03 | 持续交付和DevOps是一对好基友

2018-07-10 王潇俊

03 | 持续交付和DevOps是一对好基友
朗读人：王潇俊　10′03″ | 4.61M

现在很多人都在困惑持续交付和 DevOps 到底是什么关系，有什么区别，或许你也感觉傻傻分不清楚。那么今天，我就来和你聊聊持续交付和 DevOps，以及它们到底是什么关系。

持续交付是什么？

我在专栏的第一篇文章中，已经跟你很详细地分享了持续交付是什么，为了加深你的印象，并与 DevOps 形成对比，我在这里再从另外一个角度给你总结一下：

> 持续交付是，提升软件交付速率的一套工程方法和一系列最佳实践的集合。

它的关注点可以概括为：持续集成构建、测试自动化和部署流水线。

那么，DevOps 又是什么呢？其实一直以来，学术界、工业界都对 DevOps 没有明确的定义，所以造成了大家对它的看法也是众说纷纭，也难免片面。

在我给出我个人的认识之前，我先给你讲讲 DevOps 是怎么被发明的吧。

DevOps 的诞生

DevOps 的故事，要从一个叫帕特里克 · 德博伊斯（Patrick Debois）的 IT 咨询师讲起。2007 年，帕特里克参与了一个政府下属部门的大型数据中心迁移的项目。

在这个项目中，帕特里克发现开发团队（Dev）和运维团队（Ops）的工作方式和思维方式有巨大的差异：

- Dev 的工作是，为软件增加新功能和修复缺陷，这要通过频繁的变更来达到；
- Ops 的工作是，保证系统的高稳定性和高性能，这代表着变更越少越不容易出错。

因此，Dev 和 Ops 长久以来，都处于对立和矛盾的状态。

2009 年 6 月 23 日，Flickr 公司的运维部门经理约翰 · 阿斯帕尔瓦（John Allspaw）和工程师保罗 · 哈蒙德在 Velocity 大会上做了一个轰动世界的演讲：《每天部署 10 次以上：Flickr 公司的 Dev 与 Ops 的合作》（10+ Deploys Per Day: Dev and Ops Cooperation at Flickr）。

这个演讲中提出了 DevOps 的核心观点：Dev 和 Ops 的矛盾可以通过技术升级和文化构建来解决，这标志着 DevOps 的诞生。

帕特里克也在网上看到了这个演讲，并且十分兴奋，因为这就是长久以来他所想解决的问题。于是，他开始筹备自己的 Velocity 大会。

2009 年 10 月，帕特里克的 Velocity 大会在比利时顺利召开，他把会议命名为 DevOpsDays。他本来想用 DOD 作为 DevOpsDays 的缩写，以提醒自己“死在交付上”（Dead On Delivery），但不知什么原因，他最后没有这么做。

这届大会出人意料的成功，许多开发工程师和运维工程师参加了这次大会，甚至还有各种 IT 管理人员参加。人们开始在 Twitter 上大量讨论 DevOpsDays 的内容。

由于 Twitter 对内容长度的限制是 140 个字符，所以大家在 Twitter 上讨论时去掉了“Days”，只保留了 “DevOps”。于是， DevOps 这个名称正式诞生。

持续交付的姗姗来迟

在 DevOps 的这段编年史里，持续交付又在哪里呢？

2006 年，杰斯 · 亨布尔（Jez Humble），克里斯 · 里德（Chris Read）和丹 · 诺斯（Dan North）在 Agile 大会上发表了一篇名为《部署生产线》（Deployment Production Line）的文章，这也是第一篇描述持续部署核心内容的会议文章。

在后面的三年里，又有一系列“持续部署”的文章被发表。2009 年，这一些系列的文章被编成为了一本叫作《持续交付：发布可靠软件的系统方法》的书，这一年也正是帕特里克举办 DevOpsDays 的那一年。

2010 年，《持续交付：发布可靠软件的系统方法》的作者之一杰斯参加了第二届的 DevOpsDays，并做了 关于“持续交付”的演讲，在这一年“DevOps”与“持续交付”终于有了交集。

从本质上说，帕特里克最初遇到的问题，在《持续交付：发布可靠软件的系统方法》一书中找到了最佳实践。如果这本书可以早两年问世，或许今天就不会有 DevOps 了。

然而，DevOps 的概念一直在向外延伸，包括了：运营和用户，以及快速、良好、及时的反馈机制等内容，已经超出了“持续交付”本身所涵盖的范畴。而持续交付则一直被视作 DevOps 的核心实践之一被广泛谈及。

这么看来，持续交付真是打了一个大盹儿。

认识 DevOps

DevOps 这几年一直在不断地演化，那么它到底是什么呢？

目前，人们对 DevOps 的看法，可以大致概括为 DevOps 是一组技术，一个职能、一种文化，和一种组织架构四种。

第一，DevOps 是一组技术，包括：自动化运维、持续交付、高频部署、Docker 等内容。

但是，如果你仅仅将 DevOps 认为是一组技术的集合的话，就有一些片面。任何技术都是为了解决某些问题而被创造出来的。比如 Docker，就是为了解决 DevOps 所提倡的“基础设施即代码”这个问题，而被创造出来的。

从这个角度来看的话，DevOps 的范畴应该远远大于一组技术了。

其实，DevOps 是一组技术这个观点，还是只站在了工程师角度去思考问题而得出的结论。虽然“DevOps”中“Dev”和“Ops”这两个角色都是工程师，但是其本质还是希望跳出工程师的惯性思维来看待问题。

第二，DevOps 是一个职能，这也是我在各个场合最常听到的观点。

你的公司有没有或者正准备成立一个叫作 DevOps 的部门，并将这个部门的工程师命名为 DevOps 工程师？至少在各大招聘网站上，是随处可见这样的职位，而招聘要求往往就是：会 Ops 技能的 Dev，或者会 Dev 技能的 Ops；或者干脆叫全栈工程师。

“DevOps 是一个职能”这个观点，源于设施的日趋完善，云服务的流行，以及各类开源工具的广泛使用，使传统 Ops 的工作重心发生了变化，使企业产生了不再需要 Ops 的错觉。

但这个观点也是错误的，原因就是忽略了 Dev 与 Ops 本质上是不同的，也就是他们掌握的技能是不同。

虽然在 DevOps 看来，Dev 和 Ops 的最终目标是一致的，都是为了快速向客户提供高质量的产品，但其达到目标的手段和方法是不一样的。比如，Ops 往往需要更多的在线处理问题的经验，而这未必是 Dev 所具备的。

所以，简单地把 DevOps 看做是一个职能，是一个彻底错误的观点。

第三，DevOps 是一种文化，推倒 Dev 与 Ops 之间的阻碍墙。

DevOps 是通过充分的合作解决责任模糊、相互推诿的问题和矛盾。在著名的演讲《每天部署 10 次以上：Flickr 公司的 Dev 与 Ops 的合作》 中，就明确的指出工具和文化是他们成功的原因。

其实，DevOps 通常想要告诉我们的是：什么行为是值得被鼓励的，而什么行为需要被惩罚。通过这样的方法，DevOps 可以促使我们形成良好的做事习惯，也就是 DevOps 文化。

所以，我们可以发现引入 DevOps 的组织，其实都是希望塑造这样的一种：信任、合作、沟通、学习、分享、共担等鼓励协作的文化。

第四，DevOps 是一种组织架构，将 Dev 和 Ops 置于一个团队内，一同工作，同化目标，以达到 DevOps 文化地彻底贯彻。

这看起来确实没有什么问题，而且敏捷团队往往都是这么去做的。但是，从另一方面来看，Ops 作为公司的公共研发资源，往往与 Dev 的配比是不成比例。所以，虽然我们希望每一个敏捷团队都有 Ops，但这可能是一种奢求。

但是，敏捷团队也说了，不一定是要有一个专职 Ops 人员，只要有负担这个角色职责的成员存在即可。这当然也讲得通，但可能真正的执行效果就没有 DevOps 所设想的那么好了。

所以，DevOps 是一种组织架构，这种说法，也对也不对，主要视组织的具体情况而定。

总结

今天，我和你一起回顾了 DevOps 产生的历程。同时，也顺便带你回顾了一下爱打盹儿的持续交付。我希望通过这篇文章，你可以理清持续交付和 DevOps 的关系：

1. DevOps 的本质其实是一种鼓励协作的研发文化；
2. 持续交付与 DevOps 所追求的最终目标是一致的，即快速向用户交付高质量的软件产品；
3. DevOps 的概念比持续交付更宽泛，是持续交付的继续延伸；
4. 持续交付更专注于技术与实践，是 DevOps 的工具及技术实现。

思考题

DevOps 大潮袭来，企业是不是真的就不需要 Ops 这个岗位了呢？

欢迎你给我留言。

通过留言可与作者互动