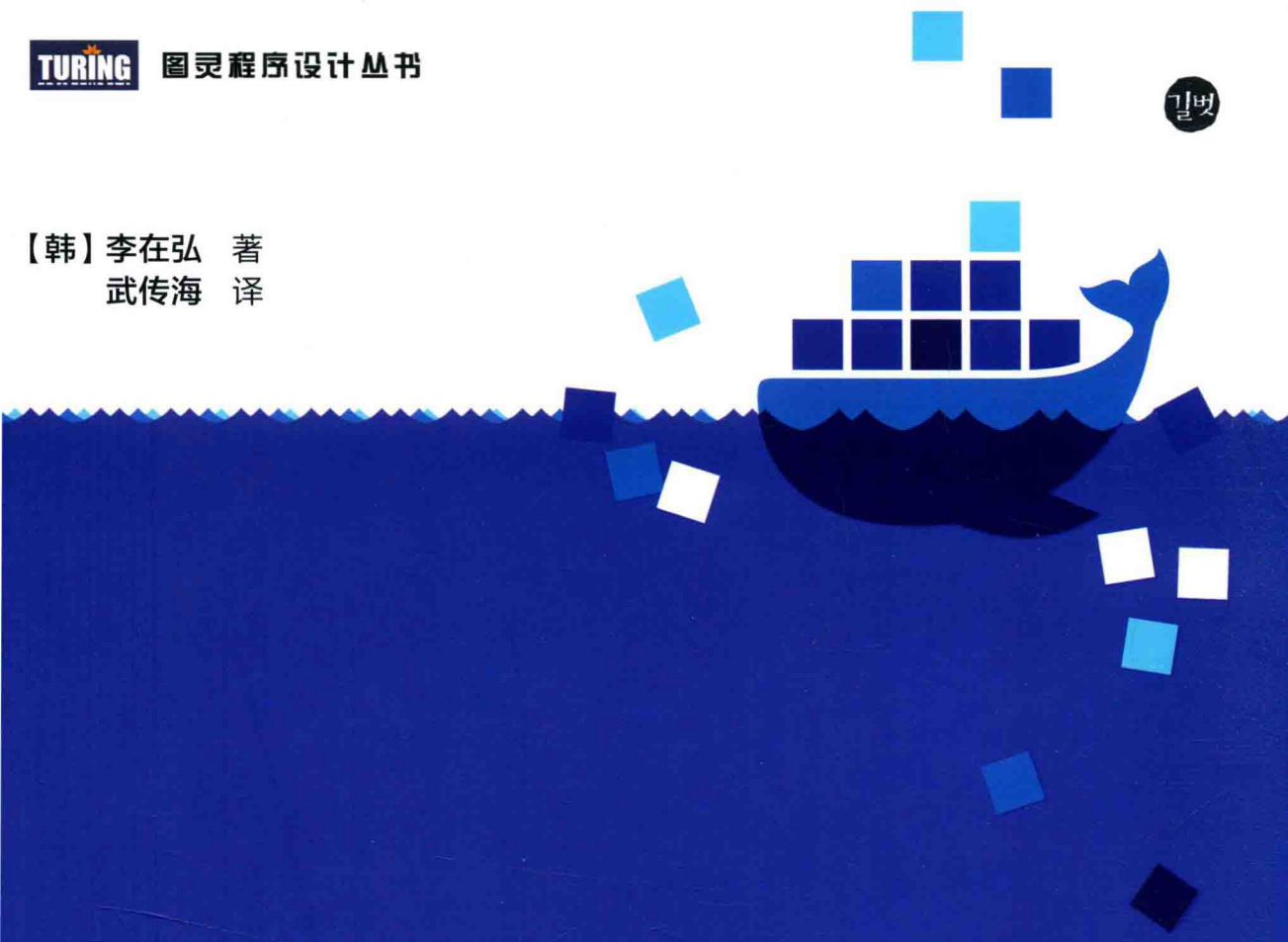
TURING 图灵程序设计丛书

【韩】李在弘 著
武传海 译

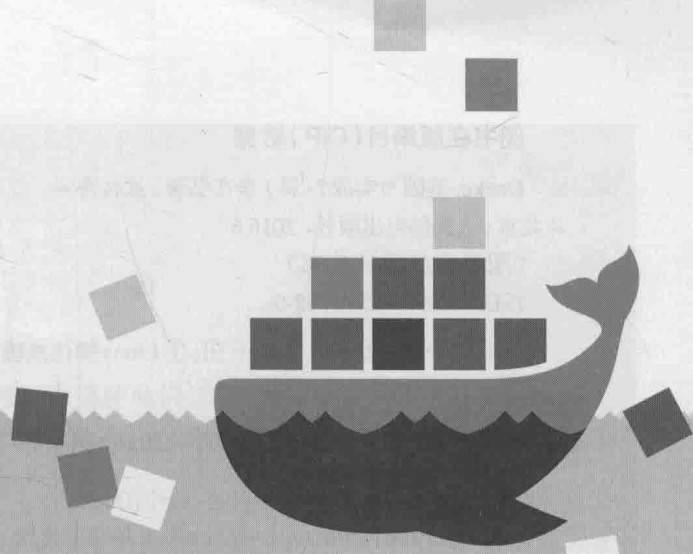


DOCKER FOR THE REALLY IMPATIENT

Docker 基础与实战

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

TURING 图灵程序设计丛书



Docker 基础与实战

DOCKER FOR THE REALLY IMPATIENT

【韩】李在弘 著
武传海 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Docker基础与实战 /（韩）李在弘著；武传海译
. -- 北京：人民邮电出版社，2016.6
（图灵程序设计丛书）
ISBN 978-7-115-41962-0

Ⅰ. ①D… Ⅱ. ①李… ②武… Ⅲ. ①Linux操作系统
—程序设计 Ⅳ. ①TP316.89

中国版本图书馆CIP数据核字（2016）第049472号

内 容 提 要

本书从Docker基础理论出发，更侧重实际业务中的技术与应用。重点在于后半部分在Amazon EC2、Google Colud Platform等平台上的使用方法，以及Rails与Django应用程序构建方法等，都是能够直接运用于实操的技术点。本书是利用Docker构建开发系统、测试系统、操作系统的优秀指南，非常适合一线开发人员。

◆ 著　[韩] 李在弘
译　武传海
责任编辑　陈　曦
责任印制　彭志环
◆ 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
三河市海波印务有限公司印刷
◆ 开本：800×1000　1/16
印张：19.5
字数：423千字　2016年6月第1版
印数：1-3 000册　2016年6月河北第1次印刷
著作权合同登记号　图字：01-2015-1716号

定价：69.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第8052号

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

前言

与其他领域相比，开源开发环境的变化速度最快。随着 GitHub 的出现，贡献代码的门槛正变得越来越低。我们也可以灵活使用 Git 这一分布式版本管理系统对著名项目进行派生（fork），然后构建自己的项目。

服务器开发及运营环境中，用于隔离目录的 chroot 或在 Linux 内核级别实现容器的 LXC 技术出现已久，但未能得到广泛应用。此时，Docker 利用 GitHub 的共享模型构建了极为便利的平台。

与从 GitHub 派生项目一样，借助 Docker，用户也可以基于 Docker Hub 中的镜像创建并分享自己的镜像。包括主要开源项目的官方镜像在内，其他用户上传的镜像以及自己上传的镜像都可以在全世界范围内共享使用，令人十分惊叹。这种统一接口并将数据集中存放的做法将效率提高到超乎想象的地步。

Linux 与生俱来的局限性是可执行文件与库之间的兼容性问题。针对该问题，多个 Linux 发行版推出了固有的包系统，但仍然无法完美解决。Docker 将应用程序或服务中可运行状态的组合用容器捆绑在一起，再通过网络共享。使用 Docker 可以帮助 Linux 服务器管理员大大缩短在系统构建与管理上浪费的时间，也不会再烦恼编译安装后无法完全删除。

灵活使用 Docker 可以帮助用户脱离从属于特定云平台的环境，只要愿意，就可以从 Amazon Web Service 轻松迁移到 Google Cloud Platform 以及 Microsoft Azure 平台。使用 Docker 镜像无需逐一搭建 Linux 服务器，只要使用原有的 Docker 镜像即可。无论何时都可以轻松转向费用更低、条件更好的服务。

本书并未涵盖 Docker 的所有内容。Docker 目前仍在发展，新的应用方法层出不穷。Docker 并不是一款固定不变的产品，其与多种程序的不同组合会产生无穷无尽的应用方法。因此，与其介绍目前出现的所有应用方法，不如先熟练掌握 Docker 的基本使用方法，不断创建服务并深入探索。

Docker 将服务器开发与运营带入一片全新的天地，下面请随我感受 Docker 的无尽魅力吧！

李在弘

2014 年 11 月

目录

第 1 章 Docker 1

1.1 虚拟机与 Docker 3
- 1.1.1 虚拟机 4
- 1.1.2 Docker 5
- 1.1.3 Linux 容器 6

1.2 Docker 镜像与容器 8

第 2 章 安装 Docker 11

2.1 Linux 11
- 2.1.1 自动安装脚本 11
- 2.1.2 Ubuntu 11
- 2.1.3 RedHat Enterprise Linux、CentOS 12
- 2.1.4 使用最新二进制文件 12

2.2 Mac OS X 13

2.3 Windows 16

第 3 章 使用 Docker 23

3.1 使用 search 命令搜索镜像 23

3.2 使用 pull 命令下载镜像 25

3.3 使用 images 命令列出镜像目录 25

3.4 使用 run 命令创建容器 25

3.5 使用 ps 命令查看容器列表 26

3.6 使用 start 命令启动容器 26

3.7 使用 restart 命令重启容器 27

3.8 使用 attach 命令连接容器 27

3.9 使用 exec 命令从外部运行容器内的命令 27

3.10 使用 stop 命令终止容器 28

3.11 使用 rm 命令删除容器 28

3.12 使用 rmi 命令删除镜像 29

第 4 章 创建 Docker 镜像 31

4.1 熟悉 Bash 31

4.2 编写 Dockerfile 36
4.3 使用 build 命令创建镜像 37

第 5 章 查看 Docker 39

5.1 使用 history 命令查看镜像历史 39
5.2 使用 cp 命令复制文件 40
5.3 使用 commit 命令从容器的修改中创建镜像 40
5.4 使用 diff 命令检查容器文件的修改 40
5.5 使用 inspect 命令查看详细信息 41

第 6 章 灵活使用 Docker 43

6.1 搭建 Docker 私有仓库 43
6.1.1 存储镜像数据到本地 43
6.1.2 使用 push 命令上传镜像 44
6.1.3 存储镜像数据到 Amazon S3 45
6.1.4 使用默认认证 46
6.2 连接 Docker 的容器 52
6.3 连接到其他服务器的 Docker 容器 53
6.4 使用 Docker 数据卷 56
6.5 使用 Docker 数据卷容器 59
6.6 创建 Docker 基础镜像 60
6.6.1 创建 Ubuntu 基础镜像 60
6.6.2 创建 CentOS 基础镜像 61
6.6.3 创建空基础镜像 62
6.7 在 Docker 内运行 Docker 64

第 7 章 详细了解 Dockerfile 67

7.1 .dockerignore 68
7.2 FROM 68
7.3 MAINTAINER 69
7.4 RUN 69
7.5 CMD 70
7.6 ENTRYPOINT 71
7.7 EXPOSE 73
7.8 ENV 73
7.9 ADD 74
7.10 COPY 76
7.11 VOLUME 77
7.12 USER 77

7.13 WORKDIR 78
7.14 ONBUILD 79

第 8 章 使用 Docker 部署应用程序 81
8.1 向一台服务器部署应用程序 81
8.1.1 在开发者 PC 安装 Git 并创建仓库 82
8.1.2 在开发者 PC 中使用 Node.js 编写 Web 服务器 83
8.1.3 在开发者 PC 中编写 Dockerfile 文件 84
8.1.4 在开发者 PC 中生成 SSH 密钥 85
8.1.5 在服务器端安装 Git 并创建仓库 86
8.1.6 在服务器中安装 Docker 87
8.1.7 在服务器中安装 SSH 密钥 88
8.1.8 在服务器中安装 Git Hook 89
8.1.9 在开发者 PC 中推送源代码 90
8.2 向多台服务器部署应用程序 91
8.2.1 在开发者 PC 安装 Git 并创建仓库 92
8.2.2 在开发者 PC 中使用 Node.js 编写 Web 服务器 93
8.2.3 在开发者 PC 中编写 Dockerfile 文件 94
8.2.4 在开发者 PC 中生成 SSH 密钥 95
8.2.5 在部署服务器安装 Git 并创建仓库 96
8.2.6 在部署服务器中生成 SSH 密钥 97
8.2.7 在部署服务器中安装 Docker 98
8.2.8 在部署服务器中安装 Docker 注册服务器 99
8.2.9 在部署服务器中安装 SSH 密钥 100
8.2.10 在部署服务器中安装 Git Hook 101
8.2.11 在应用程序服务器中安装 Docker 103
8.2.12 在应用程序服务器中安装 SSH 密钥 104
8.2.13 在开发者 PC 中推送源代码 105

第 9 章 Docker 监控 107
9.1 编写监控服务器 Dockerfile 108
9.2 编写应用程序服务器 Dockerfile 111
9.3 在 Web 浏览器中查看图表 114

第 10 章 在 Amazon Web Services 中使用 Docker 117
10.1 在 Amazon EC2 中使用 Docker 117
10.2 在 AWS Elastic Beanstalk 中使用 Docker 119
10.2.1 在 AWS 控制台部署 Docker 应用程序 119
10.2.2 使用 Docker Hub 公开仓库镜像 129

10.2.3 使用 Docker Hub 私有仓库的镜像 131
10.2.4 使用 Git 部署 Elastic Beanstalk Docker 应用程序 139

第 11 章 在 Google Cloud Platform 中使用 Docker 145

11.1 安装 Goolge Cloud SDK 145
11.2 在 Compute Engine 中使用 Docker 147
11.3 在 Container Engine 中使用 Docker 148

第 12 章 使用 Docker Hub 151

12.1 加入 Docker Hub 151
12.2 使用 push 命令上传镜像 153
12.3 创建 Docker Hub 私有仓库 155
12.4 使用 Docker Hub Automated Build 157

第 13 章 使用 Docker Remote API 167

13.1 使用 Docker Remote API Python 库 169
13.1.1 创建并启动容器 169
13.1.2 创建镜像 173
13.1.3 显示容器列表 175
13.1.4 显示镜像列表 176
13.1.5 其他示例与函数 176
13.2 使用 Docker Remote API Python 库进行 HTTPS 通信 187
13.2.1 创建证书 187
13.2.2 使用 Python 库 191

第 14 章 使用 CoreOS 193

14.1 在 VirtualBox 中安装 CoreOS 196
使用 systemd 运行服务 205
14.2 使用 Vagrant 安装 CoreOS 206
14.3 使用 etcd 211
14.3.1 创建 etcd 键与目录 211
14.3.2 输出 etcd 键与目录列表 212
14.3.3 设置自动删除 etcd 键与目录 212
14.3.4 监视 etcd 键 213
14.3.5 etcd 其他命令 214
14.4 使用 fleet 214
14.4.1 输出 fleet 机器列表 215
14.4.2 使用 fleet 运行 Unit 215
14.4.3 输出 fleet Unit 列表 217

14.4.4 查看 fleet Unit 状态 217
14.4.5 测试 fleet 的自动恢复功能 218
14.4.6 使用 fleet 专用选项 219
14.4.7 灵活使用 fleet Unit 文件模板 222
14.4.8 灵活使用 fleet sidekick 模型 224
14.4.9 fleet 其他命令 227
14.5 在云服务中使用 CoreOS 227
14.5.1 在 Amazon EC2 中使用 CoreOS 227
14.5.2 在 Google Compute Engine 中使用 CoreOS 229

第 15 章 使用 Docker 搭建 WordPress 博客 231

15.1 编写 WordPress Dockerfile 文件 232
15.2 编写 MySQL 数据库 Dockerfile 文件 233
15.3 创建 WordPress 与数据库容器 236

第 16 章 使用 Docker 构建 Ruby on Rails 应用 237

16.1 安装 Ruby 与 Rails 238
16.2 编写 Rails Dockerfile 240
16.3 编写 PostgreSQL 数据库 Dockerfile 文件 245
16.4 创建 Rails 与数据库容器 247

第 17 章 使用 Docker 构建 Django 应用 249

17.1 安装 Django 250
17.2 编写 Django Dockerfile 文件 253
17.3 编写 Oracle 数据库 Dockerfile 文件 258
17.4 创建 Django 与数据库容器 261

第 18 章 Docker 应用案例 263

18.1 与负载平衡相关的自动伸缩 263
18.2 整合开发、测试、运营 264
18.3 轻松迁移服务 265
18.4 用于测试 267

第 19 章 Docker 命令与选项列表 269

19.1 attach 270
19.2 build 271
19.3 Commit 273
19.4 cp 273

19.5 create 274
19.6 diff 277
19.7 events 277
19.8 exec 278
19.9 export 280
19.10 history 280
19.11 images 281
19.12 import 281
19.13 info 282
19.14 inspect 283
19.15 kill 284
19.16 load 284
19.17 login 285
19.18 logout 286
19.19 logs 286
19.20 port 287
19.21 pause 287
19.22 ps 287
19.23 pull 288
19.24 push 289
19.25 restart 289
19.26 rm 289
19.27 rmi 290
19.28 run 291
19.29 save 296
19.30 search 297
19.31 start 297
19.32 stop 298
19.33 tag 298
19.34 top 299
19.35 unpause 299
19.36 version 300
19.37 wait 300

附录 编译 Docker 301

第1章 Docker

DOCKER

Docker是Docker公司（原dotCloud）2013年3月推出的开源容器项目，上市至今已有3年，在世界范围内拥有超高人气。

进入2010年，服务器市场急速向云环境转移。人们开始更多地租用虚拟服务器，只要缴纳一定租金即可，不需要购买实际的物理服务器。尤其在搭建物理服务器时，服务器硬件的购买及安装都需要耗费相当长的时间。但在云环境下，无论是1台还是1000台，只需单击几次即可轻松创建虚拟服务器。

创建虚拟服务器后，还要在其中安装各种软件，进行各种设置。如果只有一两台服务器，那么能够轻松进行设置；但随着服务器数量的增加，采用人工设置就难了。因此，在云环境中进行安装与部署存在很大困难。

Linux/Unix环境中，虽然可以借助沿用至今的shell脚本进行自动安装与设置，但这种方式存在一定局限性。使用shell脚本很难实现集中式管理功能和其他复杂功能。并且，Linux环境中需要安装很多应用程序，设置也比较复杂。特别是一些微小的设置可能会对操作系统与服务的稳定性产生影响。

此时出现了“不可变基础设施”（Immutable Infrastructure）这一概念，指的是主机OS与服务运行环境（服务器程序、源代码、已编译的二进制文件）分离，只设置一次运行环境，之后不发生变更（Immutable）。也就是说，将服务运行环境创建为镜像后，部署至各服务器运行。此时若更新服务，则运行环境本身不会发生变更，只要重新生成镜像并再次部署即可。就像云平台中对服务器“用过即扔”，不可变基础设施中的服务运行环境镜像也是用过一次后就扔掉。

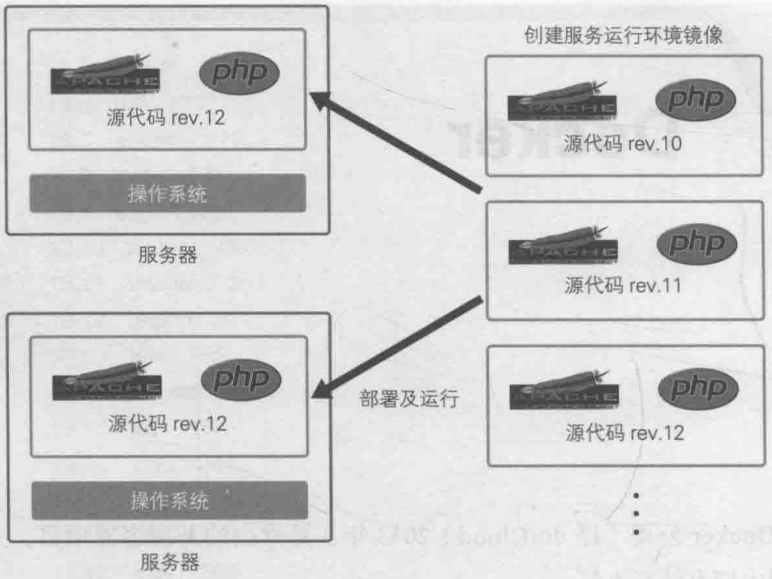


图 1-1 不可变基础设施

不可变基础设施拥有多种优点。

- 管理方便：由于服务运行环境以镜像形式存在，所以只要管理镜像本身即可。特别是可以集中管理镜像，实现系统部署与管理。此外，镜像生成设置也以文件形式存在，可以灵活用于版本管理系统。
- 扩展：可以利用一个镜像不断创建服务器。与云平台的自动伸缩功能（Auto Scaling）配合使用，能够轻松实现服务扩展。
- 测试：只要在开发人员 PC 或测试服务器中运行镜像，就可以搭建与实际服务运行环境一致的环境，非常容易测试。
- 轻量：分离操作系统与服务运行环境，实现轻量化，提供可以随时运行的环境。

Docker 项目实现了不可变基础设施，本书将详细讲解 Docker 有关内容。

从 Docker 图标与名称本身可以大致猜到 Docker 的功能——一头鲸鱼驮着多个集装箱——这很容易让人联想到服务器运行多个容器（镜像）的场景。

另外，这也意味着 Docker 不仅用于创建并运行镜像，还可以存储与部署（搬运）镜像。“Docker”一词的字典含义为港口（码头）上卸载集装箱的工人，与操纵容器的 Docker 功能类似。

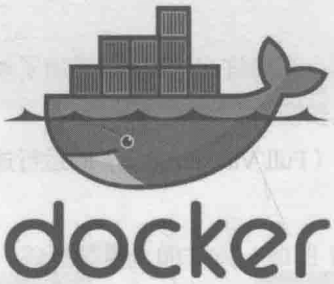


图 1-2 Docker 图标

如同用集装箱装载货物一样，将运行服务所需的所有“元素”全部集中到 Docker 容器之中。这些“元素”可以是常用开源软件，也可以是自己编写的程序。

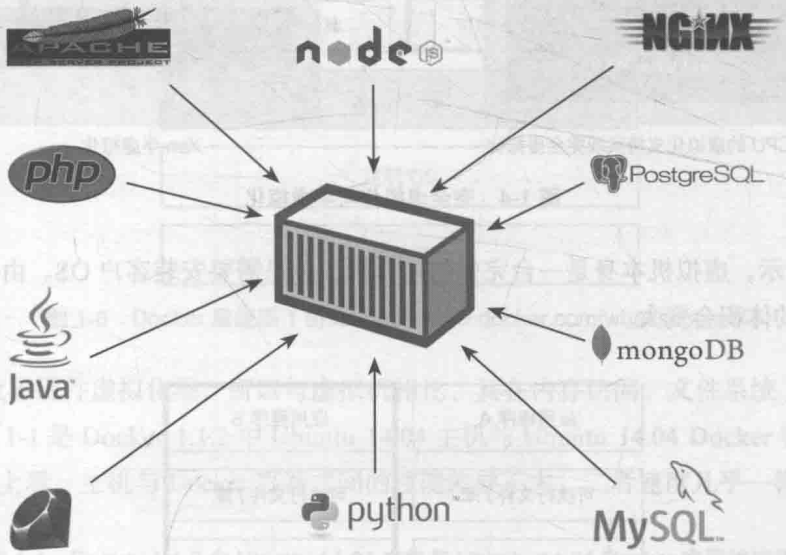


图 1-3 Docker 容器

1.1 虚拟机与 Docker

Docker 与我们之前使用的 VMware、Microsoft Hyper-V（Virtual PC）、Xen、Linux KVM 等虚拟机类似。

在虚拟机中安装 Linux 后，可以安装各种服务器程序与 DB，运行已开发的应用程序与网站。将搭建好的虚拟机镜像复制到多台服务器中运行，之后即可用一个镜像不断创建服务器。

虚拟机服务器通常单独运行，也可以使用以服务形式提供的 Amazon Web Services、Microsoft Azure、Google Cloud Platform。

1.1.1 虚拟机

虚拟机非常方便，但性能不佳。当前许多 CPU 都添加了对虚拟化功能的大量支持，但与物理机器相比，虚拟机的运行速度比较慢。

为了进一步改善“完全虚拟化”（Full Virtualization）的运行速度，半虚拟化（Paravirtualization）技术登场，现在正得到广泛应用。

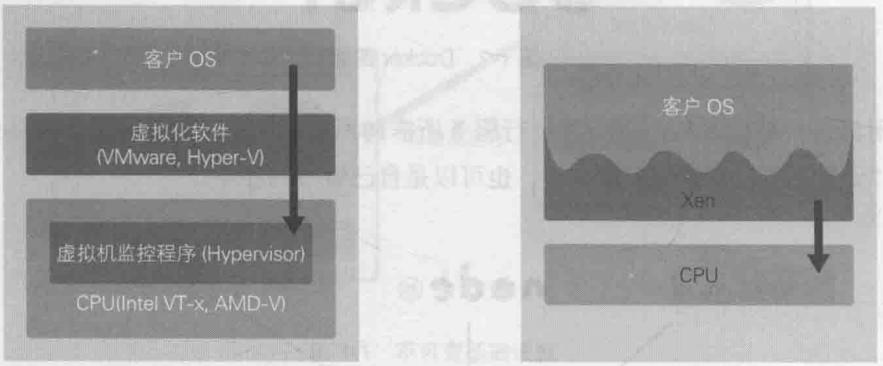


图 1-4 完全虚拟化与半虚拟化

如图 1-5 所示，虚拟机本身是一台完整的计算机，总是需要安装客户 OS。由于镜像中含有 OS，所以镜像的体积会变大。

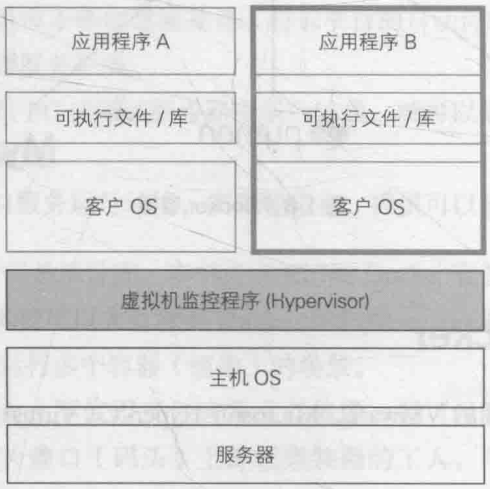


图 1-5 虚拟机层级图（出处：http://www.docker.com/whatisdocker/）

无论网速多快，收发虚拟化镜像都会非常耗时。尤其是开源虚拟化软件，其重点在于 OS 虚拟化，只提供镜像创建与运行功能，在部署与管理功能上存在不足。

提示 **虚拟机分部署**

对虚拟机进行集中管理与部署的商业产品有 VMware vCenter、Microsoft System Center。

1.1.2 Docker

与半虚拟化相比，Docker 是一种更轻量化的方式。如图 1-6 所示，使用 Docker 则不需要安装客户 OS。Docker 镜像中只隔离并安装服务器运行所需的程序与库，与主机共享 OS 资源（系统调用），这样就大大减小了镜像的体积。

图 1-6 Docker 层级图（出处：http://www.docker.com/whatisdocker/）

Docker 没有硬件虚拟化层，所以与虚拟机相比，其在内存访问、文件系统、网络速度上明显快得多。表 1-1 是 Docker 1.1.2 中 Ubuntu 14.04 主机与 Ubuntu 14.04 Docker 容器的性能测试结果。从数值上看，主机与 Docker 容器之间的性能差异不大，二者速度几乎一样。

表 1-1 Docker 1.1.2 中 Ubuntu 14.04 主机与 Ubuntu 14.04 Docker 容器性能测试

	性能测试工具	主机	Docker
CPU	sysbench	1	0.9945
写内存	sysbench	1	0.9826
读内存	sysbench	1	1.0025
磁盘 I/O	dd	1	0.9811
网络	iperf	1	0.9626

与虚拟机不同，Docker 提供了专门创建并部署镜像的功能。如同在 Git 中管理源代码一样，Docker 也提供了镜像版本管理功能。此外，为了进行集中管理，Docker 也提供镜像上传与下载

功能（Push/Pull）。就像 GitHub 一样，Docker Hub 提供帮助用户共享的 Docker 镜像（类似于 Github，也提供个人付费存储服务）。

Docker 提供了多种 API，用户可以轻松实现自动化，这对开发与服务器运营非常有用。

1.1.3 Linux 容器

Linux/Unix 环境一直提供 chroot 命令，用于更改文件系统的根目录（/）。使用 chroot 命令将指定目录设置为根目录后，即可创建 chroot jail（监牢）环境，该环境中不能访问外部文件与目录。由于 chroot 可以这样隔离目录路径，所以使用它可以最大限度地防止服务器信息泄露或受损。

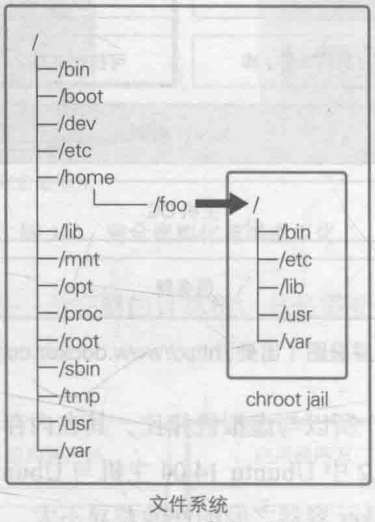


图 1-7 chroot 的目录结构

使用 chroot 命令时，我们必须自己准备要放入 chroot jail 的可执行文件与共享库，设置方法比较复杂。此外，由于不是完美的虚拟环境，故存在诸多制约。后来，Linux 提供了名为 LXC（LinuX Container）的系统级虚拟化。

LXC 并不是将整台电脑虚拟化以运行 OS，它是 Linux 内核级别提供的一种隔离虚拟空间。该虚拟空间未安装 OS，所以不能称之为虚拟机，而称作“容器”。

Linux 内核的 Control Groups（cgroups）分配 CPU、内存、磁盘、网络资源，提供完全虚拟化空间。此外，还要隔离进程树、用户账户、文件系统、IPC 等，创建与主机不同的空间，这称为 Namespace isolation(namespaces)。

LXC 利用 Linux 内核的 cgroups 与 namespaces 功能提供虚拟空间。