# CentOS Docker 安装

Docker支持以下的CentOS版本：

* CentOS 7 (64-bit)
* CentOS 6.5 (64-bit) 或更高的版本

## 前提条件

目前，CentOS 仅发行版本中的内核支持 Docker。

Docker 运行在 CentOS 7 上，要求系统为64位、系统内核版本为 3.10 以上。

Docker 运行在 CentOS-6.5 或更高的版本的 CentOS 上，要求系统为64位、系统内核版本为 2.6.32-431 或者更高版本。

## 使用 yum 安装（CentOS 7下）

Docker 要求 CentOS 系统的内核版本高于 3.10 ，查看本页面的前提条件来验证你的CentOS 版本是否支持 Docker 。

通过 **uname -r**命令查看你当前的内核版本

[root@server1 ~]# uname -r



### **安装 Docker**

从 2017 年 3 月开始 docker 在原来的基础上分为两个分支版本: Docker CE 和 Docker EE。

Docker CE 即社区免费版，Docker EE 即企业版，强调安全，但需付费使用。

本文介绍 Docker CE 的安装使用。

移除旧的版本：

$ sudo yum remove docker \

docker-client \

docker-client-latest \

docker-common \

docker-latest \

docker-latest-logrotate \

docker-logrotate \

docker-selinux \

docker-engine-selinux \

docker-engine

安装一些必要的系统工具：

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

添加软件源信息

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

wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

更新 yum 缓存：

yum makecache fast

安装 Docker-ce：

yum -y install docker-ce

启动 Docker 后台服务

systemctl start docker

systemctl status docker

systemctl enable docker

docker疑难杂症：docker命令Tab无法自动补全

一、安装bash-complete

yum install -y bash-completion

二、刷新文件

source /usr/share/bash-completion/completions/docker

source /usr/share/bash-completion/bash\_completion

查看版本信息

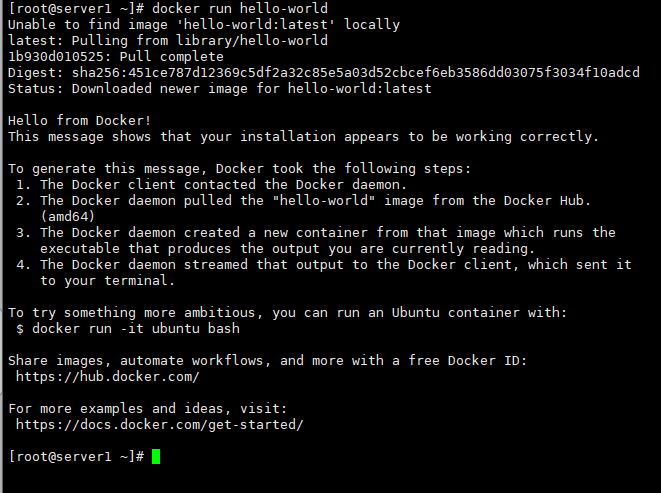
# docker version

# docker info

测试运行 hello-world

[root@server1 ~]# docker run hello-world --运行 hello-world 这个 images，如果本地没有，则会从

docker hub 去下载;



由于本地没有hello-world这个镜像，所以会下载一个hello-world的镜像，并在容器内运行。

## 使用脚本安装 Docker

1、使用 sudo 或 root 权限登录 Centos。

2、确保 yum 包更新到最新。

$ sudo yum update

3、执行 Docker 安装脚本。

$ curl -fsSL https://get.docker.com -o get-docker.sh

$ sudo sh get-docker.sh

执行这个脚本会添加 docker.repo 源并安装 Docker。

4、启动 Docker 进程。

sudo systemctl start docker

5、验证 docker 是否安装成功并在容器中执行一个测试的镜像。

$ sudo docker run hello-world

docker ps

到此，Docker 在 CentOS 系统的安装完成。

## 镜像加速

鉴于国内网络问题，后续拉取 Docker 镜像十分缓慢，我们可以需要配置加速器来解决，我使用的是网易的镜像地址：**http://hub-mirror.c.163.com**。

新版的 Docker 使用 /etc/docker/daemon.json（Linux） 或者 %programdata%\docker\config\daemon.json（Windows） 来配置 Daemon。

请在该配置文件中加入（没有该文件的话，请先建一个）：

{

"registry-mirrors": ["http://hub-mirror.c.163.com"]}

## 删除 Docker CE

执行以下命令来删除 Docker CE：

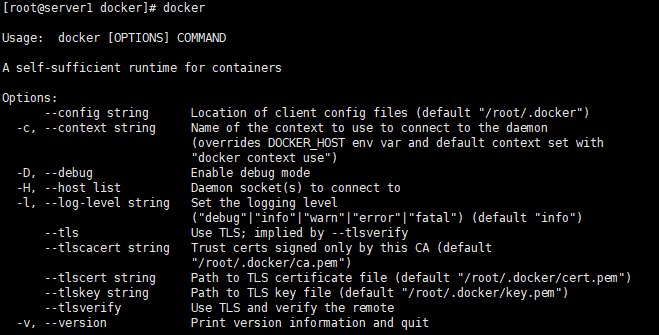
$ sudo yum remove docker-ce

$ sudo rm -rf /var/lib/docker

## Docker 客户端

docker 客户端非常简单 ,我们可以直接输入 docker 命令来查看到 Docker 客户端的所有命令选项。

[root@server1 docker]# docker



可以通过命令 **docker command --help** 更深入的了解指定的 Docker 命令使用方法。

例如我们要查看 **docker stats** 指令的具体使用方法：

docker stats --help

## 运行一个web应用

使用 docker 构建一个 web 应用程序。

在docker容器中运行一个 Python Flask 应用来运行一个web应用

[root@server1 ~]# docker pull training/webapp # 载入镜像

[root@server1 ~]# docker run -d -P training/webapp python app.py



参数说明

**-d:**让容器在后台运行。

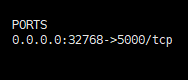
**-P:**将容器内部使用的网络端口映射到我们使用的主机上。

## 查看 WEB 应用容器

使用 docker ps 来查看我们正在运行的容器：

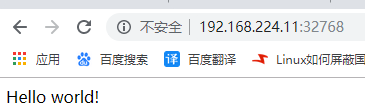


这里多了端口信息。



Docker 开放了 5000 端口（默认 Python Flask 端口）映射到主机端口 32768 上。

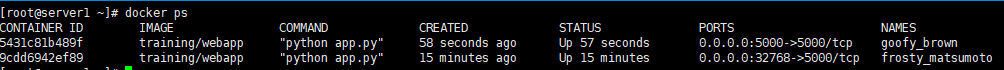
这时我们可以通过浏览器访问WEB应用



我们也可以通过 -p 参数来设置不一样的端口：

[root@server1 ~]# docker run -d -p 5000:5000 training/webapp python app.py

**docker ps**查看正在运行的容器



容器内部的 5000 端口映射到我们本地主机的 5000 端口上。

## 网络端口的快捷方式

通过 **docker ps** 命令可以查看到容器的端口映射，**docker** 还提供了另一个快捷方式 **docker port**，使用 **docker port** 可以查看指定 （ID 或者名字）容器的某个确定端口映射到宿主机的端口号。

上面我们创建的 web 应用容器 ID 为 **5431c81b489f** 名字为goofy\_brown。

我可以使用 **docker port 5431c81b489f** 或 **docker port goofy\_brown**来查看容器端口的映射情况。

[root@server1 ~]# docker port 5431c81b489f

5000/tcp -> 0.0.0.0:5000

[root@server1 ~]# docker port goofy\_brown

5000/tcp -> 0.0.0.0:5000

## 查看 WEB 应用程序日志

docker logs [ID或者名字] 可以查看容器内部的标准输出。

[root@server1 ~]# docker logs -f 5431c81b489f

\* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

192.168.224.1 - - [24/Aug/2019 16:31:27] "GET / HTTP/1.1" 200 -

192.168.224.1 - - [24/Aug/2019 16:31:34] "GET / HTTP/1.1" 200 -

192.168.224.1 - - [24/Aug/2019 16:31:34] "GET /favicon.ico HTTP/1.1" 404 -

192.168.224.1 - - [24/Aug/2019 16:31:36] "GET / HTTP/1.1" 200 -

192.168.224.1 - - [24/Aug/2019 16:31:36] "GET / HTTP/1.1" 200 -

**-f:** 让 **docker logs** 像使用 **tail -f** 一样来输出容器内部的标准输出。

从上面，我们可以看到应用程序使用的是 5000 端口并且能够查看到应用程序的访问日志。

## 查看WEB应用程序容器的进程

我们还可以使用 docker top 来查看容器内部运行的进程

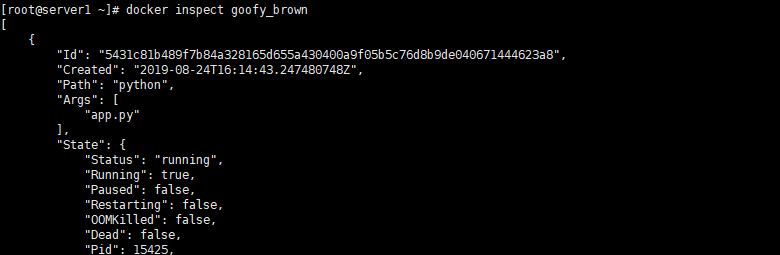
[root@server1 ~]# docker top goofy\_brown



## 检查 WEB 应用程序

使用 **docker inspect** 来查看 Docker 的底层信息。它会返回一个 JSON 文件记录着 Docker 容器的配置和状态信息。

[root@server1 ~]# docker inspect goofy\_brown



## 停止 WEB 应用容器

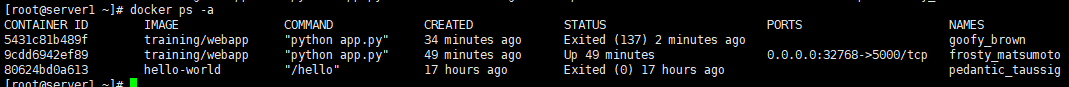
[root@server1 ~]# docker stop goofy\_brown

goofy\_brown

## 重启WEB应用容器

已经停止的容器，我们可以使用命令 docker start 来启动

docker ps -a 来查看所有容器



然后 找到需要重启的Id或名字

[root@server1 ~]# docker start goofy\_brown

goofy\_brown

现在容器又启动起来了，



docker ps -l 查询最后一次创建的容器：

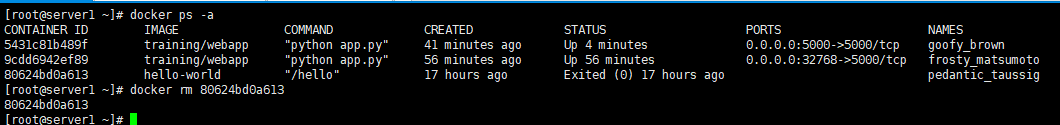


正在运行的容器，我们可以使用 docker restart 命令来重启

## 移除WEB应用容器

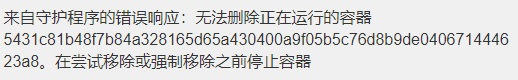
我们可以使用 docker rm 命令来删除不需要的容器

删除第一个hello-world容器



删除容器时，容器必须是停止状态，否则会报如下错误





# Docker 镜像使用

当运行容器时，使用的镜像如果在本地中不存在，docker 就会自动从 docker 镜像仓库中下载，默认是从 Docker Hub 公共镜像源下载。

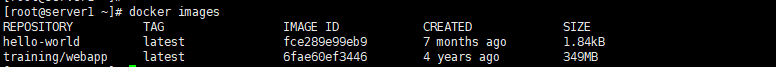
下面我们来学习：

* 1、管理和使用本地 Docker 主机镜像
* 2、创建镜像

## 列出镜像列表

我们可以使用 **docker images** 来列出本地主机上的镜像

[root@server1 ~]# docker images



各个选项说明:

**REPOSITORY：**表示镜像的仓库源

**TAG：**镜像的标签

**IMAGE ID：**镜像ID

**CREATED：**镜像创建时间

**SIZE：**镜像大小

同一仓库源可以有多个 TAG，代表这个仓库源的不同个版本，如ubuntu仓库源里，有15.10、14.04等多个不同的版本，我们使用 REPOSITORY:TAG 来定义不同的镜像。

所以，我们如果要使用版本为15.10的ubuntu系统镜像来运行容器时，命令如下：

docker run -t -i ubuntu:15.10 /bin/bash

如果你不指定一个镜像的版本标签，例如你只使用 ubuntu，docker 将默认使用 ubuntu:latest 镜像。

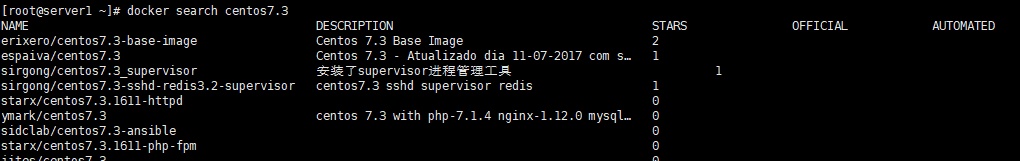
## 获取一个新的镜像

当我们在本地主机上使用一个不存在的镜像时 Docker 就会自动下载这个镜像。如果我们想预先下载这个镜像，我们可以使用 docker pull 命令来下载它。

先通过docker search 来查询

# docker search centos7.3 --通过 centos7.3 去 docker.io 上查找相关的 images;使用

man docker-search 得到参数说明



# docker pull docker.io/erixero/centos7.3-base-image --通过 pull 命令下载

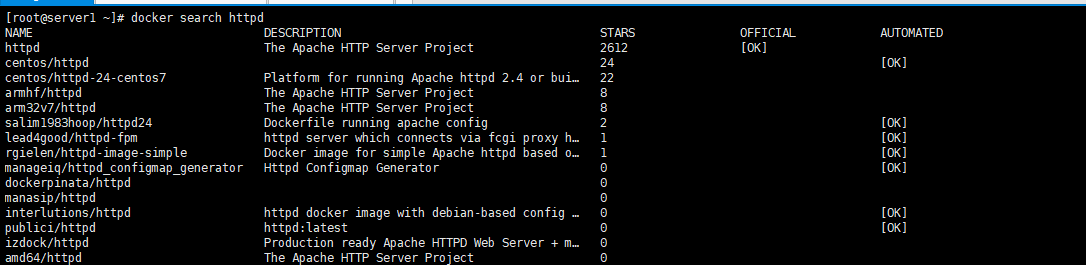
images,后面的就是 image 的名称(前提网速要好,此镜像大概 200 多 M);使用 man docker-pull

得到参数说明

## 查找镜像

我们可以从 Docker Hub 网站来搜索镜像，Docker Hub 网址为： https://hub.docker.com/

我们也可以使用 docker search 命令来搜索镜像。比如我们需要一个httpd的镜像来作为我们的web服务。我们可以通过 docker search 命令搜索 httpd 来寻找适合我们的镜像。



**NAME:**镜像仓库源的名称

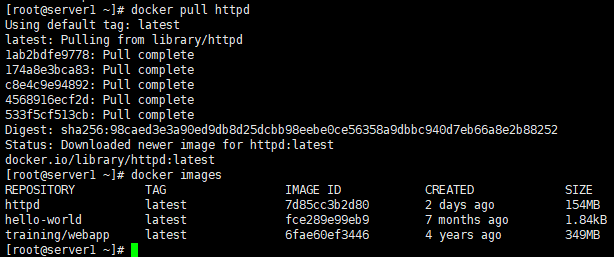
**DESCRIPTION:**镜像的描述

**OFFICIAL:**是否docker官方发布

## 拖取镜像

我们决定使用上图中的httpd 官方版本的镜像，使用命令 docker pull 来下载镜像。

[root@server1 ~]# docker pull httpd



下载完成后，我们就可以使用这个镜像了

[root@server1 ~]# docker run -d httpd

## 创建镜像

当我们从docker镜像仓库中下载的镜像不能满足我们的需求时，我们可以通过以下两种方式对镜像进行更改。

* 1.从已经创建的容器中更新镜像，并且提交这个镜像
* 2.使用 Dockerfile 指令来创建一个新的镜像

## 更新镜像

更新镜像之前，我们需要使用镜像来创建一个容器

[root@server1 ~]# docker pull erixero/centos7.3-base-image 下载镜像

[root@server1 ~]# docker run -t -i erixero/centos7.3-base-image /bin/bash

--使用 man

docker-run 得到参数说明;使用 exit 或 ctrl+d 来退出

其中，-t 选项让Docker分配一个伪终端（pseudo-tty）并绑定到容器的标准输入上，-i 则让容器的标准输入保持打开。

当利用 docker run 来创建容器时，Docker 在后台运行的标准操作包括：   
（1）检查本地是否存在指定的镜像，不存在就从公有仓库下载   
（2）利用镜像创建并启动一个容器   
（3）分配一个文件系统，并在只读的镜像层外面挂载一层可读写层   
（4）从宿主主机配置的网桥接口中桥接一个虚拟接口到容器中去   
（5）从地址池配置一个 ip 地址给容器   
（6）执行用户指定的应用程序   
（7）执行完毕后容器被终止

在运行的容器内使用 yum update 命令进行更新。

bash-4.2# yum update

在完成操作之后，输入 exit命令来退出这个容器。

docker ps -l 查看最后一次创建的容器



此时ID为4af25ccf22e6的容器，是按我们的需求更改的容器。我们可以通过命令 docker commit来提交容器副本。

[root@server1 ~]# docker commit -m="vim update lsof" -a="look" 4af25ccf22e6 look/centos7.3:1



各个参数说明：

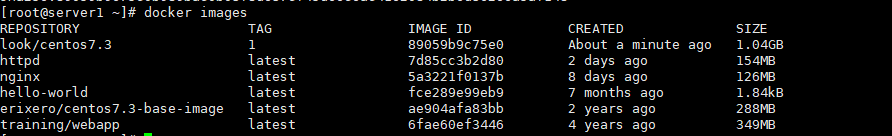
**-m:**提交的描述信息

**-a:**指定镜像作者

**e218edb10161：**容器ID

**runoob/ubuntu:v2:**指定要创建的目标镜像名

我们可以使用 **docker images** 命令来查看我们的新镜像 **look/centos7.3:1**



使用我们的新镜像**look/centos7.3** 来启动一个容器

[root@server1 ~]# docker run -i -t look/centos7.3:1 /bin/bash

现在之前安装的软件可以用了；

bash-4.2# lsof -i:80

bash-4.2# lsof -i:22

bash-4.2# vim /etc/hosts

bash-4.2#

这里交互操作时，你可以在里面 touch 创建一个文件，退出后，再查看，会发现创建的文件没

了(原因是你退出再登录，是又产生了一个新的 container)

# docker ps -a --可以查看到前面所有运行或退出状态的 container

# docker logs e28478d4b604 --查看此 container id 的结果；id 可以换成名称;使用 man

docker-logs 得到参数说明

问题一:

前面我们交互连到 images 里，touch 的文件，退出再连就看不到了（因为退出再连又产生了一个

新的容器），如何让 touch 的文件还能看到?

答案:要继续连接先前的容器

# docker run -i -t docker.io/erixero/centos7.3-base-image /bin/bash

bash-4.2# touch /1111111

bash-4.2# ls /

1111111 bin boot dev etc home lib lib64 media mnt opt proc root

run sbin srv sys tmp usr var

bash-4.2# exit

# docker ps -l (Find id of instance) docker ps -l 查询最后一次创建的容器：

CONTAINER ID IMAGE COMMAND

CREATED STATUS PORTS NAMES

fe2ff7a78f55 docker.io/erixero/centos7.3-base-image "/bin/bash"

49 seconds ago Exited (0) 19 seconds ago

thirsty\_meninsky

# docker attach fe2ff7a78f55 --使用 docker attach 可以连接上一个运行中的容器，

停止状态的需要先 start; man docker-attach 查看更多使用参数说明

You cannot attach to a stopped container, start it first

# docker start fe2ff7a78f55

# docker attach fe2ff7a78f55 --连上去后，可以看到先前 touch 的 111111 这个文件

bash-4.2# ls /

1111111 bin boot dev etc home lib lib64 media mnt opt proc root

run sbin srv sys tmp usr var

bash-4.2#

扩展:

删除一个 container

#docker stop e28478d4b604

#docker rm e28478d4b604

停止所有的 container

#docker stop $(docker ps -a -q) （-q参数用于仅仅显示ID）

如果想要删除所有 container 的话再加一个指令

#docker rm $(docker ps -a -q)

删除一个镜像（先要把此镜像相关的容器都删掉，才能再删除镜像)

#docker rmi 镜像名

## 构建镜像

我们使用命令 **docker build** ， 从零开始来创建一个新的镜像。为此，我们需要创建一个 Dockerfile 文件，其中包含一组指令来告诉 Docker 如何构建我们的镜像。

[root@server1 ~]# cat Dockerfile

FROM centos:6.7

MAINTAINER Fisher "fisher@sudops.com"

RUN /bin/echo 'root:123456' |chpasswd

RUN useradd runoob

RUN /bin/echo 'runoob:123456' |chpasswd

RUN /bin/echo -e "LANG=\"en\_US.UTF-8\"" >/etc/default/local

EXPOSE 22

EXPOSE 80

CMD /usr/sbin/sshd -D

每一个指令都会在镜像上创建一个新的层，每一个指令的前缀都必须是大写的。

第一条FROM，指定使用哪个镜像源

RUN 指令告诉docker 在镜像内执行命令，安装了什么。。。

然后，我们使用 Dockerfile 文件，通过 docker build 命令来构建一个镜像。

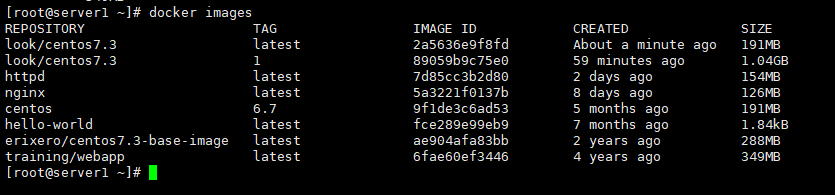
[root@server1 ~]# docker build -t look/centos7.3 .

参数说明：

**-t** ：指定要创建的目标镜像名

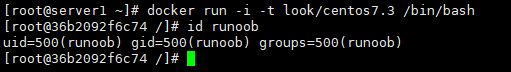
**.** ：Dockerfile 文件所在目录，可以指定Dockerfile 的绝对路径

使用docker images 查看创建的镜像已经在列表中存在,镜像ID为2a5636e9f8fd



我们可以使用新的镜像来创建容器

[root@server1 ~]# docker run -i -t look/centos7.3 /bin/bash



从上面看到新镜像已经包含我们创建的用户runoob

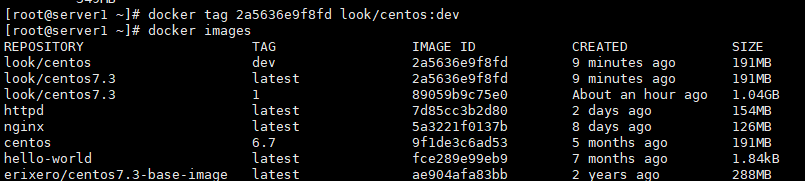
## 设置镜像标签

我们可以使用 docker tag 命令，为镜像添加一个新的标签

[root@server1 ~]# docker tag 2a5636e9f8fd look/centos:dev

docker tag 镜像ID，这里是  2a5636e9f8fd ,用户名称、镜像源名(repository name)和新的标签名(tag)。

使用 docker images 命令可以看到，ID为860c279d2fec的镜像多一个标签。



# Docker 容器连接

前面我们实现了通过网络端口来访问运行在 docker 容器内的服务。下面我们来实现通过端口连接到一个 docker 容器

## 网络端口映射

创建了一个 look/centos7.3应用的容器。

[root@server1 ~]# docker run -d -i -t -P look/centos7.3:1 /bin/bash

我们也可以使用 **-p** 标识来指定容器端口绑定到主机端口。

两种方式的区别是:

**-P :**是容器内部端口**随机**映射到主机的高端口。

**-p :**是容器内部端口绑定到**指定**的主机端口。

[root@server1 ~]# docker run -d -i -t -p 2222:22 look/centos7.3:1 /bin/bash

另外，我们可以指定容器绑定的网络地址，比如绑定 127.0.0.1。

这样我们就可以通过访问 127.0.0.1:5001 来访问容器的 5000 端口。

上面的例子中，默认都是绑定 tcp 端口，如果要绑定 UDP 端口，可以在端口后面加上 **/udp**。

## Docker容器连接

端口映射并不是唯一把 docker 连接到另一个容器的方法。

docker 有一个连接系统允许将多个容器连接在一起，共享连接信息。

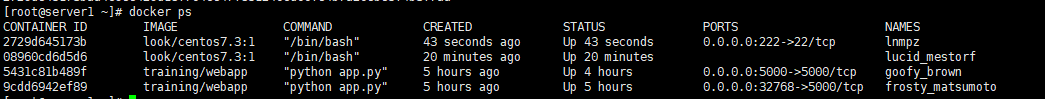
docker 连接会创建一个父子关系，其中父容器可以看到子容器的信息。

### **容器命名**

当我们创建一个容器的时候，docker 会自动对它进行命名。另外，我们也可以使用 **--name** 标识来命名容器，例如：

[root@server1 ~]# docker run -d -i -t -p 222:22 --name lnmpz look/centos7.3:1 /bin/bash

我们可以使用 **docker ps** 命令来查看容器名称。



可以使用下面命令来查看CentOS版本信息：

$ cat /etc/redhat-release

**修改root密码**

使用passwd密码来修改密码（如提示没有这个命令行使用yum install passwd安装）：

$ passwd    root 密码    确认密码123

passwd 修改用户密码

passwd a 修改a用户密码

**安装Openssh**

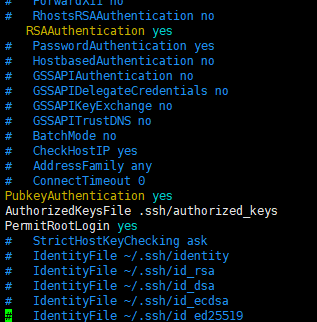
使用下面命令安装ssh server/ssh client：

$ sudo yum -y install openssh-server$ sudo yum -y install openssh-clients

修改SSH配置文件以下选项，去掉#注释，将四个选项启用：

$ vi /etc/ssh/sshd\_config RSAAuthentication yes #启用 RSA 认证PubkeyAuthentication yes #启用公钥私钥配对认证方式AuthorizedKeysFile .ssh/authorized\_keys #公钥文件路径（和上面生成的文件同）

PermitRootLogin yes #root能使用ssh登录123



重启ssh服务，并设置开机启动：

$ service sshd restart$ chkconfig sshd on

退出容器并保存更改

使用exit命令或者ctrl+C来退出当前运行的容器：

**打包镜像保存**

docker ps -l 查看最后一次创建的容器



此时ID为4af25ccf22e6的容器，是按我们的需求更改的容器。我们可以通过命令 docker commit来提交容器副本。

[root@server1 ~]# docker commit -m="vim update lsof" -a="look" 4af25ccf22e6 look/centos7.3:1



各个参数说明：

**-m:**提交的描述信息

**-a:**指定镜像作者

4af25ccf22e6**：**容器ID

look/centos7.3:1**:**指定要创建的目标镜像名

我们可以使用 **docker images** 命令来查看我们的新镜像 **look/centos7.3:1**

****docker save :**将指定镜像保存成 tar 归档文件。**

### **语法**

docker save [OPTIONS] IMAGE [IMAGE...]

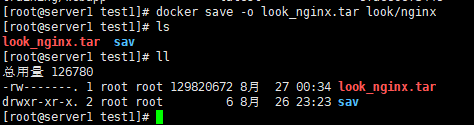
OPTIONS 说明：

**-o :**输出到的文件。

### **实例**

将镜像 look/nginx 生成 look\_nginx.tar 文档

docker save -o look\_nginx.tar look/nginx



****docker load :**导入使用 [docker save](https://www.runoob.com/docker/docker-save-command.html" \t "https://www.runoob.com/docker/_blank) 命令导出的镜像。**

### **语法**

docker load [OPTIONS]

OPTIONS 说明：

**-i :**指定导出的文件。

**-q :**精简输出信息。

### **实例**

导入镜像：

docker load -i look\_nginx.tar

docker load < look\_nginx.tar

两种都可以导入

****docker import :** 从归档文件中创建镜像。**

### **语法**

docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]

OPTIONS说明：

**-c :**应用docker 指令创建镜像；

**-m :**提交时的说明文字；

### **实例**

从镜像归档文件my\_ubuntu\_v3.tar创建镜像，命名为runoob/ubuntu:v4

runoob@runoob:~$ docker import my\_ubuntu\_v3.tar runoob/ubuntu:v4

sha256:63ce4a6d6bc3fabb95dbd6c561404a309b7bdfc4e21c1d59fe9fe4299cbfea39

runoob@runoob:~$ docker images runoob/ubuntu:v4

REPOSITORY TAG IMAGE ID CREATED SIZE

runoob/ubuntu v4 63ce4a6d6bc3 20 seconds ago 142.1 MB

dock run -it 镜像的名称：镜像的标签   【启动命令】  （启动镜像）  
[root@node docker\_images]# docker run -it --privileged=true centos:latest /bin/bash    
--privileged=true  开启容器的时候加入这个参数 #大约在0.6版，privileged被引入docker。使用该参数，container内的root拥有真正的root权限。  
否则，container内的root只是外部的一个普通用户权限。  
privileged启动的容器，可以看到很多host上的设备，并且可以执行mount。  
甚至允许你在docker容器中启动docker容器。  
latest标签    （默认标签）

命令  
docker run      //运行容器  
  
-i  交互式  
-t  终端  
-d  后台进程  
[root@node docker\_images]# docker run -d nginx  (启动一个后台服务)  
[root@node docker\_images]# docker run -itd centos (启动一个交互式的后台进程)  
systmd   pid为1   上帝进程   这就是系统，系统就是它 ，没有父进程   
  
docker ps        //查看容器列表  
[root@node docker\_images]# docker ps -aq  (查看系统所有的容器ID)  
  
docker stop   //关闭容器  
[root@node docker\_images]# docker stop 2c0cfd654026  
  
docker start   //启动容器  
[root@node docker\_images]# docker start 2c0cfd654026  
  
docker restart   //重启容器  
[root@node docker\_images]# docker restart 2c0cfd654026  
  
docker attach|exec  //进入容器  
[root@node docker\_images]# docker exec -it 7fc65424bdc2 /bin/bash  
[root@node docker\_images]# docker attach 7fc65424bdc2 (最好不要使用这种方式，进入init为1的进程容器，如果退出则销毁容器退出,ctrl+p,q退出并不销毁容器，attach可以用排错)  
  
  
docker inspect     //查看容器底层信息  
[root@node docker\_images]# docker inspect 2c0cfd654026  
注：主要查看ip地址  
  
deocker top     //查看容器进程列表  
[root@node docker\_images]# docker top 7fc65424bdc2  (容器必须要正在运行中才能查询)  
      
docker rm    //删除容器  
[root@node docker\_images]# docker rm $(docker ps -aq)  (后面的结果当成前面的参数,删除所有运行过的容器)  
  
注：容器与真机是共享的，隔离性很差