TTS 10.0 COOKBOOK

(NSD CLOUD DAY05)

版本编号 10.0

2018-08

达内 IT 培训集团

NSD CLOUD DAY05

1. 案例 1：安装 Docker

- **问题**

本案例要求配置 yum 源并安装 Docker：

- 安装 docker-engine 和 docker-engine-selinux
- 关闭防火墙

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：配置第三方 yum 源（真机操作）

```
    [root@zrj ~]# mkdir /var/ftp/docker
    [root@zrj ~]# mv docker-engine-* /var/ftp/docker
    [root@zrj ~]# ls /var/ftp/docker
    docker-engine-1.12.1-1.el7.centos.x86_64.rpm
docker-engine-selinux-1.12.1-1.el7.centos.noarch.rpm
    [root@zrj ~]# createrepo /var/ftp/docker/
    Spawning worker 0 with 1 pkgs
    Spawning worker 1 with 1 pkgs
    Spawning worker 2 with 0 pkgs
    Spawning worker 3 with 0 pkgs
    Spawning worker 4 with 0 pkgs
    Spawning worker 5 with 0 pkgs
    Workers Finished
    Saving Primary metadata
    Saving file lists metadata
    Saving other metadata
    Generating sqlite DBs
    Sqlite DBs complete
```

步骤二：配置 IP（虚拟机配置静态 ip）docker1 和 docker2 主机同样操作

```
    [root@localhost ~]# echo docker1 > /etc/hostname
    [root@localhost ~]# hostname docker1
    [root@localhost ~]# echo docker2 > /etc/hostname
    [root@localhost ~]# hostname docker2
    [root@docker1 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
    # Generated by dracut initrd
    DEVICE="eth0"
    ONBOOT="yes"
    IPV6INIT="no"
    IPV4_FAILURE_FATAL="no"
    NM_CONTROLLED="no"
    TYPE="Ethernet"
    BOOTPROTO="static"
    IPADDR="192.168.1.10"
    PREFIX=24
    GATEWAY=192.168.1.254
    [root@docker1 ~]# systemctl restart network
```

```
[root@docker2 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
# Generated by dracut initrd
DEVICE="eth0"
ONBOOT="yes"
IPV6INIT="no"
IPV4_FAILURE_FATAL="no"
NM_CONTROLLED="no"
TYPE="Ethernet"
BOOTPROTO="static"
IPADDR="192.168.1.20"
PREFIX=24
GATEWAY=192.168.1.254
[root@docker1 ~]# systemctl restart network
```

步骤三：配置 yum 客户端（docker1 和 docker2 主机同样操作）

```
[root@docker1 ~]# vim /etc/yum.repos.d/local.repo
[local_repo]
name=CentOS-$releasever - Base
baseurl="ftp://192.168.1.254/system"
enabled=1
gpgcheck=1

[loca]
name=local
baseurl="ftp://192.168.1.254/docker"
enabled=1
gpgcheck=0
```

步骤四：安装 docker（docker1 和 docker2 主机同样操作）

```
[root@docker1 ~]# yum -y install docker-engine
[root@docker1 ~]# systemctl restart docker
[root@docker1 ~]# systemctl enable docker
[root@docker1 ~]# ifconfig     //有 docker0 说明环境部署完成
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        ether 02:42:3e:e7:3f:6e  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
[root@docker2 ~]# docker version       //查看版本

[root@docker2 ~]# yum -y install docker-engine
[root@docker2 ~]# systemctl restart docker
[root@docker2 ~]# systemctl enable docker
[root@docker2 ~]# ifconfig     //有 docker0 说明环境部署完成
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        ether 02:42:53:82:b9:d4  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
[root@docker2 ~]# docker version       //查看版本
```

2. 案例 2：镜像基本操作

- **问题**

本案例要求熟悉镜像的基本操作：

- 导入镜像
- 导出镜像
- 启动镜像

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：docker 镜像

1）下载镜像

```
[root@docker1 ~]# docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
8c5a7da1afbc: Pull complete
Digest: sha256:cb63aa0641a885f54de20f61d152187419e8f6b159ed11a251a09d115fdff9bd
Status: Downloaded newer image for busybox:latest
```

2）上传镜像

```
[root@docker1 ~]# docker push busybox
```

3）查看镜像

```
[root@docker1 ~]# docker images
REPOSITORY       TAG             IMAGE ID         CREATED          SIZE
busybox          latest          e1ddd7948a1c      4 weeks ago      1.163 MB
```

4）查找 busybox 镜像

```
[root@docker1 ~]# docker search busybox
```

5）导出 busybox 镜像为 busybox.tar

```
[root@docker1 ~]# docker  save  busybox:latest >busybox.tar
[root@docker1 ~]# ls
busybox.tar
```

6）导入镜像

```
[root@docker1 ~]# scp  busybox.tar 192.168.1.20:/root
[root@docker2 ~]# ls
busybox.tar
[root@docker2 ~]# docker load <busybox.tar
f9d9e4e6e2f0: Loading layer [==================================================>]
1.378 MB/1.378 MB
Loaded image: busybox:latest[=>                                                  ]
32.77 kB/1.378 MB
[root@docker2 ~]# docker images
REPOSITORY         TAG             IMAGE ID          CREATED          SIZE
busybox            latest          e1ddd7948a1c       4 weeks ago      1.163 MB
```

7）删除镜像

```
[root@docker2 ~]# docker rmi  busybox
Untagged: busybox:latest
Deleted: sha256:e1ddd7948a1c31709a23cc5b7dfe96e55fc364f90e1cebcde0773a1b5a30dcda
Deleted: sha256:f9d9e4e6e2f0689cd752390e14ade48b0ec6f2a488a05af5ab2f9ccaf54c299d
```

步骤二：一次性导入多个镜像

```
[root@docker1 ~]# yum -y install unzip
[root@docker1 ~]# unzip docker_images.zip
Archive:  docker_images.zip
   creating: docker_images/
  inflating: docker_images/nginx.tar
  inflating: docker_images/redis.tar
  inflating: docker_images/centos.tar
  inflating: docker_images/registry.tar
  inflating: docker_images/ubuntu.tar
[root@docker1 ~]# ls
busybox.tar  docker_images  docker_images.zip  eip
[root@docker1 ~]# cd docker_images
[root@docker1 docker_images]# ls
centos.tar  nginx.tar  redis.tar  registry.tar  ubuntu.tar
[root@docker1 docker_images]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
busybox             latest              e1ddd7948a1c        4 weeks ago         1.163
MB
[root@docker1 docker_images]# for i in *; do docker load <$i; done
```

导入多个镜像如图-1 所示：

```
[root@docker1 docker_images]# for i in *; do docker load <$i; done
43e653f84b79: Loading layer [==================================================>] 207.2 MB/207.2 MB
Loaded image: centos:latest [>                                                  ] 557.1 kB/207.2 MB
142a601d9793: Loading layer [==================================================>] 128.9 MB/128.9 MB
40e298e9673a: Loading layer [==================================================>] 60.57 MB/60.57 MB
8d8bfe3cd5e4: Loading layer [==================================================>] 3.584 kB/3.584 kB
Loaded image: nginx:latestr [=======>                                           ]    512 B/3.584 kB
1cc8aacad4a1: Loading layer [==================================================>] 344.6 kB/344.6 kB
40ef78f2da08: Loading layer [==================================================>] 41.21 MB/41.21 MB
652c8a715c4f: Loading layer [==================================================>] 2.703 MB/2.703 MB
fa4e25f53e04: Loading layer [==================================================>] 16.46 MB/16.46 MB
c215f3ad270b: Loading layer [==================================================>] 1.536 kB/1.536 kB
644be81b61f9: Loading layer [==================================================>] 3.584 kB/3.584 kB
Loaded image: redis:latestr [=======>                                           ]    512 B/3.584 kB
e53f74215d12: Loading layer [==================================================>]  5.06 MB/5.06 MB
febf19f93653: Loading layer [==================================================>] 7.894 MB/7.894 MB
59e80739ed3f: Loading layer [==================================================>] 22.79 MB/22.79 MB
621c2399d41a: Loading layer [==================================================>] 3.584 kB/3.584 kB
9113493eaae1: Loading layer [==================================================>] 2.048 kB/2.048 kB
Loaded image: registry:latest============>                                     ]    512 B/2.048 kB
65bdd50ee76a: Loading layer [==================================================>] 82.09 MB/82.09 MB
ec75999a0cb1: Loading layer [==================================================>] 15.87 kB/15.87 kB
67885e448177: Loading layer [==================================================>] 8.192 kB/8.192 kB
8db5f072feec: Loading layer [==================================================>] 5.632 kB/5.632 kB
059ad60bcacf: Loading layer [==================================================>] 3.072 kB/3.072 kB
Loaded image: ubuntu:latest [========>                                          ]    512 B/3.072 kB
[root@docker1 docker_images]#
```

图-1

步骤三：启动镜像

1）启动 centos 镜像生成一个容器

备注： 启动镜像时若不知道后面的命令加什么：

1、可以猜（如：/bin/bash、/bin/sh）

2、可以不加后面的命令，默认启动

```
[root@docker1 docker_images]# docker run -it centos /bin/bash
[root@7a652fc72a9f /]# ls /
anaconda-post.log  bin  dev  etc  home  lib  lib64  media  mnt  opt  proc  root
run  sbin  srv  sys  tmp  usr  var
[root@7a652fc72a9f /]# cd /etc/yum.repos.d/
[root@7a652fc72a9f yum.repos.d]# ls
CentOS-Base.repo            CentOS-Debuginfo.repo            CentOS-Sources.repo
CentOS-fasttrack.repo
CentOS-CR.repo    CentOS-Media.repo      CentOS-Vault.repo
[root@7a652fc72a9f yum.repos.d]# rm -rf C*
[root@7a652fc72a9f yum.repos.d]# ls
[root@7a652fc72a9f yum.repos.d]#vi dvd.repo    //在容器里面配置一个 yum 源
[local]
name=local
baseurl=ftp://192.168.1.254/system
enable=1
gpgcheck=0
[root@7a652fc72a9f yum.repos.d]# yum -y install net-tools  //安装软件
[root@7a652fc72a9f yum.repos.d]# exit
exit
```

3. 案例 3：镜像与容器常用指令

- **问题**

本案例要求掌握镜像与容器的常用命令：

- 镜像常用指令练习
- 容器常用指令练习

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：镜像常用命令

1）查看后台运行的容器

备注：docker ps

```
[root@docker1 ~]# docker run -d nginx  //启动 nginx 的镜像
[root@docker1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS               NAMES
56ec8154f8e0        nginx:latest        "nginx -g 'daemon off"   17 minutes ago
Up 12 minutes       80/tcp, 443/tcp     zen_darwin
```

2）只显示容器 ID

备注：docker ps -q

```
[root@docker1 docker_images]# docker ps -q
56ec8154f8e0
85c6b0b62235
f7ee40a87af5
```

3）显示所有的容器，包括没有启动的

备注：docker ps -a

```
[root@docker1 docker_images]# docker ps -a
```

4）显示所有的容器 ID

备注：docker ps -qa

```
[root@docker1 docker_images]# docker ps -qa
56ec8154f8e0
2b68c3960737
85c6b0b62235
f7ee40a87af5
b261be571648
fb2fb8c3d7a8
```

5）查看 centos 镜像历史（制作过程），如图-2 所示：

备注：docker history centos

```
[root@docker1 docker_images]# docker history nginx
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
a5311a310510        23 months ago       /bin/sh -c #(nop)  CMD ["nginx" "-g" "daemon    0 B
<missing>           23 months ago       /bin/sh -c #(nop)  EXPOSE 443/tcp 80/tcp        0 B
<missing>           23 months ago       /bin/sh -c ln -sf /dev/stdout /var/log/nginx/   0 B
<missing>           23 months ago       /bin/sh -c apt-key adv --keyserver hkp://pgp.   58.43 MB
<missing>           23 months ago       /bin/sh -c #(nop)  ENV NGINX_VERSION=1.11.5-1   0 B
<missing>           23 months ago       /bin/sh -c #(nop)  MAINTAINER NGINX Docker Ma   0 B
<missing>           23 months ago       /bin/sh -c #(nop)  CMD ["/bin/bash"]            0 B
<missing>           23 months ago       /bin/sh -c #(nop) ADD file:c6c23585ab140b0b32   123 MB
[root@docker1 docker_images]#
```

图-2

7）删除镜像，启动容器时删除镜像会失败，先删除容器，再删除镜像

备注：docker rmi 镜像名

```
    [root@docker1 docker_images]# docker rmi nginx
    Error response from daemon: conflict: unable to remove repository reference "nginx"
(must force) - container 4f83871aa42e is using its referenced image a5311a310510
    [root@docker1 docker_images]# docker stop 4f
    4f
    [root@docker1 docker_images]# docker rm 4f
    4f
    [root@docker1 docker_images]# docker rmi nginx
    Untagged: nginx:latest
    Deleted: sha256:d1fd7d86a8257f3404f92c4474fb3353076883062d64a09232d95d940627459d
    Deleted: sha256:4d765aea84ce4f56bd623e4fd38dec996a259af3418e2466d0e2067ed0ae8aa6
    Deleted: sha256:5d385be69c9c4ce5538e12e6e677727ebf19ca0afaff6f035d8043b5e413003a
    Deleted: sha256:adb712878b60bd7ed8ce661c91eb3ac30f41b67bfafed321395863051596a8e9
    Deleted: sha256:55a50a618c1b76f784b0b68a0b3d70db93b353fb03227ea6bd87f794cad92917
    Deleted: sha256:e53f74215d12318372e4412d0f0eb3908e17db25c6185f670db49aef5271f91f
```

8）修改镜像的名称和标签，默认标签为 latest

备注：docker tag

```
    [root@docker1 docker_images]# docker tag  centos:latest cen:v1
```

9）查看镜像的底层信息，如图-3 所示：

```
    [root@docker1 docker_images]# docker inspect centos
```

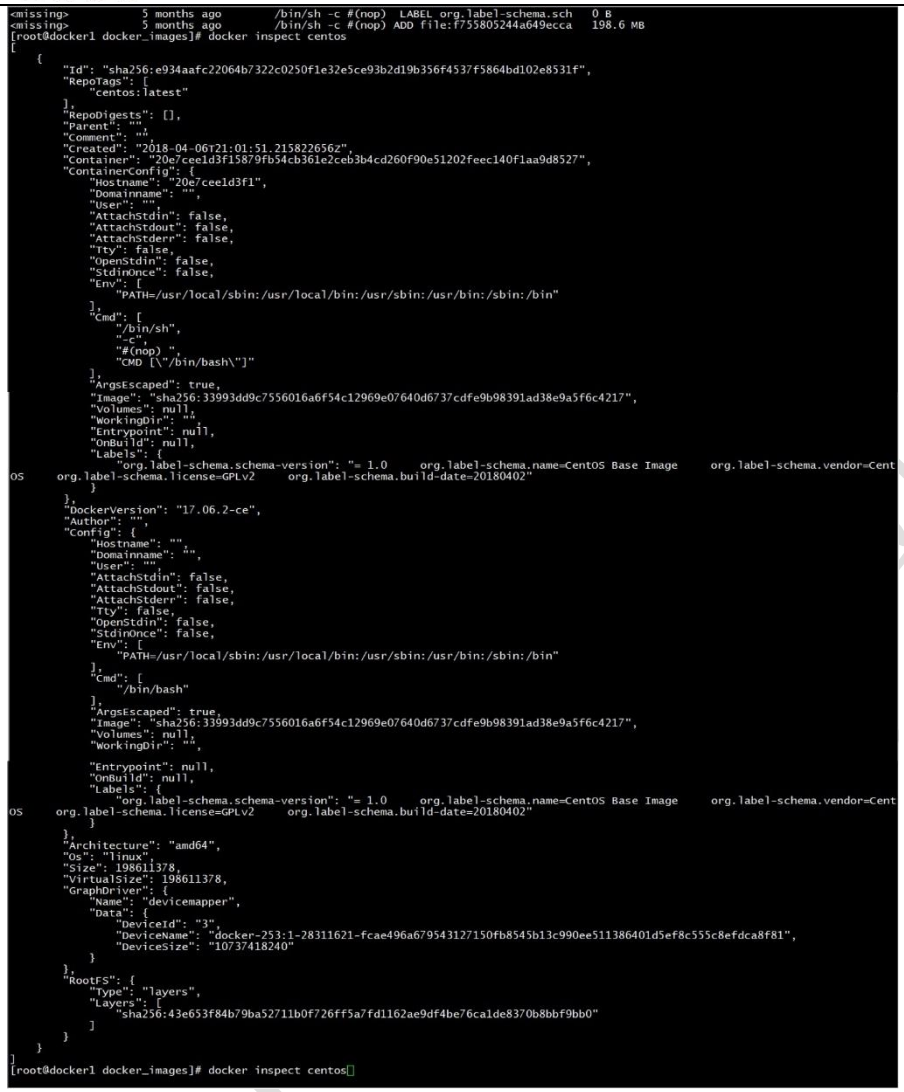

图-3

10）修改镜像的标签

```
[root@docker1 docker_images]# docker tag  centos:latest cen:v1
[root@docker1 docker_images]# docker images
REPOSITORY   TAG            IMAGE ID         CREATED          SIZE
    cen        v1             e934aafc2206   5 months ago    198.6 MB
[root@docker1 docker_images]# docker rmi centos    //删除 centos
[root@localhost ~]# docker run -it centos   //启动的时候，因为是用标签标签启动的，所以会重新通过 ID 下载
[root@localhost ~]# docker run -it centos
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
Digest: sha256:989b936d56b1ace20ddf855a301741e52abca38286382cba7f44443210e96d16
Status: Downloaded newer image for centos:latest
[root@localhost ~]# docker run -it cen:v1   //通过新建的标签启动
```

步骤二：容器命令

1）关闭容器

备注：docker stop 容器 ID

```
[root@docker1 docker_images]# docker stop 0f
0f
```

2）启动容器

备注：docker start 容器 ID

```
[root@docker1 docker_images]# docker start 0f
0f
```

3）重启容器

备注：docker restart 容器 ID

```
[root@docker1 docker_images]# docker restart 0f
0f
```

4）删除容器

备注：docker rm （容器 id） //运行中删除不掉，先关闭容器

```
[root@docker1 docker_images]# docker rm 0f
Error response from daemon: You cannot remove a running container 0f63706692e15134a8f07655a992771b312b8eb01554fc37e1a39b03b28dd05c. Stop the container before attempting removal or use -f
[root@docker1 docker_images]# docker stop 0f
0f
[root@docker1 docker_images]# docker rm 0f
0f
[root@docker1 docker_images]#
```

5）连接容器 attach|exec

```
[root@docker1 docker_images]# docker attach  0f
[root@docker1 docker_images]# docker ps    //容器关闭
CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES
[root@docker1 docker_images]#  docker exec -it 0f /bin/bash
[root@docker1 docker_images]# docker ps        //容器不会关闭
CONTAINER ID  IMAGE         COMMAND        CREATED          STATUS          PORTS
NAMES
0b3c50284a1c   centos:v1        "/bin/bash"    15 minutes ago    Up 15 minutes
tiny_lamarr

[root@docker1 docker_images]# docker top f7     //查看容器进程列表
[root@localhost ~]# docker run -itd centos:latest
[root@0b3c50284a1c /]# ps
  PID TTY          TIME CMD
    1 ?        00:00:00 bash
   13 ?        00:00:00 ps
 [root@docker1 docker_images]# docker exec -it 85 /bin/bash
root@85c6b0b62235:/# sleep 50 &
[1] 9
root@85c6b0b62235:/# exit
exit


[root@docker1 docker_images]#docker top 85

UID    PID  PPID C   STIME     TTY      TIME     CMD
root  2744 2729  0   18:01    pts/4  00:00:00   /bin/bash
```

6）过滤查看 mac 和 ip 地址

```
[root@docker1 docker_images]# docker inspect -f '{{.NetworkSettings.MacAddress}}' 4f
02:42:ac:11:00:03
[root@docker1 docker_images]# docker inspect -f '{{.NetworkSettings.IPAddress}}' 4f
172.17.0.3
```

7）修改 nginx 的显示内容

```
[root@docker1 docker_images]# docker run -it nginx:latest
```

```
[root@docker1 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
56ec8154f8e0        nginx:latest        "nginx -g 'daemon off"   5 minutes ago       Up 18 seconds       80/tcp, 443/tcp     zen_darwi
85c6b0b62235        nginx:latest        "/bin/bash"              14 hours ago        Up 14 hours         80/tcp, 443/tcp     desperate
4f83871aa42e        nginx:latest        "/bin/bash"              14 hours ago        Up 14 hours         80/tcp, 443/tcp     backstabb
f7ee40a87af5        centos:latest       "/bin/bash"              14 hours ago        Up 14 hours                             goofy_cra
[root@docker1 ~]# docker exec -it 56  /bin/bash
```

```
[root@docker1 docker_images]# docker exec -it 56 /bin/bash
root@56ec8154f8e0:/# nginx -T /usr/share/nginx/html/
nginx: invalid option: "/usr/share/nginx/html/"   //查找并显示结果
root@56ec8154f8e0:/# echo  aaa > /usr/share/nginx/html/index.html  //修改主页显示的内容
root@56ec8154f8e0:/# nginx -T
root@56ec8154f8e0:/# cat /usr/share/nginx/html/index.html
aaa
```

8）过滤查看 nginx 的 ip 地址

```
[root@docker1 ~]# docker inspect -f '{{.NetworkSettings.IPAddress}}' 56
172.17.0.5
[root@docker1 ~]# curl   172.17.0.5
aaa
```