**Linux容器技术与虚拟机的区别：**

虚拟机运行与系统架构上，虚拟机需要模拟硬件，对cpu、内存等资源损耗较大

容器只需要应用（app）和所依赖的库（Bins/Libs）即可运行，资源占用少

**容器：**

* 1. 一种虚拟化方案
  2. 只能运行底层相同的或相似的内核的操作系统
  3. 依赖于linux内核，只能运行与linux操作系统，主要用到Namespace和Cgroups两个内核模块
  4. Namespace：（命名空间）用于容器的隔离，例如PID Namespace使得容器中的进程无法感知宿主机以及其他容器中的进程。

1. PID（Process ID）进程隔离
2. NET(Network) 管理网络接口
3. IPC（InterProcess Communication）管理跨进程通信的访问
4. MNT（mount） 管理挂载点
5. UTS（Unix Timesharing System）隔离内核和版本标识
   1. Cgroups: （控制组）用于容器的资源控制， 比如限制容器所使用的内存大小或者CPU个数。管理Namespace那些隔离，用来分配资源，来源于google资源限制，优先级设定，资源计量，资源控制达到以下目的：
6. 文件系统隔离：每个容器都有自己的root文件系统
7. 进程隔离：每个容器都运行在自己的进程环境中
8. 网络隔离：容器间的虚拟网络接口和ip地址都是分开的
9. 资源隔离和分组：将CPU和内存之类的资源独立分配给每个Docker容器

**Docke简介：**

* + 1. 将应用程序自动部署到容器开源软件，GO语言编写的，基于Apache2.0开发
    2. 增加了一个应用部署引擎，提供简单轻量的建模方式，
    3. 职责的逻辑分离，
    4. 快速高效的开发生命周期，可使一个程序的开发，测试，交付部署处于同一个环境中
    5. 推荐单个容器只运行一个应用服务或者进程，形成一个封闭式的应用模型，使调试简单

**Docker应用场景：**

1. 使用docker容器开发、测试、部署服务
2. 创建隔离的运行环境
3. 搭建测试环境
4. 构建多用户的平台即服务（PaaS）基础设施
5. 提供软件即服务（SaaS）应用程序
6. 高性能、超大规模的宿主机部署

**Docker组成：**

1. Docker Client 客户端
2. Docker Daemon 守护进程
3. Docker Image 镜像
   1. 容器的基石，
   2. 层叠的只读文件系统，联合加载，文件打包加载
4. Docker Container 容器
   1. 通过镜像启动，文件的启动和执行，容器启动在层叠的只读文件系统上加载一层读写层
   2. 第一次启动容器，初始读写层是空的，文件系统变化时，将下面的只读文件复制到上面读写层，下面的只读文件还存在于下面的只读镜像层（写时复制技术copy on write），下面的只读镜像层永远不会变化
5. Docker Registry 仓库

公有：Docker公司的Docker Hub,可以去注册使用，如github一样

私有：自己架设私有仓库

**各组成的关系：**Docker Client 客户端--🡪访问Docker Daemon 守护进程-🡪-操作Docker Container 容器

Docker Image 镜像-🡪生成Docker Container容器 Docker Image 镜像🡪存在于 Docker Registry 仓库中

**启动容器：**

]# docker run IMAGE [COMMAND] [ARG] #run 是创建并启动容器，IMAGE:镜像COMMAND:命令 ARG:命令参数

]# docker run centos7 echo ‘ hello world’ #一次启动：启动容器并执行输出命令，然后关闭容器

**启动交互式容器：**

]# docker run - i -t IMAGE

]# docker run - i -t --name=test01 IMAGE /bin/bash

#IMAGE 是镜像名 --name 自定义容器名字为test01

#-i 是--interactive=true的意思，默认是false , -i告诉docker守护进程为容器始终打开标准输入

#-t 是—tty=true的意思，默认是false , -t 告诉docker要为创建的容器创建一个伪tty的终端（命令输入）

root@85c6b0b62235:/# #进入交互界面，

root@85c6b0b62235:/# exit #退出容器也将停止

]# docker start -i 容器名 #重新启动停止的容器

]# docker ps #列出所有正在运行的容器

]# docker ps –al #列出所有容器， -a 所有 -l 最新容器

]# docker rm 容器名， #删除已停止的容器，启动的不能删除

**守护式容器：**

]# docker run - i -t 镜像名 /bin/bash #启动容器

root@85c6b0b62235:/# #进入交互界面不用exit退出，用ctrl+P 或 ctrl+q 退出交互界面

]# docker attach 容器名 #重新连接已运行的容器，附加到运行中的容器，

]# docker run -d 镜像名 /bin/bash #容器放入后台运行

]# docker logs [-f|-t|--tail] 容器名 #查看容器运行情况，查看容器日志

#-f 一直耿总变化并返回结果 -t 在返回的结果上加上时间戳 --tail 指定返回最后的信息数量

]# docker top 容器名 #查看容器进程.

]# docker exec [-d|-i|-t] 容器名 [command] [arg] #在运行中的容器内启动新进程

root@85c6b0b62235:/# #用ctrl+P 或 ctrl+q 退出交互界面，新进程继续运行 ctrl+d 直接退出

**停止与启动守护式容器**：

]# docker stop 容器名 #发送信号给容器，等待停止

]# docker kill 容器名 #直接停止容器

]# docker start 容器名 #启动已停止的容器

**Docker端口映射：**

**大写P 将为容器暴露的所有端口进行映射 命令：** ]# docker **run –P –i –t centos /bin/bash**

**小写p 能指定映射容器那些端口，有四种映射方式**

]# docker run –p 80 –i –t centos /bin/bash #只指定了容器端口，宿主机端口随机映射

]# docker run –P 8080:80 –i –t centos /bin/bash #宿主机8080与容器80端口映射

]# docker run –P 0.0.0.0:80 –i –t centos /bin/bash #ip地址与容器80端口映射

]# docker run –P 0.0.0.0:8080:80 –i –t centos /bin/bash # #ip:宿主机端口:容器端口

**Docker安装nginx案例：**

[root@server pub]# docker pull centos #直接拉取centos镜像是连接国外docker官网，拉取慢要换国内镜像站

Using default tag: latest

[root@server docker]# vim daemon.json #没有的话就创建

{

"registry-mirrors":["https://mj9kvemk.mirror.aliyuncs.com"] #用的是阿里云镜像站

}

[root@server docker]# docker pull nginx #下载nginx镜像，也可以下载centos镜像

Using default tag: latest

latest: Pulling from library/centos

8ba884070f61: Pull complete

Digest: sha256:b5e66c4651870a1ad435cd75922fe2cb943c9e973a9673822d1414824a1d0475

Status: Downloaded newer image for centos:latest

[root@server docker]# docker images #查看镜像文件

REPOSITORY TAG IMAGE ID CREATED SIZE

nginx latest 53f3fd8007f7 3 days ago 109 MB

centos latest 9f38484d220f 8 weeks ago 202 MB

[root@server docker]# docker run -p 8080:80 -it --name=nginx01 -d nginx /bin/bash #启动容器

root@c9808037f17b:/# which nginx[root@server docker]# #ctrl + q退出交互界面容器未停

[root@server docker]#

[root@server docker]# docker port nginx01 #查看在运行的容器的 端口映射 情况

80/tcp -> 0.0.0.0:8080

[root@server docker]# docker top nginx01 #查看在运行的容器的 进程 情况

[root@server docker]# docker inspect nginx01 #查看容器详细信息、底层信息、容器IP地址且宿主机可访问

[root@server docker]# curl <http://172.17.0.2> #宿主机访问容器ip也是可以访问的，外面访问宿主机地址

<!DOCTYPE html>

<html>

[root@server docker]# docker stop nginx01 #停止容器

nginx01

[root@server docker]#

[root@server docker]# docker start nginx01 #再次启动容器，

nginx01

[root@server docker]# curl <http://172.17.0.2> #nginx服务并未启动

curl: (7) Failed connect to 172.17.0.2:80; 拒绝连接

[root@server docker]# docker stop nginx01

nginx01

[root@server docker]# curl <http://172.17.0.2> #无法连接nginx

[root@server docker]# docker exec nginx01 nginx #exec连接容器，容器必须是运行状态

Error response from daemon: Container c9808037f17b627f32f11258d8d1165b9abe5e82febf1e77bd938ea41568cd2b is not running

[root@server docker]# docker start nginx01

nginx01

[root@server docker]# curl http://172.17.0.2

curl: (7) Failed connect to 172.17.0.2:80; 拒绝连接

[root@server docker]# docker exec nginx01 nginx #exec连接nginx01容器的同时 执行nginx启动

[root@server docker]# curl <http://172.17.0.2> #连接成功

**Dokcer镜像**

容器的基石，层叠的只读文件系统，联合加载，

[root@server ~]# ls /var/lib/docker/

containers image network overlay plugins swarm tmp trust volumes

]# docker info #查看docker使用的存储驱动和存储位置

]# docker images [OPTSIONS选项] [REPOSITORY仓库] #查看镜像格式

~]# docker images –a #-a显示所有默认不显示中间层的镜像，无名和标签的镜像为中间层镜像

~]# docker images -f 过滤条件 #带显示时的过滤条件

~]# docker images -q #-q只显示镜像的唯一ID

~]# docker images --no-trunc #显示的时候，id值不截断，可看到完整的

[root@server ~]# docker images nginx #查看nginx仓库的镜像

[root@server ~]# docker images –a #-a显示所有默认不显示中间层的镜像

REPOSITORY**（仓库名）** TAG**（标签名）**  IMAGE ID**（唯一ID）** CREATED（镜像建立时间） SIZE（镜像打小）

Nginx latest 53f3fd8007f7 3 days ago 109 MB

centos latest 9f38484d220f 8 weeks ago 202 MB

# REPOSITORY仓库（一系列镜像的集合） 与 REGISTRy仓库（docker镜像的存储服务）有区别

# REGISTRy仓库含有很多REPOSITORY仓库，REPOSITORY仓库中包含一个一个独立镜像

#TAG标签：仓库名+标签名=镜像名，如果未加标签名，默认使用latest标签，如nginx = ngins:latesr

#一个镜像可打多个标签

]# docker images nginx #查看镜像详细信息，底层信息

]# docker rmi [OPTIONS选项] IMAGE [IMAGE…镜像名] #删除镜像格式

#选项 -f 强制删除 - -no-prune 保留被打标签的父镜像

]# docker rmi busybox #删除镜像

]#docker images #查看镜像

]#docker save busbox:lotest >busybox.tar #导出镜像

]#docker load < busybox.tar #导入镜像

]# for i in \* ; do docker load <$i ; done #一次性导入多个镜像

**获取和推送镜像**

REGISTRY远程仓库 与REPOSITORY本地仓库，之间操作

#可到docker hub网站查找镜像，有许多官方的仓库

]# docker search [OPTIONS选项] TERM #查找镜像最多一次返回25个结果

]# docker search --automated nginx #--automated自动化选项，只会显示自动化创建的

]# docker search -s 3 nginx # -s限定查找镜像评定的星级，此处查找星级为4以上的nginx镜像

]#docker pull [OPTIONS选项] busybox #下载busybox镜像-a 配置国内镜像站会更快

# vim daemon.json 添加 "registry-mirrors"参数内容，再重启docker

]#docker pull busybox #下载busybox镜像

#docker push busybox #上传镜像,可结合docker hub使用，类似于github

**自定义构建镜像：**

保存对容器的修改并再次使用， 自定义镜像的能力， 以软件的形式打包并分发服务及其运行环境

有两种方式：docker commit命令方式 与 docker build命令方式

**docker commit [OPTIONS选项] CONTAINER [REPOSITORY仓库]命令方式：**

选项：-a 记录作者信息， -m 记录镜像信息 -p 表示不暂停正在运行的容器，

通过容器构建，进行打包容器成新的镜像，提交一个具有新配置的容器成为镜像，后面跟容器的name 或者容器Id ，最后是生成新镜像的名字

[root@server ~]# docker run centos -it --name ftpserver centos /bin/bash

[root@b25e4e56e863 /]# yum -y install vsftpd

[root@b25e4e56e863 /]# #

[root@server ~]# docker ps

[root@server ~]# docker commit -a 'zdd01' -m '1703' ftpserver ftpservertest1/test2

[root@server ~]# docker commit -a 'zdd01' -m '1703' ftpserver ftpservertest1:test3

[root@server ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

ftpservertest1 test3 c726e4c66495 3 seconds ago 292 MB

ftpservertest1/test2 latest 77a2e5affdf0 32 seconds ago

**docker build命令方式：**

通过dockerfile文件创建镜像，dockerfile文件所需要的全部指令。基于在这些的指令，我们可以使用Docker build命令来创建镜像。通过减少镜像和容器的创建过程来简化部署。

]# docker build -t centos\_sshd\_1 . # -t 选项来docker build新的镜像以便于标记构建的镜像，. 表示当前目录，也可以指定dockerfile 文件所在目录。