

# Docker入门

---

学习资料：

菜鸟教程<http://www.runoob.com/docker/docker-hello-world.html>

书本《docker技术入门与实践》

## 一、认识Docker容器

---

Docker是基于Go语言实现的开源容器项目。Docker为应用的开发、运行和部署提供了“一站式”的实用解决方案。

Docker的构想是要实现“Build, Ship and Run Any App, Anywhere”，即通过对应用的**封装（Packaging）**、**分发（Distribution）**、**部署（Deployment）**、**运行（Runtime）**生命周期进行管理，达到应用组件级别的“一次封装，到处运行”。这里的组件可以是一个Web应用、一个编译环境、也可以是一套数据库平台服务，甚至是一个操作系统或集群。

### 1、Docker容器虚拟化的好处

Docker通过容器来打包应用、解耦应用和运行平台。只需在新的服务器上启动需要的容器就可以。（无论新旧服务器是不是同一类型的平台）。这样节约了大量的宝贵时间，并降低部署过程出现问题的风险。

### 2、Docker在开发和运维中的优势

#### （1）更快速的交付和部署

使用 Docker，开发人员可以使用镜像来快速构建一套标准的开发环境；开发完成之后，测试和运维人员可以直接使用完全相同的环境来部署代码。只要是开发测试过的代码，就可以确保在生产环境无缝运行 Docker 可以快速创建和删除容器，实现快速迭代，节约开发、测试、部署的大量时间。

#### （2）更高效的资源利用

运行 Docker 容器不需要额外的虚拟化管理程序（Virtual Machine Manager, VMM，以及 Hypervisor）的支持，Docker 是内核级的虚拟化，可以实现更高的性能，同时对资源的额外需求很低 与传统虚拟机方式相比，Docker 的性能要提高 1~2 个数量级

#### （3）更轻松的迁移和扩展

Docker 容器几乎可以在任意的平台上运行，包括物理机、虚拟机、公有云、私有云 个人电脑 服务器等，同时支持主流的操作系统的发行版本，这种兼容性让用户可以在不同平台之间轻松地迁移应用。

#### （4）更简单的更新管理

使用 Dockerfile ，只需要小小的配置修改，就可以替代以往大量的更新工作。所有修改都以增 的方式被分发和更新，从而实现自动化并且高效的容器管理。

### 3、Docker与虚拟机比较

表 1-1 Docker 容器技术与传统虚拟机技术的比较

特 性	容 器	虚 拟 机
启动速度	秒级	分钟级
性能	接近原生	较弱
内存代价	很小	较多
硬盘使用	一般为 MB	一般为 GB
运行密度	单机支持上千个容器	一般几十个
隔离性	安全隔离	完全隔离
迁移性	优秀	一般

## 二、Docker启动与停止

没有权限则切换管理员执行，或者使用 `sudo` 如：`sudo systemctl start docker`

### ( 1 ) 启动docker

```
systemctl start docker
```

或

```
service docker start
```

### ( 2 ) 重启docker

```
systemctl restart docker
```

或

```
service docker restart
```

### ( 3 ) 关闭docker

```
systemctl stop docker
```

或

```
service docker stop
```

## 三、 Docker镜像

镜像是Docker三大核心概念中最重要的。 Docker运行容器前需要本地存在对应的镜像，如果镜像不存在， Docker会尝试先从默认镜像仓库下载（默认使用Docker Hub公共注册服务器中的仓库），用户也可以通过配置使用自定义镜像仓库。

### 1、获取镜像

使用 `docker pull` 命令直接从Docker Hub镜像源来下载镜像。

命令格式：`docker pull NAME[:TAG]`

- NAME 是镜像仓库名称（用来区分镜像）
- TAG 是镜像的标签（往往用来表示版本信息）

**注意：**

如果不显式指定TAG, 则默认会选择latest标签，这会下载仓库中最新版本的镜像。镜像的latest 标签意味着该镜像的内容会跟踪最新版本的变更而变化，内容是不稳定的。因此，从稳定性上考虑，不要在生产环境中忽略镜像的标签信息或使用默认的latest 标记的镜像。

如果：下载ubuntu

```
# docker pull ubuntu:18.04
```

```
[root@zxxh ~]# docker pull ubuntu:18.04 下载镜像
18.04: Pulling from library/ubuntu
898c46f3b1a1: Pull complete
63366dfa0a50: Pull complete 下载进度
041d4cd74a92: Pull complete
6e1bee0f8701: Pull complete
Digest: sha256:017eef0b616011647b269b5c65826e2e2ebddbe5d1f8c1e56b3599fb14fabec8
Status: Downloaded newer image for ubuntu:18.04
[root@zxxh ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
daocloud.io/library/nginx	latest	2bcb04b8db83f	2 weeks ago	109MB
ubuntu	18.04	94e814e2efa8	4 weeks ago	88.9MB
hello-world	latest	fce289e99eb9	3 months ago	1.84kB
my-test	1.0.0	fce289e99eb9	3 months ago	1.84kB
daocloud.io/library/tomcat	latest	1a51cb5e3006	3 months ago	462MB
daocloud.io/ubuntu	12.04	5b117edd0b76	24 months ago	104MB
training/webapp	latest	6fae60ef3446	3 years ago	349MB

```
[root@zxxh ~]#
```

刚刚下载的镜像

## 2、搜寻镜像

使用 `docker search` 命令搜索Docker Hub官方仓库中的镜像

```
# docker search nginx
```

```
[root@zxxh ~]# docker search --help

Usage:  docker search [OPTIONS] TERM

Search the Docker Hub for images

Options:
  -f, --filter filter  Filter output based on conditions provided
  --format string      Pretty-print search using a Go template
  --limit int          Max number of search results (default 25)
  --no-trunc           Don't truncate output
```

命令参数：

- `-f, --filter filter`：过滤输出内容；
- `--format string`：格式化输出内容；
- `--limit int`：限制输出结果个数，默认为 25 个；
- `--no-trunc`：不截断输出结果。

## 3、查看镜像信息

### (1) 使用images命令列出镜像

```
# docker images
```

或

```
# docker image ls
```

```
[root@zxxh ~]# docker images
REPOSITORY 来自那个仓库    TAG  标签          IMAGE ID  镜像ID    CREATED  创建时间    SIZE  镜像大小
daocloud.io/library/nginx    latest    2bcb04bdb83f    2 weeks ago    109MB
hello-world                    latest    fce289e99eb9    3 months ago    1.84kB
daocloud.io/library/tomcat    latest    1a51cb5e3006    3 months ago    462MB
daocloud.io/ubuntu            12.04     5b117edd0b76    24 months ago    104MB
training/webapp                latest    6fae60ef3446    3 years ago     349MB
[root@zxxh ~]# docker images --help

Usage:  docker images [OPTIONS] [REPOSITORY[:TAG]]

List images

Options:
  -a, --all            Show all images (default hides intermediate images)
  --digests            Show digests
  -f, --filter filter  Filter output based on conditions provided
  --format string      Pretty-print images using a Go template
  --no-trunc           Don't truncate output
  -q, --quiet          Only show numeric IDs
[root@zxxh ~]#
```

可选参数

- `-a, --all` 式`true | false` : 列出所有（包括临时文件）镜像文件，默认为否；
- `--digests=true | false` : 列出镜像的数字摘要值，默认为否；
- `-f, --filter=[]` : 过滤列出的镜像，如`dangling` 式`true` 只显示没有被使用的镜像；也可指定带有特定标注的镜像等；
- `--format="TEMPLATE"` : 控制输出格式，如. `ID`代表ID信息，`.Repository`代表仓库信息等；
- `--no-trunc=true | false` : 对输出结果中太长的部分是否进行截断，如镜像的ID信息，默认为是
- `-q, --quie=true | false` : 仅输出ID信息，默认为否

## （2）使用tag命令添加镜像标签

为了方便工作中使用特定镜像，可以使用 `docker tag` 命令为本地镜像添加新标签。

如：添加新标签`my-test:1.0.0`

```
# docker tag hello-world:latest my-test:1.0.0
```

```
[root@zxh ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
daocloud.io/library/nginx	latest	2bcb04bdb83f	2 weeks ago	109MB
hello-world	latest	fce289e99eb9	3 months ago	1.84kB
daocloud.io/library/tomcat	latest	1a51cb5e3006	3 months ago	462MB
daocloud.io/ubuntu	12.04	5b117edd0b76	24 months ago	104MB
training/webapp	latest	6fae60ef3446	3 years ago	349MB

```
[root@zxh ~]# docker tag hello-world:latest my-test:1.0.0
```

添加新标签 my-test:1.0.0

```
[root@zxh ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
daocloud.io/library/nginx	latest	2bcb04bdb83f	2 weeks ago	109MB
hello-world	latest	fce289e99eb9	3 months ago	1.84kB
my-test	1.0.0	fce289e99eb9	3 months ago	1.84kB
daocloud.io/library/tomcat	latest	1a51cb5e3006	3 months ago	462MB
daocloud.io/ubuntu	12.04	5b117edd0b76	24 months ago	104MB
training/webapp	latest	6fae60ef3446	3 years ago	349MB

可以看出多了my-test:1.0.0

```
[root@zxh ~]#
```

### (3) 使用inspect命令查看镜像详细信息

```
# docker inspect daocloud.io/ubuntu:12.04
```

```
training/webapp          latest          6fae60ef3446    3 years ago    349MB
[root@zxh ~]# docker inspect daocloud.io/ubuntu:12.04
[
  {
    "Id": "sha256:5b117edd0b767986092e9f721ba2364951b0a271f53f1f41aff9dd1861c2d4fe",
    "RepoTags": [
      "daocloud.io/ubuntu:12.04"
    ],
    "RepoDigests": [
      "daocloud.io/ubuntu@sha256:18305429afa14ea462f810146ba44d4363ae76e4c8dfc38288cf73aa07485005"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2017-04-12T21:05:30.976274223Z",
    "Container": "07f96280130f9446fcac0587da01f732228c62b460ab261fc4316f9ac32e6d76",
    "ContainerConfig": {
      "Hostname": "eb5b2868ea5d",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      "Cmd": [

```

### (4) 使用history命令查看镜像历史

既然镜像文件由多个层组成，那么怎么知道各个层的内容具体是什么呢？这时候可以使用 history 子命令，该命令将列出各层的创建信息

```
# docker history my-test:1.0.0
```

```
[root@zxxh ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
daocloud.io/library/nginx   latest             2bcb04bdb83f       2 weeks ago        109MB
my-test               1.0.0              fce289e99eb9       3 months ago        1.84kB
hello-world           latest             fce289e99eb9       3 months ago        1.84kB
daocloud.io/library/tomcat  latest             1a51cb5e3006       3 months ago        462MB
daocloud.io/ubuntu       12.04              5b117edd0b76       24 months ago      104MB
training/webapp         latest             6fae60ef3446       3 years ago         349MB
[root@zxxh ~]# docker history my-test:1.0.0
IMAGE                CREATED             CREATED BY          SIZE                COMMENT
fce289e99eb9         3 months ago       /bin/sh -c #(nop)  CMD ["/hello"]      0B
<missing>             3 months ago       /bin/sh -c #(nop)  COPY file:f77490f70ce51da2...  1.84kB
[root@zxxh ~]#
```

## 4、清除和清理镜像

删除镜像使用 `docker rmi 镜像标签` 或 `docker rmi 镜像ID`

help命令查看参数：`docker rmi --help`

- `-f,force`：强制删除镜像，即使有容器依赖它（慎用）
- `-no-prune`：不要清理未带标签的父镜像

### （1）使用标签删除镜像

```
# docker rmi ubuntu:latest
```

```
[root@zxxh ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zxxh/ubuntu          v2                  3f6cf6a07dcc       5 days ago         137MB
daocloud.io/library/nginx   latest             2bcb04bdb83f       2 weeks ago        109MB
ubuntu               latest             94e814e2efa8       4 weeks ago        88.9MB
hello-world           latest             fce289e99eb9       3 months ago        1.84kB
daocloud.io/library/tomcat  latest             1a51cb5e3006       3 months ago        462MB
daocloud.io/ubuntu       12.04              5b117edd0b76       24 months ago      104MB
ubuntu               15.10              9b9cb95443b5       2 years ago         137MB
training/webapp         latest             6fae60ef3446       3 years ago         349MB
[root@zxxh ~]# docker rmi ubuntu:latest
Untagged: ubuntu:latest
Deleted: sha256:017eef0b616011647b269b5c65826e2e2ebddbe5d1f8c1e56b3599fb14fabec8
Deleted: sha256:94e814e2efa8845d95b2112d54497fbad173e45121ce9255b93401392f538499
Deleted: sha256:e783d8ee44ce099d51cbe699f699a04e43c9af445d85d8576f0172ba92e4e16c
Deleted: sha256:cc7fae10c2d465c5e4b95167987eaa53ae01a13df6894493efc5b28b95c1bba2
Deleted: sha256:99fc3504db138523ca958c0c1887dd5e8b59f8104fbd6fd4eed485c3e25d2446
Deleted: sha256:762d8e1a60542b83df67c13ec0d75517e5104dee84d8aa7fe5401113f89854d9
```

列出所有镜像

删除镜像

删除的信息

通过执行 `docker rmi` 命令来删除只有一个标签的镜像，可以看出会删除这个镜像文件的所有文件层

### （2）使用镜像ID删除镜像

```
# docker rmi 3f6cf6a07dcc
```

```
[root@zxxh ~]# docker rmi 3f6cf6a07dcc 根据id 删除镜像
Error response from daemon: conflict: unable to delete 3f6cf6a07dcc (must be forced) - image is being used by stopped container 4e88b441ad9
[root@zxxh ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zxxh/ubuntu          v2                  3f6cf6a07dcc        5 days ago         137MB
daocloud.io/library/nginx latest             2bcb04bdb83f        2 weeks ago        109MB
hello-world          latest             fce289e99eb9        3 months ago       1.84kB
daocloud.io/library/tomcat latest            1a51cb5e3006        3 months ago       462MB
daocloud.io/ubuntu   12.04             5b117edd0b76        24 months ago      104MB
ubuntu               15.10             9b9cb95443b5        2 years ago        137MB
training/webapp       latest            6fae60ef3446        3 years ago        349MB
[root@zxxh ~]# docker rmi -f 3f6cf6a07dcc 根据镜像ID 强制删除镜像
Untagged: zxxh/ubuntu:v2
Deleted: sha256:3f6cf6a07dcc22433ba9717950d9278b0993f77f412cda07af34c6eb923b29b4
```

### (3) 清理镜像

使用Docker一段时间后，系统中可能会遗留一些临时的镜像文件，以及一些没有被使用的镜像。通过一些命令进行清理。

```
# docker image prune
```

```
[root@zxxh ~]# docker image prune --help
Usage:  docker image prune [OPTIONS]

Remove unused images

Options:
  -a, --all          Remove all unused images, not just dangling ones
  --filter filter    Provide filter values (e.g. 'until=<timestamp>')
  -f, --force        Do not prompt for confirmation

[root@zxxh ~]# docker image prune 清理镜像
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
[root@zxxh ~]# docker image prune -f 强制清理镜像
Total reclaimed space: 0B
[root@zxxh ~]#
```

支持选项：

- `a,-all`：删除所有无用镜像，不光是临时镜像；
- `-filter filter`：只清理符合给定过滤器的镜像；
- `-f,-force`：强制删除镜像，而不进行提示确认；

## 5、创建镜像

创建镜像的方法主要由三种：

1. 基于已有镜像的容器创建。
2. 基于本地模板导入。
3. 基于Dockerfile创建。

### (1) 基于已有容器创建



语法： `docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]`

```
[root@zxxh ~]# docker commit --help

Usage:  docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

Create a new image from a container's changes

Options:
  -a, --author string      Author (e.g., "John Hannibal Smith <hannibal@a-team.com>")
  -c, --change list        Apply Dockerfile instruction to the created image
  -m, --message string     Commit message
  -p, --pause              Pause container during commit (default true)
[root@zxxh ~]#
```

- `-a, --author=""` : 作者信息；
- `-c, --change=[]` : 提交的时候执行Dockerfile指令，包括 `CMD` | `ENTRYPOINT` | `ENV` | `EXPOSE` | `LABEL` | `ONBUILD` | `USER` | `VOLUME` | `WORKDIR` 等；
- `-m, --message=""` : 提交消息；
- `-p, --pause=true` : 提交时暂停容器运行。

## 实例：

创建自己的ubuntu镜像：

首先，如果没有启动容器。则启动容器

```
# docker run -t -i ubuntu:18.04 /bin/bash
```

得到容器的id为：17d88cbcacef

根据容器创建镜像

```
# docker commit -m "add new ubuntu" -a "zxxh" 17d88cbcacef zxxh/ubuntu:0.0.1
```

- `-m` : 提交信息
- `-a` : 作者信息
- 17d88cbcacef : 容器ID
- zxxh/ubuntu:0.0.1 : 镜像名称:标签

```
[root@zxxh ~]# docker images 查看镜像
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
daocloud.io/library/nginx   latest            2bcb04bdb83f      2 weeks ago       109MB
ubuntu               18.04             94e814e2efa8      4 weeks ago       88.9MB
hello-world          latest            fce289e99eb9      3 months ago      1.84kB
daocloud.io/library/tomcat  latest            1a51cb5e3006      3 months ago      462MB
daocloud.io/ubuntu       12.04             5b117edd0b76      24 months ago     104MB
training/webapp        latest            6fae60ef3446      3 years ago       349MB

[root@zxxh ~]# docker run -t -i ubuntu:18.04 /bin/bash 运行ubuntu容器
root@17d88cbcacef:/# 得到容器ID
root@17d88cbcacef:/# exit
exit
[root@zxxh ~]# docker commit -m "add new ubuntu" -a "zxxh" 17d88cbcacef zxxh/ubuntu:0.0.1 根据容器ID创建 镜像
sha256:94482201ec61230126f3f82189cede349e00776f658912217ed533d310384db8
[root@zxxh ~]# docker images
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
zxxh/ubuntu         0.0.1             94482201ec61      6 seconds ago     88.9MB 已创建的镜像
daocloud.io/library/nginx   latest            2bcb04bdb83f      2 weeks ago       109MB
ubuntu               18.04             94e814e2efa8      4 weeks ago       88.9MB
hello-world          latest            fce289e99eb9      3 months ago      1.84kB
daocloud.io/library/tomcat  latest            1a51cb5e3006      3 months ago      462MB
daocloud.io/ubuntu       12.04             5b117edd0b76      24 months ago     104MB
training/webapp        latest            6fae60ef3446      3 years ago       349MB
[root@zxxh ~]#
```

## (2) 基于本地模板导入

直接从一个操作系统模板文件导入镜像，使用 `docker import` 命令。

### 案例

```
docker import training-webapp.tar.gz zxxh/training-webapp:0.0.1
```

- training-webapp.tar.gz：本地镜像文件
- zxxh/training-webapp:0.0.1：镜像名称:标签

```
[root@zxxh docker_image]# docker images 查看已有镜像，方便之后对比
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
zxxh/ubuntu         0.0.1             94482201ec61      9 minutes ago     88.9MB
daocloud.io/library/nginx   latest            2bcb04bdb83f      2 weeks ago       109MB
ubuntu               18.04             94e814e2efa8      4 weeks ago       88.9MB
hello-world          latest            fce289e99eb9      3 months ago      1.84kB
daocloud.io/library/tomcat  latest            1a51cb5e3006      3 months ago      462MB
daocloud.io/ubuntu       12.04             5b117edd0b76      24 months ago     104MB

[root@zxxh docker_image]# ls
training-webapp.tar.gz 离线镜像包，用来导入使用

[root@zxxh docker_image]# docker import training-webapp.tar.gz zxxh/training-webapp:0.0.1 导入镜像包，创建镜像
sha256:267f8e304f9fde8fe656f2d2a6ccf22e846b385bfbf5bd7aa9a324eed3355093 返回镜像ID

[root@zxxh docker_image]# docker images
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
zxxh/training-webapp 0.0.1             267f8e304f9f      11 seconds ago    364MB 已创建成功的镜像
zxxh/ubuntu         0.0.1             94482201ec61      11 minutes ago    88.9MB
daocloud.io/library/nginx   latest            2bcb04bdb83f      2 weeks ago       109MB
ubuntu               18.04             94e814e2efa8      4 weeks ago       88.9MB
hello-world          latest            fce289e99eb9      3 months ago      1.84kB
daocloud.io/library/tomcat  latest            1a51cb5e3006      3 months ago      462MB
daocloud.io/ubuntu       12.04             5b117edd0b76      24 months ago     104MB
[root@zxxh docker_image]#
```

### (3) 基于Dockerfile创建

基于Dockerfile创建是最常见的方式。Dockerfile是一个文本文件，利用给定的指令描述基于某个父镜像创建新镜像的过程。

#### 1、编写Dockerfile

```
FROM centos:6.7
MAINTAINER Fisher "zhixinhua"

RUN /bin/echo 'root:123456' | chpasswd
RUN useradd zxh
RUN /bin/echo 'zxh:123456' | chpasswd
RUN /bin/echo -e "LANG=\\"en_US.UTF-8\\" >/etc/default/locale
EXPOSE 22
EXPOSE 80
CMD /usr/sbin/sshd -D
```

指令必须大写

- FROM：指定使用哪个镜像源
- MAINTAINER：作者
- RUN：在镜像执行的命令
- EXPOSE：暴露端口

#### 2. 通过 docker build 构建镜像

```
# docker build -t zxh/centos:6.7 .
```

- -zxh/centos:6.7：镜像名称:标签
- .：指定当前目录的dockerfile文件

## 6、存出和载入镜像

### (1) 存出镜像

使用 docker save 命令导出镜像到本地文件

```
# docker save -o ubuntu_18.04.tar ubuntu:18.04
```

- -o：导出的文件

将ubuntu:18.04 镜像导出为ubuntu\_18.04.tar

```
[root@zxxh ~]# docker save --help
Usage:  docker save [OPTIONS] IMAGE [IMAGE...]

Save one or more images to a tar archive (streamed to STDOUT by default)

Options:
  -o, --output string  Write to a file, instead of STDOUT
[root@zxxh ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zxxh/centos          6.7                 79d626fa3245       2 hours ago        191MB
zxxh/training-webapp 0.0.1              267f8e304f9f       2 hours ago        364MB
zxxh/ubuntu          0.0.1              94482201ec61       2 hours ago        88.9MB
daocloud.io/library/nginx latest             2bcb04bdb83f       2 weeks ago        109MB
centos               6.7                9f1de3c6ad53       3 weeks ago        191MB
ubuntu               18.04              94e814e2efa8       4 weeks ago        88.9MB
hello-world          latest             fce289e99eb9       3 months ago        1.84kB
daocloud.io/library/tomcat latest             1a51cb5e3006       3 months ago        462MB
daocloud.io/ubuntu   12.04              5b117edd0b76       24 months ago      104MB
[root@zxxh ~]# pwd
/root
[root@zxxh ~]# docker save -o ubuntu_18.04.tar ubuntu:18.04
[root@zxxh ~]# ls
anaconda-ks.cfg  ubuntu_18.04.tar
[root@zxxh ~]#
```

可用参数

导出ubuntu到本地文件

导出的文件

## (2) 载入镜像

使用 `docker load` 将tar文件导入本地镜像库中。

```
# docker load -i ubuntu_18.04.tar
```

或

```
# docker load < ubuntu_18.04.tar
```

- `-i` : 指定导入的文件

```
[root@zxxh ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zxxh/centos          6.7                79d626fa3245       2 hours ago        191MB
zxxh/training-webapp 0.0.1              267f8e304f9f       2 hours ago        364MB
zxxh/ubuntu          0.0.1              94482201ec61       2 hours ago        88.9MB
daocloud.io/library/nginx latest             2bcb04bdb83f       2 weeks ago        109MB
centos               6.7                9f1de3c6ad53       3 weeks ago        191MB
hello-world          latest             fce289e99eb9       3 months ago        1.84kB
daocloud.io/library/tomcat latest             1a51cb5e3006       3 months ago        462MB
daocloud.io/ubuntu   12.04              5b117edd0b76       24 months ago      104MB
[root@zxxh ~]# docker load -i ubuntu_18.04.tar
Loaded image: ubuntu:18.04
[root@zxxh ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zxxh/centos          6.7                79d626fa3245       2 hours ago        191MB
zxxh/training-webapp 0.0.1              267f8e304f9f       2 hours ago        364MB
zxxh/ubuntu          0.0.1              94482201ec61       2 hours ago        88.9MB
daocloud.io/library/nginx latest             2bcb04bdb83f       2 weeks ago        109MB
centos               6.7                9f1de3c6ad53       3 weeks ago        191MB
ubuntu               18.04              94e814e2efa8       4 weeks ago        88.9MB
hello-world          latest             fce289e99eb9       3 months ago        1.84kB
daocloud.io/library/tomcat latest             1a51cb5e3006       3 months ago        462MB
daocloud.io/ubuntu   12.04              5b117edd0b76       24 months ago      104MB
[root@zxxh ~]#
```

载入镜像

载入的镜像

## 7、上传镜像

使用 `docker push` 命令上传镜像到仓库，默认上传到Docker Hub。

需要先在Docker Hub网站注册才可以上传自制镜像。

```
# docker tag hello-world:latest zxxh/hello:0.0.1 --新添加标签zxxh/hello:0.0.1
# docker push zxxh/hello:0.0.1 --将zxxh/hello:0.0.1镜像上传到Docker Hub仓库
```

输入用户名

输入密码

## 8、镜像下载加速

### DaoCloud仓库加速器

菜鸟教程中，运行 `docker run ubuntu:15.10 /bin/echo "Hello world"` 下载ubuntu比较慢。

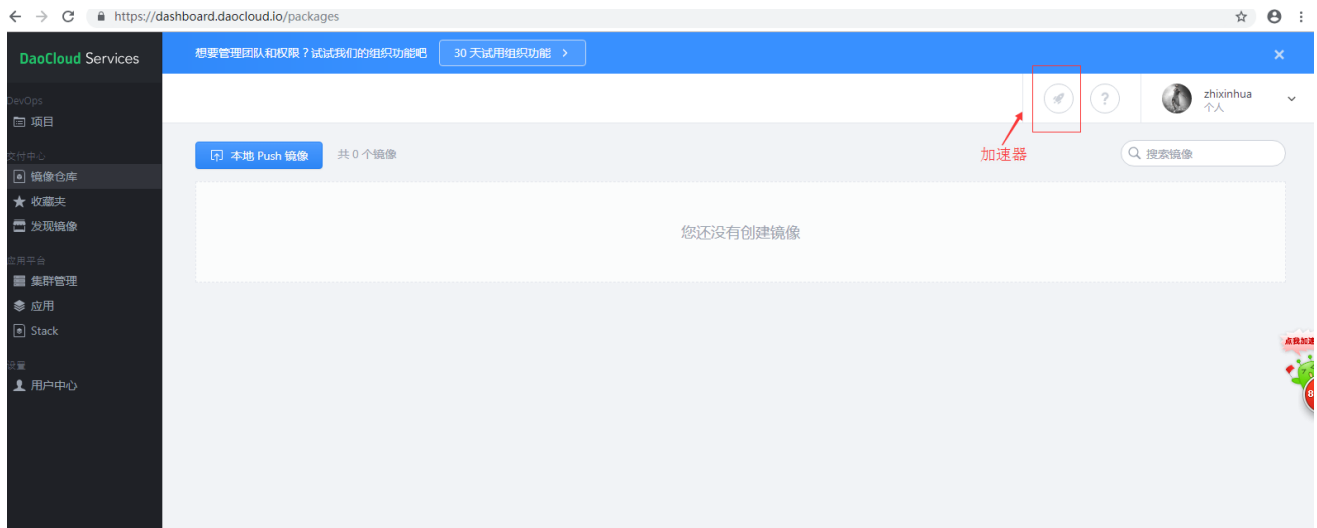
所以使用DaoCloud加速器：

参考文章：<https://www.cnblogs.com/tianfen/p/6387220.html>

## 操作步骤：

1、注册登录DaoCloud，地址：<https://dashboard.daocloud.io>

2、点击加速器



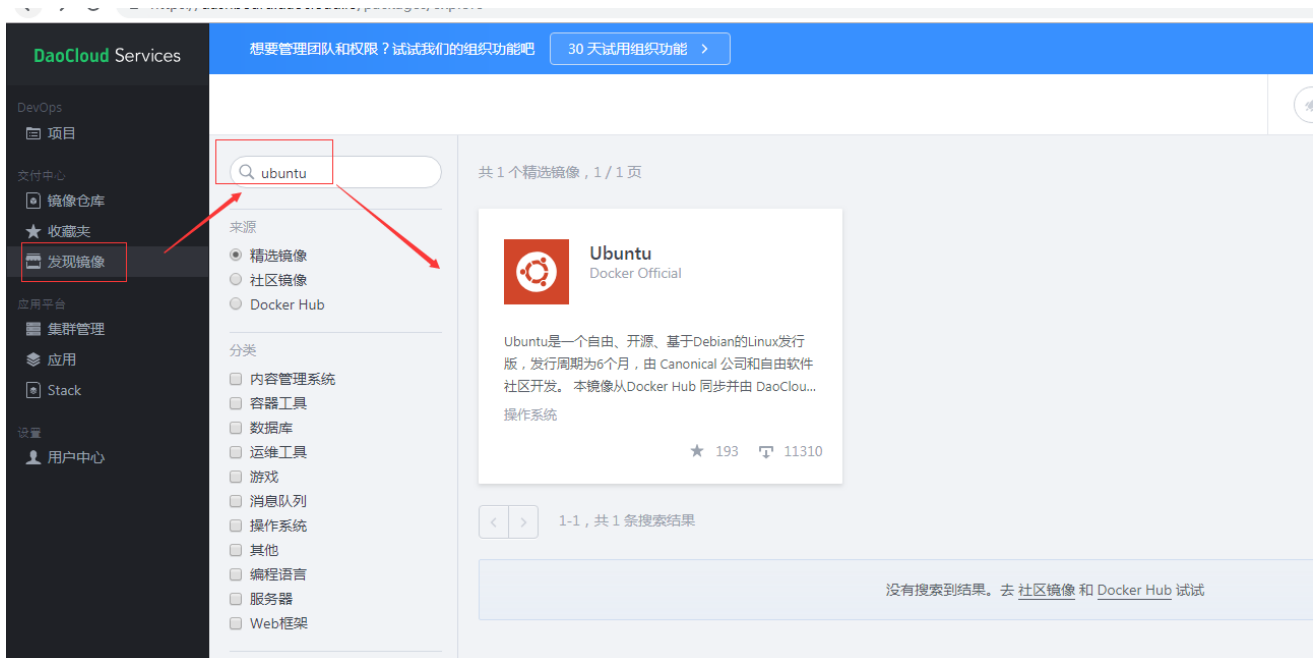
点开，可以看到加速命令



linux指定该命令即可

4、在DaoCloud上下载ubuntu：

(1) 搜索ubuntu



(2) 点开镜像，可以看到下载镜像的命令



(3) copy命令到linux服务器执行。

## 离线安装

如安装镜像 training/webapp；离线下载，下载地<https://download.csdn.net/download/wengjixi/10712479>

把training/webapp镜像放到服务器上，执行命令：`docker load -i training-webapp.tar.gz`

## 四、Docker容器

容器是 Docker 的另一个核心概念 简单来说，**容器是镜像的一个运行实例**。镜像是静态的只读文件，而容器带有运行时需要的可写文件层。

### 1、创建容器

使用 `docker create` 创建容器。更多参数查看 `docker create --help`

```
# docker create -it ubuntu:latest
```

`create` 命令新建的容器处于停止状态，可以使用 `docker [container] start` 命令来启动它

- `-i`：保持标志数据打开，默认为false
- `-t`：是否分配一个伪终端，默认为false

### 2、启动容器

使用 `docker [container] start` 命令启动一个已经创建的容器。

使用 `docker [container] restart` 命令重启容器

```
# docker start 5cedf7880194daf99ed63f0f28af5bc99b6b26b5e34302b46345425682a75820
```

启动后，可以使用 `docker ps` 命令查看运行中的容器



```
[root@zxxh ~]# docker create -it ubuntu:latest
5cedf7880194daf99ed63f0f28af5bc99b6b26b5e34302b46345425682a75820
[root@zxxh ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
5cedf7880194daf99ed63f0f28af5bc99b6b26b5e34302b46345425682a75820
[root@zxxh ~]# docker start 5cedf7880194daf99ed63f0f28af5bc99b6b26b5e34302b46345425682a75820
[root@zxxh ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
5cedf7880194daf99ed63f0f28af5bc99b6b26b5e34302b46345425682a75820
[root@zxxh ~]#
```

创建容器

start 启动容器

已启动的容器

创建的容器默认是没有启动的

### 3、新建并启动容器

使用 `docker [container] run` 创建并启动容器，等价于先执行 `docker [container] create` 命令，再执行 `docker [container] start` 命令。

```
# docker run ubuntu /bin/echo 'hello docker'
```

```
[root@zxxh ~]# docker run ubuntu /bin/echo 'hello docker'
hello docker
[root@zxxh ~]#
```

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

1. 检查本地是否存在指定的镜像，不存在就从公有仓库下载；
2. 利用镜像创建一个容器，并启动该容器；
3. 分配个文件系统给容器，并在只读的镜像层外面挂载一层可读写层；
4. 从宿主主机配置的网桥接口中桥接一个虚拟接口到容器中去；
5. 从网桥的地址池配置一个 IP 地址给容器；
6. 执行用户指定的应用程序；
7. 执行完毕后容器被自动终止

下面的命令启动一个 bash 终端，允许用户进行交互：

```
# docker run -it ubuntu:l8.04 /bin/bash
```

### 4、守护态运行

使用 `-d` 参数让docker容器以后台形式运行。

```
# docker run -d ubuntu /bin/sh -c "while true; do echo hello docker;sleep 3;done"
```

### 5、查看容器输出日志

使用 `docker [container] logs` 命令输出，更多查看--help

```
# docker logs 3605f038d653
```

```
[root@zxxh ~]# clear
[root@zxxh ~]# docker run -d ubuntu /bin/sh -c "while true; do echo hello docker;sleep 3;done" 后台运行容器
3605f038d6534bc7eee12061978e5be74856444e04b15519e5ca78ae6dc3b31e
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
3605f038d653   ubuntu   "/bin/sh -c 'while t_   6 seconds ago Up 5 seconds           focused_sammet
5cedf7880194   ubuntu:latest   "/bin/bash"            About an hour ago Up About an hour       festive_liskov
[root@zxxh ~]# docker logs 3605f038d653 后台运行的容器
hello docker
hello docker
hello docker
hello docker
hello docker
hello docker
hello docker
hello docker
hello docker
hello docker
hello docker
[root@zxxh ~]# 查看容器日志
```

## 6、停止容器

Docker容器可以使用 `pause/unpause`、`stop` 停止容器

### 6.1、暂停容器

使用 `docker [container] pause` 命令暂停一个运行中的容器

使用 `docker [container] unpause` 命令恢复运行状态

```
# docker pause 3605f038d653
# docker unpause 3605f038d653
```

```
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
3605f038d653   ubuntu   "/bin/sh -c 'while t_   13 minutes ago Up 13 minutes           focused_sammet
5cedf7880194   ubuntu:latest   "/bin/bash"            About an hour ago Up About an hour       festive_liskov
[root@zxxh ~]# docker pause ubuntu
Error response from daemon: No such container: ubuntu
[root@zxxh ~]# docker pause 3605f038d653 暂停容器
3605f038d653
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
3605f038d653   ubuntu   "/bin/sh -c 'while t_   13 minutes ago Up 13 minutes (Paused)           focused_sammet
5cedf7880194   ubuntu:latest   "/bin/bash"            About an hour ago Up About an hour       festive_liskov
[root@zxxh ~]# docker unpause 3605f038d653 重新运行容器
3605f038d653
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
3605f038d653   ubuntu   "/bin/sh -c 'while t_   14 minutes ago Up 14 minutes           focused_sammet
5cedf7880194   ubuntu:latest   "/bin/bash"            About an hour ago Up About an hour       festive_liskov
[root@zxxh ~]# 正在运行
```

### 6.2、终止容器

使用 `docker [container] stop [-t]` 来终止运行中的容器。该命令首先向容器发送SIGTERM信号，等待一段超时时间后（默认10秒），再发送SIGKILL信号来终止容器。

```
# docker stop 3605f038d653
```

```
[root@zxxh ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
3605f038d653       ubuntu             "/bin/sh -c 'while t..." 15 hours ago        Up 15 hours                    focused_sammet
5cedf7880194       ubuntu:latest      "/bin/bash"        16 hours ago        Up 16 hours                    festive_liskov
[root@zxxh ~]# docker stop 3605f038d653  停止容器
3605f038d653
[root@zxxh ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
5cedf7880194       ubuntu:latest      "/bin/bash"        16 hours ago        Up 16 hours                    festive_liskov
[root@zxxh ~]# docker start 3605f038d653  启动容器
3605f038d653
[root@zxxh ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
3605f038d653       ubuntu             "/bin/sh -c 'while t..." 15 hours ago        Up 4 seconds                    focused_sammet
5cedf7880194       ubuntu:latest      "/bin/bash"        16 hours ago        Up 16 hours                    festive_liskov
[root@zxxh ~]# docker stop -t 3 3605f038d653  3秒停止容器
3605f038d653
[root@zxxh ~]#
```

当Docker 容器中指定的应用终结时，容器也会自动终止

## 7、进入容器

在使用 `-D` 参数时，容器启动后会进入后台，用户无法看到容器中的信息，也无法进行操作。

推荐使用官方的 `attach` 或 `exec` 命令进入容器操作。

### attach 命令

```
# docker attach bb7f1d133260
```

```
[root@zxxh ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
5cedf7880194       ubuntu:latest      "/bin/bash"        17 hours ago        Up 16 hours                    festive_liskov
[root@zxxh ~]# docker run -itd ubuntu  后台运行一个容器
bb7f1d13326022bc70aa19c3afb7cdacb30e3b79b2696a1465eb8879c92dd60c
[root@zxxh ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
bb7f1d133260       ubuntu             "/bin/bash"        3 seconds ago        Up 3 seconds                    serene_sammet  新运行的容器
5cedf7880194       ubuntu:latest      "/bin/bash"        17 hours ago        Up 16 hours                    festive_liskov
[root@zxxh ~]# docker attach bb7f1d133260
root@bb7f1d133260:/# pwd
/
root@bb7f1d133260:/#
```

**注意：**使用 `attach` 命令有时候并不方便 当多个窗口同时 `attach` 到同一个容器的时候，所有窗口都会同步显示；当某个窗口因命令阻塞时，其他窗口也无法执行操作了。

### exec命令

Docker 1.3.0 版本起，Docker 提供了一个更加方便的工具 `exec` 命令，可以在运行中容器内直接执行任意命令。

```
# docker exec -it ed5464b73219 /bin/bash
```

```
[root@zxxh ~]# docker exec -it ed5464b73219 /bin/bash
root@ed5464b73219:/# w
 01:02:38 up 18:20,  0 users,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
root@ed5464b73219:/#
```

通过指定 `-it` 参数来保持标准输入打开，并且分配一个伪终端 通过 `exec` 命令对容器执行操作是最为推荐的方式

## 8、删除容器

使用 `docker [container] rm` 命令删除处于**终止或退出**状态的容器。如果要直接删除一个运行中的容器，可以添加 `-f` 参数 强制删除；Docker 会先发送 `SIGKILL` 号给容器，终止其中的应用，之后强行删除。

```
# docker rm -f ed5464b73219
```

```
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS   NAMES
ed5464b73219   ubuntu    "/bin/bash"             9 minutes ago Up 9 minutes          serene_lederberg
5cedf7880194   ubuntu:latest "/bin/bash"             17 hours ago  Up 17 hours          festive_liskov
[root@zxxh ~]# docker rm -f ed5464b73219
ed5464b73219
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS   NAMES
5cedf7880194   ubuntu:latest "/bin/bash"             17 hours ago  Up 17 hours          festive_liskov
[root@zxxh ~]#
```

强制删除运行中的容器

rm名称参数选项：

- `-f`：是否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；

## 9、导入和导出容器

如果想将容器从一个系统迁移到另一个系统，可以使用Docker的导入和导出功能。

### 9.1、导出容器

导出容器：导出一个已经创建的容器到一个文件，不管此时这个容器是否处于运行状态。

命令格式：`docker [container] export`

```
# docker export -o training.tar 69c2f01a392f
```

- `-o` 选项来指定导出的 tar文件名

```
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS   NAMES
69c2f01a392f   training/webapp "python app.py"         17 minutes ago Up 17 minutes          vibrant_tu
bb7fd1d33260   ubuntu    "/bin/bash"             2 hours ago    Up 21 minutes          serene_sammet
[root@zxxh ~]# docker export -o training.tar 69c2f01a392f
[root@zxxh ~]# ls
anaconda-ks.cfg data training.tar
[root@zxxh ~]#
```

导出容器到文件training.tar 压缩包中

## 9.2、导入容器

使用 `docker [container] import` 命令导入变成镜像

```
# docker import training.tar zhixinhua/training/webapp:v1.0
```

```
[root@zxxh ~]# ls
anaconda-ks.cfg  data  training.tar
[root@zxxh ~]# docker training.tar zhixinhua/training/webapp:v1.0
docker: 'training.tar' is not a docker command.
See 'docker --help'
[root@zxxh ~]# docker import training.tar zhixinhua/training/webapp:v1.0
sha256:4068d538afb3f25ba69827705c65467f1993df361af601275e2f78e6f68b49fb
[root@zxxh ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
zhixinhua/training/webapp	v1.0	4068d538afb3	18 seconds ago	324MB
zxxh/centos	6.7	79d626fa3245	2 weeks ago	191MB
zxxh/training-webapp	0.0.1	267f8e304f9f	2 weeks ago	364MB
zxxh/ubuntu	0.0.1	94482201ec61	2 weeks ago	88.9MB
nginx	latest	bb776ce48575	2 weeks ago	109MB
daocloud.io/library/nginx	latest	2bcb04bdb83f	4 weeks ago	109MB
centos	6.7	9f1de3c6ad53	6 weeks ago	191MB
ubuntu	18.04	94e814e2efa8	7 weeks ago	88.9MB
ubuntu	latest	94e814e2efa8	7 weeks ago	88.9MB
hello-world	latest	fce289e99eb9	3 months ago	1.84kB
zxxh/hello	0.0.1	fce289e99eb9	3 months ago	1.84kB
daocloud.io/library/tomcat	latest	1a51cb5e3006	4 months ago	462MB
daocloud.io/ubuntu	12.04	5b117edd0b76	2 years ago	104MB
training/webapp	latest	6fae60ef3446	3 years ago	349MB

→ 导入为镜像

与导入镜像 `docker load` 命令来导入一个镜像文件相似

## 10、查看容器

使用 `ps`、`inspect`、`top` 和 `stats` 命令查看容器

### 10.1、ps命令查看容器

```
# docker ps
```

显示当前正在运行的容器。

`docker ps -a` 显示所有容器，更多参数查看 `docker ps --help`

### 10.2、inspect查看容器详情

语法 `docker [container] inspect [options]`

查看某容器的具体信息，会以 json 格式返回包括容器 Id 创建时间、路径、状态、镜像、配置等在内的各项信息：

```
# docker inspect ubuntu:18.04
```

```
[root@zxxh ~]# docker inspect ubuntu:18.04
[
  {
    "Id": "sha256:94e814e2efa8845d95b2112d54497fbad173e45121ce9255b93401392f538499",
    "RepoTags": [
      "ubuntu:18.04",
      "ubuntu:latest"
    ],
    "RepoDigests": [
      "ubuntu@sha256:fd41f8a687e94b926b45a455c1c75fb29b4d7f206a969b6a16e073efa39d2ce5"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2019-03-12T00:20:17.419392342Z",
    "Container": "94772adb8869651410d18062838667884a555cf1878d567734e99dc695d6f5bf",
    "ContainerConfig": {
      "Hostname": "94772adb8869",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ]
    }
  }
]
```

## 10.3、查看容器内进程

使用 `docker [container] top` 命令查询容器进程。会打印出容器内的进程信息，包括 PID、用户、时间、命令等。

```
# docker top bb7f1d133260
```

```
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
bb7f1d133260   ubuntu   "/bin/bash"             About an hour ago    Up 18 seconds          serene_sammet

[root@zxxh ~]# docker top bb7f1d133260
UID            PID            PPID           C              STIME          TTY           TIME          CMD
root           4014           3998           0              10:02          pts/0         00:00:00      /bin/bash
```

容器内进程

## 10.4、查看统计信息

使用 `docker [container] stats` 命令，会显示CPU、内存、存储、网络等使用情况的统计

```
# docker stats bb7f1d133260
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
bb7f1d133260	serene_sammet	0.00%	408KiB / 1.795GiB	0.02%	1.31kB / 0B	8.27MB / 0B	1

## 11、其它容器命令

### 11.1、复制文件

`cp`命令支持在容器和主机直接复制文件。

```
# docker cp data bb7f1d133260:/tmp/
```

将data目录复制到容器 ( bb7f1d133260 ) 的/tmp目录下

```
[root@zxxh ~]# ls
anaconda-ks.cfg data
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
bb7f1d133260   ubuntu   "/bin/bash"             2 hours ago   Up 24 minutes                serene_sammet
[root@zxxh ~]# docker cp data bb7f1d133260:/tmp/
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
bb7f1d133260   ubuntu   "/bin/bash"             2 hours ago   Up 25 minutes                serene_sammet
[root@zxxh ~]# docker attach bb7f1d133260
root@bb7f1d133260:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  usr  var
root@bb7f1d133260:/# cd tmp
root@bb7f1d133260:/tmp# ls
data
root@bb7f1d133260:/tmp#
```

## 11.2、查看变更

使用 `docker [container] diff` 查看容器内数据修改

```
# docker diff bb7f1d133260
```

```
[root@zxxh ~]# docker diff bb7f1d133260
C /root
A /root/.bash_history
C /tmp
A /tmp/data
A /tmp/data/test.txt
[root@zxxh ~]#
```

## 11.3、查看端口映射

使用 `docker container port` 查看容器的端口映射情况

```
[root@zxxh ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
69c2f01a392f   training/webapp  "python app.py"       14 seconds ago   Up 13 seconds   0.0.0.0:5000->5000/tcp   vibrant_tu
bb7f1d133260   ubuntu   "/bin/bash"             2 hours ago     Up 4 minutes                serene_sammet
[root@zxxh ~]# docker port 69c2f01a392f
5000/tcp -> 0.0.0.0:5000
[root@zxxh ~]#
```

## 11.4、更新配置

`docker [container] update` 命令可以更新容器的一些运行时配置，主要是一些资源限制份额

```

3000/cup -> 0.0.0.0.3000
[root@zxxh ~]# docker update --help

Usage:  docker update [OPTIONS] CONTAINER [CONTAINER...]

Update configuration of one or more containers

Options:
  --blkio-weight uint16      Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
  --cpu-period int           Limit CPU CFS (Completely Fair Scheduler) period
  --cpu-quota int            Limit CPU CFS (Completely Fair Scheduler) quota
  --cpu-rt-period int        Limit the CPU real-time period in microseconds
  --cpu-rt-runtime int       Limit the CPU real-time runtime in microseconds
  -c, --cpu-shares int       CPU shares (relative weight)
  --cpus decimal             Number of CPUs
  --cpuset-cpus string       CPUs in which to allow execution (0-3, 0,1)
  --cpuset-mems string       MEMs in which to allow execution (0-3, 0,1)
  --kernel-memory bytes      Kernel memory limit
  -m, --memory bytes         Memory limit
  --memory-reservation bytes Memory soft limit
  --memory-swap bytes        Swap limit equal to memory plus swap: '-1' to enable unlimited swap
  --restart string           Restart policy to apply when a container exits

```

- `--blkio-weight` `uint16` : 更新块IO限制，10~1000，默认值为0，代表着无限制；
- `--cpu-period` `int` : 限制CPU调度器CFS (Cpmpletely Fair Scheduler) 使用时间，单位为微妙，最小1000；
- `--cpu-quota` `int` : 限制 CPU 调度器 CFS 配额，单位为微妙，最小 1000;
- `--cpu-rt-period` `int` : 限制 CPU 调度器的实时周期，单位为微妙；
- `--cpu-rt-runtime` `int` : 限制 CPU 调度器的实时运行时，单位为微妙；
- `--cpu-shares` `int` : 限制 CPU 使用份额；
- `--cpus` `decimal` : 限制 CPU 个数；
- `--cpuset-cpus` `string` : 允许使用的 CPU 核，如 0-3, 0,1;
- `--cpuset-mems` `string` : 允许使用的内存块，如 0-3' 0, 1;
- `--kernel-memor` `bytes` : 限制使用的内核内存；
- `--m, -memory` `bytes` 限制使用的内存；
- `--memory-reservation` `bytes` : 内存软限制；
- `--memory-swap` `bytes` : 内存加上缓存区的限制，-1表示为对缓冲区无限制；
- `--restart` `string` 容器退出后的重启策略

```
docker update --cpu-quota 100000 69c2f01a392f
```

## 总结

容器是直接提供应用服务的组件，也是Docker整个技术栈中最为核心的概念。

可以使用 `docker container help` 命令查看Docker支持的容器操作命令



```
[root@zxxh ~]# docker container help
```

```
Usage:  docker container COMMAND
```

```
Manage containers
```

Commands:

attach	Attach local standard input, output, and error streams to a running container
commit	Create a new image from a container's changes
cp	Copy files/folders between a container and the local filesystem
create	Create a new container
diff	Inspect changes to files or directories on a container's filesystem
exec	Run a command in a running container
export	Export a container's filesystem as a tar archive
inspect	Display detailed information on one or more containers
kill	Kill one or more running containers
logs	Fetch the logs of a container
ls	List containers
pause	Pause all processes within one or more containers
port	List port mappings or a specific mapping for the container
prune	Remove all stopped containers
rename	Rename a container
restart	Restart one or more containers
rm	Remove one or more containers
run	Run a command in a new container
start	Start one or more stopped containers
stats	Display a live stream of container(s) resource usage statistics
stop	Stop one or more running containers
top	Display the running processes of a container
unpause	Unpause all processes within one or more containers
update	Update configuration of one or more containers
wait	Block until one or more containers stop, then print their exit codes

Run 'docker container COMMAND --help' for more information on a command.

```
[root@zxxh ~]#
```