CLOUD NATIVE COMPUTING FOUNDATION

Webinar Series

Cloud Native Networking

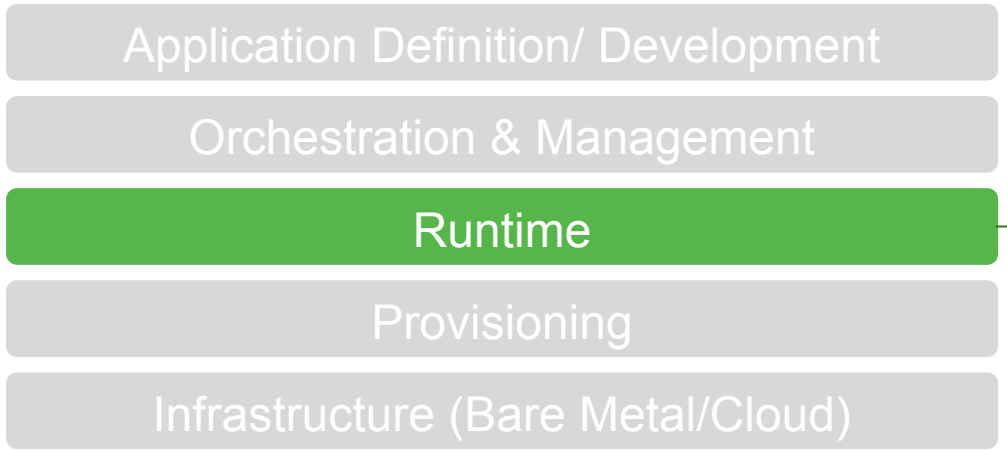
January 12, 2017

Your Presenters

Christopher Liljenstolpe
CTO, Tigera / Founder,
Project Calico

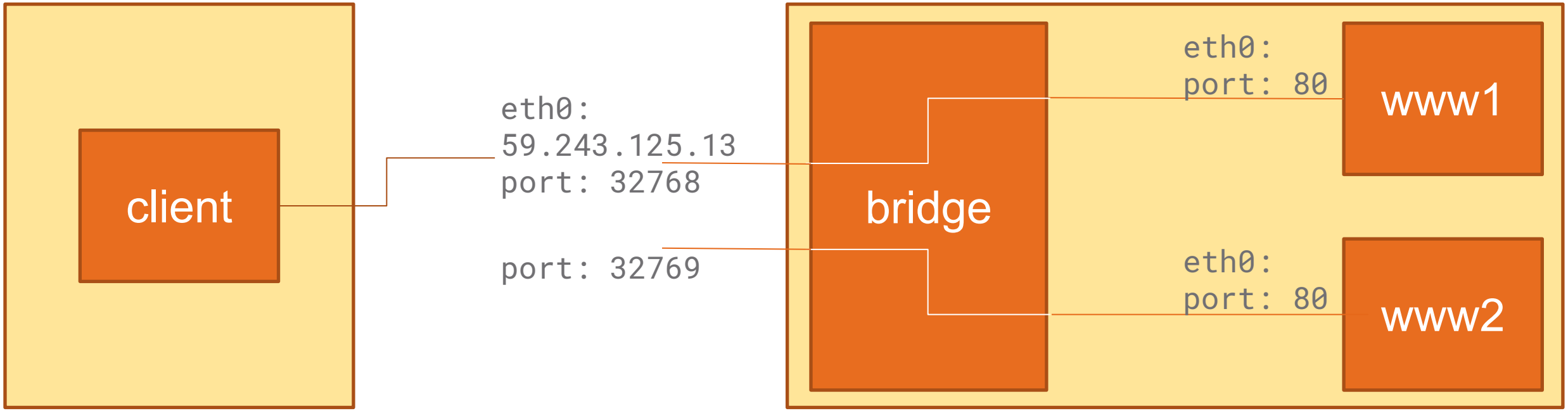
Bryan Boreham
Director of Engineering,
WeaveWorks

Networking in CNCF Reference Architecture



- Resource Management
  - Image Management
  - Container Management
  - Compute Resources
- **Cloud Native – Network**
  - **Network Segmentation and Policy**
  - **SDN & APIs (eg CNI, libnetwork)**
- Cloud Native- Storage
  - Volume Drivers/Plugins
  - Local Storage Management
  - Remote Storage Access

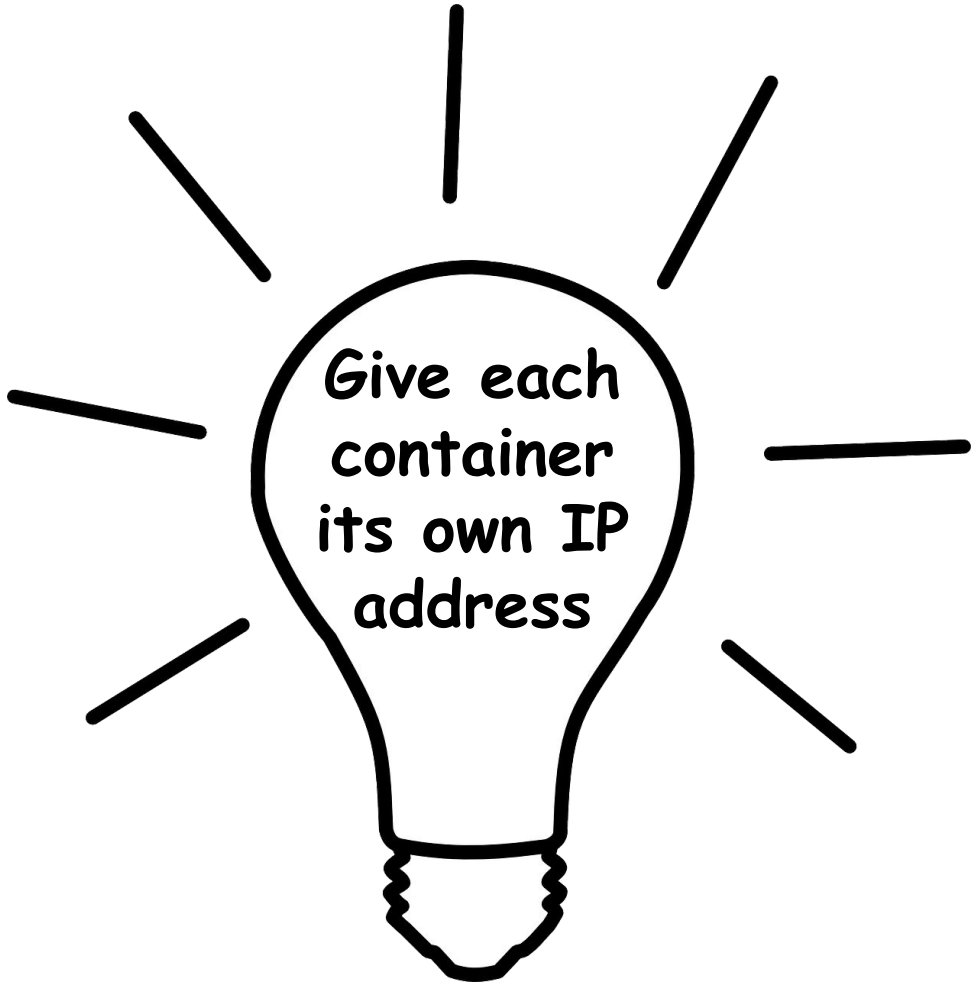
First Iteration of Container Networking: Port Mapping



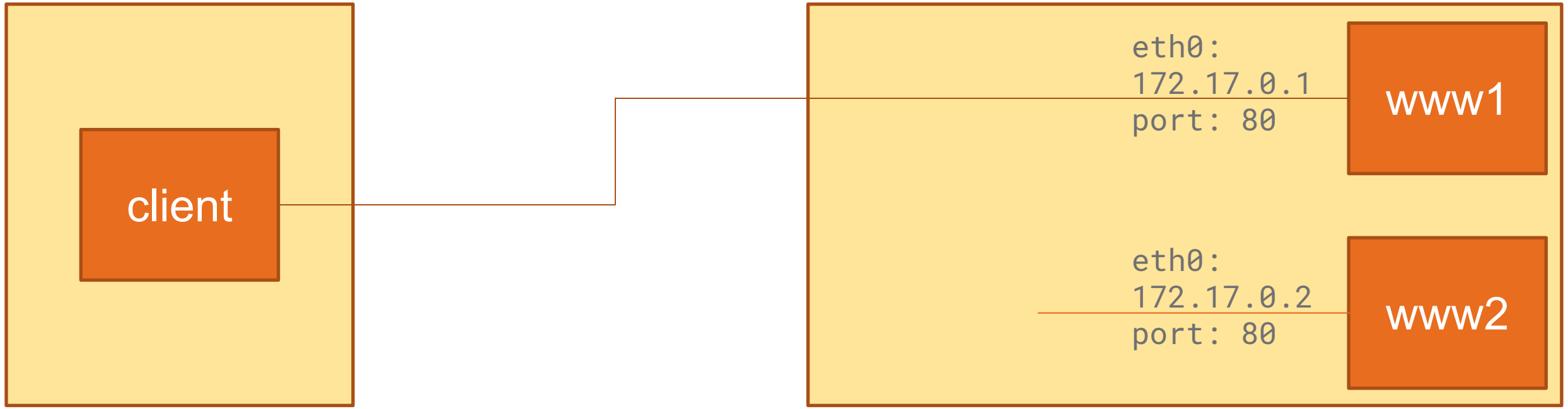
Kinda works… But…

- Port clashes (as above)
- Service discovery (custom code required)

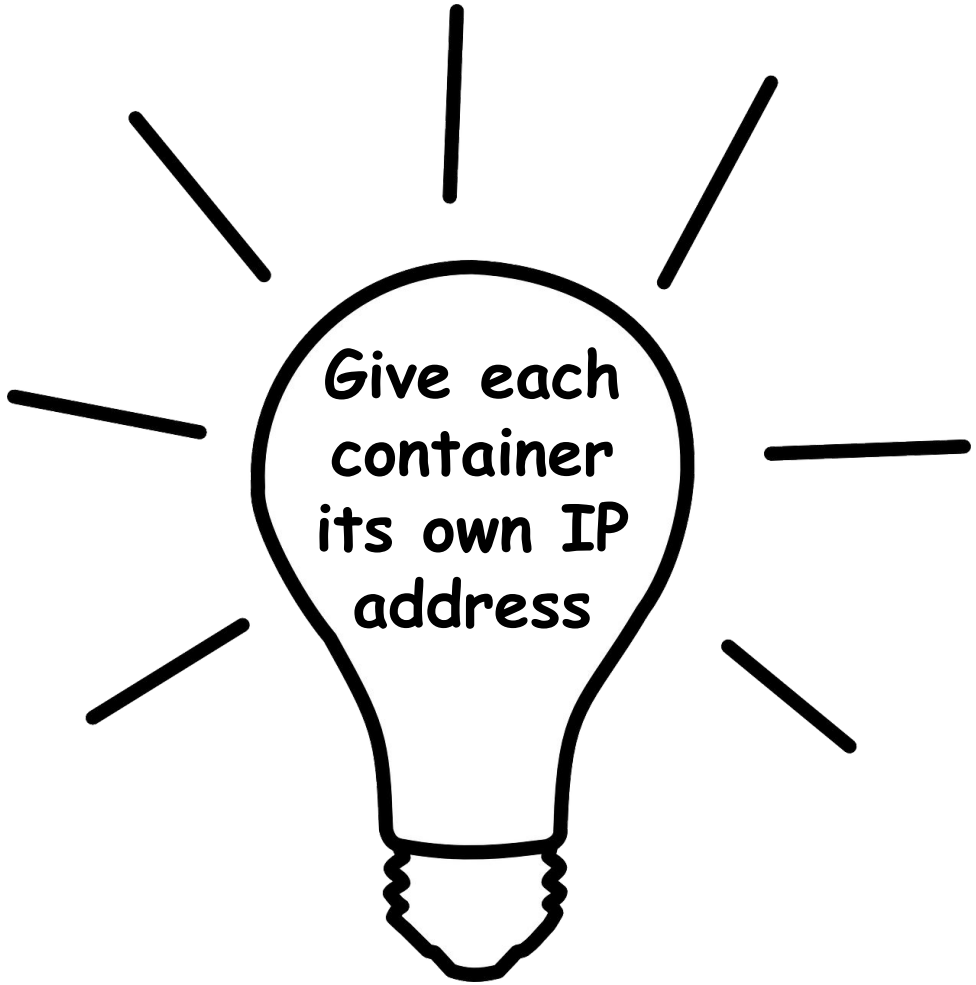
Enter Cloud Native Networks...



Give each container its own IP address



Give each container its own IP address



- ✓ Port clash disappears
- ✓ Workload discovery: as easy as a DNS lookup
- ✓ Kubernetes took this approach from outset
- ✓ We know this works at large scale

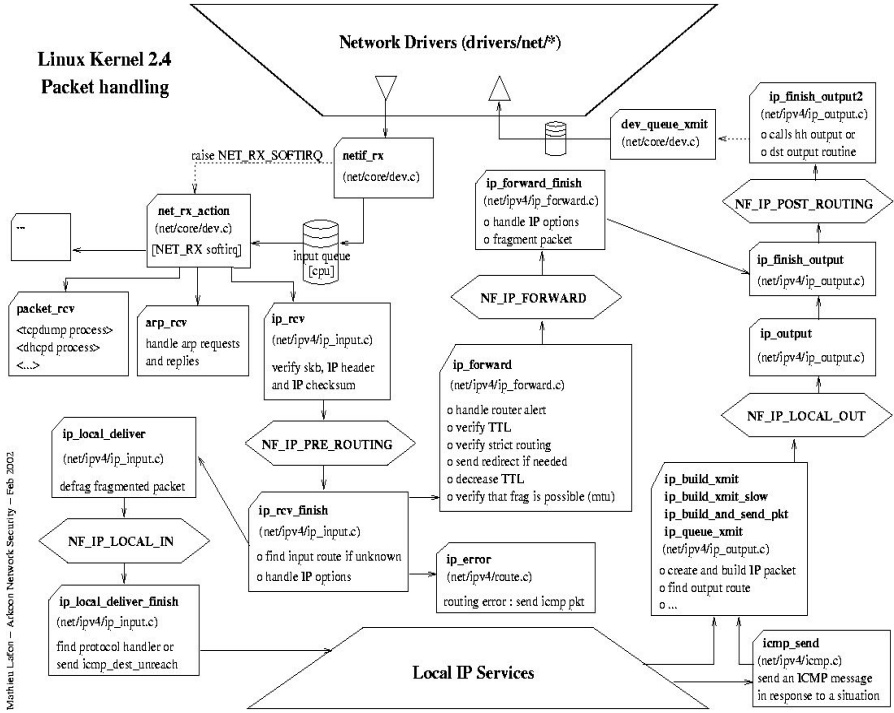
Linux kernel: the ultimate networking toolkit

20M

lines of code

~35%

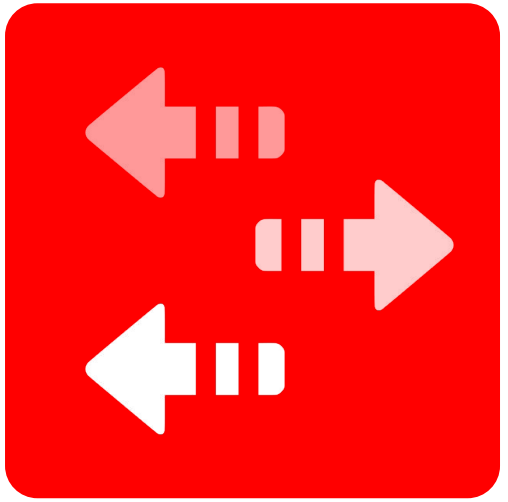
of which is networking



What's in a Cloud Native Network solution?

- assigns IPs (from a pool given to it)
- distributes routing information (i.e. how to get to this workload)
- distributes policy (e.g. who can connect to whom)

Control Plane



Data Plane

for each packet to/ from the workload:

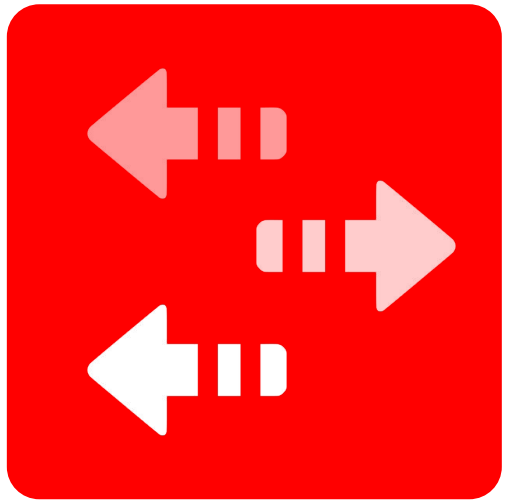
- enforces policy
- forwards it to the right destination

Control plane implementation options

Control Plane

- Distributed key/value store
  - e.g. etcd (used by flannel, Calico)
- Routing protocols
  - e.g. BGP (used by Calico)
- Gossip protocol
  - e.g. Weave Mesh (used by Weave Net)
- Centralized controller
  - e.g. traditional SDNs

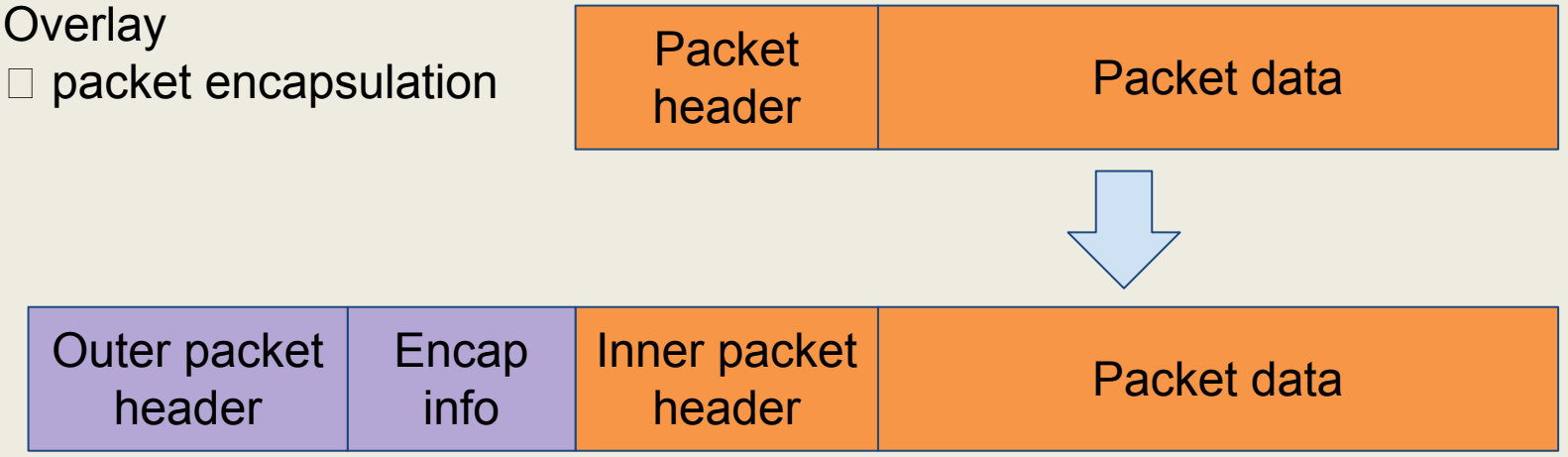
Data plane implementation options



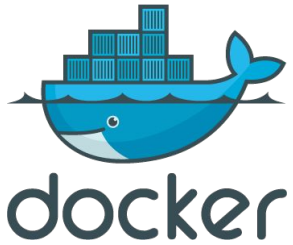
Data Plane

- Forwarding engine:
  - Kernel forwarding or user space
- Transport mechanism
  - overlay or natively using the underlying network

Overlay
- packet encapsulation



Plug-in Models



Container Network Model (CNM / libnetwork)

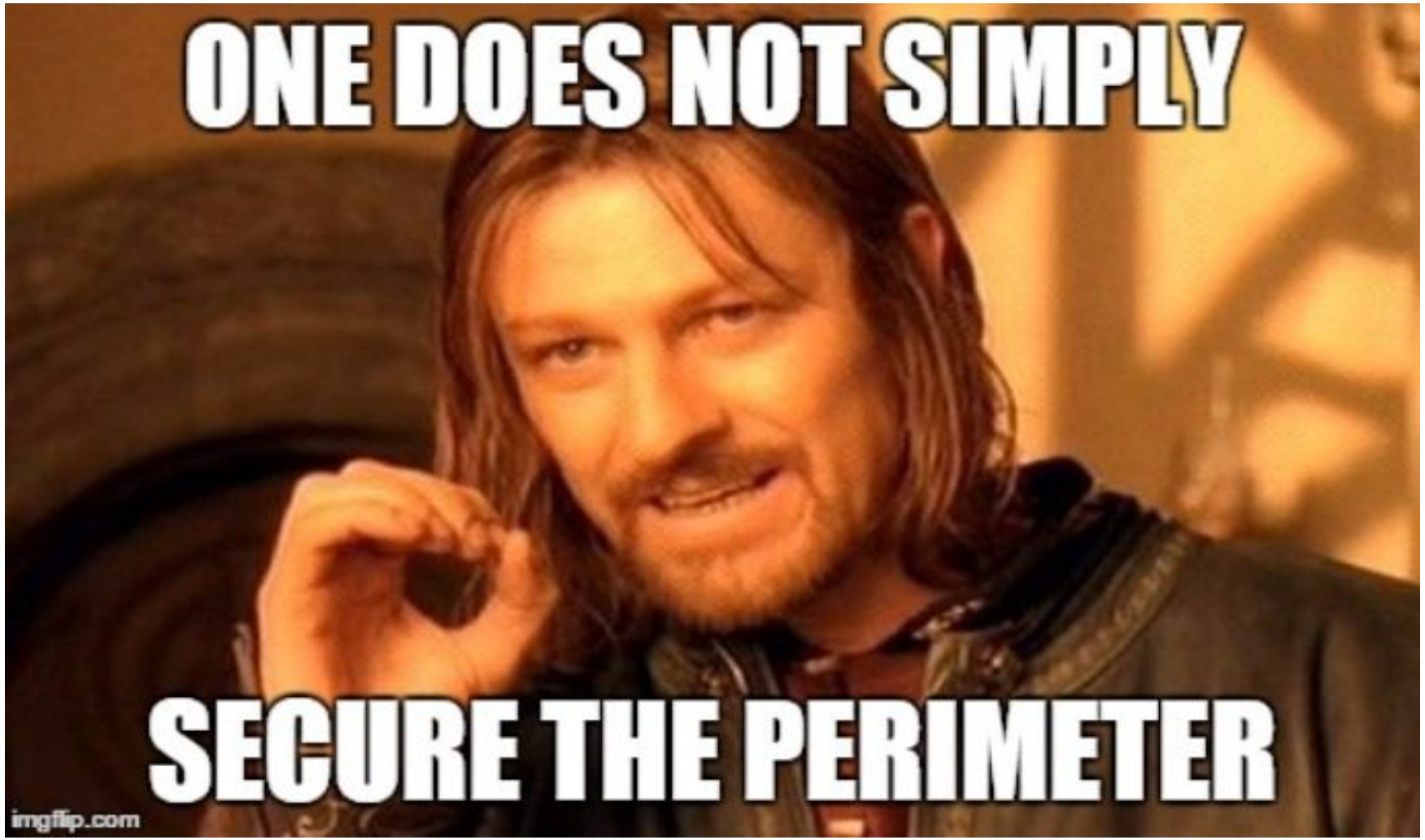
Container Network Interface (CNI)

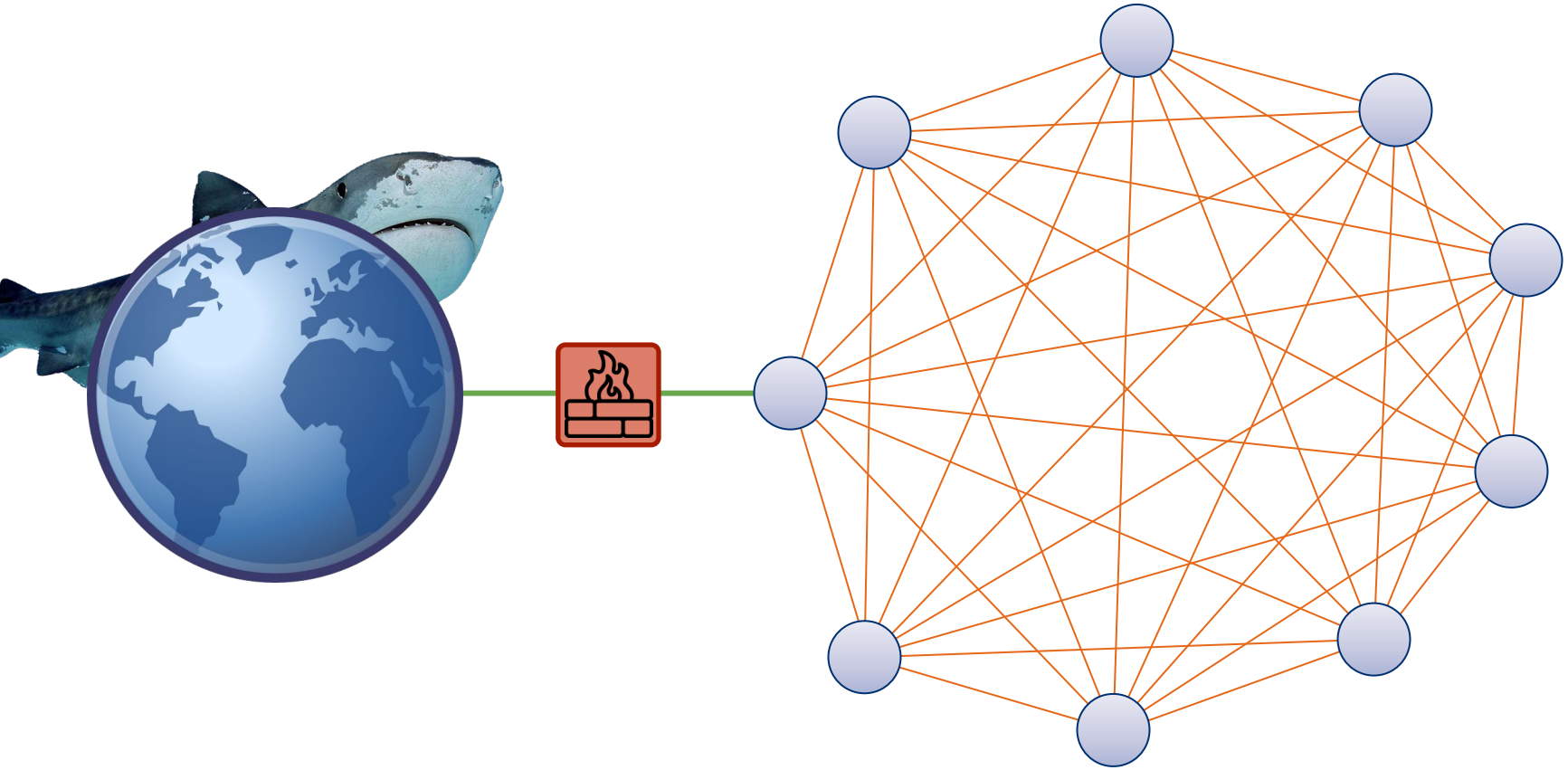
Selecting the right network plug-in

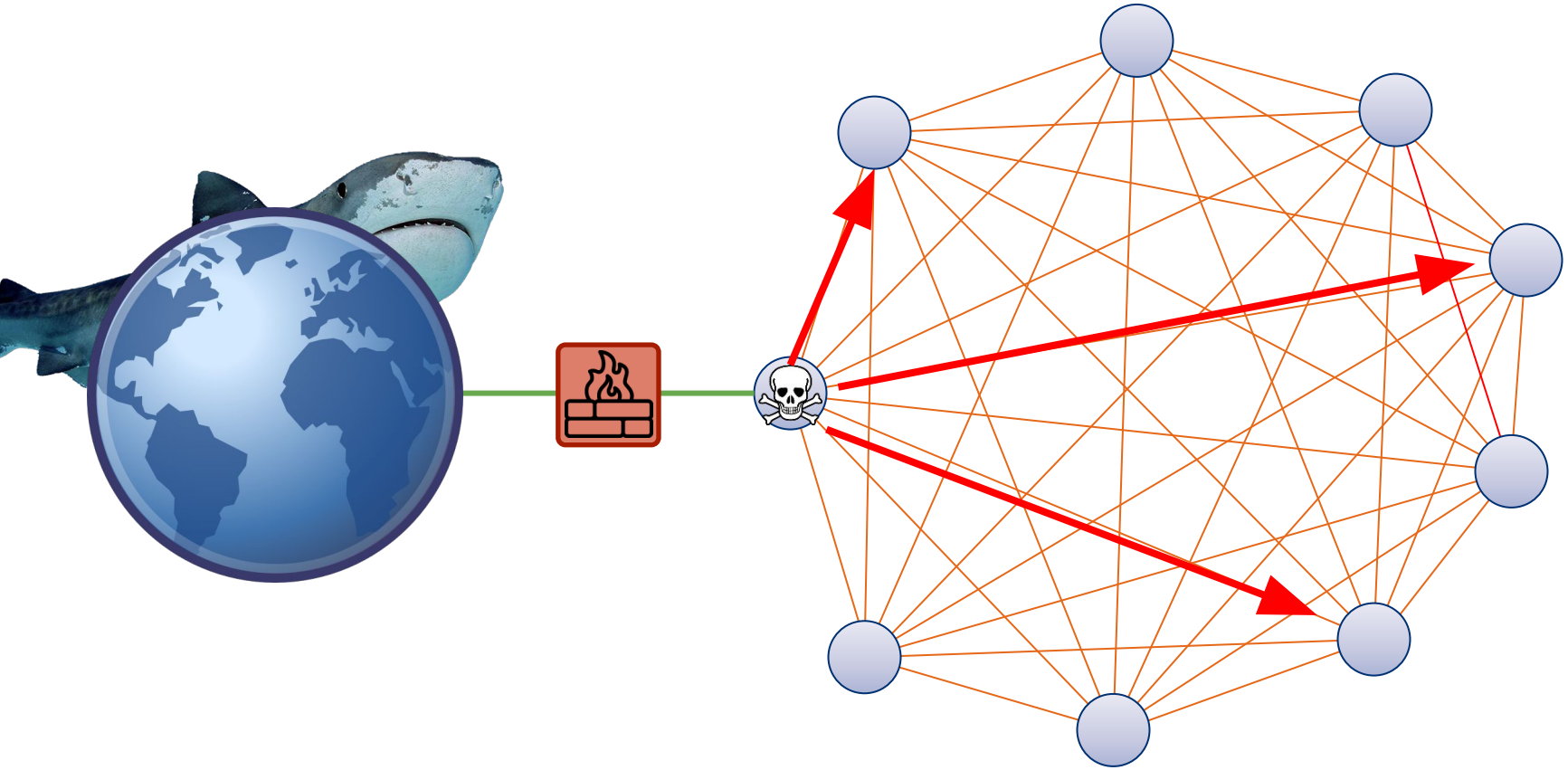
- **Features:**
  - Do I need specific network features such as multicast or encryption?
- **Flexibility:**
  - Does it have to work in my own datacenter; on my laptop; in the cloud; across combinations of these?
  - In the cloud do I need my container network to cross zones or regions?
  - Are there limits on how many hosts I can connect?
- **Ease of configuration**
  - What do I have to install before the container network?
  - What do I have to configure before it will work?
- **Resilience**
  - What are the solution's failure modes / reliability profile?
  - What events is it resilient to? (loss of one node, link, data center, ...)

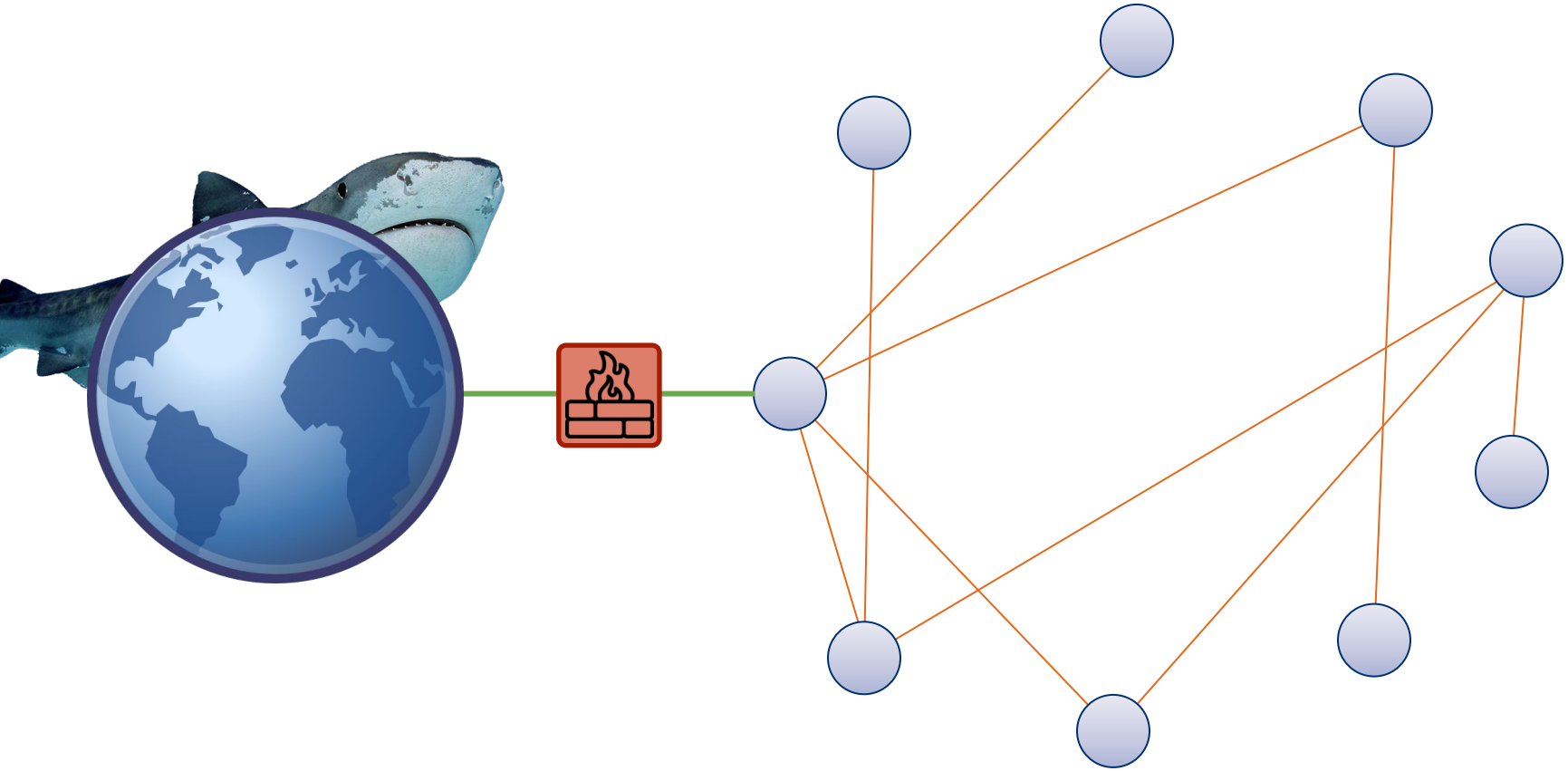
- **Monitoring and Troubleshooting**
  - What tools do I need to monitor the network?
  - What expertise do I need to troubleshoot?
- **Security - does the container network give me protection against:**
  - Snooping
  - Unwanted communication between services
- **Scale and Performance:**
  - What is the necessary 'convergence' time?
  - What are the performance requirements of my application?
  - What are the solution's scaling characteristics? Does it "scale out" as my cluster grows, or depend on a centralized controller that must "scale up"?

Securing the Network with Policy

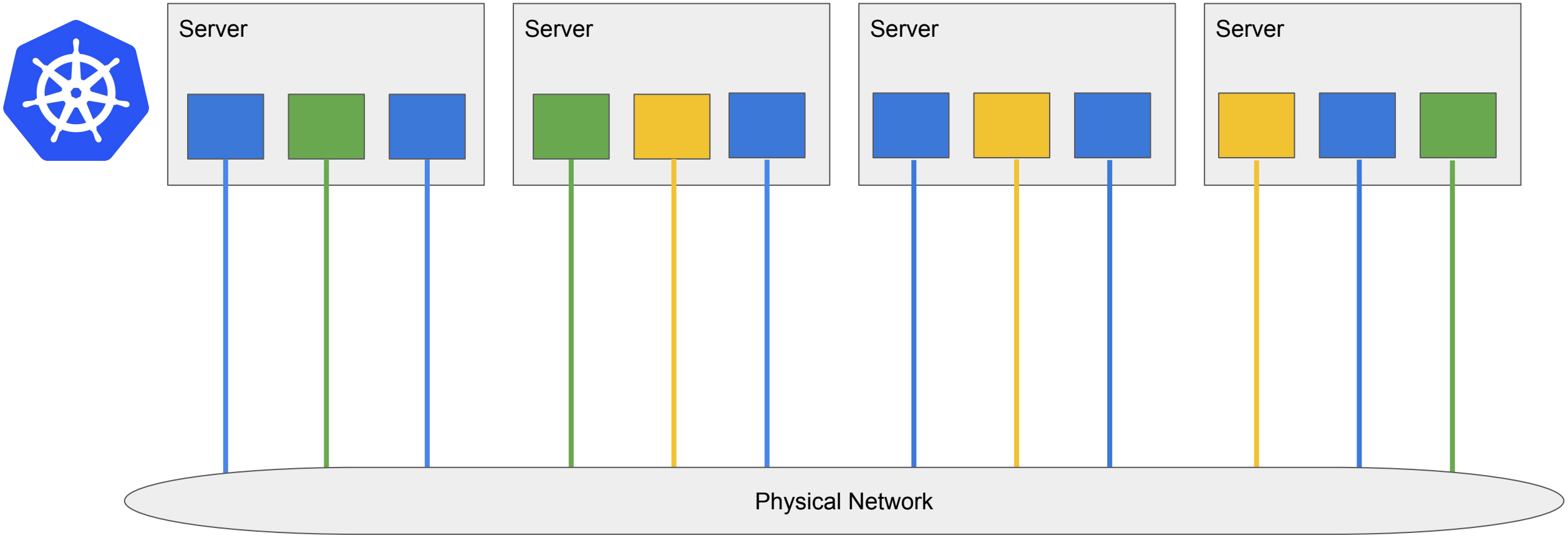








Using policy to separate application tiers



Using policy to separate application tiers

Frontend Tier Policy

```
kind: NetworkPolicy
metadata:
  name: frontend-policy
spec:
  podSelector:
    tier: frontend
  ingress:
  - ports:
    - protocol: tcp
      port: 80
```

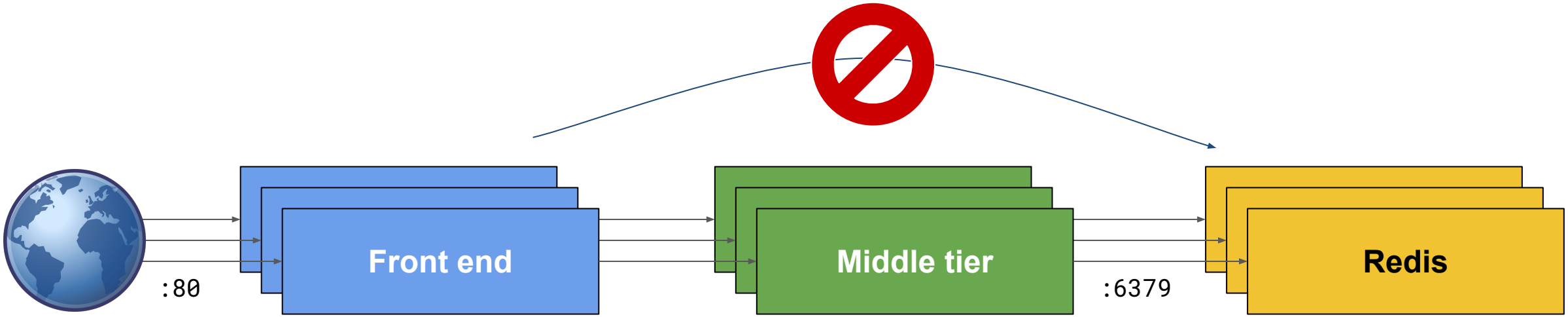
Middle Tier Policy

```
kind: NetworkPolicy
metadata:
  name: middle-tier-policy
spec:
  podSelector:
    tier: middle
  ingress:
  - from:
    - podSelector:
        matchLabels:
          tier: frontend
```

Database Tier Policy

```
kind: NetworkPolicy
metadata:
  name: database-policy
spec:
  podSelector:
    tier: database
  ingress:
  - from:
    - podSelector:
        matchLabels:
          tier: middle
    ports:
    - protocol: tcp
      port: 6379
```

Enforced container topology



Summary

Networking is a key element of Cloud Native computing

IP-per-container is now established best practice, simplest for developers & operations

Multiple ways to implement – decide what is right for your application deployment environment

Thank You