IPv6 support in container technologies

Pieter Lewyllie

Systems Engineer @ Cisco

Co-Chair @ IPv6 Council BE

IPv6 & containers…

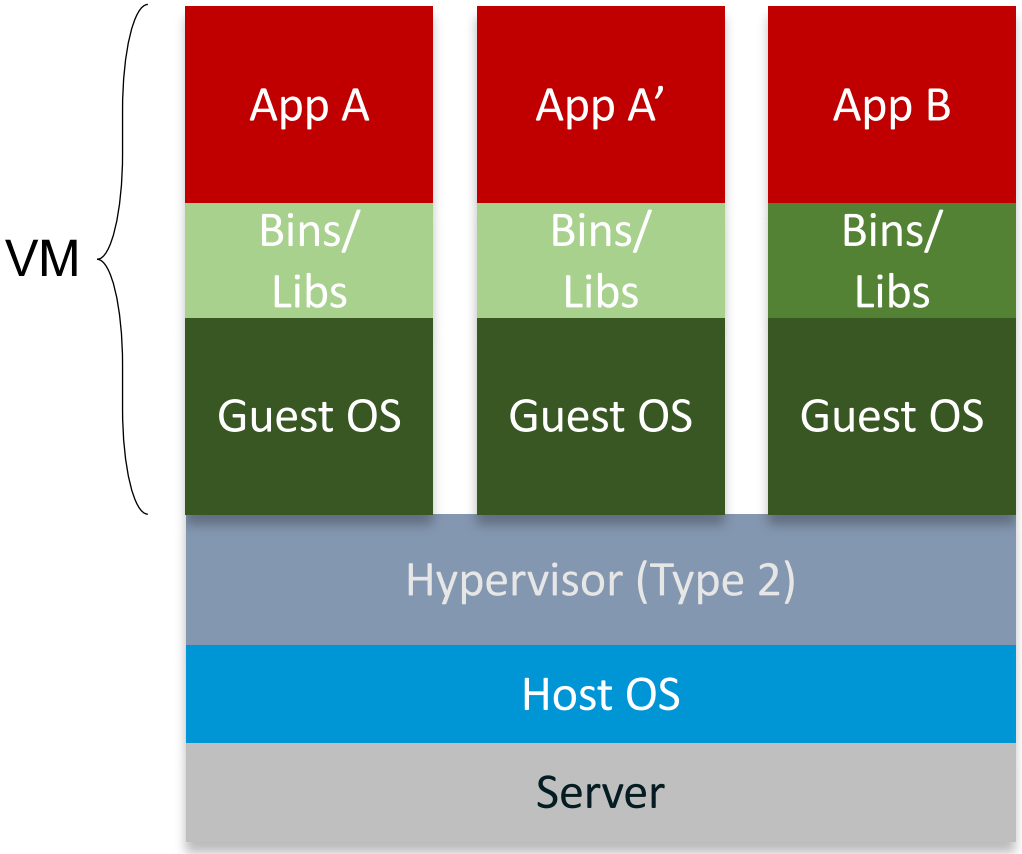
APNIC

IPv6 and containers – a horror story

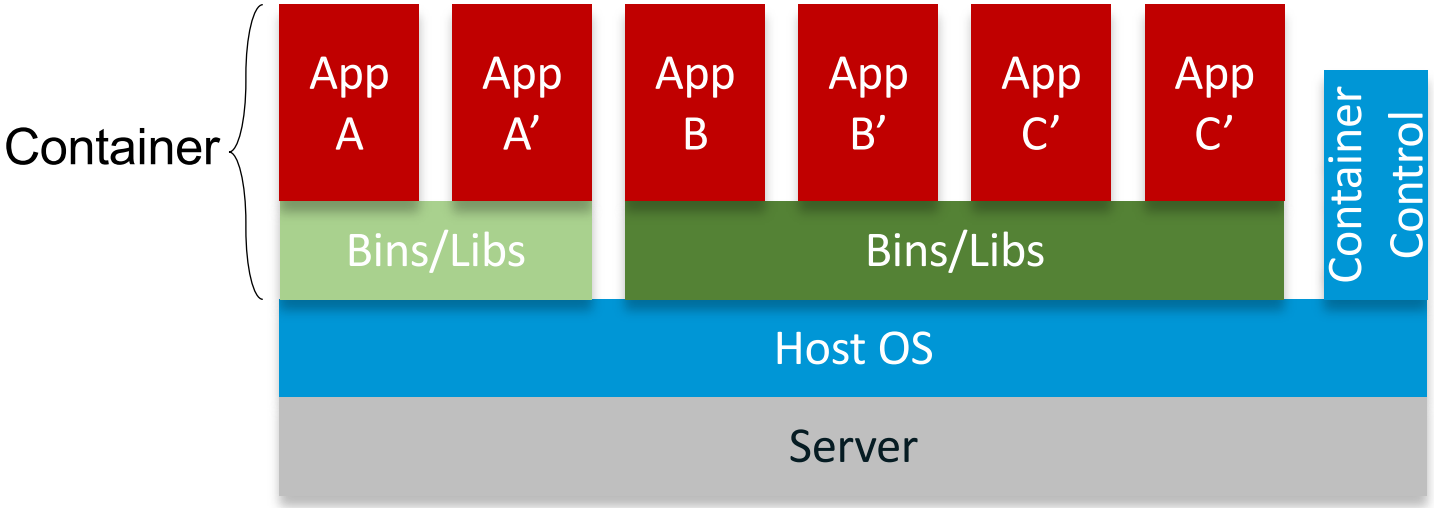
By Matt Palmer on 22 Mar 2018

https://blog.apnic.net/2018/03/22/ipv6-and-containers-a-horror-story/
https://thenewstack.io/kubernetes-warms-up-to-ipv6/

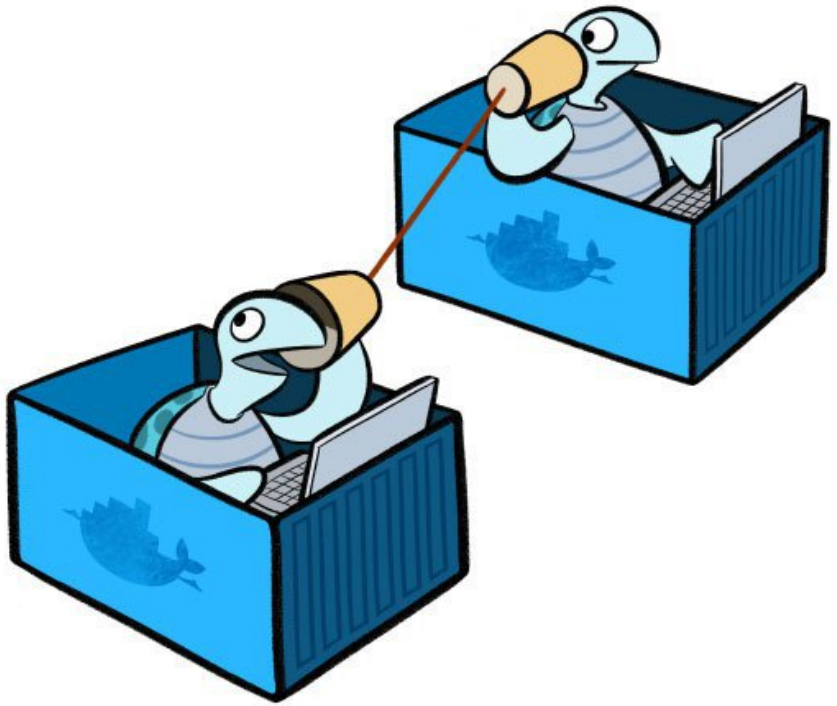
Containers and Virtual Machines



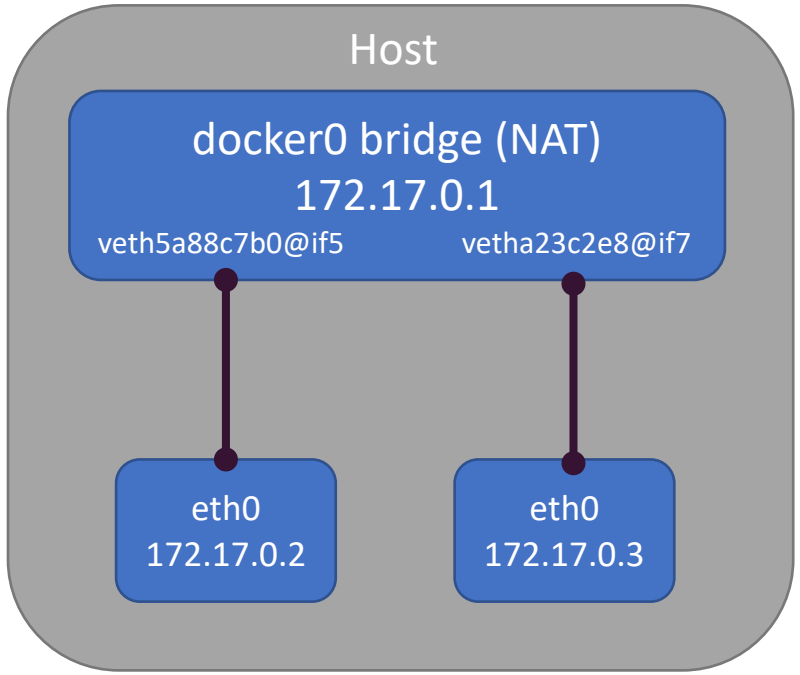
Containers are isolated but share OS and where appropriate bins/libraries



Docker networking



Docker Containers are connected using a bridge




```
pilewyll@ubuntu:~$ ifconfig docker0
docker0   Link encap:Ethernet  HWaddr 02:42:58:13:7b:9e
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
pilewyll@ubuntu:~$ ip link show | grep veth
6: veth5a88c7b@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP mode DEFAULT group default
8: vetha23c2e8@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP mode DEFAULT group default
pilewyll@ubuntu:~$
```

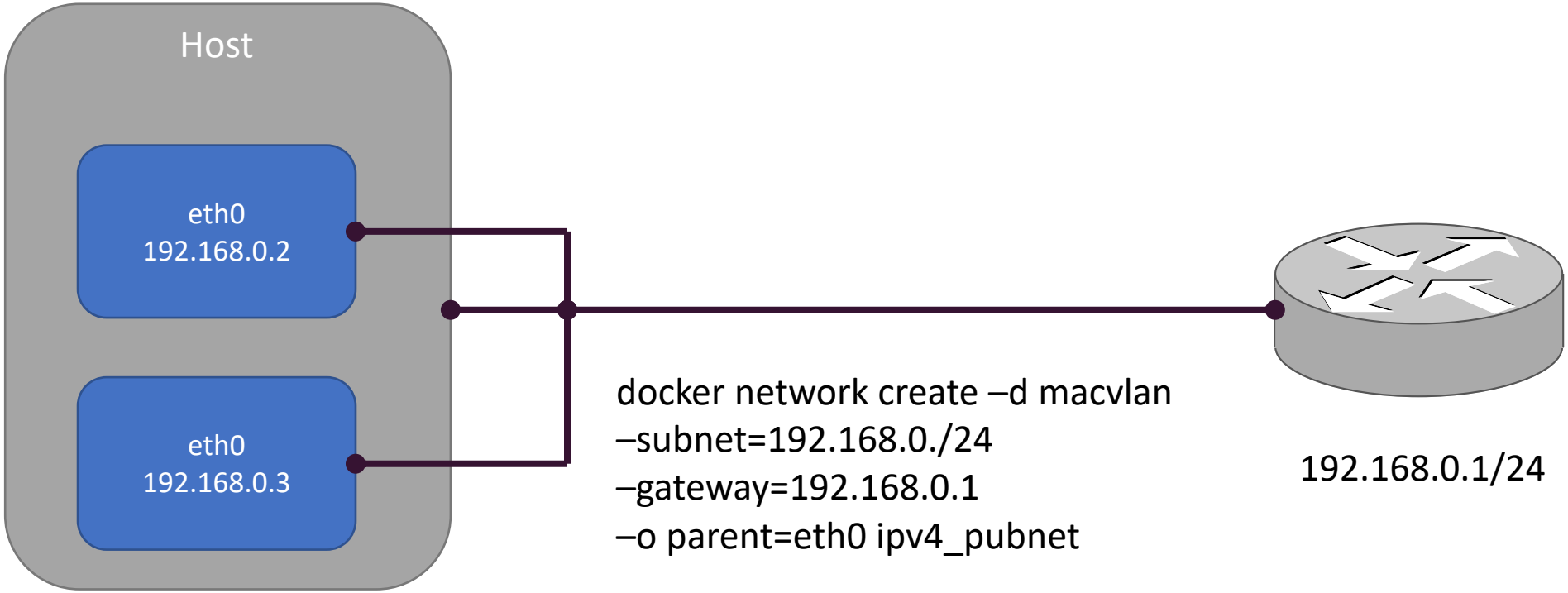
Docker bridge with IPv6

- No more NAT, all ports are exposed
- Docker assigns IPv6 addresses sequentially
- Default GW needs to be in same subnet
- Set accept_ra to 2

```
pilewyll@ubuntu:~$ docker network create testv6 --ipv6 --subnet 2001:db8:1234::/64
98bf984bf7cc9e43179ec128c519acffd28fcb031723d210e66931241b85b360
pilewyll@ubuntu:~$ docker run -i -t --network testv6 pieter/v6test /bin/bash
bash-4.4# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:12:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.2/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1234::2/64 scope global nodad
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe12:2/64 scope link
       valid_lft forever preferred_lft forever
bash-4.4# ping 2001:db8:1234::1
PING 2001:db8:1234::1(2001:db8:1234::1) 56 data bytes
64 bytes from 2001:db8:1234::1: icmp_seq=1 ttl=64 time=0.114 ms
64 bytes from 2001:db8:1234::1: icmp_seq=2 ttl=64 time=0.152 ms
^C
--- 2001:db8:1234::1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1015ms
rtt min/avg/max/mdev = 0.114/0.133/0.152/0.019 ms
```

https://docs.docker.com/v17.09/engine/userguide/networking/default_network/ipv6

Macvlan


Host
eth0
192.168.0.2
eth0
192.168.0.3
docker network create –d macvlan
–subnet=192.168.0./24
–gateway=192.168.0.1
–o parent=eth0 ipv4_pubnet
192.168.0.1/24

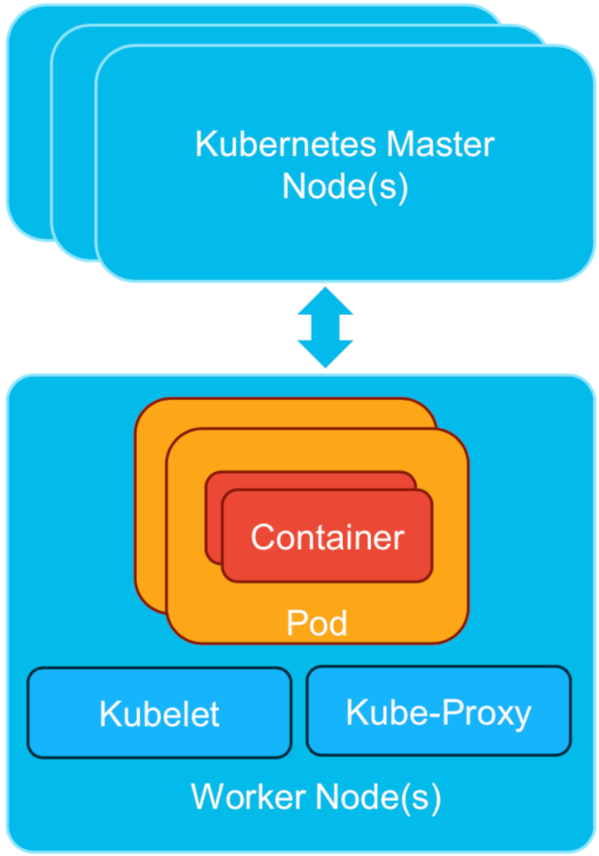
IPv6 Macvlan

```
pilewyll@ubuntu:~$ docker network create -d macvlan --subnet=192.168.0.0/24 --gateway=192.168.0.1 --ipv6 --subnet=2a02:2788:724:eb8:4ae3:
:/64 --subnet=fe80::/10 --gateway=fe80::8237:73ff:fee2:50fa -o parent=ens33 ipv6_dualstack_macvlan
19ed4504f9fd01009f002576b306b478f72be16992e707418bf065da09438bd5
pilewyll@ubuntu:~$ docker run -i -t --network ipv6_dualstack_macvlan pieter/v6test /bin/bash
bash-4.4# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
14: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a8:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.0.2/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2a02:2788:724:eb8::2/64 scope global nodad
       valid_lft forever preferred_lft forever
    inet6 fe80::42:c0ff:fea8:2/64 scope link tentative
       valid_lft forever preferred_lft forever
bash-4.4# ping6 google.be
PING google.be(ams16s30-in-x03.1e100.net (2a00:1450:400e:805::2003)) 56 data bytes
64 bytes from ams16s30-in-x03.1e100.net (2a00:1450:400e:805::2003): icmp_seq=2 ttl=50 time=30.2 ms
64 bytes from ams16s30-in-x03.1e100.net (2a00:1450:400e:805::2003): icmp_seq=3 ttl=50 time=29.0 ms
64 bytes from ams16s30-in-x03.1e100.net (2a00:1450:400e:805::2003): icmp_seq=4 ttl=50 time=28.8 ms
64 bytes from ams16s30-in-x03.1e100.net (2a00:1450:400e:805::2003): icmp_seq=5 ttl=50 time=28.8 ms
```

https://docs.docker.com/network/macvlan

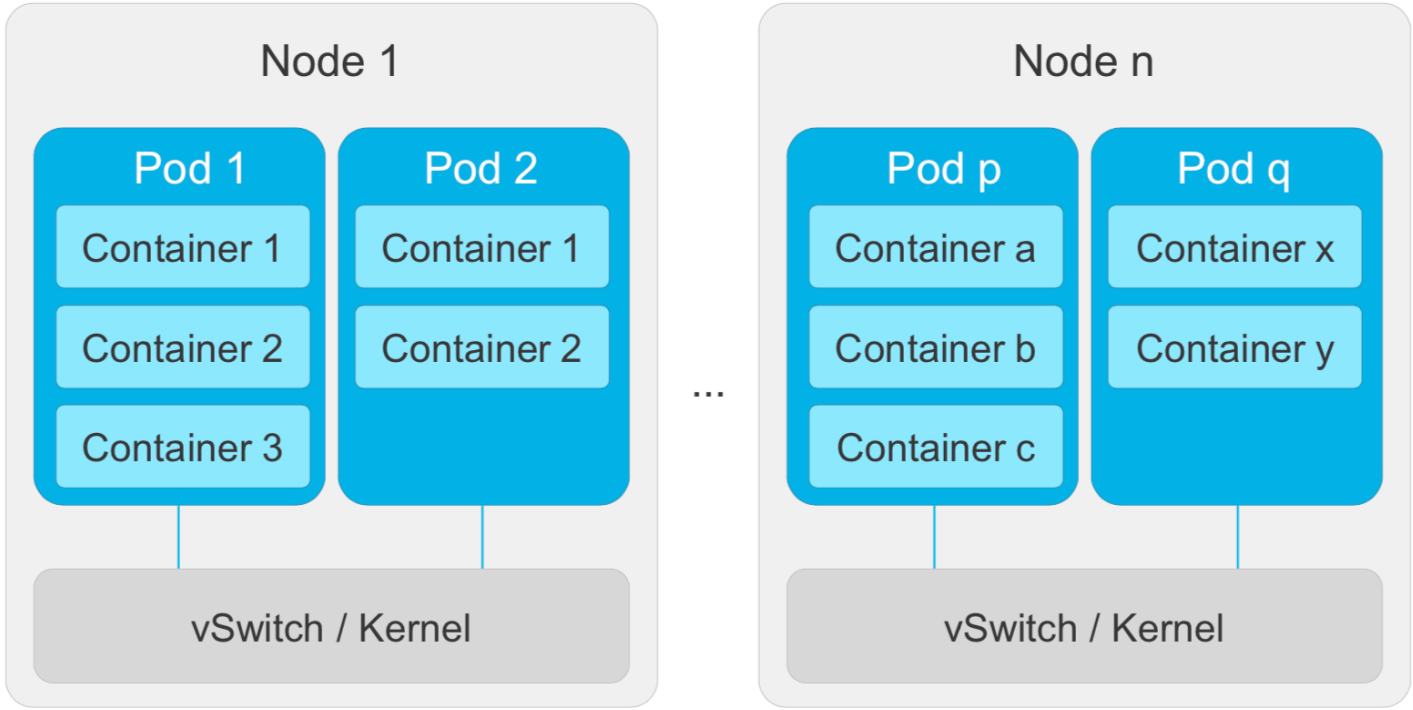
Kubernetes

- Container orchestrator
- Runs and manages containers
- Supports multiple cloud and bare-metal environments
- Inspired and informed by Google's experiences and internal systems
- 100% Open source, written in Go
- Manage applications, not machines
- Rich ecosystem of plug-ins for scheduling, storage, networking



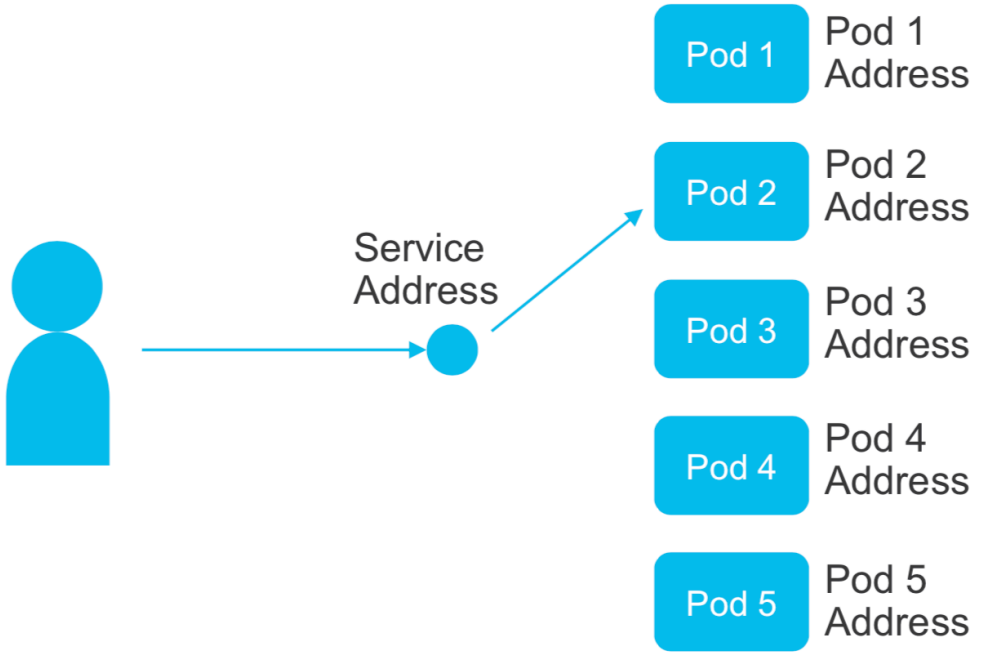
Nodes, Pods, Containers

- Node:
  - A server
- Cluster:
  - Collection of nodes
- Pod:
  - Collection of containers;
  - Nodes can run multiple Pods



Services overview

- “Pods can come and go, services stay”
- Define a single IP/Port combination that provides access to a pool of pods
- By default a service connects the client to a Pod in a round- robin fashion
- This solves the dilemma of having to keep up with every transient IP address assigned by Docker

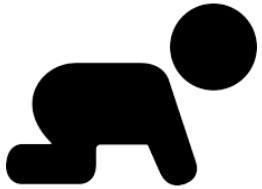


Why IPv6 for K8?

- Cleaner
- Easier diagnosis
- We need lots of IPs
- Not easy to find remaining IPv4 space in organization
- Multi cluster
- VNFs: Mobile packet core, 5G…
- IoT

IPv6 in Kubernetes

- IPv4 Parity, no API Changes
- CNI 0.6.0
  - Bridge & Host-Local IPAM
- ip6tables & ipvs
- kubeadm



Rel 1.9 (Alpha)

- Moving to CoreDNS

Rel 1.13

- Phase 1 of dual-stack KEP
- Multiple IPs per pod

Rel 1.16 (targeting)

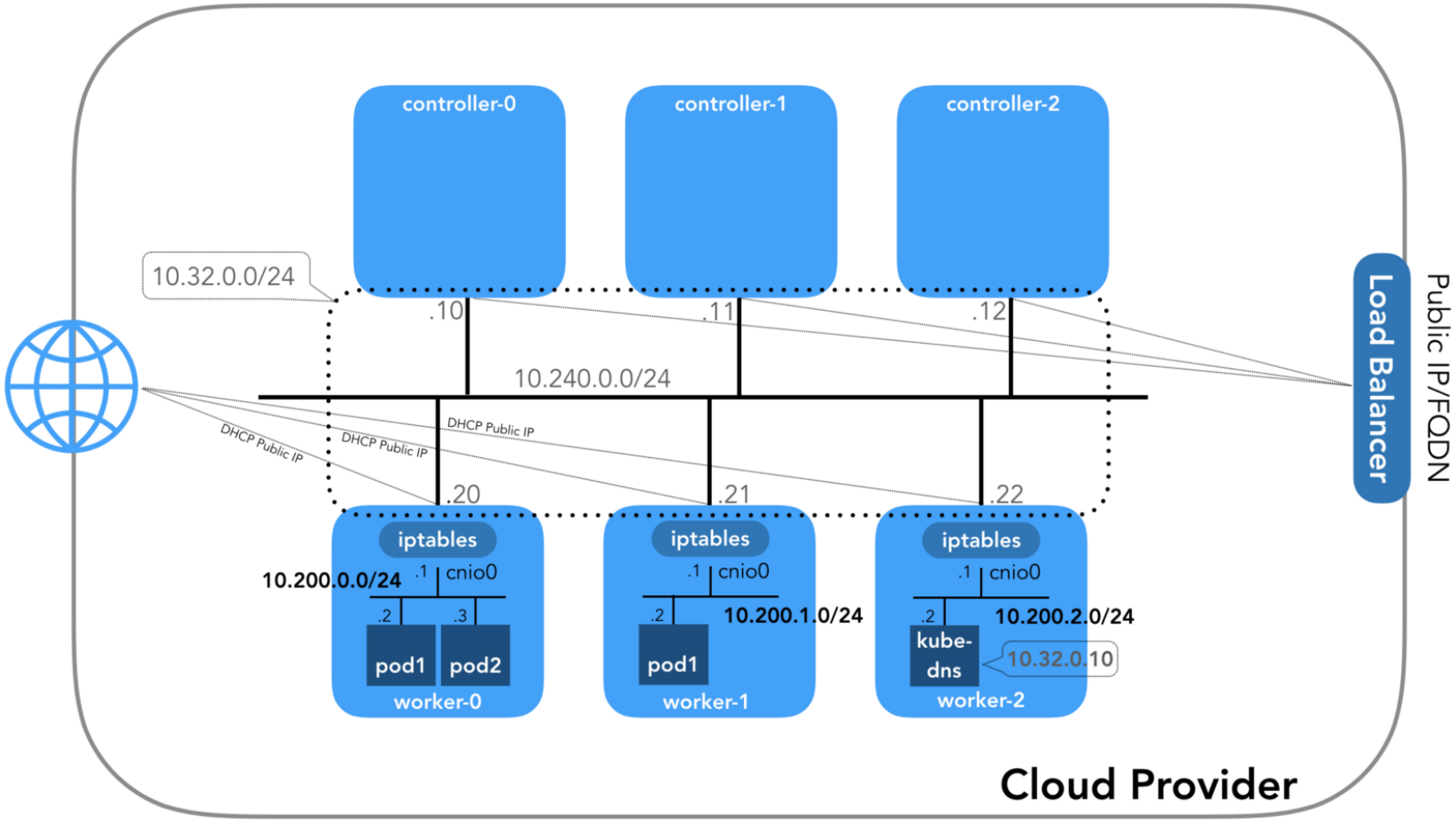
- Phase 2 of dual-stack KEP
- Dual-stack service CIDRs
- Istio IPv6
- …

Planning and Preparing

Original slide source: SRv6LB @ Kubecon https://www.youtube.com/watch?v=RRKUeyFaqEA

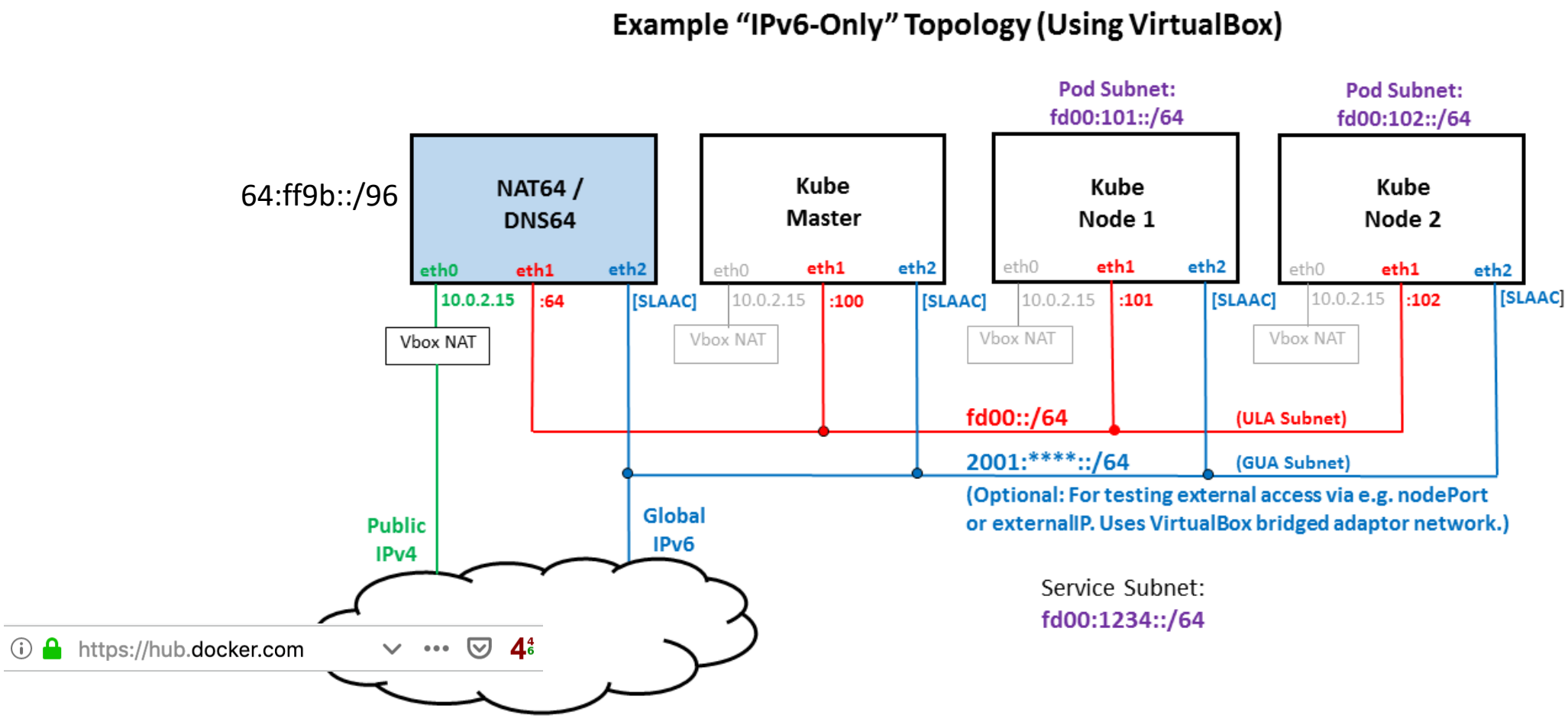
Dual stack KEP:
https://github.com/kubernetes/enhancements/blob/master/keps/sig-network/20180612-ipv4-ipv6-dual-stack.md#implementation-plan

IPv4 Kubernetes



Source: https://itnext.io/kubernetes-networking-behind-the-scenes-39a1ab1792bb from Nicolas Leiva

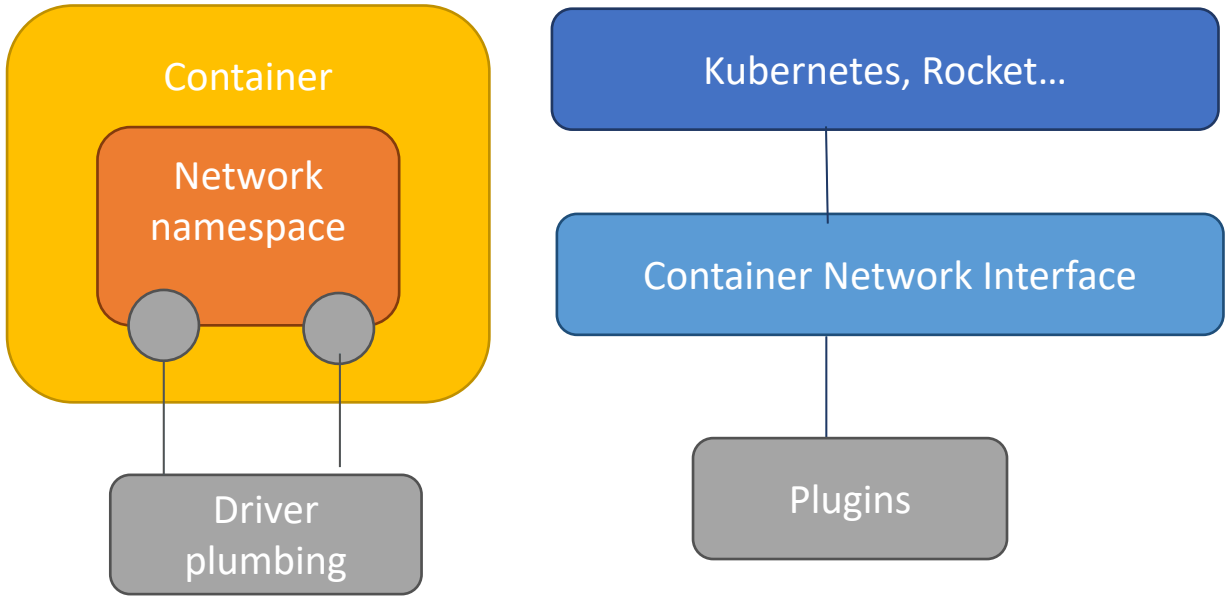
Multi-node, IPv6-only K8 cluster



Guide: https://github.com/leblancd/kube-v6

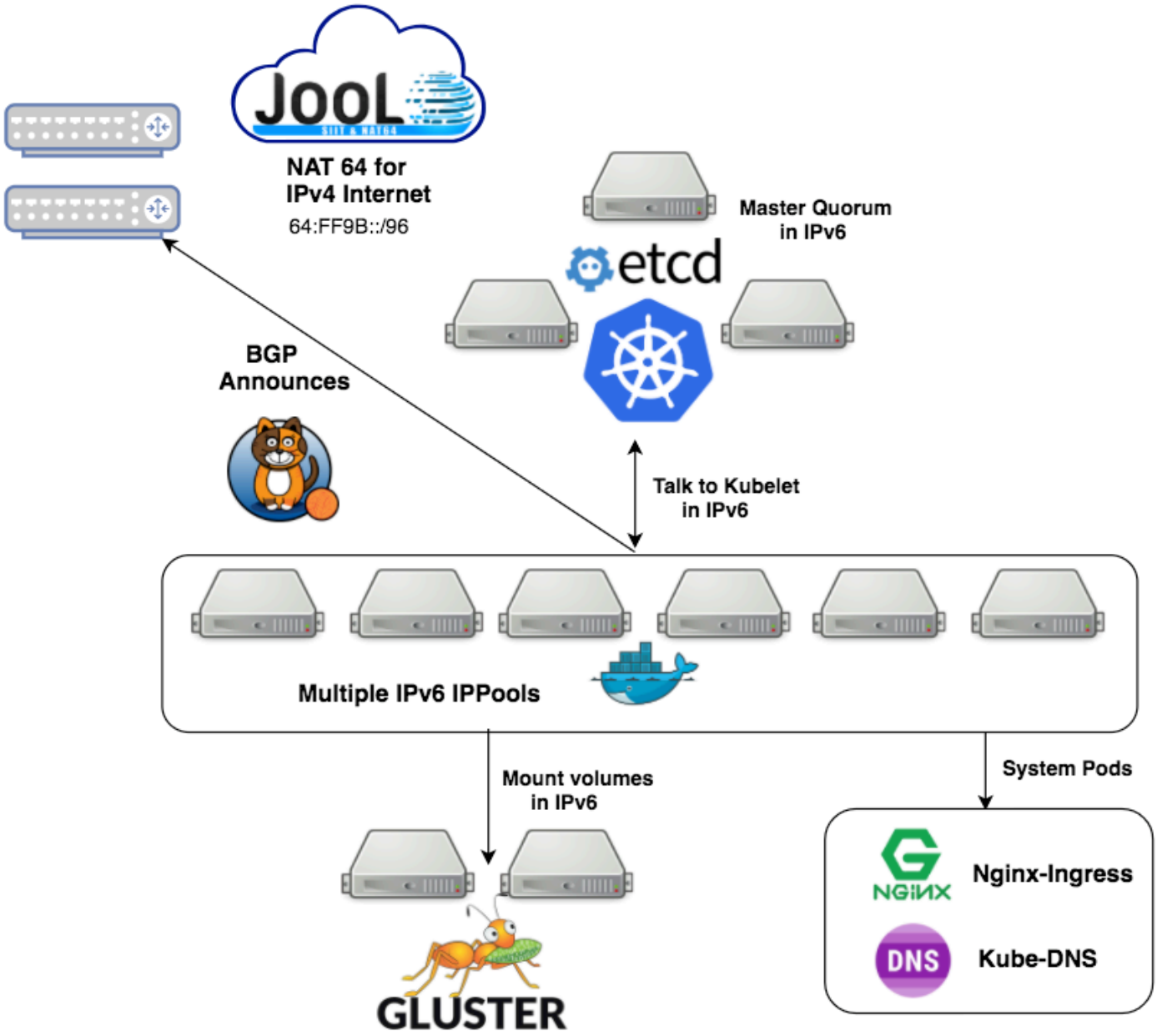
Container Network Interface (CNI)

- Proposed by CoreOS as part of appc specification
- Common interface between container run time and network plugin
- Gives driver freedom to manipulate network namespace
- Network described by JSON config
- Many CNI plugins available:
  - Calico, Flannel, Weave, Contiv…



CNI: Calico

- Pure L3 networking with BGP
- IPv6 only clusters
- ULA range by default for PODs
- By default breaks into /122 per node
- clusterIP: None on every defined Service
- Nginx-ingress controller



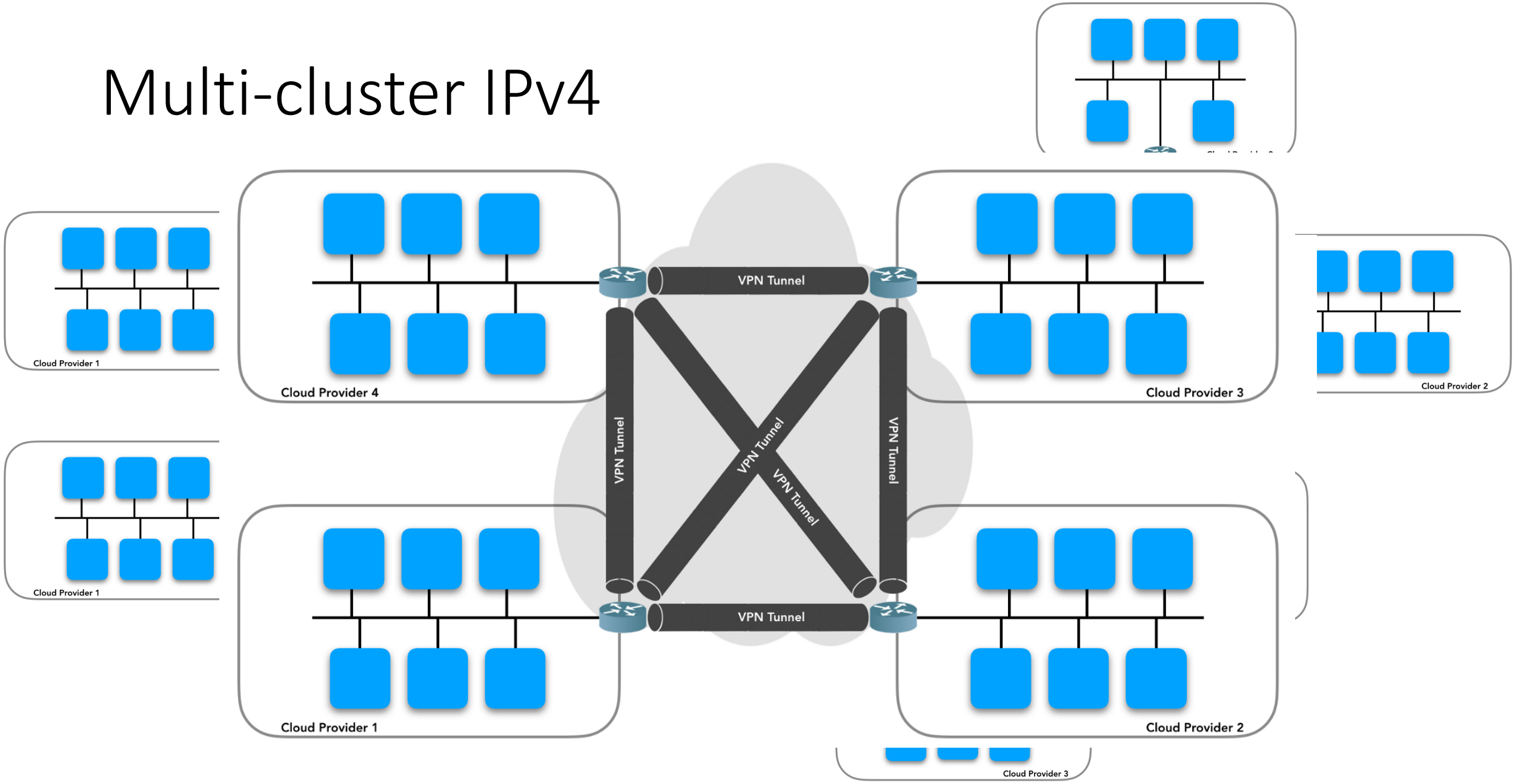
https://opsnotice.xyz/kubernetes-ipv6-only/
https://www.projectcalico.org/enable-ipv6-on-kubernetes-with-project-calico/

CNI: Contiv-VPP

- Allocates IP addresses to Pods (IPAM)
- Programs the underlying infrastructure it uses (Linux TCP/IP stack, OVS, VPP, ...) to connect the Pods to other Pods in the cluster and/or to the external world.
- Implements K8s network policies that define which pods can talk to each other.
- Implements K8s services; a service exposes one or more (physical) service instances implemented as K8s pods to the other pods in the cluster and/or to external clients as a virtual instance (e.g. as a virtual “service” IP address).
- VPP fast data processing runs completely in user space
- DPDK for fast access to network IO layer
- IPv6 with VXLAN
- SRv6

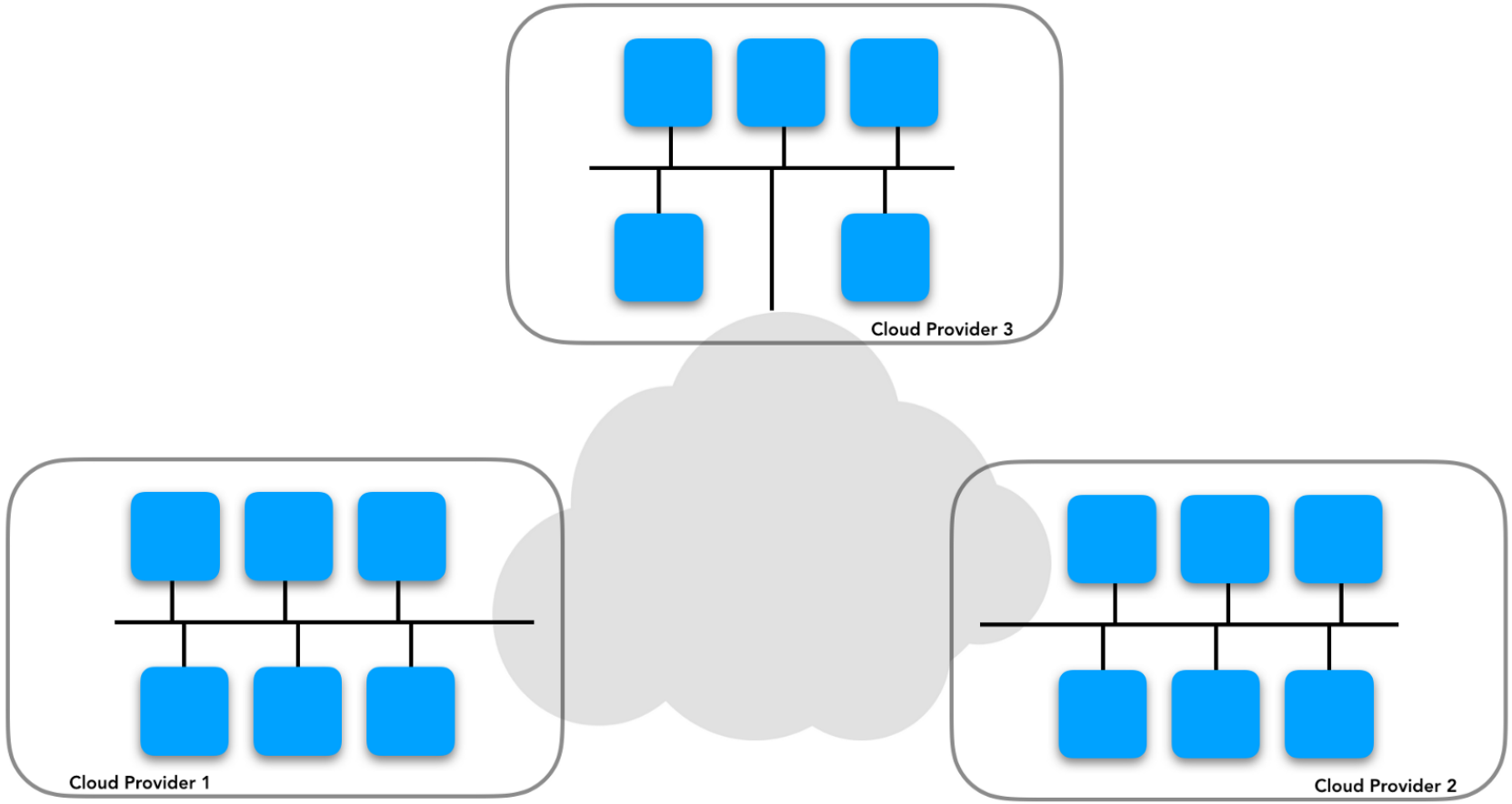
https://github.com/contiv/vpp

Multi-cluster IPv4



Source: https://itnext.io/kubernetes-multi-cluster-networking-made-simple-c8f26827813 from Nicolas Leiva

Multi-cluster IPv6



Source: https://itnext.io/kubernetes-multi-cluster-networking-made-simple-c8f26827813 from Nicolas Leiva

The scale of IPv6 for containers

- Every docker host a routed /64
- **Never** re-use IPv6 address again
- How long would it take to burn through that /64?
- How about 10,000,000 per second ?
- A standard /64 prefix in IPv6 is 18,446,744,073,709,600,000 addresses.
- 18,446,744,073,709,600,000 IPv6 addresses / (10,000,000 IPv6 addresses/second * 60 sec/min * 60 min/hr * 24 hr/day * 365 days/yr) = 58,494 years
- A single /48 contains 65536 /64s
- 58,494 years * 65536 = 3,833,478,626 (3.8 *billion* years)

Ed Horley (VP engineering Groupware)
http://www.howfunky.com/2015/06/ipv6-docker-and-building-for-scale.html

Where do I track the latest?

- https://github.com/kubernetes/enhancements/issues/508
- https://github.com/kubernetes/enhancements/issues/563
- https://github.com/kubernetes/enhancements/blob/master/keps/sig-network/20180612-ipv4-ipv6-dual-stack.md
- https://discuss.kubernetes.io/t/kubernetes-ipv4-ipv6-dual-stack-support-status/4974
- Phase 1 KEP: https://github.com/kubernetes/kubernetes/pull/73977
- Phase 2 KEP: https://github.com/kubernetes/kubernetes/pull/79386
- #k8s-dual-stack channel on Kubernetes.slack.com

Thanks!