Container Networking

Baohua Yang

2016-06-27

Outline

- Introduction
- Access Mechanisms
- Carrier Network Mechanisms
- Potential Solutions
- Conclusion

Introduction

- Two fundamental questions (since VMs)
  - How to access/bind to the under carrier networks
  - How to provide the carrier networks?

- More requirements with Container Cloud
  - Scalability
  - Isolation
  - Dynamics
  - Virtual address (IP, Port)
  - Service Discovery
  - QoS
  - Integration
  - And …

Today, we try to answer the 2 basic questions.

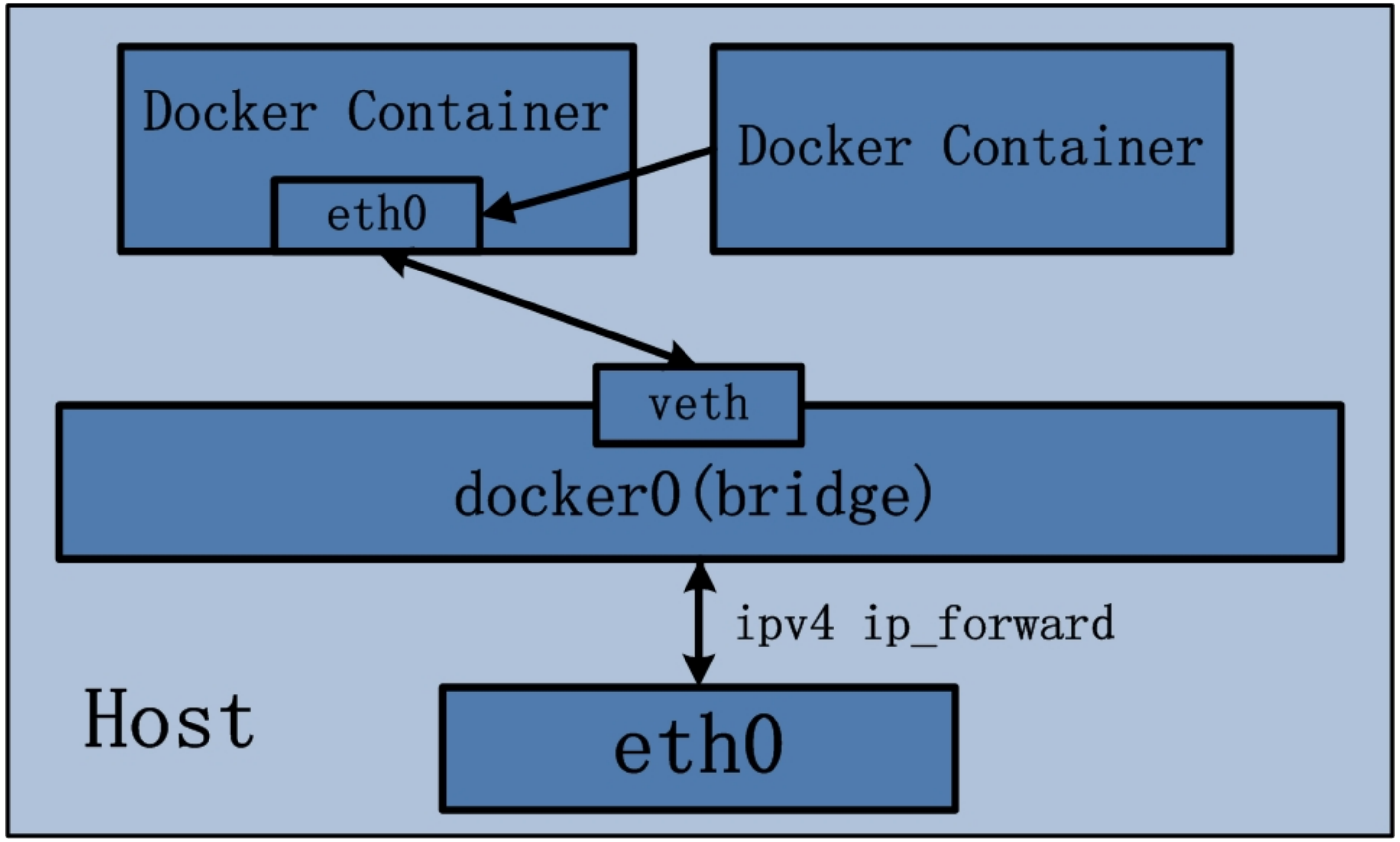
1

How to access the network?

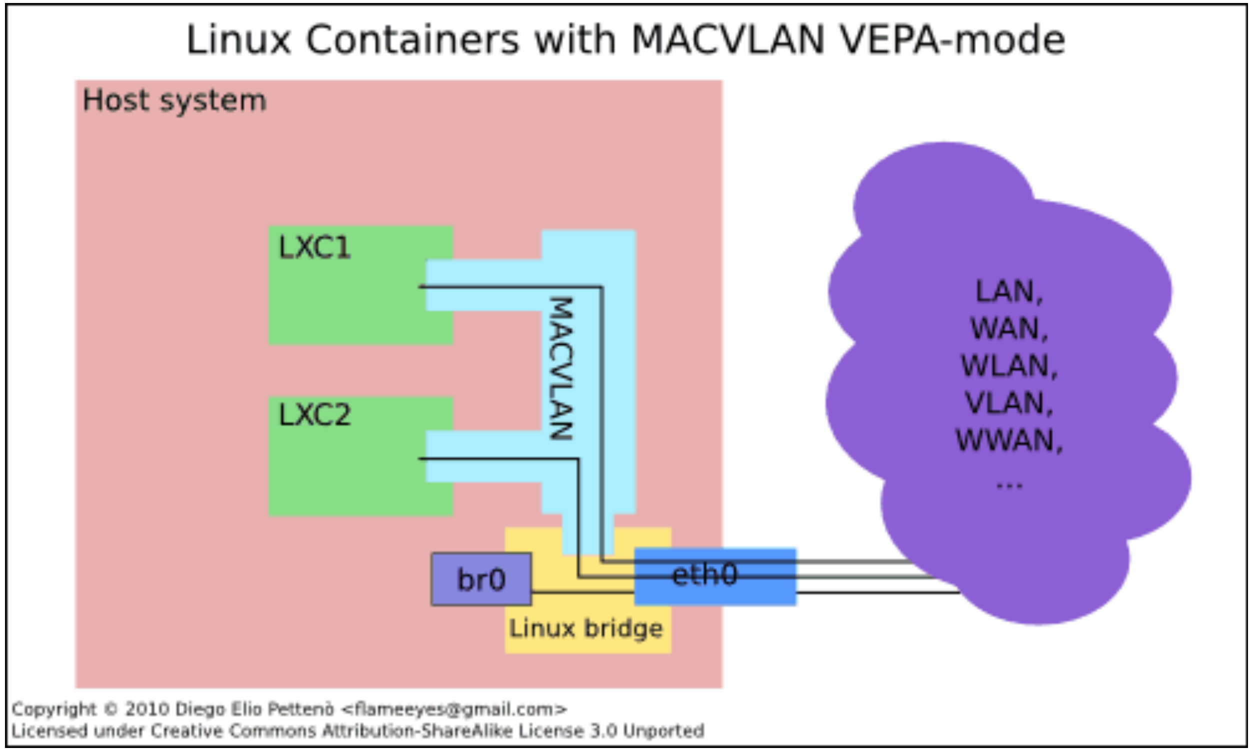
Access Mechanisms

- Software Based
  - Virtual ports + Linux Bridges/ovs
- Hardware Based
  - Mapping virtual Nic
- Libnetwork
  - Supported since *1.7.0* (2015-06-16)
  - Docker official API to create network and bind container

Software Based

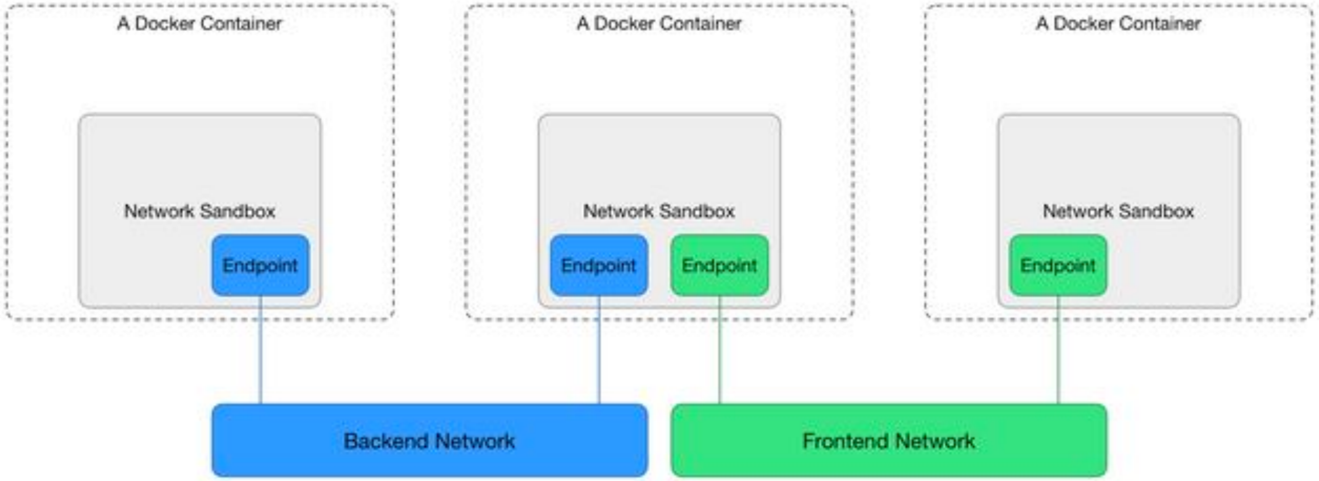


Hardware Based



Libnetwork

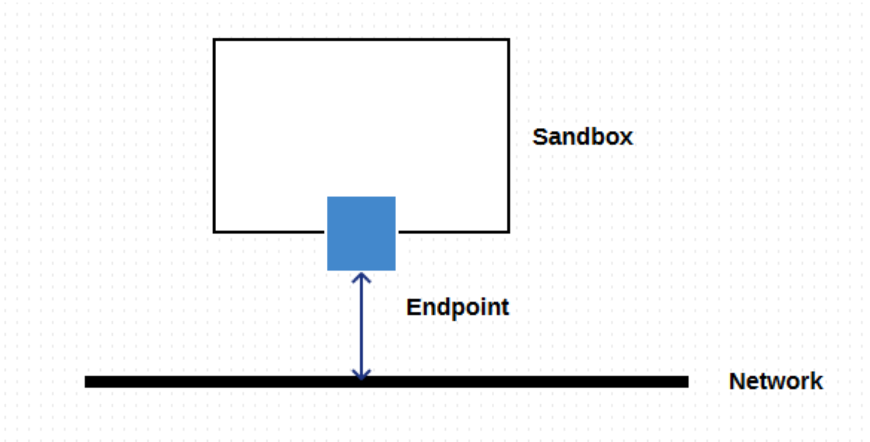
- Experimentally support cross-host networking
- Based on socketplane
- Separated networking module from the core Docker engine
- New pluggable architecture
- Implements the Container Network Model (CNM)
- New network API/CLI



Libnetwork (cont.)

- Main Concepts:
  - Sandbox: Configuration of a container’ s network stack
  - Network: Endpoints that can communicate with Each other
  - Endpoint: Connects sandbox to networks
  - NetworkController: controller, exposes REST API to Docker Engine
  - Driver: handler real works (IPAM) for networkcontroller

- SDN style
  - Similar to Neutron Core concepts



Libnetwork (cont.)

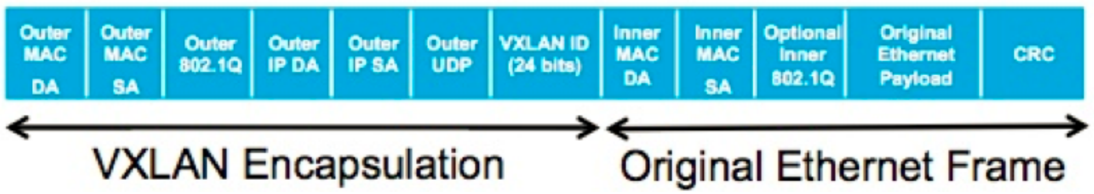
- Supported drivers:
  - Built-in
    - Null: no network
    - Host: user host network stack
    - Bridge: traditional Docker networking (new implementation)
    - Overlay: multi-host networking
  - Remote: Connecting to Docker Network plugins
    - Uses JSON-RPC
    - Can utilize 3rd party networking solution

2

How to design the network?

Carrier Network Mechanisms

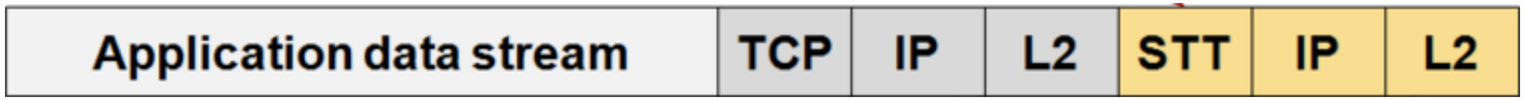
- NAT
- L2 switching
- L3 routing
- Overlay
  - VLAN
  - GRE
  - VXLAN
  - STT
  - NVGRE
  - GENEVE



Outer MAC DA	Outer MAC SA	Outer 802.1Q	Outer IP DA	Outer IP SA	Outer UDP	VXLAN ID (24 bits)	Inner MAC DA	Inner MAC SA	Optional Inner 802.1Q	Original Ethernet Payload	CRC

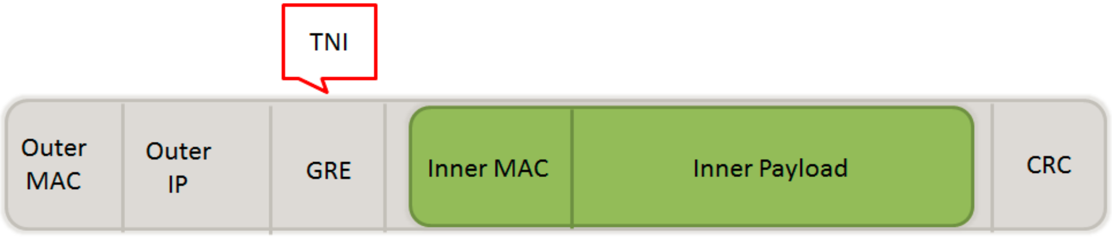
VXLAN Encapsulation

Original Ethernet Frame



Application data stream	TCP	IP	L2	STT	IP	L2

NVGRE Encapsulation



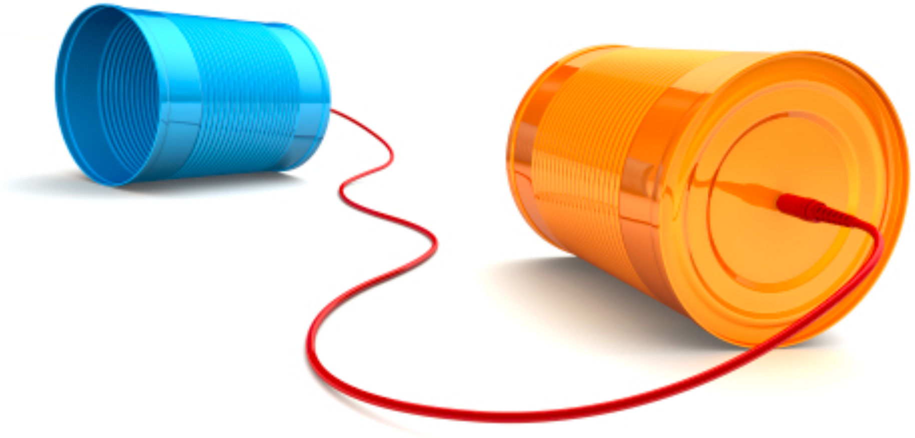
TNI

Outer MAC	Outer IP	GRE	Inner MAC	Inner Payload	CRC

Potential Solutions

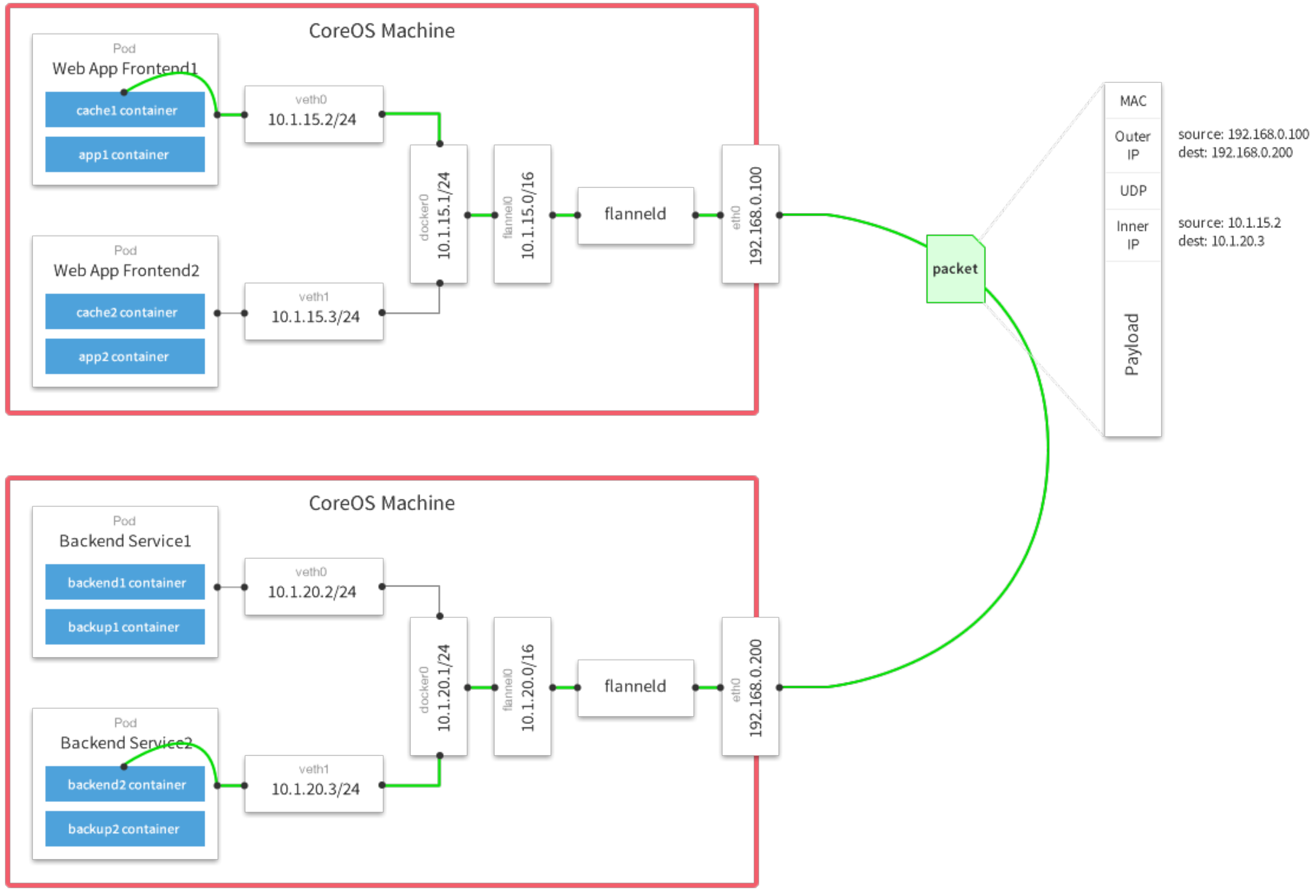
- NAT
- L2/L3 forwarding
- Overlay

- Flannel
- Weave
- Calico
- Neutron + Kuryr



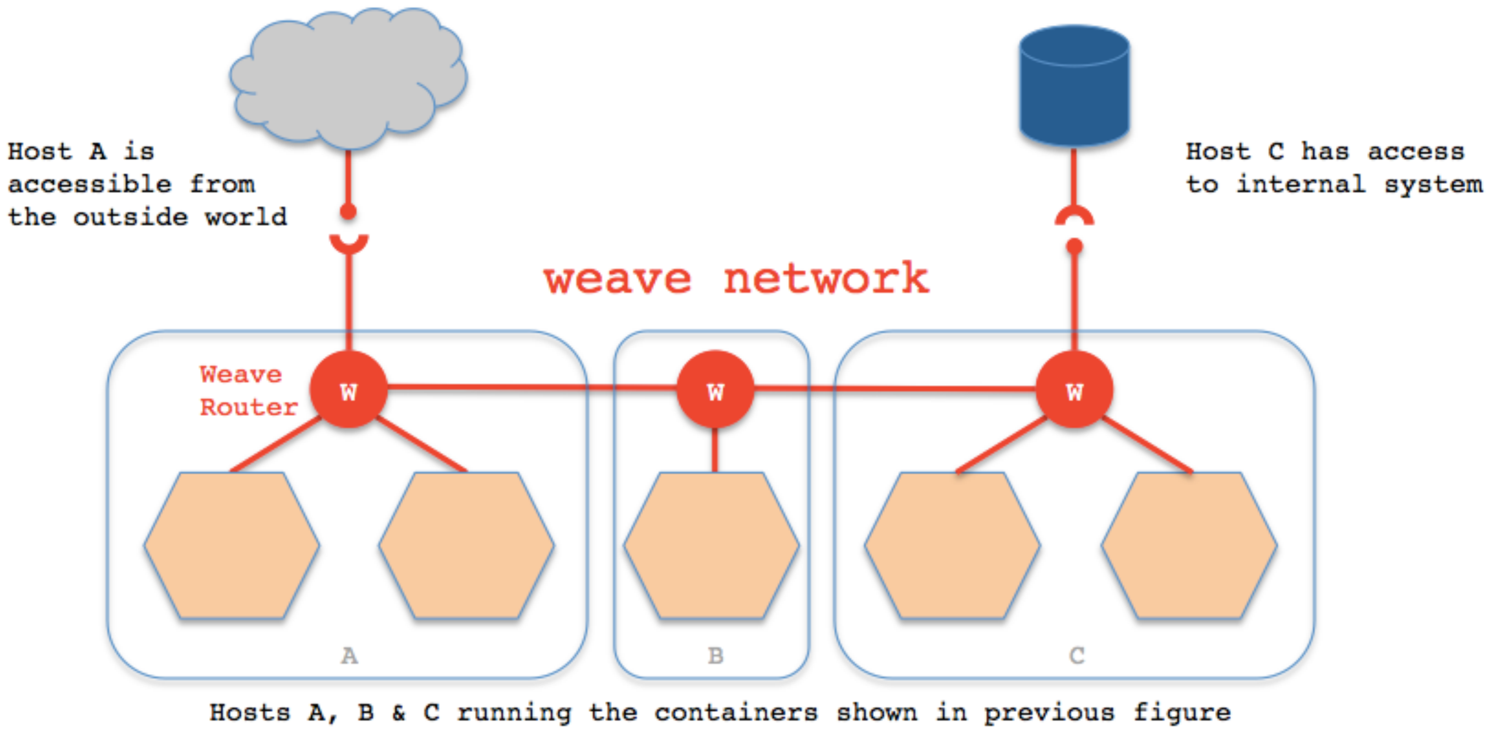
Flannel

- CoreOS
- bridge + UDP Tunnel



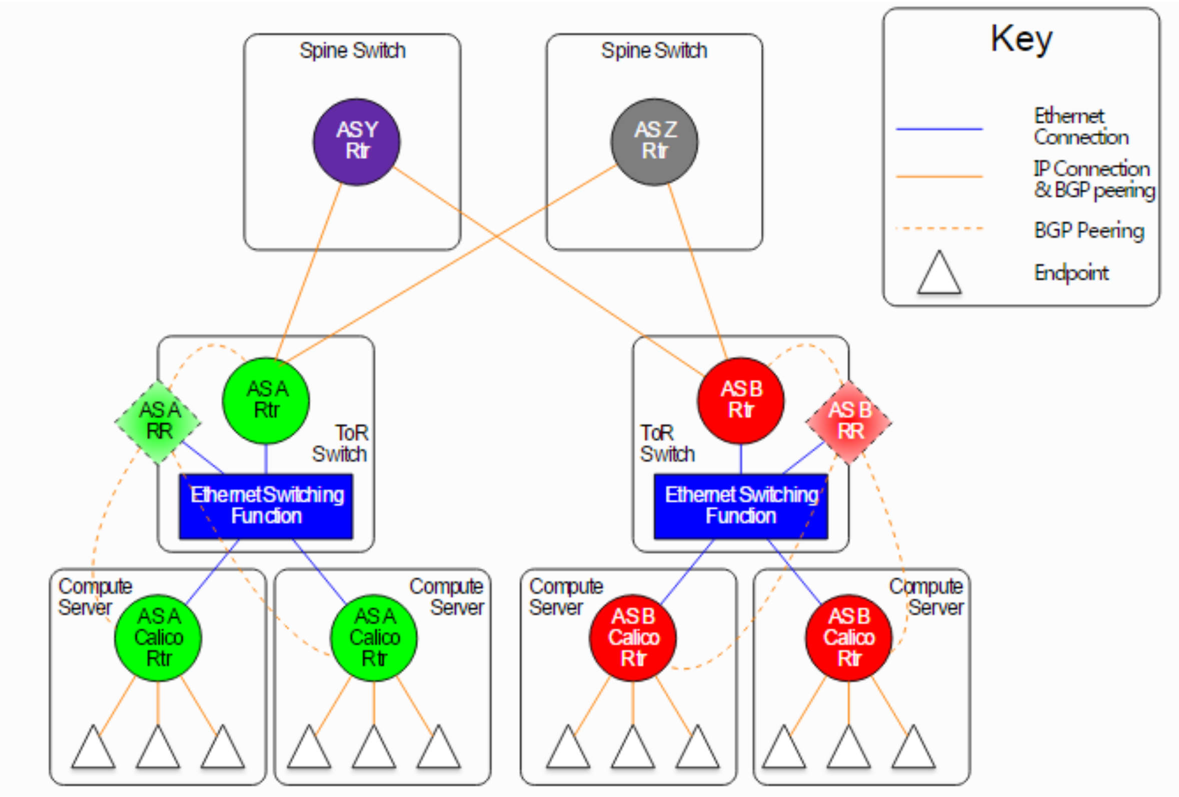
Weave

- Zett.io
- Pcap+UDP Tunnel


Host A is accessible from the outside world
Host C has access to internal system
weave network
Weave Router
W
W
W
A
B
C
Hosts A, B & C running the containers shown in previous figure

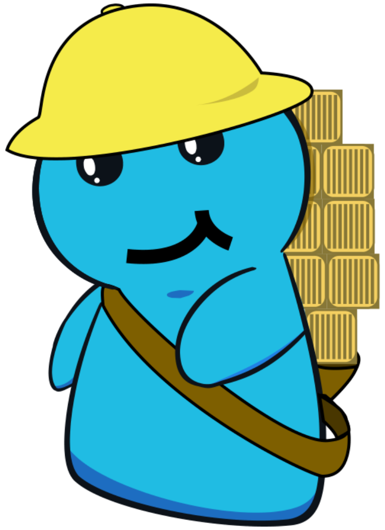
Calico

- Pure Layer 3 approach using vRouter + BGP
- Linux routing + iptables



Kuryr

- Bridge CNM with Openstack Neutron
- Kuryr is a Docker network plugin that uses Neutron
  - Provides networking services to Docker
  - Provides containerized images for the common Neutron plugins
  - Provides volume plugin for Docker (planed)



Kubernetes

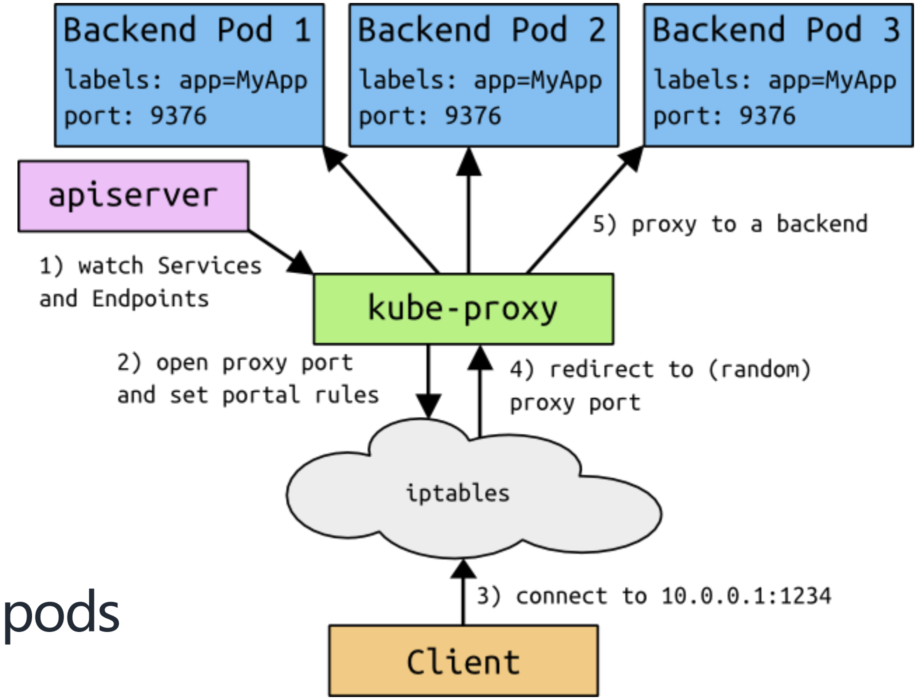
- CNI
  - Rkt networking model
  - Container: a linux network namespace
  - Network: uniquely addressable entities that can communicate with each other
  - Noop, bridge, local-host

Kuryr cont.

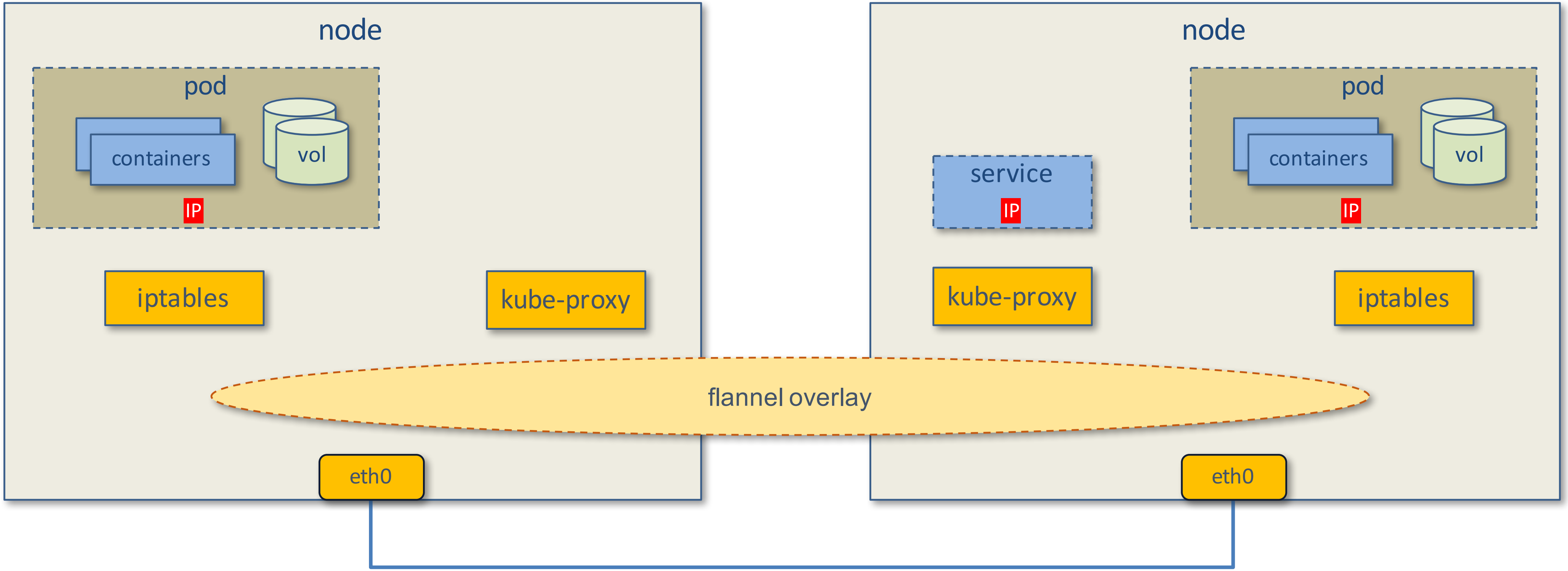
- Docker Networks: Neutron networks
- Docker Endpoints: Neutron ports
  - Neutron subnets gets created from a predefined Neutron subnetpool
  - Docker IPAM driver for Kuryr in the works
- Docker Join/Leave: plug/unplug

Kubernetes

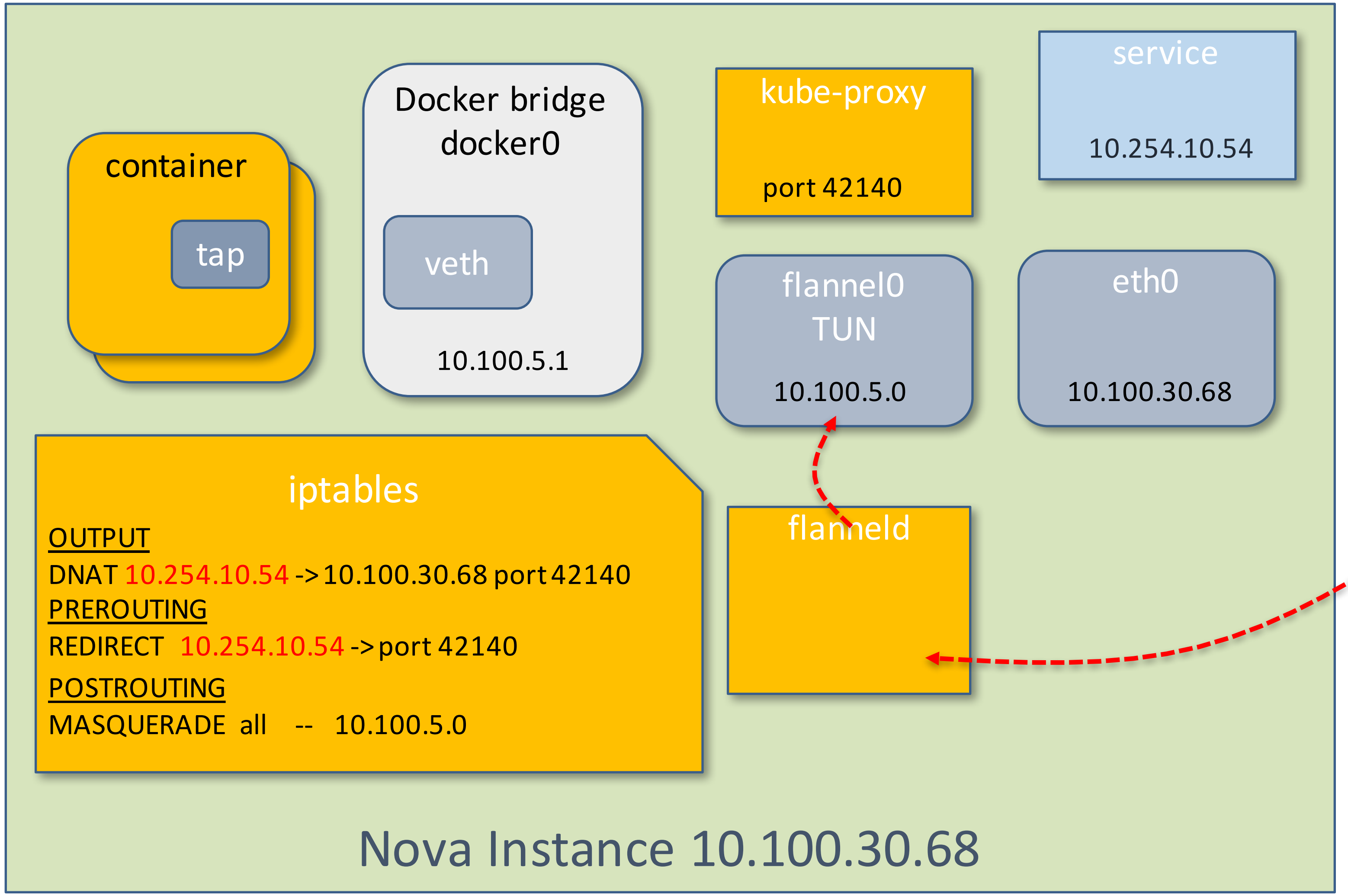
- Abstractions
  - pods:
    - group of containers on same host
    - IP per pod
  - service:
    - proxy, load balancing
    - IP per service
    - replication controller: maintain exact number of pods
- Networking support
  - kube-proxy: a Kubernetes component
  - flannel: an overlay network (other options available)
  - iptables rules: kernel support



Kubernetes Cluster in Operation

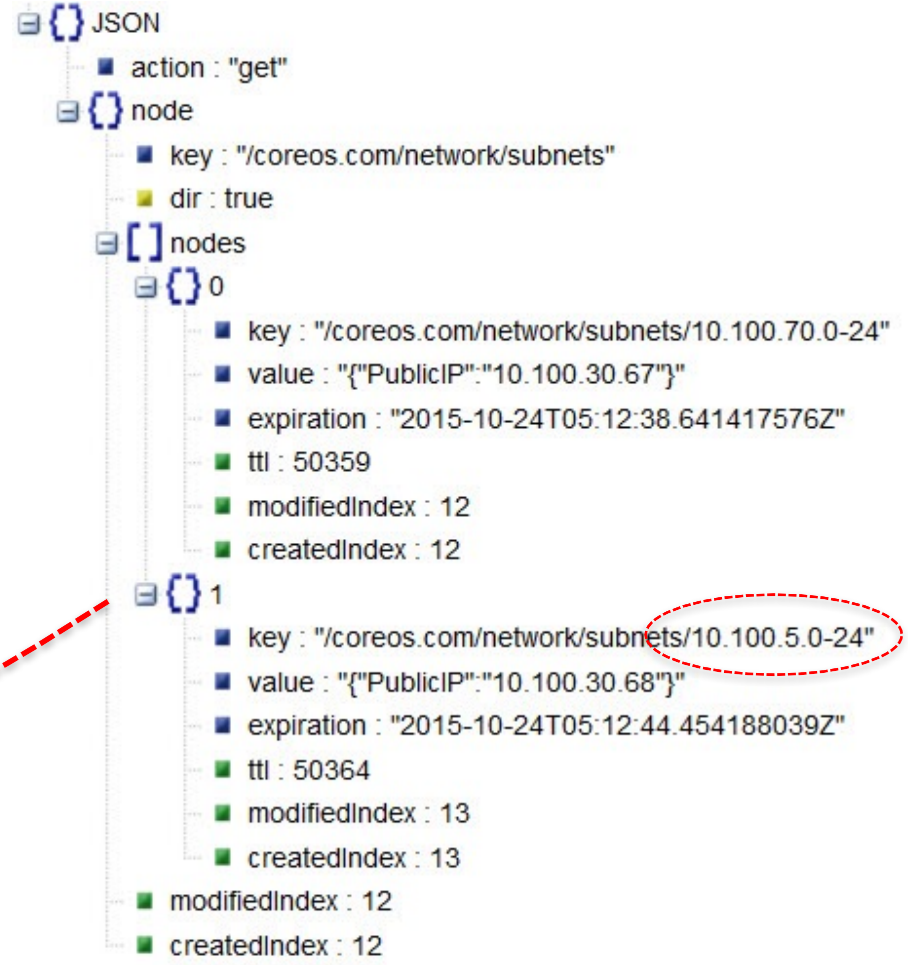


Networking Setup



iptables

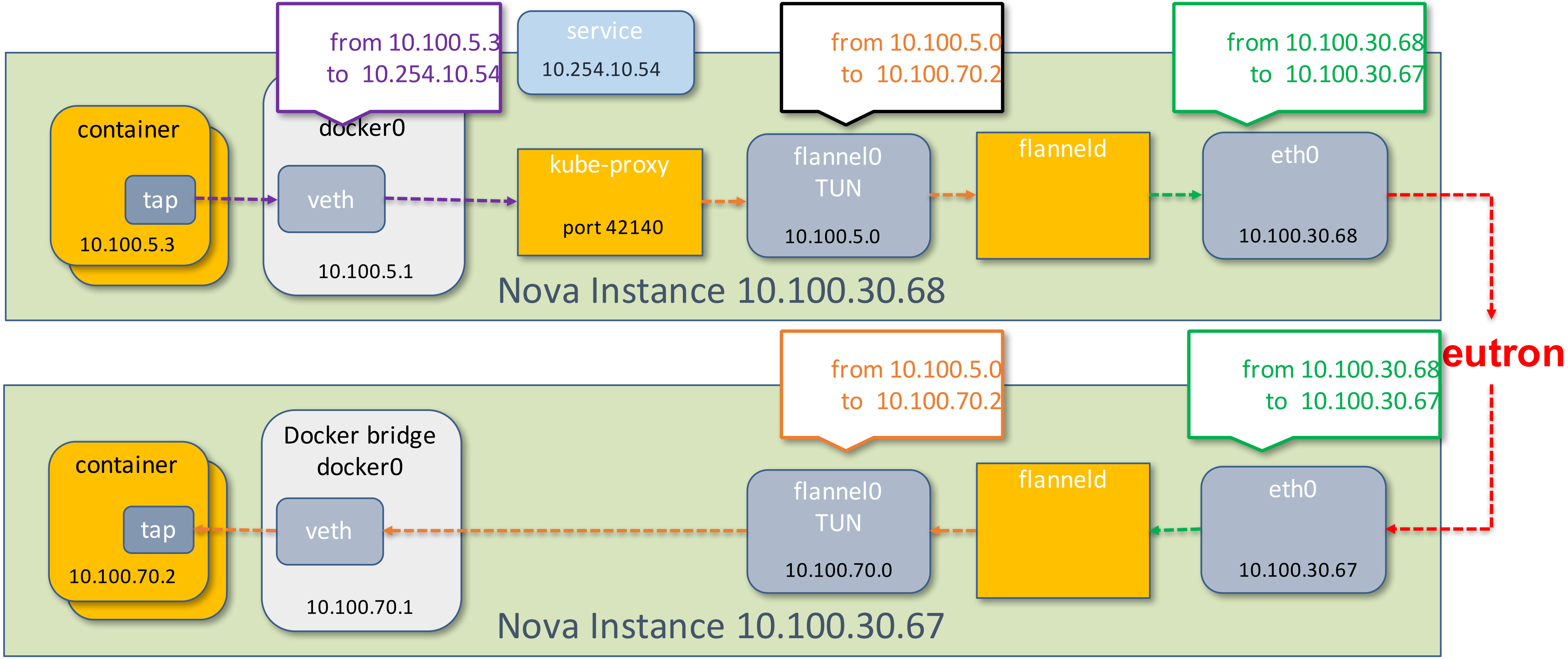
OUTPUT
DNAT 10.254.10.54 ->10.100.30.68 port 42140
PREROUTING
REDIRECT 10.254.10.54 -> port 42140
POSTROUTING
MASQUERADE all -- 10.100.5.0



```
{} JSON
    action : "get"
  {} node
      key : "/coreos.com/network/subnets"
      dir : true
    [] nodes
      {} 0
          key : "/coreos.com/network/subnets/10.100.70.0-24"
          value : "{"PublicIP":"10.100.30.67"}"
          expiration : "2015-10-24T05:12:38.641417576Z"
          ttl : 50359
          modifiedIndex : 12
          createdIndex : 12
      {} 1
          key : "/coreos.com/network/subnets/10.100.5.0-24"
          value : "{"PublicIP":"10.100.30.68"}"
          expiration : "2015-10-24T05:12:44.454188039Z"
          ttl : 50364
          modifiedIndex : 13
          createdIndex : 13
      modifiedIndex : 12
      createdIndex : 12
```

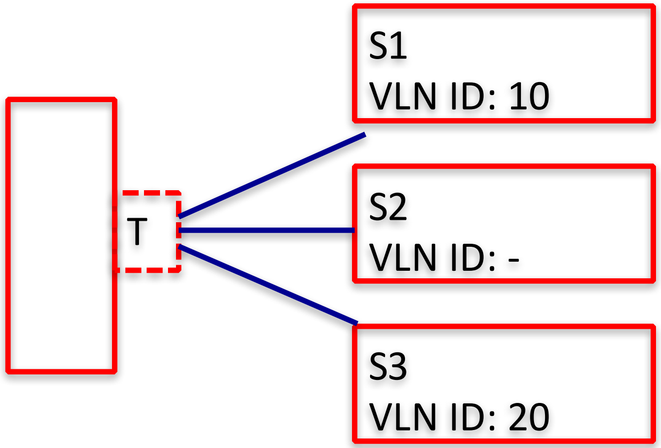
etcd

Container-Container Communication



Neutron: VLAN-Aware VMs

- Provides an efficient way for interconnecting containers deployed within VMs
- Avoids using overlays on top of overlays
- Define new types of Neutron Ports
  - Trunk ports
  - Parent/Children relationship
- Initial patches under review
- Building momentum with increased interest



T: Trunk Port
S: Subport

Optimistic Comparison

	Swam	Kubernetes	Mesos	OpenStack	CoreOS	CloudFoundry
NAT	Y	Y	Y	N	Y	Y
L2/L3 forwading	Y	Y	Y	Y	Y	Y
Flannel	Y	Y	N	N	Y	N
Weave	Y	Y	Y	N	Y	N
Calico	Y	Y	Y	Y	Y	N
Neutron + Kuryr	Y	Y	Y	Y	N	N

* CF is discussing to support overlay.

Conclusion

- SDN is the de-facto standard in Cloud, everyone supports it!
- Libnetwork is key *in Docker ecosystem.*
- Host side networking is becoming more important.
- Many opportunities for new Cloud vendors.

Q&A