Docker镜像服务部署帮助文档

武汉微创光电股份有限公司

2018年8月6日

版本历史信息

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 时间 | 作者 | 修改主要原因 |
| 1.0 | 2018-8-6 | 段文杰 | 文档创建 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1：项目介绍

帮助管理人员部署和配置socker镜像服务

2:软件安装列表

|  |  |
| --- | --- |
| 软件 | 说明 |
| CentOS-6.5-x86\_64-bin-DVD1.iso | 64位操作系统安装文件 |
| VMware-workstation-full-10.0.0-1295980.exe | 虚拟机客户端安装文件 |
| docker1.7.1离线安装依赖包及自动安装脚本 | device-mapper-libs-1.02.117-12.el6.x86\_64.rpm  docker-io-1.7.1-2.el6.x86\_64.rpm  libcgroup-0.40.rc1-23.el6.x86\_64.rpm  lua-alt-getopt-0.7.0-1.el6.noarch.rpm  lua-filesystem-1.4.2-1.el6.x86\_64.rpm  lua-lxc-1.0.9-1.el6.x86\_64.rpm  lxc-1.0.9-1.el6.x86\_64.rpm  lxc-libs-1.0.9-1.el6.x86\_64.rpm |

3 操作系统安装

3.1准备软件列表

|  |  |
| --- | --- |
| 软件 | 说明 |
| CentOS-6.5-x86\_64-bin-DVD1.iso | 64位操作系统安装文件 |
| VMware-workstation-full-10.0.0-1295980.exe | 虚拟机客户端安装文件 |

3.2详细启动歩骤

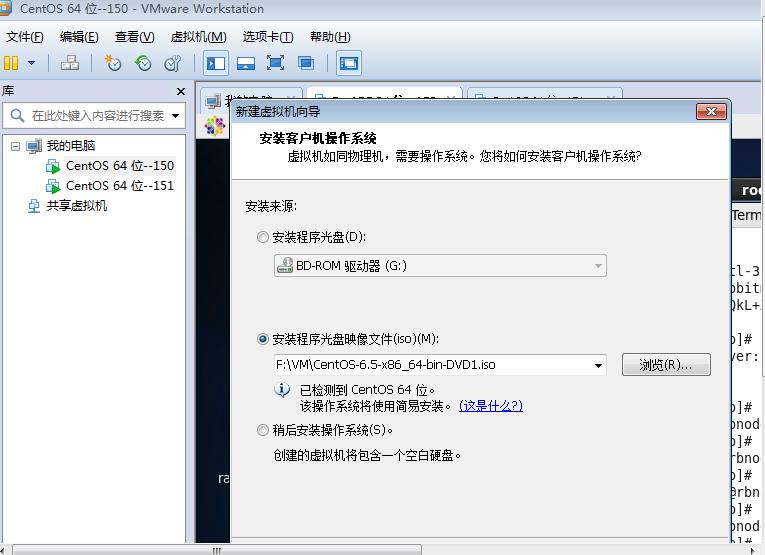
【sept1】打开新建虚拟机



【sept2】安装程序映像光盘iso



【sept3】选中iso文件



【sept4】简易安装信息



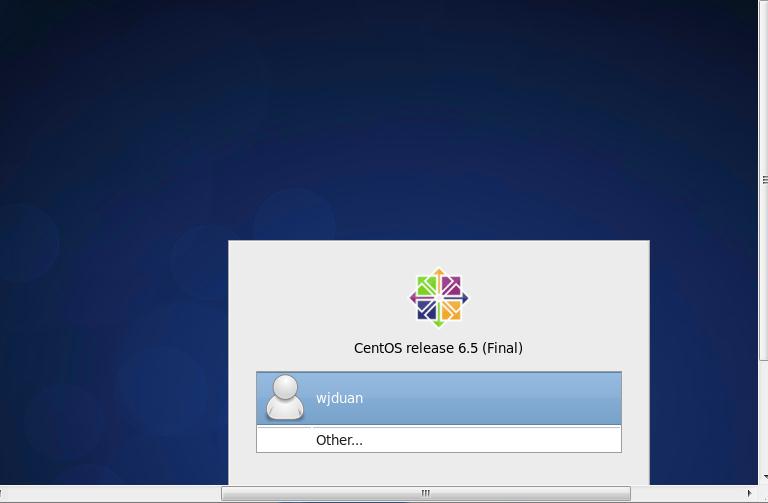
【sept5】指定安装位置



【sept6】安装过程中

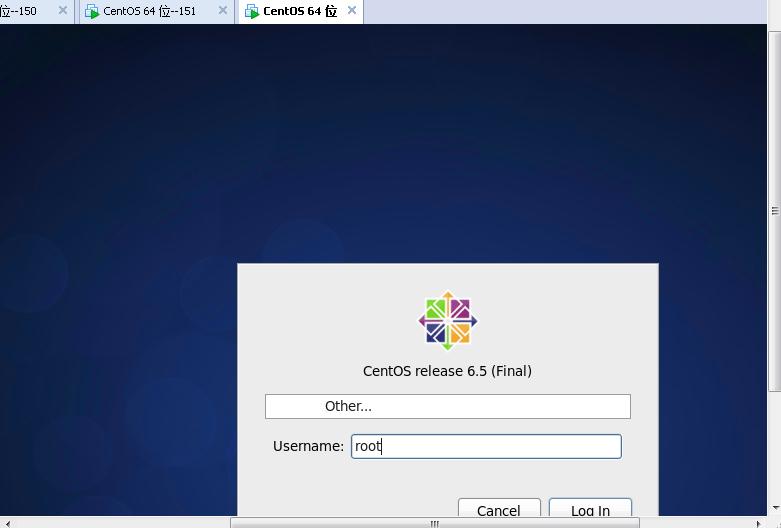


【sept7】安装完成

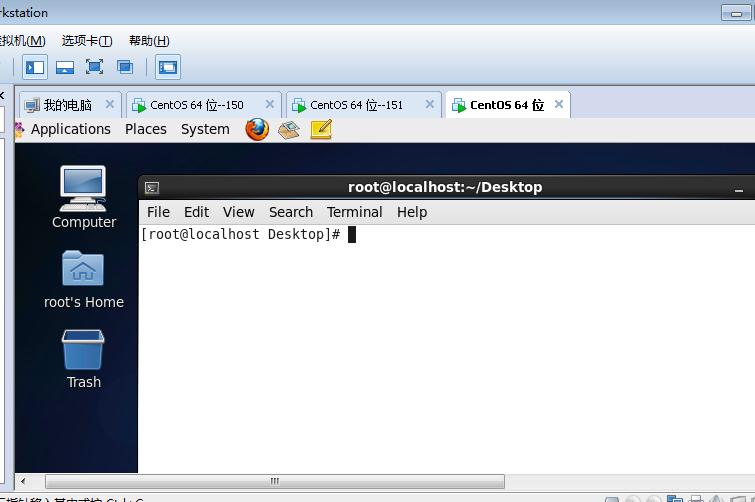


【sept8】点击other

使用root登录 密码：111111



【sept9】登录成功



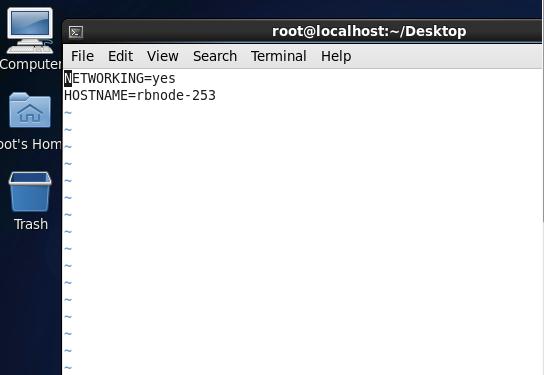
4：linux下静态IP设定

4.1：linux下设置静态IP

【sept1】修改主机名称

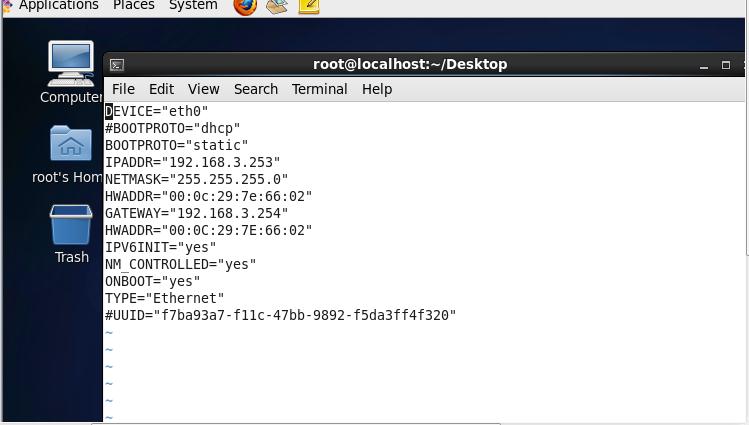
hostname=rbnode-253

命令：vi /etc/sysconfig/network



【sept2】配置静态ip地址

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



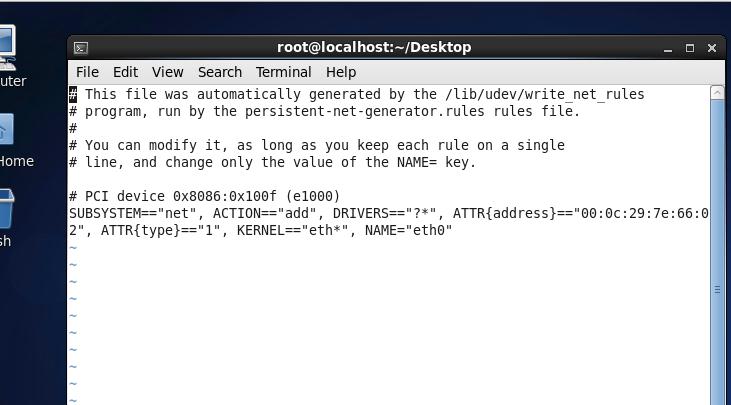
4.2：liunx下mac报错导致问题问题

Error:No suitable device found :no device found for connection ‘system eth0’

解决方案：由于网络的mac地址不同导致，修改mac即可

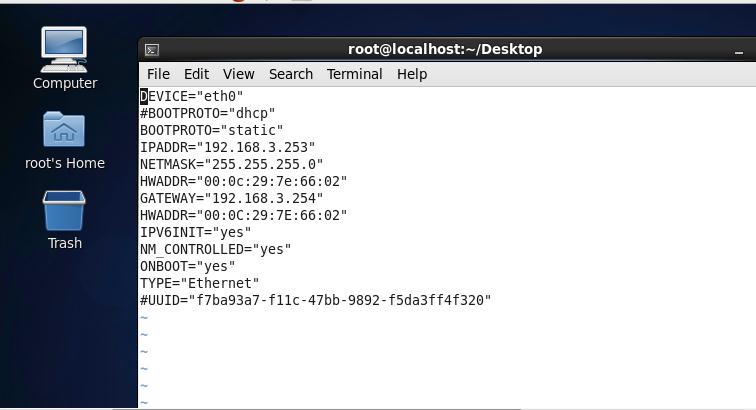
vim /etc/udev/rules.d/70-persistent-net.rules 找到里面的Attr{address}==”00:0c

:29:7e:66:02”



拷贝mac地址到ifcfg-eth0中

Vi /ect/sysconfig/network-script/ifcfg.eth0



测试是否链通ping 192.168.3.76

5：安装docker服务

5.1准备软件列表；

【sept0】文件存放路径:

[\\192.168.3.3\pis-publish\项目开发环境\Java\Java后台服务组件\docker内网部署及服务组件](\\\\192.168.3.3\\pis-publish\\项目开发环境\\Java\\Java后台服务组件\\docker内网部署及服务组件)

前提：拷贝软件到linux /home/tools/目录下;tools是本人新创建的；

|  |  |
| --- | --- |
| 软件 | 说明 |
| Linux centos 7.2 | Centos7.2 |
| Docker17.06.0-ce | Docker version 17.06.0-ce |
| Docker-compose-1.21.1 | Docker-compose-1.21.1 |
| Harbor-version：1.1.2 | Harbor-version：1.1.2 |
| centos7.2 --docker内网安装包在目录docker\_packages下 | audit-2.6.5-3.el7\_3.1.x86\_64.rpm  audit-libs-2.6.5-3.el7\_3.1.x86\_64.rpm  audit-libs-python-2.6.5-3.el7\_3.1.x86\_64.rpm  checkpolicy-2.5-4.el7.x86\_64.rpm  container-selinux-2.10-2.el7.noarch.rpm  docker-ce-17.06.0.ce-1.el7.centos.x86\_64.rpm  libcgroup-0.41-11.el7.x86\_64.rpm  libgudev1-219-30.el7\_3.7.x86\_64.rpm  libseccomp-2.3.1-2.el7.x86\_64.rpm  libselinux-2.5-6.el7.x86\_64.rpm  libselinux-python-2.5-6.el7.x86\_64.rpm  libselinux-utils-2.5-6.el7.x86\_64.rpm  libsemanage-2.5-5.1.el7\_3.x86\_64.rpm  libsemanage-python-2.5-5.1.el7\_3.x86\_64.rpm  libsepol-2.5-6.el7.x86\_64.rpm  libtool-ltdl-2.4.2-21.el7\_2.x86\_64.rpm  policycoreutils-2.5-11.el7\_3.x86\_64.rpm  policycoreutils-python-2.5-11.el7\_3.x86\_64.rpm  python-IPy-0.75-6.el7.noarch.rpm  selinux-policy-3.13.1-102.el7\_3.16.noarch.rpm  selinux-policy-targeted-3.13.1-102.el7\_3.16.noarch.rpm  setools-libs-3.3.8-1.1.el7.x86\_64.rpm  systemd-219-30.el7\_3.7.x86\_64.rpm  systemd-libs-219-30.el7\_3.7.x86\_64.rpm  systemd-python-219-30.el7\_3.7.x86\_64.rpm  systemd-sysv-219-30.el7\_3.7.x86\_64.rpm |

【sept1】安装docker环境

linux命令：

#cd /home/tools/docker\_packages/ #进入目录

#rpm -ivh \*.rpm --force --nodeps # 执行目录下所以的rpm程序

【sept2】授权docker

#docker ps //看到container id等字样代表安装成功

#设置开机启动项

#systemctl enable docker

#查看

#systemctl list-unit-files docker.service

【sept3】卸载docker环境

linux命令：

# yum list installed|grep docker #查看已安装的docker安装docker-ce.x86\_64 17.06.0.ce-1.el7.centos installed

#yum -y remove docker-ce.x86\_64 # 卸载

6：harbor私服仓库搭建

6.1准备软件列表；

前提：拷贝软件到linux /home/tools/目录下;tools是本人新创建的；

|  |  |
| --- | --- |
| 软件 | 说明 |
| docker-compose | docker-compose安装包 |
| harbor-offline-installer-v1.2.2.tgz | harbor-offline-installer-v1.2.2.tgz安装包 |

6.2harbor私服仓库搭建

【sept1】docker-compose服务安装

#拷贝docker-compose到/usr/local/bin/目录下

# chmod +x /usr/local/bin/docker-compose #授权

# docker-compose -version #安装

【sept2】关闭防火墙服务

# systemctl stop firewalld #关闭防火墙

# systemctl disable firewalld #禁止启动

【sept3】安装harbor服务

#拷贝harbor-offline-installer-v1.2.2.tgz到目录/home/tools/下

# tar -zxvf harbor-offline-installer-v1.2.2.tgz -C /usr/local/ #解压

#修改ip

#cd /usr/local/harbor

#vi harbor.cfg

解压缩之后，目录下会生成harbor.cfg文件，该文件就是Harbor的配置文件。

# 1. hostname设置访问地址，可以使用ip、域名，不可以设置为127.0.0.1或localhost# 2. 默认情况下，harbor使用的端口是80，若使用自定义的端口，除了要改docker-compose.yml文件中的配置外，# 这里的hostname也要加上自定义的端口，都在在docker login、push时会报错

hostname = 10.0.11.150:5000

# 访问协议，默认是http，也可以设置https，如果设置https，则nginx ssl需要设置on

ui\_url\_protocol = http

# mysql数据库root用户默认密码root123，实际使用时修改下

db\_password = root123

#Maximum number of job workers in job service

max\_job\_workers = 3

#The path of secretkey storage

secretkey\_path = /data

# 启动Harbor后，管理员UI登录的密码，默认是Harbor12345# 若修改了此处的admin登录密码。则登录后台时使用修改后的密码

harbor\_admin\_password = Harbor12345

# 认证方式，这里支持多种认证方式，如LADP、本次存储、数据库认证。默认是db\_auth，mysql数据库认证

auth\_mode = db\_auth

# 是否开启自注册

self\_registration = on

# Token有效时间，默认30分钟

token\_expiration = 30

# 用户创建项目权限控制，默认是everyone（所有人），也可以设置为adminonly（只能管理员）

project\_creation\_restriction = everyone

#修改端口

#vi docker-compose.yml

打开 docker-compose.yml（此处只摘录需要修改端口号的部分配置）：

proxy:

image: vmware/nginx:1.11.5-patched

container\_name: nginx

restart: always

volumes:

- ./common/config/nginx:/etc/nginx:z networks:

- harbor ports:

- 5000:80 # 修改端口号则修改此处，前面的位宿主机的端口，后面的是容器中的端口，只修改前面的即可 - 443:443 - 4443:4443 depends\_on:

- mysql - registry - ui - log logging:

driver: "syslog"

options:

syslog-address: "tcp://127.0.0.1:1514"



如上面配置中的注释所说，原来的端口映射为：80:80，此处我修改为5000:80，根据需要修改即可。****最重要的是，此处修改完之后，一定要将**harbor.conf**配置文件中的hostname一块修改下，改成：**ip:port**。ip就是部署**harbor**的机器IP，port就是这里设置的端口号。****

#访问url

[http://192.168.3.180:9000](http://192.168.3.180:5000)

#登录用户密码

admin Harbor12345

【sept3】harbor服务启动

# cd /usr/local/harbor #进入目录

# chmod +x install.sh #授权

# chmod +x prepare #授权

# ./install.sh #启动

#更改docker-compose.yml重新启动harbor

#docker-compose stop

#./install.sh

#重新启动harbor

#docker-compose down #关闭所有服务

#docker-compose start #开启所有服务

【sept4】修改docker客户端启动项

# vim /usr/lib/systemd/system/docker.service #修改

ExecStart=/usr/bin/dockerd --insecure-registry=192.168.3.180

# systemctl daemon-reload #重启守护进程

# systemctl restart docker #重启docker服务

【sept5】docker 登录并上传和下载镜像

#docker 需要配置--insecure-registry=192.168.3.180

#harbor网址需要创建用户，用户名：wenjie 密码：Duan678890

#harbor创建项目wjduan属性为公开

#docker login 192.168.3.180 -u admin -p Harbor12345 #登录

#docker push 192.168.3.180/wjduan/tomcat8080 #推送镜像

#docker push 192.168.3.180/library/tomcat8080 #上传镜像

#docker pull 192.168.3.180/library/tomcat8080 #下载镜像

7：tomcat镜像服务启动

7.1tomcat镜像服务搭建

【sept1】载入文件tomcat8080.tar

#save load --input /home/tools/loadMenu/tomcat8080.tar

#载入成功后，查看

#docker images

【sept2】启动tomcat容器服务

#默认bridge模式;带network namespace和ip、端口

#docker run -d -p 8080:8080 -v /home/tools/data/tomcat/webapps:/usr/local/tomcat/webapps -v /home/tools/data/tomcat/logs/:/usr/local/tomcat/logs/ --name jwzh\_tomcat-rm tomcat8080

#以host方式启动；以宿主机的ip和端口为准。容器不带ip和端口

#docker run --net=host -d -v /home/tools/data/tomcat/webapps:/usr/local/tomcat/webapps -v /home/tools/data/tomcat/logs/:/usr/local/tomcat/logs/ --name jwzh\_tomcat-rm tomcat8080

【sept3】生成文件tomcat8080.tar

# docker save -o /home/tools/saveMenu/tomcat8080.tar tomcat8080:latest

8：mysql镜像服务启动

8.1mysql镜像服务搭建

【sept1】载入文件mysql-jwzh.tar

#save load --input /home/tools/loadMenu/mysql-jwzh.tar

#载入成功后，查看

#docker images

【sept2】启动mysql容器服务 --ulimit nofile=65536:65536 解决最大连接为214问题

#docker run -d -p 3306:3306 -v /home/tools/data/mysql/:/var/lib/mysql/ -e MYSQL\_ROOT\_PASSWORD=123456 --ulimit nofile=65536:65536 --name mysql-jwzh-rm mysql-jwzh:latest

【sept3】生成文件mysql3306.tar

# docker save -o /home/tools/saveMenu/mysql3306.tar mysql3306:latest

【sept4】登录并创建user用户

# 创建用户密码

# CREATE USER 'user'@'localhost' IDENTIFIED BY '123456';

# 授权

# GRANT ALL PRIVILEGES ON \*.\* TO 'user'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;

#刷新权限

#flush privileges;

完成

9：redis镜像服务启动

9.1redis镜像服务搭建

【sept1】载入文件redis3.0.tar

# docker load --input /home/tools/tar包/redis3.0.tar

#载入成功后，查看

#docker images

【sept2】启动redis容器服务

#docker run -p 6379:6379 -v /home/tools/data/redis:/data -d --name redis-rm redis:3.0 redis-server --appendonly yes

【sept3】生成文件redis3.0.tar

# docker save -o /home/tools/saveMenu/redis3.0.tar redis:3.0

10：mongo镜像服务启动

10.1mongo镜像服务搭建

【sept1】载入文件mongo3.0.tar

# docker load --input /home/tools/tar包/mongo3.0.tar

#载入成功后，查看

#docker images

【sept2】启动mongo容器服务

#docker run -p 27017:27017 -v /home/tools/data/mongo/db:/data/db -d --name mongo-rm mongo:3.0.6

【sept3】生成文件mongo3.0.tar

# docker save -o /home/tools/saveMenu/mongo3.0.tar mongo:3.0.6

【sept4】进入容器创建用户密码和数据库

# docker exec -it 镜像id /bin/bash （进入容器）

# mongo （进入mongodb）

下面是安装完之后，正确的执行步骤，创建管理账户用户和普通用户的步骤，

创建管理账户，然后退出。

use admin  
 db.createUser(  
 {  
 user: "admin",  
 pwd: "password",  
 roles: [ { role: "root", db: "admin" } ]  
 }  
 );

use jwzh

db.createUser(

{

user: "user",

pwd: "123456",

roles: [ { role: "readWrite", db: "jwzh" } ]

}

);

exit;（退出容器）

11：rabbitmq镜像服务启动

11.1rabbitmq镜像服务搭建

【sept1】载入文件rabbitmq-3.6.1-management.tar

# docker load --input /home/tools/tar包/rabbitmq-3.6.1-management.tar

#载入成功后，查看

#docker images

【sept2】启动rabbitmq容器服务

#docker run -d --name myrabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3.6.1-management

【sept3】生成文件rabbitmq-3.6.1-management.tar

# docker save -o /home/tools/saveMenu/rabbitmq-3.6.1-management.tar rabbitmq:3.6.1-management

【sept4】默认帐号密码都为:guest

12：nginx镜像服务启动

12.1nginx镜像服务搭建

【sept1】载入文件nginx1.10.tar

# docker load --input /home/tools/tar包/nginx1.10.tar

#载入成功后，查看

#docker images

【sept2】启动nginx容器服务

#docker run -p 80:80 --name mynginx -v /home/tools/data/nginx-gridfs/www:/www -v /home/tools/data/nginx-gridfs/logs:/wwwlogs -d nginx:1.10.3

【sept3】生成文件nginx1.10.tar

# docker save -o /home/tools/saveMenu/nginx1.10.tar nginx:1.10.3

13：centos-nginx-gridfs镜像服务启动

13.1centos-nginx-gridfs镜像服务搭建

【sept1】载入文件centos-nginx-gridfs.tar

# docker load --input /home/tools/tar包/centos-nginx-gridfs.tar

#载入成功后，查看

#docker images

【sept2】启动centos-nginx-gridfs容器服务

#docker run -itd -p 80:80 --name centos-nginx-gridfs-rm nginx-gridfs-centos /home/auto\_service.sh

【sept3】生成文件centos-nginx-gridfs.tar

#docker save -o /home/tools/tar包/centos-nginx-gridfs.tar centos-nginx-gridfs

【sept4】步骤

#载入centos6.9.tar

#docker load --input /home/tools/centos6.9.tar

#启动容器

#docker run -it -name centos-nginx centos:6.9

#查容器 信息

# docker ps

#查容器ID对应的32位ID

# docker inspect -f '{{.Id}}' 4b414d90bb8b

#拷贝文件到mycentos容器中

#centos7 拷贝文件

#将主机的mongodb-linux-x86\_64-3.0.6.tgz 拷贝到容器4b414d90bb8b的/usr/local目录下：

#docker cp /home/tools/mongodb-linux-x86\_64-3.0.6.tgz 4b414d90bb8b:/usr/local/

# cp /home/tools/mongodb-linux-x86\_64-3.0.6.tgz /var/lib/docker/devicemapper/mnt/4b414d90bb8b2e1e52aa6229af5bf330a83e98abf9a39f34eaacf7b706048d29/rootfs/mnt

# cp /home/tools/nginx-gridfs-rpm.tar.gz /var/lib/docker/devicemapper/mnt/4b414d90bb8b2e1e52aa6229af5bf330a83e98abf9a39f34eaacf7b706048d29/rootfs/mnt

#进入容器

# docker exec -it centos-nginx /bin/bash

拷贝到usr的local下

# cp /mnt/nginx-gridfs.tar.gz /usr/local/

# cp /mnt/nginx-gridfs-rpm.tar.gz /usr/local/

#解压

#cd /usr/local/

#tar -zxvf nginx-gridfs.tar.gz

#tar -zxvf nginx-gridfs-rpm.tar.gz

#安装rpm

#rpm -ivh \*.rpm --force --nodeps

#授权

#cd /usr/local/nginx-gridfs/sbin/

#chmod 777 nginx

#nginx 启动测试

#/usr/local/nginx-gridfs/sbin/nginx -c /usr/local/nginx-gridfs/conf/nginx.conf

#设置自动重启

Vi /home/auto\_service.sh

#添加一下信息

#!/bin/sh

/usr/local/nginx-gridfs/sbin/nginx -c /usr/local/nginx-gridfs/conf/nginx.conf

/bin/bash

#保存授权：wq

Chmod 777 /home/auto\_service.sh

#根据容器打镜像

docker commit ce88f2ba643c centos-nginx-grifds

#运行容器带开机启动脚本

#docker run -itd -p 80:80 --name centos-nginx-gridfs2 centos-nginx-gridfs-v2 /home/auto\_service.sh

#启动脚本：

/home/auto\_service.sh

14：harbor搭建docker私有仓库

14.1harbor服务搭建

【sept1】安装Docker-compose

# 内网服务器环境(无法连接互联网)中安装docker-compose

在内网环境中，执行docker-compose安装程序并适时终止；即执行   
 $ curl -L <https://github.com/docker/compose/releases/download/1.1.0/doc> ker-compose-uname -s-uname -m` > /usr/local/bin/docker-compose   
 将外网环境中安装docker-compose后的安装文件/usr/local/bin/docker-compos e拷贝至内网环境中同路径下，并执行

$ chmod a+x /usr/local/bin/docker-compose   
 后执行docker-compose,执行结果如下，则安装成功。

$ docker-compose

【sept2】启动nginx容器服务

#docker run -p --net=host --name mynginx

-v /home/docker/volume/nginx-gridfs/www:/www

-v /home/docker/volume/nginx-gridfs/logs:/wwwlogs

-v /home/docker/volume/nginx-gridfs/conf/nginx.conf:/usr/local/nginx-gridfs/conf/nginx.conf

-d nginx-gridfs-centos

-v /docker/nginx/conf/nginx.conf:/etc/nginx/nginx.conf

【sept3】生成文件nginx1.10.tar

# docker save -o /home/tools/saveMenu/nginx1.10.tar nginx:1.10.3

pwd: "123456",

roles: [ { role: "readWrite", db: "jwzh" } ]

}

);

exit;

15：nginx-gridfs-mongodb镜像服务启动

15.1nginx-gridfs-mongodb镜像服务搭建

【sept1】载入文件nginx-gridfs-mongodb.tar

# docker load --input /home/tools/tar包/nginx-gridfs-mongodb.tar

#载入成功后，查看

#docker images

【sept2】启动nginx-gridfs-mongodb容器服务

#docker run --name=nginx-gridfs-mongodb-rm --net=host --restart=always -v /etc/localtime:/etc/localtime:ro -v /home/docker/data/nginx-gridfs/conf/nginx.conf:/usr/local/nginx-gridfs/conf/nginx.conf -v /home/docker/nginx-gridfs/logs:/usr/local/nginx-gridfs/logs -v /home/docker/data/mongodb/data/db:/usr/local/mongodb/data/db -v /home/docker/data/mongodb/logs:/usr/local/mongodb/logs -v /home/docker/data/mongodb/etc/mongodb.conf:/usr/local/mongodb/etc/mongodb.conf -itd nginx-gridfs-mongodb

#修改启动方式

#自动重启

#docker update --restart always nginx-gridfs-mongodb-rm

#不自动重启

#docker update --restart no nginx-gridfs-mongodb-rm

【sept3】生成文件nginx-gridfs-mongodb.tar

# docker save -o /home/tools/tar包/nginx-gridfs-mongodb.tar gridfs-mongodb-images

【sept4】配置文件目录

#/home/docker/volume/mongodb

#/home/docker/volume/nginx-gridfs

#存放路径