1. Docker常用命令
   1. 仓库管理：

可以在/etc/docker/docker.json或者/etc/docker/docker.xml中设置要链接的远程仓库，有的需要在/etc/docker文件中设置要链接的远程仓库

也可以在启动docker 服务的时候指定要链接的远程仓库地址

* 1. 镜像开发
     1. Dockerfile:

**FROM** 基础镜像

**MAINTAINER**镜像制作者的信息

**RUN** 命令操作

RUN命令将在当前image中执行任意合法命令并提交执行结果。命令执行提交后，就会自动执行Dockerfile中的下一个指令。

层级 RUN 指令和生成提交是符合Docker核心理念的做法。它允许像版本控制那样，在任意一个点，对image 镜像进行定制化构建。

RUN 指令缓存不会在下个命令执行时自动失效。比如 RUN apt-get dist-upgrade -y 的缓存就可能被用于下一个指令. --no-cache 标志可以被用于强制取消缓存使用。

**ENV**

ENV指令可以用于为docker容器设置环境变量  
ENV设置的环境变量，可以使用 docker inspect命令来查看。同时还可以使用docker run --env <key>=<value>来修改环境变量

**USER** 用户名

USER 用来切换运行属主身份的。Docker 默认是使用 root，但若不需要，建议切换使用者身分，毕竟 root 权限太大了，使用上有安全的风险。

**WORKDIR** 工作目录

WORKDIR 用来切换工作目录的。Docker 默认的工作目录是/，只有 RUN 能执行 cd 命令切换目录，而且还只作用在当下的 RUN，也就是说每一个 RUN 都是独立进行的。如果想让其他指令在指定的目录下执行，就得靠 WORKDIR。WORKDIR 动作的目录改变是持久的，不用每个指令前都使用一次 WORKDIR

**ADD** 添加压缩包到镜像中，并且将压缩包自动解压

ADD 将文件从路径 <src> 复制添加到容器内部路径 <dest>。

<src> 必须是想对于源文件夹的一个文件或目录，也可以是一个远程的url。<dest> 是目标容器中的绝对路径。

所有的新文件和文件夹都会创建UID 和 GID。事实上如果 <src> 是一个远程文件URL，那么目标文件的权限将会是600。

**COPY** 复制文件到镜像中，但是不解压

**VOLUME**创建一个可以从本地主机或其他容器挂载的挂载点，一般用来存放数据库和需要保持的数据等

**EXPOSE** 指令指定在docker允许时指定的端口进行转发。

**CMD** 定义容器启动命令执行文件

**ENTRYPOINT** 指定 Docker image 运行成 instance (也就是 Docker container) 时，要执行的命令或者文件。

例如：

* + 1. Docker镜像压缩，将多层镜像压缩称为一层,缩小占用内存：
       1. 使用Dockerfile删除无用文件，并生成镜像respository:tag

注意：要使用CMD添加容器生成时的启动命令。

* + - 1. 使用docker create创建容器container

docker create –name container respository:tag

* + - 1. 使用export将容器导出为docker\_temp.tar文件

docker export container > docker\_temp.tar

* + - 1. 使用import将docker\_temp.tar导出为新镜像respository\_new:tag\_new

docker import docker\_temp.tar respository\_new:tag\_new或者cat docker\_temp.tar | docker import -respository\_new:tag\_new

* + - 1. 验证镜像并删除临时文件：

docker rm –f container

rm –f docker\_temp.tar

* 1. 镜像仓库：
     1. 登录和登出镜像仓库：docker login和docker logout

docker login:登陆到一个Docker镜像仓库，如果未指定镜像仓库地址，默认为官方仓库Dokcer hub

docker logout: 登出一个Docker镜像仓库，如果未指定镜像仓库地址，默认为官方仓库Dokcer hub

|  |
| --- |
| 语法：  docker login [options] [server]  docker logout [options] [server] |
| options说明：  -u登录用户名  -p登录的密码 |
| 示例：  登录docker hub:  dokcer login –u username –p passwd  登出docker hub:  docker logout |

* + 1. 从docker hub检索镜像：docker search

docker search image\_name

|  |
| --- |
| 语法：  Docker search [options] term |
| Options说明：  --automated:只列出automated build类型的镜像；  --no-trunc:显示完整的镜像描述  -s:列出收藏数不小于指定值的镜像 |
| 示例：  列出收藏数不小于20的含有java字段的镜像：  docker search –s 20 java |

* + 1. 从仓库拉取镜像到本地宿主机：docker pull
    2. 将本地镜像推送到镜像仓库：docker push
  1. 本地镜像管理
     1. 查看本地宿主机中docker镜像：docker images
     2. 删除本地宿主机中docker镜像：docker rmi

docker rmi repository:tag

* + 1. 显示一个镜像的历史：docker history

docker history image\_name

* + 1. 给镜像命名：docker tag

docker tag old\_repository:old\_tag new\_repository:new\_tag

* + 1. 使用Dockerfile创建镜像：docker build
    2. 保存和加载镜：docker save 和docker load

docker save image\_name -o file\_path：保存镜像到一个tar包; -o, --output="" Write to an file

docker load -i file\_path：加载一个tar包格式的镜像; -i, --input="" Read from a tar archive file

* + 1. 将归档tar文件导入指定镜像：docker import

docker import docker\_temp.tar respository:tag或者cat docker\_temp.tar | docker import - respository:tag

* 1. 容器生命周期管理：
     1. 创建docker 容器，并启动：docker run

例如：

Docker run –d –p 宿主机端口:容器端口 –e 变量名=变量值 -h=docker容器主机名（需要和镜像中的一致，否则启动后需要自己修改容器中主机名）--name 容器名 --privileged repository:tag /etc/init.d/after.local

* + 1. 创建docker容器但不启动：docker create
    2. 启动docker容器：docker start

docker start container\_id，除id外还可以是其他标识，只要能唯一标识该容器

* + 1. 停止docker容器：docker stop

docker stop container\_id，除id外还可以是其他标识，只要能唯一标识该容器

* + 1. 重启docker容器：docker restart

docker restart container\_id，除id外还可以是其他标识，只要能唯一标识该容器

* + 1. 删除docker容器：docker rm

docker rm container\_id,容器必须处于停止状态，如果要强制删除可用docker rm -f

* + 1. 杀死一个docker容器：docker kill

docker kill container\_id，除id外还可以是其他标识，只要能唯一标识该容器

* + 1. 暂停/恢复容器中的所有进程docker pause/docker unpause
    2. 在运行的容器中执行命令docker exec

docker exec -i container\_id

docker exec -it和我们平常操作console界面类似。而且也不会像attach方式因为退出，导致

整个容器退出。

* 1. 容器操作
     1. 查看docker容器：docker ps
     2. 查看容器信息：docker info

docker info container\_id，除id外还可以是其他标识，只要能唯一标识该容器

* + 1. 获取容器/镜像的元数据：docker inspect container\_id
    2. 查看容器中运行的进程信息，支持ps命令参数：docker top
    3. 链接到正在运行中的容器：docker attach

docker attach container\_id，可以在宿主机端连接上一个容器的标准输入（stdin），从而在容器内执行命令操作，但是如果从标准输入（stdin）退出（exit），会导致容器的停止。docker events

* + 1. 查看容器日志信息：docker logs

docker logs container\_id

* + 1. 阻塞运行直到容器停止，然后打印出它的退出码：docker wait
    2. 将容器导出为一个归档tar文件：docker export

docker export container\_name/container\_id > docker\_temp.tar docker port

* + 1. 查看容器端口映射信息：docker port

docker port container\_id，除id外还可以是其他标识，只要能唯一标识该容器

* 1. 容器rootfs命令
     1. 将容器提交成镜像：docker commit

docker commit container\_id repository:tag

* + 1. 在容器和宿主机之间拷贝文件：docker cp

从容器拷贝到本地

docker cp Name:/container\_path to\_path

docker cp ID:/container\_path to\_path

或者反过来，从本地拷贝到容器

* + 1. 列出一个容器里面被改变的文件或者目录，list列表会显示出三种事件，A 增加的，D 删除的，C 被改变的：docker diff
  1. 批量操作
     1. 批量删除所有镜像

docker rmi ` docker images | awk ’{print $3}’`

或

docker rmi $(docker images | awk ’{print $3})

或

docker rmi $(docker images -q)

* + 1. 批量删除已经停止的容器：

docker rm –f `docker ps –a | grep “Exited” | awk ‘{print $1}’ `

或

docker rm –f $(docker ps –a | grep “Exited” | awk ‘{print $1}’ )

或

sudo docker rm $(sudo docker ps -a -q)

或

sudo docker rm $(sudo docker ps -qf status=exited)