01 | DevOps的“定义”：DevOps究竟要解决什么问题？

2019-10-08 石雪峰

DevOps实战笔记

进入课程 >

讲述：石雪峰

时长 11:38　大小 10.67M

你好，我是石雪峰。今天我们来聊一聊 DevOps 的“定义”。

近些年来，DevOps 在我们身边出现的频率越来越高了。各种大会上经常出现 DevOps 专场，行业内的公司纷纷在都招聘 DevOps 工程师，企业的 DevOps 转型看起来迫在眉睫，公司内部也要设计和开发 DevOps 平台……这么看来，DevOps 似乎无处不在。

可回过头来想想，关于 DevOps，很多问题我们真的想清楚了吗？所谓的 DevOps 平台，是否等同于自动化运维平台，或持续交付平台呢？DevOps 工程师的岗位描述中又需要写哪些技能要求呢？另外，该如何证明企业已经实现了 DevOps 转型呢？这些问题真是难倒了一众英雄好汉。说到底，听了这么久的 DevOps，它的“定义”到底是什么，好像从来没有人能说清楚。

现在，我们先来看看维基百科对 DevOps 的定义。不过，估计也没谁能看懂这到底是在说什么。

> DevOps（开发 Development 与运维 Operations 的组合词）是一种文化、一场运动或实践，强调在自动化软件交付流程及基础设施变更过程中，软件开发人员与其他信息技术（IT）专业人员彼此之间的协作与沟通。它旨在建立一种文化与环境，使构建、测试、软件发布得以快速、频繁以及更加稳定地进行。

于是乎，每当提及 DevOps 是什么的时候，最常出现的比喻就是“盲人摸象”。有意思的是，DevOps 之父 Patrick 第一次参加 DevOpsDays 中国站活动的时候，也使用了这个比喻，看来在这一点上，中西方文化是共通的。毕竟每个人的视角都不相同，看到的 DevOps 自然也是千差万别。

DevOps 大潮汹涌而来，很多人都被裹挟着去探索和实践 DevOps，甚至有一种极端的看法认为一切好的实践都属于 DevOps，而一切不好的实践都是 DevOps 的反模式。

当年敏捷开始流行的时候，似乎也是相同的论调，但这种笼统的定义并不能帮助我们理清思路，甚至会带来很多负面的声音，比如 DevOps 就是开发干掉运维，又或者，DevOps 就是要让运维抛弃老本行，开始全面转型做开发。这让很多 IT 从业人员一度很焦虑。

客观来说，从 DevOps 运动诞生开始，那些先行者们就从来没有试图给 DevOps 下一个官方的定义。当然，这样做的好处很明显，由于不限定人群和范围，每个人都能从自己的立场来为 DevOps 做贡献，从而使 DevOps 所涵盖的范围越发宽广。

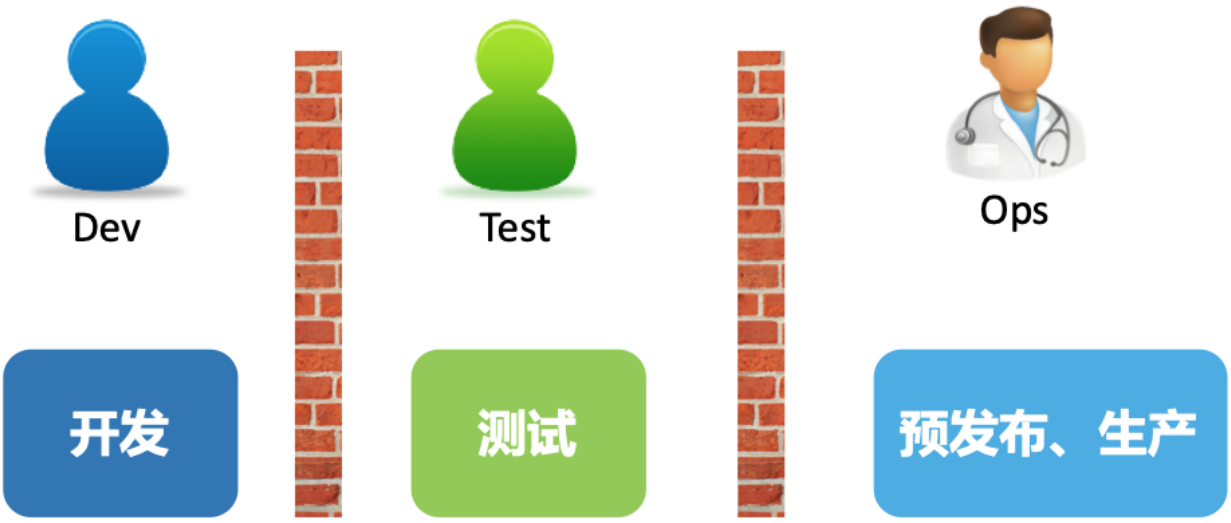
但是，坏处也是显而易见的。随着 DevOps 的不断发展，刚开始接触 DevOps 的人往往不得要领，只见树木不见森林，认知的偏差使得 DevOps 越发地神秘起来。

与其纠结于 DevOps 的定义，不如让我们一起尝试回归原始，来看看 DevOps 究竟要解决的是什么问题。

其实，DevOps 的秘密就来源于它的名字所代表的两种角色——**开发和运维**。那么这两种角色之间究竟有什么问题呢？我们从软件工程诞生以来所历经的三个重要发展阶段说起。

瀑布式开发模式

瀑布模式



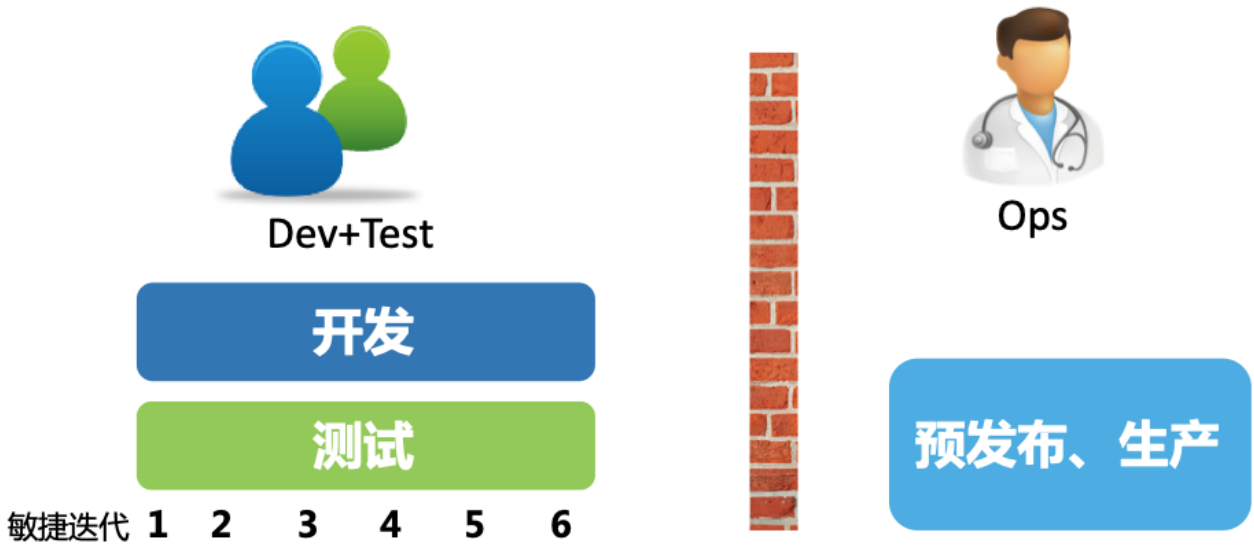
瀑布式开发模式将软件交付过程划分成几个阶段，从需求到开发、测试和运维，它的理念是软件开发的规模越来越大，必须以一种工程管理的方式来定义每个阶段，以及相应的交付产物和交付标准，以期通过一种重流程，重管控，按照计划一步步推进整个项目的交付过程。

可是，随着市场环境和用户需求变化的不断加速，这种按部就班的方式有一个严重的潜在问题。

软件开发活动需要在项目一开始就确定项目目标、范围以及实现方式，而这个时间点往往是我们对用户和市场环境信息了解最少的时候，这样做出来的决策往往带有很大的不确定性，很容易导致项目范围不断变更，计划不断延期，交付上线时间不断推后，最后的结果是，即便我们投入了大量资源，却难以达到预期的效果。

从业界巨头 IBM 的统计数字来看，有 34% 的新 IT 项目延期交付，将近一半的应用系统因为缺陷导致线上回滚，这是一件多么令人沮丧的事情。

敏捷式开发模式



基于这种问题，敏捷的思潮开始盛行。它的核心理念是，既然我们无法充分了解用户的真实需求是怎样的，那么不如将一个大的目标不断拆解，把它变成一个个可交付的小目标，然后通过不断迭代，以小步快跑的方式持续开发。

与此同时，将测试工作从研发末端的一个独立环节注入整个开发活动中，对开发交付的内容进行持续验证，保证每次可交付的都是一个可用的功能集合，并且由于质量内建在研发环节中，交付功能的质量也是有保障的。

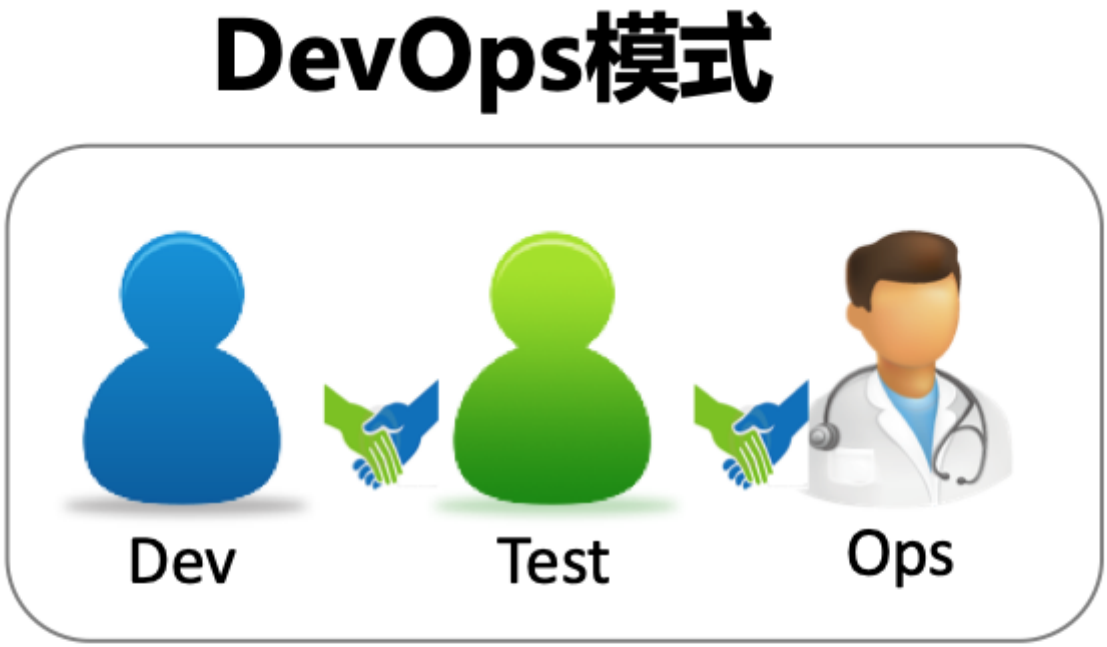
很显然，敏捷是一种更加灵活的研发模式。经常有人会问，敏捷会直接提升团队的开发速度吗？答案是否定的。试想一下，难道说采用了敏捷方法，研发编码的速度就会提高两倍甚至三倍吗？回想一下很多年前在 IT 行业广为流传的“人月神话”，我们就能发现正确的认知有多么重要。

敏捷之所以更快，根本原因在于持续迭代和验证节省了大量不必要的浪费和返工。关于这一点，我会在敏捷和精益的相关内容中做更加详细的介绍。

说到底，敏捷源于开发实践，敏捷的应用使得开发和测试团队抱团取暖。可是问题又来了，开发和测试团队发现，不管研发的速度变得多快，在软件交付的另一端，始终有一群人在冷冰冰地看着他们，一句“现在没到发布窗口”让多少新开发的功能倒在了上线的门槛上。

毕竟，无论开发了多少“天才”的功能，如果没有经过运维环节的部署上线，并最终发布给真实用户，那么这些功能其实并没有什么用。

DevOps 模式



于是，活在墙的另一端的运维团队成了被拉拢的对象。这些在软件交付最末端的团队始终处于一种“背锅”的状态，他们也有改变的意愿，所以 DevOps 应运而生，也就是说，DevOps 最开始想要打破的就是开发和运维之间的对立和隔阂。

在传统模式下，度量开发团队效率的途径就是看开发完成了多少需求。于是，开发为了达成绩效目标，当然也是为了满足业务需求，不断地堆砌新功能，却很少有时间认真思考这些功能的可运维性和可测试性，只要需求状态流转到开发完成就万事大吉了。

而对于运维团队而言，他们的考核指标却是**系统的稳定性、可用性和安全性**。但现代 IT 系统是如此复杂，以至于每一次的上线发布都是一场战役，整个团队如临大敌，上线失败的焦虑始终如影随形。

很多时候，我们并不知道上线之后会发生什么，只能按照部署手册一步步操作，完成之后就听天由命。所以，每逢大促活动，就会有各种“拜服务器教”的照片广为流传。

另一方面，在无数次被开发不靠谱的功能缺陷蹂躏得体无完肤之后，运维团队意识到，变更才是影响他们绩效目标的最大敌人。于是，预先设立的上线窗口就成了运维团队的自留地，不断抬高的上线门槛也使得开发团队的交付变成了不可能完成的任务，最后，“互相伤害”就成了这个故事注定的结局。

即便到了今天，部署上线在大多数公司依然是一件很神圣的事。我给你讲一件有趣的事情。

去年我在欧洲拜访 DevOps 之父 Patrick 的时候，曾经去过他的公司。那天风雪交加，比利时根特显得非常冷清。我们停好车后，刚要推门进入他们公司，恰好碰到 Patrick 和他的一个同事下楼抽烟。

简单寒暄之后，我们才知道，原来 Patrick 公司负责的一个系统要在 15 分钟后上线，他们趁这个间歇出来换换脑子，然后再回去大干一场。所以你看，连 DevOps 之父在面临上线的时候都如此正式，可见，DevOps 的发展之路依然任重而道远啊。

从一开始想要促进开发和运维的协作，团队慢慢发现，**其实在整个软件交付过程中，不仅只有开发和运维，业务也是重要的一环。**

比方说，如果业务制定了一个不靠谱的需求，那么无论开发和运维怎样协作，得到的终究是一个不靠谱的结果，以及对人力的浪费。可是业务并不清楚用户的真实情况，于是运维团队慢慢转向运营团队，他们需要持续不断地把线上的真实数据和用户行为及时地反馈给需求团队，来帮助需求团队客观评估需求的价值，并及时作出有利于产品发展的调整，这样一来，业务也被引入到了 DevOps 之中，甚至诞生了 BizDevOps 这样一个专门的词汇。

那么，既然沟通协作放之四海皆准，安全也开始积极地参与进来。安全不再是系统上线发布之后的“定时炸弹”，而是介入到整个软件开发过程中，在每个过程中注入安全反馈机制，来帮助团队在第一时间应对安全风险，那么，对于安全团队来说，**DevSecOps**就成了他们眼中的 DevOps。

这样的例子比比皆是，包括职能部门、战略部门等，都纷纷加入其中，使得 DevOps 由最开始的点，扩展为线，再到面，不断发展壮大。每个人都参与其中，这使得 DevOps 成了每一个 IT 从业人员都需要学习和了解的知识和技能体系。

说到最后，我还是希望基于我对 DevOps 的理解，给出一个我自己的“定义”：

DevOps 是通过平台（Platform）、流程（Process）和人（People）的有机整合，以 C（协作）A（自动化）L（精益）M（度量）S（共享）文化为指引，旨在建立一种可以快速交付价值并且具有持续改进能力的现代化 IT 组织。

总结

今天，我带你一起梳理了 DevOps 的发展历程，以及软件开发模式的变迁。有人说，DevOps 是软件工程发展至今的第三次革命，可见它带给整个行业的影响是很深远的。人云亦云并不能帮助我们更好地理解 DevOps，建立正确的认知才是体系化学习的第一步，希望你能通过今天的课程，建立起你自己对于 DevOps 的独特认知。

思考题

最后，给你提一个问题：我给出的定义符合你心目中对 DevOps 的预期吗？DevOps 具有与生俱来的开放性，你能谈一谈你对 DevOps 的理解和定义吗？

欢迎在留言区写下你的思考和答案，我们一起讨论，共同学习进步。如果你觉得这篇文章对你有所帮助，欢迎你把文章分享给你的朋友。

上一篇　开篇词 | 从默默无闻到风靡全球，DevOps究竟有什么魔力?

下一篇　02 | DevOps的价值：数字化转型时代，DevOps是必选项?

精选留言 (33)

写留言

Jxin
2019-10-08

1.与个人认知有点分歧。您对devops的描述个人认为在how的范涛，也就是怎么做。而描述一个东西，感觉用what来描述会合适点，也就是是什么，解决什么问题。
2.个人认为，devops的出现，是当下软件工程模式的痛点和互联网环境共同催生的。在敏捷开发的模式下，沟通协作成本和多工序质检成本对软件工程的阻碍性高于职责分工带来的促进性。故采用这种垂直整合的方式，虽失去了分工带来的效能提升，但减少了沟通...
展开

作者回复: 感谢你的回复，其实对于DevOps的定义，我是加了双引号的，因为这个定义也只是一家之言。
的确，DevOps要解决的还是软件研发交付能力和业务需求快速多变之间的矛盾。职责划分是企业精细化运营的必然结果，因为每一个领域的知识厚度已经超越了以前英雄的时代，也就是一个人搞定一切，所以才有了不同的职能。现在忽然发现，职能和组织划分又产生了部门墙和沟通成本，甚至这个成本已经超出了做事本身，于是现在又开始推崇全栈工程师。
以我在企业中的观察，所谓全栈还为时尚早，更多的还是通过平台来解决只有专家才能做事的问题。顺便说一下，QA的能量依然很强大呀😁

7　9

Geek_5bc409
2019-10-08

瀑布模式在项目初始阶段对花费了大量的时间对业务进行梳理分析，确定了整体的架子和大概方向。敏捷并没有花费大量的时间去研究业务，从一个小系统开始不停迭代，会不会因为对业务没有分析透，出现一叶障目，方向偏离或者说系统开发到一半的时候，发现整个的总体业务架构是不合理的，需要大返工的情况
展开

作者回复: 你好，感谢你的留言。
说说我个人的看法，首先采用敏捷开发方式，与需求是否靠谱没有直接的关联关系，比如瀑布模式下，也有大力出悲剧的案例，比如摩托罗拉的铱星计划，再比如诺基亚智能手机的案例。
其次，说句实话，即便使用敏捷模式，现在很多需求都是拍脑袋拍出来的，至于说这个需求价值到底有多少，始终缺少直接的衡量，这就导致拍的越多，成功率越高，从而产生无穷无尽的需

求。
那么这也代表了，其实企业对于用户想要什么并不清楚，谁能想到电商行业两大巨头格局已定的情况下，偏偏杀出来个拼多多呢？
所以，无论是精益创业的MVP理论，还是反脆弱中的从不确定中收益的角度来说，企业以可控的成本不断试错，以博取一个无限的收益，已经成为了常态。
那么对软件开发来说，就需要有这样的能力，可以快速的把一个原型想法交付上线，并收集线上数据实时反馈给业务方，判断是否需要继续投入资源。这个过程的成本越低，企业的成长性就越好，从而不断发展下去。
至于说软件架构这个东西，我还没见过哪个架构做出来10年不变的，即便我们自己的内部平台，也是重构重构再重构，所以没必要一步到位，更多是还是演进式思路，要不然开发怎么磨砺自己的技能呢？

2 5

he7yong
2019-10-11

老师是否可以把业务需求分析，架构设计DDD，这些和devops之间是否有什么关系，也讲讲？

作者回复: 你好，你是指Deadline Driven Development 嘛 😆
开个玩笑，我对DDD的理解还没有那么深，所以就不胡说八道啦，推荐你参考张逸的专栏哈😄

1 2

Oliver
2019-10-09

移动端的开发如何去实现DevOps呢，目前我们做的只是通过Jenkins来持续集成打包而已，不知道在移动端DevOps的方面还可以做哪些东西

展开

作者回复: 你好，我目前在公司就是重点负责移动端的DevOps建设，除了你提到的自动打包，其实能做的事情非常多，比如自动发布，自动多渠道管理，测试和代码质量门禁的建设，组件化的能力建设等等，相当于一整套移动端的持续交付链路能力，重点看你们当前的阶段，和急需解决的问题哈，可以提出来一起讨论。

1 2

leslie
2019-10-09

粗谈一下个人的理解吧：可能我个人最深的感受是OPS只会部署，调教和解决代码问题能力很差。自己是从程序员-数据库开发-数据库开发兼数据库运维然后就一直在OPS和DBA之间不断主从切换多年：运维只是运维，开发只是开发的问题碰到太多了。

最初了解这个概念是在Google的SRE一书中：近半年一直在努力强化和完善自己，课程开课之前其实就在学<全栈工程师指南>;运维其实是各个环节的中间层，尤其是现在数...

展开

作者回复: 果然运维都是自带万能光环出现的，而且又无处不在，我从开发的角度深刻能体会到对运维环境的未知和恐惧，只要不是代码的问题，就全部是运维的问题，这种思维定势，直接导致了很多问题直接都会丢给运维解决，这对运维团队的能力和压力都是双重考验啊。
回到全栈工程师的问题，目前看到更多的还是领域内的全栈，比如前后端开发，比如全栈测试，真正的全栈还是凤毛麟角吧。
Anyway，感谢你的留言，也期待你的更多分享。

2

就不告诉你
2019-10-09

如果一个公司本身就是集需求开发测试运维于一体的一站式服务，那DevOps意味着什么？

作者回复: 我的理解一站式服务并不能解决交付效率和质量的问题哈，当然现在基于DevOps的一体化平台无论在公司内部，还是外部，都有不错的需求，比如阿里云效，我相信基础设施云化只是一个开始，后面很多能力都会SaaS化，也包括效率建设方面的能力哈。

1 2

阿硕
2019-10-08

寻找志同道合的一帮人，达成共识，实现目标，研发，测试，运维的所做所为，这很DevOps

作者回复: 赞，我特别喜欢你的这个评论，希望越来越多的公司都能“很DevOps”哈！

1 2

无言的约定
2019-10-08

做java后台开发的适合学习吗?

展开

作者回复: 你好，我只能说，单从软件形态上划分，基于服务端的业务相对最适合于DevOps，因为没有固定的发布周期，服务拆分治理又有相对比较成熟的框架，比如Spring Boot，各种配套的工具平台比较完善，更不要说容器应用后的高可扩展性，所以在企业里面推行DevOps，优选的还是服务端业务。
就像我在专栏中说的那样，我觉得DevOps应该是所有IT从业人员都需要了解的内容，也许不需要那么精通，但也不能完全不清楚，毕竟DevOps是大势所趋哈。

3 2

helloworld
2019-10-12

之前对DevOps的理解就是开发+运维的综合体，看了老师的文章后感觉整个公司的所有部门都是DevOps的一部分

展开

作者回复: 因为DevOps太火啦，所以大家都不能置身事外呀，其实沟通协作不仅仅在dev和ops两个部门之间有，在外部的安全，业务部门也同样存在，而且部门墙更加严重，比如安全要求应用上线发布前必须经过审核，每次审核要一天时间，而有时候为了早一天发布，开发测试都是通宵达旦，这要如何解决呢？还是自动化，对其目标，共享上线计划，把安全团队拉进来才行嘛。

1 1

鲍建飞
2019-10-10

设备端可以集合devops么

展开

作者回复: 你所说的设备端是指嵌入式或者智能硬件设备吗？其实这些产品同样需要提高软件交付效率和质量，我记得在DevOps实践指南里面就提到过一个POS机实践DevOps的案例，他们通过分批下发，自主部署的方式，解决了一次性部署的问题和冲突，你可以找来看看。其实在我接触过的公司，同样有类似的平台支持远程升级，核心理念还是怎样通过工程能力建设解决实际的业务问题哈。

1

灬黑 礼服 ~
2019-10-10

我们在践行 gitops了，，，一股劲的冲。。。总感觉没有运维什么事情了，，，，开发门

槛高了，，，，运维技能要求高了，，，，全都自动化了

展开

作者回复: 赞啊，Gitops我也是前年在Jenkins X项目中开始接触的，的确非常DevOps，欢迎分享你们的实战经验。
其实很多人在说咖啡运维，也就是喝着咖啡把运维做了，但其实都依赖于运维能力的平台化自动化输出，我相信运维会更加寻求智能化和数据运营能力的建设，期待AIOps的专栏哈

1

悟空
2019-10-09

之前看了赵成老师的运维体系管理课程，还有部分同学提到的沟通成本问题，简单说一下，当做自己的输出
1. 古老的瀑布模式存在一个问题，研发、测试、运维各自有各自的“方言”，造成没有统一的标准，当业务规模越来越大，简单的配置管理都变得异常困难，更别提怎么用自动化取代人工操作；...

展开

作者回复: 嗯嗯，协作的前提是大家有相同的认知，如果你觉得重要的事情，对他来说并没有什么意义，那就没有合作的基础，所以DevOps最重要的就是让大家认同价值交付是共赢的事情，并在这个基础上先前一步吧。

1

陈斯佳
2019-10-13

光看评论都能学到好多...感谢老师的认真！

展开

陈斯佳
2019-10-13

DevOps对我而言，核心是快速反馈，让团队能以最小代价最快的速度获得真实程序反馈，从而让各个部门对不确定的目标更加清晰一点。平台和流程的自动化只能能保证效率的提高，但其中推动这一反馈闭环形成的还是人，一个愿意拥抱变化，不断精进迭代的人。希望在这个平台能遇上这样的一群人

展开

小白啊
2019-10-12

DevOps是各团队(已不单指开发与运维)一起紧密协作工作，以更快更好的构建、测试、发布软件交付价值为目标。DevOps发展过程中渐渐形成了DevOps文化和方法论，同时各种工具平台不断发展和出现，有了方法论和工具，接下来就是实践，大家又在实践过程中不断完善发展方法论和工具。

展开

作者回复: 精辟的总结！如果总想着一步到位，那就失去了DevOps持续改进的精髓呀！这也是理论篇想要传递的核心思想。

昕宇
2019-10-11

老师好，请问一下devops与微服务之间是什么关系，不是很懂

作者回复: 你好，维服务是一种应用架构的设计风格，简单来说吧，以前的应用大多是单体的，也就是所有服务都打到一个软件包里面，这样的问题就是，哪怕任何人改一点代码，整个软件包都要重新生成，重新部署，这样肯定不够灵活。
而DevOps追求的是软件的灵活快速发布，所以如果把一个大应用拆成一堆小的组件，彼此独立部署发布，那就不用大家一起齐步走了，搞得快的人可以快点跑。
所以这两者没有啥因果关系，但是微服务应用更加容易发挥DevOps的威力，再加上云和容器，就构成了现在一体化的技术变革。

刘丹
2019-10-09

DevOps = 敏捷开发 + 持续集成 + 自动化测试 + 持续发布 ？

作者回复: 我理解你说的这部分属于工程实践，也是工程师日常打交道最多的内容，这些固然很重要，但是比如需求拆分是否合理，应用架构是否支持独立部署，持续发布了安全风险如何管控，团队协作是否高效，这些都会影响工程实践的效果哈

陈文涛
2019-10-09

文章拿瀑布模型做了举例，那是不是意味着所谓devops其实很接近于迭代模型呢？把周期很长的不可控切分为较小的迭代，再增加上对于组织架构人员分工导致各自为政问题的解决....devops跟迭代模型有什么区别呢？

作者回复: 你好，感谢你的留言。关于这一点我会在实践部分讨论到。简单来说，迭代开发方式是敏捷开发中的一种方法，以Scrum为典型代表，但是迭代并不等于敏捷，因为只是单纯的把大的任务拆分成小任务，并没有用到敏捷的精髓。而对于DevOps可以理解为敏捷的延伸，就像我在文章中提到的那样，DevOps之所以诞生就是希望将开发过程中的敏捷方法拓展到运维环节。也就是说敏捷更加关注研发活动，而DevOps会覆盖到整个交付过程。
当然，对于这些概念的理解，要看你站在什么角度，类似LESS这种大规模敏捷框架中，DevOps也只偏安一角哈😄

1

Bryan Bai
2019-10-09

DevOps 是个双核系统，技术方面的关键字是automation，自动化的是以前手动的流程。这里有个关键问题就是这个手动过程是不是正确的，如果本身是个错误的流程，或者识别错误，那么这个automation只会放大错误。文化方面的关键词是collaberation ,也就是通过文化和工具的带动，使得沟通更快速和双向。

展开

作者回复: 非常认同你的观点，其实与其说是错误的流程，不如说是陈旧僵化的流程，毕竟流程始终存在，也的确依靠流程解决了历史上的问题。但是随着标准化和自动化的演进，为流程改进带来了更多的可能性，比如以前依靠人的环节，可以被自动化辅助甚至取代，那么这就是一种进步。

洋阿让
2019-10-09

期待学会能实用，理论尽量精确，精确不了的别硬精确，把可能性留出来。

作者回复: 你好，关于DevOps一切皆有可能哈😄