**一、docker安装**

1、

curl https://download.docker.com/linux/centos/docker-ce.repo -o /etc/yum.repos.d/docker.repo

2、yum install -y docker-ce

3、systemctl start docker //启动进程

4、service iptables save //启动服务后，会生成一线规则，需要保存

5、下载镜像

docker pull centos//可以下载centos镜像，速度很慢

配置docker加速器（参考 http://blog.csdn.net/xlemonok/article/details/71403534） vi /etc/docker/daemon.json//加入如下内容

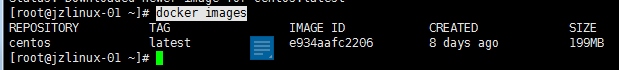
{

"registry-mirrors": ["https://dhq9bx4f.mirror.aliyuncs.com"]

}

6、查看本地有哪些镜像

docker images

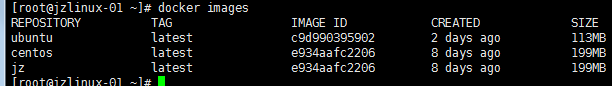


7、docker search jumpserver //搜索镜像

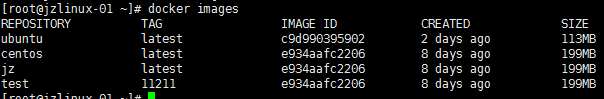
8、systemctl restart docker 重启服务

9、给“centos”，打上名为“jz”的标签，这时候会生成新的镜像

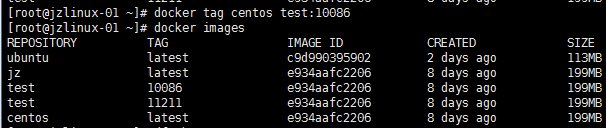
docker tag centos jz //并不是改变TAG，而是第一个参数发生了变化



10、docker tag centos test:11211



11、docker tag centos test:10086



12、docker run -itd centos //把镜像启动为容器

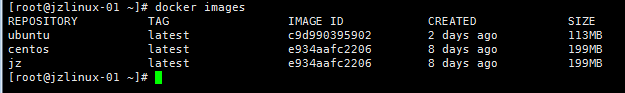


13、docker ps //查看运行的容器，加a选项可以查看所有容器，包括未启动的



14、删除镜像

docker rmi test:10086（TAG名字，不指定会优先删除latest）



**二、通过容器创建镜像**

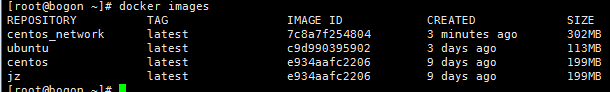
1、docker exec -it 85424 bash //进入启动的容器

2、进入到容器后，安装一些命令



3、安装好命令后，ctrl+D退出来，这样生成一个带上面命令的新镜像

docker commit -m "install net-tools" -a "jz" 85424 centos\_network



**三、通过模板创建镜像**

1、下载模版

http://openvz.org/Download/templates/precreated

2、将该模版导出成镜像

cat centos-6-x86-minimal.tar.gz |docker import - centos6



3、导出一个镜像

docker save -o ubuntu1.tar（导出的文件名） ubuntu（已有镜像的名字）

4、用导出的镜像文件，恢复本地镜像

docker load < ubuntu1.tar 或者 docker load --input ubuntu1.tar

**四、容器管理**

1、docker create -it centos6 bash //创建一个容器，但容器并不会启动

2、docker start f5cd8a07b5de //开启容器

3、docker run -itd --name nginx centos6 bash //给容器自定义一个名字



4、docker exec -it nginx bash //根据名字进入容器

**五、仓库管理**

1、docker pull registry

2、docker run -d -p 5000:5000 registry   //以registry镜像启动容器，-p会把容器的端口映射到宿主机上，:左边为宿主机监听端口，:右边为容器监听端口，就是把服务器上的5000端口和仓库的5000端口做个映射，因为PC是无法直接访问容器的IP的，它需要直接访问你服务器的IP和对应的端口；eg:容器的IP：172.17.0.3:5000，服务器的IP：192.168.1.19:5000,客户端的PC只能访问192.168.1.19:5000，因为做了端口映射，其实就是访问的172.17.0.3:5000



3、docker tag centos6 192.168.1.19:5000/centos6 //传一个镜像到仓库之前，需要将这个镜像打tag，IP为服务器的IP

4、vim /etc/docker/daemon.json //添加下面的内容

{ "insecure-registries":["192.168.1.19:5000"] }

5、重启docker

systemctl restart docker

6、重启docker服务后，私有仓库及容器都停止了，需要启动起来

docker start 82a2 //就是启动registry，数字是他的ID

7、将打tag的镜像push到私有仓库

docker push 192.168.1.19:5000/centos6

8、curl 192.168.1.19:5000/v2/\_catalog //查看私有仓库上有那些镜像



9、上传另外的镜像到私有仓库

docker tag ubuntu 192.168.1.19:5000/ubuntu

docker push 192.168.1.19:5000/ubuntu



**六、数据管理**

1在另外一台设备安装好docker

vim /etc/docker/daemon.json //地址为docker私有仓库的IP地址

{ "insecure-registries":["192.168.1.17:5000"] }

2、systemctl start docker

3、把镜像拉到这台设备上来

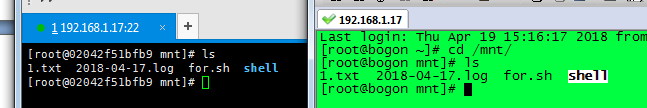
docker pull 192.168.1.17:5000/ubuntu

4、将宿主机的目录挂载到容器目录，避免重启docker导致数据丢失（docker重启后，容器里面数据就没了）

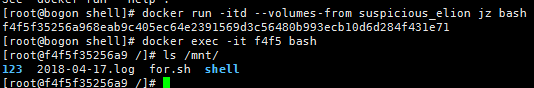
docker run -itd -v /mnt/:/mnt/ centos6 bash //-v 用来指定挂载目录，:前面的/mnt/为宿主机本地目录，:后面的/mnt/为容器里的目录，会在容器中自动创建

docker exec -it 0204 bash //进入上面的容器

容器里面的/mnt/目录内容和宿主机/mnt/里面的东西是一样的，这样2边的目录数据就能实时同步了



5、docker run -itd --volumes-from suspicious\_elion jz bash //，我们使用jz镜像创建了新的容器,并且使用suspicious\_elion容器(将宿主机目录挂载到容器的那个容器)的数据卷，jz是已经存在的镜像。进入该容器后，该容器的/mnt/也和宿主机及上面的容器实时同步了



**七、数据卷备份恢复**

1、docker run -itd -v /data/ --name xp jz bash //将jz镜像开启一个名字为xp的数据卷,进入该容器后，在/data/目录下创建一些数据

2、docker run --volumes-from xp -v /data/backup/:/backup centos tar cvf /backup/data.tar /data/ //将宿主机的/data/backup/和centos镜像开启的容器的/backup目录映射，并且挂载xp数据卷，并将容器的/data/目录打包放到

3、数据恢复

进入第一不启动的容器，删除该容器下/data/目录里面所有数据

docker run --volumes-from xp -v /data/backup/:/backup ubuntu tar xf /backup/data.tar //数据恢复原样

//下面的是将数据恢复到其他新建的容器

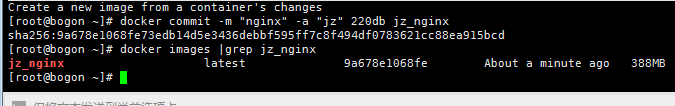
docker run -itd -v /data/ --name w7 centos bash //新建数据卷

docker run --volumes-from w7  -v /data/backup/:/backup centos tar xf /backup/data.tar //数据恢复到了名字为w7的里面

**八、Docker网络模式**

1、进入一个容器，让后下载好ngnix

2、docker commit -m "nginx" -a "jz" 220db jz\_nginx //将安装好nginx的容器导出为一个名字为jz\_nginx的新镜像（-m "自定义" -a "自定义"）

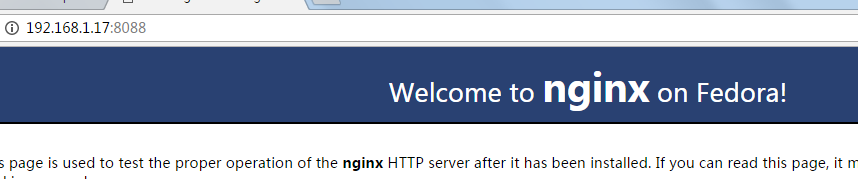


3、docker run -itd --privileged -e "container=docker" -p 8088:80 jz\_nginx /usr/sbin/init //将宿主机的8088和容器的80端口映射，客户端不能直接访问容器，这样后客户端访问宿主机的IP:8080就是访问容器的IP:80

4、进入容器，启动nginx

docker exec -it 62ce bash

systemctl start nginx



**九、 配置桥接网络**

1、cp ifcfg-eno16777736 ifcfg-br0 //拷贝一个新的网卡

[root@bogon network-scripts]# cat ifcfg-eno16777736

TYPE=Ethernet

BOOTPROTO=none

DEFROUTE=yes

PEERDNS=yes

PEERROUTES=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_PEERDNS=yes

IPV6\_PEERROUTES=yes

IPV6\_FAILURE\_FATAL=no

NAME=eno16777736

#UUID=e87877c1-8e81-452e-8410-b48f0e62eff1

DEVICE=eno16777736

ONBOOT=yes

BRIDGE=br0

#IPADDR=192.168.1.17

#PREFIX=24

#DNS1=202.102.224.68

#DNS2=202.102.227.68

#GATEWAY=192.168.1.1

2、

TYPE=Bridge

BOOTPROTO=none

DEFROUTE=yes

PEERDNS=yes

PEERROUTES=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_PEERDNS=yes

IPV6\_PEERROUTES=yes

IPV6\_FAILURE\_FATAL=no

NAME=br0

UUID=e87877c1-8e81-452e-8410-b48f0e62eff1

DEVICE=br0

ONBOOT=yes

IPADDR=192.168.1.17

PREFIX=24

DNS1=202.102.224.68

DNS2=202.102.227.68

GATEWAY=192.168.1.1

3、重启网络服务 systemctl restart network

正常bro有IP,eno16777736无IP

4、安装pipwork

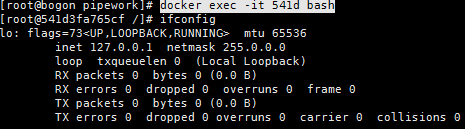
git clone https://github.com/jpetazzo/pipework

5、cd pipework/ //进入该目录

6、cp pipework /usr/local/bin/ //拷贝到PATH下面

7、docker run -itd --net=none jz\_nginx bash //开启一个容器

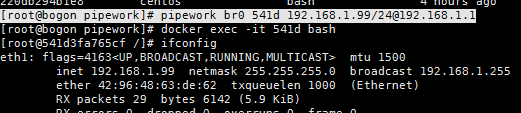
8 docker exec -it 541d bash //进入该容器，没有网络服务



9、退出容器给容器指定IP为

[root@bogon pipework]# pipework br0 541d 192.168.1.99/24@192.168.1.1

10、docker exec -it 541d bash //进入容器，出现了指定的IP



**十、Dockerfile（安装nginx）**

1、vim Dockerfile //在root目录下创建

## Set the base image to jz

FROM jz //指定一个已经存在的镜像

# File Author / Maintainer

MAINTAINER jz jz@jzlinux.com

# Install necessary tools

RUN yum install -y pcre-devel wget net-tools gcc zlib zlib-devel make openssl-devel

# Install Nginx

ADD http://nginx.org/download/nginx-1.8.0.tar.gz .

RUN tar zxvf nginx-1.8.0.tar.gz

RUN mkdir -p /usr/local/nginx

RUN cd nginx-1.8.0 && ./configure --prefix=/usr/local/nginx && make && make install

RUN rm -fv /usr/local/nginx/conf/nginx.conf

ADD http://www.apelearn.com/study\_v2/.nginx\_conf /usr/local/nginx/conf/nginx.conf //下载

# Expose ports

EXPOSE 80 //暴露容器的80端口

# Set the default command to execute when creating a new container

ENTRYPOINT /usr/local/nginx/sbin/nginx && tail -f /etc/passwd

2、docker build -t centos\_nginx . //在当前目录下会自动去执行Dockerfile，-t指定生成新的镜像的名字

3、docker run -itd -p 82:80 centos\_nginx bash //将新生成的镜像启动为容器，并且将

宿主机的82端口和容器的80端口映射



4、生成了新的镜像



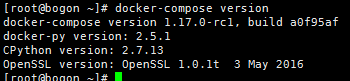
5、docker exec -it d106 bash //进入容器

**十一、docker compose部署服务**

1、wget https://coding.net/u/aminglinux/p/yuanke\_centos7/git/raw/master/25docker/docker-compose-Linux-x86\_64 //下载这个文件

mv docker-compose-Linux-x86\_64 /usr/local/bin/docker-compose

2、chmod 755 /usr/local/bin/docker-compose



3、vim docker-compose.yml

version: "2"

services:

app1:

image: centos\_nginx

ports:

- "8080:80"

networks:

- "net1"

volumes:

- /data/:/data

app2:

image: centos\_network

networks:

- "net2"

volumes:

- /data/:/data1

entrypoint: tail -f /etc/passwd

networks:

net1:

driver: bridge

net2:

driver: bridge

4、docker-compose up -d //同时启动定义的2个容器