实训18：docker综合实训1

**一、实验目的**

复习常用docker命令，达到熟练掌握的目标

**二、实验描述**

1. 通过对之前知识和操作的复习，熟练掌握docker的常用命令
2. 主要步骤：
   * 通过Dockerfile文件构建新的镜像
   * 调试Dockerfile文件，排除简单的错误。并对新镜像重命名
   * 使用docker commit命令创建新的镜像
   * 如何运行容器和如何进入容器
   * Docker容器的常用操作

**三、实验环境**

* 系统版本：centos 7.6
* docker版本： 20.06.1-ce

**四、实验知识点**

* linux系统基础配置
* docker基础命令

**五、实验步骤**

**1. 下载或导入已保存在本地的CentOS基础镜像。**

docker pull centos:7

或

docker load -i /root/images/centos.tar.gz

**2. 进入/root/addvim目录。**

cd /root/addvim

**3. 编写Dockerfile。**

vim Dockerfile

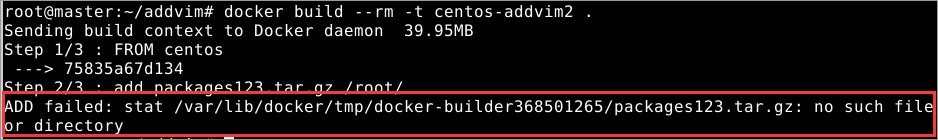
FROM centos

ADD packages123.tar.gz /root/

RUN yum install -y /root/addvim/\*.rpm

**4. 运行docker build构建新镜像，并新命名为centos-addvim2。但会出现明显错误提示。图片显示第二步出错，找不到文件。**

docker build –rm -t centos-addvim2 .



**5. 修改Dockerfile文件。**

vim Dockerfile

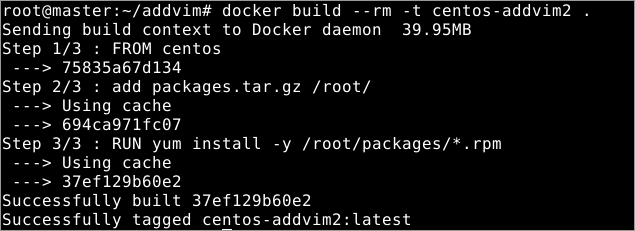
FROM centos

ADD packages.tar.gz /root/

RUN yum install -y /root/addvim/\*.rpm

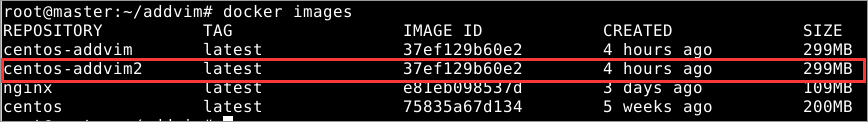
**6. 重新执行docker build。**

docker build –rm -t centos-addvim2 .



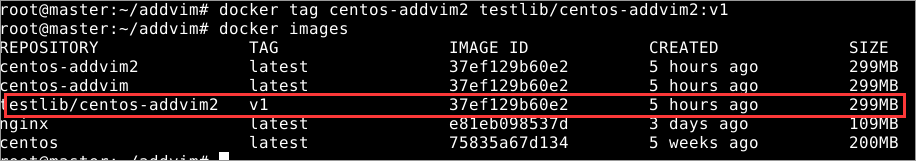
**7. 查看新镜像。**

docker images



**8. 更改镜像tag，并删除旧tag。将刚才的镜像tag更改为testlib/centos-addvim:v1。**

docker tag centos-addvim2 testlib/centos-addvim:v1



**5.9 删除centos2-addvim2。**

docker rmi centos-addvim2

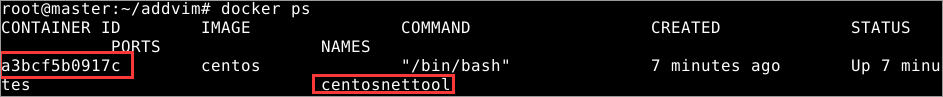


**10. 基于centos基础镜像，运行一个容器，并将容器命名为centosnettool：**

docker run -it -d --name centosnettool centos:7

**11. 查看运行的容器状态。**

docker ps



**12. 向容器中添加文件，将宿主机/root/nettoolsrpm目录复制到容器的/root目录下（更换红色部分为你的容器的CONTAINER ID）。**

docker cp nettoolsrpm a3bcf5b0917c:/root



**13. 进入容器（更换红色部分为你的容器的CONTAINER ID）。**

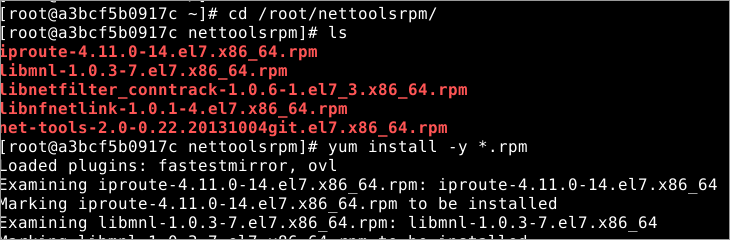
docker exec -it a3bcf5b0917c bash



**14. 安装net-tools等软件包。**

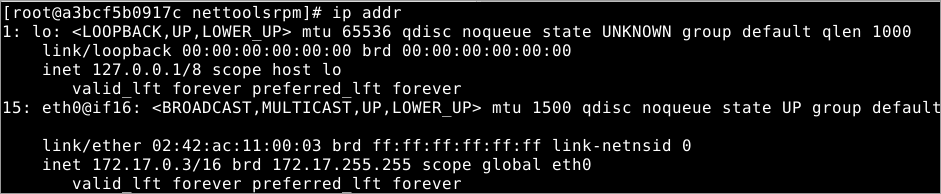
cd /root/nettoolsrpm

yum install -y \*.rpm



**15. 使用ip addr命令，测试软件包是否安装成功。**

ip addr



**16. 退出容器。**

exit



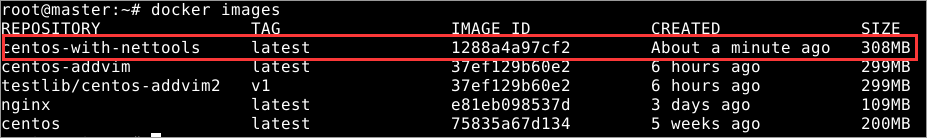
**17. 使用docker commit构建新镜像，新镜像名为centos-with-nettools：**

docker commit centosnettool centos-with-nettools



**18. 查看新生成的镜像。**

docker images



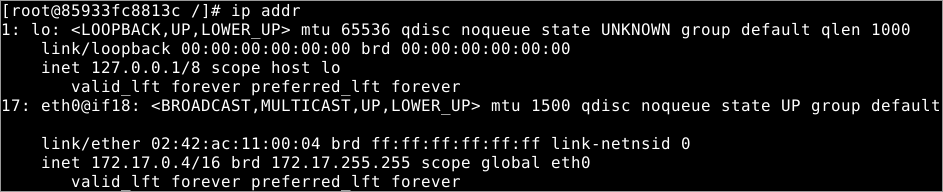
**19. 使用新镜像运行容器。**

docker run -it centos-with-nettools



**20. 使用ip addr命令：**

ip addr



**21. 运行centos容器并执行pwd命令。**

docker run centos pwd



**22. 使用docker ps命令查看，发现之前启动的centos容器不存在。**

docker ps



**23. 使用docker ps -a命令查看所有状态的容器，显示之前的容器状态为Exited。**

docker ps -a



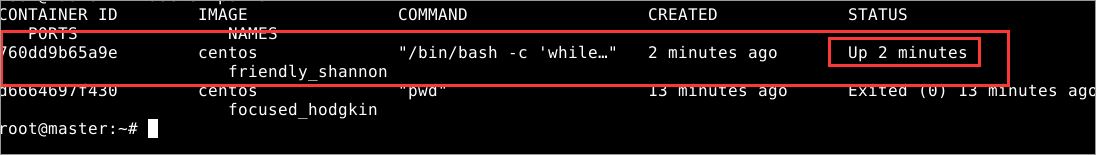
**24. 再运行一个centos容器，并执行一个无限循环命令。发现命令没有退出。**

docker run centos /bin/bash -c "while true;do sleep 1;done"



**25. 在另外一个终端查看容器状态。**

docker ps -a



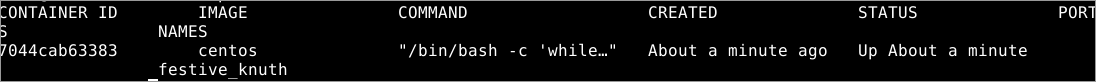
**26. 再运行一个centos容器，并执行一个无限循环命令，加上-d参数。**

docker run -d centos /bin/bash -c "while true;do sleep 1;done"



**27. 查看容器状态，发现容器状态为运行中。**

docker ps



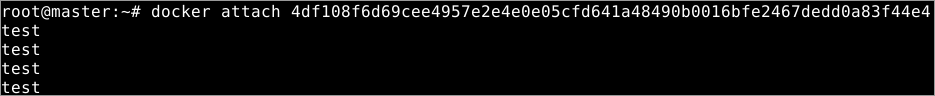
**28. 运行一个后台运行的centos容器，并执行一个无限打印文本“test”：**

docker run -d centos /bin/bash -c "while true;do sleep 1;echo test;done"



**29. 使用docker attach进入容器启动时使用命令的终端，发现每1秒输出一个test(替换红色部分为你的容器ID)。**

docker attach 4df



**30. 使用docker exec进入容器，并启动一个新的bash终端。命令中-it代表以交互模式打开 pseudo-TTY，执行 bash，其结果就是打开了一个 bash 终端。**

docker exec -it 4df108f6d69c bash



**31. 输入exit命令可以退出容器。**

exit



**32. 使用docker logs查看容器启动命令的日志输出。**

docker logs -f 4df108f6d69c

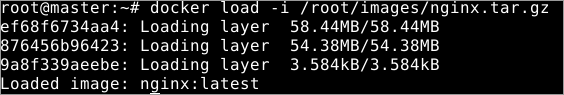


**33. 下载或导入已保存在本地的nginx基础镜像。**

docker pull nginx

或

docker load -i /root/images/nginx.tar.gz



**34. 启动一个nginx容器，并使其在后台运行。**

docker run -d --name nginx1 nginx



**35. 查看刚刚启动的nginx1容器的状态。**

docker ps



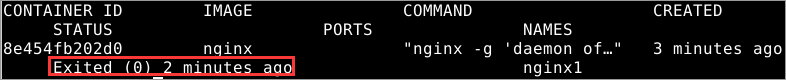
**36. 停止刚刚创建的名为nginx1的容器。**

docker stop nginx1



**37. 查看nginx1的状态。**

docker ps -a



**38. 启动nginx1容器。**

docker start nginx1



**39. 重启nginx1容器。**

docker restart nginx1



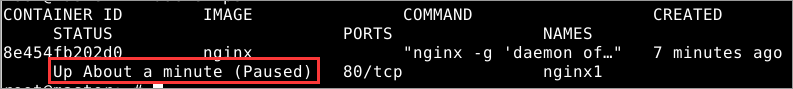
**40. 暂停nginx1容器，被暂停的容器不会消耗CPU资源。**

docker pause nginx1



**41. 查看nginx1的状态：**

docker ps



**42. 恢复nginx1容器。**

docker unpause nginx1



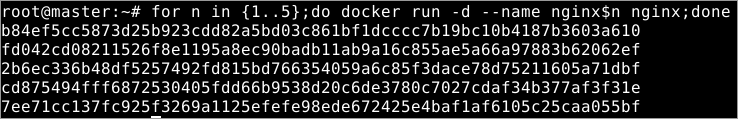
**43. 删除nginx1容器，-f选择表示强制删除，负责需要先stop容器，才能删除。**

docker rm -f nginx1



**44. 批量启动5个nginx容器，名称为nginx1到nginx5。**

for n in {1..5};do docker run -d --name nginx$n nginx;done





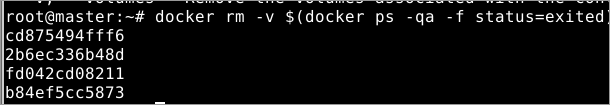
**45. 将刚才启动的nginx1到nginx4停止。**

for n in {1..4};do docker stop nginx$n;done



**46. 删除所有退出状态的容器，-v表示删除所有卷。**

docker rm -v $(docker ps -qa -f status=exited)



**六、总结**

本实验介绍了容器的一些常见操作。大家可以通过docker –help查看更多操作。