3.什么是容器

Kubernetes使用Linux容器技术来实现应用的隔离。因此在深入学习k8s之前，我们需要先学习容器的基础知识以便更好地去理解k8s的原理机制。

揭开容器的神秘面纱

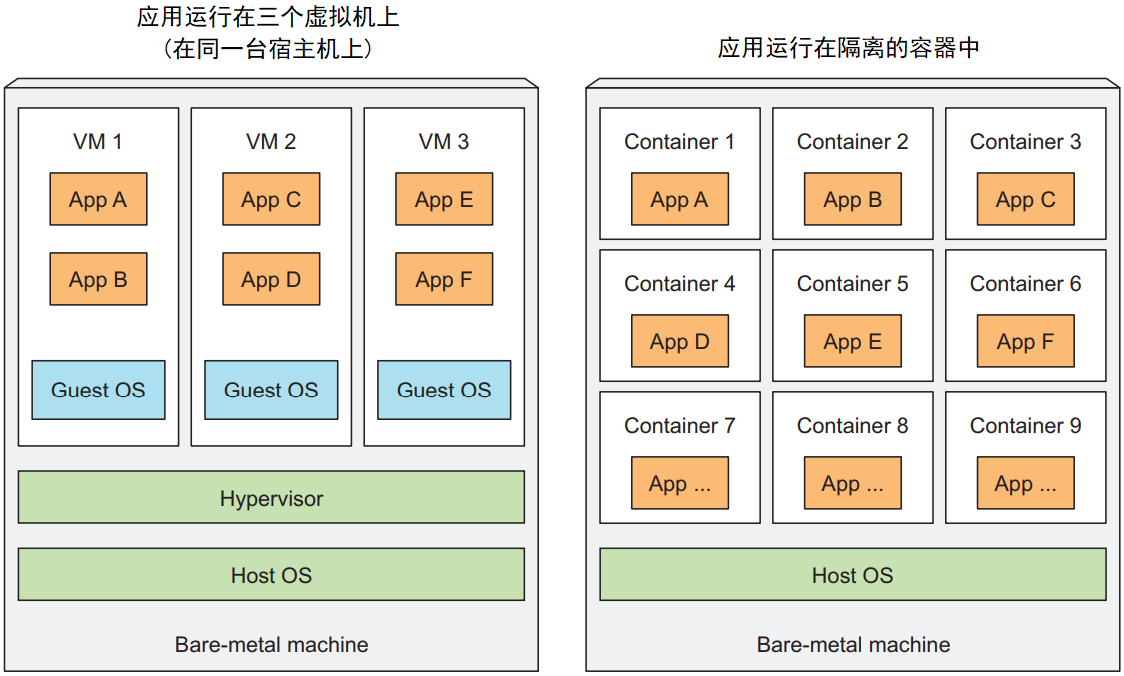
当一个应用只由较少数量的大组件构成时，完全可以给每个组件分配一个专用的虚拟机，以及通过给每个组件分配它自己的操作系统实例来隔离它们的环境。
但是当这些组件变得越来越小且数量变得越来越多时，就不能给每个组件分配专用的VM，除非你不想节约硬件资源、降低硬件成本。但是硬件资源只是一方面，每个VM通常还需要独立配置和管理，想象一下有大量的虚拟机等着你去配置和管理时你会是什么心情。这不仅是力资源的浪费，还会让系统管理员的工作变得更加高负荷。

利用Linux容器技术隔离应用组件

越来越多的开发人员开始转向Linux容器技术，而不是使用虚拟机来隔离每个微服务环境（或者通常说的进程）。容器技术使我们能够在同一台宿主机上运行多个服务，不仅为每个服务提供不同的环境，还将它们彼此隔离，就像虚拟机一样，但是需要的开销更小。
容器中运行的进程实际上还是在宿主机的操作系统上，跟其他进程没啥区别，不像虚拟机，进程是运行在不同的操作系统上的。但是需要注意的是，容器中的进程仍然是与其他进程隔离的。对于进程本身而言，看起来就像是机器和操作系统上运行的唯一进程。

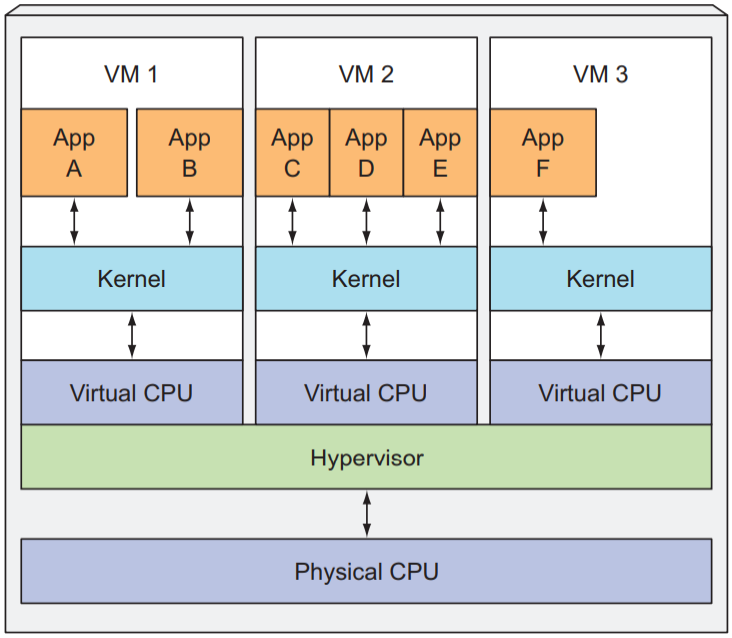
虚拟机 VS 容器

与虚拟机相比，容器更加轻量级。同样的硬件情况下，基于容器，我们可以运行更多的应用组件。这主要是因为虚拟机还需要运行它自己的一组系统进程，这些进程需要额外的计算资源，而且应用本身所在的进程也需要消耗资源。而对于容器来说，其本身实际上就是一个运行在宿主机上被隔离的进程，只消耗应用本身需要消耗的资源，再无任何其他进程的开销。
比较尴尬的情况是，由于虚拟机比较耗资源，导致没有足够的资源为每个应用程序分配一个专用的虚拟机，我们最终可能会将多个应用分组部署到每个虚拟机内。**当使用容器时，我们可以而且应该为每个应用分配一个容器**。因此，在同一台裸机上可以运行更多的应用程序。

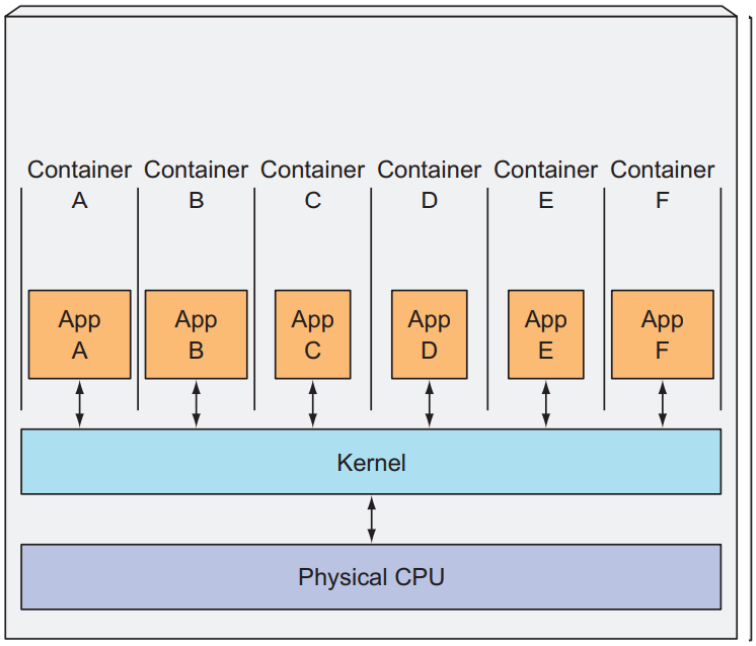


当在一台主机上运行三个虚拟机的时候，你就拥有了三个完全隔离的操作系统，它们运行并共享一台裸机。宿主机操作系统和一个**虚拟层（Hypervisor）**位于这些虚拟机下面，这个虚拟层会将物理硬件资源分成一些更小块的虚拟资源，从而被每个虚拟机内的操作系统使用。应用程序在虚拟机中运行，会调用虚拟机操作系统内核程序，然后内核程序会通过Hypervisor在宿主机的物理CPU上执行x86指令。而多个容器则会对运行在宿主机操作系统上的同一个内核执行调用，该内核是唯一一个在宿主机CPU上执行x86指令的内核。通过使用容器技术，CPU不需要做任何虚拟化，而虚拟机是需要对CPU做虚拟化的。可以通过下图了解应用程序在虚拟机或容器中使用CPU方式的差异：

应用程序运行在多个VM中



应用程序运行在隔离的容器中

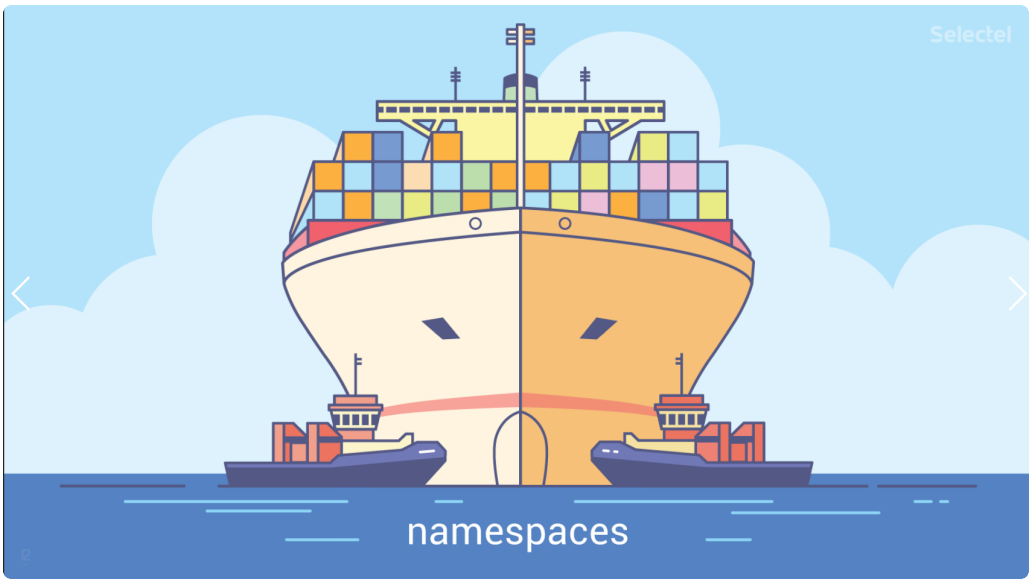


虚拟机的主要优点是提供了一个完全隔离的环境，因为每个虚拟机都运行在自己的Linux内核之上，而容器都是调用同一个内核，这显然是有安全隐患的。硬件资源有限的情况下，如果需要隔离的进程不多，使用虚拟机可能是一个选择。然而如果要在同一台机器上运行更多相互隔离的进程的话，容器技术才是更好的选择，因为它开销更少。需要注意的是，每个虚拟机都会运行它自己的一组系统服务（也即进程）而所有的容器是不会的，因为它们都运行在同一个OS之上。这就意味着运行一个容器是不需要像虚拟机那样还要先开机。容器上的进程是可以很快就可以被启动起来的。

容器隔离机制

可能你会好奇，容器是怎么对运行在同一个操作系统上的进程进行隔离的呢。这里就不得不提到两个机制。第一个就是Linux的**命名空间（Namespace）**，
它可以确保每个只能看到它自己的系统视图（文件、进程、网络接口、主机名等等）。第二个是**Linux控制组（cgroups）**，它限制了进程可以使用的资源量，包括CPU、内存、网络带宽等等。

命名空间



默认情况下，每个Linux系统初始化的时候只有一个命名空间。所有的系统资源，比如文件系统、进程ID、用户ID、网络接口以及其他资源都是属于这个命名空间。但是我们也可以创建额外的命名空间以及在它们之间分配资源。对于一个进程，我们可以选择让其在某一个命名空间下运行，该进程就只能看到这个命名空间下的资源视图。不过，命名空间是有多种类型的，一个进程并不是属于某一个命名空间，而是属于每种类型的一个命名空间，因此一个进程可能属于多个命名空间，只是这些命名空间是不同类型的。

命名空间的种类有如下几种：

- MNT: 管理系统文件挂载点 (MNT: Mount)
- PID: 进程隔离 (PID: Process ID)
- NET: 管理网络接口 (NET: Networking)
- IPC: 管理对IPC资源的访问 (IPC: Inter-Process Communication)
- UTS: 隔离内核和版本标识符 (UTS: Unix Timesharing System)
- User ID (user)

每一种命名空间被用来隔离一组特定的资源。例如，UTS命名空间就决定了运行在该命名空间下的进程所能看到的主机名和域名。通过给两个进程指定两个不同的UTS命名空间，能够使它们看到不同的本地主机名。换句话说，对于这两个进程来说，就好像运行在两个不同的机器上（至少就主机名来说是这样）。

同样的，一个进程属于什么网络命名空间决定了运行在进程中的应用程序所能看到的网络接口。每个网络接口属于一个命名空间，但是可以从一个命名空间转移到另一个命名空间。每个容器使用它自己的网络命名空间，因此每个容器看到的都是属于它自己的一套网络接口。现在我们应该明白命名空间是如何被用来隔离运行在容器中的应用。

cgroup

还可以通过cgroups来限制容器能够消耗的资源来实现进程的隔离。cgroups是Linux的一个内核功能，它可以限制一个进程或者一组进程的资源使用。一个进程的资源（CPU、内存、网络带宽等等）使用量不能超过分配给它的量。通过这样方式，进程就不能过度地使用为其他进程保留的资源，这就和每个进程运行在单独的机上上类似。

如果想了解关于cgoups的知识，可以参考：https://tech.meituan.com/2015/03/31/cgroups.html