Docker 项目的目标是实现轻量级的操作系统虚拟化解决方案，缩短项目从编码到上线的周期。保证开发环境与应用程序要部署的生产环境具备一致性。他的基础是linux容器等技术。

Docker推荐一个容器只运行一个程序。

使用Docker，开发人员只需要关心容器中运行的应用程序，

运维人员只需要关心如何管理容器。

Docker是CS架构程序。

镜像与容器

容器是基于镜像启动起来的，用户基于镜像运行容器

Docker借鉴了标准集装箱的概念。标准集装箱将货物运往世界各地，Docker将这个

模型运用到自己的设计中，唯一不同的是：集装箱运输货物，而Docker运输软件。

设置镜像加速器

vi /etc/docker/daemon.json

为

{

"registry‐mirrors": ["https://docker.mirrors.ustc.edu.cn"]

}

#### 1，启动，关闭，查看状态

systemctl start docker

systemctl stop docker

systemctl status docker

#### 2，开机自启

systemctl enable docker

其他容器一开机就自启：

docker update --restart=always postgresql

docker update --restart=no postgresql

启用ipv4网络，虚拟机可以连接外部网络

在docker的宿主机中更改以下

vi /usr/lib/sysctl.d/00-system.conf

添加如下代码：

    net.ipv4.ip\_forward=1

重启network服务

# systemctl restart network

#### 3，查看镜像

docker images

#### 4，查询镜像

docker search centos

#### 5，拉取镜像

docker pull tutum/centos

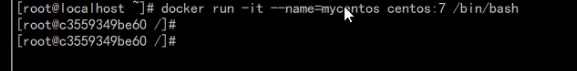
#### 6，删除镜像

docker rmi 镜像id

#### 7，创建容器：

##### 7.1交互式

docker run ‐it 容器名称+版本号 (或者容器ID) /bin/bash



##### 7.2守护式

docker run -di --name=别名 容器名称+版本号

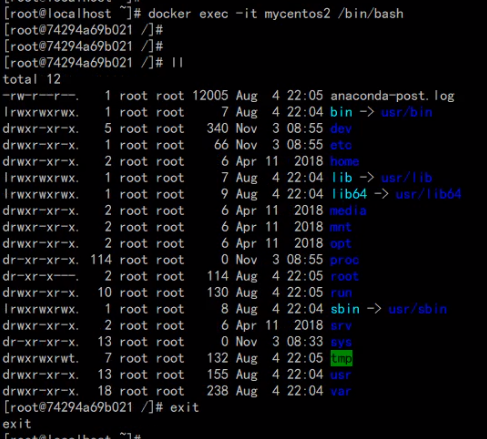
docker run -di --name=mycentos2 centos:7

##### 7.3进入容器

docker exec ‐it 别名 (服务名ID) /bin/bash

docker exec -it mycentos2 /bin/bash

IMG_257



##### 7.4停止容器

docker stop 别名 (服务名ID)

##### 7.5查看容器运行情况

docker ps

或者 docker ps -a

##### 7.6启动容器

docker start别名 (服务名ID)

#### 8，文件拷贝

要将文件拷贝到容器内可以使用cp命令

docker cp 需要拷贝的文件或目录 容器名称:容器目录

将文件从容器内拷贝出来

docker cp 容器名称:容器目录 需要拷贝的文件或目录

#### 9，目录挂载(加-v参数)

docker run ‐di ‐v /usr/local/myhtml:/usr/local/myhtml ‐‐name=mycentos3

centos:7

如果共享的是多级目录，可能会报没有权限，这是因为安全模式把权限禁掉了，我们需要添加参数 --privileged=true 来解决挂载的目录没有权限的问题

#### 10，查看容器ip地址

docker inspect 容器名称（容器ID）

也可以直接执行下面的命令直接输出IP地址

docker inspect ‐‐format='{{.NetworkSettings.IPAddress}}' 容器名称（容器ID）

#### 11，删除容器

docker rm 容器名称（容器ID）

#### 12，应用部署

##### 1,mysql

###### 拉取mysql

docker pull centos/mysql-57-centos7

###### 创建容器msyql

docker run -di --name=mysql1 -p 33306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 centos/mysql-57-centos7

###### 测试

在docker的宿主机中更改以下

vi /usr/lib/sysctl.d/00-system.conf

添加如下代码：

    net.ipv4.ip\_forward=1

重启network服务

# systemctl restart network

本地电脑连接远程mysql

mysql -h ip -P33306 -uroot -p123456

###### 停止

docker stop 服务id

###### 启动

docker start 服务id

##### 2,tomcat

###### 拉取tomcat7

docker pull tomcat:7-jre7

###### 创建容器

docker run -di --name=mytomcat -p 8085:8080 -v /usr/local/webapps/mytomcat:/usr/local/tomcat/webapps/mytomcat tomcat:7-jre7

****docker run并挂载数据卷****

docker run -d -p 8083:8080 -v /yundata/docker-data/webapps/myapps/mytomcat/:/usr/local/tomcat/webapps/mytomcat/ tomcat:7-jre7

###### 测试

可以用ip:8083进行

###### 停止

docker stop 服务id

###### 启动

docker start 服务id

##### 3,nginx

###### 拉取

docker pull nginx

###### 创建容器

docker run -di --name=mynginx1 -p 80:80 nginx

###### 测试

访问http://ip/

###### 进入容器

docker exec -it mynginx1 /bin/bash

###### 停止

docker stop 服务id

###### 启动

docker start 服务id

##### 4,Redis

###### 拉取

docker pull redis

###### 创建容器

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

###### 测试

用RedisDesktopManager连接ip

###### 停止

docker stop 服务id

###### 启动

docker start 服务id

#### 13，容器保存为镜像

docker commit mynginx mynginx\_i

#### 14，镜像保存为tar 文件

docker save -o mynginx.tar mynginx\_i

还原为镜像

docker load ‐i mynginx.tar

docker save和docker export的区别

* 对于Docker Save方法，会保存该镜像的所有历史记录
* 对于Docker Export 方法，不会保留历史记录，即没有commit历史
* docker save -o 保存的是镜像（image），docker export -o保存的是容器（container）；
* docker load用来载入镜像包，docker import用来载入容器包，但两者都会恢复为镜像；
* docker load不能对载入的镜像重命名，而docker import可以为镜像指定新名称。

#### 15，Dockerfile

1、对于开发人员：可以为开发团队提供一个完全一致的开发环境；

2、对于测试人员：可以直接拿开发时所构建的镜像或者通过Dockerfile文件构建一个新

的镜像开始工作了；

3、对于运维人员：在部署时，可以实现应用的无缝移植。

创建文件夹

mkdir -p /usr/local/java/jdk1.8.0\_171

把jdk‐8u171‐linux‐x64.tar.gz上传到/usr/local/java/jdk1.8.0\_171下

cd /usr/local/java/jdk1.8.0\_171

创建文件

vi Dockerfile

写入

#依赖镜像名称和ID

FROM centos:7

#指定镜像创建者信息

MAINTAINER TELLHOW

#切换工作目录

WORKDIR /usr

RUN mkdir /usr/local/java

#ADD 是相对路径jar,把java添加到容器中

ADD jdk‐8u171‐linux‐x64.tar.gz /usr/local/java/

#配置java环境变量

ENV JAVA\_HOME /usr/local/java/jdk1.8.0\_171

ENV JRE\_HOME $JAVA\_HOME/jre

ENV CLASSPATH $JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib:$CLASSPATH

ENV PATH $JAVA\_HOME/bin:$PATH

运行命令

docker build -t='jdk8' .

查看是否创建成功

Docker images

#### 16,私有仓库

##### 16.1拉取私有仓库镜像（此步省略）

docker pull registry

##### 16.2启动私有仓库容器

docker run -di --name=registry -p 5000:5000 registry

##### 16.3打开浏览器

输入地址http://192.168.101.137:5000/v2/\_catalog看 到 {"repositories":[]} 表示私有仓库搭建成 功并且内容为空

##### 16.4编辑daemon.json

vi /etc/docker/daemon.json

{

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

"insecure-registries": ["192.168.101.137:5000"]

}

##### 16.5重启

systemctl restart docker

##### 16.6标记此镜像为私有仓库镜像

docker tag jdk8 192.168.101.137:5000/jdk8

##### 16.7启动registry

docker start registry

##### 16.8上传标记的镜像

docker push 192.168.101.137:5000/jdk8

###### 16.9从私服下载

docker pull 192.168.101.137:5000/jdk8