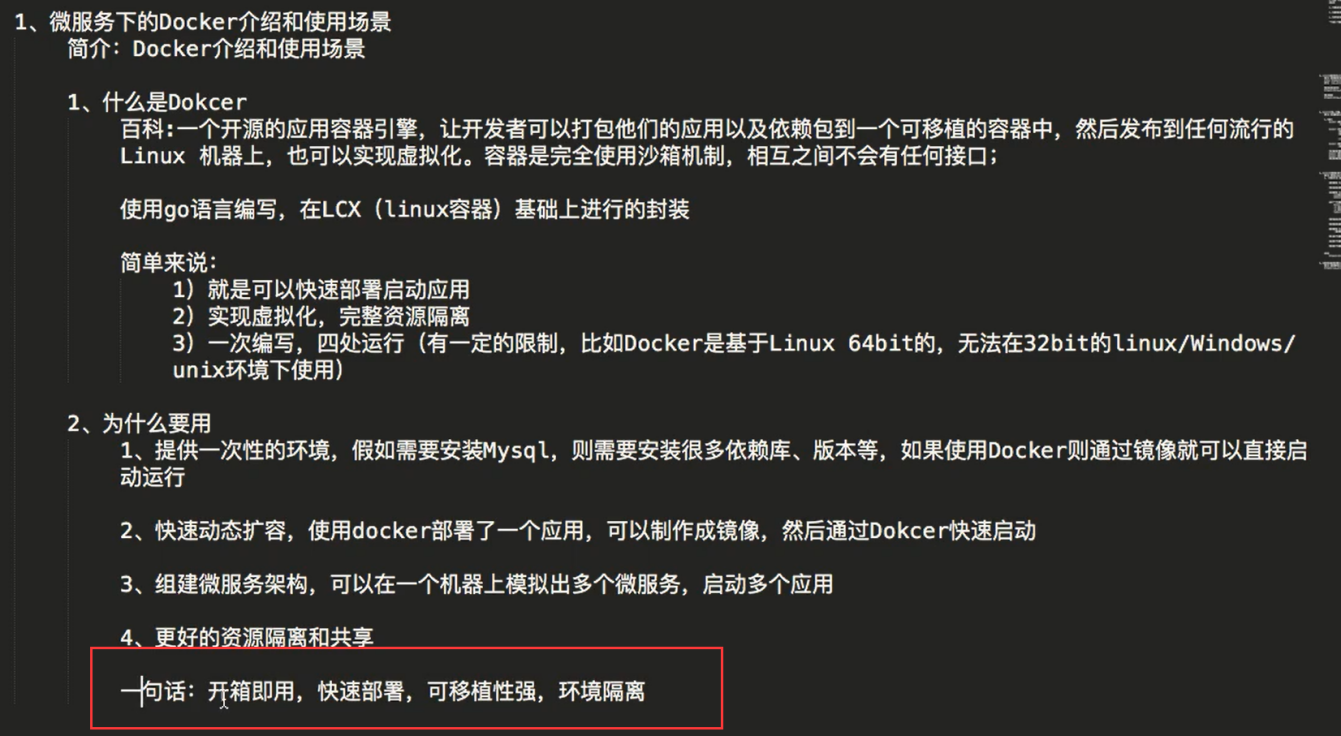
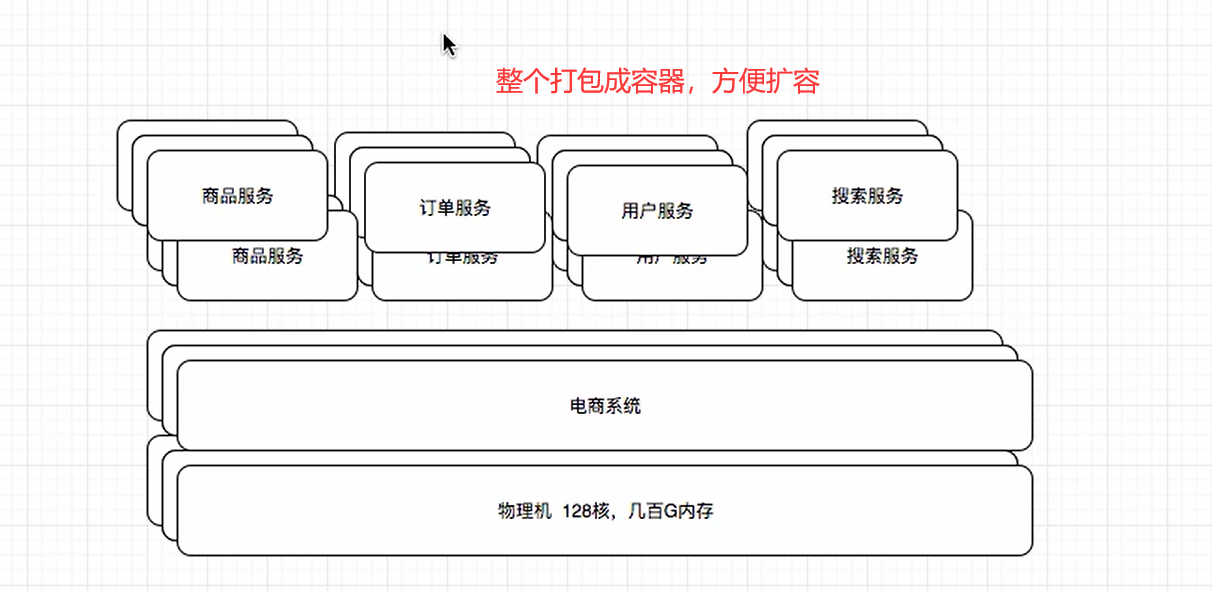
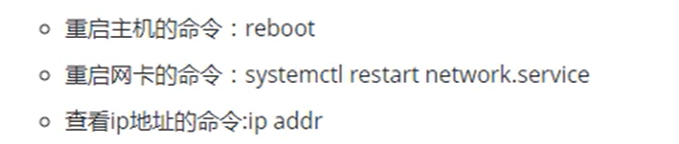
# 为什么要用Docker:





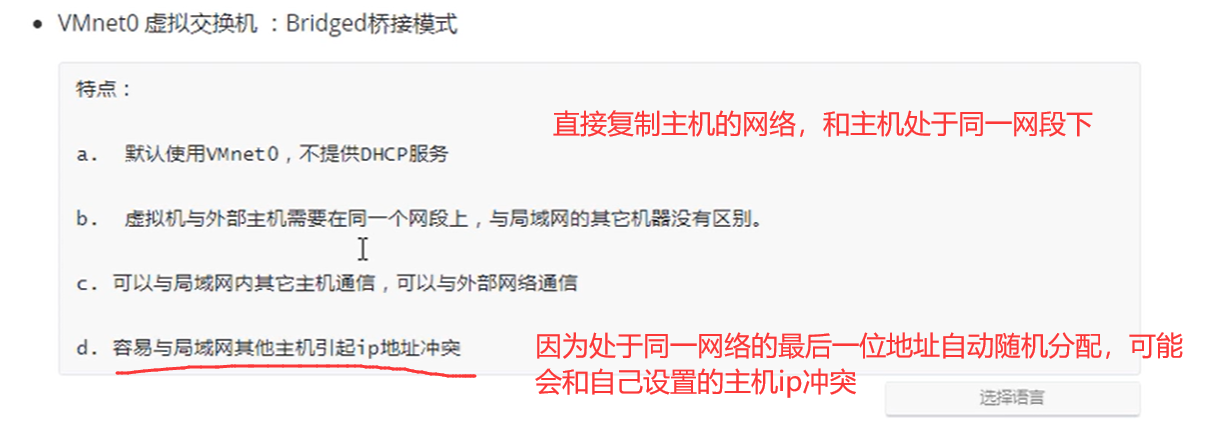
# 2.虚拟机三种网络模式：



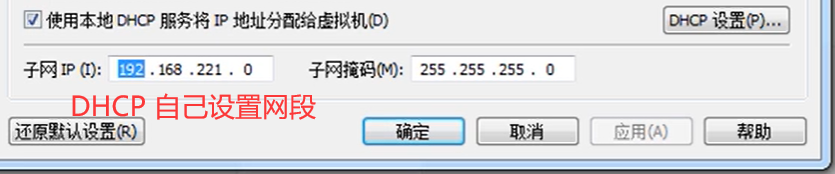
1. 桥接网络

桥接网络直接复制主机的网络连接模式(有线网卡，或者无线网卡)

（懒人模式）



DHCP服务，即手动设置网络地址，桥接模式不提供，直接复制了主机的网段，所以容易ip冲突（docker是默认桥接到容器，但docker0网卡不会和容器ip冲突）



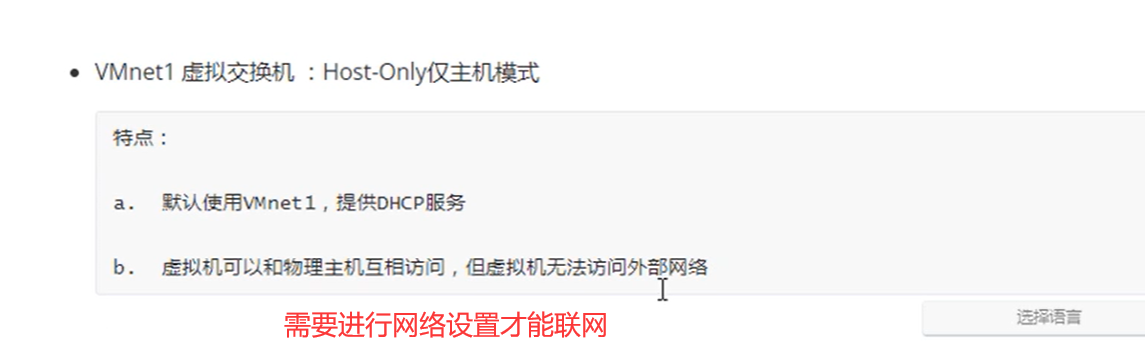
1. Host-only,主机模式:（仅主机和虚拟机互联，隔离外部环境，保证虚拟机安全）

（安全模式）

提供DHCP，自动分配一个不同于主机的网段

同一宿主机下的多个使用host-only模式的虚拟机之间可以互相访问，反正也不上外网，自己玩自己的，安全得很。

该模式是出于安全考虑的，将虚拟机和外部隔离，只和主机进行通信，需要设置才能上网。



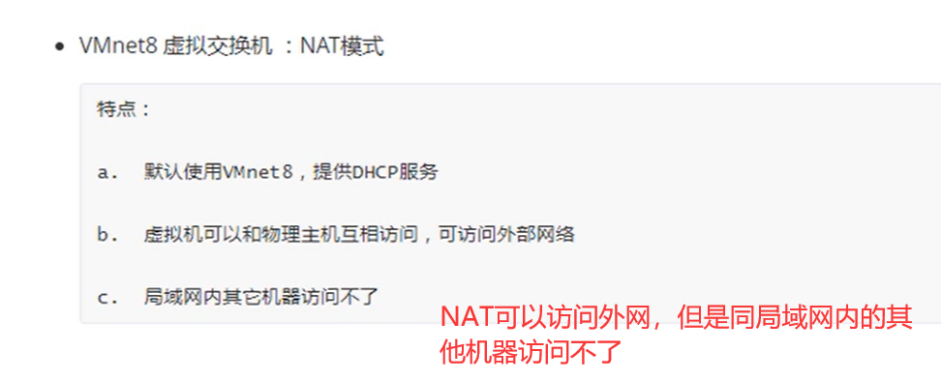
1. NAT模式: （主机和虚拟机互联+可以上外网，但隔离同局域网内其他机器）

（虚拟机霸权模式）

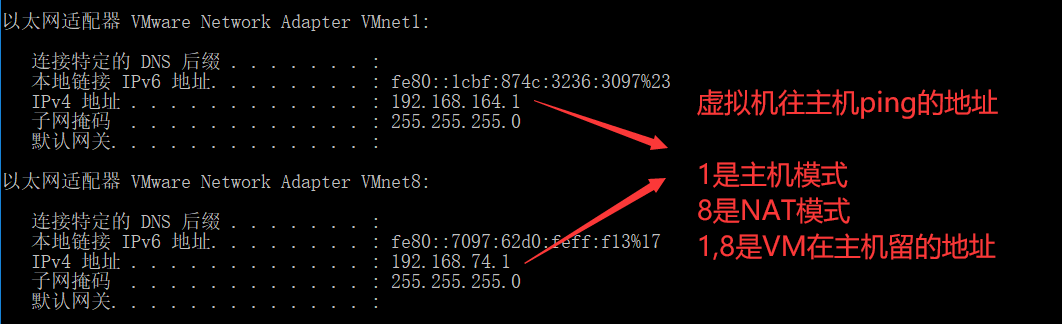
提供DHCP，自动分配一个不同于主机的网段，通过主机来连接外网，此情况下虚拟机可以访问同主机局域网的其他主机，但是同局域网内的其他机器访问不了虚拟机。

（些许霸道，又是靠主机连接外网，还可以自由访问别人，别人还访问不了他）

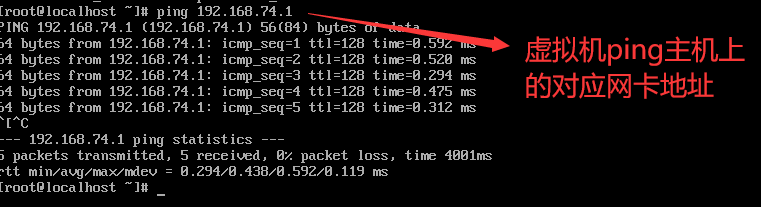
（docker0和主机使用的NAT模式连接）



主机上的VM网卡信息:

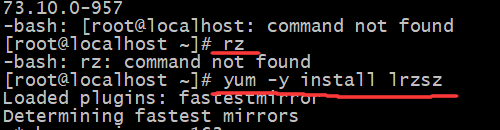


比如NAT模式:

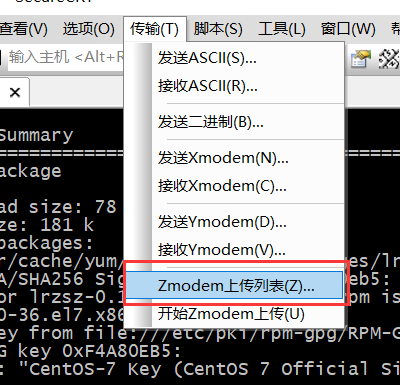


# 3.免FTP的上传和下载

直接使用SecureCRT上的传输即可:

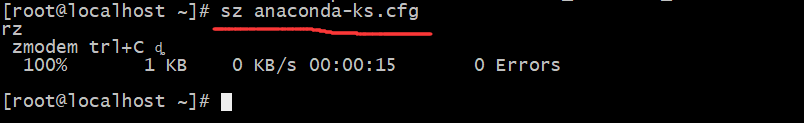


如果上传显示没有rz，要下载这个。然后就可以上传了，直接上传到当前路径

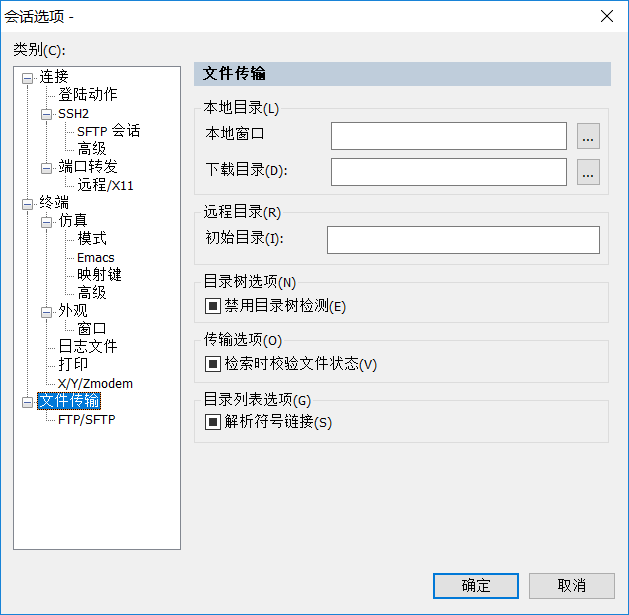
、

下载虚拟机内容到本地:

使用sz +文件名即可: 下载时会提示选择本机路径



上传和下载的路径都可以设置:



# docker安装+镜像制作基础

 sudo yum install -y yum-utils device-mapper-persistent-data lvm2

sudo yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

这两步才能配好yum源

照图上走步骤即可

## 1.docker的基础命令（增删改查，文件挂载）



对容器的操作





文件复制:

Docker cp xxx.txt（文件名） hhh(容器名):/home/

从容器复制到本机（位置交换）:

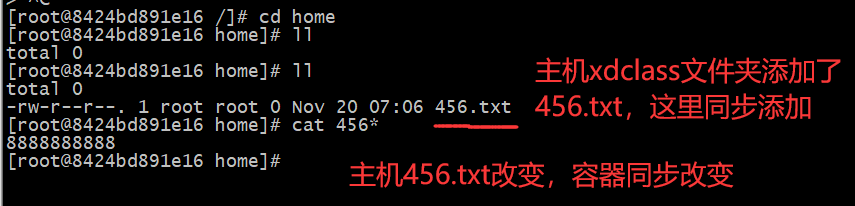
Docker cp hhh(容器名):/home/xxx.txt（文件名） /home/(本机地址)

文件挂载：

这是一个很强大的功能，相当于整个文件夹映射到了容器中的某个文件夹，而且完全同步改变：

Docker run -itd -v /test/xdclass/:/home centos:7 （注意后面冒号不能有空格）

表示主机的xdclass文件夹挂载到容器的/home文件夹，两者以后更新完全同步



相当于多了一份拷贝，以后某个容器宕掉了，直接开一个新的容器，将主机文件夹挂载上去即可继续使用。（数据备份的作用）

## 配置阿里镜像加速



{ "registry-mirrors": ["https://m03o0yz4.mirror.aliyuncs.com"] }

改一下那个json文件就行

## 容器保存为镜像

将修改过后的容器保存为新镜像的方法：

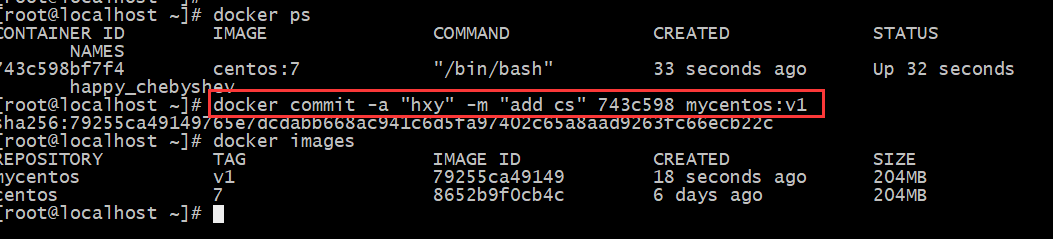
docker commit -a "hxy" -m "add cs" 743c598 mycentos:v1

-a 表示谁修改的“author”

-m 修改信息 “message”

mycentos镜像名

v1:tag号



## 4.使用dockerfile创建镜像:

创建一个dockerfile的文件：

# this is a dockerfile

FROM centos:7 (使用哪一个镜像)

MAINTAINER XD [123456@qq.com](mailto:123456@qq.com) （表示作者，发送邮件）

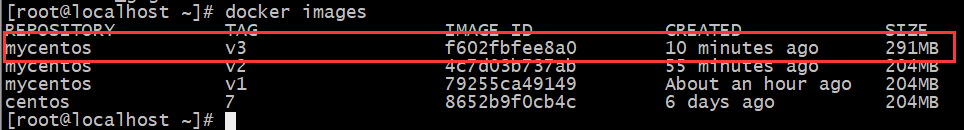
RUN echo "正在构建镜像！！！"

WORKDIR /home/xdclass （工作路径，一进入这个容器，就直接到这个文件夹）

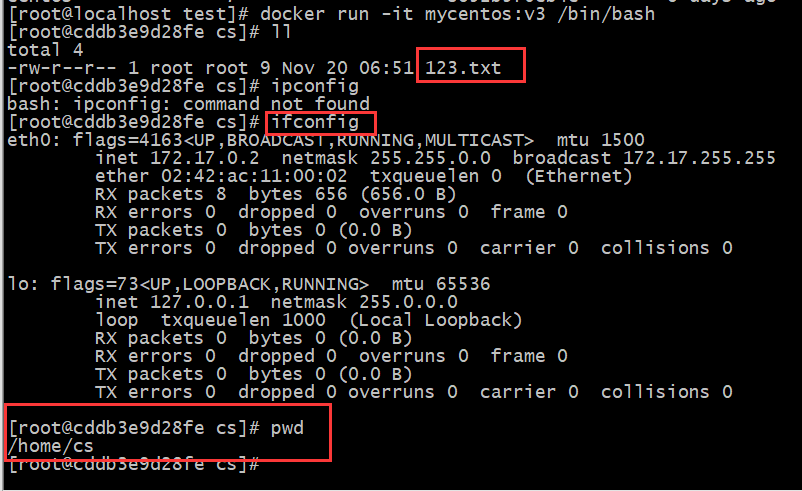
COPY 123.txt /home/xdclass （复制当前路径下的123文件到新容器的xdclass文件夹下，注意这里的123.txt是相对路径！！！）

RUN yum install -y net-tools (完成镜像初始化后需要执行的命令)

docker build -t mycentos:v3 . 构建命令(注意最后有一个”.”表示当前路径下的dockerfile)

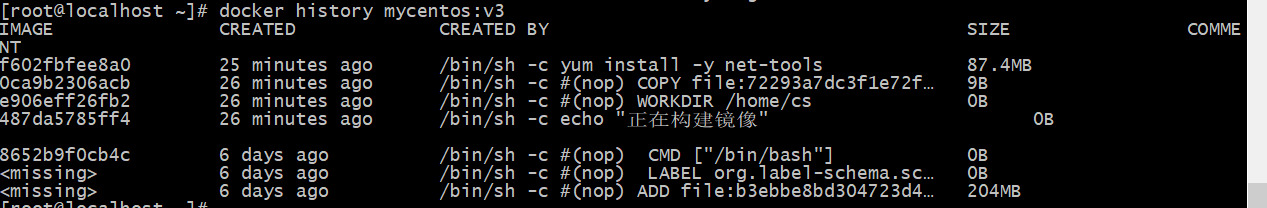


这样就创建成功了



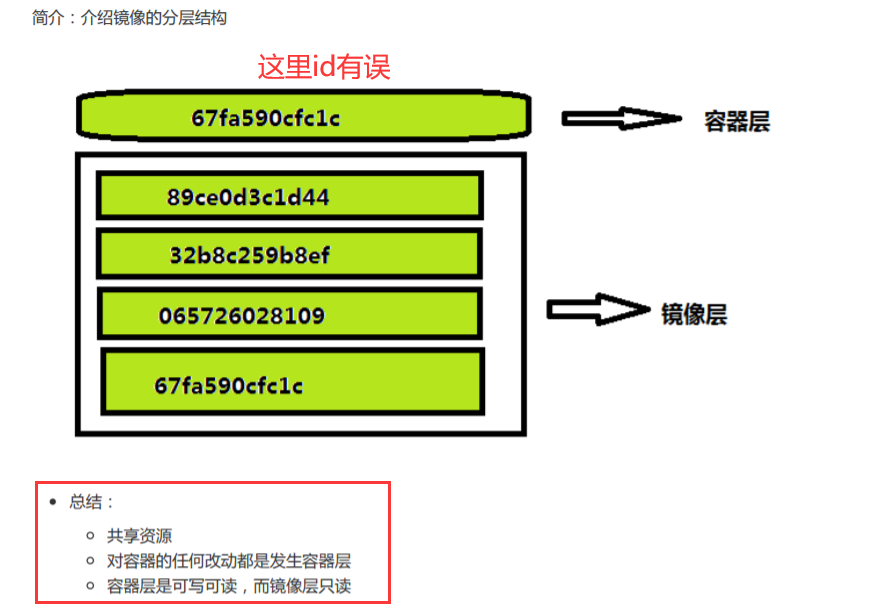
一进入镜像就直接到了/home/class，123.txt也复制好了，net-tools也安装成功了。

构建分层:

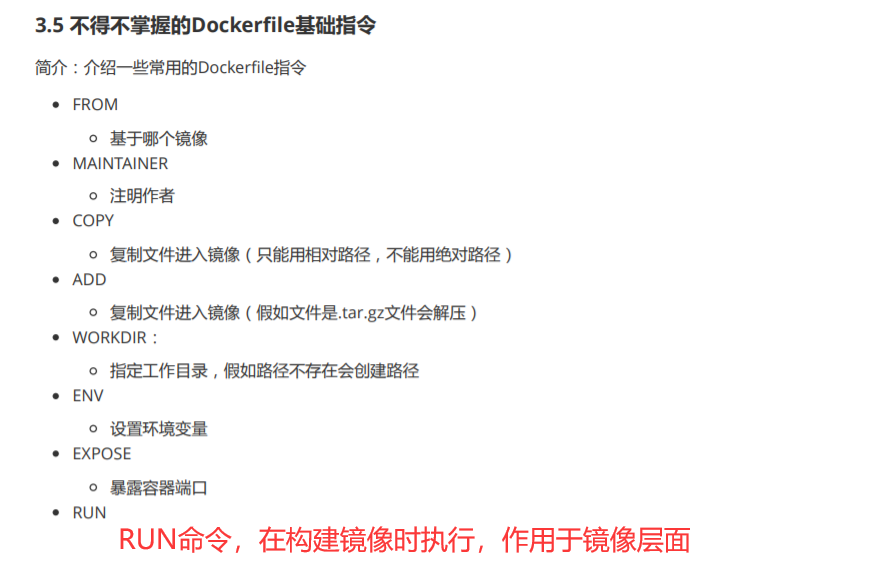


每一步dockerfile中的操作都有一个id

下面的id有无，看得懂意思就行。



## 5.dockerfile基础命令



ADD 和COPY的区别在于 ：ADD在添加命令后会解压或者执行，而COPY就只是简单的复制过去。





注意这里的可接受传参，CMD 和 ENTRYPOINT都是exec的命令格式。

可以通过inspect来查看容器的IP地址



## 6.使用dockerfile创建jdk环境+tomcat环境镜像

大致上就是用脚本来创建镜像，镜像中需要什么东西，就在dockerfile中写，复制进去就完了。

#this is a dockerfile

FROM centos:7

ADD jdk-8u202-linux-x64.tar.gz /usr/local #添加jdk到镜像的/usr/local目录

RUN mv /usr/local/jdk1.8.0\_202 /usr/local/jdk #改名

#配置jdk环境变量

ENV JAVA\_HOME=/usr/local/jdk

ENV JRE\_HOME=$JAVA\_HOME/jre

ENV CLASSPATH=$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

ENV PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH

#配置tomcat

ADD apache-tomcat-8.5.60.tar.gz /usr/local #添加tomcat到镜像的/usr/local目录

RUN mv /usr/local/apache-tomcat-8.5.60 /usr/local/tomcat #改名

EXPOSE 8080 #暴露8080端口

ENTRYPOINT ["/usr/local/tomcat/bin/catalina.sh","run"] #启动容器时，使用前台开起的方式#打开tomcat

docker build –t mycentos:jdk . 会生成一个自带jdk和tomcat的镜像。

开启容器

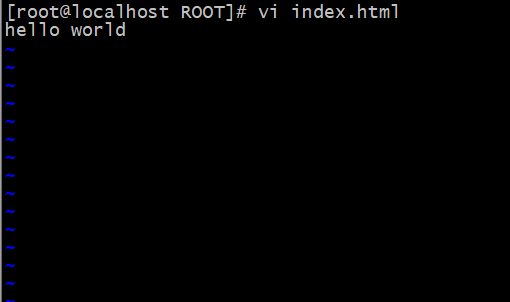
docker run -itd -p 80:8080 -v /test/ROOT:/usr/local/tomcat/webapps/ROOT mycentos:jdk /bin/bash

-p 80:8080是宿主机80端口映射容器8080端口的意思

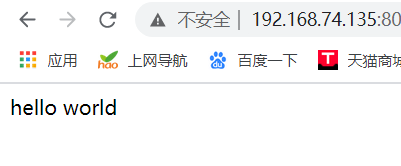
-v /test/ROOT:/usr/local/tomcat/webapps/ROOT 是

挂载宿主机ROOT到容器ROOT的意思

比如在宿主机的Root中添加一个index.html：



端口映射之后，反映如此:



## 7.dockerfile创建nginx环境（脚本使用）:

Nginx 的环境安装有点复杂，需要用到shell脚本，去其文件夹内部安装

脚本nginx\_install.sh如下：

#!/bin/bash //声明脚本

yum install -y gcc gcc-c++ make pcre pcre-devel zlib zlib-devel //安装环境依赖

cd /usr/local/nginx-1.16.1 //进入指定文件夹

./configure --prefix=/usr/local/nginx && make && make install //安装指令

注意：脚本在notepad++写完后，会有换行转义问题，要vi 脚本使用命令:set ff=unix 脚本使用linux换行符，才能正常执行，否则会找不到文件夹。

dockerFile文件如下:

FROM centos:7 //指定源镜像

ADD nginx-1.16.1.tar.gz /usr/local //解压nginx包到指定路径

COPY nginx\_install.sh /usr/local //复制脚本到指定路径

RUN sh /usr/local/ nginx\_install.sh //运行脚本

EXPOSE 80 //暴露80端口，给宿主机使用

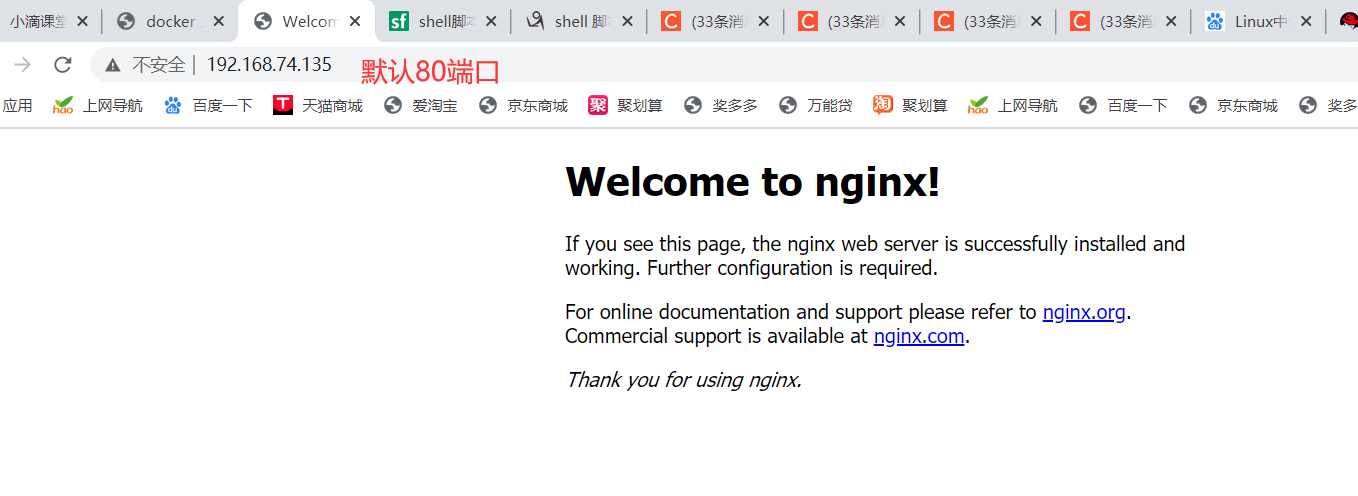
制作打包镜像:

docker build -t mycentos:nginx . (最后有个.)

启动打包好的镜像：也是进入nginx内部文件夹使用前台运行的方式开启

docker run -itd -p 80:80 mycentos:nginx /usr/local/nginx/sbin/nginx -g "daemon off;"

结果:



## 8.dockerfile创建redis环境（脚本使用）:

和nginx差不多的。

redis\_install.sh文件：

#!/bin/bash

yum install -y gcc gcc-c++ make openssl openssl-devel

cd /home/redis-5.0.10

make && make PREFIX=/usr/local/redis install

mkdir -p /usr/local/redis/conf/

cp /home/redis-5.0.10/redis.conf /usr/local/redis/conf/

sed -i '69s/127.0.0.1/0.0.0.0/' /usr/local/redis/conf/redis.conf

sed -i '88s/protected-mode yes/protected-mode no/' /usr/local/redis/conf/redis.conf

最后两行是修改文件配置的命令（后面有解释）

dockerfile 文件:

#this is dockerfile for nginx\_install

FROM centos:7

ADD redis-5.0.10.tar.gz /home

COPY redis\_install.sh /home

RUN sh /home/redis\_install.sh

ENTRYPOINT /usr/local/redis/bin/redis-server /usr/local/redis/conf/redis.conf

//进入容器时执行这一条，启动redis

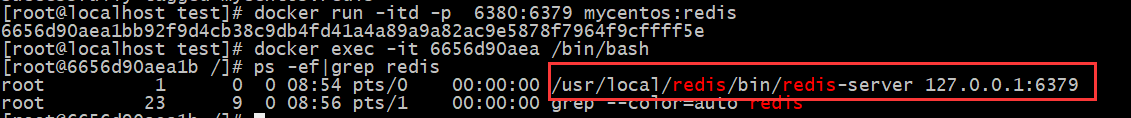
构建镜像：

Docker build –t mycentos:redis .

启动容器:

docker run -itd -p 6380:6379 mycentos:redis #6380是宿主机端口，6379是容器的端口

redis默认是6379端口



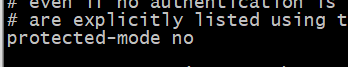
进入容器后，看到红框部分，表示启动redis ok

127.0.0.1表示本机，其他主机连接不进来这个redis

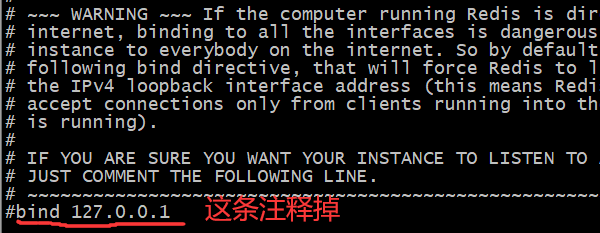
这个redis需要修改配置，外面的机器才能连接:

vi /usr/local/redis/con/redis.conf

:/protected-mode 查到这个保护模式，关闭

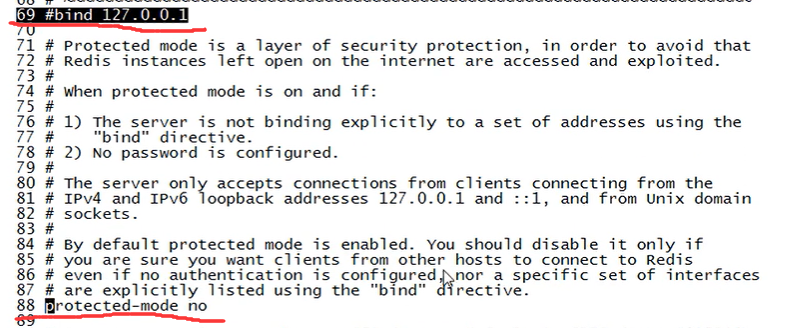


注释掉127.0.0.1单机模式



这个可以脚本进行修改，

首先使用:set number 找到要修改的行号。



分别是69行和88行

Shell命令 sed –i 修改源文件：

sed -i '69s/127.0.0.1/0.0.0.0/' /usr/local/redis/conf/redis.conf

sed -i '88s/protected-mode yes/protected-mode no/' /usr/local/redis/conf/redis.conf

用”/”去隔开要修改的行数，原内容，和修改后的内容，后面加上文件位置。

**有两种方式操作这个redis**

1. 宿主机端口操作:

/usr/local/redis/bin/redis-cli -p 6380

宿主机上的端口映射为6380，所以在宿主机操作，直接上6380端口

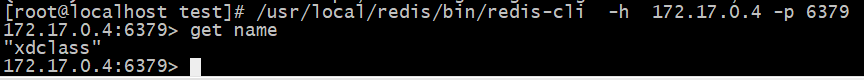


1. 远程容器操作

先 docker inspect查看容器的ip地址

/usr/local/redis/bin/redis-cli -h 172.17.0.4 –p 6379

这里是远程到容器，所以端口为6379



可以看到连接成功

## 9．创建中文可用环境的mysql（基于从官网拉下来的镜像）

从官网拉下来镜像后，需要上官网查看执行创建容器的命令:

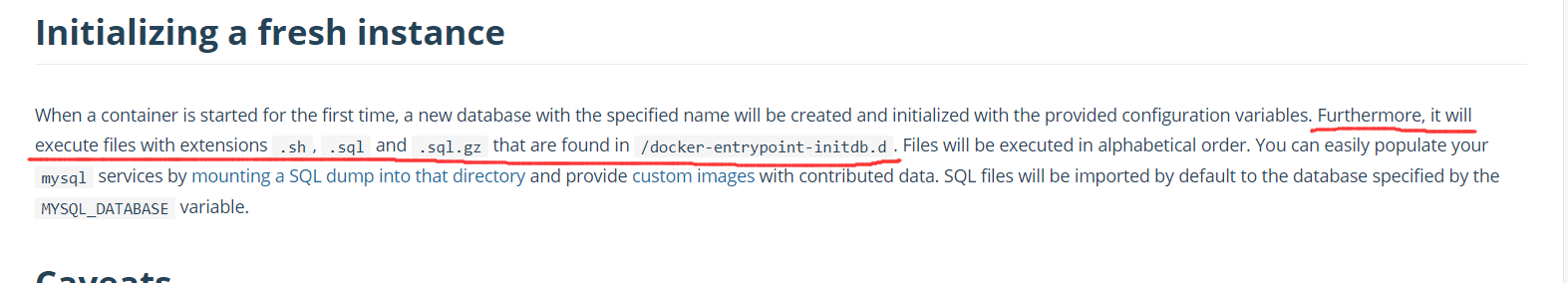


这里就是指定mysql密码和打标签，自己可以加命令，如端口映射。

需要先写一个init.sql 创建mysql容器时，指定执行的sql命令:

就是一些基础数据的添加:





指定 /docker-entrypoint-initdb.d 文件路径为工作路径，执行sql文件

Dockerfile文件:

FROM mysql:5.7

WORKDIR /docker-entrypoint-initdb.d (工作路径，一进入这个容器，就直接到这个文件)

ENV LANG=C.UTF-8

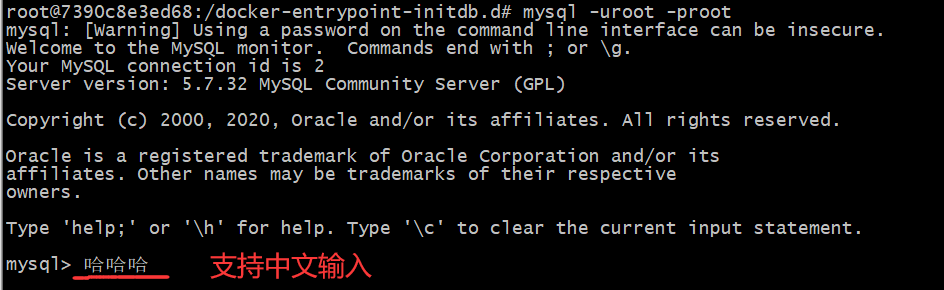
ADD init.sql . //（后面有一个 “.”）表示当前工作路径 ADD表示复制文件init之后立即执行，

制作好镜像之后，打开容器要用官方给的方式打开:

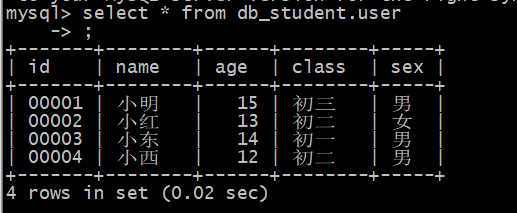
docker run --name some-mysql -p 3307:3306 -e MYSQL\_ROOT\_PASSWORD=root -d mysql:5.7

指定密码和容器名，-d是后台打开的方式

进入验证:



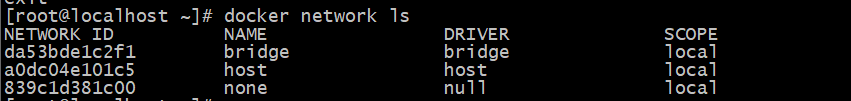
Init.sql执行成功：



# 5.docker的网络模式

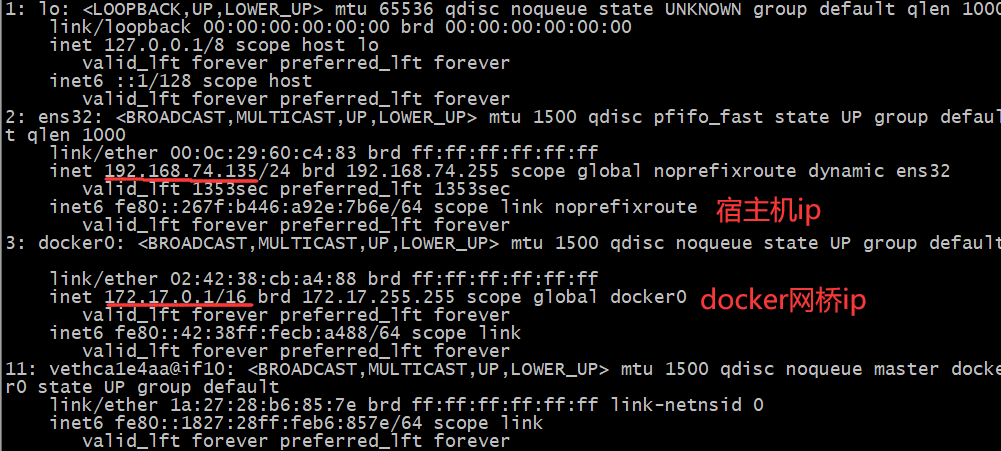
Docker在创建时有三种默认的模式:

和虚拟机的方式很像。



## 桥接模式（默认懒人模式）:

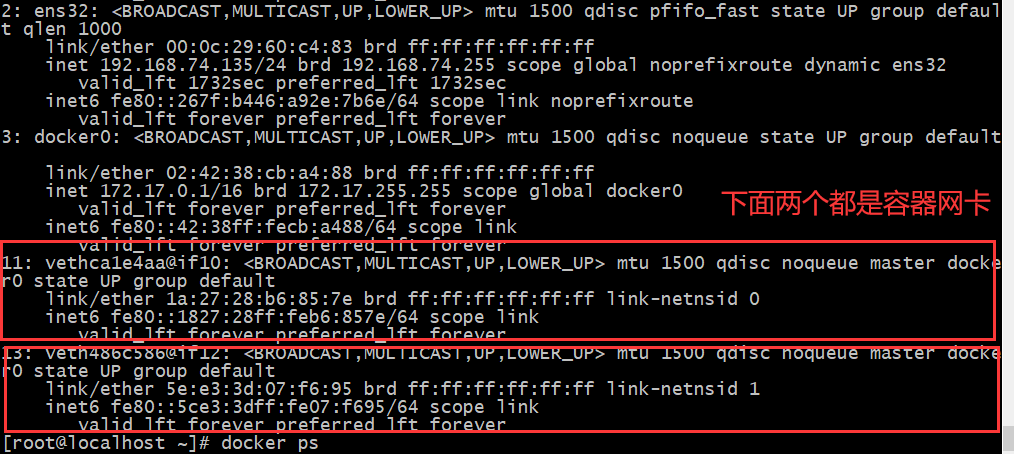
Docker在创建时，会自己生成一个不同于宿主机的ip网桥:



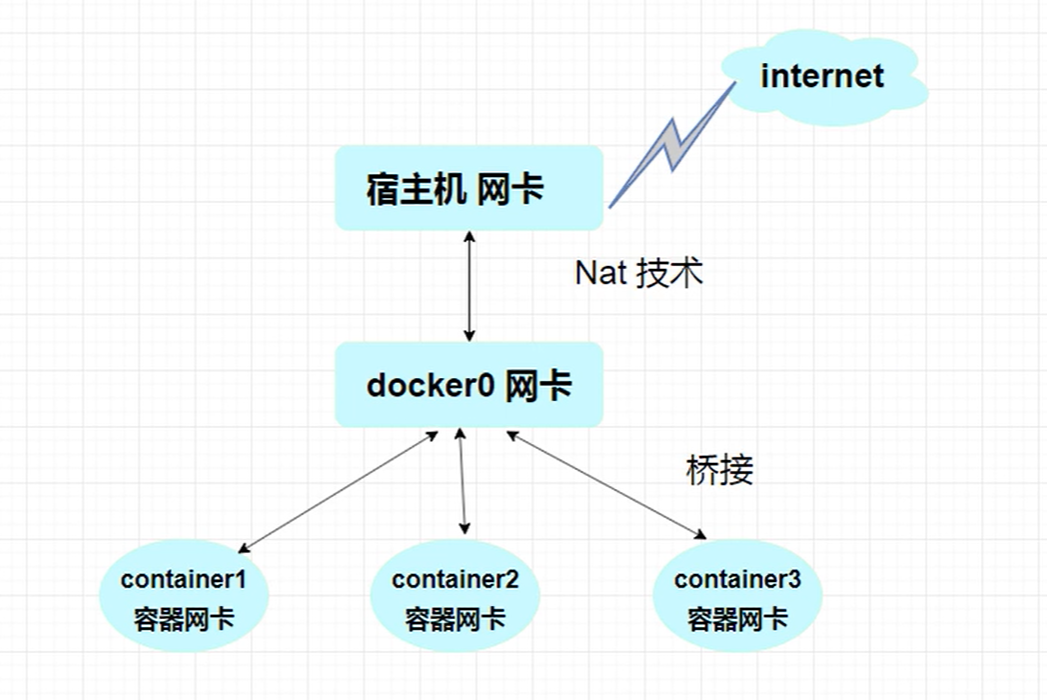
如果使用桥接模式，容器会连接上docker的这个地址来通信，然后docker用这个地址来连通宿主机，访问外网。

每多一个容器，就会多出现一个网卡：

比如现在开两个容器



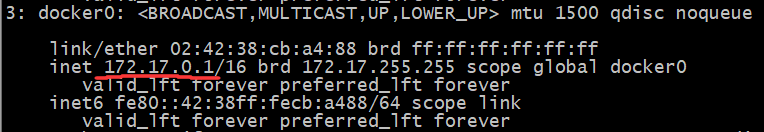
架构图（注意：容器之间的网络是互通的，通过docker0互通，docker0相当于一个路由）



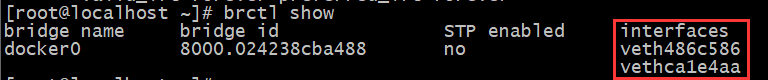
可以在容器内部验证这个图:



而这个地址，就是docker0网卡的地址。

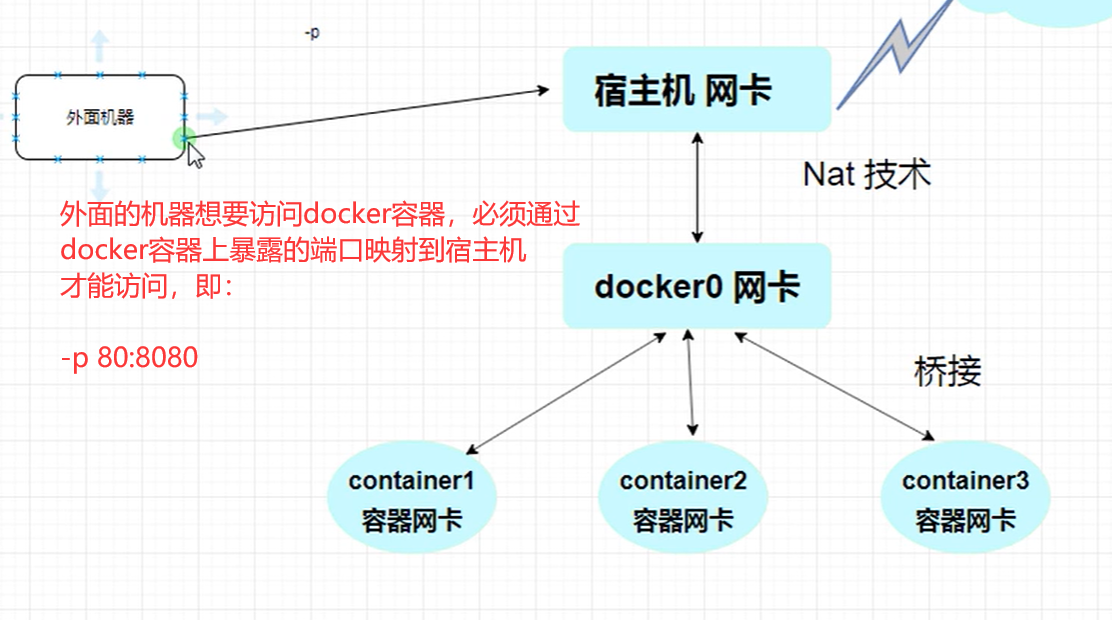


另外，通过桥接检测器，也可以发现docker0和容器网卡的桥接关系:



外面的机器想要访问docker容器，必须要通过容器暴露的端口和宿主机的映射，

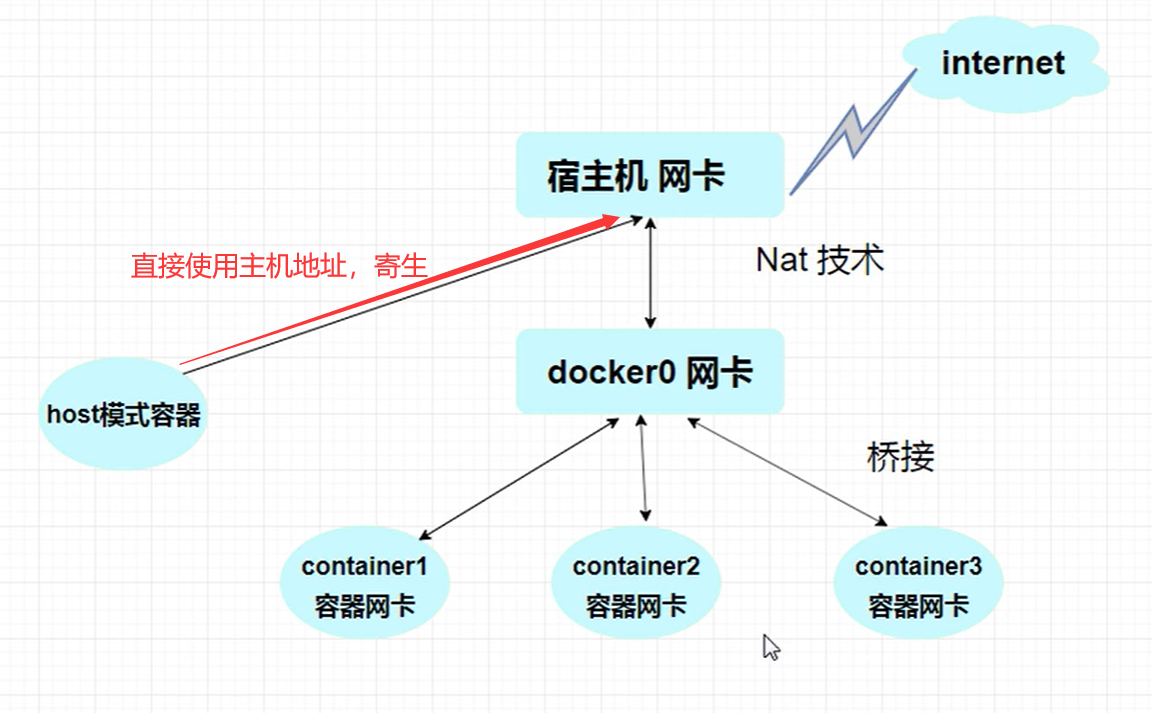
外部机器访问宿主机的端口才行。



## host主机模式:（寄生于主机）

此主机模式和虚拟机的host模式大有不同。这个主机模式是真依赖主机。

主机模式下的容器没有ip地址，而是直接使用主机的ip，应用端口也直接使用主机的端口，相当于“寄生”在主机上。



这样的好处时连接外网不需要NAT转换，效率更高

缺点是直接占用宿主机端口，容易造成端口冲突





验证：开放主机防火墙的80端口

firewall-cmd --zone=public --add-port=80/tcp –permanent

firewall-cmd --reload



## none模式（无网络模式）

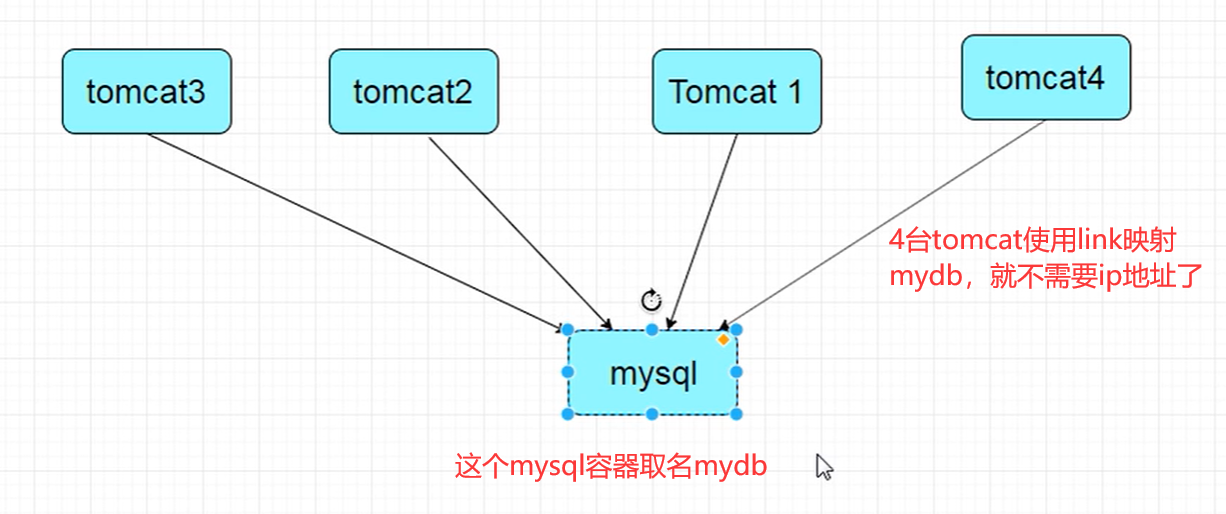
即单机无网络模式。

无ip地址。

很少用，一般用来测试。

## 容器间的代号单向通信Link

就是ip映射机器名，有了映射之后，该机器ping其容器只需要其名称，不需要ip地址。（原来hadoop里面也是这样）

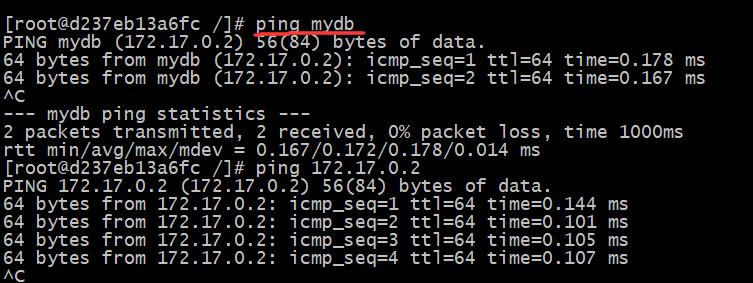


在启动容器时进行映射:

docker run -itd --name tomcat2 --link mydb centos:7

这个mydb容器一定要存在才能映射。

测试:



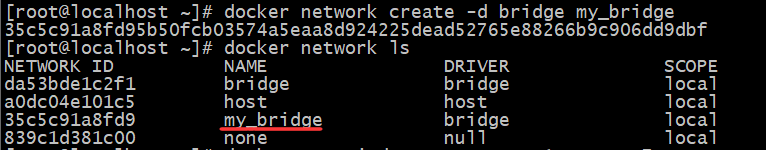
没有进行映射的容器之间只能进行ip地址通信。

## 容器间的代号双向通信（使用网桥）

容器之间代号双向通信需要使用到网桥:，将多个不同的容器，以名称代号的方式放到网桥的list当中，网桥会进行ip地址和名称代号的映射，从而不需要ip地址，也可以进行通信。

先创建一个网桥：

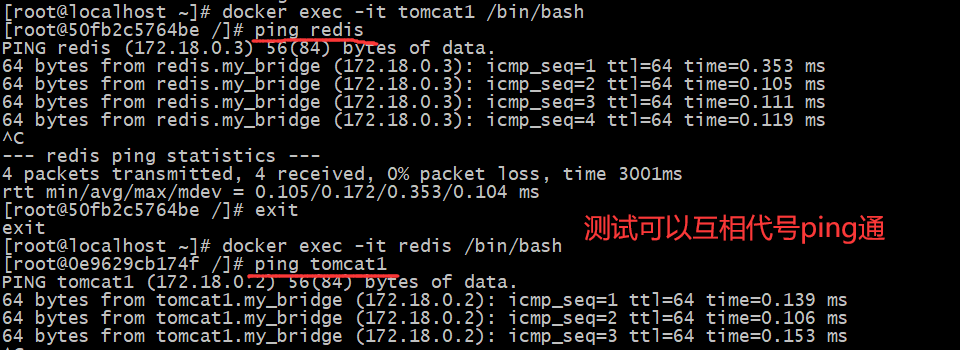
docker network create -d bridge my\_bridge



再将各个容器代号名称加入到网桥当中，这一步进行了ip和名称映射



测试:



## 特权模式



## volume数据共享（挂载详解）

查看容器的挂载路径，使用docker inspect指令查看

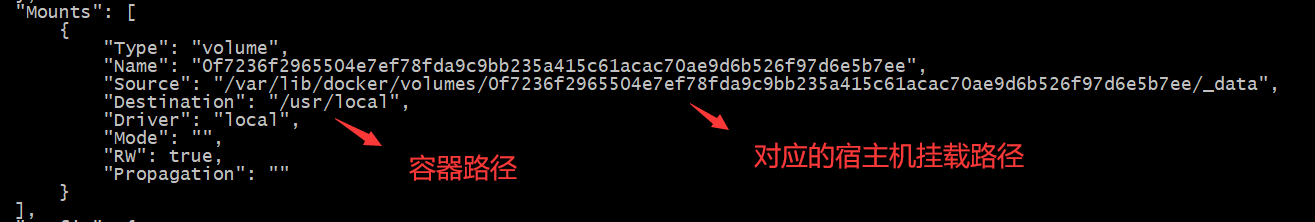


有两部分，dockerfile中多了一个指令VOLUME[“”] 用来从容器挂载到主机（不常用）：

Dockerfile:

FROM centos:7

VOLUME ["/usr/local"] //容器挂载到宿主机,宿主机docker会生成一个文件夹去映射。



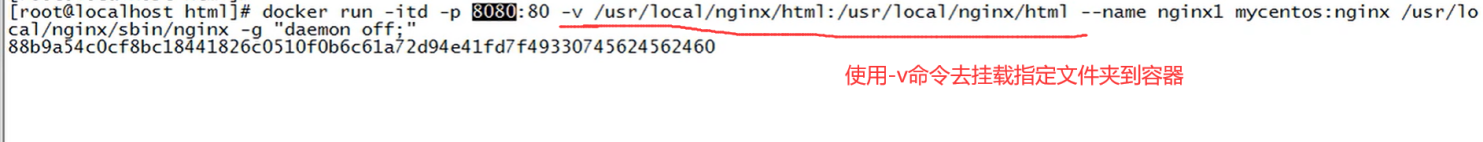
进入宿主机路径查看:



**主要是下面这种从宿主机挂载到容器的模式:**

启动容器时使用命令 –v 去挂载:

Docker run –itd –p 8080:80 –v /usr/local…:/usr/local… --name……



如果需要开多个相同的容器，组成集群，又是同样的挂载

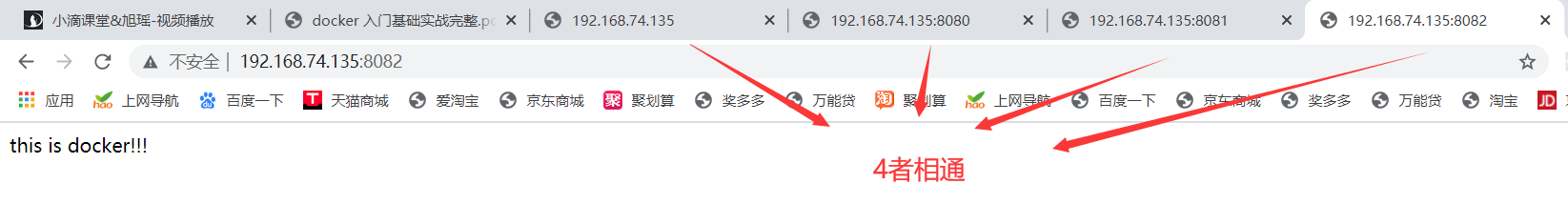
可以使用 --volume-from + 容器名

来实现容器和容器之间的挂载一致。（比写-v详细挂载会方便）

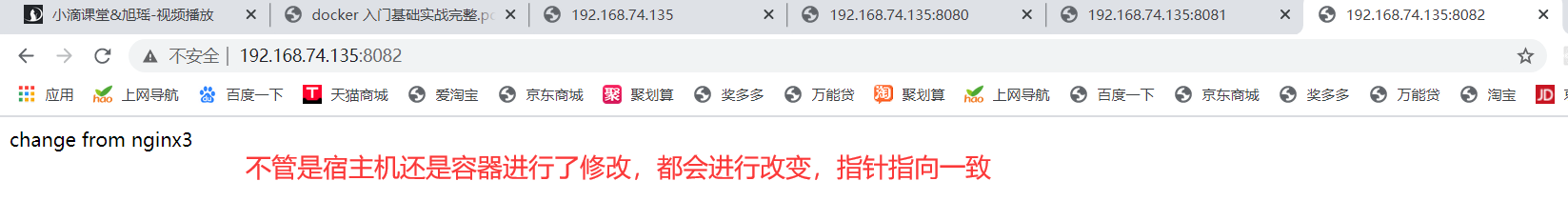
docker run -itd -p 8082:80 --volumes-from nginx1 --name nginx2 mycentos:nginx /usr/local/nginx/sbin/nginx -g "daemon off;"

docker run -itd -p 8081:80 --volumes-from nginx1 --name nginx3 mycentos:nginx /usr/local/nginx/sbin/nginx -g "daemon off;"

这样就打开了两个和nginx1挂载一致的nginx2和nginx3，他们的挂载文件夹指针一致。



挂载就是多出指针指向同一内存，一变都变。



# 6.docker-compose 容器启动+管理工具

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple docker-compose==1.24.1

一个docker的启动工具，原来讲的是怎样制作镜像，并且用命令行的方式启动容器，

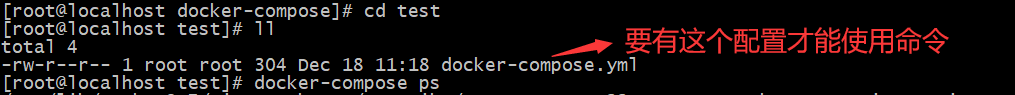
Docker-compose可以使用yml配置文件方式合理的启动各个镜像成容器。

Docker-Compose三大要点:

Version, Services, Network

Docker-compose命令:

注意:docker-compose命令只有在当前目录下有docker-compose.yml文件的时候，才会有用！



## 1. Nginx+redis案列：



一个nginx的docker-compose.yml：



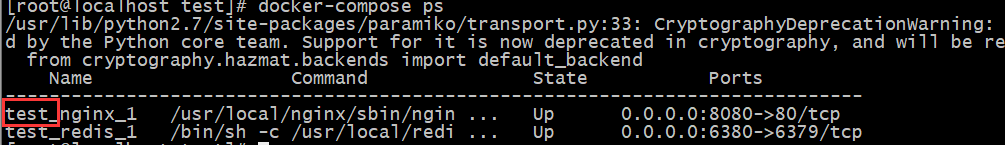
全面文件：



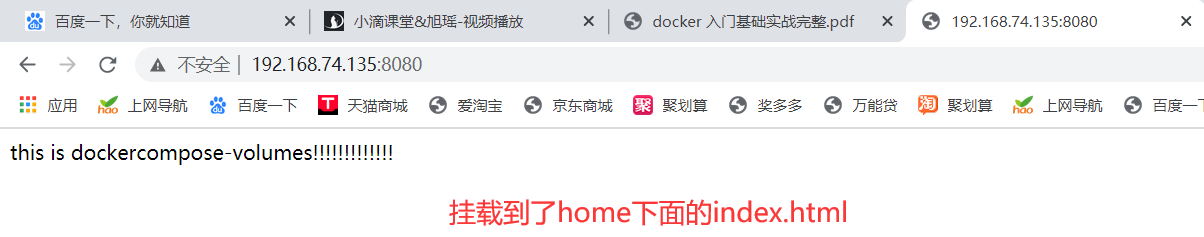
启动：docker-compose up –d

启动之后，名字最前面的单词是当前文件夹的名字

这样就成功映射到宿主机的8080端口和6379端口了。

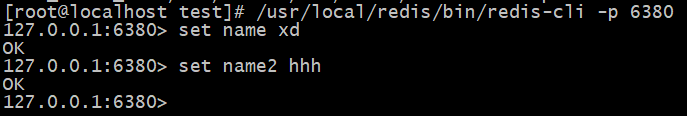


Nginx:

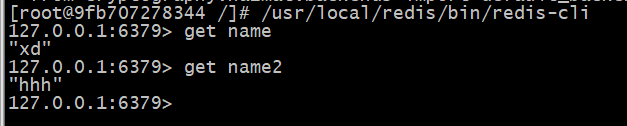


Redis：

主机设置变量：



容器内验证：



成功

## 2.mysql+wordPress 个人博客官网案列：

最后的效果就是一个已经连上数据库的个人博客

通过两个镜像完成，一个mysql镜像，一个wordPress镜像，这两者都可以使用官网的镜像源。

主要看docker-compose的yml文件是怎么启动的：

version: '3.3'

services:

db:

image: mysql:5.7 //镜像源

volumes:

- db\_data:/var/lib/mysql //将db\_data挂载到后面的路径，这里db\_date路径是一个变量，如果不写，docker会在宿主机一个不固定的位置生成文件夹去挂载。这个位置可以通过docker volume指令去查找

restart: always //如果不小心容器挂了，则会自动重启容器

environment: //启动容器时需要指定的参数，docker run … -e MYSQL\_ROOT\_PASSWORD=root -d mysql:5.7

MYSQL\_ROOT\_PASSWORD: somewordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

wordpress:

depends\_on:

- db //相当于--link，将db这个容器的ip地址和“db”连接

image: wordpress:latest //如果本地没有镜像，则会去网上下载

ports:

- "8000:80" //端口映射

restart: always

environment: //启动时参数配置

WORDPRESS\_DB\_HOST: db:3306 //已经经过了link映射，db即代表db的Ip地址。

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

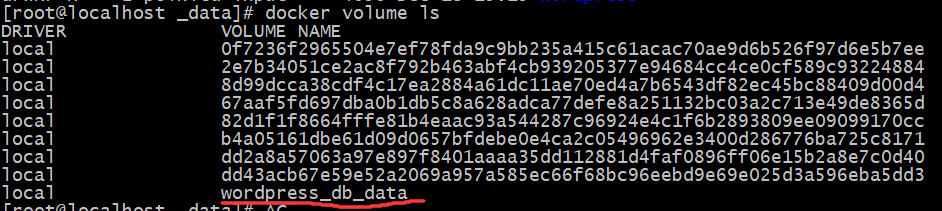
WORDPRESS\_DB\_NAME: wordpress

volumes: //定义挂载路径

db\_data: {} //这里默认为空，可以自己写路径，不写的话docker会在宿主机不固定的位置进行映射

查询挂载的方法:

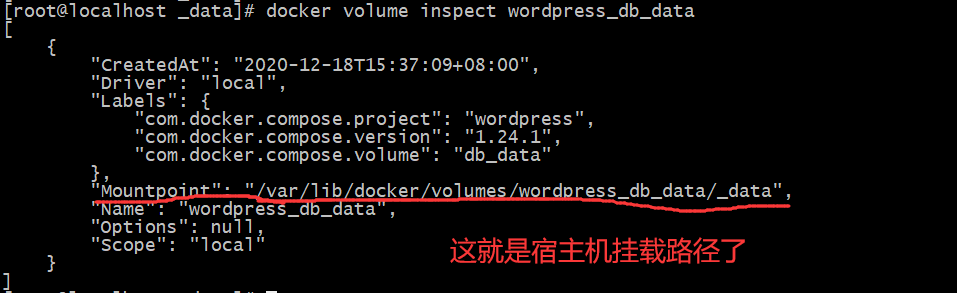
docker volume ls查看使用卷情况:



看到有卷是wordpress和db的。（通过docker-compose启动的容器，卷才会有名字，

上面的全是命令行启动的，就是随机字符）

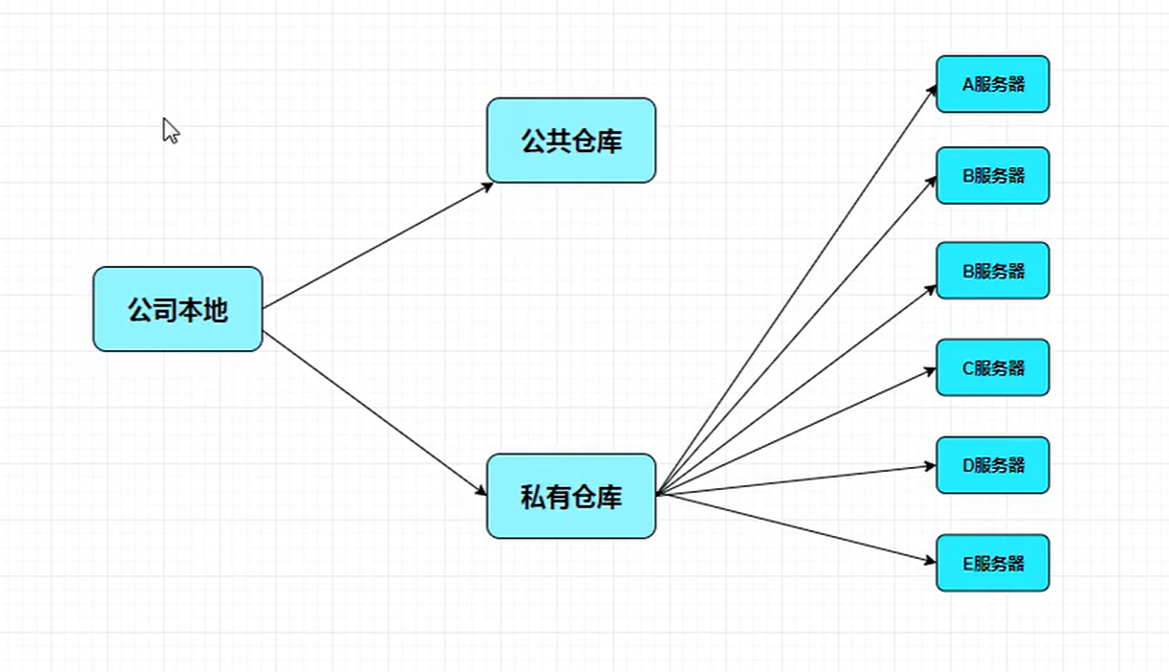
docker volume inspect xxx 查看挂载卷情况。



宿主机这里路径的文件和db容器里/var/lib/mysql是一样的（挂载）

# 7.镜像仓库

除了docker官网的总镜像仓库，公司一般会有自己的研发好的镜像组成的仓库。



## 1.阿里镜像仓库的搭建：

<https://account.aliyun.com/login/login.htm?oauth_callback=https%3A%2F%2Fcr.console.aliyun.com%2Fcn-shanghai%2Finstances%2Frepositories&lang=zh>

阿里云镜像仓库申请地址

创建一个镜像仓库后，里面会有拉去镜像，和上传镜像的方法。

上传镜像前需要把镜像名和版本号更改。

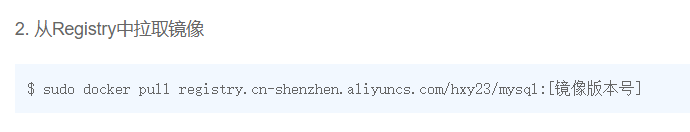


上传的镜像可以在镜像版本查看。



在新机器上下拉镜像，则使用：

前提要登录



即可下拉上传的应用镜像，开心的使用。

## 2.harbor镜像仓库的搭建



直接进入主机地址进行验证,harbor的端口是80,为默认启动端口

