**安装calico**

**方式一 IPIP模式**

****安装calico网络插件****

|  |
| --- |
| ****curl https://docs.projectcalico.org/manifests/calico.yaml -O**** |

**修改NetworkManager服务的配置以允许Calico管理网卡**

Calico 必须能够在宿主机上管理 cali\* 网卡。当 IPIP 被启用时(默认)，Calico 还需要能够管理 tunl\* 网卡。

注意: 许多Linux发行版都安装了 NetworkManager。默认情况下 NetworkManager 不允许 Calico 管理网卡。如果你的节点有 NetworkManager，在安装 Calico 之前，你需要按照这个步骤去 防止 NetworkManager 控制 Calico 网卡。

创建如下配置文件，/etc/NetworkManager/conf.d/calico.conf防止 NetworkManager 干扰接口：

|  |
| --- |
| [keyfile]  unmanaged-devices=interface-name:cali\*;interface-name:tunl\*;interface-name:vxlan.calico;interface-name:wireguard.cali |

cat > /etc/NetworkManager/conf.d/calico.conf <<EOF

[keyfile]

unmanaged-devices=interface-name:cali\*;interface-name:tunl\*;interface-name:vxlan.calico;interface-name:wireguard.cali

EOF

1. 配置kubelet

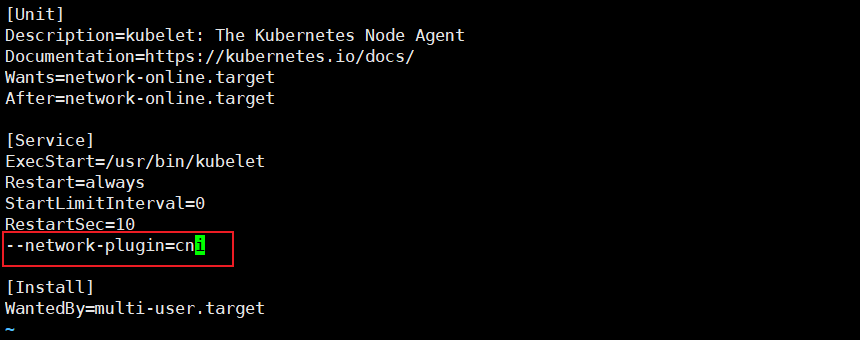
/usr/lib/systemd/system/kubelet.service

需要配置kubelet 让pod启动时使用calico网络插件，kubelet可以配置使用calico在启动时配置参数：

--network-plugin=cni

--network-plugin-dir=/etc/cni/net.d

Kubelet启动时，通过--network-plugin=cni启用CNI，通过--cni-bin-dir参数指定CNI插件所在主机目录（默认为/opt/cni/bin/）、通过--cni-conf-dir参数指定CNI配置文件所在主机目录（默认为/etc/cni/net.d）



**方式二 Tigera Calico Operator**



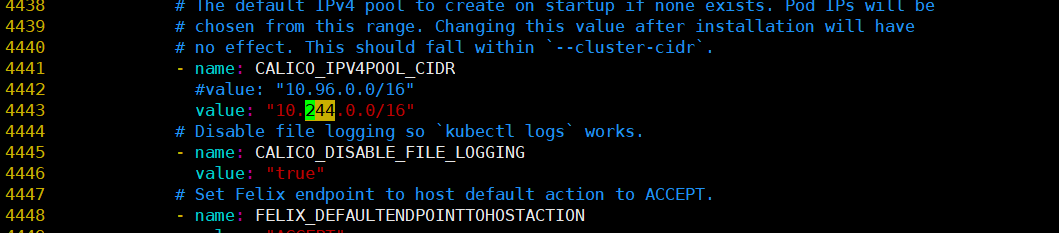
**方式3 vxlan模式**

**3.1 普通模式**

##### 3.1.1.配置PodIP范围（PodCIDR）

**- name: CALICO\_IPV4POOL\_CIDR**

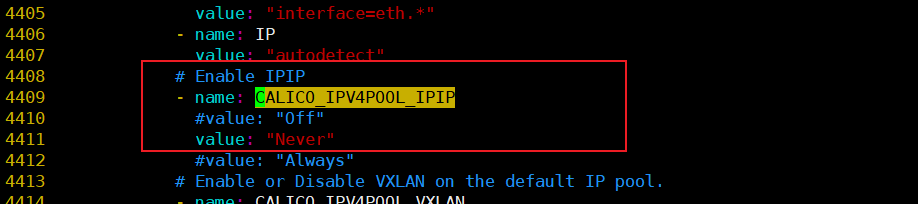
PodIP的范围应该与kubeadm init的清单文件中的"podSubnet"字段或者"--pod-network-cidr"选项填写的值相同

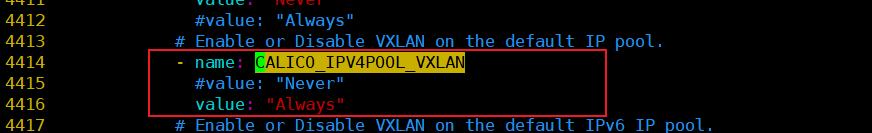


##### **3.1.2切换IPIP为VXLAN**

默认情况下，Calico清单启用IPIP封装。如果你期望Calico使用VXLAN封装模式，则需要在安装时候做以下操作

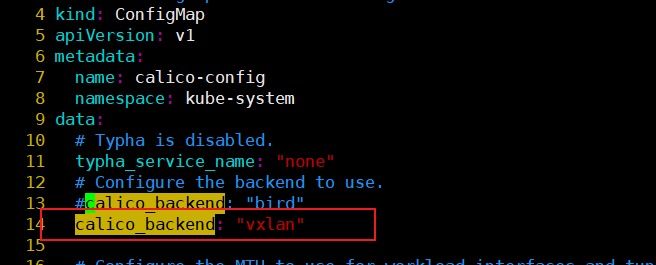
1. 将"CALICO\_IPV4POOL\_IPIP "设置为"Never"，或者“Off”
2. 将"CALICO\_IPV4POOL\_VXLAN"设置为"Always".

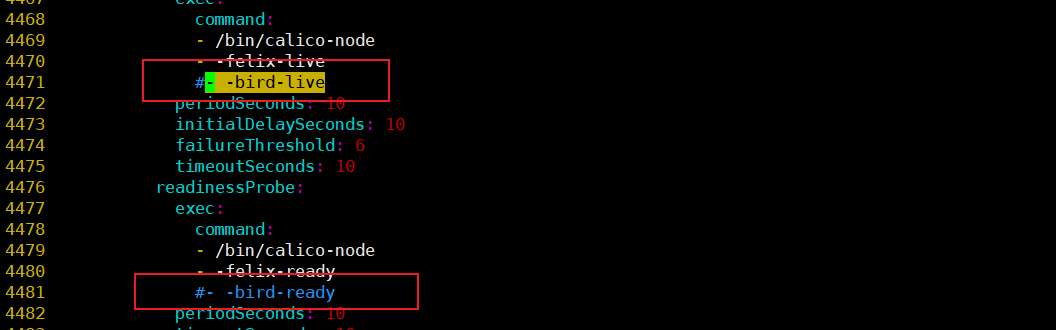




2、如果你只想集群仅使用基于VXLAN协议的覆盖网络模式，用不到BGP动态路由模式的话，即为了节省一点资源，可以选择完全禁用Calico基于BGP的网络：

* 将"calico\_backend: "bird""修改为"calico\_backend: "vxlan""，这将禁用BIRD。
* 从calico/node的readiness/liveness检查中去掉"- -bird-ready"和"- -bird-live"。



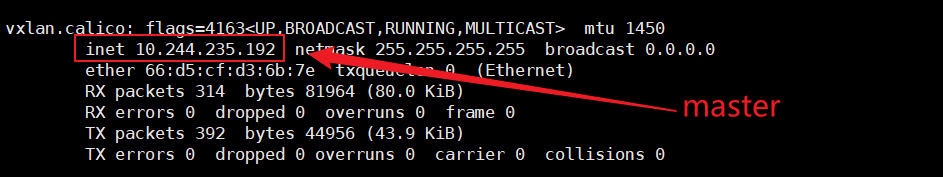


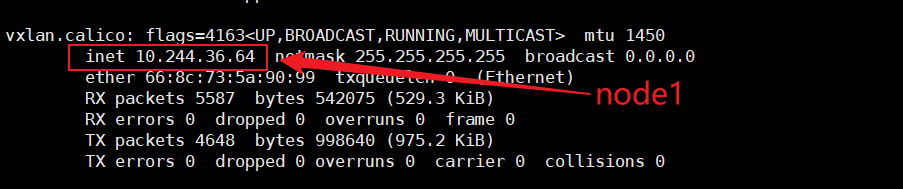
##### **3.1.3 applys**

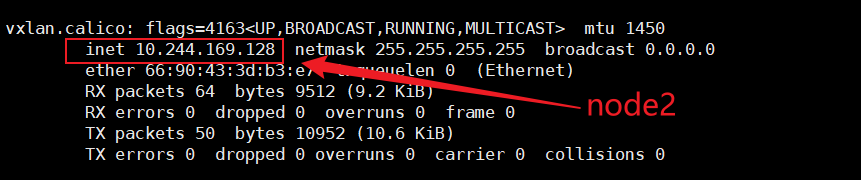
k apply -f calico-vxlan.yaml

##### **3.1.4 阿里云部署calico网络不通**

#ifconfig







**方式4 etcd**

curl https://projectcalico.docs.tigera.io/manifests/calico-etcd.yaml -O

根据方式3修改内容

修改设置Calico连接etcd，使用etcd的证书

单个etcd

|  |
| --- |
| ****ETCD\_ENDPOINTS="https://172.28.10.18:2379"****  ****sed -i "s#.\*etcd\_endpoints:.\*# etcd\_endpoints: \"${ETCD\_ENDPOINTS}\"#g" calico.yaml****  ****sed -i "s#\_\_ETCD\_ENDPOINTS\_\_#${ETCD\_ENDPOINTS}#g" calico.yaml**** |

集群etcd

|  |
| --- |
| **sed -i 's#etcd\_endpoints: "http://<ETCD\_IP>:<ETCD\_PORT>"#etcd\_endpoints: "https://10.10.7.48:2379,https://10.10.7.49:2379,https://10.10.7.50:2379"#g' calico.yaml** |

设置etcd证书路径。

|  |
| --- |
| ****ETCD\_CA=`cat /etc/kubernetes/pki/etcd/ca.crt | base64 | tr -d '\n'`****  ****ETCD\_CERT=`cat /etc/kubernetes/pki/etcd/server.crt | base64 | tr -d '\n'`****  ****ETCD\_KEY=`cat /etc/kubernetes/pki/etcd/server.key | base64 | tr -d '\n'`**** |

替换修改etcd证书路径。

方式一

|  |
| --- |
| sed -i "s#.\*etcd-ca:.\*# etcd-ca: ${ETCD\_CA}#g" calico.yaml  sed -i "s#.\*etcd-cert:.\*# etcd-cert: ${ETCD\_CERT}#g" calico.yaml  sed -i "s#.\*etcd-key:.\*# etcd-key: ${ETCD\_KEY}#g" calico.yaml  sed -i 's#.\*etcd\_ca:.\*# etcd\_ca: "/calico-secrets/etcd-ca"#g' calico.yaml  sed -i 's#.\*etcd\_cert:.\*# etcd\_cert: "/calico-secrets/etcd-cert"#g' calico.yaml  sed -i 's#.\*etcd\_key:.\*# etcd\_key: "/calico-secrets/etcd-key"#g' calico.yaml  sed -i "s#\_\_ETCD\_CA\_CERT\_FILE\_\_#/etc/kubernetes/pki/etcd/ca.crt#g" calico.yaml  sed -i "s#\_\_ETCD\_CERT\_FILE\_\_#/etc/kubernetes/pki/etcd/server.crt#g" calico.yaml  sed -i "s#\_\_ETCD\_KEY\_FILE\_\_#/etc/kubernetes/pki/etcd/server.key#g" calico.yaml  sed -i "s#\_\_KUBECONFIG\_FILEPATH\_\_#/etc/cni/net.d/calico-kubeconfig#g" calico.yaml |

方式二

|  |
| --- |
| **替换etcd中的证书base64编码后的内容**  sed -i "s@# etcd-key: null@etcd-key: ${ETCD\_KEY}@g; s@# etcd-cert: null@etcd-cert: ${ETCD\_CERT}@g; s@# etcd-ca: null@etcd-ca: ${ETCD\_CA}@g" calico.yaml  **#打开 etcd\_ca 等默认设置（calico启动后自己生成）**  sed -i 's#etcd\_ca: ""#etcd\_ca: "/calico-secrets/etcd-ca"#g; s#etcd\_cert: ""#etcd\_cert: "/calico-secrets/etcd-cert"#g; s#etcd\_key: "" #etcd\_key: "/calico-secrets/etcd-key" #g' calico.yaml |

**# 修改自己的Pod网段 196.16.0.0/16**

POD\_SUBNET="196.16.0.0/16"

sed -i 's@# - name: CALICO\_IPV4POOL\_CIDR@- name: CALICO\_IPV4POOL\_CIDR@g; s@# value: "192.168.0.0/16"@ value: '"${POD\_SUBNET}"'@g' calico.yaml

**#确认calico是否修改好**

grep "CALICO\_IPV4POOL\_CIDR" calico.yaml -A 1

安装Calico

****kubectl apply -f calico-etcd.yaml****

将证书文件拷贝到Node节点

|  |
| --- |
| ****scp -r /etc/kubernetes/pki/etcd root@k8s-node1:/etc/kubernetes/pki/****  ****scp -r /etc/kubernetes/pki/etcd root@k8s-node2:/etc/kubernetes/pki/**** |

<https://support.huaweicloud.com/instg-kunpengcpfs/openmind_kunpengcalico_03_0004.html>

**测试用例**

集群部署完成之后我们在k8s集群中部署一个nginx测试一下是否能够正常工作。首先我们创建一个名为nginx-quic的命名空间（namespace），然后在这个命名空间内创建一个名为nginx-quic-deployment的deployment用来部署pod，最后再创建一个service用来暴露服务，这里我们先使用nodeport的方式暴露端口方便测试。

****vim nginx-quic.yaml****

|  |
| --- |
| --- apiVersion: v1 kind: Namespace metadata:  name: nginx-quic  ---  apiVersion: apps/v1 kind: Deployment metadata:  name: nginx-quic-deployment  namespace: nginx-quic spec:  selector:  matchLabels:  app: nginx-quic  replicas: 4  template:  metadata:  labels:  app: nginx-quic  spec:  containers:  - name: nginx-quic  image: tinychen777/nginx-quic:latest  imagePullPolicy: IfNotPresent  ports:  - containerPort: 80  ---  apiVersion: v1 kind: Service metadata:  name: nginx-quic-service  namespace: nginx-quic spec:  selector:  app: nginx-quic  ports:  - protocol: TCP  port: 8080 # match for service access port  targetPort: 80 # match for pod access port  nodePort: 30088 # match for external access port  type: NodePort |

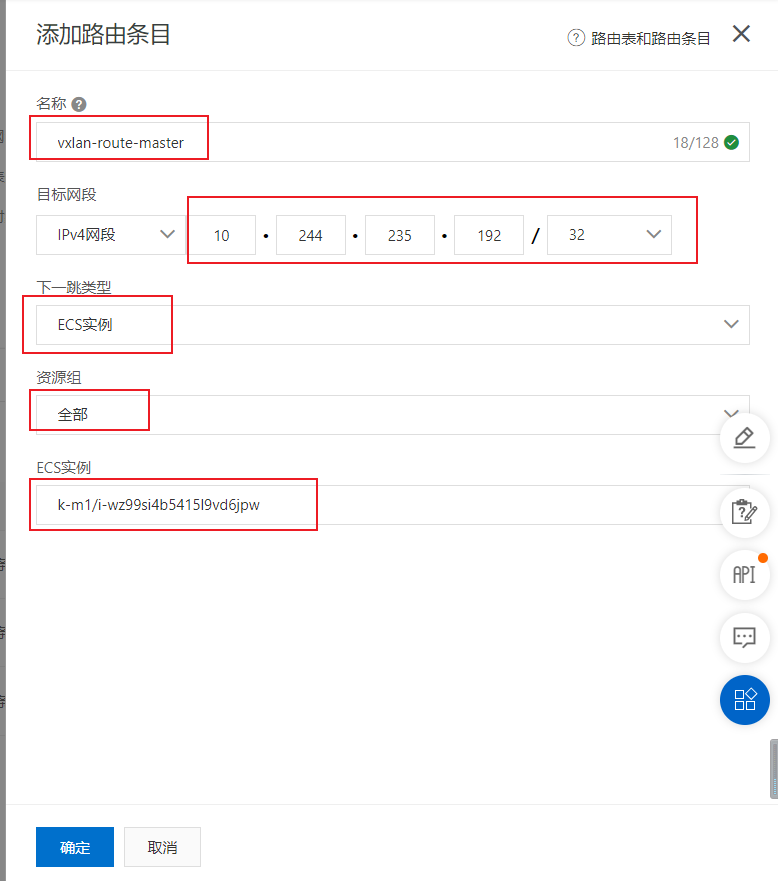
部署完成后我们直接查看状态

|  |
| --- |
| **# 直接部署** $ kubectl apply -f nginx-quic.yaml namespace/nginx-quic created deployment.apps/nginx-quic-deployment created service/nginx-quic-service created  **# 查看deployment的运行状态** $ kubectl get deployment -o wide -n nginx-quic  **# 查看service的运行状态** $ kubectl get service -o wide -n nginx-quic   **# 查看pod的运行状态** $ kubectl get pods -o wide -n nginx-quic    **# 查看IPVS规则** $ ipvsadm -ln IP Virtual Server version 1.2.1 (size=4096) Prot LocalAddress:Port Scheduler Flags  -> RemoteAddress:Port Forward Weight ActiveConn InActConn TCP 172.17.0.1:30088 rr  -> 10.88.84.68:80 Masq 1 0 0  -> 10.88.84.69:80 Masq 1 0 0  -> 10.88.120.71:80 Masq 1 0 0  -> 10.88.120.72:80 Masq 1 0 0 TCP 10.31.88.1:30088 rr  -> 10.88.84.68:80 Masq 1 0 0  -> 10.88.84.69:80 Masq 1 0 0  -> 10.88.120.71:80 Masq 1 0 0  -> 10.88.120.72:80 Masq 1 0 0 TCP 10.88.52.168:8080 rr  -> 10.88.84.68:80 Masq 1 0 0  -> 10.88.84.69:80 Masq 1 0 0  -> 10.88.120.71:80 Masq 1 0 0  -> 10.88.120.72:80 Masq 1 0 0 |

最后我们进行测试，这个nginx-quic的镜像默认情况下会返回在nginx容器中获得的用户请求的IP和端口

|  |
| --- |
| # 首先我们在集群内进行测试 # 直接访问pod $curl 192.168.36.66:80  $curl 192.168.169.130:80  # 直接访问service的ClusterIP，这时请求会被转发到pod中 $ curl 10.96.29.119:8080  # 直接访问nodeport，这时请求会被转发到pod中，不会经过ClusterIP $ curl 172.28.10.19:30088   # 接着我们在集群外进行测试 # 直接访问三个节点的nodeport，这时请求会被转发到pod中，不会经过ClusterIP # 由于externalTrafficPolicy默认为Cluster，因此nginx拿到的IP就是我们访问的节点的IP，而非客户端IP $ curl 39.108.249.154:30088  $ curl 120.78.8.189:30088  $ curl 120.76.218.103:30088 |

<https://tinychen.com/20220508-k8s-03-deploy-k8s-with-calico/#1%E3%80%81%E5%87%86%E5%A4%87%E5%B7%A5%E4%BD%9C>





<https://blog.csdn.net/qq_14962891/article/details/117223573>

**安装calicoctl**

<https://projectcalico.docs.tigera.io/getting-started/kubernetes/hardway/the-calico-datastore#calicoctl>

安装

将二进制文件下载calicoctl到可以访问 Kubernetes 的 Linux 主机。

|  |
| --- |
| #https://github.com/projectcalico/calico/releases/download/v3.23.1/calicoctl-linux-amd64  #chmod +x calicoctl-linux-amd64  #sudo mv calicoctl-linux-amd64 /usr/local/bin/calicoctl |

配置calicoctl访问 Kubernetes。（看情况）

|  |
| --- |
| export KUBECONFIG=/path/to/your/kubeconfig  export DATASTORE\_TYPE=kubernetes |

在大多数系统上，kubeconfig 位于~/.kube/config. 您可能希望将这些export行添加到您的行中，~/.bashrc以便在您下次登录时它们会持续存在。

测试

验证是否calicoctl可以通过运行访问您的数据存储

|  |
| --- |
| #calicoctl node status |

尝试获取由自定义资源支持的对象

|  |
| --- |
| #calicoctl get ippools --allow-version-mismatch |

参考文档

<https://blog.51cto.com/foxhound/2536925>