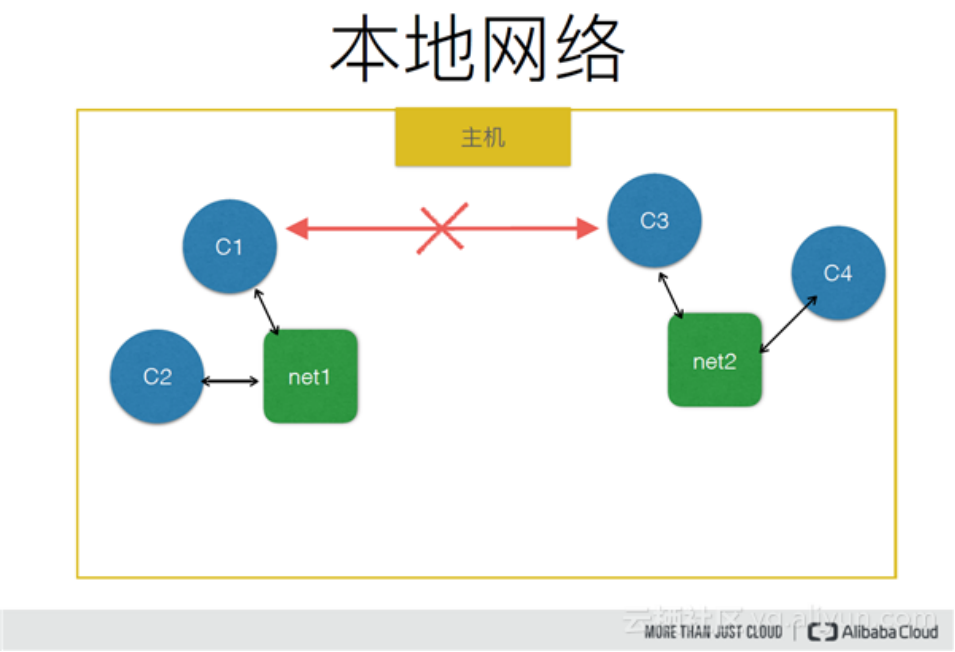
Docker 网络深度解读

摘要：在云栖 TechDay：Docker 深度实践专场，来自阿里云容器服务的王炳燊分享了题为《Docker 网络深度解读》的演讲。他主要介绍了 Docker 概念和默认网络、Docker 跨主机网络、阿里云服务的网络方案。

Docker 概念和默认网络

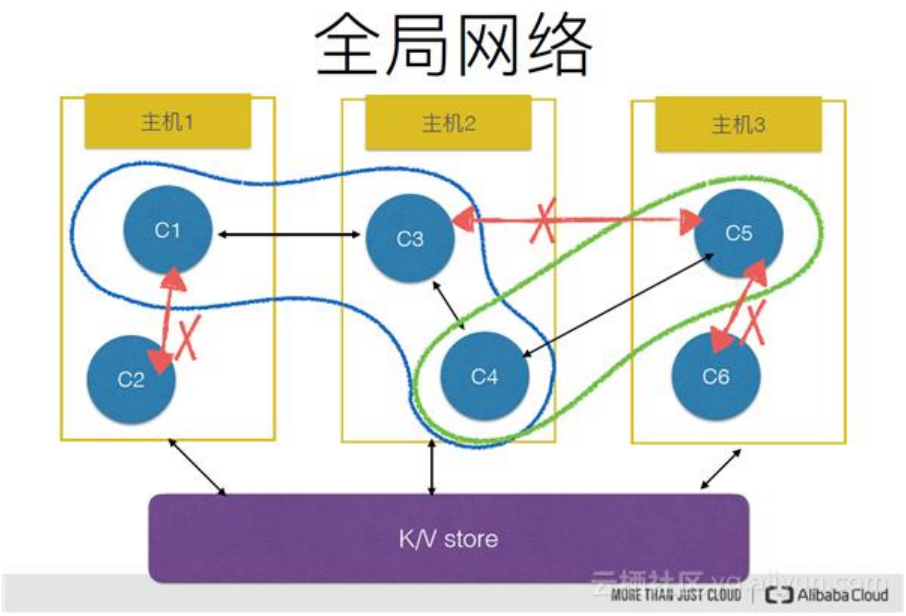
什么是 Docker 网络呢？总的来说，网络中的容器们可以相互通信，网络外的又访问不了这些容器。具体来说，在一个网络中，它是一个容器的集合，在这个概念里面的一个容器，它会通过容器的 IP 直接去通信，又能保证在这个集合外的一些容器不能够通过这个容器 IP 去通信，能做到网络隔离。网络这个概念是由网络的驱动去创建、管理的。网络的驱动又分为全局的和本地的，全局的意思是这个网络可以跨主机，没必要说我的两个容器非要在一个主机上才能通过这个网络去通信，我可以在不同主机上还是通过容器的 IP 相互通信。

本地网络



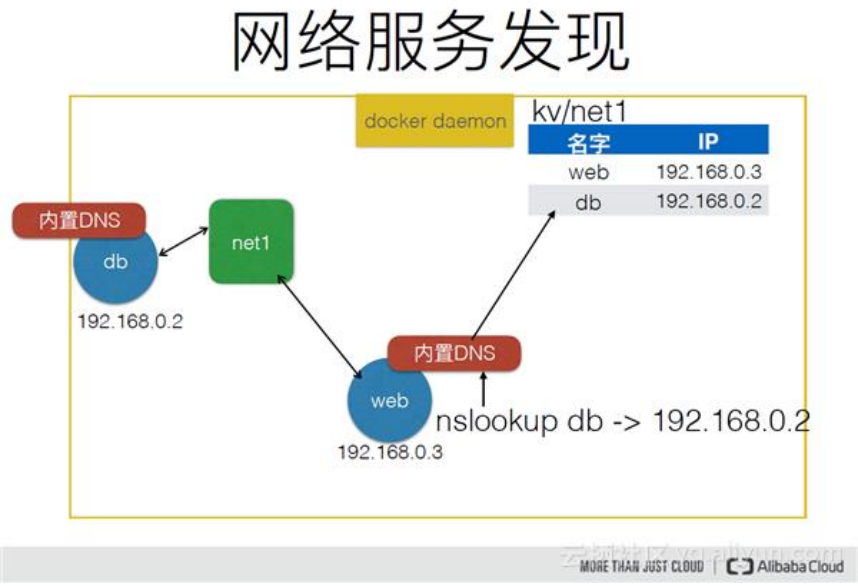
首先我们看一个本地网络的图，这个图中有一台主机，上面跑了四个容器，分别分布在两个网络里面，一个是 NET1，一个是 NET2。NET1 里面的两个容器是可以互相通过自己的 IP 通信的，但是 NET1 里面的容器和 NET2 里面的容器又是隔离的，这样就是 Docker 的一个网络概念。

全局网络



全局网络意思是，一个网络里面的容器是可以跨主机的。C1 的容器可以和主机 2 上的 C3 容器，通过它们 IP 直接通信，而不用管所在的主机在哪个地方。这个图里面我们还可以看到一个细节，同一个容器可以加入到两个网络里面，比如 C4 它可以和网络 1 里面的两个容器通信，也可以和网络 2 里面的 C5 进行通信，这样我们就可以实现更加复杂的一些组合。比如 C5 是一个数据库的容器，我不想让这个数据被前面的 WEB 应用或者其他的应用去访问到，只想让我的中间键应用去访问到。这样就可以通过让中间键应用加入到一个后台网络、一个前台网络的组合来实现这种互联和网络的控制。

网络服务发现

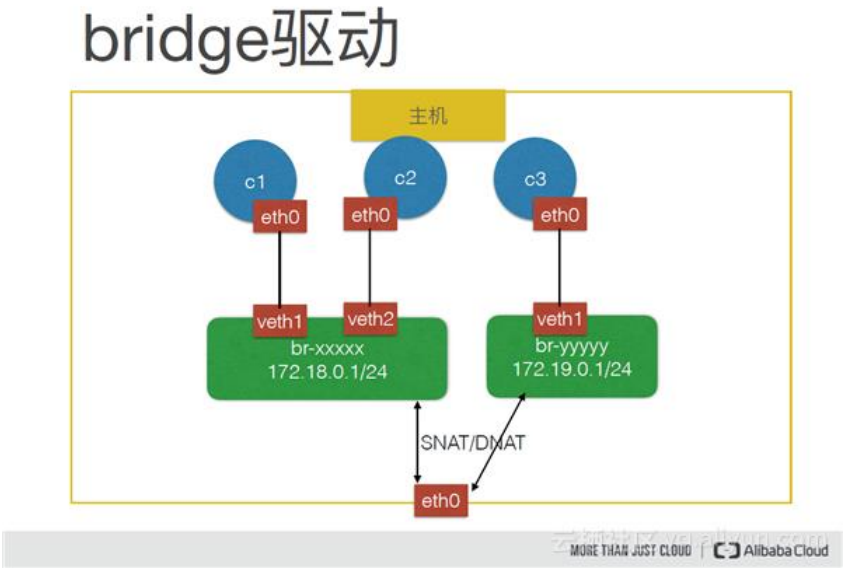


在 1.10 版本增加了网络服务发现，什么意思呢？比如，我的一个网络有两个容器，我们可以直接通过 IP 进行通信，但是我的容器它可能某一段时间跪掉了，它可能在跪掉之后再去

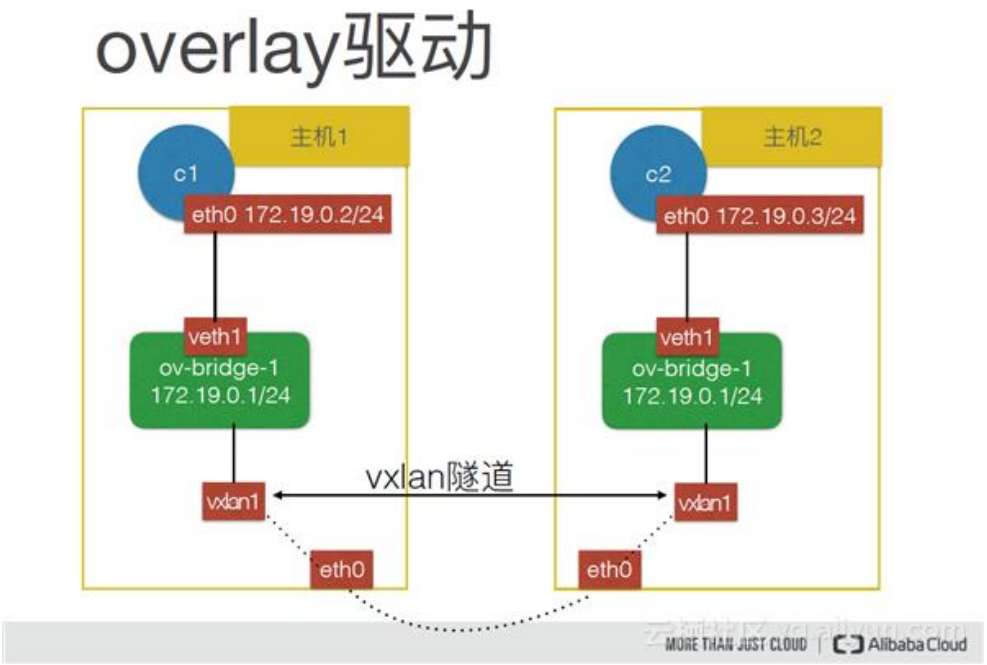
重启，我这时就不知道它的 IP 了，或者它一关闭之后，我这边感知不到，这样就要触发很多复杂的一些处理流程。在 1.10 时它去做这样的一件事，它在每个容器里面内置了一个 DNS 的服务器，在服务器 RUN 起来之后，会写一个容器名，或者是这个容器的一个别名和它 IP 的一个解析。外部容器去访问我的 DB 容器时，它能够通过 DB 这个名字来解析到 DB 容器的 IP 地址，这样在 DB 容器重启之后，这个解析会动态去调整，就能实现一个服务的发现。

Docker 默认的网络驱动

Docker 默认的会有一些网络驱动，它本地的网络驱动有这几种。none 网络驱动意思是，我创建一个容器，但是没有它的一些网络配置，单独一个命名空间，它不能和其他容器通信，也不能访问外网。host 的网络驱动和宿主机共享网络栈、它和宿主机看到同样的接口，包括它自己暴露的端口，都是和宿主机在同一个网络空间里面。



bridge 网络驱动，它是一个本地的网络驱动，上图主机上有两个 bridge 网络，它实际上是通过 Linux 的 bridge 去驱动的，通过容器里面挂载一个网卡，对应的是在宿主机上一个 veth，容器的请求会直接转发到这个 veth 上面，bridge 上面会对这个请求进行洪泛，它就能到达这个网络里面的另外一个容器。同理，另外一个网络，它跟这个网络里面的容器没有在同一个网桥上面，所以他们之间是互相不能通信的。bridge 驱动还会实现另外一个功能，比如说 C1 的容器，他要访问公网，它通过 bridge 出去的时候，他最终到达宿主机的接口，他会做一个网络地址的转换，把容器的 IP 转换成宿主机的 IP，这样就能访问公网。或者我的容器可能希望服务映射到外面去，让别的机器或别的公开服务可以访问到，他可以做一个 DNAT 一个规则，比如我的容器是 Nginx，它里面是 80 端口，我想映射到主机的一个 9080，可以直接通过主机的 9080 去访问，他可以去做一个 DNAT 的配置，这是 bridge 的驱动。



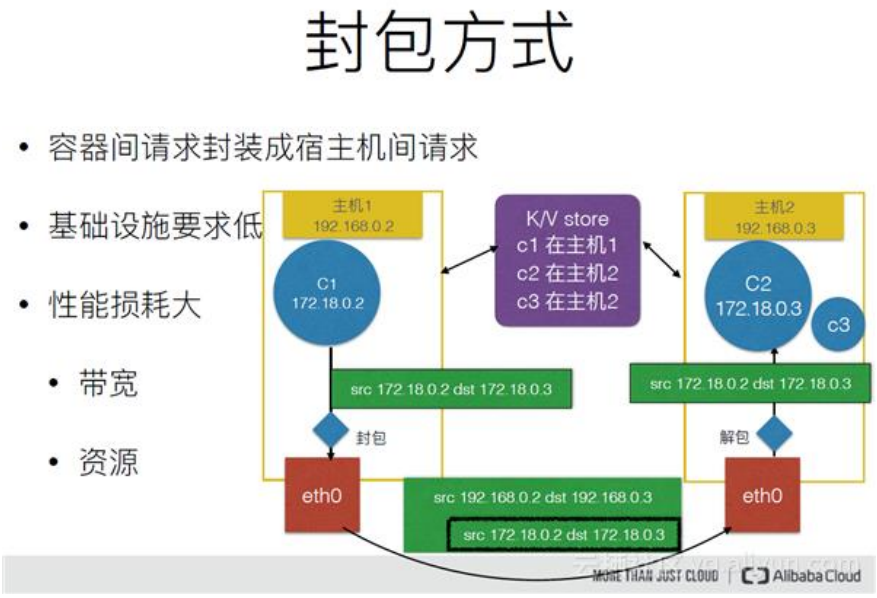
Docker 默认还有一个 overlay 的驱动，它主要是去实现跨主机的一个通信，它会在主机之间通过 vxlan 的方式打一个隧道，实现我的主机上的容器去访问主机上的容器，最终会转换成一个 vxlan 里面的通信。

Docker 跨主机网络

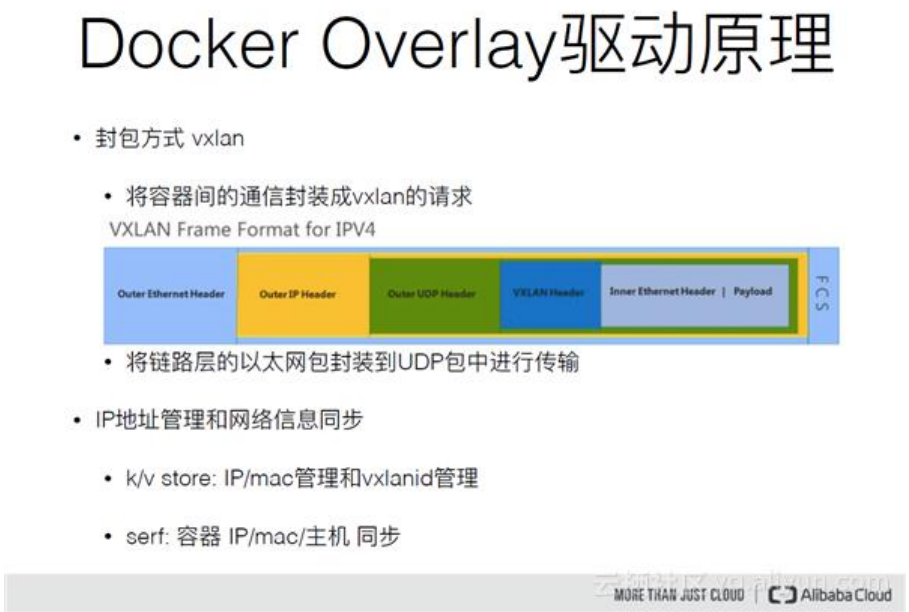
没有跨主机网络时，比如一个应用有两个服务，两个服务不可能部署到同一个机器上，这样没办法做到高可用，所在的主机一挂，上面所有的服务都挂了。如果让它部署到两个主机上面，但是两个主机上面这个容器的 IP 在主机外面是看不到的。所以只能把容器的 IP 转换成了一个宿主机的一个 IP，或者宿主机的一个访问端点，才能让外部的主机访问。所以，以前的方式是通过端口映射，把容器的一个端口映射到主机的一个端口，让另外一个主机和这个主机映射到这个容器上去通信。这样会带来哪些问题？首先我通过端口映射，比如说我映射到 8090 端口，我另外再想起一个容器。他就得映射到另外一个端口，因为宿主机上只有这一个 8090 端口，它不可能说共享这个端口，所以一个主机上去起容器的时候，需要管理端口的列表。映射出去之后是宿主机的一个端口，这样这个端口就是对外暴露的，可能要去控制一下，这个端口只能说主机要去访问，这样要去做一些安全的配置，所以这样的整个架构是非常复杂的。在 Docker1.9 之后，增加了跨主机的网络，它就可以直接通过容器的 IP 去通信，通过这个外部跟它是隔离的，这样又能做到安全，我就算再起一个容器，它容器的 IP 端口是不会冲突的。

封包模式

封包模式，比如 overlay 驱动，它是用 VXLAN 的协议去把容器之间的请求封装成 VXLAN 的请求，将链路层的以太网包封装到 UDP 包中进行传输。



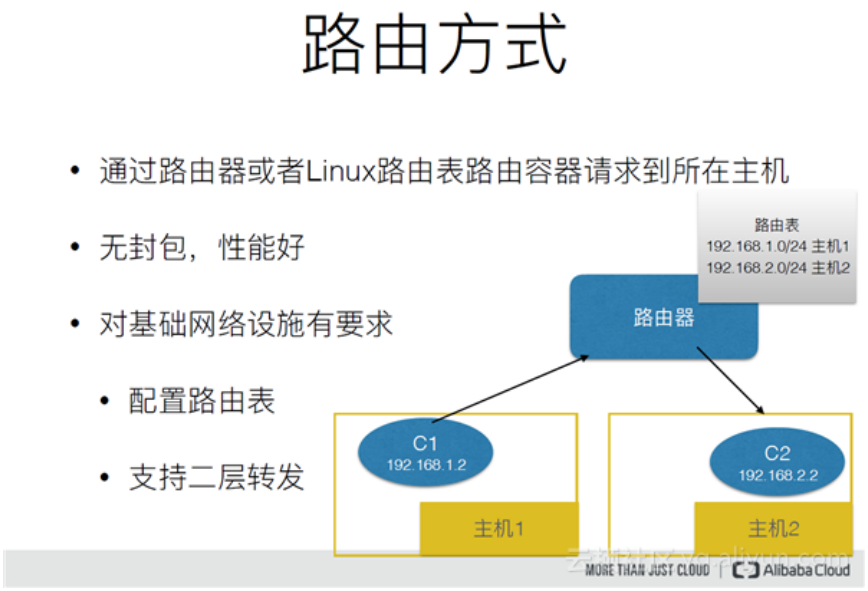
这里有一个简单的图来介绍封包模式，首先举个例子，从 C1 去访问宿主机上 C2 的容器，先通过 C1 发出这个包之后，它首先进行封包，要查找中心化存储 C2 是在哪个主机上的，查到的是这个主机上，就把这个请求的包封装成宿主机之间的包，把这个包送达到对应宿主机上，再通过一定的约定去解包，最终转发到另外一个主机上的一个容器。其缺点是对带宽的损耗比较大，因为看到这里去封包，它肯定是把这个容器的包上面又加了一层包，这个包上面肯定会有一些自己的占用，一般像封包模式，MTU 最终都会小于宿主机之间的 MTU，它还会造成一些资源占用。比如说在这个地方去封包的时候，它可能会占用 CPU 去做一下封包操作，还会占用 CPU 去做解包的操作，所以会增加主机上的负载，但是它的好处是对基础设施要求比较低，它只需要两个宿主机之间可以三层互通。



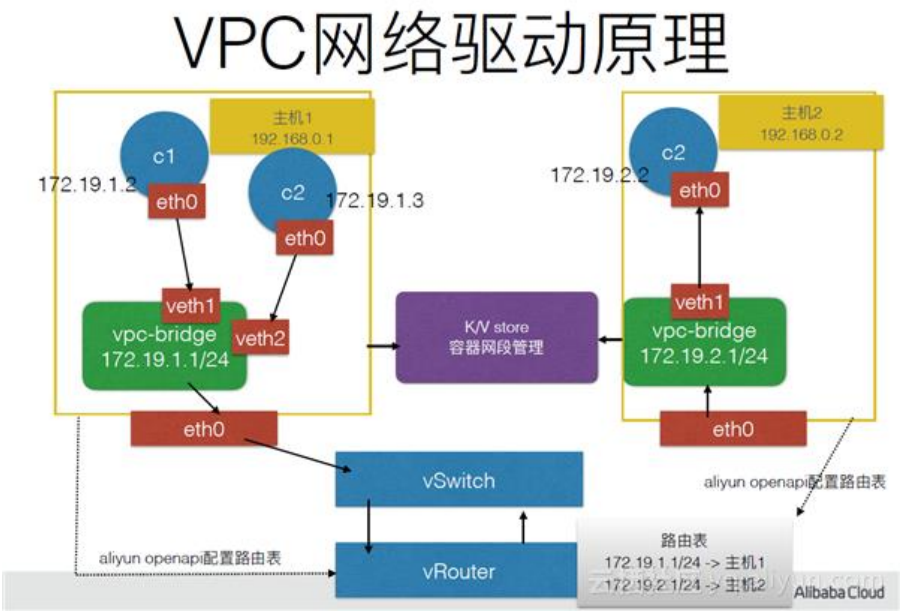
下面我们以 Docker 的 overlay 驱动来介绍下封包的实现，它的封包方式是基于 VXLAN。上图是它的针格式，VXLAN 内部把一个二层的包、链路层的一个请求封装成一个 VXLAN 的一个

包，这里面会有一个约定的信息，比如 VXLAN ID，还有一个外部的 UDP 的头，最终会把这个包通过 UDP 发往对应的主机上面去，对应的主机再做 VXLINE 的解包。Docker 还会做 IP 的地址管理和网络信息同步。

路由模式



如上图，比如，主机 1 上的容器想要访问主机 2 的容器，首先这个机器上是没有主机 2 的 IP 地址，它出了机器之后到达路由器，再通过路由器上的路由表，去把对应的请求网端转发到对应的主机 2 上面，主机 2 上面是有这个容器的 IP，所以它就能够做到这个容器的跨主机通信。它的好处在于它没有封包，所以它的性能很好，但是它对于基础设施有一些要求，比如我的基础设施要支持、能够配置这个路由表，如果用 Linux 路由要支持二层的转发。



VPC 网络驱动是在阿里云的 VPC 基础上，能够实现容器互通的一种方案。首先在 Docker DM

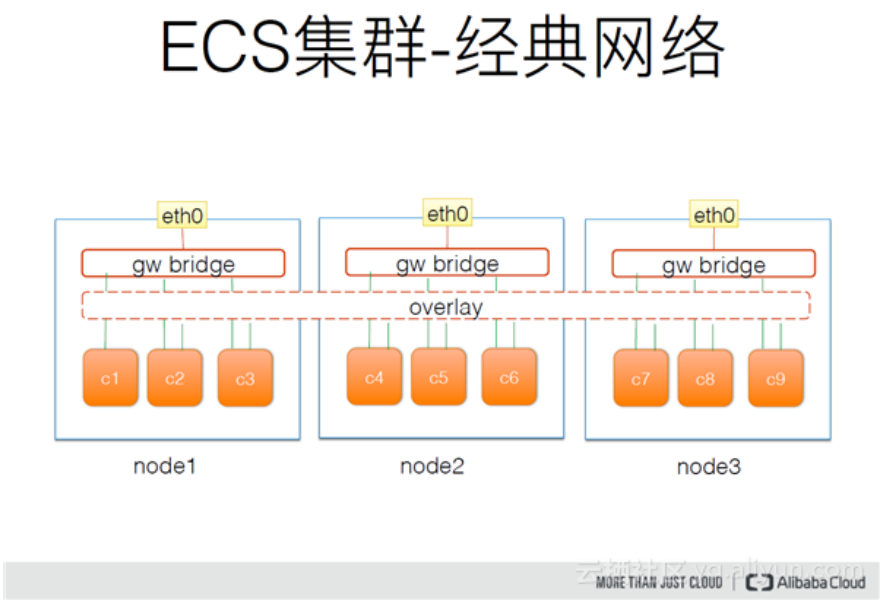
启动时，或者是在创建网络时，会从每个机器的一个中心化存储里面，拿到两个自己的网端，比如这个机器上的网端，比如左边主机上的网端就是 172.19.1.1/24，它通过阿里云的 openapi 去配置这个转发表，把这个网端的地址都转发到这个主机上面。另外一个主机启动时，把它拿到一个可用的网端，去配置阿里云的一个路由表，把这个网端的请求转发到自己的主机上面，比如说我这边的一个容器想要去访问主机 2 上 C2 的容器时，它这个包最终到达，通过 vSwitch 到达阿里云 VPC 的 vRouter 时，通过查询路由表把实时的请求包转发到主机 2 上面，最终实现 C1 和 C2 的通信。

跨主机网络方案对比

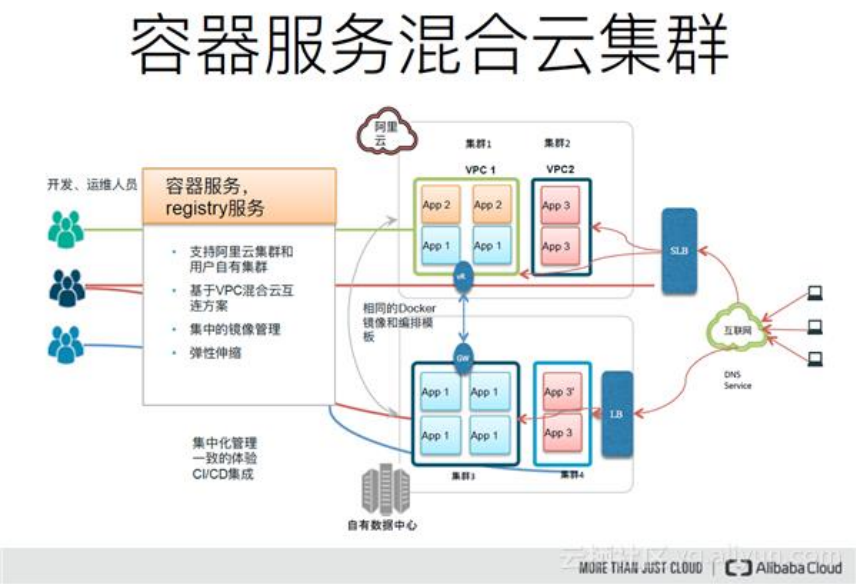
封包模式的优点是对基础设施的要求比较低，但是它性能损耗比较大，路由模式是没有封包的，所以它性能比较好，但是对基础设施有一定要求，比如说要支持路由表，或者要支持二层的转发。

阿里云服务的网络方案

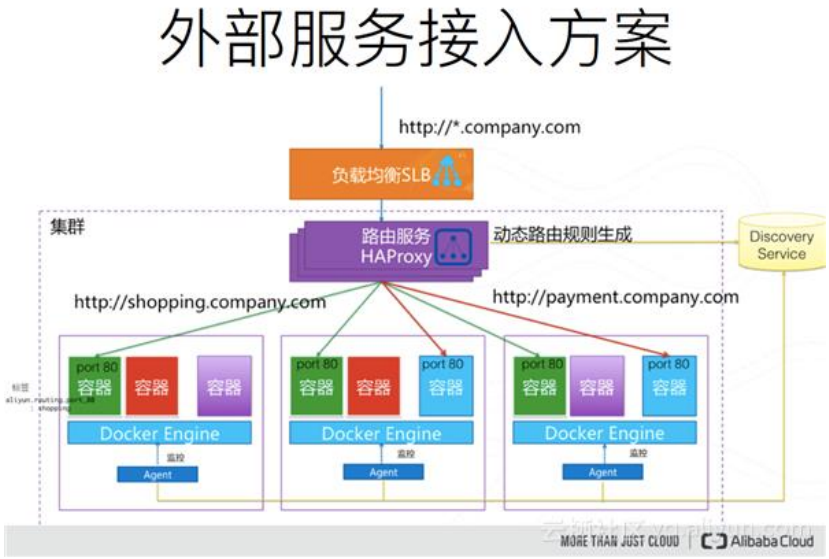
容器服务上面现在有两种集群，一种是阿里云的 ECS 集群，这种集群的节点都是在阿里云的同地域或者同一个 VPC 下面的 ECS，或者是混合云的集群，阿里云的混合云集群节点是在用户自己建立的机房，或者分布在不同的云供应商上面。



在阿里云的 ECS 集群上面，网络架构是这样的，容器之间去通信，还是通过 overlay 驱动，它的好处是对基础设施要求比较低，所以它只需要主机之间互通就可以了，容器去访问外网，或者容器想要暴露端口给外网就通过一个 gateway_bridge 的网桥、本地的一个驱动，这个是供容器访问外网和容器的端口映射。在 ECS 集群上面，VPC 网络是通过 VPC 驱动的路由表把对应的请求转发到容器所在机器上面，它去访问外网和做端口映射也是和刚才所介绍的 overlay 是同样的。



混合云集群是结合阿里云的 VPC 技术去实现容器之间的网络互通。在混合云集群里面是通过专线的方式，把对应的数据中心的请求转发到我的数据中心里面，或者把数据中心的请求访问 VPC 的，通过专线的一个路由也可以转发到 VPC 里面，再通过路由表把容器的请求转发到对应的主机上面，最终实现的方式是把我 VPC 里面的容器和数据中心里面的容器互通。



刚才我们介绍的都是容器之间通信，我们可以不通过端口映射的方式做到端口不冲突，怎么能够做到我想要外部访问时，也能做到就一个负载均衡，怎么能做到一个机器上面部署多个容器，然后端口不冲突呢？这时候阿里云就提供了这样的一个负载均衡方案。比如说我的网站有两个域名，一个是购物的域名，一个是付款的，但是我总共就三台机器，每个服务要做到高可用，所以起了很多的容器，这样在同一个主机上就要去处理，怎么让它保障单个不冲突，容器服务是提供一个这样的方案，我们在集群里面会部署一套路由的一个服务，首先请求，比如说我的公司域名的泛解析会解析到我的路由服务上面去，路由服务再通过请求的域名 Host，去把这个请求直接转发到容器上面，最终实现我的一个请求可以转发到容器上面，但是我的容器并没有映射出端口，也不需要去解决端口冲突的问题。