

镜像是Docker容器的基石，容器是镜像的运行实例，有了镜像才能启动容器。base镜像----提供了一个基本的操作系统环境，用户可以根据需要安装和配置软件

base镜像通常是各种Linux发行版的Docker镜像比如ubuntu、Debian、centos等。

base镜像只是在用户空间与发行版一致，kernel版本与发行版是不同的。比如说虚机系统是ubuntu14.04，运行一个容器是centos系统的，那么centos使用的其实就是本虚机的kernel。Docker使用主机的内核运行容器中的程序。传统虚拟机方式运行 10 个不同的应用就要起 10 个虚拟机，而Docker 只需要启动 10 个隔离的应用即可。

容器只能使用本地虚机的kernel且不能修改。

每一个镜像都会有一个文本文件Dockerfile，定义了如何构建Docker镜像。编写Dockerfile文件有专门的讲解，百度一下~

镜像的分层结构：DockerHub中绝大多数的镜像都是通过在base镜像中安装和配置需要的软件构建出来的。base镜像是from scratch开始，新镜像是直接在base镜像上构建from centos......

新镜像是从base镜像一层一层叠加生成的，每安装一个软件就等于在现有的镜像上增加一层。

Docker镜像采用分层结构的好处就是共享资源。

可写的容器层：当容器启动时，一个新的可写层被加载到镜像的顶部，这一层就叫容器层，容器层之下都叫镜像层。只有容器层是可写的，容器层下面的所有镜像层都是只读的。对容器的任何改动都只会发生在容器层中。

这里，所有的镜像层联合一起组成一个统一的文件系统，用户在容器层看到的就是一个叠加之后的文件系统。镜像层内部是有上下之分的：

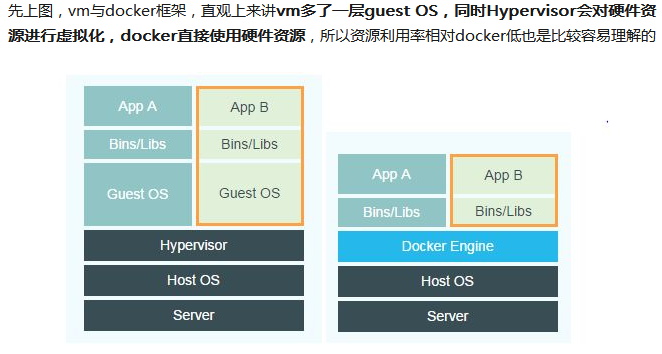
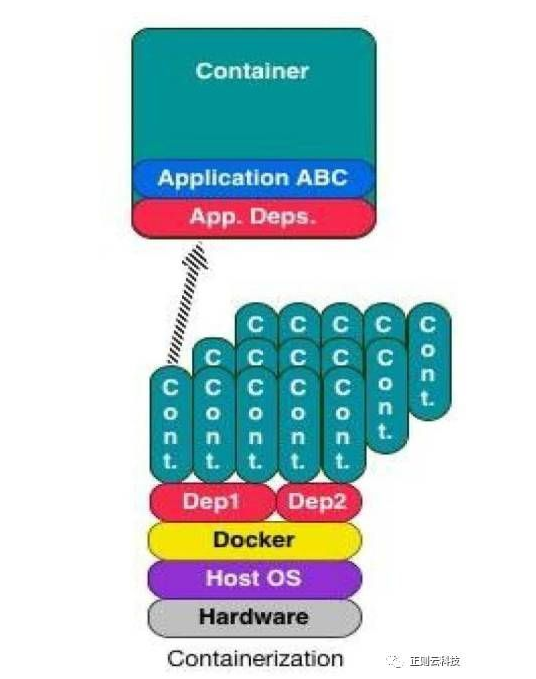
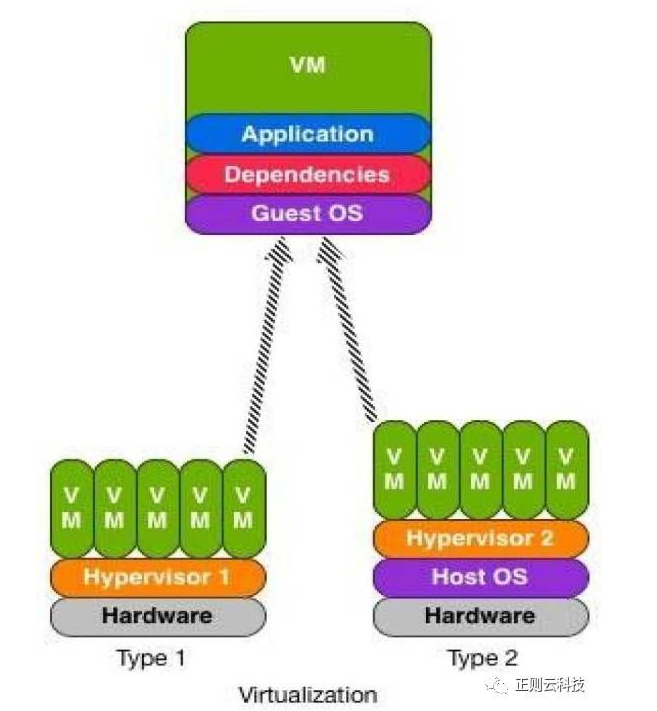
- 添加文件：在容器中创建文件时，新文件被添加到容器层中。

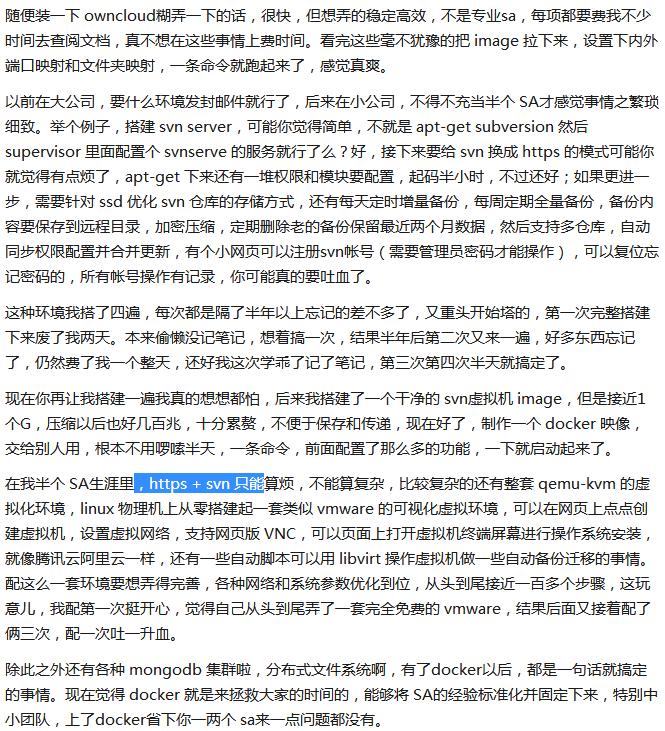
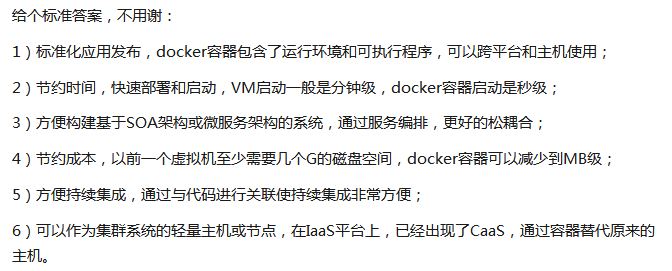
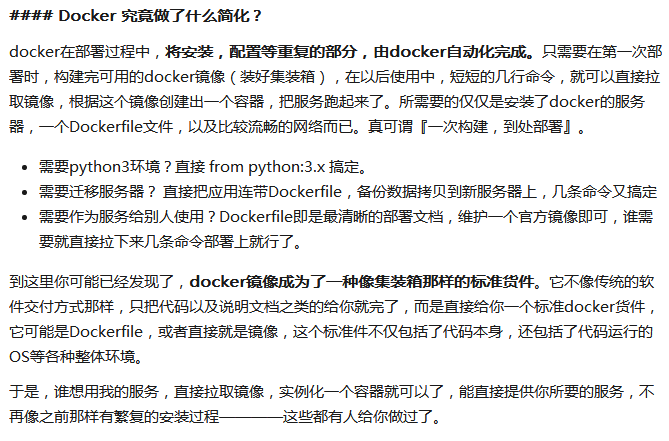
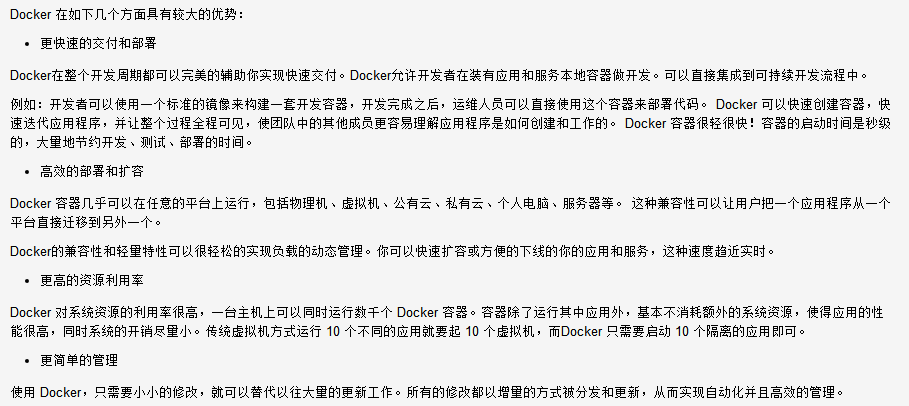
- 读取文件：当在容器中读取某个文件时，Docker会从上往下依次在各镜像层中查找此文件，一旦找到打开并读入内存。

- 修改文件：在容器中修改已存在的文件时，Docker会从上往下依次在各个镜像层中查找此文件，一旦找到立即将其复制到容器层中，然后才修改。（copy-on-write特性）

- 删除文件：在容器中删除文件时，Docker会从上往下依次在镜像层中找，找到后，会在容器层记录下此删除操作。

copy-on-write特性说明容器层保存的是镜像变化的部分，不会对镜像本身进行任何修改。所以镜像可以被多个容器共享。

但是最根本的差别，其实在于容器是轻量的，体现在容器的秒级启动和占用资源少。至于为什么容器是轻量的，是怎么做到了，前面很多回复都说明了，就不多说了。但很多人可能觉得这只是一个简单的差别，但其实不是。因为就是这一点使得容器可以成为一种 标准化的应用发布方式。想想看，没有人会用虚拟机去发布一个应用吧。而且，这个标准化非常重要。



我们先看看 Docker 官方对其的定义：

Docker 是为开发人员和系统管理员构建，分发和运行分布式应用程序的开放平台。 Docker 包括 Docker Engine（便携式，轻量级运行时和打包工具）以及 Docker Hub（用于共享应用程序和自动化工作流的云服务），Docker 使应用程序能够以组件快速组装，并消除开发，QA 和生产环境之间的不同。 因此，IT 可以更快地分发程序，并在笔记本电脑，数据中心虚拟机和任何云上运行相同的应用程序。