现代开发者都应掌握的CICD —— 第一章: 什么是DevOps

来自： 编程导航

Hask

2023年04月28日 12:32

在谈论CICD之前，我们首先得先了解一下DevOps。

相信大家或多或少都听说过DevOps，当然没有听说过也没有关系。DevOps从抽象的意义上来说就是一种方法论，一种实践方式。

那么什么是DevOps呢?

我们引入Wiki上的一段话:

> DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from the Agile way of working.

DevOps 是一套结合了软件开发 (Dev) 和 IT 运维 (Ops) 的实践方式。 它旨在缩短系统开发生命周期并提供具有高软件质量的持续交付。DevOps 与敏捷软件开发相辅相成； DevOps 的几个方面来自敏捷的工作方式。

回顾历史

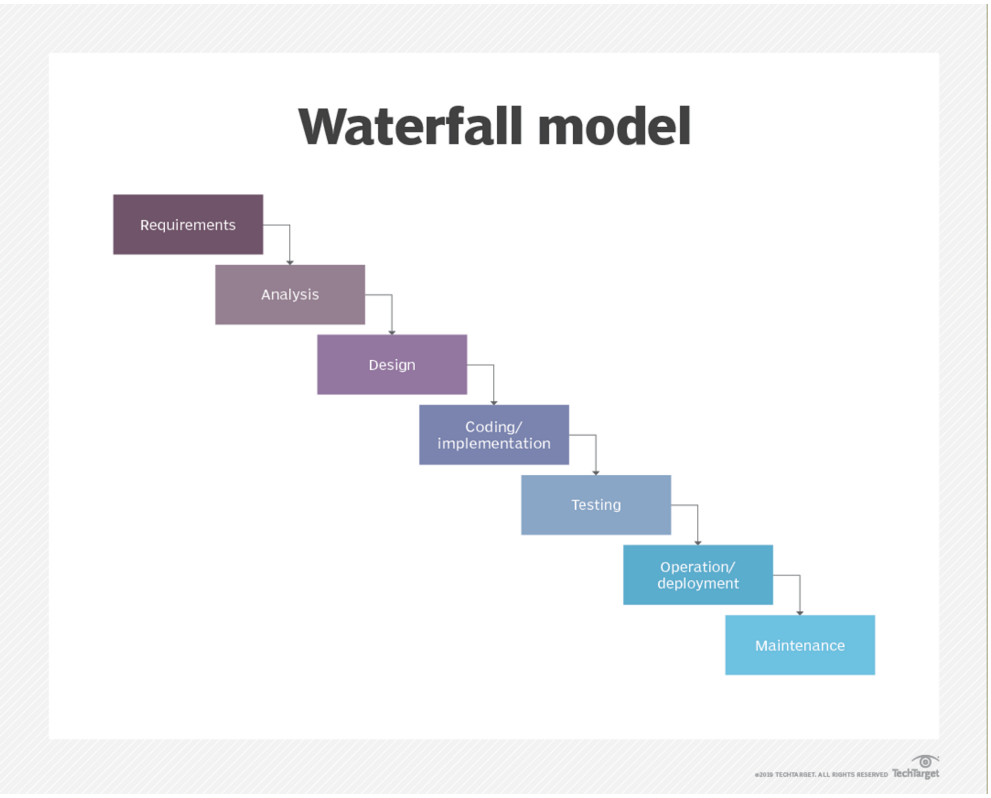
我相信能够学习历史是人类能够长远发展的重要方式，我们得从过去的失败中吸取足够的教训，最终才能走的更远，在带领大家学习这门课程时，我们先来讲讲在此之前人们是如何做的。

在计算机行业发展的早期，那时的业务复杂度还没有达到很高的程度，当时的程序员仅需要自身一人就能满足业务开发的需要。那时候，我们常常听说过“独立开发者”这么一个说法，至于现在则很少在知名的项目中听到了。那时候一个人的精力足以承担整个项目，在前端领域比较知名的一个案例就是JS之父Brendan Eich用了10天设计了JavaScript的原型。

（图为JS之父 Brendan Eich）

当然，随着计算机行业的不断发展，业务的需求不断的增加。面对高度复杂的需求往往不是一个人就能单打独斗完成工作的时代了，因此如同历史上很多类似的行业做的一样，软件工程师们把原来的工作逐步的进行了工作的细分：分析客户需求的分析师、设计界面与交互的设计师、实现逻辑的程序员、负责测试的测试工程师、负责发布的运维工程师……

并且发明了第一个可以实践的软件工程方式: 瀑布模型（Waterfall Model）。



瀑布模型是一种很好利用了不同职能的工作只会直接的与他们的直接上级和直接下级进行沟通的特性，将工作一级一级往下传递，只要每个节点都能保证自己的输入与输出最后就能得到比较好的结果。

这个方案确实是一种比较完美的解决方案，将软件开发变成了工厂的流水线，使得软件开发能够稳定、高效。但是与工厂的流水线不同的是，软件开发的成果往往是定制化或者说特殊的。在实际的落地过程中，往往会出现客户在过程中提出新的建议，甚至变更最早期的需求。这就意味着原来计划是单向的瀑布模型常常会面临着返工的困境，也就意味着项目的延期甚至失败。更加可悲的是，当时的人们没有好的解决方案。

但是就似乱世出英雄，每个时代都会有这么一些杰出的英雄跳出来，带领大家打破当时的困境，走向新的时代。在2001年，17位当时顶尖的行业人员聚在一起，发布了名为 **敏捷宣言(The Agile Manifesto)** 的倡议，引出了敏捷开发这一理念。

在敏捷宣言中，列举出十二条**敏捷原则**:

- 我们最重要的目标，是通过持续不断地 及早交付有价值的软件使客户满意。
- 欣然面对需求变化，即使在开发后期也一样。 为了客户的竞争优势，敏捷过程掌控变化。
- 经常地交付可工作的软件， 相隔几星期或一两个月，倾向于采取较短的周期。
- 业务人员和开发人员必须相互合作， 项目中的每一天都不例外。
- 激发个体的斗志，以他们为核心搭建项目。 提供所需的环境和支援，辅以信任，从而达成目标。
- 不论团队内外，传递信息效果最好效率也最高的方式是 面对面的交谈。
- 可工作的软件是进度的首要度量标准。
- 敏捷过程倡导可持续开发。 责任人、开发人员和用户要能够共同维持其步调稳定延续。
- 坚持不懈地追求技术卓越和良好设计，敏捷能力由此增强。
- 以简洁为本，它是极力减少不必要工作量的艺术。
- 最好的架构、需求和设计出自自组织团队。
- 团队定期地反思如何能提高成效， 并依此调整自身的举止表现。

其中最重要的是，打破了原先一次性交付给用户一整个软件的概念 —— 通俗的讲，既然用户内心不清楚自己想要什么，那么我们也不会去按照我们的想法去假设。而是先给用户一个**最小可用模型**，确认一下方向性，然后再一点点完善。最重要的就是在软件开发的过程中引入了用户的参与，确保每个阶段的产物都是用户想要的结果，以保证方向不会偏离。

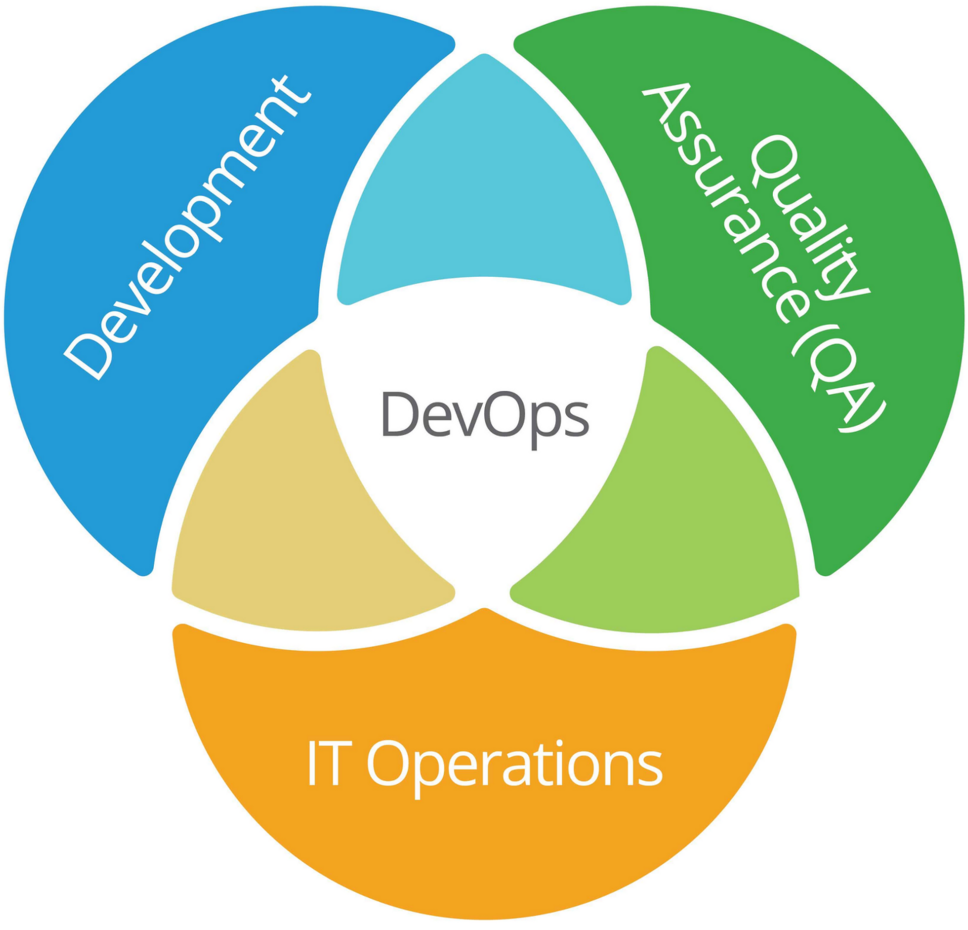
从此，在软件开发的流程中。用户与开发者会经常碰头，不断交流新的想法与意见，在思想的碰撞中诞生一些伟大的产品，此时用户也是开发团队的一员，参与到整个开发流程中。敏捷开发的内核在于拥抱变化、及时调整。虽然还有很多其他的问题，但是从大方向来看敏捷开发是一个比较好的解决方式了。

但是敏捷开发在落地过程中诞生了新的问题，频繁的变化会让团队不由把目光放在过程而不是结果中。此时，开发者们从丰田汽车制造商上获得了一些灵感。丰田汽车的生产方式有一套名为精益制造的方法论，将其变体到软件行业中，我们获得了 **精益软件开发** 七条原则:

- 消除浪费
- 增强学习
- 尽量延迟决定
- 尽快发布
- 下放权力
- 嵌入质量
- 全局优化

有兴趣的同学可以自行查阅一下相关的Wiki，在此就不过多赘述了。 而到此为止，我们解决的都是软件开发行业的事情。在此之前我们往往忽略了运维也是软件交付中非常重要的一环。是时候把运维也拉上变革的战车了。

于是, DevOps 就应运而生了。因为此时的我们需要一套方法论，将开发、测试、运维三者打通。

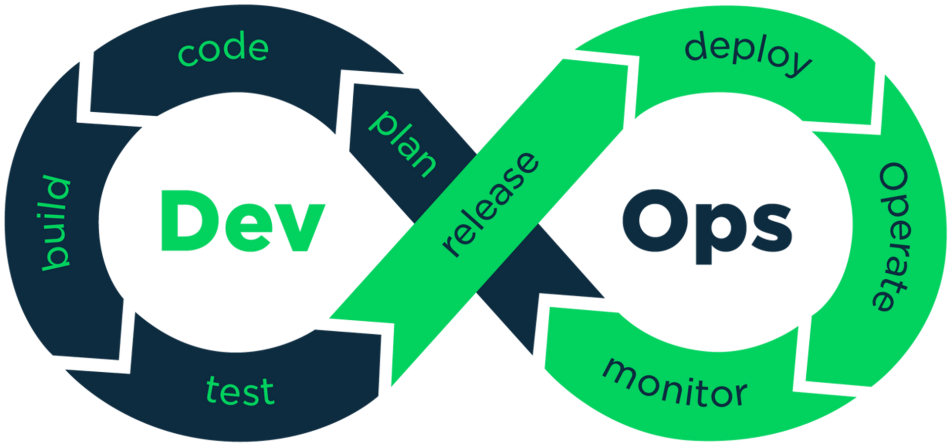


如图所示，我们将开发、测试(质量保障)、运维画成三个相互有交集的三个圆，其中三者的公共交集就是我们所说的DevOps。

我们回过头来再看一下文章开头对DevOps的定义，就更加好了解了:

> DevOps 是一套结合了软件开发 (Dev) 和 IT 运维 (Ops) 的实践方式。 它旨在缩短系统开发生命周期并提供具有高软件质量的持续交付。DevOps 与敏捷软件开发相辅相成； DevOps 的几个方面来自敏捷的工作方式。

DevOps的来源就是自敏捷开发而来，作为一种方案，可以理解为敏捷开发在之后流程的延伸。DevOps通过一些工具或者手段，将软件开发的流程在工程师的角度上实现其独有的“闭环”。



如上图所示，一个完整的软件生命周期流程闭环就是一个计划(调研) —— 编码 —— 构建 —— 测试 —— 发布 —— 部署 —— 运维 —— 监控。然后根据监控获得的数据（用户主动反馈或者用户行为分析）来进行下一轮的流程。

而DevOps 就是加速这一流程的“高速公路”。

看呐，这和敏捷开发是不是有异曲同工之妙呢？当然，现在我们谈论的都是方法论，是比较抽象的“空中楼阁”，从学术界到工程界的演进需要更加落地的实践。

接下来，我们聊聊DevOps的重要组成与落地手段 —— CICD。