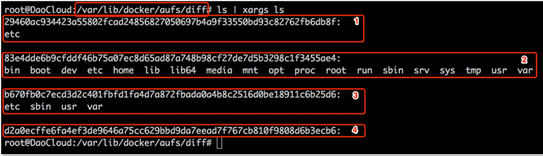
**Docker镜像**

Docker images

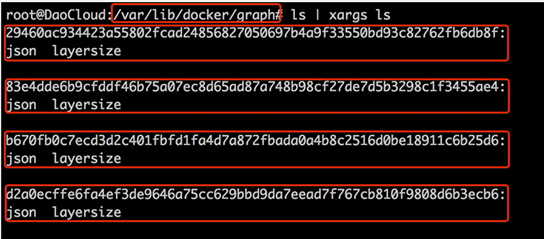
Docker rmi [image\_name/image\_id]

传统虚拟机安装ubuntu:14.04会包含两部分，第一，某一个Linux内核的发行版本，比如Linux 3.8版本的内核；第二，第一个特定的Ubuntu发行版，这部分内容不包含Linux内核，但是包含Linux之外的软件管理方式，软件驱动，如 apt-get软件管理包等。

Linux内核＋ubuntu操作系统发行版，组成一台工作的机器让用户体验。那么灵活替换ubuntu操作系统发行版，那是不是也可以实现呢。那么Docker很方便的利用了这一点，技术手段就是Docker镜像。

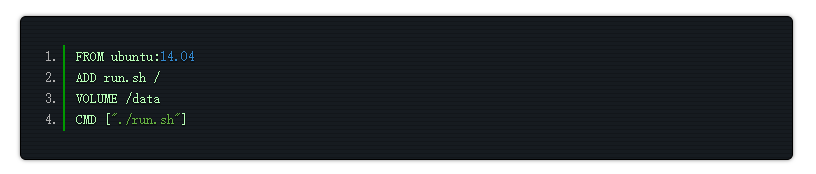


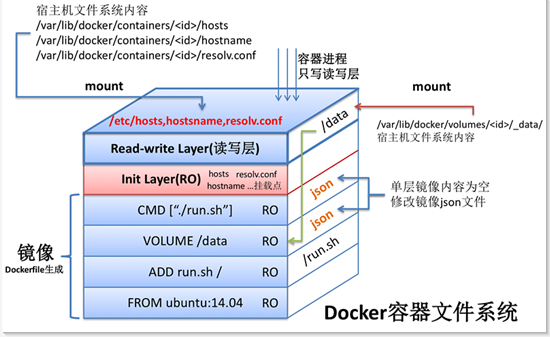
研究镜像与容器的关系：容器是一个动态的环境，每一层镜像中的文件属于静态内容，然而 Dockerfile 中的 ENV、VOLUME、CMD 等内容最终都需要落实到容器的运行环境中，而这些内容均不可能直接坐落到每一层镜像所包含的文件系统内容中，那此时每一个Docker镜像还会包含 json文件记录与容器之间的关系。



因此，Docker镜像的内容主要包含两个部分：第一，镜像层文件内容；第二，镜像json文件。

简单来讲，Dockerfile构建出Docker镜像，通过Docker镜像运行Docker容器。





FROM ubuntu:14.04：设置基础镜像，此时会使用基础镜像 ubuntu:14.04 的所有镜像层，为简单起见，图中将其作为一个整体展示。

ADD run.sh /：将 Dockerfile 所在目录的文件 run.sh 加至镜像的根目录，此时新一层的镜像只有一项内容，即根目录下的 run.sh。

VOLUME /data：设定镜像的 VOLUME，此 VOLUME 在容器内部的路径为 /data。需要注意的是，此时并未在新一层的镜像中添加任何文件，即构建出的磁层镜像中文件为空，但更新了镜像的 json 文件，以便通过此镜像启动容器时获取这方面的信息。

CMD ["./run.sh"]：设置镜像的默认执行入口，此命令同样不会在新建镜像中添加任何文件，仅仅在上一层镜像 json 文件的基础上更新新建镜像的 json 文件。

因此，通过以上分析，以上的Dockerfile可以构建出一个新的镜像，包含4个镜像层，每一条命令会和一个镜像层对应，镜像之间会存在父子关系。图中很清楚的表明了这些关系。

大家可以把Docker容器，理解为一个或多个运行进程，而这些运行进程将占有相应的内存，相应的CPU计算资源，相应的虚拟网络设备以及相应的文件系统资源。而Docker容器所占用的文件系统资源，则通过Docker镜像的镜像层文件来提供。

那么作为静态的镜像，如何才有能力转化为一个动态的Docker容器呢？此时，我们可以想象：第一，转化的依据是什么；第二，由谁来执行这个转化操作。

其实，转化的依据是每个镜像的json文件，Docker可以通过解析Docker镜像的json的文件，获知应该在这个镜像之上运行什么样的进程，应该为进程配置怎么样的环境变量，此时也就实现了静态向动态的转变。

谁来执行这个转化工作？答案是Docker守护进程。也许大家早就理解这样一句 话：Docker容器实质上就是一个或者多个进程，而容器的父进程就是Docker守护进程。这样的，转化工作的执行就不难理解了：Docker守护进程 手握Docker镜像的json文件，为容器配置相应的环境，并真正运行Docker镜像所指定的进程，完成Docker容器的真正创建。

Docker容器运行起来之后，Docker镜像json文件就失去作用了。此时Docker镜像的绝大部分作用就是：为Docker容器提供一个文件系统的视角，供容器内部的进程访问文件资源。

Docker守护进程会在Docker镜像的 最上层之上，再添加一个可读写层，容器所有的写操作都会作用到这一层中。而如果Docker容器需要写底层Docker镜像中的文件，那么此时就会涉及一 个叫Copy－on－Write的机制，即aufs等联合文件系统保证：首先将此文件从Docker镜像层中拷贝至最上层的可读写层，然后容器进程再对读 写层中的副本进行写操纵。对于容器进程来讲，它只能看到最上层的文件。

Docker容器中不存在Docker镜像中的内容主要有以下几点：

1. /proc以及/sys等虚拟文件系统的内容

2. 容器的hosts文件，hostname文件以及resolv.conf文件，这些事具体环境的信息，原则上的确不应该被打入镜像。

3. 容器的Volume路径，这部分的视角来源于从宿主机上挂载到容器内部的路径

4. 部分的设备文件

导出docker镜像为文件：

docker save ubuntu:14.04 -o ubuntu