LXC vs Docker

LXC vs Docker

先说和虚拟化技术的区别：

难道虚拟技术就做不到吗?

不不不，虚拟技术也可以做到，但是会有一定程度的性能损失，灵活度也会下降。容器技术不是模仿硬件层次，而是 在Linux内核里使用cgroup和namespaces来打造轻便的、将近裸机速度的虚拟技术操作系统环境。因为不是虚拟化存储，所以容器技术不会管 底层存储或者文件系统，而是你放哪里，它操作哪里。

LXC vs Docker

这从根本上改变了我们如何虚拟化工作负载和应用程序，因为容器速度比硬件虚拟化技术更快，更加便捷，弹性扩容的更加高效，只是它的工作负载要求操作系统，而不是Linux或特定的Linux内核版本。

LXC vs Docker

那VMWare就这样玩完了?

没那么快！虚拟技术相对成熟，又有广泛的工具，还有生态系统来支持它在不同环境下的配置。至于工作负载，它要求非Linux操作系统，或者只能使用特定的核心虚拟化技术。

LXC vs Docker

Docker并不是LXC的替代品，Docker的底层就是使用了LXC来实现的。LXC将Linux进程沙盒化，使得进程之间相互隔离，并且能够控制各进程的资源分配。

在LXC的基础之上，Docker提供了一系列更强的功能。

LXC vs Docker

可移植性

Docker定义了一种新的格式，将应用和其依赖环境全部打包到一个单一对象中，这个对象可以在任何安装有Docker的机器上共享，在任何机器上 执行这个对象的效果都是一样的。LXC仅仅实现了进程沙盒化，并不能在不同机器上进行移植。Docker将应用的所有配置进行抽象，打包到一个容器中，使 得该容器具有可移植性。

LXC vs Docker

以应用为中心

Docker针对应用的部署做了优化，反映在其API，用户接口，设计原理及文档上面。而LXC仅仅关注容器作为一个轻量级的服务器。

LXC vs Docker

自动化构建

Docker中支持Dockerfile，将应用的所有依赖项，构建工具和包都以源码的形式写在Dockerfile中，然后Docker可以根据Dockerfile构建镜像。该镜像在任何机器上面运行的效果都一样。

LXC vs Docker

版本控制

Docker对容器提供了类Git的版本控制功能，支持版本回滚等功能。Docker也实现了增量上传和下载的功能，节约了上传和下载时的带宽资源。

LXC vs Docker

组件重用

一个镜像可以作为基础镜像来创建更多特定的镜像，镜像之间支持多层重用。

LXC vs Docker

镜像共享

Docker开发了一个Docker Hub，里面包含了各种常用的镜像，非常方便，我们也可以将自己的镜像上传到Docker Hub中。用户也可以在私有环境中搭建自己的Docker仓库，用来满足镜像的内部共享。

LXC vs Docker

工具生态系统

Docker定义了一个API，用于自动化和本地化容器的创建和部署。已经存在大量的集成了Docker的工具集，例如Deis，mesos，docker-ui，jenkins等等。