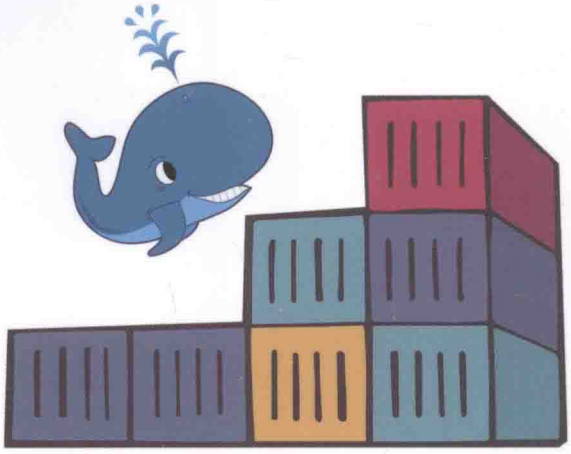
HZ BOOKS 华章IT

HUAWEI

容器技术系列



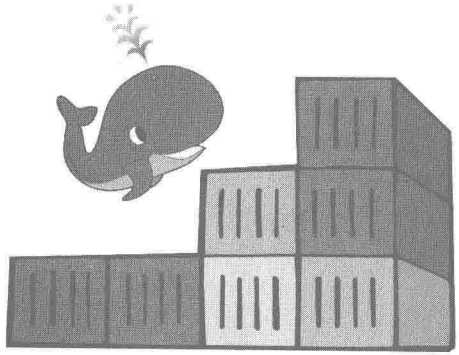
DOCKER PRO

Docker 进阶与实战

华为Docker实践小组 著

- 作者团队为华为一线开发者和Docker社区活跃的贡献者，在Docker社区贡献中，国内排名第一。
- 以功能模块为粒度，对每一个重要的模块单独进行深入的分析和讲解，力求将“代码与产品，理论与实践”完美结合。

机械工业出版社
China Machine Press



DOCKER PRO

Docker进阶与实战

华为Docker实践小组 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Docker 进阶与实战 / 华为 Docker 实践小组著 . —北京：机械工业出版社，2016.2
（容器技术系列）

ISBN 978-7-111-52339-0

I. D… II. 华… III. Linux 操作系统 - 程序设计 IV. TP316.89

中国版本图书馆 CIP 数据核字（2015）第 303557 号

Docker 进阶与实战

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：余 洁 责任校对：董纪丽
印 刷：三河市宏图印务有限公司 版 次：2016 年 2 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：16.5
书 号：ISBN 978-7-111-52339-0 定 价：79.00 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

Forword 序

我们这个团队的主业是操作系统内核开发。“太阳底下没有新鲜事”，这句话对于操作系统来说，有着深刻的意义。一个爆红的技术，寻根溯源，你会发现它往往已经在操作系统里潜伏很久。这种例子俯拾皆是。

虚拟化技术的源头可以追溯到20世纪70年代初期IBM的S370，但直到2003年的SOSP会议上一篇关于虚拟化的论文《Xen and the Art of Virtualization》引起广泛关注之后，虚拟化才走上发展的快车道。在软件领域，虚拟化技术把VMware打造成400亿美元量级的行业明星，又在硬件领域搅动了CPU、网络、存储等各个市场，迫使市场上的行业领袖做出相应的创新。现在，计算虚拟化、网络虚拟化、存储虚拟化这些概念已经深入人心。

而容器技术也不是全新的概念，系统容器最早可以追溯到20世纪80年代初期的chroot；打着轻量级虚拟化旗号的商用软件也是在21世纪之初由Virtuozzo提出的。但当时这个技术只是在系统管理员的小圈子里口耳相传，不愠不火地发展着。直到2013年，有一家叫作dotCloud的小公司开源了一个叫Docker的小项目……

若将Docker的核心技术层层剥离开来分析，作为操作系统开发人员，我们是无法理解Docker为什么会爆发成为行业里的新星的。因为严格来说，Docker用的所有关键技术都早已存在：

- Cgroup (Control Group)是Google在2006年启动开发的，算起来也有将近10年的历史了。
- 对于Namespace，从最早的Mount namespace算起，不断迭代到今天，已成为包括UTS（系统标识）、IPC（进程间通信）、PID（进程标识）、Network（网络设备、IP地址以及路由表）、User（用户标识）等的技术，可谓洋洋大观。
- Aufs的历史可以追溯到1993年的Inheriting File System，虽然Aufs没有进入Linux主线，但也已经在Debian、Gentoo这样的主流发行版中得到应用。

这些“大叔辈”的技术，通过Docker引擎的组合，焕发出“小鲜肉”的吸引力。而从另一个方面看，那些在技术和理念上更先进的项目，比如OSv，反而远没有得到这种众星

捧月般的待遇。

为什么会这样？这个疑问促使我们摘下操作系统开发人员的帽子，带上系统运维人员的帽子，带上应用开发者的帽子，换个角度审视自己从前的工作。

在这个角色转换的过程中，我们得到了很多的收获：

首先，我们代表国内的技术人为 Docker 社区做出了一些贡献，此为收获一。

因为换了一个角度，对这个技术兴起背后的原因有了更深刻的理解，此为收获二。

利用工作之余，将技术经验转化为文字，把容器技术传播给更广泛的受众，此为收获三。

如果读者在阅读本书和实践后，不仅知其然，而且知其所以然，并与我们一同把容器技术的发展推向下一个阶段，那可以算是最大的收获了。

是以为序！

华为 2012 实验室 操作系统专家　胡欣蔚

2015 年 11 月

Preface 前　言

为什么要写这本书

在计算机技术日新月异的今天，Docker 也算是其中异常璀璨的一员了。它的生态圈涉及内核、操作系统、虚拟化、云计算、DevOps 等热门领域，受众群体也在不断扩大。

Docker 在国内的发展如火如荼，短短一两年时间里就陆续出现了一批关于 Docker 的创业公司。华为公司作为国内开源领域的领导者，对 Docker 也有很大的投入，我们认为有必要把自己的知识积累和实践经验总结出来分享给广大开发者。除了吸引更多的人投入到 Docker 的生态建设以外，我们也希望通过本书帮助更多的读者更好、更快地掌握 Docker 关键技术。

关于本书

目前市场已经有一些不错的 Docker 入门图书，但多侧重于入门和具体的应用，本书会介绍一些 Docker 关键技术原理和高级使用技巧，适合有一定基础的读者。另外，本书会对 Docker 涉及的各个模块、关系和原理进行系统梳理，帮助读者对 Docker 加深认识，更好地应用 Docker 部署生产环境，最大程度安全有效地发挥 Docker 的价值。

本书不仅适合一般的 Docker 用户，也适合 Docker 生态圈中的开发者，希望它可以成为一本 Docker 进阶的图书，帮助读者快速提升。

本书是由华为整个 Docker 团队合作完成的，笔者包括（排名不分先后）：邓广兴、胡科平、胡欣蔚、黄强、雷继棠、李泽帆、凌发科、刘华、孙远、谢可杨、杨书奎、张伟、张文涛、邹钰。

本书的内容

本书的定位是有一定 Docker 基础的读者，所以在基本的概念和使用上，我们不会花过

多的篇幅讲解，而是给出相应有价值的链接，作为读者的延伸阅读。

在内容上，除了对 Docker 进行系统的梳理外，同时还会对 Docker 背后的核心技术（即容器技术）及其历史进行介绍，进一步帮助读者更好地理解 Docker。

章节划分则以功能模块为粒度，对每一个重要的模块进行了深入分析和讲解，同时也为热门领域单独开辟了章节。在每一章的最后都会讲解一些高级用法、使用技巧或实际应用中遇到的问题。虽然各章节的内容相对独立，但也会有一些穿插的介绍和补充，以帮助读者融会贯通，系统深入地理解 Docker 的每一个细节。

另外，本书的笔者都是一线的开发者和 Docker 社区活跃的贡献者，因此书中还专门准备了一个章节来介绍参与 Docker 开发的流程和经验。同时，伴随 Docker 的发展，Docker 生态圈也在不断扩大并吸引了越来越多的人的关注。Docker 集群管理和生态圈的介绍也将作为本书重点章节详细讲解。此外，Docker 测试也是比较有特色的内容，分享了笔者在测试方面的经验。最后，附录中所包含的常用的 Docker 相关信息，可供读者需要时查询。

本书的内容和代码都是基于 Docker 1.8 版本的。在代码示例中，使用“#”开头的命令表示以 root 用户执行，以“$”开头的命令表示以普通用户执行。

勘误和支持

由于笔者水平有限，编写的时间也很仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。读者可以把书中发现的问题或建议发送到邮箱 docker@huawei.com，我们会尽快回复大家的疑问，并把收集的信息整理修正。

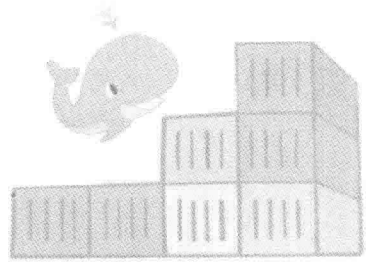
致谢

本书是由整个 Docker 团队协作完成的，由于繁忙的工作书稿撰写几度中止。感谢我们的项目经理裴斐月女士，正是她的整体协调和督促，以及与出版社的大量沟通，才促成了本书的出版。感谢李泽帆，他不仅参与了本书的写作，而且承担了全书的审读工作，给出了大量有价值的建议。还要感谢 Stephen Li、陈佳波、杨开封、胡欣蔚和张殿芳，以及其他华为公司主管对我们写书的大力支持，感谢机械工业出版社的编辑耐心专业的指导和审核。最后，感谢我们每一位家人的支持陪伴，我们的工作因为有了家人的支持和期待才变得更有意义。

华为 Docker 实践小组

2015 年 11 月

Contents 目　录



第 1 章 Chapter 1

Docker 简介

1.1 引言

1.1.1 Docker 的历史和发展

自从 2013 年年初一个叫 dotCloud 的 PaaS 服务供应商将一个内部项目 Docker 开源之后，这个名字在短短几年内就迅速成为一个热词。似乎一夜之间，人人都开始谈论 Docker，以至于这家公司干脆出售了其所持有的 PaaS 平台业务，并且改名为 Docker.Inc，从而专注于 Docker 的开发和推广。

对于 Docker，目前的定义是一个开源的容器引擎，可以方便地对容器（关于容器，将在第 2 章详细介绍）进行管理。其对镜像的打包封装，以及引入的 Docker Registry 对镜像的统一管理，构建了方便快捷的“Build, Ship and Run”流程，它可以统一整个开发、测试和部署的环境和流程，极大地减少运维成本。另外，得益于容器技术带来的轻量级虚拟化，以及 Docker 在分层镜像应用上的创新，Docker 在磁盘占用、性能和效率方面相较于传统的虚拟化都有非常明显的提高，所以理所当然，Docker 开始不断蚕食传统虚拟化的市场。

随着 Docker 技术的迅速普及，Docker 公司持续进行融资，并且其估值也在不断攀升，同时，Docker 公司也在不断地完善 Docker 生态圈，这一切使得 Docker 正慢慢成为轻量级虚拟化的代名词。在可预见的未来，很可能需要不断地刷新对 Docker 的定义。

目前 Docker 已加入 Linux 基金会，遵循 Apache 2.0 协议，其代码托管于 [Github]（https://github.com/docker/docker）。

1.1.2 Docker 的架构介绍

要了解 Docker，首先要看看它的架构图，如图 1-1 所示。

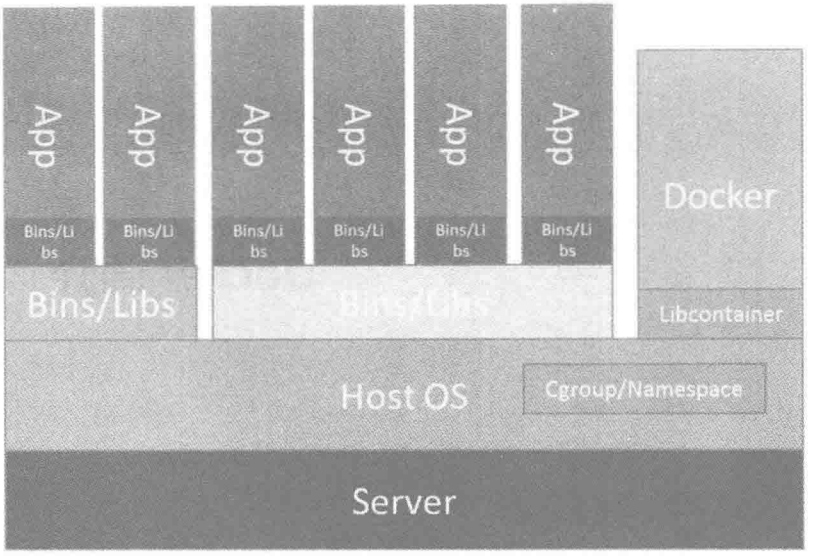


图 1-1 Docker 架构图

从图 1-1 可知，Docker 并没有传统虚拟化中的 Hypervisor 层。因为 Docker 是基于容器技术的轻量级虚拟化，相对于传统的虚拟化技术，省去了 Hypervisor 层的开销，而且其虚拟化技术是基于内核的 Cgroup 和 Namespace 技术，处理逻辑与内核深度融合，所以在很多方面，它的性能与物理机非常接近。

在通信上，Docker 并不会直接与内核交互，它是通过一个更底层的工具 Libcontainer 与内核交互的。Libcontainer 是真正意义上的容器引擎，它通过 clone 系统调用直接创建容器，通过 pivot_root 系统调用进入容器，且通过直接操作 cgroupfs 文件实现对资源的管控，而 Docker 本身则侧重于处理更上层的业务。

Libcontainer 的详细介绍可参见第 9 章。

Docker 的另一个优势是对层级镜像的创新应用，即不同的容器可以共享底层的只读镜像，通过写入自己特有的内容后添加新的镜像层，新增的镜像层和下层的镜像一起又可以作为基础镜像被更上层的镜像使用。这种特性可以极大地提高磁盘利用率，所以当你的系统上有 10 个大小为 1GB 的镜像时，它们总共占用的空间大小可能只有 5GB，甚至更少。另外，Docker 对 Union mount 的应用还体现在多个容器使用同一个基础镜像时，可极大地减少内存占用等方面，因为不同的容器访问同一个文件时，只会占用一份内存。当然这需要使用支持 Union mount 的文件系统作为存储的 Graph Driver，比如 AUFS 和 Overlay。

1.2　功能和组件

Docker 为了实现其所描述的酷炫功能，引入了以下核心概念：

- Docker 客户端
- Docker daemon
- Docker 容器
- Docker 镜像
- Registry

下面就分别来简单地介绍一下。

1.2.1　Docker 客户端

Docker 是一个典型的 C/S 架构的应用程序，但在发布上，Docker 将客户端和服务器端统一在同一个二进制文件中，不过，这只是对于 Linux 系统而言的，在其他平台如 Mac 上，Docker 只提供了客户端。

Docker 客户端一般通过 Docker command 来发起请求，另外，也可以通过 Docker 提供的一整套 RESTful API 来发起请求，这种方式更多地被应用在应用程序的代码中。

1.2.2　Docker daemon

Docker daemon 也可以被理解成 Docker Server，另外，人们也常常用 Docker Engine 来直接描述它，因为这实际上就是驱动整个 Docker 功能的核心引擎。

简单地说，Docker daemon 实现的功能就是接收客户端发来的请求，并实现请求所要求的功能，同时针对请求返回相应的结果。在功能的实现上，因为涉及了容器、镜像、存储等多方面的内容，daemon 内部的机制会复杂很多，涉及了多个模块的实现和交互。

1.2.3　Docker 容器

在 Docker 的功能和概念中，容器是一个核心内容，相对于传统虚拟化，它作为一项基础技术在性能上给 Docker 带来了极大优势。

在功能上，Docker 通过 Libcontainer 实现对容器生命周期的管理、信息的设置和查询，以及监控和通信等功能。而容器也是对镜像的完美诠释，容器以镜像为基础，同时又为镜像提供了一个标准的和隔离的执行环境。

在概念上，容器则很好地诠释了 Docker 集装箱的理念，集装箱可以存放任何货物，可以通过邮轮将货物运输到世界各地。运输集装箱的邮轮和装载卸载集装箱的码头都不用关心集装箱里的货物，这是一种标准的集装和运输方式。类似的，Docker 的容器就是“软件界的集装箱”，它可以安装任意的软件和库文件，做任意的运行环境配置。开发及运维人员在转移和部署应用的时候，不用关心容器里装了什么软件，也不用了解它们是如何配置的。

而管理容器的 Docker 引擎同样不关心容器里的内容，它只要像码头工人一样让这个容器运行起来就可以了，就像所有其他容器那样。

容器不是一个新的概念，但是 Docker 在对容器进行封装后，与集装箱的概念对应起来了，它之所以被称为“软件界的创新和革命”，是因为它会改变软件的开发、部署形态，降低成本，提高效率。Docker 真正把容器推广到了全世界。

1.2.4 Docker 镜像

与容器相对应，如果说容器提供了一个完整的、隔离的运行环境，那么镜像则是这个运行环境的静态体现，是一个还没有运行起来的“运行环境”。

相对于传统虚拟化中的 ISO 镜像，Docker 镜像要轻量化很多，它只是一个可定制的 rootfs。Docker 镜像的另一个创新是它是层级的并且是可复用的，这在实际应用场景中极为有用，多数基于相同发行版的镜像，在大多数文件的内容上都是一样的，基于此，当然会希望可以复用它们，而 Docker 做到了。在此类应用场景中，利用 Unionfs 的特性，Docker 会极大地减少磁盘和内存的开销。

Docker 镜像通常是通过 Dockerfile 来创建的，Dockerfile 提供了镜像内容的定制，同时也体现了层级关系的建立。另外 Docker 镜像也可以通过使用 docker commit 这样的命令来手动将修改后的内容生成镜像，这些都将在后续的章节详细介绍。

1.2.5 Registry

在前面提到的镜像中，曾指出 Docker 通过容器集装箱可以很方便地转运软件，其实，Registry 也在其中扮演了重要的角色。

Registry 是一个存放镜像的仓库，它通常被部署在互联网服务器或者云端。通常，集装箱是需要通过邮轮经行海洋运输到世界各地的，而互联网时代的传输则要方便很多，在镜像的传输过程中，Registry 就是这个传输的重要中转站。假如我们在公司将一个软件的运行环境制作成镜像，并上传到 Registry 中，这时就可以很方便地在家里的笔记本上，或者在客户的生产环境上直接从 Registry 上下载并运行了。当然，对 Registry 的操作也是与 Docker 完美融合的，用户甚至不需要知道 Registry 的存在，只需要通过简单的 Docker 命令就可以实现上面的操作。

Docker 公司提供的官方 Registry 叫 Docker Hub，这上面提供了大多数常用软件和发行版的官方镜像，还有无数个人用户提供的个人镜像。其实，Registry 本身也是一个单独的开源项目，任何人都可以下载后自己部署一个 Registry。因为免费的 Docker Hub 功能相对简单，所以多数企业会选择自己部署 Docker Registry 后二次开发，或者购买功能更强大的企业版 Docker Hub。

1.3　安装和使用

1.3.1　Docker 的安装

Docker 的安装和使用有一些前提条件，主要体现在体系架构和内核的支持上。对于体系架构，除了 Docker 一开始就支持的 x86-64，其他体系架构的支持则一直在不断地完善和推进中，用户在安装前需要到 Docker 官方网站查看最新的支持情况。对于内核，目前官方的建议是 3.10 以上的版本，除了内核版本以外，Docker 对于内核支持的功能，即内核的配置选项也有一定的要求（比如必须开启 Cgroup 和 Namespace 相关选项，以及其他的网络和存储驱动等）。如果你使用的是主流的发行版，那通常它们都已经打开了，如果使用的是定制化的内核，Docker 源码中提供了一个检测脚本（目前的路径是 ./contrib/check-config.sh）来检测和指导内核的配置。

在满足前提条件后，安装就非常的简单了，对于多数主流的发行版，通常只需要一条简单的命令即可完成安装，比如在 Ubuntu 下，可以使用如下命令安装：

```
$ sudo apt-get install docker.io
```

当然，实际情况可能会相对复杂些，比如，虽然 Ubuntu 中通常自带了 Docker，但用户常常需要使用最新版本的 Docker，以至于不得不对其进行升级。对于安装和升级，以及不同发行版上的操作方法，官方网站上提供了更加详细的说明，本书不做过多的赘述，下面的链接给出了常用发行版的安装方法：

- [Ubuntu](http://docs.docker.com/installation/ubuntulinux/)
- [Fedora](http://docs.docker.com/installation/fedora/)
- [Debian](http://docs.docker.com/installation/debian/)
- [Centos](http://docs.docker.com/installation/centos/)
- [Gentoo](http://docs.docker.com/installation/gentoolinux/)
- [Arch Linux](http://docs.docker.com/installation/archlinux/)
- [Windows](http://docs.docker.com/installation/windows/)
- [Mac OS X](http://docs.docker.com/installation/mac/)

另外，用户也可以直接获取 Docker binary 来运行，http://docs.docker.com/installation/binaries/ 网址介绍了获取的方法。虽然这样更简单，但还是推荐使用完整安装的方式，因为通过软件包安装的 Docker，除了有可执行文件之外，还包括了 Shell 自动完成脚本、man 手册、服务运行和配置脚本等内容，可以帮助用户更好地配置和使用 Docker。

Docker 还有一些其他更方便的安装方式，这将在后面的章节中详细介绍。