Docker入门教程（六）另外的15个Docker命令

【编者的话】DockerOne组织翻译了Flux7的Docker入门教程，本文是系列入门教程的第六篇，继续介绍Docker命令。之前的第二篇文章中我们就介绍了一些基本的Docker命令，本文过后，你将会接触到所有的Docker常用命令。努力学习吧。

在之前的文章中，我们介绍了15个Docker命令，并分享了它们的实践经验。在这篇文章中，我们将学习另外的15个Docker命令。它们分别是：

daemon：

Docker daemon是一个用于管理容器的后台进程。一般情况下，守护进程是一个长期运行的用来处理请求的进程服务。`-d`参数用于运行后台进程。

build：

如之前所讨论的，可以使用Dockerfile来构建镜像。简单的构建命令如下：
`docker build [options] PATH | URL`
还有一些Docker提供的额外选项，如：
`--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

下面是一张使用`Docker build`命令的截图。

```
root@instance-1:/mnt# sudo docker build --rm  -t image_role2 .
Sending build context to Docker daemon  2.56 kB
Sending build context to Docker daemon
Step 0 : FROM ubuntu:12.04
 ---> ae8682f4ff20
Step 1 : RUN mkdir /home/ubuntu
 ---> Running in 06a367059692
 ---> 817a8318f158
Removing intermediate container 06a367059692
Step 2 : RUN useradd -G sudo -d /home/ubuntu ubuntu
 ---> Running in 3a1d9b7e4a85
 ---> 2f01dd8e82e3
Removing intermediate container 3a1d9b7e4a85
Step 3 : RUN chown ubuntu /home/ubuntu
 ---> Running in 9e823ea94d0d
 ---> 308d4244051b
Removing intermediate container 9e823ea94d0d
Step 4 : RUN echo ubuntu:ubuntu | chpasswd
 ---> Running in 41dcb82d5a43
 ---> 8c65eabd5fe2
Removing intermediate container 41dcb82d5a43
Step 5 : RUN mkdir /app
 ---> Running in d8af79f89e05
 ---> d360f014f19d
Removing intermediate container d8af79f89e05
Successfully built d360f014f19d
```

attach:

Docker允许使用`attach`命令与运行中的容器交互，并且可以随时观察容器内进程的运行状况。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）

`attach`命令的语法是：

`docker attach container`

下面是一张显示执行`attach`命令的截图。

```
root@instance-1:/mnt# sudo docker run -d 1af296ca16e3  /usr/bin/top -b
60817477d98cb714c1550f8a15d4ac1853320d60f11efe7dbc661e8ce050e489
root@instance-1:/mnt# sudo docker attach 60817477d98cb714c1550f8a15d4ac1853320d60f11efe7dbc661e8ce050e489


top - 15:41:19 up 18:56,  0 users,  load average: 0.00, 0.01, 0.05
Tasks:   1 total,   1 running,   0 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.3%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:   3803636k total,   973172k used,  2830464k free,    58744k buffers
Swap:        0k total,        0k used,        0k free,   760544k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      20   0 17212 1164  944 R  0.0  0.0   0:00.00 top


top - 15:41:22 up 18:56,  0 users,  load average: 0.00, 0.01, 0.05
Tasks:   1 total,   1 running,   0 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3803636k total,   973296k used,  2830340k free,    58744k buffers
Swap:        0k total,        0k used,        0k free,   760544k cached
```

diff：

Docker提供了一个非常强大的命令`diff`，它可以列出容器内发生变化的文件和目录。这些变化包括添加（A-add）、删除（D-delete）、修改（C-change）。该命令便于Debug，并支持快速的共享环境。

语法是：

```
docker diff container
```

截图显示`diff`的执行。

```
root@instance-1:/mnt# docker diff 76ffa5450d48
C /home/ubuntu
A /home/ubuntu/.cache
A /home/ubuntu/.cache/motd.legal-displayed
C /root
A /root/.bash_history
C /run
C /run/motd
A /run/sshd.pid
C /run/utmp
C /var/lib/sudo
A /var/lib/sudo/ubuntu
A /var/lib/sudo/ubuntu/0
C /var/log/lastlog
C /var/log/wtmp
```

events：

打印指定时间内的容器的实时系统事件。

import：

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。从远程位置导入的语法是：

```
docker import http://example.com/example.tar
```

截图表示本地文件：

```
root@instance-1:/mnt# sudo tar -c image.tar | sudo docker import - image_app
0b27a4c1b03edaef45f9b7198fdd24eda5a3582c4d6bc8de3ba8eb9046dcd4af
root@instance-1:/mnt#
```

export：

类似于`import`，`export`命令用于将容器的系统文件打包成tar文件。

下图描述了其执行过程：

```
root@instance-1:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
60817477d98c        image_api:latest    /usr/bin/top -b     28 minutes ago      Exited (130) 28 minutes ago                     trusting_mcc
lintock
76ffa5450d48        image_api:latest    /usr/sbin/sshd -D   34 minutes ago      Up 20 minutes               0.0.0.0:80->80/tcp  api_role

root@instance-1:~# sudo docker export 76ffa5450d48 > image.tar
root@instance-1:~#
```

cp：

这个命令是从容器内复制文件到指定的路径上。语法如下：

`docker cp container:path hostpath.`

截图展示了`cp`命令的执行。

```
root@instance-1:~# docker cp  76ffa5450d48:/test /opt
root@instance-1:~# ls /opt/test/
testfile
root@instance-1:~#
```

login：

此命令用来登录到Docker registry服务器，语法如下：

`docker login [options] [server]`

如要登录自己主机的registry请使用：

`docker login localhost:8080`

```
root@instance-1:~# docker login
Username: vikramnc
Password:
Email: itsvikram06@gmail.com
Login Succeeded
root@instance-1:~# ls -l .dockercfg
-rw------- 1 root root 107 Jun 12 16:09 .dockercfg
root@instance-1:~#
```

inspect：

`Docker inspect`命令可以收集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

该命令的语法是：

`docker inspect container/image`

```
root@instance-1:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                          PORTS               NAMES
60817477d98c        image_api:latest    /usr/bin/top -b     31 minutes ago      Exited (130) 30 minutes ago                         trusting_mcc
lintock
76ffa5450d48        image_api:latest    /usr/sbin/sshd -D   36 minutes ago      Up 22 minutes                   0.0.0.0:80->80/tcp  api_role

root@instance-1:~# sudo docker inspect --format='{{.NetworkSettings.IPAddress}}' 76ffa5450d48
172.17.0.11
root@instance-1:~#
```

kill：

发送SIGKILL信号来停止容器的主进程。语法是：

`docker kill [options] container`

```
root@instance-1:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
76ffa5450d48        image_api:latest    /usr/sbin/sshd -D   39 minutes ago      Up 24 minutes       0.0.0.0:80->80/tcp  api_role
root@instance-1:~# docker kill 76ffa5450d48
76ffa5450d48
root@instance-1:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
76ffa5450d48        image_api:latest    /usr/sbin/sshd -D   39 minutes ago      Exited (-1) 1 seconds ago                       api_role

root@instance-1:~#
```

rmi：

该命令可以移除一个或者多个镜像，语法如下：

`docker rmi image`

镜像可以有多个标签链接到它。在删除镜像时，你应该确保删除所有相关的标签以避免错误。下图显示了该命令的示例。

```
root@instance-1:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
image_app           latest              0b27a4c1b03e        13 minutes ago      328.3 MB
image_role2         latest              d360f014f19d        41 minutes ago      210.5 MB
image_api           latest              1af296ca16e3        10 hours ago        374.4 MB
ubuntu              12.04               ae8682f4ff20        7 days ago          210.1 MB
root@instance-1:~# docker rmi 0b27a4c1b03e
Untagged: image_app:latest
Deleted: 0b27a4c1b03edaef45f9b7198fdd24eda5a3582c4d6bc8de3ba8eb9046dcd4af
root@instance-1:~#
```

wait：

阻塞对指定容器的其它调用方法，直到容器停止后退出阻塞。

```
root@instance-1:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
76ffa5450d48        image_api:latest    /usr/sbin/sshd -D   45 minutes ago      Exited (-1) 6 seconds ago                       api_role

root@instance-1:~# docker wait 76ffa5450d48
-1
root@instance-1:~#
```

load：

该命令从tar文件中载入镜像或仓库到STDIN。

截图显示载入app_box.tar到STDIN：

```
root@instance-1:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
image_api           latest              1af296ca16e3        10 hours ago        374.4 MB
ubuntu              12.04               ae8682f4ff20        7 days ago          210.1 MB
root@instance-1:~# docker load -i app_box.tar
root@instance-1:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
<none>              <none>              d360f014f19d        50 minutes ago      210.5 MB
image_api           latest              1af296ca16e3        10 hours ago        374.4 MB
ubuntu              12.04               ae8682f4ff20        7 days ago          210.1 MB
```

save：

类似于`load`，该命令保存镜像为tar文件并发送到`STDOUT`。语法如下：

`docker save image`

简单截图示例如下：

```
root@instance-1:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
image_role2         latest              d360f014f19d        43 minutes ago      210.5 MB
image_api           latest              1af296ca16e3        10 hours ago        374.4 MB
ubuntu              12.04               ae8682f4ff20        7 days ago          210.1 MB
root@instance-1:~# docker save d360f014f19d > app_box.tar
root@instance-1:~# ls -ltrh app_box.tar
-rw-r--r-- 1 root root 207M Jun 12 16:17 app_box.tar
root@instance-1:~#
```