TechTarget 中国

# 为什么使用 DevOps

# 为什么使用 DevOps

有多种不同的技术变革都在深刻影响着企业 IT 部门工作的方式。虚拟化，云计算，软件定义一切，大数据，一切皆服务——这些都迫使 IT 做出改变，并关注新的工作方案，DevOps

。

## 初识 DevOps

DevOps 是 IT 交付过程令人兴奋和具有深远意义的转变。承诺很诱人： 从根本上提高生产力，更低的成本和更可靠的系统。

- 重塑 IT 组织 实现 DevOps 战略
- DevOps 如何提升 IT 运营人员的形象？
- 如何使用 IT 新利器——DevOps

## 部署 DevOps

将软件开发和 IT 运维团队整合到单一的 DevOps 组织可以带来更强大的软件开发项目交付能力，但两类团队的文化差异、以及缺乏有效的工具都会阻碍 DevOps 的成功。

- 如何辨识 DevOps 是否合适？
- DevOps 成功的两个关键
- 利用 DevOps 工具搭建开发与运维之间的桥梁
- 五大 DevOps 最佳实践实现安全、可伸缩和性能

## 敏捷与 DevOps

DevOps（开发运营）运动在修补残缺的部署流程方面取得很大进步。从表面上看，为开发者提供的解决方案是更多地站在运营者的角度去思考，反之亦然。

- 在大型机上使用 agile DevOps 的四点好处

# 重塑 IT 组织 实现 DevOps 战略

DevOps 是 IT 交付过程令人兴奋和具有深远意义的转变。承诺很诱人： 从根本上提高生产力，更低的成本和更可靠的系统。

那么，你应该跟随潮流，让你的 IT 部门去参与开发运营融合（DevOps），对吧？大家都开始干了，所以你也别落后。最大的问题不是干不干，而是怎样干，从哪里入手？

首先，走出去，招聘一些 DevOps 人才。等，等 DevOps 可不是一个岗位哦。
好吧，你应该创建一个 DevOps 工作组或者部门，对吗？在一个 DevOps 主管的带领下，你可以训练出一个伟大的 DevOps 团队——再等等！！——DevOps 既不是一个部门，也不是一种职能。

好吧，如果它不是一个岗位、 一个职能或一个部门，那 DevOps 到底是什么？

DevOps 是文化、流程、技术和人。DevOps 战略是将许多学科融合为一套有凝聚力的组织原则。DevOps 是一种新的 IT 系统交付方法，承诺更快交付、更高质量、终结全球饥荒、获得永生！好吧，也许不包括最后两个……DevOps 也意味着 IT 组织在时间、经历和资源各方面的庞大的投入。

DevOps 也就是应用程序开发和系统运维的融合 可以改造和提升 IT 交付能力。DevOps 概念同样也极易被滥用，所以我认为应该立一个警示牌。警告：DevOps 滥用可能导致丢饭碗、 业务中断、和 CEO 郁闷地汇报。

**DevOps 为什么不容易**

首先，让我们弄清楚 DevOps“不是什么”。DevOps 不是 Chef、Puppet 或者 Salt Stack，或者任何其它工具、脚本环境或者技术。虽然自动化是 DevOps 的核心理念，但它不等于自动化。

DevOps 也不会是你组织里深层次技术和专业技能的替代品。去专业化，并不意味着你要解雇你所有的 Linux 或者 Oracle 专家。

对于高度集中的科技公司例如 Yahoo 、Facebook 或 Yammer，那里的所有 DevOps 主管想要取得 DevOps 成功非常艰难。对于传统企业，特别是大型分布式组织，在整体意义上的 DevOps 成功往往是不可能实现的。

为什么这么难？一个原因：文化。

DevOps 要求深层次的文化和组织变革。需要改变行为习惯 ——要改变的太多太多。这意味着大家要扔掉奉行了几十年的显规则和潜规则。你不得不告诉老部下们 ,大部分他们知道的和每天做的事物都已经过时了。

IT 组织结构调整很容易。我可以把开发人员和运维人员安排在同一个房间，安排他们完成工作，这也是一种显式的调整。但是 ,这两个不同的群体的人 ,带着多年不一样的培训经历和不同的技能 ,真能够融入 DevOps 组织吗？要知道，他们可能来自不同的星球！

**怎样搞定 DevOps**

想要为 DevOps 和应用灵活性而重塑你的团队，需要领导层的勇气和无私奉献。当然，它也需要花费时间和金钱，并且需要在团队成员筛选上做出艰难的决定。

为了促进 DevOps 战略，调整考核和激励机制是必要的。如果你依然只根据敲出代码的生产力来奖励开发人员，或者根据基础设施的可靠性来奖励运维人员，那么，什么都不会改变。相反，应该奖励系统创建和运维的整体团队，并且根据团队工作的全部要素来确定奖励。

围绕业务系统而不是职责来组织工作，这就是 DevOps 打破 IT 分组壁垒的寓意。一个团队应该有开发人员创建代码，从用户界面到业务逻辑和数据结构，也应该有运维人员负责操作自动化和部署。

人们需要知道他们需要对什么样的系统负责，而并不仅仅是毫无责任地从一个系统换到另一个系统。唯一的例外是专家。

团队待在一起，共同为他们的应用和系统负责。

不要制造一个团队支持太多应用的局面。在这个预算缩减的年代很难考虑周全，但是经历这种融合转变之后，你的团队将更加富有成效，并且不需要额外人员就能应对需求的增长。

你需要充足的专家参与项目和为团队提供支持 ——这些人都是不同学科的大师和专家。他们为系统提供支持，但是不会长期指派给某一个系统。他们不需要对这些系统负责。

全面自动化 —— 部署、 升级、 扩展、 维护、 数据卫生、 测试、 监测、 安全和策略管理。在自动化方面投入巨资，目标是 100%的自动化，不考虑低于 90%的可能性。但是，全面自动化也可能会引起自动化泛滥。集中审查和调整可以控制 Chef 或 Puppet 脚本库的无序增长。

针对整个团队进行奖励和表彰。成功离不开大家共同的努力，同样，问题需要整个团队自我反省。系统团队应该承认专家们的贡献 ，对表现杰出的专家给予奖励。

围绕建设运营融合的原则制定新的企业体系结构标准。这将确保系统符合运维的需求。

DevOps 战略必须获取本组织自顶向下的全面支持。整个行政领导团队 ——不只是首席信息官 ——应知道它为什么重要和怎样使它取得成功。

请记住，这是重大的文化和组织变革，多分配些时间开展培训和组织开发活动。如果你变革得太快而忘了和您的所有团队步调一致，你将整天陷入失误和冲突——最后满盘皆输。

*(来源：TechTarget 中国　作者：John Treadway 译者：徐继军)*

# DevOps 如何提升 IT 运营人员的形象？

当 IT 运维人员开始失宠时，来自 DevOps 运动中的“形象提升”方法应该是最佳的应对方案。

不像 DevOps（开发运营组合），IT 运营的地位倍受争议， 他们被普遍认为存在公关问题。DevOps 的解决方案能否像把运营专家安插在产品开发团队中，让其自吹自擂这么简单呢？一位来自 Etsy.com 的高级副总裁认为：“这是前进之路的一部分。”

DevOps 运动的倡导者 John Allspaw，也是 Web 运营和容量规划相关书籍的作者，曾效力于多个国际著名网站。他目前是 Etsy.com（一个类似 ebay 但专注于手工艺品的网站）的技术运营高级副总裁。我们咨询了 Allspaw 关于 IT 运营人员引入 DevOps 运动之前与之后的对比，然后为他们的 IT 运营人员如何提升形象提出一些建议。

**开发和运维在 Etsy 是怎么工作的？**

**John Allspaw**：在 Etsy，开发团队是沿着功能的边界分开的，比如：移动、支付、防欺诈和搜索功能，每个团队都有一位特定的运维工程师，他参加团队的周会，获悉项目的进展情况，得知是否有新的需求要制作出来。他同样负责培训其他的产品运维人员关于项目的进展情况。因为这位特定的运维工程师提前参与了开发过程，我们在代码还没有写之前就具备了警示和配套基础设施，并且运维人员在设计时也有更大的发言权。

你认为 DevOps 模式可以用在非 Web，非产品开发导向的公司吗？

**Allspaw**：要问我 DevOps 模式能否用在别的公司？当然。但有没有效果呢？我现在只能说，也许没有，因为大家意识上还是把 IT 视为“开支”而非“盈利”，和请个清洁工倒垃圾一样。

**IT 能做些什么来改变此窘境呢？**

**Allspaw**：IT 普遍被认为是一个“黑盒子”，人们很难看到里面是什么。我们解决此观念的一个方法是把用于网站的监控应用到公司 IT。比如无线。原来无线网络相当复杂。对它的期望是大部分时间工作，但是看不出你做什么也许导致它不工作。因此我们在一个仪表板上制作一幅带有无线接入点信号和链接到此接入点人数

的图表，让每个人随时查看，包括接线员。这样，如果无线出现异常，她就不会只是无奈的抱怨：“什么垃圾网络啊！”这使她现在能够看看是不是无线接入点超负荷了，这也打开了这个“黑盒子”。

还有就是找到便宜简单易实现的方法让组织的非技术部分更加高效和更受公众称赞。Etsy 的打印机神奇般地从来没有缺过打印纸。为什么？应为我们有一张图标显示每个打印机的打印纸的数量，同时还有预警，就像我们在服务器上做的一样。每当我告诉别人这事，甚至在像 Google 一样的公司，他们都会惊道：“哇，这真是一个好主意！”

在你开始在 DevOps 办公环境工作之前，IT 运维人员的工作是怎么样的？
Allspaw：在传统企业中，IT 运维人员职责是可用性和在某种程度上性能。只有高级管理才清楚了解开发的进度情况。 通常开发人员自己会独自写下一堆东西，然后当他们完成了，却发现不能发布，因为运维没有购买相应的服务器或没有预警或其它。这很悲剧，因为运维全蒙在鼓里，这让我们联想：如果运维人员把 XXX 搞定，开发应该可以更加快。

**你用公用云吗？**

**Allspaw**：当然， 我们试着利用那些我们感觉不需要自己去搭建的服务。我们使用 Google Apps、PagerDuty 和一些托管的 B2B 软件即服务（SaaS）用于例如欺诈检测和支付处理。我们利用 Amazon Web Services 合理的放置：我们通常用 S3 存储长尾图片（long-tail photo）， 但用于需要低延时热门图片，我们在我们的数据中心有一层缓存。

但是，认为所有事情都能外包出去的组织终究会“梦想破灭”。我们业内有句话：“自己的可用性属于自己。” Etsy 可以用 Gmail 用于公司邮件，但我们不抱幻想它让我们从我们关注整个事情上解脱。它仅仅是另一个选择。

*(来源：TechTarget 中国　作者：Alex Barrett 译者：黄永晔)*

# 如何使用 IT 新利器——DevOps

有多种不同的技术变革都在深刻影响着企业 IT 部门工作的方式。虚拟化，云计算，软件定义一切，大数据，一切皆服务——这些都迫使 IT 做出改变，并关注新的工作方案，DevOps。

IT 部门的大多数改变都是为了追求速度——对于用户的应用程序的响应速度;改变功能，应用程序和流程的速度，以更好地响应企业的需求。

传统的 IT 工作方式被迫做出改变。 传统的 IT 项目计划遵循一个长期的时间表 – 根据用户定义文件开发合理的项目管理标准，项目计划，审查，变更管理流程等，这一时间计划跨度在 18 个月。

问题在于花费 18 个月来解决问题，并不能解决真正的问题，届时，这个问题可能已经消失了，或者被其他新问题所取代。

DevOps 就是解决方案 - 本质上，通过自动化技术把开发和运营团队紧密联系起来，来支持业务需求变化的速度。

DevOps 是一种将企业的应用开发团队和那些系统运营团队相整合，合作进行任务执行的 IT 工作方案。软件定义的基础设施和云计算要求企业打破开发和运营团队之间的孤立。

传统上，开发团队倾向于作为一个独立团体进行工作，根据不同的定义文件，在其项目上持续工作，直到他们创造出他们所认可的一个完整的应用程序。然后，它会经过一个测试阶段 - 通常还是处于开发名义下 - 在此之后，运营团队开始接管这一应用程序，并在企业内进行运行。

通常来说，新的应用程序会有一些暂时的问题，开发团队会从支持部门那里获得问题名单，会对它们的优先级别进行评估， - “这个问题看起来很有趣?” - 并在程序环境内对他们认为必需的问题进行修改，然后再次提交测试，并再次进行运行验证。

但是，如果有比“暂时的问题”更严重的问题，一切就要复杂很多。在操作环境中有重大问题的应用程序必须中止运行，其包含的数据必须同步到原先的应用程序中，这样员工，合作伙伴和客户可以重新进行工作。这些问题会令开发团队十分费解，因为在他们自己的测试环境中，一切运行良好。

孤立工作会导致挫折和效率低下，因为每个团队都不理解其他团队的局限性或挑战。但更重要的是，IT 和业务都受到了严重影响。

以技术为中心的公司，如 Facebook, Yahoo, Yammer, Amazon, Google 和 VMware 都已经采用了 DevOps 方案。

**DevOps 是如何解决工作孤立问题的**

开发人员必须更密切地与运营团队，以及整个操作环境进行合作。开发团队应该尽可能跟进实时数据集，这样任何问题都可以在早期被发现。

DevOps 方案的关键在于尽可能多的使用自动化技术 - 无论是把开发项目从测试阶段转移到运行阶段的流程，还是如何解决任何运行中产生的问题。

项目必须被分解成更小的模块 - 所需的功能，必须严格进行优先排列，开发团队必须在操作环境中迅速开发出功能系统，来解决最高优先的需求。 “迅速”应该以星期来衡量 – 根据经验，第一次运行应该在 12-18 周之内进行。

这就把需求从单一而庞大的应用程序，转移到由各种资源汇总的混合功能应用程序。数据中心内现有的功能应该与云应用和云服务相结合起来，这样就不需要持续重复开发，使用专业知识和数据集对整体复合应用程序所适用的领域进行增值。

测试阶段应该在运行环境中并行实现，这样更多的性能问题可以尽早被发现，或者在虚拟环境中，由一些真实用户来进行测试。

对于并行实现，通过使用虚拟化比较容易实现。现有数据库的副本可以对比运行 DevOps 代码的虚拟计算机。 然后测试可以针对现有网络负载进行运行，这样其性能可以在代码有效同时进行评估。

确保异常情况可以迅速并有效地获得处理。测试阶段中人员力量是很重要的 –他们可以比计算机更有效的观察人们在使用应用程序时的规律和问题。多一些人员参与测试阶段 – 尽可能多的进行观测和评估。

一旦测试阶段完成了， 然后就应该把应用程序切换到实时数据库， 把用户切换到新的代码。

必须明白，DevOps 方案，就其本质而言，是一个反复的过程。获得 60%，70%或 80%的解决方案，意味着下一步仍然有 40%，30%或 20%的问题需要解决。我们的目标应该始终是解决还剩下问题中最优先的部分，以及在最后代码运行之后出现的新需求。

**如何让你的 DevOps 战略硕果累累**

代码元素由开发团队所完成，尽可能使用自动化，完整的，全面认证的方式，把这些元素从开发团队的测试阶段，过渡到运行环境中，将有助于确保增量变化，并取得改善。

重新运行 ,也可以是自动化的——尽管一个运行良好的 DevOps 项目不应该被要求重新运行 ,再执行，不过这也应该作为计划 B。能够回到一个已知点 - 相当于一个备份 ,恢复时间目标和恢复点目标( RTO/ RPO ) - 应该是任何 DevOps 项目设置的一部分。

反馈也必须是自动化的。 没有必要等待支持部门收集意见和反馈，然后汇报给 DevOps 团队。获取反馈应该作为应用程序的一部分; 要求反馈应该作为流程的一部分。根据反馈对于业务的影响程度采取相应措施- 而不是这一问题从技术层面上听上去多么有趣。

总体而言，DevOps 对于加快 IT 部门确保其满足业务需求而言，是一个很好的方案。 然而，一个无法妥善执行的 DevOps 方案将导致更多的错误，会使用户对于他们获得的体验感到不满。

实施 DevOps 策略并不是一件容易的事 ,因为它要求企业必须接受新的文化。 DevOps 是一种文化，流程，技术和人文。

它在文化和企业上需要重大改变，其中涉及到相当多的行为改变。这意味着要摒弃很多已经深入企业很多年的或显性或隐性的惯例。其中包括告诉那些已经工作多年的员工 ,他们每天惯于做事的方式已经落伍了，这是一个艰难的过程。

只是简单把开发和运营团队放在同一个房间里，并不会带来一个成功的 DevOps 战略 - 每个团队都必须了解和认识到在云计算和软件定义时代进行合作工作的重要性。

自动化技术，激励开发和运营团队合作工作，分配时间用于培训员工，根据创建即可执行原则来开发新的企业架构标准，将 IT 与业务目标相结合，所有这些步骤都将确保 DevOps 战略不会受到损害。确保一些细微的方面都涵盖到，将有助于 DevOps 更好发挥作用。

# 如何辨识 DevOps 是否合适

如果让 Simon Maple 说出一件他所了解的事情，那就是 DevOps。但是，要想真正深入了解某件事情是要付出代价的。Maple 是 ZeroTurnaround 的技术专员，每当发展趋势变得模糊不清时，他都会感觉像是头部正在猛烈地撞击石墙。因为这种模糊的情况经常发生，因此 Maple 已经患上了头痛病。最糟糕的是，人们甚至不知道自己所说的词汇究竟是什么意思。“就我个人而言，最大的烦恼是 DevOps。每个人都想要跟随潮流，因此他们给那些从事 DevOps 工作的研究人员打电话。”那么，DevOps 究竟指什么呢？

Maple 认为，DevOps 包括三部分。首先，DevOps 可以试图找到一种让开发人员快速研究出新产品的方法。这就意味着，在要将开发过程和运行过程作为一个整体工作流程，而不是将其分为两个独立的共组流程。有人更愿意站在背后，绘制更大的图景。

其次，DevOps 是一种反馈循环，使整个周期越短越好。一种较短的反馈循环意味着，要尽快发现问题，并将相关信息尽快反馈给项目开发团队。将事情堆在一起，并告诉别人稍后将处理此事，这种做法让人难以接受。这就是要强调反馈重要性的原因之一，因此不能忽略反馈问题。开发团队必须监控反馈循环，成员必须对反馈信息进行实时跟踪，并处理反馈中提出的问题。

（检验 HCL 科技公司的 Employees First 是否发展到某种极端，这是一个非常有趣的检验。该公司实际上对过去反馈路径进行了延伸，这种创新方式涉及到一线员工的反馈信息，让开发人员直接负责销售和客户服务，开发人员可以了解他们的解决方案是如何处理问题的。）

最后，公司必须面对这样的现实：开发人员和运营团队需要了解彼此，已及他们所分配的角色。Simon 认为：“如果你没有掌握该领域及其周围领域的 DevOps，那么就不能实现第一步和第二步。”实际上，进入第三步既是最基础的环节，也是最困难的环节，因为这需要一个全面的文化转变。

**DevOps 指的不是……**

Maple 清楚地解释说 DevOps 并不是一个标题。它与在 LinkedIn 自称是 DevOps 工程师的人数没有关系。与 DevOps 招聘多少人也没有关系，关键在于是否能找到符合技能要求的人员。DevOps 根本不是一个人就能完成的工作。“它是构建团队的一种方式，是引导团队工作的一种方式。”

有几种工具可以保证持续整合以及发布管理（例如 LiveRebel 的解决方案），这样就可以实现上述所说的三个步骤。但是，仅仅依靠工具是不能完成 DevOps 的。 “这是一种习惯。公司必须知道 DevOps 是买不来的。要以团队、流程和生命周期的形式对 DevOps 进行研究和投资。” 只有当以上所说的重要部分都被购买后，或者此部分逐渐成为公司改革的一部分时，这种流程才会实现。

**DevOps 工具**

假设一家公司资源尝试了 DevOps，LiveRebel 就是一个改革成功的例子。这些工具使得开发和运行团队工作的更顺利，提高了工作速度，降低了人工失败率。简单来说，该产品实现了应用程序部署环节的自动化编排。在这个流程中编排是非常重要的，通过点击按钮让整个工作变得非常容易，但是如果你在错误的时间，用错误的方式做了一件错事，那么这种简单易操作的方法实际上就会引起一个大灾难。

为了避免这些问题的发生，LiveRebel 在解决方案对配置、数据库和应用程序代码进行了约束，目的是为了实现以一种完美的升级体验。Dev 团队为了达到更高的水平而选择了特定环境和应用程序，并且这种升级体验是自动协调和实施的。要定时地避免现有 HTTP 会话的消失，因此用户就不会遇到无法访问的情况了。如果任何一个阶段出现了异常，LiveRebel 可以自动回到异常发生之前的工作状态。你可以直接离开，当你回来时一切问题都解决了——要么是进入一个新状态，要么是回到以前良好的状态下。

为什么说在运行阶段增加开发流程的可见性是一件好事呢？当开发人员完全掌握了自动化流程，理解了早期部署阶段，他们就开始真正理解了选择的内涵。正如 Maple 所提出的：“开发阶段的选择是为生产阶段服务的。” 深入地理解这些相互关联关系后，Dev 可以做出更明智的选择，在他们工作中为 Ops 团队提供更多的支持。这种良性反馈可以使整个工作变得更好。

*(来源：TechTarget 中国　作者：Jason Tee　译者：邹雅玲)*

# DevOps 成功的两个关键

将软件开发和 IT 运维团队整合到单一的 DevOps 组织可以带来更强大的软件开发项目交付能力，但两类团队的文化差异、以及缺乏有效的工具都会阻碍 DevOps 的成功。

越来越多的公司开始向 DevOps 模式转型，希望能更迅速地将越来越多的软件更新和修补程序交付到用户手上，而且能实现比传统模式更短的软件更新周期。

在持续的竞争压力下，能够更快发布软件产品的公司显然会拥有更强的竞争优势。

Gartner 研究总监 Colin Fletcher 在 2014 Gartner IT 基础架构和运营管理峰会上表示，“它（DevOps）将决定你进军新市场或者推出新产品的速度”。Fletcher 认为，新兴市场的竞争压力、适应新的计算平台，为产品增加新功能、提升能力，这些都是转向 DevOps 的理由。

传统的软件开发是一个人工过程，每一代产品从源代码到测试到发布的过程都会跨越组织内部多个相对分离的领域。虽然让开发人员、测试人员和其它岗位人员融合无间，消除相互之间的合作壁垒是 DevOps 的终极目标，但每个公司都会用不同的办法将 DevOps 付诸实现。

Gartner 公司资深 VM 分析师 Ronni Colville 表示，开发人员和运营人员对于软件发布管理的看法完全不同。在软件开发人员眼里，软件是代码、功能和性能。然而，运营人员眼里的软件是致力于将代码数据传送到目的地的一个流程或者行为。如果能整合两方的看法，就能大幅精简流程，减少理解上的误差。

1.应对 DevOps 带来的文化冲击

平稳的文化过渡是让 DevOps 获得长期成功应用和增强发布软件产品的综合能力的关键。第一步是，明确 DevOps 的定义，调动开发和运营部门之间的协作，鼓励运营人员采纳软件开发方法，并利用云计算基础设施来完成真实的测试和代码部署。

在软件开发、测试、质量保证（QA）、集成、预生产和生产部署等方面的任何旧小团队必须打散，因为每个小团队都可能拖延开发周期并且带来不可预料的问题。

以上策略能更好地整合开发和运营人员，通过整合团队成员来产生效益。例如，在讨论运营解决方案或扰乱事后评估报告时应该邀请开发人员加入。相反地，应该邀请运营人员列席开发人员规划会议。让交叉组合的工作模式成为制度 ,可以让团队之间合作融洽 ,消除沟通不畅导致的延误或疏忽 ,使 DevOps 的推进更加有效。

这种文化上的改革并不容易。它需要公司提供统一的考核标准，以相同的形式衡量开发人员和运维人员的业绩。培养一种团队精神 ,让大家一起向一个共同的目标努力 ,而不再只是为了从前各自的狭隘的小团体目标。在这里有时可以运用岗位轮换或者知识共享的方法。Colville 鼓励大家勇于尝试——尝试用创造性的新方法处理问题，谨慎应对风险，从失败中积累经验。

在 Bob Jones 大学的案例里，开发和运营团队之间的文化差异成为巨大的挑战。

“从前我们的客户都没有好脸色”，Bob Jones 大学 IT 运营总监 Terry Worley 表示，“我不得不拿着鞭子驱赶运营和开发人员，迫使他们在工作上认真地合作。我深有体会，文化差异是拦路虎，我要么坐着等死，要么就该去杀了它！”

Worley 已经证明了文化变革的好处，他指出，在采用了优化后的 DevOps 模式完成两个项目之后，开发人员们就再也不想回到从前的传统软件开发模式了。

2.DevOps 需要齐全的工具箱

想要超越文化的影响，组织还必须依靠各种 DevOps 工具。例如，开发人员编写代码需要工具、QA 测试人员需要用工具完成新版软件的部署，环境准备、将新代码在测试系统和生产系统之间迁移也必须用到云资源调度工具。Fletcher 表示，工具本身都不是问题，重要的是能够让各种工具互相配合，在软件的生命周期内提供支持。

当前在应用程序发布自动化工具市场已经存在众多的供应商。运营工具方面的供应商有 BMC Software, CA Technologies Inc. 和 XebiaLabs Inc.等公司。软件开发工具方面的供应商包括 IBM，Electric Cloud Inc. 和 Serena Software Inc。

关于开发人员工作流程、架构设计和软件发布工具方面的专业供应商也在不断涌现，这些供应商包括 Atlassian, CollabNet Inc., Rally Software, ThoughtWorks Inc. ，OpenMake Software Inc.等公司。在评估这些新供应商时，应明智地预计到这些公司随时可能会被并购，其产品可用性和未来发展也会因此受影响——请记得在选择新的软件发布自动化产品时多留个心眼。

*(来源：TechTarget 中国　作者：Stephen J. Bigelow 译者：徐继军)*

# 利用 DevOps 工具搭建开发与运维之间的桥梁

运维与开发在公司内互相角力已经成为每日必修课。选择适合的 DevOps 工具，有机会让开发与运维双剑合璧。

DevOps 初期对开发非常有优势，但随着版本更新与供应商的合作关系，运维逐渐成为重要角色。

有很多方法可以让开发者像 IT 运维经理那样思考——把他们加入凌晨 2 点的值班名单是个不错的方法——但为组织内扮演不同职责的两类人提供信息共享工具，可以在整个应用程序生命周期内让所有人都获得一致的信息。

若应用程序没有按照设计的那样工作，将是一场灾难，PJM 互联电网公司 IT 运营中心经理 Colin Brisson 说。该公司位于美国东北部，经营着一家日前市场。如果应用程序无法在同一时间关闭掉每位用户的买盘，那么市场是不公平的。

“你需要了解这些应用程序何时会出现问题，” Brisson 说，只是这样做是无法发现开发中所有问题的。当留个主要程序在同一个基础设施上运行后——将产生复杂的相互作用，他说。

适合的工具

共享某个 DevOps 工具并不像听起来那么容易——“发布”、“部署”与“自动化”对双方皆有不同含义。

“开发与运营之间存在着一个完整的语言障碍 ,这里指的不是方言 ,”Gartner 公司的分析师 Gonni Colvile 在 2014 年 IT 基础架构与运营管理峰会期间说。

工作负载自动化与管理简化，是应对业务快速变化必不可少业务应用程序，所以选择适合的 DevOps 工具对应用程序性能的影响与底层硬件是一样的。

然而 ,这是一个困难的决定。部分原因是 DevOps 工具集功能无法一刀切 ,没法双击鼠标就安装好 DevOps。“要明白每个工具在特定场景下，能为应用程序生命周期的每个阶段提供何种功能，” Colville 说。需要找到一款可以与开发、测试再到生产相适应的工具。

应用程序生命周期

在应用程序生命周期的初期，开发者占据统治地位。然后，重点将转向上线前的部署任务，接着再转移到监控系统，以监控运行的应用程序。如果应用程序的价值持续增长，生产效率率——应用程序如何工作，其可以怎样帮助公司增长或怎样才是最好的服务架构将正为之后的焦点。

对于易扩展和更新的应用程序，开发需要考虑自动化。Chef，一款资历悠久的 DevOps 工作流管理工具，与 Docker 集装箱容器平台集成，通过镜像部署的轻量级虚拟化方式独立运行应用程序。容器集装箱主机能够提供共享内核的轻量级虚拟机，满足业务孤立运行。

Docker 集装箱式部署替代传统运用程序部署采用安装包的方法，可以快速将应用程序更新同开发环境部署到正式环境中。

Chef 可以基于服现有的服务器模版创建 Docker 镜像。这些镜像存在一个库中，类似另外一款开源工具 Git。DevOps 团队可以选择何时何地进行部署，并且可以进行双向配置变更，例如数据库服务器的地址并没有包含在初始镜像中。

团队可以依赖的 DevOps 工具，除了 Chef 还有其他竞争者，Puppet 可以关联生产环境管理人员与代码之间的交流沟通。

另一家厂商，JumpCloud，提供自动化平台，可以让 IT 团队使用最擅长的脚本语言进行操作。这样开放的应用程序对运维人员来说，只需要掌握脚本语言，而不需要开发人员的专业编码知识，JumpCloud 高管说。

基于日志的 DevOps 工具同样可以让运营与开发链更紧密衔接。Splunk 和 Logentries 这样的工具，以及其他各种开源软件，可以跟踪日志时间并分析数据以帮助分析和故障诊断。这样能够为不同的团队提供一个共同的预约，方便大家在代码问题上达成共识。团队也可以通过其他来源添加额外数据，包括服务器或应用程序性能监控工具。

整体目标是通过应用程序层级或更低层级的配置变更，让应用程序和运营团队看到代码更新对应用程序效率的影响，或协助排查故障问题，这些都是根据实际情况而定。

Logentries 是一款可以添加注释到 DevOps 工具仪表板上的工具，IT 团队的不同人员，可以查看日志事件中上下文信息和知识。

“记录事件描述与解决方案，这样的情报可以节约其他人四小时的工作时间，”公司首席科学家 Trevor Parsons 说

由于不同产品能够在开发、质量保证、测试、预生产和生产阶段提供更好的的功能，投资 DevOps 工具组合是十分必要的。你还可能需要一套工具集，因为不同应用程序有着不同的生命周期，而且基于云计算的应用程序可能需要不同的 DevOps 工具，与托管的物理或虚拟机有所区别。

*(来源：TechTarget 中国　作者：杨旭)*

# 五大 DevOps 最佳实践实现安全、可伸缩和性能

随着 TheServerSide 在如何在 DevOps 相关的时间和金钱方面进行最好的投入的调查的继续，为了把一些业界专家建议的各种有助于打造出可伸缩、安全及高性能的部署的最佳实践关联起来，现在的关注点已经转移了。以下就是业界最重要的、专家亲自试过的 5 个技巧和最佳实践：

1、警惕总体安全风险—Tufin 联合创始人兼 CTO Reuven Harrison 强调了网络不断增长的复杂性。他说，虚拟化、云、BYOD 以及软件定义网络（SDN）等新兴技术不断得到采用意味着网络变得越来越复杂，愈发的异构化，安全风险也是如此。“随着 SDN 和网络虚拟化继续走向成熟，以任何程度的效率和安全性来管理这些网络的唯一办法是把关键的管理功能自动化，”他说：“这就是 DevOps 的前提。但是 DevOps 必须把安全作为一个关键组件包括进来，因为没有它的话，SDN 和虚拟化等技术所引入的网络变化的体量和节奏会把该环境下的 IT 风险的水平提高到天那么高。”这其中的巨大挑战是迄今为止，安全被视为是事后想法，而安全组织又被认为是企业的抑制因子，只会告诉企业什么做不了而不是如何安全地做事情。这是一个文化问题，需要安全、开发者以及运营团队培育出此前未有过的一定水平的信任和协作。做到这一点的唯一办法是逐步地、带着警惕地去做。

2、观察安全风险变化—ASG Software Solutions 负责产品管理与云的副总裁 Torsten Volk 说，把 DevOps 看作一种可将开发者和 IT 运营引向更快更高效的部署、运营及升级应用的协作理念和流程很重要。“在 DevOps 之前每一次新的发布都会引发同样一组的安全考虑，”他说：“然而，一旦新发布以一种高得多的节奏进行时，安全也不得不成为一个不能间断的关注点。”就这一点而言，DevOps 工具可以通过主动保证基础设施和软件组件配置的一致性来提供帮助。这些工具甚至还可以通过经常验证安全最佳实践的恰当运用，来自动化修复安全隐患并自动采取对策。尽管后一种场景听起来似乎很先进，但端点才是每一个 DevOps 团队都渴望到达的。

3、注意可伸缩性—根据 PythianDevOps 团队主管 Aaron M. Lee 的说法，往往有两类伸缩性是 DevOps 工程师需要处理的：应用和组织。“App 的可伸缩性真的就是开发和运营成功交付特定水平的并发性的系统需要花多长时间花多少钱的问题；即达到经过一段时期之后可匹配或超过用户需求的水平，” Lee 说。“评估这些问题的答案对于许多公司来说是一个至关重要的成功要素，而做到这一点的能力往往不为人知” Lee 说可伸缩性是每一个人的问题。企业和技术的人必须在功能、推向市场的时间、成本以及风险承受能力等方面做出权衡。你需要有合适的衡量目标，包括特定模式下的那些端点上有多少用户，有多少并发请求。

4、争取实现易用—DevOps 就是自动化和可重复性。Andy Piper 博士是总部位于伦敦的 Push Technology 的 CTO，他说这需要有可配置的虚拟环境，而且要有很多。“要想可伸缩就得自动化，”他说：“因此，要确保你在使用 Puppet 及 Chef 这样的工具来对 VM 的建设和配置进行自动化。类似地，要确保你有能力对其进行备份，如在本地进行的备份，这种方式要想实现动态伸缩会更棘手，或者在云端备份，如果你的产品必须如此的话。”到最后，令产品易于安装、配置和运行会令整个 DevOps 过程容易得多。

5、管理网关—InfoZen 负责 Cloud Practice 的项目管理总监 Susan Sparks 说，尽管新的目标是在开发和运营团队之间建设最好的文化，但为了确保产品环境保持稳定，在这两个职能之间保留一些网关仍然是好的。“我们的团队建构方式就是为了把运营人员纳入到开发讨论和日常的敏捷开发当中，以便运营团队理解未来的各种发布中会出现哪些变化，”她说：“运营团队负有维持产品运营稳定性的责任。我们发现这一方法对我们很有效。我们建议测试和运营都采用自动化。我们的集成测试允许我们在到达生产前就能找到问题，我们的运营自动化实现了成本效率和更优质的运营。通过自动化，更少的人接触生产环境，因此可显著减少人为错误。这还帮助改善安全形势，因为更少的人需要接触产品环境。”

“DevOps 并不难。难的是在组织没有采取 DevOps 方法来进行集成、开发和部署时引起的挑战的应对，”TheServerSide 的软件架构师兼编辑 Cameron McKenzie 说，这种观点很难质疑。通过采用 DevOps 的办法，并注意这 5 个步骤，那么一个成功的 DevOps 环境只是一两次实施罢了。

*(来源：TechTarget 中国　作者：Jim Romeo　译者：boxi)*

# 在大型机上使用 agile DevOps 的四点好处

对于历史悠久的大型机来说，Agile DevOps 可能是一个新的概念，但是大型机却可以借助于 Agile DevOps 来提高生产力和可靠性。

敏捷开发——将应用程序生命周期中的发行前和发行后状态联系在一起——通常被用做传统大型机的工作流程。但是，相比于大规模 Linux 和 Windows 服务器环境，在大型机上使用 agile DevOps 可以发挥更大的作用。

在大型机上使用 agile DevOps 有四个优势：加快应用程序交付和升级的速度；在运行过程中不断改进的软件可以更好的适应应用程序短期和长期发展；增强 IT 组织协作能力——在大型机团队内部和整合环境当中——可以在减少故障时间的同时提供更高质量的应用程序；提高大型机管理员工作效率。

**1. 加快交付速度**

对于 IT 领域中的大型机来说，交付这个词具有多个含义。开发人员在交付代码之前需要保证其通过一系列常规测试，之后将其交给运行专家。而 IT 管理员则认为交付和上线是同一个意思：就是真正地执行所有代码。敏捷开发只能够加速代码交付之前的那部分任务；而 DevOps 填补了之前的空白部分，通过建立一个完整的生命活动周期，关注如何更好地获取 IT 运维团队的反馈。Agile DevOps 将敏捷原则应用于管理领域，使得开发人员和管理员可以进行毫无障碍的沟通。

大型机上的管理流程通常都是已经建立好的，但是很多开发人员具有更为丰富的 Linux 和 Windows 专业知识。因此，在开发者的眼中，大型机上 agile DevOps 还有很多不足，导致代码交接容易出现延迟。同样的情况也会出现在重大 bug 的修复过程中。

通过减少测试环境和实际部署环境之间的差异，并且缩短升级和修复 bug 所需的频率和时间，Agile DevOps 实现了“交付速度的提升”。

**2.运行时软件优化**

在大型机运行时软件开发当中，agile DevOps 可以在两个方面提升知识水平和程序质量。首先，对于许多较新的、面向对象的操作系统，比如 Linux，很有可能不关机而一直保持运行状态。因此，它们容易出现问题，比如错误的垃圾回收机制以及不能正确重新组织关系型数据存储。

Agile DevOps 借鉴了大型机管理员积累的经验来重新认识软件平台类型，以及可能引起这些类型问题的开发和/或测试流程。开发团队可以使用嵌入式模式保护代码来部署代码库和测试环境。

第二个领域是认识到测试工具之间的差距。现在很少的——如果存在——综合测试套件能够使用相同的技术同时处理非大型机和大型机的复杂活动（网络、代码层和数据库）。大型机通常会被人所忽视。

agile DevOps 的目标是在测试环境中，或者以代码的形式嵌入到应用程序自身当中以获取大型机复杂性的现有知识，而不是希望大型机管理员发现问题所在。这并不仅可以使得开发人员和测试人员的工作更加轻松，同样可以简化管理员的工作。

简化开发人员和管理员的工作可以实现“快速交付”并且开发出更为灵活的软件，在升级过程中不需要管理员进行干预。

**3.更好的协作以及更高的质量**

大型机管理员和开发人员在许多问题的理解上通常会存在较大差异。两者可能永远都无法实现目标统一，所以管理员和开发者的目标在于使用“敏捷性”技术创建一个最小化摩擦的通用架构。

通过设计并有效实施 agile DevOps 流程，CIO 应该可以看到代码质量方面巨大改进，以及对于不经常出现问题的应用程序更为精简的管理——哪怕 CIO 并不关心交付速度或者软件在运行过程中的不断改进。

**4.提高大型机管理员工作效率**

通常我们都会说大型机（或者非大型机）管理员是救火队员。对于大型机管理来说，许多“救火行动”都涉及了新的 Linux 环境，以及与跨网络外部环境的相互影响。

Agile DevOps 可以改善这种大型机管理模式，从而提高大型机管理员的工作效率。首先，通过实现标准配置和 Linux 相关任务的自动化，Agile DevOps 可以保证管理员拥有更多时间来“救火”。通过确保解决方案是长期有效和高质量的来减少对于处理紧急情况的处理需求。此外，让管理员也参与敏捷开发流程，和开发团队进行沟通，当开发团队拥有了一个能够快速定位问题并且修复运行时问题的测试工具或者代码库之后，agile DevOps 就可以减少管理员修复 bug 以及与开发部门协调所花费的时间。

**发展前景**

虽然 IT 部门仍然处在 agile DevOps 部署的初级阶段，但是已经产生了一些最佳实践：

寻找并获取适用于大型机管理员的大型机和不同环境自动化工具。

在大型机管理员和开发人员之间创建一个非正式的或者基于软件的协作流程。

关注于不同环境之间，而不只是大型机专用应用程序的生命周期管理。运行速度变慢和崩溃通常会涉及环境之间的相互影响，而不只是大型机自身（比如加快交付速度、改善运行时软件优化、增强 IT 间组织协作能力和提高管理员工作效率）。

*(来源：TechTarget 中国　作者：Wayne Kernochan　译者：王学强)*

# 遗留现代化：为什么使用敏捷 DevOps 方法

为了不升级应用，采用敏捷 DevOps 的方法进行遗留现代化可确保资源得到最好的利用。敏捷 DevOps 方法是有益的，因为它可确保现代化应用如预期一样高效运行。

遗留现代化有三种主要敏捷 DevOps 方法，都能暴露出现代化应用的运维问题。每一种方法还可以显示出哪些预测设想没有实现，即使在运维问题没有出现时。

**新平台上重新托管遗留系统**

新平台上重新托管遗留系统，而不必须对系统做重要的更改，这看起来似乎是一个非常篇章的选择。这一方法包括内部运行良好的遗留系统，以及托管在云中没有任何问题的系统。

时常的更新和不断的修补很可能会增加依赖的复杂性。

当重新托管遗留系统不可扩展，或资源使用不还没旧系统高效时，这一方法就不太理想了。在新平台上运行可能需要对具体资源使用庇的遗留代码做出重大更改。

**迁移遗留系统到关系型数据库系统**

虽然把旧的应用迁移到关系型数据库中看起来是一个不错的遗留现代化技术。敏捷 DevOps 方法可能揭示了查询关系型数据库的运维问题。这类问题包括慢查询响应、网络延迟、托管数据库服务器超载、资源消耗过多，以及不正确的数据划分。

为了最小化运营问题，从而达到查询优化，从数据库优化的增量迭代开始敏捷 DevOps 方法。这包括表格标准化和支持高效查询的设计指标。

**分解遗留系统成服务相关组件**

如果你想进行遗留系统分解，那么在开发敏捷 DevOps 方法时，就要有现成的业务流程改善计划。这一计划帮助决定提取哪个遗留系统服务相关的组件来现代化应用，如软件即服务。

为了使用解开组件依赖变得更容易，从增量迭代分解开始。经常更新和修补，依赖的复杂性很可以会增加。

一旦成功解开依赖，服务相关组织就可以接受，可拒绝，或结合起来。可接受的组件存储在库中，以供其它的遗留现代化项目参考。

也可以不接受服务组件，如果他们不再使用、过期了，或会导致反应慢的话。可接受的服务组件并不意味着他们之间的依赖也可以接受。

可接受的依赖可能需要重新调整可接受服务组件。使用敏捷 DevOps 进行组件重构增量迭代。迭代过程中，结合依赖性会产生较少的服务组件。运维的 DevOps 迭代将会揭示操作可接受服务组件应用的问题。

*(来源：TechTarget 中国　作者：Judy Myerson　译者：蒋红冰)*

# 三种实现敏捷 DevOps 的方式

DevOps（开发运营）运动在修补残缺的部署流程方面取得很大进步。从表面上看，为开发者提供的解决方案是更多地站在运营者的角度去思考，反之亦然。然而，这种表象级的想法也只能到这里了。需要具体的、可操作的步骤去克服技术方面的欠债，向 Amazon（顺便说一下，今年 5 月该公司的平均部署时间间隔为 11.6 秒）之类的应用开发组织的效能靠近。

在 Agile 2013 上，演讲者 Gene Kim 展示了一个在应用开发和部署流程中做出可操作改变的可行框架。Kim 是 Tripwire 的创始人，也是该公司 2010 年夏以前的 CEO。现在，3 年之后，Kim 出版了一本小说，名字叫做《凤凰项目》，书中解释了如何给死板的开发组织带来灵活性。这本小说生动描述了 Kim 对 DevOps 现象 14 年的研究当中学到的 3 个主要概念。

根据 Kim 的说法，相对于没有开始 DevOps 计划的组织，那些参与 DevOps 计划达到或超过 12 个月的组织展现出了显著的效能提升。Kim 说高绩效的 DevOps 团队会更加敏捷（部署频率高 30 倍，周期时间快 8000 倍）且更可靠（成功变更次数最多可增加 1 倍，响应时间仅是平均响应时间的一小部分）。

组建一支高绩效的 DevOps 团队需要 3 个要素，Kim 说。在其最新的小说中，Kim 将每一个要素阐述为“3 种方法”之一，由他的神秘大师展现给领导者。这三种方法可归结为过程流、反馈及持续学习。

**方法 1：流**

在构建过程流中，Kim 概括了 5 项主要规则。首先，理解工作流。如果缺乏理解，任何变更都会有随机效应。第 2 和第 3，永远都要增加流，永远不要把缺陷传给下游。如果你把生命周期管理流程想象为一条河流，就很容易看清为了修正而逆流回溯项目有多困难，尤其是在过程迅速流转的时候。

第 4，永远都不要让本地优化导致到全局的退化。在内部开发组织各自为政的大型组织当中这是很棘手的。每一个团队都希望自己的部分成为演出的明星。然而，让一切保持平衡对于整个系统来说好处要大得多。最后第 5 点，对整个系统要有深刻的理解。这一点跟第 1 点相呼应。你必须理解每一部分的细微差别，知道它们是如何相互适应的，这样才能维持一切平衡有序。

从这些规则当中，Kim 发现了一个改进开发时间的重大方法。其诀窍是找出流程瓶颈然后打破它。Kim 引用了 Eliyahu Goldratt《目标（The Goal）》里面自己喜欢的台词，说的是“任何地方的变化都能带来改进，但是在瓶颈之处那只是一种幻觉。”也就是说，整个部署流程链的速度取决于最慢的环节。

Kim 说大型软件组织有 6 个领域是最有可能成为瓶颈的：环境创建、代码部署、设置以及运行测试、过度紧密的架构、开发及产品经理。Kim 说，根据 2012 年 Puppet Labs 的一项调查，组织当中两个最常见的展现出一致速度和品质的链环是基础设施版本控制( 89%采用版本控制 )及自动部署流程( 82%对部署进行自动化 )。

**方法 2：持续反馈**

第 2 种办法是反馈，也有 4 条重要规则。首先，理解并响应所有客户的需求，包括内部和外部的。其次，缩短并放大所有的反馈环。Kim 说这条规则效仿流量丰田精益生产线，后者让任何一名员工在看到有缺陷的情况下均可停止生产线。第 3，从源头建立品质。意思是说积极地将品质注入到一切东西上。最后是第 4 点，在需要的地方创建并嵌入知识。

Kim 建议把脆弱的服务返还给开发者，让他们对服务的稳定性负责。为了说明为什么这么做有效，Kim 引用了 BrowserMob CEO Patrick Lightbody 的话，他发现“凌晨 2 点叫醒开发者时缺陷被修复的时间就会前所未有的快，”当然，他们并不是为了激怒开发者而在半夜叫醒他们。除非代码出错开发者才要凌晨 2 点起床。这当然会激发开发者写出高质量的代码。“这可以归结为有难同当。”Kim 解释说。

Kim 还建议为各种指标设立简单的自动化监视器。比方说，开发者也许可以建立简单的增量计数器来跟踪成功及失败登录尝试。这一监控可为团队提供验证服务运作情况的反馈。甚至还可以找出恶意活动。

**方法 3：持续学习**

Kim 的第 3 种方法是培育一种鼓励实验（哪怕要冒风险）、奖励成功、从失败中学习以及认识到重复是精通的先决条件的文化。Kim 说成功组织形成了一种靠着能令自己在危险中生存的习惯不断向危险区推进的文化。

Kim 援引了 Netflix 云架构师 Adrian Cockcroft 的说法。“痛苦的事情做得越多你就能让它带来的痛苦感觉越少，”Cockcroft 建议说。他说自己的开发者不会回避痛苦的任务，因为他们知道这可以让大家的部署流程流畅的多。Cockcroft 的方法在 2011 年 4 月得到验证，当时 EC2 中断导致 Reddit 和 Quora 服务停止，但是并未严重影响到 Netflix—即便 Netflix 也一样跟 EC2 捆绑在一起。

Cockcroft 的团队开发了一种名为 Chaos Monkey 的测试工具。Chaos Monkey 的目的是随机禁用一些服务以便测试其他所有服务的独立性。当然这是个多少有点极端的例子，但是的确说明了尽早失败及经常失败的价值。这么做让开发组织成长，一旦应用进入生产阶段，可处置任何有可能出问题的情况。

为了坚持这 3 种方法，Kim 提出了 3 点建议。首先，处处都要强制要求一致性—代码本身如此，环境和配置也一样。其次，在代码中使用 Assert（断言）语句找出配置错误并执行最佳实践。最后，采用自动化持续集成测试的同时也要利用静态代码分析。

*(来源：TechTarget 中国　作者：James A. Denman　译者：boxi)*

本期电子书由 TechTarget 出品