CLOUD NATIVE
COMPUTING
FOUNDATION

Kubernetes Networking for ‘Stackers

Karthik Prabhakar
Tigera
kp@tigera.io

Today

1. How did we get here? (a.k.a. Neutron's Design Thinking)
2. Where are we headed? (a.k.a. Kubernetes Network Design Thinking)
3. Kubernetes Network concepts
4. Live Demo: Kubernetes networking can be simple

Tomorrow (Wed): Hynes Level 3 - Ballroom B - 4:30pm

A. Simple, Scalable Approaches to deploy Kubernetes with OpenStack
   a. Kubernetes alongside OpenStack
   b. Kubernetes in OpenStack
   c. OpenStack in Kubernetes

B. AT&T Overview & Demo: Deploying OpenStack on Kubernetes (with OpenStack-Helm and Calico)

Remnants from the early days of VM Networking

=> Conflate isolation with network topology

=> Complex SDN controllers to layer overlay spaghetti onto physical infra

=> Random mesh of bridges, vswitches, tunnels, L3 backhauls, and security enforcement points

=> Coarse security group rules decoupled from dynamic workloads

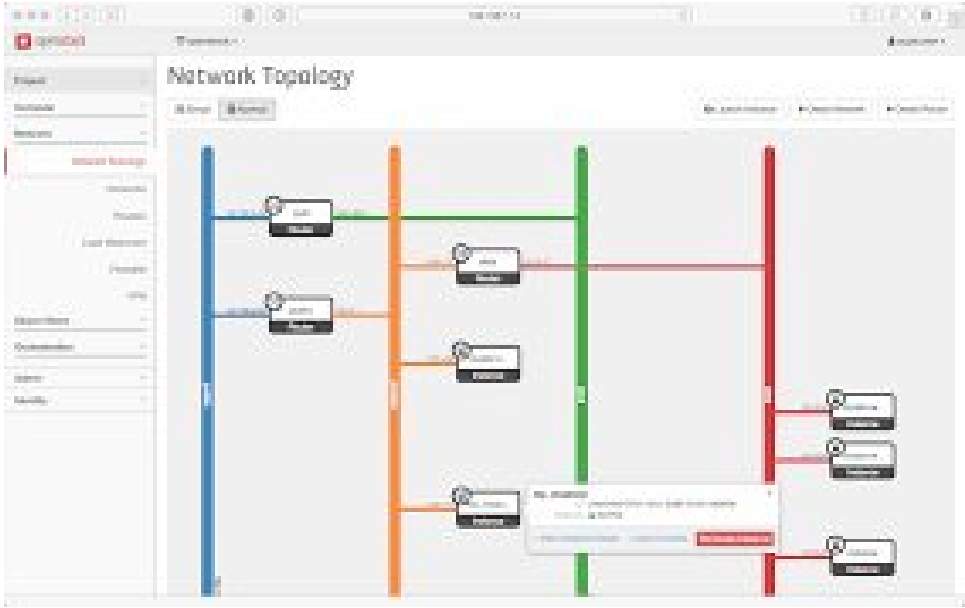


Exhibit A: Neutron with OpenvSwitch (L2)

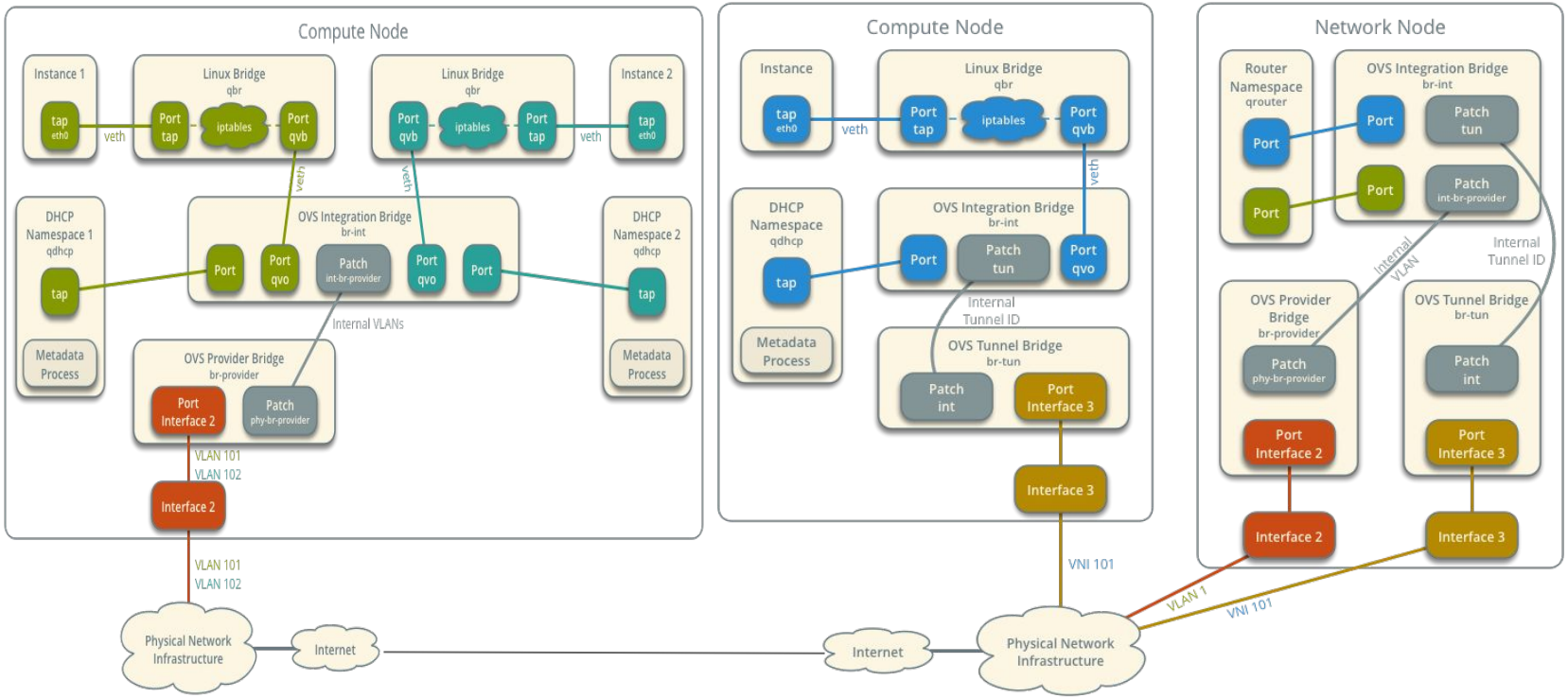
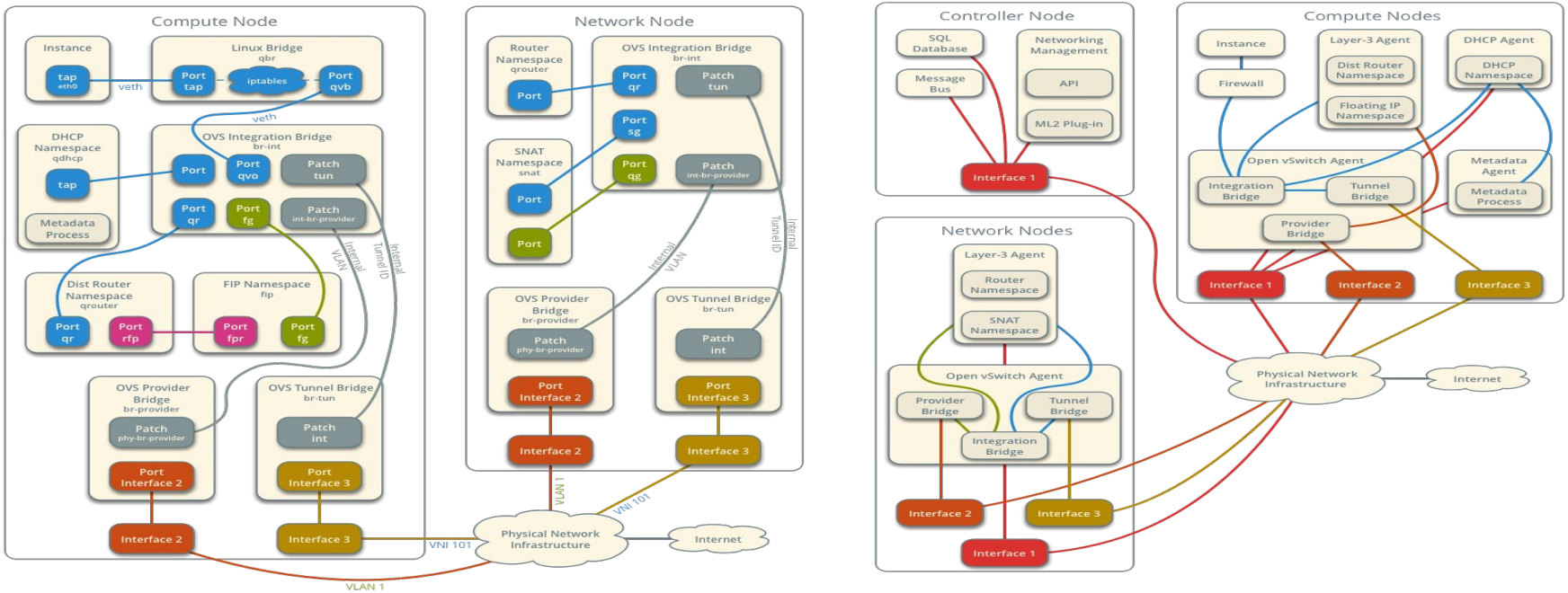
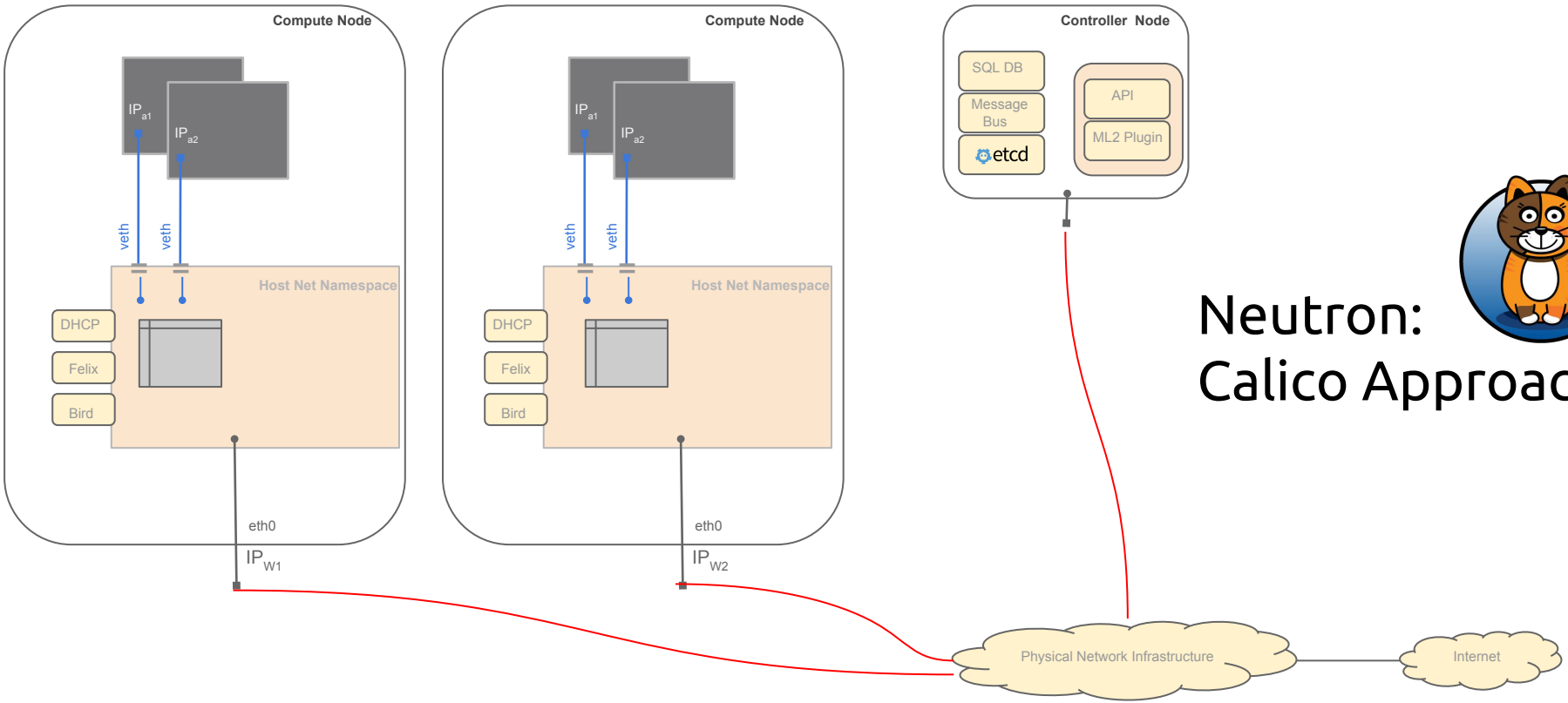


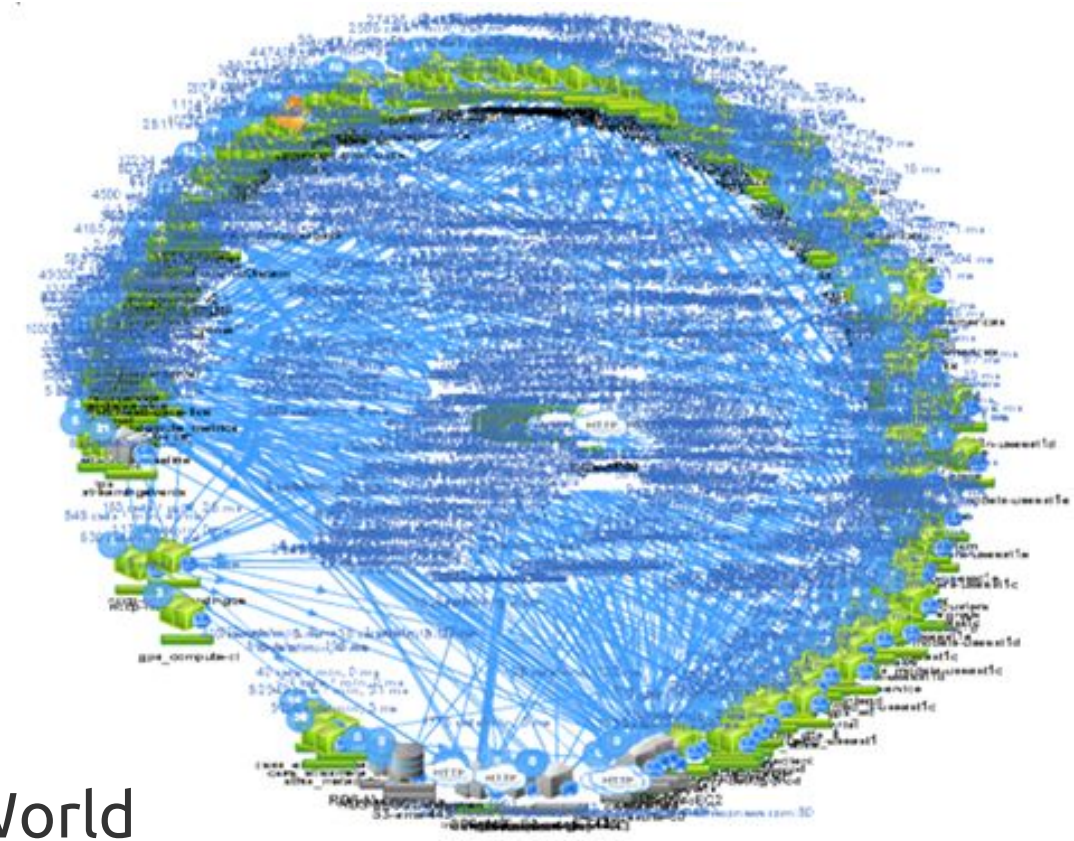
Exhibit B: Neutron with OpenvSwitch and DVR/VRRP




"More Overlays Please"
said no SRE ever



Neutron: Calico Approach



The Dynamic World of Microservices

NETFLIX

The Kubernetes Approach

=> Assume **IP connectivity** between nodes within cluster

=> **IP per Pod**

=> **CNI** plugin abstraction enabling choice of network plugin for endpoint traffic

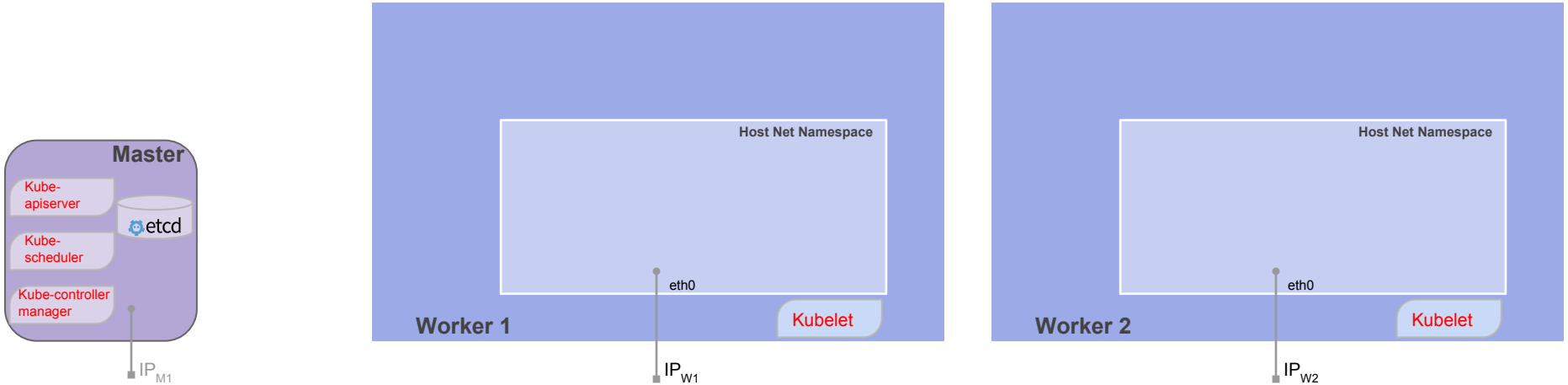
=> Declarative isolation using rich **Network Policy** constructs

=> "**Services**" map to pods dynamically

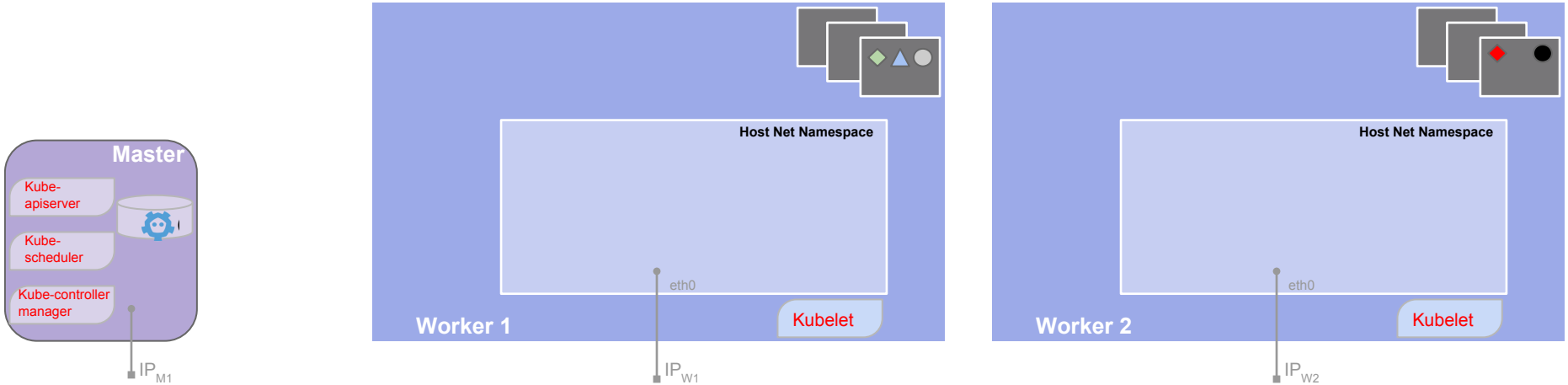
=> Optional **Ingress** controllers and resources

=> Choice of **Service Discovery** options, including KubeDns

The Kubernetes Approach: Nodes

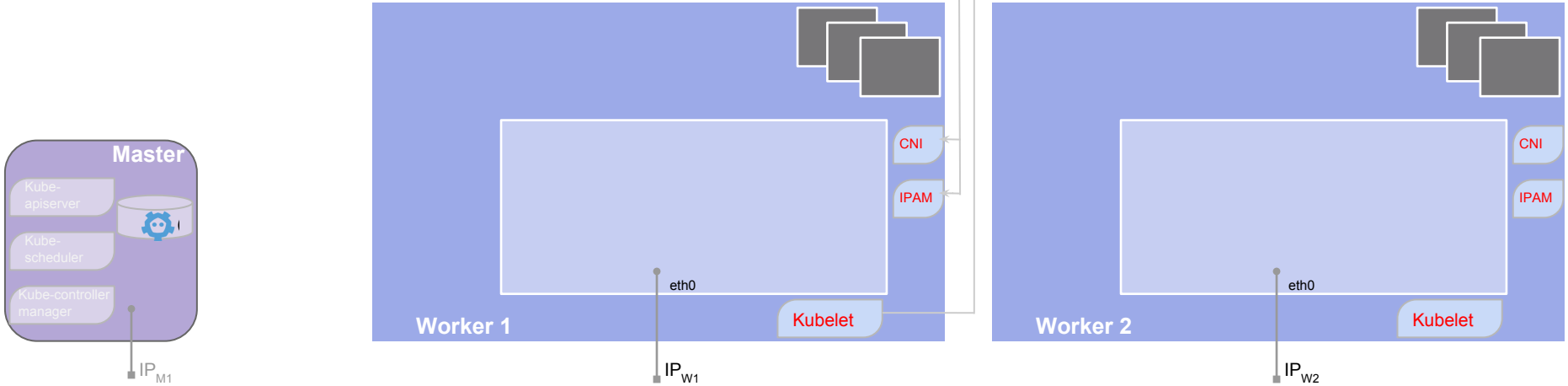


The Kubernetes Approach: Pods



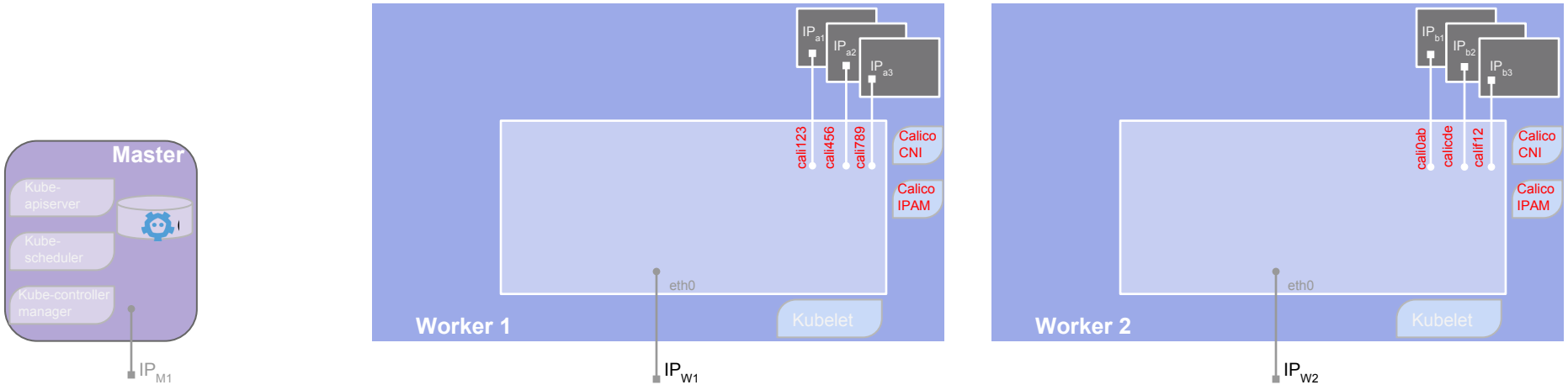
The Kubernetes Approach:
CNI and IPAM

```
cat >/etc/cni/net.d/10-calico.conf <<EOF
{
    "name": "calico-k8s-network",
    "type": "calico",
    "etcd_endpoints": "http://<ETCD_IP>:<ETCD_PORT>",
    "log_level": "info",
    "ipam": {
        "type": "calico-ipam"
    },
    "policy": {
        "type": "k8s"
    },
    "kubernetes": {
        "kubeconfig": "</PATH/TO/KUBECONFIG>"
    }
}
EOF
```



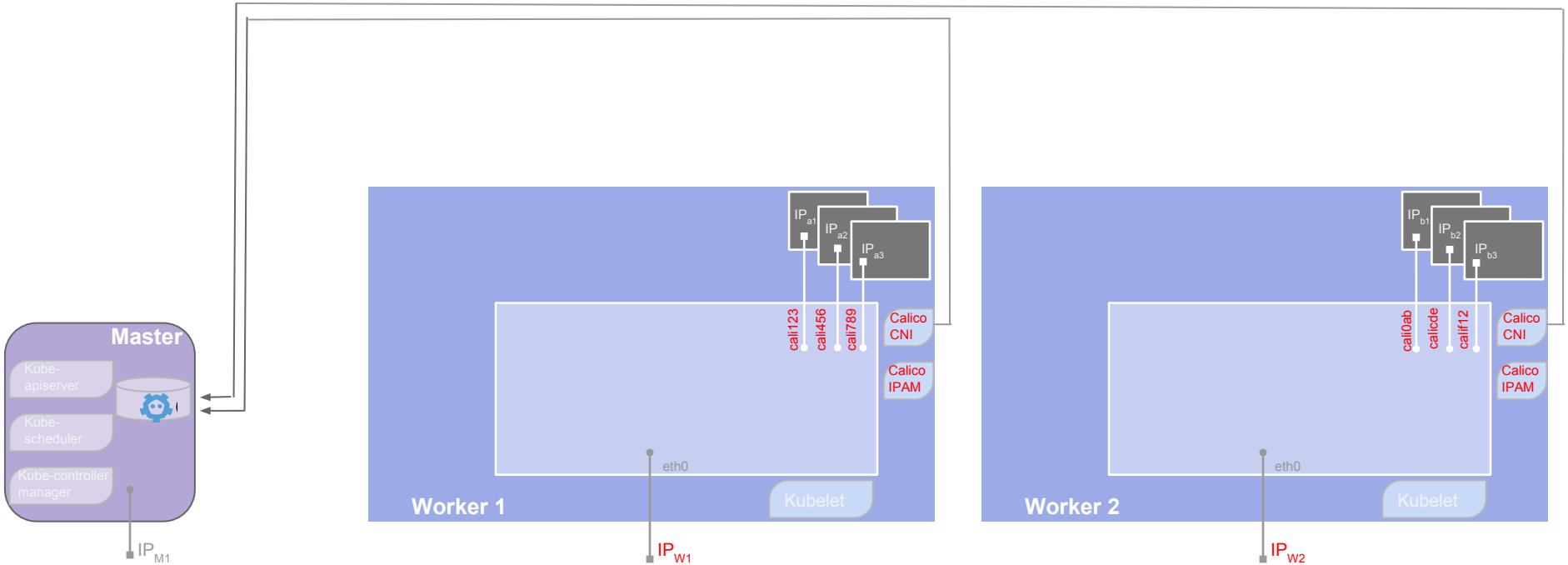
The Kubernetes Approach: Calico

(Pod-Pod Networking)



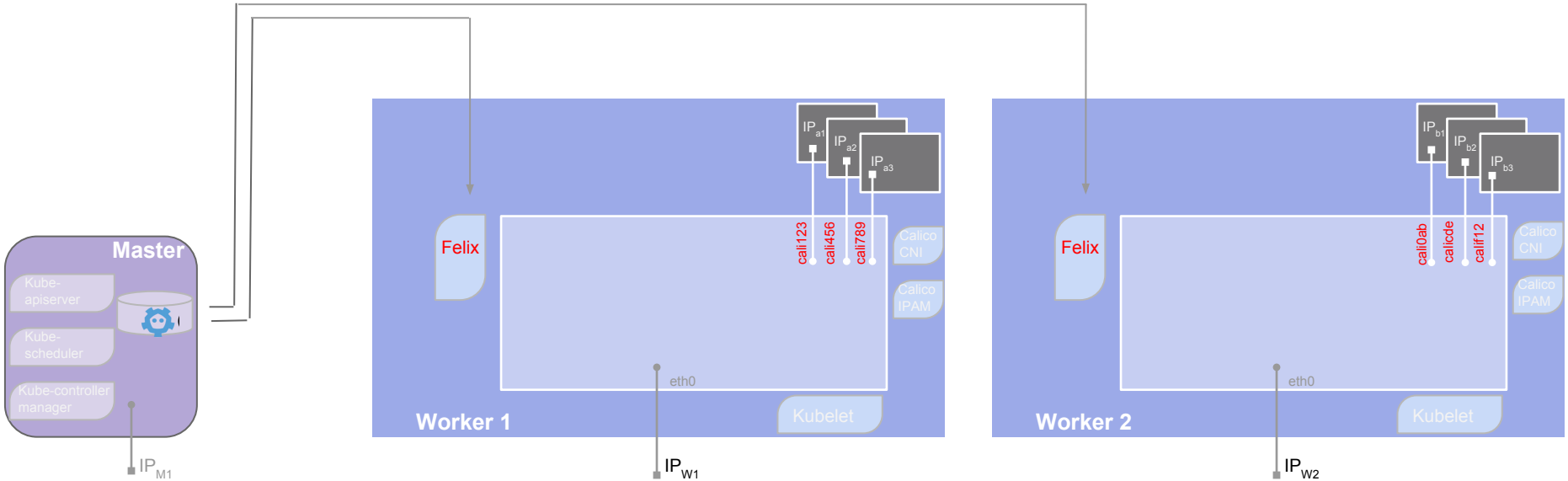
The Kubernetes Approach: Calico

(Pod-Pod Networking)



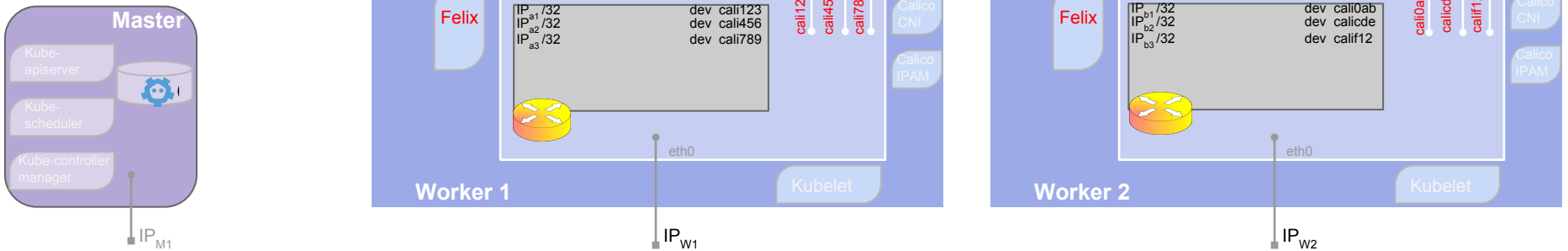
The Kubernetes Approach: Calico

(Pod-Pod Networking)



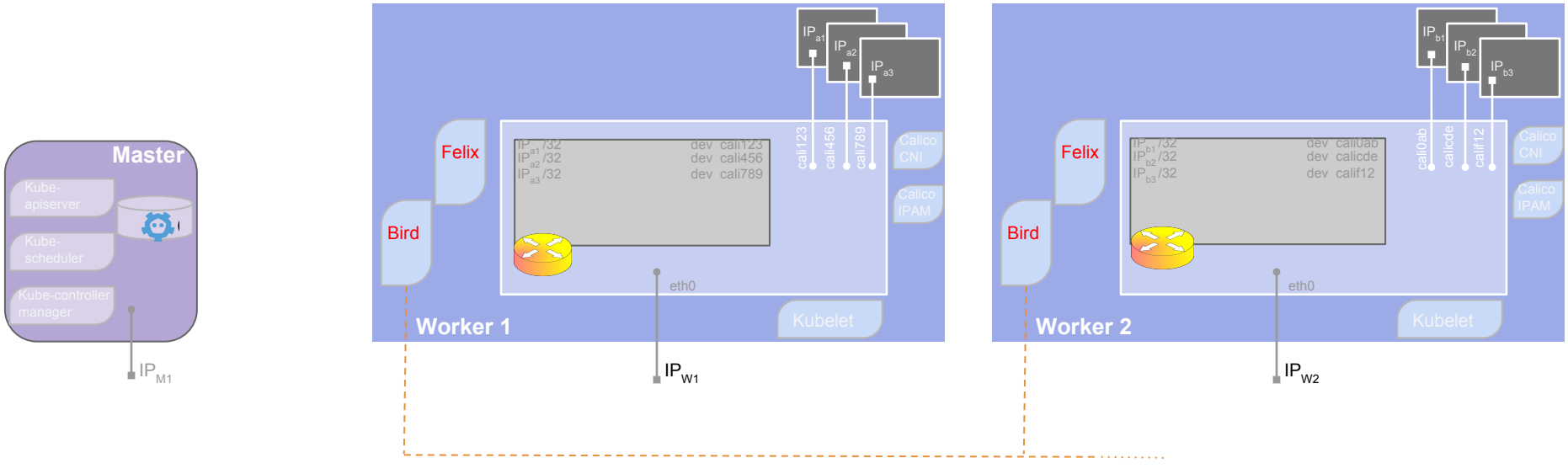
The Kubernetes Approach: Calico

(Pod-Pod Networking)



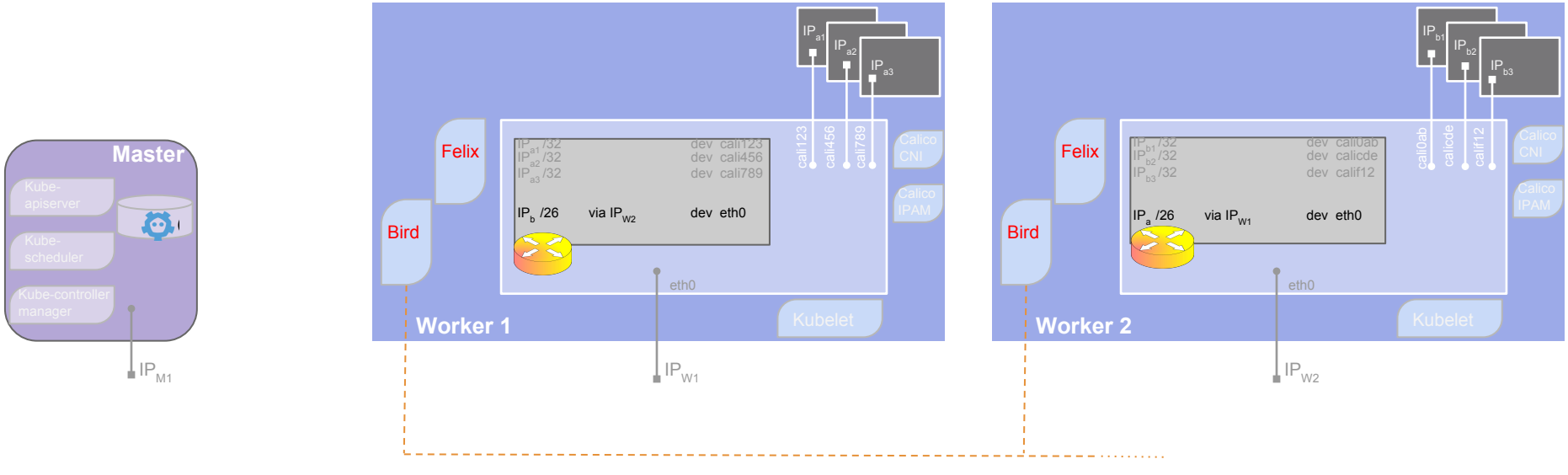
The Kubernetes Approach: Calico

(Pod-Pod Networking)



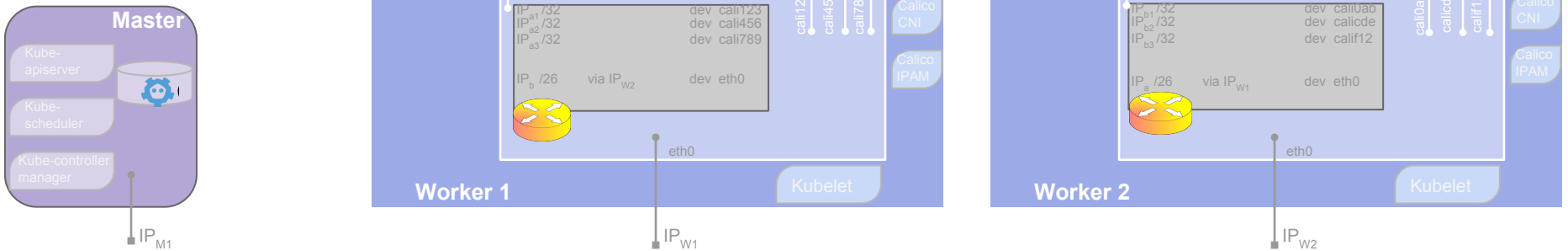
The Kubernetes Approach: Calico

(Pod-Pod Networking)



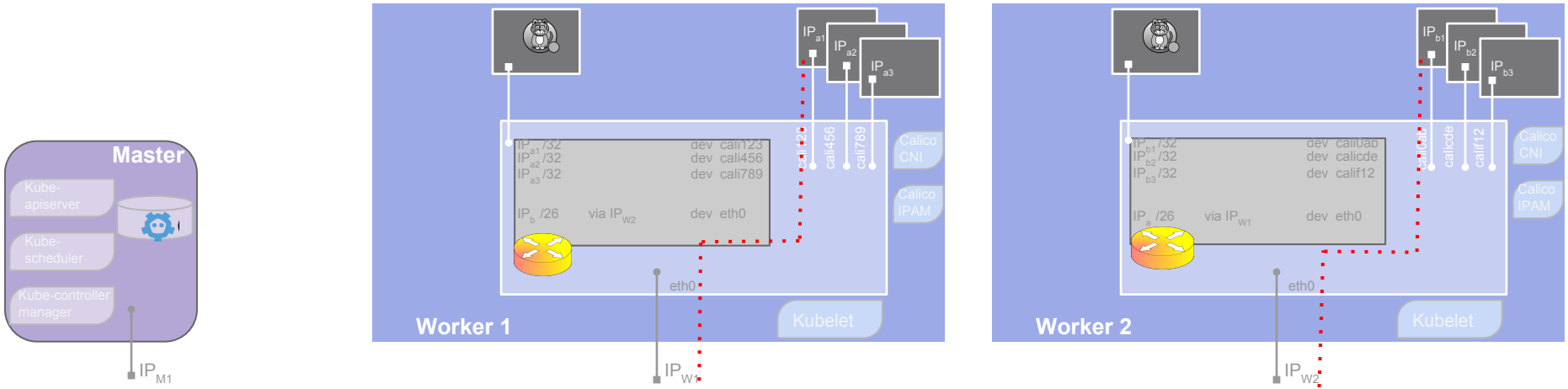
The Kubernetes Approach: Calico

(Pod-Pod Networking)



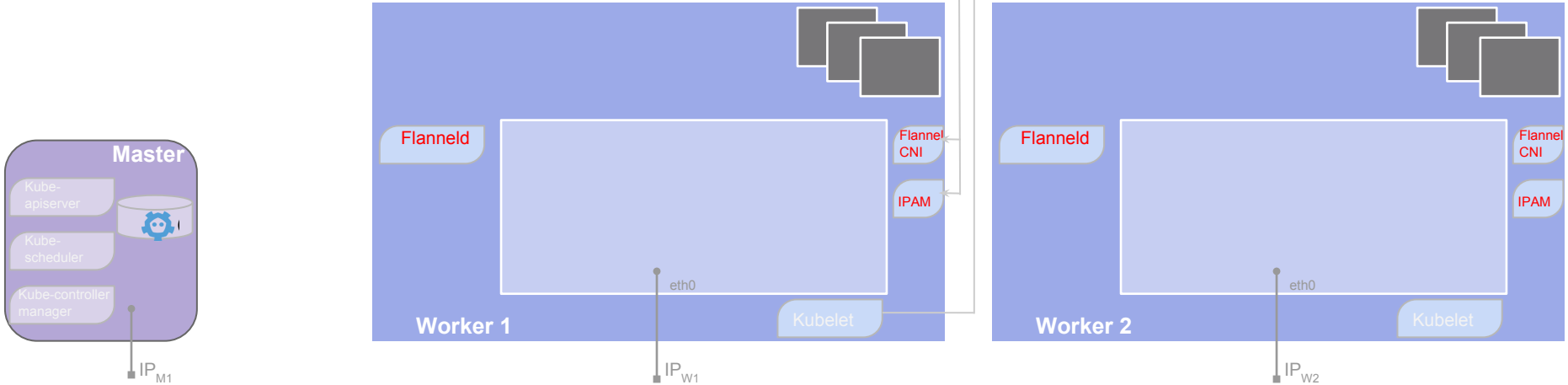
The Kubernetes Approach: Calico

(Pod-Pod Networking)

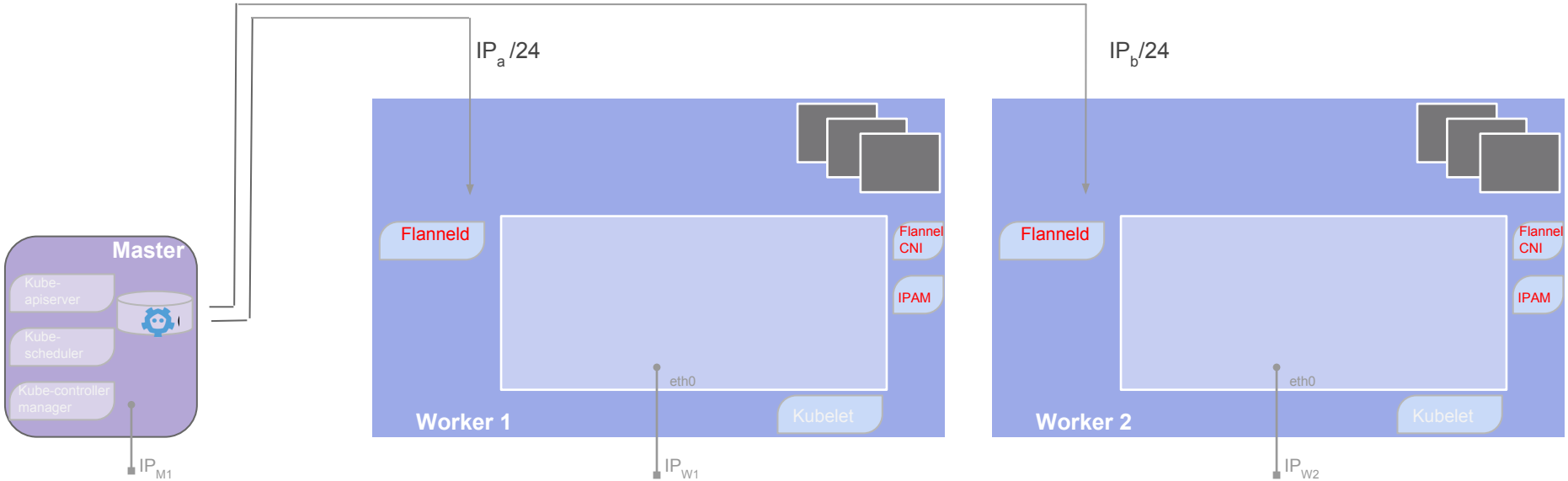


The Kubernetes Approach: flannel
(Pod-Pod Networking)

```
{
    "name": "mynet",
    "type": "bridge",
    "mtu": 1472,
    "ipMasq": false,
    "isGateway": true,
    "ipam": {
            "type": "host-local",
            "subnet": "10.1.17.0/24"
    }
}
```



The Kubernetes Approach: flannel
(Pod-Pod Networking)

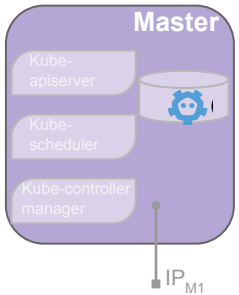


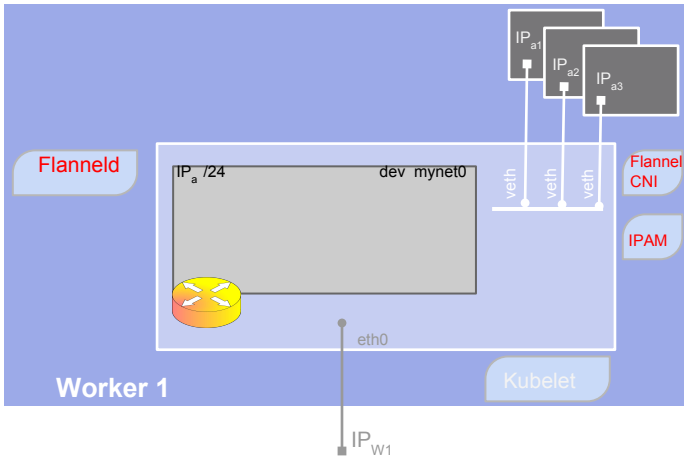
The Kubernetes Approach: flannel

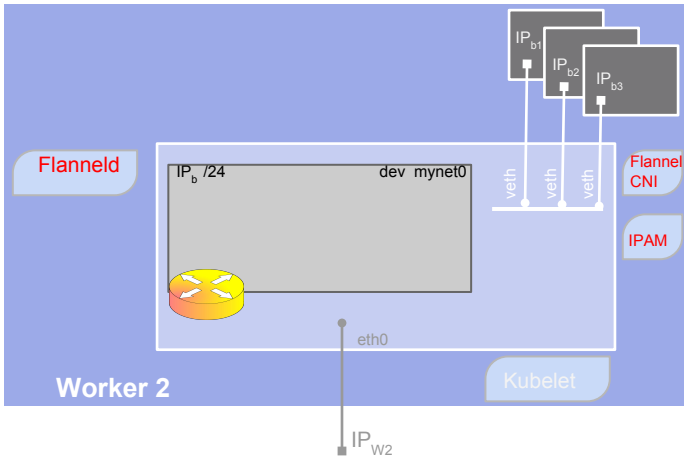
(Pod-Pod Networking)

Flannel Backends:

- UDP
- VXLAN
- Host-gateway
- AWS-VPC
- GCE
- Ali-VPC
- Alloc

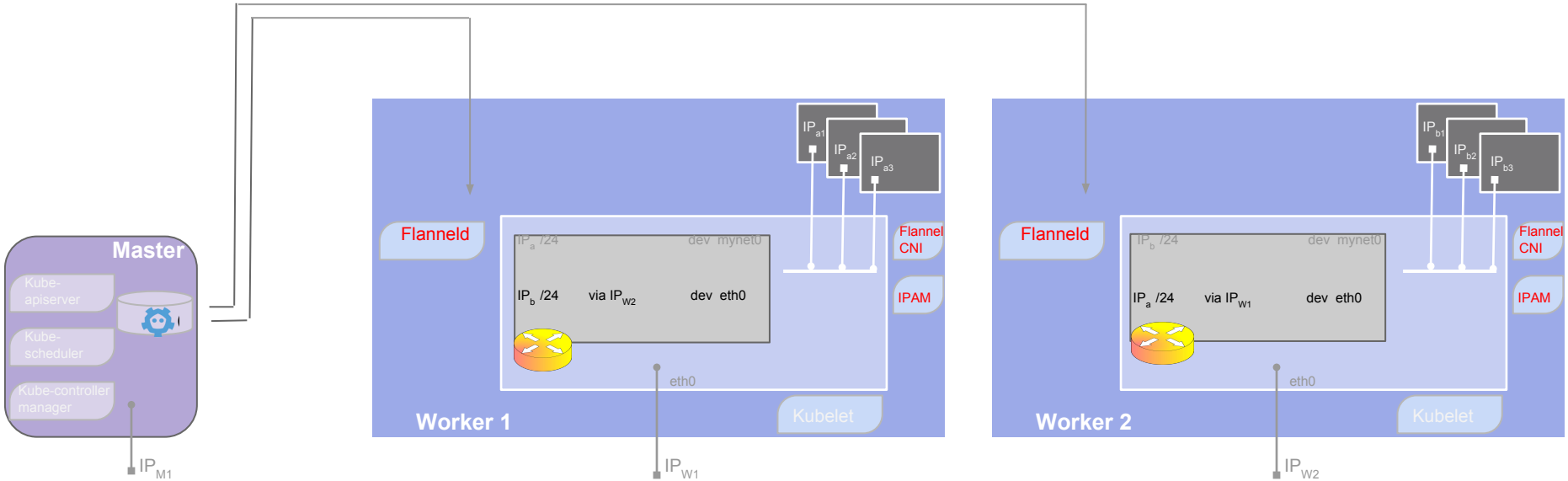






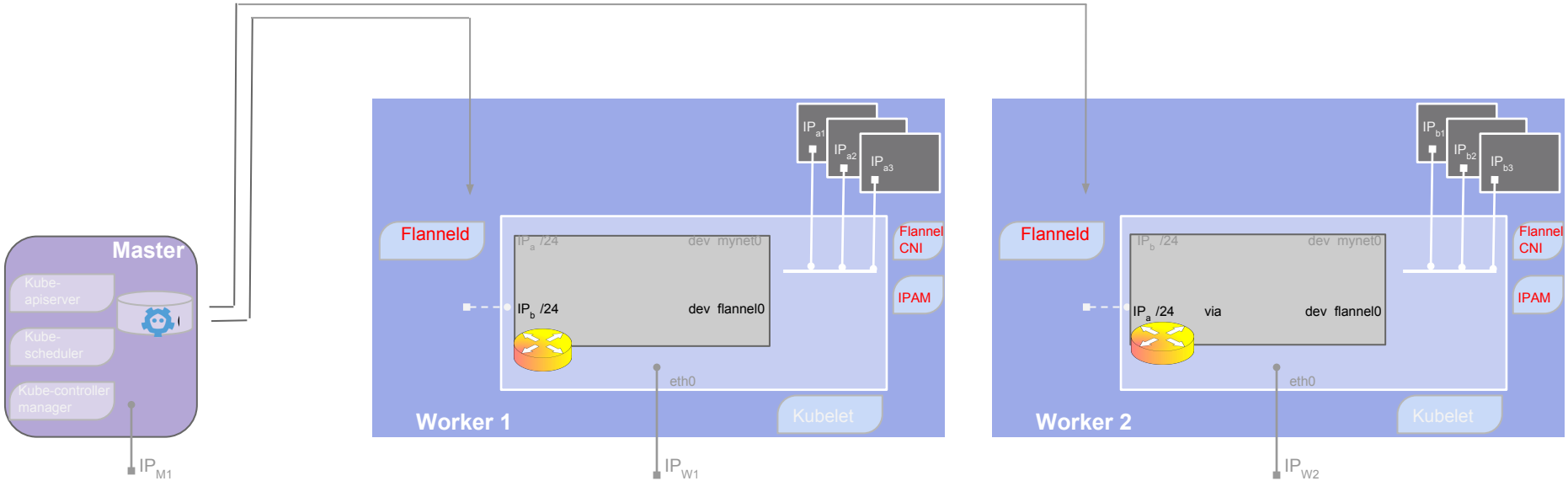
The Kubernetes Approach: flannel: Host Gateway mode

(Pod-Pod Networking)



The Kubernetes Approach: flannel: VXLAN mode

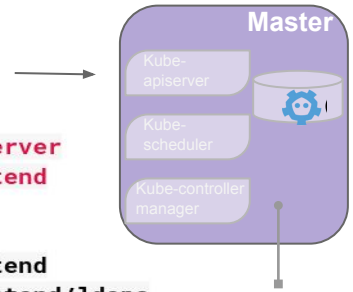
(Pod-Pod Networking)

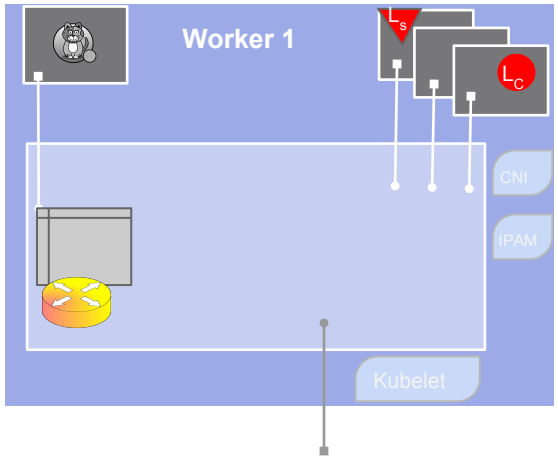


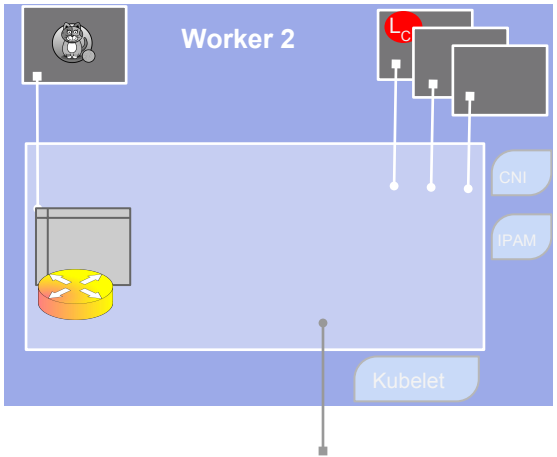
The Kubernetes Approach: Network Policy

Namespaces, Labels and Selectors

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: frontend
  namespace: ProjectRed
spec:
  replicas: 2
  template:
    metadata:
      labels:
        tier: QE
        app: ldapserver
        role: frontend
spec:
      containers:
      - name: frontend
        image: frontend/ldaps
```

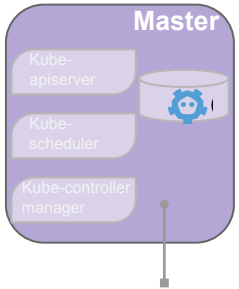


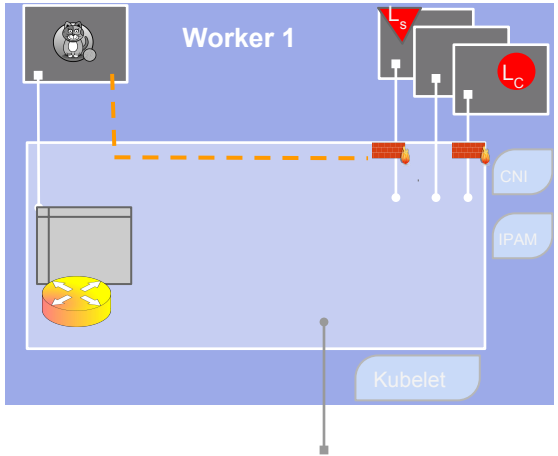


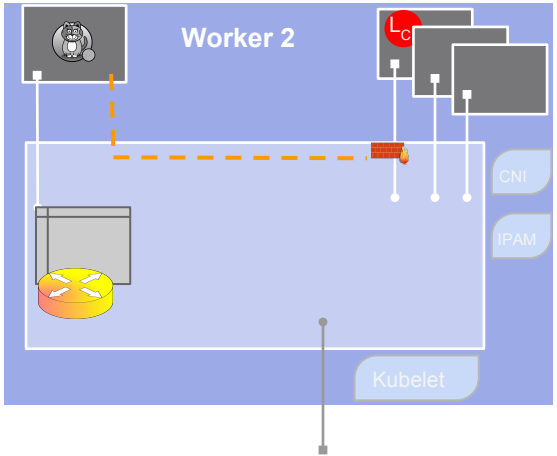


The Kubernetes Approach: Network Policy

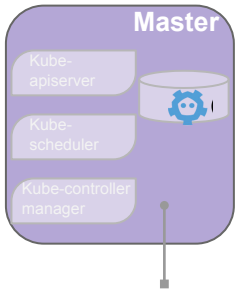
```
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  namespace: ProjectRed
  name: ldap-frontend
spec:
  podSelector:
    matchLabels:
      tier: QE
      role: frontend
      app: ldapserver
    - from:
  ingress:
    - from:
        - podSelector:
            matchLabels:
              tier: QE
              role: frontend
              app: ldapclient
      ports:
        - protocol: TCP
          port: 636
```

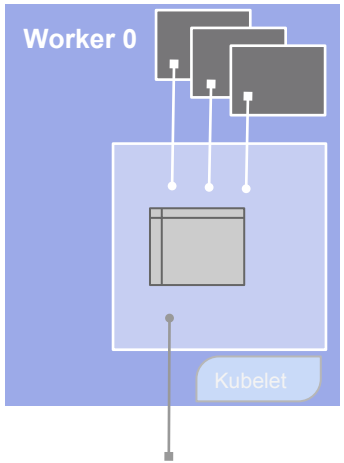


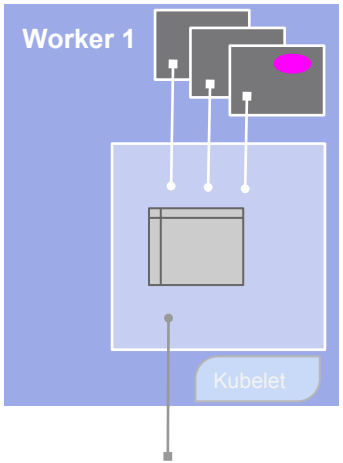


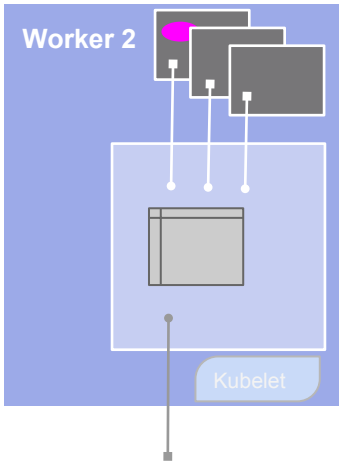


The Kubernetes Approach: Services



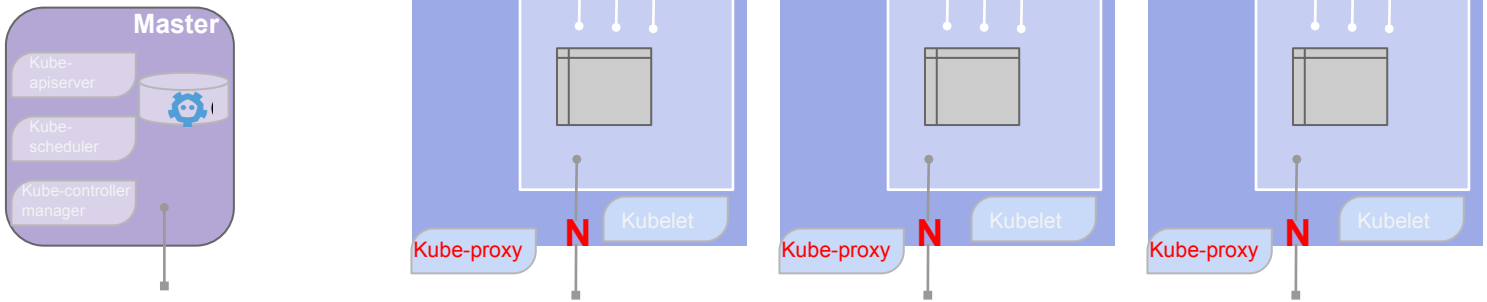






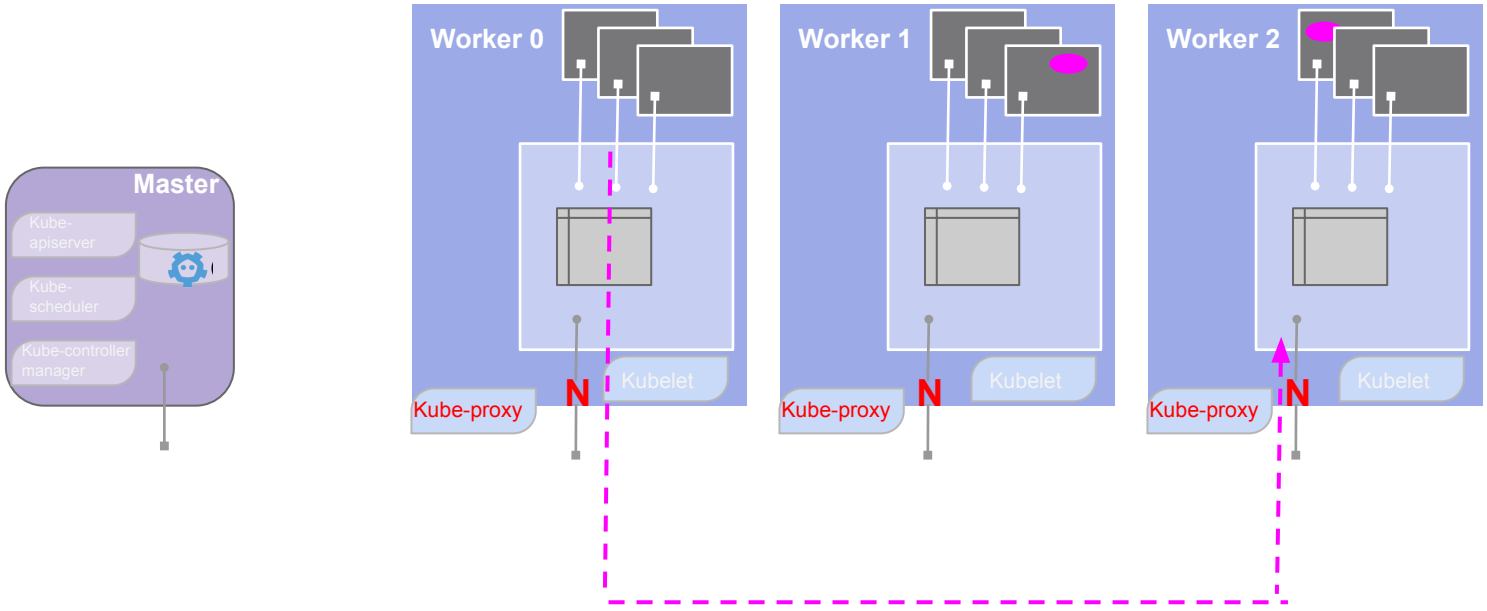

```
kubectl expose --namespace=ProjectPink deployment nginx --port=80
```

The Kubernetes Approach: Services



The Kubernetes Approach: Services

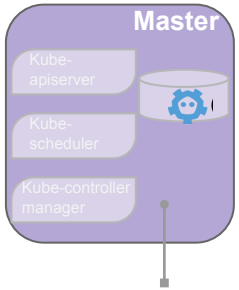
Cluster IP

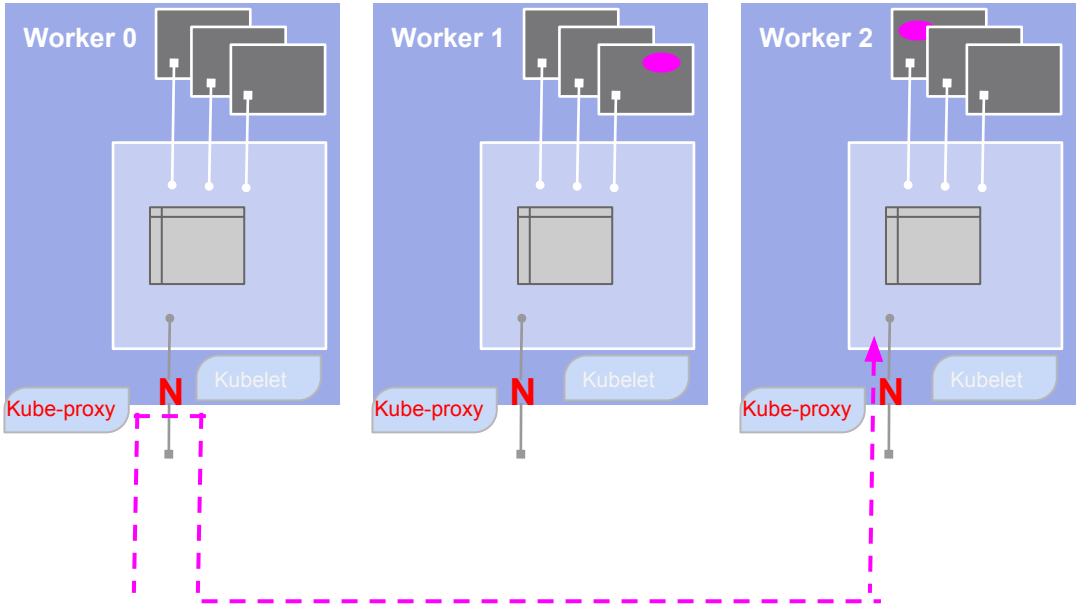


The Kubernetes Approach: Services

Node Port

```
apiVersion: v1
 kind: Service
 metadata:
  name: nginx-service
  labels:
    app: nginx
 spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30080
  selector:
    app: nginx
```

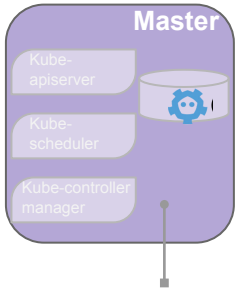


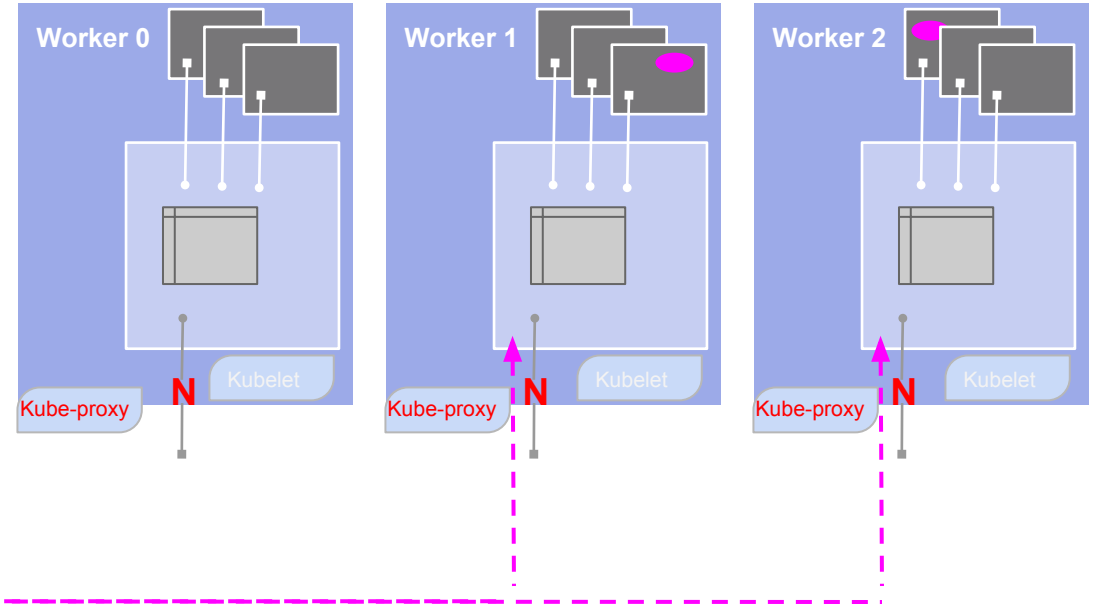


The Kubernetes Approach: Services

External Load Balancers

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
      nodePort: 30061
  clusterIP: 10.0.171.239
  type: LoadBalancer
  loadBalancerIP: 78.11.24.19
status:
  loadBalancer:
    ingress:
      - ip: 146.148.47.155
```



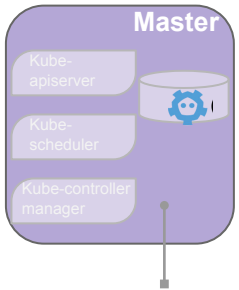


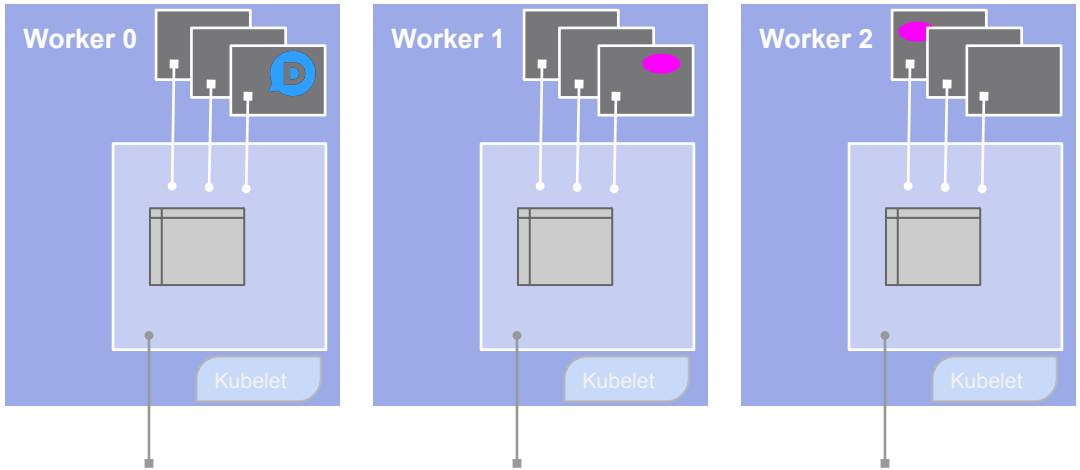
The Kubernetes Approach: KubeDNS

Naming and Service Discovery

```
apiVersion: v1
kind: Service
metadata:
  name: webserver
  namespace: project-red
spec:
  selector:
    name: nginx
    tier: frontend
  clusterIP: None
```

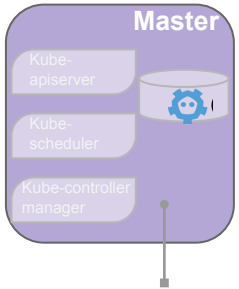
⟹ webserver.project-red.svc.cluster.local

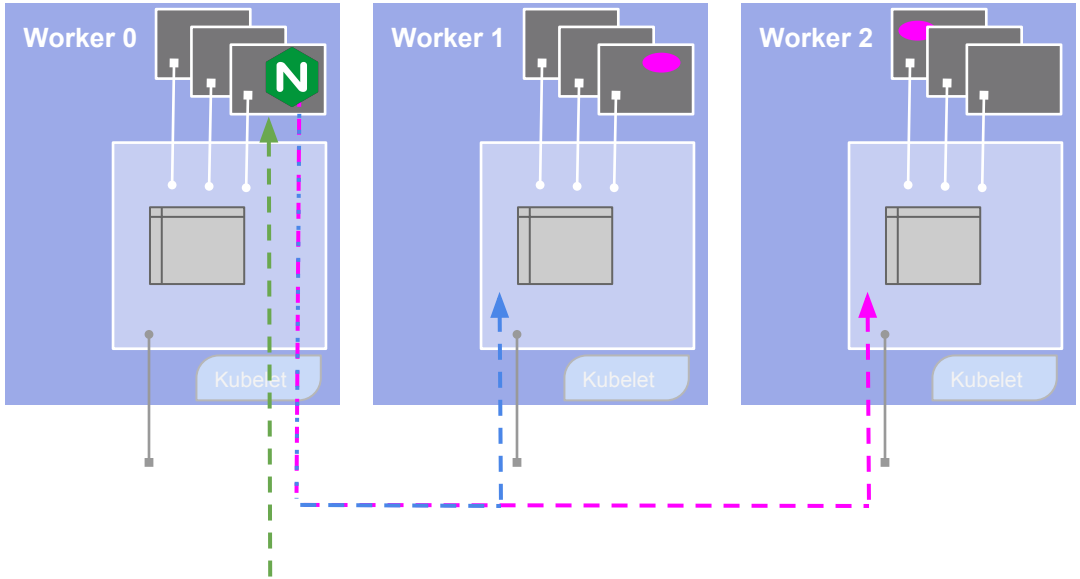




The Kubernetes Approach: Ingress Resources & Controllers

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: s1
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: s2
          servicePort: 80
```



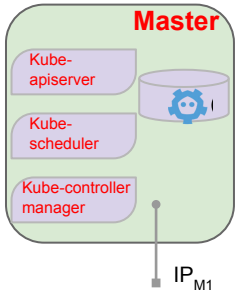


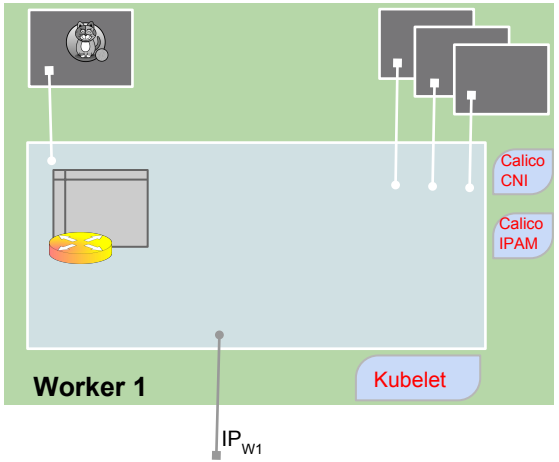
Demo: Self-hosted Kube+Calico Cluster with *kubeadm*

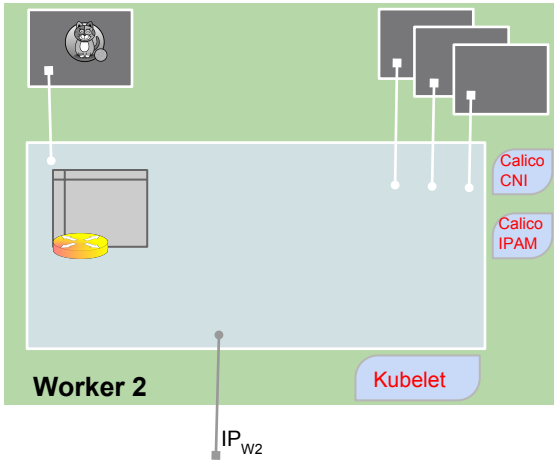
Step 1: *kubeadm init* (on master)

Step 2: *kubectl apply -f calico.yaml*

Step 3: *kubeadm join* (on each worker)

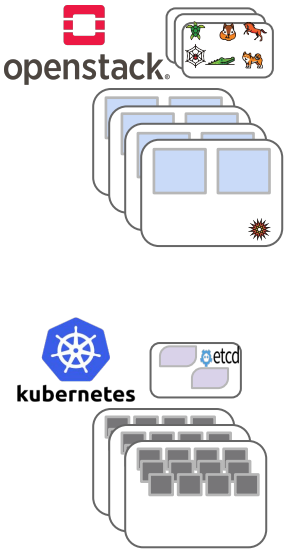




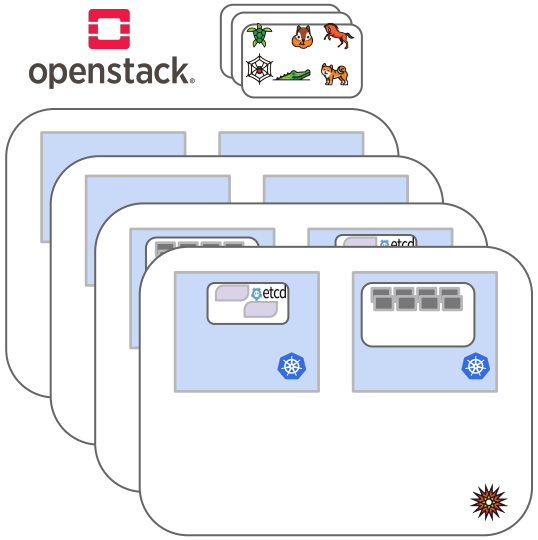


Common Network Operations Across K8s & OpenStack

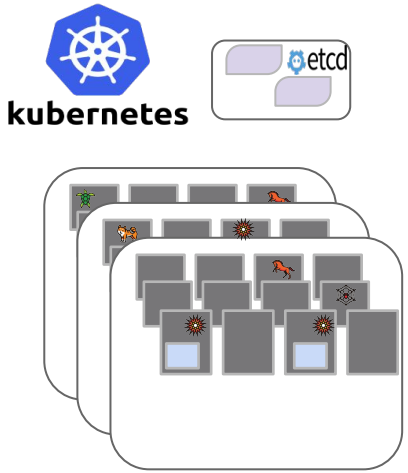
Tomorrow (Wed): Hynes Level 3 - Ballroom B - 4:30pm



Kubernetes alongside OpenStack



Kubernetes in OpenStack



OpenStack in Kubernetes

Takeaways

Networking at VM-scale != Networking at Container Scale

Takeaways

Kubernetes Network Abstractions are different from Neutron

Takeaways

Keep network simple/scalable, Network Policy for isolation