1. Docker的前世今生

什么是Docker？Docker的意思是码头工人，为啥会取这么一个奇怪的名字，当然来自于程序员的恶趣味。它隐喻的是“集装箱”，它意味着“赶快来使用Docker吧，用了Docker，世界上就有了集装箱，码头工人就可以下岗了”。同时Docker公司的口号也是“Build，Ship，and Run Any App，Anywhere”。既然，它隐喻的是集装箱，那么它也就是有集装箱的特性，大家应该基本都理解集装箱是怎么改变世界的吧？在集装箱之前，货物运输没有统一的标准方式进行搬运，于是铁路、公路、海洋等各种运输之间，需要大量的人力作为货物中转，效率极低，而且成本很高。集装箱出现之后，世界上绝大多数的货物运输都可以放到这个神奇的箱子里，然后在公路、铁路、海洋等所有运输场景下，这个箱子都可以不用变化形态直接可以承运，而且中间的中转工作，都可以通过大型机械搞定，效率大大提升。从此全球化开始，商业的潜力被进一步挖掘...... 可是这个箱子为什么这么神奇呢？答案其实也就在上面的描述中，无非就是两个字：标准。将所有的货物标准的打包，将搬运的过程标准化。因为规范了集装箱的大小和尺寸的规格标准，于是相应的船舶、卡车、列车才能按照规格制造出来使联运成为可能，所有的运输中转的自动化工具才能被设计建造出来并且高效的使用，才可以极大的提高效率，提升自动化水平，以至于码头工人才会失业。集装箱本身是一个产品，而这个产品无非就是“标准化”的这个概念穿上了马甲，马甲可以有红的、绿的、蓝的、花的，但是大小规格必须都一样。按照这个思路，docker 其实跟集装箱一样，或者说它想跟集装箱一样，成为穿着马甲的“标准化”。这样开发工程师就可以把它们开发出来的 bug 们放到“集装箱”里，然后运维人员就可以使用标准化的操作工具去运维这些的 bug 们。于是实现了“海陆联运”，就好像运维工程师根本不需要了解其运维的软件架构而开发工程师也并不需要了解其软件运行的操作系统一样……

为什么需要这个标准化呢？描述一个现实的场景。你是不是正在运维着一个使用 php、java、C#甚至 C/C++等用各种语言编写的应用都在运行的环境里？这个环境是不是因为某种历史原因，使你的操作系统运行着各个版本的内核甚至还有 windows？即使是同样语言编写的业务也运行着不同版本的库？你的整个系统环境是不是甚至找不出来两台硬件、操作系统、库版本以及语言版本完全一样的环境？于是你每次遇到问题都要去排查到底那个坑到底在那里？从网络、内核到应用逻辑。你每次遇到产品升级都要在各种环境上做稳定性测试，发现不同的环境代码 crash 的原因都不尽相同。你就像一个老中医一样去经历各种疑难杂症，如果遇到问题能找到原因甚至都是幸运的，绝大多数情况是解决了但不知道原因和没解决自动好了也不知道原因。于是Docker的出现就是为了解决这类的问题，将你的应用，环境配置打包成一个独立镜像。

1. Docker的基本使用

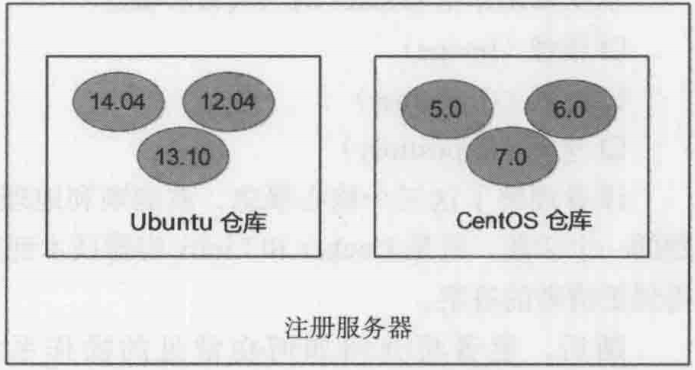
2.1 先认识一下Docker是的构成

Docker Client : Docker提供给用户的客户端。Docker Client提供给用户一个终端，用户输入Docker提供的命令来管理本地或者远程的服务器。

Docker Daemon : Docker服务的守护进程。每台服务器（物理机或虚机）上只要安装了Docker的环境，基本上就跑了一个后台程序Docker Daemon，Docker Daemon会接收Docker Client发过来的指令,并对服务器的进行具体操作。

Docker Images : 俗称Docker的镜像，理解为一个面向Docker引擎的只读模板。可以认为这个就像我们要给电脑装系统用的系统CD盘，里面有操作系统的程序，并且还有一些CD盘在系统的基础上安装了必要的软件，做成的一张 “只读” 的CD。

Docker Registry : 这个可认为是Docker Images的仓库的仓库，就像git的仓库一样，用来管理Docker镜像的，提供了Docker镜像的上传、下载和浏览等功能，并且提供安全的账号管理可以管理只有自己可见的私人image。就像git的仓库一样，docker也提供了官方的Registry，叫做Dock Hub(<http://hub.Docker.com>)。注：Registry和Repository不太一样



Docker Container : 俗称Docker的容器，这个是最关键的东西了。Docker Container是真正跑项目程序、消耗机器资源、提供服务的地方，Docker Container通过Docker Images启动，在Docker Images的基础上运行你需要的代码。 你可以认为Docker Container提供了系统硬件环境，然后使用了Docker Images这些制作好的系统盘，再加上你的项目代码，跑起来就可以提供服务了。可能会觉得像一个VM利用保存的备份或者快照跑起来环境一样，其实是挺像的，但是实际上是有本质的区别，一开始感性理解可以理解成为一个快照。

3. Docker的技术细节