

循序渐进学 Docker

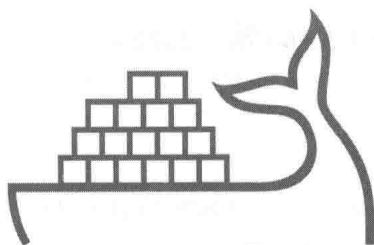
LEARNING DOCKER STEP BY STEP

李金榜 尹焯 刘天斯 陈纯 著

腾讯4位资深专家大规模应用Docker的
技术心得与经验总结



机械工业出版社
China Machine Press



循序渐进学 Docker

LEARNING DOCKER STEP BY STEP

李金榜 尹焜 刘天斯 陈纯 著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

循序渐进学 Docker / 李金榜等著. —北京: 机械工业出版社, 2016.9
(容器技术系列)

ISBN 978-7-111-54854-6

I. 循… II. 李… III. Linux 操作系统—程序设计 IV. TP316.85

中国版本图书馆 CIP 数据核字 (2016) 第 252572 号

循序渐进学 Docker

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: 李 艺

责任校对: 殷 虹

印 刷: 中国电影出版社印刷厂

版 次: 2016 年 11 月第 1 版第 1 次印刷

开 本: 186mm×240mm 1/16

印 张: 14.75

书 号: ISBN 978-7-111-54854-6

定 价: 59.00 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

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光/邹晓东

感谢家人一直以来对我工作的理解与支持，还有宝宝刘思彤今年马上就要上一年级了，相信你已经准备好了，加油！没有你们背后默默的支持与鼓励，绝对没有今天的自己，谢谢你们！

刘天斯

感谢 SNG 社交网络运营部平台技术运营中心同事们的支持和帮助。

李金榜

感谢家人和同事对我的支持和帮助，感谢一起编写本书的同事，没有你们的鞭策，我无法坚持下来。

尹烨

很荣幸能有机会参与本书的创作。感谢在编写本书过程中家人和同事对我生活的照顾和工作的支持，同时感谢其他三位作者对我参与写书的鼓励和支持。

陈纯

前 言 *Preface*

为什么要写这本书

Docker 自 2013 年诞生以来，在短短几年就迅速引爆 IT 技术圈，全球各大知名 IT 企业也纷纷加入。Docker 社区的火爆程度也是前所未有的，周边的技术案例、平台工具也是层出不穷，其中也不乏一线 IT 公司的身影，比如 Google、微软、Red Hat、VMware 等，放眼国内，基于 Docker 技术的创业公司也如雨后春笋，国内互联网公司的代表 BAT 也开始尝试在企业内部运用落地。在这样的大背景下，大家对掌握及运用 Docker 技术的欲望也越来越强烈。因此，四位笔者走到了一起，开始谋划这本书籍。

笔者都来自腾讯不同事业群及中心，都有针对各自不同应用场景做 Docker 技术研究及应用的实践经验，在研究的过程中，大家也将自己的研究历程、成果做了聚合，最终形成了本书的初稿，包括读者比较关心的 Docker 网络及存储、日常运营到源码探索，循序渐进的内容组织结构，可以让不同水平层次的读者均能有效地阅读和吸收。

本书的初衷是将研究、使用 Docker 过程中可能碰到的问题，以及解决的方法与思路做个自我梳理与总结，同时与大家分享。最终目的是让每位关注 Docker 技术的人受益。

读者对象

- 系统架构师、运维人员
- 运营开发、DevOps 人员
- 云计算工程师
- 系统管理员或企业网管
- 高等院校计算机专业的学生与教师

如何阅读本书

本书分为四部分：

第一部分为基础篇，包括第 1 至第 4 章，介绍 Docker 的基础知识及原理，介绍 Docker 是什么，可以做什么，以及如何使用 Docker 技术，包括了安装、创建容器与镜像、运行等。

第二部分为高级篇，包括第 5 至 11 章，着重讲解如何实现容器管理、镜像管理、仓库管理、网络和存储管理及项目日常维护，又补充了最新版本 Docker Swarm 容器集群和 Docker 插件开发等内容。

第三部分为案例篇，包括第 12 至第 15 章，通过对 3 个不同编排技术实现的 Docker 服务案例讲解，让读者了解一个完整的平台的搭建。

第四部分为源码探索篇，为第 16 章，介绍了 Docker 的源码结构和如何修改和编译 Docker，为读者更深入学习研究 Docker 提供一种新思路。

其中第三部分以接近实战的实例来讲解，相比于前两部分更独立。如果你是一名经验丰富的 Linux 管理员且具有 Docker 基础，可以直接切入高级篇；但如果你是一名初学者，请一定从 Docker 的基础理论知识开始学习；如果你对 Docker 的源码分解比较感兴趣，可以直接阅读第 16 章。

勘误和支持

由于水平有限，且编写时间仓促，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。为此，特意创建一个在线支持与应急方案问答站点 <http://qa.liuts.com>。你可以将书中的错误发布到“错误反馈”分类中，同时如果你遇到任何问题或有任何建议，也可以访问问答站点进行发表，我将尽量在线上为读者提供最满意的解答。我也会将相应的功能更新及时更正出来。如果你有更多的宝贵意见，欢迎加入“循序渐进学 Docker”读者 QQ 群（QQ 群账号 559435845 或者扫描以下二维码），期待能够得到你们的真挚反馈。



致谢

首先要感谢 dotCloud 公司，是他们创立了 Docker 这个容器引擎，同时也要感谢为 Docker 整个生态圈贡献大量周边组件的所有作者，是你们让 Docker 技术发展得越来越好，开源的精神与力量在你们身上体现得淋漓尽致。

感谢王冬生兄贡献他在工作中的案例（Docker 离线系统应用案例），内容具有非常高的实用价值，感谢公司各位领导及同事，感谢本书的所有作者，在大家的努力下终于促成了这本书的合作与出版。

感谢机械工业出版社华章公司的编辑杨福川、姜影老师，在这一年多的时间中始终支持我的写作，你的鼓励和帮助引导我能顺利完成全部书稿。



Contents 目 录

前言

第一部分 基础篇

第 1 章 全面认识 Docker 2

1.1 Docker 是什么 2

1.1.1 Docker 的由来 2

1.1.2 Docker 为什么这么火 4

1.1.3 Docker 究竟是什么 6

1.2 Docker 的结构与特性 8

1.2.1 Docker 构成 8

1.2.2 Docker 化应用的存在形式 10

1.2.3 Docker 对变更的管理 14

1.3 为什么使用 Docker 15

1.3.1 从代码管理说起 16

1.3.2 当前的优化策略 17

1.3.3 Github 版的应用部署解决方案 18

1.3.4 Docker 应用场景 19

1.3.5 Docker 可以解决哪些痛点 21

1.3.6 Docker 的使用成本 22

1.4 本章小结 23

第 2 章 初步体验 Docker 24

2.1 Windows 下安装 Docker 24

2.2 利用 Docker 搭建个人博客 27

2.2.1 传统的安装方法 27

2.2.2 使用 Docker 进行安装 27

2.2.3 解惑 31

2.2.4 其他注意事项 32

2.3 本章小结 32

第 3 章 Ubuntu 下使用 Docker 33

3.1 Docker 的运行平台 33

3.2 安装 Windows 和 Ubuntu 双系统 34

3.2.1 制作 Ubuntu 安装 U 盘 35

3.2.2 通过 U 盘安装 Ubuntu 36

3.3 在 Ubuntu 下安装 Docker 38

3.4 再次体验 Docker 40

3.4.1 再看个人博客 WordPress 的 搭建 40

3.4.2 开源的版本控制利器—— GitLab	40	6.3 项目中的镜像分层	83
3.4.3 项目管理系统—— Redmine	42	6.4 定制私有的基础镜像	84
3.5 本章小结	44	6.5 本章小结	85
第4章 Docker 的基础知识	45	第7章 Docker 仓库管理	86
4.1 Docker 的基本概念和常用 操作指令	45	7.1 镜像的公有仓库	86
4.1.1 Docker 三大基础组件	46	7.1.1 创建 Docker Hub 账户	86
4.1.2 常用的 Docker 指令	47	7.1.2 基本操作	87
4.1.3 Docker 的组织结构	49	7.2 私有仓库	88
4.2 10 分钟的动手教程	50	7.2.1 安装 docker-registry	88
4.3 本章小结	60	7.2.2 配置文件	91
		7.3 构建安全的私有仓库	92
		7.3.1 Nginx 安装与配置	92
		7.3.2 SSL 证书	94
		7.3.3 客户端配置	96
		7.4 本章小结	97
第二部分 高级篇		第8章 Docker 网络和存储 管理	98
第5章 Docker 容器管理	62	8.1 Docker 网络	98
5.1 单一容器管理	62	8.1.1 Docker 的通信方式	98
5.1.1 容器的标示符	63	8.1.2 网络配置	100
5.1.2 查询容器信息	64	8.2 Docker 数据管理	101
5.1.3 容器内部命令	65	8.2.1 基本介绍	101
5.2 多容器管理	66	8.2.2 数据卷	102
5.2.1 Docker Compose	67	8.2.3 数据卷容器	105
5.2.2 配置文件	69	8.2.4 备份、恢复和迁移 数据卷	107
5.3 本章小结	73	8.3 Docker 存储驱动	108
第6章 Docker 镜像管理	74	8.3.1 Docker 存储驱动历史	108
6.1 认识 Docker 镜像	74		
6.2 Dockerfile	78		

8.3.2 Docker overlaysfs driver	109
8.4 本章小结	112

第9章 Docker 项目日常维护

9.1 宿主机的管理	113
9.1.1 安装 Docker 并启动	113
9.1.2 网桥模式	115
9.2 GitLab 的日常维护	116
9.2.1 项目的创建	116
9.2.2 代码版本控制	118
9.2.3 日常维护	119
9.3 本章小结	122

第10章 Docker Swarm 容器 集群

10.1 Swarmkit 核心设计	123
10.2 Swarmkit 集群搭建	124
10.2.1 创建 Manager 节点	125
10.2.2 创建 Worker 节点	126
10.3 Swarmkit 基本功能	127
10.3.1 service 创建与删除	127
10.3.2 service 扩容与缩容	128
10.3.3 service 灰度升级	128
10.3.4 service 网络配置、域名 解析和负载均衡	129
10.3.5 Swarmkit 节点管理	131
10.3.6 Manager 节点和 Worker 节点角色切换	133
10.4 Swarmkit 负载均衡原理 分析	134

10.5 本章小节	137
-----------	-----

第11章 Docker 插件开发

11.1 Docker 插件工作机制	138
11.1.1 Docker 插件接口	138
11.1.2 插件发现机制	139
11.1.3 JSON 文件格式	139
11.1.4 插件的生命周期	140
11.1.5 利用 systemd socket activation 功能管理 插件	140
11.1.6 API 格式	141
11.2 Docker volume 插件开发	141
11.2.1 cgroups 使用方法和 工作原理	142
11.2.2 docker volume 接口	143
11.2.3 实现 cgroups-volume volume 插件	145
11.3 本章小节	147

第三部分 案例篇

第12章 Docker 离线系统应用 案例

12.1 为什么使用 Docker	150
12.2 离线系统业务架构	152
12.3 Clip 名字服务	153
12.4 Clip 名字服务与 Docker 应用	156
12.5 本章小结	158

第 13 章 Etcd、Cadvisor 和**Kubernetes 实践** 159

13.1 Etcd 实践 159

13.1.1 安装 Etcd 160

13.1.2 使用方法 160

13.2 Cadvisor 实践 164

13.2.1 安装 Cadvisor 164

13.2.2 Cadvisor API 165

13.3 Kubernetes 实践 166

13.3.1 基本概念 167

13.3.2 环境说明 168

13.3.3 环境部署 169

13.3.4 API 常用操作 173

13.3.5 创建 pod 单元 173

13.3.6 实战案例 176

13.4 本章小结 181

第 14 章 构建 Docker 高可用及**自动发现架构实践** 182

14.1 架构优势 182

14.2 架构介绍 183

14.3 架构搭建 184

14.3.1 组件环境部署 185

14.3.2 Etcd 配置 186

14.3.3 Confd 配置 186

14.3.4 容器提交注册 190

14.4 业务上线 195

14.5 本章小结 198

第 15 章 Docker Overlay Network**实践** 199

15.1 环境介绍 199

15.2 容器与容器之间通信 200

15.2.1 启动 docker daemon 200

15.2.2 创建网络 200

15.2.3 启动容器 201

15.3 Docker 的 VXLAN 实现 204

15.3.1 VXLAN 帧结构 205

15.3.2 Docker 内部实现 205

15.3.3 Linux VXLAN 设备 207

15.4 容器访问外部网络 207

15.5 外部网络访问容器 209

15.6 本章小结 212

第四部分 源码探索篇**第 16 章 Docker 源码探索** 214

16.1 Docker 源码目录结构 214

16.2 源码编译 Docker 219

16.2.1 修改 Dockerfile 220

16.2.2 其他 222

16.2.3 编译源码的好处 222

16.3 输出函数调用关系 223

16.4 本章小结 225



第一部分 *Part 1*

基础篇

- 第 1 章 全面认识 Docker
 - 第 2 章 初步体验 Docker
 - 第 3 章 Ubuntu 下使用 Docker
 - 第 4 章 Docker 的基础知识
-

全面认识 Docker

欢迎来到 Docker 的世界。

Docker, Golang 社区杀手级的应用, 是 Github 上最活跃的项目之一, 也是开源社区最受欢迎的项目。

Docker, 号称要成为所有云应用的基石, 并把互联网升级到下一代。

开发、测试、运维人员看到 Docker, 都激动地说: “太好了, 这正是我所需要的!”

Docker 是什么, 能解决什么问题, 为什么这么火? 本章将一一道来。

1.1 Docker 是什么

首先, 我们了解下 Docker 产生的历史背景和当前发展情况, 通过和一些熟悉的事物做类比, 让大家对 Docker 有一个初步认识和了解。

1.1.1 Docker 的由来

Docker 是 dotCloud 公司开源的一款产品。dotCloud 公司是 2010 年新成立的一家公司, 主要基于 PaaS (Platform as a Service, 平台即服务) 平台为开发者提供服务。在 PaaS 平台下, 所有的服务环境已经预先配置好了, 开发者只需要选择服务类型、上传代码就可对外服务, 不需要花费大量的时间搭建服务和配置环境。dotCloud 的 PaaS 平

台已经做得足够好了，它支持几乎所有主流的 Web 编程语言和数据库，可以让开发者随心所欲地选择自己需要的编程语言、数据库和编程框架，而且它的设置非常简单，每次编码后只需要运行一条命令就能把整个网站部署上去；并且利用多层次平台的概念，理论上，它的应用可以运行在各种类型的云服务上。两三年下来，虽然 DotCloud 也在业界获得不错的口碑，但由于整个 PaaS 市场还处于培育阶段，dotCloud 公司表现得不温不火，没有出现爆发性的增长。

2013 年，dotCloud 的 CEO Solomon Hykes 决定把 dotCloud 内部使用的 Container 容器技术单独拿出来开源。2013 年 3 月发布 Docker 的 V0.1 版本，并且基本保持每月一个版本的迭代速度，到了 8 月，Docker 已经足够火爆，并广受好评，各种各样的技术论坛和技术峰会都开始热烈讨论与推荐 Docker，这时 Docker 才只发布到 V0.6 版本。

随着 Docker 的流行，越来越多的优秀开发者加入 Docker 社区参加开发。这里值得一提的是，Docker 是基于 Linux3.8 以上内核，在 aufs 分层文件系统下构建的，主要运行在 Ubuntu 的系统下。REHL/Centos 当时最新版 6 系列还是基于 Liunux2.6.32 内核，无法运行 Docker。为了让 REHL/Centos 尽快支持 Docker，RedHat 公司的工程师亲自出马，加班加点为 Docker 贡献代码，新增对 devicemapper 的支持来实现文件系统分层，终于顺利地让 Docker 在 REHL/Centos 运行起来。

随着 Docker 在业界的知名度越来越高，到了 2013 年 10 月，dotCloud 公司索性更名为 Docker 股份有限公司，工作的重心也从 PaaS 平台业务转向全面围绕 Docker 来开发。到了 2014 年 1 月，Docker 公司宣布完成 15 000 万美元的融资，雅虎联合创始人杨致远也参与跟投。

虽然 Docker 迟迟没有发布 1.0 版，但好多公司已纷纷把 Docker 应用到生产环境。其中，美国奢侈品电商 Gilt 的 CTO 说：“使用 Docker 以后，突然之间，传统方式中的各种问题都消失了，我们接下来要考虑如何进一步提高软件生产效率，让软件开发更加安全和创新。这种转变太不可思议了！”

千呼万唤，到了 2014 年 6 月 9 日，Docker 终于发布了 V1.0 版，并举办了 DockerCon 2014 大会，大会上来自 Google、IBM、RedHat、Rackspace 等公司的核心人物均发表了主题演讲，纷纷表示支持并加入 Docker 的阵营。Docker 的 CTO Solomon Hykes 充满雄心壮志地说：“我们能把互联网升级到下一代！”Google 的基础架构部副总裁 Eric Brewer 也附和道：“容器技术曾是 Google 的基础，我们和 Docker 联手，把容器技术打造为所有云应用的基石。”

Google 自 2004 年就开始使用容器技术，目前他们每周要启动超过 20 亿个容器，每秒钟新启动的容器就超过 3000 个，在容器技术方面有大量的积累。曾相继开源了 Cgroup 和 Imctfy 这两个重量级项目。Google 对 Docker 的支持力度非常大，不仅把 Imctfy 先进之处融入 Docker 中，还把自己的容器管理系统 (kubernetes) 也开源出来。

2014 年 8 月，不缺钱的 Docker 再次融资，融资超 4 千万美元，估值达到 4 亿美元。

所有的云计算大公司，如 Azure、Google 和亚马逊等都在支持 Docker 技术，这实际上也让 Docker 成为云计算领域的一大重要组成部分。

2014 年 10 月 15 日，Azure 副总裁 Jason Zander 宣布了微软与 Docker 的合作伙伴关系；2014 年 11 月 5 日，Google 发布支持 Docker 的产品 DockerGoogle Container Engine；2014 年 11 月 13 日，Amazon 发布支持 Docker 的产品 AWS Container Service。至此，几个重要的云计算大公司都已经支持 Docker 技术，这不仅让 Docker 成为云计算领域的一个重要级成员，也让 Docker 成为云应用部署的事实上的标准。

2014 年 12 月，Docker 发布了 Docker 集群管理工具 Machine 和 Swarm，标志着 Docker 开始突破一个标准的容器框架，打造属于 Docker 自己的集群平台和生态圈。

2015 年 4 月，Docker 公司宣布完成了 9500 万美元的 D 轮融资。

2015 年 10 月，Docker 收购 Tutum，Tutum 本身已经实现对亚马逊网络服务 (AWS)、Digital Ocean、微软的 Azure 等主流云服务商的良好支持。

2016 年 1 月，Docker 官方计划全面支持自身的 Alpine Linux，使用它构建的基础镜像最小只有 5M。

截至 2016 年 3 月，Docker 在 Github 上收获 29 962 个关注 (star)、8437 个拷贝 (Fork)，在 Github 所有项目中排第 7 位，在云平台管理领域排名第一，远远超 Openstack 项目的 1316 个关注、768 个拷贝。

1.1.2 Docker 为什么这么火

Docker 从诞生到现在，短短两年时间，已经成为开源社区最火爆的项目，风头已经远远盖过了近年来很流行的 Puppet 和 OpenStack。那么 Docker 的火爆到底是一种炒作、一种跟风，还是它确实名副其实、众望所归呢？

要回答这个问题，首先看看当前我们所处的环境和面临的问题。

随着计算机近几十年的蓬勃发展，产生了大量优秀系统和软件。比如：

- 操作系统，如 REHL/Centos、Debian/Unbuntu、FreeBSD、OpenSuse 等。
- 编程语言，如 Java、C/C++、Python、Ruby、Golang 等。
- Web 服务器，如 Apache、Nginx、Lighttpd 等。
- 数据库，如 Mysqld、Redis、Mongodb 等。

现在的软件开发人员真是幸运，可以在这么多种类中自由选择。自由选择的结果是，维护一个非常庞大的开发、测试和生产环境，开发、测试和运维人员都被种类繁多的环境折腾得筋疲力尽，不得不收缩战线，每种类型的软件只选择一两种来支持。许多优秀的开发框架和软件尽管有不少优秀特性，但因为维护麻烦，便没有了用武之地。

即便每种类型的软件只选择一两种来支持，随着操作系统和软件版本的更新迭代，维护工作还是变得越来越庞大。

面对这种情况，业界大牛群策群力，给出了很多解决方案，比较有代表的是 Puppet 和 OpenStack。

- Puppet 是集成的配置管理系统，它把文件、用户、cron 任务、软件包、系统服务等抽象为资源，并通过自有的语言描述资源间的依赖关系，集中管理各类资源的安装配置。Puppet 主要适用于需要大批量部署相同服务的应用场景。
- OpenStack 是开源的云计算管理平台项目，可以帮助企业内部实现类似于 Amazon EC2 的云基础架构服务。虽然灵活，但组件繁多、构建复杂，比较适合中大型企业使用。

Puppet 和 OpenStack 虽然比较流行，但适应的场景有限，不具备通用性。正当大家在众多方案中左右为难时，Docker 出现了，它作为一个开源的应用容器引擎，让开发者可以打包他们的应用及依赖环境到一个可移植的容器中，然后发布到任何运行有 Docker 引擎的机器上。它集版本控制、克隆继承、环境隔离等特性于一身，提出一整套软件构建、部署和维护的解决方案，可以非常方便地帮助开发人员，让大家可以随心所欲地使用软件而又不会深陷到环境配置中。

这只是 Docker 的一个应用场景而已，Docker 还能干更多的事情。

作为计算机的从业人员，下面场景你或许碰到过。

- 小 A 是一名资深码农，作为新招聘实习生的导师，小 A 要给实习生的开发机装一套和自己开发机一样的运行环境，不仅要安装 Nginx、Java、Mysqld 和一些依赖库等，还要修改相关的配置文件。结果花了一天时间，小 A 也没把实习生的开发环境搞定，在徒弟面前颜面尽失，尴尬不已。

- ❑ 小 B 是一名 QA 测试工程师，他按开发给的文档、部署的服务，测试出一大堆问题，通过和开发的沟通，发现是开发和测试环境不一致引起的。
- ❑ 小 C 作为一名业务运维工程师，同时维护开发、测试、生产三套环境，经常在不同环境下装相同的包，做大量重复工作。
- ❑ 小 D 同时在为三个项目开发功能模块，他要不停地修改他的开发环境为适应在三个项目间开发、联调测试。
- ❑ 小 E 发现服务器被入侵过，他想知道什么文件被篡改过。
- ❑ 小 F 从离职同事那里接手一个系统，文档不全，突然一台机器硬件故障，他不知道该如何重新部署这个应用。
- ❑ 小 G 新上线一个游戏，游戏火爆超预期，需要紧急扩容，花了一两个小时才完成扩容，期间用户体验很卡，流失不少潜在用户。
- ❑ 小 H 和小 I 共同维护一套系统，分工轮流值夜班，但一出现突发故障，排查问题时，即便半夜，还需要把对方叫醒，确认下对方在前一天有没有变更过什么配置。
- ❑ 小 M 的一个机房要裁撤了，该机房的数千个应用都要迁移到其他机房，小 M 觉得这项工作非常庞大，半年时间都未必能完成。

但是如果使用 Docker，这些根本不算事儿，分分钟就能搞定。

Docker 的解决方案简单、灵活、高效，还很直观，甚至不需要过多地改变现有的使用习惯，就可以和已有的工具，如 Puppet、OpenStack 等配合使用。各种优势让 Docker 脱颖而出，有鹤立鸡群的感觉，Docker 的火爆也就不难理解了。

1.1.3 Docker 究竟是什么

按照官方的说法，Docker 是一个开源的应用容器引擎。很多人觉得这个说法太抽象，不容易理解。

那我们就从最熟悉的事物说起吧，但凡从事过计算机相关行业的人，对 Java、Android 和 Github 都很熟悉。

先说 Java，在 Java 之前的编程语言，像 C/C++，是严重依赖平台的，在不同平台下，需要重新编译才能运行。Java 的一个非常重要的特性就是与平台无关性，而使用 Java 虚拟机是实现这一特性的关键。Java 虚拟机屏蔽了与具体平台相关的信息，使得 Java 语言编译程序只需生成可以在 Java 虚拟机上运行的目标代码（字节码），就可以在