

The background is a solid blue color. In the four corners, there are decorative graphics consisting of interconnected dots and lines, resembling a network or molecular structure. A white line forms a triangle at the top center, and another white line forms a triangle at the bottom center. Two white diagonal lines cross the background, one from the top-left to the bottom-right, and another from the top-right to the bottom-left.

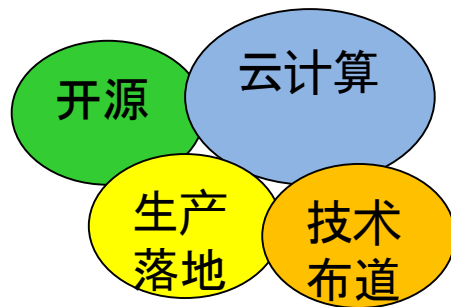
Gdevops

全球敏捷运维峰会

改变与被改变
传统企业的开源之路

演讲人：任明

who am i



技术委员会专家 信息总中心架构室负责人
高可用、问题、容量性能经理 布道师、企业讲师
自主开源落地机制推进 复旦大学MSE客座讲师
devops 理论与实践者 获devops master 认证
从事过网络协议开发、省级银行数据大集中建设 银联二代系
统建设
银联云计算、云运维平台总负责人



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

引子-四年前



中国银联
China UnionPay



山雨欲来



满楼

2012年11月



引子-四年前

引子 从IT发展说起 硬件
















| X86-CISC | POWER-RISC |
|---|---|
|  SYSTEM1.0 | |
|  1984 IBM PC Jr. IBM released its PC Jr. and PC-AT. The PC Jr. failed but the PC-AT proved to be several times faster than original PC. |  1990 IBM POWER1 型号 7011-220 火星探险器 |
|  1984 Apple Macintosh Apple launches the first successful mouse-driven computer with a graphic user interface. |  1993 IBM POWER2 型号 7011-250 32位处理器 深蓝1997 卡斯帕罗夫 |
|  1994 Pentium Processor 100 MHz Intel releases the 100 MHz version of the Pentium Processor. |  1999 IBM POWER3 型号 7044-270 64位处理器 |
|  1995 Sony Playstation Sony releases its first Playstation - To date, over 100 million have been sold. |  2001 IBM POWER4 型号 7040-681 64位处理器 第一个双核处理器 |
|  2001 Mac OS X / Windows XP / Linux 2.4.0 New versions of the three major operating systems are released. |  2004 IBM POWER5 型号 9113-55A 64位处理器 SMT 出售PC业务 IBM转型服务商 |
|  2008 HTC Dream / Google Android The HTC Dream is released - the first commercially available phone to run the newly released Android operating system by Google. |  2007 IBM POWER6 型号 9119-FHA 64位处理器 虚拟化+ 高主频 5GHz |
|  2010 iPad Apple releases the iPad, a tablet computer that bridges the gap between smartphones and laptops. |  2010 IBM POWER7 型号 9117-MHB 64位处理器 SMT++ 八核 |

引子-四年前

IT发展 OS



| WINDOWS/DOS | | LINUX | | AIX-UNIX | |
|--|---|--|--|---|--|
|  | 1981 DOS 1.0 MS-DOS PC-DOS 16位 单用户 单任务 命令行 |  | 1991 MINIX→linux诞生 Linus Torvalds 开源/免费的代名词 | 1986 AIX v1 第一个支持商用RISC的操作系统 | |
|  | 1990 windows 3.0 视窗操作系统 32位 提供可编程API |  | 1994 linux 内核1.0 1996 内核2.0 2001 内核2.4 2003 内核2.6 2011 内核3.0 | 1988 AIX v2 使用在RT/PC UNIX系统上 | |
|  | 1995 windows95 32位 史上最成功的操作系统之一 个人 桌面时代来临 |  | 商业发行版 redhat redhat suse novell OEL oracle redflag 中科红旗 中标麒麟 中标软件 | 1990 AIX v3 POWER ODM LVM JFS SMIT | |
|  | 1998 windows98 32位 Windows95的加速版 中国家庭电脑启 蒙版本 | | | 1997 AIX v4.3 64位 WLM 96G | |
|  | 2001 windowsXP 32位 持续了10年的版本 至今仍有不少市 场 | | | 2001 IBM POWER5L POWER 4 LPAR JFS2 itanium SMT | |
|  | 2009 windows7 32/64位 特效 小工具 搜索 易用 | | | 2007 AIX 6 开发代号AIX 5.4 POWER6 JFS2 快照 | |
|  | 2011 windows8 32/64位 视IOS/andriod为竞争对手 |  | 社区发行版 centos Debian Ubuntu Gentoo opensuse | 2010 IBM POWER7 WPAR 并发内核更新 RBAC SMT++ | |
|  | 遗忘的角落 WINDOWS ME/VISTA | | | | |
| | |  | 嵌入式linux uclinux uc/OS ARM android 航空业 机顶盒 网络 系统 | | |

引子-四年前

IT发展 网络



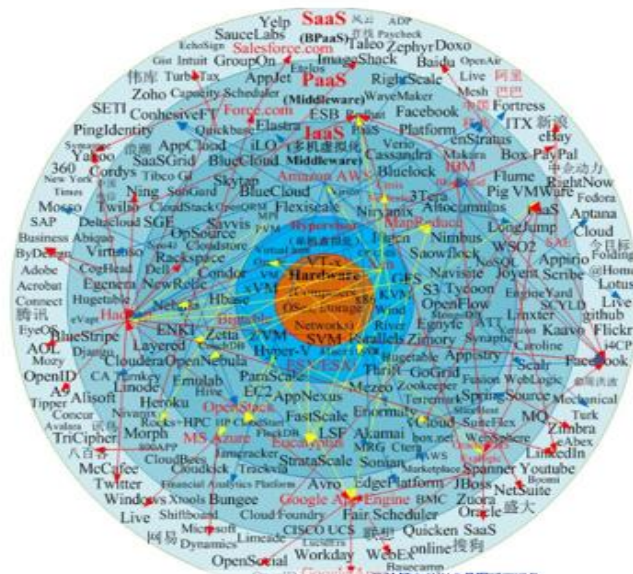
| 网络 | 网络设备 | 网络应用 |
|--|---|---|
|  1969 ARPAnet (美国国防部) 建立了四所大学4台大型机的4节点网络 50kbps |  1986 cisco第一个商业路由器ags发布 |  1971 第一个电子邮件 洛杉矶大学和斯坦福大学之间 |
|  1973 ARPAnet首次国际联网, 用户达到2000人 |  1994 cisco推出catalyst交换机 |  1978 史上第一个BBS 水木清华/日月光华/小百合 |
|  1974 TCP/IP协议发布 1983 TCP/IP取代NCP成通用网络协议 |  1995 cisco推出7500系列路由器, 从此互联网开始快速发展 |  1979 史上第一个MUD 泥巴 文字网游 |
|  1984 IEEE (美国电气和电子工程师协会) 发布OSI模型 |  1996 cisco推出GSR12000系列路由器, 主干网和运营商纷纷采用 |  1990 archie 第一个搜索引擎 |
|  1987 CANET中国第一份电子邮件: Across the Great Wall we can reach every corner in the world |  2003 cisco推出7500系列路由器 |  1994 第一家网上银行 first virtual 诞生 |
|  1993 “三金”工程启动, “金桥”工程推动中国互联网发展 |  2004 cisco推出7600系列路由器, 光纤大行其道 |  1999 OICQ 即时多对多聊天工具 |
|  1996 中国四大骨干网络: CSTNET、CHINANET、CERNET、CHINAGBN | |  现在 网游 电子商务 在线视频 facebook/开心网 twitter/微博 维基 |
|  2000 中国九大骨干网络: CSTNET、CHINANET、CERNET、CHINAGBN、UNINET、CNCNET、CMNET、CGWNET、CIETNET | | |

引子-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

...

引子-四年前

一折 风起云涌 1.0 or 2.0



云计算1.0 ? 云计算2.0 ?

云计算1.0是以技术为导向，以IT基础架构建设为主的阶段，重心在IT部门及信息部门，以IAAS建设为主

云计算2.0是以业务需求为导向，主导者变化，重心在业务部门，以PAAS、SAAS应用为主

1.0

各企业目前构建的基础IAAS云平台，一般为私有云

2.0

互联网企业目前提供的PAAS、SAAS为主的应用

引子-四年前

一折 风起云涌 看起来很美



中国银联
China UnionPay

让我们来学一下《浪潮之巅》的语法：

在此我们可以大胆的预测，根据一个新兴技术从热捧到抨击再到慢慢升温的三段发展规律来看，三年后的云计算必然是今天PAAS/SAAS为主导的以google amazon facebook等公有云的市场天下，而以IBM HP为主等供应商企业如果不能有效的摆脱其自身的“软硬件为王”“高成本”的IAAS为主的私有云部署将仅仅会“看起来很美”

引子-四年前

三折 软硬件及虚拟化



中国银联
China UnionPay

舶来的名词

SUSE XEN
VMVARE LINUX
HYPER-V
KVM
OPENSTACK
MYSQL
MONGODB
NGINX HADOOP
Blade



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

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

技术发展历程

业务特点变化

从后台到前台

从2B到2C

2B

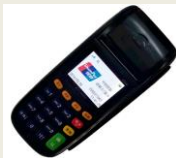
第三方支付

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

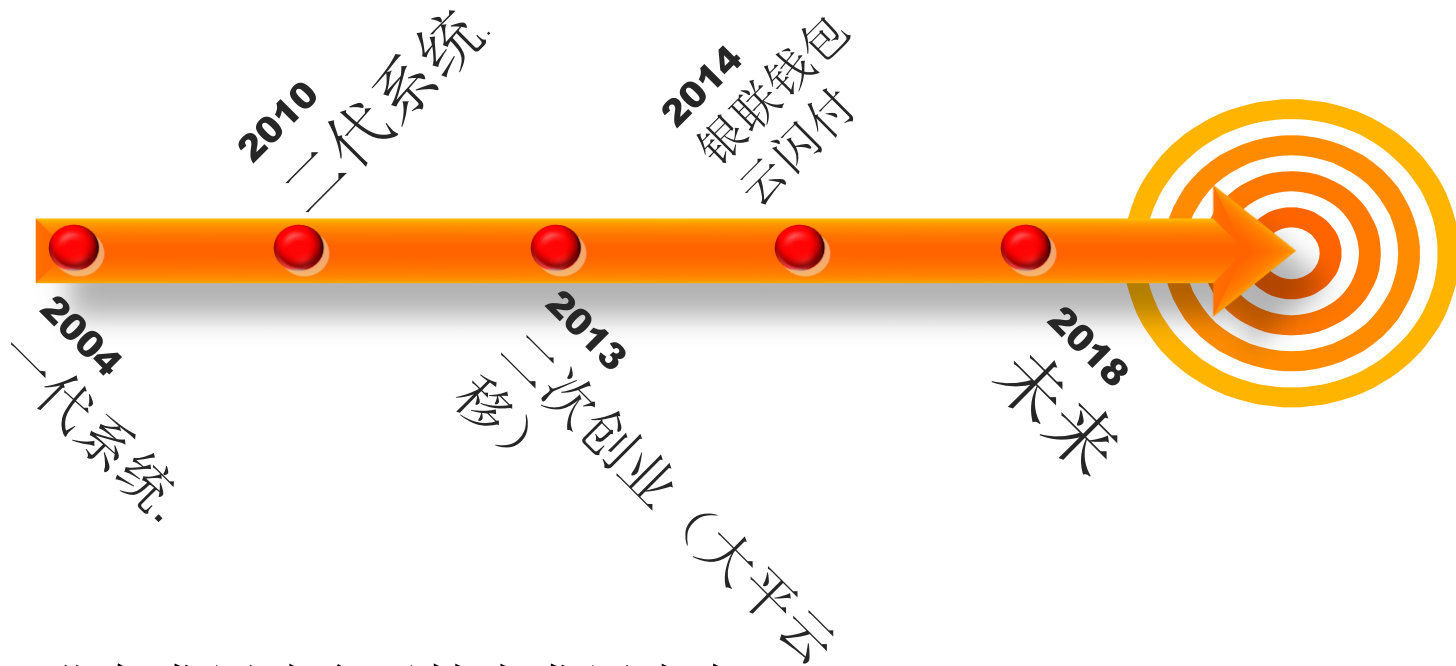
持卡人

行业方

典型业务：银联钱包、全渠道、APPLY PAY



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)



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

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

传统与开源之争

模式比较

| | 传统模式 | 自主模式 (<u>devops</u>) |
|-------|--|--|
| 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 |



传统与开源之争

“双态”并存

传统

开源

稳态

敏态

ITIL

devops



四年前

技术发展历程

传统与开源之争

开源成熟度评估

机制与文化

开源成熟度评估

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。

开源成熟度评估

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

开源成熟度评估 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

➤ 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：

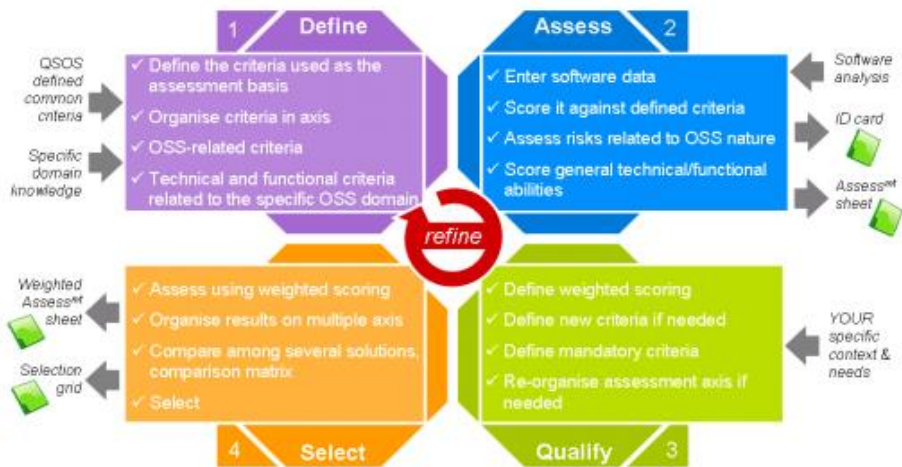
- Software (软件)
- Support (支持)
- Documentation (文档)
- Training (培训)
- Integration (整合)
- Professional Services (专业服务)

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。

开源成熟度评估 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：



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。

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。

开源成熟度评估 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, 不排除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。

开源成熟度评估 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

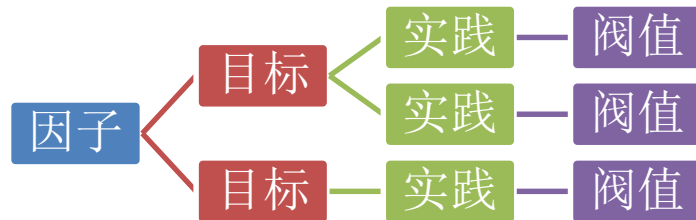
- 十一类评估条件：
- 可用性
 - 软件开发质量
 - 安全性
 - 性能
 - 可扩展性
 - 架构设计
 - 服务与支持
 - 文档
 - 接受度（知名度）
 - 社区建设
 - 专业化程度

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

开源成熟度评估 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或多个具体实践，每项实践设置阈值，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。



开源成熟度评估 业界模型总结

OSMM-C

对于软件质量及成熟度覆盖较少。在开源社区很少应用

OSMM-N

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。

QSOS

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。

OpenBRR

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。

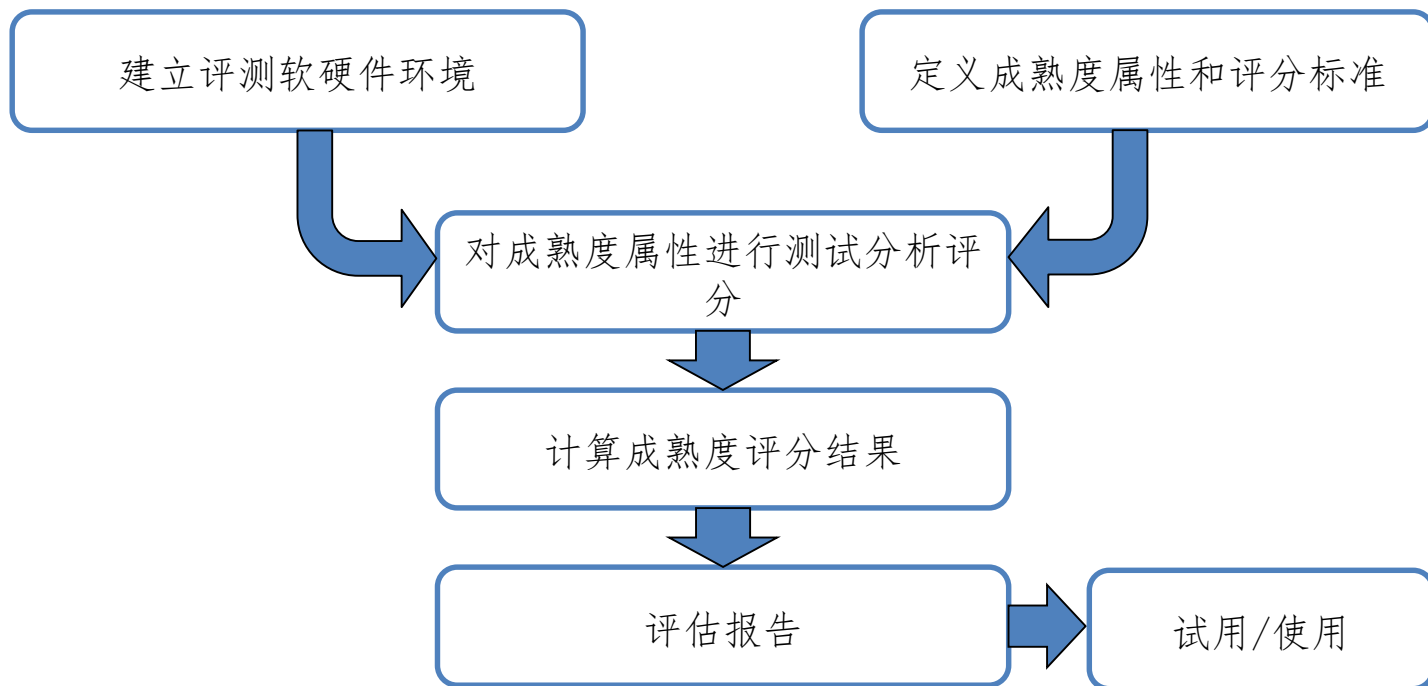
OMM

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。

开源成熟度评估 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。



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。

开源成熟度评估 完整评估方法—属性

我们将成熟度评估属性分为通用属性和特殊属性，并按类聚合。

通用属性的选择主要把握几个软则：典型性、可测性、简明性、完备性、客观性。

| | 属性类 | 属性 | 备注 |
|----------------|--------|-------|---|
| 通用属性 (内在部分) | 技术架构设计 | 架构合理性 | 分析软件的架构、模块划分等软件设计问题，评估其技术特性。 |
| | | 模块划分 | |
| | | 对外接口 | |
| | 软件功能 | 需求功能 | 功能满足实际使用需求 |
| | | 扩展功能 | 提供需求外的附加功能 |
| | 软件“五高” | 高性能 | 通过五高测试对开源软件进行评测，使用的重要指标。例如性能、扩展性、高可用、灾备方案、监控、api接口、命令行及管理控制台等维度。 故障性测试包含长时间压力、断电、拔线、多次重启、反复切换，并且在测试时观察cpu、内存、磁盘、网络、连接数等关键指标。 |
| | | 高可用性 | |
| | | 高可管理性 | |
| | | 高可扩展性 | |
| | | 高安全性 | |
| | 代码质量 | 代码错误率 | 通过辅助工具对开源代码的直接分析，评估代码的质量和可信度，生产一个代码评估报告。 |
| | | 代码可读性 | |
| | | 代码复杂度 | |

开源成熟度评估 完整成熟度评估方法一属性

| | 属性类 | 属性 | 备注 |
|----------------|-------|-----------|---|
| 通用属性 (外在部分) | 社区活跃度 | 组织架构、社区管理 | 包含开源项目的组织形式、社区管理、社区书籍、文档、论坛的支持度、社区软件测试情况等特征，从侧面了解开源项目的发展。 |
| | | 社区支持度 | |
| | | 社区测试团队情况 | |
| | | 版本发布周期 | 通过对项目的活跃度进行统计分析，判断项目的受欢迎程度、开发速度、应用的支持力度等成熟度因素。 |
| | | 软件下载量 | |
| | 应用与发展 | 商业公司支持度 | 从国内外商业公司的支持程度、国内外企业度实际应用案例、与其它知名开源软件的协同发展情况、roadmap清晰度评价其应用成熟度。 |
| | | 企业应用案例 | |
| | | 开源协同发展 | |
| | | 未来路线规划 | |
| | | 版本正式程度 | 一般建议使用1.0以上版本 |
| | 法律问题 | 软件版权协议 | 分析开源软件采用的版权协议，分析软件可能前置的版权、专利等法律风险。 |
| | | 商标、专利与纠纷 | |
| | 落地可行性 | 兼容性 | 与现有技术架构的兼容 |
| | | 替换性 | 与目前使用软件的对比，替换方案的实施可行性。 |
| | | 人员支持 | 现有人力对该开源技术的掌握难度 |

开源成熟度评估 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 |

- 属性类权重设计

| W2（权重） | 1 | 2 | 4 |
|--------|-----|------|----|
| 评价 | 不重要 | 较为重要 | 重要 |

开源成熟度评估 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

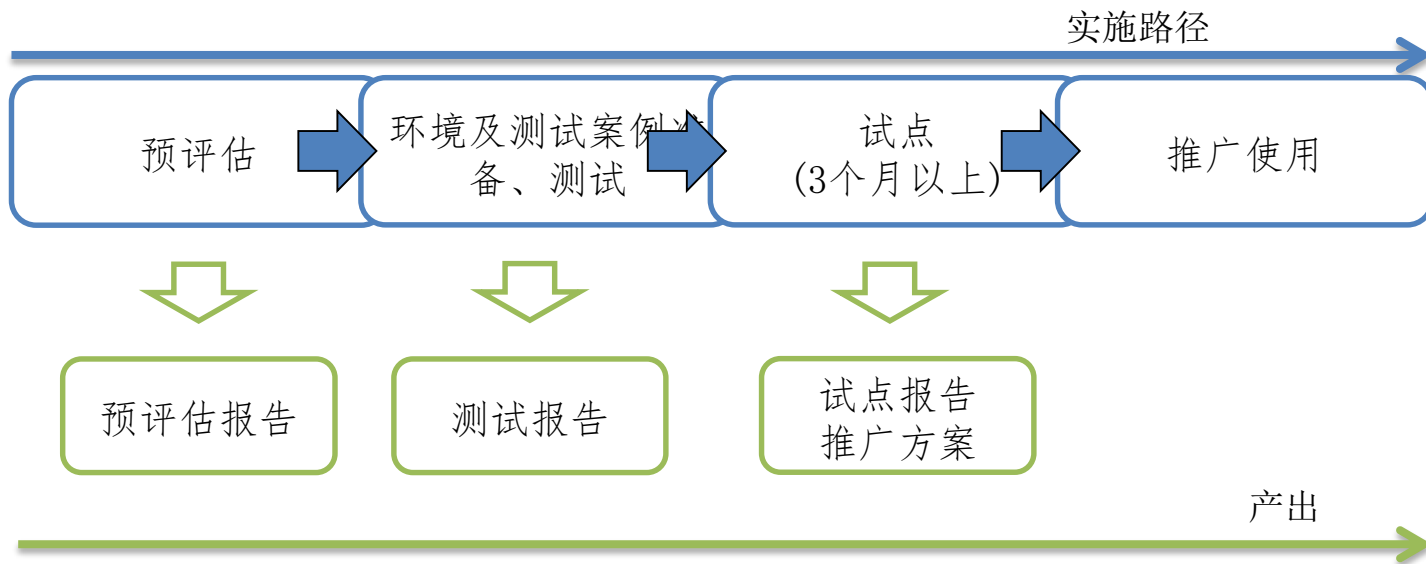
- 结合各项评估给出选项建议及落地方案

相关资料附录

- 测试报告、调研报告、许可证协议等辅助信息

开源成熟度评估 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。



开源成熟度评估 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 |
| 应用与发展 | 商业公司支持度 |
| | 企业应用案例 |
| | 开源协同发展 |
| | 未来路线规划 |
| | 版本正式程度 |
| 落地可行性 | 兼容性 |
| | 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 |
| 软件“五高” | 高可用性 |
| | 高可管理性 |
| | 高可靠性 |
| 代码质量(如准备自己二次开发) | 高安全性 |
| | 代码错误率 |
| | 代码可读性 |
| 落地可行性 | 代码复杂度 |
| | 兼容性 |
| | 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。

- 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。

例如OS、DB、Storage、hypervision、cloud等通用基础性软件

- 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。

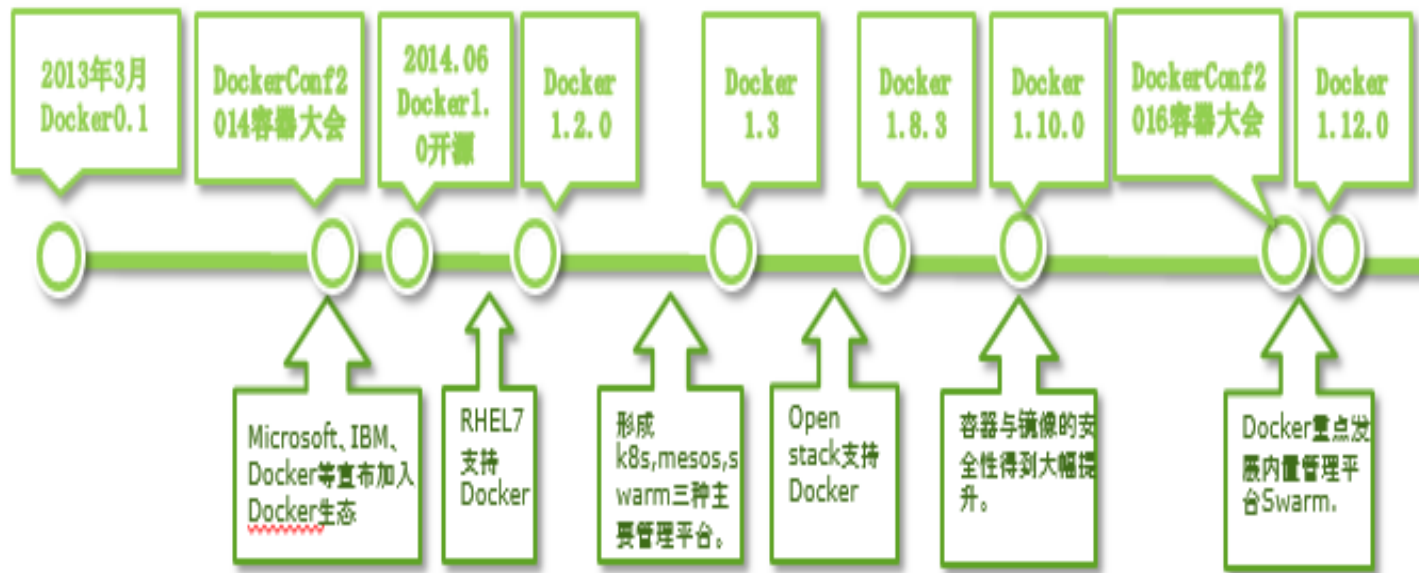
例如bind、haproxy、apache等

总结



- ❏ LXC可以满足X86物理机跑更多容器，部署更多应用或数据库，目前来看适合部署apache、mysql等有性能需求的应用，实现MySQL/Apache的多实例运行，保证多实例之间的资源能够得到有效的分配和隔离。
- ❏ LXC技术目前在金融行业无成熟案例，但有快速发展的趋势。
- ❏ 若单独使用cgroup仅可以进行计算资源的隔离，无法实现用户、文件系统级别的隔离，且cgroup无成熟命令行封装，需要进行api级别开发。
- ❏ 如使用LXC技术需要满足以下条件：
 - X86物理机资源紧张，需要单机多实例。
 - 充分研究测试，规避bug。
 - 与现有运维平台对接，遵循开源协议，需要定制化开发统一管理平台。
 - 采用SUSE自带版本，确保代码级别维保。或者组建研究团队，吃透代码后自行开发使用。
- ❏ 建议对该LXC及cgroup技术保持跟踪，暂不在信息中心维护的生产环境使用。待X86物理机资源紧张，需要多实例资源隔离时，再投入人力进行深入研究、测试、开发。
- ❏ 据了解，研究院正在对cgroup技术进行研究开发，用于分隔云平台中的mysql实例，在该功能正式上线且维护方提供充足的技术支持情况下，可以视为平台功能使用。

Docker发展历程



2016年5月，统一数据分发管理平台使用容器化发布方式。涉及的组件包括 upias, apache, haproxy。部署上线顺利、演练结果符合要求。

2015年-2016年，银联DBAAS平台设计阶段，经调研后采用 docker 作为底层轻量级虚拟化工具。

2014年9月份，针对 Docker 的研究，使用 ubuntu 部署。主要针对其应用场景和功能进行分析。

2016年底，构建容器调度平台

2015年-2016年，云运营平台的研究、测试全部使用容器化工具。生产上线通过容器化方式发布。

2015年7月份，较为全面的对比了 xen 虚拟化和 docker 虚拟化的优缺点，针对其性能、稳定性等进行对比测试。同时，利用主流开源软件设计了一个实验性的容器管理平台，并且和谷歌的管理平台 k8s 进行对比测试。

2013年10月份，开始针对Linux内核LXC容器方案的研究



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

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；

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：MemCache d。



开发级

- 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。

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

2. 如何开始你的开源代码阅读之旅？



3. Ansible的结构解读与生产应用。



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？

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？

激励与学习

1. 分享人员

支付学院讲师积分与奖励

信总季度奖励基金选取最佳开源推进奖

2. 学习人员

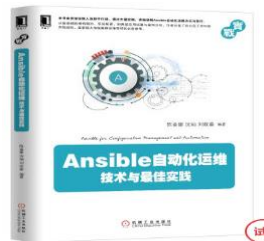
支付学院学习积分

3. 每年将材料汇装成册

4. 对应用广、规划通过、测试优异、社区贡献的的
奖励基金开源落地奖

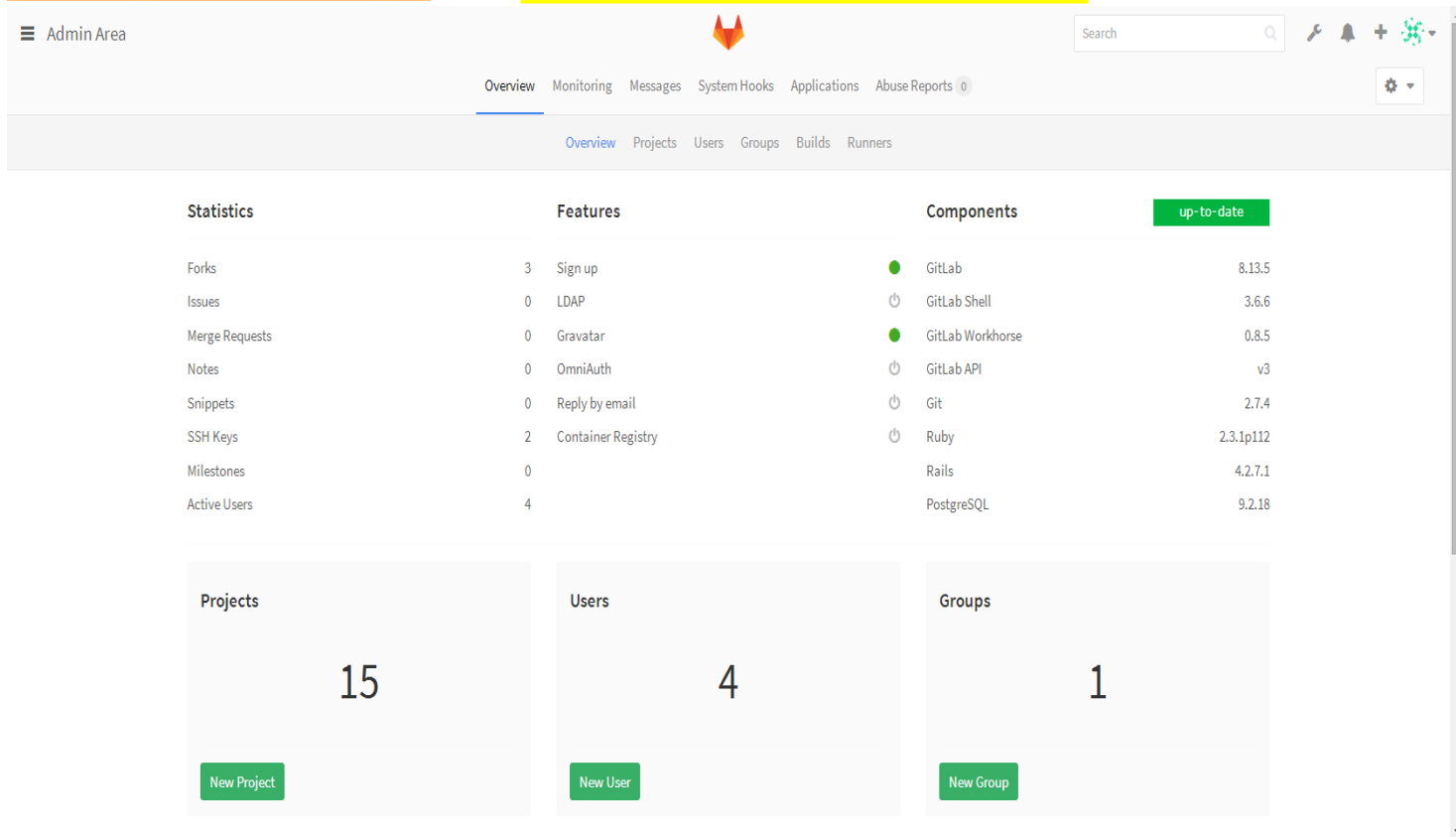
5. 书籍增补

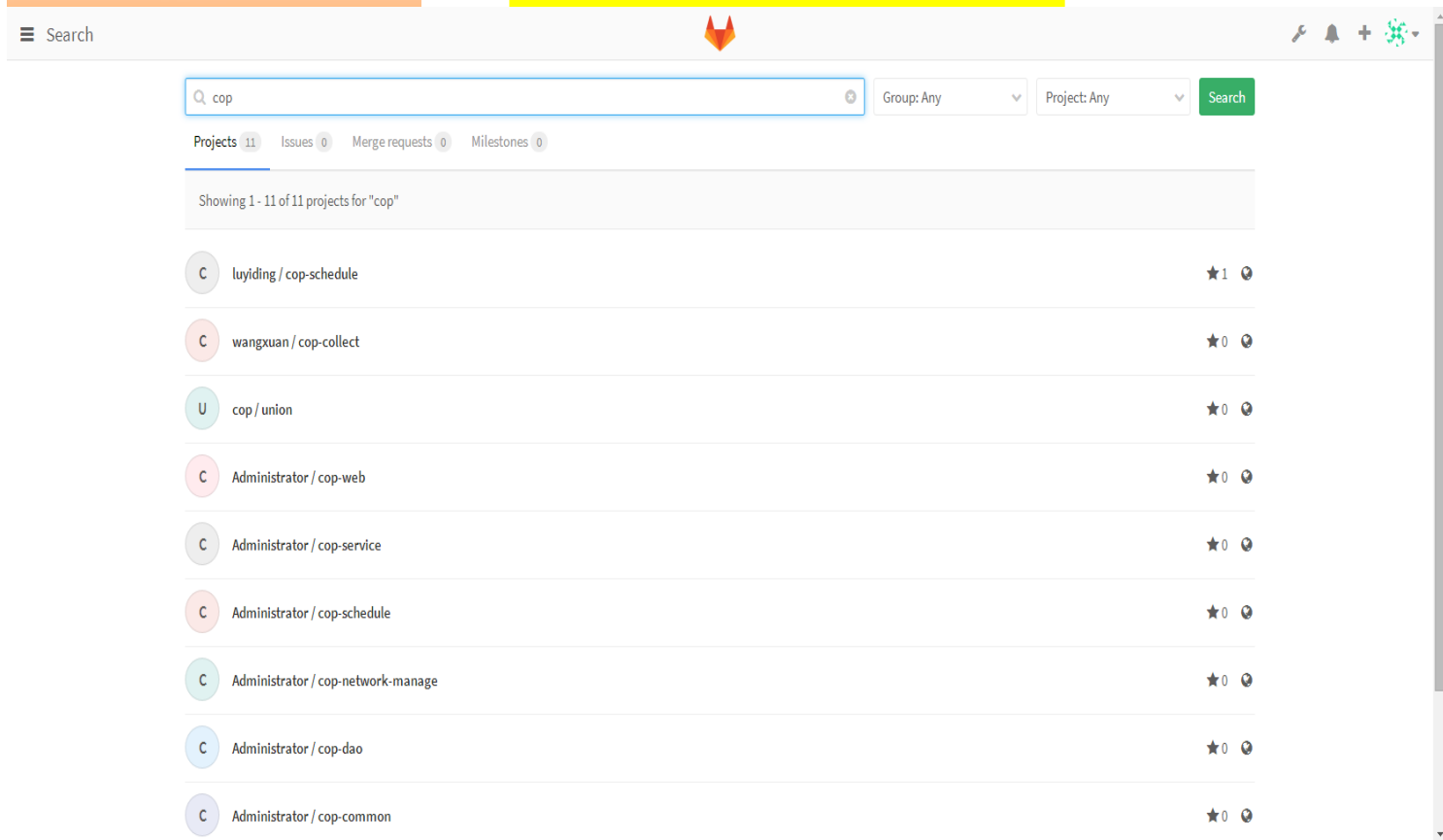
6. 构建开源成长配合资源环境











The background is a solid blue color. In the four corners, there are decorative geometric patterns consisting of interconnected lines and dots, resembling a network or molecular structure. These patterns are in shades of dark blue and black. Additionally, there are four white line segments, each forming a large 'V' shape pointing upwards, positioned around the central text.

Gdevops

全球敏捷运维峰会

THANK YOU!