4.Docker是什么

Docker相关概念

构建、分发和运行Docker镜像

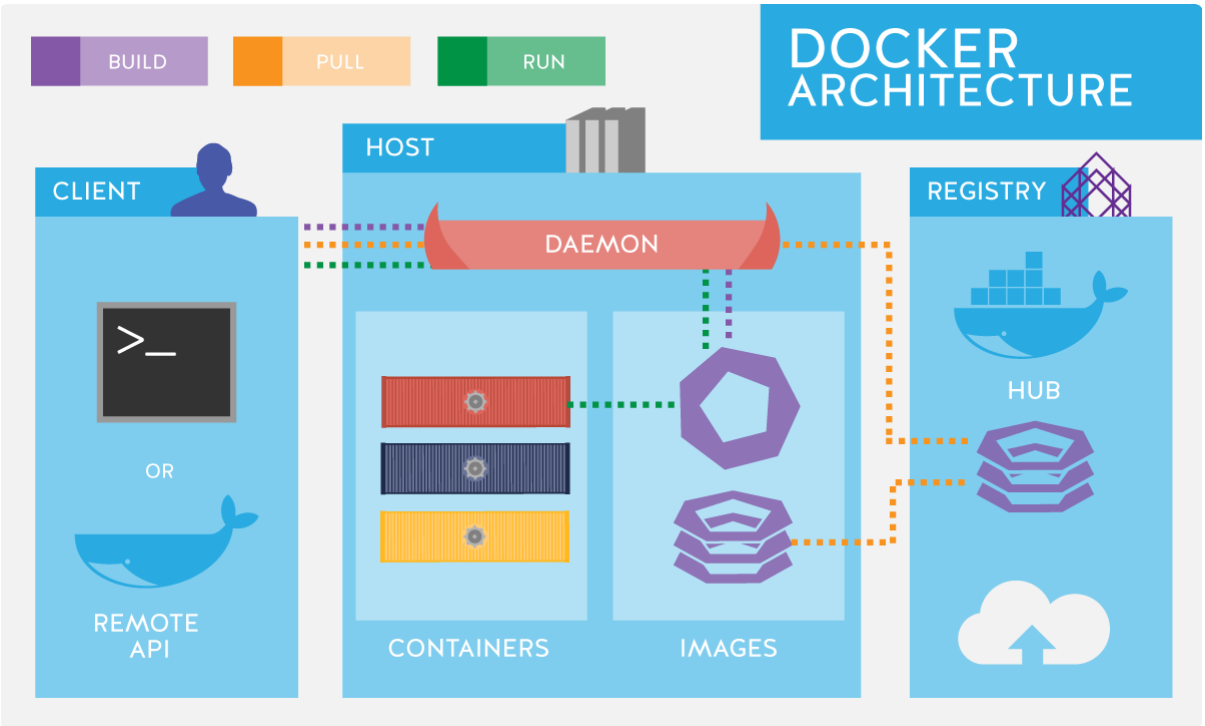
虚拟机 VS Docker容器

什么是镜像层

“Docker”可以指代的对象可以有如下几个：

- Docker 容器技术：可以创建和使用Linux容器
- Docker 社区：开源Docker社区致力于改进这些技术，使所有用户受益（https://forums.docker.com/）
- Docker 公司：以Docker社区的工作为基础，使其更加安全，并将这些进步回馈给更大的社区。为企业客户提供经过增强的技术。

本文要介绍的是Docker容器技术。



Docker是一种容器技术，正是因为它，容器技术才变得广为人知。Docker是第一个使容器能更加容易地在不同机器之间移植的容器系统。它简化了应用程序及其相关依赖的打包流程，甚至整个操作系统的

文件。Docker将它们都打包进一个简单的、可移植的包，这样，一个应用程序就可以以docker包（即镜像）的形式分发到任何安装了docker软件的机器上，然后快速地启动。

当我们使用docker运行被打包的应用程序时，它能看到其附带的文件系统内容。不管我们是在开发环境还是生产环境运行这个应用程序，docker看到的文件都是一致的，即使生产服务器是一个完全不同体系的Linux操作系统。应用程序接触不到所在服务器上的任何东西，因此即使生产服务器与开发环境所在服务器安装的库完全不一样也无关紧要。

举个例子，如果我们使用了整个红帽子Linux系统（RHEL）的文件对应用程序打包，然后让其运行在装有Fedora或者Debian等其它Linux发行版系统的服务器，应用程序都会认为它运行在RHEL系统中。只是内核有可能不同。
与通过在VM中安装一个操作系统得到一个镜像，再将应用程序打包地镜像里，最后再将整个镜像分发地主机并运行起来类似，Docker也能够达到同样的效果，但是不是使用VM来隔离应用程序，而是使用Linux容器技术来实现与VM几乎同一级别的隔离机制。容器不会使用大的单体VM镜像，而是使用通常来说更小的容器镜像。

使用Docker，我们可以将容器视为非常轻量级的、模块化的虚拟机。这些容器还提供了灵活性——可以在不同环境中创建、部署、复制和移动它们，这有助于为云平台优化应用程序。

基于Docker的容器镜像与VM镜像最大的一个不同之处在于容器镜像是由多层组成的，这些层可以在多个镜像之间**共享和复用**。这就意味着，如果一个镜像和另外某个或某些镜像具有相同的层，那么当下载这个镜像的时候，这些相同的层就会被提前下载下来，后面的镜像运行的时候就无需再重复下载这些公共层了，只需要下载其他层即可。

Docker相关概念

Docker 是一个打包、分发和运行应用程序的平台。它是我们能够将应用程序及其所依赖的整个环境合在一起打包。这个依赖的环境可以是一些应用程序需要的库，甚至也可以是一个安装好的操作系统通常情况下所有可用的文件。通过使用Docker，我们可用将这个包上传到一个**中央仓库**，然后这个包就可以被分发到任何安装了Docker的机器上去执行。
这里就要提到三个概念：

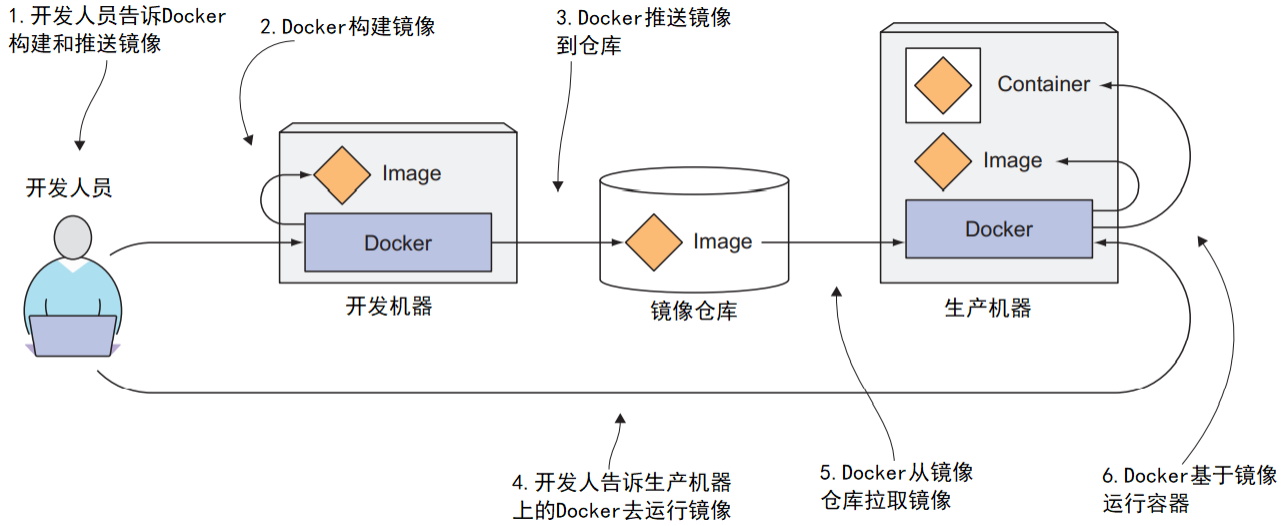
- **镜像（Images）**：Docker容器镜像里包含了打包的应用程序和其所依赖的环境。这个镜像包含了应用程序可用的文件系统和其他元素据，比如镜像运行时需要执行的可执行文件的路径。
- **镜像仓库（Registry）**：Docker镜像仓库存储Docker镜像，有利于在不同的人群和电脑之间共享镜像。当构建好镜像后，我们既可以在构建镜像的电脑上运行它，也可以将其**推送（push）**到镜像仓库，然后在另外一台电脑上**拉取（pull）**它并运行。这一点其实与通过git推送和拉取代码类似。Git也有自己的仓库，只是这个仓库存储的是代码文件而已。镜像仓库也可以做权限控制，比如某些

镜像仓库是公开的，任何人都可以从这个仓库拉取镜像；有些镜像是私有的，只有某一部分人或机器可访问。

- **容器（Container）**：Docker容器是一个常规的Linux容器，它基于Docker镜像创建。一个运行中的容器就是一个运行在装有Docker的主机上的进程，但这个进程是完全与主机以及运行在主机上的所有进程隔离的。这个进程也是资源受限的，意味着它只能访问和使用分配给它的一定量的资源（CPU、内存等等）。

构建、分发和运行Docker镜像

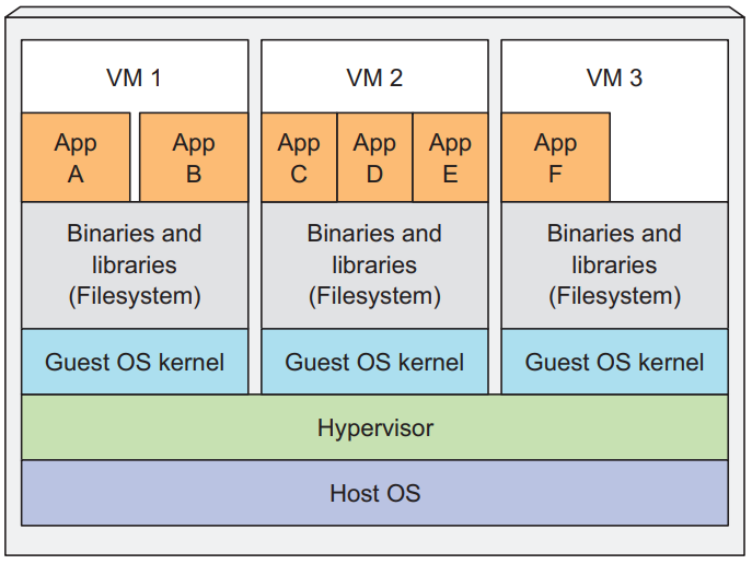
开发人员首先构建好镜像，然后将其推送到仓库。到这一步，镜像就能被任何能访问镜像仓库的人获取到。可以拉取这个镜像到任何其他运行了Docker的机器上，然后运行该镜像。Docker基于镜像创建一个隔离的容器，然后执行二进制可执行文件，该文件被指定为镜像的一部分。从下图可以看出镜像、仓库和容器三者之间的关系：



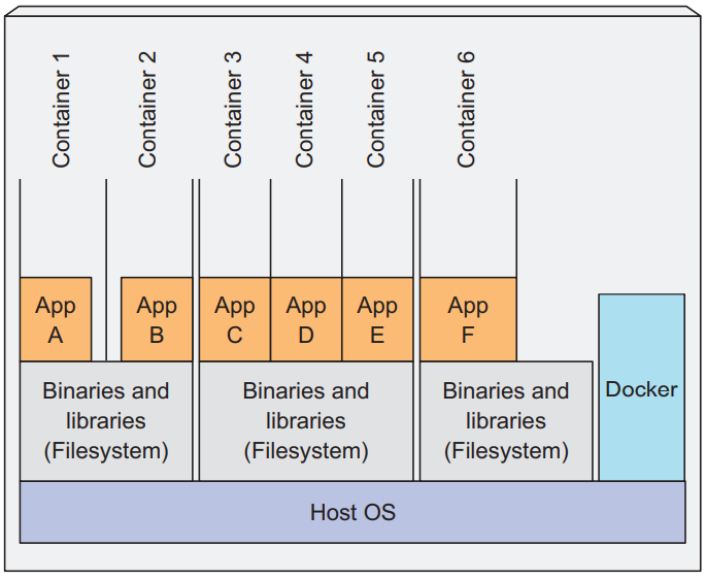
虚拟机 VS Docker容器

如下图所示，相同的6个应用A、B、C、D、E、F分别运行在虚拟机上（3个VM）和容器中（6个Docker 容器）：

运行多个VM的主机

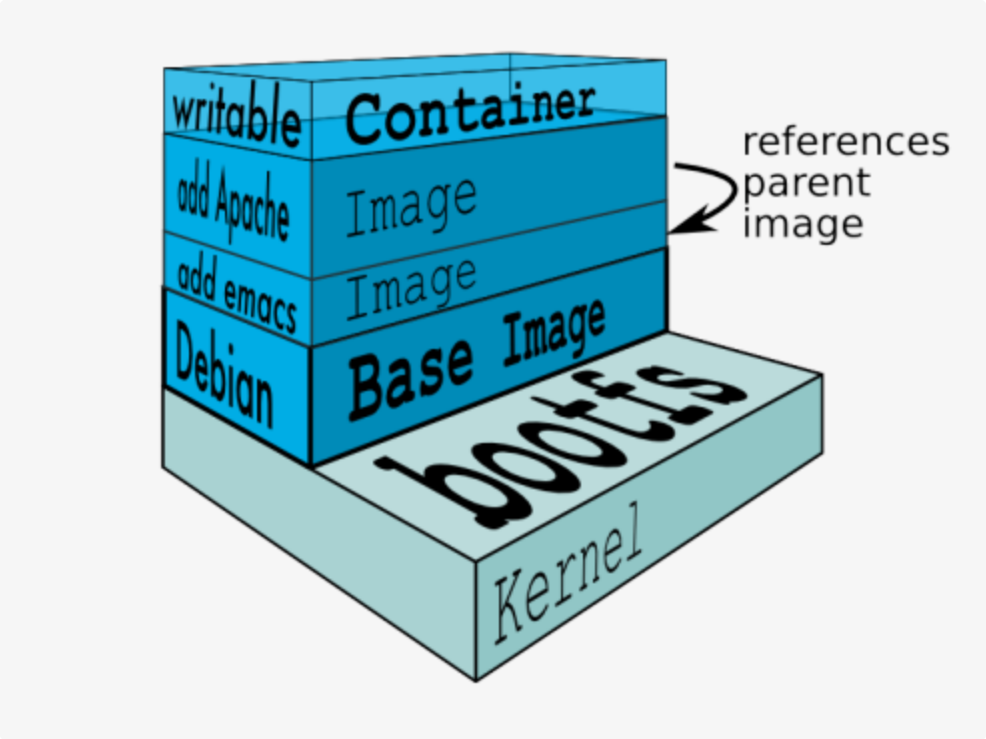


运行多个Docker容器的主机



我们注意到，不管是在一个VM中运行还是在两个分离的容器中运行，A和B能够访问相同的二进制文件和库。在VM中这是很显然的，因为两个应用程序看到都是同一个VM中的相同文件系统。但我们知道，每个容器有它主机的隔离的文件系统，那么应用程序A和B又是如何共享文件的呢?

什么是镜像层



Dockers镜像由多层构成。不同的镜像可以包含完全相同的层，因为每个Docker镜像都是在另一个镜像的基础上构建的，而且两个不同的镜像都可以使用相同的父镜像作为基础镜像。这就提升了镜像在网络上的分发速度，因为当传输某个镜像时，如果相同的层已被之前的镜像传输， 那么这些层就不需要再被传输了。

镜像层（Layer）不只使分发效率变得更高，而且也有助于减少镜像的存储空间。每一层仅被存储一次。因此，由基于相同基础层的两个镜像创建的两个容器都可以读取相同的文件，但是如果其中一个容器写入某些文件，另一个是无法看见文件变更的。因此，即使它们共享文件，但仍然是彼此隔离的。这是因为容器镜像层是只读的。

当运行一个容器的时候，一个新的可写层在镜像的顶层上被创建。当容器中的进程写入位于底层的一个文件时，此文件的一个拷贝在最顶层被创建，进程写入的是此拷贝。