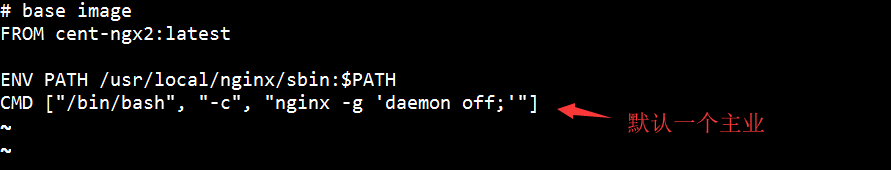
### nginx镜像制作实战

#### docker容器的主业

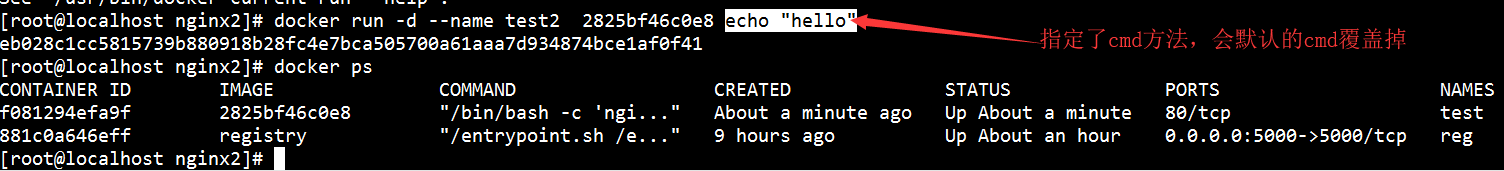
docker理念里，容器启动时，应当为它指定主业是什么，如nginx容器主业就是nginx代理服务，tomcat容器就是web服务等等

1、容器创建时，必须指定主业任务，如不指定，则容器无事可干立即退出。

2、在dockerfile打包镜像时，可以使用cmd命令来指定一个默认的主业，如下：



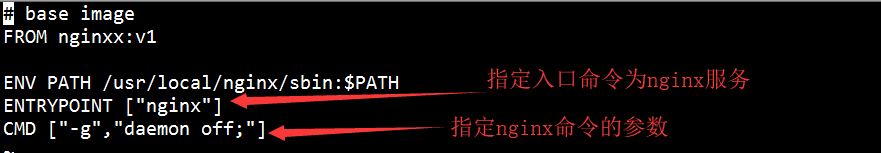
3、既然镜像里是默认主业，即意味着创建容器时，可以覆盖此默认命令，如下



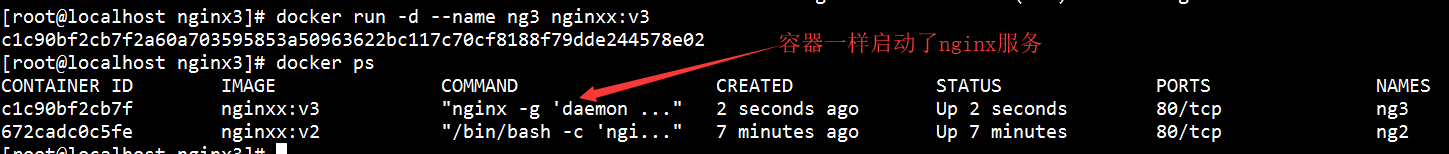
#### 推荐的ENTRYPOINT方式

1、镜像本身应该有稳定的主业，应当指定后即不能更改用途，于是引入ENTRYPOINT

2、使用ENTRYPOINT字义即容器入口，它不能被run中cmd覆盖，如下例：



执行：docker build -t nginxx:v3 .



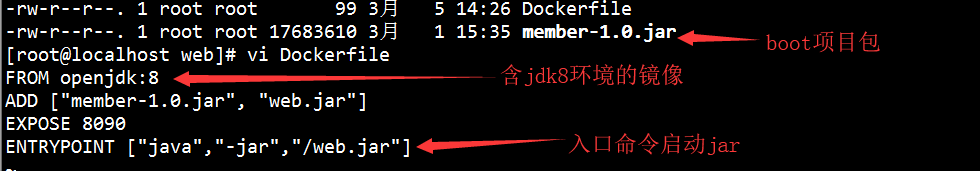
以后使用nginxx:v3这个镜像时，只能做nginx服务来使用啦

#### 手动打包springboot镜像

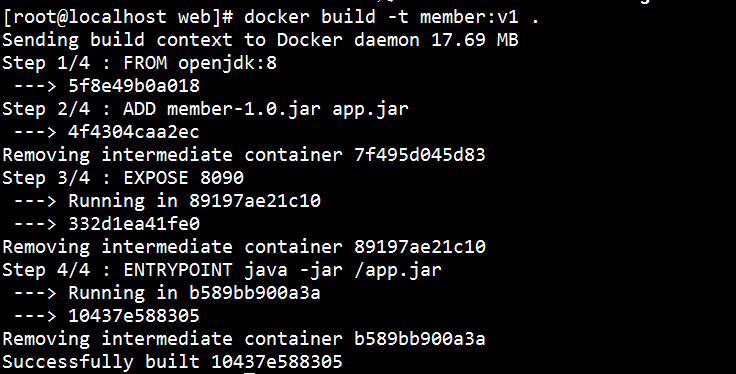
我们需要对业务项目打包发布，一样需要制作成为业务镜像，供运维使用，下面讲述springboot的制作过程：

1、将springboot打好的jar包上传

2、在同级目录下，创建Dockerfile文件，内容如下：

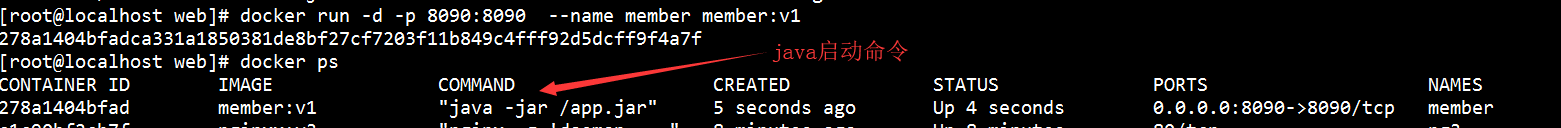


3、dockerfile打包业务镜像

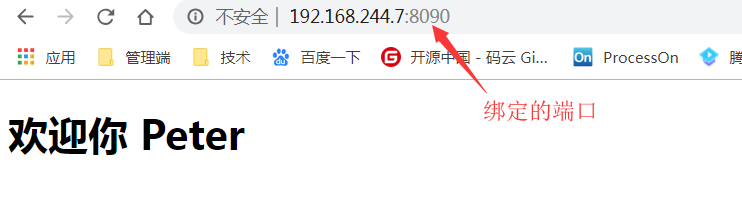


4、启动镜像，即得到业务运行

docker run -d -p 8090:8090 --name member member:v1



5、浏览器打开页面校验：http://192.168.244.7:8090/



#### maven源码打包用法

更多的情况，我们是直接在运维环境里，上传源码，直接maven打包jar，然后再进一步打包成镜像，与手动打包过程类似

如果环境中没有安装maven，请手动安装，脚本如下：

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

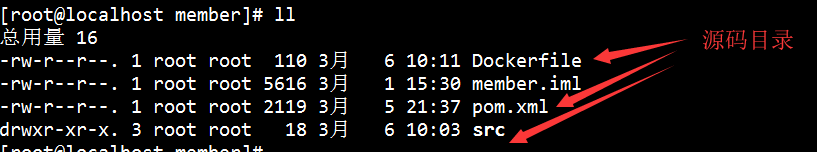
# yum-config-manager --add-repo http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo

# yum-config-manager --enable epel-apache-maven

// 安装maven

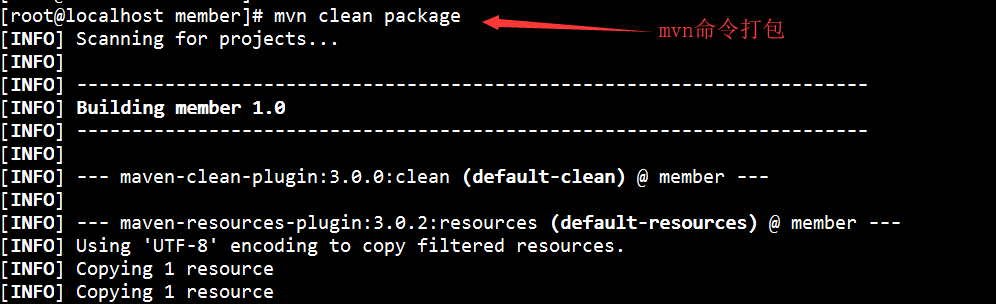
# yum install -y apache-maven

1、上传原码到docker环境中（一般是git/svn直接拉取源码）

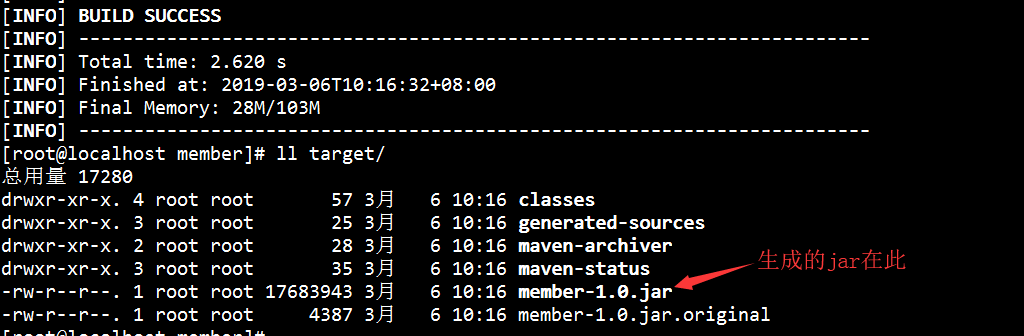


2、maven打包

mvn clean package

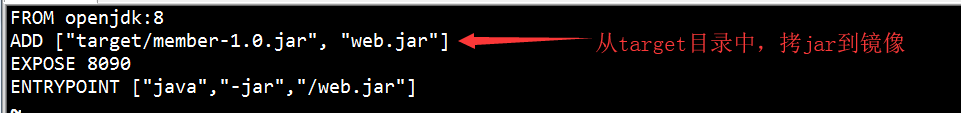


生成的jar在同级target目录下

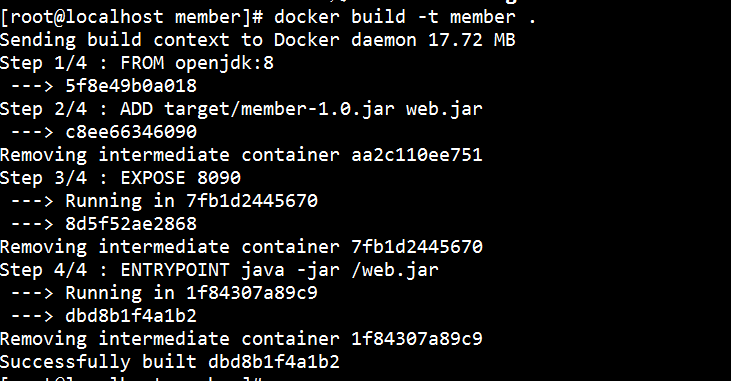


3、执行docker命令生成镜像

dockerfile文件内容



命令创建镜像



#### maven插件打包

前面打springboot包的方式，需要手动上传项目jar或者源码到服务器（违和感很强），这对于开发人员日常发布开发环境项目，极为不便

下面，演示一个maven插件：docker-maven-plugin用法，来打通环境。

##### 前提条件

1、需要我们windows上安装docker服务

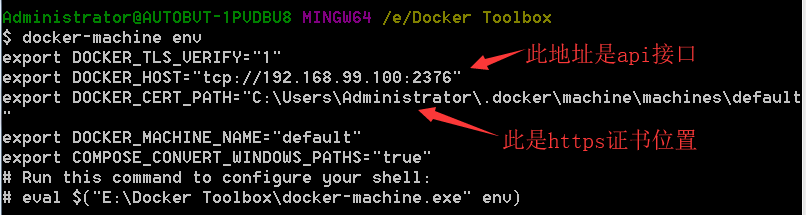
2、需要docker服务配置http仓库接口，windows上docker服务配置如下（传统配置模式无权限修改文件）

##### 本地环境配置

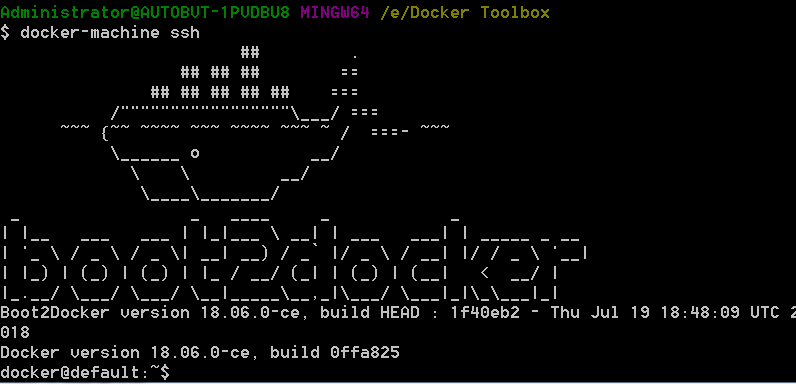
1、windows上**安装docker-toolbox，傻瓜安装即可。**

**2、**打开Docker Quickstart Terminal终端，等待初始始化完成后。

3、输入docker-machine env命令，返回docker服务的api接口和证书位置，如下：

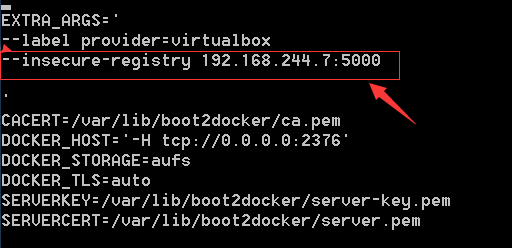


4、输入docker-machine ssh命令，进入sh环境中，配置http仓库路径



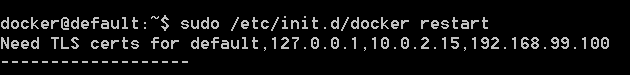
修改文件配置（当前用户是docker不是root，要sudo提升至root）：

sudo vi /var/lib/boot2docker/profile



5、修改完成，保存。重启docker服务

sudo /etc/init.d/docker restart



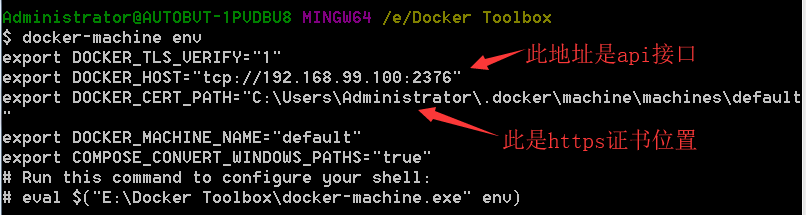
##### 项目环境配置maven插件

在我们的工程pom中加入docker-maven-plugin插件的配置，如下



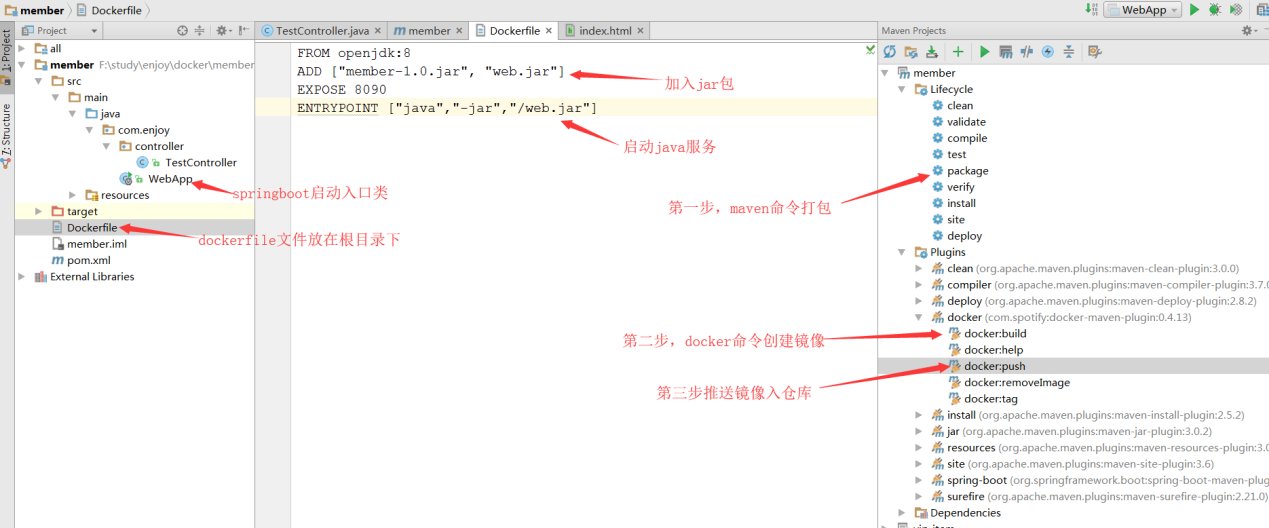
1、其中，imageName配置镜像的全路径名，即指定私库的名称

2、dockerHost和dockerCertPath对应配置上一步中docker的api和证书值

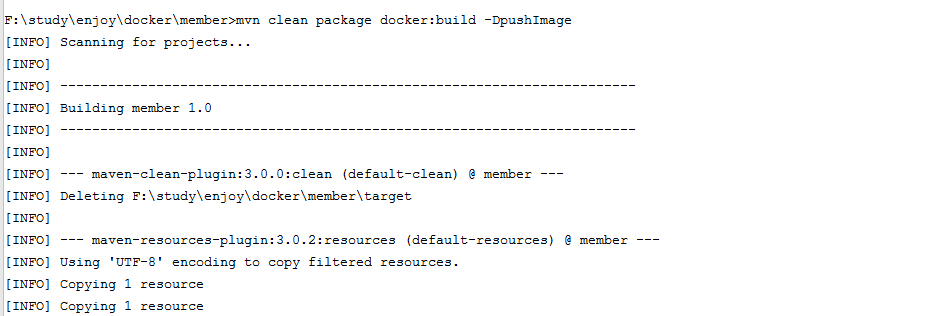


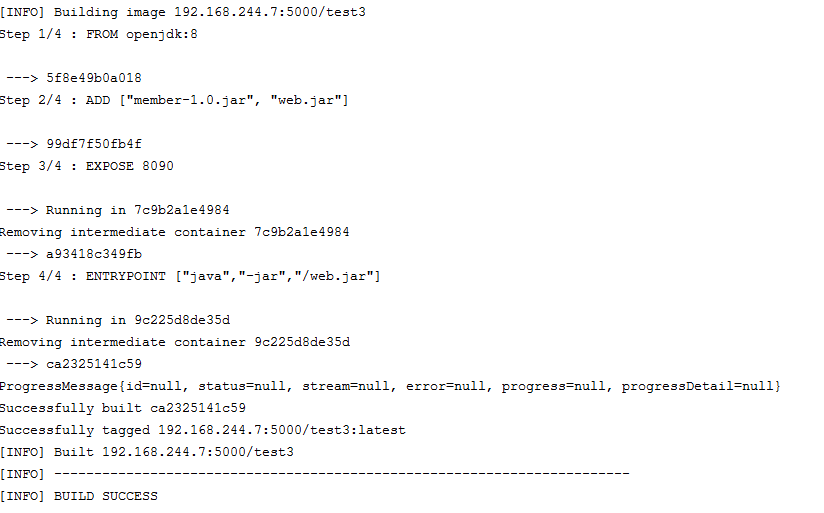
##### 打包运行

以idea为例，整个项目装配完成，只需要操作maven的一二三步骤，即直接镜像进入仓库，整个过程毫无违和感



若使用的不是idea工具，可直接使用maven命令，一句完成打包，如下：





##### 校验镜像仓库结果



至此，我们的服务器环境，已经可以直接运行docker run 镜像得到结果了

# Docker-Compose使用

当项目涉及容器较多时，需要一个管理容器的工具

## docker-compose安装

### curl方式安装

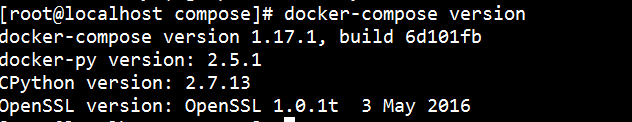
sudo curl -L https://github.com/docker/compose/releases/download/1.17.1/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

### 增加可执行权限

sudo chmod +x /usr/local/bin/docker-compose

### 查看版本

docker-compose version



## docker-compose.yaml命令

docker-compose的命令与docker命令极为相似，用法上没有区别，下面列出它特有的几种命令：

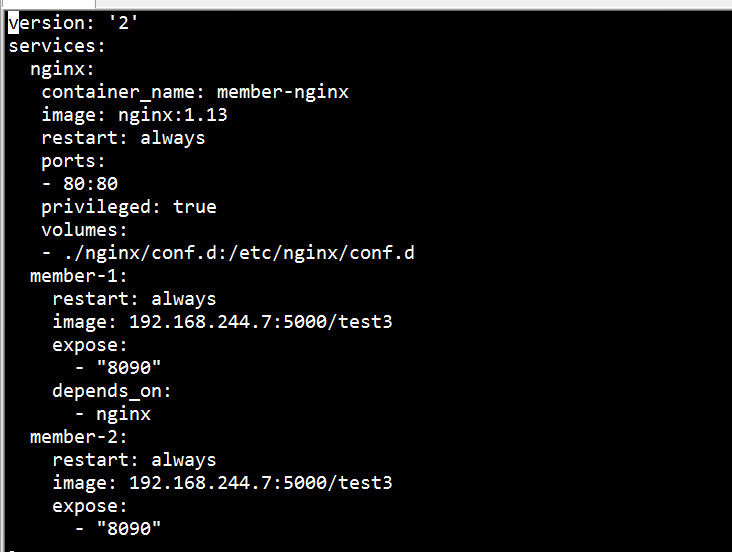
up 创建并启动容器：docker-compose up -d --scale 服务名=数字

---------- d表示后台运行，scale是表示对应的服务同时启动几个容器

down 停止并删除容器： docker-compose down

---------- 会停掉容器，并删除掉容器。如果不希望删除容器，请使用stop

## docker-compose实战

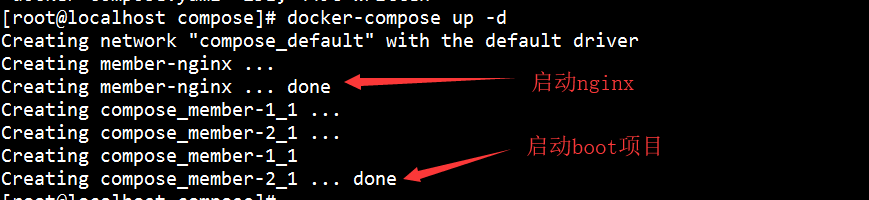


编写一个项目整体服务，一个网关nginx + springboot的集群，如上图

其中nginx服务，将配置文件挂载在主机当前项目目录的路径下：nginx/conf.d/

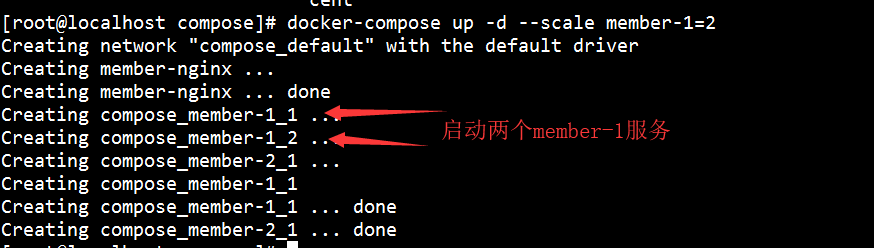


命令：docker-compose up -d



docker-compose up -d --scale member-1=2

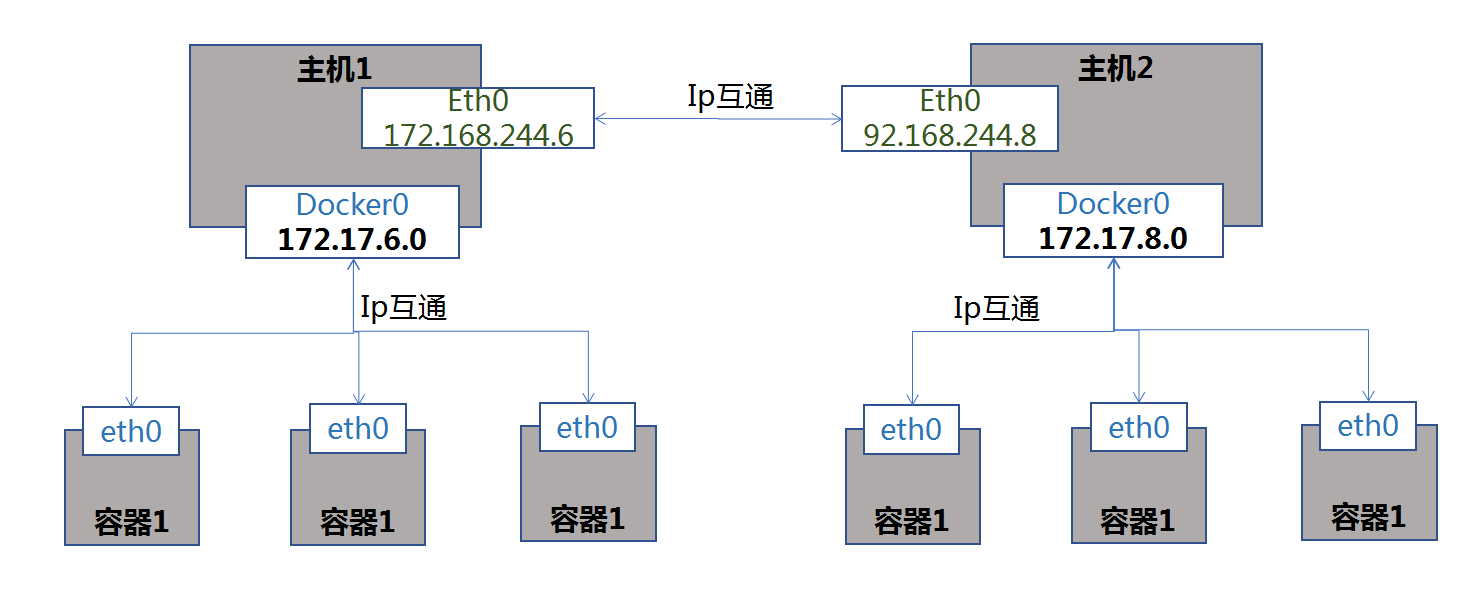
把member-1服务启动两个容器



# Docker网络路由

## docker的跨主机网络路由

假设我们现在有两台docker主机，各启动了自己的容器在运行



### 问题由来

1、在网桥模式下，同一个主机下的容器，使用同一个网桥docker0，它们组成一个局域网，如上图主机1的172.17.6.0网段下的三个容器

2、同一个主机下的容器，相互之间网络是通的

3、但不同主机下，是不同的局域网，它们之间网络不能互通。如：172.17.6.2的容器，想要访问172.17.8.2的容器

### 方案

a机192.168.244.7，容器网段172.17.6.1/16，a机起了容器ip是172.17.6.2

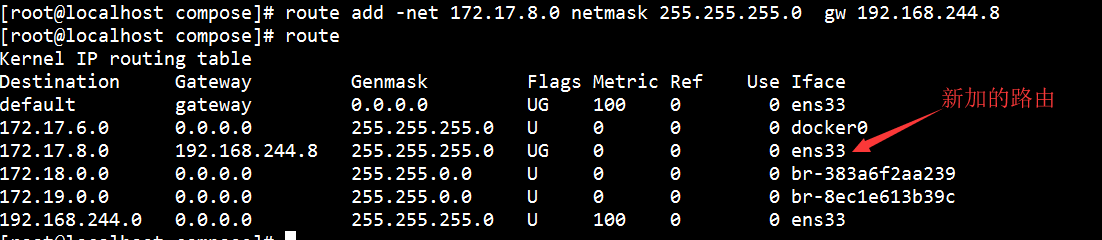
b机192.168.244.8，容器网段172.17.8.1/16，b机起了容器ip是172.17.8.2

#### 两台机分别配置路由表

a机，route add -net 172.17.8.0 netmask 255.255.255.0 gw 192.168.244.8

b机，route add -net 172.17.6.0 netmask 255.255.255.0 gw 192.168.244.7

添加好后，路由表类似下图

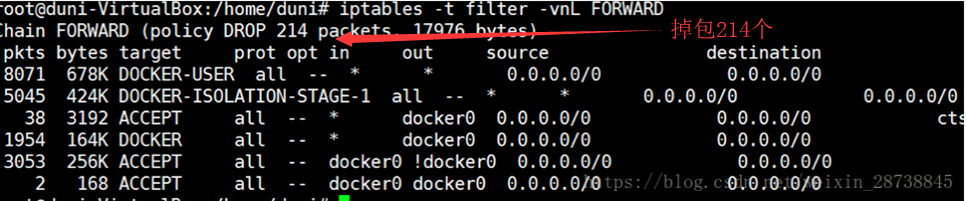


然后a机ping b机容器，发现仍是ping不通，卡住ping不通，就是数据包被drop掉了

#### ip\_forward配置

我们在b机上使用以下命令查看网络包转发情况，发现有掉包

iptables -t filter -nvL FORWARD



我们需要b机上配置，寻找172.17段ip的网络包不要丢掉，要转发

a机： iptables -I DOCKER  --dst 172.17.0.0/16 -j ACCEPT

b机： iptables -I DOCKER  --dst 172.17.0.0/16 -j ACCEPT

网络ok，整个网络包的流程，完整如下：

