Kubernetes Network Models

Tim Hockin, Google
@thockin

Sept. 10, 2020

What does “network model” mean?

Kubernetes clusters are made up of nodes

- Machines - virtual or physical

Those nodes exist on some network

Pods run on those nodes

Pods get IP addresses

“Network model” describes how those pod IPs integrate with the larger network

Wait, what?

Kubernetes networking in 2 bullets

1) Pods on a node can communicate with all pods on all nodes without NAT

2) Agents on a node (e.g. system daemons, kubelet) can communicate with all pods on that node

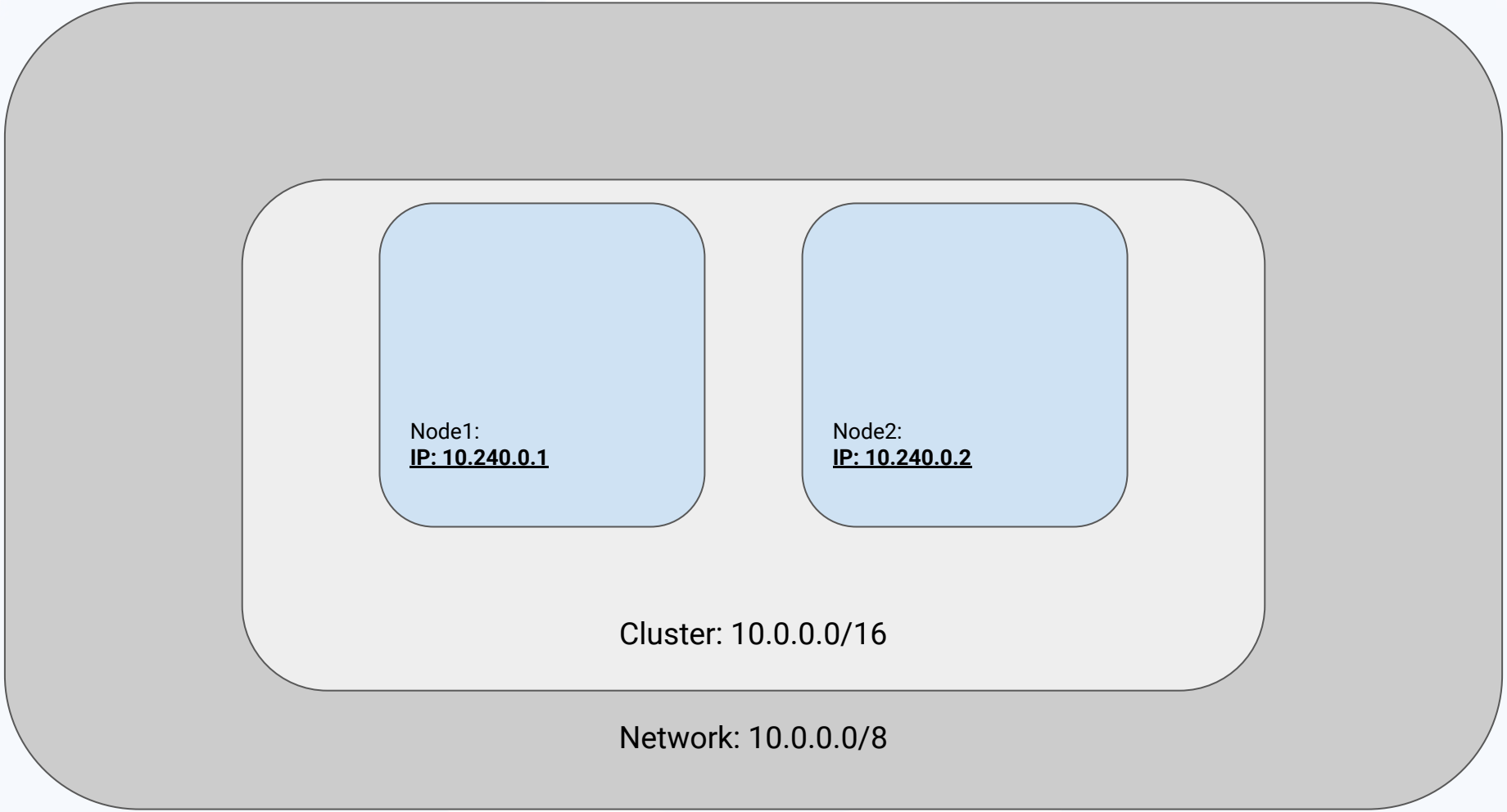
Let’s start with a “normal” cluster

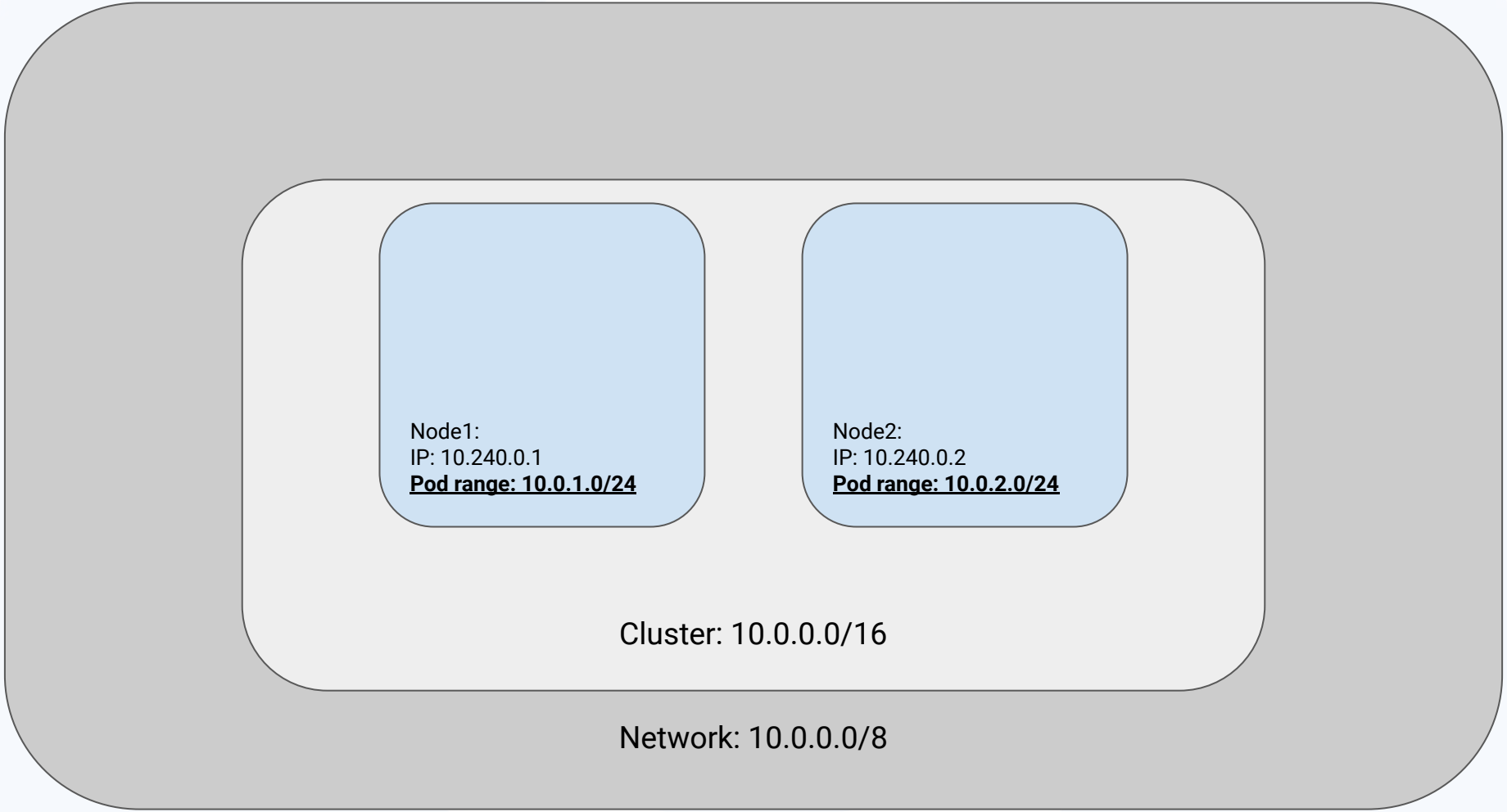
Network: 10.0.0.0/8


Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

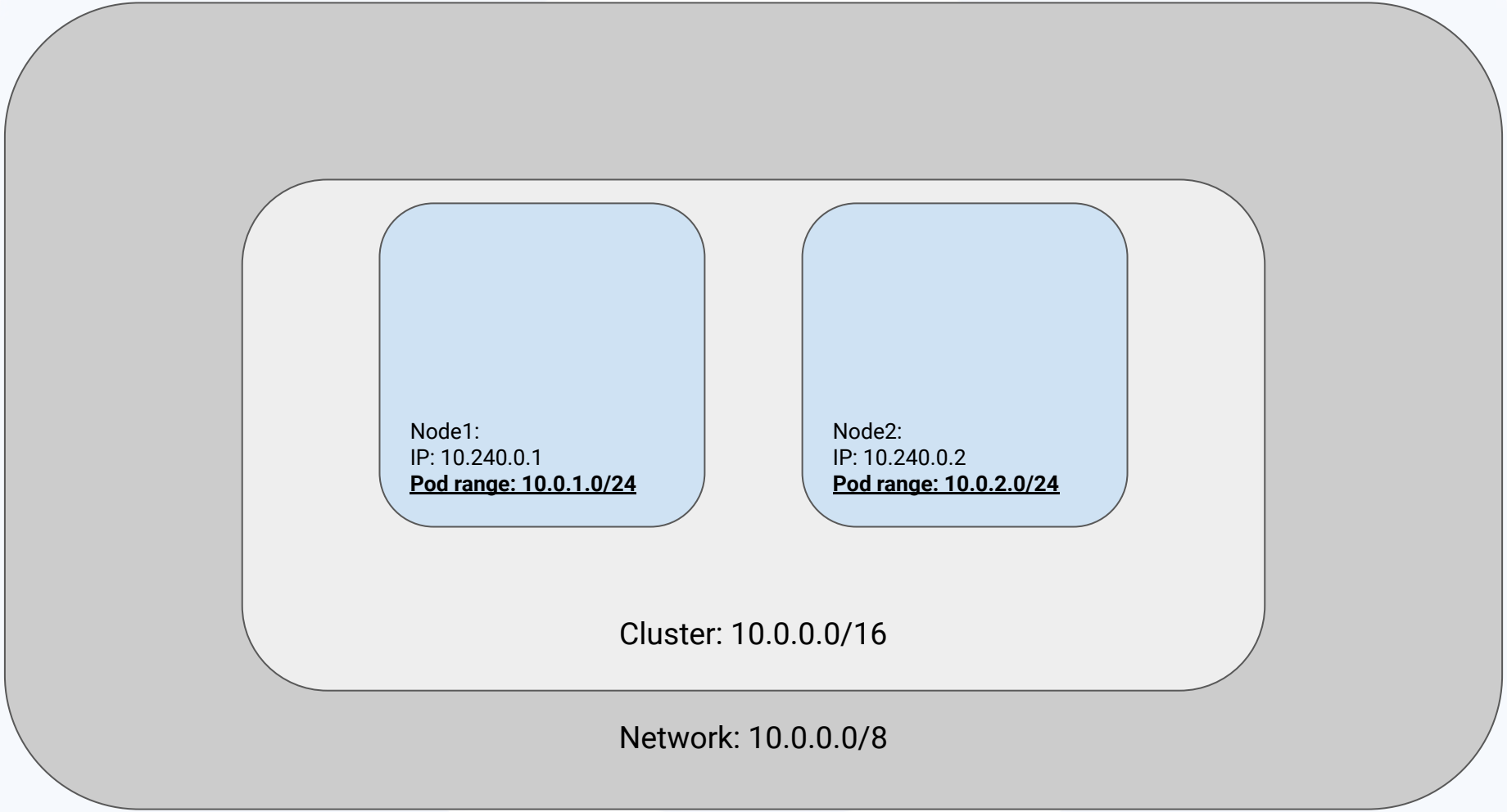
NOTE: It’s not **required** that a cluster be a single IP range, but it’s very common and makes the pictures easier

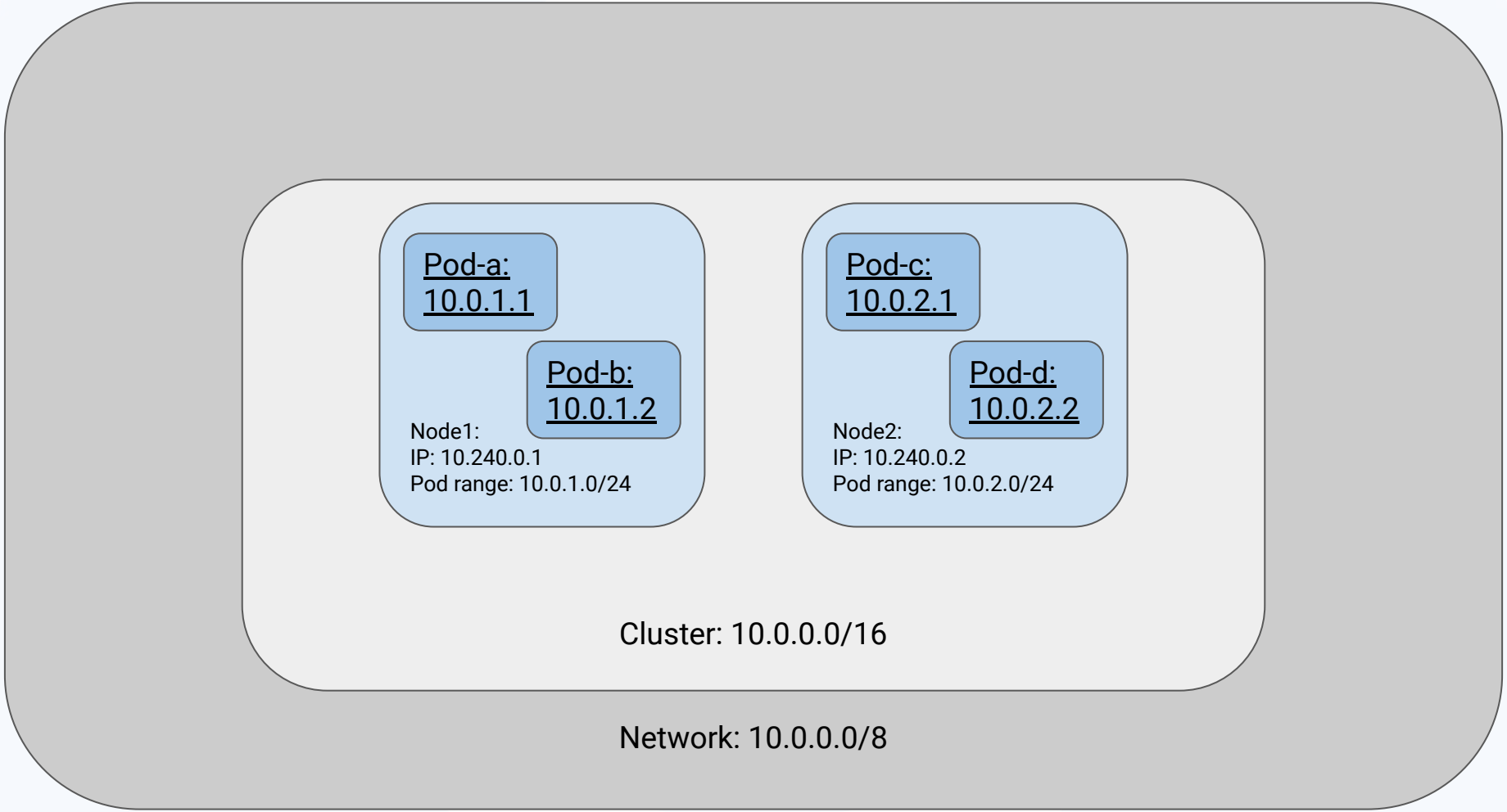

Cluster: 10.0.0.0/16
Network: 10.0.0.0/8


Node1:
IP: 10.240.0.1
Node2:
IP: 10.240.0.2
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

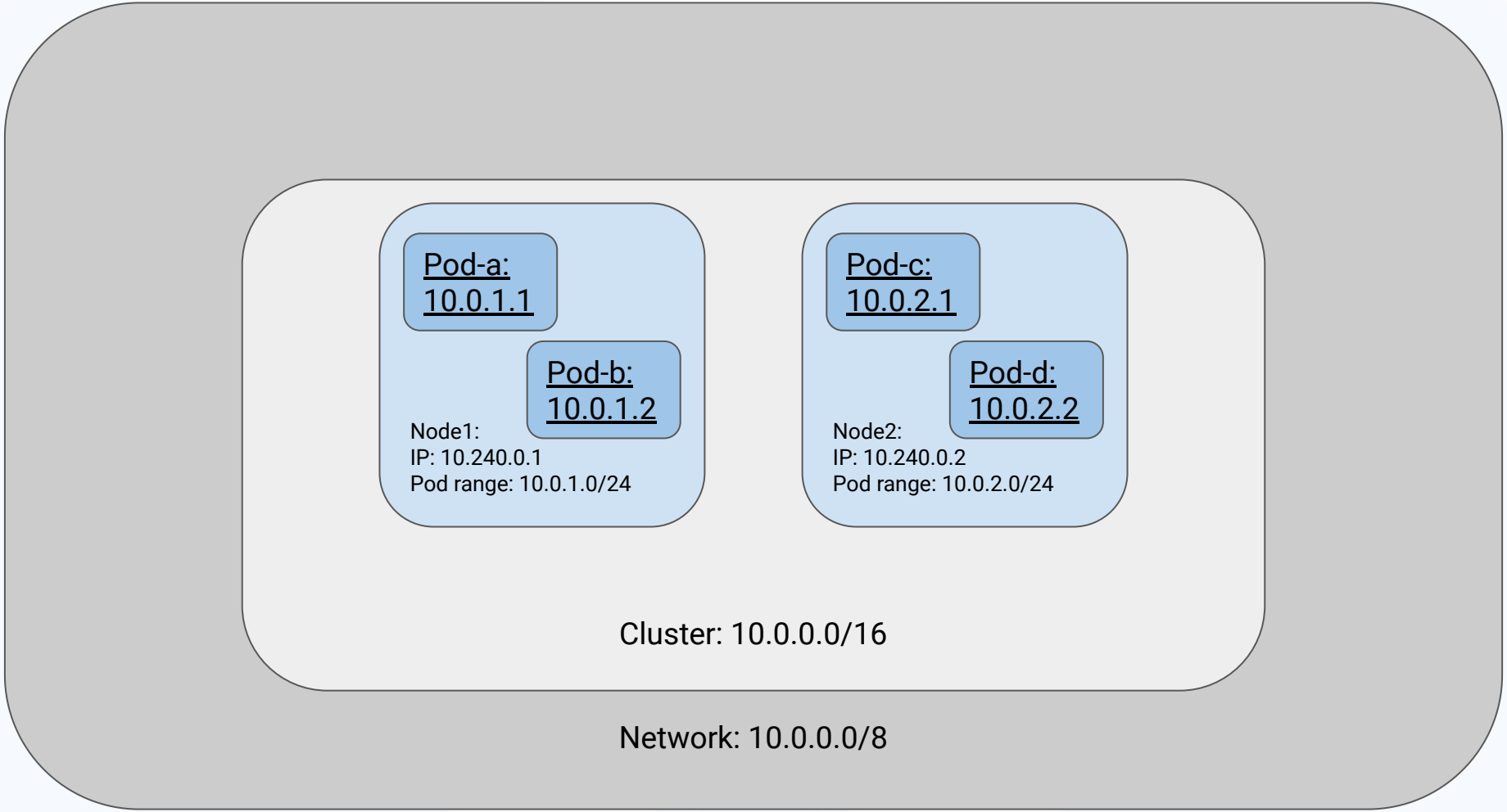

Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

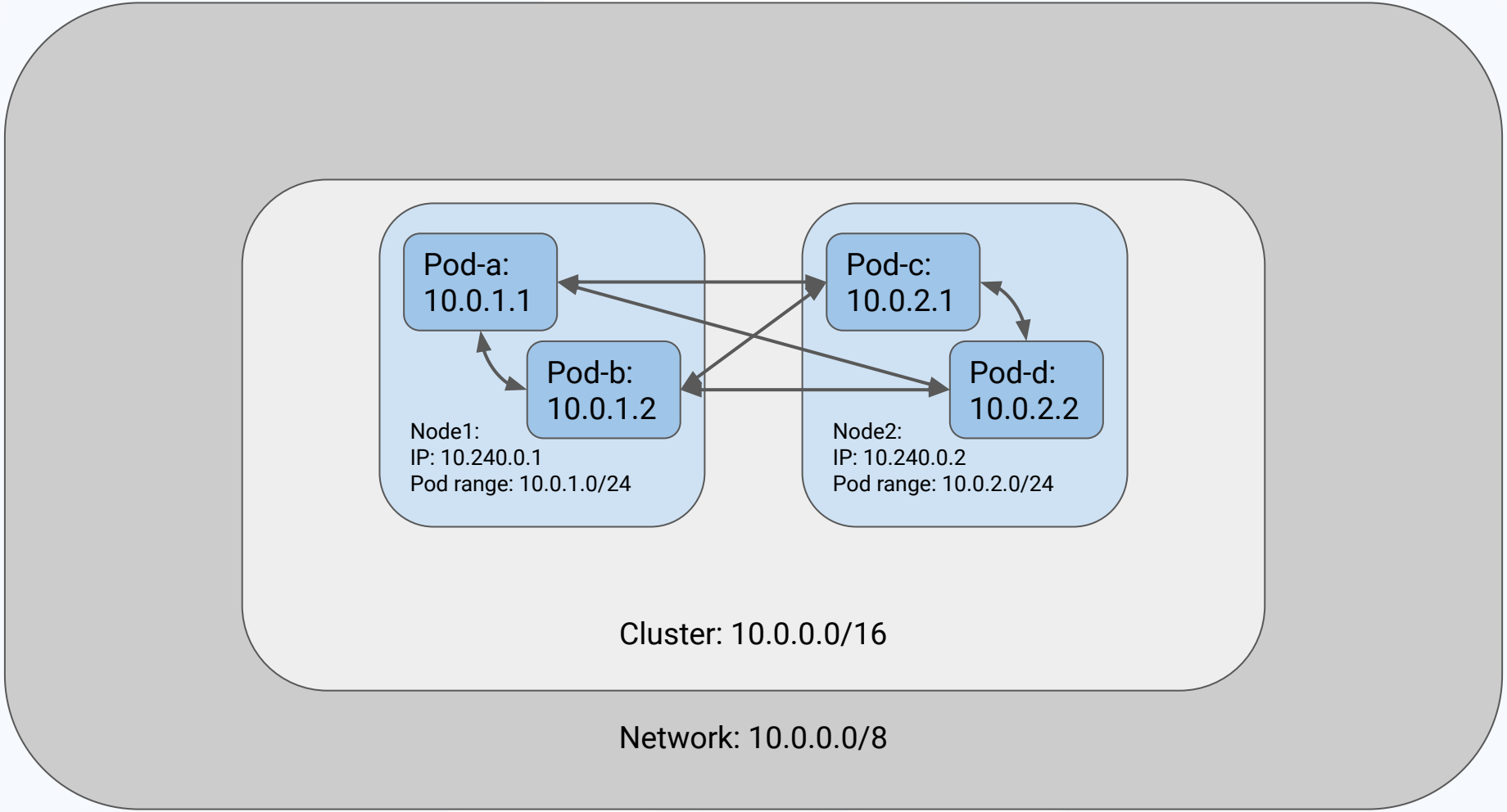
NOTE: It’s not **required** that nodes have a predefined IP range, but it’s very common and makes the pictures easier


Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

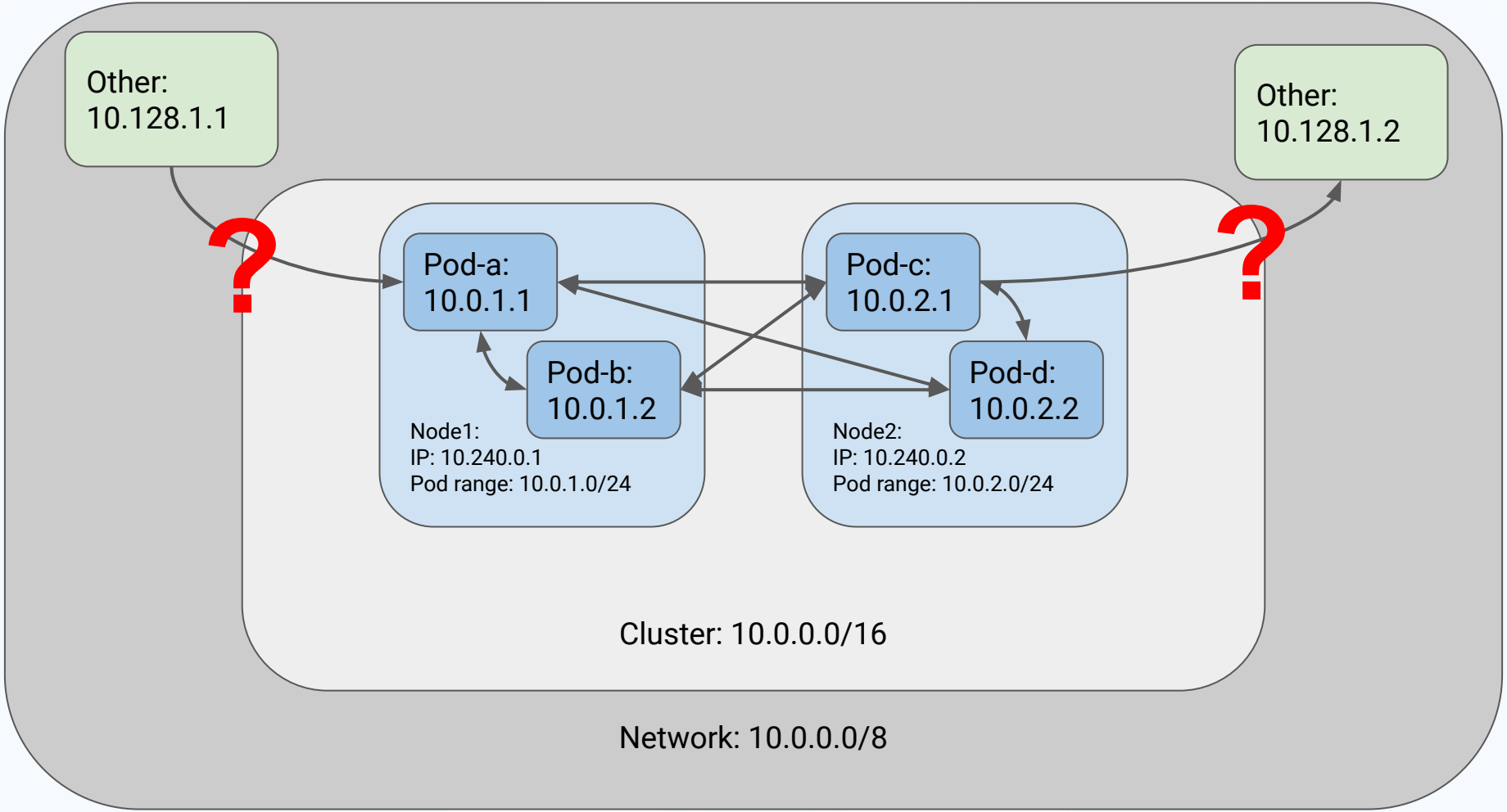

Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

Pods get IPs from the node’s IP range (again, not **always,** but usually)

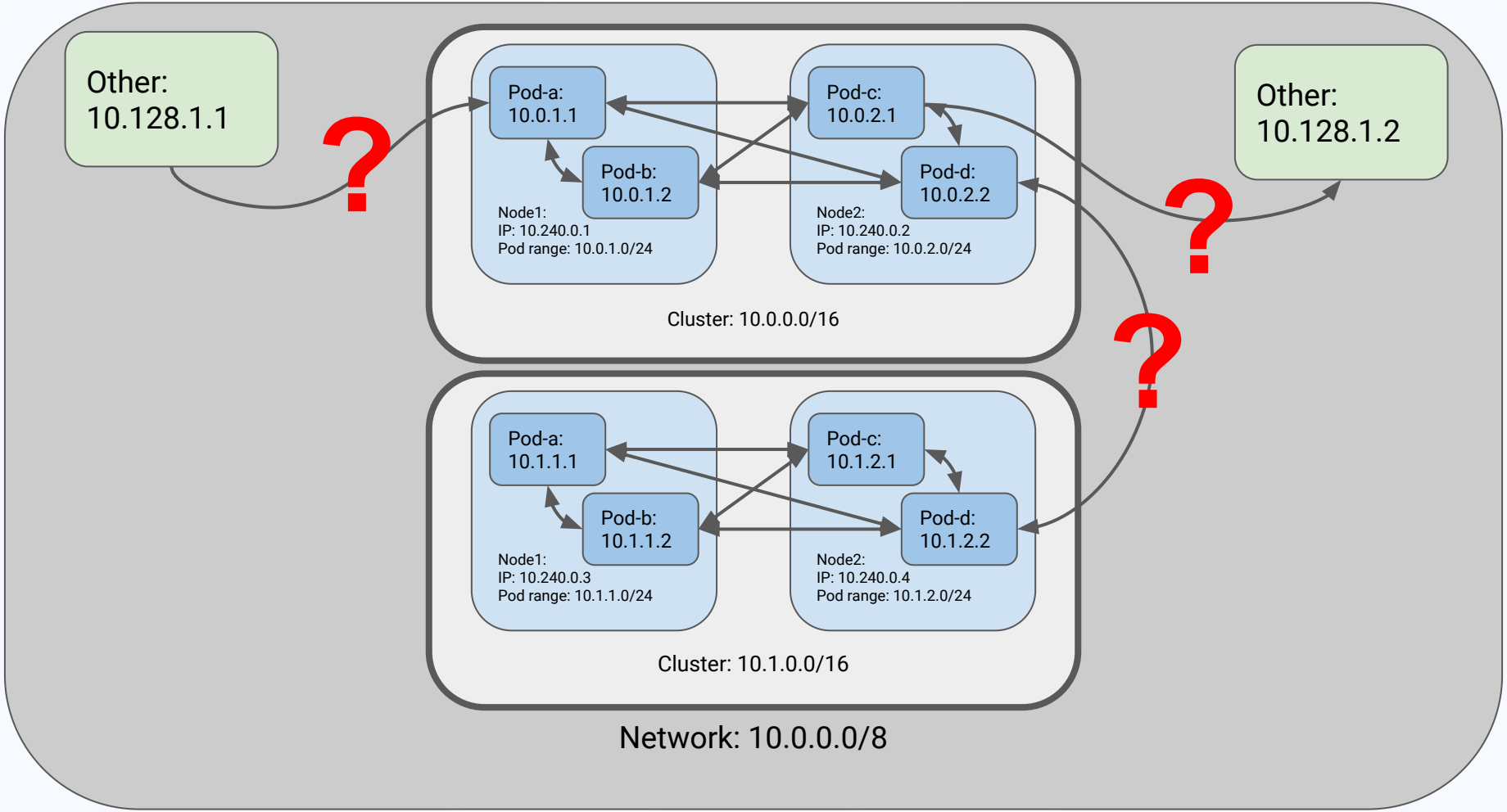

Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8


Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

Kubernetes **does not** say anything about things outside of the cluster

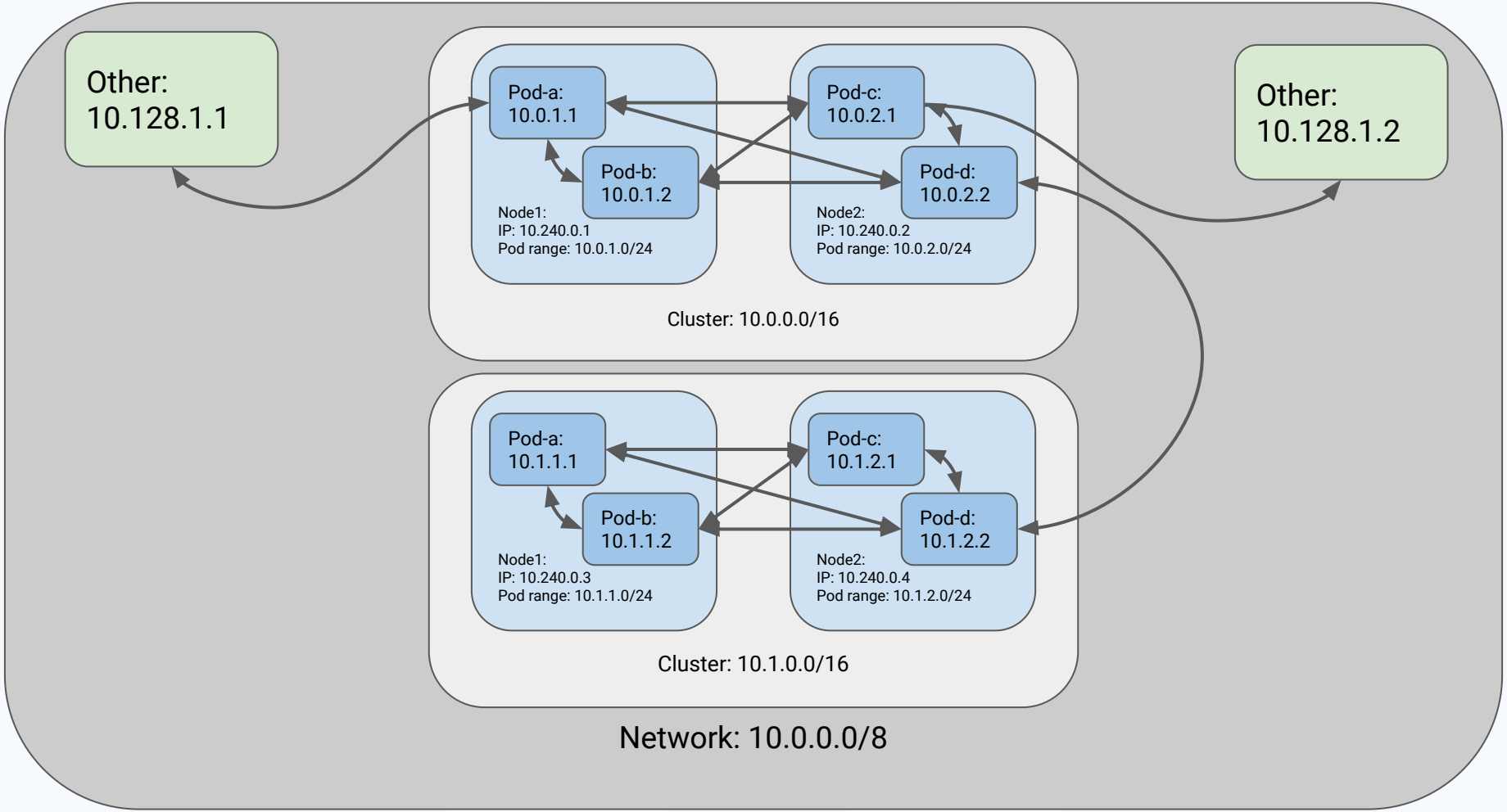

Other:
10.128.1.1
Other:
10.128.1.2
?
?
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

Multi-cluster makes it even more confusing


Other:
10.128.1.1
?
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
?
?
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.1.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

Network models (not exhaustive)

Fully-integrated (aka flat)


Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.1.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

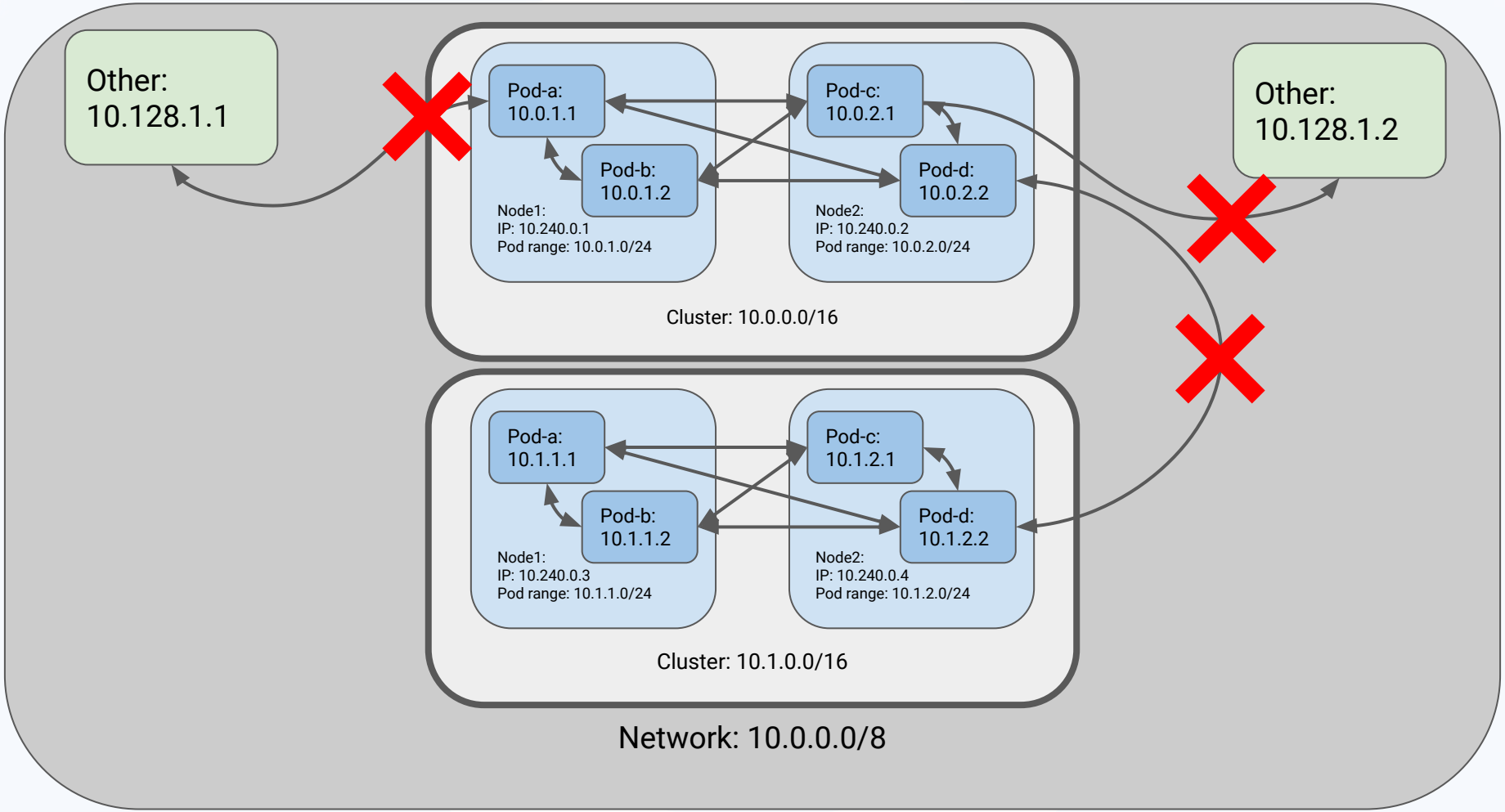
Good when:

- IP space is readily available
- Network is programmable / dynamic
- Need high integration / performance
- Kubernetes is a large part of your footprint

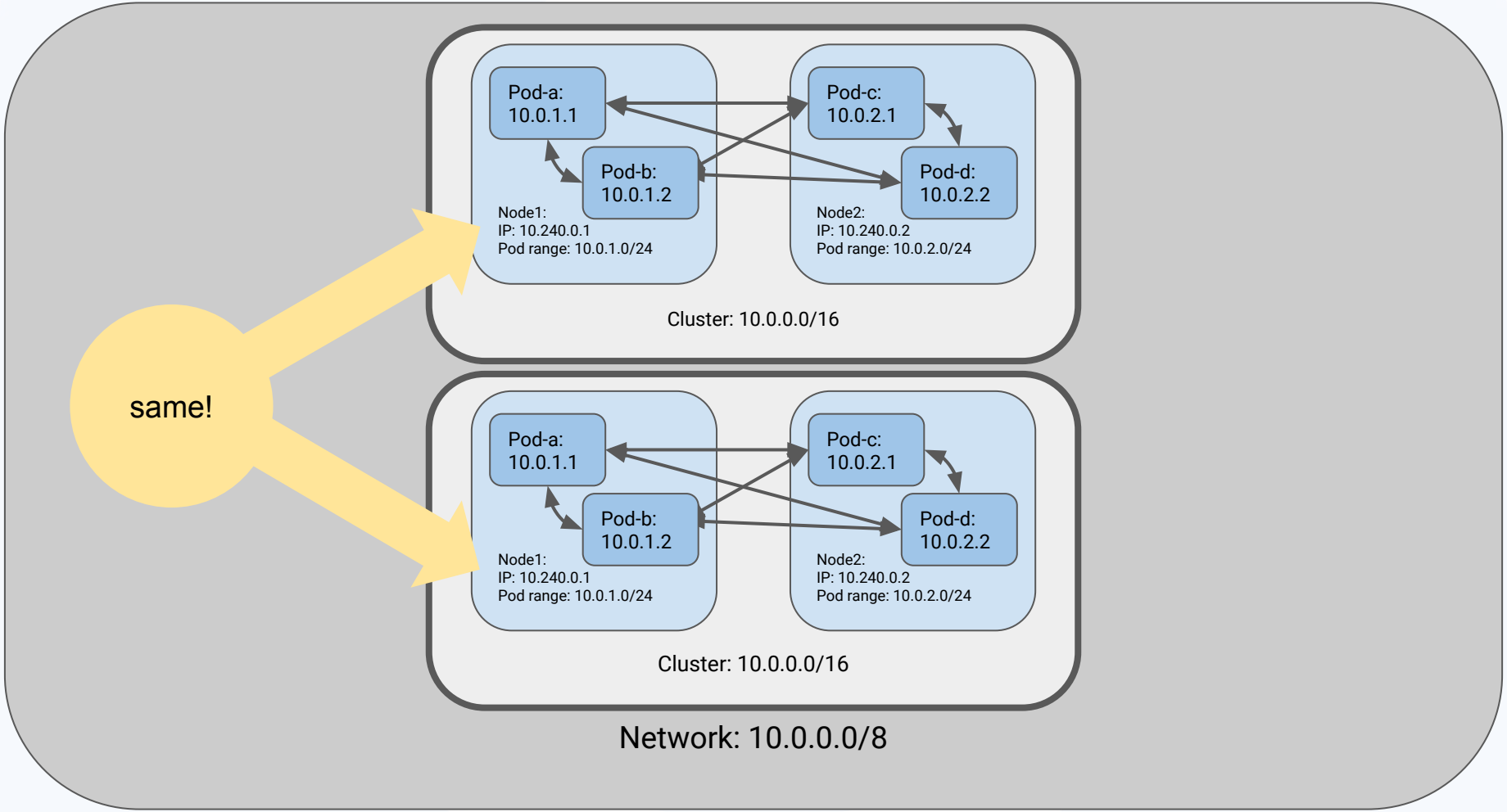
Bad when:

- IP fragmentation / scarcity
- Hard-to-configure network infrastructure
- Kubernetes is a small part of your footprint

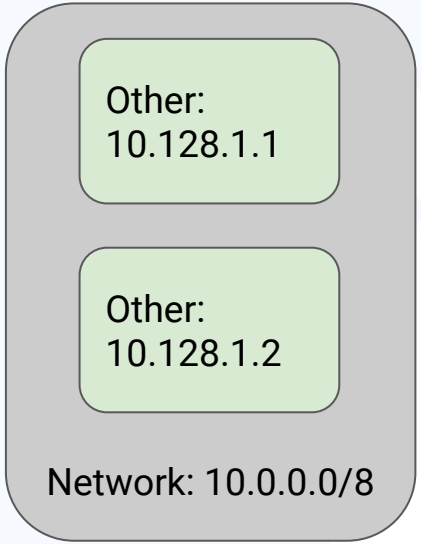
Fully-isolated (aka air-gapped)

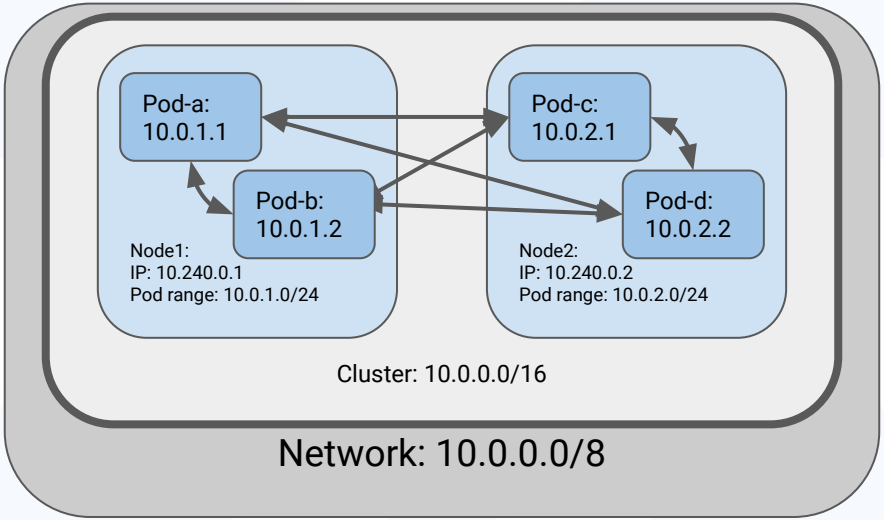

Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.1.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

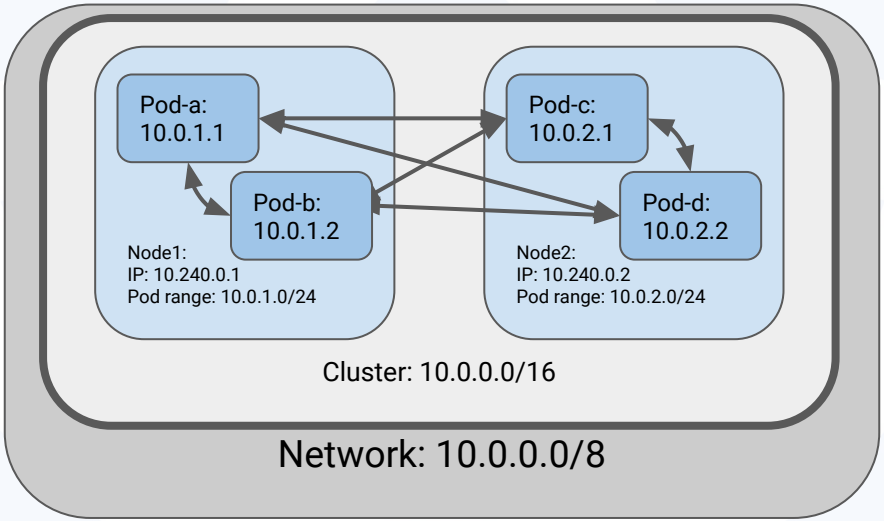
In fact, you can re-use all of the IPs


Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
same!
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

In fact, they are basically on different networks


Other:
10.128.1.1
Other:
10.128.1.2
Network: 10.0.0.0/8


Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8


Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

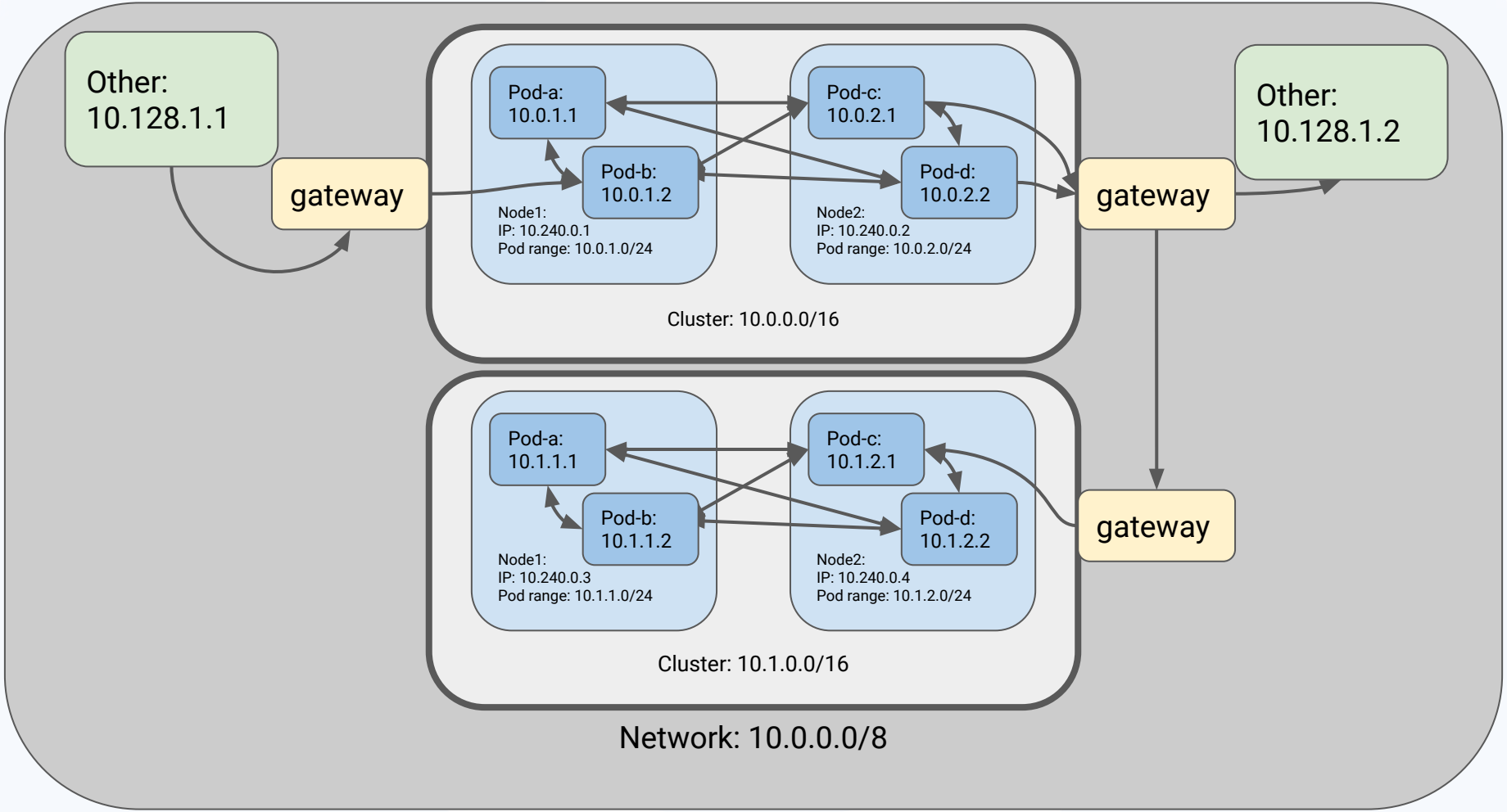
Good when:

- Don’t need integration
- IP space is scarce / fragmented
- Network is not programmable / dynamic
- May be easier to reason about security boundaries

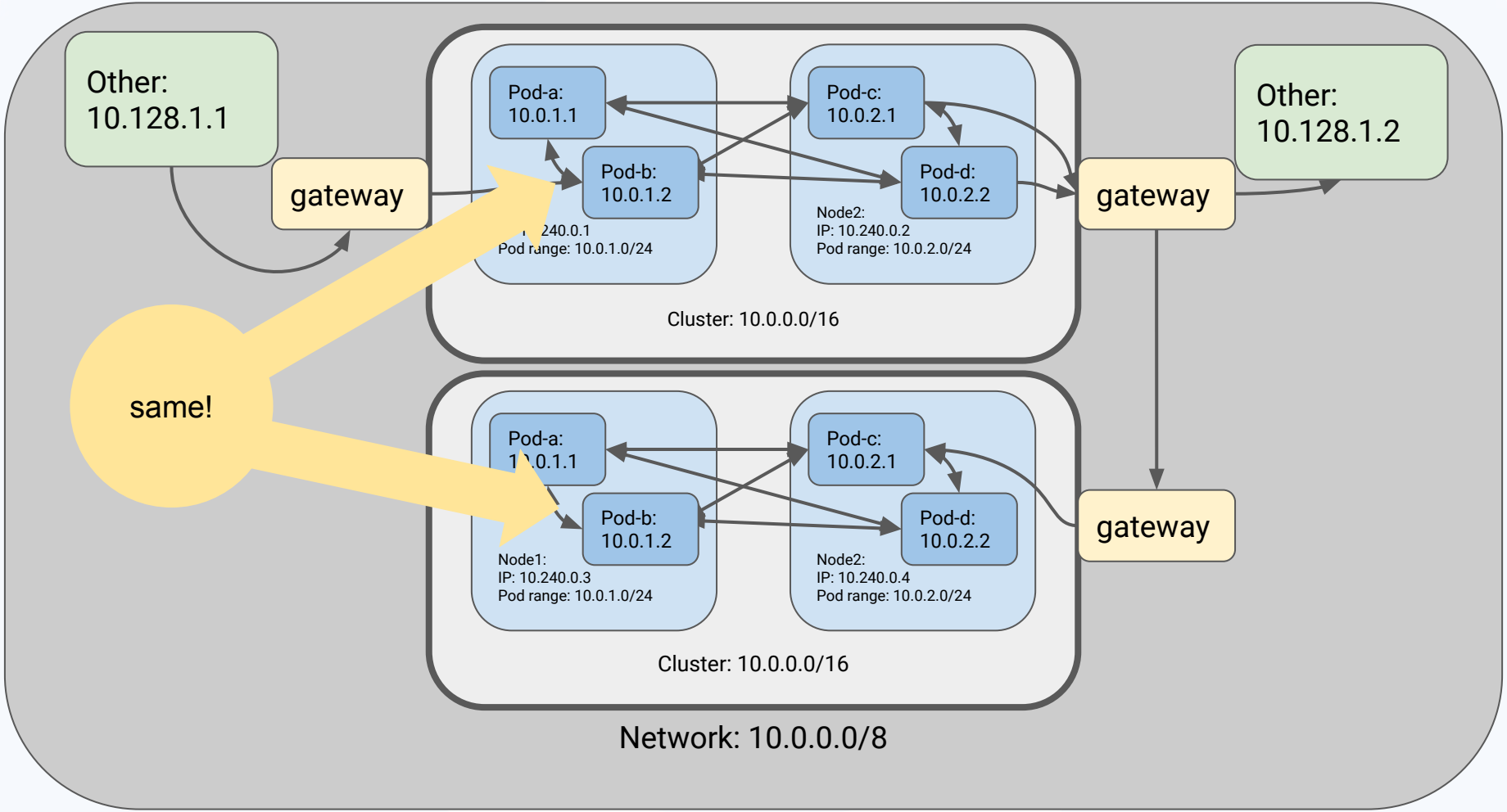
Bad when:

- Need communication across a cluster-edge

Bridged (aka island mode)


Other:
10.128.1.1
gateway
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
gateway
Other:
10.128.1.2
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.1.2.0/24
Cluster: 10.1.0.0/16
gateway
Network: 10.0.0.0/8

You can re-use the Pod IPs in each cluster (a major motivation for this model)


Other:
10.128.1.1
gateway
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Pod range: 10.0.1.0/24
Cluster: 10.0.0.0/16
gateway
Other:
10.128.1.2
same!
Pod-a:
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
gateway
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

Good when:

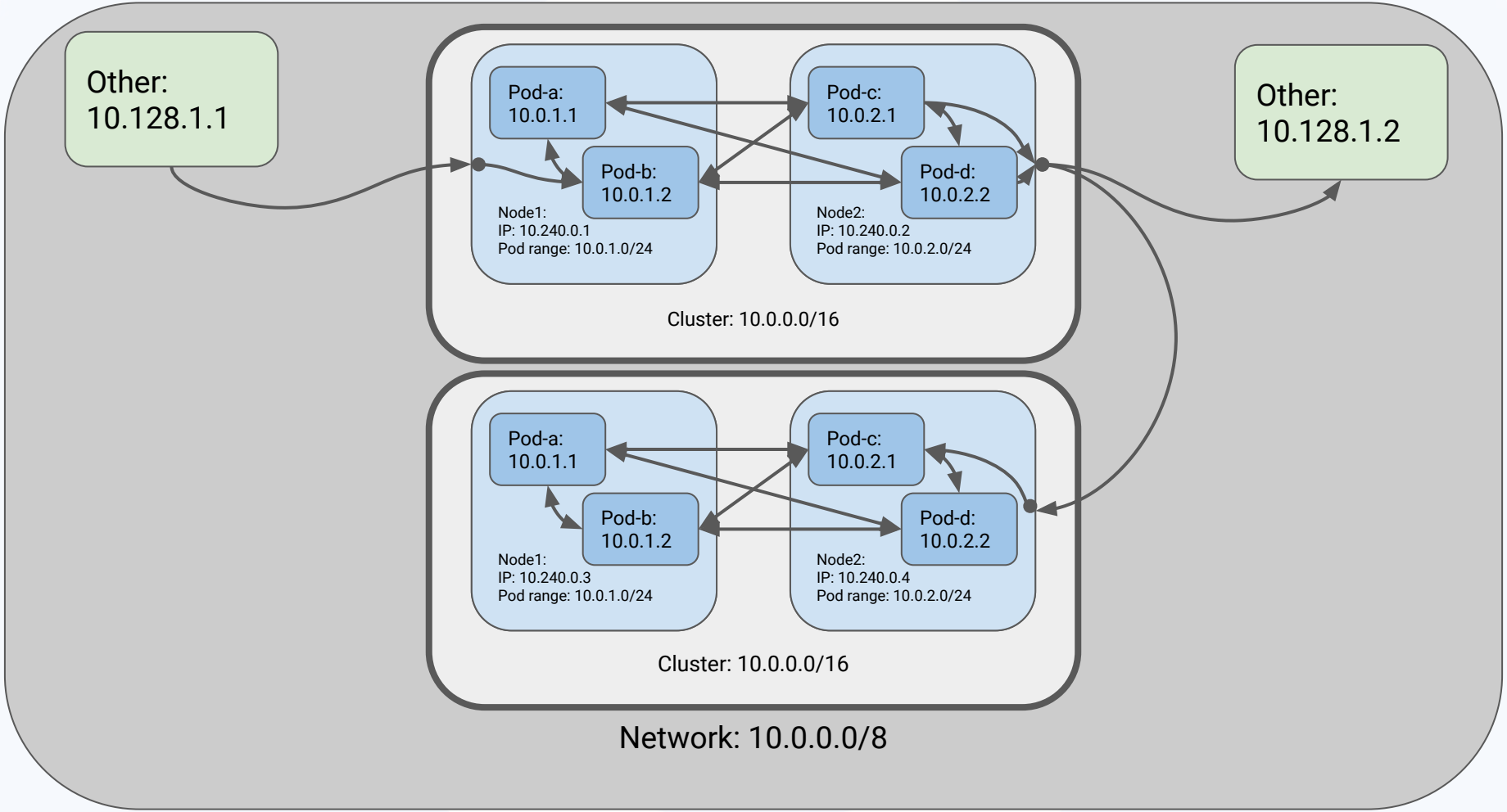
- Need some integration
- IP space is scarce / fragmented
- Network is not programmable / dynamic

Bad when:

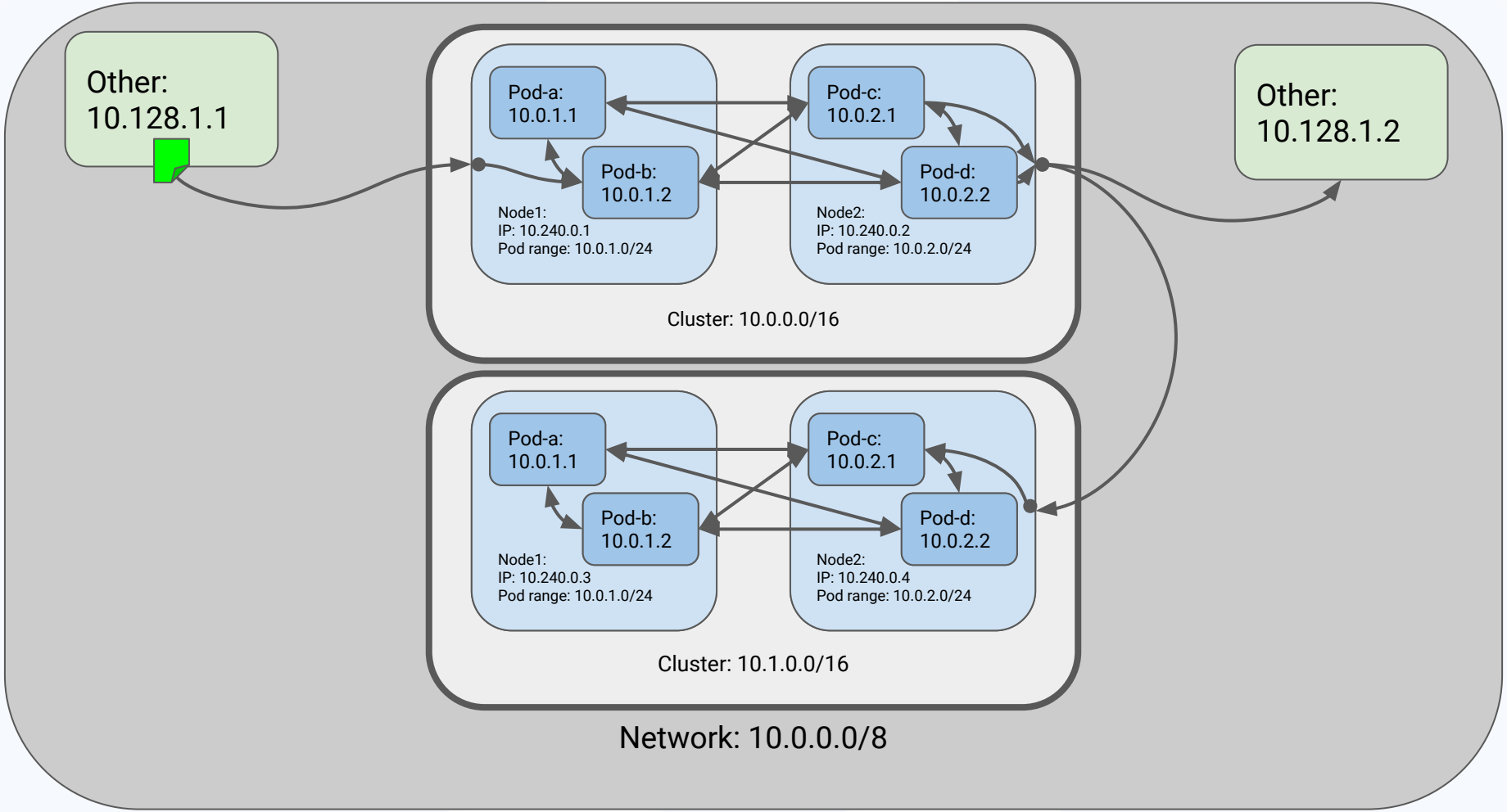
- Need to debug connectivity
- Need direct-to-endpoint communications
- Need a lot of services exposed (especially non-HTTP)
- Rely on client IPs for firewalls
- Large number of nodes

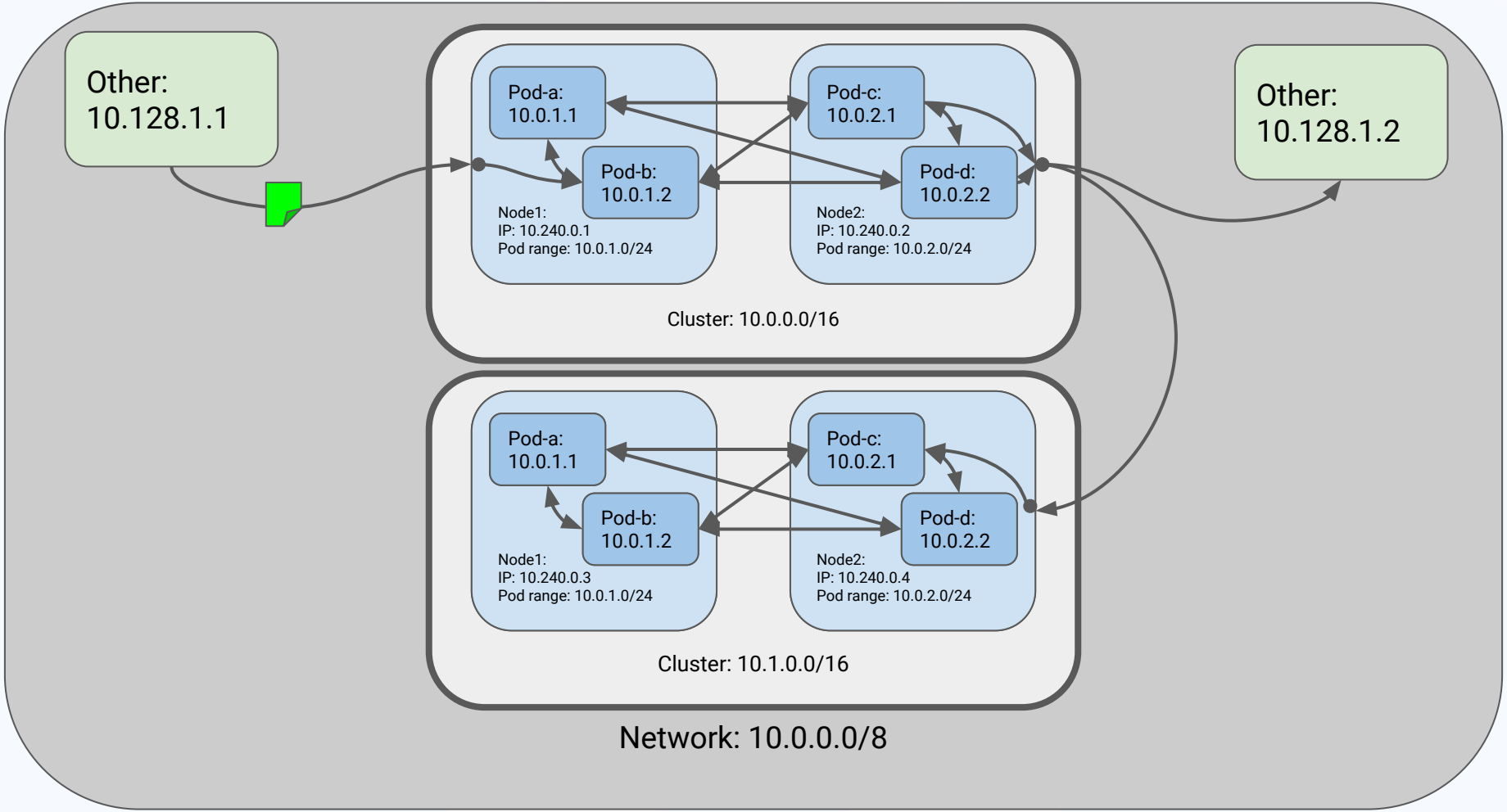
Various forms of “gateway”

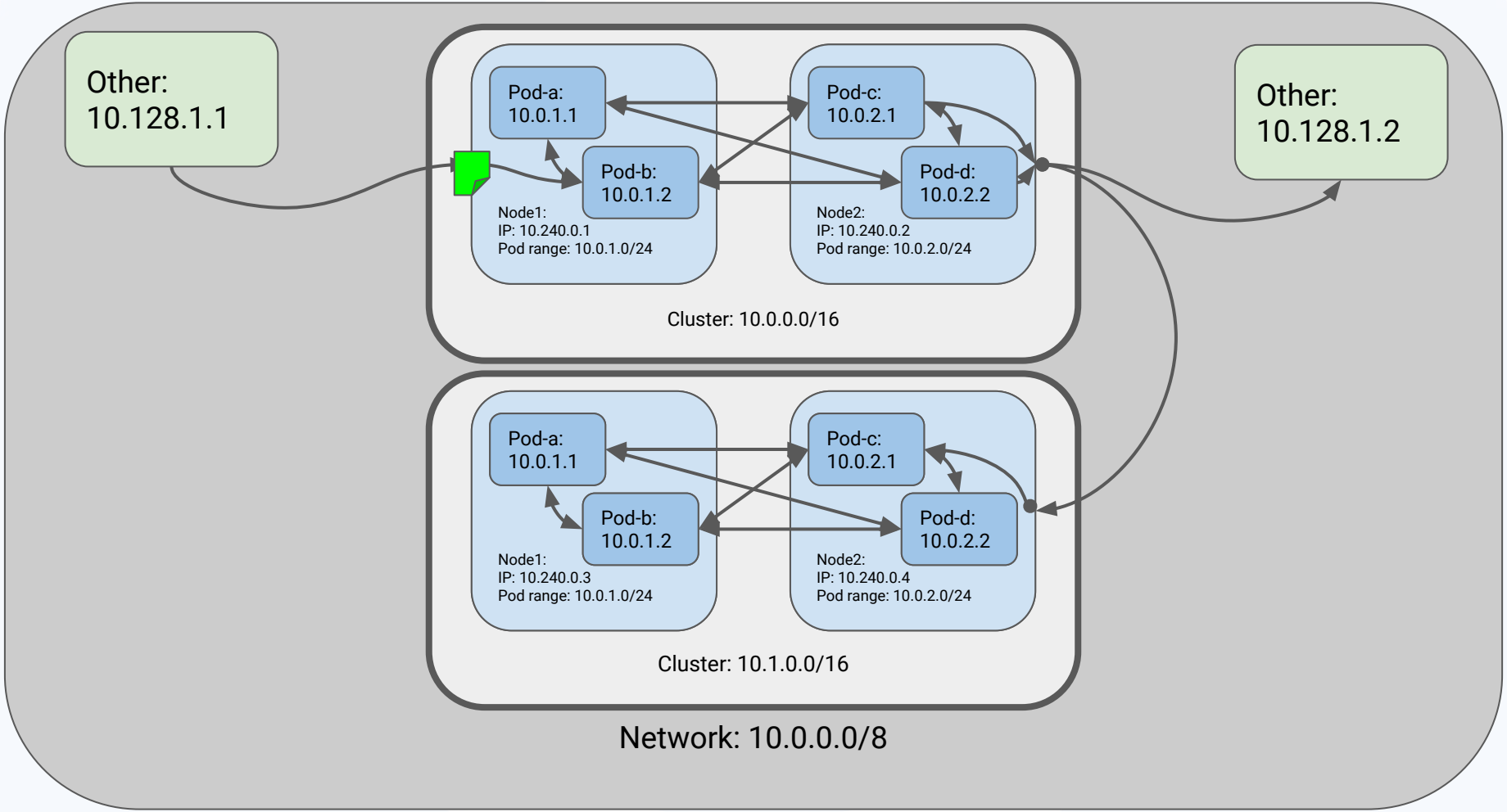
Gateway: nodes


Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/8

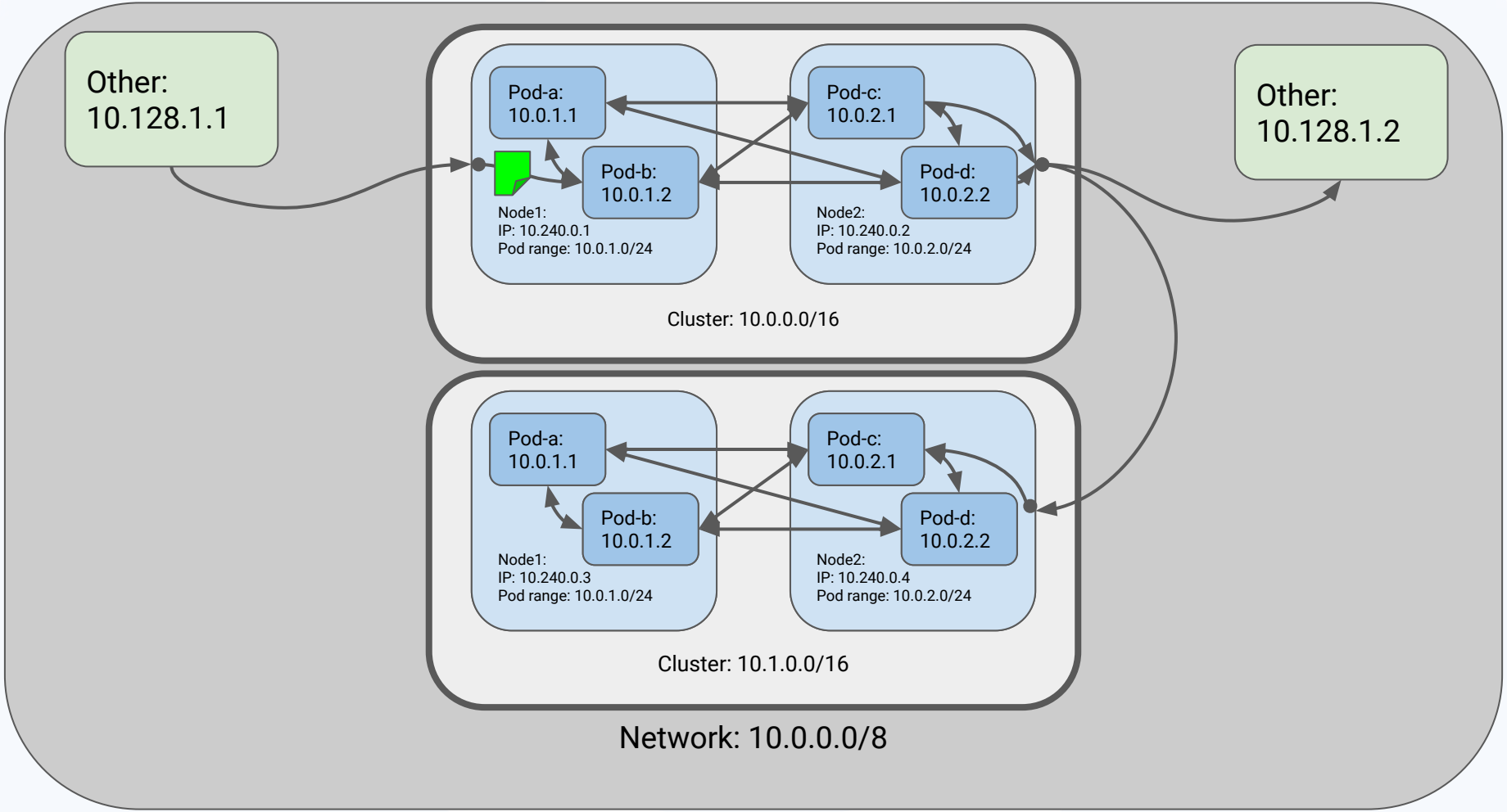
Ingress: Service NodePorts

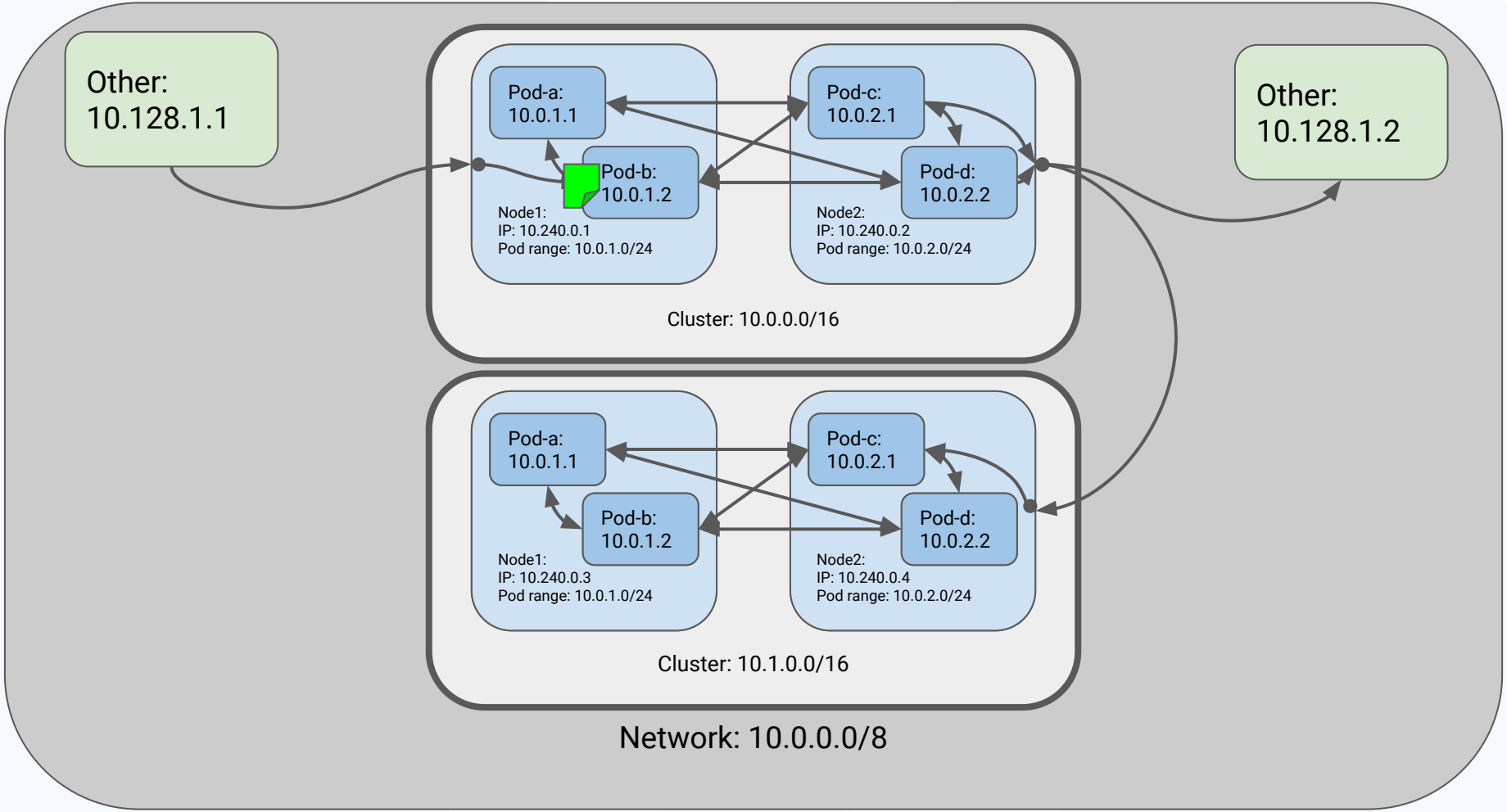

Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8


Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8


Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

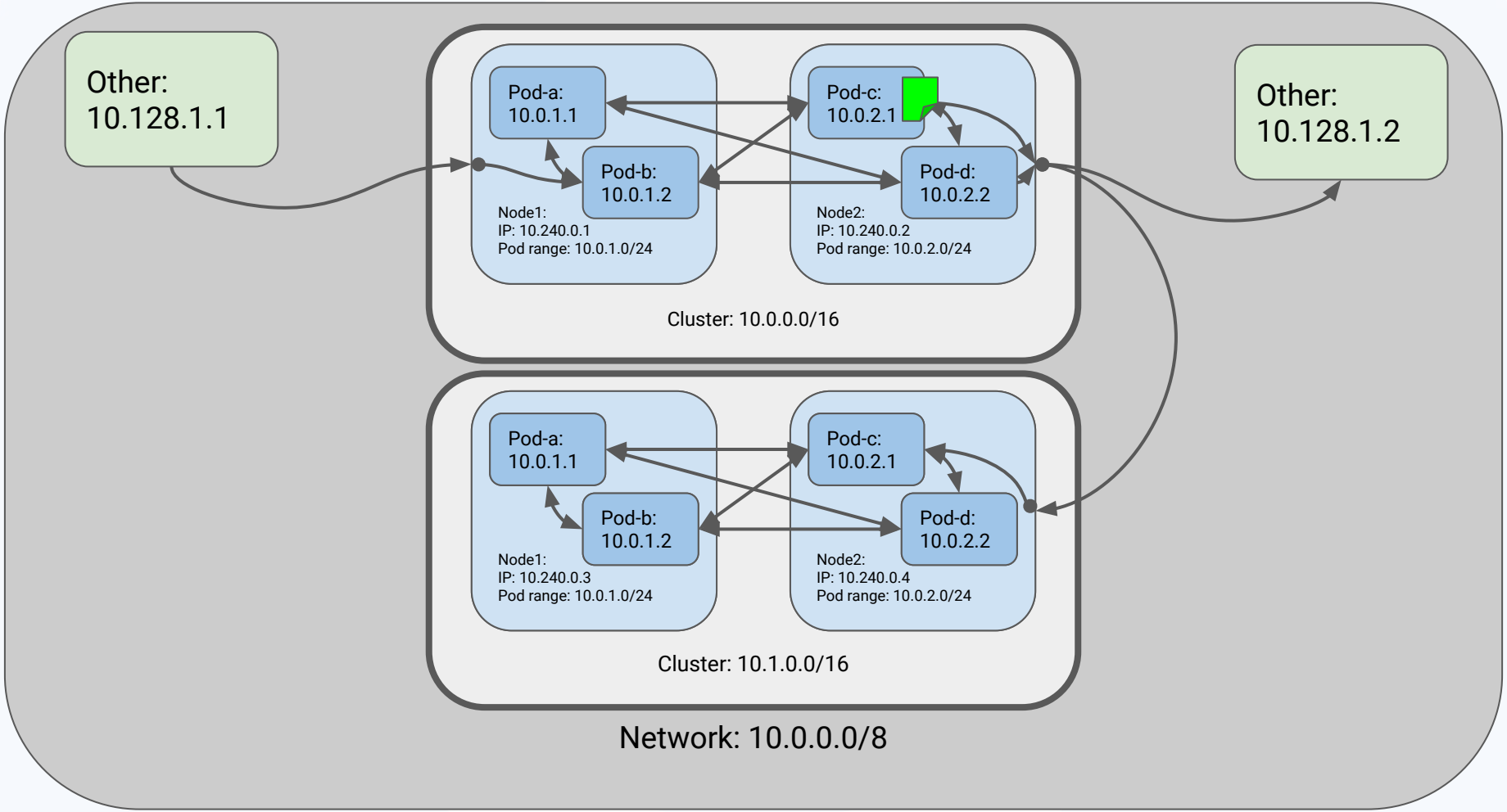
Node uses IP **dst_port** to route to correct service (DNAT)

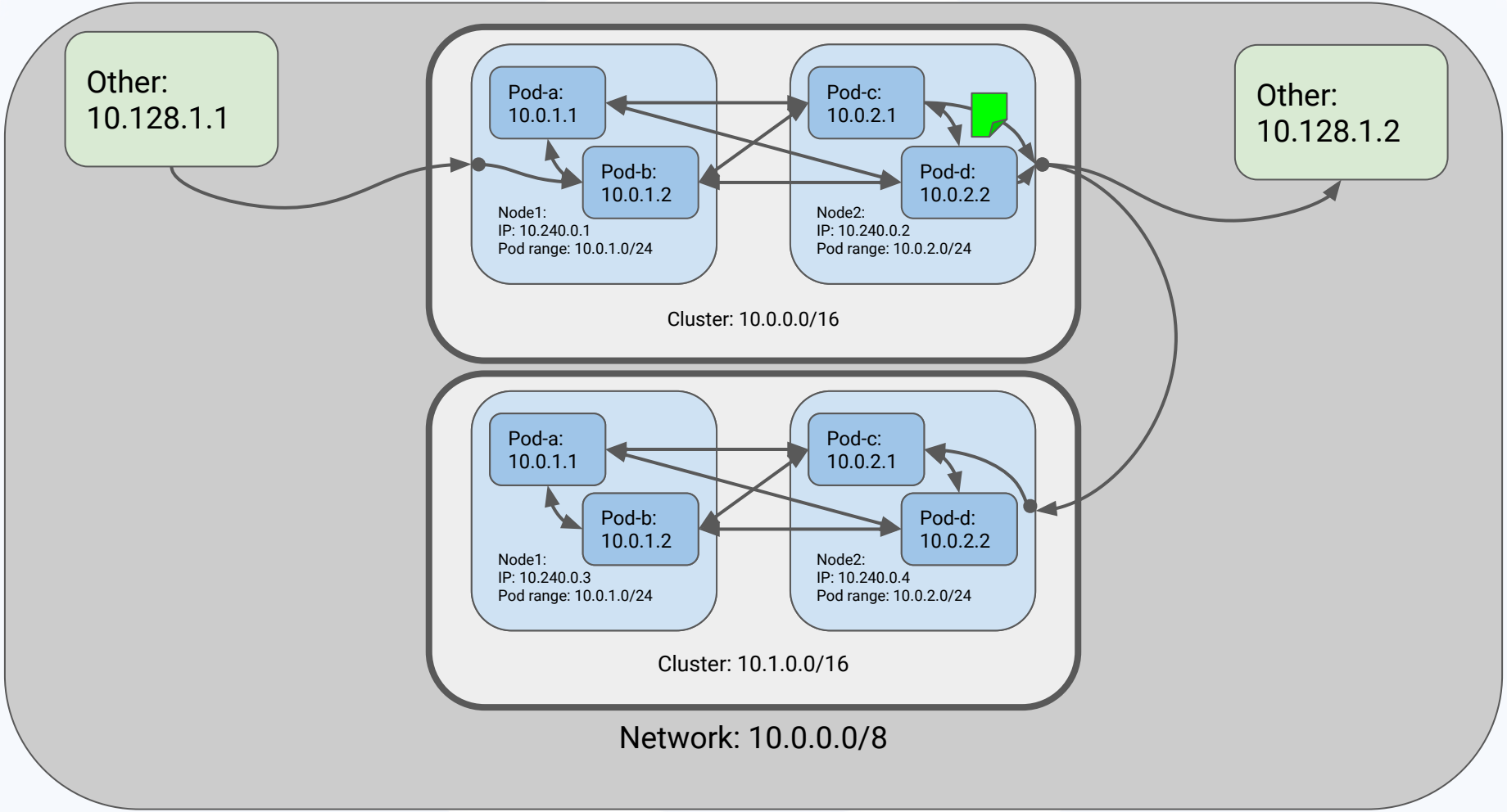

Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

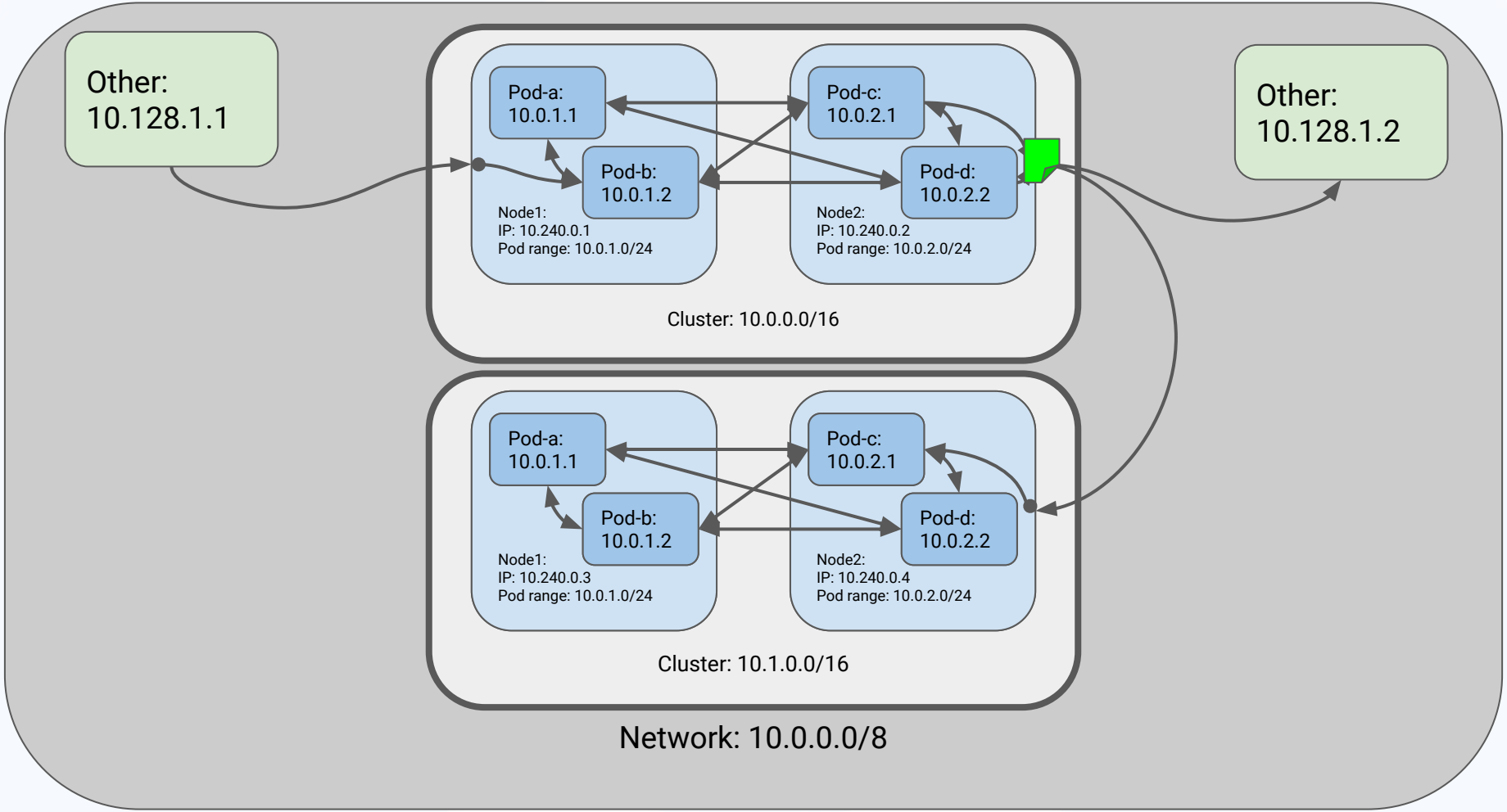

Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

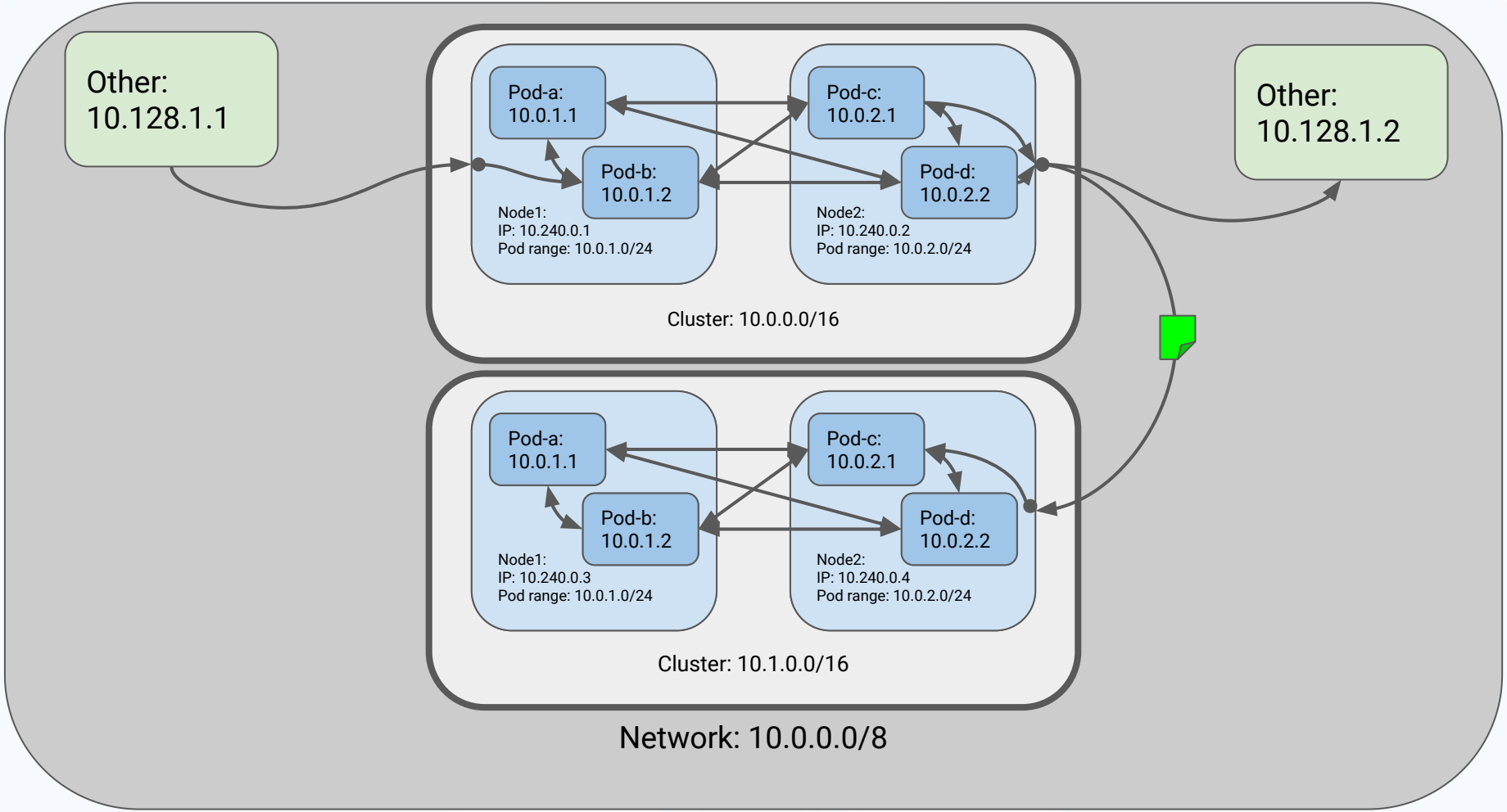
You can ingress L4 into an L7 proxy and forward from there (e.g. in-cluster ingress controllers)

Egress: IP Masquerade (aka SNAT)

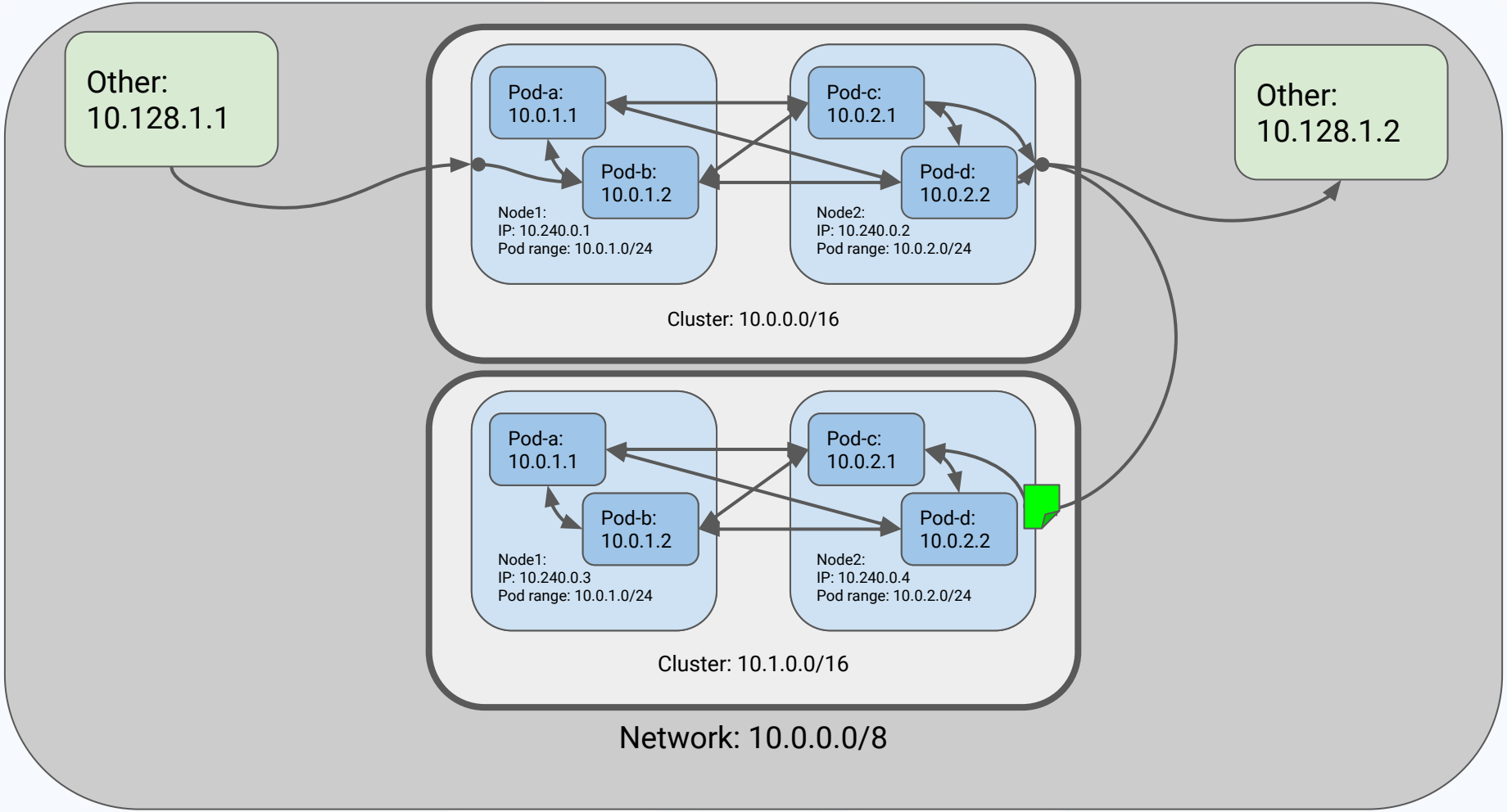

Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

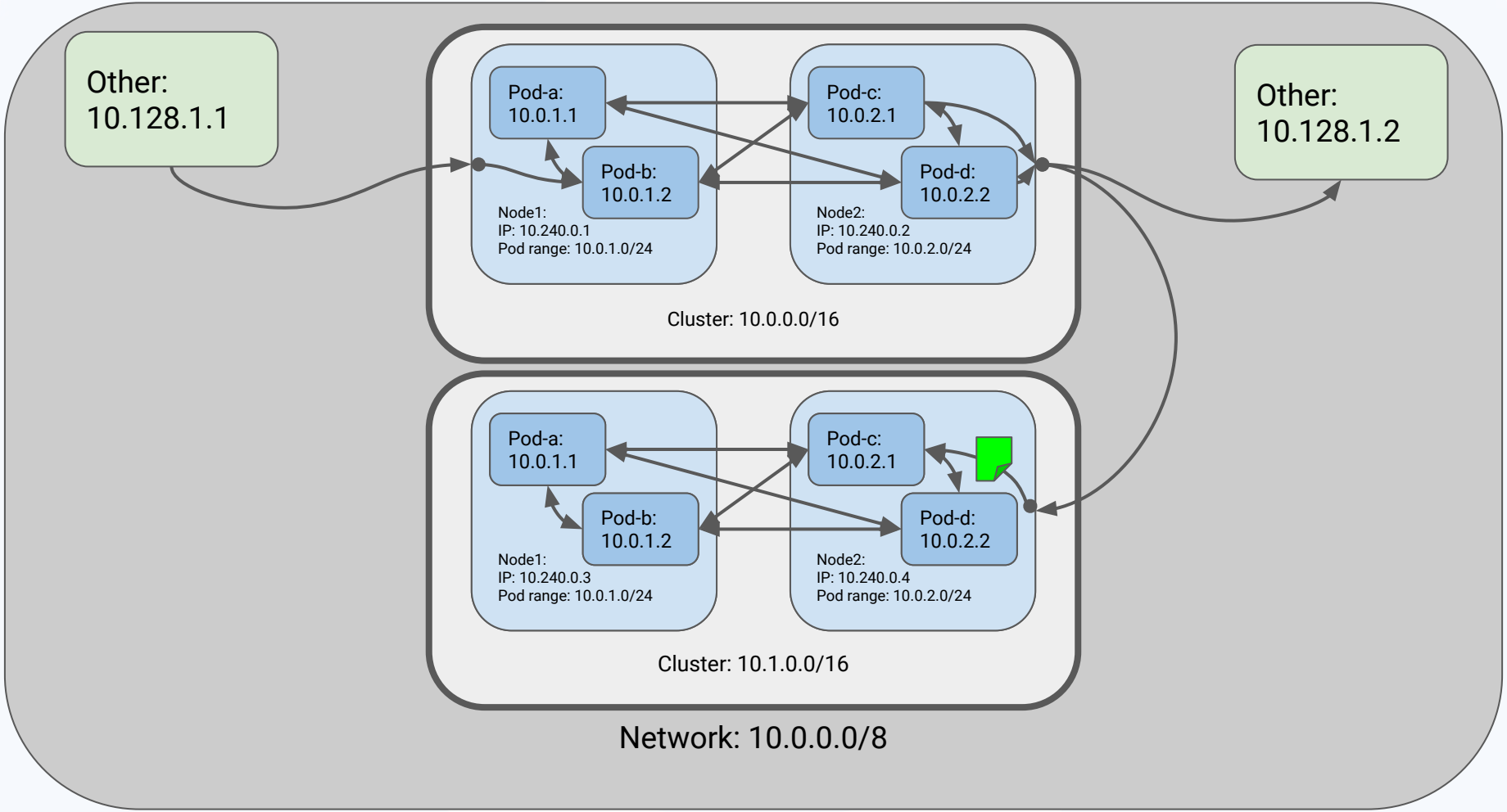

Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

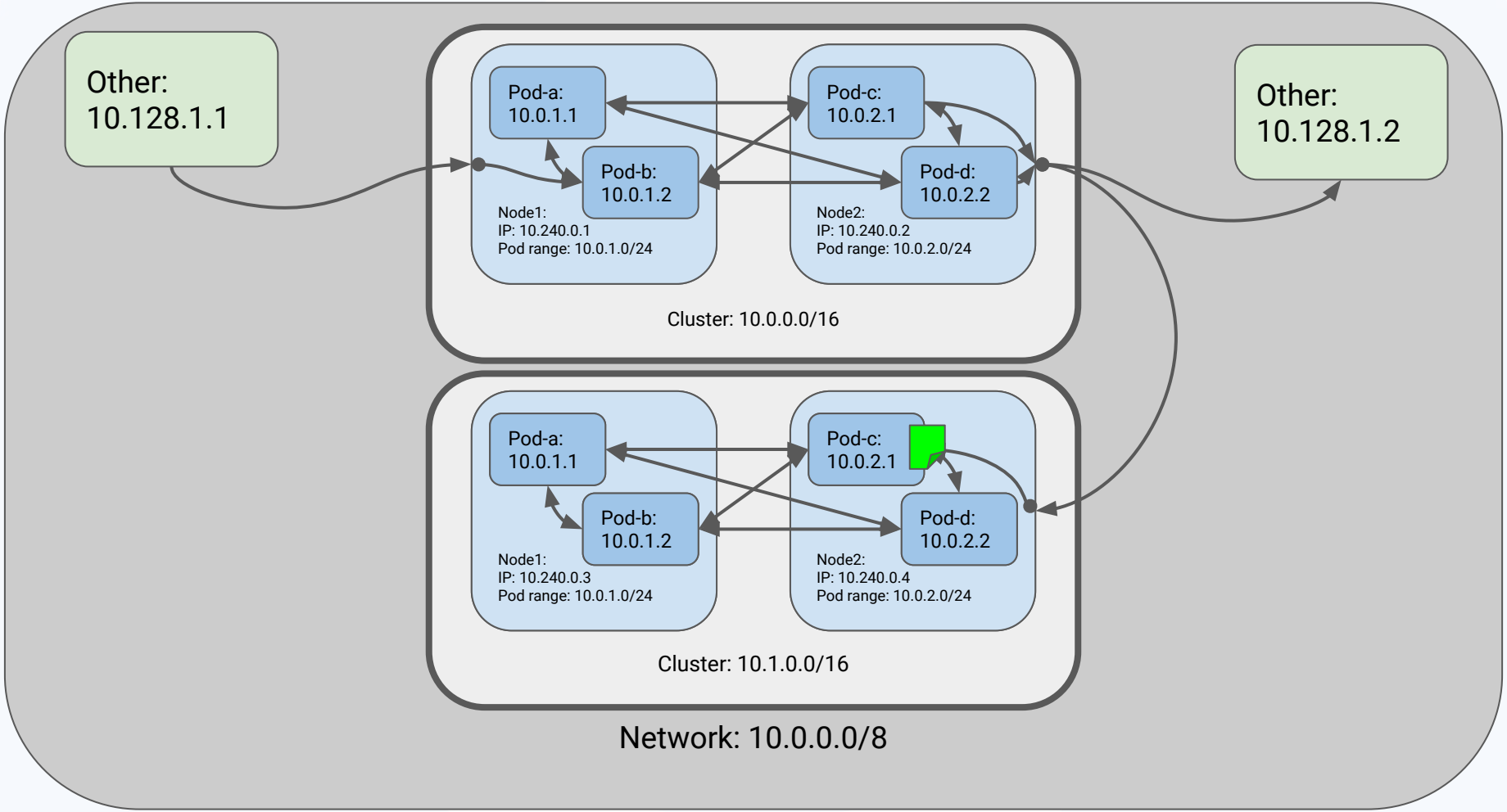

Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8


Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

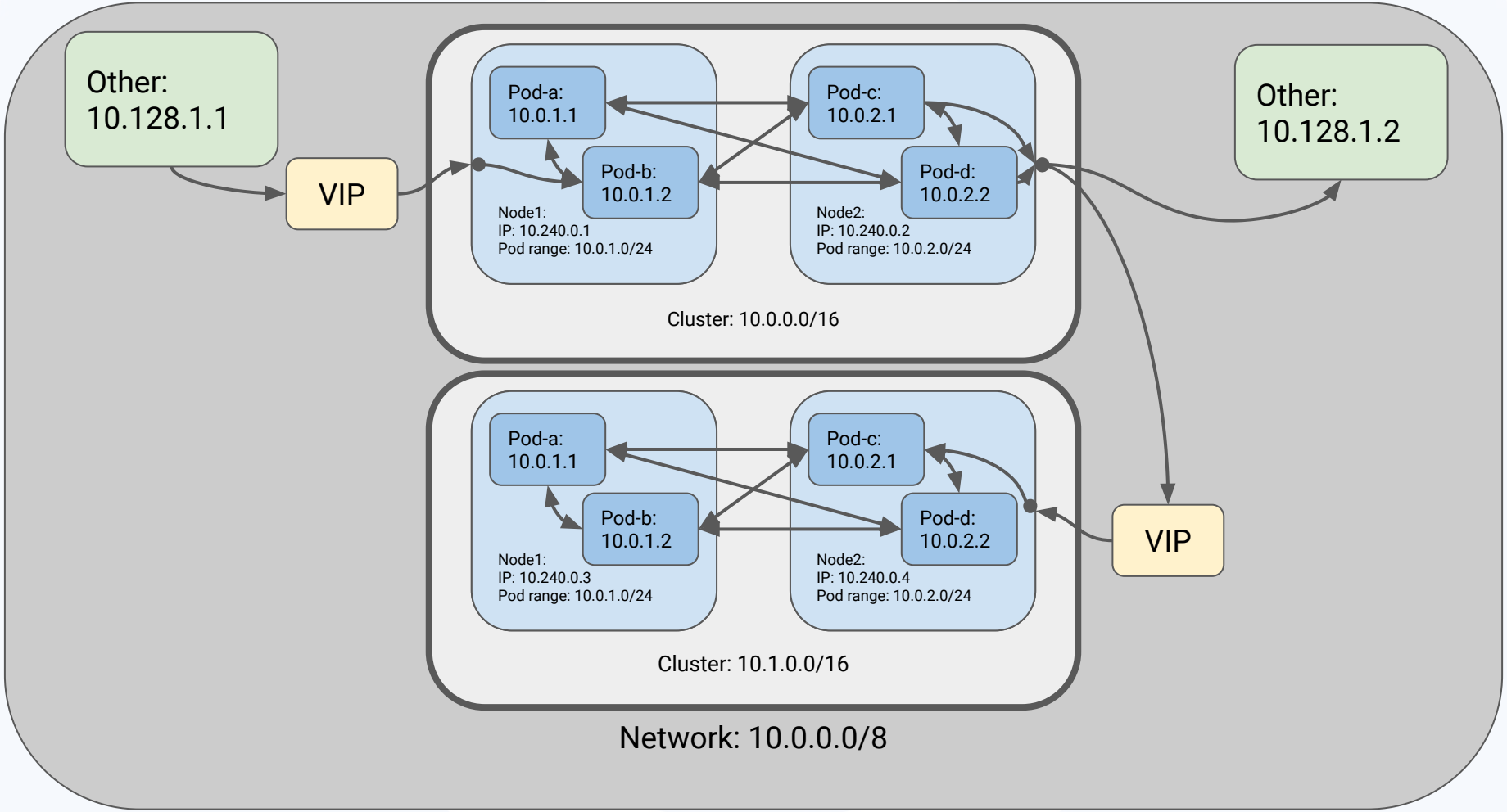
SNAT obscures client IP (Traffic from pods on a node appears to come from that node’s IP)


Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8


Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8


Other:
10.128.1.1
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

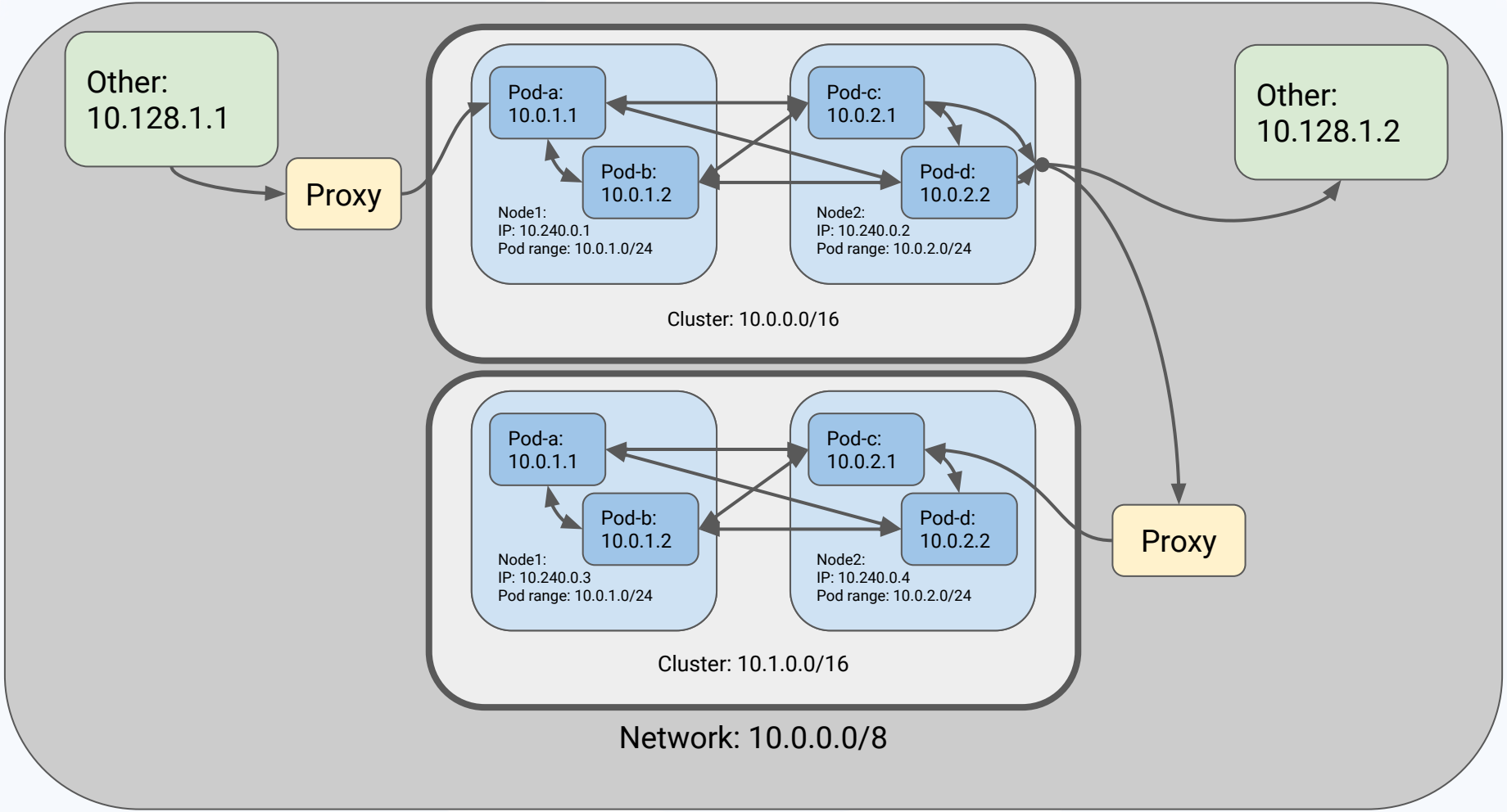
Gateway: VIP (ingress)


Other:
10.128.1.1
VIP
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
VIP
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

Similar to NodePort, but node uses IP **dst_ip** to route

Still needs something like SNAT to egress

Gateway: Proxy (ingress)


Other:
10.128.1.1
Proxy
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
Other:
10.128.1.2
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.0.2.0/24
Cluster: 10.1.0.0/16
Proxy
Network: 10.0.0.0/8

Can either route to NodePort or directly to pod IPs (e.g. proxy knows how to “get onto the island”)

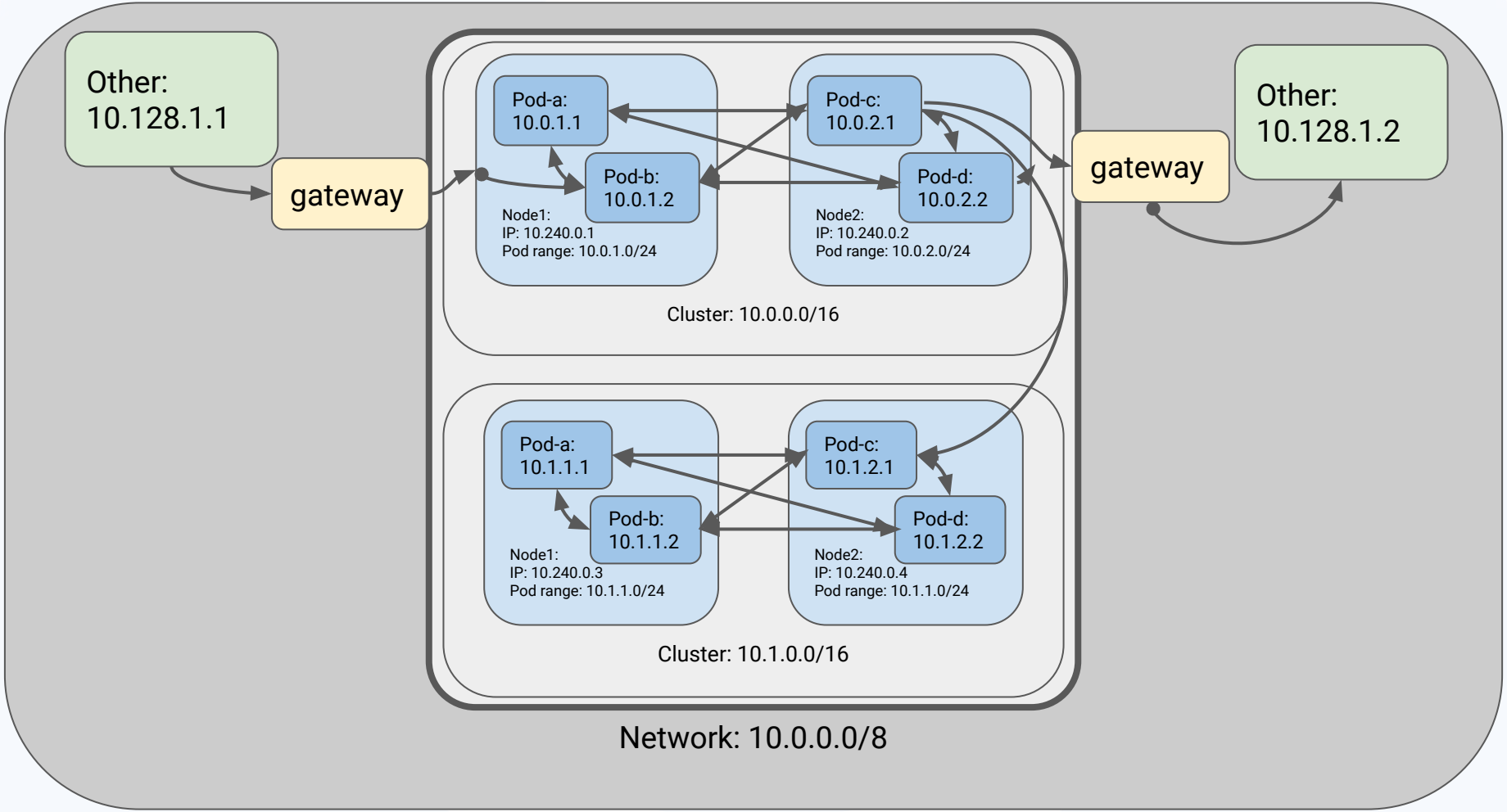
Proxy obscures client IP (Traffic appears to come from the proxy’s IP)

Still needs something like SNAT to egress

There’s a LOT more to know about ingress (for another presentation)

Options for egress are poorly explored, so far

Archipelago
(aka bigger islands)


Other:
10.128.1.1
gateway
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1:
IP: 10.240.0.1
Pod range: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2:
IP: 10.240.0.2
Pod range: 10.0.2.0/24
Cluster: 10.0.0.0/16
gateway
Other:
10.128.1.2
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1:
IP: 10.240.0.3
Pod range: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2:
IP: 10.240.0.4
Pod range: 10.1.1.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/8

Can’t reuse pod IPs between clusters, but can between archipelagos

Good when:

- Need high integration across clusters
- Need some integration with non-kubernetes
- IP space is scarce / fragmented
- Network is not programmable / dynamic

Bad when:

- Need to debug connectivity
- Need direct-to-endpoint communications
- Need a lot of services exposed to non-k8s
- Rely on client IPs for firewalls
- Large number of nodes across all clusters

Gateway options are similar to plain island mode

Which one should you use?

There is no “right answer”. You have to consider the tradeoffs.

Sorry.

Questions?

Sept 25:
Ambassador webinar

Kaslin Fields and Bowei Du will present the webinar
“The evolution of Ingress through the Gateway API”

Follow https://www.cncf.io/upcoming-webinars/
for more details