# kubernetes之Flannel网络插件部署

Kubernetes系统上Pod网络的实现依赖于第三方插件，而Flannel是由CoreOS主推的目前比较主流的容器网络解决方案，CNI插件有两种功能：网络配置和网络策略，由于flannel比较简单，并不支持网络策略，flannel项目自身只是一个框架，真正提供网络功能的是它的后端实现，目前，Flannel支持三种不同后端实现，分别是：

* UDP
* VXLAN
* host-gw

UDP是Flannel项目最早支持的一种方式，是性能最差的方式，目前已被废弃。

用的最多的是VXLAN和host-gw模式的部署

## Flannel VXLAN模式配置

### Flannel部署

在刚好安装完k8s集群之上部署flannel。

直接应用官方的yaml文件：

root@k8s-master:~*# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681ece4de4c0d86c5cd2643275/Documentation/kube-flannel.yml*

clusterrole.rbac.authorization.k8s.io/flannel created

clusterrolebinding.rbac.authorization.k8s.io/flannel created

serviceaccount/flannel created

configmap/kube-flannel-cfg created

daemonset.extensions/kube-flannel-ds-amd64 created

daemonset.extensions/kube-flannel-ds-arm64 created

daemonset.extensions/kube-flannel-ds-arm created

daemonset.extensions/kube-flannel-ds-ppc64le created

daemonset.extensions/kube-flannel-ds-s390x created

输出如下结果表示运行正常：

root@k8s-master:~*# kubectl get ds -n kube-system*

NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE

kube-flannel-ds-amd64 2 2 2 2 2 beta.kubernetes.io/arch=amd64 98s

kube-flannel-ds-arm 0 0 0 0 0 beta.kubernetes.io/arch=arm 98s

kube-flannel-ds-arm64 0 0 0 0 0 beta.kubernetes.io/arch=arm64 98s

kube-flannel-ds-ppc64le 0 0 0 0 0 beta.kubernetes.io/arch=ppc64le 98s

kube-flannel-ds-s390x 0 0 0 0 0 beta.kubernetes.io/arch=s390x 98s

运行正常后，flanneld会在宿主机的/etc/cni/net.d目录下生成自已的配置文件，kubelet将会调用它。

网络插件运行成功后，Node状态才Ready

root@k8s-master:~*# kubectl get node*

NAME STATUS ROLES AGE VERSION

k8s-master Ready master 18m v1.13.1

k8s-node01 Ready <none> 16m v1.13.1

flannel运行后，在各Node宿主机多了一个网络接口：

root@k8s-master:~*# ifconfig*

flannel.1 Link encap:Ethernet HWaddr 6a:43:8c:e4:2a:77

inet addr:10.244.0.0 Bcast:0.0.0.0 Mask:255.255.255.255

inet6 addr: fe80::6843:8cff:fee4:2a77/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1450 Metric:1

root@k8s-node01:~*# ifconfig*

flannel.1 Link encap:Ethernet HWaddr 7a:a1:2e:85:a9:1c

inet addr:10.244.1.0 Bcast:0.0.0.0 Mask:255.255.255.255

inet6 addr: fe80::78a1:2eff:fe85:a91c/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1450 Metric:1

从上面的结果可以知道：

1. flannel默认就是VXLAN模式，即Overlay Network。
2. flanneld创建了一个flannel.1接口，它是专门用来封装隧道协议的，默认分给集群的Pod网段为10.244.0.0/16。
3. flannel给k8s-master节点配置的Pod网络为10.244.0.0段，给k8s-node01节点配置的Pod网络为10.244.1.0段，如果有更多的节点，以此类推。

启动一个副本为3的nginx容器：

root@k8s-master:~*# kubectl run nginx --image=nginx:1.10 --port=80 --replicas=3*

查看pod：

root@k8s-master:~# kubectl get pod -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

nginx-6b647cb88-24j29 1/1 Running 0 23m 10.244.0.3 k8s-master <none> <none>

nginx-6b647cb88-ft8wc 1/1 Running 0 23m 10.244.0.2 k8s-master <none> <none>

nginx-6b647cb88-g4mqt 1/1 Running 0 33m 10.244.1.4 k8s-node01 <none> <none>

其中，两个Pod运行在节点k8s-master上，其中一个Pod配置的IP为10.244.0.2，

现在，在此node查看网络接口

root@k8s-master:~*# ifconfig*

cni0 Link encap:Ethernet HWaddr 0a:58:0a:f4:01:01

inet addr:10.244.0.1 Bcast:0.0.0.0 Mask:255.255.255.0

inet6 addr: fe80::c048:c9ff:fe09:f54e/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1450 Metric:1

RX packets:3316 errors:0 dropped:0 overruns:0 frame:0

TX packets:3387 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

当有容器运行后，节点之上多了个虚拟接口cni0，其IP为10.244.0.1，它是由flanneld创建的一个虚拟网桥叫cni0，在Pod本地通信使用。

flanneld为每个Pod创建一对veth虚拟设备，一端放在容器接口上，一端放在cni0桥上。

使用brctl查看该网桥：

root@k8s-master:~*# brctl show cni0*

bridge name bridge id STP enabled interfaces

cni0 8000.0a580af40001 no veth0fda9673

veth6388ea61*#刚好有两个容器的网络接口挂在了cni0网桥之上。*

测试正常访问：

*#在宿主机上测试*

root@k8s-master:~*# ping 10.244.0.2*

PING 10.244.1.4 (10.244.1.4) 56(84) bytes **of** data.64 bytes **from** 10.244.1.4: icmp\_seq=1 ttl=63 time=2.01 ms

在现有的flannel VXLAN网络中，两台主机上的pod间通信，肯定是可以的，如下两pod：

*#进入Pod测试*

root@k8s-master:~*# kubectl exec -it nginx-6b647cb88-ft8wc -- /bin/sh*

*# ping 10.244.1.4 http://www.gw638.cn*

PING 10.244.1.4 (10.244.1.4): 56 data bytes64 bytes **from** 10.244.1.4: icmp\_seq=0 ttl=62 time=2.587 ms64 bytes **from** 10.244.1.4: icmp\_seq=1 ttl=62 time=3.880 ms

那么容器跨主机是如何通信的呢，查看路由信息：

root@k8s-master:~*# ip route* 10.244.1.0/24 via 10.244.1.0 dev flannel.1 onlink

去往10.244.1.0/24网络的数据包发给本机的flannel.1接口，即进入二层隧道，然后封装VXLAN包，到达目标Node后，由目标Node上的flannel.1解封装。

一旦Node启动并加入Flannel网络之后，其它Node上的flanneld就会添加一条类似这样的路由规则，这就是默认的VXLAN网络。

因为是在k8s-master上 ping别人的，所以k8s-master是封装过VXLAN包的，抓包：

**#k8s-master**抓物理网卡的包

**tcpdump** **-i** **ens33** **-nn** **host** 10.3.1.20

16:46:09.302335 **IP** 10.3.1.20.53051 > 10.3.1.21.8472: **OTV**, **flags** [I] (0**x08**), **overlay** 0, **instance** 1**IP** 10.244.0.2 > 10.244.1.4: **ICMP** **echo** **request**, **id** 20, **seq** 360, **length** 64

16:46:09.302395 **IP** 10.3.1.20.59519 > 10.3.1.21.8472: **OTV**, **flags** [I] (0**x08**), **overlay** 0, **instance** 1**IP** 10.244.1.4 > 10.244.0.2: **ICMP** **echo** **reply**, **id** 20, **seq** 360, **length** 64

可以看到，在overlay里面是一个是上面ping的ICMP包。

VXLAN是Linux内核本身支持的一种网络虚拟化技术，是内核的一个模块，在内核态实现封装解封装，构建出覆盖网络，其实就是一个由各宿主机上的Flannel.1设备组成的虚拟二层网络。

由于VXLAN由于额外的封包解包，导致其性能较差，所以Flannel就有了host-gw模式，即把宿主机当作网关，除了本地路由之外没有额外开销，性能和calico差不多，由于没有叠加来实现报文转发，这样会导致路由表庞大。因为一个节点对应一个网络，也就对应一条路由条目。

host-gw虽然VXLAN网络性能要强很多。，但是种方式有个缺陷：要求各物理节点必须在同一个二层网络中。  
物理节点必须在同一网段中。这样会使得一个网段中的主机量会非常多，万一发一个广播报文就会产生干扰。  
在私有云场景下，宿主机不在同一网段是很常见的状态，所以就不能使用host-gw了。

VXLAN还有另外一种功能，VXLAN也支持类似host-gw的玩法，如果两个节点在同一网段时使用host-gw通信，如果不在同一网段中，即 当前pod所在节点与目标pod所在节点中间有路由器，就使用VXLAN这种方式，使用叠加网络。  
结合了Host-gw和VXLAN，这就是VXLAN的**Directrouting模式**

因此Flnnel的VXLAN模式有两种：

1. VXLAN： 原生的VXLAN，即扩展的虚拟LAN
2. Directrouting：直接路由型

## Flannel VXLAN之Directrouting模式配置

修改下载的kube-flannel.yml，将flannel的configmap对象改为：

net-conf.json: |

{

"Network": "10.244.0.0/16", *#默认网段*

"Backend": {

"Type": "VXLAN",

"Directrouting": true *#增加http://www.f-1.cc*

}

}

然后把原来的flannel删除，再重新apply：

root@k8s-master:~*# kubectl apply -f kube-flannel.yml*

clusterrole.rbac.authorization.k8s.io/flannel created

clusterrolebinding.rbac.authorization.k8s.io/flannel created

serviceaccount/flannel created

configmap/kube-flannel-cfg created

daemonset.extensions/kube-flannel-ds-amd64 created

daemonset.extensions/kube-flannel-ds-arm64 created

daemonset.extensions/kube-flannel-ds-arm created

daemonset.extensions/kube-flannel-ds-ppc64le created

daemonset.extensions/kube-flannel-ds-s390x created

删除重新部署需要删除原来的Flannel，所以应该在一开始就把Flannel规划好。

再来查看路由：

root@k8s-master:~*# ip route show***default** via 10.3.1.1 dev ens33 onlink 10.244.0.0/24 dev cni0 proto kernel scope link src 10.244.0.1 10.244.1.0/24 via 10.3.1.21 dev ens33

去往10.244.1.0/24网络的下一跳是10.3.1.21，从本机的物理接口ens33出去。这就是Directrouting。如果两个节点跨网段，则flannel自动降级为VXLAN模式。

## Flannel host-gw 配置

与Directrouting类似，将flannel的configmap对象改为：

**net-conf.json: |**

{

"Network": "10.244.0.0/16",

"Backend": {

"Type": "host-gw" *#修改*

}

}

其路由信息显示和Directrouting是相同的。

这就是flannel的配置方式。