Docker容器使用

Docker 客户端

docker 客户端非常简单 ,我们可以直接输入 docker 命令来查看到 Docker 客户端的所有命令选项。

```
runoob@runoob:~# docker
```

```
runoob@runoob:/root$ docker
Usage: docker [OPTIONS] COMMAND [arg...]
       docker daemon [ --help | ... ]
       docker [ --help | -v | --version ]

A self-sufficient runtime for containers.

Options:

  --config=~/.docker              Location of client config files
  -D, --debug                     Enable debug mode
  -H, --host=[]                   Daemon socket(s) to connect to
  -h, --help                      Print usage
  -l, --log-level=info            Set the logging level
  --tls                           Use TLS; implied by --tlsverify
  --tlscacert=~/.docker/ca.pem    Trust certs signed only by this CA
  --tlscert=~/.docker/cert.pem    Path to TLS certificate file
  --tlskey=~/.docker/key.pem      Path to TLS key file
  --tlsverify                     Use TLS and verify the remote
  -v, --version                   Print version information and quit

Commands:
    attach    Attach to a running container
    build     Build an image from a Dockerfile
    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 on a container's filesystem
```

可以通过命令 **docker command --help** 更深入的了解指定的 Docker 命令使用方法。

Docker容器使用

容器使用

获取镜像

如果我们本地没有 ubuntu 镜像，我们可以使用 docker pull 命令来载入 ubuntu 镜像：

```
$ docker pull ubuntu
```

启动容器

以下命令使用 ubuntu 镜像启动一个容器，参数为以命令行模式进入该容器：

```
$ docker run -it ubuntu /bin/bash
```

参数说明：

- **-i**: 交互式操作。
- **-t**: 终端。
- **ubuntu**: ubuntu 镜像。
- **/bin/bash**：放在镜像名后的是命令，这里我们希望有个交互式 Shell，因此用的是 /bin/bash。

要退出终端，直接输入 **exit**:

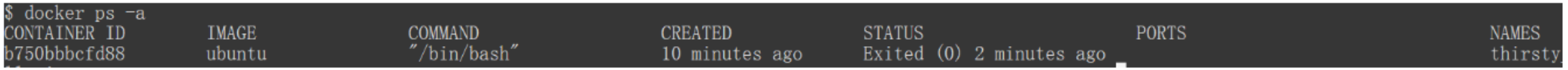
Docker容器使用

启动已停止运行的容器

查看所有的容器命令如下：

```
$ docker ps -a
```

点击图片查看大图：



使用 docker start 启动一个已停止的容器：

```
$ docker start b750bbbcfd88
```

Docker容器使用

后台运行

在大部分的场景下，我们希望 docker 的服务是在后台运行的，我们可以过 `-d` 指定容器的运行模式。

```
$ docker run -itd --name ubuntu-test ubuntu /bin/bash
```

点击图片查看大图：

```
$ docker run -itd --name ubuntu-test ubuntu /bin/bash
1e560fca39064157ba003abbdfe2a4dafc77e68a824c38ff0a2104327ca08cb6
```

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                                      NAMES
1e560fca3906        ubuntu              "/bin/bash"         9 seconds ago       Up 9 seconds                                                   ubuntu-test
```

注：加了 `-d` 参数默认不会进入容器，想要进入容器需要使用指令 **docker exec**（下面会介绍到）。

停止一个容器

停止容器的命令如下：

```
$ docker stop <容器 ID>
```

```
$ docker stop 1e560fca3906
1e560fca3906
```

停止的容器可以通过 docker restart 重启：

```
$ docker restart <容器 ID>
```

```
$ docker restart 1e560fca3906
1e560fca3906
```

Docker容器使用

进入容器

在使用 **-d** 参数时，容器启动后会进入后台。此时想要进入容器，可以通过以下指令进入：

- **docker attach**
- **docker exec**：推荐大家使用 docker exec 命令，因为此命令会退出容器终端，但不会导致容器的停止。

attach 命令

下面演示了使用 docker attach 命令。

```
$ docker attach 1e560fca3906
```

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
1e560fca3906        ubuntu              "/bin/bash"         16 minutes ago      Up 5 minutes                            ubuntu-test

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project/swoft-master
$ docker attach 1e560fca3906
root@1e560fca3906:/# exit
exit

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project/swoft-master
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project/swoft-master
```

注意： 如果从这个容器退出，会导致容器的停止。

Docker容器使用

导出和导入容器

导出容器

如果要导出本地某个容器，可以使用 **docker export** 命令。

```
$ docker export 1e560fca3906 > ubuntu.tar
```

导出容器 1e560fca3906 快照到本地文件 ubuntu.tar。

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
1e560fca3906        ubuntu              "/bin/bash"         22 minutes ago      Up 2 minutes                            ubuntu-test

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ docker export 1e560fca3906 > ./docker/ubuntu.tar

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ ll ./docker/
total 65012
-rw-r--r-- 1 LINSHAODONG 197121 66571264 10月 28 10:58 ubuntu.tar
```

这样将导出容器快照到本地文件。

导入容器快照

可以使用 docker import 从容器快照文件中再导入为镜像，以下实例将快照文件 ubuntu.tar 导入到镜像 test/ubuntu:v1:

```
$ cat docker/ubuntu.tar | docker import - test/ubuntu:v1
```

```
$ cat docker/ubuntu.tar | docker import - test/ubuntu:v1
sha256:15bf275a8d44d7a39203163714df9985c7600958b7c85353c21a3889711ae75f

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
test/ubuntu         v1                  15bf275a8d44        3 seconds ago       64.2MB
redis               4-alpine            b77690502ebe        6 days ago          20.4MB
ubuntu              latest              cf0f3ca922e0        9 days ago          64.2MB
```

Docker容器使用

删除容器

删除容器使用 **docker rm** 命令：

```
$ docker rm -f 1e560fca3906
```

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
1e560fca3906        ubuntu              "/bin/bash"         29 minutes ago      Up 9 minutes                            ubuntu-test

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ docker rm -f 1e560fca3906
1e560fca3906

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
```

下面的命令可以清理掉所有处于终止状态的容器。

$ docker container prune

运行一个 web 应用

前面我们运行的容器并没有一些什么特别的用处。

接下来让我们尝试使用 docker 构建一个 web 应用程序。

我们将在docker容器中运行一个 Python Flask 应用来运行一个web应用。

```
runoob@runoob:~# docker pull training/webapp  # 载入镜像
runoob@runoob:~# docker run -d -P training/webapp python app.py
```

```
runoob@runoob:~$ docker run -d -P training/webapp python app.py
4ecb9ce5113ded09117a94462fee760c89f9db7bfc10365ca20f2d5a4430871b
runoob@runoob:~$
```

Docker容器使用

导出和导入容器

导出容器

如果要导出本地某个容器，可以使用 **docker export** 命令。

```
$ docker export 1e560fca3906 > ubuntu.tar
```

导出容器 1e560fca3906 快照到本地文件 ubuntu.tar。

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
1e560fca3906        ubuntu              "/bin/bash"         22 minutes ago      Up 2 minutes                            ubuntu-test

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ docker export 1e560fca3906 > ./docker/ubuntu.tar

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ ll ./docker/
total 65012
-rw-r--r-- 1 LINSHAODONG 197121 66571264 10月 28 10:58 ubuntu.tar
```

这样将导出容器快照到本地文件。

导入容器快照

可以使用 docker import 从容器快照文件中再导入为镜像，以下实例将快照文件 ubuntu.tar 导入到镜像 test/ubuntu:v1:

```
$ cat docker/ubuntu.tar | docker import - test/ubuntu:v1
```

```
$ cat docker/ubuntu.tar | docker import - test/ubuntu:v1
sha256:15bf275a8d44d7a39203163714df9985c7600958b7c85353c21a3889711ae75f

LINSHAODONG@HTJT-LSDONG MINGW64 /e/project
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
test/ubuntu         v1                  15bf275a8d44        3 seconds ago       64.2MB
redis               4-alpine            b77690502ebe        6 days ago          20.4MB
ubuntu              latest              cf0f3ca922e0        9 days ago          64.2MB
```

Docker镜像使用

列出镜像列表

我们可以使用 **docker images** 来列出本地主机上的镜像。

```
runoob@runoob:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              14.04               90d5884b1ee0        5 days ago          188 MB
php                 5.6                 f40e9e0f10c8        9 days ago          444.8 MB
nginx               latest              6f8d099c3adc        12 days ago         182.7 MB
mysql               5.6                 f2e8d6c772c0        3 weeks ago         324.6 MB
httpd               latest              02ef73cf1bc0        3 weeks ago         194.4 MB
ubuntu              15.10               4e3b13c8a266        4 weeks ago         136.3 MB
hello-world         latest              690ed74de00f        6 months ago        960 B
training/webapp     latest              6fae60ef3446        11 months ago       348.8 MB
```

各个选项说明:

- **REPOSITORY：**表示镜像的仓库源
- **TAG：**镜像的标签
- **IMAGE ID：**镜像ID
- **CREATED：**镜像创建时间
- **SIZE：**镜像大小

同一仓库源可以有多个 TAG，代表这个仓库源的不同个版本，如 ubuntu 仓库源里，有 15.10、14.04 等多个不同的版本，我们使用 REPOSITORY:TAG 来定义不同的镜像。

所以，我们如果要使用版本为15.10的ubuntu系统镜像来运行容器时，命令如下：

Docker镜像使用

查找镜像

我们可以从 Docker Hub 网站来搜索镜像，Docker Hub 网址为： https://hub.docker.com/

我们也可以使用 docker search 命令来搜索镜像。比如我们需要一个 httpd 的镜像来作为我们的 web 服务。我们可以通过 docker search 命令搜索 httpd 来寻找适合我们的镜像。

```
runoob@runoob:~$  docker search httpd
```

点击图片查看大图：

```
root@iZbp1193qvhd03n3bb5hd8Z:~# docker search httpd
NAME                                DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
httpd                               The Apache HTTP Server Project                  2703                [OK]
centos/httpd-24-centos7             Platform for running Apache httpd 2.4 or bui…   26
centos/httpd                                                                        26                                      [OK]
armhf/httpd                         The Apache HTTP Server Project                  8
arm64v8/httpd                       The Apache HTTP Server Project                  4
salim1983hoop/httpd24               Dockerfile running apache config                2                                       [OK]
lead4good/httpd-fpm                 httpd server which connects via fcgi proxy h…   1                                       [OK]
appertly/httpd                      Customized Apache HTTPD that uses a PHP-FPM …   0                                       [OK]
itsziget/httpd24                    Extended HTTPD Docker image based on the off…   0                                       [OK]
dockerpinata/httpd                                                                  0
```

NAME: 镜像仓库源的名称

DESCRIPTION: 镜像的描述

OFFICIAL: 是否 docker 官方发布

stars: 类似 Github 里面的 star，表示点赞、喜欢的意思。

AUTOMATED: 自动构建。

Docker镜像使用

拖取镜像

我们决定使用上图中的 httpd 官方版本的镜像，使用命令 docker pull 来下载镜像。

```
runoob@runoob:~$ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
8b87079b7a06: Pulling fs layer
a3ed95caeb02: Download complete
0d62ec9c6a76: Download complete
a329d50397b9: Download complete
ea7c1f032b5c: Waiting
be44112b72c7: Waiting
```

下载完成后，我们就可以使用这个镜像了。

```
runoob@runoob:~$ docker run httpd
```

删除镜像

镜像删除使用 **docker rmi** 命令，比如我们删除 hello-world 镜像：

```
$ docker rmi hello-world
```

```
$ docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Deleted: sha256:fce289e99eb9bca977dae136fbe2a82b6b7d4c372474c9235adc1741675f587e
Deleted: sha256:af0b15c8625bb1938f1d7b17081031f649fd14e6b233688eea3c5483994a66a3
```

Docker镜像使用

创建镜像

当我们从 docker 镜像仓库中下载的镜像不能满足我们的需求时，我们可以通过以下两种方式对镜像进行更改。

- 1、从已经创建的容器中更新镜像，并且提交这个镜像
- 2、使用 Dockerfile 指令来创建一个新的镜像

更新镜像

更新镜像之前，我们需要使用镜像来创建一个容器。

```
runoob@runoob:~$ docker run -t -i ubuntu:15.10 /bin/bash
```

进入容器后，更新系统：/p>

```
apt-get update
apt-get upgrade -y
```

在完成操作之后，输入 exit 命令来退出这个容器。

```
exit
```

此时 ID 为 e218edb10161 的容器，是按我们的需求更改的容器。我们可以通过命令 docker commit 来提交容器副本。

```
runoob@runoob:~$ docker commit -m="has update" -a="runoob" e218edb10161 runoob/ubuntu:v2
sha256:70bf1840fd7c0d2d8ef0a42a817eb29f854c1af8f7c59fc03ac7bdee9545aff8
```

Docker镜像使用

构建镜像

我们使用命令 **docker build**， 从零开始来创建一个新的镜像。为此，我们需要创建一个 Dockerfile 文件，其中包含一组指令来告诉 Docker 如何构建我们的镜像。

```
runoob@runoob:~$ cat Dockerfile
FROM    centos:6.7
MAINTAINER      Fisher "fisher@sudops.com"

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

每一个指令都会在镜像上创建一个新的层，每一个指令的前缀都必须是大写的。

第一条FROM，指定使用哪个镜像源

RUN 指令告诉docker 在镜像内执行命令，安装了什么。。。

然后，我们使用 Dockerfile 文件，通过 docker build 命令来构建一个镜像。

```
runoob@runoob:~$ docker build -t runoob/centos:6.7 .
Sending build context to Docker daemon 17.92 kB
Step 1 : FROM centos:6.7
 ---&gt; d95b5ca17cc3
Step 2 : MAINTAINER Fisher "fisher@sudops.com"
 ---&gt; Using cache
 ---&gt; 0c92299c6f03
Step 3 : RUN /bin/echo 'root:123456' |chpasswd
 ---&gt; Using cache
 ---&gt; 0397ce2fbd0a
Step 4 : RUN useradd runoob
......
```

Docker镜像使用

设置镜像标签

我们可以使用 docker tag 命令，为镜像添加一个新的标签。

```
runoob@runoob:~$ docker tag 860c279d2fec runoob/centos:dev
```

docker tag 镜像ID，这里是 860c279d2fec ,用户名称、镜像源名(repository name)和新的标签名(tag)。

使用 docker images 命令可以看到，ID为860c279d2fec的镜像多一个标签。

```
runoob@runoob:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
runoob/centos       6.7                 860c279d2fec        5 hours ago         190.6 MB
runoob/centos       dev                 860c279d2fec        5 hours ago         190.6 MB
runoob/ubuntu       v2                  70bf1840fd7c        22 hours ago        158.5 MB
ubuntu              14.04               90d5884b1ee0        6 days ago          188 MB
php                 5.6                 f40e9e0f10c8        10 days ago         444.8 MB
nginx               latest              6f8d099c3adc        13 days ago         182.7 MB
mysql               5.6                 f2e8d6c772c0        3 weeks ago         324.6 MB
httpd               latest              02ef73cf1bc0        3 weeks ago         194.4 MB
ubuntu              15.10               4e3b13c8a266        5 weeks ago         136.3 MB
hello-world         latest              690ed74de00f        6 months ago        960 B
centos              6.7                 d95b5ca17cc3        6 months ago        190.6 MB
training/webapp     latest              6fae60ef3446        12 months ago       348.8 MB
```