Kubernetes, the wild side: para nostálgicos y legacy

kubernetes

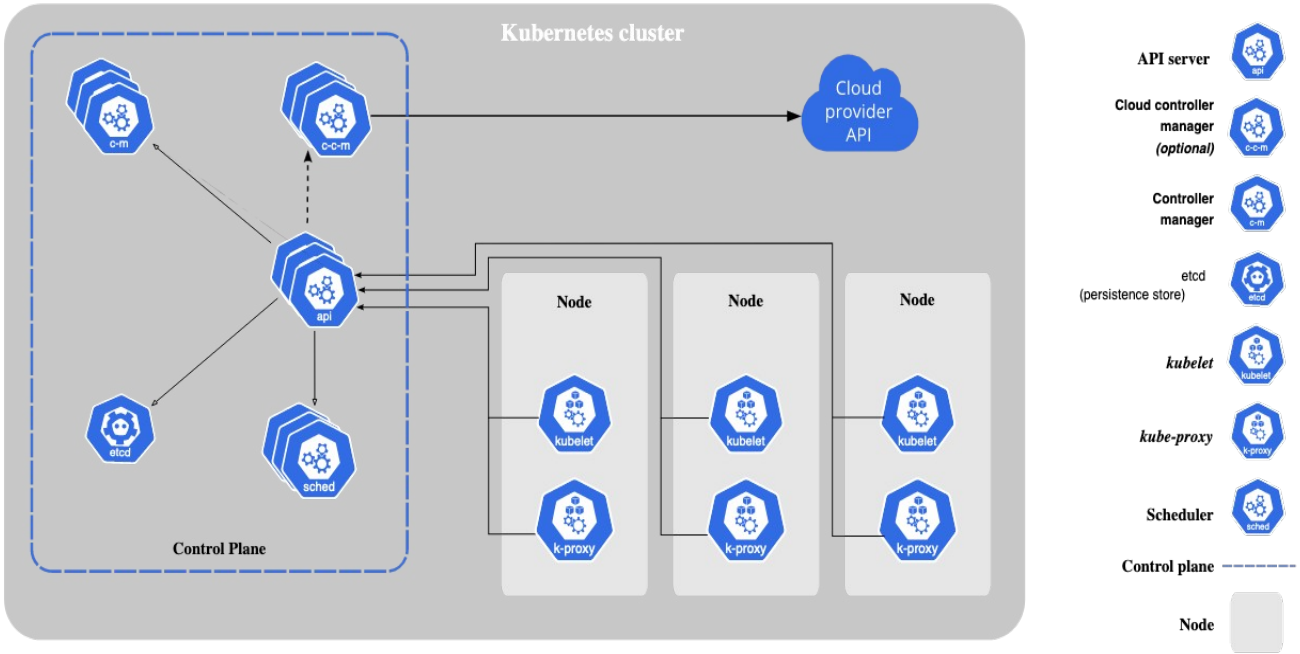
Agenda

1. A grandes rasgos ¿Qué es Kubernetes?
2. La red en kubernetes
3. Redes para nostalgicos
4. Calico Cni
5. Calico en un fabric

¿Por qué?

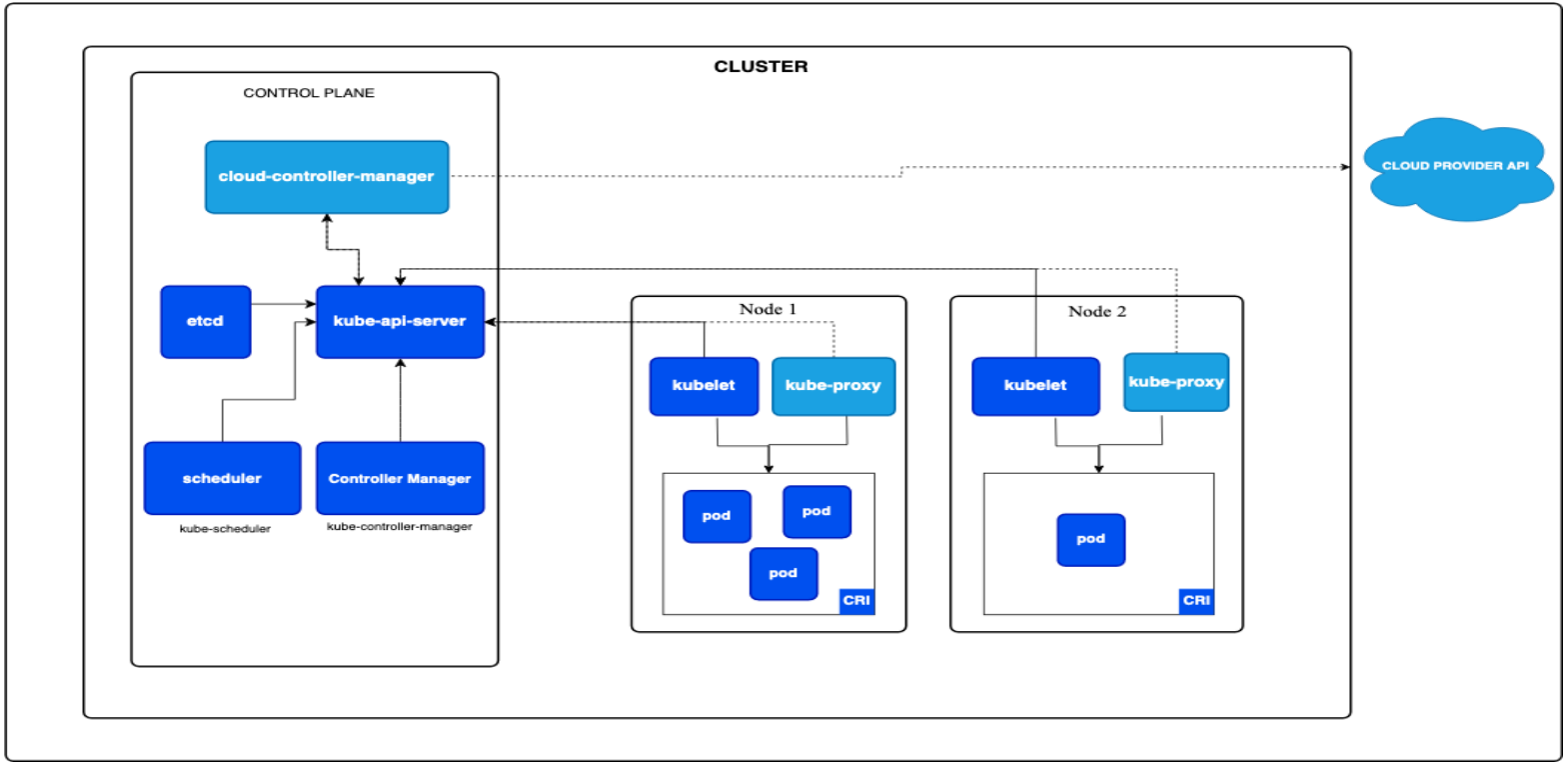
1. Kubernetes
2. On premises
3. Fabric y calico

Arquitectura kubernetes: Visión Global



Kubernetes.io. (n.d.). *kubernetes components*.

Arquitectura kubernetes: Visión Global



Kubernetes.io. (n.d.). *kubernetes components*. https://kubernetes.io/docs/concepts/overview/arquitecture/.

Arquitectura: componentes

Otros componentes:

Services: Abstracción para exponer aplicaciones que se ejecutan en un conjunto de pods.

ConfigMap y Secrets: Permiten separar la configuración y los secretos de la imagen del contenedor para reusabilidad y desacoplamiento.

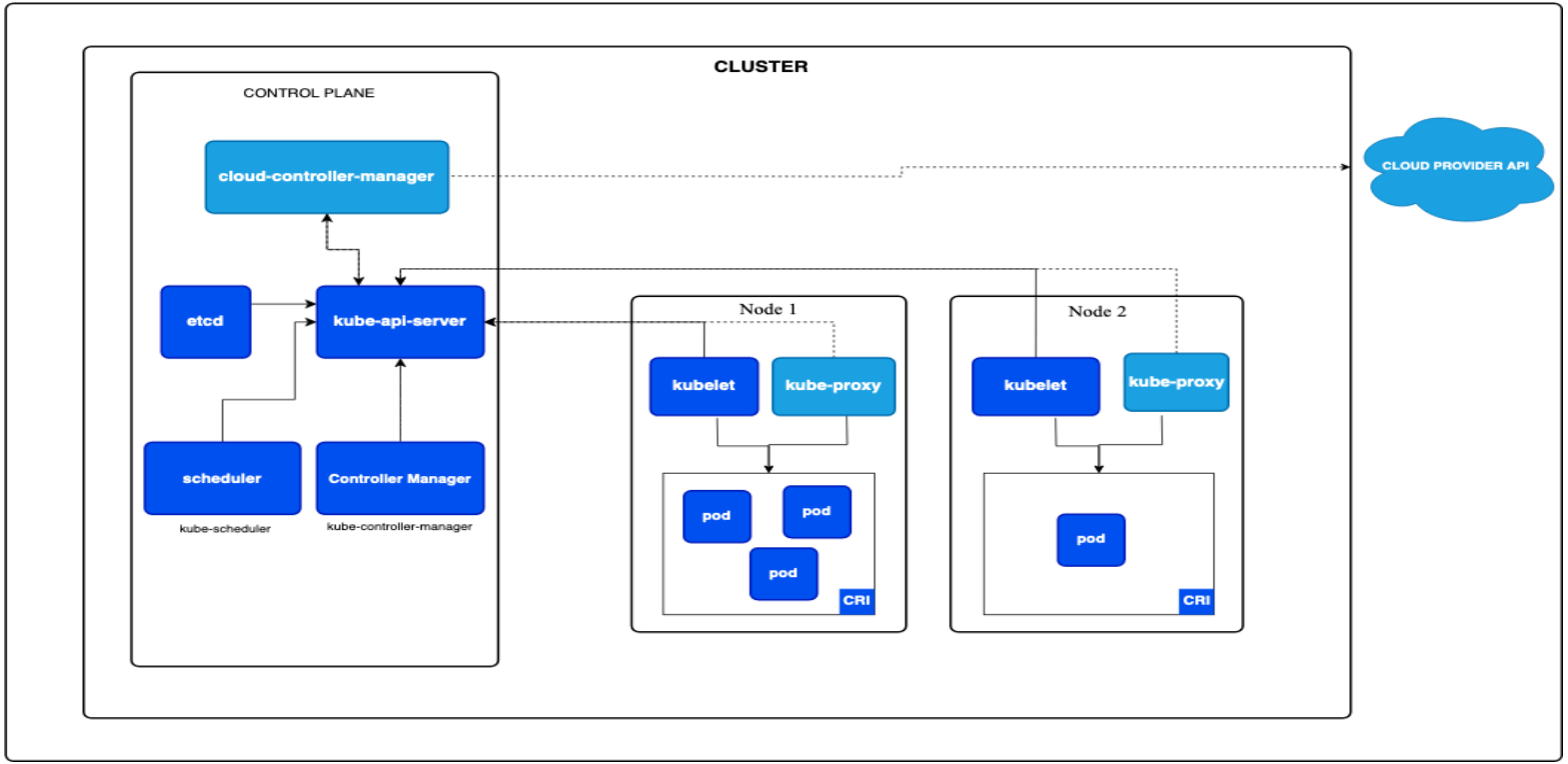
Volumes: Almacenamiento persistente para que los datos sobrevivan a reinicios de pods.

Arquitectura: componentes

Ingress: Gestiona el acceso externo a los servicios en un clúster, generalmente HTTP.

Helm: Es un gestor de paquetes que facilita la definición, instalación y actualización de aplicaciones en Kubernetes.

Arquitecturas: redes, CNI



Kubernetes.io. (n.d.). *kubernetes components*. https://kubernetes.io/docs/concepts/overview/arquitecture/.

Arquitecturas: redes, CNI

Un CNI (Container Network Interface) en Kubernetes es un estándar y una especificación que define cómo deben ser configuradas las interfaces de red para los contenedores. Kubernetes utiliza plugins CNI para gestionar la conectividad de red de los pods, permitiendo a diferentes soluciones de red integrarse y operar con Kubernetes de manera uniforme. Estos plugins garantizan que los contenedores dentro de los pods tengan una adecuada conectividad de red y configuraciones correctas, facilitando la comunicación entre los pods, tanto dentro del mismo nodo como entre diferentes nodos del clúster.

Arquitectura: redes

Cuando Kubernetes crea una red para un pod utilizando un complemento de Interfaz de Red de Contenedor (CNI, por sus siglas en inglés), sigue una serie de pasos para garantizar el aislamiento y la conectividad de la red.

Arquitectura: redes

Cada pod en Kubernetes recibe su propio espacio de nombres de red. Un espacio de nombres de red es una pila de redes aislada que incluye interfaces, rutas y reglas de firewall. Este aislamiento asegura que la configuración de red de un pod no interfiera con la de otro pod.

Arquitectura: redes

Uno de los pasos comunes que realizan los complementos CNI es crear un par de dispositivos Ethernet virtuales (veth). Estos pares veth se utilizan para la comunicación entre el espacio de nombres de red del pod y el espacio de nombres de red del nodo.

```
ip link show type veth
```

Arquitectura: redes

Uno de los extremos del par veth se coloca dentro del espacio de nombres de red del pod. El otro extremo se coloca en el espacio de nombres de red del nodo.

El par veth permite que el tráfico fluya entre el pod y el nodo, que actúa como puente hacia la red externa.

Arquitectura: redes

El complemento CNI asigna una dirección IP y otras configuraciones de red al dispositivo veth dentro del espacio de nombres de red del pod. Esta dirección IP se obtiene generalmente de un grupo de direcciones IP designado y gestionado por el complemento CNI.

Arquitectura: redes

Dentro del espacio de nombres de red del pod, se configuran tablas de enrutamiento para dirigir el tráfico de manera apropiada. Esto incluye la definición de una ruta predeterminada, que generalmente se establece para apuntar al extremo del veth en el espacio de nombres de red del nodo.

Para comprobarlo:

Arquitectura: redes

```
kubectl exec -it <nombre-del-pod> -- ip route show

default via 172.17.0.1 dev eth0

172.17.0.0/16 dev eth0 proto kernel scope link src 172.17.0.2
```

Arquitectura: redes

Cuando el pod envía tráfico a destinos fuera de su subred, sigue la ruta predeterminada, enviando paquetes a través del par veth hacia el espacio de nombres de red del nodo.

El espacio de nombres de red del nodo luego reenvía los paquetes al destino apropiado según su propia tabla de enrutamiento y configuración de red.

Arquitectura: redes

En el espacio del red del nodo coloca una ruta hacia la subnet del pod.

Arquitecturas: redes

Tres tipos de redes:

- Red de los nodos.
- Red de los pods.
- Red del cluster

Arquitectura: exposición de servicios, cluster IP

En Kubernetes, cuando un paquete está destinado a un servicio, inicialmente se envía a la IP de clúster del servicio. El servicio actúa como un balanceador de carga, dirigiendo el tráfico a uno de los pods detrás del servicio según la estrategia de equilibrio de carga del servicio.

Arquitectura: exposición de servicios

La IP de destino original del paquete es la IP de clúster del servicio de destino.

El servicio, a su vez, reenvía el paquete a uno de los pods que forman parte del servicio, y la IP de destino del paquete se traduce a la dirección IP del pod seleccionado.

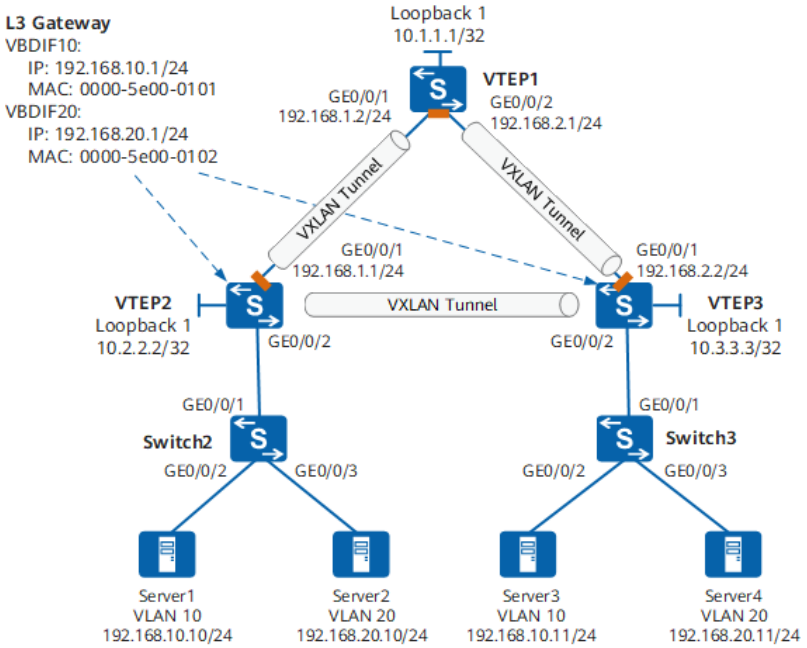
Calico: Consideraciones de diseño

Vamos a desplegar Calico en un entorno on-premise sin overlay. El plano de control de enrutamiento se realizará mediante BGP con nuestro amigo BIRD versión 2 que se ejecuta en cada nodo de Kubernetes.

Equipos de red: Consideraciones de diseño

En nuestros equipos de red tendremos un VXLAN EVPN Fabric. Con un gateway anycast en todos los switches leaf que se conectan a los nodos de Kubernetes, lo que significa que se puede utilizar una sola subred para las direcciones IP de los nodos de todo el clúster de Kubernetes. De esta forma solo se necesita una VLAN/VNI/subred para todos los nodos de Kubernetes, independientemente de dónde estén conectados con lo que conseguimos además de reducir los recursos simplificar la configuración.

Configuración fabric de los equipos Huawei



https://support.huawei.com/enterprise/en/doc/EDOC1100197300/916d2c2e/example-for-configuring-vxlan-for-a-virtual-network-in-distributed-gateway-mode-bgp-evpn-mode

Configuración fabric de los equipos Huawei

Como se muestra anterior, los nodos están desplegados en diferentes datacenters. Server1 y Server3 están en un segmento de red, mientras que Server2 y Server4 están en otro segmento de red. Se deben utilizar túneles Virtual Extensible LAN (VXLAN) para la interconexión de capa con las direcciones ips de los nodos y para la interconexión de capa 3 entre direcciones de los pods.

Configuración fabric de los equipos Huawei

Equipo	Interfaz	Dirección IP
VTEP1	Loopback 1	10.1.1.1
VTEP1	GE/0/0/1	192.168.1.2/24
VTEP1	GE/0/0/2	192.168.2.1/24
VTEP2	Loopback 1	10.2.2.2
VTEP2	GE/0/0/1	192.168.1.1/24
VTEP3	Loopback 1	10.3.3.3
VTEP3	GE/0/0/1	192.168.2.2/24

Equipo	Interfaz	Dirección IP
VTEP2	vbdif 10	192.168.10.1/24
VTEP2	vbdif 20	192.168.20.1/24
VTEP3	vbdif 10	192.168.10.1/24
VTEP3	vbdif 20	192.168.20.1/24

Configuración básica VTEP2

```
<HUAWEI> system-view
[HUAWEI] sysname VTEP2
[VTEP2] isis 1
[VTEP2-isis-1] is-level level-2
[VTEP2-isis-1] cost-style wide
[VTEP2-isis-1] network-entity 4900.0100.0202.0202.00
[VTEP2-isis-1] quit
[VTEP2] interface loopback 1
[VTEP2-LoopBack1] ip address 10.2.2.2 32
[VTEP2-LoopBack1] isis enable 1
[VTEP2-LoopBack1] quit
[VTEP2] interface gigabitethernet 0/0/1
[VTEP2-GigabitEthernet0/0/1] undo portswitch
[VTEP2-GigabitEthernet0/0/1] ip address 192.168.1.1 24
[VTEP2-GigabitEthernet0/0/1] isis enable 1
[VTEP2-GigabitEthernet0/0/1] isis circuit-level level-2
[VTEP2-GigabitEthernet0/0/1] quit
```

Configuración básica Switch2

```
<HUAWEI> system-view
[HUAWEI] sysname Switch2
[Switch2] vlan batch 10 20
[Switch2] interface gigabitethernet 0/0/1
[Switch2-GigabitEthernet0/0/1] port link-type trunk
[Switch2-GigabitEthernet0/0/1] port trunk allow-pass vlan 10 20
[Switch2-GigabitEthernet0/0/1] quit
[Switch2] interface gigabitethernet 0/0/2
[Switch2-GigabitEthernet0/0/2] port link-type access
[Switch2-GigabitEthernet0/0/2] port default vlan 10
[Switch2-GigabitEthernet0/0/2] quit
[Switch2] interface gigabitethernet 0/0/3
[Switch2-GigabitEthernet0/0/3] port link-type access
[Switch2-GigabitEthernet0/0/3] port default vlan 20
[Switch2-GigabitEthernet0/0/3] quit
```

Configuración básica VTEP3

```
<HUAWEI> system-view
[HUAWEI] sysname VTEP3
[VTEP3] isis 1
[VTEP3-isis-1] is-level level-2
[VTEP3-isis-1] cost-style wide
[VTEP3-isis-1] network-entity 4900.0100.0303.0303.00
[VTEP3-isis-1] quit
[VTEP3] interface loopback 1
[VTEP3-LoopBack1] ip address 10.3.3.3 32
[VTEP3-LoopBack1] isis enable 1
[VTEP3-LoopBack1] quit
[VTEP3] interface gigabitethernet 0/0/1
[VTEP3-GigabitEthernet0/0/1] undo portswitch
[VTEP3-GigabitEthernet0/0/1] ip address 192.168.2.2 24
[VTEP3-GigabitEthernet0/0/1] isis enable 1
[VTEP3-GigabitEthernet0/0/1] isis circuit-level level-2
[VTEP3-GigabitEthernet0/0/1] quit
```

Configuración básica Switch3

```
<HUAWEI> system-view
[HUAWEI] sysname Switch3
[Switch3] interface gigabitethernet 0/0/1
[Switch3-GigabitEthernet0/0/1] port link-type trunk
[Switch3-GigabitEthernet0/0/1] port trunk allow-pass vlan 10 20
[Switch3-GigabitEthernet0/0/1] quit
[Switch3] interface gigabitethernet 0/0/2
[Switch3-GigabitEthernet0/0/2] port link-type access
[Switch3-GigabitEthernet0/0/2] port default vlan 10
[Switch3-GigabitEthernet0/0/2] quit
[Switch3] interface gigabitethernet 0/0/3
[Switch3-GigabitEthernet0/0/3] port link-type access
[Switch3-GigabitEthernet0/0/3] port default vlan 20
[Switch3-GigabitEthernet0/0/3] quit
```

Configuración básica VTEP1

```
<HUAWEI> system-view
[HUAWEI] sysname VTEP1
[VTEP1] isis 1
[VTEP1-isis-1] is-level level-2
[VTEP1-isis-1] cost-style wide
[VTEP1-isis-1] network-entity 4900.0100.0101.0101.00
[VTEP1-isis-1] quit
[VTEP1] interface loopback 1
[VTEP1-LoopBack1] ip address 10.1.1.1 32
[VTEP1-LoopBack1] isis enable 1
[VTEP1-LoopBack1] quit
[VTEP1] interface gigabitethernet 0/0/1
[VTEP1-GigabitEthernet0/0/1] undo portswitch
[VTEP1-GigabitEthernet0/0/1] ip address 192.168.1.2 24
[VTEP1-GigabitEthernet0/0/1] isis enable 1
[VTEP1-GigabitEthernet0/0/1] isis circuit-level level-2
[VTEP1-GigabitEthernet0/0/1] quit
[VTEP1] interface gigabitethernet 0/0/1
[VTEP1-GigabitEthernet0/0/2] undo portswitch
[VTEP1-GigabitEthernet0/0/2] ip address 192.168.2.1 24
[VTEP1-GigabitEthernet0/0/2] isis enable 1
[VTEP1-GigabitEthernet0/0/2] isis circuit-level level-2
[VTEP1-GigabitEthernet0/0/2] quit
```

Configuración bridge VTEP2

```
[VTEP2] bridge-domain 10
[VTEP2-bd10] quit
[VTEP2] vcmp role silent
[VTEP2] interface gigabitethernet 0/0/2
[VTEP2-GigabitEthernet0/0/2] port link-type trunk
[VTEP2-GigabitEthernet0/0/2] quit
[VTEP2] interface gigabitethernet 0/0/2.10
[VTEP2-GigabitEthernet0/0/2.10] encapsulation dot1q vid 10
[VTEP2-GigabitEthernet0/0/2.10] bridge-domain 10
[VTEP2-GigabitEthernet0/0/2.10] quit
[VTEP2] bridge-domain 20
[VTEP2-bd20] quit
[VTEP2] interface gigabitethernet 0/0/2.20
[VTEP2-GigabitEthernet0/0/2.20] encapsulation dot1q vid 20
[VTEP2-GigabitEthernet0/0/2.20] bridge-domain 20
[VTEP2-GigabitEthernet0/0/2.20] quit
```

Configuración bridge VTEP3

```
[VTEP3] bridge-domain 10
[VTEP3-bd10] quit
[VTEP3] vcmp role silent
[VTEP3] interface gigabitethernet 0/0/2
[VTEP3-GigabitEthernet0/0/2] port link-type trunk
[VTEP3-GigabitEthernet0/0/2] quit
[VTEP3] interface gigabitethernet 0/0/2.10
[VTEP3-GigabitEthernet0/0/2.10] encapsulation dot1q vid 10
[VTEP3-GigabitEthernet0/0/2.10] bridge-domain 10
[VTEP3-GigabitEthernet0/0/2.10] quit
[VTEP3] bridge-domain 20
[VTEP3-bd20] quit
[VTEP3] interface gigabitethernet 0/0/2.20
[VTEP3-GigabitEthernet0/0/2.20] encapsulation dot1q vid 20
[VTEP3-GigabitEthernet0/0/2.20] bridge-domain 20
[VTEP3-GigabitEthernet0/0/2.20] quit
```

Configuración evpn VTEP2

```
[VTEP2] evpn vpn-instance evpn10 bd-mode
[VTEP2-evpn-instance-evpn10] route-distinguisher 1:10
[VTEP2-evpn-instance-evpn10] vpn-target 10:1 both
[VTEP2-evpn-instance-evpn10] vpn-target 1:100 export-extcommunity
[VTEP2-evpn-instance-evpn10] quit
[VTEP2] bridge-domain 10
[VTEP2-bd10] vxlan vni 10
[VTEP2-bd10] evpn binding vpn-instance evpn10
[VTEP2-bd10] quit
[VTEP2] evpn vpn-instance evpn20 bd-mode
[VTEP2-evpn-instance-evpn20] route-distinguisher 1:20
[VTEP2-evpn-instance-evpn20] vpn-target 20:1 both
[VTEP2-evpn-instance-evpn20] vpn-target 1:100 export-extcommunity
[VTEP2-evpn-instance-evpn20] quit
[VTEP2] bridge-domain 20
[VTEP2-bd20] vxlan vni 20
[VTEP2-bd20] evpn binding vpn-instance evpn20
[VTEP2-bd20] quit
```

Configuración evpn VTEP3

```
[VTEP3] evpn vpn-instance evpn10 bd-mode
[VTEP3-evpn-instance-evpn10] route-distinguisher 2:10
[VTEP3-evpn-instance-evpn10] vpn-target 10:1 both
[VTEP3-evpn-instance-evpn10] vpn-target 1:100 export-extcommunity
[VTEP3-evpn-instance-evpn10] quit
[VTEP3] bridge-domain 10
[VTEP3-bd10] vxlan vni 10
[VTEP3-bd10] evpn binding vpn-instance evpn10
[VTEP3-bd10] quit
[VTEP3] evpn vpn-instance evpn20 bd-mode
[VTEP3-evpn-instance-evpn20] route-distinguisher 2:20
[VTEP3-evpn-instance-evpn20] vpn-target 20:1 both
[VTEP3-evpn-instance-evpn20] vpn-target 1:100 export-extcommunity
[VTEP3-evpn-instance-evpn20] quit
[VTEP3] bridge-domain 20
[VTEP3-bd20] vxlan vni 20
[VTEP3-bd20] evpn binding vpn-instance evpn20
[VTEP3-bd20] quit
```

Configuración instancia vpn VTEP1

```
[VTEP1] ip vpn-instance vpn1
[VTEP1-vpn-instance-vpn1] ipv4-family
[VTEP1-vpn-instance-vpn1-af-ipv4] route-distinguisher 1:100
[VTEP1-vpn-instance-vpn1-af-ipv4] vpn-target 1:100 both evpn
[VTEP1-vpn-instance-vpn1-af-ipv4] quit
[VTEP1-vpn-instance-vpn1] vxlan vni 100
[VTEP1-vpn-instance-vpn1] quit
```

Configuración instancia vpn VTEP2

```
[VTEP2] ip vpn-instance vpn1
[VTEP2-vpn-instance-vpn1] ipv4-family
[VTEP2-vpn-instance-vpn1-af-ipv4] route-distinguisher 2:100
[VTEP2-vpn-instance-vpn1-af-ipv4] vpn-target 1:100 both evpn
[VTEP2-vpn-instance-vpn1-af-ipv4] quit
[VTEP2-vpn-instance-vpn1] vxlan vni 100
[VTEP2-vpn-instance-vpn1] quit
[VTEP2] interface vbdif 10
[VTEP2-Vbdif10] ip binding vpn-instance vpn1
[VTEP2-Vbdif10] quit
[VTEP2] interface vbdif 20
[VTEP2-Vbdif20] ip binding vpn-instance vpn1
[VTEP2-Vbdif20] quit
```

Configuración instancia vpn VTEP3

```
[VTEP3] ip vpn-instance vpn1
[VTEP3-vpn-instance-vpn1] ipv4-family
[VTEP3-vpn-instance-vpn1-af-ipv4] route-distinguisher 3:100
[VTEP3-vpn-instance-vpn1-af-ipv4] vpn-target 1:100 both evpn
[VTEP3-vpn-instance-vpn1-af-ipv4] quit
[VTEP3-vpn-instance-vpn1] vxlan vni 100
[VTEP3-vpn-instance-vpn1] quit
[VTEP3] interface vbdif 10
[VTEP3-Vbdif10] ip binding vpn-instance vpn1
[VTEP3-Vbdif10] quit
[VTEP3] interface vbdif 20
[VTEP3-Vbdif20] ip binding vpn-instance vpn1
[VTEP3-Vbdif20] quit
```

Configuración BGP VTEP1

```
[VTEP1] bgp 100
[VTEP1-bgp] router-id 10.1.1.1
[VTEP1-bgp] peer 10.2.2.2 as-number 100
[VTEP1-bgp] peer 10.2.2.2 connect-interface LoopBack1
[VTEP1-bgp] peer 10.3.3.3 as-number 100
[VTEP1-bgp] peer 10.3.3.3 connect-interface LoopBack1
[VTEP1-bgp] l2vpn-family evpn
[VTEP1-bgp-af-evpn] peer 10.2.2.2 enable
[VTEP1-bgp-af-evpn] peer 10.2.2.2 advertise irb
[VTEP1-bgp-af-evpn] peer 10.3.3.3 enable
[VTEP1-bgp-af-evpn] peer 10.3.3.3 advertise irb
[VTEP1-bgp-af-evpn] quit
[VTEP1-bgp] ipv4-family vpn-instance vpn1
[VTEP1-bgp-vpn1] advertise l2vpn evpn
[VTEP1-bgp-vpn1] import–route direct
[VTEP1-bgp-vpn1] quit
[VTEP1-bgp] quit
```

Configuración BGP VTEP2

```
[VTEP2] bgp 100
[VTEP2-bgp] router-id 10.2.2.2
[VTEP2-bgp] peer 10.1.1.1 as-number 100
[VTEP2-bgp] peer 10.1.1.1 connect-interface LoopBack1
[VTEP2-bgp] peer 10.3.3.3 as-number 100
[VTEP2-bgp] peer 10.3.3.3 connect-interface LoopBack1
[VTEP2-bgp] l2vpn-family evpn
[VTEP2-bgp-af-evpn] peer 10.1.1.1 enable
[VTEP2-bgp-af-evpn] peer 10.1.1.1 advertise irb
[VTEP2-bgp-af-evpn] peer 10.3.3.3 enable
[VTEP2-bgp-af-evpn] peer 10.3.3.3 advertise irb
[VTEP2-bgp-af-evpn] quit
[VTEP2-bgp] ipv4-family vpn-instance vpn1
[VTEP2-bgp-vpn1] advertise l2vpn evpn
[VTEP2-bgp-vpn1] import–route direct
[VTEP2-bgp-vpn1] quit
[VTEP2-bgp] quit
```

Configuración BGP VTEP3

```
[VTEP3] bgp 100
[VTEP3-bgp] router-id 10.3.3.3
[VTEP3-bgp] peer 10.1.1.1 as-number 100
[VTEP3-bgp] peer 10.1.1.1 connect-interface LoopBack1
[VTEP3-bgp] peer 10.2.2.2 as-number 100
[VTEP3-bgp] peer 10.2.2.2 connect-interface LoopBack1
[VTEP3-bgp] l2vpn-family evpn
[VTEP3-bgp-af-evpn] peer 10.1.1.1 enable
[VTEP3-bgp-af-evpn] peer 10.1.1.1 advertise irb
[VTEP3-bgp-af-evpn] peer 10.2.2.2 enable
[VTEP3-bgp-af-evpn] peer 10.2.2.2 advertise irb
[VTEP3-bgp-af-evpn] quit
[VTEP3-bgp] ipv4-family vpn-instance vpn1
[VTEP3-bgp-vpn1] advertise l2vpn evpn
[VTEP3-bgp-vpn1] import–route direct
[VTEP3-bgp-vpn1] quit
[VTEP3-bgp] quit
```

Configuración túnel VXLAN VTEP1

```
[VTEP1] interface vne 1
[VTEP1-Nve1] source 10.1.1.1
[VTEP1-Nve1] quit
```

Configuración túnel VXLAN¿? VTEP2

```
[VTEP2] interface vne 1
[VTEP2-Nve1] source 10.2.2.2
[VTEP2-Nve1] vni 10 head-end peer-list protocol bgp
[VTEP2-Nve1] vni 20 head-end peer-list protocol bgp
[VTEP2-Nve1] quit
```

Configuración túnel VXLAN¿? VTEP3

```
[VTEP3] interface vne 1
[VTEP3-Nve1] source 10.3.3.3
[VTEP3-Nve1] vni 10 head-end peer-list protocol bgp
[VTEP3-Nve1] vni 20 head-end peer-list protocol bgp
[VTEP3-Nve1] quit
```

Configuración VXLAN gateway distribuido¿? VTEP2

```
[VTEP2] interface vbdif 10
[VTEP2-Vbdif10] ip address 192.168.10.1 24
[VTEP2-Vbdif10] arp distribute-gateway enable
[VTEP2-Vbdif10] arp collect host enable
[VTEP2-Vbdif10] mac-address 0000-5e00-0101
[VTEP2-Vbdif10] quit
[VTEP2] interface vbdif 20
[VTEP2-Vbdif20] ip address 192.168.20.1 24
[VTEP2-Vbdif20] arp distribute-gateway enable
[VTEP2-Vbdif20] arp collect host enable
[VTEP2-Vbdif20] mac-address 0000-5e00-0102
[VTEP2-Vbdif20] quit
```

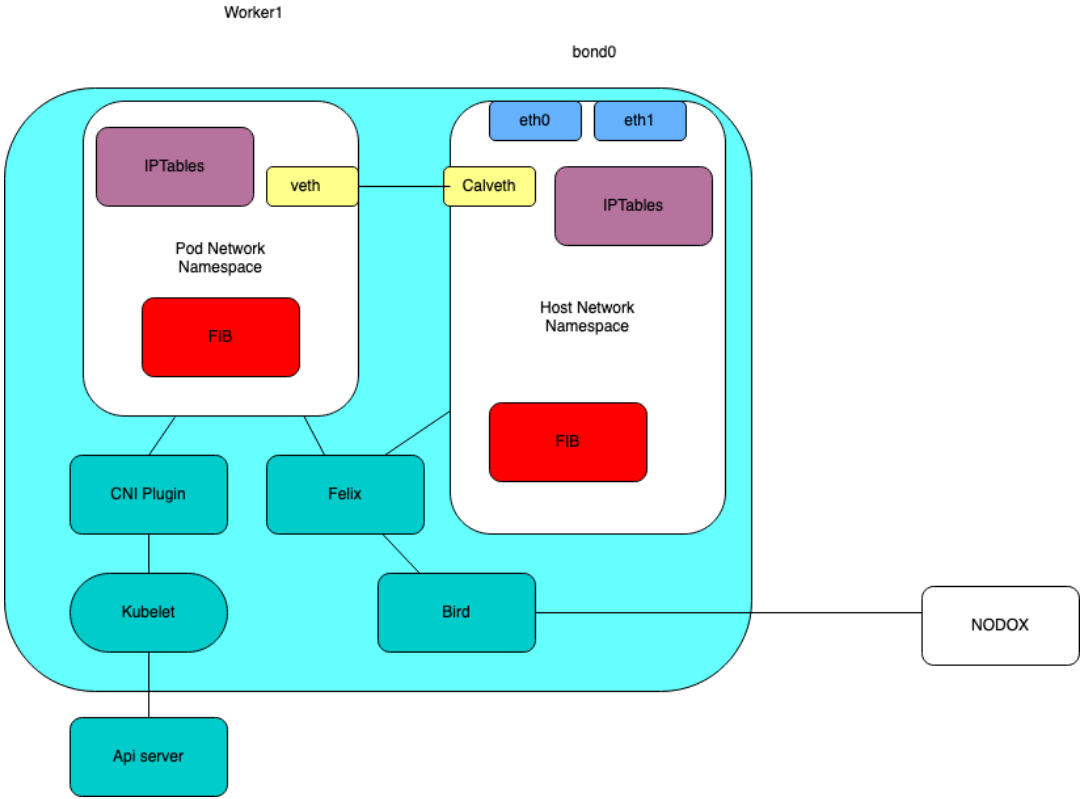
Configuración VXLAN gateway distribuido¿? VTEP3

```
[VTEP3] interface vbdif 10
[VTEP3-Vbdif10] ip address 192.168.10.1 24
[VTEP3-Vbdif10] arp distribute-gateway enable
[VTEP3-Vbdif10] arp collect host enable
[VTEP3-Vbdif10] mac-address 0000-5e00-0101
[VTEP3-Vbdif10] quit
[VTEP3] interface vbdif 20
[VTEP3-Vbdif20] ip address 192.168.20.1 24
[VTEP3-Vbdif20] arp distribute-gateway enable
[VTEP3-Vbdif20] arp collect host enable
[VTEP3-Vbdif20] mac-address 0000-5e00-0102
[VTEP3-Vbdif20] quit
```

Calico: Componentes

- CNI
- Bird
- Felix

Calico: Componentes



Calico: Componentes

Felix actúa como el componente crucial del plano de datos, siendo responsable de configurar las reglas de iptables y facilitar el reenvío de paquetes dentro del clúster. Colabora estrechamente con el plugin Calico CNI (Interfaz de Red de Contenedor) para crear espacios de nombres de red y configurar interfaces para los pods.

Calico: Componentes

BIRD (Daemon de la Base de Información del Protocolo de Puerta de Enlace) desempeña un papel fundamental en la enrutación dinámica de Calico, intercambiando información de enrutamiento y asegurando una comunicación eficiente entre los pods y las redes externas.

Calico: Felix, gestión de interfaces

Programa información sobre interfaces en el kernel para que el kernel pueda gestionar correctamente el tráfico desde ese punto final. En particular, se asegura de que el host responda a las solicitudes ARP de cada carga de trabajo con la dirección MAC del host y habilita el reenvío de IP para las interfaces que administra. También supervisa las interfaces para garantizar que la programación se aplique en el momento adecuado.

Calico: Felix, ACL’s

Programa listas de control de acceso (ACL) en el kernel de Linux para garantizar que solo el tráfico válido pueda ser enviado entre puntos finales y que los puntos finales no puedan eludir las medidas de seguridad de Calico.

Calico: Felix, Monitoring

Monitoriza el estado de la red. En particular, informa sobre errores y problemas al configurar su host. Estos datos se escriben en el almacén de datos para que sean visibles para otros componentes y operadores de la red.

Calico: Bird

BIRD, que significa "Bird Internet Routing Daemon", es un avanzado software de enrutamiento de código abierto diseñado principalmente para sistemas operativos tipo Unix. Actúa como un daemon de protocolo de enrutamiento, gestionando de manera efectiva el intercambio de información de enrutamiento entre routers en redes. Soporta varios protocolos de enrutamiento nos centraremos en BGP.

Calico: Bird

Modularidad

Soporte para IPv4 e IPv6

Enrutamiento basado en políticas

Filtrado y Manipulación de Rutas

Agregación de Rutas

Validación de Rutas

Actualizaciones de Rutas Dinámicas

Calico: Bird, configuración RR

Los reflectores de rutas BGP se configuran con frecuencia en despliegues grandes en lugar de utilizar un cliente BGP estándar. Los reflectores de rutas BGP actúan como un punto central para conectar clientes BGP. (El BGP estándar requiere que cada cliente BGP esté conectado a todos los demás clientes BGP en una topología de malla, lo que es difícil de mantener).

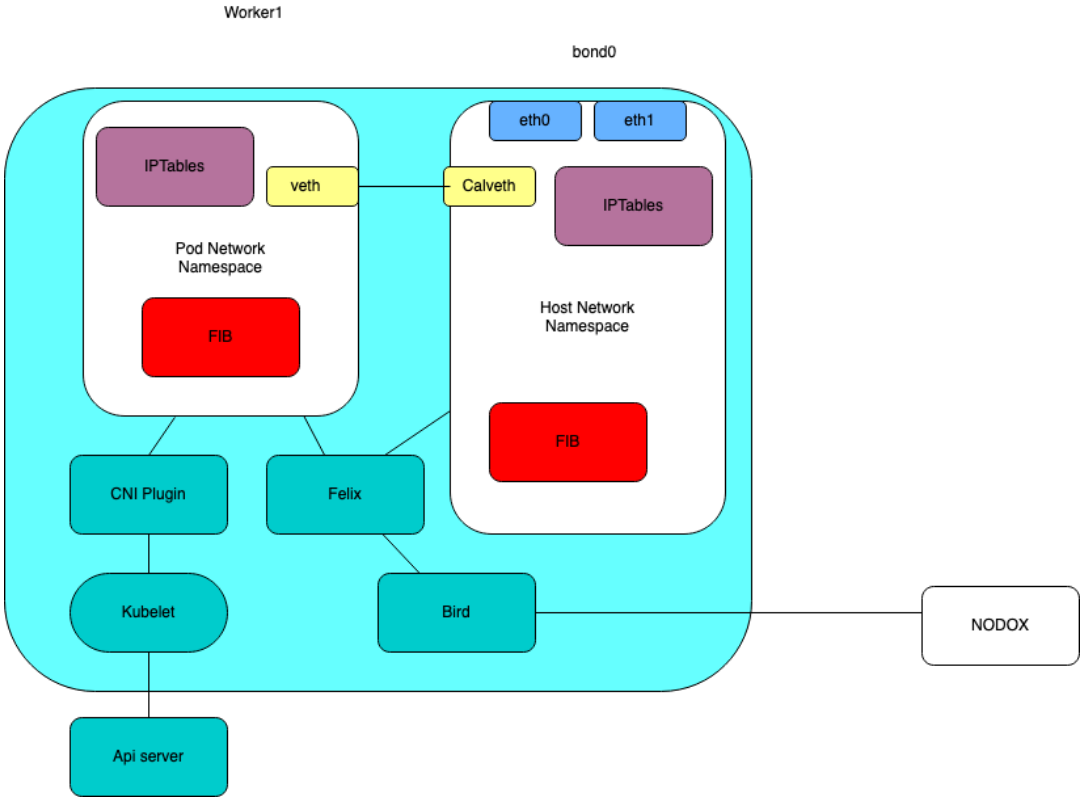
Calico: Bird, configuración RR

Para redundancia, puedes implementar de manera transparente múltiples reflectores de ruta BGP. Los reflectores de ruta BGP solo están involucrados en el plano de control de la red: ningún dato de forwarding a través de ellos. Cuando el cliente BGP de Calico anuncia rutas desde su FIB al reflector de ruta, el reflector de ruta anuncia esas rutas a los otros nodos en la implementación.

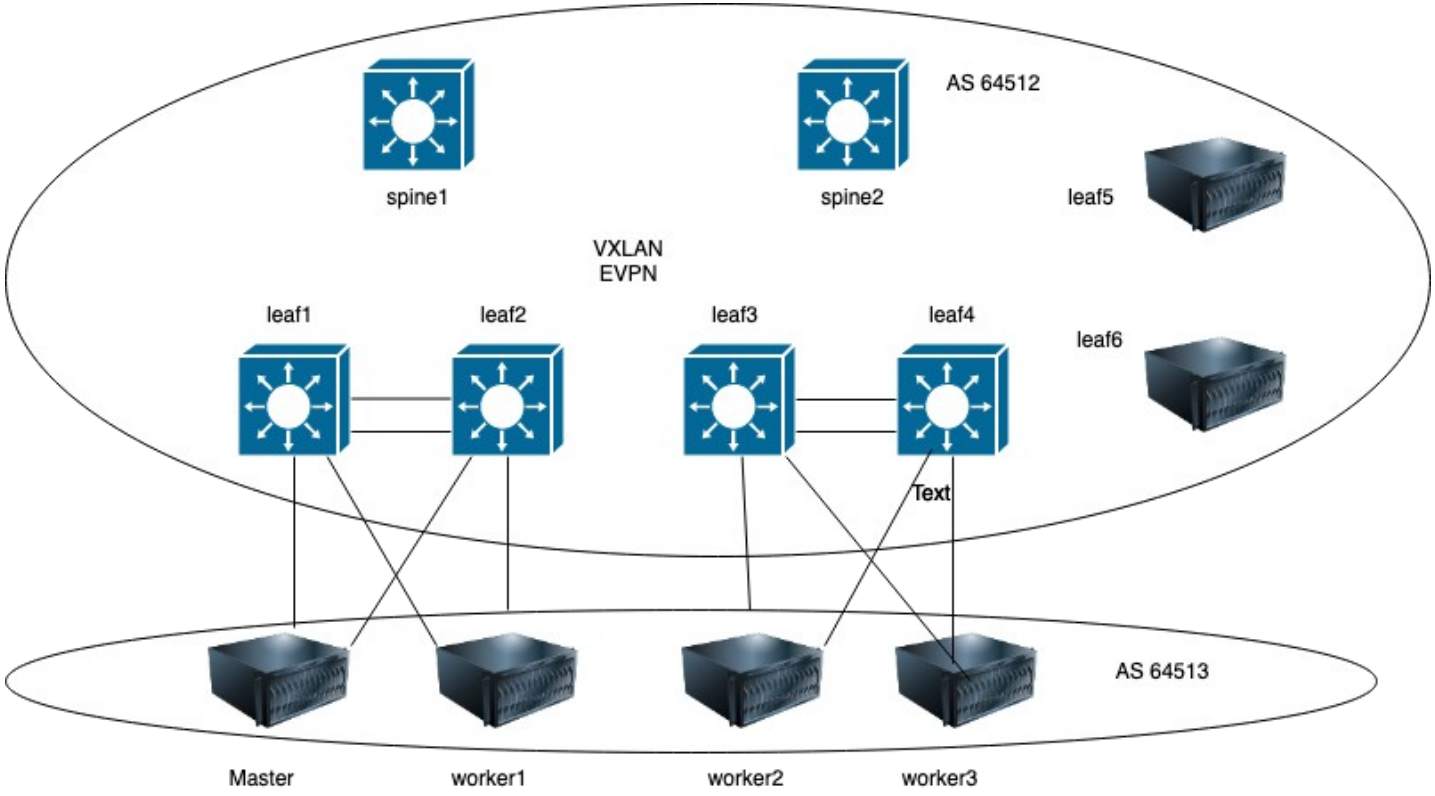
Calico: Binario CNI

El binario de Calico que presenta esta API a Kubernetes se llama el complemento CNI, y debe instalarse en cada nodo del clúster de Kubernetes. El complemento CNI de Calico te permite utilizar la red de Calico para cualquier orquestador que haga uso de la especificación de redes CNI. Se configura a través del mecanismo de configuración estándar de CNI y el complemento CNI de Calico.

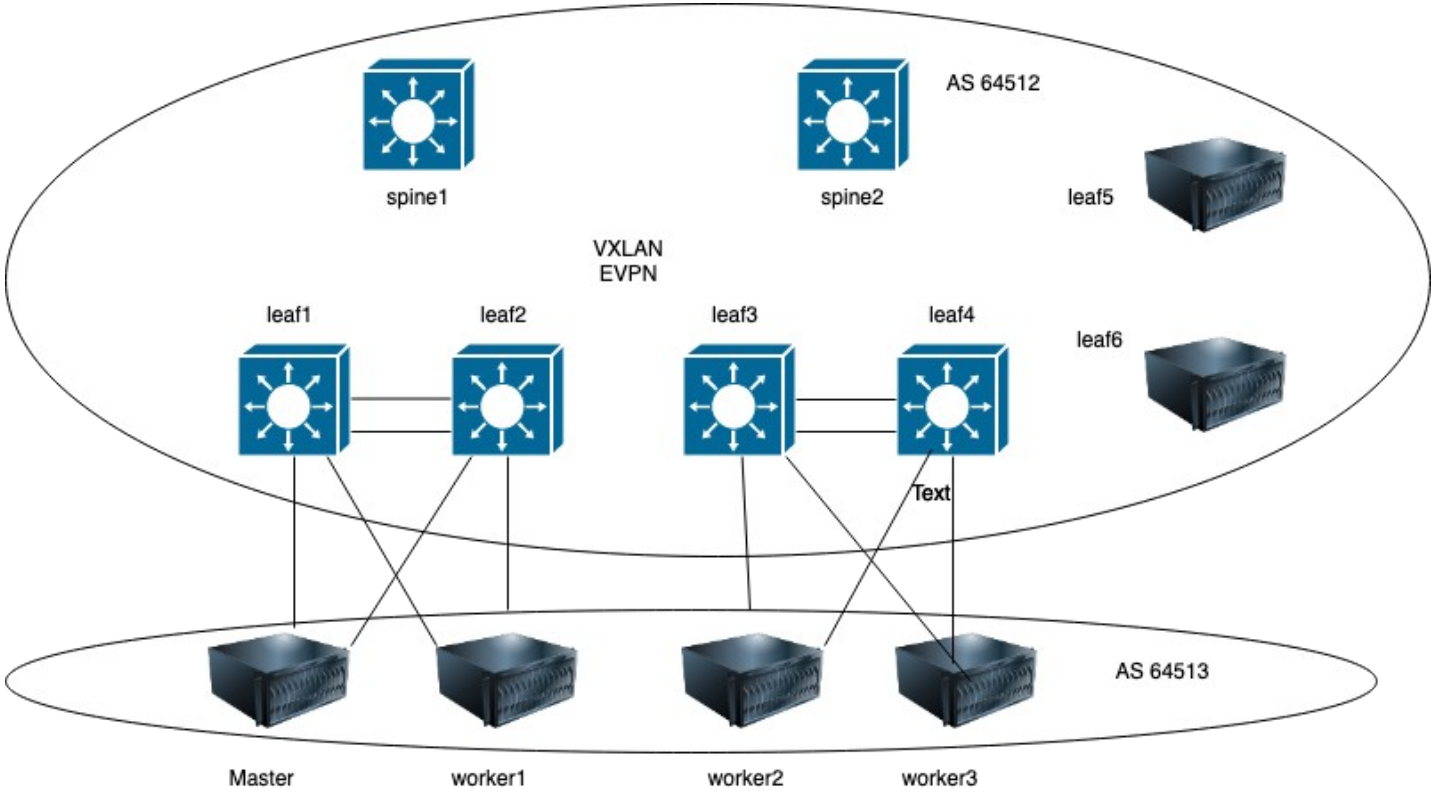
Calico: en un fabric



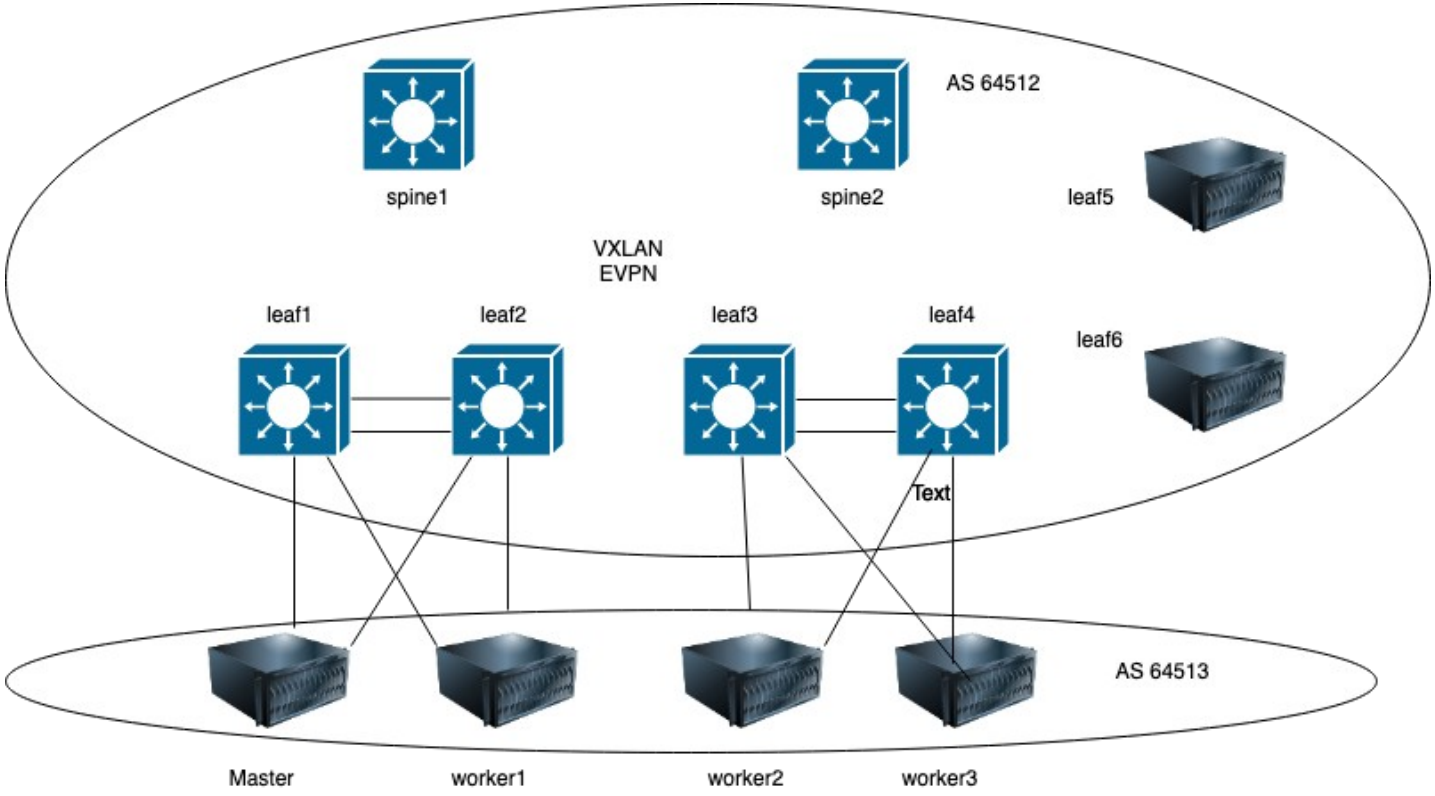
Calico en fabric: Solución de peering centralizado



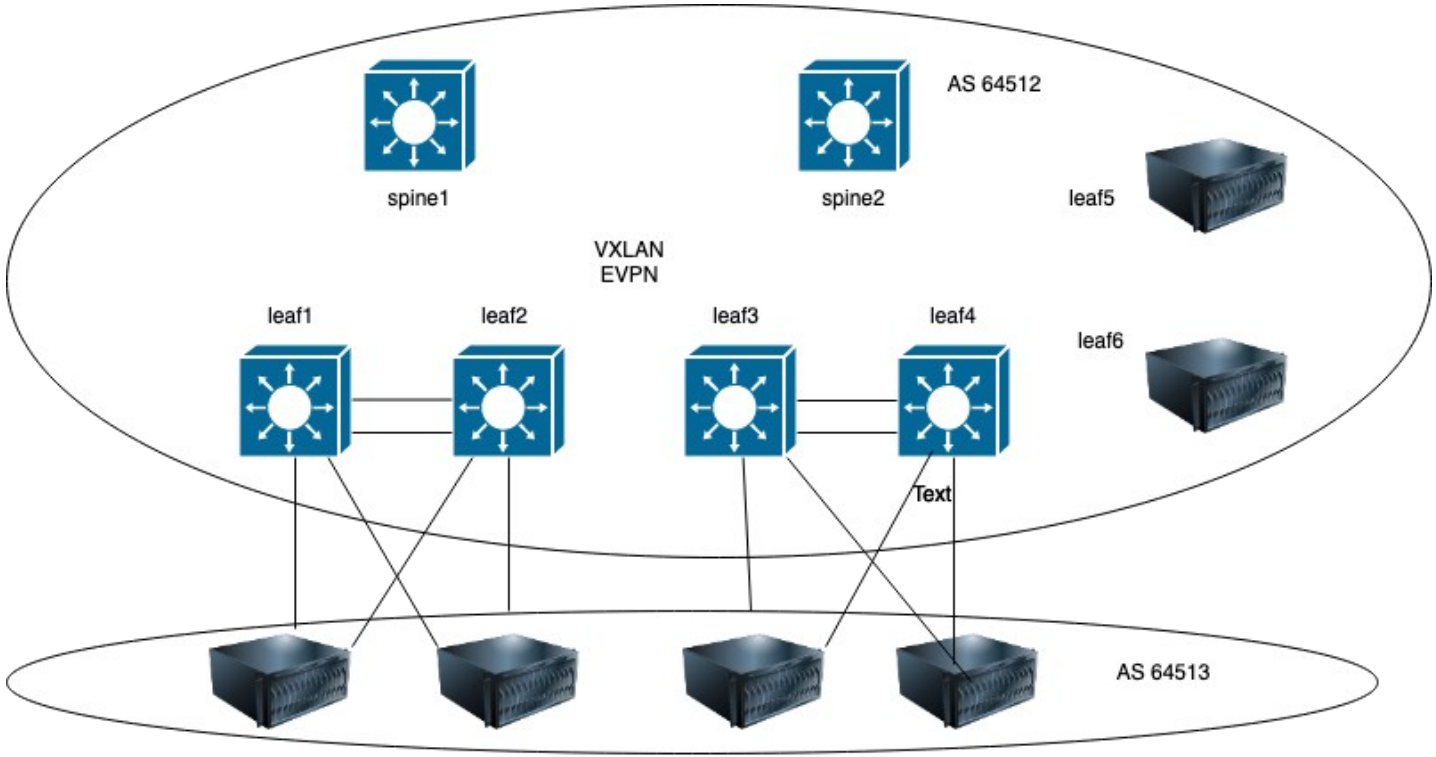
Calico en fabric: ¿Qué pasa con los anuncios del bgp?



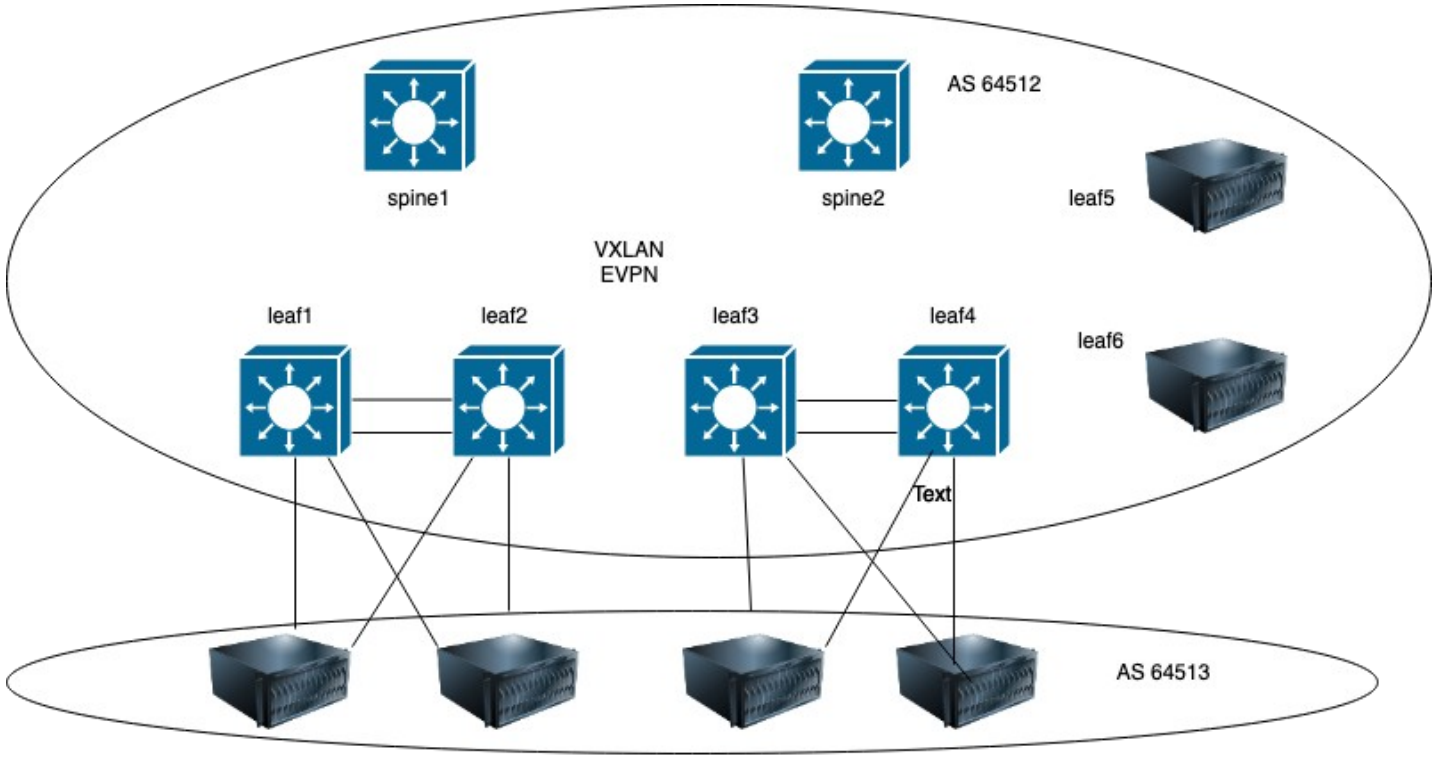
Calico en fabric: ¿Cómo se resuelve la congestión?



Calico en fabric: ¿Cómo se resuelven varios caminos a la subred de servicio?



Calico en fabric: ¿Cómo se configura Calico?



Muchas gracias a todos y a chatgpt.

jose.roman@fibercli.com

Referencias

https://blog.container-solutions.com/prototyping-on-premises-network-infrastructure-for-kubernetes-clusters-part-4

https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/cisco-nx-os-calico-network-design.html

https://docs.tigera.io/calico/latest/reference/architecture/design/l2-interconnect-fabric

https://brewpackets.blog/2023/02/20/calico-and-metallb-working-together-with-bgp/

https://itnext.io/kubernetes-service-type-lb-for-on-prem-deployments-89e9b2a73a0c