Overview to Cisco Network Function Virtualization Infrastructure

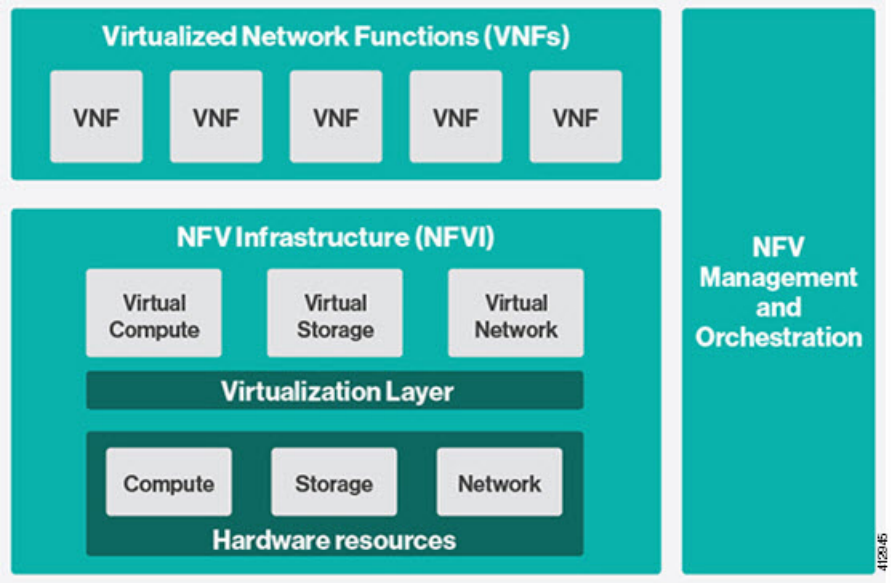
This section contains the following topics:

Cisco Network Function Virtualization Infrastructure Overview

Cisco Network Function Virtualization Infrastructure (NFVI) provides the virtual layer and hardware environment in which virtual network functions (VNFs) can operate. VNFs provide well-defined network functions such as routing, intrusion detection, domain name service (DNS), caching, network address translation (NAT), and other network functions. While these network functions require a tight integration between network software and hardware, the use of VNF enables to decouple the software from the underlying hardware.

The following figure shows the high level architecture of Cisco NFVI.

Figure 1: General NFV Infrastructure



Cisco NFVI includes a virtual infrastructure layer (Cisco VIM) that embeds the Red Hat OpenStack Platform (OSP 13). Cisco VIM includes the Queens release of OpenStack, which is an open source cloud operating system that controls large pools of compute, storage, and networking resources. Cisco VIM manages the OpenStack compute, network, and storage services, and all NFVI management and control functions. Key Cisco NFVI roles include:

- Control (including Networking)
- Compute
- Storage
- Management (including logging, and monitoring)

Hardware that is used to create the Cisco NFVI pods include a specific combination of the following based on pre-defined BOMs. For more details, contact Cisco VIM Product Management.

- Cisco UCS® C240 M4/M5: Performs management and storage functions and services. Includes dedicated Ceph (UCS 240-M4 or UCS 240-M5) distributed object store and file system. (Only Red Hat Ceph is supported).
- Cisco UCS C220/240 M4/M5: Performs control and compute services.

- HP DL360 Gen9: It is a third-party compute where the control plane is Cisco UCS servers.
- Cisco UCS 220/240 M4/M5 (SFF): In a Micropod environment, expandable to maximum of 16 computes.
- Cisco UCS B200 M4 blades: It can be used instead of the UCS C220 for compute and control services. The B200 blades and C240 Ceph server are connected with redundant Cisco Fabric Interconnects managed by UCS Manager.
- Combination of M5 series servers are supported in M5-based Micropod and VIC/NIC (pure 40G) based Hyper-Converged and Micropod offering.
- Quanta servers as an alternate to Cisco UCS servers: Use of specific Quanta servers for the installation of the cloud both at the core and edge. An automated install of Central Ceph cluster to the edge pods is supported for Glance image services.

The UCS C240 and C220 servers are of type M4 or M5 Small Form Factor (SFF) models where the nodes can boot off a pair of HDDs or SSD as specified in BOM. Each UCS C240, UCS C220, and UCS B200 have two 10 GE Cisco UCS Virtual Interface Cards.

The B-Series pod consists of Cisco UCS B200 M4 blades for the Cisco NFVI compute and controller nodes with dedicated Ceph on a UCS C240 M4. The blades and Ceph server are connected via redundant fabric interconnects (FIs) managed by Cisco UCS Manager. The Cisco VIM installer performs bare metal installation and deploys OpenStack services using Docker™ containers to allow for OpenStack services and pod management software updates.

The following table shows the functions, hardware, and services managed by Cisco NFVI nodes.

Table 1: Cisco NFVI Node Functions

Function	Number	Hardware	Services
Management	1	• UCS C240 M4 SFF with 8, 16, or 24 1.2 TB HDDs (24 is recommended) • UCS C240 M5 SFF with 8, 16, or 24 1.2 TB HDDs (24 is recommended) • UCS C220 M5 SFF with 8x1.2 TB HDDs • Quanta Server (D52BE-2U) with 2x1.2TB HDD • Quanta Server (D52BQ-2U 3UPI) with 2x.3.8TB HDD	• Cisco VIM Installer • Cobbler server • Docker Registry • ELK server • CVIM MON components: Prometheus and TSDB

Function	Number	Hardware	Services
Control	3	• UCS C220/C240 M4/M5 with 2x 1.2 TB HDDs or 2x960G SSDs (in a Micropod or Full Pod environment) • UCS B200 with two 1.2 TB HDDs • Quanta Server (D52BE-2U) with 2x960 G SSD • Quanta Server (D52BQ-2U 3UPI) with 2x960 G SSD for edge pod	• Maria Database/Galera • RabbitMQ • HA Proxy/Keepalive • Identity Service • Image Service • Compute management • Network service • Storage service • Horizon dashboard • Fluentd
Compute	2+	• UCS C220/C240 M4/M5 with two 1.2 TB HDDs, or 2x9.6 GB SSDs (in a Micropod or Full Pod environment) • UCS B200 with two 1.2 TB HDDs • HP DL360 Gen9 • Quanta Server (D52BE-2U/ D52BQ-2U 3UPI) with 2x960 G SSD	• Virtual Networking Service • Compute service • Fluentd

Function	Number	Hardware	Services
Storage	3 or more	SSD and HDD drives must be in a 1:4 ratio per storage node minimum. Storage node configuration options: Fullon environment:: • UCS C240 M4/M5 with two internal SSDs, 1-4 external SSD, 4-20x- 1.2 TB HDDs • SSD-based Ceph: UCS C240 M4/M5 with 2 internal SSDs, minimum of 4 external SSDs, expandable to 24 SSDs • Quanta Server (D52BE-2U) HDD Based: 4 SSD 960GB for Journal + 16 SAS HDD (16x2.4 TB) for OSD + 2 (2x2.4 TB SAS 10krpm HDD) for OS • Quanta Server (D52BE-2U) SSD Based: 20 SSD (3.8 TB) OSD + 2 OSBoot (2x3.8TB SSD) Micropod/UMHC/NGENAHC environment: • UCS C240 M4/M5 with two 1.2TB HDD for OS boot, one/2 SSDs and 5/10x1.2 TB HDDs • UCS C240 M4/M5 with 2x960GB SSD for OS boot and 4 or 8 x960 GB SSDs	• Storage service
Top of Rack (ToR)	2	Recommended Cisco Nexus 9000 series switch software versions: • 7.0(3)I4(6) • 7.0(3)I6(1) Cisco NCS 5500 as ToRs or Cisco Nexus 9000 switches running ACI 3.0 (when ACI is used)	ToR services • Cisco NCS 5500 provides ToR service with VIM running on C-series with Intel NIC and VPP as the mechanism driver for deployment.

Note

- Internal SSD is the boot device for the storage node
- You can use any ToR that supports virtual port channel. Cisco recommends you to use Cisco Nexus 9000 SKUs as ToR, which is released as part of Cisco VIM. When Cisco NCS 5500 acts as a ToR, auto-ToR config is mandatory.
- You must use the automated ToR configuration feature for Cisco NCS 5500.

.

Software applications that manage Cisco NFVI hosts and services include:

- Red Hat Enterprise Linux 7.6 with OpenStack Platform 13.0—Provides the core operating system with OpenStack capability. RHEL 7.6 and OPS 13.0 are installed on all target Cisco NFVI nodes.
- Cisco Virtual Infrastructure Manager (VIM)—An OpenStack orchestration system that helps to deploy and manage an OpenStack cloud offering from bare metal installation to OpenStack services, taking into account hardware and software redundancy, security and monitoring. Cisco VIM includes the OpenStack Queens release with more features and usability enhancements that are tested for functionality, scale, and performance.
- Cisco Unified Management—Deploys, provisions, and manages Cisco VIM on Cisco UCS servers.
- Cisco UCS Manager—Used to perform certain management functions when UCS B200 blades are installed. Supported UCS Manager firmware versions are 2.2(5a) and above.
- Cisco Integrated Management Controller (IMC)— Cisco IMC 2.0(13i) or later is supported, when installing Cisco VIM 2.4.

  For the Cisco IMC lineup, the recommended version is as follows:

UCS-M4 servers	Recommended: Cisco IMC 2.0(13n) or later. It is also recommended to switch to 3.0(3a) or later for pure intel NIC based pods.

  For Cisco IMC 3.x and 4.y lineup, the recommended version is as follows:

UCS-M4 servers	Cisco IMC versions are 3.0(3a) or later, except for 3.0(4a). Recommended: Cisco IMC 3.0(4d). Expanded support of CIMC 4.0(1a), 4.0(1b), 4.0(1c). You can move to 4.0(2f) only if your servers are based on Cisco VIC.

UCS-M5 servers	We recommend that you use CIMC 3.1(2b) or higher versions. Do not use 3.1(3c) to 3.1(3h), 3.0(4a), 4.0(2c), or 4.0(2d). Support of CIMC 4.0(4e) or later. We recommend that you use CIMC 4.0(4e). The bundle version of a minimum of CIMC 4.0 (4d) is needed for Cascade Lake support. For GPU support, you must ensure that the server is running with CIMC 4.0(2f).

Enables embedded server management for Cisco UCS C-Series Rack Servers. Supports Cisco IMC firmware versions of 2.0(13i) or greater for the fresh install of Cisco VIM. Because of recent security fixes, we recommend you to upgrade Cisco IMC to 2.0(13n) or higher. Similarly, Cisco IMC version of 3.0 lineup is supported. For this, you must install Cisco IMC 3.0 (3a) or above.

The Quanta servers need to run with a minimum version of BMC and BIOS as listed below:

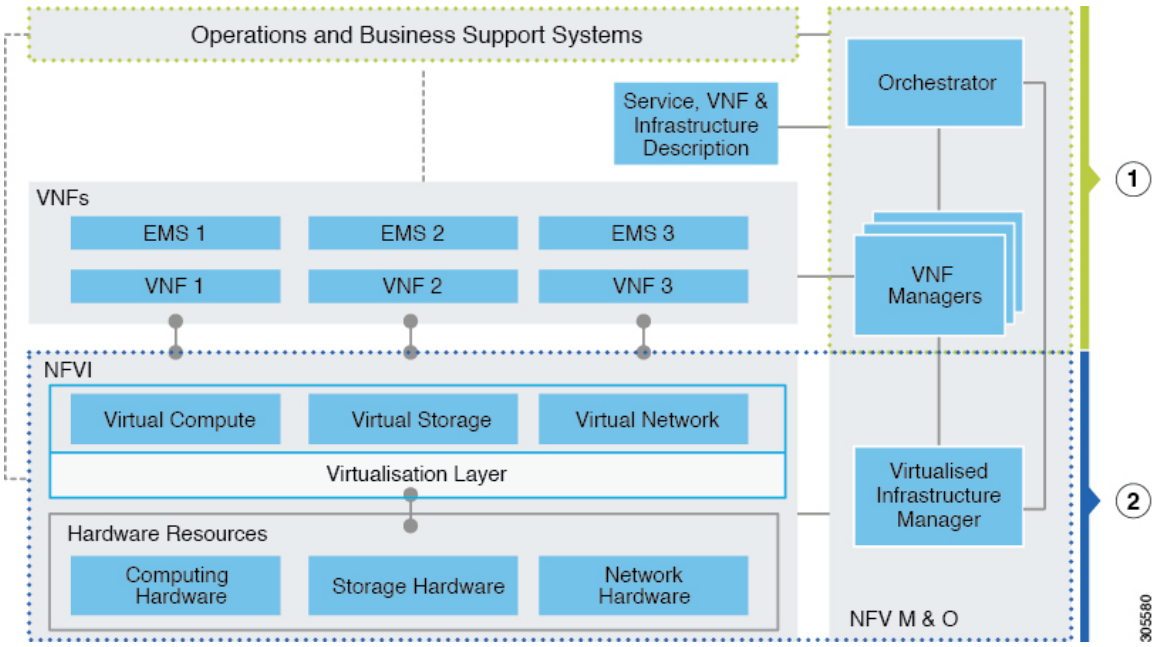
SKU Type	BMC Version	BIOS Version
D52BQ-2U 3UPI (CDC SKU)	4.68.22	3A11.BT17
D52BE-2U (GC SKU)	4.68.22	3A11.BT17

- Cisco Virtual Topology System (VTS)—It is an open, overlay management and provisioning system for data center networks. VTS automates DC overlay fabric provisioning for physical and virtual workloads. This is an optional service that is available through Cisco VIM.
- Cisco Virtual Topology Forwarder (VTF)—Included with VTS. VTF leverages Vector Packet Processing (VPP) to provide high performance Layer 2 and Layer 3 VXLAN packet forwarding.

Two Cisco VNF orchestration and management applications that are used with Cisco NFVI include:

- Cisco Network Services Orchestrator, enabled by Tail-f—Provides end-to-end orchestration spanning multiple network domains to address NFV management and orchestration (MANO) and software-defined networking (SDN). For information about Cisco NSO, see Network Services Orchestrator Solutions.
- Cisco Elastic Services Controller—Provides a single point of control to manage all aspects of the NFV lifecycle for VNFs. ESC allows you to automatically instantiate, monitor, and elastically scale VNFs end-to-end. For information about Cisco ESC, see the Cisco Elastic Services Controller Data Sheet.

Figure 2: NFVI Architecture With Cisco NFVI, Cisco NSO, and Cisco ESC



At a high level, the NFVI architecture includes a VNF Manager and NFV Infrastructure.

1	• Cisco Network Services Orchestrator • Cisco Elastic Services Controller
2	Cisco NFVI: • Cisco VIM + • Cisco UCS/Quanta/3rd Party Compute and Cisco Nexus Hardware + • Logging and Monitoring Software + • Cisco Virtual Topology Services (optional) + • Accelerated Switching with VPP (optional) • Cisco Unified Management (optional) • Pod Monitoring (optional)

For cloud networking, Cisco NFVI supports Open vSwitch over VLAN as the cloud network solution for both UCS B-series and UCS C-Series pods. Both B-Series and C-Series deployments support provider networks over VLAN.

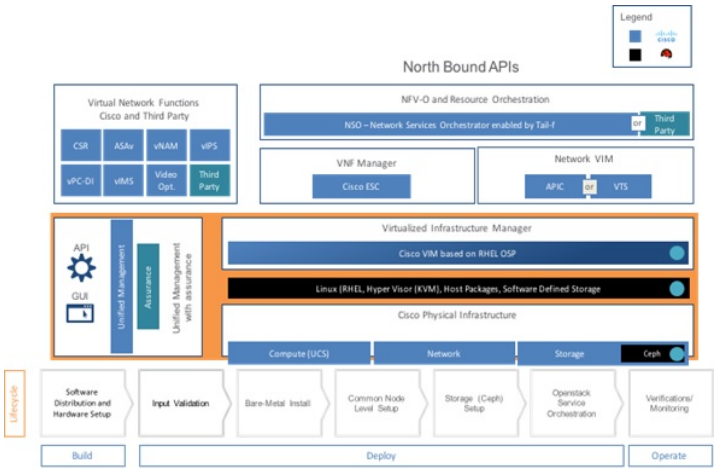
In addition, with a C-series pod, you can choose:

- To run with augmented performance mechanism by replacing OVS/LB with VPP/VLAN (for Intel NIC).
- To have cloud that is integrated with VTC which is an SDN controller option.

The Cisco NFVI uses OpenStack services running inside containers with HAProxy load balancing and providing high availability to API and management network messaging. Transport Layer Security (TLS) protects the API network from external users to the HAProxy. Cisco VIM installation also includes service assurance, OpenStack CloudPulse, built-in control, and data plane validation. Day two pod management allows you to add and remove both compute and Ceph nodes, and replace the controller nodes. The Cisco VIM installation embeds all necessary RHEL licenses as long as you use the Cisco VIM supported BOM and the corresponding release artifacts.

The following illustration shows a detailed view of the Cisco NFVI architecture and the Cisco NFVI installation flow.

Figure 3: Detailed Cisco NFVI Architecture View

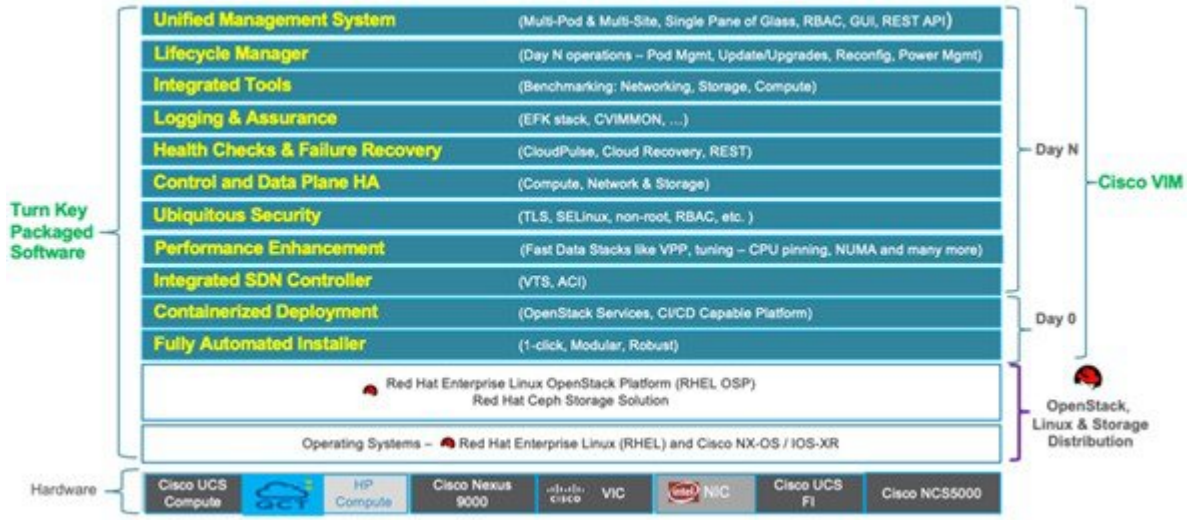


Cisco Virtualized Infrastructure Manager Overview

Cisco Virtualized Infrastructure Manager (VIM) is a fully automated cloud lifecycle management system. Cisco VIM helps to bring up a fully functional cloud in hours, with integrated end-to-end control and data plane verification in place. Cisco VIM offers fully automated day 1 to day n cloud lifecycle management. These include capabilities such as pod scaling (expansion), software update, upgrade, or reconfigure parameters, consolidated logging with rotation and export, software update and upgrade. These have been implemented in line with the operational and security best practices of service providers and enterprises.

The following figure provides the high-level overview of all day-0 and day-n items of Cisco VIM.

Figure 4: Cisco VIM Capability Overview



Cisco VIM Features

Cisco VIM is the only standalone fully automated cloud lifecycle manager offering from Cisco for a private cloud. Cisco VIM integrates with Cisco UCS C-series (with or without HP as third-party compute) or B-series, or Quanta (D52BQ-2U 3UPI or D52BE-2U) servers, and Cisco VIC or Intel NIC. This document and its accompanying administrator guide help the cloud administrators to set up and manage the private cloud.

Following are the features of the Cisco VIM:

Feature Name	Comments
OpenStack Version	RHEL 7.6 with OSP 13 (Queens).

Hardware Support Matrix	• UCS C220/B200 M4 controller or compute with Intel V3 (Haswell). • UCS C240/220 M4 controller or compute with Intel V4 (Broadwell). • UCS C240/220 M5 controller or compute or Ceph with Intel Skylake or Cascade Lake CPU. • HP DL360 Gen 9 with control plane on Cisco UCS M4 servers. • UCS C220/240 M5 in a Micropod environment, with an option to add up to 16 UCS C220/240 M5 computes. • UCS C240/220 M5 controller or compute with Intel X710 support with SRIOV and Cisco Nexus 9000 or Cisco NCS 5500 series switch as ToR. • UCS C240/220 M5 servers with Cisco 1457 (for control plane) and Intel XXV710 NIC (for data plane with VPP) and SR-IOV. • Support of physical GPU for M5. • Quanta servers act as an alternative to Cisco UCS servers for full on and micro (D52BQ-2U 3UPI), and edge (D52BE-2U) deployment of the cloud. • Quanta servers for central Ceph (D52BQ-2U 3UPI), cluster for edge pod to offer glance image services. • SATA M.2 (960G) as an option for a boot drive on M5.
NIC support	**1.** Cisco VIC: VIC 1227, 1240, 1340, 1380, 1387 (for M5) in 40G VIC/NIC offering, 1457. **2.** Intel NIC: X710, 520, XL710, xxv710 (25G).

POD Type	

	1. Fullon: Dedicated control, compute, and storage (C-series) node running on Cisco VIC (M4) or Intel X710 (for M4 or M5) (full on) with Cisco Nexus 9000 or Cisco NCS 5500 series switch (only for Intel NIC and VPP as mechanism driver) as ToR. For fullon pods based on Quanta (D52BE-2U) servers, the NIC is xxv710 (25G) with Cisco Nexus 9000 as ToR. Support of UCS M4 (10G VIC with 2-XL710) compute with UCS M5 (Cisco VIC 1457 with 2-XL710). 2. Dedicated control, compute, and storage (C-series) node running on Cisco VIC and Intel NIC (full on) with Cisco Nexus 9000 as ToR. Only SRIOV is supported on Intel NIC. Support of Intel X520 (with 2 NIC cards/compute) on M4 pods or XL710 (2 or 4 NIC cards/compute) on M4/M5 pods for SRIOV cards in the VIC/NIC combination. Few computes can run with/without SRIOV in a given pod. For M4 pods, VIC/NIC computes running XL710 and X520 can reside in the same pod. 3. Dedicated control, compute, and storage (UCS M5 SFF C-series) node running on Cisco VIC 1457 and Intel xxv710 NIC (full on) with Cisco Nexus 9000 as ToR. Only SRIOV is supported on Intel NIC. With VPP and OVS as the mechanism driver, the number of SRIOV ports are 2 or 4, respectively. 4. Expanding the VTS support to include UCS-M5 computes with Cisco VIC 1457 in an existing M4-based pod running on VIC 1227 5. Dedicated control, compute, and storage (B-Series) node running on Cisco NIC. 6. Micropod: Integrated (AIO) control, compute, and storage (C-series) node running on Cisco VIC, Intel X710X or VIC and NIC combo. Micropod can be optionally expanded to accommodate more computes (up to 16) running with the same NIC type. This can be done as a day-0 or day-1 activity. The computes can boot off an HDD or SSD. Intel NIC-based Micropod supports SRIOV, with the M5-based Micropod supporting only XL710 as an option for SRIOV. From Cisco VIM 3.4.1, the micropod option has been extended to Quanta (D52BE-2U) servers with Intel XXV710 NIC (25G) with Cisco Nexus 9000 (-FX) as ToR. 7. Hyper-converged on M4(UMHC): Dedicated control and compute nodes, with all storage acting as compute nodes (M4 C-series) and running on a combination of 1-Cisco VIC (1227) and 2x10GE 520 or 2x40GE 710XL Intel NIC

with an option to migrate from one to another. The pod can be extended to M5-based computes with 40G Cisco VIC along with 2x40GE 710XLNIC (optionally).

Note In a full-on (VIC based), or hyper-converged pod, computes can either have a combination of 1-Cisco VIC (1227) and (2x10GE 520/2x40GE 710XL Intel NIC) or 1-CiscoVIC (1227). The compute running pure Cisco VIC does not run SR-IOV. In 2.4, Cisco supports HP DL360 Gen9 as third-party compute.

A mix of computes from different vendors for the same pod is not supported.

8. NGENA Hyper-converged (NGENAHC): Dedicated control and compute nodes, with all storage acting as compute (C-series) nodes. All nodes have a combination of 1-Cisco VIC (1227) for control plane, and 1x10GE 710X Intel NIC for data plane over VPP.

   Support of M5 as controller and hyper-converged nodes (with 1457 for control plane, and 1x10GE X710 (2 port) Intel NIC for Data plane) in an existing M4 based pod*

9. Hyper-converged on M5: Dedicated control and compute nodes, with all storage acting as compute (C-series) nodes, running on a combination of 1-Cisco VIC (40G) and 2x40GE 710XL Intel NIC. Support of M5 as controller and hyper-converged nodes (with 1457 for control plane, and 1x10GE X710 (2 port) Intel NIC for data plane) in an existing M4-based pod.

10. Edge: Available with restricted power and limited rack space. Quanta (D52BQ-2U 3UPI) servers with three converged control and compute nodes, expandable to 16 additional compute nodes. The edge cloud communicates with Quanta server based Central Ceph cluster for glance service. Persistent storage is not available.

11. Ceph: Designed to provide glance image services to edge cloud. Quanta (D52BE-2U) servers with three converged ccephontrol and cephosd nodes, expandable to additional cephosd nodes for additional storage.

ToR and FI support	**1.** For VTS-based installation, use the following Nexus version: 7.0(3)I2(2a) 7.0(3)I6(2) andd 7.0(3)I7(5a). **2.** For the mechanism driver other than VTS, use the following Nexus software version: 7.0(3)I4(6) 7.0(3)I6(1). If you are using auto-ToR configuration and CONFIGURE_TORS set to True, the nxos version - 7.0(3)I6(1) automation fails irrespective of the mechanism driver due to the defect CSCve16902. **3.** UCS-FI-6296. **4.** Support of Cisco NCS 5500 (with recommended Cisco IOS XR version 6.1.33.02I or 6.5.1) with splitter cable option. Also, extending day-0 configuration to support user defined route-target and ethernet segment id (ESI)
IPv6 Support for Management Network	**1.** Static IPv6 management assignment for servers **2.** Support of IPv6 for NTP, DNS, LDAP, external syslog server, and AD. **3.** Support of IPv6 for the Cloud API endpoint. **4.** Support of CIMC over IPv6 **5.** RestAPI over IPv6 **6.** Support of UM over IPv6
Mechanism drivers	OVS/VLAN, VPP (19.04)/VLAN (Fast Networking, Fast Data FD.io VPP/VLAN, based on the FD.io VPP fast virtual switch over intel NIC).
SDN controller integration	VTS 2.6.2 with optional feature of Managed VTS; ACI (ships in the night, or ToR automation via APIC API) with Cisco VIC or Intel NIC on the UCS C-series M4 platform.
Install methodology	• Fully automated online or offline installation. • Support of Cisco VIM Software Hub to mitigate the problem associated with logistics of USB distribution for air-gapped installation. • Support of USB 3.0 64GB for M5 and Quanta based Management node. Support of UCS 2.0 64GB for M4 based management node.

Scale	1. LA: Total of 128 nodes (compute and OSD) with Ceph OSD max at 20. **Note** It is recommended to deploy 30 nodes at a time. Also, after day-0, you can add only one ceph node at a time. 2. Micropod: Maximum of 16 standalone compute nodes. **Note** Ceph OSDs can be either HDD or SSD based across the pod. Computes can boot off 2x1.2TB HDD or 2x960GB SSD). In the same pod, some computes can have SSD, while others can have HDD.
Automated pod life cycle management	1. Add or remove compute and Ceph nodes and replace the controller node. 2. Static IP management for storage network 3. Reduction of tenant/provider VLAN via reconfiguration to a minimum of two. 4. Reconfiguration of passwords and selected optional services. 5. Automated software update

Platform security	• Secure OS, RBAC, network isolation, TLS, source IP filtering, Keystone v3, Bandit, CSDL-compliant, hardened OS, SELinux. • Change the CIMC password post install for maintenance and security. • Non-root log in for administrators. • Read-only role available for OpenStack users. • Enabling custom policy for VNF Manager. • Ability to disable the reachability of the management node to the cloud API network. • Hosting Horizon behind NAT or with a DNS alias. • Cinder volume encryption using Linux Unified Key Setup (LUKS). • Configurable login banner for SSH sessions. • Access to management node through LDAP. • Support for IPv6 filters for administration source networks. • Access to NFVIMON for non-root user. • Introduction of Vault with reconfigure option to encrypt secrets. Enablement of Vault as an option on day 2. • Extend permit_root_login to the Unified Management node. • CIMC authentication using LDAP. • Support of RedHat identity, policy and audit (IPA) system. • Support of Horizon and Keystone login settings.
EPA	NUMA, CPU pinning, huge pages, SRIOV with Intel NIC. Ability to allocate user defined CPU (upto 6) cores to VPP. Ability to allocate user defined CPU (upto 12) cores to Ceph for Micropod and hyper-converged nodes.
HA and Reliability	1. Redundancy at hardware and software level. 2. Automated backup and restore of the management node.
Unified Management Support	Single pane of glass in a single or multi instance (HA) mode. Supports multi-tenancy and manages multiple pods from one instance.
Central Logging	EFK integrated with external syslog (over v4 or v6) for a log offload, with optional support of NFS with EFK snapshot.

External Syslog Servers	Support of multiple external syslog servers over IPv4 or IPv6. The minimum and the maximum number of external syslog servers that is supported is 1 and 3, respectively
VM Migration	• Cold migration and resizing. • Live migration with OVS as mechanism driver.
Storage	• Object store with SwiftStack, Block storage with Ceph, or NetApp. • Option to use Ceph for Glance and SolidFire for Cinder. • Option to have multi-backend (HDD and SSD based) Ceph in the same cluster to support various I/O requirements and latency.
Monitoring	• Monitor CVIM pods centrally using the Highly Available CVIM Monitor (HA CVIM-MON) over v4 and v6. • Monitor CVIM pods individually using the local CVIM Monitor (CVIM-MON) over v4 and v6. Support of non-CVIM managed external servers running RHEL, CentOS, or both. • Ceilometer for resource tracking and alarming capabilities across core OpenStack components is only supported on fullon pod. • Third-party integration with Zenoss (called NFVIMON) • Traffic monitoring with OVS for debugging.
Optional OpenStack features	• Enable trusted Virtual Function on a per server basis. • DHCP reservation for virtual MAC addresses. • Enable VM_HUGE_PAGE_SIZE and VM_HUGE_PAGE PERCENTAGE on a per server basis. • Enable CPU and RAM allocation ratio on a per server basis.
Support of External Auth System	**1.** LDAP with anonymous bind option. **2.** Active Directory (AD)
Software update	Update of cloud software for bug fixes on the same release.
Software upgrade	• Upgrade of non-VTS cloud from release 3.2.1 or 3.2.2 to release 3.4.1. • Software upgrade of non-VTS cloud from Cisco VIM 2.4.y to Cisco VIM 3.4.1, where y=15,16, or 17.

CIMC Upgrade Capability	Central management tool to upgrade the CIMC bundle image of one or more servers.
VPP port mirroring	• Ability to trace or capture packets for debugging and other administrative purposes. • Automated update of BMC or BIOS and firmware of Quanta server.
VXLAN extension into the cloud	Extending native external VXLAN network into VNFs in the cloud. Support of Layer 3 adjacency for BGP. Support of single VXLAN or multi-VXLAN (with head-end replication as an option) network terminating on the same compute node. **Note** Only two-VXLAN network is supported for now.
Power Management of Computes	Option to power off or on computes selectively to conserve energy.
Technical support for CIMC	Collection of technical support for CIMC.
Enable TTY logging as an option	Enables TTY logging and forwards the log to external syslog server and EFK stack running on management node. Optionally, log is sent to remote syslog if that option is available
Remote installation of management node	Automated remote installation of management node over Layer 3 network.
Power management of computes	Option to selectively turn OFF or ON the power of computes to conserve energy
Unified Management authentication	Supports authentication through local and LDAP.
CIMC authentication via LDAP	Support of authentication through LDAP as an option.
Support of workload types	• Extending Cisco VIM to support baremetal (Ironic-based) and container (Cisco Container Platform) based workloads. • Support of bonding on the Ironic network.

Cloud adaptation for low latency workload	• Enable real-time kernel to support on edge pod • Automated BIOS configuration • Introduction of custom flavor. • Support of Intel N3000 card on selected servers to handle vRAN workloads. • Support of Cache Allocation Technology (CAT) to handle vRAN workloads. • Support of INTEL_SRIOV_VFS (defines SRIOV support for Intel NIC) and INTEL_FPGA_VFS (defines support for Intel N3000 FPGA card) at a per server level.
Automated enablement of Intel X710/XL710 NIC's PXE configuration on Cisco UCS-C series	Utility to update Intel X710/XL710 NIC's PXE configuration on Cisco UCS-C series.
Disk maintenance for Pod Nodes	Ability to replace faulty disks on the Pod nodes without the need for add, remove or replace node operation.
Integrated Test Tools	1. Open Source Data-plane Performance Benchmarking: VMTP (an open source data plane VM to the VM performance benchmarking tool) and NFVbench (NFVI data plane and a service chain performance benchmarking tool). Extending VMTP to support v6 over provider network. 2. NFVbench support for VXLAN. 3. Services Health Checks Integration: Cloudpulse and Cloudsanity.

Note Configure the LACP on the data plane ports of the Cisco Nexus 9000 ToR, when Cisco VIM is running on Intel NIC for data plane with VPP as the mechanism driver. When Cisco NCS 5500 is the ToR (with mechanism driver VPP), the LACP configuration on the data plane is done through the Auto-ToR configuration feature of Cisco VIM.

Cisco VIM Networking Overview

Cisco VIM supports installation on two different type of pods. The blade B-series and rack C-series based offering supports NICs that are from Cisco (called as Cisco VIC). You can choose the C-series pod to run in a pure Intel NIC environment, and thereby obtain SRIOV support on the C-series pod. This section calls out the differences in networking between the Intel NIC and Cisco VIC installations.

To achieve network level security and isolation of tenant traffic, Cisco VIM segments the various OpenStack networks. The Cisco NFVI network includes six different segments in the physical infrastructure (underlay). These segments are presented as VLANs on the Top-of-Rack (ToR) Nexus switches (except for the provider

network) and as vNIC VLANs on Cisco UCS servers. You must allocate subnets and IP addresses to each segment. Cisco NFVI network segments include: API, external, management and provisioning, storage, tenant and provider.

API Segment

The API segment needs one VLAN and two IPv4 addresses (four if you are installing Cisco VTS) in an externally accessible subnet different from the subnets assigned to other Cisco NFVI segments. These IP addresses are used for:

- OpenStack API end points. These are configured within the control node HAProxy load balancer.
- Management node external connectivity.
- Cisco Virtual Topology Services (VTS) if available in your Cisco NFVI package.
- Virtual Topology Controller (VTC). It is optional for VTS.

External Segment

The external segment needs one VLAN to configure the OpenStack external network. You can provide the VLAN during installation in the Cisco NFVI setup_data.yaml file, but you must configure the actual subnet using the OpenStack API after the installation. Use the external network to assign OpenStack floating IP addresses to VMs running on Cisco NFVI.

Management and Provisioning Segment

The management and provisioning segment needs one VLAN and one subnet with an address pool large enough to accommodate all the current and future servers planned for the pod for initial provisioning (PXE boot Linux) and, thereafter, for all OpenStack internal communication. This VLAN and subnet can be local to Cisco NFVI for C-Series deployments. For B-Series pods, the UCS Manager IP and management network must be routable. You must statically configure Management IP addresses of Nexus switches and Cisco UCS server Cisco IMC IP addresses, and not through DHCP. They must be through the API segment. The management/provisioning subnet can be either internal to Cisco NFVI (that is, in a lab it can be a non-routable subnet limited to Cisco NFVI only for C-Series pods), or it can be an externally accessible and routable subnet. All Cisco NFVI nodes (including the Cisco VTC node) need an IP address from this subnet.

Storage Segment

Cisco VIM has a dedicated storage network used for Ceph monitoring between controllers, data replication between storage nodes, and data transfer between compute and storage nodes. The storage segment needs one VLAN and /29 or larger subnet internal to Cisco NFVI to carry all Ceph replication traffic. All the participating nodes in the pod will have IP addresses on this subnet.

Tenant Segment

The tenant segment needs one VLAN and a subnet large enough to manage pod tenant capacity internal to Cisco NFVI to carry all tenant virtual network traffic. Only Cisco NFVI control and compute nodes have IP addresses on this subnet. The VLAN/subnet can be local to Cisco NFVI.

Provider Segment

Provider networks are optional for Cisco NFVI operations but are often used for real VNF traffic. You can allocate one or more VLANs for provider networks after installation is completed from OpenStack.

Cisco NFVI renames interfaces based on the network type it serves. The segment Virtual IP (VIP) name is the first letter of the segment name. Combined segments use the first character from each segment for the VIP, with the exception of provisioning whose interface VIP name is "mx" instead of "mp" to avoid ambiguity

with the provider network. The following table shows Cisco NFVI network segments, usage, and network and VIP names.

Table 2: Cisco NFVI Networks

Network	Usage	Network Name	VIP Name
Management/Provisioning	• OpenStack control plane traffic. • Application package downloads. • Server management; management node connects to servers on this network. • Host default route. • PXE booting servers during bare metal installations.	Management and provisioning	mx
API	• Clients connect to API network to interface with OpenStack APIs. • OpenStack Horizon dashboard. • Default gateway for HAProxy container. • Integration with endpoints served by SwiftStack cluster for native object storage, cinder backup service or Identity service with LDAP or AD.	api	a
Tenant	VM to VM traffic. For example, VXLAN traffic.	tenant	t
External	Access to VMs using floating IP addresses.	external	e
Storage	Transit network for storage back-end. Storage traffic between VMs and Ceph nodes.	storage	s
Provider Network	Direct access to existing network infrastructure.	provider	p
Installer API	• Administrator uses installer API network to ssh to the management node. • Administrator connects to installer API to interface with secured services. For example, Kibana on the management node.	VIM installer API	br_api

For each C-series pod node, two vNICs are created using different ports and bonded for redundancy for each network. Each network is defined in setup_data.yaml using the naming conventions listed in the preceding table. The VIP Name column provides the bonded interface name (for example, mx or a) while each vNIC name has a 0 or 1 appended to the bonded interface name (for example, mx0, mx1, a0, a1).

The Cisco NFVI installer creates the required vNICs, host interfaces, bonds, and bridges with mappings created between all elements. The number and type of created vNICs, interfaces, bonds, and bridges depend on the Cisco NFVI role assigned to the UCS server. For example, the controller node has more interfaces than

the compute or storage nodes. The following table shows the networks that are associated with each Cisco NFVI server role.

Table 3: Cisco NFVI Network-to-Server Role Mapping

	Management Node	Controller Node	Compute Node	Storage Node
Management/Provisioning	+	+	+	+
API		+		
Tenant		+	+	
Storage		+	+	+
Provider		+**	+	
External		+		

Note *ACIINFRA is only applicable when using ACI as a mechanism driver.

** Provider network is extended to controller nodes when VMs are on provider network with virtio.

The network arrangement on third-party HP compute is slightly different from that of Cisco compute running with Intel NIC, because the HP computes have 2 less NIC ports than that are available in the Cisco Intel NIC BOM.

Following table lists the differences in the network arrangement between the Cisco compute and third-party HP compute.

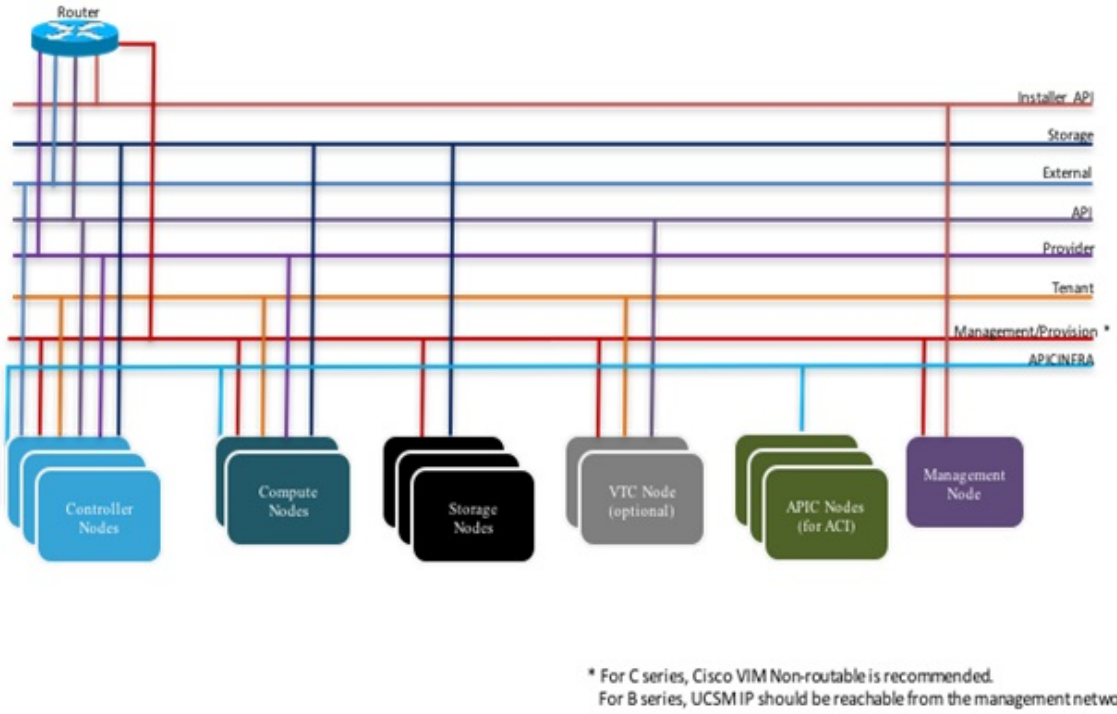
Network Interface	Cisco UCS Ce220/Ce240M4/M5 Compute	HPE ProLiant DL360 Gen9 and Quanta Compute
mx	Management control plane network	N/A

Network Interface	Cisco UCS Ce220/Ce240M4/M5 Compute	HPE ProLiant DL360 Gen9 and Quanta Compute
samxpet		Control and data plane network for everything other than SRIOV: 1. Management network on "br_mgmt" bridge interface with "samxpet" main interface as one of the member interface (native VLAN configuration required on the top-of-rack switches) 2. Storage network on the sub-interface "samxpet.<storage VLAN>" 3. Tenant and provider networks on veth interface "pet/pet-out" as one of the member interface with "br_mgmt" bridge interface
p	Provider data plane network	
sriov[0-3]	Provider data plane SRIOV networks	Provider data plane SRIOV networks
s	Storage control and data plane network	N/A
t	Tenant data plane network	N/A

In the initial Cisco NFVI deployment, two bridges are created on the controller nodes, and interfaces and bonds are attached to these bridges. The br_api bridge connects the API (a) interface to the HAProxy. The HAProxy and Keepalive container has VIPs running for each OpenStack API endpoint. The br_mgmt bridge connects the Management and Provisioning (mx) interface to the HAProxy container as well.

The following diagram shows the connectivity between Cisco NFVI nodes and networks.

Figure 5: Cisco NFVI Network Connectivity



Supported Layer 2 networking protocols include:

- VLAN over Open vswitch(SRIOV with Intel 710NIC).
- VLAN over VPP/VLAN for C-series Only.
- Single Root Input/Output Virtualization (SRIOV) for UCS B-Series pods. SRIOV allows a single physical PCI Express to be shared on a different virtual environment. The SRIOV offers different virtual functions to different virtual components, for example, network adapters, on a physical server.

The footprint of the cloud offering supported by Cisco VIM has continued to evolve over multiple releases to support customer needs that can vary across multiple dimensions such as cloud capacity, power, physical space, and affordability. The following table shows the available Cisco NFVI hardware and data path deployment combinations.

Table 4: Cisco NFVI Hardware and Data Path Deployment Combination

POD Type	NIC Type	Hardware Vendor	Mechanism Driver	TOR Type
fullon	Cisco VIC	UCS C series M4 UCS C series M5	OVS/VLAN	N9K
fullon	Cisco VIC	UCS B Series	OVS/VLAN with SRIOV	N9K

POD Type	NIC Type	Hardware Vendor	Mechanism Driver	TOR Type
fullon	Cisco VIC	UCS C series M4 UCS C series M5 with 1457 computes	VTF with VTC (VXLAN)	N9K
fullon	Intel NIC	UCS C series M4 UCS C series M5	OVS/VLAN with SRIOV	N9K
fullon	Intel NIC	Quanta D52BQ-2U 3UPI	OVS/VLAN with SRIOV	N9K
fullon	Intel NIC	UCS C series M4 UCS C series M5	VPP/VLAN with SRIOV	N9K NCS-5500
fullon	VIC for Control & Intel NIC for Data Plane	UCS C series M4 with HP as third-party Compute	OVS/VLAN with SRIOV	N9K
fullon	Cisco VIC with Intel NIC	UCS C series M4/M5 computes UCS C series M5	OVS/VLAN (VIC) with SRIOV (Intel NIC)	N9K
micro	Cisco VIC	UCS C series M4 UCS C series M5	OVS/VLAN	N9K
micro	Intel NIC	UCS C series M4 UCS C series M5	OVS/VLAN	N9K
micro	Intel NIC	UCS C series M4 UCS C series M5	VPP/VLAN	N9K NCS-5500
UMHC	Cisco VIC with Intel NIC	UCS C series M4 UCS C series M5	OVS/VLAN (VIC) with SRIOV (Intel NIC)	N9K
NGENAHC	VIC for Control & Intel NIC for Data Plane	UCS C series M4	VPP/VLAN	N9K
edge	Intel NIC	Quanta D52BE-2U	OVS/VLAN with SRIOV	N9K
ceph	Intel NIC	Quanta D52BQ-2U 3UPI	N/A	N9K

In the above table:

- fullon indicates the dedicated control, compute and ceph nodes.
- micro indicates converged control, compute and ceph nodes with expandable computes.

- Hyperconverged (HC) indicates dedicated control and compute nodes, but all ceph nodes are compute nodes.
- edge indicates converged control and compute nodes with expandable computes. It communicates with Central ceph cluster for Glance Image service. Persistent storage is not supported.
- ceph indicates converged cephcontrol & cephosd nodes, with an option to add cephosd nodes for glance image services.

Note The SRIOV support is applicable only for Intel NIC-based pods.

Note VTF with VTC is only supported on C-series Cisco VIC.

Pod with Intel NICs— In case of the pod having Intel NICs (X710), the networking is slightly different. You need to have atleast two NICs (4x10G) on a single server to support NIC level redundancy. Each NIC is connected to each ToR (connections explained later in the chapter). Since vNICs are not supported in the Intel card, bond the physical interfaces at the host and then create sub-interfaces based on the segment VLAN. Lets call the two NIC cards as NIC_1 and NIC_2 and call their four ports as A, B, C, D. Unlike Cisco VIC based pod, the traffic here is classified as follows:

1. Control plane
2. Data plane (external, tenant and non-SRIOV provider network).
3. SRIOV (optional for provider network). If SRIOV is used, the data plane network only carries external and tenant network traffic.

Control Plane

The control plane is responsible for carrying all the control and management traffic of the cloud. The traffic that flows through control plane are:

1. Management/Provision
2. Storage
3. API

The control plane interface is created by bonding the NIC_1 A port with NIC_2 A port. The bonded interface name is called as samx, indicating that it is carrying Storage, API, Management/Provision traffic (naming convention is similar to Cisco VIC pod). The slave interfaces (physical interfaces) of the bonded interface are renamed as samx0 and samx1. samx0 belongs to NIC_1 and samx1 belongs to NIC_2. Sub interfaces are then carved out of this samx interface based on the Storage, API VLANs. The management/provision traffic will be untagged/native VLAN in order to support pxe booting.

Data Plane

The data plane is responsible for carrying all the VM data traffic. The traffic that flows through the data plane are

- Tenant

- Provider
- External

The data plane is created by bonding the NIC_1 B port with NIC_2 B port. The bonded interface name here would be pet, indicating that it is carrying Provider, External and Tenant traffic. The slave interfaces of this bonded interface would be visible as pet0 and pet1. pet0 belongs to the NIC_1 and pet1 belongs to NIC_2.

In case of OVS/VLAN, the "pet" interface is used as it is (trunked to carry all the data VLANs) to the Openstack cloud, as all the tagging and untagging happens at the Openstack level. In case of Linux Bridge/VXLAN, there will be sub-interface for tenant VLAN to act as the VXLAN tunnel endpoint.

SRIOV

In case of Intel NIC pod, the third (and optionally the fourth) port from each NIC can be used for SRIOV traffic. This is optional and is set or unset through a setup_data.yaml parameter. Unlike the control and data plane interfaces, these interfaces are not bonded and hence there is no redundancy. Each SRIOV port can have maximum of 32 Virtual Functions and the number of virtual function to be created are configurable through the setup_data.yaml. The interface names of the SRIOV will show up as sriov0 and sriov1 on each host, indicating that sriov0 belongs to NIC_1 C port and sriov1 belongs to NIC_2 C port.

In the case of Intel NIC pod, the following table summarizes the above discussion

Network	Usage	Type of traffic	Interface name
Control Plane	To carry control/management traffic	Storage, API, Management/Provision	samx
Data Plane	To carry data traffic	Provider, External, Tenant	pet
SRIOV	To carry SRIOV traffic	SRIOV	sriov0, sriov1

The following table shows the interfaces that are present on each type of server (role based).

	Management Node	Controller Node	Compute Node	Storage Node
Installer API	+			
Control plane	+	+	+	+
Data plane		+	+	
SRIOV			+	

Note On an Intel pod, all kind of OpenStack networks are created using the **physnet1** as the physnet name.

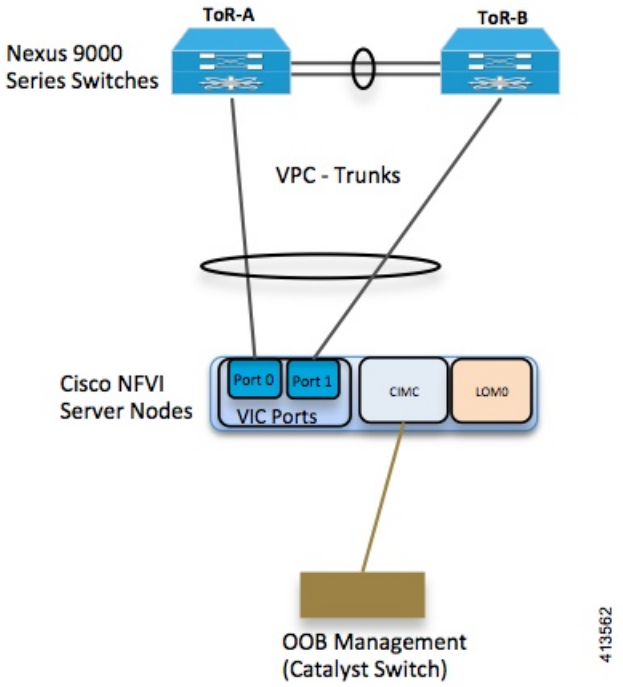
UCS C-Series Network Topologies

Cisco NFVI UCS servers connect to the ToR switches using Cisco UCS dual-port Virtual Interface Cards (VICs). The VIC is an Enhanced Small Form-Factor Pluggable (SFP+) 10 Gigabit Ethernet and Fiber Channel

over Ethernet (FCoE)-capable PCI Express (PCIe) card designed for Cisco UCS C-Series Rack Servers. Each port connects to a different ToR using a Virtual Port Channel (VPC). Each VIC is configured with multiple vNICs that correspond to specific Cisco VIM networks. The UCS Cisco IMC port is connected to an out-of-band (OOB) Cisco management switch.

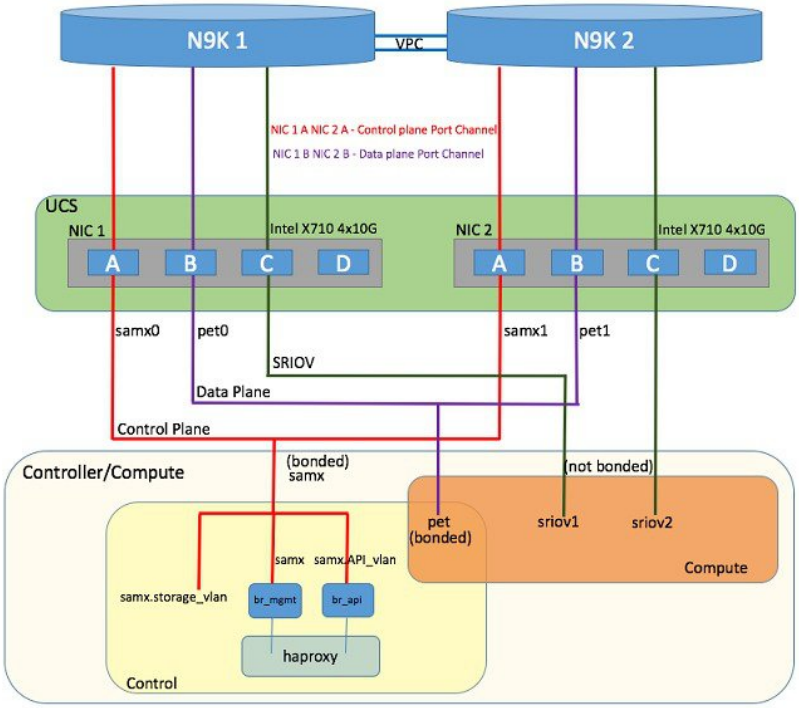
The following figure shows the UCS C-Series pod Cisco NFVI host to ToR topology.

Figure 6: UCS C-Series Host to ToR Topology



In the case of Intel NIC, a single two port Cisco VIC in the preceding diagram, is replaced with two 4-port 710 Intel NIC. An extra Intel NIC is added to provide card level redundancy.

Figure 7: UCS C-Series Intel NIC Details



Of the four ports that are available in each NIC card, port A is used for management traffic (provision, API, storage, etc), whereas the port B is used for data plane (tenant and provider network) traffic. Port C (and optionally Port D) is dedicated for SRIOV (configured optionally based on setup_data.yaml). Sub-interfaces are carved out of the data and control plane interfaces to provide separate traffic based on specific roles. While the ports A and B from each NIC help in forming bonded interface, the ports C and D over which SRIOV traffic for provider network flows is not bonded. Extreme care should be taken during pod setup, so that ports A, B and C for the Intel NIC is connected to the ToRs. Port D can be optionally used as a second pair of SRIOV ports by appropriate intent defined in the setup_data.yaml file. From Cisco VIM release 2.4.2 onwards, this port option is available for both M4 and M5 based systems or pods.

The following table provides the default link aggregation member pairing support for the pods based on server type:

Table 5: Default Link Aggregation Members Pairing

Server/POD Type	Target Functions	Default NIC Layout
M4 Intel NIC based	Control Plane	NIC-1 A + NIC-2 A
	Data Plane	NIC-1 B + NIC-2 B
	SRIOV 0/1	NIC-1 C + NIC-2 C
	SRIOV 2/3	NIC-1 D + NIC-2 D

Server/POD Type	Target Functions	Default NIC Layout
M5 Intel NIC based	Control Plane	NIC-1 A + NIC-1 B
	Data Plane	NIC-1 C + NIC-1 D
	SRIOV 0/1	NIC-2 A + NIC-2 B
	SRIOV 2/3	NIC-2 C + NIC-2 D

Note In M5, a NIC_LEVEL_REDUNDANCY option is introduced to support the M4 default option for link aggregation settings.

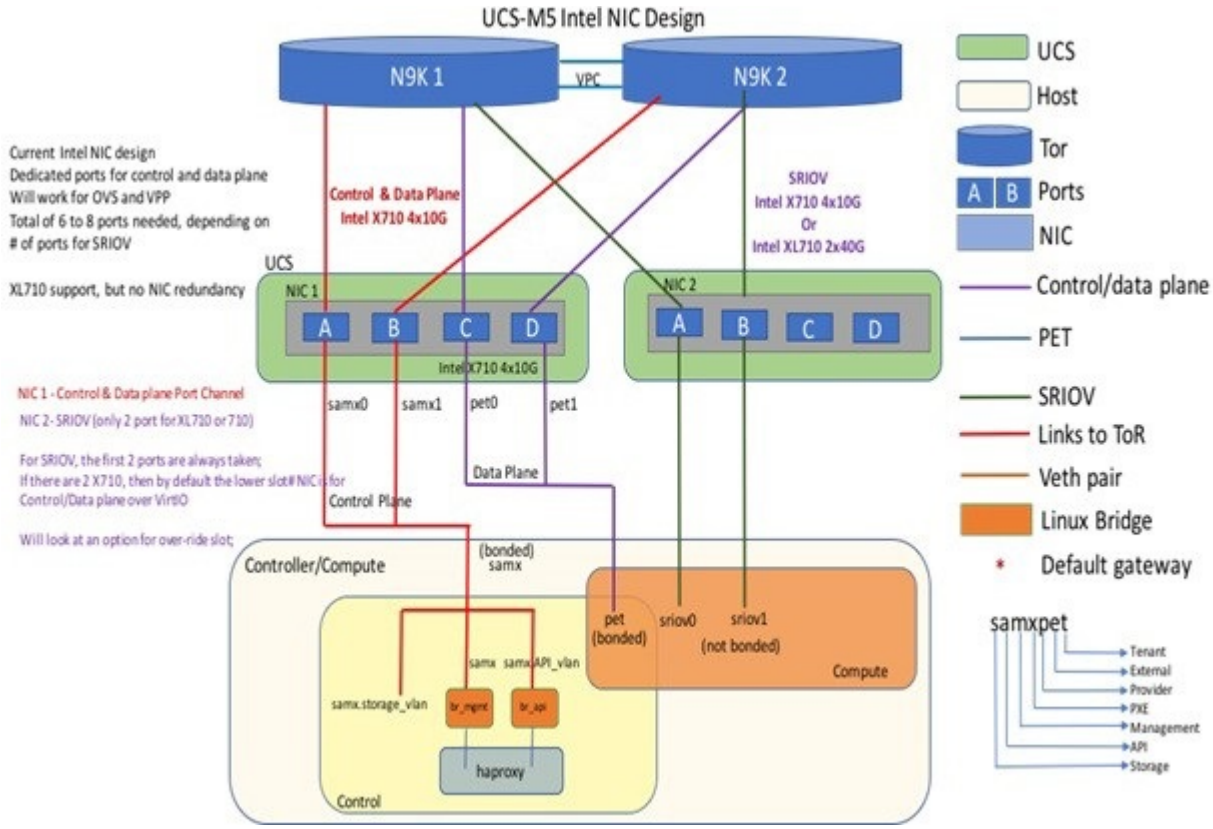
From Cisco VIM 2.4.2 onwards, support of M5 full on pods with two port XL-710 across control, compute and dedicated Ceph Nodes, and with NIC_LEVEL_REDUNDANCY is available. This deployment can be achieved with Cisco Nexus 9000 series or Cisco NCS 5500 as ToR. SRIOV is not supported in computes with XL-710. However, the pod can also support computes with four port X-710, where SRIOV is over port C and D.

In Cisco VIM, computes (M4 based testbed) running a Cisco 1227 VIC, and 2 2-port Intel 520 NIC are supported. In this combination, SRIOV is running on the Intel NIC, whereas the control and data plane are carried by virtual interfaces over Cisco VIC.

Cisco VIM 2.4 introduces the support of C220/C240 M5 servers in a micropod configuration with an option to augment the pod with additional computes (upto a max of 16). The M5 micropod environment is based on X710 for control and data plane and an additional XL710 or 2xX710 for SRIOV. The SRIOV card is optional. Once the SRIOV card is chosen, all the computes must have same number of SRIOV ports across the pod.

The following diagram depicts the server network card diagram for the M5 setup.

Figure 8: Networking Details of UCS-M5 Micropod Deployment

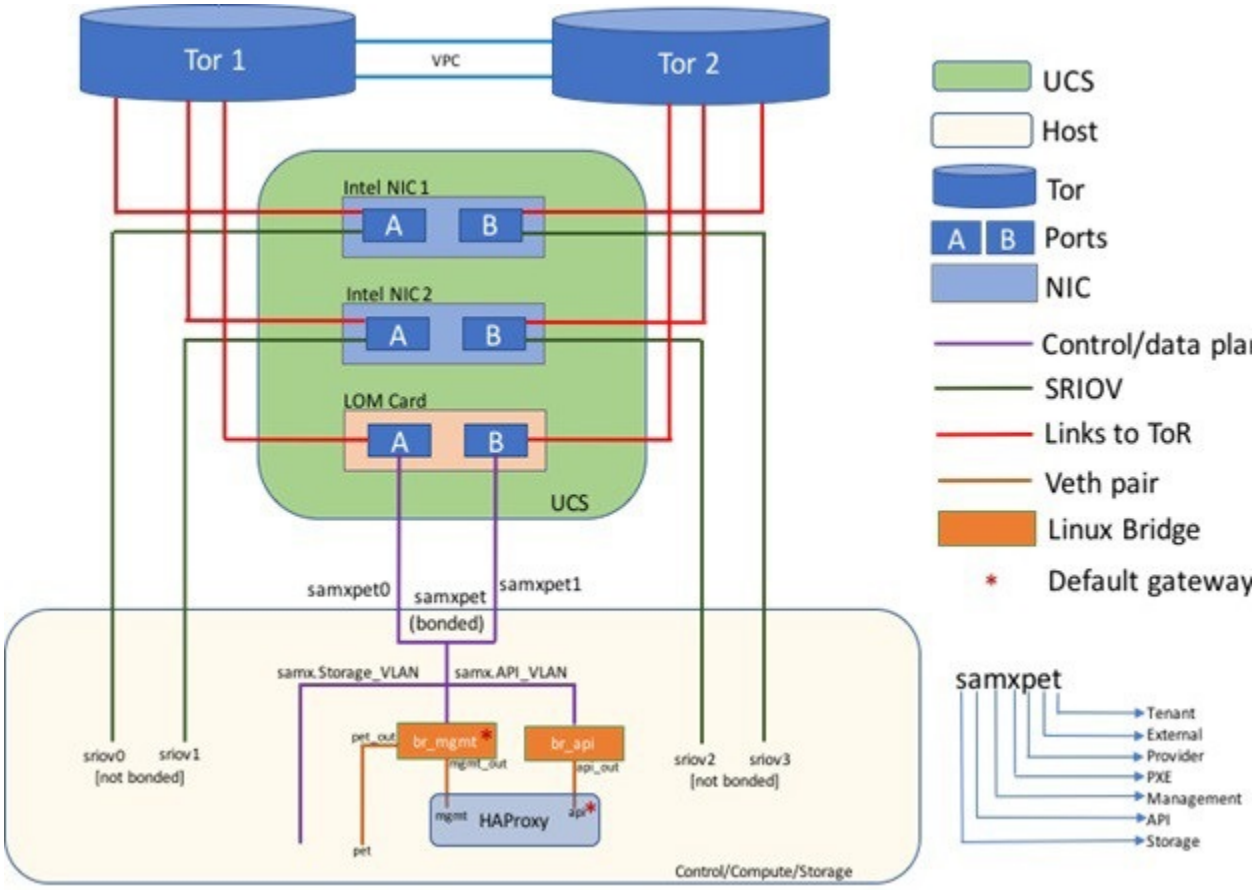


Cisco VIM 2.4 introduces the first third-party compute. The first SKU chosen is HPE ProLiant DL360 Gen9. In Cisco VIM 2.4, the supported deployment is a full-on pod, with OVS as the mechanism driver, where the management, control, and storage nodes are based on existing Cisco UCS c220/240M4 BOM, and the compute nodes are on HPE ProLiant DL360 Gen9 hardware:

```
ProLiant DL360 Gen9 with HP Ethernet 1Gb 4-port 331i Adapter - NIC (755258-B21) 2 x E5-2695
 v4 @ 2.10GHz CPU
8 x 32GB DDR4 memory (Total 256GB)
1 x Smart Array P440ar hardware RAID card with battery
2 x 1.2 TB - SAS 12GB/S 10k RPM HDD
1 x FlexLOM HP Ethernet 10Gb 2-port 560FLR-SFP+ Adapter
2 x PCIe HP Ethernet 10Gb 2-port 560SFP+ Adapter
System ROM: P89 v2.40 (02/17/2017)
iLO Firmware Version: 2.54 Jun 15 2017
```

In the case of HP Computes, the FlexLOM HP Ethernet 10Gb interface is used for management and tenant network, and the two additional HP Ethernet 10Gb 2-port 560SFP+ Adapters are used for SRIOV for the provider network. Listed below is network schematic of the HP Compute node.

Figure 9: Networking details of HP DL360GEN9



The Cisco NFVI controller node has four bonds: mx, a, t, and e. Each has a slave interface that is named with the network name association and a mapped number. For example, the management and provisioning network, mx, maps to mx0 and mx1, the API network, a, to a0 and a1, and so on. The bonds map directly to the vNICs that are automatically created on the controller node when it is deployed.

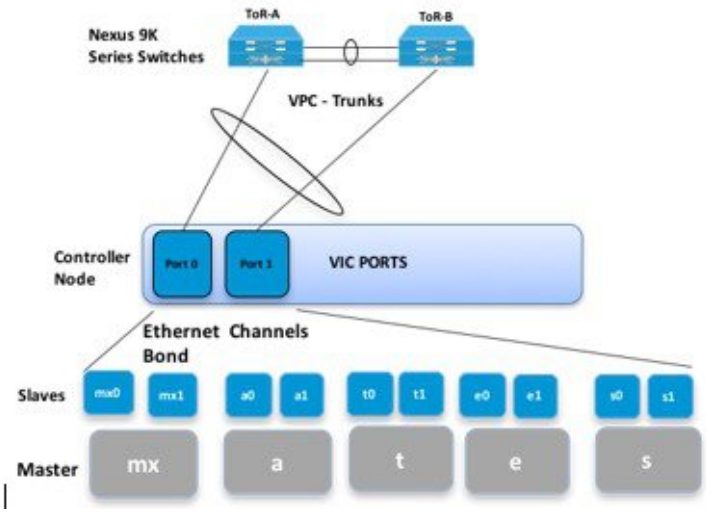
Cisco VIM 3.0 manages a third-party infrastructure based on Quanta servers, thereby bringing in true software abstraction. In the implementation, the supported deployment is a full-on or edge pod, with OVS as the mechanism driver. With the power limitation and rack restrictions on the edge pod, it cannot support hard-drives for the Ceph service. As the Edge pod does not need persistent storage, it is designed to communicate with a central ceph cluster for providing glance image services only.

The installation and management of the Central Ceph cluster is fully automated and it is assumed that the management network of the edge cloud is routable to that of the central Ceph cluster.

In the case of Quanta servers, the networking is similar to that of the HP computes except for the two port 25G (xxv710) Intel NICs. The 2x25GE OCP card is used for control and data plane network over virtio, and the two additional 25GE 2-port xxv710 based Intel NIC Adapters are used for SRIOV via the provider network.

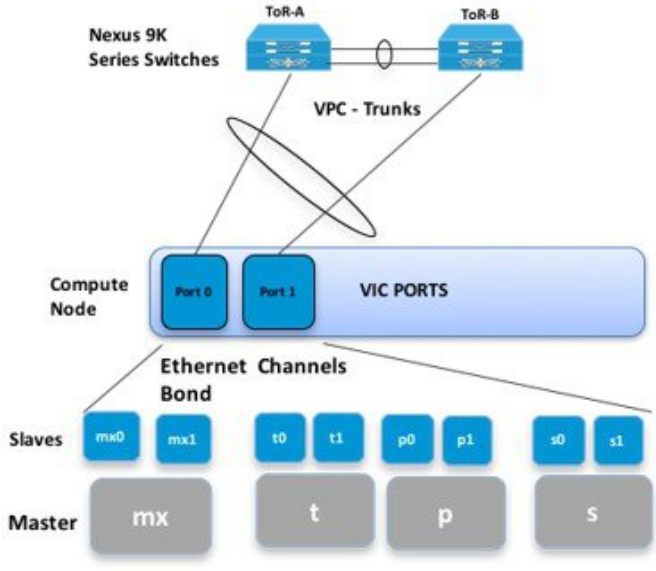
The following figure shows the controller node network-to-bond-to-vNIC interface mapping.

Figure 10: Controller Node Network to Bond Mapping



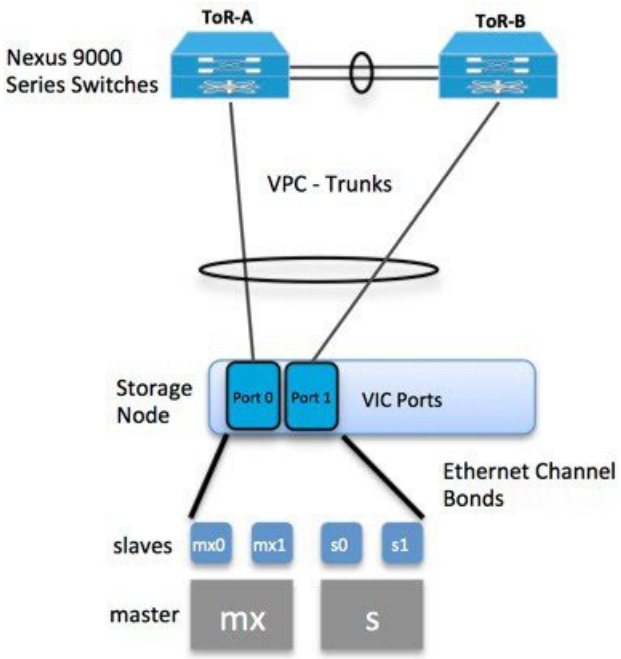
The Cisco NFVI compute node has three bonds: mx, t, and p. Each has a slave interface that is named with the network name association and a mapped number. For example, the provider network, p, maps to p0 and p1. The bonds map directly to the vNICs that are automatically created on the compute node when it is deployed. The following figure shows the compute node network-to-bond-to-vNIC interfaces mapping.

Figure 11: Compute Node Network to Bond Mapping



The Cisco NFVI storage node has two bonds: mx and s. Each has a slave interface that is named with the network name association and a mapped number. For example, the storage network, s, maps to s0 and s1. Storage nodes communicate with other storage nodes over the mx network. The storage network is only used for Ceph backend traffic. The bonds map directly to the vNICs that are automatically created on the storage node when it is deployed. The following figure shows the network-to-bond-to-vNIC interfaces mapping for a Cisco NFVI storage node.

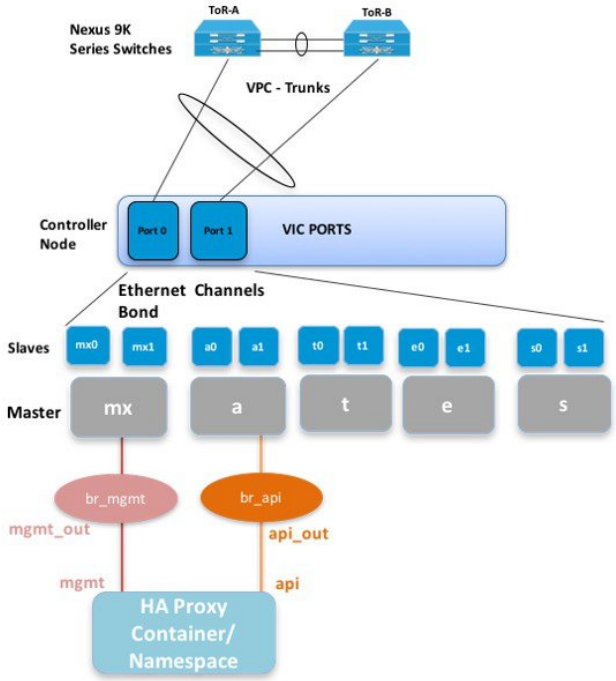
Figure 12: Storage Node Networking to Bond Mapping



Cisco NFVI installation creates two bridges on the controller nodes and interfaces and bonds are attached to the bridges. The br_api bridge connects the API (a) interface to the HAProxy container. The HAProxy and Keepalive container has VIPs running for each OpenStack API endpoint. The br_mgmt bridge connects the Management and Provisioning (mx) interface to the HAProxy container as well.

The following figure shows the connectivity between the mx interface and the br_mgmt bridge. It also shows the connectivity between the br_mgmt and the HAProxy container/namespace using mgmt_out and mgmt interfaces. The figure shows the connectivity between the api interface and the br_api bridge as well as the link between the br_mgmt bridge and the HAProxy container using api_out and mgmt_out interfaces.

Figure 13: Bridge and Network Namespace Layout



A sample routing table is shown below. br_api is the default route and br_mgmt is local to the pod.

```
[root@c43-bot-mgmt ~]# ip route
default via 172.26.233.193 dev br_api  proto static  metric 425
172.26.233.0/25 dev br_mgmt  proto kernel  scope link  src 172.26.233.104  metric 425
172.26.233.192/26 dev br_api  proto kernel  scope link  src 172.26.233.230  metric 425

[root@c43-bot-mgmt ~]# ip addr show br_api
6: br_api: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 58:ac:78:5c:91:e0 brd ff:ff:ff:ff:ff:ff
    inet 172.26.233.230/26 brd 172.26.233.255 scope global br_api
       valid_lft forever preferred_lft forever
    inet6 fe80::2c1a:f6ff:feb4:656a/64 scope link
       valid_lft forever preferred_lft forever

[root@c43-bot-mgmt ~]# ip addr show br_mgmt
7: br_mgmt: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 58:ac:78:5c:e4:95 brd ff:ff:ff:ff:ff:ff
    inet 172.26.233.104/25 brd 172.26.233.127 scope global br_mgmt
       valid_lft forever preferred_lft forever
    inet6 fe80::403:14ff:fef4:10c5/64 scope link
       valid_lft forever preferred_lft forever
```

Cisco VIM Management Node Networking

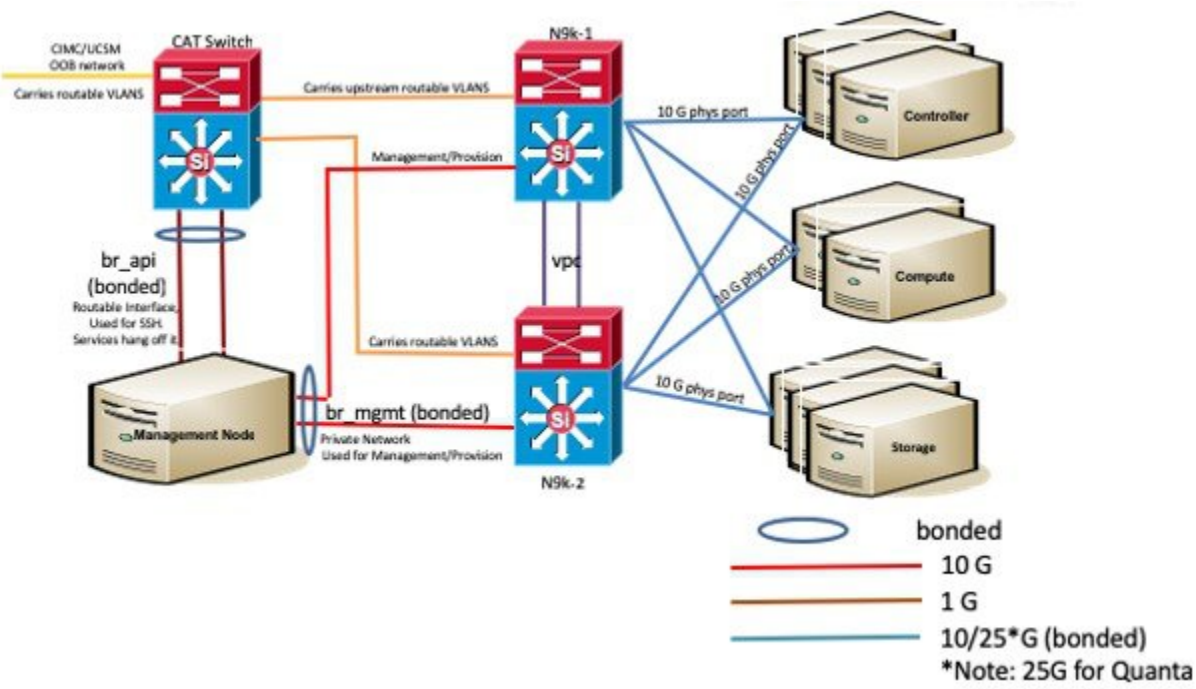
In Cisco VIM, the management node has an interface for API and another interface for provisioning. This is primarily done for security reasons so that internal pod management or control plane messages (RabbitMQ, Maria DB, and so on) do not leak out, and hence reduce the attack vector to the pod. As the name indicates, the API interface is to access the VIM installer API and is also used to SSH to the management node. All

external services (installer API, Insight, ELK, and so on) are password that is protected and hang off the API interface. Default route of the management node points to the API interface.

The second interface, also called the provisioning interface is used to PXE boot the various nodes that constitute the OpenStack pod. Typically, provisioning interface is a non-routable interface that is reserved for OpenStack management traffic.

In B-series pod, the networks between provisioning and the UCSM IP need to be routable. Proper ACL has to be applied in the upstream router so that other networks do not interfere with the provisioning network. Depending on the overall deployment, the management node acts as a jump-server to the OpenStack nodes.

Figure 14: Cisco VIM Management Node Networking

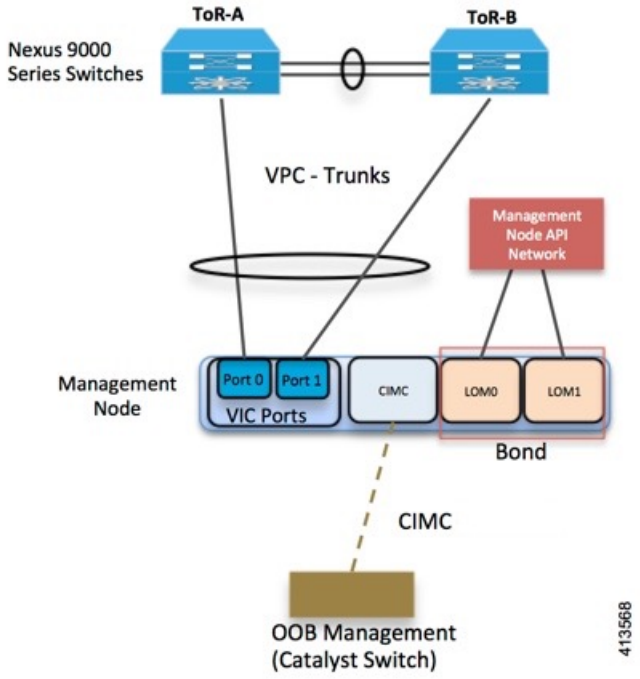


Cisco NFVI UCS C-series management node physically connects to the network. Unlike other nodes, the management node does not use multiple vNICs corresponding to specific Cisco NFVI networks. Instead, it connects to the management and API networks using two different physical connections. The management node connects to the management network using a Cisco two-port VIC or first two ports of intel X710-DA4, with each port connecting to a different ToR switch in a VPC configuration. The Cisco VIC card utilizes the default vNICs, but requires the vNICs to be in trunk mode and the default VLAN set to the management network VLAN. The management node connects to the API network using both one Gbps LAN On Motherboard (LOM) ports connected in a port channel configuration. These ports can either connect to the Nexus 9000 Series switch in a VPC configuration, or to an operator-managed switch(es), depending on how the operator wants to segment their network. The Cisco IMC port can optionally be connected to an out-of-band management Catalyst switch.

Management node services, which are required to start the other topology nodes, listen on the management network and the traffic flowing over the vNICs or NICs on that network. These services and the other management network services are unsecured. Secure management node services listen on the management node API network, and their traffic flows over the LOM ports. This service division allows tenants to utilize tighter network access control to the management network than the management node API network. The following figure shows the Cisco NFVI management node (UCS C-Series) API network connections.

Note Connecting Cisco IMC port to a Cisco OOB management switch is optional.

Figure 15: Management Node API Network Connections



For the day-0 server automation in Cisco VIM, ensure that the reachability to:

CIMC/ILO/BMC of the individual servers from the management node is available through the br_api network.

Cloud API, external network (for ssh to floating IPs) and provider network from the management node is available, as the VMTP and NFVbench are typically run from the management node.

Note From the Cisco VIM release 2.4.3 onwards, you can enable or disable the default behavior of the management node reachability from cloud API, external network, and provider network as part of their day-0 configuration.

If you disable the reachability to cloud api, external, and provider network for security reasons, then:

- VMTP and NFVbench are not accessible from the management node.
- Cloud api, external network and provider network must be properly routed as the Cisco VIM cannot automatically valid the same.

IPv6 Support on Management Network

You can switch from IPv4 to IPv6 as the number of available routable IPv4 networks is limited. In Cisco VIM, the management network uses the default IPv4 route to reach external service like NTP, DNS, AD/LDAP, SwiftStack, and so on, if it is not locally hosted.

Due to the limited availability of IPv4 address space, if you cannot provide a routable IPv4 network or local or dual-home of the external services that require routing, for example, AD or LDAP, deployment hindrance can occur.

IPv4 is obligatory in Cisco VIM, as the provision network colocates with the management network (mx/samx interface) for baremetal PXE install and Ansible orchestration.

As CEPH and OpenStack control plane communication are on the same management network, you cannot completely remove IPv4 from the management network. However, you can run IPv4+IPv6 dual stack in which IPv4 network can exist in a non-routable private network and IPv6 network can exist in a routable semi private network. This ensures to satisfy the requirements of the CiscoVIM and accessibility to the external services.

In Cisco VIM, the management network supports IPv6 addresses for servers, while the management node is statically allocated from a given pool. The external services that support both IPv4 and IPv6 addresses are DNS, NTP, and AD or LDAP. You can run IPv4+IPv6 (optionally) as the cloud API endpoint. CIMC/BMC can have IPv6 addresses.

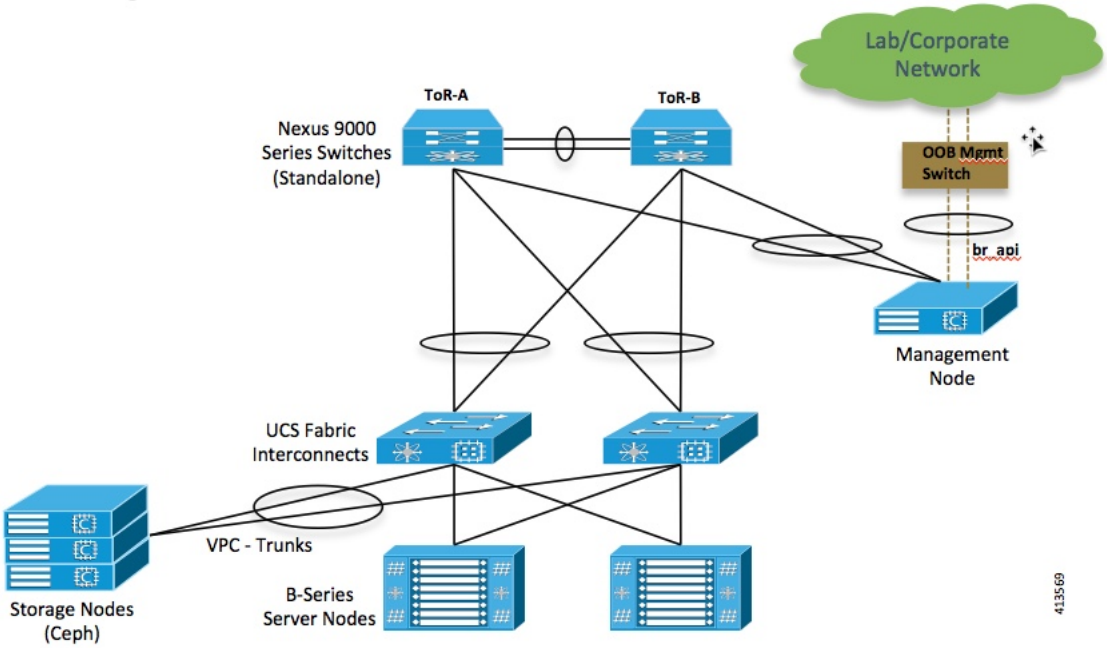
UCS C-Series and B-Series -Topologies

You can deploy Cisco NFVI using a combination of Cisco C-Series and B-Series servers. The C-Series management node is connected to the Cisco Nexus 9000 Series ToRs through the Cisco VIC in a VPC configuration. The UCS Fabric Interconnects (FIs) are connected to the ToRs and the UCS B-Series blade chassis is connected to the FIs. The C-Series storage nodes are connected to the ToRs as well. For C-series implementation, see *Cisco NFVI Networking Overview*. For the combination of the C-Series and B-Series implementation, two exceptions are listed below:

- For UCS B-Series, the Cisco UCS Manager IP address must be available to the Cisco NFVI management network. For UCS C-Series, this requirement is optional.
- The UCS Manager cluster and VIP connections are not attached to one of the Cisco NFVI network segments.

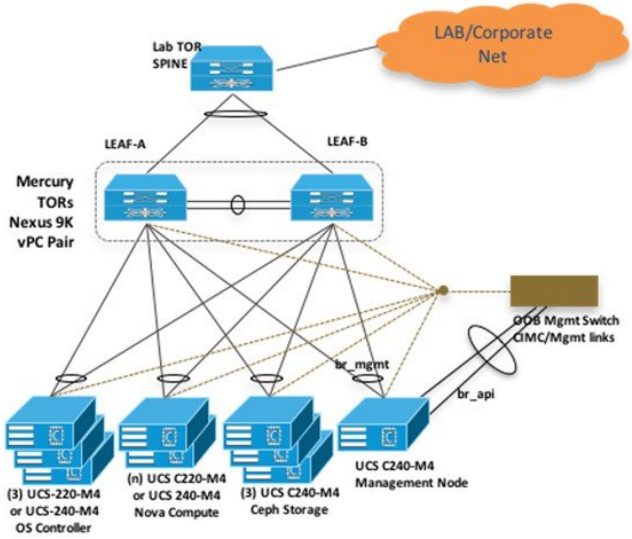
Following figure shows a high-level view of Cisco UCS C-Series and B-Series servers that are used in a Cisco NFVI deployment.

Figure 16: UCS B-Series Topology



For C-Series pods, each host has a 2x10-GE Cisco network card 1227 from which the installer creates two vNICs for each network to ensure that the network topology has built-in redundancy. The provider network, if needed, is also created from the same network card. Each link of a given network type terminates to a unique Cisco Nexus 9000 switch, which acts as the ToR. The Cisco Nexus 9000s are configured in VPC mode to ensure that the network redundancy. The networking redundancy is extended to the management node, which has a redundant vNIC for the installer API and management or provisioning networks. The following figure shows the C-Series topology.

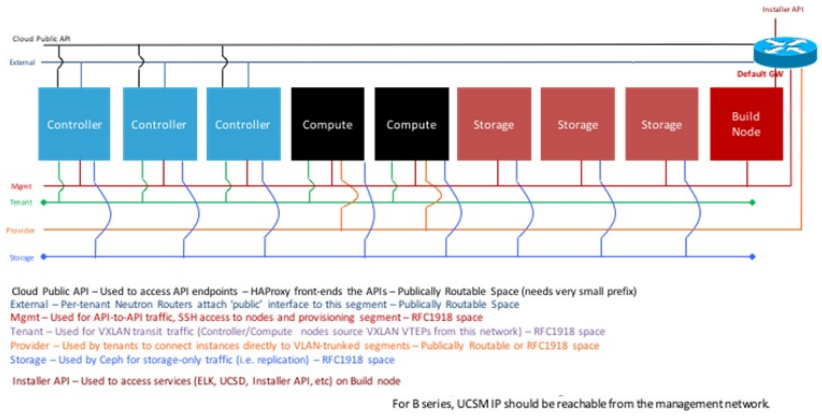
Figure 17: Cisco NFVI C-Series Topology



Note While the figure depicts UCS 220 M4s as the controller and compute, it also supports UCS 240 M4s as control and compute nodes.

Cisco NFVI uses multiple networks and VLANs to isolate network segments. For UCS C-series management and storage nodes, VLANs are trunked between the ToR switches and the Cisco VICs on the C-Series nodes. For UCS B-series controllers and compute nodes, VLANs are trunked between the ToR switches, the UCS Fabric interconnects, and the B-Series blades. The figure shows the network segments and how each node is attaches to them. The network segments are VLANs that are trunked between the respective upstream switch/FI and the C-Series or B-Series node.

Figure 18: Network and VLAN Layout for Combined C-Series and B-Series Installation



Cisco NFVI High Availability

Cisco NFVI high availability (HA) is provided by HAProxy, a single-threaded, event-driven, non-blocking engine combining a fast I/O layer with a priority-based scheduler. HAProxy architecture is layered with bypass mechanisms at each level to ensure that the data does not reach higher levels than needed. Most processing is performed in the kernel.

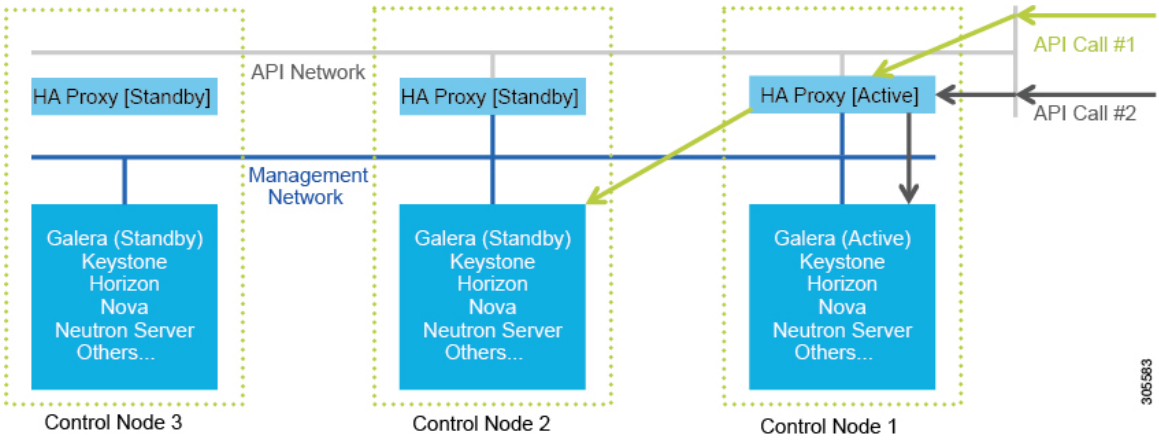
The following figure shows a detailed view of Cisco NFVI controllers connecting to the API and Management and Provisioning network. It also shows how the bridges are configured and the roles of the HAProxy container and network namespace. The dedicated HAProxy container network namespace was created to avoid split default gateway problems. The namespace allows API segment ingress and egress traffic to have a different default gateway than the one configured on each controller host for non-API traffic. In the illustration, two of the three Cisco NFVI controllers have HAProxy containers and a dedicated Linux network namespace. (Cisco NFVI supports three HAProxy containers).

In the figure, Control Node 1 is attached to the API network segment through the br_api bridge. The br_api bridge connects to the Linux network namespace where the HAProxy container has an interface that is mapped through the api < > api_out interface mapping. The HAProxy container has a default gateway configured that points to the upstream API Layer 3 First Hop Redundancy Protocol (FHRP) VIP. This gateway is used for the HAProxy container incoming and outgoing API traffic.

Outside traffic coming in through the API interface is routed into the API network. The traffic traverses the br_api bridge, goes into the Linux network namespace and then the API VIP (based on the IP address or port)

that is listening on the HAProxy container. The HAProxy container establishes a connection with the backend API endpoint (for example, the OpenStack Horizon dashboard) and the return traffic passes through the container and back out the API network following the default gateway for the container on the API network. All other non-API traffic such as the management access over SSH to the Cisco VIM controller comes into the management or provisioning network and access the node directly. Return traffic uses the host-level default gateway that is configured on the Linux (RHEL) operating system.

Figure 19: HAProxy Control Node Flow



If an HA event occurs in a Cisco NFVI pod, Cisco VIM automatically shuts down machines by failing over services. Examples include:

- For API servers, HAProxy automatically ensures that the other redundant control services handle requests, avoiding the shutdown/terminated/non-responding one.
- For quorum services, such as Galera, the remaining members of the quorum continue to provide service and HAProxy ensures that new requests go to the remaining processes.
- For an active/standby process such as HAProxy, the system moves the endpoint IP to a standby copy and continues to operate.

All these behaviors are automatic and do not require manual intervention. When the server is restarted, the services automatically come into service and are added to the load balancing pool, joining their quorums or are added as backup services, depending on the service type.

While manual intervention is not needed, some specific failure scenarios (for example, Mariadb, rabbit) can cause problems that require manual intervention. For example, if a complete network failure occurs, the Galera and RabbitMQ clusters can go into three-way partition. While the Cisco NFVI cluster is resilient to single-point failures, two switches failing simultaneously—something highly unlikely in long-running systems—can sometimes happen due to administrative error, in which case, manual intervention is needed. To repair the pod, the management node must be up and running and all the nodes accessible through password-less SSH from the management node. From the installer<tagid> dir, execute:

```
# ciscovim cluster-recovery
```

Control nodes recovers after the network partitions are resolved. After executing this command, control nodes services come back to working state. To make sure that the Nova services are good across the compute nodes, execute the following command after sourcing /root/openstack-configs/openrc:

```
# nova service-list
```

To check for the overall cloud status, execute the following:

```
# ciscovim cloud-sanity create test all
```

To view the results of cloud-sanity, use the following command:

```
#ciscovim cloud-sanity show result all –id <uid of the test >
```

Cisco NFVI Storage Node Overview

Block Storage

Cisco NFVI storage nodes utilize Ceph, an open source software for creating redundant, scalable data storage using clusters of standardized servers to store petabytes of accessible data. OpenStack Object Storage is a long-term storage system for large amounts of static data that can be retrieved, leveraged, and updated. It uses a distributed architecture with no central point of control, providing greater scalability, redundancy, and permanence. Objects are written to multiple hardware devices, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally by adding new nodes. if a node fail, OpenStack replicates its content across other active storage nodes. Because Ceph uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used in lieu of more expensive equipment.

Cisco NFVI storage nodes include object storage devices (OSDs), hard disk drives (HDDs), and solid state drives (SSDs). OSDs organize data into containers called objects that a user or application determines are related. The objects reside in a flat address space where they all exist at the same level and cannot be placed inside one another. Each OSD has a unique object identifier (OID) that allows the Cisco NFVI control node to retrieve it without knowing the physical location of the data it contains.

HDDs store and retrieve digital information using one or more rigid rapidly rotating disks coated with magnetic material. The disks are paired with magnetic heads arranged on a moving actuator arm, which read and write data to the disk surfaces. Data is accessed in a random-access manner; individual data blocks can be stored or retrieved in any order and not only sequentially. HDDs are a type of non-volatile memory, retaining stored data even when powered off.

SSDs are solid-state storage devices that use integrated circuit assemblies as memory to store data persistently. SSDs primarily use electronic interfaces compatible with traditional block input/output (I/O) hard disk drives, which permit simple replacements in common applications.

Cisco NFVI storage nodes are managed by the control node applications including Ceph monitoring dashboard, Glance, and Cinder. The Ceph monitoring dashboard provides a view into the overall storage node health. Glance virtualizes pools of block storage devices and provides a self-storage API to request and consume those resources. Cinder is an OpenStack block storage service designed to present storage resources to the OpenStack compute node.

In Cisco VIM, depending on the needs of the user, the number of OSDs a pod can have is between 3 and 20. From release Cisco VIM 3.0.0 onwards, you can choose to have multi-backend Ceph in the same pod, to support different I/O requirements. Currently, this is a day-0 decision. You must decide whether to start with single or multi back-end ceph, with a minimum of three nodes for each backend type. Only 2 backends (one of type HDD and another of type SSD) for each pod is supported. For details on how to use HDD or SSD based ceph, see *Cisco Virtualized Infrastructure Administrator Guide*.

Cisco VIM supports NetApp devices running ONTAP 9.X or higher. NetApp devices are added as an alternate to Ceph for block storage. Cisco VIM has been integrated and tested with FAS2650 SKU of NetApp, however it does not preclude Cisco VIM from working with SKUs of NetApp that are compatible FAS2650. Now, you have to choose the blockstorage and the hardware from Day 0.

Object Storage

Cisco VIM provides an integration with SwiftStack, an object storage solution. In this case, the SwiftStack is installed and managed outside the Cisco VIM ahead of time, and the VIM orchestrator adds the relevant Keystone configuration to access the SwiftStack endpoint. In addition to Keystone integration, the Cinder service is also configured to support backup of the volumes to SwiftStack object store. In the current integration, the SwiftStack endpoint has to be in a network routable to/from the CiscoVIM API network (as the VIM API is the same as the Keystone public endpoint network). In the current release, because of limitations in SwiftStack, Cisco VIM is integrated only with KeystoneV2.

In Cisco VIM, you can choose to use Solidfire as an option for block storage along with Ceph. In this scenario, the backend for Glance is Ceph, and the customers have a choice for the Cinder backend to be Ceph or Solidfire. The Cinder block storage service manages the creation, attachment, and detachment of these volumes between a storage system, such as, SolidFire, and different host servers. Also, in Cisco VIM, the date in Solidfire will be backed by Ceph. The Solidfire cluster is pre-deployed and has 2 networks: management and storage. It is recommended that:

- The storage network for Cisco VIM is same as that for Solidfire.
- The management network for Solidfire is reachable from Cisco VIM control nodes.

Overview to Cisco Virtual Topology System

The Cisco Virtual Topology System (VTS) is a standards-based, open, overlay management and provisioning system for data center networks. It automates the data center overlay fabric provisioning for both physical and virtual workloads.

Cisco VTS provides a network virtualization architecture and software-defined networking (SDN) framework that meets multitenant data center cloud service requirements. It enables a policy-based approach for overlay provisioning.

Cisco VTS automates network overlay provisioning and management tasks, integrates with OpenStack and simplifies the management of heterogeneous network environments. Cisco VTS provides an embedded Cisco VTS GUI and a set of northbound Representational State Transfer (REST) APIs that is consumed by orchestration and cloud management systems.

Cisco VTS architecture has two main components: the Policy Plane and the Control Plane. These perform core functions such as SDN control, resource allocation, and core management function.

- Policy Plane—Enables Cisco VTS to implement a declarative policy model that captures user intent and converts it into specific device-level constructs. Cisco VTS includes a set of modular policy constructs that can be organized into user-defined services for use cases across service provider and cloud environments. The policy constructs are exposed through REST APIs that is consumed by orchestrators and applications to express user intent, or instantiated through the Cisco VTS GUI. Policy models are exposed as system policies or service policies.
- Control Plane—Serves as the SDN control subsystem that programs the various data planes including the VTFs residing on the x86 servers, hardware leafs, DCI gateways. The control plane hosts the Cisco IOS XRv Software instance that provides route peering capabilities between the DCI gateways or to a BGP route reflector. (Cisco IOS XRv is the virtualized version of Cisco IOS XR Software.) The control plane enables an MP-BGP EVPN-based control plane for VXLAN overlays originating from leafs or software VXLAN tunnel endpoints (VTEPs)

The Cisco NFVI implementation of Cisco VTS includes the VTS Virtual Topology Forwarder (VTF). VTF provides a Layer 2/Layer 3 (L2/L3) software switch that can act as a software VXLAN terminal endpoint

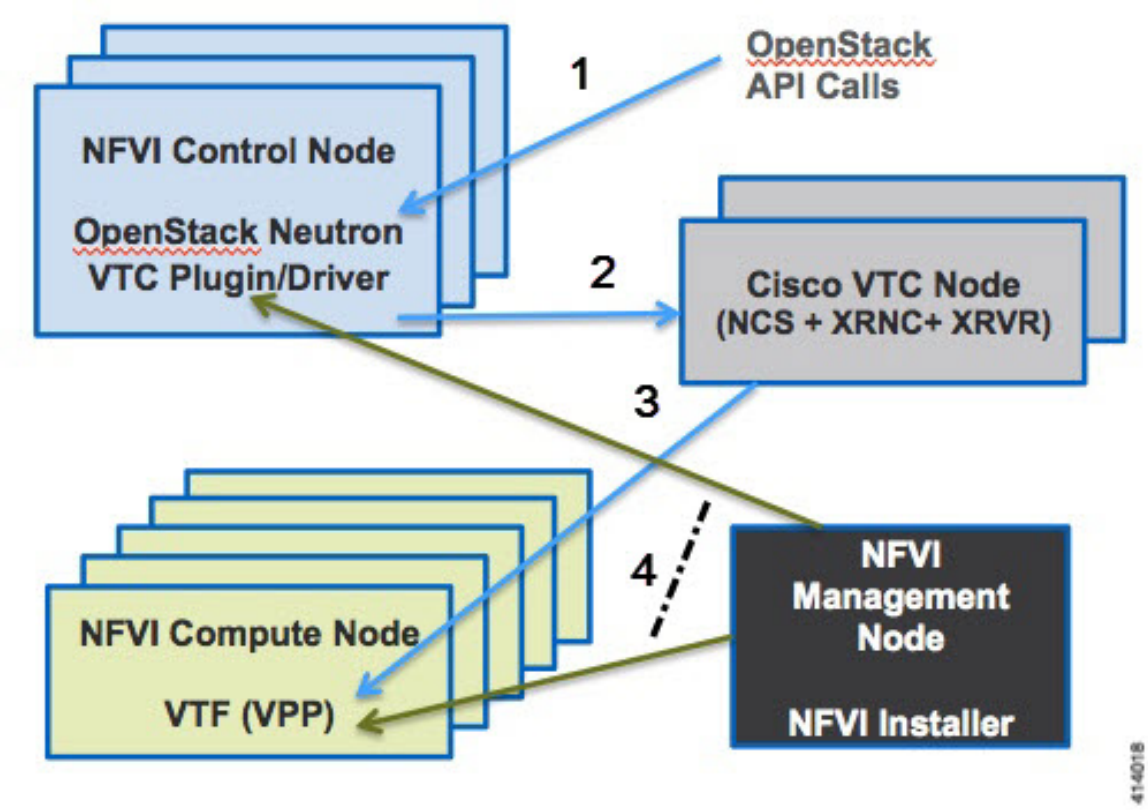
(VTEP). VTF is a lightweight, multitenant software data plane designed for high performance packet processing on x86 servers. VTF uses Vector Packet Processing (VPP). VPP is a full-featured networking stack with a software forwarding engine. VTF leverages VPP and the Intel Data Path Development Kit (DPDK) for high performance L2, L3, and VXLAN packet forwarding.

VTF allows Cisco VTS to terminate VXLAN tunnels on host servers by using the VTF as a software VXLAN Tunnel Endpoint (VTEP). Cisco VTS also supports hybrid overlays by stitching together physical and virtual endpoints into a single VXLAN segment.

The figure below shows the Cisco VTS architecture and high-level flow when installed in Cisco NFVI. Cisco VTS is installed on separate UCS servers, the Virtual Topology Controller plugin is installed on the control node, and the VTF is installed on the compute node.

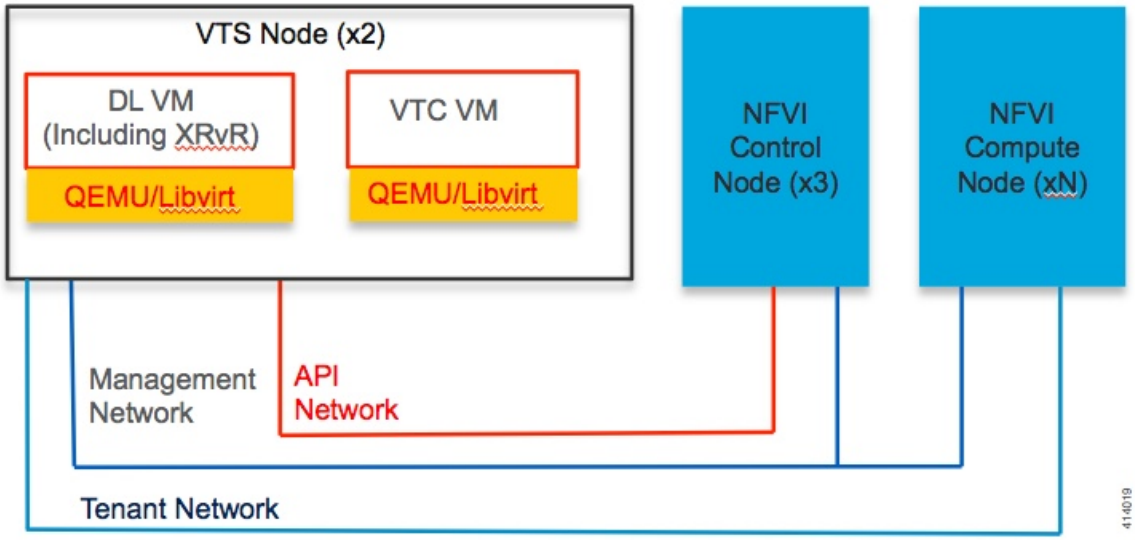
1. The OpenStack user invokes the OpenStack Neutron API.
2. Neutron uses the VTS plugin and driver to make calls to the VTC REST API.
3. VTS control components interact with the VTF agent to carry out the corresponding dataplane setup.
4. During Cisco NFVI installation, the Cisco NFVI Installer installs the OpenStack Neutron VTC plugin and driver on the Cisco NFVI controller node, and installs the VTF component (including VPP) on the Cisco NFVI compute node.

Figure 20: Cisco VTS in Cisco NFVI



The following illustration shows that the Cisco NFVI networking after the Cisco VTS is installed. The SDN controller nodes are an addition to the existing Cisco NFVI pod.

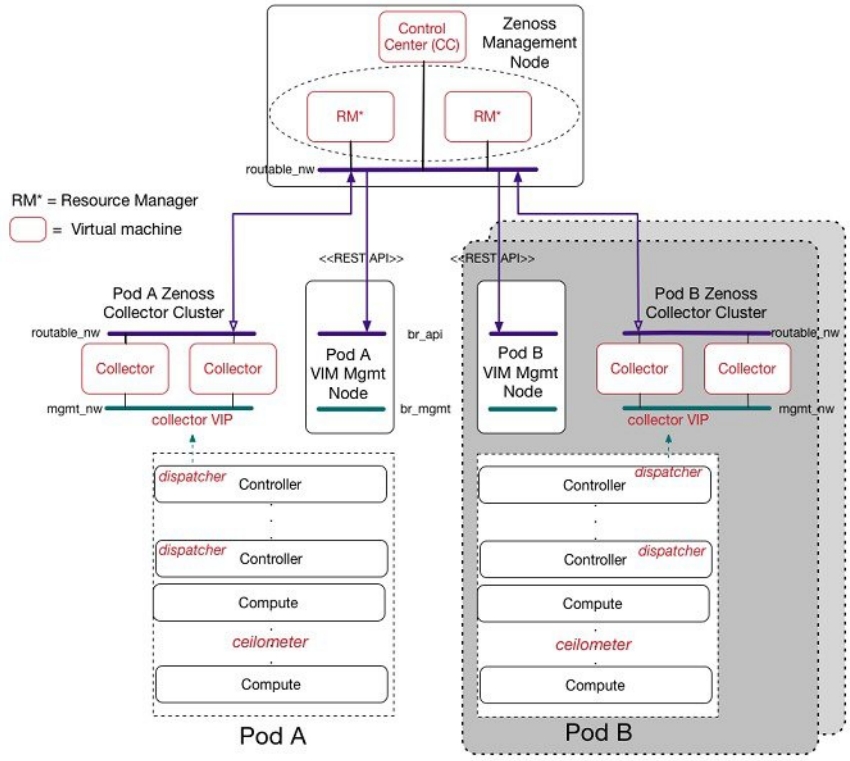
Figure 21: Cisco VTS Networking Inside Cisco NFVI



Overview to Cisco NFVIMON

Cisco VIM solution uses Cisco NFVI Monitor (NFVIMON) to monitor the health and performance of the NFVI. This includes monitoring both the physical and logical components of one or multiple NFVI pods. NFVIMON feature is enabled by the Zenoss which provides for extensive monitoring and collection of performance data for various components of the cloud infrastructure including Cisco UCS blade and rack servers, service profiles, Nexus top of rack switches, fabric interconnects, and also the OpenStack instances. The monitoring system is designed such that it can monitor single or multiple pods from a single management system. NFVIMON is integrated into Cisco VIM as an optional component. NFVIMON is enabled by extending the `setup_data.yaml` file with relevant information. To enable the NFVIMON, refer to *Enabling NFVIMON on Cisco VIM*. Also, NFVIMON can be enabled on an existing pod, through the reconfigure option. To reconfigure through Insight UI, refer to *Reconfiguring Optional Services*. Then, the pod is added as a new VIM resource to be monitored in the Monitoring UI.

Figure 22: NFVIMON Architecture



The NFVIMON architecture supports monitoring of one or more Cisco VIM pods. There is no limit on the number of pods, but note that the setup supports up to ~ 25000 managed resources across pods, where a managed resource is a physical device, network device or virtual machine tracked from a monitoring perspective.

NFVIMON consists of four components: ceilometer services (for data collection), collector, resource manager (RM), and control-center (CC) with Cisco Zenpacks. As NVIFMON is a third-party software, its integration with the VIM is loosely coupled and the VIM automation only deals with installing the ceilometer service software required to monitor the pod.

The installation of other NFVIMON components (collector, RM, and CC with Cisco NFVI Zenpacks) are Cisco Advance Services led activity and those steps are outside the scope of the current install guide. Make sure that you have engaged with Cisco Advance Services on the planning, image information (of collector with Cisco NFVI Zenpacks and RM), and installation of the NFVIMON accessories along with its network requirements.

Start with one Cisco VIM pod (Pod A in the picture) and two external nodes (one to host 2 Collector VMs and one for remote management to host 1 control-center with Cisco Zenpacks and 2 RM VMs) of multiple pods.

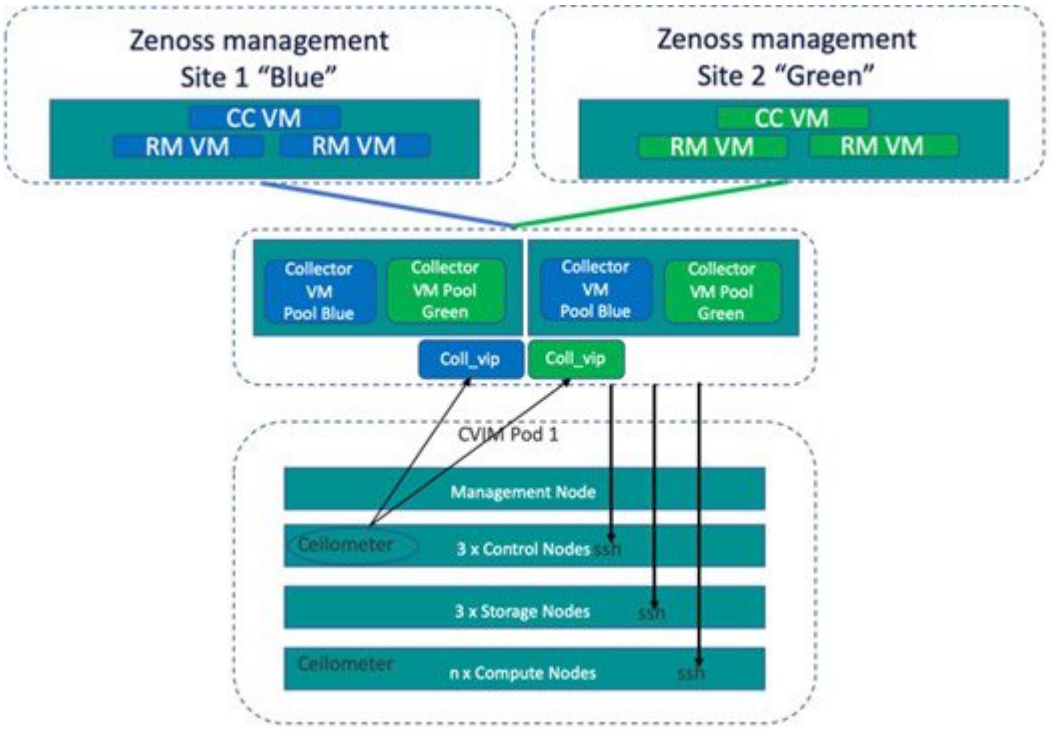
Monitor the Cisco VIM pods at the time of installation if NFVIMON is enabled, or by adding NFVIMON after installation. Install the collectors manually in the external collector node, so that the pod is added for monitoring in the control center.

Note From Cisco VIM 3.2.0, you can use non-root admin keys for monitoring purposes.

Note From Cisco VIM 3.4.1, you can run NFVIMON along with CVIM MON. This capability eases the transition from NFVIMON to CVIM MON and allows operators to evaluate CVIM MON while using NFVIMON.

Overview to Cisco NFVIMON High Availability

NFVIMON supports the functionality of high availability (HA). HA is achieved through dual polling of the redundant collectors over an active-active deployment. VM is deployed between the two physical collector servers with two sets of collectors. Two separate Zenoss CC-RMs are connected to one set of collectors each, to aid in simultaneous monitoring of the pod. Ceilometer is deployed in Cisco VIM pod such that it sends data to two separate collector VIPs simultaneously. To enable the NFVIMON, refer to Enabling NFVIMON on Cisco VIM. The NFVIMON HA architecture is depicted in the below figure.



Note Pods running with NFVIMON in standalone mode, cannot be moved to HA mode through reconfiguration.

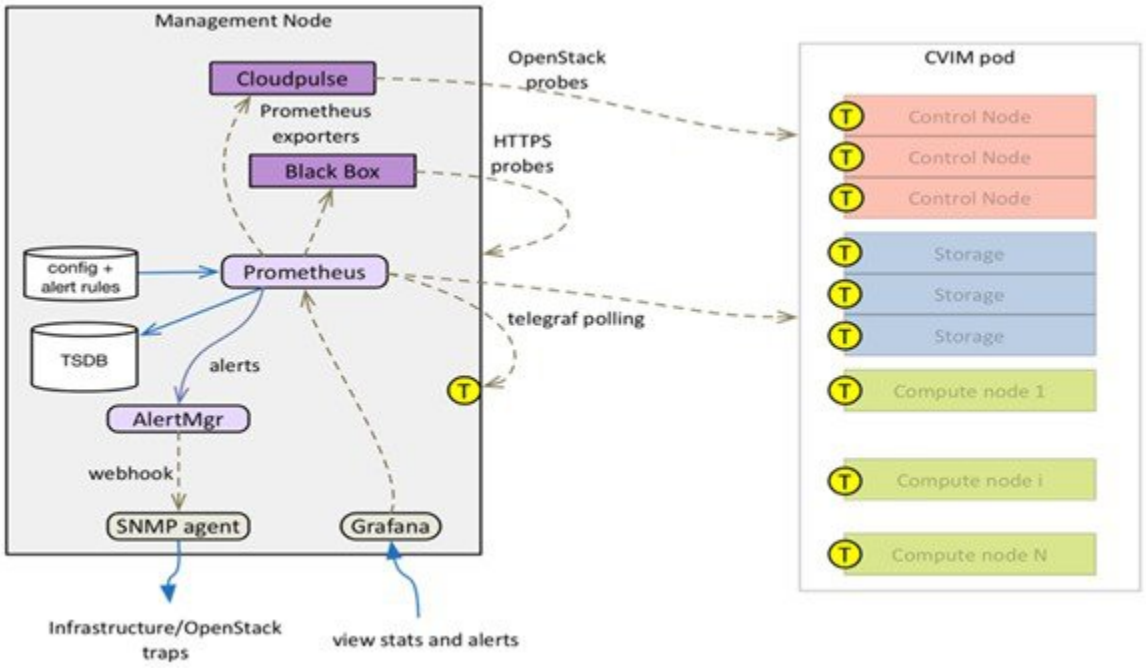
You can enable NFVIMON HA on day-0 or day-1, when the pod is not running with NFVIMON in the first place.

Overview to CVIM-MON

You can deploy Cisco VIM with a lightweight pod-level monitoring solution known as CVIM-MON which is based on the open source PTG stack (Prometheus, Telegraf, Grafana). This solution is available as an add-on from both commercial and feature point of view, and provides the following services:

- Infrastructure-level metric collection based on metric collection agents installed on all nodes in the pod and on specialized collectors running on the management node.
- Metric aggregation into a time series database (TSDB) installed on the management node.
- Rule-based alerting engine integrated in the management node.
- TSDB visualization web server installed on the management node with pre-defined dashboards customized for Cisco VIM.

Figure 23: CVIMMON Architecture



All CVIM-MON components are containerized, except for the Telegraf agents which run on bare metal on all nodes in the pod (including the management node). The two sub-components of CVIM-MON are:

CVIM_MON: Provides the base functionality of monitoring and KPIs.

SNMP: It is enabled only if CVIM_MON is enabled.

Comparative Analysis

The comparison of the two monitoring solutions of Cisco VIM is listed below:

Table 6: Comparison of CVIM-MON and NFVIMON

Features	CVIM-MON	NFVIMON/Zenoss
Open source	Yes	Yes
Collector	Telegraf and Prometheus exporters	Direct ssh to each node
Metrics manager	Prometheus	Zenoss
TSDB	Prometheus	Zenoss
Typical metric frequency	Few seconds or more	Few minutes
Web UI	Grafana	Zenoss
Smart metrics	Yes	No
Alerts	Yes	Yes
SNMP traps	Yes	No
Installation	Integrated with Cisco VIM	External/separate
Hardware requirements	Runs on management node	Requires additional servers

Note From Cisco VIM 3.4.1, you can run NFVIMON along with CVIM MON. This capability eases the transition from NFVIMON to CVIM MON and allows operators to evaluate CVIM MON while using NFVIMON.

TSDB size and Retention Policy

The size of the TSDB depends on the frequency of the polling (configurable) and the number of compute nodes. By default, the metrics collected in each management node are kept for 15 days.

Smart Metrics

The Cisco VIM deployment blueprint assigns different roles to different hardware or software resources for operational and optimization purposes. CVIM-MON leverages the metric labelling feature in Telegraf and Prometheus, to associate important contextual information with the metrics associated to the resources.This labelling enables monitoring the pod in a precise manner than with traditional unlabelled metrics.

Node Type Label

The nodes in a Cisco CVIM pod can play different roles based on the deployment model. All metrics originating from a node are labelled with the node type (label name = "node_type") and the node name (label name="host").

The following node types are defined:

Table 7: Node Type and its metric source

Node Type	Source of Metric
mgmt	Management node
controller	Controller node
compute	Compute node
storage	Storage node
aio	all-in-one node(micro-pod deployment)
hc	hyper-converged node (hyper-converged deployment)

CPU Role Label

CPUs in a Cisco VIM pod are statically categorized to perform specific functions. This partitioning is critical to guarantee proper level of service for each subsystem independent of the load in the other subsystem. For example, it is imperative to isolate the CPUs reserved for the VPP virtual switch, from any other activity on the same compute node, to guarantee the virtual switch forwarding performance. The CPU metrics are labeled with a role (label name = "role") to indicate the function of each CPU. This allows to aggregate CPU metrics based on category, which is a lot more useful than aggregating all CPUs.

This categorization cannot be done with unlabeled metrics (by reading CPU time series from a TSDB), due to the following reasons:

- Identification of CPU role based on the core number.
- Existence of multiple types of nodes.
- Each node type has a different CPU partitioning map. The CPU partitioning map may depend on the Cisco VIM release default mapping or customer specific deployment configuration (for example, on a hyper converged node, the number of cores reserved for CEPH can vary from deployment to deployment).

CVIM-MON uses the following roles to label CPU metrics:

Table 8: Role label and static CPU assignment

Role	Static CPU Assignment
host	System and OpenStack tasks
ceph	CEPH OSD tasks (note that ceph-mon is in the host category)
vpp	VPP virtual switch
vm	VM vCPUs
mgmt	Management tasks on the management node

Metrics Collection

Telegraf Metrics

CVIM-MON collects hundreds of different metrics from each node through the Telegraf plugin. The metrics range from low-level kernel to infrastructure services. The interval between metrics collections is configurable between 10 seconds to 5 minutes.

The following table describes the Telegraf plugins installed as part of the CVIM-MON deployment:

Table 9: List of plug-in and their metric name

Plug-in	Metric Name	Notes
ceph	ceph_osdmap_* ceph_pgmap_* ceph_pool_* ceph_usage_total_*	Collects performance metrics from the MON and OSD nodes in a Ceph storage cluster
cpu	cpu_usage_*	Detailed stats for every CPU (with role label)
conntrack	conntrack_ip_conntrack_*	Collects stats from Netfilter's conntrack-tools
cvim_net_stats	cvim_net_stats_if_*	Detailed metrics for physical and virtual network interfaces in Cisco VIM environment
disk	disk_*	Detailed stats for every disk
diskio	diskio_*	Disk activity
docker	docker_container_* docker_n_containers	Detailed metrics on running docker containers
exec	directory_plugin_bytes	Monitor EFK and Prometheus own storage usage
haproxy	haproxy_*	
http_response	http_response_*	Monitor HTTP services availability
hugepages	hugepages_*	Monitors huge pages usage per NUMA node
internal	internal_*	Collects metrics about the telegraf agent itself
ipmi_sensor	ipmi_sensor_*	Bare metal metrics, including power usage, fan speeds, temperatures, and voltage

Plug-in	Metric Name	Notes
kernel	kernel_boot_time kernel_context_switches kernel_interrupts kernel_processes_forkewd kernel_entropy_avail	
libvirt	libvirt_*	Nova and libvirt data and metrics from VMs running on compute or aio nodes
linkstate	linkstate_actor linkstate_sriov linkstate_partner	Monitoring LACP, SRIOV links status
mem	mem_*	Host level memory stats
net	net_bytes_* net_packets_* net_conntrack_* net_drop_* net_err_* net_icmp_* net_ip_* net_tcp_* net_udp_*	Metrics about network interface and protocol usage (only for interfaces used by CVIM)
ntpq	ntpq_*	NTP query metrics
openstack	cp_hypervisor_up_* cp_openstack_* cp_ceph_health	OpenStack related metrics, comes as a replacement to cloudpulse
processes	processes_*	
rabbitmq	rabbitmq_overview_* rabbitmq_node_* rabbitmq_queue_* rabbitmq_exchange_*	RabbitMQ metrics, currently disabled by default
swap	swap_*	

Plug-in	Metric Name	Notes
system	system_*	Checks system load, uptime, and number of users logged in

Note All metrics are part of the high frequency collection group. The collection interval is in seconds or minutes:

Table 10: Frequency group and metrics collection interval

Frequency_group	Default Interval	Min	Max
High	15s	10s	medium_frequency
Medium	deprecated	high_frequency	low_frequency
Low	deprecated	medium_frequency	5m

OpenStack and infrastructure service metrics

Each Cisco VIM pod can monitor the essential OpenStack services. CVIM-MON gathers OpenStack services data through a custom telegraf plugin. The following metrics are available in Prometheus:

Metric	Metric Name	Notes
ceph check	ceph_health	Checks if ceph is healthy
hypervisor checks	cp_hypervisor_up	Check the state of each hypervisor.
openstack service	cp_openstack_service_upcp	Checks the state of an openstack service. Monitors nova, glance, cinder, keystone, and neutron
rabbitmq status	rabbitmq_	Describes the state of each rabbitmq server. RabbitMQ monitoring is disabled by default.

Etcd monitoring

When the ML2/VPP Neutron plug-in is deployed, Telegraf is configured to poll directly the etcd cluster to retrieve etcd metrics every 15 seconds.

Alerting Rules

CVIM-MON provides a list of predefined alerting rules that trigger the alerts based on the value of time series metrics polled by Prometheus. To avoid flapping caused by transient conditions, the rules are set to have a grace period and an alert is defined to be in one of the two states:

- Pending — Rule is triggered but the grace period has not expired.

- Fired — Rule is triggered for a period longer than the grace period.

The alerts can be monitored using the web user interface or API and can optionally be converted into SNMP traps. You can configure CVIM-MON to send alerts as SNMP traps to any registered SNMP managers. The maximum number of SNMP managers supported is three, and a combination of SNMPv2 or v3 managers in different servers is supported.

Table 11:

Alert Name	Fault Code	Severity	Description
instance_down	serviceFailure	critical	The node is not reachable or is down, when Prometheus server tries to scrape a target to retrieve its metrics. An instance down means that metrics from that target cannot be retrieved.
disk_used_percent	resourceThreshold	major	The storage device is used at over 90% capacity.
disk_filling_up_in_4h	resourceUsage	critical	The storage device is likely to run out of space in less than 4h
docker_container_down	serviceFailure	critical	The docker container running a Cisco VIM infrastructure service is down. This should never happen and indicates that an infrastructure container is failed or could not start.
link_down_lacp	hardwareFailure	warning	The LACP bonded link is in an error state, if one of the two bonded links is no longer operating properly. For example, the error could be caused by the defective cable connection with the NIC, ToR, or a ToR port misconfiguration. The connectivity may still allow traffic to pass but at half the usual throughput. The defective link must be repaired quickly, to reinstate full bandwidth.

Alert Name	Fault Code	Severity	Description
link_down_sriov	hardwareFailure	warning	The SRIOV link is in down state. Usually indicates an issue with the physical cable wiring or a misconfiguration of the corresponding port on the ToR.
mem_available_percent	resourceThreshold	informational	There is less than 10% of available system memory. Regular 4K pages memory is used by both the system and openstack infrastructure services, and does not include huge pages. This alert can indicate either insufficient amount of RAM or an abnormal memory usage by the system or infrastructure
memory_running_out_in_4h	resourceUsage	critical	This node is likely to run out of system memory in less than 4h. Based on the historical memory usage, this alert predicts that all the system memory will be used up in less than 4h. This condition should never happen and requires immediate troubleshooting by TAC before the system memory runs out.
swap_used_percent	resourceThreshold	warning	The node is using over 80% of the available swap space. Nodes should normally use only very little swap space. More than that the nodes will not use any swapping at all.

Alert Name	Fault Code	Severity	Description
conntrack_percent	resourceThreshold	warning	The node is using more than 80% of the available conntrack objects. This is mostly useful for OVS deployments. This indicates an abnormal use of host kernel conntrack resources.
reboot	hardwareFailure	warning	The node is rebooted in less than 10 minutes. Node reboots should be infrequent and be triggered only by the administrator when the node can safely be rebooted. Spontaneous and spurious node reboots should never happen.
system_n_users	resourceThreshold	warning	The node has more than 10 logged-in users.
ceph_error	serviceFailure	critical	The CEPH cluster is in error state and needs to be repaired immediately.
ceph_warning	serviceFailure	warning	The CEPH cluster is in warning state. Requires attention for the repair to be done.
ceph_osdmap_num_in_osds	resourceThreshold	critical	The CEPH cluster has at least 1 OSD in the OUT state.
ceph_osdmap_num_up_osds	resourceThreshold	critical	The CEPH cluster has at least 1 OSD in the DOWN state.
ceph_pgmap_state_count	resourceUsage	critical	The CEPH cluster has at least 1 placement group that is not in active+clean state
ceph_pgmap_bytes_avail_filling_up_in_4h	resourceUsage	critical	CEPH may run out of space within 4 hours.
ceph_pgmap_bytes_used_percent	resourceThreshold	warning	CEPH used capacity is over 70%.

Alert Name	Fault Code	Severity	Description
ceph_pgmap_bytes_used_percent	resourceThreshold	critical	CEPH used capacity is over 80%.
haproxy_plugin_data_absent	other	informational	Not receiving any metrics from HAproxy for 10 minutes or more (should never happen).
haproxy_active_servers_down	serviceFailure	critical	Indicates that one or more HAProxy active server is not in the UP state.
haproxy_active_servers_backend	serviceFailure	critical	The number of haproxy active server backends is not three.
haproxy_active_servers_galera	serviceFailure	critical	The number of haproxy active galera servers is not one.
haproxy_backup_servers_galera	serviceFailure	critical	The number of haproxy backup galera servers is not two.
http_service_unavailable	serviceFailure	warning	The infrastructure HTTP service at given URL is not responding or is not reachable. This should never happen and may indicate an issue with the availability of the infrastructure service.
rabbitmq_node_running	serviceFailure	critical	At least one of the three rabbitMQ nodes is not running.
rabbitmq_node_mem_used_percent	resourceThreshold	critical	Memory used by rabbitMQ is at 90% of its maximum configured limit.
rabbitmq_queue_consumers	resourceThreshold	critical	One or more rabbitMQ queues have no consumer.
rabbitmq_queue_messages	resourceUsage	critical	The number of queued/unread ready and unacked messages is over 300.

Alert Name	Fault Code	Severity	Description
ntpq_offset	resourceThreshold	warning	The mean offset (phase) in the times reported between the local host and remote peer or server is over 2500 milliseconds.
cp_openstack_service_down	serviceFailure	critical	The indicated openstack service is not reachable and likely to be down.
cp_hypervisor_down	serviceFailure	critical	The Nova hypervisor is down.
certificate_expiring_5d	other	critical	The certificate is expiring in less than 5 days and must be replaced.
certificate_expiring_10d	other	warning	The certificate is expiring in less than 10 days.
certificate_expiring_45d	other	informational	The certificate is expiring in less than 45 days .

CVIM-MON Web User Interface

The CVIM-MON graphical user interface allows the pod administrator to monitor the status of the pod using any web browser. This interface is based on Grafana and comes with a set of predefined dashboards.

Access Login

The CVIM-MON web user interface is available by pointing a web browser to the management node IPv4 or IPv6 address (br_api) at port 3000 using https. To access this interface, enter **admin** as username and password.. The password is auto-generated at the time of deployment and can be retrieved from the Cisco VIM password repository (`openstack-configs/secrets.yaml` file) in the CVIM_MON_PASSWORD entry.

From release Cisco VIM 3.2.1, an additional read-only user is created. To access the interface, enter **cvim** as the username and CVIM_MON_READ_ONLY_PASSWORD (from `openstack-configs/secrets.yaml`) as the password.

Note

- The **Forgot your password?** option in the Grafana login page is disabled.
- New password can be generated for Grafana, by running Cisco VIM reconfiguration with the regenerate secrets option.

Pod <pod-name> Dashboard

The pod dashboard is named as Pod <pod-name> where <pod-name> is configured in setup_data.yaml under the option PODNAME) to provide the following:

- High level view of the pod.
- Total number of nodes grouped by node type.
- Total number of cores grouped by role.
- Total load in the pod or sum of the load for all nodes.
- Average usage of all the CPUs reserved for VMs.
- Hardware information of the pod.
- Dataplane statistics of the pod (Networking metrics like throughputs, errors and packet sizes)

Node Level Metrics Dashboard

This dashboard provides a detailed view of the state of the most important resources for any node in the pod including the management node. A list of drop-down menus allows to select:

- Node to display (only one)
- Disk devices to display (all or any selection)
- Network interfaces to display (all or any selection)
- CPUs to display (all or any selection)

The dashboard provides the utilization charts for the following:

- Alerts
- System
- CPU
- Memory
- Processes
- Disks
- Network interfaces

Pod Level Metrics Dataplane Statistics Dashboard

This dashboard provides a detailed view of the networking metrics and data coming from the libvirt and cvim_net_stats telegraf plugins. The following panels are available as part of the dataplane statistics:

- Top 5 nodes drop rate: Top nodes with physical interfaces TX/RX drops rate out of all TX/RX packets in a 20m timeslot.
- Top 10 VMs drop rate : Top VMs with virtual interfaces TX/RX drops rate out of all TX/RX packets in a 20m timeslot.
- Pod throughput in packet-per-second (pps): Total throughput in pps on all physical interfaces.
- Top 5 nodes throughput in pps: Top nodes throughput in pps on node physical interfaces.
- Top 10 VMs throughput in pps: Top VMs throughput in pps on VM virtual interfaces.

- Pod throughput in bits-per-second (bps): Total throughput in bps on all physical interfaces.
- Top 5 nodes throughput in bps : Top nodes throughput in bps on node physical interfaces.
- Top 10 VMs throughput in bps: Top VMs throughput in bps on VM virtual interfaces.
- Top 5 Nodes error rate: It is the error rate on physical interfaces TX/RX out of all TX/RX packets in a 20m timeslot.
- Average pod packet size: It is calculated from total per interface bytes divided by total packets on all pod physical interfaces.

Node Dataplane Statistics Dashboard

This dashboard provides per node and per VM view of networking metrics and data coming from the libvirt and cvim_net_stats telegraf plugins. The following panels are available as part of the nde dataplane statistics dashboard:

- Two gauges with aggregated (all TX+RX) throughputs in PPS and bps across physical interfaces on the specific node.
- One gauge with total virtual interfaces (attached to VMs) running on the specific node.
- Specific VM drop rate: The specific VMs virtual interfaces TX/RX drops rate out of all TX/RX packets on that VM in a 20m timeslot.
- Node throughput in packet-per-second (pps): It is the total throughput in pps on all physical interfaces on that specific node.
- Node throughput in bits-per-second (bps): It is the total throughput in bps on all physical interfaces on that specific node.
- Average Node packet size: It is calculated from total per interface bytes divided by total packets on all node's physical interfaces.
- VM throughput in packet-per-second (pps) : It is the total throughput in pps on all physical interfaces on that specific VM and per VM interface.
- VM throughput in bits-per-second (bps) : It is the total throughput in bps on all physical interfaces on that specific VM and per VM interface.
- Average VM packet size: It is calculated from total per interface bytes divided by total packets on all VM's virtual interfaces.
- VM error rate: It is the error rate on virtual interfaces TX/RX out of all TX/RX packets in a 20m timeslot.

Specialized Dashboards

Table 12: List of specialized dashboards

Dashboard Name	Description
OpenStack services	Chart shows the state of all OpenStack services, infrastructure containers and hypervisors.

Dashboard Name	Description
Alerts	Alerts that are triggered passed the grace period or pending (triggered but still within their grace period).
HAProxy	Chart to monitor the HAProxy service.
CEPH	CEPH storage chart, for example, overall OSD CPU load.
NTP	Chart to monitor NTP on the pod.
RabbitMQ	Chart related to rabbitMQ
Etcd	Chart related to etcd. Only available for ML2/VPP deployments.
Memcached	Chart to monitor Memcached on the pod.
Advanced Metrics	Chart that monitor the management node activity such as: • Prometheus and Elasticsearch disk usage • Prometheus scraping stats
IPMI	Chart that monitor all the nodes and presents bare metal information: • Temperature • Voltage • Fan Speed • Power

SNMP for Monitoring

Along with CVIM-MON, you can enable SNMP in Cisco VIM to send SNMP Traps to the remote SNMP managers. The SNMP traps are identified from the following, only when the SERVER-MON is enabled in the setup_data.yaml file.

- Alerts collected on Prometheus
- Faults reported by the CIMC of the Cisco C-series-servers via SERVER-MON option

The SNMP Trap sends a notification, when the fault occurs or gets resolved. The notification types are listed below:

- cvimFaultActiveNotif: Notification sent when the fault gets triggered.
- cvimFaultClearNotif: Notification sent when the fault gets resolved.

The SNMP trap contains the following information:

- cvimPodID: PODNAME configured in setup_data.yaml file
- cvimNodeID: Node that generated the fault, or N/A
- cvimFaultSource: Component name that generated the fault
- cvimFaultSeverity: Severity of the fault following the guidelines:
  - emergency (1): System level fault impacting multiple services.
  - critical (2): Critical fault specific to a service.
  - major (3): Component level fault within a service.
  - alert (4): Warning condition for service. It may eventually impact the service.
  - informational (5): Informative message and does not impact any service.
- cvimFaultCode: Code. Guidelines followed for code:
  - other(1) : Type of event not specified in the other labels.
  - resourceUsage(2): Resource usage exhausted event.
  - resourceThreshold(3): Resource threshold reached event.
  - serviceFailure(4): Software failure service event.
  - hardwareFailure(5): Hardware failure event.
  - networkConnectivity(6) :Networking issues.

For more details, refer CISCO-VIM-MIB.my.4.0 definition of the MIB at ftp://ftp.cisco.com/pub/mibs/v2/.

CVIM-MON is integrated into Cisco VIM as an optional component, and is offered as an add-on with additional license. CVIM-MON is enabled by extending the setup_data.yaml file with relevant information. To enable CVIMON, refer to Enabling CVIM-MON on Cisco VIM.

You can enable CVIM-MON on an existing pod through the reconfigure option, if the pod is fresh installed with Cisco VIM 2.4.3 or later versions. To reconfigure through Unified Management, refer to Reconfiguring Optional Services. Then, add the pod as a new VIM resource to be monitored so that it is available through the Unified Management portal.

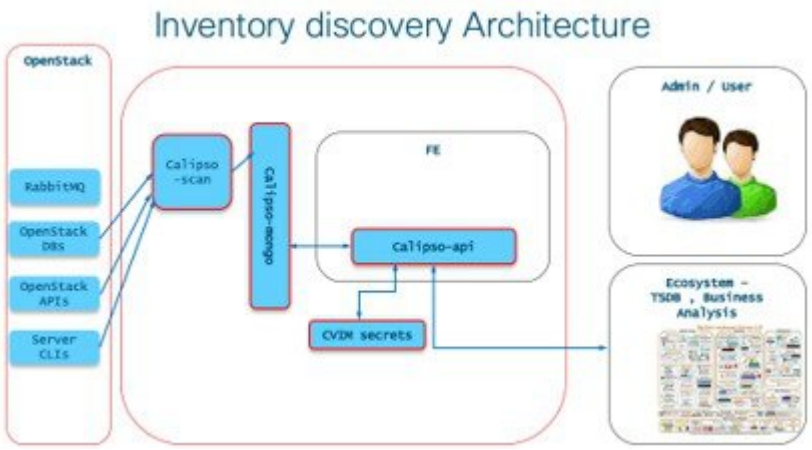
Inventory Discovery Using CVIMMON

From Cisco VIM 3.4.0, CVIMMON includes Inventory Discovery API to extract details from worker level objects, analyze data for links and dependencies ('cliques') using remote HTTP requests.

Inventory Discovery API is a RESTful web API built to offer:

- Resource Oriented Architecture (ROA).
- Declarative API definition using JSON schemas.
- Easy integration for third-party management tools.

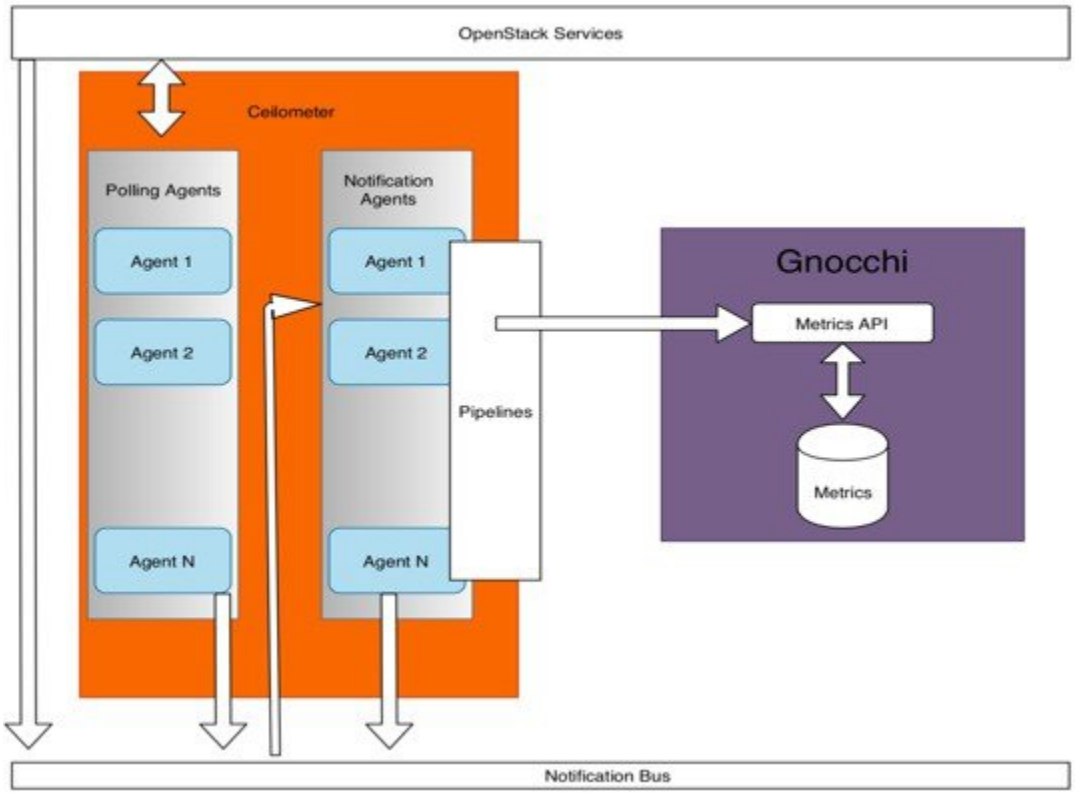
It adds inventory and dependency document repository on top of the time-series repositories already available in CVIMMON. Using RESTful API, information from multiple sources can be viewed from a central location as depicted in the architecture below:



Telemetry Service through OpenStack

Cisco VIM provides telemetry services to collect meters within an OpenStack deployment. Cisco VIM Telemetry service is built on Ceilometer and Gnocchi in OpenStack Queens release. You can retrieve metrics using OpenStack CLI and REST APIs. Pods must have Ceph for persistent storage of the metrics which are collected every five minutes and retained for 48 hours. As Ceph is required for ceilometer, you can install ceilometer as part of fresh installation of the cloud, that is, ceilometer cannot be brought in as a reconfigure option. Also, the ceilometer is supported only on fullon pod. The diagram below illustrates the high-level architecture of the telemetry services.

Figure 24: Architecture of Telemetry Services in Cisco VIM



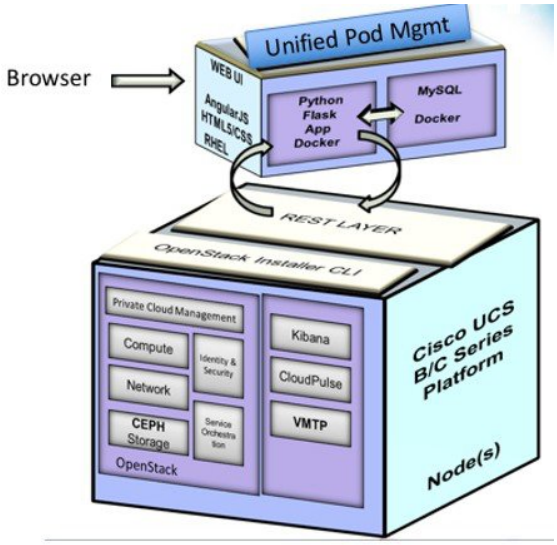
To view the summary of the metrics, see Telemetry for OpenStack of *Cisco Virtualized Infrastructure Admin Guide, Release 3.0.0*.

Overview to Cisco VIM Unified Management

Cisco VIM UM, a light-weight UI, is introduced in Cisco VIM to ease the deployment and management of the NFVI platform. This feature is available as an add-on from both commercial and feature point of view. Also, Cisco VIM Insight offers a single pane of glass service to provide deployment visualization and to manage multiple Cisco VIM pods thereby reducing user-errors.

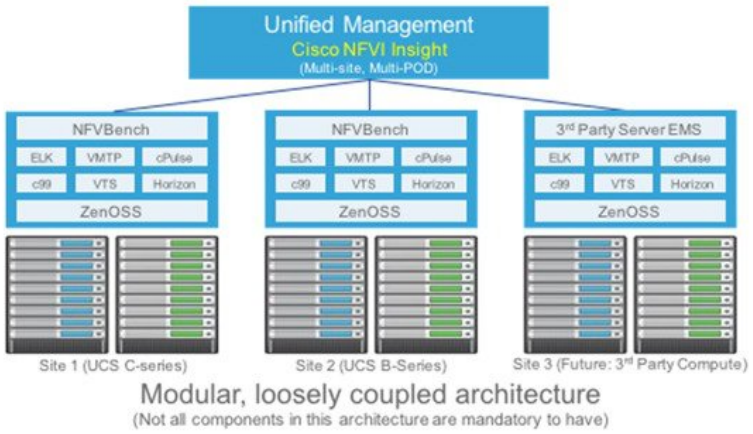
Cisco VIM UM supports multi-tenancy with local RBAC support and is easily integrated with the CiscoVIM REST layer. The container based UI platform is loosely coupled, and can help manage multiple CiscoVIM pods right from day-0, or later in the lifecycle of the cloud.

Figure 25: Cisco VIM UM Interaction with a Pod



The architecture of the CiscoVIM UM is light-weight, hierarchical and scalable. While it introduces an ease of management from the global UI, each local site is autonomous with localized toolsets. The Global Unified Management UI, provides ease of management with multi-site multi-pod capability for distributed NFV deployment at scale. Also, CiscoVIM UM is designed to operate in HA as an option. The platform is a modular, loosely coupled architecture, that will provide the capability to manage multiple pods, with RBAC support as shown in the figure .

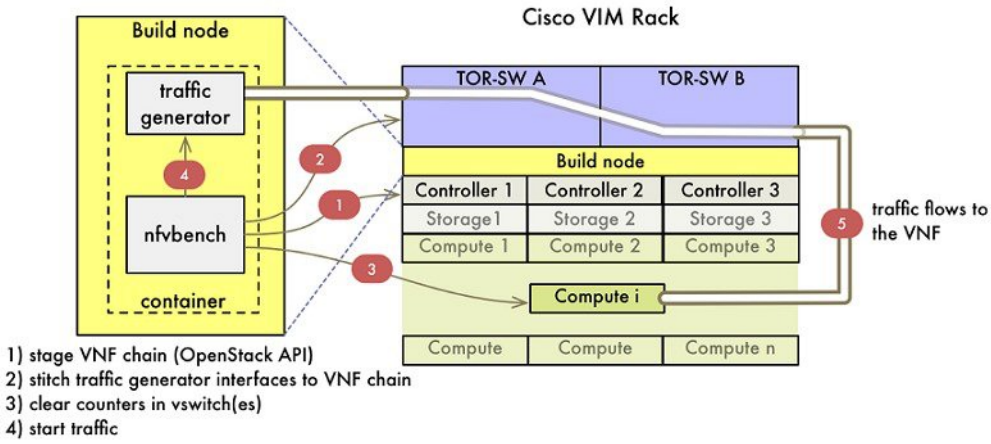
Figure 26: Cisco VIM UM Architecture



Overview to NFVbench

NFVbench is a containerized network benchmarking tool that is introduced in Cisco VIM, to bring consistent methodology to measure the network performance of the cloud. NFVbench is released in a container that is preinstalled on the management node if the NFVBENCH option is selected in the Cisco VIM configuration file.

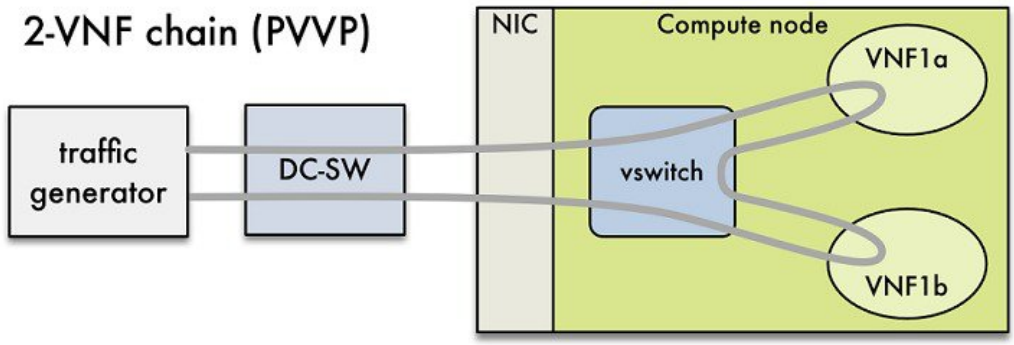
Figure 27: Order of Steps Performed in NFVbench Test



The main goal of NFVbench is to measure the cloud performance that is based on real cloud deployment traffic patterns. During the test, the packet path traverses through every network element that participates in the production environment; that is traffic flows through a switch (ToR) to v-switch on compute node, continues to VM representing any basic VNF in NFV deployment and comes back in similar way on different ports. Network performance or throughput is computed based on sent and received traffic.

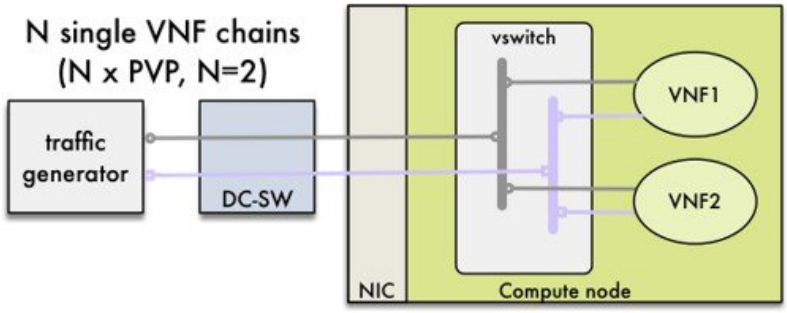
NFVbench can bring up one or more chains of test VMs, where each chain consists of one or two VMs. The example below illustrates a case with a two-VM chain.

Figure 28: Packet Path with Two VNFs



Reports from NFVbench show data measurements from every hop in the path, which makes it easier to detect configuration errors or potential bottlenecks. NFVbench sends UDP packets that are generated by open-source traffic generator (TRex) already included in the container. Advanced testing using NFVbench allows to conduct multi-chain and multi-flow testing. Multi-chain testing enables running multiple parallel independent packet paths at the same time, while the multi-flow testing performs IP ranging in packet headers within every chain. The below figure illustrates a NFVbench result test execution with two parallel chains with one VM each.

Figure 29: Multi-chain Example with Two Chains



NDR/PDR and Fixed Rate Tests

NDR/PDR Test: NFVbench offers a more advanced test (called the NDR/PDR test), provides information about network throughput using any of the standard defined packet sizes - 64B, IMIX,1518B. NDR (No Drop Rate) value represents throughput at which no packets are dropped (satisfied by less than 0.001% of packets being dropped). Similarly, PDR (Partial Drop Rate) represents throughput at which only small number of packets is dropped (less than 0.1% of packets sent).

Fixed Rate Test: NFVbench offers a simple test to run traffic at fixed rate, which verifies that every network component of packet path works properly. It is useful for identifying bottlenecks in the test environment. Traffic generator generates packets at fixed rate for the given time by the user. From the statistics that is collected, drop rates and latencies are computed and displayed.

Both the NDR/PDR Test and Fixed Rate Test types of test provide a way of verifying network performance of NFV solution.

Supported Encapsulation and Protocols

NFVbench supports all networking options that can be deployed with Cisco VIM:

- OVS
- VPP with VLAN or VxLAN
- SR-IOV

Auto-ToR Configuration via ACI API

While the use of ACI plugin brings in the flexibility of dynamic allocation of tenant and provider VLANs on demand, it also ties the OVS version to the ACI plugin. This leads to an extreme tight coupling of Cisco VIM and ACI. Also, with an APIC plugin there are might be gaps to cover certain use-cases, for example, where there is a need to have flexibility of different access type (tagged vs non-tagged) for the same VLAN but for different servers.

To address such use-case or avoid tight coupling of OVS with ACI plugin, an optional solution is available to automate the target VLANs on the right switch port based on server role on day-0 along with corresponding fabric access and tenant policy configurations via the ACI API.

With this option, the setup_data for each Cisco VIM instance is the single source for the server to switch port mappings. This solution can handle switch provisioning with the correct VLANs during addition/removal of

server and provider/tenant VLAN range expansion via reconfiguration option. This solution is based on the fact that the PV (port*VLAN) count in a given ACI Fabric domain is under the scale limits 10000 PV/ToR and 450000 PV/Fabric.

NCS-5500 as a ToR Option

Cisco VIM supports NCS-5500 as an alternate to a Nexus ToR. NCS-5500 is an IOS XR-based router, which is similar to Nexus switches. You can use the 48 10/25G ports or the 6 40/100G uplink ports model to implement NCS-5500 (port-numbers depend on NCS version). Also, other SKUs of NCS-5500 are supported as long as the NCS-5500 software supports the EVLAG feature. NCS-5500 uses the technology of bridge domain to connect to the server. Enable the Auto ToR configuration feature to support NCS-5500 as ToR. NCS-5500 supports a micropod with more computes running on Intel 710 NICs with the mechanism driver of VPP over LACP. The support is extended to include 40G/100G based NCS-5500 SKUs with splitter cables (of 4x10) connecting to the servers, which helps in increasing the server port density by four folds.

Disk Management in VIM

Cisco VIM uses the disk-maintenance tool that gives you the ability to check the status of all hard disk drives present in the running and operational mode in the following nodes:

- management node
- specific or all controller servers
- specific or all compute servers

Status of the disks such as online, offline, rebuilding helps you to identify which particular disks in which slot has potentially gone bad and require to be physically replaced in the server. It can be run on servers that have either a RAID controller or an SAS passthrough controller.

Once the disk is physically replaced, Disk management tool can be used to add the new disk back into the system as part of the RAID system (recommended one server at a time).

Note Disk Maintenance tool is useful only when one or at most two (in RAID6) go bad. Failure of more than one disk at a time puts the entire server in an irrecoverable state. Replace the server using remove and add operations through ciscovim. Disk management is not supported on a third party compute due to the licensing issue with the HPE SmartArray Utility tool.

OSD Maintenance

OSD maintenance tool gives you the ability to check the status of all OSDs and their corresponding physical hard disk drives present in the running and operational storage nodes. The status of the OSDs is reported along with the HDD mapping.

OSD Maintenance tool helps you to identify the status of the OSD (Up or Down) and its corresponding hard disk drive slot in the server that requires to be physically replaced. OSD Maintenance tool can run on servers that have either a RAID or an SAS passthrough controller.

Once the HDD to be physically replaced is identified, the same OSD tool can be used to rebalance the ceph tree, remove the OSD from the cluster, and unmount the disk drive, in preparation for the disk removal. After the disk has been physically replaced, the tool can be used to add the new disk back into the system as part of the Ceph cluster and recreate the OSD (only one HDD/OSD at a time). It ensures to replace a bad HDD, it is not required to remove the ceph cluster from operation and then add it back through remove-storage and add-storage options in ciscovim.

Note OSD tool does not support the replacement of the internal OS drives and the external journal drives, for which you still have to use add or remove of OSD nodes.

Power Management of Computes for C-Series

Cisco VIM pods has many compute servers, but the actual usage of the compute servers are limited at times. To optimize the overall power consumption of the data center, we have to power down the server through an API/CLI.

To prevent the cloud destabilization, you cannot power off all the compute nodes. For example, one cannot power off all the compute nodes, at least one pod has to be Active.

Pod management operation(s) applies to the entire pod during updating and reconfigure, the server.

Updating and reconfiguration are not possible under the following circumstances:

- If one or more compute nodes are powered off.
- Computes on which VMs are running cannot be powered-off.
- Computes with. All-in-one (AIO) nodes in a micro-pod) cannot be powered-off through this API.

When there is a power-off, internally cloud-sanity is run and if the cloud sanity fails, then the power-off action is aborted.

Physical Cores and Memory Reserved for Cisco VIM Infrastructure

Cisco VIM has been tuned to deliver performance from an infrastructure and VNF point of view. The following are the details of the physical cores (regardless of hyper-thread enabled or not) that the infrastructure needs. Number of cores that are reserved for the system (host system + OpenStack services) is 2 in all cases and is included in the count that is shown in the following table.

Table 13: Number of Physical Cores and RAM Reserved for Cisco VIM Infrastructure

Pod Type/Node Types	Control	Storage	Compute	AIO	HC
FullOn	all	all	CPU: 2+V cores RAM: 25+Vr GB	n/a	n/a
Hyper-converged (hc)		n/a		n/a	CPU: 2+C+V cores RAM: 41+Vr GB
Micropod (aio)	n/a	n/a		CPU: Q+C+V cores RAM: 41+Vr GB	N/A
Edge-pod	Q	n/a		n/a	n/a

Table 14: Number of Physical Cores and RAM Reserved for Cisco VIM Infrastructure

Variables	Usage	Valid range	Default
Q	Cores reserved for control role (aio and edge)	2 to 12	2
C	Cores reserved for CEPH (aio and hc)	2 to 12	2
V	Cores reserved for VPP vswitch	2 to 6	2
Vr	RAM reserved for VPP		2GB

For OVS deployments, use V=0 and Vr=0

Some VPP deployments with high throughput requirements may require more than 2 VPP cores.

Cisco VIM Software Hub

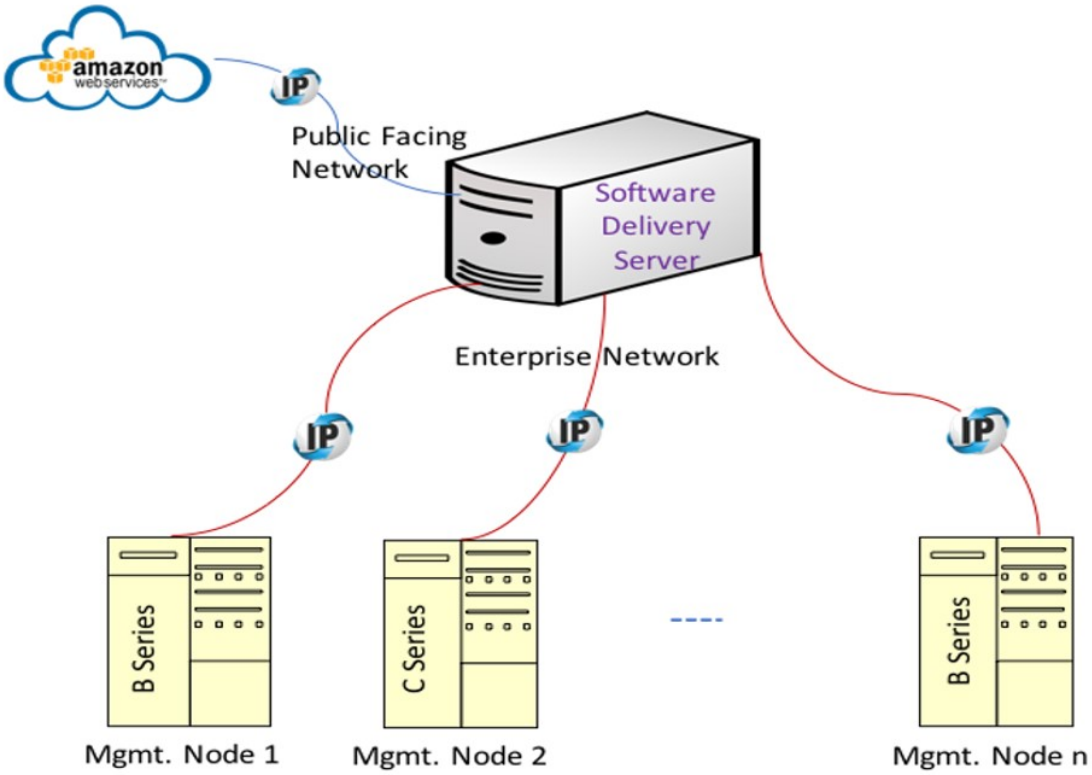
Cisco VIM is supported in an air-gapped (disconnected mode) environment. You can use a USB or Cisco VIM Software Hub for an air-gapped install. When the number of pods is more, shipping USBs for an air-gapped install and update is not scalable. In such scenarios, we recommend that you use Cisco VIM Software Hub.

Cisco VIM Software Hub contains the Cisco VIM release artifacts such as buildnode ISO, Cisco VIM code, docker registry, and docker images. Using the management node, you can access the release artifacts from the Cisco VIM Software Hub.

You can install the artifacts available on the Cisco VIM Software Hub server through a connected or a disconnected install procedure. For a connected install, one end of the Cisco VIM Software Hub server is connected to the internet, and the other end is connected to the datacenter.

The following figure shows the architecture of a connected install.

Figure 30: Architecture of a Connected Install

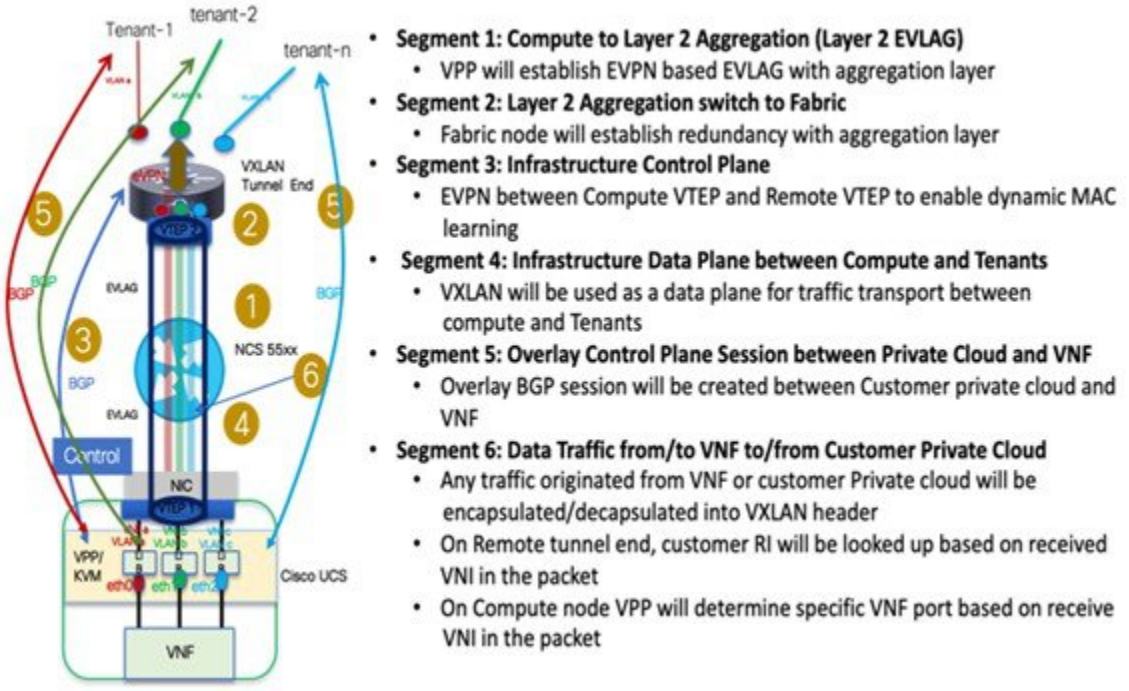


For a disconnected install, both interfaces are private and the artifacts are installed on the Cisco VIM Software Hub using the USB procedure. You must ensure that the ssh interface (br_api) of the management node for each Cisco VIM pod can connect to the enterprise facing interface of the Cisco VIM Software Hub server through Layer 2 or Layer 3 networking. From release Cisco VIM 3.0.0, the Cisco VIM Software Hub is supported over dual-stack network.

Cisco VIM VXLAN EVPN Design

From release Cisco VIM 2.4.3 onwards, seamless connectivity from VNFs of the private cloud to the customer premise private cloud is enabled. The architecture of the Cisco VIM Tenant L2 Connectivity is depicted below:

Figure 31: High Level NFVI Tenant L2 Connectivity Architecture

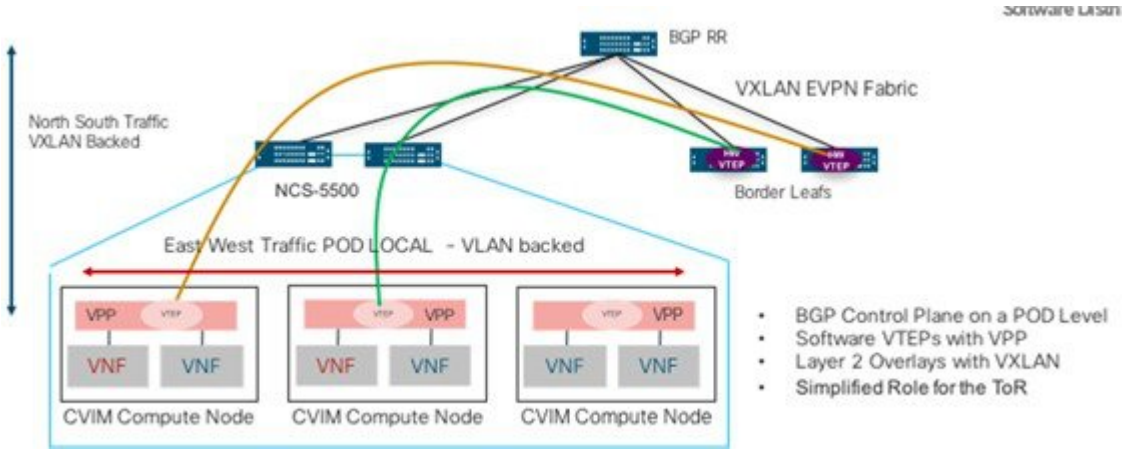


To set up Cisco VIM tenant L2 connectivity architecture, the following assumptions are made:

- OpenStack can manage VLAN allocation.
- You must manage VXLAN network and subnet for overlays, and enable OpenStack to use the EVI/VNID by creating appropriate networks/subnets in OpenStack. Cisco VIM supports VNI ranging from 1 to 65535.
- BGP configuration (peer, ASes) will be provided at the time of Cisco VIM cloud deployment through setup_data.yaml.

VXLAN tunnel is used for traffic between the VNF and customer Private cloud, while the VLAN is used for the traffic within the pod or across VNFs. EVPN is used to share L2 reachability information to the remote end, and Cisco NCS 5500 in EVLAG mode acts as a conduit for the traffic. For the VXLAN/EPVN solution to work, Cisco VIM and VXLAN tunnel peers with an external BGP route reflector to exchange IP address to Mac Binding information as shown in the below figure.

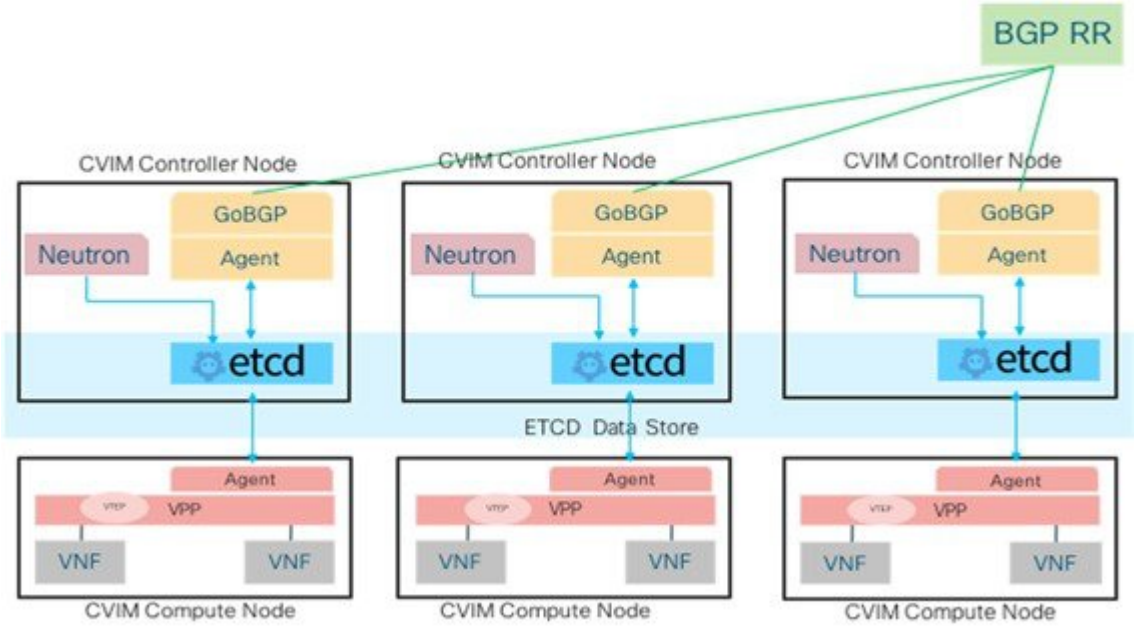
Figure 32: Cisco VIM VXLAN EVPN Setup



From a control plane point of view, three instances of GoBGP (in Active-Active-Active mode) run on the controller nodes to establish L3 peering with the external BGP RR for importing or exporting VxLAN routes into or from Cisco VIM respectively. The imported information is then pushed into etcd, to maintain a single source of the information within Cisco VIM.

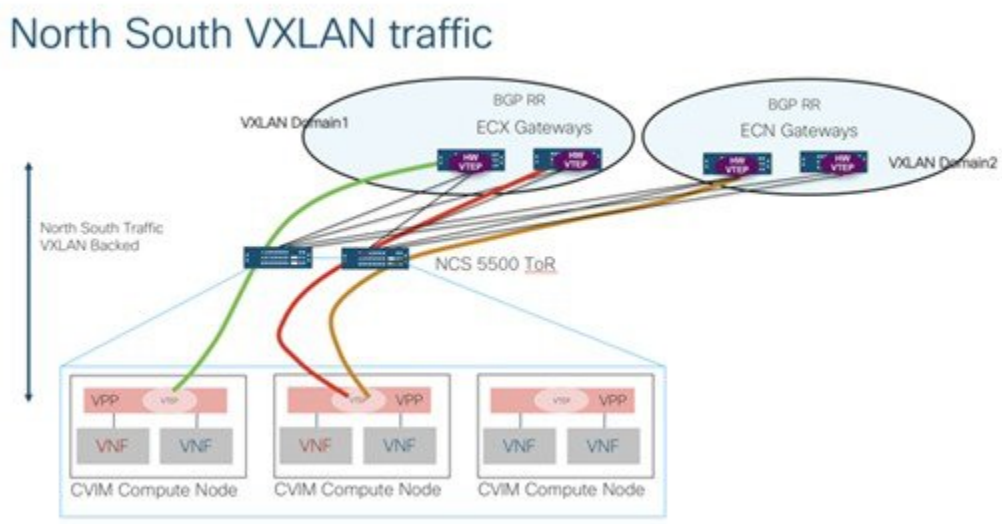
VPP agents create and program VTEP on VPP, and also create a VXLAN tunnel interface for the VM based on the VNI information from Neutron. VPP updates VNF IP/MAC mapping in etcd, which gets exported out through EVPN to the BGP RR.

Figure 33: Cisco VIM VXLAN EVPN Control Plan Design



Multi-VXLAN EVPN Design

From release Cisco VIM 2.4.6 onwards, multiple-AS VXLAN EVPN overlay networks are supported. The following image depicts the schematic view of the multiple-AS VXLAN EVPN overlay network.



One set of VXLAN overlays manage the Cloud exchange traffic, while the other set of VXLAN overlays manage the Cloud management traffic. The multi-VXLAN (multi refers to 2) is used to conserve the number of bridge domains (BD) consumed on the Cisco NCS 5500 ToR.

From the control plane point of view, it is similar to that of a single VXLAN architecture.

The multi-VXLAN EVPN based design optionally supports a static implementation of VXLAN technology through head-end replication (HER). HER helps leverage the VXLAN technology, regardless of the hardware/software limitation in the VXLAN feature set at the remote end of the VTEP tunnel.

With the static information defined in the setup_data, VPP performs the HER to all defined remote VTEPs and updates L2FIB (MAC-IP) table based on flood and learn. If EVPN co-exists with HER, Cisco VIM treats it as if two different sets of BGP speakers exist and provides information from each speaker in the same etcd FIB table.

Only drawback of this implementation is that VPP may peform unnecessary flooding. Cisco VIM uses EVPN as the primary mechanism and HER as the fallback methodology. You can add or remove HER to or from an existing EVPN pod through Cisco VIM reconfigure option.

VPP Port Mirroring Support

From release CVIM 2.4.3 onwards, all the network traffic between the VM and VPP is over a vhost interface which is in memory and does not use a traditional kernel side interface, when VPP is used as the vSwitch in OpenStack. The network interface is no longer on the host and available within VM, to trace packets or capture them for debugging or other administrative purposes.

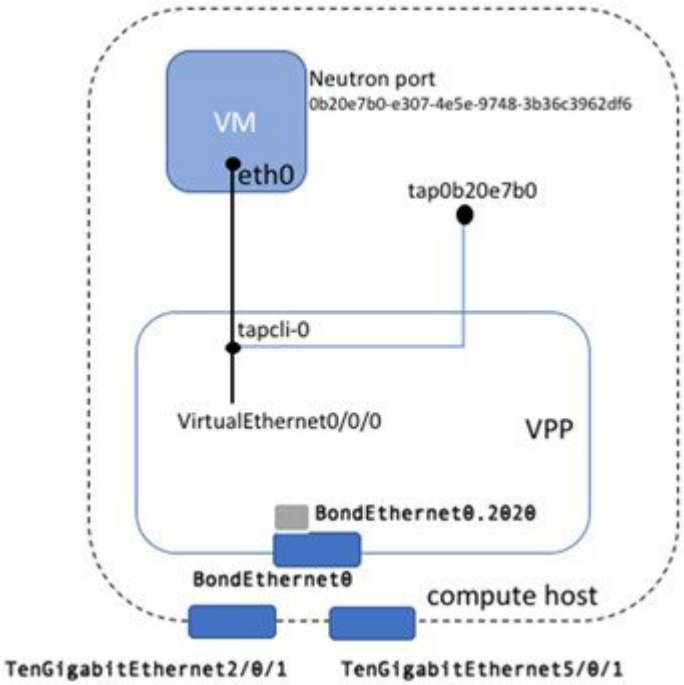
Underlying Architecture of the Port Mirroring Tool

Port mirroring works by setting up the following:

1. A span port on vpp to mirror the VirtualEthernet interface corresponding to the VMs vhost interface. This is a tap interface in VPP

   .
2. A tap device (tap0b20e7b0) on the compute host side is set as a kernel interface. A veth pair is created between the tap device on the VPP side (tapcli-0) and kernel side tap device (tap0b20e7b0) as shown in the below figure.

Figure 34: Port mirror components



Limitations of the Port Mirroring Tool

- The port mirror feature uses tap as the interface type for the mirrored traffic. VPP may drop packets designated for this interface, under high load conditions or high traffic scenarios.
- You can only run the Port mirror CLI tools from the VPP container. This require access to the compute node where the VM is running.
- You can only mirror the neutron ports managed by vpp-agent. This means that these have to be vhost interfaces belonging to Openstack VMs. Non VirtualEthernet interfaces are not supported.

Cisco VIM Segment Routing EVPN Design

Note This feature is released as a tech preview in Cisco VIM 3.4.1.

An important aspect of any Telco cloud is how the cloud is connected to the rest of the SP network. From release 3.4.1, Cisco VIM can attach to an existing Segment Routing (SR) Ethernet VPN (EVPN). This implementation is an evolution of the existing VPP-based standard VLAN and VXLAN EVPN designs. Cisco VIM attaches to the SR EVPN without an additional SDN controller, by peering with the route reflectors of the EVPN and the ToR with BGP Labeled Unicast (BGP-LU). There is no contention with any controller that manages the EVPN. MPLS is used as the data plane for the SR-labelled traffic.

Overview of Cisco VIM VPP Architecture

To connect a Cisco VIM pod to an SR EVPN, the existing VPP forwarding architecture has been enhanced. Networking-vpp is the Vector Packet Processing (VPP) based software accelerated virtual switch that is part of Cisco VIM. The architecture of networking-vpp is similar to a distributed SDN controller. However, it is not a separate controller and is integrated with Cisco VIM as a core component. Networking-vpp makes installation, updates and operational day-2 tasks a seamless experience.

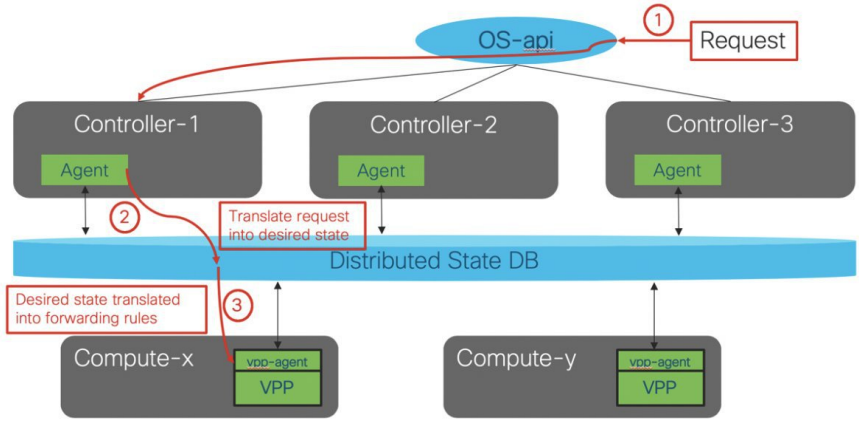
Networking-vpp uses a publish and subscribe model to configure the virtual switches running on each compute node. The controllers translate any requests to the OpenStack API into the desired forwarding behavior of the VPP forwarders and publish that information into the distributed state database. The VPP agents that run on each compute node monitor the distributed state database and program according to the desired state that is published in the distributed state database.

The controllers do not have to:

- Push configuration directly to the compute nodes
- Validate the configuration
- Continuously monitor the configuration of the compute nodes.

This behavior makes networking-vpp extremely efficient, resilient, and scalable. When a compute node restarts, it only needs to look at the distributed state database to check how it forwards traffic to and from the virtual machines running on this compute node and configure itself. All this is transparent to the controllers.

Figure 35: VPP Architecture



The following steps explain the figure above:

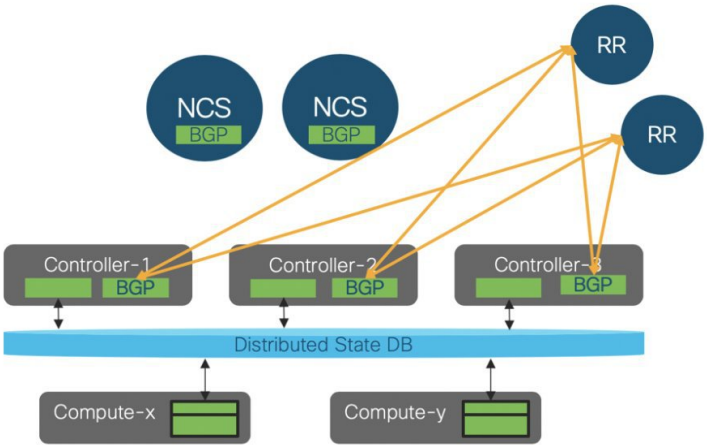
1. The agent running on the controllers sees incoming requests and processes the API calls that relate to the state of the network data plane forwarding.

2. The agent translates the request into the desired state of the network data plane and publishes this state into a distributed state database.
3. The vpp-agents running on the compute nodes watch the distributed state database. When they see a change relevant to the compute node, they translate the desired state and update the VPP forwarding on the compute node.

Overview of Cisco VIM SR EVPN Architecture

To extend the networking-vpp architecture to support Segment Routing over MPLS, Cisco VIM 3.4.1 adds several additional components.

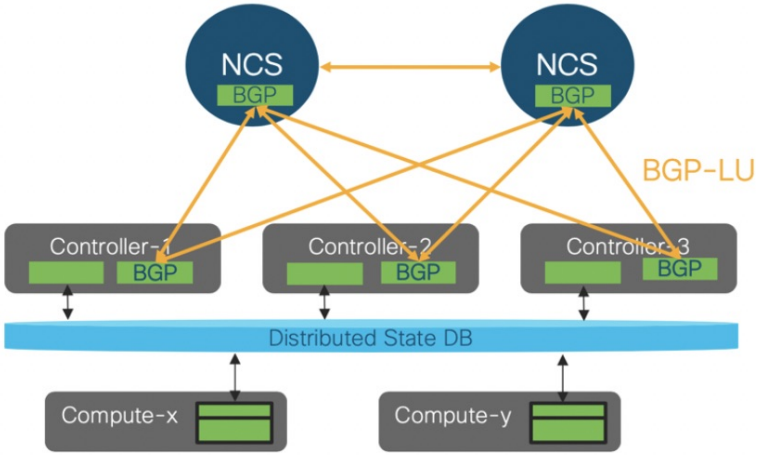
Figure 36: SR EVPN RR Connections



For the control plane, a BGP process runs on each of the three controller nodes. The controllers peer to the EVPN route reflectors. This peering allows the Cisco VIM pod to exchange reachability information with the EVPN. Cisco VIM does an L2 stretch with only single-homed L2 routes. The peering with the route reflectors allows Cisco VIM to exchange Type-2 routes that have MAC to IP bindings. Type-3 route updates are exchanged to handle broadcast, unknown unicast and multicast (BUM) traffic. There are unidirectional MPLS tunnels in both directions.

In addition to the BGP peering with the EVPN route reflectors, the controllers have a BGP Labeled Unicast (BGP-LU) peering with the NCS ToRs to exchange label information.

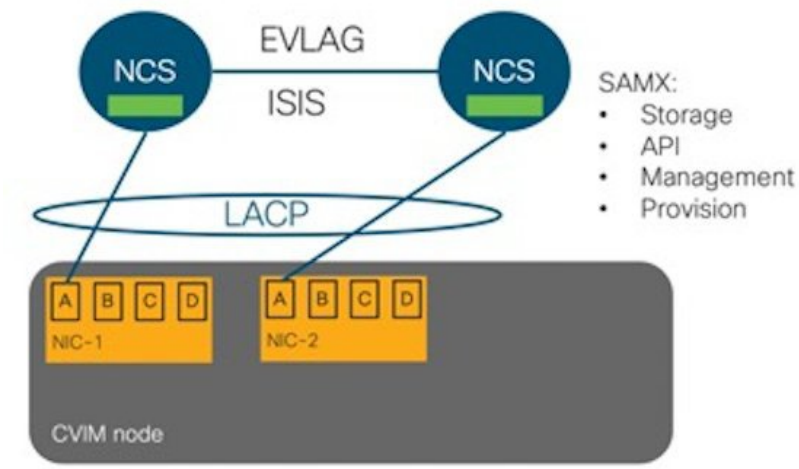
Figure 37: SR EVPN BGP-LU Peering with the ToR



Cisco VIM does not process any SR policies coming from BGP or an SDN controller to program any explicit static or dynamic paths into VPP. BGP-LU and EVPN address family routes are processed by the BGP process running on each of the controllers. The BGP process publishes the SR policy into the distributed state database so that the VPP agents can configure VPP forwarding on the compute nodes to apply the MPLS label to the outbound traffic. The label between the compute node and the NCS ToR represents the path to node SID that identifies the VPP node. Bridge domain is identified by the Openstack Segmentation ID that is used for tenant isolation.

The data plane ports in the Cisco VIM pod are connected to the NCS with an L3 link and EVLAG provides link redundancy to the nodes. The VPP agents encapsulate the traffic leaving the compute node with an MPLS label representing the Segment Routing segment ID. The virtual machines are unaware that traffic will be sent over an SR-enabled network. For traffic arriving on the compute node from the NCS, the MPLS label represents the bridge domain, and the resulting L2 lookup determines the virtual machine to which to forward the traffic after stripping the MPLS header.

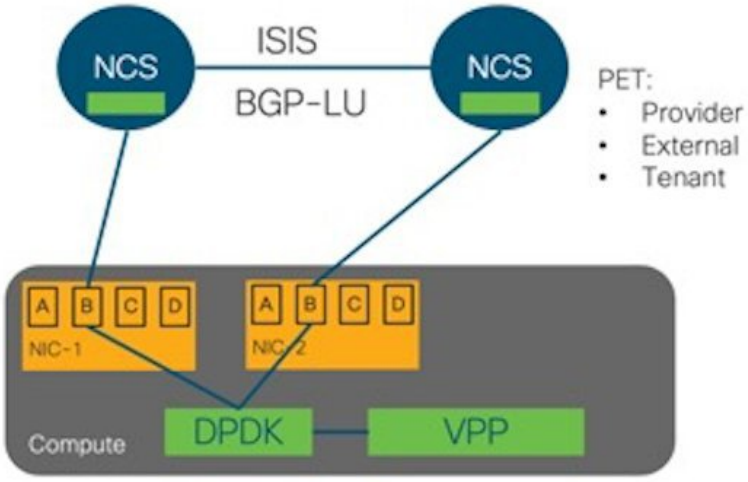
Figure 38: Control Plane Port Connectivity



The control plane ports on the Cisco VIM nodes have two uplinks, one to each NCS. From the node, the two ports are bundled in a port-channel using the linux teaming driver. ISIS runs on the NCS and provides link

redundancy between the two ToRs using EVLAG. The control plane ports carry any or all of the following networks: storage network to access the CEPH cluster, API network hosting the OpenStack API, Management and provision networks used at install time to provision the nodes and during day-2 operations for management tasks.

Figure 39: Data Plane Port Connectivity



The VPP backed data plane ports are configured differently. Each NIC has a port connected to one NCS. This link is a routed link with an IP address both on the NIC port in the Cisco VIM node and an IP address on the NCS port. These addresses come from a /30 subnet. ISIS runs between the NCS and BGP-LU. On the Cisco VIM nodes, ECMP load balancing occurs over the two links to the NCS. On VPP, there is a loopback interface configured that is used as the SR node SID.

Ports C and D are SR-IOV ports that give the virtual machines direct access to the NIC through the VF interfaces. These are out of scope for the SR EVPN implementation on Cisco VIM.

Overview of Cisco VIM Installation for SR EVPN

Before You Begin

- A pair of NCS 5500 acts as the ToR. As Cisco VIM currently supports only a single pair of NCS, you must properly scale your NCS and choose the right NCS-5500 SKU so that all ports of all the nodes can be connected to this single pair of NCS. You must ensure that you include future growth in the port count when selecting the NCS model.
- You cannot enable both SR EVPN and VXLAN EVPN on the same Cisco VIM pod. Only one forwarding mechanism can exist within a single pod.
- Cisco VIM administrators must provide the SR network and subnet information for overlays, and the EVI or BD label. Cisco VIM does not manage the SR EVPN, it only attaches to an existing SR EVPN.
- You must provide the BGP configuration to the Cisco VIM installer during deployment.
- Cisco VIM 3.4.1 does not support traffic engineering policies for dynamic path selection through BGP SR extensions, or any explicit path selection through CLI or any interaction with an SDN controller.
- Only single homing L2 routes are supported. Only EVPN Type 2 and Type 3 are exchanged with the EVPN Route Reflectors (RR) in this release.

- You can enable this feature in Cisco VIM as a day-0 option using either setup_data.yaml file or Unified Management when you install the pod.
- You cannot convert an existing pod to this SR-enabled forwarding mechanism, even if it is set up to use VPP for VLAN or VXLAN based forwarding.

Note Cisco VIM 3.4.1 does not support VMTP and NFVbench with SR EVPN.

You must update the `setup_data.yaml` file as shown below for the SR EVPN deployment:

```
SERVERS:
  control-server-1:
    bgp_speaker_addresses: {sr-mpls-tenant: <ip address from sr-mpls-tenant network but not
 part of the pool>}
    bgp_mgmt_addresses: {sr-mpls-tenant: <ip address from management network but not part
of the pool>}  # <== optional, only when NETWORK_OPTIONS is sr-mpls network, for controller
 node only, needed when BGP peer is over L3
    vtep_ips: sr-mpls-tenant: <ip address from sr-mpls-tenant network and part of the pool>}
  # <== needed for compute and control nodes
    sr_global_block: {prefix_sid_index: <int between 0 and 8000>, base: <int between 16000
 and 1048576>}  # for sr-mpls, unique and must exist for all controllers/computes, sum of
prefix_sid_index and base has to be unique across all servers
    dp_tor_info: {switch_a_hostname: ethx/y, switch_b_hostname: ethx/y} #For sr-mpls
dp_tor_info is mandatory and has no po info

sr_global_block will exist for all computes and controllers. These are the segment identifiers
 used to associate with the Loopback IPs of the VPP nodes
tor_info will exist only for the control plane links.
dp_tor_info will not be port channel. SR being a Layer 3 feature will rely on ECMP links
for data plane redundancy.
vtep_ips is the loopback IP of the VPP which will be distributed by BGP LU for VPP
reachability.
prefix sid is computed from sr_global_block section where base and index is defined.
NETWORK OPTIONS:
A new NETWORK OPTIONS section called sr-mpls has been introduced to support SR MPLS.
#  ######sr-mpls options #####
#  physnet_name: <unique_name> # Optional. Default is "physnet1"
#  enable_ecmp: <true or false>. # Optional (For SR, it is true. default is false), and
only for vpp
#  ecmp_private_pool: <cidr> # Optional, Only if enable_ecmp is true.
#  sr-mpls:     # mutually exclusive to vxlan; will work with VPP and NCS-5500 only for now
#    sr-mpls-tenant:
#       physnet_name: <unique_name>
#       bgp_as_num: <int value between 1 and 4294967295> # unique across all AS
#       bgp_peers: ['ip1'] ---> list of length 1, Peer Route Reflector IPs
#       bgp_router_id: 'ip3' ---> The router ID to use for local GoBGP cluster

ecmp_private_pool is the range used to burn private IPs for the VPP uplinks to the aggregation
 layer.
enable_ecmp is used to enable ecmp links
sr-mpls-tenant is the key word to enable sr provider networking.
bgp_as_num is the bgp as number of the remote PE device
bgp_peer is the bgp speaker loopback address
bgp_router_id will be the router id address used by the gobgp speakers


NETWORKING:
A new networking section called sr-mpls-tenant has been introduced to support SR MPLS.
NETWORKING:
  Networks:
```

```
-
    vlan_id: <vlan_id>
    subnet: <subnet in cidr>
    gateway: <gateway address>
    pool:
      - <a.b.c.d,e.f.g.h>
      - <i.j.k.l to m.n.o.p>
      - <q.r.s.t>

    segments:
      - sr-mpls-tenant

gateway is the BVI ip address to be used for the L3 gateway for CVIM control plane
pool is the address pool of the public addresses
segments will contain sr-mpls-tenant as a provider network
vlan_id is the id used to carve a sub interface for SR control plane from bonded samx CP
links
```

Container Workload Support

Cisco VIM supports VM, baremetal, or container-based workloads. To support the container-based workloads, Cisco VIM hosts Cisco Container Platform as an application. The orchestrator creates a common OpenStack tenant and deploys the Cisco Container Platform control plane on it. The orchestrator can also create a tenant cluster if needed.

The Kubernetes clusters deployed are multi-master clusters with three master nodes and N worker nodes. The Cisco Container Platform control plane consists of three masters and three workers. The master and worker nodes run as VMs on OpenStack.

For more information on enabling Cisco Container Platform over Cisco VIM, see Container Support in Cisco VIM.

P-GPU Support

From Cisco VIM 3.4.1, GPU is supported in a passthrough mode. In this mode, the entire physical GPU card is available to the VM. Currently, Cisco VIM supports only Tesla T4 GPU cards.

For more information on enabling P-GPU, see Enabling P-GPU.

Traffic Monitoring with Open vSwitch

Cisco VIM supports Tap As a Server (TAAS) with Open vSwitch (OVS) as the mechanism driver. Tap-as-a-Service (TaaS) is an extension to the OpenStack network service (Neutron). It provides remote port mirroring for tenant virtual networks. Port mirroring involves sending a copy of all packets of one port to another port.

The TAAS implementation has two parts:

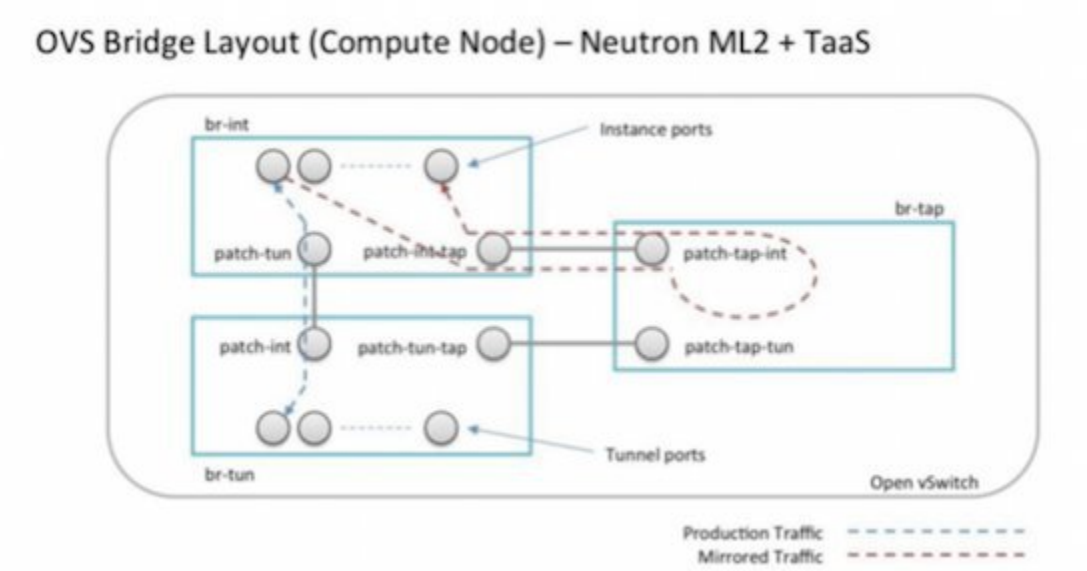
1. Tap-service: Represents the neutron port that receives the mirrored traffic (mirror destination). Tap service creation mirrors the VLAN allocation from an available pool and installs flow rules into the OVS tables.

2. Tap-flow: Represents the neutron port from which the traffic needs to be mirrored (mirror source). It determines the location from where traffic should be copied and the direction to which it should be forwarded.

Tap-flow creation initiates flow rules to copy the mirror source traffic to a VLAN associated with a tap-service. All mirrored traffic is redirected from br-int to br-tap with a specific mirror VLAN number (VLAN pool is preallocated in a configuration). Depending upon the mirror destination location, traffic is redirected back to the br-int (if the destination is on the same compute) or to br-tun (if the destination is on a different compute). If traffic is redirected to br-tun it sends traffic towards the compute node where the mirror destination is located using GRE encapsulation. br-tap is an intermediate point where specific mirrored traffic is filtered towards the mirror destination.

The figure below is a schematic of the layout for the TAAS networking diagram with OVS as the mechanism driver.

Figure 40: TAAS with OVS in a Compute



Following are the assumptions for this feature in CVIM:

1. Filtering of specific traffic is not supported.
2. TaaS is used for debugging and not for mirroring production traffic as it affects OVS performance.
3. If ovs_vswitch container restarts or a compute node reloads, all tap services and tap flows have to be recreated.
4. Tap-flow and Tap-service can be created only for existing neutron ports, that is the corresponding VMs must be in running state. If the neutron ports point to a remote traffic destination, the remote ports will stay in down state.
5. Tap-flow with the same source mirror port cannot be used with different tap-services.

For information on setting up remote and local TaaS OVS, see TAAS Support.