GOPS2016
Beijing

2016 全球运维大会 北京站

DevOps 2.0：重塑运维价值

会议时间：12月16日 - 12月17日

会议地点：北京国际会议中心

主办单位：

传统企业运维演进
&
开源实践

2016.12.15

北京　任明

Interstellar

Anything that can go wrong will go wrong.

Murphy’s Law --Edward A.Murphy

Do not go gentle into that good night.

--Dylan Thomas

运维发展阶段

企业运维技术演进

开源实践

运维发展阶段　方法定义

缤纷多彩的运维世界

运维覆盖这么多东西，如何系统性学习、分类

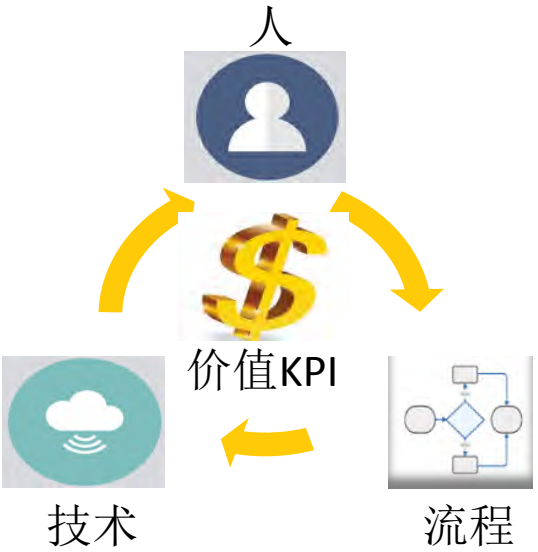
我们公司运维到底到什么阶段了，与外面比如何

我们还能如何优化自己的运维，下一步规划如何

开源到底是洪水猛兽，还是灵丹妙药

流程与技术如何平衡，人员组织如何分配

运维发展阶段　方法定义



人　人员组织、技能水平
文化与机制

技术　技术路线、技术架构

流程　流程管控

价值KPI　运维的价值KPI指标

通过对人、技术、流程的描述完成运维发展阶段的定义
并通过描述每个阶段的价值KPI标识运维的价值指标

运维发展阶段　阶段定义

Gartner 定义

Level0　survival
Level1　awareness
Level2　committed
Level3　proactive
Level4　service-aligned
Level5　business partnership

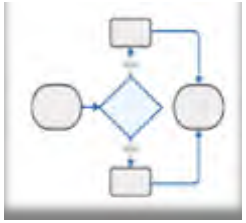
实践定义

Level0　0运维
Level1　全干工程师
Level2　初级分工
Level3　专业化分工
Level4　强流程管控
Level5　自动化/规模化
Level6　“0”运维/智能化

运维发展阶段　Level0　0运维

就这么几个
开发，0（专职）
运维

就这么几个
开发，0（专职）
运维



就这么几个
开发，要什么
运维流程

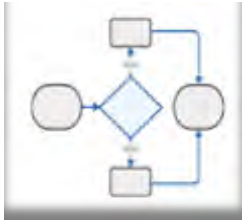
KPI是什么？
省了（专职）
运维的钱算么

企业生存是根本，如果业务都没有，系统好与坏有何区别

运维发展阶段　Level1 全干工程师

1-2人，跟着
开发一起玩
弄各种OS、DB
network、布线
写脚本、修门禁
又叫做系统管理员
网络管理员

基本开发都选好型了
有啥玩啥吧
Aix/linux、Oracle、db2、
informix、
Sybase、sql server、mysql
Tomcat、jboss、apache
Cisco、shell
服务器、存储、交换机、
路由器、机柜、加密机搬
起来
。 。 。

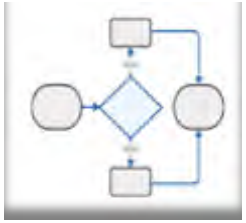


啥是流程？有版本
就陪着熬夜，出了
问题就赶紧救火

搞定技术工作
别经常宕机就行

为了确保开发的专业性以及增加运行的稳定性，终于有专职的运维了

运维发展阶段　Level2 初级分工



约3-20人
一个运维组/部
系统、网络、
基础环境等
专业分工

Aix/linux、Oracle、db2、informix、
Sybase、sql server、mysql
Tomcat、jboss、apache
Cisco、shell
服务器、存储、交换机、路由器、
机柜、加密机搬起来
。 。 。
会配置就行了
搞不定就到处问（穷人）
/找800（土豪）

变更写个邮件告知
老大
监控写个脚本短信
告警

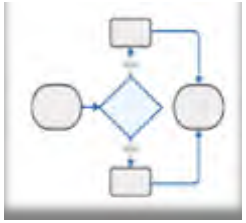
配合开发完成应用
上线
维护的对象不出问
题
省钱（土豪忽略）

随着运维应用及节点规模增长，有专门的运维团队了，内部也按专业有了技术细分，流程还没有显性化

运维发展阶段　Level3 专业分工

约20-70人，一个运维部
有系统（OS、DB Middle）、安全、网络、应用、基础环境等细化专业角色

Aix/linux、Oracle、db2、informix、
Sybase、sql server、mysql
Tomcat、jboss、apache
Cisco、shell
服务器、存储、交换机、路由器、机柜、加密机搬起来
各种技术安安装/配置规范
各种快速应急手册
编写大量脚本加快速度



知道有个叫ITIL/ITSM的流程
变更有了邮件/文档
监控有记录与报警
应急/巡检有记录

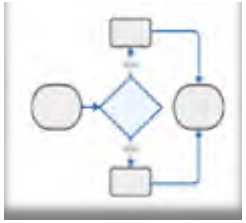
将N个9放在了KPI中
省钱（土豪忽略）
速度（别影响业务上线时间）
别出安全审计问题

运维团队有了更细化的分工了，有了变更、应急、监控报警，增加了大量技术脚本与文档，有明确的KPI指标（稳定、安全、效率、金钱）

运维发展阶段　Level4 强流程管控

约70-N百人（一般N小于5），2个数据中心
有系统（OS、DB Middle）、安全（审计、安全工具、安全管控）、网络（局域网、广域网）、应用、基础环境、流程、架构、值班/服务台等更细化专业分工

Aix/linux、Oracle、db2、informix、
Sybase、sql server、mysql
Tomcat、jboss、apache
Cisco、shell、堡垒机、时钟同步
服务器、存储、交换机、路由器、机柜、加密机搬起来
各种技术安装/配置规范
各种快速应急手册
编写大量脚本加快速度
上下线规范/高可用规范/监控规范
灾备建设



ITIL已经落在平台上了
对变更/服务（外部请求）/事件（监控告警）/问题/BCM/容量/应急处理与报告等有了明确的处理要求

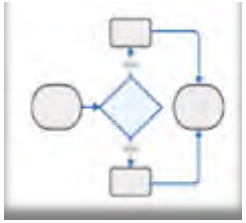
将N个9放在了KPI中
省钱（土豪忽略）
SLA速度/质量（提高运维效率）
别出安全审计问题
每年技术项目完成

团队的快速扩容形成了通过平台进行强流程控制（一般为ITIL）的数据中心管理架构，增加了事件、问题、业务连续性、容量性能、应急处理与报告等明确流程机制，有明确的KPI指标（稳定、安全、效率、金钱、项目）

运维发展阶段 Level5 规模化/自动化

约100-N百人（一般N小于10），N个数据中心
有系统（OS、DB Middle）、安全（审计、安全工具、安全管控）、网络（局域网、广域网）、应用/业务分析、基础环境、流程、架构/平台开发、值班/服务台等更细化专业分工，招人有了明确的开发要求

Aix/linux、Oracle、db2、informix、Sybase、sql server、mysql、Tomcat、jboss、apache、Cisco、shell、堡垒机、时钟同步
服务器、存储、交换机、路由器、机柜、加密机搬起来
各种技术安装/配置规范
各种快速应急手册
编写大量脚本加快速度
上下线规范/高可用规范/监控规范
灾备建设
建设开源软件维护机制
建设云计算/运维平台等将提高开发运维效率，通过场景化切入
通过平台大量降低人员数量和质量依赖



ITIL已经落在平台上了
对变更/服务（外部请求）/事件（监控告警）/问题/BCM/容量/应急处理与报告等有了明确的处理要求
与开发对高可用、性能、持续发布等进行了有效结合
由于业务快速发展的压力，对创新业务一定程度上做了适应性变化（devops/轻量级ITSM）

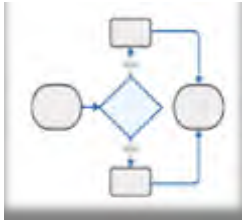
将N个9放在了KPI中
省钱/人（资源利用率、单人头节点维护数）
速度（整个开发运维效率）
给出运营数据的分析及业务优化建议
别出安全审计问题
每年技术项目完成

为了适应业务的快速增长与变化以及维护对象的几何级增长，通过构建平台将运维场景进行平台化服务（自助）提供，对创新业务有了快速的流程适应，KPI中增加了对资源利用率、单人头维护数等要求，能分析运营数据并向前端反馈建议

运维发展阶段　Level6 “0”运维/智能化

硬件维护团队（可外包）
（开源）软件及平台构建团队
应用运维/运营团队
安全团队
只招聘开发运维人员

通过将OS、DB、MIDWARE、应用运维等各种运维工作进行平台化的OAAS（operation as a services）并提供给开发部门
实现datacenter as a machine



devops
通过平台智能化实现数据的运维一线（自动化）、二线（智能化）的场景构建与数据交互

将N个9放在了KPI中
省钱/人（资源利用率、单人头节点维护数）
速度（整个开发运维效率）
给出运营数据的分析及业务优化建议
别出安全审计问题
每年技术项目完成
创造新的技术赢利点/技术输出
开源社区贡献度

一般维护节点数已过10万，几乎看不见传统运维的身影，高科技公司基因，运维的最终幻想

运维发展阶段

企业运维技术演进

开源实践

企业运维技术演进 四年前

企业运维技术演进 四年前

企业运维技术演进 四年前

企业运维技术演进 四年前

企业运维技术演进 四年前

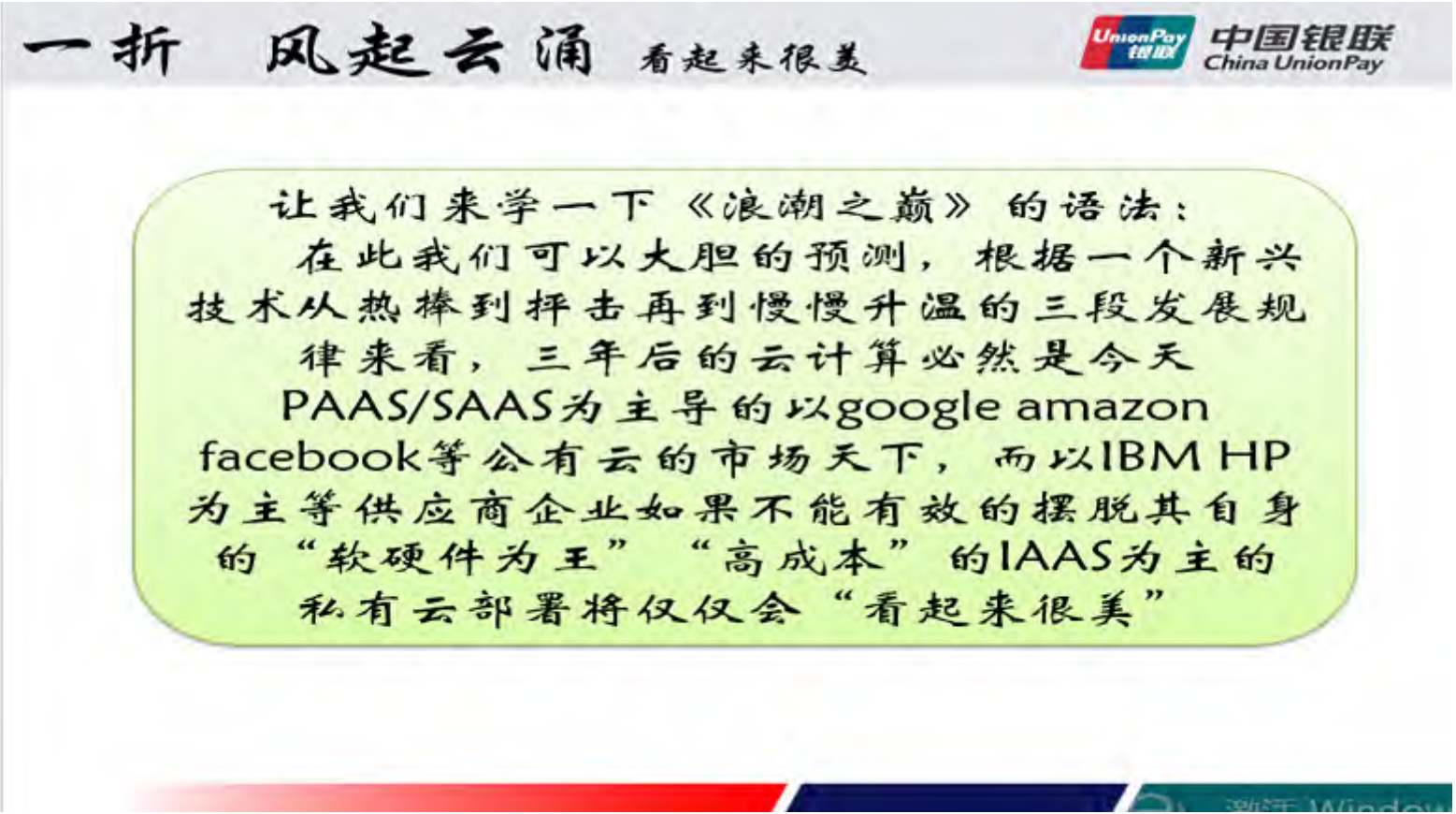
楔子 必然的产物 云计算

中国银联 China UnionPay

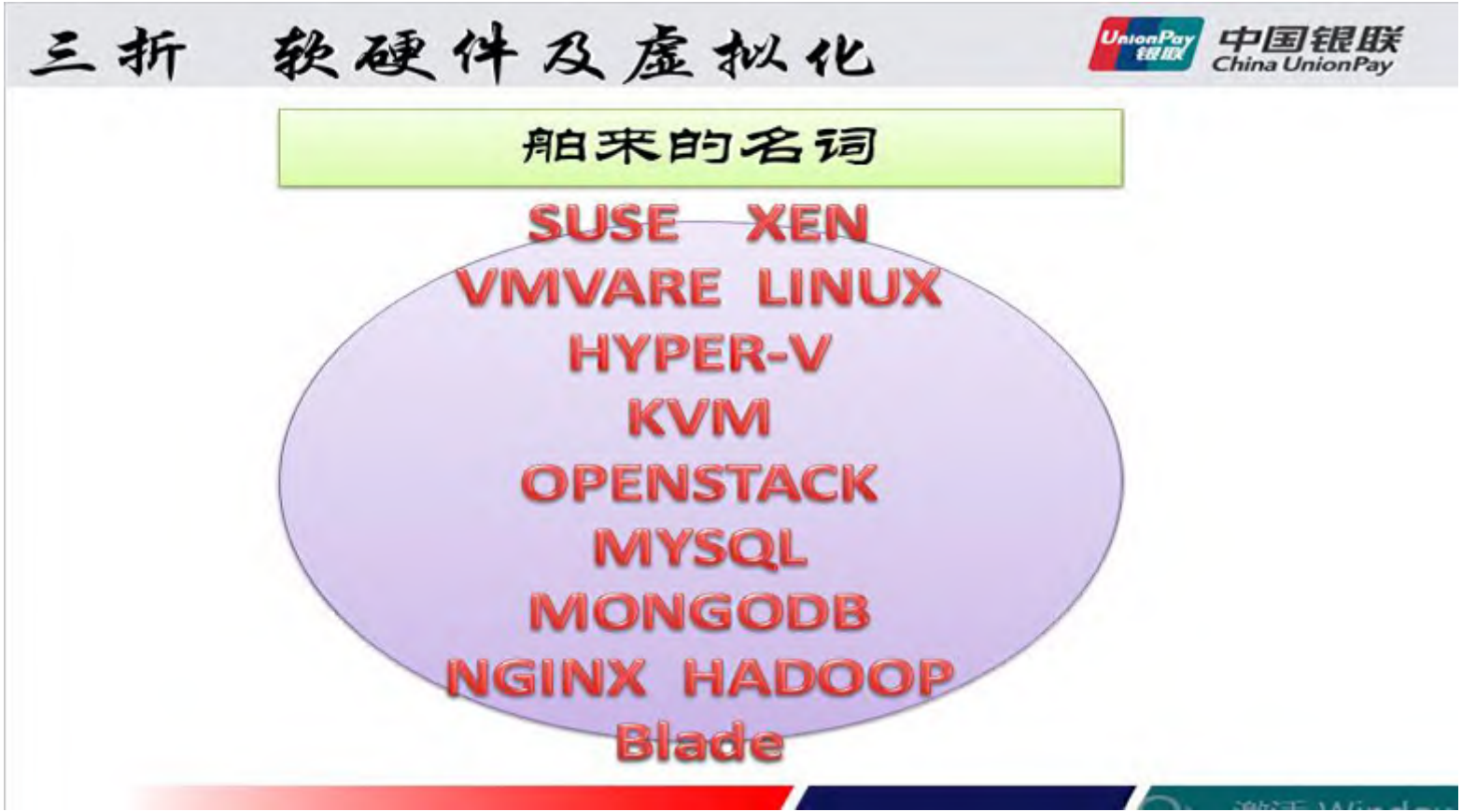
云计算是IT产业发展至今的必然产物

1.虚拟化计算的发展
2.网络技术的发展
3.计算机理论的发展
4.软硬件性能的发展
5.IT商业模式的发展
6.开源软件的发展
7.绿色IT的需求
8.以业务为导向的需求
。。。

企业运维技术演进 四年前



企业运维技术演进 四年前



企业运维技术演进 技术发展历程

国家战略：自主可控 | 国产化 | 云计算

公司战略：高科技 | 云计算/大数据/移动互联网 | 二次创业

业务市场变化：更快更灵活的应用系统

更低的成本 | 更快的版本需求 | 更好的UED/UI

更便捷的接入 | 更快的后台响应 | 更稳定的系统

FinTech 互联网+

开源自主 云计算大数据

企业运维技术演进 技术发展历程

业务特点变化

从后台到前台

从2B到2C

2B

第三方支付　　银联　　银行　　行业方

典型业务：转接、收单

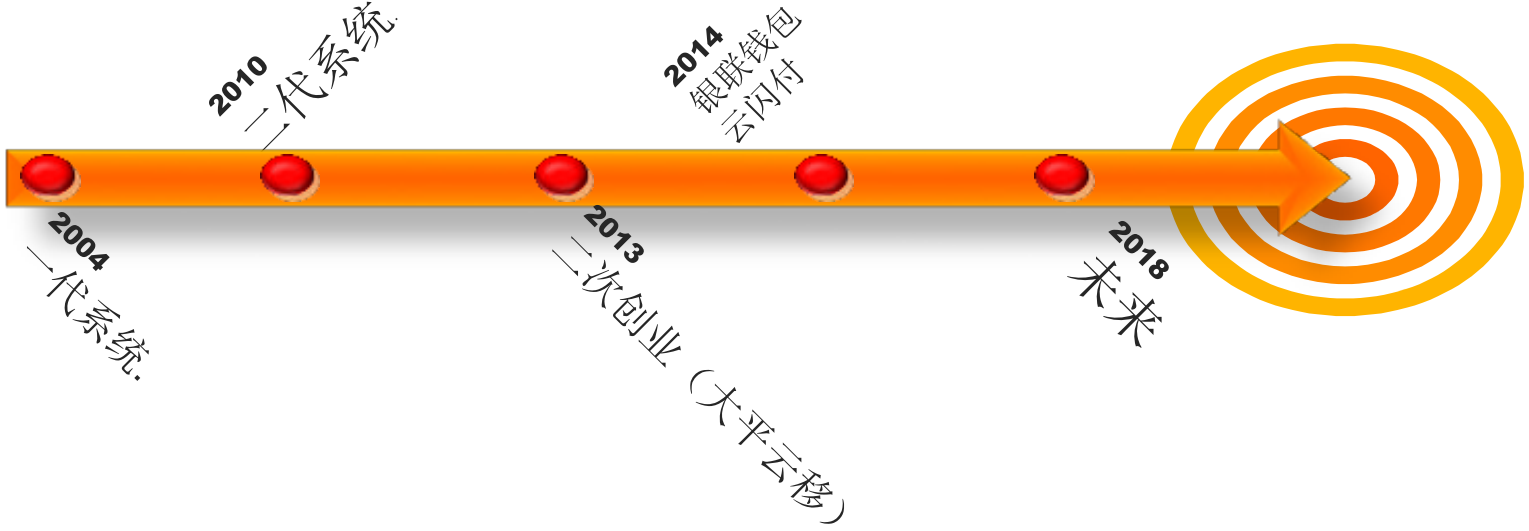
2C

第三方支付　　银联　　银行

持卡人　　行业方

典型业务：银联钱包、全渠道、APPLY PAY

企业运维技术演进 技术发展历程



业务发展决定了技术发展走向

单一 到多元　　版本快速迭代

互联网化　　2B（机构） 2B（商户） 2C（持卡人）

运维发展阶段

企业运维技术演进

开源实践

四年前

技术发展历程

传统与开源之争

开源成熟度评估模型

机制与文化

开源实践 传统与开源之争

为什么要使用开源
金融行业这么重要，敢上生产么
以前出了事有厂商顶着，用了开源谁顶
老的技术大家都熟悉了，新的东西没动力学习
开源这么多坑，有些又没出来几年，能用么

开源实践 传统与开源之争

	传统模式	自主模式（devops）
人员结构	运维人员+软硬件厂商+集成商	运维人员+开发人员
人员技能	运维人员以规划、配置、管理、事件处理为主；代码、原理问题依靠厂商或集成商	运维人员进行规划、设计、配置、管理、事件处理，代码/原理问题处理；开发人员可以修复bug、实现新功能
应急时效	配置问题处理较快；bug类处理时效慢，需要到国外解决，无法深入掌握原理	所有问题可以自主分析、快速修复、及时解决。
需求实现	依靠厂商产品实现，与实际需求不能完全匹配，无法完全满足实际需求。	自主开发实现，可以完全匹配实际需求。实现效率快。
规划与优化	自行研究成熟的软硬件产品、方案+传统行业交流厂商推介	自行研究+开源社区+互联网行业交流，通过代码级的POC选择合适的产品或方案
拥有成本	价格由厂商控制，通常与部署数量强关联，初期成本及维保成本较高较高 需要采购流程	自行开发，成本主要是人力成本，通常与部署数量弱关联
知识产权	知识产权厂商掌控	自主掌控
技术实力	应用技术实力较强， 没有底层软硬件技术实力	从平台到组件、到应用均有强大的技术实力
平台集成	受限于各厂商产品，无法实现完全的平台集成	可以实现完全的自主平台，架构整体性强

开源实践 传统与开源之争

“双态”并存（双模IT）

传统	开源
稳态	敏态
ITIL	devops
核心系统	创新系统

开源实践 开源成熟度评估模型

开源软件成熟度评估-需求与目标

开源软件成熟度评估的需求：

- 越来越多开源软件在运用，为了更加安全稳定使用开源软件，需要对其成熟度进行评估，主要包括三个角度：
  - 需要对代码进行评估，从而了解开源软件内部质量。
  - 需要对社区进行评估，从而知晓其发展质量与参与度。
  - 需要对软件授权许可协议进行评估，从而规避潜在的产权、法律风险。

开源软件成熟度评估的目标：

- 形成一套模型方法，对开源软件的质量、成熟度、发展、管理进行评价。
- 为开源软件在生产长期稳定的选项、应用、推广提供合理的参考依据。

开源实践 开源成熟度评估模型

开源软件成熟度评估-业界发展

开源软件的成熟度评估，在国际上已经提出多种模型

名称	OSMM-C	OSMM-N	QSOS	OpenBRR	OMM
始于	2003	2004	2004	2005	2008
创始团体	Capgemini公司	Navicasoft公司	AtosOrigin公司	卡梅隆大学，Intel公司	欧盟QualiPSo项目
评估模型	实际应用	实际应用	实际应用	科研	科研
技术/功能规范	无	无	有	有	有
评分模型	1-5级	1-10级	0-2级	1-5级	1-4级
权重规范	是	是	是	是	是
结果比对	是	否	是	否	否

OSSM:Open Source Maturity Model开源软件成熟度模型
QSOS:Qualification and Selection of Open Source software开源软件资评与选择模型
OpenBRR:Open Business Readiness Rating开源代码商业完备度评价模型
OMM:Opensource Maturity Mode 开源成熟度模式

开源实践 开源成熟度评估模型

OSSM-C

- OSSM Capgemini模型的主要评估指标和因素

因素	释义
可用性	软件功能和操作是否符合用户预期
应用接口	如果需要与用户现有系统整合，该软件是否有能力提供整合接口
性能	预期负载和处理能力
可靠性	产品应该具备什么级别的可靠性
安全	什么安全的措施是必要的，有什么必要的限制
成熟的技术	是否该软件产品使用的技术已经广泛应用于生产
厂商独立性	供应商与用户之间需要什么样等级的保障与承诺
平台独立性	软件产品是否只是用于特殊的信息和通信技术环境，或可以使用在更多的平台上
支持	必须提供什么级别的应用支持服务
报表	需要什么形式的应用支持服务
管理	产品是否允许使用用户现有的维护工具，是否符合业务管理的要求
建议	用户是否需要第三方对产品进行验证，提出建议，如果需要，验证什么
培训	必要的培训和设施
人员编制	用户是否需要产品专家作为专业技术顾问、教学人员或维护者
实施	对用户来说，哪种实施方案是最佳的

该模型强调软件特性是否符合用户需求，对于软件质量及成熟度覆盖较少，在开源社区很少应用

开源实践 开源成熟度评估模型

OSSM-N

- OSSM Navica评估模型通过6个条件的评价分值经加权求和得到开源软件的成熟度：

- Software(软件)
- Support(支持)
- Documentation(文档)
- Training(培训)
- Integration(整合)
- Professional Services(专业服务)

该评估模型覆盖了软件的功能性、易用性、代码质量、维护支持、技术文档、社区建设等领域，在一定程度上反映了软件的成熟度；但对于软件性能、安全性、可扩展性等方面则考虑较少，尤其是缺少对开源软件许可协议实用性的分析。该模型在开源社区使用较少。

开源实践 开源成熟度评估模型

QSOS（1）

➢ QSOS是一个由开源社区发展维护的开源软件成熟度评估方案，QSOS评估模型由四个步骤组成：

该评估模型拥有一个活跃的社区（www.qsos.org），帮助项目在评估方案和评估工具上不断改进，并系统的针对各种开源软件成熟度评估进行实践。是目前开源软件质量评估方面最流行的方案之一。

1. 定义：定义在以后步骤中用到的评估准则和评估要素。
2. 初步评估：按照评估准则，基于评估条件和评估环境利用测试或收集的数据对软件进行评估，包括功能覆盖、用户的风险和服务提供商的风险三个层面。
3. 用户条件：根据用户需求制定选择软件的标准，并将不同权重指标根据情况分布于不同的轴层面。
4. 软件选项：根据前三步的结果吗综合对比么选择满足需求的软件。

开源实践 开源成熟度评估模型

QSOS（2）

四个步骤的具体内容

定义阶段：

- **关注的问题**
  - 该软件发展的历史。
  - 授权许可类型。
  - 社区类型和社区建设情况。
  - 软件功能。
- **具体评估内容**
  - 基本信息：软件名称、参考、发现时间、作者、软件类型、基本描述、相关许可协议、兼容操作系统、派生与来源历史。
  - 现有服务：文档信息、现有技术支持提供者、现有培训提供者、现有咨询提供者。
  - 功能与技术：应用技术、技术前提、详细功能描述、路线图。
  - 综合方面：软件发展大体趋势、释义。

评估过程阶段：

- **评测**：指标评测、数据搜集/统计。
- **打分**：依据前面定义的准则对相关指标进行打分。
- **原生风险**：评估该开源软件的原生风险。

用户条件阶段：

- **功能列表筛选**：根据每个功能在需求中的不同要求来筛选，包括：必须功能、可选功能、非所需功能。
- **用户风险筛选**：根据需求的不同相关风险标准被分布在不同的轴层，包括不相关标准、相关标准和临界标准。
- **服务提供者风险筛选**：供服务提供商使用，评价软件和相关服务能够以何种层次进行集成。

软件选型阶段：

- **严格选择**：只要有软件不符合第三步的情况就立刻剔除。
- **宽泛选择**：相对于前一种比较宽泛，不排至不合标准软件而是将它们进行分类，并有一套自己的权重体系。

由于QSOS自身是一个开源项目，其本身开发了一些了评估工具，包括：模板编辑器，工作表编辑器，验证、比较、展示控制引擎、CVS数据仓库等。

开源实践 开源成熟度评估模型

OpenBRR

- OpenBRR由Intel于2005年提出，其评估条件分11类，采用1-5的5级度量，采用加权求和的方式计算最终结果，评估分为四个阶段：

A. 软件初步过滤
B. 裁剪与确定评估指标模板
C. 搜集评估原始数据并度量
D. 通过加权和计算软件的评估值

十一类评估条件：

- 可用性
- 软件开发质量
- 安全性
- 性能
- 可扩展性
- 架构设计
- 服务与支持
- 文档
- 接受度（知名度）
- 社区建设
- 专业化程度

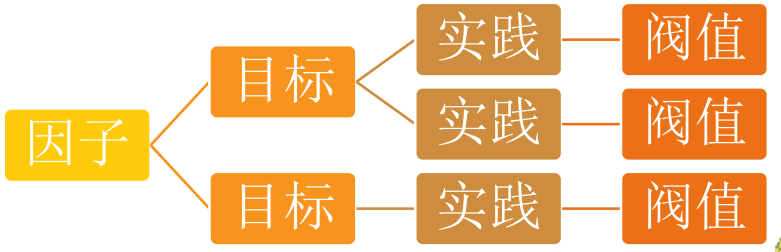
由于OpenBRR的评估模型项目没有建立自己活跃的网络社区，使得模型的推广、应用受到一些影响，
其范围较QSOS模型比较小

开源实践 开源成熟度评估模型

OMM

- OMM评估模型是欧盟第六框架重大项目中开源软件质量保障平台(QualiPSo)项目提出的开源软件成熟度模型；同时，该项目还基于此模型设计开发了一个软件过程自动评估原型系统。OMM模型分为两个阶段：
- 第一阶段：通过调查问卷、访问等信息得到影响开源软件质量的12个评判因子，包括：产品文档、产品认可度、开发标准、项目进度表的应用、测试计划的质量、参与人员的管理、许可证、技术开发环境、缺陷数目和错误报告、可维护性和稳定性、软件公司对开源软件的贡献、第三方公司评估结果。以及从CMMI中挑选符合开源软件质量评估的因子。
- 第二阶段：OMM定义采用了类似于CMMI模型的方法，将每个因子分解为一个或多个目标，每个目标在分解成一个多多个具体实践，每项实践设置阀值，最后计算各项时间的分数达到标准则通过，形成OMM树形结构。通过一些自动评测工具进行跟踪，呈现统计图表给用户或开发人员及时发现问题并修改。

与前几种成熟度评估模型项目相比，OMM主要服务于对软件开发过程中成熟度要素的考察。



开源实践 开源成熟度评估模型

业界模型总结

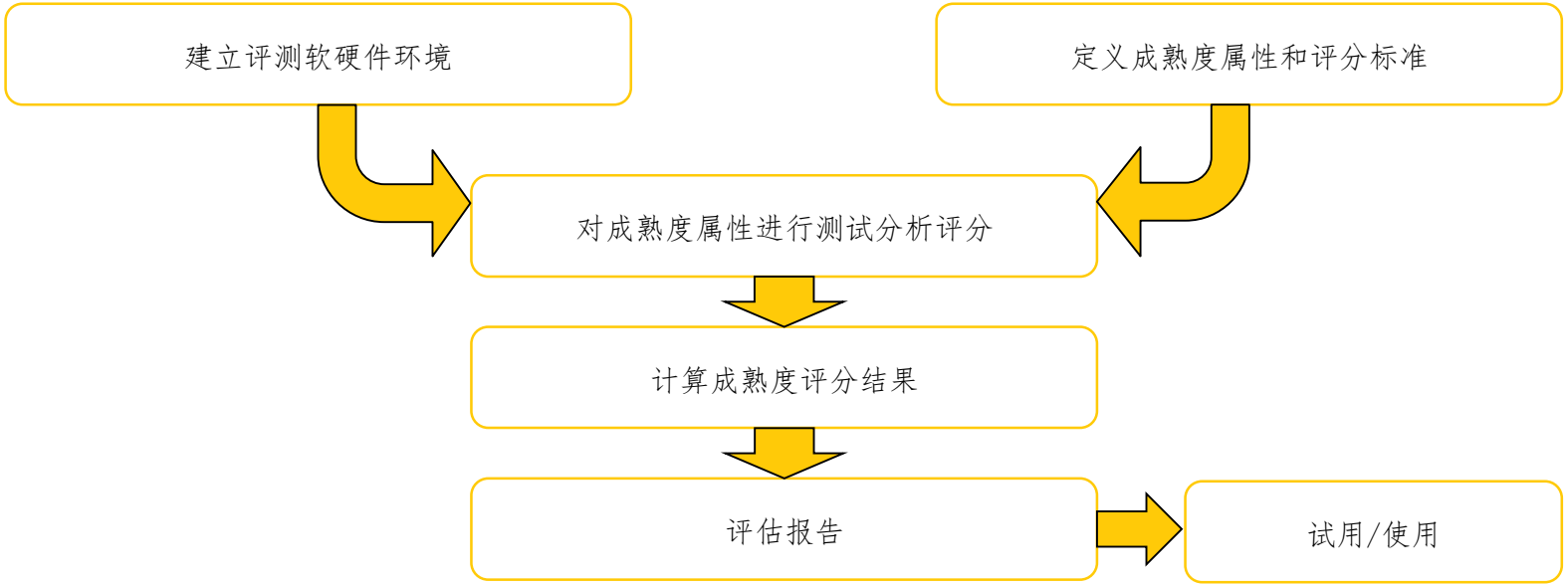
OSMM-C	OSMM-N	QSOS	OpenBRR	OMM
对于软件质量及成熟度覆盖较少。 在开源社区很少应用	对于软件性能、安全性、可扩展性、开源协议等方面则考虑较少。在开源社区使用较少。	拥有一个活跃的社区帮助项目在评估方案和评估工具上不断改进。 是目前开源软件质量评估方面最流行的方案之一。	没有建立自己活跃的网络社区，使得模型的推广、应用受到一些影响，其范围较QSOS模型比较小。	OMM更重视对软件开发过程中成熟度要素的考察。

- 我们参考业界已有的一些开源软件成熟度评估模型，结合以往的评估经验，设计了一个开源软件成熟度完整评估方法和一个开源软件成熟度敏捷评估方法。

开源实践 开源成熟度评估模型

开源软件成熟度完整评估方法

- 该方法首先建立评测环境、定义属性及评分标准；然后通过测试及分析进行成熟度试算；最后出具评估报告，评审后进行使用或试用。



开源实践 开源成熟度评估模型

完整评估方法—评测环境

评测硬件环境

- 根据待评估开源软件的类型、用途搭建相应的测试环境。
- 同类型软件比较时，评估环境尽量相同，兼顾各类软件专有的特性。

评测软件环境

- 通过代码评估工具探测代码级的安全问题和可能的不良代码植入。
- 通过黑箱测试工具反映用户直观体验状况。

开源实践 开源成熟度评估模型

完整评估方法—评分标准

✓ 我们通过给属性类、属性设置权重；属性评估结果进行打分的方式；综合进行开源软件成熟度评估。

- 属性权重设计

W1（权重）	0	1	2	3	4
评价	不考虑	不重要	一般	重要	非常重要

- 属性评估结果打分设计

U(分值)	0	1	2	3	4	5
评价	无	差	较差	一般	较好	好

- 属性类权重设

W2（权重）	1	2	4
评价	不重要	较为重要	重要

开源实践 开源成熟度评估模型

完整评估方法—评估报告

✓ 完成了测试、评分、计算工作后，建议根据实际情况撰写开源软件成熟度评估报告（或选项报告），建议包含如下部分：

软件介绍
- 介绍历史、功能、兼容平台等内容

软件架构分析
- 介绍架构、模块、技术路线对外借口等软件设计问题。

软件代码分析
- 通过测试及分析生产代码评测报告

软件“五高”分析
- 对五高情况进行评测分析

社区分析
- 分析软件的社区发展及支撑力度

应用与发展分析
- 分析软件的实际使用及商业支持情况

法律风险分析
- 分析版权及协议相关内容

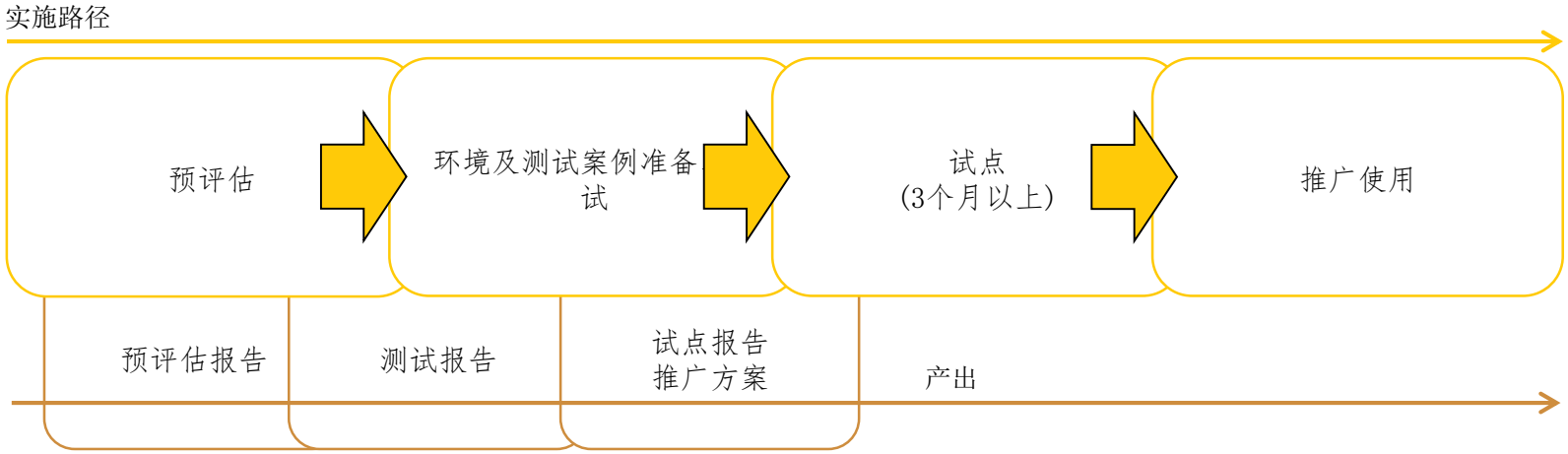
选项建议及落地方案
- 结合各项评估给出选项建议及落地方案

相关资料附录
- 测试报告、调研报告、许可证协议等辅助信息

开源实践 开源成熟度评估模型

敏捷评估方法

- 开源成熟度评估完整评估方法在实际操作过程中消耗的时间比较长，适合大型开源软件的选型活动，例如xen/kvm，suse/redhat。
- 结合近期开源软件的使用经验，结合devops、灰度等思想，设计了一个开源软件敏捷评估方法。



开源实践 开源成熟度评估模型

敏捷评估方法

- 各阶段评估点

预评估

属性类名称	属性名称
技术架构	模块划分
	组织架构社区管理
	社区支持度
社区活跃度	社区测试团队情况
	版本发布周期
	软件下载量
	商业公司支持度
	企业应用案例
应用与发展	开源协同发展
	未来路线规划
	版本正式程度
	兼容性
落地可行性	替换性
	人员支持

明确是否适合银联业务，是否需要POC测试。

环境及测试案例准备、测试

属性类名称	属性名称
技术架构	架构合理性
	对外接口
	高性能
	高可用性
软件“五高”	高可管理性
	高可靠性
	高安全性
代码质量(如准备自己二次开发)	代码错误率
	代码可读性
	代码复杂度
	兼容性
落地可行性	替换性
	人员支持

完成测试选型，明确是否落地，选取试点应用。

试点

属性类名称	属性名称
生产分析	稳定性分析
	容量分析
	高可用分析
法律问题	软件版权协议
	商标、专利与纠纷
	商业公司支持
应用与发展	未来路线规划
	版本正式程度
	兼容性
落地可行性	替换性
	人员支持

试点三个月以上，结合试点情况，明确是否推广、推广范围、开发模型、运维模式。

推广使用

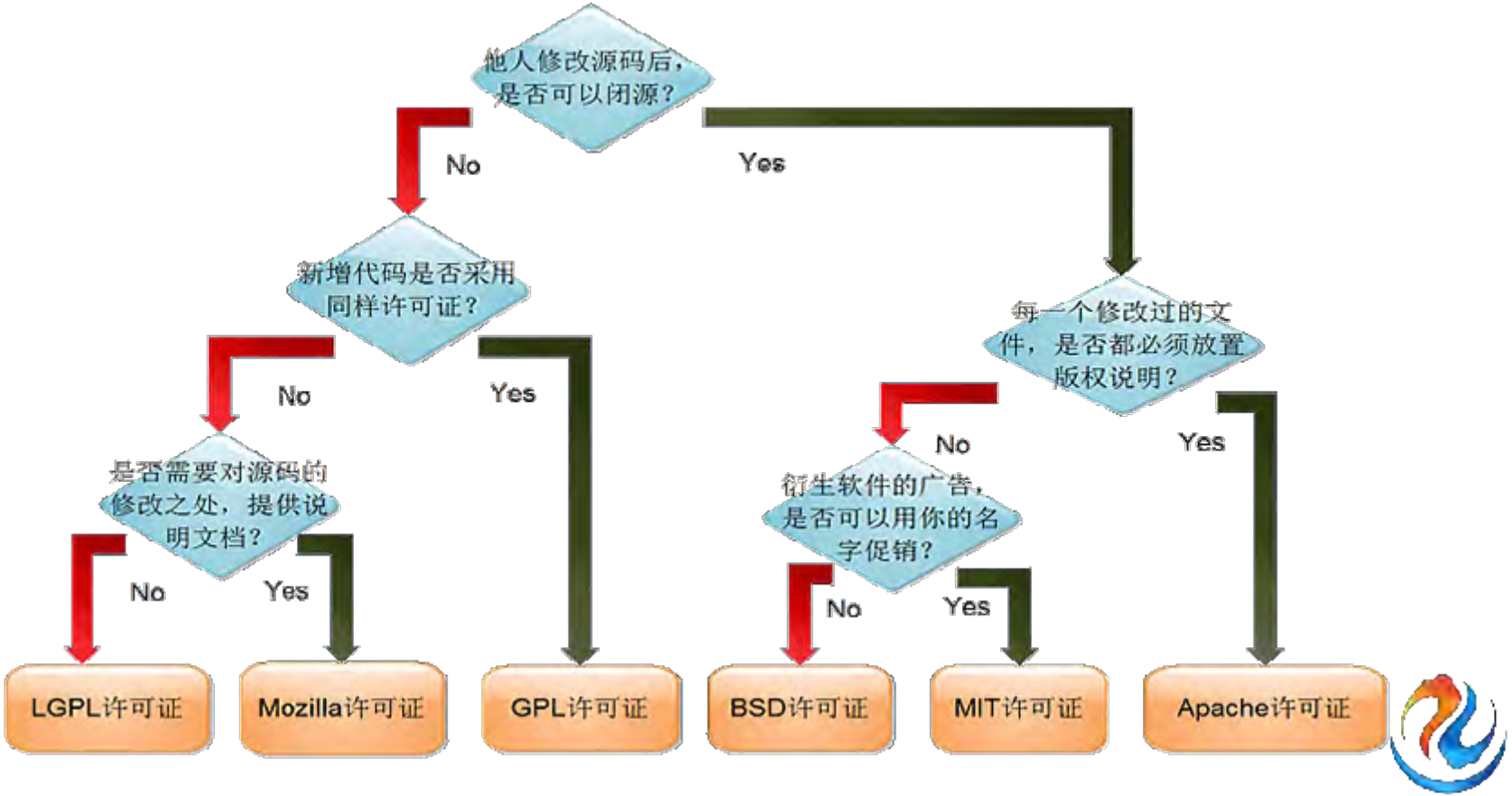
按照预定义的开发运维模式使用、优化该开源软件。

开源实践 开源成熟度评估模型

- 对于大型软件或运用于核心系统的软件试行完整开源软件成熟度模型。OS、DB、Storage、hypervisor、cloud等通用基础性软件
  例如centos、mysql、ceph、kvm、openstack等

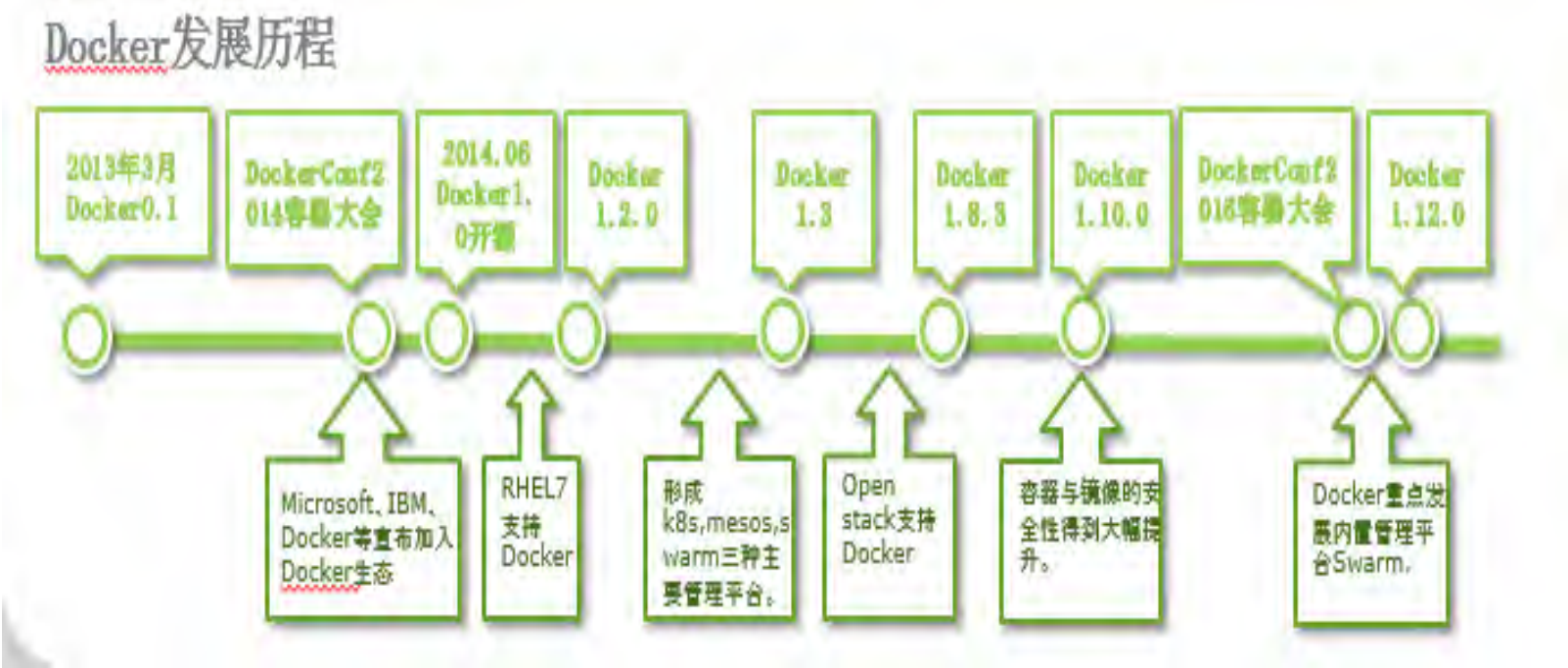
- 对于中小型软件或运用于外围系统的软件试行敏捷开源软件成熟度模型。
  例如bind、haproxy、apache等

开源实践 开源成熟度评估模型



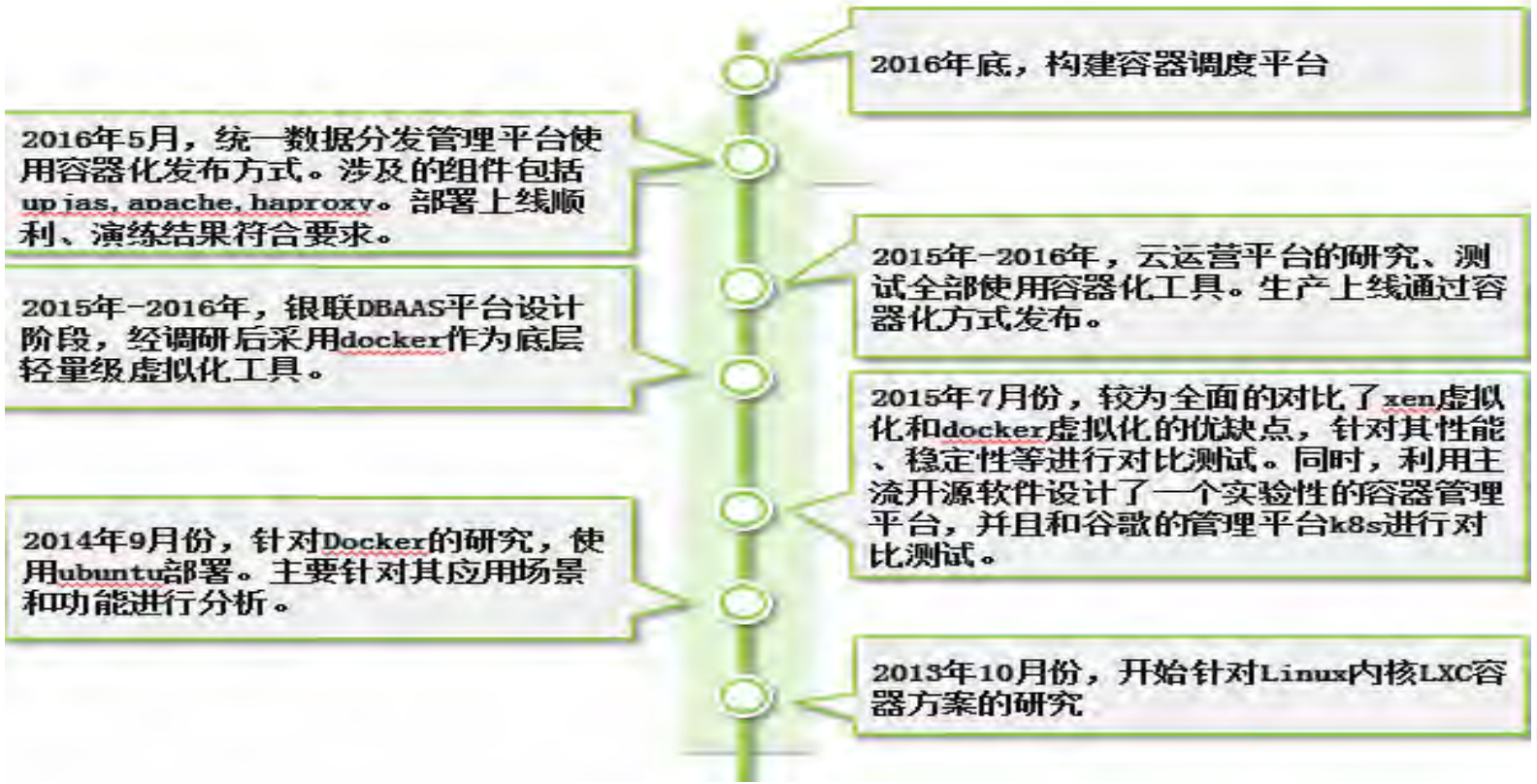
开源实践 开源成熟度评估模型

DOCKER举例



开源实践 开源成熟度评估模型

DOCKER举例



开源实践 开源成熟度评估模型

openstack举例

附件名称	附件大小(KB)
ACI SDN与OpenStack对接方案1023.docx	609
Ceph方案1023.docx	478
Keystone与SSO认证集成方案1023.docx	178
NAS集成方案1023.docx	668
大规模部署方案1023.docx	2133
单物理bond口多网络平面的划分和QoS方案1023.docx	156
高可用方案1023.docx	365
双中心管理方案1023.docx	523
银联容器平台集成方案1023.docx	512
云运营平台对接方案1023.docx	72
云资源平台升级方案1023.docx	236
自动化部署方案1023.docx	428
硬件设备建议方案1023.docx	64
01-评审意见.xlsx	16

开源实践 开源成熟度评估模型

HAPROXY举例

	Haproxy	LVS
优点	1、支持4层、7层负载均衡 2、配置丰富，功能多	1、支持4层TCP、UDP服务 2、性能好
缺点	不支持UDP服务	1、不支持7层应用负载均衡 2、可配置参数少，功能少
可用范围	四层负载均衡（TCP） 七层负载均衡	四层负载均衡 （TCP、UDP）

开源实践 机制与文化

公司级支持

开发运维有效协作

分享、鼓励

包容、开放、试错

开源实践 机制与文化

从招聘开始

开源研发	1、upsql、updis、openstack、ceph ... 2、ELK、ansible、docker、haproxy、bind 3、社区、云计算

3、对IT软件系统运维/测试/信息安全的新技术的学习，研究及引入（如Hadoop、Docker、OpenStack、DevOps、ELK等或各类数学模型和建模工具），持续提升针对IT/云软件系统的运维/测试质量及效率；

4、攻城狮们，这里，可以发挥你们举天之力，GOOGLE全球宕机，5分钟就恢复，你们一台PC机故障，要多少时间才能恢复，这就是你们的天地，python/java/开源/云计算，你就是full stack devops；

开源实践 机制与文化

开源推进小组
&
会议

开源机制的制定与优化
开源相关工作的协调

源享会

意为“开源、分享”
对推进中或将要推进的开源工作的
产出物（成果）的分享、学习、讨论

源享会

2. 如何开始你的开源代码阅读之旅？

运维人如何读源码

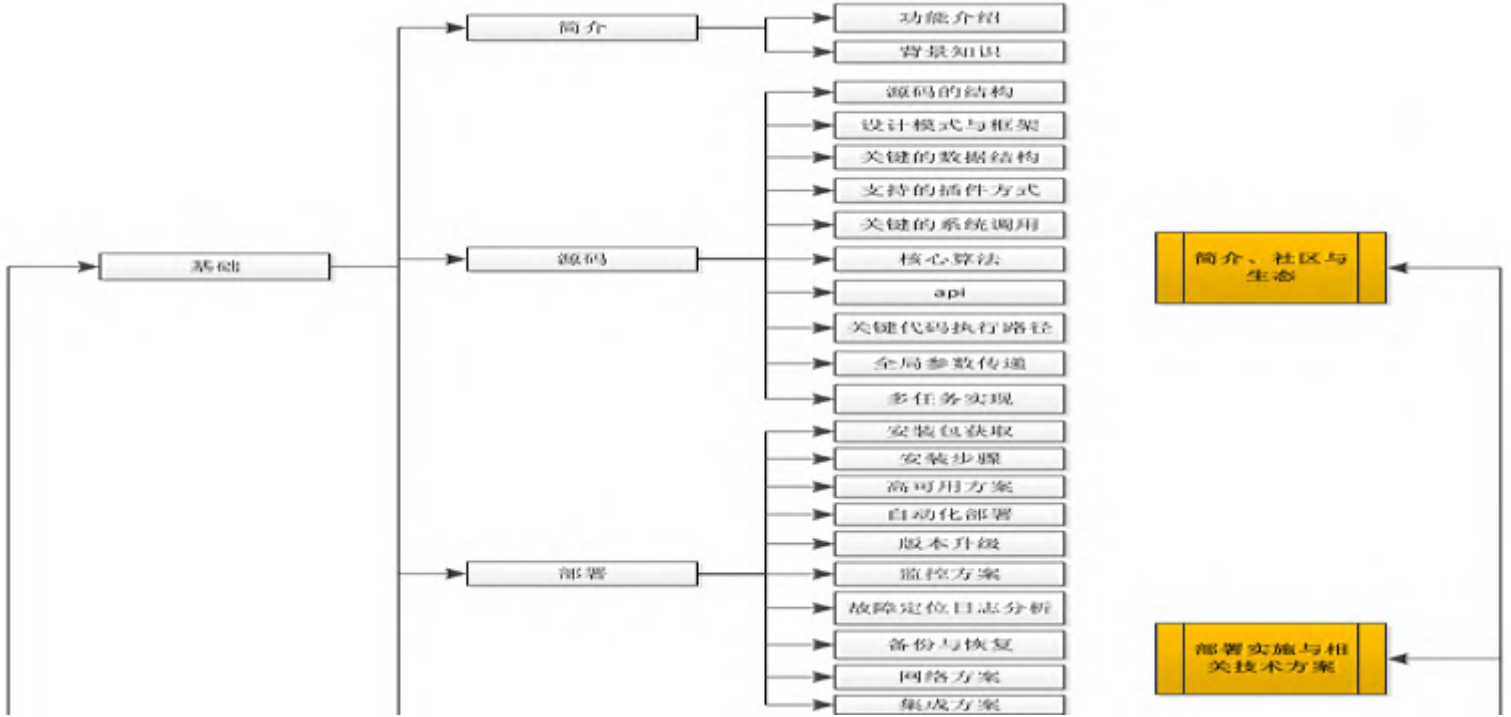
- 我们运维人的优势；
- 你离可以读懂源码还缺少哪些技能？
- 哪些技巧让我们事半功倍？
- 如何展现我们的阅读成果？

3. Ansible的结构解读与生产应用。

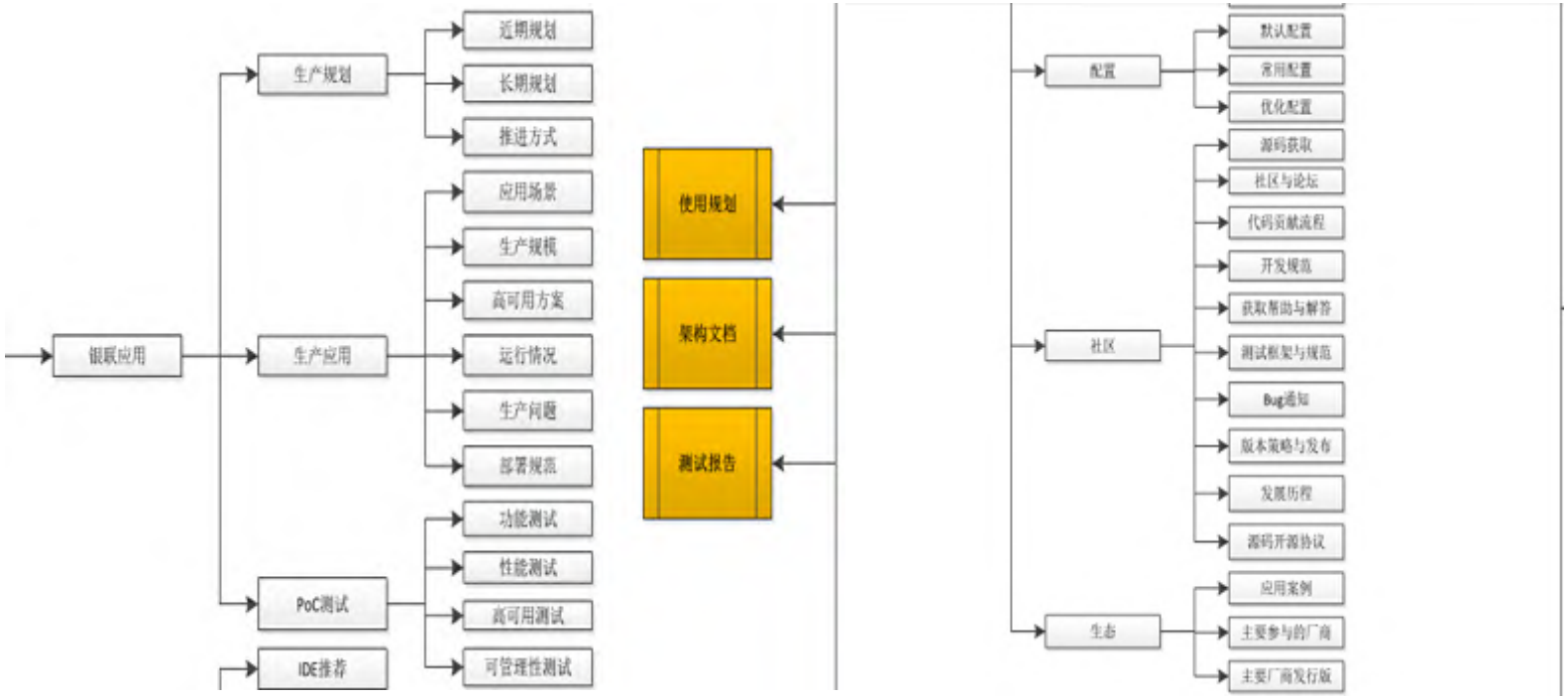
Ansible在生产上已经实现了管理几千台服务器的任务。

- Ansible难吗？
- Ansible的关键代码在哪里？
- 如何给Ansible增加新功能？
- 如何利用Ansible实现大并发？

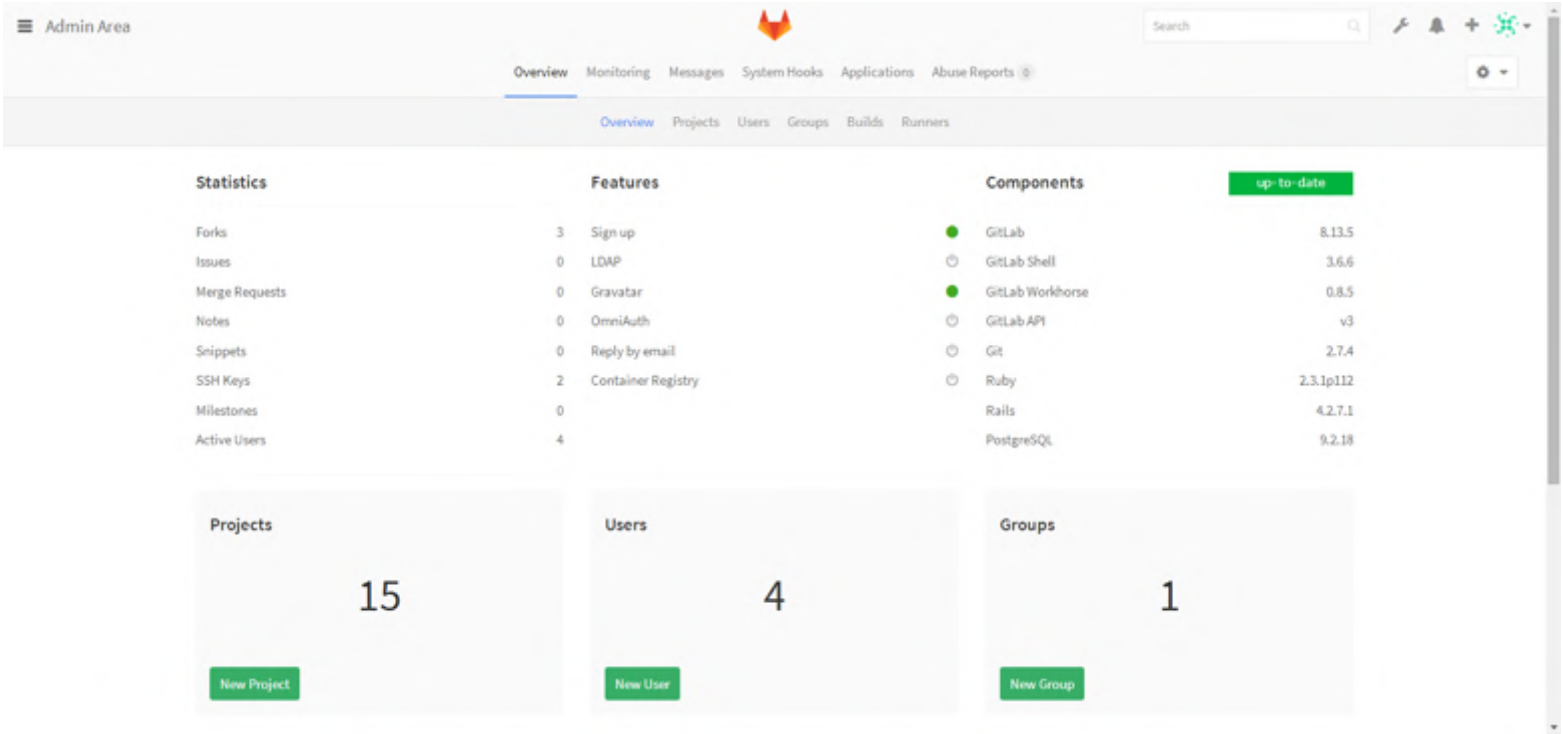
产出物质量


基础
简介
源码
部署
功能介绍
背景知识
源码的结构
设计模式与框架
关键的数据结构
支持的插件方式
关键的系统调用
核心算法
api
关键代码执行路径
全局参数传递
多任务实现
安装包获取
安装步骤
高可用方案
自动化部署
版本升级
监控方案
故障定位日志分析
备份与恢复
网络方案
集成方案
简介、社区与生态
部署实施与相关技术方案

产出物质量


银联应用
生产规划
生产应用
PoC测试
IDE推荐
近期规划
长期规划
推进方式
应用场景
生产规模
高可用方案
运行情况
生产问题
部署规范
功能测试
性能测试
高可用测试
可管理性测试
使用规划
架构文档
测试报告
配置
社区
生态
默认配置
常用配置
优化配置
源码获取
社区与论坛
代码贡献流程
开发规范
获取帮助与解答
测试框架与规范
Bug通知
版本策略与发布
发展历程
源码开源协议
应用案例
主要参与的厂商
主要厂商发行版

工具-gitlab


Admin Area
Search
Overview
Monitoring
Messages
System Hooks
Applications
Abuse Reports
Overview
Projects
Users
Groups
Builds
Runners
Statistics
Features
Components
up-to-date
Forks 3
Issues 0
Merge Requests 0
Notes 0
Snippets 0
SSH Keys 2
Milestones 0
Active Users 4
Sign up
LDAP
Gravatar
OmniAuth
Reply by email
Container Registry
GitLab 8.13.5
GitLab Shell 3.6.6
GitLab Workhorse 0.8.5
GitLab API v3
Git 2.7.4
Ruby 2.3.1p112
Rails 4.2.7.1
PostgreSQL 9.2.18
Projects
15
New Project
Users
4
New User
Groups
1
New Group

开源实践 开源运用

使用级：

- 可以正确安装，正确使用大部分功能，但还不具备自行调优的能力。
- 需要有全面的外部支持。
- 代表软件:SUSE。

参数级

- 可以有效进行参数调优，理解绝大部分核心功能，但对代码的阅读和理解还不够.
- 外部支持以咨询和疑难问题解决为主。
- 代表软件:APACHE、AIX、DB2。

代码级

- 对主要代码进行阅读并理解，可以解决绝大部分问题。
- 外部支持以咨询为主。
- 需要有经验的技术专家。
- 代表软件:MemCached。

开发级

UP化

- 有能力进行代码重构，定制化开发，问题解决。
- 外部支持以咨询为主。
- 需要专门的开发团队。
- 代表软件:UPJAS、upsql、updj

devops
开源落地机制
技术布道

云计算平台
云运营平台
开源软件
传统类IOE

devops
高可用/问题
性能容量/监控上
下线/开发运营

DevOpsDays 即将首次登陆中国

DevOps 之父 Patrick Debois 与您相约

DevOpsDays 北京站 2017年3月18日

门票早鸟价仅限前100名，请从速哟

http://2017-beijing.devopsdayschina.org/

想第一时间看到
高效运维社区公众号
的好文章吗？

请打开高效运维社区公众号，点击右上角小人，如右侧所示设置就好

Thanks

高效运维社区

开放运维联盟

荣誉出品