docker镜像操作

一 镜像的创建?

1. 基于已有镜像的修改创建【不推荐】
2. Dockerfile创建镜像【推荐】
3. 本地文件系统导入【不推荐】

1和2 的比较:

1.构建镜像的不透明化，任意修改容器的文件系统后进行发布，这种修改对于镜像使用者来说是不透明的，镜像构建者一般也不会将对容器文件系统的每一步修改，记录进文档中，供镜像使用者参考。对**容器镜像**进行修改后，生成新的**容器镜像**，会多一层，而且镜像的体积只会增大，不会减小。长此以往，镜像将变得越来越臃肿。

2. Dockerfile**镜像**是完全透明的，所有用于构建镜像的指令都可以通过Dockerfile看到。甚至你还可以递归找到本镜像的任何父镜像的构建指令。也就是说，你可以完全了解一个镜像是如何从零开始，通过一条条指令构建出来的。需要修改的时候直接修改就行再构建。

二.基于已有镜像的修改创建

首先说明一下镜像是只读的，镜像没法修改。所以我们这里通过运行一个容器，因为容器是可读和可写的，所以我们要基于容器修改镜像。

1.修改前先启动容器

```
systemctl start docker
```

2.查看本地镜像

```
docker images
```

列出主机本地的镜像

参数:

```
docker images -a : 列出本地所有镜像
docker images -q : 显示镜像的id
docker  images -aq : 显示所有镜像的id
```

3.查看远程镜像

```
docker search nginx
```

官方：https://hub.docker.com/

阿里云-容器Hub服务控制台：

https://cr.console.aliyun.com/cn-beijing/instances/images

以上可以自己点击查看

4.下载镜像

```
docker pull nginx
```

```
docker pull nginx等价于 docker pull nginx:latest
```

5.并运行一个基于镜像的容器

```
docker run -ti nginx: latest  /bin/bash
```

-i: 以交互模式运行容器，通常与 -t 同时使用； -t: 为容器重新分配一个伪输入终端，通常与 -i 同时使用；

中的/bin/bash的作用是因为docker后台必须运行一个进程，否则容器就会退出，在这里表示启动容器后启动bash。

以交互模式启动一个容器【-i -t】，在容器内执行/bin/bash命令

6.执行完直接就进入到容器中

Root@后面有字符串就是运行的容器的id，hash计算值生成的64位的数字加字母的字符串【唯一】

现在我们已经进入到这个容器，就可以在这个容器中进行修改镜像

1创建文件夹：

```
mkdir demo
```

2创建文件：

```
touch abc.txt
```

以上只是做一些简单的修改

我们基于容器的镜像已经修改了，这个之后我们就要退出来

Exit 退出当前容器

```
exit
```

镜像已经修改好了所以开始封装：

封装命令：docker commit -m “this is new images” -a “用户名” 修改时的容器号 新的镜像的名称{mynginx}：tag标签{v1}

其中，-m 来指定提交的说明信息，跟我们使用的版本控制工具一样；-a** 可以指定更新的用户信息；之后是用来创建镜像的容器的 ID**；最后指定目标镜像的仓库名和** tag 信息（也可以直接镜像名称则默认保存到本地）。创建成功后会返回这个镜像的 ID 信息。**

查看镜像：

```
docker images
```

使用新的镜像来启动容器：

```
docker run -it mynginx:v1  /bin/bash
```

删除镜像：

```
docker rmi 镜像id
```

因为我们这里并没有推送到远程所有我们暂时没法通过外部网络进行访问。

例子：我们可以启动nginx的容器进行访问

```
docker run -d -p 80:80 nginx:latest
```

通过浏览器进行访问：

```
47.102.153.168:80
```

回顾：首先基于原有的镜像运行一个容器，容器中进行镜像的修改，修改之后退出这个容器

然后把这个容器变成一个镜像，它的格式是这样的【解释下命令】，现在我们知道了如何通过原有镜像生成一个新的镜像。