

# Docker实践

作者：伊恩·米尔(Ian Miell)

## 目 录

[版权信息](#)

[版权声明](#)

[内容提要](#)

[译者简介](#)

[序](#)

[前言](#)

[致谢](#)

[关于本书](#)

[关于封面插画](#)

[第一部分 Docker基础](#)

[第1章 Docker初探](#)

[1.1 Docker是什么以及为什么用Docker](#)

[1.1.1 Docker是什么](#)

[1.1.2 Docker有什么好处](#)

[1.1.3 关键的概念](#)

[1.2 构建一个Docker应用程序](#)

[1.2.1 创建新的Docker镜像的方式](#)

[1.2.2 编写一个Dockerfile](#)

[1.2.3 构建一个Docker镜像](#)

[1.2.4 运行一个Docker容器](#)

[1.2.5 Docker分层](#)

[1.3 小结](#)

[第2章 理解Docker——深入引擎室](#)

[2.1 Docker的架构](#)

[2.2 Docker守护进程](#)

[技巧1 向世界开放Docker守护进程](#)

[技巧2 以守护进程方式运行容器](#)

[技巧3 将Docker移动到不同分区](#)

[2.3 Docker客户端](#)

[技巧4 使用socat监控Docker API流量](#)

[技巧5 使用端口连接容器](#)

[技巧6 链接容器实现端口隔离](#)

[技巧7 在浏览器中使用Docker](#)

[2.4 Docker注册中心](#)

[技巧8 建立一个本地Docker注册中心](#)

[2.5 Docker Hub](#)

[技巧9 查找并运行一个Docker镜像](#)

[2.6 小结](#)

[第二部分 Docker与开发](#)

[第3章 将Docker用作轻量级虚拟机](#)

[3.1 从虚拟机到容器](#)

[技巧10 将虚拟机转换为容器](#)

[技巧11 类宿主机容器](#)

[技巧12 将一个系统拆成微服务容器](#)

[3.2 管理容器的服务](#)

[技巧13 管理容器内服务的启动](#)

[3.3 保存和还原工作成果](#)

[技巧14 在开发中“保存游戏”的方式](#)

[技巧15 给Docker打标签](#)

[技巧16 在Docker Hub上分享镜像](#)

[技巧17 在构建时指向特定的镜像](#)

[3.4 进程即环境](#)

[技巧18 在开发中“保存游戏”的方式](#)

[3.5 小结](#)

## [第4章 Docker日常](#)

### [4.1 卷——持久化问题](#)

[技巧19 Docker卷——持久化的问题](#)

[技巧20 通过BitTorrent Sync的分布式卷](#)

[技巧21 保留容器的bash历史](#)

[技巧22 数据容器](#)

[技巧23 使用SSHFS挂载远程卷](#)

[技巧24 通过NFS共享数据](#)

[技巧25 开发工具容器](#)

### [4.2 运行容器](#)

[技巧26 在Docker里运行GUI](#)

[技巧27 检查容器](#)

[技巧28 干净地杀掉容器](#)

[技巧29 使用Docker Machine来置备Docker宿主机](#)

### [4.3 构建镜像](#)

[技巧30 使用ADD将文件注入到镜像](#)

[技巧31 重新构建时不使用缓存](#)

[技巧32 拆分缓存](#)

### [4.4 保持阵型](#)

[技巧33 运行Docker时不加sudo](#)

[技巧34 清理容器](#)

[技巧35 清理卷](#)

[技巧36 解绑容器的同时不停掉它](#)

[技巧37 使用DockerUI来管理Docker守护进程](#)

[技巧38 为Docker镜像生成一个依赖图](#)

[技巧39 直接操作——对容器执行命令](#)

### [4.5 小结](#)

## [第5章 配置管理——让一切井然有序](#)

### [5.1 配置管理和Dockerfile](#)

[技巧40 使用ENTRYPOINT创建可靠的定制工具](#)

[技巧41 在构建中指定版本来避免软件包的漂移](#)

[技巧42 用perl-p-i-e替换文本](#)

[技巧43 镜像的扁平化](#)

[技巧44 用alien管理外来软件包](#)

[技巧45 把镜像逆向工程得到Dockerfile](#)

### [5.2 传统配置管理工具与Docker](#)

[技巧46 传统方式：搭配make和Docker](#)

[技巧47 借助Chef Solo构建镜像](#)

[技巧48 从源到镜像的构建](#)

### [5.3 小即是美](#)

[技巧49 保持构建镜像更小的Dockerfile技巧](#)

[技巧50 让镜像变得更小的技巧](#)

[技巧51 通过BusyBox和Alpine来精简Docker镜像](#)

[技巧52 Go模型的最小容器](#)

[技巧53 使用inotifywait给容器瘦身](#)

[技巧54 大也可以美](#)

### [5.4 小结](#)

## [第三部分 Docker与DevOps](#)

## [第6章 持续集成：加快开发流水线](#)

### [6.1 Docker Hub自动化构建](#)

[技巧55 使用Docker Hub工作流](#)

### [6.2 更有效的构建](#)

[技巧56 使用eatmydata为I/O密集型构建提速](#)

[技巧57 设置一个软件包缓存用于加快构建速度](#)

[技巧58 在Docker内部运行Selenium测试](#)

### [6.3 容器化CI过程](#)

[技巧59 包含一个复杂的开发环境](#)

[技巧60 在一个Docker容器里运行Jenkins主服务器](#)

[技巧61 使用Jenkins的Swarm插件扩展CI](#)

### [6.4 小结](#)

## [第7章 持续交付：与Docker原则完美契合](#)

### [7.1 在CD流水线上与其他团队互动](#)

#### [技巧62 Docker契约——减少摩擦](#)

### [7.2 推动Docker镜像的部署](#)

#### [技巧63 手动同步注册中心镜像](#)

#### [技巧64 通过受限连接交付镜像](#)

#### [技巧65 以TAR文件方式共享Docker对象](#)

### [7.3 为不同环境配置镜像](#)

#### [技巧66 使用etcd通知容器](#)

### [7.4 升级运行中的容器](#)

#### [技巧67 使用confd启用零停机时间切换](#)

### [7.5 小结](#)

## [第8章 网络模拟：无痛的现实环境测试](#)

### [8.1 容器通信——超越手工链接](#)

#### [技巧68 一个简单的Docker Compose集群](#)

#### [技巧69 一个使用Docker Compose的SQLite服务器](#)

#### [技巧70 使用Resolvable通过DNS查找容器](#)

### [8.2 使用Docker来模拟真实世界的网络](#)

#### [技巧71 使用Comcast模拟有问题的网络](#)

#### [技巧72 使用Blockade模拟有问题的网络](#)

### [8.3 Docker和虚拟网络](#)

#### [技巧73 使用Weave建立一个基底网络](#)

#### [技巧74 Docker的网络与服务功能](#)

### [8.4 小结](#)

## [第四部分 生产环境中的Docker](#)

## [第9章 容器编排：管理多个Docker容器](#)

### [9.1 简单的单台宿主机](#)

#### [技巧75 使用systemd管理宿主机上的容器](#)

#### [技巧76 使用systemd编排宿主机上的容器](#)

### [9.2 多宿主机Docker](#)

#### [技巧77 使用Helios手动管理多宿主机Docker](#)

#### [技巧78 基于Swarm的无缝Docker集群](#)

#### [技巧79 使用Kubernetes集群](#)

#### [技巧80 在Mesos上构建框架](#)

#### [技巧81 使用Marathon细粒度管理Mesos](#)

### [9.3 服务发现：我们有什么](#)

#### [技巧82 使用Consul来发现服务](#)

#### [技巧83 使用Registrator进行自动化服务注册](#)

### [9.4 小结](#)

## [第10章 Docker与安全](#)

### [10.1 Docker访问权限及其意味着什么](#)

#### [你在乎吗](#)

### [10.2 Docker中的安全手段](#)

#### [技巧84 限制能力](#)

#### [技巧85 Docker实例上的HTTP认证](#)

#### [技巧86 保护Docker API](#)

### [10.3 来自Docker以外的安全](#)

#### [技巧87 OpenShift——一个应用程序平台即服务](#)

#### [技巧88 使用安全选项](#)

### [10.4 小结](#)

## [第11章 一帆风顺——生产环境中的Docker以及运维上的考量](#)

### [11.1 监控](#)

#### [技巧89 记录容器的日志到宿主机的syslog](#)

#### [技巧90 把Docker日志发送到宿主机的输出系统](#)

#### [技巧91 使用cAdvisor监控容器](#)

### [11.2 资源控制](#)

#### [技巧92 限制容器可以运行的内核](#)

#### [技巧93 给重要的容器更多CPU](#)

#### [技巧94 限制容器的内存使用](#)

### [11.3 Docker的系统管理员用例](#)

[技巧95 使用Docker来运行cron作业](#)

[技巧96 通过“保存游戏”的方法来备份](#)

[11.4 小结](#)

[第12章 Docker生产环境实践——应对各项挑战](#)

[12.1 性能——不能忽略宿主机](#)

[技巧97 从容器访问宿主机资源](#)

[技巧98 Device Mapper存储驱动和默认的容器大小](#)

[12.2 在容器出问题时——调试Docker](#)

[技巧99 使用nsenter调试容器的网络](#)

[技巧100 无须重新配置，使用tcpflow进行实时调试](#)

[技巧101 调试在特定宿主机上出问题的容器](#)

[12.3 小结](#)

[附录A 安装并使用Docker](#)

[附录B Docker配置](#)

[附录C Vagrant](#)

[欢迎来到异步社区！](#)

# 版权信息

书名：Docker实践

ISBN：978-7-115-47458-2

本书由人民邮电出版社发行数字版。版权所有，侵权必究。

---

您购买的人民邮电出版社电子书仅供您个人使用，未经授权，不得以任何方式复制和传播本书内容。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

---

- 著 [美] 伊恩·米尔 (Ian Miell)  
[美] 艾丹·霍布森·塞耶斯 (Aidan Hobson Sayers)
- 译 吴佳兴 梁晓勇 黄博文 杨 锐
- 责任编辑 杨海玲
- 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>
- 读者服务热线：(010)81055410  
反盗版热线：(010)81055315

# 版权声明

Original English language edition, entitled *Docker in Practice* by Ian Miell and Aidan Hobson Sayers published by Manning Publications Co., 209 Bruce Park Avenue, Greenwich, CT 06830. Copyright © 2016 by Manning Publications Co.

Simplified Chinese-language edition copyright © 2018 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由Manning Publications Co.授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

# 内容提要

本书由浅入深地讲解了Docker的相关内容，涵盖从开发环境到DevOps流水线，再一路到生产环境的整个落地过程以及相关的实用技巧。书中介绍Docker的核心概念和架构，以及将Docker和开发环境有机、高效地结合起来的方法，包括用作轻量级的虚拟机以及构建和宿主编排、配置管理、精简镜像等。不仅如此，本书还通过“问题/解决方案/讨论”的形式，将“Docker如何融入DevOps流水线”“如何在生产环境落地”等一系列难题拆解成101个相关的实用技巧，为读者提供解决方案以及一些细节和技巧方面的实践经验。阅读本书，读者将学到的不只是Docker，还包括持续集成、持续交付、构建和镜像管理、容器编排等相关领域的一线生产经验。本书编写时一些案例参考的Docker版本是Docker 1.9。

本书要求读者具备一定的容器管理和运维的基础知识，适合想要将Docker投入实践的相关技术人员阅读，尤其适合具有中高级DevOps和运维背景的读者阅读。

# 译者简介

吴佳兴，毕业于华东理工大学计算机系，目前是bilibili基础平台研发团队的一员，主要研究方向有CI/CD、监控和运维自动化、基于容器的PaaS平台建设、微服务架构等。2014年年底有幸加入DockOne社区，作为译者，利用闲暇时间为社区贡献一些微薄的力量。个人博客devopstarter.info。欢迎邮件联系（wjsx\_colstu@hotmail.com）。

黄博文，ThoughtWorks资深软件工程师/咨询师，担任过开发、测试、运维、技术经理等角色，在国内外多家企业做过技术教练以及技术咨询，拥有丰富的敏捷团队工作经验。目前专注于DevOps技术及云端架构，在搭建持续集成及部署平台、自动化构建基础设施、虚拟化环境、云端运维等方面有着丰富的经验。拥有AWS解决方案架构师以及开发者证书。个人博客为www.huangbowen.net，个人邮箱为huangbowen521@gmail.com。译作有《Effective JavaScript》《Html5和CSS3响应式设计指南》《C#多线程编程实战》《面向对象的思考过程》《基础设施即代码》等。

杨锐，前ThoughtWorks咨询师，DevOps领域持续关注者，任职期间曾任某海外大型项目DevOps工程师，对其持续交付、基础设施即代码、流水线即代码等方面进行了持续推动，对云计算、容器化、持续交付等有一定经验。现供职美团点评。

梁晓勇，毕业于厦门大学，现任某互联网金融公司架构师，DockOne社区编外人员。长期奋战在技术研发第一线，在网络管理、技术开发、架构设计等方面略有心得。热爱互联网技术，积极投身开源社区，对Docker等容器技术具有浓厚兴趣。欢迎邮件联系（sunlxy@yahoo.com）。



# 序

可能是鄙人的拙见，不过Docker真的相当了不起。

在不久以前应用程序还是一些庞大的单体软件，独自运行在一堆钢铁和硅块上。这样的情况持续了很多年，它们拒绝改变，不思进取。对于那些想要快速前行的组织而言，这的确是一个难题，因此虚拟机的崛起也就不足为奇了。应用程序不必再和这些硬件捆绑在一起，这使得一切可以更快更替，也更加灵活。

但是，虚拟机本身也很复杂。它们在其他机器上模拟出整台计算机，而这台虚拟计算机仍然是异常复杂并且需要管理的。再者，因为虚拟机更小而且更容易被创建，所以围绕着它们也就产生了更多需要管理的东西。

我们到底该如何管理所有的复杂性呢？通过配置管理可以办到，当然，这又是另外一个极其复杂的系统。

Docker则采取了一种截然不同的做法。如果用户将软件放到一个容器里，它就会将应用程序本身的复杂度与基础设施相隔离开来，这使得基础设施本身可以变得更加简单，而应用程序也更加易于装配。基于这样的组织效率，与虚拟机相比，技术速度和执行效率都有了巨大的飞跃。容器的启动是毫秒级的，而不是分钟级的。内存是共享的，而不是预先分配的。这使得用户的应用程序能够以更低的成本运行，同样也意味着他可以按照想要的方式来架设应用，而不用再受缓慢的、不太灵活的基础设施的制约。

在我第一次见到Solomon Hykes（即“Docker之父”）谈论Docker并将其与装载集装箱做类比的时候，我便意识到他正在做的是一件了不起的大事。全球航运业建立标准之前的混乱状态可以很好地用来类比容器出现以前管理软件的复杂状态。Solomon的观点非常有说服力，我甚至为此开了一家公司，专门围绕Docker构建工具，这家公司最终被Docker公司收购了，并且最终变成了我们现在所熟知的Docker Compose。

初次见到Ian是在伦敦我们组织的某个Docker聚会（meetup）上。当时，我们坚称：“Docker还没有为生产环境做好准备，请不要在生产环境里使用它！”然而，Ian是何许人也，他无视这个看似明智的建议，一意孤行，非要让它运行在生产环境里。那时，他和Aidan一起为OpenBet公司工作，当我看到他们用我们当时的代码处理过的货币数额时，真的让我感到有点儿懵。

Ian和Aidan发现，他们在使用Docker的过程中收获的价值超越了其测试阶段使用它时伴随而来的不便之处。他们在很早的时候便使用上了这一技术，因此，对怎样更好地运用它也有自己独到的见解。他们在OpenBet内部构建的工具指出了Docker本身缺失的一些东西，而我们之间私下的一些闲聊也实际影响了我们所采用的设计和产品方向。

Docker自Ian和Aidan第一次使用以来，已然有了一些长足的进步，到如今已有成千上万的组织在用它来解决遇到的真实问题，如更快地装配软件、管理惊人的复杂性、提高基础设施的效率、解决“这个在我机器上运行得好好的”问题等。这促使我们在构建、部署和管理软件方式上的巨大转变，并且围绕着它正在形成一套全新的工具及理念。容器化的美好前景的确是让人兴奋的，但是它同我们之前所习惯的方式也迥然不同。

对读者而言，恐怕很难从这本书中看到怎样由此及彼，但是这本书中涵盖了大量如何应用Docker去解决自己当下遇到的种种问题的实用建议。遵循这些建议，用户的组织将能够快速前行。同时或许更重要的是，构建和部署应用的过程将会变得更有意思。

Ben Firshman

Docker公司产品管理总监、Docker Compose联合创始人



欢迎访问：电子书学习和下载网站 (<https://www.shgis.cn>)

文档名称：《Docker实践》伊恩·米尔(Ian Miell) 著.epub

请登录 <https://shgis.cn/post/269.html> 下载完整文档。

手机端请扫码查看：

