Overview to Cisco NFVI

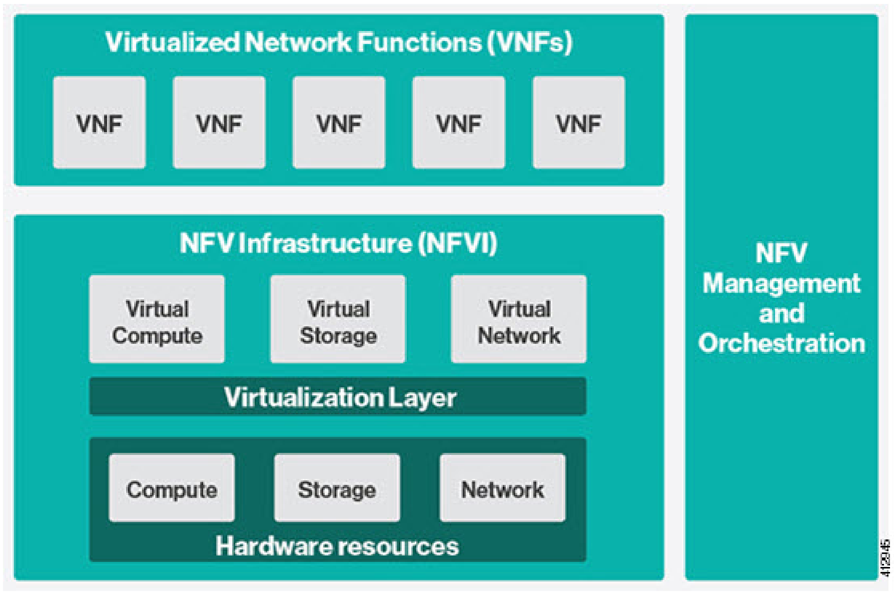
This section contains the following topics:

Overview to Cisco NFV Infrastructure

Cisco Network Function Virtualization Infrastructure (NFVI) provides the virtual layer and hardware environment in which virtual network functions (VNFs) can operate. VNFs provide a 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 in the past, the introduction to VNFs has helped decouple (losely couple) the software from the underlying hardware.

The following figure shows the high-level NFVI architecture.

Figure 1: General NFV Infrastructure



Cisco NFVI features a virtual infrastructure layer (Cisco VIM) that embeds the Red Hat OpenStack Platform (OSP 10).Cisco VIM includes the Newton release of OpenStack, the 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:

- Cisco UCS® C240 M4—Performs management and storage functions and services. Includes dedicated Ceph (UCS 240-M4) distributed object store and file system. (Only Red Hat Ceph is supported).
- Cisco UCS C220/240 M4—Performs control and compute services.
- HP DL360 Gen9 – As third party compute.
- Cisco UCS 220/240 M5 (SFF) – In an 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 micro-pod, and VIC/NIC (pure 40G) based Hyper-Converged and micro-pod offering.

The UCS C240 and C220 servers are M4 Small Form Factor (SFF) models where the computes can boot from a pair of HDDs or SSD. Each UCS C240, C220, and B200 have two 10 GE Cisco UCS Virtual Interface Cards. M5 versions of the UCS C240 and C220 are also of Small Form Factor (SFF) models where the nodes can boot from a pair of HDDs or SSD-based on the BOM type.

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 the Ceph server are connected to redundant fabric interconnects (FIs) managed by Cisco UCS Manager. When you install Cisco VIM on a B-Series pod, you can dynamically allocate VLANs on the provider networks for both Virtio and SRIOV using the optional Cisco UCS Manager plugin. 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 performed 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	• Cisco VIM Installer • Cobbler server • Docker Registry • ELK server
Control	3	• UCS C220/C240 M4 with two 1.2 TB HDDs, or • UCS B200 with two 1.2 TB HDDs • HP DL360 Gen9 • UCS 220/240 M5 with 2x1.2 TB HDDs, or 2x960G SSDs (in a micro pod environment)	• 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 with two 1.2 TB HDDs, or 2x1.6 TB SSDs • UCS B200 with two 1.2 TB HDDs • UCS 220/240 M5 with 2x1.2 TB HDDs, or 2x960G SSDs (in a micro pod environment)	• 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: • UCS C240 M4 with two internal SSDs*, 1 external SSDs, 4 or 5- 1.2 TB HDDs, or • UCS C240 M4, with 2 internal SSDs*, 4 SSDs and 20 1.2 TB HDDs • For UMHC or NGENAHC, UCS C240 M4, with two 1.2TB HDD for OS boot, one/2 SSDs and 5/10 1.2 TB HDDs • SSD-based Ceph: UCS C240 M4 with 2 internal SSDs*, minimum of 4 external SSDs, expandable to 24 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) NCS-5500 as ToRs or Nexus 9K 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.

Note You can use any ToR that supports virtual port channel. We recommend you to use N9K SKUs as TOR, which is released as part of Cisco VIM. When NCS-5500 acts as a TOR, auto-TOR config is mandatory.

Note 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 7.4 with OpenStack Platform 10.0—Provides the core operating system with OpenStack capability. RHEL 7.4 and OPS 10.0 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 Newton 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)—When installing Cisco VIM 2.4, Cisco IMC 2.0(13i) or later is supported.

  For the Cisco IMC 2.0 lineup, the recommended version is as follows:

UCS-M4 servers	Recommended: Cisco IMC 2.0(13n) or later.

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

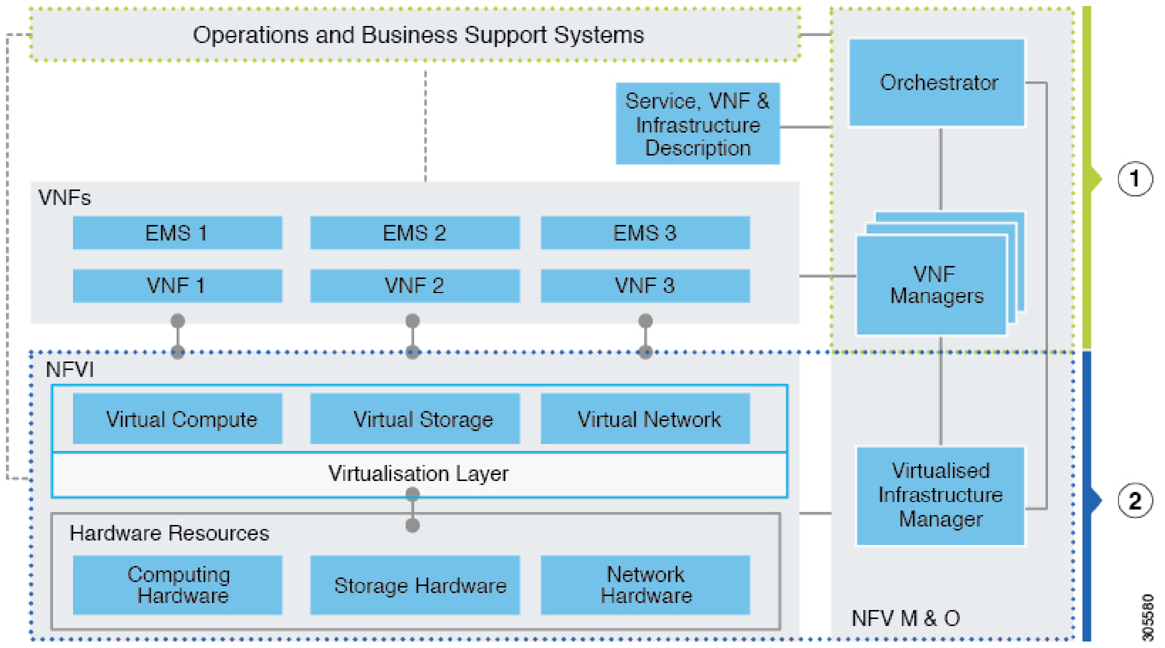
UCS-M4 servers	Cisco IMC versions are 3.0(3a) or later, except for 3.0(4a); Recommended: Cisco IMC 3.0(4d).
UCS-M5 servers	CIMC 3.1(2b) or later. Recommended to stay with 3.1(2b), 3.1(3d), and 3.1(3g).

  Provides embedded server management for Cisco UCS C-Series Rack Servers. Supports Cisco IMC firmware versions for the fresh install of Cisco VIM 2.2 is: 2.0(13i) or greater. Because of recent security fixes, we recommend you to move CIMC to 2.0(13n) or higher. Before upgrade of Pod from Cisco VIM 1.0 to Cisco VIM 2.2, it is expected that you manually upgrade to 2.0(13n) or greater. Similarly, CIMC version of 3.0 lineup is supported; for CIMC 3.0. For this, you must choose a version greater or equal to 3.0 (3a).
- Cisco Virtual Topology System (VTS)—It is a standards-based, 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 the NFV Infrastructure.

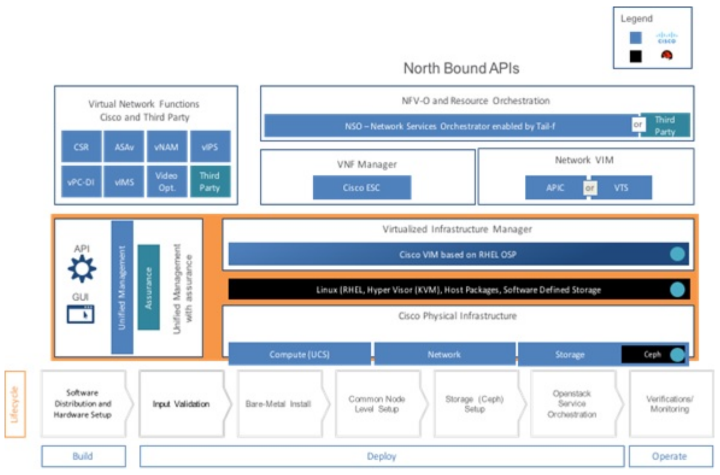
1	• Cisco Network Services Orchestrator • Cisco Elastic Services Controller
2	Cisco NFVI: • Cisco VIM + • Cisco UCS and Cisco Nexus Hardware + • Logging and Monitoring Software + • Cisco Virtual Topology Services (optional) + • Accelerated Switching with VPP (Optional) • Cisco Unified Management (optional)

For cloud networking, Cisco NFVI supports either Linux bridge over Virtual Extensible LAN (VXLAN) or Open vSwitch over VLAN as the cloud network solution for both UCS B- and C-Series pods. However, the UCS B-Series pods using the Cisco UCS Manager plugin supports only OVS/VLAN as a tenant network. Both B-Series and C-Series deployments that support provider networks over VLAN. In addition, in a C-series pod, you can choose to run with augmented performance mechanism by replacing OVS/LB with VPP/VLAN or ACI/VLAN (virtual packet processor). Also, in a C-series pod, you can choose to have the cloud that is integrated with VTC (virtual topology system), 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 compute and Ceph nodes, and replace controller nodes. The Cisco VIM installation embeds all necessary RHEL licenses as long as you use the Cisco VIM 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

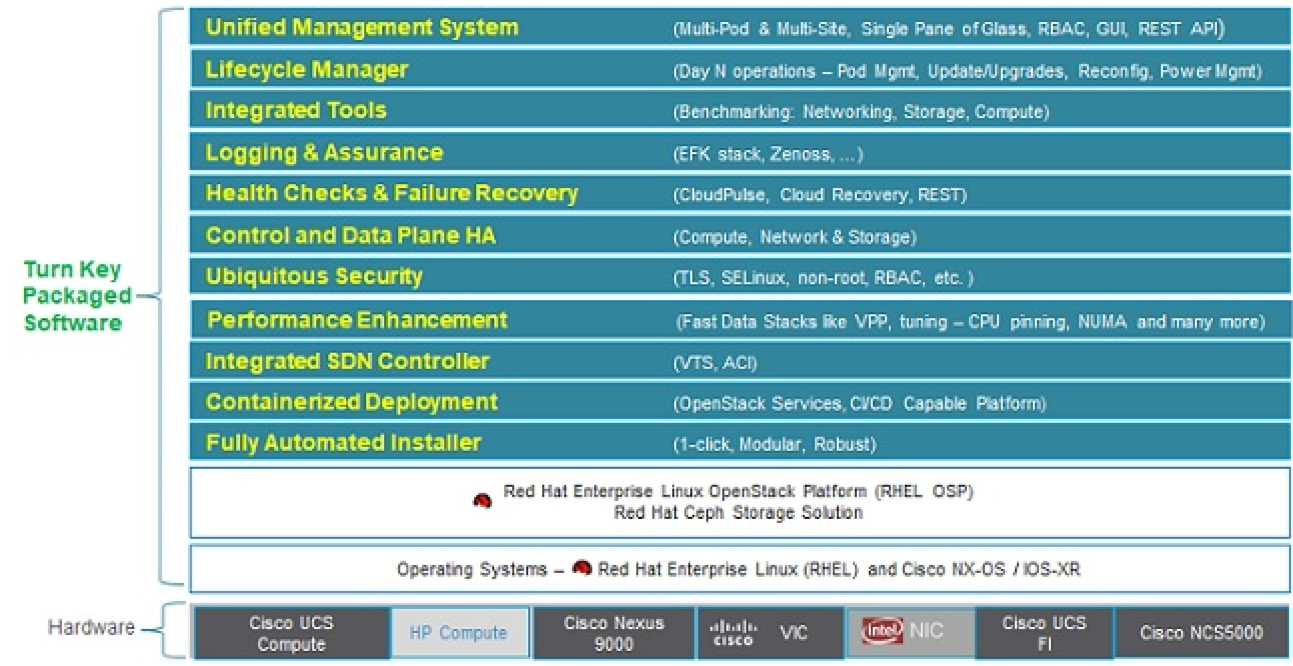

Legend
North Bound APIs
Virtual Network Functions Cisco and Third Party
CSR
ASAv
vNAM
vIPS
vPC-DI
vIMS
Video Opt.
Third Party
NFV-O and Resource Orchestration
NSO – Network Services Orchestrator enabled by Tail-f
or
Third Party
VNF Manager
Cisco ESC
Network VIM
APIC
or
VTS
API
GUI
Unified Management
Assurance
Unified Management with assurance
Virtualized Infrastructure Manager
Cisco VIM based on RHEL OSP
Linux (RHEL, Hyper Visor (KVM), Host Packages, Software Defined Storage
Cisco Physical Infrastructure
Compute (UCS)
Network
Storage
Ceph
Lifecycle
Software Distribution and Hardware Setup
Input Validation
Bare-Metal Install
Common Node Level Setup
Storage (Ceph) Setup
Openstack Service Orchestration
Verifications/ Monitoring
Build
Deploy
Operate

Overview to Cisco Virtual Infrastructure Manager

Cisco Virtual Infrastructure Manager (VIM) is a fully automated cloud lifecycle management system. VIM helps to bring up a fully functional cloud in hours, with integrated end-to-end control and data plane verification in place. Beyond day 0 cloud brings up and deployment, 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


Unified Management System (Multi-Pod & Multi-Site, Single Pane of Glass, RBAC, GUI, REST API)
Lifecycle Manager (Day N operations – Pod Mgmt, Update/Upgrades, Reconfig, Power Mgmt)
Integrated Tools (Benchmarking: Networking, Storage, Compute)
Logging & Assurance (EFK stack, Zenoss, ...)
Health Checks & Failure Recovery (CloudPulse, Cloud Recovery, REST)
Control and Data Plane HA (Compute, Network & Storage)
Ubiquitous Security (TLS, SELinux, non-root, RBAC, etc.)
Performance Enhancement (Fast Data Stacks like VPP, tuning – CPU pinning, NUMA and many more)
Integrated SDN Controller (VTS, ACI)
Containerized Deployment (OpenStack Services, CI/CD Capable Platform)
Fully Automated Installer (1-click, Modular, Robust)
Red Hat Enterprise Linux OpenStack Platform (RHEL OSP)
Red Hat Ceph Storage Solution
Operating Systems – Red Hat Enterprise Linux (RHEL) and Cisco NX-OS / IOS-XR
Turn Key Packaged Software
Hardware
Cisco UCS Compute
HP Compute
Cisco Nexus 9000
VIC
NIC
Cisco UCS FI
Cisco NCS5000

Features of Cisco VIM

Cisco VIM is the only standalone fully automated cloud lifecycle manager offering from Cisco for a private cloud. The current version of VIM, integrates with Cisco C or B-series UCS servers and Cisco 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 7.4 with OSP 10 (Newton).
Hardware Support Matrix	**1.** UCS C220/B200 M4 controller or compute with Intel V3 (Haswell). **2.** UCS C240/220 M4 controller or compute + Intel V4 (Broadwell). **3.** HP DL360 Gen 9. **4.** UCS C220/240 M5 in micropod environment, with an option to add up to 16 220/240-M5 computes. **5.** UCS C240/220 M5 controller or compute with Intel X710 support with SRIOV and Cisco Nexus 9000 /Cisco NCS-5500 series switch as ToR.

NIC support	**1.** Cisco VIC: VIC 1227, 1240, 1340, 1380. **2.** Intel NIC: X710, 520, XL710.
POD Type	**1.** Dedicated control, compute, and storage (C-series) node running on Cisco VIC, or Intel 710 X (full on) with Cisco Nexus 9000 or Cisco NCS 5500 series switch (only for Intel NIC and VPP as mechanism driver) as ToR. **2.** Dedicated control, compute, and storage (B-Series) node running on Cisco NIC. **3.** MICRO POD: Integrated (AIO) control, compute, and storage (C-series) node running on Cisco VIC, or Intel 710 X or VIC/NIC combo. Micro pod can be optionally expanded to accommodate for more computes (max of 16) running with the same NIC type. This can be done as a day-0 or day-1 activity. The computes can boot off HDD or SSD ; Intel NIC-based micropod supports SRIOV, with the M5 based micropod supporting only XL710 as an option for SRIOV. **4.** Hyper-Converged on M4(UMHC): Dedicated control and compute nodes, with all storage acting as compute (M4 C-series) nodes, 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. **Note** In a full-on (VIC based), or Hyper-Coverged 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 will not be running SR-IOV. In 2.4, we support HP DL360 Gen9 as a third party compute. Currently, we do not support a mix of computes from different vendors for the same pod. **5.** 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. **6.** 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.

ToR and FI support	1. For VTS-based installation, use the following Nexus version-7.0(3)I2(2a) and 7.0(3)I6(2). 2. For the mechanism driver other than VTS, use the following Nexus software version 7.0(3)I4(6) 7.0(3)I6(1). **Note** If you are using Auto-ToR configuration that is TORSWITCH CONFIGURE_TORS: True then irrespective of the MECHANS nxos version - 7.0(3)I6(1) automation fails due to a defect - https://bst.cloudapps.cisco.com/bugsearch/bug/CSCve16902/?r 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. 5. Cisco Nexus 9000 series switches running ACI 3.0 (for the mechanism driver ACI).
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.
Mechanism Drivers	OVS/VLAN, Linuxbridge/VXLAN, ACI/VLAN, VPP/VLAN (Fast Networking, Fast Data FD.io > VPP/VLAN, based on the FD.io VPP fast virtual switch). **Note** VPP with LACP is now the default configuration for the data plane.
SDN Controller Integration	VTS; ACI (with Unified ACI Plugin) with Cisco VIC or Intel NIC on the UCS C-series M4 platform.
Install Methodology	• Fully automated online or offline installation. • Support of Software Distribution Server (SDS), to mitigate the problem associated with logistics of USB distribution for air-gapped installation.
Scale	1. Full on: Total of 60 (compute and OSD) nodes (with Ceph OSD max at 20). 2. Micropod: max of 16 standalone compute nodes. **Note** For, Ceph OSDs can be HDD or SSD based; it has to be uniform across the pod. Computes can boot off 2x1.2TB HDD or 2x1.6TB 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. 2. Reconfiguration of passwords and selected optional services. 3. 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 after post install for maintenance and security. Non-root log in for Administrators. Enabling Custom Policy for VNF Manager. Optionally, you can disable the reachability of the management node to the cloud API network.
EPA	NUMA, CPU pinning, huge pages, SRIOV with Intel NIC.
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	ELK integrated with external syslog (over v4 or v6) for a log offload, with optional support of NFS with ELK 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
Storage	• Object store with SwiftStack, Block storage with Ceph, or NetApp. • Option to use Ceph for Glance and SolidFire for Cinder.
Monitoring	• Third-party integration with Zenoss (called NFVIMON). • Automated ToR configuration of collector ToR ports, when Cisco NCS 5500 is used as ToR. • CVIMMON for monitoring, a Cisco solution as a technical preview.

Support of External Auth System	1. LDAP 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 2.2.24 to release 2.4.3.
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.
VXLAN extension into the cloud	Extending native external VXLAN network intoVNFs in the cloud.
Power Management of Computes	Option to power off or on computes selectively to conserve energy.
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 disk(s) on the Pod node(s) without the need for add/remove/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), NFVBench (NFVI data plane and a service chain performance benchmarking tool) 2. Services Health Checks Integration: Cloudpulse and Cloudsanity.

Note Configure the LACP on the data plane ports of the Cisoc 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 NFVI Networking Overview

Cisco VIM supports installation on two different type of pods. The B-series and C-series 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) (not a full subnet) 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.
- The Cisco Virtual Topology Services (VTS) (if included in your Cisco NFVI package) Virtual Topology Controller (VTC) node (optional for VTS).
- VTC (optional for VTS).

External Segment

The external segment needs one VLAN to configure the OpenStack external network. Provide the VLAN during installation in the Cisco NFVI setup_data.yaml file, but configure the actual subnet using the OpenStack API after the installation. Then 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, 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/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
ACIINFRA	Internal ACI Network for Policy management (only allowed when deployed with ACI)	aciinfra	o
Installer API	• Administrator uses installer API network to ssh to the management node. • Administrator connects to installer API to interface with secured services. 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	+	+	+	+
ACIINFRA*		+	+	
API		+		
Tenant		+	+	
Storage		+	+	+
Provider			+	
External		+		

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

The network arrangement on HP third- party 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 compute of the two Vendors.

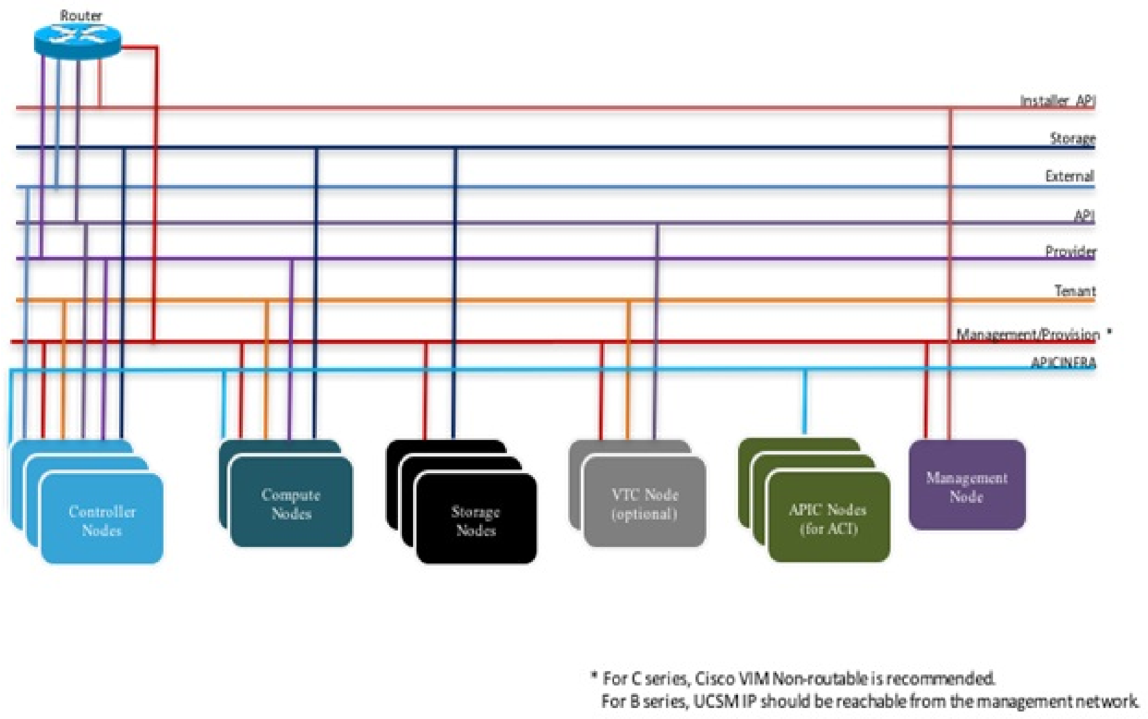
Network Interface	**Cisco UCS c220/c240M4 Compute**	**HPE ProLiant DL360 Gen9 Compute**
mx	Management control plane network	N/A

Network Interface	Cisco UCS c220/c240M4 Compute	HPE ProLiant DL360 Gen9 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:

- Virtual extensible LAN (VXLAN) over a Linux bridge.
- VLAN over Open vSwitch(SRIOV with Intel 710NIC).
- VLAN over VPP/VLAN for C-series Only.
- For UCS B-Series pods, Single Root Input/Output Virtualization (SRIOV). 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.

Any connection protocol can be used unless you install UCS B200 blades with the UCS Manager plugin, in which case, only OVS over VLAN can be used. The following table shows the available Cisco NFVI data path deployment combinations.

Table 4: Cisco NFVI Data Path Deployment Combinations

NFVI Pod Type	Pod Type	Mechanism Driver	Tenant Virtual Network Encapsulation		Provider Virtual Network Encapsulation	SRIOV for VM	PCI Passthrough Ports	MTU Values	
			VLAN	VxLAN	VLAN			1500	9000
UCS C-series	Full on	LinuxBridge	No	Yes	Yes	No	No	Yes	No

NFVI Pod Type	Pod Type	Mechanism Driver	Tenant Virtual Network Encapsulation		Provider Virtual Network Encapsulation	SRIOV for VM	PCI Passthrough Ports	MTU Values	
UCS C-series	Full on , micro (M4 or M5 based), , HC	Openvswitch	Yes	No	Yes	Yes*	No	Yes	Yes
UCS C-series	Full on, micro (M4 or M5 based),	VPP	Yes	No	Yes	Yes*	No	Yes	Yes
UCS C-series	Full on, micro (M4 based),	ACI	Yes	No	Yes	Yes*	No	Yes	Yes
UCS C-series	Full on	VTF with VTC***	No	Yes	Yes	No	No (except through DPDK)	Yes	Yes
UCS B	Full on	Openvswitch	Yes	No	Yes	Yes	No	Yes	Yes

Note Fullon: Indicates dedicated control, compute and ceph nodes.

Micro: Indicates converged control, compute and ceph nodes with expandable computes.

HC (Hyperconverged): Indicates dedicated control, compute, but all ceph nodes are compute nodes also.

Note * The SRIOV support applies to only with 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. First of all, the requirement is to have atleast two NICs (4x10G) single server, so that we can 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, the idea is to 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 into the following.

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/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			+	

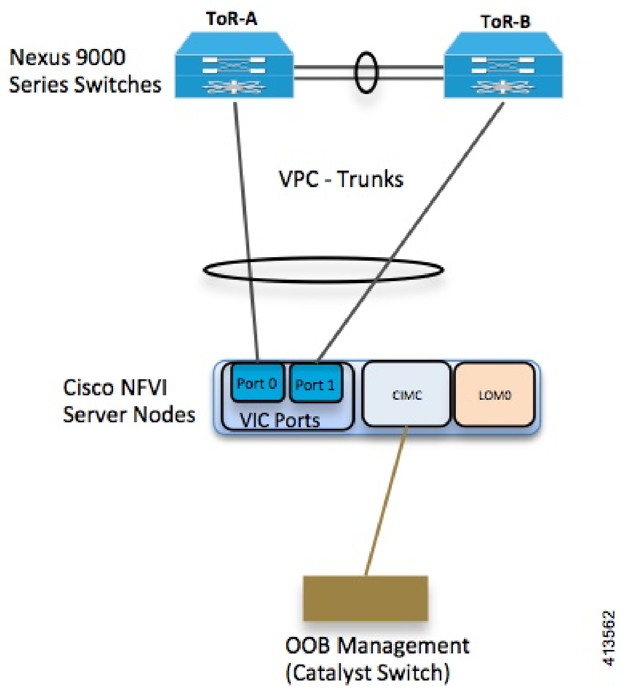
Note On an Intel pod, all kind of OpenStack networks are created using **physnet1** as the physnet name.

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.

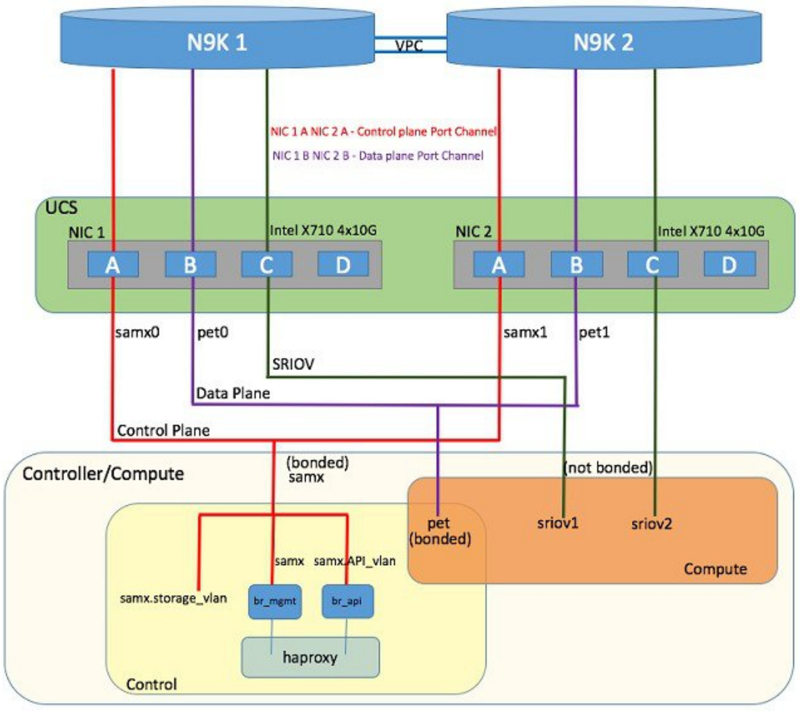
The following figure shows the UCS C-Series pod Cisco NFVI host to ToR topology.

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. The addition of an extra Intel NIC has been done to incorporate the user request of providing card level redundancy which the Cisco VIC solution does not have.

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 port A and B from each NIC help in forming bonded interface, 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.

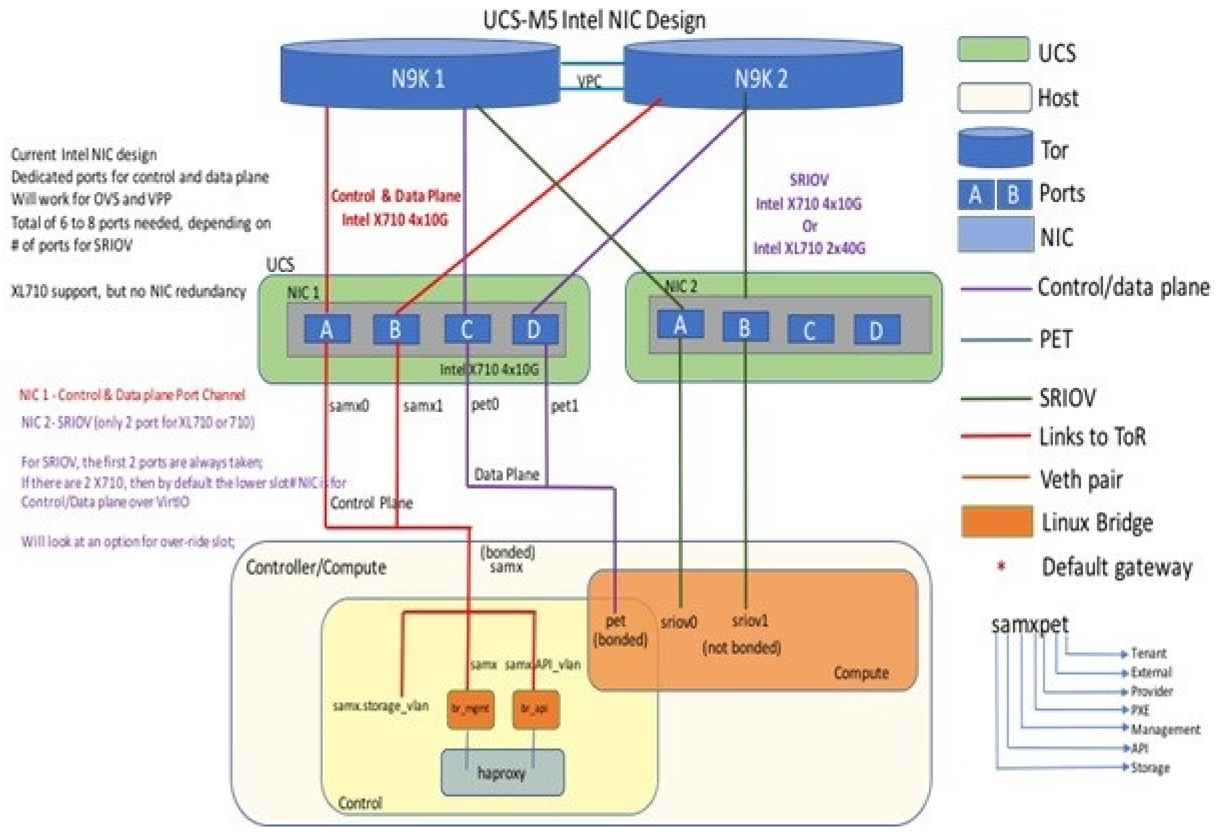
From Cisco VIM Release 2.4.2 onwards, support of M5 full on pods with two port XL-710 across control, compute and dedicated Ceph Nodes, and with NIC_LEVEL_REDUNDANCY is available. This deployment can be achieved with Cisco Nexus 9000 series or NCS-5500 as ToR. SRIOV is not supported in computes with 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. While the SRIOV card is optional, once the SRIOV card is chosen, all the computes must have same number of SRIOV ports across the pod. As NIC redundancy is an over-kill, in the M5 micropod environment, the samx and pet ports hang off 1 4 port X710 Intel NIC card.

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

Figure 8: Networking Details of UCS-M5 Micropod Deployment

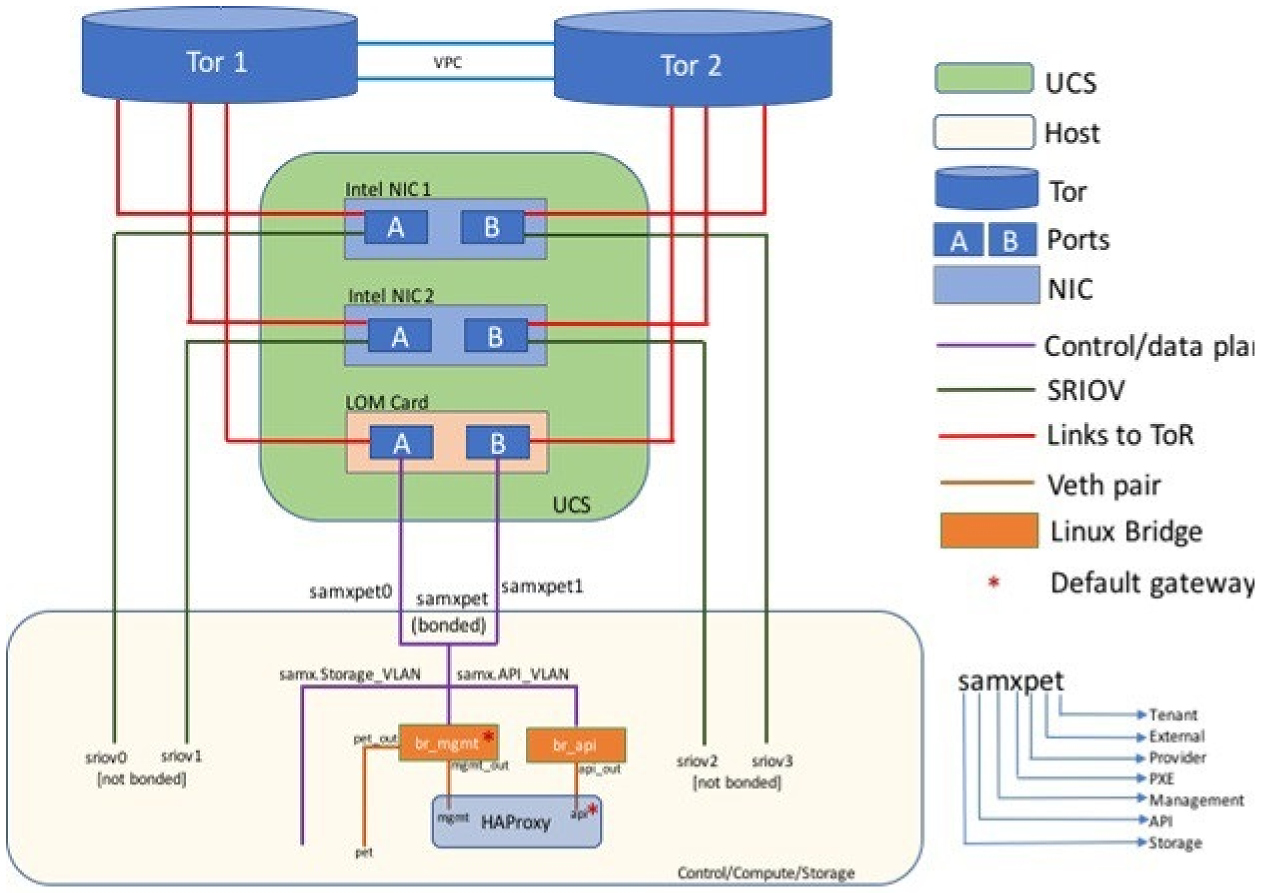

UCS-M5 Intel NIC Design
N9K 1
VPC
N9K 2
Current Intel NIC design
Dedicated ports for control and data plane
Will work for OVS and VPP
Total of 6 to 8 ports needed, depending on # of ports for SRIOV
XL710 support, but no NIC redundancy
Control & Data Plane Intel X710 4x10G
SRIOV Intel X710 4x10G Or Intel XL710 2x40G
UCS
NIC 1
NIC 2
A B C D
Intel X710 4x10G
NIC 1 - Control & Data plane Port Channel
NIC 2 - SRIOV (only 2 port for XL710 or 710)
For SRIOV, the first 2 ports are always taken; If there are 2 X710, then by default the lower slot# NIC is for Control/Data plane over VirtIO
Will look at an option for over-ride slot;
samx0
samx1
pet0
pet1
Data Plane
Control Plane
(bonded) samx
Controller/Compute
pet (bonded)
sriov0
sriov1
(not bonded)
Compute
samx
samx API_vlan
samx.storage_vlan
br_mgmt
br_api
haproxy
Control
UCS
Host
Tor
Ports
NIC
Control/data plane
PET
SRIOV
Links to ToR
Veth pair
Linux Bridge
Default gateway
samxpet
Tenant
External
Provider
PXE
Management
API
Storage

Cisco VIM 2.4 introduces the first third-party compute. The first SKU chosen is HPE ProLiant DL360 Gen9. In CVIM 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+ Adapter are used for SRIOV for the provider network. Listed below is network schematic of the HP Compute node

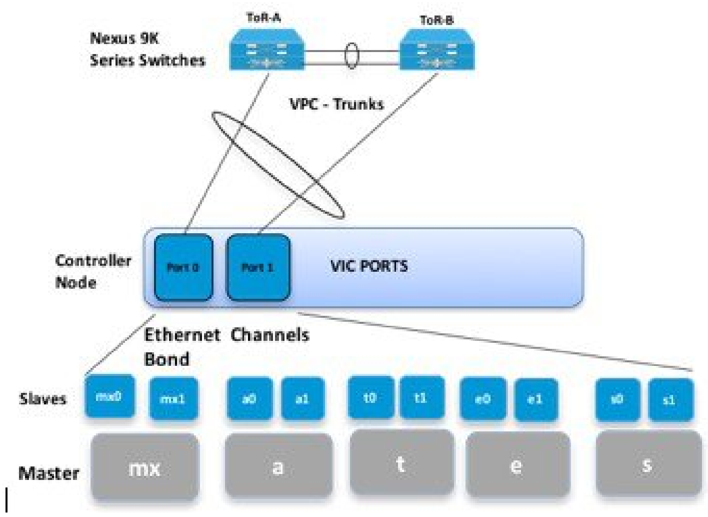
Figure 9: Networking details of HP DL360GEN9



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.

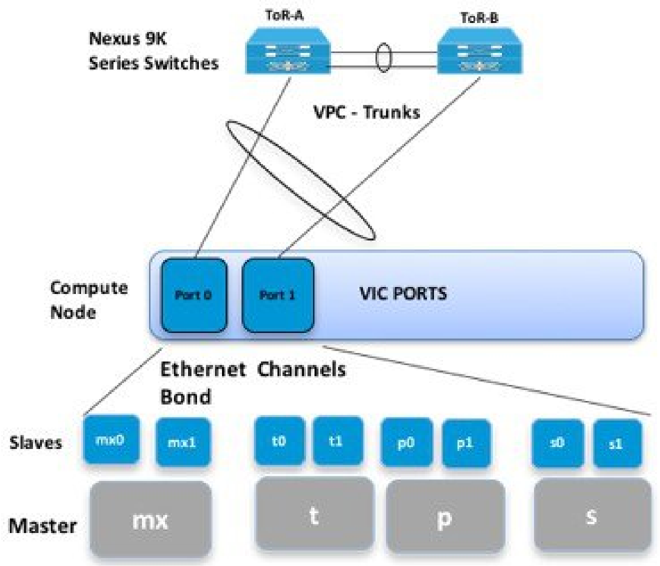
The following figure shows the controller node network-to-bond-to-vNIC interface mapping.

Figure 10: Controller Node Network to Bond Mapping



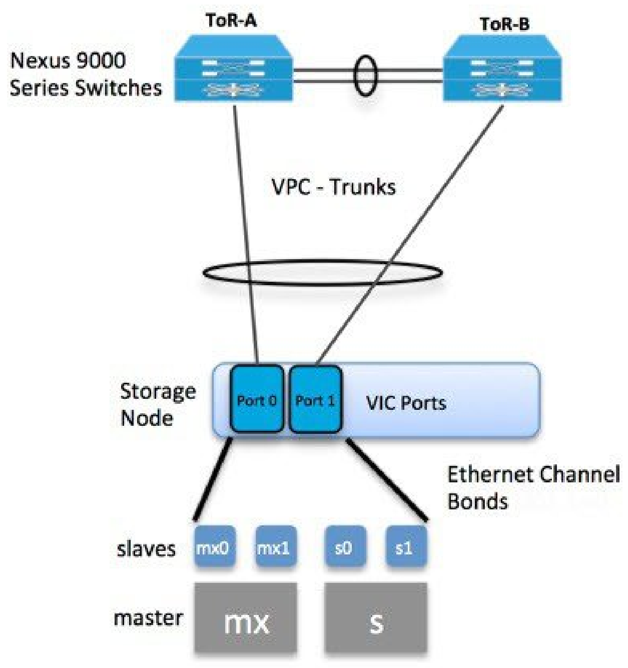
The Cisco NFVI compute node has three bonds: mx, t, and p. Each has a slave interface that is named with the network name association and a mapped number. For example, the provider network, p, maps to p0 and p1. The bonds map directly to the vNICs that are automatically created on the compute node when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

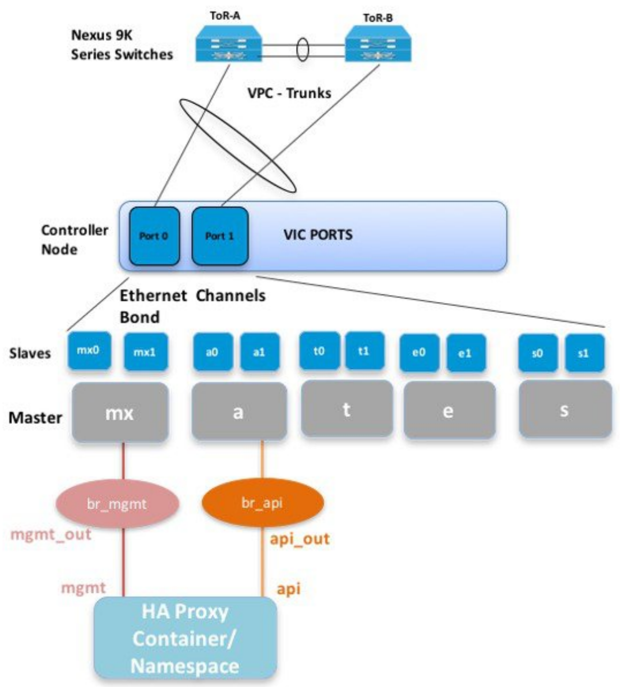
Figure 12: Storage Node Networking to Bond Mapping



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
```

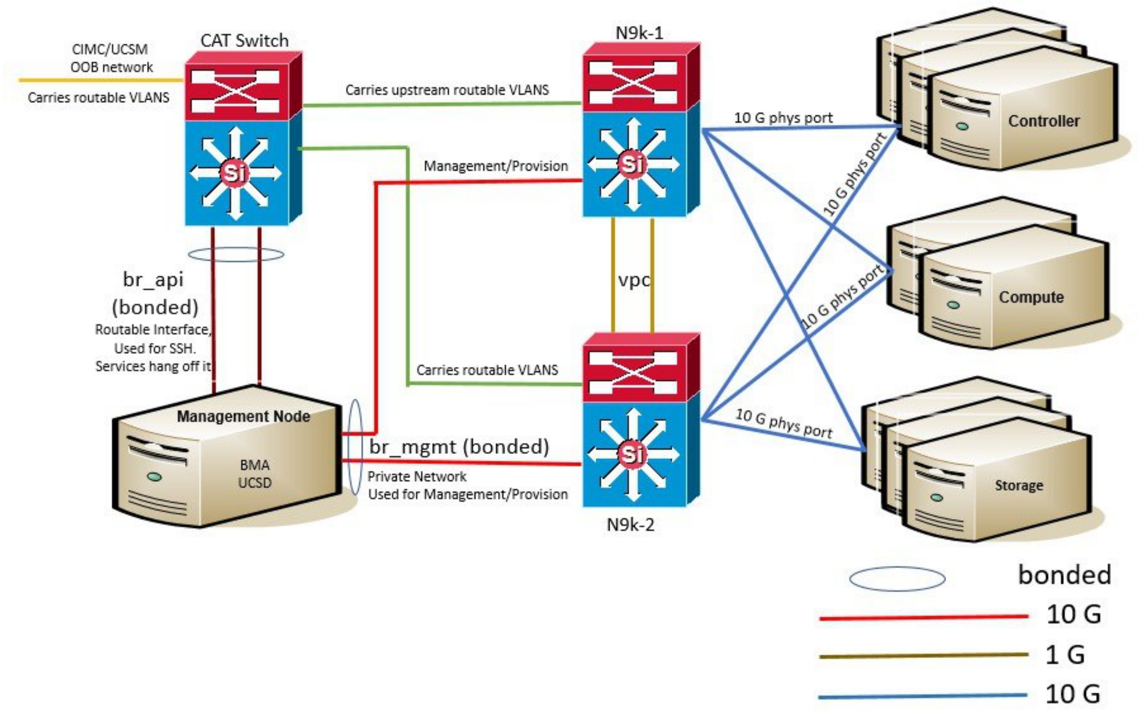
Cisco VIM Management Node Networking

In Cisco VIM, the management node has two interfaces. One for API and the other 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

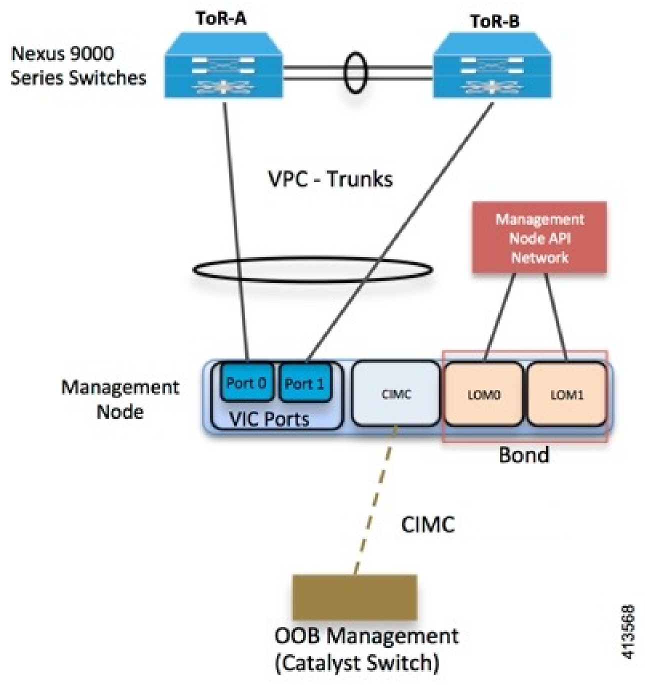


Cisco NFVI UCS C-Series management node physically connects to the network. Unlike other nodes, the management node does not use multiple vNICs corresponding to specific Cisco NFVI networks. Instead, it connects to the management and API networks using two different physical connections. The management node connects to the management network using a Cisco two-port VIC 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


ToR-A
ToR-B
Nexus 9000 Series Switches
VPC - Trunks
Management Node API Network
Management Node
Port 0
Port 1
VIC Ports
CIMC
LOM0
LOM1
Bond
CIMC
OOB Management (Catalyst Switch)
413568

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 CVIM 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

Users are transiting from IPv4 to IPv6 due to the limited number of available routable IPv4 networks. Today in Cisco VIM, the management network is in IPv4. 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.

With limited availability of IPv4 address-space, it can become a deployment hindrance for users who cannot provide a routable IPv4 network or local or dual-home of their external services (AD/LDAP is an example where it is hosted in the corporate network and require routing to get it).

IPv 4 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, it is difficult to completely remove IPv4 from the management network. So we recommend you to run IPv4+IPv6 dual stack, in which IPv4 network can exist in a non-routable private network and IPv6 network can be in a routable semi private network. This satisfies both requirements of the CiscoVIM and the user's accessibility to their external services.

In CiscoVIM, IPv6 addresses of the management network for servers and management node is statically allocated from a given pool. The external services, which support both IPv4 and IPv6 addresses, are DNS, NTP, AD/LDAP. Users, can run IPv4+IPv6 (optionally) as their cloud api end point.

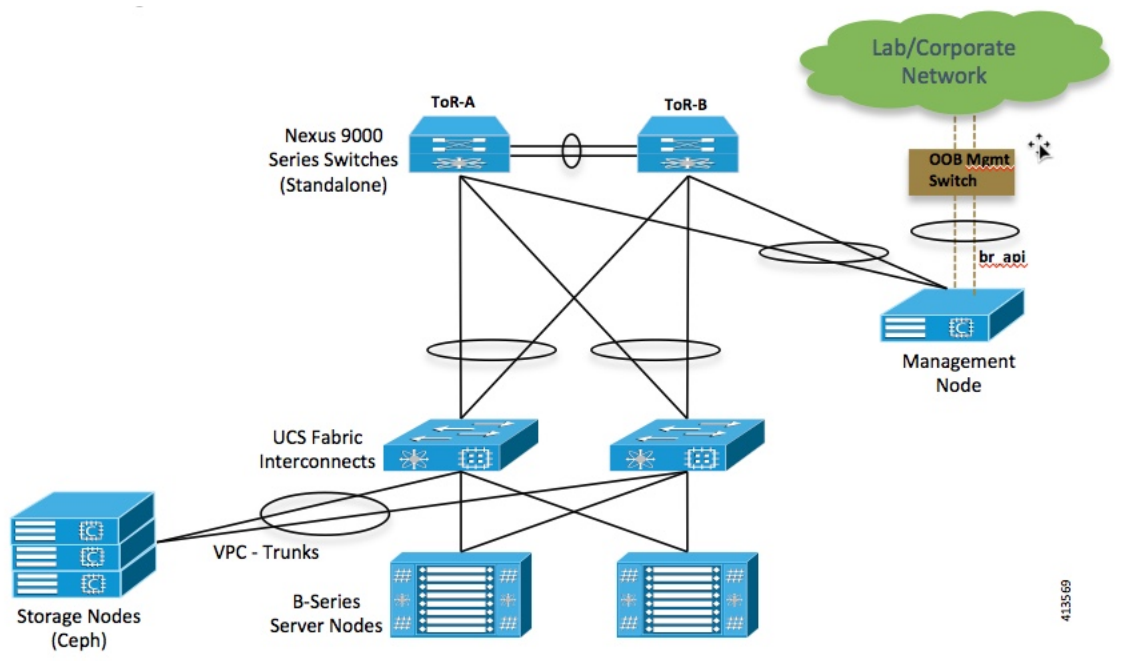
UCS C-Series and B-Series -Topologies

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 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. Networking segment layout is discussed in, *Cisco NFVI Networking Overview* is the same for a C-Series-only implementation or the C-Series and B-Series design that is shown in the following two exceptions:

- For the 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.

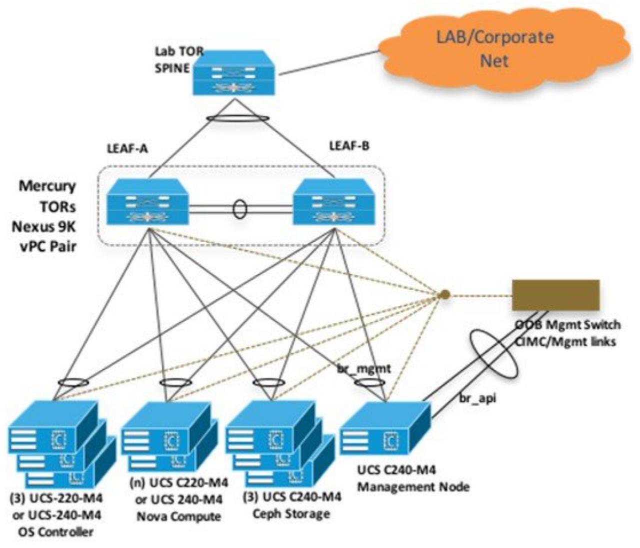
Following figure shows a high-level view of Cisco UCS C-Series and B-Series servers that are used in a Cisco NFVI deployment

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 Nexus 9000 switch, which acts as the ToR. The Nexus 9000s are configured in VPC mode to ensure that the network redundancy is built into the design from the beginning. The networking redundancy is extended to the management node, which has a redundant vNIC for the installer API and management or provisioning networks. The figure shows the C-Series topology.

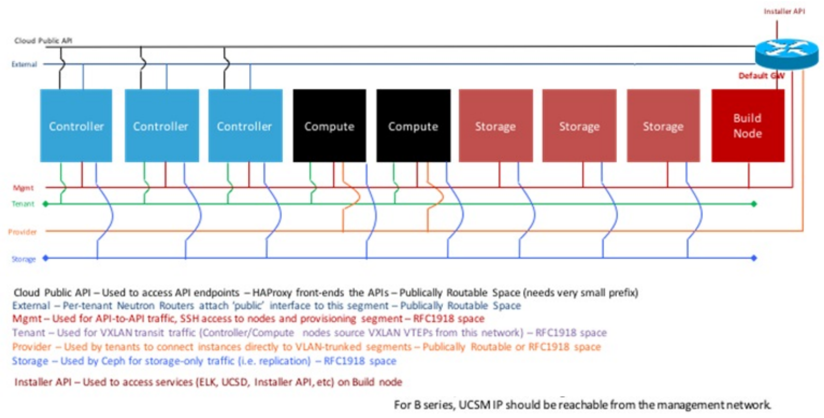
Figure 17: Cisco NFVI C-Series Topology


Lab TOR SPINE
LAB/Corporate Net
LEAF-A
LEAF-B
Mercury TORs Nexus 9K vPC Pair
OOB Mgmt Switch CIMC/Mgmt links
br_mgmt
br_api
(3) UCS-220-M4 or UCS-240-M4 OS Controller
(n) UCS C220-M4 or UCS 240-M4 Nova Compute
(3) UCS C240-M4 Ceph Storage
UCS C240-M4 Management Node

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 the UCS C-Series management and storage nodes, VLANs are trunked between the ToR switches and the Cisco VICs on the C-Series nodes. For the 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



Cloud Public API – Used to access API endpoints – HAProxy front-ends the APIs – Publically Routable Space (needs very small prefix)
External – Per-tenant Neutron Routers attach 'public' interface to this segment – Publically Routable Space
Mgmt – Used for API-to-API traffic, SSH access to nodes and provisioning segment – RFC1918 space
Tenant – Used for VXLAN transit traffic (Controller/Compute nodes source VXLAN VTEPs from this network) – RFC1918 space
Provider – Used by tenants to connect instances directly to VLAN-trunked segments – Publically Routable or RFC1918 space
Storage – Used by Ceph for storage-only traffic (i.e. replication) – RFC1918 space
Installer API – Used to access services (ELK, UCSD, Installer API, etc) on Build node

For B series, UCSM IP should be reachable from the management network.

Cisco NFVI High Availability

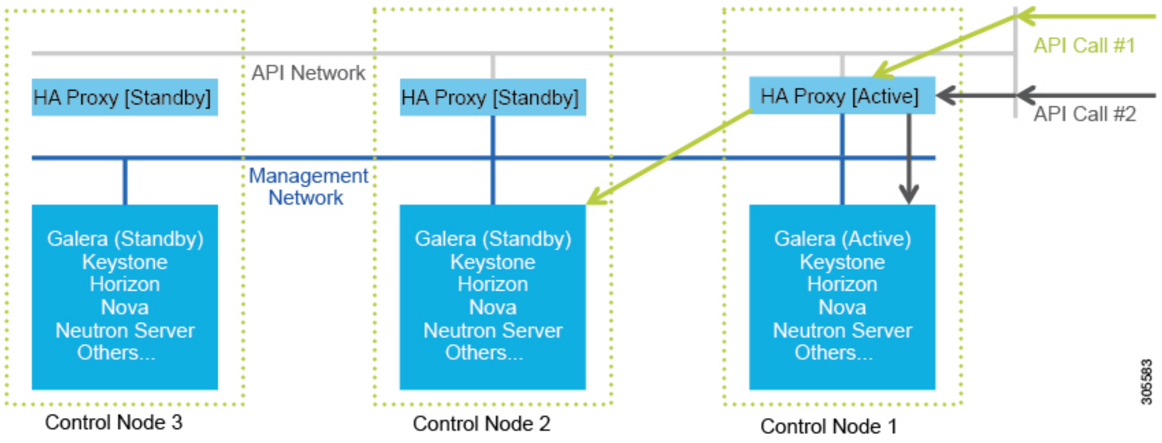
Cisco NFVI high availability (HA) is provided by HAProxy, a single-threaded, event-driven, non-blocking engine combining a fast I/O layer with a priority-based scheduler. HAProxy architecture is layered with bypass mechanisms at each level to ensure that the data does not reach higher levels than needed.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


API Network
HA Proxy [Standby]
HA Proxy [Standby]
HA Proxy [Active]
API Call #1
API Call #2
Management Network
Galera (Standby) Keystone Horizon Nova Neutron Server Others...
Galera (Standby) Keystone Horizon Nova Neutron Server Others...
Galera (Active) Keystone Horizon Nova Neutron Server Others...
Control Node 3
Control Node 2
Control Node 1
305583

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:

```
# cd installer-<tagid>/tools
# ./cloud_sanity.py -c all
```

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.

Cisco VIM 2.4 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)

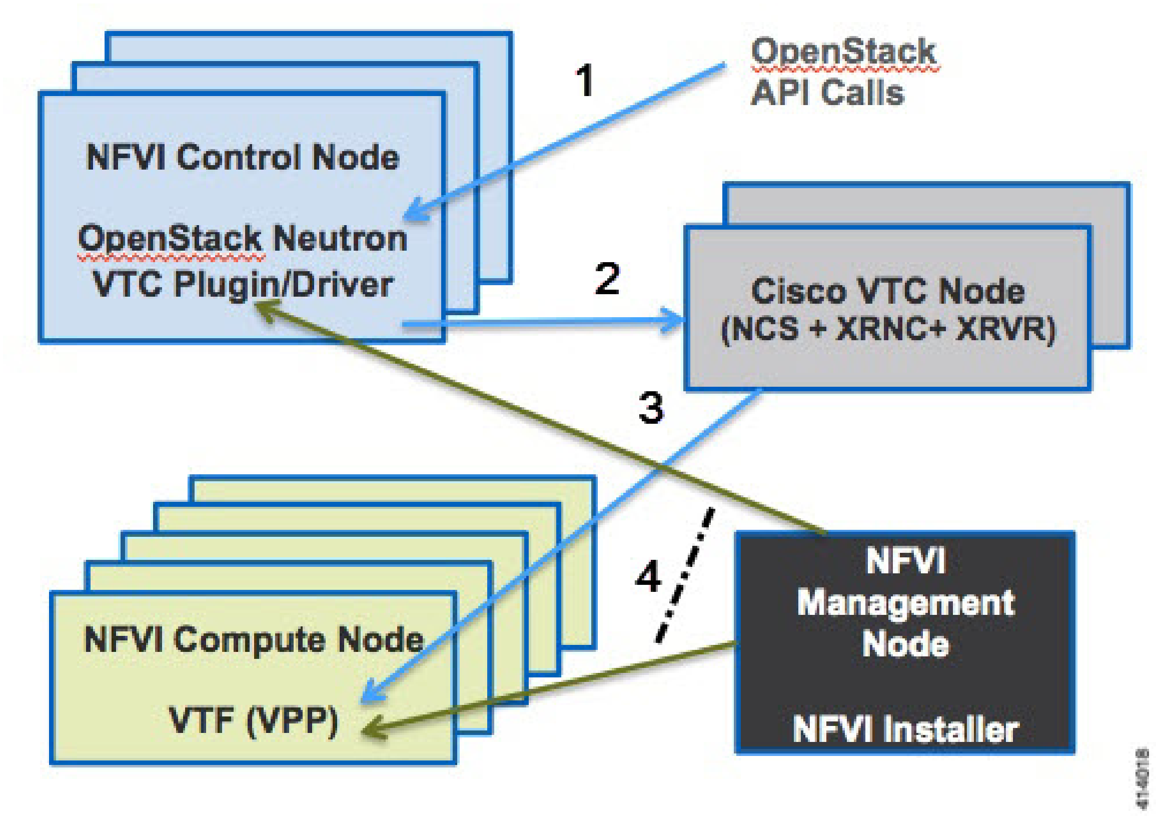
The Cisco NFVI implementation of Cisco VTS includes the VTS Virtual Topology Forwarder (VTF). VTF provides a Layer 2/Layer 3 (L2/L3) software switch that can act as a software VXLAN terminal endpoint (VTEP). VTF is a lightweight, multitenant software data plane designed for high performance packet processing 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.

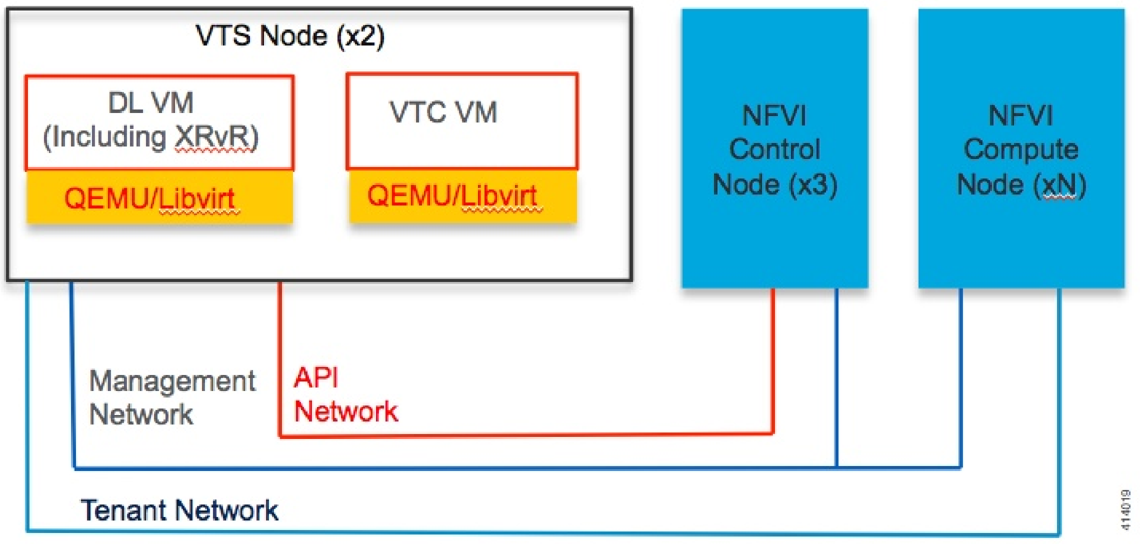
1. The OpenStack user invokes the OpenStack Neutron API.
2. Neutron uses the VTS plugin and driver to make calls to the VTC REST API.
3. VTS control components interact with the VTF agent to carry out the corresponding dataplane setup.
4. During Cisco NFVI installation, the Cisco NFVI Installer installs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

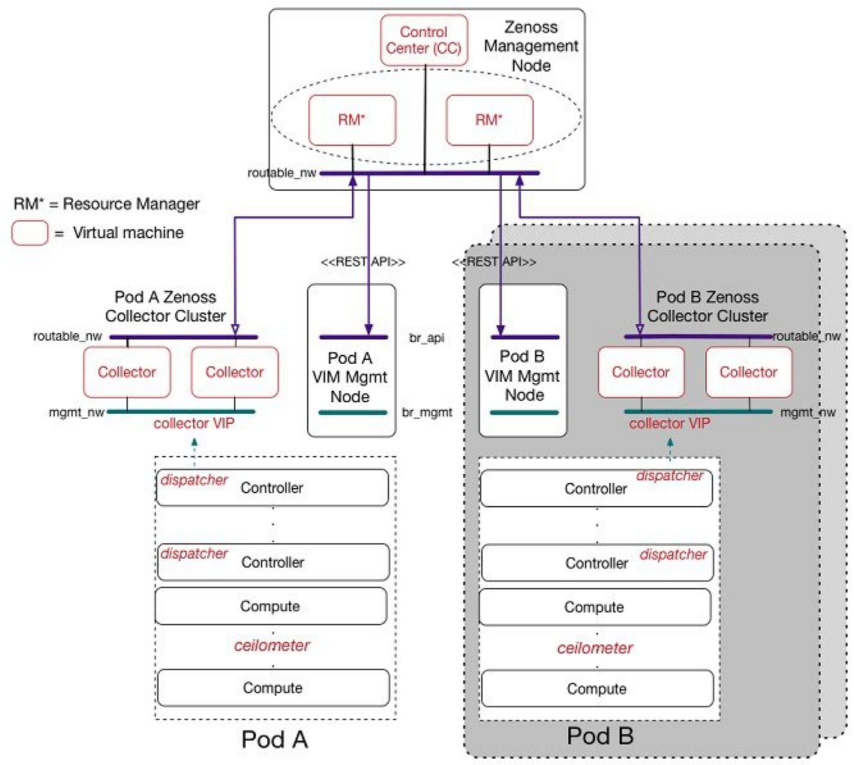
Figure 21: Cisco VTS Networking Inside Cisco NFVI



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 **2600 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: dispatcher, 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 minimal software piece (dispatcher) required to monitor the pod. The installing of the other NFVIMON components (collector, resource manager (RM), and control-center (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(CC) 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 when NFVIMON is enabled, or by adding NFVIMON post installation. Install the collectors manually in the external collector node,so that the pod is added for monitoring in the control center.

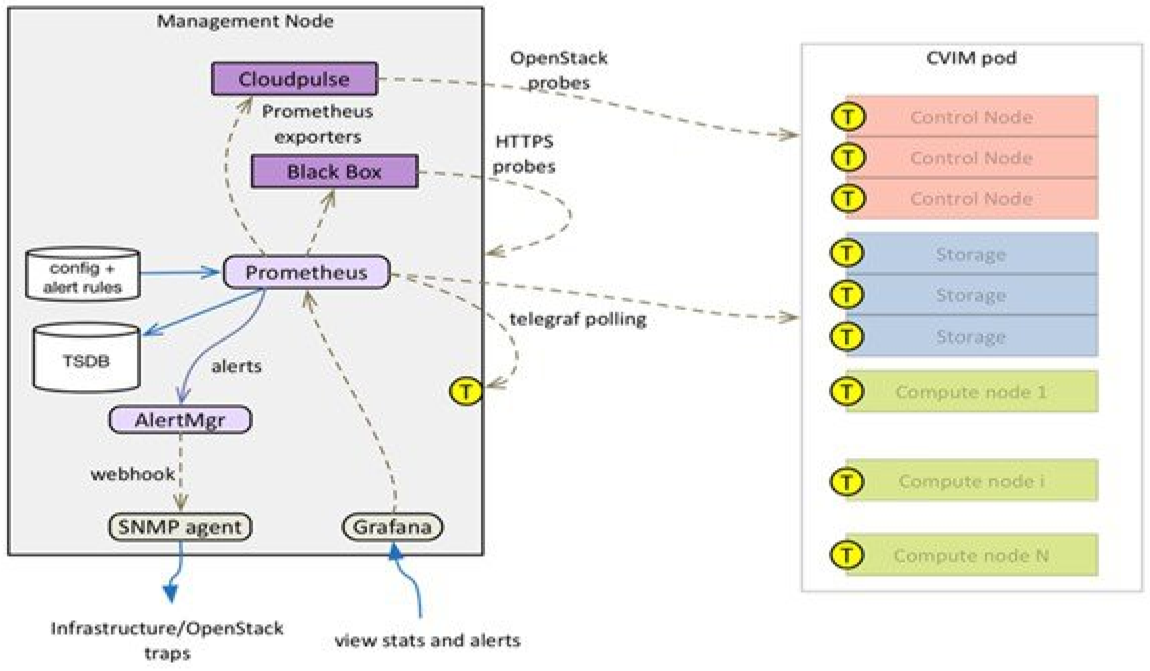
Note NFVIMON is only supported on a pod running Keystone v2, and requires root privileges as non-root admin keys cannot be used for monitoring purposes.

Overview to CVIMMON

Cisco VIM can be deployed through a lightweight pod-level monitoring solution known as CVIMMON which is based on the open source PTG stack (Prometheus, Telegraf, Grafana). This solution is available as an add-on in technical preview mode for 2.4 lineup with plans of getting productized in release 3.0. This feature is available as an add-on, from both commercial and feature point of view. This solution 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 CVIMMON 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 CVIMMON are:

CVIM_MON—Provides the base functionality of monitoring and KPIs.

CVIM_TRAP—It is enabled through SNMP and available only if CVIM_MON is enabled. Optionally, you can enable SNMP at the server/infrastructure level.

Comparative Analysis

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

Table 6: Comparison of CVIMMON and NFVIMON

Features	CVIMMON	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

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. CVIMMON 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

Node Type	Source of Metric
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).

CVIMMON 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

CVIMMON 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 1 hour.

The following table describes the Telegraf plugins installed as part of the CVIMMON deployment:

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

Plug-in	Metric Name	Notes
cpu	cpu_usage_*	Detailed stats for every CPU (with role label)
disk	disk_*	Detailed stats for every disk
diskio	diskio_*	Disk activity
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_*	Stats for interfaces used by CVIM
kernel	kernel_boot_time kernel_context_switches kernel_interrupts kernel_*	
processes	process_* processes_*	
swap	swap_*	
system	system_*	
docker	docker_container	
ntp	ntpq	

Plug-in	Metric Name	Notes
ceph	ceph_osdmap_* ceph_pgmap_* ceph_pool_usage_* ceph_usage_total_*	Medium frequency collection group
rabbitmq	rabbitmq_overview_* rabbitmq_node_* rabbitmq_queue_* rabbitmq_exchange_*	Low frequency collection group
conntrack	ip_conntrack_count ip_conntrack_max	
exec	directory_plugin_bytes	Monitor EFK and Prometheus own storage usage
haproxy	haproxy_*	

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	10s	10s	60m
Medium	30s	30s	60s
Low	1m	1m	60m

OpenStack and infrastructure service metrics

Each Cisco VIM pod provides the Cloudpulse service to monitor the essential OpenStack services. CVIMMON integrates Cloudpulse results by mapping the state of OpenStack services into actionable time series data. Cloudpulse provides the following metrics to Prometheus:

Metric	Metric Name	Notes
openstack service	checkcp_openstack_service_up	Checks the state of an openstack service. Monitors nova, glance, cinder, keystone, and neutron.
rabbitmq status	cp_rabbitmq_server_up	Describes the state of each rabbitmq server.

Metric	Metric Name	Notes
galera status	cp_galera_server_up	Describes the state of each galera server.
ceph check	cp_ceph_health	Checks if ceph is healthy
docker check	cp_container_up	Describes the state of each container. The host label helps to differentiate the containers that have same name on different nodes, for example, neutron-vpp, nova-compute, ceph-mon.
hypervisor checks	cp_hypervisor_up	Check the state of each hypervisor.
Service Down	cp_service_down	Indicated only when exception has occurred during metric creation.

Note The collection interval for all cloudpulse metrics is set to 4 minutes and are not configurable for the release CVIM 2.4.3.

Etcd monitoring

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

Alerting Rules

CVIMMON 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 have a grace period and an alert can 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 and can optionally be converted into SNMP traps. You can configure CVIMMON 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:

Fault Source	Fault Code	Severity	Description
instance_down	hardwareFailure	major	Instance down
disk_used_percent	resourceThreshold	alert	

Fault Source	Fault Code	Severity	Description
disk_filling_up_in_4h	resourceUsage	major	Disk may fill up within 4 hours.
mem_available_percent	resourceThreshold	alert	|mem_available_percent
memory_running_out_in_4h	resourceUsage	major	|mem_available_percent
swap_used	resourceThreshold	alert	
conntrack_percent	resourceThreshold	alert	conntrack_ip_conntrack_count is more than 80% of max
reboot	hardwareFailure	alert	system_uptime
system_n_users	resourceThreshold	alert	Number of logged in users
docker_n_containers_running	serviceFailure	major	Not running any Docker containers.
docker_container_memcached	serviceFailure	major	Memcached containers missing or down
docker_container_heatapicfn	serviceFailure	major	heatapicfn containers missing or down
docker_container_heatapi	serviceFailure	major	heatapi containers missing or down
docker_container_heatengine	serviceFailure	major	heatengine containers missing or down
docker_container_horizon	serviceFailure	major	horizon containers missing or down
docker_container_cloudpulse_server	serviceFailure	major	cloudpulse_server containers missing or down
docker_container_novanovncproxy	serviceFailure	major	novanovncproxy containers missing or down
docker_container_novaconsoleauth	serviceFailure	major	novaconsoleauth containers missing or down
docker_container_novassh	serviceFailure	major	novassh containers missing or down
docker_container_novacompute	serviceFailure	major	novacompute containers missing or down

Fault Source	Fault Code	Severity	Description
docker_container_novaapi	serviceFailure	major	novaapi containers missing or down
docker_container_novascheduler	serviceFailure	major	novascheduler containers missing or down
docker_container_novaconduct	serviceFailure	major	novaconduct containers missing or down
docker_container_novalibvirt	serviceFailure	major	novalibvirt containers missing or down
docker_container_novacommon	serviceFailure	major	novacommon containers missing or down
docker_container_cindervolume	serviceFailure	major	cindervolume containers missing or down
docker_container_cinderscheduler	serviceFailure	major	cinderscheduler containers missing or down
docker_container_cinderapi	serviceFailure	major	cinderapi containers missing or down
docker_container_neutron_metadata_agent	serviceFailure	major	neutron_metadata_agent containers missing or down
docker_container_neutron_l3_agent	serviceFailure	major	neutron_l3_agent containers missing or down
docker_container_neutron_dhcp_agent	serviceFailure	major	neutron_dhcp_agent containers missing or down
docker_container_neutron_server	serviceFailure	major	neutron_server containers missing or down
docker_container_neutron_common	serviceFailure	major	neutron_common containers missing or down
docker_container_glanceapi	serviceFailure	major	glanceapi containers missing or down
docker_container_glancer	serviceFailure	major	glancer containers missing or down
docker_container_keystone	serviceFailure	major	keystone containers missing or down

Fault Source	Fault Code	Severity	Description
docker_container_rabbitmq	serviceFailure	major	rabbitmq containers missing or down
docker_container_mariadb	serviceFailure	major	mariadb containers missing or down
docker_container_haproxy	serviceFailure	major	haproxy containers missing or down
docker_container_cephmon	serviceFailure	major	cephmon containers missing or down
docker_container_fluentd	serviceFailure	major	fluentd containers missing or down
docker_container_prometheus	serviceFailure	major	prometheus containers missing or down
docker_container_cvim_mon	serviceFailure	major	cvim_mon containers missing or down
docker_container_alertmanager	serviceFailure	major	alertmanager containers missing or down
docker_container_vmtp	serviceFailure	major	vmtp containers missing or down
docker_container_vimconfig	serviceFailure	major	vimconfig containers missing or down
docker_container_fluentd_aggr	serviceFailure	major	fluentd_aggr containers missing or down
docker_container_curator	serviceFailure	major	curator containers missing or down
docker_container_kibana	serviceFailure	major	kibana containers missing or down
docker_container_elasticsearch	serviceFailure	major	elasticsearch containers missing or down
docker_container_tftp_server	serviceFailure	major	tftp_server containers missing or down
docker_container_my_cobbler	serviceFailure	major	my_cobbler containers missing or down
docker_container_repo_mirror	serviceFailure	major	repo_mirror containers missing or down

Fault Source	Fault Code	Severity	Description
docker_container_container_registry	serviceFailure	major	container_registry containers missing or down
ceph_osdmap_num_in_osds	resourceThreshold	major	|Some ceph OSDs are not IN
ceph_osdmap_num_up_osds	resourceThreshold	major	|Some ceph OSDs are not UP
ceph_pgmap_state_count	resourceUsage	major	Ceph PG Map State not all active+clean
ceph_pgmap_bytes_avail_filling_up_in_4h	resourceUsage	major	ceph_pgmap_bytes_avail will drop to zero within 4 hours.
ceph_pgmap_bytes_used_percent	resourceThreshold	alert	ceph_pgmap_bytes_used percent
ceph_pgmap_bytes_used_percent	resourceThreshold	Major	ceph_pgmap_bytes_used percent is
haproxy_plugin_data_absent	other	informational	HAProxy Telegraf plugin not returning data.
haproxy_active_servers_down	serviceFailure	major	HAProxy active server status not UP.
haproxy_active_servers_backend	serviceFailure	critical	HAProxy active server backends should be 3 but it is in $value.
haproxy_active_servers_galera	serviceFailure	major	HAProxy galera_cluster-internal_vip active should be singular but it is in $value.
haproxy_backup_servers_galera	serviceFailure	major	HAProxy galera_cluster-internal_vip backup should be 2 for quorum but it is in $value.
rabbitmq_node_running	serviceFailure	major	Rabbitmq nodes running should be 9 but is $value.
rabbitmq_queue_messages	resourceUsage	major	Rabbitmq queued message total is too high, but is $value.

Fault Source	Fault Code	Severity	Description
rabbitmq_node_mem_used_percent	resourceThreshold	major	Rabbitmq node {{ $labels.node }} memory usage is $value || printf "%.2f" }}%.
rabbitmq_node_disk_free_limit_percent	resourceThreshold	major	Rabbitmq node {{ $labels.node }} disk usage is $value || printf "%.2f" }}%.
ntpq_offset	resourceThreshold	alert	ntpq_offset is $value
memcached_restarted	serviceFailure	alert	memcached restarted
cp_galera_down	serviceFailure	major	Galera Down on Node(s)
cp_container_down	serviceFailure	major	Container(s) Down on Node(s)
cp_openstack_service_down	serviceFailure	major	Openstack Service(s) Down
cp_rabbitmq_down	serviceFailure	major	RabbitMQ Service Down on Node(s)
cp_ceph_error	serviceFailure	major	CEPH Storage in Error State
cp_hypervisor_down	serviceFailure	major	Hypervisor(s) in Down State

The following OpenStack services alerts based on Cloudpulse metrics are supported:

Table 12: Openstack service alerts and their parameters

Alert	Parameter	Description
openstack service is down	service name (nova, glance, cinder, keystone, neutron)	Triggers if any of the openstack services is down
rabbitmq server is down		Triggers whenever any rabbitmq server goes down
galera server is down		Triggers whenever any galera server goes down
ceph is in error state	error description	Triggers whenever the ceph cluster gets into error state
hypervisor down	hypervisor name	Triggers whenever a hypervisor goes down

Alert	Parameter	Description
infra container is down	container name or host name	Triggers whenever any CVIM container goes down

CVIMMON Web User Interface

The CVIMMON 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 CVIMMON web user interface is available by pointing a web browser to the management node IP address 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.

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.

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 allow 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:

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

Specialized Dashboards

Table 13: List of specialized dashboards

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

CVIM-TRAP

Along with CVIM-MON, CVIM-Trap enables 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 Series-C servers

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/.

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

You can enable CVIMMON on an existing pod through the reconfigure option, if the pod is fresh installed with CVIM 2.4.3. 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.

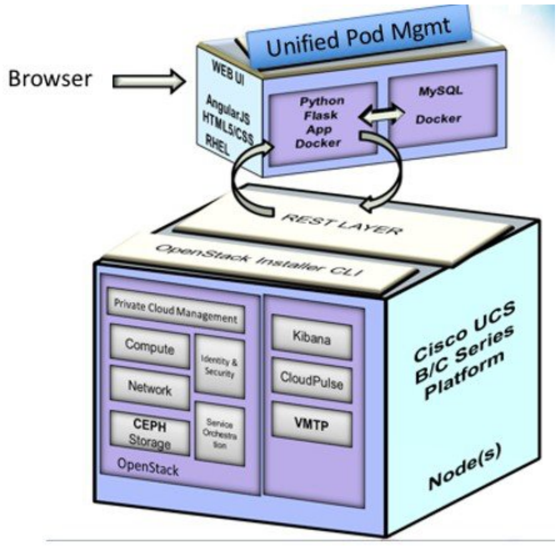
Overview to Cisco VIM Unified Management

CiscoVIM 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.

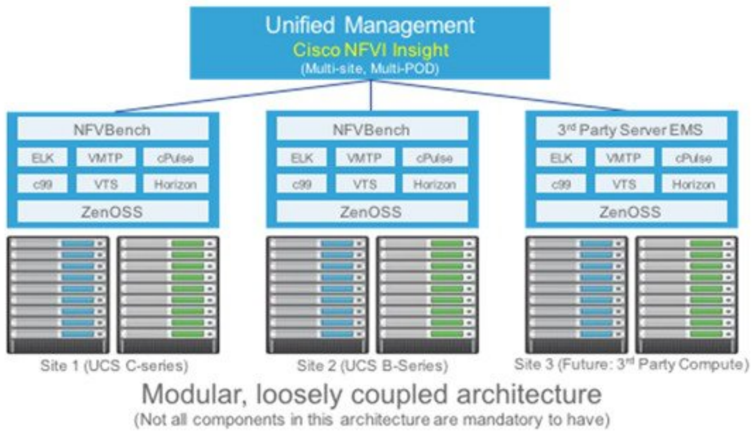
Cisco VIM UM supports multi-tenancy with local RBAC support and is easily integrated with the CiscoVIM REST layer. The container based UI platform is loosely coupled, and can help manage multiple CiscoVIM pods right from day-0, or later in the lifecycle of the cloud.

Figure 24: 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 .

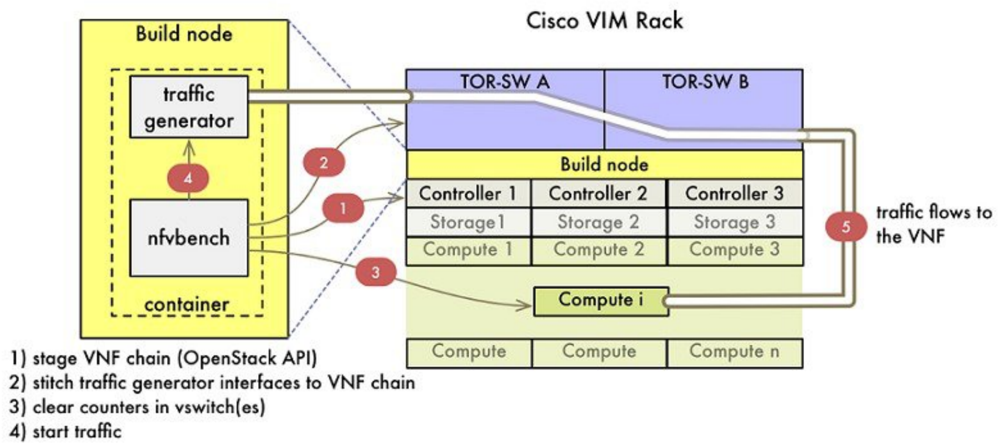
Figure 25: Cisco VIM UM Architecture



Overview to NFVBench

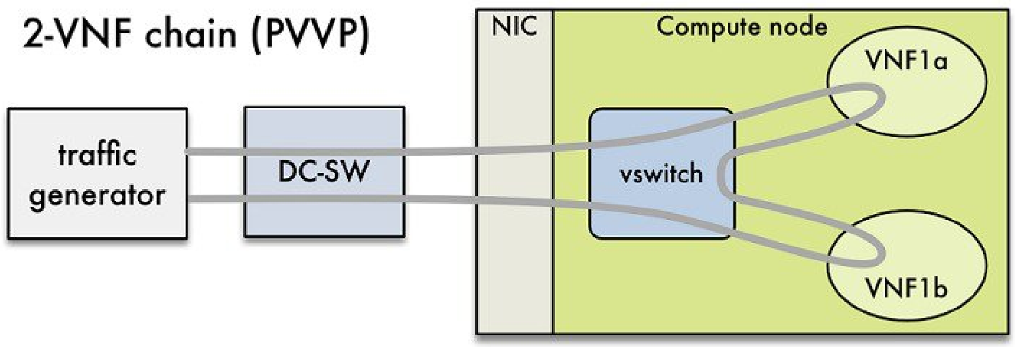
NFVBench is a containerized network benchmarking tool that is introduced in Cisco VIM, to bring consistent methodology to measure the network performance of the cloud. NFVBench is offered in a container that is preinstalled on the management node.

Figure 26: Order of Steps Performed in NFVBench Test



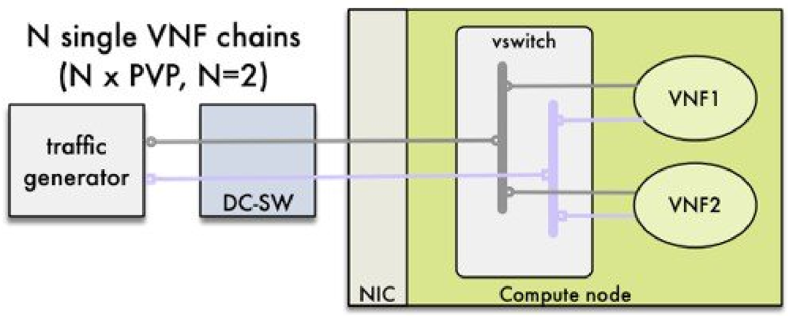
The main goal of NFVBench is to measure the cloud performance that is based on real cloud deployments and not on synthetic, hypothetical lab test environment. So, during the test the packet path must traverse through every network element that participates in the production environment; that is traffic flows through switch (ToR) to v-switch on compute node, continues to VM representing any basic VNF in NFV deployment and comes back similar way on different ports. Network performance or throughput is computed based on sent and received traffic.

Figure 27: Packet Path with Two VNFs



Also it helps to verify network configuration and possible bottlenecks. Reports from NFVBench show data measurements from every element in path, which makes it easier to detect configuration errors or potential bottlenecks. NFVBench sends Layer2 or Layer3 packets that are generated by open-source traffic generator (TRex) already included in the container. Advanced testing using NFVBench allows you to conduct the multichaining and multiflow testing. Multichaining testing enables you to run multiple parallel independent packet paths at the same time, while the multiflow testing performs IP ranging in packet headers within every chain.

Figure 28: Multichaining 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.

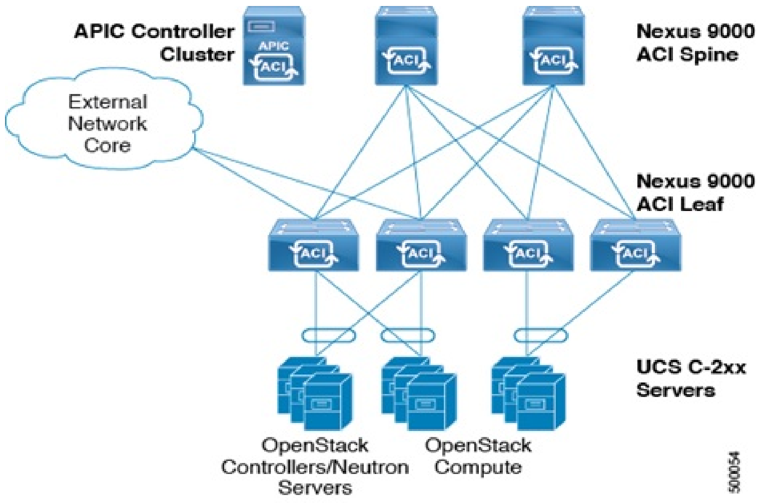
Both the NDR/PDR Test and Fixed Rate Test types of test provide a way of verifying network performance of NFV solution.

Overview to ACI Plugin Integration

The following section gives you an overview of a typical architecture for an ACI fabric with an OpenStack deployment. An ACI with OpenStack deployment consists of a Nexus 9000 Spine/Leaf topology, an APIC cluster, a minimum of 3-node cluster of Controllers (which also acts as the Neutron network node), and two or more compute nodes to host Virtual Machine (VM) instances.

ACI External Routed Network connection is a Layer 3 connection outside the fabric. It is used to provide connectivity outside the OpenStack cloud, as depicted in the following figure.

Figure 29: ACI with OpenStack Physical Topology



Note Basic ACI architecture can be obtained at documentation available in CCO.

In Cisco VIM, we have integrated the Opflex ML2 plugin (in Unified mode) to manage the tenant VLANs dynamically, as VMs come and go in the cloud. By utilizing OpFlex, the policy model native to ACI can be extended all the way down into the virtual switches running on OpenStack Nova compute hosts. OpFlex extension to the compute host allows ACI to use Open vSwitch (OVS) to support common OpenStack features such as Source NAT (SNAT) and Floating IP in a distributed manner.

Cisco VIM extends the automation to include the day-0 ToR level configuration to work with ACI, except for L3 out. The exception for L3 out was made because you can configure their upstream infrastructure in different ways. In the current offering, Cisco VIM with the address scope along with ACI is not supported.

Note Cisco VIM is validated against APIC 3.0, hence it is imperative to use APIC 3.0 version only.

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

Pod Type/Node Types	Control	Storage	Compute	AIO	HC
Full On	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
Micro-Pod (aio)	n/a	n/a		CPU: 2+C+V cores RAM: 41+Vr GB	N/A

Table 15: Number of Physical Cores and RAM Reserved for Cisco VIM Infrastructure

Variables	Usage	Valid range	Default
C	Cores reserved for CEPH (aio and hc)	2..12	HDD: • aio:4 • hc: 8 all SSD: • 2 cores/SSD
V	Cores reserved for VPP vswitch	2.4	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.

Software Distribution Server (SDS)

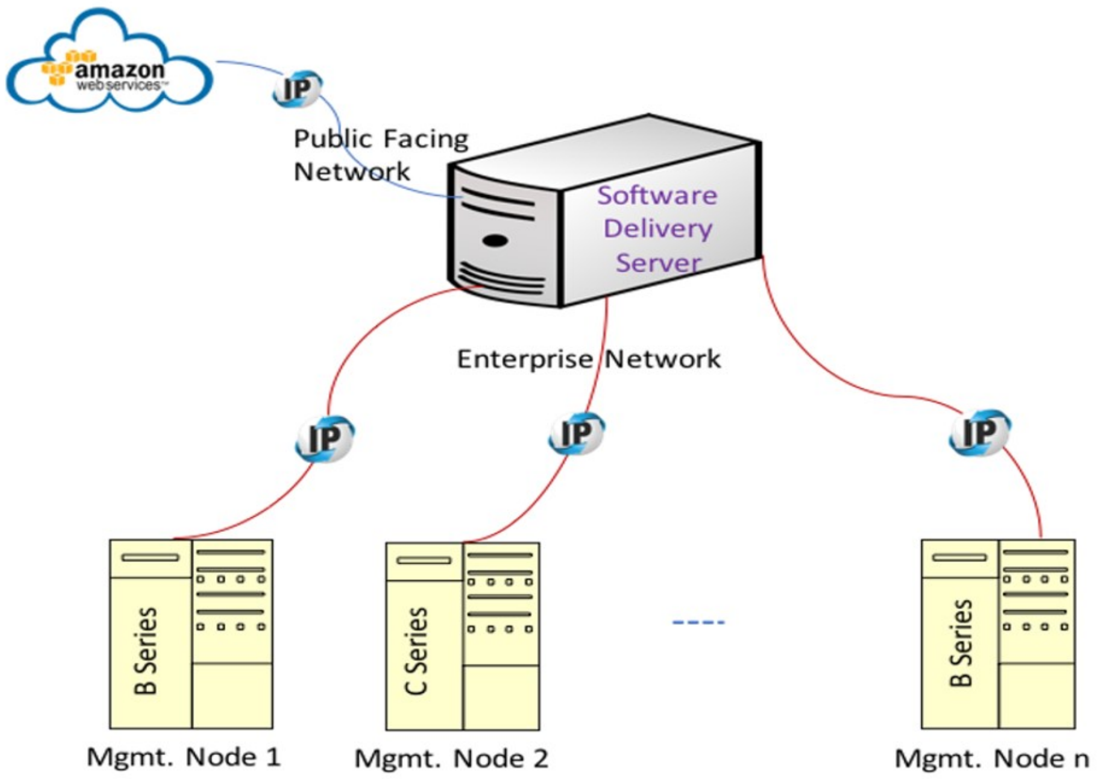
Cisco VIM is supported in an air-gapped (disconnected mode) environment. You can use a USB or Cisco VIM-SDS (Software Delivery Server) for an air-gapped install. When the number of pods is more, shipping USBs for an air-gapped install and update is not scalable. In such scenarios, we recommend that you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

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

You can install the artifacts available on the SDS server through a connected or a disconnected install procedure. For a connected install, one end of the SDS 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

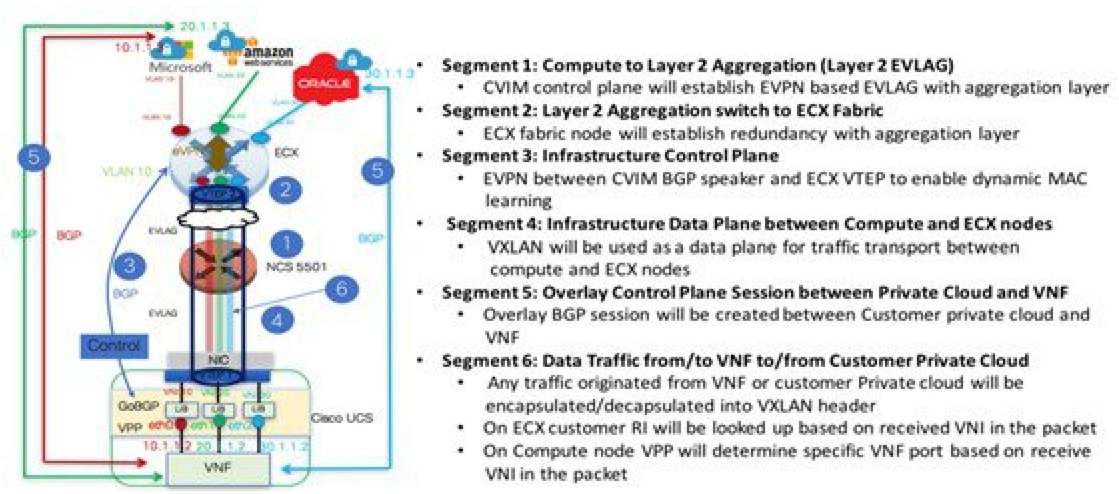


For a disconnected install, both interfaces are private and the artifacts are installed on the SDS using the USB procedure. You must ensure that the ssh interface (br_api) of the management node for each Cisco VIM pod can connect to the enterprise facing interface of the SDS server through Layer 2 or Layer 3 networking.

Cisco VIM VXLAN EVPN Design

From the release Cisco VIM 2.4.3, 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



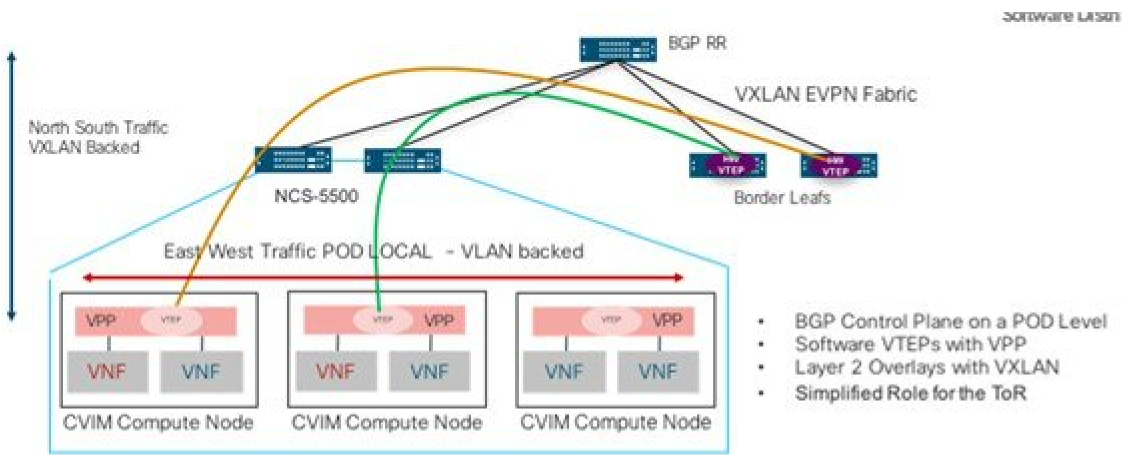
- **Segment 1: Compute to Layer 2 Aggregation (Layer 2 EVLAG)**
  - CVIM control plane will establish EVPN based EVLAG with aggregation layer
- **Segment 2: Layer 2 Aggregation switch to ECX Fabric**
  - ECX fabric node will establish redundancy with aggregation layer
- **Segment 3: Infrastructure Control Plane**
  - EVPN between CVIM BGP speaker and ECX VTEP to enable dynamic MAC learning
- **Segment 4: Infrastructure Data Plane between Compute and ECX nodes**
  - VXLAN will be used as a data plane for traffic transport between compute and ECX nodes
- **Segment 5: Overlay Control Plane Session between Private Cloud and VNF**
  - Overlay BGP session will be created between Customer private cloud and VNF
- **Segment 6: Data Traffic from/to VNF to/from Customer Private Cloud**
  - Any traffic originated from VNF or customer Private cloud will be encapsulated/decapsulated into VXLAN header
  - On ECX customer RI will be looked up based on received VNI in the packet
  - On Compute node VPP will determine specific VNF port based on receive VNI in the packet

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

- OpenStack will manage VLAN allocation.
- Customer will manage VXLAN network and subnet for overlays, and enable OpenStack to use the EVI/VNID by creating appropriate networks/subnets in OpenStack.
- 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

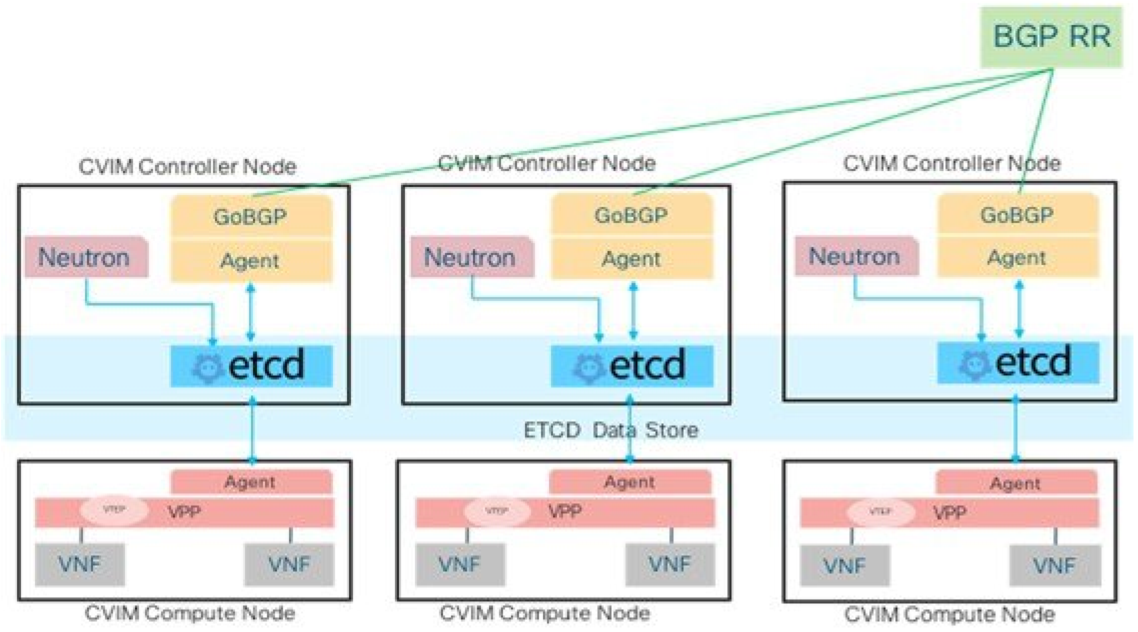


- BGP Control Plane on a POD Level
- Software VTEPs with VPP
- Layer 2 Overlays with VXLAN
- Simplified Role for the ToR

From a control plane point of view, three instances of GoBGP (in Active-Active-Active mode) run on the controller nodes, and peer with the external BGP RR, by which VxLAN routes are imported into and exported from Cisco VIM. 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



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

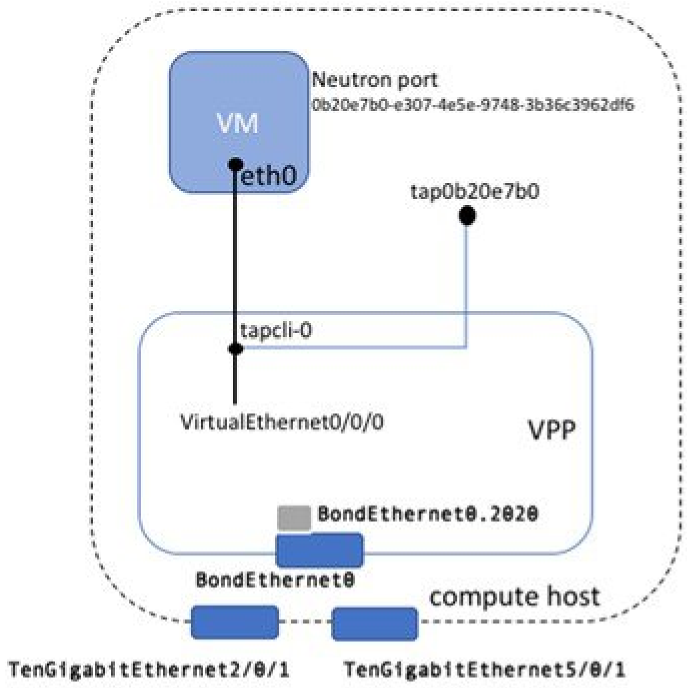
Port mirroring works by setting up the following:

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.