链滴

Kubernetes 网络 -flannel 网络插件详解 个人笔记总结

作者：IwasawaMasami

原文链接：https://ld246.com/article/1579223129877

来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20190720.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h2 id="前言-">前言：</h2>
因为 Kubernetes 的网络可以使用第三方网络插件，所以给我们提供了多样化的网络解决方案，
我们可以根据自身情况选择自己需要的网络方案。
<h3 id="CNM---CNI阵营-">CNM & CNI 阵营：</h3>
• 容器网络发展到现在，形成了两大阵营，就是 Docker 的 CNM 和 Google、CoreOS、Kubere
etes 主导的 CNI。首先明确一点，CNM 和 CNI 并不是网络实现，他们是网络规范和网络体系，从
发的角度他们就是一堆接口，你底层是用 Flannel 也好、用 Calico 也好，他们并不关心，CNM 和 CN
关心的是网络管理的问题。
• CNM (Docker LibnetworkContainer Network Model)
• CNI（Container Network Interface）

首先我们得先了解 k8s 网络的设计原则，然后才能更好的理解 flannel 网络的应用</stron
>
<h3 id="Kubernetes网络设计模型-">Kubernetes 网络设计模型：</h3>
•在 Kubernetes 网络中存在两种 IP（Pod IP 和 Service Cluster IP），Pod IP 地址是实际存在
某个网卡(可以是虚拟设备)上的，Service Cluster IP 它是一个虚拟 IP，是由 kube-proxy 使用 Iptabl
s 规则重新定向到其本地端口，再均衡到后端 Pod 的。
基本原则：
• 每个 Pod 都拥有一个独立的 IP 地址（IPper Pod），而且假定所有的 pod 都在一个可以直接连通
、扁平的网络空间中。
设计原因：
• 用户不需要额外考虑如何建立 Pod 之间的连接，也不需要考虑将容器端口映射到主机端口等问题。
/p>
网络要求：
• 所有的容器都可以在不用 NAT 的方式下同别的容器通讯；所有节点都可在不用 NAT 的方式下同所
容器通讯；容器的地址和别人看到的地址是同一个地址。
<h3 id="K8S网络主要解决以下网络通信问题-">K8S 网络主要解决以下网络通信问题:</h3>
• 同一 pod 下容器与容器的通信；
• 同一节点下不同的 pod 之间的容器间通信；
• 不同节点下容器之间的通信；
• 集群外部与内部组件的通信；
• pod 与 service 之间的通信；
<h3 id="1-容器间通信-">1、容器间通信：</h3>
同一个 <a href="https://ld246.com/forward?goto=https%3A%2F%2Fwww.kubernetes.org
cn%2Ftags%2Fpod" target="_blank" rel="nofollow ugc">Pod 的容器共享同一个网络命名
间，它们之间的访问可以用
localhost 地址 + 容器端口就可以访问。
<img src="https://ld246.com/images/img-loading.svg" alt="t1.png" data-src="https://b3
ogfile.com/file/2019/09/t1-eae44b28.png?imageView2/2/interlace/1/format/jpg">
<h3 id="2-同一Node中Pod间通信-">2、同一 Node 中 Pod 间通信：</h3>
同一 Node 中 Pod 的默认路由都是 docker0 的地址，由于它们关联在同一个 docker0（cni0
网桥上，地址网段相同，所有它们之间应当是能直接通信的。
<img src="https://ld246.com/images/img-loading.svg" alt="t2.png" data-src="https://b3
ogfile.com/file/2019/09/t2-0540cce4.png?imageView2/2/interlace/1/format/jpg">
<h3 id="3-不同Node中Pod间通信-">3、不同 Node 中 Pod 间通信：</h3>
不同 Node 中 Pod 间通信要满足 2 个条件：
Pod 的 IP 不能冲突； 将 Pod 的 IP 和所
的 Node 的 IP 关联起来，通过这个关联让 Pod 可以互相访问。所以就要用到 flannel 或者
calico 的网络解决方案。
对于此场景，情况现对比较复杂一些，这就需要解决 Pod 间的通信问题。在 Kubernetes 通过 fl
nnel、calico 等网络插件解决 Pod 间的通信问题。以 flannel 为例说明在 Kubernetes 中网络模型，f
annel 是 kubernetes 默认提供网络插件。Flannel 是由 CoreOs 团队开发社交的网络工具，CoreOS

团队采用 L3 Overlay 模式设计 flannel， 规定宿主机下各个 Pod 属于同一个子网，不同宿主机下的od 属于不同的子网。
<img src="https://ld246.com/images/img-loading.svg" alt="t3.png" data-src="https://b3ogfile.com/file/2019/09/t3-29411050.png?imageView2/2/interlace/1/format/jpg">
<h3 id="容器网络方案-">容器网络方案：</h3>
隧道方案（ Overlay Networking ）
隧道方案在 IaaS 层的网络中应用也比较多，大家共识是随着节点规模的增长复杂度会提升，而出了网络问题跟踪起来比较麻烦，大规模集群情况下这是需要考虑的一个点。
• Weave：UDP 广播，本机建立新的 BR，通过 PCAP 互通
• Open vSwitch（OVS）：基于 VxLan 和 GRE 协议，但是性能方面损失比较严重
• Flannel：UDP 广播，VxLan
• Racher：IPsec
<h3 id="CNI-Plugin---Flannel插件解决方案-">CNI Plugin – Flannel 插件解决方案：</h3>
• Flannel 是 CoreOS 团队针对 Kubernetes 设计的一个网络规划服务，简单来说，它的功能是集群中的不同节点主机创建的 Docker 容器都具有全集群唯一的虚拟 IP 地址。
• 在默认的 Docker 配置中，每个节点上的 Docker 服务会分别负责所在节点容器的 IP 分配。这导致的一个问题是，不同节点上容器可能获得相同的内外 IP 地址。并使这些容器之间能够之间通过 IP 地址相互找到，也就是相互 ping 通。
• Flannel 的设计目的就是为集群中的所有节点重新规划 IP 地址的使用规则，从而使得不同节点的容器能够获得“同属一个内网”且”不重复的”IP 地址，并让属于不同节点上的容器能够直接通过网 IP 通信。
• Flannel 实质上是一种“覆盖网络(overlaynetwork)”，也就是将 TCP 数据包装在另一种网络里面进行路由转发和通信，目前已经支持 udp、vxlan、host-gw、aws-vpc、gce 和 alloc路由等数据转发方式，默认的节点间数据通信方式是 UDP 转发。
例如通过使用后端（backend）为 vxlan 的转发方式的 flannel 架构图:
<img src="https://ld246.com/images/img-loading.svg" alt="t4.png" data-src="https://b3ogfile.com/file/2019/09/t4-93aac0c9.png?imageView2/2/interlace/1/format/jpg">
简而言之就是 Flannel 之所以可以搭建 kubernets 依赖的底层网络，是因为它可以实现以下两：
• 它给每个 node 上的 pod 分配相互不想冲突的 IP 地址；
• 它能给这些 IP 地址之间建立一个覆盖网络，同过覆盖网络，将数据包原封不动的传递到目标容内。
下图是通过 vxlan 的方式实现节点间的通信：
<img src="https://ld246.com/images/img-loading.svg" alt="t5.png" data-src="https://b3logfie.com/file/2019/09/t5-f35761a2.png?imageView2/2/interlace/1/format/jpg">
<h3 id="flannel数据包转发方式-backend原理解析-">flannel 数据包转发方式(backend 原理解析)</h3>
<h4 id="1-hostgw">1.hostgw</h4>
• hostgw 是最简单的 backend，它的原理非常简单，直接添加路由，将目的主机当做网关，直路由原始封包。
例如下图，我们从 etcd 中监听到一个 EventAdded 事件 subnet 为 10.1.15.0/24 被分配给主机Public IP 192.168.0.100，hostgw 要做的工作就是在本主机上添加一条目的地址为 10.1.15.0/24，关地址为 192.168.0.100，输出设备为上文中选择的集群间交互的网卡即可。对于 EventRemoved件，只需删除对应的路由。
<img src="https://ld246.com/images/img-loading.svg" alt="t6.png" data-src="https://b3ogfile.com/file/2019/09/t6-1a7fbaa2.png?imageView2/2/interlace/1/format/jpg">
但是 hostgw 有个不足的地方就是，我们知道当 backend 为 hostgw 时，主机之间传输的就是始的容器网络封包，封包中的源 IP 地址和目的 IP 地址都为容器所有。这种方法有一定的限制，就是求所有的主机都在一个子网内，即二层可达，否则就无法将目的主机当做网关，直接路由。
<h4 id="2-udp">2.udp</h4>
• 而 udp 类型 backend 的基本思想是：既然主机之间是可以相互通信的（并不要求主机在一个网中），那么我们就可以直接将容器的网络封包作为负载数据在集群的主机之间进行传输呢？这就是谓的 overlay。具体过程如图所示：
<img src="https://ld246.com/images/img-loading.svg" alt="t7.png" data-src="https://b3logfi

e.com/file/2019/09/t7-20bb5ebe.png?imageView2/2/interlace/1/format/jpg">
当容器 10.1.15.2/24 要和容器 10.1.20.2/24 通信时，因为该封包的目的地不在本主机 subnet，因此封包会首先通过网桥转发到主机中。最终在主机上经过路由匹配，进入网卡 flannel0。需要注的是 flannel0 是一个 tun 设备，它是一种工作在三层的虚拟网络设备，而 flanneld 是一个 proxy，会监听 flannel0 并转发流量。当封包进入 flannel0 时，flanneld 就可以从 flannel0 中将封包读出由于 flannel0 是三层设备，所以读出的封包仅仅包含 IP 层的报头及其负载。最后 flanneld 会将获取封包作为负载数据，通过 udp socket 发往目的主机。同时，在目的主机的 flanneld 会监听 Public IP 所在的设备，从中读取 udp 封包的负载，并将其放入 flannel0 设备内。由此，容器网络封包到达目主机，之后就可以通过网桥转发到目的容器了。
最后和 hostgw 不同的是，udp backend 并不会将从 etcd 中监听到的事件里所包含的 lease 息作为路由写入主机中。每当收到一个 EventAdded 事件，flanneld 都会将其中的 subnet 和 Public P 保存在一个数组中，用于转发封包时进行查询，找到目的主机的 Public IP 作为 udp 封包的目的地。
<h4 id="3-vxlan">3.vxlan</h4>
• 首先，我们对 vxlan 的基本原理进行简单的叙述。从下图所示的封包结构来看，vxlan 和上文到的 udp backend 的封包结构是非常类似的，不同之处是多了一个 vxlan header，以及原始报文中了个二层的报头。
<img src="https://ld246.com/images/img-loading.svg" alt="t8.png" data-src="https://b3ogfile.com/file/2019/09/t8-a5a38075.png?imageView2/2/interlace/1/format/jpg">
上图所示，当主机 B 加入 flannel 网络时，和其他所有 backend 一样，它会将自己的 subnet 10 1.16.0/24 和 Public IP 192.168.0.101 写入 etcd 中，和其他 backend 不一样的是，它还会将 vtep 备 flannel.1 的 mac 地址也写入 etcd 中。
之后，主机 A 会得到 EventAdded 事件，并从中获取上文中 B 添加至 etcd 的各种信息。这个候，它会在本机上添加三条信息：
1.路由信息：所有通往目的地址 10.1.16.0/24 的封包都通过 vtep 设备 flannel.1 设备出，发往的网关地址为 10.1.16.0，即主机 B 中的 flannel.1 设备。
2.fdb 信息：MAC 地址为 MAC B 的封包，都将通过 vxlan 首先发往目的地址 192.168 0.101，即主机 B
3.arp 信息：网关地址 10.1.16.0 的地址为 MAC B
现在有一个容器网络封包要从 A 发往容器 B，和其他 backend 中的场景一样，封包首先通过网转发到主机 A 中。此时通过，查找路由表，该封包应当通过设备 flannel.1 发往网关 10.1.16.0。通进一步查找 arp 表，我们知道目的地址 10.1.16.0 的 mac 地址为 MAC B。到现在为止，vxlan 负载分的数据已经封装完成。由于 flannel.1 是 vtep 设备，会对通过它发出的数据进行 vxlan 封装（这步是由内核完成的，相当于 udp backend 中的 proxy），那么该 vxlan 封包外层的目的地址 IP 地该如何获取呢？事实上，对于目的 mac 地址为 MAC B 的封包，通过查询 fdb，我们就能知道目的主的 IP 地址为 192.168.0.101。
最后，封包到达主机 B 的 eth0，通过内核的 vxlan 模块解包，容器数据封包将到达 vxlan 设备 fl nnel.1，封包的目的以太网地址和 flannel.1 的以太网地址相等，三层封包最终将进入主机 B 并通过由转发达到目的容器。
<h4 id="查看flannel配置文件-">查看 flannel 配置文件：</h4>
可以通过 <code>cat /var/run/flannel/subnet.env</code> 的命令查看 flannel 的子网分配况
• flannel 在每个 Node 上启动了一个 flanneld 的服务，在 flanneld 启动后，将从 etcd 中读取置信息，并请求获取子网的租约。所有 Node 上的 flanneld 都依赖 etcd cluster 来做集中配置服务，et d 保证了所有 node 上 flanned 所看到的配置是一致的。同时每个 node 上的 flanned 监听 etcd 上数据变化，实时感知集群中 node 的变化。flanneld 一旦获取子网租约、配置后端后，会将一些信息入/run/flannel/subnet.env 文件。
<h3 id="demo演示过程-待更新-">demo 演示过程（待更新）</h3>