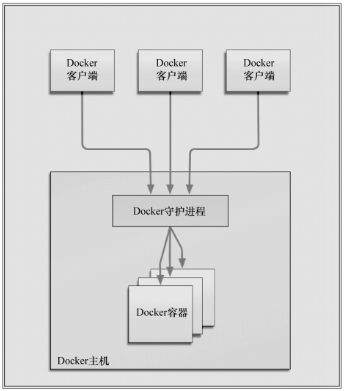
# Docker 架构

Docker是客户－服务器(C/S)架构的程序。Docker客户端只需向Docker服务器或守护进程发出请求，服务器或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 APICD。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。

**Docker的架构图：**



# Docker 三大要素

## 镜像(image)：

Docker镜像（ Image）就是一个只读的模板。镜像可以用来创建 Docker容器，一个镜像可以创建很多容器。

镜像是一种轻量级、可执行的独立软件包，用来打包件运行环境和基于运行环境开发的软件，它包含运行某个软件所需的所有内容，包括代码、运行时、库、环境变量和配置文件。

## 容器(Container)

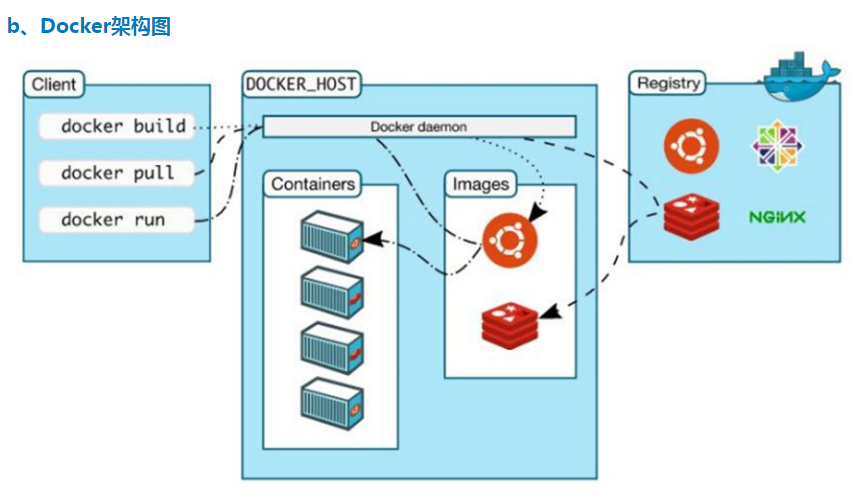
* Docker利用容器（ Container）独立运行的一个或一组应用。容器是用镜像创建的运行实例。（相对于Java中的类和对象的概念）
* 它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。
* 可以把容器看做是一个简易版的 Linux环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序。
* 容器的定义和镜像几乎一模一样，也是一堆层的统一视角，唯一区别在于容器的最上面那一层是可读可写的

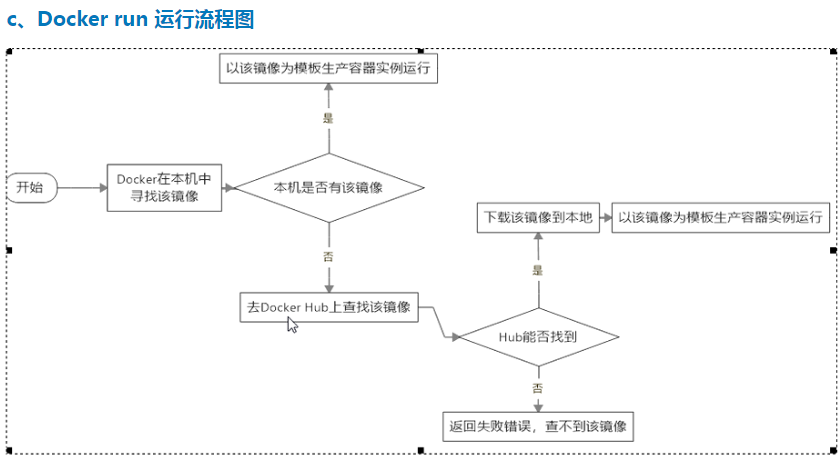
## 仓库(Repository)

* 仓库（ Repository）是集中存放镜像文件的场所。
* 仓库（ Repository）和仓库注册服务器（ Registry）是有区别的。仓库注册服务器上往往存放着多个仓库，每个仓库中又包含了多个镜像，每个镜像有不同的标签（tag）。
* 仓库分为公开仓库（ Public）和私有仓库（ Private）两种形式。
* 最大的公开仓库是 Docker Hub（htps:/ /hub docker. com/），存放了数量庞大的镜像供用户下载。国内的公开仓库包括阿里云、网易云等

三要素小结：

Docker本身是一个举器运行载体或称之为管理引擎。我们把应用程序和配置依赖打包好形成一个可交付的运行环境，这个打包好的运行环境就似乎 Image镜像文件。只有通过这个镜像文件才能生成 Docker容器。 Image文件可以看作是容器的模板。 Docker根据iage文件生成容器的实例。同一个mage文件，可以生成多个同时运行的容器实例。



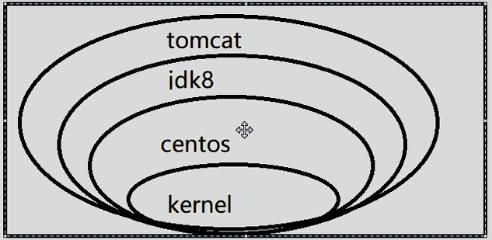


# 联合文件系统(UnionFS)

联合文件系统(UnionFS)是一种分层、轻量级、高性能的文件系统。它支持对文件系统的修改作为一次提交来一层层的叠加，同时可以将不同目录挂载到同一虚拟文件系统下。Union文件系统是Docker镜像的基础。镜像可以通过分层来进行继承，基于基础镜像（没有父镜像），可以制作各种具体的应用镜像。

典型的Linux文件系统由bootfs(boot file system)和rootfs(root file system)两部分组成:

* bootfs(boot file system)主要包含 bootloader和kernel，bootloader主要是引导加载kernel，Linux刚启动时会加载bootfs文件系统，在Docker镜像的最底层是 bootfs。这一层与我们典型的 Linux/Unⅸ系统是一样的，包含boot加载器和内核。当boot加载完成之后整个内核就都在内存中了，此时内存的使用权已由 bootfs转交给内核，此时系统也会卸载 bootfs。当kernel被加载到内存中后 bootfs就被umount了。
* rootfs (root file system) 包含的就是典型 Linux 系统中的/dev，/proc，/bin，/etc等标准目录和文件，rootfs就是不同操作系统的发行版。比如 Ubuntu, Centos等等。

以tomcat镜像为例：

  Docker容器的文件系统类似于Linux虚拟化栈。Docker镜像是由文件系统叠加而成。最底端是一个引导文件系统，即bootfs, 很像典型的Linux/Unix的引导文件系统。当一个容器启动时，首先加载bootfs用于引导重要文件，当容器启动后，bootfs引导文件系统会被卸载(unmount)。

  传统的Linux引导过程中，root文件系统会最先以只读的方式加载，当引导结束并完成了完整性检查之后，rootfs将会从read-only改为read-write。

Docker的镜像加载时，在bootfs自检完毕之后并不会把rootfs的read-only改为read-write。而是利用联合加载union mount（UnionFS的一种挂载机制）将一个或多个read-only的rootfs加载到之前的read-only的rootfs层之上。在加载了这么多层的rootfs之后，仍然让它看起来只像是一个文件系统，在Docker的体系里把union mount的这些read-only的rootfs叫做Docker的镜像。

  一个镜像可以放到另一个镜像的顶部。位于下面的镜像称为父镜像(parent image), 依次类推，直到镜像栈的最底部，最底部的镜像称为基础镜像( base image)。最后，当从一个镜像启动容器时，此时的每一层rootfs都是read-only的，镜像可以通过分层来进行继承，基于基础镜像可以制作各种具体的应用镜像。在镜像启动容器的最后阶段，Docker会在该镜像的最顶层加载一个读写read-write的rootfs文件系统，我们想在Docker中运行的程序就是在这个读写层中执行的。

  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。该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，这种机制被称为写时复制(copy on write), 这也是使Docker如此强大的技术之一。 每个只读镜像层都是只读的，以后永远不会变化。当创建一个新容器时，Docker会构建出一个镜像栈，并在栈的最顶端添加一个读写层。这个读写层再加上其下面的镜像层以及一些配置数据，就构成了一个容器。容器的这种特点加上镜像分层框架( image-layering framework), 使我们可以快速构建镜像并运行包我们自己的应用程序和服务的容器。

docker本地镜像保存在 /var/lib/docker ，容器保存在 /var/lib/docker/containers 。

  总结：Docker镜像（image）就是一个只读的模板。 镜像可以用来创建Docker容器，一个镜像可以创建很多容器。特性：一次同时加载多个文件系统，但从外面看起来，只能看到一个文件系统，联合加载会把各层文件系统叠加起来，这样最终的文件系统会包含所有底层的文件和目录

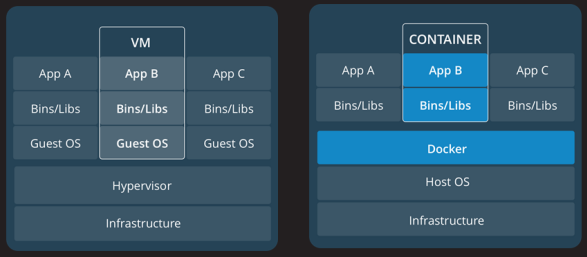
镜像分层优势：共享资源：通过引入分层技术，可以增加镜像复用度，减少磁盘占用；同时可以实现差分上传下载，加快镜像pull、push性能。

# Docker 原理

Docker是一个 Client-Server结构的系统， Docker守护进程运行在主机上，然后通过Socket连接从客户端访间，守护进程从客户端接受命令并管理运行在主机上的容器。容器，是一个运行时环境，就是我们前面说到的集装箱。

Docker优点

* 相比于传统的虚拟化技术，容器更加简洁高效
* 传统虚拟机需要给毎个ⅤM安装操作系统
* 容器使用的共享公共库和程序





# Centos Docker安装

Docker支持以下的 Centos版本

* CentoS 7（64-bit）
* CentoS 6.5（64bi）或更高的版本

前提条件

* 目前， Centos仅发行版本中的内核支持 Docker
* Docker运行在 Centos7上，要求系统为64位、系统内核版本为3.10以上
* Docker运行在 Centos6.5或更高的版本的 CentoS上，要求系统为64位、系统内核版本为2.6.32-431國者更高版本查看自己的内核uname -r命令用于打印当前系统相关信息（内核版本号、硬件架构、主机名称和操作系统类型等）。

## Centos 6.5安装

Docker使用EPEL发布，RHEL系的OS首先要确保已经持有EPEL仓库。

1. yum install -y epel-release

2.yum install -y docker-io

3./etc/sysconfig/docker 安装后的配置文件：

4.service docker start 启动 Docker后台服务

5.docker version 验证

## Centos 7 安装

1、卸载旧版本，（有就操作，没有就不操作）

yum remove docker docker-client docker-client-latest docker-common docker-latest

docker-latest-logrotate docker-logrotate docker-engine

2、yum install -y epel-release

3、yum install -y yum-utils device-mapper-persistent-data lvm2

#yum-utils提供了yum-config-manager实用程序，并device-mapper-persistent-data和lvm2由需要 devicemapper存储驱动程序。

4、yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

#设置稳定存储库。

5、yum install -y docker-ce docker-ce-cli containerd.io

systemctl start docker

docker info

ps -ef |grep docker

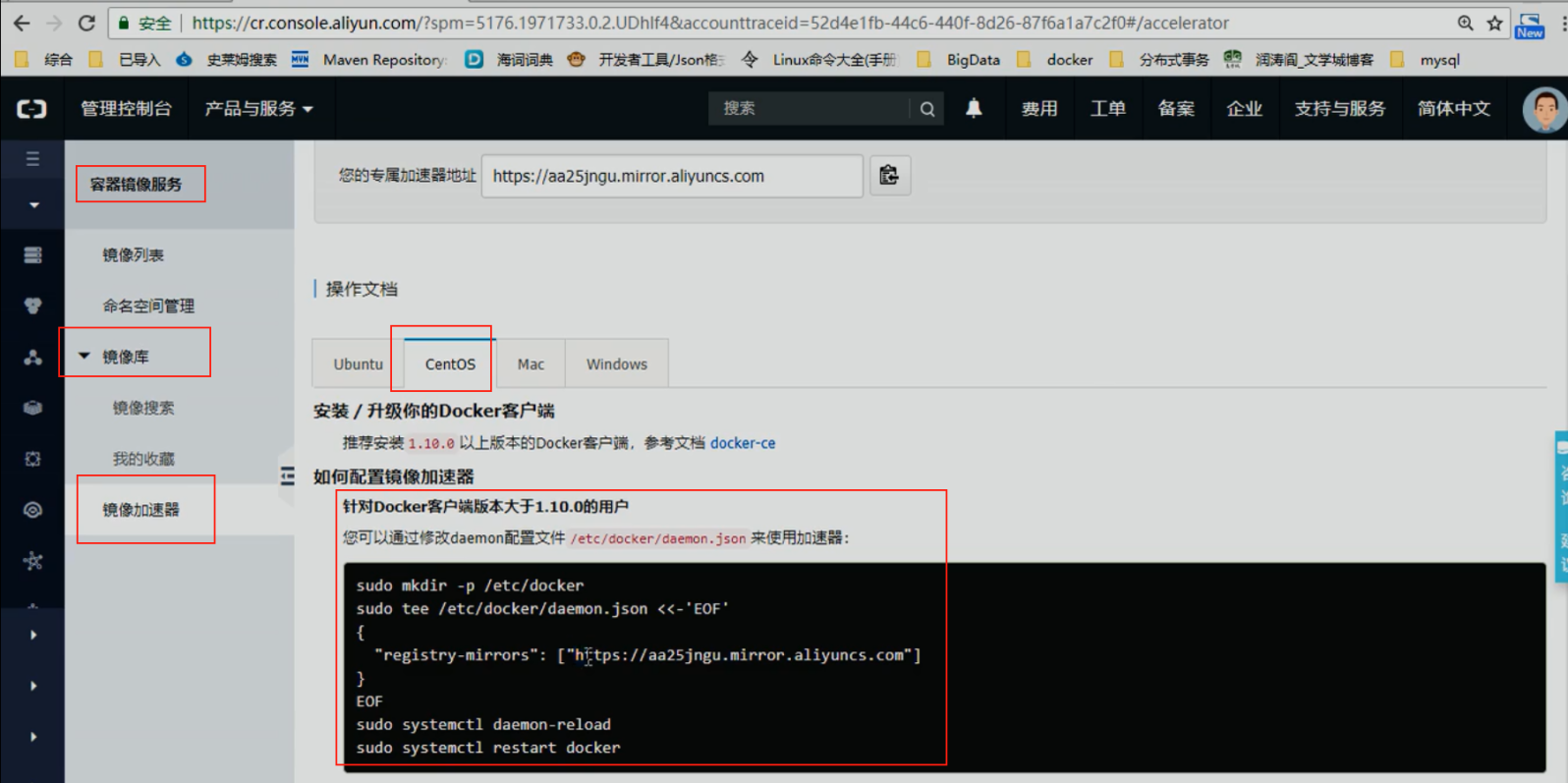
# Docker使用：

## 阿里云镜像加速：

地址：https://dev.aliyun.com/search.html

1. 注册一个属于自己的阿里云账户（可复用淘宝账号）
2. 获得加速器地址连接

下图为：https://aa25jngu.mirror.aliyuncs.com



1. 配置本机 Docker运行镜像加速器，

Centos6.5配置：

1、vim /etc/sysconfig/docke 配置文件

other\_args="- -registry-mirrors=https://aa25jngu.mirror.aliyuncs.com” #每个账户地址不一样

2、service docker restart #重启 Docker后台服务

3、ps - ef l grep docker #检测配置

root 7595 1 1 04:58 pts/ 000:00:00 /usr/bin/docker-d--registry-mirrorshttps://aa25jngu.mirroraliyuncs root

Centos7.0以上配置：

1、vim /etc/docker/daemon jsonl #此文件没有，新建并添加如下Json串内容

{

"registry-mirrors"：["https://aa25jngu.mirror.aliyuncs.com"]

}

2、 systemctl daemon-reload

3、 systemctl restart docker

## 网易云镜像加速：

就Json串的地方不同，其他无区别

## Docker命令集

### 帮助命令：

docker version 查看版本

docker info 查看docker详细信息,正在运行的容器数量等等数据

docker --help 使用帮助，可借助翻译工具

### 镜像命令

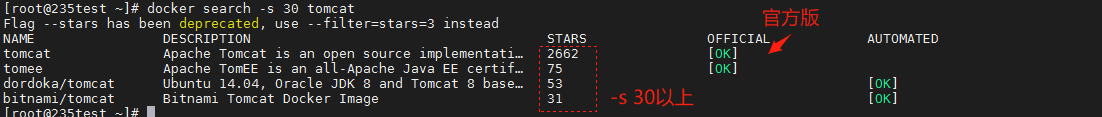
]#docker images #查看本地镜像

]#docker images -qa #查看本地所有镜像只显示ID

* REPOSITORY：表示镜像的仓库源 TAG：镜像的标签
* 同一仓库源可有多个TAG，代表这个仓库源的不同个版本，我们使用 REPOSITORY:TAG来定义不同的镜像
* 如果你不指定一个镜像的版本标签，例如你只使用centos, docker将默认使用 centos:latest镜像
* -a:列出所有本地镜像(含中间映像层) -q:只显示镜像ID
* - -digests:显示镜像的摘要信息 - -no-trunc:显示完整镜像信息

]# docker search [opshion] tomcat #从docker hub（https://hub.docker.com）上查询镜像

* 就算配置了阿里云镜像也是从docker hub（https://hub.docker.com）上查询镜像
* -s:列出收藏数不小于指定值得镜像， 例：docker search -s 30 tomcat
* - -no-trunc:显示完整的镜像描述 - -automated:只列出automated build类型镜像



]# docker pull tomcat #下载tomcat镜像

* 未设置tag等价于docker pull tomcat:latest 拉取指定版本，tomcat:3.2、tomcat:5.7

]# docker rmi hello-world #删除hello-world镜像

]# docker rmi hello-world nginx #删除hello-world和nginx 镜像，删除多个，空格隔开

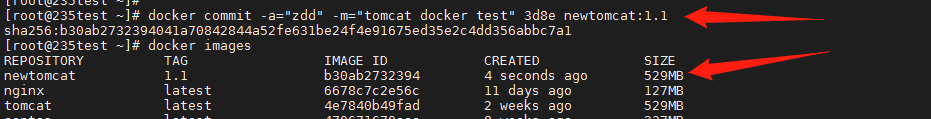
]# docker rmi -f $(docker images -qa) #强制删除全部镜像-----------慎用、慎用、慎用

* -f:强制删除

]# docker commit #提交镜像

docker commit -m=”提交的描述信息” -a=”作者” 正在运行容器ID 要创建的目标镜像名:标签名

]# docker commit -a="zdd" -m="tomcat docker test" 3d8e newtomcat:1.1



]#docker push busybox #上传镜像

]#docker save busbox:lotest >busybox.tar #导出镜像到一个tar包中，对应【load】

]#docker load <busybox.tar #从一个tar包中导入镜像，对应【save】

]# docker rmi busybox #删除镜像

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

]# docker history centos #查看centos镜像历史（制作过程）

]# docker tag centos:latest cen:v1 #修改镜像名为cen:v1

]# docker inspect centos #查看镜像底层信息.

]# docker run -it –d centos /bin/bash - -name XX

#启动centos镜像，并创建容器 -it :进行交互式操作 –d：放后台 –name重新起名

build #定制镜像

commit #提交镜像

### 容器命令

]#docker run 【OPTIONS】 IMAGE 【COMMAND】 【ARG.】

]# docker run -it 470671 #交互式启动以ID号为470671的镜像启动一个容器，并提供一个伪终端

]# docker run -it centos /bin/bash #以/bin/bash命令格式交互，默认为这个

]# docker run -it - -name centostest centos #以centos的镜像启动一个容器，并命名为：centostest

]# docker run -d - -name centostest centos #以后台模式启动一个容器

* --name="容器新名字∵为容器指定一个名称
* -d：后台运行容器，并返回容器ID，也即启动守护式容器；
* -i：以交互模式运行容器，通常与-t同时使用；
* -t：为容器重新分配一个伪输入终端，通常与-同时使用P：随机端口映射
* -P：随机端口映射
* -p：指定端口映射，有以下四种格式 例如：]# docker run -d -p 5566:8080 tomcat

ip:hostPort:containerPort

ip::containerPort

hostPort:containerPort

containerPort

以后台模式启动一个容器docker run -d centos问题：然后 docker ps -a进行查看，会发现容器已经停止退出

很重要的要说明的一点：Docker容器后台运行，就必须有一个前台进程容器运行的命令如果不是那些一直挂起的命令（比如运行top,ta），就是会自动退出的这个是 docker的机制问题，

比如你的web容器，我们以 ngInx为例，正常情况下，我们配置启动服务只需要启动响应的 service即可。例如service nginx start但是，这样做，ngInx为后台进程模式运行，就导致 docker前台没有运行的应用，这样的容器后台启动后，会立即自杀因为他觉得他没事可做了所以，最佳的解决方案是，将你要运行的程序以前台进程的形式运行

]# docker ps #查看正在运行的容器

]# docker ps -aq

* -a：列出当前所有正在运行的容器+历史上运行过的
* -q：静默模式，只显示容器编号。
* -l：显示最近创建的容器。
* -n：显示最近n个创建的容器。例：docker ps -n 3
* --no- trunc：不截断输出。

退出容器 组合键：ctrl + q + p 、exit

* ctrl + p + q：容器不停止退出 用docker attach a9重新进入docker容器
* exit：容器停止退出

]# docker stop 1310f #停止容器

]# docker kill 1310f #强制停止容器

]# docker start 1310f #启动容器，可用容器名或者容器ID

]# docker restart 1310f #重启容器，可用容器名或者容器ID

]# docker rm 1310f #删除容器，删除前先停止运行容器

]# docker rm 1310f 2345d #删除多个容器，删除前先停止运行容器

]# docker rm -f $(docker ps -aq) #强制删除所有容器，包括在运行的

]# docker ps -aq |xargs docker rm -f #同上，强制删除所有容器，包括在运行的

* -f :强制删除,正在运行容器的也将被删除

]# docker logs -t -f --tail a9 #打印容器运行日志

* -t:是加入时间戳

"while true; do echo hello zzyy;sleep 2;done"为shell编程命令：每隔2s循环输出zzyy

这样的容器不会自杀，docker ps -a 能看到运行

* -f:跟随最新的日志打印
* - -tail数字:显示最后多少条
* 实验案例

]# docker run -d centos /bin/sh -c "while true; do echo hello zzyy;sleep 2;done"

]# docker ps -a

]# docker logs -t a9

]# docker logs -t -f a9

]# docker logs -t -f --tail 5 a9 #从第5行开始，持续输出日志

]# docker top a9 #查看容器内运行的进程信息

]# docker inspect a9 #查看容器内部的细节，以JSON串形式体现

]# docker inspect -f 64852f #查看容器详细信息,底层信息

]# docker inspect -f '{{.NetworkSettings.MacAddress}}' a9 #查看mac地址

]# docker inspect -f '{{.NetworkSettings.IPAddress}}' a9 #查看ip地址

]# docker attach a9 #进入容器

]# docker exec -it a9 /bin/bash #以bashshell进入容器，

* -i:interactive交互方式 -t:tty命令行意思
* attach：直接进入容器启动命令的终端，不会启动新的进程，exit退出容器将关闭
* exec：在容器中打开新的终端，并且可以启动新的进程，exit退出容器不关闭
* docker exec -it a9 ls -l :不进入容器，即在容器中执行ls -l 命令

]# docker cp a3:/tmp/aaaa.txt /root #将容器内的文件拷贝到宿主机

docker cp 容器ID:容器内路径 目的主机路径

create #创建新的容器，同run 但不启动

diff #查看容器变化

events #从docker服务获取容器实时事件

login #注册或者登陆一个docker源服务器

logout #从当前的Docker registry 退出

export #导出容器的内容流作为一个tar归档文件【对应import】

import #从tar包中的内容创建一个新的文件系统映像【对应export】

port #查看映射端口对

pause #暂停容器

unpause #取消暂停容器

wait #截取容器停止时的退出状态值

**自定义镜像：**

**docker commit 命令方式：**进行打包成新的镜像，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

**dockerfile + docker build命令方式：**dockerfile包含创建镜像所需要的全部指令。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。

**创建私有镜像仓库：**

]# systemctl restart docker #重启docker

]# docker run -d -p 5000:5000 registry #docker重启之后要重新运行

]# docker tag busybox:latest 192.168.1.100:5000/busybox:latest #制作镜像

]# docker push 192.168.1.100:5000/busybox:latest #上传镜像

]# docker run -it 192.168.1.100:5000/busybox:latest /bin/bash #直接起订进行