### Dicker

问题引入：

在微服务中，有一个让人头疼的问题，就是部署问题，因为微服务非常的多，部署起来很麻烦，docker正好可以解决这个问题：

### 什么是Docker:

一个大型的项目，组件往往很多，运行环境也会很复杂，部署时会碰到一些问题，特别是微服务项目：各种各样的微服务，而且这些微服务还会依赖于各种各样的应用，比如说：前端部分，肯定会依赖node.js，而我们的服务端，还需要数据库（mysql）、缓存系统（redis）、异步通信（MQ）、等等，所有的应用都需要部署到服务器上，而大多数的服务器都会采用linux操作系统，这些应用在安装到Linux操作系统之前，往往要做很多的准备工作，因为这些应用都会有自己所需要的依赖和函数库，如：gcc、epel、libstdc++、glibc、scoat、binutils等等。

而每一个不同的应用，他们所需要的依赖和函数库可能有不同；或者一样，但是版本不同，这个时候，如此复杂依赖关系，很容易产生兼容性问题，而且就算你花了很大功夫，终于解决了以上这些问题，你会发现，这仅仅是个开始，搞定了开发环境，还有测试环境、生产环境、预发布环境。。。在等着你，最可怕的是，这些环境的linux系统还有可能不同，比如说，一个环境是centos系统，还有一个环境装的是Ubuntu系统。。。你在这个环境下配好了各种关系，你换到另外一个环境的操作系统几乎可以肯定是不能正常运行的，这就是我们目前在开发、部署时遇到的问题：效率极低！

**以上问题，Docker是如何解决的？**

1：即然每一个应用，都有自己所需要的依赖和函数库，那为什么不把这些依赖、函数库和应用一起打包呢，比如，nodejs需要的libs（函数库、deps（依赖）、配置，我把你打包起来，形成一个整体；而后把这个整体放到一个隔离的容器中去运行，避免相互干扰。每一个应用可以都这么做，应用需要什么依赖、需要什么函数库，都提前准备好，然后和应用一起，打成一个包，将来要部署时，直接运行这个包，此时就不需要关心依赖的问题了！

问：把上面应用扔到一台机器上，应用之间的依赖不会有干扰吗？

答：不会！因为docker会把这些打包好的程序用一个隔离的容器去运行，会应用到沙箱的技术，他们相互之间是不可见的！

2：当然，1只是解决了我们说的依赖混乱的问题，即有了docker，依赖就不用去管了，应用程序随时都可以运行，但是，是仅限于同一个操作系统，因为我们在打包一个应用时，是基于某种操作系统去打包的，如这个应用是ubuntu版本的，那么这个应用的依赖和函数库肯定也是Ubuntu版本的，把这个打包好的程序扔到centos上，是不能运行的。即：

不同环境的操作系统不同，Docker如何解决？我们要先了解下操作系统结构：



其实所有的Linux内核的操作系统都可以分为两层：一层是共享的一样的linux内核，二是上层的不同的系统应用。内核负责与计算机硬件进行交互，比如调用cpu、调用内存、读写文件、开辟进程等等，这些都是内核做的事，而内核会把这些事情变成一个一个指令，你调用这些指令就可以操作计算机硬件了，但是这些指令是比较简陋的，如果要基于这些指令去开发应用，那就太麻烦了，所以就有了系统应用，比如Ubuntu，Ubuntu会将内核指令进行封装，形成函数，许许多多的函数就形成了函数库，程序员基于这些函数库进行开发，实现功能：程序调用函数库，函数库调用内核指令，指令去调用计算机硬件，从而实现应用的执行，以上就是linux系统的结构以及应用的执行原理！

那么问题来了，一个Ubuntu的应用为什么不能在Centos上运行呢？

Ubuntu和Centos的内核是一样的，系统应用不同，也就是说函数库不同，存在Ubuntu上有的函数库，Centos上不一定有，或者说函数名字不一样，此时，如果把Ubuntu的应用（如Mysql）迁移到Centos上，尝试去执行，因为函数库不一样，应用去调用一个函数库时，会发现函数库在Centos不存在，肯定会程序报错。这就是应用不能跨系统运行的原因！

**针对这样的问题，Docker干了什么事？**

即然每个应用都依赖于系统函数库，为什么不把应用需要的系统函数库一起打包呢？这样一来，就不愁找不到函数库了，所以Docker的解决方案就是：将用户程序与所需要调用的系统函数库（如Ubuntu）一起打包！