倾情奉献

IBM

DevOps

for dummies

IBM 限量版，第 3 版

DevOps 的业务需求和价值

DevOps 功能和采用途径

云技术如何加速 DevOps

Sanjeev Sharma
Bernie Coyne

DevOps

for dummies®

DevOps

IBM 限量版，第 3 版

作者：Sanjeev Sharma 和 Bernie Coyne

for dummies®

DevOps For Dummies®，IBM 限量版，第 3 版

出版商：
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN：978-1-119-52345-1 (pbk)；ISBN：978-1-119-52343-7 (ebk)

美国印刷

10 9 8 7 6 5 4 3 2 1

目录

出版商致谢

感谢以下人员为将本书推向市场提供大力帮助和支持：

项目编辑：Carrie A. Burchfield

业务开发代表：Sue Blessing

高级策划编辑：Katie Mohr

制作编辑：Vasanth Koilraj

编辑部经理：Rev Mengle

导言

DevOps 是 development（开发）和 operations（运维）的缩写，像大多数新方法一样，在许多人眼中不过是一个时髦词儿。人们都在讨论 DevOps，但并非人人都知道其中的含义。从广义上讲，DevOps 是一种基于精益和敏捷原则的方法，业务、开发、运维以及质量保证部门运用此方法可以协作持续交付软件，支持企业更快速地抓住市场机遇并缩短融合客户反馈的时间。事实上，企业应用程序十分多样化，由多种技术、数据库、最终用户设备等组成，只有 DevOps 方法能够成功解决这些复杂难题。但是，人们在如何运用 DevOps 方面众说纷纭。

有人说 DevOps 仅适用于从业人员，还有人说它以云为依托。IBM 从广泛而全面的视角出发，认为 DevOps 是一种业务驱动的软件交付方法，这种方法带来了一种覆盖从创意到生产阶段的全新或增强的业务能力，通过这种能力，企业可以高效地向客户提供业务价值，并收集客户反馈。要做到这一点，您不仅需要开发和运维团队的参与，还需要让利益相关方参与其中。真正的 DevOps 方法需要业务部门、从业人员、高管人员、合作伙伴、供应商等配合完成。

本书简介

本书将从以企业为中心的视角介绍 DevOps。当今世界瞬息万变，所有企业必须具备足够的敏捷性和精益性，快速应对客户需求、竞争压力或法规要求等方面的变化。在这种背景下，DevOps 方法成为了所有企业的必备“法器”。

如果您正在阅读本书，想必一定听闻过 DevOps，同时希望能够深入了解其中内涵以及自己的企业如何从中获得业务优势。本书面向以下类型的高管、决策者和从业人员：他们刚刚接触 DevOps 领域，期望了解

有关这种方法的更多信息，并希望透过围绕这个观念的繁杂炒作而触及本质。

本书中的图标

在本书页边空白处，您会看到一些图标。以下是这些图标的含义。

“提示”图标表示有关 DevOps 各方面的有用信息。

任何带有“牢记”的图标表示您希望记住的信息。

“警告”图标提醒您这是关键信息。

“技术内容”超出 DevOps 基础知识的范畴。非必读内容。

参考资料

您可通过访问以下网站，了解有关 DevOps 以及 IBM 的方法和服务的更多信息：

- **IBM DevOps 解决方案：** ibm.com/devops
- **DevOps — IBM 方法（白皮书）：** ibm.biz/BdEnBz
- **软件优势（调研）：** ibm.co/156KdoO
- **采用 IBM DevOps 方法（文章）：** ibm.biz/adoptingdevops
- **DevOps Services for Bluemix（服务）：** bluemix.net

本章概要

» 了解 DevOps 的业务需求

» 认识 DevOps 的业务价值

» 理解 DevOps 原则

第 1 章
什么是 DevOps?

对日常业务运营做出改变总是困难重重，通常需要投入时间和精力进行研究。每当企业采用新技术、新方法或新策略时，那一定是出于业务需要。要开发业务案例来支持采用 DevOps，企业必须了解 DevOps 的业务需求，包括这种方法可以解决的挑战。本章将向您介绍构建业务案例所需的基础。

了解 DevOps 的业务需求

企业希望通过打造创新的应用程序和服务来解决业务问题。他们可能希望解决内部业务问题（例如开发更好的客户关系管理系统），或者是为他们的客户或最终用户提供帮助（例如提供新的移动应用程序）。

然而，许多企业在软件项目方面并不成功，失败原因常常与软件开发和交付方面的难题有关。虽然大多数企业认为软件开发与交付至关重要，但是 IBM 最近的一项调查显示，只有 25% 的企业认为他们的团队在这个方面卓有成效。这种执行缺口常会导致企业错失商机。

与此同时，企业所需交付的应用程序类型发生了重大变化，从记录系统转变为互动系统，这导致问题进一步恶化。

- **记录系统：**传统的软件应用程序都是大型系统，起记录系统的作用。这些系统中包含海量数据和/或事务，目的是确保高度稳定、可靠。因为这些应用程序不需要经常修改，所以组织每年只需要发布一两个新的大版本就可以满足客户以及他们自己的业务需求。
- **互动系统：**随着移动通信的到来以及网页应用程序的发展成熟，互动系统正在成为记录系统的补充，而客户可直接访问互动系统，与企业进行交互。此类应用程序必须要简单易用，具有高性能，并能够快速修改，以满足瞬息万变的客户需求并支持不断演进的市场力量。

由于互动系统直接供客户使用，因此需要高度关注用户体验，实现高速度和敏捷性 — 换句话说，就是 DevOps 方法。


记住

互动系统并非单独存在，往往与记录系统相互交织，因此互动系统的快速修改势必会引起记录系统的调整。实际上，任何需要快速交付创新成果的系统类型都需要 DevOps 的支持。此类创新的主要推动力量是新兴技术趋势，例如云计算、移动应用程序、大数据和社交媒体，这些趋势可能会影响所有类型的系统。我们将在第 4 章和第 5 章中围绕 DevOps 探讨这些新兴技术。

认识 DevOps 的业务价值

DevOps 在整个软件供应链中运用敏捷和精益原则。DevOps 支持企业最大限度地提高产品或服务的交付速度，覆盖从初步创意、生产发布、客户反馈到基于反馈实施改进的各个环节。

记住

DevOps 可以改进企业向客户、供应商和合作伙伴交付价值的方式，因而成为一种不可或缺的业务流程，而不仅仅是一种 IT 能力。

DevOps 可在以下三个方面带来显著投资回报：

- 增强客户体验
- 提高创新能力
- 加速实现价值

我们将在下文逐一讨论这三个方面。

增强客户体验

提供增强（即独具特色、引人入胜）的客户体验，可以提高客户忠诚度并扩大市场份额。为了营造这种体验，企业必须持续收集并响应客户反馈。这需要一种机制，从交付的软件应用程序所涉及的所有利益相关方那里快速获取反馈，包括客户、业务部门、用户、供应商、合作伙伴等。

在如今的互动系统（请参阅本章前面的“了解 DevOps 的业务需求”）世界中，如果企业能够敏捷地做出响应和改变，势必能够提升客户体验和忠诚度。

提高创新能力

现代企业纷纷利用精益思维方式来提高自身的创新能力。他们的目标是减少浪费和返工，并将资源投入到价值更高的活动中。

技术内容

*A-B 测试*是一个常见的精益思维实践示例。在测试过程中，企业要求一小组用户对两套或多套具有不同功能的软件进行测试和评分。然后，功能更佳的软件将会推广给所有用户，而不成功的版本将会就此终止。这种 A-B 测试只有通过高效的自动化机制（如 DevOps 支持的机制）才能实现。

加速实现价值

加速实现价值需要创造一种文化、实践和自动化机制来支持实现快速、高效且可靠的软件交付和生产。作为一种业务能力使用时，DevOps 能提供所需的工具和文化，帮助实现高效的发布规划、可预测能力和成功交付。

*价值*的含义因组织甚至因项目而异，但 DevOps 的目标始终是更快速、更高效地交付价值。

了解 DevOps 的运作方式

DevOps 运动浪潮衍生出一些不断演变的原则。而 IBM 等几家解决方案供应商开发出了自己的原则。不过，所有这些原则都采取了整体性的 DevOps 方法，适用于各种规模的企业。这些原则是

- 针对类生产系统进行开发和测试
- 利用可重复的可靠流程进行部署
- 监控并验证运维质量
- 放大反馈回路

我们将在以下章节对这些原则进行详细介绍。

针对类生产系统进行开发和测试

这一原则源于 DevOps 的左移概念，旨在使运维问题在软件交付生命周期早期阶段暴露出来，更靠近开发阶段（见图 1-1）。

这一概念的目标是允许开发和质量保证团队 (QA) 依据类似于生产系统的系统进行开发和测试，确保他们可以在应用程序准备投入部署之前及早了解应用程序的行为和性能。

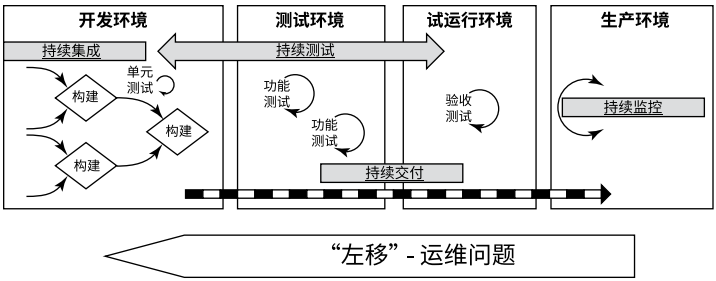


图 1-1： 左移概念可让运维问题在开发生命周期早期阶段暴露出来。

提示

为解决两大潜在挑战，应在生命周期中尽早将应用程序投入类生产系统中。首先，这支持在接近实际生产环境的环境中对应用程序进行测试；其次，这可以让应用程序交付流程本身提前接受测试和验证。

从运营角度来看，这一原则也具有极大的价值。它让运维团队可以在软件开发周期的早期阶段看到他们的环境在支持应用程序方面的表现，从而让他们创造一种经过优化的应用程序感知型环境。

利用可重复的可靠流程进行部署

顾名思义，这个原则可帮助开发和运维团队实现一个覆盖生产环节的敏捷（或至少迭代）式软件开发流程。自动化对于创建迭代、频繁、可重复和可靠的流程至关重要，因此企业必须创建一个交付管道，以实现持续的自动部署和测试。关于交付管道，我们将在第 3 章详细讲解。

提示

频繁部署还可让团队测试部署流程本身，从而降低发布时部署失败的风险。

监控并验证运维质量

企业通常善于监控生产环境中的应用程序和系统，也因此拥有很多可实时捕捉生产系统指标的工具。但是他们的监控方式彼此孤立、互不关联。这一原则通过要求在生命周期中的早期阶段经常执行自动化测试，以便监控应用程序的功能性和非功能性特征，从而将监控活动移至生命周期的早期阶段。每当部署和测试应用程序时，都应捕捉并分析质量指标。频繁的监控可以提前预示生产环境中可能发生的运维问题和质量问题。


记住

这些指标应该通过一种所有业务利益相关方都能理解和使用的格式进行收集。

放大反馈回路

DevOps 的目标之一是支持企业更快速地做出响应和调整。在软件交付中，这一目标需要企业快速获得反馈，然后从每次采取的行动中汲取经验教训。这一原则要求企业建立沟通渠道，允许所有利益相关方访问反馈并据此采取行动。

- 开发部门可以通过调整其项目计划或优先事项发挥作用。
- 生产部门可以通过改善生产环境发挥作用。
- 业务部门可以通过修改其发布计划发挥作用。

本章概要

» 了解 DevOps 的参考架构

» 思考四种 DevOps 采用方法

第 2 章
了解 DevOps 的功能

DevOps 的能力非常广泛，遍布在整个软件交付生命周期中。企业从何处着手部署 DevOps 取决于自己的业务目标，包括正在努力应对的挑战，以及需要弥补的软件交付能力短板。

在本章中，我们将探讨 DevOps 参考架构及该架构下企业使用 DevOps 的各种方式。

DevOps 采用方法

*参考架构*通过使用一组首选的方法和功能，提供一个经过验证的解决方案模板。本章讨论的 DevOps 参考架构可帮助从业人员访问和使用他们所需的指南、指令和其他材料，以构建或设计一个可满足人员、流程和技术需求的 DevOps 平台（参见第 3 章）。

参考架构通过各种组件提供功能。这些功能可能由单个组件提供，也可能由一组组件协同提供。因此，您可以从该架构提供的核心功能角度出发，了解图 2-1 中所示的 DevOps 参考架构。随着架构的形式由抽象转

为具体，这些功能的实现依托于一系列得到有效支持的人员、明确的实践和自动化工具。

图 2-1 中所示的 DevOps 参考架构提供了四种采用方法：

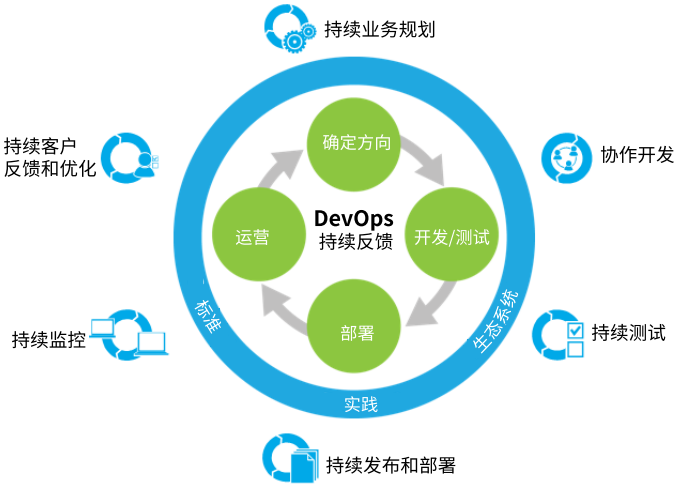


图 2-1： DevOps 参考架构。

- 确定方向
- 开发/测试
- 部署
- 运营

在接下来的部分，我们将分别详细介绍这四种采用方法。

确定方向

这种采用途径包含一种实践，重点在于建立业务目标，并根据客户反馈进行调整：*持续业务规划*。

企业需要足够敏捷，才能快速响应客户反馈。实现目标的关键在于组织正确做事的能力。遗憾的是，传统的产品交付方式无法跟上目前开展业务的速度，部分原因在于这些方法取决于定制开发和手动处理以及团队的单打独斗。快速规划和重新规划所需的信息，以及最大程度实现价值的能力，都呈现出碎片化和不一致的状态。通常无法尽早获得适当的反馈意见，因此无法借助高质量的洞察真正实现价值。

在整合各种反馈信息，帮助明智地确定投资优先级，然后作为组织开展合作，以推动持续交付模式方面，团队也显得勉为其难。一些团队并未将规划活动视为有助于加快实现价值的活动，而是视为侵入性的监管负担，觉得这会让其放慢速度。

快速交付提供了更高的业务敏捷性，但您必须在完全相信自己能够交付正确内容的前提下，加快执行速度。如果您不相信业务目标、测量结果和平台的准确性，就无法快速交付软件。

DevOps 有助于协调这些相互存在竞争关系的视角，帮助各个团队共同建立业务目标，并根据客户反馈持续对目标进行调整，从而提高敏捷性和业务成果。同时，企业还需要控制成本。通过发现并消除开发过程中的浪费现象，团队不但可以提高效率，还能降低成本。这种方法可以帮助团队在转向持续交付模式的过程中，确保在 DevOps 生命周期所有阶段的所有考虑因素之间实现最佳平衡。

开发/测试

这种采用方法涉及两种实践：协作开发和持续测试。正因如此，这种方法构成了开发和质量保证功能的核心。

协作开发

企业的软件交付工作涉及多个跨职能团队，包括业务部门负责人、业务分析师、企业和软件架构师、开发人员、质量保证从业人员、运维人员、安全专家、供应商以及合作伙伴。这些团队的从业人员在多个平台上工作，并可能分散在多个地点。*协作开发*通过提供一套可用于开发和交付软件的通用实践和一个通用平台，支持这些从业人员协同合作。

协作开发的一项核心功能是*持续集成*（见图 2-2），在这项实践中，软件开发人员可持续或频繁地将其工作成果与开发团队其他成员的成果进行集成。

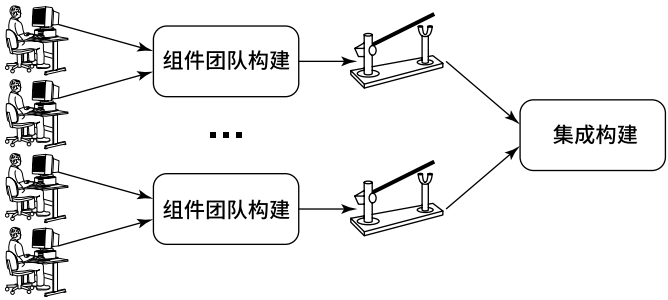


图 2-2： 通过持续集成进行协作。

持续集成的流行得益于敏捷软件开发运动。具体思路是让开发人员定期将他们的工作成果与团队中其他开发人员的工作成果进行集成，然后对集成后的成果进行测试。对于由多个系统或服务组成的复杂系统，开发人员也会定期将其工作成果与其他系统和服务进行集成。对工作结果进行定期集成有利于提前发现和暴露集成风险。在复杂的系统中，这种做法还可以提前暴露与技术或时间表相关的已知和未知风险。

持续测试

持续集成（参见上一节）有几个目标：

- 支持持续的代码测试和验证
- 验证已编写的代码和与其他开发人员的代码进行集成的效果，以及应用程序的其他组件在功能和性能方面是否符合设计
- 持续测试正在开发中的应用程序

*持续测试*意味着在生命周期中提早进行持续测试，从而降低成本、缩短测试周期并实现持续的质量反馈。这个过程也被称为*左移测试*，它强调集成开发和测试活动，以确保尽可能早地在生命周期内确保质量，而不是延后。这种方法可以通过采用自动化测试和服务虚拟化等功能来实现。服务虚拟化是一种模拟类生产环境的新功能，使持续测试成为可能。

部署

“部署”采用途径衍生出了 DevOps 的大多数基本功能。持续发布和持续部署将持续集成概念推进一步。支持发布和部署方法的实践也支持创建交付管道（参见第 3 章）。交付管道有助于企业通过高效的自动化方式，将软件持续部署到质量保证环境和生产环境中。持续发布和持续部署的目标是尽快向客户和用户发布新功能。


技术内容

构成 DevOps 技术核心的多数工具和流程都可以促进持续集成、持续发布和持续部署。我们将在后续章节详细讨论这些话题。

运营

“运营”采用途径包括两种方法，支持企业监控发布的应用程序在生产环境中的表现以及收集客户反馈。这些数据可帮助企业敏捷地做出响应，并根据需要调整业务计划。

持续监控

*持续监控*可以在交付周期的不同阶段向运维、质量保证、开发、业务部门人员和其他利益相关方提供有关应用程序的数据和指标。

这些指标不仅限于生产环境，它们可帮助利益相关方通过增强或调整正在交付的功能和/或交付这些功能所需的业务计划，进而做出响应。

持续客户反馈和优化

软件交付团队可获得两类最重要的信息，一类是关于客户如何使用软件的数据，另一类是客户就软件使用体验所提供的反馈。在新技术的支持下，企业能够在客户使用应用程序的过程中及时捕捉客户行为和痛点。这种反馈可帮助不同的利益相关方采取适当的行动来改善应用程序和客户体验。业务部门可以调整他们的业务计划，开发团队可以调整所交付的功能，而运维团队可以改进应用程序部署环境。这种持续反馈环路是 DevOps 的一个重要组成部分，可以让企业变得更加敏捷，更快的响应客户的需求。

本章概要

- 提高人员效率
- 精简流程
- 选择合适的工具

第 3 章
采用 DevOps

通常，采用任何一种新功能都需要制定计划，综合考虑人员、流程和技术等。如果不能从所有这三个方面考虑要采用的新功能，那注定会失败，对于拥有多个分布在不同位置的利益相关方的企业尤其如此。

在本章中，我们将从人员、流程和技术方面探讨 DevOps。

虽然从字面意思来看，DevOps 是指开发和运维能力，但 DevOps 实际上一种是企业能力，囊括业务部门、架构部门、设计部门、开发部门、质量保证部门、运维部门、安全部门、合作伙伴和供应商等所有利益相关方。排除任何内部或外部的利益相关方都会导致 DevOps 的实施残缺不全。

了解切入点

本节提供有关如何开始使用 DevOps 的指导，包括建立合适的文化，确定业务挑战，以及发现要消除的瓶颈。

确定业务目标

文化建设工作的首要任务是让所有人都朝着同一个方向前进、朝着同一目标努力，这意味着要为团队和整个公司确定共同的业务目标。根据业务成果，而非相互冲突的团队激励计划，来激励整个团队至关重要。如果人们了解他们努力奋斗的共同目标及工作进展衡量方法，那么拥有各自优先任务计划的团队或从业人员就会极少遇到冲突挑战。

DevOps 并非目标，而是帮助实现目标的方法。

第 4 章和第 5 章将重点介绍 DevOps 可以解决的几项新业务挑战。您的企业可以将这些挑战作为起点，帮助确定您想要实现的目标；然后，您可以针对这些目标创建一系列的通用里程碑，以供不同利益相关方使用。

确定交付管道的瓶颈

交付管道中效率低下的最大原因包括：

- 不必要的开销（反复传达相同的信息和知识）
- 不必要的返工（在测试或生产环境中未发现的缺陷，导致任务重新交给开发团队返工）
- 过度生产（开发不需要的功能）

交付管道中最大的瓶颈之一是部署基础架构。采用 DevOps 方法有助于提高应用程序交付的速度，但会对基础架构造成压力，因为需要更快的响应速度。这就是软件定义环境的用武之地，使您能够将基础架构作为一种可编程和可重复的模式，从而加快部署。请阅读本章后面的“基础架构即代码”部分以了解更多信息。

进一步深入研究后，您可能希望实现平均的端到端工作流，从而优化管道。每个流程的吞吐量必须相同，才能避免积压。为实现该平衡，需要在关键点对交付管道进行测量，以便最大程度减少积压队列中的等待时间，优化正在进行的工作并调整产能和工作流。

DevOps 中的人员

本节将介绍 DevOps 的人员要素，包括营造必要的文化氛围问题。

DevOps 文化

从根本上看，DevOps 是一种以人为中心的文化运动。组织可采用最高效的流程或自动化工具，但是如果最终没有人来执行这些流程和使用这些工具，那么它们就毫无用处。因此，建设 DevOps 文化是采用 DevOps 方法的核心。


牢记

DevOps 文化的特点是跨角色的高度协作，注重业务目标（而非部门目标）、相互信任以及通过实验进行学习。

衡量文化

文化衡量工作极其困难。您如何准确地衡量团队的协作能力是否得到增强或团队士气是否得到提高？您可以通过调查来直接衡量员工的工作态度和团队士气，但由于团队规模较小，调查的统计误差率可能较高。

相反，您可以进行间接衡量，具体来说就是，跟踪开发团队成员联系运维或质量保证团队成员来解决问题的频率，而不是通过正式渠道或多个管理层。

在利益相关方之间进行协作和沟通便是 DevOps 文化。

建设企业文化不同于采用流程或工具，这需要（姑且说“需要”）在人人皆有独特个性、不同经历和偏好倾向的团队成员之间实施社会工程。这种多元化让文化建设工作充满了困难和挑战。


提示

“扩展敏捷框架”(SAFe)、“规则敏捷交付”(DAD) 以及 Scrum 等精益和敏捷转型实践是 DevOps 的核心，如果您的企业已经应用了这些实践，那您可以借此来帮助建设 DevOps 文化。


提示

DevOps 文化建设工作的下一步就是，企业领导者要与其团队一起努力创建一个协作共享的文化环境。领导者必须消除自己强加的任何协作障碍。典型的举措是奖励运维团队在正常运行时间和运营稳定性方面所做的贡献，奖励开发团队交付的新功能，不过，这会导致各个团队之间形成对立竞争关系。举例来说，运维团队深知，保护生产活动的最佳方式就是不接受任何变更，而开发团队基本上没有关注质量的动力。所以，用责任共担机制来替换这些举措才能确保快速安全地交付新功能。

企业领导者应该通过提高透明度来进一步鼓励协作。创建一套通用的协作工具至关重要，对于因分散各地而不能当面协作的团队尤其如此。让所有利益相关方都能了解项目的目标和状态，这对于建设以信任和协作为基础的 DevOps 文化至关重要。


警告

有时，建设 DevOps 文化需要人们做出改变。对于那些不愿改变的人员 – 也就是不愿意采用 DevOps 文化的人员，企业需要对他们重新分配工作。

DevOps 团队

自 DevOps 概念诞生以来，支持和反对成立独立 DevOps 团队的声音一直不绝于耳。Netflix 等企业根本没有设立单独的开发和运维团队，而是让“NoOps”团队担负起这两项职责。其他企业通过建立 DevOps 联络团队实现了成功，解决了所有冲突并推进了协作。这样的团队可能

是一个现有的工具小组或流程小组，也可能是与应用程序交付相关的所有团队派出的代表所组建的新团队。

如果您选择组建一个 DevOps 团队，那么最重要的目标是确保这个团队发挥“卓越中心”的作用，积极促进协作，而不是新添了一层新的官僚机构，也不应将其打造成可以独自解决所有 DevOps 相关问题的团队，因为这种做法与建设 DevOps 文化的目标背道而驰。

DevOps 中的流程

在前一节，我们探讨了人员和文化在 DevOps 采用过程中的作用。流程可以界定人员的工作内容。您的企业可能已经拥有卓越的协作文化，但是如果员工对工作内容理解错误，或者做事方式不当，失败仍然不可避免。

DevOps 包含很多流程，由于数量过多，本书将不逐一介绍。本节将探讨企业广泛采用的一些关键流程。

DevOps 即业务流程

作为一种可影响全局的能力，DevOps 可提高企业的敏捷性，并改善企业向客户交付功能的能力。此外，还可将 DevOps 视为一种业务流程来进一步扩展，即作为一组活动或任务，为客户带来特定结果（服务或产品）。

在第 2 章介绍的参考架构中，DevOps 业务流程包含了来自创意（通常由业务负责人确定）、开发、测试和生产方面的力量。

虽然这个业务流程还不够成熟，无法汇集成一套简单的过程流，但您应该关注企业已经用于交付功能的过程流。然后，您就可以通过改进流程本身和引入自动化技术（参见本章后面的“DevOps 中的技术”一节），发现有待改善的方面。

变更管理流程

*变更管理*是指一系列旨在通过识别可能发生变更的工作产品以及用于实施变更的流程来控制、管理和跟踪变更的活动。企业所使用的变更管理流程是广泛的 DevOps 过程流的一个固有组成部分。变更管理可以推动 DevOps 流程收集和响应变更请求和客户反馈的方式发生改变。

采用应用程序生命周期管理（ALM）流程的企业已经具备了定义明确且（可能）自动化的变更管理流程。

变更管理应包含可支持下述功能的流程：

- 工作项管理
- 可配置的工作项工作流
- 项目配置管理
- 规划（敏捷和迭代）
- 基于角色的工件访问控制

传统的变更管理方法往往局限于变更请求或缺陷管理，而追踪变更请求或缺陷与相关代码或要求之间事件的能力有限。这些传统的方法不能提供跨整个生命周期的一站式工作项管理功能，也不具备追踪各类资源的内置功能。然而，DevOps 要求所有利益相关方能够在整个软件开发生命周期中查看所有变更并就此开展协作。

以 DevOps 或应用生命周期管理 (ALM) 为中心的变更管理包括为所有项目、任务和相关资产提供工作项管理的流程，而不仅仅是覆盖那些受变更请求或缺陷影响的项目、任务和资产。此外，还包括支持企业将工作项关联到所有工件、项目资产和其他工作项的流程，所谓的其他工作项是指由任何相关从业人员创建、修改、引用或删除的项目。这些流程可让团队成员基于角色访问所有变更相关信息，并支持迭代和敏捷项目开发工作。

DevOps 方法

以下是您采用 DevOps 时需要注意的的几个具体方法：

- 持续改进
- 发布规划
- 持续集成
- 持续交付
- 持续测试
- 持续监控和反馈

以下章节将对这些方法进行详细介绍。

持续改进

在真正的精益思维方式中，流程采用并非一蹴而就，而是一个持续的过程。企业应具备内置的流程，以便企业随着自身不断发展成熟以及从所采用的流程中汲取经验教训，找到需要改进的地方。许多企业都拥有流程改进团队，负责根据观察结果和经验教训改进流程；而其他企业则允许采用流程的团队进行自我评估，并确定他们自己的流程改进方法。无论使用何种方法，目标都是实现持续改进。

发布规划

发布规划是一项关键业务功能，由业务需求驱动，为客户提供需要的功能和满足这些需求的日程表。因此，企业需要定义明确的发布规划和管理流程，用于推进发布路线图、项目计划和交付日程表，同时提升跨流程的端到端可跟踪性。

为完成这项任务，如今大多数企业都通过使用电子表格和与所有利益相关方举行会议（通常时间漫长），来跟踪所有正在开发的业务需求应用程序及其开发状态和发布计划。但是，定义明确的流程和自动化工具可

以消除对电子表格和会议的需要，并实现精简、（更重要的是）可预测的发布模式。利用精益和敏捷实践还有助于实现更小、更频繁的发布，从而加大对质量的关注力度。

持续集成

持续集成（如第 2 章中所述）为 DevOps 带来了巨大的价值，让大型开发团队能够在多个地点共同处理跨技术组件，从而实现敏捷的软件交付。它还可以确保每个团队的工作成果与其他开发团队的成果实现持续集成和验证。因此，持续集成可以降低风险，并在软件开发生命周期中提早发现问题。

持续交付

持续集成自然会促成持续交付的实现。持续交付就是指自动将软件部署到测试、系统测试、试运行和生产环境的流程。虽然有些企业未能进入生产环节，但采用 DevOps 的企业通常在所有环境中都使用相同的自动化流程，以提高效率并降低由流程不一致所带来的风险。

在测试环境中，首先是自动执行配置、刷新测试数据，然后将软件部署到测试环境，最后执行自动化测试，从而缩短将测试结果返回到开发环节的反馈周期。

采用持续交付实践通常是 DevOps 采用过程中的最关键部分。对于许多 DevOps 从业人员来说，DevOps 的作用范围仅限于持续交付，因此大多数所谓的 DevOps 工具只适用于这一流程。但是，正如您在本书中所看到的，DevOps 的范畴更为广泛。持续交付是 DevOps 的必要组成部分但并不是唯一的组成部分。

根据贵公司的业务需求和急需解决的问题，您可以选择利用第 2 章中介绍的其他流程或采用方法来启动 DevOps 采用工作。

持续测试

我们已经在第 2 章中介绍了持续测试。从流程的角度来看，您需要在三个领域中采用各种流程来实现持续测试：

- 测试环境资源设置和配置
- 测试数据管理
- 测试集成、功能、性能和安全

在企业中，质量保证团队需要确定为每个领域采用哪些流程。他们采用的流程可能因项目而不同，具体取决于每个测试需求和服务协议要求。例如，相比内部应用程序，面向客户的应用程序可能需要更多的安全性测试。相比采用瀑布方法、数月仅需测试一次的项目，测试环境配置和测试数据管理对于使用敏捷方法和持续集成实践的项目来说更为重要。同样，相比简单的单体 Web 应用程序，所含组件拥有不同交付周期的复杂应用程序的功能和性能测试要求也有不同。

持续监控和反馈

客户反馈的形式多种多样，例如客户提交的凭单、正式变更请求、非正式抱怨以及应用商店中的评分等。特别需要注意的是，由于社交媒体和应用商店的流行（参见第 5 章），企业需要定义明确的流程，用于从一系列来源收集反馈，并将其纳入软件交付计划。这些流程也需要足够敏捷以适应市场和法规变化。

监控数据也会带来反馈。这些数据可能来自运行应用程序的服务器，来自开发、质量保证和生产环节，或者来自应用程序中用于捕获用户操作的嵌入式度量工具。

警告

可能会出现数据过载的情况，因此企业需要数据收集和数据使用流程，用于改进他们的应用程序及其运行环境。

衡量流程采用情况

通过观察一系列效率和质量指标是否随时间的推移而不断改进，可以衡量流程采用成功与否。这种类型的测量需要两个先决条件：

- 您必须确定一套合适的效率指标和质量指标。这些指标应对业务发展具有重要意义。
- 您需要建立一个衡量改进程度的基准。

您可以使用多种定义明确的框架来衡量流程的成熟度。对于特定于 DevOps 的流程，全新的 IBM DevOps 成熟度模型等可以评估这些流程的成熟度。有关 IBM 成熟度模型的更多信息，请访问 ibm.biz/adoptingdevops。

DevOps 中的技术

技术可让人们专注于高价值的创造性工作，而将日常工作交给自动化工具。技术还能让从业人员团队利用和延展他们的时间和能力。

如果企业正在构建或维护多个应用程序，那么所有工作都必须具备可重复性和可靠性，以确保所有应用程序的质量。企业不能每次发布应用程序的新发行版或修复程序时都从头开始，而是应该重复利用资产、代码和实践，进而实现成本效益和高效。

同时，实现自动化工具的标准化也可以提高人员的效率（参见本章前面的“DevOps 中的人员”）。企业可能会遭遇员工、承包商或资源供应商流失问题，也可能需要将人员从一个项目转移到另一个项目。但是，利用一组通用的工具可以让从业人员在任何地方工作，而新团队成员只需要掌握一组工具就可以，也就是掌握一个高效率、性价比高、可重复、可扩展的流程。

基础架构即代码

*基础架构即代码*是 DevOps 的一项核心功能，可让企业管理开发环境设置和配置的规模和速度，从而支持持续交付。

围绕基础架构即代码概念演变而来的一个概念是*软件定义环境*。基础架构即代码涉及将节点定义和配置作为代码进行处理，而软件定义环境使用各种技术来定义由多个节点组成的整个系统，其中不仅包括节点配置，还包括节点定义、拓扑、角色、关系、工作负载和工作负载策略及行为。

目前市场上有三种自动化工具可用于管理基础架构即代码：

- **以应用程序或中间件为中心的工具：**这些工具通常能够将应用程序服务器及其运行的应用程序作为代码进行管理。这些是专用工具，捆绑有支持技术的典型自动化任务库。这些工具不可用于执行低级任务，如配置操作系统 (OS) 设置，但可以完全自动执行服务器和应用程序级别的任务。
- **环境与部署工具**这些都是新型工具，能够部署基础架构配置和应用程序代码。
- **通用工具：**这些工具并不专门用于任何技术，可以通过编写脚本来执行多种类型的任务，从配置虚拟或物理节点上的操作系统到配置防火墙端口等无一例外。虽然与以应用程序或中间件为中心的工具相比，这些工具需要做更多的前期工作，但它们可以处理更多的任务。

通过使用像 IBM UrbanCode Deploy with Patterns 这样的环境管理和部署工具，企业可以快速设计、部署和复用环境，加速执行交付管道。

交付管道

*交付管道*包含从应用程序开发到生产的各个阶段。图 3-1 显示了一系列典型的阶段。然而，这些阶段可能因企业而异，还可能因应用程序而异，具体取决于企业的需求、软件交付流程与成熟度。自动化水平也可能有所不同。一些企业实现了交付管道的完全自动化；其他企业由于法规或公司的要求，需要对软件实施手动检查。您无需同时处理所有阶段，可以先专注于企业的关键部分，然后逐渐扩大关注范围，最后涵盖所有阶段。

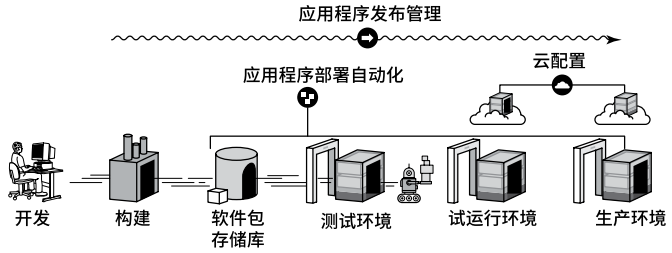


图 3-1： 典型 DevOps 交付管道的各个阶段。

典型的交付管道所包含的各个阶段将在下边的章节中详述。

开发环境

应用程序的开发工作在*开发环境*中进行，该环境为开发人员提供了多种用于编写和测试代码的工具。除了开发人员用于编写代码的集成开发环境 (IDE) 工具之外，此阶段还包括支持协作开发的工具，例如用于源代码控制管理、工作项管理、协作、单元测试和项目计划的工具。此阶段的工具通常是跨平台、跨技术的，具体取决于正在开展的开发工作类型。

构建阶段

*构建阶段*是编译代码的阶段，以创建要部署的二进制文件并对其进行单元测试。此阶段可能会使用多种构建工具，具体取决于跨平台和跨技

术需求。开发组织通常使用构建服务器来不断地构建所需的大量版本，从而支持持续集成。

软件包存储库

软件包存储库（又称为*资源存储库*或*工件存储库*）是一种通用存储机制，可存储构建阶段所创建的二进制文件。这些存储库还需要存储与二进制文件相关的资产，例如配置文件、基础架构即代码文件和部署脚本，以便促进二进制文件的部署。

测试环境

*测试环境*是质量保证、用户验收和开发/测试团队进行实际测试工作的环节。这个阶段会使用众多不同的工具，具体取决于质量保证要求。以下是一些工具示例：

- **测试环境管理工具：**这些工具可帮助设置和配置测试环境，其中包括基础架构即代码技术、以及（如果环境位于云中）云配置和管理工具。
- **测试数据管理工具：**对于任何希望实现持续测试的企业而言，管理测试数据都是一项重要功能。可执行的测试数量和频率取决于可用于测试的数据数量以及数据刷新速度。
- **测试集成、功能、性能和安全：**自动化工具可用于这些类型的测试。这些工具应该与通用的测试资源管理工具或存储库相集成，上述存储库可以存储所有测试场景、测试脚本和相关结果，并建立可追溯到代码、需求和缺陷的可跟踪性。
- **服务虚拟化工具：**现代应用程序并非简单的单体应用程序，而是复杂的系统，依赖于其他应用程序、应用程序服务器、数据库，甚至第三方应用程序和数据源。遗憾的是，在测试阶段，这些组件可能不可用或非常昂贵。而服务虚拟化解决方案可模拟应用程序中选定组件的行为（功能和性能），从而支持对整

个应用程序实施端到端测试。这些工具创建运行测试所需的应用程序和服务的存根（虚拟组件）。当应用程序与这些存根进行交互时，便可以测试应用程序的行为和性能。IBM Rational Test Virtualization Server 提供了这种测试虚拟化功能。

试运行和生产环境

应用程序可以部署到试运行和生产环境中。这些阶段使用的工具包括环境管理和设置工具。基础架构即代码工具在这些阶段也发挥着关键作用，因为这些阶段存在于大规模的环境当中。随着虚拟化和云技术的出现，试运行和生产环境可能由数百甚至数千台服务器组成。监控工具可支持组织监控生产环境中已经部署的应用程序。

部署自动化和发布管理

要对应用程序从一个阶段自动部署到下一个阶段的过程进行管理，就需要专门的工具。我们将在后面的章节中介绍其中一些工具。

部署自动化

部署自动化工具是 DevOps 领域的核心工具。这些工具可执行编排齐整的部署任务，并在构建和交付管道的所有阶段跟踪哪个版本部署到了哪个节点上。这些工具还可以管理所有阶段的环境配置，而应用程序组件必须部署到这些环境中。

部署自动化工具可管理已经部署的软件组件、需要更新的中间件组件和中间件配置、需要变更的数据库组件，以及对这些组件部署环境的配置变更。这些工具还能捕捉并自动执行实施这些部署和配置变更的流程。IBM UrbanCode Deploy 就是这样一款部署自动化工具。

衡量技术采用情况

衡量工具和技术的投资回报是一件相当简单的事情。通常，您可以衡量自动化工具所实现的效率。此外，自动化工具可以让您提高任务的可扩展性和可靠性，而手动工具在这方面并非总是有效。最后，使用一套集成式自动化工具可以提高协作水平、可追溯性、可用性。

发布管理

要协调与每次发布活动相关的发布计划和部署工作，需要业务、开发、质量保证和运维团队进行密切协作。发布管理工具可让企业规划和执行发布工作，为发布活动涉及的所有利益相关方提供单一协作门户，并在构建和交付管道的所有阶段为发行版及其组件提供可跟踪性。IBM UrbanCode Release 就能提供这种发布管理功能。

本章概要

- 使用云技术作为 DevOps 的推动者
- 了解完整组合部署
- 了解不同的云服务模型
- 揭示混合云的概念

第 4 章
了解云技术如何加速 DevOps

DevOps 和云技术互为催化剂和推动者。随着许多组织开始采用云技术，利用云平台来运行 DevOps 工作负载的价值主张就不证自明了。云平台的灵活性、灾备能力、敏捷性以及种类丰富的服务有助于精简在云端运行的应用程序交付管道。企业可以根据需要，随时配置包括开发、测试、生产在内的所有环境。该流程能够最大程度减少交付过程中与环境相关的瓶颈。许多组织还希望利用云平台降低开发和测试环境的成本，或者为他们的从业人员提供精简的现代化开发人员体验。这些都构成了极具吸引力的云技术和 DevOps 业务用例。

本章主要研究针对 DevOps 的各种云模型，讨论将 DevOps 作为云平台上工作负载的价值主张。

使用云技术作为 DevOps 的推动者

DevOps 的主要目标是最大程度减少交付管道中的瓶颈，使其更加高效、更为精益。组织面临的最大瓶颈之一是环境可用性和配置问题。对于从业人员，尤其是开发人员和测试人员来说，通过正规的凭单流程请求创建环境的情况十分常见，但这种流程请求即使不需要几周也需要几天才能实现。

DevOps 的一个原则就是在类生产环境中进行开发和测试。加剧环境可用性瓶颈的主要原因是存在可用环境与生产环境不匹配的难题。这种不匹配小则可能只是环境在操作系统 (OS) 或中间件级别的配置不同，大则可能是开发环境与生产环境的操作系统或中间件类型完全不同。


警告

环境可用性的缺乏导致从业人员可能面临等待漫长的时间。开发环境与生产环境的不匹配可能引发重大质量问题，因为开发人员无法验证开发的应用程序在生产环境中将如何表现，也不确定是否可以通过部署到测试环境时所用的流程将应用程序部署到生产环境。

云技术可通过下述方法解决这些问题：

- 提高云平台上环境的配置速度，为从业人员提供自助服务，使他们能够根据需要使用和访问环境。
- 能够按需动态配置和取消配置这些环境，通过减少对静态持久测试环境的需求，实现更出色的环境管理和成本节约。
- 能够利用“模式”技术，帮助组织将环境定义为软件，并实施版本控制，以便根据从业人员的需求提供和配置环境（特别是类生产环境）。
- 从自动化角度来看，如果能够提供应用程序部署自动化技术（比如 IBM UrbanCode Deploy），就可以利用单一工具配置

云环境，并在需要时将正确版本的应用程序部署到这些环境。它们还可以快速配置环境和应用程序，以便满足从业人员的需求。

- 提供与云环境结合使用的服务虚拟化技术（比如，IBM Rational Test Virtualization Server），实现测试所需的服务模拟，而无需配置这些服务的真正实例。

图 4-1 显示云环境如何与部署自动化和服务虚拟化技术协同工作，提供端到端的开发/测试环境。

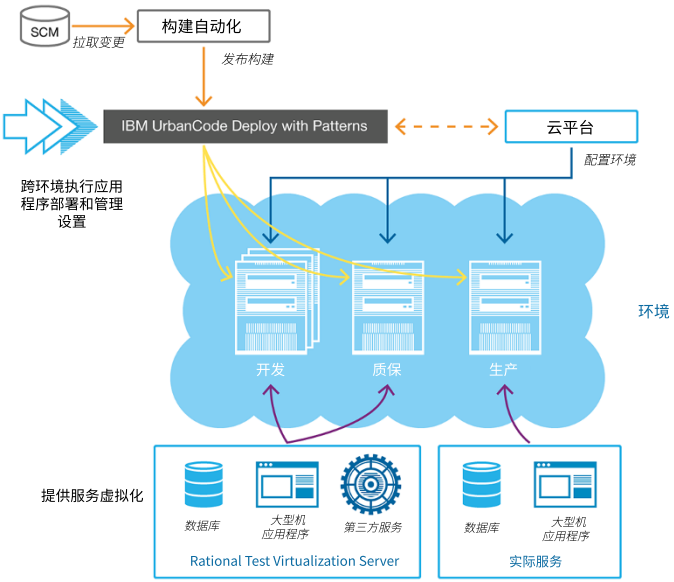


图 4-1：云中的端到端开发/测试。

没有 DevOps 的云环境无法充分利用云技术的全部优势。采用 DevOps 和在云端托管的环境能够为组织提供云技术的全部优势，从而更有效地交付软件应用程序。

完整组合部署

部署云应用程序的任务包括部署应用程序和配置用于运行应用程序的云环境。这两项任务可以单独执行，但如果结合执行，就是所谓的*完整组合部署*。我们将在这个部分较为详细地讨论这两种方法。

第一种方法是将云环境配置与应用程序部署分隔开。在这种情况中，不会为云环境和在其中部署的应用程序建立单一统筹点。应用程序部署自动化工具仅将云环境视为静态环境。这种情况无法最大程度发挥部署到云环境的优势。

第二种方法是利用部署自动化工具作为单一统筹工具，进行云境配置以及向配置的环境部署应用程序。为此，可创建用于捕获云环境定义和拓扑结构的“蓝图”，然后将应用程序组件和配置映射到云环境中定义的节点。

可使用多种模式技术（比如 IBM Virtual System Patterns 和 OpenStack HOT 模板），将云环境定义为模板。可使用部署自动化工具（比如 IBM UrbanCode Deploy with Patterns），根据这些蓝图交付完整组合配置。这包括配置蓝图中定义的云环境，并将应用程序部署到配置的环境中。配置环境后，可将进一步的应用程序、配置和内容变更作为更新持续部署到云环境中。

或者，组织可以选择始终进行完整组合部署，这样环境和相关应用程序始终作为单一可部署的资产进行配置。在这种情况下，不会对现有环境进行任何更新。

为 DevOps 选择云服务模型

采用云平台时，首先需要决定的是计划迁移到云平台的工作负载责任范围以及希望自己企业内部承担的责任。主要有两种云服务模型：基础架构即服务 (IaaS) 以及平台即服务 (PaaS)。

IaaS

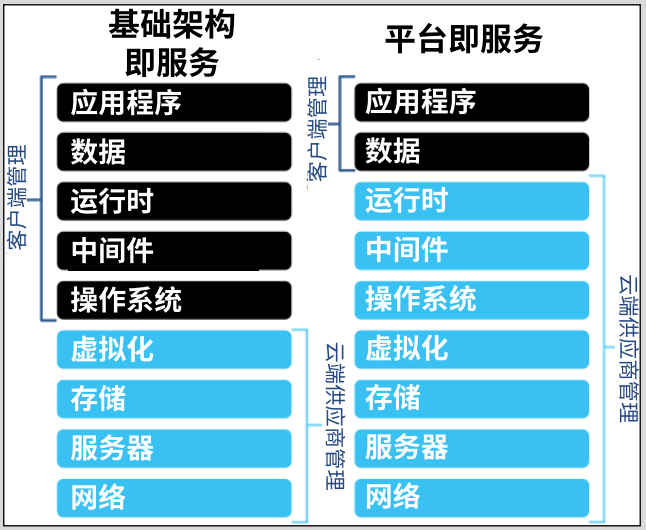
采用 IaaS 云模型时，云平台管理底层基础架构，提供管理所有虚拟化基础架构所需的功能和服务。操作系统、中间件、数据以及应用程序的安装、打补丁和管理仍由用户负责。

在采用 DevOps 作为云平台上工作负载的情况下，决定使用哪种云服务模型就决定了 DevOps 的采用方式。在 IaaS 模型中，用户组织负责管理整个交付管道。所有的工具和交付管道的集成均由用户组织负责，包括获取适当的工具集以及确保集成这些工具集以形成交付管道。此外，他们还需要确保开发团队和运维团队之间的合作遵循 DevOps 文化。使用云平台并不意味着不需要消除交付代码的开发人员与交付基础架构（现在是基于云的服务）的运维团队之间的条块分割状况。

虽然通过向应用程序交付团队提供 IaaS，云平台带来了巨大的价值，但仍需要部署所有适当的 DevOps 功能，以便能够实现期望的 DevOps 价值。

职责分离

在 IaaS 云上利用 DevOps 时，要明确一个重要问题，也就是如何界定云平台与应用程序部署工具之间的职责分离。每个工具具体担负什么职责？回答这个问题的简单方法就是了解云组合中缓慢变化和快速变化的资产。下图显示应用程序组合的不同层次，从操作系统层、存储层、网络层一直到应用程序层。



应用程序、数据和中间件配置层从本质上而言都在快速变化。这些变化往往是因为应用程序、应用程序数据和应用程序的使用发生迭代。这种变化的速度对于仍在开发中的应用程序来说非常之快。再往下的层次包括中间件（应用程序服务器、数据库等等）、操作系统和存储层，它们不会经常变化。由于为了执行只影响应用程序及其内容或配置的简单变更就去更新和重新配置所有层次并不划算，效率也不高，因此有必要对应用程序部署和云管理工具中快速变化的层次和缓慢变化的层次进行职责分离。快速变化的层次由应用程序部署工具管理并自动执行，缓慢变化的层次则由云平台提供的云管理软件负责。

PaaS

采用 PaaS 云模型时，您作为用户，仅负责应用程序和数据。所有其他功能均由云平台以服务方式提供。这样能够显著改善应用程序交付团队从业人员的体验。应用程序开发和测试工具现在都作为服务在从业人员可以访问的平台上提供。应用程序交付组织不再负责管理交付管道。交付管道嵌入到 PaaS 中，这样从业人员就能够将精力集中于快速交付应用程序。开发和测试工具以及基础架构配置都作为服务从从业人员一端抽象出来，以便支持他们专注于交付应用程序的核心责任。

IBM Bluemix 是一种 PaaS。IBM 及其合作伙伴管理该平台以及在该平台上提供的服务。该平台嵌入了 IBM DevOps Services — 这是一系列服务，提供团队采用 DevOps 所需的全部功能，更准确地说，这是作为一系列服务提供的应用程序交付管道。应用程序交付团队可以使用这些服务，而无需关心这些服务的托管和交付方式。DevOps 服务包含以下内容：

- 基于 Web 的集成开发环境 (IDE) 即服务
- 构建即服务
- 规划和任务管理即服务
- 安全扫描即服务
- 部署即服务
- 监控和分析即服务

该平台还针对在整个交付生命周期的不同环境（包括开发、测试、试运行和生产环境）中运行的应用，提供可扩展的运行时环境。

了解何为混合云

混合云已成为云技术领域一个极其常见的词汇。这个词可能被过度用于描述多云场景，要么是多种云技术共存，要么是云基础架构和实体基础架构共存。要轻松定义混合云，首先需要了解以下多云场景：

- **云基础架构和实体基础架构：**这是极其常见的混合云场景。除非组织“诞生在云端”，否则这实际就是默认场景。所有特定组织目前都有在现有实体基础架构上运行的工作负载和应用程序。在许多情况下，其中一些应用程序将继续在实体基础架构上运行。典型示例包括大型机应用程序和数据密集型记录系统应用程序，因为技术或成本方面的限制，很难将它们迁移至云端。即使组织将所有工作负载都迁移至云端，迁移过程也不可能一蹴而就，可能会出现实体基础架构和云基础架构在很长一段时间并存的局面。
- **内部云和外部云：**在这种场景中，组织可能为一些应用程序和工作负载采用外部云（公共云或虚拟专业云），为另一些采用内部云（私有云）。例如，组织利用低成本的外部云作为开发环境，而在组织数据中心实施自己管理的内部云来运行所有生产工作负载。
- **IaaS 和 PaaS：**在这个场景中，客户采用 PaaS 云模型处理一些工作负载（例如，创新的新兴互动系统应用程序），此外采用 IaaS 处理一些较为传统的记录系统工作负载。

对于采用 DevOps 而言，混合云的存在带来了一些新挑战，因为它会导致应用程序交付管道分布在复杂的混合云和实体环境中。这些混合云环境的示例还有：

- 组织选择使用公共云用于开发、测试以及其他非生产环境，使用内部云甚至实体基础架构用于生产环境。
- 组织将一些互动系统应用程序部署到云环境，同时将为核心业务应用程序提供后端服务的记录系统应用程序仍保留在实体基础架构，比如大型机上。
- 组织利用公共 PaaS 进行创新型应用程序试验，一旦试验成功，就将它们迁至私有云。
- 组织希望在多个云平台之间实现应用程序工作负载的可移植性，以确保不会出现被特定供应商“套牢”的状况，或者实现在多个云供应商的平台上部署关键工作负载的能力。

要采用 DevOps 与混合云结合的开发方法，核心要求是跨多种云环境和物理环境部署应用程序。像 IBM UrbanCode Deploy with Patterns 这类的应用程序可以利用应用程序蓝图将应用程序和配置映射到多种环境中，包括实体环境和云环境，从而能够跨复杂的混合云环境实现自动化应用程序部署。

本章概要

- 支持移动应用程序
- 处理 ALM 流程
- 扩展敏捷实践
- 管理多层应用程序
- 了解企业中的 DevOps
- 配合供应链工作
- 了解物联网

第 5 章
利用 DevOps 应对新挑战

DevOps 起源于所谓的网上诞生的公司，也就是在互联网上诞生的公司，例如 Etsy、Flickr 和 Netflix。这类公司使用相当简单的架构便能应对大规模的复杂技术难题，不同于围绕遗留系统和/或通过收购和兼并成长起来的大型企业，后者所拥有复杂的多技术系统，这类系统必须协同工作才能支持企业发展。同时，随着移动等新技术和软件供应链等应用交付模式不断对现代企业提出新要求，进一步加剧了这些挑战。

本章将探讨 DevOps 可以帮助企业解决的一些难题。

移动应用程序

在企业中，移动应用通常不是独立的。移动设备自身很少包含业务逻辑，更多的是作为企业正在使用的多种企业级应用程序的前端。这些后端企业应用程序可能包括事务处理系统、员工门户及客户获取系统。移动开发和交付非常复杂，需要使用一系列通过协同、可靠和高效的方式交付的依赖性服务。

对于企业移动应用程序而言，发布周期和新功能发布需要与同移动应用程序交互的企业应用程序和服务实现协调一致。因此，DevOps 的采用需要移动应用程序团队以及其他企业软件开发团队的共同参与，并且前者的参与更为重要。

DEVOPS 和应用商店

移动应用的特色之一是需要部署到应用商店。大多数移动应用都无法直接部署到移动设备中，必须通过供应商管理的应用商店进行部署。苹果公司在其应用商店中率先推出了这种分发形式（并锁定其设备，防止直接安装应用开发商或供应商的应用）。Research In Motion、谷歌和微软等设备制造商曾一度允许直接安装应用，但现在也都采用了苹果公司的模式。

这种情况为部署流程添加了一个异步步骤。开发人员不再能够根据需要向应用部署更新。即便是为了修补严重漏洞，新应用版本也必须通过应用商店的提交和审核流程。由于新增提交和等待审批环节，持续交付的战线被拉长。但是，仍然可以将应用持续部署到开发和测试环境，其中测试环境是部署应用程序的设备的模拟器或一系列物理设备。

全球 80% 的企业数据产生自大型机，70% 的事务与大型机有关。如果将这些大型机功能迁移到移动平台，就能转变您开展业务和客户互动的方式，但这说起来容易做起来难。您可能面临技能缺口、孤岛式组织架构和众多平台，导致发布周期长、经常发生不必要的延迟以及浪费宝贵资源。为提供企业应用程序的移动访问途径，企业纷纷采用 DevOps 这种软件交付方法，在强调速度和效率的同时不牺牲稳定性和质量。

没有仅适用于移动应用程序的具体 DevOps 概念或原则。不过，移动应用程序的开发周期较短，且需要快速变更，因此增加了对 DevOps 的需求。

ALM 流程

应用程序生命周期管理（ALM）是一套用于管理应用程序生命周期的流程，覆盖从创意（业务需求）到应用程序部署和维护的所有环节。因此，若将 DevOps 视为端到端业务功能，应用程序生命周期管理将成为 DevOps 流程的基础概念。DevOps 拓宽了 ALM 的范围，使企业所有者、客户以及运维组织都包含在该流程中。

DevOps 开发/测试采用途径（请参阅第 2 章以了解更多信息）最贴合 ALM 传统的需求管理、变更管理、版本控制、可追溯性以及测试管理等功能。但是，跟踪和规划等其他 ALM 功能包含在“确定目标”采用途径中，而仪表板和报告功能则包含在“运营”采用途径中。

扩展敏捷实践

精益和敏捷开发是 DevOps 开发的两大支柱，其成效之一是减少浪费，提高团队效率。提高效率和重复最佳实践可以缩短开发周期，使团队提高创新能力和响应能力，从而提高客户价值。DevOps 方法的核心在于，

将精益和敏捷原则扩展到开发团队之外的多个团队中，扩展到整个产品和软件交付周期中。

许多团队已经采用了敏捷开发方法，并想要扩展现有流程，使其融入到 DevOps 采纳过程中。有许多常用框架可以帮助扩展敏捷开发，包括“扩展敏捷框架”（SAFe）和“规范敏捷交付”（DAD）等。一些组织还将 Scrum 流程有效地扩展到超大规模团队中。这些框架旨在提供一种方法，以便在企业级别采用敏捷开发。这意味着不仅要考虑代码开发，还要包括架构、项目融资和流程治理以及管理层需要的角色等，并应用在团队级别十分有效的精益和敏捷原则。不论使用什么框架来扩展敏捷开发，都需要采用那些基本的敏捷原则并运用最佳实践来利用这些原则，以提高整个企业的效率和有效性。

多层应用程序

在典型的大型 IT 企业中，跨越多个平台的多层应用程序并不罕见，每个应用程序都有其独特的开发流程、工具和技能要求。这些多层系统通常集结了前端和后端系统上 Web 应用程序、桌面应用程序以及移动应用程序，例如打包应用程序、数据仓库系统、大型机上运行的应用程序和中端系统。多层系统的组件可能位于不同的平台上，因此，这些组件的发布管理和协调工作堪称一项巨大的挑战，即便最训练有素的 IT 企业也会束手无策。

一个明智的做法就是，在所有开发阶段中采用一致的自动化构建、配置和部署流程。这种方法可以确保根据需要构建组件，避免无用功。这种方法还可以确保随着变更的引入以及项目在测试、质量保证和生产周期中的推进，应用程序始终保持完整。IBM UrbanCode Deploy 拥有一个应用程序模型，可帮助自动执行多层应用程序的复杂部署任务。

如今，多平台、多品牌世界带来的一个现实问题是，企业需要根据平台为不同的团队提供和维护不同的工具。而这正是 IBM Jazz 等开放平台的优势所在，这类平台可以集成不同工具，从而提供统一的解决方案。一致的部署实践可以确保团队使用可靠且可重复的跨平台部署流程，进而实现真正的业务价值。

企业中的 DevOps

IT 交付软件的速度对于当今企业至关重要。这些企业通常运行部署在大型机和中端系统上的记录系统应用程序（内部开发的应用程序或打包应用程序）。他们面临诸多挑战：

- 监管障碍
- 流程复杂性
- 技能缺口
- 孤岛式组织架构
- 导致发布周期长、不必要的延迟以及资源浪费的平台和工具

企业级的 DevOps 支持规划、开发、测试和运维方面的利益相关方在自己组织内持续交付软件。当今企业需要部署真正的跨平台应用程序，涵盖从移动平台到大型机的所有平台。DevOps 方法使用精益原则，创建高效的交付管道，支持开发、测试和交付应用程序，同时帮助提高质量、加快速度和降低开发成本。

鉴于当今企业真正具备了多平台性质，以及必须创建、集成、部署和运营移动、云、分布式和大型机应用程序，因此 DevOps 的效率、精简和协作功能将成为关键的差异化竞争优势。

供应链

随着企业越来越多地借助服务外包和战略合作伙伴来获取技能和能力，软件*供应链*逐渐成为常态。供应链是一个由产品或服务从供应商转移到客户的过程中所涉及的组织、人员、技术、活动、信息和资源构成的系统。供应链中的各个供应商可能属于企业内部或外部。

在采用供应链这种软件交付模式的企业中，采用 DevOps 可能是一项挑战，因为供应商之间的关系更多地是通过合同和服务级别协议进行管理，而不是通过协作和沟通进行管理。但是，此类企业仍然可以采用 DevOps。核心项目团队仍然掌握规划和衡量职能，其他职能则由其他供应商共享。在交付管道中，不同的供应商可能处于不同的阶段。因此使用通用工具集和通用资源库是非常重要的。例如，工作项目管理工具可以提供详细报告，内容涉及所有供应商正在处理的所有工作项，以及不同供应商之间工作项所有权的转移情况。使用通用资产存储库可提供一种支持资产在整个通道中传递的机制，从而实现持续交付。

物联网

DevOps 的下一个重大发展动向是进入系统或嵌入式设备领域，这个领域通常称为*持续工程*。在互联网发展早期，网络上共享的大部分数据都是人为生成的。如今，无数接入互联网的设备（如传感器和执行器）产生的数据远远超过人所生成的数据。这种由互联网上的互联设备所构成的网络通常被称为*物联网*。

在这个领域，由于硬件和其运行的嵌入式软件之间存在相互依赖性，DevOps 可能更加不可或缺。在持续工程中体现 DevOps 原则，可确保交付给设备的嵌入式软件是符合正规工程规范的高质量软件。

在持续工程下，“运维”环节将由负责为设备设计和构建定制硬件的硬件或系统工程师取代。尽管硬件和软件开发需要遵循不同的交付周期，但开发和测试团队与系统工程师之间的协作对于确保软硬件的协调开发和交付至关重要。开发和测试团队对持续交付和持续测试的需求保持不变。模拟器用于在开发过程中测试软件和硬件。

反模式

在现实世界中，采用 DevOps 原则时总会遇到限制条件。其中一些限制是企业所处的行业和环境带来的，例如监管合规、复杂的硬件系统或不成熟的软件交付能力等。在这种情况下，采用 DevOps 需要考虑*反模式*（无效或反生产模式），企业可能由于自身的业务需求无法接受反模式。

瀑布式 Scrum

全球性研究和咨询公司 Forrester (`www.forrester.com`) 率先提出了 *Water-SCRUM-fall*（瀑布式 Scrum）这一术语，用来描述敏捷软件开发方法的当前采用状态。从 DevOps 交付来看，虽然开发团队可能采用了敏捷实践，但他们周围的团队可能仍然使用不支持持续交付的手动瀑布式流程。在一些企业中，这种局面是由企业文化造成的。采用 DevOps 的企业必须在更广泛的 DevOps 实践中嵌入手动流程。

NoOps

在 NoOps 企业中，运维部门不再是独立部门，其职责并入开发部门。互联网电视提供商 Netflix 就是这种方法的倡导者。NoOps 可能只适用于部分企业，但仍需耐心等待一段时间才能看清这种组织模式是否具有更广泛的实际吸引力。

本章概要

- 了解面向高管的最佳实践
- 组建团队
- 确定 DevOps 目标
- 记录 DevOps 转型
- 借鉴 DevOps 成果经验

第 6 章
让 DevOps 发挥作用：IBM 历程

DevOps 在 IBM 整个公司范围全面推广，并且不断与时俱进。IBM 软件部 (SWG) Rational 产品部门率先使用 DevOps 方法取得了成功，为全公司的采用奠定了基础，现在，DevOps 已广泛应用于 Watson 部门、Tivoli 部门、全球企业咨询服务部以及其他部门。本章介绍 IBM 软件部的 IBM Rational Collaborative Lifecycle Management 产品团队采用 DevOps 功能的成功案例。

这项软件交付工作的独到之处在于，它采用*开放*方法完成 — 软件交付团队在 jazz.net 上交付所有开发工件和日常工作，包括所有详细的工作项。这个网站对公众开放，任何注册用户都可以查看计划中的工作、进行中的工作以及所有已完成的软件产品开发工作的历史记录。

了解高管的作用

文化是组织的暗藏脉络。它以管理层和其他员工的价值观和行为作为基础，不断发展完善。许多时候，如果没有重大的变革发生，人们也许不会真正了解企业的文化。总有些怀疑论者持观望态度，不确定这是企业文化，还是昙花一现的现象。领导者终将出现。必须制定方法，用于了解这些动态和负责企业文化的管理人员，以便解决阻碍企业文化发展的真正因素，这至关重要。


牢记

为及时掌控文化动态，IBM 软件部高管运用了一系列方法：

- **选择合适的领导者。**领导者的作用是综合不同的观点，帮助团队确定一系列共同的目标、障碍、流程变化以及决策的起点。
- **让利益相关方参与。**领导层、管理层以及不同开发领域的个人贡献者都必须对这些变革提供支持。这必须包括利益相关方、架构师、开发人员、测试人员以及运维部门，以及这些领域负责支持变革的领导。
- **衡量改进和成果。**必须建立一系列关键指标，综合所需的效率目标和业务成果目标。这些目标和衡量指标必须高标准严要求，既要鼓励人们承于担责，但也不要让人望而生畏，打退堂鼓。
- **通过早期成功创造前进动力。**了解低效环节，衡量每个领域的改进情况，从而形成变革动力。
- **沟通与倾听。**作为领导者，必须了解变革如何在团队中扎根的真实动态。投入时间，与技术团队、管理层以及业务领导开展一对一对话和定期的面对面沟通，这有助于评估团队对于变革的接受程度、他们对障碍的看法，同样重要的是，让管理层有机会分享他们对于优先任务和进展情况的看法。

如果您是高管，应当对团队给予支持，并亲身参与以了解并消除障碍。怀着明确的业务目标，以整个团队形式开展运营，这是让所有人都朝同一个方向进发的必然选择。

整合团队

IBM 软件部的 Rational Collaborative Lifecycle Management 产品团队是较大部门中的一个团队，他们开发了超过 80 种软件开发工具，涵盖软件交付规划、软件开发、应用程序部署、软件质量管理以及应用程序监控和分析等领域。

这个 IBM 软件部产品团队是一个大型全球性组织，拥有四个核心产品团队，分布于 10 个国家或地区的超过 25 个地点。在采用 DevOps 方法之前，该团队总是制定年度发布计划，需要提前三到六个月决定年度发布计划中的内容。

树立 DevOps 目标

该 IBM 软件部团队感觉自己响应市场变化和客户需求变化所用的时间太长。于是他们决定缩短交付周期，不仅仅包括开发和测试阶段，还涉及与业务利益相关方和客户的协作与互动。设定的目标从年度发布计划转变为季度发布计划。

除了需要加快开发，更频繁地交付新功能外，该团队还必须更迅速地采用云交付模式、移动开发、移动测试以及其他能力，以便能够从容应对技术变化。该团队选择采用 DevOps 原则和实践，转变团队开发软件的方式，从而更早更频繁地为客户实现价值。

进行这种规模的转变需要改变组织文化，所以他们成立了四个工作组，由管理团队的成员和技术领导组成。这些工作组从头到尾检查软件交付流程，承担改变工作方式的责任。他们制定了一组特定的衡量指标和

行动计划，以解决开发流程中的重点问题。他们成立了持续交付支持团队，包含在整个组织内为团队提供培训和分享最佳实践的宣传推广专员。

该 IBM 软件部团队通过确定以下目标，开启 DevOps 采用之旅：

- 精简流程，引进新方法。
- 利用工具实现一致性、可追溯性和指标，确保能够将实践扩展到其他团队。
- 培养持续改进的文化。

从 DevOps 转型中学习

本部分介绍该 IBM 软件部团队用于促进 DevOps 转型的步骤。

扩展敏捷实践

将现有敏捷实践扩展到开发和测试领域之外，将客户、业务利益相关方以及运维团队包含在内，从而打破孤岛式组织架构，改善业务成果。这种更广泛的敏捷模型使各团队能够通力合作，开发出风格一致、质量卓越的软件，通过将流程中的每个步骤都进行整合，为企业实现更高价值。

采用“一个团队”方法，结合产品管理、设计和开发职能。开发团队既包含传统的开发经理和团队负责人角色，还引入了运维管理和架构师角色，用于支持端到端生命周期战略。

配备专门资源，在整个组织中为团队提供有关敏捷和持续交付方面的指导。关注能力而不是产品组件，这有助于打破传统的孤岛式组织边界，为每次冲刺开发和自动化项目做好准备。这些功能团队还会获得专门开发经理的大力支持。开发组织的所有层级都会定期召开 Scrum 会议，找到并解决障碍，跟踪关键指标，研读实时仪表板数据，沟通关键信息。

为了让开发优先任务能够始终跟上市场变化的步伐，该团队成立了战略产品委员会，由产品管理层、开发总监、架构师以及业务负责人构成。他们的责任包括：

- 分配资金，确保项目取得成功
- 推动、协助和支持项目执行
- 制定长期业务愿景和方向
- 针对符合长期愿景的年度发布计划，划分史诗故事和用户案例的优先顺序

充分利用测试自动化

为消除漫长的传统后端测试周期，提高发布质量，该团队采用包含自动化和虚拟化功能的敏捷持续测试方法。他们确定节奏，每四周为一个迭代周期，结束时进行一次演示；每四周为一个里程碑周期，结束时发布一个客户可用的版本。每个里程碑后的回顾以及对技术欠缺情况的了解都有助于消除未来迭代中的浪费。该 IBM 软件部团队的格言是“尽早测试，经常测试”。

该团队采用以下最佳实践，实现测试自动化：

- 自动执行劳动密集型重复测试。
- 在错误频发的领域实现测试自动化。
- 对每个构建版本运行自动化测试；尽早运行，经常运行。
- 创建用于抵制用户界面 (UI) 变更的自动化测试 — 使用框架将 UI 与测试分开。
- 轻松创建、交付和维护自动化测试，建立强有力的功能团队负责制。
- 规划估算自动化开发工作，确保开发人员有充足时间开展工作。

- 建立指标，以便评估自动化测试是否有效（无法衡量就无法改进）。
- 不断重新评估自动化测试是否能够找到错误，如果无法找到，则要重新考虑自动化测试。

为支持测试自动化，该团队部署了 IBM Rational Test Workbench 用于功能和性能测试，实现更频繁的测试和自动部署构建版本。通过使用 IBM UrbanCode Deploy 自动执行构建版本部署，包括自动指定任何必需的应用程序和数据库服务器配置设置，该团队使部署成本降低了 90%。

构建交付管道

该 IBM 软件部团队决定构建交付管道，利用“工具即服务”，使得开发人员仅用大约 60 分钟就能提交代码、进行测试以及将代码部署到生产环境，而以前这需要一到两天时间。这个流程避免了重复工作，最大程度提高了生产力。

该团队在部署过程中，认识到持续交付管道需要采用以下最佳实践：

- 左移测试，尽可能采取自动化测试。
- 始终使用相同的部署机制。
- 争取保持一致的交付准备状态。
- 将基础架构视为代码。

在图 6-1 中，可以看到由该 IBM 软件部团队采用持续交付管道开发的产品和功能。

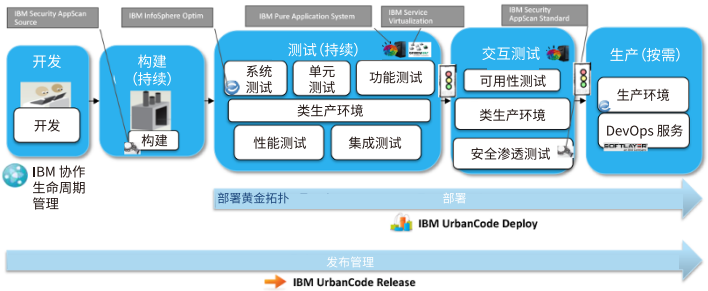


图 6-1：持续交付管道。

在实施持续交付管道的过程中，最为关键的一项最佳实践就是“将基础架构视为代码”。这意味开发人员可以编写脚本作为应用程序代码的一部分，用于配置应用程序所需的基础架构。在过去，这通常由系统管理员或运维人员完成，但现在可由开发人员直接完成，提供所需的控制和效率。Puppet、Chef 以及 IBM UrbanCode Deploy with Patterns 都是新型基础架构自动化工具的示例，可使基础架构即代码变成现实。

该 IBM 软件部团队现在将基础架构视为代码，并遵循以下最佳实践：

- 将模式定义、脚本包以及服务视为代码。
- 对一切进行版本控制。
- 将拓扑结构模式自动部署到云端。
- 管理多个云环境中不同版本的模式。
- 自动执行模式测试。
- 清理目录资源，避免无序扩张。

快速试验

持续交付的概念不仅包含持续集成和持续部署等软件开发活动，还包含更基础的学习活动，这种活动最好通过频繁的试验和评估结果来实现。

在向应用程序添加特性和功能时，您永远无法确知客户是否会获得预期的效益。所以 IBM 必须尽早开展试验，频繁开展试验，征求客户反馈，了解他们真正需要怎样的功能，摒弃那些几乎没有用处甚至是障碍的功能。图 6-2 概括描述了这种战略。

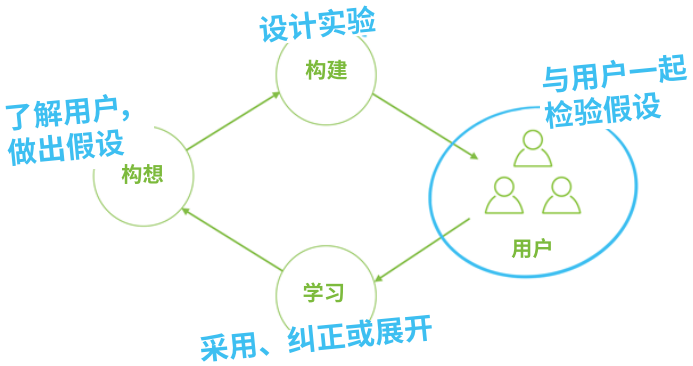


图 6-2：假设驱动型开发一览。

该 IBM 软件部团队从频繁试验中学到了很多，并开发出以下最佳实践：

- 建立指标和成功/失败标准。
- 弄清试验的适用对象 — 针对少量用户的小型测试有助于确定功能的可用性。
- 持续运行多个试验。
- 快速做出基于事实的决策。
- 加速交付就能加快试验速度。
- 建立用于支持系统范围试验的机制（Google Analytics、IBM Digital Analytics 等）。
- 考虑不同的试验模型（经典 A/B 测试、多臂赌博机等）。
- 针对相关项目同时采取两种途径：对云项目进行试验，使用来自试验的数据，不仅推动该项目的方向，还推动相关内部项目的方向。

持续改进

该 IBM 软件部团队希望建立持续改进的组织文化，利用效果与效率衡量指标，确保实现实际改进。该团队管理敏捷项目等持续改进工作。他们通过跟踪成熟度目标、痛点以及相关改进措施，解决各种问题，从而支

持持续改进。他们跟踪持续改进工作（包括其他开发工作），确保投资得到广泛理解。成熟度目标（例如，能力）可能需要一个或多个季度才能完成制定工作并实际采用。重大痛点可能需要数月时间才能缓解或消除。但在任何情况下，具体改进措施的周期都会限制在一个月以内。

该 IBM 软件部团队开展回顾活动，使持续改进形成制度。*回顾*就是定期评审工作进展情况以及改进措施。如果不做回顾，就意味着还没有达到尽善尽美的软件开发水平。在大型团队中，应当建立回顾层次结构。在该 IBM 软件部团队中，每个下级团队都会执行回顾活动，这些回顾活动的结果作为应用程序级别回顾活动的输入信息，而应用程序级别回顾活动的结果又作为更高的解决方案级别回顾活动的输入信息。从回顾活动总结出的措施将记录为痛点，并确定相应的改进措施，用于消除或缓解痛点。

为确保下属团队不断进步，该 IBM 软件部团队制定了业务指标和运维指标，以衡量 DevOps 转型的有效性。业务指标主要衡量以下方面的改进：

- 加快交付速度
- 提高客户满意度
- 减少维护支出，增加创新投资
- 提高客户采用率

随着时间推移，运维指标会影响团队的效率，这些指标主要衡量以下方面：

- 启动新项目所需的时间
- 构建时间
- 迭代测试时间

了解 DevOps 成果

DevOps 方法帮助该 IBM 软件部团队提高了客户满意度和客户采用率，并且实现了两位数的收入增长。开发周期的缩短，激励着 IBM 的交付团队快速交付升级版的内部解决方案和新的云服务，例如 Bluemix、DevOp Services for Bluemix 以及 Collaborative Lifecycle Management as a Managed Service (CLM aaMS)。

图 6-3 是 DevOps 方法在 IBM 取得成功的具体示例，展示了 IBM 软件部 Rational Collaborative Lifecycle Management 产品团队实现的可衡量的成果。

生命周期衡量	2008	2010	2012 – 2014	总体改进
项目启动	30天	10天	2天	**28天**
清理的积压工作	90天	45天	持续	**89天**
总体开发时间	120天	55天	3天	**117天**
复合构建时间	36小时	12小时	5小时	**700%**
BVT 可用性	N/A	18小时	<1小时	**17小时**
迭代测试时间	5天	2天	14小时	**4天**
部署总时间	2天	8小时	4小时->20分钟	**2天**
交付到生产环境的总时间	9天	3天	2天	**7天**
发布时间间隔	12月	12月	3月	**9月**

图 6-3： IBM 软件部团队衡量的改进情况。

本章概要

» 了解 DevOps 适用的领域

» 了解 DevOps 不适用的领域

第 7 章
十大 DevOps 误区

DevOps 运动方兴未艾，广大企业正在如火如荼地推行这种模式。与任何新运动或新趋势一样，出现关于 DevOps 的误区和谬见也是在所难免。其中一些误区可能源自曾经尝试采用 DevOps 但以失败告终的公司或项目。所以，根本不存在放之四海而皆准的万能方案。以下是关于 DevOps 一些常见的误区和真相。

DevOps 仅适用于“网上诞生”的组织

DevOps 起源于所谓的“网上诞生”的公司，也就是在互联网上诞生的公司，例如 Etsy、Netflix 和 Flickr。但是，数十年来，大型企业一直在使用与 DevOps 一致的原则和实践来交付软件。此外，正如本书所述，当前流行的 DevOps 原则已经达到一定的成熟度，可适用于拥有多平台技术和分布式团队的大型企业。

DevOps 需要运维团队学习编码

运维团队一直采用编写好的脚本来管理环境和重复性任务，但随着基础架构即代码的发展，运维团队发现需要通过软件工程实践（例如代码版本控制、检入、检出、分支以及合并），管理大量的此类代码。如今，运维团队可以通过创建用于定义环境的新版本代码，创建新版本的环境。但是，这并不意味着运维团队需要学习如何使用 Java 或 C# 编写代码。大多数基础架构即代码技术使用 Ruby 等语言，这些语言相对容易上手，对于具有脚本编写经验的人员来说尤其如此。

DevOps 只适用于开发和运维

尽管顾名思义，DevOps 是指开发和运维，但 DevOps 却适用于整个团队。DevOps 关乎软件交付过程中所涉及的所有利益相关方，包括业务部门、从业人员、高管人员、合作伙伴、供应商等。

DevOps 不适用于 ITIL 组织

有人担心，DevOps 的持续交付等功能与信息技术基础架构库 (ITIL) 所规定的检查和流程不兼容。ITIL 是一套用于 IT 服务管理的最佳实践。实际上，ITIL 的生命周期模型与 DevOps 是兼容的。ITIL 定义的大部分原则与 DevOps 原则高度一致。但是，ITIL 在一些企业中声誉不佳，这是因为 ITIL 的部署方式主要采用缓慢的瀑布式流程，而这些流程不支持快速变更和改进。在开发和运维两大领域之间实现这些实践的协调统一正是 DevOps 的本质所在。

DevOps 不适用于管制行业

管制行业的首要需求是制约与平衡，同时需要从负责合规性与可审计性的利益相关方获得批准。如果执行得当，采用 DevOps 实际上可以改善合规性。此外，通过实现流程自动化，以及使用具备内置审计跟踪捕获功能的工具，也可以提供帮助。

管制行业中的企业都会设有手动检查点或检查关口，但这些元素与 DevOps 并不冲突。

DevOps 不适用于外包开发

在 DevOps 交付管道中，外包团队应被视为供应商或能力提供商。但是，企业应确保供应商团队的实践和流程与其内部项目团队的实践和流程相互兼容。

使用通用发布规划、工作项管理和资产存储库工具，可显著改善业务部门与供应商和项目团队之间的沟通与协作，从而支持 DevOps 实践。通过使用应用程序发布管理工具，可以极大地提高企业在整个发布过程定义和协调所有参与者的能力。

无云技术即无 DevOps

当您想到 DevOps 时，您经常会联想云技术，因为云技术可以为开发人员和测试人员动态调配基础架构资源，快速获得测试环境，而手动请求得到满足则需要等待数天甚至数周时间。但是，云技术并非采用 DevOps 实践的必要条件，只要企业拥有高效的流程，且能够获取资源来部署和测试应用程序变更就可以了。


技术内容

虚拟化技术本身就是可选可不选的功能。如果可以按照必要的速度配置和部署服务器，则持续交付到物理服务器即有可能实现。

DevOps 不适用于大型复杂系统

复杂系统更需要 DevOps 提供的原则和协作功能。这种复杂系统通常具有多个软件和/或硬件组件，其中每个都拥有自己的交付周期和时间。而 DevOps 则有助于协调这些交付周期和系统级发布计划。

DevOps 仅涉及沟通

DevOps 社区的一些成员创造了幽默的术语，例如，*ChatOps*（团队通过互联网中继聊天等通信工具来进行所有沟通工作）和 *HugOps*（DevOps 仅关注协作和沟通）。这些术语源于沟通与协作可以解决所有问题这一错误观念。


牢记

DevOps 依赖于沟通，但更好的沟通结合低效的流程并不会带来更好的部署成果。

DevOps 意味着持续变更部署

这种误解来自仅部署 Web 应用程序的企业。一些公司在他们的网站上自豪地宣称，他们每天都会将应用程序部署到生产环境。然而，对于需要部署复杂应用程序的大型企业而言，这种部署模式不仅不切实际，而且可能会因监管或公司限制而根本无法实现。DevOps 不仅仅涉及部署环节，也绝不仅仅关乎持续部署到生产环境。通过采用 DevOps，企业可以根据需要将应用程序发布到生产环境，而不是依照日历上标记的特定日期执行发布。

Notes

Notes

运行在软件上的世界

DevOps 是任何渴望实现精益和敏捷的企业必不可少的方法。DevOps 帮助您持续实现软件驱动型创新，快速应对不断变化的客户和市场需求。本书可帮助您了解 DevOps 概念以及您的企业如何从中获得切实的商业利益。此外，您还可以了解涵盖整个软件交付生命周期的 DevOps 整体观点在持续交付领域能够为您带来怎样的竞争优势。

Inside...

- DevOps 的功能
- 如何结合使用移动、云和其他技术
- 让 DevOps 与 ALM 流程保持一致
- How to manage multi-tier applications
- IBM 的 DevOps 实际运用

IBM 杰出工程师 **Sanjeev Sharma** 是 DevOps 和云转型思想领袖、IT 架构师、技术高管和技术著作作者。**Bernie Coyne** 是 IBM 内部的一名资深 DevOps 传播者，致力于推广面向 DevOps 的思想领导力文章和解决方案。他在软件开发方面拥有丰富背景，也因此成为了一名演讲者和作者。

ISBN: 978-1-119-52345-1
Print part #: RAM14025USEN-00
ePDF part #: RAM14026USEN-00
Not for resale

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.