文档名称：ZDNS-Zcloud-kubernetes CNI-Flannel 部署
文档分类：ZDNS-Zcloud-kubernetes 文档
服务等级：
文档编号：
作　　者: 余春云
联系电话：18811483030
电子邮件：yuchunyun@zdns.cn

Kubernetes CNI-Flannel 部署

文档变更历史记录：

变更日期	变更人	变更内容摘要	组长确认
2019-03-19	余春云	初次建立文档	

目录

https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

前提

1：Kubelet 开启 CNI 接口

"--cni-bin-dir=/opt/cni/bin",
"--cni-conf-dir=/etc/cni/net.d"

2：Kube-Controller-Manager 指定集群 Pod 网络并开启分配

“--cluster-cidr=10.42.0.0/16”
“--allocate-node-cidrs=true”

配置部署

RBAC 配置

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: flannel
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: flannel
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: flannel
subjects:
- kind: ServiceAccount
  name: flannel
  namespace: kube-system
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
```

```
  name: flannel
rules:
  - apiGroups:
      - ""
    resources:
      - pods
    verbs:
      - get
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - nodes/status
    verbs:
      - patch
---
```

ConfigMap 配置

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: kube-flannel-cfg
  namespace: "kube-system"
  labels:
    tier: node
    app: flannel
data:
  cni-conf.json: |
    {
      "name":"cbr0",
      "cniVersion":"0.3.1",
      "ipMasq": true,
      "plugins":[
        {
          "type":"flannel",
          "delegate":{
            "forceAddress":true,
```

```
                "bridge": "mycni8",                #控制主机上桥接网卡的名称
                "isDefaultGateway":true
            }
        },
        {
            "type":"portmap",
            "capabilities":{
                "portMappings":true
            }
        }
    ]
  }
  net-conf.json: |
    {
      "Network": "10.42.0.0/16",                #集群 pod 网络断，需要与 kube-controller-manager 的 cluster-cidr 保持一致
      "SubnetLen": 24,
      "Backend": {
        "Type": "vxlan",
        "Directrouting": true
      }
    }
```

DaemonSet 配置

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: kube-flannel
  namespace: "kube-system"
  labels:
    tier: node
    k8s-app: flannel
spec:
  template:
    metadata:
      labels:
        tier: node
        k8s-app: flannel
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
```

```
                    - matchExpressions:
                      - key: beta.kubernetes.io/os
                        operator: NotIn
                        values:
                          - windows
      serviceAccountName: flannel
      containers:
      - name: kube-flannel
        image: rancher/coreos-flannel:v0.10.0
        imagePullPolicy: IfNotPresent
        resources:
          limits:
            cpu: 300m
            memory: 500M
          requests:
            cpu: 150m
            memory: 64M
        command:                ["/opt/bin/flanneld","--ip-masq","--kube-subnet-mgr","--
iface=ens3"]
        securityContext:
          privileged: true
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        volumeMounts:
        - name: run
          mountPath: /run
        - name: cni
          mountPath: /etc/cni/net.d
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      - name: install-cni
        image: rancher/coreos-flannel-cni:v0.3.0
        command: ["/install-cni.sh"]
        env:
        # The CNI network config to install on each node.
        - name: CNI_NETWORK_CONFIG
          valueFrom:
```

```
                    configMapKeyRef:
                      name: kube-flannel-cfg
                      key: cni-conf.json
                - name: CNI_CONF_NAME
                  value: "10-flannel.conflist"
                volumeMounts:
                - name: cni
                  mountPath: /host/etc/cni/net.d
                - name: host-cni-bin
                  mountPath: /host/opt/cni/bin/
            hostNetwork: true
            tolerations:
            - operator: Exists
              effect: NoSchedule
            - operator: Exists
              effect: NoExecute
            volumes:
              - name: run
                hostPath:
                  path: /run
              - name: cni
                hostPath:
                  path: /etc/cni/net.d
              - name: flannel-cfg
                configMap:
                  name: kube-flannel-cfg
              - name: host-cni-bin
                hostPath:
                  path: /opt/cni/bin
    updateStrategy:
      rollingUpdate:
        maxUnavailable: 20%
      type: RollingUpdate
```

说明

DaemonSet 控制的 pod 中包含了两个容器

install-cni

用途

1） 将容器中/opt/cni/bin/目录下的 CNI 插件（二进制文件）拷贝到主机/opt/cni/bin/目录下
2） 将 ConfigMap 中 cni-conf.json 内容写入到主机/etc/cni/net.d/目录下

目的

是为了给 kubelet 提供 CNI 插件和配置，从而使得 kubelet 在创建 pod 的时候可以遵照标准的 CNI 流程给 pod 分配 IP 地址

kube-flannel

用途

主要负责启动 flanneld，监听 kube-api 的 subnet 事件

目的

根据 subnet 事件，增加或删除路由表/Arp 表/Fdb 表，将 pod 间网络打通

注意

由于 flanneld 需要修改主机路由表等信息，因此该容器需要运行在特权模式，且需要使用主机的网络空间

```
hostNetwork: true
  securityContext:
  privileged: true
```

Flanneld 启动命令参数

--kube-subnet-mgr：指定使用 kube 类型的 subnet-manager
--ip-masq：指定需要为 pod 配置 SNAT，以便出访外网

--iface=ens3：指定各主机间通信的接口

ConfigMap 中的参数

cni-conf.json

"ipMasq": true

为 pod 配置 SNAT，使其能够访问集群外部（默认为 false）

“plugins.delegate.bridge”: “mycni8”

Pod 的桥接网卡名称（默认为 cni0）

net-conf.json

Network：flannel 管理的网络段（需要与 podCidr 一致）

SubnetLen：子网分配时使用的掩码（注：使用 kube-subnet-manager 时该项没有意义）

SubnetMin：子网分配时起始网段（注：使用 kube-subnet-manager 时该项没有意义）

SubnetMax：子网分配时终止网段（注：使用 kube-subnet-manager 时该项没有意义）

Backend：后端类型

Backend：后端类型

vxlan 模式

推荐使用 vxlan，利用 linux kernel 的 vxlan 实现。参数：

Type: string，backend 类型，就是"vxlan"

VNI: number，vxlan 协议中的 vni 编号，不同的 vni 号码代表不同网段，类似 vlan 号，默认 1

Port: number, 宿主机的 udp 端口，用来发送封装后的报文，使用 linux 内核默认配置，8472

GBP: boolean，是否使用 vxlan Group Policy，默认 false

DirectRouting: boolean，是否启用直接路由，当两台宿主机位于同一个网段时，不封装通过路由直接送达，默认 false 。（建议开启）

GBP 特性参考：vxlan: Group Policy extension。

host-gw 模式

通过直接路由的方式传送虚拟网络报文。Host-gw 原理和 vxlan 中的 DirectRouting 相同，但是要求所有宿主机都支持直接路由方式（即在同一个二层网络中），并全部采用采用直接路由的方式。参数只有一个

Type: string，backend 类型，就是"host-gw"

Udp 模式

不建议使用，参数如下：
Type：string，backend 类型，就是"udp"
Port: number，宿主机 udp 端口，默认 8285
注：vxlan 和 UDP 的区别是 vxlan 是内核封包，而 UDP 是 flanneld 用户态程序封包，所以 UDP 的方式性能会稍差。

aws-vpc、gce、ali-vpc

是针对公有云环境的