1. 镜像的特性
2. 镜像的ID

容器创建时需要指定镜像，每个镜像都有唯一的标识Image ID，和容器的Container ID一样，默认是128位的，可以使用前16位缩写形式，也可以使用镜像名+版本号的组合形式，如果省略版本号，则使用最新版本。

1. 镜像的分层

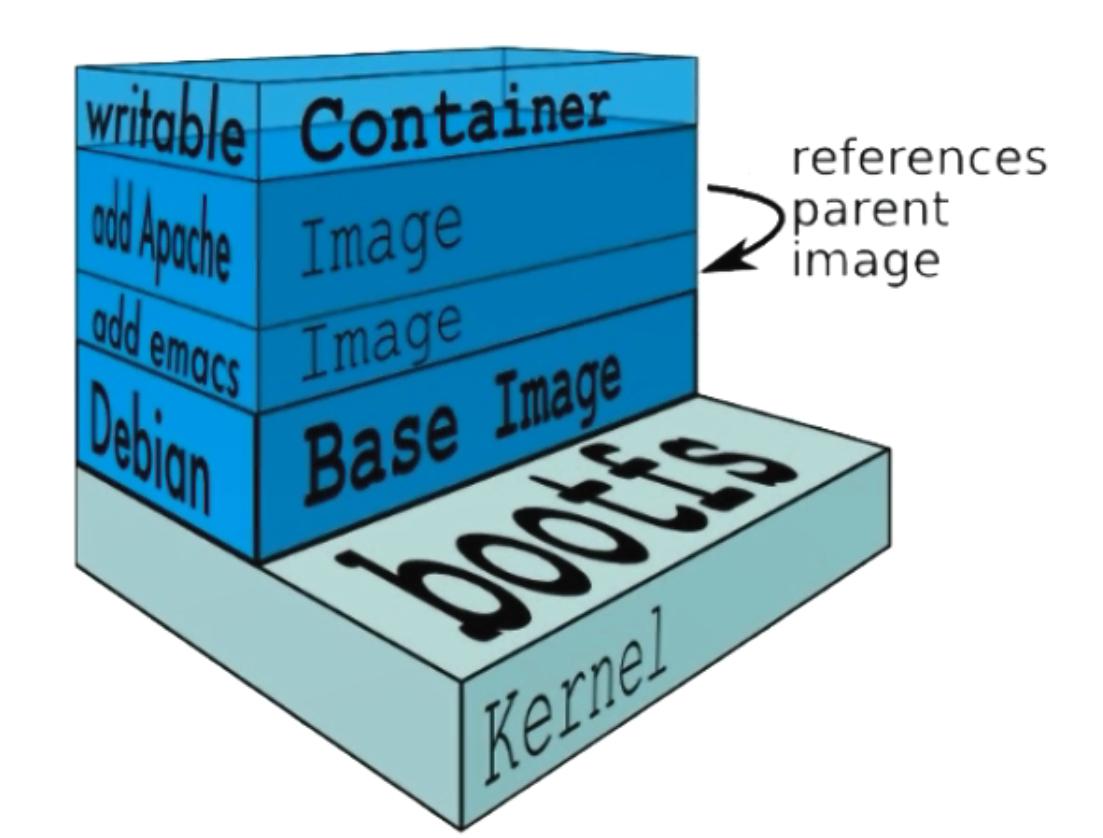
之前说过镜像是分层的，通过联合文件系统（union filesystem，ufs）将各层文件系统叠加在一起，对外展示为一个统一的整体。对于已有的镜像分层是只能读而不能修改的，这就保证了当一个镜像生成容器后，该镜像不会被改变；且如果多层镜像中有相似的内容，那么容器在获取该内容时，上层镜像的优先级是大于底层镜像的。

1. bootfs

首先加载的bootfs，这个是用于系统引导的文件系统，包括bootloader和kernel，主要作用就是用于引导系统的，加载内核，传统模式和Docker中都会存在，容器启动完成后会被卸载以节省内存资源，之后就会引入rootfs。

1. rootfs

位于bootfs之上，表现为Docker容器的根文件系统，在bootfs加载完成后加载，获取了内核的指令，然后对整个操作系统进行自检操作。在传统模式中，系统启动时，内核挂载rootfs时会首先将其挂载为“只读”模式，完成完整性自检后会将其挂载为读写模式。而在Docker中，rootfs始终挂载为只读模式，而后通过UFS技术挂载一个可写层在rootfs之上，用于读写操作，所有的容器读写操作都在读写层中，不会影响rootfs的根文件系统，即不会对镜像造成影响。



1. DockerFile

通常，我们会自己构建镜像，生成镜像的方式有两种，一种是容器直接转镜像，另一种是使用DockerFile。注意，一个镜像在生成容器运行时，至少容器中得有一个工作在前台的守护进程，如果一个镜像中干净地连一个前台守护进程都没有，那么由该镜像生成的容器就会自动退出。如Nginx通过配置daemon来转为前台运行，如mysqld这个前台的守护进程，只要有一个前台守护进程在镜像中，那么就不会自动退出。每个镜像在启动为容器时，都会使用一个命令/bin/bash，该命令就是寻找前台的守护进程的，如果没有找到，则会默认退出。

1. 容器和镜像的转换

容器与镜像是可以互相转换的，要将镜像转换为容器，即开启一个新的容器即可，使用docker run命令；要将容器转换为镜像，使用docker commit命令即可，就可以将一个容器转换为Docker镜像。

docker commit CID xx.xx.xx

CID是容器的ID号

xx.xx是镜像的标签号

commit命令只是将容器当时的状态保存为镜像，但是它并不知道将该镜像运行为容器的时候应该怎么做。所以它依然会沿用镜像中的启动命令，如果我们想要让该镜像在运行为容器时连带着启动镜像中的其他软件，那么就需要将启动命令替换为启动软件的命令。commit命令只能说是一个备份的作用，而DockerFile就可以通过脚本来实现容器的定制化启动。

1. DockerFile
2. 概述

DockerFile相当于创建Docker镜像的语法，它是一种被Docker程序解释的脚本，由一条条的指令组成，每条指令对应Linux下面的一条命令，Docker程序将这些DockerFile指令翻译成真正的Linux命令，DockerFile有自己的书写格式和支持的命令，Docker程序解决这些命令之间的依赖关系，类似于MakeFile。Docker程序将读取DockerFile，根据指令生成定制的镜像，生成命令为：

docker build -t xx

1. DockerFile语法
2. FROM image

构建指令，必须指定且 DockerFile的其他指令之前，后续的命令都是依赖于该指令的image，每个镜像都会依赖于一个基础镜像image，可以是官方镜像，也可以是本地仓库的镜像，但是该命令只能用一次，只能引入一个基础镜像。

FROM centos:7.2

1. MAINTAIN

构建指令，指定该镜像的制作者信息，当我们对该镜像执行docker inspect命令时，会输出该信息。

MAINTAIN [wxx@qq.com](mailto:wxx@qq.com)

1. RUN（用于安装软件）

构建指令，RUN可以运行任何被基础image支持的命令，如基础image选择了Centos，那么软件管理部分就只能使用Centos的包管理命令。

RUN cd /tmp && echo “exx”

1. CMD （设置容器启动时执行的操作）

设置指令，用于容器启动时指定的操作，该操作可以是执行自定义脚本，也可以是执行系统命令，该指令只能在文件中存在一次，如果有多个，只执行最后一条。如果要执行多次操作，就将多次操作使用&&连接起来。

CMD echo “wxx”

1. ENTRYPOINT（设置容器启动时执行的操作）

设置指令，指定容器启动时执行的命令，可多次设置，但是只执行最后一个。

ENTRYPOINT ls -l

该命令可以和CMD配合使用，前提是CMD不是一个完整的可执行命令，仅仅代表命令的参数部分，而ENTRYPOINT代表着命令的命令部分，而不能指定参数，注意，使用的是JSON格式。如下所示：

FROM ubuntu

CMD [“-l”]

ENTRYPOINT [“/usr/bin/ls”]

如果CMD和ENTRYPOINT都使用完整的命令，那么只有一个会执行。

1. USER（设置容器的用户）

设置指令，默认是root用户：

USER daemon=ENTRYPOINT[“memcached”, “-u”, “daemon”]

1. EXPOSE

指定容器需要暴露的端口A，如果该端口需要和宿主机端口进行映射，则该端口A会映射到宿主机器的某个端口B中。这是一个设置指令，当我们需要访问该容器时，可以直接指定-p选项访问容器，如果是通过宿主机的端口映射来访问的，则通过宿主机的端口B映射到容器的端口A来访问的，在运行容器时指定-p选项加上 B:A即可。

EXPOSE port1

运行时： docker run -p port1 image

EXPOSE port1, port2, port3

运行时： docker run -p port1, -p port2, -p port3

与宿主机的映射：

docker run -p B:A image

1. ENV（设置环境变量）

构建指令，在镜像中设置环境变量。后续的RUN命令都可以使用该变量，容器启动后，可以通过docker inspect来查看这个环境变量，也可以通过在

docker run –env key=value

来设置或者修改环境变量。

例如，安装了Java程序，需要设置JAVA\_HOME，则设置：

ENV JAVA\_HOME /path/to/java

1. ADD（复制文件，下载文件，解压文件）

从URL下载文件，或者从源地址复制文件，解压文件到目的地。

ADD <src> <dest>

src是相对被构建的源目录的相对路径，可以是文件或目录的路径，也可以是一个URL，可以下载；

dest 是容器中的绝对路径；

1. COPY

也是复制文件，但是不支持下载文件以及解压文件。

1. WORKDIR（切换目录）

设置指令，可以多次切换，相当于cd命令，可以在RUN，CMD，ENTRYPOINT中使用。

WORKDIR /P1 WORKDIR /P2 RUN vim a.txt

1. ONBUILD（子镜像中执行）

指定的命令在构建的当前镜像中不会执行，也不会在构建的容器中执行，如果我们将该镜像作为基础镜像再来构建一个新的镜像时，这些命令就会执行了，可以用于数据保护等操作，保护子镜像中的数据等。

ONBUILD ADD /APP/SRC

1. VOLUME（指定挂载点）

设置指令，使容器中的一个目录具有持久化存储数据的功能，该目录可以被容器本身使用，也可以共享给其他容器使用。由于容器使用的是UFS，这种文件系统不能持久化数据，当容器关闭后，所有的更改都会丢失，当容器中的应用有持久化数据的需求时，可以在DockerFile中使用该命令：

VOLUME [“/tmp/data”]

1. 注意点

DockerFile最多不超过128层，因为每一层指令都会更新为一个新的镜像，所有的镜像最终会堆叠在基础镜像之上，形成最终的镜像。