

Docker

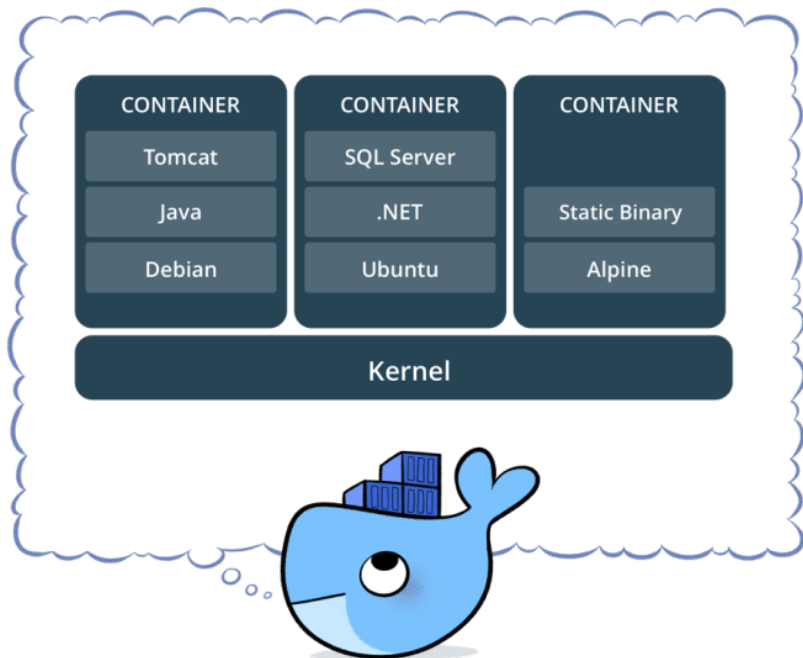
原作者github: <https://github.com/CyC2018/CS-Notes>

PDF制作github: <https://github.com/sjsdfg/CS-Notes-PDF>

一、解决的问题

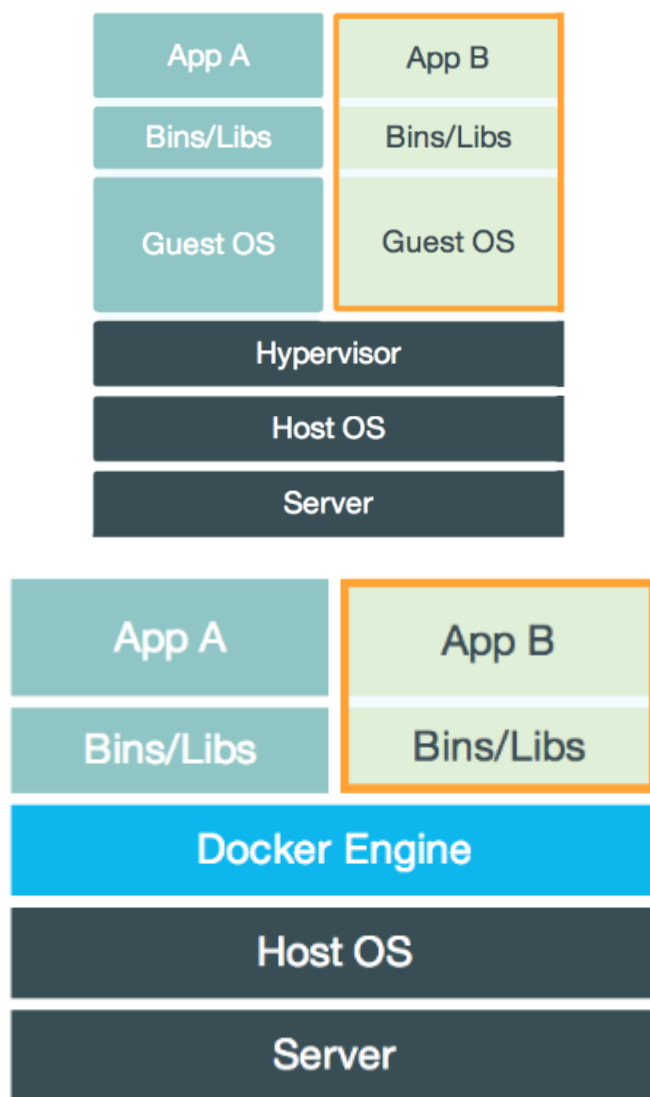
由于不同的机器有不同的操作系统，以及不同的库和组件，在将一个应用部署到多台机器上需要进行大量的环境配置操作。

Docker 主要解决环境配置问题，它是一种虚拟化技术，对进程进行隔离，被隔离的进程独立于宿主操作系统和其它隔离的进程。使用 Docker 可以不修改应用程序代码，不需要开发人员学习特定环境下的技术，就能够将现有的应用程序部署在其他机器中。



二、与虚拟机的比较

虚拟机也是一种虚拟化技术，它与 Docker 最大的区别在于它是通过模拟硬件，并在硬件上安装操作系统来实现。



启动速度

启动虚拟机需要启动虚拟机的操作系统，再启动应用，这个过程非常慢；

而启动 Docker 相当于启动宿主操作系统上的一个进程。

占用资源

虚拟机是一个完整的操作系统，需要占用大量的磁盘、内存和 CPU，一台机器只能开启几十个的虚拟机。

而 Docker 只是一个进程，只需要将应用以及相关的组件打包，在运行时占用很少的资源，一台机器可以开启成千上万个 Docker。

三、优势

除了启动速度快以及占用资源少之外，Docker 具有以下优势：

更容易迁移

提供一致性的运行环境，可以在不同的机器上进行迁移，而不用担心环境变化导致无法运行。

更容易维护

使用分层技术和镜像，使得应用可以更容易复用重复部分。复用程度越高，维护工作也越容易。

更容易扩展

可以使用基础镜像进一步扩展得到新的镜像，并且官方和开源社区提供了大量的镜像，通过扩展这些镜像可以非常容易得到我们想要的镜像。

四、使用场景

持续集成

持续集成指的是频繁地将代码集成到主干上，这样能够更快地发现错误。

Docker 具有轻量级以及隔离性的特点，在将代码集成到一个 Docker 中不会对其它 Docker 产生影响。

提供可伸缩的云服务

根据应用的负载情况，可以很容易地增加或者减少 Docker。

搭建微服务架构

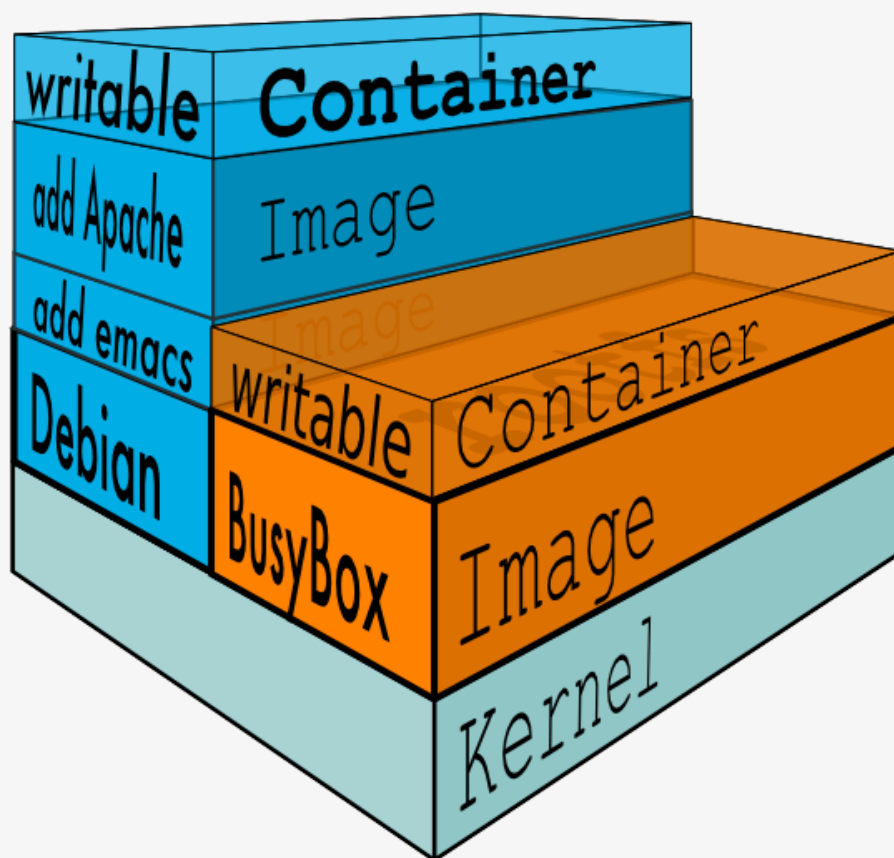
Docker 轻量级的特点使得它很适合用于部署、维护、组合微服务。

五、镜像与容器

镜像是一种静态的结构，可以看成面向对象里面的类，而容器是镜像的一个实例。

镜像包含着容器运行时所需要的代码以及其它组件，它是一种分层结构，每一层都是只读的（read-only layers）。构建镜像时，会一层一层构建，前一层是后一层的基础。镜像的这种分层存储结构很适合镜像的复用以及定制。

构建容器时，通过在镜像的基础上添加一个可写层（writable layer），用来保存着容器运行过程中的修改。



参考资料

- [DOCKER 101: INTRODUCTION TO DOCKER WEBINAR RECAP](#)
- [Docker 入门教程](#)
- [Docker container vs Virtual machine](#)
- [How to Create Docker Container using Dockerfile](#)
- [理解 Docker \(2\) : Docker 镜像](#)
- [为什么要使用 Docker?](#)
- [What is Docker](#)
- [持续集成是什么?](#)