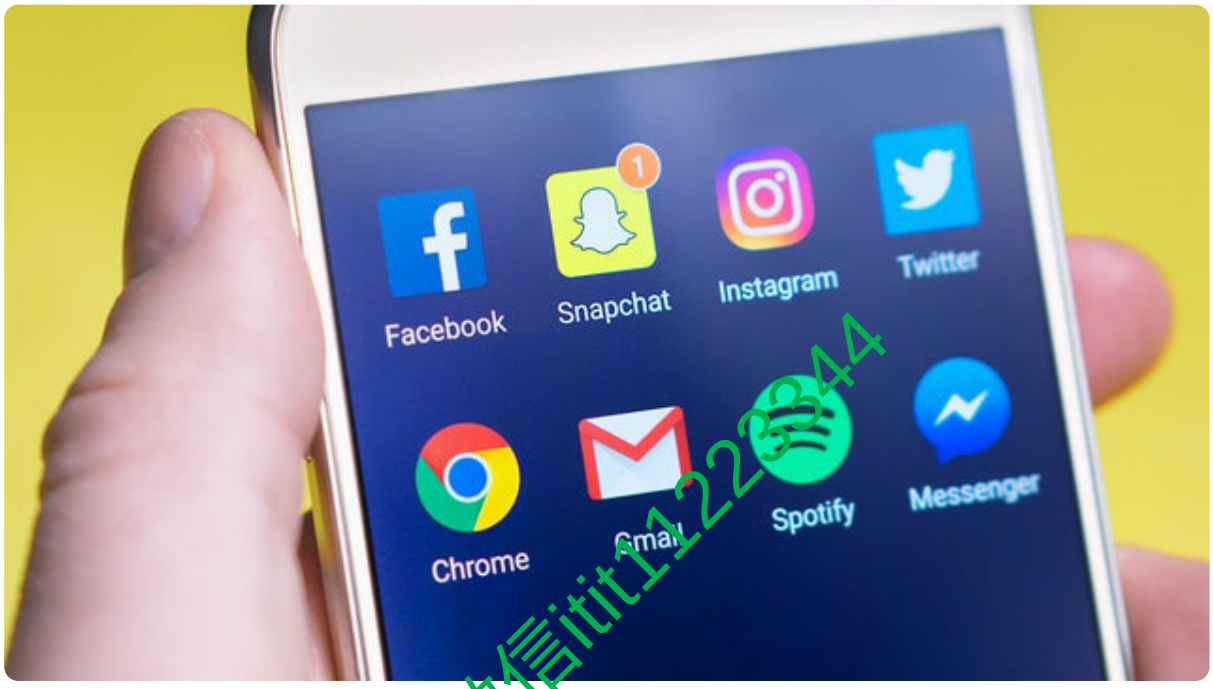
33 打破传统壁垒：容器化与TestOps

更新时间：2019-10-28 11:25:29



> 天才就是百分之二的灵感，百分之九十八的汗水。——爱迪生

记得在写 AI 测试的时候，刚好有老友到访，老友很诧异：你不是一向支持新技术么，怎么会对一个人工智能这么大的反应？我仔细想了一下，其实我并不是对新技术或者对人工智能有什么不好的情绪，毕竟再智能我也肯定不至于先抢了我的饭碗，大约让我非常有情绪的是测试人或者一些机构对于人工智能的盲目吹捧，觉得测试行业还是需要有人去泼一泼冷水，让大家走回到原本的路线上来。

但是，我对于新技术新事物，其实非但不排斥，反而是希望能够去探索去学习，择其善者而从之，其不善者而改之。所以今天我的主题是：打破传统壁垒。

传统的开发、测试、运维都是各司其职的，但是发现难免发生交集，而且这个交集还很重要，最明显的就是环境，我们就从环境聊起，像曾经我们有超过四套环境，包括归属于开发的 dev 环

境，属于测试的 test 环境和属于运维的灰度和生产环境，再加上增加的性能环境、联调环境，多个环境由于环境配置、数据库等等的不同引发了很多问题。印象比较深刻的就是几个方面：

1. 重复测试：每个环境都需要重新进行回归测试才能够放心部署运行
2. 在 A 环境中没问题的程序切换到 B 环境就不正常了
3. 由于需要配置的环境太多，复杂度太高，有了 BUG 也让开发人员难以复现和排查

于是 Docker 出现了。

```
<img src="assets/31-1.jpg" width="40%"  align="left"/>
```

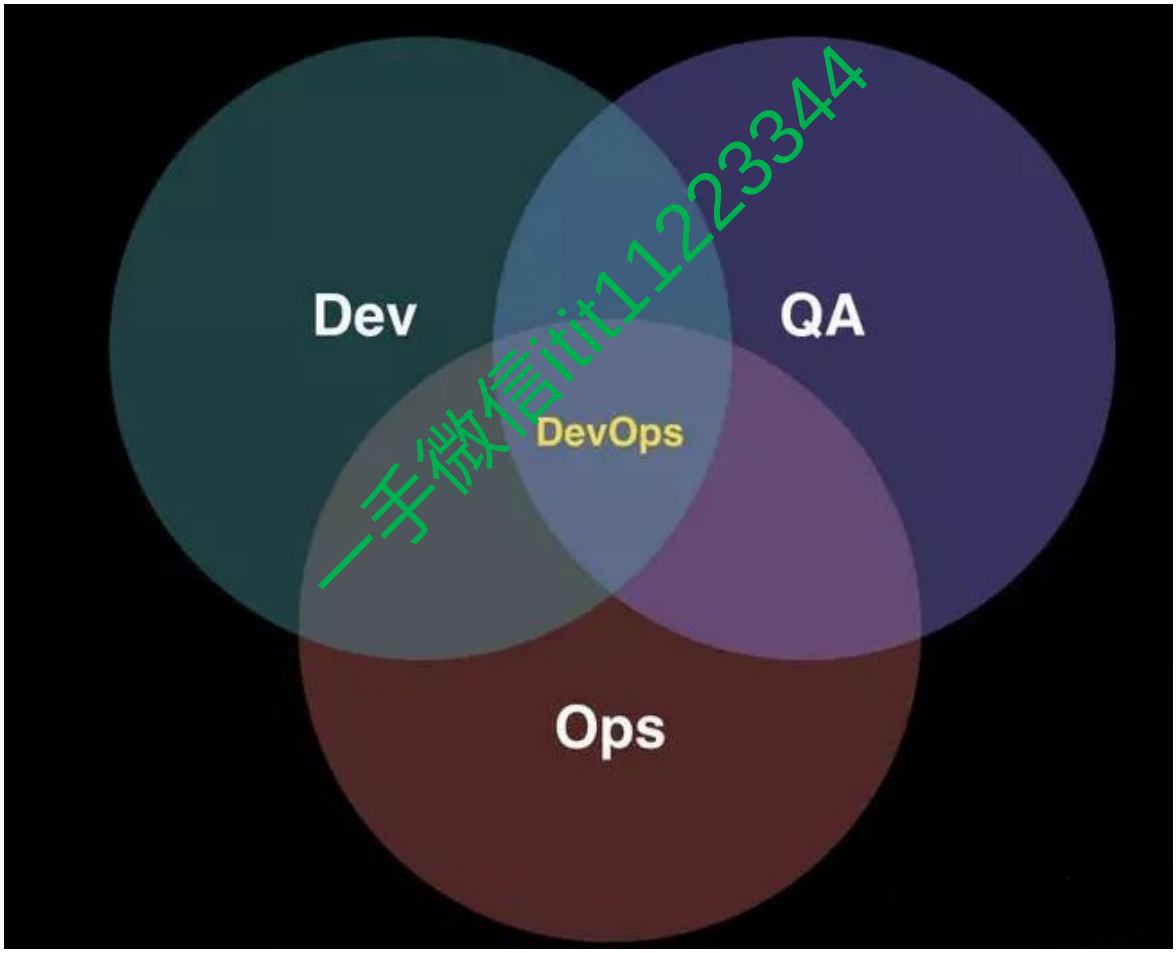
Docker 到底是什么我不想在这里跟大家细细普及，我们主要聊聊 Docker 对于测试的促进作用：

- **环境的统一**：在 Docker 下我们不用再纠结多套环境运维、管理的问题，也更加不用考虑不同环境之间的差异。 Docker 完全可以快速的部署一套干净的环境，供我们进行各项软件生命周期活动。
- **降低配置的压力**：很多时候应用会依赖于很多配置，比如数据库、防火墙、队列、缓存等等，Docker 呢，就可以通过打包镜像，将这些配置一起打包到镜像中，也进一步保证了环境的统一。
- **资源利用**：由于容器不需要进行硬件虚拟、也不需要运行完整的操作系统，所以容器的执行速度、内存消耗都要比虚拟机更高效，性能也更好。
- **问题的复现**：传统的环境配置里，生产上的问题往往很难在测试环境重现，让我们在 BUG 复现上会耽误太多的时间，更难定位问题出现的原因。而 Docker 的特殊性就在于可以快递通过镜像复制出现问题的场景，更快速的定位分析问题。同样的，在对于测试与开发的沟通上亦是如此。
- **持续集成的支持**：传统的持续集成还是停留在应用方向，而配合 Docker，通过 Docerfile 来进行镜像构建，效果更好。
- **环境的迁移**：我们经常遇到某些环境的迁移，比如机房的迁移、平台的迁移等等。传统情况下我们的部署、迁移和迁移后的测试是一个非常大的工程，而有了 Docker，基于 Docker 的兼容性，可以很轻松的进行迁移。如果再辅助后边要聊的 Devops 或者 Testops，就更加快捷和安全了。

所以，Docker 这项技术飞快的被所有的测试开发运维所接受，而配合上我们原本就有的自动化测试和精准测试，这让我们的测试体系更加立体化，更有效率。

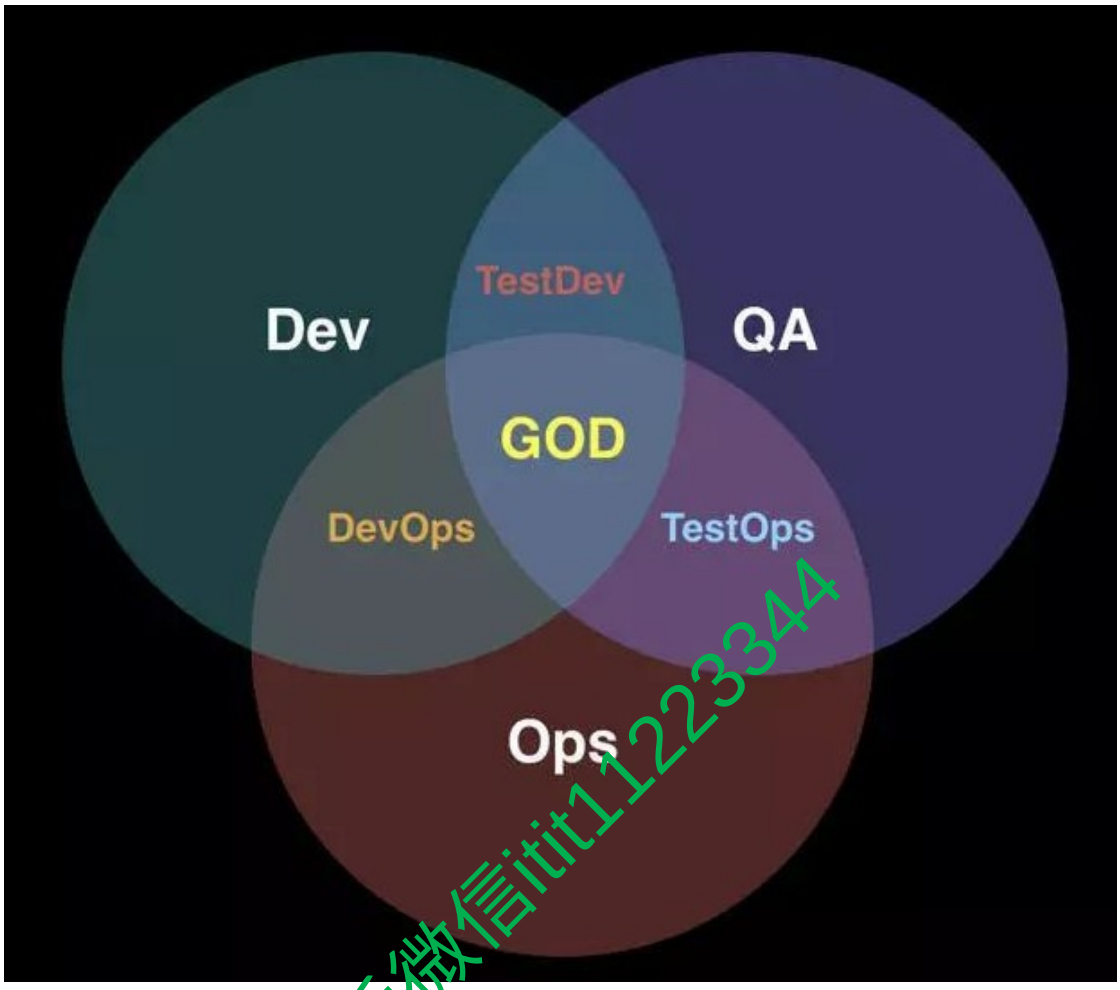
在容器化的基础上，运维的思路们也开始发生了变化。在传统方式中，开发和运维当然是各行其是的，所以也经常会阻碍向客户交付高质量的应用，这种方式某种程度上已经没办法适应高速发展和变化的互联网时代。而 DevOps 旨在消除开发和运维之间的这种不协调，加快交付速度，同时也要保证质量。DevOps 的目的就在于消除开发约束，实现自动化，并且在开发与运维之间创造反馈机制。用维基百科的解释就是：

> DevOps（英文 Development 和 Operations 的组合）是一组过程、方法与系统的统称，用于促进开发（应用程序 / 软件工程）、技术运营和质量保障（QA）部门之间的沟通、协作与整合。它的出现是由于软件行业日益清晰地认识到：为了按时交付软件产品和服务，开发和运营工作必须紧密合作。



最初的 DevOps 目的就是解决研发、测试、运维之间的那些灰度地带的事情。换句话说在我的理解中，DevOps 就是用一系列的自动化工具改进测试、研发和运维之间的工作流程，保证能够快速、持续又安全的部署。随着微服务时代的到来，指望运维通过手动去部署成百甚至上千的微服务，不现实也不安全。

当然，DevOps 不是万能的，它可以解决流程上的问题，但是不能解决质量细节的问题，所以，以质量为核心的 TestOps 就产生了。由于 TestOps 实际上是有 DevOps 衍生之下的产物，所以也是与 DevOps 成对存在，共同作用改进我们的整体架构。



TestOps 同样是高度依赖自动化的，在理想的状态下，开发提交代码后，系统开始自动触发静态检查体系，静态检查通过后，开始完成自动的单元测试，确定所有单元测试用例通过则自动打包发布到测试平台，否则将错误信息返回通知相关人员。接下来测试平台发布完成后，测试平台会进行接口及 UI 的自动化，如果都正常通过生成测试报告，如果自动化测试足够完善的话，可以直接接入到 DevOps 开始进行快速部署。

如此看来，TestOps 更多的就是对于测试在整个生命周期中的工作方式的指导，也更加强调自动化测试。为什么要这么依赖自动化呢？我们举个例子，有时候我们生产上出现了一个问题，已经造成了损失，我们不可能直接把服务下线不让客户使用，更不可能放任损失继续扩大。所以我们在第一时间定位问题，紧急修改完成后，很难再给测试两个小时、三个小时的时间去回归了，那么怎样保证这样的紧急修改不会影响其他功能呢？这就是 TestOps 的意义，尽管有时候我们为了抢出 1 个小时的时间需要完善 10 个小时的自动化代码，但是在互联网敏捷时代，这是必须

那么 DevOps 与 TestOps 的关系就是：

DevOps 推动整个测试和运维团队统一整个研发流程，帮助团队更敏捷的提交产品。 TestOps 站在专业的测试角度推动开发和运维一起辅助进行，保证质量。TestOps 和 DevOps 形成了一个完整的持续集成和持续交付体系。

这是一个大的方向，我没有介绍一些技术细节、工具应用，不然可能单单这里就要连载起几个月了，但是核心我们发现了。而对于测试来说，提升自己的综合能力才是最重要的。在这个时代，编程能力、单元测试、接口 UI 自动化、运维技术都要学，不仅仅有炒锅、原料、调味，还要会洗菜、切菜、烹饪，这样才能够把一盘好菜端上桌。不要想一下子做出满汉全席，从小处做起，持续优化。忘记了在哪里看过这么一句话，我觉得很适合作为这一趴的结束语：仰望星空，脚踏实地。

← 32 更高更快更强 - 谈精准测试

34 不打无准备之仗：如何备战面试 →

精选留言 1

欢迎在这里发表留言，作者筛选后可公开显示

土豆稀饭

docker真的是很方便了，用了才知道好

1 回复 2019-10-28

风落几番 回复 **土豆稀饭**

那必须的，用了都说好！

回复 2019-10-30 16:45:06

千学不如一看，千看不如一练