08 | DevOps、SRE的共性：应用全栈思路打通开发和运维

2019-09-09 葛俊

研发效率破局之道

进入课程 >

讲述：葛俊

时长 18:02　大小 16.52M

你好，我是葛俊。今天，我来跟你聊一聊 DevOps 和 SRE。

DevOps 和 SRE，尤其是 DevOps，是最近几年软件研发方法的重要趋势。因为它们都跟打通开发和运维流程相关，所以常常被混淆。比如，SRE 等同于 Google 的 DevOps、SRE 是升级版的 DevOps，就是两个常见的误区。

事实上，DevOps 和 SRE 虽然关系紧密，但差别还是蛮大的。所以今天，我首先会通过 DevOps 和 SRE 的对比，引出它们背后的 Why、How 和 What（也就是它们的目标、原则和具体实践）。然后，我会结合自己在一个创业公司的经验，向你推荐如何在团队落地 DevOps。

DevOps 和 SRE 的定义和异同

因为 DevOps 和 SRE 都是比较新的概念，而且在不断地发展变化，所以学术界和工业界对它们的定义并未达成一致。

接下来，我参考已有定义，并加入自己的理解，对 DevOps 和 SRE 的大致定义如下：

DevOps，Development 和 Operations 的组合词，是一种重视“软件开发人员（Dev）”和“IT 运维技术人员（Ops）”之间沟通合作的文化、活动或惯例。 通过自动化“软件交付”和“架构变更”的流程，使得软件的构建、测试、发布更加快捷、频繁和可靠。

SRE，全称是 Site Reliability Engineer ，网站可靠性工程师，是一个职位，是软件工程师和系统管理员的结合，主要目标是创建可扩展且高度可靠的软件系统。

为达到这个目标，SRE 需要掌握如下相关知识：算法、数据结构、编程、网络编程、分布式系统、可扩展架构、故障排除等。SRE 使用工具和系统支撑其完成工作，比如自动化发布系统、监控系统、日志系统、服务器资源分配和编排工具等，而这些工具往往需要他们自己开发和维护。

所以总结来说，**DevOps 是打通开发和运维的文化和惯例，而 SRE 是 DevOps 的具体实践之一。**说到相同点，它们都是为了打通 Dev 和 Ops，提高研发效能；说到区别，DevOps 是文化，而 SRE 是职位。如果要类比的话，**DevOps 与 SRE 的关系，就像敏捷跟 Scrum 的关系。**

理解了 DevOps 和 SRE 的定义和异同后，我们再来看看它们的目标、原则和具体实践吧。

DevOps 和 SRE 的 Why、How、What

不知道你有没有考虑过，传统定义中，开发人员和运维人员的利益其实是冲突的。

开发人员的职责是开发功能。功能上线越多，对团队和公司的贡献就越大。所以，开发人员倾向于多开发功能，并快速上线。而运维人员的主要职责是，上线功能，以及保证系统的稳定性，更关注系统的稳定性。新功能上线越多，工作量就越大，服务越容易不稳定，所以运维人员不愿意功能多上线、快速上线。

另外，职能竖井越严重，这些问题就越严重。因为在这种情况下，开发人员写完代码就扔给运维人员撒手不管了，运维人员也很少能从开发人员手里拿到有效信息。此时，线上问题的

修复对运维人员来说挑战很大。于是，他们会更加谨慎。

这就是 DevOps 和 SRE 最初要解决的问题：开发和运维这两个角色的目标不一致，导致研发和上线流程的不顺畅，最终严重影响软件上线的效率。比如，运维团队倾向设置多而严格的上线检查门禁，限制上线频率。而开发人员则很可能把一些功能“伪装”成 Bug 修复，来绕过针对版本发布的严格检查。

那怎么解决这个问题呢？接下来，我向你推荐以下几个原则和具体方法。

原则 1：协调运维和开发人员的目标、利益

目标、利益不一致，是导致开发团队和运维团队矛盾的首要问题，所以想办法让它们的目标、利益变得一致，是整个 DevOps 中最重要的一条。因为只有协调好了目标和利益，它们才有动力去解决问题。实现这个原则的一个最主要的方法，我称之为“**全栈开发**”。

什么意思呢？全栈开发，就是每一个工程人员的工作涵盖了不止一个领域。虽然他专攻某一个领域，但对负责的领域都有责任。说白了，就是对产品结果负责，而不是只对某一个具体环节负责。

具体的实施方法，主要包括两大方面。

第一，增加一个新的运维角色，用开发的方式去做运维。不同于传统运维人员主要关注网站和服务的稳定和快速，这种新型运维角色负责帮助开发人员推动业务快速开发、快速上线，具体工作包括：优化流程，提供自助工具。

在 Facebook，这个角色由一个专门的发布工程师团队承担。而在 Google，这个角色就是 SRE。具体来讲，SRE 的工作职责是负责日常运维、紧急响应、工具研发、建设平台化服务体系、容量规划和容量管理、Oncall 等。他们会被指派到具体的产品团队中，深入到开发第一线，拿出至少 50% 的时间进行编程工作，针对性地自动化和优化 CI/CD 中的流程、工具等。这里请注意，他们的开发工作不是业务开发，而是工具开发和自动化等。

与 Facebook 的发布工程师团队相比，SRE 的最大特点是更多地参与到具体产品和项目中。为了方便讨论，**我把 SRE 和发布工程师团队这种新的运维角色，统称为“类 SRE”。**类 SRE 顺利推进的关键是，找到高质量的开发、运维多面手，也就是有很强的开发能力，同时有系统维护、网络问题排查等运维技能的工程师。

需要注意的是，除了类 SRE，传统的运维角色（比如网络工程师、系统工程师）仍然存在，他们也使用类 SRE 提供的 DevOps 工具链提高工作效率，所以人员数量比以前少了很多。在遇到线上问题时，开发、类 SRE 和传统运维需要配合解决。

第二，修改开发人员的职责描述为，快速开发和上线稳定的高质量产品，让他们也参与到一部分运维工作中去。

开发人员最主要的工作仍然是开发，但会使用类 SRE 团队提供的工具、流程来进行发布相关的工作，包括代码部署、线上问题定位和处理等部分工作。这样，他们会在代码开发时就注意提高服务的稳定性和代码质量。

比如在 Facebook 进行日部署的时候，开发人员写好了修复的代码提交，还需要去找到提交所依赖的、还没有上线的其他提交，跟这些提交的作者进行确认，没有问题后才可以一起上线。

上线时，这个开发人员对这一组提交负责，把它们全部提交到日部署流程工具上，并负责功能验证。而负责部署的发布工程师，则会使用工具自动化地把这些提交部署到日部署环境中，并进行系统验证。可以看到，开发人员较多地参与了整个部署过程，这正是持续开发的体现。

另一个比较直观地体现开发人员参与运维工作的实践是，让开发人员 Oncall，也就是身上别个 BP 机，线上出现问题要马上响应，即使半夜三更也要爬起来，让功能的开发人员承担起解决一线问题的职责。

通过引入类 SRE 角色和修改开发者职责描述这两种方法，开发人员和部分运维人员的职责，都从原来的单一职能扩展到了产品的高效开发上线，从根本目标和利益上实现了这两个角色的对齐。

所以说，“全栈开发”方式非常重要，效果也非常好。这，也正是我把“全栈”用在文章标题中的原因。

原则 2：推动高效沟通

之所以强调推动高效沟通，就是为了解决开发和运维因为存在“部门墙”而导致的信息不流通的问题。

具体的实施方法包括：

设置聊天室（比如 IRC），用于沟通部署进展和问题。

引入 ChatOps，也就是通过自动化的方式，提高部署相关的沟通。比如，可以让聊天机器人在 ChatOps 聊天室中自动发送发布流程的进展，也可以向聊天机器人询问服务部署进展等。这些自动化可以节省很多部署人员的时间。

把任务、代码提交、发布的关联关系，在工具中呈现出来，并提供比如任务看板、系统监控看板等可视化工具，显示开发、运维重点信息。这样可以大大提高运维人员定位问题的效率。

原则 3：优化从开发到部署的整个上线流程

优化上线流程，主要目的是解决频繁上线工作量大、产品质量不稳定的问题。

这个原则的主要方法是优化代码入库和部署上线流程，并最大化地利用工具来自动化流程。最近 10 年这个原则发展得非常快，逐渐涵盖了快速、高质量上线的各方面工作。这些方面的效率也被称作**交付效能**，具体实践包括：基于主干开发、持续集成、持续交付、持续部署、实现系统松耦合等。

前 4 条实践，我们已经在前面的文章中讲解过了；而关于系统松耦合，我会在“工程方法”模块中与你详细讨论。

理解了 DevOps 和 SRE 的目标、方法和具体实践后，我们再来看看具体怎么落地吧。

落地步骤推荐

在落地 DevOps 时，我经常看到有些团队一上来就去寻找工具，比如使用 Jenkins 来搭建流程。但正如上面所说，DevOps 的本质是解决开发和运维之间的冲突，所以落地时首先要从人出发，然后是流程，最后才是工具。

所以，我推荐的具体落地步骤是：

1. 对团队目标达成共识，并重新定义职责；
2. 设计 CI、CD、快速反馈，以及团队沟通等流程；
3. 引入工具，实现自动化。

接下来，我就用我在一家创业公司的实践来帮助你理解这些落地步骤。

落地实践案例

这个案例，是我在一个创业公司落地 DevOps 的实例。当时公司刚成立不久，我的角色是技术总负责人，操作的自由度很大。为了提高产品研发的交付效能，也就是团队快速发布高质量软件的能力，我采用了以下步骤和方法。

第一，在团队内部统一认识

研发团队内部统一认识，主要是让团队成员明确我们的目标一致。同时在做绩效考评时，对大家的要求都是产品快速、高质量上线。

我还把“统一认识”这一点，扩展到研发以外的部分，比如市场、设计团队。在推广时，我始终注意以公司利益为出发点，而不是按照职能划分来追责，获得了 CEO 和其他团队管理者的支持。最终，推广得比较顺利。

第二，增加“发布工程师”角色

这个角色的职责，相当于 Facebook 发布工程师团队的职责。因为我在部署、运维方面的经验相对丰富，便担起了这个工作，投入相当一部分精力去设计、优化产品的开发和上线流程，包括持续集成、持续交付、手动部署流程，以及问题监控流程和 Oncall 机制。

其他开发人员则使用这个流程来部署、上线、监控、解决问题，提高交付效能。在这个过程中，我作为发布工程师，主要起到了“使能”的作用，即日常的部署上线由开发人员自己负责，而我只参与解决一部分运维相关的难题。

第三，设计问题沟通流程

在解决了“人”的问题之后，我在流程方面，设计了部署沟通流程，以及部署系统常见问题及解决办法收集流程。

部署沟通流程是：建立了专门的聊天室沟通持续集成、持续交付，以及手动部署的进展情况，确保发布上线流程相关信息的顺畅流通。

部署系统常见问题及解决办法收集流程是：在解决完线上问题之后，记录下问题的细节和解决步骤。很简单，就是因为很多问题会重复出现。每个人都可以更改、搜索这些记录，对后

续解决问题帮助非常大。所以，我们每次遇到问题，都会先搜索这个知识库，找到答案和线索的概率能达到百分之七八十。

第四，用工具来实现流程自动化

完成了有关人和流程的工作之后，工具的实现就不那么难了。DevOps 相关工具主要包括两大类：CI/CD 工具链和沟通工具。

我已经在第 5、第 6和第 7篇文章中，与你详细讲述了 CI/CD 流水线的内容；而沟通工具的内容，将会是下一篇文章的主题。这里，我给出当时我们使用的工具链，也就是一个小型创业公司落地 DevOps 具体工具链的示例，供你参考。

任务	工具
CI/CD	GitLab + CodeCI + AWS EC2
任务管理	Pivotal Tracker，后来转为MeisterTask
监控、分析	Fabric、MixPanel
Wiki、文档中心	Google Docs
问题讨论	邮件、MeisterTask、Slack

图 1 CI/CD 工具链和沟通工具

通过这一系列针对人、流程、工具的措施，整个研发团队实现了全栈开发，目标一致、流程顺畅，能够聚焦于开发，交付效能非常高，实现了后端服务每周上线三次，热修复上线时间 5 分钟，MTTR（Mean Time To Recovery）大概 15 分钟，线上也很少出问题，系统的可用性达到 4 个 9（即 99.99%）。

同时，对于开发者来说，全栈开发机制也优化了开发体验，提升了技术成长速度。

当然，这些实践能够顺利推广，与这是一个小的初创公司有关。一方面，公司存亡直接关乎个人利益；另一方面，一切都在建设期，引入具体实践的阻力非常小。如果是在大的团队，或者是职能已经成型的团队，实施的困难会相对较大。这时，我建议你从以下 3 个方面入手：

在团队普及全栈开发的理念；

在你管理范围内推行全栈开发，做出效果；

通过实际运行效果获得公司管理层的支持，让他们了解全栈开发模式为提升研发效能带来的好处，从而逐步改变职责定义和流程。

小结

今天，我给你介绍了 DevOps 这个很火的话题，以及 SRE 和它的关系。文章中，我与你介绍了 DevOps 的目标、原则和具体实践。这三条原则分别是协调开发和运维的目标、推动高效沟通和”优化从开发到部署的整个上线流程。最后，我通过自己的实战经验，介绍了用“人、流程、工具”的步骤来落地 DevOps。

在我看来，打通开发和运维的“部门墙”，最关键、最根本的就是解决人的问题，也就是第一条原则，” 协调开发和运维的目标 “，要解决的问题。而全栈开发就是一个非常棒的解决方法。

简单来说，全栈开发就是让工程师不再只是对某一个单一职能负责，而是对最终产品负责。全栈开发是一个很好的抓手，逐步提高全栈开发的程度，大家的目标自然就会对齐，从而主动去提高，那其他方面的提高就容易得多了。

事实上，测试工作也是全栈开发的一部分。在 Facebook，没有专门的功能测试团队，而是有一个测试工具团队提供测试平台、流程和工具，供开发者使用。以此，让开发者更进一步地对整个产品负责。这种开发人员全栈，其他职能提供支持的工作方式，可以总结为如下所示的图片。

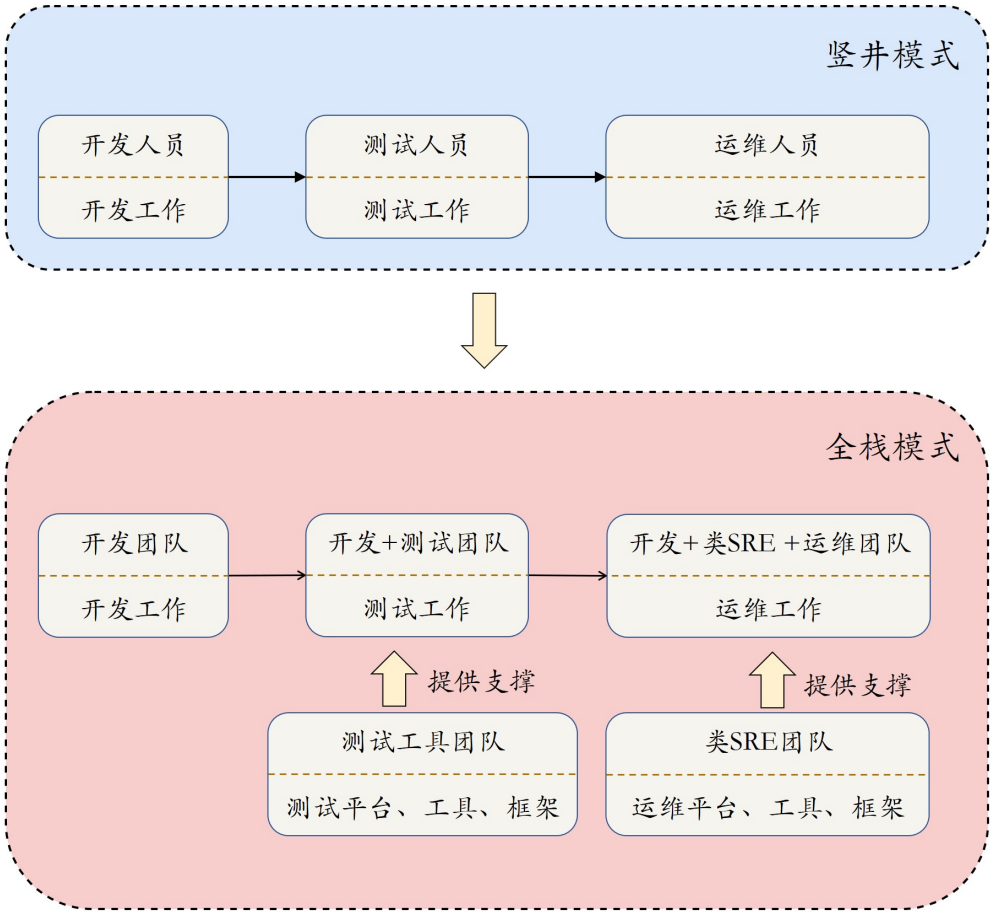

图 2 全栈开发方式示意图

最后，我再与你分享一点我的体会。

云技术尤其是 Docker 和 Kubernetes 的流行，使得越来越多的底层运维工作被自动化，导致对传统的系统工程师、运维工程师的需求越来越少。所以，懂得开发的运维人员，会越来越重要；同时，更关注部署、测试，甚至产品的全栈工程师，也会越来越受欢迎。

思考题

有些人认为“全栈工程师”要求太高，一个人不能掌握那么多领域的知识，不现实。你怎么看呢？

感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎你把这篇文章分享给更多的朋友一起阅读。我们下期再见！

上一篇　07 | 分支管理：Facebook的策略，适合我的团队吗?

精选留言 (5)

写留言

刘丹
2019-09-09

全栈工程师不一定全领域，准确的说法应该是多领域，掌握大部分领域，精通少数领域。

作者回复: 是的。全栈的开发是T字型人才。一横表示了解多领域。艺术时表示精通少数领域。

1

MiffyLiye
2019-09-09

现有的工具越来越成熟，个人/小团队能掌控的复杂度越来越高。
同时社会发展变快，提高研发效率，对市场需求快速反应越来越重要。
这些因素就让成为全栈工程师的投入产出比越来越高，变成了非常值得去发展的方向。

展开

作者回复: 👍👍👍

桃源小盼
2019-09-09

全栈工程师，在面对一些非常见问题时，会更快的定位问题，而不是以前那样，大家都觉得跟自己没关系，就等着别人去解决。

展开

作者回复: 是的。这就是目标一致的好处。

刘晓光
2019-09-09

全栈!=全干
全栈的支撑基础是服务化和技术封装。其实还是底层支撑越来越牛逼了。如果一个组织没有很好的底层技术支撑，

作者回复: 全栈!=全干 简介明了!

兴国
2019-09-09

成为全栈工程师也是分阶段，分领域的。比如平常主要做java的开发人员，基于工作需要，也需要写前端，会golang/php等语言；也需要对运维服务器进行部署/使用，使用运维工具进行发布等。刚开始可能对前端等只是会写，并不能搭建框架，不了解其原理；对服务器部署原理，网络搭建原理也不了解。但是能够满足日常的开发，运维需要。随着时间的推移，业务的发展以及问题的不断出现和解决。慢慢的能够更深入了解除java开发之...

展开

作者回复: 分时间段这个思考维度是对的。同时也说明了我们软件开发挑战和有趣的地方：不断有新的东西出现需要学习。