Hola buenas me paso algo similar con virtualbox, me tomaba la interfaz virtual de virtualbox.
Mi Cluster esta en una ip privada de pruebas 192.168.10.0/24. configurada en la interfaz eno1 del host físico (Mi host físico es mi primer nodo y esta como Control Plane).

Con ayuda de Chatgpt encontré que Calico está autodetectando la IP en la interfaz equivocada.

- Está usando vboxnet0 → 192.168.56.1/24, que no es la red que querés (192.168.10.x).
- Por eso en mi pod calico-node-2v8tr en el master (k8s-nodo0) queda en 0/1 READY.

En Chatgpt encontre que se puede pasar la interfaz correcta como variable de entorno en el DaemonSet.

Primero modifique la variable de entorno de **kube_editor** para poder editar con **nano** y no con **vim**.

$ export KUBE_EDITOR="nano"

Con este comando me permite modificar el archivo y definir que interfaz quiero use Calico, forzando la detección de la red.

$ kubectl -n kube-system edit daemonset calico-node

en spec.template.spec.containers.env agregue:

```
- name: IP_AUTODETECTION_METHOD
  value: "interface=eno1"
```

Adjunto una imagen de como quedo en mi caso mas abajo.

Reinicie los pods para que tomen el cambio:

$ kubectl -n kube-system rollout restart daemonset calico-node

Esto es lo que creo que soluciono mi problema con mis primeros nodos.

Todos los nodos de Kubernetes usan la IP de la interfaz correcta (**eno1**) para Calico.
Se eliminó la interferencia de **vboxnet0** en mi caso.

Encontré otra opción que es Configurar kubelet en todos los nodos. Esto si agregas una virtual que este en otro host con virtualbox puede pasar.
En este caso interfiere la interfaz NAT de las vms y se anuncian con ips de su primera interfaz.
Me paso que se anunciaban con el rango 10.0.2.x que es propio de NAT de vbox y no en mi rango 192.168.10.0/24.

En cada nodo:

$ sudo nano /var/lib/kubelet/kubeadm-flags.env

KUBELET_KUBEADM_ARGS="--node-ip=<IP ASIGNADA> --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock --pod-infra-container-image=registry.k8s.io/pause:3.10"

$ sudo systemctl daemon-reload
$ sudo systemctl restart kubelet

En este ultimo caso donde mis nodos están en otro virtualbox, lo mas fácil que encontré fue:

Desactivar la interfaz NAT de las vms de VBox y deje solo la interfaz **bridge** desde la configuración de interfaz de Vbox,a estas les asigno la IP por MAC desde pfsense sin problemas.

Reinicie los nodos con:

$ kubectl -n kube-system rollout restart daemonset calico-node

para que tomaran el cambio y detectaran la interfaz disponible.

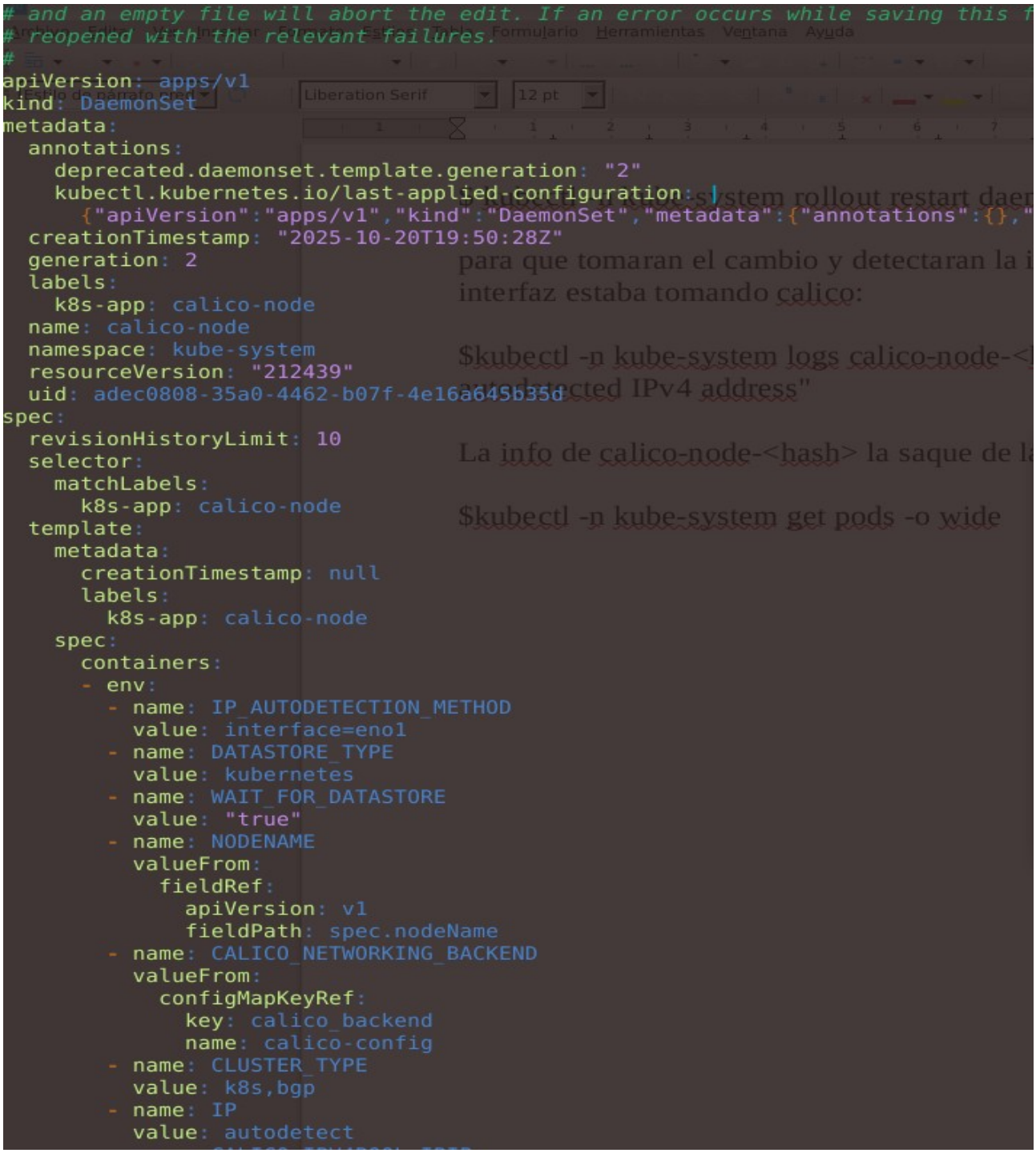
Con este comando verifique en los logs que interfaz estaba tomando Calico:

$kubectl -n kube-system logs calico-node-<hash de calico> -c calico-node | grep "Using autodetected IPv4 address"

La información de calico-node-<hash> la obtuve con este comando:

$kubectl -n kube-system get pods -o wide

Parte del archivo modificado en mi Control Plane para que tome la interfaz correcta de red con Calico (*kubectl -n kube-system edit daemonset calico-node*).



```
# and an empty file will abort the edit. If an error occurs while saving this f
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: DaemonSet
metadata:
  annotations:
    deprecated.daemonset.template.generation: "2"
    kubectl.kubernetes.io/last-applied-configuration:
      {"apiVersion":"apps/v1","kind":"DaemonSet","metadata":{"annotations":{},"
  creationTimestamp: "2025-10-20T19:50:28Z"
  generation: 2
  labels:
    k8s-app: calico-node
  name: calico-node
  namespace: kube-system
  resourceVersion: "212439"
  uid: adec0808-35a0-4462-b07f-4e16a645b35d
spec:
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: calico-node
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: calico-node
    spec:
      containers:
      - env:
        - name: IP_AUTODETECTION_METHOD
          value: interface=eno1
        - name: DATASTORE_TYPE
          value: kubernetes
        - name: WAIT_FOR_DATASTORE
          value: "true"
        - name: NODENAME
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: spec.nodeName
        - name: CALICO_NETWORKING_BACKEND
          valueFrom:
            configMapKeyRef:
              key: calico_backend
              name: calico-config
        - name: CLUSTER_TYPE
          value: k8s,bgp
        - name: IP
          value: autodetect
```

Comparto como quedaron las instalaciones previas de rango del pool de ip **metallb**:

```
cat <<EOF | kubectl apply -f -
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: ippool
  namespace: metallb-system
spec:
  addresses:
  - 192.168.10.251-192.168.10.254
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: l2advertisement
  namespace: metallb-system
EOF
```

Mi IP flotante (**192.168.10.250**) en kuve-vip la configure de interfaz de mi host **eno1.**

El Ingress lo instale de esta forma:

```
helm --install ingress-nginx ingress-nginx/ingress-nginx \
  --version 4.11.8 \
  --namespace ingress-nginx \
  --create-namespace \
  --set controller.service.type=LoadBalancer \
  --set controller.service.loadBalancerIP=192.168.10.251
```

Pobre con la reinstalación y configuración de Metallb por las dudas y volví a configurar los pool de Ips para que me tomara los cambios que había realizado en Calico.

En el **nodo master** (no importa cuál, pero con `kubectl` funcional) y que MetalLB tenga un rango como el definido, por ejemplo en mi caso:

```
$ kubectl get ipaddresspools -n metallb-system

NAME   AUTO          ASSIGN   ADDRESSES
ippool   true        ["192.168.10.251-192.168.10.254"]
```

Resumen de como quedo mi cluster y el flujo:

Componente / Rol	Descripción	Rango / IP asignada	Ejemplo / Observación
Red física (host y VMs)	Red base donde se comunican los nodos (bridge o LAN local)	`192.168.10.0/24`	Red privada real de tu laboratorio
Master (Control Plane)	Nodo principal del cluster Kubernetes	`192.168.10.245`	Hostname: `k8s-master` o `k8s-nodo0`
Workers (nodos)	Nodos que ejecutan Pods y servicios	`192.168.10.246` → `.249`	`k8s-worker1` a `k8s-worker4`
IP flotante (virtual / VIP)	IP común para balanceo interno (keepalived o ingress interno)	`192.168.10.250`	Puede usarse para HAProxy o Ingress
Pool MetalLB (LoadBalancer)	Rango de IPs que MetalLB asignará a servicios tipo LoadBalancer	`192.168.10.251` → `.254`	Asignadas dinámicamente, ej. Ingress-NGINX usa `.251`
Red de Pods (Calico / CNI)	Red interna de Kubernetes para Pods	`172.21.0.0/16`	Tráfico interno pod ↔ pod, no visible fuera del cluster
Red de Servicios (ClusterIP)	Red interna para servicios Kubernetes	`10.96.0.0/16`	Kubernetes reserva IPs para servicios virtuales
DNS del cluster	Dominio interno o externo para el cluster	`k8stest.unpa.edu.ar`	Usado para resolver nombres Ingress
IP de Ingress (NGINX)	IP asignada por MetalLB al Ingress Controller	`192.168.10.251`	Primera IP del pool MetalLB
Cert-Manager (ClusterIssuer)	Servicio para emitir certificados SSL	Usa Ingress y DNS del cluster	Certificados para `*.k8stest.unpa.edu.ar`
API Server (Kubernetes)	Interfaz de control del cluster	`192.168.10.245:6443`	Acceso administrativo (kubeadm, kubectl)