CoreOS in Action

CoreOS 实战

在Container Linux上运行应用

[美] Matt Bailey 著
蒲 成 译

MANNING

清华大学出版社

CoreOS 实战

[美] Matt Bailey 著
蒲 成 译

清华大学出版社

Matt Bailey
CoreOS in Action
EISBN: 978-1-61729-374-0

北京市版权局著作权合同登记号　图字：01-2017-7949

图书在版编目(CIP)数据

CoreOS 实战 / (美) 马特·贝利(Matt Bailey) 著；蒲成 译. —北京：清华大学出版社，2018
书名原文：CoreOS in Action
ISBN 978-7-302-49452-2

Ⅰ. ①C…　Ⅱ. ①马…　②蒲…　Ⅲ. ①操作系统　Ⅳ. ①TP316

中国版本图书馆 CIP 数据核字(2018)第 020929 号

责任编辑：王　军　于　平
装帧设计：思创景点
责任校对：曹　阳
责任印制：刘海龙

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：010-62770175　　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：三河市少明印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：11.25　　字　　数：274 千字
版　　次：2018 年 2 月第 1 版　　印　　次：2018 年 2 月第 1 次印刷
印　　数：1～4000
定　　价：49.80 元

产品编号：075915-01

谨以本书献给我的妻子 Jenn 以及我的孩子 Adam 和 Melanie。

译者序

DevOps 和容器化差不多是目前 IT 行业最热门的两个词。这要归因于软件行业日益清晰地认识到，为了按时交付软件产品和服务，开发和运营工作必须紧密合作。特别是在如今移动互联网和大数据分析应用大行其道的背景下，如何成为一名真正的消费者用户并且像消费者用户那样来考虑整件事情的意义，就成为各个企业追求的目标。而 DevOps 和容器化正是为了实现这一目标而被广泛应用的工具。

CoreOS 正是在此背景下诞生的，它是一个基于 Docker 的轻量级容器化 Linux 发行版，专为大型数据中心而设计，旨在通过轻量级系统架构和灵活的应用程序部署能力降低数据中心的维护成本和复杂度。CoreOS 作为 Docker 生态圈中的重要一员，日益得到各大云服务商的重视，目前所有的主流云服务商都提供了对 CoreOS 的支持，其发展风头正劲。CoreOS 是为了计算机集群的基础设施建设而诞生的，专注于自动化、轻松部署、安全、可靠和规模化。作为一个操作系统，CoreOS 提供了在应用容器内部署应用所需的基础功能环境，以及一系列用于服务发现和配置共享的内建工具。

本书从 CoreOS 的基础组成部分开始，深入浅出地讲解了 CoreOS 在私有化部署和云端部署的完整步骤，从而为读者描述了以 CoreOS 为基础的完整生态。如果读者之前从未接触过 CoreOS，那么相信读者在阅读完本书之后，会对 CoreOS 有一个全面的认识，并且具备部署 CoreOS 和在其上构建应用程序栈的基本能力。

本书侧重于介绍 CoreOS 的组件、特性以及部署方式，并辅之以从易到难的示例，以便读者可以动手实践，从而由浅入深地了解 CoreOS 的方方面面。通过这些示例，本书也抽丝剥茧般阐述了 CoreOS 作为一种集成方式如何从其所运行的计算资源池中提取抽象，这样，开发运营人员就能专注于应用和服务，而不会受到 OS 本身各种依赖项的干扰了。作为一本 CoreOS 实战类书籍，本书的内容完全可以满足需要进行 CoreOS 实践的用户的需求。

在此要特别感谢清华大学出版社的编辑，在本书翻译过程中他们提供了颇有助益的帮助，没有其热情付出，本书将难以付梓。

本书全部章节由蒲成翻译，参与翻译的还有何东武、李鹏、李文强、林超、刘洋洋、茆永锋、潘丽臣、王滨、陈世佳、申成龙、王佳、赵栋、潘勇、贠书谦、杨达辉、赵永兰、郑斌、杨晔。

由于译者水平有限，难免会出现一些错误或翻译不准确的地方，如果有读者能够指出并勘正，译者将不胜感激。

译　者

致　谢

我要感谢 Manning 出版社联系我编写本书，并且我要表达对于出版人 Marjan Bace 的谢意，还要感谢 Cynthia Kane 在本书的长时间编写过程中给予我的指导，感谢 Ivan Kirkpatrick 在对本书技术评审过程中所进行的非常细致的工作，感谢 Tiffany Taylor 帮助推动越过终点线的最后一部分内容形成文字，并且要感谢编辑和制作团队的每一个人，其中包括 Janet Vail、Katie Tennant、Dottie Marsico，以及许多在幕后工作的人。此外，我还想感谢#gh 和#omgp 中的所有朋友，你们总是在激励我前进。

我无法用言语来感谢由 Ivan Martinovic 领导的技术评审部门，你们是非常棒的团队，其中包括 Michael Bright、Raffaello Cimbro、Luke Greenleaf、Mike Haller、Sriram Macharla、Palak Mathur、Javier Muñoz Mellid、Thomas Peklak、Austin Riendeau、Kent Spillner、Antonis Tsaltas、Filippo Veneri 以及 Marco Zuppone，还要感谢天赋甚高的论坛贡献者。他们的贡献包括，找出了技术性错误、专业术语错误、打字错误，并且提供了主题建议。每一轮评审过程以及通过论坛主题而实现的每一批反馈，都帮助到本书的成形。

前　言

正如本书的许多读者一样，我也是作为 Linux 和 UNIX 系统以及网络的系统管理员而开启技术行业职业生涯的。另外，就像许多人一样，我从未对可用的自动化程度感到满意过，也从未对其无条件信任过。我们中的一些人或多或少使用过 CFEngine、Puppet 和 Chef 来进行管理，并且使用我们的技术进行更严谨的工程设计和承担较少的系统管理工作。之后容器变得流行起来，并且 CoreOS 的发布大规模地填平了容器与系统管理之间的沟壑。

我是在 2013 年末 CoreOS 刚刚问世时开始使用它的。它是一款大部分系统管理员都认为迟早会出现的 OS。它提供了一种集成方式，以便将服务编制为从其所运行的计算资源池中提取的抽象。Manning 出版社在 2015 年末开始联系我，想要知道我是否有兴趣编写一本 CoreOS 方面的书籍，我接受了这个提议并且开始奋笔疾书。当我由于这个项目而无法在业余时间陪伴我的孩子们时，我也感到愧疚。这是我的第一本书，我发现，内容构思以及在 Vim 中输入这些内容并不是最难的部分，最难的是同时找到充满动力的书籍编写时间和不受打扰的自由时间。而这种情况很少会同时出现，尤其是在家有幼儿的情况下。

我希望《CoreOS 实战》能够引导读者并且为读者带来一些挑战。从某种程度上说，这本书的内容发展遵循了我职业生涯的发展轨道以及此技术领域的发展轨道。具体而言，CoreOS 和类似的系统都旨在将单调乏味的运营工作转变成软件开发，并且将系统管理救火式的工作转变成声明式的工程设计。因此，《CoreOS 实战》是从基础组成部分开始介绍的，并且以完整的软件栈作为结束。

关于本书

《CoreOS 实战》为应用程序架构、系统管理员以及寻求如何在不牺牲开发工作流或者运营简单性的情况下进行规模化计算的信息的人提供了一个有效资源。CoreOS 及其组件套装提供了一种切实可行的方法来进行系统设计，其中高可用性、服务发现以及容错性变得不难实现，并且从一开始就成为核心基础设施和应用程序架构的组成部分。CoreOS 和它所倡导的概念对于开发人员和运营专家来说都是有用的，CoreOS 意识到在某种程度上容器化的意图正变得更易于投入运营、维护和迭代。

如果读者正在阅读本书，那么大概已经注意到了，技术领域的普遍行动就是分解竖井并且将开发和运营这两方面结合到一起。在许多组织中，运营专家和应用程序架构师的角色正在被结合成一个角色，例如开发运营(DevOps)或者站点稳定工程(Site Reliability Engineering)。因而，一些人可能最终面临知识缺口。有时候，本书可能看起来使用更高级的主题组合了对读者而言显而易见的信息，不过那是因为我在尝试为可能不具备成功使用 CoreOS 所需的部分基础知识的人提供完整的全局观念。

本书读者对象

本书的读者对象是系统管理员、软件工程师以及对构建可扩展容错系统感兴趣的人。本书研究了使用 CoreOS 进行运营化和构建服务的软件架构；如果读者有兴趣了解构建可扩展的具有容错性的系统，那么本书就是很好的资料来源。

本书中并没有大量的功能性代码——我基本上是在介绍配置文件以及一些用于 Amazon Web Services 的 YAML 模板。对于 Bash 和通用 Linux 系统管理的基础理解应该就足以让读者入门了。在本书后面的内容中，会提供具有 JavaScript 前端的 Node.js 示例，不过 JavaScript 经验并不是必要的。

在描述本书章节之前，先介绍一些技术背景知识。

背景介绍

大约从 2008 年开始，扩展系统以便满足组织顾客的需要已经催生了包括服务、工具和咨询公司的整个行业。这些行业的最终目标一直都是管理具有较少资源的更大规模的系统——并且要非常快速地进行管理。这些平台即服务(Platform-as-a-Service，PaaS)、基础设施即服务(Infrastructure-as-a-Service，IaaS)以及配置管理套件都旨在将系统管理的重担转换成自动化系统，这样组织才能“轻易地”从规模化目标中将 IT 人力资源释放出来。其理念可以用一个比喻来形容(这个比喻是由 Bill Baker 提出的，这是我能找到的最贴切的比喻)，我们应该将基础设施当作家畜而非宠物来对待。也就是说，计算资源单元是日用品或电器，而非具有名称的独立的、精心维护的服务器。当家畜出现问题时，我们会处理掉它们；而在宠物生病时，我们需要对其进行护理以便它们恢复健康。我们应该充分利用自动化，并且不应该过多关心是否必须进行重构；这样做应该是容易并且可复制的。

不过现实情况是，尝试达成这些可复制性和瞬时性目标通常会极其复杂。这样做的具体方式会变成竖井逻辑和工作流的黑盒，即使是在使用广泛引用的工具也会如此。像 Chef 和 Puppet 这样的配置管理系统对于此复杂性而言尤其脆弱——不是因为它们的设计就是如此，而是因为组织通常会遇到阻碍(技术性和非技术性的)，而这些阻碍的最终解决都是以与这些工具的最佳实践完全无关的方式来处理的。在 IaaS 领域，组织通常会像处理其现场资源那样处理其公有云计算资源，这主要是因为 IaaS 具有允许这样做的灵活性，即使这样做会导致系统不可维护。下面介绍容器。

容器

LXC 是在 Linux 用户空间中创建虚拟化运行时的早期实践。与 chroots 和 jails 相比较，它是一种比较重的抽象，但又比完全虚拟化轻。在 Docker 于 2013 年推出并且围绕 LXC 技术增加大量特性之前，很少有人使用过或者听说过 LXC，最终，Docker 用自己的组件完全替换了 LXC 的组件。在我看来，大体而言，Docker 和容器化解决了虚拟化打算解决的问

题：关注点的简单隔离、系统的复制以及不可变的运行时状态。其优势很明显：依赖性管理变得被轻易包含其中；运行时是标准化的；并且其方法对开发人员足够友好，开发和运营可以使用相同的工具，且每个字节都在使用同一容器。因此，我们已经越来越少地听到“它仅对我适用，而不适用于生产”这样的话了。CoreOS 在某种程度上正是此计算模型的运营化，它利用了在通用、分布式系统模型中容器化的优势。

本书从头至尾都在介绍如何利用此计算模型的优势。读者将了解如何同时在原型环境和云端生产环境中部署和管理 CoreOS。还将了解到如何设计和调整应用程序栈以便它能在此上下文中很好地运行。除了该 OS，还将详细介绍 CoreOS 的每个组件及其应用：etcd 用于配置和发现，rkt 用于另一种方式的容器运行时，fleet 用于分布式服务调度，flannel 用于网络抽象。

分布式计算并非新概念；许多用于分布式系统的模型和软件包自从计算的广泛应用开始就已经问世了。不过这些系统中的大多数模型和软件包都不为人所知，具有高度的专属权，或者隔绝在像科学计算这样的特定行业中。最老的一些设计如今仍然存在的唯一原因就是支持 20 世纪 70 年代的遗留系统，它们为大型机和小型机驱动着分布式计算。

CoreOS 背后的历史与推动因素

单系统映像(Single System Image，SSI)计算的概念是一种 OS 架构，自 20 世纪 90 年代以来并没有看到它有多么活跃，它只在一些长期支持遗留系统的场景中得到了应用。SSI 是一种架构，它将集群中的多台计算机作为单一系统来提供。其中有单一的文件系统、通过共享运行时空间来共享的进程间通信(Interprocess Communication，IPC)，以及进程检查点/迁移。MOSIX/openMosix、Kerrighed、VMScluster 和 Plan 9(原生支持的)都是 SSI 系统。Plan 9 上大概曾进行过大部分当前的开发活动，这应该表明了此计算模型当初的流行性。

SSI 的主要缺陷在于，首先，这些系统通常难以配置和维护，并且并非旨在实现通用性。其次，该领域的发展已经明显停滞了：SSI 中没有什么新东西出现，并且它已经无法跟上发展以用作一个流行模型。我认为这是因为科学和其他大数据计算已经拥抱了网格计算，比如像 Condor、BOINC 和 Slurm 这样的批处理操作模型。这些工具旨在在集群中运行计算任务并且交付结果；SSI 的共享 IPC 无法为这些应用程序提供多少好处，因为数据传输的(时间)成本超过了阻塞式批处理过程的成本。在应用程序服务栈的领域中，通过像 HTTP 这样的协议的抽象以及分布式队列也让人们不再值得对共享 IPC 进行投入。

目前，对于分布式计算而言，问题域是如何有效管理大规模的系统。无论我们是在使用 Web 栈还是分布式批处理，可能都不需要共享 IPC，不过 SSI 带来的其他内容具有更多显而易见的价值：共享文件系统意味着我们仅需要配置一个系统，并且进程检查点和迁移意味着结点都是可丢弃的并且“更类似家畜”。在不使用共享 IPC 的情况下，这些解决方案会难以实现。一些组织转而使用将配置应用到多台机器的配置管理系统，或者设置复杂的具有完全自定义逻辑的监控系统。根据我的经验来看，配置管理系统无法达成目标，因为它仅会完全确保运行时的所有状态；在它们运行完成之后，状态就会变成未知。这些系统更专注于可复制性而非一致性，这是一个好的目标，但无法提供通过分布式文件系统进

行共享配置的可靠性。尝试同时管理进程的监控系统通常要么特定于应用程序，要么难以实现和维护。

无论是有意或无意，像 Docker 这样的容器系统都为重新利用 SSI 的优势奠定了基础，而不需要实现共享的 IPC。Docker 确保了运行时状态，并且提供了从 OS 中抽象出来的执行模型。“不过，”大家可能会想，“这完全与 SSI 相反。现在每一个独立的系统甚至都具有了更为隔离的配置和运行时，而非共享式的!”的确，此方法是不相关的，不过它实现了相同的目标。如果运行时状态仅被定义一次(比如在 Dockerfile 中)，并且在整个容器生命周期中都对其进行维护，那么我们就达成单点配置的目标。并且，如果可以同时远程和独立于其运行之上的 OS 与集群结点来编制独立进程状态的话，我们就达成通用服务在集群范围内的进程调度这一目标。

意识到那些可能性就是需要独立于容器化系统之外的工具的地方。这正是 CoreOS 及其系统套件发挥作用的地方。CoreOS 提供了足够的 OS 以供运行一些服务；其余的都是由 etcd 和 fleet 的编制工作来处理的—— etcd 提供了分布式配置，从中容器可以定义其运行时特征，而 fleet 管理着分布式初始化和容器调度。从内部看，CoreOS 也使用 etcd 来提供分布式锁以便自动管理 OS 升级，这转而又会使用 fleet 在整个集群中平衡服务，这样结点就可以自行升级了。

本书路线图

第 1 章首先简要介绍 CoreOS 生态系统。我提供了容器 OS 中核心系统的一些阐释，以及一个并非真正旨在用于执行而是揭示这些部分如何适配到一起的简要示例。

第 2 章介绍设置一个本地 CoreOS 环境的过程，我们将在本书大部分后续内容中使用它作为沙盒。这也是人们在现实环境中使用的过程，以便为 CoreOS 构建组件，因此进一步关注该章的内容会是一个好的做法。

第 3 章讲解与 CoreOS 容错性和系统升级的方式有关的内容，并且介绍设置一个容错性 Web 应用的处理步骤。我们在本书其余内容中基于这个“Hello World”进行构建。

第 4 章探讨了现实世界的需求和 CoreOS 生产部署的目标，以及与如何处理集群中分布式文件系统选项有关的一个现实示例。

第 5 章会研究十二要素应用方法论以及如何将之应用到希望在 CoreOS 中部署的应用程序栈上。该章会以如何在第 6 章中应用此方法论的概述作为结束。

第 6 章将第 3 章的示例扩展成一个具有许多层的更为真实的 Web 应用。我们还将引入一个持久化数据库层。

第 7 章使用了第 6 章的持久化层并且深入探究了如何让它具有容错性和在所有集群机器中的可扩展性。

第 8 章深入研究 Amazon Web Services(AWS)中 CoreOS 的实践部署。

第 9 章讲解如何使用第 6 章和第 7 章中所构建的整个软件栈，并且以自动化方式将它部署到第 8 章所构造的 AWS 环境中。

第 10 章通过探讨 CoreOS 的系统管理部分总结了本书内容，其中包括日志记录、备份、

扩展以及 CoreOS 的新 rkt 容器系统。

源代码下载

本书中所有示例的源代码，包括一些非常长的 AWS 模板，都可以在 www.manning.com/books/coreos-in-action 下载。也可扫描封底的二维码下载源代码。

作者简介

Matt Bailey 目前是 ZeniMax 的技术主管。他曾致力于高等教育行业，并且曾供职于科学计算、医疗和网络技术公司，以及一些初创型公司。读者可以通过 http://mdb.io 以在线方式联系他。

作者在线

购买了《CoreOS 实战》的读者可以免费访问 Manning 出版社所运营的一个私有网络论坛，读者可以在其中对本书进行评论，提出技术问题，并且接受来自作者和其他读者的帮助。要访问该论坛并且进行订阅，可以将 Web 浏览器导航到 www.manning.com/books/coreos-in-action。这个页面提供了相关的信息，其中包括如何在注册之后登录该论坛，可以得到哪些帮助，以及该论坛上的行为准则。

Manning 出版社对于读者的承诺旨在提供一个场所，其中读者与读者之间以及读者与作者之间可以展开有意义的对话。作者方面的参与程度是无法得到保证的，但对于作者在线的贡献仍旧是自愿的(并且免费的)。我们建议读者尝试向作者提出一些具有挑战性的问题以免他没兴趣关注！

只要本书还在印刷，就可以从出版商的网站上访问作者在线论坛和前述探讨内容的归档。

本书封面介绍

《CoreOS 实战》封面上的图画是一个“叙利亚苦行僧”。穆斯林苦行僧生活在宗教团体中，他们与世隔绝并且过着物资匮乏且冥想式的生活；他们是众所周知的智慧、医药、诗歌、启迪和妙语的源泉。该图例来自于伦敦老邦德街的 William Miller 于 1802 年 1 月 1 日出版的奥斯曼帝国服装图集。该图集的扉页已经丢失，并且我们至今都无法找到它的下落。这本书的目录同时使用英语和法语来标识插图，每张插图都有创作它的两位艺术家的名字，他们无疑一定会为自己的作品被装饰到 200 年后的一本计算机编程书籍的封面上而感到惊讶。

自那时起，衣着习惯已经改变了，当时如此丰富的地区多样性已经逐渐消失。如今通常从衣着很难区分不同国家的居民。也许，尝试从乐观的角度来看，我们已经用文化和视

觉上的多样性换来了更为多样化的个人生活——或者说是更为丰富以及有趣的知识技术生活。Manning 出版社的同仁崇尚创造性、进取性，这个图集中的图片使得两个世纪以前丰富多彩的地区生活跃然于纸上，以其作为图书封面会让计算机行业多一些趣味性。

目　录

第 I 部分

增进了解 CoreOS

在前三章中，大家将了解 CoreOS 到底是什么。我将介绍一些专业术语以及组成 CoreOS 的系统，并且让大家掌握和运行一个沙盒环境。大家还将开始着手处理要贯穿本书而构建的一个应用程序栈。