目录

[Docker应用 2](#_Toc452370889)

[一 Docker介绍 2](#_Toc452370890)

[1.1 什么是Docker 2](#_Toc452370891)

[1.2 Docker核心理念（官方） 2](#_Toc452370892)

[1.3 Docker结构 3](#_Toc452370893)

[1.4 Docker与OpenStack 4](#_Toc452370894)

[1.5 Docker能干什么 4](#_Toc452370895)

[二 Docker安装 5](#_Toc452370896)

[2.1 安装 5](#_Toc452370897)

[三 Docker镜像和容器 5](#_Toc452370898)

[3.1 导出镜像 6](#_Toc452370899)

[3.2导入镜像 6](#_Toc452370900)

[3.3查看镜像 6](#_Toc452370901)

[3.4删除镜像 6](#_Toc452370902)

[3.5创建查看容器 6](#_Toc452370903)

[3.6创建容器并且起名进入容器 6](#_Toc452370904)

[3.7启动容器 7](#_Toc452370905)

[3.8进入容器 7](#_Toc452370906)

[3.9nsenter进入容器 通过pid退出容器还启动 7](#_Toc452370907)

[3.9.1通过pid进入容器 7](#_Toc452370908)

[3.9.2 编写脚本 7](#_Toc452370909)

[3.9.3删除容器 8](#_Toc452370910)

[3.9.4删除当前正在运行的容器 8](#_Toc452370911)

[四 Docker网络和存储 8](#_Toc452370912)

[4.1查看iptables和网络映射 8](#_Toc452370913)

[4.2映射nginx 10](#_Toc452370914)

[4.3查看容器日志nginx 10](#_Toc452370915)

[4.4指点端口映射 10](#_Toc452370916)

[4.5数据卷应用 10](#_Toc452370917)

[4.6查看容器卷信息 10](#_Toc452370918)

[4.7指定目录挂载数据卷 11](#_Toc452370919)

[4.8指定目录挂载数据卷指定权限 11](#_Toc452370920)

[4.9让一个容器去访问另一个容器的卷 11](#_Toc452370921)

[4.9.1容器转换 11](#_Toc452370922)

[五 手动构建镜像： 12](#_Toc452370923)

[5.1基于容器做镜像 12](#_Toc452370924)

[5.2制作镜像文件 12](#_Toc452370925)

[5.3制作过程截图 13](#_Toc452370926)

[5.4服务开机自启动 13](#_Toc452370927)

# Docker应用

# 一 Docker介绍

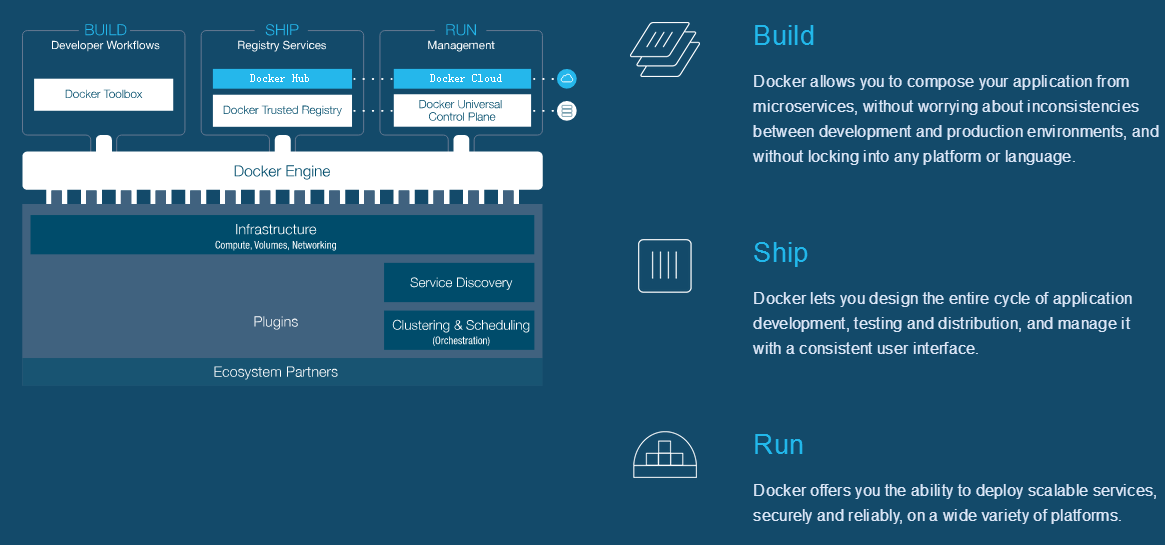
## 1.1 什么是Docker

Dockr是Docker.lnc公司开源的一个基于LXC技术之上构建的container容器引擎，源代码托管在GitHub上，基于Go语音并遵从Apache2.0协议开源，Docker是通过内核虚拟化技术（namespaces及cgroups等）来提供容器的资源隔离与安全保障等，由于docker通过操作系统层的虚拟化实现隔离，所以Docker容器在运行时，不需要类似虚拟机（vm）额外的操作系统开销，提高资源利用。

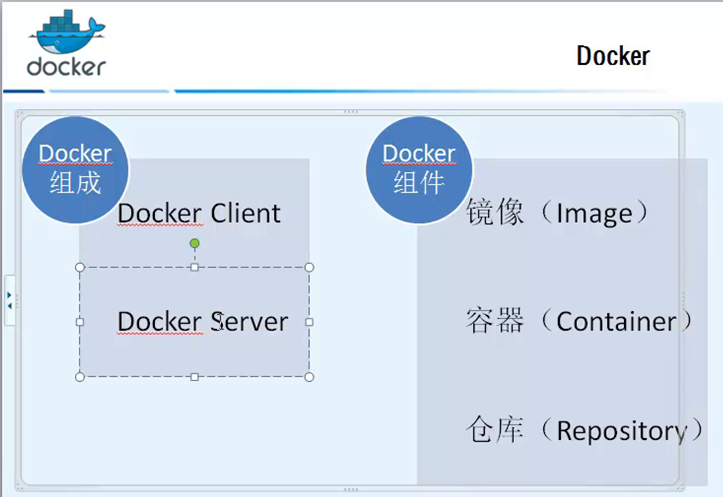
## 1.2 Docker核心理念（官方）

[http://www.docker.com/ #](http://www.docker.com/%20%20%20%20#)官网

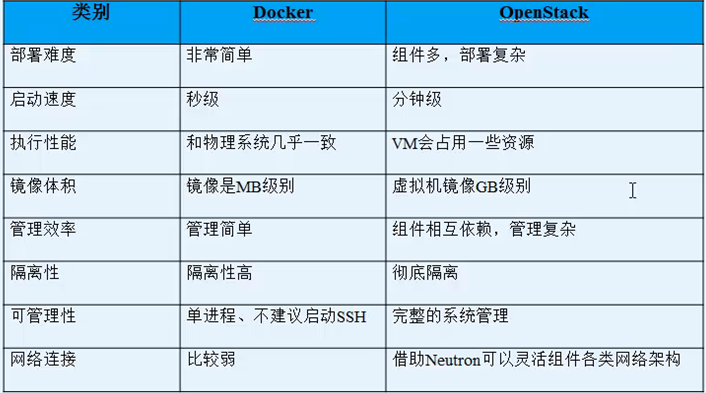
-------构建---------运输--------运行



## 1.3 Docker结构



## 1.4 Docker与OpenStack



## 1.5 Docker能干什么

1.简化配置

2.代码流水线管理

3.开发效率

4.应用隔离

5.整合服务器

6.debugging调试能力

7.多租户

8.快速部署

# 二 Docker安装

## 2.1 安装

centos6.x

yum install -y epel-release

yum install -y docker-io

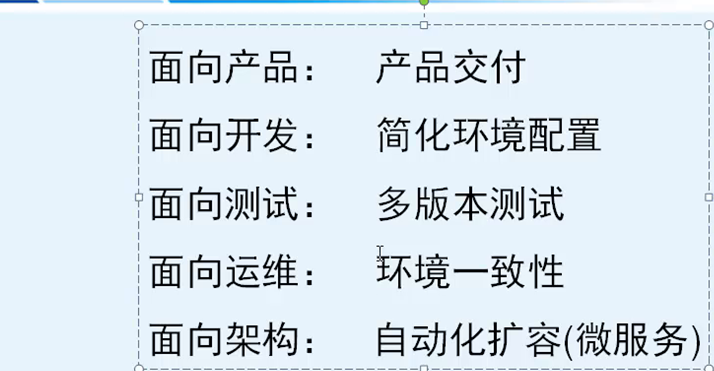
centos7.x

yum install –y docker

启动

systemctl start docker

# 三 Docker镜像和容器



## 3.1 导出镜像

docker save centos >/opt/centos.tar.gz

## 3.2导入镜像

docker load < /opt/centos.tar.gz

## 3.3查看镜像

docker images

## 3.4删除镜像

docker rmi eeb3a076a0be

## 3.5创建查看容器

docker run centos /bin/echo "hehe"

docker ps –a

## 3.6创建容器并且起名进入容器

docker run --name mdocekr -t -i centos /bin/bash

[root@947e07df3cd1 /]#

查看进程

ps aux

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.0 11748 1916 ? Ss 07:01 0:00 /bin/bash

root 16 0.0 0.0 47420 1656 ? R+ 07:03 0:00 ps aux

## 3.7启动容器

docker start 947e07df3cd1

## 3.8进入容器

docker attach 947e07df3cd1 #退出容器关闭 ctrl p+q不会关闭容器

## 3.9nsenter进入容器 通过pid退出容器还启动

docker inspect --format "{{.State.Pid}}" 947e07df3cd1

30623 #获取pid号

yum install util-linux #安装nsenter命令

## 3.9.1通过pid进入容器

nsenter -t 30623 -u -i -n –p

## 3.9.2 编写脚本

cat ns.sh

#!/bin/bash

PID=$(docker inspect --format "{{.State.Pid}}" $1)

nsenter -t $PID -u -i -n –p

./ns.sh 947e07df3cd1

## 3.9.3删除容器

docker rm 3a78f5c7a5b9

## 3.9.4删除当前正在运行的容器

docker rm –f 3a78f5c7a5b9

# 四 Docker网络和存储

## 4.1查看iptables和网络映射

iptables -t nat -vnL

Chain PREROUTING (policy ACCEPT 38 packets, 2298 bytes)

pkts bytes target prot opt in out source destination

17550 970K DOCKER all -- \* \* 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT 13 packets, 965 bytes)

pkts bytes target prot opt in out source destination

722 45061 MASQUERADE all -- \* !docker0 192.168.42.0/24 0.0.0.0/0

0 0 MASQUERADE tcp -- \* \* 192.168.42.38 192.168.42.38 tcp dpt:80

0 0 MASQUERADE tcp -- \* \* 192.168.42.44 192.168.42.44 tcp dpt:80

0 0 MASQUERADE tcp -- \* \* 192.168.42.48 192.168.42.48 tcp dpt:80

0 0 MASQUERADE tcp -- \* \* 192.168.42.77 192.168.42.77 tcp dpt:80

Chain OUTPUT (policy ACCEPT 13 packets, 965 bytes)

pkts bytes target prot opt in out source destination

0 0 DOCKER all -- \* \* 0.0.0.0/0 !127.0.0.0/8 ADDRTYPE match dst-type LOCAL

Chain DOCKER (2 references)

pkts bytes target prot opt in out source destination

44 2325 DNAT tcp -- !docker0 \* 0.0.0.0/0 0.0.0.0/0 tcp dpt:66 to:192.168.42.38:80

9 456 DNAT tcp -- !docker0 \* 0.0.0.0/0 0.0.0.0/0 tcp dpt:77 to:192.168.42.44:80

24 1268 DNAT tcp -- !docker0 \* 0.0.0.0/0 0.0.0.0/0 tcp dpt:88 to:192.168.42.48:80

0 0 DNAT tcp -- !docker0 \* 0.0.0.0/0 0.0.0.0/0 tcp dpt:100 to:192.168.42.77:80

[root@iZ28w2vkp1eZ ~]# brctl show

bridge name bridge id STP enabled interfaces

docker0 8000.424c9dd9376a no veth71f728d

veth92f2df2

veth9968c28

vethee90c00

## 4.2映射nginx

[root@jumpserver scripts]# docker run -d -P nginx

438381aa8557cf968e86966e928324f3c6d194c15bef5e43103791313545d188

## 4.3查看容器日志nginx

docker logs 438381aa8557cf968

## 4.4指点端口映射

docker run -d -p 81:80 nginx

## 4.5数据卷应用

docker run -it --name volume-test1 -v /data centos 挂载/data目录

## 4.6查看容器卷信息

docker inspect 25c3cfd24505

## 4.7指定目录挂载数据卷

docker run -it -v /opt:/opt centos

## 4.8指定目录挂载数据卷指定权限

docker run -it -v /opt:/opt:rw centos

## 4.9让一个容器去访问另一个容器的卷

docker run -d --name nfs -v /data centos

docker run -it --name test1 --volumes-from nfs centos

## 4.9.1容器转换

容器转成镜像：  
docker commit <CONTAINER ID> imagename01  
容器转成文件：  
docker export <CONTAINER ID> > /home/export.tar  
文件转成镜像：  
cat /home/export.tar | sudo docker import - imagename02:latest

# 五 手动构建镜像：

## 5.1基于容器做镜像

docker run --name mygl -it centos

rpm –ivh <http://mirrors.aliyun.com/epel/epel-release-latest-7.noarch.rpm>

yum install –y nginx

## 5.2制作镜像文件

docker commit -m "my gl" 512b2ac7db4a gengleiedu/mygl:v1

412834f52f75a38c26893bd53fdf1444efdbe8cf21e17db9a3fb446100ac14e0

docker run -it --name nginxv1 gengleiedu/mygl:v1

[root@885400e40193 /]# vi /etc/nginx/nginx.conf

# \* For more information on configuration, see:

# \* Official English Documentation: http://nginx.org/en/docs/

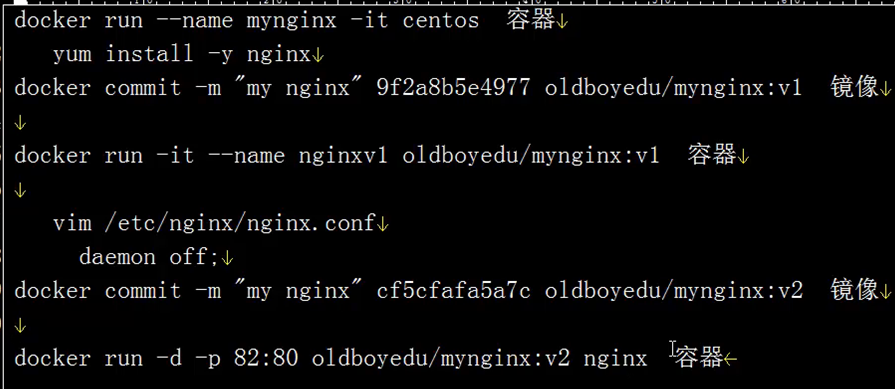
# \* Official Russian Documentation: http://nginx.org/ru/docs/

daemo off; 不在后台运行

docker commit -m "my gl" 885400e40193 gengleiedu/mygl:v2

docker run -d -p 87:80 gengleiedu/mygl:v2

## 5.3制作过程截图



## 5.4服务开机自启动

