[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/CLOUD/DAY05/CASE/01/index.html" \l "page_top_case)

# NSD CLOUD DAY05

1. [案例1：制作自定义镜像](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/CLOUD/DAY05/CASE/01/index.html" \l "case1)
2. [案例2：创建私有镜像仓库](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/CLOUD/DAY05/CASE/01/index.html" \l "case2)
3. [案例3：NFS共享存储](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/CLOUD/DAY05/CASE/01/index.html" \l "case3)
4. [案例4：创建自定义网桥](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/CLOUD/DAY05/CASE/01/index.html" \l "case4)

## 1 案例1：制作自定义镜像

### 1.1 问题

本案例要求制作自定义镜像：

* 基于centos镜像使用commit创建新的镜像文件
* 基于centos镜像使用Dockerfile文件创建一个新的镜像文件

### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用镜像启动容器

1）在该容器基础上修改yum源

1. [root@docker1 docker\_images]# docker run -it docker.io/centos
2. [root@8d07ecd7e345 /]# rm -rf /etc/yum.repos.d/\*
3. [[root@8d07ecd7e345](mailto:root@8d07ecd7e345) /]# vi /etc/yum.repos.d/dvd.repo
4. [dvd]
5. name=dvd
6. baseurl=<ftp://192.168.1.254/system>
7. enabled=1
8. gpgcheck=0
9. [[root@8d07ecd7e345](mailto:root@8d07ecd7e345) /]# yum clean all
10. [[root@8d07ecd7e345](mailto:root@8d07ecd7e345) /]# yum repolist

2）安装测试软件

1. [root@8d07ecd7e345 /]# yum -y install net-tools iproute psmisc vim-enhanced

3）ifconfig查看

1. [root@8d07ecd7e345 /]# ifconfig
2. eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
3. inet 172.17.0.3 netmask 255.255.0.0 broadcast 0.0.0.0
4. inet6 fe80::42:acff:fe11:3 prefixlen 64 scopeid 0x20<link>
5. ether 02:42:ac:11:00:03 txqueuelen 0 (Ethernet)
6. RX packets 2488 bytes 28317945 (27.0 MiB)
7. RX errors 0 dropped 0 overruns 0 frame 0
8. TX packets 1858 bytes 130264 (127.2 KiB)
9. TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
10. [root@8d07ecd7e345 /]# exit
11. exit

步骤二：另存为另外一个镜像

1）创建新建镜像

1. [root@docker1 docker\_images]# docker start 8d07ecd7e345
2. //可以简写为8d，要保证唯一性
3. 8d07ecd7e345
4. [root@docker1 docker\_images]# docker commit 8d07ecd7e345 docker.io/myos:latest
5. sha256:ac3f9c2e8c7e13db183636821783f997890029d687b694f5ce590a473ad82c5f

2）查看新建的镜像

1. [root@docker1 ~]# docker images
2. REPOSITORY TAG IMAGE ID CREATED SIZE
3. docker.io/myos latest 87feda116c17 6 weeks ago 321.6 MB
4. docker.io/busybox latest d8233ab899d4 7 weeks ago 1.199 MB
5. docker.io/nginx latest f09fe80eb0e7 9 weeks ago 109.2 MB
6. docker.io/redis latest 82629e941a38 11 weeks ago 94.98 MB
7. docker.io/ubuntu latest 20bb25d32758 11 weeks ago 87.47 MB
8. docker.io/registry latest 116995fd6624 11 weeks ago 25.76 MB
9. docker.io/centos latest 1e1148e4cc2c 4 months ago 201.8 MB

3）验证新建镜像

1. [root@docker1 docker\_images]# docker run -it docker.io/myos:latest
2. [root@497c7b4664bf /]# ifconfig
3. eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
4. inet 172.17.0.6 netmask 255.255.0.0 broadcast 0.0.0.0
5. inet6 fe80::42:acff:fe11:6 prefixlen 64 scopeid 0x20<link>
6. ether 02:42:ac:11:00:06 txqueuelen 0 (Ethernet)
7. RX packets 0 bytes 0 (0.0 B)
8. RX errors 0 dropped 0 overruns 0 frame 0
9. TX packets 7 bytes 578 (578.0 B)
10. TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

步骤三：使用Dockerfile文件创建一个新的镜像文件

Dockerfile语法格式：

– FROM:基础镜像

– MAINTAINER:镜像创建者信息(说明)

– EXPOSE:开放的端口

– ENV:设置环境变量

– ADD:复制文件到镜像

– RUN:制作镜像时执行的命令,可以有多个

– WORKDIR:定义容器默认工作目录

– CMD:容器启动时执行的命令,仅可以有一条CMD

1）创建一个Apache的镜像文件

1. [root@docker1 ~]# mkdir bulid
2. [root@docker1 ~]# cd bulid
3. [root@docker1 bulid]# touch Dockerfile    //Dockerfile文件第一个字母要大写
4. [root@docker1 bulid]# cp /etc/yum.repos.d/local.repo ./
5. [root@docker1 bulid]# vi Dockerfile
6. FROM docker.io/myos:latest
7. RUN yum -y install httpd
8. ENV EnvironmentFile=/etc/sysconfig/httpd
9. WORKDIR /var/www/html/            //定义容器默认工作目录
10. ADD index.html index.html
11. EXPOSE 80                //设置开放端口号
12. EXPOST 443
13. CMD ["/usr/sbin/httpd", "-DFOREGROUND"]
14. [root@docker1 bulid]# docker build -t docker.io/myos:http .
15. [root@docker1 bulid]# docker run -d docker.io/myos:http
16. d9a5402709b26b42cd304c77be442559a5329dc784ec4f6c90e4abac1c88e206
17. [root@docker1 bulid]# docker inspect d9
18. [root@docker1 bulid]# curl 172.17.0.7
19. test

## 2 案例2：创建私有镜像仓库

### 2.1 问题

本案例要求创建私有的镜像仓库：

* 在Docker1上创建私有仓库
* 上传镜像到 docker1
* 在 docker2 上配置使用 docker1 的私有仓库
* 在 docker2 上使用 docker1 的远程仓库启动容器步骤

实现此案例需要按照如下步骤进行。

步骤一：自定义私有仓库

1）定义一个私有仓库

1. [root@docker1 bulid]# yum install docker-distribution
2. [root@docker1 bulid]# systemctl start docker-distribution
3. [root@docker1 bulid]# systemctl enable docker-distribution
4. [root@docker1 ~]# docker tag docker.io/busybox:latest 192.168.1.31:5000/docker.io/busybox:latest
5. //打标签
6. [root@docker1 ~]# docker push 192.168.1.31:5000/docker.io/busybox:latest
7. //上传
8. [root@docker1 ~]# docker tag docker.io/myos:http 192.168.1.31:5000/ docker.io/myos:http
9. [root@docker1 ~]# docker push 192.168.1.31:5000/docker.io/myos:http

2）客户端配置

1. [root@docker2 ~]# vim /etc/sysconfig/docker
2. INSECURE\_REGISTRY='--insecure-registry docker1:5000'
3. ADD\_REGISTRY='--add-registry docker1:5000'
4. [root@docker2 ~]# systemctl restart docker
5. [root@docker2 ~]# docker run -it myos:http /bin/bash
6. //直接启动

步骤二：查看私有仓库

1）查看里面有什么镜像

1. [root@docker1 bulid]# curl http://docker1:5000/v2/\_catalog
2. {"repositories":["docker.io/busybox","docker.io/myos"]}

2）查看里面的镜像标签

1. [root@docker1 bulid]# curl http://docker1:5000/v2/docker.io/busybox/tags/list
2. {"name":"docker.io/busybox","tags":["latest"]}
3. [root@docker1 bulid]# curl http://192.168.1.31:5000/v2/docker.io/myos/tags/list
4. {"name":"docker.io/myos","tags":["http"]}

## 3 案例3：NFS共享存储

### 3.1 问题

本案例要求创建NFS共享，能映射到容器里：

* 服务器创建NFS共享存储，共享目录为/content，权限为rw
* 客户端挂载共享，并将共享目录映射到容器中

### 3.2 方案

本方案要求需要一台NFS服务器（NFS用真机代替），ip为192.168.1.254，一台客户端docker1主机，ip为192.168.1.10，一台户端docker2主机，ip为192.168.1.20，实现客户端挂载共享，并将共享目录映射到容器中，docker1更新文件时，docker2实现同步更新，方案如图-2所示：

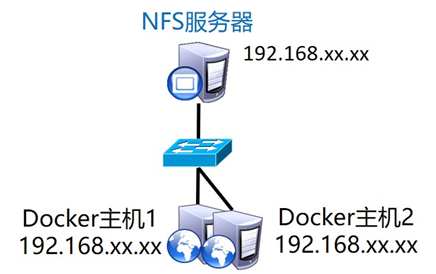


图-2

### 3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置NFS服务器

1. [root@room9pc01 ~]# yum -y install nfs-utils
2. [root@room9pc01 ~]# mkdir /content
3. [root@room9pc01 ~]# vim /etc/exports
4. /content \*(rw，no\_root\_squash)
5. [root@room9pc01 ~]# systemctl restart nfs-server.service
6. [root@room9pc01 ~]# systemctl restart nfs-secure.service
7. [root@room9pc01 ~]# exportfs -rv
8. exporting \*:/content
9. [root@room9pc01 ~]# chmod 777 /content
10. [root@room9pc01 ~]# echo 11 > /content/index.html

步骤二：配置客户端

1. [root@docker1 bulid]# yum -y install nfs-utils
2. [root@docker1 bulid]# systemctl restart nfs-server.service
3. [root@docker1 bulid]# showmount -e 192.168.1.254
4. Export list for 192.168.1.254:
5. /content \*
6. [root@docker1 ~]# mkdir /mnt/qq
7. [root@docker1 ~]# mount -t nfs 192.168.1.254:/content /mnt/qq
8. [root@docker1 ~]# ls /mnt/qq
9. index.html
10. [root@docker1 ~]# cat /mnt/qq/index.html
11. 11
12. [root@docker1 ~]# docker run -d -p 80:80 -v /mnt/qq:/var/www/html -it docker.io/myos:http
13. 224248f0df5d795457c43c2a7dad0b7e5ec86abdc3f31d577e72f7929f020e01
14. [root@docker1 ~]# curl 192.168.1.31
15. 11
16. [root@docker2 ~]# yum -y install nfs-utils
17. [root@docker2 ~]# showmount -e 192.168.1.254
18. Export list for 192.168.1.254:
19. /content \*
20. [root@docker2 ~]# mkdir /mnt/qq
21. [root@docker2 ~]# mount -t nfs 192.168.1.254:/content /mnt/qq
22. [root@docker2 ~]# docker run -d -p 80:80 -v /mnt/qq:/usr/share/nginx/html -it 192.168.1.10:5000/docker.io/myos:http
23. 00346dabec2c7a12958da4b7fee6551020249cdcb111ad6a1058352d2838742a
24. [root@docker2 ~]# curl 192.168.1.32
25. 11
26. [root@docker1 ~]# touch /mnt/qq/a.sh
27. [root@docker1 ~]# echo 22 > /mnt/qq/index.html
28. [root@docker2 ~]#ls /mnt/qq/
29. a.sh index.html
30. [root@docker2 ~]# cat /mnt/qq/index.html
31. 22

## 4 案例4：创建自定义网桥

### 4.1 问题

本案例要求：

* 启动4台容器
* 要求： 容器1 与 容器2 能够互通
* 容器3 与 容器4 能够互通
* 容器（12） 与 容器（34） 不能互通

如图-1所示

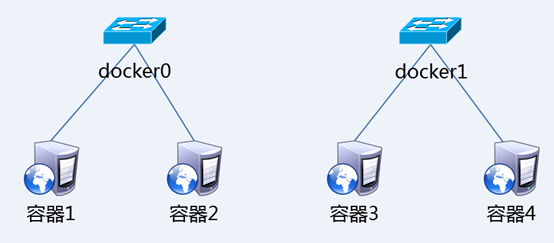


图-1

### 4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：新建Docker网络模型

1）新建docker1网络模型

1. [root@docker1 ~]# docker network create --subnet=10.10.10.0/24 docker1
2. b447cacc0373631ff7c534f119047946be5c1498b5b2e31a31180c5ee6320ab5
3. [root@docker1 ~]# docker network list
4. NETWORK ID NAME DRIVER SCOPE
5. 996943486faa bridge bridge local
6. b447cacc0373 docker1 bridge local
7. 63c88dcc3523 host host local
8. 5e5ab3d45e27 none null local
9. [root@docker1 ~]# ip a s
10. [root@docker1 ~]# docker network inspect docker1
11. [
12. {
13. "Name": "docker1",
14. "Id": "b447cacc0373631ff7c534f119047946be5c1498b5b2e31a31180c5ee6320ab5",
15. "Scope": "local",
16. "Driver": "bridge",
17. "EnableIPv6": false,
18. "IPAM": {
19. "Driver": "default",
20. "Options": {},
21. "Config": [
22. {
23. "Subnet": "10.10.10.0/24"
24. } ]
25. },
26. "Internal": false,
27. "Containers": {},
28. "Options": {},
29. "Labels": {}
30. }
31. ]

2）使用自定义网桥（docker1）启动容器

1. [root@docker1 ~]# docker run --network=docker1 -itd docker.io/myos
2. 5270cba305c06c3da3f56185b35dc059aabcf2884a12ef717d89a768360e5326
3. [root@docker1 ~]# docker run --network=docker1 -itd docker.io/myos
4. 4b4a4e8bebfbcc18a0deaa17225f0b5dec8c6d5d52e513617849c9579b0b1813
5. [root@docker1 ~]# docker network inspect docker1 //可以看到容器的ip
6. [
7. ......
8. "Internal": false,
9. "Containers": {
10. "4b4a4e8bebfbcc18a0deaa17225f0b5dec8c6d5d52e513617849c9579b0b1813": {
11. "Name": "big\_spence",
12. "EndpointID": "d5894002a9fdfd65daf52473de1735ecdc32ef53832099afc1dcfa6e86a5e8f4",
13. "MacAddress": "02:42:0a:0a:0a:03",
14. "IPv4Address": "10.10.10.3/24",
15. "IPv6Address": ""
16. },
17. "5270cba305c06c3da3f56185b35dc059aabcf2884a12ef717d89a768360e5326": {
18. "Name": "infallible\_lalande",
19. "EndpointID": "492a8cdda204f23775978758f364d577783272c83cf1a5de2d20bf640f060f05",
20. "MacAddress": "02:42:0a:0a:0a:02",
21. "IPv4Address": "10.10.10.2/24",
22. "IPv6Address": ""
23. }
24. },
25. "Options": {},
26. "Labels": {}
27. }
28. ]

3）使用默认网桥（docker0）启动容器

1. [root@docker1 ~]# docker run -itd docker.io/myos
2. 63e99284b1a78d7d5fe17d25697424502054c59e0cc61b58c3070758fff1c35d
3. [root@docker1 ~]# docker run -itd docker.io/myos
4. f41cb77a6fe0574ce5b810498d6f42223e55d677df391d050a2901c678dfea3f
5. [root@docker1 ~]# docker inspect -f '{{.NetworkSettings.IPAddress}}' f41
6. 172.17.0.3
7. [root@docker1 ~]# docker inspect -f '{{.NetworkSettings.IPAddress}}' 63e
8. 172.17.0.2

4）测试

1. [root@docker1 ~]# docker exec -it ae /bin/bash
2. [root@ae1500292be8 /]# ping 172.17.0.3 //可以ping通
3. PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
4. 64 bytes from 172.17.0.3: icmp\_seq=1 ttl=255 time=0.140 ms
5. 64 bytes from 172.17.0.3: icmp\_seq=2 ttl=255 time=0.107 ms
6. [root@ae1500292be8 /]# ping 10.10.10.2 //ping不通
7. PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data

步骤二：扩展实验

1）新建一个网络模型docker02

1. [root@docker1 ~]# docker network create --driver bridge docker02
2. //新建一个 名为docker02的网络模型
3. 5496835bd3f53ac220ce3d8be71ce6afc919674711ab3f94e6263b9492c7d2cc
4. [root@docker1 ~]# ifconfig
5. //但是在用ifconfig命令查看的时候，显示的名字并不是docker02，而是br-5496835bd3f5
6. br-5496835bd3f5: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
7. inet 172.18.0.1 netmask 255.255.0.0 broadcast 0.0.0.0
8. ether 02:42:89:6a:a2:72 txqueuelen 0 (Ethernet)
9. RX packets 8 bytes 496 (496.0 B)
10. RX errors 0 dropped 0 overruns 0 frame 0
11. TX packets 8 bytes 496 (496.0 B)
12. TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
13. [root@docker1 ~]# docker network list            //查看显示docker02（查看加粗字样）
14. NETWORK ID NAME DRIVER SCOPE
15. bc189673f959 bridge bridge local
16. 5496835bd3f5 docker02 bridge local
17. 53bf43bdd584 host host local
18. ac52d3151ba8 none null local

2）若要解决使用ifconfig命令可以看到docker02的问题，可以执行以下几步命令

1. [root@docker1 ~]# docker network list //查看docker0的NETWORK ID（加粗字样）
2. NETWORK ID NAME DRIVER SCOPE
3. bc189673f959 bridge bridge local
4. 5496835bd3f5 docker02 bridge local
5. 53bf43bdd584 host host local
6. ac52d3151ba8 none null local

3）查看16dc92e55023的信息，如图-3所示：

1. [root@docker2 ~]# docker network inspect bc189673f959



图-3

4）查看图片的倒数第六行有"com.docker.network.bridge.name": "docker0"字样

5）把刚刚创建的docker02网桥删掉

1. [root@docker1 ~]# docker network rm docker02 //删除docker02
2. docker02
3. [root@docker1 ~]# docker network create \
4. docker02 -o com.docker.network.bridge.name=docker02
5. //创建docker02网桥
6. 648bd5da03606d5a1a395c098662b5f820b9400c6878e2582a7ce754c8c05a3a
7. [root@docker1 ~]# ifconfig     //ifconfig查看有docker02
8. docker02: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
9. inet 172.18.0.1 netmask 255.255.0.0 broadcast 0.0.0.0
10. ether 02:42:94:27:a0:43 txqueuelen 0 (Ethernet)
11. RX packets 0 bytes 0 (0.0 B)
12. RX errors 0 dropped 0 overruns 0 frame 0
13. TX packets 0 bytes 0 (0.0 B)
14. TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

6）若想在创建docker03的时候自定义网段（之前已经创建过docker01和02，这里用docker03），执行以下命令

1. [root@docker1 ~]# docker network create docker03 --subnet=172.30.0.0/16 -o com.docker.network.bridge.name=docker03
2. f003aa1c0fa20c81e4f73c12dcc79262f1f1d67589d7440175ea01dc0be4d03c
3. [root@docker1 ~]# ifconfig //ifconfig查看，显示的是自己定义的网段
4. docker03: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
5. inet 172.30.0.1 netmask 255.255.0.0 broadcast 0.0.0.0
6. ether 02:42:27:9b:95:b3 txqueuelen 0 (Ethernet)
7. RX packets 0 bytes 0 (0.0 B)
8. RX errors 0 dropped 0 overruns 0 frame 0
9. TX packets 0 bytes 0 (0.0 B)
10. TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0