O'REILLY®

TURING 图灵程序设计丛书

基础设施即代码

云服务器管理

Infrastructure as Code: Managing Servers in the Cloud

实现IT基础设施管理向云时代转型，提升自动化程度、效率和可靠性

[美] 基夫·莫里斯 著

金明 钱伟 马博文 黄博文 禚娴静 译

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

基础设施即代码：云服务器管理

Infrastructure as Code: Managing Servers in the Cloud

[美] 基夫·莫里斯 著
金明 钱伟 马博文 黄博文 禚娴静 译

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®
O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北 京

图书在版编目（CIP）数据

基础设施即代码 : 云服务器管理 / (美) 基夫·莫里斯 (Kief Morris) 著 ; 金明等译. -- 北京 : 人民邮电出版社, 2018.9
（图灵程序设计丛书）
ISBN 978-7-115-49063-6

Ⅰ. ①基… Ⅱ. ①基… ②金… Ⅲ. ①软件开发 Ⅳ. ①TP311.52

中国版本图书馆CIP数据核字(2018)第179196号

内 容 提 要

本书旨在解释如何利用“云时代”基础设施即代码的方法来管理IT基础设施。主要内容包括：组织在采用新一代基础设施技术时经常掉进的陷阱以及避免这些陷阱的核心原则和基础设施即代码的关键实践；动态基础设施平台的性能和服务模型；提供、确认核心基础设施资源的工具；规定服务器、构建服务器模板和更新运行服务器的最佳实践和模型。

本书适合系统管理员、架构师、运维人员和开发人员阅读。

◆ 著 [美] 基夫·莫里斯
译 金 明 钱 伟 马博文 黄博文 禚娴静
责任编辑 杨 琳
责任印制 周昇亮

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
三河市君旺印务有限公司印刷

◆ 开本：800×1000 1/16
印张：16.75
字数：396千字 2018年9月第1版
印数：1-3 500册 2018年9月河北第1次印刷
著作权合同登记号 图字：01-2018-2895号

定价：89.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

译者介绍

金明

益辅金服CTO，ThoughtWorks前首席咨询师，ScaleWorks云创始人及首席架构师。拥有超过十年的互联网产品以及云计算的研发管理经验，为国内外多家银行、华为、中兴等大中型企业提供了技术变革的咨询服务，并多次在国内外软件大会上做主题演讲。译有《敏捷软件开发实践》《项目百态》等书。

钱伟

千米网内部敏捷教练，在通信行业有十年研发、售后、交付经验，两年IT咨询经验，深信“只要姿势对，敏捷治百病”。

马博文

ThoughtWorks前咨询师，AWS助理架构师、开发者。拥有多年Web开发和DevOps经验，熟悉持续交付、微服务。曾参与翻译《Scala编程实战》《DevOps实践》和《DevOps实践指南》，是西安DevOps Meetup活动的发起人。

黄博文

阿里巴巴技术专家，多年一线开发老兵，在持续集成、持续部署等DevOps领域拥有丰富的经验。曾在国内外多家企业从事过技术教练以及技术咨询工作，擅长敏捷工作方式。拥有AWS解决方案架构师以及开发者证书，译有《面向对象的思考过程》。

禚娴静

ThoughtWorks咨询师，拥有多年企业和互联网应用的一线开发经验，参与和主导过多个大型敏捷项目的技术交付、遗留系统重构和微服务架构转型。曾参与翻译《遗留系统重建实战》，享受跳跃的代码和专注带来的乐趣。

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

版权声明

O'Reilly Media, Inc.介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 *Make* 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过图书出版、在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——*Wired*

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野，并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

中文版推荐序

这本我非常期待的书终于要出版了，很高兴有机会为它写推荐序。我对基础设施的配置管理一直保持着浓厚的兴趣，而本书将传统的配置管理提升到了一个新的高度。

大家是否也和我一样有个疑问：运维人员如何才能像程序员一般维护和管理 IT 基础设施呢？本书就帮助我们回答了这个问题！

IT 基础设施大体经历了几个重要的发展阶段：从一开始裸金属架构的“钢铁时代”，到基于刚性架构的“虚拟化时代”，再到现在基于弹性架构的“云计算时代”。

我对于作者笔下的“手摇云”反模式有深刻的共鸣，同时也观察到大多数人只能用虚拟化技术池化大量硬件资源，却不能为内部客户提供自助服务，他们更执着于工单和 SLA 驱动的手工劳动。在 DevOps 大潮里，运维人员表现得非常积极，但是也很被动，加班和背锅的痛点与日俱增。我相信本书第 1 章描述的所有挑战可能就是大家正在经历的各种令人头疼的难题。

本书充满了作者的精彩见解，不仅全面地覆盖了与基础设施即代码（IaC）这一主题相关的所有基础知识，还合理地与 DevOps 实践体系融为一体。它应该是“配置管理经理”案头必备的一本参考书。作为一名早期的 CMDB 建设者、配置管理的超级粉丝，我在前一段时间详细研究了基础设施的持续集成，但是对于运维人员应该遵循什么测试方法、工作流程和组织结构并没有得出确切的答案。阅读本书之后，我认为找到了一个系统解决这一系列问题的途径。本书还提供了大量相关的实践、模式和反模式，非常值得深入学习。

想要成功地应用 IaC，也和应用 DevOps 一样，都需要先转变思维方式。运维人员再也不能故步自封于“生产环境的守门人”和“应用服务器的保姆”。相反，他们都应该变成基础设施的架构师和专家，在更高的层次定义 IT 架构和标准化的 IT 服务。需要构建让所有人都能够部署标准 IT 服务的工具，当其他人需要得到服务的时候，只需运行 IaC 代码即可，更重要的是还能保持服务持续在线和运行。

大部分 IaC 的相关工具软件都已经存在很长时间了，而 IaC 这个概念确实还比较新。可以确定的是，它为运维团队指出了一个提升的方向，为实施 DevOps 增加了一块重要的实践基石。

本书的作者是一名 DevOps 社区的活跃分子，曾多次参加 DevOpsDays 大会，书中也描述了一些他与其他大拿交流的相关经历。希望在本书出版之后，IaC 可以成为国内 DevOps 社区的一个热议话题，我本人也能对推广这项实践尽一份力。

中国 DevOpsDays 社区组织者
刘征
2018 年 7 月 20 日

译者序

把时间拨回 2008 年，敏捷软件方法在全球范围内日益流行。全球性敏捷大会 AgileConf 自 2003 年起已经举办了五届，Agile China 大会自 2006 年起也连续举办了两届。越来越多不同角色的技术人员参与到敏捷浪潮之中。他们拥抱敏捷思想和原则，重新审视和改造传统的软件流程。于是在这一年的 AgileConf 上，继开发、测试和需求之后，Andrew Clay Shafer 和 Patrick Debois 分享的“敏捷基础设施及运维”议题打破了硬件和软件的边界，让更多人关注起敏捷在基础设施领域的应用。星星之火逐步形成了旷日持久的 DevOps 燎原之势。

ThoughtWorks 一直是敏捷方法的信仰者和提倡者，不仅积极将内部积累的软件开发模式与实践对外分享，同时将社区总结的好模式与实践引入内部。例如，自动化构建、测试驱动开发、持续集成、自动化部署、用户故事、迭代式开发与发布等实践都是经过 ThoughtWorks 的同事们吸收与提炼，再进一步向社区倡导和推广的。Kief Morris 也是其中的一员，在基础设施和运维方面的丰富经验让他在公司内部的分享与讨论之中经常有令人印象深刻的观点。

2012 年，我第一次见到了 Kief。当时我们分别作为来自 ThoughtWorks 中国和 ThoughtWorks 英国的持续交付实践带头人，与 Jez Humble、Joanne Molesky、Sam Newman 以及其他国家和地区的持续交付实践带头人一起，在美国芝加哥召开了为期 5 天的“持续交付峰会”。在芝加哥千禧公园里云豆的掩映下，在美食与棒球之余，我们一起讨论了持续交付的定义、宣言和原则，梳理了持续交付的典型实践和工具箱。会议上，我们讨论得最多的就是各地的团队在开发、发布、监控、运维等过程中的最佳和最新 DevOps 实践。

2013 年，我开始了 ThoughtWorks 中国内部的 DevOps 云平台创业项目 ScaleWorks。ScaleWorks 是聚焦于云资源综合管理、服务器配置、应用发布、生产监控以及聊天即运维（ChatOps）等产品的整体解决方案。2014 年，我们向 Kief 介绍了 ScaleWorks，并和他在管理数千台 XenServer 及服务器的某个项目上进行了一些合作，取得了不错的效果。作为我们的产品顾问，Kief 在项目上不断思考、总结 DevOps 和持续交付的实践。他在博客上持续发表了一些精彩的文章，引起了很多共鸣。本书中的许多概念，例如“服务器蔓延”“配置漂移”等反模式，都是由他总结，再被社区接受、认同和传播开来。

从2015年起，Kief开始撰写本书，将他自己的博客、公司内外分享的文章以及更多的经验心得总结成文字，希望能够帮助大家掌握和理解“基础设施即代码”。得到这个消息之后，我第一时间联系了Kief和图灵，希望能够将本书引入国内。2016年，图灵获得了本书的中文版权。经过一段时间的翻译和编辑，本书的中文版终于要和大家见面了。

回顾整个历程，不由得感慨万千。革命性技术实践的发起、传播和成熟，往往需要无数人投入常年的心血和热情。正是这些人不辞辛劳、不计得失地付出和布道，才让整个技术社区实现了颠覆式的突破。时至今日，软件从业人员坐拥公有云、微服务、容器等技术以及新型IDE和自动化工具，可以专注于业务需求的开发和实现，无须考虑底层服务器的细节与软件的构建部署流程。他们何尝能够想象“铁器时代”的基础设施和开发效率呢？光鲜的背后，是先行者们将脏活累活代码化、工具化乃至服务化的不懈努力。

顾名思义，“基础设施即代码”是将基础设施及其完整生命周期的操作通过代码的方式进行描述和执行。这个概念来自敏捷思想中出现已久的“测试即代码”（测试自动化）、“构建即代码”（构建自动化），甚至软件程序中更久远的“配置即代码”（例如，Java Annotation）。相比这些“××即代码”，“基础设施即代码”则另辟蹊径，将代码化的风潮扩展到了服务器、网络、系统配置、发布、监控等环节。不过从本质上看，各类“××即代码”的核心价值观和出发点是一以贯之的。

不妨以更广阔的视角来看，“××即代码”运动也和通用程序语言（GPL）拥抱函数式编程以及领域特定语言（DSL）的兴起密不可分。无论是软件构建工具Ant、Maven、Gradle等，还是系统配置工具Puppet、Chef、Ansible等，抑或是云原生的容器工具Docker、Kubernates、Prometheus等，都选择了将各自管理领域的对象和行为进行抽象和建模。它们基于XML、YAML等文本格式或Groovy、Ruby等脚本语言，定义了特定的DSL来描述这些领域的对象和行为，最终深刻地改变了这些领域的传统工作方式，让一切领域的对象和行为都可以通过代码进行定义和执行。

再联想一下，最近几年在以太坊、区块链社区流行的智能合约和被社区奉为圭臬的“Code is Law”更是百尺竿头，再进一步。人类各种活动的规则和状态（商业、公共事务，甚至投票、选举等）都可以通过代码来描述，通过计算机来执行、管理和调度。随着计算、存储和网络的云端集中，物联网的边缘计算以及人工智能的机器学习，在不远的未来，人类所有的活动和数据是否都可以数字化成代码？一场更深刻、更广泛的“××即代码”运动正在徐徐拉开大幕，不论会令人类欣喜或忧虑，都是必然的趋势。

回到本书，其体系化结构非常清晰，整体上分为三部分：基础概念、模式与实践。本书的内容涵盖了从动态基础设施平台（虚拟化、云、容器等）到通用基础设施服务（监控、服务发现、分布式进程管理、软件部署等）的各种工具和实践。内容虽多，却繁而不乱。本书考虑到了读者的阅读习惯，贴心地按照模式、实践以及组织变革的次序，由浅入深、由理论到落地地对基础设施即代码进行了颇为深刻的分析。

“基础设施即代码”的具体模式和实践是本书的重点。本书围绕广义的基础设施，分为“基础设施－服务器－服务”三层架构，着重从服务器的创建、配置、打包、远程执行、中央注册等不同的环节来对工具和方法进行细致入微的介绍。在模式和实践的实施方面，本书也结合了敏捷开发方法，从构建、测试、实施和组织变革这几个维度进行了深入浅出的讲解。层层叠叠却脉络清晰，令人豁然开朗。

无论你是开发工程师、运维工程师、架构师抑或管理层，阅读本书都将大有裨益。本书的每个部分不仅告诉你“是什么”，而且从目标出发，分析其中的需求和原则，再对比不同的工具和选项，从而让读者能够深刻理解“为什么”和“怎么做”。如果你想完善自己的知识体系，本书的知识网络是很好的基础；如果你想应用合适的工具，本书的分析脉络是很好的助手；如果你想推广模式、让工具的落地，本书的实操方法论可以让你事半功倍。

诚然，“一千个读者心中有一千个哈姆雷特”，背景、经历和工作的不同让每个人都有自己的视角和关注点。书越读越厚，又越读越薄，留下的是自己的思考和心得。祝福每位读者在阅读本书的过程中，都能找到属于自己的乐趣。

本书的翻译是由 ThoughtWorks 中国的几位同事集体完成的。钱伟、马博文、黄博文和禚娴静都是非常优秀的技术人员，帮助很多客户实施了敏捷和 DevOps 方法，尤为可贵的是他们热心于社区的分享和知识的传递。与他们合作翻译本书是一次非常有趣和难忘的经历。由于译者的时间、经验有限，难免出现纰漏，我们诚挚欢迎所有读者的反馈和批评，让我们一起把“基础设施即代码”的思想、原则和实践更准确地传播给更多的人。

最后，要感谢我的妻子徐婷婷，以及我们亲爱的宝贝金宥文。正是他们的支持，让我可以专心地翻译，对本书的整体译稿进行统稿和校稿，反复锤炼词句，呈现给读者。我还要感谢图灵的朱巍、杨琳等编辑老师，没有她们认真的审校和排版，这本书不会如此完美地展现在读者前面。

金明

2018 年 7 月 31 日

前言

在构建和管理基础设施的过程中，基础设施团队与软件开发团队正在越来越多地使用名为“基础设施即代码”的自动化工具。这些工具要求用户在类似软件源代码的文件中定义服务器、网络以及基础设施的其他元素。它们会编译并解读这些文件，以决定采取哪些行动。

这类工具随着 DevOps 运动[1]蓬勃发展。DevOps 运动主要是关于软件开发人员与软件运维人员之间的文化和协作。基于软件开发范式的基础设施管理工具促进了开发与运维的融合。

基础设施即代码的管理方式与传统的基础设施管理方式迥然不同。我遇到过不少团队努力地去实现这一转变，但是有效应用这些工具的理念、模式与实践却只在各种大会演讲、博客以及文章里提及过。我一直在等待一本书，能把这些想法整合到一起，但迟迟没有看到有人写作的迹象，于是我最终决定亲自动手。你正在阅读的这本书就是我努力的成果。

我是怎样学着不再担忧并且爱上云的

1992 年，我搭建了自己的第一台服务器——一个拨号的 BBS[2]，由此成为 Unix 系统管理员，随后负责为不同规模的公司构建与运维托管式软件系统［后来我们称之为 SaaS（software as a service，软件即服务）］，其中既包括创业公司也包括大型企业。在我听说“基础设施即代码”这个术语之前，就始终在朝着基础设施即代码的方向前进。

虚拟化让一切变得难以为继。对很多人而言，我磕磕绊绊地实施虚拟化和云计算的经历可能并不陌生，它揭示了基础设施即代码要在现代 IT 运维中扮演的角色。

我的第一组虚拟机集群

2007 年，当我的团队获得预算，可以购买两台强劲的 HP 机架服务器以及 VMware ESX 服务器的软件许可证时，我兴奋极了。

我们当时的办公室机架上有大概 20 台 1U 和 2U 服务器，分别以水果（Linux 服务器）和

注 1：Andrew Clay Shafer 与 Patrick Debois 在 Agile 2008 大会上的演讲引发了 DevOps 运动。Debois 组织的一系列 DevOpsDays 大会极大推动了 DevOps 运动。

注 2：BBS 是电子公告板系统（bulletin board system）的简称。

莓果（Windows 数据库服务器）命名，上面运行着开发团队使用的测试环境。我们每天的工作就是大量使用这些服务器，测试不同的版本、分支，以及高优先级、验证概念型的应用程序。这些服务器上塞满了网络服务，如 DNS、文件服务器和电子邮件，并且运行了多个应用程序实例、Web 服务器与数据库服务器。

因此，我们非常确信这些新的虚拟服务器将改变我们的生活。我们可以清晰地将以上服务分离到它们各自的虚拟机（virtual machine，VM）上面，而 ESX Hypervisor 虚拟化管理软件会帮助我们从多核服务器和内存上榨取尽可能多的性能。我们可以很容易地复制服务器来创建新的环境，并且将不再需要的服务器归档到磁盘上面，而不用担心在未来需要的时候无法恢复它们。

这些服务器真真切切地改变了我们的生活。不过，虽然解决了很多老问题，我们却发现了新的问题，不得不学习采用完全不同的方式来思考基础设施。

虚拟化使服务器的创建与管理变得更容易了。它带来的负面影响是，我们最终创建的服务器数量远远超出了想象。产品和市场部门的同事高兴极了，因为我们一天之内就能提供一个全新的演示环境，而不用他们提供预算并等待几周时间以便我们完成硬件服务器的采购和设置。

魔法师的学徒

一年之后，我们运行了超过 100 台虚拟机，而且这个数字还在增长。我们当时正在将生产服务器转向虚拟化，并且在尝试亚马逊新推出的云计算托管服务。虚拟化给业务人员带来的好处意味着，我们有钱购买更多的 ESX 服务器和强大的 SAN（存储网络）设备，以满足基础设施对于存储的惊人“胃口”。

但是，我们发现自己有点像动画电影《幻想曲》里面“魔法师的学徒”情节中的米老鼠。虚拟机的数量不断增加，最后多到了我们无法承受的地步。当出现问题的时候，我们追踪到具体的虚拟机，然后修复故障，但是从不跟踪记录在哪些地方做了哪些修改。

> 噢，完美的一击！
> 看我把它砍成两半！
> 现在我又重拾希望了，
> 可以无忧无虑地呼吸了！
>
> 唉，我高兴得太早了！
> 两半都变成了完整的一个，
> 现在，有两个仆人
> 听从我的吩咐！
> 伟大的神啊，救救我吧！
> 我恳求你！
>
> ——摘录自歌德的十四行诗《魔法师的学徒》[3]

注 3：1797 年，德国诗人歌德创作了十四行叙事诗《魔法师的学徒》，讲述了一位学徒魔法师在老魔法师离开后，使用半吊子魔法指挥拖把、水桶等工具进行清扫工作，最后因为不能掌控局面而弄得一片混乱，几乎将房间淹没。老魔法师回来后才将一切复原。作者此处借用了这个典故，比喻不加节制地使用虚拟机反而导致基础设施团队陷入困境。——译者注

每当操作系统、Web 服务器、应用服务器、数据库服务器、Java 虚拟机和其他各类软件推出更新版本，我们就不得不忙于在所有系统上安装更新。有些服务器可以成功安装更新，但有些服务器上的更新则会导致一些东西无法工作，而我们没有时间来解决每一个不兼容性的问题。随着时间流逝，几百台服务器上最终混杂了很多不同的版本。

在引入虚拟化之前我们就在使用配置自动化软件了，这些软件本该帮助我们解决这些问题。我在之前的公司里使用过 CFEngine，刚来这个团队时，也尝试过一款名为 Puppet 的新工具。之后，在研究 AWS 基础设施的时候，我的同事 Andrew 又引入了 Chef。所有这些工具都很有用，然而却并没有把我们从服务器差异巨大的困境之中解救出来，尤其是在最初的日子里。

问题在于，虽然 Puppet（以及 Chef 等其他工具）应该在所有的服务器上都设置好并且无人值守地运行，但我们却不放心。服务器的差异实在太大了。我们可能针对某一台特定应用服务器编写了一些配置清单文件（manifest）来进行配置和管理；但是当在另一台理论上相似的应用服务器上运行这些配置清单文件时，我们发现不同版本的 Java、应用程序和 OS 组件会导致 Puppet 运行失败，甚至更糟——导致应用服务器无法工作。

我们最终只是临时使用 Puppet。我们能在新的虚拟机上放心地运行 Puppet，不过每次运行之后可能还要再做一些小调整。我们会针对某个特殊的任务编写一些配置清单文件，然后逐一在每台服务器上运行，小心翼翼地检查结果，并按照需要做一些调整。

因此配置自动化是一个有用的辅助手段，比批处理脚本更好，但是我们的用法并没有避免服务器不一致的蔓延。

从零开始的云

当我们开始往云上迁移时，一切都改变了。使情况好转的并不是这项技术本身，因为我们用自己的 VMware 服务器也能做到同样的事情。但由于是从零开始，我们采纳了新的服务器管理方法——基于从虚拟服务器集群中学到的经验，以及从 Flickr、Etsy 与 Netflix 等公司的 IT 运维团队那里读到、听到的经验。在将服务器迁移到云上的时候，我们就把这些新的想法融入到了服务的管理方式里面。

这种新方法的关键在于，每台服务器都能够自动化地从零开始重新创建，并且配置工具会持续而非临时地运行。添加到新基础设施的每台服务器都要遵循这种方法。如果自动化在某些边界情况下失败了，我们要么修改自动化脚本，处理这个问题，要么修改该服务的设计，使其不再是边界情况。

这个新方法并非一帆风顺。我们不得不培养新习惯，不得不寻找各种办法来克服高度自动化的基础设施所带来的挑战。当团队成员进入其他组织、参与到诸如 DevOpsDays 的社区里，我们都会不断学习和提高。随着时间的推移，我们终于开始习惯数百台服务器规模的自动化基础设施。与曾经的“魔法师的学徒”岁月相比，我们只需要投入很少的工作量，遇到的烦心事也更少了。

加入 ThoughtWorks 为我打开了一扇新的大门。与我一起工作的开发团队非常热衷于使用来自于极限编程的工程实践，例如测试驱动开发（test-driven development，TDD）、持续集

成（continuous integration，CI）和持续交付（continuous delivery，CD）[4]。由于我已经使用过源代码控制系统来管理基础设施的脚本和配置文件，将这些严格的开发和测试方法应用于基础设施领域也是水到渠成。

在 ThoughtWorks 的工作让我可以与很多 IT 运维团队交流，他们中的很多人都在使用虚拟化、云计算和自动化工具来解决大量问题。在工作中与他们一起分享和学习新的想法与技巧，是一段非常棒的经历。

为什么写作本书

我遇见过很多团队处于我几年前的状态：虽然使用了云计算、虚拟化和自动化工具，但却没有竭尽所能发挥其功用。

时间是最大的问题。系统管理员的日常就是疲于应付无休止的紧急任务——灭火、解决问题以及启动新的业务关键项目，没有给简化日常工作的基础性改善任务留下太多的时间。

我希望本书能够从实用的视角，针对如何管理 IT 基础设施介绍一些团队可以尝试和使用的技巧与模式。我将避免陷于配置和使用具体工具的细节之中，而是使内容适用于各种不同的工具，甚至那些尚未面世的工具。同时，我会以现有工具作为例子来展示观点。

基础设施即代码的方法对于管理任何实际规模或复杂性的云基础设施都是非常关键的，但是并不局限于使用公有云的组织。本书介绍的技巧和实践已经被证实适用于虚拟化的环境，甚至未虚拟化的裸机服务器。

基础设施即代码是 DevOps 的基石之一。它就是 CAMS 中的“A”。[CAMS 是指 culture（文化）、automation（自动化）、measurement（度量）和 sharing（分享），详见 IT Revolution 网站上的文章“DevOps Culture (Part 1)”。]

本书适合哪些读者

本书适合于维护 IT 基础设施的人，尤其是管理服务器和服务器集群的人，包括系统管理员、基础设施工程师、团队领导、架构师或者对技术感兴趣的经理。本书也适合有意构建和使用基础设施的软件开发人员。

本书默认读者对于虚拟化或 IaaS（infrastructure as a service，基础设施即服务）有一些基础认识，知道如何创建虚拟机和配置操作系统，至少体验过 Ansible、Chef 或 Puppet 等配置自动化软件。

除了作为基础设施即代码的入门指南，本书对已经采取这种工作方式的读者也有所裨益，让大家借此分享想法、交流意见，以便做得更好。

本书涵盖哪些工具

本书不会介绍具体脚本语言或工具的用法。书中会提供某些特定工具的代码示例，但都仅

注 4：可参阅曾获得第 21 届 Jolt 大奖的《持续交付》一书，其中文版已由人民邮电出版社出版，图书主页为 ituring.cn/book/758。——编者注

仅是为了说明概念和方法，而非教程。无论你是在 OpenStack 上使用 Chef，在 AWS 上使用 Puppet，还是在裸机上使用 Ansible，抑或是完全迥异的技术栈，本书都会有所帮助。

书中提及的特定工具都是我所知晓的、在本领域有一定影响力的。不过这个领域在持续变化，存在着大量其他相关工具。

本书示例中使用的工具一般都是我所熟悉的，能够编写例子来演示我的观点。例如，我使用 Terraform 来编写基础设施定义文件的例子，因为它是一款非常不错、语法干净的工具，而且我在很多项目上用过。本书的很多示例使用了亚马逊的 AWS 云平台，因为读者对它可能最为熟悉。

如何阅读本书

阅读第 1 章，至少快速浏览一遍，以理解本书中的术语以及提倡的原则。你可以基于它们来决定应该专注于本书的哪个部分。

如果你对这些自动化、云和基础设施编排工具比较陌生，需要先了解第一部分，再继续第二部分。对前两部分都有所得之后，再阅读第三部分。

如果你使用过本书介绍的自动化工具，但在阅读第 1 章之后，觉得之前并没有遵照这些工具的设计意图来使用，那么可以跳过或者快速浏览第一部分，集中精力在第二部分。第二部分描述了使用动态和自动化基础设施的方法，并且这些方法与第 1 章概括的原则一致。

如果你对第 1 章描述的动态基础设施与自动化方法已经很熟悉了，可以快速浏览第一部分和第二部分，集中精力在第三部分。第三部分更深入地探讨了基础设施管理领域：架构方式与团队工作流。

排版约定

本书使用以下排版约定。

- **黑体**
  表示新术语和重点强调的内容。
- 等宽字体（`constant width`）
  表示代码程序，以及正文中出现的变量名、函数名、数据库、数据类型、环境参数、语句和关键词等。
- 等宽粗体（**`constant width bold`**）
  表示由用户输入的命令或者其他文本。
- 等宽斜体（*`constant width italic`*）
  表示需要由用户输入的值或根据上下文确定的值进行替换的文本。

该图标表示提示或建议。