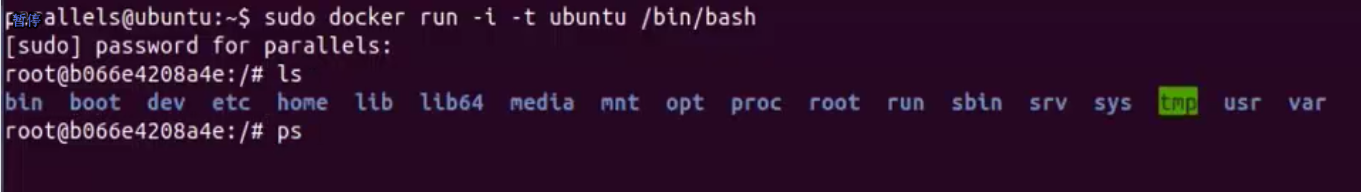


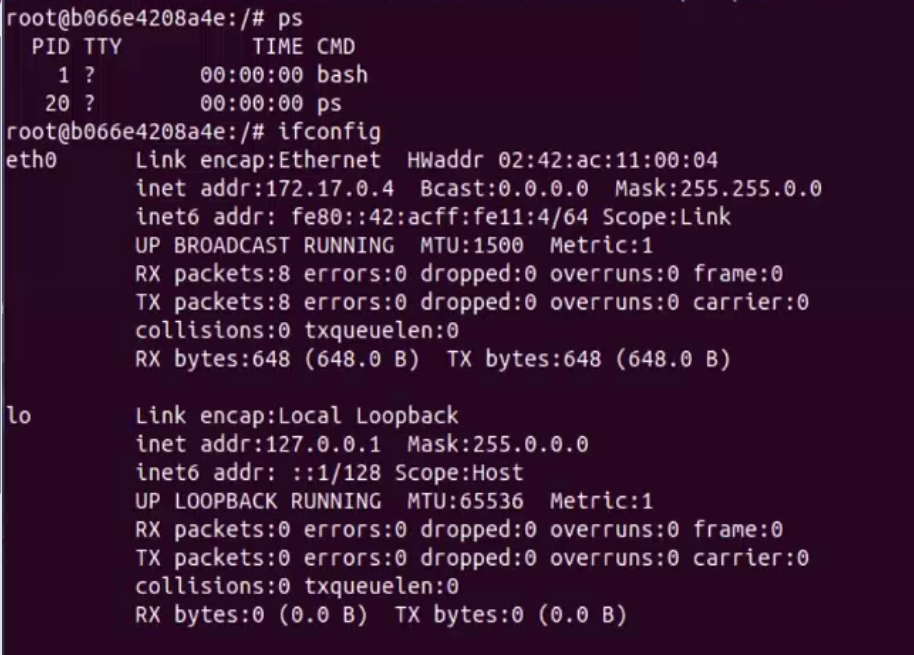


启动一个docker的容器

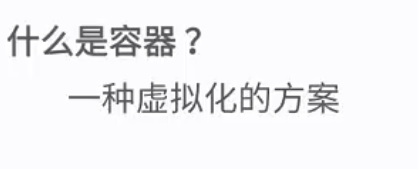
指定运行容器的操作系统是ubuntu



这个是使用root用户来登录的docker --- 后面的一串数字 是docker image的名字



在docker中使用Linux命令 可以看到 有文件系统 进程 网络设备 和 虚拟机非常相似 实际上 docker的容器技术 就是虚拟化的一种



传统的虚拟机 是讲一台或者多台独立的机器 虚拟运行在硬件之上

但是 容器 是直接运行在OS内核上的用户空间

容器虚拟化 也叫操作系统虚拟化

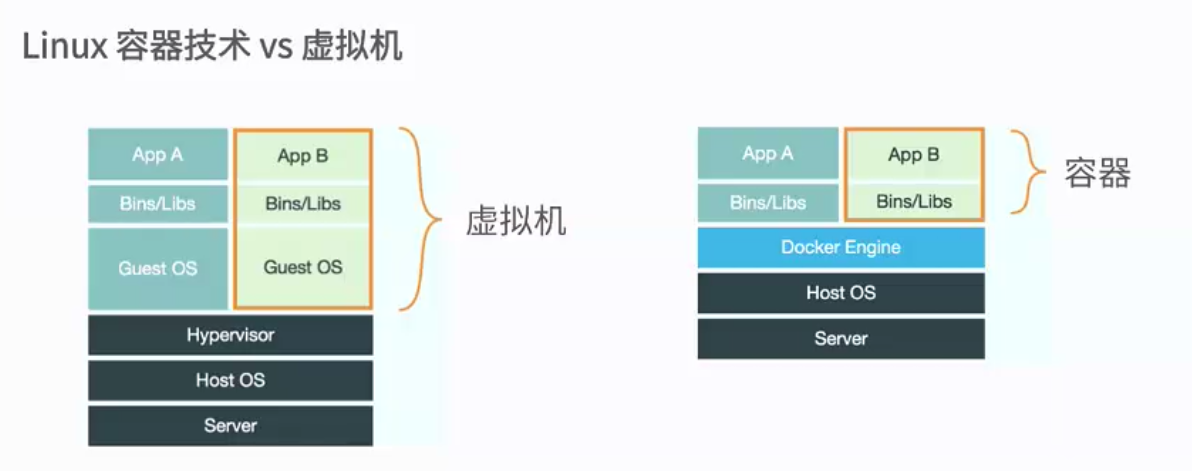
依赖于OS特性 所以 

**Docker依赖的os的特性 就是Namespace和Cgroups ----所以 docker上面不能运行在windows系统上**

这也是容器技术相对于虚拟机上上的劣势



容器技术的优点



一个显而易见的特点 就是磁盘占用空间更少

使用虚拟机部署应用 需要包含应用和依赖的库 还需要包含完整的操作系统

原本几十兆 现在动辄几十个G

但是 容器 不需要包含完成的操作系统上面 所以 磁盘占用率大大降低

【深绿色的部分 是第一个应用 ---- 有自己的app 依赖的bins、libs 和 完整的操作系统 ---- 墨绿色的 是宿主机的操作系统 服务器 ======

但是docker上面部署的app 就仅仅需要自身的app和依赖就可以了 运行在docker引擎上面 不需要完整的操作系统】

容器 仅仅包含app和依赖的库 这样 占用的资源大大减少

另外虚拟机模拟硬件的行为 对CPU和内存损耗很大

所以 同样配置的服务器 使用容器 比 虚拟机 提供更多的能力 更多的用户

既然容器的优势这么大 为什么容器技术在之前不流行？

**-------- 容器本身就比较复杂 依赖linux的内核很多特性 很难自动化 管理**

Docker的出现改变了这一现状

Docker能够把开发应用程序自动部署到容器的开源引擎

使用go编写



13年叫dotCloud公司

特别之处：Docker在虚拟化的容器执行环境中 增加了一个**应用程序部署引擎** --- 目标就是提供轻量快速的环境 能运行开发者的程序 并且能够把开发者的程序部署测测试环境 然后部署到生产环境

Docker希望实现的目标/特点

 容易上手 大多数docker容器不到1s就可以运行起来 ---开始运行的docker容器 启动非常快 由于去除了管理程序的开销 docker容器性能高

同一台宿主机 可以运行更多的docker容器 是用户尽可能利用系统资源

 开发人员只需要关心docker容器里面的应用程序逻辑 而运维人员只需要关心如何管理docker容器

**Docker设计的目的** 就是加强开发人员写代码的**开发环境 与应用程序要部署生产环境**的一致性

 docker出现就是为了缩短从开发 到测试 到部署上线的生命周期

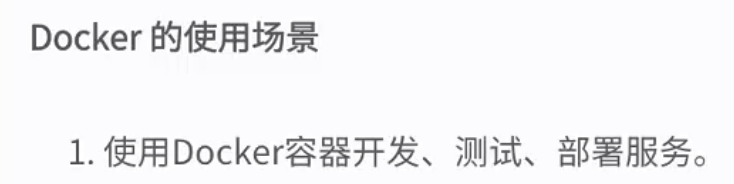
让应用程序具备可移植性 ---在容器中开发 并且以容器的形式进行交付 和 分发

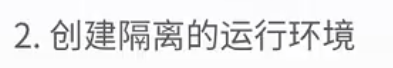
这样 开发 测试 生产都使用先通的环境 就避免了额外的调试 和 部署的开销

有效缩短产品的上线周期

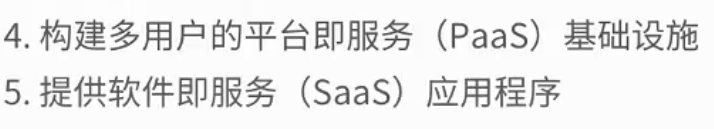
 docker推荐单个容器仅仅部署一个进程 形成一个分布式的应用程序模型 --- -这种情况 应用程序或者服务就表现为一系列内部互联的容器   
 使得分布式 部署应用程序 调试 和扩展 变得非常容易 ---- 这就要求在开发遵循高内聚 低耦合 单一任务 --- 能够避免在统一服务器中 部署不同服务之间的相互影响 ---- 也容易在服务发生问题的时候 进行问题的定位

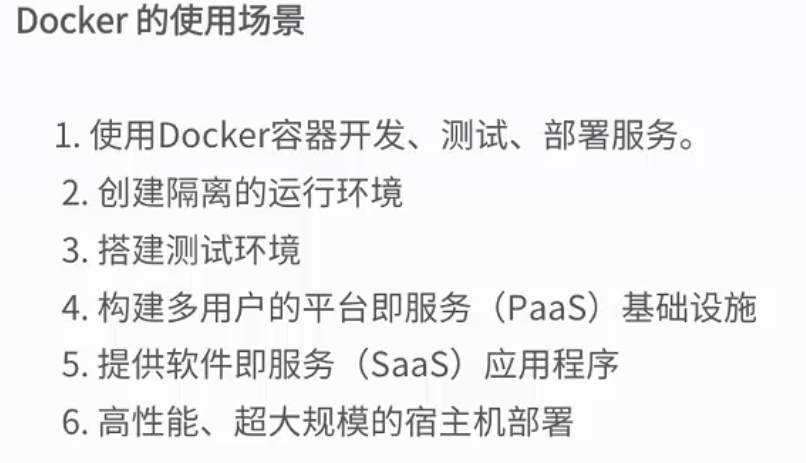
====================

 docker轻量 所以 容器可以在开发环境中创建 然后提交到测试 最终进入生产环境

 很多企业应用中 同一服务的不同版本可能服务于不同的用户 使用docker非常容易创建不同的生产环境 来运行不同的服务

 由于docker的轻量化 所以 开发者本身非常容易在本地搭建测试环境 –用来测试应用程序在不同系统下的兼容性 甚至搭建集群部署测试



 很多公有云服务都支持docker

很多开发者使用openstack和docker开发SAAS和Paas

回顾：

 依托于linux内核功能的虚拟化技术

 docker是能够把应用程序自动部署到容器的开源引擎

-----------------

Docker的基本组成

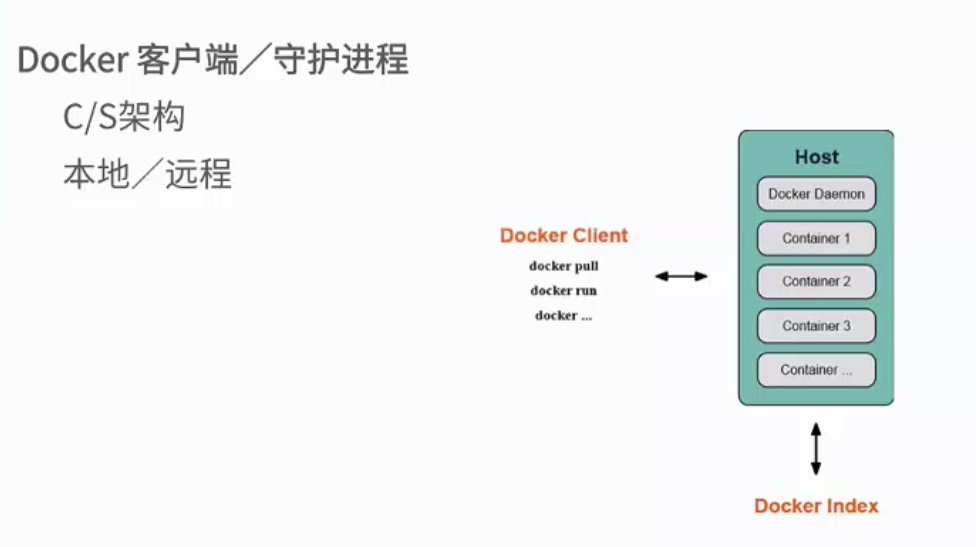
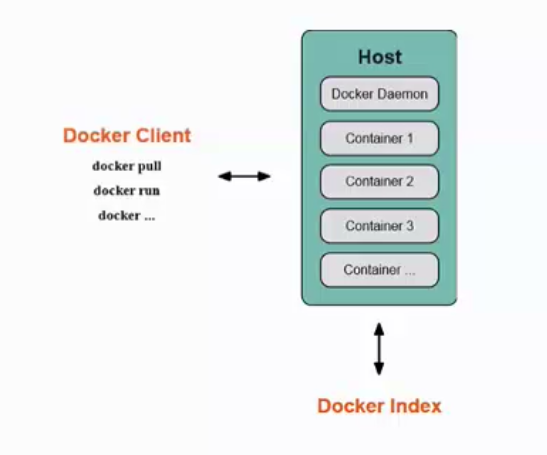




CS架构的程序

守护进程就是S

客户端可以本地 也可以远程



Docker客户端可以发送各种docker的命令给主机上的Docker守护进程 Docker Daemon

然后 守护进程可以返回信息给Docker Client ---- 这样可以通过Docker的客户端可以查看Docker命令执行的结果

Docker镜像

 镜像是容器启动的基石 镜像好比容器的源代码 保存了启动容器的各种条件

Docker镜像的深入理解 是层级的只读文件系统 最底层是引导文件系统 就是BootFs 很像linux的引导文件系统

Docker用户几乎永远不会和bootfs进行交互

----- 实际上container启动后 这个bootfs会被移动到内存中 然后这个引导文件系统会被卸载

Docker镜像的第二层是root文件系统 位于bootfs之上 -- -rootfs可以是一种或者多种的操作系统 比如ubuntu或者centos

传统linux引导过程中 rootfs最先以只读的方式加载 ---只有引导完成并且验证完成之后 rootfs才会切换成读写模式 --- docker中rootfs 永远只能只读的 并且 在docker中 rootfs利用联合加载技术 在bootfs之上 加载更多的只读文件系统

联合加载指的是 一次加载多个文件系统 但是外部看来 就是一个文件系统



图中的add Apache和 add emacs就是联合加载的多个只读文件系统

联合加载会将各层文件系统合并到一起 --- 最终的文件系统最终会包含所有的底层文件和目录 ---docker将这样的文件系统称为镜像

一个镜像可以放到另一个镜像的顶部 ---- 下面的镜像称为父镜像 ---最底部的镜像称为基础镜像 这里面就是bootfs

----- Docker Container

Container通过镜像来启动 ----- Docker的容器是Docker的执行单元 （线程是程序的执行路径） ---- 一个容器中 可以运行一个或者多个用户的进程 ------

如果说镜像是docker生命周期的构建和打包阶段 那么容器是启动和执行阶段

当一个容器启动的时候 会在最顶层加载一个3:47