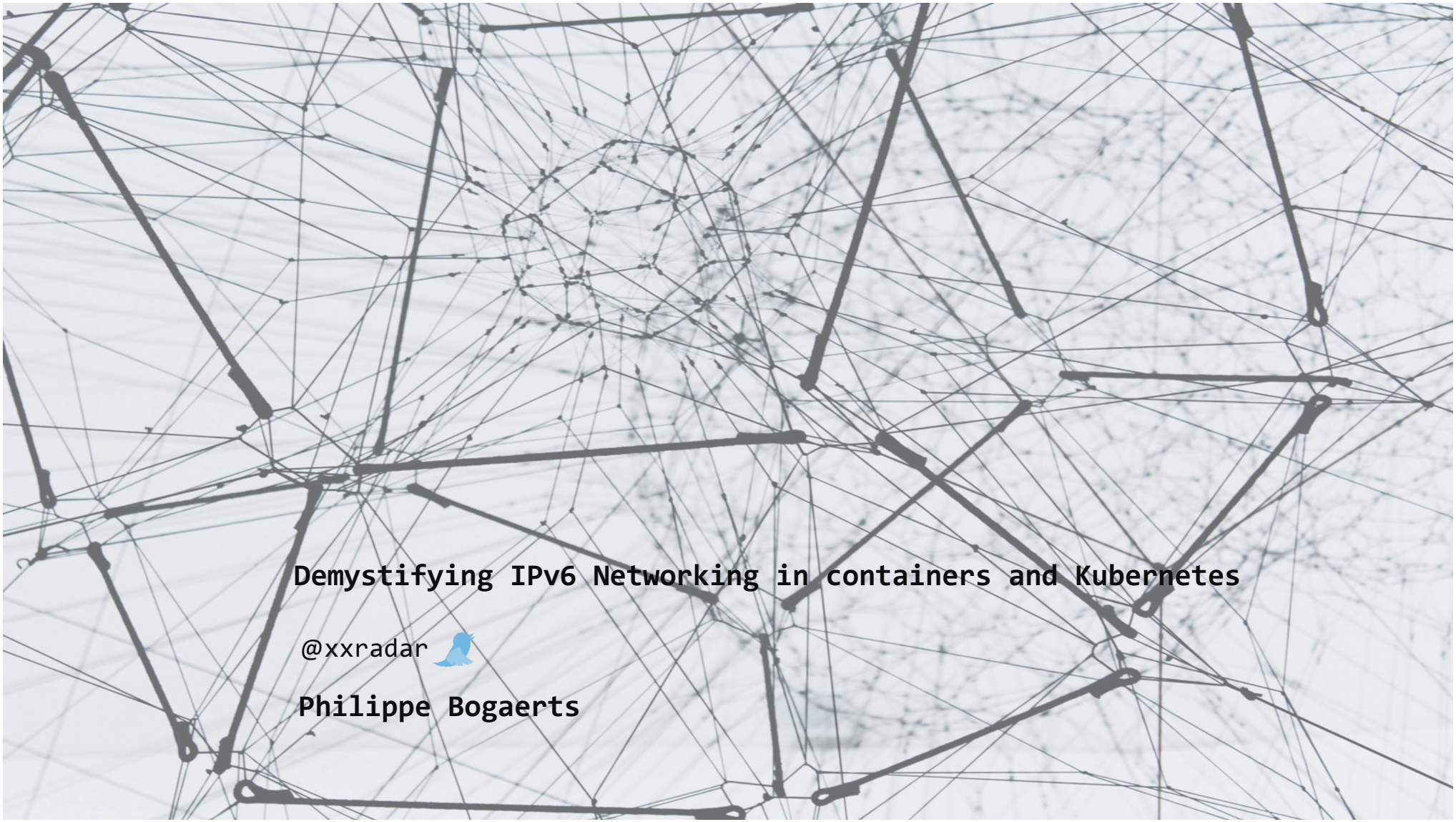

Demystifying IPv6 Networking in containers and Kubernetes
@xxradar
Philippe Bogaerts

IPv6 experience

- IPv6 training (anno 1998)
  - “Ipv6 over encrypted USB-3 to connect home automation ☺”
  - WIFI was hardly available
  - Mobile (GPRS anno 2001)
- Some tutorials (anno 2001)
  - https://www.radarhack.com/tutorial/ipv6.pdf
  - https://www.radarhack.com/tutorial/ipv6_2.pdf
- World IPv6 Launch (2011 – 2012)

Connecting to the web server via netcat6.

```
[root@localhost root]# nc6 -6 -n 3ffe::3 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 04 Nov 2003 19:16:44 GMT
Server: Apache/2.0.47 (Win32)
Content-Location: index.html.en
Vary: negotiate,accept-language,accept-charset
TCN: choice
Last-Modified: Thu, 03 May 2001 16:01:18 GMT
ETag: "fbb3-5d6-8ba74f80;fbcf-9dc-d0c58f00"
Accept-Ranges: bytes
Content-Length: 1494
Connection: close
```

whoami

- Public Cloud Consultant System Engineer EMEA
- Co-founder and co-organizer https://brucon.org
- Training and pen-testing https://kubiosec.tech/

Breaking Stuff as a Hobby | Cloud Native Stuff | DevSecOps | Network and Application security | Container and K8S security | K8s Networking | Security Advocate & Research | Low and slow BBQ | Cocktails

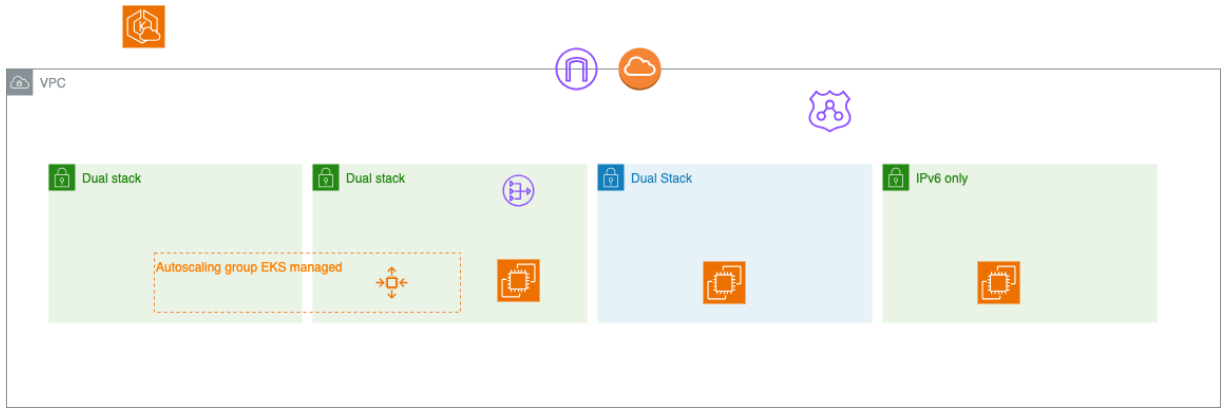
https://www.linkedin.com/in/philippebogaerts/ @xxradar

My IPv6 testing environment

[opinions expressed are solely my own]

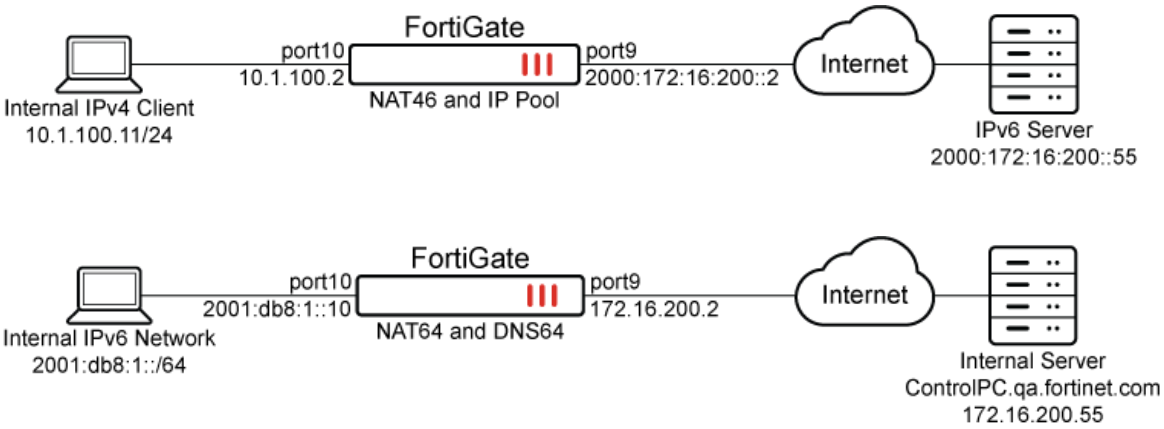
My IPv6 lab

- AWS
  - IPv4 and 6 dual stack
  - IPv6 only
  - DNS64/NAT64
  - Egress only gateway

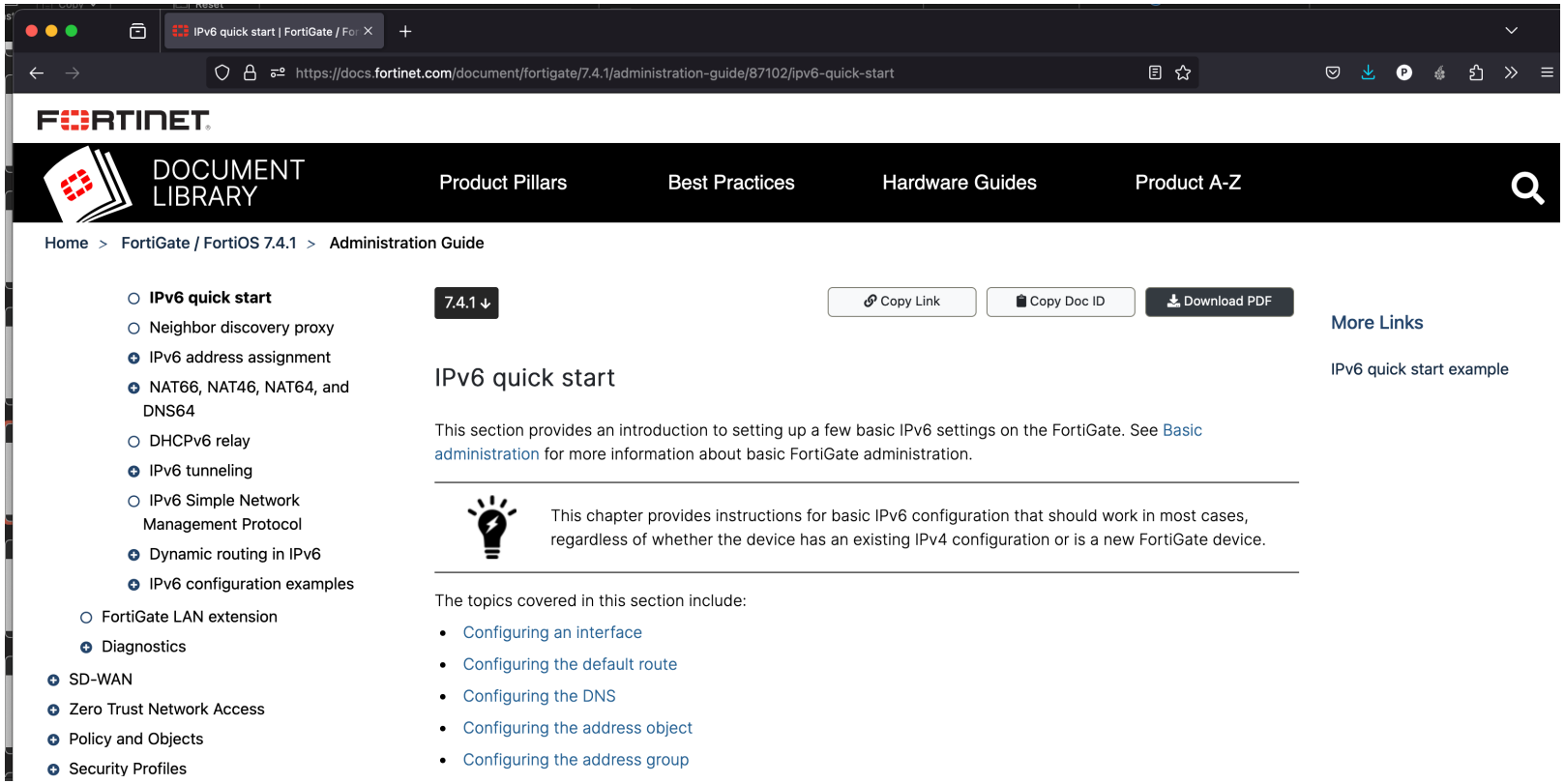


My IPv6 lab

- GCP
- AWS
- DigitalOcean
- Azure



Fortinet


IPv6 quick start | FortiGate / For
https://docs.fortinet.com/document/fortigate/7.4.1/administration-guide/87102/ipv6-quick-start
FORTINET
DOCUMENT LIBRARY
Product Pillars
Best Practices
Hardware Guides
Product A-Z
Home > FortiGate / FortiOS 7.4.1 > Administration Guide
IPv6 quick start
Neighbor discovery proxy
IPv6 address assignment
NAT66, NAT46, NAT64, and DNS64
DHCPv6 relay
IPv6 tunneling
IPv6 Simple Network Management Protocol
Dynamic routing in IPv6
IPv6 configuration examples
FortiGate LAN extension
Diagnostics
SD-WAN
Zero Trust Network Access
Policy and Objects
Security Profiles
7.4.1
Copy Link
Copy Doc ID
Download PDF
IPv6 quick start
This section provides an introduction to setting up a few basic IPv6 settings on the FortiGate. See Basic administration for more information about basic FortiGate administration.
This chapter provides instructions for basic IPv6 configuration that should work in most cases, regardless of whether the device has an existing IPv4 configuration or is a new FortiGate device.
The topics covered in this section include:
Configuring an interface
Configuring the default route
Configuring the DNS
Configuring the address object
Configuring the address group
More Links
IPv6 quick start example

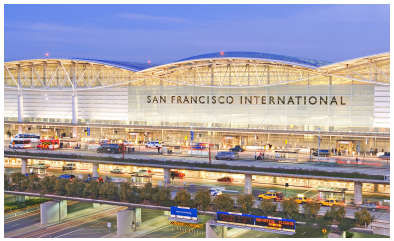
IPv6 and containers

[opinions expressed are solely my own]

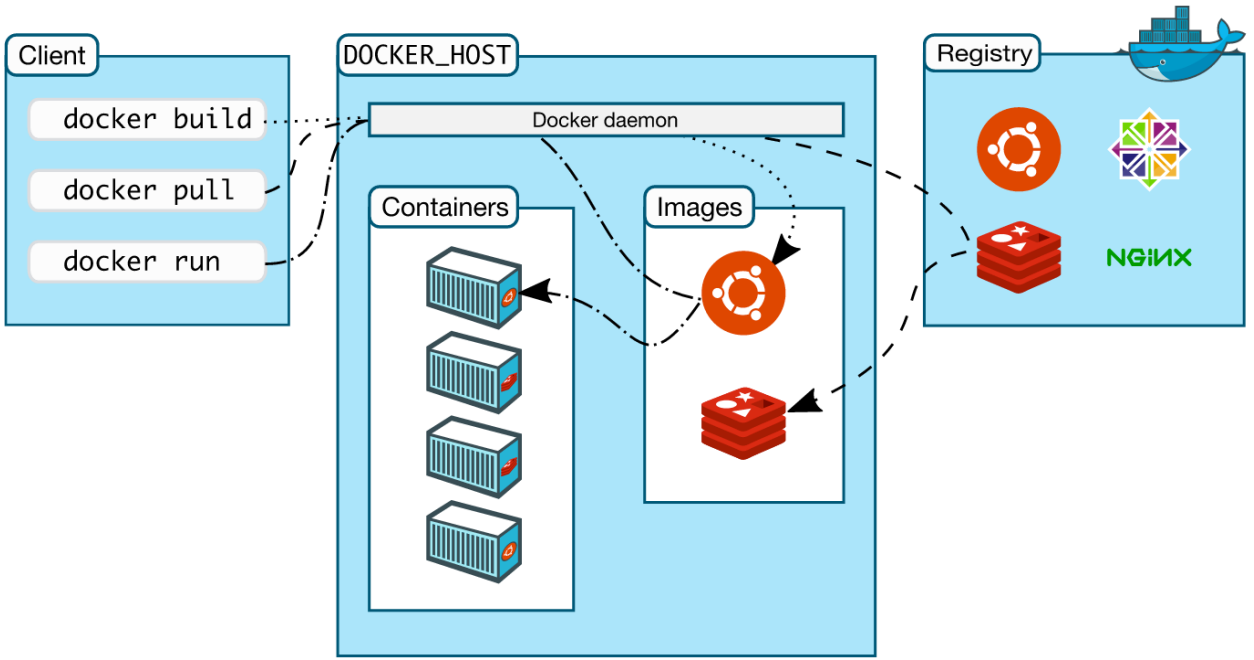
Why are containers so popular ?

During a Wednesday back in 2016 in SFO during booth duty …
Containers ?? Don't know anything about it … what am I doing here ??

The next Friday evening in SFO airport while waiting for a plane back home,
I googled 'docker', installed docker on my MacBook and "build, ship and run" my first container …
and then I boarded the plane …



How docker works



Container support IPv6

- IPv6 is only supported on Docker daemons running on Linux hosts
  - https://docs.docker.com/config/daemon/ipv6/
- Podman
  - https://developers.redhat.com/articles/2022/08/10/how-conifgure-podman-40-ipv6

Registry support

- Dockerhub (ipv6 support)
- Quay (ipv6 support)
- GHCR (afaik no ipv6 support)
  - Needs NAT64 / DNS64
- Google Artifact Registry
- ECR
- ACR
- …

How to enable IPv6 on Docker (1)

- IPv6 “masquerade”

```
# cat /etc/docker/daemon.json
{
"experimental": true,
"ip6tables": true
}
```

- Accessible on the host ipv6 address

  *AWS internet vs. egress-only gateway / firewalls …

- Port mapping required

```
# sudo docker network create --ipv6 --subnet 2001:0DB8::/112 ip6net
# sudo docker run –d –net ip6net –p 80:80 –name www nginx
```

How to enable IPv6 on Docker (2)

- IPv6 native (default bridge)

```
# cat /etc/docker/daemon.json
{
"ipv6": true,
"fixed-cidr-v6": "2a03:b0c0:2:d0::1010:8001/124",
}
```

- Accessible on an assigned ipv6 address from prefix

  *AWS internet vs. egress-only gateway / firewalls …

- Port mapping are ***NOT*** required anymore

```
# sudo docker run –d –name www nginx
```

How to enable IPv6 on Docker (3)

- IPv6 native
- IPv6 pool used to provision `docker network ...`
- IPv6 prefix delegation

```
# cat /etc/docker/daemon.json
{
"ipv6": true,
"fixed-cidr-v6": "2a05:d012:d41:8008:90dd::/80",
  "default-address-pools": [
    { "base": "172.17.0.0/16", "size": 16 },
    { "base": "172.18.0.0/16", "size": 16 },
    { "base": "172.19.0.0/16", "size": 16 },
    { "base": "172.20.0.0/14", "size": 16 },
    { "base": "172.24.0.0/14", "size": 16 },
    { "base": "172.28.0.0/14", "size": 16 },
    { "base": "192.168.0.0/16", "size": 20 },
    { "base": "2a05:d012:d41:8008:9d0e::/96", "size": 112 }
  ]
}
```

How to enable IPv6 on Docker (4)

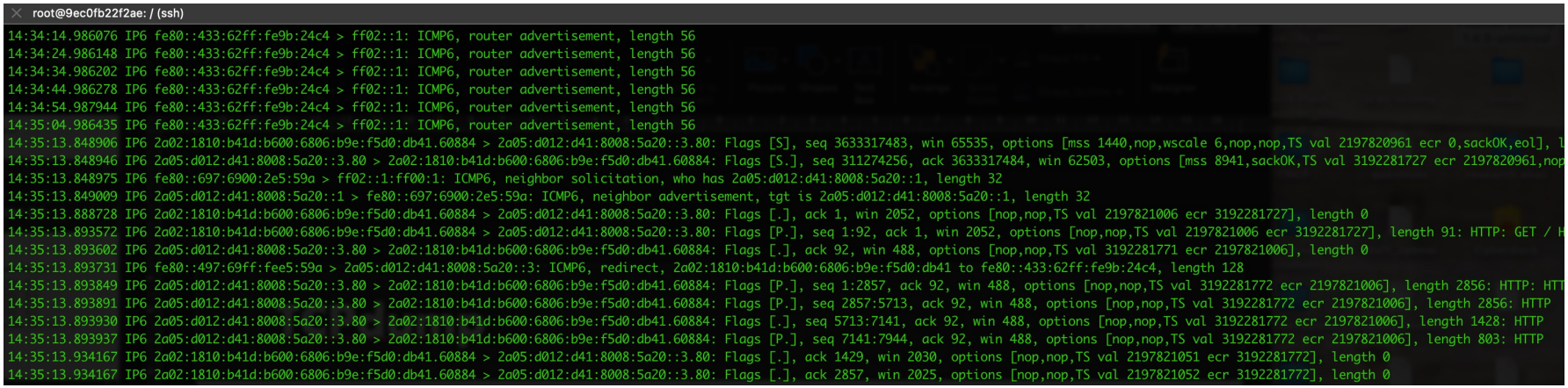
- MACVLAN
- IPVLAN
- --net=host
- --net=container:id
- Overlay (swarm)

```
docker network create --ipv6 -d ipvlan \
    -o parent=ens5 \
    --subnet 2a05:d012:d41:8008:5a20::/80 \
    --ip-range 2a05:d012:d41:8008:5a20::/96 ip6vlan
```

Tip: https://xxradar.medium.com/docker-pentester-series-1-macvlan-be4bca3062f2

Troubleshooting w/ TCPdump

```
docker run -it --net=container:www3 xxradar/hackon tcpdump -n
```



https://xxradar.medium.com/how-to-tcpdump-effectively-in-docker-2ed0a09b5406

Considerations

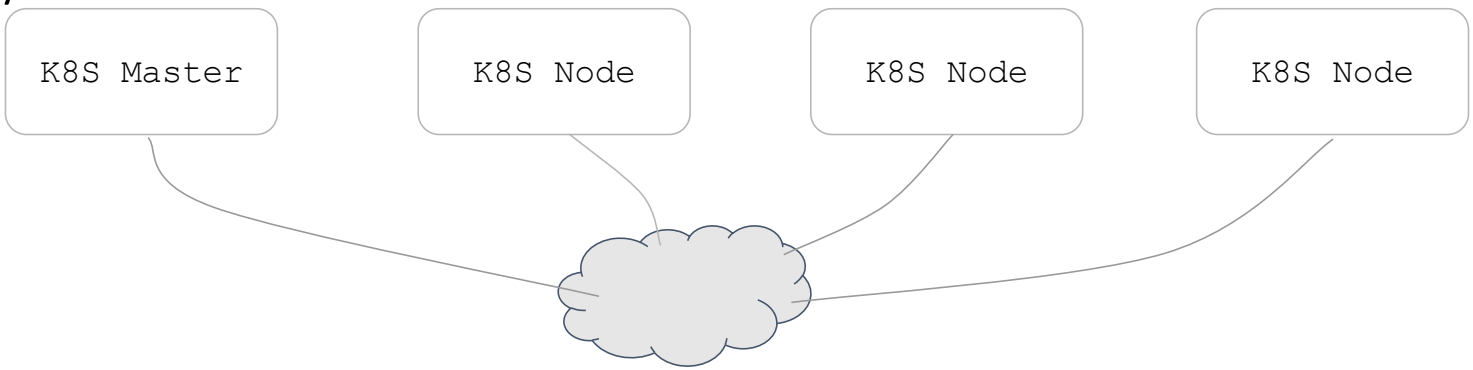
- What about network security ?
- What about container security ?
- What about registry support ?
- What about CI/CD support ?

IPv6 and Kubernetes

[opinions expressed are solely my own]

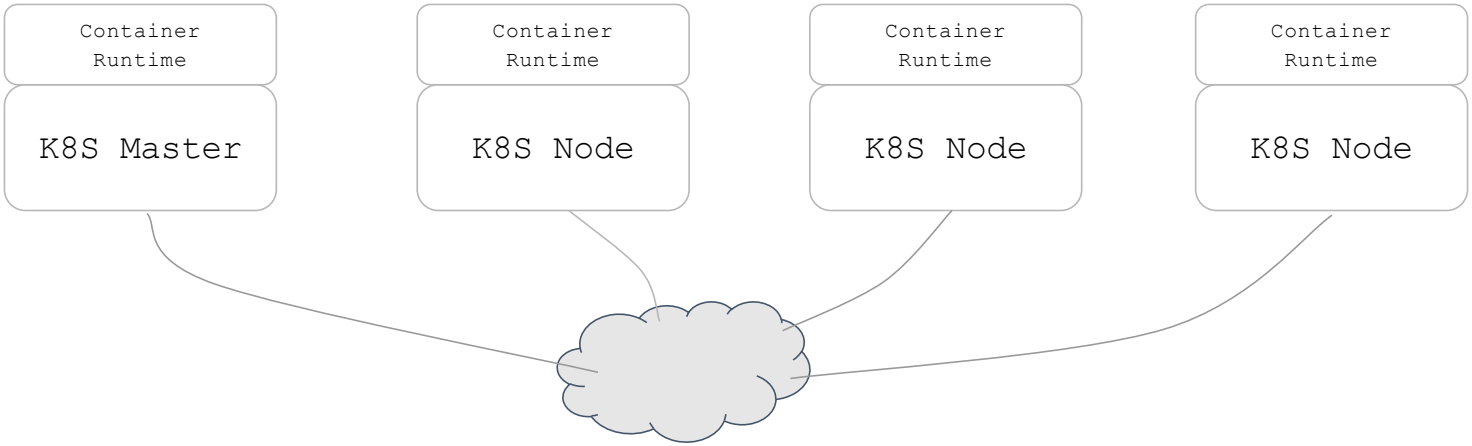
Kubernetes - Nodes

- Hardware or VM
- Master node(s)
  - Linux
- Worker nodes
  - Linux / Windows



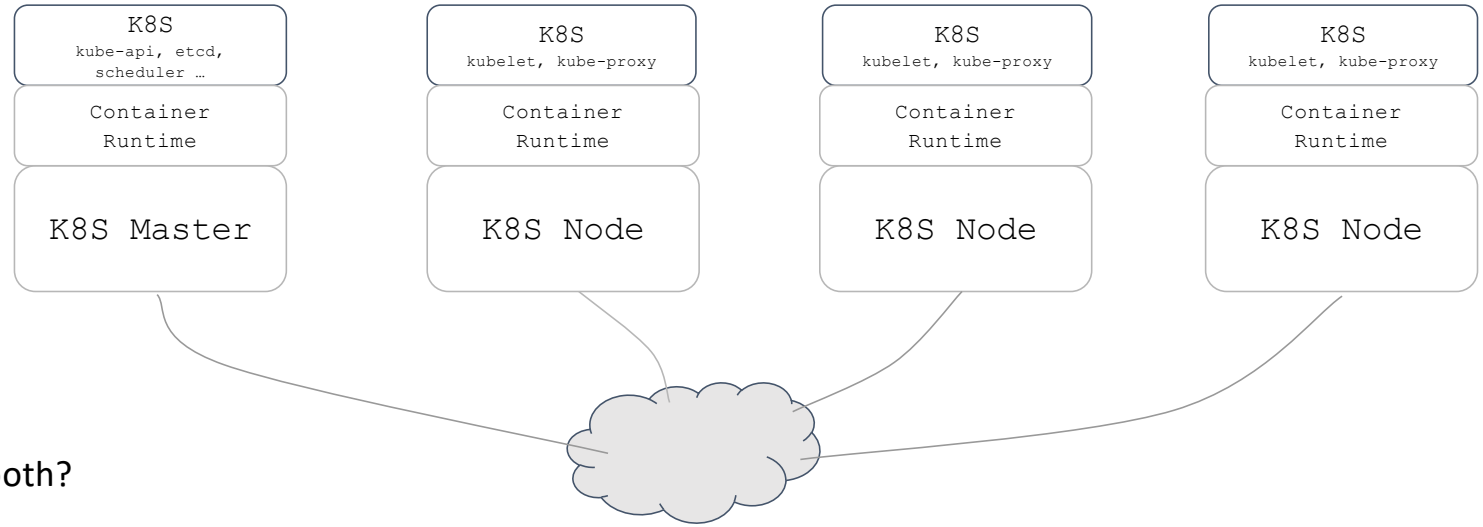
Kubernetes – Container Runtime

- Container runtimes
  - CRI-O
  - Containerd
  - Docker is not supported anymore in latest releases !!
  - …



Kubernetes – Control Plane

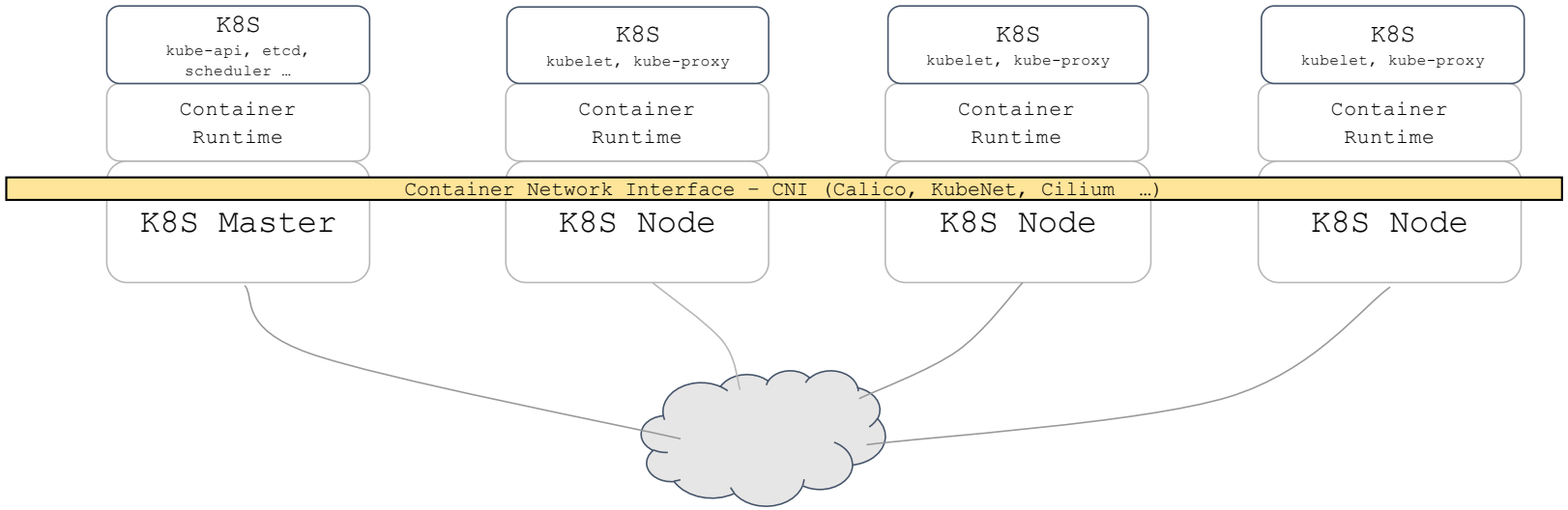
- K8S components are typically binaries or pods that communicate over the network using the host network IP address



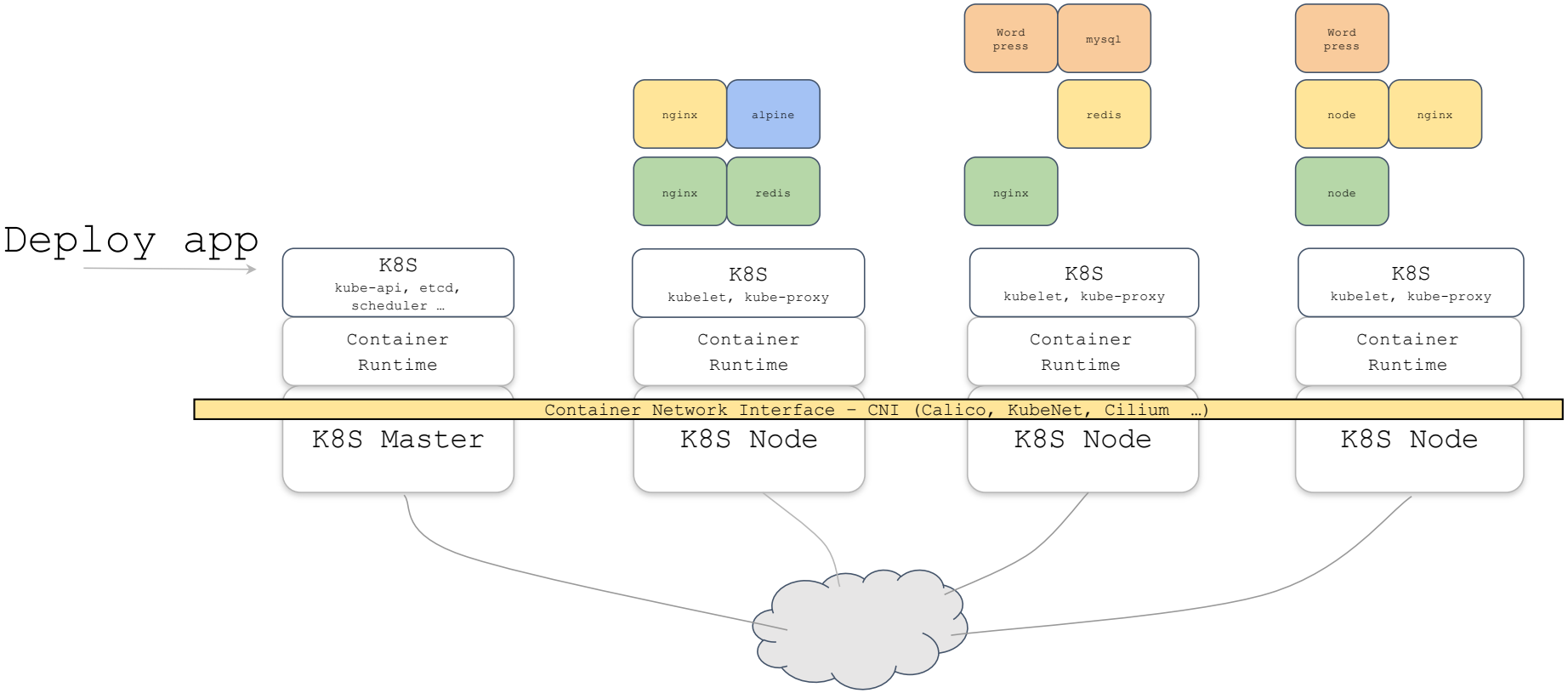
IPv6, IPv4 or both?

CNI - Container Network Interface

- K8S worloads (ex. Pods) need to communicate using IP networking. The networking, IPAM, routing … is handled by the CNI (and not K8S)

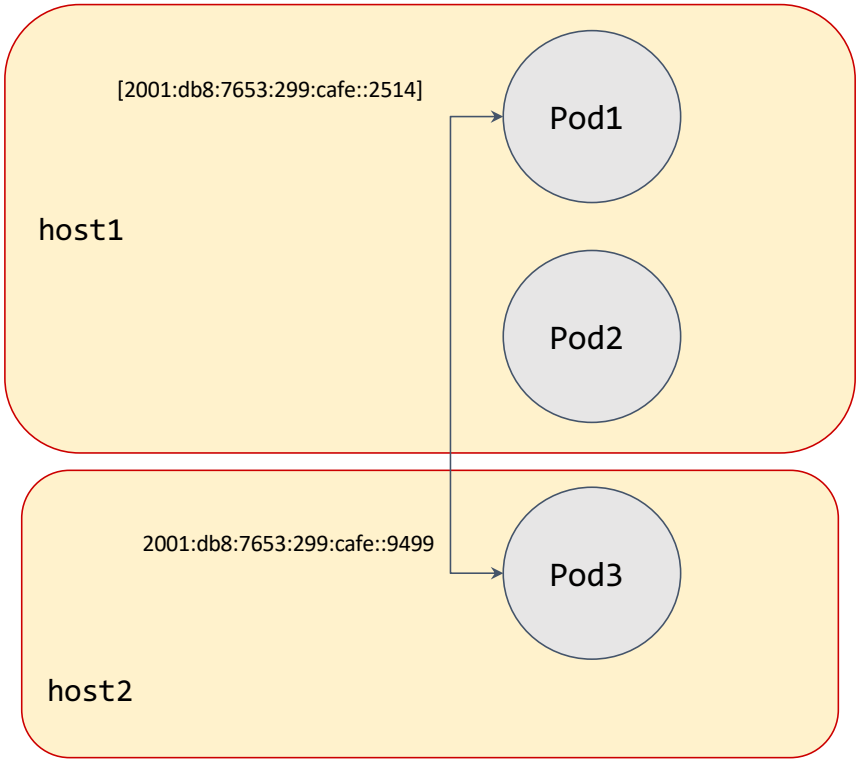


Kubernetes – Basic principles



Kubernetes networking considerations

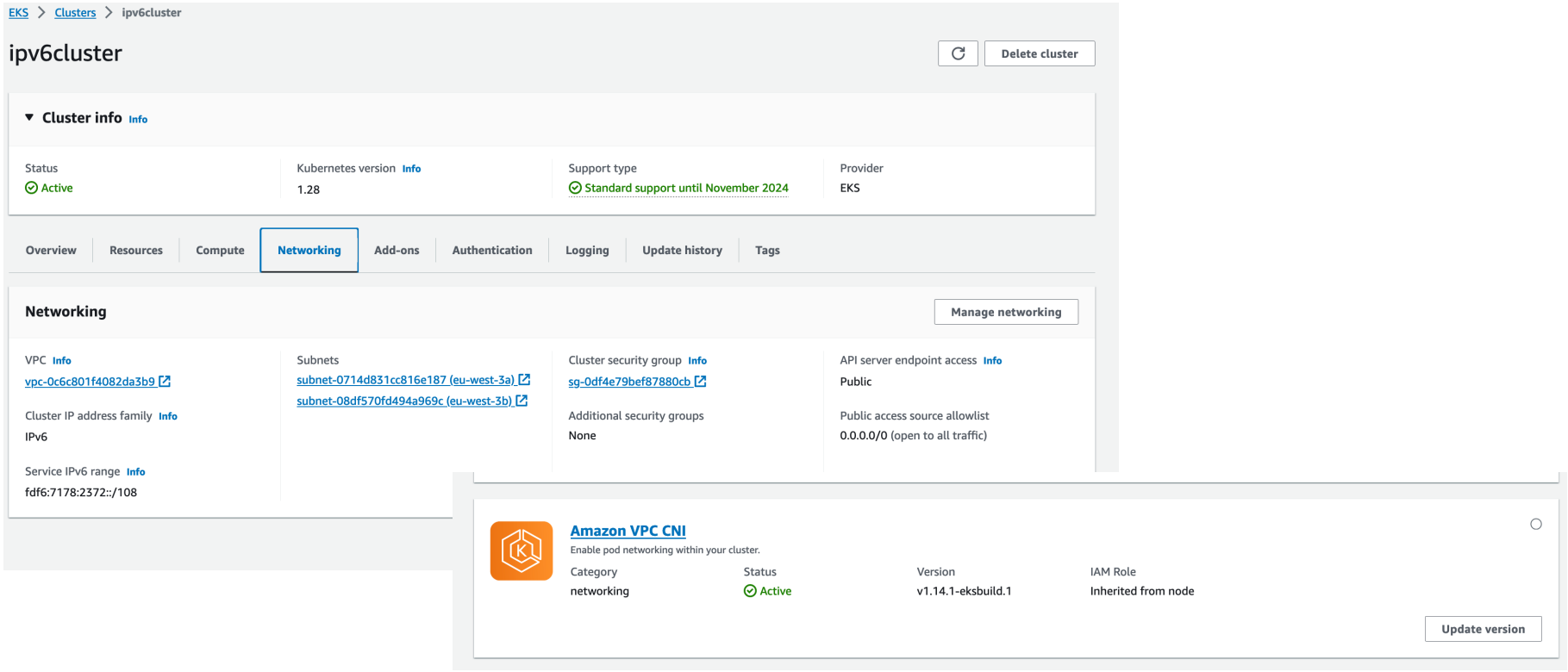
- A single default network plugin (CNI)
- Each pod has a unique IP address
- Pods on one node can communicate with all pods on all other nodes without NAT
- How do I choose a pod CIDR ?
  - private or externally routable
- How do I choose a Service CIDR ?
  - see later



My K8S cluster (dual stack) – Cilium CNI

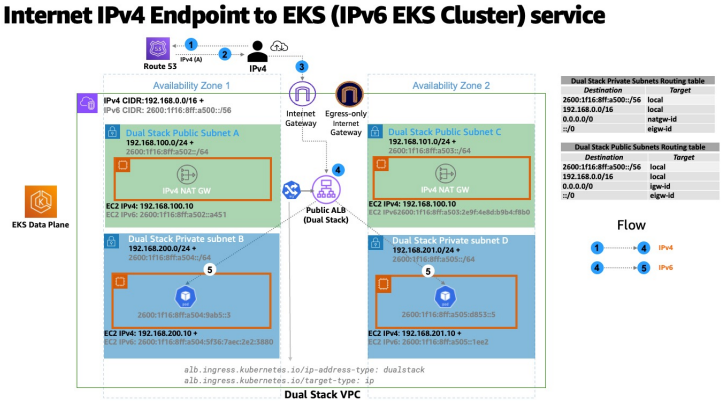
```
helm install cilium cilium/cilium --version 1.14.2 \
--namespace kube-system \
--set auto-create-cilium-node-resource=true \
--set ipv4.enabled=true \
--set ipv6.enabled=true \
--set ipam.operator.clusterPoolIPv4PodCIDRList="10.244.0.0/16" \
--set ipam.operator.clusterPoolIPv6PodCIDRList="2001:db8:7653:299:cafe:0::/96" \
--set ipam.operator.clusterPoolIPv4MaskSize=24 \
--set ipam.operator.clusterPoolIPv6MaskSize=112 \
--set ipam.mode=cluster-pool \
--set hubble.relay.enabled=true \
--set hubble.ui.enabled=true \
--set cluster.id=1 \
--set cluster.name="cluster1"
```

My EKS IPv6 only cluster



EKS pro/cons

- Prefix assignment mode (VPC-CNI) for IPv6 clusters
  /80 => ~10^14 addresses per ENI)
- Only works with AWS Nitro-based EC2 instances
- IPv4 ***or*** IPv6 as the IP address family for the cluster
- Control plane dual stack



IPAM and Cloud support

- In K8S, the CNI is responsible for managing the IP addressing
- On-premise vs cloud
- Integration needed by 3rd party CNI

- GKE now supports dual stack

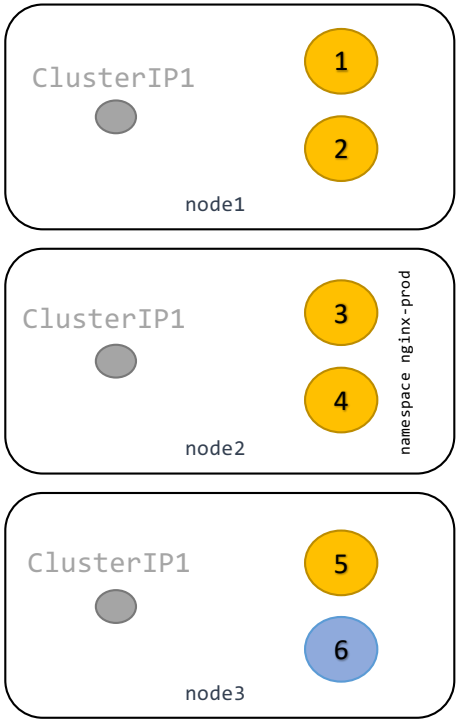
https://cloud.google.com/blog/products/networking/gke-networking-operational-readiness-for-dual-stack-traffic

- Dual-stack kubenet networking in Azure Kubernetes Service (AKS)

ClusterIP

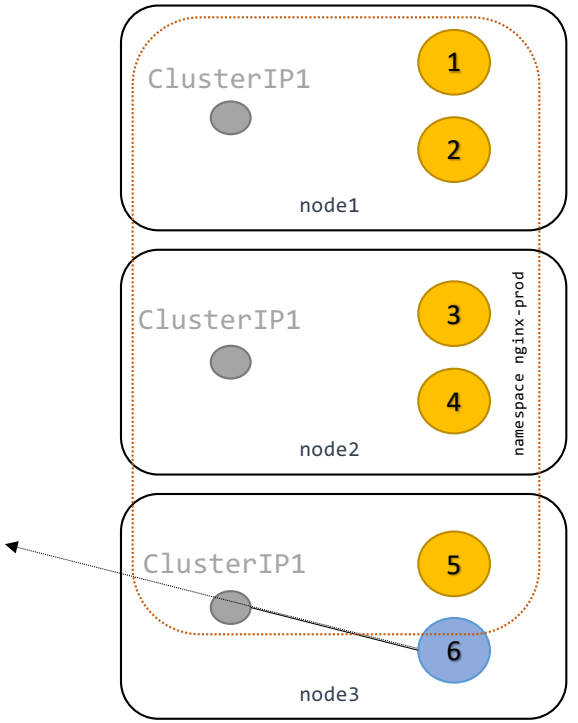
- **A K8S service** is an abstract way to expose an application running on **a set of pods** (selection of pod labels) as **a network service (IP:PORT)**
- A ClusterIP is chosen form the **SERVICE-CIDR** configured at cluster setup
  (kubectl cluster-info dump | grep -m 1 service-cluster-ip-range)
- A ClusterIP has an assoiciated DNS name

```
kind: Service
metadata:
  name: zone1
spec:
  ipFamilyPolicy: PreferDualStack
  ipFamilies:
  - IPv6
  - IPv4
  ports:
  - name: echo-http
    port: 80
    protocol: TCP
  - name: echo-https
    port: 443
    protocol: TCP
  selector:
    app: nginx-zone1
```



ClusterIP (2)

- ClusterIP is implemeted by kube-proxy/iptables (or EBPF)
- Pod6 makes DNS request to ex. 'ClusterIP1.nginx-prod.svc.cluster.local'
- CoreDNS replies with a Service IP address

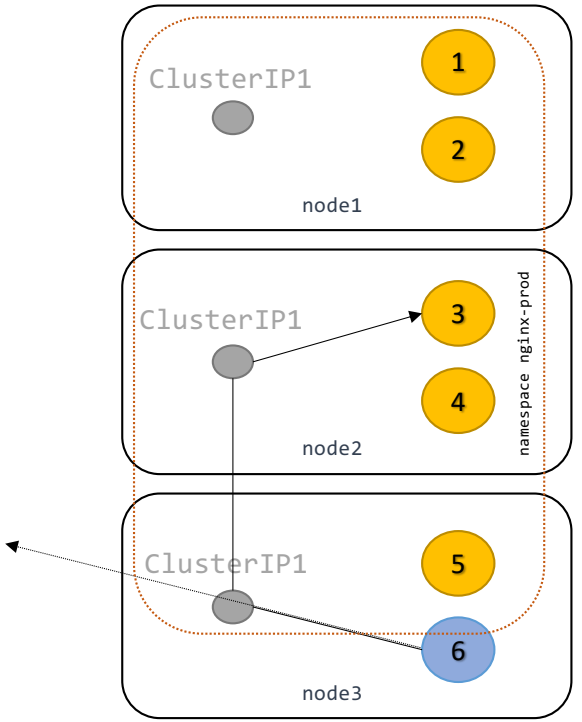


CoreDNS SVC lookup

```
root@demo:/# dig AAAA zone1.app-routable-demo.svc.cluster.local.

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> AAAA zone1.app-routable-demo.svc.cluster.local.
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42359
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 97e37329221ae471 (echoed)
;; QUESTION SECTION:
;zone1.app-routable-demo.svc.cluster.local. IN AAAA

;; ANSWER SECTION:
zone1.app-routable-demo.svc.cluster.local. 30 IN AAAA 2001:db8:42:1::9e9c

;; Query time: 0 msec
;; SERVER: 10.96.0.10#53(10.96.0.10) (UDP)
;; WHEN: Wed Oct 11 15:51:21 UTC 2023
;; MSG SIZE  rcvd: 151
```

ClusterIP (3)

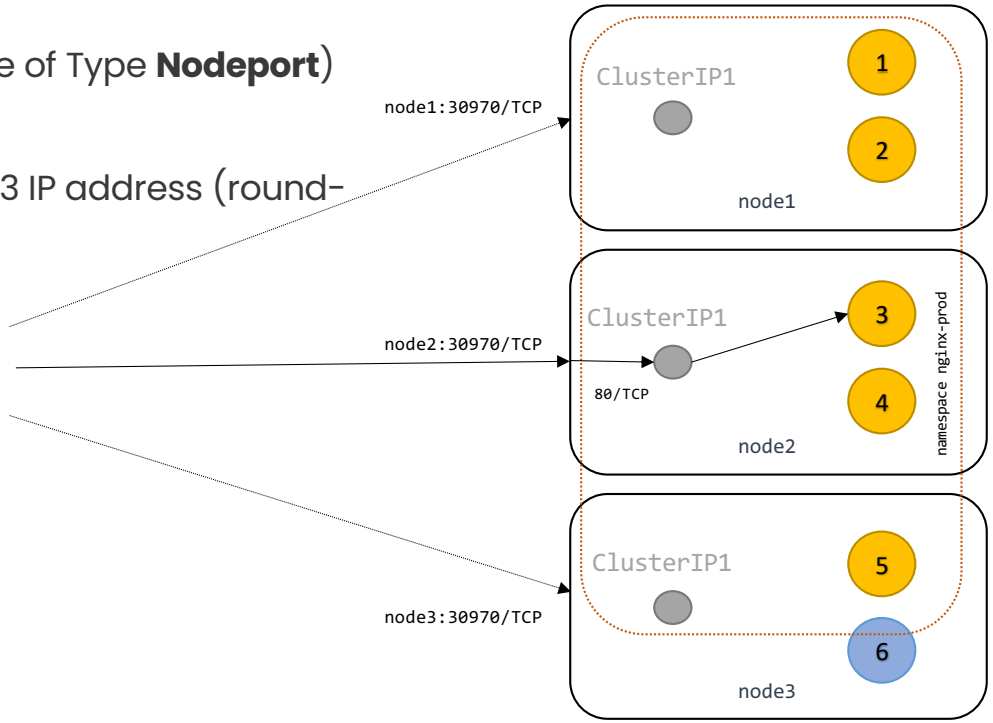
- POD6 connects to the **ClusterIP** and gets intercepted by IPTables
- IPTables will **DNAT** to the POD3 IP address (round-robin) using the overlay/routing setup. The receiving POD sees the **original source IP address** of POD6
- IPTables will also make sure return packets arrive at node3/pod6



NodePort services

- External clients connect to a(Service of Type **Nodeport**) and gets intercepted by IPTables
- IPTables will **SNAT/DNAT** to the POD3 IP address (round-robin / cluster-first)

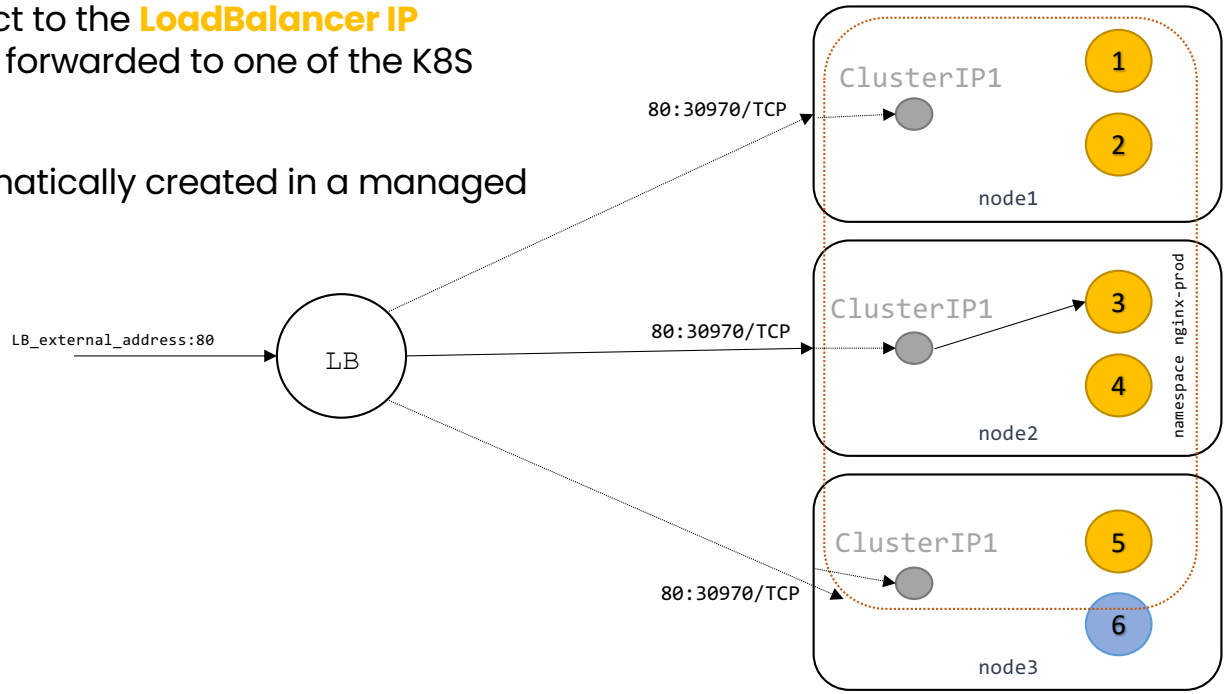
```
apiVersion: v1
kind: Service
metadata:
  name: ClusterIP1
  namespace: nginx-prod
spec:
 ipFamilyPolicy: PreferDualStack
 ipFamilies:
 - IPv6
 - IPv4
  type: NodePort
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
```



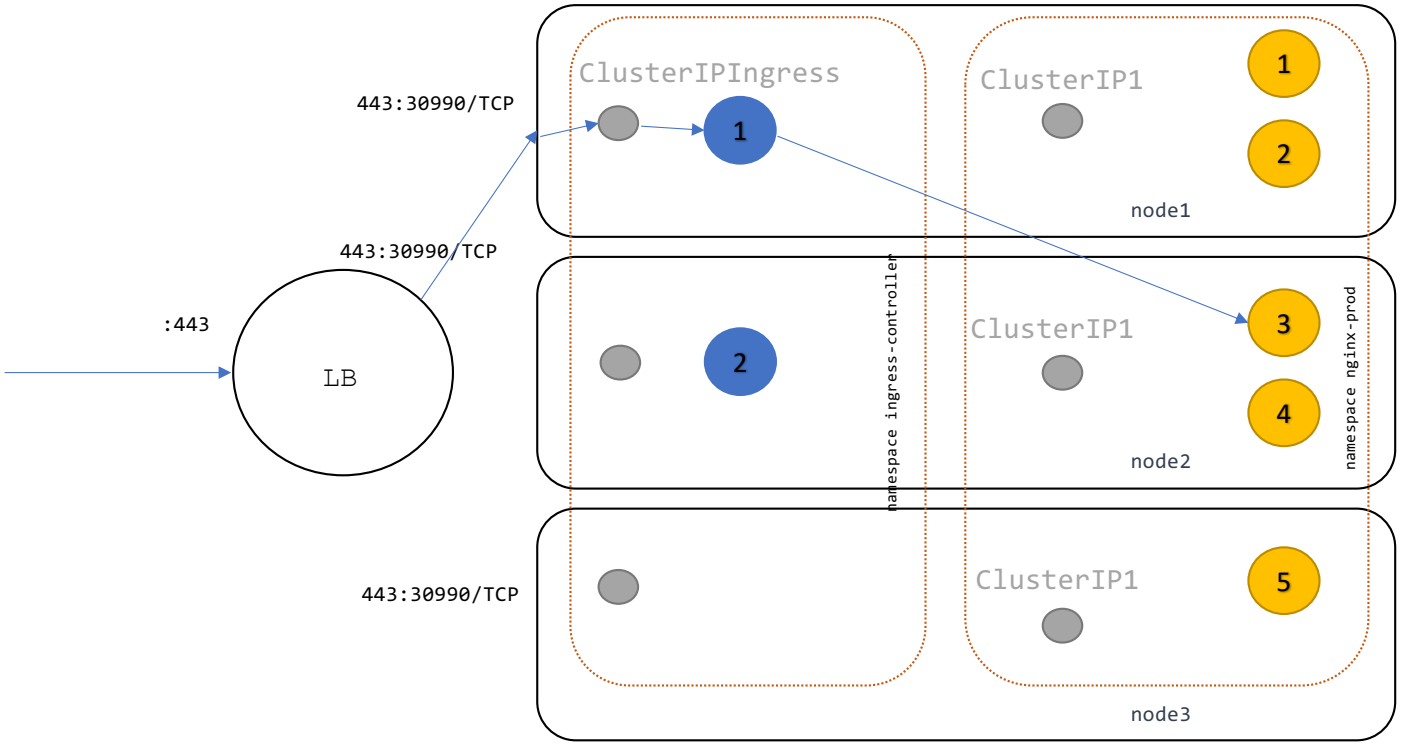
LoadBalancer services

- External clients connect to the **LoadBalancer IP address and port** and forwarded to one of the K8S nodes.
- The LB might be automatically created in a managed K8S env.

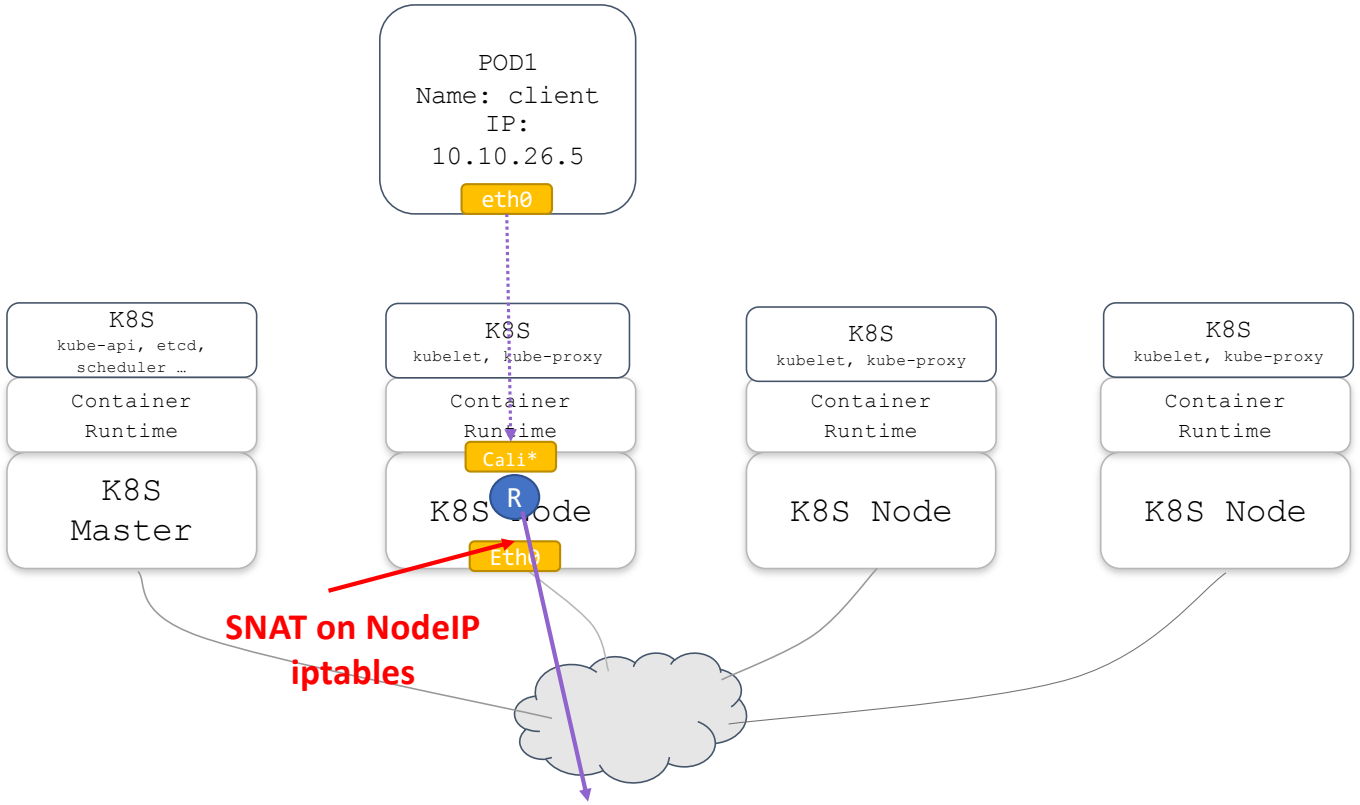
```
apiVersion: v1
kind: Service
metadata:
  name: ClusterIP1
  namespace: nginx-prod
spec:
 ipFamilyPolicy: PreferDualStack
 ipFamilies:
 - IPv6
 - IPv4
  type: LoadBalancer
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
```



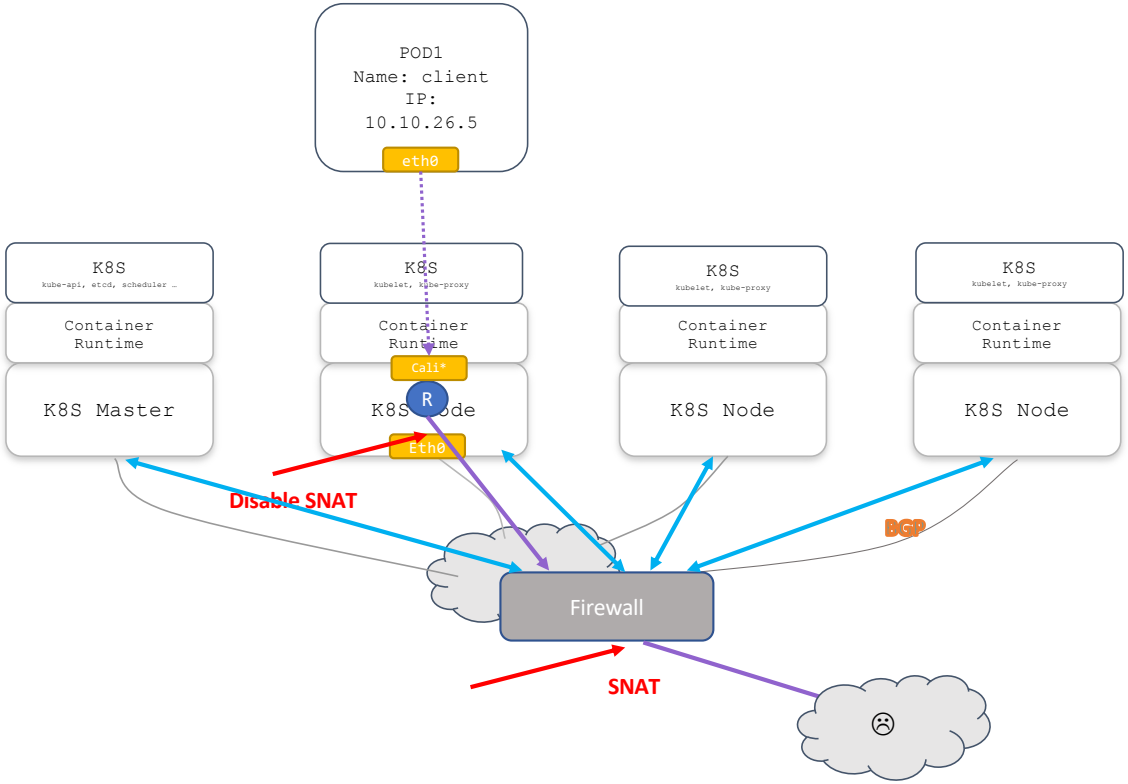
Kubernetes Ingress Controller



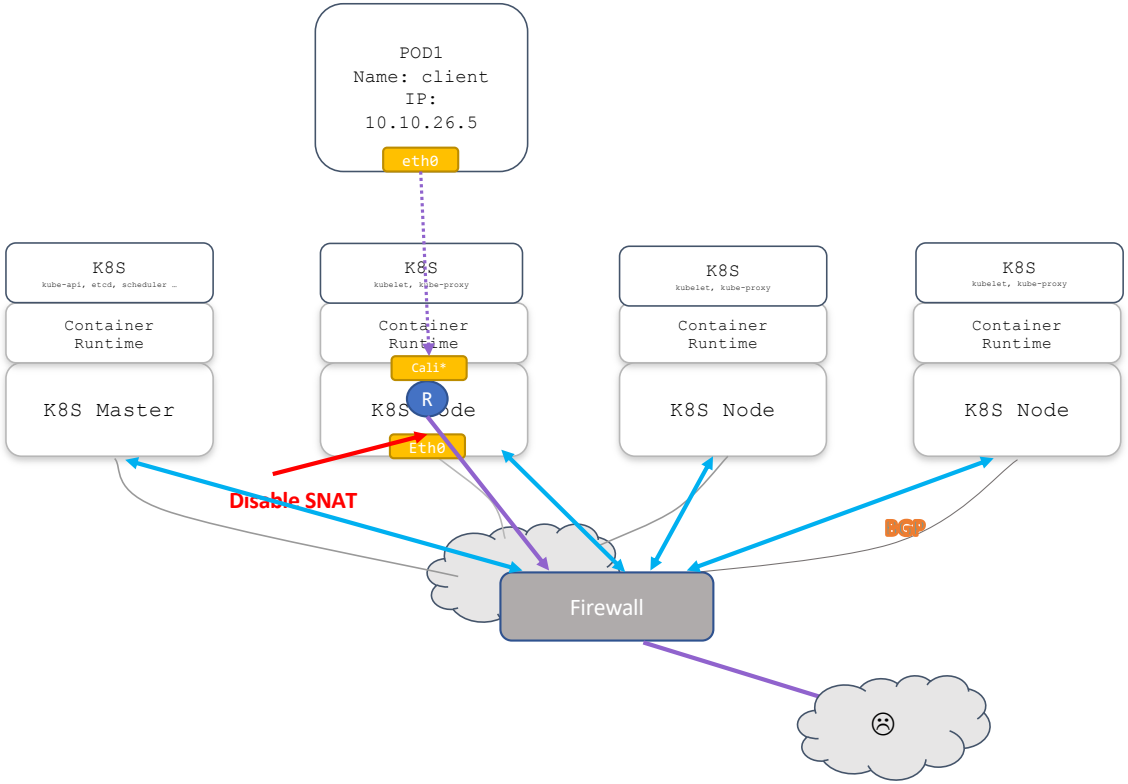
Cluster egress (SNAT)



Cluster egress (NVA)

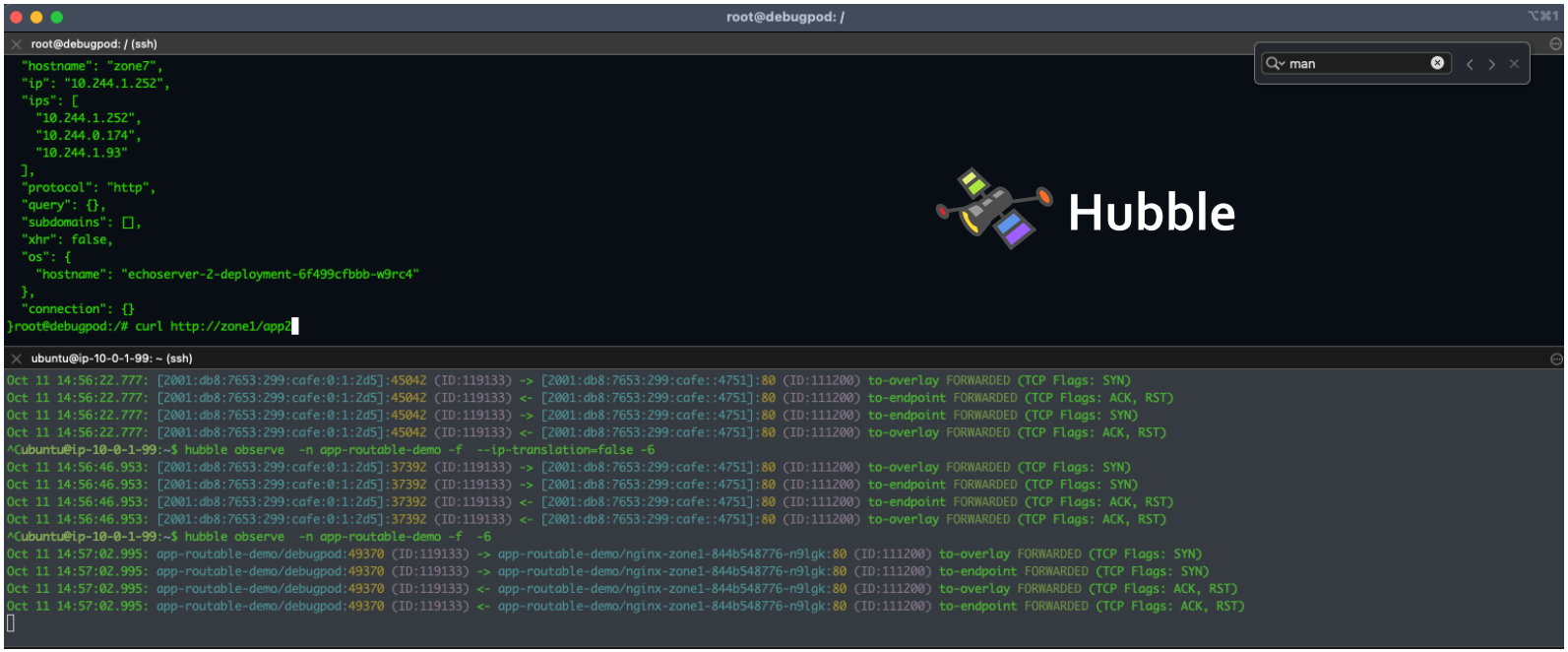


Cluster egress (IPv6 routing)



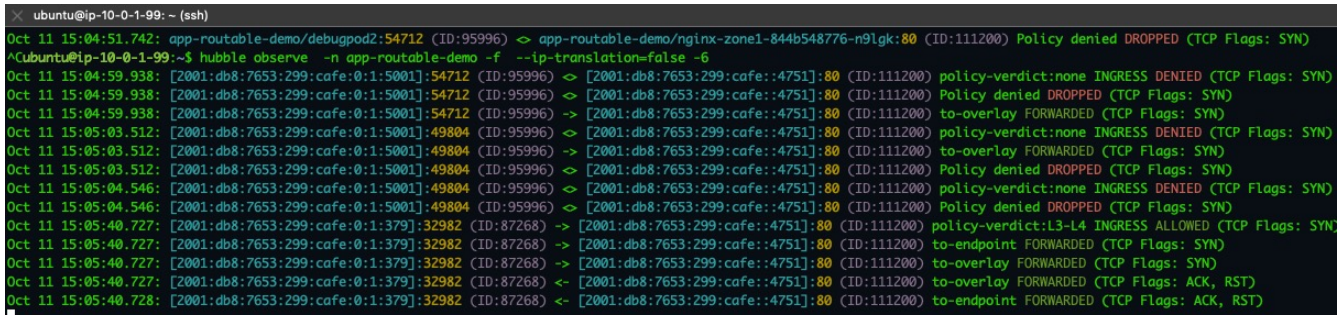
Observability and troubleshooting

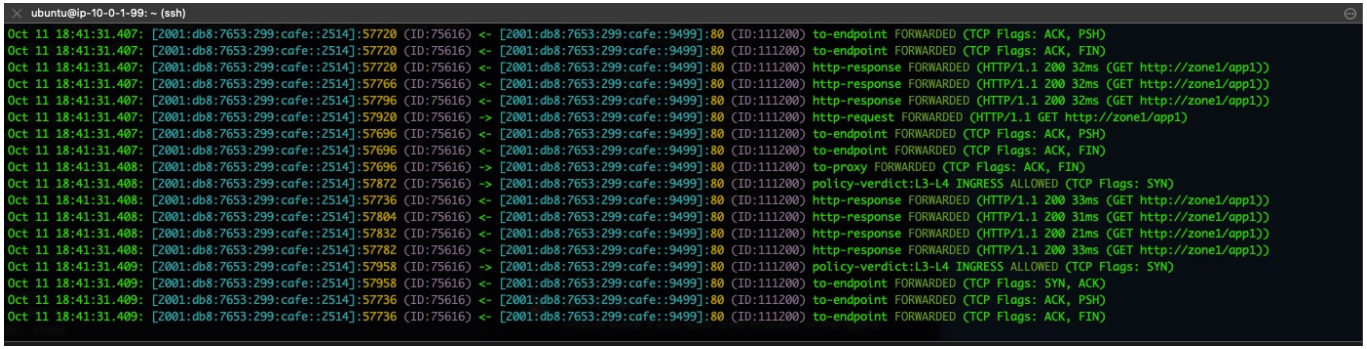
- TCPdump
- EBPF



Network Security Policies

```
kubectl apply -f - <<EOF
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-access-from-siege
  namespace: app-routable-demo
spec:
  endpointSelector:
    matchLabels:
      app: nginx-zone1
  ingress:
    - fromEndpoints:
        - matchLabels:
            app: siege
      toPorts:
        - ports:
            - port: "80"
              protocol: TCP
EOF
```





Runtime security and monitoring

- Tetragon

```
ubuntu@ip-10-0-1-99: ~ (ssh)
sendmsg app-routable-demo/demopo /usr/bin/curl tcp 10.244.0.64:46862 -> 10.96.107.51:80 bytes 73
connect app-routable-demo/nginx-zone1-844b548776-n9lgk /usr/sbin/nginx tcp 10.244.0.174:37488 -> 10.96.227.74:80
sendmsg app-routable-demo/nginx-zone1-844b548776-n9lgk /usr/sbin/nginx tcp 10.244.0.174:37488 -> 10.96.227.74:80 bytes 143
close   app-routable-demo/nginx-zone1-844b548776-n9lgk /usr/sbin/nginx tcp 10.244.0.174:37488 -> 10.96.227.74:80
sendmsg app-routable-demo/nginx-zone1-844b548776-n9lgk /usr/sbin/nginx tcp 10.244.0.174:80 -> 10.244.0.64:46862 bytes 830
close   app-routable-demo/demopo /usr/bin/curl tcp 10.244.0.64:46862 -> 10.96.107.51:80
close   app-routable-demo/nginx-zone1-844b548776-n9lgk /usr/sbin/nginx tcp 10.244.0.174:80 -> 10.244.0.64:46862
exit    app-routable-demo/demopo /usr/bin/curl -kv http://zone1/app1 0
process app-routable-demo/demopo /usr/bin/curl https://www.facebook.com
connect app-routable-demo/demopo /usr/bin/curl tcp 2001:db8:7653:299:cafe::8c5b:49732 -> 2a03:2880:f17b:187:face:b00c:0:25de:443
sendmsg app-routable-demo/demopo /usr/bin/curl tcp 2001:db8:7653:299:cafe::8c5b:49732 -> 2a03:2880:f17b:187:face:b00c:0:25de:443 bytes 517
sendmsg app-routable-demo/demopo /usr/bin/curl tcp 2001:db8:7653:299:cafe::8c5b:49732 -> 2a03:2880:f17b:187:face:b00c:0:25de:443 bytes 64
sendmsg app-routable-demo/demopo /usr/bin/curl tcp 2001:db8:7653:299:cafe::8c5b:49732 -> 2a03:2880:f17b:187:face:b00c:0:25de:443 bytes 46
sendmsg app-routable-demo/demopo /usr/bin/curl tcp 2001:db8:7653:299:cafe::8c5b:49732 -> 2a03:2880:f17b:187:face:b00c:0:25de:443 bytes 49
sendmsg app-routable-demo/demopo /usr/bin/curl tcp 2001:db8:7653:299:cafe::8c5b:49732 -> 2a03:2880:f17b:187:face:b00c:0:25de:443 bytes 35
sendmsg app-routable-demo/demopo /usr/bin/curl tcp 2001:db8:7653:299:cafe::8c5b:49732 -> 2a03:2880:f17b:187:face:b00c:0:25de:443 bytes 63
```


```
ubuntu@ip-10-0-1-99: ~ (ssh)
close   app-routable-demo/siege-deployment-8c895f649-js68b /usr/bin/siege tcp 10.244.0.219:58716 -> 10.96.109.226:80
sendmsg app-routable-demo/echoserver-2-deployment-6f499cfbbb-v89lx /usr/local/bin/node tcp 10.244.0.242:8080 -> 10.244.0.73:33578 bytes 869
close   app-routable-demo/echoserver-2-deployment-6f499cfbbb-v89lx /usr/local/bin/node tcp 10.244.0.242:0 -> 10.244.0.73:33578
process app-routable-demo/mycurler /usr/bin/curl -v -H "Cookie: loc=client" http://zone1/app3
connect app-routable-demo/mycurler /usr/bin/curl tcp 2001:db8:7653:299:cafe::15fe:48182 -> 2001:db8:42:1::14fe:80
sendmsg app-routable-demo/mycurler /usr/bin/curl tcp 2001:db8:7653:299:cafe::15fe:48182 -> 2001:db8:42:1::14fe:80 bytes 93
close   app-routable-demo/mycurler /usr/bin/curl tcp 2001:db8:7653:299:cafe::15fe:48182 -> 2001:db8:42:1::14fe:80
exit    app-routable-demo/mycurler /usr/bin/curl -v -H "Cookie: loc=client" http://zone1/app3 0
```

Considerations

- What about network security ?
- What about container security ?
- What about registry support ?
- What about CI/CD support ?
- What about cluster support ?

Links and references

- Blogging and research
  - https://www.kubiosec.tech
  - https://github.com/xxradar
  - https://xxradar.medium.com
  - https://cloudyuga.guru
- Presentation examples
  - https://github.com/xxradar/ipv6_containers_k8s

Summary

- Cluster Configuration
- Network Plugins
- Cloud Provider Support
- Application Readiness
- DNS and Service Discovery
- Network Policies
- Version of Kubernetes

Questions ?

https://meetups.kubiosec.tech