快速构建 AI 应用

——AWS 无服务器 AI 应用实战

[美] 彼得•埃尔格(Peter Elger)
伊恩•沙纳吉(Eóin Shanaghy) 著
殷海英 译

清華大學出版社
北 京

北京市版权局著作权合同登记号　图字：01-2022-0248

Peter Elger，Eóin Shanaghy
AI as a Service, Serverless Machine Learning with AWS
EISBN: 978-161729-615-4

图书在版编目(CIP)数据

快速构建 AI 应用：AWS 无服务器AI 应用实战 / (美) 彼得・埃尔格 (Peter Elger)，(美) 伊恩・沙纳吉 (Eóin Shanaghy) 著；殷海英译. —北京：清华大学出版社，2022.6
书名原文：AI as a Service，Serverless Machine Learning with AWS
ISBN 978-7-302-60942-1

Ⅰ. ①快…　Ⅱ. ①彼… ②伊… ③殷…　Ⅲ. ①人工智能－应用　Ⅳ. ①TP18

中国版本图书馆 CIP 数据核字(2022)第 091153 号

责任编辑：王　军
封面设计：孔祥峰
版式设计：思创景点
责任校对：成凤进
责任印制：曹婉颖

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：天津鑫丰华印务有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　印　　张：18.75　　字　　数：368 千字
版　　次：2022 年 7 月第 1 版　　印　　次：2022 年 7 月第 1 次印刷
定　　价：98.00 元

产品编号：090582-01

译 者 序

10 年前，当我们谈论人工智能时，会认为那是一个未来的发展方向，与我们的生活还有一段距离。而在 5 年前，我们的生活已经充斥着各种人工智能技术，从智能电话的各种美颜软件到高速公路上行驶的带有自动驾驶功能的汽车，人工智能技术几乎随处可见。即便如此，在那个时候，如果要搭建一套人工智能系统，往往也需要数天甚至数周的时间。记得当时，我在大学里讲授机器视觉和自然语言处理两门课程，在开课前，我要摸索很久来解决各种软件以及硬件的兼容性问题，并制作成一个系统镜像分发给学生。但由于学生们的硬件设备存在差异，经常导致程序报错。为了解决这个问题，我所在的学院开始与 AWS 合作，使用 AWS 的 EC2 作为计算资源。于是，我只需要创建好标准的软件环境镜像，然后分享给学生，大家便有了相同的学习环境。为什么选择 AWS？2021 年底，Gartner 发布了 IaaS+PaaS 解决方案能力评估报告，AWS 总评分排名第一，并且是唯一一家评分超过 90 的云提供商。此前，AWS 已经连续 11 年被 Gartner 评为云服务的领导者。现在是一个云计算的时代，如果学生能够在学习期间熟悉未来工作经常使用的云环境，必将在就业市场更具竞争力。

对计算机专业的学生来说，使用系统镜像通过代码的形式完成各种人工智能任务，似乎是一件不太困难的事情。但是在选课的学生名单中，我发现越来越多的非计算机专业的学生，他们也对人工智能感兴趣。这些学生有来自我们学院对面的医学院的学生，还有许多工程专业、金融专业的学生。这让我感到欣慰的同时，也有些担心，因为他们需要花更多的时间学习计算机专业学生在其他课程中已掌握的专业技能。有没有一种更好的解决方案，让这些非计算机专业的学生能够以最小的学习成本，掌握并使用先进的人工智能技术呢？答案是肯定的，AWS 将许多人工智能技术以“服务”的形式提供，只需要几行简单的脚本，甚至只需要单击几下鼠标即可完成图像识别、图像分类等人工智能工作。医学院的学生甚至实现了通过 AWS 图像识别服务诊断眼底病变。当然，在进行这项研究时，我们请了几位计算机专业的学生帮助他们完成了一些基础性工作。

很高兴曼宁出版社出版了本书。本书介绍了 AWS 提供的 AI 服务，包括机器视觉、语音识别、自然语言识别、语言翻译等，用通俗易懂的方式介绍了如何在

短时间内搭建自己的人工智能系统。例如，通过示例讲解如何让 AI 识别图中的物体、如何通过几种技术的组合搭建聊天机器人等。这些简单而有趣的案例，可以让人工智能学习之旅变得更加轻松。希望你阅读完本书后即可快速搭建自己的人工智能系统。

在这里，我想衷心感谢清华大学出版社的王军老师，感谢他帮助我出版了多本机器学习、人工智能以及高性能计算的译作，感谢他为我提供一种新的与大家分享知识的方式。另外，我还想感谢我的学妹陈嘉雯，感谢她对本书译稿的文字及代码进行修改和校对。

殷海英

埃尔赛贡多，加利福尼亚州

2022 年 4 月

序　言

在过去的 20 年里，人工智能在我们的生活中扮演了十分重要的角色。它在幕后为我们提供各种服务，世界各地的公司都在使用人工智能技术改善搜索结果、产品推荐和广告，甚至帮助医护人员提供更准确的诊断。人工智能技术无处不在，在不久的将来，我们都将搭乘自动驾驶汽车出行！

随着人工智能地位的不断上升，对相关技能的需求也随之上升。拥有机器学习或深度学习专长的工程师常常被大型科技公司高薪聘用。与此同时，我们接触到的所有应用程序都希望使用人工智能来改善其用户体验。但具有相关技能，并获取必要的数据来训练这些 AI 模型的能力仍然是进入人工智能领域的重大障碍。

幸运的是，云服务提供商正在提供越来越多的人工智能服务，不再需要你淹没在收集、清理数据和训练人工智能模型的繁重劳动中。例如，AWS 允许你通过使用 Amazon Personalize 实现与 Amazon.com 提供产品推荐的相同服务，或者通过 Amazon Transcribe 实现 Alexa 的语音识别技术。其他云提供商(GCP、Azure、IBM 等)也提供类似的服务，通过这些服务，日常应用程序中随处可见 AI 的身影。随着这些服务不断改进并更容易获得，除了更专业的工作之外，人们将不再需要训练自己的 AI 模型。

很高兴终于看到这本专注于利用这些 AI 服务而不是训练 AI 模型的书出版。本书以通俗易懂的语言解释了人工智能和机器学习中的重要概念，并准确描述了它们的含义，内容翔实。本书的美妙之处在于它不仅仅是“如何使用来自 AWS 的这些 AI 服务”，还包括如何以无服务器方式构建应用程序。它涵盖了从项目组织到持续部署的各个方面，以及有效的日志记录策略，还包括如何使用服务和应用程序指标来监控应用程序。本书后面的章节介绍了集成模式，以及如何将 AI 技术融入现有应用程序的真实示例。

无服务器是一种心态，也是一种思考软件开发的方式，它将业务及客户的需求放在首位，旨在通过利用尽可能多的托管服务，以最少的努力创造最大的业务价值。这种思维方式提高了开发人员的生产力和程序运行的速度，并且通常利用 AWS 等顶级云服务商所提供的服务，开发更具扩展性、弹性和安全性的应用程序。

无服务器是软件开发的趋势和未来，本书将帮助你开启无服务器开发之旅，并向你展示如何将 AI 服务集成到无服务器应用程序中，从而增强用户体验。可以说是一石二鸟！

崔　岩

AWS 无服务器英雄

独立顾问

前　言

第四次工业革命即将来临！未来十年，基因编辑、量子计算，当然还有人工智能(AI)等领域可能会取得巨大进展。我们大多数人每天都在与人工智能技术互动。这不仅仅是指自动驾驶汽车或自动除草机。人工智能在我们的日常生活中无处不在。例如当你访问亚马逊网站时，网站即刻给出产品推荐；你与航空公司进行的航班改签的在线聊天对话，或者银行给你发送的简讯，警告你的账户可能存在欺诈交易。所有这些例子都是由人工智能和机器学习技术驱动的。

现实将越来越多地要求开发人员在他们所构建的产品和平台上添加人工智能功能及相关接口。当然，人工智能和机器学习的早期使用者已经在这样做了。然而，这需要大量的研发投入，通常需要一个数据科学家团队来训练、测试、部署和操作定制化的人工智能模型。借由强大的商品化力量，这种情况正在迅速改变。

在 Nicholas Carr 2010 年的畅销书 *The Big Switch* 中，他将云计算比作电力，并预测我们最终将把计算资源作为一种消费能源。虽然我们还没有真正达到这种程度，但可以预示，这种消费模式正在迅速成为现实。

你可以从云原生服务的范围和能力的爆炸性增长中看到这一点。云堆栈的商品化催生了无服务器计算范式。我们相信，无服务器计算将成为未来构建软件平台和产品的标准架构。

随着应用程序堆栈商业化的广泛应用，人工智能也正在迅速成为一种商品，例如主流云服务提供商在图像识别、自然语言处理和聊天机器人等领域提供的 AI 服务，这些 AI 服务的数量和能力正逐月迅猛增长。

我们工作的 fourTheorem 公司每天都在使用这些技术，通过应用人工智能服务来帮助客户扩展和改进他们现有的系统。我们帮助客户采用无服务器架构和相关工具来加速他们的平台开发工作，并且利用我们的经验帮助重构客户的遗留系统，以便它们可以更有效地在云端运行。

正是无服务器和 AI 服务这两种技术的快速增长和商业化，以及我们将它们应用于实际项目的经验，促使我们撰写了本书。我们希望提供一份工程师的工作指南，帮助你在 AI 即服务方面取得成功，并祝你在探索这个精彩的软件开发新世界时一切顺利！

作 者 简 介

彼得·埃尔格是 fourTheorem(一家技术咨询公司、AWS 合作伙伴)的联合创始人兼 CEO。彼得在英国 JET Joint Undertaking 公司开始他的职业生涯，在那里他花了 7 年时间为核聚变研究创建采集、控制和数据分析系统。他在广泛的研究和商业软件领域中担任过技术领导职务，包括软件灾难恢复、电信和社交媒体。在创立 fourTheorem 之前，彼得是两家公司的联合创始人兼 CTO：Stitcher Ads(一个社交广告平台)和 NearForm(一家 Node.js 咨询公司)。彼得目前研究的重点是通过应用尖端的无服务器技术、云架构和机器学习为客户提供商业价值。他的经验涵盖了从构建大型分布式软件系统到领导实施这些系统的国际团队的各个方面。彼得拥有物理学和计算机科学双学位。

伊恩·沙纳吉有幸从 20 世纪 80 年代中期开始在 Sinclair ZX Spectrum 上编程。这是第一件他不想拆卸的电子产品。如今，他试图将软件系统拆开。伊恩是 fourTheorem 的联合创始人兼 CTO。他是架构师也是开发人员，在为初创公司和大型企业构建和扩展系统方面拥有丰富的经验。伊恩经历过许多不同的技术时代，从 2000 年的基于 Java 的分布式系统到近年来的全栈多语言容器和无服务器应用程序。伊恩拥有都柏林圣三一学院的计算机科学学士学位。

致　谢

咨询任何一本技术书籍的作者，他们都会告诉你完成一本书需要大量的时间和精力，也需要他人的大力支持。我们要感谢很多人，本书的完成离不开他们的帮助。

首先，我们要感谢我们的家人，感谢他们给予我们的支持、理解和耐心。Eóin 要感谢他的妻子 Keelin，感谢她无尽的耐心、精神支持和不可或缺的技术审查。他还想感谢 Aoife 和 Cormac，他们是世界上最优秀的孩子。Peter 想感谢他的女儿 Isobel 和 Katie，感谢她们带来的欢乐。

Eóin 和 Peter 要感谢 fourTheorem 的联合创始人 Fiona McKenna，感谢她对本书的信任，以及她在诸多领域的支持和专业知识。如果没有她，我们不可能完成本书。

万事开头难，我们感谢那些在开始时帮助我们的人。Johannes Ahlmann 提供了很多思路，并经常与我们讨论本书的内容，本书的成功离不开他的帮助。James Dadd 和 Robert Paulus 在本书创作中提供了宝贵的支持和反馈。

我们也要感谢曼宁出版社团队，是他们让本书顺利出版。我们特别要感谢开发编辑 Lesley Trites，感谢她的耐心和支持。还要感谢技术开发编辑 Palak Mathur 和 Al Krinker，他们的审查和反馈提供了极大的帮助。感谢项目编辑 Deirdre Hiam、文稿编辑 Ben Berg、校对 Melody Dolab 和审查编辑 Ivan Martinović，感谢他们的大力支持与帮助。

我们要感谢崔岩为本书撰写序言。他是一位杰出的架构师和无服务器技术的拥护者，感谢他对我们的支持。

非常感谢所有审阅者对文本和示例的反馈和改进建议：Alain Couniot、Alex Gascon、Andrew Hamor、Dwight Barry、Earl B. Bingham、Eros Pedrini、Greg Andress、Guillaume Alleon、Leemay Nassery、Manu Sareena、Maria Gemini、Matt Welke、Michael Jensen、Mykhaylo Rubezhanskyy、Nirupam Sharma、Philippe Vialatte、Polina Keselman、Rob Pacheco、Roger M. Meli、Sowmya Vajjala 和 Yvon Vieville。

特别感谢技术校对 Guillaume Alleon，他对代码示例进行了仔细的审查和测试。

最后，我们要感谢更广泛的开源社区，我们为能够参与其中感到自豪。这让我们能够站在巨人的肩膀上不断攀升。

关于封面插图

《快速构建 AI 应用——AWS 无服务器 AI 应用实战》封面插图的标题是“Homme de la Forêt Noire”，意思是“来自黑森林的人”。这幅插图摘自雅克·格拉斯特·德·圣索维(1757—1810)的《法国服饰》(*costume civil actuels de tous les peuples connus*)，于 1788 年在法国出版。书中收集了来自不同国家的华服，每一幅插图都是手工精细绘制和着色的。格拉斯特·德·圣索维繁多的藏品种类，鲜活地提醒我们，200 年前，世界各地的城镇和地区在文化上的差异如此巨大。人们彼此隔绝，说着不同的方言和语言。无论是在街上还是在乡村，只要看一看他们的衣着，就很容易知道他们住在哪里，从事什么行业，在社会中处于什么地位。

从那时起，我们的穿着方式发生了改变。地域的多样性，在当时是如此丰富，但现在已经逐渐消失了。现在很难区分不同地域的居民，更不用说区分不同的城镇、地区或国家了。也许我们用文化多样性换取了更多样化的个人生活——当然是为了更多样化和快节奏的科技生活。

在这个计算机书籍同质化严重的时代，曼宁出版社用两个世纪前地区生活的丰富多样性的书籍封面，来庆祝计算机行业的创造性和主动性，利用格拉斯特·德·圣索维的图片提醒我们生活中存在的多样性。

关 于 本 书

《快速构建 AI 应用——AWS 无服务器 AI 应用实战》是作为构建支持人工智能的平台和服务的指南而编写的。这本书的目的是教会你运行人工智能程序，并快速生成结果，远离困境。人工智能和机器学习是一个很大的话题，如果你想掌握这些学科，有很多知识需要学习。我们无意劝阻任何人这样做，但如果你需要快速取得成果，本书将助你一臂之力。

本书探讨了两项不断发展且日益重要的技术：无服务器计算和人工智能。我们从开发人员的角度审视这些内容，从而为你提供实用的操作指南。

所有主要的云供应商都在竞相提供相关的人工智能服务，例如：

- 图像识别
- 语音到文字及文字到语音的转换
- 聊天机器人
- 语言翻译
- 自然语言处理
- 推荐系统

上面所列的服务会随着时间的推移而不断增加。

让人兴奋的是，你不必成为 AI 或机器学习专家即可使用这些产品。本书将指导你把这些服务应用于日常开发工作中。

随着 AI 服务的发展，现在可以使用无服务器技术，以最少的运营开销构建和部署应用程序。我们相信，在未来几年内，本书中描述的工具、技术和架构将成为企业平台开发标准工具包的一部分。本书将带你快速上手，并帮助你使用无服务器架构创建新系统，并将 AI 服务应用于现有的平台。

本书的目标读者

《快速构建 AI 应用——AWS 无服务器 AI 应用实战》是为负责实施 AI 增强型平台和服务的全栈及后端开发人员编写的。本书对于希望了解如何通过 AI 增强和改进其系统的解决方案架构师及产品经理也很有帮助。DevOps 专业人员将获得

有关构建和部署系统的“无服务器方式”的有价值见解。

本书组织路线图

本书分为 3 个部分，涵盖 9 个章节。

第 I 部分提供了一些背景知识，并介绍了一个简单的无服务器 AI 系统：

- 第 1 章介绍了过去几年无服务器计算的兴起，解释了为什么无服务器代表了真正的以效用为中心的云计算。在此之后，本章提供了人工智能的简要概述，以便让没有该主题经验的读者了解必要信息。
- 第 2 章和第 3 章快速构建了一个使用现成图像识别技术的无服务器 AI 系统。读者可以部署并测试这个系统，探索如何使用图像识别技术。

第 II 部分深入探讨开发者在使用无服务器和现成 AI 时需要掌握的工具和技术：

- 第 4 章讨论了如何构建和部署一个简单的无服务器 Web 应用程序，然后，介绍更重要的如何以无服务器的方式保护应用程序。
- 第 5 章探讨了如何将人工智能驱动的接口添加到无服务器的 Web 应用程序中，包括语音到文本、文本到语音和会话聊天机器人接口。
- 第 6 章提供了关于如何成为一名使用这种新技术堆栈的开发人员的具体建议，包括项目结构、CI/CD 和监控——大多数开发人员掌握无服务器和 AI 技术时都需要用到这些组件。
- 第 7 章详细介绍无服务器 AI 如何应用于现有系统或遗留系统。在这里，提供了关于通用模式的建议，并通过一些解决方案说明这些模式的应用场景。

第III部分着眼于如何将前两部分的内容整合到一个完整的 AI 驱动系统中：

- 第 8 章通过一个无服务器的网络爬虫例子讨论如何大规模地收集数据。
- 第 9 章着眼于如何使用 AI 即服务，从无服务器网络爬虫收集的数据中提炼有价值的数据。

读者应该仔细阅读第 1 章的内容，从而了解本书研究内容的基础，并密切关注第 2 章的内容，在第 2 章描述了如何创建开发环境。建议你按照顺序阅读本书，因为每一章都通过实例展开讲解，并且内容都逐章递进。

关于代码

本书提供了许多实例的源代码，包括有编号的代码清单和正文中的代码段。在这两种情况下，源代码均以固定字体显示，以区分正文。有时，代码也用粗体

显示，以突出显示对本章前述代码的改动，例如当将一个新特性添加到现有代码行时。

在许多情况下，原始源代码已经被重新格式化，添加了换行符，并重新调整了缩进，以适应印刷版面。在少数情况下，即使这样还是不够，所以在代码清单中使用行延续标记(➥)。此外，正文中引用代码时，通常会从代码清单中删除源代码中的注释。大多代码清单中都使用代码注释来强调重要概念。本书的示例源代码可以扫描封底二维码下载。

目　录

第Ⅰ部分　基础知识

第Ⅰ部分讲述的是人工智能的基础内容，可帮助你快速了解 AI 即服务。第 1 章着重讲述人工智能和无服务器计算的历史和发展，以及当前最先进的技术，并将 AWS 上的服务按照标准架构结构进行分类。第 2 章和第 3 章深入研究并创建一个无服务器图像识别系统，作为我们的第一个 AI 即服务平台。

第 1 章

两种技术

本章主要内容：

- 云端环境
- 什么是无服务器
- 什么是人工智能
- 摩尔定律的普及
- 标准的 AI 即服务架构
- Amazon Web Services 上的规范架构

欢迎阅读本书！本章从工程视角探索两种爆炸性发展的技术：无服务器计算和人工智能。所谓工程视角是指本书将为你提供一个实际操作指南，让你在无须学习大量理论内容的情况下，将人工智能作为一种服务应用到工作中。

我们假设，你像大多数人一样听说过这些主题，并好奇为什么我们把这两个看似不相关的主题合并到一本书中。我们将在接下来的章节中看到，这些技术的结合有可能成为企业级开发的现实标准。这一组合将为软件开发人员以及他们所服务的企业提供巨大的力量，以增强和改进现有系统，并快速开发和部署新的人工智能平台。

世界正变得越来越数字化——你可能听说过“数字化转型”这个词。这通常是指将当前使用的电子表格、本地数据库甚至根本没有软件运行的现有人工业务流程，转换为在云端平台运行的过程。无服务器(serverless)为我们提供了一个工具链来加速数字化转型，并且越来越多的 AI 技术成为这些转型的核心部分，并用计算机取代所有或部分这些人工操作的业务流程。

要求软件开发人员实现这些平台的声音日益增多，大多数从事软件行业的人都需要熟练地设计、开发和维护这些类型的系统。

你现在是不是在想，“我对人工智能一无所知！这听起来真的很难，我是否需要成为人工智能专家？”别着急，你不必成为数据科学家或机器学习专家即可构建无服务器 AI 系统。正如你将在本书中看到的，大部分艰苦的工作已经以“现成的”云端 AI 服务的形式完成了。软件开发人员只需要使用这些组件设计解决方案即可。

让我们用一个简单的例子说明这个概念。假设有一家连锁酒店。要想成功经营酒店并盈利，需要完成很多的工作。例如，客房定价。如果定价太高，没有人会预订；如果定价太低，公司会有损失。负责该项工作的员工需要依据经验来核定房价，并综合考虑同业竞争、全年销售时长、天气以及当地可能举办的活动等因素。定价后，这些价格将在广告中发布，但会随着当地条件的变化和房间预订的状况不断调整。

这个过程非常适合 AI 即服务的平台，因为它是一个不断优化的问题。利用云原生服务，我们可以通过快速开发服务来获取和存储适当的数据：通过 API 访问或从网站上获取有关当地事件的信息。可以使用现成的 AI 模型来解释抓取的数据，并且可以交叉训练现有的神经网络，来预测最佳房价。房价可以通过另一个服务自动发布。现在，仅通过连接云原生 AI 服务和数据服务，就可以利用非常有限的 AI 知识来实现这一点。

如果你的主要兴趣是开发简单的网站或低级通信协议，那么 AI 即服务可能不会引起你的关注。然而，对于绝大多数软件专业人士来说，AI 即服务将对你的职业生涯产生重大影响，而且这种影响将很快发生。

1.1 云端环境

从事软件行业的人都至少对云计算有一个基本的了解。云计算最初是在别人的硬件上运行虚拟服务器的一种机制，这通常被称为基础设施即服务(Infrastructure as a Service，IaaS)。它现在已经发展成为一套更丰富的按需服务，可以满足各种计算负载。目前主要的云服务商有 3 家：亚马逊、谷歌和微软。AWS(Amazon Web Services)一直是云端基础设施的最主要提供商，AWS 提供了一系列功能极其丰富的产品。

截至 2020 年 3 月，三大平台提供的服务范围非常相似。表 1-1 列出了来自亚马逊、谷歌和微软的常用服务的数量[1]。

1 信息来源：https://aws.amazon.com/products/、https://cloud.google.com/products/以及 https://azure.microsoft.com/en-us/services/。

表 1-1　截至 2020 年 3 月，三大云服务商的云服务数量

服务类型	AWS	Google	Azure
AI 和机器学习	24	20	42
计算	10	7	20
容器	4	8	10
开发	12	16	11
数据库	12	6	12
存储	10	6	17
IoT(物联网)	12	2	22
网络	11	11	20
安全	18	28	10
其他	85	119	115
总计	198	223	279

需要了解的云服务种类繁多，每种服务都有自己特定的 API。我们永远都无法了解所有这些服务的细节，那么如何才能更好地理解这一切，并成为出色的工程师呢？随着新服务的不断添加和更新，这种情况也在不断变化。

我们的目标应该是理解架构原则，以及如何从这些服务组合中实现特定的业务目标；持续关注这些服务类型，并深入研究一个子集，以便可以根据需要(想实现的结果)快速启用一个全新的服务。

图 1-1 展示的是 AI 即服务平台的抽象框架。

该模型建立在对 4 大支柱的理解之上。

- 体系结构：采用无服务器计算的有效架构模式是什么？
- 开发：最好的开发工具、框架和技术是什么？
- AI：可用的机器学习和数据处理服务有哪些？如何更好地使用它们解决业务问题？
- 运维：我们如何有效地将这些服务投入生产并管理它们的运行状态？

本书将通过构建一个包含机器学习服务(如聊天机器人和语音转文字)的示例系统来探索人工智能的每个应用；将探索高效的无服务器开发框架和工具，并提供关于如何在无服务器环境中有效调试的帮助和建议。本书还将介绍如何将 AI 工具和技术应用于平台操作，以及如何保护一个无服务器的平台。

我们可以看到现有的软件架构经验如何转移到无服务器领域，并为 AI 即服务平台开发一个规范的架构，这将帮助我们将每种云服务应用到工作中。全书都会使用这个体系结构，并将它作为示例系统的参考模型。

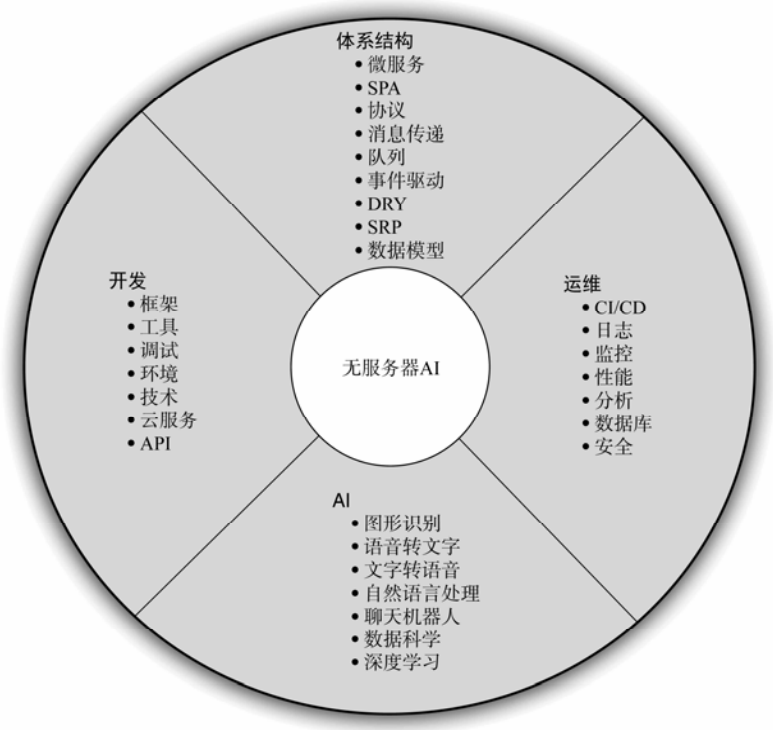

图 1-1 高效的 AI 即服务工程

接下来将探索无服务器和人工智能的发展，并略述每个主题的简史。这个重要的背景将帮助我们理解这些技术的发展，以及人工智能和云计算这两个看似复杂的领域如何发展到今天。本章集中讲解大部分的理论知识，从第 2 章开始进入代码实操环节。

1.2 什么是无服务器

鉴于术语“无服务器”没有正式的定义，我们将它描述为一个解决方案。

> 无服务器计算是云端应用计算的一种形式，其中云服务供应商为用户动态管理底层资源。它提供了底层基础设施之上的抽象级别，消除了终端用户的管理负担。

无服务器软件是云端软件的一种形式，它避免了基础设施(例如服务器或容器)的显式创建和管理。服务器和容器这些传统的计算资源由云服务提供商进行管理和运维。这就是所谓的功能即服务(Functions-as-a-Service，FaaS)。无服务器应用程序不必创建大量的专用资源，如数据库、文件存储或消息队列。它们依赖云提供商提供的管理服务，这些服务可以自动扩展，从而处理大量的工作负载。无服

务器应用程序的收费模式也很友好。与无论资源是在使用还是空闲都为其付费不同，云服务提供商通常只在调用相关功能和使用托管服务时收费。这可以节省大量成本，并确保基础设施成本与具体使用量保持一致。

无服务器计算的原理概括如下：

- 服务器和容器被按需执行的云函数所取代。
- 托管服务和第三方 API 优先于定制资源。
- 架构主要是事件驱动和分布式的。
- 开发人员专注于构建核心产品，而不是底层基础设施。

术语“无服务器”可能有点言过其实，因为这个服务链中必须有一台服务器。该术语是想强调，使用无服务器技术时，作为用户，我们不再需要关心底层的基础设施。云服务供应商通过 FaaS 和其他托管服务，在底层基础设施上提供了抽象级别。

从某种意义上说，计算的历史与网络抽象化程度息息相关。在操作系统抽象化之前，早期用户只能寄希望于物理磁盘扇区和寄存器。伴随着一系列越来越复杂的抽象化，编程语言已经从低级的语言(如汇编语言)发展到动态的现代编程语言(如 Python)，无服务器也诞生了。

任何从事软件开发的人，无论是开发人员、DevOps 专家、产品经理还是高级技术人员，都知道我们行业的变化速度不同于其他任何行业。想象一下，其他职业，例如医生、牙医、律师或土木工程师，必须以软件行业那样疯狂的速度更新他们的知识库，这简直让人无法想象。

技术的不断更新是一把双刃剑，许多人都喜欢使用最新的、最好的技术堆栈，但又经常面临选择悖论，因为可供选择的语言、平台和技术种类繁多，层出不穷。

选择的悖论

《选择的悖论：用心理学解读人的经济行为》(*The Paradox of Choice Why More is Less*)是心理学家巴里·施瓦茨(Barry Schwartz)所写的一本书，他在书中提出的论点是，更多的选择实际上会导致消费者焦虑。他认为，一个成功的产品应该将选择的数量限制在几个不同的类别中。这与编程语言、框架和平台的情况类似，我们确实有太多的可选方案。

许多在这个行业工作了一段时间的人都对最新的技术趋势或框架持怀疑态度。然而，我们相信无服务器和当前的 AI 浪潮代表了一个真正的范式转变，而不仅仅是一个短期趋势。

1.3　对速度的需求

计算机工业的历史和发展是一个引人入胜的话题，许多书籍都介绍了这个主

题。虽然我们不能在这里深入地讨论这个话题，但了解一些关键的历史趋势及其背后的力量很重要。这将帮助我们看到，无服务器实际上是这段历史的下一个发展阶段。

1.3.1 早期情况

计算的历史可以追溯到古代，当时有算盘之类的工具。历史学家认为，第一个已知的计算机算法是由 Ada Lovelace 在 19 世纪为 Charles Babbage 实现的。计算的早期发展关注的是单一目的、笨拙的系统，它被设计用来实现单一目标。随着 1964 年第一个多任务操作系统 MULTICS 的研发，现代软件时代才真正开始，其后是 UNIX 操作系统的研发。

1.3.2 UNIX 哲学

UNIX 操作系统是 20 世纪 70 年代由肯·汤普森和丹尼斯·里奇在贝尔实验室发明的。最初的 AT&T 版本衍生了许多衍生产品，最著名的也许是 Linux 内核和相关的发行版。对计算历史感兴趣的读者可以好好看看，图 1-2 展示的是 UNIX 族谱。正如图中所绘的，最初的系统产生了许多成功的派生系统，包括 Linux、Mac OS X 和 BSD 系列操作系统。

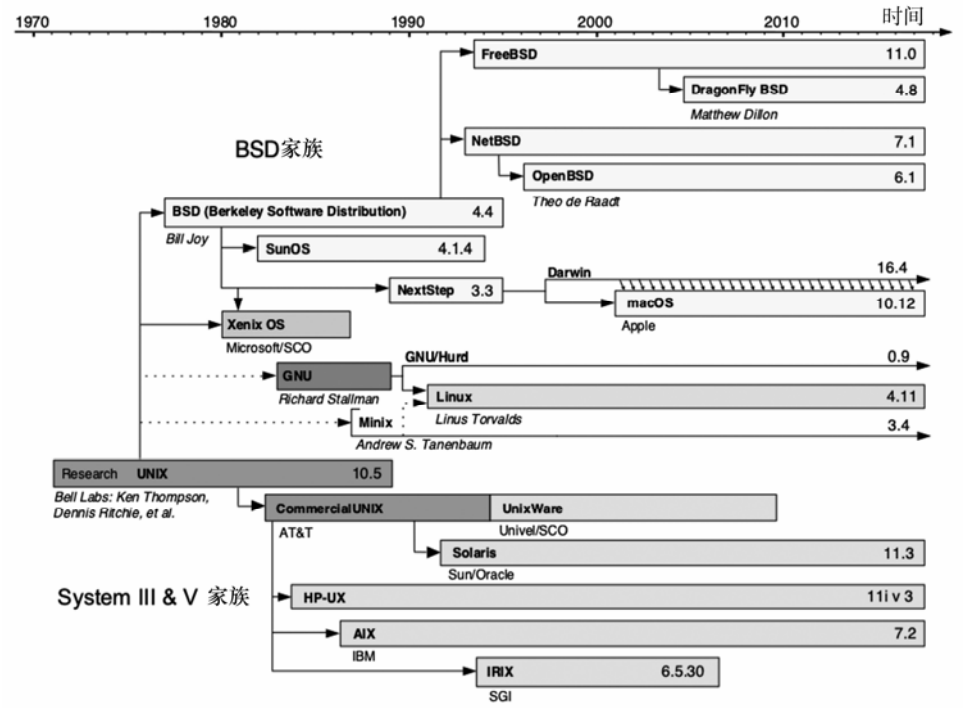

图 1-2 UNIX 族谱。来源：http://mng.bz/6AGR

也许比操作系统更重要的是围绕原始 UNIX 技术发展起来的相关理论，概括如下：

- 编写只处理一件事、并把事情做好的程序。
- 编写可以一起工作的程序。
- 编写处理文本流的程序——通用接口。

提示 关于这个主题的完整解释，请参阅 Brian Kernigan 和 Rob Pike 编著的 *The UNIX Programming Environment* (Prentice-Hall，1983)。

这种系统设计方法首次将模块化的概念引入软件开发中。在此，我们应该注意到，这些原则可以应用于任何底层操作系统或语言。例如，在 Windows 编程环境中使用 C#就可以完全有效地应用 UNIX 理论。

这里要理解的关键是单一职责原则——编写具有单一焦点的程序或模块。

单一焦点

为了说明程序单一焦点的概念，请考虑以下 UNIX 命令行工具：

- ls 知道如何列出目录中的文件。
- find 在目录树中搜索文件。
- grep 知道如何在文本中搜索字符串。
- wc 知道如何计算文本中的行数或单词数。
- netstat 列出打开的网络连接。
- sort>按数字或字母顺序排序。
- head 返回输入内容的前 *n* 行。

上面每个单独工具都相当简单，但可以将它们组合在一起，来完成更复杂的任务。例如，以下代码给出了系统上侦听 TCP 套接字的数量：

```
$ netstat -an | grep -i listen | grep -i tcp | wc -l
```

下例显示目录树中的最大的 5 个文件：

```
find . -type f -exec ls -s {} \; | sort -n -r | head -5
```

在软件领域，把一件事做好的哲学是一股强大的力量。它推动着我们编写更小的代码单元。相比那些大型的相互连接的代码单元，小的代码单元更容易理解，也更容易得到正确的结果。

1.3.3　面向对象和模式

这个最初的简捷的、模块化的方法在很大程度上已经被业界遗忘，人们转而使用支持面向对象的范式。在 20 世纪 80 年代末和 90 年代初，C++等语言越来越

受欢迎。在软件模式运动的推动下，面向对象的范式承诺：代码可以通过继承和多态等机制在对象级别重用。事实证明，这一愿景从未实现，正如著名的香蕉、猴子、丛林问题中讽刺的一样。

香蕉、猴子和丛林问题

香蕉、猴子、丛林问题是指现实世界中面向对象代码库的重用问题。它可以这样理解：我想要一根香蕉，但当我伸手去拿时，却发现一只猴子也抓住了这根香蕉。不仅如此，这只猴子还抓住了一棵树，所以我把整个丛林都抓住了。

下面的代码片段说的正是这个问题：

```
public class Banana {
    public Monkey Owner {get;}
}
public class Monkey {
    public Jungle Habitat {get;}
}

public class Jungle {
}
```

为了使用 Banana 类，首先需要为它提供一个 Monkey 实例。为了使用 Monkey 类，则需要为它提供一个 Jungle 实例，等等。这种耦合是作者在大多数面向对象代码基中遇到的一个真实的问题。

在这个时期，Web 出现之前，系统倾向于作为单一的模块来开发和构建。对于大型系统来说，由超过一百万行代码组成一个单一的可执行文件是很正常的。

1.3.4 Java、J2EE 和.NET

面向对象的流行趋势从 20 世纪 90 年代一直持续到 21 世纪，在此期间，Java 和 C#等语言风靡一时。然而，这些系统的性质开始从桌面交付转向分布式、网络感知的应用程序。这一时期见证了应用服务器模型的兴起，其特点是大型的单个代码库、大量具有海量存储过程的关系数据库以及用于分布式通信和互操作性的 CORBA/COM。部署操作通常需要每 3～6 个月进行一次，需要几周的计划和系统停机窗口。

CORBA 和 COM

公共对象请求代理体系结构(Common Object Request Broker Architecture)是一种遗留的二进制通信协议，它在 21 世纪早期非常流行。公共对象模型(Common Object Model，COM)是特定于微软的 CORBA 替代方案。目前，这两种技术大部分都被 RESTful API 取代了，见图 1-3。

图 1-3　2000 年左右的企业软件开发(*Ploughing with Oxen*，George H. Harvey，1881)。来源：http://mng.bz/oRVD

仔细想想，21 世纪早期的软件研发可以比作农业的早期。在当时，它是革命性的。但与后来的情况相比，它是缓慢的，笨拙的，不灵活的，劳动密集的。

1.3.5　XML 和SOAXML 以及 SOA

从那时起，业界开始采用 XML(可扩展标记语言)作为配置和通信的一切手段，随着 SOAP 和所谓的 SOA(面向服务的架构)的出现，它的发展达到顶峰。这是由对解耦和互操作性的渴望所驱动的，并以对开放标准收益的初步理解为基础。

SOAP

SOAP(简单对象访问协议)是一种基于 XML 的文本协议，被吹捧为 CORBA 和 COM 的高级替代方案。由于其基于文本的特性，SOAP 具有比 CORBA 或 COM 更好的跨平台互操作特性。然而，与现代基于 JSON 的 RESTful API 相比，它的体量仍然较大，且使用起来非常麻烦。

1.3.6　Web speed

在互联网繁荣(以及随后的崩溃)的推动下，企业软件开发发生了变化，软件即服务(SaaS)模型开始受到关注。该行业正在将 Web 转变为主要的应用程序交付机制，最初用于面向外部客户的使用，后来越来越多地用于内部企业交付。在此期间，对快速交付软件的需求不断增加，包括初始部署需求，以及可以立即部署

到服务器上的功能添加需求。此时，主要的 SaaS 托管模型被部署到位于数据中心内的本地服务器上。

因此，该行业存在两个主要问题。首先，需要提前预测所需的容量，以便可以分配足够的资金来购买所需的硬件，从而处理预期的负载。其次，大型的、单一的、面向对象的代码库并不适合 Web-speed 开发模型。

很明显，体量大且封闭的企业模型不适合 Web-speed 交付。这导致人们越来越多地采用基于开放标准的方法，并更多地使用开源技术。这一运动由 FSF(自由软件基金会)、Apache 和 GNU/Linux 等组织领导。

向开源的转变导致企业软件架构定义方式发生了一个关键的、不可逆转的变化。最佳实践、标准和工具曾经是由 Sun Microsystems、Oracle 和 Microsoft 等企业领导者的利益决定的。而使用开源，业余爱好者、初创公司和学者能够快速创新，并以前所未有的频率进行分享和迭代。以往，行业需要等待主要参与者就复杂的标准文档达成一致，而现在模式转变为利用社区的联合力量和敏捷性来证明有效的、务实的解决方案，这些解决方案不仅即刻生效，而且还在以惊人的速度不断改进。

1.3.7 云计算

2006 年，云计算首次崭露头角，亚马逊推出了弹性计算云产品，现在被称为 Amazon EC2。紧随其后的是谷歌的 App Engine，以及 2010 年的微软 Azure。说云计算已经从根本上改变了软件行业，这并不为过。2017 年，AWS 报告收入 174.6 亿美元。

按需提供算力使得个人和资金短缺的初创企业能够负担得起算力费用，并快速构建真正的创新项目，这对行业产生巨大的影响。更重要的是，行业开始转向寻找最具创新性的软件工具终端用户，而不再是企业软件供应商。对于企业来说，云计算的兴起引发了几次重大变革。其中的关键是：

- 成本模式的转变——从大量的前期资本支出转向较低的持续运营支出。
- 弹性扩展——资源可按需使用和付费。
- DevOps 和基础设施即代码——随着云 API 的成熟，可以通过工具将整个部署堆栈作为代码来实现。

1.3.8 微服务(重新认识)

开源技术的广泛应用，加上向运营支出和弹性扩展的转变，导致在企业平台开发方面重新审视 UNIX 哲学，并有助于推动所谓的微服务架构的应用。虽然微服务还没有正式的定义，但业内大多数从业者都认同它具有以下特征：

- 微服务体积小、粒度细，并执行单一功能。

- 组织文化必须主动接受测试和部署的自动化。这减轻了管理和运营的负担，并允许不同的开发团队处理可独立部署的代码单元。
- 组织文化和设计原则必须包容失败和错误，类似于反脆弱系统。
- 每个服务都是有弹性的、可组合的、最小的和完整的。
- 服务可以是单独的，也可以是水平扩展的。

微服务背后的理念并不新鲜。分布式系统自 20 世纪 70 年代就存在了。Erlang 在 20 世纪 80 年代就开始做微服务，从 CORBA 到 SOA，所有的一切都试图实现分布式、网络化组件的目标。促使微服务大规模应用的因素有云、容器和社区。

- 像 AWS 这样的云基础设施服务，可帮助快速、廉价地部署和销毁具有高可用性的安全机器集群。
- 容器(Docker)可帮助构建、打包和部署包含软件的固化单元。以前，将几百行代码部署为一个单元是不可行的或不可理解的。
- 在处理大量的、小的部署单元时，可使用社区产品提供的用来管理新型复杂性的工具。这些工具包括 Kubernetes 形式的编排，ELK(Elasticsearch、Logstash 和 Kibana)或 Zipkin 形式的监控，以及大量的工具，例如 Netflix 工程团队开源的工作。

微服务模型非常适合现代云基础架构，因为每个组件都可以单独扩展。此外，每个组件也可以单独部署。这使开发周期更短，并且确实导致了后来被称为“持续部署的开发模式”的产生，开发人员提交的代码在没有人为干预的情况下立即交付到生产中——当然，前提是它通过了一系列严格的自动化测试。

微服务的完整解决方案可参阅 Richard Rodger 编著的 *The Tao of Microservices*，该书由曼宁出版社出版。

1.3.9　云原生服务

诸如 Amazon 的 EC2 的服务通常被称为“基础设施即服务(IaaS)”。虽然这是一个强大的概念，但其运营和管理的负担仍旧在最终用户端。大多数系统需要特定形式的数据库和基础设施才能运行。如果在 IaaS 之上构建系统，则需要安装和维护一个数据库服务器集群，并处理诸如“备份、地理位置冗余，以及扩展集群以处理必要负载”等问题。使用云原生服务即可避免所有这些开销。在这种模式下，云提供商为我们处理数据库的管理和操作，只需要通过配置或使用 API 告诉系统该做什么即可。

具体的示例参看亚马逊的 DynamoDB 服务。这是一个完全托管的大规模键值存储。要使用 DynamoDB，只需要进入 AWS 控制台的 DynamoDB 设置页面，输入几条配置信息，不到一分钟就会获得一个可以读写的表。而在 EC2 实例上自行安装键值存储并设置，往往需要数小时的时间。

云服务最令人兴奋的发展之一是能够在云上运行托管代码单元，而无须关心底层服务器。这通常称为“功能即服务(FaaS)”。在 AWS 上，FaaS 是使用 Lambda 服务实现的，而 Google 的对应产品称为 Cloud Functions。

1.3.10 发展趋势：速度

究其原因，不难发现，其背后主要的驱动力就是对速度的需求。毕竟时间就是金钱——尽快编写、使用代码，并快速管理和扩展。这推动了微服务和云原生服务的广泛应用，因为这些技术提供了快速开发和部署功能的途径。

随着技术领域的不断变化，业界中的软件开发方法论也在不断发展。图 1-4 概括了这些趋势。

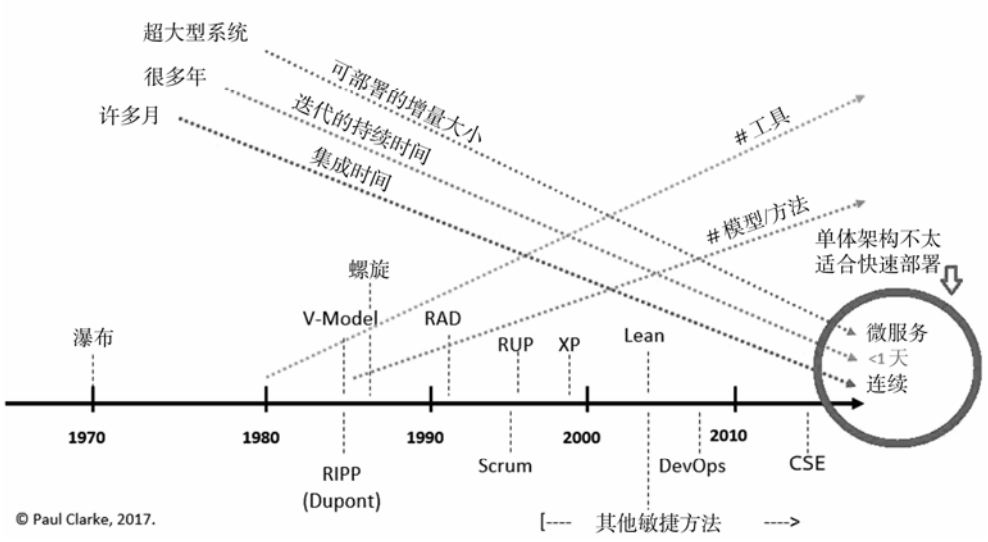

图 1-4 迭代时间和代码量的变化。Paul Clarke 2017 年绘制。计算机科学课堂笔记，都柏林城市大学和 Lero(爱尔兰软件研究中心)

如前所述，迭代时间正在迅速变短。20 世纪 80 年代和 90 年代初，使用基于瀑布的方法论，其迭代时长是整个项目的长度——对于大多数项目来说可能是一年或更长时间。进入 20 世纪 90 年代中期，迭代时间随着早期类似敏捷方法(例如 Rational Unified Process，RUP)的采用而下降。21 世纪极限编程(XP)等敏捷方法的出现，使得迭代时间缩短为 2～3 周的时间，现在一些更快速的敏捷实践则只需大约一周的迭代时间。

软件迭代周期已从 20 世纪八九十年代的一年多缩短到今天更短的发布周期。更有甚者，使用连续部署技术的最高效的组织可以一天多次将软件发布到生产环境中。

这种疯狂的发布速度是由另一个趋势实现的：即规模单位的减少。每个部署单元的代码量都在持续减少。20 世纪 80 年代和 90 年代，大型的单个代码库是当时的潮流。由于这些代码库的耦合特性，测试和部署是一个困难且耗时的过程。随着 20 世纪 90 年代末和 21 世纪初面向服务的体系结构的出现，部署单元的规模不断下降。随后，随着微服务的兴起，下降更显著。

图 1-5 说明了规模单位的减少和部署周期的缩减是如何伴随着远离底层硬件的抽象级别的增加而改变的。

图 1-5　规模单位的变化

在很大程度上，业界已经从通过使用虚拟服务器和安装、运行物理硬件，转向了基于容器的部署。目前常见的技术是将构建为容器的小型服务部署到某种编排平台(如 Kubernetes)，并使用 IaaS 配置的数据库或云原生数据服务。然而可以清楚地看到，如果加快部署速度和减少规模单位的趋势持续，(经济上的激励表明这是一个理想的目标)，那么这个过程的下一个阶段是转向完全无服务器的系统。

图 1-6 说明了引领行业走向无服务器开发的路径和技术里程碑。

总之，对软件的快速开发和部署的需求导致了规模单位的减少。这个进程的下一个合乎逻辑的阶段是完全采用无服务器架构。

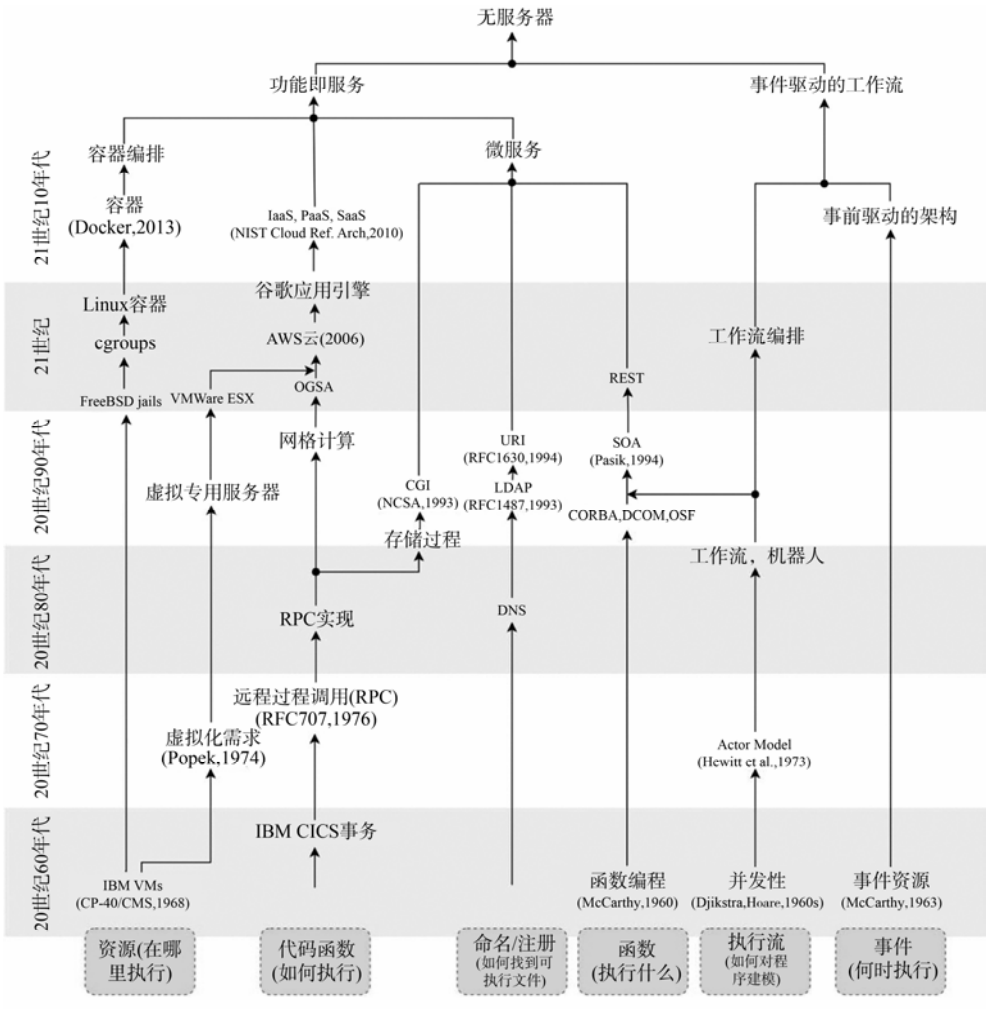

图 1-6 无服务器计算发展史

1.4 什么是 AI

人工智能(AI)是一个涵盖了计算机科学中一系列技术和算法的术语。对许多人来说，它经常会让人联想到失控的杀人机器人的形象，而这主要是源于《黑客帝国》和《终结者》系列电影中描绘的反乌托邦未来！

对这个词更清醒、更明智的定义可能是：

人工智能指的是计算机表现出的类似人类和其他动物的学习、决策能力。

它只是代码

虽然现代人工智能系统所展示的一些能力经常显得不可思议，但我们应该始终牢记，说到底，它只是代码。例如，可以识别图像的算法可能是一个非常强大的工具，但其基本面只是一个相互连接的非常简单的单元集。AI 算法中的“学习”过程实际上是基于训练数据调整数值的非常简单的问题。正是这些数值的不断涌现，才使“学习”产生了显著的结果。

1.4.1　AI 的历史

为什么突然对人工智能感兴趣？为什么人工智能和机器学习的需求越来越大？人工智能的发展是一个引人入胜的话题，它本身就可以写几本书，对它完整的讨论超出了本书的范围，在此不再赘述。

人类一直沉迷于创造自己的人工复制品，但直到 17 世纪，莱布尼茨和笛卡儿等哲学家才开始探索用系统的、数学的方式描述人类思想的概念之路——也许这更易于被非人类的机器复制。

从这样的第一次哲学尝试开始，直到 20 世纪早期，Russel 和 Boole 等思想家才对这些思想提出更正式的定义。这些发展，加上数学家库尔特・哥德尔的杰出工作，促成了艾伦・图灵的基础工作。图灵的关键见解是，任何可以在哥德尔不完备定理范围内正式定义的数学问题，理论上都可以通过计算设备，即所谓的图灵机解决。

图灵和弗劳尔斯的开创性工作，在英国布莱切利公园开发了 Bombe 和 Colossus 系统，最终促成了 1956 年夏天著名的达特茅斯学院会议，这被普遍认为是人工智能学科的正式创立。

早期有很多乐观情绪，导致了一些疯狂的乐观预测，比如：

- H.A. Simon 和 A. Newell，1958 年，“10 年内，数字计算机将成为世界国际象棋冠军。”
- H.A. Simon，1965 年，“在 20 年内，机器将能够完成人类可以完成的任何工作。”
- M. Minsky，1967 年，“在一代人之内，创造人工智能的问题将得到根本性的解决。”

在 20 世纪 70 年代早期，该领域的进展均未达到预期。随着时间的推移，人们未能取得实质性进展，由于资金来源枯竭，该领域的研究进展缓慢。这段时间被称为第一个人工智能冬季。

20 世纪 80 年代专家系统兴起：Prolog 等基于规则的问题解决语言引起了商界的关注和极大的兴趣。历史重演，到 20 世纪 80 年代末，很明显，早期专家系统的承诺也没有实现。这一事实，再加上商用 PC 硬件的兴起，意味着公司将不

再对这些系统所需的昂贵定制硬件进行投资，第二个人工智能冬天开始了。

在这个背景下，研究人员在神经网络领域取得了长足的进展，包括网络结构和改进的训练算法，比如反向传播。这个领域缺乏一个关键因素：计算能力。

在整个 20 世纪 90 年代和 21 世纪初，摩尔定律(计算能力的指数增长)持续快速发展。这种能力的增长使研究人员能够构建越来越复杂的神经网络，缩短训练周期，促进该领域的发展，并以更快的速度进行创新。从 1997 年 IBM 的“深蓝”击败加里 • 卡斯帕罗夫开始，人工智能已经扩展到许多领域，并迅速商业化。这意味着该技术现在可以在许多商业环境中以极低的成本应用，而不需要一个专家研究团队。

1.4.2 真实的 AI 世界

自 20 世纪 50 年代以来，人们一直在致力于创造能够展示人类能力的机器，这些能力指的是制定目标并找出实现它的方法。在过去几年中，现实世界中涌现出了大量的 AI 解决方案并应用于日常生活中。最新的电视剧、音乐、网上购物以及最新的新闻资讯，都有可能是人工智能提供的服务。让我们看看人工智能技术产生重大影响的一些领域。

零售和电子商务

在网上零售店和实体零售店中，人工智能被用于向购物者推荐最有可能购买的产品。在电子商务的早期，我们看到了推荐程序的简单例子(“买了这个商品的人也买了……”)。如今，许多零售商都在密切监控用户浏览行为，并使用这些数据和实时 AI 算法向用户突出展示那些他们更有可能购买的产品。

娱乐

在线电影、电视和音乐消费的显著增长给供应商提供了大量的消费数据。所有主要提供商都在使用这些数据进一步推动消费。Netflix 表示，80%的用户选择该平台算法推荐的影片。Spotify 是另一个从用户行为中学习，并提供音乐推荐的流媒体平台[1]。

新闻和媒体

人工智能在社交媒体和在线新闻中得到了广泛的使用。Facebook 和 Twitter 都使用了大量人工智能技术，来选择出现在用户时间轴上的帖子。大约 2/3 的美国成年人通过社交媒体网站获取新闻，因此人工智能对我们看到的新闻产生了重大影响(来源：http://www.journalism.org/2018/09/10/newsuse-across-social-media-platforms-2018/)。

1 信息来源：http://mng.bz/4BvQ 和 http://mng.bz/Qxg4。

广告

广告是一个深受人工智能影响的领域。人工智能用于根据用户的在线行为和偏好进行广告的精准投放。广告发布者通过移动端和网络争夺消费者注意力的过程是实时的，由人工智能实现自动化。谷歌和 Facebook 都拥有大型人工智能研究部门，在这一过程中广泛使用人工智能技术。2017 年，Facebook 和谷歌占据了新广告中 90%的业务(来源：https://www.marketingaiinstitute.com/blog/ai-in-advertising-whatyou-need-to-know)。

客户服务

随着网络的发展，消费者与企业互动的方式也在改变。许多人都习惯了自动电话应答系统，通过在键盘上选择数字或者使用语音识别来查找合适的服务。客户支持服务现在正在使用各种先进技术降低成本并改善客户体验。例如使用情感分析来检测语气并确定某些交互的重要性，或者使用聊天机器人回答常见的问题，而不需要任何人工参与。

随着这些系统的能力不断提升，语音识别和语音合成在这些场景中也发挥了重要的作用。2018 年的 Google Duplex 为大家很好地展示了如何使用语音技术服务民众(http://mng.bz/v9Oa)。每天都有越来越多的人使用 Alexa、Siri 或 Google Assistant 作为数字助理，来获取信息、安排生活和购物。

数据与安全

企业、消费者和监管机构越来越意识到数据隐私和安全的重要性——这在围绕如何存储、保留和处理数据的法规中可以看出。此外，安全漏洞越来越引起人们的关注。人工智能在解决这些问题上都可以发挥作用。个人数据的处理、分类和识别已经在 Amazon Macie 等服务中实现并应用。在威胁和漏洞检测领域，人工智能被用于预防和告警。Amazon Guard Duty 就是一个很好的例子。

除了信息安全之外，人工智能还在物理安全领域发挥着重要的作用。最近在图像处理和面部识别方面的重大改进，使人工智能技术被广泛应用于城市、建筑和机场的安全检测。人工智能还可以有效地应用于检测来自爆炸物、枪支和其他武器的威胁。

金融

通常，金融应用程序中的数据是时间序列数据。例如一个包含特定年份每天产品销售数量的数据集。可以利用这些时间序列数据来训练 AI 模型，并利用模型进行预测和资源规划。

医疗保健

人工智能在医疗保健领域的发展主要是人工智能辅助诊断，尤其是放射学和微生物学领域的图像解释。最近对该领域深度学习研究的一项调查表明，近年来

对该领域的兴趣激增，模型性能得到显著提高。虽然一些模型和产品声称其表现优于医学专家，但人们更希望将人工智能技术用作检测和测量细微异常的工具。

在许多发展中国家，医疗专业知识短缺，这使得人工智能的应用变得更有价值。例如，结核病的检测可通过人工智能自动解读胸部 X 射线影像，见图 1-7。

图 1-7 发展中国家使用人工智能协助移动 X 光机诊断结核病(本图已获代尔夫特成像系统的授权)

1.4.3 AI 服务

表 1-2 列举了人工智能的一些常见应用，AWS(和其他云提供商)为许多同类案例提供了基于预训练模型的服务。

表 1-2 人工智能应用和服务

应用程序	用例	服务
自然语言处理	机器翻译	AWS Translate
文档分析		AWS Textract
关键词		AWS Comprehend
情绪分析		
主题建模		
文档分类		
实体提取		
会话界面	聊天机器人	AWS Lex
语音	语音转文字	AWS Transcribe
文字转语音		AWS Polly
机器视觉	对象、场景和活动检测	AWS Rekognition
面部识别		
面部分析		
图片中的文字		
其他	时间序列预测	AWS Forecast
实时个性化推荐		AWS Personalize

我们将在后面的章节中使用其中的大部分服务，因此你将会对它们有比较深刻的理解。在这里，我们先对它们做一个简要的介绍。

- AWS Translate 是一个神经机器翻译服务。这意味着它使用深度学习模型，可以提供比传统的基于统计和规则的翻译算法更准确、更自然的翻译结果。
- AWS Textract 使用光学字符识别(OCR)和文本分类模型的组合，从扫描的文档中自动提取文字和数据。
- AWS Comprehend 是一种自然语言处理(NLP)服务，它使用机器学习查找文本中的见解和关系。
- AWS Lex 是一种用于构建语音和文本对话界面的服务，也称为聊天机器人。它通过使用深度学习模型进行自然语言理解(NLU)和自动语音识别(ASR)来实现这一服务。
- AWS Transcribe 使用深度学习模型将语音从音频文件转换为文字。
- AWS Polly 使用高级深度学习模型将文字转换为逼真的语音。
- AWS Rekognition 是一种图像识别服务，它使用深度学习模型识别图像和视频中的对象、人物、文字、场景和活动，并可以检测任何不当内容。
- AWS Forecast 基于与 Amazon.com 使用的相同技术。它使用机器学习将时间序列数据与附加变量相结合来构建预测。
- AWS Personalize 提供个性化的产品和内容推荐。它基于 Amazon.com 上正在使用的推荐技术。

1.4.4　人工智能和机器学习

在过去 10 年中，人工智能的大部分重点和进步都集中在机器学习领域，即“通过经验自动改进计算机算法的研究”(Tom Mitchell, *Machine Learning*, McGraw Hill, 1991)。关于人工智能和机器学习的具体含义以及它们的细微差别存在一些争论。在本书中，当我们谈论人工智能在软件系统中的应用时，主要指机器学习。

机器学习的过程通常包括训练阶段和测试阶段。不管算法是什么，机器学习算法都是基于一组数据进行训练的。对于图像识别算法，这可能是一组图像；对于金融预测模型，这可能是一组结构化记录。这些算法的目的是根据它从训练数据中“学到”的特征对测试数据做出判断。

机器学习可以分为如下类别，如图 1-8 所示。

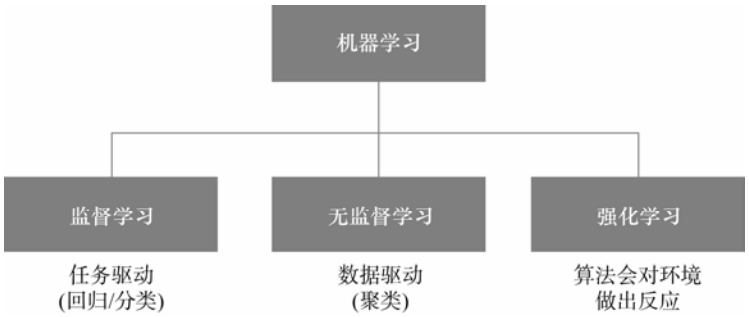

图 1-8 机器学习的类型(来源：Analytics Vidhya)

注意 特征是机器学习中的一个重要概念。为了在图像中识别猫，可能要寻找三角形的耳朵、胡须和尾巴等特征。选择正确的特征集对算法的表现至关重要。

在传统的机器学习算法中，特征由人工指定。在神经网络中，特征则由网络自动选择。

机器学习可以分为以下几类：

- 监督学习
- 无监督学习
- 强化学习

监督学习

监督学习是指在算法中提供一组带标签的训练数据。例如，一组文档用它们的分类进行标记。这些标签可能代表每个文档的主题。通过使用该组数据训练给定的算法，有望通过算法来预测未标记的测试文档中的主题。如果有足够的、标记良好的训练数据，这将非常奏效。当然，这种算法的缺点是很难找到具有足够数量的有标签的训练数据。

无监督学习

无监督学习试图在不访问任何带标记的训练数据(标签)的情况下，提取数据中的相关模式。无监督算法的例子包括聚类、降维和异常检测。当我们想要从一个数据集提取模式而不需要特定的预期结果时，可以使用无监督学习技术。无监督方法有一个明显的优势，即不需要带标签的数据作为训练数据。但另一方面，结果可能对人类来说很难解释，并且学习到的模式可能与预期不符。

强化学习

强化学习是从直接经验中学习。它提供了一个环境和一个激励函数，目标是最大化它的奖励。算法采取行动并观察这些行动的结果，然后尝试生成与结果的可取程度相匹配的激励函数。目前，强化学习最有可能的应用是合成的计算机模

拟环境，它允许在短时间内进行数百万或数十亿的探索性交互。

1.4.5　深度学习

深度学习基于 20 世纪 50 年代着手研发的人工神经网络(ANN)。人工神经网络被组织成连接的节点层或感知器。输入以一组数字的形式提供给输入层，结果也通常以数字的形式提供给输出层。输入和输出之间的层称为隐藏层。人工神经网络的目标是迭代地学习每个感知器的权重，以便在输出层产生期望结果的近似值。“深度”一词指的是网络中有许多层(至少七八层，但也可能有数百层)。深度学习网络的例子如图 1-9 所示。

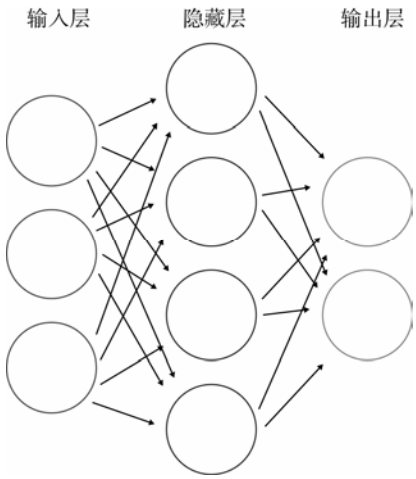

图 1-9　深度神经网络的层次结构

用神经网络模拟人脑的概念早在人工智能研究初期就出现了。然而，这些方法的原始计算能力不足以实现它们的潜力。在 20 世纪末期和 21 世纪初，随着更强大的处理技术的出现，神经网络和深度学习开始成为人工智能的主要方法。算法的进步以及来自互联网的大量训练数据进一步促进了深度学习的发展。标签训练数据的任务通常是通过众源(crowdsourcing)来解决的(例如 Amazon Mechanical Turk)。

Alpha Go

证明深度学习取得巨大进步的一个关键事件是 Alpha Go 战胜了最优秀的人类围棋大师。Alpha Go 最初由总部位于英国的 DeepMind 科技公司开发。该公司于 2014 年被谷歌收购。

要理解这一点的关键是，网络必须“学习”围棋。这与“深蓝”击败加里•卡斯帕罗夫时所采取的方法明显不同。这是由于二者游戏状态的数量不同。在国际象棋中，大约有 10^{45} 个游戏状态，而在围棋中大约有 10^{170} 个。因此，可以将游戏规则和机器学习算法技术结合起来，将深蓝编程为国际象棋专家。而可观测的

宇宙中的原子数大约只有 10^{80}——当你对比这几个数字就能理解围棋游戏的复杂性，以及尝试使用类似专家系统的方法的不可能性。因此，Alpha Go 使用了一种深度神经网络，这种网络是通过观察数百万场围棋比赛，在比赛中不断训练出来的。

图 1-10 将前面介绍的机器学习工具和技术总结在一个图表中。有兴趣的读者可以参照本图来了解机器学习的分类及相互之间的关系。

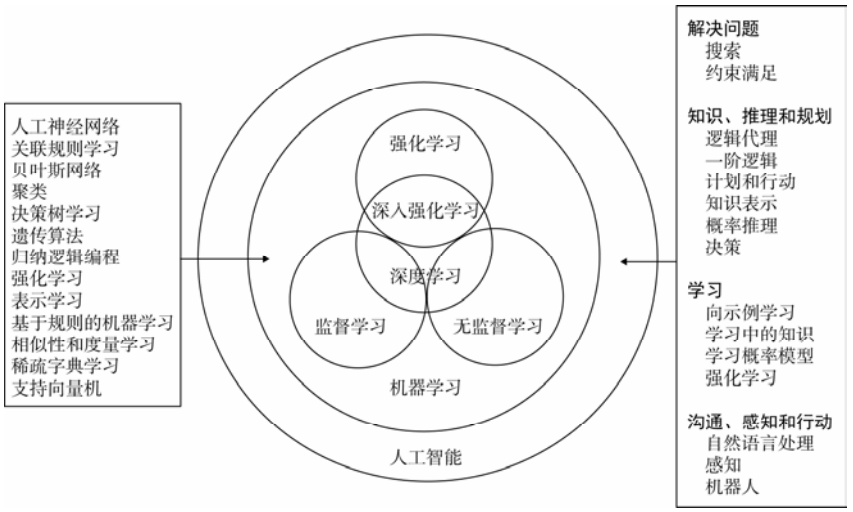

图 1-10 人工智能、机器学习算法和应用。*Deep reinforcement learning：an overview*，Yuxi Li，https://arxiv.org/abs/1701.07274

1.4.6 人工智能面临的挑战

目前，人工智能以监督学习为主，需要数据进行训练。其中一个挑战是如何标记数据来表示网络要学习的所有场景。因此，无监督模型的开发是热门的研究话题。对于许多希望利用 AI 的用户来说，通常无法获得足够的数据。使用有限的数据集进行训练，算法往往会存在偏差，从而导致那些与数据不相似的数据生成错误的预测结果。

人工智能在法律和道德领域也面临挑战。如果机器学习算法作出了一个不受欢迎的判断，很难知道哪一方应该负责。如果一家银行判定某个人无权获得抵押贷款，那么可能不清楚为什么会做出这样的决定，以及谁应该对此负责。

1.5 计算能力和人工智能的普及

摩尔定律在过去几十年中得到了印证。有的程序员还记得最早人们是将程序提交到穿孔卡片上执行。在大型机和小型机时代，计算时间是一种稀缺的资源，只有少数特权可以使用。然而，现在我们大多数人口袋里的智能手机的计算能力

都是这些早期系统无法企及的。

云计算也有类似的效果。在互联网时代早期，专业硬件工程师需要在同一地点的设施中构建服务器机架。今天，只要有足够的资金，就可以编写程序，并且在相当于整个数据中心规模的 IT 基础设施上运行程序，而且可以随时创建和删除这种大规模的环境。

人工智能也是如此。以前，为了构建一个具有语音识别功能的系统，需要使用高度专业的定制硬件和软件，甚至躬身研究这些主题。今天，只需要在程序中调用一种云原生的语音识别服务，就可以向平台添加语音接口。

1.6　规范的 AI 即服务架构

在探讨任何像“AI 即服务”这样广泛的新主题时，重要的是构建出各部分如何整合在一起的蓝图。图 1-11 是无服务器 AI 平台的典型结构图：一个参考架构框架。我们将在整本书中详细介绍这个规范的架构，并将其作为一种常见的参考心智模型。

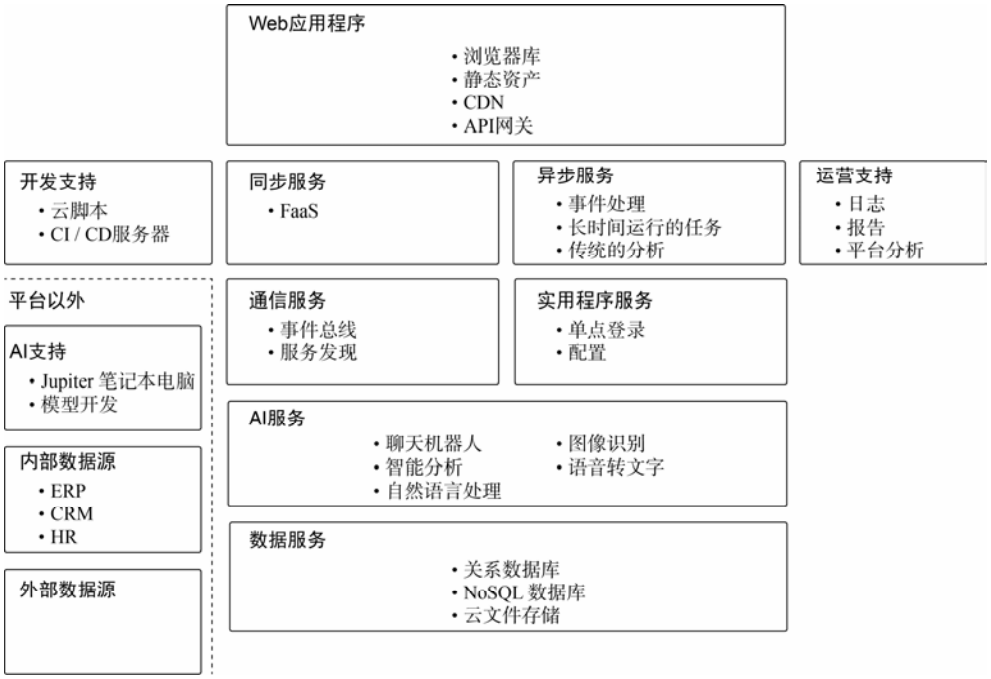

图 1-11　规范的 AI 即服务平台架构

要实现这个体系结构的关键点列举如下：

- 它可以完全通过云原生服务来实现——不需要物理机或虚拟机，也不需要容器。
- 它可以在主要厂商提供的多种云平台上实现。

让我们依次查看这个体系结构的每个元素。

1.6.1 Web 应用程序

典型的平台通过 Web 应用程序层提供功能，即使用 HTTP(S)协议。这一层通常由多个元素组成，包括：

- 静态资产，如图像、样式表和客户端 JavaScript
- 某种形式的内容交付网络(CDN)或页面缓存
- RESTful API
- GraphQL 接口
- 用户注册登录/注销
- 移动API
- 应用程序防火墙

这一层充当客户端请求的主平台的网关。

1.6.2 实时服务

这些服务通常由 Web 应用程序层使用，以便对客户端请求进行即时响应。这些服务代表了平台所有部分之间的公共融合层。例如，一个服务可能负责获取图像并将其传递给人工智能服务进行分析，然后将结果返回给客户端。

1.6.3 批处理服务

通常，这些服务针对运行时间较长的异步任务，包括 ETL (Extract Transform Load)流程、长时间运行的数据加载和传统分析。批处理服务通常将知名的分析引擎(如 Hadoop 或 Spark)作为云原生服务使用，而非自行管理。

1.6.4 通信服务

大多数平台都需要某种形式的异步通信——这通常在某种形式的消息传递基础设施或事件总线上实现。这个通信结构还应包含诸如发现和服务注册的功能。

1.6.5 基础事务服务

这类服务包括安全服务，例如单点登录和联合身份管理，以及网络和配置管理服务，例如 VPC(虚拟私有云)和证书管理。

1.6.6 AI 服务

这是无服务器 AI 平台的智能核心，可以根据平台的关注点，组成一系列 AI 服务。例如，在这里可以找到聊天机器人、自然语言处理或图像识别模型和服务。在很多情况下，这些服务都是通过预封装的现成云原生 AI 服务连接到平台上的。

如果不使用云原生 AI 服务，在部署到平台之前可能需要对模型进行交叉训练。

1.6.7　数据服务

支撑无服务器 AI 堆栈的是数据服务。它们通常混合使用关系数据库、NoSQL 数据库、云文件存储以及其他服务。与系统的其他组件一样，数据层是通过使用云原生数据服务实现的，而不是通过自安装和管理实例实现的。

1.6.8　运营支持

这个部分包含平台操作所需的管理工具，如日志记录、日志分析、消息跟踪警报等。与系统的其他部分一样，可以在不需要安装和管理基础设施的情况下实现操作支持服务。值得注意的是，这些运维支持服务本身可能会使用人工智能服务，从而辅助警报和异常检测。后面的章节将详细介绍这一点。

1.6.9　开发支持

这部分与平台的部署有关，并包含为其他服务类别创建云结构所需的脚本。它还为每个其他服务类别提供持续集成/持续交付管道的支持，以及平台的端到端测试。

1.6.10　平台之外

“平台之外”纳入了一组非平台元素，是否纳入这些元素取决于平台的运营模型。

AI 支持

这包括数据科学类型的调查、定制模型训练，以及调查。稍后我们将看到，训练机器学习系统的过程与实际使用它的过程非常不同。许多使用过程中并不需要训练。

内部数据源

企业平台通常具有与内部系统或遗留系统的连接点。这可能包括 CRM(客户关系管理)和 ERP(企业资源规划)类型的系统或与传统内部 API 的连接。

外部数据源

大多数平台并不是孤立存在的，可能会使用来自第三方 API 的数据和服务。这些将充当我们的无服务器 AI 平台的外部数据源。

1.7　在 Amazon Web Services 上实现

为了有一个更具象的认知，图 1-12 展示了 Amazon Web Services 上的规范体系结构。

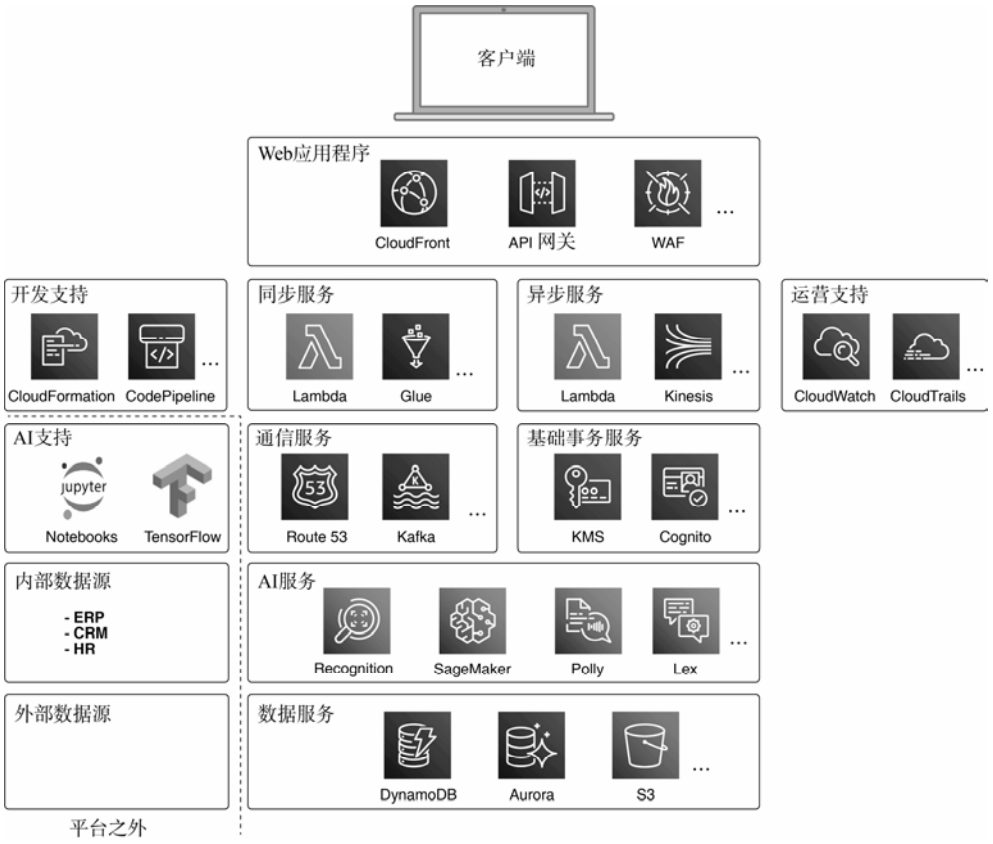

图 1-12 AI 即服务平台在 AWS 上的实现

当然，AWS 平台上所有可用的云原生服务不尽如此。它只是用图例的方式阐释了如何将这些服务组合成一致的体系结构。

为什么选择 AWS？

书中的代码和示例均基于 AWS 平台。这样做有两个原因：

- 就市场占有率而言，AWS 目前是云计算领域的市场领导者。在撰写本文时，AWS 占有 48%的市场份额。这意味着书中的例子将被更广泛的读者所熟悉。
- AWS 在创新方面处于领先地位。我们最近比较了AWS 和其他云提供商之间的多个类别的服务发布日期。我们发现 AWS 发布的服务平均领先竞争对手 2.5 年。这也意味着 AWS 服务产品更加成熟和完整。

此外，在 3 个不同的云(AWS、谷歌、微软)上构建示例系统需要更多的工作。

从这个映射中得到的关键信息如下：

- 在这个系统中，不需要安装和管理服务器。这减少了大量与管理、扩展、

容量规划等有关的操作开销。

- 这些服务的所有创建和部署都是通过一组部署脚本进行控制的，这些脚本可以作为代码资产进行版本控制和管理。
- 人工智能服务可直接使用——不需要让机器学习专家构建系统。

本章已提供了足够的行业趋势背景信息，证明 AI 作为一种服务和无服务器将在未来几年成为平台开发的实际标准。

本书后部将重点介绍实用的配置和示例，切入无服务器 AI 开发的前沿；讨论如何参照本章介绍的规范架构构建一系列复杂的人工智能支持系统。

要强调的是，虽然本书使用的是 AWS，但其架构、原则和实践可以很容易地迁移到其他云上。Azure 和 GCP 都提供了类似于本书中 AWS 示例所使用的相关产品。

接下来，我们将直接构建你的第一个人工智能服务系统！

1.8　本章小结

- 规模单位在持续缩小。下一个逻辑阶段将是功能即服务(FaaS)。
- 无服务器在很大程度上消除了管理复杂 IT 基础设施的负担。
- 服务扩展由云提供商处理，因此不必进行容量规划或复杂的自动扩展设置。
- 无服务器允许企业更多地关注平台特性的开发，较少地关注基础设施和操作。
- 随着商业和技术分析数据量和复杂性的增加，对人工智能服务的需求将越来越大。
- 云原生人工智能服务正在普及，现在即使不是人工智能专家也可以使用这些技术。人工智能服务产品将不断增加。
- 所有这些技术都支持工程驱动化方式来构建无服务器平台，以及人工智能服务消费。