

# NSD CLOUD DAY05

1. [案例1：安装Docker](#)
2. [案例2：镜像基本操作](#)
3. [案例3：镜像与容器常用指令](#)

## 1 案例1：安装Docker

### 1.1 问题

本案例要求配置yum源并安装Docker：

- 准备两台虚拟机，IP为192.168.1.10和192.168.1.20
- 安装docker-engine 和 docker-engine-selinux
- 关闭防火墙

### 1.2 步骤

实现此案例需要按照如下步骤进行。

#### 步骤一：配置yum源

1) 配置第三方yum源（真机操作）

```
01. [root@room9pc01 ~]# mkdir /var/ftp/docker
02. [root@room9pc01 ~]# mv docker-engine-*/var/ftp/docker
03. [root@room9pc01 ~]# ls /var/ftp/docker
04. docker-engine-1.12.1-1.el7.centos.x86_64.rpm
05. docker-engine-selinux-1.12.1-1.el7.centos.noarch.rpm
06. [root@room9pc01 ~]# createrepo /var/ftp/docker/
07. Spawning worker 0 with 1 pkgs
08. Spawning worker 1 with 1 pkgs
09. Spawning worker 2 with 0 pkgs
10. Spawning worker 3 with 0 pkgs
11. Spawning worker 4 with 0 pkgs
12. Spawning worker 5 with 0 pkgs
13. Workers Finished
14. Saving Primary metadata
15. Saving file lists metadata
16. Saving other metadata
17. Generating sqlite DBs
18. Sqlite DBs complete
```

[Top](#)

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

```

01. [ root@localhost ~] # echo docker1 > /etc/hostname
02. [ root@localhost ~] # hostname docker1
03. [ root@localhost ~] # echo docker2 > /etc/hostname
04. [ root@localhost ~] # hostname docker2
05. [ root@docker1 ~] # vim /etc/sysconfig/network-scripts/ifcfg-eth0
06. # Generated by dracut initrd
07. DEVICE="eth0"
08. ONBOOT="yes"
09. IPV6INIT="no"
10. IPV4_FAILURE_FATAL="no"
11. NM_CONTROLLED="no"
12. TYPE="Ethernet"
13. BOOTPROTO="static"
14. IPADDR="192.168.1.10"
15. PREFIX=24
16. GATEWAY=192.168.1.254
17. [ root@docker1 ~] # systemctl restart network
18.
19. [ root@docker2 ~] # vim /etc/sysconfig/network-scripts/ifcfg-eth0
20. # Generated by dracut initrd
21. DEVICE="eth0"
22. ONBOOT="yes"
23. IPV6INIT="no"
24. IPV4_FAILURE_FATAL="no"
25. NM_CONTROLLED="no"
26. TYPE="Ethernet"
27. BOOTPROTO="static"
28. IPADDR="192.168.1.20"
29. PREFIX=24
30. GATEWAY=192.168.1.254
31. [ root@docker1 ~] # systemctl restart network

```

### 3) 配置yum客户端 ( docker1和docker2主机同样操作 )

```

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

```

[Top](#)

```

06.  gpgcheck=1
07.
08.  [ loca]
09.  name=local
10.  baseurl="ftp://192.168.1.254/docker"
11.  enabled=1
12.  gpgcheck=0
13.
14.  [ root@docker2 ~] # vim /etc/yum.repos.d/local.repo
15.  [ local_repo]
16.  name=CentOS-$releasever - Base
17.  baseurl="ftp://192.168.1.254/system"
18.  enabled=1
19.  gpgcheck=1
20.
21.  [ loca]
22.  name=local
23.  baseurl="ftp://192.168.1.254/docker"
24.  enabled=1
25.  gpgcheck=0

```

#### 4) 安装docker ( docker1和docker2主机同样操作 )

```

01.  [ root@docker1 ~] # yum -y install docker-engine
02.  [ root@docker1 ~] # systemctl restart docker
03.  [ root@docker1 ~] # systemctl enable docker
04.  [ root@docker1 ~] # ifconfig    //有docker0说明环境部署完成
05.  docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
06.      inet 172.17.0.1 netmask 255.255.0.0 broadcast 0.0.0.0
07.      ether 02:42:3e:e7:3f:6e txqueuelen 0 (Ethernet)
08.      RX packets 0 bytes 0 (0.0 B)
09.      RX errors 0 dropped 0 overruns 0 frame 0
10.      TX packets 0 bytes 0 (0.0 B)
11.      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
12.  [ root@docker2 ~] # docker version    //查看版本
13.
14.  [ root@docker2 ~] # yum -y install docker-engine
15.  [ root@docker2 ~] # systemctl restart docker
16.  [ root@docker2 ~] # systemctl enable docker
17.  [ root@docker2 ~] # ifconfig    //有docker0说明环境部署完成

```

[Top](#)

```

18. docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
19.         inet 172.17.0.1 netmask 255.255.0.0 broadcast 0.0.0.0
20.         ether 02:42:53:82:b9:d4 txqueuelen 0 (Ethernet)
21.         RX packets 0 bytes 0 (0.0 B)
22.         RX errors 0 dropped 0 overruns 0 frame 0
23.         TX packets 0 bytes 0 (0.0 B)
24.         TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
25. [root@docker2 ~]# docker version //查看版本

```

## 2 案例2：镜像基本操作

### 2.1 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

### 2.2 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

```

01. [root@docker1 ~]# docker pull busy box
02. Using default tag: latest
03. latest: Pulling from library /busy box
04. 8c5a7da1afbc: Pull complete
05. Digest: sha256:cb63aa0641a885f54de20f61d152187419e8f6b159ed11a251a09d115fddf9bd
06. Status: Downloaded newer image for busy box: latest

```

##### 2) 上传镜像

```

01. [root@docker1 ~]# docker push busy box

```

##### 3) 查看镜像

```

01. [root@docker1 ~]# docker images
02. REPOSITORY      TAG          IMAGE ID      CREATED      SIZE

```

[Top](#)

03. busy box latest e1ddd7948a1c 4 weeks ago 1.163 MB

#### 4) 查找busybox镜像

01. [root@docker1 ~]# docker search busy box

#### 5) 导出busybox镜像为busybox.tar

01. [root@docker1 ~]# docker save busy box: latest >busy box.tar  
 02. [root@docker1 ~]# ls  
 03. busy box.tar

#### 6) 导入镜像

01. [root@docker1 ~]# scp busy box.tar 192.168.1.20:/root  
 02. [root@docker2 ~]# ls  
 03. busy box.tar  
 04. [root@docker2 ~]# docker load <busy box.tar  
 05. f9d9e4e6e2f0: Loading layer [=====]  
 06. Loaded image: busy box: latest[=>] 32.77 kB/1.378 MB  
 07. [root@docker2 ~]# docker images  
 08. REPOSITORY TAG IMAGE ID CREATED SIZE  
 09. busy box latest e1ddd7948a1c 4 weeks ago 1.163 MB

#### 7) 删除镜像

01. [root@docker2 ~]# docker rmi busy box  
 02. Untagged: busy box: latest  
 03. Deleted: sha256:e1ddd7948a1c31709a23cc5b7dfe96e55fc364f90e1cebcde0773a1b5a30dcdk  
 04. Deleted: sha256:f9d9e4e6e2f0689cd752390e14ade48b0ec6f2a488a05af5ab2f9ccaf54c299

#### 步骤二：一次性导入多个镜像

[Top](#)

01. [root@docker1 ~]# yum -y install unzip

```

02. [root@docker1 ~]# unzip docker_images.zip
03. Archive:  docker_images.zip
04.   creating:  docker_images/
05.   inflating: docker_images/nginx.tar
06.   inflating: docker_images/redis.tar
07.   inflating: docker_images/centos.tar
08.   inflating: docker_images/registry.tar
09.   inflating: docker_images/ubuntu.tar
10. [root@docker1 ~]# ls
11. busybox.tar  docker_images  docker_images.zip  eip
12. [root@docker1 ~]# cd docker_images
13. [root@docker1 docker_images]# ls
14. centos.tar  nginx.tar  redis.tar  registry.tar  ubuntu.tar
15. [root@docker1 docker_images]# docker images
16. REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
17. busybox              latest             e1ddd7948a1c       4 weeks ago        1.163 MB
18. [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:latest [=====>] 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:latest [=====>] 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

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

[Top](#)

```
01. [root@docker1 docker_images]# docker run -it centos /bin/bash
```

```

02. [root@7a652fc72a9f /] # ls /
03. anaconda post.log bin dev etc home lib lib64 media mnt opt proc root run sbin sr
04. [root@7a652fc72a9f /] # cd /etc/yum.repos.d/
05. [root@7a652fc72a9f yum.repos.d] # ls
06. CentOS-Base.repo CentOS-Debuginfo.repo CentOS-Sources.repo CentOS-fasttrack.repo
07. CentOS-CR.repo CentOS-Media.repo CentOS-Vault.repo
08. [root@7a652fc72a9f yum.repos.d] # rm -rf C*
09. [root@7a652fc72a9f yum.repos.d] # ls
10. [root@7a652fc72a9f yum.repos.d] # vi dvd.repo //在容器里面配置一个yum源
11. [local]
12. name=local
13. baseurl=ftp://192.168.1.254/system
14. enable=1
15. gpgcheck=0
16. [root@7a652fc72a9f yum.repos.d] # yum -y install net-tools //安装软件
17. [root@7a652fc72a9f yum.repos.d] # exit
18. exit

```

### 3 案例3：镜像与容器常用指令

#### 3.1 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

#### 3.2 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

```

01. [root@docker1 ~] # docker run -d nginx //启动nginx的镜像
02. [root@docker1 ~] # docker ps //查看后台运行的容器
03. CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS
04. 56ec8154f8e0    nginx:latest    "nginx -g 'daemon off'"    17 minutes ago    Up 12 min

```

##### 2) 只显示容器ID

[Top](#)

```

01. [ root@docker1 docker_images] # docker ps - q
02. 56ec8154f8e0
03. 85c6b0b62235
04. f7ee40a87af5

```

### 3 ) 显示所有的容器,包括没有启动的

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

### 4 ) 显示所有的容器ID

```

01. [ root@docker1 docker_images] # docker ps - qa
02. 56ec8154f8e0
03. 2b68c3960737
04. 85c6b0b62235
05. f7ee40a87af5
06. b261be571648
07. fb2fb8c3d7a8

```

### 5 ) 查看centos镜像历史 ( 制作过程 ) , 如图-2所示 :

```
01. [ root@docker1 docker_images] # docker history centos
```

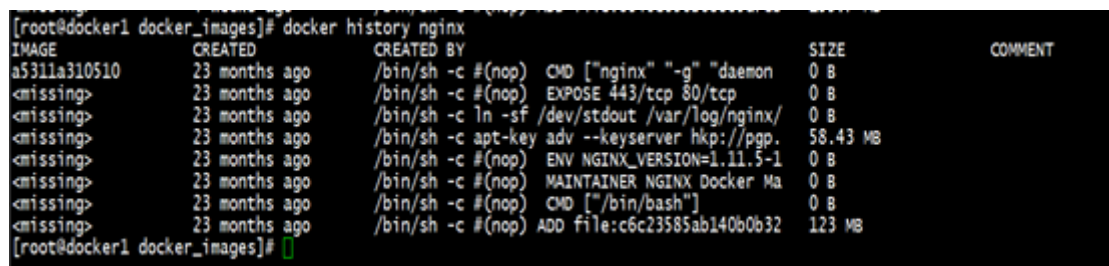


IMAGE	CREATED	CREATED BY	SIZE	COMMENT
a5311a310510	23 months ago	/bin/sh -c #(nop) CMD ["nginx" "-g" "daemon"]	0 B	
<missing>	23 months ago	/bin/sh -c #(nop) EXPOSE 443/tcp 80/tcp	0 B	
<missing>	23 months ago	/bin/sh -c ln -sf /dev/stdout /var/log/nginx/	0 B	
<missing>	23 months ago	/bin/sh -c apt-key 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	

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

格式 : docker rmi 镜像名

```

01. [ root@docker1 docker_images] # docker rmi nginx //nginx为镜像名
02.

```

[Top](#)



```
03. Error response from daemon: conflict: unable to remove repository reference "nginx" ( m
04. [ root@docker1 docker_images] # docker stop 4f
05. 4f
06. [ root@docker1 docker_images] # docker rm 4f
07. 4f
08. [ root@docker1 docker_images] # docker rmi nginx //成功删除
09. Untagged: nginx:latest
10. Deleted: sha256:d1fd7d86a8257f3404f92c4474fb3353076883062d64a09232d95d94062745
11. Deleted: sha256:4d765aea84ce4f56bd623e4fd38dec996a259af3418e2466d0e2067ed0ae8a
12. Deleted: sha256:5d385be69c9c4ce5538e12e6e677727ebf19ca0afaf6f035d8043b5e413003
13. Deleted: sha256:adb712878b60bd7ed8ce661c91eb3ac30f41b67bfafed321395863051596a8e
14. Deleted: sha256:55a50a618c1b76f784b0b68a0b3d70db93b353fb03227ea6bd87f794cad929
15. Deleted: sha256:e53f74215d12318372e4412d0f0eb3908e17db25c6185f670db49aef5271f911
```

## 8) 修改镜像的名称和标签,默认标签为latest

```
01. [ 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:

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

[Top](#)

```

missing: 5 months ago /bin/sh -c #(nop) LABEL org.label-schema.sch 0 B
missing: 5 months ago /bin/sh -c #(nop) ADD file:755805244a649ecca 198.6 MB
[root@docker1 docker_images]# docker inspect centos
[
  {
    "Id": "sha256:e934aafc22064b322c0250f1e3e3ce93b2d19b356f4537f3864bd102e8531f",
    "RepoTags": [
      "centos:latest"
    ],
    "RepoDigests": [
      ""
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2018-04-06T21:01:51.215822656Z",
    "Container": "20e7ce1d3f15d79fb34cb361e2ceb3b4cd260f90e51202fec140f1aa9d8527",
    "ContainerConfig": {
      "Hostname": "20e7ce1d3f1",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      "Cmd": [
        "/bin/sh",
        "-c",
        "#(nop) \nCMD [\"/bin/bash\"]"
      ],
      "ArgsEscaped": true,
      "Image": "sha256:33993dd9c7556016a6f54c12909e07640d6737cdf9e98391ad38e9a5f6c4217",
      "Volumes": null,
      "WorkingDir": "",
      "Entrypoint": null,
      "OnBuild": null,
      "Labels": {
        "org.label-schema.schema-version": "~ 1.0",
        "org.label-schema.name": "CentOS Base Image",
        "org.label-schema.vendor": "CentOS",
        "org.label-schema.license": "GPLv2",
        "org.label-schema.build-date": "20180402"
      }
    },
    "DockerVersion": "17.06.2-ce",
    "Author": "",
    "Config": {
      "Hostname": "",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      "Cmd": [
        "/bin/bash"
      ],
      "ArgsEscaped": true,
      "Image": "sha256:33993dd9c7556016a6f54c12909e07640d6737cdf9e98391ad38e9a5f6c4217",
      "Volumes": null,
      "WorkingDir": "",
      "Entrypoint": null,
      "OnBuild": null,
      "Labels": {
        "org.label-schema.schema-version": "~ 1.0",
        "org.label-schema.name": "CentOS Base Image",
        "org.label-schema.vendor": "CentOS",
        "org.label-schema.license": "GPLv2",
        "org.label-schema.build-date": "20180402"
      }
    },
    "Architecture": "amd64",
    "Os": "linux",
    "Size": 198611378,
    "VirtualSize": 198611378,
    "GraphDriver": {
      "Name": "devicemapper",
      "Data": {
        "DeviceId": "3",
        "DeviceName": "docker-253:1-28311621-fcae496a67954127150fb8545b13c990ee511386401d5ef8c555c8efdc8f81",
        "DeviceSize": "10737418240"
      }
    },
    "RootFS": {
      "Type": "layers",
      "Layers": [
        "sha256:43e653f84b79ba52711b0f726ff3a7fd1102ae9df4be76ca1de8370b8b6f9660"
      ]
    }
  }
]
[root@docker1 docker_images]# docker inspect centos

```

图-3

## 10 ) 修改镜像的标签

01. [ root@docker1 docker\_images] # docker tag centos:latest cen:v1
02. [ root@docker1 docker\_images] # docker images
03. REPOSITORY TAG IMAGE ID CREATED SIZE
04. cen v1 e934aafc2206 5 months ago 198.6 MB
05. [ root@docker1 docker\_images] # docker rmi centos //删除centos
06. [ root@localhost ~] # docker run -it centos
07. //启动的时候，因为是用标签启动的，所以会重新通过ID下载
08. [ root@localhost ~] # docker run -it centos
09. Unable to find image 'centos:latest' locally
10. latest: Pulling from library/centos
11. Digest: sha256:989b936d56b1ace20ddf855a301741e52abca38286382cba7f44443210e96d16
12. Status: Downloaded newer image for centos:latest
13. [ root@localhost ~] # docker run -it cen:v1 //通过新建的标签启动cen:v1

[Top](#)

## 步骤二：容器命令

## 1) 关闭容器

命令：docker stop 容器ID

```
01. [root@docker1 docker_images] # docker stop Of //Of为容器ID
02. Of
```

## 2) 启动容器

```
01. [root@docker1 docker_images] # docker start Of
02. Of
```

## 3) 重启容器

```
01. [root@docker1 docker_images] # docker restart Of
02. Of
```

## 4) 删除容器

运行中删除不掉，先关闭容器

```
01. [root@docker1 docker_images] # docker rm Of //删除失败
02. Error response from daemon: You cannot remove a running container Of 63706692e15134a
03. [root@docker1 docker_images] # docker stop Of //关闭容器
04. Of
05. [root@docker1 docker_images] # docker rm Of //删除成功
06. Of
07. [root@docker1 docker_images] #
```

## 5) 连接容器attach|exec

```
01. [root@docker1 docker_images] # docker attach Of
02. [root@docker1 docker_images] # docker ps //容器关闭
03. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
04. [root@docker1 docker_images] # docker exec -it Of /bin/bash
05. [root@docker1 docker_images] # docker ps //容器不会关闭
06. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

[Top](#)

```

07. 0b3c50284a1c centos:v1  "/bin/bash" 15 minutes ago Up 15 minutes tiny_lan
08.
09. [ root@docker1 docker_images] # docker top f7 //查看容器进程列表
10. [ root@localhost ~] # docker run - itd centos:latest
11. [ root@0b3c50284a1c /] # ps
12. PID TTY          TIME CMD
13.   1 ?        00:00:00 bash
14.  13 ?        00:00:00 ps
15. [ root@docker1 docker_images] # docker exec - it 85 /bin/bash
16. root@85c6b0b62235: /# sleep 50 &
17. [ 1] 9
18. root@85c6b0b62235: /# exit
19. exit
20.
21.
22. [ root@docker1 docker_images] #docker top 85
23.
24. UID PID PPID C STIME TTY          TIME CMD
25.  root 2744 2729 0 18:01 pts/4 00:00:00 /bin/bash

```

## 6 ) 过滤查看mac和ip地址

```

01. [ root@docker1 docker_images] # docker inspect - f '{{ .NetworkSettings.MacAddress }}' 4f
02. 02:42:ac:11:00:03
03. [ root@docker1 docker_images] # docker inspect - f '{{ .NetworkSettings.IPAddress }}' 4f
04. 172.17.0.3

```

## 7 ) 修改nginx的显示内容

```

01. [ root@docker1 docker_images] # docker run - it nginx:latest
02.

```

[Top](#)

```
[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
```

01. [ root@docker1 docker\_images] # docker exec - it 56 /bin/bash
02. root@56ec8154f8e0: /# nginx - T /usr/share/nginx/html/
03. nginx: invalid option: "/usr/share/nginx/html/" //查找并显示结果
04. root@56ec8154f8e0: /# echo aaa > /usr/share/nginx/html/index.html
05. //修改主页显示的内容
06. root@56ec8154f8e0: /# nginx - T
07. root@56ec8154f8e0: /# cat /usr/share/nginx/html/index.html
08. aaa

## 8 ) 过滤查看nginx的ip地址

01. [ root@docker1 ~] # docker inspect - f '{{ .NetworkSettings.IPAddress }}' 56
02. 172.17.0.5
03. [ root@docker1 ~] # curl 172.17.0.5
04. aaa

[Top](#)