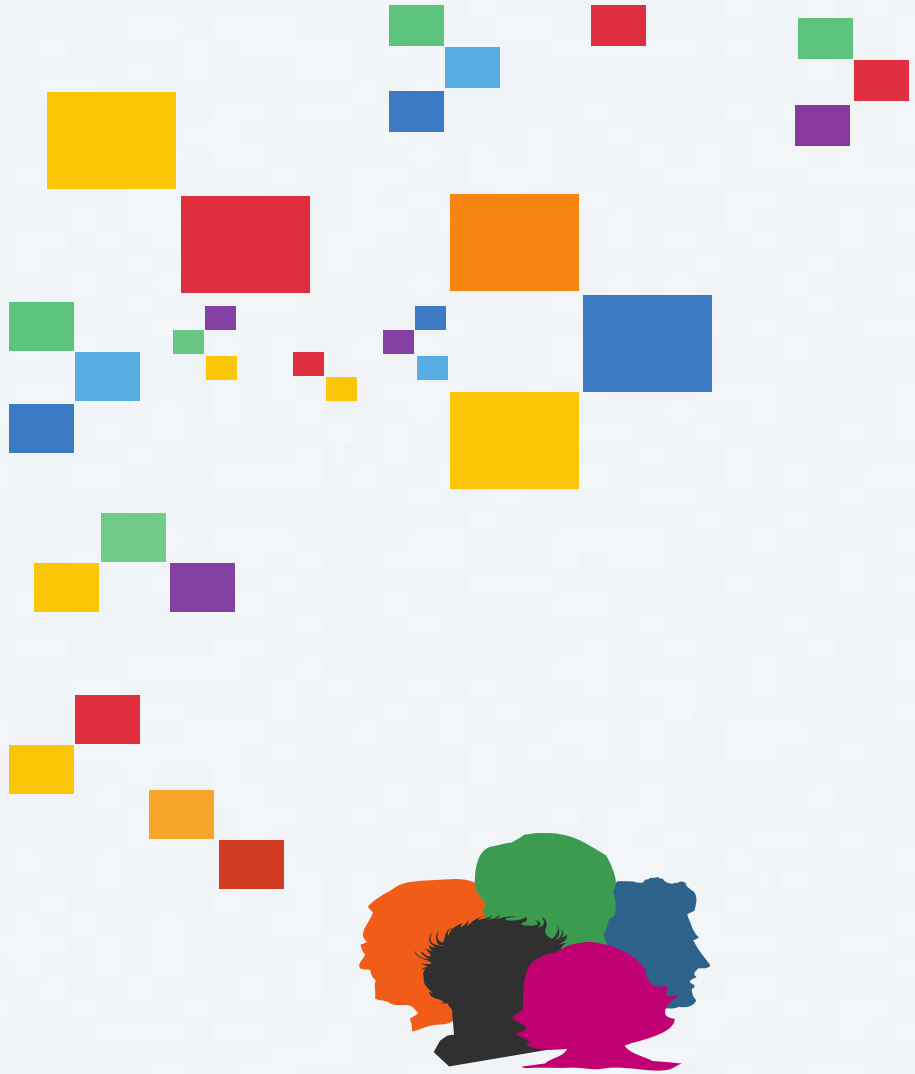
如何赋予 DevOps“生命力”

刘宇 / 斜杠青年



大纲

- DevOps来龙去脉
- DevOps从0到1
- 主流自动化运维工具
- 金山如何实践从0到1
- 讨论环节

Make Professional Discovery
MPD

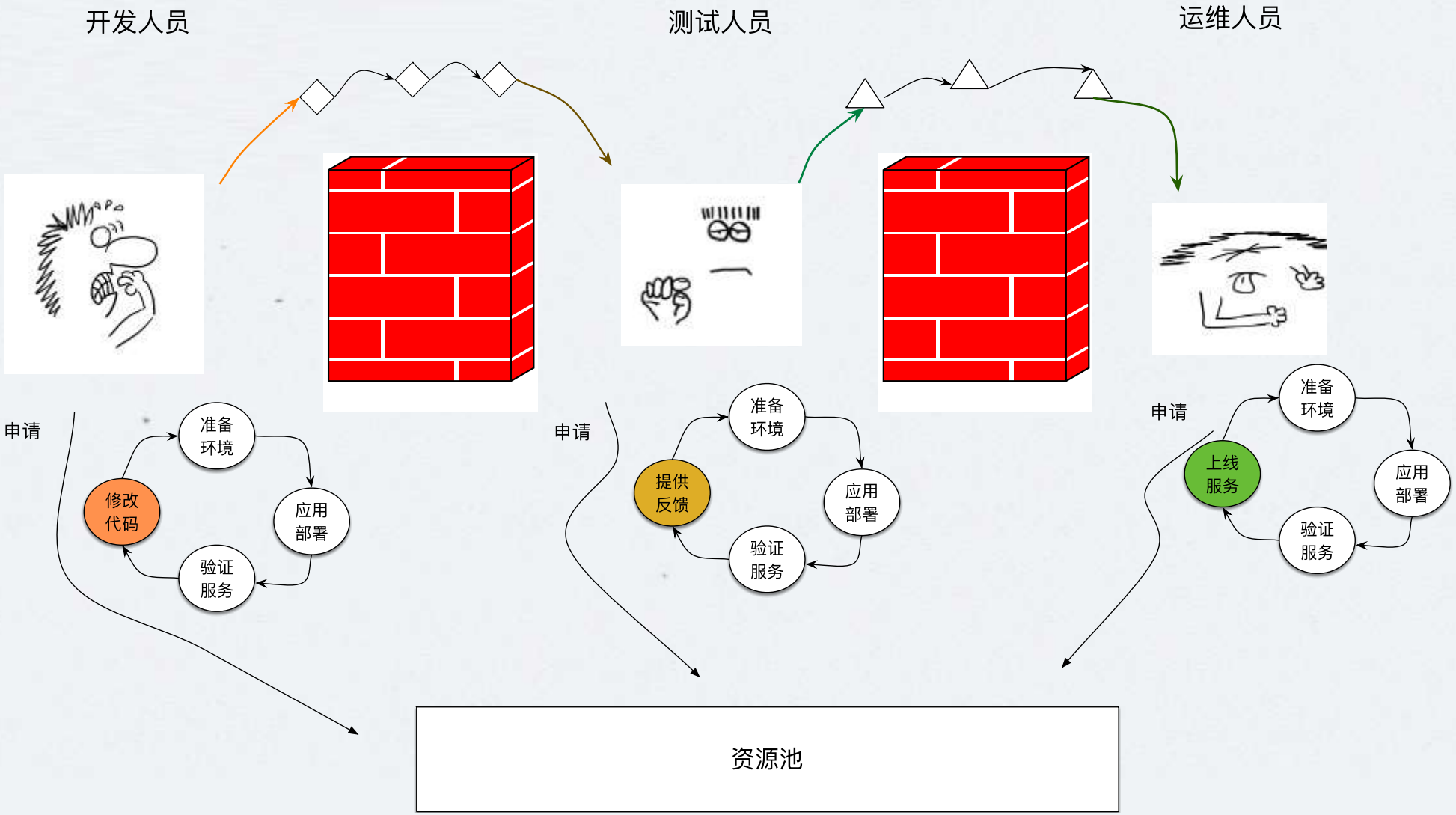
先来玩一个游戏！

什么是DevOps

“DevOps（英文Development和Operations的组合）代表一种文化、运动或实践。旨在促进软件交付和基础设施变更软件开发人员（Dev）和IT运维人员（Ops）之间的合作和沟通。它的目的是构建一种文化和环境使构建、测试、发布软件更加快捷、频繁和可靠。”

Source: http://en.wikipedia.org/wiki/DevOps

为什么需要DevOps?



大神级人物

Patrick Debois

比利时人

独立顾问

大多个大型企业项目中担任：开发者、网络专家、系统管理员、测试、项目经理

缘起

缘

Agile Infrastructure
&
Operations

2008年 Agile

“10+ Deploys per Day:
Dev and Ops Cooperation at Flickr.”

2009年 Velocity

比利时
“life will be much more fun”

2009年

思想领袖

催化剂

诞生 ！

DevOpsDays

- 包含：Develop
- 包含：Operation
- 将会举办2天，花1天的时间找出他们的问题所在
- 就叫：DOD 吧！
- 早期支持者
- John Willis： 系统管理专家
- Kris Buytaert： Linux开源顾问

大会的举办


2009
Ghent 2009
2010
Sydney 2010
Mountain View 2010
Hamburg 2010
Sao Paulo 2010
2011
Boston 2011
Mountain View 2011
Melbourne 2011
Bangalore 2011
Göteborg 2011
Manila 2011
2012
Austin 2012
Tokyo 2012
Delhi 2012
Mountain View 2012
New York 2012
2013
New Zealand 2013
London 2013
Paris 2013
Austin 2013
Berlin 2013
Amsterdam 2013
Silicon Valley 2013
Downunder 2013
Bangalore 2013
London Autumn 2013
Barcelona 2013
Vancouver 2013
Portland 2013
New York 2013
Atlanta 2013
Tel Aviv 2013
Tokyo 2013
17次
2014
Nairobi 2014
Ljubljana 2014
Austin 2014
Pittsburgh 2014
Amsterdam 2014
Silicon Valley 2014
Minneapolis 2014
Brisbane 2014
Boston 2014
Toronto 2014
New York 2014
Warsaw 2014
Belgium 2014
Helsinki 2014
Vancouver 2014
Tel Aviv 2014
Bangalore 2014
19次
2015
Ljubljana 2015
Paris 2015
Denver 2015
New York 2015
Austin 2015
Toronto 2015
Washington, DC 2015
Amsterdam 2015
Minneapolis 2015
Melbourne 2015
Pittsburgh 2015
Chicago 2015
Boston 2015
Tel Aviv 2015
Singapore 2015
Berlin 2015
Charlotte 2015
Silicon Valley 2015
Detroit 2015
Ohio 2015
2016
Vancouver - Apr 15 & 16
London - Apr 19 & 20
Denver - Apr 21 & 22
Atlanta - Apr 26 & 27
Austin - May 2 & 3
Kiel - May 12 & 13
Seattle - May 12 & 13
Toronto - May 26 & 27
Istanbul - Jun 3
Washington, DC - Jun 8 & 9
Salt Lake City - Jun 14 & 15
Silicon Valley - June 24 & 25
Amsterdam - Jun 29, 30 & Jul 1
Minneapolis - Jul 20 & 21
Portland - Aug 9 & 10
Chicago - Aug 30 & 31
Dallas - Sep 15 & 16
New York - Sep 23 & 24
Singapore - Oct 8 & 9
Detroit - Oct 12 & 13
Philadelphia - Oct 26 & 27
建立开源工具
2011年
2012年
2013年
预测：2015 年全球2000家企业将有20%拥抱DevOps
DevOpsDays 开始在全球爆发式增长
书，来了！！！

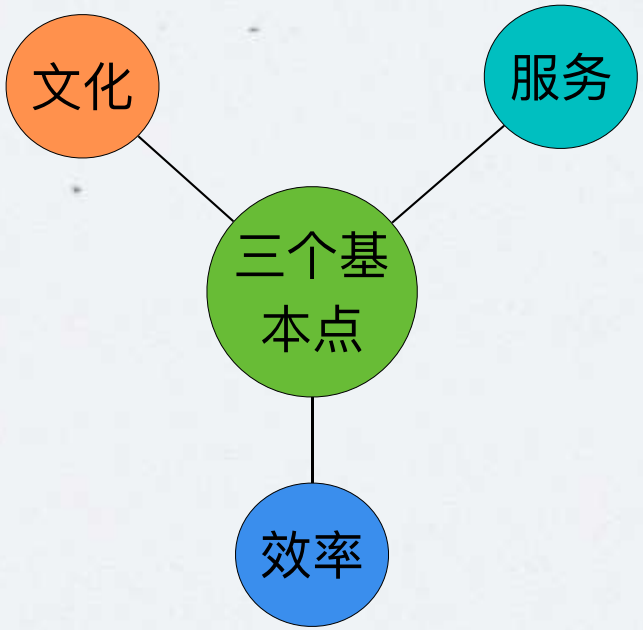
What is DevOps?

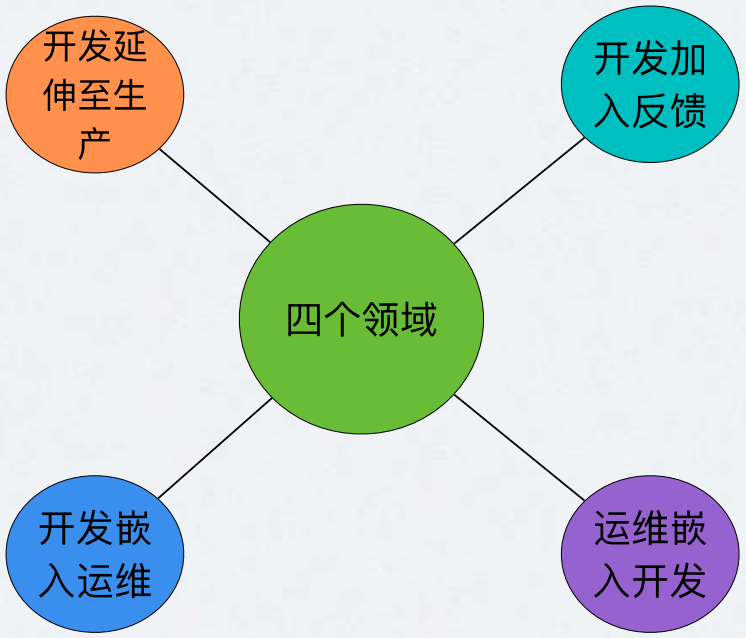
重大的新闻：O‘Reilly 内容战略副总裁Mike Loukides和Debois 一起发表DevOps的文章，指出：“What is DevOps? "

作者	书名	译名
Gene Kim, Kevin Behr and George Spafford	《The Phoenix Project》	《凤凰项目 一个IT运维的传奇故事》
Mary and Tom Poppendiek	《Implementing Lean Software Development》	《精益软件开发》
Eric Ries	《The Lean Startup》	《精益创业》
John Allspaw	《Web Operations》	《网站运维》
Jez Humble and David Farley	《Continuous Delivery》	《持续交付》
Dr. Eliyahu M. Goldratt	《The Goal》	《目标》

特征

不难看出，DevOps在企业的成功运用离不开《凤凰项目》和《DevOps cookbook》强调三个基本点以及四个领域。

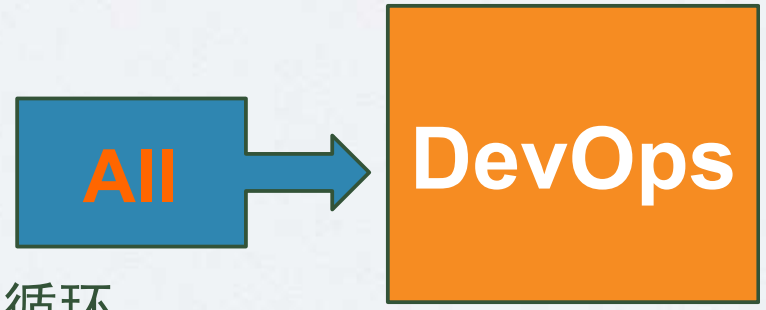




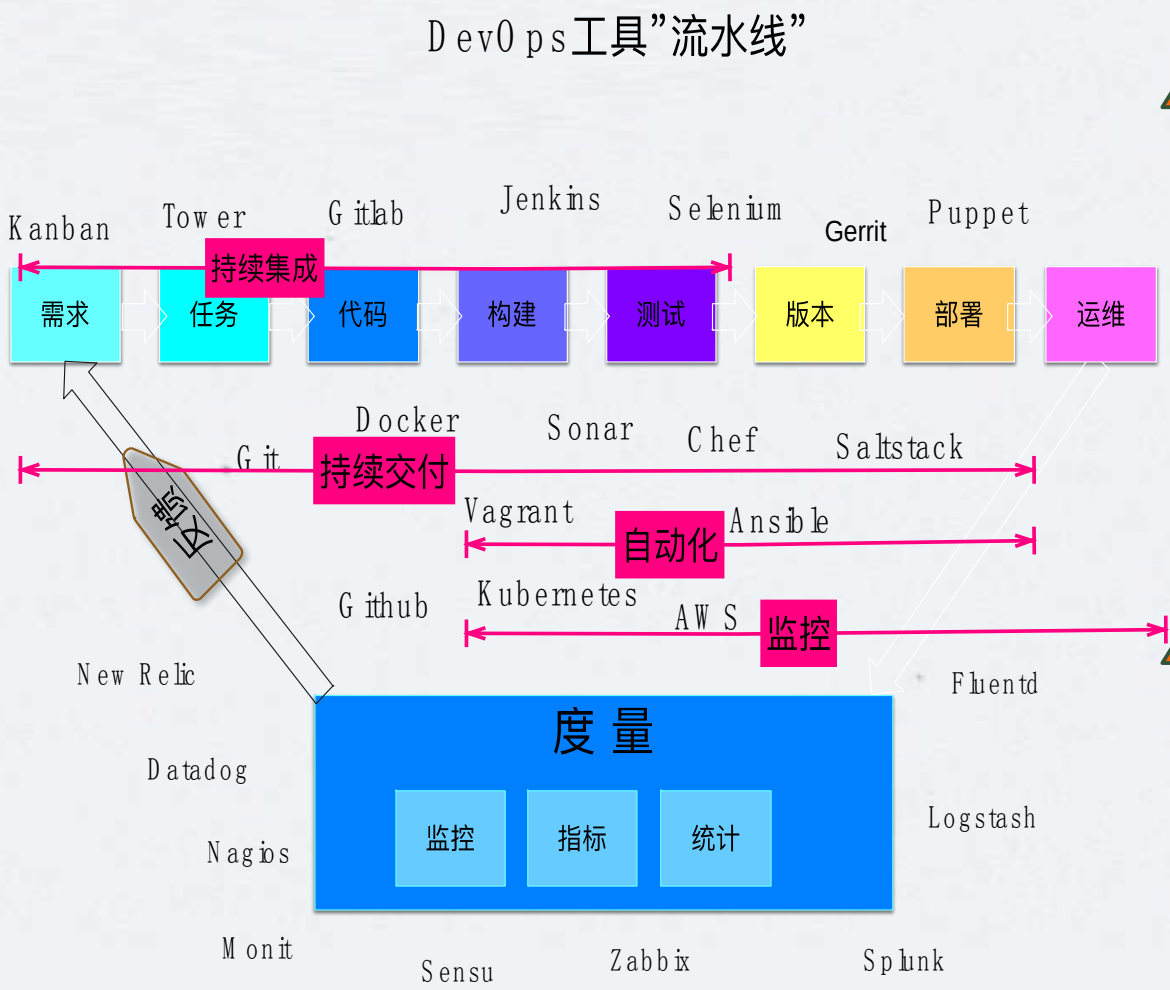
5个关键点（CALMS）

5个关键点（CALMS）

Culture	文化：自身的改变，推进沟通与合作
Automation	自动化：利用工具减少人工操作
Lean	精益：利用精益原则，实现更高的价值循环
Measurement	度量：衡量一切，用数据来精练产品生命周期
Sharing	分享：分享经验，不管成功与否，供别人学习



DevOps最佳实践



- 工具即服务
  - 共享所有的工具集
  - 基础设施、研发工具、构建/测试/部署、运维工具
  - 工具可独立使用，也可以协作
- 沟通、改进、创新
  - ✓ 站立会，复盘
  - ✓ Hack day
  - ✓ 分享一切

大纲

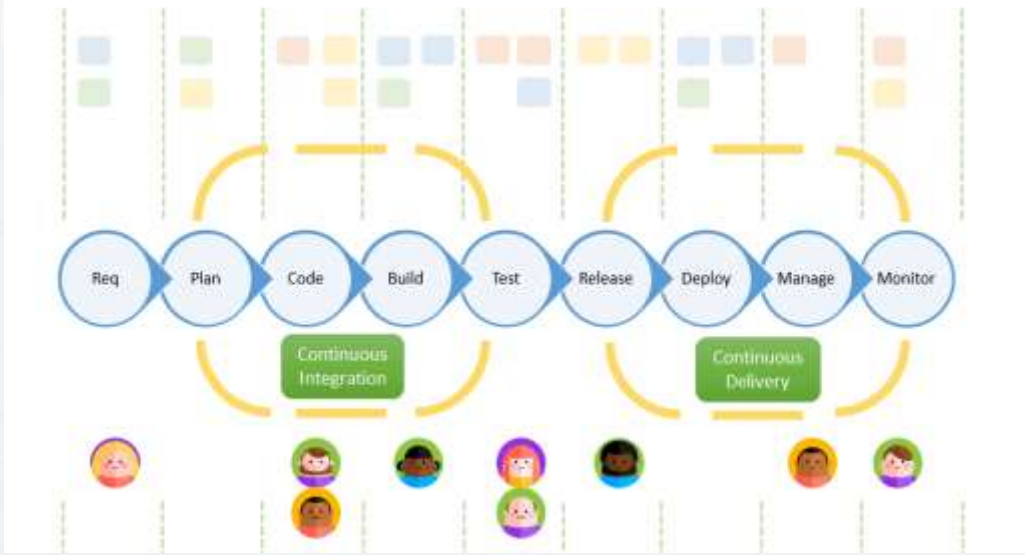
- DevOps来龙去脉
- DevOps从0到1
- 主流自动化运维工具
- 金山如何实践从0到1
- 讨论环节

Culture（文化）：站立会

- DevOps是一场文化运动。因此文化是最重要的一点！
- 信任、沟通、创新是DevOps文化的精髓
- 站立会是增强沟通和建立信任的第一步。

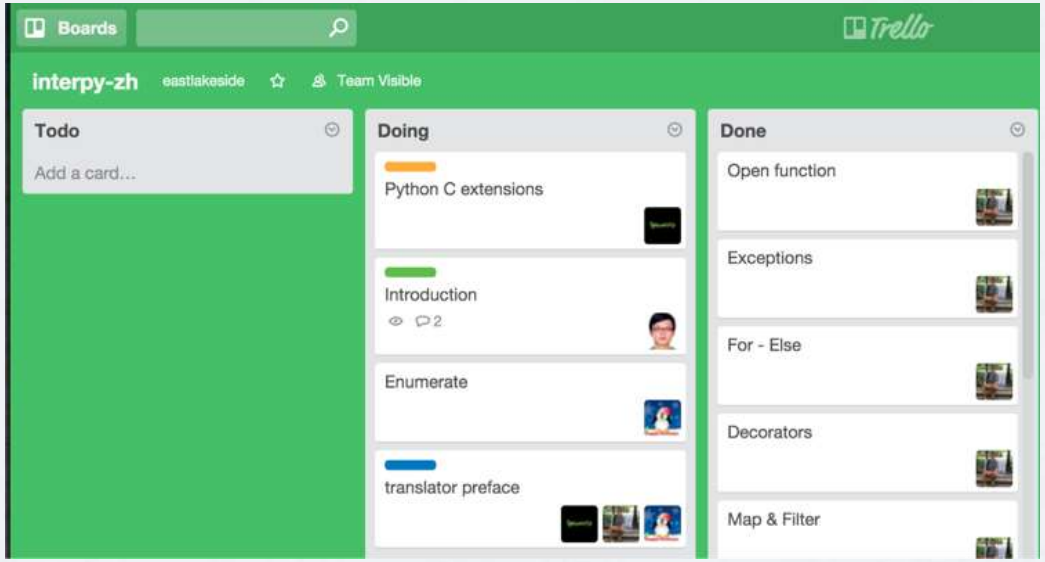
Culture（文化）：看板

- 看板
  - 明确项目需求
  - 制定里程碑
  - 任务与Issue结合
- 复盘
  - 总结
  - 发现问题
  - 制定改进方案

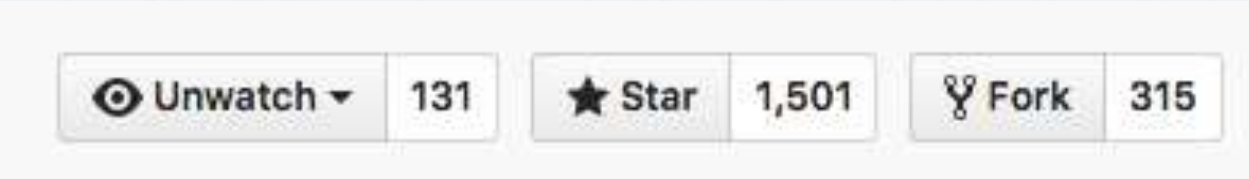


Culture（文化）：翻译

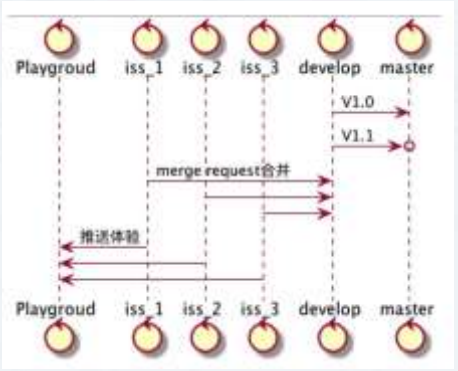
- 《Python进阶》
- https://github.com/eastlakeside/interpy-zh



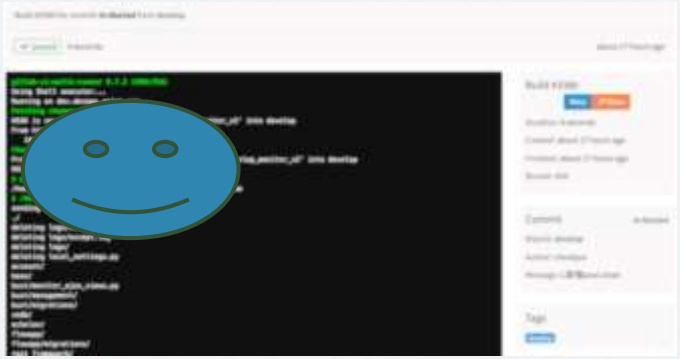
- 1天1000+Star



Automation（自动化）

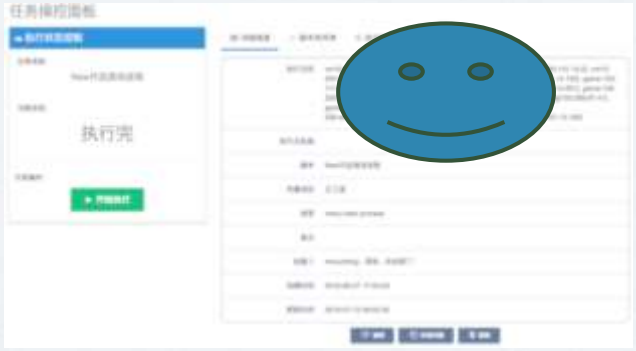


定好开发流程



Gitlab-ci持续集成

Sonar代码质量管理



多环境自动化部署

Lean（精益）

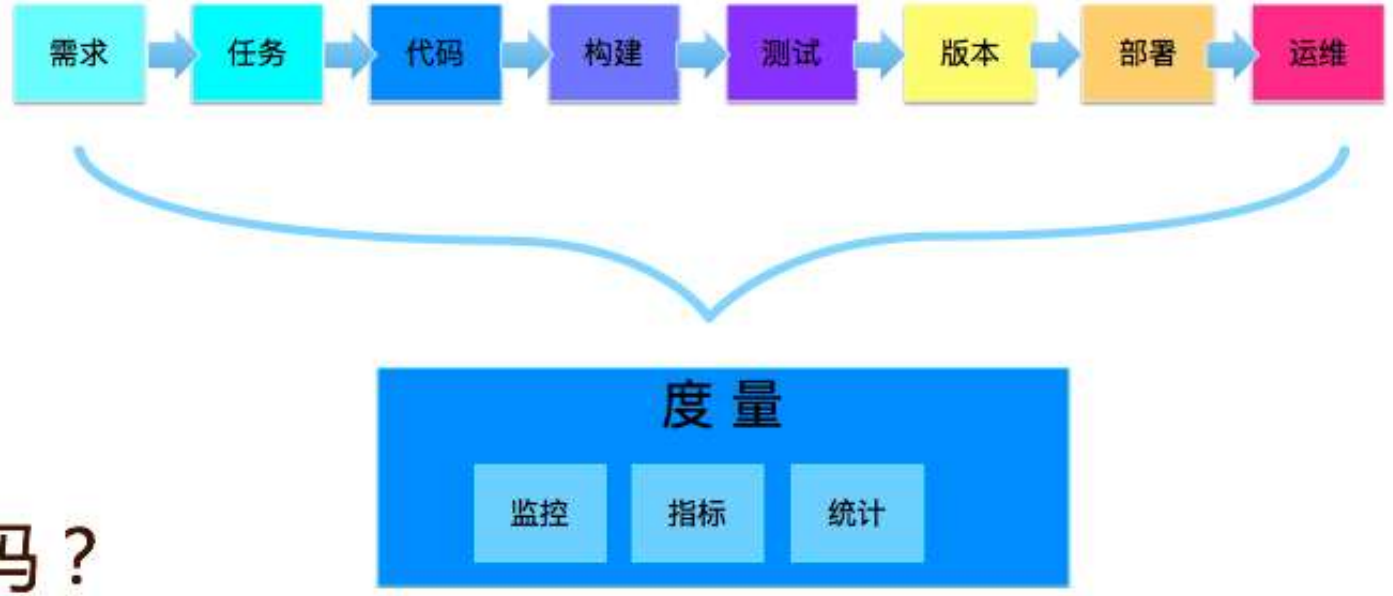
不重复劳动，保持简单高效

不要害怕失败

拆 掉 思 维 的 墙 ！ ！ ！

Measurement（度量）

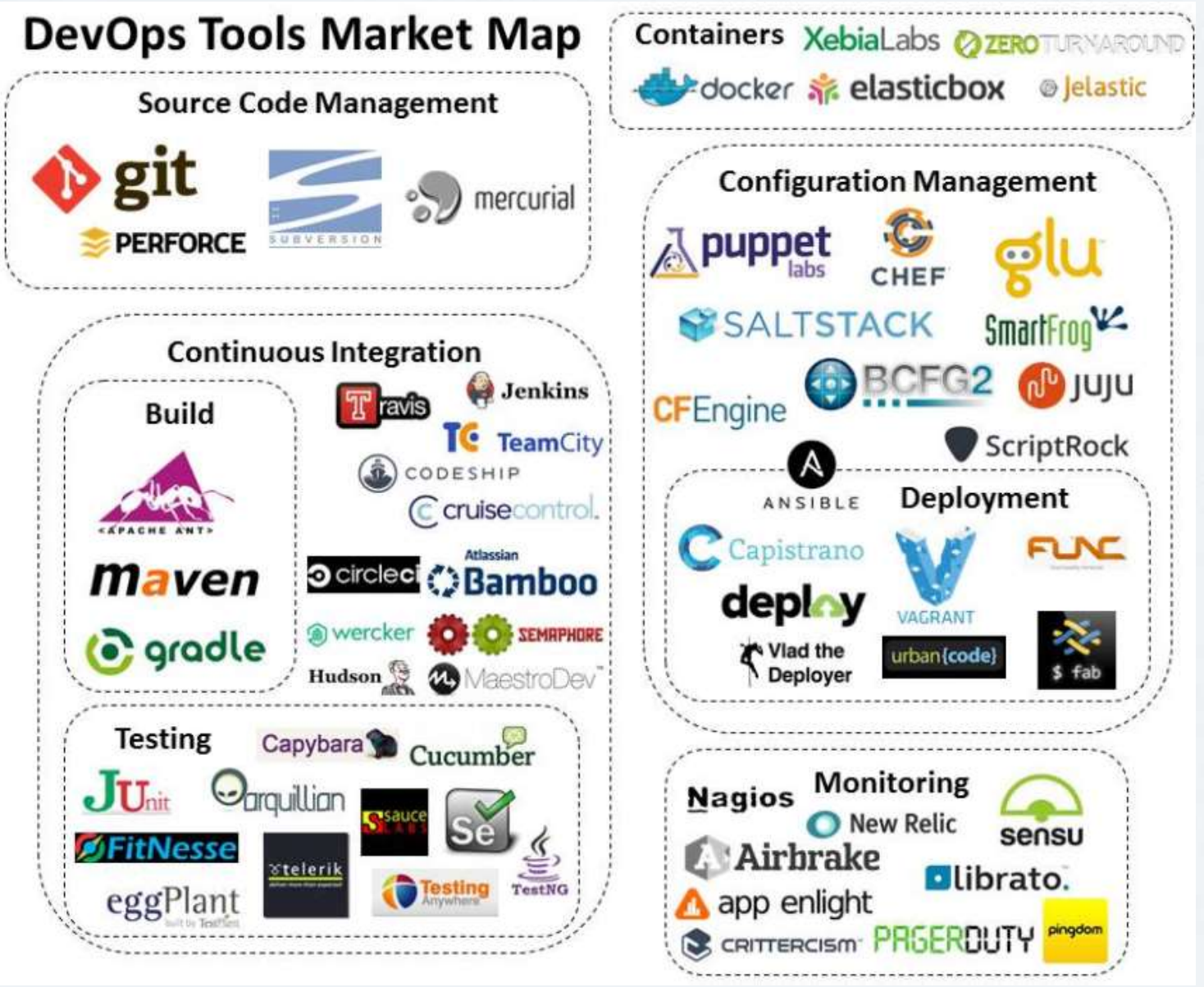
- 每一个点加入度量
- 故障的分析



- 能做到用钱衡量吗？

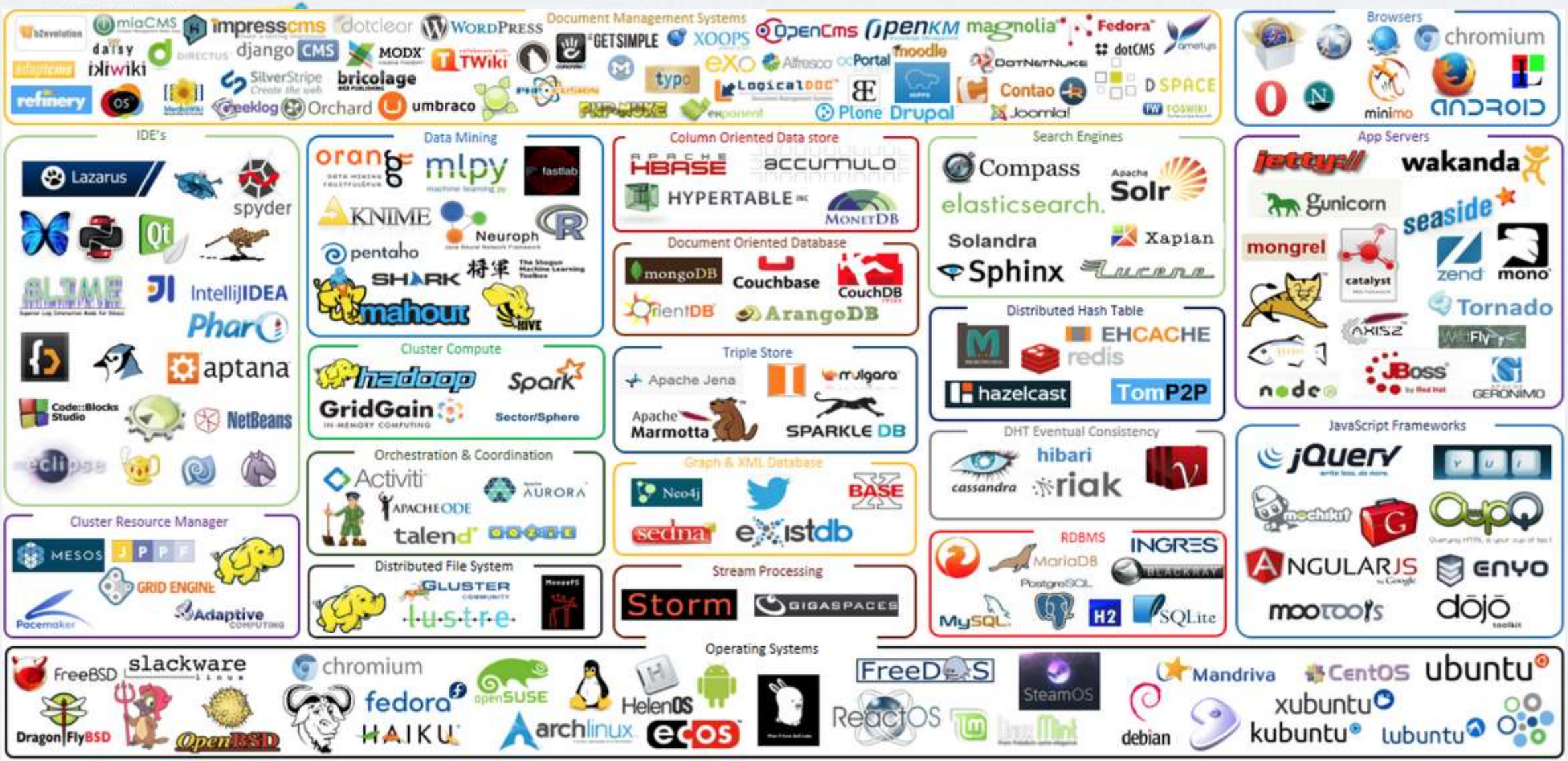
Make Professional Discovery

DevOps Tools Market Map



大纲

- DevOps来龙去脉
- DevOps从0到1
- 主流自动化运维工具 
- 金山如何实践从0到1
- 讨论环节


Document Management Systems
b2evolution
miaCMS
impresscms
dotclear
WordPress
daisy
DIRECTUS
django CMS
MODX
TWiki
GETSIMPLE
XOOPS
OpenCms
OpenKM
magnolia
Fedora
dotCMS
ikiwiki
SilverStripe
bricolage
eXo
Alfresco
Portal
moodle
DotNetNuke
refinery
OS
geeklog
Orchard
umbraco
typo
LogicalDOC
Contao
DSPACE
Plone
Drupal
Joomla!
FOSWIKI
Browsers
chromium
minimo
android
IDE's
Lazarus
spyder
Qt
SLIME
IntelliJIDEA
Phar
aptana
Code::Blocks Studio
NetBeans
eclipse
Data Mining
orange
mlpy
fastlab
KNIME
Neuroph
pentaho
SHARK
mahout
HIVE
Column Oriented Data store
HBASE
accumulo
HYPERTABLE
MonetDB
Document Oriented Database
mongoDB
Couchbase
CouchDB
OrientDB
ArangoDB
Search Engines
Compass
elasticsearch.
Apache Solr
Solandra
Xapian
Sphinx
lucene
App Servers
jetty://
wakanda
gunicorn
seaside
mongrel
catalyst
zend
mono
Tornado
AXIS2
JBoss
GERONIMO
node
Cluster Compute
hadoop
Spark
GridGain
Sector/Sphere
Triple Store
Apache Jena
Mulgara
Apache Marmotta
SPARKLE DB
Distributed Hash Table
EHCACHE
redis
hazelcast
TomP2P
DHT Eventual Consistency
hibari
cassandra
riak
Orchestration & Coordination
Activiti
AURORA
APACHE ODE
talend
Graph & XML Database
Neo4j
BASE
sedna
existdb
JavaScript Frameworks
jQuery
YUI
mochikit
OUPO
ANGULARJS
ENYO
MOOTOOLS
dojo
Cluster Resource Manager
MESOS
JPPF
GRID ENGINE
Pacemaker
Adaptive COMPUTING
Distributed File System
GLUSTER
lustre
Stream Processing
Storm
GIGASPACES
RDBMS
MariaDB
PostgreSQL
INGRES
MySQL
H2
SQLite
Operating Systems
FreeBSD
slackware
chromium
fedora
openSUSE
HelenOS
HAIKU
archlinux
ecos
FreeDOS
SteamOS
ReactOS
Mandriva
CentOS
ubuntu
debian
xubuntu
kubuntu
lubuntu
DragonFlyBSD
OpenBSD

OS层

云计算IaaS

- CloudStack：基于xen虚拟化技术，归属apache基金会，主要由思杰公司(xen母公司)支持，是企业私有云快速实现的首选。

- OpenStack：基于kvm虚拟化技术，有自己单独的基金会，python开发，是目前云计算创业公司普遍采用的技术。

云计算PaaS

- OpenShift：RedHat开源的PaaS平台
- CloudFoundry：VMWare开源的PaaS平台。最早的PaaS开源实现。

容器

- LXC：早期Debian系统上带有的cgroup管理工具
- OpenVZ：早期容器，所谓超售VPS大多基于该技术实现
- Docker：当前最火的轻量级容器
- Rocket：CoreOS和Docker决裂后自主开发的容器
- warden：cloudfoundry内使用的自主开发的容器

Docker简介

- 大致在底层技术上可以理解为 cgroups+namespace+aufs。

- 使用上主要特点是：仿照Git的操作方式，每次对镜像的变更，可以单独commit、单独push、pull使用。

分布式存储

- GlusterFS：C，RedHat项目。P2P架构。
- MooseFS：C，元数据信息在master内部记录，唯一一个只提供POSIX接口的项目，倍受运维欢迎。单点，数据安全性不足。
- MogileFS：Perl，livejournal开源。使用MySQL存储源数据，PB级别的扩展性依赖于MySQL集群的运维能力。最早的开源分布式文件系统。
- FastDFS：C，淘宝工程师个人开源项目。
- Swift：rackspace开源，openstack项目。分布式对象存储。
- HDFS：Java，Hadoop项目。目前运用最广。文件以64MB块存在。
- Ceph：分布式对象存储。在最近的docker浪潮中比重开始变大。

WEB应用

- apache：曾经最流行的Web应用服务器，通过mod_php方式构建了开源技术代表LAMP。
- nginx：目前最流行的Web应用服务器，加载mod_lua方式成为目前PaaS平台路由层的普遍选择。
- Tomcat/Jboss：最流行的Java应用服务器。
- php-fpm：和Nginx一起作为替代LAMP的当前最流行选择。

- Lighttpd：一度因为最早支持flv拖拽播放成为视频公司首选。
- squid：最流行的Cache应用服务器。
- varnish：因为对内存利用较好，在中小公司流行的Cache应用服务器。
- trafficserver：雅虎开源，在淘宝、Apple、Linkedin使用的Cache应用服务器。

负载均衡

- Bind9：最知名的DNS服务器软件，同时也是最通用的全局负载均衡软件。通过分zone解析的方式实现基于来源地址的负载均衡。社区还另外有专门的bind-dlz方案，将基于地理位置的bind配置储存在MySQL里，方便动态管理。

- LVS+keepalived：最通用的本地负载均衡软件。章文嵩博士开源，Linux内核模块。支持NAT、DR、TUN三种模式，rr、wrr、dh、lc、wlc、sed、nq、lblc、lblcr九种调度算法。

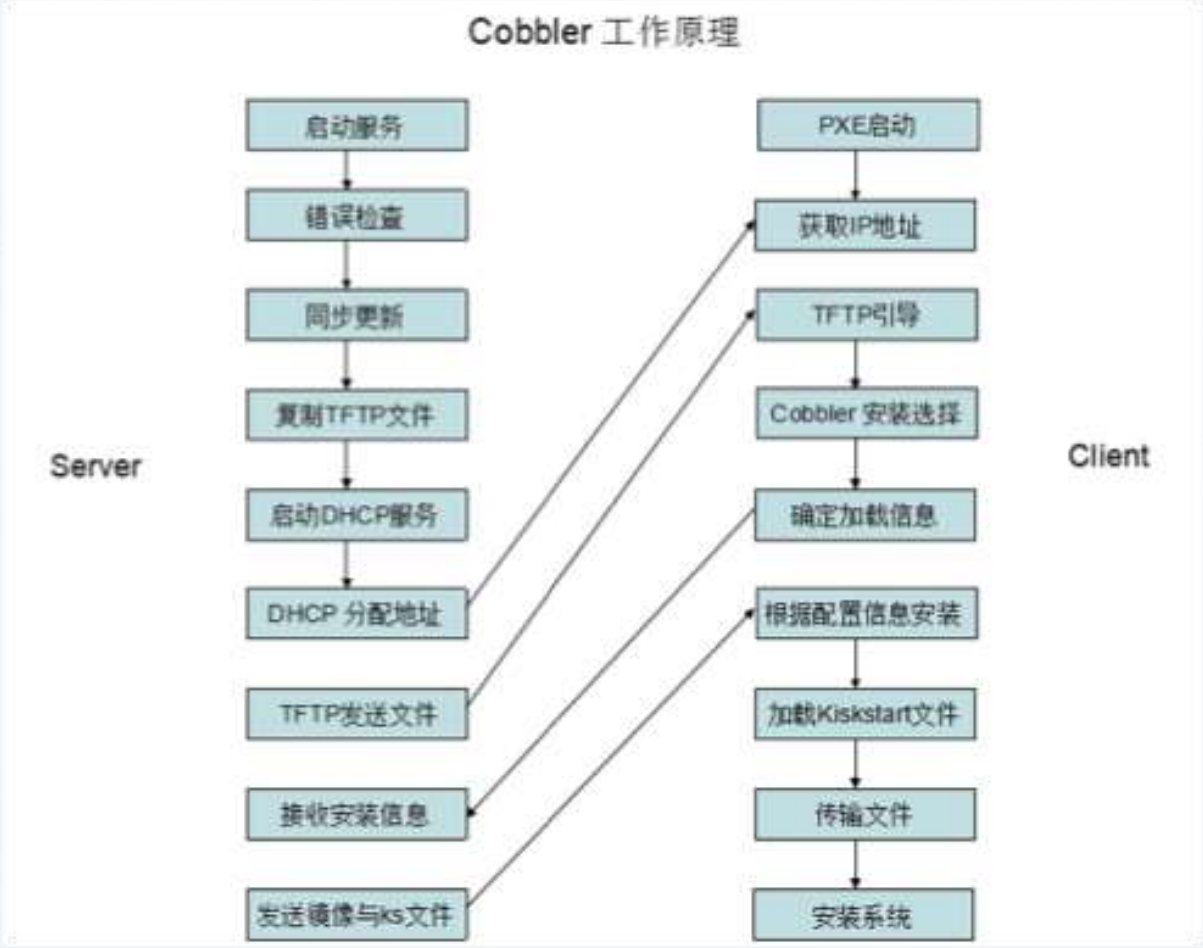
- HAproxy：OSI七层上实现的HTTP负载均衡。后期也提供TCP层功能。其实nginx、squid也可以理解为这个层面。

数据库

- MySQL、PostgreSQL：关系型数据库
- memcached、Redis、Tokyo-Tyrant：No-SQL缓存
- MongoDB、Cassandra：分布式No-SQL存储（文档式、列式）
- zookeeper、etcd、consul：分布式key-value存储
- elasticsearch、solr、sphinx：全文搜索引擎
- Influxdb:开源分布式时序、事件和指标数据库

系统安装

- Kickstart：结合pxe，dhcp，tftp
- Cobbler：结合pxe，macdhcp，reporsync
- 二者基于相同原理，仅在易用性区别



软件包分发

- 软件分发，或者说文件分发，有多种方式。从最传统的文件拷贝，到yum/apt软件仓库，到ClouderaManager的自管理。这里提两个在更普遍的运维场景中值得一试的开源工具：

- FPM(Effing package management)：通过命令行参数的方式，辅助完成跨平台的软件包构建和转换。对源码编译，不懂rpmbuild的人，极为有用。

- murder：Twitter开源的P2P文件分发命令，基于BitTorrent和Capistrano构建。注意：BT方式只对大文件有比较好的加速效果。Twitter和Facebook都是用在GB级别的分发。

代码&项目管理

- 偏重代码管理：
- GitLab：Ruby，类似Github私有化项目管理，目前最为流行
- Gogs：Golang，和Gitlab类似
- GitPrep：Perl，和Gitlab类似
- Gerrit：Java，偏重在review流程控制
- Phabricator：PHP，支持Git、SVN、CVS多种后端
- 偏重bug和工单管理：
- BugZilla：Perl，最流行的开源bug管理系统
- RequestTracker：Perl，最流行的工单管理系统
- Redmine：Ruby，新型团队较多采用
- Jira：Java，最流行的bug管理系统，但是收费

自动化构建

- Apache Ant是一个将软件编译、测试、部署等步骤联系在一起加以自动化的一个工具，大多用于Java环境中的软件开发

- Maven 除了以程序构建能力为特色之外，还提供 Ant 所缺少的高级项目管理工具。

持续集成

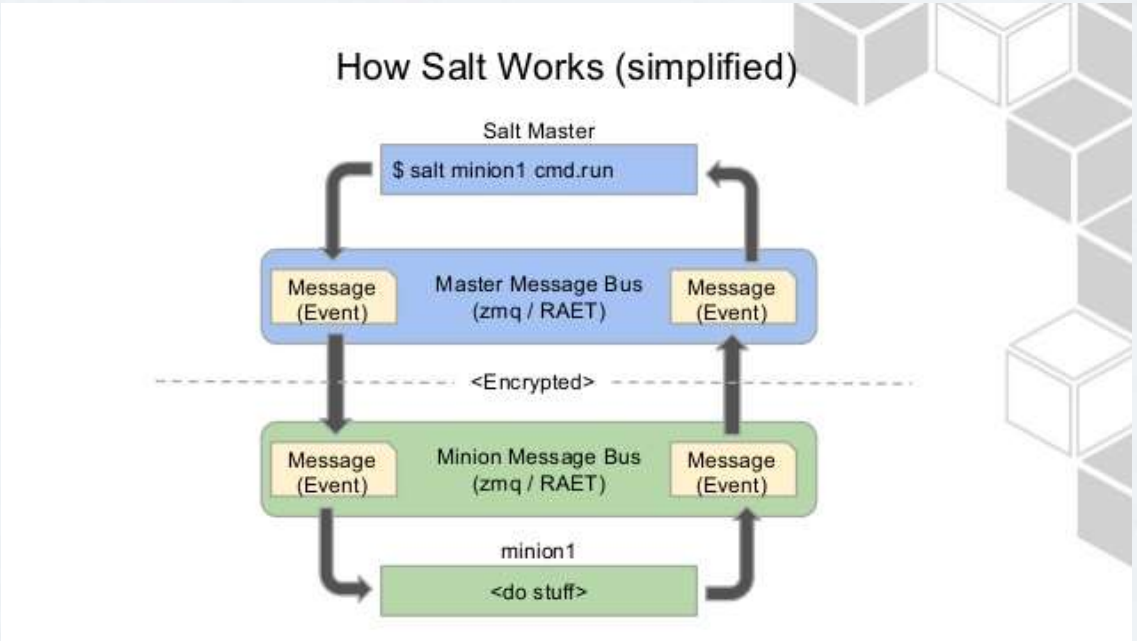
- Jenkins 的前身是 Hudson 是一个可扩展的持续集成引擎。
- Gitlab-ci Gitlab自带的CI工具，目前正在往DevOps方面发展
- Travis CI 是一个基于云的持续集成项目， 目前已经支持大部分主流语言了，比如：C，PHP，Ruby，Python, Nodejs等等。

命令执行

- Expect：TCL语言的一部分，配合bash直接循环调用。会导致用户名密码明文传输。
- Pssh：基于Python的命令执行工具。基于密钥双向验证。
- Fabric：基于Python的命令执行工具。基于pssh。
- Capistrano：基于Ruby的远程任务编排工具。可以认为是Rake（Ruby里的Makefile）的SSH版。

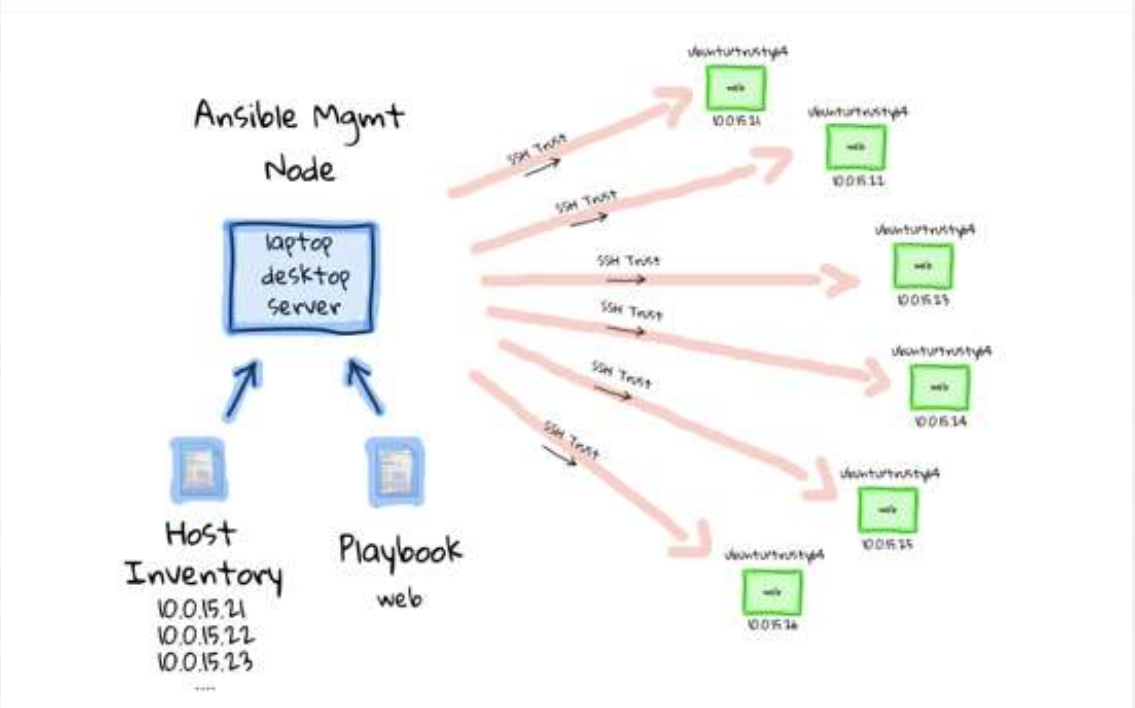
Saltstack消息队列架构

- saltstack目前抽象了消息总线层，支持了SSH、raet等多种方式。
- 消息队列方式，不足以支持大规模集群——小包、长链接。
- raet：基于UDP传输协议+CurveCP握手协议



Ansible

- 无需Master，基于Python，采用SSH通信
- Playbook（yaml）描述配置
- Inventory跟踪主机



如何选？

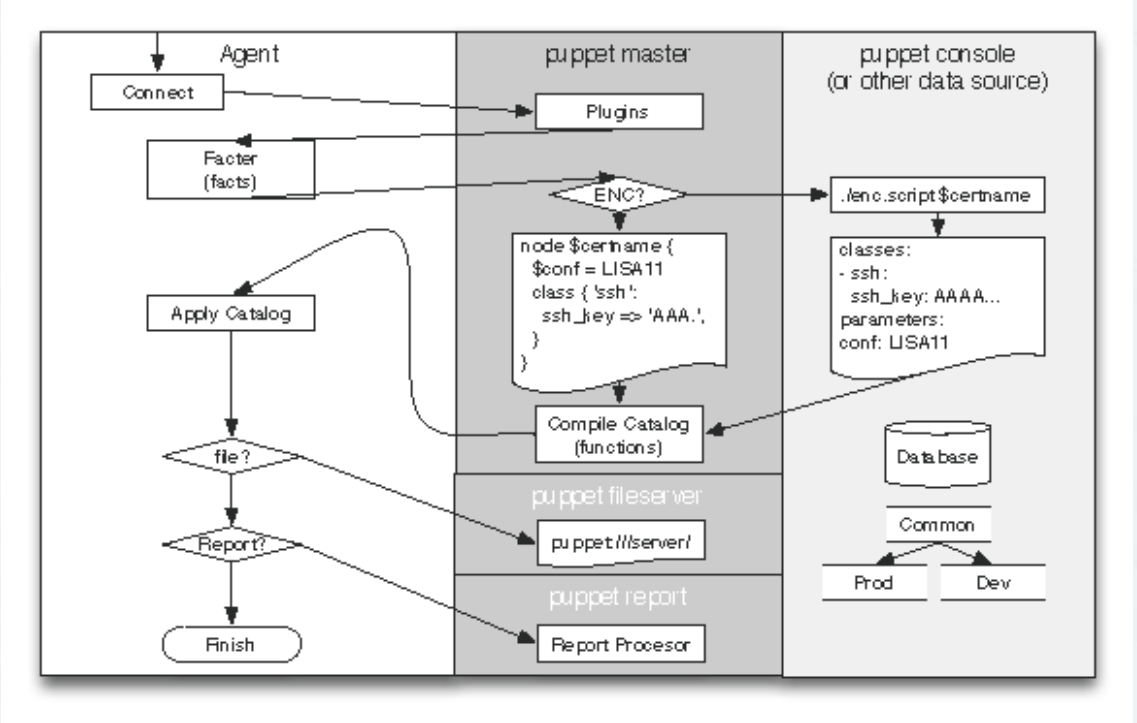
- Ansible市场占有率最高
- Salt其次
- Pssh可以用在小规模内部

配置管理

- CFEngine：C语言，采用头文件加载编译的方式。配管系统的鼻祖。
- Chef：Ruby+Erlang，类似ruby子集概念的DSL设计。
- Puppet：Ruby+Clojure，非常完善的DSL(领域描述语言)设计。被视作DevOps的代表性工具。
- Saltstack：Python，基于0MQ网络库，采用YAML描述配置。
- Ansible：Python，基于SSH链接，采用YAML描述配置。

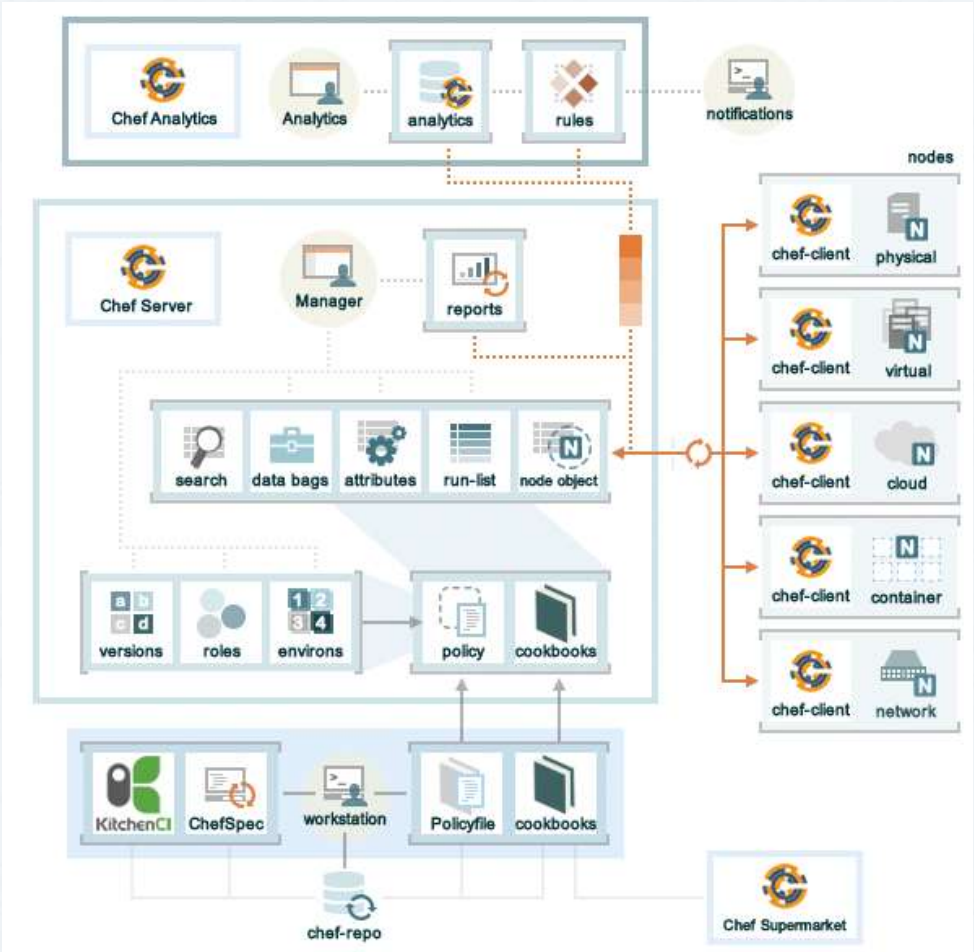
Puppet 数据流程架构

- Facter：采集服务器配置数据
- manifest：DSL式的资源配置清单
- catalog：由manifest编译而来的实际清单，可以直接单机apply生效
- ENC：读取外部CMDB的进程
- report：运行报表



Chef配管网络架构

- Chef在整体设计上和Puppet非常类似。
- 使用PostgreSQL和Solr提供存储和标签搜索等功能。
- 提供chef-solo的单机执行方式
- 配置语法依然在ruby体系内，对运维不如puppet友好。



如何选？

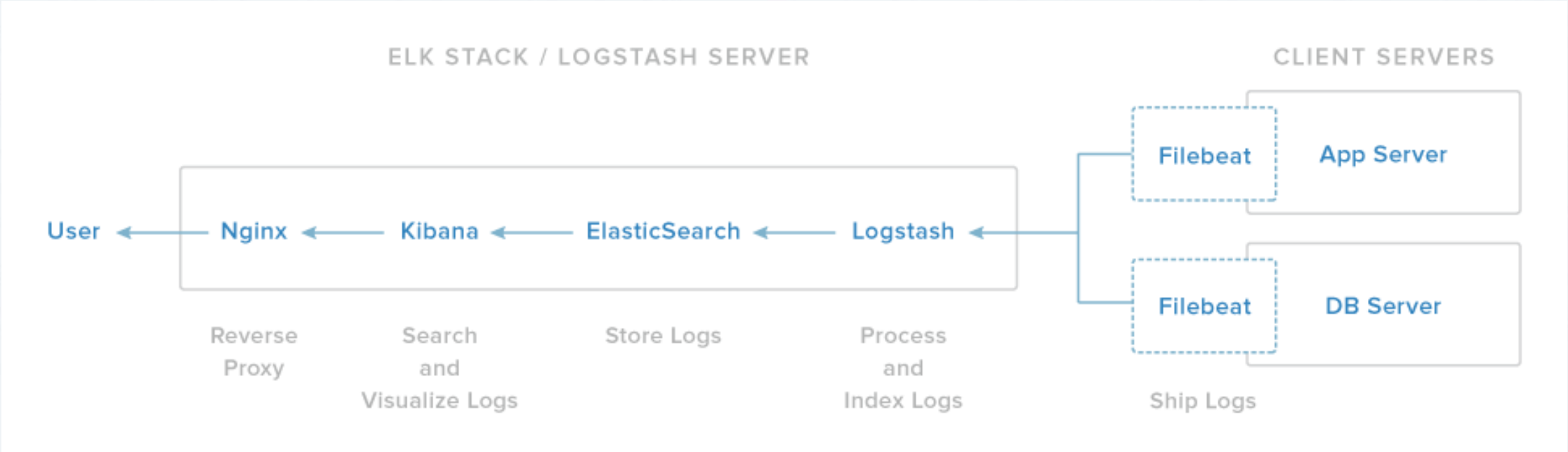
- Puppet与Chef市场占用率一致
- 国内Puppet 居多
- 各大互联网企业在使用
- 中国各银行都以Puppet为代表

监控

- Mrtg - 老牌网络监控工具
- Cacti - 基于rrdtools的网络监控工具
- Munin – 网络资源监控工具
- Smokeping – 基于fping的网络延迟、丢包监控
- Nagios - 服务器、网络、性能的监控平台
- Zabbix – PHP界面操作的监控系统
- Graphite – 灵活报表类的监控工具
- Grafana - 强大的数据可视化展示
- open-falcon - 自动采集、灵活定义可视化

ELK

- 由Elasticsearch、Logstash、Kibana、Beats等不同的开源软件，联合组成。
- 通过核心的全文索引引擎，提供对文本的实时检索和聚合统计。



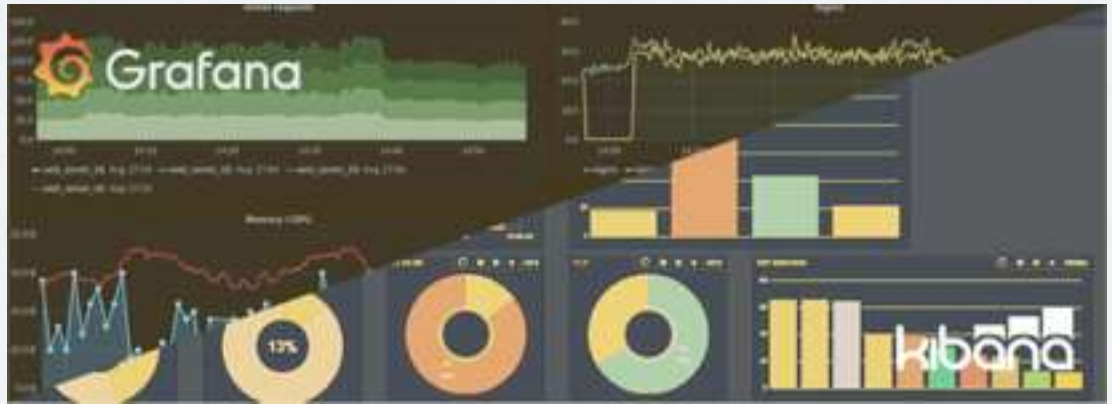
日志传输方案

- Flume：Java，apache基金会，Hadoop生态圈首选
- Scribe：C++，Facebook开源，已弃坑
- Rsyslog：C，RedHat默认自带
- Fluentd：Ruby， treasuredata开源，messagepack作者
- Logstash：JRuby，Elastic开源，fpm作者
- Heka：Golang，mozilla开源，模仿Logstash

最新展示方案

- Influxdb是Go语言写的一个开源，分布式，时间序列，事件，可度量和无外部依赖的数据库。

- Grafana 是一个开源仪表盘工具，它可用于Graphite、InfluxDB与 OpenTSDB一起使用。最新的版本还可以用于其他的数据源，比如Elasticsearch。



Make Professional Discovery

MPD

开源的正确使用姿势

1. 选对

- 业务场景需求
- 项目成熟度
- 社区活跃度
- 可运维性

2. 用对

- 提前阅读官方文档，至少是使用指南
- 广泛收集社区的用例和文章
- 针对自身需求设计测试场景和监控指标
- 进行长期压测和指标观察
- 测试不同配置项下的指标波动
- 准备好回退和备份方案
- 灰度上线

3. 基于开源研发

- 尽量保证成社区版本的同步更新
- 尽量复用原有接口
- 多构建外围扩展
  - 监控：serverspec对puppet
  - 扩展：twitter的Twemproxy对Redis
  - 配管：etsy的nagios-api对nagios

企业运用

- Zabbix监控、Grafana展示
- open-falcon二次研发（小米、滴滴、美团、金山云、京东、赶集等）
- Ansible + Puppet 组合
- Salt + Puppet 组合

大纲

- DevOps来龙去脉
- DevOps从0到1
- 主流自动化运维工具
- 金山如何实践从0到1
- 讨论环节

金山的从0到1

我们的0

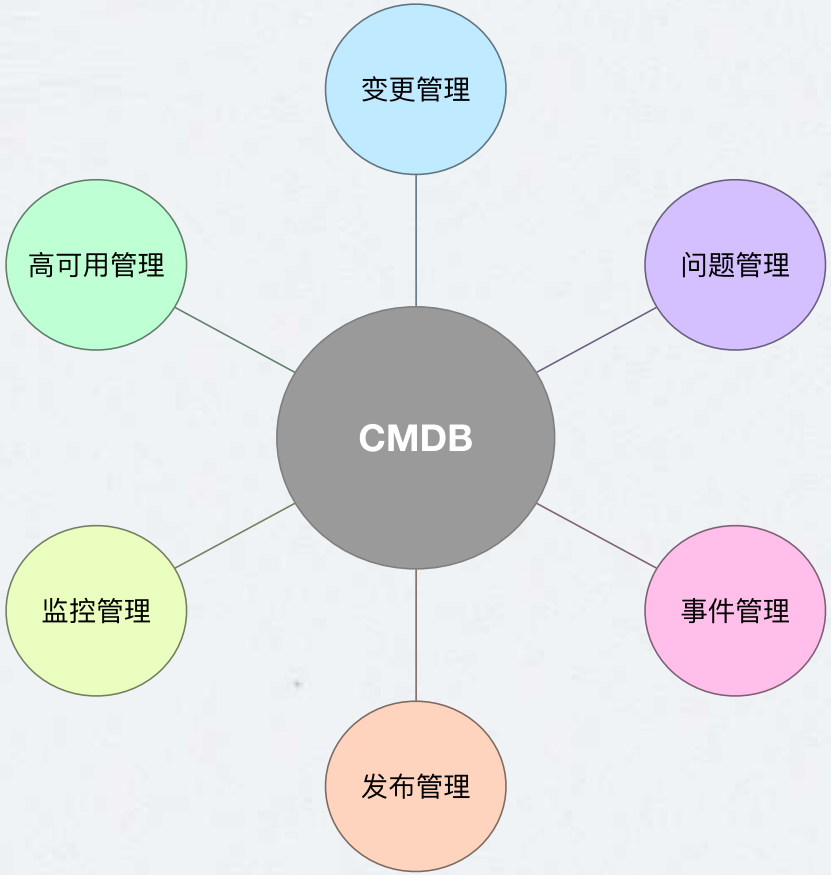
- 资产靠Excel
- 年底盘点（你杀了我吧）
- 手工代表一切
- 故障靠吼
- 排查靠吼
- 还有啥？

CMDB

CMDB，配置管理，即是ITIL的核心，也是运维的基础核心系统。

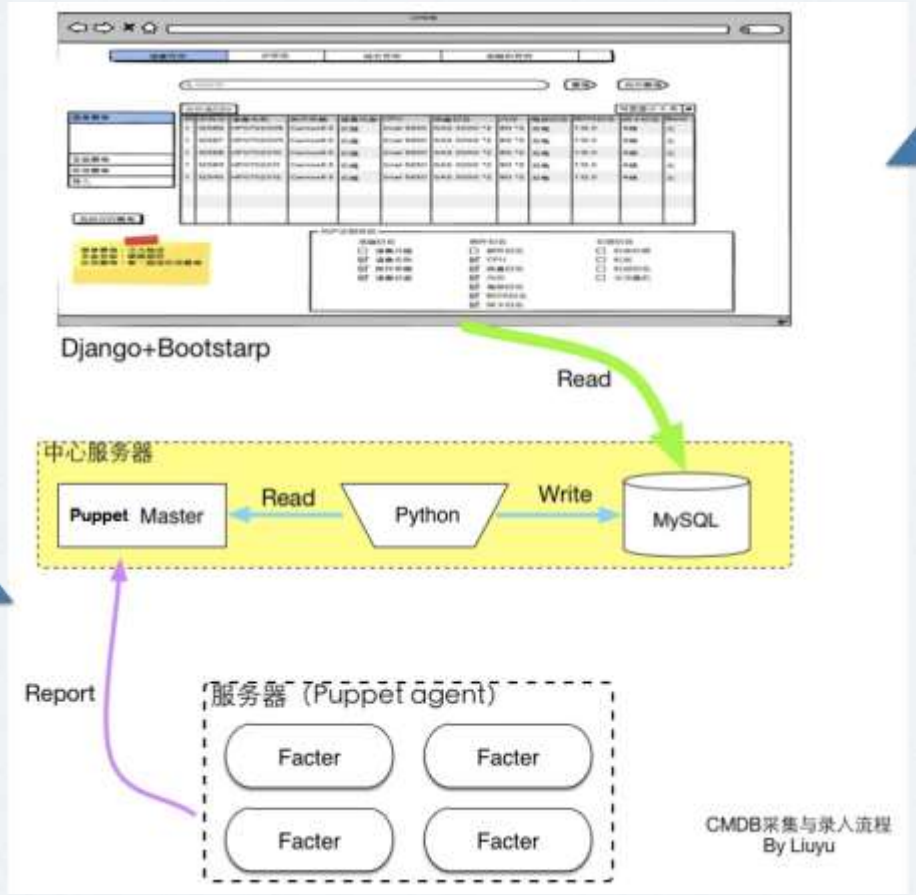
通常位于公司架构的底层，在运维自动化体现中，起到数据枢纽的关键作用。

稳固根基



自动采集

- 减少人为干预
- 主动采集
- 机器匹配
- 差异人工核对



- 节省80%的工作量

终于不用维护EXCEL了！

Zabbix监控

监控一切

	故障发生时间	故障历时	通知	事件是否处理	动作
分区空间小于15%，且小于5GB	06 七月 22:44:34	12h 1m 39s		否	6
er的/分区空间小于20%，且小于10GB	06 七月 16:18:38	18h 27m 35s		否	15
分区空间小于20%，且小于10GB	06 七月 11:34:34	23h 11m 39s		否	6
s(13-16)	02 七月 15:24:12	4d 19h 22m		否	11
及时清理	27 六月 23:53:20	9d 10h 52m		否	10
于95%	25 六月 07:28:24	12d 3h 17m		否	11

Make Professional Discovery
MPD

有了这三套，查故障嗖嗖的。

命令执行

在这里输入命令:

who

戳我执行

快捷命令:

ls -l

ps

who

uname -a

任务:

who

指令结果展示:

```
Liuyu    console  Jun  2 08:58
```

Django+Boostrap

```python
import os
import urllib2

import subprocess as sub
import threading

class Command(object):
    def __init__(self, cmd):
        self.cmd = cmd
        self.process = None
        self.stdout = ""

    def run(self, timeout):
        def target():
            self.process = sub.Popen(self.cmd,
                                     stdout=sub.PIPE, stderr=sub.STDOUT,
                                     shell=True)
            #self.process.communicate()
            self.stdout = self.process.stdout.read()


        thread = threading.Thread(target=target)
        thread.start()

        thread.join(timeout)

        if thread.is_alive():
            self.process.terminate()
            thread.join()

        return self.stdout
```

```python
#@login_required()
def run_command(request, app_id=None):

    CMD_TIMEOUT = 3             # 注意命令超时时间

    if request.method == 'GET':
        command_str = request.GET.get('cmd', '')

    if request.method == 'POST':
        command_str = request.POST.get('cmd', '')

    # 转换URL编码
    command_str = urllib2.unquote(command_str)

    command = Command(command_str)
    cmd_stdout = command.run(timeout=CMD_TIMEOUT)

    # p = sub.Popen(['/bin/bash', '-c', command_str],
    #               stdout=sub.PIPE, stderr=sub.STDOUT)

    output = urllib2.unquote(cmd_stdout)

    return render_to_response("command/cli.html", {
        "cmd_str": command_str,
        "cmd_result": output,

        "commands": ["ls -l",
                     "ps",
                     "pwd",
                     "who",
                     "uname -a",
                     "ssh someone@gmhost ls -l",
                     ]
    }, context_instance=RequestContext(request))
```

HTML

cli.html × views.py ×

```
<form method="post" action="/gmcli" class="form-vertical" id="form-cli">
  {% csrf_token %}

  <div class="form-group">
    <label class="control-label" for="id_cmd"> 在这里输入GM指令: </label>

    <input id="id_cmd_str" class="form-control" name="cmd" value="{{cmd_str}}"
           placeholder="ls -l" style="color:grey;">

    <button class="btn btn-primary">戳我执行</button>
  </div>

  <div class="form-group">
    <label class="control-label"> GM快捷命令: </label>

    {% for cmd in commands %}
      {% if forloop.first or forloop.counter0|divisibleby:3 %}
        <div class="row">
      {% endif %}

      <div class="col-md-4">
        <div class="wrap">
          <a class="btn btn-default" type="submit" href="#" value="{{ cmd }}"
             onclick="$('#id_cmd_str').val($(this).attr('value'));$('#form-cli').submit();">
            <i class="glyphicon glyphicon-tasks"></i> {{cmd}}</a>
        </div>
      </div>

      {% if forloop.last or forloop.counter|divisibleby:3 %}
        </div>
      {% endif %}
    {% endfor %}
  </div>
</form>

<strong> {{ cmd_str }} </strong> <p>

GM指令结果展示:
<pre> {{ cmd_result }} </pre>

{% endblock %}
```

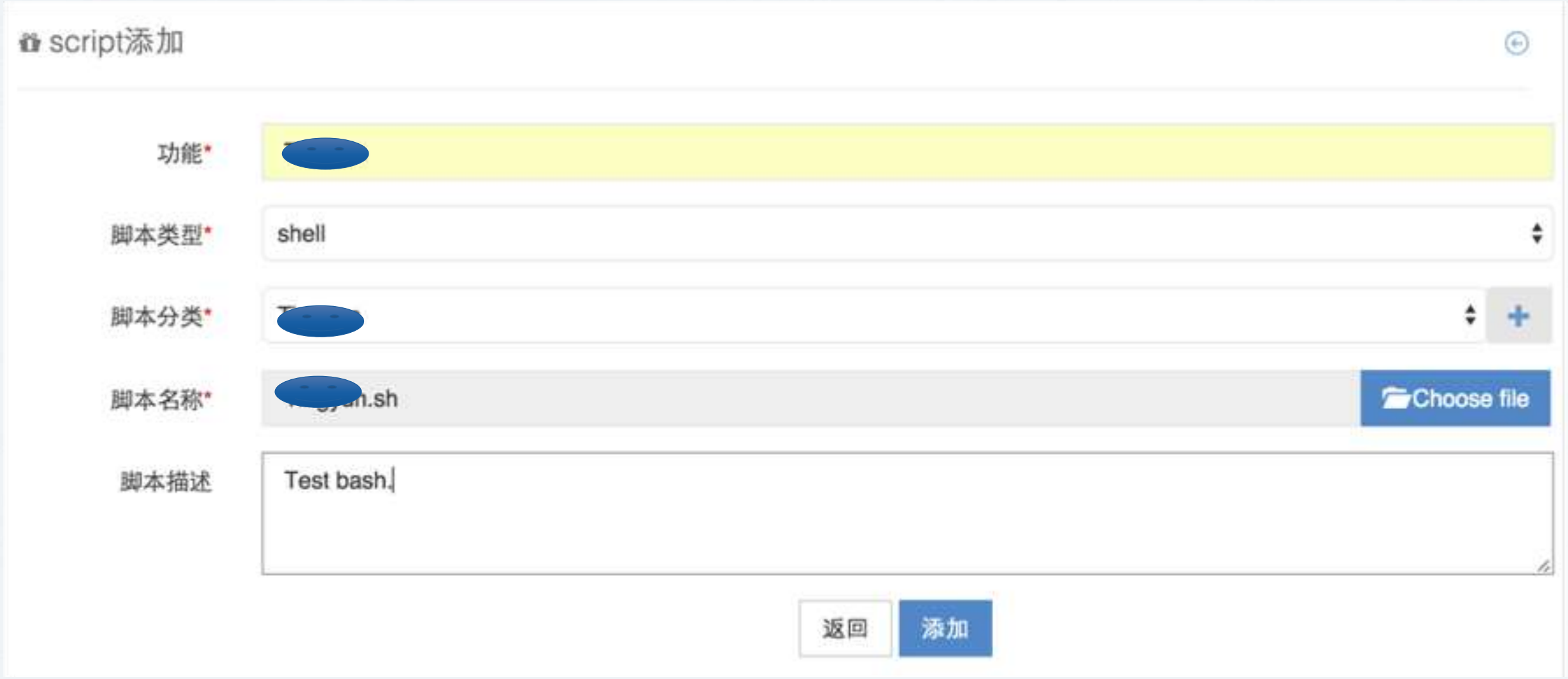
命令的弊端

1. 设置黑名单
2. 特殊符号转义
3. 复杂命令难以实现
4. 批量任务无法执行

Make Professional Discovery
MPD

那还是跑脚本吧！

上传


script添加
功能*
脚本类型*
shell
脚本分类*
脚本名称*
.sh
Choose file
脚本描述
Test bash
返回
添加

核心思想

- 自定义脚本
- 自定义编排任务
- 根据业务进行脚本分类
- 标准化、流程化

- So. 几乎所有工作都可以采用脚本来完成

将运维工具：产品化

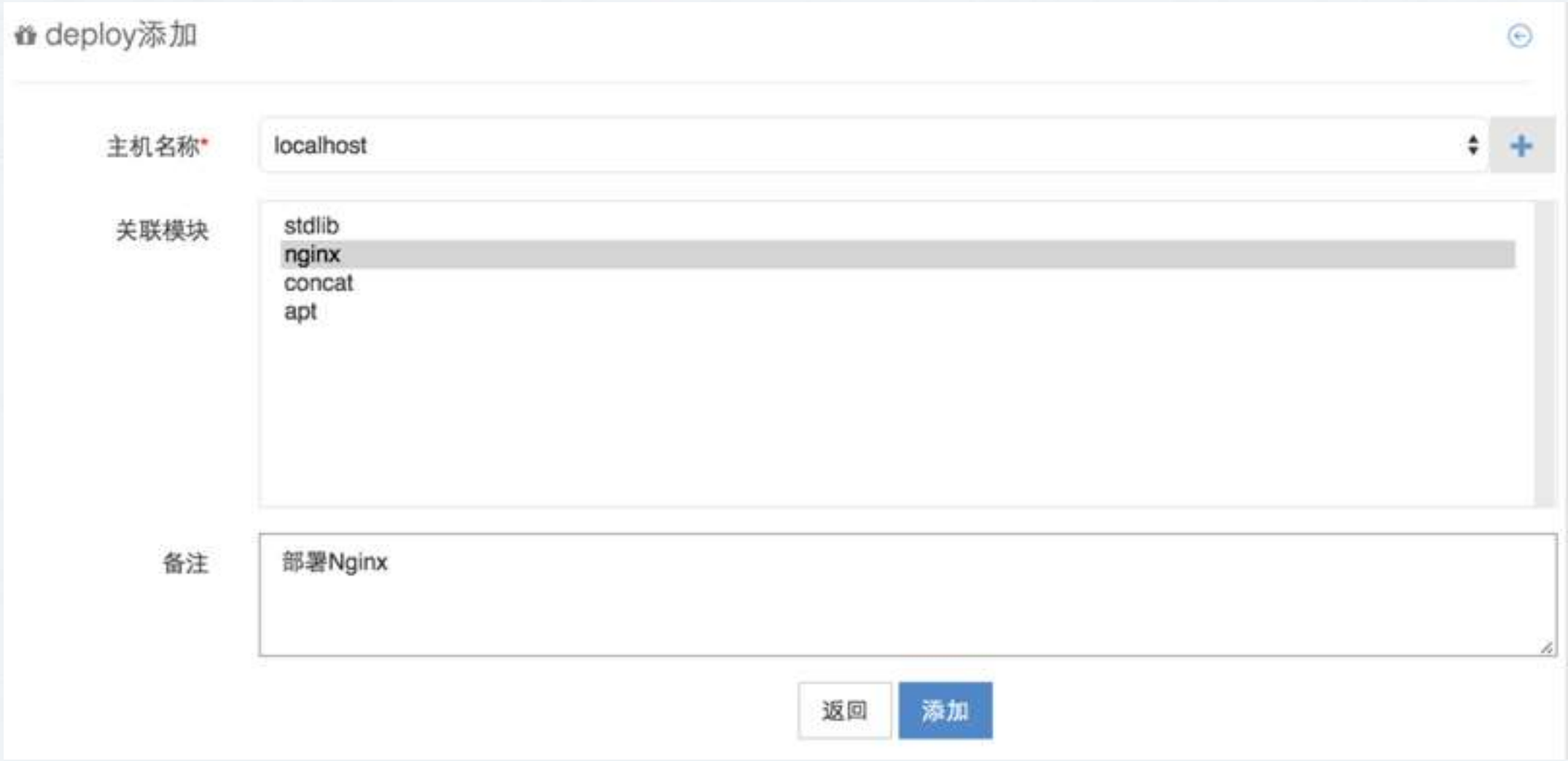
应用列表

任意字段

puppetmodule 4

应用名称	模块标示
stdlib	defalut
nginx	defalut
concat	defalut
apt	defalut

一键部署


deploy添加
主机名称*
localhost
关联模块
stdlib
nginx
concat
apt
备注
部署Nginx
返回
添加

中心思想

我们遵循一个理念，能用程序跑，就不去人操作。

集成平台

- CMDB
- 命令执行
- 配置管理
- 监控管理
- 流程管理
- 事件管理
- 等

架构是需要不断完善的
产品是需要不断磨炼的

感谢您参加本届MPD！

www.mpd.org.cn
400-812-8020