ppt在最后

###############

Agenda:

     网站运维体系

     网站运维常见问题

     理想的运维体系

          标准化：

          技术组件化、&服务化，人员专业化

          自动化运维

          数据化运维

          监控

          安全

     About DevOps

###############

大型网站的定义：

     200-5000服务器

     5-50运维人员

     组织结构：

          系统运维(IDC/network/server)

          应用运维(web/CDN/DB/Game)

          数据库

          运维平台开发

          监控值班团队

          安全

          流程

运维的要求：

     稳定

     低成本

     快速响应

**中小网站运维都有些什么问题？**

     人员：

          不稳定 总体比较初级 人员缺乏

     规划：

          建设快 CDN与核心网问题多

          应用部署规范少

          机房太多，利用率低，跨机房调用严重

     数据库：

          跑在虚拟机，外网稳定性差

          网络**大量单点**

          系统大量驱动问题

          应用：代码问题和部署问题一样多

     监控：

          没有有效的监控，没有容量分析，想怎么干怎么干

          应用各自为政

     几乎全部手工操作

          装机

          应用上机

          应用升级、部署

     安全问题突出

**How应该如何应对：**

     标准化&规范化：

          kickstart/puppet/LDAP/DNS/zabbix

     自动化：

          装机；发布；监控

     监控：

          系统监控；应用监控；业务监控

     服务化：

          搭建运维基础架构，应用基础架构

     数据化运维：

          CMDB；日志分析；容量规划

     网站安全：

          网络，系统，应用

     推动架构优化、简化

**具体如何操作：**

     标准化&规范化：

          统一了才不会出现诡异的配置问题，才能做自动化部署、分析、排障

**统一规范化-Dev**

               统一公司层面运维支持技术开发平台

                    控制研发技术使用

                    不是想用什么技术就用什么技术，不要随便引入新技术

                    简化技术平台

                    允许适当引入新技术

                    维护一个稳定版，一个推进版

**统一规范化-Ops**

**操作系统：**

                    OS kickstart全部从Cobbler安装，取消光盘安装

                    硬件规格尽量统一，几个模板

                    OS内核、package一致

**Puppet控制系统和CDN应用**

                    服务器命名

                    统一LB配置

**应用部署模式：**

                    各类应用基础架构

                         日志格式，路径

                         空间管理，日志删除

                         参数配置文件

                         java、php、MySQL、Redis安装模板

                    指导方针：

                         IDC 考虑；ip分配：默认网内

                         LB共用？

                         DB read高可用，写不管？

                         cache考虑

                         logging consideration

     服务化：

          为什么要服务化：避免重复开发，加快部署；更好的积累经验，用精技术，不用每个业务有自己的solution；避免单点复用服务

**基础运维基础组件：**

                         kickstart/cobbler/git/puppet/LDAP/DNS/zabbix/CMDB

**Puppet** 大规模运维必备利器：统一；标准；快速

                              控制：os初始化；app环境初始化；ldap、ssh安全加固；全网或部分推送某配置变化；应用直接部署；CDN应用

               zabbix：

                    why zabbix：基于数据库可以分析数据；有agent可以跑任务；柔韧性

                    模板：linux基础模板；app默认模板；自定义

               CMDB：

                    资产：多少机房带宽；ip段；服务器；型号；IDC分布；

                    应用：多少应用；各自负责人；什么监控点；告警邮件直接抄送研发；上线自动添加监控，下线取消、维护自动暂停

**各类服务：**

**日志相关：**

               app日志接收平台

               各类hadoop日志收集平台

               系统/安全日志：rsyslog

               Hive日志分析

               实时日志分析：fluentd+mongo+zabbix

          监控平台：zabbix+URLMon

          自动化release平台

          单点登录平台：CAS-SSO

          DFS

          WEB源站统一：统一管理优化

          负载均衡服务化：LB Pair

          cache分层：共享cache服务MC，Redis

          数据库作为服务：MySQL

**app/web/客户端日志收集：**

**日志传输平台：**

     why：有十种以上的不同日志进hadoop需求；有五种以上不同进Hadoop的方案；断断续续轮流出问题；

     how：一个统一方案；吸取各个方案教训；尽量非常简单；非实时；新应用要用配置客户端即可；

**实时日志分析平台：**

     apache      /var/log/apache2/access\_log  tail->Fluentd 写到mongoDB

**离线日志分析：**

     Hadoop

     Hive

**安全日志分析：**

     系统：rsyslog+php

          bash日志

          /var/log/message

          登陆日志

          sudu日志

          。。。

     web安全：Fluentd +mongoDB

          LB post日志

**监控：**

     监控

     告警

     应用日志监控

     业务数据监控

     监控

          定义KPI和下限

          基础监控

          应用监控/用户体验监控

               用户端URL监控

               程序接口监控

               LB traffix/code/exceptions

          业务数据监控

          安全监控

     告警

     应用日志监控

          linux系统；nginx；php日志(关键字规律)

          java oom

          LB 4xx/5xx错误

          CDN错误统计

          。。。

     业务数据监控

          实时业务数据的监控最重要

          介于运营统计和运维监控之间

**自动化运维**

     运维自动化

     CDN机房自动化装机

     一键发布

     自动化排错

     运维自动化

          统一标准的基础上才能自动化运维

          自动化才能保证标准化

          减少人力；提高响应速度

          全网装机自动化；

          puppet保证应用配置与环境统一(90%以上)；          (这个具体怎么实现待学习，买了puppet实战)

          web应用部署90%自动化；

          cdn应用100%自动化；

          上包自动化；

          监控自动化：上线下线自动添加去除

     CDN机房自动化装机

          cobbler+git+controlTier

     一键发布

          PHP+ControlTier

               逐个发布

               与cmdb集成

               与zabbix/URLmon集成

               CAS认证

               借助LB日志分析

               一键回滚

     自动化排错

               zabbix rule engine

                    适应删除日志

                    自动重启：php/resin

                    告警邮件自动分析

                    自动java thread dump

**数据化运维：监控和日志数据的分析**

     监控告警阈值的制定

     日志分析：服务域名的可用性；CDN命中率、性能、流量；TOP-down分析

     容量分析和优化：提前发现问题

     结合CMDB：机器究竟在干嘛？数据！

**cdn优化**