CLOUD-day06

云平台部署与管理

# 自定义镜像内容

## 自定义镜像

### docker commit

使用镜像启动容器，在该容器基础上修改

另存为一个新镜像

[root@docker1 ~]# docker run -it centos

[root@df0a0022f8b2 /]# vim /etc/yum.repos.d/CentOS.repo

[Centos]

name=centos

baseurl=ftp://192.168.1.254/CentOS-1807

gpgcheck=0

enabled=1

[root@df0a0022f8b2 /]# yum -y install httpd vim net-tools

[root@df0a0022f8b2 /]# exit

[root@docker1 ~]# docker ps -a

[root@docker1 ~]# docker start 678

[root@docker1 ~]# docker commit 678 myos:latest

[root@docker1 ~]# docker ps -a

[root@docker1 ~]# docker images

[root@docker1 ~]# docker run -it myos /bin/bash

### Dockerfile

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

### 使用Dockerfile工作流程

mkdir build; cd build

vim Dockerfile

docker build -t imagename Dockerfile所在目录

**示例1：**

[root@docker1 ~]# mkdir asd

[root@docker1 ~]# cd asd

[root@docker1 asd]# vim Dockerfile

FROM centos:latest

RUN rm -f /etc/yum.repos.d/\*.repo //在容器内执行命令

ADD CentOS-1807 /etc/yum.repos.d/local.repo //将本地文件复制到镜像

RUN yum -y install vim net-tools psmisc iproute && yum clean all

[root@docker1 asd]# cp /etc/yum.repos.d/CentOS-1807.repo ./

[root@docker1 asd]# docker build -t myasd:asd .

//根据当前目录下的Dockerfile文件创建镜像

**示例2：**

[root@docker1 ~]# cd nsd

[root@docker1 nsd]# vim Dockerfile

FROM myos:latest

RUN yum install -y httpd

ENV EnvironmentFile=/etc/sysconfig/httpd

RUN echo "hello world" > /var/www/html/index.html

EXPOSE 80

EXPOSE 443

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

[root@docker1 nsd]# docker build -t myos:httpd .

[root@docker1 nsd]# docker ps

[root@docker1 nsd]# docker ps -a

[root@docker1 nsd]# docker images

[root@docker1 nsd]# docker run -itd myos:httpd

**示例3：**

[root@docker1 ~]# mkdir sshttp

[root@docker1 ~]# cd sshttp/

[root@docker1 sshttp]# vim Dockerfile

FROM myos:sshd

RUN yum -y install httpd

ENV EnvironmentFile=/etc/sysconfig/httpd

RUN echo hello world > /var/www/html/index.html

EXPOSE 80

EXPOSE 443

ADD run.sh /etc/rc.d/init.d/run.sh

CMD ["/etc/rc.d/init.d/run.sh"]

[root@docker1 sshttp]# vim run.sh

#!/bin/bash

/usr/sbin/sshd -D &

/usr/sbin/httpd -DFOREGROUND

[root@docker1 sshttp]# docker build -t myos:sshttp .

### 启动命令的获取方式：

[root@871d771cd118 /]# cat /usr/lib/systemd/system/httpd.service

EnvironmentFile=/etc/sysconfig/httpd

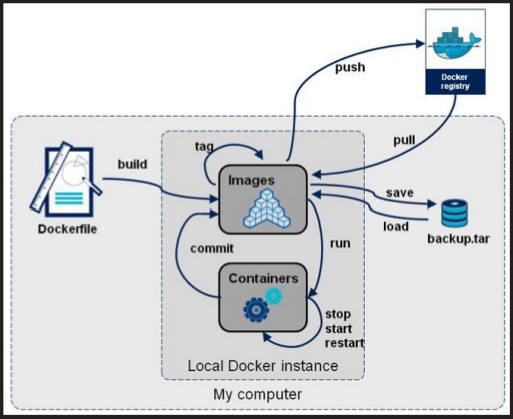
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND

[root@871d771cd118 /]# cat /etc/sysconfig/httpd

## 自定义镜像仓库

## registry基本概念

### 共享镜像的一台服务器（镜像化的一台服务器）



## 自定义私有仓库

### • 流程:

– docker pull registry

– vim /etc/docker/daemon.json

{

"insecure-registries" : ["192.168.1.10:5000"]

}

– docker run -d -p 5000:5000 registry

– docker tag 镜像 IP:5000/镜像:label

– docker push IP:5000/镜像:label

### • 进入registry容器查看/etc/docker/registry/config.yml

[root@docker1 ~]# vim /etc/docker/daemon.json

{

"insecure-registries" :["192.168.1.12:5000"]

}

[root@docker1 ~]# systemctl restart docker

### 启动私有仓库

[root@docker1 ~]# docker run -itd -p5000:5000 registry

### 设置标签

### [root@docker1 ~]# docker tag busybox:latest 192.168.1.12:5000/busybox:latest

### 上传镜像

[root@docker1 ~]# docker push 192.168.1.12:5000/busybox:latest

### 网页查看自定义的镜像源：

[root@docker1 ~]# http://192.168.1.11:5000/v2/\_catalog

[root@docker1 ~]# http://192.168.1.12:5000/v2/myos/tags/list

### 使用镜像

[root@docker2 ~]# scp 192.168.1.12:/etc/docker/daemon.json /etc/docker/

[root@docker2 ~]# systemctl restart docker

[root@docker2 ~]# docker run -it 192.168.1.12:5000/busybox:latest

# 持久化存储

## 存储卷

### 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为卷

### 主机卷的映射

将真实机目录挂载到容器中提供持久化存储

## 共享存储

### 共享存储基本概念

一台共享存储服务器可以提供给所有Docker主机使用

共享存储服务器(NAS、SAN、DAS等)

如：使用NFS创建共享存储服务器

客户端挂载NFS共享，并最终映射到容器中

### 示例：

1.安装一台NFS服务器，共享一个目录

2.docker1 mount nfs 的共享目录

并且把这个目录映射进容器1（nginx）

1. docker2 mount nfs 的共享目录

并且把这个目录映射进容器2（httpd）

实现nginx 和 httpd共享一个web主页

### nfs服务器：

[root@localhost ~]# yum -y install nfs-utils

[root@localhost ~]# mkdir /public

[root@localhost ~]# vim /etc/exports

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

注：no\_root\_squash----->[root@localhost ~]# man exports

[root@localhost ~]# systemctl restart nfs-server

### docker1:

[root@docker1 ~]# yum -y install nfs-utils

[root@docker1 ~]# mkdir /nfsmount

[root@docker1 ~]# mount -t nfs 192.168.1.14:/public /nfsmount

[root@docker1 ~]# cd /nfsmount/

[root@docker1 ~]# docker run -itd -v /nfsmount/:/usr/share/nginx/html nginx

[root@docker1 ~]# echo "hello nginx" > /nfsmount/index.html

[root@docker1 ~]# curl http://172.17.0.5

docker2:

[root@docker2 ~]# yum -y install nfs-utils.x86\_64

[root@docker2 ~]# mkdir /nfsmount

[root@docker2 ~]# mount -t nfs 192.168.1.14:/public /nfsmount

[root@docker2 ~]# docker run -itd -v /nfsmount/:/var/www/html 192.168.1.12:5000/myos:sshttp

[root@docker2 ~]# echo hello httpd > /nfsmount/index.html

通过docker inspect 容器id号查看网址

[root@docker2 ~]# curl 172.17.0.3

# Docker网络架构

## Linux网桥

### 创建虚拟网卡

真实网卡配置文件

cat /etc/sysconfig/network-scripts/ifcfg-eth0

# Generated by dracut initrd

DEVICE="eth0"

ONBOOT="yes"

NM\_CONTROLLED="no"

TYPE="Ethernet"

BOOTPROTO="static"

IPADDR="192.168.1.12"

NETMASK="255.255.255.0"

GATEWAY="192.168.1.254"

虚拟网卡配置文件

cat /etc/sysconfig/network-scripts/ifcfg-eth0:0

# Generated by dracut initrd

DEVICE="eth0:0"

ONBOOT="yes"

NM\_CONTROLLED="no"

TYPE="Ethernet"

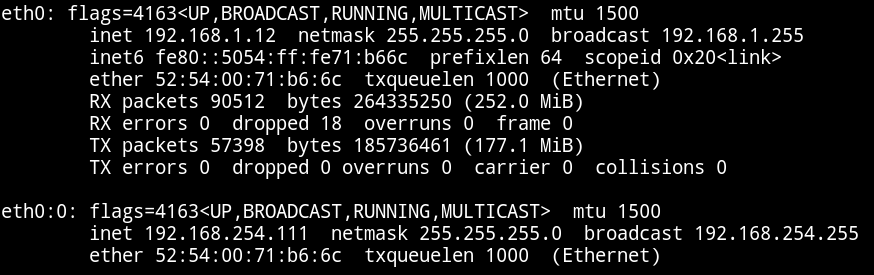
BOOTPROTO="static"

IPADDR="192.168.254.111"

NETMASK="255.255.255.0"

GATEWAY="192.168.1.254"

[root@docker1 ~]# ifconfig



### 创建虚拟网桥

[root@docker1 network-scripts]# vim ifcfg-br0

DEVICE="br0"

ONBOOT="yes"

NM\_CONTROLLED="no"

TYPE="Bridge"

BOOTPROTO="static"

IPADDR="192.168.1.2"

NETMASK="255.255.255.0"

[root@docker1 network-scripts]# vim ifcfg-eth0

DEVICE="eth0"

ONBOOT="yes"

NM\_CONTROLLED="no"

TYPE="Ethernet"

DRIDGE="br0"

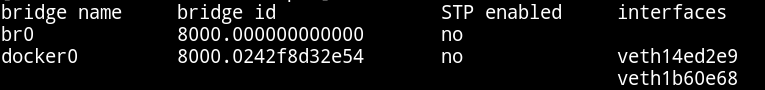
BOOTPROTO="static"

IPADDR="192.168.1.12"

NETMASK="255.255.255.0"

GATEWAY="192.168.1.254"

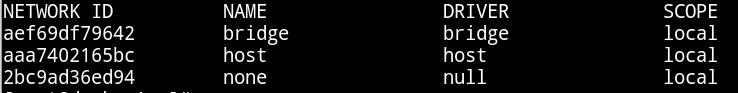
[root@docker1 network-scripts]# brctl show



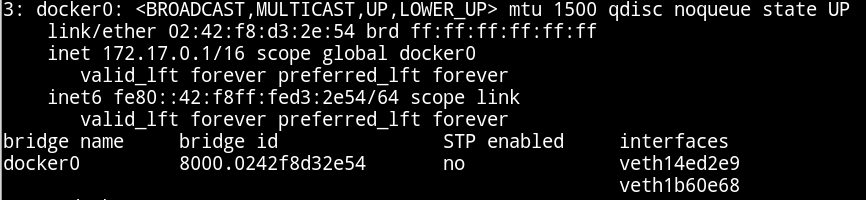
## Docker 网络拓扑

## 查看Docker默认网络模型

[root@docker1 ~]# docker network list



[root@docker1 ~]# ip a s docker0 && brctl show docker0(启动容器后会绑定该内核)

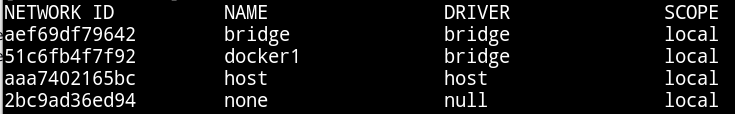


## 使用Docker创建网桥

### 新建Docker网络模型

[root@docker1 ~]# docker network create --driver bridge docker1

[root@docker1 ~]# docker network list



[root@docker1 ~]# ip a s



[root@docker1 ~]# docker network inspect docker1

"Subnet": "172.18.0.0/16",

"Gateway": "172.18.0.1/16"

### 查看默认Docker创建的网络模型

自定义网段

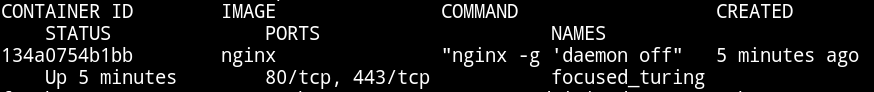
[root@docker1 ~]# docker network create --subnet 172.30.0.0/16 docker1

## 使用自定义网桥

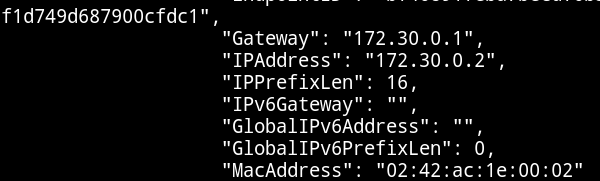
### 启动容器，使用刚刚创建的自定义网桥

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

[root@docker1 ~]# docker ps



[root@docker1 ~]# docker inspect 134



## 客户端访问容器内的资源

### 默认容器通过SNAT可以访问外网

### 但外部网络的主机不可以访问容器内的器源

### 端口映射

- 使用端口映射可以实现外部网络访问容器内的资源

[root@docker1 ~]# docker run -p 80:80 -id myos:httpd

//真实机IP为192.168.1.12，使用-p映射真实机的80端口到容器中的80端口

[root@docker1 ~]# curl http://192.168.1.12