Projetos Openstack: Cinder, Swift e Neutron

Aula Teórica nº6

2020/2021

Storage Concept

- **Ephemeral storage** – if only **Nova** is deployed, by default users do not have access to any form of persistent storage: the disks associated with VMs are ephemeral, meaning that they disappear when a virtual machine is terminated
- **Persistent storage** – the storage resource outlives any other resource and is always available, regardless of the state of a running instance

OpenStack clouds explicitly support three types of persistent storage: *Object Storage*, *Block Storage*, and *File-based storage*

Persistent Storage: Block

- Implemented in OpenStack by the CINDER project
- Because these volumes are persistent, they can be detached from one instance and re-attached to another instance and the data remains intact
- The Block Storage service supports multiple back ends in the form of drivers
- Users interact by attaching volumes to VM instances

Persistent Storage: Object

- Implemented in OpenStack by the SWIFT project
- Users access binary objects through a REST API
- Amazon S3 is a well-known example of an object storage system
- Used to archive or manage large datasets
- Additional benefits include:
  - OpenStack can store your VM images inside of an Object Storage system, as an alternative to storing the images on a file system
  - Better support for distributed deployments across multiple datacenters through support for asynchronous eventual consistency replication

Persistent Storage: File-based

- In multi-tenant OpenStack cloud environment, the Shared File Systems service (manila) provides a set of services for management of shared file systems
- The Shared File Systems service is persistent storage and can be mounted to any number of client machines
- You can mount a share and access a share from several hosts by several users at a time

Differences between storage types (1/3)

	Ephemeral storage	Block storage	Object storage	Shared File System storage
Application	Run operating system and scratch space	Add additional persistent storage to a virtual machine (VM)	Store data, including VM images	Add additional persistent storage to a virtual machine
Accessed through…	A file system	A block device that can be partitioned, formatted, and mounted (such as, /dev/vdc)	The REST API	A Shared File Systems service share that can be partitioned, formatted and mounted (such as /dev/vdc)
Accessible from…	Within a VM	Within a VM	Anywhere	Within a VM

Differences between storage types (2/3)

	Ephemeral storage	Block storage	Object storage	Shared File System storage
Managed by…	OpenStack Compute (nova)	OpenStack Block Storage (cinder)	OpenStack Object Storage (swift)	OpenStack Shared File System Storage (manila)
Persists until…	VM is terminated	Deleted by user	Deleted by user	Deleted by user
Sizing determined by…	Administrator configuration of size settings, known as *flavors*	User specification in initial request	Amount of available physical storage	•User specification in initial request •Requests for extension •Available user-level quotes •Limitations applied by Administrator

Differences between storage types (3/3)

	Ephemeral storage	Block storage	Object storage	Shared File System storage
Encryption configuration	Parameter in nova.conf	Admin establishing encrypted volume type, then user selecting encrypted volume	Not yet available	Shared File Systems service does not apply any additional encryption above what the share's back-end storage provides
Example of typical usage…	10 GB first disk, 30 GB second disk	1 TB disk	10s of TBs of dataset storage	Depends on the size of back-end storage specified when a share was being created. In case of thin provisioning it can be partial space reservation

Storage Back-end Technologies

Commodity Storage	Proprietary Solutions
RADOS Block Devices (Ceph)	IBM
LVM	NetApp
Gluster	HP

And others like iSCSI, NFS, Sheepdog and ZFS…

Cinder

OpenStack Block Storage

- OpenStack block storage component is codenamed as Cinder
- Spun off from Nova-Volume, Cinder was born during Folsom development cycle (2012)
- It virtualizes pools of block storage devices and provides end users with a self service API to request and consume the resources

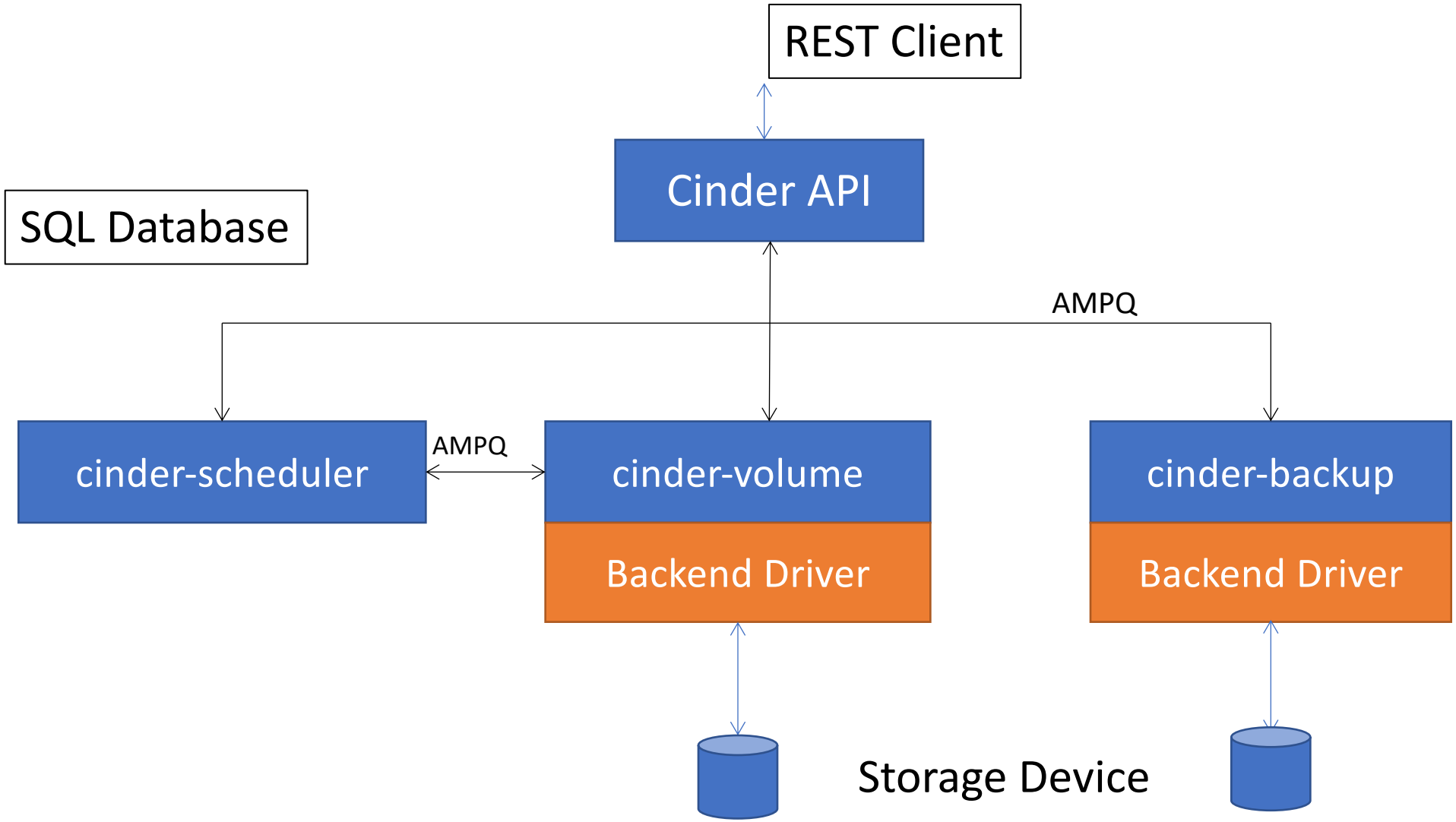
Cinder Overview

- Data volumes that are attached to the VM instances
- Boot from volume
- Volumes have a lifecycle independent of VM instances
  - Cinder: create volume

    cinder create --display-name testvolume 1
  - Nova: attach volume to instance

    nova volume-attach <Nova_ID> <Cinder_ID> auto

Cinder Architecture



Cinder components

- **cinder-api**
  - Is a WSGI app that authenticates and routes requests throughout the Block Storage system
  - Supports the OpenStack API's only
- **cinder-scheduler**
  - Responsible for scheduling/routing requests to the appropriate volume service
  - The default Filter Scheduler enables filtering on attributes like Capacity, Availability Zone, Volume Types and Capabilities as well as custom filters
- **cinder-volume**
  - Responsible for managing Block Storage devices, specifically the back-end devices themselves
- **cinder-backup**
  - Provides a means to back up a Cinder Volume to OpenStack Object Store (SWIFT)

Cinder API

- Volume create / delete / list / show
- Create volume from volume, image, snapshot
- Snapshot create / delete / list / show
- Volume attach/detach functions are called by Nova
- Others
  - Volume Types
  - Quotas
  - Backups

Cinder Scheduler

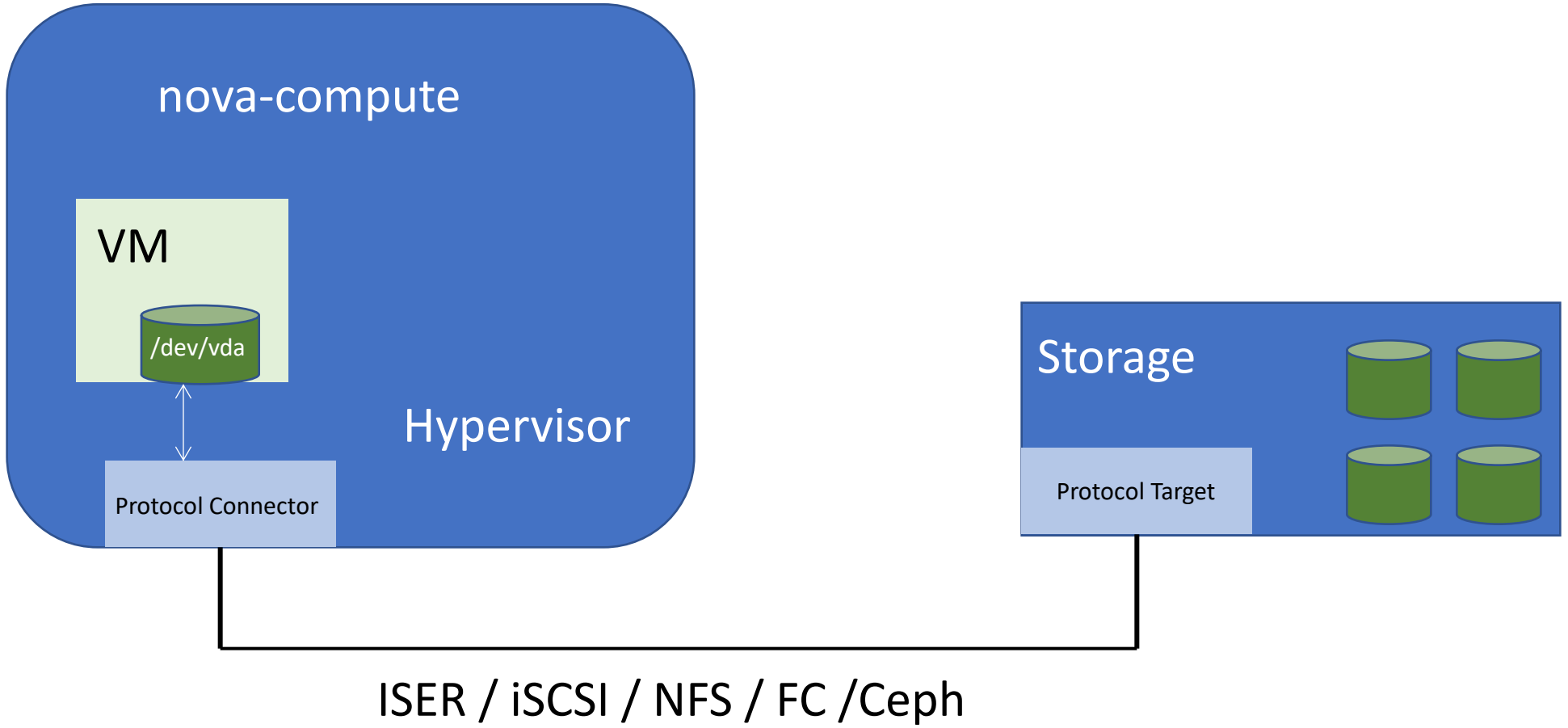
- Chooses which backend a new volume should be placed on
- Configurable plugins for schedulers
- Filter scheduler has plugins for filter and weights
- Filter Scheduler
  - *Starts with list of all backends*
  - *Filters according to capabilities*
  - *Sorts according to weights*
  - *Returns the best candidate*

Drivers

- Contain code to interact with a specific backend storage
- Three main types:
  - Local machine, where cinder-volume is installed on the storage to be used. For example: LVM
  - Vendor Storage controllers, where a proprietary API is used to communicate requests (eg NetApp or EMC)
  - Distributed file systems

https://wiki.OpenStack.org/wiki/CinderSupportMatrix

Data Flow


nova-compute
VM
/dev/vda
Hypervisor
Protocol Connector
Storage
Protocol Target
ISER / iSCSI / NFS / FC /Ceph

Swift

Object Storage Use Cases

- Developing high traffic websites; unstructured objects can be stored and retrieved dynamically with minimal latency
- Unstructured data
  - Media (images, music, video)
  - Web Content
  - Documents
  - Backups/Archives
- Archival and storage of structured and semi-structured data
  - Databases
  - Sensor data
  - Log files

OpenStack Object Storage

- OpenStack object storage component is codenamed as Swift
- Swift is a multi-tenant, highly scalable and durable object storage system designed to store large amounts of unstructured data
- Data accessed via RESTful HTTP API
- Swift is designed to be horizontally scalable – there is no single point-of-failure

Swift Requests

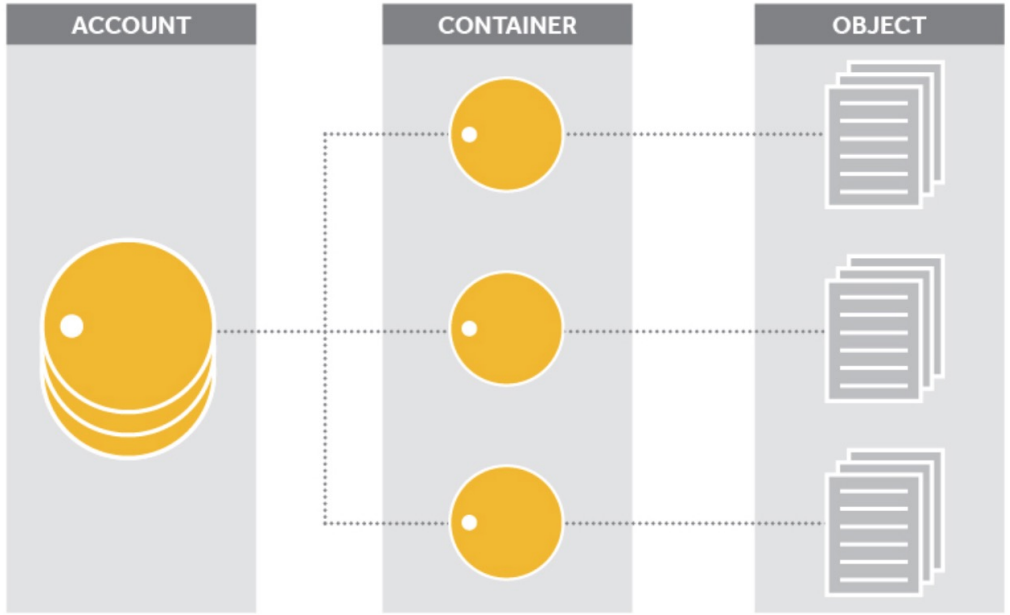
- All requests to Swift contain
  - HTTP verb (e.g., GET, PUT, DELETE)
  - Authentication information
  - Storage URL
  - Optional: any data or metadata to be written

- Example - GET
  https://swift.example.com/v1/account/container/object

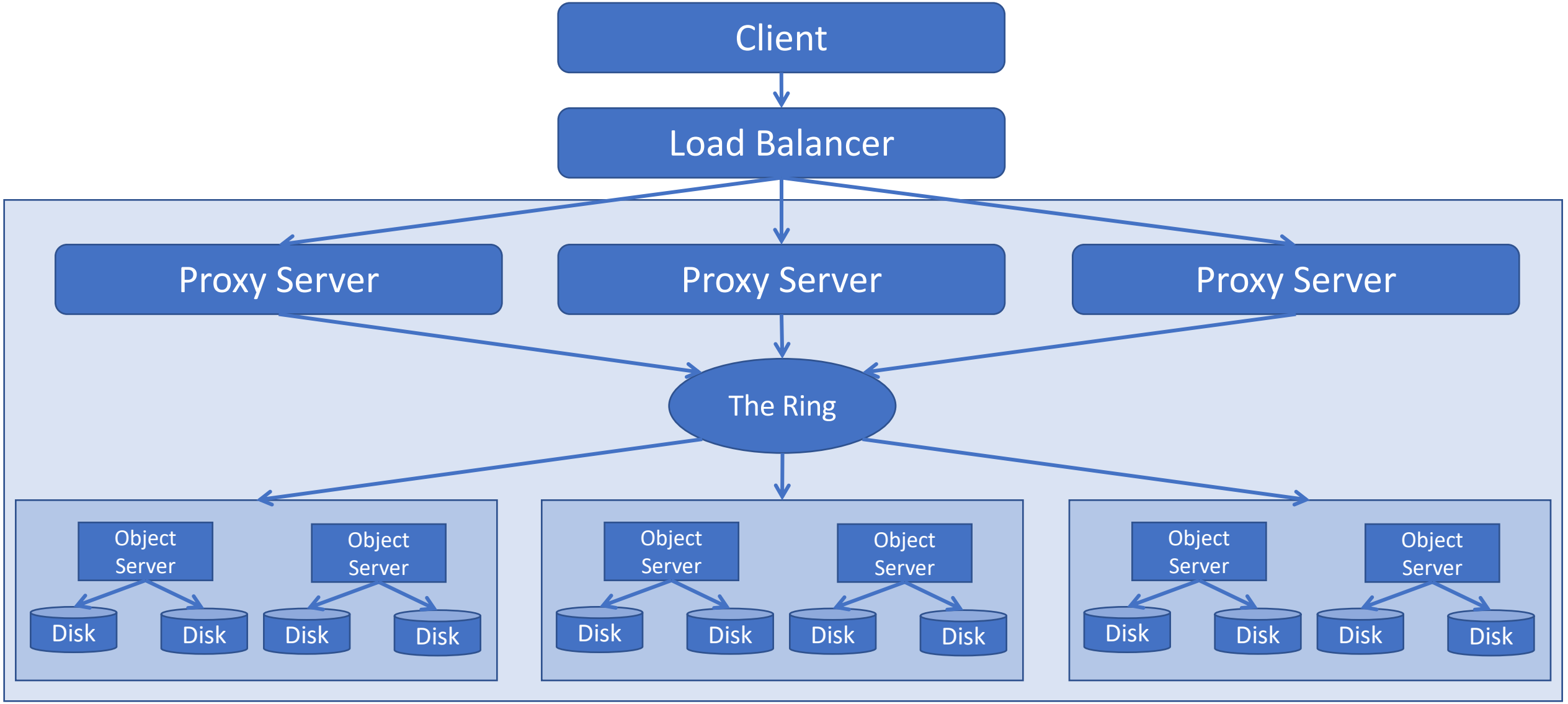
Deciphering Swift URL –

https://swift.example.com/v1/account/container/object

- /account
  - The account storage location is a uniquely named storage area that contains the metadata (descriptive information) about the account itself as well as the list of containers in the account
- /account/container
  - The container storage location is the user-defined storage area within an account where metadata about the container itself and the list of objects in the container will be stored.
- /account/container/object
  - The object storage location is where the data object and its metadata will be stored.



Swift Architecture



Swift Building Blocks

- **Proxy Servers:** Handles all incoming API requests
- **Rings:** Maps logical names of data to locations on particular disks
- **Zones:** Each Zone isolates data from other Zones. A failure in one Zone doesn't impact the rest of the cluster because data is replicated across the Zones

Swift Processes

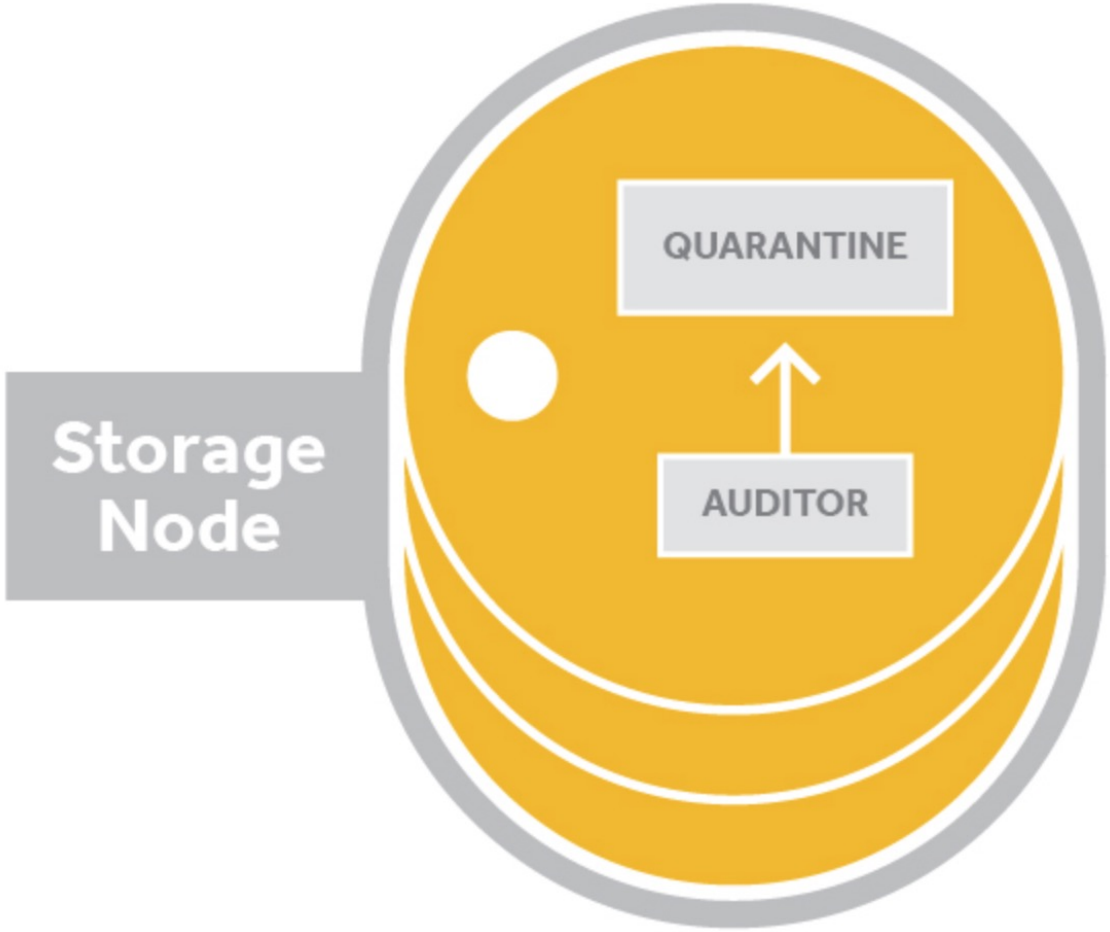
- Proxy Node
  - Interfaces with external clients
  - Share nothing architecture that can be scaled
  - Looks up multiple locations (3 copies by default)
- Account
- Container
- Object

Storage Node

  - Account and Container processes handle metadata info relating to account or container
  - Object process handles the actual storage of the data (along with metadata info)

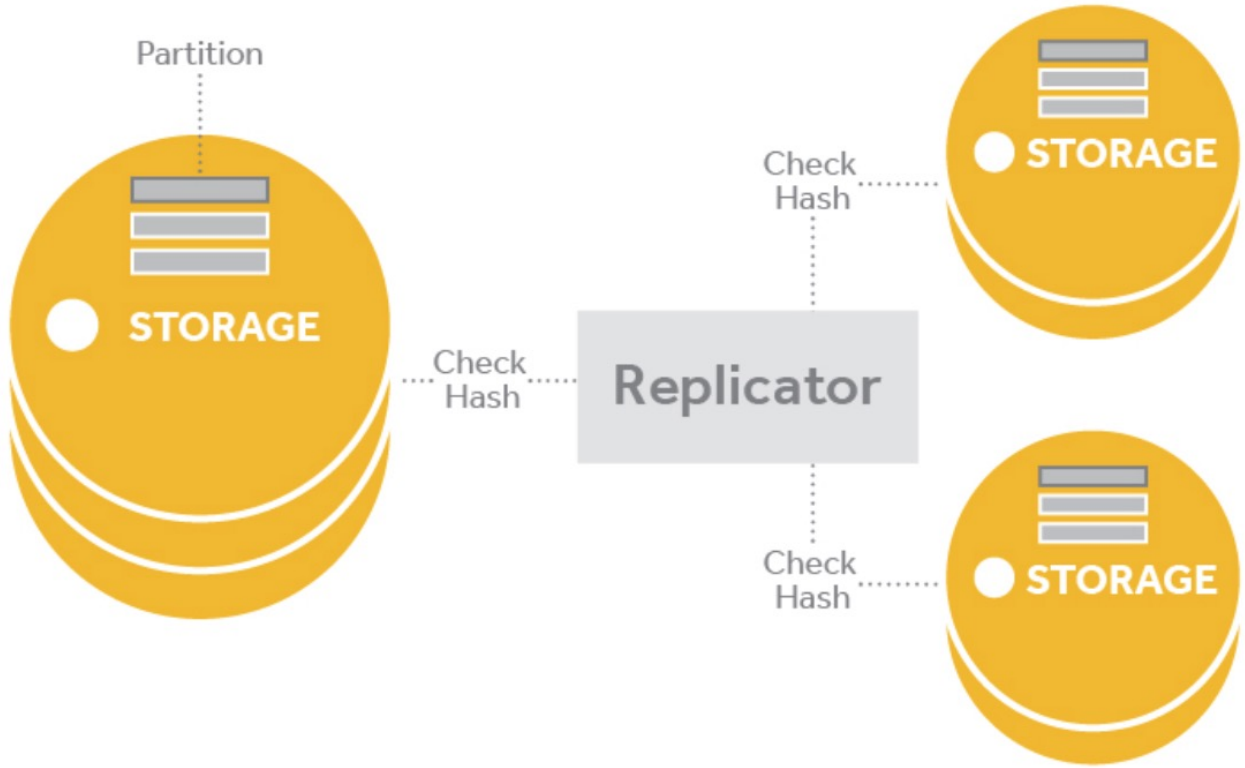
Consistency Services – Auditors

- Run on every storage node and continually scan the disks to ensure that the data stored on disk has not suffered any bit-rot or file system corruption – account, container, object auditors.
- If an error is found, the auditor moves the corrupted object to a quarantine area



Consistency Services – Replicators

- A replicator will continuously examine its local node and compare the accounts, containers, or objects against the copies on other nodes in the cluster.
- If one of other nodes has an old or missing copy, then the replicator will send a copy of its local data out to that node
- Object deletion starts by creating a zero-byte tombstone file that is the latest version of the object. This version is then replicated to the other nodes and the object is removed from the entire system.



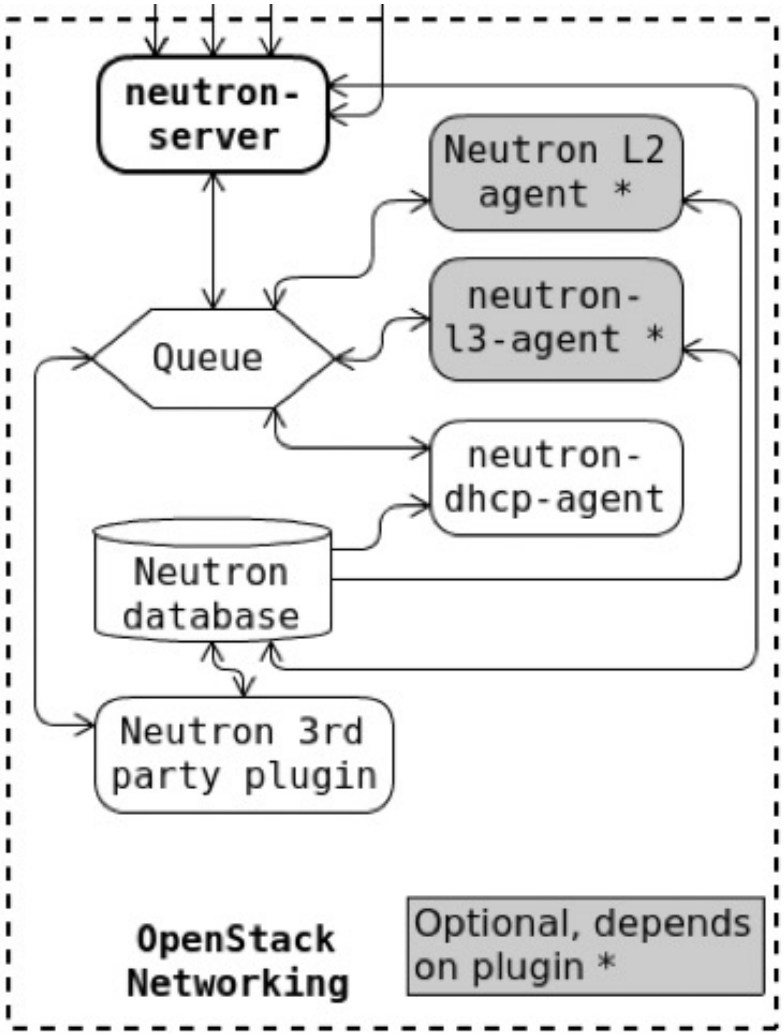
Neutron

Neutron Overview

- Allows users to set up and define network connectivity and addressing in the cloud
- Handles the creation and management of a virtual networking infrastructure, including networks, switches, subnets, and routers for devices managed by Nova
- Advanced services such as firewalls or virtual private networks (VPNs) can also be used
- API interface

Neutron Architecture

- Neutron-server
- Database for persistent storage
- Any number of plug-in agents, which provide other services such as interfacing with native Linux networking mechanisms, external devices, or SDN controllers
- Entirely standalone and can be deployed to a dedicated host (alternatively can be added to the controller node used for compute)
- Integrates with Keystone, Nova and Horizon (and other non core services like Heat)



Namespace

- A namespace is a way of scoping a particular set of identifiers
- Using a namespace, you can use the same identifier multiple times in different namespaces
- You can also restrict an identifier set visible to particular processes
- Linux provides namespaces for networking and processes, among other things
- For example, if a process is running within a process namespace, it can only see and communicate with other processes in the same namespace
- So, if a shell in a particular process namespace ran **ps waux**, it would only show the other processes in the same namespace

Linux Network Namespaces

- In a network namespace, the scoped ‘identifiers’ are network devices
- A given network device, such as eth0, exists in a particular namespace
- Each network namespace has its own routing table (enables multi-tenancy) and its own set of iptables (for both IPv4 and IPv6), i.e., we can apply different security to flows with the same IP addressing in different namespaces, as well as different routing
- Any given Linux process runs in a particular network namespace

Network Address Translation (NAT)

- In OpenStack deployments it is typically Linux servers that implement the NAT functionality
- These servers use the iptables software package to implement the NAT functionality
- *Source Network Address Translation* (SNAT), commonly used to enable hosts with *private addresses* to communicate with servers on the public Internet (also named PAT or NAT overload)
- *Destination Network Address Translation* (DNAT), where the IP address of the destination is modified used in OpenStack to route packets from instances to the metadata service (to obtain metadata such as SSH keys)
- *One-to-one NAT*, where a one-to-one mapping between private IP addresses and public IP addresses is maintained (to implement floating IP addresses)

Network Elements in Neutron

Network

- Isolated Layer 2 broadcast domain

Subnet

- Block of IP addresses (IPv4 or IPv6)
- One or many subnets per network possible

Port

- Connection point to attach a NIC of a virtual server to a virtual network
- Also describes the associated network configuration, such as the MAC and IP addresses to be used on that port

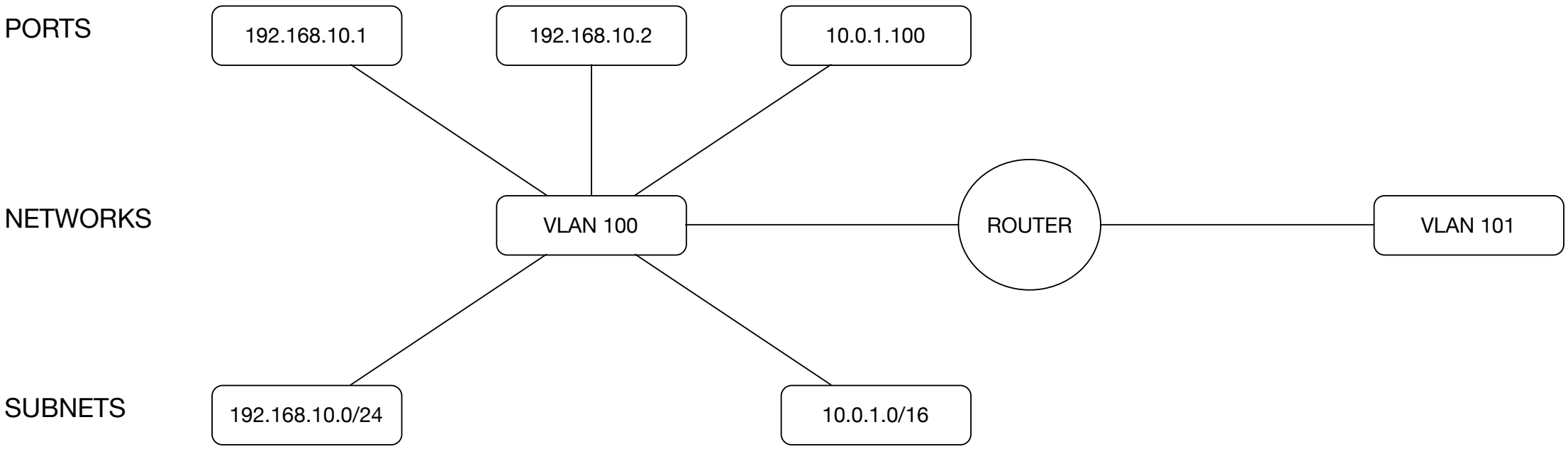
Router

- Entity that routes traffic between subnets and networks
- Providing other L3 capabilities such as NAT
- Neutron uses a layer-3 agent to manage routers via namespaces

Security Groups and Extensions

- Provide a container for vFW rules that control ingress and egress traffic at the port level
- Security group rules are stateful
- Extensions allow the introduction of new features in the API as well as the introduction of vendor specific functionality

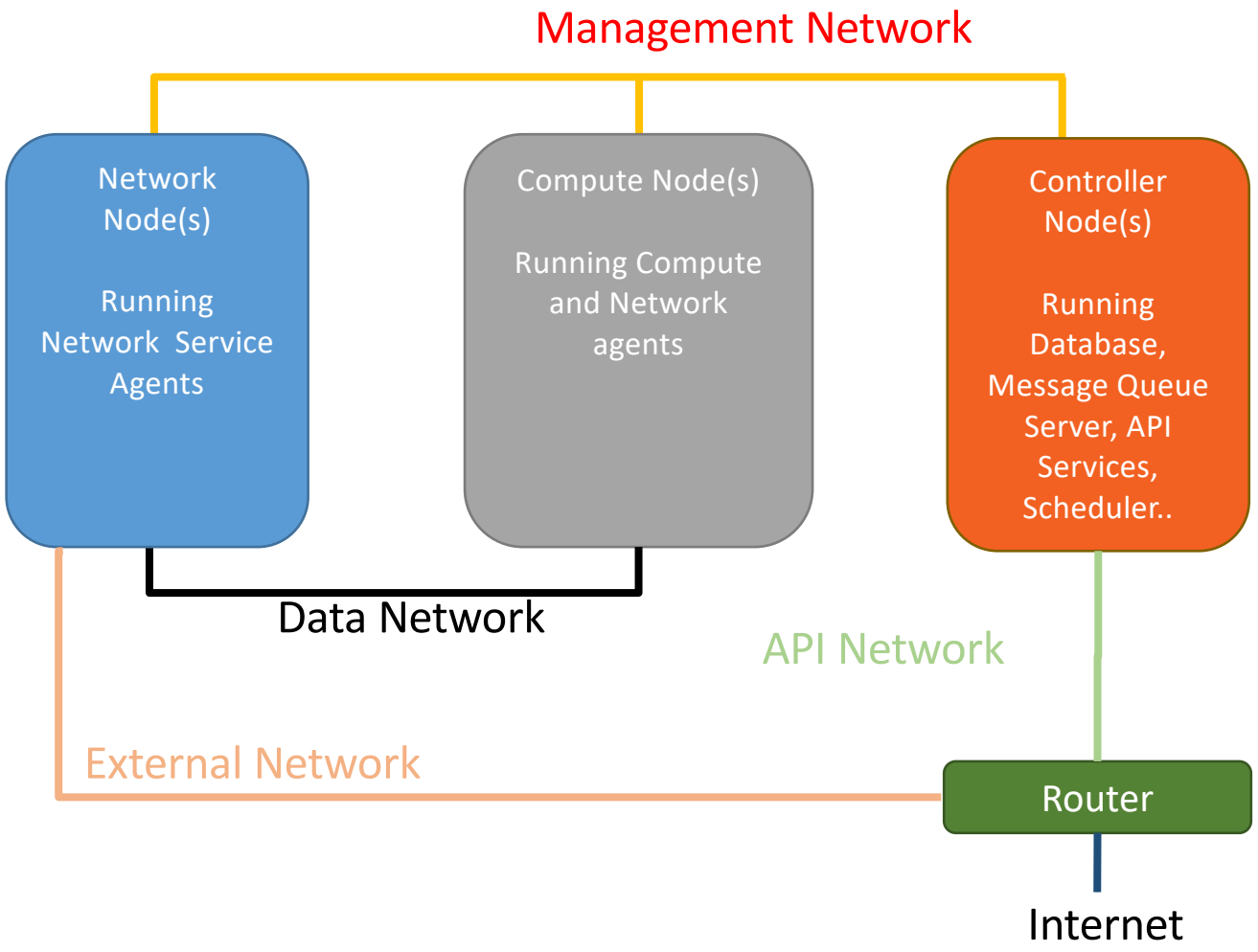
Network Elements in Neutron – An Example



Provider and Self-Service Networks

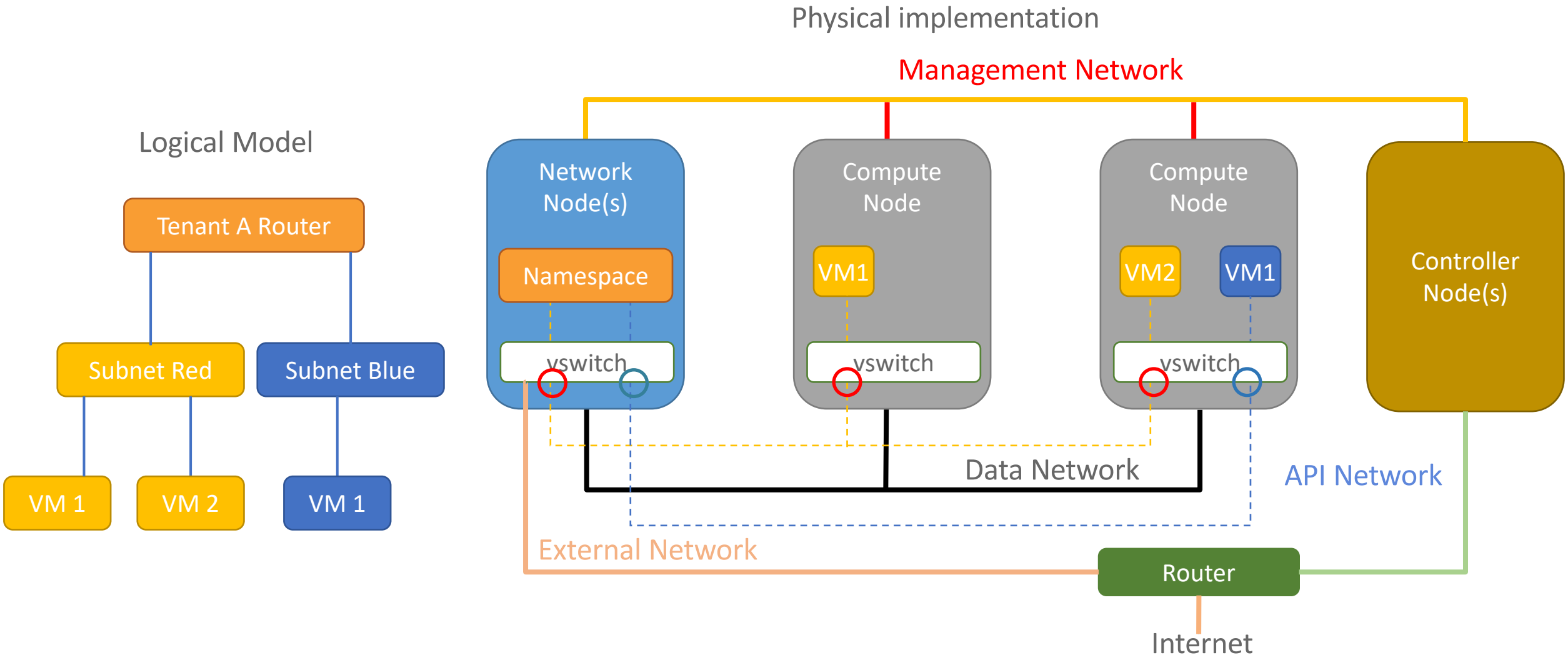
- We can create and configure networks and subnets and attach virtual devices to ports on these networks
- A project can have multiple private networks and we can choose their own IP addressing scheme
- **Provider Networks** connect instances to existing layer-2 networks in the data center, typically using VLAN (802.1q) tagging
- **Self-Service Networks** enable general (non-privileged) projects to manage networks without involving administrators. These networks are entirely virtual and require virtual routers to interact with provider and external networks such as the Internet
- Both typically provide DHCP and metadata services to instances

OpenStack Network Architecture



Network	Purpose	IP Address
Management Network	Used for internal communication between OpenStack Components	Reachable only within the data center
External Network	Used to provide VMs with Internet access	Reachable by anyone from the Internet
API Network	Exposes all OpenStack APIs, including the OpenStack Networking API, to tenants	Reachable to Tenants
Data Network	Used for VM data communication within the cloud deployment.	Reachable within the Tenant address space

Neutron Overview

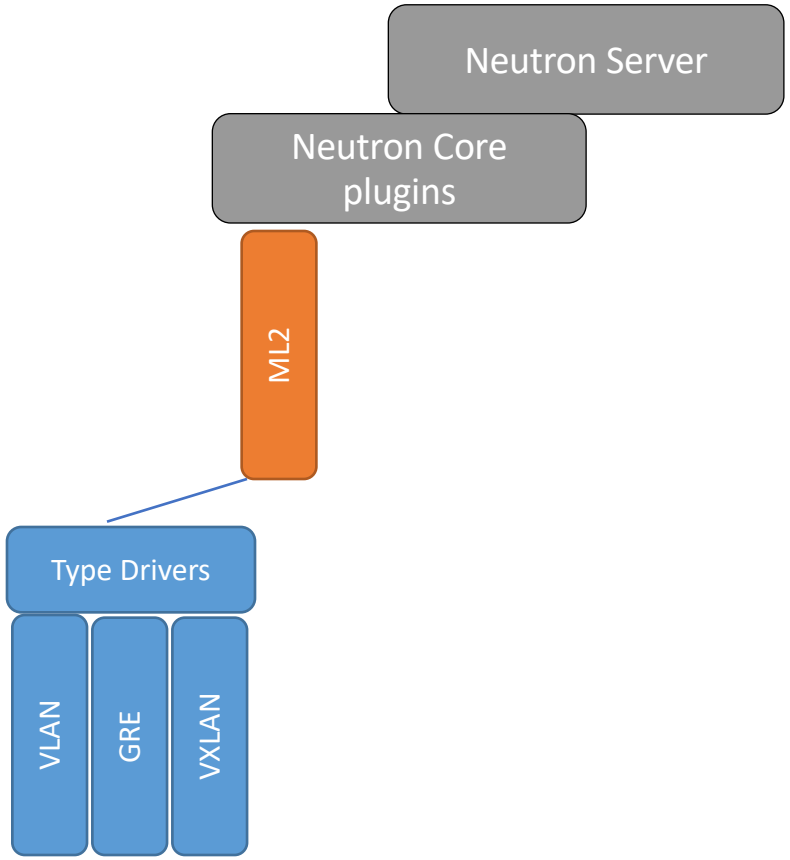


L3 Agent

- An API extension to allow administrators and tenants to create "routers" that connect to L2 networks.
- It uses the Linux IP stack and iptables to perform L3 forwarding and NAT.
- Defaults to using Linux Network namespaces to provide isolated forwarding contexts (VRF).
- Each router will have its own namespace with a name based on its UUID.

OpenStack Neutron Architecture

Neutron – ML2 Type Drivers

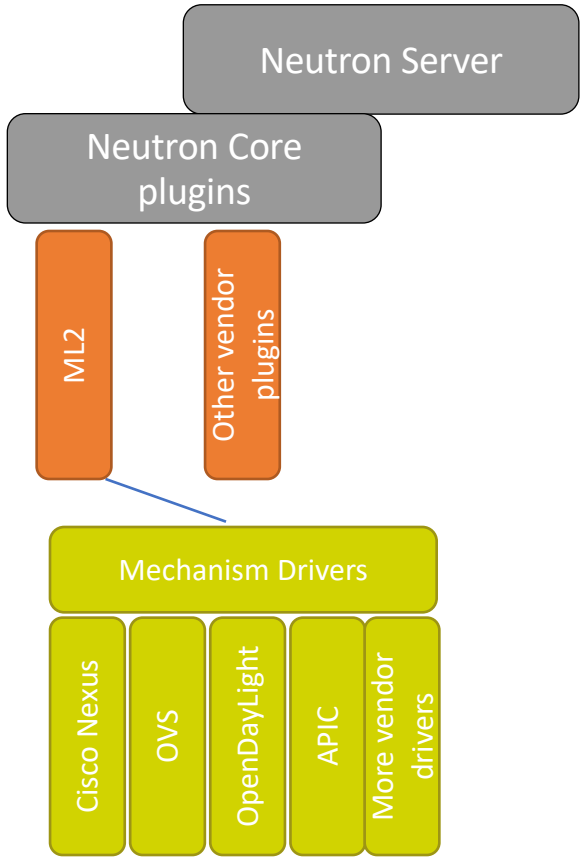


- Modular Layer 2 (ML2) architecture replaces legacy monolithic plugins
- Maintains type-specific network state
- Supported network types include local, flat, VLAN, GRE and VxLAN

- Virtual Local Area Networks (VLAN)
  - Layer 2 Technology with 12 bit address space (up to 4096 segments)
  - Defines Broadcast Domain
  - Legacy Technology introduced for departmental segmentation
- Virtual Extensible LANs (VxLAN)
  - Address space extended to 16 million isolated segments
  - Tunnel technology encapsulating IP packets into VxLAN specific header (42 bytes overhead)
  - Allows for L2-overlays network atop L3 endpoints using Virtual Tunnel Endpoints (VTEPs)
- Generic Routing Encapsulation (GRE)
  - Encapsulates any IP packets into point-to-point links
  - Complex and introduces large administrative overhead

OpenStack Neutron Architecture

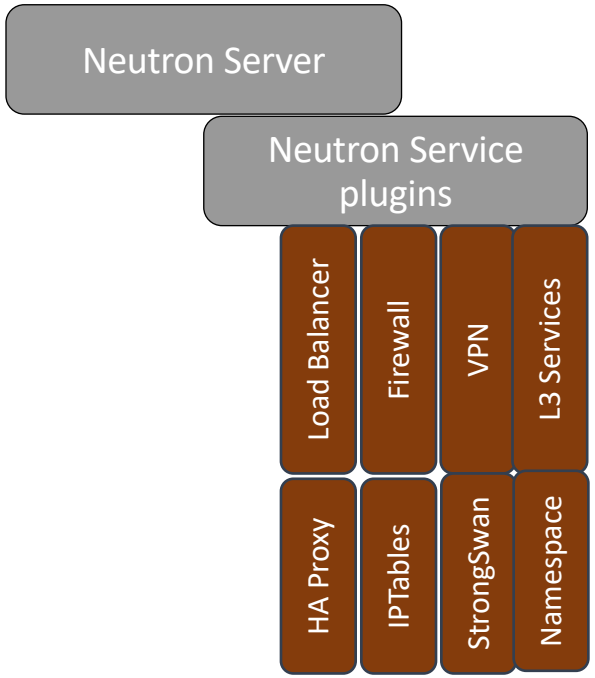
Neutron – Mechanism Driver



- Responsible for taking information supplied by TypeDrivers and ensuring it is properly applied given the specific networking mechanisms which have been enabled
- Current Mechanism Drivers:
  - Arista, Cisco Nexus, Cisco VTS, Hyper-V, L2 Population, LinuxBridge, Open vSwitch, Tail-F NCS
- Mechanism Drivers can work with many different technologies
  - Agent Based Mechanism Drivers (Hyper-V, LinuxBridge, OVS)
  - Controller based Mechanism Drivers (Tail-f NCS, VTS, OpenDaylight)
  - ToR Switch Mechanism Drivers (Arista and Cisco Nexus switches)

OpenStack Neutron Architecture

Neutron – Service Plugins

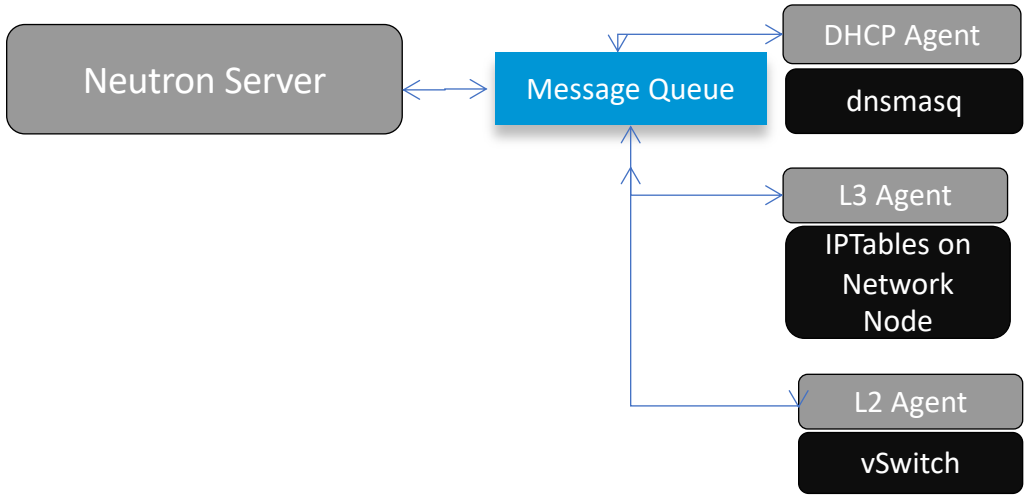


- Service Plugins offer extended services to OpenStack Networks:
  - Firewall capabilities
  - Data plane load balancers
  - Virtual Private Network
- Modularity provides means for extensibility

- Recent Releases introduced QoS and BGP as Service Plugins

OpenStack Neutron Architecture

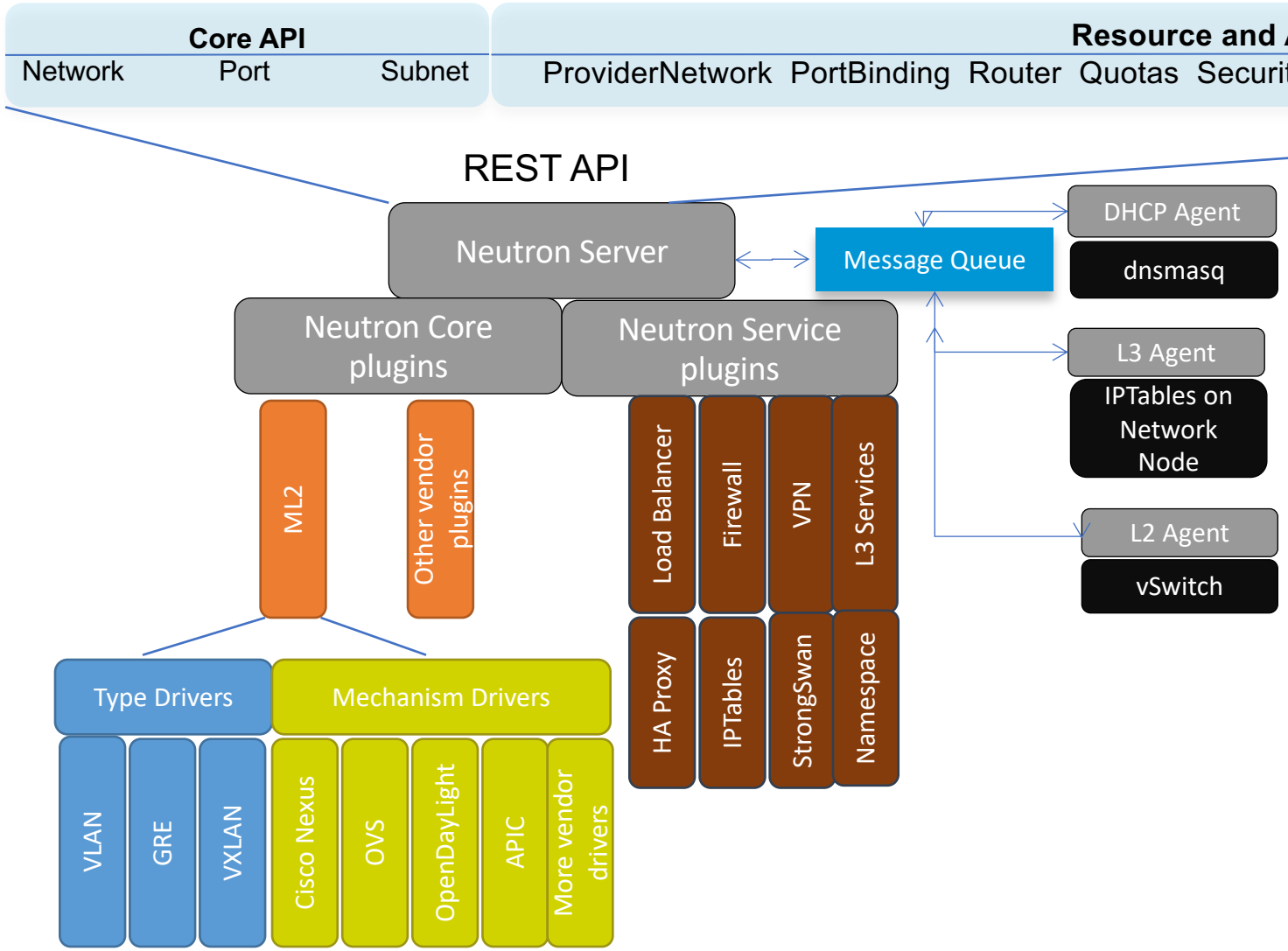
Neutron – Agents



- Neutron supports different agents that operate as exchange modules
- Reference Architecture includes L3 Agent(s), DHCP Agent(s), and L2 Agent(s)

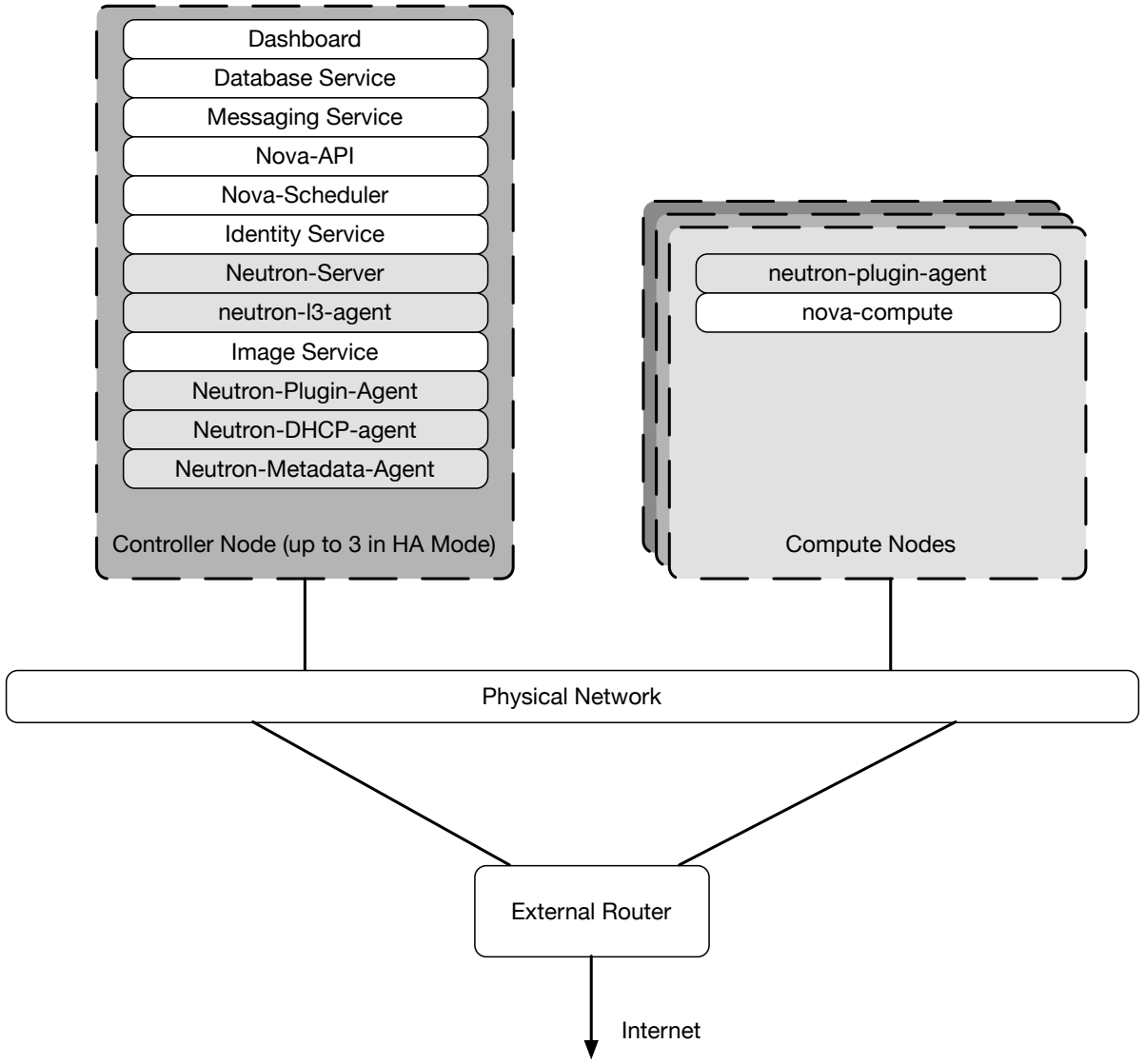
- L2 Agent (or plugin agent) – neutron-*-agent)
  - Runs on every hypervisor to perform local vSwitch configuration
  - Dependent on the plugin used (OVS, LinuxBridge, etc.)
- DHCP Agent
  - Runs either on controller or network node (depending on architecture)
  - Provides DHCP services to tenant networks
  - Separated by tenant and network using namespaces
- Layer 3 (L3) Agent
  - Provides Routing and NAT capabilities to tenant networks
  - Separated by tenant and network using namespaces

OpenStack Neutron Architecture – Summary



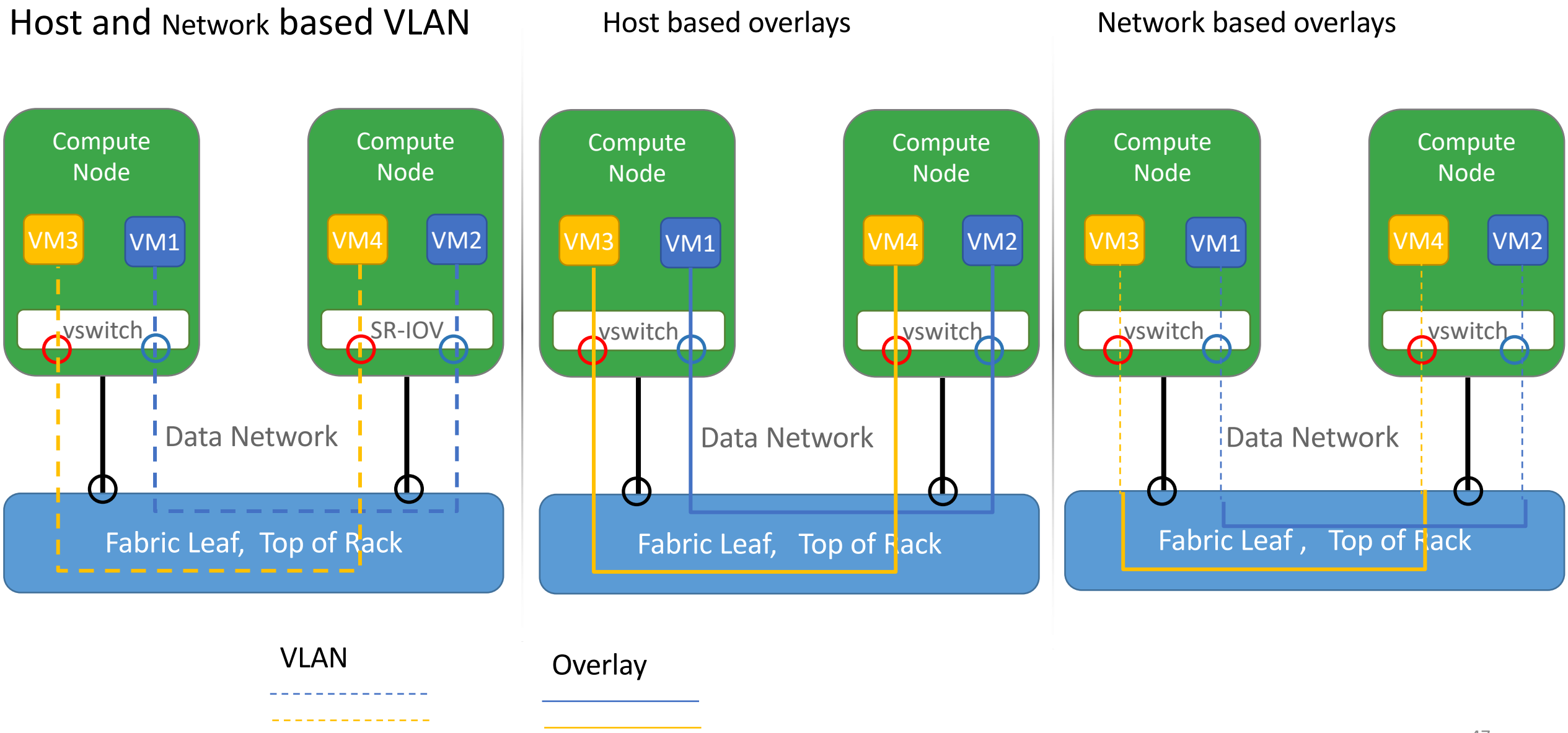
- Core + Extension REST API's
- Message Queue for communicating with Neutron Agents
- Core and Service Plugins
- Different vendor core plugins
- Different network technology support
- ML2 plugin with Type and Mechanism Drivers
- Service plugins with backend drivers

Neutron in an OpenStack Deployment

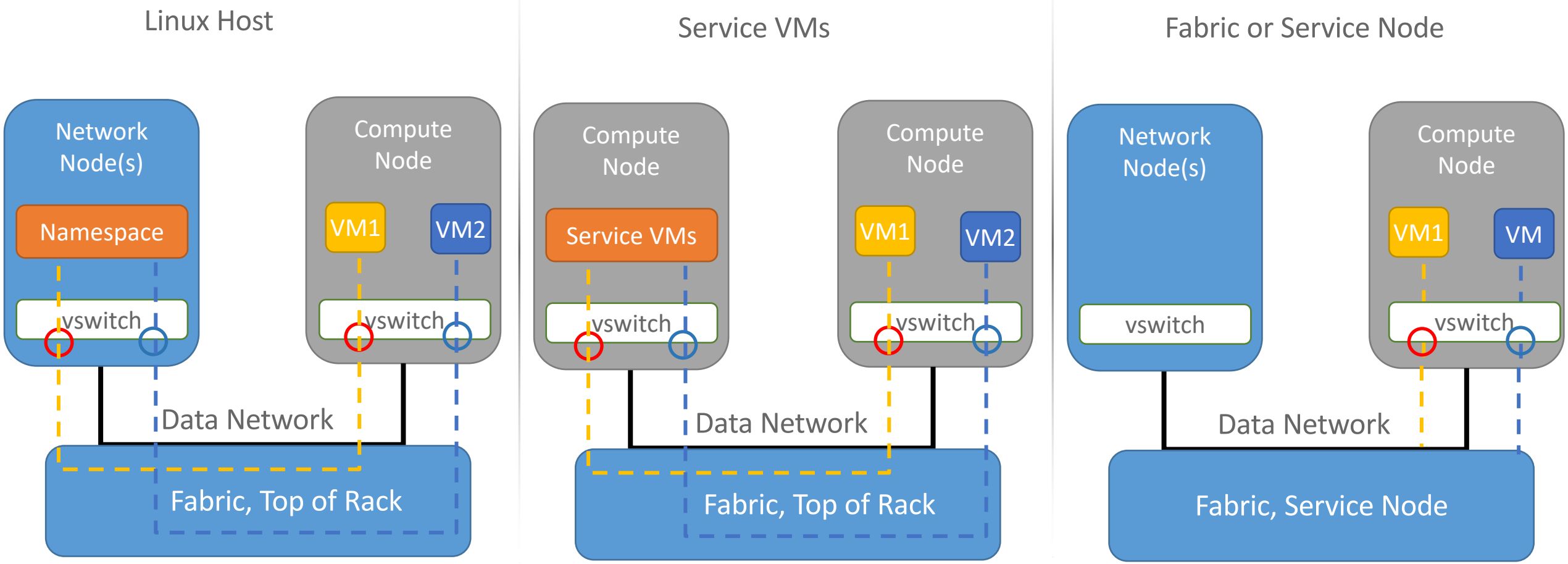


- Different deployment types exist to deploy neutron control components
  - Control Node
  - Dedicated Network Node
- L3, DHCP and additional services run on controller/network node
- L2 defined locally on compute node

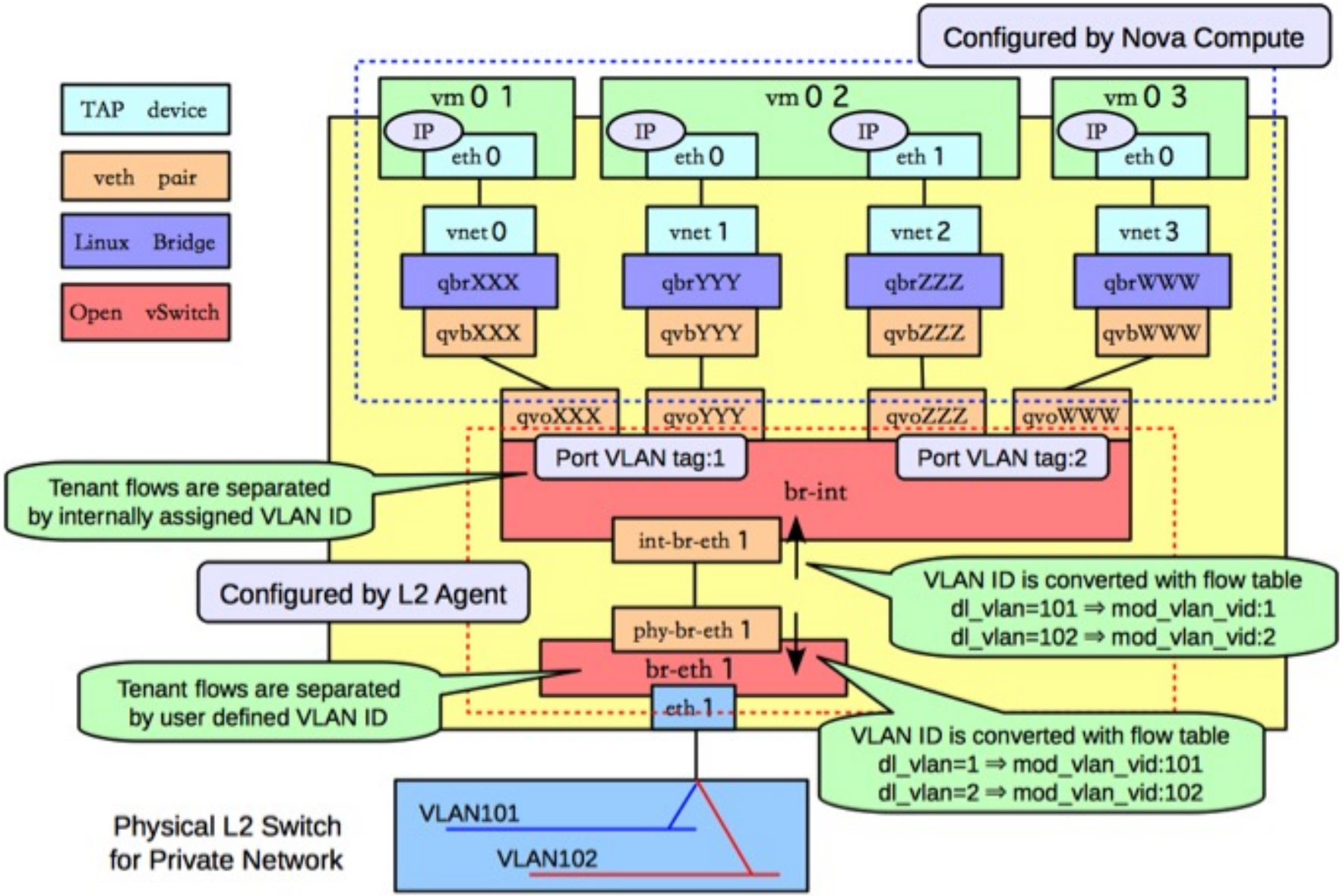
Layer 2 network tenant topologies



Layer 3 tenant network topologies



Neutron OVS Model



Neutron Services

LBaaS

- LBaaS – Load Balancing as a Service
- Distributes clients requests across multiple servers
- This improves:
  - Server fault tolerance
  - Minimize end user response time
  - Prevent bottlenecks and optimize resource utilization

VPNaaS

- Extends L3_agent
- Adds ability to create point to point and point to multipoint IPSEC tunnels
- Default model leverages OpenSwan to control p2p links

FWaaS

- FWaaS – Fire Wall as a Service
- Provides OpenStack users with the ability to deploy firewalls to protect their networks
- Features of OpenStack
  - Apply firewall rules on traffic entering and leaving tenant networks
  - Support for applying tcp, udp, icmp, or protocol agnostic rules
  - Creation and sharing of firewall policies which hold an ordered collection of the firewall rules
  - Ability to audit firewall rules and policies