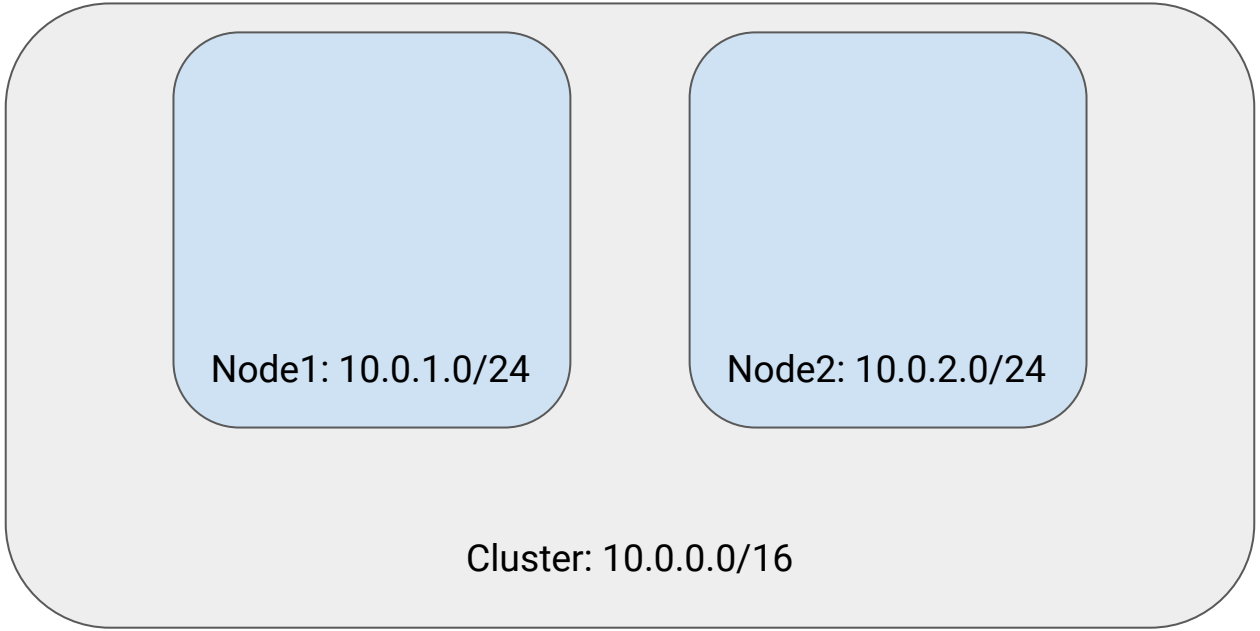
Kubernetes and networks

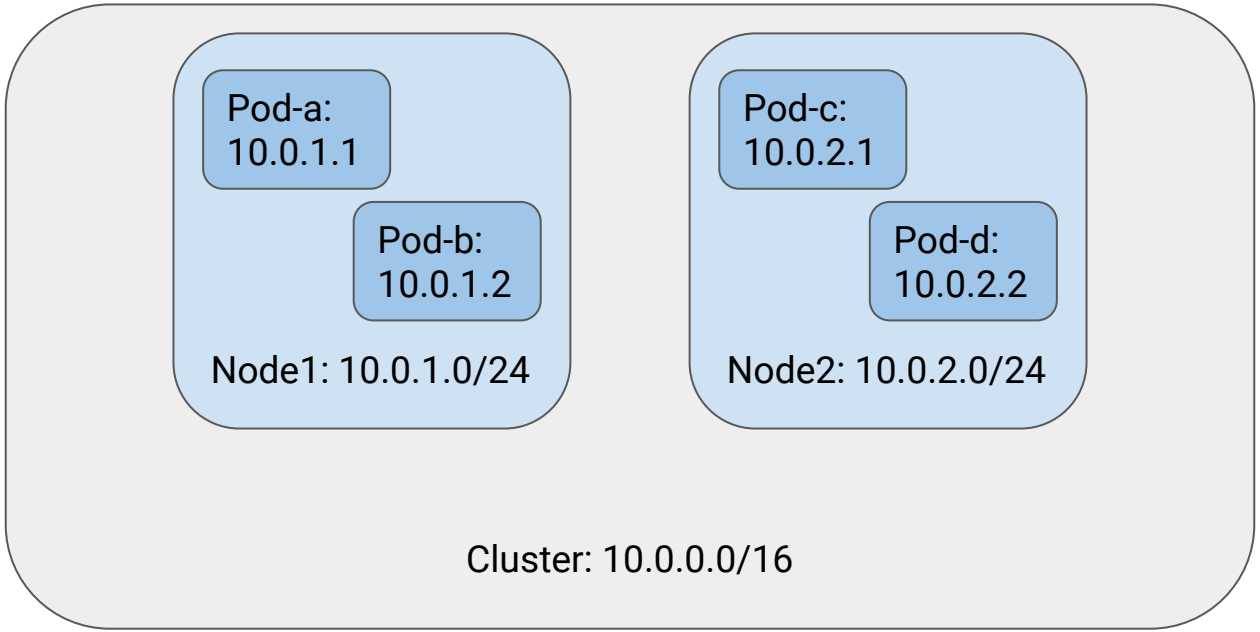
Why is this so dang hard?

Tim Hockin
@thockin

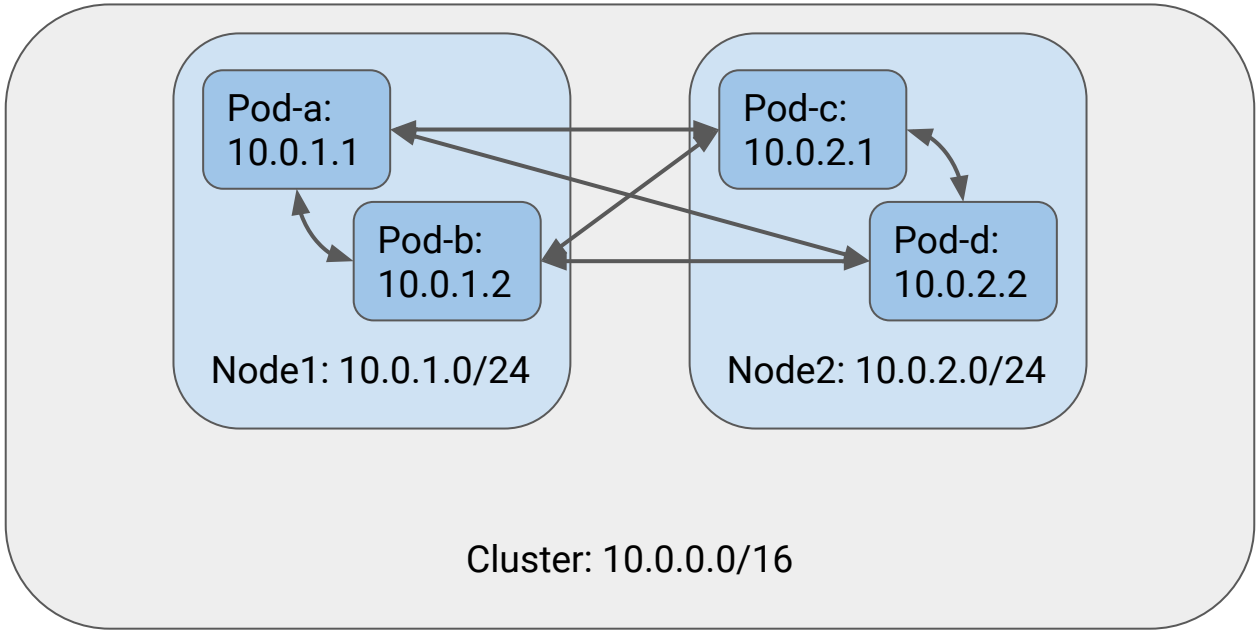
Start with a “normal” cluster


Cluster: 10.0.0.0/16

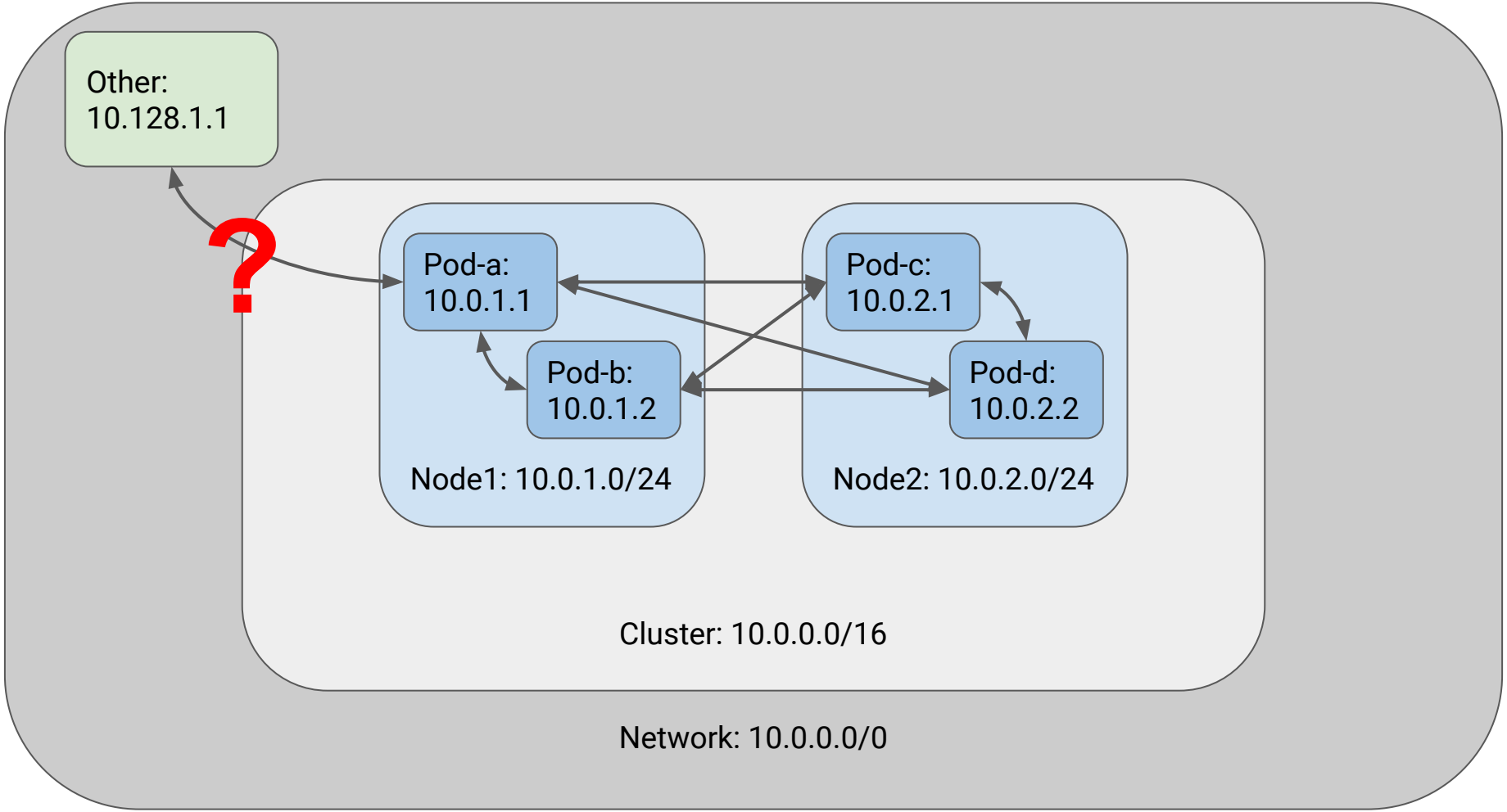

Node1: 10.0.1.0/24
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16


Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16

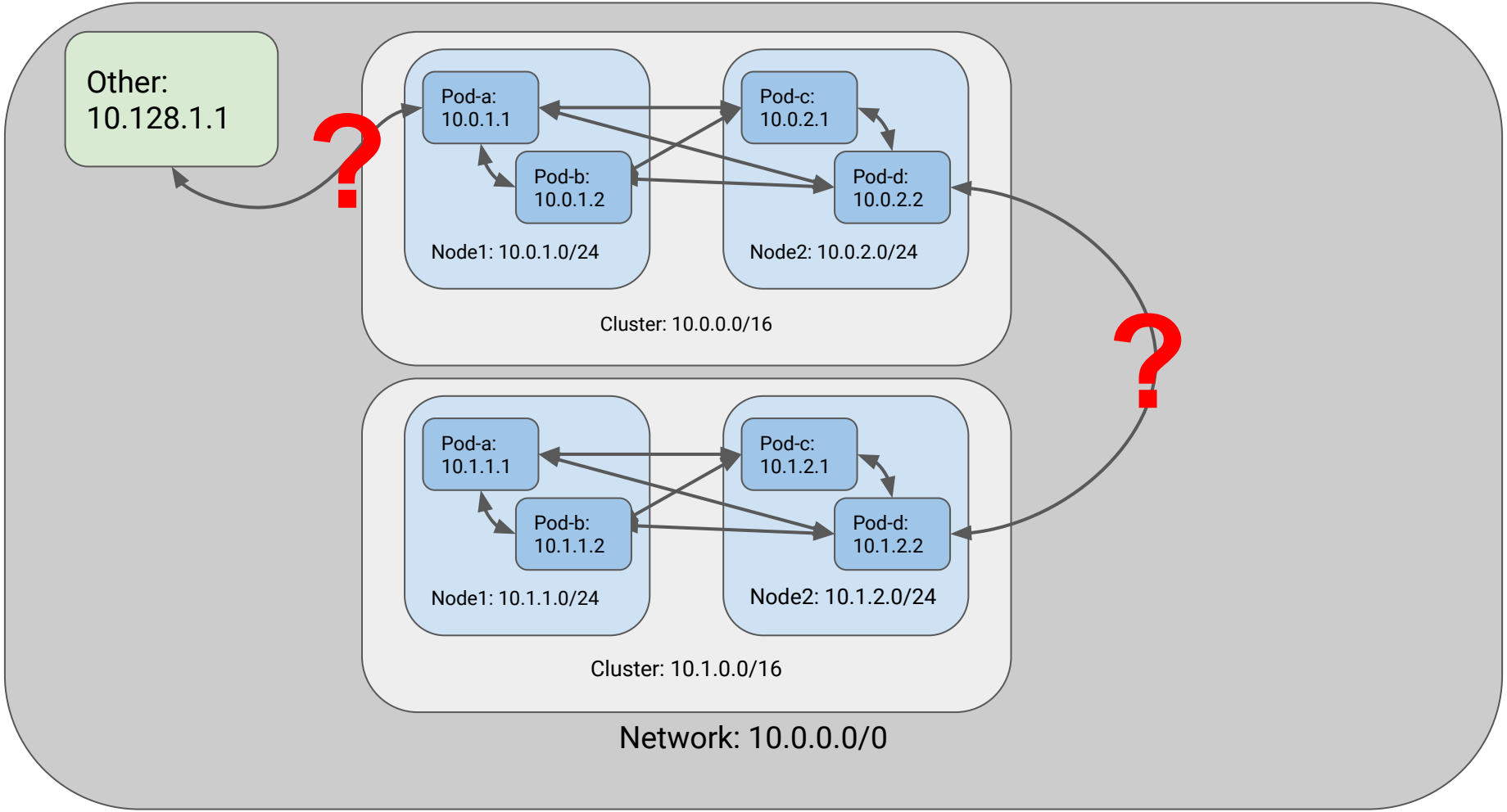
Kubernetes demands that pods can reach each other


Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16

Kubernetes does not say anything about things outside of the cluster

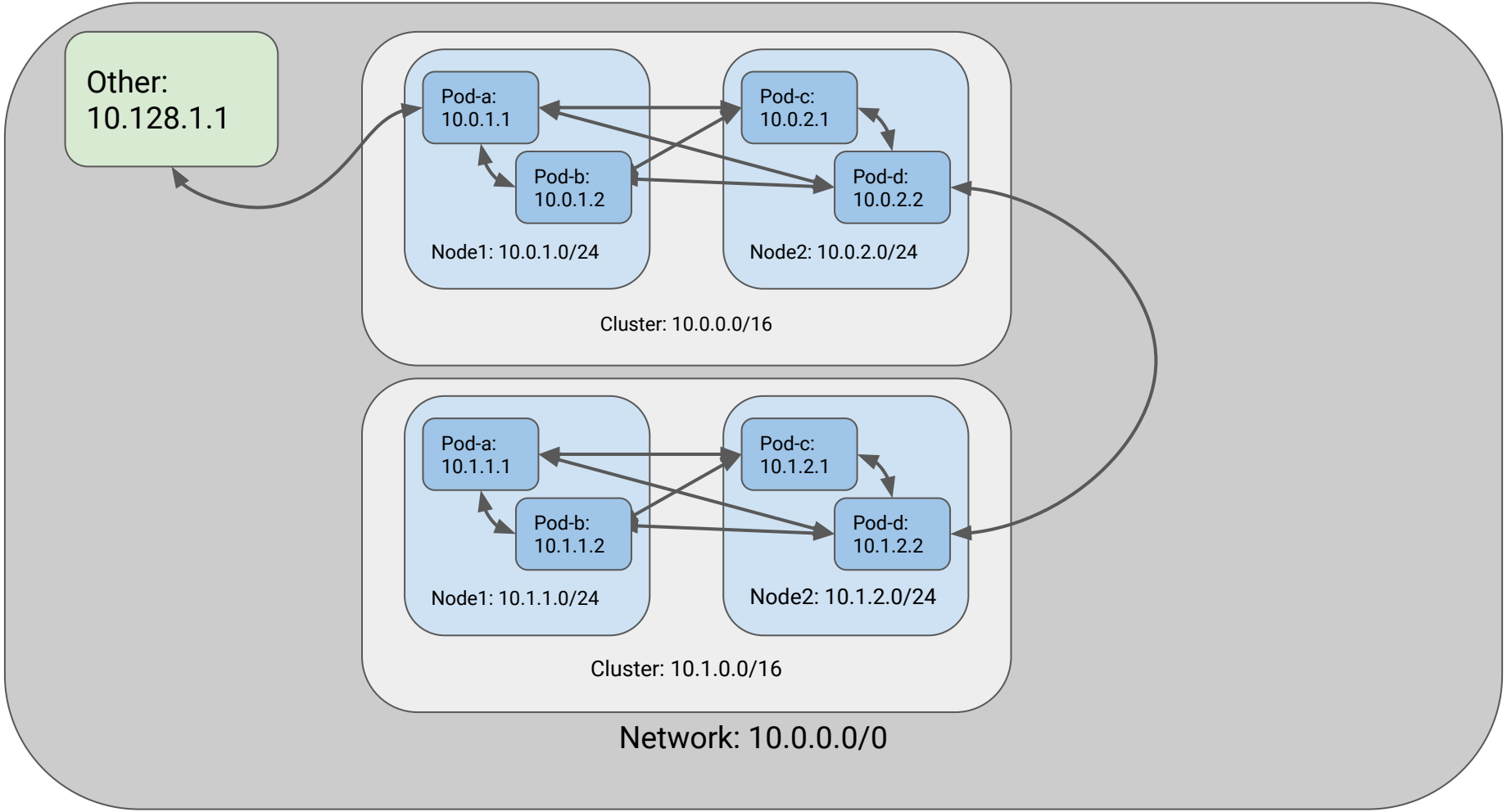

Other:
10.128.1.1
?
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/0

Multi-cluster makes it even more confusing


Other:
10.128.1.1
?
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: 10.0.1.0/24
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
?
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node1: 10.1.1.0/24
Node2: 10.1.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/0

Network models (not exhaustive)

Fully-integrated (flat)


Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2: 10.1.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/0

Each node owns an IP range

Everyone in the network knows how to deal with that

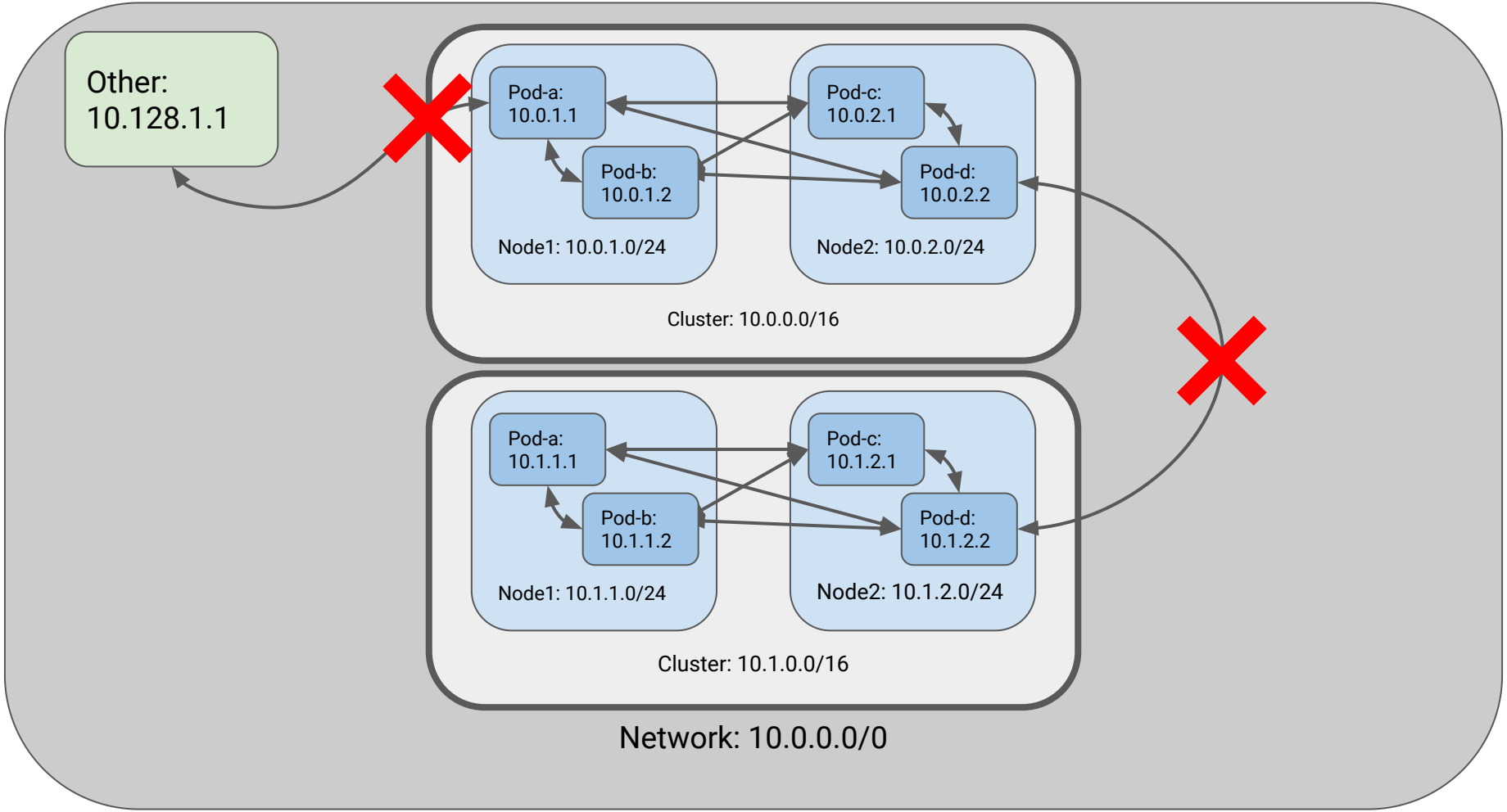
Good when:

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

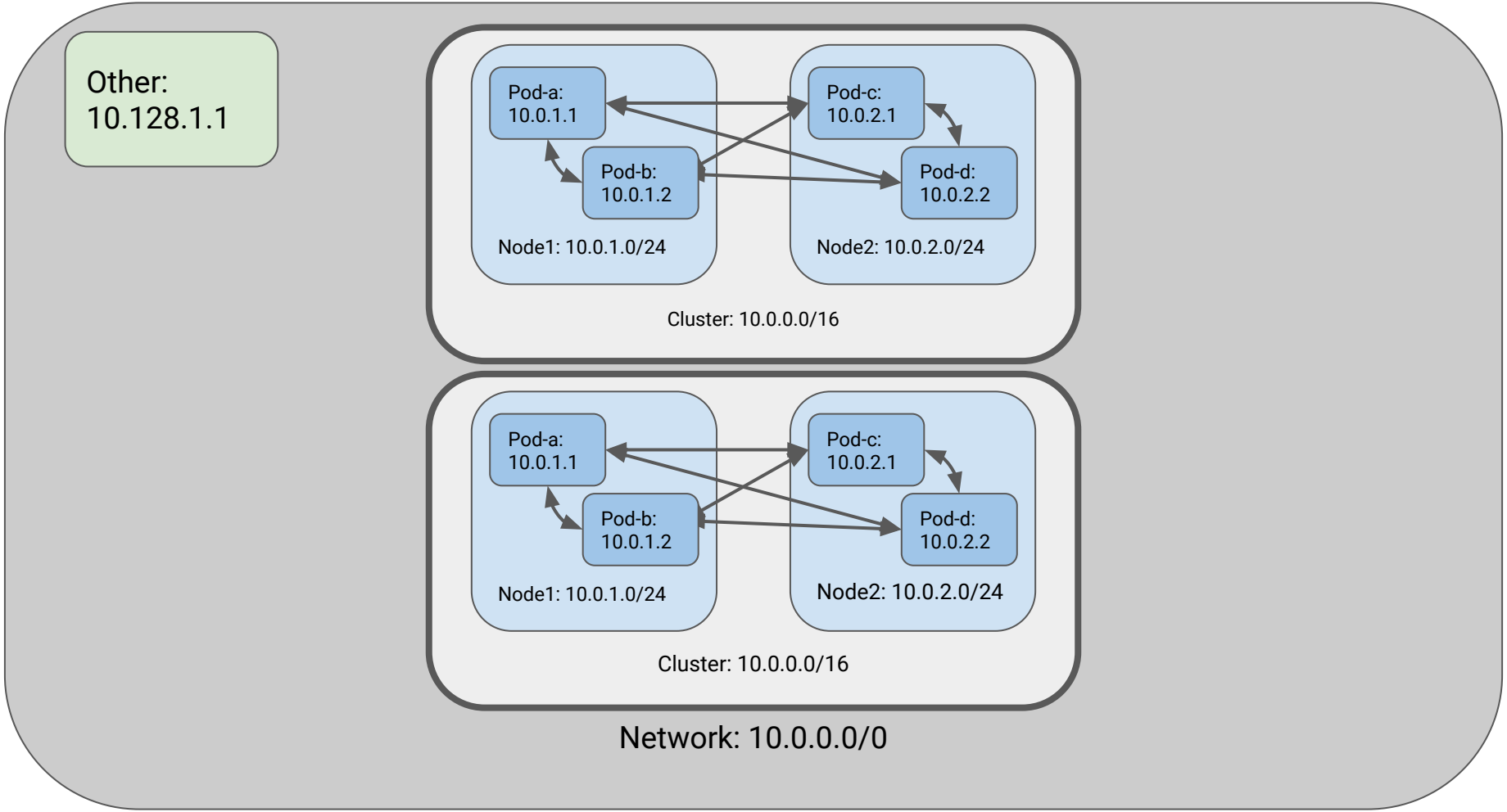
Bad when:

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

Fully-isolated


Other:
10.128.1.1
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2: 10.1.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/0

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


Other:
10.128.1.1
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: 10.0.1.0/24
Node2: 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: 10.0.1.0/24
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
Network: 10.0.0.0/0

No connectivity from inside to outside or vice-versa!

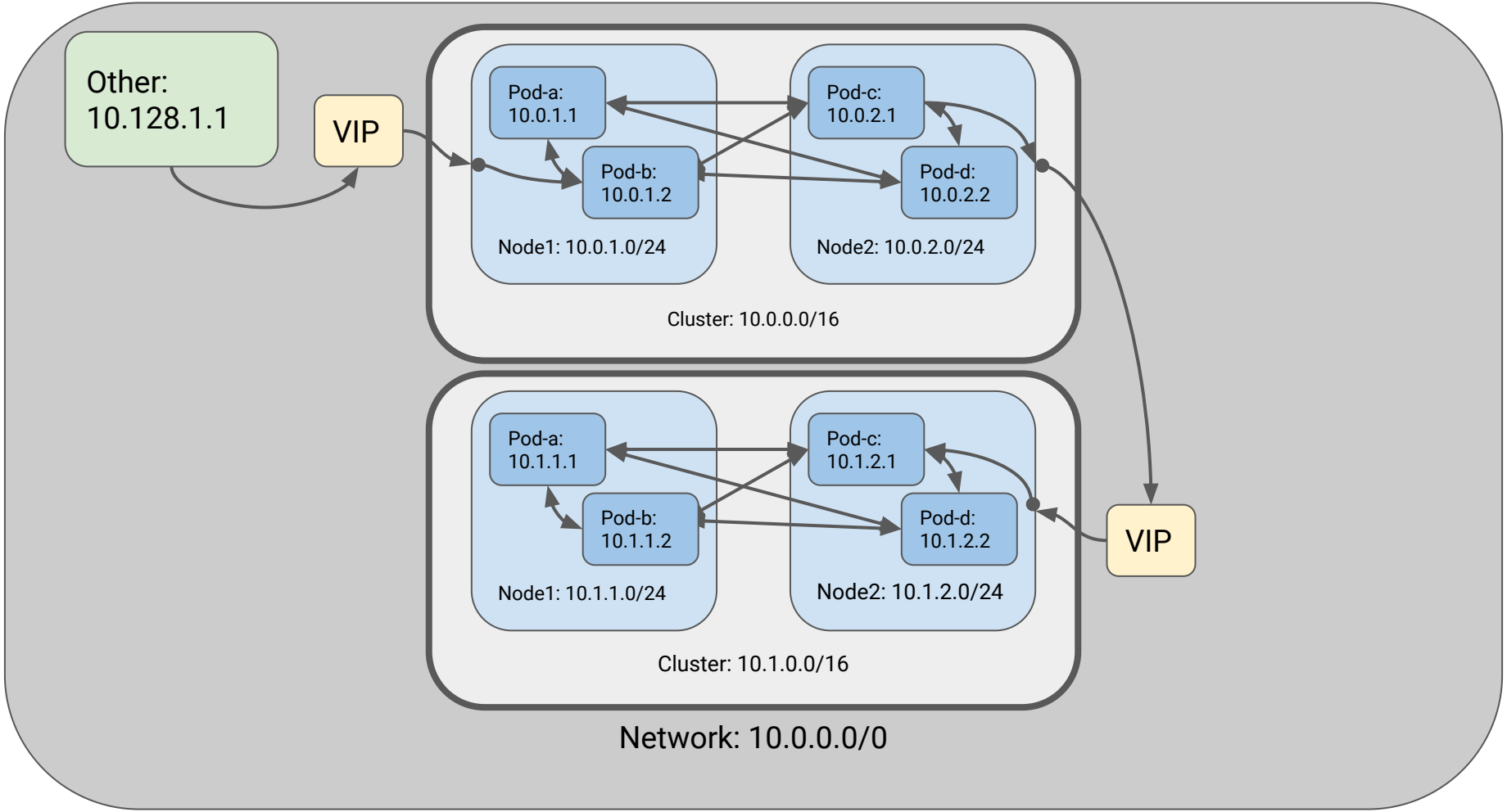
Good when:

- Don’t need integration
- IP space is scarce / fragmented
- Network is not programmable / dynamic
- Want high security

Bad when:

- Need communication across a cluster-edge

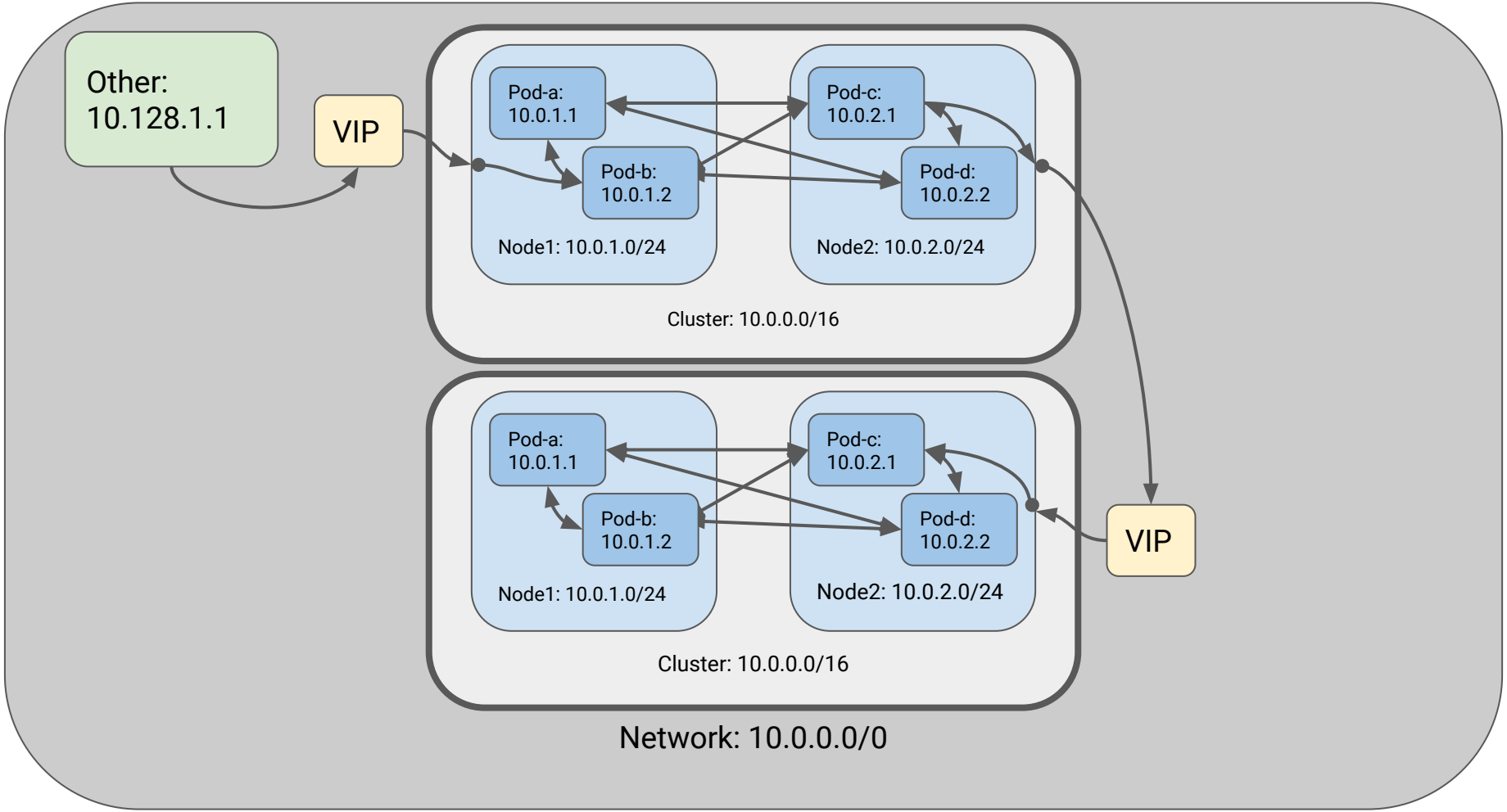
Island mode


Other:
10.128.1.1
VIP
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2: 10.1.2.0/24
Cluster: 10.1.0.0/16
VIP
Network: 10.0.0.0/0

Ingress traffic comes thru one or more VIPs

Egress traffic uses node’s IP (masquerade)

You can re-use the Pod IPs, but node IPs come from the larger network


Other:
10.128.1.1
VIP
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
VIP
Network: 10.0.0.0/0

Can be implemented as an overlay network or not

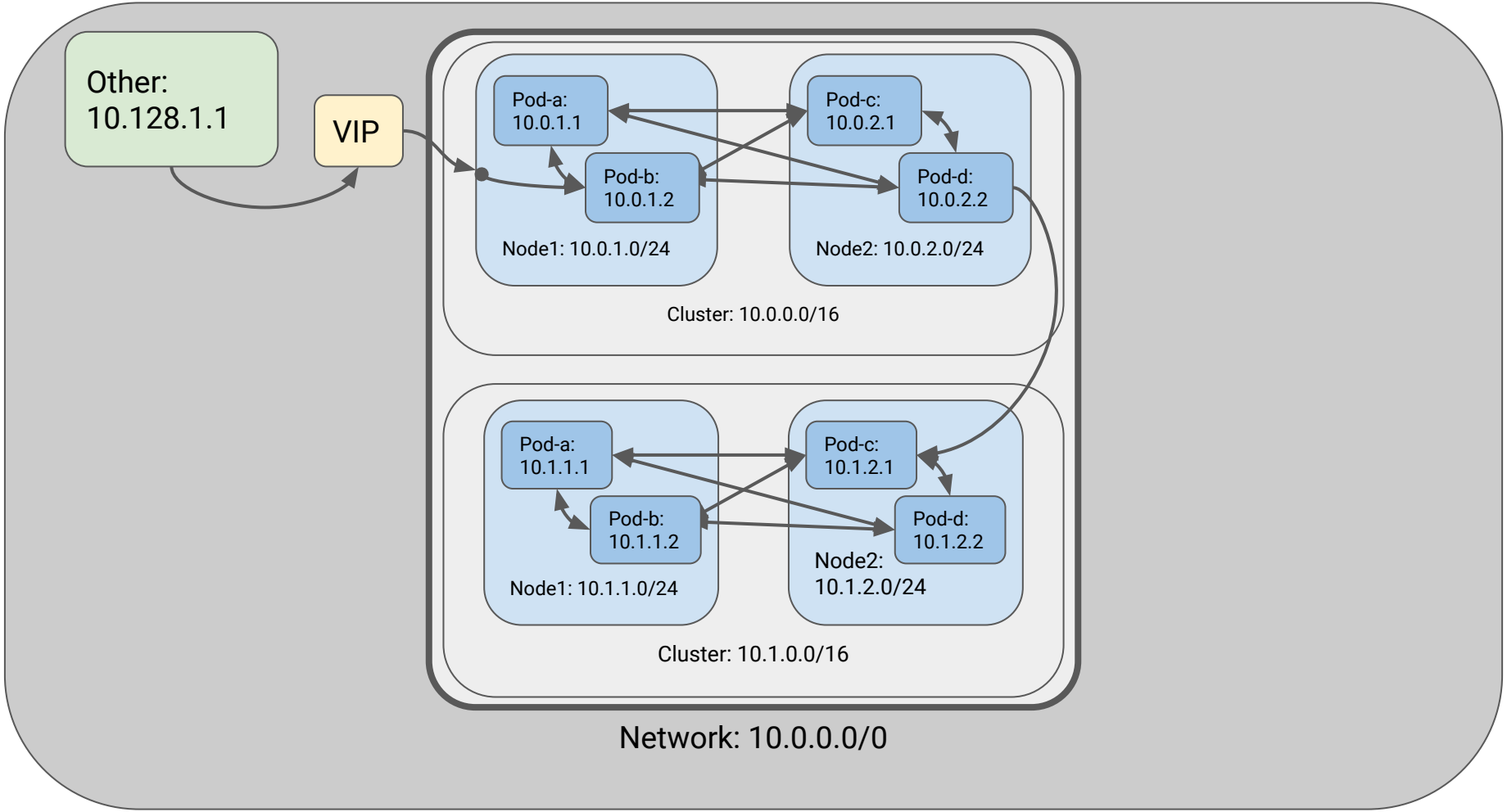
Good when:

- 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
- Rely on client IPs for firewalls
- Large number of nodes

Archipelago


Other:
10.128.1.1
VIP
Pod-a:
10.0.1.1
Pod-b:
10.0.1.2
Node1: 10.0.1.0/24
Pod-c:
10.0.2.1
Pod-d:
10.0.2.2
Node2: 10.0.2.0/24
Cluster: 10.0.0.0/16
Pod-a:
10.1.1.1
Pod-b:
10.1.1.2
Node1: 10.1.1.0/24
Pod-c:
10.1.2.1
Pod-d:
10.1.2.2
Node2:
10.1.2.0/24
Cluster: 10.1.0.0/16
Network: 10.0.0.0/0

Flat within the archipelago

Can be implemented as an overlay network or not

Good when:

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

There is no “right answer”

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