Container and Kubernetes Networking 101

Before we begin

Slide deck: https://goo.gl/qmYMPE

Poll

Arun Sriraman
(Software Engineer)
Platform9 Systems Inc
@arun_sriraman sarun87

Agenda

Part I - Container Networking 101

- Why container networking?
- Linux namespaces
- container-to-container communication
- Docker n/w'ing - CNM model
- libnetwork & driver types

Part II - Kubernetes Networking 101

- K8s networking fundamentals
- Kubernetes communication
  - Container-to-Container
  - Pod-to-Pod
  - Pod-to-Service
  - Service-to-external
- Container Network Interface
- CNI backend (Flannel, Calico)

Part I - Container Networking

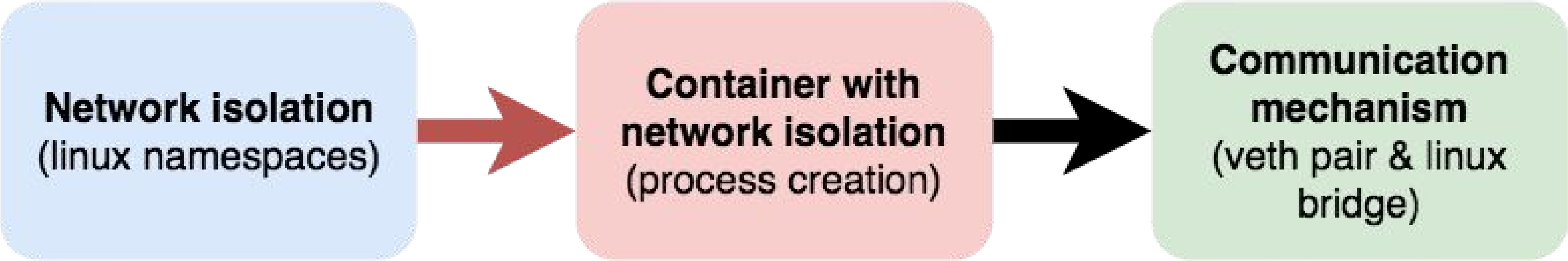
Why container networking?

- container ←→ container communication
- container ←→ host communication
- user ←→ container communication

Why should we learn the nuts & bolts of the system?

- troubleshooting
- debugging
- building solutions (Service Discovery, Load Balancing)

3 steps to container networking



Network isolation
(linux namespaces)

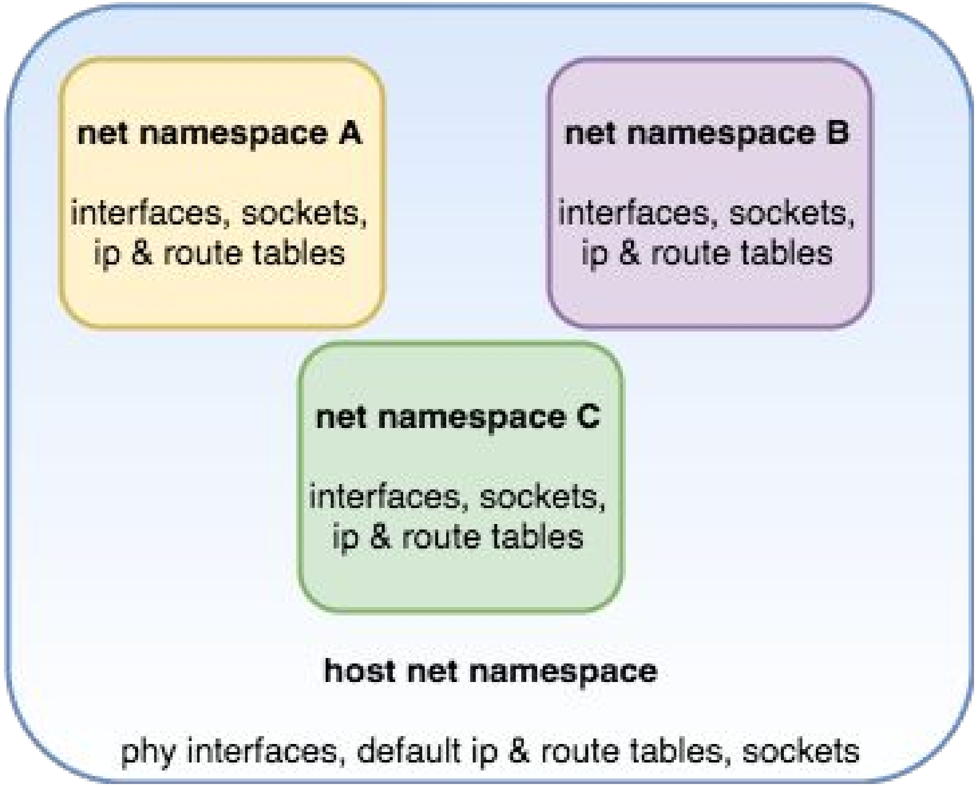
Container with network isolation
(process creation)

Communication mechanism
(veth pair & linux bridge)

Network Isolation - Network namespaces

A network namespace gets its own private network stack with

- network interfaces (including lo)
- routing tables
- iptables rules
- sockets (ss, netstat)



net namespace A

interfaces, sockets, ip & route tables

net namespace B

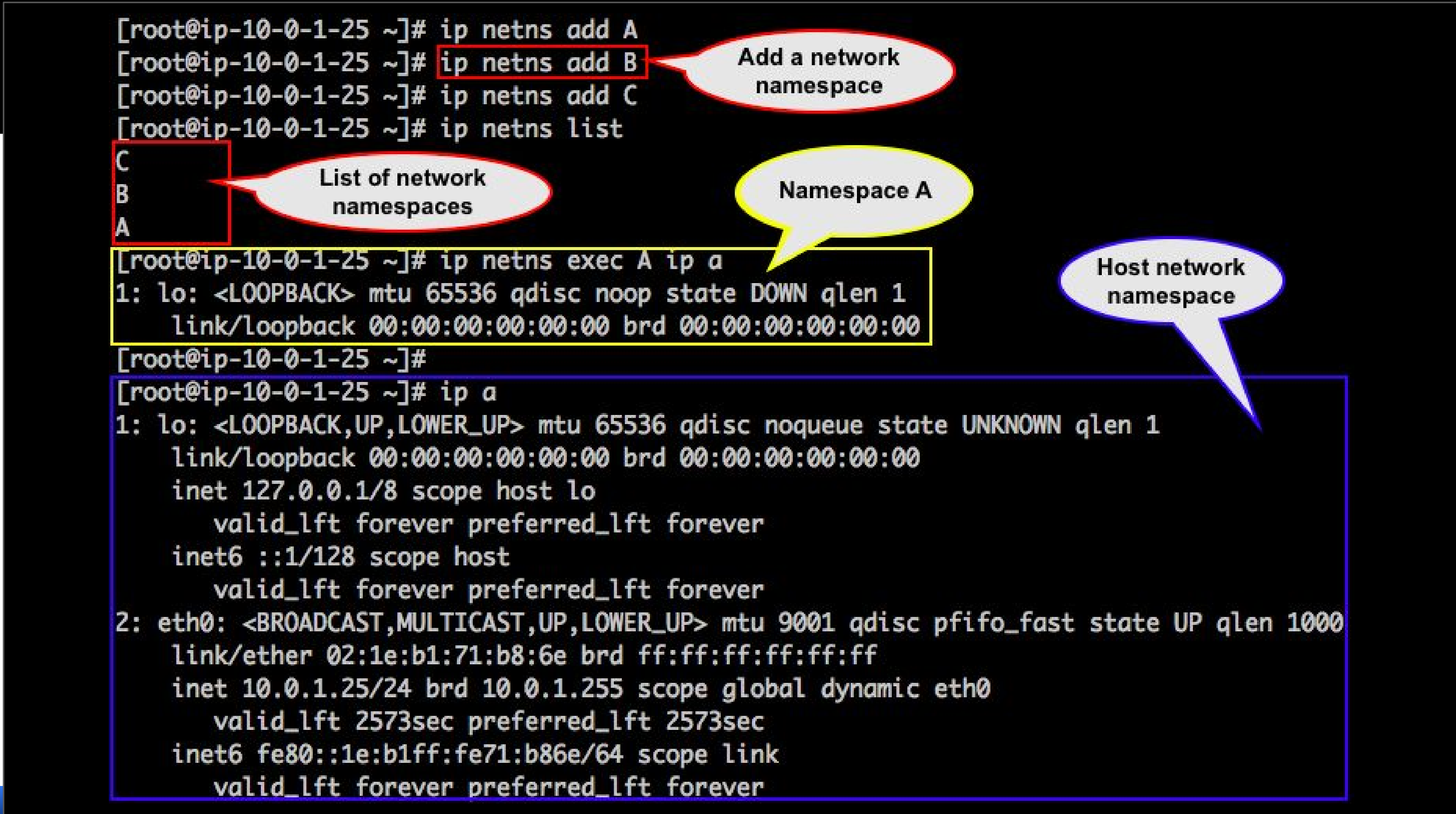
interfaces, sockets, ip & route tables

net namespace C

interfaces, sockets, ip & route tables

host net namespace

phy interfaces, default ip & route tables, sockets



```
[root@ip-10-0-1-25 ~]# ip netns add A
[root@ip-10-0-1-25 ~]# ip netns add B
[root@ip-10-0-1-25 ~]# ip netns add C
[root@ip-10-0-1-25 ~]# ip netns list
C
B
A
[root@ip-10-0-1-25 ~]# ip netns exec A ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
[root@ip-10-0-1-25 ~]#
[root@ip-10-0-1-25 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP qlen 1000
    link/ether 02:1e:b1:71:b8:6e brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.25/24 brd 10.0.1.255 scope global dynamic eth0
       valid_lft 2573sec preferred_lft 2573sec
    inet6 fe80::1e:b1ff:fe71:b86e/64 scope link
       valid_lft forever preferred_lft forever
```

Add a network namespace

List of network namespaces

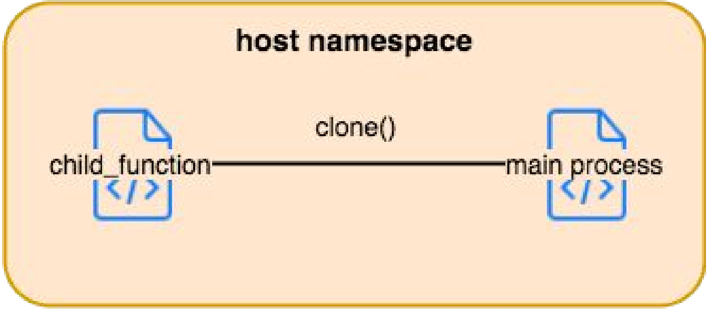
Namespace A

Host network namespace

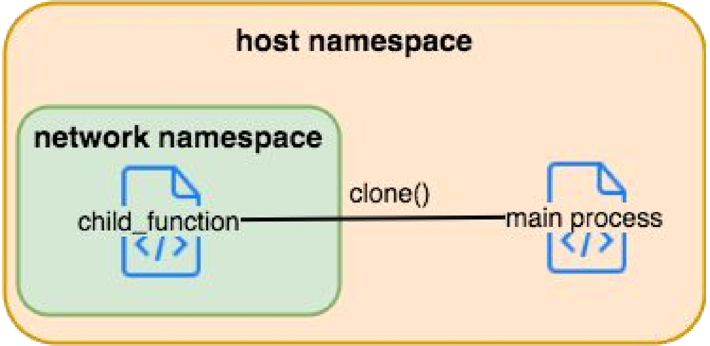
Container and isolation

```
flags = CLONE_NEWPID|
    CLONE_NEWNS|CLONE_NEWNET;
cpid = clone(child_function,
    childstack,
    flags, (void *)argv);
```

without CLONE_NEWNET flag

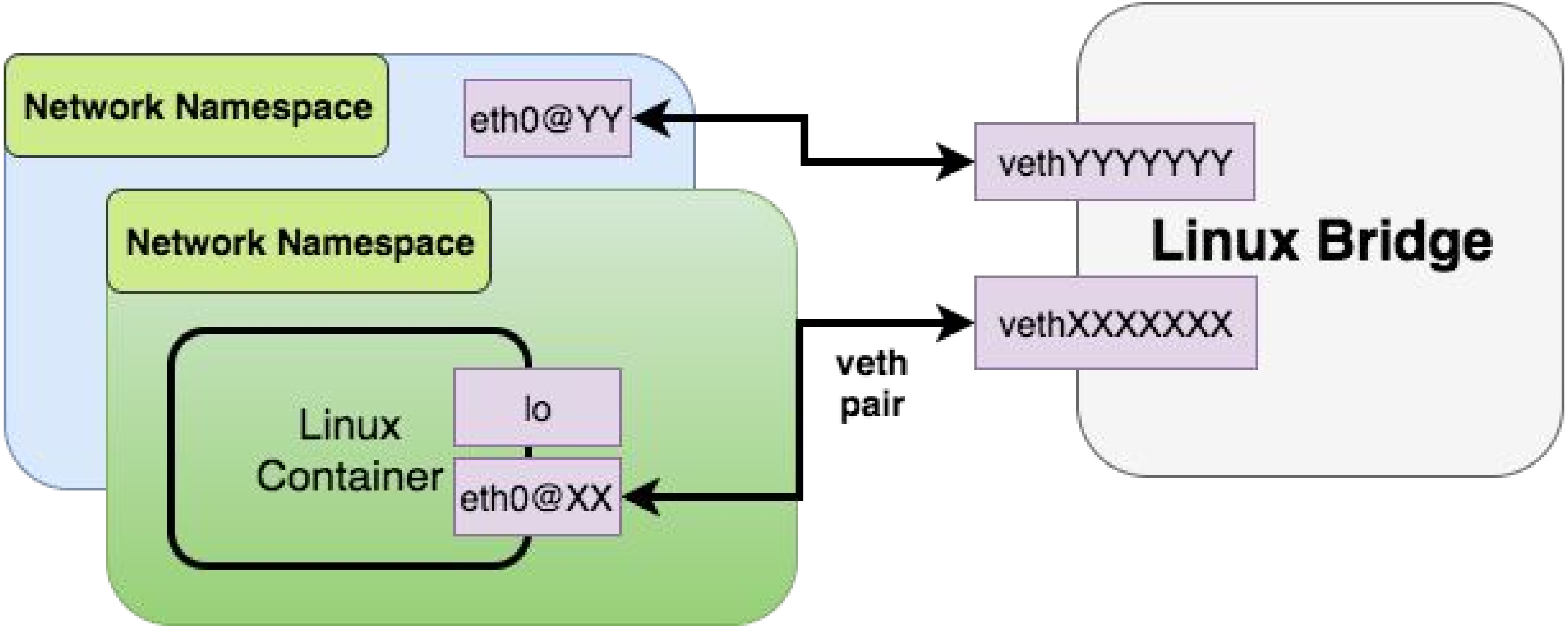


with CLONE_NEWNET flag



Communication mechanism

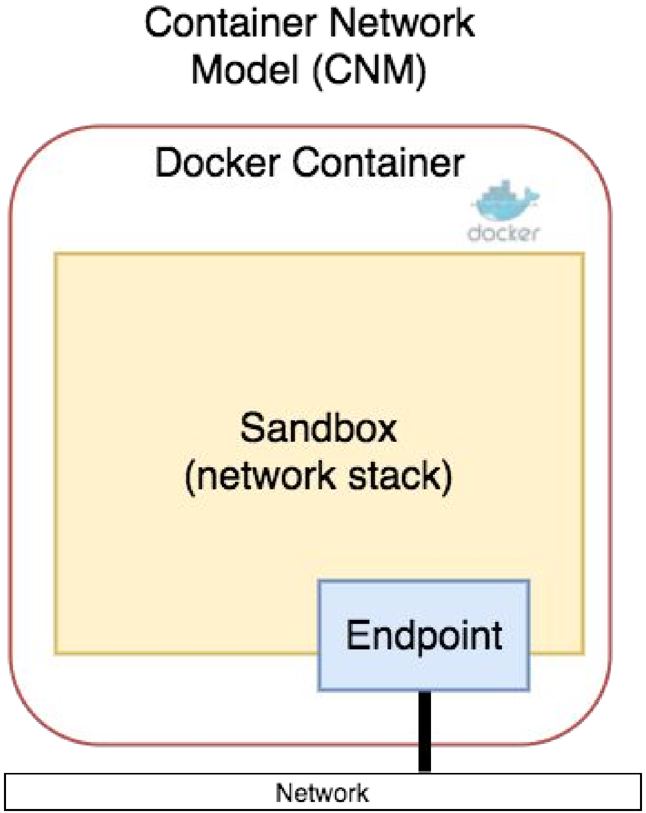
Linux bridge and veth pairs



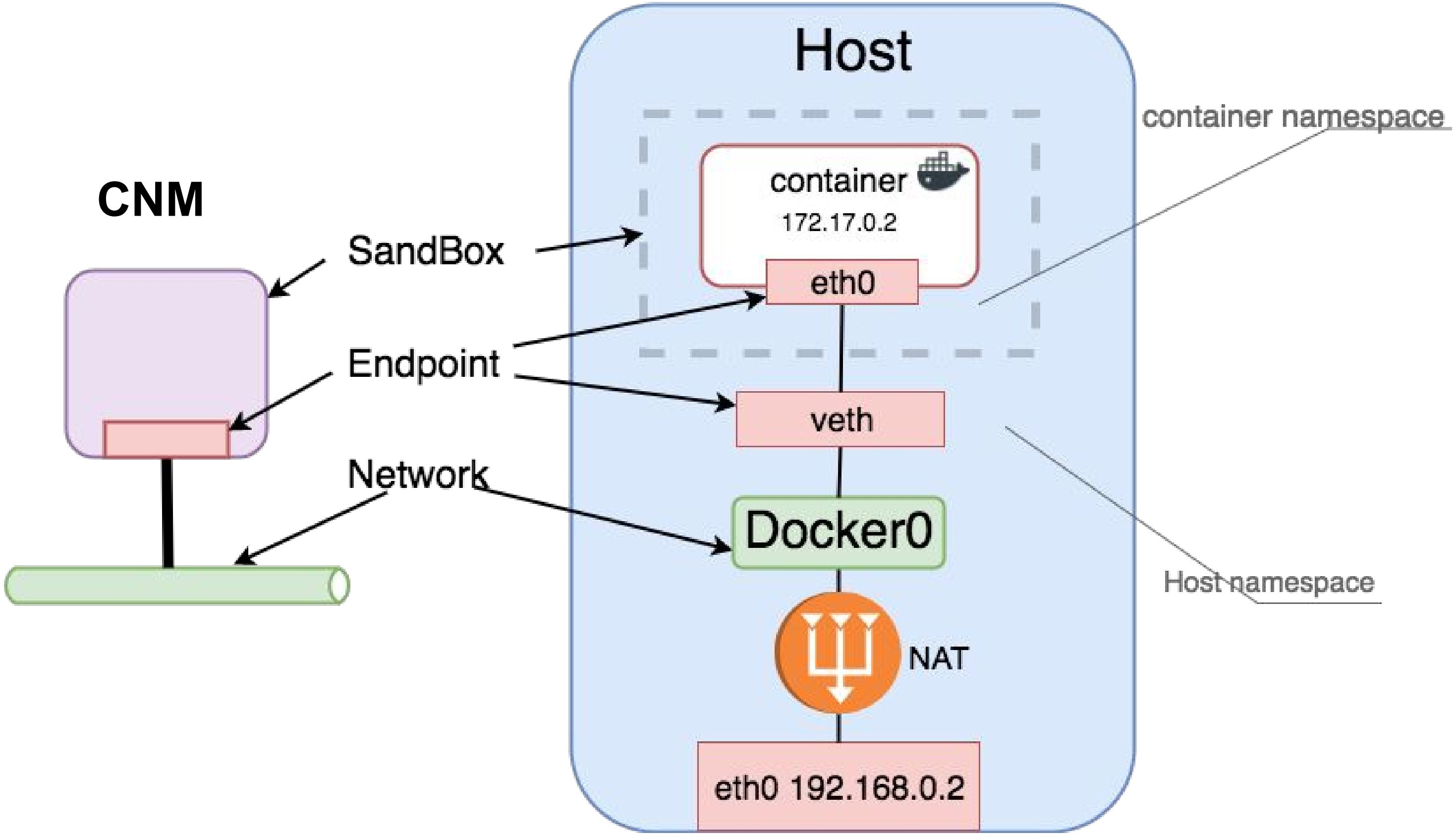
http://blog.arunsriraman.com/2017/01/container-namespaces-deep-dive-into.html

Container Network Model (CNM)

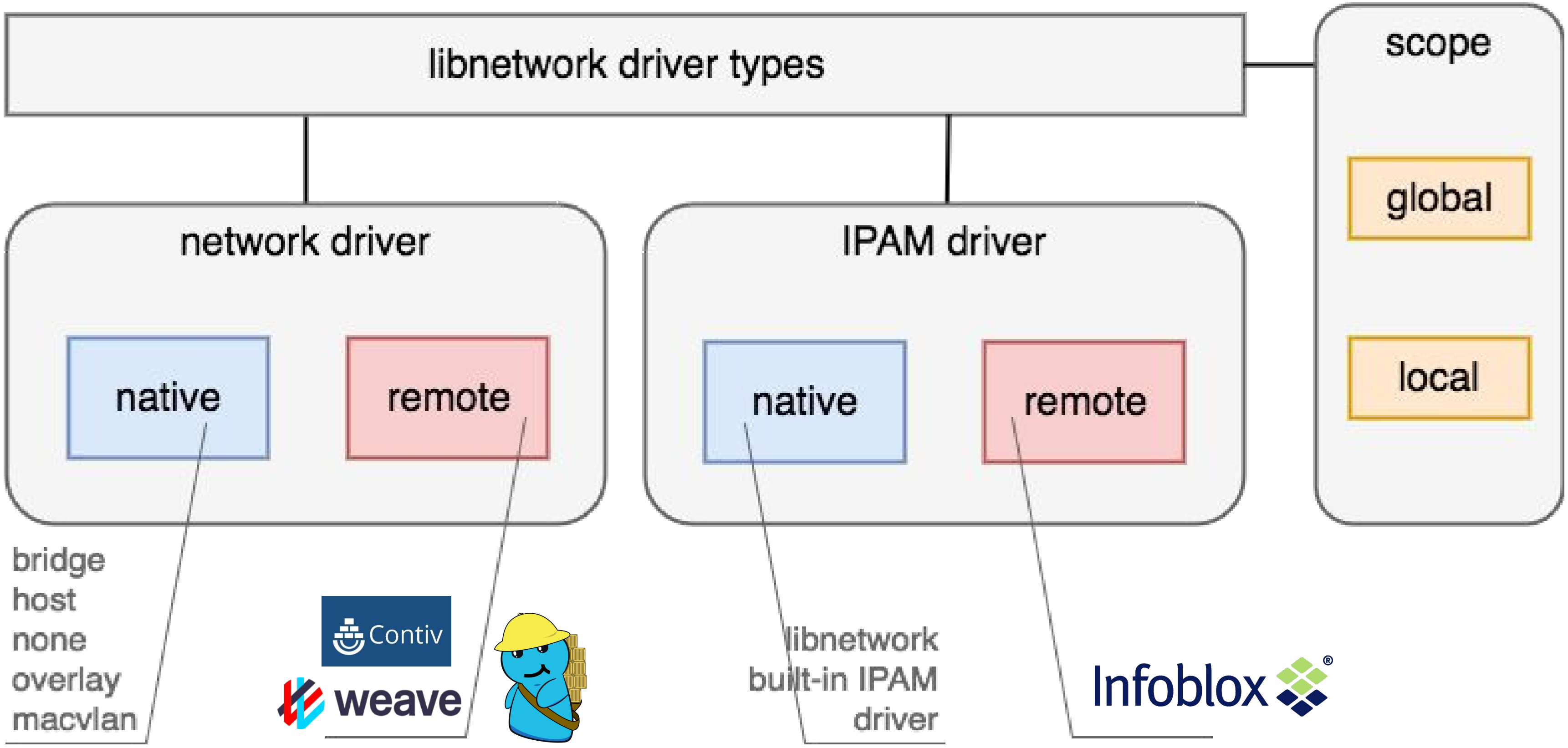
- Project started by Docker
- Separate networking from container runtime as a library
- Components
  - Sandbox
  - Endpoint
  - Network
- Implemented using libnetwork



Mapping CNM to Libnetwork (Docker)



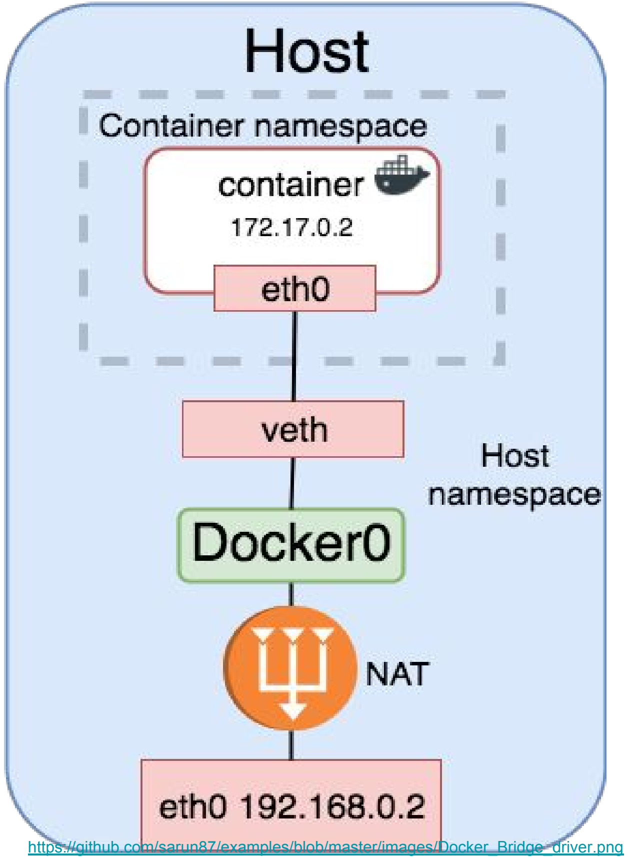
libnetwork driver types



Bridge Driver

- Connects docker containers to the network using a veth pair
- Provides out-of-the-box support for bridge based container networking
- Allows creation of user-defined bridges

```
docker network create --driver bridge
<name>
```



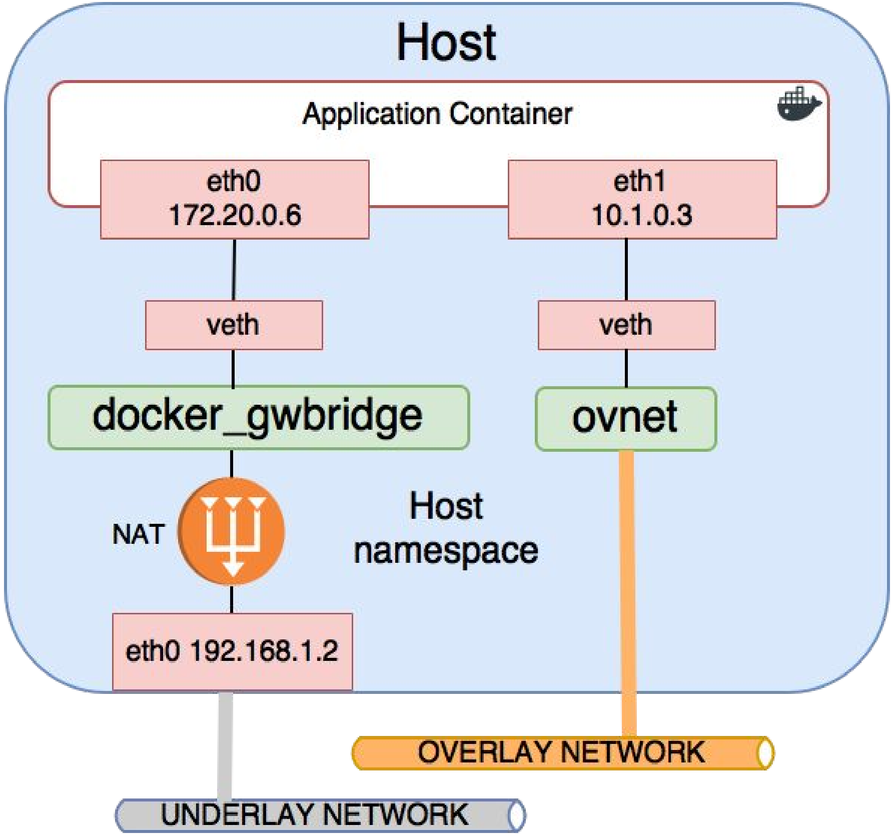
https://github.com/sarun87/examples/blob/master/images/Docker_Bridge_driver.png

External Access for Containers

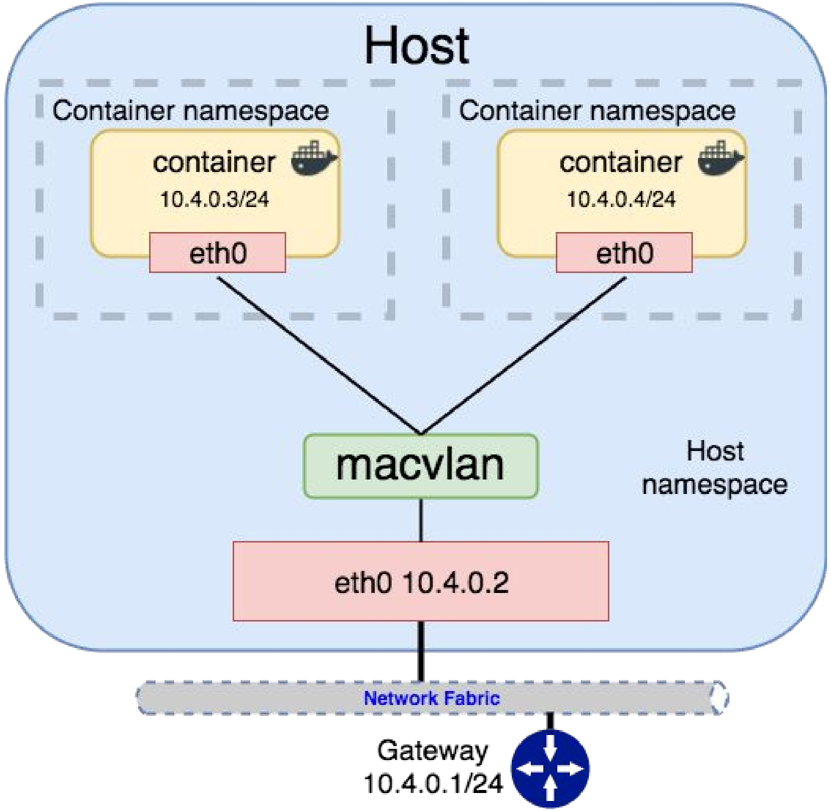
port mapping & NAT

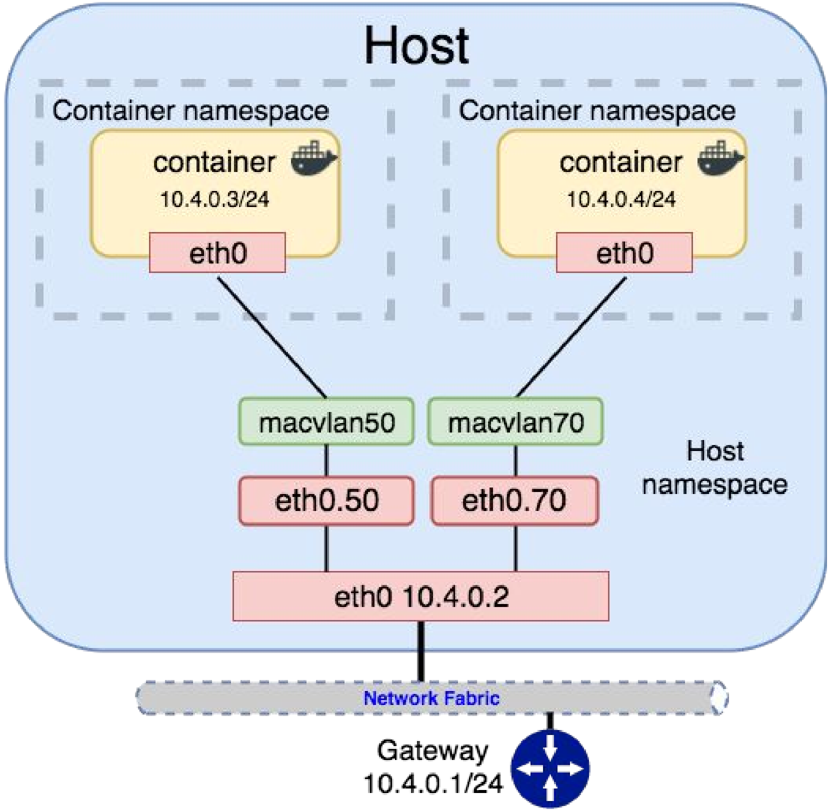
Overlay Driver

- Multi-host networking
- First-class citizen in docker networking
- Uses swarm-distributed control plane for centralized mgmt, stability & security
- Uses VXLAN encap (decouples container n/w from physical n/w)
- Overlay datapath entirely in kernel space



Macvlan Driver





Default Networks Created by Docker

‘bridge’ using **bridge** driver, ‘none’ using **null** driver, ‘host’ using **host** driver

```
arun-neotrekker:~ arunsriraman$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
544fd2b5b674        bridge              bridge              local
790b79d68240        host                host                local
6aaec591a006        none                null                local
```

Don’t want the bridge driver? Remove it by specifying OPTIONS

```
/etc/sysconfig/docker
OPTIONS="--bridge=none --log-driver=json-file"
```

Recap - Docker network drivers

Part II - Kubernetes Networking

Fundamental requirements

All containers can communicate with all other containers without NAT

All nodes can communicate with all containers (and vice-versa) without NAT

The IP that a container sees itself as is the same IP that others see it as

Quoted from K8s docs

Kubernetes networking

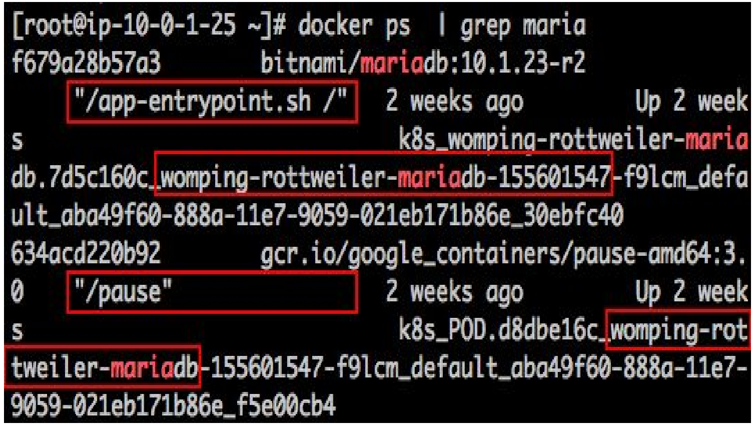
- Container-to-Container communication
- Pod-to-Pod communication
- Pod-to-Service (cluster internal) communication
- External-to-Service (cluster external) communication

Container CIDR
Service CIDR

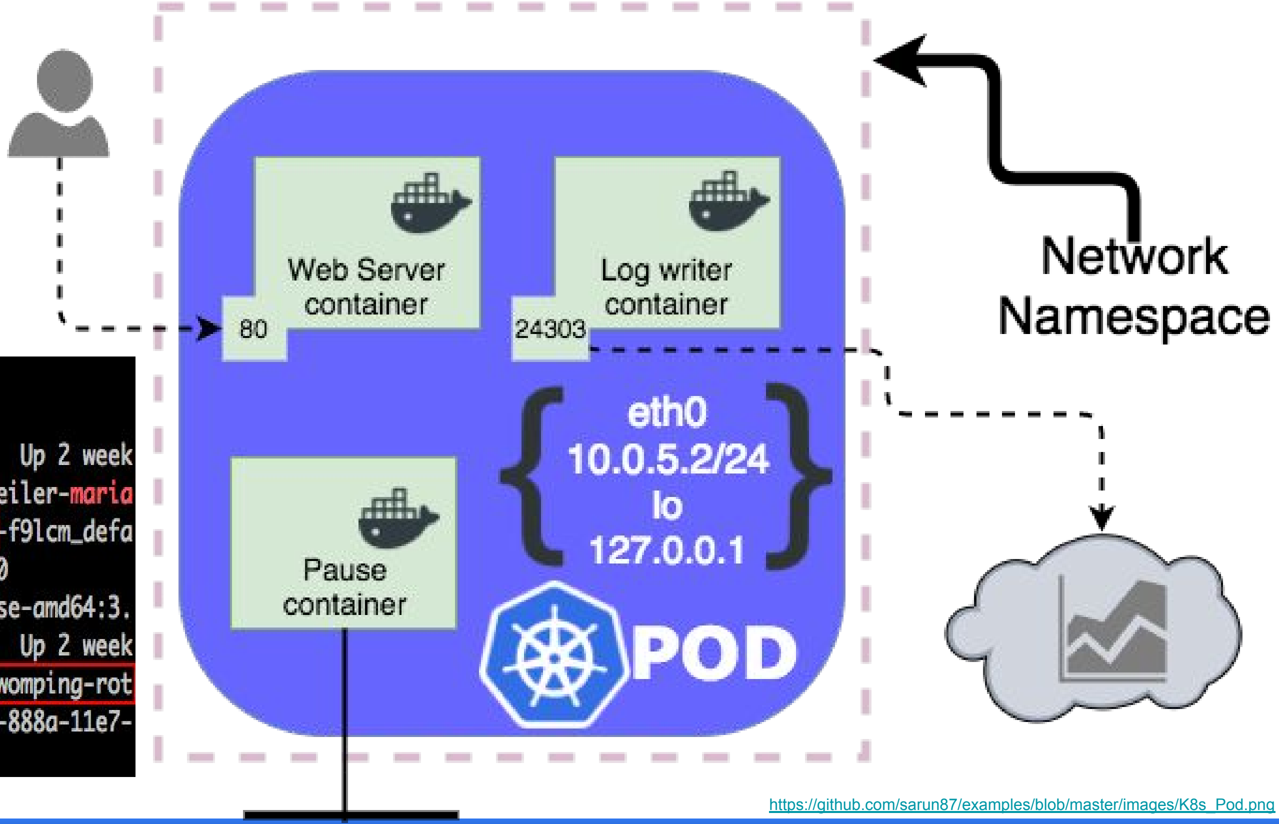
Container-to-Container

Pod

Group of one or more containers with shared storage/network



```
[root@ip-10-0-1-25 ~]# docker ps  | grep maria
f679a28b57a3        bitnami/mariadb:10.1.23-r2
    "/app-entrypoint.sh /"   2 weeks ago         Up 2 week
s                          k8s_womping-rottweiler-maria
db.7d5c160c_womping-rottweiler-mariadb-155601547-f9lcm_defa
ult_aba49f60-888a-11e7-9059-021eb171b86e_30ebfc40
634acd220b92        gcr.io/google_containers/pause-amd64:3.
0   "/pause"                 2 weeks ago         Up 2 week
s                          k8s_POD.d8dbe16c_womping-rot
tweiler-mariadb-155601547-f9lcm_default_aba49f60-888a-11e7-
9059-021eb171b86e_f5e00cb4
```

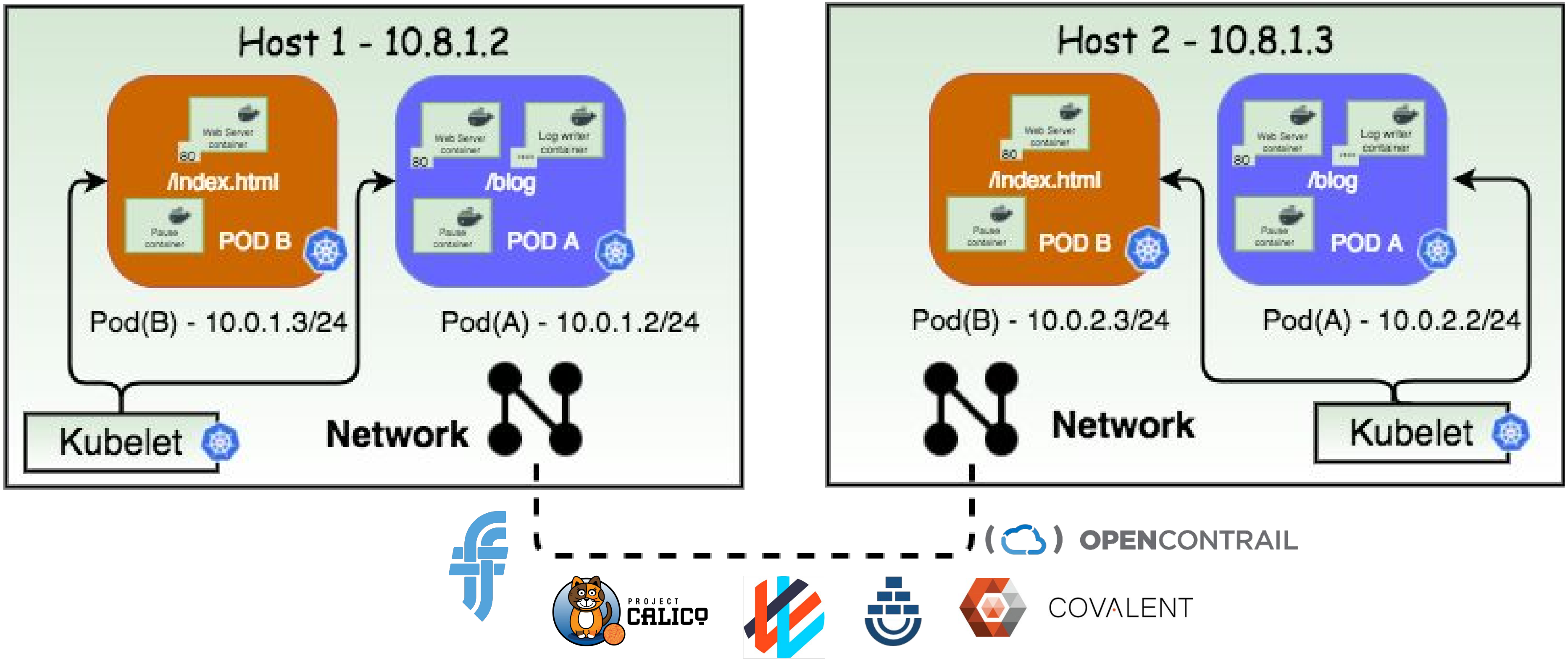


https://github.com/sarun87/examples/blob/master/images/K8s_Pod.png

Container-to-Container takeaways

- Containers in a pod run on the same host.
- A pod generally represents a service unit of an application.
- Uses localhost (127.0.0.1) within the pod’s network namespace to communicate with each other
- Containers in the same Pod cannot reuse ports ⚠
- Pause container - Keeps the networking alive
- New concepts: Pod, Pause container

Pod-to-Pod



https://github.com/sarun87/examples/blob/master/images/K8s_Pod_to_Pod.png

Pod-to-Pod takeaways

Currently supported networking models -

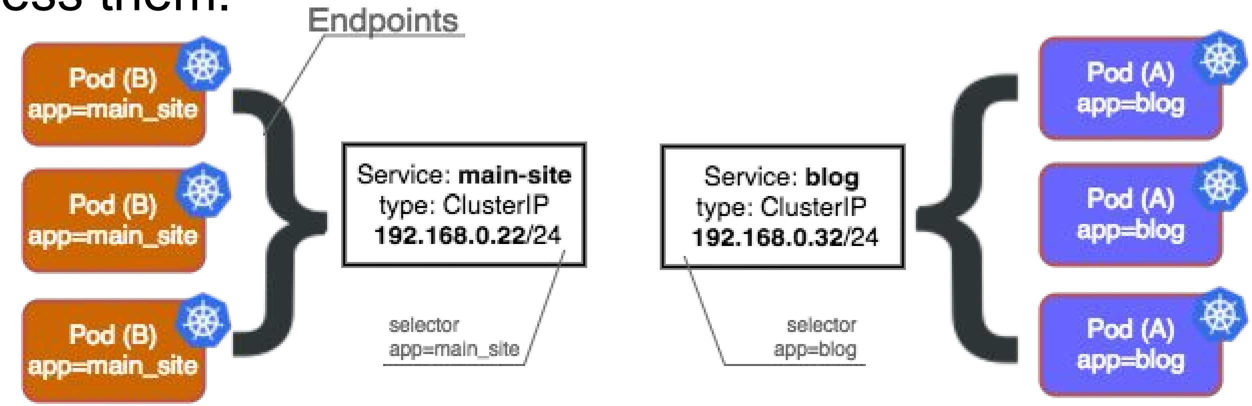
- Kubenet via kubelet (will be moved out to CNI)
- Multiple network backends via CNI (We'll discuss this in depth later)

Network backend responsible for -

- Pod networking setup
- Pod-to-Pod networking setup (uses L3 BGP like Calico, network overlay like weave, flannel)

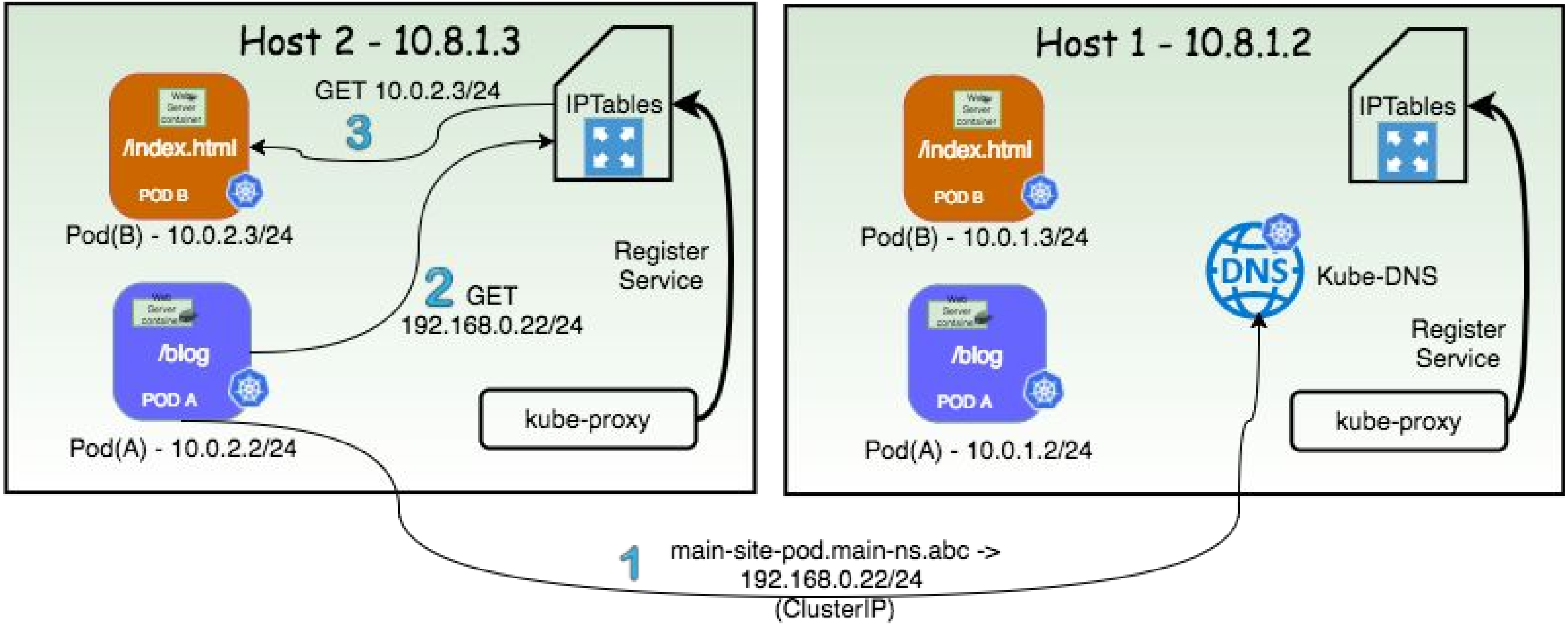
Kubernetes “Service” Primer

Service - an abstraction which defines a logical set of Pods and a policy by which to access them.



- A service is “generally” backed by pods (endpoints) using a “label selector”.
- Users can explicitly define an endpoint that isn’t backed by pods
- K8s defines many types of services
  - Internal: ClusterIP
  - External: NodePort, LoadBalancer, Ingress

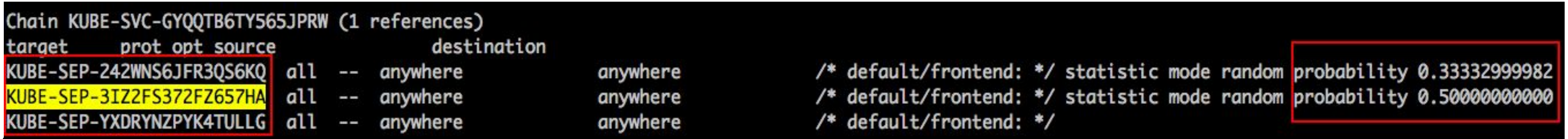
Pod-to-Service (Cluster Internal)



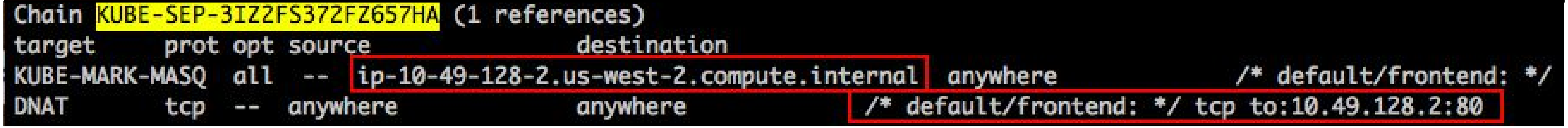
https://github.com/sarun87/examples/blob/master/images/K8s_Pod_to_Service.png

Pod-to-Service takeaways

- Service is a logical definition/collection of pods.
- ClusterIP is allocated from the Services CIDR
- kube-proxy modes
  - userspace
  - iptables (our discussed example)
- New concepts: kube-proxy, kube-dns, Service, clusterIP,

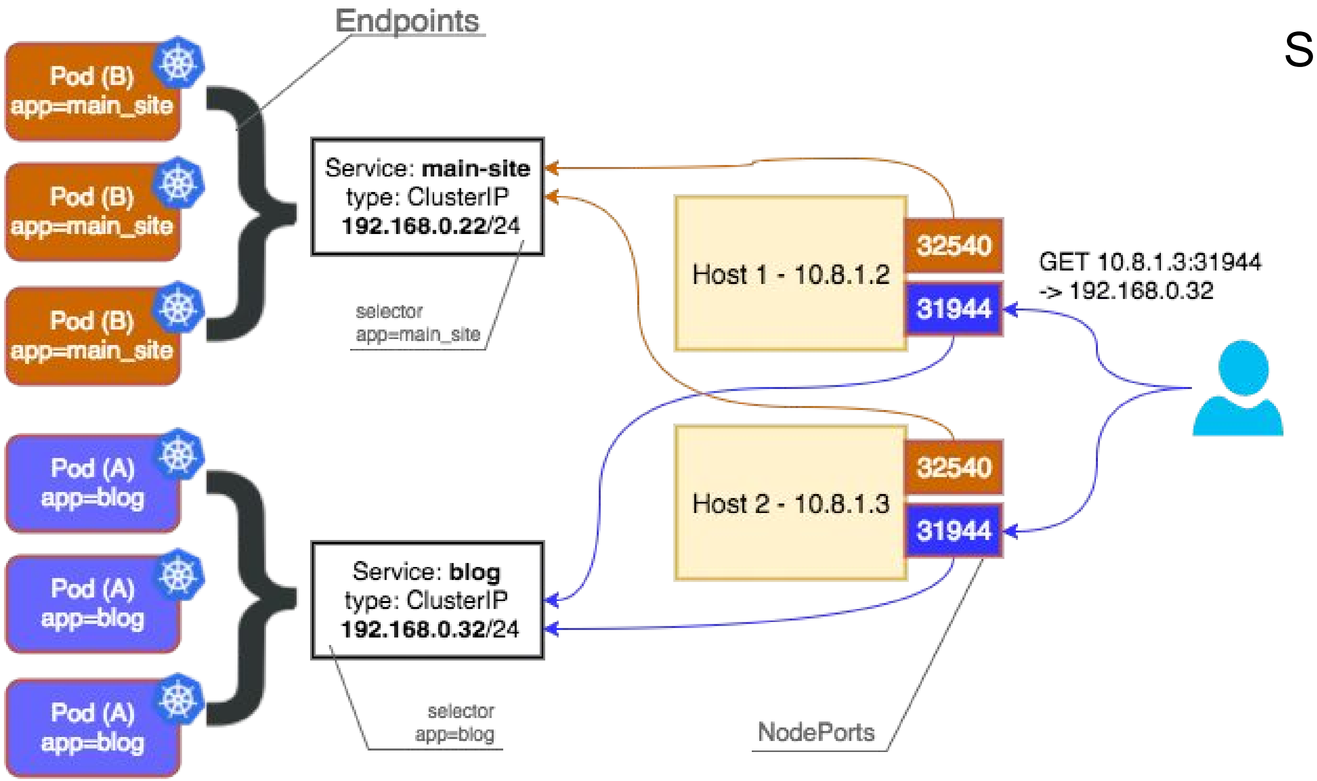


```
Chain KUBE-SVC-GYQQTB6TY565JPRW (1 references)
target     prot opt source               destination
KUBE-SEP-242WNS6JFR3QS6KQ  all  --  anywhere             anywhere             /* default/frontend: */ statistic mode random probability 0.33332999982
KUBE-SEP-3IZ2FS372FZ657HA  all  --  anywhere             anywhere             /* default/frontend: */ statistic mode random probability 0.50000000000
KUBE-SEP-YXDRYNZPYK4TULLG  all  --  anywhere             anywhere             /* default/frontend: */
```



```
Chain KUBE-SEP-3IZ2FS372FZ657HA (1 references)
target     prot opt source               destination
KUBE-MARK-MASQ  all  --  ip-10-49-128-2.us-west-2.compute.internal  anywhere             /* default/frontend: */
DNAT       tcp  --  anywhere             anywhere             /* default/frontend: */ tcp to:10.49.128.2:80
```

External-to-Service

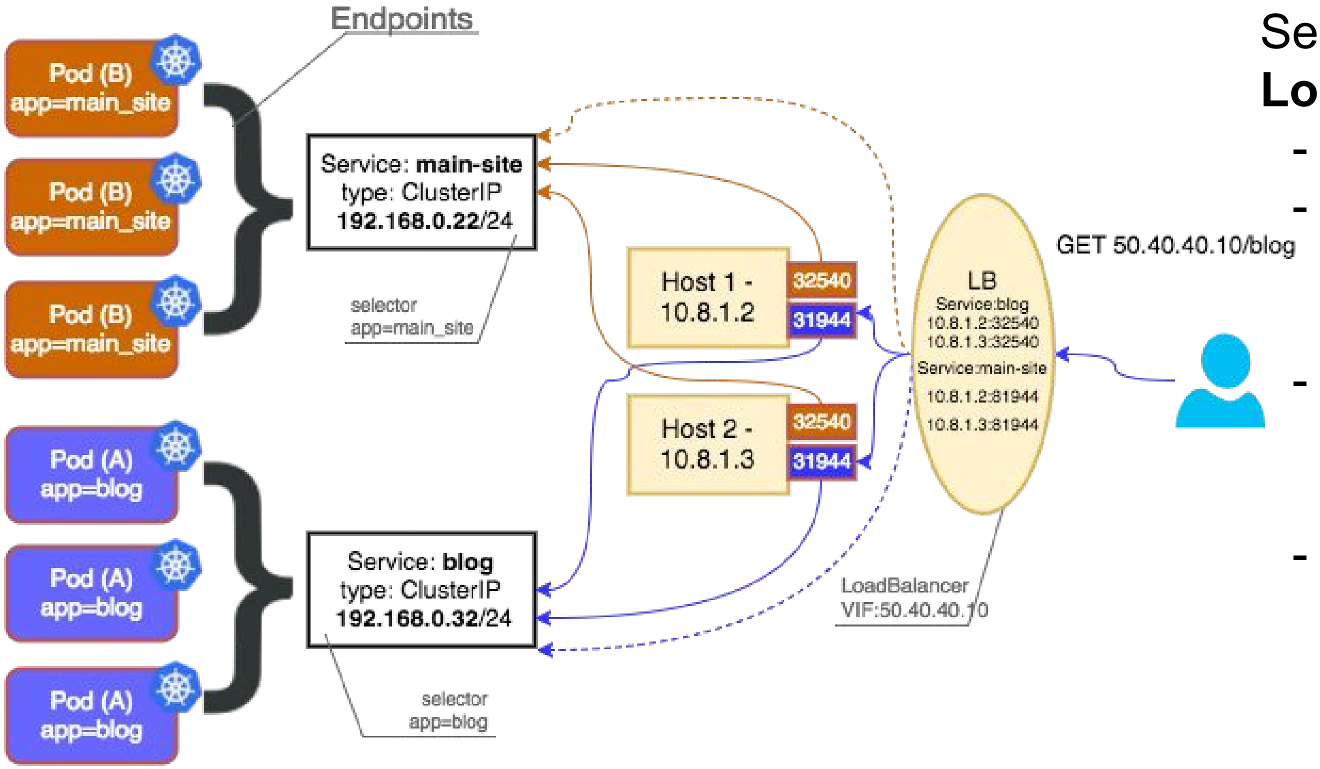


Service type: **NodePort**

- Kubernetes master allocates a port from a flag-configured range (default: 30000-32767).
- Each Node will proxy that port (the same port number on every Node) into your Service

https://github.com/sarun87/examples/blob/master/images/K8s_NodePort.png

External-to-Service - II



Service type:
LoadBalancer

- Fronts the K8s Service
- Traffic from load balancer is directed to backend Pods
- Exactly how that works depends on the cloud provider
- NodePort and ClusterIP to which LB will route are created automatically

External-to-Service III

Ingress

- An Ingress is a collection of rules that allow inbound connections to reach the cluster services.
- Ingress is useful since services typically have internal IPs/endpoints
- All traffic that ends up at an edge router is either dropped or forwarded elsewhere
- Gives services externally-reachable URLs, load balance traffic, terminate SSL, offer name based virtual hosting

External IPs

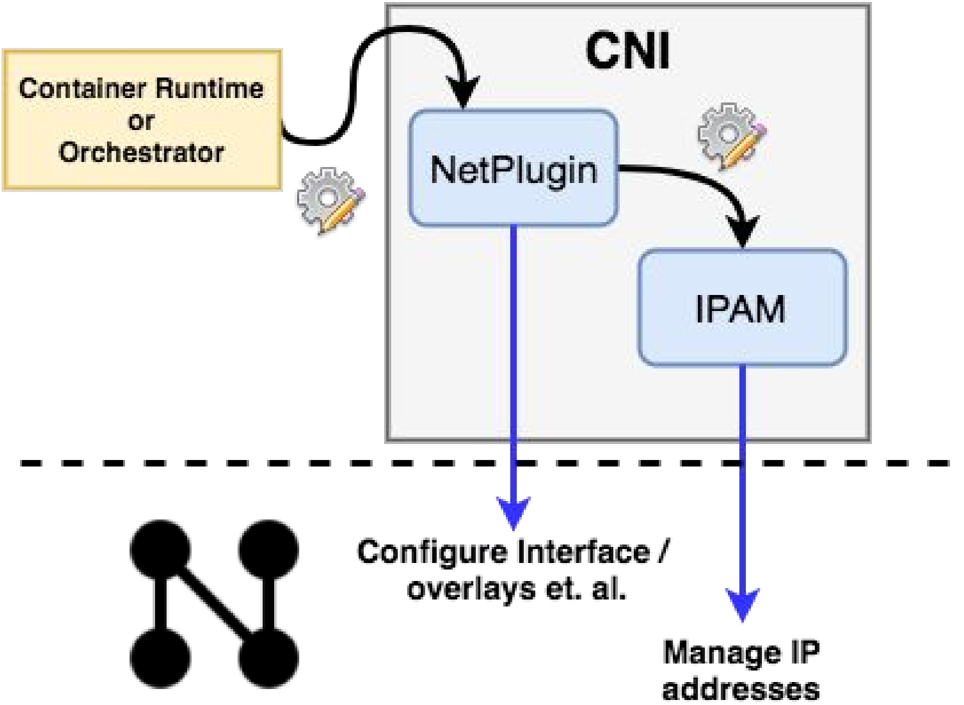
- A public/external IP points to a node of the cluster
- Service ingresses the requests from the external IP
- Are not managed by K8s

Note: If you came here to understand ingress specifically, let's chat offline. I will cover this if time permits

CNI - Container Network Interface

- Simple interface between container runtime & network
- **CNCF** project. Started by CoreOS for the **rkt** runtime

- Config passed to the NetPlugin by runtime then passed to IPAM
- CNI Interfaces - ADD, DEL



CNI - plugins

CNI Maintained

Plugins that create/delete interfaces

- bridge
- ipvlan
- lo
- macvlan
- vlan
- ptp

IPAM - IP address management

- dhcp

3rd party/others

- flannel (now under CNI)
- calico
- canal
- weave
- Cilium
- Contrail
- Contiv
- Infoblox
- Romana
- Nuage
-

Github repo - https://github.com/containernetworking/cni

Using CNI with individual containers

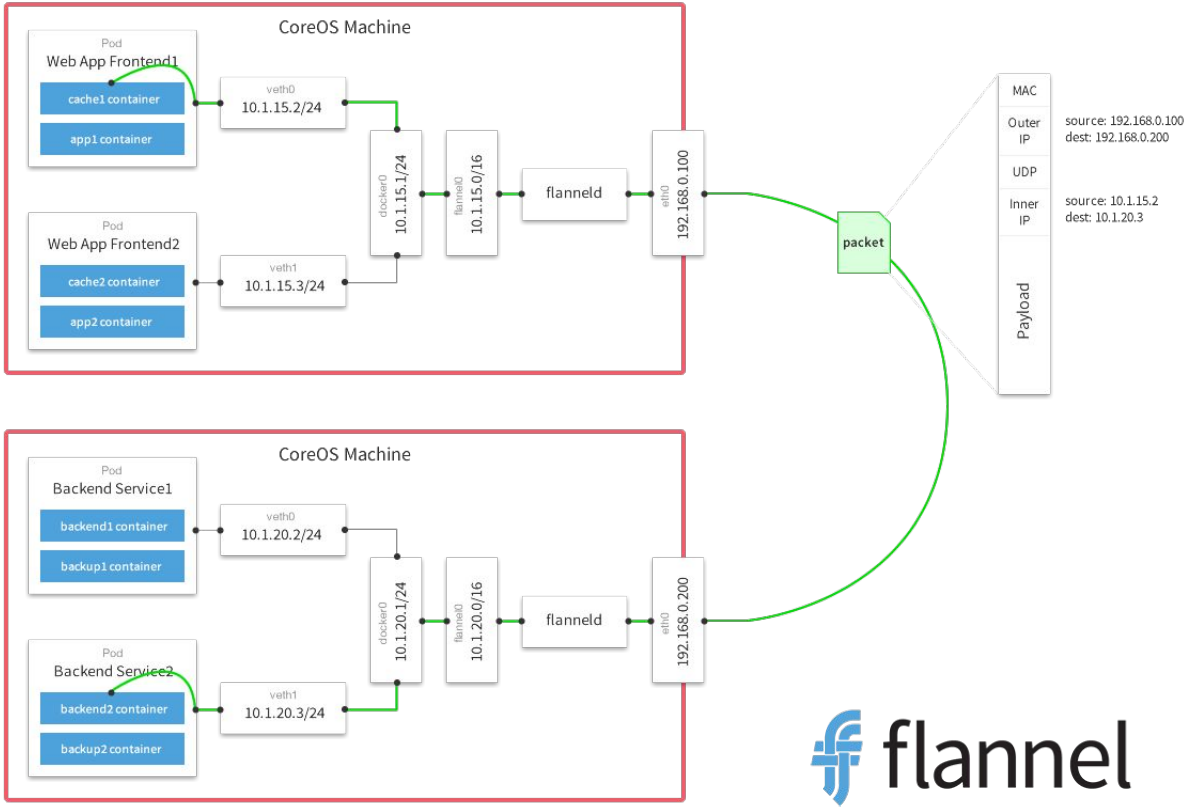
Eg: host-local IPAM. To ADD n/w to a container

```
$ CNI_COMMAND=ADD \
CNI_CONTAINERID=arun_container_01 \
CNI_NETNS=/var/run/netns/cni_ipam_eg \
CNI_IFNAME=eth0 \
CNI_PATH=/home/ubuntu/cni/bin \
./host-local < sample_ipam_config
```

```
{
        "cniVersion": "0.3.1",
        "ips": [{
                  "version": "4",
                  "address": "10.10.10.2/24",
                  "gateway": "10.10.10.1"
    }],
        "dns": {}
}
```

```
$ cat sample_ipam_config
{
  "cniVersion": "0.3.1",
  "name": "example-network",
  "ipam": {
    "type": "host-local",
    "subnet": "10.10.10.0/24”,
    "dataDir":
"/home/ubuntu/sample_ipam_datadir
"
  }
}
```

Flannel network backend



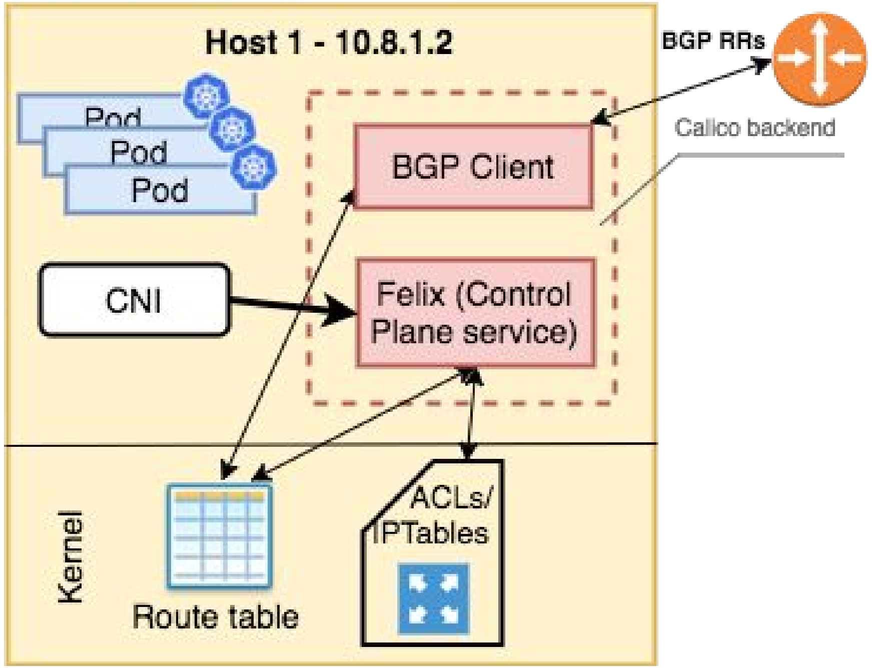
flannel

- Uses overlay network for host-host connectivity
- Backends - UDP, vxlan
- flanneld binary runs on every host
- Does **not** perform host - container networking.
- Via CNI, flannel delegates interface operations to bridge driver.

https://github.com/coreos/flannel/blob/master/packet-01.png

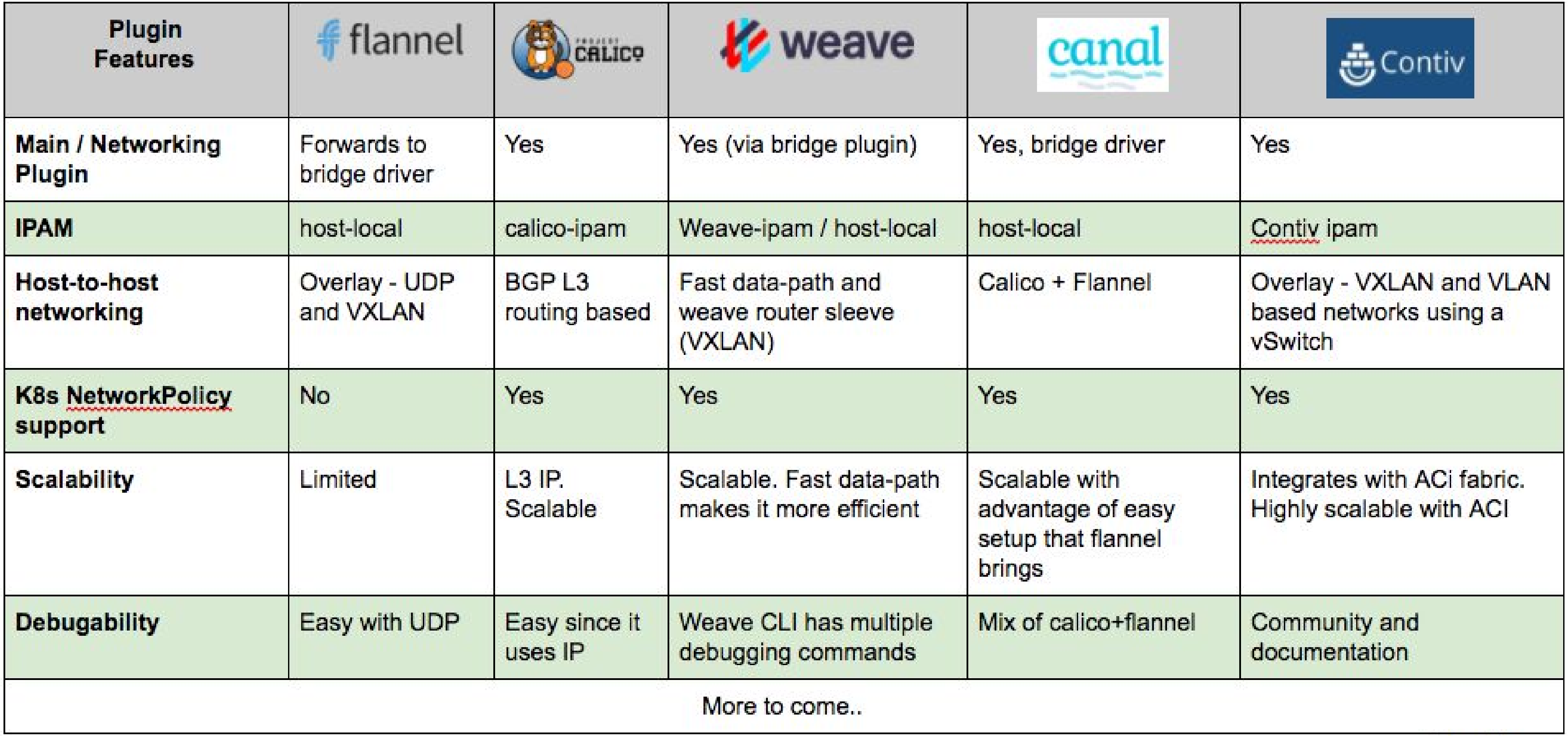
Calico network backend

- Pure L3 based network solution
- Router per node
- Uses BGP
- via CNI plugin - has its own IPAM driver as well
- Supports Kubernetes NetworkPolicy constructs
- BIRD protocol (BGP stack)
- ACL and L3 forwarding performed in the linux kernel
- Ease of debugging
- Scalable



https://github.com/sarun87/examples/blob/master/images/Calico_Architecture.png

CNI backends summarized



Plugin Features	flannel	PROJECT CALICO	weave	canal	Contiv
Main / Networking Plugin	Forwards to bridge driver	Yes	Yes (via bridge plugin)	Yes, bridge driver	Yes
IPAM	host-local	calico-ipam	Weave-ipam / host-local	host-local	Contiv ipam
Host-to-host networking	Overlay - UDP and VXLAN	BGP L3 routing based	Fast data-path and weave router sleeve (VXLAN)	Calico + Flannel	Overlay - VXLAN and VLAN based networks using a vSwitch
K8s NetworkPolicy support	No	Yes	Yes	Yes	Yes
Scalability	Limited	L3 IP. Scalable	Scalable. Fast data-path makes it more efficient	Scalable with advantage of easy setup that flannel brings	Integrates with ACi fabric. Highly scalable with ACI
Debugability	Easy with UDP	Easy since it uses IP	Weave CLI has multiple debugging commands	Mix of calico+flannel	Community and documentation
More to come..					

Thank You

Help me to better help you next time.
Questions/Feedback:

🐦 **@arun_sriraman**