Linux
eBPF

Standard
Linux

Windows
HNS

Features

- *Multiple dataplane support: Standard Linux dataplane and ebpf. HNS for windows*
- *Native kubernetes network policy*
- *Choice of overlay (ipipMode or Vxlan) or non-overlay connectivity option*
- *Has in-built IPAM*
- *IPv4/IPv6 Support (IPv6 is supported in non-overlay mode).*
- *Different IP Pools within a cluster depending on the requirement*
- *SNAT pod IP to Node IP support for outbound connections (known as natOutgoing in calico)*
- *BGP support (non-overlay and IPnIP mode)*

Overlay Mode

IPinIP CrossSubnet Mode

```
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: ippool-ipip-cross-subnet-1
spec:
  cidr: 192.168.0.0/16
  ipipMode: CrossSubnet
  natOutgoing: true
```

Full IPinIP Mode

```
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: ippool-ipip-1
spec:
  cidr: 192.168.0.0/16
  ipipMode: Always
  natOutgoing: true
```

Overlay Mode

VxLAN CrossSubnet Mode

```
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: ippool-vxlan-cross-subnet-1
spec:
  cidr: 192.168.0.0/16
  vxlanMode: CrossSubnet
  natOutgoing: true
```

Full VxLAN Mode

```
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: ippool-vxlan-1
spec:
  cidr: 192.168.0.0/16
  vxlanMode: Always
  natOutgoing: true
```

Non-Overlay Mode

```
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: ippool-vxlan-cross-subnet-1
spec:
  cidr: 192.168.0.0/16
  natOutgoing: true
```

Calico Client Binary

Install calicoctl

curl -O -L https://github.com/projectcalico/calicoctl/releases/download/v3.17.1/calicoctl

chmod +x calicoctl

mv calicoctl /usr/local/bin/

Set environment variable

vi /etc/profile

export DATASTORE_TYPE=kubernetes
export KUBECONFIG=~/.kube/config

Sample commands

calicoctl get ipPool

calicoctl get workloadendpoints

calicoctl ipam show

LAB_1

Create A New IPPool And Use It In A Deployment

```
cat > pool1.yaml <<EOF
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: pool1
spec:
  cidr: 10.168.0.0/18
  ipipMode: Never
  natOutgoing: true
  disabled: false
  nodeSelector: all()
EOF
```

Command to apply the config:

```
calicoctl create -f pool1.yaml
```

Check if pool was created:

```
calicoctl get ippool
```

LAB_2

Use Specific IP Pool In A Deployment
Create A Deployment Manifest File

kubectl create deployment calico-ippool --image=nginx:stable-alpine --dry-run=client -o yaml > ippool-deployment.yaml

Open ippool-deployment.yaml and add the following annotations to the POD section and create it.

cni.projectcalico.org/ipv4pools: "[\"pool1\"]"

Use Specific/Static IP In A Deployment
Create A Deployment Manifest File

kubectl create deployment calico-static --image=nginx:stable-alpine --dry-run=client -o yaml > static-deployment.yaml

Open static-deployment.yaml and add the following annotations to the POD section and create it.

"cni.projectcalico.org/ipAddrs": "[\"10.168.0.1\"]"

LAB_3

Assign IP Pool To A Specific Worker Node And Schedule A Pod To That Node

Label k8s-worker-1 and k8s-worker-2

```
kubectl label node k8s-worker-1 rack=0

kubectl label node k8s-worker-2 rack=1
```

Create IP pool for worker-1

```
calicoctl create -f -<<EOF
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: rack-0-ippool
spec:
  cidr: 10.168.0.0/18
  ipipMode: Never
  natOutgoing: true
  nodeSelector: rack == "0"
EOF
```

LAB_3

Create IP pool for worker-2

```
calicoctl create -f -<<EOF
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: rack-1-ippool
spec:
  cidr: 10.168.64.0/18
  ipipMode: Never
  natOutgoing: true
  nodeSelector: rack == "1"
EOF
```

Create deployment: *kubectl create deploy node-pool-demo --image nginx:stable-alpine --replicas 5*

LAB_4

Assign IP Pool To A Specific NameSpace And Schedule A Pod To That Namespace

Create IP pool

```
calicoctl create -f -<<EOF
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: pool2
spec:
  cidr: 10.168.128.0/18
  ipipMode: Never
  natOutgoing: true
  nodeSelector: all()
EOF
```

#Create Namespace manifest

```
kubectl create ns test-ip-pool --dry-run=client -o yaml > test-ip-pool-namespace.yaml
```

LAB_4

Add following annotation to test-ip-pool-namespace.yaml file

cni.projectcalico.org/ipv4pools: "[\"pool2\"]"

Create deployment:

kubectl -n test-ip-pool create deploy ns-pool-demo --image nginx:stable-alpine --replicas 5

LAB_5

Configure Vyos As BGP Peer And Check If Routes Are Been Advertised

```
calicoctl create -f -<<EOF
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
  name: default
spec:
  asNumber: 63400
EOF

calicoctl create -f -<<EOF
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: vyos-gns3
spec:
  peerIP: 10.10.100.1
  asNumber: 63400
EOF
```

LAB_5

Configure Vyos As BGP Peer And Check If Routes Are Been Advertised

N.B - You can also start wireshark to see some of the traffic flow

Configure vyos:

set protocols bgp 63400 neighbor 10.10.100.3 remote-as '63400'

set protocols bgp 63400 neighbor 10.10.100.4 remote-as '63400'

Check IP route in the vyos router:

show ip route

show ip bgp neighbors

LAB_6

Advertise Service ClusterIP Via BGP

Create deployment:

kubectl create deploy svc-bgp-demo --image nginx:stable-alpine

Expose deployment via service:

kubectl expose deploy svc-bgp-demo --port 80

Test to see if service clusterIP can be accessed using the Alpine container, this will fail since service cidr has not been advertised to the vyos

curl service_cluster_ip

Retrieve the kubernetes service cidr i.e. ClusterIP subnet

kubectl get cm -o yaml -n kube-system kubeadm-config | grep serviceSubnet

LAB_6

Add service cidr to to the BGP configuration:

```
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
 name: default
Spec:
  asNumber: 63400
 serviceClusterIPs:
 - cidr: 10.96.0.0/12
```

Check routes in the vyos router:

```
show ip route
```

LAB_6

Access service from the Alpine container:

curl service_cluster_ip

Issue with advertising all the service cidr is that all kubernetes services are accessible, this can be mitigated by using security features of the vyos router to restrict access to specific cluserIP.

Another way is to use external IP, this will just be a small IP subnet which will only be allocated to services that needs to exposed outside the kubernetes cluster.

Add externalIP range in calico BGP config:

```
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
 name: default
Spec:
 asNumber: 63400
 serviceClusterIPs:
 serviceExternalIPs:
 - cidr: 172.16.0.0/24
```

LAB_6

Patch existing service to add external IP:

kubectl patch svc svc-bgp-demo -p '{"spec": {"externalIPs": ["172.16.0.1"]}}'

Check routes in vyos and verify connectivity from alpine container

curl service_cluster_ip

LAB_7

By default kubernetes services uses all worker nodes when there is need to communicate with a service even when a worker node does not have a pod replica that is running on it for that particular service. The feature that controls this feature is called *externalTrafficPolicy*. For this case it will be set to "*Cluster*" which is the default configuration.

The downside to this is that the source IP will not be visible in the application logs of the POD, also if a worker node does not have a pod replica of a particular service it will increase the latency since the traffic will have to pass through before getting to the worker node that hosts a replica of the application POD.

If the desired outcome is to have only worker nodes that are hosting a pod replica of the needed service then the service *externalTrafficPolicy* parameter needs to be changed to "*Local*".

To change the *externalTrafficPolicy* to Local, the kubernetes service needs to be edited/patched:

kubectl patch service svc-bgp-demo -p '{"spec":{"type": "NodePort", "externalTrafficPolicy":"Local"}}'

References

https://docs.projectcalico.org/about/about-kubernetes-networking

https://docs.projectcalico.org/networking/vxlan-ipip

https://docs.projectcalico.org/networking/workloads-outside-cluster

https://docs.projectcalico.org/getting-started/clis/calicoctl/install

https://docs.projectcalico.org/networking/ipam

https://docs.projectcalico.org/networking/bgp

https://docs.projectcalico.org/networking/advertise-service-ips

https://docs.projectcalico.org/reference/cni-plugin/configuration

https://docs.projectcalico.org/networking/assign-ip-addresses-topology