有问题，就找黑马程序员问答精灵！

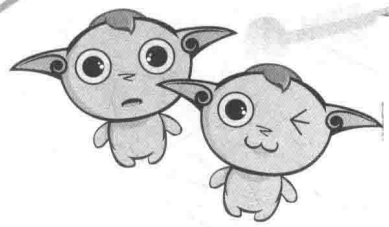
微服务架构基础

Spring Boot+Spring Cloud+Docker

黑马程序员 编著

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

有问题，就找黑马程序员问答精灵！



微服务架构基础

Spring Boot + Spring Cloud + Docker

黑马程序员 ◉ 编著

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

微服务架构基础 : Spring Boot+Spring Cloud+Docker / 黑马程序员编著. -- 北京 : 人民邮电出版社, 2018.4
ISBN 978-7-115-44320-5

Ⅰ. ①微… Ⅱ. ①黑… 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(2017)第325569号

内 容 提 要

本书以 Spring Boot+Spring Cloud+Docker 技术为基础，从当下流行的微服务架构理念出发，详细讲解了微服务和微服务架构方面的技术知识。全书共分为四部分：第一部分“微服务概述”，主要讲解微服务的由来、概念、特点和微服务架构等；第二部分“微服务的开发”，主要讲解微服务开发框架 Spring Boot 的使用；第三部分“微服务架构的构建”，主要讲解如何使用 Spring Cloud 的相关组件来构建微服务架构；第四部分“微服务的部署”，主要讲解 Docker 技术，以及如何在 Docker 中部署微服务项目。

本书适合所有 Java 开发人员，尤其适合正在学习微服务，以及正在尝试使用微服务架构开发项目的人员阅读和参考。

◆ 编　　著　黑马程序员
责任编辑　范博涛
责任印制　马振武

◆ 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
北京隆昌伟业印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：12　　2018 年 4 月第 1 版
字数：293 千字　　2018 年 4 月北京第 1 次印刷

定价：35.00 元

读者服务热线：(010)81055256　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

学科介绍 Subject Introduction

Java 是世界上非常流行的计算机编程语言，在全球编程语言排行榜居于首位，其中 Java EE 技术目前占据着绝大部分的企业级开发市场。在智联招聘上搜索“Java”关键字，仅北京的招聘职位就可搜索到 3 万多个，可见人才需求量很大。

黑马程序员的 Java EE 学科经过十一年的磨炼，已形成全面且系统的课程体系，并不断地紧随市场进行课程升级，打造契合企业需求的核心内容。课程以实战为导向，通过“理论 + 实践”的学习模式，帮助学员完成从理论学习到实战开发的蜕变，从而使学员真正获得实际开发的能力。

课程设置 Curriculum Provision

唯有与时俱进的课程，才能成就高薪传奇。Java EE 课程结合市场，不断更新课程体系，完全贴合企业的最新人才需求。

课程特点

- **夯实基础** – 课程内容经过不断地优化和完善，涵盖了 Java EE 基础的所有技术。
- **由浅入深** – 知识讲解循序渐进，由入门案例到详细的工作原理及特性解析，逐步深入。
- **真实案例** – 针对知识点的不同特性设计切合企业项目的案例。
- **项目实战** – 项目课程融合当前流行的框架和技术，引入企业真实开发环境。

序言 PREFACE

江苏传智播客教育科技股份有限公司（简称传智播客）是一家致力于培养高素质软件开发人才的科技公司，“黑马程序员”是传智播客旗下高端 IT 教育品牌。

“黑马程序员”的学员多为大学毕业后，想从事 IT 行业，但各方面条件还不成熟的年轻人。“黑马程序员”的学员筛选制度非常严格，包括了严格的技术测试、自学能力测试，还包括性格测试、压力测试、品德测试等。百里挑一的残酷筛选制度确保学员质量，并降低企业的用人风险。

自“黑马程序员”成立以来，教学研发团队一直致力于打造精品课程资源，不断在产、学、研 3 个层面创新自己的执教理念与教学方针，并集中“黑马程序员”的优势力量，有针对性地出版了计算机系列教材 50 多册，制作教学视频数十套，发表各类技术文章数百篇。

“黑马程序员”不仅斥资研发 IT 系列教材，还为高校师生提供以下配套学习资源与服务。

为大学生提供的配套服务：

1. 专业的辅助学习平台“博学谷”（http://yuanxiao.boxuegu.com），专业老师在线为您答疑解惑。

2. 针对高校学生在学习过程中存在的压力等问题，我们还面向大学生量身打造了“播妞”。“播妞”不仅致力推行快乐学习，还会有定期的助学红包雨。同学们快来添加“播妞”微信/QQ：208695827。

3. 高校学生也可扫描下方二维码，加入“播妞”粉丝团，获取最新学习资源，与“播妞”一起快乐学习。

为 IT 教师提供的配套服务：

针对高校教学，“黑马程序员”为 IT 系列教材精心设计了“教案+授课资源+考试系统+题库+教学辅助案例”的系列教学资源，高校老师请关注码大牛老师微信/QQ：2011168841，获取配套资源，也可以扫描下方二维码，加入专为 IT 教师打造的师资服务平台——“教学好助手”，获取“黑马程序员”最新教师教学辅助资源的相关动态。

传智播客和黑马程序员

前言 FOREWORD

为什么要学习本书

最近两年，微服务一词逐渐地进入了技术人员的视野，并已成为当下最火的技术名词之一。这里的微服务并不是指某一个技术或者某个服务，而是一种理念。通过此理念的使用，逐渐地发展出了一种流行的架构——微服务架构。

微服务架构是指由一系列职责单一的细粒度服务构成的分布式网状结构，其基本思想在于围绕着业务领域创建应用，这些应用可独立地进行开发和管理。简单来说，微服务架构的目的就是有效拆分应用，实现敏捷开发和部署。

微服务架构适合有一定的扩展复杂度，且有很大用户增量预期的应用。通常来说，比较适合新兴的互联网公司项目或有升级需求的传统企业应用。随着技术的不断发展，各种企业对微服务架构的使用需求将越来越多。

虽然使用微服务架构技术的市场需求在不断增加，但掌握相关技术的人员却很少。尤其当前市面上的相关资料、书籍并不多，且讲解 Spring Boot+Spring Cloud+Docker 技术的资料更是少之又少，这也在一定程度上制约了微服务架构技术的发展。

为了帮助更多的技术人员了解并掌握微服务架构技术的使用，本书以通俗易懂的语言、典型翔实的案例，对微服务架构技术的使用进行详细讲解。对于想快速学习微服务技术的人员，以及对 Spring Boot、Spring Cloud 和 Docker 有兴趣的人员来说，本书将是您很好的选择。

如何使用本书

本书适用于具有一定 Java Web 框架（如 Spring 框架）、Maven 工具和 Linux 系统使用基础的技术人员，以及对微服务感兴趣的业务人员学习。对于想深入学习的非技术人员，建议先掌握 Java 框架技术、Maven 工具以及 Linux 系统的使用。

本书在 Spring Boot + Spring Cloud + Docker 的基础上，详细讲解了微服务架构技术使用的相关知识。在编写时，作者力求将一些非常复杂、难以理解的思想和问题简单化，使读者能够轻松、快速地掌握这些知识点。

本书共 10 章，每章的内容如下。

- 第 1 章讲解微服务及其相关的技术。主要包括微服务和微服务架构的概念、产生背景、微服务架构的优势与不足、如何搭建微服务架构，以及如何选择微服务架构技术。通过本章的学习，读者将对微服务及其相关概念有一定的了解，并熟识常用的微服务架构技术。
- 第 2 章对 Spring Boot 的由来、特点、使用要求、入门程序以及工作机制进行详细讲解。通过本章的学习，读者可以体会到 Spring Boot 框架的方便和高效，并能了解 Spring Boot 的执行过程。

- 第 3 章讲解如何使用 Spring Boot 与其他技术进行集成开发，内容包括 Spring Boot 与 MyBatis 框架的集成、与 Redis 的集成，以及与 ActiveMQ 的集成。通过本章的学习，读者将熟悉如何在实际开发中应用 Spring Boot。
- 第 4 章讲解微服务架构中的服务发现以及客户端负载均衡。服务发现是通过 Spring Cloud Eureka 实现的，而客户端负载均衡是通过 Spring Cloud Ribbon 实现的。
- 第 5 章讲解微服务架构中的服务容错保护、API 网关服务，以及分布式配置管理的使用知识，其中服务容错保护使用的是 Spring Cloud Hystrix，API 网关服务使用的是 Spring Cloud Zuul，分布式配置管理使用的是 Spring Cloud Config。学习完本章后，结合前面所学知识，读者将可以搭建一个比较完整的微服务架构。
- 第 6 章讲解 Docker 入门的一些基础知识，内容包括 Docker 的概念和特点、安装要求和安装方式，以及运行机制。通过本章的学习，读者可以对 Docker 的概念及其体系架构有一个初步的了解，并能够掌握在 Ubuntu 系统上安装 Docker 的几种方式。
- 第 7 章讲解 Docker 的基本使用及镜像管理的一些知识，内容涉及 Dockerfile 文件、Docker 客户端的常用指令等。通过本章的学习，读者可以掌握 Docker 的基本使用，同时能够掌握 Docker 中的镜像管理。
- 第 8 章讲解 Docker 中的网络与数据管理知识，内容包括 Docker 的默认网络和自定义网络管理、Docker Swarm 的集群、Docker 的数据存储，以及 Volumes 数据卷的管理。通过本章的学习，读者可以对 Docker 中的网络、数据管理以及 Docker Swarm 的基本知识有一定的了解，同时能够掌握 Docker 中自定义的网络管理和 Volumes 数据卷管理的具体使用方法。
- 第 9 章讲解微服务项目的整合以及接口测试的相关知识，内容包括使用微服务架构搭建的一个商城管理系统，以及接口可视化工具 Swagger-UI 的使用。通过本章的学习，读者可以对微服务项目的使用有进一步的认识，熟悉 Spring Boot 和 Spring Cloud 相关组件的整合开发，同时还可以掌握接口测试工具 Swagger-UI 的简单使用。
- 第 10 章讲解有关微服务部署的相关知识，内容涉及 Docker Comopse 编排工具、微服务与 Docker 的整合、微服务手动部署，以及使用 Jenkins 完成微服务的自动化部署等内容。通过本章的学习，读者可以掌握微服务与 Docker 的整合，同时能够掌握如何使用 Jenkins 完成微服务项目的自动化集成和部署。

在学习过程中，读者一定要亲自实践书中的案例代码，如果不能完全理解书中所讲的知识点，可以登录博学谷平台，通过平台中的教学视频进行辅助学习。另外，如果读者在理解知识点的过程中遇到困难，建议不要纠结于某个地方，可以先往后学习。通常来讲，随着对后面知识的不断深入了解，前面看不懂的知识点一般就能理解了。如果读者在动手练习的过程中遇到问题，建议多思考，理清思路，认真分析问题发生的原因，并在问题解决后多总结。

本书采用基础知识+案例相结合的编写方式，通过基础知识的讲解与案例的巩固，可以使读者快速地掌握技能点。

致谢

本书的编写和整理工作由传智播客教育科技股份有限公司完成，其中主要的参与人员有吕春林、陈欢、韩永蒙、石荣新、杜宏、梁桐、王友军、冯佳等。全体人员在近一年的编写过程中，付出了很多辛勤的汗水，在此一并表示衷心的感谢。

意见反馈

尽管我们尽了最大的努力，但书中难免会有不妥之处，欢迎各界专家和读者朋友们来函给予宝贵意见，我们将不胜感激。您在阅读本书时，如发现任何问题或不认同之处，可以通过电子邮件（itcast_book@vip.sina.com）与我们取得联系。

传智播客.黑马程序员
2017 年 12 月 4 日于北京

目 录 CONTENTS

Spring Boot+Spring Cloud+Docker

Chapter 1

第1章 认识微服务架构

学习目标

- 了解微服务和微服务架构的概念
- 熟悉微服务架构的优点与不足
- 了解微服务架构的技术选型

微服务（Microservice）概念的提出已经有很长一段时间了，但在最近几年才开始频繁地出现。虽然现在很多公司都开始采用微服务及其架构来满足实际需求，但这种方式并没有普及，很多开发人员还只停留在听说过“微服务”或“微服务架构”这些词上。那么为什么需要微服务架构？什么是微服务架构？又如何去构建微服务架构呢？本章将针对这些问题进行一一讲解。

1.1 为什么需要微服务架构

任何一个新事物或者新技术的出现，必然有其出现的原因，微服务架构也不例外。随着互联网技术的发展，传统的应用架构已满足不了实际需求，微服务架构就随之产生。那么传统应用架构到底出了什么问题呢？又如何解决？接下来的两个小节中，我们将从传统单体架构的问题开始，对为什么需要微服务架构进行详细讲解。

1.1.1 传统单体应用架构的问题

通常我们所使用的传统单体应用架构都是模块化的设计逻辑，程序在编写完成后会被打包并部署为一个具体的应用，而应用的格式则依赖于相应的应用语言和框架。例如，在网上商城系统中，Java Web 工程通常会被打成 WAR 包部署在 Web 服务器上，而普通 Java 工程会以 JAR 包的形式包含在 WAR 包中，如图 1-1 所示。

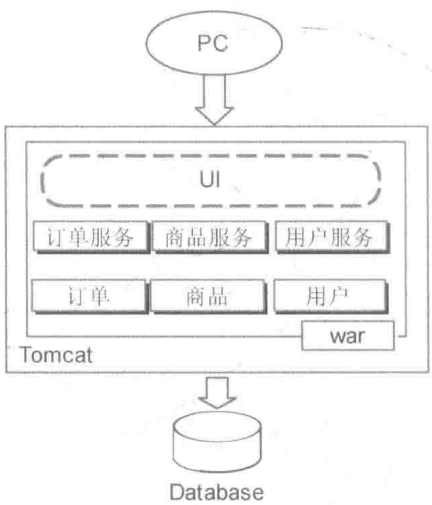


图1-1 早期单体架构

图 1-1 中的这种应用开发风格很常见，它易于开发和调试，并且易于部署。在用户量不多时，此种架构方式完全可以满足需求，但随着用户人数的增加，一台机器已经满足不了系统的负载，此时我们就会考虑系统的水平扩展。通常情况下，我们只需要增加服务器的数量，并将打包好的应用拷贝到不同服务器（如 Tomcat），然后通过负载均衡器（如 Apache、Nginx）就可以轻松实现应用的水平扩展，如图 1-2 所示。

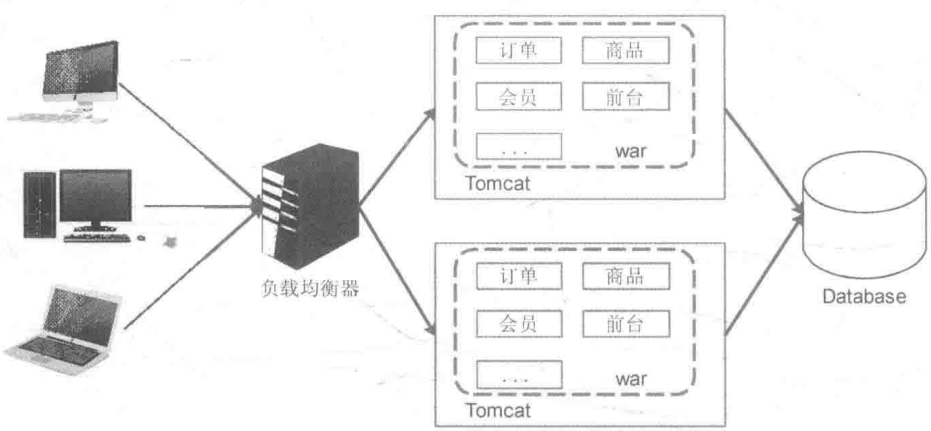


图1-2 传统单体应用架构

在早期，单体架构的这种扩展方式可以很好地满足使用需求，但随着时间的推移，这种方式就会产生很多问题，具体表现如下。

1. 应用复杂度增加，更新、维护困难

一个简单的应用会随着时间的推移而逐渐变大。一旦应用变得庞大而又复杂，开发团队将会面临很多问题，其中最主要的问题就是这个应用太复杂，以至于任何单个开发者都很难进行二次开发或维护。

2. 易造成系统资源浪费

虽然使用负载均衡的方式可以对项目中的服务容量进行水平扩展，但由于传统单体架构的代码中只有一个包含所有功能的 WAR 包，所以在对服务容量扩容时，只能选择重复地部署这个 WAR 包来扩展服务能力，而不仅仅是扩展了所需的服务。这样就会导致其他不需要扩展的服务也进行了相应的扩展，但这些扩展是不需要的，因此这种方式会极大地浪费资源。

3. 影响开发效率

当一个应用越大时，启动时间就会越长。开发和调试的过程中，如果有很大一部分时间都要在等待中度过，那么必然会对开发效率有极大的影响。

4. 应用可靠性低

传统单体应用架构在运行时的可靠性比较低，当所有模块都运行在一个进程中时，如果任何一个模块中出现了一个 Bug，可能会导致整个进程崩溃，从而影响到整个应用。

5. 不利于技术的更新

传统单体应用架构一旦选定使用某些技术，则后期的开发和扩展将在这些技术的基础上实现。如果需要更改某种技术，则可能需要将整个应用全部重新开发，这种成本是非常大的。

当然，传统单体应用架构的问题还不只这些，但出现这些问题的根本原因可以说就是由于传统单体架构中一个 WAR 包内包含了系统的所有服务功能所导致的。随着业务变得越来越多，问题也就越来越多。

1.1.2　如何解决传统应用架构的问题

针对传统单体架构的问题，大部分企业通过 SOA（Service-Oriented Architecture，面向服务的架构）来解决上述问题。SOA 的思路是把应用中相近的功能聚合到一起，以服务的形式提供出去，因此基于 SOA 架构的应用可以理解为一批服务的组合。

同样以网上商城为例，一个简单的 SOA 系统如图 1-3 所示。

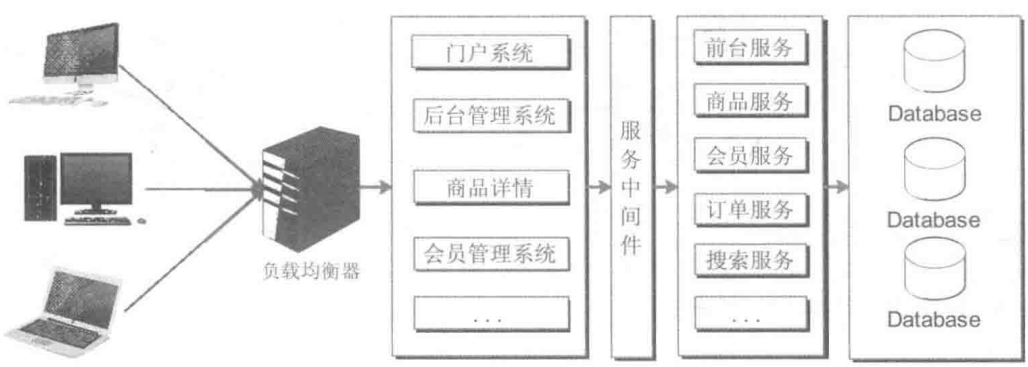


图1-3　SOA系统

从图 1-3 中可以看出，SOA 将原来的单体架构按照功能细分为不同的子系统，然后再由各个子系统依赖服务中间件（这里指企业服务总线 Enterprise Service Bus，简称 ESB）来调用所需服务。

使用 SOA 可以将系统切分成多个组件服务，这种通过多个组件服务来完成请求的方式有很多好处，具体如下。

- 把项目拆分成若干个子项目，不同的团队可以负责不同的子项目，从而提高开发效率。
- 把模块拆分，使用接口通信，降低了模块之间的耦合度。
- 为企业的现有资源带来了更好的重用性。
- 能够在最新的和现有的应用之上创建应用。
- 能够使客户或服务消费者免予服务实现的改变所带来的影响。
- 能够升级单个服务或服务消费者而无需重写整个应用，也无需保留已经不再适用于新需求的现有系统。

虽然使用 SOA 解决了单体架构中的问题，但多数情况下，SOA 中相互独立的服务仍然会部署在同一个运行环境中（类似于一个 Tomcat 实例下，运行了很多 web 应用）。和单体架构类似，随着业务功能的增多，SOA 的服务会变得越来越复杂。本质上看，单体架构的问题并没有因为使用 SOA 而变得更好。

针对单体架构和 SOA 的问题，许多公司（如 Amazon、eBay 和 NetFlix）通过采用微处理结构模式解决了系统架构中的问题。其思路不是开发一个巨大的单体式的应用，而是将应用分解为小的、互相连接的微服务。随着微服务的使用，微服务架构的思想也随之产生。

1.2 微服务架构是什么

1.2.1 微服务架构的概念

微服务架构是一种架构风格和架构思想，它倡导我们在传统软件应用架构的基础上，将系统业务按照功能拆分为更加细粒度的服务，所拆分的每一个服务都是一个独立的应用，这些应用对外提供公共的 API，可以独立承担对外服务的职责，通过此种思想方式所开发的软件服务实体就是“微服务”，而围绕着微服务思想构建的一系列体系结构（包括开发、测试、部署等），我们可以将它称之为“微服务架构”。

根据微服务架构的定义，将传统单体架构拆分为微服务架构的方式，如图 1-4 所示。

从图 1-4 中可以看出，微服务架构已将传统单体架构中的订单服务、商品服务和用户服务拆分为了独立的服务，其中的每一个服务都是一个独立的应用，可以访问自己的数据库，这些服务对外提供公共的 API，并且服务之间可以相互调用。

注意

微服务和微服务架构是两个不同的概念。微服务强调的是服务的大小，它关注的是某一个点，而微服务架构是一种架构思想，需要从整体上对软件系统进行全面的考虑。

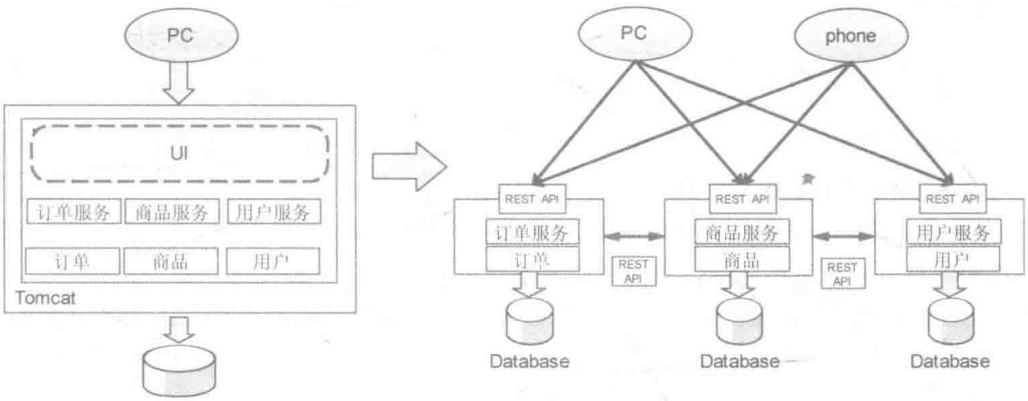


图1-4　传统单体架构拆分为微服务架构

1.2.2　微服务架构的优点

与传统单体应用架构相比，微服务架构有很多优点，具体表现如下。

1. 复杂度可控

微服务架构在将应用分解的同时，规避了原本复杂度无止境的积累。每一个微服务专注于单一功能，并通过定义良好的接口清晰地表述服务边界。由于体积小、复杂度低，每个微服务可由一个小规模开发团队完全掌控，易于保持高可维护性，并提高了开发效率。

2. 可独立部署

由于微服务具备独立的运行进程，所以每个微服务都可以独立部署。当某个微服务发生变更时，无需编译、部署整个应用。由微服务组成的应用相当于具备一系列可并行的发布流程，使得发布更加高效，同时降低了对生产环境所造成的风险，最终缩短应用交付周期。

3. 技术选型灵活

微服务架构下，技术的选型是多样化的。每个团队都可以根据自身服务的需求和行业发展的现状，自由选择最适合的技术。由于每个微服务相对简单，当需要对技术进行升级时，所面临的风险较低，甚至完全重构一个微服务也是可行并容易的。

4. 易于容错

当架构中的某一组件发生故障时，在单一进程的传统架构下，故障很有可能在进程内扩散，导致整个应用不可用。在微服务架构下，故障会被隔离在单个服务中。若设计良好，其他服务可通过重试、平稳退化等机制实现应用层面的容错。

5. 易于扩展

单个服务应用也可以实现横向扩展，这种扩展可以通过将整个应用完整地复制到不同的节点中实现。当应用的不同组件在扩展需求上存在差异时，微服务架构便体现出其灵活性，因为每个服务可以根据实际需求独立进行扩展。

6. 功能特定

每个微服务都有自己的业务逻辑和适配器，并且一个微服务一般只完成某个特定的功能，例如商品服务只管理商品、客户服务只管理客户等。这样开发人员可以完全地专注于某一个特定功能的开发，而不用过多地考虑其他，从而提高开发效率。除此之外，微服务架构还有很多其他优势，由于篇幅有限，这里就不一一列举了，但从微服务架构的优势可以看出，使用微服务可以很

好地解决传统单体架构中的问题。

1.2.3 微服务架构的不足

微服务架构除了有上面所讲的各种优点外，还存在着一些不足，这些不足的具体表现如下。

1. 开发人员必须处理创建分布式系统的复杂性

- 开发工具（或 IDE）是面向构建传统的单体应用程序的，不为开发分布式应用程序提供全面功能上的支持。
- 测试更加困难。在微服务架构中，服务数量众多，每个服务都是独立的业务单元，服务主要通过接口进行交互，如何保证依赖的正常，是测试面临的主要挑战。
- 开发人员必须实现服务间的通信机制。
- 实现用例跨多个服务时，需要面对使用分布式事务管理的困难。
- 实现跨多个服务的用例，需要团队之间进行仔细的协调。

2. 部署的复杂性

在部署和管理时，由许多不同服务类型组成的系统的操作比较复杂，这将要求开发、测试及运维人员有相应的技术水平。

3. 增加内存消耗

微服务架构用多个服务实例取代了 1 个单体应用程序实例，如果每个服务都运行在自己的 JVM 中，那么有多少个服务实例，就会有多少个实例在运行时的内存开销。

1.2.4 微服务架构与 SOA 的区别

通过前 3 个小节的学习，相信读者对微服务架构已经有了一定的了解。在学完后，细心的读者可能会有这样一个疑问，微服务架构与 SOA 都是对单体架构的拆分，那么它们有什么不同呢？

下面通过一个表格对两者的区别进行对比，如表 1-1 所示。

表 1-1 微服务架构与 SOA 的区别

微服务架构	SOA
一个系统被拆分成多个服务，粒度细	服务由多个子系统组成，粗粒度
团队级，自底向上开展实施	企业级，自顶向下开展实施
无集中式总线，松散的服务架构	企业服务总线，集中式的服务架构
集成方式简单（HTTP/REST/JSON）	集成方式复杂（ESB/WS/SOAP）
服务能独立部署	服务相互依赖，无法独立部署

1.3 如何构建微服务架构

了解了微服务架构的概念、优点与不足后，相信很多人对微服务架构都会产生这样一些疑问，例如我要何时使用微服务架构？又如何将应用程序分解为微服务？分解后，要如何去搭建微服务架构？同时，在微服务架构中，因为会涉及多个组件，那么这些组件又可以使用什么技术来实现呢？接下来的几个小节中，我们将对这些问题进行详细地讲解。