## 第1章 初识Docker

* 个人主机时代：关键是CPU主频的高低和内存的大小。
* 云计算时代：虚拟化技术无疑是整座信息技术大厦最核心的一块基石。

伴随着信息技术产业的发展，虚拟化技术已经应用到各种关键场景中。从最早上世纪60年代IBM推出的大型主机虚拟化到后来X86平台上的虚拟化，虚拟化技术自身也在不断丰富和创新。

虚拟化既可以通过硬件模拟来实现，也可以通过操作系统来实现。而近年来出现的容器虚拟化方案，更是充分利用了操作系统本身已有的机制和特性，可以实现轻量级的虚拟化，甚至有人将其称为新一代的虚拟化技术，Docker就是其中的佼佼者。

### 1.1 什么是Docker

#### Docker开源项目

基于Go语言实现的云开源项目，诞生于2013年初，最初的发起者是dotCloud公司（后改名Docker Inc）。目前已有多个相关项目，逐渐形成了围绕Docker的生态体系。

Docker项目目前已加入了Linux基金会，遵循Apache 2.0协议，全部开源代码均在<https://github.com/moby/moby>上维护，最近一次Linux基金会的调查中，Docker是仅次于OpenStack最受欢饮的云计算开源项目。

主流的Linux操作系统都已支持Docker。（Redhat RHEL 6.5/CentOS 6.5往上的操作系统、Ubuntu 14.04操作系统，都已经默认带有Docker软件包。）

Docker的主要目标是“Build，Ship and Run Any App，Anywhere”，即通过对应用组件的封装（Packing）、分发（Distribution）、部署（Deployment）、运行（Runtime）等生命周期的管理，达到用应用组件级别的“一次封装，到处运行”。这里的应用组件既可以是一个Web应用，也可以是一套数据库服务，甚至是一个操作系统或编译器。

Docker是基于Linux的多项开源技术提高了效率、敏捷和轻量级的容器方案，并且支持在多种主流云平台（PaaS）和本地系统上部署。Docker为应用的开发和部署提供了“一站式”的解决方案。

#### Linux容器技术

Docker引擎的基础是Linux容器（Linux Containers，LXC）技术。IBM developerWorks上给出了关于容器技术的准确描述：

容器有效地将由单个操作系统管理的资源划分到孤立的组中，以便更好地在孤

立的组之间平衡有冲突的资源使用需求。与虚拟化相比，这样既不需要指令级模拟，也不需要即时编译。容器可以在核心CPU本地运行指令，而不需要任何专门的解释机制。此外，也避免了准虚拟化（paravirtualization）和系统调用替换中的复杂性

Linux容器其实不是一个全新的概念。最早的容器技术可以追溯到1982年Unix系列操作系统上的chroot（change root）工具（直到今天，主流的Unix、Linux操作系统仍然支持和带有该工具）。早期的容器实现技术包括Sun Solaris操作系统上的Solaris Containers（2004年发布），FreeBSD操作系统上的FreeBSD jail（2000年左右出现），以及GNU/Linux上的Linux-VServer（2001年10月）和OpenVZ（2005年）。虽然这些技术经过多年的演化已经成熟，但是这些容器技术并没有被集成到主流的Linux内核中，使用起来并不方便。例如，如果用户要使用OpenVZ技术，就需要先给操作系统打上特定的内核补丁方可使用。后来LXC被集成到了主流Linux内核中，进而成为Linux系统轻量级容器技术的事实标准。

#### 从Linux容器到Docker

在LXC的基础上，Docker进一步优化了容器的使用体验。Docker提供了各种容器管理工具（如分发、版本、移植等）让用户无需关注底层的操作，可以简单明了地管理和使用容器。用户操作Docker容器就像操作一个轻量级的虚拟机那样简单。

可以简单地将Docker容器理解为一种沙盒（Sandbox）。每个容器内运行一个应用，不同的容器相互隔离，容器之间也可以建立通信机制，容器的创建和停止都十分快速，容器自身对资源的需求也十分有限，远远低于虚拟机。很多时候，甚至直接把容器当做应用本身也没有任何问题。

### 1.2 为什么要使用Docker

#### Docker容器虚拟化的好处

高效地构建应用。现在开发者需要能方便地创建运行在云平台上的应用，也就是说应用必须能够摆脱底层机器，而且同时必须是“任何时间任何地点”可获取的。因此，开发者们需要一种创建分布式应用程序的方式，这也是Docker能够提供的。

应用场景例子：

假设用户试图基于最常见的LAMP（Linux+Apache+MySQL+HP）组合来运维一个网站。安装传统的做法，首先，需要安装Apache、MySQL和PHP以及它们各自运行所依赖的环境；之后分别对它们进行配置（包括创建合适的用户、配置参数等）；经过大量的操作后，还需要进行功能测试，看是否工作正常；如果不正常，则意味着更多的时间代价和不可控的风险。可以想象，如果再加上更多的应用，事情会变得更加难以处理。更为可怕的是，一旦需要服务器迁移（例如从阿里云迁移到腾讯云），往往需要重新部署和测试。这些琐碎而无趣的“体力活”，极大地降低了工作效率。而Docker提供了一种聪明的方式，通过容器来打包应用，意味着迁移只需要在新的服务器上启动需要的容器就可以了。这无疑将节约大量的宝贵时间，并降低部署过程出现问题的风险。

#### Docker在开发和运维中的优势

对开发和运维（DevOps）人员来说，可能最梦寐以求的就是一次性地创建或配置，可以在任意环境、任意时间让应用正常地运行。而Docker恰恰是可以实现这一要求。

具体来说，Docker在开发和运维过程中，具有如下几个方面的优势：

* 更快速地交付和部署。使用Docker，开发人员可以使用镜像来快速构建一套标准的开发环境；开发完成之后，测试和运维人员可以直接使用相同环境来部署代码。Docker可以快速创建和删除容器，实现快速迭代，大量节约开发、测试、部署的事件。并且，各个步骤都有明确的配置和操作，整个过程全程可见，使团队更容易理解应用的创建和工作过程。
* 更高效的资源利用。Docker容器的运行不需要额外的虚拟化管理程序（Virtual Machine Manager，VMM，以及Hypervisor）支持，它是内核级的虚拟化，可以实现更高的性能，同时对资源的额外需求很低。
* 更轻松的迁移和扩展。包括物理机、虚拟机、公有云、私有云、个人电脑、服务器等。这种兼容性让用户可以在不同平台之间轻松地迁移应用。
* 更简单地更新管理。使用Dockerfile，只需要小小的配置修改，就可以替代以往大量的更新工作。并且所有修改都以增量的方式进行分发和更新，从而实现自动化并且高效地容器管理。

#### Docker与虚拟机比较

作为一种轻量级的虚拟化方式，Docker在运行应用上跟传统的虚拟机方式相比具有显著优势：

* Docker容器很快，启动和停止可以在秒级实现，这比传统虚拟机方式要快得多。
* Docker容器对系统资源需求很少，一台主机上可以同时运行数千个Docker容器。
* Docker通过类似Git的操作来方便用户获取、分发和更新应用镜像，指令简明，学习成本较低。
* Docker通过Dockerfile配置文件来支持灵活的自动化创建和部署机制，提高工作效率。
* Docker容器除了运行其中的应用之外，基本不消耗额外的系统资源，保证应用性能的同时，尽量减少系统开销。传统虚拟机方式运行N个不同的应用就要启动N个虚拟机（每个虚拟机需要单独分配独占内存、磁盘等资源），而Docker只需要启动N个隔离的容器，并将应用放到容器内即可。当然，在隔离性方面，传统的虚拟机方式多了一层额外的隔离。但这并不意味着Docker就不安全。Docker利用linux系统上的多种防护机制实现了严格可靠的隔离。从1.3版本开始，Docker引入了安全选项和镜像签名机制，极大地提高了使用Docker的安全性。

|  |  |  |
| --- | --- | --- |
| **特性** | **容器** | **虚拟机** |
| 启动速度 | 妙级 | 分钟级 |
| 性能 | 接近原生 | 较弱 |
| 内存代价 | 很小 | 较多 |
| 硬盘使用 | 一般为MB | 一般为GB |
| 运行密度 | 单机支持上千个容器 | 一般几十个 |
| 隔离性 | 安全隔离 | 安全隔离 |
| 迁移性 | 优秀 | 一般 |

### 1.3 虚拟化与Docker

虚拟化技术是一个通用的概念，在不同领域有不同的理解。在计算领域，一般指的是计算机虚拟化（Computing Virtualization），或通常说的服务器虚拟化。

维基百科上的定义：

在计算机技术中，虚拟化（Virtualization）是一种资源管理技术，是将计算机的各种实体资源，如服务器、网络、内存即存储等，予以抽象、转换后呈现出来，打破实体结构间的不可切割的障碍，使用户可以用比原本的组态更好的方式来应用这些资源。

可见，虚拟化的核心是对资源进行抽象，目标往往是为了在同一个主机上运行多个系统或应用，从而提高系统资源的利用率，同时带来降低成本、方便管理和容错容灾等好处。

从大类上分，虚拟化技术可分为基于硬件的虚拟化和基于软件的虚拟化。其中，真正意义上的基于硬件的虚拟化技术不多见，少数如网卡中的单根多IO虚拟化（Single Root I/O Virtualization and Sharing Specification，SR-IOV）等技术。基于软件的虚拟化从对象所在的层次，又可以分为应用虚拟化和平台虚拟化（通常说的虚拟技术即属于这个范畴）。前者一般指的是一些模拟设备或Wine这样的软件。后者又可以细分为如下几个子类：

* 完全虚拟化。虚拟机模拟完整的底层硬件环境和特权指令的执行过程，客户操作系统无需进行修改。例如VMware Workstation、VirtualBox、QEMU
* 硬件辅助虚拟化。利用硬件（主要是CPU）辅助支持（目前x86体系结构上可用的硬件辅助虚拟化技术包括Intel-VT和AMD-V）处理敏感指令来实现完全虚拟化的功能，客户操作系统无需修改，例如VMware Workstation、Xen、KVM。
* 部分虚拟化。只针对部分硬件资源进行虚拟化，客户操作系统需要进行修改。现在有些虚拟化技术的早期版本仅支持部分虚拟化。
* 超虚拟化（Paravirtualization）。部分硬件接口以软件的形式提供给客户机操作系统，客户操作系统需要进行修改，例如早期的Xen。
* 操作系统级虚拟化。内核通过创建多个虚拟的操作系统实例（内核和库）来隔离不同的进程。容器相关技术即在这个范畴。

可见Docker以及其他容器技术都属于操作系统级虚拟化这个范畴。Docker虚拟化方式之所以拥有众多优势，这跟操作系统的虚拟化自身的特点是分不开的。下图比较了Docker和常见的虚拟机方式的不同之处：



传统方式是在硬件层面实现虚拟化，需要有额外的虚拟机管理应用和虚拟机操

作系统层。

Docker容器是在操作系统层面上实现虚拟化，直接复用本地主机的操作系统，因此更加轻量级。

通过为Linux容器（LXC）技术提供更简便的使用和管理方案、更高效的版本控制机制，Docker让容器技术一下子变得前所未有的方便易用。

## 第2章 Docker的核心概念和安装

### 2.1 核心概念

#### Docker Image

Docker Image类似于虚拟机镜像，可以将它理解为一个面向Docker引擎的只读模板，包含了文件系统。例如：一个镜像可以只包含一个完整的Ubuntu操作系统环境，可以把它称为一个Ubuntu镜像。镜像也可以安装了Apache应用程序（或用户需要的其他软件），可以把它称为一个Apache镜像。

镜像是创建Docker容器的基础。通过版本管理和增量的文件系统，Docker提供了一套十分简单的机制来创建和更新现有的镜像，用户甚至可以从网上下载一个已经做好的应用镜像，并通过简单的命令就可以直接使用。

#### Docker Container

Docker Container类似于一个轻量级的沙箱，Docker利用容器来运行和隔离应用。容器是从镜像创建的应用运行实例，可以将其启动、开始、停止、删除，而这些容器都是相互隔离、互不可见的。

可以将容器看成一个简易版的Linux系统环境（这包括root用户）