链滴

Docker 常用指令

作者：linhuanzhen

原文链接：https://ld246.com/article/1501810075337

来源网站：链滴

Docker images：查看本机的docker所有镜像

Docker tag：为本地镜像添加新的标签 **

** #docker tag {REPOSITORY}:{TAG} {NEW-REPOSITORY }:{NEW-TAG}**

** #将 本地镜像中名为 hub.c.163.com/public/centos tag 为 6.7-tools 的镜像添加标签名 loc
l/centos：1st**

** #docker tag hub.c.163.com/public/centos:6.7-tools local/centos:1st**

** #再次查看docker时发现多出一个名为 local/centos:1st 的镜像，注意点，创建后的镜像IM
GE ID 已原来的相同**

**

**

** REPOSITORY TAG IMAGE ID CREATED SIZE**

** hub.c.163.com/public/centos 6.7-tools b2ab0ed558bb 5 weeks ago
01.9 MB**

** local/centos 1.0-ts b2ab0ed558bb 5 weeks ago 601.9 MB**

Docker inspect:获取某个镜像的详情

** #docker inspect {IMAGES_ID} 或者 #docker inspect {REPOSITORY}:{TAG}**

** #该命令返回镜像详情的json字符串，所以可以根据[option] -f 返回特定的属性详情**

** #docker inspect -f DockerVersion {IMAGES_ID（可以只输入镜像ID前几位代替完整的ID）
##查询某镜像的 DockerVersion**

**Docker rmi: 删除镜像 **

** #docker rmi IMAGE #IMAGE 可以为镜像TAG 或 IMAGE_ID**

** # 1 该指令会优先删除指向该镜像的标签，当只剩下一个标签及最初的镜像时，这时候执行rm
会彻底删除镜像本身。**

** # 2 该镜像有正在运行中的容器时无法删除，要删除该镜像**

** # a.使用[option] -f ： docker rmi -f b2ab0 #强制删除 id开头为 b2ab0的镜像**

** # b.停止 正在运行中容器**

Docker 创建镜像（包含多个指令）

** 创建镜像有三种方式：1.基于已有镜像的容器创建；2.基本本地模板导入创建； 3.基于Docke
file文件创建**

** 1.基于已有镜像的容器创建：**

** docker commit : 格式 docker commit [OPTION] CONTAINER ****[REPOSITORY:[TG]] **

** ## [OPTION] -a,作者信息 -m,提交信息 -p 提交时暂停容器运行**

** 演示：**** **

** ##使用镜像运行一个容器**

** #docker run -ti local/centos:1st /bin/bash ##运行容器 容器指令见下方**

** ##此时进入容器内部的**

** #touch test ##创建 test 文件夹，修改容器内容**

** #exit ##退出容器**

**

**

** 2.**

**

**

Docker create/run 创建容器

** docker create [options] IMAGES 新建容器，创建的容器属于停止状态，需要#docker start命启动它**

** docker run IMAGES [options] 新建容器并启动** ** **

** #docker run centos:7.2 /bin/echo 'hello word'**

** 利用docker run 来创建并启动容器时，Docker在后台运行的标准操作包括：**

-
  -
    -
      -
        - **检查本地是否存在指令的镜像，不存在就从共有仓库下载。**
        - **利用镜像创建并启动一个容器。**
        - **分配一个文件系统，并在只读的镜像层外面挂载一层可读写层。

**
        - **从宿主机配置的网桥接口中桥接一个虚拟借口到容器中去。**
        - **从地址池配置一个IP地址给容器。**

- **执行用户指定的应用程序。**
    - **执行完毕后容器被终止。**

** 演示：**

** #docker run -t -i centos:latest /bin/bash**

** #root@as2342sdffsd:/# ###此时进入容器内部的终端**

** 通过Ctrl+d 或者输入exit命令来退出容器,对于创建的bash容器，当退出后，该容器就处于止状态。这是因为对于docker容器来说，当运行的应用退出后，容器也就没有继续运行的必要了。**

** 更多的时候，需要让Docker容器在后台以守护态形式运行。通过 -d 参数实现。**

** 演示：**

** #docker run -d centos:6.7 /bin/sh -c "while true;do echo hello world;sleep 1;done"**

** ##运行一个容器，在容器运行时，每隔1秒输出一次hello world；**

** ##容器启动后会返回一个唯一的ID，可以通过docker ps 命令查看容器的信息；**

** #docker ps ##查看运行中的容器**

** #docker logs [容器ID] ##获取容器的输出信息**

** options：**

** -t： 让docker分配一个伪终端（pseudo-tty）并绑定到容器的标准输入上**

** -i： 让容器的标准输入保持打开**

** -d： 守护运行状态，及后台运行**

Docker stop 容器终止

** #docker stop [容器ID] ##终止容器**

** #docker ps -a -q ##查看处于终止状态的容器的ID信息**

** #docker start [容器ID] ##打开处于停止状态的容器**

** #docker restart [容器ID] ##重启容器**

**

**

Docker attach/exec/nsenter 进入容器

**　　1.attach命令**

**　　演示：**

**　　#docker run -itd centos:6.7 /bin/bash　　##运行一个容器**

**　　#docker ps　　##查看当前运行中的容器列表**

**　　#docker attach [容器NAME]/[容器ID]　　##进入容器**

**　　缺点：当多个窗口同时attach到同一个容器时，所有的窗口都会同步显示。当某个窗口因命阻塞时，其他窗口也无法继续操作。**

**　　实验时必须在运行容器时'****/bin/bash** **'，否则无法进入容器内；**

**　　2.exec命令 可以直接在容器内运行命令。**

**　　演示：**

**　　#docker exec -it [容器NAME]/[容器ID] /bin/bash**

**　3.nsenter 工具**

**　安装：**

Docker rm 删除容器

**　　#docker rm [options] [容器ID]/[容器NAME]**

**　　options:**

**　　-f 强行终止并删除一个运行中的容器**

**　　-l 删除容器的连接，但保留容器**

**　　-v 删除容器挂载的数据卷**

**　　演示：**

**　　#docker ps -a -q　　##查询所有状态的容器**

**　　#docker rm esd234sdfw2　　##删除容器ID为esd234sdfw2的容器**

Docker 导入和导出容器 （未完成）

**

**

Docker 数据卷 -v

** 概念：数据卷是一个可供容器使用的特殊目录，它绕过文件系统，可以提供很多有用的特性 **

-
  -
    -
      - **数据卷可以在容器之间共享和重用。**
      - **对数据卷的修改会立马生效。**
      - **对数据卷的更新，不会影响镜像。**
      - **卷会一直存在，知道没有容器使用。**

** docker run -v: -v 使用-V标记可以在容器内创建一个数据卷，多次使用-v可以创建多个数卷。**

** 演示：现在宿主机创建文件夹 ‘training/webapp’ **

** #cd / ##返回根目录**

** #touch training/webapp || #mkdir training/webapp ##创建文件夹**

** #docker run -d -P --name web --restart=always -v /training/webapp:/usr/local/tmcat/webapps --privileged=true tomcat:8.0**

** ## -v [宿主机映射文件夹路径]:[容器总的文件夹路径]，可指定多个**

** ## -P 是允许外部访问容器需要暴露的端口； **

** ## --privileged=true 进入容器时的root用户权限**

** ## --restart=always 设置开启自动重启容器**

Docker 端口映射 -p

** 概念:容器初始化的时候，如果没有指定端口参数，外部网络是无法直接范文容器内部的应用**

**

**

** docker run -p

**

**　　演示：**

**　　##以 【数据卷】为基础演示**

**　　#docker run -d --name demo -p 8081:8080 -v /training/webapp:/usr/local/tomcat webapps --restart=always --privileged=true tomcat:8.0**

**　　##-p : [宿主机指定映射端口]:[容器中的端口]，可同时指定多个**

**　　##-p : 可映射到指定地址 如：-p 127.0.0.1:8081:8080 ##指定宿主机127.0.0.1:8081才访问容器**

**　　##外部反问本机的8081端口时就可以直接访问容器内的应用**

**　　## docker port:查看容器的端口详情**

docker run -d --name nginx -p 80:80 -v /data/apps/nginx/conf.d/:/etc/nginx/conf.d --restar =always --privileged=