

- 容器技术是继大数据和云计算之后又一炙手可热的技术，而且未来相当一段时间内都会非常流行
- 对 IT 从业者来说，掌握容器技术是市场的需要，也是提升自我价值的重要途径
- 每一轮新技术的兴起，无论对公司还是个人既是机会也是挑战



Docker is the leading Containers as a Service platform

# 每天5分钟 玩转Docker容器技术

CloudMan 著

清华大学出版社





# 每天5分钟 玩转Docker容器技术

CloudMan 著

清华大学出版社  
北京

## 内 容 简 介

Docker 和容器技术是当下最火的 IT 技术，无论是互联网还是传统企业都在研究和实践如何用容器构建自己的 IT 基础设施。学习本书能够让读者少走弯路，系统地学习、掌握和实践 Docker 和容器技术。

本书共分为三部分。第一部分介绍容器技术生态环境。第二部分是容器核心知识，包括架构、镜像、容器、网络和存储。第三部分是容器进阶知识，包括多主机管理、跨主机网络方案、监控、日志管理和数据管理。读者在学习的过程中，可以跟着教程进行操作，在实践中掌握 Docker 容器技术的核心技能。在之后的工作中，可以将本教程作为参考书，按需查找相关知识点。

本书主要面向微服务软件开发人员，以及 IT 实施和运维工程师等相关人员，也适合高等院校和培训学校相关专业的师生教学参考。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

### 图书在版编目 (CIP) 数据

每天 5 分钟玩转 Docker 容器技术 / CloudMan 著. —北京：清华大学出版社，2017  
ISBN 978-7-302-47970-3

I. ①每… II. ①C…III. ①Linux 操作系统—程序设计 IV. ①TP316.85

中国版本图书馆 CIP 数据核字 (2017) 第 207273 号

责任编辑：夏毓彦  
封面设计：王 翔  
责任校对：闫秀华  
责任印制：李红英

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>，<http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社总机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969，[c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈：010-62772015，[zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印 装 者：三河市君旺印务有限公司

经 销：全国新华书店

开 本：190mm×260mm 印 张：16 字 数：410 千字

版 次：2017 年 9 月第 1 版 印 次：2017 年 9 月第 1 次印刷

印 数：1~3500

定 价：49.00 元

# 前言

## 写在最前面

《每天 5 分钟玩转 Docker 容器技术》是一个有关容器技术的教程，有下面两个特点：

### 1. 系统讲解当前最流行的容器技术

从容器的整个生态环境到各种具体的技术，从整体到细节逐一讨论。

### 2. 重实践并兼顾理论

从实际操作的角度带领大家学习容器技术。

## 为什么要写这个

简单回答是：容器技术非常热门，但门槛高。

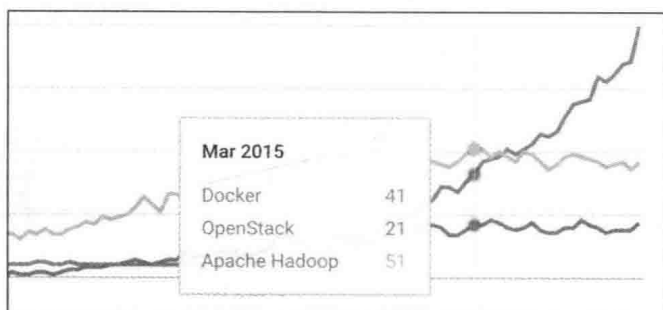
容器技术是继大数据和云计算之后又一炙手可热的技术，而且未来相当一段时间内都会非常流行。

对 IT 行业来说，这是一项非常有价值的技术。而对 IT 从业者来说，掌握容器技术是市场的需要，也是提升自我价值的重要途径。

拿我自己的工作经历来说，毕业后的头几年是做 J2EE 应用开发。后来到一家大型 IT 公司，公司的产品从中间件到操作系统、从服务器到存储、从虚拟化到云计算都有涉及。

我所在的部门是专门做 IT 基础设施实施服务的，最开始是做传统的 IT 项目，包括服务器配置，双机 HA 等。随着虚拟化技术成熟，工作上也开始涉及各种虚拟化技术的规划和实施，包括 VMWare、KVM、PowerVM 等。后来云计算兴起，在公司业务和个人兴趣的驱动下，开始学习和实践 OpenStack，在这个过程中写了《每天 5 分钟玩转 OpenStack》教程并得到大家的认可。

现在以 Docker 为代表的容器技术来了，而且关注度越来越高，这一点可以从 Google Trend 中 Docker 的搜索上升趋势中清楚看到，如下图所示（图中曲线上扬最高的为 Docker）。



每一轮新技术的兴起，无论对公司还是个人既是机会也是挑战。

我个人的看法是：如果某项新技术未来将成为主流，就应该及早尽快掌握。因为：

### 1. 新技术意味着新的市场和新的需求

初期掌握这种技术的人不会很多，而市场需求会越来越大，因而会形成供不应求的卖方市场，物以稀为贵，这对技术人员将是一个难得的价值提升机会。

### 2. 学习新技术需要时间和精力，早起步早成材

机会讲过了，咱们再来看看挑战。

新技术往往意味着技术上的突破和创新，会有不少新的概念和方法，而且从大数据、云计算和容器技术来看，这些新技术都是平台级别，覆盖的技术范围非常广，包括计算、网络、存储、高可用、监控、安全等多个方面，要掌握这些新技术对 IT 老兵尚有不小困难，更别说新人了。

由于对技术一直保持着很高的热诚和执着，在掌握了 OpenStack 相关 IaaS 技术后，我便开始调研 PaaS 技术栈。正好这时 Docker 也越来越流行，自然而然便开始了容器相关技术的学习研究和实践。

学习容器技术的过程可以说是惊喜不断，经常惊叹于容器理念的先进和容器生态环境的完整和强大。很多传统软件开发和运维中的难题在容器世界里都能轻松解决，也渐渐理解了容器为何如此受到青睐。

不夸张地说，容器为我打开了一扇通往另一个软件世界的大门，让我沉浸其中，激动不已。高兴之余，我也迫不及待地想把我所看到、所学到和所想到的有关容器的知识介绍给更多的人，让更多的 IT 工程师能够从容器技术中受益。

我希望这个教程也能为大家打开这扇门，降低学习的曲线，系统地学习和掌握容器技术。

## 写给谁看

这套教程的目标读者包括：

### 1. 软件开发人员

相信微服务架构（Microservice Architectur）会逐渐成为开发应用系统的主流，而容器则

是这种架构的基石。市场将需要更多能够开发出基于容器的应用程序的软件开发人员。

## 2. IT 实施和运维工程师

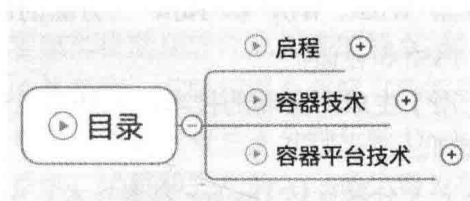
容器为应用提供了更好的打包和部署方式，越来越多的应用将以容器的方式在开发、测试和生产环境中运行。掌握容器相关技术将成为实施和运维工程师的核心竞争力。

## 3. 我自己

我坚信最好的学习方法是分享。编写这个教程同时也是对自己学习和实践容器技术的总结。对于知识，只有把它写出来并能够让其他人理解，才能说明真正掌握。

## 包含哪些内容

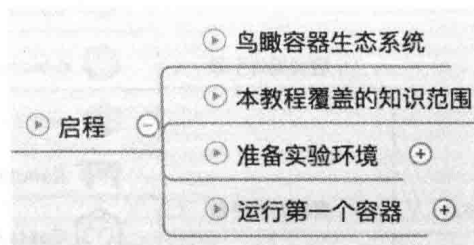
本系列教程分为《每天 5 分钟玩转 Docker 容器技术》和《每天 5 分钟玩转 Docker 容器平台》两本，包括以下三大块内容：



下面分别介绍各部分包含的内容。

### 1. 启程

如下图所示，“启程”会介绍容器的生态系统，让大家先从整体上了解容器包含哪些技术，各种技术之间的相互关系是什么，然后再来看我们的教程都会涉及生态中的哪些部分。



为了让大家尽快对容器有个感性认识，我们会搭建实验环境并运行第一个容器，为之后的学习热身。

### 2. 容器技术

“容器技术”主要内容如下图所示，包含“容器核心知识”和“容器进阶知识”两部分。



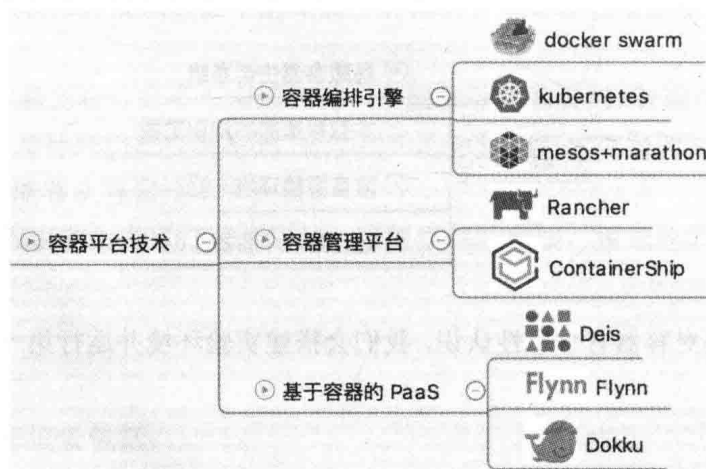
核心知识主要回答有关容器 What、Why 和 How 三方面的问题，其中以 How 为重，将展开讨论架构、镜像、容器、网络和存储。

进阶知识包括将容器真正用于生成所必需的技术，包括多主机管理、跨主机网络、监控、数据管理、日志管理和安全管理。

这部分内容将在本书《每天 5 分钟玩转 Docker 容器技术》中详细讨论。

### 3. 容器平台技术

如下图所示，“容器平台技术”包括容器编排引擎、容器管理平台和基于容器的 PaaS。容器平台技术在生态环境中占据着举足轻重的位置，对于容器是否能够落地，是否能应用于生产至关重要。



我们将在本系列教程的另一本书《每天 5 分钟玩转 Docker 容器平台》中详细讨论容器编排引擎、容器管理平台和基于容器的 PaaS，学习和实践业界最具代表性的开源产品。

## 怎样的编写方式

我会继续采用《每天 5 分钟玩转 OpenStack》（本书已在清华出版）的方式，通过大量的实验由浅入深地探讨和实践容器技术，力求达到如下目标：

- (1) 快速上手：以最直接、最有效的方式让大家把容器用起来。
- (2) 循序渐进：由易到难、从浅入深，详细分析容器的各种功能和配置使用方法。
- (3) 理解架构：从设计原理和架构分析入手，深入探讨容器的架构和运行机理。
- (4) 注重实践：以大量实际操作案例为基础，让大家能够掌握真正的实施技能。

在内容的发布上还是通过微信公众号（cloudman6）每周一、三、五定期分享。欢迎大家通过公众号提出问题和建议，进行技术交流。

## 为什么叫《每天 5 分钟玩转 Docker 容器技术》

为了降低学习的难度并且考虑到移动端碎片化阅读的特点，每次推送的内容大家只需要花 5 分钟就能看完（注意这里说的是看完，有时候完全理解可能需要更多时间哈），每篇内容包含 1~3 个知识点，这就是我把本书命名为《每天 5 分钟玩转 Docker 容器技术》的原因。虽然是碎片化推送，但整个教程是系统、连贯和完整的，只是化整为零了。

好了，今天这 5 分钟算是开了个头，下面我们正式开始玩转 Docker 容器技术。

编者  
2017 年 7 月



# 目 录

## 第一篇 启程

第 1 章 鸟瞰容器生态系统 .....	3
1.1 容器生态系统 .....	3
1.2 本教程覆盖的知识范围 .....	10
1.3 准备实验环境 .....	10
1.3.1 环境选择 .....	10
1.3.2 安装 Docker .....	10
1.4 运行第一个容器 .....	11
1.5 小结 .....	12

## 第二篇 容器技术

第 2 章 容器核心知识概述 .....	15
2.1 What —— 什么是容器 .....	15
2.2 Why —— 为什么需要容器 .....	16
2.2.1 容器解决的问题 .....	16
2.2.2 Docker 的特性 .....	20
2.2.3 容器的优势 .....	20
2.3 How —— 容器是如何工作的 .....	21
2.4 小结 .....	24
第 3 章 Docker 镜像 .....	26
3.1 镜像的内部结构 .....	26
3.1.1 hello-world —— 最小的镜像 .....	26
3.1.2 base 镜像 .....	27
3.1.3 镜像的分层结构 .....	30
3.2 构建镜像 .....	32
3.2.1 docker commit .....	32
3.2.2 Dockerfile .....	34
3.3 RUN vs CMD vs ENTRYPOINT .....	42
3.3.1 Shell 和 Exec 格式 .....	42
3.3.2 RUN .....	44
3.3.3 CMD .....	44
3.3.4 ENTRYPOINT .....	45
3.3.5 最佳实践 .....	46

3.4	分发镜像 .....	46
3.4.1	为镜像命名 .....	46
3.4.2	使用公共 Registry .....	49
3.4.3	搭建本地 Registry .....	51
3.5	小结 .....	52
<b>第 4 章</b>	<b>Docker 容器 .....</b>	<b>55</b>
4.1	运行容器 .....	55
4.1.1	让容器长期运行 .....	56
4.1.2	两种进入容器的方法 .....	57
4.1.3	运行容器的最佳实践 .....	59
4.1.4	容器运行小结 .....	59
4.2	stop/start/restart 容器 .....	60
4.3	pause / unpause 容器 .....	61
4.4	删除容器 .....	61
4.5	State Machine .....	62
4.6	资源限制 .....	65
4.6.1	内存限额 .....	65
4.6.2	CPU 限额 .....	66
4.6.3	Block IO 带宽限额 .....	68
4.7	实现容器的底层技术 .....	69
4.7.1	cgroup .....	70
4.7.2	namespace .....	70
4.8	小结 .....	72
<b>第 5 章</b>	<b>Docker 网络 .....</b>	<b>74</b>
5.1	none 网络 .....	74
5.2	host 网络 .....	75
5.3	bridge 网络 .....	76
5.4	user-defined 网络 .....	78
5.5	容器间通信 .....	84
5.5.1	IP 通信 .....	84
5.5.2	Docker DNS Server .....	85
5.5.3	joined 容器 .....	85
5.6	将容器与外部世界连接 .....	87
5.6.1	容器访问外部世界 .....	87
5.6.2	外部世界访问容器 .....	90
5.7	小结 .....	91
<b>第 6 章</b>	<b>Docker 存储 .....</b>	<b>92</b>
6.1	storage driver .....	92
6.2	Data Volume .....	94

6.2.1	bind mount.....	94
6.2.2	docker managed volume.....	96
6.3	数据共享.....	99
6.3.1	容器与 host 共享数据.....	99
6.3.2	容器之间共享数据.....	99
6.4	volume container.....	100
6.5	data-packed volume container.....	102
6.6	Data Volume 生命周期管理.....	103
6.6.1	备份.....	104
6.6.2	恢复.....	104
6.6.3	迁移.....	104
6.6.4	销毁.....	104
6.7	小结.....	105

### 第三篇 容器进阶知识

第 7 章	多主机管理.....	109
7.1	实验环境描述.....	110
7.2	安装 Docker Machine.....	111
7.3	创建 Machine.....	112
7.4	管理 Machine.....	114
第 8 章	容器网络.....	117
8.1	libnetwork & CNM.....	117
8.2	overlay.....	119
8.2.1	实验环境描述.....	120
8.2.2	创建 overlay 网络.....	121
8.2.3	在 overlay 中运行容器.....	122
8.2.4	overlay 网络连通性.....	124
8.2.5	overlay 网络隔离.....	126
8.2.6	overlay IPAM.....	127
8.3	macvlan.....	127
8.3.1	准备实验环境.....	127
8.3.2	创建 macvlan 网络.....	128
8.3.3	macvlan 网络结构分析.....	130
8.3.4	用 sub-interface 实现多 macvlan 网络.....	131
8.3.5	macvlan 网络间的隔离和连通.....	132
8.4	flannel.....	136
8.4.1	实验环境描述.....	137
8.4.2	安装配置 etcd.....	137
8.4.3	build flannel.....	138

8.4.4	将 flannel 网络的配置信息保存到 etcd .....	139
8.4.5	启动 flannel .....	139
8.4.6	配置 Docker 连接 flannel .....	141
8.4.7	将容器连接到 flannel 网络 .....	143
8.4.8	flannel 网络连通性 .....	144
8.4.9	flannel 网络隔离 .....	146
8.4.10	flannel 与外网连通性 .....	146
8.4.11	host-gw backend .....	146
8.5	weave .....	148
8.5.1	实验环境描述 .....	148
8.5.2	安装部署 weave .....	149
8.5.3	在 host1 中启动 weave .....	149
8.5.4	在 host1 中启动容器 .....	150
8.5.5	在 host2 中启动 weave 并运行容器 .....	153
8.5.6	weave 网络连通性 .....	154
8.5.7	weave 网络隔离 .....	155
8.5.8	weave 与外网的连通性 .....	156
8.5.9	IPAM .....	158
8.6	calico .....	158
8.6.1	实验环境描述 .....	159
8.6.2	启动 etcd .....	159
8.6.3	部署 calico .....	160
8.6.4	创建 calico 网络 .....	161
8.6.5	在 calico 中运行容器 .....	161
8.6.6	calico 默认连通性 .....	164
8.6.7	calico policy .....	167
8.6.8	calico IPAM .....	169
8.7	比较各种网络方案 .....	170
8.7.1	网络模型 .....	171
8.7.2	Distributed Store .....	171
8.7.3	IPAM .....	171
8.7.4	连通与隔离 .....	172
8.7.5	性能 .....	172
<b>第 9 章</b>	<b>容器监控 .....</b>	<b>173</b>
9.1	Docker 自带的监控子命令 .....	173
9.1.1	ps .....	173
9.1.2	top .....	174
9.1.3	stats .....	175
9.2	sysdig .....	175
9.3	Weave Scope .....	179

9.3.1	安装 .....	179
9.3.2	容器监控 .....	181
9.3.3	监控 host .....	184
9.3.4	多主机监控 .....	186
9.4	cAdvisor .....	189
9.4.1	监控 Docker Host .....	189
9.4.2	监控容器 .....	191
9.5	Prometheus .....	194
9.5.1	架构 .....	194
9.5.2	多维数据模型 .....	195
9.5.3	实践 .....	196
9.6	比较不同的监控工具 .....	204
9.7	几点建议 .....	205
<b>第 10 章</b>	<b>日志管理 .....</b>	<b>207</b>
10.1	Docker logs .....	207
10.2	Docker logging driver .....	209
10.3	ELK .....	211
10.3.1	日志处理流程 .....	211
10.3.2	安装 ELK 套件 .....	212
10.3.3	Filebeat .....	214
10.3.4	管理日志 .....	216
10.4	Fluentd .....	220
10.4.1	安装 Fluentd .....	221
10.4.2	重新配置 Filebeat .....	221
10.4.3	监控容器日志 .....	221
10.5	Graylog .....	222
10.5.1	Graylog 架构 .....	222
10.5.2	部署 Graylog .....	223
10.5.3	配置 Graylog .....	225
10.5.4	监控容器日志 .....	227
10.6	小结 .....	229
<b>第 11 章</b>	<b>数据管理 .....</b>	<b>230</b>
11.1	从一个例子开始 .....	230
11.2	实践 Rex-Ray driver .....	232
11.2.1	安装 Rex-Ray .....	232
11.2.2	配置 VirtualBox .....	234
11.2.3	创建 Rex-Ray volume .....	236
11.2.4	使用 Rex-Ray volume .....	237
	<b>写在最后 .....</b>	<b>243</b>

# 第一篇 启程

---

对于像容器这类平台级别的技术，通常涉及的知识范围会很广，相关的软件，解决方案也会很多，初学者往往容易迷失。

那怎么办呢？

我们可以从生活经验中寻找答案。

当我们去陌生城市旅游了解一下这个城市，一般我们会怎么做？

我想大部分人应该会打开手机看一下这个城市的地图：

- (1) 城市大概的位置和地理形状是什么？
- (2) 都由哪几个区或县组成？
- (3) 主要的交通干道是哪几条？

同样的道理，学习容器技术我们可以先从天上鸟瞰一下：

- (1) 容器生态系统包含哪些不同层次的技术？
- (2) 不同技术之间是什么关系？
- (3) 哪些是核心技术？哪些是辅助技术？

首先得对容器技术有个整体认识，之后我们的学习才能够有的放矢，才能够分清轻重缓急，做到心中有数，这样就不容易迷失了。

接下来我会根据自己的经验帮大家规划一条学习路线，一起探索容器生态系统。

学习新技术得到及时反馈是非常重要的，所以我们马上会搭建实验环境，并运行第一个容器，感受什么是容器。

千里之行始于足下，让我们从了解生态系统开始吧。



# 第 1 章

## ◀ 鸟瞰容器生态系统 ▶

### 1.1 容器生态系统

一谈到容器，大家都会想到 Docker。

Docker 现在几乎是容器的代名词。确实，是 Docker 将容器技术发扬光大。同时，大家也需要知道围绕 Docker 还有一个生态系统。Docker 是这个生态系统的基石，但完善的生态系统才是保障 Docker 以及容器技术能够真正健康发展的决定因素。

大致来看，容器生态系统包含核心技术、平台技术和支持技术，如图 1-1 所示。下面分别介绍。

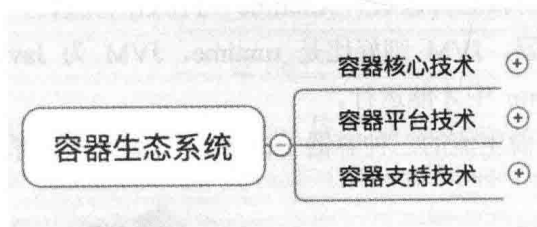


图 1-1

#### 1. 容器核心技术

容器核心技术是指能够让 Container 在 host 上运行起来的那些技术，如图 1-2 所示。

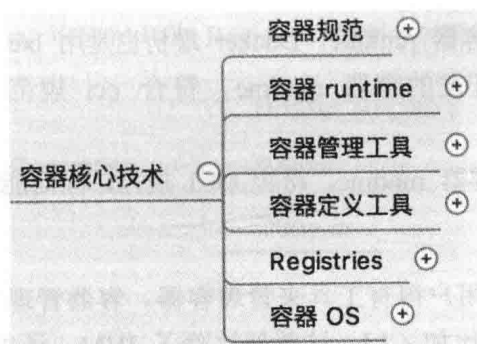


图 1-2



从上图可以看出，这些技术包括容器规范、容器 runtime、容器管理工具、容器定义工具、Registry 以及容器 OS，下面分别介绍。

### (1) 容器规范

容器不光是 Docker，还有其他容器，比如 CoreOS 的 rkt。为了保证容器生态的健康发展，保证不同容器之间能够兼容，包含 Docker、CoreOS、Google 在内的若干公司共同成立了一个叫 Open Container Initiative (OCI) 的组织，其目的是制定开放的容器规范。

目前 OCI 发布了两个规范：runtime spec 和 image format spec。

有了这两个规范，不同组织和厂商开发的容器能够在不同的 runtime 上运行。这样就保证了容器的可移植性和互操作性，如图 1-3 所示。

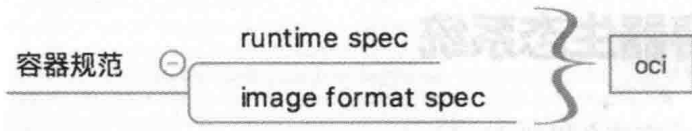


图 1-3

### (2) 容器 runtime

runtime 是容器真正运行的地方。runtime 需要跟操作系统 kernel 紧密协作，为容器提供运行环境。

如果大家用过 Java，可以这样来理解 runtime 与容器的关系：

Java 程序就好比是容器，JVM 则好比是 runtime，JVM 为 Java 程序提供运行环境。同样的道理，容器只有在 runtime 中才能运行。

lxc、runc 和 rkt 是目前主流的三种容器 runtime，如图 1-4 所示。

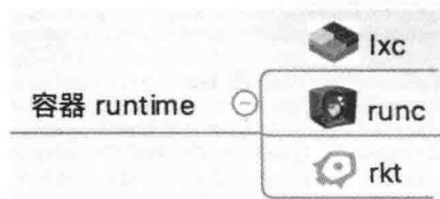


图 1-4

lxc 是 Linux 上老牌的容器 runtime。Docker 最初也是用 lxc 作为 runtime。

runc 是 Docker 自己开发的容器 runtime，符合 oci 规范，也是现在 Docker 的默认 runtime。

rkt 是 CoreOS 开发的容器 runtime，符合 OCI 规范，因而能够运行 Docker 的容器。

### (3) 容器管理工具

光有 runtime 还不够，用户得有工具来管理容器。容器管理工具对内与 runtime 交互，对外为用户提供 interface，比如 CLI。这就好比除了 JVM，还得提供 Java 命令让用户能够