Docker的常用命令：

1. 查看docker的信息(version、info)

$docker version

$docker info

1. 对image(镜像)的操作(search、pull、images、rmi、history)

$docker images search image\_name #检索镜像

$docker images pull image\_name #下载一个镜像

$docker images -a,--all; --no-trunc; -q,--quiet image\_name #-a显示所有镜像；--no-trunc显示部分的镜像信息；-q只显示镜像的ID

$docker images rmi -f,--force; --no-prune image\_name #rmi删除镜像，-f强制删除镜像，--no-prune不删除没有标签的镜像

$docker images history --no-trunc; -q,--no-quiet image\_name #histtory显示镜像的历史信息，--no-trunc不跟踪输出，-q只显示image\_ID

1. 启动container(容器)（run）

$docker run -i -t image\_name /bin/bash #以交互的方式启动一个镜像，并进入该镜像环境中

$docker run image\_name echo “Hello World” #在容器运行echo命令

$docker run image\_name yum install net-tool vim -y #在容器中安装程序(注：若不加-y，则会进入yum的交互环境，但是镜像无法响应该参数)

docker容器可以理解为在沙盒中运行的进程。这个沙盒包含了该进程运行所必须的资源，包括文件系统、系统类库、shell 环境等等。但这个沙盒默认是不会运行任何程序的。你需要在沙盒中运行一个进程来启动某一个容器。这个进程是该容器的唯一进程，所以当该进程结束的时候，容器也会完全的停止。

1. 查看container容器（ps）

$docker ps #列出当前正在运行的container容器

$docker ps -a #列出所有的container容器

$docker ps -l #列出最近一次启动的容器

1. 保存对container(容器)的修改（commit）

$docker commit ID new\_image\_name [-a,--author=; -m,--messages=] #保存对容器的修改，-a作者，-m修改信息等

注：image相当于类，container相当于实例，不过可以动态给实例安装新软件，然后把这个container用commit命令固化成一个image。

1. 对container(容器)的操作（rm、stop、start、kill、logs、diff、top、cp、restart、attach）

$docker rm `docker ps -a -q` #删除所有的容器，”docker ps -a -q”列出所有的容器ID

$docker rm [-f,--force; -l,--link; -v,--volumes] CONTAINER\_NAME/ID #删除单个的容器，-f强制删除；-l删除指定的连接而不是容器；-v删除与容器相关联的磁盘/卷

$docker stop/kill/start CONTAINER\_NAME/ID #停止/杀死/启动一个容器

$docker logs [-f,--follow; -t,--timestamps] CONTAINER\_NAME/ID #显示容器产生的日志，-f跟踪显示日志；-t显示时间戳

$docker diff CONTAINER\_NAME/ID #列出容器中改变的文件或者目录，显示增加的、删除的、被改变的

$docker top CONTAINER\_NAME/ID #列出容器中产生的进程信息

$docker cp CONTAINER\_NAME/ID:/container\_path to\_path #从容器里面复制文件到目录下

$docker restart [-t,--time=10] CONTAINER\_NAME/ID #重启一个正在运行的容器/ID

$docker attach ID #附加到一个运行的容器上，允许你查看一个运行的容器，可在同一时间attach同一容器。也可从一个容器中脱离出来(CTRL-C)

1. 保存和加载镜像（save、load），把一台机器上的镜像迁移到另一台机器的时候，需要保存和加载镜像。

$docker save image\_name -o file\_path #保存镜像到一个tar包文件

$docker load -i file\_path #加载一个tar包格式的镜像

例：

PC1：$docker save image\_name > /home/save.tar #保存镜像到save.tar文件，并scp到PC2

PC2：$docker load < /home/save.tar #加载该镜像文件

八、登录register server（login）

$docker [-e,--email=; -p,--password=; -u,--username=] login #登录到docke服务器上，可交互式输入邮箱密码账户信息

九、发布image（push）

$docker push new\_image\_name #发布一个镜像

十、根据Dockfile构建出一个container容器

$docker build [--no-cache; -q,--quiet; --rm] -t IMAGE\_NAME Dockfile\_path #根据Dockfile文件构建一个镜像,--no-cache不使用缓存，-q不产生输出信息；--rm构建成功后删除中间容器

十一、给镜像设置标签（tag）

$docker tag IMAGE\_ID AUTH/IMAGE\_NAME:TAR #IMAGE\_ID添加镜像名称、标签和作者信息

例：docker tag 860c279d2fec maohua/centos:dev

十二、在网站(hub.docker.com)下载一个镜像到本地：

$docker search httpd #找到httpd的镜像名称

$docker pull IMAGE\_NAME #将IMAGE\_NAME镜像下载到本地

十三、网络端口映射

$docker run -d -P training/webapp python app.py

注释： -P ：是容器内部端口随机映射到主机的高端口。

-p：是容器内部端口绑定到指定的主机端口或者指定绑定的IP地址---🡪$docker run -d -p 127.0.0.1:5000:5000 training/webapp python app.py

--name：可以指定创建容器的名称

-v /root/www/:/www：将主机中当前目录下的www挂载到容器的/www

-d：后台运行容器

$docker inspect COTAINER\_ID/CONTAINER\_NAME | grep '"IPAddress"' #查看容器(container)的IP地址

十四、查看docker的进程信息：

$docker stats CONTAINER\_ID #查看CONTAINER\_ID的进程信息