云架构指南

Cloud Native 分布式架构

原理与实践

柳伟卫 ◎ 著

北京大学出版社
PEKING UNIVERSITY PRESS

云架构指南

Cloud Native
分布式架构
原理与实践

柳伟卫◎著

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 提 要

Cloud Native（云原生）是以云架构为优先的应用开发模式。目前，越来越多的企业已经开始大规模地“拥抱云”——在云环境下开发应用、部署应用及发布应用等。未来，越来越多的开发者也将采用 Cloud Native 来开发应用。本书是国内 Java 领域关于 Cloud Native 的著作。

本书全面讲解了基于 Cloud Native 来构建应用需要考虑的设计原则和实现方式，涵盖 REST 设计、测试、服务注册、服务发现、安全、数据管理、消息通信、批处理、任务调度、运营、容器部署、持续发布等方面的 Cloud Native 知识。同时，书中所讲解的技术方案皆为业界主流的技术，极具前瞻性。最后，本书除了讲解 Cloud Native 的理论知识，还会在每个知识点上辅以大量的代码案例，使理论可以联系实践，具备更强的可操作性。

本书主要面向对分布式系统、微服务、Cloud Native 开发感兴趣的计算机专业的学生、软件开发人员和系统架构师。

图书在版编目(CIP)数据

Cloud Native 分布式架构原理与实践 / 柳伟卫著. — 北京 : 北京大学出版社, 2019.3
ISBN 978-7-301-30089-3

Ⅰ. ①C… Ⅱ. ①柳… Ⅲ. ①程序语言—程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2018)第274491号

书　　名	Cloud Native 分布式架构原理与实践 CLOUD NATIVE FENBUSHI JIAGOU YUANLI YU SHIJIAN
著作责任者	柳伟卫　著
责任编辑	吴晓月
标准书号	ISBN 978-7-301-30089-3
出版发行	北京大学出版社
地　　址	北京市海淀区成府路205 号　100871
网　　址	http://www.pup.cn　　新浪微博: @ 北京大学出版社
电子信箱	pup7@pup.cn
电　　话	邮购部 010-62752015　发行部 010-62750672　编辑部 010-62570390
印 刷 者	北京市科星印刷有限责任公司
经 销 者	新华书店
	787毫米×1092毫米　16开本　21印张　488千字
	2019年3月第1版　2019年3月第1次印刷
印　　数	1-4000册
定　　价	79.00 元

本书献给我的妻子 Funny，愿她永远青春靓丽！

前言

Preface

写作背景

未来越来越多的企业将会“拥抱云”。特别是对于中小企业及个人开发者而言，以云架构为优先的Cloud Native应用开发模式将会深入人心。Cloud Native能帮助企业快速推出产品，同时节省成本。

笔者结合自身的云计算工作经验，以及对于Cloud Native的思考，将这方面的知识整理成册，内容涵盖REST设计、测试、服务注册、服务发现、安全、数据管理、消息通信、批处理、任务调度、运营、容器部署、持续发布等方面的知识，希望帮助读者从理论和实践两方面来深刻理解Cloud Native。

源代码

本书提供源代码下载，下载地址为https://github.com/waylau/cloud-native-book-demos。

本书所涉及的技术和相关版本

技术的版本是非常重要的，因为不同版本之间存在兼容性问题，而且不同版本的软件所对应的功能也是不同的。本书所列出的技术在版本上相对较新，都是经过笔者大量测试的。这样读者在自行编写代码时，可以参考本书所列出的版本，从而避免版本兼容性所产生的问题。建议读者将相关开发环境设置得跟本书一致，或者不低于本书所列的配置。详细的版本配置，可以参阅本书“附录”中的内容。

本书示例采用Eclipse编写，但示例源码与具体的IDE无关，读者可以选择适合自己的IDE，如IntelliJ IDEA、NetBeans等。运行本书示例，请确保JDK版本不低于JDK 8。

勘误和交流

本书如有勘误，会在https://github.com/waylau/cloud-native-book-demos上进行发布。笔者在编写本书的过程中，已竭尽所能地为读者呈现最好、最全的实用功能，但错漏之处在所难免，欢迎读者批评指正，也可以通过以下方式直接联系我们。

博客：https://waylau.com

邮箱：waylau521@gmail.com

微博：http://weibo.com/waylau521

开源：https://github.com/waylau

致谢

感谢北京大学出版社的各位工作人员为本书的出版所做的努力。

感谢我的父母、妻子和两个女儿。由于撰写本书，牺牲了很多陪伴家人的时间，在此感谢家人对我工作的理解和支持。

柳伟卫

目录

Contents

第1章

Cloud Native 概述

1.1 当今软件发展的现状

当今软件行业正发生着巨变。自 20 世纪 50 年代计算机诞生以来，软件从最初的手工作坊式的交付方式，逐渐演变成了职业化开发、团队化开发，进而制订了软件行业的相关规范，形成了软件产业。

今天，无论是大型企业还是个人开发者，都或多或少采用了云的方式来开发应用、部署应用。不管是私有云还是公有云，都将给整个软件产业带来变革。个人计算机或以手机为代表的智能设备已经走进寻常百姓家。几乎每个人都拥有手机，手机不仅是通信工具，还能发语音、看视频、玩游戏，让人与人之间的联系变得更加紧密。智能手环随时监控用户的身体状况，并根据每天的运动量、身体指标提供合理的饮食运动建议。出门逛街甚至不需要带钱包，吃饭、购物、搭车时使用手机就可以支付费用，多么便捷。智能家居系统更是用户生活中的“管家”，什么时候该睡觉了，智能家居系统就自动拉上窗帘并关灯；早上起床了，智能家居系统会自动拉开窗帘并播放动人的音乐，让用户可以愉快地迎接新的一天；用户再也不用担心家里的安全情况，智能家居系统会监控一切，有异常情况时会及时发送通知到用户的手机中，让用户第一时间掌握家里的情况。未来，每个人都能拥有《钢铁侠》（*Iron Man*）中所描述的智能管家 Jarvis，而这一切都离不开背后那个神秘的巨人——分布式系统。正是那些看不见的分布式系统，每天处理着数以亿计的计算，提供可靠而稳定的服务。这些系统往往是以 Cloud Native 方式来部署、运维的。

1.1.1 软件需求的发展

早期的软件大多数是由使用该软件的个人或机构研制的，所以软件往往带有非常强烈的个人色彩。早期的软件开发也没有什么系统的方法论可以遵循，完全是“个人英雄主义”，也不存在所谓的软件设计，纯粹就是某个人的思想的表达。而且，当时的软件往往是围绕硬件的需求来定制化开发的，有什么样的硬件就有什么样的软件。所以，软件缺乏通用性。同时，由于软件开发过程不需要与他人协作，因此，除了源代码外，往往没有软件设计、使用说明书等文档。这样，就造成了软件行业缺乏经验的传承。

20 世纪 60 年代中期到 70 年代中期是计算机系统发展的第二个时期，在这一时期软件开始被当作一种产品广泛使用。所谓产品，就是可以提供给不同的人使用，从而提高软件的重用率，降低软件开发的成本。例如，以前一套软件只能专门提供给某个人使用，而现在，同一套软件可以批量卖给不同的人。就分摊到相同软件上的开发成本而言，卖得越多，成本自然就越低。这个时期，出现了类似“软件作坊”的专职替别人开发软件的团体。虽然是团体协作，但软件开发的方法基本上仍然沿用早期的个体化软件开发方式，这样导致的问题是，软件的数量急剧膨胀，软件需求日趋复杂，软件的维护难度也就越来越大，开发成本变得越来越高，从而导致软件项目频频失败。这就演变成了“软件危机”。

“软件危机”迫使人们开始思考软件的开发方式，使人们开始对软件及其特性进行了更加深入的研究，人们对待软件的观念也在悄然改变。由于早期计算机的数量很少，只有少数军方或者科研机构才有机会接触到计算机，这就让大多数人认为，软件开发人员都是稀少且优秀的（一开始也确实如此，毕竟计算机最初的制作者都是数学界的天才）。由于软件开发的技能只能被少数人所掌握，因此大多数人对于“什么是好的软件”缺乏共识。实际上，早期那些被认为是优秀的程序常常很难被别人看懂，其中充斥着各种程序技巧，加之当时的硬件资源比较紧缺，迫使开发人员在编程时往往需要考虑更少地占用计算机资源，从而采用不易阅读的“精简”方式来开发，这加重了软件的个性化。而现在人们普遍认为，优秀的程序除了功能正确、性能优良之外，还应该让人容易看懂、容易使用、容易修改和扩充。这就是软件可维护性的要求。

1968 年 NATO（北大西洋公约组织）会议上首次提出“软件危机”（Software Crisis）这个名词，同时，提出了期望通过“软件工程”（Software Engineering）来解决“软件危机”。“软件工程”的目的就是要把软件开发从“艺术”和“个体行为”向“工程”和“群体协同工作”转化，从而解决“软件危机”包含的两方面的问题。

（1）如何开发软件，以满足不断增长、日趋复杂的需求。

（2）如何维护数量不断膨胀的软件产品。

事实证明，事先对软件进行可行性分析，可以有效地规避软件失败的风险，提高软件开发的成功率。

在需求方面，软件行业的规范是，需要提供相应的软件规格说明书、软件需求说明书，从而让开发工作有依据，划清开发边界，并在一定程度上减少“需求蔓延”情况的发生。

在架构设计方面，需要提供软件架构说明书，划分系统之间的界限，约定系统间的通信接口，并将系统分为多个模块。这样更容易将任务分解，从而降低系统的复杂性。

今天，制定软件需求的方式越来越多样化了。客户与系统分析师也许只是经过简单的口头讨论，制订了粗略的协议，就安排开发工程师进行原型设计了。开发工程师开发一个微服务，并部署到云容器中，从而实现软件的交付。甚至不用编写任何后台代码，直接使用云服务供应商所提供的 API，就可以使应用快速推向市场。客户在使用完这个应用时，马上就能将自己的体验反馈到开发团队，使开发团队能够快速响应客户的需求变化，并促使软件进行升级。

通过敏捷的方式，最终软件形成了“开发—测试—部署—反馈”的良性循环，软件产品也得到了进化。而这整个过程，都比传统的需求获取方式将更加迅捷。

1.1.2 开发方式的巨变

早些年，瀑布模型还是标准的软件开发模型。瀑布模型将软件生命周期划分为制订计划、需求分析、软件设计、程序编写、软件测试和运行维护 6 个基本活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。在瀑布模型中，软件开发的各项活动严格按照线性方