OCP and OSP

NIRS 강하고 빠른 리뷰

오픈스택

오픈스택 아키텍처


WE ARE
OPEN
STACK

오픈스택 아키텍처

Oslo

Oslo

오슬로는 파이썬 기반으로 구성이 되어 있는 오픈스택 라이브러리. 이 라이브러리 기반으로 API 및 Messaging ProtocoI에 대한 사양이 정의가 되어 있다.

https://wiki.openstack.org/wiki/Oslo

Oslo

Oslo

오슬로는 파이썬 기반으로 구성이 되어 있는 오픈스택 라이브러리. 이 라이브러리 기반으로 API 및 Messaging ProtocoI에 대한 사양이 정의가 되어 있다.

https://wiki.openstack.org/wiki/Oslo

libvirtd

libvirtd

Libvirtd는 말 그대로 라이브러리 데몬이다. 리눅스에서 주로 사용하는 qemu/kvm경우에는 기본적으로 libvirt기반으로 가상머신을 생성 및 관리한다.

그 이유는 qemu/kvm경우에는 하이퍼바이저가 없이 동작하는 type-1형태의 가상머신 구조를 가지고 있기 때문에 libvirt와 같은 서비스가 필요하다.

Libvirt는 qemu/kvm에서 제공하지 못하는 DOM(Domain)영역을 실제로 존재하지는 않지만, Process Info를 통해서 Net, Disk, Process, Mem와 같은 영역 정보를 가져온다.

libvirtd

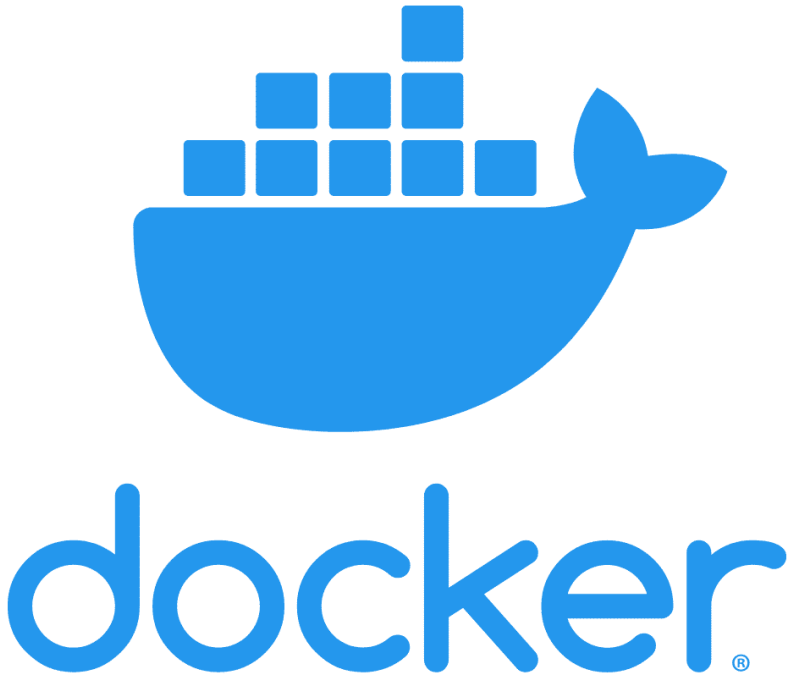
오픈스택 경우에는 "nova-compute"에서 libvirtd와 함께 같이 사용한다.

일반적으로 컴퓨트 서비스에서 장애가 발생하면 "nova-compute"컨테이너 혹은 alias 설정하여 compute에서 관리하고 있는 가상머신 정보를 관리한다.

podman vs docker

podman

podman vs docker



podman vs docker

레드햇 오픈스택 기준으로 OpenStack 14이후 부터는 본격적으로 컨테이너 기반으로 오픈스택 서비스를 제공하고 있다.

여기에는 두 가지 이유가 있다.

1. 안정적인 업그레이드 및 확장
2. 호스트 보안에 대한 강화

podman vs docker

KOLLA

an OpenStack Community Project

podman vs docker

커뮤니티 버전은 **ansible kolla 혹은 koala**라고 부르는 버전 기반으로 관리가 되고 있다.

현재 커뮤니티 버전은 docker-engine은 CRI 표준 규약에 맞는 버전으로 운영이 되고 있지만, 컨테이너 커뮤니티에서는 Docker보다는 Podman기반으로 사용 및 운영을 권장하고 있다.

현재 여러분이 사용하는 컨테이너 런타임은 runc(crun), conmon, podman같은 조합으로 동작 및 운영이 되고 있다.

podman vs docker

운영 시 사용하는 컨테이너들은 소프트웨어만 격리가 되어 있으며, 하드웨어 자원은 호스트에서 직접 접근을 해야 하기 때문에, "privileged mode"기반으로 동작하고 있다.

이러기 때문에 컨테이너에서 동작하는 모든 프로세스는 호스트 하드웨어에 직접 접근이 가능하며, 이를 통해서 KVM 및 Block Storage에 직접 접근이 가능하다.

Kolla

Kolla 프로젝튼 앤서블 기반으로 오픈스택 설치. 앞에서 언급한 컨테이너 기반으로 주요 오픈스택 프로그램을 설치한다.

다만, 레드햇에서 사용하는 Kolla경우에는 커뮤니티와 다르게 docker대신 podman를 사용한다.

Kolla는 기본적으로 **우분투, 데비안, 센트**를 지원하며, 레드햇에서 제공하는 Kolla경우에는 **RHEL기반**으로 지원한다.

Kolla More…

레드햇과 Kolla는 둘 다 동일하게 HAProxy 및 Pacemaker(Corosync)기반으로 H/A기능을 제공하고 있다.

이를 통해서 이전 레드햇 오픈스택 14버전과 다르게 구성 및 설치에 대한 편의성을 제공 하고 있다.

다만, 이러한 구조적인 즉, Hosted 및 Containerd형식 차이로 인하여 버전 업그레이드가 어려운 부분이 있다.

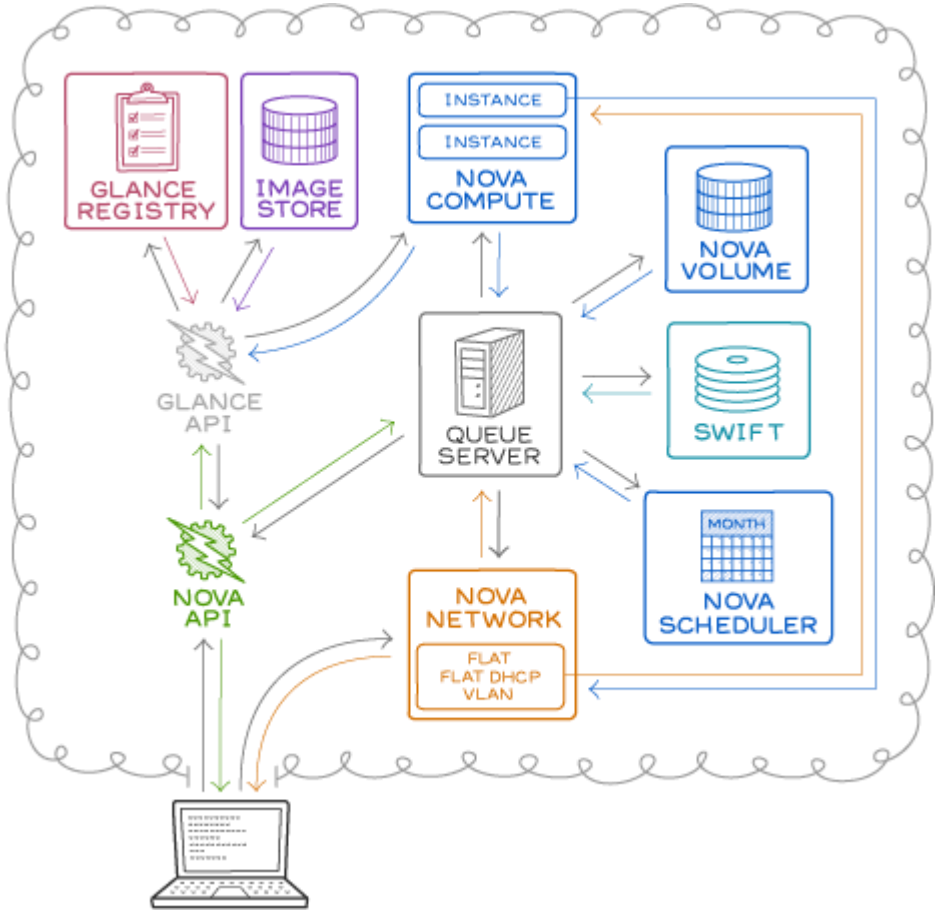
nova, nova-compute

오픈스택은 기본적인 목적은 **가상머신 생성 및 구성**이 주요 목적이다.

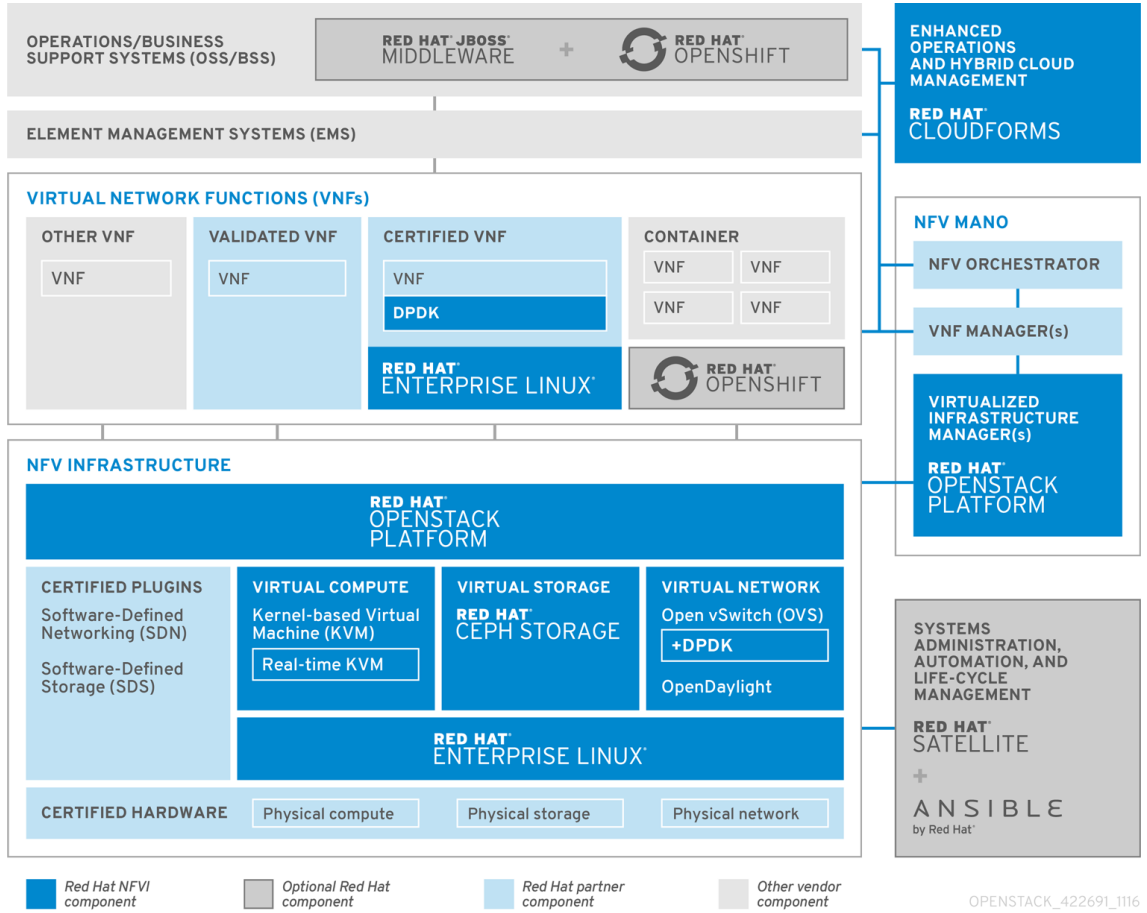
현재 오픈스택의 기본 하이퍼바이저 혹은 가상화 기술로 선택하고 있는 기술은 바로 QEMU/KVM이다.

QEMU/KVM기반으로 사용하는 경우 여러대의 가상머신을 관리하기가 어렵기 때문에 일반적으로 libvirt와 함께 사용한다.

OpenStack


GLANCE REGISTRY
IMAGE STORE
INSTANCE
INSTANCE
NOVA COMPUTE
NOVA VOLUME
GLANCE API
QUEUE SERVER
SWIFT
MONTH
NOVA SCHEDULER
NOVA API
NOVA NETWORK
FLAT
FLAT DHCP
VLAN

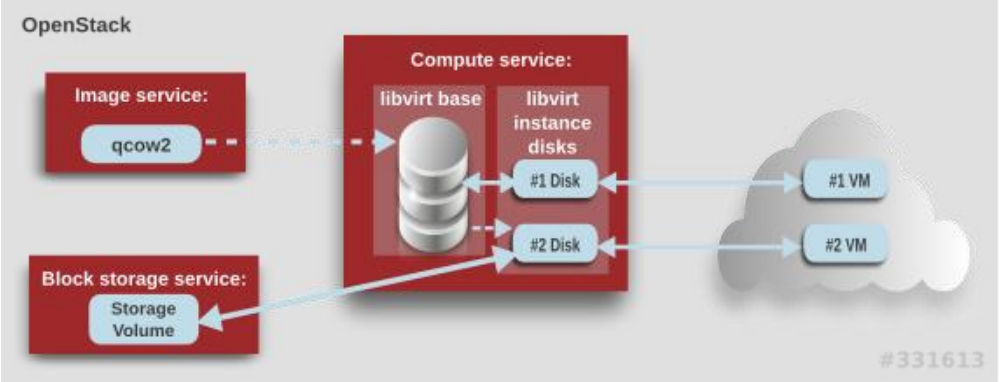
OpenStack


OPERATIONS/BUSINESS SUPPORT SYSTEMS (OSS/BSS)
RED HAT JBOSS MIDDLEWARE
RED HAT OPENSHIFT
ELEMENT MANAGEMENT SYSTEMS (EMS)
VIRTUAL NETWORK FUNCTIONS (VNFs)
OTHER VNF
VNF
VALIDATED VNF
VNF
CERTIFIED VNF
VNF
DPDK
CONTAINER
VNF
VNF
VNF
VNF
RED HAT ENTERPRISE LINUX
RED HAT OPENSHIFT
NFV INFRASTRUCTURE
RED HAT OPENSTACK PLATFORM
CERTIFIED PLUGINS
Software-Defined Networking (SDN)
Software-Defined Storage (SDS)
VIRTUAL COMPUTE
Kernel-based Virtual Machine (KVM)
Real-time KVM
VIRTUAL STORAGE
RED HAT CEPH STORAGE
VIRTUAL NETWORK
Open vSwitch (OVS)
+DPDK
OpenDaylight
RED HAT ENTERPRISE LINUX
CERTIFIED HARDWARE
Physical compute
Physical storage
Physical network
ENHANCED OPERATIONS AND HYBRID CLOUD MANAGEMENT
RED HAT CLOUDFORMS
NFV MANO
NFV ORCHESTRATOR
VNF MANAGER(s)
VIRTUALIZED INFRASTRUCTURE MANAGER(s)
RED HAT OPENSTACK PLATFORM
SYSTEMS ADMINISTRATION, AUTOMATION, AND LIFE-CYCLE MANAGEMENT
RED HAT SATELLITE
+
ANSIBLE by Red Hat
Red Hat NFVI component
Optional Red Hat component
Red Hat partner component
Other vendor component
OPENSTACK_422691_1116

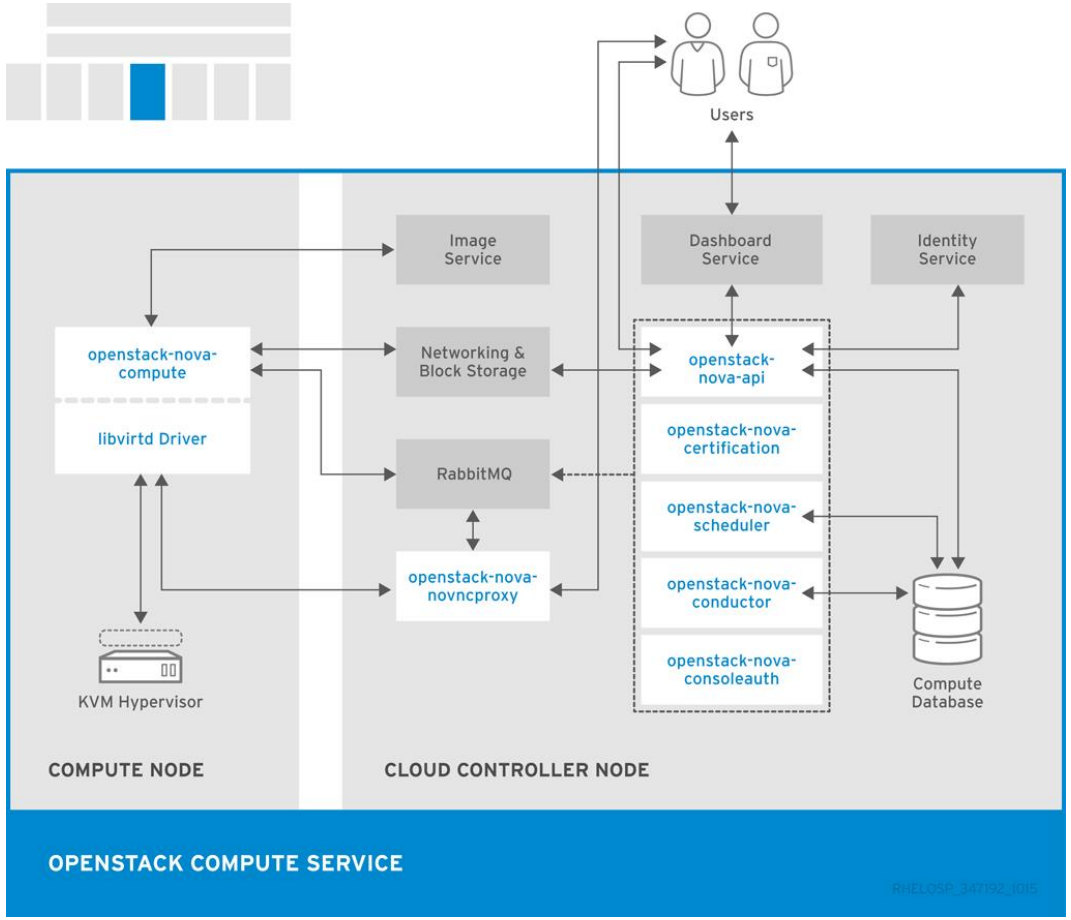
OpenStack


IaaS+
MONITORING
DATA ANALYTICS
SHARED FILESYSTEM
ORCHESTRATION
SECRETS
LOAD BALANCING
SHARED SERVICES
DEPLOYMENT and MANAGEMENT
IDENTITY
DIRECTOR
TELEMETRY
OpenShift / Ceph
MANILA
HEAT
BARBICAN
OCTAVIA
KEYSTONE
IaaS
COMPUTE
STORAGE
BLOCK
IMAGE
OBJECT
NETWORKING
BARE-METAL PROVISIONING
DASHBOARD
NOVA
CINDER
GLANCE
SWIFT
NEUTRON
IRONIC
HORIZON
TRIPLEO

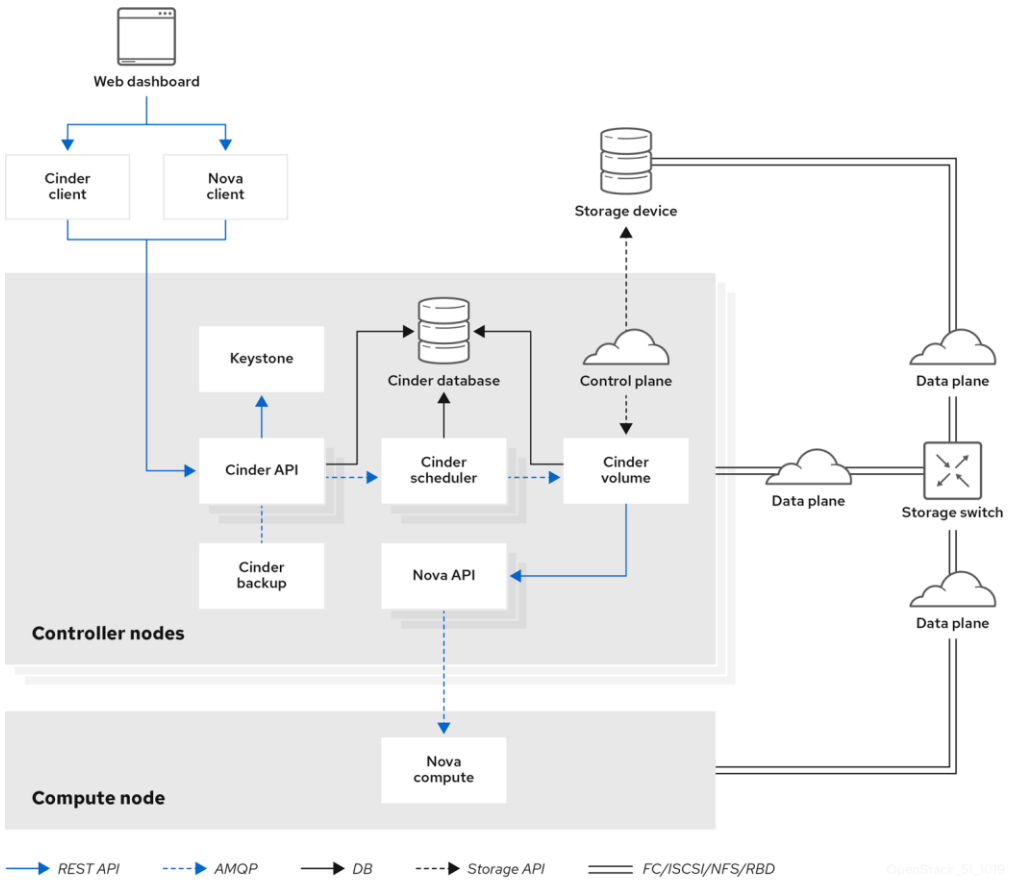
OpenStack


OpenStack
Image service:
qcow2
Compute service:
libvirt base
libvirt instance disks
#1 Disk
#2 Disk
#1 VM
#2 VM
Block storage service:
Storage Volume
#331613

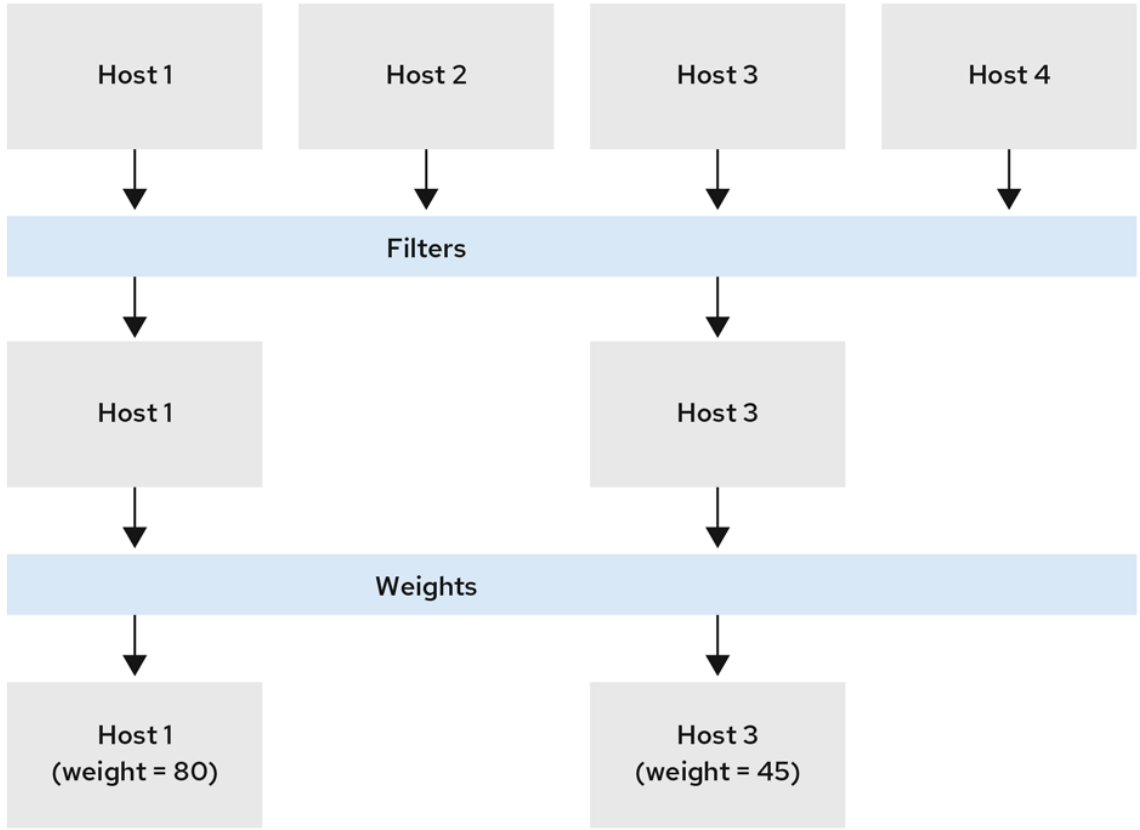
OpenStack


Users
Image Service
Dashboard Service
Identity Service
openstack-nova-compute
libvirtd Driver
Networking & Block Storage
RabbitMQ
openstack-nova-novncproxy
openstack-nova-api
openstack-nova-certification
openstack-nova-scheduler
openstack-nova-conductor
openstack-nova-consoleauth
KVM Hypervisor
Compute Database
COMPUTE NODE
CLOUD CONTROLLER NODE
OPENSTACK COMPUTE SERVICE

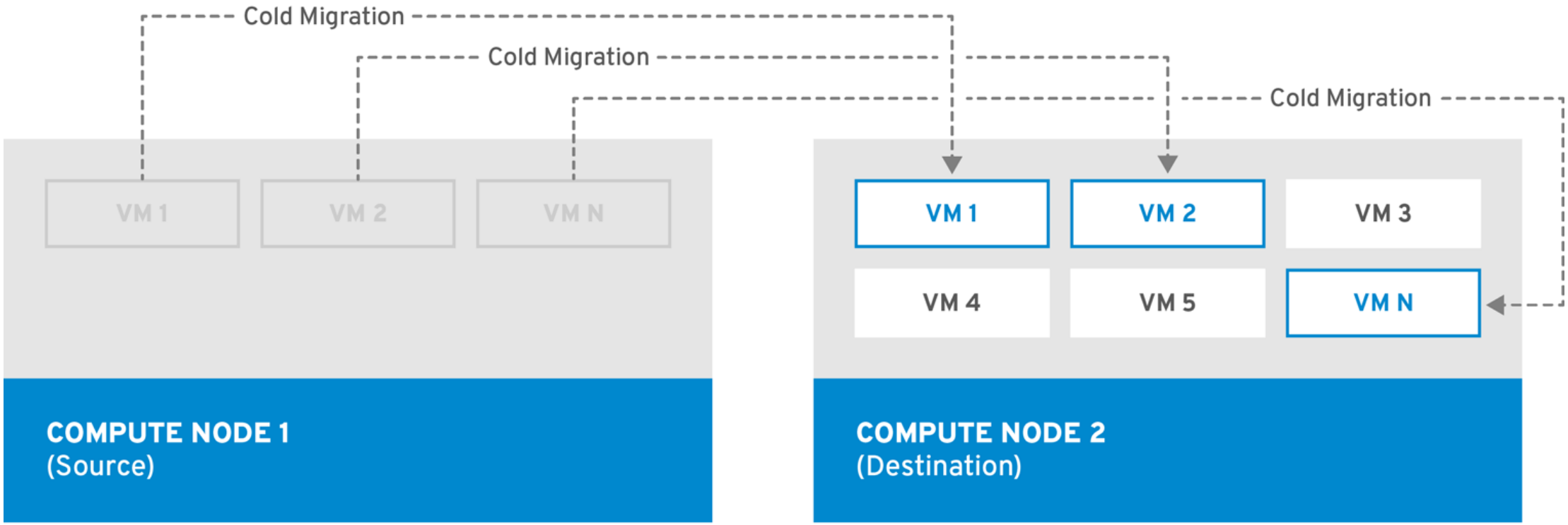
OpenStack


Web dashboard
Cinder client
Nova client
Storage device
Keystone
Cinder database
Control plane
Data plane
Cinder API
Cinder scheduler
Cinder volume
Data plane
Storage switch
Cinder backup
Nova API
Data plane
Controller nodes
Nova compute
Compute node
REST API
AMQP
DB
Storage API
FC/ISCSI/NFS/RBD

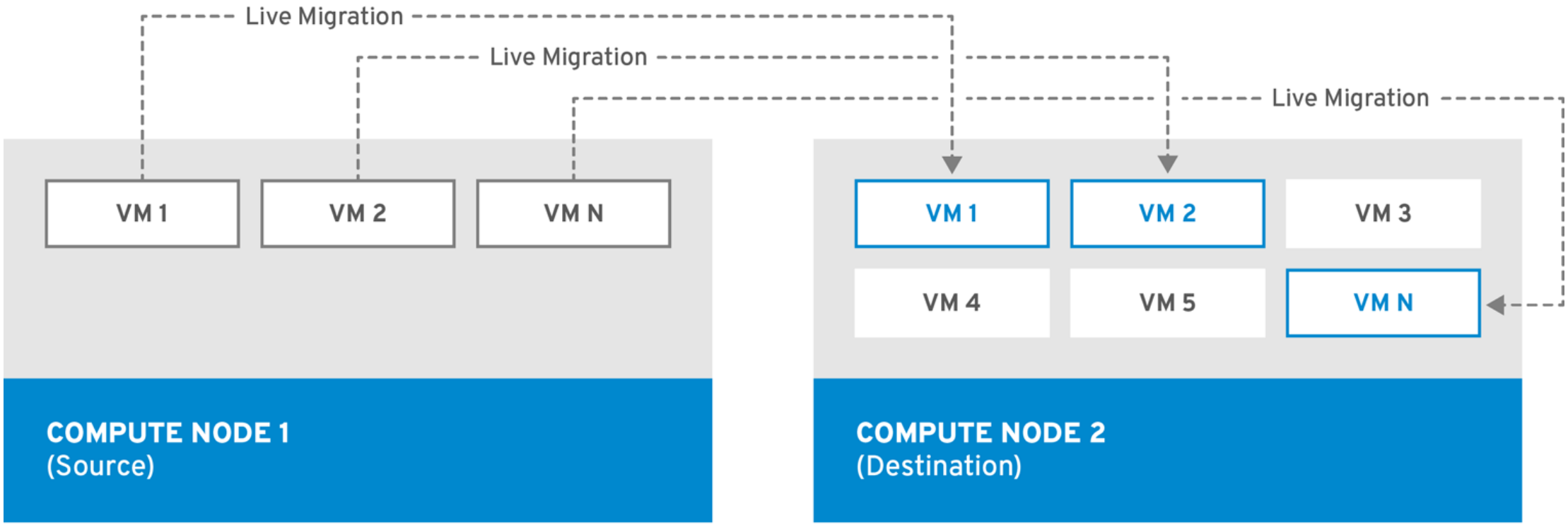
Nova



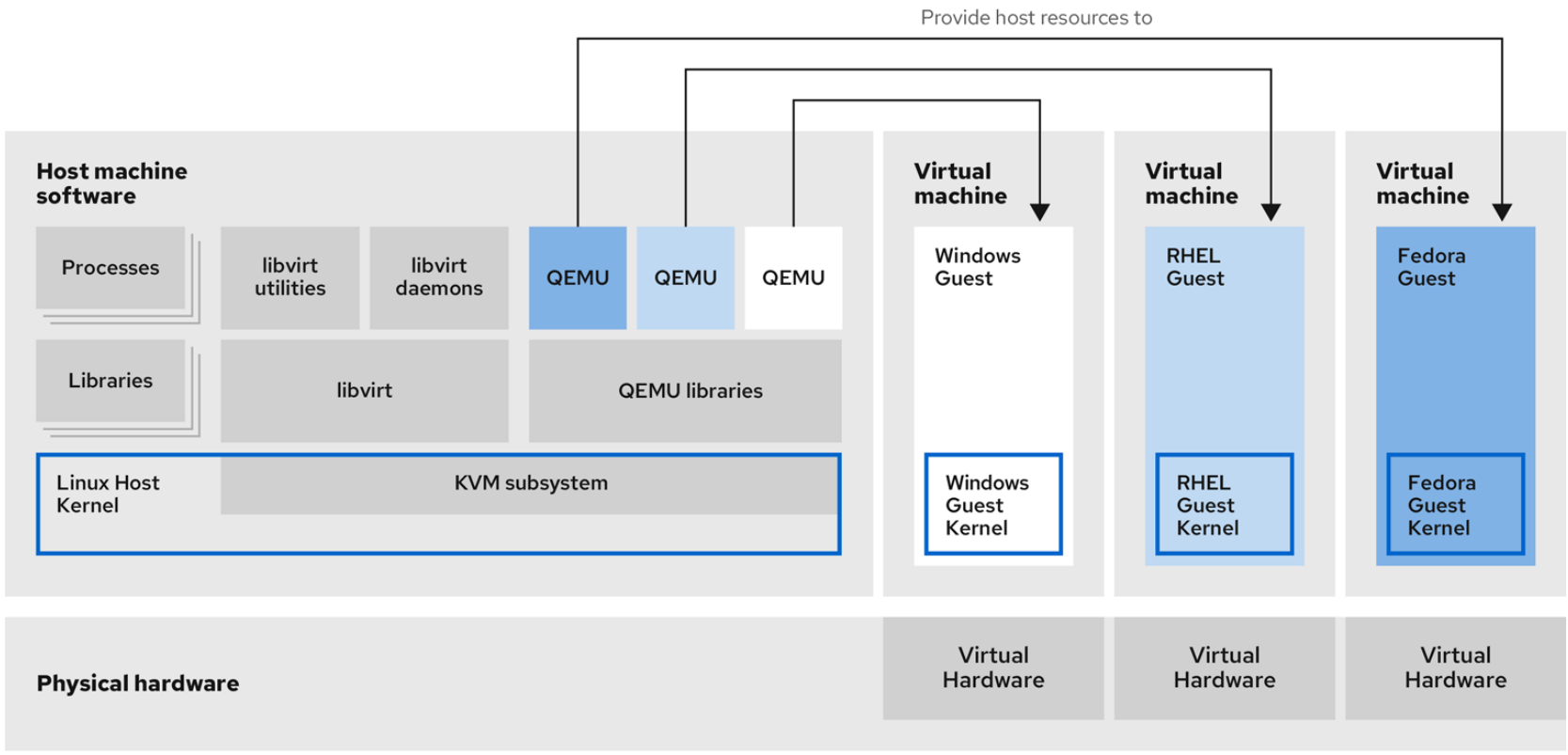
Nova


Cold Migration
Cold Migration
Cold Migration
VM 1
VM 2
VM N
COMPUTE NODE 1
(Source)
VM 1
VM 2
VM 3
VM 4
VM 5
VM N
COMPUTE NODE 2
(Destination)
OPENSTACK_11_0419

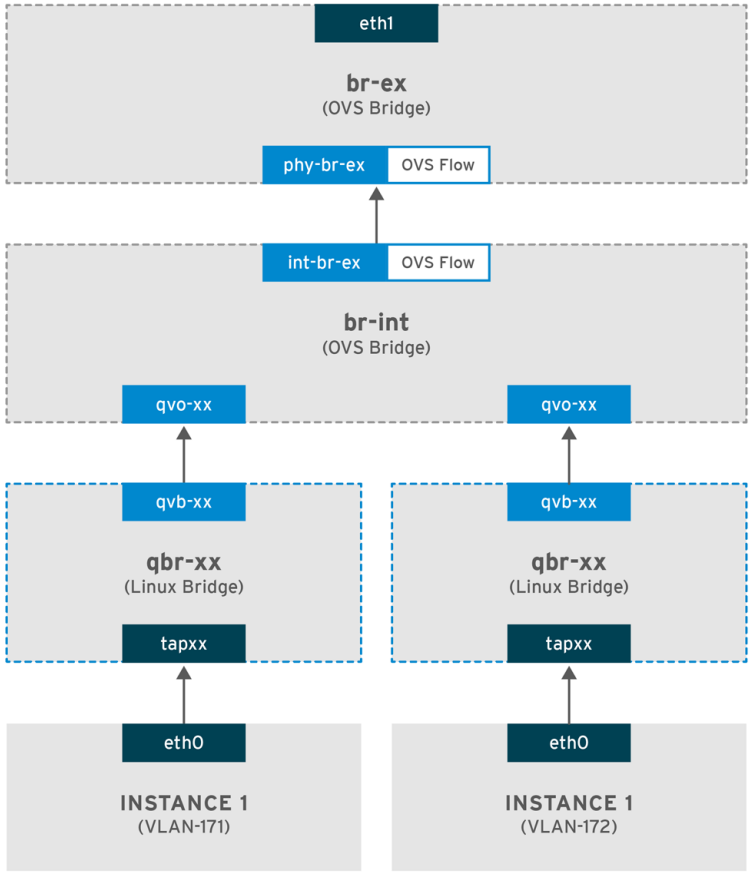
Nova


Live Migration
Live Migration
Live Migration
VM 1
VM 2
VM N
COMPUTE NODE 1
(Source)
VM 1
VM 2
VM 3
VM 4
VM 5
VM N
COMPUTE NODE 2
(Destination)
OPENSTACK_11_0419

libvirt

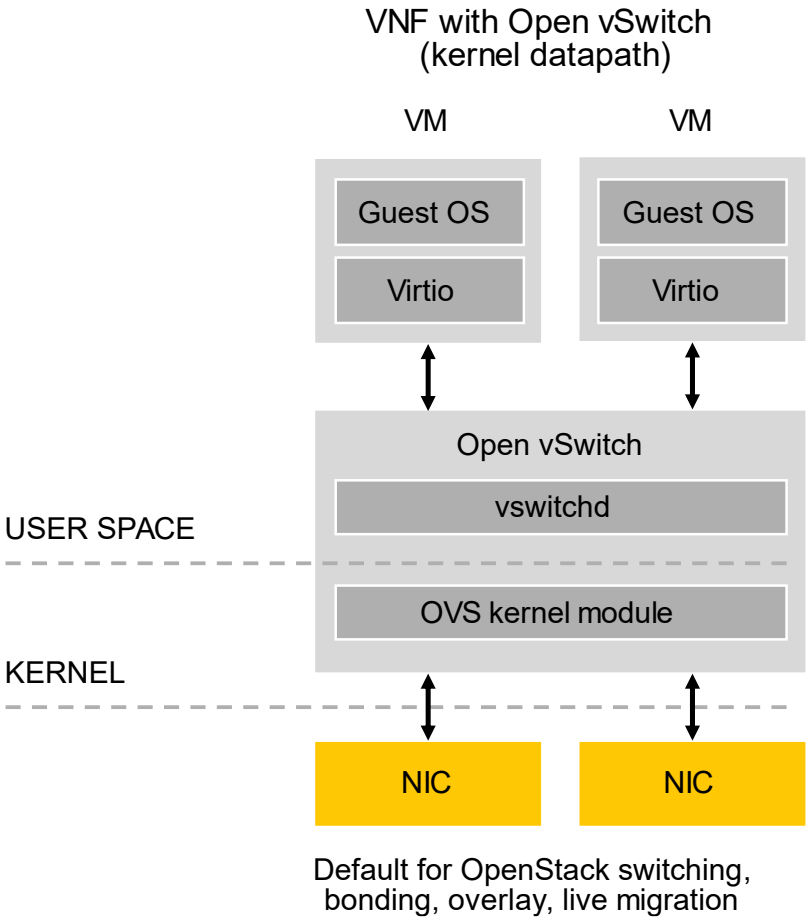

Provide host resources to
Host machine software
Processes
libvirt utilities
libvirt daemons
QEMU
QEMU
QEMU
Libraries
libvirt
QEMU libraries
Linux Host Kernel
KVM subsystem
Virtual machine
Windows Guest
Windows Guest Kernel
Virtual machine
RHEL Guest
RHEL Guest Kernel
Virtual machine
Fedora Guest
Fedora Guest Kernel
Physical hardware
Virtual Hardware
Virtual Hardware
Virtual Hardware
244_RHEL_0422

Instance qb bridge with namespace TAP DEV

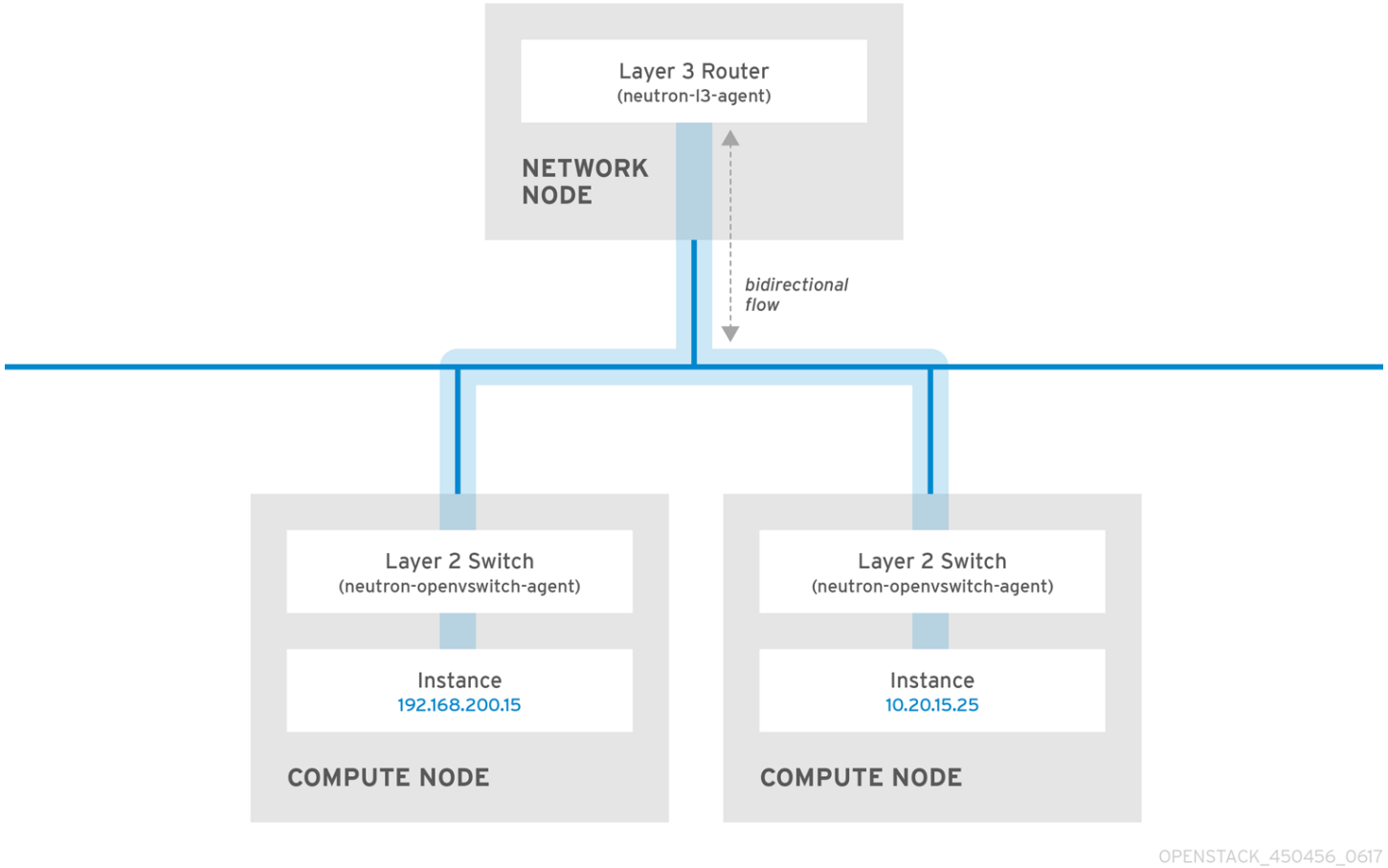


OPENSTACK_450456_0617

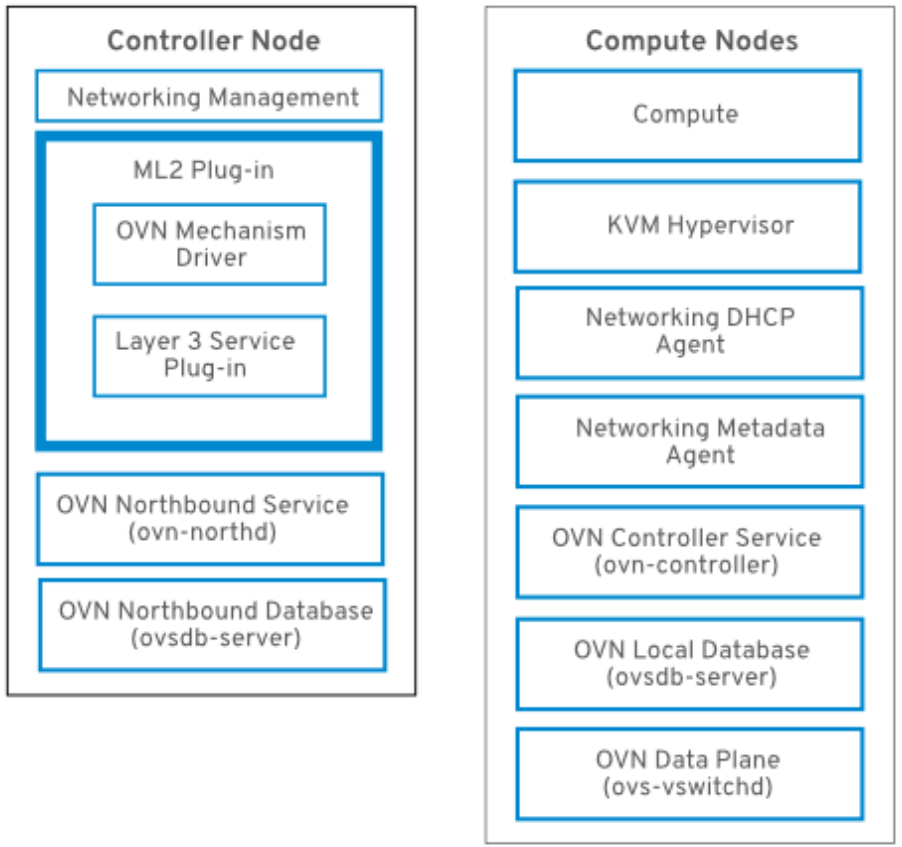
OVS


VNF with Open vSwitch
(kernel datapath)
VM
VM
Guest OS
Virtio
Guest OS
Virtio
Open vSwitch
vswitchd
USER SPACE
OVS kernel module
KERNEL
NIC
NIC
Default for OpenStack switching,
bonding, overlay, live migration

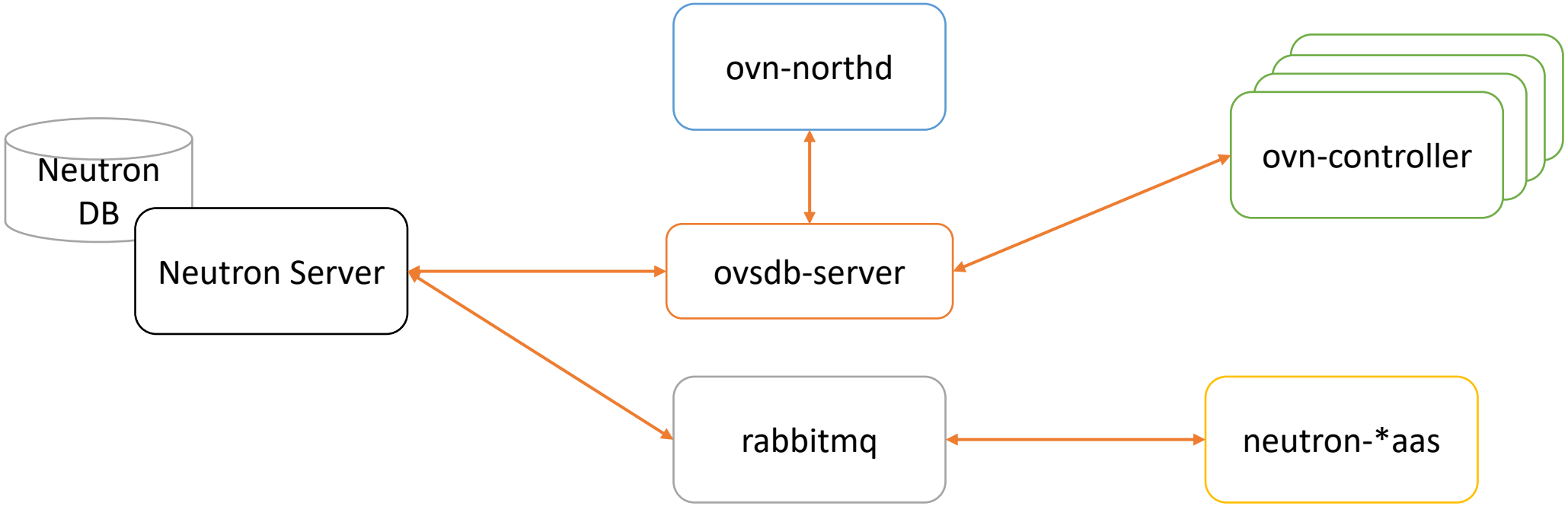
OVS


Layer 3 Router
(neutron-l3-agent)
NETWORK NODE
bidirectional flow
Layer 2 Switch
(neutron-openvswitch-agent)
Instance
192.168.200.15
COMPUTE NODE
Layer 2 Switch
(neutron-openvswitch-agent)
Instance
10.20.15.25
COMPUTE NODE
OPENSTACK_450456_0617

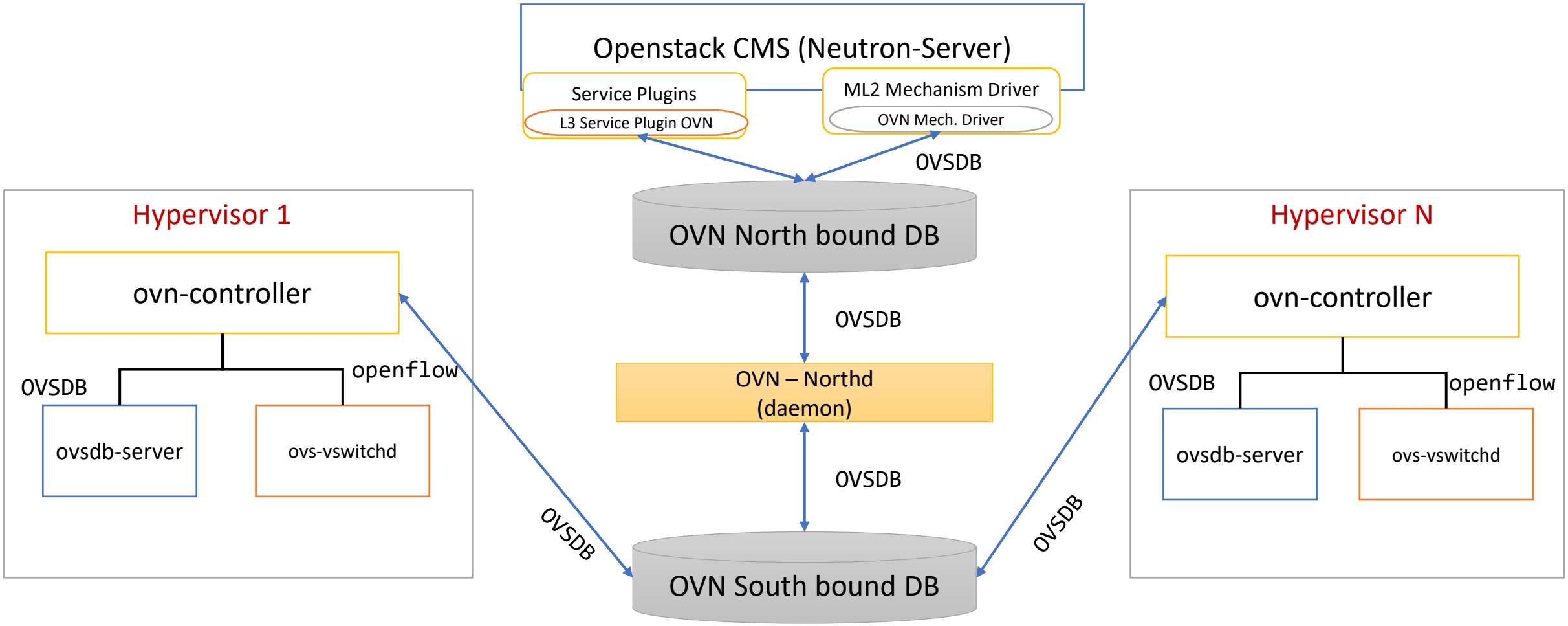
OVN


Controller Node
Networking Management
ML2 Plug-in
OVN Mechanism Driver
Layer 3 Service Plug-in
OVN Northbound Service (ovn-northd)
OVN Northbound Database (ovsdb-server)
Compute Nodes
Compute
KVM Hypervisor
Networking DHCP Agent
Networking Metadata Agent
OVN Controller Service (ovn-controller)
OVN Local Database (ovsdb-server)
OVN Data Plane (ovs-vswitchd)

Openstack Neutron with OVN - Overview



OVN – Architecture

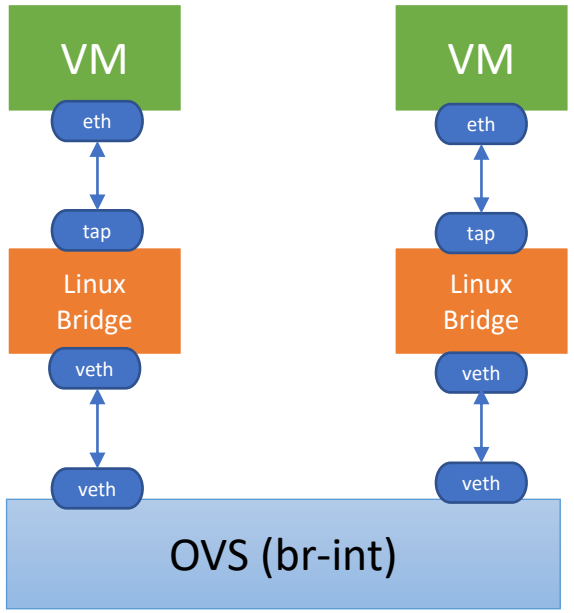


ovn-northd

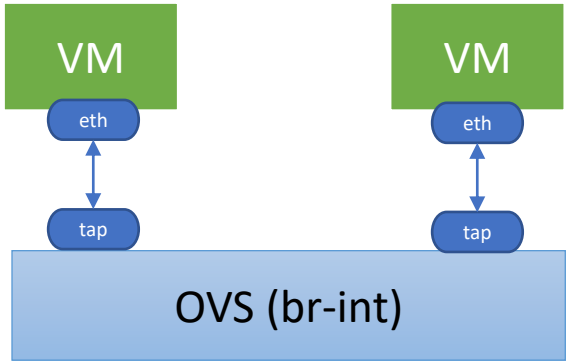
Translate between the logical network elements configured by the CMS to the Northbound DB and model them to the Southbound DB tables, which holds the physical/infrastructure bindings and the logical flows which enable the logical connectivity.

OVN – Security Groups

- Existing way
  - Requires extra linux bridge and vEth pair per VM.
  - Uses Iptables.

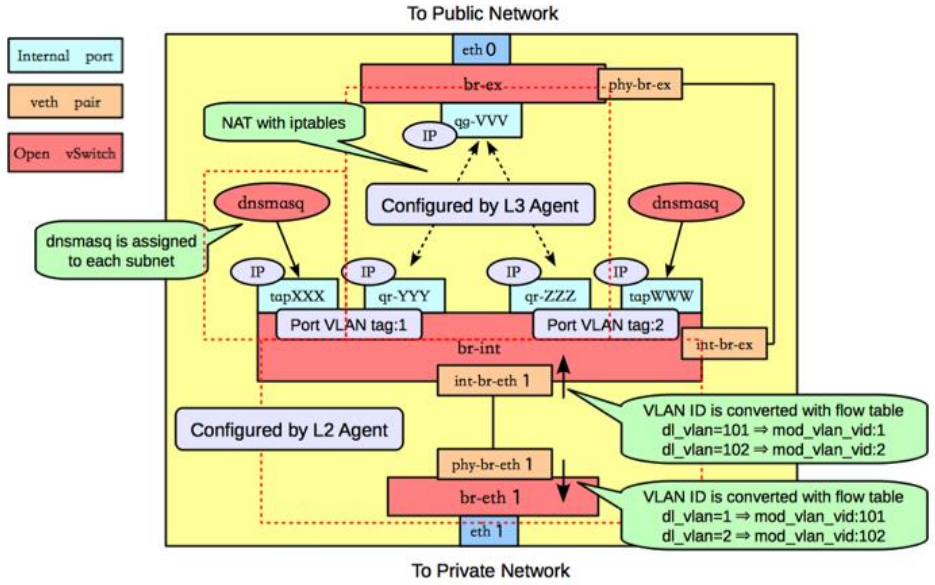


- Using OVN ACLs
  - Uses kernel conntrack module directly from OVS.
  - Design benefits.
    - No complicated pipeline.
    - Faster* -- Fewer hops and veth ports.



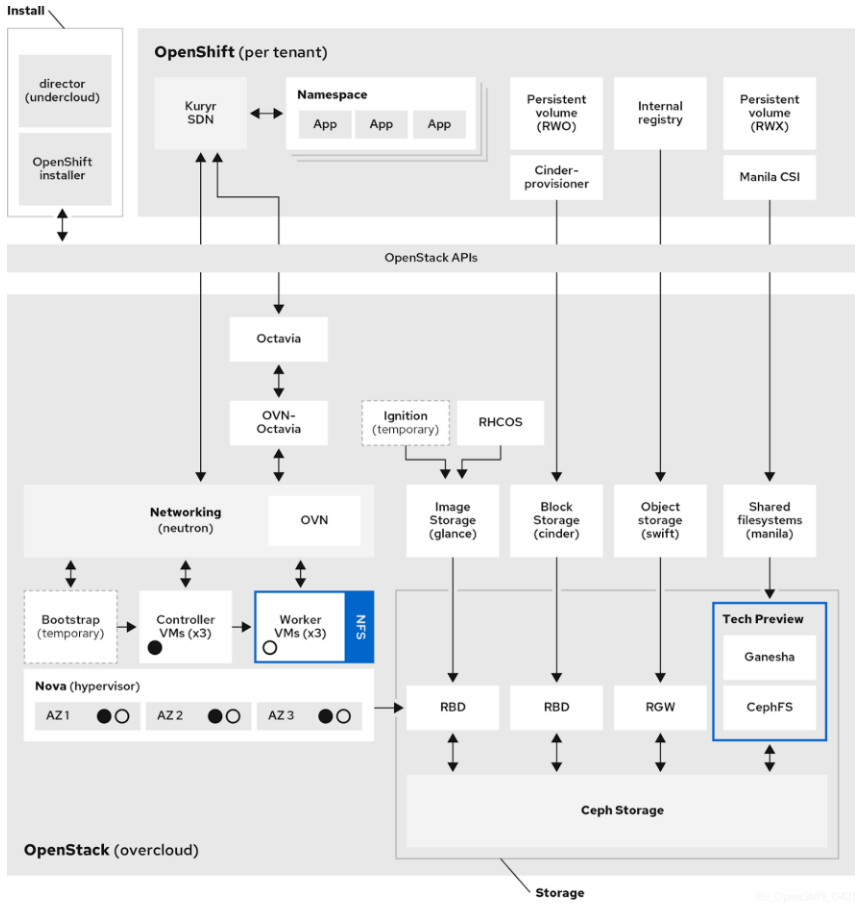
OVN – L3 design

- Neutron L3 Agent – Current design
  - Agent based.
  - Used the Linux IP stack and iptables.
    - Forwarding.
    - NAT.
  - Overlapping IP address support using namespaces

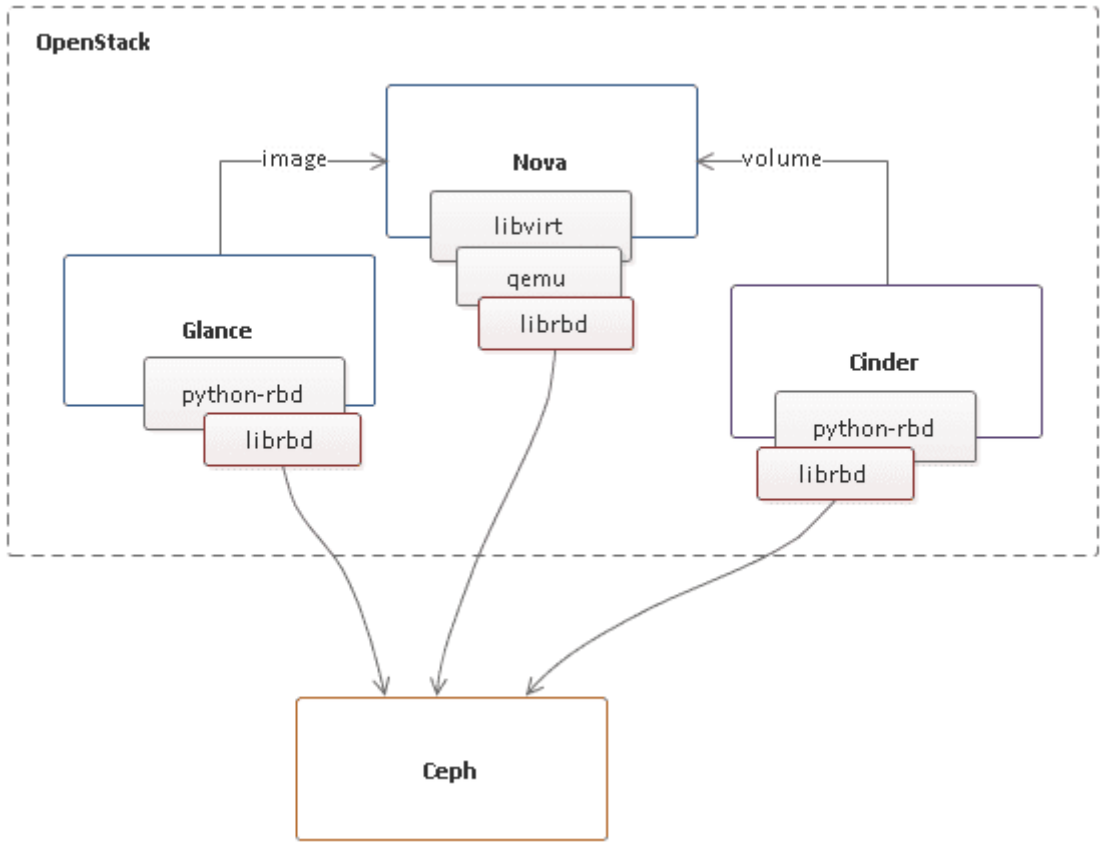


- OVN L3 design
  - Native support for IPv4 and IPv6.
  - Distributed.
  - ARP/ND suppression.
  - Flow caching improves performance.
    - Without OVN: multiple per-packet routing layers.
    - With OVN: cache sets dest mac, decrements TTL.
  - No use of Neutron L3 agent

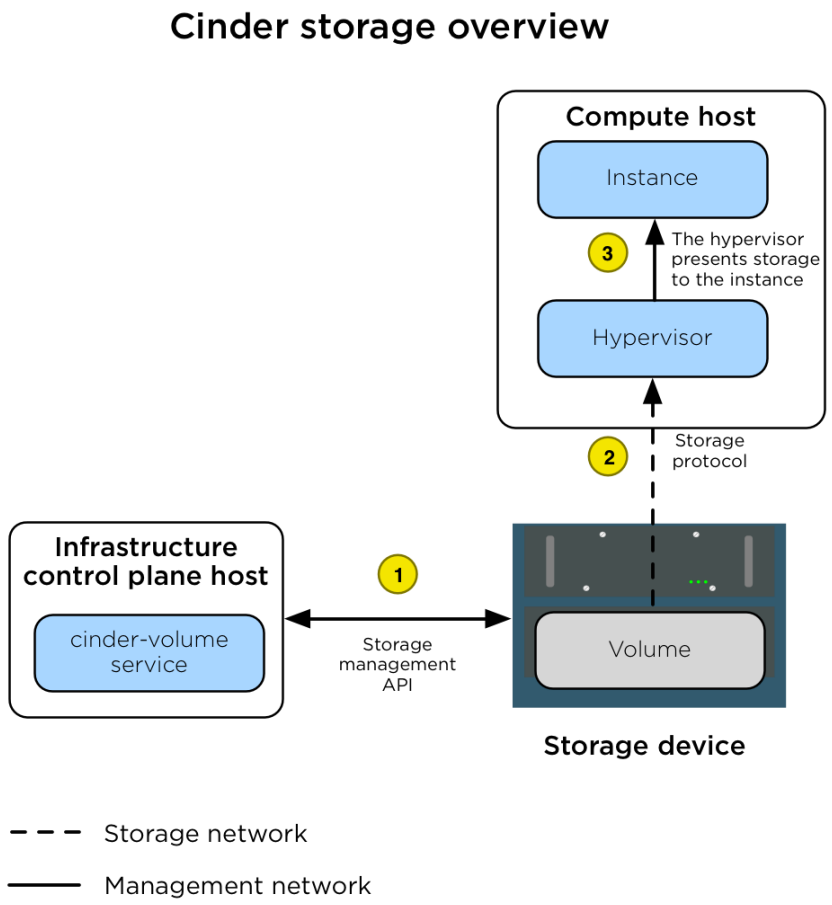
OpenShift on OpenStack



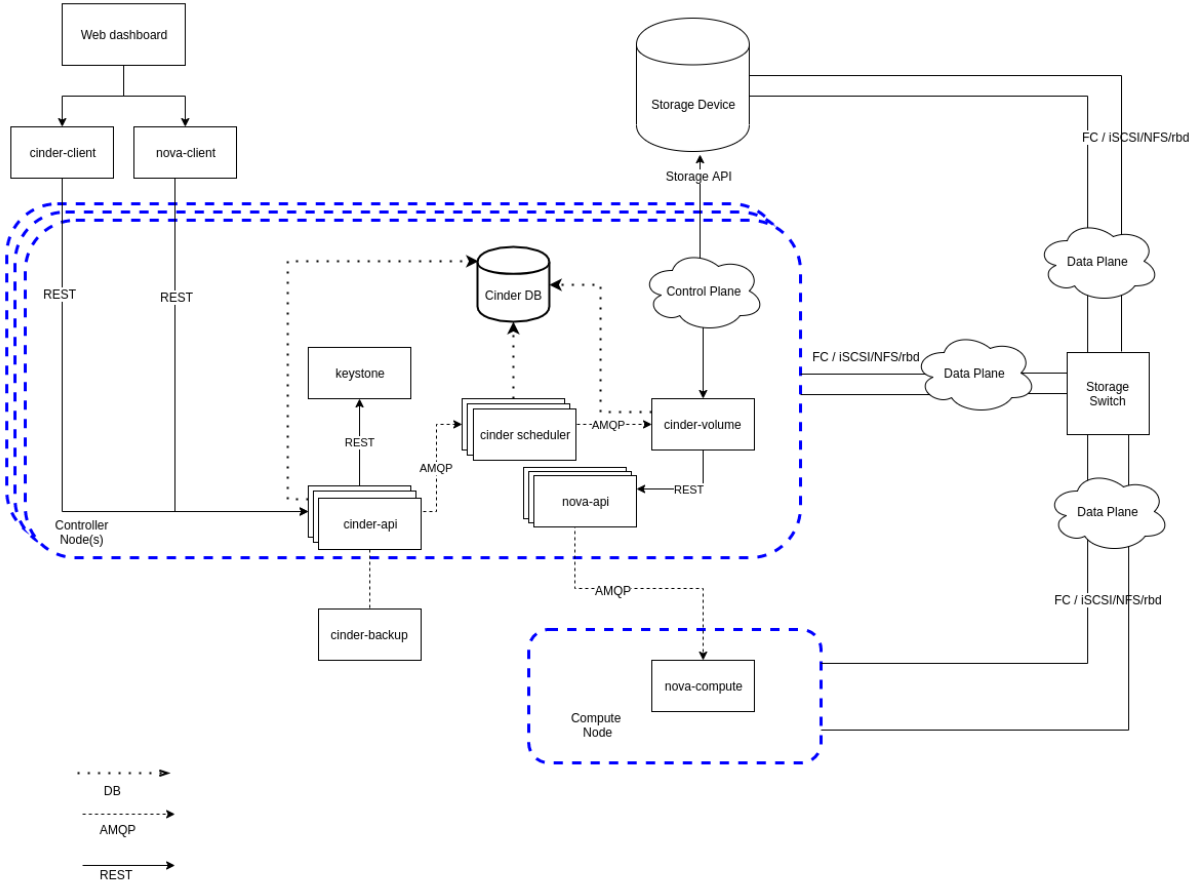
nova with storage



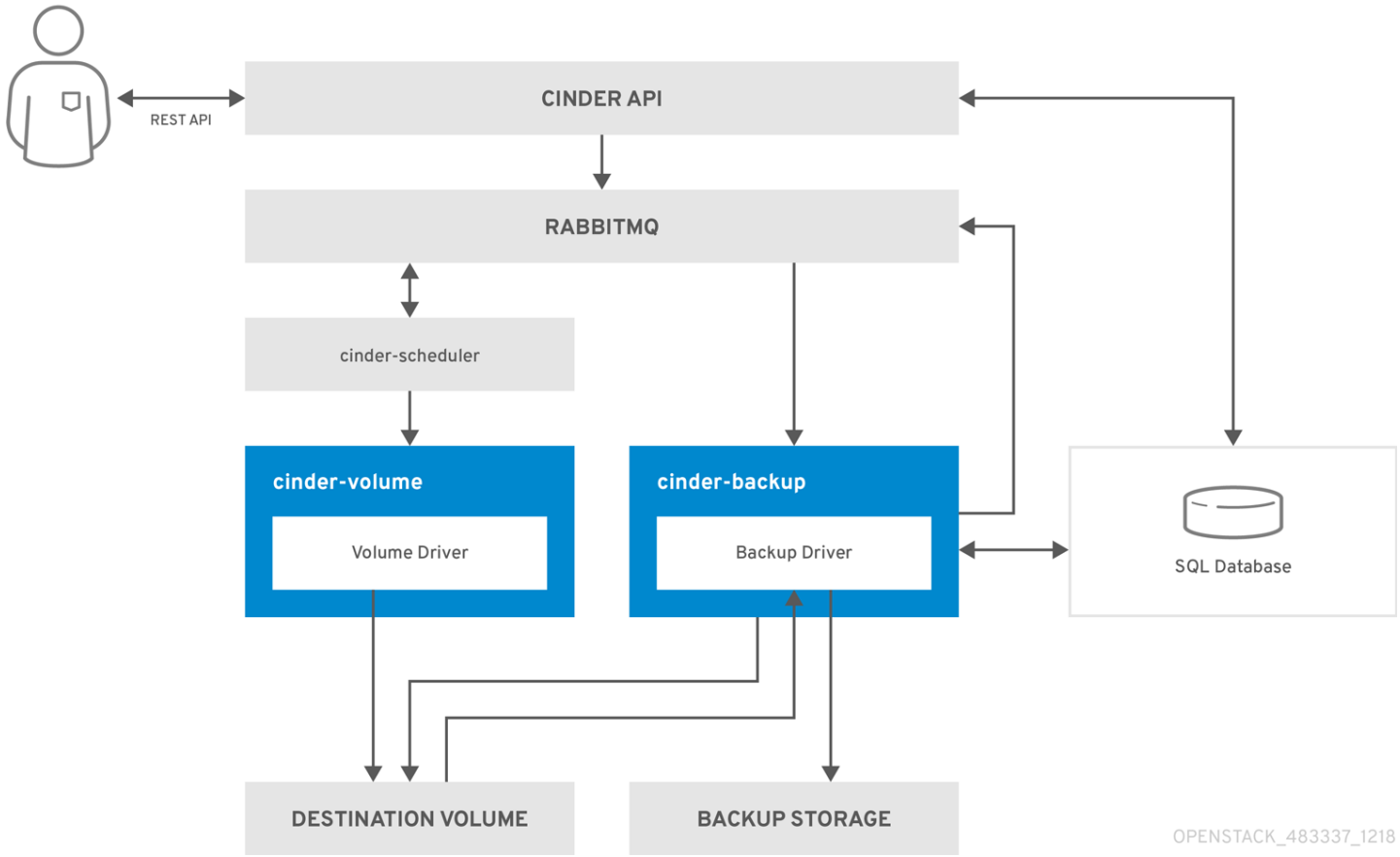
nova with storage



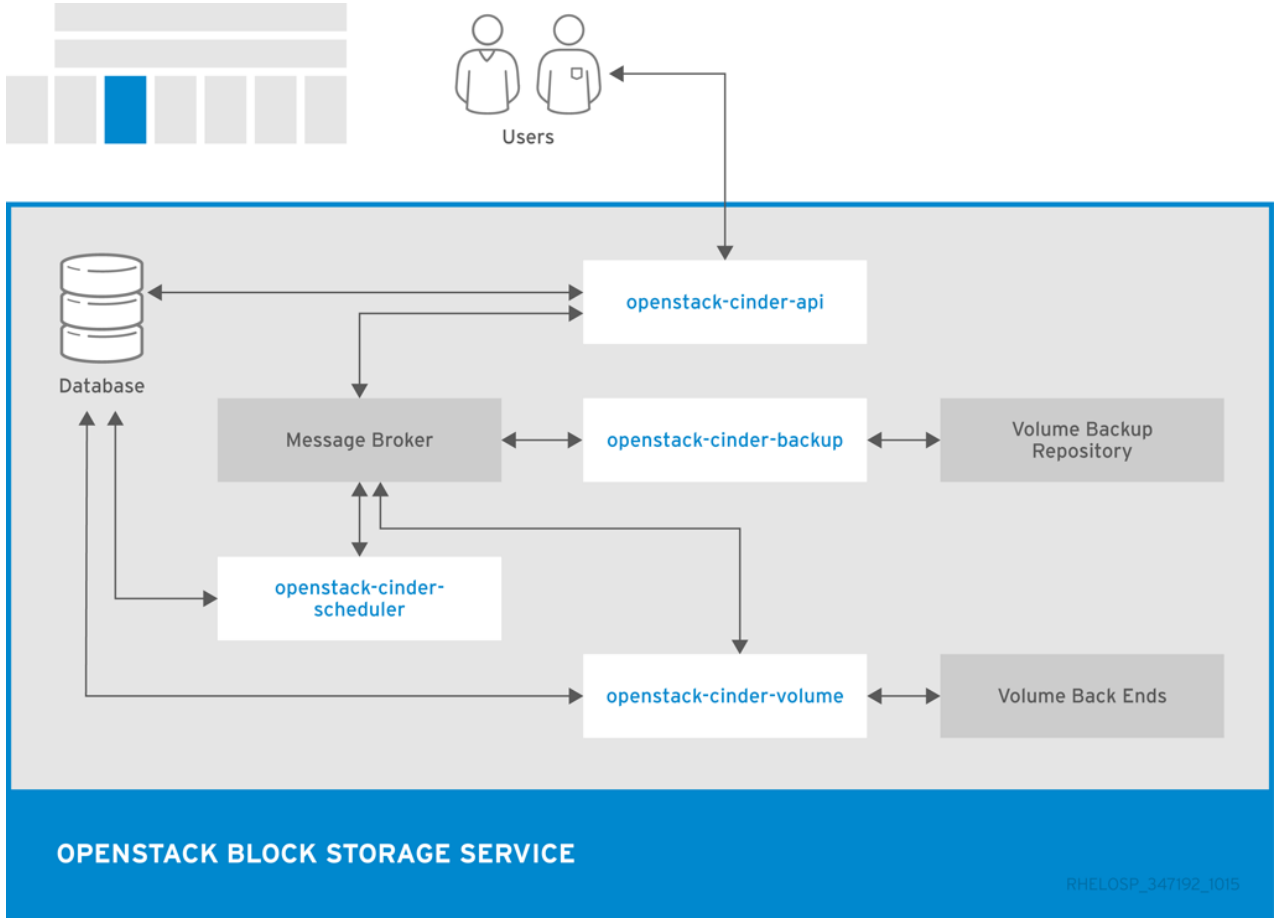
nova with storage



nova with storage



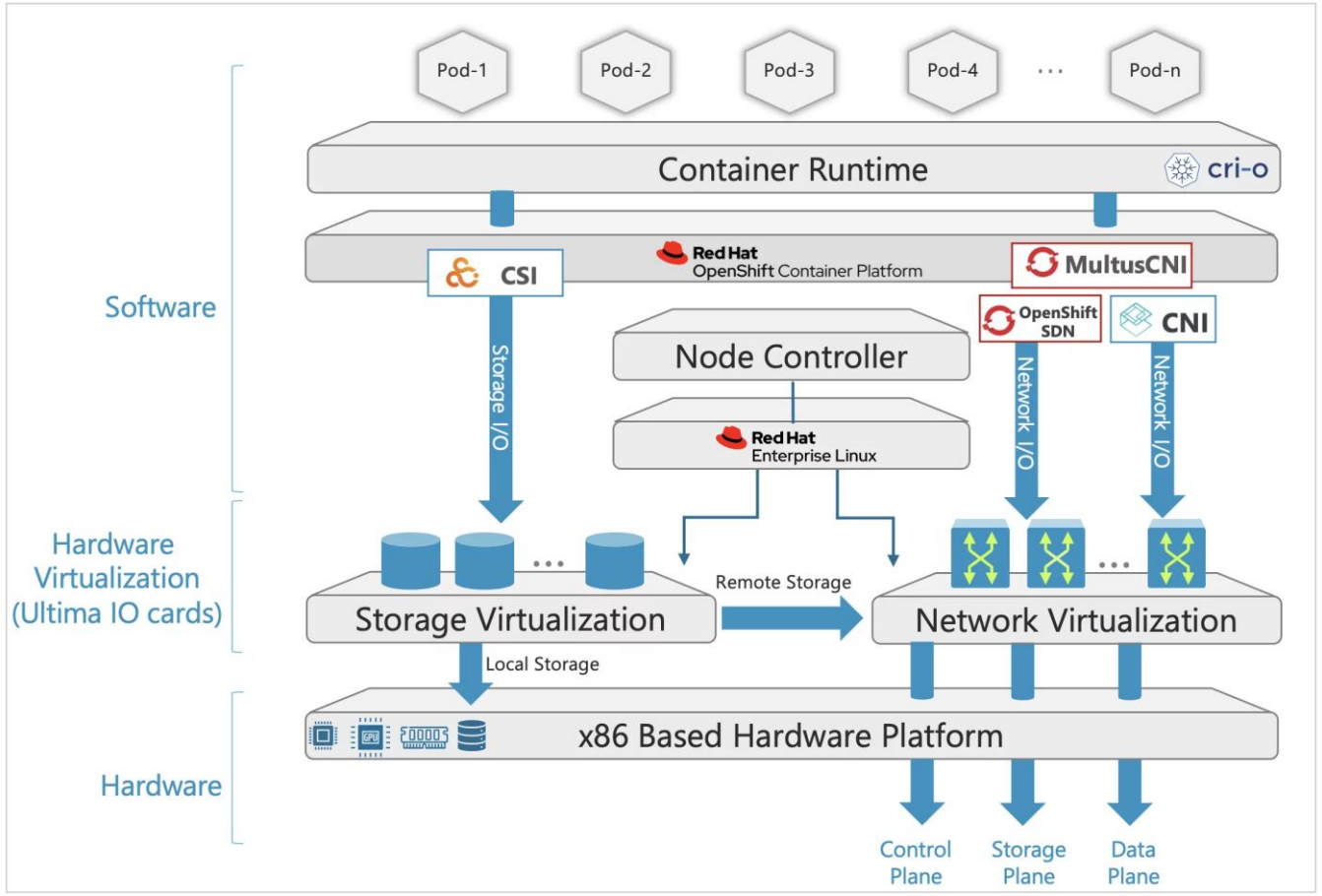
nova with storage



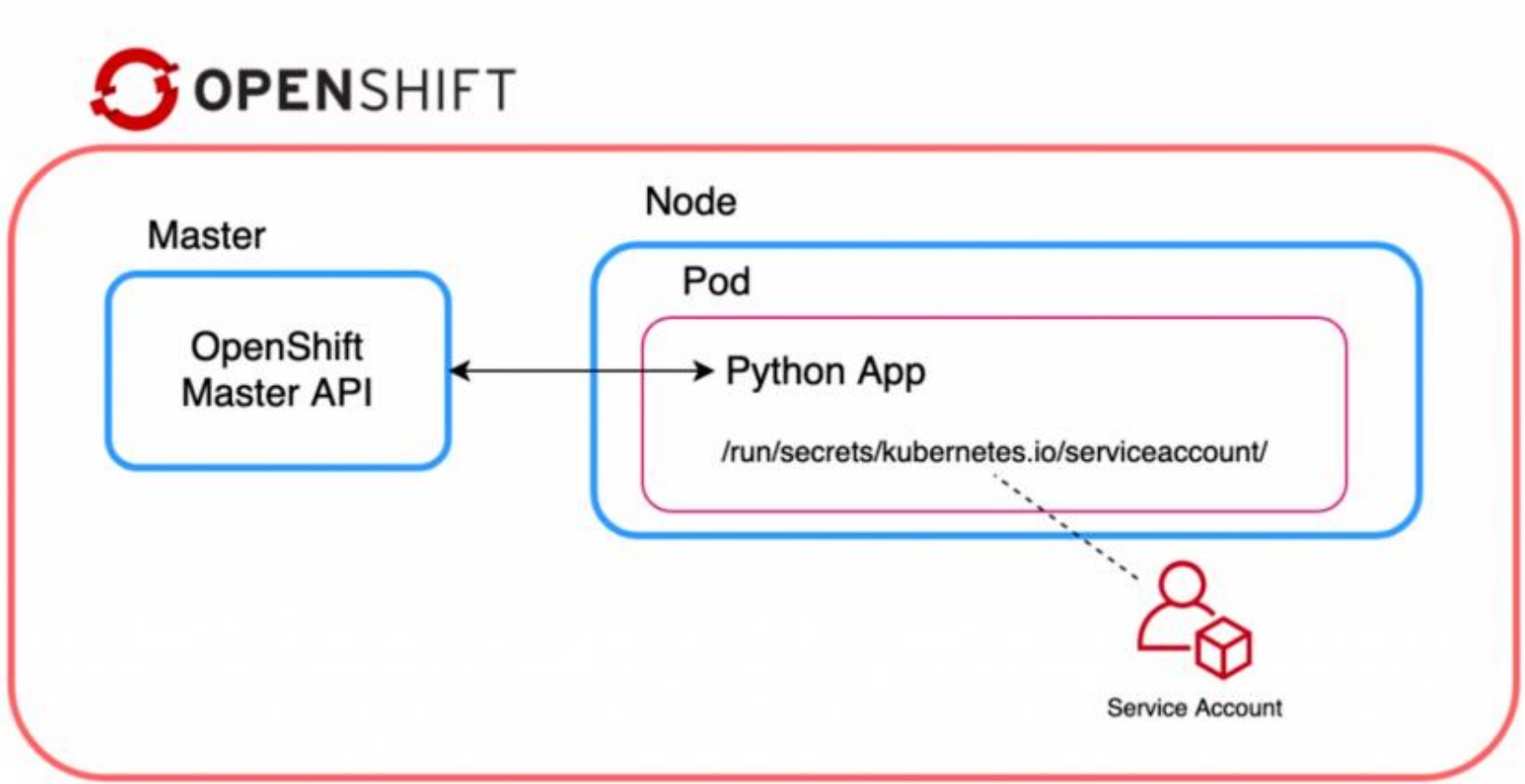
오픈 시프트

오픈 시프트 아키텍처

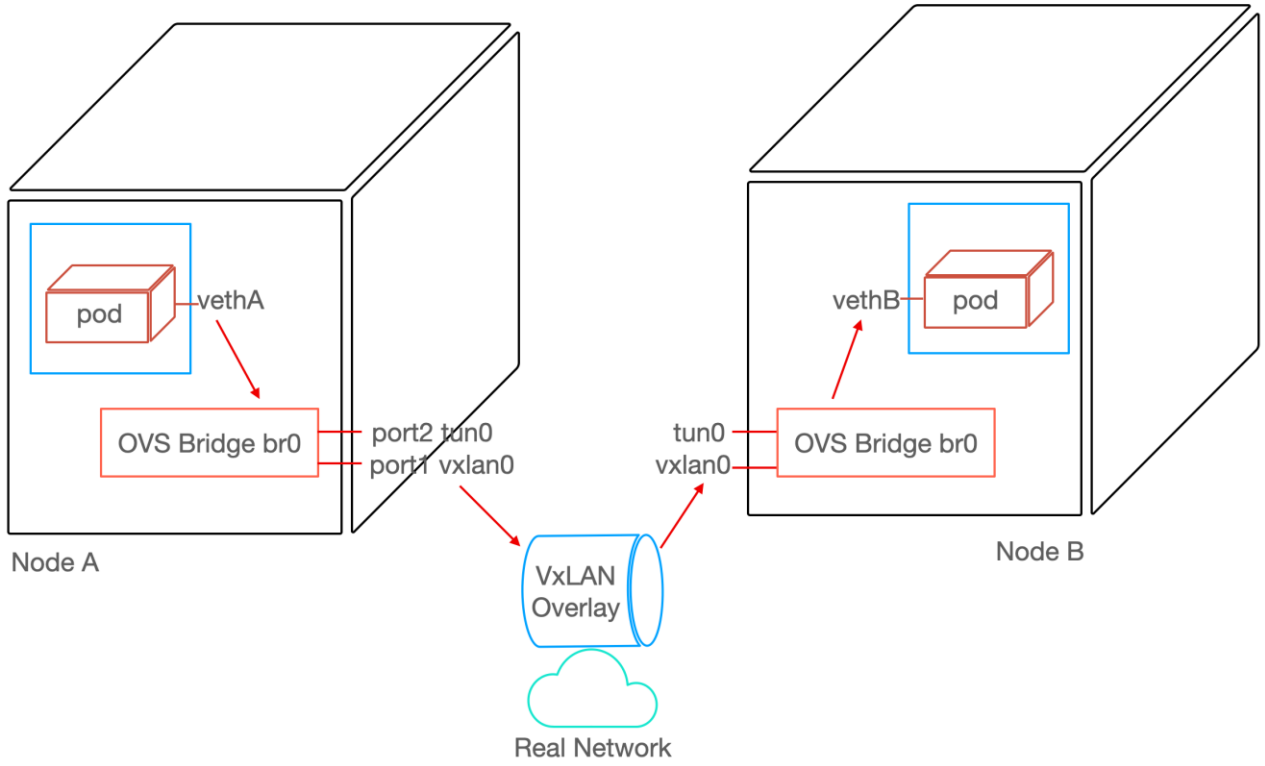
OCP


Pod-1
Pod-2
Pod-3
Pod-4
...
Pod-n
Container Runtime
cri-o
CSI
Red Hat
OpenShift Container Platform
MultusCNI
OpenShift
SDN
CNI
Node Controller
Red Hat
Enterprise Linux
Storage I/O
Network I/O
Network I/O
Software
Hardware
Virtualization
(Ultima IO cards)
Remote Storage
Storage Virtualization
Network Virtualization
Local Storage
x86 Based Hardware Platform
Hardware
Control
Plane
Storage
Plane
Data
Plane

Pod


OPENSHIFT
Master
OpenShift
Master API
Node
Pod
Python App
/run/secrets/kubernetes.io/serviceaccount/
Service Account

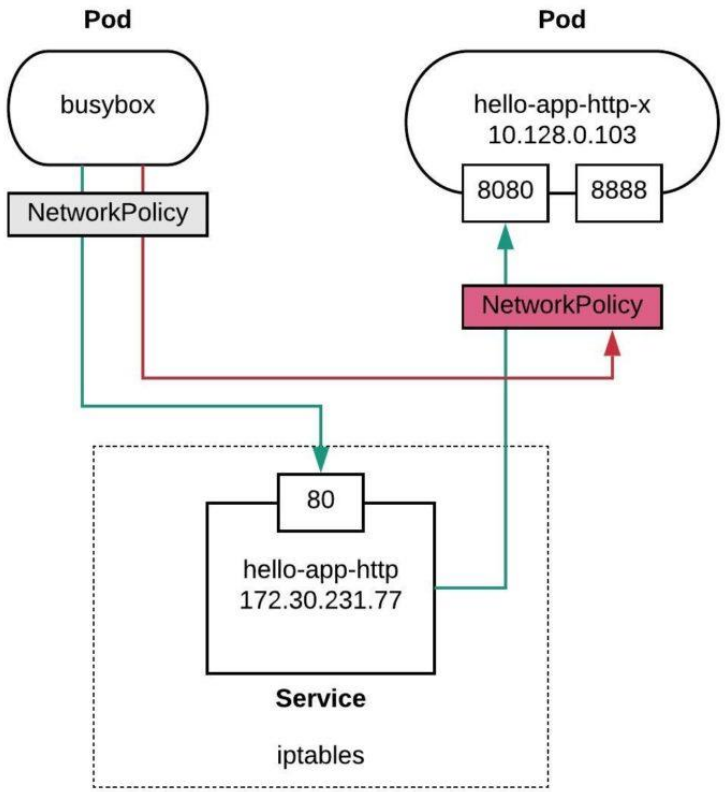
Pod



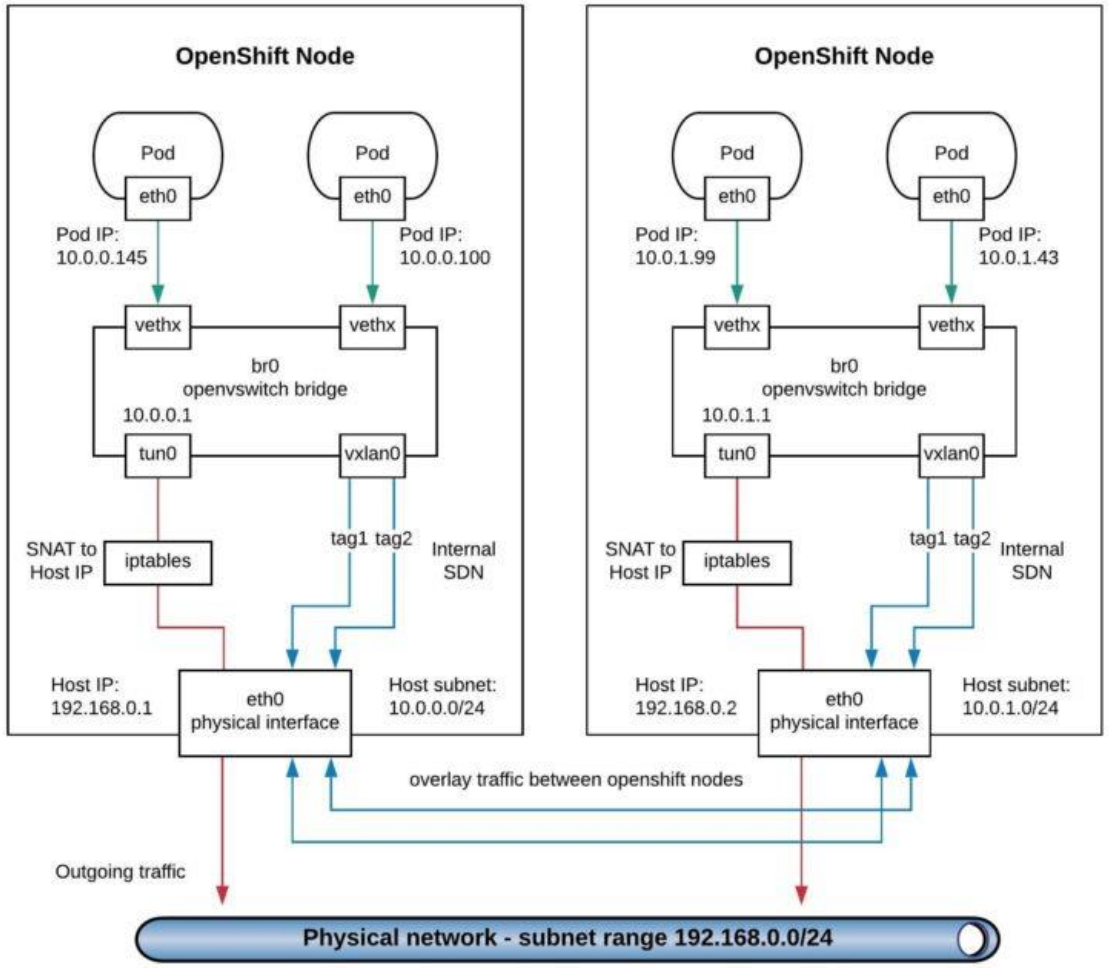
Traffic Flow

eth0 - vethA - br0 - vxlan0 - network - vxlan0 - br0 - vethB - eth0

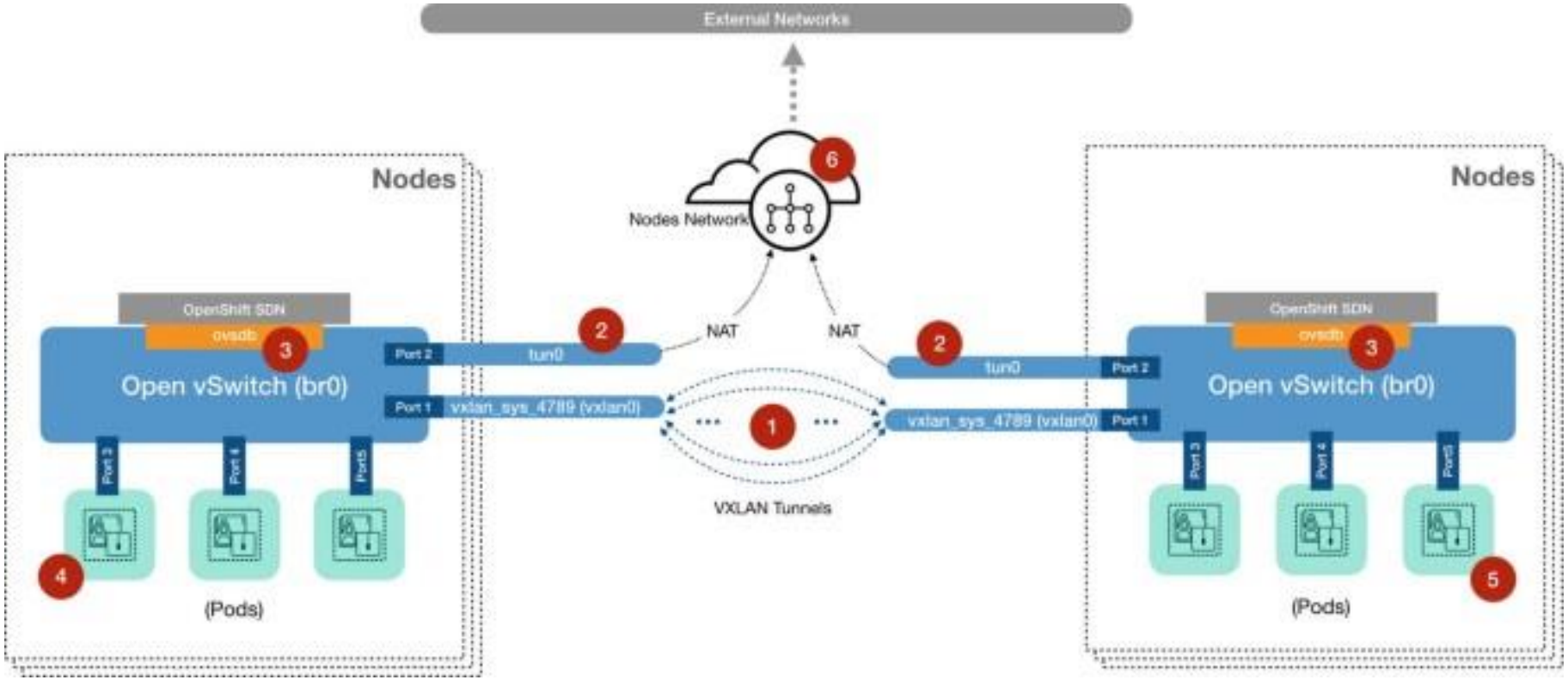
Pod


Pod
Pod
busybox
hello-app-http-x
10.128.0.103
8080
8888
NetworkPolicy
NetworkPolicy
80
hello-app-http
172.30.231.77
Service
iptables

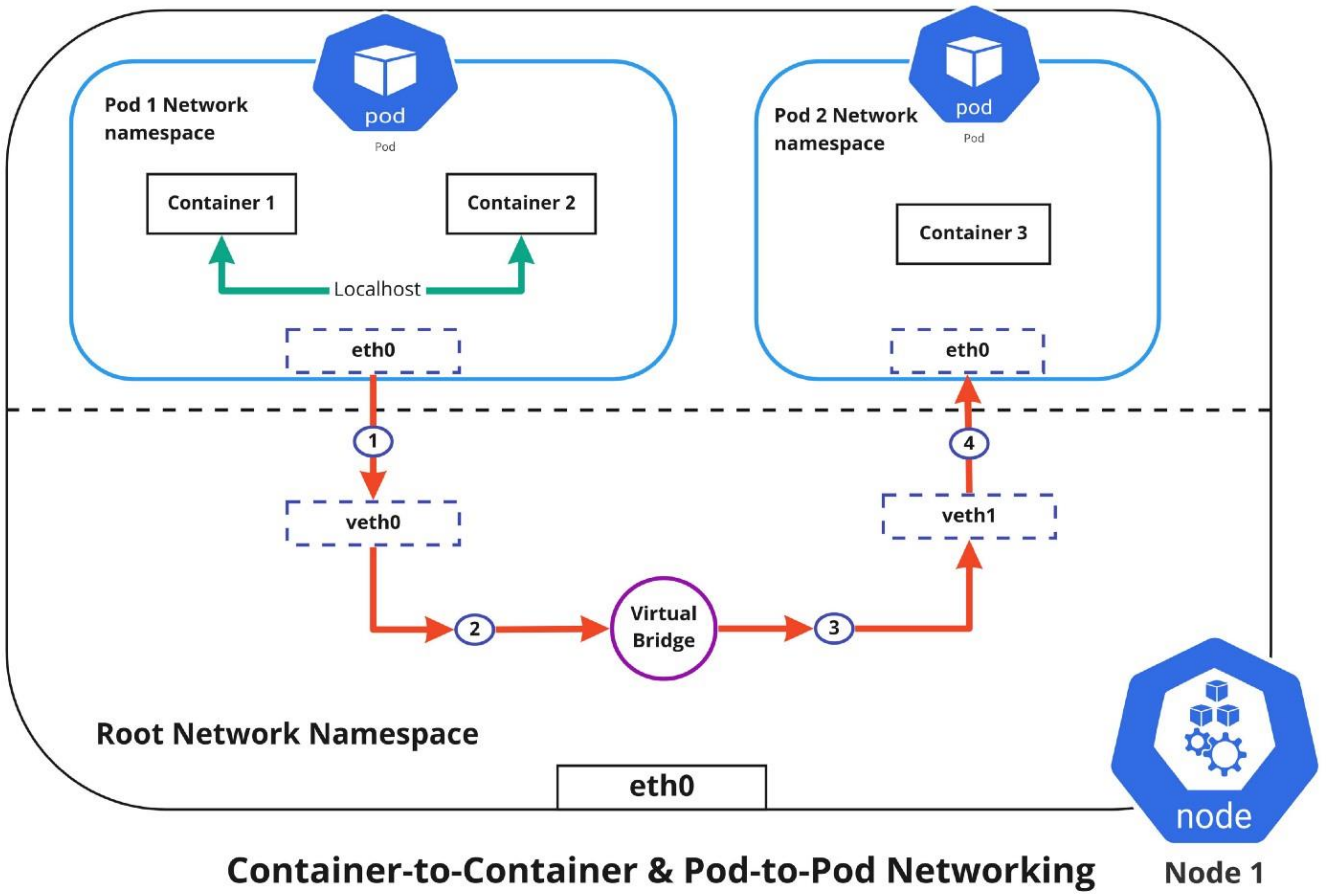
Pod


OpenShift Node
Pod
eth0
Pod IP:
10.0.0.145
Pod
eth0
Pod IP:
10.0.0.100
vethx
vethx
br0
openvswitch bridge
10.0.0.1
tun0
vxlan0
SNAT to
Host IP
iptables
tag1 tag2
Internal
SDN
Host IP:
192.168.0.1
eth0
physical interface
Host subnet:
10.0.0.0/24
OpenShift Node
Pod
eth0
Pod IP:
10.0.1.99
Pod
eth0
Pod IP:
10.0.1.43
vethx
vethx
br0
openvswitch bridge
10.0.1.1
tun0
vxlan0
SNAT to
Host IP
iptables
tag1 tag2
Internal
SDN
Host IP:
192.168.0.2
eth0
physical interface
Host subnet:
10.0.1.0/24
overlay traffic between openshift nodes
Outgoing traffic
Physical network - subnet range 192.168.0.0/24

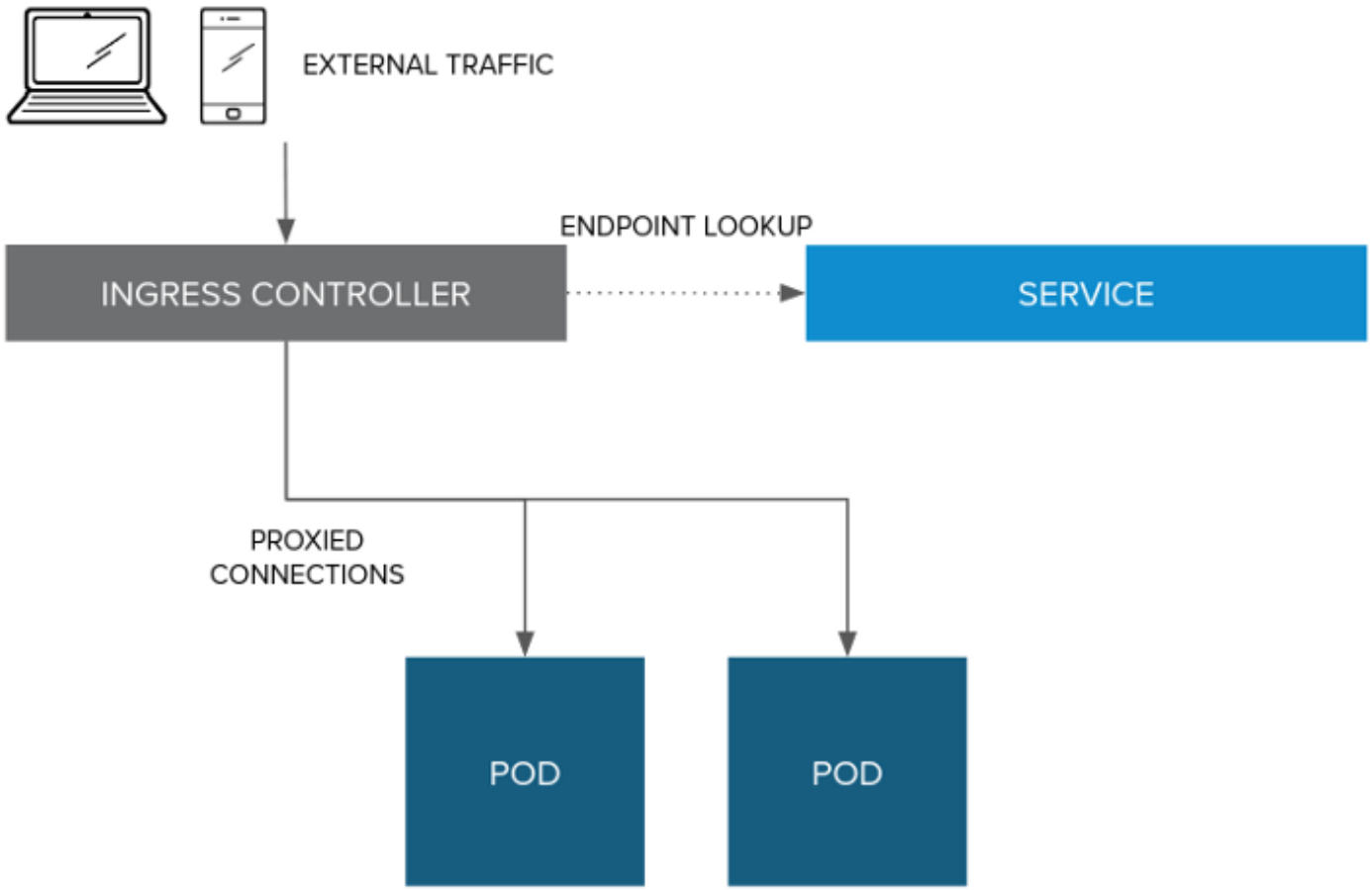
Network


External Networks
Nodes
Nodes Network
6
OpenShift SDN
ovsdb
3
Open vSwitch (br0)
Port 2
tun0
2
NAT
NAT
Port 1
vxlan_sys_4789 (vxlan0)
1
VXLAN Tunnels
Port 3
Port 4
Port5
4
(Pods)
5

Network


Pod 1 Network namespace
pod
Pod
Container 1
Container 2
Localhost
eth0
Pod 2 Network namespace
pod
Pod
Container 3
eth0
1
4
veth0
veth1
2
Virtual Bridge
3
Root Network Namespace
eth0
node
Container-to-Container & Pod-to-Pod Networking
Node 1

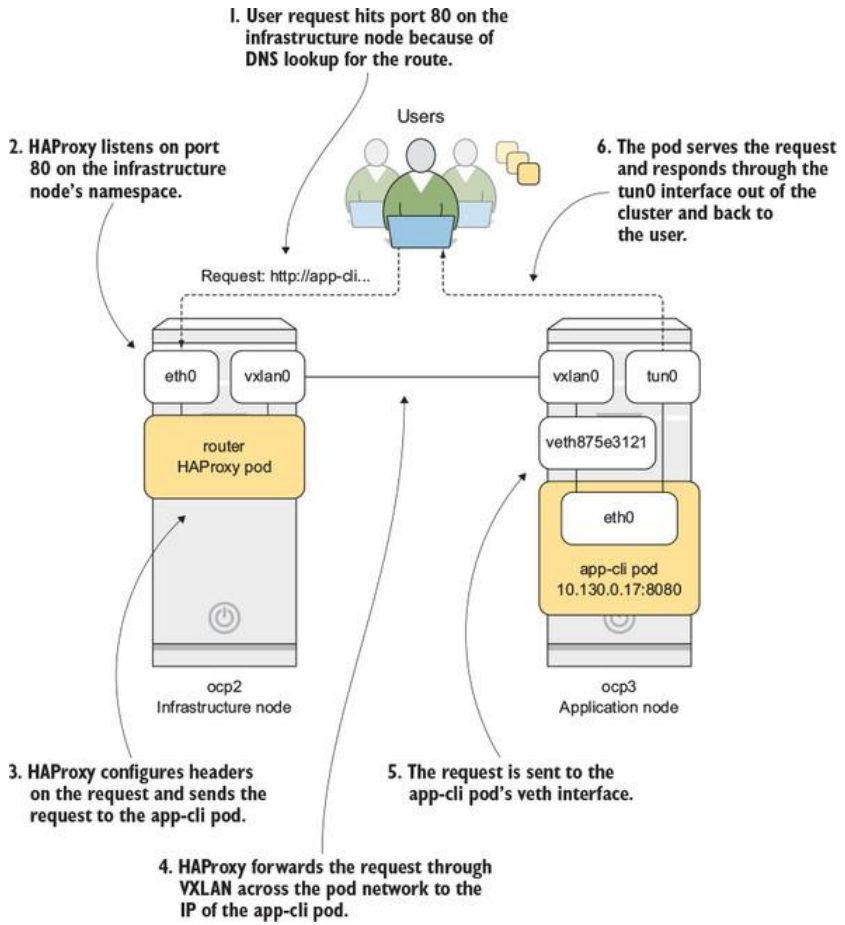
Network


EXTERNAL TRAFFIC
ENDPOINT LOOKUP
INGRESS CONTROLLER
SERVICE
PROXIED
CONNECTIONS
POD
POD

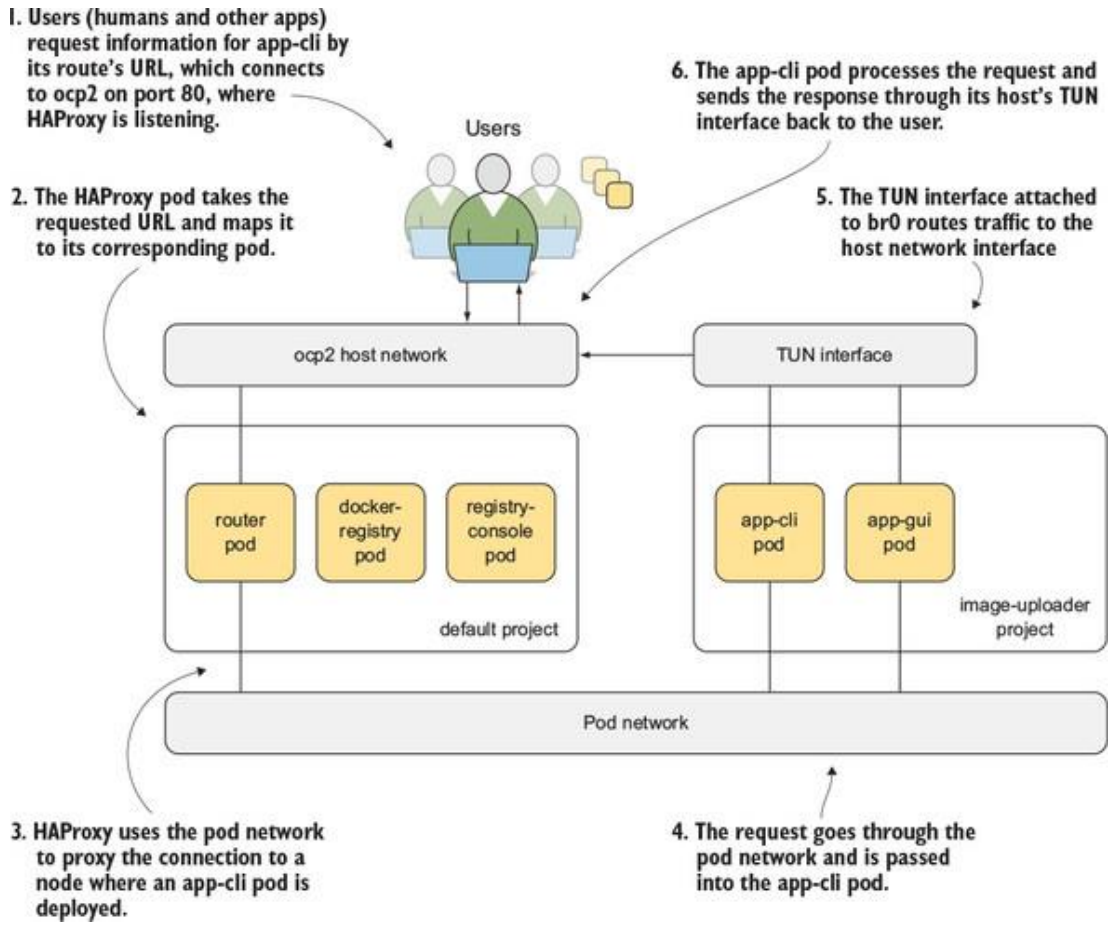
Network

Feature	Ingress on OpenShift	Route on OpenShift
Standard Kubernetes object	X	
External access to services	X	X
Persistent (sticky) sessions	X	X
Load-balancing strategies (e.g. round robin)	X	X
Rate-limit and throttling	X	X
IP whitelisting	X	X
TLS edge termination for improved security	X	X
TLS re-encryption for improved security		X
TLS passthrough for improved security		X
Multiple weighted backends (split traffic)		X
Generated pattern-based hostnames		X
Wildcard domains		X

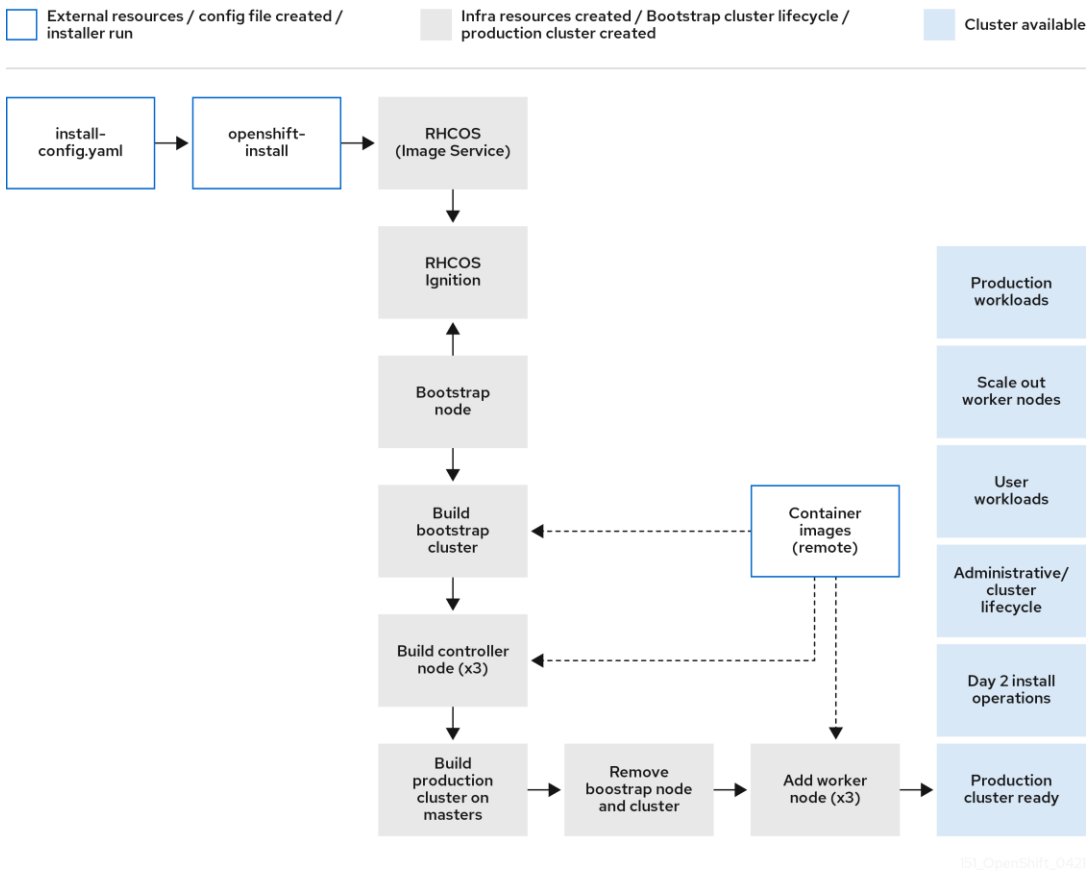
Network


1. User request hits port 80 on the infrastructure node because of DNS lookup for the route.
Users
2. HAProxy listens on port 80 on the infrastructure node's namespace.
6. The pod serves the request and responds through the tun0 interface out of the cluster and back to the user.
Request: http://app-cli...
eth0
vxlan0
router
HAProxy pod
ocp2
Infrastructure node
vxlan0
tun0
veth875e3121
eth0
app-cli pod
10.130.0.17:8080
ocp3
Application node
3. HAProxy configures headers on the request and sends the request to the app-cli pod.
5. The request is sent to the app-cli pod's veth interface.
4. HAProxy forwards the request through VXLAN across the pod network to the IP of the app-cli pod.

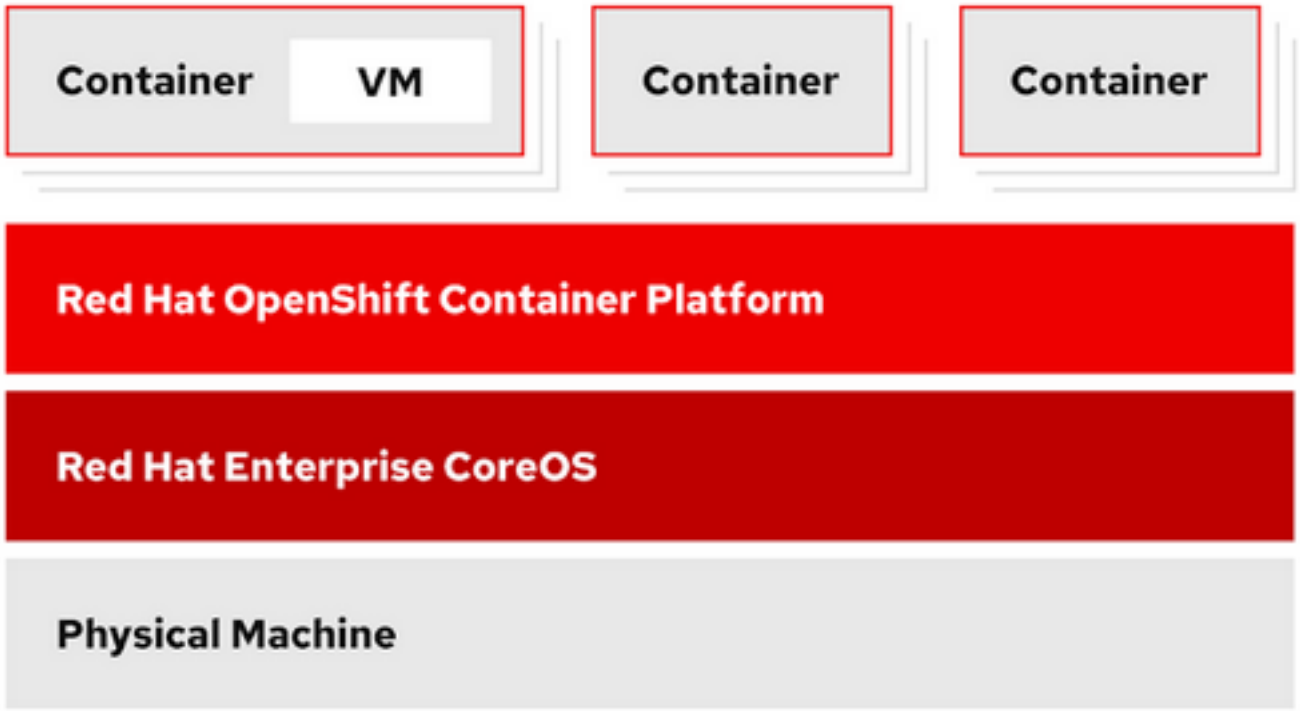
Network


1. Users (humans and other apps) request information for app-cli by its route's URL, which connects to ocp2 on port 80, where HAProxy is listening.
Users
6. The app-cli pod processes the request and sends the response through its host's TUN interface back to the user.
2. The HAProxy pod takes the requested URL and maps it to its corresponding pod.
5. The TUN interface attached to br0 routes traffic to the host network interface
ocp2 host network
TUN interface
router pod
docker-registry pod
registry-console pod
default project
app-cli pod
app-gui pod
image-uploader project
Pod network
3. HAProxy uses the pod network to proxy the connection to a node where an app-cli pod is deployed.
4. The request goes through the pod network and is passed into the app-cli pod.

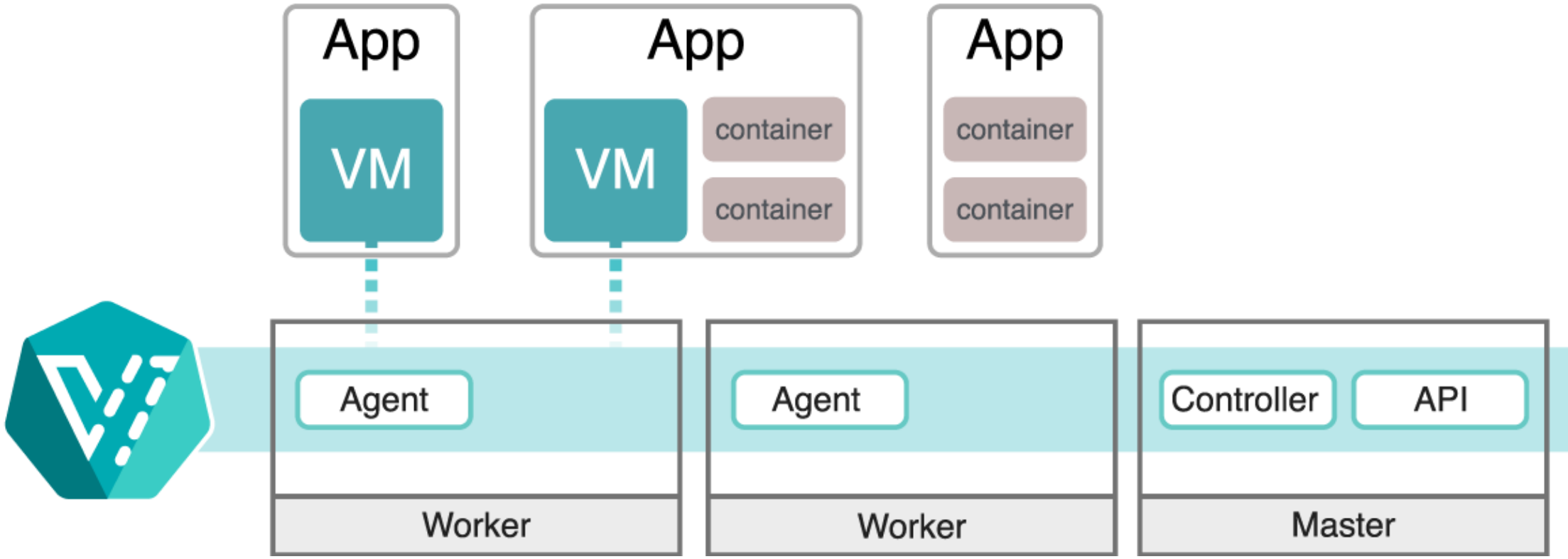
RHOCP


External resources / config file created / installer run
Infra resources created / Bootstrap cluster lifecycle / production cluster created
Cluster available
install-config.yaml
openshift-install
RHCOS (Image Service)
RHCOS Ignition
Bootstrap node
Build bootstrap cluster
Container images (remote)
Build controller node (x3)
Build production cluster on masters
Remove boostrap node and cluster
Add worker node (x3)
Production cluster ready
Production workloads
Scale out worker nodes
User workloads
Administrative/ cluster lifecycle
Day 2 install operations

RHOCP


Container
VM
Container
Container
Red Hat OpenShift Container Platform
Red Hat Enterprise CoreOS
Physical Machine

Virtualization



Virtualization