容器

一. 容器是镜像的运行实例，带有额外的可写文件层。Docker容器层是独立运行的一个或一组应用，以及必需运行环境

二. 当利用Docker run 来创建并启动容器，Docker在后台运行的标准操作

1. 检查本地是否存在指定的镜像，不存在就在公有仓库下载
2. 利用本地镜像创建并启动一个容器
3. 分配一个文件系统，并在只读的镜像层外面挂载一层可写层
4. 从宿主主机配置的网桥接口中桥接一个虚拟接口到容器中去
5. 从地址池配置一个IP地址给容器
6. 执行用户指定的应用程序
7. 执行完毕后容器被终止

三. 新建并启动容器

[root@Docker ~]# **docker run centos /bin/echo hiya**

[root@Docker ~]# **docker run -it centos /bin/bash**

- ✓ -t:让Docker分配一个伪终端，并绑定到容器的标准输入上
- ✓ -i:让容器的标准输入保持打开

注意：当Ctrl+D或exit结束后，容器自动处于终止状态

- [root@Docker ~]# **docker run -d centos /bin/sh -c "while true;do echo hello world;sleep 1;done"** 守护态运行
- [root@Docker ~]# **docker logs dfc**获取容器的输出消息

四. 终止容器

- [root@Docker ~]# **docker stop dfc**
- [root@Docker ~]# **docker kill dfc**直接发送SIGKILL信号强制终止容器
- [root@Docker ~]# **docker stop -t=10 dfc** 先向容器发送SIGTERM信号，等待一段时间后（默认10秒）再发送SIGKILL信号终止容器
- [root@Docker ~]# **docker start dfc**
- [root@Docker ~]# **docker restart dfc**

五. 进入容器：attach命令、exec命令、nsenter工具等

- [root@Docker ~]# **docker run -itd centos /bin/bash**

  322cd6b80e25c670e4e94cfed6c391c9bcd7c8ea05008d0c975db9a2c8257bf7
- [root@Docker ~]# **docker attach 322**

  注意：attach，当多个窗口同时attach到同一个容器的时候，所有窗口同步显示，当某个窗口因命令堵塞，其他窗口无法执行操作
- docker exec

  [root@Docker ~]# **docker exec -ti 322 /bin/bash** 直接在容器内运行命令
- nsenter工具

  [root@Docker ~]# **PID=$(docker inspect --format "{{.State.Pid}}" 737)** 先找到容器的进程的PID

  [root@Docker ~]# **nsenter --target 12397 --mount --uts --ipc --net --pid** 连接容器

六. 删除容器

[root@Docker ~]# **docker rm 737**

[root@Docker ~]# **docker stop 737**

[root@Docker ~]# **docker rm -f 737**

-l：删除容器的连接，但保留容器

-v：删除容器挂载的数据卷

七. 导入和导出容器cat t

[root@Docker ~]# **docker ps -a**

[root@Docker ~]# **docker export 433 > test.tar**导出433容器到test.tar文件中

[root@Docker ~]# **cat test.tar | docker import - test/centos**

附： 导入镜像存储文件到本地的镜像库，导入容器快照到本地镜像库的本质区别：容器快照文件将丢弃所有的历史记录和元数据信息（保留容器当时的快照状态）镜像存储文件将保存完整记录。从容器快照文件导入时，可以重新指定标签等元数据信息