Kubernetes容器网络

Docker容器网络

节点内容器间通信

每一个运行的容器都可以有自己的Network Namespace，也可以称之为独立的网络栈，包括网卡，本地回环设备，路由表以及iptables规则等内容。
可以将容器看作是一个个不同的"主机"，容器间的网络通信将转换成主机间的网络通信

网桥

如果想要实现两台主机的通信，最简单的就是用一根网线连起来。如果想要实现多台主机的通信，最简单的做法就是将它们连接到同一个交换机上

在 Linux 中，能够起到虚拟交换机作用的网络设备，是网桥。工作于数据链路层，主要做的事情就是根据 MAC 地址学习来将数据包转发到网桥的不同端口上

为什么是 MAC 地址而不是IP地址？ → 在以太网协议中规定，同一局域网中的一台主机要和另一台主机进行直接通信，必须要知道目标主机的MAC地址

所以，为了实现容器间的通信，Docker会在宿主机创建名为docker0的网桥 → Linux使用 ifconfig 查看

连接容器

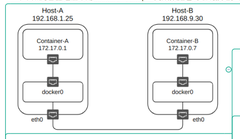
有了网桥，现在要做的就是将容器和网桥连接起来，通过 Veth Pair实现 → Veth Pair总是成对出现，一端在网桥中，一端在容器中

Ubuntu中，可使用 sudo apt-get install bridge-utils 安装brctl工具，查看 Veth Pair 对 → brctl show

docker0完全可以看做是一台基于二层网络的交换机，而位于节点中的容器就是连接在同一个交换机上的主机

跨节点容器通信

单节点内容器通信采用docker0+veth pair完成数据包的准确投递，但是，在多节点网络中，容器并不是连接在同一个docker0网桥上的



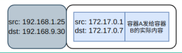
A、B两台主机IP互通，容器A(172.17.0.1)想要和容器B(172.17.0.7)通信，如何处理？

假设容器A知道容器B所在的节点IP

由于A、B节点互通，那么容器A发给容器B的包就可以让节点A的某个进程帮忙处理下

具体做法为: 将容器A的IP包封装在节点A发送给节点B的IP中，就像套娃一样，此时IP包的源地址为192.168.1.25，目标地址为192.168.9.30

IP包到达节点B以后，让某进程对该IP包进行解包，拿出容器A发给容器B的IP包，由OS将包转发给docker0，最终达到容器B



如同特洛伊木马一样，真正的容器通信包封装在节点通信的包内

容器A不知道容器B所在的节点IP

仍然可以使用"套娃"的方式，但是需要建立容器IP与节点IP的映射关系表

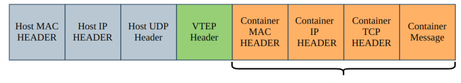
在发送节点间通信的包之前，就需要在这张表中查找容器B所在的节点IP，而后将封装的数据包发送至该节点

不管哪种方式，都要求容器的IP在整个集群中唯一

VXLAN

Virtual eXtensible Local Area Network，虚拟可扩展的局域网，通过三层的网络来搭建虚拟的二层网络 → 即在现有的IP运输层构建出基于MAC地址通信的链路层

在上面的容器通信中有提到，容器A发送的数据包需要某进程将其包裹，作为数据封装在节点间通信的数据包上，那么在VXLAN的实现中，则由vtep负责报文的封包和解包

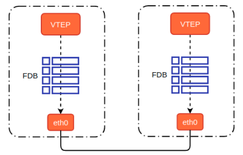


VXLAN为什么使用UDP协议，而不是TCP协议？

在前面的介绍中有提到，VXLAN实际上是在三层的网络基础之上来搭建一个虚拟的二层网络，也就是说，VXLAN的目的是提供一个虚拟的二层网络: 容器集群的通信可达性

所以，数据包的可靠传输并不由VXLAN进行保证，而是更上层的TCP协议，只不过上层发出的TCP报文被UDP协议包裹而已，当出现丢包时TCP协议层仍然会重传该数据包，以达到可靠数据传输的目的

既然VXLAN不需要保证数据传输的可靠性，那么传输速度更快的UDP协议当然是首选



Linux VXLAN关键组件

VTEP → VXLAN 网络的边缘设备，用来进行 VXLAN 报文的处理(封包和解包)，对原始包添加UDP头部和取下UDP头部

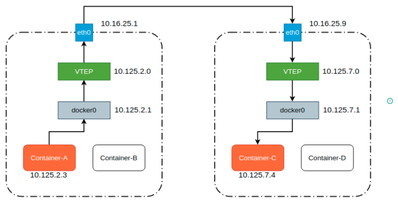
VNI → VXLAN 网络标识符，可以认为是虚拟网络标志，用户标志网络租户

FDB → Linux 网桥维护的一个二层转发表，用于保存VXLAN网络中VTEP的MAC地址、IP地址信息

82:2b:db:4e:2a:66 dev vxlan0 dst 192.168.2.16 self

FDB表可手动维护，或者使用广播组进行自动学习

VXLAN通信详述



在每一个节点中都有一个VTEP设备，用于构建二层虚拟网络

基本模型

目的: 容器A(10.125.2.3)能够与容器C(10.125.7.4)进行通信

过程

假设容器A和容器B的通信协议使用TCP协议，那么从容器A出来的TCP包的目的地址即为10.125.7.4，源目的地址为10.125.2.3，由于10.125.7.4并不属于docker0的网段，所以docker0将会将该数据包转发给VTEP

源IP地址: 10.125.2.3	目的IP地址: 10.125.7.4	源端口号: 23650	目的端口号: 80

VTEP在收到目的IP地址为10.125.7.4的数据包之后，查阅本地路由表，将数据包准备发送至节点B上的VTEP设备

目的VTEP的MAC地址: XXXX	源IP地址: 10.125.2.3	目的IP地址: 10.125.7.4	源端口号: 23650	目的端口号: 80

节点B上的VTEP设备的MAC如何确定？查询本机的ARP记录表

查询FDB数据库

在有了VTEP的MAC地址之后，还需要知道VTEP所在的节点的IP地址，这个信息如何获取？

UDP Header	VXLAN Header	目的VTEP的MAC地址: XXXX	源IP地址: 10.125.2.3	目的IP地址: 10.125.7.4	源端口号: 23650	目的端口号: 80

所以，VTEP的IP地址、路由信息、MAC地址等信息需要在节点上线时广播给集群中的所有节点，以建立查找表

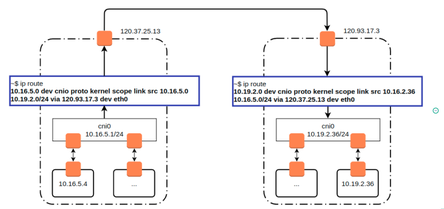
得到了目标主机的IP地址之后，就是标准的主机通信过程了，通过ARP获取目标主机的MAC地址，即可将该包发送出去

目标主机MAC地址: XXXX	目标主机IP地址: 10.16.25.9	UDP Header	VXLAN Header	目的VTEP的MAC地址: XXXX	源IP地址: 10.125.2.3	目的IP地址: 10.125.7.4	源端口号: 23650	目的端口号: 80

简单来说，VXLAN的过程就是借助了预先设置的查找表来实现的通信过程

Flannel—host-gw

前面所述的VXLAN容器网络解决方案主要依赖于Linux内核对VXLAN的实现，属于在三层网络之上虚拟出来的二层网络解决方案。除了VXLAN的解决方案以外，还存在着纯三层网络解决方案，例如Flannel的host-gw模式，以及Calico项目



如左图所示，host-gw模式其实就是 Host Gateway，让某一个主机节点作为网关

10.19.2.0/24 via 120.93.17.3 dev eth0 → 该条规则的含义为: 所有目的地址为10.19.2.0/24网段的数据包，均使用eth0网卡发送，并且下一个路由地址为120.93.17.3

而 120.93.17.3 该IP地址正好是Node-B的IP地址，并且内部IP地址为 10.19.2.36的容器也在该节点上

可以看到，host-gw 模式只需要维护节点的路由规则即可实现容器间的跨节点通信，其效率要优于VXLAN模式(更少的封包、解包) → BTW，Flannel插件中host-gw模式下的相关"信息"，例如容器与节点的映射关系，是维护在Etcd中的

但是，host-gw 由于是将节点作为路由网关使用，那么根据路由协议，两个路由之间必须是二层连通的。所以，host-gw模式要求集群中节点之间是二层连通的