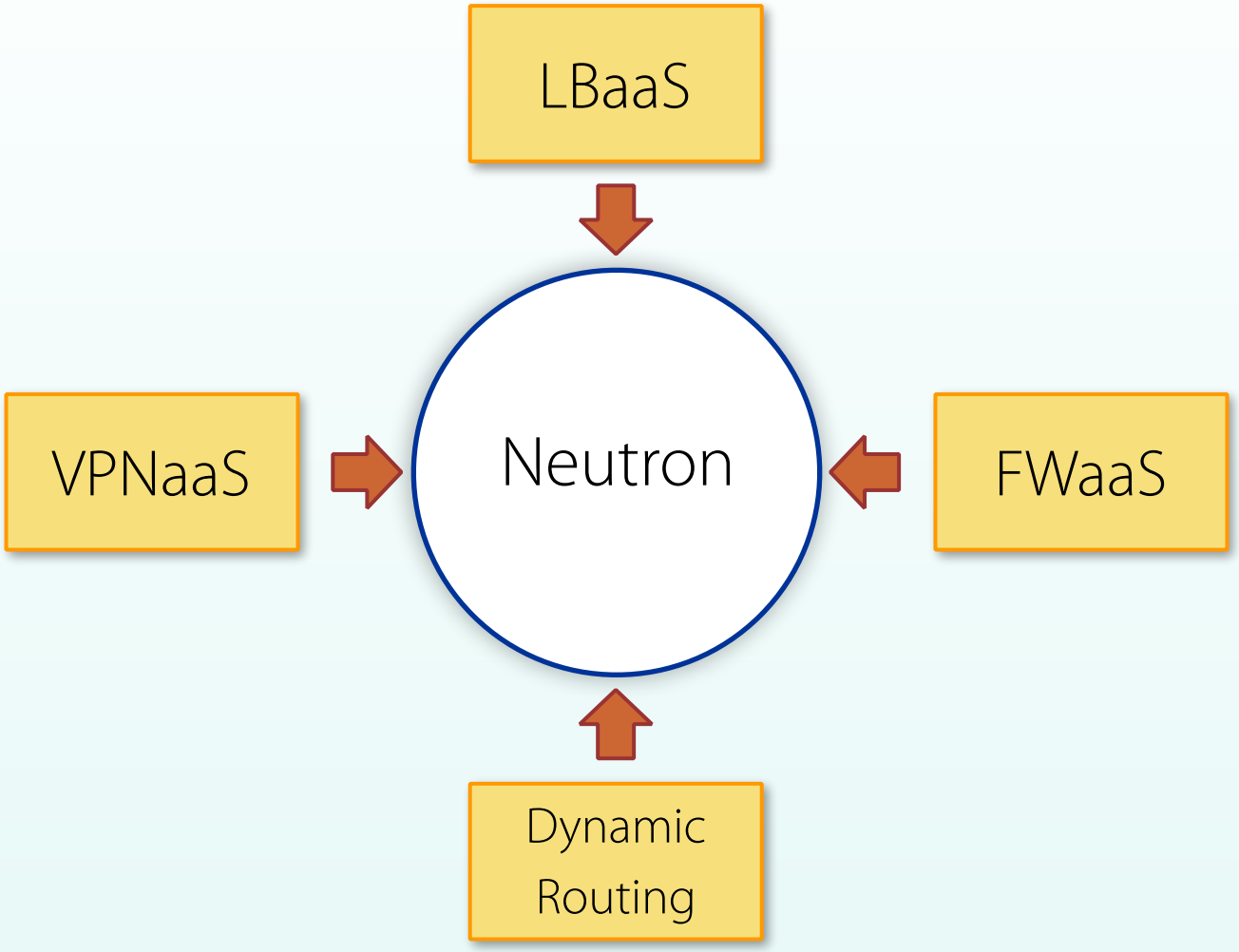
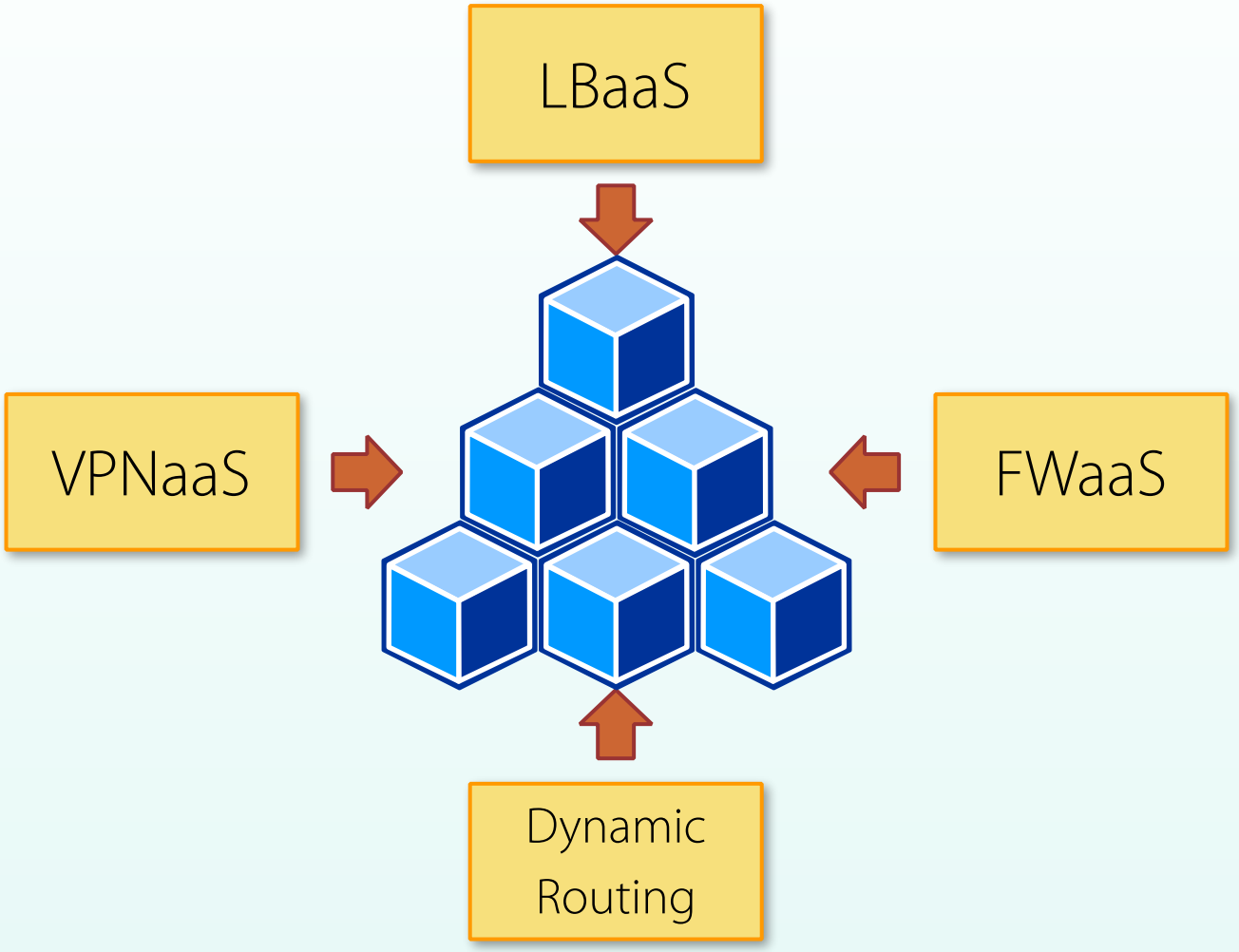
Containerizing Network Services

Alex Bikfalvi · Xavier León

Network Services



Why Containers?



Why Containers?

Similar lifecycle

Virtualizing networks functions requires lightweight isolation

Scalability

Scale-out according to the compute resources

Resiliency

Container health detection and fail-over

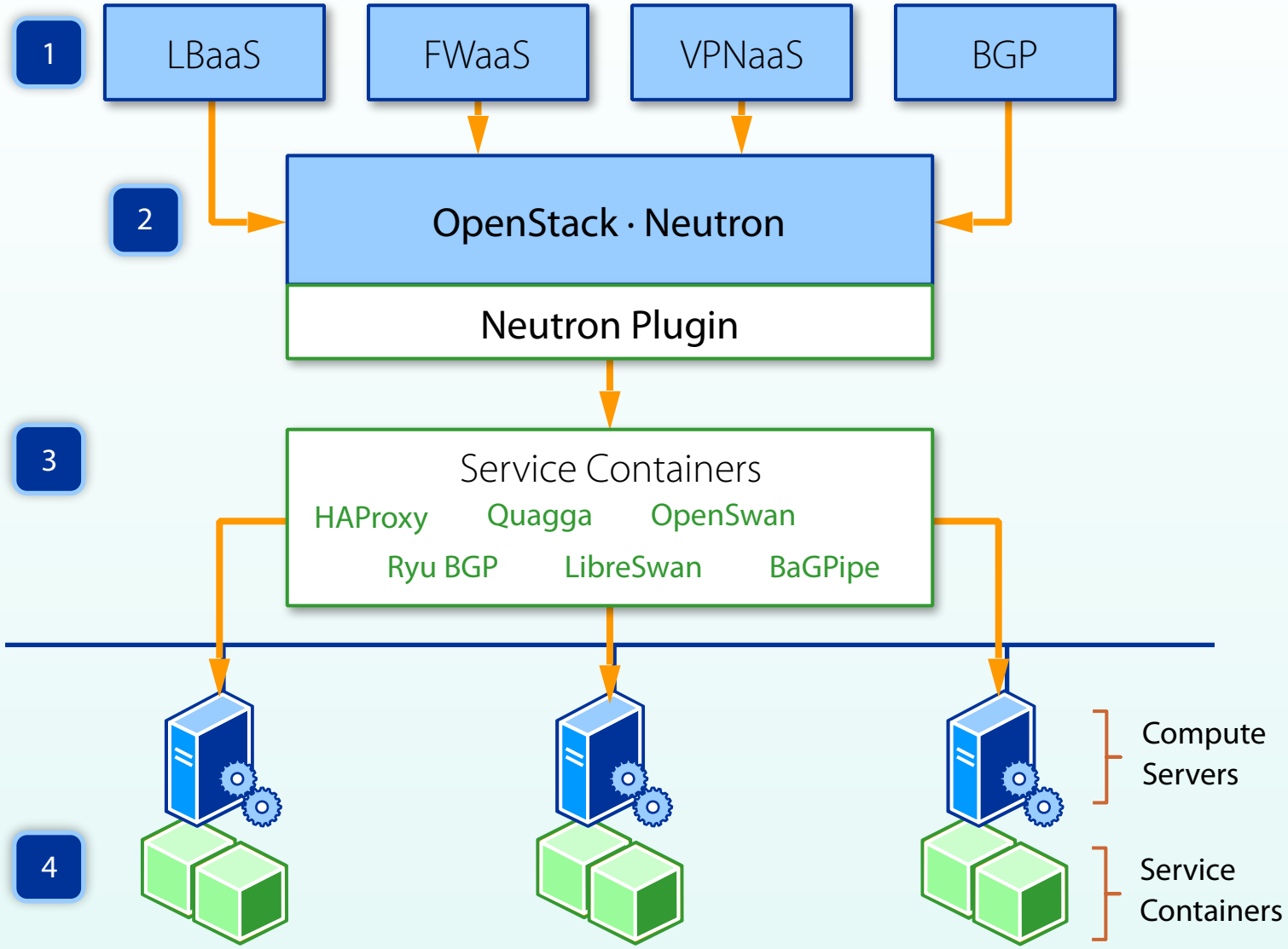
Multi-vendor or project

Alternative solutions can be leveraged side-by-side

Management

Allow operators to adjust container workload across hardware infrastructure

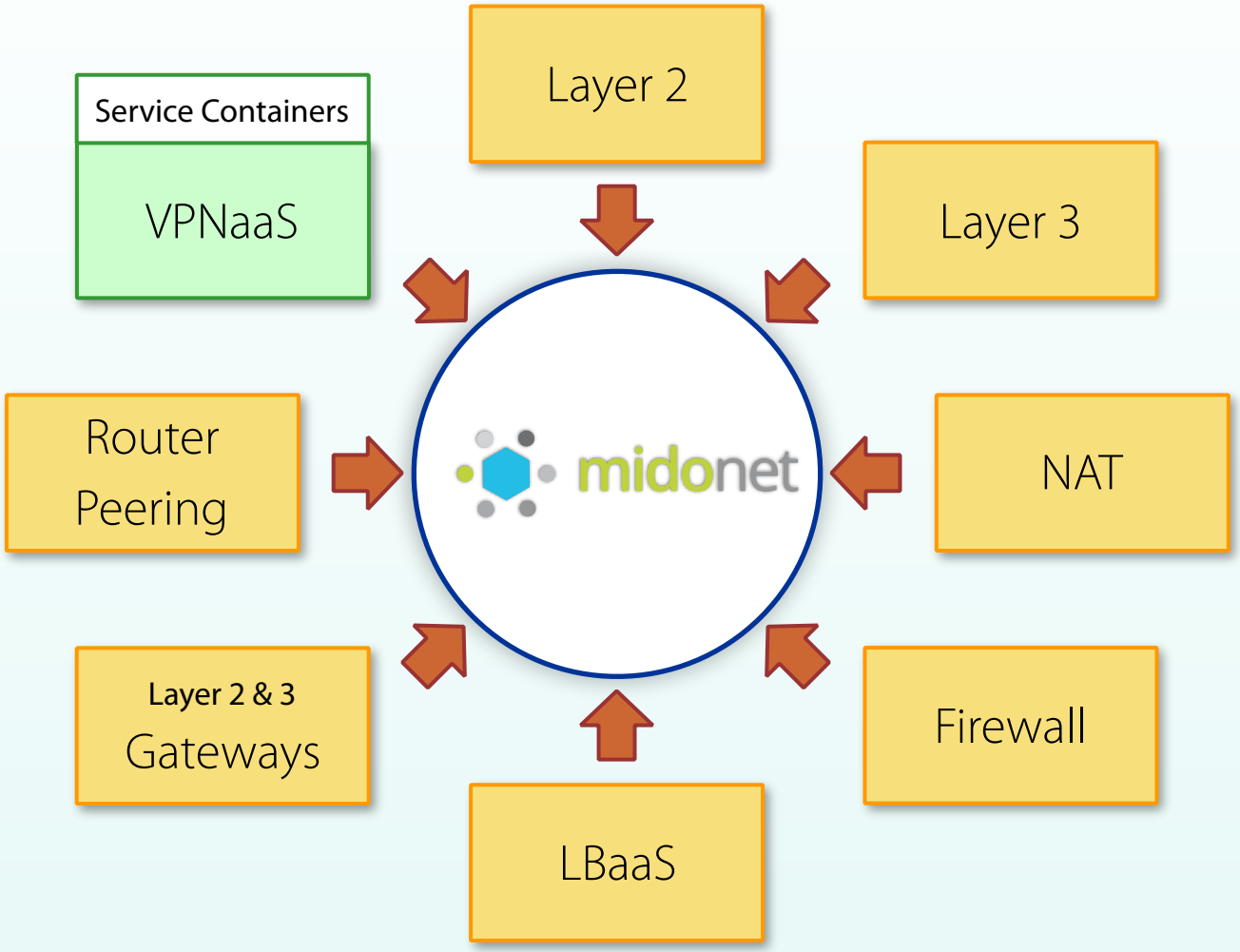
Service Containers



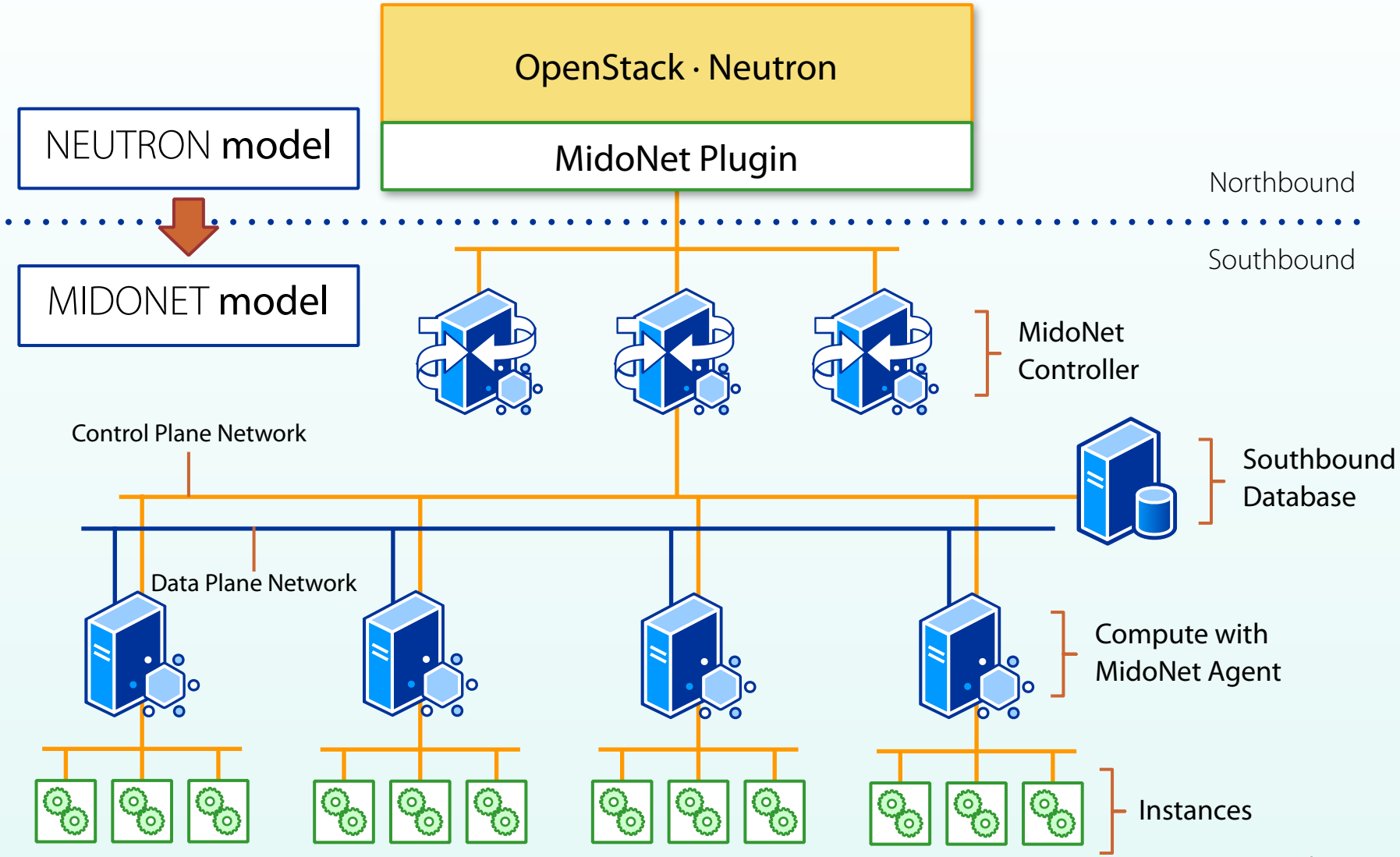
Key Requirements

1. **Scalability**
   Containers scale-out with the number of available compute nodes
2. **High Availability**
   Seamless failover on container or compute failure
3. **Container Health**
   Report the running status of the network service software
4. **Container Migration**
   Cloud operator tools to manage network service containers
5. **Scheduling Policies**
   Container affinity, host selection and fate-sharing

Containers in MidoNet

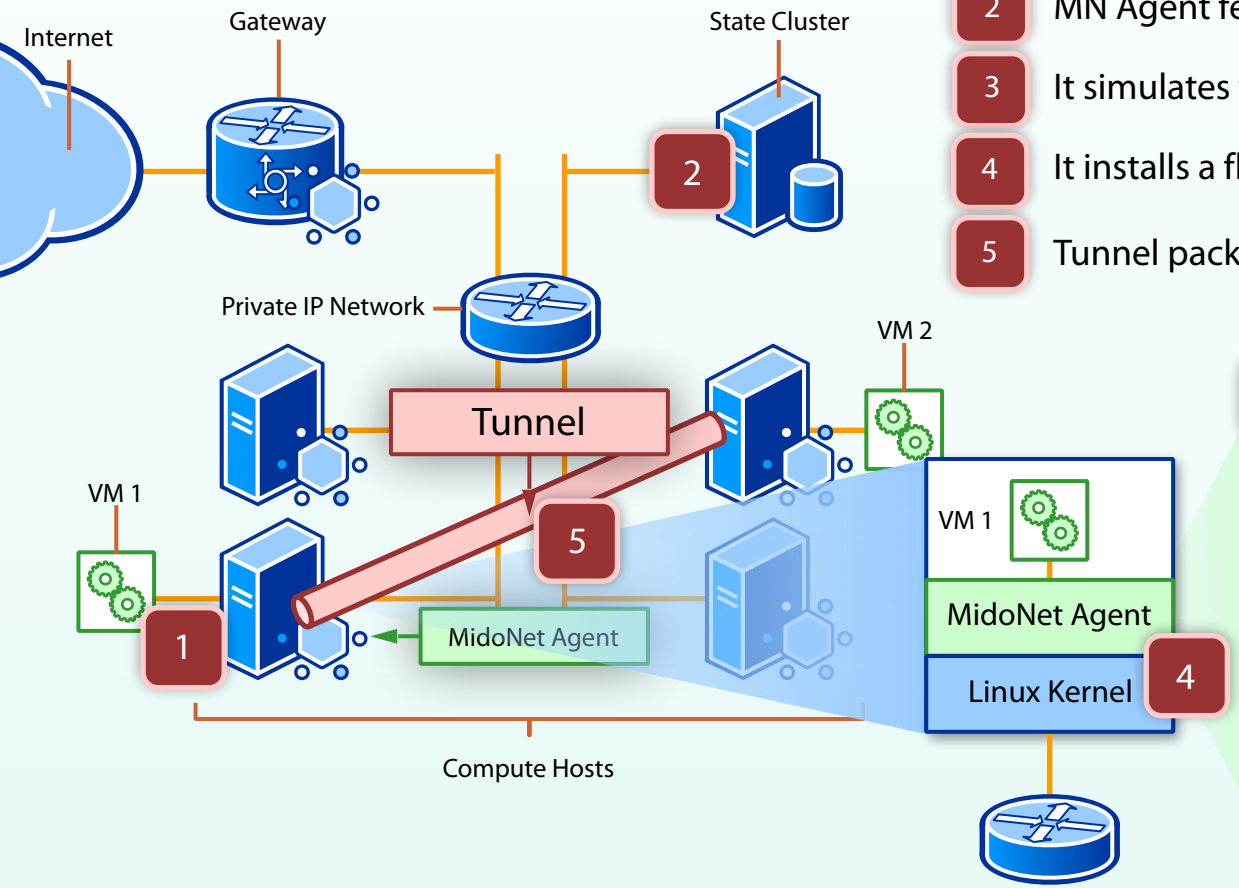


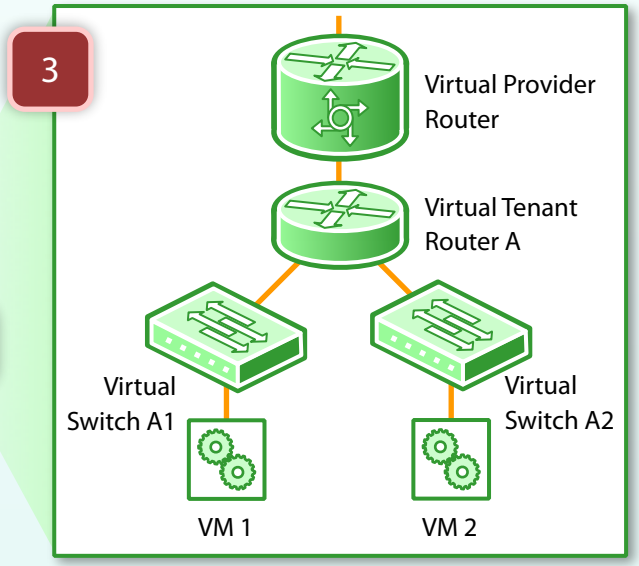
OpenStack with MidoNet



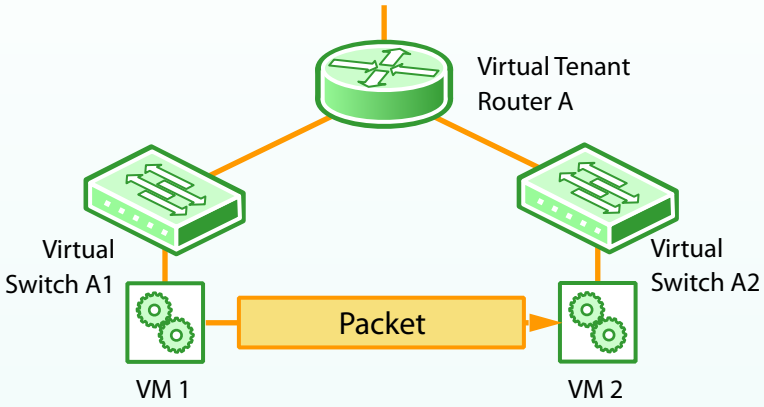
Intelligence at the Edge

1. VM 1 sends a packet through the virtual network
2. MN Agent fetches the virtual topology/state
3. It simulates the packet through the virtual network
4. It installs a flow rule in the kernel at the ingress host
5. Tunnel packets to egress host





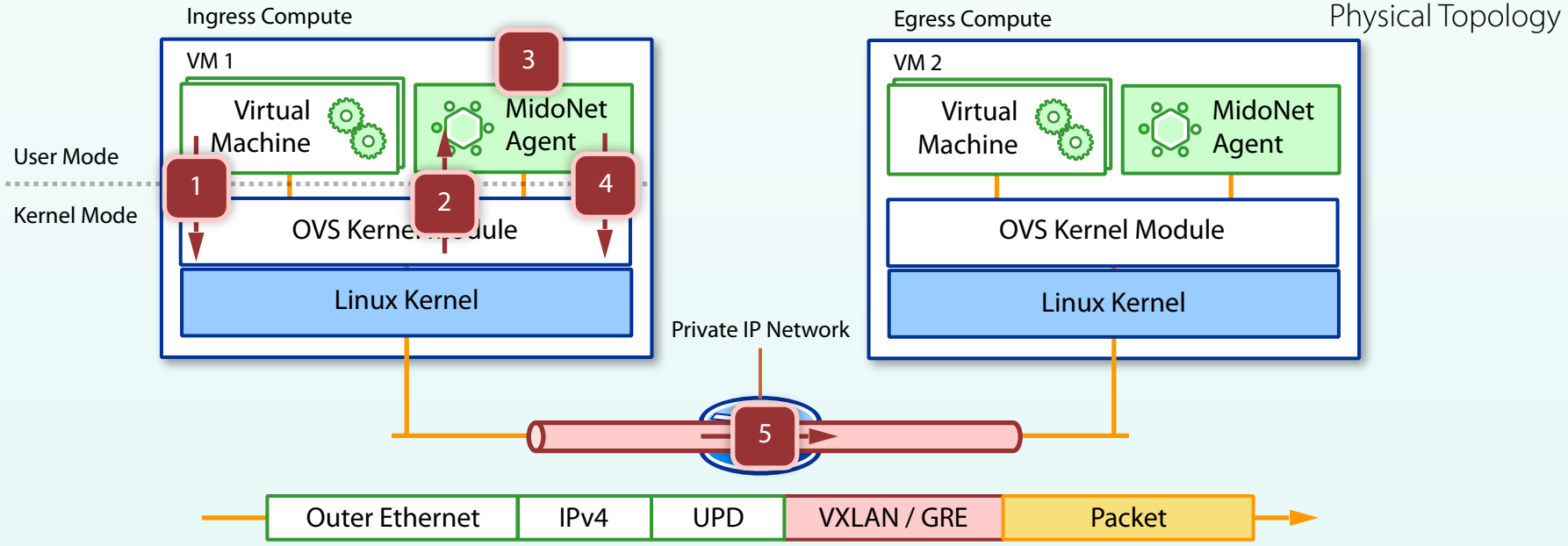
Peeking Under the Hood



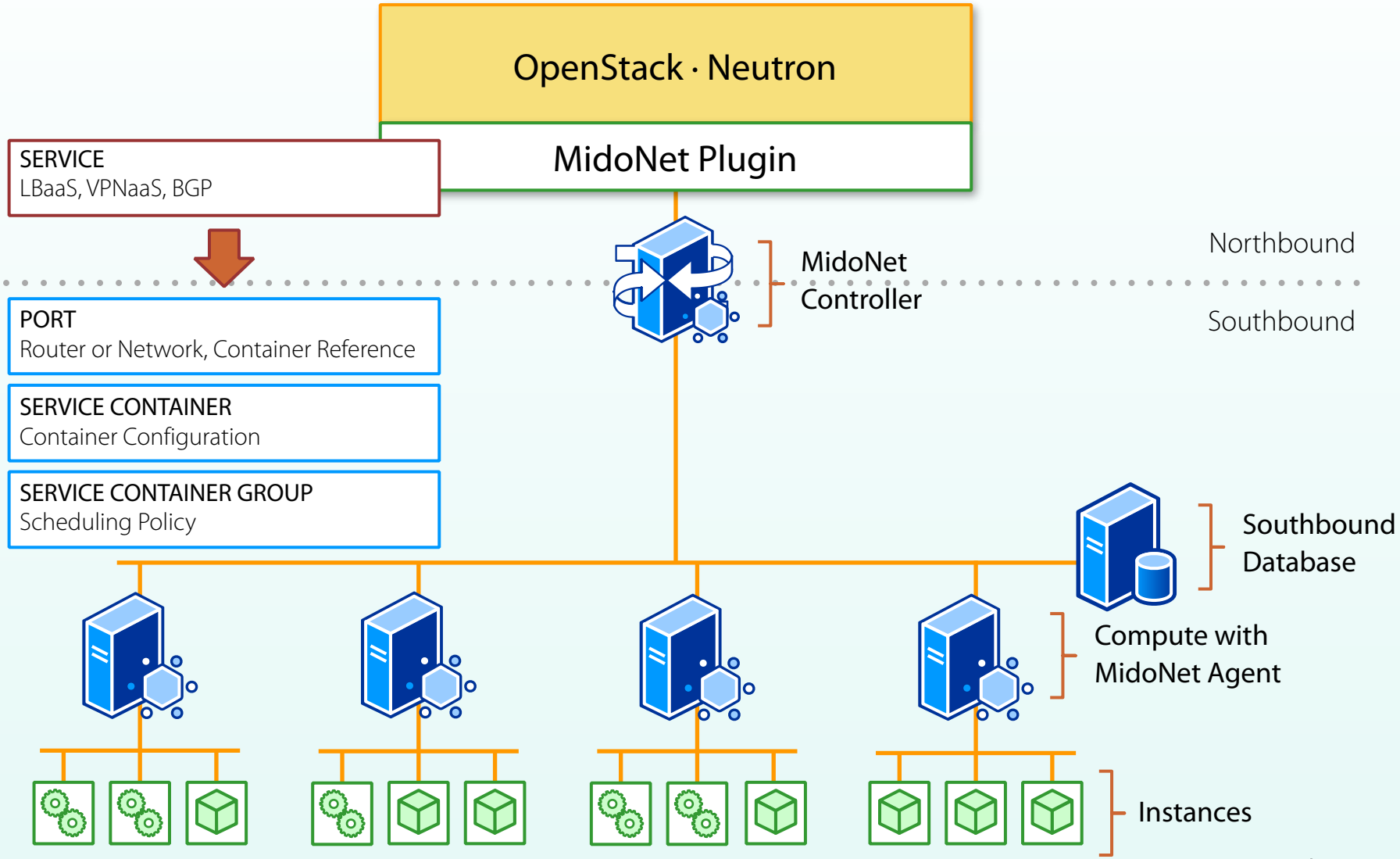
1. Packet sent by VM1 misses the OVS datapath
2. Packet sent to the MidoNet Agent via Netlink
3. The MidoNet Agent processes and simulates the packet
4. It installs a flow in the kernel at the ingress host
5. Tunnel packets to egress host

Virtual Topology

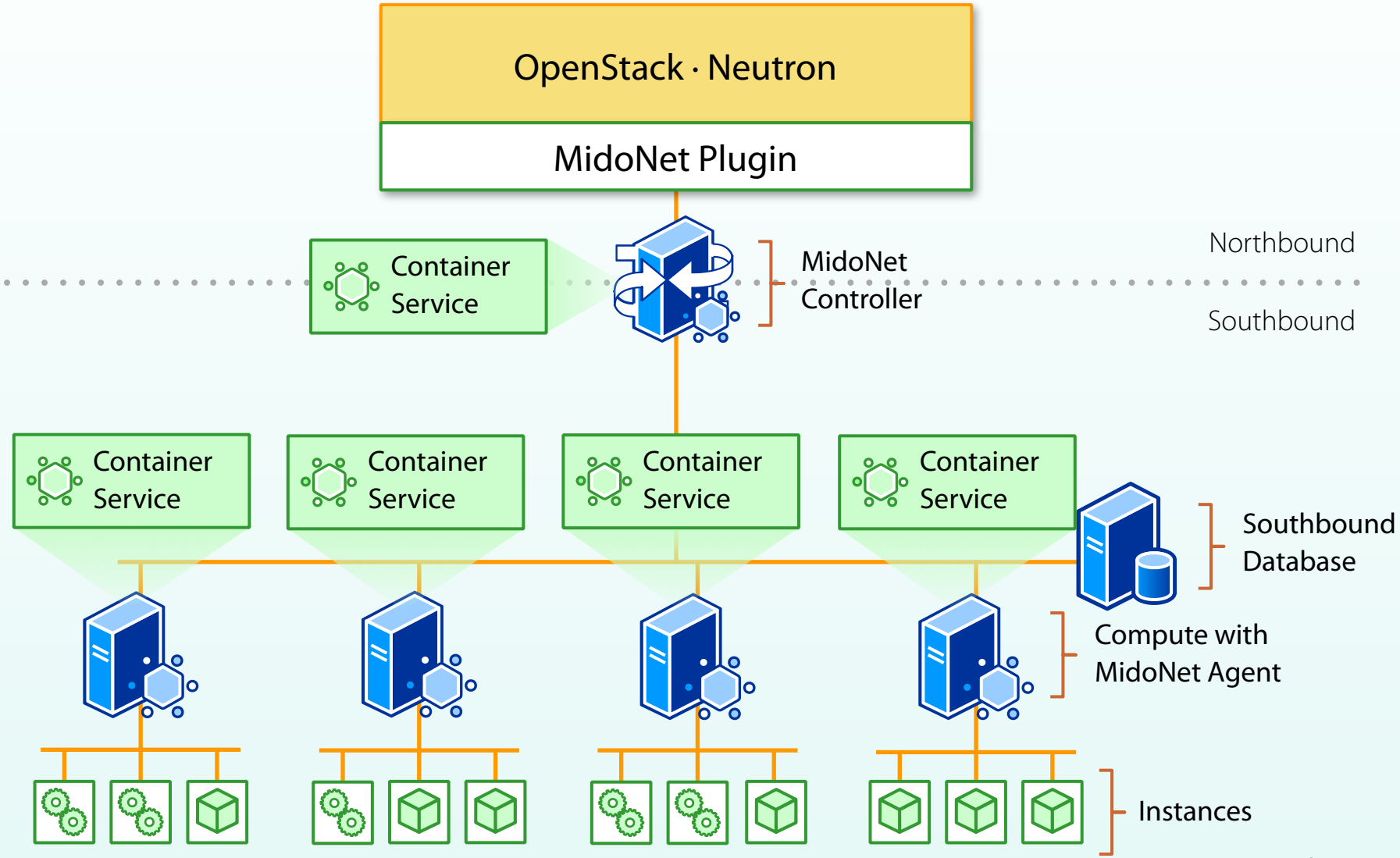
Physical Topology



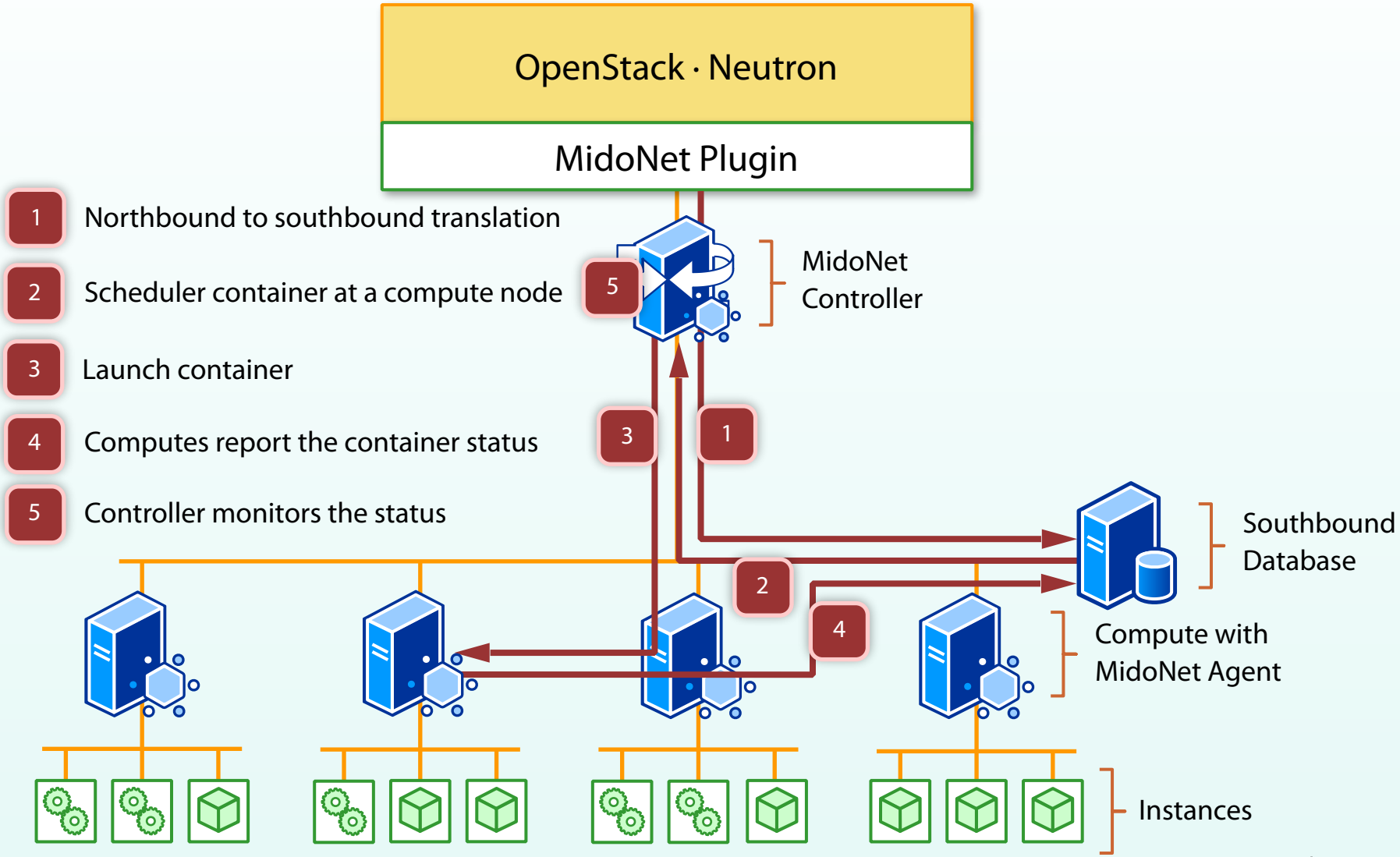
MidoNet with Containers



MidoNet with Containers



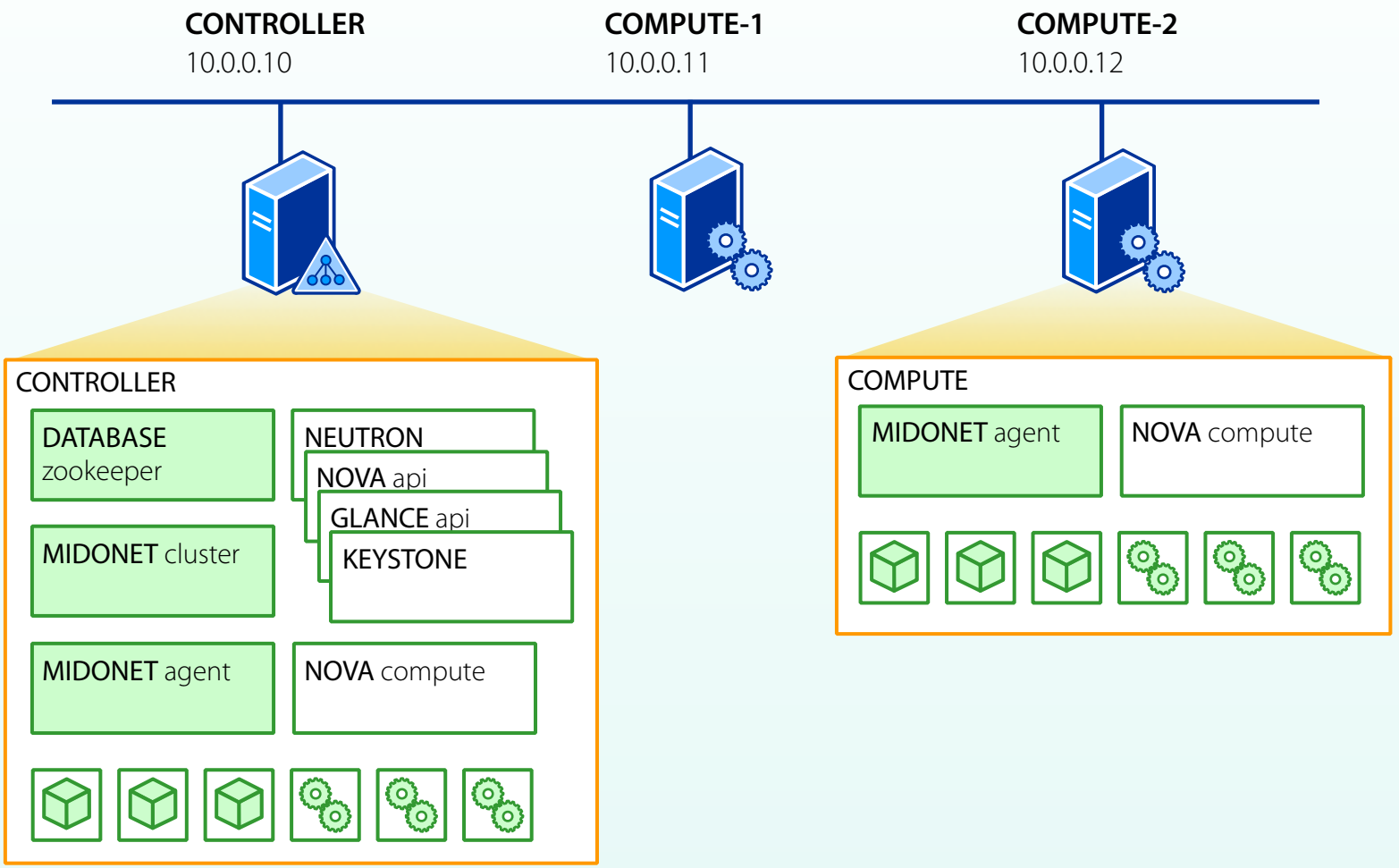
MidoNet with Containers



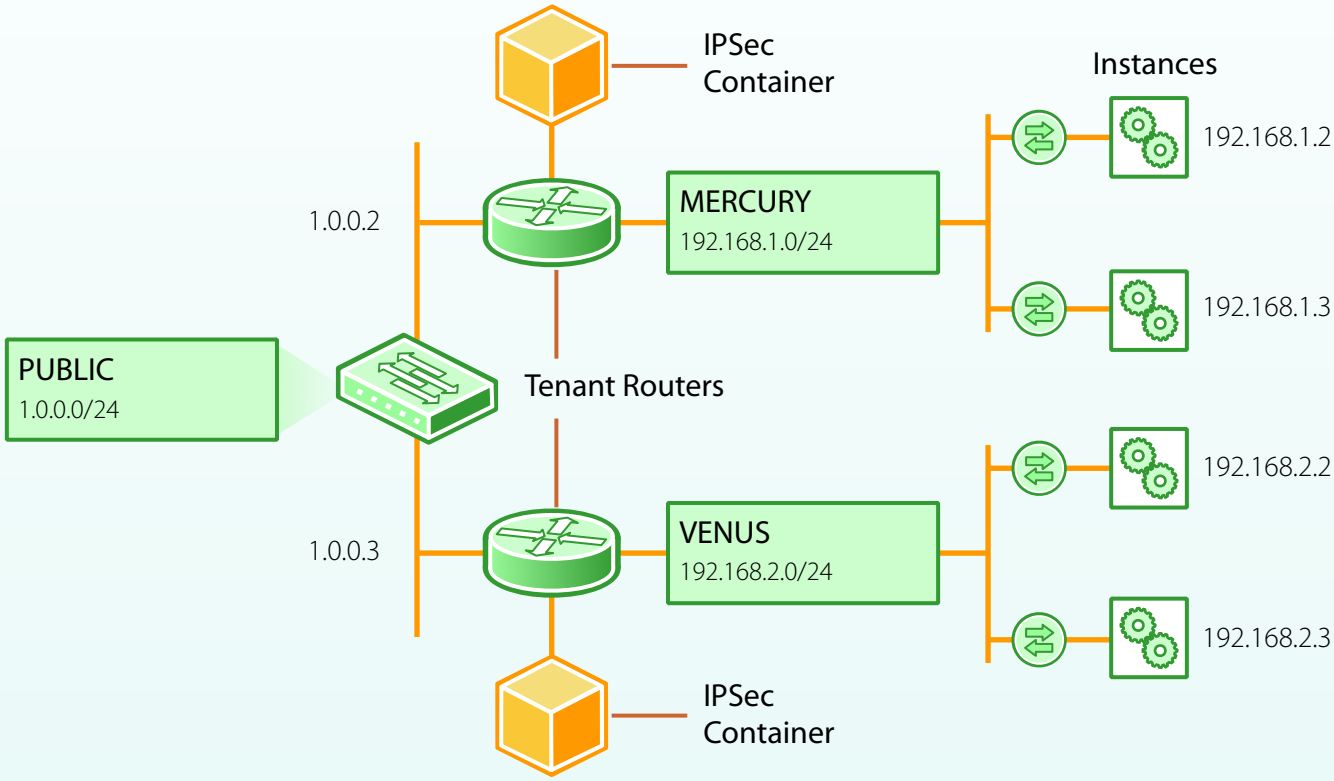
Live Demo

VPNaaS with Service Containers

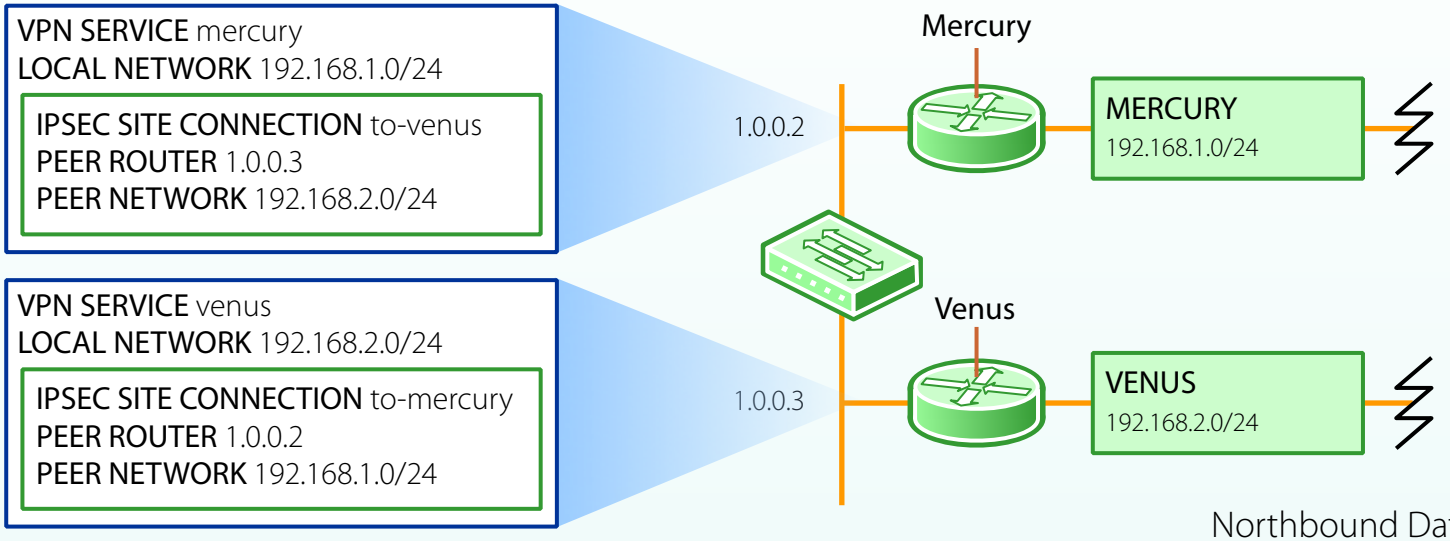
Physical Layer



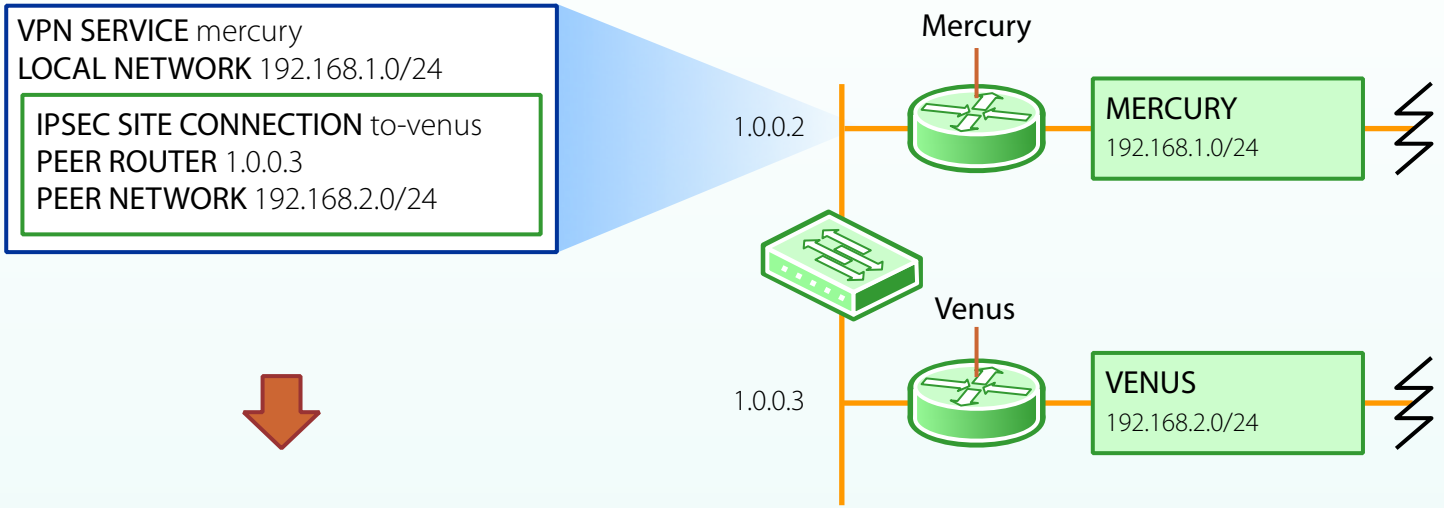
Virtual Topology



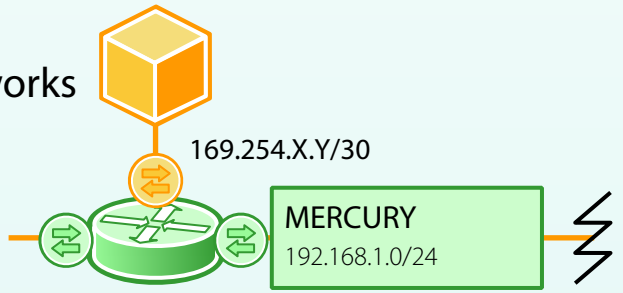
Service Translation



Service Translation



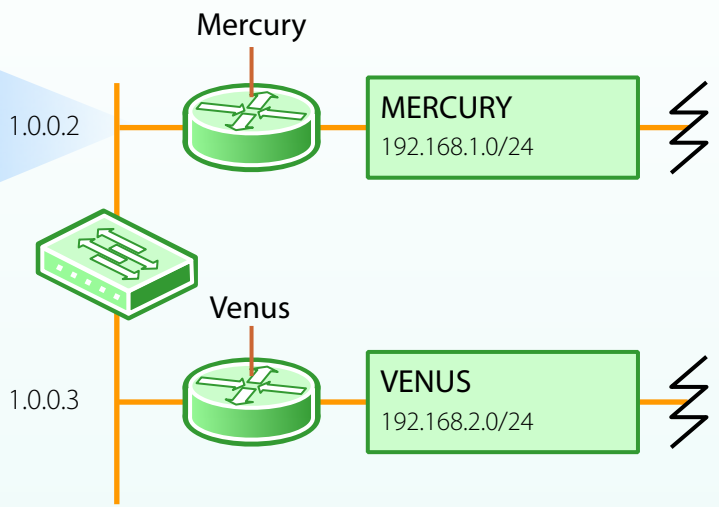
1. **Router port for the service container**
   Includes routes that forward packets to the container
2. **Redirect rules matching traffic between peer networks**
   Match IPSec (protocol 50) and IKE (UDP ports 500 and 4500)
3. **Container and container group policy**
   Include container type and configuration
4. **Bind the container port to a compute host**
   Tells the compute to launch the container



Service Translation

VPN SERVICE mercury
LOCAL NETWORK 192.168.1.0/24

IPSEC SITE CONNECTION to-venus
PEER ROUTER 1.0.0.3
PEER NETWORK 192.168.2.0/24



Northbound Database

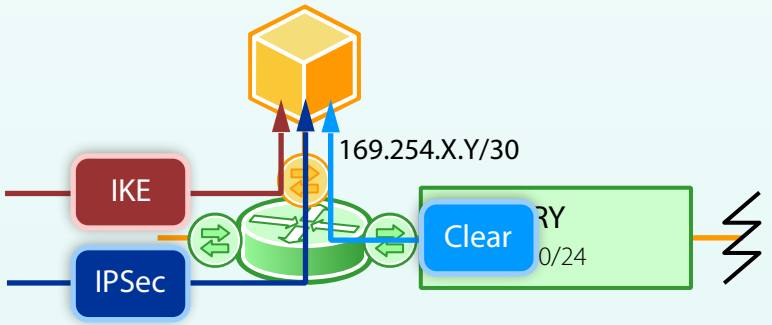
Southbound Database

ROUTE
Source 192.168.1.0/24
Destination 192.168.2.0/24

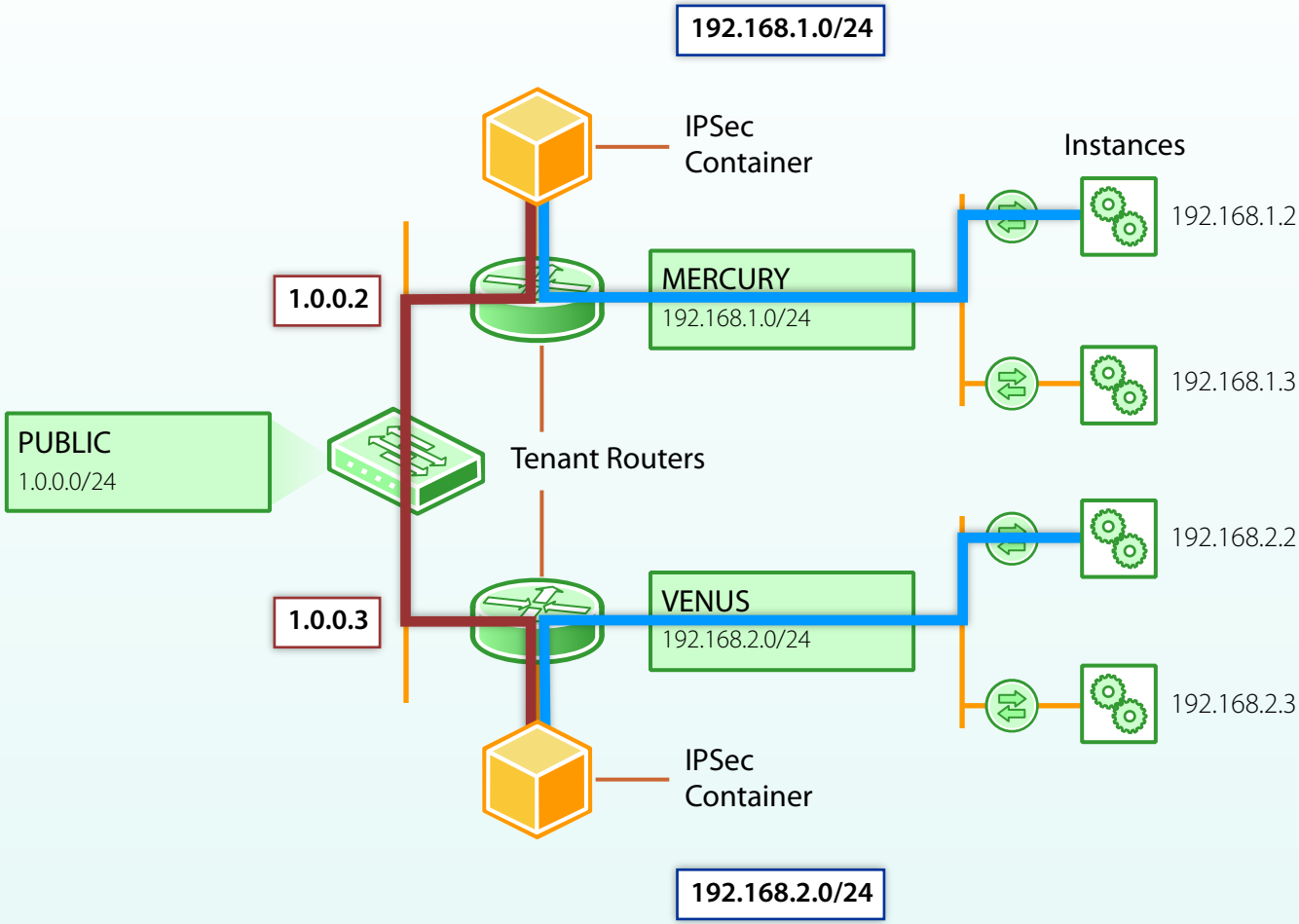
RULE REDIRECT
Protocol 17 Port 500

RULE REDIRECT
Protocol 17 Port 4500

RULE REDIRECT
Protocol 50



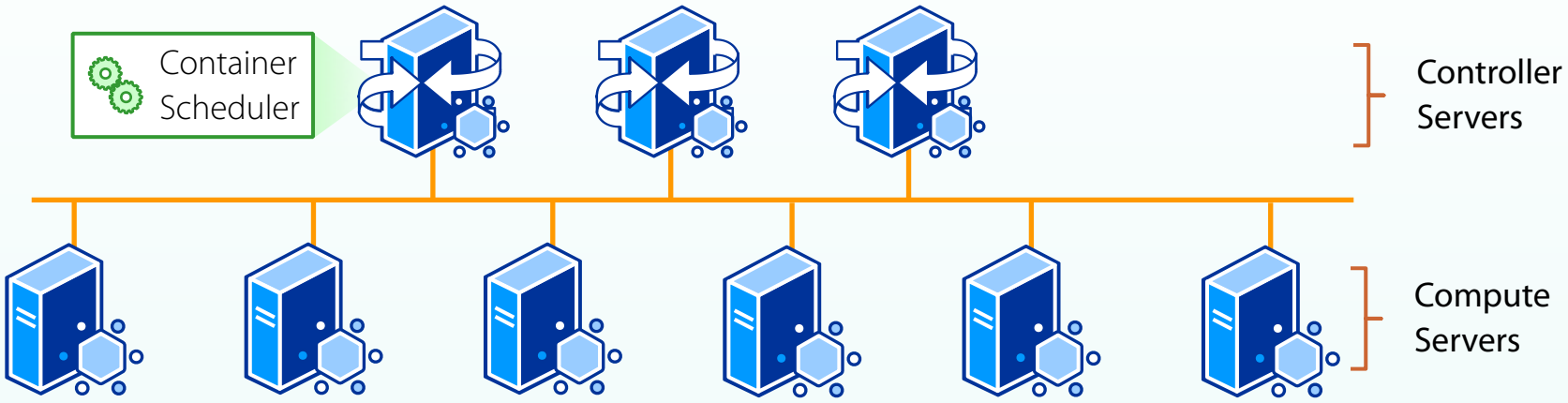
Traffic and IPSec Containers



Live Demo

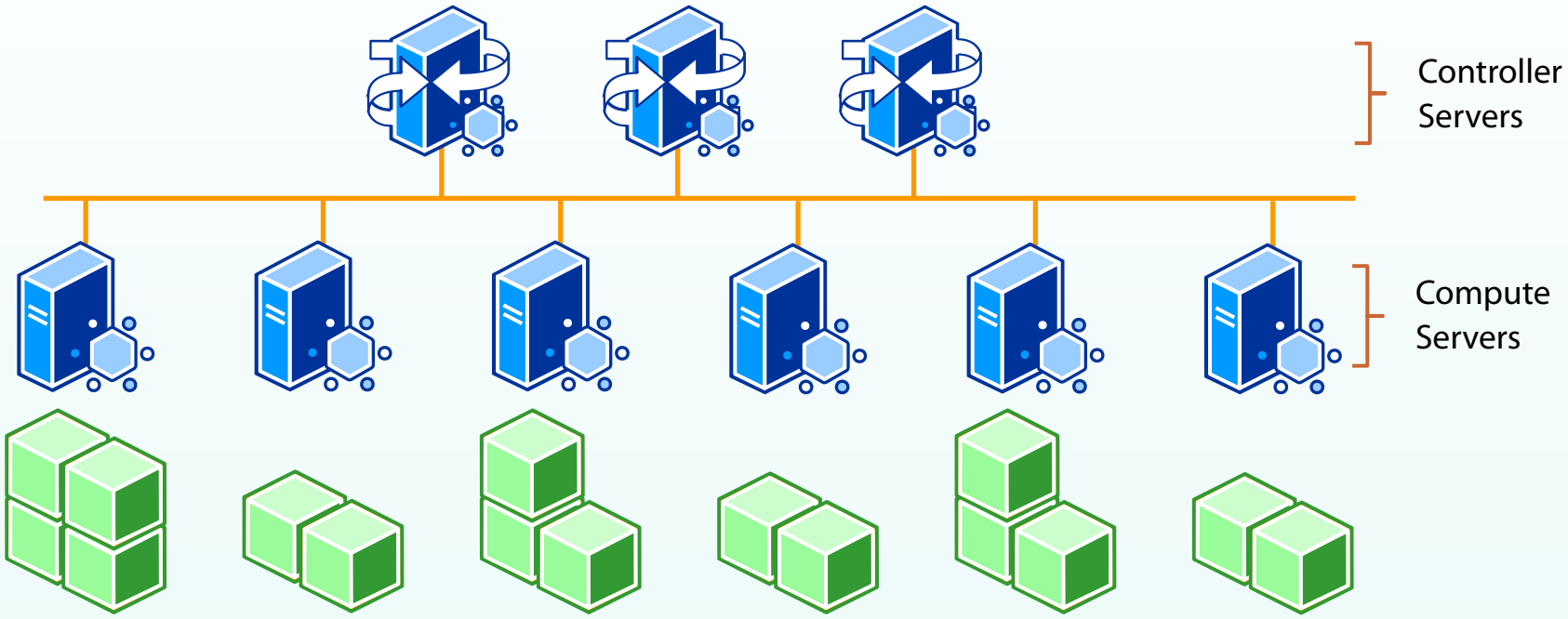
VPNaaS with Service Containers

Container Scheduling



Controller nodes coordinate in an active-passive fashion and are restart tolerant

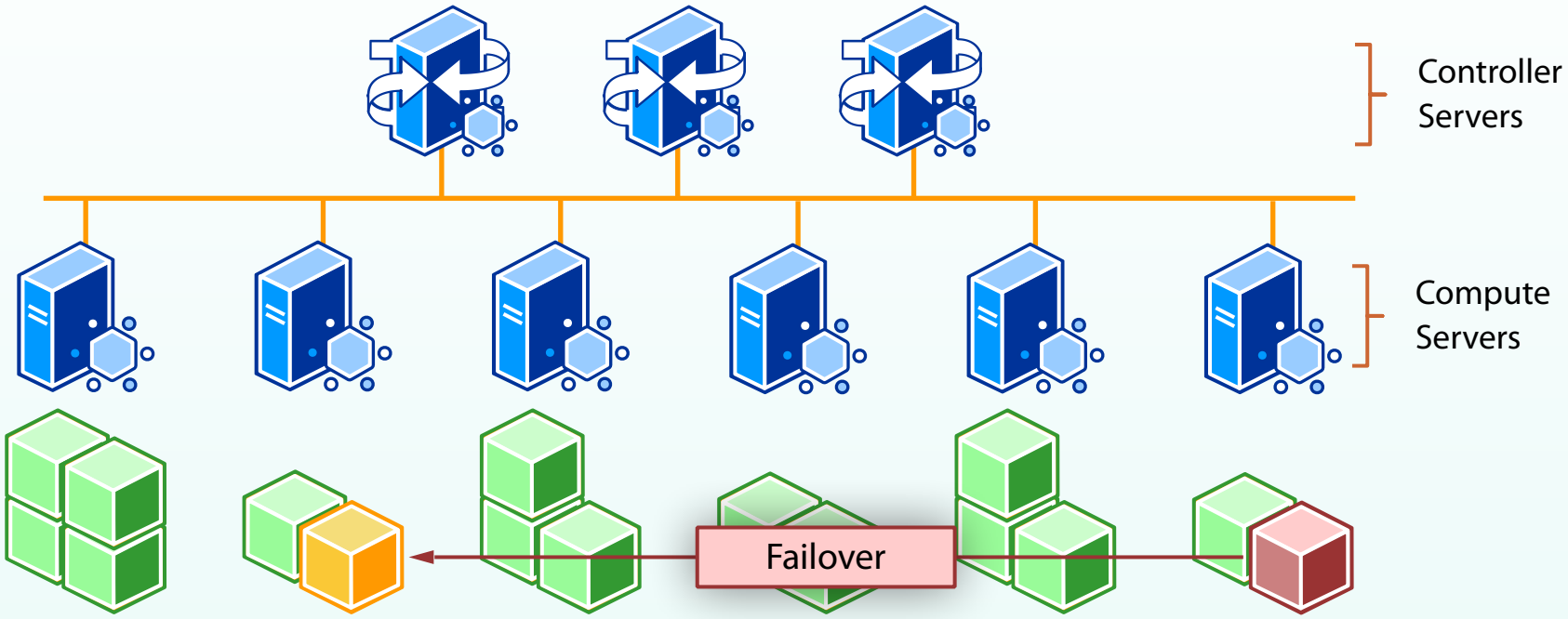
Container Scheduling



1 **Select a compute host when creating a new container**

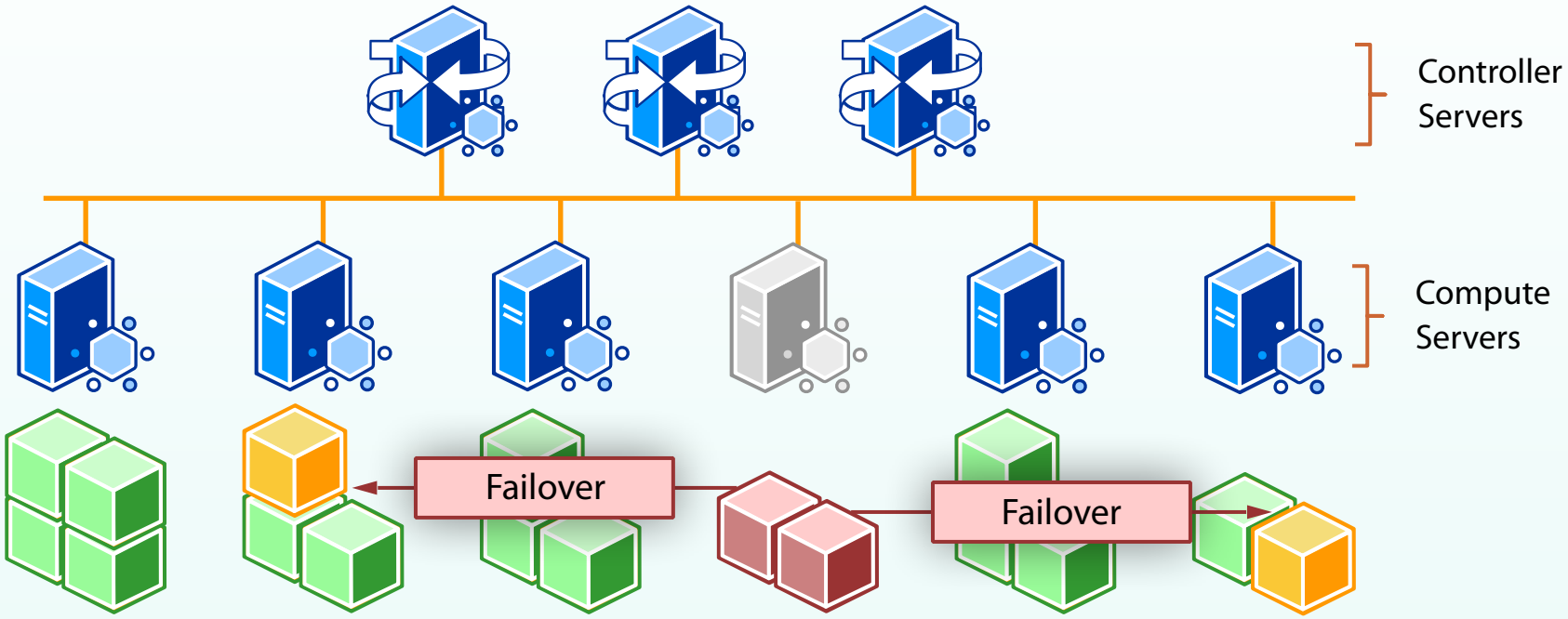
Host eligibility is determined by availability and the operator or service policy

Container Scheduling



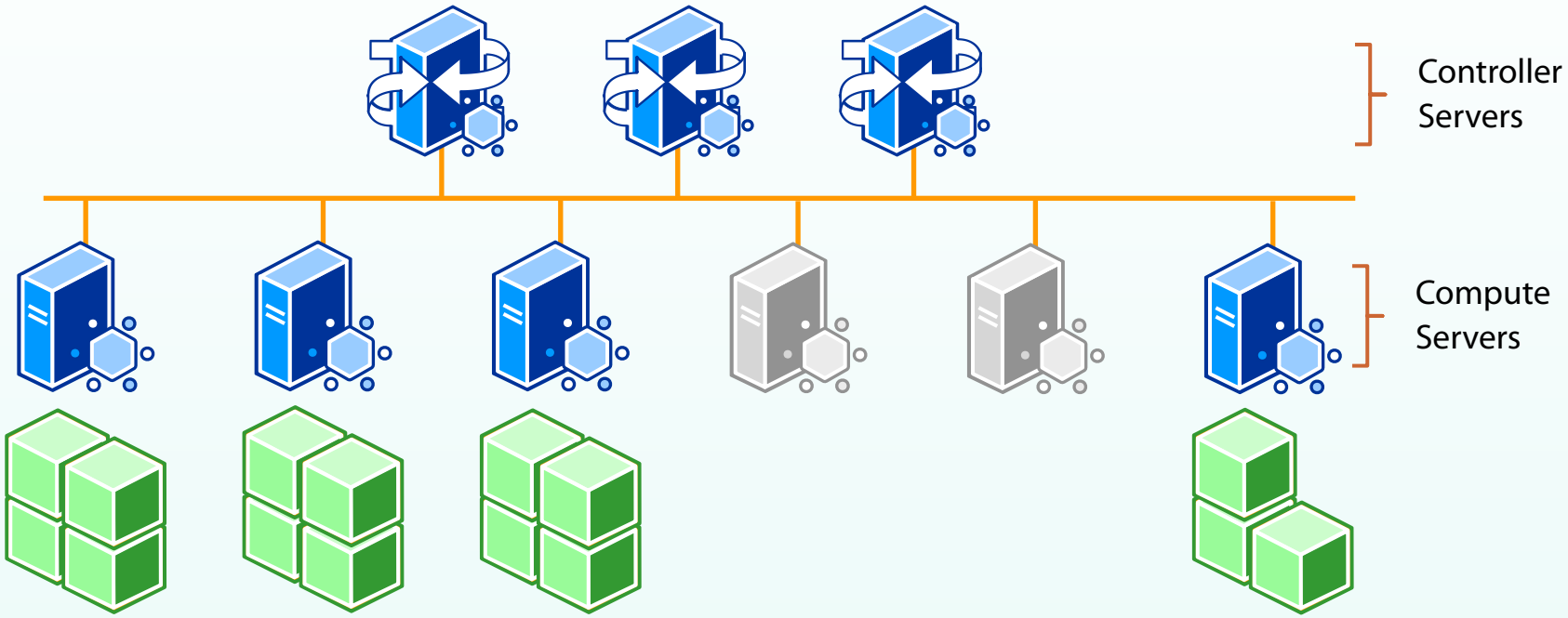
1. **Select a compute host when creating a new container**
   Host eligibility is determined by availability and the operator or service policy
2. **Monitor container health**
   Containers report their status to their supervising agent

Container Scheduling



1. **Select a compute host when creating a new container**
   Host eligibility is determined by availability and the operator or service policy
2. **Monitor container health**
   Containers report their status to their supervising agent
3. **Monitor compute host health and availability**
   Agents reports their running status to the controllers via the southbound messaging channel

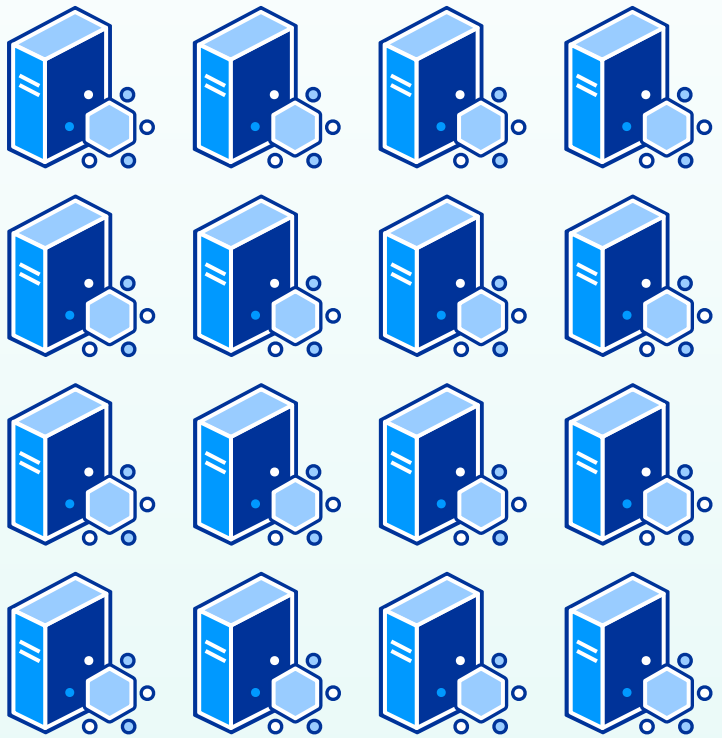
Container Scheduling



1. **Select a compute host when creating a new container**
   Host eligibility is determined by availability and the operator or service policy
2. **Monitor container health**
   Containers report their status to their supervising agent
3. **Monitor compute host health and availability**
   Agents reports their running status to the controllers via the southbound messaging channel
4. **Allow operator orchestration of containers**
   Manage scheduling via policies or manual migration

Group Scheduling Policies

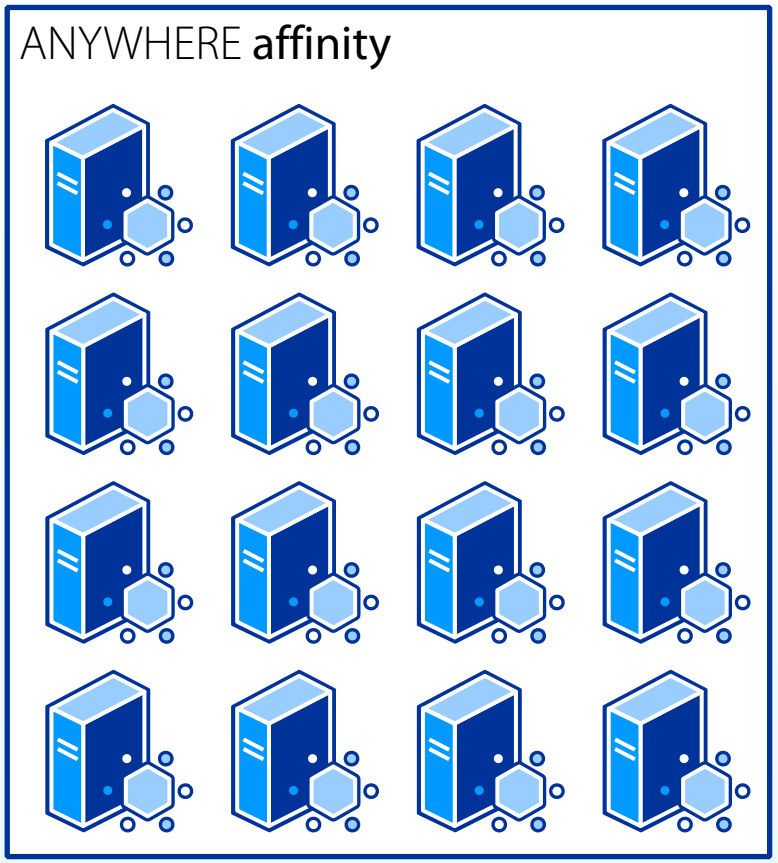
1 Affinity Policies



Define the set of computes that can host a container for a particular network service

Group Scheduling Policies

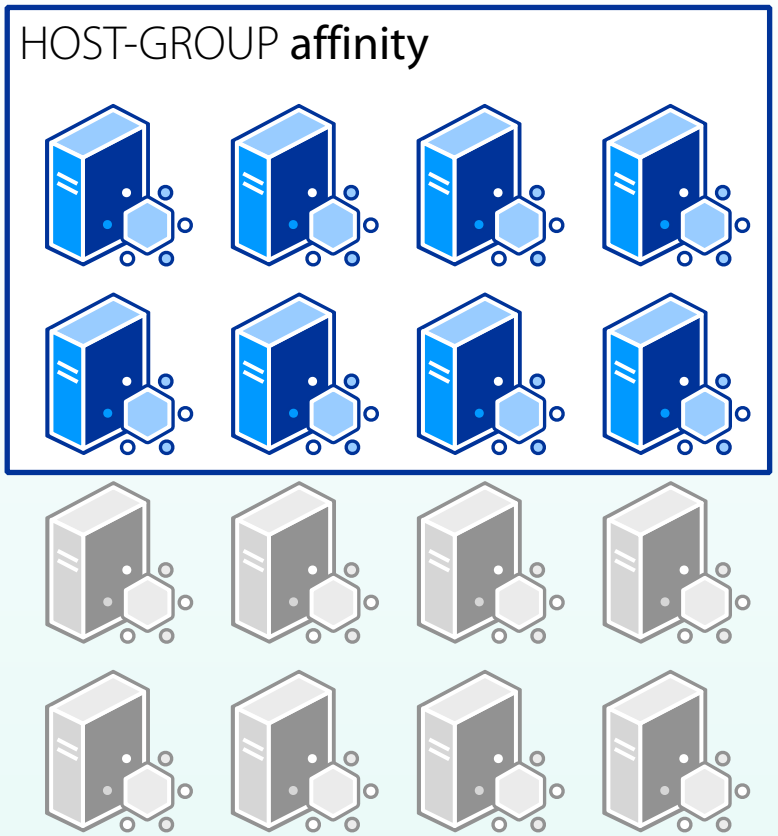
1 Affinity Policies



Define the set of computes that can host a container for a particular network service

Group Scheduling Policies

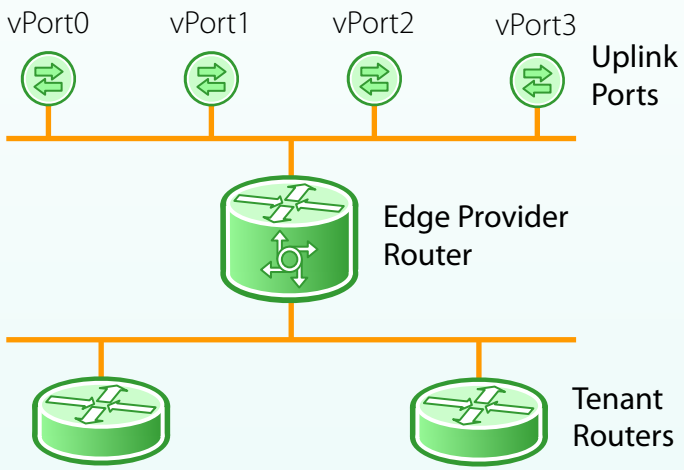
1 Affinity Policies

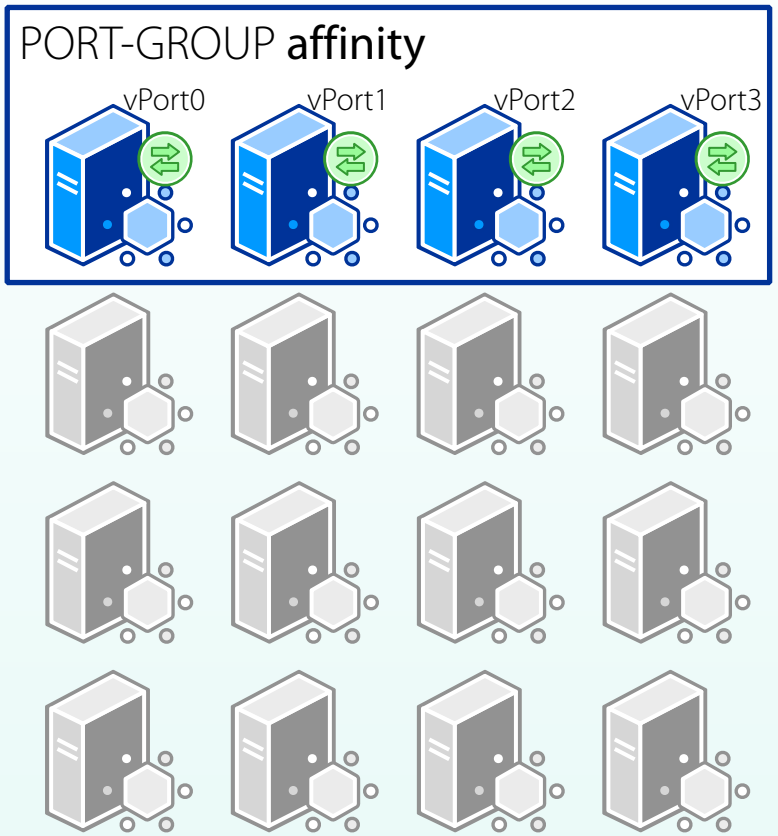


Define the set of computes that can host a container for a particular network service

Group Scheduling Policies

1 Affinity Policies

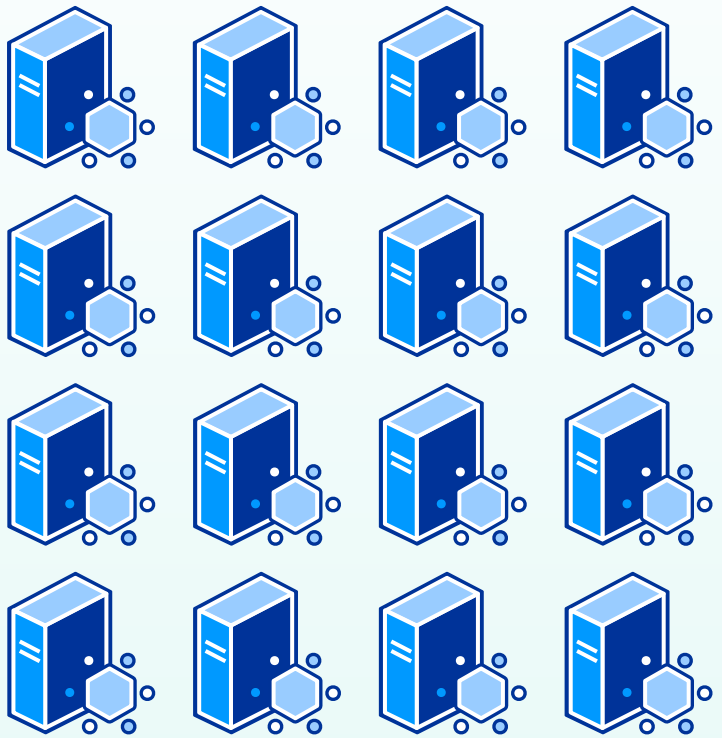




Define the set of computes that can host a container for a particular network service

Group Scheduling Policies

2 Selection Policies



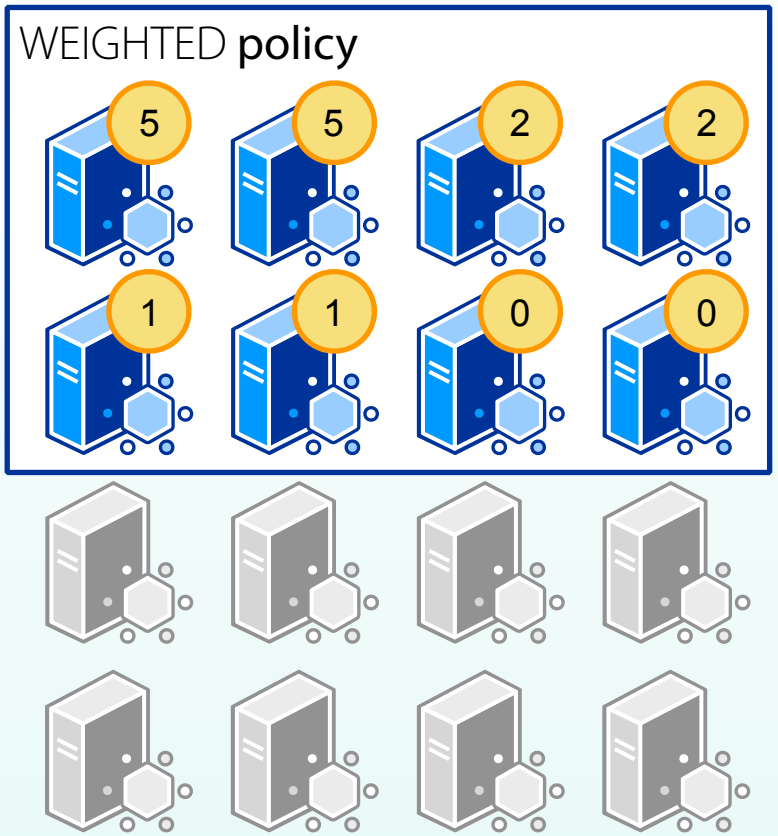
Choosing a particular compute for a container based on a static or dynamic metric

Group Scheduling Policies

2 Selection Policies

Static metric

```
host host0 set container-weight 5
host host6 set container-weight 0
```



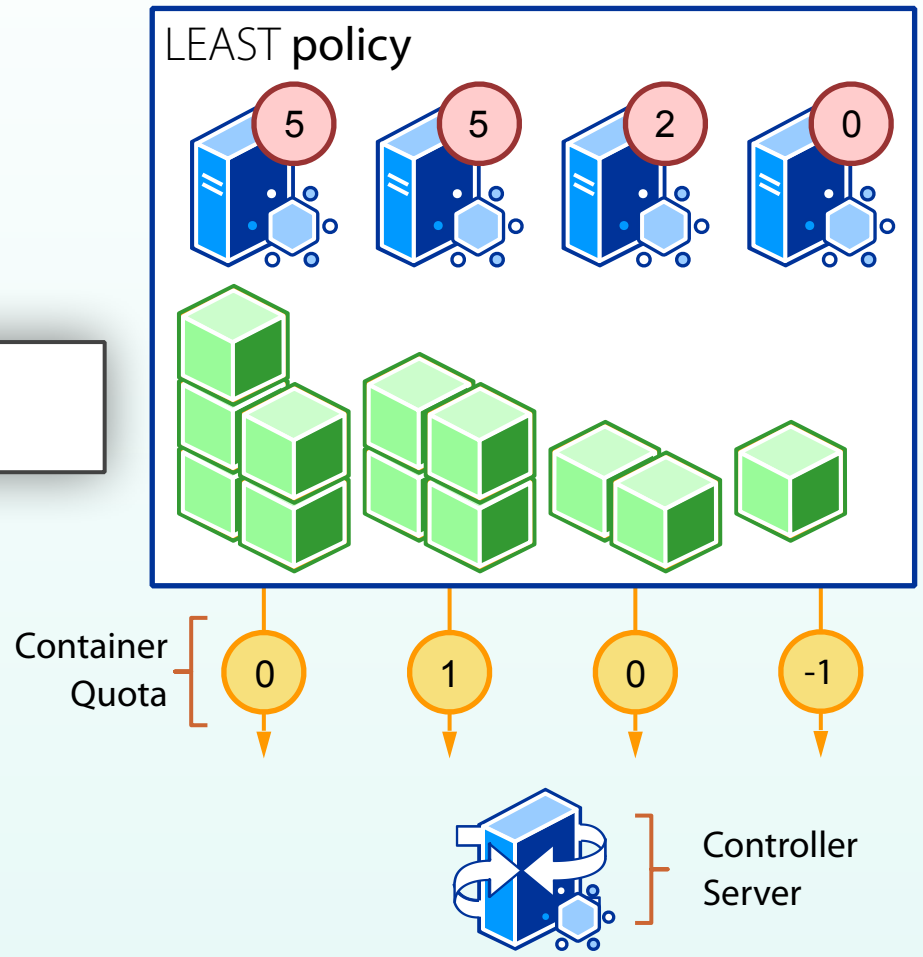
Choosing a particular compute for a container based on a static or live metric

Group Scheduling Policies

2 Selection Policies

Live metric

```
host host0 set container-limit 5
host host3 set container-limit 0
```



Choosing a particular compute for a container based on a static or live metric

Live Demo

Container Scheduling

Test Drive

Quickstart **midonet.org**

Packages **builds.midonet.org**

GitHub **github.com/midonet**

Chat **slack.midonet.org**

Q&A