Docker

# 一、Docker简介

## 1、虚拟化

### 1.1 什么是虚拟化

在计算机中，虚拟化（英语：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、内存及存储等，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，使用户可以比原本的组态更好的方式来应用这些资源。这些资源的新虚拟部份是不受现有资源的架设方式，地域或物理组态所限制。一般所指的虚拟化资源包括计算能力和资料存储。

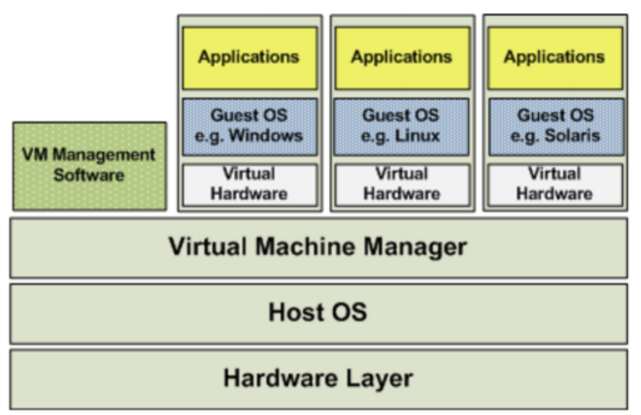
在实际的生产环境中，虚拟化技术主要用来解决高性能的物理硬件产能过剩和老的旧的硬件产能过低的重组重用，透明化底层物理硬件，从而最大化的利用物理硬件 对资源充分利用

虚拟化技术种类很多，例如：软件虚拟化、硬件虚拟化、内存虚拟化、网络虚拟化(vip)、桌面虚拟化、服务虚拟化、虚拟机等等。

### 1.2 虚拟化种类

（1）全虚拟化架构

虚拟机的监视器（hypervisor）是类似于用户的应用程序运行在主机的OS之上，如VMware的workstation，这种虚拟化产品提供了虚拟的硬件。



名词解释：

Hardware Layer 本地硬件层

Host OS 本地操作系统

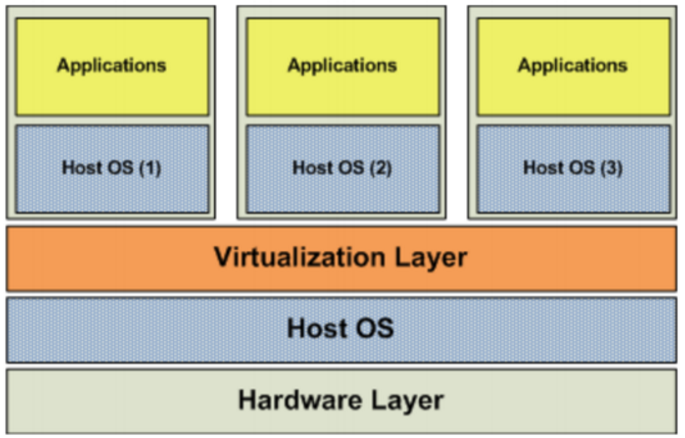
Virtual Machine Manager 虚拟软件管理系统

Virtual Hardware 虚拟硬件

Applications 应用程序

VMware就是全虚拟化架构

（2）OS层虚拟化架构（不对硬件进行虚拟化）



名词解释：

Virtualization Layer 虚拟化层

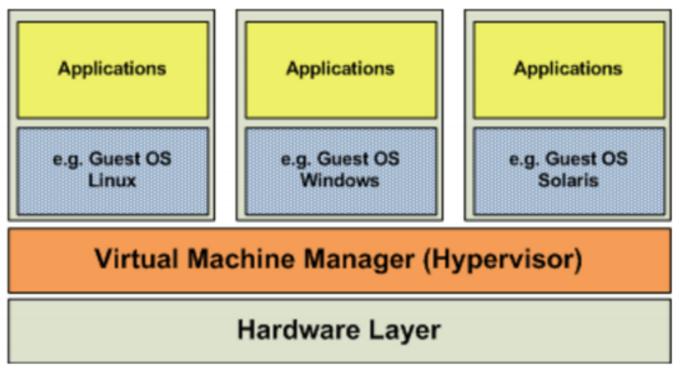
优点：

比全虚拟化架构运行效率快

缺点：

因为没有虚拟硬件层，所以两个操作系统必须是一类的，如windows和windows、linux和linux

（3）硬件层虚拟化



硬件层的虚拟化具有高性能和隔离性，因为hypervisor直接在硬件上运行，有利于控制VM的OS访问硬件资源，使用这种解决方案的产品有VMware ESXi 和 Xen server

Hypervisor是一种运行在物理服务器和操作系统之间的中间软件层,可允许多个操作系统和应用共享一套基础物理硬件，因此也可以看作是虚拟环境中的“元”操作系统，它可以协调访问服务器上的所有物理设备和虚拟机，也叫虚拟机监视器（Virtual Machine Monitor，VMM）。

**Hypervisor是所有虚拟化技术的核心**。当服务器启动并执行Hypervisor时，它会给每一台虚拟机分配适量的内存、CPU、网络和磁盘，并加载所有虚拟机的客户操作系统。 宿主机Hypervisor是所有虚拟化技术的核心，软硬件架构和管理更高效、更灵活，硬件的效能能够更好地发挥出来。常见的产品有：VMware、KVM、Xen等等。Openstack

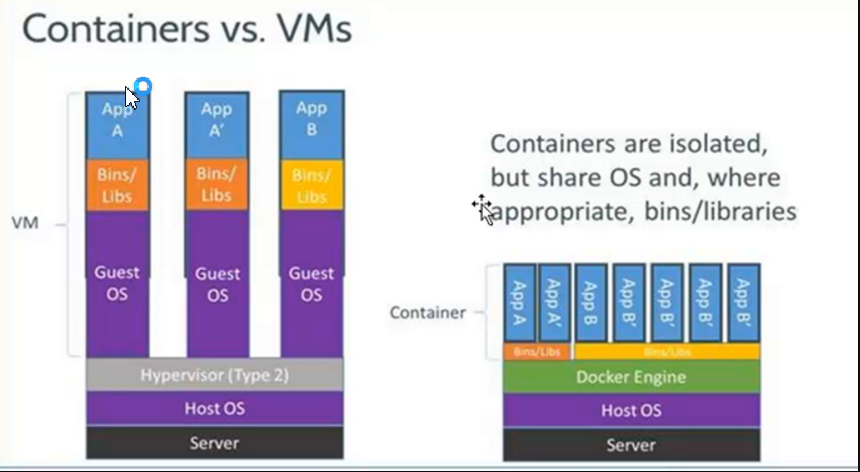
### 1.3 什么是Docker（集装箱）

容器与管理程序虚拟化（hypervisor virtualization，HV）有所不同，管理程序虚拟化通过中间层将一台或者多台独立的机器虚拟运行与物理硬件之上，而容器则是直接运行在操作系统内核之上的用户空间。因此，容器虚拟化也被称为“操作系统级虚拟化”，容器技术可以让多个独立的用户空间运行在同一台宿主机上。

由于“客居”于操作系统，容器只能运行与底层宿主机相同或者相似的操作系统，这看起来并不是非常灵活。例如：可以在Ubuntu服务中运行Redhat Enterprise Linux，但无法再Ubuntu服务器上运行Microsoft Windows。

#### 1.3.1 容器与虚拟机的比较

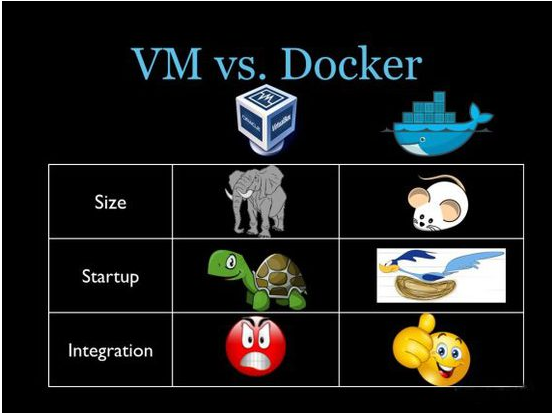
（1）本质上的区别



虚拟机每一层占的空间都很大

（2）使用上的区别

大小、运行速度、软件集成



#### 1.3.2 Docker特点

（1）上手快

（2）职责的逻辑分类

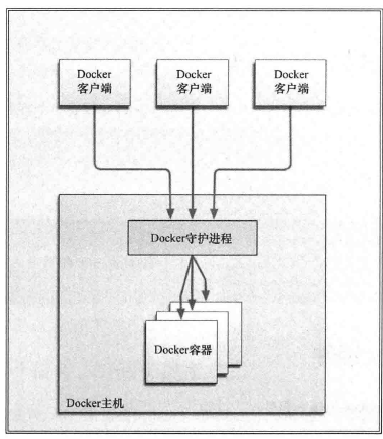
（3）快速高效的开发生命周期

（4）鼓励使用面向服务的架构

### 1.4 Docker组件

#### 1.4.1 Docker客户端和服务器

Docker是一个客户端-服务器（C/S）架构程序。Docker客户端只需要向Docker服务器或者守护进程发出请求，服务器或者守护进程将完成所有工作并返回结果。Docker提供了一个命令行工具Docker以及一整套RESTful API。你可以在同一台宿主机上运行Docker守护进程和客户端，也可以从本地的Docker客户端连接到运行在另一台宿主机上的远程Docker守护进程。



#### 1.4.2 Docker镜像

镜像是构建Docker的基石。用户基于镜像来运行自己的容器。镜像也是Docker生命周期中的“构建”部分。镜像是基于联合文件系统的一种层式结构，由一系列指令一步一步构建出来。例如：

添加一个文件；

执行一个命令；

打开一个窗口。

也可以将镜像当作容器的“源代码”。镜像体积很小，非常“便携”，易于分享、存储和更新。

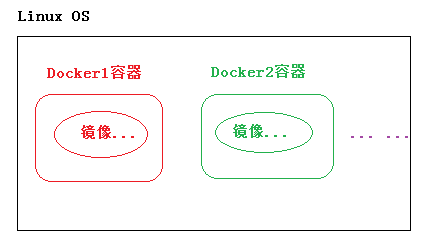
#### 1.4.3 Registry（注册中心）

Docker用Registry来保存用户构建的镜像。Registry分为公共和私有两种。Docker公司运营公共的Registry叫做Docker Hub。用户可以在Docker Hub注册账号，分享并保存自己的镜像（说明：在Docker Hub下载镜像巨慢，可以自己构建私有的Registry）。

#### 1.4.4 Docker容器

容器和镜像之间的关系：类似对象和类之间的关系。

Docker可以帮助你构建和部署容器，你只需要把自己的应用程序或者服务打包放进容器即可。容器是基于镜像启动起来的，容器中可以运行一个或多个进程。我们可以认为，镜像是Docker生命周期中的构建或者打包阶段，而容器则是启动或者执行阶段。 容器基于镜像启动，一旦容器启动完成后，我们就可以登录到容器中安装自己需要的软件或者服务。



所以Docker容器就是：

一个镜像格式；

一些列标准操作；

一个执行环境。

Docker借鉴了标准集装箱的概念。标准集装箱将货物运往世界各地，Docker将这个模型运用到自己的设计中，唯一不同的是：集装箱运输货物，而Docker运输软件。

和集装箱一样，Docker在执行上述操作时，并不关心容器中到底装了什么，它不管是web服务器，还是数据库，或者是应用程序服务器什么的。所有的容器都按照相同的方式将内容“装载”进去。

Docker也不关心你要把容器运到何方：我们可以在自己的笔记本中构建容器，上传到Registry，然后下载到一个物理的或者虚拟的服务器来测试，在把容器部署到具体的主机中。像标准集装箱一样，Docker容器方便替换，可以叠加，易于分发，并且尽量通用。

使用Docker，我们可以快速的构建一个应用程序服务器、一个消息总线、一套实用工具、一个持续集成（CI）测试环境或者任意一种应用程序、服务或工具。我们可以在本地构建一个完整的测试环境，也可以为生产或开发快速复制一套复杂的应用程序栈。

## 2、Docker安装与启动

### 2.1 安装环境说明

必须安装在Linux系统上，官方建议安装在Ubuntu，可以安装在Centos7.x以上的版本。

### 2.2 使用yum命令在线安装

yum install docker

### 2.3 查看docker版本

docker -v

### 2.4 启动与停止docker

启动docker：systemctl start docker

停止docker：systemctl stop docker

重启docker：systemctl restart docker

查看docker状态：systemctl status docker

开机启动：systemctl enable docker

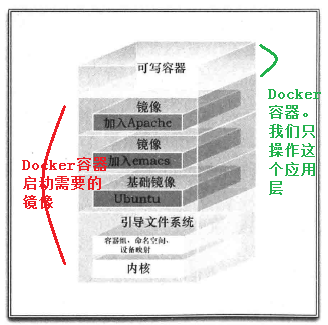
查看docker概要信息：docker info 包括镜像和容器数

查看docker帮助文档：docker –help

## 3、Docker镜像操作

### 3.1什么是Docker镜像

Docker镜像是由文件系统叠加而成（是一种文件的存储形式）。最底端是一个文件引导系统，即bootfs，这很像典型的Linux/Unix的引导文件系统。Docker用户几乎永远不会和引导系统有什么交互。实际上，当一个容器启动后，它将会被移动到内存中，而引导文件系统则会被卸载，以留出更多的内存供磁盘镜像使用。Docker容器启动是需要的一些文件，而这些文件就可以称为Docker镜像。



### 3.2 列出镜像（查本地）

命令：docker images

* REPOSITORY：镜像所在的仓库名称
* TAG：镜像标签
* IMAGE ID：镜像ID，唯一标识
* CREATED：镜像的创建日期（不是获取该镜像的日期）
* SIZE：镜像大小
* 这些镜像都是存储在Docker宿主机的/var/lib/docker目录下

### 3.3搜索镜像（查网络）

命令：docker search 镜像名称

### 3.4 拉取镜像（下载镜像）

#### 3.4.1 从Docker Hub（注册中心）拉取

命令：docker pull centos:7

从官网下载速度慢，所以需要使用镜像加速器（私服）来下载：

#### 3.4.2 ustc的镜像（先配置，后拉取，如果还是很慢，则重启docker或宿主机）

<https://lug.ustc.edu.cn/wiki/mirrors/help/docker>

步骤：

（1）编辑该文件：/etc/docker/daemon.json（如果文件不存在就手动创建）：

vi /etc/docker/daemon.json

\*解释为什么不用vim？

mini安装没有vim命令。

（2）在该文件中输入如下内容：

|  |
| --- |
| {  "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]  } |

（3）注意：一定要重启docker服务

systemctl restart docker.service

（4）下载镜像

docker pull zookeeper

### 3.5 删除镜像

（1）docker rmi $IMAGE\_ID 删除指定镜像

（2）docker rimi `docker images -q` 删除所有镜像（只删除容器未运行的镜像）

注意：`不是引号，是1旁边的按键

## 4、Docker容器操作

### 4.1 查看容器

查看正在运行的容器：

|  |
| --- |
| docker ps |

查看所有的容器（启动过的历史容器）：

|  |
| --- |
| docker ps -a |

查看最后一次运行的容器

|  |
| --- |
| docker ps -l |

查看停止的容器

|  |
| --- |
| docker ps -f status=exited |

退出容器

|  |
| --- |
| exit |

### 4.2 创建与启动容器（即使容器关闭，name也不能重复；通过一个镜像，可以创建多个容器）

创建并运行容器命令： docker run

举例： docker run -i(或-i -t或-it) --name=名称

参数：

-i： 表示运行容器

-t: 表示容器启动后会进入其命令行。加入这两个参数后，容器创建就能登录进去。即分配一个伪终端。

--name 为创建的容器命令

-d 在run后面加上-d参数,则会创建一个守护式容器在后台运行（这样创建容器后不会自动登录容器，如果只加-i -t两个参数，创建后就会自动进去容器）。

-v 表示目录映射关系（前者是宿主机目录，后者是映射到宿主机上的目录），可以使用多个－v做多个目录或文件映射。注意：最好做目录映射，在宿主机上做修改，然后共享到容器上。

-p：表示端口映射，前者是宿主机端口，后者是容器内的映射端口。可以使用多个－p做多个端口映射

#### 4.2.1 交互式容器（创建后马上能进入到容器内，退出后自动关）

|  |
| --- |
| docker run -it --name=mycentos centos:7 /bin/bash |

#### 4.2.2 守护式容器（也叫后台式容器，创建后不进入容器，后台已运行的）

|  |
| --- |
| 创建命令：  docker run -di --name=mycentos1 centos:7  进容器命令：  语义：docker exec -it 容器名 /bin/bash  举例：docker exec -it mycentos1 /bin/bash |

### 4.3 停止与启动容器

|  |
| --- |
| 启动：docker start mycentos  停止：docker stop mycentos |

### 4.4文件拷贝（容器必须是开启的，容器之间不能cp）

|  |
| --- |
| 如果我们需要将文件拷贝到容器内可以使用cp命令  docker cp 需要拷贝的文件或目录 容器名称：容器目录 |
| 也可以将文件从容器内拷贝出来  docker cp 容器名称：容器目录 需要拷贝的文件或目录 |

### 4.5 目录挂载（宿主机的目录和容器的目录之间的映射）

|  |
| --- |
| 创建并挂载  语法：docker run -di --name=容器名 -v 宿主机目录：容器目录 镜像名：镜像tag  举例：docker run -di --name=mycentos2 -v /usr/local/myhtml:/usr/local/mh centos:7 |

若挂载多层的文件夹，这是因为CentOS7中的安全模块selinux把权限禁掉了，我们需要添加参数 --privileged=true 来解决挂载的目录没有权限的问题

在创建容器时需要把权限打开

|  |
| --- |
| docker run -di –name=容器名 -v 宿主机路径：容器路径 --privileged=true centos:7 |

### 4.6 查看容器IP地址

|  |
| --- |
| 查看容器运行的各种数据：  docker inspect mycentos |
| 直接输出IP地址：（引号）  docker inspect --format=’{{.NetworkSettings.IPAddress}}’ mycentos1 |

### 4.7删除容器

|  |
| --- |
| 删除一个容器（只能删除停止的）：  docker rm $CONTAINER\_ID/NAME  删除所有：  docker rm `docker ps -a -q` |

## 5、部署应用

### 5.1 MySQL部署

#### 5.1.1 拉取MySql镜像

docker pull mysql

#### 5.1.2 创建MySql容器

|  |
| --- |
| 语义：（-e 设置环境变量）  docker run -di –name=容器名 -p 宿主端口：容器端口 -e MYSQL\_ROOT\_PASSWORD=123456 mysql |
| 举例：  docker run -di –name=test\_mysql -p 33306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 mysql |

#### 5.1.3 查看容器IP地址并记录下来

### 5.2 tomcat部署

#### 5.2.1 拉取tomcat镜像

docker pull tomcat:7-jre7

#### 5.2.2 创建tomcat容器（以cas单点登录为例）

（1）修改cas的数据库配置

（2）文件挂载到容器中

|  |
| --- |
| docker run -di --name=tomcat -v /opt:/usr/local/tomcat/webapps/ --privileged=true -p 9000:8080 tomcat |

### 5.3 nginx部署

#### 5.3.1 拉取Nginx镜像

docker pull nginx

#### 5.3.2 创建Nginx容器

docker run -di --name=nginx -p 80:80 nginx

#### 5.3.3 测试Nginx

#### 5.3.4 反向代理

（1）先查看反向代理的tomcat的ip地址

docker inspect tomcat

（2）修改docker容器中的/etc/nginx/nginx.conf配置（此处将容器中的文件cp到宿主机，在宿主机修改后再cp回去）

命令：docker cp nginx:/etc/nginx/nginx.conf nginx.conf

在http{}中增加upstream等信息

|  |
| --- |
| upstream 自定义名称 {  server 宿主机地址:端口;  }  server {  listen 80;  server\_name 域名;  location / {  proxy\_pass <http://上面的名称>;  index index.html index.htm;  }  } |

（3）把修改后的文件cp到容器中

docker cp nginx.conf nginx:/etc/nginx

（4）进到容器后去查看是否修改成功

cat nginx.conf

（5）修改配置文件后，需要重启容器

docker restart nginx

### 6、Redis部署

#### 6.1 拉取redis镜像

docker pull redis

#### 6.2 创建redis容器

docker run -di --name=redis -p 6379:6379 --privileged=true redis

### 7、备份与迁移（用于传递给运维或其他人）

（1）从容器保存为镜像

语法：docker commit 容器名 要保存的镜像名

例如：docker commit nginx mynginx

（2）将镜像备份，导出成压缩包文件

语法：docker save -o 压缩包文件 镜像名

例如：docker save -o mynginx.tar mynginx

（3）压缩包恢复成镜像：

语法：docker load -i 压缩包文件

例如：docker load -i mynginx.tar