# Docker使用指南

日期：2018.06.13

制作人： 伍奕东

## 1. Docker介绍

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化。

Docker 的基础是 Linux 容器（LXC）等技术。在 LXC 的基础上 Docker 进行了进一步的封装，让用户不需要去关心容器的管理，使得操作更为简便。用户操作 Docker 的容器就像操作一个快速轻量级的虚拟机一样简单，并且容器是完全使用沙箱机制，相互之间不会有任何接口。开发者在笔记本上编译测试通过的容器可以批量地在生产环境中部署，包括VMs（虚拟机）、bare metal、OpenStack 集群和其他的基础应用平台。

Docker通常用于如下场景：web应用的自动化打包和发布；自动化测试和持续集成、发布；在服务型环境中部署和调整数据库或其他的后台应用；从头编译或者扩展现有的OpenShift或Cloud Foundry平台来搭建自己的PaaS环境。

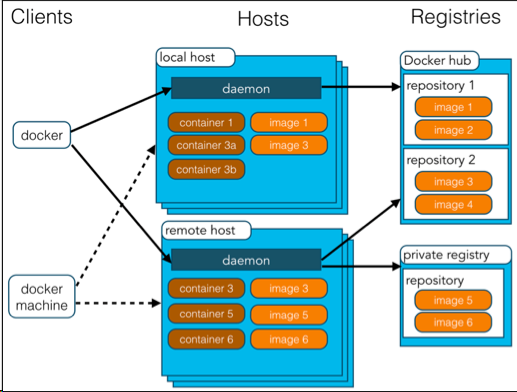
Docker 的特点：更快速的交付和部署；更高效的虚拟化；更轻松的迁移和扩展；更简单的管理。

Docker与虚拟机的比较：



## 2. Docker的主要概念

### 2.1 Docker系统架构



Docker 镜像(Images)：Docker 镜像是用于创建 Docker 容器的模板。

Docker 容器(Container)：容器是独立运行的一个或一组应用。

Docker 客户端(Client)：Docker 客户端通过命令行或者其他工具使用 Docker API (https://docs.docker.com/reference/api/docker\_remote\_api) 与 Docker 的守护进程通信。

Docker 主机(Host)：一个物理或者虚拟的机器用于执行 Docker 守护进程和容器。

Docker 仓库(Registry)：Docker 仓库用来保存镜像，可以理解为代码控制中的代码仓库。

Docker Hub(https://hub.docker.com) 提供了庞大的镜像集合供使用。

Docker Machine：Docker Machine是一个简化Docker安装的命令行工具，通过一个简单的命令行即可在相应的平台上安装Docker，比如VirtualBox、 Digital Ocean、Microsoft Azure。

Docker 使用客户端-服务器 (C/S) 架构模式，使用远程API来管理和创建Docker容器。

Docker系统有两个程序：docker服务端和docker客户端。其中docker服务端是一个服务进程，管理着所有的容器。docker客户端则扮演着docker服务端的远程控制器，可以用来控制docker的服务端进程。大部分情况下，docker服务端和客户端运行在一台机器上。

在Docker中，容器与镜像的关系类似于面向对象编程中的对象与类。容器可认为是对象，镜像可认为是类，故Docker 容器通过 Docker 镜像来创建。

### 2.2 镜像（Image）

Docker 镜像就是一个只读的模板，镜像可以用来创建 Docker 容器。Docker 提供了一个很简单的机制来创建镜像或者更新现有的镜像，用户甚至可以直接从其他人那里下载一个已经做好的镜像来直接使用。

镜像是一种文件结构。Dockerfile中的每条命令都会在文件系统中创建一个新的层次结构，文件系统在这些层次上构建起来，镜像就构建于这些联合的文件系统之上。Docker官方网站专门有一个页面来存储所有可用的镜像，网址是：index.docker.io。

### 2.3 容器（Container）

容器是从镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。可以把容器看做是一个简易版的 Linux 环境，Docker 利用容器来运行应用。镜像是只读的，容器在启动的时候创建一层可写层作为最上层。

docker容器可以理解为在沙盒中运行的进程。这个沙盒包 含了该进程运行所必须的资源，包括文件系统、系统类库、shell 环境等等。但这个沙盒默认是不会运行任何程序的。你需要在沙盒中运行一个进程来启动某一个容器。这个进程是该容器的唯一进程，所以当该进程结束的时候，容器也会完全的停止。

### 2.4 仓库（Repository）

仓库是集中存放镜像文件的场所，仓库注册服务器（Registry）上往往存放着多个仓库，每个仓库中又包含了多个镜像，每个镜像有不同的标签（tag）。目前，最大的公开仓库是 Docker Hub，存放了数量庞大的镜像供用户下载。

Docker仓库用来保存我们的images，当我们创建了自己的image之后我们就可以使用push命令将它上传到公有或者私有仓库，这样下次要在另外一台机器上使用这个image时候，只需要从仓库上pull下来就可以了。Docker 仓库的概念跟 Git 类似，注册服务器可以理解为 GitHub 这样的托管服务。

## \*3. 常用指令

与git很像

### 3.1 查看

1）docker images：查看所有镜像, 列出所有镜像

|  |
| --- |
| REPOSITORY TAG IMAGE ID CREATED SIZE  容器名 版本号 |

列出本地镜像中REPOSITORY为ubuntu的镜像列表。docker images ubuntu

2）docker inspect container/image：查看有关容器和镜像的底层信息。这些信息包括：容器实例的IP地址；端口绑定列表；特定端口映射的搜索；收集配置的详细信息。有两种方法：一种是docker inspect [REPOSITORY]:[TAG]；另一种是docker inspect [IMAGE ID]

3）docker search image\_name：检索image

4）docker diff [Name/ID]：列出一个容器里面被改变的文件或者目录，list列表会显示出三种事件，A 增加的，D 删除的，C 被改变的。

5）docker top：显示一个运行的容器里面的进程信息

6）docker logs [Name/ID]：从一个容器中取日志;

7）docker stats：查看容器的运行状态，查看容器的状态 CPU 内存消耗等

8）docker ps：查看正在运行中的容器；

例子：docker ps -a，查看所有容器

例子：docker ps -a -q，查看所有已退出的容器

9）docker version：检查docker的版本，这样可以用来确认docker服务在运行并可通过客户端链接。

10）docker info：显示docker系统的信息。可以用来检查Docker的安装是否正确。

11）ctop工具

一个容器的终端监控界面，监控容器的运行状况，就像是容器的top命令。

12）systemctl status docker.service

查看docker的状态。

### 3.2 加载和保存/pull和push等等

**镜像的加载和保存：**

1）docker load：加载镜像，加载的是tar格式的镜像。

使用方法："docker load -i file\_path" 或" docker load < /home/save.tar"

例子：docker load -i app\_box.tar，载入app\_box.tar到STDIN。

例子：docker load < /home/fengzheng/Docker/ubtuntu12.04.tar

2）docker save：保存镜像到一个tar包。会将镜像完整保存，包括历史版本和元数据信息，所以文件可能比较大，保存镜像为一个tar文件。注意与docker export区分开来。类似于load，该命令保存镜像为tar文件并发送到STDOUT。

使用方法："docker save image\_name -o file\_path"或"docker save image\_name > /home/save.tar"。

参数：-o表示以文件的形式导出，紧跟在-o后面的是导出文件保存的位置；最后是镜像（仓库：TAG,或者镜像id）。

例子：docker save -o /home/fengzheng/dockerImages/mmm.tar ubuntu:12.04。

例子：docker save d4121vgh321 > app\_box.tar，载入app\_box.tar到STDIN。

例子：将镜像web\_servers 通过save打包命令，以为.tar格式保存在路径/home下。

docker save web\_servers:2.0 -o /home/web\_servers.tar

**容器的导入和导出**

3）docker import：容器的导入。Docker可以导入远程文件、本地文件和目录。使用HTTP的URL从远程位置导入，而本地文件或目录的导入需要使用-参数。

例子：docker import http://example.com/example.tar，从远程位置导入。

4）docker export：容器的导出。类似于import，export命令用于将容器的系统文件打包成tar文件。export其实是容器的保存，只把镜像当前的状态保存下来，没有历史版本等信息，所有文件比使用docker save来保存要小。

例子：docker export 7691a814370e > ubuntu.tar，将容器id为7691a814370e 的导出到当前目录下，且名称为ubuntu.tar。

导入导出容器与镜像的区别

用户既可以使用 docker load 来导入镜像存储文件到本地镜像库，也可以使用 docker import 来导入一个容器快照到本地镜像库。这两者的区别在于

docker import和docker export是容器快照操作，文件将丢弃所有的历史记录和元数据信息，相当于虚拟机快照。

docker load和docker save是镜像存储操作，文件将保存完整记录，体积也要大。

**获取和发布镜像**

5）docker pull image\_name：获取镜像，即下载image。默认从docker hub上获取，如果下载太慢的话可以考虑国内的镜像仓库，例如dl.dockerpool.com:5000或者阿里云镜像库下载。

例子：sudo docker pull ubuntu:12.04，从docker官方仓库下载tag为“12.04”的镜像。

例子：sudo docker pull dl.dockerpool.com:5000/ubuntu:12.04，从dl.dockerpool获取镜像。

6）docker push new\_image\_name：发布docker镜像，即使用以下命令将镜像推送到registry。非常重要的一点是，你必须要知道存储库不是根存储库，它应该使用此格式(user)/(repo\_name)。

例子：上传本地镜像myapache:v1到镜像仓库中。docker push myapache:v1

|  |
| --- |
| 例子：[docker push到私有仓库](https://www.cnblogs.com/jifeng/p/9410003.html) |
| 1、登录  docker login http://xxxxx.com  2、登录私有hub创建项目     例如项目叫：abc-dev  2、给镜像打tag  　　docker tag 2e25d8496557 xxxxx.com/abc-dev/arc:1334  　　2e25d8496557：IMAGE ID，可以用docker images 查看  　　xxxxx.com：私有hub域名  　　abc-dev：项目名称  　　arc：镜像名称  　　1334：镜像版本号  4、推送  　　docker push xxxxx.com/abc-dev/arc:1334 |

7）docker commit ID new\_image\_name：将容器的状态保存为镜像，和git的commit命令很像。当你对某一个容器做了修改之后（通过在容器中运行某一个命令），可以把对容器的修改保存下来，这样下次可以从保存后的最新状态运行该容器。

注意，镜像名称只能取字符[a-z]和数字[0-9]。

例子：docker commit -a "fengzheng" -m "this is a commit message" ae2a59b86dd9 fengzheng/ubuntu，# -a:修改者信息；-m:注释、说明，后面紧跟着当前操作的容器id；最后是要生成的新的镜像名称。

docker commit edcc11c2f4ae hub.flexiv.cloud/ai/flexiv\_ai\_3rdparty:11.0.3-cudnn8-devel-ubuntu20.04-wuyidong\_1028

### 3.3 启动、运行和停止

1）docker attach [容器id]：进入容器。使用attach命令与运行中的容器交互，并且可以随时观察容器內进程的运行状况。

附加到一个运行的容器上面，attach命令允许你查看或者影响一个运行的容器。你可以在同一时间attach同一个容器。退出容器可以通过两种方式来完成：Ctrl+C 直接退出；Ctrl-\ 退出并显示堆栈信息（stack trace）。

2）docker run [镜像id]：新建并启动镜像。

参数解释：

-t 表示让Docker分配一个伪终端并绑定到容器的标准输入上；

-i 表示让容器的标准输入保持打开，一般-i和-t是一起使用的；

-d 表示以后台守护方式启动（即非交互模式，让容器在后台运行）；

-p表示将容器内部使用的网络端口映射到所使用的主机上，做网络配置的端口映射时使用。

例子

|  |  |
| --- | --- |
| docker run -it -p 3222:22 -p 3280:80 87e5b6b3ccc1 bash | 将宿主机的3222端口映射到容器的22端口、将宿主机的3280端口映射到容器的88端口，87e5b6b3ccc1是镜像id。 |
| docker run -p 8080:8080 tomcat:7 | 启动tomcat:7镜像。#-p 设置端口映射 -p 宿主机端口:容器端口 |
| docker run busybox /bin/echo Hello Docker | 运行“Hello World” |
| docker run -d tomcat:7 |  |
| docker run -i -t -p 8081:8080 tomcat:7 /bin/bash | 以交互的方式运行 |
| sudo docker run -it --rm -v $PWD(本地目录):/python(容器目录) python /bin/bash | 将本机的当前目录，挂载到镜像中的/python目录 |
| sudo docker run --rm -it -v /home/hyzhou/docker(本地目录):/data(容器目录) ubuntu:14.04 /bin/bash | 将本机的/home/hyzhou/docker，挂载到镜像中的/data目录 |
| docker run --gpus all \  -e DISPLAY=$DISPLAY \  --privileged \  --network host \  -v /home/ai/work/flexiv\_sw:/root/Desktop/flexiv\_sw \  -it hub.flexiv.cloud/ai/application:r2.9\_DY bash |  |

2.5）docker exec -it [image\_id]：进入正在运行容器并以命令行交互

例子

docker exec -it e9410ee182bd bash

docker exec -it e9410ee182bd bash /bin/sh

3）docker start [Name/ID]：启动指定容器。-name 可以给容器起一个名字，新建并启动容器。

4）docker restart [Name/ID]：重启一个正在运行的容器

5）docker kill [Name/ID]：杀死指定容器

6）docker stop [Name/ID]：停止指定容器

### 3.4 增加和删除

1）docker rm [Name/ID]：删除容器。需要先将该容器停止(docker stop)，然后才能删除。

例子：docker rm $(docker ps -a -q)，删除所有已退出的容器。

2）docker rmi [镜像id]：删除镜像。当要删除一个镜像时，必须是没有此镜像的容器在运行。另外，删除镜像前，要先把依赖于这个镜像的容器删除（有很多时候容器只是停止了，即exit了，并没有删除），docker rm [Name/ID] 或者docker ps -a -q。

### 3.5 其他操作

1）docker cp [Name/ID]:path hostpath：从容器里面拷贝文件/目录到指定的路径上。

2）docker build -t image\_name Dockerfile\_path：根据Dockerfile 构建出一个容器。还有一些Docker提供的额外选项，"--rm=true"表示构建成功后，移除所有中间容器；"--no-cache=false"表示在构建过程中不使用缓存。

|  |
| --- |
| docker build -t Image:ID <https://yeasy.gitbooks.io/docker_practice/content/image/build.html> |
| docker build -t openalpr https://github.com/openalpr/openalpr.git |

1. docker tag：镜像重命名

|  |
| --- |
| 将镜像ubuntu:15.10标记为 runoob/ubuntu:v3 镜像。  docker tag ubuntu:15.10 runoob/ubuntu:v3  docker tag 2630004d29cb runoob/ubuntu:v3 |

4）不需要打开docker，直接在该docker中使用命令行

|  |
| --- |
| 在docker中使用ps指令  docker exec -it nostalgic\_antonelli bash -c 'ps -ef|grep train' |

5）另外，见http://dockone.io/article/110，还有10个镜像相关的API。

### 3.6 Docker仓库管理和登录

1）docker login [options] [server]：用来登录到Docker registry服务器。

例子：docker login localhost:8080，登录自己主机的registry。

见http://dockone.io/article/107，包括了创建仓库、删除仓库、取出镜像等等。

### 3.7 重启docker

systemctl restart docker

有时候docker缓存过多，或者不同镜像之间冲突，则需要重启docker服务。该命令的作用就是重启docker服务，重启后可以利用“dockers version”命令验证是否重启成功。

### 3.8 给镜像打tag

　　docker tag 2e25d8496557 xxxxx.com/abc-dev/arc:v1

|  |
| --- |
| 例如：  对docker.io/centos这个镜像打标签：  docker tag docker.io/centos docker.io/centos:v1  IMG_256  以此类推，当需要开发第二个版本时，继续打标签：v2  docker tag docker.io/centos docker.io/centos:v2 |

## 参考资料

Docker官网：http://www.docker.com

Github Docker源码：https://github.com/docker/docker

dockor hub地址：https://hub.docker.com/

http://www.runoob.com/docker/docker-command-manual.html

http://www.ruanyifeng.com/blog/2018/02/docker-tutorial.html

http://www.docker.org.cn/

https://www.jb51.net/article/104679.htm

http://dockone.io/article/101

https://www.cnblogs.com/fengzheng/archive/2015/11/16/4958571.html