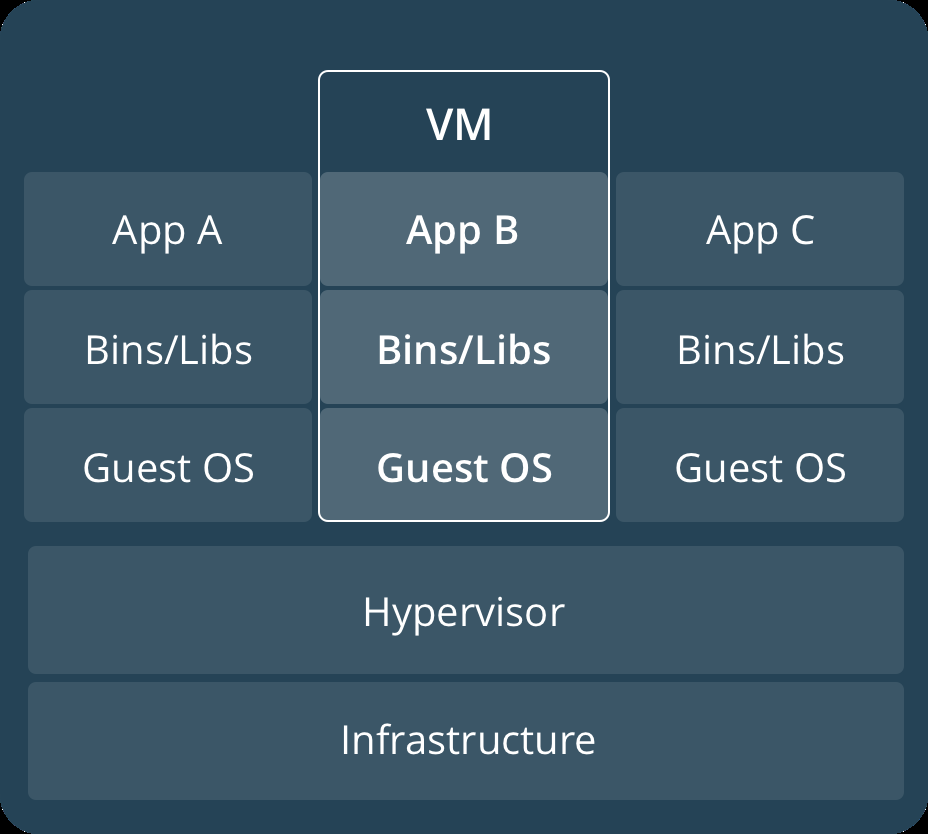
Docker 学习笔记

## 定位和设置

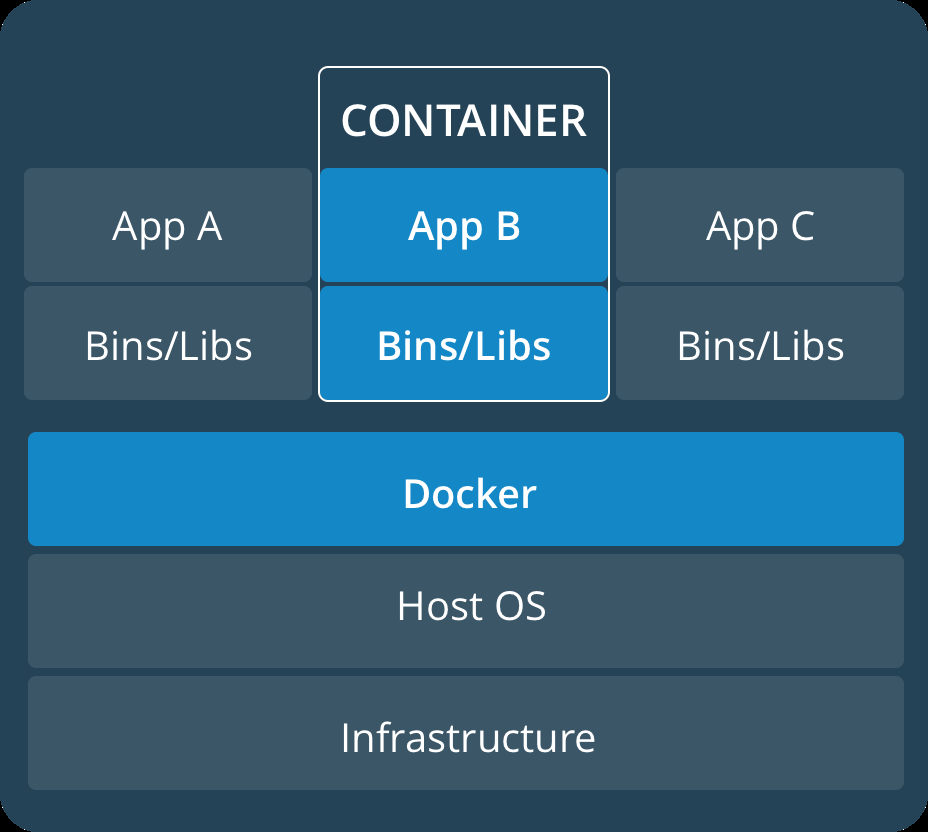
容器 vs. 虚拟机：

虚拟机图：



虚拟机运行客户操作系统 - 注意每个框中的操作系统层。这是资源密集型的，产生的磁盘映像和应用程序状态是操作系统设置，系统安装的依赖关系，操作系统安全补丁和其他易于丢失，难以复制的纠缠。

容器图：



容器可以共享一个内核，并且唯一需要在容器映像中的信息是可执行文件及其包依赖关系，它们永远不需要安装在主机系统上。这些进程像本地进程一样运行，您可以通过运行命令来单独管理它们，就像在Linux上docker ps运行一样ps，以查看活动进程。最后，因为它们包含了所有的依赖关系，所以没有配置纠缠。一个集装箱化的应用程序“随处运行”。

## 容器（Container）

docker build -t friendlyname . # Create image using this directory's Dockerfile

docker run -p 4000:80 friendlyname # Run "friendlyname" mapping port 4000 to 80

docker run -d -p 4000:80 friendlyname # Same thing, but in detached mode

docker container ls # List all running containers

docker container ls -a # List all containers, even those not running

docker container stop <hash> # Gracefully stop the specified container

docker container kill <hash> # Force shutdown of the specified container

docker container rm <hash> # Remove specified container from this machine

docker container rm $(docker container ls -a -q) # Remove all containers

docker image ls -a # List all images on this machine

docker image rm <image id> # Remove specified image from this machine

docker image rm $(docker image ls -a -q) # Remove all images from this machine

docker login # Log in this CLI session using your Docker credentials

docker tag <image> username/repository:tag # Tag <image> for upload to registry

docker push username/repository:tag # Upload tagged image to registry

docker run username/repository:tag # Run image from a registry