## 第一章 DevOps概述

DevOps是一种思维方式和工作方式。作为一个框架，它支持人们和团队分享经验、建立同理心，以有效而持久的方式实战使用。它也是文化组成的一部分，反映了我们要如何工作以及为什么这样做。很多人认为DevOps就是Chef或Docker之类的特定工具，不过这些工具本身并不是DevOps。工具之所以被认为是“DevOps”，关键在于它们的使用方式而不是工具本身的基本特性。

除了用来实战使用的工具，我们文化中还有一个同样重要的部分，是我们的价值观、标准和知识。通过分析人们如何工作、所使用的技术、技术如何影响我们的工作方式及人如何影响技术，可以有助于明确我们的组织以及行业的格局，做出有意义的决策。

DevOps不仅是一个软件开发方法论。虽然它与软件方法（Agile）或极限编程（XP）等软件开发方法论有关，甚至受其影响，而且DevOps实践确实也包括软件开发方法或基础设施自动化和持续交付等特性，但它不仅是上面这些部分。虽然这些概念与之相关，而且在DevOps环境中会经常看到，但是如果只关注这些部分就会一叶障目，而无法看到全貌，将忽视真正赋予DevOps强大功能的文化和人际关系等方面。

### DevOps文化快照

成功的DevOps文化是怎样的？为了说明这一点，我们将分析Etsy的员工、工作流程和工具的相互融合（Etsy是一个销售手工艺品和古董的全球在线集市）。之所以选择Etsy，不仅是因为它们的技术和文化实践在业界很知名，此外，Ryn（本书作者之一）在Etsy的工作经历可以为我们提供第一手资源，使我们能够从内部更详细地了解DevOps文化。

在Etsy，新入职的工程师第一天会拿到一个笔记本电脑，开发虚拟机(VM)已经建立了有适当访问权限和授权的账户，而且已将最常用的GitHub仓库克隆到本地，还预建了重要工具的别名和快捷键，另外电脑桌面上有一个指南，其中包括新员工的新消息以及公司其他资源的链接。由于不同团队使用的工具和实践得到了标准化，所以不论新人加入哪一个团队，都能快速进入状态，不过每个团队也有一定的灵活性，在适当情况下可以有所调整。

Etsy会安排老员工与新员工结对，老员工要带着新员工熟悉日常工作中会使用哪些测试和开发过程。首先在本地开发虚拟机上编写代码，这台VM已经做好了配置管理几乎和生产环境配置一样。开发VM设置为在本地运行和测试代码所以在这些早期阶段，他能快速地运行和修改他的代码，而不会影响其他人的开发工作。

通过运行一组本地单元测试和功能测试，Etsy工程师可以很有信心地认为他们做的修改在本地能正常工作。在此基础上，他们会在try server上测试所做的修改，这是一个Jenkins集群，与公司的生产持续集成（continuous integration，CI）集群几乎完全相同，不过还有一个额外的优点，不需要向master提交任何代码。如果通过了try测试，工程师就能更有信心地确信不会因为他们所做的修改导致任何测试失败。

取决于修改的规模和复杂性，新来的工程师可能选择新建一个拉取请求（pull request）或者不那么正式地请某个同事做代码审查。不是每一个修改都要求必须这么做，通常这由个人自行决定，Etsy提供一个高度信任的无问责环境，在这个环境中，人们得到了充分的信任和授权，可以决定是否有必要做代码审查。会为新员工或初级员工提供指导，帮助他们确定哪些修改需要代码审查，以及谁来完成代码审查。作为一个新员工，在具体部署所做的修改之前，可以先让团队伙伴做粗略的检查。

如果已经通过本地测试和try测试，开发者将会加入Etsy所称的推送队列（push queue），把他的修改部署到生产环境。当多个开发人员同时推送修改时，该队列系统使用互联网中继聊天（Internet Relay Chat，IRC）和一个IRC机器人协调部署工作。轮到这个工程师时，他提交的修改将推送到所用仓库库的 master，并使用[Deployinator](https://github.com/etsy/deployinator)把这些修改部署到QA。这会在QA服务器上自动开始构建并运行整个CI测试套件。

成功完成构建和测试后，这个开发人员对网站的QA版本及日志做一个快速地人工检查，查找自动测试没能捕捉到的问题。之后，再使用同样的Deployinator过程将代码部署到生产环境，并确保测试和日志一切正常。如果确实存在测试未能捕获的问题，可以利用大量图表盘来监视，而且Nagios监控检查会提醒人们出现了问题。另外，许多团队都有自己的轮班Nagios检查，鼓励所有人共同承担责任，保证一切平稳运行，人们会共同协作来帮助解决问题，无问责事后分析意味着人们可以从错误中学习，而不会因为犯错收到惩罚。

这个过程相当高效，从开始到完成平均只需要10分钟的时间，Etsy工程师们每天总共部署约 60次。任何人都能得到他想要仔细研读的文档，在一个团队成员的知道下，每个工程师可以迅速熟悉这个过程，第一天就能向生产环境推送代码。即使是其他员工而非工程师，也通过“第一个推送项目”（First Push Program）鼓励其参与，他们可以与工程师结对来部署一个很小的修改，如把他们的照片增加到网站员工页面。除了用于常规的软件开发，try测试和Deployinator过程非常好用，可以用于几乎任何可部署的项目，从开发人员用来构建虚拟机的工具到搜索日志的Kibana仪表盘，从Nagios监控检查到Deployinator工具本身，都可以采用这个过程。

### 文化演进

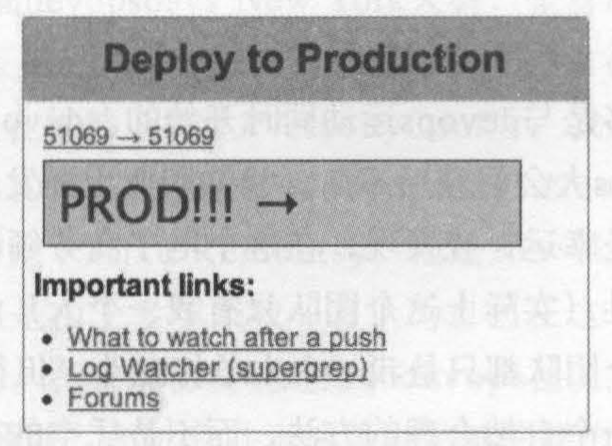
几年前在Etsy的部署是一个缺乏透明度和极易出错的流程，每次部署需要花费近4小时，这是非常鲜明的对比。开发人员在自己的刀片服务器上工作，而不是虚拟机，不过那时刀片服务器的功能不够强大，还不能完整地运行自动测试套件。在过渡环境中运行的测试需要几小时才能完成，而且这些测试很不稳定，所以测试结果不是太有用。

工程组织中的团队就像一个个孤岛，相互割据。开发与运维之间存在“隐喻墙”，大量开发人员把代码扔给墙对面的运维人员，而运维人员只负责部署和监视，所以往往很抗拒改变。开发人员编写代码，执行手写的shell脚本来创建一个新SVN分支，然后使用svn merge部署（这不算是最容易使用的合并工具），将所有开发人员的修改合并到这个部署分支。开发人员告诉运维工程师谁有权限部署哪一个要使用的分支，由此将开始一个痛苦的部署过程，可能长达几小时（见图）。由于这个过程太麻烦，所以可能两、三周才会部署一次。



人们对这一过程很厌倦。他们意识到必须要做出改变，否则部署会变得更糟。

最开始由一个运维工程师向两个开发人员分享部署领域的核心知识，让这些开发人员有时间、有基础根据他们的需要调整这个过程。他们说，一个手里头有榔头的人，看到的一切都是钉子，如果有一个Web应用开发人员，那么所有问题看起来都需要一个Web应用，第一个Deployinator就此诞生（如图）。最初这是一个包装现有shell脚本的Web包装器，不过，经过一段时间后，更多的人加入进来对它进行改进。尽管完成具体工作的底层机制有变化，但总的界面基本保持不变。



所有人都清楚地看到，通过允许这些员工创建工具来改进他们的工作，会让完成类似工作的人轻松得多。原先人们不愿意部署，认为部署很碍事，现在部署则能帮助他们达成目标，更好地为用户展示产品的特性。测试也不再是浪费时间，而是可以帮助捕捉bug。基于日志、图表和提醒，所有人（而不只是少数的几个人）都能看到他们的工作产生的影响。

关于所有这些工具的故事，我们得到的重要收获不是工具本身的特性，而是人们意识到需要构建这样一些工具，而且要提供时间和资源来构建这些工具。

一方面得到了管理层的支持，可以自由地试验和处理不直接面向客户的代码，另一方面不同团队之间的信任使得这些工具得到有效开发，而且伴随着这些工具的开发，也形成了一个融合工具、分享和协作的文化。

这些要素使Etsy称为了如今人们称道的“devops独角兽”（尽管他们更愿意把自己称作“sparkly horse”），而且维护这种文化也已经称为公司的重中之重。这个例子充分体现了我们对“有效devops”（effective devops）的定义，一个组织利用文化变革来影响个人如何考虑有效工作、评价个人的不同角色，提升业务价值，并度量改变所产生的影响。正是devops的这些原则帮助Etsy取得了现在的发展，从一个充满挫败感、相互割据的状态变成了业界知名的协作组织和工具构建者。尽管具体细节可能各不相同，但我们已经实现类似转变的组织提供一个指南。

每个人都按自己的思维方式思考，但概念是共同的。

### Ryn的故事

### Jennifer的故事

### 通过故事描述DevOps

我们选择大量案例来描述有效的devops文化会以哪些不同的方式体现。不过，这些故事的目的并不是提供一些模板，要求你严格遵循，如果盲目地复制另一个组织或个人的做法，这会忽视他们做出选择时的具体环境和必要理由。

这些故事只是一些实力或指南。希望你能好好读一读这些故事，并对你的经验进行反思。可能是你现在的经验，也可能是将来得到的经验。我们提供了来自不同方面的很多故事，包括正式地案例研究以及非正式的个人故事。有些故事来自比较知名的组织，不过我们还有意地从一些不太有名的组织选择了一些故事，以便向你展示多种不同的devops形式。

阅读案例时，不仅要考虑所做的选择和相应的结果，还要考虑它们的具体环境和情境。哪些环境与你自己的环境有哪些类似的地方，另外有哪些关键的差别？如果在你自己的组织中做同样的选择，你的工作环境中的哪些特定因素可能会改变结果？通过阅读和理解这些故事，我们希望你能发现他们深层次的主题，并把它们应用到你自己的devops故事中。

只读这些故事还不够，我们的学习不能就此止步，还要具体实验新的过程、工具、技术和理念。要度量你的进展，最重要的是，要理解相应的原因。一旦认识到哪些尝试适用额哪些不适用，就可以开始做更复杂的试验了。

## 第二章 什么是DevOps

devops是一种文化运动，会改变个人对其工作的想法，重新所做工作的多样性，支持加快企业实现价值的过程，以及度量社会和技术改变带来的影响。这是一种思维方式，也是一种工作方式，会促使个人与组织发展和维护可持续的方式实战运用。

### 文化处方

devops是一个文化处方。任何一个文化运动都不可能存在于真空中；社会结构与文化紧密交织在一起。组织中的层次结构、行业关联以及全球化都会影响文化，还会影响价值观、标准、信念以及反映这些方面的工件。我们创建的软件不能脱离使用和创建这些软件的人独立存在。devops就是要找出方法来调整和变革社会结构、文化和技术，从而能更有效地工作。

### DevOps是多方面的综合

一个标榜自己很新的运动往往有一个危险，它可能视图涵盖所有一切新东西。

实现devops并没有一种唯一正确的方法，这本书也不是这样一个“单一处方”。尽管我们会介绍常见的误区和反模式，不过更有兴趣描述成功的devops文化看上去是怎样的，会有怎样的表现，以及这些原则如何应用到各种不同的组织和环境。

尽管devops这个词本身是“development”（开发）和“operation”（运维）合成的一个词，但devops的核心概念可以（而且）应当应用到整个组织。一个可持续的、成功的企业并不只有开发和运维团队。如果加以限制，只考虑具体写软件或者在生产环境部署软件的团队，这回对整个企业带来危害。

### DevOps作为大众模式

在很多方面，devops已经成为一个大众模式，这个词有不同的含义，相应地也存在一些误解。在认知科学领域，大众模式往往作为一个抽象用来表述更具体的想法，而且通常会替代具体想法，与最后讨论的概念相比，这个抽象更容易理解。情境意识就是这样一个例子，这通常用来表示更特定的概念，如感知和短期记忆。大众模式不一定不好，但是不同的组织使用同一个术语却有着不同的内涵时，这就会很成问题。

人们常常花更多的时间争论“devops”是指什么（用来表示怎样的大众模式），而不是花更多的时间关注他们真正想要讨论的想法。有时，为了绕开如何定义devops的问题，人们讨论具体的概念和原则时，会夸大“不好的”行为，由此强调哪些才是“devops”的“好”行为。例如，为了讨论有效的团队间协作，有人可能会用一个漫画例子来说明，展示在一个建立了devops团队的公司中，devops团队只是作为开发和运维团队的中间人（就像我们在前言中所提到的）。这是一个极端的例子，但是这会让人们讨论比定义更有意义、更实用的内容。

### 老观点和新观点

如果有人犯错误就受到责备和惩罚，在这样一个环境中，会发展一种恐惧文化，这会建起高墙，阻碍清晰、透明的沟通。与之相反的是无问责环境，在这种环境中，问题将通过协作的方式加以解决，而且会把问题看作是个人和组织的学习机会。Sidney Dekker教授在他的《The Field Guide to Understanding Human Error》一书中把这两种环境描述为人为错误地“老观点”和“新观点”。

第一种环境把“人为错误看作是问题的根源”。这种“老观点”可以描述为一种思维模式，其重点在于消除人为错误。错误是“坏苹果”造成的，应当把这些坏苹果扔掉。这种观点常见于问责文化中，因为它认为错误通常是由于恶意或者能力不足造成的。对失败负责的个人必须受到责罚（或者直接开除）。