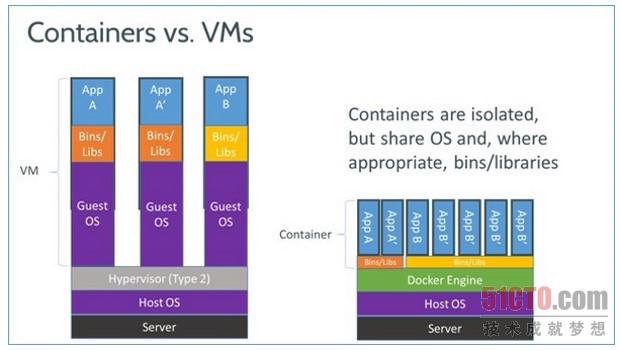
Docker



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。

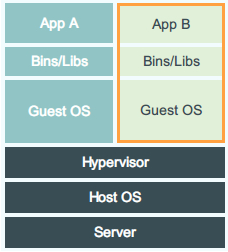
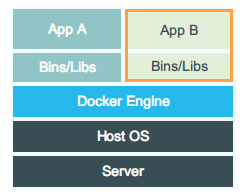
支持者：红帽，Canonical, linux, microsoft。

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。

Docker能做什么？

* 隔离应用依赖
* 创建应用镜像并进行复制
* 创建容易分发的即启即用的应用
* 允许实例简单、快速地扩展
* 测试应用并随后销毁它们

Docker背后的 想法：创建软件程序可移植的轻量容器，让其可以在任何安装了Docker的机器上运行，而不用关心底层操作系统。

Docker两个最重要的概念是：镜像和容器。另外，链接和数据卷也很重要。

**镜像**

Docker的镜像类似虚拟机的快照，但更轻量。

创建Docker镜像的方式：

多数是在一个现有镜像基础上创建新镜像，因为几乎你需要的任何东西都有了公共镜像，包括主流Linux发行版。要创建一个镜像，可以拿一个镜像，对它进行修改来创建它的子镜像。目前的方式有两种

* 在一个文件中指定一个基础镜像及需要完成的修改。（推荐）
* 通过“运行”一个镜像，对其进行修改并提交。

镜像拥有唯一ID（比如：ubuntu:precise）。

**容器**

可以从镜像中创建容器，这等同于从快照中创建虚拟机，不过更轻量，应用是由容器运行的。

容器与虚拟机一样，是隔离的，也拥有现代战争唯一ID和唯一的供人阅读的名字。容器对外公开服务是必要的，因此Docker允许公开容器的特定端口。

容器与虚拟机的最大差异：

它们被设计用来运行单进程，无法很好的模拟一个完整的环境（如果那是你需要的，请看LXC）。如果选择在一个容器中运行多个进程，唯一情况是出于调试目的，运行类似ssh的东西来访问运行中的容器，不过，docker exec命令解决了这个问题。

容器是设计来运行一个应用的，而非一台机器，Docker提供了用于分离应用与数据的工具，使得你可以快捷地更新运行中的代码/系统，而不影响数据。

**数据卷**

数据卷让你可以不受容器生命周期影响进行数据持久化。它们表现为容器内的空间，但实际保存在容器之外，从而允许你在不影响数据的情况下销毁、重建、修改、丢弃容器。Docker允许你定义应用部分和数据部分，并提供工具让你可以将它们分开。卷还可以用来在容器间共享数据。使用Docker的最大思维变化之一就是：容器应该是短暂和一次性的。

**链接**

容器启动时，将被分配现代战争随机的私有IP，其它容器可以使用这个IP地址与其进行通讯。一提供了容器间相互通信的渠道，二共享一个本地网络。限制容器间通讯是可行的。

要开启容器间通讯，Docker允许你在创建一个新容器时引用其它现存容器，在你刚创建的容器里被引用的容器将获得一个（你指定的）别名。我们就说，这两个容器链接在了一起。

因此，如果DB容器已经在运行，可以创建web服务器容器，并在创建时引用这个DB容器，给它一个别名，比如dbapp。在这个新建的web服务器容器里，可以在任何时候使用主机名dbapp与DB容器进行通讯。

更进一步，Docker要求你声明容器在被链接时要开放哪些端口给其他容器，否则将没有端口可用。

镜像必须完全可移植，Docker不允许 例外。

小结：创建镜像、用它们创建容器、在需要时暴露端口和创造卷、通过链接将几个容器连接在一起。

Docker如何完成它需要完成的东西？

cgroups和union文件系统。Docker使用cgroup来提供容器隔离，而union文件系统用于保存镜像并使容器变得短暂。

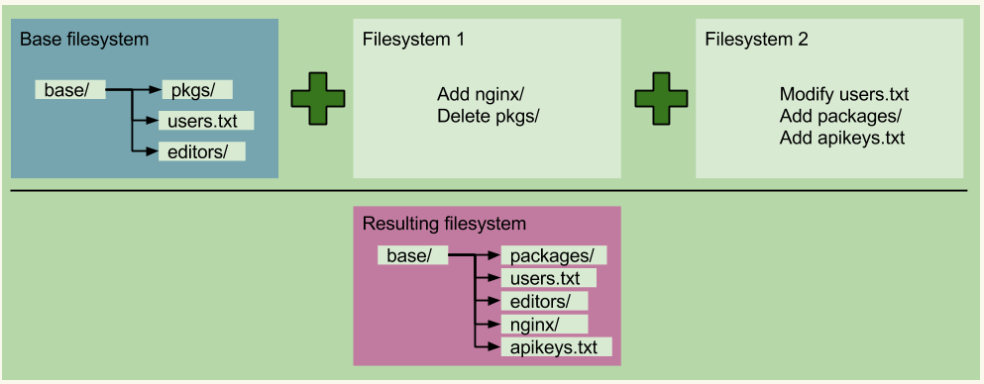
Cgroups

这是Linux内核功能，它让两件事情变成可能：

* 限制Linux进程组的资源占用（内存、CPU）
* 为进程组制作PID、UTS、IPC、网络、用户及装载命名空间

Union文件系统

Union文件系统允许通过union装载来达到一个分层的积累变化，在union文件系统里，文件系统可以被装载在其他文件系统之上，其结果就是一个分层的积累变化。每个装载的文件系统表示前一个文件系统之后的变化集合，就像是一 个diff。



镜像只是一个json，它指定了从该镜像运行的容器的特性，union装载点保存在哪里，要坐骑开什么端口等等。每个镜像与一个union文件系统相关联，每个Docker上的union文件系统都有一个上层。

容器

容器之所以是短暂的，是因为当你从镜像上创建一个容器，Docker会创建一个空白的union文件系统（当容器停止，该容器的union文件系统会被丢弃。）加载在与该镜像关联的union文件系统之上。

卷所做的是在容器内指定一个目录，以便在union文件系统之外保存它。

操作步骤

1. 安装Docker

Docker命令工具需要root权限才能工作。你可以将你的用户放入docker组来避免每次都要使用sudo。

1. 从公共registry下载镜像：

$>docker pull ubuntu:latest

1. 列出你的镜像：

$>dock images

1. 从该镜像上创建一个容器：

$>docker run --rm -ti ubuntu /bin/bash

root@4638a40c2fbb:/# ls

root@4638a40c2fbb:/# exit

--rm：告诉Docker一旦运行的进程退出就删除容器。（测试时非常有用，可免除杂乱）

-ti：告诉Docker分配一个伪终端并进入交互模式。（不要在生产容器中打开这些标志）

ubuntu：这是容器立足的镜像

/bin/bash：要 运行的命令，因为我们以交互模式启动，它将显示一个容器的提示符。

在运行run命令时，可指定链接、卷、端口、窗口名称（如果没提供，Docker将分配一个默认名称）等。

1. 在后台运行容器

$>docker run -d ubuntu ping 8.8.8.8

输出的是分配的ID。

1. 检查容器是否开始运行了：

$>docker ps

1. 检查容器里正在发生什么（自动分配的名称：loving\_mcclintock）：

$>docker exec –ti loving\_mcclintock /bin/bash

root@31c68e9c09a0:/# ps -aux|grep ping

root@31c68e9c09a0:/# exit

1. 工