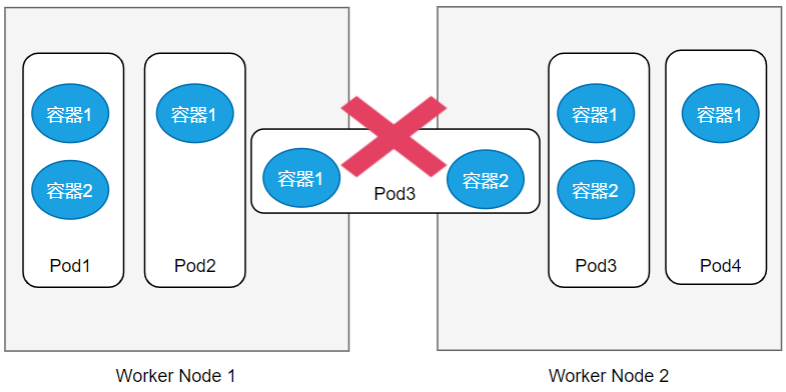
12.什么是Pod

1.什么是Pod

Pod是Kubernetes的基本构建单元，通常我们不会单独地对容器进行部署，而是对容器所在的Pod进行部署、操作。Pod可以包含一个或者多个容器，在实际使用过程中，一个Pod只包含一个容器是比较常见的，而且是最常用的应用方式，除非特别需要，比如应用之间耦合度和依赖比较高，一般都不推荐使用多容器Pod的方式。如果一个Pod包含多个容器，那么这些容器肯定都是在一个工作节点上运行。Pod是不会跨多个工作节点进行部署的，就像下图所展示的一样：



先思考一下如下几个问题：

- 为什么需要Pod?
- 为什么不直接使用容器?
- 为什么需要同时运行多个容器?
- 可以将所有应用程序放到一个容器吗?

2.为什么运行多个容器比单个容器运行多个进程更好

假设一个应用程序由多个进程组成，这些进程彼此之间通过IPC（进程间通信）或者本地存储的文件进行通信，这就要求这些进程运行在同一台机器上。由于在K8s中我们总是在容器中运行进程而且每个容器就像一个隔离的机器，因此你可能会觉得在一个容器中运行多个进程是合理的，但是实际上我们不应该这样做。

容器的设计初衷就是单个容器只运行一个进程（除非进程本身产生子进程）。如果在单个容器中运行多个毫不相关的进程，那么你就需要负责保证所有进程的正常运行、管理进程的日志等等。例如，必须引入一种机制，以便在进程崩溃时自动重新启动它们。同样，所有这些进程都会将日志记录到相同的标准输出中，因此我们将很难弄清楚哪个进程记录了什么。

因此，我们需要让每个进程运行在它自己的容器中。这才是使用Docker和Kubenetes的正确姿势。

3.为什么需要Pod

既然不能将多个进程放到一个容器中运行，那么很显然我们就需要一个更高级别的结构来组合容器，并将它们作为一个单元来进行管理。将相关的容器放到同一个Pod中，让紧密相连的进程一起运行在几乎完全相同的环境中，就好像容器变成了进程，Pod变成了容器，进程运行在同一个容器中，同时彼此又保持一定的隔离。这样一来，我们就能利用容器提供的所有的特性，并给进程一种运行在一起的错觉。

4.同一Pod中容器之间的部分隔离

我们知道容器彼此之间是完全隔离的，但有时我们并不希望它们完全隔离。可以通过隔离一组容器的方式来达到这个目的。在每个组中的所有容器可以共享某些资源，但并不是全部，因而这些容器并不是完全隔离的。

Kubernetes通过配置Docker来让一个Pod中的所有容器共享同一组Linux命名空间，而不是每个容器都有自己的一组命名空间。

由于一个Pod中的所有容器都是在相同的Network和UTS命名空间下运行，因此它们共享相同的主机名和网络接口。同样地，一个Pod中的所有容器也在相同的IPC命名空间下运行，因此可以通过IPC进行通信。

在最新的Kubernetes和Docker版本中，它们还可以共享同一个PID命名空间。当一个Pod中的容器使用单独的PID命名空间时，如果在容器中执行ps aux命令的话，我们就只能看到容器自己的进程。

但是对于文件系统，情况就有些不一样了。因为大多数容器的文件系统来自于容器镜像。默认情况下，每个容器的文件系统是与其他容器完全隔离的。但是也可以使用Kubernetes中的Volume来使这些容器共享文件目录。

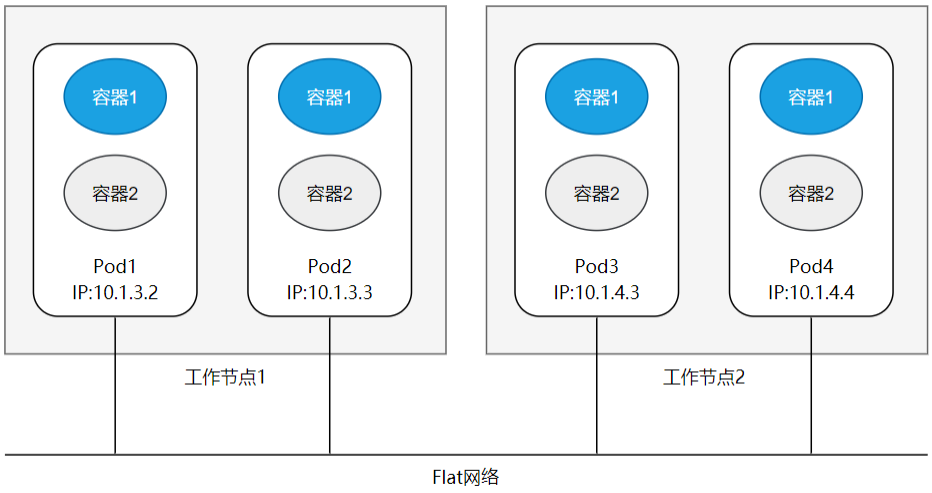
5.容器是如何共享相同的IP和端口空间

这里需要强调的一点是，因为一个Pod中的容器都运行在同一个Network命名空间下，因此它们共享相同的IP地址和端口空间。这意味着在同一个Pod中的容器里运行的进程不能绑定到同一个端口上，否则会造成端口冲突。但这只针对同一个Pod中的容器，不同Pod中的容器是不会有端口冲突的问题的，因为每个Pod都有单独的地址和端口空间。

同一个Pod中的所有容器也拥有相同的loopback网络接口（即环回网卡），因此一个容器可以通过localhost与同一个Pod中的其他容器进行通信。

6.Pod间的扁平网络

Kubernetes集群中的所有Pod都是在一个扁平的、共享的网络地址空间中，这意味着每个Pod可以通过其他Pod的IP地址来对其进行访问。它们之间不存在NAT（网络地址转换）网关。
当两个Pod彼此之间发送网络数据包的时候，它们都会把对方实际的IP地址看成数据包中的源IP。
如下图，每个Pod都获得一个可路由的IP地址，其他Pod都可以在该IP地址下看到这个Pod：



因此，Pod之间的通信是很简单的。不管是将两个Pod调度到同一个工作节点还是不同的工作节点上，这都是不重要的。在这两种情况下，无论工作节点间实际的网络拓扑结构如何，这两个Pod中的所有容器都能彼此之间跨无NAT的扁平网络进行通信，就像局域网（LAN）中的计算机一样。每个Pod都会获得它自己的IP地址，而且可以被其他Pod通过专门为Pod间通信而建立的网络访问到。
总而言之，Pod就是逻辑主机，其行为与非容器世界的物理主机或者虚拟机非常相似。同一个Pod中运行的进程与运行在同一个物理主机或者虚拟机上的进程相似，只是每个进程被封装到了一个容器之中。