运行(cloud-native App)？

Factor12规范

平安APP8原则

进程无状态化，session保存在share pool中

日志是流式事件，而不是文件

去中心化，管理进程一次性运行

在复杂的事物上试图通过不做任何改变的将其管理标准化，实际上是另一个过程复杂化的开始

如果开发不接受遗留系统的任何改变，互联网运维将变成“伪”互联网运维。

Step5：Weaken ITIL by Automation

推荐-高效运维-“舍本求末的运维自动化技术热潮”。

Shell、Python编程能力是基础

Ansible 批量管理

Docker 计算单元打包

合适的工具比选择无所不能的工具更重要

四、平安运维的一些实践经验

复杂度

多种类中间件及其海量的管理复杂度

Weblogic、tomcat、apache、nginx、keepalived……

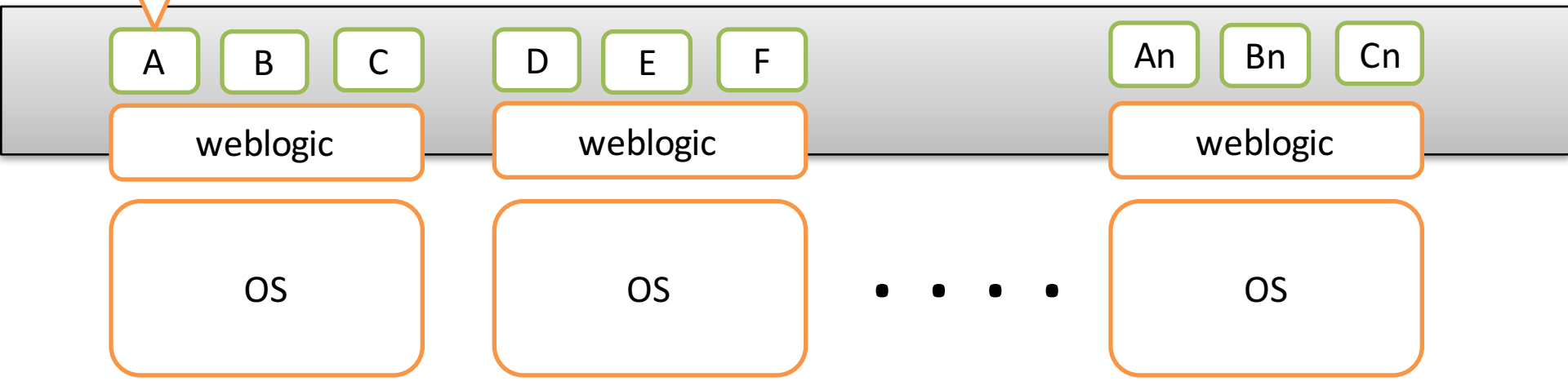
无数种中间件（介质）

2w+ instance（进程）

instance

Process different:
Logfile
Config
Business logic
Ip address
Domain name
Load banlace
Fw policy
Securiy

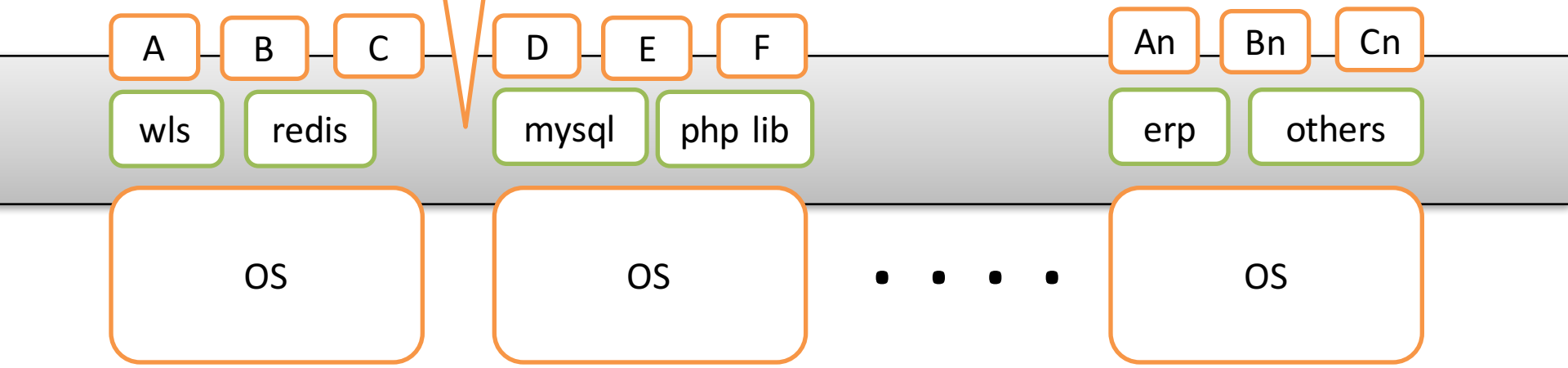
遵循软件的配置规范，上承应用、下接软件，将两者融合进来。
未启动、运行前，规范下的属性配置文件
规范易变而多样，具有不可完全预知
启动后、运行时，独立的进程或进程组



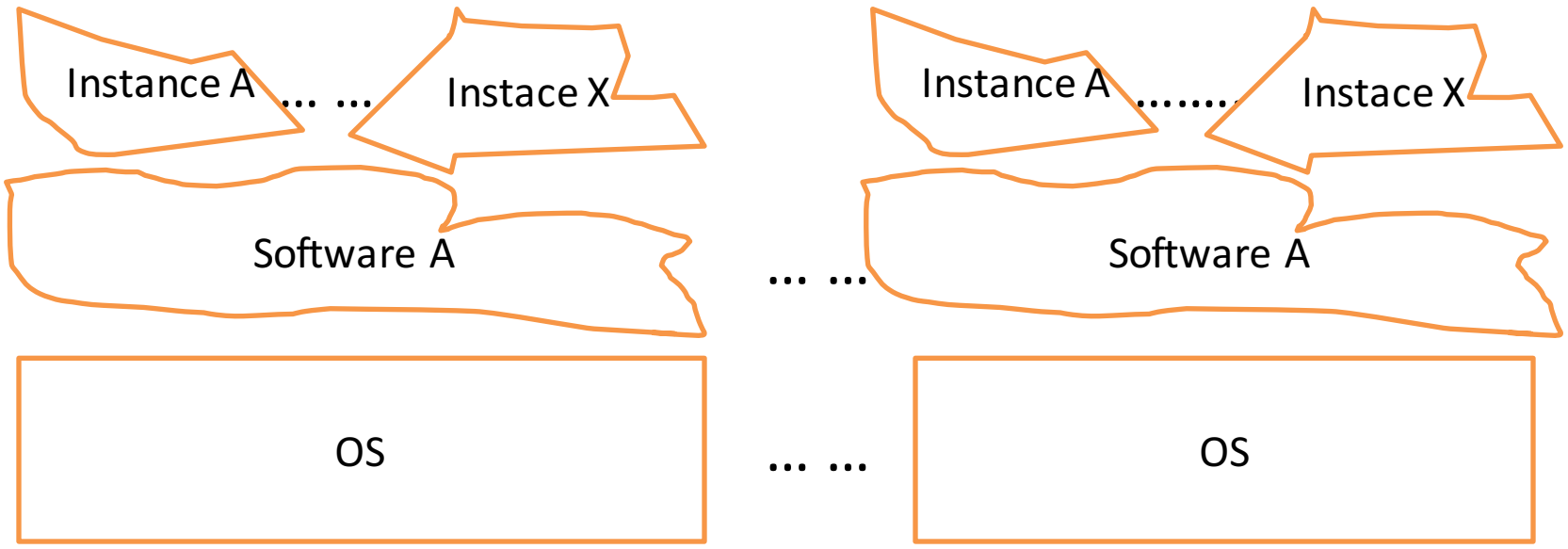
middleware

Software different:
A need tomcat 7.0
B need weblogic
C need other thing
D install MQ
E install other C lib
……

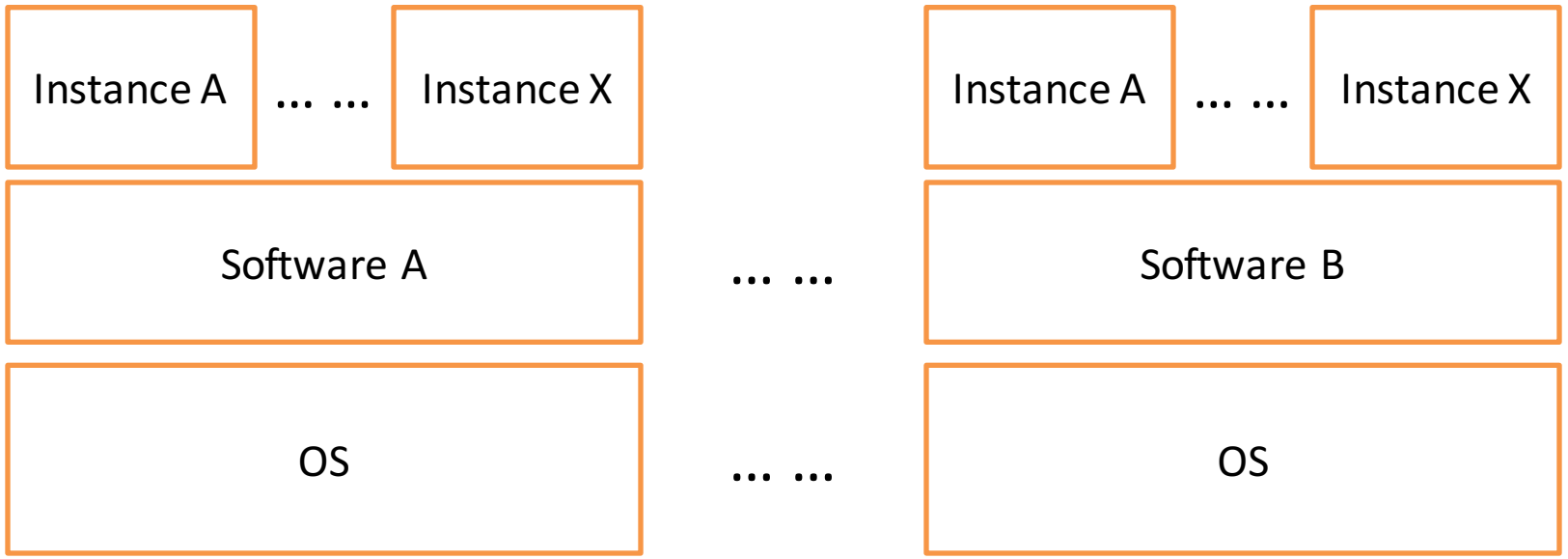
软件或强或弱的依赖于底层操作系统库文件
不同软件对库版本有不同要求，同一个OS出现冲突
软件安装无法简单认同为文件复制，看不到“安装”做了什么
软件类型太多、安装步骤太多、一次安装，多次使用，还不冲突
怎么做？



2014年前 方案

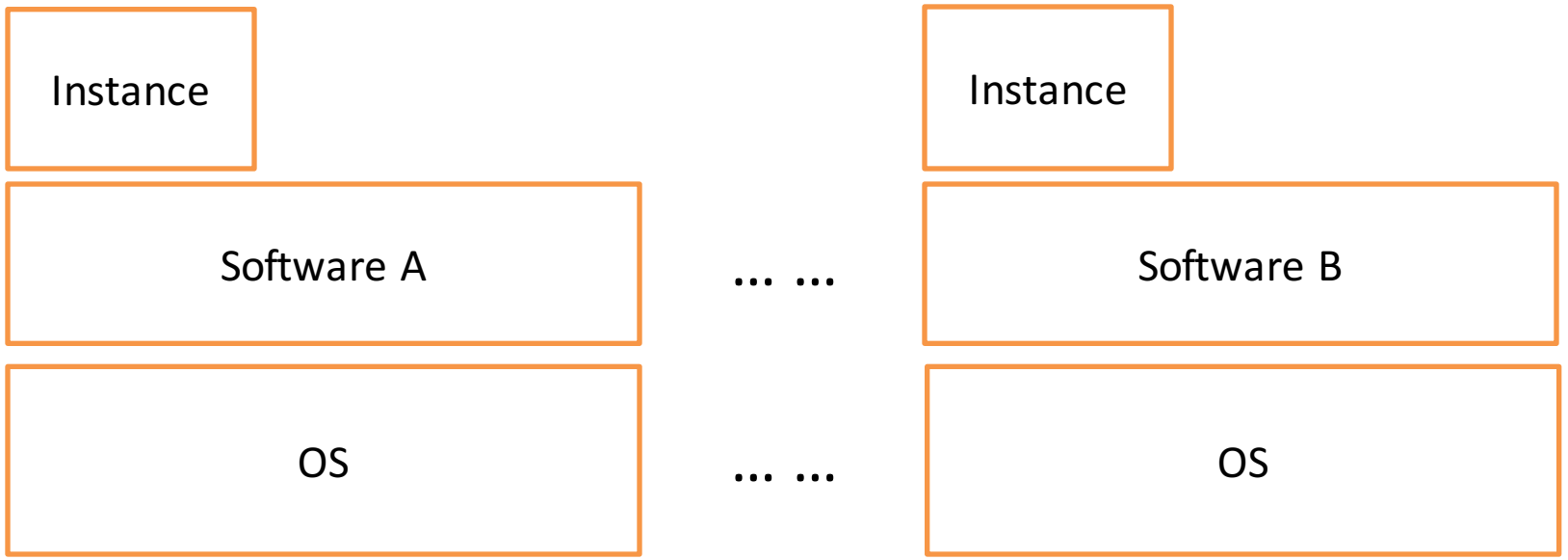

Instance A
... ...
Instace X
Software A
OS
... ...
Instance A
.....
Instace X
Software A
OS

统一软件使用标准，控制种类 weblogic、tomcat、apache、nginx



统一软件使用标准，控制种类（ weblogic、tomcat、apache、nginx ）

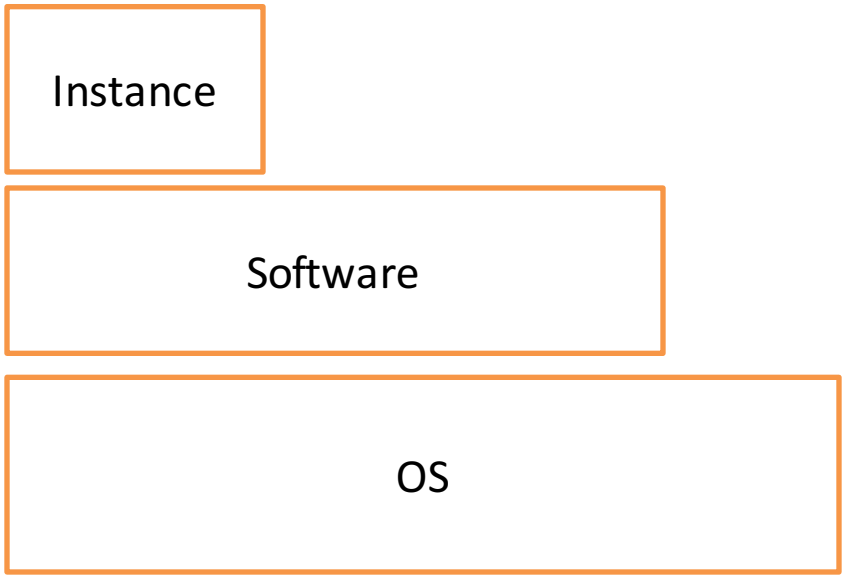
统一实例使用标准，管理方式一致，提高运维管理效率



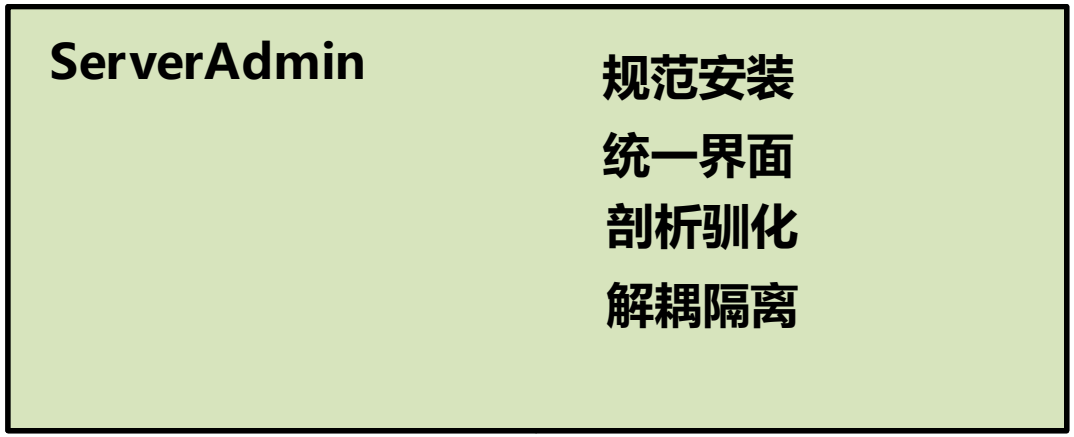
统一软件使用标准，控制种类（ weblogic、tomcat、apache、nginx）

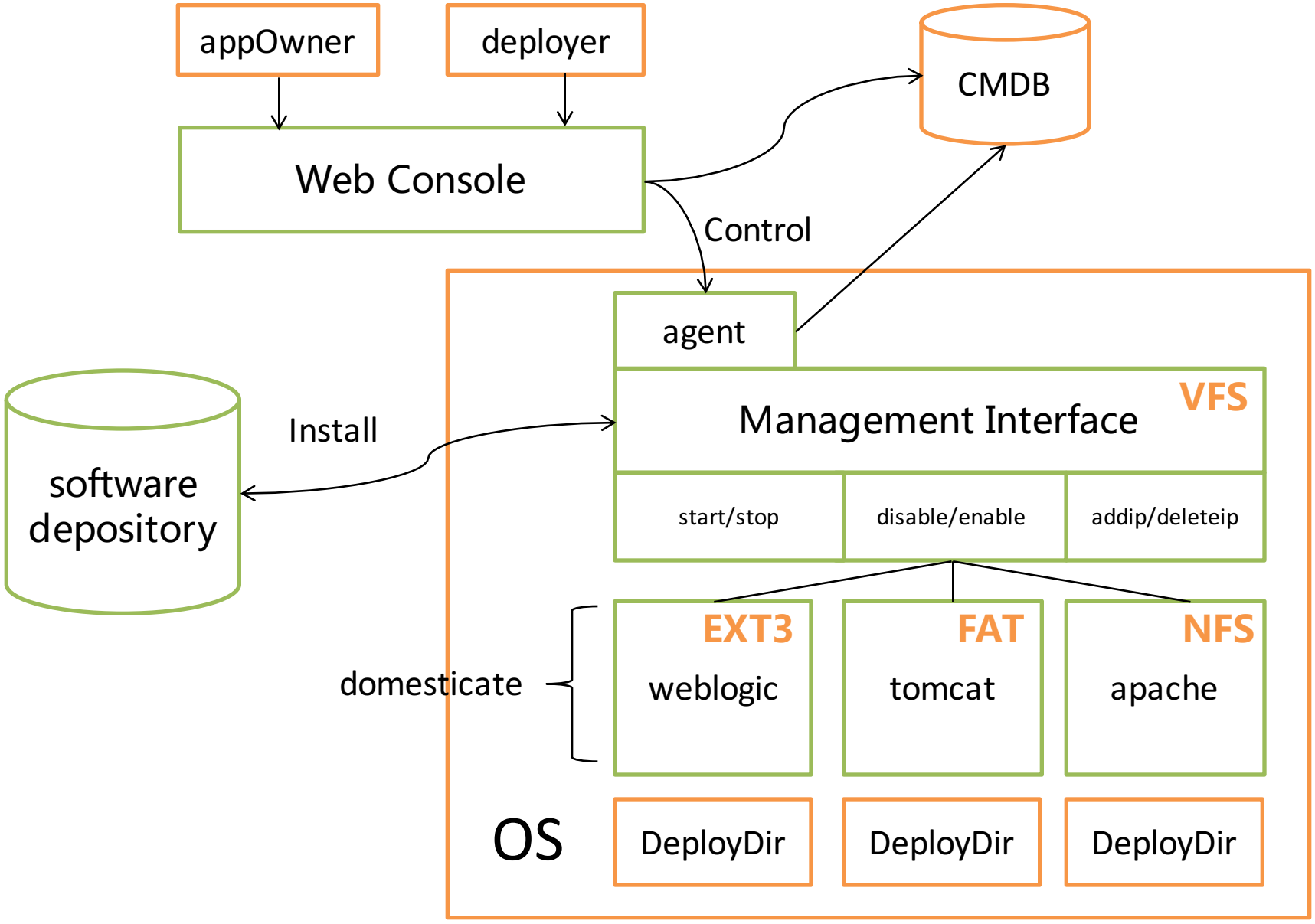
统一实例使用标准，管理方式一致，提高运维管理效率

采用大物理机器，避免虚拟化，提高资源使用率，降低底层组件维护成本



ServerAdmin




appOwner
deployer
CMDB
Web Console
Control
agent
VFS
Management Interface
Install
software depository
start/stop
disable/enable
addip/deleteip
domesticate
EXT3
weblogic
FAT
tomcat
NFS
apache
OS
DeployDir
DeployDir
DeployDir

ServerAdmin-规范安装

仅仅是软件介质

规范安装目录
建立软件仓库
一次编译多次使用(green)
统一安装界面

--prefix -g (that's is import)

--x86 & x86_64

--independent

请相信，所有依赖全部在独立目录中

ServerAdmin-统一界面

定义接口

定义管理界面
规范调用API(适用于一切)
(环境变量、执行程序)
规范通用文件、目录
(配置、日志、应用)
审计方式

--Return value

--Input & output

--Object Oriented,OO 封装、继承与多态

所有软件管理方式一致

ServerAdmin-剖析驯化

具体实现

启停脚本
执行文件
环境变量
引用库(--prefix独立)
配置目录
日志目录
应用目录

配置、使用、裸跑、lsof分析、拆分、link

ServerAdmin-部署发布

开放目录
制定规则

ServerAdmin

- **操作系统层（解耦、隔离）**

IP 跟着应用走

CPU 资源受控

IO 内存替代磁盘

NET/MEM 预知，未遇

ServerAdmin-IP 跟着应用走

Listen() 主动方式配置解决

connect() 无法配置，hack systemCall

Default Route 降级处理，插入模块，解决本机访问本机问题

ServerAdmin-CPU 资源受控

Cpulimit 控制进程CPU使用率

ServerAdmin-IO 内存替代磁盘

Override fsync systemCall

之后。。牺牲了并不重要的数据一致性,解决IO问题

内存即磁盘，压力测试报告印证，在930生产正式环境业务高峰期，IO瓶颈不再是磁盘，很可能是应用程序的开源日志框架。

2014后，下一站 Docker

Docker轻易解决ServerAdmin驯化过程
Mesos让计算资源得到复用（价值无法估量）

当然，偏地是坑

做什么

4A

Anything is Container
Assign by Scheduler
Automate use API
Analysis through PALog

Demo


PADIS
首页
应用创建
应用管理
文档
支持
登陆
注册
Padis平安分布式平台
一站式、全流程APP平台，支持APP生命周期管理
科技创新、自主研发，ITIL流程的敏捷革新者
1.选择App
2.选择租户
3.拓扑图
应用名：
应用镜像：
Weblogic
Tomcat
Nginx
App
自定义App
数量：2
服务端口：7001
Cpus：0.2
Mems：1536
监控协议：HTTP
监控路径：/f5monweb

什么是APP8

互联网兴起的时代，APP作为一种服务发布，即APP as a Service；APP8是将APP构建成Service的方法论。遵循APP8规则的APP在Padis平台上方可如鱼得水。

APP即纯粹逻辑计算单元，与下层MW、OS解耦，单元间同构无差异。

APP享受到云平台带来的弹性扩展、快速发布优势，亦即APP转化为Service。

APP8（1）

1.　去中心

APP由单个或多个Process组成，其中没有独立于业务逻辑的中心管理进程。管理类工作通过配置文件同构或一次性配置动作解决。

2. 配置

配置信息不含有底层资源信息，例如主机名、IP；配置信息按环境类型（STG/PRD)管理，拒绝嵌入代码。容器配置与容器介质分离，跟着应用走。

3. 日志流

日志输出是一条一条的消息输出流，而不是写文件；日志输出尽量与文件系统解耦，可重定向到远程日志管理服务中。

APP8（2）

4. 无状态
APP尽量做到无状态化，将状态信息记录到back-service中，例如将session信息记录到集中的cache server中

5. TCP服务
APP进程对外提供服务的方式统一为IP绑定PORT的TCP单播方式。拒绝其他交互方式，包括写文件、共享内存等等。

6. 依赖
精确的定义APP每一项依赖；库文件依赖打包成一个独立单元；back-service依赖信息保存在配置文件中。

APP8（3）

7. 环境一致

Dev、Stg、Prd环境的full-stack保持最大一致性

8. 后端服务

APP的后端服务作为一种资源引入，这些后端服务包括数据库、缓存、消息队列，后端服务的引用保存在环境配置文件中

谢谢，超时不怪我