Docker进阶与实战

作者：华为Docker实践小组 著

容器技术系列

Docker进阶与实战

华为Docker实践小组　著

ISBN：978-7-111-52339-0

本书纸版由机械工业出版社于2016年出版，电子版由华章分社（北京华章图文信息有限公司，北京奥维博世图书发行有限公司）全球范围内制作与发行。

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @华章数媒

微信公众号 华章电子书（微信号：hzebook）

目录

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

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

第1章　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。

提示

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。