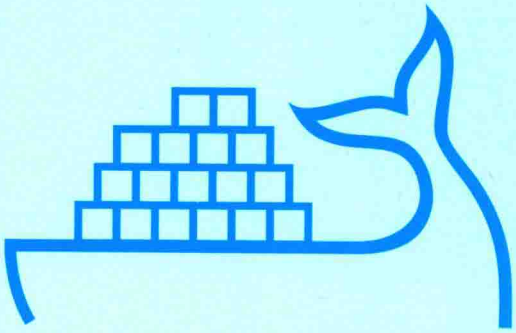
HZ BOOKS
华章IT

容器技术系列



Docker
技术入门与实战

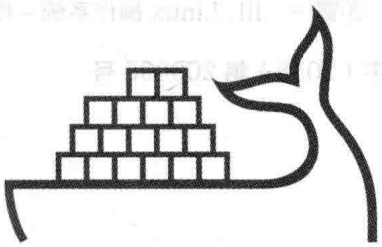
第3版

DOCKER PRIMER

杨保华 戴王剑 曹亚仑 编著

入门 Docker 的首本书，系统化掌握容器技术栈
基于 Docker 最新 18.x 系列版本

机械工业出版社
China Machine Press



Docker
技术入门与实战

第3版

DOCKER PRIMER

杨保华 戴王剑 曹亚仑 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Docker 技术入门与实战 / 杨保华，戴王剑，曹亚仑编著 . —3 版 . —北京：机械工业出版社，2018.8
（容器技术系列）

ISBN 978-7-111-60852-3

I. D… II. ①杨… ②戴… ③曹… III. Linux 操作系统 – 程序设计 IV. TP316.85

中国版本图书馆 CIP 数据核字（2018）第 209095 号

Docker 技术入门与实战　第 3 版

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：吴　怡　　责任校对：李秋荣
印　　刷：北京市兆成印刷有限责任公司　　版　　次：2018 年 9 月第 3 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：26.25
书　　号：ISBN 978-7-111-60852-3　　定　　价：89.00 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

作者简介

杨保华 博士，现为甲骨文架构师。研究方向包括分布式系统、大数据和算法设计等，是容器、网络虚拟化、区块链等技术的早期研究者和布道者。他倡导技术创新与产品、市场相结合，曾负责多个大型平台的架构和设计，以及企业系统的实现和实施。他热爱开源文化，曾积极贡献了多个开源项目。

戴王剑 资深计算机专家。十多年来一直从事系统平台、计算机网络、服务器架构设计，负责过多个省级项目的架构设计。热衷于开源事业，曾积极推动开源技术在生产实践中的应用。

曹亚仑 阿里云高级系统工程师，负责多个专有云/容器云的架构优化与基础运维。2014年开始从事Docker相关技术研究，并在阿里云平台大量实践。擅长大规模集群SRE与DevOps，尤其在自动化运维方面有丰富的经验。

Preface 第3版前言

Docker 诞生于云计算第一个十年的尾巴上。眨眼间，它所代表的现代容器技术，已经占据了云计算的半壁江山。

过去十年里，信息科技依然保持了飞跃式的发展：深度学习的突破给人类摆脱重复劳动带来曙光；分布式账本的崛起为赛博空间奠定信任基础；物联网的成熟让整个星球都将变得更加智慧……这一切都离不开底层计算技术的持续演化，特别是新一代容器化计算平台，为经典计算结构释放出了巨大的潜力。

而计算科技的进步，一直以来就与开源技术和开放文化息息相关。无论是早期的 Unix/Linux 操作系统，还是后来包括 Docker 在内的诸多应用软件，都积极推动了整个信息产业的发展。当下正是新一波科技浪潮来临前的关键时期，掌握最前沿的科技成果，学习最先进的开源工具，对于推动我国乃至全球信息产业的进步都至关重要。

信息科技是全人类的宝贵财富，也是现代文明的基础支撑。每一个信息行业从业人员都应该意识到，持续推动科技创新和文明进步，是时代赋予的重要责任。

Docker 容器技术臻于成熟后，社区涌现出众多优秀的开源项目。这些项目或让计算更加高效便捷，或让平台更加稳定智能，共同构建了繁荣的容器计算生态。围绕这些最新进展，本书第 3 版重点介绍了容器核心技术的最新特性，让读者可以更好地掌握和使用最先进的容器技术。

出版之际，本书开源版本的访问量已经突破一千万，真诚感谢近百位同仁对图书内容的积极建议和反馈。

祝愿世界更加美好，祝愿人人都能快乐幸福！

杨保华

2018 年 7 月于北京

内容简介

本书从 Docker 基本原理开始，深入浅出地讲解 Docker 的构建与操作，内容系统全面，可帮助开发人员、运维人员快速部署 Docker 应用。本书分为四大部分：基础入门、实战案例、进阶技能、开源项目，第一部分（第1～8章）介绍 Docker 与虚拟化技术的基本概念，包括安装、镜像、容器、仓库、数据卷、端口映射等；第二部分（第9～16章）通过案例介绍 Docker 的应用方法，包括与各种操作系统平台、SSH 服务的镜像、Web 服务器与应用、数据库的应用、各类编程语言的接口、容器云等，还介绍了作者在容器实战中的思考与经验总结；第三部分（第17～21章）介绍一些进阶技能，如 Docker 核心技术实现原理、安全、高级网络配置、libnetwork插件化网络功能等；第四部分（第22～28章）介绍与容器开发相关的开源项目，包括 Etcd、Docker Machine、Docker Compose、Docker Swarm、Mesos、Kubernetes 等。第3版根据 Docker 18.x 系列版本，对全书内容进行了全面修订。

目 录 *Contents*

第一部分 Part 1

基础入门

本部分共 8 章，将介绍 Docker 以及容器的基础知识。

第 1 章将介绍容器和 Docker 的来源以及它与现有的虚拟化技术，特别是 Linux 容器技术的关系。

第 2 章将介绍 Docker 的三大核心概念，以及如何在常见的操作系统环境中安装 Docker。

第 3 章～第 5 章通过具体的示例操作，讲解使用 Docker 的常见操作，包括镜像、容器和仓库。

第 6 章将剖析如何在 Docker 中使用数据卷来保存持久化数据。

第 7 章将介绍如何使用端口映射和容器互联来方便外部对容器服务的访问。

第 8 章将介绍如何编写 Dockerfile 配置文件，以及如何使用 Dockerfile 来创建镜像的具体方法和注意事项。

第 1 章 Chapter 1

初识 Docker 与容器

如果说主机时代比拼的是单个服务器物理性能（如 CPU 主频和内存）的强弱，那么在云时代，最为看重的则是凭借虚拟化技术所构建的集群处理能力。

伴随着信息技术的飞速发展，虚拟化的概念早已经广泛应用到各种关键场景中。从 20 世纪 60 年代 IBM 推出的大型主机虚拟化，到后来以 Xen、KVM 为代表的虚拟机虚拟化，再到现在以 Docker 为代表的容器技术，虚拟化技术自身也在不断进行创新和突破。

传统来看，虚拟化既可以通过硬件模拟来实现，也可以通过操作系统软件来实现。而容器技术则更为优雅，它充分利用了操作系统本身已有的机制和特性，可以实现远超传统虚拟机的轻量级虚拟化。因此，有人甚至把它称为“新一代的虚拟化”技术，并将基于容器打造的云平台亲切地称为“容器云”。

毫无疑问，Docker 正是众多容器技术中的佼佼者，是容器技术发展过程中耀眼的一抹亮色。那么，什么是 Docker？它会带来哪些好处？它跟现有虚拟化技术又有何关系？

本章首先会介绍 Docker 项目的起源和发展过程，之后会为大家剖析 Docker 和相关容器技术，以及它为 DevOps 等场景应用带来的巨大便利。最后，还将阐述 Docker 在整个虚拟化领域中的技术定位。

1.1 什么是 Docker

1. Docker 开源项目背景

Docker 是基于 Go 语言实现的开源容器项目。它诞生于 2013 年年初，最初发起者是 dotCloud 公司。Docker 自开源后受到业界广泛的关注和参与，目前已有 80 多个相关开源组件项目（包括 Containerd、Moby、Swarm 等），逐渐形成了围绕 Docker 容器的完整的生态体系。

dotCloud公司也随之快速发展壮大，在2013年年底直接改名为Docker Inc，并专注于Docker相关技术和产品的开发，目前已经成为全球最大的Docker容器服务提供商。官方网站为docker.com，如图1-1所示。

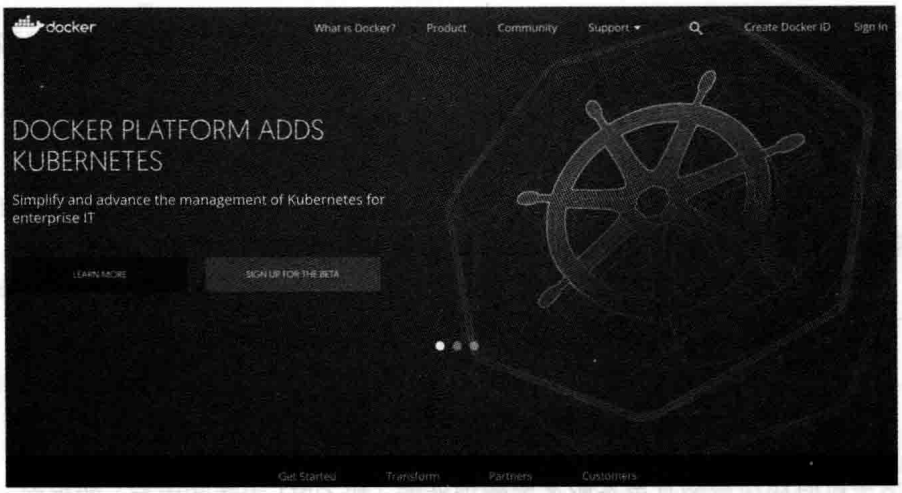


图1-1 Docker官方网站

Docker项目已加入Linux基金会，并遵循Apache 2.0协议，全部开源代码均在https://github.com/docker项目仓库进行维护。在Linux基金会最近一次关于“最受欢迎的云计算开源项目”的调查中，Docker仅次于2010年发起的OpenStack项目，并仍处于上升趋势。2014年，Docker镜像下载数达到了一百万次，2015年直接突破十亿次，2017年更是突破了惊人的百亿次。

现在主流的操作系统包括Linux各大发行版、macOS、Windows等都已经支持Docker。例如，Redhat RHEL 6.5/CentOS 6.5、Ubuntu 16.04以及更新的版本，都已经在官方软件源中默认带有Docker软件包。此外，各大云服务提供商也纷纷推出了基于Docker的服务。Google公司在其Platform as a Service（PaaS）平台及服务中广泛应用了Docker容器；IBM公司与Docker公司达成了战略合作伙伴关系，进行云业务上的深入技术合作；Microsoft公司在其Azure云平台上支持安全可扩展的Docker集群方案；公有云提供商Amazon在其AWS云平台上集成了对Docker的支持，提供高性能快速的部署。

Docker的构想是要实现“Build, Ship and Run Any App, Anywhere”，即通过对应用的封装（Packaging）、分发（Distribution）、部署（Deployment）、运行（Runtime）生命周期进行管理，达到应用组件级别的“一次封装，到处运行”。这里的应用组件，既可以是一个Web应用、一个编译环境，也可以是一套数据库平台服务，甚至是一个操作系统或集群。

基于Linux平台上的多项开源技术，Docker提供了高效、敏捷和轻量级的容器方案，并支持部署到本地环境和多种主流云平台。可以说，Docker首次为应用的开发、运行和部署提供了“一站式”的实用解决方案。

2. Linux 容器技术——巨人的肩膀

与大部分新兴技术的诞生一样，Docker 也并非“从石头缝里蹦出来的”，而是站在前人的肩膀上。其中最重要的就是 Linux 容器（Linux Containers，LXC）技术。IBM DeveloperWorks 网站关于容器技术的描述十分准确：“容器有效地将由单个操作系统管理的资源划分到孤立的组中，以更好地在孤立的组之间平衡有冲突的资源使用需求。与虚拟化相比，这样既不需要指令级模拟，也不需要即时编译。容器可以在核心 CPU 本地运行指令，而不需要任何专门的解释机制。此外，也避免了准虚拟化（para-virtualization）和系统调用替换中的复杂性。”

当然，LXC 也经历了长期的演化。最早的容器技术可以追溯到 1982 年 Unix 系列操作系统上的 chroot 工具（直到今天，主流的 Unix、Linux 操作系统仍然支持和带有该工具）。早期的容器实现技术包括 Sun Solaris 操作系统上的 Solaris Containers（2004 年发布），FreeBSD 操作系统上的 FreeBSD jail（2000 年左右发布），以及 GNU/Linux 上的 Linux-VServer（http://linux-vserver.org，2001 年 10 月）和 OpenVZ（http://openvz.org，2005 年）。

在 LXC 之前，这些相关技术经过多年的演化已经十分成熟和稳定，但是由于种种原因，它们并没有被很好地集成到主流的 Linux 内核中，使用起来并不方便。例如，如果用户要使用 OpenVZ 技术，需要先手动给操作系统打上特定的内核补丁方可使用，而且不同版本并不一致。类似的困难造成在很长一段时间内这些优秀的技术只在技术人员的小圈子中交流。

后来 LXC 项目借鉴了前人成熟的容器设计理念，并基于一系列新引入的内核特性，实现了更具扩展性的虚拟化容器方案。更加关键的是，LXC 终于被集成到到主流 Linux 内核中，进而成为 Linux 系统轻量级容器技术的事实标准。从技术层面来看，LXC 已经趟过了绝大部分的“坑”，完成了容器技术实用化的大半历程。

3. 从 Linux 容器到 Docker

在 LXC 的基础上，Docker 进一步优化了容器的使用体验，让它进入寻常百姓家。首先，Docker 提供了各种容器管理工具（如分发、版本、移植等），让用户无须关注底层的操作，更加简单明了地管理和使用容器；其次，Docker 通过引入分层文件系统构建和高效的镜像机制，降低了迁移难度，极大地改善了用户体验。用户操作 Docker 容器就像操作应用自身一样简单。

早期的 Docker 代码实现是直接基于 LXC 的。自 0.9 版本开始，Docker 开发了 libcontainer 项目作为更广泛的容器驱动实现，从而替换掉了 LXC 的实现。目前，Docker 还积极推动成立了 runC 标准项目，并贡献给开放容器联盟，试图让容器的支持不再局限于 Linux 操作系统，而是更安全、更开放、更具扩展性。

简单地讲，读者可以将 Docker 容器理解为一种轻量级的沙盒（sandbox）。每个容器内运行着一个应用，不同的容器相互隔离，容器之间也可以通过网络互相通信。容器的创建和停止十分快速，几乎跟创建和终止原生应用一致；另外，容器自身对系统资源的额外需求

也十分有限，远远低于传统虚拟机。很多时候，甚至直接把容器当作应用本身也没有任何问题。

笔者相信，随着 Docker 技术的进一步成熟，它将成为更受欢迎的容器虚拟化技术实现，并在云计算和 DevOps 等领域得到更广泛的应用。

1.2 为什么要使用 Docker

1. Docker 容器虚拟化的好处

Docker 项目的发起人、Docker 公司 CTO Solomon Hykes 认为，Docker 在正确的地点、正确的时间顺应了正确的趋势——如何正确地构建应用。

在云时代，开发者创建的应用必须要能很方便地在网络上传播，也就是说应用必须脱离底层物理硬件的限制；同时必须是“任何时间任何地点”可获取的。因此，开发者们需要一种新型的创建分布式应用程序的方式，快速分发和部署，而这正是 Docker 所能够提供的最大优势。

举个简单的例子，假设用户试图基于最常见的 LAMP（Linux+Apache+MySQL+PHP）组合来构建网站。按照传统的做法，首先需要安装 Apache、MySQL 和 PHP 以及它们各自运行所依赖的环境；之后分别对它们进行配置（包括创建合适的用户、配置参数等）；经过大量的操作后，还需要进行功能测试，看是否工作正常；如果不正常，则进行调试追踪，意味着更多的时间代价和不可控的风险。可以想象，如果应用数目变多，事情会变得更加难以处理。

更为可怕的是，一旦需要服务器迁移（例如从亚马逊云迁移到其他云），往往需要对每个应用都进行重新部署和调试。这些琐碎而无趣的“体力活”，极大地降低了用户的工作效率。究其根源，是这些应用直接运行在底层操作系统上，无法保证同一份应用在不同的环境中行为一致。

而 Docker 提供了一种更为聪明的方式，通过容器来打包应用、解耦应用和运行平台。这意味着迁移的时候，只需要在新的服务器上启动需要的容器就可以了，无论新旧服务器是否是同一类型的平台。这无疑将帮助我们节约大量的宝贵时间，并降低部署过程出现问题的风险。

2. Docker 在开发和运维中的优势

对开发和运维（DevOps）人员来说，最梦寐以求的效果可能就是一次创建或配置，之后可以在任意地方、任意时间让应用正常运行，而 Docker 恰恰是可以实现这一终极目标的“瑞士军刀”。具体说来，在开发和运维过程中，Docker 具有如下几个方面的优势：

- **更快速的交付和部署**。使用 Docker，开发人员可以使用镜像来快速构建一套标准的开发环境；开发完成之后，测试和运维人员可以直接使用完全相同的环境来部署代码。只要是开发测试过的代码，就可以确保在生产环境无缝运行。Docker 可以快速