

docker

C-3 创建: 许荣仁, 最后修改: 许荣仁 2019-06-03 16:49

基本概念

- 仓库：是集中存放镜像的地方。
- 镜像：特殊的文件系统，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的一些配置参数（如匿名卷、环境变量、用户等）。镜像不包含任何动态数据，其内容在构建之后也不会被改变。
- 容器：镜像运行时的实体。
- 数据存储：数据卷和本地目录。

基本配置

Linux中要先启动，或者设置开机自启

```
systemctl start docker
```

国内加速器 Registry mirrors: <https://dockerhub.azk8s.cn>

镜像

拉取镜像

```
docker pull [OPTIONS] NAME[:TAG]
```

options:

- a : 拉取所有 tagged 镜像
- disable-content-trust : 忽略镜像的校验，默认开启

NAME格式:

[Docker Registry 地址[:端口号]]/仓库名

- Docker 镜像仓库地址：地址的格式一般是 <域名/IP>[:端口号]。默认地址是 Docker Hub。仓库名：两段式名称，即 <用户名>/<软件名>。对于 Docker Hub，如果不给出用户名，则默认为 library，也就是官方镜像。例如：

```
docker pull ubuntu:18.04
```

- 命令中没有给出 Docker 镜像仓库地址，因此将会从 Docker Hub 获取镜像。
- 没有给出用户名，因此将会获取官方镜像 library/ubuntu。
- 标签为18.04。如不指定标签，默认为latest。

列出本地镜像

```
docker image ls
```

```
C:\Users\rrxu
λ docker image list
REPOSITORY              TAG         IMAGE ID      CREATED      SIZE
docker.elastic.co/logstash/logstash  7.0.1      55b534744e48  2 weeks ago  820MB
sebp/elk                 latest     c6e5137650c3  5 months ago  1.56GB
sebp/elk                 502       252b1a67ca48  2 years ago  969MB
```

删除本地镜像

```
docker image rm [option] <img1> [<img2> ...]
docker rmi [option] <img1> [<img2> ...]
```

可以是name或ID，ID一般只要前几位，能够区分不同镜像就可以

自定义镜像

容器

列出所有容器

```
docker ps [option]
```

默认列出正在运行的容器

创建新的容器

```
docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]
```

常用参数 [options]

```
-p 端口映射 主机(宿主)端口:容器端口
-v 挂载目录 主机目录:容器目录
-d 后台运行
```

```
--name 指定一个名字
```

开始与结束容器

```
docker start container_name
docker stop container_name
docker kill container_name #强制结束
docker restart container_name #重启
```

查看占用的空间

`docker system df`

自定义镜像

commit 镜像

1. 进入 webserver 容器，作出修改

```
docker exec -it webserver bash
```

```
echo '<h1>Hello, Docker!</h1>' > /usr/share/nginx/html/index.html
```

```
exit
```

2. 使用 `docker commit` 命令将容器存储层的内容保存下来构建成新的镜像

```
docker commit --author "rrxu" --message "change index html" webserver nginx:v2
```

格式：`docker commit [选项] <容器ID或容器名> [<仓库名>[:<标签>]]`

3. 用新的镜像创建一个容器

```
docker run --name web2 -d -p 81:80 nginx:v2
```

- 慎用 `docker commit`

使用 Dockerfile 定制镜像

镜像的定制实际上就是定制每一层所添加的配置、文件。

Dockerfile 是一个文本文件，其内包含了一条条的指令，每一条指令构建一层，因此每一条指令的内容，就是描述该层应当如何构建。

- 在一个空白目录中，建立一个文本文件，并命名为 Dockerfile：

```
FROM nginx
RUN echo '<h1>Hello, Docker! I am from Dockerfile.</h1>' > /usr/share/nginx/html/index.html
```

FROM 指定基础镜像

RUN 执行命令

- 每一个指令都会建立一层。每一个 RUN 的行为，和手工建立镜像的过程一样：新建一层，在其上执行这些命令，执行结束后，commit 这一层的修改，构成新的镜像。
- Union FS 是有最大层数限制的，比如 AUFS，曾经是最大不得超过 42 层，现在是不得超过 127 层。
- 在撰写 Dockerfile 的时候，要记得这并不是在写 Shell 脚本，而是在定义每一层该如何构建，应当确保自己在每一层只添加了真正需要的东西，并在最后清理无关文件。
- 举个例子

```
FROM debian:stretch
```

```
RUN apt-get update
RUN apt-get install -y gcc libc6-dev make wget
RUN wget -O redis.tar.gz "http://download.redis.io/releases/redis-5.0.3.tar.gz"
RUN mkdir -p /usr/src/redis
RUN tar -xzf redis.tar.gz -C /usr/src/redis --strip-components=1
RUN make -C /usr/src/redis
RUN make -C /usr/src/redis install
```

```
FROM debian:stretch
```

```
RUN buildDeps='gcc libc6-dev make wget' \
  && apt-get update \
  && apt-get install -y $buildDeps \
  && wget -O redis.tar.gz "http://download.redis.io/releases/redis-5.0.3.tar.gz" \
  && mkdir -p /usr/src/redis \
  && tar -xzf redis.tar.gz -C /usr/src/redis --strip-components=1 \
  && make -C /usr/src/redis \
  && make -C /usr/src/redis install \
  && rm -rf /var/lib/apt/lists/* \
  && rm redis.tar.gz \
  && rm -r /usr/src/redis \
  && apt-get purge -y --auto-remove $buildDeps
```

两种书写方式构建出镜像的体积差异：

```
λ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
redis	v2	c183a96e0627	2 minutes ago	138MB
redis	v1	321dd50da8a3	About an hour ago	403MB

数据管理

数据卷 是一个可供一个或多个容器使用的特殊目录

- 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
- 数据卷 默认会一直存在，即使容器被删除

创建数据卷

```
docker volume create newvol
```

查看所有的 数据卷

```
docker volume ls
```

启动一个挂载数据卷的容器

```
docker run -d -p 85:80 --name web -v newvol:/webapp nginx
```

在Linux下可以从 `docker` 的路径下查看数据卷中的内容，从而在宿主机中直接查看管理数据卷的内容，但在Windows中不能直接管理，需要使用容器访问虚拟机的文件系统 [SSH into mobylinux \(docker for windows\)](#)

删除数据卷

```
docker volume rm newvol
```

数据卷 是被设计用来持久化数据的，它的生命周期独立于容器，Docker 不会在容器被删除后自动删除 数据卷，并且也不存在垃圾回收这样的机制来处理没有任何容器引用的 数据卷。如果需要在删除容器的同时移除数据卷。可以在删除容器的时候使用 `docker rm -v` 这个

参考资料

[docker —— 从入门到实践](#)