docker 3.镜像、Dockerfile

什么是镜像?

我们都知道，操作系统分为内核和用户空间。对于 Linux 而言，内核启动后，会挂载 `root` 文件系统为其提供用户空间支持。而 Docker 镜像（Image），就相当于是一个 `root` 文件系统。比如官方镜像 `ubuntu:17.10` 就包含了完整的一套 Ubuntu 17.10 最小系统的 `root` 文件系统。

Docker 镜像是一个特殊的文件系统，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。

分层存储

因为镜像包含操作系统完整的 `root` 文件系统，其体积往往是庞大的，因此在 Docker 设计时，就充分利用 Union FS 的技术，将其设计为分层存储的架构。所以严格来说，镜像并非是像一个 ISO 那样的打包文件，镜像只是一个虚拟的概念，其实际体现并非由一个文件组成，而是由一组文件系统组成，或者说，由多层文件系统联合组成。

镜像构建时，会一层层构建，前一层是后一层的基础。每一层构建完就不会再发生改变，后一层上的任何改变只发生在自己这一层。比如，删除前一层文件的操作，实际不是真的删除前一层的文件，而是仅在当前层标记为该文件已删除。在最终容器运行的时候，虽然不会看到这个文件，但是实际上该文件会一直跟随镜像。因此，在构建镜像的时候，需要额外小心，每一层尽量只包含该层需要添加的东西，任何额外的东西应该在该层构建结束前清理掉。

分层存储的特征还使得镜像的复用、定制变的更为容易。甚至可以用之前构建好的镜像作为基础层，然后进一步添加新的层，以定制自己所需的内容，构建新的镜像。

镜像常用相关操作命令:

build　构建镜像(docker build -t xxx:tag -- file path)

history　历史构建过程(docker history xxx)

images　查看所有镜像(docker images)

import　导入从export导出的容器作为镜像(docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]])

load　从stdin或者压缩文件中加载镜像(docker load -i filename)

pull　pull镜像(docker pull [REPOSITORY[:TAG]])

push　push镜像(docker push [REPOSITORY[:TAG]])

rmi　删除镜像（前提是镜像未被容器引用）(docker rmi xxx)

save 导出镜像到压缩文件(docker save xxx1 xxx2 xxx3 -o name)

search 在docker hub搜索镜像(docker search xxx)

tag 镜像打版本号(docker tag xxx [REPOSITORY[:TAG]])

Dockerfile

Dockerfile 是一个文本文件，其内包含了一条条的指令**(Instruction)**，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。

Dockerfile 常用相关命令

COPY 复制文件

ADD 更高级的复制文件（可以是url下载的文件，自动解压）

RUN 执行命令

ENV 指定run的环境变量

ARG 构建参数

EXPOSE 暴露端口

WORKDIR 指定工作目录

USER 创建指定用户

ENTRYPOINT 入口点（容器第一次启动才会调用）

CMD 容器启动命令（run时可以覆盖）

Dockerfile 构建一个镜像

（构建镜像需要用FROM指定镜像源，就像类的继承，需要指定superclass）

通过dev_env这个仓库的cas镜像讲解：ssh://git@192.168.1.251:7999/dok/dev_env.git

官方**mysql 5.6**的**Dockerfile**：https://github.com/docker-library/mysql/blob/883703dfb30d9c197e0059a669c4bb64d55f6e0d/5.6/Dockerfile

```
FROM debian:jessie

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever dependencies get added
RUN groupadd -r mysql && useradd -r -g mysql mysql
```

```
5
6  # add gosu for easy step-down from root
7  ENV GOSU_VERSION 1.7
8  RUN set -x \
9      && apt-get update && apt-get install -y --no-install-recommends ca-certificates wget
   && rm -rf /var/lib/apt/lists/* \
10     && wget -O /usr/local/bin/gosu
   "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --print-
   architecture)" \
11     && wget -O /usr/local/bin/gosu.asc
   "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --print-
   architecture).asc" \
12     && export GNUPGHOME="$(mktemp -d)" \
13     && gpg --keyserver ha.pool.sks-keyservers.net --recv-keys
   B42F6819007F00F88E364FD4036A9C25BF357DD4 \
14     && gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu \
15     && rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc \
16     && chmod +x /usr/local/bin/gosu \
17     && gosu nobody true \
18     && apt-get purge -y --auto-remove ca-certificates wget
19
20 RUN mkdir /docker-entrypoint-initdb.d
21
22 # FATAL ERROR: please install the following Perl modules before executing
   /usr/local/mysql/scripts/mysql_install_db:
23 # File::Basename
24 # File::Copy
25 # Sys::Hostname
26 # Data::Dumper
27 RUN apt-get update && apt-get install -y perl pwgen --no-install-recommends && rm -rf
   /var/lib/apt/lists/*
28 RUN set -ex; \
29     # gpg: key 5072E1F5: public key "MySQL Release Engineering " imported
30     key='A4A9406876FCBD3C456770C88C718D3B5072E1F5'; \
31     export GNUPGHOME="$(mktemp -d)"; \
32     gpg --keyserver ha.pool.sks-keyservers.net --recv-keys "$key"; \
33     gpg --export "$key" > /etc/apt/trusted.gpg.d/mysql.gpg; \
34     rm -r "$GNUPGHOME"; \
35     apt-key list > /dev/null
36
37
38 ENV MYSQL_MAJOR 5.6
39 ENV MYSQL_VERSION 5.6.38-1debian8
40 RUN echo "deb http://repo.mysql.com/apt/debian/ jessie mysql-${MYSQL_MAJOR}" >
   /etc/apt/sources.list.d/mysql.list
41
42
43 # the "/var/lib/mysql" stuff here is because the mysql-server postinst doesn't have an
   explicit way to disable the mysql_install_db codepath besides having a database already
   "configured" (ie, stuff in /var/lib/mysql/mysql)
44 # also, we set debconf keys to make APT a little quieter
45 RUN { \
46         echo mysql-community-server mysql-community-server/data-dir select ''; \
47         echo mysql-community-server mysql-community-server/root-pass password ''; \
48         echo mysql-community-server mysql-community-server/re-root-pass password ''; \
49         echo mysql-community-server mysql-community-server/remove-test-db select false;
```

```
\
50      } | debconf-set-selections \
51      && apt-get update && apt-get install -y mysql-server="${MYSQL_VERSION}" && rm -rf
/var/lib/apt/lists/* \
52      && rm -rf /var/lib/mysql && mkdir -p /var/lib/mysql /var/run/mysqld \
53      && chown -R mysql:mysql /var/lib/mysql /var/run/mysqld \
54      # ensure that /var/run/mysqld (used for socket and lock files) is writable
regardless of the UID our mysqld instance ends up having at runtime
55      && chmod 777 /var/run/mysqld \
56      # comment out a few problematic configuration values
57      && find /etc/mysql/ -name '*.cnf' -print0 \
58      | xargs -0 grep -lZE '^(bind-address|log)' \
59      | xargs -rt -0 sed -Ei 's/^(bind-address|log)/#&/' \
60      # don't reverse lookup hostnames, they are usually another container
61      && echo '[mysqld]\nskip-host-cache\nskip-name-resolve' >
/etc/mysql/conf.d/docker.cnf
62
63 VOLUME /var/lib/mysql
64 COPY docker-entrypoint.sh /usr/local/bin/
65 RUN ln -s usr/local/bin/docker-entrypoint.sh /entrypoint.sh # backwards compat
66 ENTRYPOINT ["docker-entrypoint.sh"]
67
68 EXPOSE 3306
69 CMD ["mysqld"]
```