**[docker基础](https://www.cnblogs.com/jiangzongyou/p/12039580.html)**

**第一步: 环境 VMware 15 + centos 7.x + xshell 5 + winscp(传文件用)**

https://www.cnblogs.com/jiangzongyou/p/12036985.html

**第二步: 安装docker**

视频地址: https://www.bilibili.com/video/av75620482?p=20

1. yum包更新到最新

sudo yum update

2. 安装需要的软件包,yum-util提供yum-config-manager功能, 另外两个是devicemapper驱动依赖包

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

3. 设置yum源为阿里云(为了防止因网络问题导致的docker安装失败)

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

4. 安装docker-ce, (ce是社区版免费的, ee是企业版收费的)

sudo yum install docker-ce

5. 查看当前docker版本

docker -v

6. 设置docker的镜像为ustc, (ustc docker mirror是老牌子的linux镜像提供者了, 不需要注册, 公共服务)

（默认是没有该文件的，请先建一个）

ustc官方介绍:  http://mirrors.ustc.edu.cn/help/dockerhub.html?highlight=docker

其实**ustc并不好用**, 阿里云好用, 参考阿里云加速  https://blog.csdn.net/julien71/article/details/79760919

mkdir -p /etc/docker  
cd /etc/docker  
touch daemon.json  
vi /etc/docker/daemon.json

在该文件中输入以下内容 :

[复制代码](javascript:void(0);)

{  
"registry-mirrors": ["https://docker.mirrors.ustc.edu.cn/"]  
}

# 每次修改这个文件后都要记得重启docker  
systemctl daemon-reload  
systemctl restart docker

[复制代码](javascript:void(0);)

7. 启动与停止docker

[复制代码](javascript:void(0);)

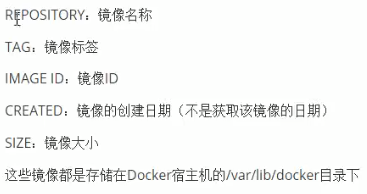
systemctl start docker # 启动  
systemctl status docker # 查看当前docker状态 "Active: active (running) 代表运行中" "Active: inactive (dead) 代表关闭"  
systemctl stop docker # 停止  
*systemctl restart docker # 重启  
systemctl enable docker # 设置成开机自动启动*docker info # 查看详细信息  
docker --help # 查看具体有哪些命令,以及怎么用

[复制代码](javascript:void(0);)

**第三步: 镜像命令**

1. 查看镜像

docker images

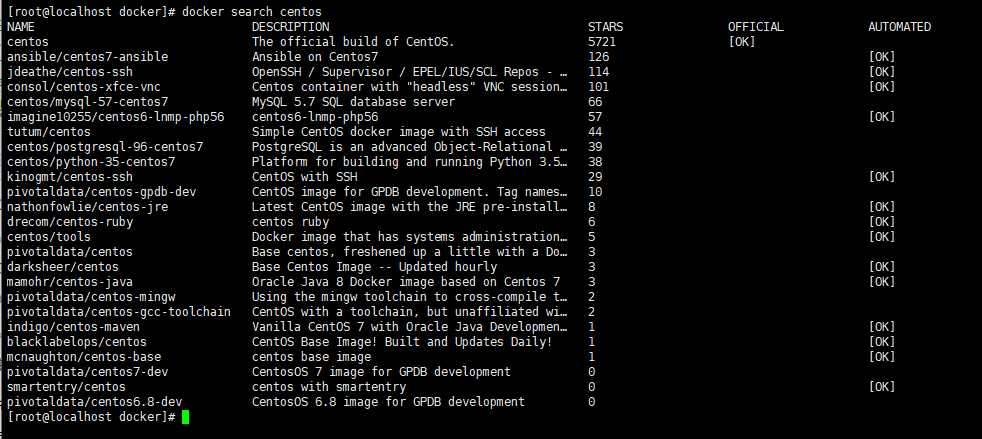


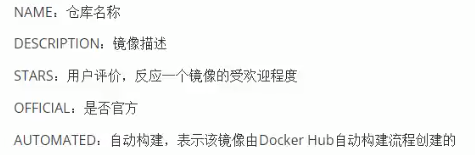
 镜像名称是可以相同的, tag经常被当做版本号来使用

Active: active (running) 代表运行中

2. 网上搜索镜像

docker search centos





3. 从中央仓库拉取镜像(如果卡主了,就ctrl+c停止, 再重新拉取)

docker pull tutum/centos # 默认是直接拉取该仓库的最新版本, 可以用 docker pull tutum/centos:14.04 来指定版本  
docker pull centos # 这个官方的当前是必备

4. 删除镜像

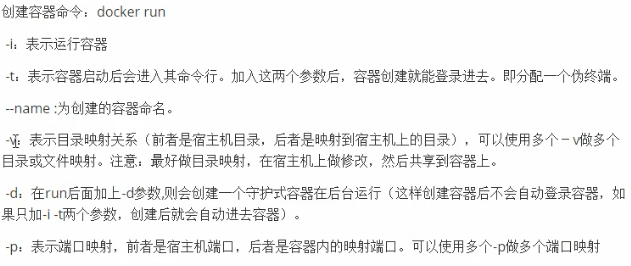
docker rmi 镜像ID #需要当前镜像下的所有容器都停止或全被删除,才能删除镜像

**第五步: 容器命令**

1. 查看容器(一个镜像可以创建多个容器)

docker ps　　 # 查看正在运行的容器  
docker ps -a # 查看所有容器,包括开机的和关机的  
docker ps -l # 查看最后一次运行的容器

 2. 创建容器



(1) 交互式方式创建容器

docker run -it --name=容器名称 镜像名称:标签 /bin/bash # 容器名称必须唯一,如果标签是lastest就可以省略不写,/ban/bash的作用是因为容器的运行至少需要一个进程, 这里就指定用命令行作为启动命令  
docker run -it --name=mycentos centos /bin/bash # 这里我们用官方的centos来创建没有centos容器,回车后会自动进行该容器,用ls命令查看便知, 再使用docker ps就能看到现在正在运行的该容器  
exit # 退出当前容器环境,docker ps里面会显示Exited

(2) 守护式方式创建容器

docker run -id --name=容器名称 镜像名称:标签 # 守护式不需要/bin/bash  
docker run -id --name=mycentos2 centos # 容器名称必须唯一,创建后不会自动进入,但其实后台在运行,返回一长串字符串代表创建成功  
docker exec -it mycentos2 /bin/bash # 需要手动进入该容器  
*exit # 退出当前容器环境, 但不会真正退出,后台还是在运行,* docker ps里面会显示Up

(3) 停止与启动

docker stop mycentos2 # 这才真正停止了这个容器, 也可以用 docker stop cd6c52a90b1b 这种方式  
docker start mycentos2 # 启动容器(必须要启动容器后,才能够进入容器)

(4) 往容器中拷贝文件

docker cp /root/anaconda-ks.cfg mycentos2:/usr/local # 往容器里拷贝文件, 需要退出容器在宿主机上才能执行这个命令  
docker cp mycentos2:/usr/local/anaconda-ks.cfg /root/anaconda-ks22.cfg # 从容器里面往外拷贝文件,需要退出容器在宿主机上才能执行这个命令

(5) 目录挂载: 方便往容器中拷贝文件( -v 宿主机目录:容器目录, 如果目录不存在会自动创建)

docker run -id --name=mycentos3 -v /usr/local/mycentos3Files:/usr/local/myfiles centos # 只要修改了宿主机/usr/local/mycentos3Files里面的内容,那么容器mycentos3的/usr/local/myfiles会相应的改变

如果共享的是多级的目录, 可能会出现权限不足的提示

因为Centos7的安全模块selinux把权限禁掉了, 我们需要添加参数 --privileged=true 来解决挂载目录没有权限的问题

(6) 查看容器ip

docker inspect mycentos3 # 查看容器mycentos3的所有信息  
docker inspect --format='{{.NetworkSettings.IPAddress}}' mycentos3 # 只看ip的内容

(7) 删除容器

docker rm mycentos # 需要改容器处于停止状态

**第六步: 常用应用部署**

**1. mysql部署**

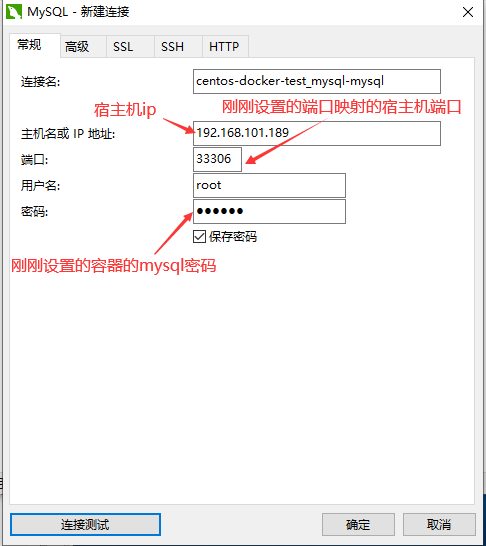
(1) 拉取mysql镜像

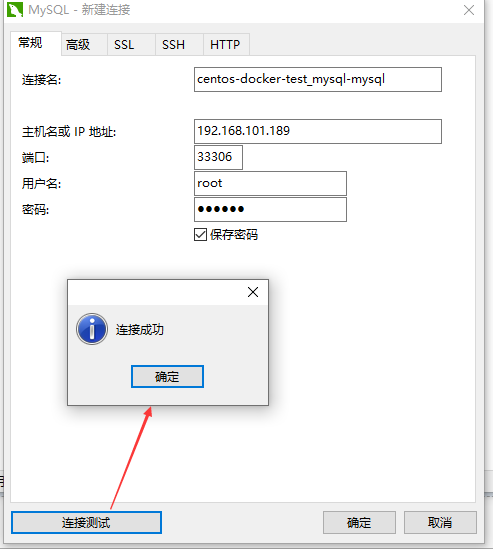
docker pull centos/mysql-57-centos7

(2) 创建容器( 端口映射: -p 宿主机端口:容器端口, MYSQL\_ROOT\_PASSWORD设置密码)

docker run -id --name=test\_mysql -p 33306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 centos/mysql-57-centos7

 (3) 接下来使用navicat连接这个mysql





**2. tomcat部署**

(1) 拉取镜像

docker pull tomcat:7-jre7

(2) 创建容器

docker run -id --name=mytomcat -p 9000:8080 -v /usr/local/webapps:/usr/local/tomcat/webapps tomcat:7-jre7

(3) 下载别人已经部署好的cas.war, 下载地址:  https://github.com/hjzgg/cas-overlay-template-.2.7/blob/master/target/cas.war

用xshell传到/root目录下后转移他, 然后在浏览器中输入  ip:9000/cas去访问

mv cas.war /usr/local/webapps

**3. Nginx部署**

(1) 拉取镜像

docker pull nginx

(2) 创建容器

docker run -id --name=mynginx -p 80:80 nginx



(3) 进入容器并查看nginx的html放在哪儿的

docker exec -it mynginx /bin/bash # 进入容器  
cat /etc/nginx/nginx.conf # 查看nginx配置, 可以发现 ''include /etc/nginx/conf.d/\*.conf''  
cat /etc/nginx/conf.d/default.conf # 进入这个配置文件,可发现 "root /usr/share/nginx/html;" 说明nginx的主页就放在这儿的  
cd /usr/share/nginx/html # 进入这个文件夹就可以发现, 50x.html  index.html 这两个文件,说明找对了

(4) 接下来自己就可以把自己想用的静态文件夹改名为html,去覆盖nginx的html文件夹就ok了

4**. Nginx部署**

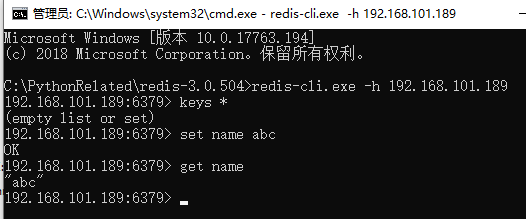
(1) 拉取镜像

docker pull redis

(2) 创建容器

docker run -id --name=myredis -p 6379:6379 redis

(3) 用windows的redis连接虚拟机上的redis容器, 以下示例说明连接成功



**第七步: 备份与迁移**

1. 容器备份

docker commit mynginx mynginx\_i # 把容器备份成镜像  
docker images　　　　　　　　　　　　 # 可以看到多了一个mynginx\_i镜像  
docker run -id --name=mynginx2 -p 81:80 mynginx\_i # 再用这个镜像来创建容器  
接下里在windows浏览器里输入"宿主机ip:81" # 访问成功, 说明这个容器备份成功

2 镜像导入导出

[复制代码](javascript:void(0);)

docker save -o mynginx.tar mynginx\_i # 把镜像导出成文件(不指定路径的话,就在当前文件夹生成)  
docker stop mynginx2　　　　　　　　　　　 # 把刚刚开启的容器停止掉  
docker rm mynginx2 　　　　　　　　　　 # 删除这个容器  
docker rmi mynginx\_i 　　　　　　　　　# 删除这个镜像  
docker images 　　　　　　　　　　　　　 # 可见,mynginx\_i这个镜像不在了  
docker load -i mynginx.tar 　　　　　　# 导入镜像  
docker images 　　　　　　　　　　　　　　# 可见,mynginx\_i这个镜像恢复了(说明是可以带名字恢复的)

[复制代码](javascript:void(0);)

**第八步: Dockerfile**

什么是Dockerfile? 转载 <https://www.cnblogs.com/edisonchou/p/dockerfile_inside_introduction.html>



1. 基于 centos 7 创建 jdk8 镜像

jdk8 官网下载地址: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

[复制代码](javascript:void(0);)

mkdir -p /usr/local/dockerjdk8 # 创建一个文件夹, 并把下载的jdk8-linux移动到里面  
cd /usr/local/dockerjdk8 # 进入到该目录  
vi Dockerfile # 创建Dockerfile文件,**固定写法,注意大小写**

　　FROM centos:7  # 指定基础镜像,如果没有centos:7, 那么它就会自动下载centos7  
　　MAINTAINER me  # 声明镜像创建者  
　　RUN mkdir /usr/local/java  # 创建一个目录  
　　ADD jdk-8u121-linux-x64.tar.gz /usr/local/java  # 自动把该文件解压到指定目录下

　　ENV JAVA\_HOME=/usr/local/java/jdk1.8.0\_121  # 创建环境变量,让我们在任何地方都能使用jdk命令,jdk1.8.0\_121是解压后的文件夹的名字,不要输错  
　　ENV JRE\_HOME $JAVA\_HOME/jre  
　　ENV CLASSPATH $JAVA\_HOME/bin/dt/jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib:$CLASSPATH  
　　ENV PATH $JAVA\_HOME/bin:$PATH

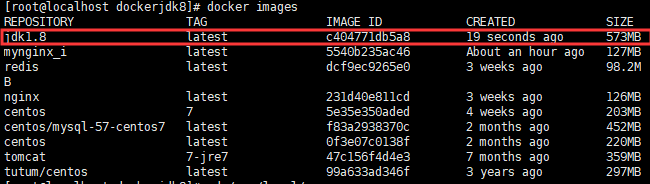
　　--- wq 保存退出 ---

docker build -t='jdk1.8' .   # 创建 jdk1.8 镜像,指定Dockfile的文件路径时当前文件夹, **有个点 .** 别忘记

docker images   # 可以看到多了一个 jdk1.8 镜像, 说明创建成功

[复制代码](javascript:void(0);)

如下

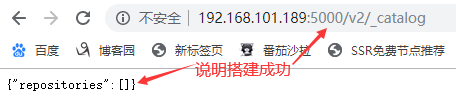


**第九步: Docker私有仓库**

扫盲转载: <https://www.cnblogs.com/jaazz/p/9334183.html>

1. 下载registry

docker pull registry # 下载  
docker run -id --name=registry -p 5000:5000 registry　 # 创建容器,映射端口为5000  
接下来去测试一下是否搭建成功



接下来去配置daimon.json文件, 让docker信任私有地址, 我们才能上传镜像到私有仓库

[复制代码](javascript:void(0);)

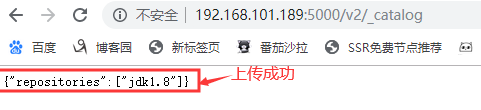
vi /etc/docker/daemon.json   
　　{  
　　"registry-mirrors": ["https://9cpn8tt6.mirror.aliyuncs.com"],  
　　"insecure-registries":["192.168.101.189:5000"] # 把自己的宿主机的ip和5000端口加进去  
　　}  
　　--- wq 保存退出 ---  
  
systemctl restart docker # 需要重启docker, 重启成功则说明配置成功,否则去检查insecure是否输错

[复制代码](javascript:void(0);)

接下来去上传镜像

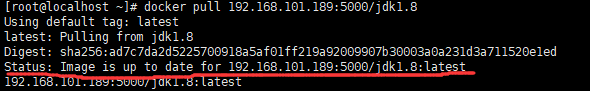
tag标签详细用法, 转载: <https://www.cnblogs.com/pzk7788/p/10180919.html>

docker start registry # 首先需要开启私服  
docker tag jdk1.8 192.168.101.189:5000/jdk1.8 # 标记此镜像为私有仓库的镜像  
docker images　　　　　# 可以看到多了一个 192.168.101.189:5000/jdk1.8 镜像, 但id其实是一样的  
docker push 192.168.101.189:5000/jdk1.8 # 上传到私有仓库



 最后再从私有仓库下载镜像,

docker pull 192.168.101.189:5000/jdk1.8 # 回车, 发现提示如下



 意思是说, 已经存在该镜像了, 所以我们可以先删掉它,再下载试试,来验证命令是否可用

docker rmi 192.168.101.189:5000/jdk1.8 # 先删掉它  
docker images # 确认不存在这个镜像  
docker pull 192.168.101.189:5000/jdk1.8 # 可以发现,下载成功了,说明下载指令没有问题

------------- 完美的分割线 -----------