

IBM 敬赠

IBM 限量版第 2 版

DevOps FOR DUMMIES®

了解：

- DevOps 的业务需求和价值
- DevOps 的功能和采用方法
- 如何利用云加速 DevOps
- 十个 DevOps 误区



Sanjeev Sharma
Bernie Coyne

[点击此处，深入探讨 DevOps](#)

DevOps

FOR
DUMMIES®

IBM 限量版第 2 版

**作者：Sanjeev Sharma
和 Bernie Coyne**

WILEY

[点击此处，深入探讨 DevOps](#)

DevOps For Dummies®，IBM 限量版第 2 版

出版商：

John Wiley & Sons, Inc.

111 River St.

Hoboken, NJ 07030-5774

www.wiley.com

版权所有 © 2015 by John Wiley & Sons, Inc.

除依据《1976 年美国版权法案》第 107 或 108 条规定允许的情况外，未经出版商事先书面许可，不得以任何方式（包括电子、机械、复印、录制、扫描或其它任何方式）对本出版物中的任何章节进行翻印、传播或将其存储在任何检索系统中。如需申请许可，请与出版商联系，地址：Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030，电话：(201) 748-6011，传真：(201) 748-6008。在线申请网址：<http://www.wiley.com/go/permissions>。

商标：Wiley、For Dummies、Dummies Man 徽标、The Dummies Way、Dummies.com、Making Everything Easier 及相关商业外观是 John Wiley & Sons, Inc. 和/或其关联公司在美国和其他国家/地区的商标或注册商标，未经书面许可，不得使用。IBM 和 IBM 徽标是 International Business Machines Corporation 的注册商标。所有其他商标均为各自所有者的财产。John Wiley & Sons, Inc. 与本书提及的任何产品或供应商无任何关系。

责任限制/免责声明：出版商和作者不对本书内容的准确性或完整性做任何陈述或保证，包括但不限于对特定用途的适用性。销售或促销材料不产生或延长任何担保。本书所含的建议和策略不一定适合所有情况。本书的销售前提是，本书出版商不借此提供任何有关法律、会计或其他专业服务。如需专业帮助，请寻求能够胜任的专业人士的服务。出版商和作者均不为由此产生的损害负责。本书提及某个企业或网站作为引用和/或补充信息的潜在来源并不意味着：本书作者或出版商认可该企业或网站可能提供的信息或可能给出的建议。此外，读者应该意识到，本书所列网站可能在本书写成后的时间里发生改变或消失。

关于我们提供的其他产品和服务的信息，或者如何为您的企业或组织定制 *For Dummies* (傻瓜书) 系列书籍，请联系我们在美国的业务发展部，电话：877-409-4177，电子邮件：info@dummies.biz，网址：www.wiley.com/go/custompub。关于如何为产品或服务申请 *For Dummies* 品牌许可，请联系：BrandedRights&Licenses@Wiley.com。

ISBN：978-1-119-17754-8 (pbk); ISBN：978-1-119-17756-2 (ebk)

美国印制

10 9 8 7 6 5 4 3 2 1

出版商鸣谢

谨此感谢帮助本书成功出版的人士：

项目编辑：Carrie A. Johnson

策划编辑：Katie Mohr

编辑经理：Rev Mengle

业务开发代表：Sue Blessing

项目协调员：Melissa Cossell

目录

引言	1
关于本书	1
本书中使用的图标	2
参考资料	2
第1章：DevOps 是什么？	3
了解 DevOps 的业务需求	3
认识 DevOps 的业务价值	4
增强客户体验	5
提高创新能力	5
更快实现价值	6
了解 DevOps 的工作原理	6
针对类生产环境进行开发和测试	6
利用可重复的可靠流程进行部署	7
监控并验证运营质量	8
放大反馈回路	8
第2章：了解 DevOps 功能	9
DevOps 采用方法	9
规划	10
开发/测试	11
协作开发	12
持续测试	13
部署	13
运维	14
持续监控	14
持续客户反馈和优化	14
第3章：采用 DevOps	15
了解从何处着手	15
确定业务目标	16
确定交付通道中的瓶颈	16
DevOps 中的人员	17
DevOps 文化	17
DevOps 团队	19
DevOps 中的流程	19
DevOps 即业务流程	19
变更管理流程	20
DevOps 技巧	21
DevOps 中的技术	24

基础架构即代码.....	25
交付通道	26
部署自动化和发布管理	28
第 4 章：如何利用云加速 DevOps.....	31
利用云推动 DevOps	32
全堆栈部署.....	34
为 DevOps 选择云服务模型	35
IaaS	35
PaaS	37
了解混合云是什么	38
第 5 章：利用 DevOps 应对新挑战	41
移动应用	42
ALM 流程	43
扩展敏捷性.....	43
多层应用	44
企业中的 DevOps.....	45
供应链	46
物联网	46
第 6 章：让 DevOps 发挥作用：IBM 的故事	49
了解管理层的职责	50
组建团队	51
设定 DevOps 目标.....	51
从 DevOps 转变吸取经验教训	52
拓展敏捷实践	52
利用测试自动化.....	53
构建交付通道	54
快速试验.....	56
持续改进	57
了解 DevOps 结果.....	58
第 7 章：十个 DevOps 误区.....	59
DevOps 仅适合“网上诞生”的组织	59
DevOps 需要运维团队学习编码.....	60
DevOps 仅适合开发和运维	60
DevOps 不适合 ITIL 组织	60
DevOps 不适合管制行业	61
DevOps 不适合外包开发	61
没有云意味着没有 DevOps	61
DevOps 不适合大型复杂系统	62
DevOps 仅涉及沟通.....	62
DevOps 意味着持续变更部署	62

引言

DevOps 是开发 (development) 和运维 (operations) 的缩写，和多数新方法一样，在许多人眼中不过是个时髦词而已。大家都在讨论 DevOps，但并非所有人都了解它是什么。广义地讲，DevOps 是一种基于精益和敏捷原则的方法，企业所有者以及开发、运维和质量保证部门运用此方法持续地交付软件，支持企业更快地抓住市场机遇并缩短融入客户反馈的时间。事实上，企业应用非常多样化，由多个不同的技术、数据库和最终用户设备等组成，只有 DevOps 方法能够成功应对如此高的复杂性。但是，人们在如何运用 DevOps 方面存在分歧。

有人说 DevOps 仅适用于实践者，还有人说它以云为中心。IBM 从广泛的全局视角出发，将 DevOps 视作业务驱动的软件交付方法 – 此方法从创意一直到生产提供了一个全新的或增强的业务功能，通过此功能高效地向客户提供业务价值，并收集客户反馈。为了做到这一点，您需要利益相关方参与进来，而不仅仅是开发和运维团队。真正的 DevOps 方法需要业务部门、实践者、高管、合作伙伴和供应商等配合完成。

关于本书

本书对 DevOps 采取以业务为中心的方法。当今世界瞬息万变，企业必须具备足够的敏捷性和精益性，快速应对客户需求、市场条件、竞争压力或法规要求等方面的变化，因此 DevOps 对于所有这类企业而言不可或缺。

如果您正在阅读本书，想必您已听闻过 DevOps，但期望了解它的具体含义以及您的公司如何从中获得业务优势。本书专为高

管、决策制定者和实践者而著，他们刚刚接触 DevOps 领域，期望获取关于此方法的更多信息，并希望看透围绕这个概念的宣传炒作而触及实质。

本书中使用的图标

您将在本书页边空白处发现多种图标。以下是它们的含义。



“提示”图标表示关于 DevOps 各方面的有用信息。



任何带“牢记”图标的内容都值得铭记于心。



“警告”图标提醒您这是关键信息。



“技术内容”资料超出 DevOps 基础知识的范畴，非必读内容。

参考资料

访问以下网页，获取更多关于 DevOps 和 IBM 的可用方法和服务的信息：

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

第1章

DevOps 是什么?

本章提要

- ▶ 了解 DevOps 的业务需求
- ▶ 认识 DevOps 的业务价值
- ▶ 了解 DevOps 的工作原理

在

不影响日常业务的情况下做出调整总是困难重重，常常需要花费大量的时间学习研究。每当组织采用任何新技术、原则或方法通常是由某种业务需求驱动的。要采用 DevOps，您必须了解 DevOps 的业务需求，包括它所面对的挑战。在本章中，我们将为您提供采用 DevOps 的基础。

了解 DevOps 的业务需求

组织期望开发创新的应用或服务来解决业务问题。他们可能期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）。

不过，许多组织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。虽然多数企业感觉软件开发和交付至关重要，但 IBM 最近一项行业调查显示，仅有 25% 的组织认为他们的团队卓有成效。这种执行的缺失导致组织错失商机。

雪上加霜的是，业务所需交付的应用类型出现重大转变，从记录系统转向交互系统：

- ✓ **记录系统：**传统软件应用是充当记录系统的大型系统，其中包含海量数据和/或事务，旨在维持高度的可靠性和稳定性。由于这些应用不需要经常变更，组织只需每年交付一、两个大型新版本，即可满足客户及其自身业务的需求。
- ✓ **交互系统：**随着移动通信兴起和 Web 应用走向成熟，记录系统得到交互系统的补充，客户可以直接访问和利用交互系统与企业打交道。这类应用必须易于使用、性能卓越并支持快速调整，以应对不断变化的客户需求和不断演变的市场力量。

由于交互系统直接供客户使用，因此需要密切关注用户体验、交付速度和敏捷性——换句话说，就是 DevOps 方法。



交互系统并非孤岛，常常与记录系统绑定，因此交互系统的快速调整势必引起记录系统的调整。事实上，任何需要快速交付创新成果的系统类型都离不开 DevOps。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、大数据和社交媒体，这可能影响各种类型的系统。我们将在第 4 和第 5 章围绕 DevOps 讨论这些新兴技术。

认识 DevOps 的业务价值

DevOps 在整个软件供应链运用敏捷和精益原则。从初步构思、生产发布、客户反馈到基于反馈进行增强，DevOps 支持业务最大限度提高产品或服务的交付速度。



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

DevOps 可在三个方面带来显著投资回报：

- ✓ 增强客户体验
- ✓ 提高创新能力
- ✓ 更快实现价值

我们将在下文讨论这三个方面。

增强客户体验

通过提供增强（即独具特色、引人入胜）的客户体验，可以建立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，这需要通过某种机制从所交付软件应用的所有利益相关方快速获取反馈，利益相关方包括客户、业务部门、用户、供应商、合作伙伴等等。

在当今的交互系统（参阅本章上文介绍的“理解业务对 DevOps 的需求”）世界中，敏捷地响应并适应的能力将增强客户体验并提高忠诚度。

提高创新能力

现代组织利用精益思考方式提高自身创新能力。他们的目标是减少浪费和返工，并将资源转移给价值更高的活动。



A-B 测试是一个精益思维通用实践的例子，组织利用此方法要求一小组用户对两套或更多套拥有不同功能的软件进行测试和评级。然后，拥有更佳功能集的软件将向所有用户推出，不成功的版本将终止。这种 A-B 测试要有现实意义，只有借助高效的自动化机制，例如 DevOps 促成的机制。

更快实现价值

更快实现价值需要创造可将软件快速、高效和可靠地交付于生产的文化、实践和自动化。若将 DevOps 作为业务能力进行采用，它可以提供必要的工具和文化，用于促进高效的发布规划、可预测性和成功。

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、更高效地交付这种价值。

了解 DevOps 的工作原理

DevOps 运动产生了一些不断演变的原则。包括 IBM 在内的多家解决方案供应商开发了自己的变体。不过，所有这些原则采取了整体性的 DevOps 方法，适合各种规模的组织采用。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

我们将在下文详细介绍这些原则。

针对类生产环境进行开发和测试

此原则源自于 DevOps 的左移概念，其中运维问题在软件交付生命周期中提前出现，更靠近开发（见图 1-1）。

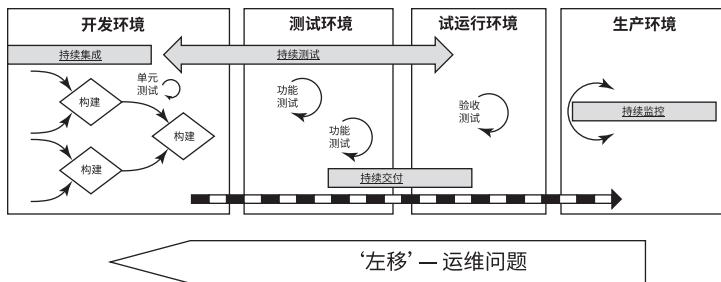


图 1-1：左移概念让运维问题提早在开发生命周期中出现。

目标是让开发团队和质量保证 (QA) 团队依据行为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准备投入部署前及早了解应用的行为和性能。



为解决两大潜在挑战，应在生命周期中尽早让应用暴露于类生产系统中。首先，这让应用可以在接近于实际生产环境的环境中进行测试；其次，这让应用交付流程本身可以提前接受测试和验证。

从运维的角度看，此原则也具有巨大价值。它让运维团队可以提早在生命周期中看到他们的环境为应用提供支持时表现如何，从而让他们可以创造经过微调的应用感知型环境。

利用可重复的可靠流程进行部署

顾名思义，此原则让开发和运维可以支持一个持续到生产的敏捷（或至少是迭代式）型软件开发流程。自动化对于创造迭代、经常性、可重复和可靠的流程不可或缺，因此组织必须打造可支持持续的自动化部署和测试的交付通道。我们将在第 3 章详细讨论交付通道。



频繁部署还让团队可以测试部署流程本身，从而降低发布时的部署失败风险。

监控并验证运维质量

组织通常善于监控生产环境中的应用和系统，因为他们拥有实时收集生产系统指标的工具。但他们的监控方式各自为营、缺乏联系。此原则要求自动化测试提早在生命周期中经常性执行，以监控应用的功能性和非功能性特征，从而让监控在生命周期中提前出现。每当部署和测试应用时，应当收集并分析质量指标。频繁监控对生产环境中可能出现的运维和质量问题提供预警。



这些指标必须通过一种所有业务利益相关方都可以理解和使用的格式进行收集。

放大反馈回路

DevOps 的目标之一让组织能够更快地做出响应和调整。在软件交付中，此目标要求组织快速获得反馈，然后迅速从其采取的每项行动中吸取经验教训。此原则要求组织创造沟通渠道，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顺序发挥作用。
- ✓ 生产部门可以通过增强生产环境发挥作用。
- ✓ 业务部门可以通过修改其发行计划发挥作用。

第 2 章

了解 DevOps 的功能

本章摘要

- ▶ 了解 DevOps 的参考架构
- ▶ 思考 DevOps 的四个采用方法

构

成 DevOps 的功能是整个软件交付生命周期中的广泛功能集。组织从何处着手部署 DevOps 取决于自己的业务目的和目标，包括正在努力应对的挑战，以及需要弥补的软件交付能力不足。

在本章中，您将了解 DevOps 参考架构及其支持业务应用 DevOps 的各种方式。

DevOps 采用方法

通过使用一套首选的方法和功能，参考架构提供了一个成熟解决方案的模板。本章讨论的 DevOps 参考架构可帮助实践者访问和使用必需的指南、指令和其他材料，以构建或设计一个可满足人员、流程和技术要求的 DevOps 平台（参见第 3 章）。

参考架构通过自己的各种组件提供功能。这些功能相应地由单个组件或一组组件共同提供。因此，您可以从 DevOps 参考架构提供哪些核心功能的角度来了解 DevOps 参考架构，如图 2-1 所示。

随着抽象的架构发展成具体的对象，这些功能将由一组得到有效支持的人员、明确的实践和自动化工具提供。

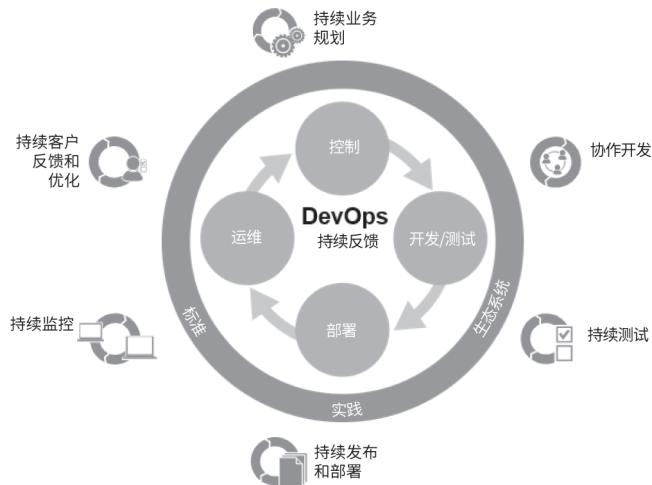


图 2-1: DevOps 参考架构

图 2-1 中显示的 DevOps 参考架构提出了四种采用方法：

- ✓ 规划
- ✓ 开发/测试
- ✓ 部署
- ✓ 运维

本文其余的内容将详细介绍这些采用方法。

规划

本采用方法包含一个专注于建立业务目标并根据客户反馈调整目标的实践：**持续业务规划**。

当今的业务需要敏捷性，能够快速响应客户反馈。实现此目标的关键是组织正确做事的能力。不幸的是，面对当今瞬息万变的商业环境，传统产品交付方法过于缓慢，一定程度上是因为这些方法依赖于定制开发和手动流程，而且各个团队是孤立运维的。进行快速规划和重新规划并最大限度提高价值交付能力所需的信息支离破碎、相互矛盾。企业常常由于无法足够早地接收正确的反馈，从而无法实现正确的质量水平，以真正交付价值。

团队还在艰难地融入可以预示投资优先顺序的反馈，进而作为组织展开协作，在持续交付模型中推动执行工作。对一些团队而言，规划是具有侵扰性并会降低效率的治理开销，而不是支持他们快速交付价值的活动。

更快地交付可实现更高的业务灵活性，但是还必须对速度进行管理，确保您的交付正确无误。如果无法确信业务目标、衡量结果和平台的准确性，就无法快速交付软件。

DevOps 有助于协调这些竞争性因素，帮助团队制定共同业务目标，并根据客户反馈持续调整目标，从而提高灵活性并改善业务成果。与此同时，企业需要对成本进行管理。通过确定并消除开发流程中的浪费，团队可以变得更加高效，并能控制成本。在转向持续交付模型的 DevOps 生命周期各个阶段，此方法可帮助团队在所有这些因素之间实现最佳平衡。



开发/测试

此采用方法涉及两种实践：协作开发和持续测试。因此，它构成了质量保证 (QA) 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，包括业务部门所有者、业务分析师、企业和软件架构师、开发人员、QA 实践者、运维人员、安全专家、供应商和合作伙伴。这些团队的实践者在多个平台上工作，并可能处于多个位置。协作开发通过向这些实践者提供软件开发和交付的一套通用实践和通用平台，支持他们协同工作。

协作开发包含的一项核心功能是持续交付（见图 2-2），在这项实践中，软件开发人员持续或频繁地将其工作成果与开发团队其他成员的成果相集成。

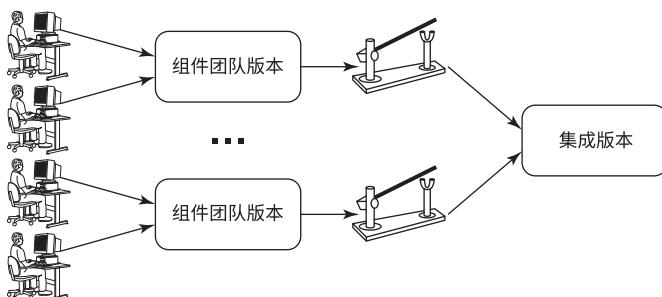


图 2-2：通过持续集成进行协作

持续集成的流行得益于敏捷移动。具体思路是让开发人员定期将其工作成果与所在团队其他开发人员的成果相集成，进而对集成后成果进行测试。对于由多个系统或服务构成的复杂系统，开发商还必须定期将其工作成果与其他系统和服务相集成。定期集成工作成果可以提早发现和暴露集成风险。在复杂的系统中，它还可以暴露已知和未知风险，包括技术和日程相关的风险。

持续测试

持续集成（参见上文）拥有多个目标：

- ✓ 支持持续的代码测试和验证
- ✓ 验证已编写并与其他开发人员的代码和其他组件进行集成的代码，在功能和表现方面符合设计目标
- ✓ 持续测试正在开发中的应用

持续测试意味着在整个生命周期中提早进行持续的测试，从而降低成本、缩短测试周期并实现持续的质量反馈。此流程还被称作左移测试，它强调对开发和测试活动进行集成，确保在产品生命周期中尽早考虑质量问题，而不是留到以后再处理。通过采用自动化测试和服务虚拟化等功能，可以加快此流程。服务虚拟化是一种可模拟生产类似环境的新功能，让持续测试成为可能。

部署

部署采用方法产生了多数 DevOps 基本功能。持续发布和部署可推动持续集成概念进入下一阶段。支持发布和部署的实践也支持交付通道的创建（参见第3章）。此通道可推动人们将软件高效、自动地持续部署到 QA 和生产环境。持续发布和部署的目标是尽快向客户和用户发布新功能。

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



运维

运维采用方法包含两种实践，允许业务监控已发布应用在生产环境中的表现并从客户收集反馈。此数据允许业务敏捷地做出响应，并根据需要调整业务计划。

持续监控

持续监控在交付周期的不同阶段向运维、QA、开发、业务部门人员和其他利益相关方提供关于应用的数据和指标。



这些指标并不限于生产。此类指标让利益相关方可增强或调整要交付的功能和/或交付功能所需的业务计划，从而做出响应。

持续客户反馈和优化

软件交付团队可获得的两类最重要的信息包括：客户如何使用应用，以及客户在使用应用的基础上所提供的反馈。新技术让业务可以在客户使用应用的过程中捕获客户行为和客户难题。此反馈让不同利益相关方可采取适当的举措，以改善应用以及客户体验。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部署环境。此持续反馈循环是 DevOps 的基本组成部分，让业务能够更灵活、更快地响应客户的需求。

第3章

采用 DevOps

本章提要

- ▶ 提高人员效率
- ▶ 简化流程
- ▶ 选择合适的工具

要

想采用任何新功能，通常需要一个涵盖人员、流程和技术的计划。如果不能从所有这三个方面考虑要交付的新功能，就无法成功采用新功能，对于有多个可能位于不同位置的利益相关方的企业而言尤其如此。

在本章中，我们将从人员、流程和技术方面探讨 DevOps。



虽然 *DevOps* 一词让人联想到开发和运维功能，但事实上，*DevOps* 是一种囊括组织中所有利益相关方的企业功能，这些利益相关方包括业务所有者、架构、设计、开发、质量保证 (QA)、运维、安全、合作伙伴和供应商。如果排除组织内部或外部的任何利益相关方，*DevOps* 的实施都是残缺不全的。

了解从何处着手

本节提供了如何开始利用 *DevOps* 的指南，包括打造正确的文化、确定业务挑战，并找到需要消除的瓶颈。

确定业务目标

文化建设的首要任务是让所有人向着共同的方向迈进，向着共同的目标努力工作，这意味着要为团队和整个组织确定共同的业务目标。根据业务成果，而不是相互冲突的团队鼓励举措来激励整个团队至关重要。如果员工了解他们努力奋斗的共同目标及其目标实现进度的衡量方法，那么拥有各自优先事项的团队或实践者就极少遇到挑战。



DevOps 并非目标。它可帮助您实现目标。

第 4 章 和第 5 章重点介绍 DevOps 要应对的多个业务挑战。您的组织可以将这些挑战作为起点，确定组织期望实现的目标；然后可以为这些目标创建一系列的通用里程碑，供不同利益相关方团队使用。

确定交付通道中的瓶颈

交付通道中最大的低效因素归类如下：

- ✓ 不必要的开销（不得不重复传达相同的信息和知识）
- ✓ 不必要的返工（测试或生产环节发现的瑕疵迫使企业将工作重新分配给开发团队）
- ✓ 过度生产（开发的功能可有可无）

基础架构部署是交付通道中的最大瓶颈之一。采用 DevOps 方法可提高应用的交付速度，迫使基础架构加快响应速度。在此，软件定义的环境支持您将基础架构作为一种可编程、可重复的模式加以捕获，从而加快部署速度。查阅本章后面的“基础架构即代码”一节，了解更多信息。

在更深入的层次上，您可能期望利用端到端的均匀流对通道进行优化。各个流程的吞吐量必须相等，以免出现积压问题。为了实现这种平衡，您需要在关键的点上装备仪器或衡量交付通道，从而最大限度减少积压队列的等待时间、优化正在进行的工作，并调整容量和流量。

DevOps 中的人员

本节介绍 DevOps 的人员因素，包括打造必要的文化。

DevOps 文化

从根本上看，DevOps 是一种文化运动，以人为核心。一家组织可能采用最高效的流程或自动化工具，但如果没有人最终执行这些流程并使用这些工具，它们都是毫无用处的。因此，建设 DevOps 文化是 DevOps 采用的核心。



DevOps 文化的特征是在各种角色间实现高度协作、专注于业务而非部门目标、充满信任，并高度重视通过体验进行学习。

打造文化不同于采用流程或工具。它需要（缺乏更好的描述词语）团队成员间的社交工程，并且每个团队成员都具备独特的性情、经历和倾向。这种多元化让文化建设工作充满了挑战和难度。



大规模敏捷框架 (SAFe)、规范敏捷交付 (DAD) 和 Scrum 等精益和敏捷转型实践是 DevOps 的核心元素，如果您的组织已应用这些实践，就可以利用它们帮助您采用 DevOps 文化。

衡量文化

衡量文化工作极其困难。您如何准确地衡量更好的协作或更高的士气？您可以通过调查直接衡量人员的态度和团队士气，但由于团队通常规模较小，所以调查统计的错误率可能比较高。

相反，您可以进行间接衡量，即跟踪开发团队成员多久联络一次运维或 QA 团队成员来解决问题，而不是通过正式渠道或多个管理层。

在所有利益相关方之间进行协作和沟通正是 DevOps 文化。



DevOps 文化建设需要组织的领导者与其团队共同创建一个协作和共享的环境与文化。领导者必须消除自己强加的任何协作障碍。

典型的衡量方法是奖励运维团队在正常运行时间和可靠性方面所做的贡献，奖励开发人员交付的新功能，不过，这会在各个团队之间形成对立竞争关系。运维团队知道，对生产的最佳保护可能就是拒绝任何变更，开发团队基本上没有关注质量的动力。把这些衡量法替换为共同的责任才能快速、安全地交付新功能。

组织领导应通过提高透明度来进一步鼓励协作。创建一套通用协作工具必不可少，对于分散于各地且无法面对面工作的团队尤其如此。让所有利益相关方了解项目的目标和状态，对于打造以信任和协作为基础的 DevOps 文化至关重要。



有时，打造 DevOps 文化需要进行人员调整。应该为不愿改变（即不采用 DevOps 文化）的人员重新分配工作。

DevOps 团队

支持和反对成立专门 DevOps 团队的争论与 DevOps 概念本身一样久远。Netflix 等组织没有单独设立开发团队和运维团队，而是让 NoOps（免运维）团队担负起这两项职责。其他组织通过 DevOps 联络团队取得了成功，解决了所有冲突并推进了协作。这种团队可能是现有的工具小组或流程小组，也可能是与待交付应用有关联的所有团队派出的代表所组建的新团队。

如果选择设立 DevOps 团队，那么最重要的目标是确保其发挥卓越中心的职责，促进协作，而不是增添了一层新的官僚机构，也不应变成独自解决所有 DevOps 相关问题的团队——这种举措将违背采用 DevOps 文化的目的。

DevOps 中的流程

在上一节，我们讨论了人员和文化在采用 DevOps 中的作用。流程可定义这些人员做什么。您的组织可能拥有卓越的协作文化，但如果员工做错误的事情或以错误的方式做事，失败将在所难免。

DevOps 包含众多流程——由于数量过多，本书不一一介绍。本节将根据在整个企业间的采用情况讨论一些关键流程。

DevOps 即业务流程

DevOps 功能会影响整个业务。它提高了业务的敏捷性，并改善了业务向客户交付功能的能力。您还可以进一步拓展 DevOps，将其视作业务流程：为客户实现特定成果（服务或产品）的一系列活动或任务。

在第 2 章介绍的参考架构中，DevOps 业务流程涉及了使用来自创意（通常由业务所有者确定）、开发、测试和生产方面的功能。



虽然此业务流程的成熟度不足以让人们在一套简单的过程流中捕获它，但您应捕获组织已经用于交付功能的过程流。然后，通过改善流程本身和引入自动化（参见本章后面的“DevOps 中的技术”一节），您可以确定需要改善的方面。

变更管理流程

变更管理是一套活动集，通过确定可能发生变更的工作产品以及用于实施此变更的流程，对变更进行控制、管理和追踪。组织所使用的变更管理流程是更广泛的 DevOps 过程流的一个固有组成部分。变更管理可推进 DevOps 收集流程，并对变更请求和客户反馈进行响应的方式。



采用应用生命周期管理 (ALM) 的组织已拥有定义明确、（并可能是）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）
- ✓ 基于角色的工作件访问控制

传统的变更管理方法通常局限于变更请求或缺陷管理，在跟踪变更请求或缺陷之间的事件以及相关代码或要求的能力有限。这些方法不提供跨生命周期的集成工件管理，也不提供可跟踪各类资产的内置功能。不过，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展开协作。

以 DevOps 或 ALM 为中心的变更管理所包括的流程可为所有项目、任务和相关资产提供工作项管理——并非仅限于受变更请求或缺陷影响的流程。它所包含的流程还支持企业将工作项关联到所有缺陷、项目资产，以及由任何相关实践者创建、修改、参考或删除的其他工作项。这些流程让团队成员可以根据角色来访问所有变更相关信息，还支持迭代和敏捷项目开发工作。

DevOps 技巧

以下是采用 DevOps 时需要融入的一些具体技巧：

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

以下各节将详细介绍这些技巧。

持续改善

在真正的精益思想中，流程采用并非一次性的行动，而是一个持续的流程。组织应该有内置的流程，以便随着组织的发展成熟并从所采用的流程中吸取经验教训，从而确定有待改善的方面。许多业务拥有流程改善团队，负责根据观察和所吸取的经验教训对流程进行改善；其他业务允许采用此流程的团队进行自我评估，并确定他们自己的流程改善方法。无论采用什么方法，目标都是实现持续改善。

发布规划

发布规划是一个关键的业务功能，在业务需求的推动下，向客户提供功能以及这些需求的时间表。因此，业务需要一个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，以及这些流程间的端到端可跟踪性。

为完成此任务，如今大多数公司使用电子表格并与整个业务中的所有利益相关方举行会议（通常时间漫长），对所有正在开发中的业务需求应用、其开发状态和发布计划进行跟踪。不过，定义明确的流程和自动化能够消除使用电子表格和举行会议的必要性，并实现简化、（更重要的是）可预测的发布。充分利用精简和敏捷实践还能实现更小、更频繁的发布，从而提高人们对质量的关注。

持续集成

持续集成（如第 2 章所述）为 DevOps 带来了巨大的价值，让大型开发团队能够在多个位置共同处理跨技术组件，从而敏捷地交付软件。它还确保每个团队的工作成果与其他开发团队的成果持续集成，并进而进行验证。因而，持续集成可以减少风险，并在软件开发生命周期中提早发现问题。

持续交付

持续集成自然而然地带来了持续交付实践：此流程将软件自动部署到测试、系统测试、试运行和生产环境。虽然一些组织停止了生产，但采用 DevOps 的组织通常在所有环境中使用相同的自动化流程，从而提高效率并降低不一致流程所引入的风险。

在测试环境中，通过自动处理配置、刷新测试数据，然后将软件部署到测试环境并执行自动化测试，可以缩短测试结果返回到开发环节的反馈周期。



采用持续交付通常是采用 DevOps 最关键的部分。对于许多 DevOps 实践者而言，DevOps 仅限于持续交付，因此多数作为 DevOps 工具推广的工具仅适合此流程。不过，正如您在本书中所看到的，DevOps 的范围更为广泛。持续交付是 DevOps 的基本组件，但不是唯一的组件。



根据组织的业务需求和面对的紧迫挑战，您可以选择使用第 2 章描述的其他流程或采用方法开始 DevOps 采用工作。

持续测试

我们在第 2 章介绍了持续测试。从流程的角度看，您需要在三个领域采用各种流程来实现持续测试：

- ✓ 测试环境设置和配置
- ✓ 测试数据管理
- ✓ 测试集成、功能、性能和安全性

在一个组织中，QA 团队需要确定为每个领域采用哪个流程。他们所采用的流程可能因项目的不同而不同，具体取决于每个测试需求和服务级别协议的要求。例如，相比内部应用，面向客户的应用可能需要更多的安全性测试。相比采用瀑布式方法、数月仅测试一次的项目，采用敏捷方法和持续集成实践的项目更重视测试环境设置和测试数据管理。同样，对于所含组件拥有不同交付周期的复杂应用而言，其功能和性能测试要求不同于简单的单一 Web 应用。

持续监控和反馈

客户反馈的形式多种多样，例如客户提交的工作单、正式变更请求、非正式抱怨以及应用商店中的评分。特别是由于社交媒体和应用商店的流行（参见第 5 章），业务需要定义明确的流程，从一系列来源收集反馈并将其融入软件交付计划。这些流程还应足够敏捷，以适应市场和法规的变化。

衡量流程采用情况

通过观察一系列效率和质量的指标是否随着时间的推移而得到改善，来衡量流程采用的成功度。此类衡量需要两个前提条件：

- ✓ 您必须确定一套合适的效率和质量指标。这些指标对业务而言应具有真正重要的意义。
- ✓ 您需要建立可衡量改进程度的基准。

您可以使用多种定义明确的架构对流程成熟度进行衡量。对于 DevOps 特定的流程，全新的 IBM DevOps 成熟度模型等模型可以评估成熟度。有关 IBM 成熟度模型的更多信息，请访问 ibm.biz/adoptingdevops。

反馈信息还来自于监控数据。此数据来自于运行应用的服务器；来自于开发、QA 和生产环节；或来自于应用中用于收集用户操作信息的嵌入式指标工具。



可能会出现数据过载的情况，因此业务需要数据收集和数据使用的流程，用于改进他们的应用及其运行环境。

DevOps 中的技术

技术让人们可以专注于高价值的创造性工作，同时自动完成各种例行的任务。技术还让实践者团队可以充分利用并扩展他们的时间与能力。

如果组织正在构建或维护多个应用，所有工作都必须具备可重复性和可靠性，同时确保所有应用的质量。对于每个应用的每个新版本或漏洞修复，组织不能从头开始处理。为了保持经济高效和有效，组织必须重用各种资产、代码和实践。

对自动化工作进行标准化也可以提高人员效率（参见本章前面的“DevOps 中的人员”）。组织可能遭遇员工、承包商和资源供应商流失问题；人员可能从一个项目转移到另一个项目。但是一套通用的工具让实践者可以在任何地方开展工作，而新团队成员只需学习一套工具，也就是一个高效、经济、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 核心功能，让组织可以管理环境设置和配置的范围与速度，从而支持持续交付。

围绕基础架构即代码概念发展起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收集，而软件定义环境使用各种技术来定义由多个节点构成的整个系统——不仅包括节点配置，还包括其定义、拓扑结构、角色、关系、工作负载和工作负载策略以及行为。

有三种自动化工具可供管理基础架构即代码：

- ✓ **以应用和中间件为中心的工具：**这些工具通常可以把应用服务器及其运行的应用作为代码进行管理。此类工具是专用工具，并捆绑了支持技术的典型自动化任务库。它们无法执行低级任务，例如配置操作系统 (OS) 设置，但完全可以自动完成服务器和应用级别的任务。
- ✓ **环境和部署工具：**这些工具属于新型工具，可以部署基础架构配置和应用代码。
- ✓ **通用工具：**这些工具并非专门针对任何技术，可以通过编写脚本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，都可胜任。相比以应用或中间件为中心的工具，它们需要更多的前期工作，但是可以处理更广泛的任务。



通过使用 IBM UrbanCode Deploy with Patterns 等环境管理和部署工具，组织可以快速设计、部署和重新利用环境，并帮助加速交付通道。

交付通道

交付通道包含应用从开发一直到生产的各个阶段。图 3-1 显示的是一系列典型的阶段。不过，这些阶段可能因组织的不同而异，还可能因应用的不同而异，具体取决于组织的需求、软件交付流程和成熟度。自动化水平也可能各不相同。部分组织实现了交付通道的完全自动化；由于法规或公司要求，其他组织必须对软件实施手动检查。您不必同时应对所有阶段。首先专注于组织的关键部分，而非面面俱到，然后再逐步扩大，以涵盖所有阶段。

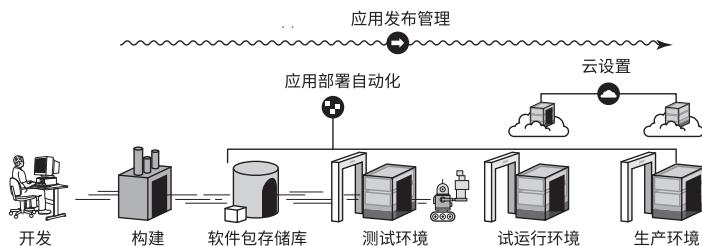


图 3-1：典型 DevOps 交付通道的各个阶段

典型交付通道所包含的各个阶段将在后续章节中阐述。

开发环境

应用开发工作在开发环境中进行，该环境为开发人员编写和测试代码提供了多种工具。除了开发人员用于编写代码的集成开发环境 (IDE) 工具，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、单元测试和项目规划的

工具。根据正在进行的开发类型，此阶段的工具通常是跨平台、跨技术的。

生成阶段

生成阶段中会编译代码，以创建要部署的二进制文件并对其进行单元测试。此阶段可能使用多种生成工具，具体取决于跨平台和跨技术需求。开发组织通常使用生成服务器，不断地生成所需的大量版本，以支持持续集成。

软件包存储库

软件包存储库（又称资产存储库或工件存储库）是一种通用存储机制，可存储生成阶段所生成的二进制文件。这些存储库还需要存储与二进制文件关联的资产，例如配置文件、基础架构即代码文件和部署脚本。

测试环境

测试环境是 QA、用户验收以及开发/测试团队执行实际测试工作的地方。此阶段使用许多不同的工具，具体取决于 QA 的需求。以下是一些示例：

- ✓ **测试环境管理：**这些工具帮助您对测试环境进行设置和配置。其中包括基础架构即代码技术，以及（如果环境在云中）云设置和管理工具。
- ✓ **测试数据管理：**对于任何期望支持持续测试的组织而言，管理测试数据是一项基本功能。可执行的测试的数量和执行频率取决于可供测试的数据数量和数据刷新速度。
- ✓ **测试集成、功能、性能和安全性：**每类测试都有适合的自动化工具。这些工具应与通用测试资产管理工具或存储库集成，该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：**现代应用并非简单、单一的应用。它们是复杂的系统，依赖于其他应用、应用服务器、数据库，甚至是第三方应用和数据源。不幸的是，在测试阶段，这些组件可能不可用或非常昂贵。服务虚拟化解决方案可模拟应用中特定组件的行为（包括功能和性能），从而支持对整个应用进行端到端测试。这些工具可创建运行测试所需的应用和服务的存根（虚拟组件）。应用与这些存根进行交互时，可以测试应用的行为和性能。IBM Rational Test Virtualization Server 提供了此类测试虚拟化功能。

试运行和生产环境

您可以将应用部署到试运行和生产环境中。此阶段所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也可在这些阶段中发挥至关重要的作用，因为这些阶段存在于大范围的环境中。随着虚拟化和云技术的出现，如今的试运行和生产环境可能由数百甚至数千个服务器构成。监控工具让组织可以监控已在生产环境中部署的应用。

部署自动化和发布管理

要对应用从一个阶段自动部署到下一个阶段的工作进行管理，需要专门的工具，其中一些工具我们将在后面章节进行讨论。

部署自动化

部署自动化工具是 DevOps 领域的核心工具。此类工具可执行编排好的部署工作，并在生成和交付通道的任何阶段跟踪哪个版本部署到哪个节点上。它们还可以管理所有阶段的环境配置，而应用组件必须部署到这些环境中。

衡量技术采用情况

衡量投资回报工具和技术相当简单。通常，您可以衡量自动化所实现的效率。此外，自动化工具让您可以提高任务的可扩展性和可靠性，而这并非总是可以通过手动工具实现的。最后，使用一套集成工具促进协作、可跟踪性，并提高质量。

部署自动化工具管理要部署的软件组件、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，以及对要部署这些组件的环境进行配置变更。这些工具还可以捕获流程并实现流程自动化，从而实施这些部署和配置变更。IBM UrbanCode Deploy 就是这种部署自动化工具。

发布管理

要对与每次发布相关的发布计划和部署工作进行编排，需要跨业务、开发、QA 和运维团队进行协作。发布管理工具让组织可以规划和执行发布工作，为发布的所有利益相关方提供单一协作门户，并在生成和交付通道的所有阶段为发布及其组件提供可跟踪性。IBM UrbanCode Release 提供了这种发布管理功能。

第4章

如何利用云 加速 DevOps

本章提要

- ▶ 利用云推动 DevOps
- ▶ 了解全堆栈部署
- ▶ 了解不同的云服务模型
- ▶ 探索混合云

DevOps 和云是彼此的催化剂和推动力。随着组织不断采用云，利用云托管 DevOps 工作负载的价值主张变得不证自明。云平台带来了灵活性、弹性、敏捷性和服务，支持对云上托管的应用交付通道进行简化。组织可以随时根据需要对从开发、测试到生产的所有环境进行设置和配置。此流程最大限度地减少了交付流程中与环境相关的瓶颈。组织还寻求利用云平台降低部署和测试环境的成本，或向其实践者提供现代的精简开发者体验。对于为 DevOps 提供和采用云而言，这些因素构成了极具说服力的商业论证。

本章探讨适合 DevOps 的不同云模式，并剖析 DevOps 作为云中工作负载的价值主张。

利用云推动 DevOps

DevOps 的主要目标是最大限度减少交付管道中的瓶颈，并提高它的效率和精益性。环境可用性和配置是组织遇到的最大瓶颈之一。对于实践者特别是开发和测试人员而言，通过正式的工作单流程来申请环境并不鲜见，这个流程请求可能需要数天，甚至是数周时间才能完成。

DevOps 基本原则之一是在类似于生产的环境中进行开发和测试。除了环境可用性瓶颈，另一大挑战是可用环境与生产环境不匹配。这种不匹配可能只是操作系统 (OS) 或中间件级别的环境配置存在差别，也可能是开发环境中的操作系统或中间件类型与生产环境中的完全不同。



缺乏环境导致实践者可能需要长时间等待。开发环境和生产环境不匹配可能带来重大的质量问题，这是因为开发人员无法验证处于开发状态中的应用在生产环境中表现如何，甚至可能无法确定能否通过部署到测试环境时所使用的流程，将此应用部署到生产环境。

云通过下述方式解决这些问题：

- ✓ 云平台上快捷的环境设置可向实践者提供自助服务，提供随需应变的环境可用性和访问。
- ✓ 能够根据需要动态设置和取消设置这些环境，可以减少对永久性静态测试环境的需求，进而改善环境管理并降低成本。
- ✓ 能够充分利用“模式”技术，设置与实践者需求相匹配的环境以及更重要的生产类似环境。此技术支持组织将环境作为软件进行定义和版本控制。

- ✓ 从自动化的角度看，随着 IBM UrbanCode Deploy 等应用部署自动化技术的出现，只需一款工具即可随需应变地设置云环境，并将正确版本的应用部署到这些环境中。这些技术还可以快速配置环境和应用，使之满足实践者的需求。
- ✓ IBM Rational Test Virtualization Server 等服务虚拟化技术可与云环境配合使用，支持对需要测试的服务进行模拟，无需设置真正的服务实例。

图 4-1 显示了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。

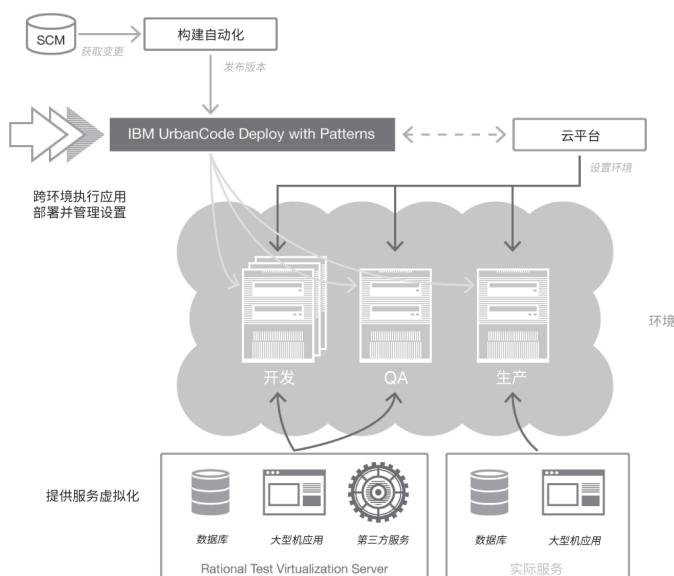


图 4-1：云中的端到端开发/测试



没有 DevOps 的云意味着无法充分利用云的所有优势。通过采用将环境托管于云中的 DevOps，可以让交付软件应用的组织获得云的所有优势。

全堆栈部署

部署云应用包括部署应用，以及对运行应用的云环境进行配置。您可分别执行这两个任务，也可以合并执行，后者称为全堆栈部署。我们将在本节详细探讨这两种方法。

第一种方法将云环境设置与应用部署分开。在此情形中，云环境以及其中部署的应用不存在单一编排点。应用部署自动化工具仅仅把云环境看作静态环境。此情形无法充分利用云中部署的全部优势。

第二种方法把部署自动化工具当作单一编排工具，执行云环境的设置和所设置环境中应用的部署工作。为了做到这一点，您可以创建“蓝图”，收集云环境定义和拓扑结构，然后将应用组件和配置映射到在云环境中定义的节点。

您可使用 IBM Virtual System Patterns 和 OpenStack HOT 等多模式技术将云环境定义为模板。IBM UrbanCode Deploy with Patterns 等部署自动化工具可以使用这些蓝图交付全堆栈设置。这包括设置蓝图中定义的云环境，以及把应用部署到已设置好的环境中。设置好环境后，可将后续的应用、配置和内容变更作为更新部署到云环境中。

组织也可以选择始终采用全堆栈部署，其中环境和相关应用始终作为单个可部署资产来设置。在这种情况下，不对现有环境实施更新。

为 DevOps 选择云服务模型

采用云时，首先期望确定您打算交付云给平台的责任范围，以及您期望自己承担什么责任。云包含两种主要服务模型：基础架构即服务 (IaaS) 和平台即服务 (PaaS)。

IaaS

如果按照 IaaS 服务模型采用云，那么云平台将管理底层基础架构，并提供功能和服务来支持您管理所有虚拟化基础架构。操作系统、中间件、数据和应用的安装、打补丁和管理工作仍是用户的责任。

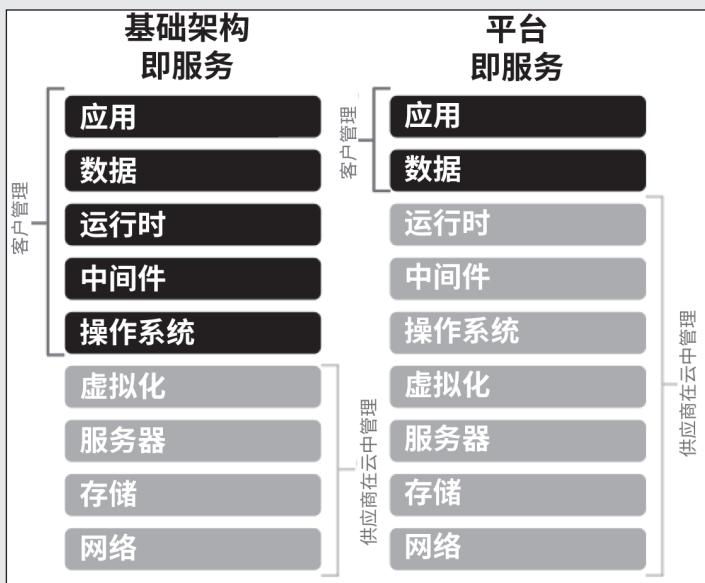
对于在云中将 DevOps 作为工作负载来采用的环境中，采用哪个云服务模型的决策将决定如何采用 DevOps。对于 IaaS 服务模型，用户组织负责管理整个交付管道。交付管道的所有工具和集成工作变成用户组织的责任，其中包括获得合适的工具集以及确保对工具集进行集成，以构建交付管道。此外，他们需要确保开发团队和运维团队依据 DevOps 文化进行协作。仅仅采用了云平台，并不意味着不再需要消除开发人员（负责交付代码）和运维团队（负责交付基础架构，现在是基于云的服务）之间的责任孤岛。

虽然云在向应用交付团队提供 IaaS 方面增添了巨大价值，但团队仍需具备合适的 DevOps 功能，以交付令人满意的 DevOps 价值。

权责分离

在 IaaS 云上利用 DevOps 时，一个关键问题是在云平台和应用部署工具之间定义权责分离。什么工具负责什么工作？一个简单的方法是从云堆

栈中慢速和快速移动资产的角度来看待这个问题。下面的图像显示的是从操作系统、存储和网络层，最终一直到应用的应用体系的不同层。



应用、数据和中间件配置层属于快速移动层。这些层变更频繁，因为应用及其数据和使用方法在不停地进行迭代。对于仍在开发中的应用，变更速度可能非常高。该体系的较低层包括中间件（应用服务器、数据库等）、操作系统和存储，它们的变化不频繁。对于仅仅

影响应用及其内容或配置的简单变更，如果更新并重新设置所有体系，那么效率是很低下的，因此，有必要在应用部署和云管理工具之间分离快速与慢速移动层的职责。快速移动层由应用部署工具管理并使之实现自动化，慢速移动层则由云平台所提供的云管理软件来负责。

PaaS

如果采用 PaaS 云模型，您作为用户的唯一责任变成了应用和数据。所有其他功能都作为服务由云平台提供。结果是大幅改进了应用交付团队的实践者体验。应用开发和测试工具现在作为该平台上的服务来提供，实践者可以访问该平台。应用交付组织不再负责管理交付通道。相反，它嵌入到 PaaS 中，允许实践者将全部精力集中在快速交付应用方面。对实践者而言，开发和测试工具以及基础架构设置全部以服务的形式提供，让实践者可以专注于应用交付的核心职责。



IBM Bluemix 是一种 PaaS。IBM 及其合作伙伴管理该平台及其提供的服务。该平台嵌入了 IBM DevOps Services——为团队提供所有功能以采用 DevOps 的一套服务，更确切地说是作为一套服务的应用交付通道。应用交付团队可以使用这些服务，同时无需担心这些服务是如何托管并交付给他们的。DevOps 服务包含：

- ✓ 基于 Web 的集成开发环境 (IDE) 即服务
- ✓ 生成即服务
- ✓ 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

在从开发、测试、试运行，一直到生产的交付生命周期内，该平台还为不同环境中运行的应用提供了可扩展的运行时环境。

了解混合云是什么

在云领域中，混合云已变成极为普通的术语。它可能被过度用于描述多种云情形，其中多种云技术并存或云和物理基础架构并存。要定义混合云，一种简单的方法是首先研究大量的云情形：

- ✓ **云和物理基础架构：**这是极其常见的混合云情形。除非组织诞生于云中，否者这实际上是默认情形。任何组织目前都有运行在现有物理基础架构上的工作负载和应用。在许多情况下，其中一些应用会继续运行在物理基础架构上。典型示例包括大型机应用和数据密集型记录系统应用，由于技术或成本限制而无法将它们迁移到云中。即使组织打算将所有工作负载迁移到云中，也不可能在一夜之间完成，可能在较长时间内物理和云基础架构都将并存。
- ✓ **内部和外部云：**在此情形中，组织可能为部分应用和工作负载采用外部云（公共或虚拟私有），其余则采用内部（私有）云。例如，一家组织将低成本外部云用于开发环境，位于其数据中心内部的自助管理云则用于生产工作负载。
- ✓ **IaaS 和 PaaS：**在此情形中，客户可能将 PaaS 云服务模型用于部分工作负载，例如最新的创新互动参与系统应用，并将 IaaS 用于更加传统的记录系统工作负载。

对于 DevOps 采用而言，混合云的存在带来了新的挑战，因为它导致应用交付管道延伸到了复杂的混合云和物理环境。这类混合云环境的示例包括：

- ✓ 组织可能选择将公共云用于开发、测试和其他非生产环境，并将内部云甚至是物理基础架构用于生产环境。

- ✓ 组织可能将部分互动参与系统应用部署到云环境，同时为核
心业务应用提供后端服务的记录系统应用可能仍驻留在物理
基础架构（例如大型机）上。
- ✓ 组织可能利用公共 PaaS 实验各种创新的应用，并期望在实验
成功后将其迁移到私有云。
- ✓ 组织可能期望能够在多个云平台间移植应用工作负载，从而
确保不会发生供应商锁定问题，同时提供跨多个云供应商部
署关键工作负载的能力。

利用混合云方法采用 DevOps 的核心要求是：需要在多个云和物
理环境间部署应用。IBM UrbanCode Deploy with Patterns 等应
用可利用应用蓝图，将应用和配置映射到多个环境、物理和云，
从而支持在复杂的混合云环境间自动部署应用。

第5章

利用 DevOps 应对新挑战

● ● ● ● ● ● ● IBM DevOps Hotline (China): 400-668-0529 ● ● ● ● ● ● ● ● ●

本章提要

- ▶ 支持移动应用
- ▶ 应对 ALM 流程
- ▶ 扩展敏捷性
- ▶ 管理多层应用
- ▶ 了解企业中的 DevOps
- ▶ 配合供应链
- ▶ 浏览物联网

DevOps 起源于网上诞生的公司（即在互联网上创办的公司），比如 Etsy、Flickr 和 Netflix。这类公司处理超大规模的复杂技术难题，却拥有相当简单的架构；与此不同的是，一些大型企业围绕旧式系统和/或通过收购与兼并实现发展，所拥有的复杂多技术系统必须协同工作。雪上加霜的是，移动之类的新技术和软件供应链等应用交付模型给现代企业提出了新要求。

本章将探讨 DevOps 可以帮助企业解决的一些难题。

移动应用

在企业中，移动应用通常不是独立式应用。它们在移动设备上的业务逻辑极少，更多是作为企业已使用的多种企业应用的前端。这些后端企业应用可能包括事务处理系统、员工门户和客户获取系统。移动开发和交付的复杂度高，需要以协同、可靠和高效的方式交付一套依赖性服务。

对于企业移动应用而言，发布周期和新功能发布需要与同移动应用交互的企业应用和服务进行协调。因此，DevOps 采用应包含移动应用团队并将其视作一等公民和参与者，此外还应包含其余企业软件开发团队。

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。大多数移动应用都不能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。Apple 的 App Store 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者或供应商的应用）。Research In Motion、Google 和 Microsoft 等设备制造商曾一度允许直接安装应用，但现在也采用 Apple 的模式。

这种情况为部署流程增添了一个异步步骤。开发者将无法再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。持续交付变成提交和等待。不过，持续部署、开发和测试仍然可行，其中测试环境是运行应用的设备的模拟器或一系列物理设备。



全球 80% 的企业数据产生于大型机，70% 的事务接触到大型机。通过开启通往这些大型机功能的移动路径，您可以转变自身开展业务以及与客户互动的方式，但实现起来并非易事。您可能会遇到技能不足、组织孤岛和多种平台问题，而这可能导致漫长的发布周期、不必要的延迟和资源浪费。为了向企业应用提供移动访问能力，企业采用 DevOps 软件交付方法，该方法专注于速度和效率，同时不会牺牲稳定性和质量。



没有具体的 DevOps 概念或原则仅适用于移动应用。不过，由于固有的短开发生命周期和快速变更，移动应用增加了对 DevOps 的需求。

ALM 流程

应用生命周期管理 (ALM) 是一套用于管理应用生命（从创意（业务需求）到部署并最终进行维护的应用）的流程。因此，如果把 DevOps 作为端到端的业务功能考虑，将使 ALM 成为支持 DevOps 流程的基础概念。DevOps 扩大了 ALM 的范围，将业务所有者、客户和运维纳入该流程中。

DevOps 测试/开发采用方法（参阅第 2 章了解更多信息）与一些传统 ALM 功能实现了最高的一致性，这些功能包括需求管理、变更管理、版本控制、可跟踪性和测试管理。不过，跟踪和规划等 ALM 功能属于“规划”采用方法的组成部分，而仪表盘和报告属于“运维”采用方法。

扩展敏捷性

精益和敏捷开发是 DevOps 方法的基础 —— 更高效的团队实现成本节省是成果之一。高效、可重复的最佳实践可以缩短开发周期，提高团队的创造性和响应速度，进而提升客户价值。将精益和敏捷

原则从开发团队扩展到一个大型团队并横跨整个产品和软件交付生命周期是 DevOps 方法的核心。

许多团队已采用敏捷原则，并希望将其现有流程扩展为 DevOps 采用的组成部分。有许多流行框架可用于帮助扩展敏捷性。这些框架包括大规模敏捷框架 (SAFe) 和规范敏捷交付 (DAD)。部分组织还有效地将 Scrum 流程扩展到超大型团队。这些框架的目的是提供在企业级别采用敏捷性的方法。这意味着不仅要考虑代码开发，还应包含架构、项目融资以及管理所需流程和角色的治理，并应用在团队级别表现良好的同一精益和敏捷原则。无论用于扩展敏捷性的框架如何，您都应采用这些基本的敏捷原则，并应用最佳实践来利用它们提高整个企业的效率和有效性。

多层应用

在典型的大型 IT 部门，横跨多个平台的多层应用并不罕见，它们都有各自的独特开发流程、工具和技能要求。这些多层系统通常集成 Web、桌面上的应用，以及前端和后端系统上的移动应用，比如套装应用、数据仓库系统、大型机上运行的应用和中端系统。多层系统的组件可能分布于不同的平台上，即便是最训练有素的 IT 组织面对这些组件的发布管理和协调工作也会不知所措。



明智做法是在所有开发阶段采用自动化、一致的构建、配置和部署流程。这种方法可确保您构建所需的所有组件——并且只是您所需要的组件。随着变更引入以及项目在测试、QA 和生产周期中推进，它还能确保应用保持完整。IBM UrbanCode Deploy 拥有一个应用模型，可以帮助实现多层应用复杂部署的自动化。

根据平台为不同团队准备不同工具是如今多平台、多供应商世界面临的现实。鉴于此，IBM Jazz 等开放平台支持集成不同的工具，从而提供统一解决方案。一致部署实践有助于确保团队使用可靠、可重复的跨平台部署，进而提供真正的业务价值。

企业中的 DevOps

如今的企业依赖于 IT 交付软件的速度。这些企业通常管理部署在大型机和中端系统上的记录应用的系统（内部开发或套装应用）。它们面临众多挑战：

- ✓ 监管障碍
- ✓ 流程复杂性
- ✓ 技能不足
- ✓ 组织孤岛
- ✓ 导致漫长的发布周期、不必要延迟和资源浪费的平台和工具

企业层面的 DevOps 支持规划、开发、测试和运维利益相关方在组织内持续交付软件。从移动到大型机，企业如今部署的应用具有真正的跨平台性。DevOps 开发方法运用精益原则打造高效、有效的交付通道，确保应用的开发、测试和交付方式能够提高质量、加快速度并降低开发成本。

如今企业具有真正的跨平台性质，移动、云计算、分布式和大型机应用应运而生，它们都需要进行创建、集成、部署和运维，这使得 DevOps 提供的效率、简化和协作成为关键的竞争优势。



供应链

随着企业越来越多地借助外包和战略合作伙伴获得技能和能力，软件供应链正在变成标准。供应链是一个由组织、人员、技术、活动、信息和资源构成的系统，涉及将产品或服务从供应商转移给客户。供应链中的不同供应商可能属于企业的内部或外部。

在采用供应链模式交付软件的组织中，采用 DevOps 可能存在难度，因为供应商之间的关系更多是由合同和服务级别协议管理，而不是协作和沟通。不过，此类组织仍然可以采用 DevOps。核心项目团队保留规划和衡量功能的所有权，其他功能则在其他供应商之间共享。在交付通道中，不同供应商可能拥有通道的不同阶段。因此，使用通用工具集和通用资产存储库是必要条件。例如，工作项管理工具提供报告，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。使用通用资产存储库提供了让资产在整个通道传递的机制，从而支持持续交付。

物联网

DevOps 的下一个重大动向是拓展到系统或嵌入式设备领域，这通常被称作持续工程。互联网起步之初，它所共享的数据多数是由人生成的。如今，无数接入互联网的设备（比如传感器和致动器）生成的数据量远多于人生成的数据量。这种由互联网上互联设备构成的网络通常被称作物联网。

在该领域，DevOps 可能更加不可或缺，因为这些硬件及其运行的嵌入式软件具有相互依赖性。持续工程体现了 DevOps 原则，确保交付给设备的嵌入式软件是具备正确工程规范的高品质软件。

持续工程中的“运维”由硬件或负责为设备设计和制造定制硬件的系统工程师替代。硬件和软件开发需要遵循不同的交付周期，而开发测试团队和系统工程师之间的协作对于确保以协作方式开发和交付软硬件至关重要。开发测试团队对于持续交付和测试的需求维持不变。模拟器被用于在开发过程中测试软硬件。

反模式

在现实世界中，采用 DevOps 原则总是存在限制。其中一些限制是由于企业所处的行业和环境，比如合规性、复杂的硬件系统或不成熟的软件交付能力。在这种情况下，采用 DevOps 需要考虑反模式（低效或反生产模式），组织可能无法接受反模式，具体取决于业务需求。

Water-SCRUM-fall

全球调研和咨询公司 Forrester (www.forrester.com) 创造出新的术语 Water-SCRUM-fall（瀑布式 SCRUM），用来描述敏捷软件开发方法采用的当前状态。从 DevOps

角度看，这意味着开发团队可能采用敏捷实践，而其周围的团队可能仍采用不支持持续交付的手动瀑布式流程。在多家企业中，这种局面是企业文化造成的。采用 DevOps 的公司必须在更广泛的 DevOps 实践中嵌入手动流程。

NoOps

在NoOps（无运维）组织中，运维不再是独立部门，其职责并入开发部门。互联网电视提供商 Netflix 是这种方法的倡导者。NoOps 可能适用于一些组织，但需要等待一段时间，以确定该组织模式是否会有更广泛的实践诉求。

第6章

让 DevOps 发挥作用： IBM 的故事

本章提要

- ▶ 了解管理层的职责
- ▶ 组建团队
- ▶ 设定 DevOps 目标
- ▶ 留心 DevOps 转变
- ▶ 从 DevOps 结果吸取经验教训

DevOps 正在部署到整个 IBM 公司，而且还在有规律地发展演变。此举是在 DevOps 方法采用取得成功的基础上做出的。DevOps 方法在 IBM 软件部 (SWG)，率先采用，现已应用于 Watson、Tivoli、Global Business Services 和其他事业部。本章介绍 IBM SWG 的 IBM Rational 协作生命周期管理产品团队采用 DevOps 功能的案例。

这项软件交付工作的独特之处在于它的开放式开发方式 —— 软件交付团队在 jazz.net 上交付所有开发工件和进行中的工作，包括所有详细工作项。该网站对外开放，所有注册用户均可查看规划的工作、进行中的工作，以及为软件产品完成的所有开发工作的历史。



了解管理层的职责

文化是组织的隐性脉络。它基于管理人员和员工所形成的价值观和行为方式。许多时候，您实际上并不了解组织的文化，直到您着手实施重大变革。一些怀疑论者持观望态度，以确定这是否是昙花一现的时尚。领导者将会出现。有必要创建一种方法来了解这些动态情况并确定构成元素，从而让您可以消除真正的障碍。



为了应对这种文化动态局面，IBM SWG 管理层运用了多种方法：

- ✓ **选择合适的领导者：**领导者的任务是汇总不同的观点，以开始为团队带来一系列共同目标、障碍、流程变更以及从何处着手的决策。
- ✓ **让利益相关方参与：**对这些变更的支持必须来自于横跨不同开发领域的领导、管理层和个人贡献者。必须包含业务利益相关方、架构师、开发人员、测试人员和涉及的运维人员，以及这些领域中倡导变革的指定领导者。
- ✓ **衡量改进和成果：**拥有一套融入所需效率和业务成果的关键指标至关重要。应为这些目标和衡量方法设定高标准，同时确保有相应人员对此负责，而且不能撤销。
- ✓ **利用早期的成功营造动力：**通过了解原有的低效因素和衡量改进后在各领域取得的进步，来为变革营造动力。
- ✓ **沟通与倾听：**作为领导者，需要了解变更是如何在团队扎根的真正动态情况。花些时间与技术团队、管理人员和业务主管进行一对一对话和定期的面对面互动，这有助于衡量团队的接纳情况和他们对障碍的观点，并让管理人员有机会分享到有关优先级和进展状况的观点。

如果您是一名高管，您应为团队提供支持，尽可能去了解和帮助消除障碍。作为整个团队的运营，拥有明确的业务目标，是让所有人团结一致、为共同目标而奋斗的必要条件。

组建团队

IBM SWG Rational 协作生命周期管理产品团队隶属于一个更大的集团，该集团从事超过 80 种软件开发工具的开发，涉及软件交付规划、软件开发、应用部署、软件质量管理以及应用监控和分析。

IBM SWG 产品团队是一个大型全球组织，拥有四个核心产品团队，办公地点遍布 10 个国家/地区的 25 个位置。采用 DevOps 方法前，该组织按照年度发布日程表开展工作，其中包括 3 到 6 个月的额外开发周期，用于最终确定哪些内容会进入年度发布日程。

设定 DevOps 目标

IBM SWG 团队认为，他们对市场变化和客户需求变化做出响应的速度过慢。该团队决定缩短交付周期，包括开发和测试阶段，以及与业务利益相关方和客户的协作和互动。目标是努力将一年一次的发布变成每季度一次。

除了需要加速开发以更频繁地交付新功能，该团队还必须加快前进步伐，以支持云交付模式、移动开发、移动测试和其他功能，从而应对技术转变。该团队选择采用 DevOps 原则和实践来变革软件开发方式，进而更早、更频繁地向客户交付价值。

实施如此规模的变革需要调整组织的文化，因此成立了四个工作组，其成员是管理团队成员和技术负责人。这些工作组从头至尾

彻查了软件交付流程，并肩负起改变工作方式的重任。创建了一套具体的衡量方法和行动计划，用于解决开发流程中的关键问题。此外，还成立了持续交付倡导者团队，其中包含一名宣讲师，负责在整个组织内培训团队并分享最佳实践。

IBM SWG 团队开始采用 DevOps 的第一步是确定下述目标：

- ✓ 简化流程并引入新方法
- ✓ 利用工具来实现一致化、与其他团队的可扩展化以及可跟踪化和衡量指标等目标。
- ✓ 发展文化以持续改进

从 DevOps 转变吸取经验教训

本节介绍 IBM SWG 团队为促进 DevOps 转变而采取的举措。

拓展敏捷实践

现有的敏捷实践得到了拓展，突破了开发和测试的范畴，涵盖了客户、业务利益相关方和运维人员，从而削减孤岛并改进成果。这个更广泛的敏捷模型让团队可以协同工作，利用一套集成到每个步骤的流程开发一致、高品质的软件，从而交付业务价值。

“同一团队”方法得到运用，实现了产品管理、设计和开发的整合。该开发团队包含开发经理和团队主管等传统角色，同时还引入了运维管理人员和架构师，用于支持端到端的生命周期战略。

还提供专用资源，用于对团队进行跨组织敏捷和持续交付方面的培训和指导。通过专注于功能而非产品组件，有助于打破传统孤岛边界，并借助每个冲刺和自动化实现交付就绪。通过分派专门开发经理，进一步壮大了这些功能团队。在整个开发组织内定期

举行 Scrum 会议，以确定并解决阻滞问题、跟踪关键指标、利用实时仪表盘数据并传达重要信息。

通过开发优先级来改善市场变化的时效性，并成立战略产品委员会，成员包括产品管理人员、开发总监、架构师和业务所有者。他们的职责包括：

- ✓ 分配并保障资金，确保项目成功
- ✓ 推动、协助并支持项目执行
- ✓ 为业务制定长期愿景和方向
- ✓ 为用于年度发布的重大事件和用户案例划分优先级，与长期愿景保持一致

利用测试自动化

为了消除传统上漫长的后端测试周期并改善发布质量，通过使用自动化和虚拟化来进行敏捷持续测试方案。此外还建立了一种工作节奏，包括最终生成演示的四周迭代，以及最终生成客户可用版本的四周里程碑。各个里程碑后的回顾和对技术责任的了解有助于消除未来迭代中的浪费情况。IBM SWG 团队的格言是“尽早测试，经常测试”。

该团队采用了下述测试自动化最佳实践：

- ✓ 自动处理重复性和劳动密集型测试。
- ✓ 自动处理经常发现漏洞的方面。
- ✓ 对每次构建执行自动化操作，尽早并经常执行。
- ✓ 创建自动保持用户界面 (UI) 变更的一致性 —— 使用可将 UI 与测试分离的框架。

- ✓ 让创建、交付和维护自动化变得更简单，建立强大的功能团队所有权。
- ✓ 把自动化开发工作安排到您的计划之中，并确保开发人员有时间处理该工作。
- ✓ 制定指标，让您可以评估自动化是否有用（您无法改善您无法衡量的内容）。
- ✓ 在自动化发现漏洞的情况下持续进行重新评估，反之则重构自动化。

为了支持测试自动化，该团队部署 IBM Rational Test Workbench 来进行功能和性能测试；为了支持更频繁的测试，实现版本部署的自动化至关重要。利用 IBM UrbanCode Deploy，该团队通过自动化版本部署削减了 90% 的测试开发成本，这种部署包括让任何必要的应用和数据库服务器配置设定实现自动化。

构建交付通道

IBM SWG 团队决定构建一个充分利用“工具即服务”理念的交付通道，让开发人员只需约 60 分钟即可完成代码提交、测试和生产环境部署，以往则需要一到两天时间。该流程减少了返工需求并最大限度地提高了生产力。

在部署过程中，该团队发现持续交付通道需要采用下述最佳实践：

- ✓ 左移测试和尽可能多的自动化
- ✓ 在所有地方使用相同的部署机制
- ✓ 努力保持持续的交付就绪状态
- ✓ 将基础架构视为代码

在图 6-1 中，您可以看到作为 IBM SWG 团队所采用持续交付通道的组成部分而提供的产品和功能。

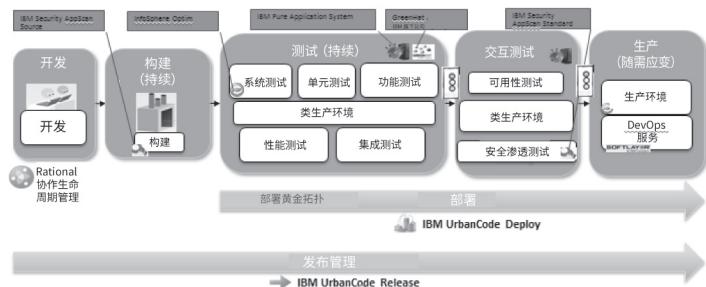


图 6-1：持续交付通道



要实施持续交付通道，一个不可或缺的关键最佳实践是“将基础架构视为代码”。这意味着，开发人员可以编写脚本，并将其作为应用代码的组成部分，对应用所需的基础架构进行配置。过去，这通常由系统管理员或运维人员完成，而现在控制及其提供的效率可以直接由开发人员完成。Puppet、Chef 和 IBM UrbanCode Deploy with Patterns 等新型基础架构自动化工具让基础架构即代码变成现实。

IBM SWG 团队现在将基础架构视为代码，并遵循下述最佳实践：

- ✓ 将模式定义、脚本包和服务视为代码。
- ✓ 对所有内容实施版本控制。
- ✓ 把拓扑模式自动部署到云中。
- ✓ 跨多个云环境管理模式版本。
- ✓ 自动测试模式。
- ✓ 清理目录资源以避免无序扩展。

快速试验

持续交付概念不仅包括持续集成和持续部署等软件开发活动，还包括更基础的学习活动，而这最好通过频繁试验和衡量结果来实现。

当特性和功能加入应用时，您永远无法确定客户是否将收到期望或预期的收益。这就是为什么 IBM 需要尽早并频繁地试验，从客户那里收集对他们实际可行的反馈，并放弃收益甚微或起妨碍作用的功能。图 6-2 中的简图说明了该策略。



图 6-2：了解假设驱动开发

IBM SWG 团队学习到大量关于频繁试验的知识，并制定了下述最佳实践：

- ✓ 建立指标和成功/失败标准。
- ✓ 通过试验确定可行功能 —— 对小部分用户执行小测试，帮助确定功能的有用性。
- ✓ 持续运行多个试验。
- ✓ 依据事实快速制定决策。
- ✓ 加速交付，从而可以加速试验。

- ✓ 制定可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）。
- ✓ 为相关项目同时采用两种方法：对基于云的项目进行试验，并使用试验数据引导该项目的方向，以及相关的内部项目。

持续改进

IBM SWG 团队希望创造持续改进的文化，并充分利用有效性和效率衡量方法，以确保他们确实实现了改进。该团队像管理敏捷项目一样管理他们的持续改进工作。他们通过跟踪成熟度目标、难题以及用于解决问题的相关改进举措，对持续改进提供支持。他们像跟踪其他开发项目一样跟踪持续改进工作，确保这项投资得到广泛了解。成熟度目标（例如功能）可能需要一个或多个季度才能真正开发和采用。较大难题可能需要数月时间才能缓解或消除。但在任何情况下，应对具体改进举措进行调整，以在一个月内交付。

IBM SWG 团队通过回顾实现持续改进的制度化。回顾是指定期审视表现好的地方、表现差的地方，以及需要采取的改善举措。如果您不进行回顾，则意味着软件达到一定程度的完善性，然而事实并非如此。在大型团队中，您将拥有一个回顾层次结构。在 IBM SWG 团队中，每个组件团队执行回顾，该回顾作为应用级回顾的输入内容，然后应用级回顾作为更高的解决方案级回顾的输入内容。将回顾过程中发现的问题作为难题记录在案，一同记录的内容还包括用于减少或缓解难题而采取的相应改进举措。

为了确保团队不断改善，IBM SWG 团队制定了业务指标和运营指标，用于衡量 DevOps 转变的有效性。业务指标包含下述方面的量化改进：

- ✓ 更短的交付时间
- ✓ 提高客户满意度
- ✓ 在加大创新投资的同时降低维护开支
- ✓ 增加客户采用率

运营指标影响团队今后效率，衡量下述内容：

- ✓ 启动新项目的时间
- ✓ 构建时间
- ✓ 迭代测试时间

了解 DevOps 结果

DevOps 方法帮助 IBM SWG 提高了客户满意度和客户采用率，并实现了两位数收入增长。更短的时间段提高了 IBM 内部交付团队的活力，从而可以快速交付升级后的内部解决方案和新型云服务，比如 Bluemix、DevOp Services for Bluemix 和协作生命周期管理即管理服务 (CLM aaMS)。

作为 IBM 成功采用 DevOps 方法的具体示例，图 6-3 显示了 IBM SWG Rational 协作生命周期管理产品团队实现的量化成果。

生命周期衡量	2008 年	2010 年	2012 年 - 2014 年	总体改进
项目开始	30 天	10 天	2 天	28 天
整理待办事项	90 天	45 天	持续进行	89 天
总开发时间	120 天	55 天	3 天	117 天
总构建时间	36 小时	12 小时	5 小时	700%
BVT 可用性	不适用	18 小时	<1 小时	17 小时
迭代测试时间	5 天	2 天	14 小时	4 天
总部署时间	2 天	8 小时	4 小时-> 20 分钟	2 天
总生产时间	9 天	3 天	2 天	7 天
发布间隔时间	12 个月	12 个月	3 个月	9 个月

图 6-3：IBM SWG 团队量化的改进

第7章

十个 DevOps 误区

本章提要

- ▶ 了解 DevOps 适用的领域
- ▶ 了解 DevOps 不适用的领域

DevOps 运动目前正方兴未艾，特别是在企业中。和任何新兴运动及趋势一样，出现关于 DevOps 的误区和谬误在所难免。其中一些误区可能来自于尝试采用 DevOps 但以失败告终的公司或项目。不过，世上并不存在放之四海而皆准的解决方案。以下是关于 DevOps 的一些常见误区和事实。

DevOps 仅适合“网上诞生”的组织

通常所称的 DevOps 起源于“网上诞生”的公司（即在互联网上创办的公司），比如 Etsy、Netflix 和 Flickr。不过，几十年来大型企业一直使用与 DevOps 一致的原则和实践交付软件。此外，正如本书所述，现有 DevOps 原则具有一定的成熟度，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

运维团队总是通过编写脚本对环境和重复性任务进行管理，但随着基础架构即代码的发展，运维团队发现需要利用软件工程实践来管理大量的此类代码，该实践包括版本控制代码、签入、签出、分支和合并等。今天，通过创建用于定义环境的新版本代码，运维团队成员可以创建新版本环境。不过，这并不意味着运维团队需要学习如何使用 Java 或 C# 编码。大多数基础架构即代码技术使用 Ruby 之类语言，掌握这类语言相对容易些，特别是对于有脚本编写经验的人员而言。

DevOps 仅适合开发和运维

DevOps 的名字表明了它的开发和运维出身，不过，它却适合整个团队。包括业务部门主管、实践者、高管、合作伙伴和供应商在内的所有软件交付利益相关方都与 DevOps 存在关联。

DevOps 不适合 ITIL 组织

有人担心，持续交付等 DevOps 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 是一套适用于 IT 服务管理的最佳实践。事实上，ITIL 的生命周期模型兼容 DevOps。ITIL 定义的大多数原则与 DevOps 原则具有高度一致性。不过，ITIL 在一些组织中名声不佳，这是因为 ITIL 的部署方式主要是采用缓慢的瀑布式流程，而该流程不支持快速变更和改进。在开发和运维之间实现这些实践的统一正是 DevOps 的本质。

DevOps 不适合管制行业

管制行业的首要需求是制约与平衡，以及从负责确保合规性和可审计性的利益相关方获得批准。如果执行得当，采用 DevOps 实际上会改善合规性。此外，通过实现过程流自动化，以及使用具备审计跟踪捕获内置功能的工具，也可以提供帮助。



监管行业中的组织始终存在手动检查点或检查站，但这些元素并非不兼容 DevOps。

DevOps 不适合外包开发

在 DevOps 交付管道中，外包团队应被视为供应商或功能提供商。不过，组织应确保供应商团队的实践和流程与组织内部项目团队的兼容性。



通过使用通用发布规划、工作项管理和资产存储库工具，可以大幅改善业务部门与供应商和项目团队之间的沟通和协作，从而支持 DevOps 实践。通过使用应用发布管理工具，可以大幅改善组织对涵盖所有参与者的整个发布流程进行定义和协调的能力。

没有云意味着没有 DevOps

当您考虑 DevOps 时，常常会想到云，这是因为云可以为开发和测试人员动态设置基础架构资源，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。不过，云并非采用 DevOps 实践的必要条件，只要组织拥有高效的流程且能为部署和测试应用变更获取资源就可以了。



虚拟化本身是可选项。如果配置并部署服务器，使其具备必要速度，则持续交付到物理服务器就是可能的。

DevOps 不适合大型复杂系统

复杂系统需要 DevOps 所提供的纪律和协作。此类系统通常拥有多个软件和/或硬件组件，其中每个组件都有自己的交付周期和时间表。DevOps 促进这些交付周期与系统级发布规划之间的协作。

DevOps 仅涉及沟通

DevOps 社区的一些成员创造了一些幽默的术语，比如 ChatOps（团队通过 Internet 中继聊天之类的沟通工具执行所有沟通）和 HugOps（DevOps 仅专注于协作和沟通）。这些术语源自于一个错误观念，即沟通和协作解决一切问题。



DevOps 依赖沟通，但更好的沟通结合低效的流程并不会实现更好的部署。

DevOps 意味着持续变更部署

这种错误观念来自于仅部署 Web 应用的组织。一些公司在其网站上自豪地宣称，他们每日都部署到生产环境中。不过，每日部署并不适合部署了复杂应用的大型组织，还可能由于监管或公司限制而无法实现。DevOps 并非仅仅涉及部署，肯定也并非仅仅涉及持续部署到生产环境。采用 DevOps 让组织可以根据需要发布到生产环境中，而不是根据日历上的特定日期。

笔记

笔记

笔记

笔记



翻开此书并了解：

世界运行在软件上

当今世界瞬息万变，DevOps 成为企业获得精益性和敏捷性的基本方法。为快速应对不断变化的客户和市场需求，DevOps 帮助您持续交付软件驱动的创新。本书帮助您了解 DevOps 以及您的组织如何从中获取真正的业务优势。您还将全面了解涵盖整个软件交付生命周期（从新业务功能的构思和概念到生产部署）的 DevOps 如何在持续交付的世界中提供竞争优势。

- **超越客户期望 — 提供更出色的体验**
- **满足业务需求 — 应对大幅提升的软件交付频率和数量**
- **利用新技术趋势 — 使用移动、云、大数据和社交技术**
- **建立更好的协作 — 与整个软件交付生命周期中的利益相关方建立密切关系**

- **DevOps 功能**
- **如何利用移动、云和其他领先技术**
- **DevOps 如何与应用生命周期管理 (ALM) 流程保持一致**
- **如何管理多层应用**
- **IBM 应用 DevOps 的故事**



关注 IBM 云计算官方微信

IBM DevOps Hotline (China): 400-668-0529

WILEY

ISBN 978-1-119-17754-8

部件号: RAM14026-CNZH-00

非卖品

[点击此处，深入探讨 DevOps](#)