**1.Docker镜像**

**1.1相关概念：**registry ：用于保存Docker镜像，包括镜像层次结构和镜像元数据，类似于git仓库之类的实体。

                   repository：某个Docker镜像所有迭代版本的构成的镜像组。registry是repository的合集，repository是镜像的合集。

                   manifest：存在于registry中作为Docker镜像的元数据文件。镜像被pull到宿主机时，自动转化为本地的镜像配置文件。

                   image / layer：docker内部的image概念是用来存储一组镜像相关的元数据，包括镜像架构，默认配置等，包含所有镜像层信息的rootfs。layer是Docker用来管理镜像层的中间概念，因为单个镜像层可能被多个镜像使用，所以docker把layer和image的概念分开，layer存放了镜像层的diff\_id，size，parent\_id等内容。

                   Dockerfile：构建镜像时用到的定义文件。

**1.2相关操作：**build / pull / push / save / export / load

**1.3镜像存储管理：**Docker镜像设计上将镜像元数据和镜像文件的存储完全分开，Docker在管理镜像层元数据时采用的是从上至下repository，image，layer三个层次。因为docker以分层的形式存储镜像，因此repository和image这两类元数据没有物理上的镜像文件与之对应，而layer则存在物理上的镜像文件与之对应。

    repository元数据：位置->/var/lib/docker/image/aufs(graph\_driver)/repositories.json，存储了所有repository的名字，tag和对应的镜像ID

    image元数据：位置->/var/lib/docker/image/aufs/imagedb/content/sha256/镜像ID，各种镜像相关信息

   layer元数据： 位置->/var/lib/docker/image/aufs/layerdb/sha256，包括diff\_id，parent，size等信息，其中diffid是通过镜像层内容用sha256算法计算得来。layerdb目录下还有一个mount目录，用于存储容器可读写层的相关信息。

**1.4镜像实际文件存放:**

    docker的镜像管理和存储是完全隔离的，以aufs为例，进入/var/lib/docker下有三个文件夹，分别为diff/，layers/，mnt/。其中mnt为aufs的挂载目录，diff为实际的数据来源，包括只读层和可读写层，layers存放层依赖有关的层描述文件。mnt和layers最初都是空目录，文件数据在diff下，但随后创建容器时mnt和layers目录会慢慢更新。

**容器创建过程中的层存储变化：**容器创建时会随机生成容器读写层的mountID，并保存在image/aufs/layerdb/mounts/容器id/目录下 ->

                在mnt和diff目录下创建以该容器层的mountID同名的子文件夹 ->

                在layers目录下创建文件来记录该层依赖的**其他镜像层** ->

                GraphDriver将diff中容器依赖的所有镜像层目录挂载到mnt目录下，并会生成一个对应当前容器的以mountID-init格式命名的文件夹作为容器的最后一个**只读层** ->

                diff目录下的mountID命名的子目录作为容器的**可读写层**，也挂载到mnt目录下。

    docker容器设计了mountID-init这一层来记录容器启动时根据系统环境和用户配置自动生成的内容，一般情况下，容器commit操作保存的镜像不会包含这一层内容。严格意义上来讲容器的文件系统有三层：可读写层 / init层 / 只读层。

    docker ps -a即可看到容器短id，然后根据这个短id去  var/lib/docker/image/aufs/layerdb/mounts/短id.../ 这个路径下可以查看当前容器的init-id mount-id等。容器未启动时，在对应的mnt目录下的mountID文件夹（/var/lib/docker/aufs/mnt/[mountID]/）是空的,容器启动后，可以看到这个文件夹下挂载了对应的系统文件（/bin /home /proc等等）。如果用户在容器中对容器进行一些操作，这些操作都会被记录在这个文件夹中，当容器被停止时，mnt下对应的mountID目录被卸载，但diff下对应的mountID文件夹还存在，但只限于宿主机。因此如果我们用docker commit把容器提交成镜像时其实就是在diff目录下生成一个新的cacheID命名的文件夹存放最新的差异变化，这样一个新的镜像层就产生了，而原来的diff下以mountID为名的文件夹在容器被删除时也会对应的被删除。