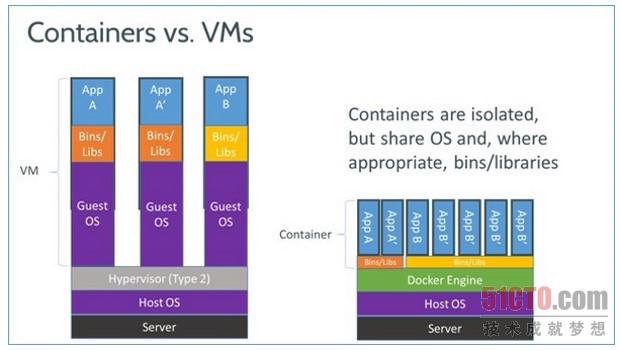
Docker



VM管理程序： KVM, hyper-V, Xen, etc.

对硬件虚拟化，基于虚拟化硬件仿真机制，对系统要求高。

容器：使用共享的操作系统，驻留在单单一个linux实例上。

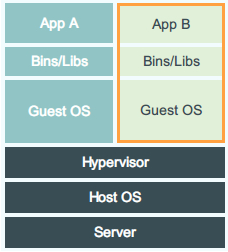
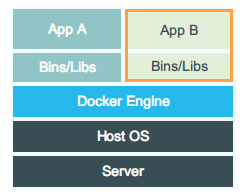
支持者：红帽，Canonical, linux, microsoft。

容器与虚拟机主要区别在于：虚拟机管理程序对整个设备进行抽象处理，而容器只是对操作系统内核进行抽象处理。

Docker容器相对于VM有以下几个优点：

* 启动速度快，容器通常在一秒内可以启动，而VM通常要更久。
* 资源利用率高，一台普通PC可以跑上千个容器。
* 性能开销小，VM 通常需要额外的 CPU 和内存来完成 OS 的功能，这一部分占据了额外的资源。

VM与Docker设计比较：

VM 的 Hypervisor 需要实现对硬件的虚拟化，并且还要搭载自己的操作系统，自然在启动速度和资源利用率以及性能上有比较大的开销。

虚拟机管理程序能做而容器做不了的是：使用不同的操作系统或内核。例如：可以使用微软Azure，同时运行Windows Server 2012的实例和SUSE Linux企业级服务器的实例。至于Docker，所有的容器必须使用同样的操作系统和内核。

容器适用场景：同一应用程序的多个副本。

使用Docker封装、交付和运行任何应用程序，应用程序成为轻型的、可移植的、自给自足的LXC容器，可以在任何地方运行。“容器让你立即享有应用程序可移植性”。

Docker:

建立在LXC基础上，就该程序而言，它有自己的文件系统 、存储系统 、处理器和内存等部件。

Docker容器易于部署到云端，Docker采用了一种特别的方式，以便可以整合到大多数DevOps应用程序当中，包括Puppet, Chef, Vagrant, Ansible，也可独自使用，以管理开发环境。

有了Docker，人们就可以搭建与活动服务器一模一样的本地开发环境，从同一个主机运行多个开发环境（每个开发环境有独特的软件、操作系统和配置），在新的或不同的服务器上测试项目，以及让任何人都可以在设置一模一样的情况下处理同一项目，无论本地主机环境怎样。”

简而言之，Docker能为你做的事情就是：相比其他技术，它能让更多数量的应用程序在同一硬件上运行；它让开发人员易于快速构建可随时运行的容器化应用程序；它大大简化了管理和部署应用程序的任务。

Docker用途：

保证线下的开发环境、测试环境和线上生产环境一致。

容器也和VM一样具有一定的隔离性，各个容器之间的数据和内存空间相互隔离，可以保证一定的安全性。

Docker使用场景：

开发直接在容器里开发，提测的时候把整个容器给测试，测好了把改动改在容器里再上线。通过容器，整个开发、测试和生产环境可以保持高度的一致。