《第一本Docker书》

图书基本信息

《第一本Docker书》

内容概要

全球第一本Docker技术图书中文版，Docker中文社区鼎力支持！
Docker核心团队成员权威著作，在技术圈中很有影响力。
既是第一本Docker书，也非常适合作为学习Docker的第一本入门书。
Docker是一个开源的应用容器引擎，让开发者可以将他们的应用和依赖包打包到一个可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化。容器完全使用沙箱机制，相互之间不会有任何接口。几乎没有性能开销，可以很容易地在机器和数据中心中运行。最重要的是，它不依赖于任何语言、框架或包装系统。
Docker 是一个开源的应用容器引擎，开发者可以利用Docker 打包自己的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的Linux 机器上，也可以实现虚拟化。
本书由Docker 公司前服务与支持副总裁James Turnbull 编写，是权威的Docker 开发指南。本书会指导读者完成Docker 的安装、部署、管理和扩展，带领读者经历从测试到生产的整个开发生命周期，让读者了解Docker 适用于什么场景。书中先介绍Docker 及其组件的基础知识，然后用Docker 构建容器和服务来完成各种任务：利用Docker 为新项目建立测试环境，演示如何使用持续集成的工作流集成Docker，如何构建应用程序服务和平台，如何使用Docker 的API，如何扩展Docker。
本书适合对Docker 或容器开发感兴趣的系统管理员、运维人员和开发人员阅读。
图书评价：
DevOps未死，ContainerOps已到
发现Docker项目还是2013年中，我正在为构架一个Micro Service的游戏云而测试各种PaaS平台和产品。研究CloudFoundry的过程中，被Warden子项目吸引，转而在GitHub中寻找类似的、更容易使用和部署的容器虚拟化解决方案，最终一个Linux Container的框架Docker成为我的首选。2013年底在深圳举行的ECUG Con（实效云时效用户组大会）是我第一次在大型的技术会议上宣讲Docker开源技术，此时它已经被Golang社区评为2013年的十大杀手级应用，也是这次会议我开始了Docker技术布道之旅。
从LXC的框架到Container引擎，再到如今的SaaS平台，Docker在开源社区的强大推动下快速向前演进，ContainerOps平台或是Docker的下一个里程碑。
对Docker研究得越多，就越容易被它在网络、安全方面的各种问题所困扰，忘却了Docker使用Union FileSystem技术带来的巨大技术变革的机会。当超越容器虚拟化引擎的标签去看Docker时，发现它是实现应用版本管理的最佳技术选择。比起从源代码的某个分支或标签起构建应用的版本，Union FileSystem更适于实现从开发到运维的版本管理。随着OverlayFS被Linux内核3.18合并到主干，Docker也会在最新的版本中支持它（也许在你读这本书的时候就已经支持了）。不管是AUFS还是OverlayFS，将摆脱被认为是嵌入式设备的文件格式，成为应用版本管理的技术基石。
在一次技术布道之后，有听众和我交流如何使用Gnome Desktop的Docker容器为团队提供标准的Android开发环境。正值Docker在刚刚发布的1.2版本中加入了Device特性，我建议他可以使用这个特性为Desktop加入真机的调试功能。此时我才意识到开发环境甚至是桌面环境是可以通过Docker容器来实现统一的。当微软公司和Docker深度合作的新闻震惊所有人时，才发现微软早在多年前就布局容器虚拟化的技术。Windows成为最后一个（FreeBSD有容器引擎Jails，Solaris有容器引擎Zones）能运行容器的主流操作系统。Windows操作系统可以通过容器化技术运行多个Windows的容器，Docker引擎也终于有了打通所有平台的机会。不管是Linux还是Windows，开发环境最终都可以被容器管理起来，开发配置管理将会变得非常简单。
当软件的开发环境、版本管理、交付和运行都以Docker为工具Container为基础进行流转时，就构成了以Container为核心的开发和运维流程，软件的构架也因此发生改变（Micro Service的构架方式可能会因此流行）。但持续集成、持续部署和自动化运维等生产理念没有改变，只是增加了Container的解决方案，未来必定会有基于Docker的平台来管理整个开发和生产的流程。
DevOps未死，ContainerOps已到。
在此感谢三位译者李兆海、刘斌和巨震的辛苦工作，把第一本Docker技术书籍带入中国。这不仅是一本Docker技术的入门书籍，也介绍了很多Docker的最佳实践，是学习Docker的绝佳选择。尽管没有参与此书的翻译，甚为遗憾，但我会继续努力在国内推广Docker开源技术。
马全一

《第一本Docker书》

Docker 中文社区和 docker.cn 项目创始人，Docker 开源技术布道师，资深构架师

这本书用了大量简短可操作的程序实例介绍Docker的工作原理，几乎页页都是满满的代码干货，程序员读者可跟着这些例子自己动手玩转Docker，这真是一部专为程序员写的好书！
——毛文波，道里云CEO，曾创建EMC中国实验室并担任首席科学家，曾参与创建HP中国实验室
这本书由曾任职于Docker公司的资深工程师编写，由国内社区以最快的速度完成翻译，是学习Docker的最佳入门书籍。如果你是一位希望让自己的代码运行在云端的程序员，现在就开始学习Docker吧！
——喻勇，Cloud Foundry社区创始人
正是因为Docker将对传统IT技术带来“革命性”的冲击，所以我们看到围绕Docker的创业项目如火如荼。IT从业人员应该及早拥抱Docker，拥抱变化。阅读本书就是最佳入门途径。
——陈轶飞，原百度PaaS平台负责人，国内最早大规模应用Docker的实践者
Docker今天已经算是明星技术了，各种技术大会都会有人谈论它，越来越多的人像我一样对这门技术着迷。Docker的发展异常迅猛，整个社区生态蓬勃向上一片繁荣。希望阅读本书的读者也尽快加入充满乐趣的Docker大家庭中来。
——程显峰，MongoDB中文社区创始人，蓝海讯通COO
本书系统而又深入浅出地介绍了与Docker部署和应用相关的各个方面，体现了Docker的最新进展，并附有大量详尽的实例。无论系统架构师、IT决策者，还是云端开发人员、系统管理员和运维人员，都能在本书中找到所需的关于Docker的内容。本书非常适合作为进入Docker领域的第一本书。
——商之狄，微软开放技术（中国）首席项目经理
我很高兴能看到第一本引进国内的Docker技术书籍——这本《The Docker Book》中文版。这本书对于迫切想了解Docker技术以及相关工具使用的技术爱好者来说，是一本值得阅读的入门书籍。
——肖德时，InfoQ《深入浅出Docker》专栏作者
阅读本书，就像参加一个Docker专家的面授课程，书中包含了很多非常实用的小型案例，让你能够循序渐进地照着学习，加深理解。James Turnbull是个写书的高手，由浅入深地慢慢引领你理解Docker的奥秘。无论你是哪个行业的程序员，这本Docker的书绝对会让你受益匪浅。
——蔡煜，爱立信软件开发高级专家
对Docker本身，已经不用我再多说，只希望大家都看看这本书，并能积极尝试Docker。纵观IT行业历史，大的技术变革从来不是诞生于大厂商口中的金蛋，而是一小撮爱好者的小玩意儿，而Docker正是这个路子。
——赵鹏，VisualOps 创始人
Go语言是近年来IT技术发展历程中最伟大的事情，而Docker的出现则是云计算发展的重要里程碑。作为Go语言的杀手级应用，Docker推动了Go语言社区的发展。《The Docker Book》是一本Docker团队成员撰写的书，是一份难得的学习Docker技术的权威教材。我很高兴见到中文翻译能够如此迅速地跟进，这是一件了不起的事情。
——许式伟，七牛云存储CEO，《Go语言编程》作者
我非常喜欢《The Docker Book》这本书，它弥补了开源项目通常缺失的文档部分。书中涉及从安装入门到业务场景下的各种应用及开发。本书作者的权威性以及译者的专业态度也保证了这本书的严谨性。这本书非常适合广大的Docker爱好者阅读。
——杜玉杰，OpenStack基金会董事

《第一本Docker书》

作者简介

James Turnbull是一位技术作家，还是一名开源极客。他最近的大作是一本讲述流行开源日志工具的书——The LogStash Book。James还写了两本关于Puppet的书，一本是Pro Puppet以，另一本是较早的Pulling Strings with Puppet: Configuration Management Made Easy。此外，James还写了Pro Linux System Administration、Pro Nagios 2.0和Hardening Linux这三本书。
James真正的工作是Kickstarter的工程副总裁。之前，James曾担任Docker公司服务与支持副总裁、Venmo公司工程副总裁和Puppet Labs的技术运维副总裁。James热爱美食、美酒、阅读、摄影，还很喜欢猫咪，但对在海滩上手牵手散步却并不热衷。
译者简介
李兆海 网名Googol Lee。使用Googol这个名字真的是因为“10的100次方”这个意思，和后来的Google公司没有一点儿关系。多年后端程序员，早期以C、C++为主，后来转向Python，现在以Go为生。曾写过《Golang初探》发表于2011年2月号《程序员》。Docker早期使用者。平时喜欢乱翻书，遇到感兴趣的都会研究一番。Twitter账户@googollee。
刘斌 具有10余年软件开发经验，关注后台开发技术和各种编程语言。做过电子商务、金融、企业系统以及Android手机开发；写过Delphi，也兼做系统管理员和DBA（现在都改叫DevOps了）；既做后台应用，也要调用前台CSS和JavaScript，可还是不敢自称Full Stack；今又舶来Growth Hacker，我想我要做一个Growth Engineer。
巨震 北京大学软件工程硕士，服务器端开发者。目前就职于创业公司，使用Node.js、Golang进行服务器端开发。2013年底开始研究Docker，是Docker中文社区的活跃贡献者，负责Docker技术文章和视频的翻译、校对工作。生活中喜欢美食、骑行，热衷于PC硬件，喜爱折腾，热爱一切计算机相关的技术，坚信技术改变世界。最崇拜的技术传奇人物是前id Software首席程序员、现Oculus VR首席技术官John Carmack。

《第一本Docker书》

书籍目录

《第一本Docker书》

精彩短评

1、比较简短，可以有一个大概的了解，但是时间还是跟着docker官方文档实践比较好
2、非常适合入门 有一定经验的 看了收获并不会很大
3、基础入门例子，还行
4、迅速翻过，像一本step by step的手册。——一个5年前做过容器的人
5、看完该看第二本了，嗯。。。
6、只能说算得上是一本入门实践参考书。
7、docker的step by step手册，从部署到使用再到CI，比较全面的步骤说明。因为有以前虚拟化的基础，所以理解起来不费力。当然，前提是懂linux系统，网络协议，ruby/python等脚本语言。
8、确实如书名，第一本看这本就行了，有基础有扩展，不深究原理快随上手，还有使用场景的章节。
9、从整体上阅读，开始了解规划自己的知识。第一本书也是一本入门书籍，快速了了解 docker 的基本使用方法，以及 docker 能用来干什么。剩下的就是好好吸收消化官方稳定上面的资料了。
10、没事别瞎比折腾了，还是读点书，涨点知识吧~
11、作为docker操作入门手册非常不错，知识点都讲到了（按照linux环境讲解），只有操作没有原理，推荐对docker想了解的同学
12、入门很好
13、docker入门不错
14、总体来说，没有传说中那么好。作为入门级的使用手册还行，想了解还是应该看源码。
15、docker使用说明书
16、简单了解最基本的docker用法。
17、涉及的知识面好广，有许多知识点不实际使用恐怕是无法消化的
18、第一本 恩。。。第一本 送人了
19、- 之前看过一本书《集装箱改变世界》，集成化的集装箱的出现促使了美国海港运输的发展。同样的思路也在改变着世界，只是这次是另外一个世界。
- 这本书其实和官方文档差别不大，甚至稍微有点过时，但是可操作性还是挺好的，我就是以这本为纲参照官方文档。
20、入门必备
21、读后可以了解一部分原理，可以参考示例使用docker。
但是整体来说都没说透。原理浅尝即止，示例将将够用。做原理理解不行，做手边字典也不行。唉！入门吧。书名翻译的很好。
22、偏用户手册类书籍
23、真.刷书
24、简洁，没有废话
25、写了读书笔记 http://mp.weixin.qq.com/s/6sYvwIv4VupSka5yyKSdfQ
26、docker入门
27、深入浅出，对容器的应用场景讲的比较清楚。不过对原理解释的比较少，不过仍然是值得读的书。
28、应该是第4次翻这本书了。。。
29、书挺简单，一天翻完，当然没全部看懂。不过感觉入门挺好的，代码和图很详实。
30、内容不错，代码多了些，有点水，整体感要自己去理解
31、入坑！
32、推荐未接触过docker的初学者阅读的“ docker上手指南 ”
33、第一本看这本就行了，有基础有扩展，不深究原理快随上手，还有使用场景的章节。简单翻过而已

34、新手还是看官方文档吧，去年的书，一些内容已经过时了
35、看的第一本docker类的书，主要是初步了解。内容还不错，大致清楚基本概念。

36、有点老，不过带领读者入个门还是绰绰有余的，至于后面就得看官网的文档了。
37、作为入门书足够了，书中只有用法，没有对原理和技术细节的深入描述。
38、还行吧，讲的不深
39、入门书籍
40、作为入门可以
41、实用的工具书
42、相对全面的hello world介绍，花三个小时就能读完
43、挺好啊，很实用的书。用过虚拟机就能看得懂。没用过虚拟机也可能看得懂
44、还不错的入门书，确实适合做第一本，代码粘的有些略多，所以看着其实蛮快，对Docker有基本认识的人应该一两天就看完了，第五章和第六章很啰嗦，第七章开始就显示出本书的过时了。
45、内容很浅，适合初学者。每一章都手把手教你怎么运用docker，要看这本书建议先看清楚docker对安装环境的要求，确定能够有个docker环境进行学习
46、一个月前翻过，内容一般般。。。
47、稍过时 2016年11月
48、就那样呗, 比看文档快, 但做事情的时候还是要 RTFM. 错别字略多, 学长不给力
49、果然是第一本docker书，我觉得用来入门还可以。
50、基础应用形式的介绍 对原理讲的不多

《第一本Docker书》

精彩书评

1、作為初學者的首選工具書，這本書在內容編排 語言表達 等方面做的都非常好，讓讀者閱讀起來非常輕鬆，毫無枯燥感。作為一本docker學習的工具書，告诉读者docker是什么，它能为我们解决什么等，也得到了專家的認可，不愧第一，是一本值得一讀的好書
。----------##
2、把The Docker Book 翻译成 第一本Docker书 可能是中文版出版方想突出这本书在Docker世界的地位。通读后觉得书如其名，此书可以作为Docker学习的第一本书。篇幅不大，200多页、字体不小、且代码示例较多，让人读起来很轻松，比动辄上千页密密麻麻小子的xx权威指南，xx实战要来的轻松多。跟着书中的例子做下来对Docker就有了直观的印象，当然想用到实际开发中读这“ 第一本书“ 是不够的，还需要进阶的材料和实践。看得出来作者在选择例子时下了一些功夫，把他丰富的项目开发经验注入了进去，从例子中可管窥大型软件的开发、部署的样貌。最后不得不提一句，由于版本升级，书中一些构建的例子已经无法正常运行，但作者托管代码的github一直保持着更新，如果按照书上例子操作无法运行，可去github上下载最新的代码。向作者的严谨态度致敬。总之一句话，这本书适合入门的小白。如果想深入了解Docker相关技术或者进阶，这本书并不适合你。

《第一本Docker书》

章节试读

1、《第一本Docker书》的笔记-第1页

由于“客居”于操作系统，容器只能运行于与底层宿主机相同或相似的操作系统……结果是 Linux 上的容器只能虚拟 Linux 环境，使得容器的灵活性不如虚拟机，但是容器直接运行在操作系统内核之上，与虚拟机相比具有性能优势。
和传统的虚拟化以及半虚拟化（paravirtualization）技术相比，容器运行不需要模拟层（emulation layer）和管理层（hypervisor layer），而是使用操作系统的系统调用接口。这降低了运行单个容器需要的开销，使得宿主机中可以运行更多的窗口。因此，单纯对这两种技术进行比较的话，容器其实拥有更为广泛的应用场景。毕竟，用 Linux 的话，虚拟化之后得到 Linux 环境往往更符合应用需求，在 Linux 上装个虚拟机跑 Windows 不是闲得蛋疼吗？

2、《第一本Docker书》的笔记-读书笔记

1. Docker简介：

Docker与传统虚拟化容器的不同之处在于Docker在虚拟化的容器环境中又增加了一个应用程序的部署引擎。这个引擎提供了一个轻量快速的环境，能够运行开发者的程序，并且方便高效地将程序从开发者的笔记本部署到测试环境和生产环境。Docker的目标是提供以下这些东西：
<1> 提供一个简单，轻量的建模方式。Docker很容易上手，用户很容易将自己的程序Docker化。另外Docker容器的创建非常快捷，大多数的Docker容器只需要不到一秒钟就可以启动，由于去除了管理程序的开销，所以Docker具有很高的性能，同一台宿主机可以运行很多的容器。
<2> 职责的逻辑分离。使用Docker，开发者只需要关心容器中的应用程序，运维人员只需要关心容器的管理。
<3> 快速高效的开发生命周期。Docker的目标之一就是缩短从开发到测试到部署上线的周期，使程序容易构建，容易移植。
<4> 鼓励面向服务的架构。由于Docker鼓励每个容器只运行一个应用程序，所以这样就形成了一个分布式的应用程序模型。在这种模型下，所有的项目都会变成一系列内部互联的容器，从而使得分布式地部署应用程序架构。

Docker的架构大致如此：

Docker是一个典型的CS架构，客户端只需要向服务端或者守护进程发出请求，然后服务端和守护进程会完成所有工作并且返回结果。镜像是Docker世界的基石，用户基于镜像运行和构建自己的容器，可以把镜像看做是容器的源代码。另外，Docker使用Registry保存用户构建的镜像。

Docker带给我们的好处分成以下几点：

<1> 加速本地开发和构建流程。本地开发人员可以构建，分享和部署Docker容器。
<2> 让独立的服务和应用程序在不同的环境中得到相同的结果。
<3> 创建一个隔离的环境进行测试。例如用Jenkins这样的集成测试工具创建一个新的用于测试的容器。
<4> Docker可以让开发者在本机就搭建好一个复杂的程序或架构，而不是上来就搞服务器。
<5> 为学习和测试构建一个轻量级的沙盒环境。
<6> 超大规模的宿主机部署

另外Docker还有个显著的特点是对不同的宿主机，应用程序和服务，可能会表现出不同的特性和架构

，这样就进一步简化了应用程序开发的工作。另外，由于Docker的容器非常轻量级，所以重建容器的代价通常都比传统的状态修复低得多。

2. Docker常用命令

首先来创建第一个容器： sudo docker run -I -t -p 127.0.0.1:80:9001 ubuntu /bin/bash。这个命令解析一下：-i的意思是保证容器的STDIN是开启的，保证持久化的标准输入。 -t是为创建的容器分配一个伪tty中断，这样新创建的容器才能提供一个交互式shell。 除非这是一个运行后台服务的容器，否则这两个参数是最基本的参数。Docker首先会检查本地是否存在Ubuntu镜像，如果本地没有该镜像的话，Docker就会连接官方维护的Docker Hub，一旦找到就会下载保存到本地的宿主机中。-p参数会将Docker的网络端口和宿主机的端口相绑定。最后告诉容器要运行/bin/bash命令，启动一个bash shell。

docker ps 会列出当前系统中正在运行的容器，docker ps -a会列出当前系统的所有容器列表。

docker run --name xxxx -i -t ubuntu 可以为容器指定一个名称，在很多Docker命令中，都可以使用name来代替uuid，容器的名称有助于分辨容器，也有助于从逻辑上帮我们理清容器关系。

docker start 和 docker restart 可以帮助我们启动或者重启一个容器，可以使用docker stop关闭守护式容器。

docker attach 可以让我们重新附着到容器的会话上。

docker run -d 可以帮助我们创建一个守护进程放在后台长期运行。如果想查看后台容器都做了什么，可以使用docker logs ubuntu来获取容器的日志。与tail -f同理，也可以使用docker logs -f来监控实时的日志。

docker top 可以监控某容器中的运行进程。在Docker 1.3之后，还可以使用docker exec在容器内额外启动新进程，可以在容器内启动的进程分成后台任务和交互式任务两类。例如sudo docker exec -d daemon touch /etc/new_config中的-d就启动了一个后台任务，而sudo docker exec -t -i deamon /bin/bash则是启动了一个交互式任务。

docker inspect 获得更多的容器信息，这个命令会对容器进行详细的检查，然后返回配置信息，包括名称，网络配置和很多有用的数据。使用--format 或 -f 支持完整的Go语言，所以可以充分发挥Go语言的模板优势。

docker rm可以删除容器。

3. Docker镜像

Docker镜像是由文件系统叠加而成，最底端是一个文件引导系统，即bootfs，很类似于Linux的文件引导系统，用户大部分时间不会和引导文件系统有什么交互。再往上的第二层是root文件系统rootfs，rootfs可以是一种或者多种操作系统。

在传统Linux系统引导中，root文件系统会最先以只读的方式加载，当引导结束后才会变成读写状态。但是在Docker中，rootfs一直都会保持只读状态，并且Docker会利用联合加载技术在rootfs上加载更多的只读文件系统(联合加载是说通过叠加的方式一次同时加载多个文件系统，但在外面又只能看到一个文件系统)。Docker将这样的文件系统称为镜像。

Docker第一次启动一个容器时，初始的读写层是空的，当文件系统发生变化时，这些变化都会应用到这一层上，例如想修改一个文件，这个文件会从该读写层下面的只读层复制到读写层，然后修改，这时指针依然指向只读版本，直到修改结束，这时读写层又变成了只读状态。这被称为写时复制。

构建Docker镜像有两种方法：<1> 使用docker commit命令 <2> 使用docker build和Dockerfile文件。并不推荐使用docker commit，而应该使用更灵活强大的dockerfile方式。Dockerfile使用基本的DSL语法指令来构建一个Docker镜像，然后使用docker build命令基于该Dockerfile中的指令构建一个新的镜像。Dockerfile由一系列的指令和参数组成。Dockerfile的指令会按顺序从上到下执行，所以应该根据需要合理安排制定的顺序。每条指令都会创建一个新的镜像并对镜像进行提交，Docker大致按照以下的步骤执行Dockerfile中的指令: Docker从基础镜像运行一个容器，执行一条指令并且对容器做出修改，执行类似docker commit的操作提交一个新的镜像层，Docker基于刚刚提交的容器创建一个新的镜像层，执行Dockerfile下一条指令直到执行结束。

每个Dockerfile的第一条指令都应该是FROM，后续指令都会基于该镜像进行，这个镜像称为基础镜像。由于Docker会将每一步构建过程都提交为镜像，所以他会将之前的镜像层看做是缓存，这样如果失败再次进行构建时不需要从头开始构建，这样可以节省大量的时间。但是这个带来的问题是比如apt-get update的命令也被缓存起来，所以可以执行REFERSHED_AT来制定最后构建时间。

4. 使用Docker进行开发测试和持续集成

sudo docker run -d -p 80 --name website -v $PWD/website:/var/www/html/website nginx /bin/nginx 中的-v参数允许我们把宿主机的目录作为卷，挂在到容器中，从而实现容器之间的共享。卷在Docker中非常重要，卷是在一个或者多个容器内被选定的目录，可以绕过分层的联合文件系统为Docker提供持久数据和共享数据。这意味着对卷的修改会直接生效，从而绕过容器本身。即便容器停止，卷的内容也依然存在。例如：<1> 希望同时对代码进行开发和测试 <2> 代码修改频繁，不想频繁在开发过程中构建镜像。 <3> 希望在多个容器中共享资源。

另外的一个应用场景，例如我们搭建了一个Web的容器和一个Redis的容器，这样容器之间需要互相通信。有两种办法：<1> 将容器的网络端口绑定到宿主机的网络端口，从而访问宿主机的网络端口 <2> 内部网络。在安装Docker时，会创建一个新的网络接口，名字叫做Docker0，每个Docker容器都会在这个接口上分配一个IP地址。Docker0是一个虚拟的以太网桥，用于连接容器和本地宿主网络，所以Docker每创建一个容器就会创建一组互联的网络接口，这组接口一段在Docker0的一端，另外一端在容器上，所以其实就相当于通过Docker0作为整个内部网络的交换机。

另外，Docker很擅长快速创建和处理一个或多个容器，这个能力显然可以为持续集成测试提供帮助。Docker可以让部署以及这些步骤和宿主机的清理变得开销很低。

5. 插件

Fig由Orchard团队开发的开源工具，用Python编写，用于简单的容器编排。使用Fig可以用一个YAML文件定义一组要启动的容器，以及容器运行时的属性，Fig称这些容器为服务，并且这样定义：容器通过某些方法并指定一些运行时的属性来和其他容器产生交互。使用中可以看出使用Fig能够非常简单地构建一个需要多个Docker容器的应用程序。

服务发现是分布式程序之间管理相互关系的一种机制，一个分布式程序一般由多个组件组成，这些组件可以放在同一台机器上，也可以分步在多个数据中心，甚至分步在不同的地理区域。这些组件通常可以为其他组件服务，或者为其他组件提供消费服务。所谓服务发现的概念是允许某个组件在想要和其他组件交互时，自动找到对方，由于这些应用本身是分布式的，所以服务发现机制本身也需要是分布式的。而且服务发现作为分布式应用不同组件之间的胶水，还需要足够动态，可靠，适应性强，并且可以快速且一致地共享关于这些服务的数据。书中是使用Consul作为服务发现的工具，Consul是使用Go开发的工具，使用了Raft一致性算法来提供确定的写入机制。Consul暴露了键值存储系统和服务分类系统，并提供高可用性，高容错能力，并且保证强一致性。服务可以将自己注册到Consul，并以高可用且分布式的方式共享这些信息。另外，Consul还内置了强大的服务监控系统。

编配工具是一个快速发展的领域，工具的功能不尽相同，但是大多属于两种类型：调度和集群管理，服务发现。

3、《第一本Docker书》的笔记-33 Docker入门

Docker开始

4、《第一本Docker书》的笔记-第1页

《第一本Docker书》

第一章 简介

镜像是构建docker的即使，用户基于镜像来运行自己的容器。镜像是基于联合（Union）文件系统的一种层式的结构。
镜像是Docker声明周期中的构建或打包阶段，而容器则是启动或执行阶段
黄金镜像模型 -> Docker分层镜像模型

第三章 Docker入门

创建 -> 管理 -> 停止 -> 删除

docker run
    docker run -i -t ubuntu /bin/bash
    -i STDIN
    -t tty
    —name 容器命名
    —restart (always / on-failure:5)

docker ps -a
    查看当前系统中容器的列表
    -l 最后一次运行的容器

docker start bob_the_container
    重新启动已经停止的容器

docker attach
    附着到容器上?（使用场景？？？）

docker run —name daemon_dave -d ubuntu /bin/sh -c ‘ …...'
    执行守护式容器

docker logs

docker top

docker exec
    在容器内部额外启动新进程（使用场景？？？）

docker stop
    停止守护式容器

docker inspect

docker rm sdf123132
    删除容器

第四章 使用Docker镜像和仓库

什么是docker镜像？

Docker镜像是由文件系统叠加而成。最低端是一个引导文件系统bootfs。第二层是root文件系统rootfs，为了bootfs之上。联合加载（union mount）同时加载多个只读文件系统，但是在外面看起来是一次同时加载一个文件系统。Docker将这样的文件系统称为镜像。

当第一次启动容器时，Docker会在该镜像的最顶层加载一个读写文件系统。当Docker第一次启动一个容器时，初始的读写层是空的。当文件系统发生变化时，这些变化都会应用到这一层上。比如，如果想修改一个文件，这个文件首先会从该读写层下面的只读层复制到该读写层（copy-on-write）。改文件的只读版本依然存在，但是已经被读写层中的该文件副本所隐藏。(image-layering framework)

构建镜像

- docker commit：提交一个新的镜像层
- Dockerfile

用Dockerfile创建镜像
命令
FROM ：指定一个已经存在的镜像
MAINTAINER：维护人
RUN：每条RUN指令都会创建一个新的镜像层
EXPOSE: 告诉Docker该容器内的应用程序将会使用容器内的指定端口。处于安全考虑，Docker并不会自动打开该端口，而是需要在docker
run运行时指定需要打开哪些端口（EXPOSE也可以帮助多个容器链接）
CMD：和RUN类似，RUN为构建竟像是需要运行的命令，CMD是容器启动时使用的命令（docker run可以覆盖CMD）
ENTRYPOINT：与CMD类似。docker run命令行中指定的任何参数都会被当做参数再次传递给ENTRYPOINT指令中的参数
WORKDIR
ENV
ADD: https://docs.docker.com/engine/articles/dockerfile_best-practices/#add-or-copy

COPY
USER
VOLUME：这个目录可以绕过联合文件系统，并提供如数据共享或者持久化的功能
ONBUILD

docker rmi【对比docker rm】

docker build

docker run -d -p 80 —name static_web jamtur01/static_web nginx -g “ daemon off; ”【需要nginx以前台的方式启动作为我们的服务器】

docker可以通过两种方法在宿主机上分配端口：

- 随机选择一个49000~49900
- 指定一个特定的端口

docker port 查看端口映射方式

第五章 在测试中使用Docker

静态网站
1.-v 挂载镜像

- ADD/COPY

Web应用
构建Sinatra
构建Redis镜像

- 构建Redis基础镜像
- 构建Redis Master
- 构建Redis Slave
- 关联

Sinatra关联Redis
方法：

- Docker自己的网络栈（内部网络）：Docker创建了一个虚拟子网，这个子网由宿主机和所有的Docker容器共享/防火墙。如果直接使用内网IP，则可能产生硬编码。 ==》Docker link
  - docker run -d —name reds xxx/redis
  - docker run -p 4567 —name web —link redis:db -t -i -v $PWD/webapp:/opt/webapp xxx/sinatra /bin/bash
  - —link创建链接两个父子容器。redis:db，redis为想要链接的容器（注意不是镜像），db为链接后的别名
  - —icc=false
  - 链接信息：/etc/hosts 和 环境变量【Dockerfile中的ENV和EXPOSE】

Jenkins
在Docker中运行Docker

第六章 使用Docker构建服务

docker run -d -P —volumes-from james_blog xxx/apache
—volumes-from 把指定容器里的所有卷都加入新创建的容器里【！！！！如果用docker rm删除了james_blog，那么卷和卷里的内容也就不存在了】

更新内容
docker start james_blog

备份卷

多容器（nodejs + redis集群）
1.redis base

docker run -h

捕获应用日志

logstash
docker run -d —name log stash —volumes-from redis_primary —volumes-from nodded xxx/logstash

第七章 使用Fig编配Docker

多个Docker宿主机进行协作

- 简单的容器编配（Fig）
- 分布式服务发现（Consul）
- kubernetes/mesos/helios/centurion/libswarm

5、《第一本Docker书》的笔记-第2页

Docker 是一个能够把开发的应用程序自动部署到容器的开源引擎。这这这，跟 Tomcat 多像啊！

《第一本Docker书》

版权说明