TTS 10.0 COOKBOOK

(NSD CLOUD DAY06)

版本编号 10.0

2018-08

达内 IT 培训集团

NSD CLOUD DAY06

1. 案例 1：制作自定义镜像

- **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

- **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

```
[root@docker1 docker_images]# docker run -it centos
[root@8d07ecd7e345 /]# rm -rf /etc/yum.repos.d/*
[root@8d07ecd7e345 /]# vi /etc/yum.repos.d/dvd.repo
[dvd]
name=dvd
baseurl=ftp://192.168.1.254/system
enabled=1
gpgcheck=0
[root@8d07ecd7e345 /]# yum clean all
[root@8d07ecd7e345 /]# yum repolist
```

2）安装测试软件

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

3）ifconfig 查看

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

步骤二：另存为另外一个镜像

1）创建新建镜像

```
[root@docker1 docker_images]# docker start 8d07ecd7e345
//可以简写为 8d，要保证唯一性
8d07ecd7e345
```

```
[root@docker1 docker_images]# docker commit 8d07ecd7e345 myos:v1
sha256:ac3f9c2e8c7e13db183636821783f997890029d687b694f5ce590a473ad82c5f
```

2）查看新建的镜像，如图-1 所示：

```
[root@docker1 docker_images]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myos                v1                  ac3f9c2e8c7e        12 seconds ago      316.4 MB
centos              latest              5182e96772bf        4 weeks ago         199.7 MB
busybox             latest              e1ddd7948a1c        5 weeks ago         1.163 MB
registry            latest              b2b03e9146e1        8 weeks ago         33.29 MB
ubuntu              latest              452a96d81c30        4 months ago        79.62 MB
cen                 v1                  e934aafc2206        5 months ago        198.6 MB
nginx               latest              a5311a310510        23 months ago       181.4 MB
redis               latest              1aa84b1b434e        23 months ago       182.8 MB
[root@docker1 docker_images]#
```

图-1

3）验证新建镜像

```
[root@docker1 docker_images]# docker run -it myos:v1
[root@497c7b4664bf /]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.6  netmask 255.255.0.0  broadcast 0.0.0.0
        inet6 fe80::42:acff:fe11:6  prefixlen 64  scopeid 0x20<link>
        ether 02:42:ac:11:00:06  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7  bytes 578 (578.0 B)
        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

```
[root@docker1 ~]# mkdir oo
[root@docker1 ~]# cd oo
[root@docker1 oo]# touch Dockerfile    //Dockerfile 文件第一个字母要大写
[root@docker1 oo]# cp /etc/yum.repos.d/local.repo  ./
[root@docker1 oo]# vi Dockerfile
FROM  myos:v1
RUN yum -y install httpd
```

```
ENV EnvironmentFile=/etc/sysconfig/httpd
WORKDIR /var/www/html/              //定义容器默认工作目录
RUN echo "test" > /var/www/html/index.html
EXPOSE 80              //设置开放端口号
CMD ["/usr/sbin/httpd", "-DFOREGROUND"]
[root@docker1 oo]# docker build -t myos:http .
[root@docker1 oo]# docker run -d  myos:http
d9a5402709b26b42cd304c77be442559a5329dc784ec4f6c90e4abac1c88e206
[root@docker1 oo]# docker inspect d9
[root@docker1 oo]# curl 172.17.0.7
test
```

2. 案例 2：创建私有镜像仓库

- **问题**

本案例要求创建私有的镜像仓库：

- Docker 主机：192.168.1.20
- 镜像仓库服务器：192.168.1.10

- **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

```
[root@docker1 oo]# vim /etc/docker/daemon.json  //不写这个文件会报错
{
"insecure-registries" : ["192.168.1.10:5000"]   //使用私有仓库运行容器
}
[root@docker1 oo]# systemctl restart docker
[root@docker1 oo]# docker run -d -p 5000:5000 registry
273be3d1f3280b392cf382f4b74fea53aed58968122eff69fd016f638505ee0e
[root@docker1 oo]# curl 192.168.1.10:5000/v2/
{}    //出现括号
[root@docker1 oo]# docker tag busybox:latest 192.168.1.10:5000/busybox:latest
//打标签
[root@docker1 oo]# docker push 192.168.1.10:5000/busybox:latest  //上传
[root@docker1 oo]# docker tag myos:http 192.168.1.10:5000/myos:http
[root@docker1 oo]# docker push  192.168.1.10:5000/myos:http
```

2）在 docker2 上面启动

```
[root@docker2 ~]# scp 192.168.1.10:/etc/docker/daemon.json  /etc/docker/
[root@docker2 ~]# systemctl restart docker
[root@docker2 ~]# docker images
[root@docker2 ~]# docker run -it 192.168.1.10:5000/myos:http /bin/bash
//直接启动
```

步骤二：查看私有仓库

1）查看里面有什么镜像

```
[root@docker1 oo]# curl  http://192.168.1.10:5000/v2/_catalog
{"repositories":["busybox","myos"]}
```

2）查看里面的镜像标签

```
[root@docker1 oo]# curl  http://192.168.1.10:5000/v2/busybox/tags/list
{"name":"busybox","tags":["latest"]}
[root@docker1 oo]# curl  http://192.168.1.10:5000/v2/myos/tags/list
{"name":"myos","tags":["http"]}
```

3. 案例 3：NFS 共享存储

- **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

- **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：

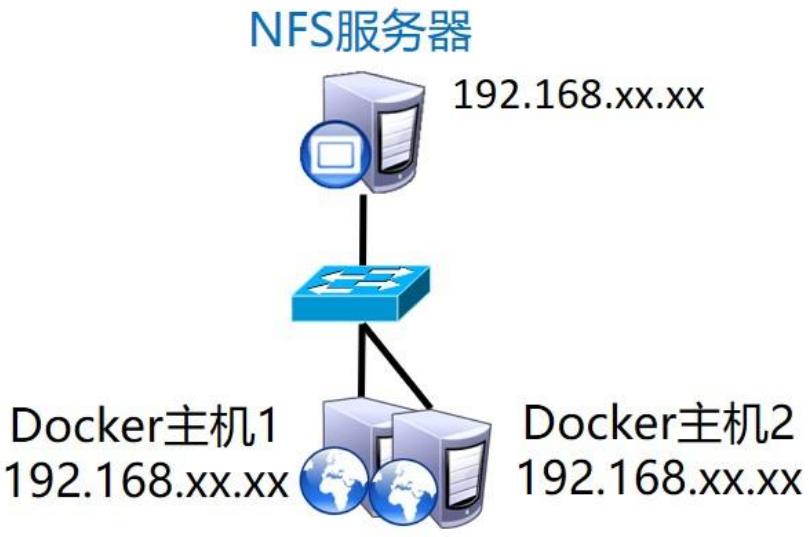

图-2

- **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

```
[root@room9pc01 ~]# yum -y install nfs-utils
[root@room9pc01 ~]# mkdir /content
[root@room9pc01 ~]# vim /etc/exports
/content     *(rw，no_root_squash)
[root@room9pc01 ~]# systemctl restart nfs-server.service
[root@room9pc01 ~]# systemctl restart nfs-secure.service
[root@room9pc01 ~]# exportfs  -rv
exporting *:/content
[root@room9pc01 ~]# chmod 777 /content
[root@room9pc01 ~]# echo 11 > /content/index.html
```

步骤二：配置客户端

1）在 docker1 上面配置

```
[root@docker1 oo]# yum -y install nfs-utils
[root@docker1 oo]# systemctl restart nfs-server.service
[root@docker1 oo]# showmount -e 192.168.1.254
Export list for 192.168.1.254:
/content *
[root@docker1 ~]# mkdir /mnt/qq
[root@docker1 ~]# mount -t nfs 192.168.1.254:/content /mnt/qq
[root@docker1 ~]# ls /mnt/qq
index.html
[root@docker1 ~]# cat  /mnt/qq/index.html
11
[root@docker1 ~]# docker run  -d -p 80:80 -v /mnt/qq:/var/www/html -it myos:http
224248f0df5d795457c43c2a7dad0b7e5ec86abdc3f31d577e72f7929f020e01
[root@docker1 ~]# curl 192.168.1.10
11
```

2）在 docker2 上面配置

```
[root@docker2 ~]# yum -y install nfs-utils
[root@docker2 ~]# showmount -e 192.168.1.254
Export list for 192.168.1.254:
/content *
[root@docker2 ~]# mkdir /mnt/qq
[root@docker2 ~]# mount -t nfs 192.168.1.254:/content /mnt/qq
[root@docker2  ~]#  docker  run    -d  -p  80:80  -v  /mnt/qq:/var/www/html  -it
192.168.1.10:5000/myos:http
00346dabec2c7a12958da4b7fee6551020249cdcb111ad6a1058352d2838742a
[root@docker2 ~]# curl 192.168.1.20
11
```

3）测试，docker1 创建和修改文件，docker2 查看

```
[root@docker1 ~]# touch /mnt/qq/a.sh
[root@docker1 ~]# echo 22 > /mnt/qq/index.html
[root@docker2 ~]#ls  /mnt/qq/
a.sh  index.html
[root@docker2 ~]# cat /mnt/qq/index.html
22
```

4. 案例 4：创建自定义网桥

- **问题**

本案例要求：

- 创建网桥设备 docker01
- 设定网段为 172.30.0.0/16
- 启动 nginx 容器，nginx 容器桥接 docker01 设备
- 映射真实机 8080 端口与容器的 80 端口

- **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

```
[root@docker1 ~]# docker  network  create  --subnet=172.30.0.0/16 docker01
c9cf26f911ef2dccb1fd1f670a6c51491e72b49133246f6428dd732c44109462
[root@docker1 ~]# docker  network  list
NETWORK ID          NAME                DRIVER              SCOPE
bc189673f959        bridge              bridge              local
6622752788ea        docker01            bridge              local
53bf43bdd584        host                host                local
ac52d3151ba8        none                null                local
[root@docker1 ~]# ip  a   s
[root@docker1 ~]# docker  network   inspect   docker01
[
    {
        "Name": "docker01",
        "Id": "c9cf26f911ef2dccb1fd1f670a6c51491e72b49133246f6428dd732c44109462",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.30.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

2）使用自定义网桥启动容器

```
[root@docker1 ~]# docker  run  --network=docker01   -id   nginx
```

3）端口映射

```
[root@docker1 ~]# docker  run  -p  8080:80  -id  nginx
e523b386f9d6194e53d0a5b6b8f5ab4984d062896bab10639e41aef657cb2a53
[root@docker1 ~]# curl 192.168.1.10:8080
```

步骤二：扩展实验

1）新建一个网络模型 docker02

```
 [root@docker1 ~]# docker  network   create   --driver  bridge  docker02
//新建一个 名为 docker02 的网络模型
5496835bd3f53ac220ce3d8be71ce6afc919674711ab3f94e6263b9492c7d2cc
[root@docker1 ~]# ifconfig
//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
br-5496835bd3f5: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.18.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        ether 02:42:89:6a:a2:72  txqueuelen 0  (Ethernet)
        RX packets 8  bytes 496 (496.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 496 (496.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@docker1 ~]# docker  network  list          //查看显示 docker02（查看加粗字样）
NETWORK ID          NAME                DRIVER              SCOPE
bc189673f959        bridge              bridge              local
5496835bd3f5        docker02             bridge              local
53bf43bdd584        host                host                local
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

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

3）查看 16dc92e55023 的信息，如图-3 所示：

```
[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

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

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