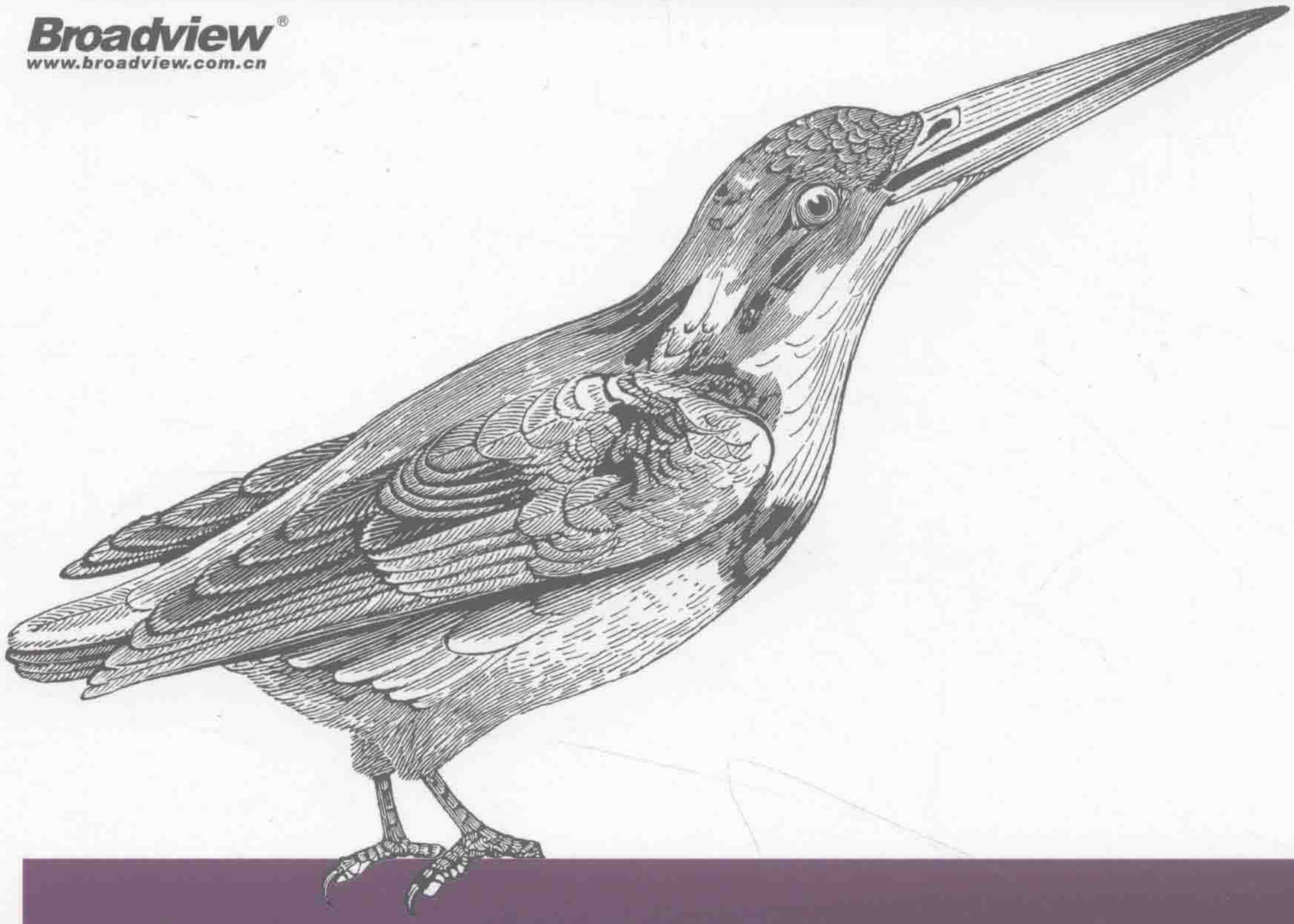
O'REILLY®

Broadview®
www.broadview.com.cn



云原生Java

Spring Boot、Spring Cloud与Cloud Foundry弹性系统设计

Cloud Native Java

[美] Josh Long　Kenny Bastani 著
张若飞　宋净超 译

中国工信出版集团
電子工業出版社·
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

O'REILLY®

云原生Java

Spring Boot、Spring Cloud与Cloud Foundry弹性系统设计

Cloud Native Java

[美] Josh Long　Kenny Bastani 著
张若飞　宋净超　译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

无论是传统 IT 行业，还是互联网行业，都正处于行业历史上最剧烈的变革中：大量的系统正在从传统的 IT 架构转向基于云的架构，开发模式也正在从开发和运维分工的传统模式，逐渐转向统一的 DevOps 模式。Java 技术也应运进入了新的生命周期，大量被用于构建现代的、基于云的应用程序。

本书深入研究了云计算、测试驱动开发、微服务与持续集成和持续交付领域的工具和方法，并指导你将传统应用程序转变为真正的云原生应用程序。其中重点介绍了微服务框架 Spring Boot，以及如何使用 Spring Boot 轻松创建任何粒度的 Spring 服务，并部署到现代的容器环境中。可以说本书是一本讲述如何使用 Spring Boot、Spring Cloud 和 Cloud Foundry 构建软件的理论和实践的完备指南。

本书主要面向正在使用 Spring Boot、Spring Cloud 和 Cloud Foundry 构建软件的 Java/JVM 开发人员。

版权贸易合同登记号　图字：01-2017-8746

图书在版编目（CIP）数据

云原生Java：Spring Boot、Spring Cloud与Cloud Foundry弹性系统设计 /（美）乔西·朗（Josh Long），（美）肯尼·巴斯塔尼（Kenny Bastani）著；张若飞，宋净超译. —北京：电子工业出版社，2018.6
书名原文：Cloud Native Java
ISBN 978-7-121-34251-6

Ⅰ. ①云… Ⅱ. ①乔… ②肯… ③张… ④宋… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字（2018）第106114号

策划编辑：张春雨
责任编辑：牛　勇
封面设计：Karen Montgomery　张　健
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：787 × 980　1/16　　印张：36.25　　字数：794千字
版　　次：2018年6月第1版
印　　次：2018年6月第1次印刷
定　　价：128.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819　faq@phei.com.cn。

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。无论是通过书籍出版、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

译者序

“Java 已死。”

这几年，我们不断听到这句话，甚至相信过这句话，认为 Java 语言已经存在了如此长的时间，影响了如此广的范围，是时候该有一些新鲜的东西来颠覆它了。但是令人失望的是，Java 语言不仅自身依然在快速地进化（本书完成时，Java 10 已经正式发布），其整个生态圈也都不曾停下前进的脚步。Spring 就是一个很好的例子。相信 Java 开发人员对它已经再熟悉不过了，大部分人每天的工作都是在围绕着它进行开发。令人惊讶的是，Spring 也已经从以前单一的核心框架，逐渐发展成为一个全栈的应用层框架，甚至在大家还没有觉察的时候，它已经开始用于构建微服务的生态系统了，而 Spring Boot 就是整个微服务生态的核心。

在 Spring Boot 出现之前，我们看到 Spring 也在努力尝试，将整个繁杂的 Spring/Spring MVC 体系精简成一个容易上手的工具，也出现了像 Spring Roo 这样的过渡产品。但是直到 Spring Boot 横空出世，我们才发现，原来开发一个应用程序，可以如此简单。这使我不由想起 10 年前，我翻译了国内第一本 Grails 书籍和 Rails 等书籍，当时那些令我陶醉的 feature，这一次终于踏踏实实地在 Spring Boot 中得到了实现。与此同时，Spring 团队秉承了其一贯的风格，简单却强大。如今，Spring Boot 不再是一个简单的框架，而是像一个灵活的骨架结构，你可以很轻松地将 Spring Data、Spring Security 这些耳熟能详的模块，像插积木一样插在 Spring Boot 上，甚至不需要写一行代码。它就像一幅绝世的中国山水画，让你既惊叹于其精致的笔法，又被它整体磅礴的气势震慑得不能一语。而这一切，都与 Java 这十年间不断的提升和整个 Spring 生态的发展密不可分。

如今的 IT 行业，已经不再是十年前的情景，云计算已经发展成熟并且逐渐占据主流。没有人愿意再去花高额的成本、时间、人手建立自己的数据中心，基于云计算的产品和项目甚至一天就可以开发完成上线。更快，永远是技术追逐的目标。所有你能想到的框架、组件、中间件，甚至是业务功能模块，都在快速地被云端化，你随便打开一个云计算产品的官网，上面都有不下上百种的产品功能供你选择，即便是最新的机器学习平台，你

也只需点几下鼠标就可以搭建起来。更简单，也永远是技术发展的趋势。在 Netflix 的成功背景下，Spring 没有停留在 Spring Boot 上止步不前，而是进一步迈向了云原生的应用开发，因此而诞生了 Spring Cloud。从严格意义上说，Spring Cloud 才是一个真正的微服务框架，它提供了一套完善的解决方案，能够将云环境中各个独立部署的微服务整合起来，因此云原生的应用，本质上就是一个分布式的应用。同时，它打破了我们之前依靠硬件或其他一些软件来实现分布式系统的做法，其将一切功能都云端化，包括中心化配置、负载均衡、路由、集群，以及衍生出来的断路器等常见设计模式。我们恍然发现，原来开发一个云上的应用也变得如此简单，但其功能依然强大。可以预见的是，掌握 Spring Boot、Spring Cloud 的开发技术，以及核心思想，一定会是下一代 Java 开发人员必备的技能。幸运的是，我们已经在当前的工作中实际运用了它们，并取得了非常不错的效果。

感谢 Josh 和 Kenny 分享了他们各自在这个领域的深刻理解和经验，编纂成此书，同时我也为能够参与翻译此书感到荣幸。相信这本书，可以为 Java 开发人员打开下一个时代的大门，让大家在原生云环境的应用中徜徉。感谢张春雨编辑的耐心和督促，不知不觉合作已十年有余。感谢净超（Jimmy）临危受命，承担了本书后半部分的翻译工作，他的高产和高效令我刮目相看，也时刻能感受到他对技术饱含的热情。感谢我身边的所有朋友，一直在支持着我，鼓励着我，这也是当我面对压力和挫折时不放弃的理由。最后，感谢我的妻子，给了我莫大的支持和鼓励，让我可以投身创业之中，愿她一生幸福满足！感恩我的母亲，让我懂得了生命如此脆弱又如此坚强，生死之外，再无大事，愿她一生健康平安！

译者　张若飞

2018 年 5 月 23 日凌晨 于北京

致 Eva 和 Makani，爱你们的 Josh 叔叔。

致我的祖父 Abbas，他等了一百年，终于等到在一本书的封面上看到自己的名字。

——*Kenny*

序言（James Watters）

你不能两次踏入同一条河流。

——赫拉克利特（希腊哲学家）

2015 年夏天，在威尼斯海滩的一家咖啡店里，我坐在 Josh Long 旁边，我知道我们要开始干一件大事了。因为开发者不断要求了解我们的新技术——Spring Boot，所以他的行程几乎都被排满了。我们的云原生平台 Pivotal Cloud Foundry，正在迅速成为流行的云原生应用程序运行时环境。随着 Spring Boot 越来越受到欢迎，Spring Cloud 的到来有望成为一个爆炸点。“这将是一件大事，而且正在发生。”我跟他说。

工作中的力量是巨大的。在 CIO 全力寻求如何提升开发者生产力的时候，Spring Boot 提供了一个微服务和 DevOps 友好的企业级开发方式。 随着 Spring Boot 和 PCF 之间的集成日益完善，生产环境部署成为一个简单的流水线和 API 调用工作。Spring Cloud 不仅提供了世界上第一个微服务网格，并且诞生了一种新的、基于云的 Java 标准。

这些技术的独特组合，不仅仅是表面看上去开发风格的变化，而是改变了大型组织的交付结构。由于对传统 Java 应用程序来说，服务器和运维的复杂性太高，因此开发人员被禁止与生产环境接触。我们经常听到，客户部署一次程序要花费几天这样恐怖的故事。我们知道我们的平台会改变他们的生活。有一些粉丝客户开始给我们写邮件，描述他们在 PCF 上采用 Spring Boot 之后，如何将生产环境的更新从几个月的时间，降低到几分钟之内。

自 2015 年以来，现实已经证明，所有迁移到这种开发方式的企业，开发速度至少提高了 50%，超过了 MTTR（平均维护时间）/ 停机时间的一半，并且他们与许多小型的平台团队一起运行了成千上万的 JVM 实例。 最重要的是，采用云原生 Java 的企业，可以拿出更多的时间来思考他们的客户和市场，同时对开发和运维复杂性的担忧大大减少。

本书对现代企业软件设计中重要的模式，逐一进行了详细的论述。 在许多例子中，你可

以看到 Josh 和 Kenny 与许多世界顶级企业一同努力所带来的实践经验。

我建议每个开发者和 IT 主管，做好充分的准备，挖掘企业的适应性和柔韧性，来享受这份工作。

James Watters，*Pivotal Cloud Foundry* 高级副总裁，
Pivotal，*@wattersjames*

序言（Rod Johnson）

我们正处于行业历史上最剧烈的变革中：从传统架构转向云的架构，从传统开发和运维分工转向统一的 DevOps。本书可帮助你进行转型，在本书中详细阐述了开发云原生应用程序的机遇和挑战，并明确指出了成功实现的方向。

转型不会一夜之间发生。这本书最好的一点，是它强调了如何将以现有经验构建的环境，逐步迁移到云上。特别是第 5 章中“用 Spring 实现服务平等”一节，提供了一个传统企业转向云端的优秀实战案例。

本书是理论和实践的完美结合，既解释了现代应用的架构原理，又给出了有效的、经过验证的实现方法。实践需要我们做出选择，不只是编程语言的选择，更主要是选择开源框架，因为现代应用程序几乎都是建立在通用问题的开源解决方案之上的。如果你选择了 Java 语言，或是对编程语言持开放态度——本书就是你需要的书。

> 我的死亡报告是夸大的。
>
> ——马克·吐温

几年前，报道 Java 死亡的消息不绝于耳。今天，Java 仍在蓬勃发展，本书会告诉你为什么。Java 已经进入了新的生命周期，部分原因是因为 Java 技术已经用于构建现代的、基于云的应用程序了。使 Netflix 成功的两个关键因素，就是开源项目和 Spring。本书在这两个方面都做得非常出色。

Spring 最初的核心思想，是降低过去 Java EE 的复杂性，它经受住了时间的考验，成为了云应用的完美基础。十多年前，我们曾经谈论过“Spring 三原则”：依赖注入、可移植服务抽象和 AOP。如今，将业务逻辑与环境完美分离比以往任何时候都重要，所有这一切都没有发生变化。

本书的重点是介绍 Spring Boot，它是在微服务时代一种使用 Spring 的新方式，我们无法拒绝使用它。 Spring Boot 可以轻松创建任何粒度的 Spring 服务，并部署到现代的容

器环境中。传统的“企业级”Java 应用程序，都是只能运行在更大服务器上的单体程序，而 Spring Boot 以它的简单和效率改变了这一切: 服务聚焦精准，以足够的服务器来运行。

书中的案例展示了 Spring 团队的最新工作，例如，Spring Cloud Stream 和改进的集成测试支持，以及与 Netflix 开源项目不断的集成。

我很高兴看到 Spring 的持续创新，以及专注于简单化开发人员的工作。虽然在过去的 5 年中，我只是以一个用户的身份与 Spring 进行了互动，但是看到它繁荣发展，成功解决了许多复杂的问题，还是很高兴的。今天，我仍然在 Atomist 从事这类工作，我们的目标是让所有事情自动化，让 Spring 来为所有的开发团队和开发过程做 Java 应用程序的事情。Spring 为 Java 开发人员所关心的每件事情，都提供了一个简单的、有效的结构和抽象，同时，Atomist 也希望为项目源代码、系统构建、问题跟踪器、环境部署等提供同样的东西。无论是为了提高团队协作能力而使用 Slack 管理项目，还是为了监控部署事件，Atomist 都可以提供强大的开发自动化能力。

自动化的基本组成部分是测试。我特别喜欢本书对测试的介绍，书中对如何解决微服务测试中的许多难题提出了宝贵的意见。我也喜欢书中许多注释详尽的代码清单，作者很好地坚持了这个 O'Reilly 一贯的风格。

我非常荣幸，可以为同道好友作序。Josh 是一个善于沟通的人。他的文字功底和他现场编写的代码一样好。Josh 和 Kenny 是两位充满激情的、好奇并且见多识广的导游，很高兴与他们一起走过这个旅程。我在旅途中学到了很多东西，我相信你也会学到很多。

Rod Johnson，Spring Framework、Atomist 创始人兼 CEO，
twitter 账号 @springrod

前言

更快！更快！更快！！每个人都想走得更快，但很少人知道如何做到。市场的需求在不断增长，机会也越来越多，但我们中的一些人根本无法跟上节奏！传统企业与亚马逊，以及 Netflix 和 Etsy 之间的区别是什么？我们知道这些公司已经成长到拥有令人疯狂的规模，但不知什么原因，它们仍然能够保持竞争优势，保持领先地位。它们究竟是如何做到的？

一个想法从概念到实现，需要做大量的工作。要了解一个想法，既要看它的效用，也要看它的价值。这些工作要经历很多不同的环节，从涉及用户体验的产品管理到测试，再到运行，最终才进入生产环境。从历史的角度看，这其中的每一道工序都会让整个工作变慢。我们作为一个开源社区，随着时间的推移，已经优化了这个流程中的一些部分。我们有了云计算，所以不再需要机架和机柜。我们使用测试驱动开发和持续集成来实现自动化测试。我们以小步迭代的方式来发布软件，通过微服务来减小代码变更的范围和成本。我们拥抱 DevOps 背后的理念（武装的同理心）来增强对整个系统的了解，增进开发人员与运维人员之间的感情，减少不同优先级事项之间的巨大协同成本。这些事情本身并不有趣，改进也不大，但是如果将它们结合起来，我们就可以将整个价值链中的每件重要事情都隔离开来。总而言之，这些东西就是我们所说的云原生。

作为行业中的一员和理论的实践者，软件开发人员今天处于一个激情的时代。我们在基础设施、测试、中间件、持续集成和交付、开发框架和云平台等方面，都拥有可靠的、开源的、稳定和自助服务的解决方案。这些基础条件让企业可以专注于如何低成本地提供高商业价值，以及扩张到更大的规模。

谁应该阅读本书

本书主要面向正在使用 Spring Boot、Spring Cloud 和 Cloud Foundry，以更快、更好地构建软件的 Java / JVM 开发人员。相信你已经听说过微服务的概念，也许你已经看到了

Spring Boot的不断发展，并且想知道为什么今天大多数企业都在使用Cloud Foundry。那么本书可以告诉你答案。

为什么我们写这本书

在Pivotal，我们通过传授持续交付的知识，以及通过Cloud Foundry、Spring Boot和Spring Cloud来帮助客户向数字化企业转型。我们已经知道了什么是可行的（以及什么不可行），并且希望把客户的实践和我们的经验总结出来。我们并不想面面俱到，但我们试图清晰地向你介绍整个云原生世界的关键概念。

浏览本书

本书的组织结构如下：

- 第1章和第2章介绍了云原生思想产生的背景，然后介绍了Spring Boot和Cloud Foundry
- 第3章介绍了如何配置Spring Boot应用程序。这是我们所要依赖的基本技能。
- 第4章介绍了如何测试Spring应用程序，从如何测试最简单的组件到测试分布式系统。
- 第5章介绍了可以将应用程序迁移到Cloud Foundry等云平台的轻量级重构方式，你可以从中获得一些有价值的额外经验。
- 第6章介绍了如何使用Spring构建HTTP和RESTful服务。你会在API和领域驱动开发中用到其中很多技巧。
- 第7章介绍了在分布式系统中控制请求进出的常用方法。
- 第8章介绍了如何构建一个响应外部请求的服务。
- 第9章介绍了如何使用Spring Data在Spring中管理数据。这为领域驱动的思想奠定了基础。
- 第10章介绍了如何使用Spring中事件驱动、消息中心化的能力，来集成分布式服务和数据。
- 第11章介绍了如何利用云平台(如Cloud Foundry)的能力来处理长期运行的工作。
- 第12章介绍了在分布式系统中管理状态的一些方法。
- 第13章介绍了如何构建具备可观测性和可操作性的系统。
- 第14章介绍了如何构建类似于Cloud Foundry平台的服务代理。服务代理可以将有状态的服务（例如消息队列、数据库和缓存）连接到云平台。
- 第15章介绍了持续交付背后的思想。这虽然是最后一章，但也可能是你旅程的开始。

如果你像我们一样，你不会从前到后来阅读本书。如果你真的像我们一样，你通常不会阅读前言。但是，既然你看到了这里，我们给出以下一些建议：

- 无论你是做什么的，请阅读第 1 章和第 2 章。这两章为本书的其余部分奠定了基础。如果没有第 1 章中所介绍的动机和业务背景，本书中所有的技术讨论就都没有了意义。而所有的技术讨论都依赖于在第 2 章中所建立的基础。
- 第 3 ~ 6 章介绍了任何 Spring 开发人员都应该注意的事项。这些概念不仅适用于较旧版本的 Spring 应用程序，也适用于新的应用程序。第 5 章介绍了如何同时兼容新旧版本的应用程序（无论是否使用了 Spring）。
- 第 7 章和第 8 章介绍了一些基于 HTTP 的微服务系统中的概念，包括安全性和路由。
- 第 9 ~ 12 章可以帮助你更好地管理和处理分布式系统中的数据。
- 第 13 章介绍了一个真正的核心概念，因为它依赖于其他一些技术概念，所以我们在本书前半部分介绍核心概念和测试时没有介绍它。可运维的应用程序应该是可观测的。尽早了解本章的基本原理，会帮助你理解本书的其他内容。
- 第 14 章介绍了如何使用 Spring（一个云原生的开发框架）来构建平台和云端的组件。本章对开放服务代理的讨论尤为深刻。
- 最后，第 15 章提炼了有关持续交付的知识。整本书是按照持续交付的方式编写的，这对于我们正在努力做的事情至关重要，因为我们选择用结果来证明一切。务必阅读本章。

在线资源

我们提供了很多有用的在线资源来帮助你理解书中的内容：

- 本书的代码可以在 GitHub 资料库（*http://github.com/cloud-native-java/*）中找到。
- 从 Spring 网站（*http://spring.io/*）上能找到关于 Spring 的一切资料，包括文档、技术问答论坛等。
- Cloud Foundry 网站（*http://cloudfoundry.org*）是由 Cloud Foundry 基金会的所有贡献者通力完成的。你会在上面找到相关的视频、教程、新闻等。

本书使用约定

本书使用以下印刷约定。

斜体（*Italic*）

斜体表示新的术语、URL、电子邮件地址、文件名和文件扩展名等。

等宽字体（Constant width）

用于程序清单，以及段落中引用的程序元素，例如变量或函数名、数据库、数据类型、环境变量、语句和关键字等。

等宽字体加粗（**Constant width bold**）

显示应该由用户输入的命令或其他文本。

等宽斜体（*Constant width italic*）

该处内容应该被由用户提供的值或者上下文所确定的值替换。

表示一个提示或建议。

表示一般注释。

表示警告或注意。

如何联系我们

请将对本书的评价和存在的问题通过如下地址告知出版者：

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网站，你可以在那里找到关于本书的相关信息，包括勘误列表、示例代码以及其他信息。本书的网站地址是：

http://bit.ly/cloud-native-java

对于本书的评论和技术性的问题，请发送电子邮件到：

bookquestions@oreilly.com

关于我们的书籍、课程、会议和新闻的更多信息，请参阅我们的网站 *http://www.oreilly.com*。

在 Facebook 上找到我们：*http://facebook.com/oreilly*
在 Twitter 上关注我们：*http://twitter.com/oreillymedia*
在 YouTube 上观看我们：*http://www.youtube.com/oreillymedia*

使用代码示例

你可以从 *http://github.com/Cloud-Native-Java* 下载补充材料（代码示例、练习等）。

这本书是为了帮助你完成工作而编写的。一般来说，如果本书提供了示例代码，那么你可以在程序和文档中使用它们。除非你大量使用本书中的代码，否则不需要与我们联系。例如，编写一段使用了本书中多段代码的程序不需要我们授权。销售或者发行 O'Reilly 书中代码示例的 CD-ROM 需要授权。引用本书内容和示例代码来回答问题不需要授权。将本书中重要的示例代码用于产品文档中需要授权。

使用我们的代码时，希望你能标明它的出处。出处一般要包含书名、作者、出版商和 ISBN，例如，“Book Title by Josh Long and Kenny Bastani (O'Reilly). Copyright 2017 Josh Long, Kenny Bastani, 978-1-449-37464-8.”。

如果还有其他使用代码的情形需要与我们沟通，可以随时与我们联系：*permissions@oreilly.com*。

致谢

首先，我们要感谢 O'Reilly 出版社的编辑 Nan Barber 和 Brian Foster，他们付出了令人难以置信的耐心，并给了我们很大的支持。

感谢所有给我们提供观点、想法和灵感的审稿人员。非常感谢 Pivotal 公司和它的庞大商业生态给我们的支持。特别感谢提供技术反馈的每位审稿人员，其中包括 Brian

Dus-sault、Dave Syer 博士、Andrew Clay Shafer、Rob Winch、Mark Fisher 博士、Mark Pollack 博士、Utkarsh Nadkarni、Sabby Anandan、Michael Hunger、Bridget Kromhout、Russ Miles、Mark Heckler、Marten Deinum、Nathaniel Schutta 和 Patrick Crocker，等等。谢谢!

最后，我们要感谢 Rod Johnson 和 James Watters。你们不求回报，给予了我们一切需要的帮助。非常感谢你们的序言、反馈和灵感。

本书是使用 Asciidoctor 工具编写的，该工具由 OpenDevise 的 Dan Allen（*http://twitter.com/mojavelinux*）和设计合作伙伴 Sarah White（*http://twitter.com/carbonfray*）领导开发。所有示例的源代码存放在 GitHub（*http://github.com*）的公共仓库中。代码持续集成使用了 Travis CI（*https://travis-ci.org/cloud-native-java*）。构建产出物托管在由 JFrog（*https://www.jfrog.com*）慷慨捐赠给我们的 Artifactory 仓库中。所有的代码示例都可以在由 Pivotal 托管的 Pivotal Web Services 上运行。如果没有这些工具，编写本书不会如此顺利，无论是开源社区还是由社区运维的公司，都在免费支持我们。非常感谢你们!我们希望你的下一个项目会考虑使用他们的工具，并支持他们，因为他们曾经支持过我们。

想对 Spring 团队（*http://spring.io/team*）和 Cloud Foundry 团队（*http://cloud foundry.org*）说，非常感谢你们为我们编写所有代码、进行测试和对我们予以所有支持。

Josh Long

我要感谢我的合著者 Kenny，加入我的这次冒险旅程! 我想感谢 O'Reilly 给我们这次机会，以及他们给了我们难以置信的宽容，因为我们为了能够完整介绍 Pivotal 的生态系统而一再延期。谢谢 Dave Syer 博士（*https://twitter.com/david_syer*）对本书致言和给予我鼓励。不管在任何城市、时区、国家和大洲，我都在不断编程，而 Pivotal 的 Spring 和 Cloud Foundry 团队无论何时都在帮助我，谢谢你们!

Kenny Bastani

首先，我要感谢我的朋友、导师和同事 Michael Hunger，他首先激发了我为开源软件写书和发声的热情。我也要感谢我非常有才华的合著者 Josh Long，当我两年前把我的第一个 Spring Boot 微服务投入生产时，他邀请我和他一起写这本书。与他一起写这本书是一次令人难以置信的冒险。从 Josh 和我在纸上写下第一行字起，我们看到了 Spring Boot 逐渐成长为最成功的开源项目之一。今天，这种增长意味着每月 Spring Boot 被下载 1300 万次，其中包括新的应用程序和持续的生产环境部署。达到这些成绩，Spring