**14\_Cloud\_5docker自定义镜像与仓库+持久化存储+docker网络架构**

**使用容器创建nginx服务,并修改首页**

**docker30:192.168.1.30,docker31:192.168.1.31,docker32:192.168.1.32**

docker31 ~]# docker run -itd docker.io/nginx

docker31 ~]# docker exec -it de /bin/bash

root@de6c1a606347:/# dpkg -l | grep nginx #相当于rpm -qa

root@de6c1a606347:/# dpkg -L nginx #查看nginx安装路径

或 root@de6c1a606347:/# nginx -h

root@de6c1a606347:/# nginx -T

root@de6c1a606347:/# cat /etc/nginx/nginx.conf

root@de6c1a606347:/# cat /etc/nginx/conf.d/\*.conf

9 root /usr/share/nginx/html;

root@de6cde6c1a606347:/# cd /usr/share/nginx/html

root@de6c1a606347:/usr/share/nginx/html# echo 'nsd1906' > index.html

docker31 ~]# docker inspect de #找到容器IP

ocker31 ~]# curl <http://172.17.0.2> #测试nginx首页

nsd1906

**使用容器创建apache服务,并修改首页**

docker31 ~]# docker run -it docker.io/centos

7dca9a4ecb09 /]# vi /etc/yum.repos.d/local.repo

7dca9a4ecb09 /]# yum -y install httpd

[root@7dca9a4ecb09 /]# yum -y install net-tools psmisc vim

[root@7dca9a4ecb09 /]# cd /var/www/html/

[root@7dca9a4ecb09 html]# vim index.html

docker31 ~]# docker exec -t 7d /bin/bash

7dca9a4ecb09 /]# find / -name httpd.service

/usr/lib/systemd/system/httpd.service

[root@7dca9a4ecb09 /]# cat /usr/lib/systemd/system/httpd.service

ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND

[root@7dca9a4ecb09 /]# cat /etc/sysconfig/httpd

#查看OPTIONS的定义,为空

[root@7dca9a4ecb09 /]# /usr/sbin/httpd -DFOREGROUND #启动httpd

容器中运行某项服务占用一个终端时,ctrl+p,然后ctrl+q,不关闭容器地退出容器

docker31 ~]# docker inspect 7d #获取IP地址

docker31 ~]# curl <http://172.17.0.3> #测试apache首页

nsd1906

**一 docker自定义镜像与仓库-自定义镜像**

**1.1 docker commit**

**docker commit 容器ID docker.io/新镜像名:标签**

**使用镜像启动容器**

docker31 ~]# docker run -it docker.io/centos:latest /bin/bash

**在该容器基础上修改yum源**

[root@cb2285a22d66 /]# rm -rf /etc/yum.repos.d/\*

[root@cb2285a22d66 /]# vi /etc/yum.repos.d/a.repo

[local\_repo]

name=CentOS-$releasever - Base

baseurl="ftp://192.168.1.254/centos-1804"

enabled=1

gpgcheck=0

**安装常用软件**

[root@cb2285a22d66 /]# yum -y install vim net-tools iproute psmisc

[root@cb2285a22d66 /]# yum -y install bash-completion #命令行tab补齐

**另存为一个新镜像**

docker31 ~]# docker commit cb docker.io/myos:latest

docker31 ~]# docker images #查看新生成的镜像

**1.2 dockerfile**

语法格式:

FROM: 基础镜像

MAINTAINER: 镜像创建者信息

EXPOSE: 开放的端口

ENV: 设置环境变量

ADD: 复制文件到镜像 #相当于简单的scp,把文件传送到容器的指定目录下

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

使用dockerfile创建一个与myos:latest相同的镜像

docker31 ~]# mkdir aa

docker31 ~]# cd aa

docker31 aa]# vim a.repo

docker31 aa]# vim Dockerfile

FROM docker.io/centos:latest

RUN rm -rf /etc/yum.repos.d/\*

ADD a.repo /etc/yum.repos.d/local.repo

RUN yum -y install vim net-tools iproute psmisc bash-completion

docker31 aa]# docker build -t docker.io/myos:test .

docker31 aa]# docker images

使用myos:latest创建一个ssh容器

docker31 ~]# mkdir ssh

docker31 ~]# cd ssh

docker31 ssh]# vim Dockerfile

FROM docker.io/myos:latest

RUN yum -y install openssh-server initscripts && sshd-keygen

RUN echo aa | passwd --stdin root

ENV EnvironmentFile=/etc/sysconfig/sshd

EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]

docker31 ssh]# docker build -t docker.io/myos:sshd .

docker31 ssh]# docker images

docker31 ~]# docker run -itd myos:sshd

3a76...444c

docker31 ~]# docker inspect 3a76 #找到IP

docker31 ~]# ssh 172.17.0.2 #密码输入aa

[root@3a7654b24bb2 ~]# ss -antulp #有端口22

案例:创建一个http镜像,启动容器,自动启动httpd服务,设置默认工作目录为/var/www/html,设置默认首页为hello world

docker31 ~]# mkdir http

docker31 ~]# cd http

docker31 http]# vim Dockerfile

FROM docker.io/myos:latest

RUN yum -y install httpd

WORKDIR /var/www/html

ADD index.html index.html

ENV EnvironmentFile=/etc/sysconfig/httpd

EXPOSE 80

EXPOSE 443

CMD ['/usr/sbin/httpd', '-DFOREGROUND']

docker31 http]# docker build -t docker.io/myos:http .

docker31 ~]# docker images

docker31 ~]# docker run -itd docker.io/myos:http

7ba4...660d

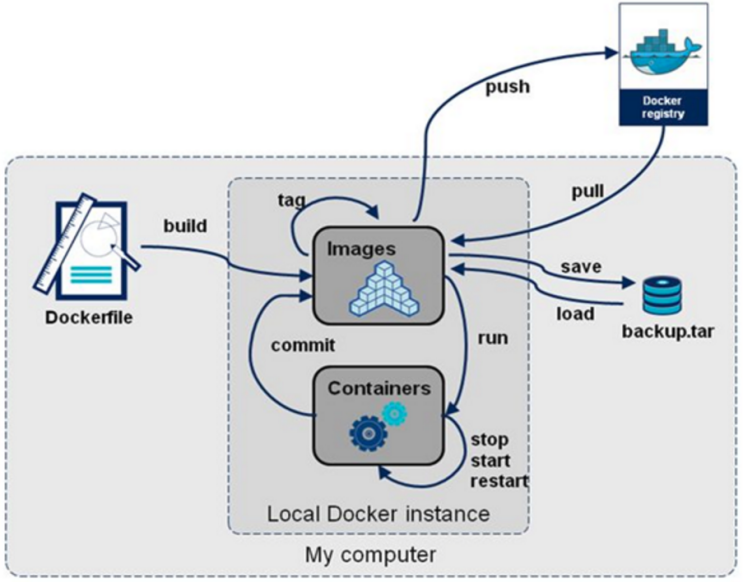
docker31 ~]# docker inspect 7ba4 #找到IP

docker31 ~]# curl 172.17.0.3 #有 hello world

**二 docker自定义镜像与仓库-自定义镜像仓库**

**2.1 registry[ˈredʒɪstri]基本概念**

共享镜像的一台服务器(镜像化的一台服务器)



**2.2 自定义私有仓库**

**2.2.1 服务端安装私有仓库**

registry ~]# yum -y install docker-distribution

**2.2.2 启动私有仓库,并设置开机自启动**

registry ~]# systemctl restart\enable docker-distribution

registry ~]# ss -antulp | grep :5000

网页访问 http://192.168.1.30:5000/v2/\_catalog

**2.2.3 服务端仓库配置文件及数据存储路径**

registry ~]# cat /etc/docker-distribution/registry/config.yml

/etc/docker-distribution/registry/config.yml

/var/lib/registry

**2.2.4 客户端配置:**

修改配置文件/etc/sysconfig/docker

docker31\32 ~]# vim /etc/sysconfig/docker

允许非加密方式访问仓库

24 INSECURE\_REGISTRY=’--insecure-registry 192.168.1.30:5000’

docker仓库地址

13 ADD\_REGISTRY=’--add-registry 192.168.1.30:5000’

重启docker服务(重启前需要停止所有容器)

docker31\32 ~]# docker stop $(docker ps -q)

docker31\32 ~]# systemctl restart docker

**2.2.5 为镜像创建标签:**

这里的地址要写 服务器主机 的IP地址或主机名

docker tag 镜像:标签 IP:5000/镜像:latest

docker31 ~]# docker tag centos:latest 192.168.1.30:5000/centos:latest

**2.2.6上传镜像**

上传镜像的标签内包含地址和端口号

docker push IP:5000/镜像:latest

docker31 ~]# docker push 192.168.1.30:5000/centos:latest

**2.2.7 查看私有镜像仓库中的镜像名称**

okcer30 ~]# curl http://192.168.1.30:5000/v2/\_catalog

{"repositories":["centos"]}

**2.2.8 查看某一仓库的标签**

docker32 ~]# curl http://192.168.1.30:5000/v2/\_catalog

docker32 ~]# curl <http://192.168.1.30:5000/v2/docker.io/centos/tags/list>

**2.2.9 docker客户机使用仓库镜像开启容器**

docker32 ~]# docker run -it 192.168.1.30:5000/centos

docker32 ~]# docker images #将远程仓库的镜像下载下来了

**三 持久化存储-存储卷**

**3.1 卷的概念**

docker容器不保持任何数据

重要数据请使用外部卷存储(数据持久化)

容器可以挂载真实机目录或共享存储卷

**3.2 主机卷的映射**

将真实机目录挂载到容器中提供持久化存存储

容器中目录不存在就自动创建

容器中目录存在就直接覆盖掉

真实机目录可以挂载到多个容器中,实现多个容器的数据共享

docker31 ~]# mkdir /var/dir1

docker31 ~]# docker run -v /var/dir1:/abc -it docker.io/centos:latest

可以同时映射多个 -v

[root@23b0e26fcb05 /]# ls / #有abc目录

**四 持久化存储-共享存储**

**4.1 共享存储基本概念**

一台共享存储服务器可以提供给所有的docker主机使用

共享存储服务器(NSA SAN DAS等)

如: 使用NFS创建共享存储服务器

客户端挂载NFS共享,病最终映射到容器中

**4.2 使用共享存储的案例**

4.2.1 NFS服务器共享目录

registry ~]# yum -y install nfs-utils

registry ~]# mkdir /nfs

4.2.2 修改nfs配置文件

registry ~]# vim /etc/exports

/nfs \*(rw,no\_root\_squash)

okcer30 ~]# systemctl restart nfs-server.service

registry ~]# systemctl enable nfs-server.service

registry ~]# systemctl status nfs-server.service

registry ~]# chmod 777 /nfs

registry ~]# echo abc > /nfs/index.html

4.2.3 配置客户端

docker31\32 ~]# yum -y install nfs-utils

docker31\32 ~]# showmount -e 192.168.1.30

Export list for 192.168.1.30:

/nfs \*

docker31\32 ~]# mkdir /mnt/a

docker31\32 ~]# mount -t nfs -o rw 192.168.1.30:/nfs /mnt/a

docker31\32 ~]# ls /mnt/a

index.html

docker31\32 ~]# cat /mnt/a/index.html

abc

docker31 ~]# docker run -v /mnt/a/:/var/www/html -itd docker.io/myos:http

f852...0a7d

docker31 ~]# docker inspect f852 #找到IP

docker31 ~]# curl http://172.17.0.3

abc

docker32 ~]# docker run -v /mnt/a/:/usr/share/nginx/html/ -itd docker.io/nginx:latest

80208934bf1b

docker32 ~]# docker exec -it 8020 /bin/bash

root@80208934bf1b:/# cat /usr/share/nginx/html/index.html

abc

root@80208934bf1b:/# /usr/sbin/nginx #启动nginx

docker32 ~]# docker inspect 8020 #找到IP

docker32 ~]# curl http://172.17.0.2

abc

**五 docker网络架构-docker网络拓扑**

docker31 ~]# docker network #查看docker network用法

**5.1 查看docker默认网络模型**

docker31 ~]# docker network list

NETWORK ID NAME DRIVER SCOPE

b8c7e5edbef7 bridge bridge local #桥接模型

e9270b435175 host host local #主机模型

26392cc5ce95 none null local #无网络

#后两种基本不用

docker31 ~]# **ip a**[ddress] **s**[how] docker0

docker31 ~]# brctl show docker0 #查看docker0信息

bridge name bridge id STP enabled interfaces

docker0 8000.0242f21a12c4 no

使用镜像启动容器时,默认连接docker0

**5.2 使用docker创建网桥**

docker31 ~]# docker network create --subnet=10.10.10.0/24 docker31

docker31 ~]# docker network list #查看docker网络类型,多了docker31

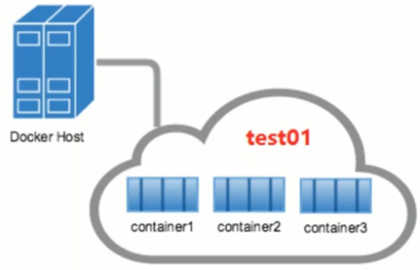
docker31 ~]# ifconfig #查看docker31的IP地址

br-cd39d21d33c2: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 10.10.10.1 netmask 255.255.255.0 broadcast 0.0.0.0

docker31 ~]# docker network inspect docker31 #查看docker31底层信息

新建Docker网络模型



**5.3 使用自定义网桥**

docker31 ~]# docker run **--network=docker31** -itd docker.io/centos

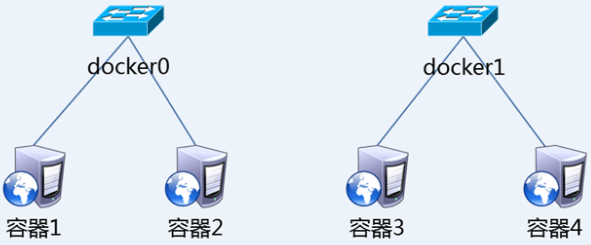
[root@21d58cf492db /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 10.10.10.2 netmask 255.255.255.0 broadcast 0.0.0.0

案例:创建自定义网桥

启动4台容器:容器1与容器2能够互通,容器3与容器4能够互通,12与34不通



docker31 ~]# docker run -it myos:latest

[root@b646cc93cf3f /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 172.17.0.2

ctrl+p ctrl+q 不关闭容器退出

docker31 ~]# docker run -it myos:latest

[root@9314b07abfce /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 172.17.0.3

docker31 ~]# docker run **--network=docker1** -it myos:latest

[root@e16fa3cbd76d /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 10.10.10.2

docker31 ~]# docker run --network=docker1 -it myos:latest

[root@a87faef6c5c9 /]# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 10.10.10.3

docker31 ~]# docker exec -it b6 /bin/bash

b646cc93cf3f /]# ping 172.17.0.3 #通

b646cc93cf3f /]# ping 10.10.10.2\3 #不通

docker31 ~]# docker exec -it e16 /bin/bash

e16fa3cbd76d /]# ping 172.17.0.2\3 #不通

e16fa3cbd76d /]# ping 10.10.10.3 #通

docker31 ~]# ping 172.17.0.2\3 #通

docker31 ~]# ping 10.10.10.2\3 #通

docker31的IP为172.17.0.1和10.10.10.1,docker31是网络docker0和docker1的宿主机

**5.4 客户端访问容器内的资源**

默认容器可以访问外网,但外网不可以访问容器内的资源

容器的特征是可以把宿主机变成对应的服务

使用-p参数把容器端口和宿主机端口绑定

-p 宿主机端口:容器端口

例如 把docker31变成httpd

docker run -itd -p 80:80 docker.io/myos:httpd

例如 把docker31变成nginx

docker run -itd -p 80:80 docker.io/nginx:latest