Broadview®
www.broadview.com.cn

追随Google | Facebook | Amazon | Netflix

化大而复杂为小而简单，用快速交付支撑持续创新

微服务架构与实践

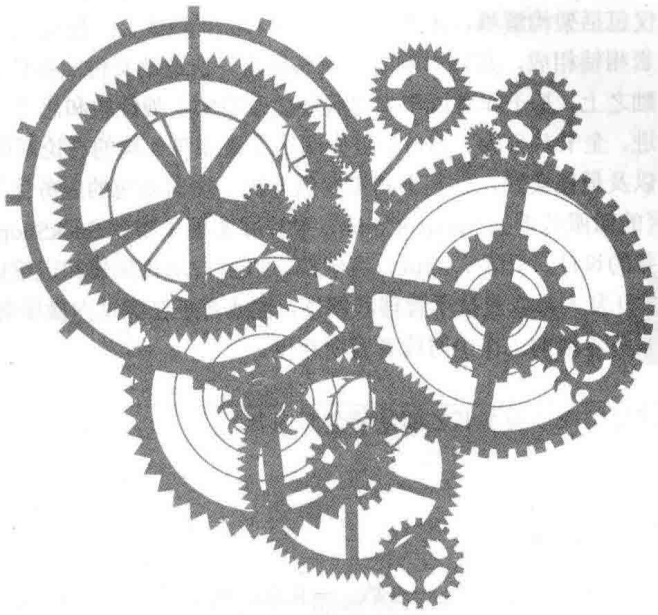
（第2版）

王磊　马博文　张琦　著

张桐　赵国庆　审校

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



微服务架构与实践

RFID

（第2版）

王磊　马博文　张琦　著
张桐　赵国庆　审校

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

内 容 简 介

微服务架构不仅延续了分布式系统与 SOA 的特征，也汲取了 DevOps、持续集成、持续交付等工程实践的成功经验，并正在借着云计算和容器化的春风开始其驰骋之旅。但是，微服务的落地并不像其概念描述的那样举重若轻，它不仅包括架构解耦，还涉及开发测试、部署运维、工程实践、团队合作与康威定律等多方面的因素，这些因素相辅相成，共同影响着如何高质量、快速地交付业务价值。

本书是在第 1 版的基础之上，基于作者近年来对服务化改造的实战经验和思考，并结合业界的技术趋势进行的一次体系化的精进。全书共分为 3 部分，首先阐述了微服务架构的理论基础。其次介绍了微服务生态系统、实施参考模型以及最佳实践，并基于真实案例分析了遗留系统的服务化改造策略与应用场景。最后基于 Apache 开源社区的微服务框架 ServiceComb，设计并实现了案例 SockShop 系统，从端到端交付的角度，指导读者完成服务的设计、开发、测试、流水线，以及自动化部署和运维体系的建立。

本书不仅适合架构师、开发人员以及技术管理者阅读，也适合正在尝试向微服务架构迁移的团队或者个人。希望本书能够在微服务落地的工作中对读者有所帮助。

图书在版编目（CIP）数据

微服务架构与实践 / 王磊，马博文，张琦著. —2 版. —北京：电子工业出版社，2019.4
ISBN 978-7-121-34994-2

Ⅰ. ①微… Ⅱ. ①王… ②马… ③张… Ⅲ. ①互联网络－网络服务器 Ⅳ. ①TP368.5

中国版本图书馆 CIP 数据核字（2018）第 206322 号

策划编辑：张春雨
责任编辑：牛　勇
印　　刷：三河市君旺印务有限公司
装　　订：三河市君旺印务有限公司
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：32.25　　字数：639 千字
版　　次：2015 年 11 月第 1 版
　　2019 年 4 月第 2 版
印　　次：2019 年 4 月第 1 次印刷
定　　价：108.00 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。

推荐序

10 多年前，对云计算提出的口号是“能够像水电一样，通过网络为全社会提供公共计算服务”。现在，计算服务化已经不再是口号，而是成为了现实。整个 IT 行业的商业模式从卖产品转向卖服务，这也就意味着，传统的面向产品的应用服务架构，也需要向面向服务的应用架构转型。

云计算技术的基本思路是将 IT 资源集中起来，利用虚拟化技术，让更多的用户能够共享资源，从而提高资源的利用率，降低服务成本。但用这种做法的同时也随之带来了一些新的技术挑战，比如如何设计新架构以适应海量并发用户和多样性的用户需求等，这导致云计算及其应用日益复杂。

软件工程师解决复杂性的核心思路的方法一直就是将架构功能模块化。云计算让软件服务化，微服务就是在云服务的基础上，让软件进一步模块化，以服务的方式提供更好的灵活性。

从微服务概念的提出，到近几年大家谈云必谈微服务，及 CloudNative 将微服务作为应用架构的事实标准可以看出，微服务架构正在成为应用架构设计的主流模式。

但是，通过业界几年的案例和实践来看，微服务架构的落地也与其他技术的落地类似，是个循序渐进的演进过程。它不仅包括架构解耦，还涉及测试机制、部署运维、工程实践、团队协作等多方面，这些因素相辅相成，共同影响着如何高质量、快速地交付业务。

本书的几位作者王磊、马博文、张琦，是国内较早接触微服务、DevOps 以及实现微服务框架的架构师、实践者，可以说本书是集方法论、指南与实践经验为一体的指导书籍。

本书的第一部分以理论基础为主，主要介绍了应用软件架构的发展历史，阐述了微服务架构的本质，并剖析了微服务架构未来的演进趋势，即 Serverless 和 Service Mesh；第二部分介绍了微服务生态系统、微服务架构的关键技术、微服务实施参考模型以及重要的落地实践。所谓“光说不练假把式”，对于一本技术书来说，“show me the code”是重要的环节；在第三部分中，本书基于 Apache 的微服务开源框架 ServiceComb 以及华为云 ServiceStage，完整地搭建了案例系统，并详细地阐述了从服务的分析、设计、开发，到测试、部署及运维端到端的内容。

感谢本书的几位作者将宝贵的经验分享出来，祝本书的读者们能够享受微服务架构的带来的种种乐趣。

中国信息通信研究院云大所所长　何宝宏博士

2018 年 11 月 23 日

通过这几年在项目中实践微服务，为客户提供微服务咨询，我越来越肯定所谓“微服务（Micro Service）”其实一点都不“微”！这非 Martin Fowler 定义之过。他所定义的概念，突出了设计每个独立服务时要分解的粒度，但对于整个架构风格而言，实践微服务并不如概念所表现的那样举重若轻，然后“轻而易举”。一个微服务从诞生到最后完成使命，经历了设计、开发、测试、上线运行到下线，其贯穿了整个生命周期。每个环节都有方方面面的因素需要考量，如设计原则的遵守、通信机制的选择、数据一致性的保障、健康状态的监控与跟踪，乃至于服务的配置、测试与运维，还有对企业的组织结构进行改革，使之与微服务架构风格相匹配，等等。

我们还能说微服务是小而美吗？

为什么还有这么多企业对其趋之若鹜呢？个人认为，微服务是结构与规模的权衡。当两者不可兼得时，选择降低规模以应对业务复杂度的增长，牺牲结构来换取对复杂系统的控制。观察前面提及的诸多考量要素，是否都属于技术实现的范畴呢？对于软件系统的实现者而言，我们无法决定业务复杂度的增长，但却可以将技术复杂度掌控在自己手中。因此，微服务利用“分而治之”的思想减小了系统的规模，使得每个微服务的开发者不用面对复杂的业务逻辑，即使业务发生了变化，在如此小规模的“微”服务中，我们也能轻松

面对——最坏的情况也就是重写一个微服务！

这种分解带来规模减小的同时，引入的却是分布式系统固有的技术复杂度。这种复杂度依旧挑战着编程的极限。但我们对于未来却不再是茫然不可知了，微服务将业务实现的难题转向了基础设施的难题，进而成为整个软件行业面对微服务的共同“焦虑点”。终于，我们能够抛开具体业务的束缚了！面对这共同的“焦虑点”，一些技术先行者开发了公共的微服务平台与框架，以图解决基础设施的问题，于是就有了 Netflix OSS、Spring Cloud、Service Fabric、ServiceComb、Istio 等微服务框架。而云端的 PaaS 与容器化技术又解决了服务部署、服务编排、服务监控等基础问题。面向业务的微服务开发者们，就可以面对真正的小而美的“微”服务了。

分解业务复杂度、控制技术复杂度为业界“拥抱”微服务的根本原因。

分解业务复杂度，是服务设计要解决的问题；控制技术复杂度，则提升到系统工程的高度，需要考虑团队组织、框架选型、部署运维等微服务全生命周期的每个环节。幸运的是，王磊、马博文和张琦的这本书全面地回答了这两方面的问题。我极为认同书中提到的“微服务生态系统”这一概念。以“生态系统”为名，就能够让大家在看到微服务之“微”的同时，还能重视微服务“不微”的另一面。面对微服务的应用开发人员，我们希望隐藏它“不微”的一面。然后将这背后的苦活儿、累活儿、脏活儿交给架构的设计者，微服务平台的实现者，让他们去构造和打磨内部精巧细微的零部件与支撑平台。

从“不微”进入到“微”，是从宏观世界进入微观世界，整个过程其实是一个艰巨的架构设计过程。无论是在新项目中“拥抱”微服务架构，还是将遗留的单块架构迁移到微服务架构，都需要慎重，在微服务的优势与不足中取得良好的平衡。“To be or not to be”，这是个问题。而王磊兄结合自己这数年来实践微服务架构的心得体会，总结了“微服务实施参考模型”，通过适用性评估、成熟度参考与度量指标，让选择的难题迎刃而解。这是本书的一大创见！

从计算机的发展历史来看，微服务是一个新生产物，但它不是从石头缝里突然蹦出来的，它的设计思想其实是分布式系统与 SOA 的延续，同时又汲取了 DevOps、持续集成、持续交付等工程实践，并借着云计算和容器化的春风开始了它的驰骋之旅。因此，要做好微服务架构，既需要在业务和技术方面钻研得够深，又需要具有涵盖整个生命周期的广博知识，但两者很难兼得，但本书的内容恰恰在这两方面都得到了很好的体现。我想，这一方面得益于本书已是第 2 版，在前一版全面介绍微服务架构的基础之上，是一次体系化的精进，内容臻于成熟。同时又依赖于王磊兄在 ThoughtWorks、华为亲身经历的众多大型项

目，收获了大量实践微服务架构的经验。

微服务是与时俱进的。从 Martin Fowler 提出“微服务”概念至今，不过三、四年时间，这其中已经诞生了许多能“呼风唤雨”的平台和框架。在软件设计领域，实现技术瞬息万变，唯有设计思想方能历久弥新。幸运的是，本书既通过实战案例为你提供了微服务落地的参考。同时又高屋建瓴地对微服务架构思想进行了全方位的梳理。本书通过合理巧妙的章节分布与匠心独运的内容编排，同时兼顾了“微”与“不微”，专与博，变与不变，真可以说是一本“贪心”之作，但对于读者而言，不正是一壶可以贪杯的佳酿吗？

架构编码实践者　张逸

2018 年 11 月 12 日于成都

我和王磊老师是在 ThoughtWorks 的同事。2014 年，我们一起在帮助当时的客户实施微服务改造，我们在不同的项目上实践着微服务。那个时候的“微服务”对我来说仅仅是“把应用中的一部分用 Restful API 调用，并且通过持续部署的方式部署到生产环境上”。

当我还对微服务的落地懵懵懂懂时，王磊老师已经带领一个团队开始规模化地实践微服务了。他把所遇到的问题和技术实践系统地总结下来，这就是《微服务架构与实践》（第 1 版）一书。作为当时国内较早的微服务领域的书籍，它揭开了微服务的帷幕，将完整的微服务技术和实践展现在我面前，使我知道原来我所了解的仅仅是微服务的冰山一角。

在接下来的几年里，我将自己所在项目的经验推广到了其他项目上，开始了我的“DevOps 和微服咨询”生涯，这段时间也正是微服务在中国开始流行起来的时候。在微服务的理论和实践在国内不断深入的同时，新的技术和挑战也在不断出现。在我们最早实现微服务时，还不存在 Spring Cloud“全家桶”，也没有 Kubernetes 这样的容器编排工具。当时能够获取的资源可能就是 Netflix 提供的开源项目（当然，现在其中一部分已经不存在了，比如 Asgard）。然而，国外的微服务实践经验对于国内而言，可借鉴的部分十分有限。而国内的微服务实践，往往侧重于工具技术的应用，却缺乏了方法论层面的指导。

作为微服务在国内的先驱者和探索者，王磊老师在微服务领域中不断地总结和探索，继续完善着微服务的理论和实践，在近几年指导并参与了更多的微服务改造项目，获得了更多的实践经验，并总结出了系统的微服务实践方法论。这些内容就融合在了这本《微服务架构与实践》（第 2 版）中。

了解第 1 版的朋友，可能会发现第 2 版比第 1 版“增容”不少，不光更新了第 1 版中

相关的内容，使得体系更加清晰，而且还介绍了微服务的业界发展，如 Serverless 和 Service Mesh。不过，我认为最重要的内容是本书中对“微服务实施参考模型”的梳理。微服务的实施就像在探险，“当你没有方向的时候，贸然行动可能是最危险的”。就像王磊老师在书中描述的那样，微服务的实施“不仅仅是架构的拆分，还涉及团队协作、工程实践、基础设施、部署运维等多个方面，这些因素相辅相成，互相依赖，共同影响着如何高质量、快速的交付业务。”

对此我深有体会。架构的演进，不光是技术的升级，还有随之带来的组织结构和工程实践等多方面问题，这些都是决定微服务实施成败的关键因素。如果忽略了其他因素的相互作用，就很容易在微服务的实践中被多方莫名的阻力困住，难以前行。书中提出的“微服务实施参考模型”就像一张地图，将微服务实施的全貌展现了出来。它汇聚了众多微服务实践者和应用的智慧，解决了“从哪开始”、“到哪里去”和“如何去”这三个关键的问题，并且把可能遇到的问题标注在了这张“地图”中。

如果你已经陷入微服务改造的泥沼，希望本书可以给你指明前行的方向。如果你是微服务的初学者，本书的第 3 部分“实战篇”则给了你一个理论结合实践，去完整“体验微服务”的机会。这部分通过案例，让你对从架构设计、技术选型开始，到使整个微服务完整地落地有深入的了解。如果你还没有完整的微服务实施经历，千万不要错过这个部分。

我依然记得第一次在项目上完成微服务改造时，那种完成架构和工程实践的美好感受，始终留存在我的心里。在这几年里，虽然服务着不同的客户、不同的技术栈、不同的项目，但每当我迷茫的时候，都会回味第一次完成微服务架构改造的那种架构之美。希望你也能通过《微服务架构与实践》（第 2 版）体验微服务的架构之美。让你在迷茫时回归初心，找到前行的方向。

愿架构之美与你同在。

埃森哲技术咨询经理　顾宇

2018 年 11 月 11 日

认识王磊的时候，我刚到华为公司开始着手华为微服务框架 ServiceComb 的开源工作，而恰逢他内部指导多个团队落地微服务架构。微服务架构的落地需要框架的支持，而微服务框架的推广需要实践作为指导。这样我们一拍即合，开始密切的合作。

在《微服务架构与实践》（第 2 版）的撰写过程中，我见证了 ServiceComb 进入 Apache

孵化的全过程。目前，ServiceComb 已从 Apache 基金会孵化器“毕业”并正式成为高级项目，而《微服务架构与实践》（第 2 版）这本书也即将出版，我应邀写下这篇序作为纪念。

传统的单体应用模块内的耦合程度很高，可谓牵一发动全身。为了应对业务的快速发展，降低业务组件的耦合方式，大家开始把应用中的各个模块按照微服务的方式进行拆分，各服务以独立进程方式进行部署，服务之间跨越网络进行通信。虽然云化和容器化的大量采用简化了服务的部署和运维，但也随之带来了很多挑战，如无法预先知道服务实例的地址、不能假设服务一直都是可用的、需要根据系统运行调整服务负载，而这些也正是微服务框架需要面临的挑战。

业界微服务框架一般会从两个视角来解决这类问题：一个是从开发视角；另一个是从运维视角。从开发视角，会将微服务架构的过程中所涉及的服务发现、负载均衡，容错熔断、限流降级，以及监控等环节，通过框架封装，使应用构建变得简单、高效；从运维视角，关注的则是网络层面的问题，通过接管服务间的网络请求，解决微服务架构带来的服务注册发现、熔断、容错、服务动态路由以及系统监控的问题。

ServiceComb 源于华为云的微服务引擎 CSE，它沉淀了华为内多个项目的高性能、低延时的能力，旨在帮助用户完成服务的快速开发以及云上工作。通过阅读《微服务架构与实践》（第 2 版），读者不仅可以系统化地学习微服务架构的落地思路，也可以通过使用 ServiceComb 框架，结合华为云所提供的工具，实现服务的持续集成、自动布署，以及服务治理方案。

愿大家在《微服务架构与实践》（第 2 版）的陪伴下开启美好的微服务探索之旅。

华为开源能力中心技术专家　姜宁

2018 年 12 月 7 日

2014 年，随着 Martin Fowler 和 James Lewis 合著的《Microservice》一文的公开，微服务的概念被业界所接受，并逐渐变成架构设计的主流模式。从那时开始，ThoughtWorks 作为微服务架构的主要推动机构，就一直在海内外帮助企业导入微服务架构，针对遗留系统进行微服务和服务化改造。在这一过程中我们认识到，无论是国内还是国外的企业，对微服务的认知从一开始的模糊不清，已经逐渐变得清晰而明朗。高内聚、松耦合的架构设计思想，配合以轻量级的 API 实现技术，使得开发工作从原来的统一步调的大团队开发模式，逐渐向小团队、自助步调的开发模式演进。在过去的四年时间里，许多企业围绕如何能够

更好地发挥微服务架构的优势，如何能够使得团队能力与微服务架构的要求相匹配，一直在不断地探索，各种微服务框架、技术平台也在技术社区中“你方唱罢我登场”。开发团队在选择不同的技术平台时，也逐渐陷入两难的境地。

自从2000年以后，企业对软件开发过程的诉求，从瀑布时代的“稳定”和“可预期”，逐渐过渡到了对于业务响应能力的追求。实现这一目标最简单的方法（最起码看起来是）是从管理思路和研发过程角度下手。敏捷、持续交付、DevOps等过程管理手段逐渐被业界所接受。回想一下，2010年前后，当软件行业大规模地开展敏捷转型的时候，500人以上的团队是很常见的。这500人要基于统一的代码库、统一的发布节奏、统一的持续集成（CI）流水线来“排兵布阵”。为了应对这样的问题，各种所谓规模化敏捷方案的实施框架开始在企业中被推广和使用。LeSS、SAFe，甚至大规模的Kanban，都是定位于解决大规模开发团队实时敏捷转型问题。但是随着时间的推移，很多团队发现仅从管理和技术实践角度入手去解决响应力的问题，会带来一定的副作用。在我曾经参与的一个项目中，200人的团队被划分成若干个特性团队（Feature Team），每个特性团队从Web界面一直做到固件（Firmware）层。但是代码库还是一个统一的Repo，包含了Java和C语言代码。团队在需求划分、持续集成、自动化测试方面都遇到了不少的困难，并且一个典型问题是，同样一个业务逻辑，由于不同的特性团队的实施，最终在代码库中形成了五个不同的实现方式。通过静态代码扫描工具完全无法发现这样的“重复代码”，而当业务规则发生变化时，团队要付出很多精力去找到并修改代码库中所有相关代码。

早在微服务出现之前，业界已经希望通过模块化和模型驱动的方式，来解决架构方面的问题。2003年，《领域驱动设计》一书出版。作者Eric希望通过一种合理的建模手段，能够帮助团队更清晰地划分职责的边界，使业务模型能够和技术实现模型统一，从而解决大规模复杂系统的研发问题。但是领域驱动设计（DDD，Domain-Driven Design）这一概念的提出，在技术社区里面一直是曲高和寡。除了个别公司在导入DDD的思想和方法，大部分企业不仅没有尝试使用DDD，甚至连这一概念都不曾听闻。但这一情况在微服务架构出现之后发生了显著变化。

微服务架构不仅要求产品从系统架构的层面将原来的单体架构拆分开，同时也要求在团队的层面，依据“康威定律”进行团队的重组。团队、系统、开发库拆分以后，很多系统仍然需要在所有微服务之间进行端到端的“联调、联测”来保证质量，这样做无法让团队可以根据自己微服务的需要来自行决定发布频率。以Pact为代表的“契约测试工具”的出现，很好地解决了这一问题。在契约被满足的条件下，端到端的测试的工作量可以被最

大程度地降低，降低到仅仅覆盖关键用户旅程的水平。各个微服务可以按照自己的频率来发布，服务内的特性也可以做到独立发布。用这样的方法，产品实现了从原来的统一的大版本发布的模式向服务级小版本发布模式的过渡。由于微服务的粒度更小，在一个服务的范围内来确保和守护领域模型也变得可行。

至此，团队还有两个问题亟待解决：第一个是如何有效地识别服务的边界；第二个是大量的微服务被开发出来以后，如何有效地进行运维和演进。针对第一个问题业界已经达成共识，领域驱动设计当中的战略层建模是一种十分有效的识别服务边界的方法。基于限界上下文来构建微服务，可以使团队最大程度地利用微服务架构所带来的好处。同时，DevOps 已经是业界针对第二个问题的标准答案。依托开源社区的不断贡献，以及各大厂商对于微服务平台孜孜不倦的研发努力，目前市场上已经有很多种工具和平台，可以帮助开发团队以最快的速度将微服务的代码部署到生产环境上。同时，完备的监控工具也实时地将生产系统上的各种信息反馈给开发团队，从而让开发团队能够针对线上的情况实时做出决策并且安全执行。

随着容器技术和云技术的不断发展，基础设施和计算资源被更好地抽象，进而为微服务技术的不断演进提供了可能。两年以前，对于微服务平台的选择基本上只有 Spring Cloud 一种。但是到了今天，各大厂商都推出了自己的微服务工具，不仅支持了 Spring Cloud 当中原有的功能，而且还提供了类似 Service Mesh 等技术，使团队可以进一步将原有单一技术栈的微服务系统，逐步演进成多语言的微服务系统。可以相信，在未来的很长一段时间里，会有更多的新技术支撑微服务的快速开发，除了为开发者提供更好的开发体验。同时也会提供更好的弹性基础设施，使应用系统对于业务响应能力的追求迈上一个新的台阶。

我认识王磊的时候，他担任 ThoughtWorks 的咨询顾问。在 ThoughtWorks 这样一个社区性质的公司内部，王磊是很有影响力的。王磊当时所参与的项目，是 ThoughtWorks 全球较早采用微服务架构的几个项目之一。在这一项目上，客户实现了契约测试工具 Pact，并开源给整个技术社区使用，而王磊也将自己的经验和感悟写成《微服务架构与实践》（第 1 版）一书。这也是我们在为客户提供咨询时，一本经常被提到的推荐读物。在经过几年的打磨和探索，结合业界的技术趋势和工程实践，王磊出版的这本《微服务架构与实践》（第 2 版），相信对很多希望和正在采用微服务架构的技术人员、架构师而言，是一本很有帮助的读物。

DDD China 联合发起人/ThoughtWorks 首席咨询师　王威

2018 年 11 月 13 日

时至今日，随着云计算、大数据、人工智能等技术的迅猛发展，企业数字化转型的步伐也正在加速。越来越多的企业已不是在探讨要不要做数字化转型，而是在探索和践行数字化之路。企业希望通过数字化转型实现业务上的敏捷、智能、高效。敏捷是指既包括商业模式快速更替，也包括产品研发快速迭代；智能是指实现业务的快速创新；高效是指实现业务的有效和高速运转。

企业的业务系统要实现“敏捷、智能、高效”，其面临的关键挑战是如何构建和驾驭一个强大而复杂的分布式系统。微服务和容器等云原生技术近几年蓬勃发展，有专家学者认为微服务和容器是企业数字化转型下大型分布式系统和技术的战略选择。通过微服务的架构设计，将整个系统拆分成若干个小而独立自治的微服务，从应用架构层面实现应用的灵活性，让“开发”的轮子快起来；容器技术实现了对运行环境更小力度、更轻量化地控制，从而使得应用程序运行标准化和具有伸缩性。

非常有幸受邀写此推荐序，在此结合自己近几年在华为从事的业务：进行微服务化的技术探索、高性能企业级微服务框架 CSE 设计、ServiceComb 开源，以及华为云微服务云应用平台 ServiceStage 在支撑企业微服务转型的经验，与大家分享一些经验。

首先，企业通过微服务化后，业务提升的效果非常显著，业务上线周期大大缩短，周期从年缩短到月，从月缩短到周，甚至是天，这解决了“需求落地慢”的难题。同时也解决了传统“烟囱”架构，诸如“多应用孤立”、“多数据打通难”等难题，进而也解决了业务层面的问题，如“政策落地慢”、“业务监管难”、“用户体验差”等难题。

其次，微服务转型之路可以说是一个系统性工程，它涉及企业的系统架构、组织、流程等方面的变革和创新。这需要我们下定决心，包括我们的业务“一把手”和技术专家；当然，心急吃不了热豆腐，微服务架构技术支持分步而做，分期演进。

再次，就是微服务框架选型，当前开源的微服务框架有很多，对于选择哪个框架不用纠结，因为没有完美的框架，选择合适的框架就好了。架构的开放性、多语言支持、性能、负载均衡、灰度发布、微服务治理能力等都是考虑因素。

最后，“专业的人干各自专业的事”，云时代的应用是相互依存的，“Service on Service”的方式是快速构建企业应用的捷径，建议我们企业的开发人员更聚焦业务的“痛点”分析、微服务设计、开发、上线，实现业务的快速创新。

行动是成功的起点，相信本书能够给你带来一段美好的微服务之旅。

华为云 PaaS 解决方案总监　饶争光

2018 年 11 月 25 日

自序

2014 年，微服务架构的概念在国外刚兴起，国内提及并付诸实践的人还并不多。我基于在 ThoughtWorks 工作期间对海外某大型房产平台的微服务改造的经验，撰写了本书的第 1 版，介绍了微服务架构的概念、背景以及优缺点，并通过一个遗留系统微服务改造的案例，阐述了微服务的理论和相关实践。

技术的发展日新月异，微服务现在已经从一个“流行语”落地到了诸多互联网公司及传统企业。在技术大会上，讨论的议题不再是微服务的概念、优缺点，而是落地的场景、相关实践以及如何应对大规模服务化带来的挑战等。另外，随着持续交付、DevOps 理念被广泛地接受，Kubernetes、云基础设施的成熟，以及 Service Mesh 等概念的兴起，实现微服务的方式正在发生快速的变化。

近几年，在帮助数个团队将其产品迁移到微服务架构后，我深切地感受到：微服务的落地不仅仅只是架构解耦，还涉及工程实践、部署运维、团队协作等多个方面，这些因素相辅相成，共同影响着如何高质量、快速地交付业务价值。同时，我也发现：有些朋友对微服务的期望过高，将其视为解决现有架构痛点、消除技术债务、提升团队能力的“银弹”。加上社区对微服务的热捧，导致他们过于迫切地希望大刀阔斧地进行改造，而忽略了演进的过程。实际上，由于存在业务模式、技术积累、组织结构等差异性，微服务改造，尤其对于存量系统庞大的组织而言，很难一蹴而就。另外，市面上出版了诸多与微服务架构相关的书籍，但对实施和演进过程做系统化梳理的并不多。随着微服务架构在更多企业的落

地，我认为如何系统化地演进和落地微服务将成为一大挑战。

因此，我萌发了更新原书的想法。首先是与读者分享在过去几年中我在数个团队中落地微服务的心得——书中的设计模式以及工程实践，大多数是来自我对亲历实践的梳理，其中的参考模型帮助我指导了多个团队有效地推进遗留系统的改造。其次，基于热心读者的反馈，也对本书的开发语言以及技术框架（使用 Java 作为开发语言，基于微服务开源框架 ServiceComb 构建样例）进行了更新。另外，还介绍了微服务架构在业界的最新发展，包括微服务与 Serverless 以及 Service Mesh 等的关系。

最后，感谢我的妻子晓丽和儿子锦熙，没有你们的支持和鼓励，我不可能完成这项工作。感谢挚友马博文、张桐、赵国庆一直以来的支持和帮助，你们对本书的服务化测试、部署运维提供了诸多思路。感谢华为的同事，你们积极参与审校并提出了宝贵建议。这些同事（排名不分先后）包括：饶争光、张琦、姜宁、陈弘、刘珊珊、吴继敏、崔毅华、张龙春、田晓亮、周天、闫敏之。感谢电子工业出版社的张春雨和负责本书审校工作的编辑们，本书能够出版，离不开你们一丝不苟的工作态度和敬业精神。

由于时间仓促及作者水平有限，书中难免有疏漏之处，在此敬请广大读者批评指正。在阅读本书的过程中，也可通过微信 5109343 或邮箱 wldandan@gmail.com 与我联系。祝读者们享受微服务的实践之旅！

王磊

2018 年 7 月 1 日于西安

2011 年入职 ThoughtWorks 后，我一直在澳洲一家房地产在线广告公司工作，负责交付项目，配合客户逐步实施敏捷开发、持续集成、自动化测试、蓝绿部署、持续交付、微服务化、云化、容器化、Serverless，我所在企业的组织结构也从职能型组织结构演进为类似 Spotify 的敏捷性组织结构。

在这个过程中，首先完成了将系统从瀑布式开发改进为敏捷式开发，将持续集成、TDD、自动化测试等推广到整个组织中。现在我依然记得当年实现 Cucumber 测试做数据准备以及在 CI 上修复环境的痛苦。这些痛苦换来的好处是，每个迭代（两周）的产出都可以部署到生产环境中。随后，服务可以通过“蓝绿部署”的方式实现“零宕机时间”部署，这样自动化部署的任务可以很容易地集成到交付流水线中。

这些工程实践的应用，就像是铺就了一条高速公路，让后面的微服务改造变得异常顺利。同时，由于部署已经自动化，基础设施管理权限和能力也下放到了全功能团队中，对运维管理的监控、日志聚合也都采取比较合理的方案，微服务数量增加而带来的维护成本，由全功能团队承担，分摊在每个人身上的 on-call 时间很少。

维护成本低带来的好处是，微服务拆分的粒度可以非常小，小到只有一个接口。这些微服务设计和实现的经验，也让客户后来可以自如地采用 Serverless 部署微服务。

基于这些实际的微服务演进经历，在开始构想本书时，我们对应地为微服务参考模型的每个维度添加了自身曾经经历过的，或者一些业界的工程实践的案例，为读者在实施微服务时，提高每个维度下的成熟度提供参考。

在参与本书撰写期间，大概半年多都几乎每天凌晨 1 点左右休息。特别感激我的爱人王嘉能一直支持我，她虽然担心我的身体健康，但更能理解本书对我的重要意义。

借此机会，感谢曾经在刚开始工作时给予我无限帮助的文静、聪明姐、吴少、黄拓和姜鹏。感谢现在团队的同事王磊、桐桐、国庆对我的帮助。同时，也感谢曾经一起工作过的朋友们，没有他们的帮助和影响，我无法对本书做出这些贡献，书中的很多实践案例也是来自他们的智慧和启发。

I feel so grateful to work with all of you: JP who coached me in coding and swearing, Trent who led me into the DevOps world, Cos who gave me comprehensive guidance to become a qualified DevOps, Jeol who set an example of an outstanding tech leader, and Clayton who inspired me to think big in the data pipeline spike. My heartfelt thanks also go to Collins, Mike Williams, and other REA personnel who helped me and gave me enlightening ideas without any reservation. I'm an introvert and not good at rhetoric, but right here is the best opportunity for me to deliver my greatest gratitude with all my readers as witnesses.

最后，祝愿各位读者能在阅读本书的过程中有所收益，既可以形成自己的微服务理论体系，也能将实践应用在具体项目中。

马博文

2018 年 7 月 5 日于西安

随着互联网对各个行业的深度渗透，它对行业的改变除了使行业有了新的业务形式，还有对业务更新节奏的提速。近两年在与处于各种不同行业的朋友的交流中，感受最深的一点就是“这世界变化太快了”。如果前两年这种“快”的影响还只在互联网领域，那么现在几乎所有的行业都已经被裹挟到这个浪潮中来了。而“微服务”便是在这样的大势之下应运而生，由前两年互联网公司的“玩具”转变为被更多企业级 IT 系统所接受和尝试东西。

IT 领域的技术更新节奏是十分迅速的，自本书第 1 版出版以来，随着容器技术、DevOps 技术的蓬勃发展，“适应业务高速变化的系统架构”已经成长了很多。Kubernetes 的火爆，将容器技术带入了复杂的企业级系统中，对有状态的服务的支持，对复杂应用的编排能力，让更多的人们认识到，不是所有的应用都可以简单到无状态，“农场散养模式”也不一定适合所有类型的应用。Service Mesh 的微服务实现形式的发展，让大家认识到，终于有人开始认为传统的微服务实现方式还是不够快；而单一语言开发、单一协议通信也并不能满足很多实际的需求。在“简单为王”的另一面，人们终于认识到这个世界本就不是那么简单的。互联网褪去了前期的浮华，终于让整个行业和技术的发展进入到“不落两边”的境界。而正是这种技术发展的趋势，让越来越多的企业级应用开始选择和使用微服务架构。本书的前两篇建立在我们对企业级应用的微服务实践上，着重讲解在这个过程中使用到的种种技术。

然而任何一个曾经尝试过在自己的项目中使用过“微服务”的读者都应该有所感受，“微服务”总是看上去很美，做起来却又“冷暖自知”了。在我参与过的几个微服务项目中，业务架构师在一开始的期望总是想让我一下就告诉他，他的系统怎么做就可以一下就变成微服务架构，可以享受微服务带来的种种好处，但是实际情况却是，不存在这种“速成”的方法。其实从开发流程、组织形式、工具选用、架构方式、实现技术等各个阶段，无不属于微服务的范畴。即便是微服务拆分，也在种种理论之下要充分结合本业务、本系统的实际情况逐步进行。所以本书在“实践篇”中花大篇幅向读者展示了一个以微服务架构构建的系统的始末。从理论的实践到工具的选用，再到各个环节的实施，这是希望读者在此基础上可以引发对实践的思考，并能够进一步以此为启发，自己可以对微服务架构和技术有所领悟。

我衷心地希望本书的读者可以从中获益，了解微服务架构，掌握微服务架构，自己实践微服务架构。这便是作为 IT 从业人员在“搬砖”之外能得到的最大慰藉了。

张琦

2018 年 9 月 1 日于北京

前言

本书的结构

本书一共分为 3 个部分，分别是基础篇、策略篇和实战篇。基础篇为第 1 章，主要介绍微服务架构相关的基础知识。该章首先介绍软件架构的演进史；其次阐述了微服务出现的背景、定义、特征及落地时面临的挑战；同时分析了微服务与 SOA、Serverless 的关系；最后介绍了微服务领域 Service Mesh 的兴起。阅读的重点为理解微服务的本质特征、挑战并了解 Service Mesh。

策略篇包含第 2 章至第 6 章，主要介绍了微服务生态系统、微服务关键技术、微服务实施参考模型以及基于参考模型的实践，并在本篇最后的部分阐述了遗留系统改造的策略与案例。

第 2 章首先介绍了微服务生态系统，并围绕生态系统阐述微服务实现中涉及的接入层、业务层、支撑层及基础设施。同时，也强调了开发框架、交付流水线与工程实践的重要性。阅读的重点是理解微服务生态系统的核心，系统化地思考微服务架构的演进过程（不仅仅是服务拆分），并在演进中持续提升团队能力。

第 3 章的内容是对第 2 章内容的延伸，详细介绍了微服务的设计（服务划分策略、服务设计模式、内部实现结构等）、微服务治理（负载均衡、注册发现、容错机制等）的原理及方案、服务运维（监控告警、日志聚合等）的实践等。第 3 章内容的重点是微服务的划