O'REILLY®

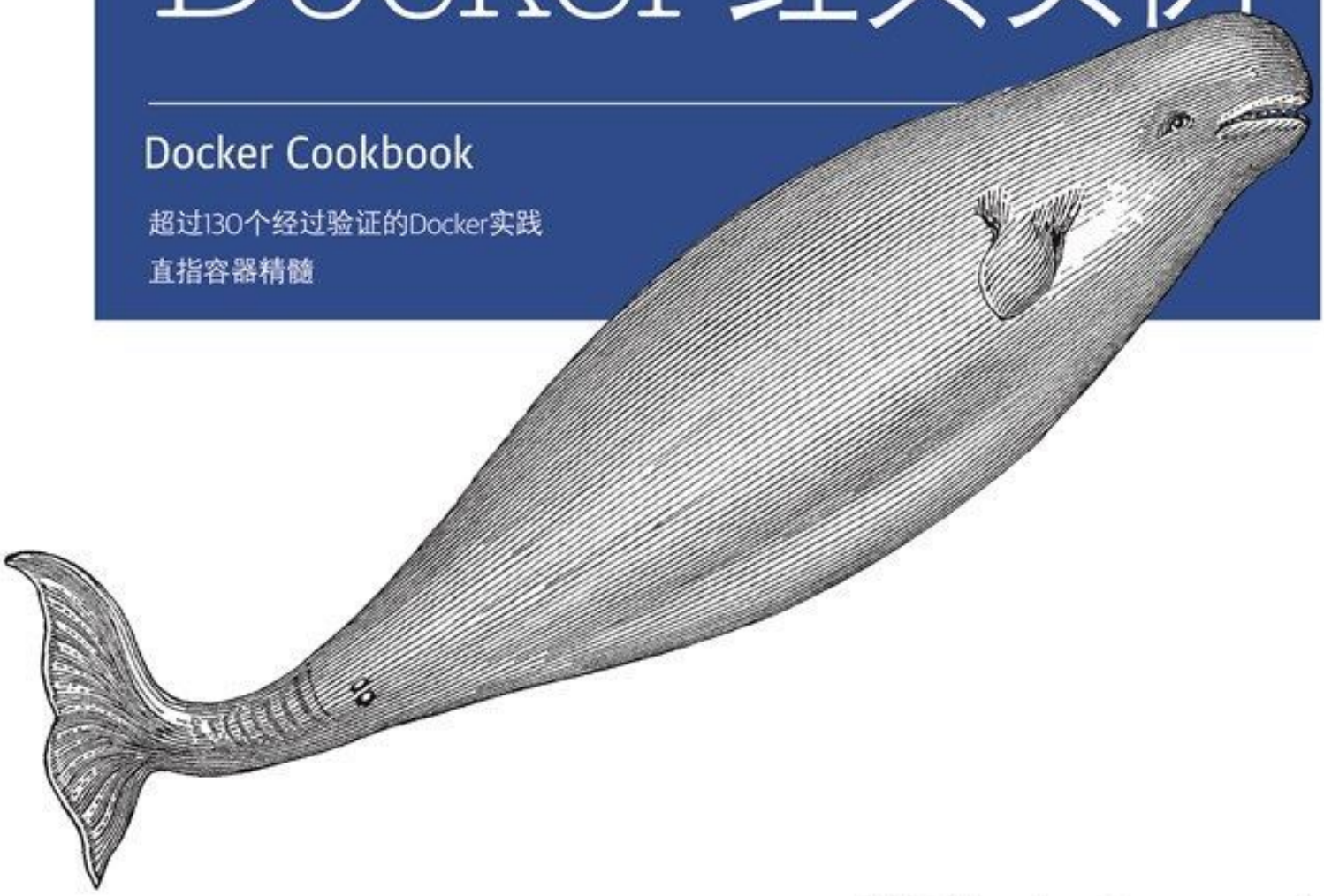
TURING 图灵程序设计丛书

Docker 经典实例

Docker Cookbook

超过130个经过验证的Docker实践

直指容器精髓



[美] Sébastien Goasguen 著

刘斌 译

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

版权信息

书名：Docker经典实例

作者：[美] Sébastien Goasguen

译者：刘斌

ISBN：978-7-115-44656-5

版权声明

O'Reilly Media, Inc.介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 *Make* 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。 O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过图书出版、在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——*Wired*

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野，并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去， Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

本书赞誉

开始使用 Docker 是一回事，真正领悟它的思想又是另一回事。我们需要对 Docker 有一个完整深入的理解。在为用户提供服务的应用程序中使用 Docker 时，这本书为我们带来了极大的帮助。

——Arjan Eriks，Schuberg Philis 公司云计算服务主管

这是一部完整实用的教程，涵盖了与 Docker 相关的各种工具和平台，并提供了具体和实用的例子。由于 Docker 的核心功能通过开放容器计划（Open Container Initiative）的努力逐渐成为事实上的行业标准，我们可以预想到，这个生态系统还会继续快速扩张。Sébastien 的这本书为从业者打下了坚实的基础，使得他们可以跟上这种快速变化的步伐。

——Chip Childers，Cloud Foundry 基金会技术副总裁

Sébastien 做了一项非常棒的工作，为初级用户集中介绍了各种 Docker 最佳实践和入门材料，涵盖了网络、镜像管理、配置以及包括 Kubernetes 和 Mesos/Marathon 在内的正在快速发展中的编排和调度生态系统。

——Patrick Reilly，Kismatic 公司 CEO

前言

写作缘由

我已经在云计算领域（主要是 IaaS 层）工作了 10 余年。Amazon AWS、Google GCE 和 Microsoft Azure 提供大规模的云计算服务已经有几年了，毫不夸张地说，访问一台服务器从未像现在这样方便、快速。对我来说，其真正的价值在于可以通过 API 来访问这些服务。我们现在可以通过编程来创建基础设施和部署应用。这些可编程层能够帮助我们达到更高级别的自动化，有利于企业更快地将产品推向市场，做出更多的创新，以及更好地为用户服务。

然而，尽管我们在配置管理和编排上耗费了大量的精力，但是对于在一个分布式环境中进行应用程序打包、配置和服务组装等方面依旧没有取得太大的进步。部署和运行一个可扩展、可容错的分布式应用程序仍然是比较困难的。

直到我试用 Docker 并明白了它为我们带来的可能性之后，我才成为 Docker 的疯狂粉丝。Docker 为 Linux 容器带来了全新的用户体验。它不是要提供与容器相对的完全虚拟化技术，而是要为应用程序的打包和运行提供便利。一旦你开始使用 Docker 并享受它所带来的全新体验，就会同时体会到另一个好处：你会开始思考如何进行合成和聚类分析。

容器帮助我们更多地思考如何进行**功能隔离**，这反过来又迫使我们在分布式环境中对应用进行解耦，然后再将其粘合在一起。

本书结构

本书共由 10 章构成。每章都基于 O'Reilly 标准的 cookbook 格式写成，由问题、解决方案和讨论等三部分组成。你可以按照顺序从前往后阅读，也可以选择特定的章节 / 范例。每个范例都是互相独立的，但如果一个范例引用了其他范例的概念，书中也会提供相应的参考引用。

- 第 1 章主要讨论与 Docker 安装相关的一些场景，包括使用 Docker Machine。然后会介绍 Docker 的一些基本命令，包括对容器进行管理、挂载数据卷、链接容器等。在该章最后，你将会得到一个可以工作的 Docker 主机，启动一些容器，还会对容器的生命周期有所了解。

- 第 2 章将会介绍 Dockerfile 和 Docker Hub，并展示如何构建镜像，以及对镜像进行打标签和提交操作。该章还将介绍如何运行自己的 Docker registry，并设置自动构建。学完该章，你将会掌握如何创建 Docker 镜像，通过公开或者私有的方式来共享镜像，以及构建持续交付工作流。

- 第 3 章将介绍 Docker 的网络机制。你将学到如何获得一个容器的 IP 地址，以及如何暴露主机端口上的容器服务。你还将学会如何把多个容器链接起来，以及如何使用定制的网络配置。该章也会通过一些范例来对容器网络进行深入剖析。对网络命名空间、使用 OVS 网桥、GRE 隧道等概念的介绍，也将为了解 Docker 容器网络奠定基础。最后，你还将了解更高级的网络配置和工具，比如 Weave、Flannel 以及目前处于实验阶段的 Docker Network 功能。

- 第 4 章将深入介绍如何对 Docker 守护进程进行配置，特别是与 Docker API 相关的安全设置和远程访问控制。该章也讨论了一些基本问题，如从源代码编译 Docker，运行 Docker 的测试用例，以及运行自己编译的 Docker 可执行程序。该章还有一些范例，对 Linux 上的命名空间和它们在容器中的使用情况进行了详细阐述。

- 第 5 章将会介绍 Google 新开发的容器管理平台。Kubernetes 提供了一种在分布式集群上部署多容器应用程序的方式。此外，它还提供了一种自动化公开服务和创建容器副本的方式。该章将会介绍如何在自己的基础设施上部署 Kubernetes：开始是使用本地 Vagrant 集群，随后是在云端启动的一组计算机上。之后，我们将介绍 Kubernetes 的核心内容：pod、service 和 replication controller。

- 第 6 章涵盖了四种新的经过优化的、专门为运行容器而生的 Linux 发行版本：CoreOS（https://coreos.com/ ）、Project Atomic（http://www.projectatomic.io/ ）、Ubuntu Core（http://www.ubuntu.com/cloud/tools/snappy ）和

RancherOS（http://rancher.com/rancher-os/ ）。这 些新的发行版本刚好为 Docker 容器的运行和编排提供了够用的操作系统。该章的范例包括如何安装和访问采用了这些发行版本的服务器。该章还会介绍这些发行版本所使用的对容器进行编排的工具，例如 `etcd` 、`fleet` 和 `systemd` 。

- 蓬勃发展的生态系统也是 Docker 的优势之一。第 7 章将会介绍过去 18 个月中出现的一些新工具。这些工具用于帮助 Docker 进行应用程序部署、持续集成、服务发现和编排。举个例子，你会读到关于 Docker Compose、Docker Swarm、Mesos、Rancher 和 Weavescope 的范例。

- Docker 守护进程可以安装在本地开发计算机上。然而，随着云计算技术的出现，我们可以按需创建服务器并立时使用。毫不夸张地说，大量基于容器的应用程序将会被部署到云中。第 8 章中的范例将会介绍如何访问位于 Amazon AWS（http://aws.amazon.com/ ）、Google GCE（https://cloud.google.com/compute/ ）和 Microsoft Azure（http://azure.microsoft.com/en-us/ ）之上的 Docker 主机。该章还将介绍两种使用了 Docker 的新云计算服务：AWS 弹性容器服务（Elastic Container Service，ECS）和 Google 容器引擎（Google Container Engine，https://cloud.google.com/container-engine/ ）。

- 第 9 章将会探讨容器使用过程中的应用程序监控问题。长久以来，基础设施和应用程序的监控和可视化一直是 DevOps 社区关注的重点。随着 Docker 作为一种开发和运维机制变得越来越普及，从其中所汲取的经验教训也需要应用于基于容器的应用程序。

- 第 10 章将介绍单主机和集群上的端到端应用的部署。虽然前面几章中已经介绍了一些基本的应用程序部署，但这一章将要介绍的范例更接近于生产部署配置。该章内容更深一些，也将促使你思考今后该如何设计更复杂的微服务。

你所需要了解的技术

这是一本中等难度的书，要求读者对一些开发和系统管理概念有最基本的理解。在深入学习本书之前，你可能需要了解以下内容。

- bash（Unix shell）

  这是 Linux 和 OS X 上默认的 Unix shell。读者最好熟悉 Unix shell，如编辑文件、设置文件权限、在文件系统中移动文件、管理用户权限，以及一些基本的 shell 编程知识。如果你对 Linux shell 不太熟悉，可以参考一下 Cameron Newham 的 *Learning the Bash Shell* ，或者 Carl Albing、JP Vossen 和 Cameron Newham 合著的 *Bash Cookbook* ，这两本书也都是由 O'Reilly 出版的。

- 软件包管理

本书中的工具往往需要通过安装一些软件包来满足多个依赖。因此这就要求读者对所用计算机系统上的软件包管理程序有所了解。这可能是 Ubuntu/Debian 系统上的 apt，或是 CentOS/RHEL 系统上的 yum，又或是 OS X 上的 port 或者 brew。不管你使用的是什么软件包管理器，请确保你知道如何安装、升级和删除软件包。

- Git

  Git 已成为分布式版本控制领域的标准。如果你已经熟悉 CVS 和 SVN，但还没有使用过 Git，那你真应该开始使用 Git。Jon Loeliger 和 Matthew McCullough 合著的《Git 版本控制管理（第 2 版）》是很好的入门教材。如果你想使用 Git，还想托管自己的代码仓库，那么 GitHub 网站（http://github.com/ ）是一个很好的资源。要想学习 GitHub，可以访问 http://training.github.com 以及相关的交互式教程（http://try.github.io/ ）。

- Python

  除了 C/C++ 或 Java 编程语言，我总是鼓励学生选择一门脚本语言。Perl 曾经统治了世界，然而现如今，Ruby 和 Go 语言则更加普及。我本人使用 Python，本书中的大多数例子将使用 Python 来编写，但也有几个例子使用了 Ruby，甚至还有一个例子使用了 Clojure。O'Reilly 出版了很多 Python 方面的图书，包括 Bill Lubanovic 的《Python 语言及其应用》[1] 、Mark Lutz 的《Python 编程》，以及 David Beazley 和 Brian K. Jones 合著的《Python Cookbook 中文版》。

- Vagrant

  Vagrant 已经成为 DevOps 工程师建立和管理虚拟环境的优秀工具之一。它非常适合用于在本地测试和快速配置虚拟机，但我们也可以使用一些插件来通过 Vagrant 连接到公共云提供商。本书将采用 Vagrant 来快速部署一个虚拟机实例，使其像 Docker 主机一样运行。你也可能想阅读一下由 Vagrant 的开发者 Mitchell Hashimoto 本人写的 *Vagrant: Up and Running* 一书。

- Go

  Docker 采用 Go 语言编写。在过去的几年里，Go 语言已成为许多初创公司首选的新编程语言。本书不是教大家如何学习 Go 编程的，而是会介绍如何编译一些 Go 项目。我希望读者至少知道如何安装一个 Go 工作区。如果想了解更多，Introduction to Go Programming（http://shop.oreilly.com/product/0636920035305.do ）这个视频培训课程是一个很好的选择。

[1] 此书已由人民邮电出版社出版。——编者注

在线内容

本书中使用的示例代码、Vagrantfile 和其他脚本都可以从 GitHub 找到（https://github.com/how2dock/docbook ）。你可以克隆这个仓库，进入相应章节和范例并使用其中的代码。比如，下面的代码显示了如何通过 Vagrant 启动一台 Ubuntu 14.04 虚拟机并在其中安装 Docker。

```
$ git clone https://github.com/how2dock/docbook.git
$ cd dockbook/ch01/ubuntu14.04/
$ vagrant up
```

这个仓库中的示例都不是最优配置，仅足以运行范例中的示例。

排版约定

本书使用了下列排版约定。

- **楷体**

  表示新术语。

- 等宽字体（`constant width` ）

  表示程序片段，以及正文中出现的变量、函数名、数据库、数据类型、环境变量、语句和关键字等。

- 加粗等宽字体（ **`constant width bold`** ）

  表示应该由用户输入的命令或其他文本。

- 等宽斜体（ *`constant width italic`* ）

  表示应该由用户输入的值或根据上下文确定的值替换的文本。

该图标表示提示或建议。

该图标表示一般注记。

该图标表示警告或警示。

Safari® Books Online

Safari Books Online（http://www.safaribooksonline.com ）是应运而生的数字图书馆。它同时以图书和视频的形式出版世界顶级技术和商务作家的专业作品。技术专家、软件开发人员、Web 设计师、商务人士和创意专家等，在开展调研、解决问题、学习和认证培训时，都将 Safari Books Online 视作获取资料的首选渠道。

对于组织团体、政府机构和个人，Safari Books Online 提供各种产品组合和灵活的定价策略。用户可通过一个功能完备的数据库检索系统访问 O'Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、 Jones & Bartlett、Course Technology 以及其他几十家出版社的上千种图书、培训视频和正式出版之前的书稿。要了解 Safari Books Online 的更多信息，我们网上见。

联系我们

请把对本书的评价和问题发给出版社。

美国：

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）

奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网站地址是：

http://shop.oreilly.com/product/0636920036791.do

对于本书的评论和技术性问题，请发送电子邮件到：

bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：

http://www.oreilly.com

我们在 Facebook 的地址：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址：http://www.youtube.com/oreillymedia

致谢

我写这本书用了 8 个月的时间。在这段时间里，我阅读了无数关于 Docker 的博客和文档，并对一切进行了反复的测试。我要感谢我的妻子和孩子，他们允许我利用周末和晚上的时间来写作这本书。我还要感谢 Brian Anderson，他一直督促我、鼓励我，多亏他的定期检查，我才能准时完成目标。如果没有 Fintan Ryan、Eugene Yakubovich、Joe Beda 和 Pini Riznik 的补充，这本书也不会这么完整。他们四个人帮我完成了非常有价值的内容，这也将会帮到许多读者。非常感谢 Patrick Debois 和 John Willis 的早期审阅，他们提供了令人鼓舞的宝贵的反馈，这也使得这本书更完善。Ksenia Burlachenko 和 Carlos Sanchez 的仔细审阅帮助我解决了很多问题，这对所有读者都非常有帮助。非常感谢他们两个。特别感谢 Funs Kessen，他是一位网络技术和应用程序设计专家，从来没有拒绝回答我提出的许多愚蠢问题。最后，非常感谢本书早期发布版的读者，特别是 Olivier Boudry，感谢他们愿意阅读内容尚不完整，并且存在拼写错误、语法错误和内容错误的版本。没有他们的更正和评论，这本书也不会像现在这样好。

电子书

扫描如下二维码，即可购买本书电子版。

第 1 章　Docker 入门

1.0　简介

入门 Docker 很简单。Docker 的核心是一个被称作 Docker 引擎的基于单主机运行的守护进程，我们可以通过这个守护进程来创建和管理容器。在深入使用 Docker 之前，你需要先在一台主机（比如台式机、笔记本电脑或者服务器）上安装 Docker 引擎。

本章中的前几个范例将会介绍在服务器上运行 Docker 所需的安装步骤。官方文档差不多涵盖了所有操作系统，这里我们会对 Ubuntu 14.04（参见范例 1.1）、CentOS 6.5（参见范例 1.2）和 CentOS 7（参见范例 1.3）进行介绍。如果你想使用 Vagrant，可以参见范例 1.4。

我们也会以树莓派为例来介绍如何在 ARM CPU 上安装 Docker（参见范例 1.5）。如果是 Windows 或者 OS X 系统的主机，你可以使用 Docker Toolbox（参见范例 1.6）。Docker Toolbox 除了包括 Docker 引擎之外，还包括其他一些实用工具。Docker Toolbox 使用 VirtualBox 将一个虚拟机作为 Docker 主机运行，这个虚拟机也就是 Boot2Docker。现在已经不再推荐使用 Boot2Docker 了，不过我们还是会在范例 1.7 中介绍一下如何使用 Boot2Docker 安装 Docker。

在这些安装范例之后，我们会介绍一下 `docker-machine` 。这是一个 Docker 工具，可以通过它在公有云上创建云主机并安装 Docker，自动配置本地 Docker 客户端来使用远程 Docker 主机。范例 1.9 中会介绍如何在 DigitalOcean 云中使用 `docker-machine` 。

一旦在自己的环境中安装好了 Docker，就可以浏览一下创建和管理容器所需的基本命令。范例 1.11 将会展示运行容器的第一步，范例 1.13 将会带你了解一个容器的标准生命周期：创建、启动、停止、终止和移除。

介绍完这些基本概念之后，我们将会直接深入到 Dockerfile（参见范例 1.14）。Dockerfile 是一个用于描述如何构建容器镜像的文件。它是 Docker 中的一个核心概念，第 2 章会详细展开讨论，这里只介绍其最简单的用法。我们通过 Dockerfile 来学习一个更为复杂的例子，即运行 WordPress 服务。

首先，我们会从头开始构建一个 Docker 镜像，在单一容器中运行多个进程（参见范例 1.15）。Docker 会改变我们的应用程序设计思想，不再将所有一切打包在一起，而是创建多个独立的服务，然后将这些独立的服务连接起来。然而，这并不意味着一个容器中不能运行多个服务。使用 supervisord 可以在一个容器中运行多个服务，范例 1.15 将会讲述如何去做。但是 Docker 的强大之处在于通过组合服务来运行应用程序。因此，在范例 1.16 中，我们会介绍如何将单一容器示例拆分为两个容器，并使用容器链接进行互连。这将是你的第一个分布式应用程序，尽管它也只是运行在同一台主机之上。

我们在本章介绍的最后一个概念是数据管理。在容器中访问数据是一个关键组件。你可以通过它来加载配置变量或数据集，或在容器之间共享数据。我们将再次使用 WordPress 的例子，告诉你如何备份数据库（参见范例 1.17），如何将宿主机的数据挂载到容器中（参见范例 1.18），以及如何创建数据容器（参见范例 1.19）。

总之，在本章中，你将会快速学到如何在一台主机上安装 Docker 引擎，并运行一个由两个容器组成的 WordPress 网站。

1.1 在Ubuntu 14.04上安装Docker

1.1.1 问题

你想在 Ubuntu 14.04 上运行 Docker。

1.1.2 解决方案

在 Ubuntu 14.04 上，可以通过至多三条 bash 命令来安装 Docker。Docker 项目推荐的安装方式是从网上下载并运行一个 bash 脚本。需要注意的是，在 Ubuntu 的软件包仓库中已经有一个 docker 软件包，不过这个软件与 Docker（http://www.docker.com ）并没有任何关系。执行推荐的安装，如下所示。

```
$ sudo apt-get update
$ sudo apt-get install -y wget
$ sudo wget -qO- https://get.docker.com/ | sh
```

可以通过查看 Docker 软件的版本来确认是否已正确安装了 Docker，如下所示。

```
$ sudo docker --version
Docker version 1.7.1, build 786b29d
```

可以停止、启动和重启 Docker 服务。比如，可以像下面这样重启 Docker 服务。

```
$ sudo service docker restart
```

如果你想直接以一个非 root 用户的身份来运行 docker 命令，可以将该用户添加到 docker 用户组，如下所示。

```
$ sudo gpasswd -a <user> docker
```

退出当前 shell 然后重新登录，或者重新启动一个新 shell，就能使用上面的配置了。

1.1.3 讨论

你可以按照 https://get.docker.com 的安装脚本，一步一步地手动安装或者进行自定义安装。在 Ubuntu 14.04（代号 trusty）上，最精简的安装步骤如下所示。

```
$ sudo apt-get update
$ sudo apt-get install -y linux-image-extra-$(uname -r) linux-image-extra-virtual
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80
                   --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
$ sudo su
# echo deb https://apt.dockerproject.org/repo ubuntu-trusty main > \
/etc/apt/sources.list.d/docker.list
# apt-get -y install docker-engine
```

1.1.4 参考

- 如果你想在其他操作系统上安装 Docker，请参考官方的安装文档（https://docs.docker.com/docker/installation/ ）。

1.2 在CentOS 6.5上安装Docker

1.2.1 问题

你想在 CentOS 6.5 上安装 Docker。

1.2.2 解决方案

在 CentOS 6.5 上，可以通过添加 EPEL（Extra Packages for Enterprise Linux）仓库使用 docker-io 包安装 Docker，如下所示。

```
$ sudo yum -y update
$ sudo yum -y install epel-release
$ sudo yum -y install docker-io
$ sudo service docker start
$ sudo chkconfig docker on
```

在 CentOS 6.5 上，Docker 的版本应该是 1.6.2，如下所示。

```
# docker --version
Docker version 1.6.2, build 7c8fca2/1.6.2
```

1.2.3 讨论

Docker 将不会继续提供对 CentOS 6.*x* 的支持。如果你想使用最新版 Docker，那么应该选择 CentOS 7（参见范例 1.3）。

1.3 在CentOS 7上安装Docker

1.3.1 问题

你想在 CentOS 7 上使用 Docker。

1.3.2 解决方案

通过 yum 管理器安装 Docker 软件包。CentOS 采用了 systemd，因此你需要使用 `systemctl` 命令来管理 docker 服务，如下所示。

```
$ sudo yum update
$ sudo yum -y install docker
$ sudo systemctl start docker
```

也可以使用 Docker 官方的安装脚本来安装，这也会使用 Docker 仓库中的软件包，如下所示。

```
$ sudo yum update
$ sudo curl -sSL https://get.docker.com/ | sh
```

1.4 使用Vagrant创建本地Docker主机

1.4.1 问题

你的本地计算机的操作系统和你想要运行 Docker 的操作系统不同。比如，你现在运行的是 OS X，但是你想在 Ubuntu 上运行 Docker。

1.4.2 解决方案

在本地通过 Vagrant（http://vagrantup.com ）启动一个虚拟机（virtual machine，VM），并使用 Vagrantfile 中的 shell 配置程序初始化 VM。

如果已经安装了 VirtualBox（http://virtualbox.org ）和 Vagrant（http://vagrantup.com ），那么你只需要创建一个名为 Vagrantfile 的文本文件，其内容如下所示。

```
VAGRANTFILE_API_VERSION = "2"

$bootstrap=<<SCRIPT
apt-get update
apt-get -y install wget
wget -qO- https://get.docker.com/ | sh
gpasswd -a vagrant docker
service docker restart
SCRIPT

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "ubuntu/trusty64"

  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |vb|
     vb.customize ["modifyvm", :id, "--memory", "1024"]
  end

  config.vm.provision :shell, inline: $bootstrap

end
```

之后就可以启动虚拟机了。Vagrant 将会从 Vagrant cloud [http://vagrantcloud.com ，现在是 Atlas（https://atlas.hashicorp.com ）的一部分] 下载 ubuntu/trusty64 镜像（box），通过 VirtualBox 创建该镜像的一个实例，再运行在 Vagrantfile 中定义的初始化脚本。这个虚拟机实例将会有 1GB 的内存和两个网络接口：一个用于与外部网络进行通信的**网络地址转换**（Network Address Translation，NAT）接口，一个本地网络接口 192.168.33.10。虚拟机启动后，就可以通过 ssh 来登录到虚拟机并使用 Docker。

```
$ vagrant up
$ vagrant ssh
vagrant@vagrant-ubuntu-trusty-64:~$ docker ps
CONTAINER ID    IMAGE     COMMAND     CREATED     STATUS     PORTS     NAMES
```

*