OPEN SOURCE REVELATION

开源启示录

第三季 2016年5月

InfoQ

Geekbang 极客邦科技 | InfoQ

案 / 例 / 剖 / 析 · 实 / 践 / 驱 / 动

CNUT
Connect Container Community

CNUTCon 2016
全球容器技术大会

2016年9月9日－10日　　中国 · 北京　喜来登长城饭店

6折优惠进行中，5人团购更划算

主办方 Geekbang. 极客邦科技　InfoQ

卷首语

陈绪

人生易老天难老，岁岁重阳。今又重阳，战地黄花分外香。

——毛泽东《采桑子·重阳》

论论开源，想想中国。

开源包括 Linux，而不止于斯。北京凝思科技宫敏博士 1991 年将 Linux 磁盘背回中国的时点，他应该没有想到今年的开源会成为空气，弥漫在中国的大街小巷，成为人们生活的一部分和必需品。

2004 年陆首群老爷子在多个国家部委的支持下创立“中国开源软件推进联盟”（COPU）的时刻，他何曾想到今日之开源，已经成为“创客潮的基础”。

开源，作为一个舶来品，已经被中国所消化吸收，成为自主创新的起点和基石。人类共有的知识财富，正在中国得到了发扬光大。

顺便强调一下“不要自造轮子”的话题，世上的轮子已经够多了，何苦再造一类新的呢？与创立自己的开源项目相比，加入国际的顶尖项目，并成为领导者，似乎更有利于开源项目的发展。

中国的 Linux 操作系统公司，已经由红旗、中标、拓林思的三架马车发展成今日的百家争鸣，其中中标、深度、一铭、红旗、思普等为其中翘楚，承担着国家信息安全的重任。

书生国际的王东临先生，他把公司开发的 SurFS 分布式存储技术开源发布，创造了国内“先自主、再开源”的典范。

中国的开源社区，由昔日的 ChinaUnix，CSDN，到今日的遍地开花。昔日的社区爱好者、

代码贡献者，大都成长为各大 IT、互联网公司的领军人物。

我个人每年最关注两个开源盛会，其一为“开源中国开源世界峰会”（OSCOW），一般在 6 月底，其二为“中国 Linux 内核开发者大会”（CLK），一般在 10 月中旬。更有意思的是他们都起始于 2006 年，今年均为第 11 届，有着顽强的生命力。前者为开源领袖论剑，纵论天下大事，开源风云；后者为内核技术专家布道，技艺切磋交流，提携后进。

云计算成了时髦，除了 OpenStack，大家开始谈 Docker 容器、Kubernetes、Tectonic、Ceph 等名词。今天多讲讲中国的 OpenStack。2012 年 8 月 9 日，剑指 OpenStack 代码贡献的中国开源云联盟（COSCL）成立；2016 年 4 月 15 日，正式移交中国电子技术标准化研究院并继续茁壮成长，成员 62 名。上午还有朋友给我电话，问 OpenStack 的发展怎么样了，我的答案是“如日中天”。

中国的 OpenStack 阵营由四大层面组成。

- 第一个层面是 OpenStack 厂商，一般称为一大八小：一大是华为，八小是 AWCloud、99Cloud、EasyStack、UnitedStack、T2Cloud、UMCloud、KeytoneCloud、KylinCloud。
- 第二个层面是 OEM 厂商，华为，联想，浪潮，中兴，宝德等，均下注 OpenStack。
- 第三个层面是大规模部署的互联网公司，百度，金山云，乐视云，京东云，爱奇艺等。
- 第四个层面是企业用户，移动，联通，电信，银联，人民日报等。

是的，你没有看错，“华为”出现了两次，他们的开源贡献也可圈可点。

中国的开源，正如中国的云计算，还方兴未艾，处于绝对的上升曲线中。你，可以选择拥抱它，成为时代的弄潮儿，但你无法忽视它，它无处不在。君不见前日之 OpenStack 奥斯汀峰会与会人数 7500，和 2010 年的第一届峰会 75 人相比，竟百倍之差。

最后，不由想起工信部信软司领导对开源的三个期许，和诸友共勉之。

其一是要“实”不要“虚”，宣传已经太多，我们更需要实际的代码，正如 Linus 说 Talk is cheap, show me the code；

其二是要“新”不要“旧”，中国更需要具备前瞻性的新技术，谢绝落后的架构和软件；

其三是要“中”不要“洋”，开源产业链的中国公司要有自己的研发能力，要强调自主创新，正如倪光南院士奔走疾呼之“自主可控”。

思绪所至，是为此文。

目录

开源启示录

第 3 季

本期主编　李建盛
流程编辑　丁晓昀
发行人　霍泰稳

提供反馈　feedback@cn.infoq.com
商务合作　sales@cn.infoq.com
内容合作　editors@cn.infoq.com

衡量开源社区的 5 大指标

作者 **Jesus M. Gonzalez-Barahona** 译者 **李建盛**

一若你决定使用一些指标来跟踪一下你的自由 / 开源软件（FOSS）项目的话，那么最大的问题是：哪些指标是我需要跟踪的？

要回答这个问题，你必须对自己需要什么样的信息有所了解。举个例子来说，你想知道关于项目社区的可持续性。社区是如何快速的响应问题的？社区是如何吸引、留住或流失贡献者的？一旦你明白自己需要什么样的信息，你就可以找出社区的哪些活动能够提供这样的结论。好消息是，FOSS 的项目遵循开放发展的模式，这就是说在它们的软件仓库中留下了公共的数据，这样就可以用来分析有用的数据。

在本文中，作者将介绍能够提供多面视野的衡量开源社区的指标。

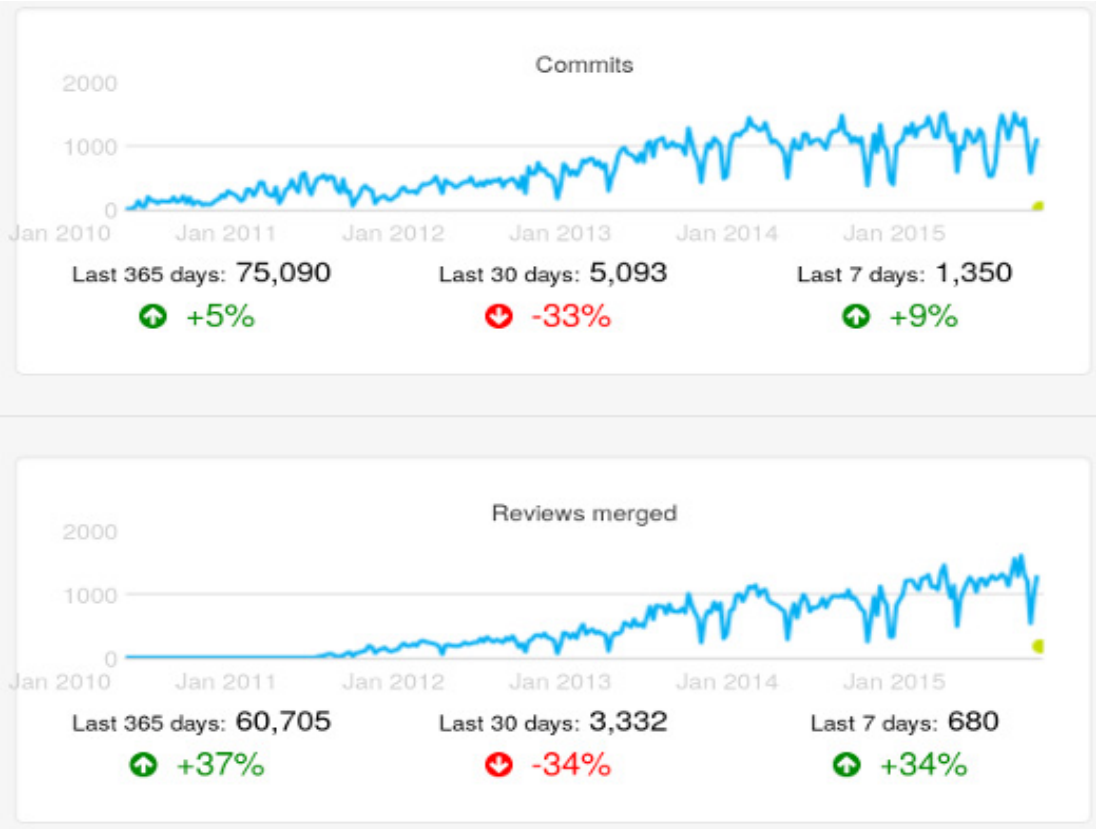


图 1

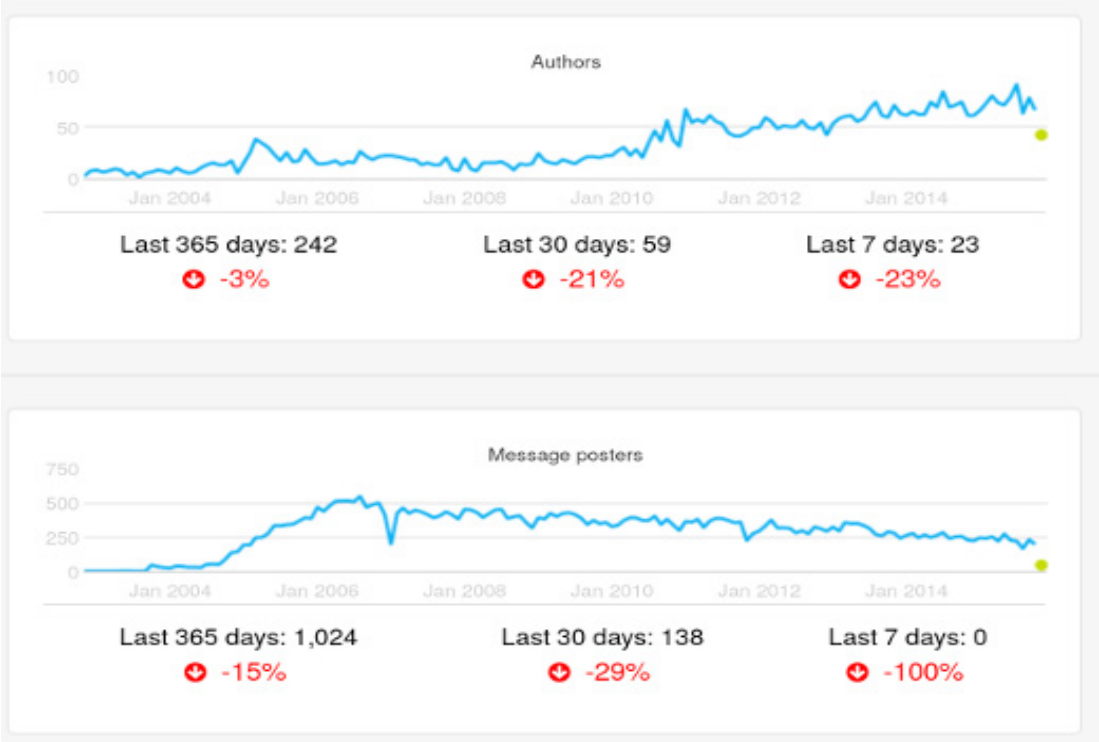


图 2

活跃度

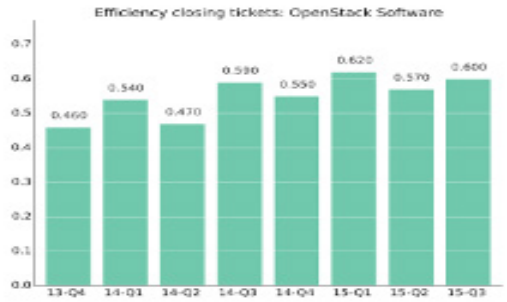
对于所有的开源社区来说，整体的社区活跃度以及如何随时间而演变都是非常有用的衡量指标。活跃度为我们提供了社区做了多少的第一印象，当然具体来说活跃度也可细分下去。举例来说，提交的次数可以表明开发者们的努力指数；处于打开状态的传票（ticket）数则可洞晓项目目前有多少 Bug 没有修复或者是新提出了多少特性需求；邮件列表中消息的数量或论坛的帖子数量让我们对所公开讨论的内容心里有个数。

图 1 所指是项目 OpenStack 的提交数以及代码审核之后的合并数，时间演进（周度数据），数据来源 OpenStack 活跃度面板。

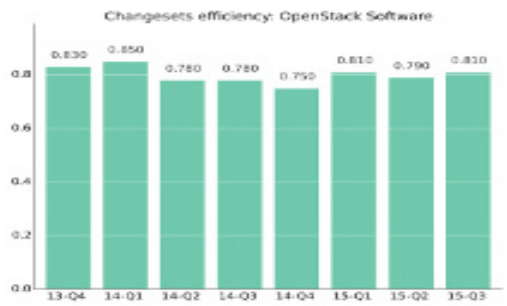
规模

社区的规模常常指的是参与到社区的人员的数量，但是，也要取决于参与的类型，规模的大小也可能有所不同。通常我们最为关注的是活跃贡献者，这当然是正确的关注点了。活跃的用户会在项目的代码仓库中留下轨迹的，这也就意味着我们可以统计到哪些撰写代码较多的贡献者，比如 git 仓库的话查看作者一项即可；又如可通过查看谁解决的问题最多来统计参与的活跃度。

活动最为基本的概念（某人做某事）可以有多种延伸。常见的跟踪活动的方法是查看有多少人所做的占了多大的份额。通常为项目的代码贡献，比如，在项目社区中占有很少的份额。了解此占有份额有助于知道核心小组的意见（比如某些人是社区的领导者）。



Period	Closed/Opened
13-Q4	0.46
14-Q1	0.54
14-Q2	0.47
14-Q3	0.59
14-Q4	0.55
15-Q1	0.62
15-Q2	0.57
15-Q3	0.6



Period	(Aband. and Merg.)/Subm.
13-Q4	0.83
14-Q1	0.85
14-Q2	0.78
14-Q3	0.78
14-Q4	0.75
15-Q1	0.81
15-Q2	0.79
15-Q3	0.81

图 3

图 2 是 Xen 项目的邮件列表的作者数量和发信人的数量图示，时间演进（月度数据），数据来源 Xen 项目开发面板。

表现

以上，我们聚焦于衡量活跃度和贡献者的数量。但是我们仍然可以分析社区成员是如何处理问题的，或者是他们的具体表现等。举例来说，我们可以评估下他们处理某个流程需要多长时间，解决或关闭一个传票的时间可以说明此项目对于新的请求活动是如何响应的，诸如修复一个报出的 bug 或者是实现了一个新提出的特性。代码审核时间的长短－从代码变动的时刻到代码被接受为止这段时间－则说明了多长时间的建议变更到社区所制定的质量标准。

其它衡量的如：如何处理项目的应对工作，诸如新的和已经关闭的传票比率，没有完成的代码审核的积压。这些因素告诉我们，所投入的资源是否足够能处理完问题。（图 3）

关闭的传票和打开的传票的比例，建议接受变更和拒绝变更的比例。数据来之于项目 OpenStack，OpenStack 开发报告，2015 年第三季度。

人员流动

贡献者的来来去去就可说明社区的变动。取决于随着时间的推移有多少人参与进来以及多少人离开，即所谓的社区的不同年龄（时间从成员参与到社区就算）。社区年龄图表很形象的讲清楚了随时间的成员变化。图表由一组横向的进度条所组成，一对进度

条表示的是“一代”加入到社区人们，对于每一代来说，attracted 进度条表示的是在此时间段内有多少新加入到社区中的人数，retained 进度条表示的是有多少仍然活跃于社区的人数。

每一代的两条进度条的关系就是留存率：就是仍然在项目中的成员的分数。整体的 attracted 进度条展示了项目在过去有多大的魅力，而整体的 retention 进度条展示的是社区当前的年龄结构。（图 4）

来自 Eclipse 社区的社区年龄图 3，数据来源 eclipse 开发者面板，此表每隔半年生成。

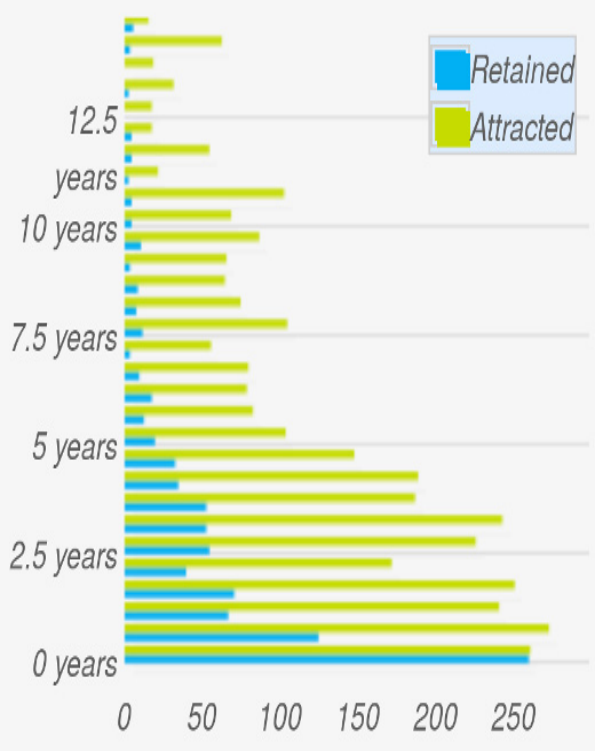


图 4

多样性

多样性对于一个健壮的社区来说是非常重要的一个因素。一般来讲，一个社区具有更多的多样性——参与的人或公司的一个术语——那么说明这个社区就更加的健壮。举例来说，当一家公司要从一个 FOSS 项目中撤离时，那么就会考虑到它的离去所带来的潜在问题就是其员工所贡献的将会缩水，从 85% 变为 5%。

小马因子，这是由 Gruno 所定义的一个术语，意思是贡献了 50% 的开发者的最少数，以此类推，大象因子表示的就是贡献了 50% 的公司的最少数。此两个数字均提供了社区所依赖的开发者和公司的数量。（图 5）

云计算领域的一些 FOSS 项目的小马和大象因子。数据来源：2015 开放云的定量状态（演示文稿）。

对于量化社区还有很多的有用的指标，当决定收集哪些指标时，要明确社区的目标，然后在去寻找哪些指标可以帮助你实现它。

	Pony Factor	Elephant Factor	Commits (excl bots)
OpenNebula	4	1	12K
Eucalyptus	5	1	25K
CloudStack	14	1	42K
OpenStack	>100	6	126K
CloudFoundry	41	1	60K
OpenShift	10	1	15K
Docker	15	1	18K
Kubernetes	12	1	7K

图 5

作者简介

Jesus M. Gonzalez-Barahona 是 Bitergia 的联合创始人，Bitergia 是一家针对软件开发的分析公司，特别是对自由 / 开源软件项目。他同时还在 Rey Juan Carlos 大学（西班牙）进行 [教学和研究](http://gsyc.es/~jgb)，也隶属于 [GSyC/LibreSoft](http://libresoft.es/) 研究小组。

Facebook 在 2015 年的 5 大开源项目

作者 **邵思华**

Facebook 相信开源的巨大力量。当整个社区能够共同致力于编码工作时，所产生的益处是不可估量的。人们会用新的眼光指出我们的问题，让我们能够更快地找出解决方案。我们将一起克服所面对的挑战、加速创新过程。社区的力量能够让我们冲破现有技术的种种限制。

当然，成功的开源项目离不开一个健壮的、通力协作的社区。在年终即将来临之际，我们将按照社区的活跃度与影响力排名，选出 Facebook 在 2015 年的 5 大开源项目。

HipHop 虚拟机（HHVM）

HHVM 是我们开发的一套虚拟机与 Web 服务器系统，并于 2013 年实现了开源，它是基于我们在 2010 年发布的 HPHPc 编译器开发的。仅在过去一年间，代码提交次数就上升了 29%，而 fork 的数量则上升了 30%。

HHVM 最常见的用途是作为单台服务器使用，它的目标是取代 Apache 与 mod_php，运行由 Hack 和 PHP 编写的程序。通过 JIT 编译方式，HHVM 能够实现更好的性能，同时保留了 PHP 开发者已习惯的各种灵活性。我们在今年实现了几个重要的里程碑：

- 新的 Async 特性默认可用，包括对 AsyncMySQL 与 MCRouter（memcached）的支持。
- 当 PHP 7 语言本身于 12 月推出的同时，我们就宣布了对 PHP 7 所有主要特性的支持，并且发布了新一版的用户文档。
- Box 宣布将使用 HHVM 作为运行其 PHP 代码的唯一引擎。
- Etsy 在 4 月份将业务迁移至 HHVM 平台，帮助该公司克服了在创建大规模移动产品时所遇到的各种挑战。

React

Facebook 在 2013 年 5 月开源了 React，而在过去一年间，我们仍然获得了来自社区的极大支持，代码提交的数量提高了 75%，而 fork 的数量更是提高了 198%。React 是由 Facebook 所设计的一种用于构建用户界面的 JavaScript 库，目前已在许多公司得到应用。React 使用了一种全新的方式构建应用：它允许开发者将应用分解为相互解耦的组件，因此每个组件都可进行独立的维护与迭代。

今年，我们为 React 推出了两个重要的发布：一是 React Native，二是新的开发者工具。我们也看到越来越多的公司开始使用 React 构建他们的产品，包括 Netflix 与 WordPress。

Presto

Presto 是由我们设计的新型分布式 SQL 引擎，它能够对各种大小（从 GB 级至 PB 级）的数据源进行交互式的分析查询。我们设计 Presto 的目标是帮助我们更快地进行数据分析，以配合我们不断增长的数据量与持续加速的产品周期。

自从我们于 2013 年 11 月开源了 Presto 之后，它的发展、接受度以及对它的支持都得到了全面的提高。在去年一年的时间内，它的代码提交数量提高了 48%，而 fork 的数量则提高了 99%。Airbnb、Dropbox 以及 Netflix 等各大公司都开始使用 Presto 作为他们的交互式查询引擎。Presto 在全球范围内的接受度也在逐步提高，包括来自日本的社交媒体游戏开发公司 Gree，以及来自中国的电子商务公司京东 JD.com。

同样在今年，Teradata 也宣布了加入 Presto 社区的计划，专注于企业级特性的改善并提供相应的支持。这也展现了整个社区对于 Presto 成为数据基础设施方面一个重要组成部分的信心。此外，Amazon Web Services（AWS）也在其 EMR 服务中将 Presto 作为一线功能提供支持，已有诸多用户在生产环境中使用该功能，包括 Nasdaq。而在业界处于领先地位的商业智能工具开发商 MicroStrategy 也在其旗舰产品 MicroStrategy 10 中提供了对 Presto 的支持。

RocksDB

我们在 2013 年 11 月开源了 RocksDB，这是一个嵌入式的持久化键值数据库，支持高速的数据存储。在过去一年间，该项目的代码提交数量提高了 52%，而 fork 的数量则提高了 57%。除了这些令人印象深刻的数据之外，该项目引起了整个社区强烈共鸣的原因在于该嵌入式数据库能够为由网络延迟引起的响应缓慢问题提供一种临时方案，并且它提供了充分的灵活性，可通过自定义的方式应对不断发展的硬件趋势。

RocksDB 为 LinkedIn 与 Yahoo 等公司提供了各种关键性的服务。我们今年的主要目

标之一是将 RocksDB 这个存储引擎的特性迁移至通用目的的数据库上，以 MongoDB 作为起点。与 Teradata 宣布提供对 Presto 的商业性支持类似，RocksDB 今年的另一个里程碑是来自 Percona 的数据性能专家宣布将为其提供企业级的支持。

React Native

React Native 是我们最新推出的开源项目之一，它在今年 3 月实现了开源。React Native 让工程师能够使用与 React 相同的方法与工具为移动设备快速地创建原生应用。Facebook 不仅在内部持续开发这些工具，并且致力于通过与开源社区的协作改善全球开发者的体验。在 React Native 诞生的第一个年头，它就在 Facebook 的开源项目流行度排名中攀升到第二的位置，在 GitHub 上已有超过 23,000 位关注者。Facebook Ads 的 iOS 与 Android 应用内部正是使用 React Native 所开发的，对于那些以 JavaScript 为核心能力的开发者来说，代码的重用率达到了 85%。React Native 为移动开发的观念带来了重要的改变，这也是我们本年度最关键的成就之一。

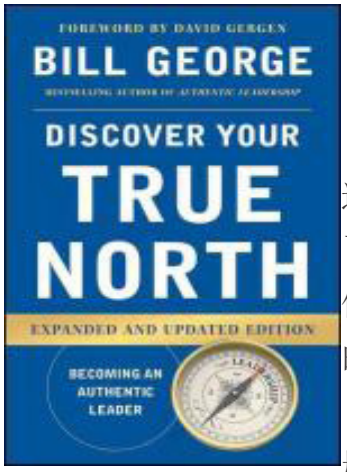
总的来说，我们仍有许多要完善的地方，但作为整个社区的一份子，我们对于目前的成就深感自豪。对于那些在百忙之中仍乐于为这些项目贡献力量、使我们在这一年中达成如此成就的人们，我们深表谢意！

8 本书，让你成为更加开放的领导者

作者 **Brayn** 译者 **李建盛**

开源的理念依然在不断的影响着新的领域，而在过去的 2015 年这一过程明显是加速的。传统组织的架构正在让位于新的模式－更加开放的模式－而且我们合作的方式也在迅速的改变。

在 Jim Whitehurst 在 2015 年的 6 月出版的图书《The Open Organization》中，它总结了多年所积淀的内容：一种新的管理模式，一种能够适应 21 世纪的模式。世界各地的领导者们都在积极的去理解在此范式下自己的角色定位，我们社区希望通过所推荐的书籍来帮助指引他们跨越不确定的形式。如果你已经决定自己在 2016 年成为一个更加开放的领导者的话，那么希望你从这里开始。

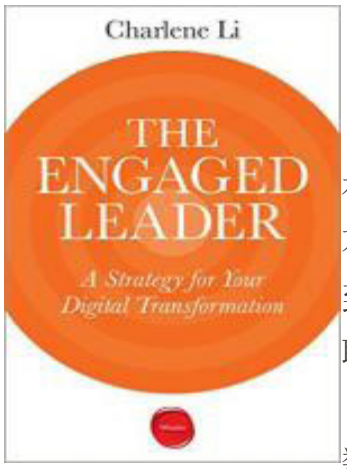


《发现你真正的方向》

作者 Bill George （由 Sam Knuth 推荐）

Bill George 曾写过多部关于领导力真伪的著作，其中包括透明度、诚实度、以及开放程度。《发现你真正的方向》 则描绘了 George 自己与多个跨行业的领导者们接触的亲身经历，将他们放在一起来尝试探讨他们的成功与失败。但书中至始至终突出的概念，和 Jim 在 《开放组织》 所倡导的领导力如出一辙。

本书不仅收集了大量的案例和实践经验，而且是真实的指导手册，对于处于职业生涯中任何阶段的人的开放领导力都适用。



《The Engaged Leader》

作者 Charlene Li （由 Jason Hibbets 推荐）

使用社交媒体来招聘和倾听不是一门科学，而是一门艺术。在本书中，李带你领略来自不同行业的实例和在他们各自组织中不同级别的领导者的情况。我将此书推荐给我在红帽（以及原来到）的同事，那些个打算利用社交媒体和其它的数字工具更多的联络他们的客户、下属、尤其是其他的一些领导者。

书中包含非常实用的故事和一些提示来帮助你参与这个新的数字时代。

《如何发展健康的情绪》

作者 Oliver James（由 Laura Hilliger 推荐）

本书短小精悍，你可以轻松读完。它够帮助你远离那些阻挡你成为开放的领导者的，整天胡扯的人。

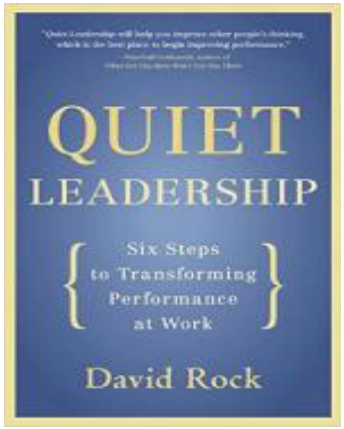
James 为大家提供了解决方案和洞见，你在过去、现在和将来如何工作，如何拥有健康的人格、快乐、而富有成效的生活。这是好领导者的试金石，事实上，就是我们说的“开源人”。

《重来》

作者 Jason Fried & David Heinemeier Hansson（由Marten Mickos推荐）

本书讨论了现代世界的组织工作，所谓的现代世界即是人们在家里工作，而且很大程度上知道自己该怎么做。本书非常的干练、且图文并茂（“孤注一掷”、“会议是有毒的”、“如果没有了灵感”），推荐希望在工作场所做出改变的人阅读。

虽然本书的作者没有使用开源、开放等术语，但是书中所展示的内容非常的明显：开放是其基本原则。



《安静的领导力》

作者 David Rock（由 Rebecca Fernandez 推荐）

不论你是一名管理者、还是一名导师，又或者是两者均是。你都要成为一个教练的角色，在激励他人成长的同时并给予反馈，帮助他人作出自主决策。

通过神经科学的探索，该书向我们展示了为什么领导者应该去停止给出建议以及提供想法，应该去学习多听、少说，以及多问一些问题的方式来帮助人们自发的去解决问题。

《开放组织》

作者 Philip A. Foster（由 Bryan Behrenshausen 推荐）

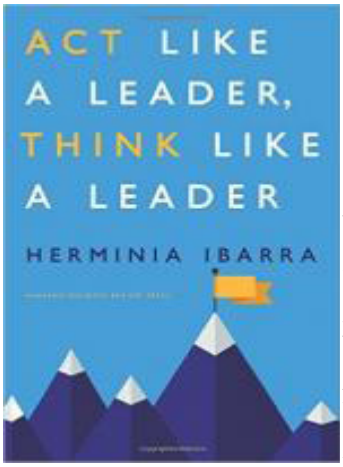
《开放组织》很坦率的说是本教科书，但两位读者的理念都是“新的管理模式”。全书内容严谨、配有完整的注释，描绘了从管理的理论到实践的全部运行的源头。Foster 还提供了一个令人侧目的案例：GitHub，他的使命是“创建惊喜”提示有趣的组织选择。对于爱好者来说，这是一本商业公司如何推进开源任务的深刻洞见之作。领导者都应该读一读。

《Consiglieri》

作者 Richard Hytner（由Laura Hilliger推荐）

虽然本书的重中之重是那些“邪恶公司”的例子，而且还擅用了他人的名义，但是它包含了一些关于成为领导者要指导多于指挥的精彩观点。

懂得尊重和鼓掌能够帮助你在分布式网络中成为一个有影响力的领导者。



《要有领导者的样子，像领导者一样思考》

作者 Herminia Ibarra（由 Jeff Mackanic 推荐）

在 本书中，Ibarra 诠释了《开放的组织》 一书中描述的领导力。她在本书中写道，为了能有个领导者的样子，我们应该去投入大量的时间去做如下一些事情：在各种人和组织之间扮演桥梁的角色、设想新的可能性、在改变的过程中走在前面、拥抱变化。Ibarra 然后提供了在这些区域中体现一个领导者的细节思路。

作者简介

Brayn 在 2011 年加入 OpenSource.com ，当他不在思考和撰写关于一切兼开源的时候，他会玩下复古的电游或者是阅读经典的科幻小说。在 2015 年，Brayn 拿到了北卡罗来纳大学教堂山分校通信工程的博士学位。在网络中，他常用的昵称是“semioticrobotic”。

开源的进化：
从开发更优质的代码到打造更优质的商业

作者 **Douglas Levin** 编译 **李建盛**

文章的一开始，作者回顾了自己对于过去十几年开源软件的简要历史。

从我创办开源软件公司以来，已经过去了 13 年了。开源在业界已经成为主流，而且软件的开发在方法论和结构上均发生了根本的变化。

早些时候，瀑布式的软件开发占据统治地位。软件的设计、编码以及质量控制均是由“自上而下”来管理的，即通过某个经理带着一帮程序员来完成的。此类方法是由 IBM 这样的大型企业在 20 世界 60 年代所引进，它并没有旨在利用互联网既是一个分布式文件系统和开发环境的力量，也没有考虑到防火墙之外无处不在的、快若闪电的网络通信。此外，它也不是为当今的移动软件开发项目所设计的。

开源 / 自由软件运动所带来的一些益处。

自由软件为用户提供了自由，这其中包括零花费。开源软件在一些许可证上又一些实用的方法，因此可能会有一些限制。但是它带来的最大的自由就是查看源代码的能力。

在早期的时候，四个开源项目－Linux、Apache、MySQL、以及 Perl/PHP/Python（简称，"LAMP"）－占有主导地位。LAMP 犹如灯塔一样，为初学者指明方向，让他们勇敢的进入这个自由的世界，而且鼓励码农们为项目提交贡献。他们使用了很多的开源许可证，但是主要用的还是 GNU 的通用公共许可证 (GPL)。

话题转到了开源的商业之路上。

开源的世界常被称之为“狂野的西部”，因为大多数的项目没有遵照死板的分层次的通话方式，多数有天赋的开发者还是没有报酬的，然而，组织生产的自由软件项目会按计划完成，重要的最终的产品质量不比专有软件的差。即便如此，很多实用主义的软件开发者和公司创始人－包括我自己－坚信自由 / 开源软件是可以和商业软件共存的。今天，专有软件的模式仍然存活着，而且，某些情况下，还在扩张。

这时，开源的公司应势而生，而且被指称为“纯粹游戏”的商业化开源冒险者。比如我所创办的－黑鸭子软件，是针对软件开发人员的审查源代码和目标代码，从而确定哪怕是一小段、一片、一块、甚至是整个树状的代码是有出处的，也会鉴定安全方面的问题。

当下的开源界对于许可证的使用，正在悄然发生着变化。

今天，在云计算中开源软件更是蓬勃发展，如 Docker、Heroku、OpenStack、及其它形成了全新一代的项目。综观起来，GNU 仍然是许可证中采用最多的，然而，MIT、Apache、还有其它一些宽松的许可证正在被很多的开源项目采用的首选许可证。

对于商业上的影响，作者列举了如下一些观点：

尽管风险投资界通过投资很多 OSS 纯粹演绎的公司或者是相关的下游公司来支持这种破坏性的创新，但在收入、产品策略、收购、以及其它因素均对诸如 IBM、HP、Microsoft、Oracle 有影响。感谢开源运动，这些个开源的公司更加的精简、更高效、且更能及时响应用户的需求。此外，无数的创新型公司不断涌现，借助开源项目的优势在支持项目中为客户提供商业服务，乃至后来，在云平台中利用他们的应用和平台。

自由和开源运动对于免费试用时代或叫做“免费文化”中起了推波助澜的作用，可以看到现在已经非常的普遍了，如在电子商务、企业级、以及 B2C 新产品发布等市场。自由和开源运动也影响了社区管理的概念，针对客户、影响力、专业人才或特别人才（如软件开发者）。互联网的社区贡献或“围观”访问量对于众包和众筹是水到渠成的事情。通过开源软件所带来的软件成本的降低也有助于大数据及其分析乃至移动电话技术。

最后，作者以自己的亲身经历来说明开源运动已经成为人们思维的主流，在波士顿体验 Uber 时遭遇了堵车，此时，司机对作者说到：“也许通过开源可以很好的解决交通堵塞问题”。作者会心一笑，知道自己所思考的开源，已经超越软件，成为社会思维的主流。

作者简介

Douglas Levin 是总部位于波士顿叫做 TechCXO 有限责任公司的兼管市场和销售的战略合作伙伴。他在 2002 年 12 月创办了黑鸭子软件，担任 CEO 一职直到 2010 年卸任，担任董事会董事到 2012 年。

如何为开源项目选择正确的品牌架构

作者 **Chris Grams** 编译 **李建盛**

大多数人在启动一个开源软件项目的时候，是不会坐下来等待和人们一起来讨论品牌架构模式的，但是他们中的绝大多数针对项目都有长远的目标，其中包括如何逐步地推出收费的产品或者是公司要基于项目的代码来提供服务和支持。

所以，尽早地考虑品牌策略是没有什么坏处的，但是随着项目的成长过一段时间考虑也是可以的。我希望你阅读下一篇文章：如何为你的开源项目选择一个具有品牌效应的名称。这是你的项目迈向成功的第一步！

另外一个非常重要的步骤是理解常见的品牌架构模式，即开源的组织如何考虑项目、产品和公司的品牌，以及这些不同的模式都有哪些优点和缺点。就我们在 New Kind 所做的工作而言，多年以来我们为很多个开源的公司提过意见，依据我们的经验对于每个项目而言没有完美的品牌模式。

我们可以看到每种模式都有优点也有缺点，理解它们的优点和缺点才是作出最好的决定的关键，那种模式才是最为适合你的企业。

一旦你开始思考关于项目、产品、以及公司品牌之间的未来的关系的话，首先应该问问自己下面几个问题：

1. 应该保持项目、产品、公司是一样的品牌吗？还是尝试创建和保护多个品牌？有很好的理由来考虑几个不同的策略的。
2. 你打算你的项目有多独立？通常一个开源项目，若是由企业进行严格的控制，那就很难吸引到社区的志愿者。社区志愿者通常不怎么乐意去为背后有公司的开源项目做贡献。如果公司和其产品的品牌是一致的，该公司若选择项目的品牌和现有的品牌一致的话还是有很大风险的。
3. 你要对于项目有多大的控制力？如果你在大型项目社区待过的话，知道很多的项目成员来自不同的组织且对不同的项目感兴趣。你会发现在公司、项目和产品品牌之间的密切协调并不能使他们有长远的工作动力。

以下是我所见过的三种最为常见的品牌架构场景，可以回答上述的问题，以及我从品牌的视角来看它们各自的优缺点。

我在上面所指出的，从品牌的角度讲只是讲了拼图中的一小部分而已。所以我邀请大家有任何的想法都可以追加到后面的评论，不过请务必谨记所描述的每个场景。另外值得注意的就是，我已经对此三个品牌架构的选项做了很大的简化。在实际的环境中，每种情形都会有不同的角度来看待，还有我尽可能集中在一篇文章中来覆盖尽可能多的场景。

场景一：项目、产品、以及公司使用相同的品牌

在此场景中，开源项目和商业的产品以及公司使用的是单一的品牌。在大多数情况下，他们会将一个付费的产品从免费的项目中分离出来，并会注明做了何种改进。也有对公司品牌和开源项目品牌稍作修改，但是无论哪种情况，核心的品牌对于这三者来说都是处于中心位置的。

案例

- 公司名称：NGINX
- 产品名称 ： NGINX plus
- 项目名称 ： NGINX
- 公司名称 ： Puppet
- 产品名称 ： Puppet 企业版
- 项目名称 ： 开源 Puppet

此种方式的优点

- 可利用品牌的规模经济：构建一个品牌是非常昂贵的。所以对于刚刚起步的小型公司，有太多的理由说明构建一个品牌要比构建两或三个更优，因为多构建一个品牌就意味着花更多的钱和精力。
- 由免费的积极的正面特质向付费产品和公司转变：如果开源的项目开发有了很好的声誉，可以声称它已经稳定、可靠，这时就可以进一步的考虑，让产品和公司都来沾项目品牌的光，有些使用免费项目的用户可能这时会考虑其产品和公司，因为他们对此熟悉而且信任项目的品牌。
- 简单而且熟悉：人们很容器获得此种方法，因为它相对较常见。而且它也毋需太多的解释，“品牌 Foo 是一家公司，是品牌 Foo 项目背后所支撑的公司，我们拥有的产品名称为品牌 Foo 专业版，是免费项目的高级付费版。” 收到，明白！

此种方式的缺点

- 不同的缺乏导致吸引付费用户的困难重重：这确实是一个大问题，若是不考虑免费的

项目和付费的产品之间的特性和功能的不同的话，仅仅考虑品牌，人们为什么要花钱买一个可以免费获得的同样的品牌？有时候一致的项目和产品有必要深入的挖掘特性，从而可以强调彼此的不同。从项目中区分出不同的产品的重任，若以不同的品牌来解决的话当然是再好不过的了。

- 越往后越难以改变策略：假如你开始了三品牌一致的道路，如果你顺利的通过了上述的考验，使哪些令人难以信服的人们成为了你的付费客户。你会发现你在公司和产品的品牌投入了很多，乃至社区的品牌也投资了很多，这时没有人愿意放弃它。抛弃有问题的品牌，再加上创建一个新的品牌或多个品牌，是件非常痛苦的事情。以我个人的经历为例，当年红帽从红帽 Linux 分离出红帽企业版 Linux（产品）和 Fedora （项目）就是铁证。在红帽的例子中，从其这么多年的运作中，无论是红帽企业版 Linux 还是 Fedora 项目，均是巨大的品牌资产。现在返回来看，当年的决定无比的英明，但是这在头一、两年非常的令人不安，无论是公司、和它的客户，还是社区均经历了分裂的痛苦。

场景二：产品和公司使用同样的品牌，但项目使用不同的名称

案例

在此场景中，公司的品牌和产品的名牌是一致的，项目是采用了不一样的品牌名。在一些例子中，公司 / 产品的品牌可能会以某种方式作出一些微妙的（也有不那么微妙的）联系，视觉上来通过 logo，或者是通过名称本身。

- 公司名称 ：Hortonworks
- 产品名称 ：Hortonworks Data Platform
- 项目名称 ： Hadoop
- 公司名称 ：Datastax
- 产品名称 ： Datastax Enterprise
- 项目名称 ： Cassandra

此种方式的优点

- 公司可以把控自己的未来：在很多情况下，组织都会在最终倾向于此种结果，因为他们既不想无法控制－或者是不想过分的绑定到一个开源的项目中。作出这样的决策有可能是因为转向需要不同的技术往下走，又或者是他们想保持开放想抛出更多的开源技术。通过不把自己定在单一的开源品牌下，是让未来有更多的选择。
- 很容易的找到不同点来让用户付费：在开源的世界里，哪些潜在的客户的想法通常都是（无论他们是否自觉的意识到）：“为什么我要花钱去购买一个我本可以免费得到的东西？”。不同的名称在什么是可以免费得到和什么是需要付费方能获得之间创建了立竿见影的效果。
- 产品和项目可以有不同的品牌特征：或许你打算将开源的项目推到使用前沿技术的境地，一致将其推到极限。但是你又会让客户所购买的付费产品是他们可以信任的稳定的、可扩展的解决方案。两个品牌的名称赋予你将一组特性关联到一个品牌，将另外不同

的一组特性关联到另外一个品牌的能力。再举红帽的例子，其红帽企业版 Linux （稳定、可靠、安全）而 Fedora 项目（最新的技术，创新），这就是最鲜明的最好的实例。当产品和项目使用同一个品牌名称时候很难做到如此显著的不同。

此种方式的缺点

- 管理更加的昂贵：更多的品牌等同于更多的品牌曝光，若是只有很少的预算以及没有足够的市场资源的话，想构建一个品牌就非常的困难了，就不用提两个了。如果你没法去承担开发者社区的重任，这个问题可能会稍轻点，但是，正如我通常给一些公司的建议一样，尽量集中投资到最小化的品牌上，因为每个额外的品牌也就意味着分裂品牌资产且会缩小经济体的规模。
- 难以构建项目和产品之间的联系：产品 / 项目的分离就是一把双刃剑，好的一面就是拥有两个不同的品牌会让你为付费产品创建独特的价值主张，它可能很难让人们会意识到产品是构建在相同技术的项目之上的。当年我们构建红帽品牌的时候，如果我们无法利用其人们和红帽 Linux 的关系的话，我们会异常困难的找到付费用户。会发生“如果你喜欢免费的红帽 Linux 版本，你应该去尝试一下你所信任的完全支持的版本。”这样的事情。

场景三：项目和产品使用相同的品牌，公司则使用另外的品牌

在此场景中，公司的品牌既和产品品牌不同也和项目品牌不一样。

案例

- 公司名称：Canonical
- 产品名称： Ubuntu Advantage
- 项目名称： Ubuntu
- 公司名称：Continuum Analytics
- 产品名称： Anaconda Pro
- 项目名称： Anaconda

此种方式的优点

- 项目和产品品牌之间的紧密联系更容易的去考量产品：如果某人已经使用了免费的项目了，而且想进一步的指望更多的特性和获得更多的支持，他会找寻相同品牌的付费产品，同样，这也是一把双刃剑，因为如果人们已经获得了免费的版本，若要吊起他们对付费版本的购买欲望，付费版本的优点要非常的明确，能够让人们有强烈的愿望去升级。
- 分离的公司品牌远离了产品和项目：这里有几个原因说明为什么一家公司要和产品的品牌保持距离。首先，公司打算投资多个产品 / 项目，且不想将整个品牌仅仅和一个产品关联起来，如果是那样的话，可能会限制发展。第二，它限制了未经公司同意的项目方向上的风险，公司可以随时将项目砍掉，切换到新的产品 / 项目品牌的方向，还不需要

进入更改公司名称 / 品牌昂贵的流程。

此种方式的缺点

管理多个品牌的昂贵代价：此策略通常会强制公司作出决定，那就是在那个品牌投入更多的钱：是产品 / 项目品牌还是公司品牌？对于一家成长性的公司在市场上全力支持和促进两个不同的品牌是一件异常困难的事情。

将项目和公司联系起来异常的困难：对于很多人来说不够清晰，因为在公司和项目之间缺乏一个强关联的纽带。Canonical 在这方面就是一个非常好的例子，绝大多数来自开源界的人们都听过 Ubuntu，但是只有很少一部分人听说过 Ubuntu 背后有个公司叫做 Canonical，所以 Canonical 做了一个慎重的决定，投资项目和产品的品牌，而不是公司的品牌，公司使用此策略的后果就是类似下面这样的对话的结局：

- 员工：“我在甲公司工作”
- 路人：“谁呀？”
- 员工：“我们就是做了非常酷炫的开源项目 Foo 的背后的公司。”
- 路人：“哦，我非常喜欢酷炫的开源项目 Foo，从来没有听说过背后还有个公司了。”
- 员工：（无语、尴尬、欲哭无泪）

可以大量推广的结论

社区和公司的关系愈紧密，越是对保持一致的公司、产品和项目有利。这里的关键在于从项目向公司转换时的品牌考量。

开源公司最大的弱项就是很难说服让哪些可以免费获得的产品的人们掏腰包，这就对于付费的产品的价值主张要明确而有力！

对于社区和公司关系较松散而言，就需要考虑多品牌的解决方案的意义。这让公司可以保持自己的选择权，但是需要更多的精力去解释以及花销更多的品牌费用。

我曾经见过一个小公司拥有三个不同的品牌，项目、产品、和公司各占一个，我以为这是在自杀，一家收入没有达到一亿美元的公司想支撑三个品牌简直是异想天开。我奉劝他们不要做，他们后来对我感激不尽。

正如我原先所强调的一样，这些基本的品牌架构策略仅仅只是冰山的一角，但是还是希望本系列文章的解释能够带给读者在考虑接下来要走的路上澄清了一些疑惑，并开始思考自己日益增长的开源项目的品牌架构的未来。

作者简介

Chris Grams 是 New Kind 的总裁及合伙人，也是《毋需广告的品牌建设：成功在数字世界建设品牌的秘密》一书的作者。他的联系方式有：Twitter 帐号、领英主页、电子邮箱：chris(at)newkind.com

如何为开源项目选择一个具有品牌效应的名称

作者 **Chris Grams** 编译 **李建盛**

当涉及到开发一个新的开源软件项目时，多数的开发者是不会花费大量的时间去考虑品牌策略的。因为当你在一个进行中的项目时，毕竟会以为一个伟大的想法、牛逼的代码、以及一个充满激情的社区才是重要的。

多数的项目发起人所做的品牌策略的第一个决定就是为项目选择一个合适的名称。对于大多数人来说，命名就是从一大堆的科幻小说列表中挑选出来，然后对照是否有其它项目采用了这个名字，或者是依照内部的一个笑话，添加一个额外的元音或辅音从而构成一个新的名称，并将之申请为让人们可访问的 URL。

但是除了这样做之外，我们是不是应该像给我们自己的孩子起名字一样，在项目开始之前应该认真的考虑清楚？

在过去的 15、6 年中，我有幸在几家开源的技术公司工作并从事品牌战略这一部分。尤其是我在红帽工作的 10 年期间，处理过很多复杂的开源品牌问题，这也是我在品牌公司 New Kind 的角色，在 New Kind 的同事中，他们都曾在一些有名的开源公司如 Ansible、NGINX、以及 Continuum Analytics 等工作过，且都是为将来业务的增长开发品牌策略的岗位。

这里我会以一系列的文章和大家分享我多年来在品牌和开源中学习到一些东西。我的目的是希望能够帮到那些在开源项目的开始时就能深入了解品牌策略，同时也希望能帮到那些已经运行了一段时间的开源项目的公司或人们，理解到一些品牌的基本概念，以在未来的工作中能够用得上。

此次，我以几个简单的贴士开始，来看下如何研究和选择一个开源项目的名称。

寻找自己可以拥有的品牌空间

在首要考虑的事情中，其中之一就是你的开源项目的名称是你的项目的长期目标。

如果你是像大多数人们一样是在项目刚起步阶段的话，你或许在梦想着有一天你的开源项目能够成为售卖的产品，或者是围绕着项目代码的支持和服务来构建一公司。而事实上是，如果你是站在未来的角度希望你创建的品牌成为宝贵的财产的话，你就必须对你能够拥有的品牌作出选择。

在开源的世界中，我经常看到人们常犯的一个错误就是，为他们的开源项目选择了一个名称，但是他们却无法保护其注册商标。通常情况下是名称已经被另外的组织或项目使用了，又或者是使用了一个太过于常见的名称，从而非常困难或干脆不可能获得保护。

在选择最终名称之前，考虑下使用一个简单的、非正式的审批流程来通过这些步骤。虽然这并不一定能够保证你的命名是免费和清晰的，但是至少会帮助快速的除去那些你没有希望拥有的名字。

在商标数据库中搜索

在美国专利和商标管理局的商标数据库中查询你的名字是否和已有的有冲突，来开启你的品牌策略之旅。测试不仅能够确定你的拼写，而且能够找出与你所选择的名称可能相混淆的名称。

如果你看到了一个已经注册的商标和你选择的一样，莫慌，这也未必意味着你就得将自己的名称扔掉。要知道使用相同的名称在任何行业的任何品牌是可以的，所以你更要额外仔细的查看其在科技行业（你通常会得到一个涵盖商标注册的类别，通过检查商标页面上的商品和服务）。如果你看到的结果是，已经注册的和你所选择的名称所在行业非常的类似，这个时候你可以继续往前，开始继续选择你的下一个名称了。

看看名称是否已被其他公司使用

如果你确定了名称是自由可用的，且在商标数据库中明确可用的话，那么接下来就是使用搜索引擎如 Google（以及可能的话也要在相关站点如 GitHub）来查找是否有其它公司或项目使用了相同或相似的名字。请记住，美国专利和商标局只能保护到在美国注册的品牌，所以你所选择的品牌名称有可能在这个世界的其它地方被使用了，又或者是仅仅是尚未注册而已（这并不意味着拥有名称是一件公平的事情，因为哪怕是未注册的商标仍然受到明确的保护）。

请自行启动在 Google 中搜索你所选择的品牌名称。如果这是一个虚构的名词的话，是非常有意义的（所谓的商标，什么律师类型等称之为“天马行空”的名词）。还有什么事情要做的吗？当然也要搜索一下和你的名字类似的一些名称。是否在技术行业或其它地方有非常近似的名称？做完这些以后，如果你觉得自己的名称空间已经比较确定了，或者说至少在技术行业中清晰的看到没有类似的名称，但是你仍然希望再进一步明确下，做到万无一失的话，要做更加具体的搜索。

举例来说，你或许对被其它开源项目所占用的品牌名称惊讶不已。做一次如“你的品牌名 开源”简单的搜索是一个不错的开始。以及如 “你的品牌名 软件”“你的品牌名 技术”“你的品牌名 公司”等尽可能的多覆盖一些领域。如果你的项目涉及到技术的特别类型如 云计算 或 容器 或 微服务 等 ，那么像上述一样搜索下特定的类型也是非常值得的。

找寻完美的网络域名

经历上述这么多步骤以后，此时的你终于对于自己所选择的品牌命名空间有信心可以拥有了，接下来要做的就是找到你最喜爱的域名提供商，开始考虑一个域名了。什么样的域名是足够简单、容易被人记住、而且还是不太贵的了呢？

如果常见的 .com 和 .org 不可用时，一定要确保它们是没有被某些公司或网站使用的，否则可能会引起混淆乃至进一步损害到你品牌的未来价值。举个例子，我们自己就有这样一次不太爽的经历，我们曾经选择的域名竟然是特别的超级令人毛骨悚然的色情网站。在此种情况下，我们最终处理掉了该域名且删除了违规的域名，但是这样的处理结果不会总是有效的，若是处理不妥的话，想要彻底解决恐怕需要花费不少钱的。

访问 FOSSmark.org 站点

如果你仅仅是刚刚开始了自己的起名之旅，而且想知道一些更多的关于起名过程中的关于法律的部分，FOSSmark.org 正是你所需要的资源。它是有一帮知识渊博的且对开源有着丰富的经验的知识产权律师们所创建的，此站点对于解释基本的商标法、一些个在选择一个名字是要时刻铭记在心的重要法律事项、如何注册一个商标、甚至是如何设计和实施一个商标的政策，均给出了非常有价值的信息。

结束语

在这里衷心希望，大家能够在开发自己的项目名称时上述的这些最为基本的研究步骤能够作为简单的提醒，你做了哪几项？根据我个人的经验，我非常的清楚，许多人都会跳过一个或多个步骤。

还有一件值得和大家提的事情，那就是这些活动不可避免、无可替代。假若资金充足的话，还是请一些知识产权律师，对于你选择名字时作一些正式的研究以及给出经法律分析后的风险，最终甚至帮助你申请商标名称。

但是，通过遵循上述这些简单的步骤的话，你可以节省出此过程所耗费的不必要的支出，以及能够避免让你进入死胡同。

17 个 Apache 基金会的运营之道

作者 **李建盛**

17 年前的今天，Apache 基金会（ASF）成立，以开发人类大众有用的软件为使命！今天来自 ASF 的主席 Brett Porter 和 主管市场、公关的副总裁 Sally Khudairi 宣布道：

我们的成功是证明了在 Apache 的产品、品牌、以及社区的背后的开发者、用户、贡献者以及“粉丝”们的成功。今天我们以如此自豪的方式庆祝：是创新让这一切变为现实！

下面我们就来自 Apache 官方博客所列出的 17 条独一无二的方法，作一一的诠释。

1.“一个修修补补”的服务

大家可能听说过 Apache 是一个双关语 “a patchy Web server”，意思为一个修修补补的 web 服务，即通过一系列的补丁做的服务。但是这并不是 Apache 真正的起源。“Apache”的名称由来是为了纪念受人尊敬的各种美洲土著统称为 Apache 的种族，他们以战争中高超的作战能力和不竭的耐力而著称。那些当年发布此 web 服务软件版本的开发者们当之无愧的称为：“Apache 团队”！ 21 年过去了，Apache HTTP web 服务器依然是这个世界上运行最多的 web 服务器。

更多内容，请移步关于和常见问题。

InfoQ 编辑点评：

中国有句成语叫“名正言顺”，此乃做一件事的根本，即初心！ 在商业社会中这叫做“品牌”！

2. 基金会的诞生

ASF 的成立是基于会员制、非盈利的形式，这样可以确保 Apache 的项目可以在没

有个人志愿者参与的情况下依然能够继续存在。独立的个体，若要有加入 Apache 的资格，需要证明自己能够在开源软件的开发中通力合作、并通过在基金会的项目中持续的参与和贡献方可。

ASF 本身由社区来治理，且是最为直接的服务——在其自己所合作的项目中来进行。今天，ASF 开发、管理和孵化了超过 350 个开源项目和计划；通过其领导力、健全的社区、以及精英主义思路，铸就了著名的“Apache 之道”！ ASF 被誉为开源界最具影响力的成功的基金会。

更多内容，请移步认证以及基金会主页。

InfoQ 编辑点评：

一件事情需要更多的人参与时，必须选择一种治理方式，这也是社会、国家存在的基石，即人类的至高理想—政治。正如 Linux 的发展一样，最初由 Linus 自己，发展到需要 Linux 基金会来治理一样。Apache 基金会这个非盈利性组织的成立是今天成功的主要原因，这里涉及到原则、策略、处理办法等。

3. Apache 羽毛 logo

原先的 Apache 羽毛 logo 以及第一个站点均是由 ASF 的合伙人 Randy Terbush 在他位于内布拉斯加州的林肯市的卧室 office 中所开发的，而这也是 Randy 注册 apache.org 域名的时间（1995 年 4 月 11 日）。他当时标注的内容是：“作为一个团队，有幸选择‘Apache’作为其名称，是出于对印第安人发自内心的尊重，尤其是，Apache 部落的组织方式 ..... 羽毛的神圣象征对于这些人来说简直是天作之合，颜色则是代表了我们这些 伴随 WWW 的出现的人，意味着‘觉醒’！”。

在 2016 年，ASF 作出了历史上的第一次，更换了 logo 及羽毛，同时保持不忘初心！

更多内容，请移步这里。

InfoQ 编辑点评：

这个细微的变化，可能很多人都没有注意到。这是一个团队成熟的标志，开始注重品牌，开始注重给人留下良好而深刻的第一印象。

4. 快速的成长

你知道嘛？在过去的 16 年，ASF 凭借着有限的资源将自己成长了 35，000% 倍。由最初的 21 名见证了 HTTP Web 服务的过程的独立成员，发展到现在的 588 名独立成员和来自六大洲的 5427 名提交者。这一切的发展都是建立在自愿的基础之上的—有一些贡献者是由他们的雇主为其时间和代码进行有偿支付，但是 ASF 本身并不会为软件或者是项目的监督支付任何的费用。

更多内容，请移步如何运转？

InfoQ 编辑 点评：

这就是开源软件崛起的明证，不仅仅是软件本身，而是关于协作、文化、人类的共同理想。很难想象某一个公司不花钱去生产出如此伟大的项目和产品。但是 Apache 真的做到了。

5. 提交者 + 成员

在 Apache 的生态世界中，“贡献者 -> 提交者 -> 成员” 这样的称之为精英路线的方式是核心的控制流程。要想成为贡献者，就是为 Apache 社区贡献代码、补丁、或者是文档，其中一些贡献者会由成员来指定为提交者，一旦拥有了提交者身份，会增加一些特权：1）直接提交（写入）到代码仓库；2）社区相关的决策的投票权；3）提出活跃的提交者。这些提交者在随着基金会的成长、进化、乃至进步中表现优秀，则会由现有的成员正式提名为 ASF 的成员。

更多内容，请移步如何运作、治理组织表、角色篇、通讯录。

InfoQ 编辑点评：

作为一个社会的团体，其发展方式一定有一套特定的规则。也就是说一个人在其中付出了时间和精力，就一定要有所威望的提高，符合“马斯洛金字塔”的说法，否则，可能就需要特别的报酬。这套规则非常的重要。

6. 超过 300 个项目和记录

你知道有 171 个委员会监督着 286 个项目嘛？这还没有包括某些项目下的众多的子项目及其相关创新计划。自从 ASF 成立以来，就被公认为是开源的，具备可互操作、适应性强、可持续的能够满足需求的网络服务、网络客户端、以及程序库的解决方案提供商。从 Abdera 到 Zookeeper，对于 ASF 的可靠的、企业级的软件需求在多个领域都在大幅的增长，特别是大数据方面，Apache Hadoop 是这个市场的领头羊。在 2015 年，Gartner 魔力象限报告对于 Apache 的产品超过 400 次。

更多内容，请移步 Apache 项目。

InfoQ 编辑点评：

这就是目前 Apache 成功的外在表现，毫不夸张。有心人可以具体的看看那些知名的项目！如 Maven、Spark、Kafka、Mahout、Lucene、Mesos、Cassandra ..... 等等。

7. “若没有在邮件列表中讨论过，则当没有发生过。”

你知道吗？所有的 ASF 官方的沟通均是通过邮件列表完成的！

这些个“虚拟的会议室”可以异步的进行对话，这是一个遍布全球覆盖几乎所有时区的（算上所有的 Apache 社区）团队所必须的。众所周知，Apache 团队是构建在透明

的文化之上的，团队的合作就在邮件列表中进行，在 1475 个可公开访问的 Apache 邮件列表中有过百万的信息归档，这纪录着 ASF 过去 17 年的成绩！

更多内容，请移步贡献、提交的状态、如何工作、邮件列表存档。

InfoQ 编辑点评：

邮件列表看起来是最为古老的沟通形式，但是它代表的是网络何以诞生的原因之一，平等、透明、明主、精英，这又何尝不是开源社区成功的基石了呢？

8. 宽松的项目管理

Apache 项目的监督均是有贡献者自己选出来的团队来搞的。ASF 不会去为 Apache 项目的技术趋势进行指导的：项目管理委员会（PMC）仅为项目的日常运维提供向导指南，包括社区开发和产品发布等。ASF 董事会会认命一名副总裁（公司的高管）来作为 PMC 的主席，PMC 主席的主要角色是行政上的，包括为董事会提交关于他们项目的健康状态的季度报告。对于某个项目来说，其有此主席的参与并不能为其带来任何额外的权重或影响，此主席和其他 PMC 成员一样只有一票的投票权。

更多内容，请移步基金会、治理、15 周年的主席声明。

InfoQ 编辑点评：

“绝对的权力导致绝对的腐败”，对于项目的管理也一样，行政不能干涉技术，让最擅长自己领域的人做自己领域中的决策，这应该是常识；而且要尽可能的控制行政力量的扩散，防止其利用手中的权利来干一些事情。

9. 孵化创新

在 ASF 中，一个项目或社区若要想成为 Apache 成熟的项目的话必须通过 Apache 孵化器。这其中既包括外部组织所捐赠的代码也包括 Apache 内部所创建的项目，比如 Groovy 这个内部项目，在 2015 年 11 月份才入选为 Apache 的顶级项目。目前在 Apache 孵化器中正在孵化的项目有 54 个（称之为“podling”），有 39 个创新项目在 Apache 实验室中的创新“沙箱”中在测试技术概念。

目前为止，有一个例外，那就是项目 Apache Zest，它目前是唯一一个没有进入过 Apache 孵化器而是直接成为 ASF 的 pTLP —临时 Apache 顶级项目——的一个项目，当然，作为审核的一部分，Apache Zest 必须满足 Apache 成熟度模型的严格要求，这也就意味着 Apache Zest 必须向 ASF 提交满足要求的项目代码、版权、许可证、版本、构建、独立性、以及其它的品质相关（Apache Zest 在 2015 年的 3 月份成功进入 ASF 官方的顶级项目）。

更多内容，请移步孵化项目、实验室、精英主义。

10. 公平的竞争环境

你知道吗？ Apache 的项目都必须是不受商业影响的独立治理的。作为一个厂商中立、非赢利性的组织，ASF 和 具体的 Apache 项目都不会去选择哪一方，赞同或支持某一个厂商压制其他的厂商。另外，ASF 认为竞争是件好事，所以不会去贬低开发具有“竞争性”的产品。

ASF 厂商中立的合作环境，是那些无论是盈利还是非盈利的商业模式的第三方能够继续热情的参与的根本。

更多内容，请移步项目独立。

11. 遍布全球的 Apache

你知道吗？这个世界上哪些点击率最高的网站，从Google到维基百科再到新浪微博，都是由 Apache 所驱动。

Apache 的产品撑起了互联网的半边天，可以管理上 EB 的数据、也可以执行浮点运算的操作、甚至几乎在每个行业中都存储着上亿的对象，它们改善了全球无数用户和开发者的日常生活，可以在几乎每一个计算终端设备中找到 Apache 产品的身影，从笔记本电脑，到平板，再到移动电话。在金融服务、航天航空等运行着关键的任务，在出版、大数据、云计算、移动、政府、医疗、研究、基础设施、开发框架、基础库、等类别，Apahce 均有所涉猎。商业友好和宽松的 Apache 许可协议、开放的开发模式是受到广泛认可缘由。

到目前为止，有成千上万的软件解决方案已经在 Apache 的许可下发布。

更多内容，请移步许可证。

12. 技术精英主义

ASF 平均每个月的代码提交行数是 16000 行！ ASF 就是负责这些由来自开源界的无数的贡献者们提交的数百万行代码。Apache 的提交者对于集体的社区创建产品有着不可懈怠的责任，且要确保提交的代码足够的清晰，能够让其他成员可以理解并基于其之上进行编码。还有，Apache 的代码提交者还对维护社区的健康成长负有一定的责任。

更多内容，请移步版本发布以及开发提交。

InfoQ 编辑点评：

其实这条很好解释，我都觉得其官方说的太过于平淡。简单一句话：“过硬的代码就是一切！少废话。” 当然，最初的那些成员以及成员所选择的提交者都要有过硬的技术水平。

13. 会议和线下活动

说实话，学习技术和项目最好的方法就是在 ASF 直接找到某些大牛去沟通！

ApacheCon 即 Apache 官方的技术研讨会议，将开发者和用户聚在一起，以“Apahce 之道”来讨论构建开源的解决方案。ASF 坚信要为所有级别的人们提供参与的机会，并尽可能提供第一手的内容，均是来自项目、软件、社区背后的开发者和专家。而且还会为那些有资格参加但是由于费用等原因无法参见 ApacheCon 会议等人提供差旅费用。

另外，除了 ApacheCon 之外，ASF 还举行其它一些大型的会议和活动，包括 BarCamps、 Hackathons、MeetUps、key signings 等等，活动范围覆盖北美、欧洲、亚洲三大区块。

这里打个广告，下一届的 ApacheCon 连同 Apache：大数据，将于今年的 5 月 9 号到 13 号在加拿大的温哥华举办，届时恳请莅临！

更多内容，请移步 Apache 会议、技术会议、线下活动、旅行。

14. 社区胜过代码

ASF 有个专门的团队：社区开发团队！此团队的责任就是帮助新手学习关于 Apache 的项目、治理、活动，并且会为新手们如何成为技术精英、Apahce 社区的全职志愿者提供指引。

“社区胜过代码”是基金会核心原则的基础！这是能够成就 ASF 今天所拥有的超过 500 名的成员以及超过 5000 名提交者，能够让他们通过尊重、诚实、专注于技术的交流的根本之所在。

社区团队的一个关键例子就是每年参与 Google 编码夏令营（GSoC），ASF 从 2005 年该活动创建伊始就开始参与指导组织。每年 ASF 会指导 30 ～ 40 名 GSoC 的学生，为他们提供并接触实际的软件，并学习“Apache 之道”的社区开发。曾经受过 Apache 社区指导过的，现在发展为长期为 Apache 上各种各样的项目作贡献的学生难以统计，有的甚至成长为 ASF 的成员！

更多内容，请移步社区、ASF 15 年文章。

InfoQ 编辑点评：

这个在本土文化中很难见到，秘方文化熏陶下的国人，多数的选择是“代码胜过社区”，君不见俗语云：”我们什么都缺，就是不缺人。”随意走进一家软件公司或所谓的社区，统计下人员流动就明白了。最难的就是文化的改变。

15. 吸收新的贡献者

ASF 专门开发了一个“招募贡献者”的应用程序，旨在帮助志愿者们匹配合适的 Apache 项目，以及那些正在寻找帮助的活动。这个项目刚刚创建不久，期望不仅能够帮到迷茫的新手们，而且还能够找到可以长期贡献即有兴趣的合适的项目。请关注我们要构建的功能列表！我们非常期待您的参与！

更多内容，请移步招聘和如何贡献。

InfoQ 编辑点评：

这个 web 应用对于目前庞大的 Apache 项目来说招募新手是个不错的想法，类似于“众包”这样的应用，简单的根据开发者、兴趣相关者所提供的信息来匹配。如果让我提建议的话，不如分析 GitHub 的账号还靠点谱。未来的路还很长，期待更多的完善。

16. 基础设施

每天大约有数百万的来自全球各地的人们在访问 ASF 的数十个服务器以及 75 个不同的主机。ASF 的基础设施团队是来自 3 个大陆的分布式的团队（其中包括 10 个轮询的志愿者和 5 个领报酬的员工），来保证 Apache 的服务在 24x7x365 不间断的接近 100% 的运行，而年度项目预算也只有不到 $5000，正如大家所知道的，ASF 是一个纯粹的虚拟组织，没有办公室或者是任何的建筑，唯一的物理上存在的就是基础设施，技术上需要实际操作的。

更多内容，请移步状态、基础设施、以及开发。

InfoQ 编辑点评：

什么时候 ASF 会将这些基础设施迁移到云平台中，比如 AWS、GCE、甚至是微软，等等，那时候 ASF 就真的是彻头彻尾的纯粹的虚拟组织了。Apache 目前的项目本身就是互联网的基石，丝毫没有任何的见惯不怪。

17. 企业角色和赞助商

ASF 隶属于美国私有的 501(c)(3) 非营利慈善组织，通过对企业、基金会、独立个体的免税捐赠而获得资助。ASF 的年度预算是一百万美金，大约有 75% 是专门用于基础设施支持服务的。ASF 除了接受通过诸如比特币、PayPal、亚马逊、以及银行卡等形式的捐助之外，还接受一些关键的赞助，以帮助日常的运营，如带宽、接入、服务器、硬件、法律和会计服务、品牌管理、公共关系、以及一般的办公支出和支持人员的薪水。ASF 不允许企业直接参与 Apache 项目管理或相关的治理活动。ASF 厂商中立，参与仅限于个人，不参杂任何的关系和雇佣状态。

更多内容，请移步赞助商贡献、以及 ASF 15 周年时的解释。

InfoQ 编辑点评：

保持中立！ 这是原则性的问题的。NGO 组织可能在本土无法存在。所以这个就看看罢了。

其中最为让人赞叹的是任何厂商无法掌控，只能以个人的身份参与。这一点就是 Apache 能够如此受欢迎的最为关键的原因。

Docker 三周年
Docker 之父 Solomon Hykesgo 如此说

作者 **Adam Herzog** 译者 **刘腾飞**

为了庆祝 Docker3 岁的生日，我们正在邀请更多还没有了解或者使用 Docker 的人加入到 Docker 社区。同时我们还准备了一个有趣的培训来让大家更好的了解到如何轻松地通过 Docker 来构建，交付以及运行分布式应用程序。在这里要非常感谢我们的活动组织者以及超过 580 位 Docker 专家一起来帮助我们完成这个培训。Docker 社区已经在全球 5 大州，接近 50 个国家组织了 120 个生日庆祝活动。

我们也和一些组织合作来确保每个人都可以参加到这些活动中来，包括那些在技术社区被忽略的少数派。我们会在这周晚些时候的博客上发布更多相关信息，敬请关注。

最近，我们有了一次机会和 Docker 的创始人、CTO 以及首席产品官 Solomon Hykes 坐到一起回顾了 Docker 一路走来的这三年。想知道 Solomon 对 Docker 社区的生日愿望，以及为什么技术的多元化和开源对他来说是如此重要么？今天我们一起来发现！

你对 Docker 社区的生日愿望是什么？

我的愿望当然是希望 Docker 社区能够围绕在它的一些核心理念下不断地发展壮大，

特别是我希望能够看到在社区里的每一个人更加理解彼此。这是一个非常庞大并且多元化的社区，我们有开发者、运维人员、企业、狂热爱好者、新手以及专家，我们会有不同的观点、不同的目标、不同的专业背景，这对于我们来说实际上是一件非常好的事情 。

和所有社区一样，你会有争执，有时候大家会有强烈的情绪因为大家都关心事情会怎么样发展。我的愿望是这些不同的观点最后能够让我们互相学习，每一个人都能在这个社区里面成长。

你觉得过去三年或者过去一年，Docker 社区做了什么事情是你特别开心的?

当我们刚刚开始设计 Docker 的时候，它只是给运维人员的一个工具。我们构建它是希望能够在 dotCloud 使用它来管理我们众多的应用环境，并且在不同的团队之间分享。只是没有想到后来有这么多的开发者直接用它来构建他们自己的应用。

开发者开始直接使用 Docker，是希望能够更好地控制他们自己的开发环境，就这样演变成最后有一半使用 Docker 的人是开发者，而不是全部都是运维人员。 在最开始的时候，我们通过培训运维人员来为其他的运维人员构建一些工具，通过这样的方式间接地帮助开发者。然而，到现在我们开发者社区已经很大了，这是我们没有预料到的一个转变。我们很开心有这样的一个转变，让我们有机会可以同时为开发人员以及运维人员提供一个这样的平台来帮助他们更有效率地工作。

为什么多元化技术多样性对你和 Docker 社区来说这么重要？

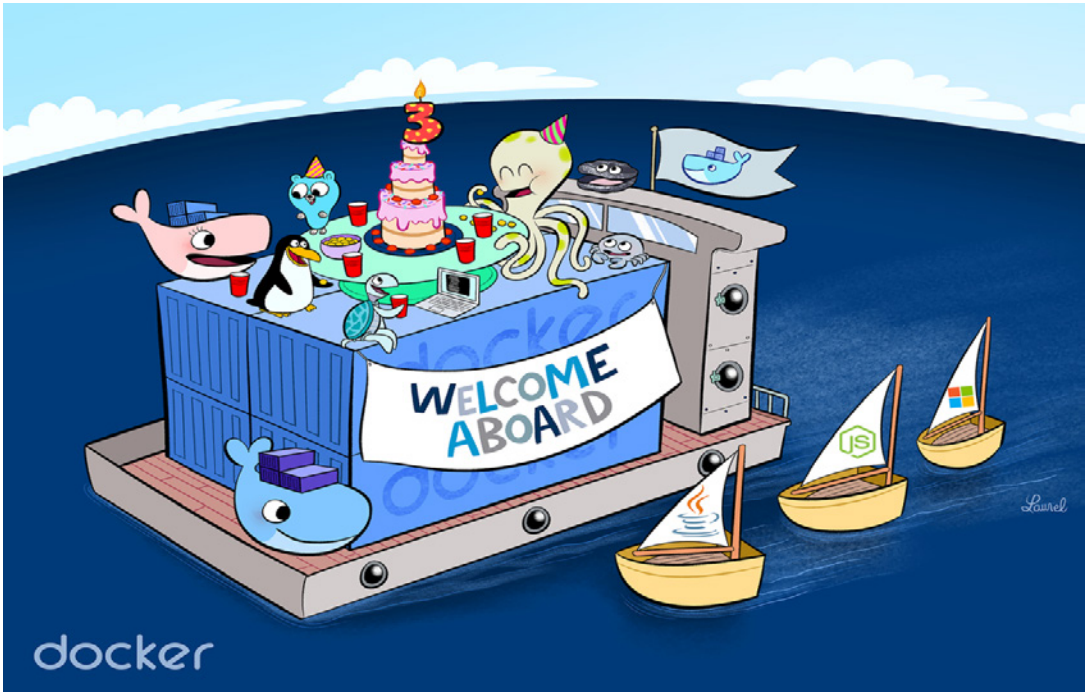
由于 Docker 现在正在做的事情，我们发现我们自己处在很多不同世界的交汇的中心。我们连接开发和运维人员，我们连接开源和企业，我们连接云端到数据中心，我们通过那些全新的前沿架构连接那些老式软件方法论。所以我们也会招聘一些有着不同背景的人，以至于在 Docker 你会发现在自己总是能容易遇见一些有着和你不一样，出奇不意的想法和观点的人，然后从中学习。慢慢地，这就变成了我们文化中的一部分。

你能举个例子来讲讲 Docker 是如何提高多样性的么?

我觉得我们还有很多事情要做，和每个在技术领域的人一样。我们绝不是多元化的最好例子，但我们的确认为它很重要。我们已经认识到我们的团队越多元化，我们就会越强大。

我很骄傲我们在公司以及全球的一些和 Docker 相关的活动中，我们已经和一些女性开发者合作来鼓励这种多元化。今年，我们将会继续在一些大型活动中保持这种合作，像 Docker 三周岁的活动，DockerCon 以及 Docker 聚会。

在这里我也非常高兴地给大家分享一个奖学金计划来加强我们的 Docker 社区，同时继续提升这个技术社区的多元化。DockerCon 奖学金计划将为那些传统领域中被忽略的少数派提供帮助使他们可以参加 2016 年的 DockerCon。 这个计划的目标是帮助这些成员可以访问一些资源，工具以及辅导来加速事业或者教育上的成长（点击这里获取更多



关于如何申请这个计划的信息）。

在技术圈里有什么人是你特别敬仰的么？

约翰逊凯瑟琳（Katherine Johnson）在计算以及航空这方面是一个杰出的开拓者，她做了很多令人难以置信的事情，比如为阿波罗 11 号计算飞行轨迹。她从 1950 年作为一个航空技术人员开始为 NASA 工作，那个时候一个科技领域的女性工作者并不容易，付出了很多坚持不懈的努力。她是很多人心中一个非常鼓舞人心的榜样，不仅仅是对于女性和技术领域的那些少数派来说，还包括我和很多其他人也是。

为什么开源对你来说这么重要？

我觉得有两个原因， 第一是它在关于怎样去创造事情这个点上是一个很大的突破。开源可以让更多的人聚集在一起来用比以前更快速的方式来创造一些野心勃勃的东西。一个开源的社区可以抛掉很多障碍，让我们更好地去创造，因为我们有一群来自五湖四海的人团结在一起，在以前这是不可能的。因为没有软件以前，你必须人在同一个地方才可以高效地合作。在没有互联网以前，让不同地域的人一起工作也很难。

第二点的话，开源软件的自由也满足了创造者最终的梦想。如果你是一个创造者，你希望处在一个创造能力最强的人总是可以获胜的地方。有很多的创造者对于自己工作环境并不满意，因为总是存在很多的限制，并且并不奖励和支持自己创造。你不可以在做出初步的原型之后通过展示给其他的人看来获得资源和帮助，在多数情况下，优秀的人被迫去制作一个幻灯片，做一些花哨的东西去获得注意力，并且学会在会议室说得比别人大声。所以，对于那些想要更好地融入到组织去实现最大价值的创业者总会有一些痛苦。

开源是一个入口，它很难但是最好的项目最终会获胜。你可以跟大家讲所有你想要的东西，但是如果它不够好，它是可以立即被所有人看到的， 所以它是一种令人难以置信的竞争方式。但是这是开源所模拟出来的最健康的方式，它让你保证诚实。这是我为什么开源对来我说很重要 。

有哪件关于你自己的事情是大家都不知道的?

很多人知道我骑摩托车，但是他们不知道的是我喜欢安全地骑着摩托车！大家可能认为我是一个疯狂的冒险者，但其实我是一个安全并且谨慎的骑行者。我觉得作为一个谨慎的骑行者也很有趣。

同时在过去的 9 年里，我也一直在参加一项传统的越南武术运动叫做：Tay Son Vo Dao。这也是我生活当中很重要的一部分，是一个平衡工作很好的方式。

有没有什么事情是你从这项武术运动中学到的然后运用到了你生活当中的其他地方?

在过去的 9 年的时间里，我遇到过很多沉浮，在人生的低潮期，你需要一些东西来保持清醒，可以说是纪律吧。参加武术运动对这点就很有用，当你的脸被人揍过，你就知道这并不好受然后不想再来一次。它能把你推出舒适区让你更严肃地对待你所做的每一件重要的事情 。在 Docker 有很多人包括我自己，不管你让他们做什么事情，他们都不会主动放弃。我觉得有了一个好的导师，你可以将你在实践中所学到的很多东西应用到一生当中所有其他的事情上。

FOSS 历史回顾：三代开源人的故事

作者 **李建盛**

现在是 2016 年，你环顾一下四周，开源早已无处不在了。开源无论是规范、形式、以及面貌都和最初的大相径庭，然而事实上，这也预示着新一代的开源程序员们的崛起。下面我们尝试解释下。（以下这一段落为作者自谦）

为了避免我是在互联网上散步谎言的嫌疑，我需要声明几代的想法纯属虚构。我很清楚这其中并没有某人是某人生理上的下一代，我清楚的意识到只是说哪些第一个写自由和开源软件的前辈们，而且说上个世纪的事情。

作者解释了题目，以免产生一些不必要的误解。比如 Python 编程语言的技术研讨会就会经常被环保主义者抗议。根据常识，一代这个概念用于衡量文化的变革是个非常不错的概念，如果读者能够从这个角度来看待自由和开源软件的话，我认为你可以很容易的看出三个分离的代。

第一代开源人

第一代是 Richard Stallman 那一代。Richard Stallman 在上世纪 80 年代发起了自由软件运动，他们创建了 GNU 和 FSF，而这就是差不多十年后 Linux 能够诞生的基础。他们也更加的倾向于视自由软件为一种道德讨伐，而且他们在相对主流的技术世界仍然是处于边缘的位置。

第二代开源人

第二代的来临则是伴随着 Linux 内核的诞生而开始的一代，他们是第一次能够访问能够正常运行的自由 / 开源操作系统的一代，而此操作系统是 Linus Torvalds 的内核加上 GNU 的套件组合而来的。（作者在此声明，他至今没有完全搞清楚自由软件和开源软件的区别。）

关于自由软件与开源软件的区别，恐怕是作者故意所为，或者是讽刺那些视自由软件，或者对 GPL 许可证无视对人们的吧。

第二代相比于第一代要少一些思想，Torvalds 和他的追随者或合作伙伴们之所以青睐开源主要是因为其带来的功能，而道德的原因很小。他们认为这是一种更加有效的编码方式，能够以更加便宜、平民的方式使用电脑。但是，他们仍然保持独立，没有成为某些大企业的爪牙。

第二代开源人也是将 GNU / Linux 带入主流的一代。他们所写就的代码不仅是开源的操作系统更加的完善和实用，而且在追求至高，能够和专业的闭源平台进行正面的竞争。他们在上世纪末和本世纪初面临的是和微软这样的大鳄的艰苦斗争，关于此，年轻一代的程序员可能难以理解。在 2000 年左右的那些个不怎么活跃的开源程序员或用户，会想当然的认为，他们使用 GNU/Linux 并不会担心会被起诉。

当下这一代

也就是说的第三代的开源程序员和用户们，他们长大成人后，GUN/Linux 已经是部署在数百万台服务器上运行的操作系统了，此时已经没有人再质疑开放源代码的价值了。对于这一代人来说，开源是毋需争议的默认选择。

基于这个原因，什么理想、什么功能主义统统都销声匿迹了。今天多数的开源程序员不曾放弃代码是因为他们认为这是在道义上正确的事情，或者说他们认为这样更加的高效。他们这样做的原因其实是没有更多的项目让他们去选择。从云计算（其中，OpenStack 更是说明这事的根本）到大数据（什么 Hadoop、Spark、以及正在吞噬专有软件地盘到 NoSQL 数据库）乃至 SDN，甚至是 NFV，开源都是占主导地位的。如果你是在这些生态系统中做事情的话，你一定得用到开源代码。

很大一部分开源的支持者都毫无疑问的认为这是好事情。在另一方面，有些人也开始担心，开源界正在失去前两代人所努力争取的东西。对于许可证的使用的趋势基本都是 Apache，而放弃了 GPL，这对哪些认为 Apache 许可太过于宽松的人来说，更加的紧张不安。

同样，在开源领域，企业对其的影响是越来越大——尤其是大家最近争论的 Linux 基金会改变了某些章程——引起了社区各界的各种紧张。

最后说一点，声明这是非常重要的一点。开源社区最近几年在拉拢着微软，这虽然对于第三代开源人来说是比较正常的，但是那些曾经在昔日战斗过的人们是坐立不安的。

开源真的已经进入了一个全新的时代，再也回不去了？这是一件好的事情吗？这些都是很主观的问题。但是值得我们去深思，尤其是我们正在准备迎接 Linux 内核 25 周年的这个时刻。

关于小米在开源上的五大原则
一位 20 年开源老兵的思辨

作者 **崔宝秋**

从 1995 年至今，从美国到中国，从自由软件到开源软件，小米首席架构师崔宝秋一直在参与开源软件的开发等工作，亲历中美两国开源的变迁与发展。如今，开源是互联网的大势所趋，个人与企业都纷纷拥抱开源。

回顾过去，开源这些年，有哪些值得铭记的瞬间、有哪些影响深远的事件、有哪些极具代表的人物？放眼未来，开源到底该怎么玩，怎样的思路与打法是值得国内互联网公司借鉴的？在 3 月 27 日举行的技术社群大会上，崔宝秋与大家分享了这些年的一点感触。本文根据崔宝秋在大会上的演讲整理而成。

大家好，我是小米首席架构师、小米云平台负责人崔宝秋，今天我分享的主题是《我看开源这些年》。本次分享中，我会介绍下过去二十多年我对开源的一些看法和理解，重点是在国内我们应该怎么玩开源。

我和开源

首先，我先简单介绍下我和开源的缘分，我是什么时候、怎么和开源结缘的。我是 1995 年出国的，那会儿国内还停留在 DOS 和 Windows 编程。1995 年去美国读书，在那里我可以说是大开眼界。我看到了 GNU、Linux、FreeBSD，还有 Emacs，被 GNU 和 Linux 庞大的功能、后面所有的软件、能力、水平所震撼，当时真有惊艳的感觉。我所在学校的几个教授也在开源上对我有很大的启发作用。之后我的四年博士研究项目，五年博士中的四年，都在做 XSB。XSB 简单来讲就是增强版的 Prolog 语言，当时我们基于 GPL 把它开源，到后来很多学术界和工业界机构也在用这个系统。从 1995 年开始到 2010 年，这

图 1

图 2

中间我参与了很多 Emacs 社区讨论，也直接贡献了一些代码。在 LinkedIn 我们也开源了一个分布式实时搜索系统 SenseiDB，后来小米也把它用到了搜索技术里。（图 1）

2012 年回到中国加入小米，我就开始推进小米的开源战略，比如在有限的人力的情况下，如何站在开源巨人肩膀上快速推出一些云存储、云计算和大数据处理能力。

在这个过程中大家可以看到，我是从早期的一个自由软件的信仰者，慢慢地转化到了一个开源的倡导者，中间也经历过学校、公司，从个人的爱好到公司的推动，从项目贡献者到项目管理者。

开源关键人物、关键事件和基金会

说到开源，讲开源过去的这些年，我觉得应该讲一下重要的开源人物。在我看来，在开源史上、在开源界这三个人是至关重要的：第一个是 Richard Stallman，他创建了 GNU，创建了自由软件基金会，为后来的 Linux 萌芽等等奠定了扎实的基础。第二个，

图 3

图 4

Linus Torvalds，Linux 的创造者，这个不用多讲，大家都很了解了。最后一个是 Eric Raymond，他创造了 Open Source 这个词，并发起了开源运动。我觉得如果没有 Eric，只凭着自由软件，开源做不到今天。这三个人在开源史上未来都很难有人取代的。（图 2）

图 3、4、5 是开源历史上一些重要的里程碑事件和重要基金会，以及重要的开源软件时间表。

版本控制系统（VCS）和代码管理平台的演进

讲开源，就离不开源代码，离不开版本控制系统。三十多年来，版本控制从早期的 RCS，到 CVS、SVN，一步步发展到了今天最流行的 Git。一个好的版本控制对提升团队的开发效率至关重要。

对于开源来说，代码管理平台也非常重要。我记得在我读博士的时候，我们的系统刚开始都是在系里自己维护的，整体来看，并不安全，版本管理也不方便，不利于与外

图 5

图 6

界的合作。开源软件，势必要有很多人参与，所以最重要的代码一定要放在易于合作、安全可靠的地方，所以后来我们就决定把 XSB 放到 SourceForge 上面。SourceForge 是当时最流行的代码管理平台，在 GitHub 出现之前，它管理了很多开源软件的源代码，项目数基本上是随着时间线性增长的。相对来讲，2008 年有了 GitHub 出现后，Git 所管理的开源项目数量就不是线性增长，早期可能有点线性，但是后来的曲线非常陡峭，有点指数级的味道。

开源软件 VS 自由软件

自由软件我觉得过于理想主义，过于强调人的自由。开源软件更加实用一些，较少涉及政治和道德，强调“开源软件对你有好处” 。确实，早期 GPL 推动了自由软件的萌芽或者早期发展，但是后期开源也一定程度上限制了自由软件的发展。所以我觉得开源软件属于工业界，我这两年参加了国内一些开源社区活动，大家对开源的热情都很高。

图 8

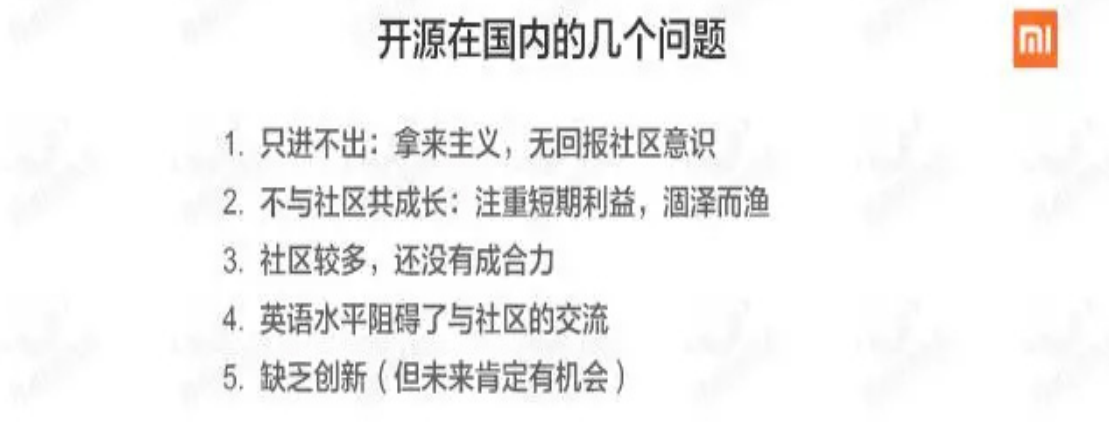


图 9

虽然早期我对 Eric 有些有负面的认识，但是他对开源的贡献是不可替代的。

开源的三种力量

开源有三种力量，这是我总结的。第一种力量是个人爱好者。这些人是狂热的技术爱好者。我很早以前就有一个目标，就是等我财务自由或者退休以后，就去写自由软件，享受那个每天 commit 代码，为大家创造价值的成就感。这第一类力量就代表了一大帮技术爱好者。其中我合作过的一个作者，他是一位天文学的教授，叫 Carsten Dominik，是一个非常好的文本管理器 Emacs Org Mode 的作者，可以帮你管理时间、记事、写作等。他编写 Org Mode 的工作几乎都是在上下班两个小时的地铁上或者轻轨上完成的。刚开始我看他的有些代码写的不太优雅，算法性能不够好，让我有些吃惊，因为我一直以为他是一个计算机编程高手，后来我发现他是学天文学的，也就理解了。虽然他的一些算法不是特别美妙，但是他对 Org Mode 这个项目的推进还是有非常大的贡献的。（图 6）

图 10

图 11

第二种力量我叫无开源商业模式组织，这里包括公司和学校，这些组织是以互联网服务或者以某种产品来盈利的，它不是直接通过开源的软件或者搭建于开源软件之上的解决方案来赚钱的。

第三种力量是有开源商业模式组织，包括 IBM，也有中国的华为，它们有一种商业模式在后面，利用网络硬件，设备，开源，有一整套的解决方案，对大的企业，通过整套的服务来盈利。

这三种开源的力量，对开源看法不同，贡献不同，投入也不同。

国内开源的几个阶段

中国的 Linux 在过去的二十年发展的非常迅猛，但在座的可能很多人不知道这件事情：这是我在 1999 年 10 月份的一天早上在 LWN 网站上看到的 Eric 写的一篇文章，我当时看了觉得不对。他就是表达了对 Linux 进入中国的反对或者不相信的态度。

图 12

图 13

图 14

图 15

他说 Linux 被中国政府采购是不可能的，不应该，也不被欢迎。这表达了他的一些政治偏见。我后来跟 Eric 和 RMS 也进行了电子邮件交流。那次交流让我认识到 RMS 确实把自由软件当成了一个政治的东西来看待的，它不是资本主义或社会主义那方面的政治，而是另一个政治观点，也就是关于自由的。（图 8、图 9）

国内开源我认为有这么几个阶段：第一，我认为 1995 年之前是与世隔绝的，这是加了引号的，我们确实落后了，因为国外已经玩了很久了，而国内还没有互联网。后来十年是初级阶段，2005-2010 年初步成熟了，现在有了 Git，开始了快速发展。

开源的玩法我简单介绍下，重点可以看演讲稿，个人的就是热爱，就像刚才讲的天文学家一样，个人玩开源的也有盈利的，成为暴发户的都有可能的，因为你可以接受赞助等等的。像大公司，有商业模式的公司的开源是另外一种玩儿法，我没有直接参与，就不讲了。我讲的是中间这个无开源商业模式的公司或组织的一些玩法，以小米为代表。大家可以看一下小米大数据做了都用了哪些开源软件，这是一个简单的全貌。为什么拥

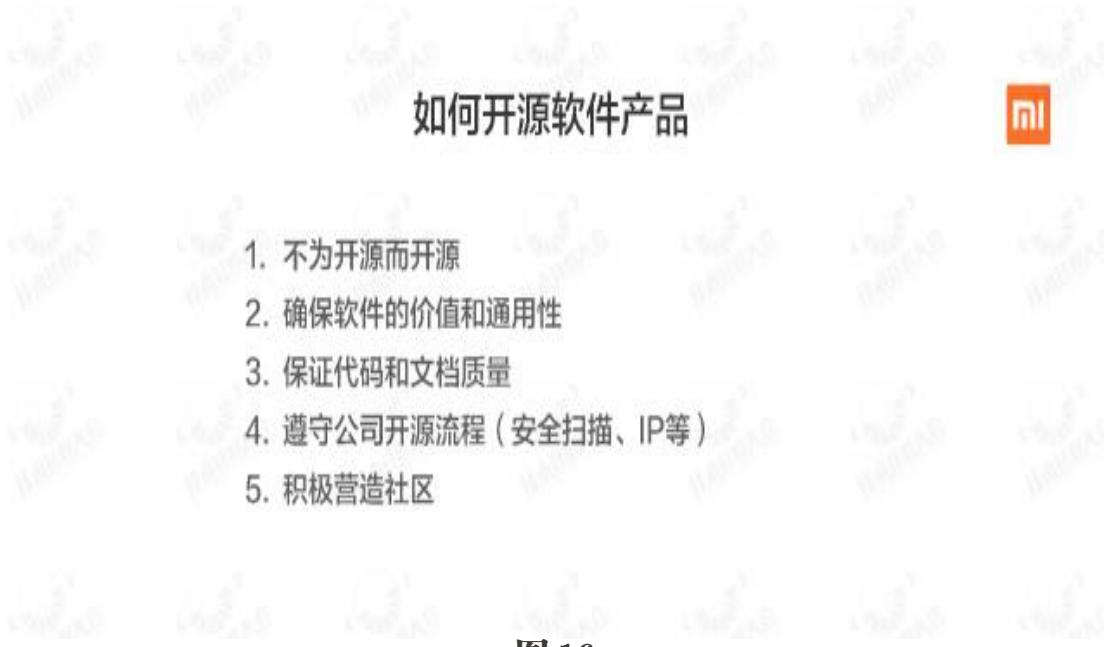


图 16

抱开源？我们首先要站在巨人肩膀上，快速推出产品，快速占领市场，快速为公司创造价值。（图 10、图 11）

所以对很多创业公司，甚至对很多大公司而言，不拥抱开源就一下子输在了起跑线上。当然吸引人才，吸引贡献，提升内部软件质量，我认为也是开源非常重要的几个好处，所以拥抱开源已经是大家公认的。

在小米我一直推行这几个原则：一，快。快速选型，快速定位，快速掌握，快速推出产品。二，绝不重造轮子。宁愿学习掌握，化为己有而不要自己重写，这个投入远远大与前者。三，不用则已，要用则精。四，永远抱着开放与共享的态度。如果讲自由、讲政治上的一些东西、讲道德层面，应该就是一块。五，与其他公司所不同的，小米在重要开源项目上会尽力推出自己的 Committer。

这里，尽力推出自己的 Committer 的目的就是想解决国内很多公司所犯的一个很粗浅的错误，造成这个错误的最根本的原因就是它们拿了开源社区中的一个版本，创建了自己的分支后就把这个分支和社区主干的联系给割断了，然后自己用起来很爽，围绕自己的业务进行改进，改动后又不回报到社区，不管是自己不愿回报也好，还是自己的改动不被社区接受也好，这个本地版本就被这个公司维护得越来越重，和社区主干分得越来越远，差别越来越大，最终这个版本实际上就没用了，成了无源之水，无本之木。开源社区是开源项目后面一个庞大的力量，相对而言甚至 BAT 的力量我认为都是小的，比不过社区后面力量的推动。如此下去公司最终会被本地分支的维护成本所拖垮，达不到持久享受开源的好处。

所以，开源 5 大原则的最后一点很重要，在座的各位如果在公司想推开源的话，这点一定要注意。

如何回报社区

怎么有效回报社区？参与交流，这个主要是针对国内开源爱好者或者参与者、贡献者。第一，要勇于参与；第二，清楚地描述想法和算法；第三，还有就是自信，坚定自己的观点，不要被社区中某些貌似大牛的人所吓到，没必要的；第四，一定要负责任，不要打一枪换一个地方，就是写了代码一定要维护，要做 QA，要做测试，要做自己的用户，要做客服，关于你的代码有人提问要及时回答。小米为什么这么快让社区接受了我们，就是我给他们展示了，小米为什么会在某些领域大力投入等等的，所以这个时候社区愿意跟你合作。（图 12、图 13、图 14、图 15、图 16）

我想总结一点结束今天的分享，一句话，我认为开源是软件的未来。

对话 Linus：Linux 25 岁啦

作者 **张天雷**

作为芬兰 Helsinki 大学计算机科学系的学生，Linus Torvalds 在 1991 年编写了 Linux 操作系统的原始内核。很快，Linux 就发展成为了一个全特征的操作系统，并开始运行在智能手机、服务器和各种设备中。在本次通过 e-mail 进行的访谈中，Torvalds 回顾了过去 25 年的历程，并展望了未来 25 年可能的发展。

Stephen Cass（以下简称 S.C.）：相比于 25 年前，现在你已经是一个经验非常丰富的程序员了。哪件事情是你现在知道，且最想让年轻时的自己尽早学习到的呢？

Linus Torvalds（以下简称 L.T.）：实际上，我一直在说一个事实：我当时并不知道我所做的工作会引起以后 Linux 的很多成功。如果我当时就一直知道了我现在所知道的事情，我可能没有勇气去编写我自己的操作系统了：我需要一定的“天真”才会认为自己能完成这项工作。我真的认为，这份“天真”是启动该项目并取得成功所必须的东西。不理解这个项目最终的形态以及在对其前景没有很多预期的时候着手去做其实是帮了很大的忙的。

我不知道该项目最终的结果形态意味着，我会比知道其最终形态抱有更加开放的心态，能够接受被人的建议或影响。我认为，这种开放的心态使得其他人加入该项目更加容易，也更加有趣。人们不用一定要采用别人的版本，他们可以自己的需要进行修改。我认为这激励了很多人的参与。

S.C.：有没有一个在 Linux 发展早起所做的技术决策是你现在觉得最好当时选择另外一个方向的呢？

L.T.：有关错误的技术决策的事，好处的你总是可以回退。当然，错误的决策会让人觉得泄气，而且很多时间和努力都白费了。但同时，它最终也不是完全浪费了：肯定有一个原因导致你做了错误的决定，而意识到该错误就教会了你一些东西。我并不是说，错误的决策是一件好事——但我并不会在做决策时特别担心。我宁可选择一个在以后证明为错误的决策，也不愿意在可能的选择中徘徊很久。

在 2001 年左右，我们在 Linux 虚拟内存子系统方面就遇到了一个非常有名的坏情况。

当时，大家对于采用那种方法起了很大的争执，而且我们在某些内存配置方面存在很大的问题。系统的主要组件在大家认为的“稳定”时期的时候被清除。人们非常不开心。

但是，现在回头去看，虚拟内存子系统最终工作很好。当时，它给人们带来了非常大的痛苦。如果能够不再中期的时候再进行逆转，那就最好了。但是，它并不是灾难性的。

S.C.：随着 Linux 的快速发展，从一个开发到多人参与的情况的转变是什么呢？

L.T.：对于我而言，的确存在两个明显的转变。一个发生在大概 1992 年，我当时开始采用别的开发人员的补丁，而不再自己重写。另外一个则比较晚，当时我自己亲自应用所有的补丁已经非常痛苦，我不得不学会信任其他的维护者。

第一步要相对容易很多——由于 Linux 内核编程的大约头六个月都是我一个人完成的，当人们开始把补丁发给我的时候，我并不习惯采用这些补丁。所以，我会亲自查看该补丁，确认编程人员的目的，然后我会重写编写代码——有时，代码类似；有时，我会采用完全不同的方法进行实现。

这种情况很快就难以持续了。在我们开始足够信任别人以致我不再重写根据其想法编写自己的代码后不久，我就开始直接应用他们的补丁。但是，我仍然会经常对补丁进行修改。这么多年过去，我已经非常擅长阅读并修改补丁，以至于我能够一直修改直至睡着。而且这种模式在很多年内都工作的非常好。

但是，正是因为“采用别人的补丁”这种模式多年来工作正常，我一直都非常适应这种模式。对我而言，改变这种模式非常痛苦。在 2000 年左右，我们在内核发展方面有了很大的进展（此时，Linux 开始被商业玩家所注意）。人们开始抱怨我的工作流成为了系统发展的障碍，并提出“Linus 不与时俱进”。但是，我们并没有好的工具来进行源代码的管理。

这最终导致了我们将 BitKeeper 引入近来，作为源代码的维护工具。尽管人们是在若干年以后因为 brouhaha 的版权问题而记住 BitKeeper 的，它的确是完成我们的工作的最合适的工具。而且，它教会了我源码控制是如何工作的以及我们如何能够根据一个更加分布式的开发模型进行合作。

当然，关于分布式源码管理的东西都是在 2005 年从 Git 学习到的。Git 已经明显成为了了源码控制方面最成功的项目之一，而且它让更多人学习到了分布式控制的优势。我们在 2000 年左右内核方面所经历的痛苦最终成为了一个很大的教训，但是这中间的过程不可置疑是痛苦的。

S.C.：还存在像分布式源码控制这样的其他项目，使得你想自己去从头开始完成的吗？

L.T.：没有。而且我也真的希望不会有。我的所有大项目都来源于“天哪，没有人替我完成这种工作！”的时候。当别人能够替我完成工作时，我肯定会比较开心我可以

不必花费时间来自己完成。相比于自己解决问题，我肯定会选择躺在沙滩的大型遮阳伞下喝喝冰镇饮料。

好吧，其实我在撒谎。过一段时间，我肯定会开始厌烦这种生活。我非常开心能够Linux 项目的陪伴——它非常有趣，而且充满了刺激。但同时，开始新的项目肯定是需要艰辛努力的。

S.C.：你为什么认为 Linux 绝对不会成为主流桌面电脑的主要操作系统？

L.T.：好吧，我还努力在推进这项工作。我认为，即使 Chromebooks 只是一个非常有限的桌面环境而非传统的全 Linux 工作站模型，它的表现越来越好。

关于 Linux 为什么不能成为桌面电脑的主流操作系统，原因有很多方面。其中一个最大的原因就是用户的惰性。在计算的世界中，桌面电脑是一个非常独特的存在：它既非常个性化（如果你需要使用计算机进行工作，你基本上每天都会和它进行交互），又非常复杂化（很多其他的计算环境不会在这些方面不会如此复杂）。

以智能手机为例。这是计算技术的直接应用场景，而且深入人们的日常生活（多亏Android 系统，Linux 在智能手机上工作的非常好）。桌面电脑由于很多历史原因要复杂很多。它是一个非常难以进入的市场。与手机更加不同的是，人们有很多自己已经习惯的应用程序和工作流，这就导致很多人非常不愿意切换到别的操作系统。选择将预装的操作系统替换为其他操作系统的人数非常低。

同时，即使整个通用桌面电脑的市场都在一定程度上存在凋零，在很多任务的更加专业化也更加简单的平台仍然是一个非常重要的市场。例如，智能手机、平板电脑和Chromebooks 就是例子，它们都是非完全的通用环境。

S.C.：Linux 的哪些应用场景最让你觉得吃惊？

L.T.：最近吗？只从我认为Linux 已经基本成为新的硬件或服务建模的默认环境后，就没有那么多了。如果你有一些古怪的、特殊的设备，或者你正在设计一些新的互联网框架，这些东西没有采用 Linux 作为操作系统会真的让我觉得吃惊。

但是，当我将 Linux 视为工作站或服务器的操作系统时，这些设备的应用领域的确曾让我非常吃惊。在一些早期的 Linux 会议中，人们会演示运行 Linux 的气泵或冰箱。当时，我感觉非常意外。当第一个 TiVo 面世的时候，它运行 Linux 的事实就像“你可以将现场直播回退”一样有趣。

S.C.：Linux 现在面对的最大挑战是什么？

L.T.：内核现在实际上工作的非常好。人们开始担心事情可能变得太复杂以至于超出人类理解和修复漏洞的能力。这是一个非常好理解的担忧。但同时，我们有很多聪明的编程人员参与其中。系统已经发展的如此巨大和复杂以及很多人都依赖 Linux 的事实使得我们提前开始准备很多事情。让系统变得功能丰富并做出很大的、能接受的改变是

非常觉有挑战性的。因此，我不会将其称为令人感觉快乐的工作。但是，我认为，内核的发展目前十分正常。很多其他拥有这些资源的的开源项目可能已经都失败了。

这就意味着，我们在内核方面的一个持续性的挑战就是硬件种类很多。我们支持很多不同的硬件——基本上肯定比其他任何操作系统都要多。但是，基本上每天都会有新的硬件面世。尤其是在嵌入式领域，硬件平台的开发周期通常要短很多（在中国，你可以在 1-2 月内创建一个新的手机平台）。在这种情况下，硬件和系统的整合就要艰难很多。好消息是很多硬件厂商也都在努力。过去，这些都不可能。

S.C.：你现在感兴趣的技术趋势是什么？有什么让你气馁的吗？

L.T.：我一直对新的硬件，尤其是 CPU，非常感兴趣。这就是我为什么一开始研究自己的操作系统，并看到新的平台后非常开心。当然，新的平台很多时候都是现有硬件的简单修改版本（而且，我也相信这就是技术发展应该走的路线）。但是，这就是我努力去追踪的东西。

从更高的层次来看，但并非局限于我所参与的领域，观察 AI 最终如何发生的是非常有趣的。以前，AI 的技术都停留在 20 年前。但它也一直超前 20 年。而且，我对于人们过去做的基于规则的模型不感兴趣。

现在，神经网络终于开启了人工智能的新篇章。我发现，神经网络非常有趣。这不是一个我正在参与或将来会参与的领域，但它的确让人觉得非常有趣。与疯狂的 LISP 和 Prolog 语言方法不同，我们所知的神经网络都是从自然界而来。而且，我觉得，AI 或许会像很多人所想的那样，最终会出现。一点也不觉得意外！

S.C.：你认为 Linux 在其 50 年庆的时候仍然会处于活跃开发阶段吗？你认为 Linux 系统在 25 年后应该怎样呢？

L.T.：我不是一个非常有远见的人。我是一个非常枯燥的宅男工程师，而且我一直努力将精力放在具体事务上。我会让其他人在预测 5 年、10 或 2 年后 Linux 的未来——我认为，只要我们尽力做好每天的工作，未来也一定会非常美好。

如果 25 年后的社会经过了大的变革或变得与现在非常不同，肯定会非常有趣。但是，有很多基本的问题，无论是在 Linux 之前、真正的操作系统刚刚出现的 19 世纪 60 年代，还是在 25 年后的未来，肯定和现在一样。我怀疑，我们在过去 50 年已经看到了很多发生在计算中的出乎意料的变化将来不会再发生。和软件工程师一样，硬件工程师已经学会了什么可以工作、什么不可以工作。

当然，神经网络等将改变整个世界，但你不会来对其进行“编程”。人工智能可以学习。他们是模糊的。我可以非常肯定的保证，他们不会替换传统的计算模型。人们希望比机器聪明，但又希望机器能够按照要求完成应该完成的工作。因此，我们老的计算模式不会消失，它将会得到增强。

版权声明

InfoQ 中文站出品

出版：极客邦控股（北京）有限公司
北京市朝阳区洛娃大厦 C 座 1607
欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译，请联系 editors@cn.infoq.com。
网 址：www.infoq.com.cn


技术 干货
活动大本营
Geekbang 极客邦科技
InfoQ
免费 大会 课程 沙龙