在谷歌云上自建K8s集群并使用VPC Native方式实现容器网络互通

Authors: eugeneyu@, hengwei@, yinghli@, jscheng@
Last Modified: 2021/08/13

概述

谷歌云使用先进的Andromeda网络来实现VPC内实例之间的相互访问，以及Google Kubernetes Engine (GKE) 的Pod的跨节点互访，避免了配置静态路由或者Overlay网络带来的运维复杂度以及性能瓶颈。使用谷歌云的VPC Native（使用IP Alias而非静态路由）方式来实现Pod间网络通讯的集群称作为VPC原生（VPC Native）集群。VPC原生集群具有以下优势。

- Pod IP 地址可在集群的 VPC 网络和通过 VPC 网络对等互连与之相连的其他 VPC 网络中进行原生路由。
- 在集群中创建 Pod 之前，Pod IP 会预留在 VPC 网络中。这可防止与 VPC 网络中其他资源发生冲突，让您更好地规划 IP 地址分配。
- Pod IP 地址范围不依赖于自定义静态路由。它们不会消耗系统生成的路由配额和自定义静态路由配额。自动生成的子网路由处理 VPC 原生集群的路由。
- 您可以创建仅应用于 Pod IP 地址范围而不是集群节点上任何 IP 地址的 防火墙规则。
- 通常，Pod IP 地址范围和子网次要 IP 地址范围可以通过 Cloud Router 与连接到 Cloud VPN 或 Cloud Interconnect 的本地网络访问。

这些先进的网络功能，也可以提供给用户在谷歌云虚机实例上自建的Kubernetes集群来使用。本文介绍了相关的配置和测试方法。

一 创建VPC

1 创建VPC

首先创建VPC：

```
gcloud compute networks create k8s-vpc \
    --subnet-mode=custom \
    --bgp-routing-mode=global \
    --mtu=1500
```

2 创建VPC防火墙规则

在VPC中添加防火墙规则：

```
gcloud compute firewall-rules create k8s-vpc-default-firewall \
    --network k8s-vpc --allow tcp:22,tcp:80,tcp:3389,icmp

gcloud compute firewall-rules create k8s-vpc-k8s-firewall \
    --network k8s-vpc \
    --allow tcp:6443,tcp:2379-2380,tcp:10250-10252,tcp:30000-32767
```

3 创建子网

创建子网asia-southeast1-sub-1。为其规划如下网段。

- Node 网段: 10.122.16.0/21
- Pod 网段: 192.168.16.0/21
- Service 网段: 172.16.16.0/21

```
gcloud compute networks subnets create asia-southeast1-sub-1 \
    --network=k8s-vpc \
    --range=10.122.16.0/21 \
    --region=asia-southeast1
```

给子网配置从属网段

```
gcloud compute networks subnets update asia-southeast1-sub-1 \
  --region=asia-southeast1 \
  --add-secondary-ranges=asia-southeast1-sub-1-pod=192.168.16.0/21
```

```
gcloud compute networks subnets update asia-southeast1-sub-1 \
  --region=asia-southeast1 \
  --add-secondary-ranges=asia-southeast1-sub-1-svc=172.16.16.0/21
```

二 创建虚拟机

1 创建K8s集群的Master和Node虚机。

每个Node会有一个由Master自动分配的Pod CIDR，需要根据这个网段来配置每个Node虚机实例的Alias IP Range, 本例分配的情况如下。

- Node-1
  - Pod CIDR: 192.168.17.0/24
  - Alias IP: 192.168.17.0/24
- Node-2
  - Pod CIDR: 192.168.18.0/24
  - Alias IP: 192.168.18.0/24

注意打开IP Forward并配置Alias IP网段给Pod使用。

A Master

注意Master需要绑定一个有权限创建负载均衡的Service Account，以及谷歌云API调用范围，否则，创建Kubernetes的Service时自动创建谷歌云负载均衡会失败。

```
gcloud compute instances create kub-m \
  --project=youzhi-lab \
  --zone=asia-southeast1-b \
  --machine-type=n2-standard-2 \
  --network-interface=network-tier=PREMIUM,subnet=asia-southeast1-sub-1 \
  --can-ip-forward \
  --maintenance-policy=MIGRATE \
  --image=centos-7-v20210721 \
  --image-project=centos-cloud \
  --boot-disk-size=100GB \
  --no-boot-disk-auto-delete \
  --boot-disk-type=pd-balanced \
  --boot-disk-device-name=kub-m \
  --no-shielded-secure-boot \
  --shielded-vtpm \
  --shielded-integrity-monitoring \
  --reservation-affinity=any \
  --service-account=247839977271-compute@developer.gserviceaccount.com \
  --scopes=https://www.googleapis.com/auth/cloud-platform
```

B Node-1

```
gcloud compute instances create kub-n-1 \
  --project=youzhi-lab \
  --zone=asia-southeast1-b \
  --machine-type=n2-standard-4 \
  --network-interface=network-tier=PREMIUM,subnet=asia-southeast1-sub-1 \
  --can-ip-forward \
  --maintenance-policy=MIGRATE \
  --image=centos-7-v20210721 \
  --image-project=centos-cloud \
  --boot-disk-size=100GB \
  --no-boot-disk-auto-delete \
  --boot-disk-type=pd-balanced \
  --boot-disk-device-name=kub-n-1 \
  --no-shielded-secure-boot \
  --shielded-vtpm \
  --shielded-integrity-monitoring \
  --reservation-affinity=any
```

C Node-2

```
gcloud compute instances create kub-n-2 \
  --project=youzhi-lab \
  --zone=asia-southeast1-b \
  --machine-type=n2-standard-4 \
  --network-interface=network-tier=PREMIUM,subnet=asia-southeast1-sub-1 \
  --can-ip-forward \
  --maintenance-policy=MIGRATE \
  --image=centos-7-v20210721 \
  --image-project=centos-cloud \
  --boot-disk-size=100GB \
  --no-boot-disk-auto-delete \
  --boot-disk-type=pd-balanced \
  --boot-disk-device-name=kub-n-2 \
  --no-shielded-secure-boot \
  --shielded-vtpm \
  --shielded-integrity-monitoring \
  --reservation-affinity=any
```

2 安装Docker

在所有的VM上安装Docker：

```
sudo yum install -y yum-utils
sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
sudo docker run hello-world
sudo mkdir /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

3 安装kubeadm及其工具

系统设置，以及工具安装：

```
sudo systemctl stop firewalld.service
sudo systemctl disable firewalld.service
sudo modprobe br_netfilter

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
```

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF

# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

sudo systemctl enable --now kubelet
```

配置kubeadm初始化参数

```
cat <<EOF | sudo tee sudo /proc/sys/net/ipv4/ip_forward
1
EOF
```

4 关闭IP Alias的local route table

在GCE中，GCE agent的network Daemon会监控IP Alias地址，并添加相应的路由。但在这种情况下，会造成container通讯的故障。通过下面的命令在Master和各个Node系统内修改GCE agent配置，并重启google guest agent服务。目的是Pod通过Alias IP地址互访时，不通过虚机实例的Nat，而访问其他地址时，采用NAT的方式。

```
sudo sed -i 's/ip\_aliases\ \=\ true/ip\_aliases\ \=\ false/' \
   /etc/default/instance_configs.cfg

sudo systemctl restart google-guest-agent
iptables -P FORWARD ACCEPT
```

```
ip route show table local  #查看本地路由表，确保alias IP段没有 在eth0上
```

三 通过kubeadm安装kubernetes

1 安装配置Master节点

A 配置Master节点

```
export name="$(hostname)"
export pod_cidr="192.168.16.0/21"
export service_cidr="172.16.16.0/21"
cat <<EOF > /tmp/kubeadm-config.yaml
apiVersion: kubeadm.k8s.io/v1beta3
kind: InitConfiguration
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: youzhi.0123456789abcdef
nodeRegistration:
  name: $name
  kubeletExtraArgs:
    cloud-provider: gce
    network-plugin: kubenet
    non-masquerade-cidr: 0.0.0.0/0
---
apiVersion: kubeadm.k8s.io/v1beta3
kind: ClusterConfiguration
networking:
  podSubnet: ${pod_cidr}
  serviceSubnet: ${service_cidr}
apiServer:
  extraArgs:
    enable-admission-plugins: DefaultStorageClass,NodeRestriction
    cloud-provider: gce
controllerManager:
  extraArgs:
    cloud-provider: gce
    configure-cloud-routes: "false"
    address: 0.0.0.0
```

```
EOF
```

B 通过配置文件安装

```
sudo kubeadm init \
  --config=/tmp/kubeadm-config.yaml
```

如果执行成功，会有以下输出。

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular
user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on
each as root:

sudo kubeadm join 10.122.16.10:6443 --token youzhi.0123456789abcdef \
  --discovery-token-ca-cert-hash
sha256:e5c96b2d0499287b6884c27b8ec7293e6aab1ab09540a0386b91ecbadfb38d4f
```

C 配置kubectl配置文件

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

D 安装ip-masq-agent

默认配置下， Pod-Pod访问时会进行SNAT。使用Alias IP方式直接路由方式时，可以通过部署ip-masq-agent的方式关闭SNAT。

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes-sigs/ip-masq-agent/master/ip-m
asq-agent.yaml
```

2 安装配置Node

A Join

在两个Node节点上运行join命令：

```
sudo kubeadm join 10.122.16.10:6443 --token youzhi.0123456789abcdef \
  --discovery-token-ca-cert-hash
sha256:e5c96b2d0499287b6884c27b8ec7293e6aab1ab09540a0386b91ecbadfb38d4f
```

B 查看Node状态

此时Node是NotReady状态，因为kubelet的网络插件配置有问题。

```
$ kubectl get node
NAME       STATUS     ROLES                  AGE    VERSION
kub-m      Ready      control-plane,master   137m   v1.22.0
kub-n-1    NotReady   <none>                 15m    v1.22.0
```

C 修改Node上kubelet配置

```
sudo vim /var/lib/kubelet/kubeadm-flags.env

KUBELET_KUBEADM_ARGS="--cloud-provider=gce --network-plugin=kubenet
--pod-infra-container-image=k8s.gcr.io/pause:3.5"

sudo systemctl restart kubelet
```

这时node的状态变成Ready：

```
$ kubectl get node
NAME       STATUS   ROLES                  AGE    VERSION
kub-m      Ready    control-plane,master   147m   v1.22.0
kub-n-1    Ready    <none>                 25m    v1.22.0
```

D 修改Node providerID

为了Service能自动创建和配置谷歌云负载均衡，需要配置每个Node的providerID参数。对于谷歌云上的Node, providerID的格式是 gce://<Project ID>/<Zone>/<Instance Name>。

在Master节点，运行kubectl修改每个Node的prividerID：

```
$ kubectl patch node kub-n-1 \
  -p '{"spec":{"providerID":"gce://youzhi-lab/asia-southeast1-b/kub-n-1"}}'
node/kub-n-1 patched
$ kubectl patch node kub-n-2 \
  -p '{"spec":{"providerID":"gce://youzhi-lab/asia-southeast1-b/kub-n-2"}}'
node/kub-n-2 patched
```

E 给GCE添加Network Tag

在Kubernetes通过cloud-provider创建实现Service的谷歌云负载均衡的时候，需要通过network tag创建firewall rules。所以需要给GCE实例添加network tag。创建的规则是所有node相同的前缀, 比如这里两台VM相同的前缀是kub-n

```
gcloud compute instances add-tags kub-n-1 \
  --zone asia-southeast1-b --tags kub-n
gcloud compute instances add-tags kub-n-2 \
  --zone asia-southeast1-b --tags kub-n
```

F 给Node虚机实例添加Alias IP

kubectl get nodes 列出各个Node的名称。

查看每个Node的Pod CIDR，如下。

```
$ kubectl get node kub-n-1 -o jsonpath={.spec.podCIDR}
192.168.18.0/24
```

使用谷歌云命令行工具，根据上面的打印结果，分别更新各个Node的Alias IP，使其与该Node的Pod CIDR一致。

- Node-1

```
gcloud compute instances network-interfaces update kub-n-1 \
    --zone asia-southeast1-b \
    --aliases "asia-southeast1-sub-1-pod:192.168.17.0/24"
```

- Node-2

```
gcloud compute instances network-interfaces update kub-n-2 \
    --zone asia-southeast1-b \
    --aliases "asia-southeast1-sub-1-pod:192.168.18.0/24"
```

3 检测配置

A 创建deployment和service

```
kubectl create deploy nginx --image nginx
kubectl scale deploy/nginx --replicas=6
kubectl expose deployment nginx \
  --port 80 --target-port 80 --name nginx-lb --type LoadBalancer
```

B 查看部署情况

查看pod情况：

```
$ kubectl get pod -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP             NODE
nginx-6799fc88d8-68fld   1/1     Running   0          17h   192.168.18.7   kub-n-2
nginx-6799fc88d8-gssw5   1/1     Running   0          17h   192.168.17.15  kub-n-1
nginx-6799fc88d8-jcdhc   1/1     Running   0          17h   192.168.17.14  kub-n-1
nginx-6799fc88d8-k7sxg   1/1     Running   0          17h   192.168.18.6   kub-n-2
nginx-6799fc88d8-l6nnv   1/1     Running   0          17h   192.168.18.5   kub-n-2
nginx-6799fc88d8-tcbmf   1/1     Running   0          17h   192.168.17.13  kub-n-1
```

查看Service：

```
$ kubectl get svc
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
kubernetes   ClusterIP      172.16.16.1     <none>          443/TCP        29h
nginx-lb     LoadBalancer   172.16.16.153   34.126.129.75   80:32761/TCP   7m31s
```

获取Service的LB IP

```
ip=$(kubectl get svc -l app=nginx \
  -o jsonpath={.items[0].status.loadBalancer.ingress[0].ip})
```

测试Pod的同Node和跨Node访问

```
$ kubectl exec -it nginx-6799fc88d8-68fld bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future
version. Use kubectl exec [POD] -- [COMMAND] instead.
root@nginx-6799fc88d8-68fld:/# curl -I 192.168.18.6
HTTP/1.1 200 OK
Server: nginx/1.21.1
Date: Thu, 12 Aug 2021 07:25:29 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
Connection: keep-alive
ETag: "60e46fc5-264"
Accept-Ranges: bytes

root@nginx-6799fc88d8-68fld:/# curl -I 192.168.17.15
HTTP/1.1 200 OK
Server: nginx/1.21.1
Date: Thu, 12 Aug 2021 07:25:35 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
Connection: keep-alive
ETag: "60e46fc5-264"
Accept-Ranges: bytes
```

测试通过Cluser IP访问Service

```
root@nginx-6799fc88d8-68fld:/# curl -I 172.16.16.153
HTTP/1.1 200 OK
Server: nginx/1.21.1
```

```
Date: Thu, 12 Aug 2021 07:25:43 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
Connection: keep-alive
ETag: "60e46fc5-264"
Accept-Ranges: bytes
```

测试Service从公网访问：

```
$ curl -I $ip
HTTP/1.1 200 OK
Server: nginx/1.21.1
Date: Wed, 11 Aug 2021 14:25:06 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
Connection: keep-alive
ETag: "60e46fc5-264"
Accept-Ranges: bytes
```

C 测试从其它子网访问Pod

现在集群上的Pod除了可以同一子网内互相访问，也可以让VPC下其它子网（同一区域或者不同区域都可以）以及通过专线或VPN与本VPC互通的线下机房网络里的虚机来访问。下面在不同区域创建一个新子网，以及该子网下的虚机，来访问集群上的Pod。

```
gcloud compute networks subnets create us-central1-sub-1 \
    --network=k8s-vpc \
    --range=10.150.16.0/21 \
    --region=us-central1

gcloud compute instances create remote-test-1 --project=youzhi-lab
--zone=us-central1-b --machine-type=n2-standard-4
--network-interface=network-tier=PREMIUM,subnet=us-central1-sub-1
--can-ip-forward --maintenance-policy=MIGRATE --image=centos-7-v20210721
--image-project=centos-cloud --boot-disk-size=100GB
--no-boot-disk-auto-delete --boot-disk-type=pd-balanced
--boot-disk-device-name=remote-test-1 --no-shielded-secure-boot
```

```
--shielded-vtpm --shielded-integrity-monitoring --reservation-affinity=any
```

登录remote-test1虚机，并运行以下命令访问Pod。

```
$ curl -I 192.168.18.7
HTTP/1.1 200 OK
Server: nginx/1.21.1
Date: Fri, 13 Aug 2021 05:35:46 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
Connection: keep-alive
ETag: "60e46fc5-264"
Accept-Ranges: bytes

$ curl -I 192.168.17.14
HTTP/1.1 200 OK
Server: nginx/1.21.1
Date: Fri, 13 Aug 2021 05:36:00 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 06 Jul 2021 14:59:17 GMT
Connection: keep-alive
ETag: "60e46fc5-264"
Accept-Ranges: bytes
```

可以确认访问成功。

D 确认Node上关闭Pod访问SNAT

如果检测结果与下面的结果有差异，确认执行Master节点配置的“安装ip-masq-agent”一节。

```
$ kubectl get ds -n kube-system
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
AGE
ip-masq-agent   2         2         2       2            2           <none>
19m
kube-proxy      3         3         3       3            3
kubernetes.io/os=linux   29h
```

在每台Node上查看NAT, IP-MASQ-AGENT Chain默认会针对目的地址为RFC1918 地址段不做地址翻译。

```
$ sudo iptables -t nat -L -n --line-numbers

Chain POSTROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    KUBE-POSTROUTING  all  --  0.0.0.0/0            0.0.0.0/0
/* kubernetes postrouting rules */
2    IP-MASQ-AGENT  all  --  0.0.0.0/0            0.0.0.0/0            /*
ip-masq-agent: ensure nat POSTROUTING directs all non-LOCAL destination
traffic t
o our custom IP-MASQ-AGENT chain */ ADDRTYPE match dst-type !LOCAL

Chain IP-MASQ-AGENT (1 references)
num  target     prot opt source               destination
1    RETURN     all  --  0.0.0.0/0            169.254.0.0/16       /*
ip-masq-agent: cluster-local traffic should not be subject to MASQUERADE */
ADDRTYPE
match dst-type !LOCAL
2    RETURN     all  --  0.0.0.0/0            10.0.0.0/8           /*
ip-masq-agent: cluster-local traffic should not be subject to MASQUERADE */
ADDRTYPE
match dst-type !LOCAL
3    RETURN     all  --  0.0.0.0/0            172.16.0.0/12        /*
ip-masq-agent: cluster-local traffic should not be subject to MASQUERADE */
ADDRTYPE
match dst-type !LOCAL
4    RETURN     all  --  0.0.0.0/0            192.168.0.0/16       /*
ip-masq-agent: cluster-local traffic should not be subject to MASQUERADE */
ADDRTYPE
match dst-type !LOCAL
5    MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0            /*
ip-masq-agent: outbound traffic should be subject to MASQUERADE (this match
must com
e after cluster-local CIDR matches) */ ADDRTYPE match dst-type !LOCAL
```

四 总结

在GCE上通过kubeadm可以非常方便的安装kubernetes集群，通过GCP VPC的各种网络功能，配合Kubernetes本身cloud-provider的功能，可以非常方便的实现Kubernetes网络的各种功能。

Appendix

1 kubeadm创建token

```
kubeadm token create --ttl=0 --print-join-command
kubeadm join 10.128.1.96:6443 \
  --token xxxx \
  --discovery-token-ca-cert-hash sha256:xxxx
```

2 alias IP模式下的配置

确认删除Alias IP的本地路由：
https://cloud.google.com/vpc/docs/configure-alias-ip-ranges#enabling_ip_alias_on_images_disables_cbr0_bridge_on_self-managed_kubernetes_clusters

通过GCP Alias IP实现跨Node的Pod互通

3 K8s网络针对GCP的配置注意事项

https://kubernetes.io/docs/concepts/cluster-administration/networking/#google-compute-engine-gce

4 Node, Pod和Service网段配置参考表

https://cloud.google.com/kubernetes-engine/docs/concepts/alias-ips#cluster_sizing_secondary_range_pods

5 Kubadmin安装步骤

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Kubeadmin init参数

https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm-init/

ip-masq-agent部署

https://kubernetes.io/docs/tasks/administer-cluster/ip-masq-agent/