HZ BOOKS
华章IT

云计算与虚拟化技术丛书

Serverless Inside Out

Architecture and Practices

深入浅出Serverless

技术原理与应用实践

陈耿 著

作者是微软全球黑带技术专家，资深云方案架构师。本书汇聚了作者在微软和红帽等知名云计算企业的工作经验与实践。

详细讲解Serverless的技术原理和应用架构，系统介绍基于主流公有云和私有云平台的各种Serverless工具和框架的原理、架构和使用细节。

机械工业出版社
China Machine Press

云计算与虚拟化技术丛书

Serverless Inside Out
Architecture and Practices

深入浅出Serverless

技术原理与应用实践

陈敏 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

深入浅出 Serverless：技术原理与应用实践 / 陈耿著 . —北京：机械工业出版社，2019.1
（云计算与虚拟化技术丛书）

ISBN 978-7-111-61347-3

I. 深… II. 陈… III. 移动终端 - 应用程序 - 程序设计 IV. TN929.53

中国版本图书馆 CIP 数据核字（2018）第 263164 号

深入浅出 Serverless：技术原理与应用实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：郎亚妹 责任校对：殷 虹
印 刷：北京市荣盛彩色印刷有限公司 版 次：2019 年 1 月第 1 版第 1 次印刷
开 本：186mm×240mm 1/16 印 张：15.25
书 号：ISBN 978-7-111-61347-3 定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

华章IT
HZBOOKS | Information Technology

Preface 前　言

容器技术是这几年IT界的热门话题，各行各业都在研究如何通过容器提升企业软件开发、交付和管理的效率。Docker和Kubernetes的成功使得仅凭几个人也可以轻易管理一个包含上千台机器的庞大的计算集群，并且在这个庞大的集群上部署各种各样的应用。云计算催生了容器技术，而容器技术也改变了云计算。凭借在Linux和开源社区的先天优势，这几年Red Hat在容器这一领域风光无限。我在Red Hat参与了各种类型的容器项目，见证了客户使用容器平台满足其各种各样的需求。容器技术的应用可谓百花齐放，范围涉及微服务、DevOps到最近的人工智能和深度学习。在当前容器技术如此火热之际，我突然想，容器会是云计算的终点吗？答案当然是否定的。如果容器不是终点，那么什么东西会成为容器之后的又一个技术热点呢？什么样的技术会让云计算更进一步，让IT及其所服务的各个行业的生产效率更上一层楼呢？

我带着疑问进行了思考和研究。经过一系列调研以及和业界一些朋友的讨论后，我认为Serverless将会是继容器之后又一项改变云计算的技术。回顾云计算发展的历程，从物理机到虚拟机，从虚拟机到容器，业界的关注点其实是一点一点地向上层移动的。通过各种技术手段，我们总是努力降低花费在管理基础设施上的时间和精力，以便将更多的时间放在应用和业务上。因此，过去十多年的云计算的历程，其实是一个“去基础架构”的过程。这个过程让用户可以更快速、更简单、更高效地将想法变成应用，变成在线的服务。

Serverless符合云计算发展的方向，让用户可以将关注点放到具体的业务功能上，而不是底层的计算资源上。Serverless特有的模式存在着潜在的巨大价值。那么，Serverless会取代容器吗？我相信不会。虽然Serverless架构在一些特定的领域会大放异彩，但是容器在未来仍然会是一种重要的应用分发和部署格式。此外容器也将成为许多Serverless平台的基础技术，成为Serverless实现的基石。在未来，Serverless与容器将会有许多结合点。

Serverless 还是一个相对较新的技术领域，各种新的观点、技术和开源项目还在不断酝酿和涌现。作为一名架构师，除了要解决企业当下和近期可能面对的问题外，还需要有一定的前瞻性，掌握未来架构可能的选项，才能对未来的架构做出合理决策。作为一名工程师，必须要紧跟技术的脚步，让自己在不断变化的 IT 洪流中屹立不倒。本书写作的初衷正是为希望了解 Serverless 领域现状的架构师和技术人员提供指南和参考。

本书主要内容

本书是一本介绍 Serverless 技术的书籍，可以让想了解 Serverless 的读者快速了解 Serverless 的概念和原理。此外，书中还用大量的篇幅介绍了当前业界最新的 Serverless 平台、框架和工具的原理、架构和使用细节，内容涵盖了公有云和私有云的 Serverless 平台。

全书共分为 11 章，循序渐进、深入浅出地讲解 Serverless 相关的知识和技术。

前三章重点介绍 Serverless 的概念和原理，为读者构建 Serverless 知识体系打下理论基础。第 1 章介绍了 Serverless 的基础知识，让读者了解 Serverless 的概念及其特点。Serverless 的存在不能脱离这个时代，所以第 2 章详细讨论了 Serverless 涉及的云计算的各种技术，如微服务、容器和 DevOps 等，让读者对 Serverless 的理解更加深入。在理解 Serverless 的基础上，第 3 章介绍了业界目前的 Serverless 的各类平台、工具和框架的实现，让读者对该技术领域的现状有更清晰的认识。

第 4 章和第 5 章详细介绍了公有云 Serverless 平台的技术细节。以 AWS Lambda 和微软的 Azure Functions 为例，向读者介绍了当前主流的公有云厂商在 Serverless 领域的实现。

第 6 章是容器技术的速成教程。容器技术是当下云计算重要的基础技术，也是许多 Serverless 平台的实现基础。通过本章读者可以快速了解当下热门的容器技术（Docker 和 Kubernetes）的原理和基本使用技巧。

第 7～10 章针对私有云的 Serverless 计算平台，分别详细介绍了 OpenWhisk、Kubeless、Fission 及 OpenFaaS 的系统架构、核心概念以及使用技巧，帮助读者了解各类 Serverless 平台的技术特点。

第 11 章针对 Serverless 技术的落地给出了具体建议，总结了本书对 Serverless 技术的观点，并对 Serverless 技术的未来进行了展望。

本书亮点

本书是关于 Serverless 与容器的原创著作。Serverless 是当前的一个热门话题，但是大家对 Serverless 概念并不了解。本书整理了业界当前对 Serverless 的主流观点，梳理了 Serverless 技术发展的现状，是一个系统的 Serverless 指南。

- 最新资讯。原创的 Serverless 著作，为读者呈现业界最新的观点和知识。
- 纵览大局。对 Serverless 的介绍结合了当下云计算的背景，也结合了容器技术。
- 细致入微。在介绍原理和观点的同时也讲解了大量 Serverless 平台的技术细节。
- 互动实操。提供了大量可操作的实验步骤，让读者可以动手体验，加深理解。

本书读者对象

本书介绍了 Serverless 架构的概念、原理以及当前公有云和私有云领域的众多 Serverless 平台的实现，能帮助云计算、容器等领域的软件架构师和技术人员快速了解 Serverless 这一领域的发展现状，为企业和组织的 Serverless 技术选型、转型和落地提供参考。此外，本书涵盖了大量关于当前云计算、容器和 Serverless 领域的观点和话题，因此，也适合作为技术爱好者开阔眼界、增长见闻的指南。

如何阅读本书

如果读者是初次接触 Serverless 的相关知识，推荐按顺序阅读本书的各个章节。通过本书既定的章节顺序，可以循序渐进地了解 Serverless 的相关原理和实现。如果读者对 Serverless 领域已有一定的研究，则可以按需直接阅读感兴趣的章节。

本书引入了大量与 Serverless、云计算、容器和开源软件相关的话题，并针对相关话题给出了相应的参考资料。笔者希望本书是读者研究 Serverless 和云计算相关技术的一张地图，希望通过本书帮助读者找到更多对自身有价值的开源项目和技术。

关于勘误

本书花费了编辑和笔者大量的时间和精力，书中的文字和图表都经过细心斟酌和校对，所有示例的命令和代码都经过笔者亲自验证。但是由于水平有限，且时间仓促，书中

难免存在一些瑕疵和需要改进的地方，欢迎读者将对本书的意见和建议发送至笔者的邮箱（nicosoftware@msn.com）进行交流讨论。读者也可以关注笔者的微信公众号“云来有道”，获取关于本书最新的信息和勘误。

致谢

本书的出版得到了许多朋友的帮助。衷心感谢机械工业出版社华章公司的杨福川老师和李艺老师对本书的策划和编审。两位编辑老师为本书的出版花费了大量心血。此外，也感谢我的妻子丽金。她是本书的第一位读者，为本书提供了许多有益的建议，并帮助审校了书中的所有文字。本书的创作占用了我大量的业余时间，感谢她的支持和包容。

谨以此书献给我的妻子和两个宝贝。

Contents 目　录

前言

第 1 章　Serverless 基础 …… 1

1.1　什么是 Serverless …… 1
1.2　Serverless 带来的价值 …… 3
1.3　Serverless 的技术实现 …… 4
1.3.1　理念与实现 …… 4
1.3.2　FaaS 与 BaaS …… 5
1.4　Serverless 应用架构 …… 7
1.4.1　传统应用架构 …… 7
1.4.2　Serverless 应用架构 …… 7
1.4.3　两种架构的比较 …… 8
1.5　Serverless 的技术特点 …… 9
1.6　Serverless 的应用场景 …… 11
1.7　Serverless 的局限 …… 12
1.8　本章小结 …… 13

第 2 章　Serverless 与相关技术 …… 15

2.1　云计算 …… 15
2.1.1　从私有数据中心到云 …… 15
2.1.2　IaaS、PaaS 与 SaaS …… 16
2.1.3　Serverless 与云计算 …… 17
2.2　微服务 …… 18
2.2.1　从 SOA 到微服务 …… 18
2.2.2　微服务的价值与挑战 …… 19
2.2.3　Serverless 与微服务 …… 19
2.3　容器 …… 20
2.3.1　容器技术的兴起 …… 20
2.3.2　Serverless 与容器 …… 21
2.4　PaaS …… 22
2.4.1　以应用为中心 …… 22
2.4.2　Serverless 与 PaaS …… 23
2.5　FaaS …… 24
2.5.1　Serverless 实现的基础 …… 24
2.5.2　FaaS 的架构 …… 24
2.5.3　函数的生命周期 …… 25
2.5.4　函数工作流 …… 26
2.6　BaaS …… 26
2.6.1　BaaS 的价值 …… 26
2.6.2　广义的 Serverless …… 27
2.7　NoOps …… 27
2.7.1　无人运维吗 …… 27
2.7.2　“无服务器”与“无人运维” …… 28
2.8　DevOps …… 28

2.9 云原生应用 …… 29
2.9.1 因云而生 …… 29
2.9.2 Serverless 与 Cloud Native …… 29
2.10 本章小结 …… 30

第 3 章 Serverless 的实现 …… 31

3.1 Serverless 技术的发展 …… 31
3.2 Serverless 与公有云 …… 33
3.2.1 Amazon Web Services …… 34
3.2.2 Microsoft Azure …… 36
3.2.3 Google Cloud Platform …… 38
3.2.4 Webtask …… 39
3.2.5 Hyper.sh …… 39
3.2.6 阿里云 …… 40
3.2.7 腾讯云 …… 42
3.2.8 小结 …… 43
3.3 Serverless 与私有化部署 …… 43
3.3.1 OpenWhisk …… 43
3.3.2 Fission …… 44
3.3.3 Kubeless …… 45
3.3.4 OpenFaaS …… 45
3.3.5 Fn …… 47
3.3.6 小结 …… 48
3.4 Serverless 框架和工具 …… 49
3.4.1 Serverless Framework …… 49
3.4.2 Chalice …… 50
3.4.3 Claudia.js …… 50
3.4.4 Apex …… 51
3.4.5 Spring Cloud Function …… 51
3.4.6 AWS SAM …… 52
3.4.7 小结 …… 53
3.5 Serverless 后台服务 …… 53
3.6 本章小结 …… 54

第 4 章 AWS Lambda …… 55

4.1 AWS …… 55
4.2 AWS Serverless …… 56
4.3 AWS Lambda 概述 …… 57
4.4 第一个 Serverless 应用 …… 58
4.4.1 获取 AWS 账号 …… 58
4.4.2 AWS Lambda 控制面板 …… 59
4.4.3 创建函数 …… 61
4.4.4 编辑函数 …… 62
4.4.5 测试函数 …… 63
4.4.6 外部访问 …… 63
4.4.7 运维监控 …… 66
4.4.8 回顾 …… 66
4.5 权限控制 …… 66
4.5.1 IAM …… 67
4.5.2 策略 …… 68
4.5.3 角色 …… 68
4.6 编程模型 …… 69
4.6.1 代码开发 …… 69
4.6.2 Handler …… 70
4.6.3 执行上下文 …… 70
4.6.4 日志输出 …… 71
4.6.5 异常处理 …… 72
4.6.6 无状态 …… 72
4.7 事件驱动 …… 73
4.7.1 事件源 …… 73
4.7.2 触发模式 …… 74
4.8 日志监控 …… 75

4.9 开发辅助 77
4.9.1 环境变量 77
4.9.2 标签 77
4.9.3 版本控制 78
4.10 运行限制 78
4.10.1 资源限制 79
4.10.2 并发控制 79
4.11 配置与部署 79
4.12 本章小结 81

第 5 章 Azure Functions 83

5.1 Microsoft Azure 83
5.2 Azure Functions 概述 85
5.3 创建 Azure Serverless 应用 86
5.3.1 注册 Azure 账号 86
5.3.2 Azure 控制台 87
5.3.3 函数应用 88
5.3.4 创建函数 90
5.3.5 调用函数 92
5.3.6 日志与监控 93
5.4 Azure Functions 命令行 95
5.4.1 安装命令行 95
5.4.2 创建本地函数 96
5.4.3 测试本地函数 97
5.4.4 发布至公有云 98
5.5 深入了解 Azure Functions 99
5.5.1 函数应用设置 99
5.5.2 Trigger 与 Bindings 101
5.5.3 函数代理 103
5.5.4 Slot 104
5.6 私有云部署 105
5.7 本章小结 107

第 6 章 容器技术基础 108

6.1 什么是容器 108
6.1.1 容器 109
6.1.2 容器镜像 110
6.1.3 镜像仓库 110
6.1.4 容器编排 111
6.1.5 容器与 Serverless 111
6.2 Docker 111
6.2.1 Vagrant 111
6.2.2 VirtualBox 112
6.2.3 安装 Docker 113
6.2.4 运行容器 114
6.2.5 构建容器镜像 116
6.2.6 分享镜像 117
6.3 Kubernetes 基础 118
6.3.1 命名空间 120
6.3.2 Pod 120
6.3.3 Service 120
6.3.4 Deployment 120
6.3.5 ReplicaSet 121
6.3.6 网络 121
6.3.7 Ingress 121
6.3.8 交互工具 122
6.4 构建 Kubernetes 环境 122
6.4.1 启动 Vagrant Box 123
6.4.2 修改默认域 124
6.5 Kubernetes 实战 124
6.5.1 部署容器 124
6.5.2 弹性扩展 126
6.5.3 服务发现 127
6.5.4 资源组织 128
6.5.5 容器调度 129

6.6 本章小结……131

第 7 章 OpenWhisk……132

7.1 OpenWhisk 项目……132
7.2 Hello Whisk……133
7.3 逻辑架构……135
7.3.1 Namespace……136
7.3.2 Package……136
7.3.3 Action……137
7.3.4 Feed……141
7.3.5 Trigger……141
7.3.6 Rule……142
7.4 系统架构……143
7.5 Kubernetes 部署……146
7.5.1 准备 Kubernetes 集群……146
7.5.2 集群基础设置……146
7.5.3 创建访问入口……149
7.5.4 部署组件……149
7.5.5 加载系统配置……153
7.5.6 测试集群……153
7.5.7 删除集群……154
7.6 Helm 部署……155
7.6.1 安装 Helm……155
7.6.2 环境配置……155
7.6.3 部署 Chart……156
7.6.4 管理应用……156
7.7 蛋糕管理服务……156
7.7.1 开发环境……157
7.7.2 准备数据库……157
7.7.3 定义 Action……157
7.7.4 创建 Package……159
7.7.5 部署 Action……160
7.7.6 发布 API……161
7.8 本章小结……162

第 8 章 Kubeless……163

8.1 Kubeless 项目……163
8.1.1 系统架构……164
8.1.2 运行时……165
8.2 Kubeless 概述……165
8.2.1 部署 Kubeless……165
8.2.2 配置客户端……166
8.2.3 部署函数……167
8.2.4 Kubeless UI……168
8.3 Function……169
8.3.1 函数部署……170
8.3.2 函数调用……172
8.3.3 资源限制……172
8.3.4 自动扩展……173
8.4 Trigger……173
8.4.1 HTTP Trigger……173
8.4.2 Cronjob Trigger……175
8.4.3 Kafka Trigger……175
8.4.4 NATS Trigger……178
8.5 本章小结……179

第 9 章 Fission……180

9.1 Fission 项目……180
9.1.1 逻辑架构……180
9.1.2 系统架构……181
9.2 部署 Fission……182
9.2.1 安装 Helm……182
9.2.2 部署 Fission Chart……182
9.2.3 命令行工具……183

9.2.4 Hello Fission ····· 183
9.3 深入探讨 Fission ····· 184
9.3.1 Environment ····· 185
9.3.2 Function ····· 187
9.3.3 Package ····· 188
9.3.4 Trigger ····· 191
9.4 执行模式 ····· 192
9.4.1 Pool-based 模式 ····· 192
9.4.2 New Deploy 模式 ····· 193
9.5 Workflows ····· 194
9.5.1 Workflows 定义 ····· 194
9.5.2 配置 Workflows ····· 195
9.5.3 Fortune Whale ····· 195
9.6 本章小结 ····· 197

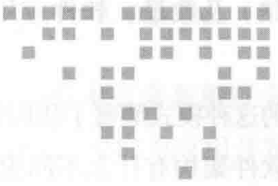
第 10 章 OpenFaaS ····· 199

10.1 OpenFaaS 项目 ····· 199
10.1.1 OpenFaaS 社区 ····· 200
10.1.2 系统架构 ····· 200
10.2 初识 OpenFaaS ····· 200
10.2.1 部署组件 ····· 201
10.2.2 命令行工具 ····· 202
10.2.3 创建函数 ····· 202
10.2.4 图形界面 ····· 203
10.3 OpenFaaS 函数 ····· 203
10.3.1 抽象方式 ····· 203
10.3.2 函数模板 ····· 204
10.3.3 创建函数 ····· 205
10.3.4 构建函数 ····· 206
10.3.5 推送镜像 ····· 207
10.3.6 部署函数 ····· 207
10.4 Watchdog ····· 208
10.4.1 工作原理 ····· 208
10.4.2 配置 Watchdog ····· 209
10.4.3 of-watchdog ····· 210
10.5 监控 ····· 210
10.5.1 监控指标 ····· 210
10.5.2 监控面板 ····· 210
10.5.3 监控预警 ····· 213
10.6 弹性扩展 ····· 214
10.6.1 基于 Alertmanager 扩展 ····· 214
10.6.2 基于 HPA 扩展 ····· 215
10.7 函数应用市场 ····· 215
10.8 本章小结 ····· 217

第 11 章 Serverless 的落地与展望 ····· 218

11.1 Serverless 的落地 ····· 218
11.2 Serverless 平台建设 ····· 219
11.2.1 公有云 ····· 219
11.2.2 私有云 ····· 220
11.2.3 混合云 ····· 220
11.3 Serverless 应用架构转型 ····· 223
11.3.1 开发模式 ····· 223
11.3.2 设计原则 ····· 225
11.3.3 迁移与重构 ····· 226
11.4 Serverless 的未来 ····· 227
11.4.1 建立行业规范 ····· 228
11.4.2 完善工具链 ····· 229
11.4.3 深入结合容器 ····· 229
11.5 本章小结 ····· 230

后记 ····· 232



第 1 章 Chapter 1

Serverless 基础

Serverless 架构即“无服务器”架构，它是一种全新的架构方式，是云计算时代一种革命性的架构模式。本章将介绍 Serverless 架构的基本概念和特点。

1.1 什么是 Serverless

与云计算、容器和人工智能一样，Serverless 是这两年 IT 行业的一个热门词汇，它在各种技术文章和论坛上都有很高的曝光度。但是和其他技术不同，Serverless 是一个不太让人能直观理解其含义的概念。按照英文的字面意思直译的话，Serverless 的中文翻译为“无服务器”，听起来很神秘，很多技术圈的朋友一时不能理解其内涵。技术圈中的人们一般称呼 Serverless 为“无服务器架构”。Serverless 不是具体的一个编程框架、类库或者工具。简单来说，Serverless 是一种软件系统架构思想和方法，它的核心思想是用户无须关注支撑应用服务运行的底层主机。这种架构的思想和方法将对未来软件应用的设计、开发和运营产生深远的影响。

提示 在专业领域的文章中，笔者比较不倾向于使用翻译后的专有名词，故本书保留使用 Serverless 一词。

所谓“无服务器”，并不是说基于 Serverless 架构的软件应用不需要服务器就可以运行，其指的是用户无须关心软件应用运行涉及的底层服务器的状态、资源（比如 CPU、内

存、磁盘及网络）及数量。软件应用正常运行所需要的计算资源由底层的云计算平台动态提供。

Serverless 的这种模式颠覆了我们传统意义上对软件应用部署和运营的认识。到底 Serverless 架构和传统的软件架构有什么不同呢？

在传统的场景里，当用户完成了应用开发后，软件应用将被部署到指定的运行环境，这个运行环境一般以服务器的方式体现，可能是物理主机，也可能是虚拟机。根据业务场景的需要，用户会申请一定数量、一定规格（包含一定数量的 CPU、内存及存储空间）的服务器以满足该应用的正常运行。当应用上线后，根据实际的运营情况，用户可能会申请更多的服务器资源进行扩容，以应对更高的访问量。在这个场景里面，用户需要关心服务器总体的数量，以运行足够的应用实例；需要关心每台服务器的资源是否充足，是否有足够的 CPU 和内存；需要关心服务器的状态，因为每台服务器上应用的部署都要花费不少时间和精力。因为用户需要花费大量的时间、精力在服务器这一计算资源的计划、管理和维护上。

在 Serverless 架构下，情况则截然不同。当用户完成应用开发后，软件应用将被部署到指定的运行环境，这个运行环境不再是具体的一台或多台服务器，而是支持 Serverless 的云计算平台。当有客户端请求到达或特定事件发生时，云计算平台负责将应用部署到某台 Serverless 云计算平台的主机中。Serverless 云计算平台保证该主机提供应用正常运行所需的计算资源。在访问量升高时，云计算平台动态地增加应用的部署实例。当应用空闲一段时间后，云计算平台自动将应用从主机中卸载，并回收资源。在这个场景中，用户无须关心应用运行在哪一台服务器上，也不用关心具体需要几台服务器。原本花费在计划、管理和维护具体服务器上的时间和精力在 Serverless 云计算平台的帮助下被省去了。具体的服务器不再是用户关注的焦点，不再是效率提升的障碍。

通过上述两种场景的对比可见，在 Serverless 架构中并不是不存在服务器，而是服务器对用户而言是透明的，不再是用户所操心的资源对象。

你会发现 Serverless 的实现和软件应用所在的 Serverless 云计算平台有着很大的关系。用户之所以不用再关注服务器是因为底层的云计算平台完成了大量的自动化工作。这个云计算平台可以是公有云，如 Amazon Web Services（AWS）、Microsoft Azure、阿里云或腾讯云，也可以是私有云，如通过 OpenStack、Kubernetes 结合一些 Serverless 框架实现。本书将在后面的章节中对公有云和私有环境里 Serverless 的实现进行详细的介绍。

1.2　Serverless 带来的价值

一项技术被广泛认可和采纳的原因往往不是因为这项技术有多新鲜或多酷，最重要的推动力是其能为业务带来巨大的商业或经济价值。Serverless 架构为应用开发和运营带来了全新的思维，从多个方面为 IT 和企业带来价值。

1. 降低运营复杂度

Serverless 架构使软件应用和服务器实现了解耦，服务器不再是用户开发和运营应用的焦点。在应用上线前，用户无须再提前规划服务器的数量和规格。在运维过程中，用户无须再持续监控和维护具体服务器的状态，只需要关心应用的整体状态。应用运营的整体复杂度下降，用户的关注点可以更多地放在软件应用的体验和改进以及其他能带来更高业务价值的地方。

2. 降低运营成本

服务器不再是用户关注的一个受管资源，运营的复杂度下降，应用运营所需要投入的时间和人力将大大降低。在最好的情况下，可以做到少数几个应用管理员即可管理一个处理海量请求的应用系统。

Serverless 的应用是按需执行的。应用只在有请求需要处理或者事件触发时才会被加载运行，在空闲状态下 Serverless 架构的应用本身并不占用计算资源。在大多数的 Serverless 公有云服务中，如 AWS 及 Azure，Serverless 应用只有处于在线状态下才进行计费，在空闲状态下用户则无须支付费用。对比而言，在传统的架构下，应用被部署到服务器后，无论应用是繁忙还是空闲，应用都将占用其所在的服务器资源。在公有云的场景下，这意味着用户需要支付应用所占用的计算资源，无论应用是否在处理请求。

在 Serverless 架构下，用户只需要为处理请求的计算资源付费，而无须为应用空闲时段的资源占用付费。这个特点将为大规模使用公有云服务的用户节省一笔可观的开销。在私有环境中，Serverless 这种按需执行的模式，可以带来更高的资源利用率。

虽然 Serverless 应用本身在空闲的状态下并不需要支付费用，但是应用所使用到的一些外部服务（如存储和数据库等）仍然可能会产生相关费用。

3. 缩短产品的上市时间

在 Serverless 架构下，应用的功能被解构成若干个细颗粒度的无状态函数，功能与功能之间的边界变得更加清晰，功能模块之间的耦合度大大减小。这使得软件应用的开发效率

更高，应用开发的迭代周期更短。应用所依赖的服务（如数据库、缓存等）可通过平台直接获取，用户无须关心底层细节，因此应用部署的复杂度降低，部署起来更加容易。应用开发和部署的效率提升，使得用户把头脑中的想法变成现实中的代码，然后将代码变成线上运行的服务的这个过程变得前所未有地快速。相对于传统应用，Serverless 架构应用的上市时间（Time To Market，TTM）将大大减少。

4. 增强创新能力

应用的开发和部署效率的提升，使得用户可以用更短的时间、更少的投入尝试新的想法和创意。通过 Serverless 的方法快速做出新创意的应用原型，快速投放给用户使用并获取反馈。如果新的想法获得成功，可以进一步快速对其进行完善和扩展。如果想法不成功，失败所消耗的时间和金钱成本相对于传统的软件应用架构方式而言也是较低的。

1.3 Serverless 的技术实现

作为一种新的思想和方法论，Serverless 可以为企业和用户带来巨大的潜在价值，因此在这几年迅速获得了业界的广泛关注。许多企业和用户都在思考如何能落地和实践 Serverless 这一思想。

1.3.1 理念与实现

首先要明确的一点是，Serverless 是一种软件的架构理念。它的核心思想是让作为计算资源的服务器不再成为用户所关注的一种资源。其目的是提高应用交付的效率，降低应用运营的工作量和成本。以 Serverless 的思想作为基础实现的各种框架、工具及平台，是各种 Serverless 的实现（Implementation）。Serverless 不是一个简单的工具或框架。用户不可能简单地通过实施某个产品或工具就能实现 Serverless 的落地。但是，要实现 Serverless 架构的落地，需要一些实实在在的工具和框架作为有力的技术支撑和基础。

随着 Serverless 的日益流行，这几年业界已经出现了多种平台和工具帮助用户进行 Serverless 架构的转型和落地。目前市场上比较流行的 Serverless 工具、框架和平台有：

- AWS Lambda，最早被大众所认可的 Serverless 实现。
- Azure Functions，来自微软公有云的 Serverless 实现。
- OpenWhisk，Apache 社区的开源 Serverless 框架。
- Kubeless，基于 Kubernetes 架构实现的开源 Serverless 框架。
- Fission，Platform9 推出的开源 Serverless 框架。