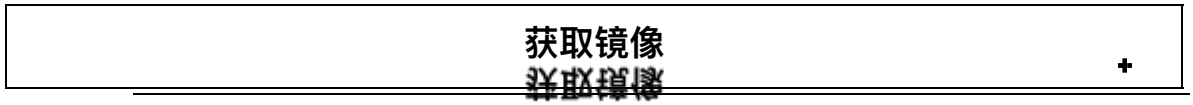
如何五分钟内学会Docker所有的镜像命令?

原创 陈同学在搬砖 陈同学在搬砖 2020-07-22 18:00

[陈同学的Docker笔记]连载企划

[第二弹] Docker镜像的那些骚操作



- 镜像是容器运行的前提
- 如果本地没有,Docker会先从 默认镜像仓库 获取,即 DockerHub
- 用户也可以通过配置使用 自定义镜像仓库

pull命令

```
格式: docker  [ image ]  pull  NAME[ : TAG]
```

- 作用:直接从DockerHub镜像源 下载镜像
- 参数: NAME表示 镜像名称 TAG 表示 镜像标签 即版本 -a 如果加了就表示获取该 仓库中所有的镜像 比如docker pull -a ubuntu --disable-content-trust=false 如果加了表示 取消镜像校验
- 例子: docker pull ubuntu: 18.04
- 注意点:
-

1.

如果不使用TAG将拉取对应镜像的 最新版本 即 latest标签 ,如果是latest则镜像的内容 会根据最新版本的变化而变化 故镜像的内容是 不稳定的 如果要部署到线上的 不要忽略版本号

-

2.

上述命令实际上是 省略了镜像仓库服务器的地址 即实际情况应该是 docker pull registry.hub.docker.com/ubuntu:18.04 如果从 非官方的镜像仓库服务器 比如网易蜂巢去下载的话应该是 docker pull hub.c.163.com/ubuntu:18.04.

- image命令可以 查看镜像
- tag命令可以 添加镜像标签
- inspect命令可以 查看镜像详细星信息

image命令

```
格式: docker  images 或者 docker Image ls
```

- 作用:

  列出本地主机上已有镜像的基本信息

- 例子:

```
docker images
```

- 参数:

  -a 加了这个参数就会列出 所有镜像文件 包括临时文件 默认是否

  --digests=true 加了这个就会列出所有镜像的 数字摘要值 默认为false 更多参数可以通过 man -docker -images

tag命令

```
格式:
 docker tag ubuntu:latest  myubuntu:latest
```

- 作用:

  给本地镜像 新建立一个标签 , 相当于是新建立一个 不同名字的副本 但是实际上是指向了 同一个镜像文件 你再用docker

images 去看的收就会 多出一个镜像名字 之后你就可以使用这个自己定义的镜像了

inspect命令

```
格式:  docker  inspect ubuntu:18.04
```

- 作用: 以JSON格式详细展示某一个镜像的信息
- 参数:

  -f 指定我们想要获取的 JSON之中的一项 比如

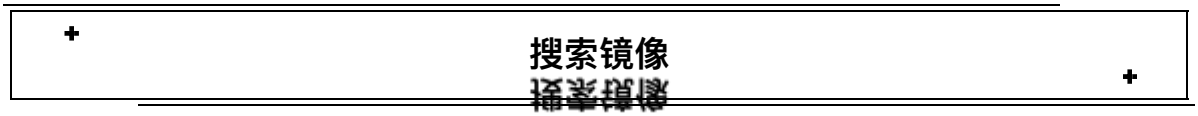
  ```
  docker inspect -f {{".Architecture"}} ubuntu:18.04
  ```

  表示获取镜像JSON信息中的Architecture项

history命令

```
格式: docker history ubuntu: 18.04
```

- 作用 : 输出镜像 各层信息
- 参数: 可以在后面加 --no -trunc 来输出 没有截断 的完整信息



- 使用search命令可以搜索在 官方镜像仓库DockerHub 中的镜像

search命令

```
格式 :
 docker search [option] keyword
```

- 作用:

  使用search命令可以搜索在 官方镜像仓库DockerHub 中的镜像

- 参数 :

  ```
      -f 过滤输出内容
  比如 docker search -f=stars=4  nginx表示在原有的结果中筛选stars大于4的镜像
  ```

```
    --limit n 表示限制输出的结果的数量小于n个
等等还有其他参数 可以看  man page
```

- 输出:

  输出结果包含关键字的镜像，输出结果每一项信息其中包括 镜像名字 、 描述 、 stars树 、 是否官方创建 、 是否自动创建 等 默认的输出结果将按照星级评价进行排序

删除和清理镜像

- rmi 删除镜像
- prune 清理镜像

rmi命令

```
格式: docker rmi  ubuntu:18.04
```

- 作用:

  删除掉ubuntu:18.04这个镜像 相当于把镜像的各个文件层删除

- 参数:

  -f 强制删除镜像 即使有镜像依赖于它 即使该镜像对应的容器正在运行 也删除 一般 不建议 这样做 正确做法是`删除所有依赖与这个镜像的容器 `` 然后再 删除镜像

- 注意点:

  1. 前面我们介绍了tag命令可以创建某个镜像的副本标签，那如果我们用rmi删除了某个镜像的 副本标签 会导致原镜像 也删除 吗? 答案是 不会的 但是如果你删除的镜像 只剩下 最后一个 副本标签的时候 就会 彻底删除 镜像

  2. 也可以通过rmi命令加 镜像ID 的形式 如果有 多个镜像 指向这个ID的话 会报错 比如你用tag命令给某个镜像 创建了多个副本

prune命令

```
格式: docker image prune命令
```

- 作用: 使用Docker一段时间后 系统中可能会 残留一些临时的镜像文件 以及一些没有被使用的镜像 可以使用该命令清理

- 参数:

  -a 加了该参数会删除 所有无用镜像 不光是临时镜像

-filter 只清理符合 筛选条件的镜像

-f 强制删除 镜像

创建镜像

- 使用commit基于 已有容器创建
- 使用import从 本地模板导入
- 使用Dockerfile创建 这个方法将单独开一期来讲

commit命令

step1

打开一个本地镜像

生成一个容器

step2

在容器里面做好自己想要的变动

比如你要创建一个文件test

step3

记录好该容器的id 即如下图中的红框部分

step4

保持上面的容器不要退出 在新的终端窗口

使用 commit命令 开始根据此容器

创建 对应的镜像 此时创建的镜像

```
格式:   docker  commit [参数]  容器id  新镜像name:新镜像的tag
```

- 参数:

```
      -a "" 添加作者信息
-m"" 添加提交信息
```

更多参数查看man page

import命令

```
格式:  docker import  my_ubuntu_v3.tar myubuntu:v3
```

- 作用:

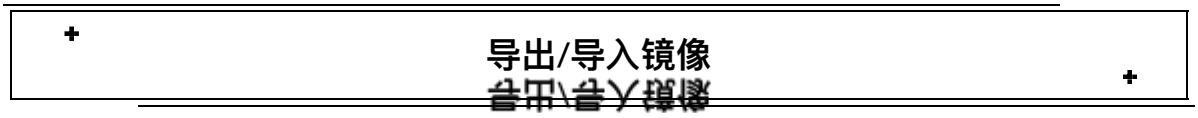
  从 本地的镜像文件 中导入为docker镜像

- 参数 :

  类似commit命令也可以通过 -m ""

  添加提交信息

  还有其他参数可以查看man page



save命令

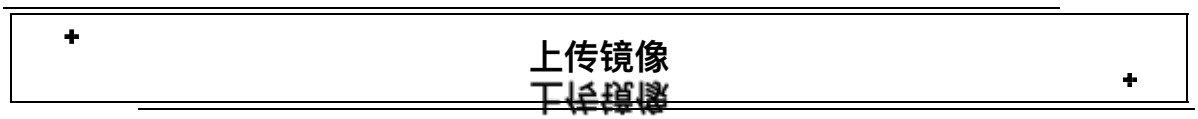
```
格式: docker save -o my_ubuntu.tar ubuntu:18.04
```

- 功能:导出docker的 ubuntu:18.04 镜像 为 文件 my_ubuntu.tar

load命令

```
格式: docker load -i ubuntu_ 18 04 tar 或者docker load < ubuntu_18 04 tar
```

- 作用:将打包的 tar文件 中的镜像导入docker



push命令

step1:

在次之前你需要

在DockerHub上 注册好自己的账户

同时在你的账户下创建好自己的一个仓库

比如我的账户下创建了一个仓库chenweijie/test

这样的话就可以开始上传了

step2:

在本地先登录好DockerHub那边的用户名和密码

step3:

将本地的镜像用tag命令设置副本 命名和仓库路径一致

step4:

然后用push命令推到远程即可

```
格式: docker push 镜像名字:镜像tag
```

- 作用: 把本地镜像推到远程仓库

- END -

- END -

我是陈同学
让技术 有温度
你的支持是我搬砖的动力

关注我
陈同学教你如何收割阿里字节腾讯

▼

往期精彩回顾

▼

你的微信消息是怎么发出去的?

1个小时学会所有Linux核心命令

一个小时学会Git

如何用最短的话讲清楚Docker的全貌?

Leetcode面试高频题汇总--数组

【设计模式】可能是东半球最透彻的单例模式讲解