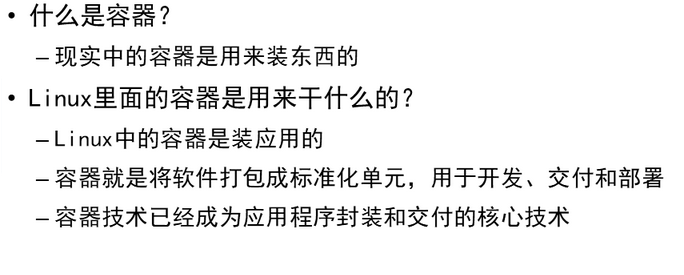
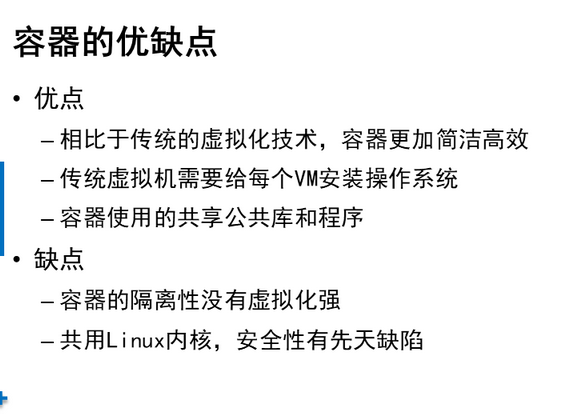
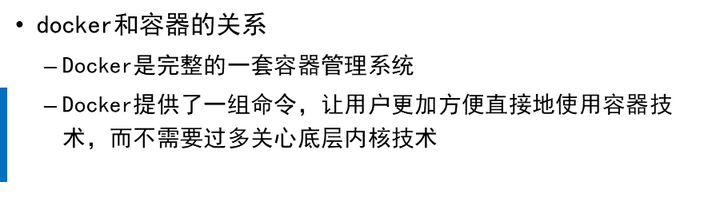
**NSD CLOUD DAY05**

****

****

****

****

1. [案例1：实验环境准备](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY05/CASE/01/index.html#case1)
2. [案例2：镜像管理命令](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY05/CASE/01/index.html#case2)
3. [案例3：镜像管理命令](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY05/CASE/01/index.html#case3)
4. [案例4：镜像管理命令二](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY05/CASE/01/index.html#case4)
5. [案例5：容器管理命令](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY05/CASE/01/index.html#case5)
6. [案例6：练习容器的执行方式](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY05/CASE/01/index.html#case6)

**1 案例1：实验环境准备**

**1.1 问题**

本案例要求为主机安装docker软件，具体要求如下：

1. 在跳板机192.168.1.252配置Docker的YUM服务器
2. 额外准备2台虚拟机，最低配置: 2CPU，2G内存，10G硬盘
3. node-0001 主机 IP:192.168.1.31
4. node-0002 主机 IP:192.168.1.32
5. 推荐CentOS7或RHEL7
6. 关闭防火墙和SELinux
7. 在两台机器上安装部署docker服务

**1.2 方案**

完成后续课程的学习需要提前准备实验用的虚拟机，实验虚拟机列表如表-1所示。

所有主机的主机名和IP必须与列表相同!!!

否则后续所有试验都无法顺利完成！！！

表-1



**1.3 步骤**

实现此案例需要按照如下步骤进行。

步骤一：在跳板机（192.168.1.252）配置Docker的YUM服务器

1）配置YUM服务器。

提示：相关软件已经提前共享到云盘，相关资料在kubernetes/目录下。

拷贝docker相关软件到跳板机，并创建私有YUM仓库服务器。

1. [root@localhost ~]# cp -a kubernetes/docker /var/ftp/localrepo/
2. [root@localhost ~]# cd /var/ftp/localrepo
3. [root@localhost ~]# createrepo --update .

步骤二：node-0001和node-0002安装Docker软件

node-0001和node-0002做相同操作，下面以一台主机操作为例。

1）关闭防火墙和SELinux。

1. [root@node-0001 ~]# vim /etc/selinux/config
2. ... ...
3. SELINUX=disabled
4. [root@node-0001 ~]# yum -y remove firewalld-\*
5. ... ...
6. [root@node-0001 ~]# reboot
7. ... ...
8. [root@node-0001 ~]# sestatus
9. SELinux status: disabled

2）配置YUM源

1. [root@node-0001 ~]# vim /etc/yum.repos.d/local.repo
2. [local\_repo]
3. name=CentOS-$releasever – Localrepo
4. baseurl=ftp://192.168.1.252/localrepo
5. enabled=1
6. gpgcheck=0
7. [root@node-0001 ~]# yum makecache #清空缓存
8. [root@node-0001 ~]# yum list docker-ce\* #查看软件列表
9. Loaded plugins: fastestmirror
10. docker-ce.x86\_64 3:18.06.3-3.el7 @local\_software

3）安装docker软件并启动服务

1. [root@localhost ~]# yum install -y docker-ce #安装软件
2. [root@localhost ~]# systemctl enable docker
3. [root@localhost ~]# systemctl start docker

**2 案例2：镜像管理命令**

**2.1 问题**

本案例熟悉docker镜像管理的命令，分别实现以下目标：

1. search 查找
2. pull 下载
3. save备份
4. load恢复

**2.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：查看帮助，熟悉命令格式（可以在node-0001或者node-0002操作）

1. [root@node-0001 ~]# docker help search
2. [root@node-0001 ~]# docker help pull
3. [root@node-0001 ~]# docker help save
4. [root@node-0001 ~]# docker help load

**3 案例3：镜像管理命令**

**3.1 问题**

本案例练习docker镜像管理的命令，分别实现以下目标：

1. 导入4个镜像（centos，nginx，redis，ubuntu）
2. 使用镜像CentOS启动容器
3. 使用镜像busybox启动容器
4. 使用镜像nginx启动容器

**3.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：导入镜像（可以在node-0001或者node-0002操作）

镜像都已经提前共享到云盘，请提前下载云盘中的镜像，并拷贝到实验虚拟机中。

所有镜像在云盘第四阶段的kubernetes/docker-images/目录下。

1. [root@node-0001 ~]# docker load -i centos.tar.gz
2. [root@node-0001 ~]# docker load -i nginx.tar.gz
3. [root@node-0001 ~]# docker load -i redis.tar.gz
4. [root@node-0001 ~]# docker load -i ubuntu.tar.gz

步骤二：使用镜像，运行容器

1）运行容器

可以通过docker help run或者man docker-run查看帮助。

法法格式如下：docker run -参数 镜像名称：镜像标签 启动命令

run命令 = 创建 + 启动 + 进入

docker run 命令的重要参数如下

* 参数 -i，交互式
* 参数 -t，终端
* 参数 -d，后台运行
* 参数 --name 容器名字

启动 centos 容器，并进入容器

1. [root@node-0001 ~]# docker run -it --name myos centos:latest /bin/bash

使用docker命令启动容器，可以通过提示符判定自己是否进入容器

1. [root@node-0001 ~]# docker run -it centos:latest /bin/bash
2. [root@10d70724abf2 /]# ps –ef #可以看到命令提示符已经变了
3. UID PID PPID C STIME TTY TIME CMD
4. root 1 0 0 12:52 ? 00:00:00 /bin/bash
5. root 14 1 0 12:53 ? 00:00:00 ps -ef

2）其他镜像的使用

所有其他镜像使用方法类似，下面再看几个例子

1. [root@node-0001 ~]# docker run -it nginx /bin/bash
2. [root@node-0001 ~]# docker run -it ubuntu /bin/bash

**4 案例4：镜像管理命令二**

**4.1 问题**

本案例要求进一步熟练掌握以下镜像管理命令，主要完成内容：

1. search、images、load、save、pull
2. tag、inspect、history、rmi、push

**4.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：镜像管理命令的使用

查看镜像

1. [root@node-0001 ~]# docker images

搜索镜像（默认需要连接外网才可以）

1. [root@node-0001 ~]# docker search 镜像名称

删除镜像

1. [root@node-0001 ~]# docker rmi 镜像名称:镜像标签

上传下载镜像（默认需要连接外网才可以）

1. [root@node-0001 ~]# docker pull 镜像名称:镜像标签 #下载镜像
2. [root@node-0001 ~]# docker push 要上传的镜像名称:镜像标签 #上传镜像

备份镜像

1. [root@node-0001 ~]# docker save 镜像名称:镜像标签 -o 文件名称
2. # -o选项指定将镜像备份到哪个文件

恢复镜像

1. [root@node-0001 ~]# docker load -i 备份文件名称

查看镜像的制作历史

1. [root@node-0001 ~]# docker history 镜像名称:镜像标签

查看镜像的信息

1. [root@node-0001 ~]# docker inspect 镜像名称:镜像标签

镜像的新名称和标签

1. [root@node-0001 ~]# docker tag 镜像名称:镜像标签 新镜像名称:新的标签

**5 案例5：容器管理命令**

**5.1 问题**

本案例要求熟练掌握以下容器管理命令，主要练习以下命令：

1. run、stop、start、restart、ps、cp
2. rm、inspect、top、attach、exec

**5.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：容器管理命令

启动容器

1. [root@node-0001 ~]# docker run -参数 镜像名称:镜像标签 启动命令

查看容器

1. [root@node-0001 ~]# docker ps [ -a ] [ -q ]
2. #[]代表可选参数，可以使用-a或-q也可以不适用
3. #-a代表查看所有容器的信息
4. #-q只显示容器的id号

删除容器

1. [root@node-0001 ~]# docker rm 容器id #根据容器ID，删除某个已启动的容器
2. [root@node-0001 ~]# docker rm $(docker ps -aq) #删除已经启动所有容器

启动、停止、重启容器的命令

1. [root@node-0001 ~]# docker start 容器id #启动容器
2. [root@node-0001 ~]# docker stop 容器id #关闭容器
3. [root@node-0001 ~]# docker restart 容器id #重启容器

将真机文件拷贝到容器中

1. [root@node-0001 ~]# docker cp 本机文件路径 容器id:容器内路径
2. #该命令可以将真机的某个文件上传到容器中的某个路径下
3. [root@node-0001 ~]# docker cp 容器id:容器内路径 本机文件路径
4. #该命令可以将容器中的某个文件下载到真机的某个路径下

查看容器信息

1. [root@node-0001 ~]# docker inspect 容器id

进入容器(退出会关闭)

1. [root@node-0001 ~]# docker attach 容器id

进入容器(退出不关闭)

1. [root@node-0001 ~]# docker exec -it 容器id 启动命令

**6 案例6：练习容器的执行方式**

**6.1 问题**

本案例要求理解容器的执行方式：

1. 前台服务
2. 后台服务
3. 创建一个 centos 的容器，并为他设置 yum 源

**6.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：概念

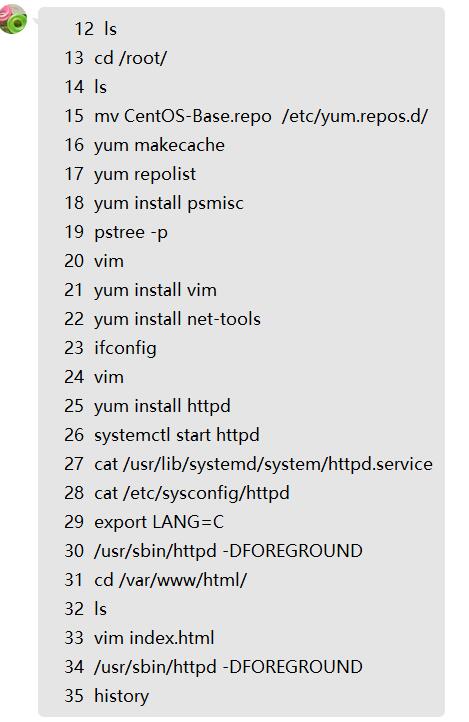
容器启动服务的方式

前台服务（-it）：一般是能与用户交互的程序，比如 /bin/bash、/bin/sh 等

后台服务（-itd）：一般是一个程序服务，比如 apache、nginx、redis 等

步骤二：命令练习

1. [root@node-0001 ~]# docker run -d centos:latest        # 失败
2. c2219228afc14e7c87b20280fcb5793f006a24a360433c107a3ab5a9dee34047
3. [root@node-0001 ~]# docker ps -a
4. CONTAINER ID IMAGE COMMAND CREATED STATUS NAMES
5. c2219228afc1 centos:latest "/bin/bash" 5 seconds ago Exited (0) xx
6. [root@node-0001 ~]# docker run -itd centos:latest        # 成功



容器里面搭建appach