链滴

docker 笔记

作者：zhangzeshan
原文链接：https://ld246.com/article/1607326203698
来源网站：链滴

docker和虚拟机的区别和优缺点

1.docker是在宿主机上 一起共享操作系统
虚拟机是在操作系统上运行操作系统
2.docker 镜像小 便于传输和存储 后者镜像庞大
3.前者无额外性能损失 后者有额外的CPU 内存消耗

在centos安装 Docker

1.安装一些必要的系统工具

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

2.添加软件源信息

```
yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-c
.repo
```

3.更新 yum 缓存

```
yum makecache fast
```

4.安装 Docker-ce

```
yum -y install docker-ce
```

5.启动 Docker 后台服务

```
systemctl start docker
```

6.测试运行 hello-world

```
docker run hello-world
```

在ubuntu安装docker

1.更换国内软件源，推荐中国科技大学的源

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
sudo sed -i 's/archive.ubuntu.com/mirrors.ustc.edu.cn/g' /etc/apt/sources.list
sudo apt update
```

2.安装需要的包

```
sudo apt install apt-transport-https ca-certificates software-properties-common curl
```

3.添加 GPG 密钥，并添加 Docker-ce 软件源，这里还是以中国科技大学的 Docker-ce 源为例

```
curl -fsSL https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://mirrors.ustc.edu.cn/docker-ce/linux/ubu
tu \$(lsb_release -cs) stable"
```

4.添加成功后更新软件包缓存

```
sudo apt update
```

5.安装docker-ce

```
sudo apt install docker-ce
```

6.设置开机自启并 启动docker

```
sudo systemctl enable docker
sudo systemctl start docker
```

7.测试运行

```
sudo docker run hello-world
```

出现Hello from Docker! 表示成功

---

docker的卸载

1.查看所安装的docker版本

```
yum list installed | grep docker
```

2.remove掉docker

yum -y remove docker版本

3.删除镜像和容器

rm -rf /var/lib/docker

Image 镜像

镜像 就相当于类
一个镜像会产生多个容器container

搜索点赞数大于30的镜像 OFFICIAL为OK 表示官方镜像:

docker search -s 30 nginx

查看持有的所有镜像

docker images （-a 显示所有属性 -q 只显示id）

ps:
REPOSITORY表示镜像的仓库源
TAG表示镜像版本标签
IMAGEID表示镜像ID

查看image的分从

docker history IMAGE ID

删除image

单个：docker rmi ab8424e38efd(IMAGEID)
删除过程中出现Error response from daemon: conflict: unable to remove repository reference

先查看在使用的容器

docker ps -a

然后根据CONTAINER ID删除 container

docker rm CONTAINER ID

或者强制删除 docker rmi -f ab8424e38efd(IMAGEID)

多个：docker rmi a b c

全部：sudo docker rmi -f $(docker images -q)

获取image 方法1-------利用docker build 创建image

1.创建一个文件夹 进入这个文件夹 然后创建 Dockerfile（就把这个file当作一个脚本 里面写入命令

行这个文件 里面的命令也会去执行）

编写内容 （安装vim）

FROM centos

RUN yum -y install vim

保存退出

2.执行Dockerfile 创建docker-image

docker build -t zhangzeshan(注册的dockerid)/centos-vim-new(image名字) .(代表运行的dockerFle是基于当前目录的)

3.执行之后 查看imgae 列表

dockers image 就会发现新创建了上面的iamge 并且进入这个image 发现已经装好了vim

获取image 方法2------- docker pull Registry(image名字)---类似于git的拉取

//拉取ubuntu 14.04 拉取私人镜像也是同样的方法

docker pull ubuntu:14.04

自己创建一个Image-----Base Image（基础镜像）

1.拉取hello-world镜像（该镜像属于baseImage)

docker pull hello-world

2.模仿做一个像上述的镜像 创建一个文件夹 hello-world

进入这个文件夹 创建hello.c 文件
编写内容

```
#include<stdio.h>
int main(){
printf("hello-world");
}
```

3.通过gcc编译该文件 首先要进行 yum install glibc-static gcc

gcc -static hello.c -o hello

4.执行命令 查看是否输出 hello-world

./hello

5.创建Dockerfile文件 编写以下内容

```
FROM scratch
ADD hello /
CMD ["/hello"]
```

6.创建image

docker build -t zhangzeshan/hello-world .

从Dockerfile构建一个镜像---(推荐！)

简写成 docker build

1.删掉上述的image

docker image rm d34ef706b552（IMAGE ID）

2.创建目录用来存放Dockerfile

mkdir docker-centos-vim
在这个文件夹下 创建Dockerfile 文件

3.编写内容

FROM centos

RUN yum intall -y vim

4.根据 Dockerfile 创建镜像

docker build -t zhangzeshan(注册的dockerid)/centos-vim(镜像名字) .

5.查看所有镜像

会发现上面的命令 产生了一个zhangzeshan/centos-vim的镜像

docker images

6.使用这个镜像

docker run -it zhangzeshan/centos-vim

执行命令 vim 发现这个镜像也已经装了vim

7.build过程中进行调试:

复制要调试的步骤的 id 去执行临时镜像

docker run -it 临时镜像id /bin/bash

Container 容器

有镜像才能创建容器

新建并启动容器

docker run options image commond

-d： 后台运行容器 并返回容器ID 也就是启动守护式容器
docker run -d centos
(docker ps -a 会发现此时容器出于退出状态
所以要用以前台进程的形式运行 -it)

-i： 交互模式运行 通常与-t结合
如： docker run -it 6fa48e047721 (随机分配名字)
docker run -it --name centos20191228 centos (自定义容器名字)
docker run -i -t -d centos 后台运行容器 不显示终端交互

-P： 随机端口映射(大写)
i:hostPort:containerPort
ip:containerPort
hostPort:containerPort
containerPort
docker run -it -P tomcat 此时访问的端口就是为指定的默认端口tomcat的8080

-p： 指定端口映射(小写)

docker run -it -p 外部访问的端口:内部访问的端口 image名字
如：docker run -it -p 8080:8080 tomcat 这样就启动了tomcat服务 可以通过地址访问

启动容器

docker start id或者名字
进入容器并进行终端交互
docker exec -it 容器id /bin/bash
例如：sudo docker exec -it redis-web redis-cli

重启容器

docker restart id或者名

停止容器

docker stop id或者名字

强制停止（杀死正在运行的容器）

docker kill id或者名字

查看容器日志

docker logs -f -t --tail 容器id
-t 加入时间戳 -f 不停的打印日志 --tail 数字 显示最后多少条
如：docker logs 9ebaa7e6f499

查看正在运行的容器

docker ps
-a 列出当前所有正在运行的容器 docker ps -a
-l 显示最近创建的容器(上一次) docker ps -l
-n 显示最近n个创建的容器 （上n次） docker ps -n 2
-q 只显示容器编号 docker ps -q

查看容器下的进程

docker top 容器id

查看容器内部细节

docker inspect 容器id

进入容器（先进行start）

docker exec -it 容器id 或者 docker attach 容器id
exec: 在容器中打开新的终端 并且可以启动新的进程
此时没有进入容器的终端 但是却拿到容器的一些数据
如：docker exec -t id ls -l /tmp 不会进入容器内 但会输出文件夹下的列表信息
docker exec -it 6188f267da9a /bin/bash （会进入容器内）
attach:直接进入容器启动命令的终端 不会启动新的进程
如：docker attach 6188f267da9a

从容器内拷贝文件到主机

在主机的终端执行 docker cp 容器id:要拷贝的地址 存放到主机的地址
docker cp 6188f267da9a:zhangzeshan.php /project/docker/

退出容器

1.容器停止并退出 exit
2.容器不停止并退出 ctrl + P + Q
执行exit出现情况：There are stopped jobs
执行kill %1 然后再次执行exit

删除容器

单个 docker rm id

多个 docker rm $(docke ps -aq)

清理所有退出状态的container 加上 -f 为强制清理

docker rm -f $(docker ps -aq)

docker container commit -------根据容器来创建新镜像

docker commit -m="提交的描述信息" -a="作者" 容器ID 要创建的目标镜像名:标签名
简写成 docker commit

一.构建自己的Docker镜像
1.进入centos镜像 此时就会产生一个centos的container

docker run -it centos

2.进入之后 给这个image 安装vim

yum install -y vim

3.安装之后退出

4.通过命令 可以看到我们刚才退出的image 会有对应的一个退出的container

docker container ls -a

5.把这个container commit 成 镜像

docker commit 50ae495256fb(NAMES的值) zhangzeshan(注册的dockerid)/centos-vim(镜像名字

6.之后查看现有的镜像

docker images
会发现多了一个镜像 名字是 zhangzeshan/centos-vim

7.进入这个镜像

docker run -it zhangzeshan/centos-vim

然后输入vim 发现这个镜像是已经安装了vim的centos镜像

容器数据卷

实现数据持久化

docker 容器产生的数据 如果不通过commit 生成新的镜像 使得数据作为镜像的一部分保存下来

那么 当容器删除后 数据自然也就没有了 为了能保存数据 在docker中使用卷

特点

1。数据卷可以在容器之间共享或重用数据

2。卷中的更改可以直接生效

3。卷中的更改不会包含在镜像的更新中

4。生命周期一直持续到没有容器使用它为止

容器内添加数据卷（也就是挂载）

一.直接命令添加(一对一)

docker run -it -v /主机的绝对路径:/容器内的路径 镜像名字
docker run -it -v /myDataVolume:/dataVolumeContainer centos
此时 主机和容器都会增加新的文件夹
此时他们之间就会同步 比如增加一个文件 两边都会有

查看是否挂载成功
根据当前的容器id 在主机执行命令
docker inspect 容器id 在出现的信息中 Binds 说明了之间已经成功挂载

不论容器是否在运行 主机在挂载项目做的任何动作都会同步

容器不能增删改查文件
docker run -it -v/myDataVolume:/dataVolumeContainer:ro centos

二.通过dockerFile添加

1.创建 docker 一个文件夹 并进入

2.在dockerFile中使用volume指令给镜像添加一个或多个数据卷

3.创建dockerFile

```
#volume test
FROM centos
VOLUME ["/dataVolumeContainer1","/dataVolumeContainer2"]
CMD echo "finished,-----success1"
CMD /bin/bash
```

4.build成一个镜像

```
docker build -f dockerfile路径 -t 镜像名字 .
```

5.运行这个镜像 就会发现已经成功挂载

```
容器对应的内容 会同步在
主机的/var/lib/docker/volumes/下
```

Dockerfile

常用命令:

保留字指令必须全部大写 后面至少跟随一个参数 否则报错

指令从上到下执行

#表示注释

每条指令会创建一个新的镜像层 并进行提交

FROM

来自于哪个基础镜像 也就是镜像基于哪个镜像 也可以是自己所创建的镜像名

FROM centos

MAINTAINER

镜像维护者的姓名和邮箱地址

MAINTAINER The Centos Project <cloud-ops@centos.org>

RUN

容器构建时 需要运行的命令

RUN groupadd -r redis && useradd -r -g redis redis

EXPOSE

当前容器对外暴露的端口号

EXPOSE 6379

WORKDIR

指定创建容器之后 终端交互进来的默认工作目录

WORKDIR /usr/mytest

ENV

构建镜像过程中设置环境变量

ENV MY_PATH /usr/mytest

使用 WORKDIR $MY_PATH

ADD

将主机的文件 拷贝到镜像 并进行自动处理和解压

COPY

将主机的文件 拷贝到镜像

COPY /project/docker /usr/local/cincontainer.txt

VOLUME

数据卷 挂载

CMD

指定容器启动时 要执行的命令

允许多个CMD指令 但只有最后一个生效 并且会被docker run 之后的参数替换掉
exec格式：CMD ["可执行文件","参数1","参数2"]
shell格式：CMD <命令>
CMD /bin/bash
启动容器的时候 docker run -it centos ls -l 此时 ls -l 会覆盖掉dockerfile的 CMD命令的内容
而 ENTRYPOINT 是 run 后面的参数 会追加到 ENTRYPOINT参数组合成新的命令 比如 ENTRYPOIN
["curl","-s","http://ip.cn"]
执行 docker run -it centos -i 就变成执行 curl -s -i http://ip.cn

ENTRYPOINT

指定容器启动时 要执行的命令

跟CMD相比 不会最后一个生效 会是追加参数的状态

ONBUILD

当构建一个被继承的dockerfile时运行命令

父镜像在被子继承后夫进行的ONBUILD被触发

也就是 只要被继承了 就进行触发这里的命令 类似于触发器

---

docker 常用安装

一.安装mysql

1.拉取mysql
docker pull mysql:5.6

2.运行mysql
docker run -p 1234:3306 --name zzsmysql -v /project/docker/mysql/conf:/etc/mysql/conf.d -v /project/docker/mysql/logs:/logs -v /project/docker/mysql/data:/var/lib/mysql -e MYSQL_OOT_PASSWORD=123456 -d mysql:5.6

3.进入mysql容器 操作mysql

4.外部链接的话 地址就是 宿主机的ip 密码是 刚才的密码123456 端口是刚才的端口1234

5.将容器内的数据库导出到宿主机
docker exec 容器id sh -C 'exec mysqldump --all-databases -uroot -p "123456"' > 宿主机目录
docker run -itd --name mysql-test -p 3306:3306 -e MYSQL_ROOT_PASSWORD=123456 mysql

---

关于docker的收集就到这里,后续还会再更新