HCLTech | Supercharging Progress™

VIM and NFVI Architecture and Installation Guide 6.0.1

First Published Date: 25th September, 2026

Contents

NFVI Architecture **1**
Network Function Virtualization Infrastructure Overview 1
Virtualized Infrastructure Manager Overview 6
Networking Overview **7**
UCS C-Series Network Topologies **15**
Management Node Networking **22**
IPv6 Support **24**
IPv6 Support on Management Network 24
UCS C-Series **24**
High Availability **26**
Storage Node Overview **28**
NFVBench **30**
Setting up NFVbench 30
Encapsulation **49**
VIM CLI **50**
NCS-5500 as ToR Option **52**
Disk Management **52**
OSD Maintenance **52**
Power Management **53**
Physical Cores and Reserved Memory **53**
Software Hub **54**
VPP Port Mirroring Support **57**
Segment Routing EVPN **58**
L3 Fabric Deployment **62**
OpenStack Barbican **62**
Management Network Over TLS **64**
Support of Pre-encrypted Image **64**
Installation **65**
NFVI Installation Overview **65**
Installation Mode **66**

Installation Preparation Without Internet Access 66

Air-gapped Installation Approach 66

Approaches for Air-Gapped Installation 66

Prerequisites for Air-Gapped Installation 66

Prerequisites for Air-gapped Installation via USB 66

NFVI Installation Setup via USB 67

NFVI Installation File-Based Image 71

Preparing for NFVI Installation 73

Note: VIM no longer supports Cisco UCS-M4 Hardware 73

NFVI Hardware Installation 73

ToR Switch Configuration for C-Series Pods 76

Preparing Cisco IMC 80

Management Node on UCS C-series (M5/M6/M7) 80

VIM Software Hub 82

UCS C-Series Pod 92

Setting Up UCS C-Series Pod 92

Out-of-Band Management Switch 96

Configuring Out-of-Band Management Switch 96

VIM Monitor 97

VIM Monitor Configurations 97

Enabling VIM-MON on VIM 97

Highly Available VIM Monitor 115

Overview of HA VIM-MON 115

Hardware Requirements for HA VIM MON 117

Networking Layout 117

Network Topologies 118

Network Topologies for HA VIM-MON 118

Architecture 120

Installation Modes 120

Installation Modes for HA VIM-MON 120

Setup File 123

Configuring Setup Data File for HA VIM-MON 123

Domain Mappings 126

Group Mappings 126

HA VIM-MON Installer **128**
Installation and Installer CLI 128
External Servers **133**
Prerequisites for Monitoring External Servers Using HA VIM MON 133
Resources **133**
Resources for Managing HA VIM-MON Cluster 133
Pod Operations **135**
Supported Pod Operations 135
Removing a Worker Node (--remove-worker) 139
Stack Operations **152**
Stack Operations 152
HA VIM-MON Update **155**
Updating Software of HA VIM MON Nodes 155
Custom Grafana Dashboards **156**
Custom Grafana Dashboards for HA VIM MON 156
Alert Rules **161**
Customizing Alerting Rules 161
Alert Manager **164**
Customizing Alert Manager and Receivers 164
Backup **168**
Backing Up HA VIM-MON 168
Restore **169**
Restoring HA VIM-MON 169
Reinstallation of VIM-MON HA Pod **170**
Reinstallation of VIM-MON HA Pod 170
VIM MON HA Cluster Monitoring **171**
VIM MON HA Cluster Monitoring 171
HA VIM Mon Upgrade **173**
Upgrade Procedure 174
Upgrade steps: HA VIM MON from 4.2.x (x=1,2,3,4) to 4.4.y (y= 3,4) 176
SERVER MON **180**
Opensearch-Fluentd-Opensearch Dashboard for VIM-MON HA Management Node **182**
HA VIMMON - Opensearch Dashboard Login: 184
LDAP Authentication for Opensearch dashboard in VIM-MON HA 187
LDAP Integration for VIM-MON HA Management Node **189**
VIM 192

Installation Sequence **192**
Installation Sequence 192
Deployment of Management Node **193**
Deployment of Management Node 193
VIM Client Details **194**
VIM Client Details 194
Pod Reinstallation **199**
Pod Reinstallation 199
Control and Data Plane Testing **200**
Control and Data Plane Testing 200
Updating VIM Software **201**
Updating VIM Software 201
Unified Management **203**
Installing VIM Unified Management 203
UM Overview **203**
UM Overview 203
UM with Internet Access **205**
Installing UM with Internet Access 205
UM with VIM Software Hub **220**
UM with VIM Software Hub 220
UM UI User Authentication and Authorization with LDAP **220**
UM UI User Authentication and Authorization with LDAP 220
UM Without SMTP **223**
Installing UM Without SMTP 223
UM Without Internet Access **226**
Installing UM without Internet Access 226
UM Optional Services **234**
UM Optional Services 234
UM Post Bootstrap Validation Checks **234**
UM Post Bootstrap Validation Checks 234
UM Admin Login for Standalone Setup **243**
UM Admin Login for Standalone Setup 243
UM Pod Admin Login for Standalone Setup **243**
UM Pod Admin Login for Standalone Setup 243
UM Host Administrators **244**
Managing UM Host Administrators 244

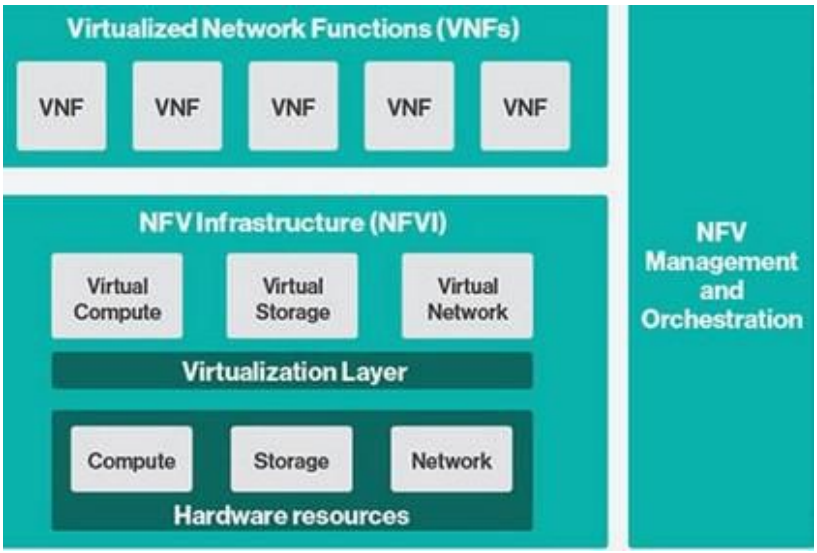
Reinstallation of UM Node **251**
Reinstallation of UM Node 251
Verifying Installation **252**
Verifying Installation 252
Displaying IP Addresses **252**
Displaying IP Addresses 252
VIM Client CLI Availability **253**
VIM Client CLI Availability 253
Displaying NFVI Logs **254**
Displaying NFVI Logs 254
Accessing OpenStack API Endpoints **256**
Accessing OpenStack API Endpoints 256
Assessing NFVI Health **256**
Assessing NFVI Health with CloudPulse 256
HA Proxy Dashboard/ELK Stack Logs **259**
Displaying HA Proxy Dashboard and ELK Stack Logs 259
Testing Pod/Cloud Infrastructure **259**
Testing NFVI Pod and Cloud Infrastructure 259
UM Without SMTP **259**
Installing UM Without SMTP 259

NFVI Architecture

- NFVI Overview
- VIM Overview
- Networking Overview
- UCS C-Series Network Topologies
- Management Node Networking
- IPv6 support
- UCS C-Series
- High Availability
- Storage Node Overview
- NFVBench
- NCS-5500 as ToR Option
- Disk Management
- OSD Maintenance
- Power Management
- Physical Cores and Reserved Memory
- Software Hub
- VPP Port Mirroring Support
- Segment Routing EVPN
- L3 Fabric Deployment
- OpenStack Barbican
- Management Network Over TLS
- Support of Pre-encrypted Image

Network Function Virtualization Infrastructure Overview

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



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

NFVI includes a virtual infrastructure layer (VIM) that embeds the Red Hat OpenStack Platform (OSP). VIM includes OpenStack, which is an open-source cloud operating system that controls large pools of compute, storage, and networking resources. VIM manages the OpenStack compute, network, and storage services, and all NFVI

© 2026 HCLTech. All rights reserved.

management and control functions. VIM is an embedded solution and must be treated as one or else it violates the support agreement of the product. The activities such as changing system files and directories, installing/upgrading rpms and/or enabling repositories outside VIM, are examples of such violations. For clarifications, reach out to HCLTech support.

Key NFVI roles include:

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

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

- Cisco UCS® C240 M5/M6: Performs management and storage functions and services. Includes dedicated Ceph (UCS 240- M5) distributed object store and file system. (Only Red Hat Ceph is supported).
- Cisco UCS C220/240M5/M6/M7: Performs control and compute services.
- Cisco UCS 220/240 M5/M6 (SFF): In a Micropod environment, it is expandable to a maximum of 16 computes.
- Cisco UCS C220 M7: In a Micropod environment that serves as control, compute and storage. In a Full-on pod environment control and compute, for storage 240 M5/M6.
- A combination of M5 series servers is supported in M5-based Micropod and VIC/NIC (pure 40G) based hyper-converged and micropod offering.
- Quanta servers are an alternative to Cisco UCS servers: Use of specific Quanta servers for the installation of the cloud both at the core and edge.
- An automated install of the Central Ceph cluster to the edge pods is supported for Glance image services.
- UCS 220/240 M5 in a nano-pod environment with 2-X710 Intel NIC or 25G/40G VIC/NIC BOM

The UCS C240 and C220 servers are of type M5 or M6 Small Form Factor (SFF) models where the nodes can boot off a pair of HDDs or SSD as specified in BOM.

The VIM installer performs bare metal installation and deploys OpenStack services using Docker™ containers to allow for OpenStack services and pod management software updates.

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

Function	Number	Hardware	Services
Management	1	• 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 • UCS C240 M6 SFF with 8x1.2 TB HDDs • UCS C220 M6 SFF with 8x1.2 TB HDDs • Quanta server (D52BE-2U) with 2x1.2TB HDD • Quanta server (D52BQ-2U 3UPI) with 2x.3.8TB HDD	• VIM Installer • Cobbler server • Docker Registry • ELK server • VIM MON components: Prometheus and TSDB
Control	3	• UCS C220/C240 M5/M6 with 2x1.2 TB HDDs or 2x960 GB SSDs (in a Micropod or Full pod environment) • Quanta server (D52BE-2U) with 2x960 GB SSD • Quanta server (D52BQ-2U 3UPI) with 2x960 GB SSD for edge pod	• Maria Database/Galera • RabbitMQ • HA Proxy/Keepalive • Identity Service • Image Service • Compute management • Network service • Storage service • Horizon dashboard

© 2026 HCLTech. All rights reserved.

Function	Number	Hardware	Services
			• Fluentd
Compute	2+	• UCS C220/C240 M5/M6 with 2x1.2 TB HDDs, or 2x960 GB • SSDs (in a Micropod or Full pod environment) • HP DL360 Gen9 • Quanta server (D52BE-2U/ D52BQ-2U 3UPI) with 2x960 GB SSD	• Virtual Networking Service • Compute service • Fluentd
Storage	3 or more	SSD and HDD drives must be in a 1:4 ratio per storage node minimum. Storage node configuration options: Fullon environment: • UCS C240 M5 with two internal SSDs, 1-4 external SSD, 4- 20x- 1.2 TB HDDs • SSD-based Ceph: UCS C240 M5 with 2 internal SSDs, minimum of 4 external SSDs, expandable to 24 SSDs • Quanta server (D52BE-2U) HDD Based: 4 SSD 960GB for Journal. + 16 SAS HDD (16x2.4 TB) for OSD + 2 (2x2.4 TB SAS 10krpm HDD) for OS • Quanta server (D52BE-2U) SSD Based: 20 SSD (3.8 TB) OSD + 2 OSBoot (2x3.8TB SSD) Micropod/UMHC/NGENAHC environment: • UCS C240 M5/M6 with two 1.2TB HDD for OS boot, one/2 SSDs and 5/10x1.2 TB HDDs • UCS C240 M5/M6 with 2x960GB SSD for OS boot and 4 or 8 x960 GB SSDs	Storage service
Top of Rack (ToR)	1 or more pairs	• Cisco Nexus 9000 series as a standalone (recommended: nxos 9.3 (3), nxos 9.3(8)) • Cisco Nexus 9000 switches controlled by ACI Cisco NCS 5500	• Cisco NCS 5500 provides ToR service with VIM running on C-series and with Intel NIC and VPP as the mechanism driver for deployment.

⚠

- Internal SSD is the boot device for the storage node other than M6.
- You can use any ToR that supports virtual port channel. Cisco recommends using Cisco Nexus 9000 SKUs as ToR, which is released as part of vim.
- When Cisco NCS 5500 acts as a ToR, the option of Auto-ToR or *ships in night* is available similar to that of NXOS. If Auto-ToR option is used, maximum of two Cisco NCS can be used.

Software applications that manage NFVI hosts and services include:

Red Hat Enterprise Linux (RHEL) with OpenStack Platform (OSP): Provides the core operating system with OpenStack capability. RHEL and OSP are installed on all target NFVI nodes.

© 2026 HCLTech. All rights reserved.

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, considering hardware and software redundancy, security, and monitoring. VIM includes OpenStack with more features and usability enhancements that are tested for functionality, scale, and performance.

- Unified Management: Deploys, provisions, and manages VIM on Cisco UCS servers.
- Cisco UCS Manager: Supported UCS manager firmware versions are 2.2(5a) and above.
- Cisco Integrated Management Controller (CIMC): When installing VIM, Cisco IMC 4.x or later is recommended.

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

UCS-M4 servers	**Starting from VIM 4.4.3, fresh install on UCS-M4 servers are no longer supported. Kindly plan to migrate/use UCS M5 or M6 servers.**
UCS-M5 servers	For UCS-M5-C220 and UCS-M5-240 • Use the UCS suggested release based on software quality, stability and longevity IMC 4.3(2.250045)*. HVIM BU tested this version in lab on Rhel VM's. • Do not use 3.1(3c) to 3.1(3h), 3.0(4a), 4.0(2c), or 4.0(2d). * Indicates that it is tested with Intel NIC BOM. Other versions of IMC may work, but you must test VIM deployment with those versions before rolling it into production. We recommend the customers to move forward with the new releases of IMC versions after testing and validating them in a non-production environment if old releases are deprecated or if new fixes solve their problem. There is no support for GPU.
UCS-M6 servers	Use the UCS suggested release based on software quality, stability and longevity IMC 4.3(2.250045) *. HVIM BU tested this version in lab on Rhel VM's
UCS-M7 servers	UCS_C220: Use the UCS suggested release based on software quality, stability and longevity IMC 4.3(6.250053)*. HVIM BU tested this version in lab on Rhel VM's

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

The Quanta servers also need to run with a minimum version of BMC and BIOS version, which is listed below:

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

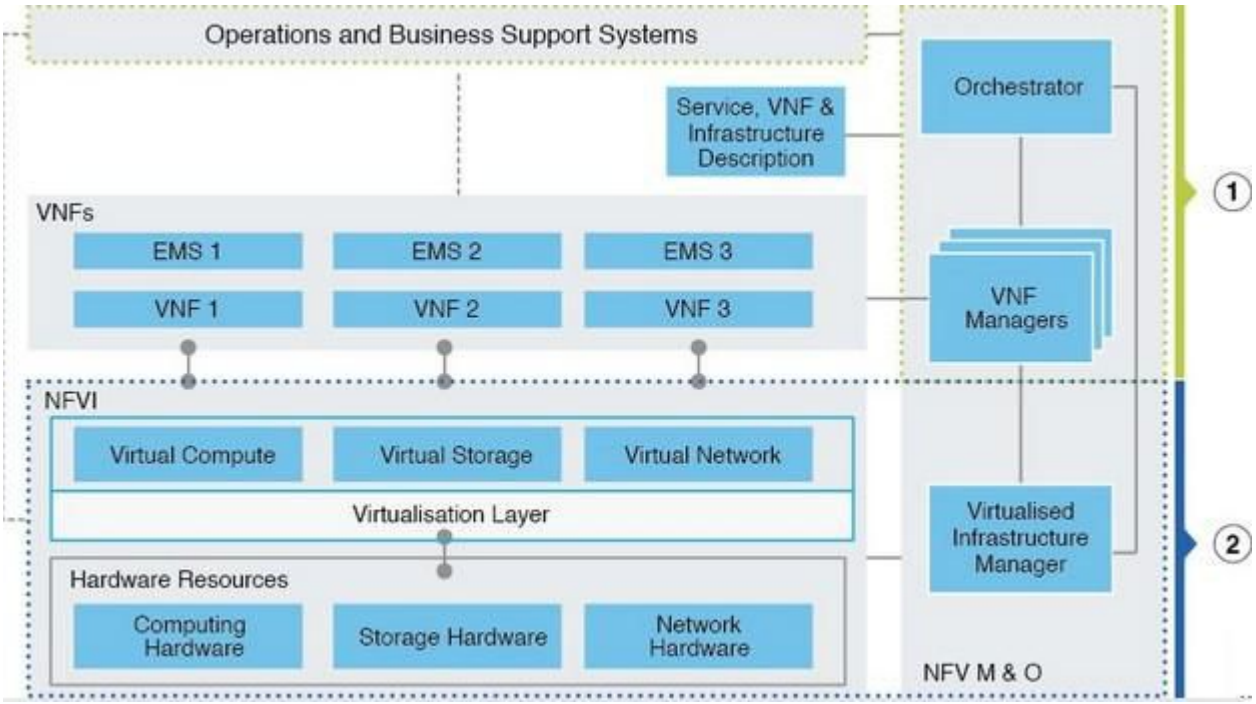
Two VNF orchestration and management applications that are used with NFVI include:

© 2026 HCLTech. All rights reserved.

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. Cisco ESC allows you to automatically instantiate, monitor, and elastically scale VNFs end-to-end. For information about Cisco ESC, see Cisco Elastic Services Controller Data Sheet.

The following figure shows the NFVI architecture with Cisco NSO and Cisco ESC.



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

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

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

In addition, with a C-series pod, you can choose to run with an augmented performance mechanism by replacing OVS/VLAN with VPP/VLAN (for Intel NIC).

The 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. VIM installation also includes service assurance, OpenStack CloudPulse, built-in

© 2026 HCLTech. All rights reserved.

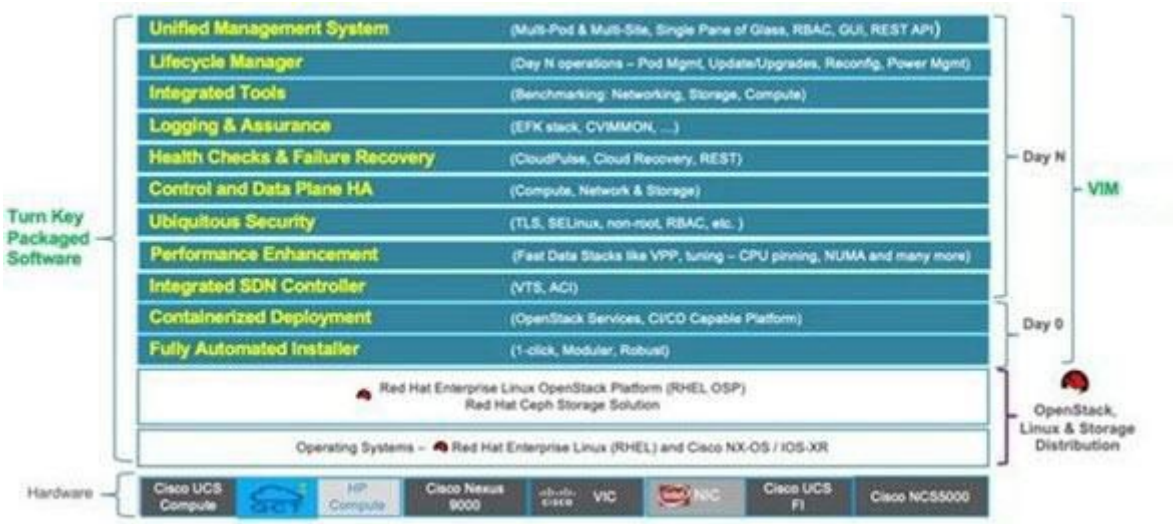
control, and data plane validation. Day two pod management allows you to add and remove both compute and Ceph nodes and replace the controller nodes. The VIM installation embeds all necessary RHEL licenses as long as you use the VIM supported BOM and the corresponding release artifacts.

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

Virtualized Infrastructure Manager Overview

Virtualized 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. VIM offers fully automated Day 1 to Day n cloud lifecycle management. These include capabilities such as pod scaling (expansion), software update, upgrade, or reconfiguration 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 the service providers and enterprises.

The following figure provides a high-level overview of all Day 0 and Day n items of VIM.



© 2026 HCLTech. All rights reserved.

Networking Overview

Networking Overview

- Introduction
- API Segment
- External Segment
- Management and Provisioning Segment
- Storage Segment
- Tenant Segment
- Provider Segment
- Pod with Intel NICs
- Control Plane
- Data Plane
- SRIOV

Introduction

VIM supports installation on two different types of pods. The rack C-series based offering supports Cisco NIC. You can choose the C-series pod to run in a pure Intel NIC environment. Thereby, obtaining 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, VIM segments the various OpenStack networks. The 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. NFVI network segments include API, external, management and provisioning, storage, tenant, and provider.

API Segment

The API segment needs one VLAN in an externally accessible subnet that is different from the subnets assigned to other NFVI segments. These IP addresses are used for:

- OpenStack API endpoints. These are configured within the control node HAProxy load balancer.
- Management node external connectivity: Ensure that L2/L3 connectivity between the SSH interface (via *br_api)* of the management node to the OpenStack API exists, so that tools like VMTP and NFVbench can work.

External Segment

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

Management and Provisioning Segment

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

© 2026 HCLTech. All rights reserved.

Storage Segment

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 NFVI to carry all Ceph replication traffic. All the participating nodes in the pod will have IP addresses on this subnet.

Tenant Segment

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

Provider Segment

Provider networks are optional for 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.

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 NFVI network segments, usage, network, and VIP names.

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 the API network to interface with OpenStack APIs. • OpenStack Horizon dashboard. • Default gateway for HAProxy container. • Integration with endpoints served by swift cluster for object storage, cinder backup service, or Identity service with LDAP or AD.	api	a
External	Access to VMs using floating IP addresses.	external	e
Storage	Transit network for storage back-end. Storage traffic between VMs and Ceph nodes.	storage	s
Provider Network	Direct access to existing network infrastructure.	provider	p
Installer API	Administrator uses the installer API network to SSH to the management node. Administrator connects to installer API to interface with secured services. For example, Opensearch on the management node.	VIM installer API	br_api

© 2026 HCLTech. All rights reserved.

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_da ta*. *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 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 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 NFVI server role.

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

© 2026 HCLTech. All rights reserved.

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

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

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

Network Interface	Cisco UCS C220/C240 M5 /M6 Compute	HPE ProLiant DL360 Gen9 and Quanta Compute	Dell PowerEdge R740
samx	Management control of the plane network	N/A	Management control of the plane network
samxpet	Converged control and data plane network based on NIC configuration	Control and data plane network for all other than SRIOV: Management network on *br_mgmt* bridge interface with *samxpet* main interface as one of the member interfaces (native VLAN configuration required on the top-of-rack switches) Storage network on the sub-interface *samxpet. <storage VLAN>* Tenant and provider networks on veth interface *pet/pet-out* as one of the member interfaces with *br_mgmt* bridge interface	N/A
pet	Dedicated data plane network (based on NIC configuration)	N/A	Dedicated data plane network
p	Provider data plane network	Provider data plane network	Provider data plane network
sriov [0-3]	Provider data plane SRIOV networks	Provider data plane SRIOV networks	Provider data plane SRIOV networks
s	Storage control and data plane network	Storage control and data plane network	Storage control and data plane network
t	Tenant data plane network	Tenant data plane network	Tenant data plane network

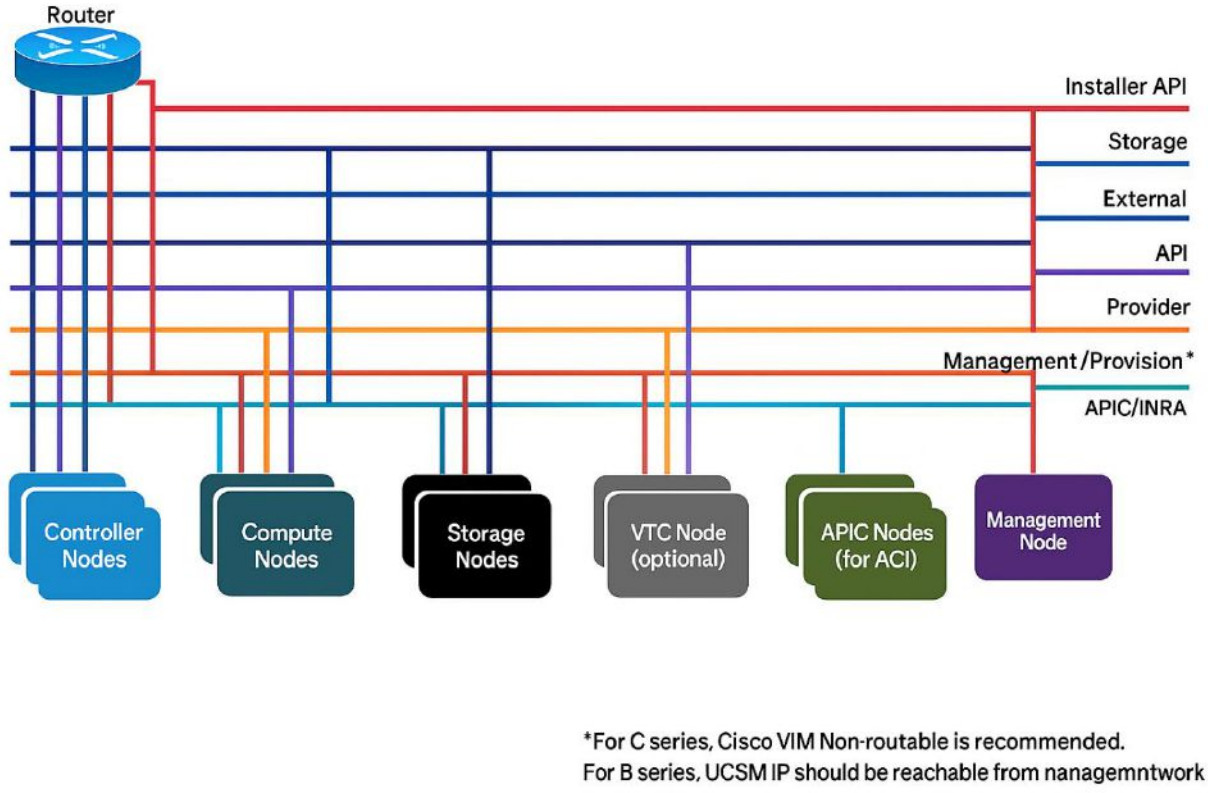
In the above table, *p*, *s,* and *t* stands for provider, storage, and tenant network respectively but their actual implementation is part of *samx* and *pet* network.

In the initial NFVI deployment, two bridges are created on the controller nodes and linked with interfaces and bonds. The *br_api* bridge connects the API.

© 2026 HCLTech. All rights reserved.

(a) interface to the HAProxy. The HAProxy and Keepalive containers have virtual IPs (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 NFVI nodes and networks.



Supported Layer 2 networking protocols include:

• VLAN over Open vswitch (SRIOV with Intel 710NIC). • VLAN over VPP/VLAN for C-series Only.

• SRIOV allows a single physical PCI Express to be shared on a different virtual environment. The SRIOV offers different virtual functions to different virtual components, for example, network adapters, on a physical server.

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

Pod Type	NIC Type	Hardware Vendor	Mechanism Driver	ToR Type
fullon	Cisco VIC	UCS C series M5 UCS C series M6	OVS/VLAN	N9K
fullon	Intel NIC	UCS C series M5	OVS/VLAN with SRIOV	N9K
fullon	Intel NIC	Quanta D52BQ-2U 3UPI	OVS/VLAN with SRIOV	N9K
fullon	Intel NIC	UCS C series M5	VPP/VLAN with SRIOV	N9K NCS-5500

© 2026 HCLTech. All rights reserved.

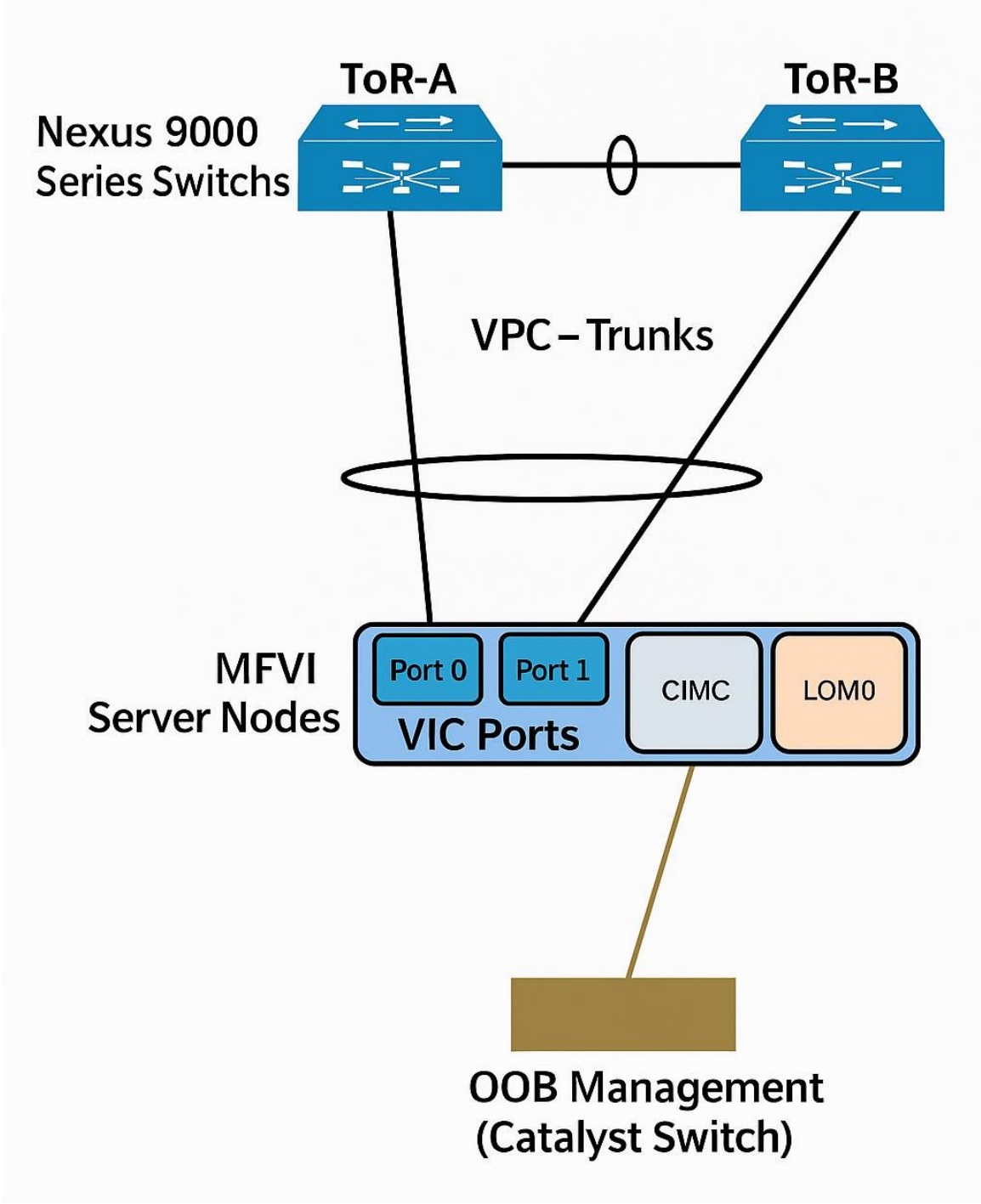
Pod Type	NIC Type	Hardware Vendor	Mechanism Driver	ToR Type
		UCS C series M6		
fullon	Intel NIC	UCS C series M5 with Dell as third-party Compute	VPP/VLAN with SRIOV	NCS-5500
fullon	Cisco VIC with Intel NIC	UCS C series M5/M6 computes UCS C series M5 UCS C series M6	OVS/VLAN (VIC) with SRIOV (Intel NIC)	N9K
micro	Cisco VIC	UCS C series M5 UCS C series M6	OVS/VLAN	N9K
micro	Intel NIC	UCS C series M5	OVS/VLAN	N9K
micro	Intel NIC	UCS C series M5 UCS C series M6	VPP/VLAN	N9K NCS-5500
UMHC	Cisco VIC with Intel NIC	UCS C series M5	OVS/VLAN (VIC) with SRIOV (Intel NIC)	N9K
nano	Cisco VIC	UCS C series M5	OVS/VLAN	N9K
nano	Intel NIC	UCS C series M5	OVS/VLAN	N9K

- fullon indicates the dedicated control, compute and ceph nodes.
- micro indicates converged control, compute, and ceph nodes with expandable computes.
- Hyperconverged (HC) indicates the dedicated control and compute nodes, but all ceph nodes are compute nodes.
- edge indicates converged control and compute nodes with expandable computes. It communicates with Central ceph cluster for
- Glance Image service. Persistent storage is not supported.

ceph indicates converged cephcontrol and cephosd nodes, with an option to add cephosd nodes for glance image services.
nano indicates a single node cloud with control and compute functionality with no persistent storage and no scope for expansion. It can be managed via a virtual or physical management node that is Layer 3 or Layer 2 distance away, respectively, from the nano pod.

The SRIOV support is applicable for Intel NICs in the pods. VTF with VTC is only supported on C-series Cisco VIC.

© 2026 HCLTech. All rights reserved.



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

- Control plane.
- Data plane (external, tenant and non-SRIOV provider network).
- 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:

- Management/Provision.

© 2026 HCLTech. All rights reserved.

- Storage.
- 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 underlying 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 *sam* interface based on the storage, API VLANs. The management/provision traffic is untagged/native VLAN 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 is pet, indicating that it is carrying provider, external and tenant traffic. The underlying interfaces of this bonded interface are visible as *pet0* and *pet1*. *pet0* belongs to the NIC_1 and *pet1* belongs to NIC_2.

SRIOV

For Intel NIC pod, the third port (and optionally the fourth port) from each NIC can be used for SRIOV traffic. This is optional and is set or unset through a *s etup_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 functions to be created are configurable through the *setup_data*. *yaml*. The interface names of the SRIOV are shown as *sriov0* and *sriov1* on each host, indicating that *sriov0* belongs to NIC_1 C port and *sriov1* belongs to NIC_2 C port.

The following table summarizes the interface name and type of traffic for each network plane:

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			+	

⚠ On Intel pod, all kinds of OpenStack networks are created using *physnet1* as the physnet name.

© 2026 HCLTech. All rights reserved.

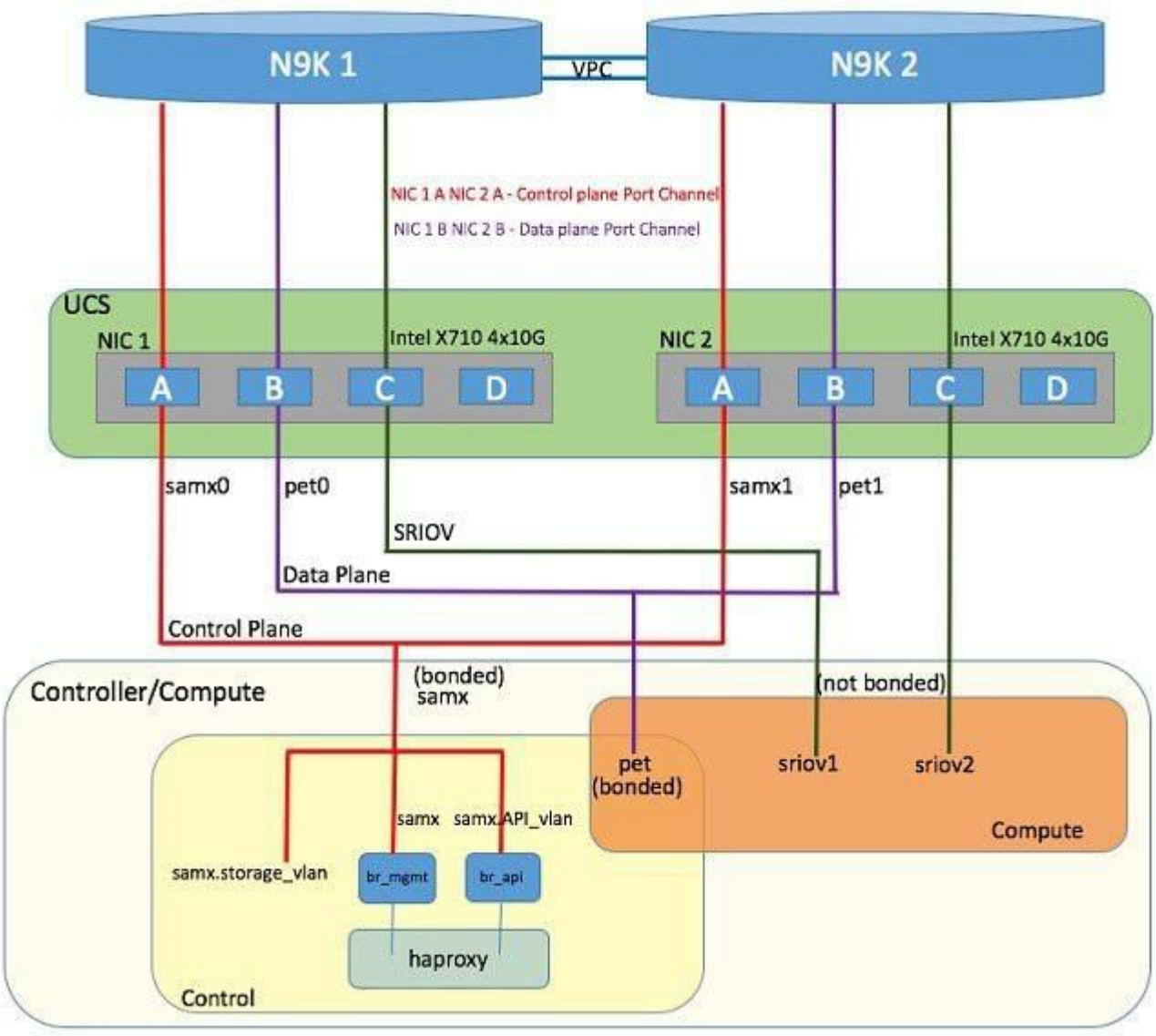
UCS C-Series Network Topologies

UCS C-Series Network Topologies

NFVI UCS servers are connected 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 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 NFVI host to ToR topology.

For Intel NIC, a single two-port Cisco VIC in the preceding figure is replaced with two 4-port 710 Intel NIC. An extra Intel NIC is added to provide card level redundancy as shown in the figure below:



Of the four ports that are available in each NIC card, port A is used for management traffic (provision, API, storage, etc), whereas the port B is used for data plane (tenant and provider network) traffic. Port C (and optionally Port D) is dedicated for SRIOV (configured optionally based on setup_data. yaml). Sub-interfaces are carved out of the data and control plane interfaces to provide separate traffic based on specific roles. While the ports A and B from each NIC help in forming bonded interface, the ports C and D over which SRIOV traffic for provider network flows are not bonded.

© 2026 HCLTech. All rights reserved.

You must take extreme care during pod setup, so that ports A, B and C for the Intel NIC is connected to the ToRs. You can optionally use port D as the second pair of SRIOV ports with appropriate intent defined in the *setup_data.yaml* file. From VIM release 2.4.2 onwards, this port option is available for M5 based systems or pods.

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

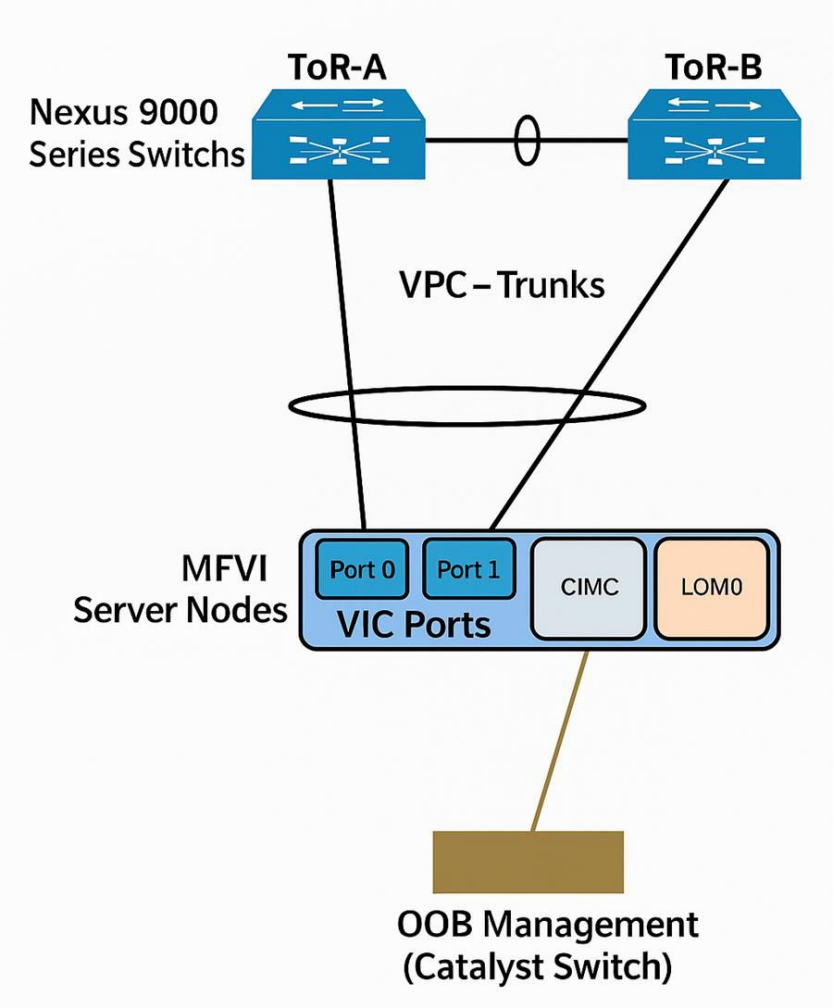
Server/POD Type	Target Functions	Default NIC Layout
M5 Intel NIC based	Control Plane	NIC-1 A + NIC-1 B
	Data Plane	NIC-1 C + NIC-1 D
	SRIOV 0/1	NIC-2 A + NIC-2 B
	SRIOV 2/3	NIC-2 C + NIC-2 D

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

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 (up to a max of 16). The M5 micropod environment is based on X710 for control and data plane and an additional XL710 or 2xX710 for SRIOV. The SRIOV card is optional. Once the SRIOV card is chosen, all the computes must have the same number of SRIOV ports across the pod.

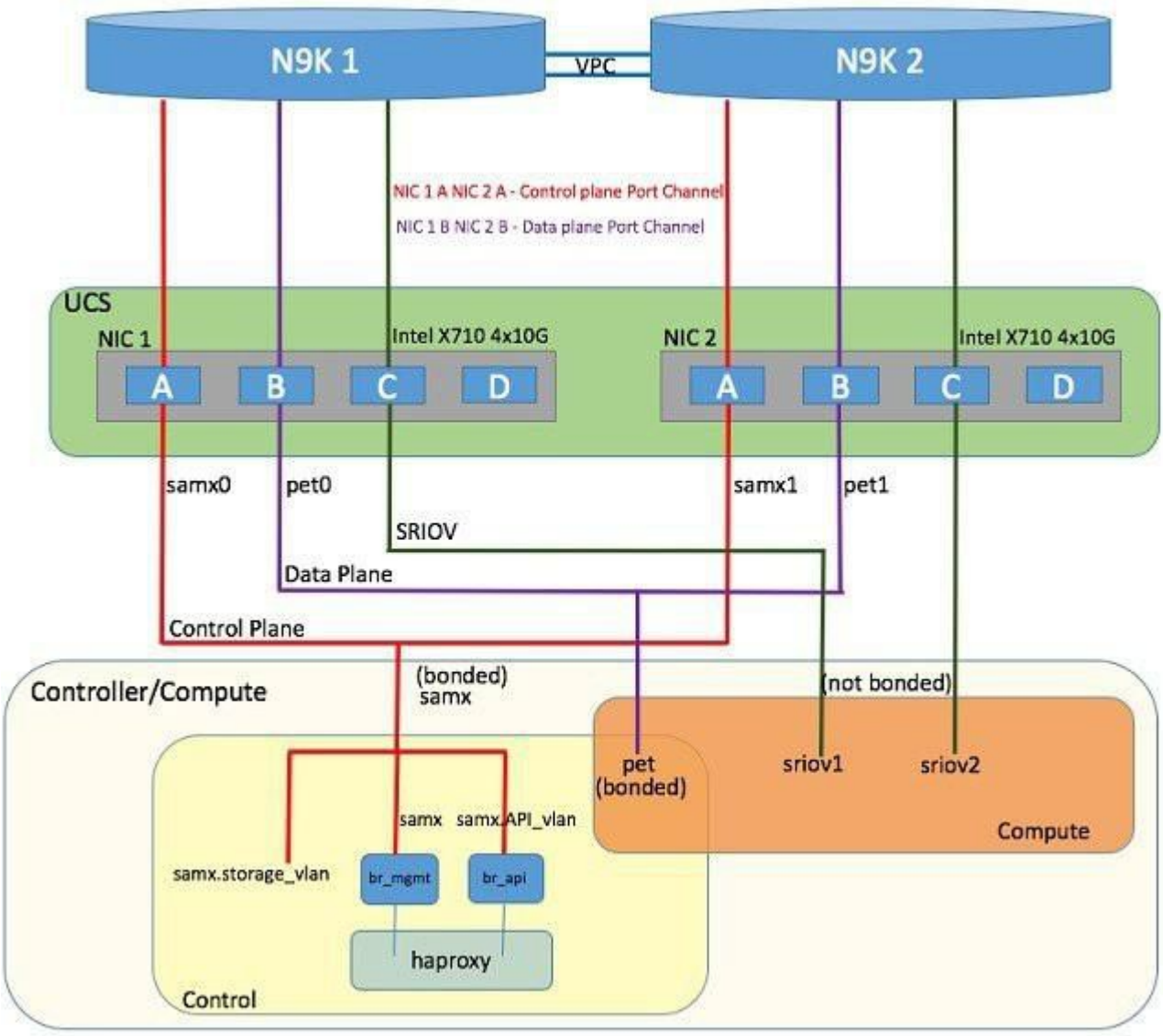
The following diagram depicts the server network card diagram for the UCS-M5 micropod setup.

© 2026 HCLTech. All rights reserved.



For Intel NIC, a single two-port Cisco VIC in the preceding figure is replaced with two 4-port 710 Intel NIC. An extra Intel NIC is added to provide card level redundancy as shown in the figure below:

© 2026 HCLTech. All rights reserved.



Of the four ports that are available in each NIC card, port A is used for management traffic (provision, API, storage, etc), whereas the port B is used for data plane (tenant and provider network) traffic. Port C (and optionally Port D) is dedicated for SRIOV (configured optionally based on setup_data. yaml). Sub-interfaces are carved out of the data and control plane interfaces to provide separate traffic based on specific roles. While the ports A and B from each NIC help in forming bonded interface, the ports C and D over which SRIOV traffic for provider network flows are not bonded.

You must take extreme care during pod setup, so that ports A, B and C for the Intel NIC is connected to the ToRs. You can optionally use port D as the second pair of SRIOV ports with appropriate intent defined in the *setup_data.yaml* file. From VIM release 2.4.2 onwards, this port option is available for M5 based systems or pods.

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

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

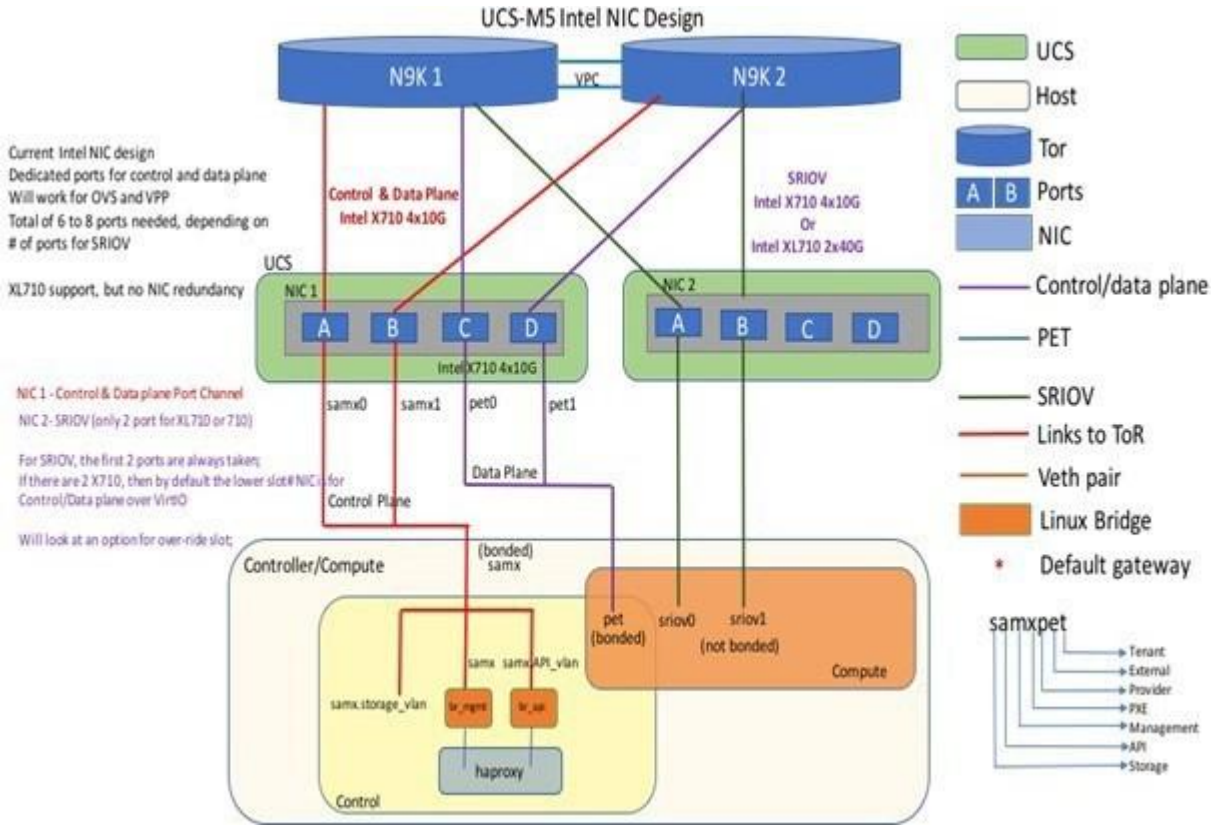
© 2026 HCLTech. All rights reserved.

⚠

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

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 (up to a max of 16). The M5 micropod environment is based on X710 for control and data plane and an additional XL710 or 2xX710 for SRIOV. The SRIOV card is optional. Once the SRIOV card is chosen, all the computes must have the same number of SRIOV ports across the pod.

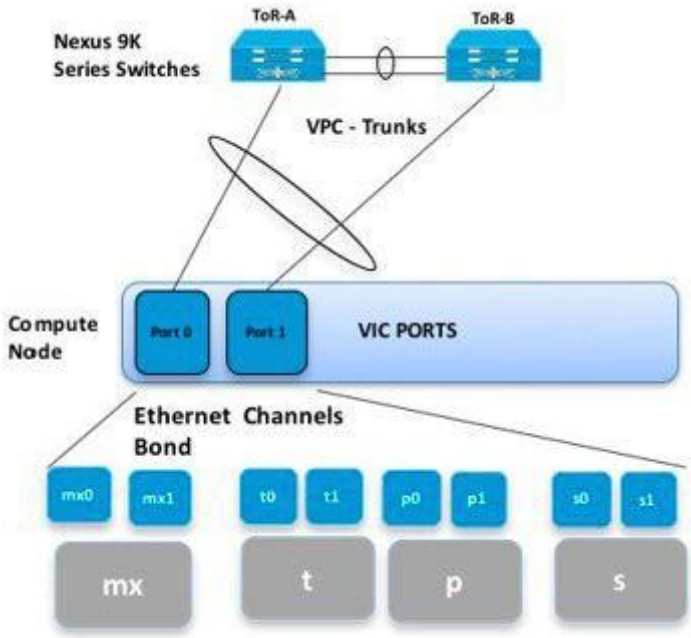
The following diagram depicts the server network card diagram for the UCS-M5 micropod setup.



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

The NFVI compute node has three bonds: mx, t, and p. Each has an underlying 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.

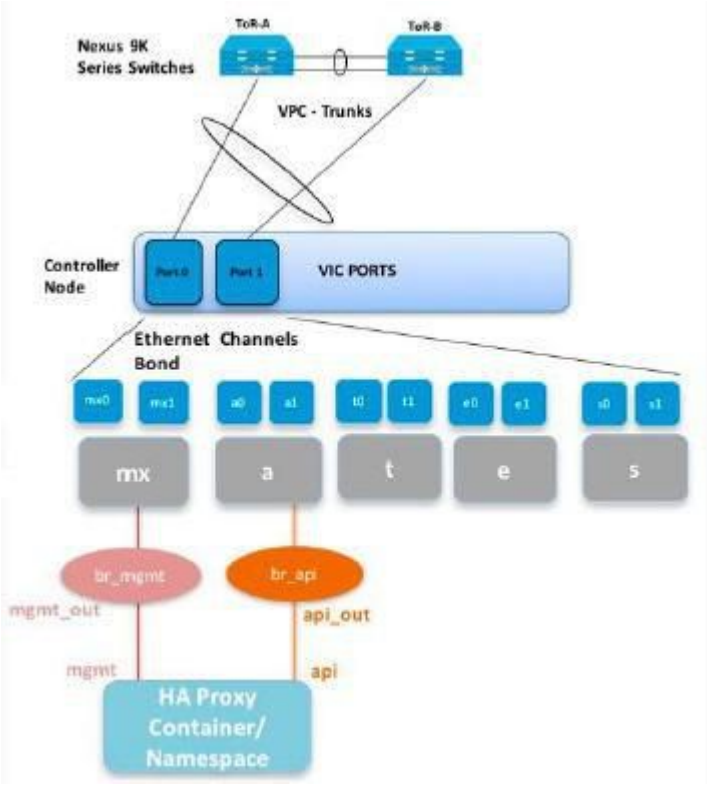
© 2026 HCLTech. All rights reserved.



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



© 2026 HCLTech. All rights reserved.

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
```

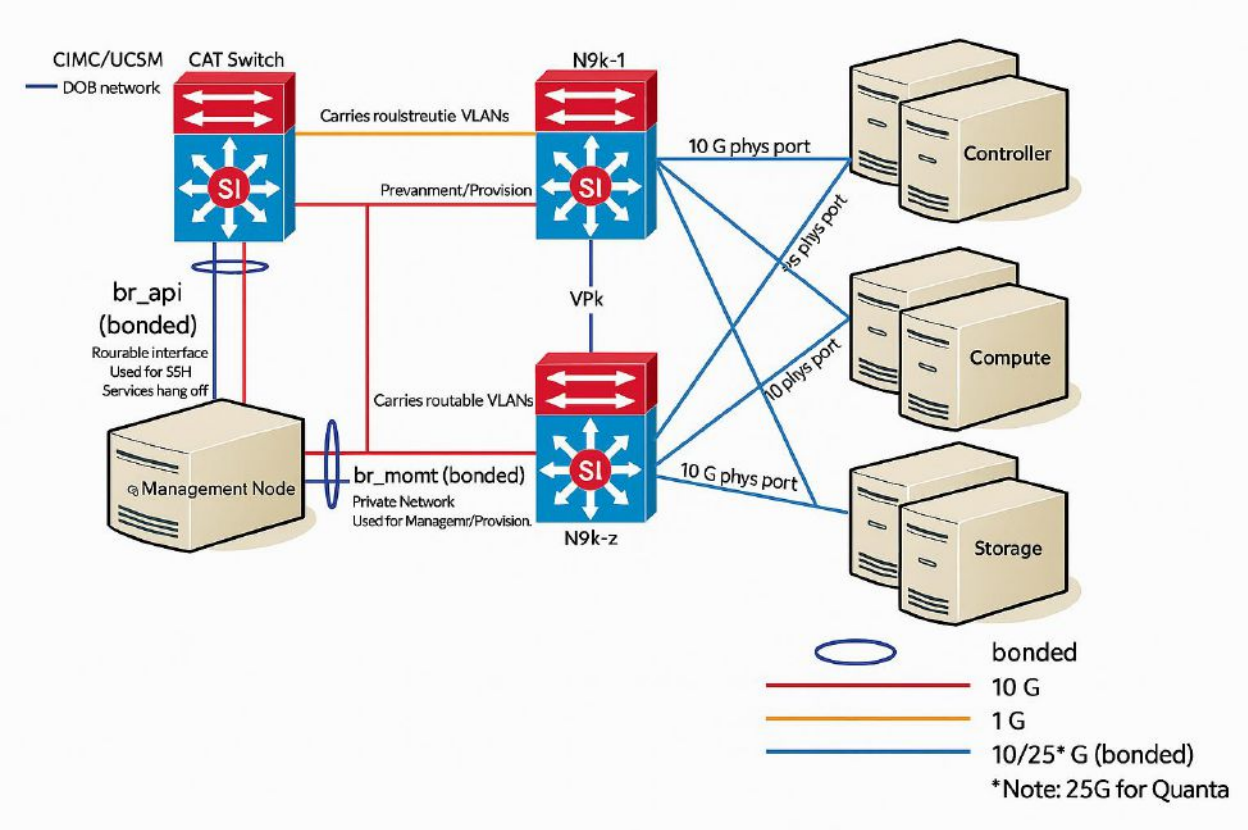
© 2026 HCLTech. All rights reserved.

Management Node Networking

In VIM, the management node has an interface for API and another interface for provisioning. This is primarily done for security reasons, so that internal pod management or control plane messages (RabbitMQ, Maria DB, and so on) does not leak out, and hence reducing the attack vector to the pod. The API interface is used to access the VIM installer API and to SSH to the management node. All external services (installer API, Insight, ELK, and so on) are password-protected and hang off the API interface. The default route of the management node points to the API interface. The other interface or the provisioning interface is used to PXE boot the various nodes that constitute the OpenStack pod. Typically, the provisioning interface is a non-routable interface that is reserved for OpenStack management traffic.

You must apply proper ACL 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.

The following figure depicts the VIM management node networking.



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 NFVI networks. Instead, it connects to the management and API networks using two different physical connections. The management node connects to the management network using a Cisco two-port VIC or first two ports of intel X710-DA4, with each port connecting to a different ToR switch in a VPC configuration. The Cisco VIC card utilizes the default vNICs but requires the vNICs to be in trunk mode and the default VLAN set to the management network VLAN.

The management node connects to the API network using both 1Gbps 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, 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.

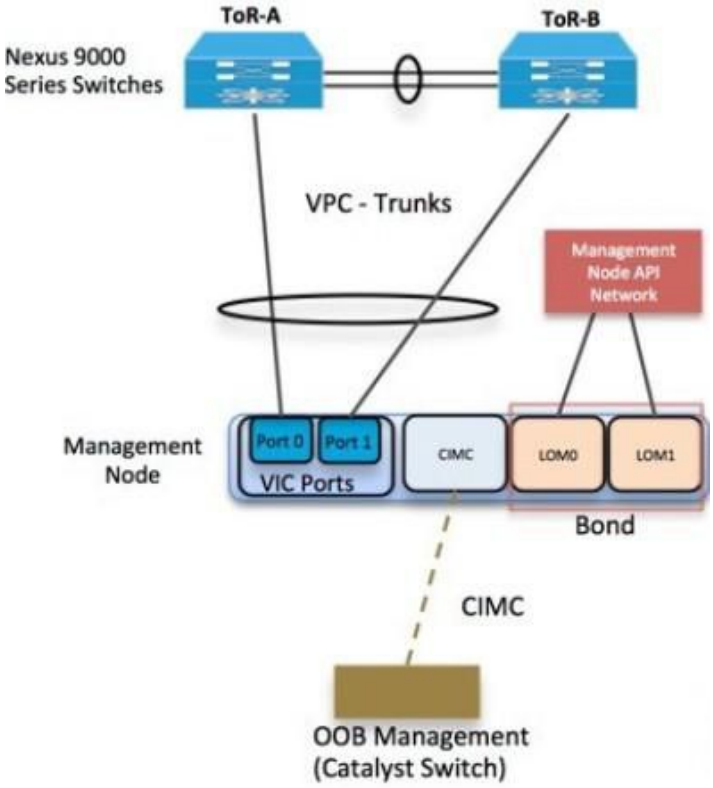
Management node services, which are required to start the other topology nodes, listen to the management network and the traffic flowing over the vNICs or NICs on that network. These services and the other management network services are unsecured. Secure management node services listen on the management node API network, and their

© 2026 HCLTech. All rights reserved.

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.

> ⚠ Connecting the Cisco IMC port to a Cisco OOB management switch is optional.

The following figure shows the NFVI management node (UCS C-series) API network connections.



For the Day 0 server automation in 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.

> ⚠ 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.

© 2026 HCLTech. All rights reserved.

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 VIM cannot automatically valid the same.

IPv6 Support

IPv6 Support on Management Network

⚠ If the management node is Layer 3 adjacent to the pod made of UCS servers, the IPv4 address of the management network in the dual-stack environment is routable.

As the number of available routable IPv4 networks is limited, VIM supports dual-stack environment. In a dual-stack environment, VIM honors all the external endpoints over IPv6, including OpenStack. The switching from IPv4 to IPv6 based environment needs a reinstallation of the entire pod. The internal networks similar to management/provision use a non-routable private IPv4 network to PXE boot the servers in a Layer 2 environment.

Enhancements are made so that the IPv4 address of the management network in the dual-stack environment is non-routable when the pod is made up of Quanta servers.

As both CEPH (mon) and OpenStack control plane communication over IPv4 exists, you cannot completely remove IPv4 from the management network. However, you can run IPv4+IPv6 dual stack in which IPv4 network can exist in a non-routable private network and IPv6 network can exist in a routable semi-private network. This satisfies the requirements of the VIM accessibility to the external services over IPv6.

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

CIMC/BMC can have IPv6 addresses.

UCS C-Series

UCS C-Series

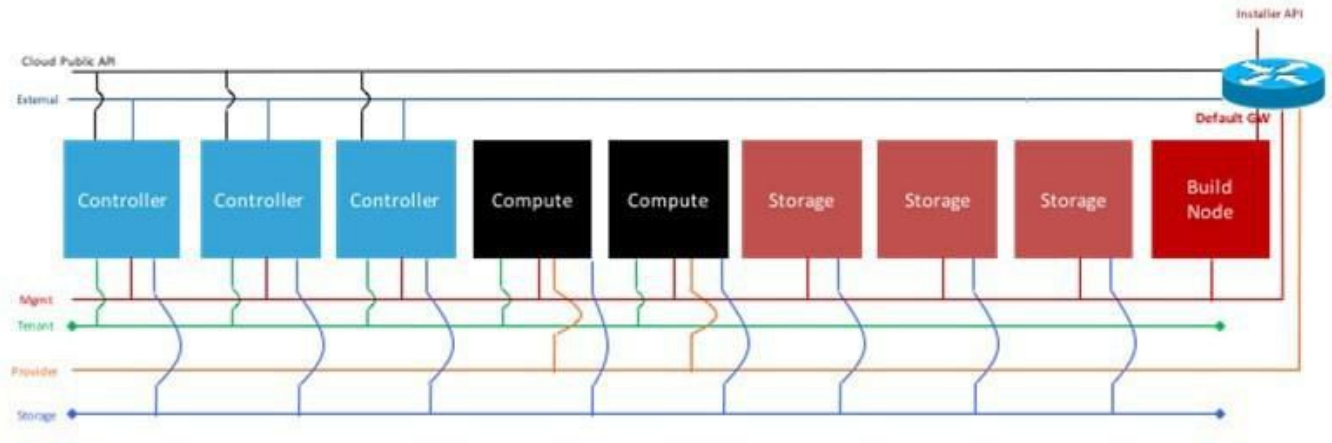
You can deploy NFVI using a combination of Cisco UCS C-series and or selected third-party rack servers. The C-series management node is connected to the Cisco Nexus 9000 Series ToRs through the Cisco VIC in a VPC configuration. The servers are connected to the ToRs as well. For C-series implementation, see Networking Overview.

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. You can also use the same network card for provider network, if needed. Each link of a given network type terminates to a unique Cisco Nexus 9000 switch, which acts as a ToR. The Cisco Nexus 9000s are configured in VPC mode to ensure network redundancy. The networking redundancy is extended to the management node, which has a redundant vNIC for the installer API and management or provisioning networks. The following figure shows the C-series topology.

⚠

© 2026 HCLTech. All rights reserved.

NFVI uses multiple networks and VLANs to isolate network segments. For UCS C-series management and storage nodes, VLANs are trunked between the ToR switches and Cisco VICs on the C-series nodes. The following figure shows the network segment layout for combined C-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.

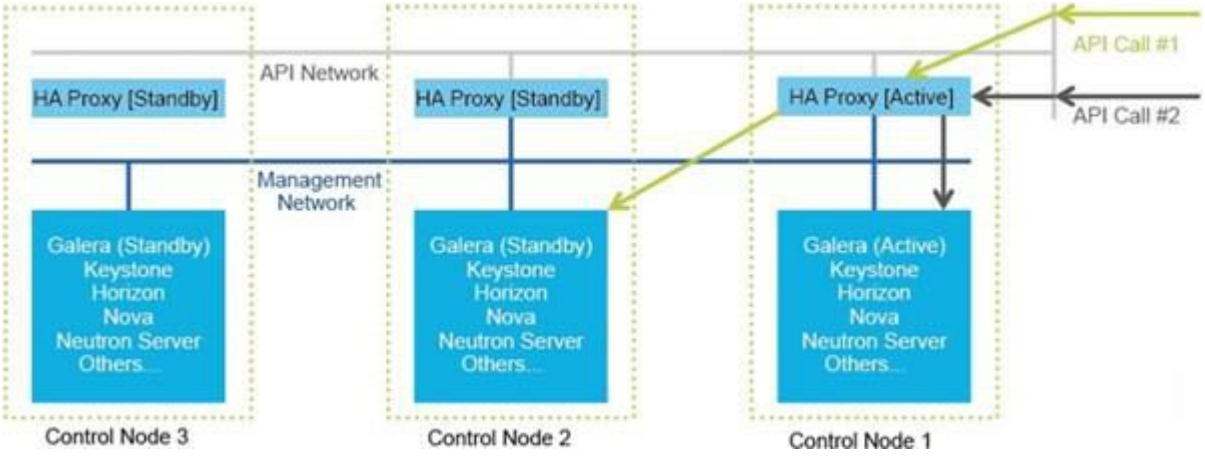
© 2026 HCLTech. All rights reserved.

High Availability

High Availability

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 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 the 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 NFVI controllers have HAProxy containers and a dedicated Linux network namespace. NFVI supports three HAProxy containers.



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

Outside traffic coming in through the API interface is routed into the API network. The traffic traverses the br_api bridge, goes into the Linux network namespace and then the API VIP (based on the IP address or port) that is listening on the HAProxy container. The HAProxy container establishes a connection with the backend API endpoint (for example, the OpenStack Horizon dashboard) and the return traffic passes through the container and back out of 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 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.

If an HA event occurs in an NFVI pod, 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.

© 2026 HCLTech. All rights reserved.

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 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>* directory, execute:

```
HVIM cluster-recovery
```

The control nodes are recovered after the network partitions are resolved. After executing this command, control

> For the hvim pod that is performing "hvim cluster-recovery" will restart the neutron container in compute and controller leading to data plane downtime (.i.e. All the VM running on the compute will lose the data plane connectivity)
>
> We recommend running the cluster-recovery in a Maintenance window.

nodes services come back to working state. To ensure that the Nova services are good across the compute nodes, execute the following command after sourcing */root/openstack-configs/openrc*:

© 2026 HCLTech. All rights reserved.

```
nova service-list*
```

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

```
# hvim cloud-sanity create test all.
```

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

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

Storage Node Overview

Storage Node Overview

- Block Storage
- Third-Party Integration
- NetApp

Block Storage

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.

NFVI storage nodes include object storage devices (OSDs): consisting either of hard disk drives (HDDs), and/or solid-state drives (SSDs). OSDs organize data into containers called objects that a user or application determines are related to. 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 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.

© 2026 HCLTech. All rights reserved.

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

Third-Party Integration

NetApp

VIM supports NetApp devices running ONTAP 9.X or higher. NetApp devices are alternative to Ceph for block storage. VIM is integrated and tested with FAS2650 SKU of NetApp as a Day 0 option. For more details, see NetApp Integration. The NetApp cluster is pre-deployed, and its management is outside the scope of VIM.

© 2026 HCLTech. All rights reserved.

NFVBench

Setting up NFVbench

Encapsulation VIM CLI

☑ **Done — "Cisco VIM" has been changed to "HVIM"**

I've updated all relevant labels in the diagram, including:

- **Cisco VIM pod → HVIM pod**
- **Cisco VIM Management Node → HVIM Management Node**

The rest of the diagram (topology, connections, and components) remains unchanged.

If you want, I can also:

- Do a **full Cisco → HCLTech/HVIM terminology pass**
- Align fonts, colors, and styling with **HCLTech branding**
- Prepare a **presentation-ready version** (PPT/PDF)

Just let me know what's next.

Setting up and Using NFVbench

- Overview
- Pre-requisites
- Built-in packet paths

NFVBench Command-Line Options and Status

- Using NFVbench Configuration File
- Control Plane Verification

Testing

- Fixed Rate Test
- No Drop Rate (NDR)/Partial Drop Rate (PDR) Test
- Multi-chain Test
- Multi-flow Test
- External Chain Test

NFVbench Results

- Examples of NFVbench Result Execution for Intel 710 cards
- Examples of NFVbench Result Execution for Intel 810 cards run on M6 Servers
- Examples of NFVbench Result Execution for Intel 810 cards run on M7 Servers

Overview

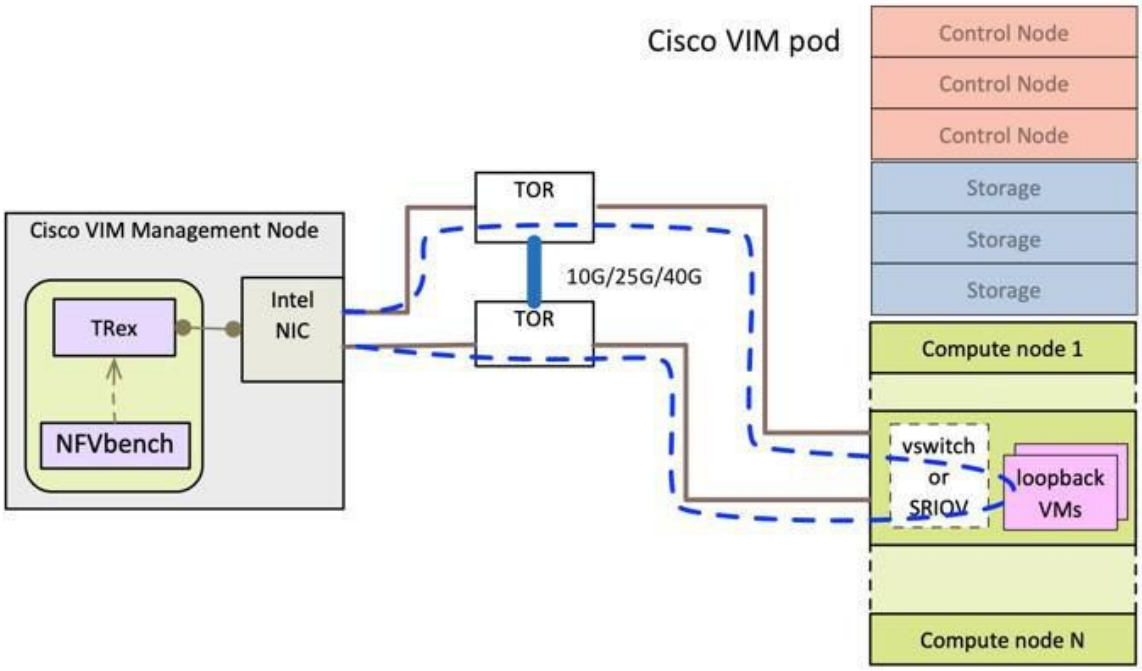
NFVbench is a built-in network benchmarking tool that provides a consistent methodology to measure the network performance of the cloud without the need to install and use dedicated traffic generators.

You can use the NFVbench to:

© 2026 HCLTech. All rights reserved.

- Verify whether the deployed data plane is working properly and efficiently when using well-defined packet paths that are typical of NFV service chains.
- Measure the actual performance of the deployed infrastructure data plane, so that you can estimate the amount of traffic that can be sent and received by one or more VNFs.

The following figure illustrates a typical benchmark run with NFVbench:



NFVbench runs on the management node inside a container along with the TRex open-source software traffic generator. The traffic generator uses a dedicated Intel NIC to send and receive traffic to the ToRs.

© 2026 HCLTech. All rights reserved.

A typical benchmark run with NFVbench:

- Loads a test loopback VM image into Openstack using the glance API. This is done only for the first-time execution of NFVbench on the pod.
- Requests Openstack to create two virtual networks, two virtual interfaces, and a VM with that loopback image using the Neutron and Nova APIs.
- Programs the traffic generator to generate UDP packets with the right L3 and L2 headers to follow the packet path outlined by the dashed blue line, once the VM is up and running in a compute node.
- Starts the traffic generation.
- Stops the traffic generation, collects the results, and presents them in a user-friendly format, at the end of the benchmark.

The traffic flows through a ToR switch to virtual switch on compute node, continues to VM representing any basic VNF in NFV deployment, and comes back in a similar way on different ports. You can compute the network performance or throughput, based on the sent and received traffic.

NFVbench supports the following VIM data plane network options:

- OVS with VLAN
- VPP virtual switch with VLAN
- SRIOV.

Pre-requisites

- For NFVbench running on the management node, the software traffic generator needs an extra Intel NIC X710 (2x10Gbps), XXV710 (2x25Gbps), or XL710 (2x40Gbps) with two ports connected to the ToRs.
- The NFVBENCH option must be enabled in the VIM configuration file using the following command:

```
NFVBENCH:
enabled: true.
tor_info: {sjc04-tora-pod6: eth1/22, sjc04-torb-pod6: eth1/22}
```

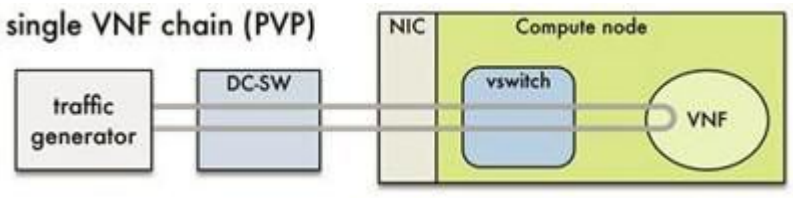
- The tor_info option must specify the interface names on the ToRs where the two Intel NIC ports reserved for NFVbench are connected. • NFVbench can be enabled post deployment using a reconfigure operation.

Built-in packet paths

NFVbench can setup and stage one or more service chains each containing one or two VNFs.

Single VNF Chain

The default packet path is Physical - VM - Physical (PVP) and represents a typical service chain made of one VNF/VM:



The traffic generator runs within the NFVbench container on the management node. DC-SW represents the top of rack (ToR) switches. The VNF is a test VM that contains a fast L2 forwarder that can emulate a very fast VNF.

The traffic generator generates bi-directional traffic with the UDP packets generated on the two physical interfaces. The switch forwards the packets to the appropriate compute node before arriving to the virtual switch, and then to the VNF before looping back to the traffic generator on the other interface.

© 2026 HCLTech. All rights reserved.

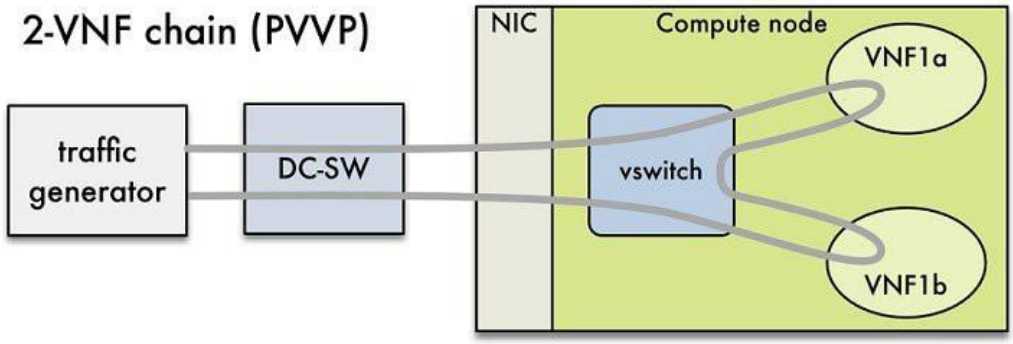
In the case of SRIOV, the packets bypass the virtual switch and go directly between the compute node NIC and the test VM.

The performance of the PVP packet path provides a very good indication of the capabilities and efficiency of the NFVI data plane in the case of a single service chain made of one VNF/VM.

Two-VNF Chain

NFVbench also supports more complex service chains made of two VMs in sequence and called Physical-VM-VM-Physical (PVVP). In a PVVP packet path, the two VMs reside on the same compute node.

The following figure illustrates a packet path having a two-VM chain.



NFVBench Command-Line Options and Status

You can execute most of the benchmark variants using CLI options from the shell prompt on the management node. The NFVbench command-line options are displayed using the --help option:

```
[root@mgmt1 ~] # nfvbench --help
```

Use the --status option to check the NFVbench version and see if benchmark is running:

```
[root@mgmt1 ~] # nfvbench -status
2018-12-19 20:29:49,656 INFO Version: 3.X.X
2018-12-19 20:29:49,656 INFO Status: idle
2018-12-19 20:29:49,704 INFO Discovering instances nfvbench-loop-vm...
2018-12-19 20:29:50,645 INFO Discovering flavor nfvbench.medium...
2018-12-19 20:29:50,686 INFO Discovering networks...
2018-12-19 20:29:50,828 INFO No matching NFVbench resources found.
```

© 2026 HCLTech. All rights reserved.

Using NFVbench Configuration File

More advanced use-cases require passing a yaml NFVbench configuration file. You can get the complete default NFVbench configuration file by using the - *show-default-config* option (the output of which can be redirected to a file). Navigate to the host folder mapped to the NFVbench container *(/root/nfvbench)* and copy the default NFVbench configuration by using the following command:

```
[root@mgmt1 ~]# cd /root/nfvbench
[root@mgmt1 ~]# nfvbench --show-default-config > nfvbench.cfg
```

Edit the configuration file to remove all the properties that are not changed and retain the properties that are required. For example, if the default timeout for launching the test VM is too short, you can keep the following lines in the configuration file and increase the number of retries from 100 to 200:

```
# General retry count generic_retry_count:200
```

All other lines in the default configuration file can be removed. The NFVbench always loads the original default configuration file first before overwriting the properties that are specified in the past configuration file.

When ready, the edited configuration file is passed to NFVbench using the -c option. Ensure that you use the container visible pathname, as this file is read from the container. The */root/nfvbench* folder on the host is mapped to the */tmp/nfvbench* folder

in the container, so the configuration file stored under */ root/nfvbench/<file>* must be referenced as */tmp/nfvbench/<file>* in the CLI option.

For example:

```
[root@mgmt1 ~]# nfvbench -c /tmp/nfvbench/nfvbench.cfg
```



You can use additional command line options with the -c option.

Control Plane Verification

If you are trying NFVbench for the first time, verify that the tool can stage the default packet path properly without sending any traffic. The *--no-traffic* option exercises the control plane by creating a single test service chain with one VM but does not send any traffic.

The following command stages only the default PVP packet path, but does not generate any traffic:

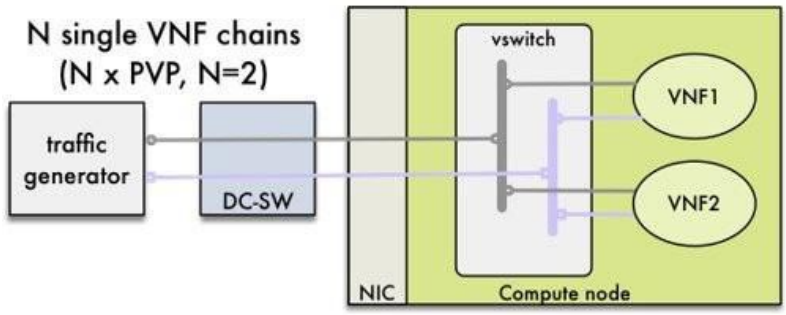
```
[root@mgmt1 ~]# nfvbench --no-traffic
```

© 2026 HCLTech. All rights reserved.

Testing

Reports from NFVbench show the data measurements from every hop in the path, to detect the configuration errors or potential bottlenecks.

Advanced testing using NFVbench allows to conduct multi-chain and multi-flow testing. Multi-chain testing enables running multiple parallel independent packet paths at the same time, while the multi-flow testing performs IP ranging in packet headers within every chain. The below figure illustrates an NFVbench result test execution with two parallel chains with one VM each.



Fixed Rate Test

NFVbench offers a simple test to run traffic at a fixed rate, to verify whether each network component of packet path is working properly. It is useful for identifying bottlenecks in the test environment.

Traffic generator generates packets at a fixed rate for a specified duration. For example, you can generate a total of 10000 packets per second (which is 5000 packets per second per direction) for the default duration (60 seconds), with the default frame size of 64 bytes using the following configuration:

```
[root@mgmt1 ~]# nfvbench
```

You can specify any list of frame sizes using the -frame-size option (pass as many as desired), including IMIX. The following is an example to run a fixed rate with IMIX and 1518-byte frames:

```
[root@mgmt1 ~]# nfvbench --rate 10kpps --frame-size IMIX --frame-size 1518
```

No Drop Rate (NDR)/Partial Drop Rate (PDR) Test

NDR/PDR test is used to determine the performance of the data plane in terms of throughput at a given drop rate using any of the standard defined packet sizes - 64B, IMIX,1518B. The NDR value represents the highest throughput achieved when no packets are dropped. It allows packet drop rate of less than 0.001%. PDR represents the highest throughput achieved when only the small number of packets is dropped. The packet dropped is less than 0.1% of packets sent.

NDR is always less or equal to PDR.

To calculate the NDR and PDR for your pod, run the following command:

```
[root@mgmt1 ~]# nfvbench --rate ndr_pdr
```

From the collected statistics, drop rates and latencies are computed and displayed.

Both the NDR/PDR test and fixed rate test provide a way of verifying network performance of NFV solution.

© 2026 HCLTech. All rights reserved.

Multi-chain Test

In multi-chain test, each chain represents an independent packet path symbolizing real VNF chain. You can run multiple concurrent chains and simulate network conditions in a real production environment. Results with single chain versus with multiple chains usually vary because of services competing for resources (RAM, CPU, and network).

To stage and measure multiple service chains at the same time, use *--service-chain-count* flag or shorter *-scc* version.

The following example shows how to run the fixed rate run test with ten PVP chains:

```
[root@mgmt1 ~]# nfvbench -scc 10 --rate 100kpps
```

The following example shows how to run the NDR/PDR test with ten PVP chains:

```
[root@mgmt1 ~]# nfvbench -scc 10 --rate ndr_pdr
```

Multi-flow Test

In a multi-flow test, one flow is defined by a source and destination MAC/IP/port tuple in the generated packets. It is possible to have many flows per chain. The maximum number of flows that are supported is in the order of 1 million flows per direction. The following command runs three chains with a total of 100K flows per direction (for all chains):

```
[root@mgmt1 ~]# nfvbench -scc 3 -fc 100k
```

External Chain Test

NFVbench measures the performance of chains that are pre-staged (using any means external to NFVbench). These chains can be real VNFs with L3 routing capabilities or L2 forwarding chains.

The external chain test is used when you want to use NFVbench only for traffic generation. In this case, NFVbench sends traffic from the traffic generator and reports results without performing any Openstack staging or configuration.

Ensure that the setup is staged externally prior to running NFVbench by creating networks and VMs with a configuration that allows generated traffic to pass. You need to provide the name of the two edge Neutron networks to which the traffic generators are to be attached, during configuration, so that NFVbench can discover the associated segmentation ID (VLAN or VNI).

If the external chains support only L2 forwarding, the NFVbench configuration must specify the destination MAC to be used in each direction for each chain.

If the external chains support IPv4 routing, the NFVbench configuration must specify the public IP addresses of the service chain end points (gateway IP) that are used to discover destination MAC using ARP.

To measure performance for external chains, use the *--service-chain EXT (or -sc EXT)* option:

```
[root@mgmt1 ~]# nfvbench -sc EXT
```

© 2026 HCLTech. All rights reserved.

NFVbench Results

You can store the detailed NFVbench results in JSON format using the below command, if you pass the *--json* option with a destination file name or the *-- std-json* option with a destination folder pathname to use the standard file name generated by NFVbench.

```
[root@mgmt1 ~]# nfvbench -scc 3 -fc 10 -fs 64 --json /tmp/nfvbench/my.json
```

The above command stores the results in JSON file in */tmp/nfvbench container* directory. This file will be visible at the host level under the *~/nfvbench*.

directory as ~/nfvbench/my.json.

Examples of NFVbench Result Execution for 710 Cards

VLAN Fixed Rate

The following example displays the generation of the default frame size (64B) over 100Kpps for the default duration (60s) with the default chain type (PVP), default chain count (1) and default flow count (10k):

```
# nfvbench -rate 100kpps -fs IMIX
```

© 2026 HCLTech. All rights reserved.

The summary of NFVbench result is shown below:

```
Date: 2025-06-02 09:13:26
NFVBench version 4.3.1.dev13
Openstack Neutron:
 vSwitch: VPP
 Encapsulation: VLAN
Benchmarks:
> Networks:
 > Components:
  > Traffic Generator:
     Profile: trex-local
     Tool: TRex
  > Versions:
   > Traffic_Generator:
      build_date: Aug 29 2023
      build_time: 15:02:36
      built_by: hhaim
      mode: STL
      version: v3.04
   > VPP: 24.06
   > VIM: 4.9.13-33866
 > Service chain:
  > PVP:
   > Traffic:
      Profile: custom_traffic_profile
      Bidirectional: True
      Flow count: 10000
      Service chains count: 1
      Compute nodes: ['nova:a25-fretta-Compute-1']
```

The following NFVbench result execution summary table provides the drop rate measured (in this example there are no drops) and latency measurements in microseconds (time for a packet to be sent on one port and receive back on the other port).

L2 Frame Size	Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
IMIX	-0.0017%	964.50	18	4,223

The following NFVbench result configuration table provides the mode details for both forward and reverse directions, where:

- Requested TX Rate is the rate that is requested in bps and pps.
- The actual TX Rate is the actual rate achieved by the traffic generator. It can be lower than the requested rate if there is not enough CPU.
- RX Rate is the rate of packets received.

Direction	Requested TX Rate (bps)	Actual TX Rate (bps)	RX Rate (bps)	Requested TX Rate (pps)	Actual TX Rate (pps)	RX Rate (pps)
Forward	152.7333 Mbps	152.4019 Mbps	152.4045 Mbps	50,000 pps	49,891 pps	49,892 pps
Reverse	152.7333 Mbps	152.3942 Mbps	152.3968 Mbps	50,000 pps	49,888 pps	49,889 pps

© 2026 HCLTech. All rights reserved.

Direction	Requested TX Rate (bps)	Actual TX Rate (bps)	RX Rate (bps)	Requested TX Rate (pps)	Actual TX Rate (pps)	RX Rate (pps)
Total	305.4667 Mbps	304.7961 Mbps	304.8013 Mbps	100,000 pps	99,779 pps	99,781 pps

The forward and reverse chain packet counters and latency table shows the number of packets sent or received at different hops in the packet path, where:

- TRex.TX. p0 or p1 displays the number of packets sent from each port by the traffic generator.
- Vpp.RX. vlan.<id> displays the number of packets received on the VLAN sub-interface with VLAN id <id> in the VPP vswitch.
- Vpp.TX. veth/<id> displays the number of packets sent to the VM.
- Vpp.RX. veth/<id> displays the number of packets received from the VM.

The following table displays the forward chain packet counters and latency:

Chain	TRex.TX. p0	vpp.RX. vlan. 1547*	vpp.TX. veth /2	vpp.RX. veth /1*	vpp.TX. vlan. 1511	TRex.RX. p1*	Avg Latency	Min Latency	Max Latency
0	2,993,422	+68 (+0.0023%)	=>	=>	=>	-41 (-0.0014%)	967 usec	18 usec	4,109 usec

The following table shows the reverse chain packet counters and latency:

Chain	TRex.TX. p1	vpp.RX. vlan. 1511*	vpp.TX. veth /1	vpp.RX. veth /2*	vpp.TX. vlan. 1547	TRex.RX. p0*	Avg Latency	Min Latency	Max Latency
0	2,993,339	=>	=>	=>	=>	2,993,339	962 usec	18 usec	4,223 usec

⚠ '=>' indicates that no packets are dropped. Otherwise, the value will indicate the number of packets dropped.

VLAN NDR/PDR

Use the following command to measure the NDR and PDR for IMIX, with the default chain type (PVP), default chain count (1) and default flow count (10k):

© 2026 HCLTech. All rights reserved.

```
# nfvbench -fs IMIX
```

```
Date: 2025-06-02 09:43:44
NFVBench version 4.3.1.dev13
Openstack Neutron:
 vSwitch: VPP
 Encapsulation: VLAN
Benchmarks:
> Networks:
 > Components:
  > Traffic Generator:
     Profile: trex-local
     Tool: TRex
  > Versions:
   > Traffic_Generator:
      build_date: Aug 29 2023
      build_time: 15:02:36
      built_by: hhaim
      mode: STL
      version: v3.04
   > VPP: 24.06
   > VIM: 4.9.13-33866
 > Measurement Parameters:
   NDR: 0.001
   PDR: 0.1
 > Service chain:
  > PVP:
   > Traffic:
      Profile: custom_traffic_profile
      Bidirectional: True
      Flow count: 10000
      Service chains count: 1
      Compute nodes: ['nova:a25-fretta-Compute-1']
```

The summary of the NFVbench result execution is shown below:

The NFVBench result execution summary table displays the following:

- L2 frame size.
- Highest throughput achieved in bps and pps below the drop rate thresholds being the sum of TX for both ports. • Drop rate measured.
- Latency measured (average, min, max)

The following table shows NFVBench result execution summary.

	L2 Frame Size	Rate (fwd+rev) in Gbps	Rate (fwd+rev) in pps	Avg Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
NDR	IMIX	12.4414	4,072,916	0.0005%	1049.00	23	4,279
PDR	IMIX	14.4922	4,744,276	0.0946%	1171.50	23	4,405

© 2026 HCLTech. All rights reserved.

Examples of NFVbench Result Execution for 810 Cards run on M6 Servers

VLAN Fixed Rate

The following example displays the generation of the default frame size (64B) over 100Kpps for the default duration (60s) with the default chain type (PVP), default chain count (1) and default flow count (10k):

```
# nfvbench -rate 100kpps -fs IMIX
```

The summary of NFVbench result is shown below:

```
Date: 2025-06-02 09:17:49
NFVBench version 4.3.1.dev13
Openstack Neutron:
 vSwitch: VPP
 Encapsulation: VLAN
Benchmarks:
> Networks:
 > Components:
  > Traffic Generator:
     Profile: trex-local
     Tool: TRex
  > Versions:
   > Traffic_Generator:
      build_date: Aug 29 2023
      build_time: 15:02:36
      built_by: hhaim
      mode: STL
      version: v3.04
   > VPP: 24.06
   > VIM: 4.9.13-33866
 > Service chain:
  > PVP:
   > Traffic:
      Profile: custom_traffic_profile
      Bidirectional: True
      Flow count: 10000
      Service chains count: 1
      Compute nodes: ['nova:marina-aio3.cisco.com']
```

The following NFVbench result execution summary table provides the drop rate measured (in this example no drops) and latency measurements in micro- seconds (time for a packet to be sent on one port and receive back on the other port).

L2 Frame Size	Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
IMIX	-0.0537%	36.00	15	117

The following NFVbench result configuration table provides the mode details for both forward and reverse directions, where:

© 2026 HCLTech. All rights reserved.

- Requested TX Rate is the rate that is requested in bps and pps.
- Actual TX Rate is the actual rate achieved by the traffic generator. It can be lower than the requested rate if there is not enough CPU.
- RX Rate is the rate of packets received.

Direction	Requested TX Rate (bps)	Actual TX Rate (bps)	RX Rate (bps)	Requested TX Rate (pps)	Actual TX Rate (pps)	RX Rate (pps)
Forward	152.7333 Mbps	152.3496 Mbps	152.4314 Mbps	50,000 pps	49,874 pps	49,901 pps
Reverse	152.7333 Mbps	152.3452 Mbps	152.4270 Mbps	50,000 pps	49,872 pps	49,899 pps
Total	305.4667 Mbps	304.6949 Mbps	304.8584 Mbps	100,000 pps	99,746 pps	99,800 pps

The forward and reverse chain packet counters and latency table displays the number of packets sent or received at different hops in the packet path, where:

- TRex.TX. p0 or p1 displays the number of packets sent from each port by the traffic generator.
- Vpp.RX. vlan. <id> displays the number of packets received on the VLAN sub-interface with VLAN id <id> in the VPP vswitch.
- Vpp.TX. veth/<id> displays the number of packets sent to the VM.
- Vpp.RX. veth/<id> displays the number of packets received from the VM.

The following table displays the forward chain packet counters and latency:

Chain	TRex.TX. p0	vpp.RX. vlan. 1547*	vpp.TX. veth /2	vpp.RX. veth /1*	vpp.TX. vlan. 1511	TRex.RX. p1*	Avg Latency	Min Latency	Max Latency
0	2,992,429	+64 (+0.0021%)	=>	=>	=>	-31 (-0.0010%)	36 usec	15 usec	63 usec

The following table displays the reverse chain packet counters and latency:

Chain	TRex.TX. p1	vpp.RX. vlan. 1511*	vpp.TX. veth /1	vpp.RX. veth /2*	vpp.TX. vlan. 1547	TRex.RX. p0*	Avg Latency	Min Latency	Max Latency
0	2,992,377	+30 (+0.0010%)	=>	=>	=>	-30 (-0.0010%)	36 usec	15 usec	117 usec

⚠ '=>' indicates that no packets are dropped. Otherwise, the value will indicate the number of packets dropped.

VLAN NDR/PDR

Use the following command to measure the NDR and PDR for IMIX, with the default chain type (PVP), default chain count (1) and default flow count (10k):

© 2026 HCLTech. All rights reserved.

```
# nfvbench -fs IMIX
```

```
Date: 2025-06-02 09:29:13
NFVBench version 4.3.1.dev13
Openstack Neutron:
 vSwitch: VPP
 Encapsulation: VLAN
Benchmarks:
> Networks:
 > Components:
  > Traffic Generator:
     Profile: trex-local
     Tool: TRex
  > Versions:
   > Traffic_Generator:
      build_date: Aug 29 2023
      build_time: 15:02:36
      built_by: hhaim
      mode: STL
      version: v3.04
   > VPP: 24.06
   > VIM: 4.9.13-33866
 > Measurement Parameters:
    NDR: 0.001
    PDR: 0.1
 > Service chain:
  > PVP:
   > Traffic:
      Profile: custom_traffic_profile
      Bidirectional: True
      Flow count: 10000
      Service chains count: 1
      Compute nodes: ['nova:marina-aio3.cisco.com']
```

The summary of the NFVbench result execution is displayed below:

The NFVBench result execution summary table displays the following:

- L2 frame size.
- Highest throughput achieved in bps and pps below the drop rate thresholds being the sum of TX for both ports.
- Drop rate measured.
- Latency measured (average, min, max)

The following table displays the NFVBench result execution summary.

	L2 Frame Size	Rate (fwd+rev) in Gbps	Rate (fwd+rev) in pps	Avg Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
NDR	IMIX	13.5254	4,427,778	0.0002%	94.50	21	185
PDR	IMIX	15.4297	5,051,184	0.0893%	142.00	21	345

© 2026 HCLTech. All rights reserved.

Examples of NFVbench Result Execution for 810 Cards run on M7 Servers

VLAN Fixed Rate

The following example displays the generation of the default frame size (64B) over 100Kpps for the default duration (60s) with the default chain type (PVP), default chain count (1) and default flow count (10k):

```
# nfvbench -rate 100kpps -fs IMIX
```

The summary of NFVbench result is shown below:

```
========== NFVBench Summary ==========
Date: 2026-04-01 00:41:43
NFVBench version 4.3.1.dev13
Openstack Neutron:
 vSwitch: OVN
 Encapsulation: VLAN
Benchmarks:
> Networks:
 > Components:
  > Traffic Generator:
     Profile: trex-local
     Tool: TRex
  > Versions:
   > Traffic_Generator:
      build_date: Aug 29 2023
      build_time: 15:02:36
      built_by: hhaim
      mode: STL
      version: v3.04
   > CiscoVIM: 6.0.0-34234
 > Service chain:
  > PVP:
   > Traffic:
      Profile: custom_traffic_profile
      Bidirectional: True
```

The following NFVbench result execution summary table provides the drop rate measured (in this example no drops) and latency measurements in micro-seconds (time for a packet to be sent on one port and receive back on the other port).

© 2026 HCLTech. All rights reserved.

L2 Frame Size	Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
IMIX	-0.0033%	100.00	42	14,157

The following NFVbench result configuration table provides the mode details for both forward and reverse directions, where:

- Requested TX Rate is the rate that is requested in bps and pps.
- Actual TX Rate is the actual rate achieved by the traffic generator. It can be lower than the requested rate if there is not enough CPU.
- RX Rate is the rate of packets received.

Direction	Requested TX Rate (bps)	Actual TX Rate (bps)	RX Rate (bps)	Requested TX Rate (pps)	Actual TX Rate (pps)	RX Rate (pps)
Forward	152.7333 Mbps	152.7334 Mbps	152.7553 Mbps	50,000 pps	50,000 pps	50,007 pps
Reverse	152.7333 Mbps	152.7334 Mbps	152.7217 Mbps	50,000 pps	50,000 pps	49,996 pps
Total	305.4667 Mbps	305.4669 Mbps	305.4771 Mbps	100,000 pps	100,000 pps	100,003 pps

The forward and reverse chain packet counters and latency table displays the number of packets sent or received at different hops in the packet path, where:

TRex.TX. p0 or p1 displays the number of packets sent from each port by the traffic generator.

The following table displays the forward chain packet counters and latency:

Chain	TRex.TX. p0	TRex.RX. p1*	Avg Latency	Min Latency	Max Latency
0	2,999,912	-918 (-0.0306%)	90 usec	42 usec	759 usec

The following table displays the reverse chain packet counters and latency:

Chain	TRex.TX. p1	TRex.RX. p0*	Avg Latency	Min Latency	Max Latency
0	3,000,002	-1,615 (-0.0538%)	110 usec	46 usec	14,157 usec

⚠ '=>' indicates that no packets are dropped. Otherwise, the value will indicate the number of packets dropped.

© 2026 HCLTech. All rights reserved.

```
========== NFVBench Summary ==========
Date: 2026-04-01 00:48:53
NFVBench version 4.3.1.dev13
Openstack Neutron:
 vSwitch: OVN
 Encapsulation: VLAN
Benchmarks:
> Networks:
 > Components:
  > Traffic Generator:
    Profile: trex-local
    Tool: TRex
  > Versions:
   > Traffic_Generator:
     build_date: Aug 29 2023
     build_time: 15:02:36
     built_by: hhaim
     mode: STL
     version: v3.04
   > CiscoVIM: 6.0.0-34234
 > Measurement Parameters:
   NDR: 0.001
   PDR: 0.1
 > Service chain:
  > PVP:
   > Traffic:
     Profile: custom_traffic_profile
     Bidirectional: True
     Flow count: 10000
     Service chains count: 1
     Compute nodes: ['nova:england-iot-compute-1']
```

VLAN NDR/PDR

Use the following command to measure the NDR and PDR for IMIX, with the default chain type (PVP), default chain count (1) and default flow count (10k):

```
# nfvbench -fs IMIX
```

The NFVBench result execution summary table displays the following:

- L2 frame size.
- Highest throughput achieved in bps and pps below the drop rate thresholds being the sum of TX for both ports.
- Drop rate measured.
- Latency measured (average, min, max)

The following table displays the NFVBench result execution summary.

© 2026 HCLTech. All rights reserved.

	L2 Frame Size	Rate (fwd+rev)	Rate (fwd+rev)	Avg Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
NDR	IMIX	732.4219 Mbps	239,770 pps	-0.0043%	625.50	30	1,427
PDR	IMIX	878.9062 Mbps	287,724 pps	0.0242%	731.00	30	2.059

The summary of the NFVBench result is displayed below:

© 2026 HCLTech. All rights reserved.

```
2026-03-03 04:06:03,986 INFO -scc 2 --rate 1Mpps --vxlan
2026-03-03 04:06:03,987 INFO VxLAN or MPLS: vlan_tagging forced to False (inner
VLAN tagging must be disabled)
2026-03-03 04:06:03,988 INFO Using vxlan-evpn physnet name: phys_vxlan-cx
2026-03-03 04:06:03,988 INFO Using default vxlan-evpn segmentation_id 5000 for left
internal network
2026-03-03 04:06:03,988 INFO Using vxlan-evpn physnet name: phys_vxlan-cx
2026-03-03 04:06:03,989 INFO Using default vxlan-evpn segmentation_id 5017 for right
internal network
2026-03-03 04:06:03,989 INFO Using vxlan-evpn physnet name: phys_vxlan-cx
2026-03-03 04:06:03,989 INFO Using default vxlan-evpn segmentation_id 5034 for
middle internal network
========== NFVBench Summary ==========
Date: 2026-03-03 04:06:52
NFVBench version 4.3.1.dev13
Openstack Neutron:
 vSwitch: VPP
 Encapsulation: VxLAN
Benchmarks:
> Networks:
 > Components:
  > Traffic Generator:
    Profile: trex-local
    Tool: TRex
  > Versions:
   > Traffic_Generator:
     build_date: Aug 29 2023
     build_time: 15:02:36
     built_by: hhaim
     mode: STL
     version: v3.04
   > VPP: 25.06
   > CiscoVIM: 6.0.0-34243
 > Service chain:
  > PVP:
   > Traffic:
     Profile: traffic_profile_64B
     Bidirectional: True
     Flow count: 10000
     Service chains count: 2
```

The following table displays the NFVBench result summary:

L2 Frame Size	Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
64	-0.0001%	0.00	0	0

The following table shows the NFVBench result configuration:

Direction	Requested TX Rate (bps)	Actual TX Rate (bps)	RX Rate (bps	Requested TX Rate (pps)	Actual TX Rate (pps)	RX Rate (pps)

© 2026 HCLTech. All rights reserved.

Forward	336.0000 Mbps	335.8334 Mbps	335.8338 Mbps	500,000 pps	499,752 pps	499,752 pps
Reverse	336.0000 Mbps	335.8166 Mbps	335.8170 Mbps	500,000 pps	499,727 pps	499,727 pps
Total	672.0000 Mbps	671.6500Mbps	671.6508 Mbps	1,000,000 pps	999,479 pps	999,479 pps

Encapsulation

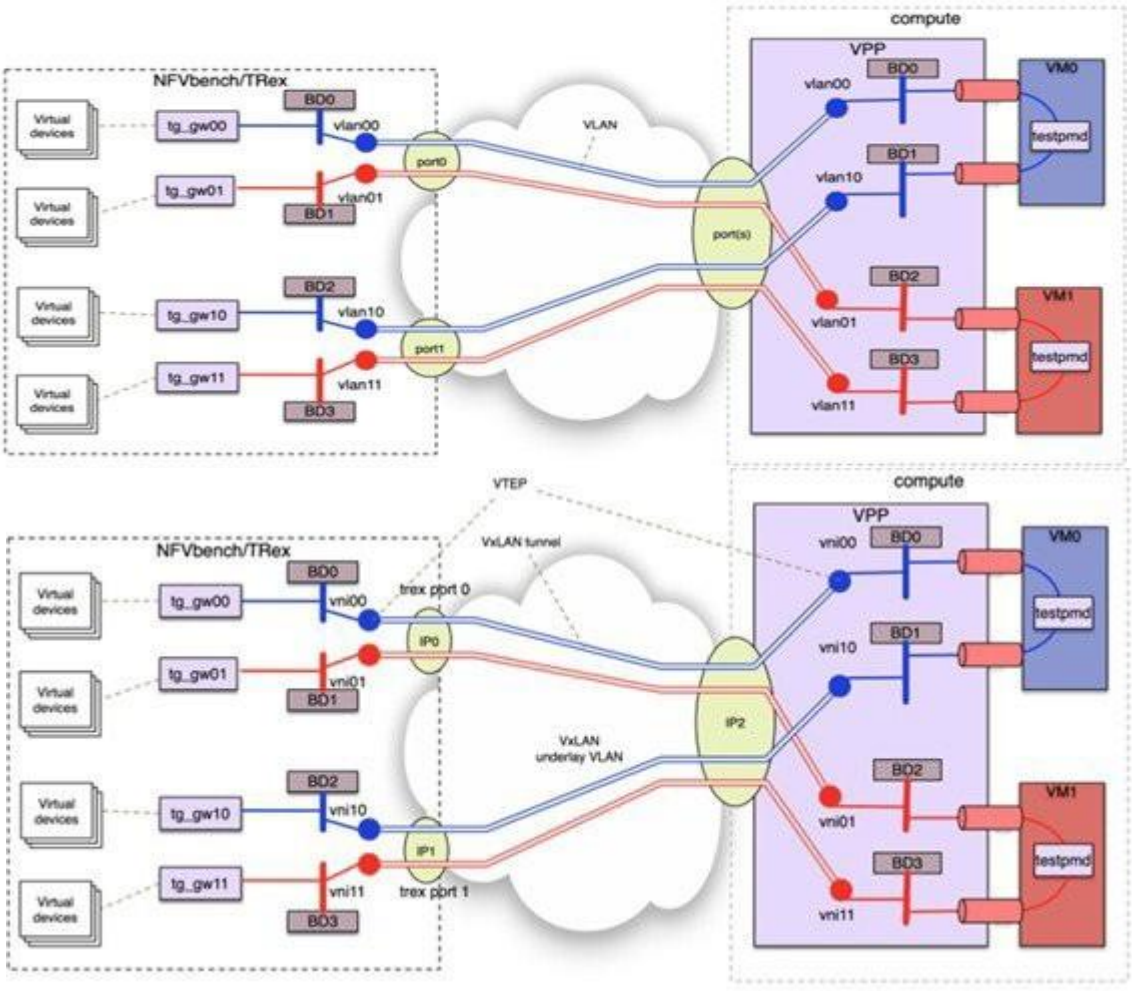
Encapsulation

NFVBench supports all networking options that can be deployed with VIM:

- OVS
- VPP with VLAN
- Single root input/output virtualization (SR-IOV)

By default, NFVBench uses VLAN tagging for the generated traffic and directs the traffic to the vswitch in the target compute node (OVS or VPP).

The following diagram illustrates an example of NFVBench execution with two chains using VLAN and when VPP is vswitch.



© 2026 HCLTech. All rights reserved.

SR-IOV

If SR-IOV is deployed, NFVBench can support to send the traffic to the test VMs that use SR-IOV instead of vswitch.

To test SR-IOV, you must have compute nodes configured to support one or more SR-IOV interfaces (also known as physical function (PF)) and OpenStack to support SR-IOV.

You need to know:

- The name of the physical networks associated with the SR-IOV interfaces (this is a configuration in Nova compute).
- The VLAN range is to be used for the switch ports that are wired to the SR-IOV ports. Such switch ports are normally configured in trunk mode with a range of VLAN IDs enabled on that port.

for example, if two SR-IOV ports exist per compute node, two physical networks are configured in OpenStack with a distinct name.

The VLAN range to use is also allocated and reserved by the network administrator and in coordination with the corresponding top of rack switch port configuration.

To enable SR-IOV test, you must provide the following configuration options to NFVbench in the configuration file.

The following example instructs NFVBench to create the left and right networks of a PVP packet flow to run on two SRIOV ports named phys_sriov0 and phys_sriov1using respective segmentation_id 2000 and 2001:

```
sriov: true
internal_networks:
left:
segmentation_id: 2000
```

The segmentation ID fields must be different.

In the case of PVVP, the middle network must be provisioned properly. The same physical network can also be shared by the virtual networks, but with different segmentation IDs.

VIM CLI

VIM CLI

An alternate way to NFVBench CLI is to use vimclient. vimclient provides an interface that is more consistent with the VIM CLI and can run remotely while the NFVBench CLI is executed on the management node.

Pass JSON configuration matching the structure of the NFVBench configuration file to start a test:

© 2026 HCLTech. All rights reserved.

```
[root@mgmt1 ~]# hvim nfvbench --config '{"rate": "10kpps"}
+------------------+--------------------------------------------------------------------------------+
| Name             | Value                                                                          |
+------------------+--------------------------------------------------------------------------------+
| status           | not_run                                                                        |
| nfvbench_request | {"rate": "5kpps"}                                                              |
| uuid             | 0f131259-d20f-420f-840d-363bdcc26eb9                                           |
| created_at       | 2017-06-26T18:15:24.228637                                                     |
+------------------+--------------------------------------------------------------------------------+
```

Run the following command with the returned UUID to poll status:

When the test is done, retrieve results in a JSON format:

```
[root@mgmt1 ~]# hvim nfvbench --json 0f131259-d20f-420f-840d-363bdcc26eb9
{"status": "PROCESSED", "message": {"date": "2017-06-26 11:15:37", ...}}
```


```
[root@mgmt1 ~]# hvim nfvbench --json 0f131259-d20f-420f-840d-363bdcc26eb9
{"status": "PROCESSED", "message": {"date": "2017-06-26 11:15:37", ...}}
```


NFVBench REST Interface

When enabled, the NFVBench container can also take benchmark request from a local REST interface. Access is only local to the management node in the VIM (that is the REST client must run on the management node). For more details on the REST interface calls, see VIM REST API > *VIM REST API Resources*.

```
[root@mgmt1 ~]# hvim nfvbench --stat 0f131259-d20f-420f-840d-363bdcc26eb9
+------------------+--------------------------------------------------------------------------------+
| Name             | Value                                                                          |
+------------------+--------------------------------------------------------------------------------+
| status           | nfvbench_running                                                               |
| nfvbench_request | {"rate": "5kpps"}                                                              |
| uuid             | 0f131259-d20f-420f-840d-363bdcc26eb9                                           |
| created_at       | 2017-06-26T18:15:24.228637                                                     |
| updated_at       | 2017-06-26T18:15:32.385080                                                     |
+------------------+--------------------------------------------------------------------------------+
+------------------+--------------------------------------------------------------------------------+
| Name             | Value                                                                          |
+------------------+--------------------------------------------------------------------------------+
| status           | nfvbench_completed                                                             |
| nfvbench_request | {"rate": "5kpps"}                                                              |
| uuid             | 0f131259-d20f-420f-840d-363bdcc26eb9                                           |
| created_at       | 2017-06-26T18:15:24.228637                                                     |
| updated_at       | 2017-06-26T18:18:32.045616                                                     |
+------------------+--------------------------------------------------------------------------------+
```

NFVBench REST Interface

When enabled, the NFVBench container can also take benchmark request from a local REST interface. Access is only local to the management node in the VIM (that is the REST client must run on the management node). For more details on the REST interface calls, see VIM REST API > *VIM REST API Resources*.

© 2026 HCLTech. All rights reserved.

NCS-5500 as ToR Option

NCS-5500 as ToR Option

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. The auto-ToR configuration is enabled 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. For more details, see ToR Management

Disk Management

Disk Management

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.

The disk status such as online, offline, and rebuilding helps you to identify the 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 pass through controller.

Once the disk is physically replaced, you can use the disk management tool to add the new disk back into the system as part of the RAID system (recommended one server at a time). For more information, see Disk and OSD Maintenance Tools.

Disk maintenance tool is useful only when one or two (in RAID6) disks is not working. Failure of more than one disk at a time sets the entire server in an irrecoverable state. Replace the server using remove and add operations through vim.

Disk management is not supported on HPE computes due to the licensing issue with the HPE SmartArray Utility tool.

Disk management is also not supported on DELL computers.

OSD Maintenance

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. It 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. It 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 re-balance 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

© 2026 HCLTech. All rights reserved.

ceph cluster from operation and then add it back through remove-storage and add-storage options in VIM. OSD Disk and OSD Maintenance Tools section has the relevant details.

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

Power Management

Power Management of Computes for C-Series

Though many compute servers are available for VIM pods, only limited compute servers are actually used at certain times. To optimize the overall power consumption of the data center, it is required to power down the server through an API/CLI.

To prevent the cloud destabilization, ensure that you do not power off all the compute nodes and maintain at least one pod in active state.

Pod management operation(s) applies to the entire pod during update and reconfiguration of the server. Update 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, the cloud-sanity is run internally. If the cloud-sanity fails, the power-off action is aborted. For more details, see Managing Power and Reboot

Physical Cores and Reserved Memory

Physical Cores and Reserved Memory

VIM has been tuned to deliver performance from an infrastructure and VNF point of view. The following table gives the details of the physical cores (regardless of whether hyperthread is enabled or not) that the infrastructure needs. The number of cores that are reserved for the system (host system + OpenStack services) is minimum of two in all cases.

Pod Type/Node Types	Control	Storage	Compute	AIO	HC	Control/Compute
FullOn	all	all	CPU: Q+V cores RAM: 25+Vr GB	n/a	n/a	n/a
Hyperconverged (hc)	all	n/a	CPU: Q+V cores	n/a	CPU: Q+C+V cores RAM: 41+Vr GB	n/a
Micropod (aio)	n/a	n/a	CPU: Q+V cores	CPU: Q+C+V cores	n/a	n/a

© 2026 HCLTech. All rights reserved.

Pod Type/Node Types	Control	Storage	Compute	AIO	HC	Control/Compute
				RAM: 41+Vr GB		
Edge-pod	Q	n/a	CPU: Q+V cores	n/a	n/a	CPU: Q+V cores

The following table shows the number of physical cores and RAM Reserved for VIM:

Variables	Usage	Valid range	Default
Q	Cores reserved for (host system and OpenStack Services)	2 to 12	2
C	Cores reserved for CEPH (AIO and HC)	2 to 12	2
V	Cores reserved for VPP vswitch	2 to 6	2
Vr	RAM reserved for VPP		2GB

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

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

Software Hub

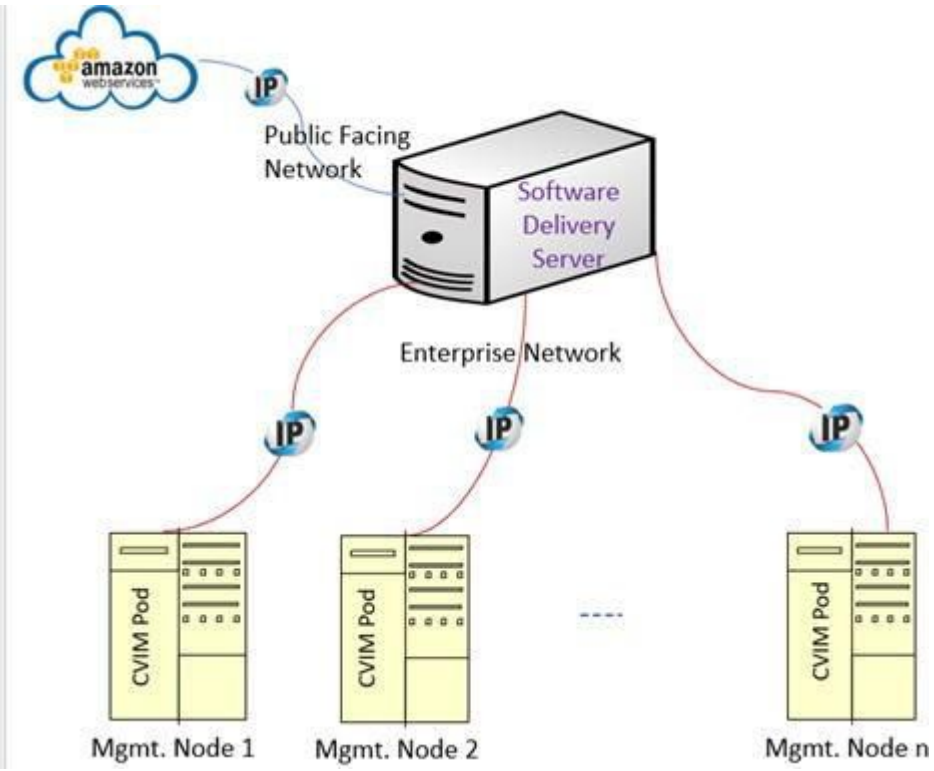
VIM Software Hub

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

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

You can install the artifacts available on the VIM Software Hub server through a connected or a disconnected installation procedure. For a connected installation, one end of the VIM Software Hub server is connected to the internet, while the other end is connected to the data center. The following figure shows the architecture of a connected installation.

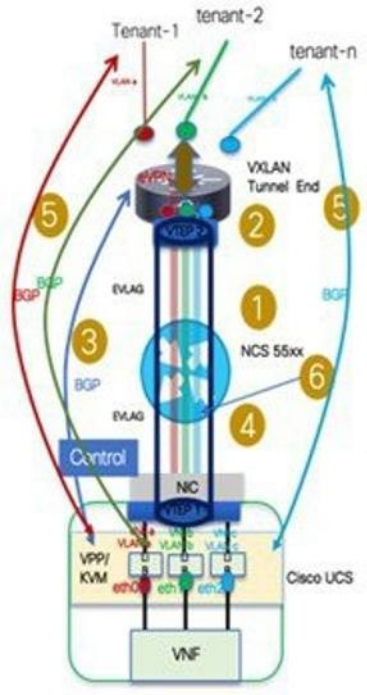
© 2026 HCLTech. All rights reserved.



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

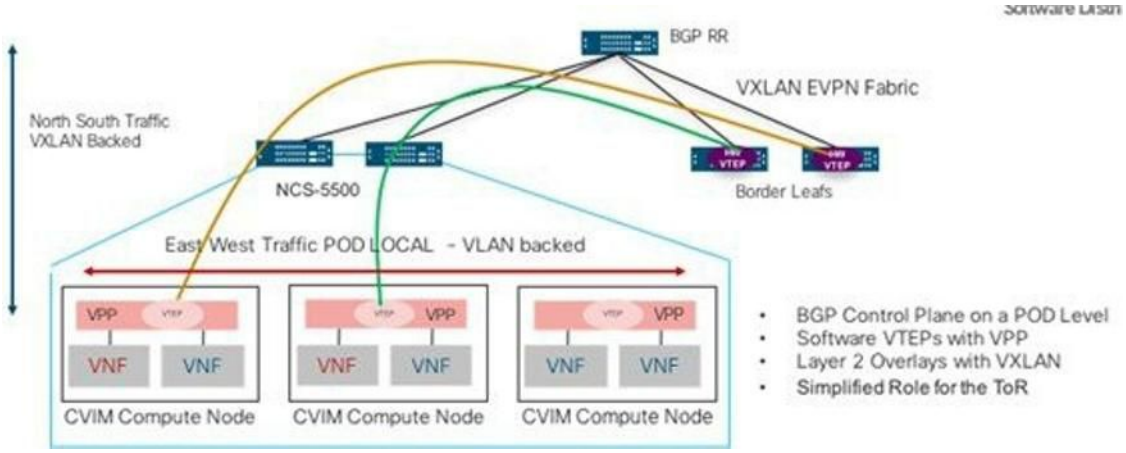
Overview

below

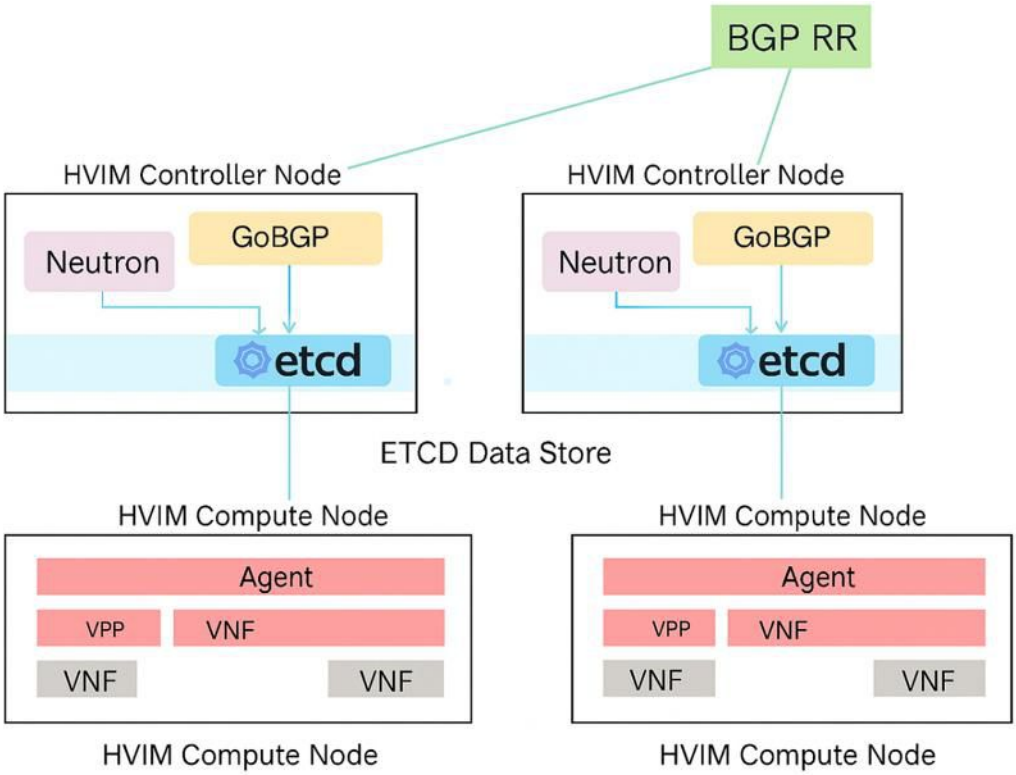


- **Segment 1: Compute to Layer 2 Aggregation (Layer 2 EVLAG)**
  - VPP will establish EVPN based EVLAG with aggregation layer
- **Segment 2: Layer 2 Aggregation switch to Fabric**
  - Fabric node will establish redundancy with aggregation layer
- **Segment 3: Infrastructure Control Plane**
  - EVPN between Compute VTEP and Remote VTEP to enable dynamic MAC learning
- **Segment 4: Infrastructure Data Plane between Compute and Tenants**
  - VXLAN will be used as a data plane for traffic transport between compute and Tenants
- **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 Remote tunnel end, 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

© 2026 HCLTech. All rights reserved.



Network



© 2026 HCLTech. All rights reserved.

VPP Port Mirroring Support

Architecture

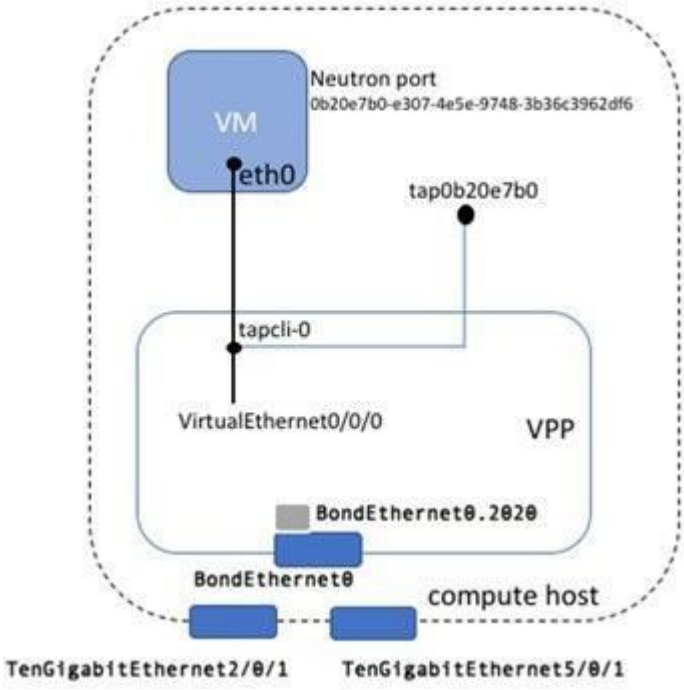
Limitations

From release VIM 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.

Architecture

Port mirroring works by setting up the following:

- A span port on vpp to mirror the Virtual Ethernet interface corresponding to the VMs vhost interface. This is a tap interface in VPP.
- 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.



Limitations

- 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 requires 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-Virtual Ethernet interfaces are not supported.

© 2026 HCLTech. All rights reserved.

Segment Routing EVPN

VIM Segment Routing EVPN Design

- Overview of Segment Routing EVPN
- Overview of VIM VPP Architecture Overview
- Overview of VIM SR EVPN Architecture

Overview of Segment Routing EVPN

An important aspect of any Telco cloud is how the cloud is connected to the rest of the service provider network. Due to the evolution of the existing VPP- based standard VLAN and EVPN designs, it is possible to connect VIM with an existing Segment Routing (SR) Ethernet VPN (EVPN). You can connect VIM to the SR EVPN without an additional SDN controller, by peering with EVPN route reflectors and ToR with BGP Labeled Unicast

(BGP-LU). There is no contention with any controller that manages the EVPN. MPLS is used as the data plane for the SR-labelled traffic. To connect a VIM pod to an SR EVPN, the existing VPP forwarding architecture is enhanced.

Overview of VIM VPP Architecture

Networking-vpp is the Vector Packet Processing (VPP) based software accelerated virtual switch that is part of VIM. The architecture of networking-vpp is similar to a distributed SDN controller. However, it is not a separate controller and is integrated with VIM as a core component. Networking-vpp makes installation, updates, and operational Day 2 tasks seamlessly.

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

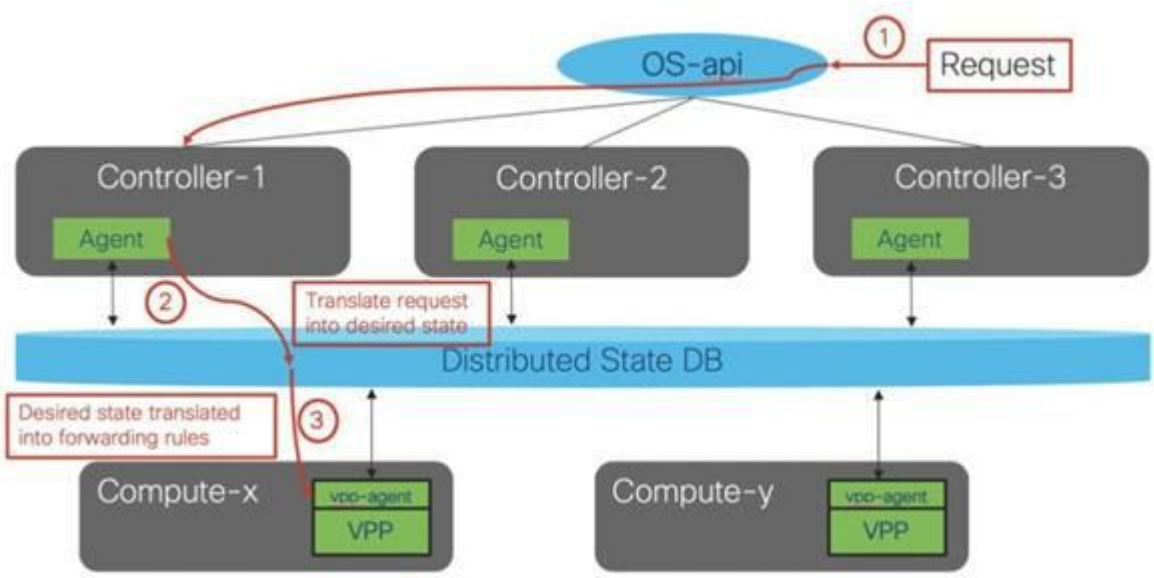
The controllers do not have to:

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

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

The figure below depicts the VPP architecture.

© 2026 HCLTech. All rights reserved.

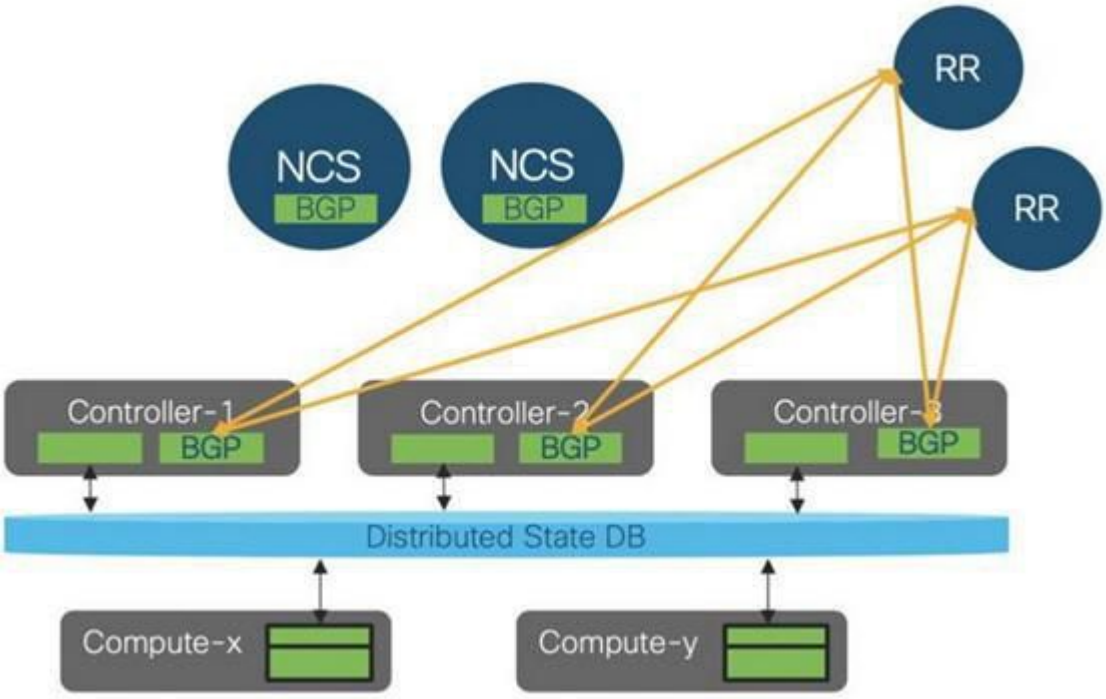


The following steps explain the figure give above:

- The agent running on the controllers gets the incoming requests and processes the API calls that relate to the state of the network data plane forwarding.
- The agent translates the request into the desired state of the network data plane and publishes this state into a distributed state database.
- The vpp-agents running on the compute nodes watch the distributed state database. When they see a change relevant to the compute node, they translate the desired state and update the VPP forwarding on the compute node.

Overview of VIM SR EVPN Architecture

To extend the networking-vpp architecture to support Segment Routing over MPLS, VIM 3.4.1 adds several additional components. The figure below depicts the SR EVPN RR connections.

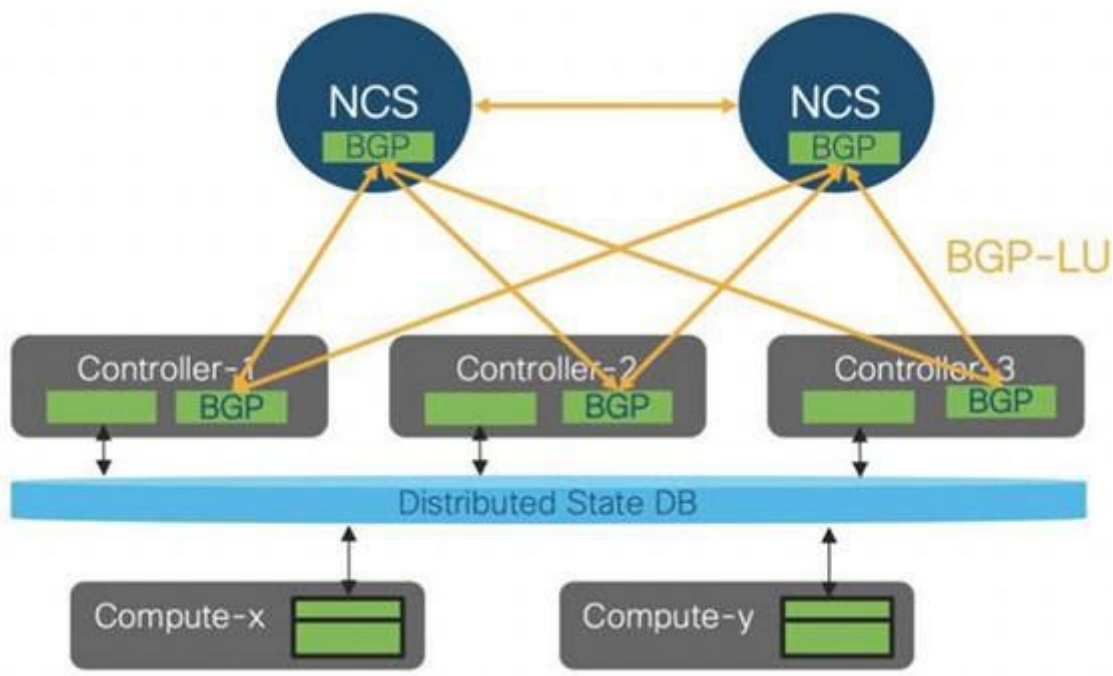


© 2026 HCLTech. All rights reserved.

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

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

The figure below depicts the SR EVPN BGP-LU peering with the ToR.

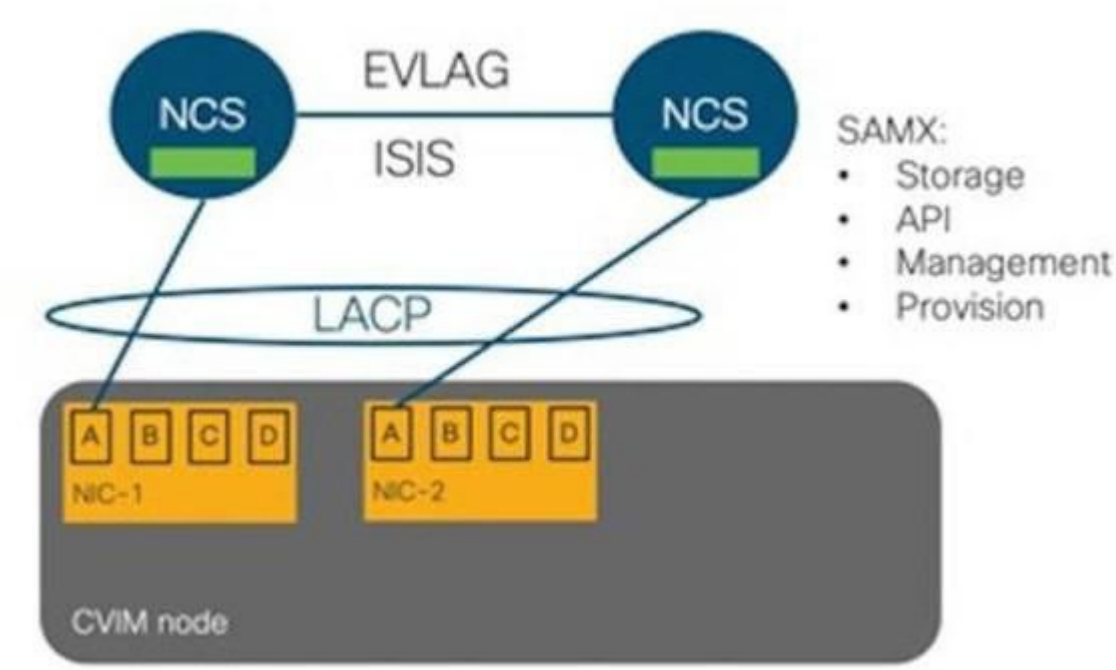


VIM does not process any SR policies coming from BGP or an SDN controller to program any explicit static or dynamic paths into VPP. BGP-LU and EVPN address family routes are processed by the BGP process running on each of the controllers. The BGP process publishes the SR policy into the distributed state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

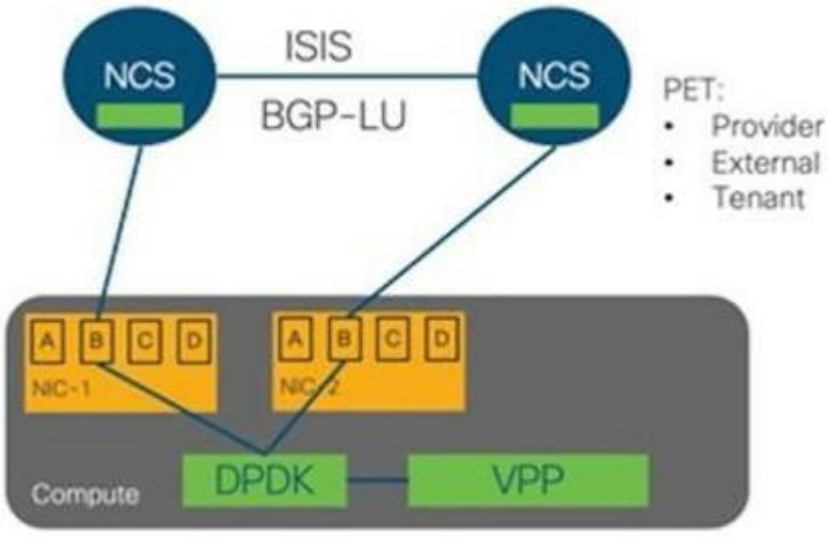
The data plane ports in the VIM pod are connected to the NCS with an L3 link and EVLAG provides link redundancy to the nodes. The VPP agents encapsulate the traffic by leaving the compute node with an MPLS label representing the Segment Routing segment ID. The virtual machines are unaware that traffic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

© 2026 HCLTech. All rights reserved.

The figure below depicts the control of plane port connectivity.



The control plane ports on the VIM nodes have two uplinks, one to each NCS. From the node, the two ports are bundled in a port-channel using the linux teaming driver. ISIS runs on the NCS and provides link redundancy between the two ToRs using EVLAG. The control plane ports carry any or all of the following networks: storage network to access the CEPH cluster, API network hosting the OpenStack API, Management and provision networks used at install time to provision the nodes and during Day 2 operations for management tasks.



The figure below depicts the data plane port connectivity.

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

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

© 2026 HCLTech. All rights reserved.

L3 Fabric Deployment

VIM in Layer 3 Fabric Deployment

For implementation details, see L3 Fabric Deployment Details

OpenStack Barbican

OpenStack Barbican

Overview

Barbican, also known as the Key Manager service, manages secrets such as keys and passwords. The implementation consists of three components: • Barbican API

- Barbican Worker
- Barbican Keystone Listener

As with other OpenStack services, Barbican uses a SQL database to store metadata about the managed secrets and a message queue for communication between its components.

Barbican supports different types of secret storage back ends, including hardware security modules. For more information, see the following:

- Key Manager service overview
- OpenStack Barbican Architecture
- Configuring Secret Store Back-end
- Barbican API Documentation
- Volume encryption supported by the key manager

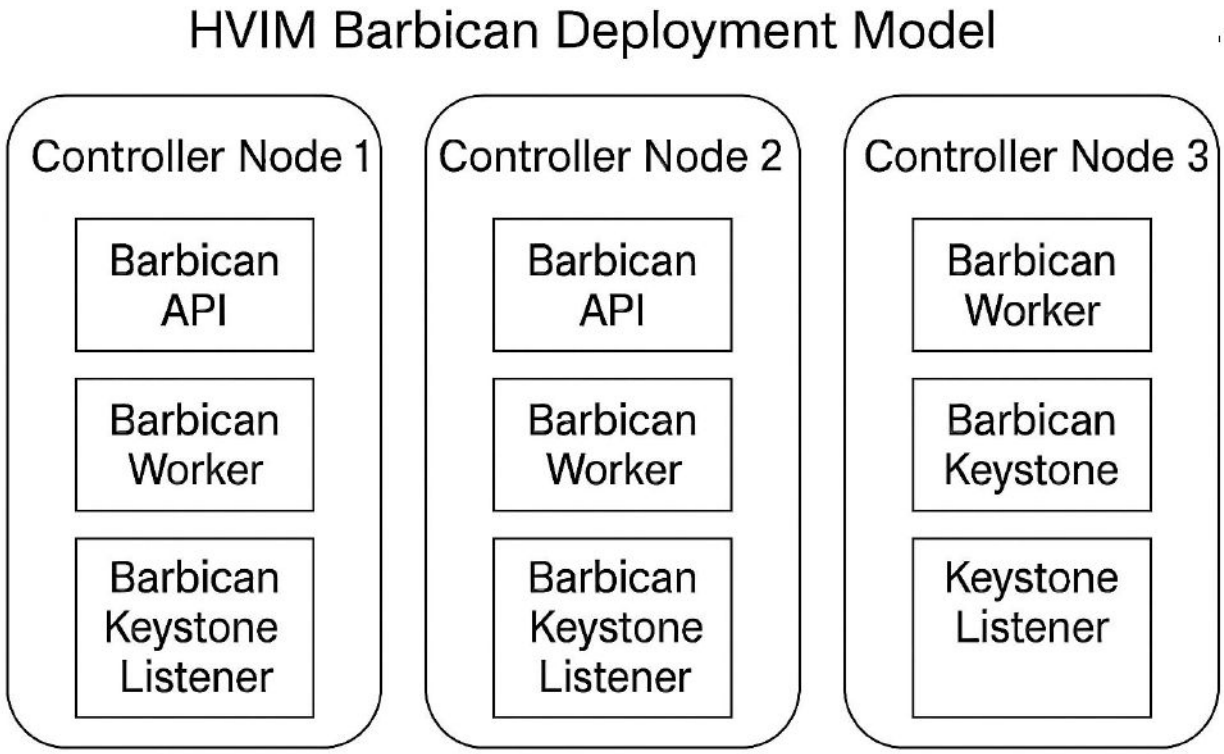
Barbican Architecture in VIM

The following are the assumptions in Barbican architecture:

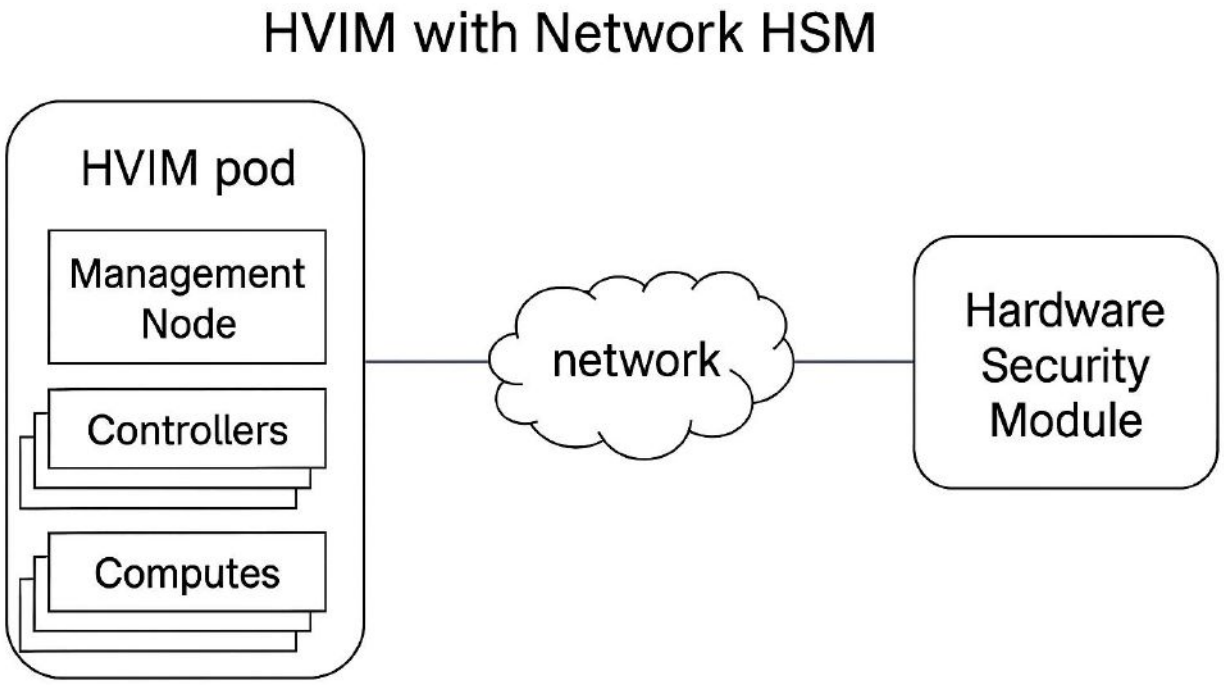
- Hardware Security Modules (HSM) exists in the network that is reachable by the VIM pod.
- Installation and management of the HSM are outside the scope of VIM and are done ahead of time.
- You must provide HSM client cert, client key, HSM server cert(s), and IP addresses.
- The HSM must be reachable from VIM controller nodes via the management network of VIM.
- All testing and validation are done by using ATOS Trustway Proteccio NetHSM.
- You need to extract the libnethsm.so file from the ATOS client software, for example, Proteccio1.09.03.iso offline, copy it into the management node, and update the relevant line of the setup data to point to libnethsm.so location in the management node.
- Though Barbican can be configured to use multiple secret stores, VIM allows only one secret store back end to be configured at a time.

In VIM, Barbican components are deployed on the VIM controller nodes only as depicted below:

© 2026 HCLTech. All rights reserved.



Also, following is a typical layout of a VIM pod interacting with a network-accessible HSM. The HSM must be reachable from the VIM controller nodes via the management network. The HSM is deployed, configured, and managed externally to VIM.



When Barbican is installed on a VIM pod, Cinder and Nova uses Barbican to store and retrieve volume encryption keys. Barbican is an optional service that can be included in the initial installation or added with a reconfigure operation. For details on how to enable Barbican to see Enabling Barbican

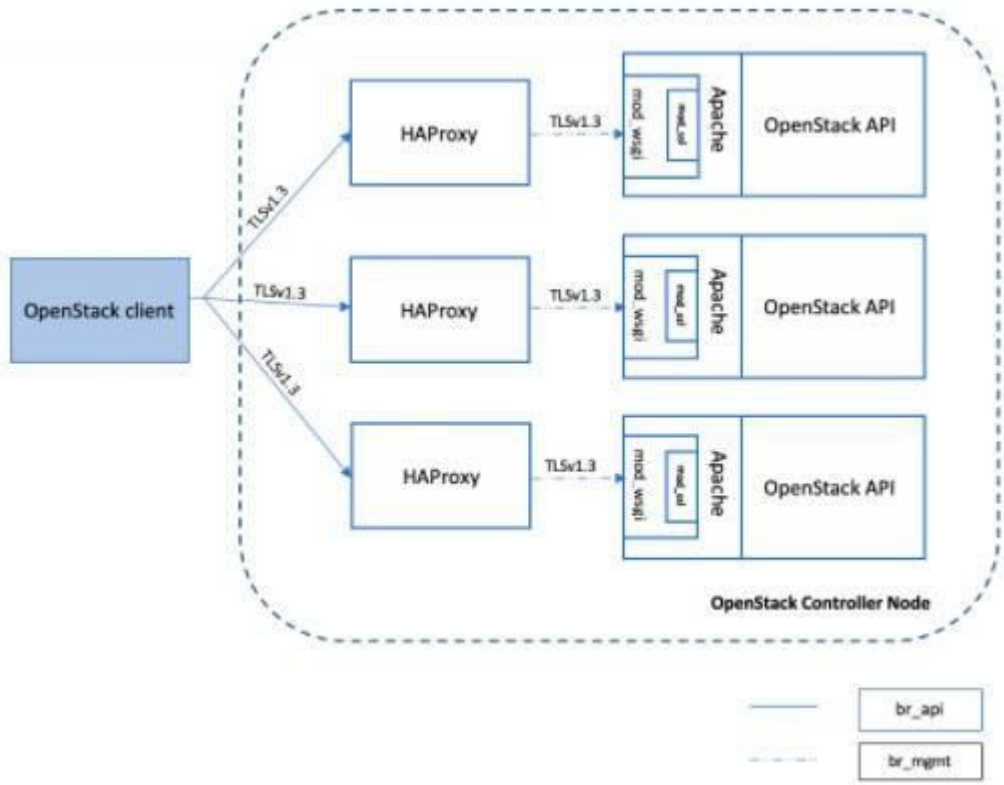
© 2026 HCLTech. All rights reserved.

Management Network Over TLS

Management Network Over TLS

VIM is designed to secure the connections between nodes in the pod over the management network using TLS. For releases earlier to VIM 4.0.0, all external interfaces are protected over TLS, but not the intra-node management traffic. If the management network over TLS is not enabled, ensure that traffic over the management network flows on a non-routable address space.

From VIM 4.0.0, you can optionally enable management network over TLS either on Day 0 via fresh installation or Day 2 via reconfiguration. For more details, see Enabling Management Network over TLS. The OpenStack control plan architecture is designed to run OpenStack API services behind the reverse proxy (apache) with *mod_wsgi/uwsgi* implementation as shown in the below figure.



With TLS management network enabled, OpenStack endpoints are hosted over https. The required certificates are created during orchestration and then distributed to VIM nodes such as controllers and computes, so that OpenStack services communicate over a secure channel.

Support of Pre-encrypted Image

Support of Pre-encrypted Image

From VIM 4.0.0, you can use pre-encrypted images under the assumption that Barbican is enabled. The following aspects of security are addressed:

- Privacy: Prevents the possibility of leaking information from the image without the encryption key.
- Integrity: Prevents the possibility of modifying the software without the encryption key.

To build encrypted images, you must have your own infrastructure. Planning and implementation of mechanism to build encrypted images is outside the scope of VIM. As VIM is an embedded solution, installation of additional software on the servers managed by VIM is never allowed. For more detas, see Usage of Pre-encrypted Images.

© 2026 HCLTech. All rights reserved.

Installation

Installation

- NFVI Installation Overview
- Installation Preparation Without Internet Access
- Preparing for NFVI Installation
- VIM Monitor
- Highly Available VIM Monitor
- VIM
- Unified Management

NFVI Installation Overview

NFVI Installation Overview

NFVI installation is divided into two processes:

- Preparation—Preparing the NFVI pod hardware and configuring all supporting applications including Cisco Integrated Management Controller (IMC) and Cisco UCS Manager.
- Installation—Installing the NFVI component applications such as Virtualized Infrastructure Manager (VIM) and Unified Management (UM) based on your NFVI package.

NFVI installation depends on the component applications that you install. When installing VIM UM, install the VIM management node and UM node in a sequence to complete the VIM installation through VIM UM. However, if you have VIM without other NFVI applications in your package, you can install the VIM alone in your system.

Consider the following factors before installing the NFVI components:

- **Internet Access**: Internet access is required to download the NFVI installation files from vim-registry.com

If you do not have an Internet access to your management node, you need an alternate server with an Internet access to download the installation files to a USB stick. You can copy the installation files from USB stick to the management node.

NFVI Configurations: NFVI configurations are included in the setup_data.yaml file. If you are installing VIM and not VIM Insight, you can enter the configurations directly into the setup_data.yaml file with a yaml editor. You can refer to the examples in setup_data file (for C-series) at the openstack-configs directory in the target install folder in the management node. For more information on NFVI data and OpenStack parameters, see OpenStack Configuration. If you are installing VIM UM, run NFVI using Insight UI wizard. For more information, see Unified Management.

The following are the license options for installing NFVI:

- NFVI Basic: Includes Virtual Infrastructure Manager (VIM) which is an OpenStack software solution used to enhance the functionality, scale, and performance of the node.
- NFVI Standard: Includes VIM and VIM Insight. VIM Insight deploys, provisions, and manages NFVI on Cisco UCS servers.
- NFVI with third-party monitoring: Includes VIM with or without VIM UM based on the license option chosen, with monitoring of the pod through VIMMON.
- NFVI with monitoring: Includes VIM with or without VIM UM based on the license option chosen, with monitoring of the pod through VIMMON

You must perform extra manual installation procedures while installing VIM. If your package includes VIM and UM, you must do VIM manual setup and configuration procedures through UM. You can manage cloud in VIM through VIM UM. Once you start managing the cloud, your recommendation is to continue using VIM UM for future use as well.

© 2026 HCLTech. All rights reserved.

Installation Mode

Installation Mode

You can deploy VIM in one of the following install modes:

- Connected: In this mode, the setup must be connected to the Internet or VIM software hub to fetch artifacts and docker images.
- Disconnected: In this mode, VIM is not connected to the Internet. The artifacts and docker images are loaded from USB device.

Based on the deployment type, select the install mode as connected or disconnected as given below:

```
# Install Mode: connected/disconnected INSTALL_MODE: connected
```

⚠ You can switch from one mode to another via the reconfigure option.

Installation Preparation Without Internet Access

Preparing for Installation on Servers Without Internet Access

- Air-gapped Installation Approach
- Prerequisites for Air-Gapped Installation
- NFVI Installation Setup via USB
- NFVI Installation File-Based Image

Air-gapped Installation Approach

Approaches for Air-Gapped Installation

Listed below are the two approaches for air-gapped installation in VIM.

- Approach 1: Getting the artifacts onto USB from a staging server running CentOS/RHEL 9.4
- Approach 2: Creating a file-based image on the staging server running CentOS/RHEL 9.4

For both the approaches, the staging server is connected to the docker registry.

In the first approach, you can do the installation independently post preparation of the USB. However, it involves an additional burden of shipping the physical USBs to each pod, which may be cumbersome. In the second approach of file-based image, the burden associated with first approach is reduced. However, each management node must be connected to the staging server.

Prerequisites for Air-Gapped Installation

Prerequisites for Air-gapped Installation via USB

- Download the Cisco NFVI installation files to a 64GB (minimum) USB 2.0 drive on a staging server with Internet access. If the management node is based on M5 or M6 or a Quanta server, you can optionally use USB 3.0 100GB to increase the installation speed significantly.
- Copy the files to the management node.

© 2026 HCLTech. All rights reserved.

Prerequisites for Air-gapped Installation via File-based image

- The CentOS/RHEL 8.4 staging server (VM, laptop, or UCS server) is connected to the docker registry.
- Each management node must be connected to this staging server.
- Ensure that the packages truncate and parted with kpartx installed on the CentOS/RHEL 9.2 staging server.

NFVI Installation Setup via USB

NFVI Installation Setup via USB

The following procedure describes how to download the NFVI installation files onto a USB drive of the staging server with Internet access. You can use the USB to load the NFVI installation files onto the management node without Internet access.

⚠ Recommendation is to use Virtual Network Computing (VNC), other terminal multiplexers, or similar screen sessions to complete these steps.

ⓘ **Before you begin**

You must have a CentOS/RHEL 9.4 staging server (VM, laptop, or UCS server) with a 64 GB USB 2.0 drive only. You can use USB 3.0 64GB if the management node is of type M5 or M6 or M7. The staging server must have wired Internet connection to download the vim installation files onto the USB drive. Once downloaded, you can copy the installation files onto the management node from USB drive.

⚠ Downloading of the installation files (over 25 GB in size) to the USB drive might take several hours depending on the speed of your Internet connection. Ensure that you disable the CentOS to the sleep mode, for faster installation.

1. On the staging server, use yum to install the following packages:
- PyYAML (*yum install PyYAML*)
- python-requests (*yum install python-requests*)

Check whether python 3.6.x version is installed. If not, perform *yum install python3*.

2. Log into VIM software download site and download the *getartifacts.py* script from external registry:

```
# Download the new getartifacts.py file (see example below)
curl -o getartifacts.py -u '<username>:<password>' https://vim-registry.com/mercury-releases/hvim50- rhel9-osp17/releases/<releaseid>/getartifacts.py
# Change the permission of getartifacts.py
chmod +x getartifacts.py
```

3. Run *getartifacts.py*. The script formats the USB 2.0 drive (or USB 3.0 drive for M5/M6/Quanta based management node) and downloads the installation files. You must provide the registry username and password, tag ID, and USB partition on the staging server.

© 2026 HCLTech. All rights reserved.

```
./getartifacts.py -h
usage: getartifacts.py [-h] -t TAG -u USERNAME -p PASSWORD [--proxy PROXY] [--retry] (-d DRIVE | -f
                       FILE)
                       [--mgmtk8s | --argus | --insight | --sds | -U]

Script to pull container images en masse

optional arguments:
  -h, --help            show this help message and exit Installer
  -t TAG, --tag TAG     version to pull
  -u USERNAME, --username USERNAME
                        Registry username
  -p PASSWORD, --password PASSWORD
                        Registry password
  --proxy PROXY         https_proxy if needed
  --retry               Try to complete a previous fetch
  -d DRIVE, --drive DRIVE
                                Provide usb drive path
                        -f FILE, --file FILE   location of image file
  --mgmtk8s             Additionally download hvim MON HA artifacts
  --argus               Additionally download argus artifacts
  --insight             Additionally download insight artifacts
  --sds                 Additionally download sds artifacts
  -U, --upgrade         Additionally download artifacts for upgrade from 2.4.x (This option is not applicable for upgrading from
4.2.x to 4.4.0)

This script pulls images from remote registry then copies the contents to USB drive or image file based on the user
```

4. To identify the USB drive, execute the *lsblk* command before and after inserting the USB drive. The command displays a list of available block devices. The output data helps you to find the USB drive location. Provide the entire drive path in the *–d* option instead of any partition as shown below. Here, the *tag_id* refers to the VIM 4.x.

For example:

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -d </dev/sdc> \[--proxy proxy.example.
com\] -
```

> ⚠ Ensure that you do not remove the USB drive during synchronization.

> ⚠ On executing *getartifacts.py*, the following message appears to notify bad superblock and mount failure. In this case, reformat the drive and use the *fsck* command to recover the drive: *fsck.ext4-pv/dev/sdc1*.
> *stderr:mount:wrongfstype,badoption,badsuperblock on /dev/sdc1, missing codepage or helper program, or other error.*

> ⚠ As the size of the artifacts is greater than 25G, it is recommended to execute this step over a wired internet connection. It takes few hours to download and populate data on the USB drive, depending on internet connectivity.

The *getartifacts.py* script downloads the following:

a. Core packages

- all_check_sum_file.sign.tar.gz
- buildnode-K9-XXXXX.iso
- buildnode-K9-XXXXX.qcow2
- VIM-upgrade-installer-XXXXX.tar.gz getartifacts.py
- ironic-images-K9.tar.gz
- mercury-installer.tar.gz
- registry-2.6.2-XXXXX.tar.gz
- VIM_upgrade_orchestrator.py

© 2026 HCLTech. All rights reserved.

> ⚠ For upgrade :*vim_upgrade_orchestrator.py* script is required.

b. Respective checksums *all_check_sum_file.sign.tar.gz*

5. Use the following command to verify the downloaded artifacts and container images:

```
# Create a directory
sudo mkdir -p /mnt/Cisco
# /dev/sdc is the USB drive, same as supplied in getartifacts.py python script
#You need to mount the partition with the steps given below:
sudo mount /dev/sdc1 /mnt/Cisco
cd /mnt/Cisco

# Execute the test-usb help to look at the options
./test-usb -h
usage: ./test-usb [-h] -- Show this program to check integrity of artifacts in this USB drive/image file
    [-l] -- Location of artifacts
    [-f] -- Location of image file

# Execute the verification script
./test-usb

# failures will be explicitly displayed on screen, sample success output below# sample output of ./test- usb with 4.4.0 release
/Test-usb -f /var/artifacts.img
INFO: Checking the integrity of artifacts on this Image file INFO: Checking artifact vim_upgrade_orchestrator.py
INFO: Checking artifact mercury-installer.tar.gz
INFO: Checking artifact ironic-images-K9.tar.gz
INFO: Checking artifact buildnode-K9-xxxxx.iso
INFO: Checking artifact registry-2.6.2-xxxxx.tar.gz
INFO: Checking artifact buildnode-K9-xxxxx.qcow2
INFO: Checking required layers:
INFO: 881 layer files passed checksum.
```

If the download fails, an error message is displayed. For example:

© 2026 HCLTech. All rights reserved.

```
# ./test-usb
INFO: Checking the integrity of this USB stick
INFO: Checking artifact buildnode-K9.iso
ERROR: Checksum for artifact buildnode-K9.iso does not match ('SHA512 (buildnode-K9.iso) =
96ec62a0932a0d69daf60acc6b8af2dc4e5eca132cd3781fc17a494592feb52a7f171eda25e59c0d326fbb09194ee
da66036cbdc3 870dafe74f59cf1f2dce225'
!= 'SHA512 (buildnode-K9.iso) =
a6a9e79fa08254e720a80868555679baeea2dd8f26a0360ad47540eda831617bea0514a117b12ee5f36415b7540
afa112a1c904cd 69e40d704a8f25d78867acf')
INFO: Checking artifact registry-2.3.1.tar.gz
ERROR: Artifact registry-2.3.1.tar.gz is not present
INFO: Checking required layers:
ERROR: Layer file sha256:002aa1f0fbdaea7ea25da1d906e732fe9a9b7458d45f8ef7216d1b4314e05207 has a bad
checksum
ERROR: Layer file sha256:5be3293a81773938cdb18f7174bf595fe7323fdc018c715914ad41434d995799 has a
bad checksum
ERROR: Layer file sha256:8009d9e798d9acea2d5a3005be39bcbfe77b9a928e8d6c84374768ed19c97059 has a
bad checksum
ERROR: Layer file sha256:ea55b2fc29b95d835d16d7eeac42fa82f17e985161ca94a0f61846defff1a9c8 has a bad
checksum
INFO: 544 layer files passed checksum.
```

To resolve download artifact failures, unmount the USB and run the *getartifacts* command again with the *--retry* option.

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -d </dev/sdc> --retry
```

Mount the USB and then run the test-usb command to validate if all the files are downloaded:

```
# /dev/sdc is the USB drive, same as supplied in get artifacts.py python script
sudo mount /dev/sda1 /mnt/Cisco
cd /mnt/Cisco
# Execute the verification script
./test-usb
# In case of failures the out of the above command will explicitly display the same on the screen
```

When the USB integrity test is done, count the USB drive by running the following command:

```
sudo unmount /mnt/Cisco
```

To import the artifacts onto the management node., execute import_artifacts.sh on the management node (ensure the USB is connected to the management node, where the USB contains the VIM artifacts):

© 2026 HCLTech. All rights reserved.

```
# cd /root/installer-<release_tag>/tools/
# ./import_artifacts.sh
# ./import_artifacts.sh -h
import_artifacts.sh : Program to import artifacts from USB


usage: ./import_artifacts.sh
-s : Specify this flag is importing on SDS server
-f : Specify this flag along with image file location as an argument to import artifacts
-h : To display this help
```

NFVI Installation File-Based Image

NFVI Installation via File-Based Image

- Installation Overview
- Assumption
- Prerequisites
- Procedure

Installation Overview

To download VIM artifacts for disconnected installation, an image file-based option is available for deployments where shipping of physical USB is not feasible for an air-gapped installation.

> **Before you begin**
>
> You must have a CentOS/RHEL 8.4 staging server (VM, laptop, or UCS server) that is connected to the docker registry. Also, each management node must be connected to this staging server.

Download the VIM artifacts to a file-based image on the staging server (running CentOS/RHEL 8.4) that is connected to the docker registry. Once the artifacts are downloaded, copy the corresponding file-based image to the management node and subsequently import the respective artifacts to the management node. You can apply the same method of importing the artifacts for an update or upgrade procedure if desired.

Assumption

- Applicable only for the fresh installation of VIM 3.4.3 and later versions. Also, you can use it to execute an update starting VIM 3.4.3. In the
- *setup_data.yaml,* INSTALL_MODE must be set to *disconnected*.

© 2026 HCLTech. All rights reserved.

Prerequisites

Ensure that the packages truncate and gets parted with *kpartx* installed on the CentOS/RHEL 8.4 staging server, which is connected to the docker registry.

Procedure

The following are the steps that outline the process:

On the staging server, use yum to install the following packages:

- PyYAML (yum install PyYAML)
- python-requests (yum install python-requests)

Check whether python 3.9.x version is installed. If not, perform *yum install python3*.

- To download artifacts to file-based image (which is approximately 60GB in size), log into VIM software download site and download the *geta rtifacts.py* script from external registry onto the staging server:

```
curl -o getartifacts.py -u '<username>:<password>' https://vim-registry.com/mercury-releases/hvim50- rhel9-osp17/releases/<releaseid>/getartifacts.py
# Change the permission of getartificats.py
chmod +x getartifacts.py
```

- On the staging server, execute the *getartifacts.py* script with the *--file* option to download the correct target artifacts:

```
#./getartifacts.py -u <username> -p   <password> -t <release-tag> --file /dir/path/to/file --proxy PROXY Example of the execution
# ./getartifacts.py -u installer -p  password -t 5.0.2 --file /var/artifacts-5-5-0.img
```

If your environment has proxy enabled, use the command below to download the target artifacts.

> ⚠ For 5.0.4 upgrade, it is mandatory to provide the -U option which will download the upgrade installer container image in to /etc/cisco/artifacts.
>
> As this process cannot be aborted in the middle, it is recommended to use either a VNC session/screen/KVM or serial console.

```
#./getartifacts.py -u <username> -p   <password> -t <release-tag> --file /dir/path/to/file --proxy PROXY - U
```

> ⚠ To download additional target artifacts, check the help option of *getartifacts.py*.

- On the staging server, verify the integrity and consistency of the downloaded artifacts in the file-based image:
- Copy the file-based image */var/artifacts-<release_tag>.img* from the staging server to */var/* of the target

```
# curl -o mercury-installer.tar.gz https://username:password@vim-registry.com/mercury-releases/hvim50-rhel9-osp17/releases/<releaseid>/mercury-installer.tar.gz
# tar --no-same-owner -xvzf mercury-installer.tar.gz
# cd installer-xxx/tools/
# ./test-usb -f /var/artifacts-<release_tag>.img
```

management node:

© 2026 HCLTech. All rights reserved.

To import the artifacts onto the management node., execute *import_artifacts.sh* with *-f* option on

the management node, and provide the location of the file-based image:

```
# scp /var/artifacts-<release_tag>.img root@<management-node-ip>:/var/

Repeat the test-usb on the management node (from the installer-xxx/tools workspace) for consistency check
# cd installer-5.0.0/tools/
# ./test-usb -f /var/artifacts-<release_tag>.img
Execution Snippet:
/test-usb -f /var/artifacts.img
INFO: Checking the integrity of artifacts on this Image file
INFO: Checking artifact vim_upgrade_orchestrator.py
INFO: Checking artifact mercury-installer.tar.gz
INFO: Checking artifact ironic-images-K9.tar.gz
INFO: Checking artifact buildnode-K9-xxxxx.iso
INFO: Checking artifact registry-2.6.2-xxxxx.tar.gz
INFO: Checking artifact buildnode-K9-xxxxx.qcow2
INFO: Checking required layers:
INFO: 881 layer files passed checksum.
```

Preparing for NFVI Installation

Note: VIM no longer supports Cisco UCS-M4 Hardware

Preparing for NFVI Installation

- NFVI Hardware Installation
- ToR Switch Configuration for C-Series Pods Preparing Cisco IMC
- Management Node on UCS C-series (M4/M5/M6)
- Management Node on Quanta Servers
- VIM Software Hub
- UCS C-Series Pod
- Out-of-Band Management Switch
- Third-Party Compute Support

NFVI Hardware Installation

NFVI Hardware Installation

Note: VIM no longer supports Cisco UCS-M4 Hardware

Switch on the servers, before you install the VIM. Set up the CIMC connection so that the servers are reachable from the management node via *br_a pi*.

The following table lists the UCS hardware options and network connectivity protocol used with VLAN over OVS, or VLAN over VPP.

© 2026 HCLTech. All rights reserved.

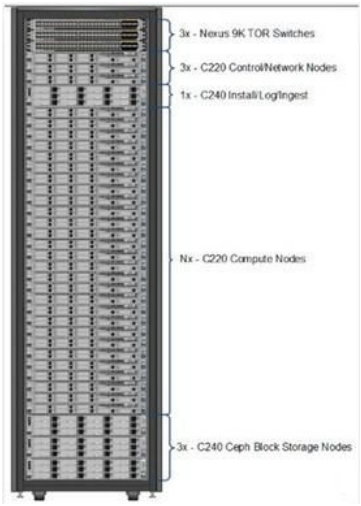
UCS Pod Type	Compute and Controller Node	Storage Node	Network Connectivity Protocol
Rack Type	UCS C220/240 M5/M6/M7	UCS C240 M5/M6 (SFF), M7	OVS/VLAN or VPP/VLAN (only on intel NIC)
Rack Type	Controller: UCS C220/240 Compute: HP DL360 Gen9 Compute: Dell PowerEdge 740 Quanta servers for fullon or edge pod	UCS C240 M7	OVS/VLAN
Micropod	UCS 240 M5/M6/M7	Not applicable as it is integrated with compute and controller	OVS/VLAN or VPP/VLAN (on intel NIC)
C-series Hyperconverged	UCS 240 M5/M7	UCS C240 M5 (SFF), M7	OVS/VLAN

⚠ For BOM details, contact vim Product Management team : nfvi-plm@hcltech.com

In addition, the Cisco Nexus 9372 or 93180YC, or 9396PX is also available to serve the NFVI ToR function. Newer versions of Nexus work as long as it supports virtual port channel and is compatible with the server NIC.

The figure below shows a schematic C-Series NFVI pod. Although the figure shows a full complement of UCS C220 compute nodes, the number of compute nodes vary depending on the implementation requirements. The UCS C220 control and compute nodes can be replaced with UCS 240 series.

However, in that case the number of computes fitting in one chassis system is reduced by half. The following figure shows the NFVI C-series pod.



For more information on wiring schematic of various pod configuration, see Wiring Diagrams.

© 2026 HCLTech. All rights reserved.

NFVI Hardware Installation

Switch on the servers, before you install the VIM. Set up the CIMC connection so that the servers are reachable from the management node via *br_a pi*.

The following table lists the UCS hardware options and network connectivity protocol used with VLAN over OVS, or VLAN over VPP.

UCS Pod Type	Compute and Controller Node	Storage Node	Network Connectivity Protocol
Rack Type	UCS C220/240 M5/M6/M7	UCS C240 M5/M6/M7 (SFF)	OVS/VLAN or VPP/VLAN (only on intel NIC, not on M6)
Rack Type	Controller: UCS C220/240 Compute: HP DL360 Gen9 Compute: Dell PowerEdge 740 or edge pod	UCS C240 M7	OVS/VLAN
Micropod	UCS 240 M5/M6/M7	Not applicable as it is integrated with compute and controller	OVS/VLAN or VPP/VLAN (on intel NIC, not on M6) OVS/VLAN for Quanta
C-series Hyperconverged	UCS 240 M5/M7	UCS C240 M5 (SFF), M7	OVS/VLAN

In addition, the Cisco Nexus 9372 or 93180YC, or 9396PX is also available to serve the NFVI ToR function. Newer versions of Nexus work as long as it supports virtual port channel and is compatible with the server NIC.

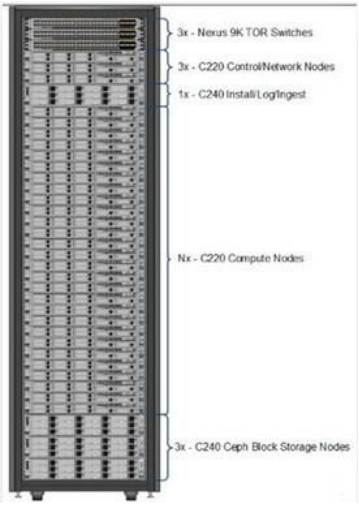
+

⚠ For BOM details, contact vim Product Management team : nfvi-plm@hcltech.com

The figure below shows a schematic C-Series NFVI pod. Although the figure shows a full complement of UCS C220 compute nodes, the number of compute nodes vary depending on the implementation requirements. The UCS C220 control and compute nodes can be replaced with UCS 240 series.

However, in that case the number of computes fitting in one chassis system is reduced by half. The following figure shows the NFVI C-series pod.

© 2026 HCLTech. All rights reserved.



For more information on wiring schematic of various pod configuration, see Wiring Diagrams.

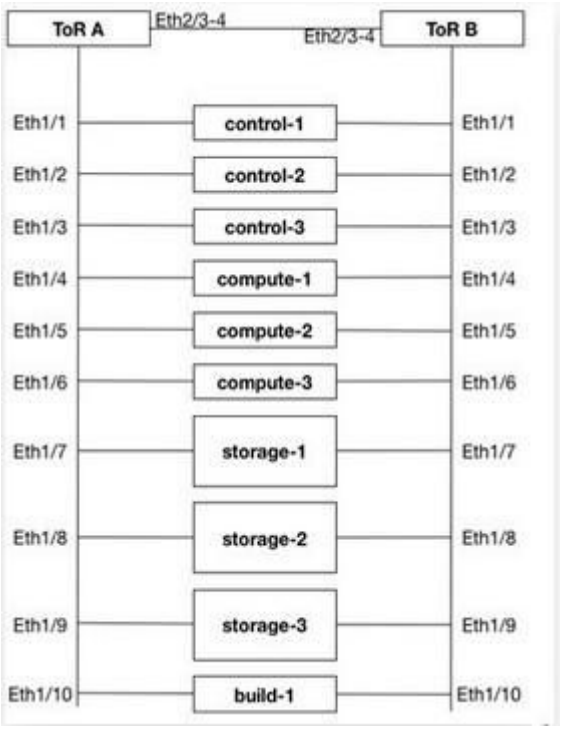
ToR Switch Configuration for C-Series Pods

ToR Switch Configuration for C-Series Pods

During installation, the VIM installer creates vNICs on each of the two physical interfaces and creates a bond for the Cisco UCS C-series pod. Before the installation, manually configure the ToR switches to create a vPC with the two interfaces connected to each server. Use identical Cisco Nexus 9372, 93180YC, or 9396PX switches for the ToRs. Cisco recommends you use the N9K ToR software versions for setup: 9.3(3). For information on the wiring

details for each pod type for a C-series-based installation, see Wiring Diagrams.

Complete the following steps to create a vPC on a pair of Cisco Nexus ToR switches. The steps use the following topology as an example. Modify the configuration as per your environment. The following figure shows a sample ToR configuration.



© 2026 HCLTech. All rights reserved.

VIM optionally supports the auto-configuration of ToR for N9K series only. If auto-configuration of ToR is enabled, you can skip the following steps:

Change the vPC domain ID for your configuration. The vPC domain ID can be a unique number. The IP address on the other switch mgmt0 port is used for the keepalive IP. Change it to the IP used for your network.

```
ToR-A (mgmt0 is 172.18.116.185)
feature vpc
vpc domain 116
peer-keepalive destination 172.18.116.186

ToR-B (mgmt0 is 172.18.116.186)
feature vpc
vpc domain 116
peer-keepalive destination 172.18.116.185
```

For the preceding example, the following is the configuration:

As both switches are cabled identically, the remaining configuration is identical for both switches. In this example, topology Eth2/3 and Eth2/4 are connected to each other and combined into a port channel that functions as the vPC peer link.

```
feature lacp
interface Ethernet2/3-4
channel-group 116 mode active
interface port-channel116
switchport mode trunk
vpc peer-link
```

For each VLAN type (mgmt_vlan, tenant_vlan_range, storage, api, external, provider), execute the following on each ToR:

```
vlan <vlan_type> no shut
```

Configure all the interfaces that are connected to the servers as the members of the port channels. In the example, only ten interfaces are shown. But you must configure all interfaces that are connected to the server.

> ⚠ If interfaces have configuration from previous deployments, you can remove them by entering default interface Eth1/1-10, then no interface Po1-10.

For deployment with any mechanism driver on Cisco VIC:

© 2026 HCLTech. All rights reserved.

There are no configuration differences among different roles (controllers/computes/storage). The same configuration applies to all interfaces.

```
interface Ethernet 1/1
channel-group 1 mode active
interface Ethernet 1/2
channel-group 2 mode active
interface Ethernet 1/3
channel-group 3 mode active
interface Ethernet 1/4
channel-group 4 mode active
interface Ethernet 1/5
channel-group 5 mode active
interface Ethernet 1/6
channel-group 6 mode active
interface Ethernet 1/7
channel-group 7 mode active
interface Ethernet 1/8
channel-group 8 mode active
interface Ethernet 1/9
channel-group 9 mode active
interface Ethernet 1/10
channel-group 10 mode active
```

For deployment with OVS/VPP with VLAN on Intel NIC:

The interface configuration is the same as the Cisco VIC as shown in the above section. However, the number of switch interfaces that are configured is more in the case of Intel NIC as it has dedicated control and data physical ports. For SRIOV switch port, no port channel is configured and the participating VLAN can be in trunk mode. In the case of pod-based on Quanta servers or HPE or DELL as computes, configure the control and data plane VLANs in trunk mode on the switch ports connected to the OCP and LOM cards, respectively.
Configure the port-channel interface as vPC and trunk all VLANs. For Intel NIC, you must configure native VLAN and set it to mgmt VLAN on the control ports so that PXE boot does not fail. Skip to listen or learn in spanning tree

```
interface port-channel 1-9
shutdown
spanning-tree port type edge trunk
spanning-tree bpdufilter enable
switchport mode trunk
switchport trunk native vlan mgmt_vlan for the control ports when Intel NIC is used
switchport trunk allowed vlan <mgmt_vlan, tenant_vlan_range, storage, api, external, provider>
no lacp suspend-individual
mtu 9216
vpc <1-9> no shutdown
```

transitions and ensure that you do not suspend the ports if LACP packets are not received. Also, configure it with a large MTU of 9216 to avoid Ceph installation failure. The last configuration allows you to start the servers before the bonding is set up.

Identify the port-channel interface that connects to the management node on the ToR:

© 2026 HCLTech. All rights reserved.

```
interface port-channel . .
shutdown
spanning-tree port type edge trunk
switchport mode trunk
switchport trunk allowed vlan <mgmt_vlan>
no lacp suspend-individual
vpc 10
no shutdown
```

Check the port-channel summary status. The ports connected to the neighbor switch have to be in *P* state. Before the server installation, the server facing interfaces must be in *I* state. After installation, they have to be in *P* state, which means they are up and in port-channel mode.

Enable automatic Cisco NX-OS *errdisable* state recovery:

```
errdisable recovery cause link-flap

errdisable recovery interval 30
```

Cisco NX-OS places links that flap repeatedly into *errdisable* state to prevent spanning tree convergence problems caused by non-functioning of hardware. During VIM installation, the server occasionally triggers the link flap threshold. Hence, enabling automatic recovery of this error is recommended.

```
errdisable recovery cause link-flap

errdisable recovery interval 30
```

Enable jumbo packets and configure 9216 MTU on the port-channel or Ethernet interfaces. For example:

```
Port channel:
interface port-channel10
switchport mode trunk
switchport trunk allowed vlan 80,323,680,860,2680,3122-3250
mtu 9216
vpc 10
Ethernet:
interface Ethernet1/25
switchport mode trunk
switchport trunk allowed vlan 80,323,680,860,2680,3122-3250
mtu 9216
```

© 2026 HCLTech. All rights reserved.

Preparing Cisco IMC

Preparing Cisco IMC

NFVI requires specific Cisco Integrated Management Controller (IMC) and firmware versions and parameters. The VIM bare metal installation uses the Cisco IMC credentials to access the Cisco IMC interface which is used to delete and create vNICS and to create bonds.

Complete the following steps to verify if Cisco IMC and UCS manager are ready for NFVI installation:

- 1. Verify that each Cisco UCS server uses Cisco IMC firmware version of either 2.0 series (2.0(13i) or greater preferably 2.0(13n)), or 3.0 series (use 3.0.3(f) or later). You can download the latest Cisco IMC ISO image from the Cisco Software Download site. For upgrade procedures, see the Cisco UCS C-Series Rack-Mount Server BIOS Upgrade Guide.
- For Cisco UCS C-series pods, verify whether the following Cisco IMC information is added: IP address, username, and password.
- Verify that no legacy DHCP/Cobbler/PXE servers are connected to your UCS servers. If so, disconnect or disable the interface connected to legacy DHCP, Cobbler, or PXE server. Also, delete the system from the legacy cobbler server.
- Verify if Cisco IMC has NTP enabled and is set to the same NTP server and same time zone as the operating system.

Management Node on UCS C-series (M5/M6/M7)

Installing Management Node on UCS C-series (M5/M6/M7)

This procedure installs RHEL with the following modifications:

- Hard disk drives are setup in RAID 6 configuration with one spare drive for deploying eight drives, two spare drives for deploying 9 to 16 drives, or four spare drives for deploying 17 to 24 drives.
- Networking: Two bridge interfaces are created; one for the installer API (br_api off the LOM interfaces) and the other for provisioning (br_mgmt off the Cisco VIC on the MLOM or off a X710 based Intel NIC depending on the BOM). Each bridge interface has underlying interfaces bonded together with 802.3ad. Provision interfaces are 10/40 GE interfaces (either off Cisco VICs or X710 Intel NIC (first 2 ports of Intel NIC)). API interfaces are 1/10 GE LOMs based on the BOM. For using NFVbench, you require another NIC card constituting off 2xIntel 520, or 2xIntel 710XL, or 4xIntel710 X. For management node BOM (Intel NIC based), ensure that you place the NIC for NFVbench at a slot higher than that of the br_mgmt based Intel NIC.
- The installer code is placed in */root/*.
- SELinux is enabled on the management node for security.

Before you Begin

- Verify whether the NFVI management node where you plan to install the Red Hat for Enterprise Linux (RHEL) operating system is a Cisco UCS C240 M5/M6 Small Form Factor (SFF) with 8, 16, or 24 hard disk (HDDs) or Solid State (SSDs) drives.
- Check if the management node is connected to your enterprise NTP and DNS servers. If your management node server does not meet these requirements, do not continue until you install a qualified Cisco UCS C240 server. Also, ensure that the pod contains Mobile Rich Media Ad Interface Definitions (MRAID) card.

- Log into the **CIMC GUI** of NFVI management node.
- Follow steps in Configuring the Server Boot Order to set the boot order to boot from local HDD.
- Follow steps in Cisco UCS Configure BIOS Parameters to set the following advanced BIOS settings:

© 2026 HCLTech. All rights reserved.

PCIe Slot:2 OptionROM—Enabled
PCIe Slot: MLOM OptionROM—Disabled
PCIe Slot:HBA OptionROM—Enabled
PCIe Slot:FrontPcie1 OptionROM—Enabled
PCIe Slot:MLOM Link Speed—GEN3
PCIe Slot:Riser1 Link Speed—GEN3
PCIe Slot:Riser2 Link Speed—GEN3
MLOM OptionROM—Enabled

For management node based on UCS M5 or M6 boxes, set the following for BIOS parameters:

All Onboard LOM Ports—Enabled
LOM Port 1 OptionROM—Disabled
LOM Port 2 OptionROM—Disabled
PCIe Slot:1 OptionROM—Enabled
PCIe Slot:2 OptionROM—Enabled
MLOM OptionROM—Enabled
MRAID OptionROM—Enabled

Other parameters must be set to default.

- Click Save Changes.
- Add the management node vNICs to the provisioning VLAN to provide the management node with access to the provisioning network:

– In the CIMC navigation area, click the **Server** tab and select **Inventory**.
– In the main window, click the **Cisco VIC Adapters > General Tab**, and then click **Reset to Default** tab.

⚠ Delete any additional vNICs configured on the UCS server beyond the two default ones.

- Download the ISO on a local server that can serve HTTP and use CIMC GUI to mount the ISO under *Cisco IMC Mapped vMedia*.
- In CIMC, launch the KVM console.
- Mount the VIM Build node ISO image as a virtual DVD.
- Reboot the UCS server, then press **F6** to enter the boot menu.
- Select the CIMC mapped DVD to boot the VIM Build node ISO image provided with the install artifacts.
- From the boot menu, select **Install VIM Management Node**. This is default selection, and it gets automatically selected after the timeout. At
- the prompts, enter values of the required parameters to install the Management node:

Select the mode this Management node will serve—Enter 1 for Standard Management node, 2 for Unified Management node, or 3 for Software Distribution System node.

- Hostname—Enter the management node hostname (The hostname length must be 32 or fewer characters).
- Select **Yes** to Install Unified Management only when required. Migration from one to another is not supported.
- API IPv4 address—Enter the management node API IPv4 address in CIDR (Classless Inter-Domain Routing) format. For example, 172.29.xxx.xxx/26
- API Gateway IPv4 address—Enter the API network default gateway IPv4 address.

© 2026 HCLTech. All rights reserved.

- MGMT IPv4 address—Enter the management node MGMT IPv4 address in CIDR format. For example, 10.30.xxx.xxx/26
- Prompt to enable static IPv6 address configuration—Enter **Yes** to continue input similar IPv6 address configuration

> ⚠ The MGMT IPv4 entry is not required if the management node is installed as unified management node only.

  for API and MGMT network, or **No** to skip if IPv6 is not needed.
- API IPv6 address—Enter the management node API IPv6 address in CIDR (Classless Inter-Domain Routing) format. For example, 2001: c5x0:xxxx:xxxx:xxxx::x/8.
- Gateway IPv6 address—Enter the API network default gateway IPv6 address.
- MGMT IPv6 address—Enter the management node MGMT IPv6 address in CIDR format. For example, 2001:c5c0:1234:xxxx:xxxx::x/80 • DNS server—Enter the DNS server IPv4 address or IPv6 address if static IPv6 address is enabled.
- Option for Teaming Driver for Link Aggregation <Yes/No> — Select **Yes** if Nexus switch is ToR, and **No** if Cisco NCS 5500 is ToR.
- Option for setting MGMT's Link Aggregation to active-active (802.3ad) mode <Yes|No> — Enter **Yes** for active-active mode or **No** for active-backup mode. Most commonly, the Active-active mode is the recommended mode. The active-backup mode is recommended only when CIMC is in in-band shared mode without using the dedicated CIMC mgmt port.

After you enter the management node IP addresses, the Installation options menu appears with several options. Fill in the options listed below (option 8 and 2) and leave the other fields as it is. If you are unable to start the installation, enter **r** to refresh the Installation menu.

- From the Installation menu, select option **8** to enter the root password.
- From the Installation menu, select option **2** to enter the time zone.
- Under the Timezone settings, select option **1,** as option **2** is not supported.
- Enter the number corresponding to your time zone.
- Enter the number for your region.
- Choose the city and then confirm the time zone settings.

> ⚠ NTP server IP must not be entered at the time of setting the time zone.

- After confirming your time zone settings, enter **b** to start the installation.
- After the installation is complete, press **Return** to reboot the server.
- After the reboot, check the management node clock using the Linux **date** command to ensure that the TLS certificates are valid, for example:

```
#date
 Mon Aug 22 05:36:39 PDT 2016
To set date:
#date -s '2016-08-21 22:40:00'
Sun Aug 21 22:40:00 PDT 2016
To check for date: #date
Sun Aug 21 22:40:02 PDT 2016
```

VIM Software Hub

Installing VIM Software Hub

- Prerequisites for VIM Software Hub Nodes
- Prerequisites for VIM Software Hub Server

© 2026 HCLTech. All rights reserved.

- Installing VIM Software Hub Node
- Setting up VIM Software Hub for VIM Artifact Distribution
- Installing VIM Software Hub in Connected Mode
- Installing VIM Software Hub in Air-Gapped Mode
- Installing Pod from VIM Software Hub Server
- Supported Day 2 Operations

VIM Software Hub alleviates the need for VIM management nodes to have internet connectivity and helps to remove the logistics of shipping USBs to multiple pods across the enterprise for software installation or update of the cloud.

vim Software Hub is also referred as Software Delivery Server; therefore, you might encounter references to SDS in the configuration files, directory paths, and automation outputs.

Prerequisites for VIM Software Hub Nodes

- Ensure that the VIM management nodes have connectivity to the VIM Software hub.
- Ensure that the VIM Software Hub node where you want to install the *buildnode.iso* file is VIM SDS BOM compliant.
- Ensure that the VIM Software Hub node is connected to the enterprise NTP and DNS servers.
- Ensure that the VIM Software Hub node contains hardware MRAID and a cache card.

Prerequisites for VIM Software Hub Server

TLS certificate (For production environment)

On the VIM Software Hub server, configure a secure registry so that the pods can obtain the container images over TLS. You need to provide a certificate signed by a trusted third-party CA authority and the **CommonName** in the certificate must match the VIM Software Hub Registry FQDN name. The certificate must contain a Subject Alternative Name with a DNS entry matching the VIM Software Hub Registry FQDN name. The *sds_setup_data.yaml* file has 3 fields:

- SSL_CERT_FILE: Path of x509 certificate obtained from a trusted CA authority.
- SSL_CERT_KEY_FILE: Path of private key obtained from a trusted CA authority.
- SSL_CERT_CHAIN_FILE: Path of a single ssl cert chain file. The trusted CA authority might provide you with the x509 cert for your domain, intermediate x509 cert, and root CA cert.

You need to create a single SSL cert chain file using the commands below:

```
cat <x509 domain cert> >> ssl_chain_file.cer
cat <intermediate ca cert> >> ssl_chain_file.cer # cat <root ca cert> >> ssl_chain_file.cer
```

Self-signed certificate (For internal use)

It is recommended to use a trusted CA-signed certificate, when a VIM Software Hub node is used in a production environment. For internal testing and POC, using self-signed certificate for VIM Software Hub node is agreeable.Use the sds_cert_gen.sh tool to generate the SSL_CERT_FILE, SSL_CERT_KEY_FILE and SSL_CERT_CHAIN_FILE in the /root/VIM_sds-<tag>/openstack-configs/ directory.

© 2026 HCLTech. All rights reserved.

For example:

```
# cd /root/cvim_sds-<tag>
# ./tools/sds_cert_gen.sh -n mysdsregistry.example.com INFO:
Creating CA certificate and key
Generating an RSA private key
. .....+++++
. .....+++++
writing new private key to 'openstack-configs/sds-ca.key'

INFO: Creating SDS registry key and certificate signing request Generating
an RSA private key
. ..... +++++
. .....+++++
writing new private key to 'openstack-configs/https_reverse_proxy.key'

INFO: Creating SDS registry certificate Signature ok
subject=CN = mysdsregistry.example.com
Getting CA Private Key
INFO: Created certificates and keys
INFO: SDS_REGISTRY_NAME: mysdsregistry.example.com
INFO: SSL_CERT_FILE: /root/cvim_sds-<tag>/openstack-configs/https_reverse_proxy.crt
INFO: SSL_CERT_KEY_FILE: /root/cvim_sds-<tag>/openstack-configs/https_reverse_proxy.key
INFO: SSL_CERT_CHAIN_FILE: /root/cvim_sds-<tag>/openstack-configs/sds-ca.crt
#
```

Generate the certificate with the same FQDN as specified in the sds_setup_data.yaml SDS_REGISTRY_NAME. Populate the SSL_CERT_FILE, SSL_CERT_KEY_FILE and SSL_CERT_CHAIN_FILE in sds_setup_data.yaml. In case of a self-signed certificate, you need to manually trust the self-signed certificate. The operator needs to execute the commands below on both VIM Software Hub server and VIM pod management node:

```
# cp <SSL_CERT_CHAIN_FILE> /etc/pki/ca-trust/source/anchors/ca.crt

# update-ca-trust extract
```

For docker registry to work with self-signed certificates, execute the commands below on the SDS server:

```
# mkdir /etc/docker/certs.d/<fqdn>
# cp <SSL_CERT_CHAIN_FILE> /etc/docker/certs.d/<fqdn>/ca.crt
```

DNS server:

Ensure that the DNS server can reach the pods and the VIM Software Hub server. The DNS server must be able to resolve the VIM Software Hub Registry FQDN. If the enterprise does not have a unified DNS, then you need to populate the */etc/hosts* file with FQDN after provisioning a node using the ISO archive file.

Installing VIM Software Hub Node

The steps to install a VIM Software Hub node are similar to the steps in Management Node on UCS C-series (M5/M6). The only difference being, in Step 11 of the task, you need to choose the option to configure the server as a VIM Software Hub server. In the subsequent prompts, you can enter information such as the hostname, ipv4 or ipv6

© 2026 HCLTech. All rights reserved.

addresses for br_public and br_private interfaces, and gateway addresses, similar to the Management Node on UCS C-series (M5/M6). From VIM 4.2.2, you will also be given the option to configure an ADMIN network, in which case you can enter ipv4 or ipv6 addressing information for a br_admin interface.

The node is installed with RHEL with the following modifications:

- Hard disk drives are set up in RAID 6 configuration with two spare HDDs for a 16 HDDs deployment or four spare HDDs for a 24 HDDs deployment.
- Two bridge interfaces are created, namely, br_public, and br_private. If the ADMIN network option is chosen, a third bridge interface br_admin is created. Each bridge interface has underlying interfaces bonded together with 802.3ad.
- In case of a connected VIM Software Hub server, the br_public interface is connected to the internet. The br_private interface is local to your datacenter. The management node for every VIM pod must be reachable to the br_private interface of VIM Software Hub server through the br_api interface. The br_admin interface is intended to provide ssh access to the Software Hub node for SDS administrators.
- For the VIM Software Hub without an ADMIN network, the private interfaces are over 10/25 GE Cisco VICs/ Intel NIC, while the public interfaces are 1 GE LOMs.
- VIM Software Hub with an ADMIN network requires a UCS C-series M5 with one quad-port Intel X710 NIC and the 2-port Intel LOM, with network connections as follows:
  - br_admin - LOM ports 1 and 2
  - br_private - X710 ports 1 and 2
  - br_public - X710 ports 3 and 4
- Security_Enhanced Linux (SELinux) is enabled on the management node for security.

The VIM Software Hub code consists of packages with installer code. After provisioning the server with ISO, the installer code is placed in the following path:

```
/root/hvim_sds-<tag>
```

Setting up VIM Software Hub for VIM Artifact Distribution

You must configure a *sds_setup_data.yaml* file for each installer workspace.

- Copy the example file from the *openstack-configs* directory and save it as *sds_setup_data.yaml*.
- If you want to install a release tag on a VIM Software Hub server, update the fields in the *sds_setup_data.yaml* file as required.

Configuration File:

This file is used as an inventory file to setup VIM SDS (software delivery server).
##

User Defined Configuration File.

Information in this file is specific to the SDS setup.
##

SSL_CERT_FILE: <abs_location_for_cert_path of x509 certificate>

SSL_CERT_KEY_FILE: <abs_location_for_cert_priv_key of x509 certificate> SSL_CERT_CHAIN_FILE:
<abs_location_for_cert_chain_file of x509 certificate>
##

Virtualized Infrastructure Manager Installation Guide, 3.0.0

94

Preparing for NFVI Installation Installing VIM Software Hub Node

Registry credentials to access the VIM registry (Cisco Supplied)
##

© 2026 HCLTech. All rights reserved.

CVIM_REGISTRY_NAME: <satellite.fqdn.com> # optional, needed to download artifacts from another SDS server; FQDN of the source SDS server

CVIM_REGISTRY_USERNAME: <username>
CVIM_REGISTRY_PASSWORD: <password>

NETWORKING:
Max. NTP servers = 4, min of 1
ntp_servers: <ntp.server1.fqdn.com, ntp.server2.fqdn.com >
or
ntp_servers: [ipv6_address, 'ipv4_address'] # "," separated IPv4 or IPv6 address info http_proxy_server: <proxy.domain.com:8080> # optional, needed if the pod is behind a proxy https_proxy_server: <proxy.domain.com:8080> # optional, needed if the pod is behind a proxy

SDS_REGISTRY_NAME: <satellite.fqdn.com> #SDS registry name needs to resolve to valid IP

```
SDS_REGISTRY_NAME: <username>
SDS_REGISTRY_PASSWORD: <password>

# (Optional)SDS users who can only pull images from SDS docker registry
SDS_READ_ONLY_USERS:
-username: <user1>
password: <password1>
-username: <user2>
password: <password2>
```

Save the sds_setup_data.yaml file in the following path:

```
openstack-configs
directory under /root/hvim_sds-<tag>
```

Installing VIM Software Hub in Connected Mode

In the connected mode, the VIM Software Hub server having a publicly routable IP address can connect to the VIM-registry or to another VIM Software Hub server having the target artifacts and the registry. When the VIM Software Hub server is initially configured with the ISO, VIM Software Hub workspace of that release is pre-installed in the */root/* directory.

- Download the *mercury-installer.tar.gz* file of the release that you want.
- Unzip the zip file manually and rename the unzipped file as *VIM_sds-<release>*.
- Perform the following steps:
  - Place a valid TLS certificate in the directory:

```
/root/hvim_sds-<tag>/openstack-configs
```

Update the fields of the VIM Software Hub setup data file and save it in the following directory:

```
/root/hvim_sds-<tag> openstack-configs
```

To install the release on the VIM Software Hub server, navigate to the following directory on the VIM Software Hub server:

© 2026 HCLTech. All rights reserved.

```
/root/hvim_sds-<target-tag>
```

Run the following command:

```
# cd to /root/hvim_sds-<target-tag>

# ./sds_runner/runner.py
```

The command validates the VIM Software Hub node hardware, the contents of the *sds_setup_data.yaml* file, and the validity of the TLS certificate, and then obtains the artifacts from the external VIM release registry and populates the VIM Software Hub server.

Installing VIM Software Hub in Air-Gapped Mode

VIM Software Hub is installed in the air-gapped mode when the VIM Software Hub server in the datacenter does not have internet connectivity. You can use the USB drive to load the installation files on the VIM Software Hub node. The installation files are over 25 GB in size. Downloading them to the USB drive may take several hours depending on the speed of your internet connection.

On the staging server, use yum to install the following packages:

- PyYAML
- python-requests

Before you begin

- Ensure that you have set up a CentOS/RHEL 9.2 staging server (VM, laptop, or UCS server) with a 64 GB USB 2.0 drive.
- Ensure that you have internet, preferably a wired connection, to download the vim installation files, which you want to load onto the USB drive.
- Ensure that you have disabled the CentOS sleep mode.

Check whether python 3.9.x version is installed. If not, perform *yum install python3*.

- Access VIM software download web site using a web browser.
- Log in with the credentials provided by your account representative and download the *getartifacts.py* script from the external registry.

```
# download the new getartifacts.py file

curl -o getartifacts.py -u '<username>:<password>' https://vim-registry.com/mercury-releases/hvim50-rhel9-osp17/releases/<releaseid>/getartifacts.py
```

Run the *getartifacts.py* script. The script formats the USB 2.0 drive (or USB 3.0 drive for M5-based management node) and downloads the installation files. You must provide the registry username and password, tag ID, and USB partition on the staging server.

© 2026 HCLTech. All rights reserved.

```
./getartifacts.py -h
usage: getartifacts.py [-h] -t TAG -u USERNAME -p PASSWORD [--proxy PROXY] [--retry] (-d DRIVE | -f
                       FILE)
                       [--mgmtk8s | --argus | --insight | --sds | -U]

Script to pull container images en masse

optional arguments:
  -h, --help            show this help message and exit Installer
  -t TAG, --tag TAG     version to pull
  -u USERNAME, --username USERNAME
                        Registry username
  -p PASSWORD, --password PASSWORD
                        Registry password
  --proxy PROXY         https_proxy if needed
  --retry               Try to complete a previous fetch
  -d DRIVE, --drive DRIVE
                                Provide usb drive path
                        -f FILE, --file FILE   location of image file
  --mgmtk8s             Additionally download hvim MON HA artifacts
  --argus               Additionally download argus artifacts
  --insight             Additionally download insight artifacts
  --sds                 Additionally download sds artifacts
  -U, --upgrade         Additionally download artifacts for upgrade from 2.4.x

This script pulls images from remote registry then copies the contents to USB drive or image file based on the
```

The *getartifacts.py* script gets the images from the remote registry and copies the contents to the USB drive.

- To identify the USB drive, execute the *lsblk* command before and after inserting the USB drive. The command displays a list of available block devices. You can use the output data to find the location of the USB drive. You must provide the entire drive path in the –d option instead of any partition.

For USB example:

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -d </dev/sdc> --sds [--proxy proxy.example.com]
```

For file-based image, follow the example below:

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -f /var/<artifacts-x-y-z.img> --sds [-- proxy proxy.example.com]
```

© 2026 HCLTech. All rights reserved.

Verify the integrity of the downloaded artifacts and container images:

```
# create a directory sudo mkdir -p /mnt/Cisco
# /dev/sdc is the USB drive, same as supplied in getartifacts.py python script sudo mount /dev/sdc1
/mnt/Cisco
cd /mnt/Cisco
# execute the test-usb help to look at the options
./test-usb -h

usage: ./test-usb [-h] -- Show this program to check integrity of artifacts in this USB drive/image file.
                  [-l] -- Location of artifacts
                 [-f] -- Location of image file

#./test-usb -f /var/artifacts_4-9-10.img
INFO: Checking the integrity of artifacts on this Image file
INFO: Checking artifact vim_upgrade_orchestrator.py
INFO: Checking artifact mercury-installer.tar.gz
INFO: Checking artifact buildnode-K9-33701.iso
INFO: Checking artifact registry-2.6.2-33701.tar.gz
INFO: Checking required layers:
INFO: 767 layer files passed checksum.
```

© 2026 HCLTech. All rights reserved.

To resolve failure in downloading artifacts, unmount the USB and run the *getartifacts* command again with the *--retry* option.

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -d </dev/sdc> --sds --retry
```

Mount the USB and then run the *test-usb* command to validate if all the files are downloaded.

```
# /dev/sdc is the USB drive, same as supplied in getartifacts.py python script sudo mount /dev/sda1 /mnt/Cisco
cd /mnt/Cisco
```

```
# ./test-usb
# In case of failures the out of the command displays a message indicating the same on the screen.
```

Execute the verification script:

For file-based image, execute the verification script:

```
# ./test-usb -f /var/<artifacts-x-y-z.img>
# In case of failures the out of the command displays a message indicating the same on the screen
```

When the USB integrity test is complete, unmount the USB using the below command:

```
sudo umount /mnt/Cisco
```

After the artifacts of a target release are saved on the USB, you must unplug the USB from the staging server, connect it to the VIM.

Software Hub server, and then perform the following steps on the VIM Software Hub server:

- Provision your VIM Software Hub server with the build node ISO of that release and then connect the USB to the VIM Software Hub server.
- To copy the contents of the USB to the VIM Software Hub server, navigate to the */root/VIM_sds-<tag>* directory, and then execute the import artifacts command.

```
#cd~/hvim_sds-<tag>/tools # ./import_artifacts.sh -s
```

© 2026 HCLTech. All rights reserved.

To copy the contents off the file-based image to the VIM Software Hub server, navigate to the /root/VIM_sds-<tag> directory, and then execute the import artifacts command.

```
# cd ~/hvim_sds-<tag>/tools
# ./import_artifacts.sh -s -f /var/<artifacts-x-y-z.img>
```

- Place a valid TLS certificate in /root/hvim_sds-<tag>/openstack-configs directory.
- Configure the VIM Software Hub setup data file with all the fields and placed the file in the /root/hvim_sds-<tag>/openstack-configs

directory.

Install the release on the VIM Software Hub server.

Navigate to the hvim_sds directory on the VIM Software Hub server and execute the following command:

```
# cd /root/hvim_sds-<tag>
# ./sds_runner/runner.py
Usage: runner.py [options]
Runner
Options:
-h, --help show this help message and exit
-l, --list_steps List steps
-s SKIP_STEPS, --skip_steps=SKIP_STEPS
Comma separated list of steps to skip. eg -s 2,3
-p PERFORM_STEPS, --perform=PERFORM_STEPS
-y, --yes Yes option to skip steps without prompt
```

Installing Pod from VIM Software Hub Server

When you want to install a VIM pod using the artifacts obtained from the VIM Software Hub server, you need to provide an additional parameter in *setup_data.yaml*. Ensure that the release artifacts are pre-installed on the VIM Software Hub server and that the *setup_data.yaml* file is populated with the pod details. Provide the registry FQDN name for install through VIM Software Hub. For example, your domain.com.

```
REGISTRY_NAME: '<registry_name>' # Mandatory Parameter.
```

VIM pod *setup_data.yaml* requires the REGISTRY_USERNAME and REGISTRY_PASSWORD to connect to the docker registry and fetch docker images. To fetch the docker images from VIM Software Hub node, provide the user credentials available in the SDS_READ_ONLY_USERS section of *sds_setup_data.yaml*. The details of an admin user with read/write access to docker registry are provided in SDS_REGISTRY_USERNAME and SDS_REGISTRY_PASSWORD field. So, it is recommended to have a read-only user on VIM pod.

© 2026 HCLTech. All rights reserved.

The VIM management node must have connectivity to the organization's DNS server to resolve the VIM Software Hub server domain.

> ⚠ The vim management node must have connectivity to the organization's DNS server to resolve the vim Software Hub server domain.

Supported Day 2 Operations

The following Day 2 operations are supported on the VIM Software Hub server:

- Reconfiguration of VIM Software Hub TLS certificate and VIM Software Hub registry credentials. VIM
- Software Hub server backup and restore.
- Execution of registry cleanup script.
- Manual update of few packages in the maintenance window.

 After Regenerating the VIM Software Hub TLS certificate, kindly restart the httpd service.systemctl reload httpd

UCS C-Series Pod

Setting Up UCS C-Series Pod

- Procedure
- Utility Details

Procedure

After you install the RHEL OS on the management node, perform the following steps to set up the Cisco UCS C-series servers:

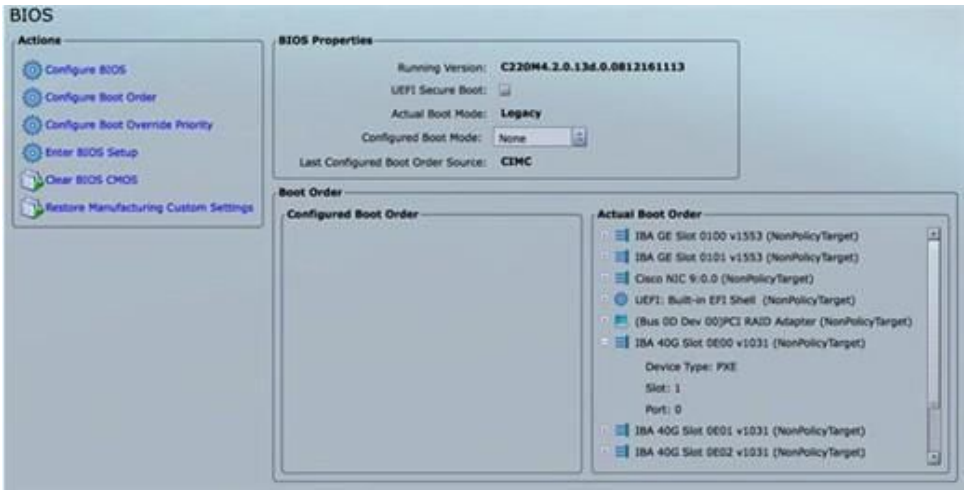
- Log into CIMC GUI of NFVI management node.
- Follow steps in Configuring the Server Boot Order to set the boot order to boot from Local HDD.
- Follow steps in Configure BIOS Parameters to set the LOM, HBA, and PCIe slots to the following settings:

For servers based on UCS M5 or M6 boxes, set the following for BIOS parameters:

- All Onboard LOM Ports—Enabled
- LOM Port 1 OptionROM—Disabled
- LOM Port 2 OptionROM—Disabled
- PCIe Slot:1 OptionROM—Enabled
- PCIe Slot:2 OptionROM—Enabled
- MLOM OptionROM—Enabled
- MRAID OptionROM—Enabled
- IPv4 HTTP Support - Disabled (M6)
- IPv6 HTTP Support - Disabled (M6)

Other parameters must be set to their default values.

© 2026 HCLTech. All rights reserved.



For UCS M5, look for **IBA 40G Slot** under the **BIOS Properties,**



While the pxe-boot tool simplifies the job of flashing the intel NIC cards, the restrictions of COSI compliance prevent us from shipping third-party utility (from CIMC 4.0 onwards). Administrators must download the PREBOOT.exe file from Intel website: https://downloadcent er.intel.com/download/27539/Ethernet-Intel-Ethernet-Connections-Boot-Utility-Preboot-Images-and-EFI-Drivers

Version: 22.10

Date: 12/7/2017 OS Independent

Language: English

Size: 16.54 MB

MD5: ace485e8a3ef9039212f52b636ce48e3 PREBOOT.EXE

Ensure that there is unrestricted network access from VIM Management node to UCS-C series server's CIMC over following ports:

- TCP/2400 - serial-over-lan (SOL)
- TCP/22 - XMLAPI

© 2026 HCLTech. All rights reserved.

Ensure that there is unrestricted network access from UCS-C series server's CIMC to VIM Management node's API interface over following port:

- TCP/80 - HTTP

This utility updates only the Intel PXE configuration and not the card's firmware or Option ROM.

Utility Details

Two scripts available in the VIM Installer's tools directory are:
- reate-bootutil-img.sh
- intel-bootutil-update.py

```
[root@cologne-mgmt tools]# ./create-bootutil-img.sh
Usage: ./create-bootutil-img.sh <PREBOOT.exe file> <output image name>
```

Usage

You can download PREBOOT.exe file from :

https://downloadcenter.intel.com/download/27862/Ethernet-Intel-Ethernet-Connections-Boot-Utility-Preboot-Images-and-EFI-Drivers

- Version: 22.1
- Date: 12/07/2017 OS Independent
- Language: English
- Size: 16.54 MB
- MD5: ace485e8a3ef9039212f52b636ce48e3 PREBOOT.EXE
- To toggle Intel PXE configuration on UCS C-series, use the script below:

```
[root@cologne-mgmt tools]# ./intel-bootutil-update.py -h
usage: intel-bootutil-update.py [-h] [--hosts HOSTS]
[--exclude-hosts EXCLUDE_HOSTS] [-v] [-y]
--setupfile SETUPFILE --bootutil-image BOOTUTIL_IMAGE --port {0,1,2,3} --state
{enable,disable}
```

Details of the optional arguments for the *intel-bootutil-update.py* script are given below:

Optional Arguments	Description
-h --help	Displays the help with details of each argument of the script.
--hosts HOSTS	Displays the list of servers for PXE configuration from the *setup_data.yaml* file.
--exclude-hosts EXCLUDE_HOSTS	Displays the list of servers to be excluded for PXE configuration from the *setup_data.yaml* file.
-v, --verbose	Enables verbose output.
--setupfile SETUPFILE	Specifies the *setup_data.yaml* file location.
--bootutil-image BOOTUTIL_IMAGE	Specifies the BootUtil image location.
--port {0,1,2,3} port #	Specifies the port to be enabled.

© 2026 HCLTech. All rights reserved.

Optional Arguments	Description
--state {enable,disable}	Enables or disables the PXE configuration.

```
/intel-bootutil-update.py --setupfile /root/openstack-configs/setup_data.yaml
--bootutil-image /root/bootutil.img --port 0 --state enable
```

Example to enable all port A:

```
/intel-bootutil-update.py --setupfile /root/openstack-configs/setup_data.yaml
--bootutil-image /root/bootutil.img --port 0 --state enable
```

Example to enable all port A and B:

```
./intel-bootutil-update.py --setupfile /root/openstack-configs/setup_data.yaml
--bootutil-image /root/bootutil.img --port 0 --port 1 --state enable
```

```
./intel-bootutil-update.py --setupfile /root/openstack-configs/setup_data.yaml
--bootutil-image /root/bootutil.img --port 2 --state disable
```

Example to disable all port C:

Workflow:

Multiple scripts are required as Intel's PREBOOT.exe utility is not packaged with VIM for COSI compliance:

- Download the PREBOOT.exe version (listed above) from Intel's website.
- Go to VIM Installer's tools directory.
- Run *create-bootutil.img* script to create a CIMC-KVM mountable USB image.
- Run *intel-bootutil-update.py* script to configure Intel NIC for enabling or disabling PXE.

Utility Examples:

```
[root@cologne-mgmt installer]# cd tools
[root@cologne-mgmt tools]#
[root@cologne-mgmt tools]# ./create-bootutil-img.sh
```

Usage: ./create-bootutil-img.sh <PREBOOT.exe file> <output image name>

```
[root@cologne-mgmt tools]#
[root@cologne-mgmt tools]# ./create-bootutil-img.sh /root/PREBOOT.exe /root/bootutil.img
...
```

© 2026 HCLTech. All rights reserved.

Unmounting temporary mount point /tmp/tmp_bootutil.img

Cleaning up temporary workspaces

Successfully created image file with BOOTUTIL64E.EFI

-rw-r--r--. 1 root root 5.0M Jul 20 17:52 /root/bootutil.img

[root@cologne-mgmt tools]#

[root@cologne-mgmt tools]# ./intel-bootutil-update.py --setupfile /root/openstack-configs/setup_data.yaml –

bootutil-image /root/bootutil.img --port 0 --state enable

All servers will be rebooted as part of PXE configuration, would you like to continue? <y|n> y 2018-07-18 18:34:36,697 INFO Enabling temporary HTTP server hosting BootUtil.img on 172.29.86.10
2018-07-18 18:34:36,790 INFO Successfully enabled temporary HTTP server hosting BootUtil.img on 172.29.86.10

...

2018-07-18 18:40:28,711 INFO Disabling temporary HTTP server hosting BootUtil.img on 172.29.86.10
2018-07-18 18:40:28,810 INFO Successfully disabled temporary HTTP server hosting BootUtil.img on 172.29.86.10
Server(s) successfully updated PXE configuration:

cologne-control-1,cologne-control-3,cologne-control-2,cologne-compute-1,cologne-compute-2,cologne-storage-1,

cologne-storage-3,cologne-storage-2

[root@cologne-mgmt tools]#

Out-of-Band Management Switch

Configuring Out-of-Band Management Switch

For VIM installer API and SSH bonded interface, use 1-GB Intel NICs that connect the NFVI management node and Cisco Catalyst switch. Following is a sample configuration for creating a port channel on a Catalyst switch. Modify the configuration for your environment:

```
interface GigabitEthernet0/39
channel-group 2 mode active speed 1000

interface GigabitEthernet0/40
channel-group 2 mode active speed 1000

interface Port-channel2 switchport
access vlan 165 switchport mode access
```

© 2026 HCLTech. All rights reserved.

VIM Monitor

VIM Monitor Configurations

- Overview of VIM Monitor
- Enabling VIM-MON on VIM
- Disabling VIM-MON on VIM Monitoring
- External Servers
- Using VIM-MON
  - Assumptions for Monitoring External Servers Using VIM-MON
  - Installation Procedure for Monitoring External Servers Using VIM-MON Local
  - Uninstallation Procedure for Monitoring External Servers Using VIM-MON Local
  - Installation Procedure for Monitoring External Servers Using VIM-MON Central
  - Uninstallation Procedure for Monitoring External Servers Using VIM-MON Central
- Enabling VIM-MON Post Pod Installation
- VIM-MON Grafana/Prometheus and AlertManager with LDAP Backend
- VIM-MON Dashboard Customization Support (Local)

Overview of VIM Monitor

The VIM Monitor feature (VIM-MON) provides a comprehensive solution for monitoring the health and for tracking the usage of resources in the VIM.

- pod infrastructure. This solution is available as a configuration option and provides the following services:
- Infrastructure-level metric collection from all nodes in the pod
- Metric aggregation into a time-series database (TSDB)

- Rule-based alerting engine integrated with the TSDB
- Web UI with pre-defined dashboards customized for VIM REST API to query the TSDB
- REST API to query and silence alerts
- Alert notifications using SNMP traps or alternate alert notification protocols
- User-configurable alerting rules
- User-configurable web UI dashboards

The software components that provide the VIM-MON service is called a VIM-MON stack. The default deployment mode for VIM-MON is to deploy the

VIM-MON stack on the pod management node.

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.

Enabling VIM-MON on VIM

You can enable VIM-MON on an existing pod that is installed with VIM 3.0.0 or later, through the reconfigure option by extending the *setup_data. Yaml* file with relevant information.

The components of VIM-MON are as follows:

- CVIM_MON: Provides the base functionality of monitoring and KPIs.
- SNMP: Enables sending VIM-MON alerts as SNMP traps, only if VIM-MON is enabled.
- SERVER_MON: Enables collecting UCS-C bare metal alerts into VIM-MON for Cisco UCS C-series servers.
- MANAGEMENT_SERVERS: Enables collecting UCS-C bare metal alerts of external UCS servers, for example, management node.
- SERVER_MON is a prerequisite for this option.

© 2026 HCLTech. All rights reserved.

To enable VIM-MON, the C**VIM_MON** and **PODNAME** keys must be added to the *setup_data.yaml* file. The CVIM_MON key has the following properties:

- enabled: A boolean value indicating whether VIM-MON is enabled.
- polling_intervals: A dictionary defining the interval of metrics sampling. If not defined, default values are used.
- ui_access: A boolean indicating whether VIM-MON UI access is enabled or not. By default, it is enabled.

PODNAME is mandatory for VIM-MON and is a name that identifies uniquely each VIM pod. SNMP traps can be enabled using the SNMP key with the following attributes:

- enabled: A boolean value indicating whether SNMP is enabled. CVIM_MON must also be enabled.
- managers: A list of SNMPv2 or SNMPv3 managers to send the SNMP traps. For SNMPv2, community and port can be set. For SNMPv3, the *Engi ne_id* and list of users must be specified, where the *Engine_id* is the EngineContextID used to send trap of the SNMP manager.

⚠ SNMP traps are sent without setting any authentication or secure *engine_id* for the user.

The following table shows the list of properties with their default values and description:

Property Group and Name	Values	Default Value	Description
PODNAME:	<string>	(required)	Must be provided for identifying each pod when CVIM_MON is enabled.
CVIM_MON: enabled	true|false	false	A boolean indicating C VIM-MON is enabled or not. Set to True to enable CVIM_MON.
CVIM_MON: ui_access	true|false	true	A boolean indicating whether CVIM-MON UI access is enabled or not.
CVIM_MON: external_ser vers	List of external server IPs (v4 or v6) that must be monitored by VIM MON	-	Optional. For more information, see Monitoring External Servers Using VIM-MON
CVIM_MON: polling_interv a l s : high_freque ncy:	10s to 10m	15s	Metric collection sampling interval in seconds or minutes. It is recommended to set to 1m in production deployments.
SNMP: enabled	true|false	false	A Boolean indicating whether VIM-Trap is enabled or not. If true, CVIM_MON:enabled must also be set to true.
SNMP: managers:	-	-	A list of up to three SNMP managers to send traps.
address	<ipv4 or ipv6>	(required)	Denotes the IPv4 or IPv6 address of the SNMP manager.
port	1-65535	162	Optional port to send traps.

© 2026 HCLTech. All rights reserved.

Property Group and Name	Values	Default Value	Description
version	v2c|v3	v2c	Indicates the SNMP manager version.
community	<string>	public	Used for SNMPv2c.
SNMP: managers: users:			Required for SNMPv3, up to three users.
engine_id	<hexadecimal string>	(required v3)	Uniquely identifies the SNMP engines and entities. The SNMP engine IDs are composed of 5 to 12 octets and have no standard display format. It must be a hexadecimal string and cannot have all zeros or all 255s ("ff"). RFC 3411 specifies that the engine can be formatted with IPv4, IPv6, MAC address, text, and octets (and must start with 8). Listed below is a summary of the constraints. ContextEngineId is unique across all managers. Minimum length is 5 and maximum length is 32. All cannot be 00s or FFs and must start with 80.
name	<string>	(required v3)	Indicates the username.
auth_key	<string>	(required v3)	Indicates the authorization password. Must contain at least eight characters.
authenticati on	SHA|MD5	SHA	Specifies the authentication protocol.
privacy_key	<str>	(auth_key)	Encryption key
encryption	'AES128'|'AES192'|'AES 256'	'AES128'	Encryption protocol
SERVER_M ON: enabled	true|false	false	Enable SNMP traps for CIMC faults (UCS C-series only)
host_info:	'ALL' or list of servers specified in SERVERS and MANAGEMENT_SERVE RS section	'ALL'	Specifies the UCS-C servers to be monitored.
rsyslog_sev erity	emergency | alert| critical | error| warning| notice| informational | debug	(Optional)	Specifies the minimum severity from the UCS C-server logs that are to be sent to remote syslog servers
MANAGEM ENT_SERV ERS:	server list	(Optional)	Enables collection of UCS-C bare metal alerts of external UCS servers, for example, management node. See example below on the entry details for this key

© 2026 HCLTech. All rights reserved.

> ⚠ If SERVER_MON.rsyslog_severity is configured, you must configure SYSLOG_EXPORT_SETTINGS as well to indicate the remote syslog servers to send the logs.

Example of *setup_data.yaml* section related to VIM-MON (not all possible optional values are represented):

```
CVIM_MON:
  enabled: true
  polling_intervals:
     high_frequency: 1m

SNMP:
 enabled: true
 managers:
 - address: 10.10.10.54

SERVER_MON:
  enabled: true
  host_info:
     - ALL
  rsyslog_severity: error

MANAGEMENT_SERVERS:  # Optional for monitoring CIMC of management node or external servers via SNMP (one or
more external servers); SERVER_MON must be enabled for this option
  <hostname>:
    cimc_info: {cimc_ip: <ip_address>, [cimc_username: <uname>, cimc_password: <password>]} #cimc_username and
cimc_password defined in CIMC-COMMON are used, if not
                                                                                    defined locally
  <hostname_2>:
    cimc_info: {cimc_ip: <ip_address>}
  :
  :
```

Disabling VIM-MON on VIM

Starting from VIM 4.4.4, the administrator can disable the VIM-MON after being enabled in the existing VIM pod via "VIM reconfigure".

> ⓘ #cvim_mon:
> # enabled: <True or False> # default is False (case-sensitive). Set True to enable hvim-MON. Set False to disable hvim-MON.
>
> > ⚠ cvim_mon cannot be disabled if SNMP and SERVER_MON is enabled. Set False SNMP and SERVER_MON and cvim_mon to disable vim-mon.

Monitoring External Servers Using VIM-MON

VIM-MON can monitor external non-VIM servers running RHEL or CentOS 8.2. VIM-MON monitors these servers by installing a Telegraf agent on them to collect bare metal and libvirt metrics. The telegraf agent runs a new systemctl service and collects metrics from the local server. These metrics are available for a remote scraper on port 9273.

Assumptions for Monitoring External Servers Using VIM-MON

The following are the assumptions and prerequisites associated with this feature:

- The external servers must be reachable from the management node from the local or central VIM-MON deployments.
- The external servers must run on UCS M5 hardware similar to the VIM management node BOM.
- The external servers must run VIM management node ISO or CentOS 8.2.
- The external nodes must run on the same site as the monitoring VIM pod. The scraping of metrics occurs over unauthenticated and unencrypted HTTP connections on port 9273.

© 2026 HCLTech. All rights reserved.

- VIM artifacts of Unified Management and VIM software hub can be monitored via VIMMON local or central installation.

Installation Procedure for Monitoring External Servers Using VIM-MON Local

To enable monitoring of external servers using VIM-MON, you must update the *setup_data.yaml* file with the details of the external_servers and run fresh installation or reconfiguration in the VIM pod. As part of the fresh installation or reconfiguration, an *external-monitoring-<telegraf version>.tar.gz* tar file is created in the management node at */opt/cisco/VIM_mon*.

An example of a setup data file with VIM-MON external server monitoring capabilities:

```
CVIM_MON:
        enabled: true
        external_servers: [<ipv4_address>, <ipv6_address>]

Note: IPv6 address is only supported when CVIM is running IPv6
```

To enable monitoring of the external servers using VIM-MON, you must execute the following steps on each of the target external servers:

- Copy the *external-monitoring-<telegraf version>.tar.gz* file from the management node of the VIM pod to root directory in the external server.
- Open port 9273 on the respective external server (as per the IP defined in the VIM pod) to allow communication between Prometheus on the management node and the Telegraf agent on the server.
- Enter the *external-monitoring-<telegraf version>.tar.gz* tar file to extract the following files:

File	Description
external.conf	Provides the Telegraf configuration.
telegraf<version>.x86_64.rpm	Installs Telegraf on the server.
monitor_external.sh	Deploys Telegraf on the external server.
monitor_external_unbootstrap.sh	Removes Telegraf agent and corresponding configurations on the external server
ext_docker.conf	Docker container Telegraf plugin.
ext_ipmi.conf	IPMI Telegraf plugin
ext_libvirt.conf	Libvirt VM Telegraf plugin.
um_x509_cert.conf	Unified Management node x509_cert Telegraf plugin.
um_http_response.conf	Unified Management node HTTP response Telegraf plugin.
sds_http_response.conf	Software Delivery Server HTTP response Telegraf plugin
sds_x509_cert.conf	Software Delivery Server x509_cert Telegraf plugin

To deploy Telegraf on the external server, run the *monitor_external bash* script in privileged mode. This script will be found in the directory *(extern al-monitoring-xxx)* created as a result of untarring *external-monitoring-<telegraf version>.tar.gz* in the external server targeted for monitoring.

```
#sudo ./monitor_external.sh
```

© 2026 HCLTech. All rights reserved.

If the script runs successfully, **Success: Telegraf installation succeeded** message appears on the console screen. You can now access either the local or central Prometheus or Grafana and monitor the external server.

Uninstallation Procedure for Monitoring External Servers Using VIM-MON Local

To remove the Telegraf agent and related configuration on the external server, go to the directory *(external-monitoring-xxx)* created as a result of untarring *external-monitoring-<telegraf version>.tar.gz* in the external server targeted for monitoring, and execute the following:

⚠ Do not forget to take out the corresponding IP Address from the *external_servers'* information in cvim_mon section of the pod and run a reconfigure operation to avoid false alerts.

Installation Procedure for Monitoring External Servers Using VIM-MON Central

Obtain the *external-monitoring-<telegraf version>.tar.gz* from a VIM pod using steps listed in *Installation Procedure for Monitoring External Servers Using VIM-MON local*. If you wish to monitor these servers independent of VIM installation, update the *setup_data* of VIM MON Central as listed below and initiate an installation or perform add-VIM-pod operation of VIMMON HA

```
cvim-mon-stacks:
  regions:
  - metros:
    - name: I
      pods:
      ....
      - ip: <target_ip_address_1>:9273
        name: <name_1>
        target_type: external
      - ip: <target_ip_address_2>:9273
        name: <name_2>
        target_type: external

Note: IPv6 address is only supported when CVIM is running IPv6
```

To deploy the Telegraf plugin into the external servers, follow the steps listed in *Installation Procedure for Monitoring External Servers Using VIM-MON Local*.

Uninstallation Procedure for Monitoring External Servers Using VIM-MON Central

To remove the Telegraf agent and related configuration on the external server, go to the directory *(external-monitoring-xxx)* created as a result of untarring *external-monitoring-<telegraf version>.tar.gz* in the external server targeted for monitoring, and execute the following:

```
#sudo ./monitor_external_unbootstrap.sh
```

⚠ Ensure that you take out the corresponding *external_servers'* information from the VIM_MON_CENTRAL *setup_data* and then run the *delete* operation to avoid false alerts. For details on how to invoke the delete operation, see Stack Operations

Enabling VIM-MON Post Pod Installation

⚠ After you enable the VIM-MON or SNMP TRAP, it cannot be disabled again.

© 2026 HCLTech. All rights reserved.

You can enable VIM-MON, SNMP traps and UCS-C bare metal alerts (SNMP, SERVER_MON) using the reconfigure option, post installation of VIM.

To enable VIM-MON and SNMP trap features or to change the individual parameters in VIM-MON, SNMP, SERVER_MON, and MANAGEMENT_SERVERS:

Take a backup of setup_data file and update it manually with the configuration details by entering the following command:

```
# cd /root/
# mkdir MyDir
# cp /root/openstack-configs/setup_data.yaml /root/MyDir
# cd /root/MyDir
```

- Edit the setup data.
- Save the file and execute the below command:

```
hvim --setupfile /root/MyDir/setup_data.yaml reconfigure
```

> ⚠ The migration from SNMPv2 to SNMPv3 is only supported, but not vice-versa.

IM-MON Grafana/Prometheus and AlertManager with LDAP Backend

The VIM-MON LDAP feature allows you to login with LDAP credentials. You can enable this feature by configuring the connection to the LDAP server and setting a valid filter to access Grafana/Prometheus and AlertManager with your LDAP credentials. Once the filter is set, you can map the user groups with specific roles of permission in Grafana.

VIM-MON Grafana supports the roles of:

- Viewer: Can only view dashboards and cannot modify them.
- Editor: Can view, create, copy, modify and save dashboards.

For Prometheus and AlertManager, users belonging to group mapped with *Admin* org_role have access.

> ⚠
> - If you are using FQDN as *ldap_uri*, the management node must be able to resolve it.
> - Only one LDAP server is allowed for authentication in Kibana.

To enable LDAP, you must modify the *setup_data.yaml* file by adding a *ldap* section under the VIM_MON section as following (replace example values as appropriate):

© 2026 HCLTech. All rights reserved.

```
CVIM_MON:

  central: false

  enabled: true

  ldap:

    domain_mappings:
    - attributes: {email: email, name: givenName, surname: sn, username: uid}
      bind_dn: <bind_dn>
      bind_password: <bind_password>
      domain_name: <domain_name>
      group_search_base_dns: ['ou=Groups,dc=org,dc=com']
      group_search_filter: (&(objectClass=posixGroup)(memberUid=%s))
      group_search_filter_user_attribute: uid
      ldap_uri: ldaps://<ldap_ip/ldap_fqdn>
          group_attribute: <group_attribute>
      group_attribute_is_dn: true/false
      root_ca_cert: <path_to_root_ca_cert>
          search_base_dns: ['dc=org,dc=com']
      search_filter: (uid=%s)
      use_ssl: true/false
      ldap_user_name: sAMAccountName # mandatory for AD

    group_mappings:
    - {group_dn: 'cn=group2,ou=Groups,dc=org,dc=com', org_role: Admin}
    - {group_dn: 'cn=group3,ou=Groups,dc=org,dc=com', org_role: Viewer}

  polling_intervals: {high_frequency: 10s, low_frequency: 2m, medium_frequency: 1m}

  ui_access: true
```

Property	Field Required	Description
search_filter	Mandatory	Filter set for the queries.
search_base_dns	Mandatory	It is the base DNS name used for all queries.
ldap_uri	Mandatory	Default port is 389. Takes the value 636, if *use_ssl* = True and port is not defined.
group_mappings	Mandatory	Must contain at least one group with org_role *Admin*. Optionally, you can add a second group with org_role *Vie wer*. You can add multiple LDAP groups mapped to org_role *Admin* or *Viewer*.
domain_name	Mandatory	Any non-empty name is acceptable.
domain_mappings	Mandatory	Must contain one domain exactly.
bind_password	Conditional	Mandatory, if LDAP supports binding. Not required for anonymous bind.
bind_dn	Optional	Mandatory, if LDAP does not support anonymous bind.
attributes	Optional	Mandatory key, but individual attributes are optional.
use_ssl	Optional	Optional. If not provided, defaults to False.
start_tls	Optional	Optional. If not provided, defaults to False.
client_cert	Optional	Authentication against LDAP servers requiring client certificates.
client_key	Optional	Authentication against LDAP servers requiring client certificates.

© 2026 HCLTech. All rights reserved.

Property	Field Required	Description
root_ca_cert	Optional	Path to your root CA certificate.
ldap_user_name	Optional	Mandatory, if MS AD is configured.
group_search_filter	Optional	To search for group members.
group_search_base_dns	Optional	Base DN to search groups.
group_search_filter_user_attribute	Optional	Indicates the distinguished name of the client username.
group_attribute_is_dn	Optional	Default is True. If set to True, the distinguished name of the client username is used for checking group membership, otherwise, the client username is used.
group_attribute	Optional	Default value is *memberUid*.

For monitoring details, see VIM-MON

VIM-MON Dashboard Customization Support (Local)

With VIM-MON, new dashboards can be created and modified. But to persist them, it is necessary to do some procedures that allows not only persist but also to maintain those dashboards json files in a private repo. The operations that are supported is basically new custom dashboard creation and custom dashboards edition. Once the user created new dashboards or makes changes, he will be able to save it in grafana and persist with it using cli command in the host. Modifications on built-in dashboards will not be persisted, but they can also be treated as custom dashboards that means the user can export them, import as a new dashboard, and makes the changes they want before persist them.

Following options are supported via VIM cli for LOCAL VIM MON.

© 2026 HCLTech. All rights reserved.

```
hvim help cvimmon-dashboard


usage: hvim cvimmon-dashboard [--dry-run] [-p] [--dir-path <dir_path>]
                [-f] [-y]
                <save|list|upload>

Execute operations for vim Mon Dashboard

Positional arguments:
 <save|list|upload>       The control command to perform

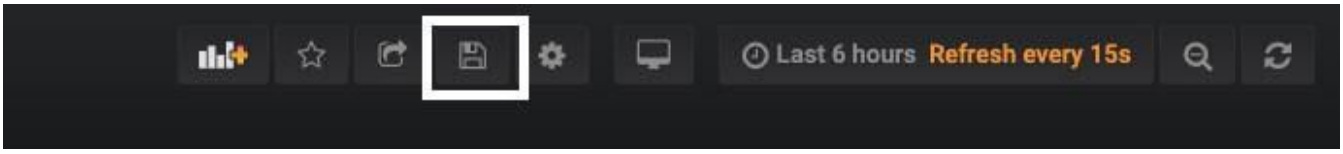
Optional arguments:
 --dry-run                Dry Run to find changes that will happen on Grafana Dashboard
 -p, --preserve           Only works with upload option. If passed all existing dashboards will be preserved with new dashboards.
 --dir-path <dir_path>  Dir where custom dashbaords are located. Mandatory for upload operation and optional for save op.
 -f, --force              Force option to delete custom dashboards with upload or save op.
 -y, --yes                Yes, option to perform the action
```

Persistence Workflow (Syncing Grafana Server with Management Node Repo) Supported arguments : --dry-run, -f , -y, --dir-path

Persistence syncs Grafana Server with Management Repo (All the changes in Grafana DB are reflected in mgmt node).

Whatever operations on dashboard executed on Grafana will sync everything to Management Node Repo once you execute this option.

- Create a new Dashboard or edit a custom dashboard on grafana UI.
- Once all new dashboards are ok save it on grafana.



1. On the management node execute list to see current status of Custom Dashboards

```
hvim cvimmon-dashboard list
```

© 2026 HCLTech. All rights reserved.

```
cvimmon-dashboard list

[INFO]: Save will save all custom dashboards in Grafana server to the dashboard repository on the management node [INFO]: Upload will upload all custom dashboards in the management node repository to the Grafana server

Fetching Custom Dashboards...

+------------------------------+------------------------------+--------------------------+
| Grafana Server | Management Node Repository | Status |
+------------------------------+------------------------------+--------------------------+
| hanew.json | (N/A) | New in Grafana Server |
| haproxy_active_servers.json | (N/A) | New in Grafana Server |
| New_dashboard_Copy.json | (N/A) | New in Grafana Server |
| random.json | (N/A) | New in Grafana Server |
+------------------------------+------------------------------+--------------------------+
```

Execute :

```
hvim cvimmon-dashboard save
```

This will sync all custom dashboards to the Management Repo and following output will be shown:

```
hvim cvimmon-dashboard save


Fetching Custom Dashboards...

+------------------------------+--------------------------------+--------------------------- +
| Grafana Server | Management Node Repository | Status |
+------------------------------+--------------------------------+--------------------------- +
| hanew.json | (N/A) | New in Grafana Server |
| haproxy_active_servers.json | (N/A) | New in Grafana Server |
| New_dashboard_Copy.json | (N/A) | New in Grafana Server |
| random.json | (N/A) | New in Grafana Server |
+------------------------------+--------------------------------+--------------------------- +
[INFO] Saving New Dashboard [hanew.json] on Management Node.

[INFO] Saving New Dashboard [haproxy_active_servers.json] on Management Node.

[INFO] Saving New Dashboard [New_dashboard_Copy.json] on Management Node.

[INFO] Saving New Dashboard [random.json] on Management Node.


Fetching Custom Dashboards...

+------------------------------+------------------------------+----------+
| Grafana Server | Management Node Repository | Status |
+------------------------------+------------------------------+----------+
| hanew.json | hanew.json | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
| New_dashboard_Copy.json | New_dashboard_Copy.json | Identical |
| random.json | random.json | Identical |
+------------------------------+------------------------------+----------+
```

NOTE: Save operation also deletes custom dashboard from Management Node Repo if deleted from grafana.

Navigate to Grafana Manage Dashboard Page and delete a dashboard. In this case we are deleting random.json from Grafana Server.

© 2026 HCLTech. All rights reserved.

```
hvim cvimmon-dashboard list


[INFO]: Save will save all custom dashboards in Grafana server to the dashboard repository on the management node
[INFO]: Upload will upload all custom dashboards in the management node repository to the Grafana server


Fetching Custom Dashboards...


+-----------------------------+-----------------------------+-------------------------+
|       Grafana Server        | Management Node Repository |          Status         |
+-----------------------------+-----------------------------+-------------------------+
|         hanew.json          |         hanew.json          |        Identical        |
| haproxy_active_servers.json | haproxy_active_servers.json |        Identical        |
|  New_dashboard_Copy.json    |  New_dashboard_Copy.json    |        Identical        |
|           (N/A)             |        random.json          | Deleted in Grafana Server |
+-----------------------------+-----------------------------+-------------------------+
```

On the Management Node execute :

```
hvim cvimmon-dashboard save


Fetching Custom Dashboards...


+-----------------------------+-----------------------------+------------------------- +
|       Grafana Server        | Management Node Repository |          Status         |
+-----------------------------+-----------------------------+------------------------- +
|         hanew.json          |         hanew.json          |        Identical        |
| haproxy_active_servers.json | haproxy_active_servers.json |        Identical        |
|  New_dashboard_Copy.json    |  New_dashboard_Copy.json    |        Identical        |
|           (N/A)             |        random.json          | Deleted in Grafana Server |
+-----------------------------+-----------------------------+------------------------- +
Failed: Stale Dashboard ['random.json'] detected on Management Node. Use force operation to remove these dashboards from Management Node.
```

Note: You need to pass -f / --force option to proceed as a file will be deleted from the Management Node Repo.

© 2026 HCLTech. All rights reserved.

```
hvim cvimmon-dashboard save -f


Fetching Custom Dashboards...


+-----------------------------+-----------------------------+-------------------------+
|       Grafana Server        | Management Node Repository |          Status         |
+-----------------------------+-----------------------------+-------------------------+
|         hanew.json          |         hanew.json          |        Identical        |
| haproxy_active_servers.json | haproxy_active_servers.json |        Identical        |
|  New_dashboard_Copy.json    |  New_dashboard_Copy.json    |        Identical        |
|           (N/A)             |        random.json          | Deleted in Grafana Server |
+-----------------------------+-----------------------------+-------------------------+
[INFO] Skipping Dashboard [hanew.json] as no change detected.


[INFO] Skipping Dashboard [haproxy_active_servers.json] as no change detected.


[INFO] Skipping Dashboard [New_dashboard_Copy.json] as no change detected.


[INFO] Deleting Dashboard [random.json] from Management Node Repo.



Fetching Custom Dashboards...


+-----------------------------+-----------------------------+----------+
|       Grafana Server        | Management Node Repository |  Status  |
+-----------------------------+-----------------------------+----------+
|         hanew.json          |         hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|  New_dashboard_Copy.json    |  New_dashboard_Copy.json    | Identical |
+-----------------------------+-----------------------------+----------+
Management Repo is in sync with Grafana Server
```

We also give privilege to user to save all custom dashboards on Grafana server to a user provided dir. The only pre-requisite is that the dir path provided should be empty.

© 2026 HCLTech. All rights reserved.

If the dir path is not empty following error will be thrown to user:

```
hvim cvimmon-dashboard save --dir-path /root/sync_dash/


Fetching Custom Dashboards...


+-----------------------------+-----------------------------+ --------- +
|       Grafana Server        | Management Node Repository |   Status  |
+-----------------------------+-----------------------------+ --------- +
|          hanew.json         |          hanew.json         | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|  New_dashboard_Copy.json    |  New_dashboard_Copy.json    | Identical |
+-----------------------------+-----------------------------+ --------- +
Failed: /root/sync_dash/ is not empty. Make sure the directory provided is empty and then proceed.
```

Once you have emptied the dir and executed the operation again:

© 2026 HCLTech. All rights reserved.

```
hvim cvimmon-dashboard save --dir-path /root/sync_dash/


Fetching Custom Dashboards...


+------------------------------+------------------------------+----------+
|       Grafana Server         |  Management Node Repository |   Status  |
+------------------------------+------------------------------+----------+

|          hanew.json          |          hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|   New_dashboard_Copy.json    |   New_dashboard_Copy.json    | Identical |
+------------------------------+------------------------------+----------+
[INFO] Syncing hanew.json to /root/sync_dash/
[INFO] Syncing haproxy_active_servers.json to /root/sync_dash/
[INFO] Syncing New_dashboard_Copy.json to /root/sync_dash/



Fetching Custom Dashboards...


+------------------------------+------------------------------+----------+
|       Grafana Server         |  Management Node Repository |   Status  |
+------------------------------+------------------------------+----------+

|          hanew.json          |          hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|   New_dashboard_Copy.json    |   New_dashboard_Copy.json    | Identical |
+------------------------------+------------------------------+----------+
User Directory [/root/sync_dash/] is synced with Grafana
```

INFO: If you pass --dry-run to any of the operations with save option, No changes will be made on the Management Node repo. Dry run will only show what is going to change for example:

© 2026 HCLTech. All rights reserved.

```
hvim cvimmon-dashboard save --dir-path /root/sync_dash/ --dry-run


No Changes made as --dry-run was executed with save operation.



Fetching Custom Dashboards...


+------------------------------+------------------------------+--------- +
|       Grafana Server         |  Management Node Repository |   Status  |
+------------------------------+------------------------------+--------- +
|         hanew.json           |          hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|   New_dashboard_Copy.json    |   New_dashboard_Copy.json    | Identical |
+------------------------------+------------------------------+--------- +
[INFO] Syncing hanew.json to /root/sync_dash/
[INFO] Syncing haproxy_active_servers.json to /root/sync_dash/
[INFO] Syncing New_dashboard_Copy.json to /root/sync_dash/


Fetching Custom Dashboards...


+------------------------------+------------------------------+--------- +
|       Grafana Server         |  Management Node Repository |   Status  |
+------------------------------+------------------------------+--------- +
|         hanew.json           |          hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|   New_dashboard_Copy.json    |   New_dashboard_Copy.json    | Identical |
+------------------------------+------------------------------+--------- +
[root@U6-i13 installer-21667]# ls -la /root/sync_dash/ total
4
drwxr-xr-x.  2 root root    6 Sep 16 15:10 .
drwx------. 18 root root 4096 Sep 16 14:25 .
```

As you can see, even after the operation the targeted dir is empty.

Upload Workflow (Syncing Management Node Repo with Grafana Server) Supported arguments : --dry-run, -f , -p, -y, --dir-path

Upload syncs Management Repo with Grafana Server.

--dir-path is a mandatory option with upload feature as all dashboards from this dir will be uploaded to Management Node Repo and then upload to Grafana Server.

© 2026 HCLTech. All rights reserved.

if -f / --force is passed with upload option then all existing dashboard in the Management Node Repo will be deleted.

If -p / --preserve option is passed then all dashboards will be preserved and new dashboards will be added to the Management Node Repo.

[**NOTE**]: --force or --preserve should be passed with upload and --dir-path option.

```
hvim cvimmon-dashboard upload  --force --dir-path /root/sync_dash/
+----------------+----------------------------+-------- +
| Grafana Server | Management Node Repository | Status |
+----------------+----------------------------+-------- +
|     (N/A)      |                (N/A)               | (N/A)  |
+----------------+----------------------------+-------+
[INFO] All existing Dashboards will be deleted.
[INFO] Saving New Dashboard [hanew.json] on Management Node.


[INFO] Saving New Dashboard [haproxy_active_servers.json] on Management Node.


[INFO] Saving New Dashboard [New_dashboard_Copy.json] on Management Node.



Fetching Custom Dashboards...


+-----------------------------+-----------------------------+----------+
|        Grafana Server       |  Management Node Repository |   Status  |
+-----------------------------+-----------------------------+----------+
|            hanew.json       |                hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|   New_dashboard_Copy.json   |   New_dashboard_Copy.json   | Identical |
+-----------------------------+-----------------------------+----------+
User Directory [/root/sync_dash/] is synced with Grafana
```

© 2026 HCLTech. All rights reserved.

If save operation is executed again with preserve option then following is the output:

```
hvim cvimmon-dashboard upload  --preserve --dir-path /root/sync_dash/


Fetching Custom Dashboards...


+------------------------------+------------------------------+----------+
|       Grafana Server         |  Management Node Repository |   Status  |
+------------------------------+------------------------------+----------+
|          hanew.json          |          hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|   New_dashboard_Copy.json    |   New_dashboard_Copy.json    | Identical |
+------------------------------+------------------------------+----------+
[INFO] All existing Dashboards will be preserved.
[INFO] Skipping Dashboard [hanew.json] as no change detected.


[INFO] Skipping Dashboard [haproxy_active_servers.json] as no change detected.


[INFO] Skipping Dashboard [New_dashboard_Copy.json] as no change detected.



Fetching Custom Dashboards...


+------------------------------+------------------------------+----------+
|       Grafana Server         |  Management Node Repository |   Status  |
+------------------------------+------------------------------+----------+
|          hanew.json          |          hanew.json          | Identical |
| haproxy_active_servers.json | haproxy_active_servers.json | Identical |
|   New_dashboard_Copy.json    |   New_dashboard_Copy.json    | Identical |
+------------------------------+------------------------------+----------+
User Directory [/root/sync_dash/] is synced with Grafana
```

NOTE: If you are uploading a dashboard with same name which is already there in the Grafana Server then the modified dashboard will be uploaded on the grafana. This works same in case of "save dashboard" option. If you modify any existing dashboard on grafana server and execute save operation then modified dashboard will be saved on the Management Node Repo.Before executing any operation like upload/save VIM VIM will check if all Repos are in sync if there is any delta then operation will be terminated with proper error message.

All the logs for hvimmon-dashboards are populated under /var/log/mercury_restapi/restapi.log file.

© 2026 HCLTech. All rights reserved.

Highly Available VIM Monitor

- Overview of HA VIM-MON
- Hardware Requirements for HA VIM MON Networking Layout
- Network Topologies
- Architecture
- Installation Modes
- Setup File
- HA VIM-MON Installer
- External Servers
- Resources
- Pod Operations
- Stack Operations
- HA VIM-MON Update
- Custom Grafana Dashboards
- Alert Rules
- Alert Manager
- Backup
- Restore
- Reinstallation of VIM-MON HA Pod
- VIM MON HA Cluster Monitoring
- HA VIM Mon Upgrade
- SERVER MON
- EFK for VIM-MON HA Management Node.
- LDAP Integration for VIM-MON HA Management Node

Overview of HA VIM-MON

Overview of Highly Available VIM Monitor

From VIM 3.4.3, you can monitor VIM pods either:

- Individually using the local VIM Monitor (VIM-MON) or
- Centrally using the new HA VIM Monitor (HA VIM-MON)

The local VIM-MON (introduced in VIM 3.0.0) provides pod-level monitoring, based on a Prometheus stack that is hosted on the pod management node. This local solution supports the largest supported VIM pod size (128 nodes).

Local VIM-MON has the following limitations:

- Not highly available as the downtime of the management node stops metric collection in the pod.
- Multi-site monitoring of large deployments with a large number of sites can be operationally complex and requires configuration of each VIM

pod.

Very small sites with severely limited hardware resources (edge cloud) cannot afford the resources to run a dedicated Prometheus stack per site.

HA VIM-MON is a complete hardware and software platform that addresses local VIM-MON limitations with the following features:

- Integrated and highly-available monitoring of a large number of VIM pods
- Centralized TSDB, alarm and web-based GUI dashboards
- It is based on bare metal Kubernetes platform
- Scalable to hundreds of VIM pods and thousands of nodes.

© 2026 HCLTech. All rights reserved.

- Provides a longer retention time for collected metrics (months instead of 15 days for the local VIM-MON)
- Low sampling interval of one minute for largest deployments.
- Monitor pods of any size including very small pods (edge deployments) and individual bare metal servers.
- Monitored pods or servers are hierarchically grouped into metros and metros in regions.

HA VIM-MON supports and requires a limited set of hardware configurations. HA VIM-MON deploys on bare metal using a fully automated installer and can be configured or reconfigured by updating the setup data configuration file.

The following table provides a comparative summary of monitoring features between VIM 3.4.x and VIM 4.0.0:

Features	VIM 3.4.x	VIM 4.0.0	Comments
VIM pod local monitoring (VIM- MON)	yes	yes	
VIM pod centralized monitoring (HA VIM-MON)	yes	yes	Local and centralized monitoring are exclusive. Requires an HA VIM-MON cluster to monitor the VIM pods.
VIM pod local to central monitoring reconfiguration	yes	yes	No service impact
VIM pod central to local monitoring reconfiguration	no	no	Not supported
HA VIM-MON unified monitoring access points	limited within stack boundary only	yes	A monitoring domain provides a set of service points (Grafana, Prometheus queries), alerting rules, and alert manager configurations, to monitor all VIM pods configured in that domain.
HA VIM-MON estimated max scale per monitoring stack (1m interval)	1000 sites /8000 nodes (per stack)	500 sites /4000 nodes (per stack)	Exact limit depends on the actual cluster hardware BOM. A monitoring stack is the software component that is responsible for collecting metrics from a group of VIM pods.
HA VIM-MON max number of monitoring stacks per cluster	min(4, N-1)	min(4, N-2)	N: number of HA VIM-MON cluster nodes (controller and worker nodes) For example, with HA VIM-MON 4.0, the minimum configuration to support 4000 sites is three controller nodes and three worker nodes.
HA VIM-MON metric collection downtime in case of stack or cluster node failure	up to 30 minutes	no downtime	The most common stack failure is caused when the cluster node running the stack goes down (system crash or hardware failure)
HA VIM-MON cluster node self- monitoring	partial	complete	The self-monitoring added in 4.0.0 includes bare metal, system, link states, and IPMI metrics with associated alerting rules - similar to VIM pod servers.

© 2026 HCLTech. All rights reserved.

Features	VIM 3.4.x	VIM 4.0.0	Comments
			UCS-C bare metal alerts are still not supported in 4.0.0 and are planned for a future version.
HA VIM-MON Grafana built-dashboards organization	flat	1-level folders	In 4.0.0, the built-in dashboards are grouped by folders
HA VIM-MON CLI integration	partial	complete	In 4.0.0, all cluster management operations are integrated under the *k8s_runner* CLI

⚠ Starting from vim 4.X there is NO support of VIM-MON HA on Quanta Platform

Hardware Requirements for HA VIM MON

Hardware Requirements for HA VIM-MON

HA VIM-MON is available for Cisco UCS C-series servers with:
- One server is used as a management node.
- Three or more servers to form a cluster managed by Kubernetes:
  - 384 GB RAM
  - All-SSD storage devices on the cluster nodes

To achieve the required network throughput, one of the following must be satisfied:

Each UCS server must have two Intel X710 cards.

Networking Layout

Networking Layout for HA VIM-MON

- Public Network
- Management and Provisioning Segment

Public Network

The public network (br_api) interfaces with:

- External applications using HA VIM-MON such as an OSS/BSS system querying the TSDB or browsers connecting to the HA VIM-MON
- GUI. Managed VIM pods (for metrics collection).
- Managed servers.
- HA VIM-MON administrators (ssh).

This public network is implemented by the br_api interface and provides external access to the following services:
- Kubernetes infrastructure administrator services.
- Kubernetes cluster nodes (ssh).
- Grafana, Prometheus, and Alertmanager HTTP services.

© 2026 HCLTech. All rights reserved.

The public network segment needs one VLAN and at least five IPv4 or IPv6 addresses in an externally accessible subnet:

- One IP address for the management node.
- One IP address for each of the cluster nodes.
- One IP address for external_lb_vip for accessing the HA VIM-MON services.

Management and Provisioning Segment

The management segment (br_mgmt) needs one separate VLAN and one subnet with an address pool large enough to accommodate all the current and future servers planned for the cluster for initial provisioning (PXE boot Linux) and for all Kubernetes internal communication. This VLAN and subnet can be local to VIM-MON for UCS C-Series and Quanta deployments. All cluster nodes need an IP address from this subnet. The BMC or CIMC network must be accessible through the public network.

Network Topologies

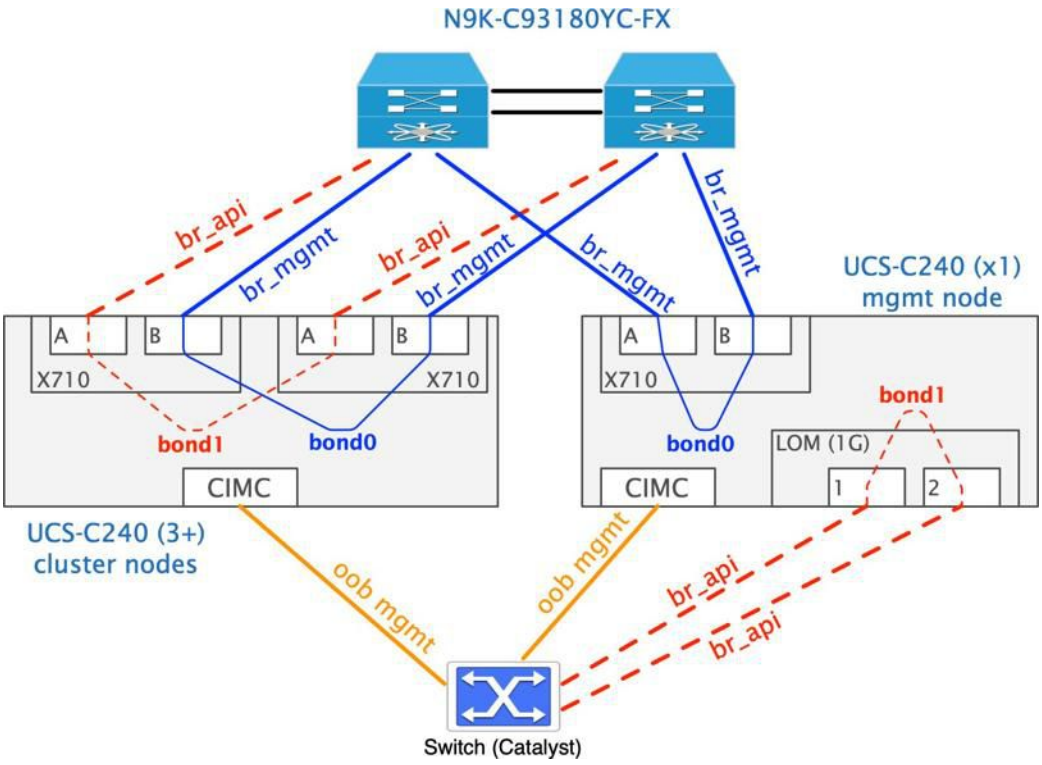
Network Topologies for HA VIM-MON

- UCS C-Series Network Topology
  - NXOS Switch Configuration for VIM MON HA UCS nodes NXOS
  - Switch Configuration for Management Node (QCT or UCS)

UCS C-Series Network Topology

UCS-C based servers use Intel X710 NIC (4x10G, two NICs for each cluster node, one NIC for the management node). Teaming is used for the *br_api* and *br_mgmt* links with dual N9K TORs.

The management node saves one Intel X710 NIC by using X710 for both *br_mgmt* links and LOM ports for the *br_api* links.



© 2026 HCLTech. All rights reserved.

NXOS Switch Configuration for VIM MON HA UCS nodes

It is assumed that switches are preconfigured in virtual port channel mode. For NXOS configuration details, see *NXOS Switch Configuration Guide*. Listed below are the corresponding NXOS switch configuration of the of the port A and B links for each of the UCS nodes that make up the VIM MON HA cluster.

The NXOS Switch configuration connected to Intel NIC port A on the UCS nodes that make up the VIM MON HA cluster:

```
interface port-channel <pc_id_3>
  description mgmt API interface
  switchport
  switchport access vlan <api_vlan_id>
  spanning-tree port type edge
  no lacp suspend-individual
  vpc <pc_id_3>


interface Ethernet1/<id_3>
  description mgmt API interface
  switchport
  switchport access vlan <api_vlan_id>
  channel-group <pc_id_3> mode active
  no shutdown
```

The NXOS Switch configuration connected to Intel NIC port B on the UCS nodes that make up the VIM MON HA cluster:

```
interface port-channel <pc_id_4>
  description mgmt interface
  switchport
  switchport access vlan <mgmt_vlan_id>
  spanning-tree port type edge
  no lacp suspend-individual
  vpc <pc_id_4>

interface Ethernet1/<id_4>
  description america2 mgmt interface
  switchport
  switchport access vlan <mgmt_vlan_id>
  channel-group <pc_id_4> mode active
  no shutdown
```

NXOS Switch Configuration for Management Node (UCS)

It is assumed that switches are preconfigured in virtual port channel mode. Listed below are the specific configuration of the *br_api* (over LOM interface via Copper) and *br_mgmt* links for the UCS management node of VIMMON HA cluster.

The NXOS switch configuration of *br_api* (over 1G) for the management node in the VIMMON HA cluster:

```
interface port-channel <pc_id_1>
  description API interface
  switchport
  switchport access vlan <api_vlan_id>
  spanning-tree port type edge
  no lacp suspend-individual
  vpc <pc_id_1>


interface Ethernet1/<id_1>
  description API interface
  switchport
  switchport access vlan <api_vlan_id_>
  channel-group <pc_id_1> mode active
  no shutdown
```

The NXOS switch configuration of *br_mgmt* (over 10G interface) for the management node in the VIMMON HA cluster:

© 2026 HCLTech. All rights reserved.

```
interface port-channel <pc_id_2>
  description mgmt interface
  switchport
  switchport access vlan <mgmt_vlan_id>
  spanning-tree port type edge
  no lacp suspend-individual
  vpc <pc_id_2>

interface Ethernet1/<id_2>
  description america2 mgmt interface
  switchport
  switchport access vlan <mgmt_vlan_id>
  channel-group <pc_id_2> mode active
  no shutdown
```

For more details on switch configuration, see *NXOS Switch Configuration Guide*.

Architecture

Overview of HA VIM-MON Architecture

The minimum hardware configuration runs with a single management node and three controller nodes. In this configuration, the three controller nodes host the Kubernetes control plane components and application containers that perform the HA VIM-MON function. You can extend this configuration with one or more worker nodes, based on computational and storage requirements. Worker nodes host only application containers.

The HA VIM-MON is based on the following software components:

- One or more Prometheus servers running in one or more *stacks*
- One Minio object storage cluster to store all monitoring data
- One set of Kubernetes pods based on Thanos open-source project to manage
  - User queries and time series deduplication
  - Storage of Prometheus time series into Minio
  - Unified alerting rules
- One SNMP agent to forward alerts to external SNMP managers
- One Grafana cluster for providing the web user interface

Minio object storage cluster handles persistent and highly available storage for monitoring data. It also takes care of replicating the storage blocks across all cluster nodes in a transparent way to the applications.

The main differences of HA VIM-MON 4.0.0 compared to HA VIM-MON 3.4.x are as follows:

- The time series persistent storage is Minio object storage instead of the Portworx block-level storage.
- A monitoring stack is now exclusively in charge of scraping metrics from targets.
- Each stack is composed mainly of two Prometheus servers that are configured in active-active for scraping the same set of VIM pods.
- Alerting, user queries, and Grafana user interface are now unified (one service point per cluster) and no longer per stack.

For more information on stacks, see Stack Operations

For information on monitoring external servers using HA VIM-MON , see External Servers

Installation Modes

Installation Modes for HA VIM-MON

You can install HA VIM-MON using three installation modes.

- Connected Mode of Install
- Air-Gapped Install using USB

© 2026 HCLTech. All rights reserved.

- Air-Gapped Installation Using Software Delivery Server

> VIM-MON installation must be done using a VNC or screen so that the session is not lost. If you do not have a VNC environment, execute the same from KVM console of the management node. Ensure that you do not run this command in background or with nohup option to avoid failure.

Connected Mode of Install

You can perform this mode of installation when the VIM management node has internet connectivity. All the artifacts and docker images needed for installation are directly fetched from the internet and utilized by the installer. This is the default mode of HA VIM-MON install. You must provide the following information in the *setup data* file to fetch artifacts from vim-registry.com:

```
REGISTRY_USERNAME: <username>
REGISTRY_PASSWORD: <password>
REGISTRY_EMAIL: <email>
```

Air-Gapped Install using USB

The following procedure describes how to download the NFVI installation files onto a USB drive of the staging server with Internet access. You can use the USB to load the ISC installation files onto the management node without Internet access.

> Ensure that you use Virtual Network Computing (VNC), a terminal multiplexer, or similar screen sessions to complete the following steps.

> Before you begin you must have a CentOS 7 staging server (VM, laptop, or UCS server) with a 64 GB USB 2.0 drive. You can use USB 3.0 64GB if the management node is of type Cisco UCS M5. The staging server must have a wired Internet connection to download the vim installation files onto the USB drive. Once downloaded, you can copy the installation files onto the management node from a USB drive.

Fetching artifacts to the staging server:

On the staging server, use yum to install the following packages:

- PyYAML
- Python-requests
- Centos-release-scl
- Python 3.6

Check if python 3.6 binary is located at /opt/rh/rh-python36/root/bin/, if not copy the python 3.6 binary to /opt/rh/rh-python36/root/bin/.

Log into VIM software download site and download the *getartifacts.py* script from the external registry:

```
# download the new getartifacts.py file (see example below)
curl -o getartifacts.py -u '<username>:<password>' https://vim-registry.com/mercury- releases /hvim50-rhel9-osp17/releases/<releaseid>/getartifacts.py
Change the permission of getartificats.py chmod +x getartifacts.py
```

© 2026 HCLTech. All rights reserved.

Run getartifacts.py.

The script formats the USB2.0 drive (or USB3.0 drive for M5/Quanta based management node) and downloads the installation files. You must provide the registry username and password, tag ID, and USB partition on the staging server.

```
# ./getartifacts.py -t <tag_id > -u <username> -p <password> -d <device_path> --mgmtk8s [--proxy]
<proxy>
```

Use the following command to verify the downloaded artifacts and container images:

```
# create a directory
sudo mkdir -p /mnt/Cisco
# You need to mount the partition with the steps given below:
sudo mount <device_path> /mnt/Cisco
cd /mnt/Cisco
# execute the verification script
./test-usb
```

If the *test-usb* script reports any failures, you can unmount the USB and run the *getartifacts* command again with the *--retry* option.

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -d <device_path> --mgmtk8s --retry
```

Mount the USB and then run the *test-usb* command to validate if all the files are downloaded:

```
# create a directory sudo mkdir -p /mnt/Cisco
# You need to mount the partition with the steps given below: sudo mount <device_path> /mnt/Cisco
cd /mnt/Cisco
# execute the verification script
./test-usb
```

When the USB integrity test is done, unmount the USB drive by using the following command:

```
# Unmount USB device
sudo umount /mnt/Cisco
```

Importing artifacts from the USB on to the management node.

2. On the VIM MON HA management node, use the prepared USB stick and complete the following steps:

- Insert the USB stick into the management node drive, after you install the *buildnode.iso*.
- Use *import_artifacts.sh* script to copy all the artifacts onto the management node. After successful completion, the installation artifacts are copied to */var/cisco/artifacts* on the management node. After the artifacts are available in the management node, the steps to

© 2026 HCLTech. All rights reserved.

install HA VIMON pod remain the same.

```
# Run import artifacts script
cd ~/installer-<tag_id>/tools
./import_artifacts.sh
```

Configure the setup data file:

HA VIM MON *setup data* file has a configuration to set the install mode. Set the install mode as disconnected to prevent the management node from fetching the artifacts from Internet. For example:

```
INSTALL_MODE: disconnected
```

Air-Gapped Installation Using Software Delivery Server

The Software Delivery Server (SDS) is also called the VIM Software Hub.

VIM Software Hub alleviates the need for VIM management nodes to have internet connectivity and helps to remove the logistics of shipping USBs to multiple pods across the enterprise for software installation or update of the cloud. You can install and download the HA VIM MON artifacts on the SDS server.

For more information on the hardware requirements of the SDS server and steps to install artifacts, see VIM Software Hub > *Installing VIM Software Hub in Air-gapped Mode*.

Configuration of Setupdata File

After you pre-install the artifacts on the VIM Software Hub, you can start the HA VIM MON installation using SDS.

⚠ Ensure that the br_api IP address can reach the br_private IP address of the SDS server.

- Install the management node with build node ISO.
- Add the following fields in the HA VIM MON *setup data* file.

```
REGISTRY_NAME: '<registry_name>' # Mandatory Parameter.
```

HA VIM MON *setup data* file requires the REGISTRY_USERNAME and REGISTRY_PASSWORD to connect to the docker registry and fetch docker images.

To fetch the docker images from VIM Software Hub node, provide the user credentials available in the SDS_READ_ONLY_USERS section of *sds_setup_data.yaml*. The details of an admin user with read or write access to the docker registry are provided in **SDS_REGISTRY_USERNA ME** and **SDS_REGISTRY_PASSWORD**. Hence, it is recommended that you have a read-only user on the VIM pod.

Setup File

Configuring Setup Data File for HA VIM-MON

- Overview

© 2026 HCLTech. All rights reserved.

- Argus Bare Metal Configuration Parameters
- General HA VIM-MON Cluster Parameters
- HA VIM-MON Stack Parameters
  - Regions, Metros, and Pods
  - SNMP
  - Syslog Export
  - LDAP Support for Grafana (VIM MON Stacks)
  - LDAP Support for Grafana (VIM MON HA Cluster Monitor)

Overview

Before you begin, install the management node ISO on the management node. From the menu describing the different types of management nodes, select HA VIM-MON management node.

Use the setup data file that describes all the parameters of HA VIM-MON cluster and the list of all monitored VIM pods, for the installation of the HA

VIM-MON cluster after the management node is up and running. Following are the naming conventions used:

- setup data file: Refers to various versions of the *setup_data.yaml* file that reside on the management node.
- setup data targets file: An optional file that contains per-stack specific targets, named <*stack_name*>*.yaml* . There can be as many target files as the number of stacks. The targets files are always associated to a setup data file and are always located in the same directory. They are also called side car files.
- setup data file set: A group of files that include a setup data file and zero or more associated targets files. In most deployments, it is simpler to use a single setup data file that contains everything including VIM target pods.

HA VIM-MON always keep a copy of the current setup data file set under */root/openstack-configs* called the *reference setup data file set*. The reference setup data file is */root/openstack-configs/setup_data.yam*l and must be considered read-only for all operations. This file must never be modified directly.

For fresh installation, the reference setup data file is not expected to be present in the current workspace. A candidate setup data file set must be created outside the */root/openstack-configs/* directory and passed as an argument to the installer *(–setupfile <pathname>*).

After a successful installation, the reference setup data file set is updated with the candidate setup data file set to reflect the current state of the cluster. You cannot directly modify or remove those files under any circumstances.

For detailed description of parameters, see the example setup data file available at: */root/openstack-configs/setup_data.yaml.HA VIM-MON.EXAMPLE*.

The following sections simply list the parameters in the setup data file.

Argus Bare Metal Configuration Parameters

You must set the following parameters for bare metal configuration for all cluster nodes:

- br_api and br_mgmt network addressing
- Type of node (controller or worker node)
- Bare metal access credentials for configuration of cluster node (CIMC or BMC)
- Linux root credentials for installation of Linux on all cluster nodes

General HA VIM-MON Cluster Parameters

Configure the general parameters of the HA VIM-MON cluster:

- Internal IP address for internal load balancer for the br_mgmt network
- External IP address for external load balancer for the br_api network
- NTP and DNS servers
- Domain suffix for all external URLs to HA VIM-MON services
- Virtual router ID for Virtual Router Redundancy Protocol (VRRP)

© 2026 HCLTech. All rights reserved.

- Location of the domain CA cert bundle to access the cluster services
- Metric scraping interval
- Metrics retention time. Defaults to 12 Weeks.
- Log rotation parameters
- Frequency of log rotation
- Maximum size of each log. When the size of the log exceeds this value, a rotation occurs.
- Number of compressed archive log files to keep for each log file. The old archive log files are deleted.

The IP addresses must be either IPv4 or IPV4+IPv6, based on selected IP version in the bare metal section.

HA VIM-MON Stack Parameters

Configure the stack properties given below:

Stack property	Description
Stack name	It is mandatory. Helps to identify each stack. The name must be in lowercase alphanumeric characters. The only special character allowed is ' -' .
Regions, Metros, Pods	Provides the list of regions, metros, and pods that are monitored by the stack.

Regions, Metros, and Pods

The region, metro, and pod names must be unique within the monitoring domain. They can be any ASCII string. These names are only used as a metric label value.

You must configure each region with the following parameters:
- A region name
- A list of metros

You must configure each metro with the following parameters:
- A metro name
- A list of VIM pods

You must configure each VIM pod with the following parameters:
- A pod name
- Pod IP address (IPv4 or IPv6)
- Pod HA proxy certificate
- User name and password to access the pod

SNMP

You must configure SNMP for each stack only if SNMP traps are enabled. When SNMP traps are enabled, all HA VIM-MON alerts in the stack are forwarded to the configured SNMP managers using the selected SNMP version.

You must configure the following parameters for SNMP:
- IPv4 or IPv6 address and port to send traps to.
- SNMP version: v2c (default) or v3.
- SNMP credentials:
  - v2c-Community string
  - v3-engine ID, credentials, and encryption settings

Syslog Export

You must configure Syslog export feature when you want the HA VIM-MON alerts logs to be sent to one or up to 4 remote syslog servers. When SYSLOG_EXPORT is configured, you must configure SNMP.

© 2026 HCLTech. All rights reserved.

Following are the parameters to be configured:

- IPv4 or IPv6 address and port for sending the alert logs.
- Facility used by the remote syslog server (only user and local[1-7] are allowed)

```
CVIMMONHA_CLUSTER_MONITOR:
  SYSLOG_EXPORT:
    remote_hosts:
    - host: 10.10.10.10
      facility: user
      port: 514
```

LDAP Support for Grafana (VIM MON Stacks)

Grafana has two default users with dynamically assigned passwords and different roles: • Viewer: Cannot create new or modify existing dashboards.

- Admin: Can create new and modify existing dashboards.

LDAP configuration consists of two main sections:

- Domain mappings
- Group mappings

Domain Mappings

- attributes
  - email
  - member_of
  - name
  - surname
  - username
- bind_dn
- bind_password
- use_ssl
- start_tls
- client_cert
- client_key
- root_ca_cert
- group_search_filter
- group_search_base_dns
- group_search_filter_user_attribute
- domain_name
- ldap_uri
- search_base_dns
- search_filter

Group Mappings

These mappings are required to authorize users from a specific group in ldap to access Grafana with *Admin* or *Viewer* permissions.

- group_dn : 'CN=group2,OU=GroupsDC=OrgName,DC=com' , org_role: Admin
- group_dn : 'CN=group3,OU=GroupsDC=OrgName,DC=com' , org_role: Viewer

Users from Group Employees get Admin access for Grafana whereas users from Read-Only Employees get Viewer privileges. For example:

© 2026 HCLTech. All rights reserved.

```
cvim_mon_stacks:
- name: qomstack
  ldap:
    domain_mappings:
      - attributes:
        email: email
        name: givenName
        surname: sn
        username: uid
        bind_dn: <bind_dn>
        bind_password: <bind_password>
        domain_name: corp_ldap1
        group_search_base_dns: ['ou=Groups,dc=OrgName,dc=com']
        group_search_filter: "(&(objectClass=posixGroup)(memberUid=%s))"
        group_search_filter_user_attribute: uid
        ldap_uri: ldap(s)://<ldap_url>:<Port>
        root_ca_cert: "/root/cvimha_certs/<cert_name>"
        search_base_dns: ['dc=OrgName,dc=com']
        search_filter: "(uid=%s)"
        start_tls: false
        use_ssl: true
    group_mappings:
      - group_dn: cn=group2,ou=Groups,dc=OrgName,dc=com
        org_role: Admin
      - group_dn: cn=group3,ou=Groups,dc=OrgName,dc=com
        org_role: Viewer
```

- Only one LDAP server can be configured per stack
- LDAP support is only enabled for Grafana Dashboard.

LDAP Support for Grafana (VIM MON HA Cluster Monitor)

VIM MON HA cluster itself is monitored by default on all VIM MON HA pods. Grafana, Prometheus, and Alertmanager are installed in an isolated hvimmon-monitor namespace. Default admin credentials are generated during installation to access Grafana dashboard. Also, the LDAP users are allowed to access the dashboard. Example snippet of *setup_data.yaml* to add LDAP support for hvimmon-monitor is provided below:

```
CVIMMONHA_CLUSTER_MONITOR:
  ldap:
    domain_mappings:
      - attributes:
        email: email
        name: givenName
        surname: sn
        username: uid
        bind_dn: <bind_dn>
        bind_password: <bind_password>
        domain_name: corp_ldap1
        group_search_base_dns: ['ou=Groups,dc=OrgName,dc=com']
        group_search_filter: "(&(objectClass=posixGroup)(memberUid=%s))"
        group_search_filter_user_attribute: uid
        ldap_uri: ldap(s)://<ldap_url>:<Port>
        root_ca_cert: "/root/cvimha_certs/<cert_name>"
        search_base_dns: ['dc=OrgName,dc=com']
        search_filter: "(uid=%s)"
        start_tls: false
        use_ssl: true
    group_mappings:
      - group_dn: cn=group2,ou=Groups,dc=OrgName,dc=com
        org_role: Admin
      - group_dn: cn=group3,ou=Groups,dc=OrgName,dc=com
        org_role: Viewer
```

External Servers

LDAP support can be added or removed as a Day 1 operation using reconfigure operation.

© 2026 HCLTech. All rights reserved.

The servers must meet a few pre-requisites. For more information, see External Servers. You must configure the following parameters for external servers:

- Server name
- Server IP address followed by port 9273

For more information on monitoring the external servers, see VIMMON > *Monitoring External Servers Using VIM MON*

HA VIM-MON Installer

Installation and Installer CLI

- Overview
- General installer CLI
- HA VIM-MON Operational State
- Operations and Operational State History
- Installation Procedure
- Installing HA VIM-MON
- Installation Failure

Overview

Following are the high-level steps for installing HA VIM-MON cluster:

- Perform hardware installation and network wiring (described earlier)
- Install the management node
- Install the HA VIM-MON cluster

The procedure to install the HA VIM-MON management node is similar to the VIM management node installation. To install the management node, see Management Node on UCS Servers. At the prompt, you must select HA VIM-MON management node to indicate that you are installing a management node for the HA VIM-MON cluster.

This section describes the procedure to install an HA VIM-MON cluster from the management node and introduces the CLI tools used to install and manage the cluster.

General installer CLI

The *k8s_runner* CLI is the main tool used to manage the HA VIM-MON cluster, including fresh installation, updates, and all Day 2 operations.

This command executes in a separate detachable screen based on tmux, with SIGINT signal disabled to prevent accidental termination of installation. If the screen session is lost, any subsequent call to *k8s_runner* results in reattaching to the session that is currently in progress. This feature prevents the accidental termination of any operation and more specifically for operations that can take a long time to complete.

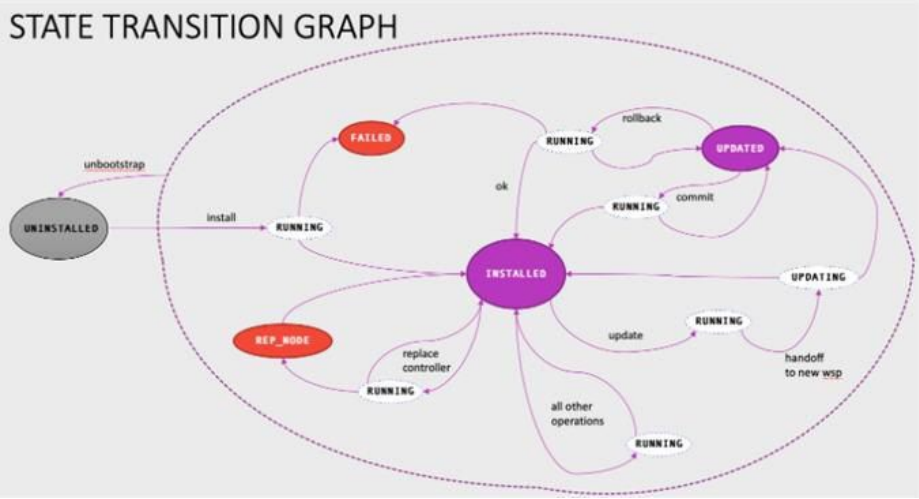
The *k8s_runner* CLI can be executed from any location on the management node. To see all available arguments, simply type k8s_runner *--help*.

HA VIM-MON Operational State

A state machine tracks the current state of the HA VIM-MON cluster on the management node and determines the operations that can be performed at any time.

© 2026 HCLTech. All rights reserved.

The following diagram shows the state transitions and allowed operations:



Following are the state color codes:

- Blue: Denotes stable/nominal state.
- Red: Denotes stable/error state.
- White: Denotes transient state indicating an operation in progress.
- Gray: Denotes stable state post fresh installation of management node.

The following table gives the list and description of each state:

State name	Description
Uninstalled	This is the initial state just after the installation of cluster management node. At that point, all other cluster nodes are normally still uninstalled.
Installed	This is the most common nominal state where the cluster is fully installed and is scraping metrics from configured VIM pod targets.
Running	This is a transient state indicating that an operation is in progress. Depending on the progress of operation, this state exists for certain time.
Updated	This state indicates that an update operation is completed, and the cluster is waiting for the administrator to either commit the update or rollback. No other operation other than the common read only operations (such as listing the state or showing endpoints) is allowed in this state. Metrics collection is running with the newly updated version of HA VIM-MON.
REP_NODE	This state indicates that the unrecoverable error has occurred during replace controller operation. This state requires TAC support to return to a nominal state.
Failed	This state indicates that an unrecoverable error has occurred during installation or rollback operation. This state requires TAC support to return to a nominal state.

The most common operations are:

- Fresh installation (Uninstalled => Installed)
- Other operation allowed in the INSTALLED state and not changing state ("all other operations" in the above diagram),
- Update to a new version of HA VIM-MON (Installed => Updated and Updated => Installed with commit)

Operations and Operational State History

The state of the cluster is preserved on the management node disk using two tables:

© 2026 HCLTech. All rights reserved.

- A state transition table or installation table that contains all past state transitions with their timestamp being limited to last 1000 state transitions.
- An operation history table or operation table that contains all past operations applied to the cluster, the operation execution steps, and their timestamps being limited to last 1000 operations.

To view the state transition table and operation history tables, execute *k8s_run_db* CLI at any time from any state and from any location.

```
# k8s_run_db -h
usage: k8s_run_db [-h] [-u] [-i] [-l NUM_ENTRIES]

This tool helps override the failed state. Reserved for internal use.

optional arguments:
  -h, --help            show this help message and exit
  -u, --force-state-uninstalled
                        This argument will reset the db but will still retain past entries
  -i, --force-state-installed
                        This will set last entry as success in all the tables
  -l NUM_ENTRIES, --show NUM_ENTRIES
                        List last entries in both install and operation tables
```

By default, when no option is provided, *k8s_run_db* displays the last 20 entries in each table. To display more entries, enter the number of entries to be displayed using the *-l* or *–show* argument.

Example of output:

```
# k8s_run_db
+-----+---------------------+---------+-------------+
| SEQ |   Timestamp (PST)   | Session | State       |
+-----+---------------------+---------+-------------+
|  2  | 2021-02-01 13:39:44 |    3    | INSTALLED   |
|  1  | 2021-02-01 11:47:58 |    1    | UNINSTALLED |
+-----+---------------------+---------+-------------+

+-----+---------------------+---------+---------+-----------+-----------------+------+---------
+---------------------------------------+
| SEQ |   Timestamp (PST)   | Session | Elapsed | State     |    Operation    | Step |  Status |
Info                                    |
+-----+---------------------+---------+---------+-----------+-----------------+------+---------
+---------------------------------------+
|  61 | 2021-02-01 14:38:39 |    12   | 0:00:01 | INSTALLED |   GET_ENDPOINT  |      | Success | log=/var/log
/cvimmonha/...                          |
|  60 | 2021-02-01 14:38:38 |    12   |         | INSTALLED |   GET_ENDPOINT  |      | Running | args=--get-
endpoint                                |
|     |                     |         |         |           |                 |      |
|                                         |
|  59 | 2021-02-01 14:35:51 |    11   | 0:03:21 | INSTALLED |    autobackup   |      | Success | backup=/var
/cisco/cvimmonha...                     |
|  58 | 2021-02-01 14:35:51 |    11   | 0:03:21 | INSTALLED |    autobackup   |      | Running
|                                         |
|  57 | 2021-02-01 14:35:51 |    11   | 0:03:21 | INSTALLED |   ADD_CVIM_POD  |      | Success | log=/var/log
/cvimmonha/...                          |
|  56 | 2021-02-01 14:35:51 |    11   | 0:03:21 | INSTALLED |   ADD_CVIM_POD  |  7   | Success
|                                         |
|  55 | 2021-02-01 14:33:24 |    11   | 0:00:54 | INSTALLED |   ADD_CVIM_POD  |  7   | Running |
step=HELM_INFRA                         |
|  54 | 2021-02-01 14:33:24 |    11   | 0:00:54 | INSTALLED |   ADD_CVIM_POD  |  1   | Success
|                                         |
|  53 | 2021-02-01 14:32:30 |    11   | 0:00:00 | INSTALLED |   ADD_CVIM_POD  |  1   | Running |
step=VALIDATION                         |
|  52 | 2021-02-01 14:32:30 |    11   |         | INSTALLED |   ADD_CVIM_POD  |      | Running | args=--add-
cvim-pod --setupfile ...                |
|     |                     |         |         |           |                 |      |
|                                         |
|  51 | 2021-02-01 14:32:25 |    10   | 0:02:56 | INSTALLED |    autobackup   |      | Success | backup=/var
/cisco/cvim...                          |
|  50 | 2021-02-01 14:32:24 |    10   | 0:02:55 | INSTALLED |    autobackup   |      | Running
|                                         |
|  49 | 2021-02-01 14:32:24 |    10   | 0:02:55 | INSTALLED | DELETE_CVIM_POD |      | Success | log=/var/log
/cvimmonha/...                          |
|  48 | 2021-02-01 14:32:24 |    10   | 0:02:55 | INSTALLED | DELETE_CVIM_POD |  7   | Success
|                                         |
|  47 | 2021-02-01 14:30:04 |    10   | 0:00:35 | INSTALLED | DELETE_CVIM_POD |  7   | Running |
step=HELM_INFRA                         |
```

© 2026 HCLTech. All rights reserved.

The following table gives the description of the components available in state transition table and operation history table:

Component	Description
SEQ	**Indicates the sequence number associated with each row. This number is monotonically increased for each entry added in the table.**
Timestamp	Denotes the timestamp associated with the entry (Local time zone).
Session	A monotonically increasing number assigned to each operation that is executed by an administrator using the *k8s_runner_db* CLI. This component is present in both tables to allow a cross-reference of the operations/steps with the state transition table. In the operations table, the entries for operations that have multiple steps share the same session number.
Elapsed	Indicates the time taken for completing each operation/step.
State	Indicates the initial step of the operation (first session entry) or the last state of each entry (all subsequent entries for the session).
Operation	Indicates the operation being performed.
Step	Indicates the steps executed for the current operation/session (if applicable). The step numbers for an operation are always increasing, but can have gaps, for example, Step 1 and Step 7.
Info	Provides additional information related to the operation or step: Command line arguments passed location of the log file or backup file step name

Installation Procedure

Once the cluster management node is installed and ready, configure the HA VIM-MON cluster with a candidate *setup_data.yaml* file, and run the HA

VIM-MON installation operation using the *k8s_runner* CLI. This install operation executes the following steps:
Validation
Verifies the hardware and software configuration.

Bootstrap Infra
Prepares the management node for VIM MON HA installation.

Setup Argus
Prepares bare metal installation.

Argus bare metal
Installs and configures the operating system on all configured cluster nodes.

VIM-MON Infra
Prepares the management node for Kubernetes and application installation.

Kubernetes Provisioner
Installs the Kubernetes infrastructure.

Helm Infra
Installs the HA VIM-MON application.

© 2026 HCLTech. All rights reserved.

You can list all steps using the *-l* argument:

```
# k8s_runner -l

CVIM-MON HA INSTALL STEPS

+-------------------------+---------------+
| Operations              | Operation ID  |
+-------------------------+---------------+
| VALIDATION              |       1       |
| BOOTSTRAP_INFRA         |       2       |
| SETUP_ARGUS             |       3       |
| ARGUS_BAREMETAL         |       4       |
| COMMON_CVIM_MON_INFRA   |       5       |
| KUBERNETES_PROVISIONER  |       6       |
| HELM_INFRA              |       7       |
+-------------------------+---------------+
#
```

Installing HA VIM-MON

The normal procedure to install HA VIM-MON is to run a complete installation where all the steps are executed. This operation is only allowed in the UNINSTALLED state.

Run the *k8s_runner* CLI passing *install* and *setupfile* arguments:

```
# k8s_runner --install --setupfile <path_to_candidate_setup_data_file>

                                ############################hvim
                                MON HA ORCHESTRATOR
                                ############################


[1/5][VALIDATION: INIT]                                                                                    [      -      ]        0min 12secs

  Input File Validations!
+-----------------------------------+--------+-------------------------------------------- +
| Rule                                | Status | Error |
+-----------------------------------+--------+-------------------------------------------- +
| Schema Validation of Input File     |    PASS    | None  |
| Check for Valid Keys                |    PASS    | None  |
| Valid Operation Check               |    PASS    | None  |
| Check for duplicate keys            |    PASS    | None  |
| Check hvim-Mon Target Nomenclature   |    PASS     | None  |
| Check duplicate hvim-Mon target      |    PASS     | None  |
| Information                          '                |            |
| Check Argus api network information |          PASS      | None  |
| Check Argus management network      |     PASS      | None  |
| information                          |                |            |
| Check Argus api network information |          PASS      | None  |
| Check Argus management network      |     PASS      | None  |
| information                         |                |            |
| Pod operations for hvim-MON-HA      |    PASS    |           |
+-----------------------------------+--------+-------------------------------------------- +
[1/5][VALIDATION: INIT]                                                                                    [      \      ]        0min 50secs
...
<snip>
```

The installation is considered complete when the HA VIM-MON application installation is completed successfully.

Installation Failure

Installation can fail for various reasons. The most common cause of failure is the use of incorrect setup data configuration. Most of the errors are detected in Step 1, but other errors can be detected at a later step.

© 2026 HCLTech. All rights reserved.

The failure recovery procedure varies based on the cluster state when the installation fails.

If the failure occurs in the early stage (Step 1 to Step 3) and the cluster is still in *the* installation. If the failure occurs in the later stage (Steps 4 to Step 7) and the cluster is in *Failed* state, contact TAC support.

External Servers

Prerequisites for Monitoring External Servers Using HA VIM MON

You can monitor the external servers such as standalone Linux servers that are not managed by any VIM pod, using the HA VIM-MON stack. You must provide the server name, and its IP address followed by port 9273 in the setup data file.

The servers must meet the following prerequisites:

- Prometheus collects the server metrics over unauthenticated and unencrypted HTTP connections on port 9273.
- VIM distinguishes the metrics collected from external servers from the VIM pod metrics by a node_type label value of external. By default, VIM
- associates the metrics for all CPUs with the label tag set to host. You can customize this with additional steps during installation.

Default built-in alerting rules and custom alerting rules equally apply to external nodes unless restricted to certain node types in the rule. For more information on how HA VIM-MON stack monitors external servers, see VIM Monitor

Resources

Resources for Managing HA VIM-MON Cluster

- Configuration File and Secrets
- List Service Endpoint URLs (--get-endpoint)
- Kubernetes Resources

You can manage your HA VIM-MON cluster using various commands.

Configuration File and Secrets

The reference setup data file is available at:

```
~/openstack-configs/setup_data.yaml
```

The secrets are saved under:

```
~/openstack-configs/secrets.yaml
```

- The *secrets.yaml* file is readable only from the root. This file contains the username and password for accessing Grafana, Prometheus, and Alertmanager for each stack.
- An example of *secrets.yaml* file is given below:

© 2026 HCLTech. All rights reserved.

Prometheus-Password-hvimmon monitor(Username:admin):YyZM5f3DdyCKxklw1vIN4i0l0M2EoRbkjb+UKm0Sa5Y=
Grafana-Password-hvimmon-monitor(Username:admin):59QRzzo+PEedz8MDfdX26+DoaMJ/OgVgoqGWdUhDS78=
Grafana-Password-hvimmon-monitor(Username:hvimon):Hhp2TZJNZCu28pIzW3F6huBKtIdH1VedtZxaC7P9dFs=
Grafana-Password-stack1(Username:admin):h72DhjEnVS/Rr4nFCZmmxKRmuK/t7qjyZJJrFbTyCtM=
Grafana-Password-stack1(Username:hvimmon):J72ChjJnVS/gh3yFCZmmxKRmuK/t9qjyZJJrFbJYCtN= Prometheus-Password-stack1(Username:admin):1Ph5AI8JUhiHgX0vjHB3W0Wzgjy2jWfiC5egAQJuuIs=
Grafana-Password-stacktsdb(Username:admin):N/ABGdX0ym5VhJ7Q/k8TO1oeqRXuzvbOmU9JeunA1As=
Grafana-Password-stacktsdb(Username:hvimmon):X/Sjcjasa9hsda23/k8TO1keqRXuzvb9mU9JfunA1Cs=
Prometheus-Password-stacktsdb(Username:admin):F8SPq+1qUSKM08EvlL+bTbL6RU8BI8Qcz/Yjzi0s7gw=

⚠From VIM 3.4.6, *cvimmon* users are read-only users for Grafana, by default. If needed, you can configure for an existing stack as well.

Post update to cvim, 3.4.6 execute *--regenerate-secrets* option with k8s_runner. For new stacks, sse *--add-stack* to add *cvimmon* user for Grafana access.

List Service Endpoint URLs (--get-endpoint)

To get information about all available service endpoint URLs, use the following command:

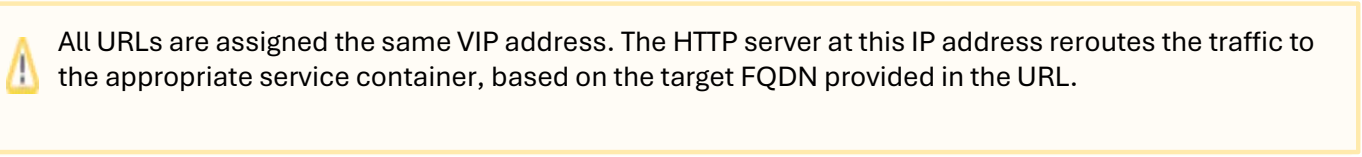
```
 # k8s_run --get-endpoint
2021-01-15 02:31:05,280 INFO Runner - Executing: GET_ENDPOINT (args: --get-endpoint)

+----------------------------------+-----------------------------------------------------
+      +
| Endpoint                 | Endpoint FQDN      |VIP  |
+----------------------------------+-----------------------------------------------------
+      +
| prometheus-hvimmon-monitor-server | https://cvmon-queensland-hvimmon-prometheus.cisco.com  |
2001:420:293: 2422::168 |
| grafana   | https://cvmon-queensland-hvimmon-grafana.cisco.com    |2001:420:293: 2422::168 |
| prometheus-thanos-querier-ingress | https://cvmon-queensland-thanos.cisco.com  |2001:420:293: 2422::168 |
| prometheus-thanos-ruler-ingress | https://cvmon-queensland-thanos-ruler.cisco.com     |2001:420:293: 2422::168 |
| thanos-alertmanager     | https://cvmon-queensland-thanos-alertmanager.cisco.com | 2001:420:293: 2422::168 |
| prometheus-scalestack-server      | https://hvimmon-prometheus-scalestack.cisco.com   |2001:420:293: 2422::168 |
| prometheus-stackresync-server   | https://hvimmon-prometheus-stackresync.cisco.com|2001:420:293: 2422::168 |
| minio      | https://cvmon-queensland-minio.cisco.com        |2001:420:293: 2422::168 |
+----------------------------------+-----------------------------------------------------
+      +
2021-01-15 02:31:07,321 INFO Runner - Operation GET_ENDPOINT completed successfully #
```

Endpoint	Description and usage	Typical use
Grafana	Access the unified Grafana dashboard for all stacks	most common / browser
prometheus-thanos-querier-ingress	Send PROMQL queries to retrieve metrics from the TSDB	most common / browser or REST API
prometheus-thanos-ruler-ingress	Evaluation of alerting and recording rules	less common / browser

© 2026 HCLTech. All rights reserved.

Endpoint	Description and usage	Typical use
thanos-alertmanager	View alerts, silence alerts	common / browser or REST API
prometheus-hvimmon-monitor-server	Access prometheus server in charge of the HA VIM-MON cluster	Reserved for troubleshooting/tech support
prometheus-<stack>-server	Access prometheus server associated with the provided stack	Reserved for troubleshooting/tech support
minio	Access to Minio storage cluster	Reserved for troubleshooting/tech support



Kubernetes Resources

To list Kubernetes nodes, use the following command:

```
# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
cvmonq1	Ready	master	24d	v1.17.8
cvmonq2	Ready	master	24d	v1.17.8
cvmonq3	Ready	master	24d	v1.17.8

In the above example, the Kubernetes cluster has three master nodes and no worker node.
To get the status of the cluster, use the following command:

```
# kubectl cluster-info
Kubernetes master is running at https://[2001:420:293:2422::168]:6443
KubeDNS is running at https://[2001:420:293:2422::168]:6443/api/v1/namespaces/kube-system/services/kube-dns:dns
/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Pod Operations

Supported Pod Operations

- Replacing a Controller Node (--replace-controller)

© 2026 HCLTech. All rights reserved.

- Adding a Worker Node
  - Configuring the Setup data File for Worker Node
  - Add Worker Command (--add-worker)
- Removing a Worker Node (--remove-worker)
- Regenerating Certificates
  - Kubernetes Certificates (--renew-k8s-certs)
  - ETCD Certificates (--renew-etcd-certs)
  - Self-signed Application Certificates (--regenerate-certs)
  - Custom Application Certificates
- Regenerating Secrets
  - View Secrets (--list-secrets)
  - Regenerate Secrets (--regenerate-secrets)
  - Custom Secrets (--set-secrets)
- Regenerating oob_parameters (--reconfigure-oob-creds)
- Global Parameters Reconfiguration (--reconfigure)

> ⓘ All VIM-MON pod-operations are wrapped in a tmux session in order to avoid failures. Once the operation is finished, the session remains open until it is manually closed with *exit*.

Replacing a Controller Node (--replace-controller)

You must replace a controller node, if there are hardware issues, power failure, disk failure, and so on. HA VIM-MON provides an option to replace the controller node to help in the recovery of the controller node. A controller node can be online or offline during controller replace operation.

The conditions for replacing the controller node are given below:

You must not change the setup data especially the node details for the replace node option.

- The replaced node and the defective node must have the same CIMC or BMC version.
- You can replace only one controller node at a time. You cannot use the replaced node if more than one controller node is defective.

When you replace a node, the controller node is removed from the Kubernetes cluster and replaced with a new controller node with the same name and the hardware details.

The following example shows how to replace a controller node:

```
# k8s_runner --replace-controller <node_name> --setupfile <path_to_candidate_setup_data_file>
[root@queensland ~]# k8s_runner --replace-controller cvmonq1 --setupfile ~/Save/new_setup_data.yaml
```

Adding a Worker Node

You can add a worker node to the HA VIM-MON cluster in the following two ways:

© 2026 HCLTech. All rights reserved.

- Pre-define in the initial *setup data* file so that the worker node is installed when you install the HA VIM-MON cluster.
- Add the worker node post-deployment using the *--add-worker* option, in a *candidate setup data* file.

Following are the conditions for adding the worker node to the HA VIM-MON cluster:

- You must add the hardware details of the worker node in a *candidate setup data* file (Argus bare metal section).
- You can add only one worker at a time to the HA VIM-MON cluster.
- The worker node hardware and the network hardware must conform to the same BOM defined by the HA VIM-MON master nodes.
- You must apply the same networking schema of the HA VIM-MON masters for all worker nodes.

Configuring the Setup data File for Worker Node

You must use the following schema for defining a worker node in a *setup data* file. The same schema is used for the master nodes except for the role field. You must explicitly define the role field as a worker for the worker node.

```
- name: Worker1
            oob_ip: 10.10.10.10
            role: worker
            ip_address:
              management_1_v4: 10.10.11.10/24
              management_1_gateway_v4: 10.10.11.1
              api_1_v4: 10.10.12.10/24
              api_1_gateway_v4: 10.10.12.1
```

Add Worker Command (--add-worker)

To add the worker to the HA VIM-MON cluster, use the following command:

k8s_runner --add-worker <node_name> --setupfile <path_to_candidate_setup_data_file>
[root@matrix ~]# k8s_run --add-worker matrix04 --setupfile /root/setupdata/setupdata_snmp.yaml
2021-01-12 16:54:38,990 INFO Runner - Executing: ADD_WORKER (args: --add-worker matrix04 --setupfile /root /setupdata/setupdata_snmp.yaml)
2021-01-12 16:54:39,055 INFO Runner - Adding Worker(s) matrix04

############################
VIM MON HA ORCHESTRATOR
############################

[1/5][VALIDATION: INIT] [-] 0min 2secs

Management Node Validations!

Rule	Status	Error
Check Kernel Version	PASS	None
...	...	
Check RHEL Pkgs Install State	PASS	None

[1/5][VALIDATION: INIT]] 0min 14secs

Input File Validations!

Rule	Status	Error
Schema Validation of Input File	PASS	None

© 2026 HCLTech. All rights reserved.

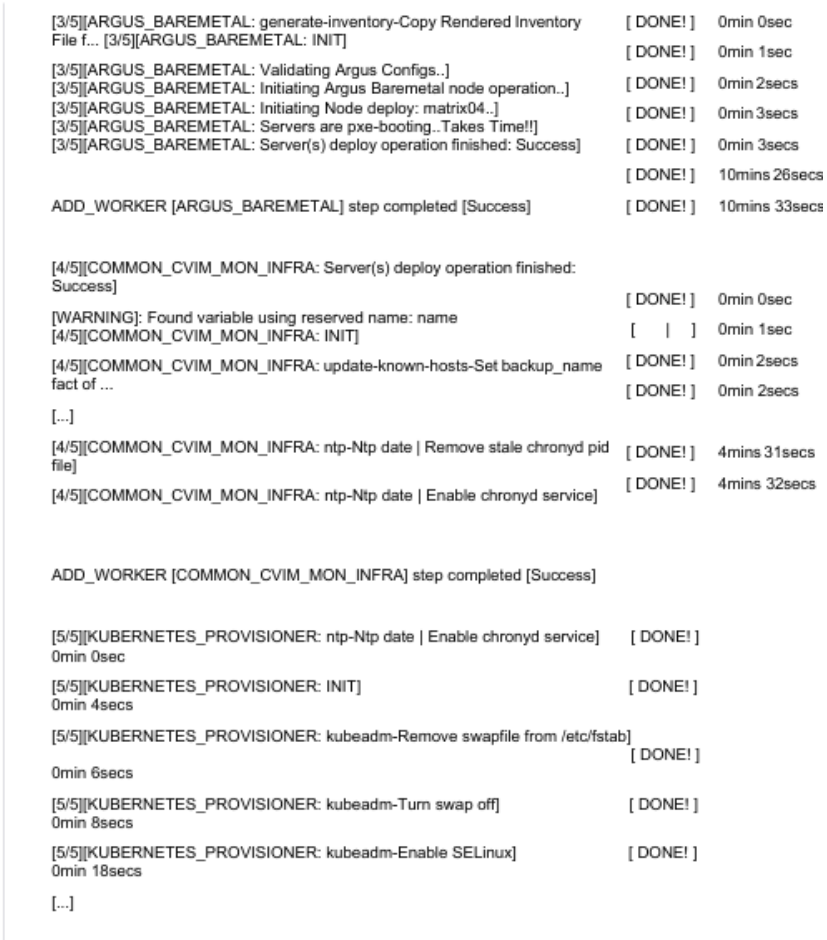
```
| ...   | ... |       |
| CVIM MON HA Check                    |  PASS  | None  |
+--------------------------------------+--------+ +
[1/5][VALIDATION: INIT]  [       \       ]       1min 13secs

UCS Hardware Validations
+-----------------------------------+--------+   +
| UseCase        | Status | Failure Reason |
+-----------------------------------+--------+   +
| CIMC Firmware Version Check            |  PASS  | None |
| PCIe Slot: HBA Status Check        |  PASS  | None |
| Redfish Enabled Check              |  PASS  | None |
| Argus NIC Adapter Check                |  PASS  | None |
| Disk Quantity and Uniformity Check     |  PASS  | None |
+-----------------------------------+--------+   +
[1/5][VALIDATION: INIT]                                          [ DONE! ]  1min 14secs

ADD_WORKER [VALIDATION] step completed [Success]

[2/5][GENERATE_INVENTORY: INIT]                                  [ DONE! ]  0min 0sec
[WARNING]: Found variable using reserved name: namees Time!!]        [       | ]     0min 1sec
[2/5][GENERATE_INVENTORY: Get Artifacts Phase...Takes Time!!][ DONE! ]  0min 2secs
[2/5][GENERATE_INVENTORY: generate-inventory-Check if Argus Site File is...
                        [ DONE! ]  0min 2secs

ADD_WORKER [GENERATE_INVENTORY] step completed [Success]
```



```
[3/5][ARGUS_BAREMETAL: generate-inventory-Copy Rendered Inventory File f... [3/5][ARGUS_BAREMETAL: INIT]   [ DONE! ]  0min 0sec
[3/5][ARGUS_BAREMETAL: Validating Argus Configs..]                          [ DONE! ]  0min 1sec
[3/5][ARGUS_BAREMETAL: Initiating Argus Baremetal node operation..]         [ DONE! ]  0min 2secs
[3/5][ARGUS_BAREMETAL: Initiating Node deploy: matrix04..]                  [ DONE! ]  0min 3secs
[3/5][ARGUS_BAREMETAL: Servers are pxe-booting..Takes Time!!]
[3/5][ARGUS_BAREMETAL: Server(s) deploy operation finished: Success]        [ DONE! ]  0min 3secs
                                                                            [ DONE! ]  10mins 26secs
ADD_WORKER [ARGUS_BAREMETAL] step completed [Success]                       [ DONE! ]  10mins 33secs


[4/5][COMMON_CVIM_MON_INFRA: Server(s) deploy operation finished: Success]
                                                                            [ DONE! ]  0min 0sec
[WARNING]: Found variable using reserved name: name
[4/5][COMMON_CVIM_MON_INFRA: INIT]                                          [   |   ]  0min 1sec
[4/5][COMMON_CVIM_MON_INFRA: update-known-hosts-Set backup_name fact of ...  [ DONE! ]  0min 2secs
                                                                            [ DONE! ]  0min 2secs
[...]
[4/5][COMMON_CVIM_MON_INFRA: ntp-Ntp date | Remove stale chronyd pid file]  [ DONE! ]  4mins 31secs
[4/5][COMMON_CVIM_MON_INFRA: ntp-Ntp date | Enable chronyd service]         [ DONE! ]  4mins 32secs


ADD_WORKER [COMMON_CVIM_MON_INFRA] step completed [Success]


[5/5][KUBERNETES_PROVISIONER: ntp-Ntp date | Enable chronyd service]  [ DONE! ]
0min 0sec
[5/5][KUBERNETES_PROVISIONER: INIT]                                  [ DONE! ]
0min 4secs
[5/5][KUBERNETES_PROVISIONER: kubeadm-Remove swapfile from /etc/fstab]
                                                                     [ DONE! ]
0min 6secs
[5/5][KUBERNETES_PROVISIONER: kubeadm-Turn swap off]                 [ DONE! ]
0min 8secs
[5/5][KUBERNETES_PROVISIONER: kubeadm-Enable SELinux]                [ DONE! ]
0min 18secs
[...]
```

© 2026 HCLTech. All rights reserved.

```
[5/5][KUBERNETES_PROVISIONER: telegraf-Add Telegraf to Docker Users]      [ DONE! ]
5mins 36secs
[5/5][KUBERNETES_PROVISIONER: telegraf-Restart service Telegraf]          [ DONE! ]
5mins 39secs
ADD_WORKER [KUBERNETES_PROVISIONER] step completed [Success]



2021-01-12 17:16:44,749 INFO Runner - Backing up setup_data.yaml
2021-01-12 17:16:44,778 INFO Runner - Operation ADD_WORKER completed successfully
2021-01-12 17:16:44,778 INFO Runner - Performing autobackup...
2021-01-12 17:16:44,806 INFO Runner - Executing autobackup to
/var/cisco/cvimmonha_autobackup
/cvimmonha_autobackup_3.9.8_2021-01-12_17:16:44, disconnected install=False
2021-01-12 17:16:44,807 INFO Runner - Validating the current workspace...
2021-01-12 17:16:44,807 INFO Runner - Creating the backup...
2021-01-12 17:16:44,878 INFO Runner - Compressing the backup...
2021-01-12 17:16:45,093 INFO Runner - Backup completed successfully:
/var/cisco/cvimmonha_autobackup
/cvimmonha_autobackup_3.9.8_2021-01-12_17:16:44.tgz
The logs for this run are available at /var/log/cvimmonha/add_worker_2021-01-12_16:54:38
```

ADD_WORKER [COMMON_VIM_MON_INFRA] step completed [Success]			
[5/5][KUBERNETES_PROVISIONER: ntp-Ntp date | Enable chronyd service]	[DONE!]	0min	0sec
[5/5][KUBERNETES_PROVISIONER: INIT]	[DONE!]	0min	4secs
[5/5][KUBERNETES_PROVISIONER: kubeadm-Remove swapfile from /etc/fstab]	[DONE!]	0min	6secs
[5/5][KUBERNETES_PROVISIONER: kubeadm-Turn swap off]	[DONE!]	0min	8secs
[5/5][KUBERNETES_PROVISIONER: kubeadm-Enable SELinux] [...]	[DONE!]	0min	18secs

Removing a Worker Node (--remove-worker)

You can remove a worker node from the cluster if required. An HA VIM-MON cluster can operate without any worker nodes. You can also remove all the worker nodes. After this operation, VIM deletes the node from the HA VIM-MON Kubernetes cluster. All the running pods are automatically migrated to the other workers or masters.

The conditions for removing a worker node are given below:

- You can remove only one worker node at a time.
- You must delete the node details of the worker node from the *candidate setup data* file, before executing the remove worker operation.

To remove the worker node, use the following command

© 2026 HCLTech. All rights reserved.

```
# k8s_runner --remove-worker <node_name> --setupfile <path_to_candidate_setup_data_file>

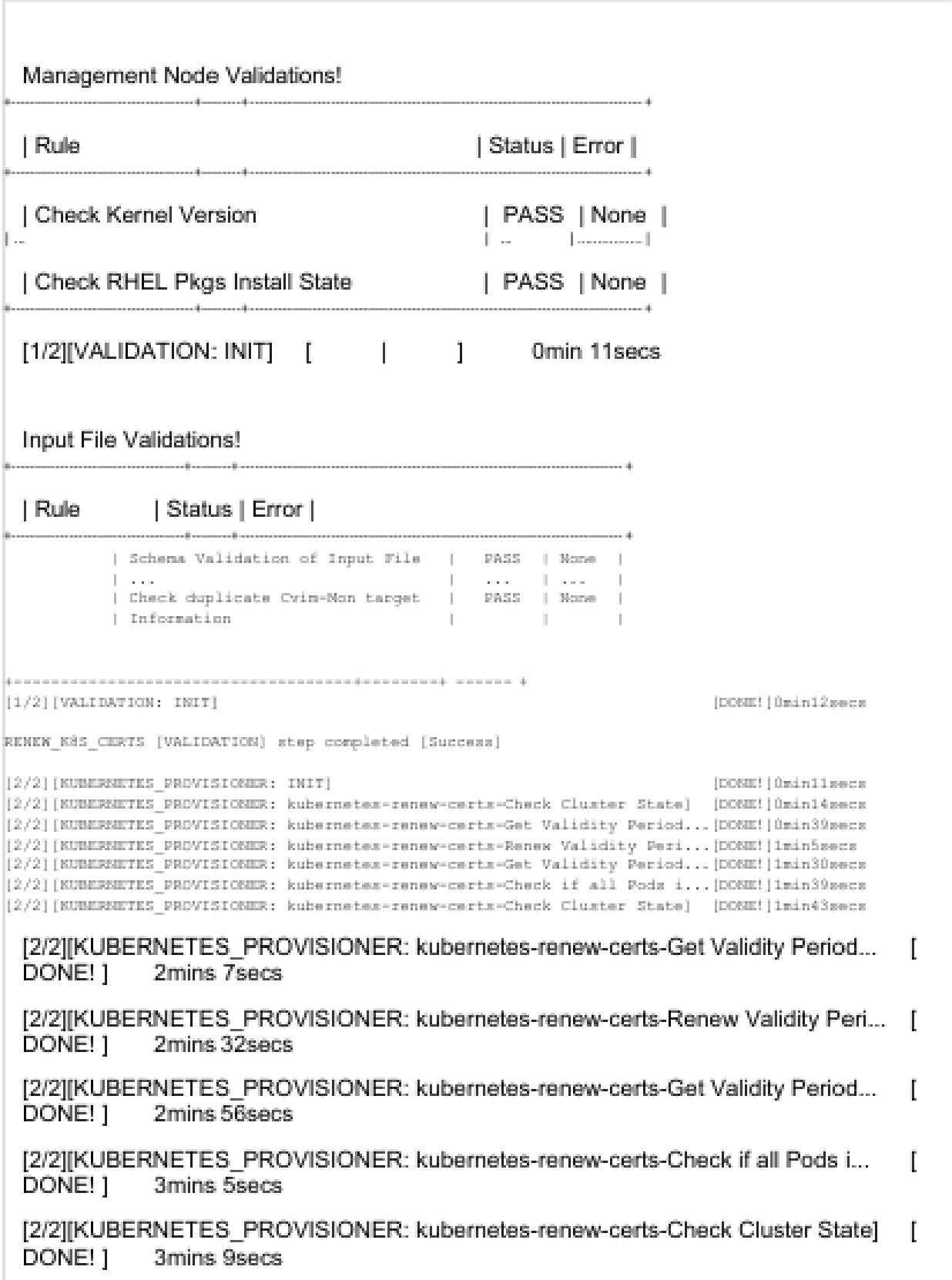
[root@matrix ~]# k8s_run --remove-worker matrix04 --setupfile /root/setupdata/setupdata_nw.yaml

2021-01-12 16:51:17,700 INFO Runner - Executing: REMOVE_WORKER (args: --remove-worker matrix04 --setupfile
/root
/setupdata/setupdata_nw.yaml)
2021-01-12 16:51:17,798 INFO Runner - Removing Worker(s) matrix04

                        ############################
                        # hvim MON HA ORCHESTRATOR
                        ############################

[1/4][VALIDATION:                                                          [   -   ]    0min
```

© 2026 HCLTech. All rights reserved.



Management Node Validations!

Rule	Status	Error
Check Kernel Version	PASS	None
Check RHEL Pkgs Install State	PASS	None

[1/2][VALIDATION: INIT] [|] 0min 11secs

Input File Validations!

Rule	Status	Error
Schema Validation of Input File	PASS	None
...	...	...
Check duplicate Cvim-Mon target Information	PASS	None

[1/2][VALIDATION: INIT] [DONE!]0min12secs

RENEW_K8S_CERTS [VALIDATION] step completed [Success]

[2/2][KUBERNETES_PROVISIONER: INIT] [DONE!]0min11secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check Cluster State] [DONE!]0min14secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period...[DONE!]0min39secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Renew Validity Peri...[DONE!]1min5secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period...[DONE!]1min30secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check if all Pods i...[DONE!]1min39secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check Cluster State] [DONE!]1min43secs

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period... [DONE!] 2mins 7secs

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Renew Validity Peri... [DONE!] 2mins 32secs

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period... [DONE!] 2mins 56secs

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check if all Pods i... [DONE!] 3mins 5secs

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check Cluster State] [DONE!] 3mins 9secs

© 2026 HCLTech. All rights reserved.

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period... [DONE!] 3mins 34secs

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Renew Validity Peri... [DONE!] 4mins 0sec

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period... [DONE!] 4mins 24secs

[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check if all Pods i... [DONE!] 4mins 25secs

RENEW_K8S_CERTS [KUBERNETES_PROVISIONER] step completed [Success]

2021-01-12 15:40:40,405 INFO Runner - Operation RENEW_K8S_CERTS completed successfully

The logs for this run are available at /var/log/cvimmonha/renew_k8s_certs_2021-01-12_15:35:59

⚠ Ensure that the kubectl commands continue to work post renew-k8s-certs operation.
For any unauthorized error during kubectl command execution, update the *admin.conf* in the management node.
Copy */etc/kubernetes/admin.conf* from Kubernetes master node to */root/.kube/config* in the HA management node.

ETCD Certificates (--renew-etcd-certs)

To regenerate ETCD certificates, use the following command:

```
root@queensland ~]# k8s_runner --renew-etcd-certs

2021-01-12 15:41:37,523 INFO Runner - Executing: RENEW_ETCD_CERTS (args: --renew-etcd-certs)
2021-01-12 15:41:40,327 INFO Runner - Running generic handler for RENEW_ETCD_CERTS

                    ##############################
                      hvim MON HA ORCHESTRATOR
                    ##############################

1/2|[VALIDATION: INIT]                                                    |   /   |    0min 6secs

 Management Node Validations!
+-------------------------------------------+--------+-------+
 Rule                                      | Status | Error |
+-------------------------------------------+--------+-------+
 Check Kernel Version                      |  PASS  | None  |
 ...                                       |  ...   |.......|
 Check RHEL Pkgs Install State             |  PASS  | None  |
+-------------------------------------------+--------+-------+
1/2|[VALIDATION: INIT]                                                    |   |   |    0min 11secs

Input File Validations!
+-------------------------------------+--------+-------+
 Rule                                | Status | Error |
+-------------------------------------+--------+-------+
```

© 2026 HCLTech. All rights reserved.

```
| Schema Validation of Input File   |   PASS  | None  |
| ...                               |   ...   | ...   |
| Check duplicate Cvim-Mon target   |   PASS  | None  |
| Information                       |         |       |


+-----------------------------------+---------+ ------ +
[1/2][VALIDATION: INIT]                                                    [ DONE! ]

RENEW_ETCD_CERTS [VALIDATION] step completed [Success]

[2/2][KUBERNETES_PROVISIONER: INIT]                                        [ DONE! ]
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Configure | Check if etcd clu...  [ DONE! ]
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Get Validity Period for all k...  [ DONE! ]
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Renew Validity Period for all...  [   |   ]
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Get Validity Period for all k...  [ DONE! ]

RENEW_ETCD_CERTS [KUBERNETES_PROVISIONER] step completed [Success]
```

Self-signed Application Certificates (--regenerate-certs)

To regenerate self-signed application certificates, use the following command:

```
# k8s_runner --regenerate-certs --setupfile <path_to_candidate_setup_data_file>
[root@queensland ~]# k8s_run --regenerate-certs --setupfile Save/setup_data_certs.yaml


2021-01-12 15:46:35,738 INFO Runner - Executing: REGENERATE_CERTS (args: --
regenerate-certs --setupfile Save

/setup_data_certs.yaml)

2021-01-12 15:44:21,396 INFO Runner - Running generic handler for REGENERATE_CER


##############################
VIM MON HA ORCHESTRATOR
##############################


[1/2][VALIDATION: INIT]    [         -         ]         0min 2secs


  Management Node Validations!
+-------------------------------------+--------+ ------------------------------------+
  | Rule                                          | Status | Error |
+-------------------------------------+--------+ ------------------------------------+
  | Check Kernel Version                          |  PASS  | None  |
| ...                                             |  ...   | ..... |
```

© 2026 HCLTech. All rights reserved.

```
| Check RHEL Pkgs Install State    | PASS  | None  |
+-----------------------------------+--------+ -----------------------------------+

[1/2][VALIDATION: INIT]    [          |          ]          0min 11secs


Input File Validations!
+-----------------------------------+--------+ ----------------------------------- +

| Rule                                          | Status | Error |
+-----------------------------------+--------+ ----------------------------------- +
              | Schema Validation of Input File  |  PASS  | None  |
              | ...                              |  ...   | ...   |
              | Check duplicate Cvim-Mon target  |  PASS  | None  |
              | Information                      |        |       |


+-----------------------------------+--------+ ---------------------- +
[1/2][VALIDATION: INIT]                                                  [  DONE  ]   0min 12secs
                                                                                      !

REGENERATE_CERTS [VALIDATION] step completed [Success]
[2/2][HELM_INFRA: INIT]                                                  [  DONE  ]   0min 4secs
                                                                                      !
[2/2][HELM_INFRA: nginx-ingress-controller-Check whether helm binary exi... [  DONE  ]   0min 5secs
                                                                                      !
[2/2][HELM_INFRA: nginx-ingress-controller-List installed Helm charts.]  [  DONE  ]   0min 5secs
                                                                                      !
[2/2][HELM_INFRA: nginx-ingress-controller-Get Helm Parent Directory]    [  DONE  ]   0min 6secs
                                                                                      !
[...]
[2/2][HELM_INFRA: prometheus-CVIMMON-stackresync->Replace CVIM-          [  DONE  ]   3mins 21secs
MON                                                                                   !
Ingre...      [2/2][HELM_INFRA: prometheus-CVIMMON-stackresync-          [  DONE  ]   3mins 21secs
>Replace CVIM-                                                                        !
MON Ingre...      [2/2][HELM_INFRA: prometheus-CVIMMON-stackresync-      [  DONE  ]   3mins 28secs
>Delete                                                                              !
  all Cert Sensit...


  REGENERATE_CERTS [HELM_INFRA] step completed [Success]

  2021-01-12 15:55:35,202 INFO Runner - Backing up setup_data.yaml

  2021-01-12 15:55:35,206 INFO Runner - Operation REGENERATE_CERTS completed
  successfully 2021-01-12 15:55:35,207 INFO Runner - Performing autobackup...

  2021-01-12 15:55:35,271 INFO Runner - Executing autobackup to
  /var/cisco/cvimmonha_autobackup

  /hvimmonha_autobackup_3.9.5_2021-01-12_15:55:35, disconnected install=False 2021-01-12
  15:55:35,271 INFO Runner - Validating the current workspace...

  2021-01-12 15:55:35,333 INFO Runner - Creating the backup... 2021-01-12 15:55:35,456
  INFO Runner - Compressing the backup...
```

© 2026 HCLTech. All rights reserved.

```
2021-01-12 15:55:35,793 INFO Runner - Backup completed successfully:
/var/cisco/cvimmonha_autobackup

/cvimmonha_autobackup_3.9.5_2021-01-12_15:55:35.tgz

The logs for this run are available at /var/log/cvimmonha/regenerate_certs_2021-01-
12_15:51:51
```

Custom Application Certificates

Get Endpoint Command (--get-endpoint)

With central VIM-MON, you can provide trusted CA signed x509 certificates which are used to access the Prometheus, Grafana, and Alertmanager applications in the HA VIM-MON cluster. If certificates are not provided, self-signed certificates are generated for these URLs by default. To find the ingress URLs configured for Prometheus, Grafana, and Alertmanager in each stack, execute the following command:

```
[root@queensland ~]# k8s_runner --get-endpoint


2021-01-12 13:05:14,471 INFO Runner - Executing: GET_ENDPOINT (args: --get-endpoint)

+-------------------------+-------------------------------------------------
+-----------------------------------------------+

| Endpoint                | Endpoint FQDN        | VIP     |
+-------------------------+-------------------------------------------------
+-----------------------------------------------+

| prometheus-cvimmon-monitor-server | https://cvmon-queensland-cvimmon-
prometheus.cisco.com  | 2001:420:293: 2422::168 |

| grafana      | https://cvmon-queensland-cvimmon-grafana.cisco.com     | 2001:420:293:
2422::168 |

| prometheus-thanos-querier-ingress | https://cvmon-queensland-thanos.cisco.com       |
2001:420:293: 2422::168 |

| prometheus-thanos-ruler-ingress  | https://cvmon-queensland-thanos-ruler.cisco.com  |
2001:420:293: 2422::168 |

| thanos-alertmanager         | https://cvmon-queensland-thanos-alertmanager.cisco.com |
2001:420:293: 2422::168 |

| prometheus-scalestack-server      | https://hvimmon-prometheus-scalestack.cisco.com
        | 2001:420:293: 2422::168 |

| prometheus-stackresync-server    | https://hvimmon-prometheus-stackresync.cisco.com
        | 2001:420:293: 2422::168 |

  | minio          | https://cvmon-queensland-minio.cisco.com | 2001:420:293: 2422::168 |
+-------------------------+-------------------------------------------------
+-----------------------------------------------+


2021-01-12 13:05:17,349 INFO Runner - Operation GET_ENDPOINT completed successfully
```

© 2026 HCLTech. All rights reserved.

Based on the result, you can use the application FQDN's as Common Name (CN) or alternative DNS names in the x509 certificate. The central VIM-MON supports two types of custom x509 Certificates:

- Domain level wildcard certificates
- Stack level certificates

A trusted CA signed certificate comprises of multiple files (root CA certificate, intermediate CA certificate, application certificate, and certification key). A Privacy Enhanced Mail (PEM) bundle comprising of all these files must be created using the following steps:

```
$ cat user-ca.crt >> user-bundle.pem ( Root CA and intermediate CA  )(Optional)
$ cat user.crt >> user-bundle.pem ( Certificate file )
$ cat user.key >> user-bundle.pem ( Key File )
$ mv user-bundle.pem /root/cvimha_certs
```

⚠ Private key of a trusted CA signed certificate must not have any passphrase (non-encrypted key)

Domain Level Wildcard Certificates

A wildcard certificate can support multiple subdomains for a particular domain. To generate this certificate, ensure that the following conditions are met:

- Must match the cvimmon_domain_suffix defined in the setup data file.
- Must be defined at a global level in the setup data file so that it can be used to access Grafana, Prometheus, and Alertmanager of all VIM-MON stacks.

An example of wildcard certificate is provided below:

```
$ grep cvimmon_domain_suffix /root/openstack-configs/setup_data.yaml
cvimmon_domain_suffix: lab.test.com

$ openssl x509 -in /root/cvimha_certs/user-bundle.pem -text -noout
. . .
Subject: C=US, ST=California, L=San Jose, O=IT, CN=*.lab.test.com
```

To define a domain level custom certificate, edit the *setup data* file and add the path to the certificate as given below:

```
$ grep cvimmon_domain /root/openstack-configs/setup_data.yaml
cvimmon_domain_suffix: lab.test.com
cvimmon_domain_ca_cert: /root/cvimha_certs/user-bundle.pem
```

⚠

- You can switch from self-signed certificate to custom certificates or vice versa using the *--regenerate-certs* command line option. You can add
- the appropriate certificate path in the *candidate setup data* file and run the *--regenerate-certs* for using the certificate.
- If you remove the certificate path from the *candidate setup data* file and re-run the *k8s_runner --regenerate-certs,* you are switched back to self-signed certificates.

Regenerating Secrets

During the deployment of VIM MON HA cluster Grafana, Prometheus and AlertManager passwords are auto generated. These passwords are for admin user and can be used to access respective dashboards. Additional CLI's have been provided with *k8s_runner.py* script to list passwords for each stack and regenerate passwords.

To configure passwords for above services, use the following:

- Regenerate Secrets
- Custom Secrets

View Secrets (--list-secrets)

To view passwords for each application, first execute *list-secrets* command to fetch the secret keys.

© 2026 HCLTech. All rights reserved.

```
[root@queensland ~]# k8s_runner --list-secrets

2021-01-12 13:26:29,171 INFO Runner - Executing: LIST_SECRETS (args: --list-secrets)

+-----------------------------------------------------+
| Secret Keys                                         |
+-----------------------------------------------------+
| Grafana-Password-monitoring(Username:admin)         |
| Grafana-ReadOnly-Password-monitoring(Username:cvimmon) |
| Thanos-Password-monitoring(Username:admin)          |
+-----------------------------------------------------+
```

After fetching the secret keys, execute *get-password* command with secret key to view the configured password for the service.

Get Password Command (--get-password)

```
[root@queensland ~]# k8s_run --get-password "Grafana-Password-monitoring(Username:admin)"

2021-01-12 13:27:10,970 INFO Runner - Executing: GET_PASSWORD (args: --get-password Grafana-Password-monitoring
(Username:admin))

+-------------------------------------------+-----------------------------------------------+
|                 Secret Key                |                   Password                    |
+-------------------------------------------+-----------------------------------------------+
| Grafana-Password-monitoring(Username:admin) | 4n6AVQ85wbWgDwqQ1RSp/XqpEASANO2ieoICDGL+E3U= |
+-------------------------------------------+-----------------------------------------------+
```

⚠ Secret key must be provided in double quotes ("") with *--get-password* cli.

Regenerate Secrets (--regenerate-secrets)

To regenerate all passwords at once, use the following command:

[root@queensland ~]# k8s_runner --regenerate-secrets

2021-01-12 18:23:19,090 INFO Runner - Executing: REGENERATE_SECRETS (args: --regenerate-secrets) 2021-01-12 18:23:21,866 INFO Runner - Regenerating Application Secrets

############################## VIM MON HA ORCHESTRATOR ##############################

[1/2][VALIDATION: INIT] [|] 0min 3secs

Management Node Validations!

```
+-------------------------------------+--------+     +
| Rule        | Status | Error |
+-------------------------------------+--------+     +
| Check Kernel Version     |  PASS  | None  |
| ...     |  ...   |        |
```

© 2026 HCLTech. All rights reserved.

```
| Check RHEL Pkgs Install State       |  PASS  | None  |
+-------------------------------------+--------+     +
[1/2][VALIDATION: INIT]  [       |       ]       0min 11secs

Input File Validations!
+-------------------------------------+--------+       +
| Rule        | Status | Error |
+-------------------------------------+--------+       +

| Schema Validation of Input File
| ...
| Check duplicate Cvim-Mon target
|  PASS  | None  |
|  ...   | ...   |
|  PASS  | None  |
| Information    |       |       |
+-------------------------------------+--------+       +
[1/2][VALIDATION: INIT]  [ DONE! ] 0min 12secs
```

REGENERATE_SECRETS [VALIDATION] step completed [Success]

```
[2/2][HELM_INFRA: INIT]  [ DONE! ]  0min 4secs
[2/2][HELM_INFRA: nginx-ingress-controller-Check whether helm binary exi...  [ DONE! ]  0min 5secs
[2/2][HELM_INFRA: nginx-ingress-controller-List installed Helm charts.]  [ DONE! ]  0min 5secs
[2/2][HELM_INFRA: nginx-ingress-controller-Get Helm Parent Directory]  [ DONE! ]  0min 6secs [  ]
[2/2][HELM_INFRA: prometheus-CVIMMON-stackresync->Delete temporary auth ...  [ DONE! ]  11mins 18secs
[2/2][HELM_INFRA: prometheus-CVIMMON-stackresync->Save the password in s...  [ DONE! ]  11mins 22secs
```

REGENERATE_SECRETS [HELM_INFRA] step completed [Success]

2021-01-12 18:34:57,512 INFO Runner - Operation REGENERATE_SECRETS completed successfully 2021-01-12 18:34:57,514 INFO Runner - Performing autobackup...
2021-01-12 18:34:57,580 INFO Runner - Executing autobackup to /var/cisco/cvimmonha_autobackup /cvimmonha_autobackup_3.9.5_2021-01-12_18:34:57, disconnected install=False 2021-01-12 18:34:57,580 INFO Runner - Validating the current workspace...
2021-01-12 18:34:57,641 INFO Runner - Creating the backup... 2021-01-12 18:34:57,785 INFO Runner - Compressing the backup...
2021-01-12 18:34:58,109 INFO Runner - Backup completed successfully: /var/cisco/cvimmonha_autobackup /cvimmonha_autobackup_3.9.5_2021-01-12_18:34:57.tgz

The logs for this run are available at /var/log/cvimmonha/regenerate_secrets_2021-01-12_18:23:19

Custom Secrets (--set-secrets)
You can set custom passwords for Grafana and Thanos applications. To configure custom passwords, use the below command:

```
[root@queensland ~]# k8s_runner --list-secrets

2021-01-12 13:26:29,171 INFO Runner - Executing: LIST_SECRETS (args: --list-secrets)

+-----------------------------------------------------------+
| Secret Keys                                               |
+-----------------------------------------------------------+
| Grafana-Password-monitoring(Username:admin)               |
| Grafana-ReadOnly-Password-monitoring(Username:cvimmon)    |
| Thanos-Password-monitoring(Username:admin)                |
+-----------------------------------------------------------+
```

To change password, create a new yaml file and the key you want to set for password. For example:

To change password for *Grafana-Password-monitoring,* create a file and update the file with following:

```
Grafana-Password-monitoring(Username:admin): "<New Password>"
```

© 2026 HCLTech. All rights reserved.

> ⚠ Ensure that you use the same key name as shown in the output including *Username:admin,* otherwise the command is not executed.

Password Policy

- Only alphanumeric and special characters are allowed for secrets.
- Passwords must contain at least one letter, one special character, a digit, without blank spaces.
- Password length must be >=8 and <= 44.
- Allowed special characters [_@./#+-=]\

Save the yaml file where you entered the custom password and execute the following command:

```
[root@queensland ~]# k8s_runner --set-secrets /root/Save/new_secrets.
yaml
2021-01-12 13:13:06,369 INFO Runner - Executing: SET_SECRETS (args: --set-secrets /root/Save/new_secrets.yaml)

############################
VIM MON HA ORCHESTRATOR
 ############################

[1/2][VALIDATION: INIT]  [      /      ]      0min 6secs

Management Node Validations!
+-------------------------------------+--------+ +
| Rule       | Status | Error |
+-------------------------------------+--------+ +
| Check Kernel Version    |  PASS  | None  |
| ...   |   ... |      |
| Check RHEL Pkgs Install State      |  PASS  | None  |
+-------------------------------------+--------+ +
[1/2][VALIDATION: INIT]  [      |      ]      0min 11secs

Input File Validations!
+-------------------------------------+--------+   +
| Rule       | Status | Error |
+-------------------------------------+--------+   +
                | Schema Validation of Input File       |     PASS  | None   |
                | ...                                   |     ...       | ...        |
                | Check duplicate Cvim-Mon target   |     PASS  | None   |
                | Information                           |           |          |

                +-------------------------------------+--------+   +
                [1/2][VALIDATION: INIT]                                          [DONE!]0min 12secs
                SET_SECRETS [VALIDATION] step completed [Success]

                [2/2][HELM_INFRA: INIT]                                          [DONE!]0min 4secs
```

© 2026 HCLTech. All rights reserved.

```
[2/2][HELM_INFRA: nginx-ingress-controller-Check whether helm binary exi...  [DONE!]0min 5secs
[2/2][HELM_INFRA: nginx-ingress-controller-List installed Helm charts.]     [DONE!]0min 5secs
[2/2][HELM_INFRA: nginx-ingress-controller-Get Helm Parent Directory]       [DONE!]0min 6secs
[...]
[2/2][HELM_INFRA: prometheus-stackresync->Get Helm Parent Directory|        [DONE!]2mins 54secs
[2/2][HELM_INFRA: prometheus-stackresync->Check whether Helm App directo... [DONE!]2mins 54secs
[2/2][HELM_INFRA: prometheus-stackresync->Checking for Required Files an... [DONE!]3mins 3secs
SET_SECRETS [HELM_INFRA] step completed [Success]
```

2021-01-12 13:16:25,601 INFO Runner - Operation SET_SECRETS completed successfully 2021-01-12 13:16:25,603 INFO Runner - Performing autobackup...

2021-01-12 13:16:25,669 INFO Runner - Executing autobackup to /var/cisco/cvimmonha_autobackup /cvimmonha_autobackup_3.9.5_2021-01-12_13:16:25, disconnected install=False 2021-01-12 13:16:25,670 INFO Runner - Validating the current workspace...

2021-01-12 13:16:25,732 INFO Runner - Creating the backup... 2021-01-12 13:16:25,869 INFO Runner - Compressing the backup...

2021-01-12 13:16:26,207 INFO Runner - Backup completed successfully: /var/cisco/cvimmonha_autobackup /cvimmonha_autobackup_3.9.5_2021-01-12_13:16:25.tgz

The logs for this run are available at /var/log/cvimmonha/set_secrets_2021-01-12_13:13:06 [root@queensland ~]#

Regenerating oob_parameters (--reconfigure-oob-creds)

For deployment of VIM-MON HA cluster, you can reconfigure the following parameters on Day 2 as an option.

- *oob_username, oob_password* at the global or node level
- *oob_password_hash* for access via KVM console

Listed below are the steps on how to achieve the reconfiguration:

- Copy the VIM-MON HA setupdata from */root/openstack-configs/* into a local location.
- Update the *oob_username*, *oob_password,* and/or *password_hash*, and manually copy of the *setup_data* created from Step 1. You need to update the *common_info* section and the individual *oob_password* for each server and then run the reconfigure option:

```
ARGUS_BAREMETAL:
  common_info:
    :
    oob_password: <"new password">
    password_hash: <password_hash>

  SITE_CONFIG:
    clusters:
    - :
      name: <cluster_name>
      servers:
      - ip_address: <ip_address>
        [oob_password] : <oob_password>
  :
```

Ensure that the new password satisfies at least three of the following conditions:

- Must contain at least one lower case letter.
- Must contain at least one upper case letter.
- Must contain at least one digit between 0 to 9.

© 2026 HCLTech. All rights reserved.

- One of these special characters !$@%^-_+=
- Must be 8 to 20 characters long with no spaces

> ⚠ Do not change OOB password directly in the exiting *setup_data in /root/openstack-configs/*

Run the reconfiguration command to update the *setup_data* as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml.CVIMMONHA.yaml <my_setup_data_cvimmon_ha.yaml>
[root@mgmt1 ~]# cp <my_setup_data_cvimmon_ha.yaml> <setup_data.yaml.CVIMMONHA_original.yaml>
[root@mgmt1 ~]# vi my_setup_data_cvimmon_ha.yaml (update the relevant sections of the file)
[root@mgmt1 ~]# k8s_runner --reconfigure-oob-creds --setupfile <location of setup data file where these
changes were made>
```

> ⚠ After successful completion of the OOB Password, the reconfiguration triggers an auto-back if the management node auto-back recovery feature is enabled. If the *oob_password* reconfigure fails, contact Cisco TAC for recovery.

Global Parameters Reconfiguration (--reconfigure)

The following parameters can be reconfigured with the --reconfigure operation:

Parameter name	Description
log_rotation_frequency	Specifies the frequency of log rotation.
log_rotation_size	Indicates the maximum size of each log. A rotation occurs if the log size exceeds the maximum value.
log_rotation_del_older	Denotes the number of compressed archive log files to be maintained for each log file. The old archive log files are deleted.
CVIMMONHA_CLUSTER_MONITOR:SNMP	All parameters related to SNMP can be reconfigured with the limitation that *enabled* cannot be set to False once it is configured to True
SYSLOG_EXPORT	All parameters related to syslog export can be reconfigured.

To reconfigure any of the above parameters, you must update them in a candidate setup data file and run the *k8s_runner* command with the reconfigure option.

The following example shows how to use the reconfigure option:

k8s_runner --reconfigure --setupfile <path_to_candidate_setup_data_file>

© 2026 HCLTech. All rights reserved.

Stack Operations

Stack Operations

- Overview of HA VIM-MON Stacks
- Adding a Stack (--add-stack)
- Deleting a Stack (--delete-stack)
- Adding, Updating, and Deleting VIM Pod Targets in a Stack
  - Adding a Target VIM OpenStack Pod (--add-CVIM-pod)
  - Reconfiguring a VIM OpenStack Pod (--reconfigure-CVIM-pod)
  - Deleting a Target VIM OpenStack Pod (--delete-CVIM-pod)

Overview of HA VIM-MON Stacks

An HA VIM-MON stack is a set of containers responsible for collecting metrics from a set of VIM pods that are grouped in the same metro or different metros from the same region or from different regions.

At least one stack must be created to collect metrics from VIM pods. In most deployments, this single stack is sufficient to monitor all VIM pods. Only in few very large deployments, it may be necessary to create more than one stack to scale properly.

Each stack is identified by its name and provides the following services:
- Collection and short term storage of all the metrics generated by monitored VIM pods.
- Persistence of collected metrics in the Minio object store.

The scraping of these pods is randomly spaced within the interval window to balance the load both on the network and VIM pods.

Each stack runs two independent instances of Prometheus servers that collect metrics from the same set of VIM pods concurrently. As the Prometheus servers run on different cluster nodes, any node failure will therefore only impact one scraper. In steady nominal state, the metrics are duplicated in the Minio storage and are deduplicated at query time.

The collected metrics are retained in local memory and local storage for blocks of two hours before they are written to the Minio object store.

Scale limitations

The main load of a stack is related to the following activities:
- Collecting metrics from all the remote VIM pods at the configured interval (networking load).
- Process and store these metrics into the Minio object store (CPU and storage load).
- Handling metric queries coming from Grafana front-end or from external applications for short term metrics (CPU and storage load).

Depending on the total number of metrics to process, each stack can use a substantial amount of CPU, memory, storage, and network bandwidth. In the current version with the recommended HA cluster hardware:
- The recommended limit per stack is 8,000 nodes or 1,000 VIM pods - whichever comes first (using 1m sampling interval).
- The recommended maximum number of stacks per HA cluster is 1+m stacks per cluster, where m is the number of worker nodes (excluding the controller nodes).

For example, an HA cluster configuration with three controller nodes and three worker nodes can run up to four stacks for a total of 32,000 nodes or 4,000 VIM pods. Such large configuration can typically process in the order of 1 million metrics per second (or a quarter million metrics per second per stack).

Adding a Stack (--add-stack)

All stack operations are allowed only when the cluster is in the INSTALLED state.

© 2026 HCLTech. All rights reserved.

After a fresh installation, the cluster will have no stack defined and adding a new stack will be the first operation in order to monitor VIM pods.

You can execute the add-stack operation after adding 1 or more stack descriptors to the *cvim-mon-stacks* list in the *candidate setup data* file. Each stack name must be unique in the *reference setup data* file. The new stack being added can be configured with or without any VIM pod targets as VIM pod targets can be added subsequently (see the --add-cvim-pod operation).

A valid stack name can be any alphanumeric string with "-" included. Any other character is not allowed (including special characters such as "_", "*", "%", ...).

Example of new stack descriptor added in the candidate *setup_data.yaml* file without any VIM pod target:

```
cvim-mon-stacks:
- name: production-stack1
```

Any changes to the *candidate setup data* file besides adding new stacks to the *cvim-mon-stacks* list result in a validation failure. After updating the *candidate setup data* file, run the *add-stack* operation using the following command:

```
# k8s_runner --add-stack --setupfile <path_to_candidate_setup_data_file>
```

Deleting a Stack (--delete-stack)

You can execute the *delete-stack* operation after deleting the stack descriptors from the *candidate setup data* file. Any changes to the *candidate setup data* file besides deleting stacks from the existing *cvim-mon-stacks* list result in a validation failure.

After updating the *candidate setup data* file, execute the *k8s_runner* command with the *delete-stack* option: The following example shows how to use the reconfigure option:

```
# k8s_runner --delete-stack --setupfile <path_to_candidate_setup_data_file>
```

Adding, Updating, and Deleting VIM Pod Targets in a Stack

The Prometheus servers in each stack are configured to pull metrics from the list of configured target VIM pods at the configured interval. These pull requests are scheduled concurrently to spread equally within the interval window. This action enables a better distribution of the bandwidth within each scrape interval.

Each VIM pod target must be associated with a given region and metro. A region is identified by its name which can have any alphanumeric character with "_" and "-". Few examples of region names are *East, West, Asia, US1,* and *US-1*.

A region can contain one or more metros. Each metro is identified by a name which can have any alphanumeric character with "_" and "-". Examples of metro names: are *LosAngeles* and *Bay-Area-3*.

You can add VIM pods as targets once they are configured or reconfigured with VIM-MON enabled and central set to True. Ensure that you explicitly use a sampling interval of one minute (instead of the default 15 seconds) to match the default 1m sampling interval configured in the HA VIM-MON cluster. Following are the details of *setup_data.yaml* for a VIM pod:

Pod Setup data for monitoring by HA CVIM-MON cluster

```
CVIM_MON:
  central: true
  enabled: true
  polling_intervals:
     high_frequency: 1m
```

⚠ One important limitation is that an HA CVIM-MON cluster configured to use IPv6 (ARGUS_BAREMETAL: DHCP_MODE: v6) also requires the Cisco VIM pods to be dual stack (IPv4 and IPv6). Cisco VIM pods configured in IPv4 are not monitored by IPv6 HA CVIM-MON.

© 2026 HCLTech. All rights reserved.

The following VIM pod information are required to monitor the VIM pod from an HA VIM-MON cluster:

Parameter Name	Description	Pod Management Node Information Location
ip	VIM pod scraping IP address and port	The IP address is available in */root/openstack-configs/setup_data. yaml*: For dual stack pod: external_lb_vip_ipv6_address For ipv4 only pod: *external_lb_vip_address* Always use the port 9283.
cvim_mon_proxy_password	Denotes the password to access the scraping service point	Use this *CLI:* *hvim list-secrets* --getpassword *CVIM_MON_PROXY_PASSW ORD*
cert	Indicates the pod CA certificate file	*/root/openstack-configs/haproxy-ca.crt*

In case of an IPv6 HA VIM-MON cluster, always use the *external_lb_vip_ipv6_address* IP address. The configured IPv6 address must be given within square brackets followed by the port. For example:

```
ip: '[2001:420:293:2426:172:29:84:252]:9283'
```

In case of IPv4 HA VIM-MON cluster, use the *external_lb_vip_address* IP address and append the 9283 port, example:

```
ip: 10.1.1.1:9283
```

The CA cert file must be copied from the VIM pod to the */root/cvimha_certs/* directory on the HA VIM-MON cluster management node and with a unique name.

In case where all the VIM pods share the same CA cert file, that cert file needs to be copied only once. If multiple CA cert files are used, they must be copied with a different name in that directory.

Adding a Target VIM OpenStack Pod (--add-VIM-pod)

You can execute the *add-cvim-pod* operation after adding 1 or more new VIM OpenStack targets to the existing stacks of the cvim-mon-stacks list in the *candidate setup data* file. Target names must be unique and must have a unique target IP. Any changes to the *candidate setup data* file besides adding new targets to existing stacks result in a validation failure.

Example of IPv6 VIM pod target added to a stack descriptor (proper indenting with spaces is important), using the region *US* and metro *BayArea:*

```
cvim-mon-stacks:
- name: production-stack1
  regions:
  - name: US
    metros:
    - name: BayArea
      pods:
      - cert: /root/cvimha_certs/stormflyc-haproxy.crt
        cvim_mon_proxy_password: n6SrcYXxpsyCKUsM
        ip: '[2001:420:293:2426:172:29:84:252]:9283'
        name: stormflyc
        username: admin
```

After updating the *candidate setup data* file, execute the *add-VIM-pod* operation using the following command:

© 2026 HCLTech. All rights reserved.

k8s_runner --add-cvim-pod --setupfile <path_to_candidate_setup_data_file>

Reconfiguring a VIM OpenStack Pod (--reconfigure-VIM-pod)

You can use the *reconfigure-cvim-pod* option to update the HA proxy certificate (cert) and HA CVIM-MON proxy password (cvim_mon_proxy_password) of each pod.

You can execute the reconfigure-cvim-pod operation after changing the certificate or the proxy password keys of the existing OpenStack targets from the existing stacks in the candidate setup data file. Any changes to the candidate setup data file besides these keys in the existing cvim-mon-stacks list result in a validation failure.

After updating the candidate setup data file, execute the *reconfigure-cvim-pod* operation using the following command:

k8s_runner --reconfigure-cvim-pod --setupfile <path_to_candidate_setup_data_file>

Deleting a Target VIM OpenStack Pod (--delete-VIM-pod)

You can execute the *delete-cvim-pod* operation after deleting the existing OpenStack targets from the existing stacks in the *candidate setup data* file. Any changes to the *candidate setup data* file besides deleting targets from the existing cvim-mon-stacks list result in a validation failure.

After updating the *candidate setup data* file, execute the *delete-cvim-pod* operation using the following command:

k8s_runner --delete-cvim-pod --setupfile <path_to_candidate_setup_data_file>

HA VIM-MON Update

There is NO, HA CVIM-MON support in 5.0.3, since this is the first release version in cvim 5.0.x where HA CVIM-MON is supported.

Updating Software of HA VIM MON Nodes

- Updating to New Version
- Rolling Back to Previous Version

Committing Updated Version

You can update the software of HA VIM-MON nodes using the following three actions:

Action	Description
Update	Gets the new software version and updates the HA VIM-MON software in the nodes. If successful, the cluster is set to the Updated state. From that state, the update must be committed or rolled back.
Rollback	Rolls back to the previous version of the software if the updated cluster is not satisfactory.
Commit	Commits the software update. You cannot perform a rollback to an older version, after you commit the software.

Updating to New Version

When HA VIM-MON is in the INSTALLED state, it is possible to update to a new version using the update operation. The update operation performs the following steps:

- Extracts the new software version and the container images.
- Updates the software and containers in the management nodes.
- Updates the software in the HA VIM-MON controller and worker nodes.
- Updates the HA VIM-MON stacks.

© 2026 HCLTech. All rights reserved.

Following are the steps to update the software of HA VIM-MON nodes:

- Get the *mercury-installer.tar.gz"* file corresponding to the new version, based on the installation type.
- Initiate the update operation using the *k8s_runner* CLI and passing the file pathname with the *--update* option from any directory:

  k8s_runner --update <pathname>

The cluster is set to *Updated* state after successful completion. The *mercury-installer.tar.gz* file can be safely deleted, after the update is completed.

Rolling Back to Previous Version

You can invoke the rollback operation from the *Updated* state to roll back to the previous software version if there are problems after the software update.

To roll back the update, execute the following command from any location:

```
k8s_runner --rollback
```

Starting from 4.4.4, CVIM-MON HA rollback to previous versions like (4.4.X (X =1,2,3) cannot be possible.
In vim 4.4.4, the kubernetes version is 1.25 whereas the previous version is 1.21

Committing Updated Version

You can invoke the commit operation from the *Updated* state and commit the software update. You cannot rollback to an older version, after you commit the software.

The commit operation performs the following:
- The old version of the software running in the containers of the management node is removed.
- The software version of the HA VIM-MON stacks is committed with the running version.
- All intermediate files and temporary files are removed.

To commit the update, execute the following CLI command from any location:

```
k8s_runner --commit
```

If successful, a backup of the management node is performed automatically at the end of the commit operation (auto-backup).

Custom Grafana Dashboards

Custom Grafana Dashboards for HA VIM MON

- Overview
- Listing Custom Dashboards

Saving Custom Dashboards from Grafana to Management Node
Uploading Custom Dashboards from Management Node to Grafana Server

Most Common Use Cases and Best Practices for Managing Custom Dashboards
- Creating and Saving First Custom Dashboard
- Adding or Modifying Dashboards
- Uploading Custom Dashboards on Freshly Installed Cluster

© 2026 HCLTech. All rights reserved.

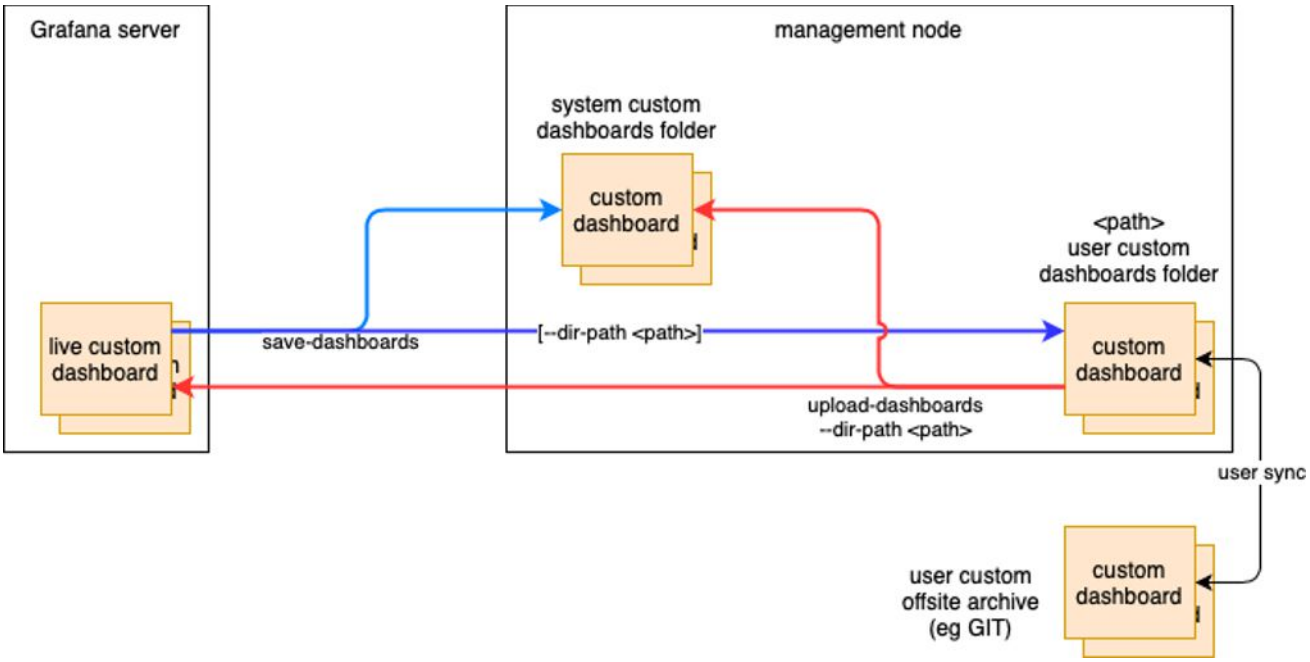
Overview

HA VIM-MON allows you to create, persist and upload custom dashboards to Grafana.

Custom dashboard can be created directly from any Grafana session when logged in as admin. Once these new dashboards are saved from the UI, they can be persisted on the cluster management node using the *k8s_runner* CLI *(--save-dashboards)*. These dashboards will be saved on the local disk in the system custom dashboard folder. This folder is backed up by any backup operation. In the current version, you must create custom dashboards at the top of the dashboards hierarchy as folders are not persisted.

Built-in dashboards may be modified but are never persisted by *k8s_runner*. The only way to make permanent changes to the content of a built-in dashboard is to duplicate that dashboard as custom dashboard and modify the copy (all from within the Grafana UI).

You can copy the custom dashboards that have been persisted on the management node to a user-specified directory, for archiving or versioning. For example, these dashboard files can be saved in a version control system. Conversely, you can upload the custom dashboards from a user-specified directory to the system custom dashboard folder and to Grafana using the *--upload-dashboard* option. This is useful, for example to install or update a set of custom, dashboards across multiple HA VIM-MON clusters starting from development cluster to production cluster.



Use the following *k8s_runner* option to manage custom dashboards:

```
# k8s_runner -h

...
--cvimmon-custom-dashboards
                Required key to execute operations on grafana custom dashboards(--list/save/upload-dashboards)
  --save-dashboards      Save custom dashboards from Grafana server to mgmt node or to folder specified by --dir-
path
  --list-dashboards      List custom dashboards
  --upload-dashboards    Upload custom dashboards to Grafana and persist to mgmt node from folder specified by --
dir-path
  --force                Delete dashboards in the destination if they are missing in the source.
  --preserve             (deprecated)
  --dir-path <custom dashboard folder pathname>
                         Upload (mandatory): upload from folder, Save (optional): save to folder
  --dry-run              To view the changes done on Grafana. No actual changes will be made.
```

© 2026 HCLTech. All rights reserved.

Use *--cvimmon-custom-dashboards* if you want to execute any operation related to custom dashboards. It must always be paired with one of the following options:

- --list-dashboards
- --save-dashboards
- --upload-dashboards

Listing Custom Dashboards

To list current custom dashboards, use the following command:

```
# k8s_runner --cvimmon-custom-dashboards --list-dashboards
```

This command lists all custom dashboards and their states. A custom dashboard can be in one of the following states:

State	Description
Uncha nged	Same copy of the custom dashboard is present in both the Grafana server and the system custom dashboard folder on the management node.
New	The custom dashboard is only present in the Grafana server but not yet saved to the system custom dashboard folder on the management node.
Modified	The custom dashboard is present in both the Grafana server and the system custom dashboard folder on the management node, but both copies are different. This means that the Grafana version was modified but not saved.
Deleted	The custom dashboard is present only in the system custom dashboard folder on the management node, but not on the Grafana server. This is likely to happen when the Grafana admin user deletes a custom dashboard.

An example of an output showing two custom dashboards, one being saved and unchanged *(Link State)* and another being created in Grafana but not saved (*Link State Revised*):

```
# k8s_runner --cvimmon-custom-dashboards --list-dashboards

2021-02-09 01:53:39,498 INFO Runner - Executing: CVIMMON_CUSTOM_DASHBOARDS (args: --cvimmon-custom-dashboards --
list-dashboard)
2021-02-09 01:53:40,532 INFO Runner - List custom dashboards:
+----------------------+--------------------+
| Dashboard Group      | Dashboard Title    |
+----------------------+--------------------+
| Unchanged dashboards | Link State         |
| New dashboards       | Link State Revised |
+----------------------+--------------------+
2021-02-09 01:53:40,552 INFO Runner - Operation CVIMMON_CUSTOM_DASHBOARDS completed successfully
The logs for this run are available at /var/log/cvimmonha/20210209_015339_cvimmon_custom_dashboards
#
```

Saving Custom Dashboards from Grafana to Management Node

The save dashboard operation synchronizes all custom dashboards from the Grafana server to the management node custom dashboard folder. To save custom dashboards from Grafana to the management node, use the following command:

```
k8s_runner --cvimmon-custom-dashboards --save-dashboards
```

If there is at least one dashboard present on the management node custom dashboard folder and not in the Grafana server (for example, if the dashboard is deleted from Grafana) during sync operation, the command fails unless *--force* option is passed. In other words, the *--force* option confirms that the deletion of dashboard(s) in Grafana can persists.

© 2026 HCLTech. All rights reserved.

Once the Grafana server and management node are synchronized, you can copy all custom dashboards to any user-provided and user-managed empty directory on the management node using the *--dir-path* option. This option is useful if you want to version and save all the custom dashboards in a version control system, for example, Git.

This operation succeeds only if the Grafana server and management node repository are in sync. Hence, a sync operation is required before copying the custom dashboards to a user directory.

Following is an example of the *--dir-path* option:

```
k8s_runner --cvimmon-custom-dashboards --save-dashboards --dir-path /root/sync_dash/
```

⚠ If you pass the *--dry-run* option to any of these options, you can see relevant logs, but no actual sync operation happens between the Grafana server and management node repository.

An example of save dashboard with dry run:

```
# k8s_runner --cvimmon-custom-dashboards --save-dashboards --dry-run

2021-02-09 02:00:55,213 INFO Runner - Executing: CVIMMON_CUSTOM_DASHBOARDS (args: --cvimmon-custom-dashboards --
save-dashboards --dry-run)
2021-02-09 02:00:55,278 INFO Runner - No Changes will be made as --dry-run option is selected
2021-02-09 02:00:55,446 INFO Runner - [DRY_RUN] Saving custom dashboards to the management node, force=False...
2021-02-09 02:00:55,446 INFO Runner - [DRY_RUN] Saving new dashboard: Link State Revised
2021-02-09 02:00:55,447 INFO Runner - [DRY_RUN] Save custom dashboards:
+---------------------+--------------------+
| Dashboard Group     | Dashboard Title    |
+---------------------+--------------------+
| Unchanged dashboards | Link State         |
| New dashboards      | Link State Revised |
+---------------------+--------------------+
2021-02-09 02:00:55,477 INFO Runner - Operation CVIMMON_CUSTOM_DASHBOARDS completed successfully
The logs for this run are available at /var/log/cvimmonha/20210209_020055_cvimmon_custom_dashboards
```

⚠ Before executing the *--save-dashboard* CLI, any modified Grafana dashboards must be explicitly saved in Grafana UI first.

The changes done in a custom dashboard in the Grafana UI are saved only to the Grafana database. Hence, it is not sufficient for the custom dashboards to persists on the management node.

k8s_runner will not allow any Grafana dashboard folder associated with custom dashboards to persist.

Uploading Custom Dashboards from Management Node to Grafana Server

There are two main use cases that require uploading of custom dashboards on Grafana server:

⚠ Installation of a set of custom dashboards stored in a user folder for a freshly deployed cluster.

Overwriting of any modification in the Grafana server with the set of dashboards that are saved in the management node custom dashboard folder. For example, recovering from accidental deletion of a custom dashboard in Grafana.

To upload custom dashboards from a user-managed directory to the management node custom dashboard folder and the Grafana server, you must provide the user-managed directory using the *--dir-path* option. This functionality works only when the Grafana server and the management node custom dashboard folder are in sync, for example, both are empty, or both contain the same set of custom dashboards. Use the *--save-dashboard* command to sync first, if needed.

The upload command fails if the Grafana server contains custom dashboards that are not present in the user-managed dashboard folder. If the upload fails, do one of the following to proceed further:

- either rerun with the --force option to upload the user managed custom dashboards and delete those extra custom dashboards in Grafana • Or manually delete these extra dashboards from Grafana and rerun the command

© 2026 HCLTech. All rights reserved.

- Or save these extra dashboards and save them in the user managed dashboard folder and rerun the command.

Following is an example of the *--force* option:

```
# k8s_runner --cvimmon-custom-dashboards --upload-dashboards --dir-path /root/sync_dash/ --force
```

The above options are useful if you want to upload a new set of custom dashboards from a git repository onto a newly deployed HA VIM-MON stack.

> ⚠ If you use the *--dry-run* option to run the operations without actual sync, you can see logs for operations between the Grafana server and the management node.

Most Common Use Cases and Best Practices for Managing Custom Dashboards

Creating and Saving First Custom Dashboard

- Create a version controlled environment for saving and managing versions of your custom dashboards, for example, create a git repository.
- Create a new custom dashboard from the Grafana UI with the admin user.
- Save the new dashboard in Grafana UI.
- List the custom dashboards and verify whether a new one is detected:

```
k8s_runner --cvimmon-custom-dashboards --list-dashboards
```

- Create an empty directory on the management node, for example, */root/cdash*. This directory must exist but must be empty.

Save the new custom dashboard to the management node using the below command to synchronize the Grafana server and management node:

```
k8s_runner --cvimmon-custom-dashboards --save-dashboards
```

Save a copy of the dashboard in the */root/cdash* folder:

```
k8s_runner --cvimmon-custom-dashboards --save-dashboards --dir-path /root/cdash
```

Save and commit the content of */root/cdash* in the git repository.

Adding or Modifying Dashboards

- Add dashboards or modify dashboards in Grafana (admin user).
- Synchronize the Grafana server and management node by saving the new custom dashboard to the management node:

```
k8s_runner --cvimmon-custom-dashboards --save-dashboards
```

- Empty the user dashboard folder or create a new empty folder.
- Save a copy of the dashboard in the */root/cdash* folder:

```
k8s_runner --cvimmon-custom-dashboards --save-dashboards --dir-path /root/cdash
```

© 2026 HCLTech. All rights reserved.

Finally save and commit the content of */root/cdash* in the git repository.

Uploading Custom Dashboards on Freshly Installed Cluster

- Copy the custom dashboards from the git repository to the management node.
- Upload the custom dashboards:

```
k8s_runner --cvimmon-custom-dashboards --upload-dashboards --dir-path /root/cdash
```

Alert Rules

Customizing Alerting Rules

- Overview
- Update Alerting Rules
- Format of Custom Alerting Rules File
- Adding Alert Rules
- Modifying Alert Rules
- Deleting Alert Rules
- Validation Script for Custom Alerting Rules

Overview

Alerting rules define how alerts must be triggered based on conditional expressions on any available metric.

You can view the alerts from the Grafana user interface or Alerting dashboard or send them optionally to a number of supported receivers. After deployment, the pod administrators can customize the alerting rules based on their requirements.

For example, you can trigger an alert when any performance metric such as CPU usage, network throughput, or disk usage reaches a certain threshold. HA VIM-MON has a set of default built-in alerting rules that cover the most important error conditions that can occur in the pod.

You can customize alerting rules by using the following steps:

- Create a custom alerting rules configuration file to add new rules, modify, or delete built-in rules.
- Verify that the custom alerting rules file is valid using a verification tool.
- Update alerting rules by applying the custom alerting rules file.

The custom alerting rules are applied to a Thanos Ruler component, which is common to all stacks. It is not possible to apply different custom alerting rules for each stack.

Update Alerting Rules

The update operation for alerting rules always merges the following two files:

- Default alerting rules file (built-in file)
- Custom alerting rules file

Applying a second custom alerting rules file does not preserve alerting rules from the previously applied custom alerting rules file. The update operation does not include previously applied custom alerting rules.

To update alerting rules, run the *k8s_runner* command with the *--alerting_rules_config* option and a path to the *custom_alerting_rules.yml* file.

For example:

```
# k8s_runner --alerting_rules_config /root/custom_alerting_rules.yaml
```

The merge tool output file consists of:

- All rules from the custom_alerting_rules.yml file that do not belong to the change-rules or delete-rules group.

© 2026 HCLTech. All rights reserved.

- Rules from the default_alerting_rules.yml that:
  - Do not duplicate rules from custom file.
  - Must not be deleted.
  - Are modified according to change-rules input.

Format of Custom Alerting Rules File

The format of *custom_alerting_rules.yml* is identical to the one used by the Prometheus configuration file with a few additional semantic extensions to support the deletion and modification of pre-built existing rules.

The groups entry contains a list of groups identified by **group_name**, where each group can include one or more rules. The labels are used for determining the severity and other SNMP trap attributes.

The limitations when setting labels are given below:

- You must set the values of severity, snmp_fault_code, and snmp_fault_severity to the values specified in the example below.
- You must set the value of snmp_fault_source to indicate the metric used in the alert expression.
- You must not change the value of snmp_node.
- You must set the value of snmp_podid as the pod name specified in the setup data file.

```
groups:
- name: {group_name}
  rules:
  - alert: {alert_name}
    annotations:
      description: {alert_description}
      summary: {alert_summary}
    expr: {alert_expression}
    for: {pending_time}
    labels:
      severity: {informational/warning/critical}
      snmp_fault_code: {other/resourceUsage/resourceThreshold/serviceFailure/hardwareFailure
/networkConnectivity}
      snmp_fault_severity: {emergency/critical/major/alert/informational}
      snmp_fault_source: {fault_source}
      snmp_node: '{{ $labels.instance }}'
      snmp_podid: {pod_id}
```

Adding Alert Rules

Any alert rule specified under a group that is not named *change-rules* or *delete-rules* is populated to the merged output file. Custom rules are prioritized over the preexisting rules. If there are two alerts with the same name in both files, only the one from the custom file is retained as a result of the merge.

Modifying Alert Rules

You can modify any preexisting rule using the following syntax:

```
groups:
- name: change-rules
  rules:
  - alert: {alert_name}
    expr: {new_alert_expression}
    annotations:
      summary: {new_alert_summary}
```

The merge script identifies only the group named *change-rules* and changes the following fields of the targeted alert and description expression for:

- When the field is not present in change-rules: reuse value from built-in alerting
- rules
- When the field is present in change-rules and contains a non-empty value:
- overwrite the value from built-in alerting rules
- When the field is present in

© 2026 HCLTech. All rights reserved.

- change-rules, but does nothave value:
- remove the property from built-in alerting rules
- Severity
- snmp_fault_code
- snmp severity
- summary

If the alert to be changed does not exist, it is not created, and changes are not made.

Deleting Alert Rules

You can delete any built-in rule by using the following construct in the *custom_alerting_rules.yml* file:

```
groups:
- name: delete-rules
  rules:
  - alert: {alert_name/regular_expression}
```

The merge script identifies only the group named *delete-rules* and deletes pre-existing rules that match the provided names or regular expressions. If the alert to be deleted does not exist, changes are not made.

The following custom configuration file includes examples of a new alerting rule, a modified alerting rule and a deleted alerting rule:

```
groups:
- name: cpu
  rules:
  - alert: cpu_idle
    annotations:
      description: CPU idle usage is too high - resources underutilized
      summary: CPU idle too high
    expr: cpu_usage_idle > 80
    for: 5m
    labels:
      severity: informational
      snmp_fault_code: resourceUsage
      snmp_fault_severity: informational
      snmp_fault_source: cpu_usage_idle
      snmp_node: '{{ $labels.instance }}'
      snmp_podid: pod7
  - alert: cpu_iowait
    annotations:
      description: CPU iowait usage is too high
      summary: CPU iowait too high
    expr: cpu_usage_iowait > 10
    for: 3m
    labels:
      severity: warning
      snmp_fault_code: resourceUsage
      snmp_fault_severity: alert
      snmp_fault_source: cpu_usage_iowait
      snmp_node: '{{ $labels.instance }}'
      snmp_podid: pod7
- name: change-rules
  rules:
  - alert: disk_used_percent
    expr: disk_used_percent > 99
    annotations:
      summary: Disk used > 99%
  - alert: reboot
    annotations:
      summary: Server rebooted
  - alert: system_n_users
    expr: system_n_users > 10
- name: delete-rules
  rules:
  - alert: disk_filling_up_in_4h
  - alert: mem.*
```

© 2026 HCLTech. All rights reserved.

Validation Script for Custom Alerting Rules

You must validate any custom alerting rules file before an update, using the following CLI command:

```
# check_promtool -v <custom_alerts_file>
```

The validation script uses the Prometheus *promtool* script but skips some of its checks for updating and deleting rules. It also checks if the SNMP severities and fault codes are supported.

The following example shows the output of the *promtool* script in case of a successful validation:

```
# check_promtool /root/alerting_custom_rules.yaml -v
check_promtool: checking /root/alerting_custom_rules.yaml
check_promtool: success:
check_promtool:  rules to be changed: 1
check_promtool:  regular expressions for rules to be deleted: 2
check_promtool:  rules to be added: 1
```

The following example shows the output of the *promtool* script in case of a failure:

```
# check_promtool /root/alerting_custom_rules.yaml -v
check_promtool: checking /root/alerting_custom_rules.yaml
check_promtool: failure:
check_promtool:    line 10: mapping key "for" already defined at line 9
Error! Custom rules file is invalid
```

Alert Manager

Customizing Alert Manager and Receivers

- Overview
- Supported Receivers
- Alert Manager Custom Configuration File Format
- Default Built-in Configuration File
- SNMP Trap Receivers
- Validation Script

Overview

The Alert Manager component in VIM-MON is in charge of the routing, grouping, and inhibiting alerts that are sent by the Prometheus alert rule engine to the appropriate receivers.

By default, VIM-MON forwards each alert to the SNMP agent to be sent to the SNMP managers as SNMP traps, if enabled in the configuration file. After deployment, you can add custom alert routes, alert grouping, alert inhibitions, and receivers by following the below steps:
Create a proper custom alert manager configuration file:

- Create a custom alert manager configuration file.
- Edit the content using your favourite editor (see format below).
- Verify that the custom alert manager configuration file is valid using the provided tool.

Once the file is validated, you can execute the following command:

```
# k8s_runner --alertmanager_config <alertmanager_config_file>
```

The custom alert manager configuration is applied to a unified alert manager component, which is common for all stacks. It is not possible to apply different custom alert manager configurations for each stack.

© 2026 HCLTech. All rights reserved.

Supported Receivers

The Alert Manager supports the following list of receivers:

- webhook
- pagerduty
- e-mail
- pushover wechat
- opsgenie victorops

Alert Manager Custom Configuration File Format

General Format

The following listing shows the general format of the alert manager configuration file. Most custom configuration files must include only a small subset of the available options.

global:

```
# ResolveTimeout is the time after which an alert is declared resolved # if it has not been updated.
[ resolve_timeout: <duration> | default = 5m ]

# The default SMTP From header field. [ smtp_from: <tmpl_string> ]
# The default SMTP smarthost used for sending emails, including port number.
# Port number usually is 25, or 587 for SMTP over TLS (sometimes referred to as STARTTLS).

# Example: smtp.example.org:587 [ smtp_smarthost: <string> ]
# The default hostname to identify to the SMTP server. [ smtp_hello: <string> | default = "localhost" ]
[ smtp_auth_username: <string> ]
# SMTP Auth using LOGIN and PLAIN. [ smtp_auth_password: <secret> ]
# SMTP Auth using PLAIN.
[ smtp_auth_identity: <string> ] # SMTP Auth using CRAM-MD5.
[ smtp_auth_secret: <secret> ]
# The default SMTP TLS requirement.
[ smtp_require_tls: <bool> | default = true ]

# The API URL to use for Slack notifications. [ slack_api_url: <secret> ]
[ victorops_api_key: <secret> ]
[ victorops_api_url: <string> | default = "https://alert.victorops.com/integrations/generic/20131114/alert/" ]
[ pagerduty_url: <string> | default = "https://events.pagerduty.com/v2/enqueue" ] [ opsgenie_api_key: <secret> ]
[ opsgenie_api_url: <string> | default = "https://api.opsgenie.com/" ] [ hipchat_api_url: <string> | default =
"https://api.hipchat.com/" ]  [ hipchat_auth_token: <secret> ]
[ wechat_api_url: <string> | default = "https://qyapi.weixin.qq.com/cgi-bin/" ] [ wechat_api_secret: <secret> ]
[ wechat_api_corp_id: <string> ]

# The default HTTP client configuration [ http_config: <http_config> ]
# Files from which custom notification template definitions are read.
# The last component may use a wildcard matcher, e.g. 'templates/*.tmpl'. templates:
[ - <filepath> ... ]

# The root node of the routing tree. route: <route>

# A list of notification receivers. receivers:
- <receiver> ...

# A list of inhibition rules. inhibit_rules:
[ - <inhibit_rule> ... ]
```

The custom configuration must be a full working configuration file with the following template. It must contain three main keys such as global, route, and receiver.

The global configuration must have at least one attribute, for example, resolve_timeout = 5m. Ensure that all new receivers must be part of the route, so the alerts are routed to the proper receivers. The receiver name cannot be snmp.

You can find the configuration details for creating route/receiver in the Prometheus Alert Manager documentation (publicly available online).

© 2026 HCLTech. All rights reserved.

```
global: resolve_timeout: 5m

route: <route>

receivers:
- <receiver> ...

The following is a custom config to add a webhook receiver.

global:
  resolve_timeout: 5m

route:
  group_by: ['alertname', 'cluster', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 8737h
  receiver: receiver-webhook

receivers:
- name: 'receiver-webhook'
  webhook_configs:
  - send_resolved: true
    url: 'http://webhook-example:####/xxxx/xxx'
```

Default Built-in Configuration File

Two different default configuration files are available to define the following in order:

- Generic route for all alerts to the SNMP agent running on the management node.
- Route to a generic receiver that can be customized to add a channel of notification (webhook, slack and others).

Default configuration file with SNMP enabled

```
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname', 'cluster', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 8737h

  # A default receiver
  receiver: snmp

receivers:
- name: 'snmp'
  webhook_configs:
  - send_resolved: true
    url: 'http://localhost:1161/alarms'
```

Default configuration file with SNMP disabled

```
route:
  receiver: recv
  group_by:
  - alertname
  - cluster
  - service
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 8737h
receivers:
- name: recv
```

SNMP Trap Receivers

You can send the SNMP traps to SNMP managers enabled in the VIM configuration file setup_data.yaml.

© 2026 HCLTech. All rights reserved.

Example: inhibit (mute) alerts matching a set of labels

Inhibit alerts is a tool that prevents certain alerts to be triggered if other alert/alerts is/are trigged.
If one alert having the target attribute matches with another alert that has source attribute, this tool inhibits the alert with target attribute.

This is the general format for inhibit alerts. You can set a regex to match both the source and target alerts and to filter the alerts per label name.

```
# Matchers that have to be fulfilled in the alerts to be muted.
target_match:
  [ <labelname>: <labelvalue>, ... ]
target_match_re:
  [ <labelname>: <regex>, ... ]

# Matchers for which one or more alerts have to exist for the
# inhibition to take effect.
source_match:
  [ <labelname>: <labelvalue>, ... ]
source_match_re:
  [ <labelname>: <regex>, ... ]

# Labels that must have an equal value in the source and target
# alert for the inhibition to take effect.
[ equal: '[' <labelname>, ... ']' ]
```

Example: Inhibit alerts if other alerts are active

The following is an example of inhibit rule that inhibits all the warning alerts that are already critical.

```
inhibit_rules:
- source_match:
    severity: 'critical'
  target_match:
    severity: 'warning'
  # Apply inhibition if the alertname is the same.
  equal: ['alertname', 'cluster', 'service']
```

This is an example of inhibit all alerts docker_container in containers that are down (which has the alert docker_container_down on).

```
inhibit_rules:
  - target_match_re:
      alertname: 'docker_container.+'
    source_match:
      alertname: 'docker_container_down'
    equal: ['job', 'instance']
```

Validation Script

When a new configuration is set, execute *amltool* script and ensure that you get a SUCCESS in the output from the configuration POV.

```
# amtool check-config <alertmanager_config_file>
Checking '<alertmanager_config_file>'  SUCCESS
Found:
 - global config
 - route
 - 0 inhibit rules
 - 1 receivers
 - 0 templates
```

© 2026 HCLTech. All rights reserved.

Backup

Backing Up HA VIM-MON

- Overview
- Auto Backup

Moving Backup files of Management Node

Overview

Backup of the management node is essential to recover the management node from catastrophic failure. You can take the backup of HA VIM-MON management node either manually or automatically through cluster management operations that change the state of the cluster, for example, commit following an update.

In any case, the backup operation creates a backup compressed file locally. You need to copy this backup compressed file from the management node to a safe location.

Auto-backup initiates only if the last executed operation changes the cluster state and completes the state change successfully.

Manual Backup

You can perform manual backup only in the INSTALLED state using *k8s_runner* CLI with the *--backup* option:

```
# k8s_runner --backup

2020-11-02 16:30:05,145 INFO Runner - Executing: BACKUP (args: --backup)
2020-11-02 16:30:05,220 INFO Runner - Executing backup to /var/cisco/cvimmonha_backup/cvimmonha_backup_3.9.2
_2020-11-02_16:30:05, disconnected install=False
2020-11-02 16:30:05,220 INFO Runner - Validating the current workspace...
2020-11-02 16:30:05,286 INFO Runner - Creating the backup...
2020-11-02 16:30:05,342 INFO Runner - Compressing the backup...
2020-11-02 16:30:05,513 INFO Runner - Backup completed successfully: /var/cisco/cvimmonha_backup
/cvimmonha_backup_3.9.2_2020-11-02_16:30:05.tgz
2020-11-02 16:30:05,514 INFO Runner - Operation BACKUP completed successfully
The logs for this run are available at /var/log/cvimmonha/backup_2020-11-02_16:30:05
```

Information about the backup compressed file and log directory location appears in the console, after the execution of the operation. All manual backup compressed files are saved under the following directory:

```
# ls -l /var/cisco/cvimmonha_backup
total 54608
-rw-r--r--. 1 root root 53988893 Oct 19 23:21 cvimmonha_backup_3.9.0_2020-10-19_23:21:24.tgz
-rw-r--r--. 1 root root  1928298 Nov  2 16:30 cvimmonha_backup_3.9.2_2020-11-02_16:30:05.tgz
#
```

© 2026 HCLTech. All rights reserved.

The *k8s_runner* CLI keeps only the last two manual backup compressed files. Hence, all older manual backup compressed files are automatically deleted.

Auto Backup

After successful completion of any state change operation, *k8s_runner* executes an auto-backup to create a backup compressed file.

The operations that do not change the installation state will not execute auto-backup. Examples of such operations:

- --get-endpoint
- --renew-k8s-certs
- --renew-etcd-certs
- --update

Examples of operations that will trigger an auto-backup:

- • --commit
- • --add-stack
- • --add-worker

All auto-backup compressed files are created under the following directory:

```
# ls -l /var/cisco/cvimmonha_autobackup/
total 5636
-rw-r--r--. 1 root root 3813199 Oct 30 12:26 cvimmonha_autobackup_3.9.1_2020-10-30_12:26:31.tgz
-rw-r--r--. 1 root root 1953866 Nov  2 13:58 cvimmonha_autobackup_3.9.2_2020-11-02_13:58:00.tgz
```

The *k8s_runner* CLI keeps only the last two auto backup compressed files. Hence, all older auto backup compressed files are automatically deleted.

Moving Backup files of Management Node

After each successful backup, you must move the created backup compressed file to a well-defined remote location, so that it can be used later when needed while restoring a management node.

Restore

Restoring HA VIM-MON

- Overview
- Prerequisites

Restoring Management node from Backup

Overview

You can restore a previous backup of the management node, typically after replacing the management node server and reinstalling it with the management node ISO.

The restore operation restores the state of the management node back to the backed-up state.

Prerequisites

Before you restore HA VIM-MON, ensure that the following conditions are met:

- The replacement management node must be installed from ISO image, kick-started with the right node type (HA-VIM-MON), and set in HA VIM- MON UNINSTALLED state.
- You must copy from your backup file storage the appropriate backup compressed file that was created from the same HA VIM-MON cluster.

© 2026 HCLTech. All rights reserved.

- Timezone, hostname, and IP configuration of the management node must be identical to the backup. If not, the restoration fails.
- You must not modify the backup compressed file. If modified, the restoration fails.

Restoring Management node from Backup

- After ISO installation, copy the backup compressed file from the remote backup server to the management node, preferably under */var/cisco*

/cvimmon_backup directory.

- Execute the *k8s_runner* restore operation from any location:

```
# k8s_runner --restore <path to the backup compressed file>
```

- When *k8s_runner* restores all the necessary configuration data on the management node, the script executes the following three steps to complete the restore operation:

```
!!  CVIM MON HA ORCHESTRATOR  !!
================================================
+-------------------------+--------------------+
| Operations              | Operation ID       |
+-------------------------+--------------------+
| VALIDATION              |       1            |
| BOOTSTRAP_INFRA         |       2            |
| SETUP_ARGUS             |       3            |
```

- After the successful completion of restoration, you can execute the *kubectl get pods --all-namespaces* command and see if all the pods are still in the running state. You can also verify by logging into Grafana using the old password and confirm if all the data is visible.

Reinstallation of VIM-MON HA Pod

Reinstallation of VIM-MON HA Pod

Due to unforeseen circumstances, there might be a need to reinstall the VIM MON HA pod with the same image version. To alleviate the need for reimaging the management node followed by re-installation of the VIM MON HA pod, reinstall the pod under the assumption that the management node is compatible with the same tag.

> ⚠ For reinstallation, ensure that you use the same target vim-MON servers in which the original installation is done. If you are using a different or subset of servers for reinstallation, power off the servers in which the previous installation is done, to prevent floating of the duplicate IPs in the network.

To reinstall the pod without reimaging the management node for VIM MON HA, follow the below steps:

1. Copy the *setup_data.yaml* from */root/openstack-configs/* directory to *~/Save/*

```
cd /root/installer
./cvim_mon_ha_unbootstrap.sh -k
```

2. To verify that no docker containers are running, use the command:

```
docker ps -a
```

3. To verify that no docker images are present, use the command:

© 2026 HCLTech. All rights reserved.

```
docker images
```

4. To re-run the VIM MON HA installation, use the command:

```
k8s_runner --install --setupfile ~/Save/setup_data.yaml
```

VIM MON HA Cluster Monitoring

VIM MON HA Cluster Monitoring

- Overview
- Components of HA VIM-MON Cluster Monitoring
- Cluster Monitoring Endpoint URL's
- Secrets for HA VIM-MON Cluster Monitoring
- Grafana Dashboards
- Alerts

Overview

HA VIM-MON has an inbuilt mechanism to monitor the cluster itself apart from monitoring the VIM pods. The HA VIM-MON cluster is made up of kubernetes master and kubernetes workers. The objective of the cluster monitor is to monitor the health of the entire kubernetes cluster. The cluster monitor provides the following information.

- Health of the cluster.
- Resource utilization of the nodes.
- Status of various Kubernetes objects.
- Portworx (Storage) monitoring
- Alerts for various conditions.

Components of HA VIM-MON Cluster Monitoring

The HA VIM-MON cluster monitor uses the following components to monitor the kubernetes cluster:

- Kube-state-metrics.
- Advisor
- Node-exporter
- Prometheus to collect metrics .
- Grafana to display the metrics.

All the tools for monitoring the cluster run under the *cvimmon-monitor* namespace.

Cluster Monitoring Endpoint URL's

To get information about cluster monitoring endpoint service URLs, use the following command:

```
# kubectl get ingress -n cvimmon-monitor
NAME HOSTS ADDRESS PORTS AGE
grafana-cvimmon-monitor ha-cluster-1-cvimmon-grafana.cisco.com 80, 443 37m
prometheus-cvimmon-monitor-alertmanager ha-cluster-1-cvimmon-alertmanager.cisco.com 80, 443 38m
prometheus-cvimmon-monitor-server ha-cluster-1-cvimmon-prometheus.cisco.com 80, 443 38m
```

© 2026 HCLTech. All rights reserved.

The prefix *ha-cluster-1* is the name of the cluster given in the *setup_data.yaml* under the section ARGUS_BAREMETAL.

Secrets for HA VIM-MON Cluster Monitoring

To get the secrets for the HA VIM-MON cluster monitor, use the following commands:

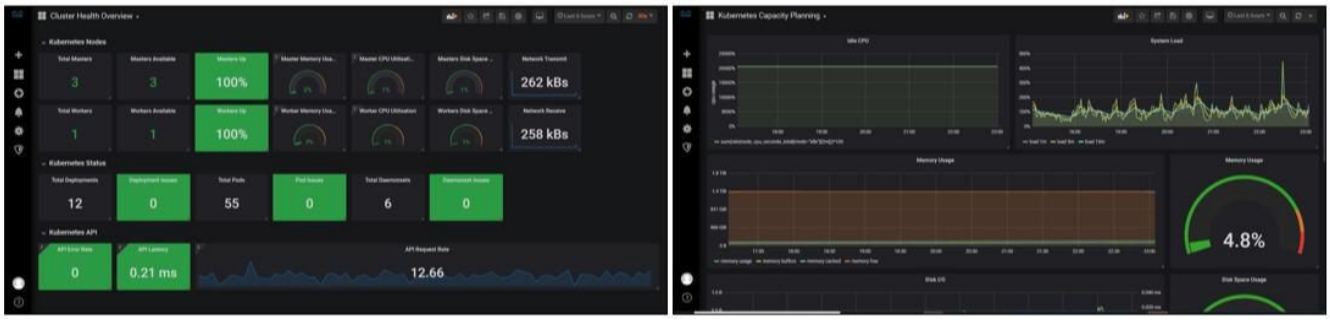
```
[root@Argentina ~]# k8s_runner --get-password "Grafana-Password-monitoring(Username:admin)"

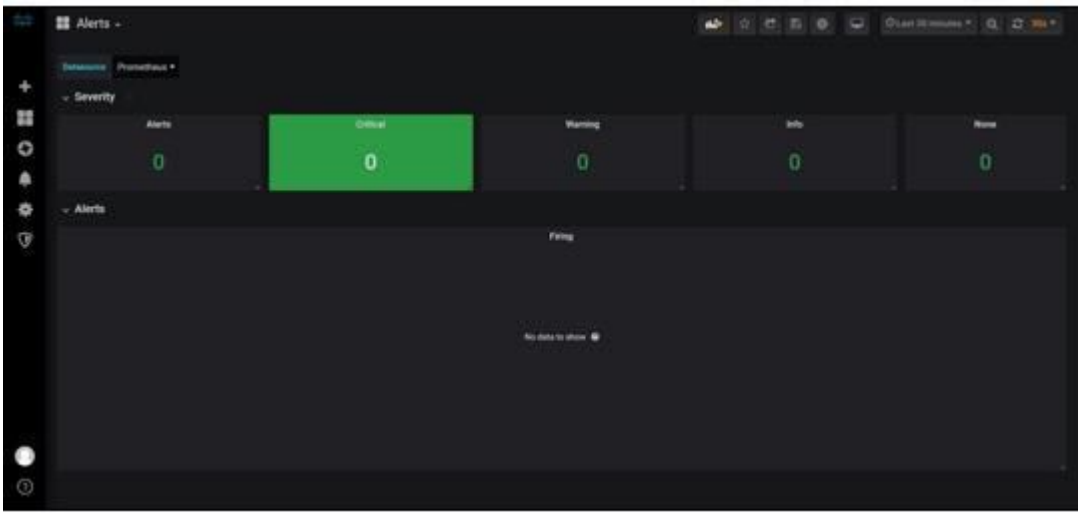
2025-12-09 10:30:24,295 INFO Runner - Executing: GET_PASSWORD (args: --get-password Grafana-Password-monitoring(Username:admin))

+-------------------------------------------+-----------------------------------------------+
|                 Secret Key                |                   Password                    |
+-------------------------------------------+-----------------------------------------------+
| Grafana-Password-monitoring(Username:admin) | cj2j8lTgSb76trRtoMbSoWXWACSTK0hKSzcWhCnaPac= |
+-------------------------------------------+-----------------------------------------------+
2025-12-09 10:30:24,332 INFO Runner - Operation GET_PASSWORD completed successfully
[root@Argentina ~]#
```

Grafana Dashboards

The grafana dashboards can be accessed with the URL from the above step. Some of the Dashboards in cluster monitoring are shown below:





© 2026 HCLTech. All rights reserved.

Alerts

The alerts are generated based on the predefined alerting rules. The predefined alerting rules are triggered based on the metrics obtained by the Prometheus server. The alerts can be monitored using the web interface or api. Some of the common alerts are mentioned in the table below:

Alert	Severity	Description
KubeAPIDown	critical	KubeAPI Server is down in some of the nodes.
KubeStateMetricsDown	critical	KubeStateMetrics is down.
KubeletDown	critical	Kubelet is down in one of the nodes.
NodeExporterDown	critical	NodeExporter is down.
KubePodCrashLooping	critical	Some of the pods are crash looping.
KubePodNotReady	critical	Some pods are not in the ready state even after 15 minutes.
KubeDeploymentReplicasMismatch / KubeStatefulSetReplicasMismatch	critical	The number of replicas do not match the number of replicas requested in kubernetes deployment / statefulset
KubePersistentVolumeUsageCritical	critical	The persistent volume claimed by a pod is nearing 85% of the allocated space.
KubePersistentVolumeFullInTwelveDays	critical	The persistent volume is expected to be full in twelve days.
KubeCronJobRunning / KubeJobCompletion	warning	Kubernetes CronJob / Job is running for a long time or more than 1 hour.
KubeCPUOvercommit,KubeMemOvercommit	warning	Overcommited CPU/memory resource requests on pods more than the CPU available in the nodes.
KubeNodeNotReady	warning	The node is in not ready state for more than an hour.
KubeClientErrors	warning	Kubernetes API client requests are experiencing errors.
KubeletTooManyPods	warning	The number of pods running in a node is more than the defined limit (110).
KubeClientCertificateExpiration	warning	Kubernetes API certificate is expiring in less than 15 days.
NodeDiskRunningFull	warning	The node disk will fill up within the next 24 hours.

HA VIM Mon Upgrade

Note: HA CVIM-MON upgrade from previous versions are supported in 5.0.4
3.4.6 >> 5.0.4
4.2.x (x=1,2,3,4) >> 5.0.4
4.4.y (y=4,5) >> 5.0.4

HA VIM Monitor (HA VIM-MON) supports upgrade to new releases. HA VIM MON 3.4.6 release is the starting point for any HA VIM MON upgrade to 4.2.x lineup. The HA VIM MON upgrade procedure is orchestrated with minimal downtime where the kubernetes cluster is redeployed with the updated components in the latest release. For the metrics retention, the historical prometheus data is restored from the backup and injected into the newly deployed HA VIM MON system. The upgraded to HA VIM-MON system in 4.x provides enhanced cluster self-monitoring, unified grafana interface, unified quering/alerting across stacks and enhanced *k8s_runner cli*.

Starting from VIM 4.4.3; VIM-MON HA upgrade is supported from 3.4.6 to 4.4.y(y=3,4) and 4.2.x (x=1,2,3,4) to 4.4.y (y=3,4)

- Comparison of HA VIM-MON Component Versions
- Upgrade Procedure
  - Pre-upgrade check
- Upgrade steps : HA VIM MON from 3.4.x (3.4.6)

© 2026 HCLTech. All rights reserved.

- Upgrade steps : HA VIM MON from 4.2.x (x=1,2,3,4) to 4.4.y (y= 3,4)

Comparison of HA VIM-MON Component Versions

Component	4.2.x	4.4.3	4.4.4	5.0.4	6.0	6.0.1
Kubernetes Version	v1.17.8	v1.21.14	v1.25.14	v1.29.5	v1.29.5	v1.34.3
Helm Version	v3.5.3	v3.7.1	v3.9.0	v3.9.0	v3.9.0	v3.9.0
Prometheus	2.24.1	v2.31.1	v2.31.1	V2.31.1	V2.31.1	V2.31.1
Alert manager	v0.21.0	v0.23.0	v0.23.0	v0.23.0	v0.23.0	v0.23.0
Grafana	7.2.0	v7.5.0	v7.5.0	v7.5.11	v7.5.11	v7.5.11
Storage	Minio	Minio	Minio	Minio	Minio	Minio
Thanos	v0.17.2	v0.17.2	v0.17.2	v0.17.2	v0.17.2	v0.17.2
Stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks
Ldap, SNMP, SYSLOG	Common	Common	Common	Common	Common	Common

Upgrade Procedure

Pre-upgrade check

- Plan for proper maintenance window and upgrade downtime. The upgrade downtime is proportional to the HA VIM MON installation time (time may varies depending on HA cluster size).
- Copy the autobackup directory from the management node to a safe location.
- The management node should have sufficient disk space for prometheus metrics data storage.
- The setup data for installation must be manually updated to be compatible with target release version .
- Refer to the new setup data format setup_data.yaml.CVIMMONHA.EXAMPLE under /installer-xx/openstack directory.
- The custom alerting rules need to be applied back to the system through reconfigure action. Ensure the backup of custom_alerting_rules.yml is completed.
- 6. The custom Grafana dashboards need to be restored to the new system manually.

Upgrade steps : HA VIM MON from 3.4.x (3.4.6)
Perform the following steps to upgrade the HA VIM MON pod from 3.4.x (x=6) to the current release of HA VIM MON POD:
1. Copy the latest autobackup from the management node.

```
## Auto backup location
# cd /var/cisco/cvimmonha_autobackup
```

© 2026 HCLTech. All rights reserved.

Move the auto backup to another remote server.

```
# rsync -e ssh -go -rtypX --numeric-ids
/var/cisco/cvimmonha_autobackup/<cvimmonha_autobackup_3.4.6_2021- 03-04_00:21:34> <Destination-Server>:/<path>
```

- Re-image the HA management node with the ISO of current release version with the same IP address.
- Copy the backup file to the management node from the Remote server.

```
# rsync -e ssh -go -rtypX --numeric-ids root@<Destination-server>:/path/ /root/
```

⚠ Ensure sufficient disk space available in the management node for performing backup/restore operation

Run the *prepare_vimmon_upgrade* tool with the backup. The tool performs the snapshot of the prometheus TSDB from the backup files.
Note : from 5.0.3 onwards prepare_cvimmon , commit_cvimmon_upgrade changed as prepare_vimmon ,commit_vimmon_upgrade as below followed steps

```
[root@Argentina ~]#  prepare_vimmon_upgrade --help
================================================================
                VIMMON PREPARE UPGRADE TOOL
================================================================

usage: prepare_vimmon_upgrade [-h] --storage_location STORAGE_LOCATION --backup_location BACKUP_LOCATION
--startimagetag
                              STARTIMAGETAG --endimagetag ENDIMAGETAG [--download_data]

optional arguments:
  -h, --help            show this help message and exit
  --storage_location STORAGE_LOCATION, -s STORAGE_LOCATION
                        Storage location to store the snapshot data
  --backup_location BACKUP_LOCATION, -b BACKUP_LOCATION
                        Location of 3.4.x backup
  --startimagetag STARTIMAGETAG, -st STARTIMAGETAG
                        Choose the starting Image tag;
  --endimagetag ENDIMAGETAG, -n ENDIMAGETAG
                        Choose the end image tag
  --download_data, -d   minio backup data restore
[root@Argentina ~]#

#prepare_vimmon_upgrade --storage_location /root/Databackup/ --backup_location /var/cisco
/cvimmonha_autobackup_3.4.6_2023-06-27_08\:12\:12/ -st 3.4.6 -n 4.9.9


================================================================
                VIMMON PREPARE UPGRADE TOOL
================================================================

2021-03-04 15:47:33,930 INFO Runner - Preparing for Cvimmon Upgrade from 3.4.x to 4.9.9
2021-03-04 15:47:33,930 INFO Runner - Checking if the backup is good
2021-03-04 15:47:33,931 INFO Runner - Copying all required files
2021-03-04 15:47:33,946 INFO Runner - Enabling Admin API in all stacks
2021-03-04 15:51:05,748 INFO Runner - Creating Prometheus Snapshots
2021-03-04 15:51:14,929 INFO Runner - Copying snapshot to storage

CVIM MON snapshot tool completed successfully!!

The logs for this run are available at /var/log/cvimmonha/20210304_154733_prepare_cvimmon_upgrade
/prepare_cvimmon_upgrade.log
```

Run the HA VIM MON installation operation using the *k8s_runner* CLI.

© 2026 HCLTech. All rights reserved.

> ⚠ Refer to the new setup data format *setup_data.yaml.CVIMMONHA.EXAMPLE* under */installer-xx/openstack* directory. GLOBAL CVIMMON MONITOR STACK PARAMETERS - section outlines the changes to be done for LDAP, SNMP, Log rotate configurations in the setup data. The parameters are moved from stack level to global POD level.

Once the install is complete, verify that HA VIM MON cluster is up and running.

```
#  kubectl get nodes
#  kubectl get pods -A
```

Run the *commit_vimmon_upgrade* tool with the snapshot path to restore the data.

```
[root@Argentina ~]# commit_vimmon_upgrade --help
====================================================================
                VIMMON COMMIT UPGRADE TOOL
====================================================================

usage: commit_vimmon_upgrade [-h] [--snapshot_location SNAPSHOT_LOCATION] [--restore_data RESTORE_DATA]

optional arguments:
  -h, --help            show this help message and exit
  --snapshot_location SNAPSHOT_LOCATION, -s SNAPSHOT_LOCATION
                        Storage location of the sanpshot prepared by prepare Tool
  --restore_data RESTORE_DATA, -r RESTORE_DATA
                        minio backup data restore path
[root@Argentina ~]#

# commit_vimmon_upgrade --snapshot_location /root/Databackup/snapshot_20241113_042523/

====================================================================
                VIMMON COMMIT UPGRADE TOOL
====================================================================

2025-01-05 16:23:03,701 INFO Runner - Commit Tool for Cvimmon Upgrade started
2025-01-05 16:23:03,870 INFO Runner - Checking Snapshot
2021-01-05 16:23:03,870 INFO Runner - Adding metadata to the Metrics
2021-01-05 16:23:03,886 INFO Runner - Copying old metrics to the Storage

CVIMMON Snapshot Tool Completed Successfully!!

The logs for this run are available at /var/log/cvimmonha/20250105_162303_commit_cvimmon_upgrade
/commit_cvimmon_upgrade.log
```

Obtain the grafana endpoint using *k8s_run* command. Login to unified Grafana UI.

```
#  k8s_run --get-endpoint
```

Upgrade steps: HA VIM MON from 4.2.x (x=1,2,3,4) to 4.4.y (y= 3,4)

Upgrade the VIM MON pod from 4.2.x (x=1,2,3,4) to the current release of HA VIM MON POD using the following steps:

NOTE : The example shown below may refer to different hvim tag. Kindly ignore that.

1. Copy the latest autobackup from the management node (running with 4.2.x)

```
## Auto backup location
# cd /var/cisco/cvimmonha_autobackup
```

© 2026 HCLTech. All rights reserved.

Move the auto backup to another Remote server.

```
# rsync -e ssh -go -rtypX --numeric-ids
/var/cisco/cvimmonha_autobackup/<cvimmonha_autobackup_4.2.x_2021- 03-04_00:21:34> <Destination-
Server>:/<path>
```

Re-image the HA management node with the ISO of current release version with the same IP address.
Copy the backup file to the management node from the Remote server.

```
# rsync -e ssh -go -rtypX --numeric-ids root@<Destination-server>:/path/ /root/
```

Ensure sufficient disk space is available in the management node for performing backup/restore operation

Run the **prepare_cvimmon_upgrade** tool with the backup. The tool would perform snapshot of the prometheus TSDB from the backup files.
For 4.4.3 onwards Minio backup is available:

We have to run prepare upgrade script as below :

prepare_cvimmon_upgrade --storage_location /root/Databackup1 --backup_location
/var/cisco/cvimmonha_autobackup_4.2.4
_20230531_054750.tgz -st 4.2.4 -n 4.4.3 -d
Note: -st for Starting TAG
-d for Downloading the data
-r for restore the minio_backup in dashboard

© 2026 HCLTech. All rights reserved.

```
[root@arkansas ~]# prepare_cvimmon_upgrade --help
================================================================
                CVIMMON PREPARE UPGRADE TOOL
================================================================

usage: prepare_cvimmon_upgrade [-h] --storage_location STORAGE_LOCATION
                               --backup_location BACKUP_LOCATION
                               --startimagetag STARTIMAGETAG --endimagetag
                               ENDIMAGETAG [--download_data]

optional arguments:
  -h, --help            show this help message and exit
  --storage_location STORAGE_LOCATION, -s STORAGE_LOCATION
                        Storage location to store the snapshot data
  --backup_location BACKUP_LOCATION, -b BACKUP_LOCATION
                        Location of 3.4.x backup
  --startimagetag STARTIMAGETAG, -st STARTIMAGETAG
                        Choose the starting Image tag;
  --endimagetag ENDIMAGETAG, -n ENDIMAGETAG
                        Choose the end image tag
  --download_data, -d   minio backup data restore
[root@arkansas ~]#

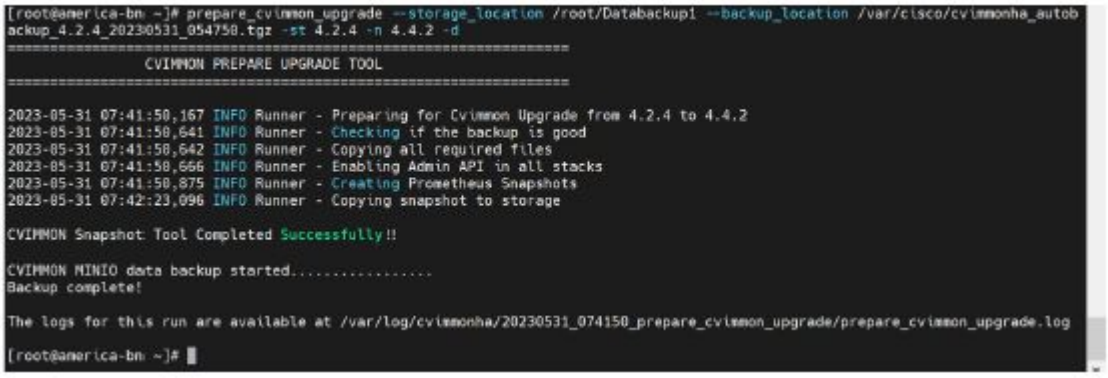
 # prepare_cvimmon_upgrade  --storage_location /var/Databackup1 --backup_location /var/cisco
/cvimmonha_autobackup_4.2.4_20230531_054750.tgz -st 4.2.4 -n 4.4.3 -
d   ================================================================
                CVIMMON PREPARE UPGRADE TOOL
================================================================

2021-03-04 15:47:33,930 INFO Runner - Preparing for Cvimmon Upgrade from 4.2.x to 4.4.2
2021-03-04 15:47:33,930 INFO Runner - Checking if the backup is good
2021-03-04 15:47:33,931 INFO Runner - Copying all required files
2021-03-04 15:47:33,946 INFO Runner - Enabling Admin API in all stacks
2021-03-04 15:51:05,748 INFO Runner - Creating Prometheus Snapshots
2021-03-04 15:51:14,929 INFO Runner - Copying snapshot to storage

CVIMMON Snapshot Tool Completed Successfully!!

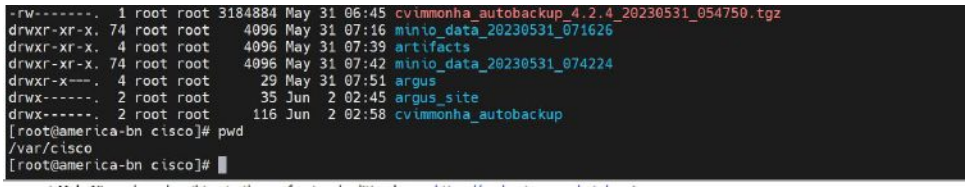
The logs for this run are available at /var/log/cvimmonha/20210304_154733_prepare_cvimmon_upgrade
/prepare_cvimmon_upgrade.log

Please find below screenshot for your reference
```



Note: from 5.0.3 onwards prepare_cvimmon , commit_cvimmon_upgrade changed as prepare_vimmon ,commit_vimmon_upgrade else use same command.

Kindly Verify Minio_backup is available after post prepare_cvimmon_upgrade activity in /var/cisco/ Path as below



© 2026 HCLTech. All rights reserved.

Run the HA VIM-MON installation operation using the *k8s_runner* CLI.

```
#  k8s_runner --install --setupfile <setupdata file >
```

Refer to the new setup data format setup_data.yaml.CVIMMONHA.EXAMPLE under /installer-xx/openstack -directory.

GLOBAL CVIMMON MONITOR STACK PARAMETERS - section outlines the changes to be done for LDAP , SNMP , Log rotate configurations in the setup data.

The parameters are moved from stack level to global POD level.

Once the install is completed, verify that the HA VIM MON cluster is up and running .

Run the commit_cvimmon_upgrade tool with the snapshot path to restore the data.

```
commit_cvimmon_upgrade --help
====================================================================
                CVIMMON COMMIT UPGRADE TOOL
====================================================================

usage: commit_cvimmon_upgrade [-h] --snapshot_location SNAPSHOT_LOCATION

optional arguments:
  -h, --help            show this help message and exit
  --snapshot_location SNAPSHOT_LOCATION, -s SNAPSHOT_LOCATION
                        Storage location of the sanpshot prepared by prepare
                        Tool

# commit_cvimmon_upgrade --snapshot_location /root/Databackup1/snapshot_20230531_071552 -r /var/cisco
/minio_data_20230531_071626/    ====================================================================
                CVIMMON COMMIT UPGRADE TOOL
====================================================================

2021-03-05 16:23:03,701 INFO Runner - Commit Tool for Cvimmon Upgrade from 4.2.x to 4.4.x
2021-03-05 16:23:03,870 INFO Runner - Checking Snapshot
2021-03-05 16:23:03,870 INFO Runner - Adding metadata to the Metrics
2021-03-05 16:23:03,886 INFO Runner - Copying old metrics to the Storage

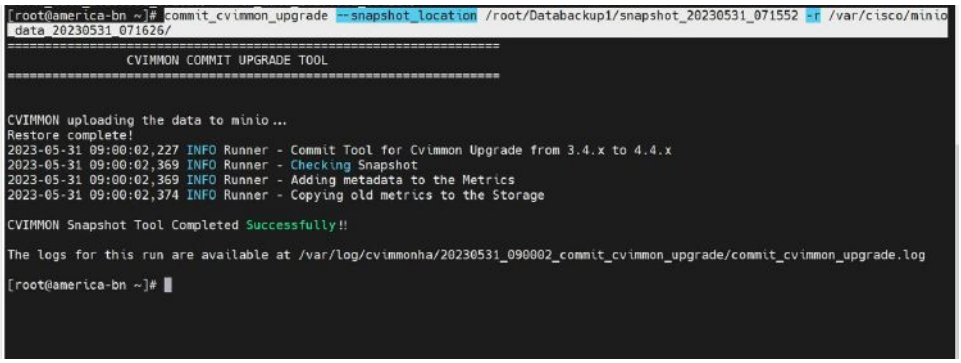
HVIMMON Snapshot Tool Completed Successfully!!

The logs for this run are available at /var/log/cvimmonha/20210305_162303_commit_cvimmon_upgrade
/commit_cvimmon_upgrade.log
```

The Cloud admin should use the below command for upgrade from 4.2.x to 4.4.y

commit_cvimmon_upgrade --snapshot_location </var/Databackup1/snapshot_20230531_071552> -r </var
/cisco/minio_data_20230531_071626/>

© 2026 HCLTech. All rights reserved.



10. Obtain the grafana endpoint using k8s_run command. Login to unified grafana UI .

```
#  k8s_run --get-endpoint
```

In Grafana UI, verify below dashboards HA VIM-MON Cluster Health

HA VIM-MON Cluster Resource Usage HA VIM-MON Pods Resource Usage HA VIM-MON Storage Health Node level metrics example:



SERVER MON

> ⚠ SERVER-MON feature support for HA vim MON is enabled as tech preview in cvim 4.2.2
>
> This feature supports only for UCS servers.

Enabling SERVER-MON on HAVIMMON POD

You can enable SERVER-MON on an existing pod that is installed with VIM 4.2.2 or later, through the reconfigure option by extending the *setup_dat a.yaml* file with relevant information.

- SERVER_MON: Enables collecting UCS-C bare metal alerts into VIM-MON for Cisco UCS C-series servers.
- MANAGEMENT_SERVERS: Enables collecting UCS-C bare metal alerts of external UCS servers, for example, management node, SDS or UM node.

© 2026 HCLTech. All rights reserved.

SERVER_MON is a prerequisite for this option.
Example of *setup_data.yaml* section

```
SERVER_MON:
  enabled: true
  host_info:
     - ALL

  rsyslog_severity: <str> # optional; supported from 4.4.x Release.


## Optional, Bare metal alerts monitoring of UCS based -  Management Node, UM or SDS servers.

# Can only be enabled when SERVER_MON is configured

MANAGEMENT_SERVERS:

 <hostname>:

    oob_info:

      oob_ip: '<ip_addr>'           # Required

      oob_username: '<username>'   # Optional, if not provides it uses the info in CIMC-COMMON section

      oob_password: '<password>'   # Optional, if not provided it uses the info in CIMC-COMMON section
```

rsyslog_severity : Adding the configuration enables the fluentd-logging service the management node. The CIMC logs are collected and forwarded to the external remote syslog server.

Checking the Target nodes getting monitored after enabling the server_mon

After the server_mon feature enabled, there are ways to see the servers being monitored by this feature.

k8s cluster will create the ucs monitor configuration and store it in configmap and this configuration contains all server details and where to send the alerts in our case it is alertmanger. Below is the reference screenshot.

Login to VIM-MON HA Management node and execute the below commands to see the ucs monitor container and its status: root@xxxxxxx ~]# **kubectl get pod -n monitoring |grep ucs-monitor**

monitoring-ucs-monitor-b786b9f67-2ch58 1/1 Running 0 17h

root@xxxxxxx ~]# kubectl exec -it monitoring-ucs-monitor-7455547944-8nnjq -n monitoring -- /bin/bash - Command to login inside the container

[ucsmonitor@monitoring-ucs-monitor-b786b9f67-2ch58 ucs-monitor]$ cd /docker/ucs-monitor/
[ucsmonitor@monitoring-ucs-monitor-b786b9f67-2ch58 ucs-monitor]$ ls ucs_monitor_config ucs_server_info.json

[ucsmonitor@monitoring-ucs-monitor-b786b9f67-2ch58 ucs-monitor]$ cat ucs_monitor_config PODNAME: "america"

ALERTMGR_URL: "http://thanos-alertmanager.monitoring.svc.cluster.local:80/api/v1/alerts" UCS_SERVER_LIST: "[ALL]"
VIMMONHA: "True"
METRO_NAME: "monitoring"
[ucsmonitor@monitoring-ucs-monitor-b786b9f67-2ch58 ucs-monitor]$ cat ucs_server_info.json
{"america1": {"cimc_ip": "10.8.x.x", "cimc_pass": "passwd", "cimc_user": "admin", "cimc_hostname": "america1", "power_status": "on"}, "america2":
{"cimc_ip": "10.8.x,x", "cimc_pass": "passwd", "cimc_user": "admin", "cimc_hostname": "america2", "power_status": "on"}, "america3": {"cimc_ip": "10.8.x. x", "cimc_pass": "passwd", "cimc_user": "admin", "cimc_hostname": "america3", "power_status": "on"}}[ucsmonitor@monitoring-ucs-monitor-b786b9f67- 2ch58 ucs-monitor]$

© 2026 HCLTech. All rights reserved.

Alert Manager configuration Lookup

```
[root@america-bn ~]# kubectl describe configmap monitoring-ucs-monitor-cm -n monitoring
Name: monitoring-ucs-monitor-cm
Namespace: monitoring
Labels: <none>
Annotations: <none>

Data
====
ucs_monitor_config:
----
PODNAME: "america"
ALERTMGR_URL: "http://thanos-alertmanager.monitoring.svc.cluster.local:80/api/v1/alerts"
UCS_SERVER_LIST: "[ALL]"
CVIMMONHA: "True"
METRO_NAME: "monitoring"

ucs_server_info.json:
----
{"america1": {"cimc_ip": "10.8.224.18", "cimc_pass": "Lab1234!", "cimc_user": "admin", "cimc_hostname": "america1", "power_status": "on"}, "america2":
{"cimc_ip": "10.8.224.19", "cimc_pass": "Lab1234!", "cimc_user": "admin", "cimc_hostname": "america2", "power_status": "on"}, "america3": {"cimc_ip":
"10.8.224.21", "cimc_pass": "Lab1234!", "cimc_user": "admin", "cimc_hostname": "america3", "power_status": "on"}}
Events: <none>
```

Opensearch-Fluentd-Opensearch Dashboard for VIM-MON HA Management Node

Starting from VIM 6.0.1, we have introduced Opensearch-Fluentd-Opensearch dashboard for VIM-MON HA.

Opensearch-Fluentd-Opensearch dashboard is a set of opensource PaaS tools belonged to group of allied open standards which can capture, store, distribute and query the logs from different services running on HA VIMMON nodes.

VIM-MON HA uses a combination of open source tools to collect and monitor the VIM-MON HA cluster and management node service logs using Opensearch, Fluentd, and Opensearch dashboard.

NOTE : VIM-MON HA uses Fluentd instead of Logstash. EFK has been replaced with Openserch – Fluentd – Opensearch Dashboard. Setup file and configurations are same as EFK. **So, whenever EFK is enabled, Opensearch components will be installed in HVIM 6.0.1.**

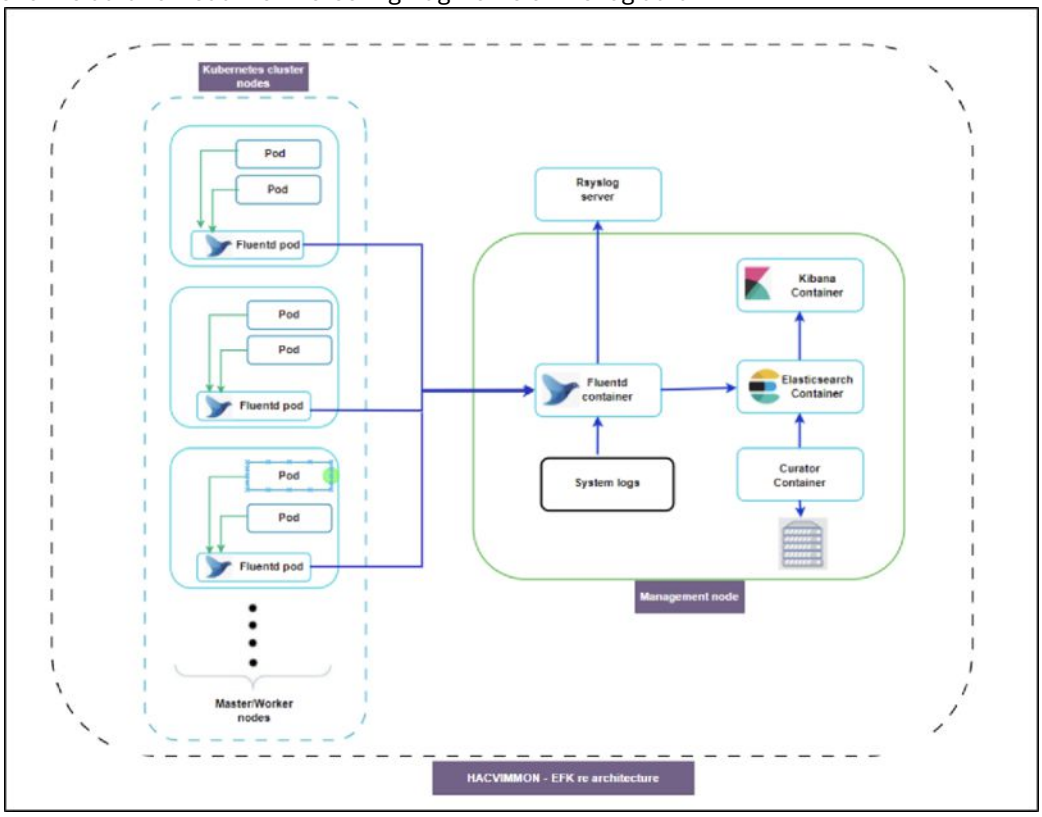
Here is the list of the components and its version used in VIM.

Component	4.2.x	4.4.3	4.4.4	5.0.4	6.0	6.0.1
Kubernetes Version	v1.17.8	v1.21.14	v1.25.14	v1.29.5	v1.29.5	v1.34.3
Helm Version	v3.5.3	v3.7.1	v3.9.0	v3.9.0	v3.9.0	v3.9.0
Prometheus	2.24.1	v2.31.1	v2.31.1	V2.31.1	V2.31.1	V2.31.1
Alert manager	v0.21.0	v0.23.0	v0.23.0	v0.23.0	v0.23.0	v0.23.0
Grafana	7.2.0	v7.5.0	v7.5.0	v7.5.11	v7.5.11	v7.5.11
Storage	Minio	Minio	Minio	Minio	Minio	Minio
Thanos	v0.17.2	v0.17.2	v0.17.2	v0.17.2	v0.17.2	v0.17.2
Stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks	Single Grafana for visualizing all stacks
Ldap, SNMP, SYSLOG	Common	Common	Common	Common	Common	Common

© 2026 HCLTech. All rights reserved.

The Opensearch-Fluentd-opensearch dashboard data flow in VIMMON HA implementation includes:

- A Fluentd container running on management and k8s cluster nodes collect the logs and forwards the logs to the Opensearch cluster running on k8s master nodes and to remote syslog server.
- Fluentd Daemon-Sets running on k8s cluster nodes collect the k8s pods logs and system logs and forwards the logs to the fluentd aggregator running on management node.
- Fluentd-aggregator running as a docker container on management node aggregates the logs received from fluentd pods along with the audit, secure and system logs on management node, forward these logs to OPensearch cluster running on k8s master nodes and also to remote syslog server.
- OPensearch stores the data, indexes it, and supports fast queries against a large amount of log data. Opensearch Curator helps to curate, or manage, Opensearch indices and snapshots by taking backup on local volumes or NFS volumes.
- OPensearch dashboard visualizes the data that is stored in Opensearch using a custom dashboard. You can also add filters to the data to visualize interesting fragments of the log data.



EFK stack component Version details:

S.No	Component	6.0.1
1	Fluentd	v4.4.1
2	Opensearch	v3.0.0
3	Opensearch Dashboard	V3.0.0
4	Curator	v5.7.5

Opensearch – Fluentd – Opensearch Dashboard stack deployment procedure on HA VIM MON:

© 2026 HCLTech. All rights reserved.

To bring up EFK stack cluster on HACVIMM MON, add the following flags in the setupdata.yaml file.

```
SERVER_MON:
    enabled: true
    host_info:
      - ALL
    rsyslog_severity: 'debug'
 EFK_STACK_K8S:
   enabled:  true
   kibana_logging:  true
```

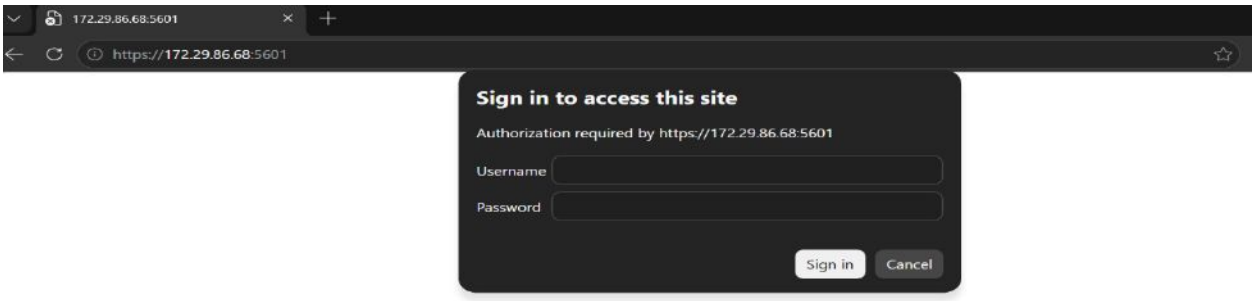
Note: 4.4.2 release supports EFK stack bring up with fresh installation only. EFK via reconfigure has been brought in from cvim 4.4.3 , 4.4.4

> EFK for CVIM-MON HA is available as Tech preview in cvim 4.4.2, 4.4.3.
>
> Starting from 4.4.4, EFK has been re-architectured; Hence while updating from 4.4.x (2,3) to 4.4.4, EFK log data porting is not possible because of the architecture change.
>
> We recommend to do fresh install of cvim-mon HA 4.4.4 for fully functional EFK.

HA VIMMON - Opensearch Dashboard Login:

Logging into Opensearch Dashboard:

To Login to Opensearch Dashboard use the https://<management_node_ip>:5601

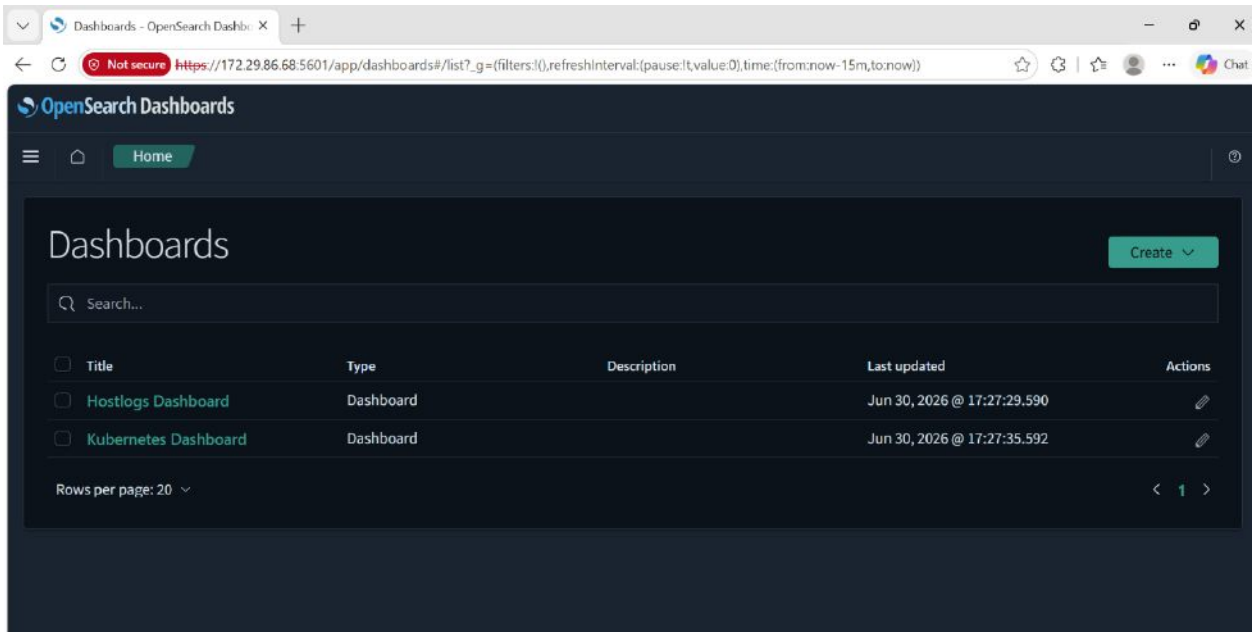


Get the KIBANA_PASSWORD from **~/openstack-configs/secrets.yaml** and username is **"admin"**

```
[root@antarctica ~]# cat ~/openstack-configs/secrets.yaml
Minio-Password-storage(Username:admin): TO6glxSWLwU2nvmNXb+acdhs/0Xz6uKxrlJstmdfY4Y=
Prometheus-Password-cvimmon-monitor(Username:admin): RB2fuzHb25KUdr2i8s4VxZdNzOvaUabODcmqUBebZuM=
Thanos-Password-monitoring(Username:admin): eI7XzXmckhlxqBsqmhz+e0OaHLgCdk7+iP6gGdsCab0=
Grafana-Password-monitoring(Username:admin): 0Rl3hO5iI4Mmq6bMiGNXXH/T1SiiQgN5nHqm4CNiQc4=
Grafana-ReadOnly-Password-monitoring(Username:cvimmon): oUsZ2t6U3fc8NSOnPjtla/o1aei3L9ULvWxyzJwhC3s=
Prometheus-Password-stack1(Username:admin): z+V7PQmO5CXc7MiQsKr84CzaxfHyK8v9XqgDW8vT90M=
Prometheus-Password-stack2(Username:admin): P7iucvd1WTNPkg665uJiwm2dmMd74X9ln1JbU5Jdvr8=
KIBANA_PASSWORD: BHcwHMnbMPD7C2NEZuSXasdc091YGSpLBaTXsBVw2MA=
[root@antarctica ~]#
```

After entering the credentials, Opensearch dashboards page will be displayed as default home page.

© 2026 HCLTech. All rights reserved.



Opensearch – Fluentd - Opensearch stack reconfigure support:

We can bring up or reconfigure Opensearch – fluentd – Opensearch dashboard stack components as day1 operation with the --reconfigure operation.

```
k8s_runner --reconfigure --setupfile
```

Reconfigure support on HA VIMMON re architecture available from 4.4.4 release cut. The below parameters used on setup_data.yaml file to reconfigure EFK stack.

```
#Optional, Enable this  to collect kubernetes, audit logs from k8s cluster
nodes using fluentd EFK_STACK_K8S:
enabled:  <true or false> # required kibana_logging:  <true or false> # required
elk_k8s_rotation_frequency: "<string>" #Optional elk_k8s_rotation_size: <float>
#Optional
elk_k8s_rotation_del_older: <int> #optional
```

EFK_STACK_K8S: parameter is optional in setup_data.yaml file.
Note: enabled & kibana_logging parameters are required under "EFK_STACK_K8S".
elk_k8s_rotation_size: Rotate all (open) indices (except .kibana) > elk_k8s_rotation_size ({{ elk_k8s_rotation_size }} GBs) to storage
Delete all (open) indices (except .kibana) > elk_k8s_rotation_size ({{ elk_k8s_rotation_size }} GBs) on opensearch
elk_k8s_rotation_frequency: # Available options: "daily", "weekly", "fortnightly", "monthly" #optional

elk_k8s_rotation_del_older: # Delete older than 8 units (where units depends on elk_rotation_frequency

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

LDAP Authentication for Opensearch dashboard in VIM-MON HA

Starting from VIM 4.4.4 release, kibana login for VIM-mon HA pod is supported via openLDAP and windows AD. This feature can be brought in during Day-0 (fresh installation) or Day-1 (via reconfigure)

OpenLDAP with non-secure and secure setup_data example

```
Non Secure LDAP configuration  with Openldap server
---------------------------------------------------
# setup_data.yaml snippet
CVIMMONHA_CLUSTER_MONITOR:
  KIBANA:
    ldap:
      domain_mappings:
      - attributes: {email: email, name: givenName, surname: sn, username: uid}
        bind_dn: dc=admin,dc=cisco,dc=com
        bind_password: XXXXX
        domain_name: corp_ldap1
        group_attribute: memberUid
        group_attribute_is_dn: false
        group_search_base_dns: ['ou=Groups,dc=cisco,dc=com']
        group_search_filter: (&(objectClass=posixGroup)(memberUid=%s))
        group_search_filter_user_attribute: uid
        ldap_uri: ldaps://172.XX.XX.231
        root_ca_cert: /root/cvimha_certs/openldap-ca.crt
        search_base_dns: ['dc=cisco,dc=com']
        search_filter: (uid=%s)
        #start_tls: false
        use_ssl: true
      group_mappings:
      - {group_dn: 'cn=group2,ou=Groups,dc=cisco,dc=com', org_role: Viewer}
      - {group_dn: 'cn=group3,ou=Groups,dc=cisco,dc=com', org_role: Viewer}

Secure LDAP configuration  with Openldap server
-----------------------------------------------
# setup_data.yaml snippet
CVIMMONHA_CLUSTER_MONITOR:
  KIBANA:
     ldap:
       domain_mappings:
       - attributes: {email: email, name: givenName, surname: sn, username: uid}
         bind_dn: dc=admin,dc=cisco,dc=com
         bind_password: XXXXXX
         domain_name: corp_ldap1
         group_attribute: memberUid
         group_attribute_is_dn: false
         group_search_base_dns: ['ou=Groups,dc=cisco,dc=com']
         group_search_filter: (&(objectClass=posixGroup)(memberUid=%s))
         group_search_filter_user_attribute: uid
         ldap_uri: ldaps://172.XX.XX.231
         root_ca_cert: /root/cvimha_certs/openldap-ca.crt
         search_base_dns: ['dc=cisco,dc=com']
         search_filter: (uid=%s)
         use_ssl: true
       group_mappings:
       - {group_dn: 'cn=group2,ou=Groups,dc=cisco,dc=com', org_role: Viewer}
       - {group_dn: 'cn=group3,ou=Groups,dc=cisco,dc=com', org_role: Viewer}
```

Windows AD Secure and non-secure setup_data example

The windows LDAP AD server details to be added in the /etc/hosts as below for both secure and non-secure.

```
# Please add following section in /etc/hosts file.
172.XX.XX.196 WIN-XXXXXX.mercury.local
```

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

```
Non  Secure KIBANA-LDAP configuration  with AD server
-----------------------------------------------------
# setup_data.yaml snippet
CVIMMONHA_CLUSTER_MONITOR:
  KIBANA:
    ldap:
      domain_mappings:
      - attributes: {email: mail, member_of: memberOf, name: name, surname: sn, username: sAMAccountName}
        bind_dn: CN=osuser1,CN=Users,DC=mercury,DC=local
        bind_password: XXXX
        domain_name: ad_mercury
        group_attribute: member
        group_attribute_is_dn: true
        group_search_base_dns: ['DC=mercury,DC=local']
        group_search_filter: (&(objectClass=group)(member=%s))
        group_search_filter_user_attribute: distinguishedName
        ldap_uri: ldap://WIN-XXXXXX.mercury.local
        ldap_user_name: sAMAccountName    # Mandatory incase using AD
        #root_ca_cert: /root/cvimha_certs/ad-ca.crt
        search_base_dns: ['DC=mercury,DC=local']
        search_filter: (sAMAccountName=%s)
        #use_ssl: true
      group_mappings:
      - {group_dn: 'CN=os-users,OU=OS-Groups,DC=mercury,DC=local', org_role: Viewer}
      - {group_dn: 'CN=cvim-group,DC=mercury,DC=local', org_role: Viewer}
      - {group_dn: 'CN=cvimmon-group,DC=mercury,DC=local', org_role: Viewer}

Secure KIBANA-LDAP configuration  with AD server
------------------------------------------------
# setup_data.yaml snippet
CVIMMONHA_CLUSTER_MONITOR:
  KIBANA:
    ldap:
      domain_mappings:
      - attributes: {email: mail, member_of: memberOf, name: name, surname: sn, username: sAMAccountName}
        bind_dn: CN=osuser1,CN=Users,DC=mercury,DC=local
        bind_password: XXXXX
        domain_name: ad_mercury
        group_attribute: member
        group_attribute_is_dn: true
        group_search_base_dns: ['DC=mercury,DC=local']
        group_search_filter: (&(objectClass=group)(member=%s))
        group_search_filter_user_attribute: distinguishedName
        ldap_uri: ldaps://WIN-XXXXXXX.mercury.local
        ldap_user_name: sAMAccountName    # Mandatory incase using AD
        root_ca_cert: /root/cvimha_certs/ad-ca.crt
        search_base_dns: ['DC=mercury,DC=local']
        search_filter: (sAMAccountName=%s)
        use_ssl: true
      group_mappings:
      - {group_dn: 'CN=os-users,OU=OS-Groups,DC=mercury,DC=local', org_role: Viewer}
      - {group_dn: 'CN=cvim-group,DC=mercury,DC=local', org_role: Viewer}
      - {group_dn: 'CN=cvimmon-group,DC=mercury,DC=local', org_role: Viewer}
```

LDAP Integration for VIM-MON HA Management Node

Starting from VIM-MON HA version 4.4.2, LDAP integration for the following are supported

- Non-root user login to VIM-mon HA Mgmt node via Ldap (Windows AD, OpenLdap) with both ldap and ldaps.
- Non-root user login to VIM-mon HA Mgmt node.

Grafana user login via LDAP (Windows AD, OpenLdap) with both ldap and ldaps are supported from 3.4.x (Note: from 4.4.1 versions ldap_uri with (ldap server FQDN) is supported)
VIMMON HA Management node (Non-root user).
Non-root user in VIM is planned to be called 'VIM admin' user.
This can be achieved by configuring the below (sample) setup_data attributes.

© 2026 HCLTech. All rights reserved.

Generate hash key using below command.

python -c 'import crypt; print(crypt.crypt("PASSWORD"))'

POD should have VIM_admin parameter in setup_data file.
VIM_admins:
1. {VIM_admin_password_hash: $6$1aSK.U17DL7Vod8.$AaR2WnQH9I5KmbPkA890/wG4oYQ0CS2Suu5R9OoSF.Hi4.ZkcIH48nyiVZHWj3md8QSt0xycfxzjfwllW3HtK/,
VIM_admin_username: testuser1}
{VIM_admin_password_hash:
6OuKyGXTeTQ2cKw9p$QDYkJb9GjfK17HiSNQgqiFdw6zES0eBAi8NqtzrODqFnhGwiCZTBsXB8EZZgAerUp8A4RfUhJtRqglzPDxzWB/, VIM_admin_username: testuser2}

Day-2 operation like creating a new user for VIM_admin is supported via reconfigure operation. Follow the above mentioned steps to create a hashkey and add the user details in the setup_data and reconfigure.VIMMON HA Management node (non-root user in VIM MON HA management node through LDAP.)

VIM supports management of the VIM administrators whose access to the VIM-MON HA management node can be authenticated through an external LDAP server (Open LDAP or Microsoft Active Directory (AD) based). Since VIM-MON HA management node is not a general purpose Linux box, all users accessing the VIM-MON HA management node have the same administrative privileges, that is, the enforcement is limited to user authentication and not authorization. It can be added as part of Day 0 or Day 1 activity. Multiple LDAP entries are allowed, as only the *domain_name* and *ld ap_uri* in each entry are mandatory. Ensure that the *ldap_uris* is secured over LDAPs and the TLS is enabled for the external api (external_lb_vip_tls: True). For LDAP certificate management, install the copy over the CA root certificate file of the LDAP server onto the management node and append it to the */root/ openstack-configs/haproxy-ca.crt chain*.

Starting from VIM 4.4.2 LDAP authentication with both secure and non-secure through an external LDAP server (Open LDAP or Microsoft Active Directory (AD) based) is supported.

To obtain sudo access to the management node and execute kubectl commands, you can choose one of the following options:

- If sudoers_group_name is not enabled,, manually add the user with root privileges to the wheel group in the corresponding LDAP domain, for example, usermode -aG wheel user1
- Use the optional parameter *sudoers_group_name*, so that all members listed in *sudoers_group_name* gain root privileges. This optional parameter is available only from VIM 3.4.6 onwards. Also, ensure that you enable the pod with external TLS before using this parameter.

VIM_ldap_admin can be bring on VIM-MON HA pod on both Day-0 (fresh install) and Day-1 (via reconfigure)
To enable VIM administrators with LDAP authentication over secure (i.e., **ldap_uri : ldaps**:// <>) or non-secure(.i.e., **ldap_uri : ldap://<>**) perform the following steps:, perform the following steps:

- Take a backup of the setupdata file and update the file manually with the configuration listed below during installation:

© 2026 HCLTech. All rights reserved.

```
vim_ldap_admins:
- domain_name: corp_ldap1
  ldap_uri: "ldaps://<ip_address_1:[port_1]>,ldaps://<ip_address_2:[port_2]>" or ldap_uri:
"ldap://<ip_address>"
  ldap_search_base: "dc=cisco,dc=com"
  ldap_schema: rfc2307 # Optional
  ldap_user_object_class: posixAccount # Optional
  ldap_user_uid_number: uidNumber # Optional
  ldap_user_gid_number: gidNumber # Optional
  ldap_group_member: memberUid # Optional
  ldap_enumerate: True or False         # Optional, defaults is True
  ldap_debug_level: <int>               # Optional, defaults(4)(4-9)

- domain_name: corp_ldap2
  ldap_uri: "ldaps:///<ip_address_3:[port_3]"
  ldap_search_base: "dc=cisco,dc=com"
  ldap_schema: rfc2307                  # Optional, supported possible values 'rfc2307' or 'ad'
  ldap_user_object_class: posixAccount  # Optional
  ldap_user_uid_number: uidNumber       # Optional
  ldap_user_gid_number: uidNumber       # Optional
  ldap_group_member: memberUid          # Optional
  ldap_default_bind_dn: "<string>"      # Optional
  ldap_default_authtok: "<string>"      # Optional
  ldap_default_authtok_type: "<string>" # Optional (password|obfuscated_password)
  ldap_group_search_base: "<string>"    # Optional
  ldap_user_search_base: "<string>"     # Optional
  access_provider: "<string>"           # Optional
  simple_allow_groups: "<string>"       # Optional
  ldap_id_use_start_tls: <boolean>      # Optional
  ldap_tls_reqcert: "<string>"          # Optional (never|allow|try|demand)
  chpass_provider: "<string>"           # Optional (ldap|krb5|ad|none)
  sudoers_group_name: <list>            # Optional (list of group that should be able to execute "sudo")
  ldap_enumerate: True or False         # Optional, defaults is True
  ldap_debug_level: <int>               # Optional, defaults(4)(4-9)
```

To support AD with or without Kerberos, provide the additional parameters as given below:

```
vim_ldap_admins:
  - access_provider: simple
    ....
    ....
    # modified attribute values
    ldap_schema: rfc2307 # supported possible values 'rfc2307' or 'ad'
    ldap_user_object_class: person # supported possible values 'posixAccount', 'person'

    # new attributes (for AD with LDAP or Kerberos)
    realm_name: <string> # optional, mandatory if krb5 is the auth_provider
    id_provider: ldap  # optional, default is ldap
    auth_provider: krb5 # optional, possible values ldap | krb5, defaults to ldap
    ldap_user_name: sAMAccountName # mandatory if ldap_schema is 'ad'
    kdc_server: '<fqdn>:<port>' # optional  # optional, mandatory if auth_provider is krb5
    kerberos_ticket_lifetime: <int>  # optional, relevant if krb5 is auth_provider (defaults to 24h in
sec) (0 - 2147483647)
    kerberos_renew_lifetime: <int>  # optional relevant if krb5 is auth_provider (defaults to 7d in sec)
(0 - 2147483647)
    ldap_enable_fallback_homedir: True or False # Optional default is False
    ldap_enable_override_shell: True or False   # Optional default is False
```

Multiple entries of the LDAP domain are allowed. For each entry, only *domain_name* and *ldap_uri* info are mandatory. Ensure that the *ldap_uri* is secured over LDAPs. As part of reconfiguration, you can add new *domain_name,* but the *domain_name* cannot be changed once configured.

© 2026 HCLTech. All rights reserved.

VIM

- Installation Sequence
- Deployment of Management Node
- VIM Client Details
- Pod Reinstallation
- Control and Data Plane Testing Updating VIM Software

Installation Sequence

Installation Sequence

Before you install VIM, complete the procedures in Preparing for NFVI Installation and ensure that NFVI network infrastructure is set up. If your management node does not have internet access, complete the procedure in Installation Preparation Without Internet Access. You can download and install the VIM installation files from the Internet, VIM software Hub, or the USB drive prepared for installation.

The bootstrap script is then kicked off to download the installer repository for installing Docker and its dependencies and starting the installer web service.

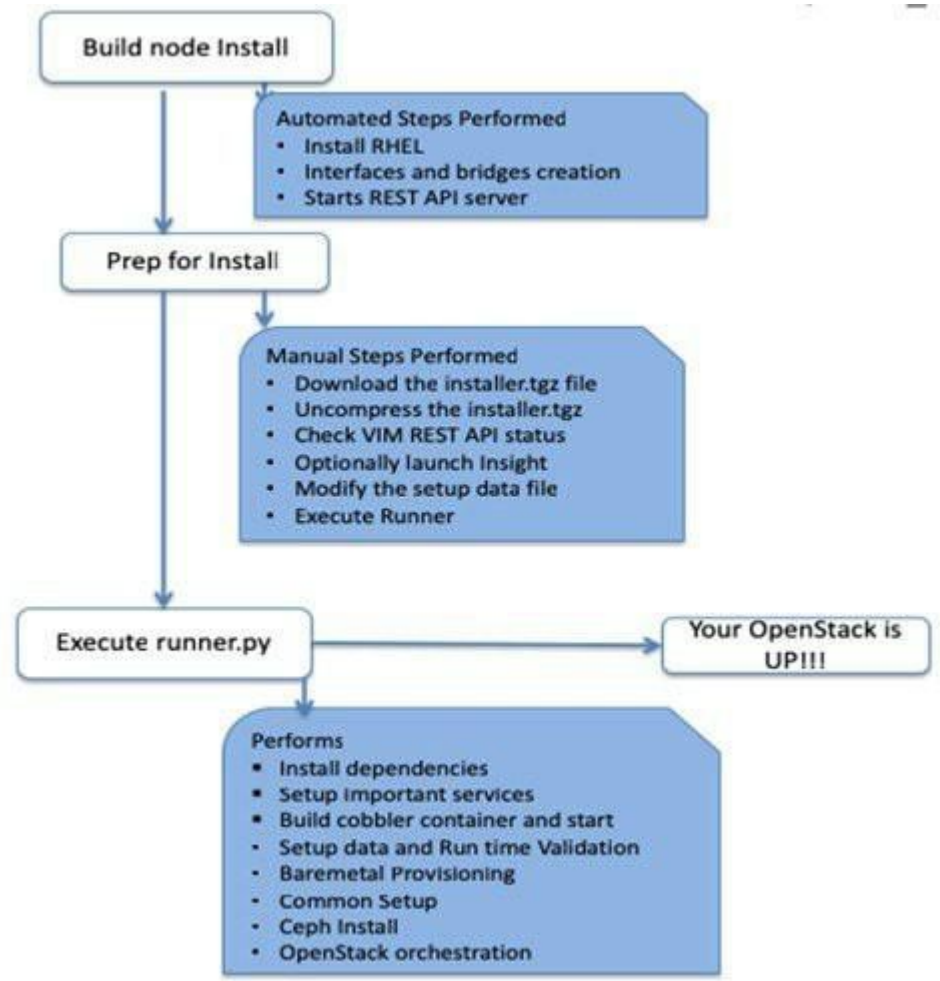
VIM installation is a multi-step process, with tests embedded in each step, to allow fixing user or environment errors and continue from the failure point of the installation. To start off, the VIM installer validates the intent file *(setup_data.yaml)* from a syntax and semantic point of view. It also validates the correctness of the management node orchestration and the hardware BOM (bill of materials) along with its BIOS settings. The orchestrator then sets up the docker registry on the management node along with other essential services, for example, cobbler, EFK in respective containers in the management node. It then creates new vNICs (for Cisco VIC) or sub-interfaces (for Intel NIC) on the controller, compute, and dedicated storage nodes based on the configuration provided in the *setup_data.yaml* file. This is followed by the Pxeboot Execution Environment (PXE) boot of RHEL onto the target nodes (control, compute, and storage) through the Cobbler server set up on the management node. Post PXE boot, the orchestrator carries out layer 2 and 3 ping checks to test out connectivity with target frame sizes.
After the installation, the VIM installer performs common steps across all the NFVI nodes.

Next, the Ceph related packages required for managing the cluster and creating Object Storage Daemon (OSD) and monitor nodes are installed on the control and storage nodes. Three Ceph monitor nodes are installed at the host level on the control nodes. These serve as management nodes and have the administration keyring. Ceph configurations such as ceph.conf and Ceph client keyrings files are stored under */etc/ceph* on each controller. Each Ceph storage node associates an OSD to a physical hard drive with a write journal on a separate SSD, to support small block random I/O.

Next, Openstack services are installed on the target servers, along with the target mechanism driver. An integrated cloud sanity is then executed to make sure that the cloud is up and running. Based on the setup_data input, an optional control, and data plane check with VMs is then run, to validate the end-to- end connectivity of the cloud from a cloud user point of view.

© 2026 HCLTech. All rights reserved.

The following illustration provides an overview of the VIM installation.



If you have Unified Management (UM), complete only part of the VIM installation procedure and proceed to the Installing UM with Internet Access procedure to complete the configuration and setup of VIM. If you do not have VIM UM, configure VIM by editing the *setup_data. yaml* file as described in the VIM installation.

Deployment of Management Node

Deployment of Management Node

This procedure allows you to install VIM on a NFVI management node:

Before you begin

- You must get NFVI installation file download site credentials from your HCLTech account representative.
  For management nodes with no Internet access, you need a USB drive containing the NFVI nstallation files.
- To prepare the USB drive, see Installation Preparation Without Internet Access.
  The private networks 192.168.1.0/24 and 192.168.2.0/24 are internally reserved for testing the cloud from a control and data plane point of view.
- We recommend that you do not use these reserved networks while preparing network layouts.
  You need to provide a valid certificate signed by a trusted certificate authority for vim deployment. It needs to be a server certificate with a common name matching the IP address and DNS name specified in the setup data file under *external_lb_vip_address* and *external_lb_vip_fqdn.* To ensure security, use only the valid certificate signed by a trusted certificate authority in a production environment.
  For details on generating a self-signed certificate, see OpenStack Configuration.

If your management node does not have Internet access, use the prepared USB drive, and complete the following steps:

- Insert the USB drive into the management node.

© 2026 HCLTech. All rights reserved.

- Run the *import_artifacts.sh* script to copy all artifacts onto the management node, for example:

```
cd ~/installer-<tag_id>/tools

./import_artifacts.sh
```

All the installation artifacts are copied *to* /var/cisco/artifacts/ on the management node.

- If you are installing VIM Insight, navigate to Unified Management and complete the VIM Insight installation. If you are not installing VIM
- Insight, complete the following steps, irrespective of the fact that it is an air-gapped or connected installation.
- Change to the installer directory by running the following command:

```
cd ~/installer-<tag_id>
```

Create a directory (for example, ~/Save/) to store a copy of the *setup_data.yaml* file that configures the NFVI for your particular implementation.

Change to openstack-configs directory and copy the example VIM *setup_data.yaml* file into the directory that you just created:

```
cd openstack-configs/

cp setup_data.yaml.<C_or_B>_Series_EXAMPLE setup_data.yaml

~/Save/setup_data.yaml
```

> ⚠ Based on your requirements, you must change the CPU and MEM allocation ratio for the target pod. Update the following parameters located in */root/installer-<xxx>/openstack-configs/openstack_config.yaml* to your target value:
>
> NOVA_RAM_ALLOCATION_RATIO: 1.5 # range of 1.0 to 4.0
>
> NOVA_CPU_ALLOCATION_RATIO: 16.0 # range of 1.0 to 16.0

In case this setting is not done on Day 0, see Memory/CPU Usage on how to set these variables.

With a yaml editor, modify the copied example *setup_data.yaml* file as the data setup file for your implementation. This includes both NFVI data and OpenStack parameters.

- If you intend to run the cloud over TLS, see OpenStack Configuration for TLS certificate generation.
- Run the installation:

```
hvim --setupfile ~/Save/setup_data.yaml run
```

After the installation is complete, you can view the installation logs at */var/log/mercury/<UUID>*.

VIM Client Details

VIM Client Details

VIM combines the CLI and API so that you can use the CLI or API installer transparently.

> ⚠ For a complete list of vim REST API commands, view the REST API Guide.

© 2026 HCLTech. All rights reserved.

Before using VIM CLI, ensure that installer API server is up and pointing to the right installer directory. Use the following command to validate the state of the API server and the installer directory:

```
# cd installer-<tagid>/tools #./restapi.py -a status

Status of the REST API Server: active (running) since Thu 2016-08-18 09:15:39 UTC; 9h ago REST API launch directory: /root/installer-<tagid>/
```

Ensure that server status is active and the REST API launch directory from where the installation began. Use the following command if the installer directory or the REST API state is not correct:

```
# cd new-installer-<tagid>/tools #./restapi.py -a setup

Check if the REST API server is running from the correct target directory #./restapi.py -a status

Status of the REST API Server: active (running) since Thu 2016-08-18 09:15:39 UTC; 9h ago REST API launch directory: /root/new-installer-<tagid>/
```

The REST API provides the options to restart, tear down, and reset password for the REST API server as listed below:

```
# ./restapi.py --h
usage:
restapi.py [-h] --action ACTION [--yes] [--verbose]

REST API setup helper optional arguments:

-h, --help            show this help message and exit
--action ACTION, -a  ACTION
                setup - Install and Start the REST API server.
                teardown -Stop and uninstall the REST API server.
                restart - Restart the REST API server.
                regenerate-password - Regenerate the password for REST API server.
                reconfigure-tls - Reconfigure SSL certificates and key.
                upgrade - Upgrade to new workspace.
                reset-password - Reset the REST API password with user given password.
                status - Check the status of the REST API server.

--yes, -y Skip the dialog. Yes, to the action.
--verbose, -v  Perform the action in verbose mode.
```

© 2026 HCLTech. All rights reserved.

If the REST API server is not active, executing *hvim* shows the following error message:

```
hvim -setupfile ~/Save/<setup_data.yaml> run
```

If the installer directory or the REST API state is not correct, or pointing to an incorrect REST API launch directory, then go to the *installer-<tagid>/tools* directory and execute the following command:

```
./restapi.py --action setup
./restapi.py --action status
```

To check whether the Rest API server state and launch directory are correct, use the following command:

If you run the REST API recovery on an existing pod, run the following command to ensure that the REST API server continues to manage the existing pod:

```
hvim --setup_file <setup_data_file_path> --perform 7 -y
```

From the CLI, enter the following command:

```
#hvim --help
usage: hvim [--setupfile <setupdata_file>] [--debug] <subcommand> ...
Command-line interface to the Virtualized Infrastructure Manager
Positional arguments:
<subcommand>
run                 Perform/terminate an install operation
install-status      Status of installation of the Openstack
cloud
list-steps          List steps
add-computes        Add compute-nodes to the Openstack cloud
add-storage         Add a storage-node to the Openstack cloud
expand-storage      Expand storage node capacity
```

© 2026 HCLTech. All rights reserved.

list-nodes	List the nodes in the Openstack cloud
power-off	Power Off compute-nodes
power-on	Power On compute-nodes
power-status	Show the power status of nodes
reboot	Reboot compute-nodes
reboot-status	List of Openstack nodes that require a reboot
remove-computes	Remove compute-nodes from the Openstack cloud
remove-storage	Remove a storage-node from the Openstack cloud
replace-controller	Replace a controller in the Openstack cloud
list-openstack-configs	List of Openstack configurations that can be changed using reconfigure
list-secrets	List of password Secrets
reconfigure	Reconfigure the Openstack cloud
cluster-recovery	Recover the Openstack cluster after a network partition or power outage
mgmtnode-health	Show status of the management node REST API
commit	Commit an update
rollback	Rollback an
update	Update the Openstack cloud
update-status	Status of the update operation
check-fernet-keys	Check whether the fernet keys are successfully synchronized across keystone nodes
nfvbench	Launch NFVbench Flows
nfvimon	NFVI Monitoring / Zenoss management operations
resync-fernet-keys	Resynchronize the fernet keys across all the keystone nodes
rotate-fernet-keys	Trigger rotation of the fernet keys on keystone
central-vm	Central Management VM Commands
client-version	Show Virtualized Infrastructure Manager Client Version
cloud-sanity	Run cloud-sanity test suite
diskmgmt	HDD maintenance helper
hardware-mgmt	Hardware validation and failure resolution

© 2026 HCLTech. All rights reserved.

```
osdmgmt                 OSD maintenance helper
version                 Show Virtualized Infrastructure Manager Version
alerts                  List all alerts | Create/Clear a test alert
hwinfo                  Get hardware info
last-run-status         Get current or last operation details
cvimmon-dashboard       Execute operations for CvimMon dashboard
help                    Display help about this program or one of its components

                                subcommands.

Optional arguments:
  -setupfile <setupdata_file>
  -debug                        Print debugging output

See "hvim help COMMAND" for help on a specific command.
To look at the help for a sub-command (e.g., run) execute the following:

# hvim help run
usage: hvim run [--setupfile <setupdata_file>] [--join]
                [--perform <perform>] [--skip <skip>] [-y] [--terminate]

Perform/terminate an install operation

Optional arguments:
  --setupfile <setupdata_file>  User setup_data.yaml
  --join                        Join the installation process
  --perform <perform>           Perform the following steps.
  --skip <skip>                 Skip the following steps.
  -y, --yes                     Yes option to skip steps without prompt
  --terminate                   Terminate the running install.

You can also run the installer in multiple smaller steps. To understand the steps involved during installation,
execute the following command:
# hvim list-steps
Virtualised Infrastructure Manager:
+---------------------+--------------+
| Operations          | Operation ID |
+---------------------+--------------+
| INPUT VALIDATION    | 1            |
| MGMTNODE ORCHESTRATION | 2         |
| VALIDATION          | 3            |
| BAREMETAL           | 4            |
| COMMONSETUP         | 5            |
| CEPH                | 6            |
| ORCHESTRATION       | 7            |
| VMTP                | 8            |
+---------------------+--------------+
```

To execute the installer in steps, include specific steps from above.

For example:

```
$ hvim run --perform 1,3 -y
```

During installation, you can use skip option to indicate which options to skip.
For example:

```
$ hvim run --skip 1,3 -y
```

⚠ During step-by-step installation, keep a track of the steps that are already completed to avoid the occurrence of unpredictable results. *hvim install-status* indicates the current state of the installation process.

© 2026 HCLTech. All rights reserved.

While the installation time varies from pod to pod, typical installation time through the Internet for a UCS C-series with three controllers, nine computes, and three storage components is listed in the following table.

Operation ID	Operation	Estimated Time
1	Input validation	6 minutes
2	Management node orchestration	40 minutes
3	Run time Validation	30 seconds
4	Bare metal	60 minutes
5	Host setup	10 minutes
6	Ceph	5 minutes
7	Orchestration	25 minutes
8	VMTP (external and provider networks)	14 minutes

Pod Reinstallation

Pod Reinstallation

You might need to reinstall the pod with the same image version. To alleviate the need for reimaging the management node followed by re-install, you can re-install the pod on assuming that the management node is compatible with the same tag.

> ⚠ For re-installation, ensure that you use the servers in which the original installation is done. If you are using a different set of servers for re-installation, power-off the servers in which the previous installation was done, to prevent duplicate IPs floating in the network.

Listed below are the steps to reinstall the pod without reimaging the management node:

Copy the setup_data.yaml from /root/openstack-configs/ directory to ~/Save/

```
cd ~/installer-<xxx>

./unbootstrap.sh -k
```

To verify that no docker containers are running, use the command:

```
docker ps -a
```

> ⚠
> - If you use NCS-55xx as a ToR, then you must execute *./unbootstrap.sh -k* before any reimage or reinstallation of the pod to a new tag.

To verify that no docker images are present, use the command:

```
docker images
```

© 2026 HCLTech. All rights reserved.

To setup RestAPI, use the command:

```
cd ~/installer-<xxx>/tools

./restapi -a setup
```

To regenerate the TLS certificate, when needed or TLS is enabled, use the command:

```
cd ~/installer-<xxx>

./tools/tls_cert_gen.sh -f ~/Save/setup_data.yaml
```

To re-run VIM installation, use the command:

```
hvim run --setupfile ~/Save/setup_data.yaml
```

Control and Data Plane Testing

Control and Data Plane Testing

VIM offers an optional integrated test to validate the control and data plane sanity of the cloud from an end-user point of view. Virtual Machine Through Put (VMTP) is an optional test available to check the Layer 2 and Layer 3 data plane traffic between NFVI compute nodes. VMTP performs ping connectivity, round trip time measurement (latency), and TCP throughput measurement for the following NFVI east to west VM-to-VM flows:

- Same network (private fixed IP, flow number 1).
- Different network using fixed IP (same as intra-tenant L3 fixed IP, flow number 2).
- Different network using floating IP and NAT (same as floating IP inter-tenant L3, flow number

3.) To enable VMTP for basic VIM installation, update the setup_data with the following commands:

VMTP_VALIDATION:
EXT_NET: # Option applicable for V4 with external network with floating IP, min of 5 cont. IP; and a min of 3 VLANS need to be defined in TENANT_VLAN_RANGES
NET_NAME: <name of external network>
NET_SUBNET: <external cidr>
NET_IP_START: <floating ip start>
NET_IP_END: <floating ip end>
NET_GATEWAY: <external net gateway>
DNS_SERVER: <dns server for external net>

PROV_NET:# Either for V4 or V6 for provider network
NET_NAME: <provider network name>
NET_SUBNET: <provider net cidr>
NET_IP_START: <starting IP for provider net>
NET_IP_END: <end IP for provider net>
NET_GATEWAY: <provider net gateway>

© 2026 HCLTech. All rights reserved.

A minimum of four continuous IP is needed for VMTP (ideally have 8 IPs). For IPv6, it is recommended to iterate over the last group to get the start and end IP addresses.

For provider network, IPv4 or IPv6 is supported, but not at the same time. Ensure that you do not enable external network for IPv6.

Updating VIM Software

Updating VIM Software

The VIM installer provides a mechanism to update all OpenStack services and some infrastructure services such as RabbitMQ, MariaDB, HAProxy, and VMTP.

Updating host-level packages and management node ELK and Cobbler containers are not supported.

As NFVI software update runs serially, component-by-component, one node at a time, the service impact is minimal.

- NFVI software updates are not supported for registry related containers and authorized_keys.
- When the management node repo containers are updated, they cannot be rolled back to the older versions as this requires node packages to be deleted, which might destabilize the cloud.
- Update of NFVI software is done within the same major version, that is, from X.Y.a to X.Y.b, and not from X.Y.z to A.B.c.

If errors occur during an update, an automatic rollback brings the cloud back to its previous state. After an update is completed, check for any functional cloud impacts. If everything is fine, you can commit the update which clears the old containers from the system. It is recommended that you commit the update before you perform any other pod management functions. Skipping the commit option might lead to double faults. If you see any functional impact on the cloud, perform a manual rollback to start the old containers again.
To prevent double faults, a cloud sanity check is done both before and after the update.

To complete the software update, perform the steps listed in Deployment of Management Node. If your management node does not have Internet, fetch the target software bits by completing the steps listed in Installation Preparation Without Internet Access first, then follow the VIM installation instructions. Differences between a software update and regular VIM installation:

- You do not need to modify setup_data.yaml like you did during the first installation. In most cases, no modifications are needed.
- You do not need to repeat the VIM Insight installation.
- Minor differences between NFVI software installation and updates are listed in the installation procedure.

- After you complete a software update and before you perform any pod management operation, you must execute the hvim commit.
- During the software update, the following operations are locked: add/remove compute/storage node, replace controllers, reconfigure, power off/on, and rotate fernet key.
- Before you commit, you can roll back the update to return the node to its previous software version.

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

Unified Management

Installing VIM Unified Management

- UM Overview
- UM with Internet Access
- UM with VIM Software Hub
- UM UI User Authentication and Authorization with LDAP
- UM Without SMTP
- UM Without Internet Access
- UM Optional Services
- UM Post Bootstrap Validation Checks
- UM Admin Login for Standalone Setup
- UM Pod Admin Login for Standalone Setup
- UM Host Administrators
- Reinstallation of UM Node

UM Overview

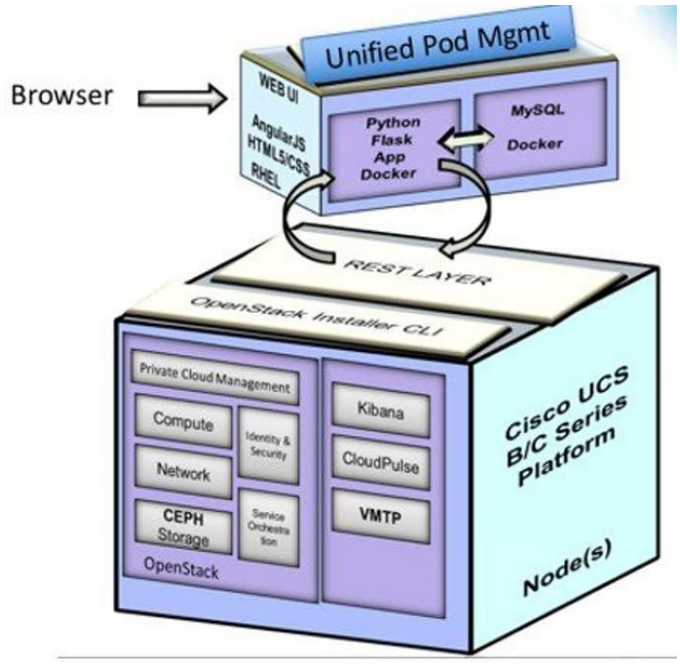
UM Overview

- Admin UI Pod UI
- UM OG/NG

Unified Management (UM) also known as VIM Insight is a light-weight UI introduced 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. VIM Insight offers a single pane of glass service to provide deployment visualization and to manage multiple VIM pods, thereby reducing user-errors. Ensure that the UM is running on a standalone VIM supported BOM.

From VIM 4.0.0, only next-generation Unified Management (UM NG) is supported. Pods upgrading from VIM 3.4.6, must run with UM NG. For installation of UM NG, ensure that the INSIGHT_NG is defined as True in the setup_data of UM. For installation details, see UM with Internet Access

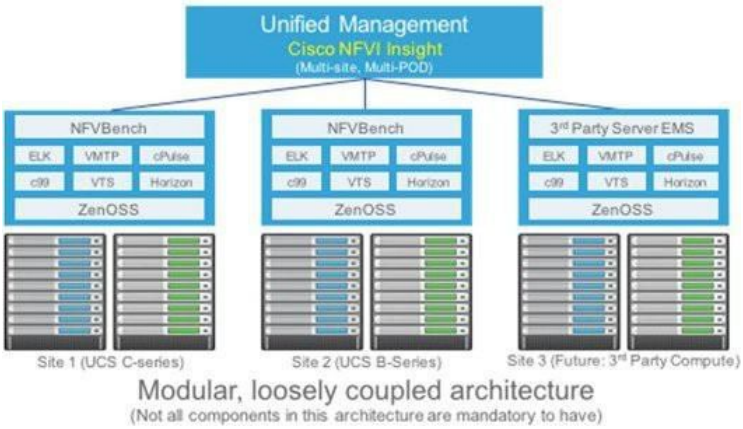
The following figure illustrates the interaction of the VIM UM with a pod.



© 2026 HCLTech. All rights reserved.

The architecture of the VIM UM is light-weight, hierarchical and scalable. It simplifies the management from the global UI. Each local site is autonomous with localized toolsets. The global UM UI provides ease of management with multi-site multi-pod capability for distributed NFV deployment at scale. VIM UM is designed to operate in HA as an option.

UM is integrated with VIM REST layer, to support multi-tenancy with local RBAC support. The container-based UI platform is loosely coupled and can help manage multiple VIM pods right from Day 0 or later in the lifecycle of the cloud as shown in the following figure.



You can install UM on a standalone (BOM same as the management node) dedicated node to manage one or more VIM pods. As the UI interacts with the REST API, it is not necessary that the pod must be managed by UM node from Day 0. The UI interacts with each pod through REST API and Role Based Access Control (RBAC) information stored in the DB.

The UI has two types of views:

- • UI Admin: UI Admin can add users as UI admin or pod admin.
- • Pod Admin: Pod admin has the privilege only at the pod level, unless Pod admin is also a UI admin.

Admin UI

Admin UI is responsible for managing the UI and Pod admin, which includes adding and revoking user privileges. The UI admin can delete an existing pod from the management pane.

Pod UI

The pod UI is responsible for managing each pod. UM provides easy access to switch between multiple pods.

Through pod UI, a pod admin can manage users and their respective roles and responsibilities.

You can execute Day 0 (installation) and Day n activities such as pod management, software update, and so on seamlessly, where n is an integer greater than 1. ELK, Horizon web UI, and so on, are also cross-launched and visible for each pod through the pod UI.

UM OG/NG

In CVIM 3.4.6, the UM is transitioned from legacy (Unified Management OS) to next-generation (Unified Management NG). While you can use both the UM versions to manage VIM pods, it is highly recommended to migrate to NG version of UM by following the steps listed in Migrating of UM to NG Instance. From VIM 4.0.0, only next-generation UM is supported. Ensure that VIM 4.0.0 pods are managed with VIM 4.0.0 UM-NG application.

© 2026 HCLTech. All rights reserved.

UM with Internet Access

Installing UM with Internet Access

- Overview
- Installation

Overview

Complete the following steps to install VIM Insight on the NFVI management node. As security is paramount to pod management, the web- service hosting the single pane of glass is protected through TLS. Following are the steps to get the TLS certificate setup going.

You can select one of the following approaches for the TLS certificate configurations:

- Provide your own certificate. You can bring in your certificate on the management node and provide the absolute path of .pem and CA certificate files in the *insight_setup_data.yaml* file. The path must be provided as a value for the key PEM_PATH in the *insight_setup_data.yaml* file.
- Generate a new certificate on the node. You can create a new certificate on the node by running the following command:
-

```
# cd /root/Insight-<tag_id>/insight/

#./tls_insight_cert_gen.py -f <path_to_insight_setup_data.yaml>/insight_setup_data.yaml
```

This script searches for the PEM_PATH inside the *insight_setup_data.yaml*. As the path is not provided, it creates a new certificate inside *install- dir/openstack-configs*.

> ⚠ The self-signed certificate generation utility script is provided for lab/testing deployment only. Ensure that you do not use a self-signed certificate generated by this utility for the production deployment.

Installation

> **Before you begin**
>
> Complete all the preparation tasks that are described in Preparing for NFVI Installation. The procedure to bootstrap the node hosting Insight is the same as installing the buildnode.iso. Ensure that you plan for a standalone unified management node for production. Click **Yes** if the node is to be used in the production.

Enter *ip a* to verify the *br_mgmt* and *br_api* interfaces are up and are bound to bond0 and bond1, respectively.

For example:

$ ip a
br_api: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP link/ether 00:42:68:6f:79:f2 brd ff:ff:ff:ff:ff:ff
inet nnn.nnn.nnn.nnn/25 brd nnn.nnn.nnn.nnn scope global br_api valid_lft forever preferred_lft forever
inet6 fe80::3c67:7aff:fef9:6035/64 scope link valid_lft forever preferred_lft forever
bond1: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue master br_api state UP
link/ether 00:42:68:6f:79:f2 brd ff:ff:ff:ff:ff:ff
br_mgmt: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP link/ether 00:78:88:46:ee:6e brd ff:ff:ff:ff:ff:ff
inet nnn.nnn.nnn.nnn/24 brd nnn.nnn.nnn.nnn scope global br_mgmt valid_lft forever preferred_lft forever
inet6 fe80::278:88ff:fe46:ee6e/64 scope link valid_lft forever preferred_lft forever

© 2026 HCLTech. All rights reserved.

bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue master br_mgmt state UP link/ether 00:78:88:46:ee:6e brd ff:ff:ff:ff:ff:ff

⚠The *br_mgmt* and *br_api* interfaces are created when you install the RHEL on the management node as in Management Node on UCS C-series (ssM5/M6).

Run the following commands to copy the installer directory and the standalone *insight_setup_data_yaml*.
Copy the installer directory to a directory in */root/*. Start the name of the new directory with *Insight-tag_id*.

```
# cd /root/
# cp -pr installer-<tag_id> <Insight-tag_id>
```

Copy the standalone *insight_setup_data.yaml*. Standalone_EXAMPLE file from the *Insight-dir/openstack-configs* to any other location on the management node or the BOM.

```
# cp /root/Insight-<tag_id>/openstack-configs/insight_setup_data.yaml.Standalone_EXAMPLE /root
/insight_setup_data.yaml
```

Modify the insight setup data according to your requirements.

```
##################################################################################
##
#                    !! INSIGHT SETUP CONFIGURATIONS !!          #
#                       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~                   #

# User Defined Configuration File
# Information in this file is specific to the user setup
# UM -> Unified Management (Insight)

##################################################################################
#                                INSIGHT NG UI                                  #
##################################################################################
# NG -> New Generation

# Optional, valid options are True and False. # Non-reconfigurable, not enabled by
default. INSIGHT_NG: <True     or False>

##################################################################################
#                         REGISTRY      INFORMATION                            #
##################################################################################
# Mandatory parameters
REGISTRY_USERNAME: '<username>'
REGISTRY_PASSWORD: '<password>'
```

© 2026 HCLTech. All rights reserved.

```
        # Mandatory Parameter when SDS is enabled # Not required when SDS is disabled
        # Example registry FQDN name [your.domain.com] REGISTRY_NAME:
'<registry_name>'

##############################################################################
#                                  INSTALLATION MODE                         #
##############################################################################
# Optional parameter # Valid options:
        # a) connected (default): Node has connectivity to https://VIM-registry.com
        #
        # b) disconnected: Node has NO connectivity to https://VIM-registry.com
        # For details, see Installing UM without Internet Access section #
        INSTALL_MODE: connected

##############################################################################
#                              NETWORKING INFORMATION
#
##############################################################################
# Optional, must have at least one sub key defined.
NETWORKING:
        # Optional, needed if the pod is behind a proxy
        # Name of the proxy server without 'https://'
        # Not required for INSTALL_MODE: disconnected
        https_proxy_server: '<proxy.domain.com:8080>'
        # Optional; max of 4 enteries in the list
        ntp_servers:
          - <1.pool.ntp.org>
        # OR
        # Support for IPv6 address
        ntp_servers: ['2001:c5c0:1234:5678:1002::1', 15.0.0.254]
```

© 2026 HCLTech. All rights reserved.

```
##########################################################################
#                          INSIGHT SUPER ADMIN                           #
##########################################################################

#User will be the default Super Admin of the system and can grant access to # all other users getting registered to VIM pods.

# Mandatory parameters UI_ADMIN_USERNAME: '<username>'

UI_ADMIN_EMAIL_ID: '<email_id@domain.com>'


##########################################################################
#                          SMTP CONFIGURATIONS                           #
##########################################################################

# Set this to False, if there is no access to SMTP server # Optional, valid options are True and False (default)

SMTP_MODE: False


# Following SMTP/EMAIL settings are mandatory "ONLY when" SMTP_MODE is True # Skip the below keys, if SMTP_MODE is False


# Define the mail server off which the Insight email alias works. # For example, outbound.cisco.com

# Mandatory: Valid SMTP Server is required for sending mails to the customers. INSIGHT_SMTP_SERVER: <smtp.domain.com>

# Optional, defaults to 25 if undefined INSIGHT_SMTP_PORT: <port no.>

# For Insight UI, you must create a mailer, so that all automated mails are # sent from that alias.

# For example: 'VIM-insight-admin@cisco.com' # Mandatory

INSIGHT_EMAIL_ALIAS: <email-alias@domain.com>

# Optional: Insight Email Alias Password is required if login on a SMTP server # requires authentication.

INSIGHT_EMAIL_ALIAS_PASSWORD: <password>

##########################################################################
#
```

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

```
#                                    TLS CERT AND HOST INFORMATION                    #
##############################################################################

# TLS certificate for INSIGHT UI.

# Absolute TLS certificate path can also be generated using the script # tls_insight_cert_gen.py located at insight-<tag_id>/insight/.

# If generated by: tls_insight_cert_gen.py, then below entry is optional. PEM_PATH: <abs_location_for_cert_path>

# Mandatory in case PEM_PATH is defined.

SSL_CERT_CHAIN_FILE: <abs_location_for_cert_chain_file of x509 certificate>


# If using tls_insight_cert_gen.py to create the cert, define the following: # Mandatory

CERT_IP_ADDR: <br_api_of_the_UM_node>

# Optional

CERT_HOSTNAME: <domain_name_for_cert>

############################################################################## #
                                LDAP CONFIGURATIONS                                  #
##############################################################################

# LDAP Configurations to enable LDAP users to manage Insight UI. # Optional, valid options are True and False (default)

LDAP_MODE: False

        # Following LDAP settings are required "ONLY when" LDAP_MODE is True.

# Skip the below keys if LDAP_MODE is False


# Mandatory, valid options are True and False. # True is for secure connection(ldaps).

LDAP_SECURE: <True or False>


# Mandatory, IP address of the LDAP Server or Hostname when LDAP_SECURE is # True with a trusted Root CA certificate.

LDAP_SERVER: <ip|hostname_of_ldap_server>


# Mandatory, LDAP configured port. LDAP_PORT: <port no.>
```

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

```
# Mandatory, Base DN of the LDAP server. # E.g., 'dc=cisco,dc=com'

LDAP_BASE_DN: '<Base_DN_of_ldap_server>'


# Optional, LDAP search filter to search groups on LDAP. # Defaults to '(objectClass=posixGroup)'
LDAP_GROUP_SEARCH_FILTER: '<ldap_group_search_filter>'


# Optional, LDAP search filter to search group-members on LDAP. # Defaults to
'(objectClass=posixAccount)' LDAP_GROUP_USER_SEARCH_FILTER:
'<ldap_group_user_search_filter>'


# TLS certificate path.

# Required when LDAP_SECURE is True for self-signed certificates only.

# In case of trusted Root-CA-signed certificate, this key is not required. LDAP_CERT_PATH:
'<abs_location_for_cert_path>'


# Mandatory, LDAP attribute which can be used as user-id # E.g., '<'uid' or 'cn' or 'mail'>'
LDAP_USER_ID_ATTRIBUTE: <ldap_user_id_attribute>


# Optional, complete DN of the admin user for bind and search. # E.g., '<dn=admin, dc=example,
dc=com>'

LDAP_ADMIN: '<user-DN for admin>'


# Optional, password of the bind admin user: LDAP_ADMIN LDAP_ADMIN_PASSWORD: '<password>'


# Optional, attribute on the LDAP server used to define group members. LDAP_GROUP_MEMBER: '<
memberUid >'


# Optional, attribute on the LDAP server used to define role associated members.
LDAP_ROLE_MEMBER: '< member >'


# Optional, valid options are True and False (default)
```

© 2026 HCLTech. All rights reserved.

```
# Enable this to use Insight permissions mapping keys with LDAP group/role.
LDAP_AUTHORIZATION: False


# Following are the optional keys to map Insight permissions with LDAP group/role
# These Group/Role-DN value and LDAP_BASE_DN are used to create full Group/Role-DN. #
UM_ADMIN_GROUP: 'cn=VIM-um-admin,ou=Roles'
# LDAP_BASE_DN: 'dc=cisco,dc=com'
# Full-DN mapped as UM-Admins:- UM_ADMIN_GROUP + LDAP_BASE_DN


# Optional, LDAP Group-DN to be mapped as Insight UM-Admins. # Defaults to 'cn=VIM-um-admin,ou=Roles'
UM_ADMIN_GROUP: <um_admin_group>


# Optional, LDAP Group-DN to be mapped as Insight Pod-Admins # Defaults to 'cn=VIM-pod-admin,ou=Roles'
POD_ADMIN_GROUP: <pod_admin_group>


# Optional, LDAP Group-DN to be mapped as Insight Pod-Users[Write and Read Permissions] # Defaults to 'cn=VIM-pod-user,ou=Roles'
POD_USER_GROUP: <pod_user_group>



# Optional, LDAP Group to be mapped as Insight Pod-Users[Read Only]
# Defaults to 'cn=VIM-read-only,ou=Roles' READ_ONLY_POD_USER_GROUP:
<read_only_pod_user_group>


############################################################################## #
                        OPTIONAL FEATURES                                     #
##############################################################################
# Automatically assigns the pod_user permissions to um_admin during pod # registration with default role 'full-pod-access'.
# Optional, valid options are True and False (default)
```

© 2026 HCLTech. All rights reserved.

```
UM_ADMIN_WITH_FULL_POD_ACCESS: False

# Display all pod-users as suggestion at new pod-user registration. # Optional, valid options are True and
False (default)

DISPLAY_ALL_POD_USERS: False


############################################################################## #
                                    UM ADMINS                                 #
##############################################################################

# Optional, options for non-root user. # Can have one or more admins.

# Each VIM admin must have a VIM_admin_password_hash, a VIM_admin_public_key, # or both.


## 1. VIM_admin_username: ADMIN USER NAME        |

##----------------------------------------------------------------------------

## VIM admin user names should satisfy following criteria: ##    a. Required

##      b. Unique

##      c. ASCII chars

##      d. No space allowed ##   e. 1 <= Length <=32

##      f. Only lower case letters

##      g. Digits, underscores, or dashes

##      h. First character must be a letter or an underscore


## 2. VIM_admin_password_hash: ADMIN USER PASSWORD HASH     |

##----------------------------------------------------------------------------

## Optional, to generate a password hash:

## -> python -c 'import crypt; print crypt.crypt(("<plaintxt_strong_pwd>"))'



## 3. VIM_admin_public_key: ADMIN USER PUBLIC KEY          |

##----------------------------------------------------------------------------
```

© 2026 HCLTech. All rights reserved.

Optional. VIM_admin_public_key is a user's public key. ## It can be generated with 'ssh-keygen'.

Should be a string that starts with "ssh-rsa AAAA" or "ssh-ed25519 AAAA"

Optional, must have at least one sub key defined. VIM_admins:

VIM admin with only password hash.

1. VIM_admins:
non_root_admin_1 VIM_admin_password_hash: $6....

VIM admin with password hash and public key

2. VIM_admins:
non_root_admin_2 VIM_admin_password_hash: $6....

VIM_admin_public_key: ssh-rsa AAAA... or ssh-ed25519 AAAA...

VIM admin with only public key

3. VIM_admins:
non_root_admin_3

VIM_admin_public_key: ssh-rsa AAAA... or ssh-ed25519 AAAA...

#
SSH ACCESS CONFIGURATIONS #
##

Optional, valid options are True and False (default) # True: root can SSH to the UM management node.

False: root cannot SSH to the UM management node.

At least one VIM_admin must be configured if this is False permit_root_login: False

Optional # Eg,

WARNING: Unauthorized access to this system is forbidden and will be

prosecuted by law. By accessing this system, you agree that your actions # may be monitored if unauthorized usage is suspected.

ssh_banner: <warning_msg>

#
UM LDAP ADMINS #
##

Optional, enable LDAP users to SSH to the UM management node. # Can be added as day-0 or day-1 as part of reconfigure.

© 2026 HCLTech. All rights reserved.

```
# During reconfigure, all but domain_name can be changed.

# New domain_name can be added as part of reconfigure, but once configured # cannot be changed.

# Optional, contains list of ldap mappings

# Must have at least one element defined in its list UM_LDAP_ADMINS:

# All below keys are mandatory
                                    4.  domain_name:corp_ldap2 ldap_search_base:"dc=hcltech,dc=com" ldap_uri:"<uri_val>"

# For multiple ldap server pass comma separated value. ldap_uri: "ldaps://172.29.84.231"


# All below keys are optional ldap_schema: rfc2307 ldap_user_object_class:posixAccount
ldap_user_uid_number: uidNumber ldap_user_gid_number: uidNumber ldap_group_member:
memberUid ldap_default_bind_dn: "<string>" ldap_default_authtok: "<string>"


# (password|obfuscated_password) ldap_default_authtok_type:"<string>" ldap_group_search_base:
"<string>" ldap_user_search_base: "<string>" access_provider: "<string>" simple_allow_groups:
"<string>" ldap_id_use_start_tls: <boolean>


# never|allow|try|demand) ldap_tls_reqcert:"<string>"


# (ldap|krb5|ad|none) chpass_provider:"<string>"


# Modified attribute values for MS-AD

# Supported possible values 'rfc2307' or 'ad' ldap_schema: rfc2307

# Supported possible values 'posixAccount', 'person' ldap_user_object_class: person



# New attributes (for MS AD with ldap or kerberos) # Optional, mandatory if krb5 is the auth_provider
realm_name: MERCURY.LOCAL

# Optional, default is ldap id_provider: ldap

# Optional, possible values ldap and krb5, defaults to ldap auth_provider: krb5

# Mandatory if ldap_schema is 'ad' ldap_user_name: sAMAccountName

# Optional, mandatory if auth_provider is krb5 kdc_server: "<fqdn>:<port>"
```

© 2026 HCLTech. All rights reserved.

Optional, relevant if krb5 is auth_provider (defaults to 24h in sec) (0 - 2147483647) kerberos_ticket_lifetime: <int>
Optional, relevant if krb5 is auth_provider (defaults to 7d in sec) (0 - 2147483647)

```
    kerberos_renew_lifetime: <int>

#############################################################################
#                        REMOTE SYSLOG                                      #
#############################################################################
# Optional, can be enabled by reconfigure action
SYSLOG_EXPORT:
   # Required: List of remote servers
   remote_hosts:
        # Required; remote syslog ipv4/ipv6 address
      - host: '<ip_addr>'
        # required; typically, 514
        port: <int>
        # Required; possible values: user, local1, local2, local3, local4,
        # local5, local6, local7
        facility: user or local[1-7]
#############################################################################
```

Save the edited insight_setup_data.yaml file.

Start the insight installation process:

```
$ cd /root/Insight-<tag_id>/
$ ./insight/insight_runner.py -h
usage: insight_runner.py [-h] [-f SETUPFILE] [-y] [-s SKIP_STEPS]
                         [-p PERFORM_STEPS] [-l] [-v]
                         [--update] [--dry-run] [--commit] [--rollback]
                         [--reconfigure] [--regenerate_secrets]
                         [--setpassword] [-i] [-u] [--list-reconfigure-keys]

INSIGHT ORCHESTRATOR

optional arguments:
  -h, --help            Show this help message and exit
  -f SETUPFILE, --setupfile SETUPFILE
                        User input file, default is insight_setup_data.yaml
  -y, --yes             Yes option to skip steps without prompt
  -s SKIP_STEPS, --skip_steps SKIP_STEPS
                        Comma separated list of steps to skip. eg -s 1
  -p PERFORM_STEPS, --perform_steps PERFORM_STEPS
                        Comma separated list of steps to perform. eg -p 2
  -l, --list_steps      List steps
  -v, --version         Display version info
  --update              Update Insight
  --dry-run             Dry Run to find all containers marked for update
  --commit              Commit Insight Update
  --rollback            Rollback Insight Update
  --reconfigure         Reconfigure Insight DB, TLS, and UI
  --regenerate_secrets  System generated Insight DB_ROOT_PASSWORD
  --setpassword         User supplied Insight DB_ROOT_PASSWORD
  -i, --install-status  Display Insight install status
  -u, --update-status   Display Insight update status
  --list-reconfigure-keys
                        List setup data reconfigurable keys

$ ./insight/insight_runner.py -l

           !! INSIGHT !!
VIM UNIFIED MANAGEMENT ORCHESTRATOR
===================================
 +--------------------+-----------------+
 |     Operations     |  Operation ID   |
 +--------------------+-----------------+
 |  INPUT_VALIDATION  |        1        |
 |  BOOTSTRAP_INFRA   |        2        |
 +--------------------+-----------------+

$ ./insight/insight_runner.py -f </root/insight_setup_data.yaml>

Perform steps ['1', '2']. Continue (Y/N)y
```

© 2026 HCLTech. All rights reserved.

```
The logs for this run are available at /var/log/insight/bootstrap/<date>_<time>

##############################
VIM INSIGHT ORCHESTRATOR
##############################

[1/2][INPUT_VALIDATION: INIT]          [          \          ]          0min 1sec

Management Node validation!
+-----------------------+------+-----------------------------------------------------------------------+
| Rule                  | Status | Error|
+-----------------------+------+-----------------------------------------------------------------------+
| Check Kernel Version          | PASS       | None |
| Check Ansible Version         | PASS    | None |
| Check Docker Version          | PASS    | None |
| Check Management Node Tag     | PASS    | None |
| Check Bond Intf. Settings     | PASS    | None |
| Root Password Check           | PASS      | None |
| Check Boot Partition Settings | PASS    | None |
| Check LV Swap Settings        | PASS    | None |
| Check Docker Pool Settings    | PASS    | None |
| Check Home Dir Partition      | PASS    | None |
| Check Root Dir Partition      | PASS    | None |
| Check /var Partition          | PASS       | None |
| Check LVM partition           | PASS       | None |
| Check RHEL Pkgs Install State | PASS    | None |
+-----------------------+------+-----------------------------------------------------------------------+

Insight standalone Input validation!
+-----------------------+------+-----------------------------------------------------------------------+
| Rule | Status | Error |
+-----------------------+------+-----------------------------------------------------------------------+
```

© 2026 HCLTech. All rights reserved.

| Insight standalone Schema Validation | PASS | None |

| Valid Key Check in Insight Setup Data | PASS | None |

| Duplicate Key Check in Insight Setup Data | PASS | None |

| VIM/Insight Workspace Conflict Check | PASS | None |

| Check Registry Connectivity | PASS | None |

| Check LDAP Connectivity | PASS | None |

| Test Email Server for Insight | PASS | None |

[2/2][BOOTSTRAP_INFRA: VIM-admins-Restart and enable sso daemon] [DONE!] 1min 23secs

VIM Insight Installed successfully!

| Description | Status | Details|

| Insight UI URL | PASS | https://<br_api>:9000 |

| UI Admin Email ID | PASS | Check for info @: /root/insight-openstack-configs/insight_setup_data.yaml |

| UI Admin Password | PASS | Check for info @: /opt/cisco/insight/secrets.yaml |

Backup Validation Passed

[***][AUTOBACKUP: Completed Step 1 of backup, backupdir: /var/cisco/insight_backup /insight_autobackup_3.4.5_<date>_<time>] [***][AUTOBACKUP: Starting post-backup, please wait!!]

[***][AUTOBACKUP: Completed VIM UM backup ...
/var/cisco/insight_backup/insight_autobackup_3.4.5 _<date>_<time>

Ended Installation [BOOTSTRAP_INFRA] [Success]

The logs for this run are available at /var/log/insight/bootstrap/<date>_<time>

Logs of Insight Bootstrap are generated at : /var/log/insight/bootstrap/ on the UM management node.
Log file name for Insight Bootstrap are in the following tar format: insight_<date>_<time>.tar.gz

© 2026 HCLTech. All rights reserved.

```
Once the bootstrap process is completed a summary table preceding provides the information of the UI URL
and the corresponding login credentials. After first login, for security reasons, we recommend you change
the Password.

Insight autobackup takes place after the installation, and is located at default backup location /var
/cisco/insight_backup; details of which is provided below the bootstrap summary table.

To add a new UI Admin in a setup that just got created, login to VIM insight and add a new UI admin user
from the Manage UI Admin Users menu. Without doing a fresh installation (that is un-bootstrap, followed
by bootstrap) of the insight application, the UI admin that was bootstrapped cannot be changed.

Refer vim Insight Post Bootstrap Validation Checks section, to verify the bootstrap status of vim
Insight.
```

UM with VIM Software Hub

UM with VIM Software Hub

To reduce the logistics of the artifact distribution during an air-gapped installation, use the VIM Software Hub. To download the artifacts to the VIM Software Hub server, follow the instructions available at VIM Software Hub. Then, you can use the connected way of installing Unified Management (UM) on the UM node.

To install UM on the UM node through VIM Software Hub, you need REGISTRY_NAME as an additional field in the setup data for the UM node.

```
# Mandatory Parameter when SDS is enabled. REGISTRY_NAME: '<registry_name>'
```

Here, the registry name is fully qualified domain name (FQDN) or your domain.com. When VIM Software Hub is not enabled, you must not use this parameter.

Once REGISTRY_NAME is defined in the setup data, the UM software fetches the artifacts from the VIM Software Hub server as long as the INSTALL_MODE is defined to be connected or not defined in the *insight_setup_data.yaml* file. By default, it is assumed to be connected.

UM UI User Authentication and Authorization with LDAP

UM UI User Authentication and Authorization with LDAP

Unified Management supports both LDAP and LDAPs (Secure over SSL) for an AD (Active Directory) environment, but you can choose only one at a time. LDAPs supports connection using both self-signed and CA-signed certificate.

If you choose self-signed certificate option for LDAPs, you must have a certificate for verification over LDAPs and for secure LDAP connection to LDAP over SSL.

© 2026 HCLTech. All rights reserved.

If you choose CA-signed certificate option, no certificate is required.

```
# LDAP configurations to enable LDAP users to manage Insight UI. # Optional, valid options are True and
False (default)

LDAP_MODE: False


# Following LDAP settings are required "ONLY when" LDAP_MODE is True.

# Skip the below keys if LDAP_MODE is False



# Mandatory, valid options are True and False.

# True is for secure connection(ldaps).

LDAP_SECURE: <True or False>


# Mandatory, IP address of the LDAP Server or Hostname when LDAP_SECURE is

# True with a trusted Root CA Certificate.

LDAP_SERVER: <ip/hostname_of_ldap_server>


# Mandatory, LDAP configured port. LDAP_PORT: <port no.>

# Mandatory, Base DN of the LDAP server. # E.g., 'dc=cisco,dc=com'.
```

© 2026 HCLTech. All rights reserved.

```
LDAP_BASE_DN: '<Base_DN_of_ldap_server>'


# Optional, LDAP search filter to search groups on LDAP.

# Defaults to '(objectClass=posixGroup)' LDAP_GROUP_SEARCH_FILTER:
'<ldap_group_search_filter>'


# Optional, LDAP search filter to search group-members on LDAP.

# Defaults to '(objectClass=posixAccount)' LDAP_GROUP_USER_SEARCH_FILTER:
'<ldap_group_user_search_filter>'


# TLS certificate path.

# Required when LDAP_SECURE is True for self-signed certificates only.

# In case of trusted Root-CA-signed certificate, this key in not required. LDAP_CERT_PATH:
'<abs_location_for_cert_path>'


# Mandatory, LDAP attribute which can be used as user-id

# E.g., '<'uid' or 'cn' or 'mail'>' LDAP_USER_ID_ATTRIBUTE: <ldap_user_id_attribute>


# Optional, complete DN of the admin user for bind and search.

# E.g., '<dn=admin, dc=example, dc=com>'

LDAP_ADMIN: '<user-DN for admin>'


# Optional, password of the bind admin user: LDAP_ADMIN LDAP_ADMIN_PASSWORD: '<password>'


# Optional, attribute on the LDAP server used to define group members. LDAP_GROUP_MEMBER: '<
memberUid >'


# Optional, attribute on the LDAP server used to define role associated members.
LDAP_ROLE_MEMBER: '< member >'


# Optional, valid options are True and False (default)
```

© 2026 HCLTech. All rights reserved.

```
# Enable this to use Insight permissions mapping keys with LDAP group/role. LDAP_AUTHORIZATION:
False

# Following are the Optional keys to map Insight permissions with LDAP group/role
# These Group/Role-DN value and LDAP_BASE_DN are used to create full
Group/Role-DN.
# UM_ADMIN_GROUP: 'cn=hvim-um-admin,ou=Roles'
# LDAP_BASE_DN: 'dc=cisco,dc=com'
# Full-DN mapped as UM-Admins:- UM_ADMIN_GROUP + LDAP_BASE_DN

# Optional, LDAP Group-DN to be mapped as Insight UM-
Admins.
# Defaults to 'cn=hvim-um-admin,ou=Roles'
UM_ADMIN_GROUP: <um_admin_group>

# Optional, LDAP Group-DN to be mapped as Insight Pod-
Admins
# Defaults to 'cn=hvim-pod-admin,ou=Roles'
POD_ADMIN_GROUP: <pod_admin_group>

# Optional, LDAP Group-DN to be mapped as Insight Pod-Users[Write and Read
Permissions]
# Defaults to 'cn=hvim-pod-user,ou=Roles'
POD_USER_GROUP: <pod_user_group>

# Optional, LDAP Group to be mapped as Insight Pod-Users[Read
Only]
# Defaults to 'cn=hvim-read-only,ou=Roles'
READ_ONLY_POD_USER_GROUP: <read_only_pod_user_group>
```

⚠ The group-DN values must exclude LDAP_BASE_DN value. For example: *UM_ADMIN_GROUP: cn=hvim-u admin,ou=Group*

UM Without SMTP

Installing UM Without SMTP

By default, a SMTP infrastructure is required for the VIM UM service.

For releases starting from VIM 2.4.2, the UM service is supported in the absence of SMTP.

⚠ The migration of the UM service to SMTP-enabled mode from the mode which does not require SMTP is not supported.

© 2026 HCLTech. All rights reserved.

To install UM without SMTP, follow the below steps:

Modify the *insight_setup_data.yaml* file and add the following key:

```
SMTP_MODE: False
```

Remove the following keys from the *insight_setup_data.yaml*:

```
INSIGHT_SMTP_SERVER INSIGHT_EMAIL_ALIAS INSIGHT_SMTP_PORT INSIGHT_EMAIL_ALIAS_PASSWORD
```

```
# cd Insight-<tag-id>
# ./insight/insight_runner.py -p 1,2 -f <path_to_setup_data>
```

Save the *insight_setup_data.yaml* file and begin the installation.

With SMTP disabled, insight runner sets both the Super Admin and Pod Admin as the default user. The default user can login and register the pod, but cannot perform the following:

- Add a new user at pod level.
- Add a new Pod Admin.
- Add a new Super Admin.

To add new user or update password for the existing user for Insight without SMTP, use the below script:

```
# cd Insight-<tag-id>/insight
# ./user_populate.py --help
usage: user_populate.py [-h] [--username USERNAME] [--emailid EMAILID]
                        [--usertype USERTYPE] [--updatepass UPDATEPASS]

Add new user or update password for existing user for Insight without SMTP.

optional arguments:
  -h, --help            show this help message and exit
  --username USERNAME, -u USERNAME
                        Name of the user.
  --emailid EMAILID, -e EMAILID
                        Email ID of the user.
  --usertype USERTYPE, -t USERTYPE
                        User Type:
                        super_admin - User is Super User for Insight
                        pod_admin - User allowed to register new pods
                        pod_user - User can only get associated with pods
  --updatepass UPDATEPASS, -p UPDATEPASS
                        Email ID of the user whose password needs to be updated.
```

following is an example to add a user:

- **-t** can take one of the following values such as *super_admin*, pod_admin, and *pod_user* as an argument.
- If the user already exists, an error stating "User already exists" is displayed. If the user is new, the script prompts to enter a new password and confirmation password.

To use forgot password functionality, use the below command:

```
#./user_populate.py -p abc@abc.com
```

If you add a user or change a password using *-p* option, you are redirected to the **Change Password** page on the first login through UM.

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

UM Without Internet Access

Installing UM without Internet Access

- Overview
- Installation

Overview

Complete the following steps to install VIM Insight on the NFVI management node. As security is paramount to pod management, the web- service hosting the single pane of glass is protected through TLS. Following are the steps to get the TLS certificate setup going.

You can select one of the following approaches for the TLS certificate configurations:

- Provide your own certificate: You can bring in your certificate on the management node and provide the absolute path of .pem and CA certificate files in the *insight_setup_data.yaml* file. The path must be provided as a value for the key PEM_PATH in the *insight_setup_data.yaml* file.
- Generate a new certificate on the node. You can create a new certificate on the node by running the following command:

```
# cd /root/Insight-<tag_id>/insight/

#./tls_insight_cert_gen.py -f <path_to_insight_setup_data.yaml>/insight_setup_data.yaml.
```

This script searches for the PEM_PATH inside the *insight_setup_data.yaml*. As the path is not provided, it creates a new certificate inside *install-dir*

/openstack-configs.

The self-signed certificate generation utility script is provided for lab/testing deployment only. Ensure that you do not use a self-signed certificate generated by this utility for the production deployment.

The UM node is installed in the air-gapped mode when the data center does not have internet connectivity. You can use the USB drive to load the installation files on the UM node. Downloading them to the USB drive may take several hours depending on the speed of your internet connection.

⚠ Only Insight Installation on a Standalone UM node is supported. Attempt to install Insight on vim management node causes unexpected behavior.

Installation

The procedure to bootstrap the node hosting Insight is the same as installing the buildnode.iso. Ensure that you plan for a standalone UM node for production. Click **Yes** if the node is to be used in the production.

Before you begin

- Ensure that you have set up a CentOS 7 staging server (VM, laptop, or UCS server) with a 64 GB USB 2.0 drive.
- Ensure that you have internet, preferably a wired connection, to download the vim installation files on the USB drive.
- Ensure that you have disabled the CentOS sleep mode.

© 2026 HCLTech. All rights reserved.

On the staging server, use yum to install the following packages:

- PyYAML
- python-requests
- Python 3.6 (rh-python36))

Check if python 3.6 binary is located at */opt/rh/rh-python36/root/bin/*, if not copy the python 3.6 binary to */opt/rh/rh-python36/root/bin/*.

Access the VIM software download website using a web browser.

Log in with the credentials provided by your account representative and download the *getartifacts.py* script from the external registry.

```
# download the new getartifacts.py file
curl -o getartifacts.py -u '<username>:<password>' https://vim-registry.com/mercury-releases/cvim34- rhel7-osp13/releases/<releaseid>/getartifacts.py
# Change the permission of getartifacts.py
chmod +x getartifacts.py
```

Run the *getartifacts.py* script. The script formats the USB 2.0 drive (or USB 3.0 drive for M5-based management node) and downloads the installation files. You must provide the registry username and password, tag ID, and USB partition on the staging server.

```
./getartifacts.py -h
usage: getartifacts.py [-h] -t TAG -u USERNAME -p PASSWORD [--proxy PROXY]
                       [--retry] (-d DRIVE | -f FILE)
                       [--mgmtk8s | --argus | --insight | --sds | -U]

Script to pull container images en masse.

optional arguments:
  -h, --help            show this help message and exit
  -t TAG, --tag TAG     Installer version to pull
  -u USERNAME, --username USERNAME
                        Registry username
  -p PASSWORD, --password PASSWORD
                        Registry password
  --proxy PROXY         https_proxy if needed
  --retry               Try to complete a previous fetch
  -d DRIVE, --drive DRIVE
                         Provide usb drive path
  -f FILE, --file FILE  location of image file
  --mgmtk8s             Additionally download hvim MON HA artifacts
  --argus               Additionally download argus artifacts
  --insight             Additionally download insight artifacts
  --sds                 Additionally download sds artifacts
  -U, --upgrade         Additionally download artifacts for upgrade from 2.4.x

This script pulls images from remote registry then copies the contents to USB
drive or image file based on the user supplied option
```

The *getartifacts.py* script gets the images from the remote registry and copies the contents to the USB drive.

To identify the USB drive, execute the *lsblk* command before and after inserting the USB drive. The command displays a list of available block devices. You can use the output data to find the location of the USB drive. You must provide the entire drive path in the –doption instead of any partition.
For example:

© 2026 HCLTech. All rights reserved.

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -d </dev/sdc> --insight [--proxy proxy. example.com]
```

Verify the integrity of the downloaded artifacts and container images.

create a directory sudo mkdir -p /mnt/Cisco
/dev/sdc is the USB drive, same as supplied in getartifacts.py python script sudo mount /dev/sdc1 /mnt/Cisco
cd /mnt/Cisco
execute the test-usb help to look at the options
./test-usb -h

usage: ./test-usb [-h] -- Show this program to check integrity of artifacts in this USB drive/image file
[-a] -- Check integrity of all (core and all) artifacts in this USB drive/image File
[-l] -- Location of artifacts
[-f] -- Location of image file
[-U] -- test upgrade artifacts from 2.4.x to 3.4.y
execute the verification script
./test-usb
failures will be explicitly displayed on screen, sample success output below
sample output of ./test-usb execution with 3.4.6 release
#./test-usb
INFO: Checking the integrity of this USB drive
INFO: Checking artifact buildnode-K9.iso
INFO: Checking artifact registry-3.0.0.tar.gz
INFO: Checking the integrity of this USB drive
INFO: Checking artifact buildnode-K9.iso
INFO: Checking artifact registry-3.0.0.tar.gz
INFO: Checking artifact haproxy-K9.tar.gz

Node
INFO: Checking required layers:
INFO: 548 layer files passed checksum.

If a failure occurs, an error message is displayed. For example:
./test-usb
INFO: Checking the integrity of this USB drive INFO: Checking artifact buildnode-K9.iso
ERROR: Checksum for artifact buildnode-K9.iso does not match ('SHA512 (buildnode-K9.iso) = 96ec62a0932a0d69daf60acc6b8af2dc4e5eca132cd3781fc17a494592feb52a7f171eda25e59c0d326fbb09194eeda66036cbdc3 870dafe74f59cf1f2dce225'
!= 'SHA512 (buildnode-K9.iso) = a6a9e79fa08254e720a80868555679baeea2dd8f26a0360ad47540eda831617bea0514a117b12ee5f36415b7540afa112a1c904cd 69e40d704a8f25d78867acf')
INFO: Checking artifact registry-3.4.2.tar.gz
ERROR: Artifact registry-3.4.2.tar.gz is not present
INFO: Checking required layers:
ERROR: Layer file sha256:002aa1f0fbdaea7ea25da1d906e732fe9a9b7458d45f8ef7216d1b4314e05207 has a bad checksum
ERROR: Layer file sha256:5be3293a81773938cdb18f7174bf595fe7323fdc018c715914ad41434d995799 has a bad checksum
ERROR: Layer file sha256:8009d9e798d9acea2d5a3005be39bcbfe77b9a928e8d6c84374768ed19c97059 has a bad checksum
Virtualized Infrastructure Manager Installation Guide, 3.4.x 97
Preparing for NFVI Installation
Installing VIM Software Hub in Air-Gapped Mode

© 2026 HCLTech. All rights reserved.

ERROR: Layer file sha256:ea55b2fc29b95d835d16d7eeac42fa82f17e985161ca94a0f61846defff1a9c8 has a bad checksum
INFO: 544 layer files passed checksum.

To resolve failure in downloading artifacts, unmount the USB and run the *getartifacts* command again with the *--retry* option.

```
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> -d </dev/sdc> --insight –retry
```

1. Mount the USB and then run the *test-usb* command to validate if all the files are downloaded:

```
# /dev/sdc is the USB drive, same as supplied in getartifacts.py python script sudo mount /dev/sda1 /mnt/Cisco

cd /mnt/Cisco
```

2. Execute the verification script:

```
# ./test-usb

# In case of failures the out of the command displays a message indicating the same on the screen
```

3. When the USB integrity test completes, unmount the USB using the following command:

```
sudo umount /mnt/Cisco
```

4. After the artifacts of a target release are saved on the USB, you must unplug the USB from the staging server, connect it to the UM node, and then perform the following steps on the UM node:

a. Provision your UM node with the build node ISO of that release and then connect the USB to UM node.
b. To copy the contents of the USB to UM node navigate to the */root/insight-<tag>* directory and then execute the import artifacts command.

```
# cd ~/insight-<tag>/tools # ./import_artifacts.sh
```

> ⚠ To update UM, do not proceed beyond Step 11 and continue with *Step 5* of Update/Upgrade UM

5. To create *insight_setup_data.yaml*, see *Step 2 and Step 3 of Installation* under UM with Internet Access
6. Save the edited *insight_setup_data.yaml* file.

© 2026 HCLTech. All rights reserved.

```
$ cd /root/insight-<tag_ctr>/
$ ./insight/insight_runner.py -h
usage: insight_runner.py [-h] [-f SETUPFILE] [-y] [-s SKIP_STEPS]
                         [-p PERFORM_STEPS] [-l] [-v]
                         [--update] [--dry-run] [--commit] [--rollback] [--reconfigure] [--
                         regenerate_secrets]
                         [--setpassword] [-i] [-u] [--list-reconfigure-keys]

INSIGHT ORCHESTRATOR

optional arguments:
  -h, --help            show this help message and exit
  -f SETUPFILE, --setupfile SETUPFILE
                        User input file, default is insight_setup_data.yaml
  -y, --yes             Yes option to skip steps without prompt
  -s SKIP_STEPS, --skip_steps SKIP_STEPS
                        Comma separated list of steps to skip. eg -s 1
  -p PERFORM_STEPS, --perform_steps PERFORM_STEPS
                        Comma separated list of steps to perform. eg -p 2
  -l, --list_steps      List steps
  -v, --version         Display version info
  --update              Update Insight
  --dry-run             Dry Run to find all containers marked for update
  --commit              Commit Insight Update
  --rollback            Rollback Insight Update
  --reconfigure         Reconfigure Insight DB, TLS, and UI
  --regenerate_secrets  System generated Insight DB_ROOT_PASSWORD
  --setpassword         User supplied Insight DB_ROOT_PASSWORD
  -i, --install-status  Display Insight install status
  -u, --update-status   Display Insight update status
  --list-reconfigure-keys
                        List setup data reconfigurable keys

$ ./insight/insight_runner.py -l

                    !! INSIGHT !!
          VIM UNIFIED MANAGEMENT ORCHESTRATOR
  ----------------------------------
```

14. Start the insight installation process:

Operations	Operation ID
INPUT_VALIDATION	1
BOOTSTRAP_INFRA	2

© 2026 HCLTech. All rights reserved.

```
$ ./insight/insight_runner.py -f </root/insight_setup_data.yaml> Perform steps ['1', '2']. Continue (Y/N)y

The logs for this run are available at /var/log/insight/bootstrap/<date>_<time>


############################## VIM INSIGHT ORCHESTRATOR
##############################


[1/2][INPUT_VALIDATION: INIT]        [          \          ]        0min 1sec



Management node validation!
+-------------------------------+--------+ ------------------------------------------ +

| Rule | Status | Error|
+-------------------------------+--------+ ------------------------------------------ +
|Check Kernel Version           | PASS| None|
|Check Ansible Version          | PASS| None|
|Check Docker Version           | PASS| None|
|Check Management Node Tag      | PASS| None|
|Check Bond Intf. Settings      | PASS| None|
|Root Password Check            | PASS| None|
|Check Boot Partition Settings  | PASS| None|
|Check LV Swap Settings         | PASS| None|
|Check Docker Pool Settings     | PASS| None|
|Check Home Dir Partition       | PASS| None|
|Check Root Dir Partition       | PASS| None|
|Check /var Partition           | PASS| None|
|Check LVM partition            | PASS| None|
|Check RHEL Pkgs Install State  | PASS| None|
+-------------------------------+--------+ ------------------------------------------ +



Insight standalone input validation!
+-----------------------------------------+--------+ ------------------------------------------+

| Rule | Status | Error |
+-----------------------------------------+--------+ ------------------------------------------+

| Insight standalone Schema Validation| PASS   | None  |

| Valid Key Check in Insight Setup Data | PASS   | None  |

| Duplicate Key Check In Insight Setup Data | PASS        | None  |

| VIM/Insight Workspace Conflict Check                    | Check LDAP Connectivity

| Check Registry Connectivity
```



© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

```
  | Test Email Server for Insight   | PASS   | None  |
+-----------------------------------------+--------+-------------------------------------------+
.2/2][BOOTSTRAP_INFRA: VIM-admins-Restart and enable sss daemon]
[ DONE! ]          1min 23secs

VIM Insight Installed successfully!
+------------------+--------+
----------------------------------------------------------------------------------------------------------------------------------
+

  | Description   | Status | Details|
+------------------+--------+
----------------------------------------------------------------------------------------------------------------------------------
+

  | Insight UI URL          | PASS   | https://<br_api>:9000   |
  | UI Admin Email ID | PASS      | Check for info @: /root/insight-openstack-        |
  |       |          | configs/insight_setup_data.yaml          |
  | UI Admin Password | PASS    | Check for info @: /opt/cisco/insight/secrets.yaml          |
+------------------+--------+
----------------------------------------------------------------------------------------------------------------------------------
+

Backup Validation Passed
[***][AUTOBACKUP: Completed Step 1 of backup, backupdir: /var/cisco/insight_backup
/insight_autobackup_3.4.6_<date>_<time>]
[***][AUTOBACKUP: Starting post-backup please
wait!!]
[***][AUTOBACKUP: Completed VIM UM backup ...
/var/cisco/insight_backup/insight_autobackup_3.4.6_2020-
<date>_<time>

Ended Installation [BOOTSTRAP_INFRA] [Success]

The logs for this run are available at /var/log/insight/bootstrap/<date>_<time>

Logs of Insight Bootstrap are generated at : /var/log/insight/bootstrap/ on the UM management
node. Log file name for Insight Bootstrap are in the following tar format:
insight_<date>_<time> tar.gz
```

Once the bootstrap process is completed a summary table preceding provides the information of the UI URL and the corresponding login credentials. After first login, for security reasons, we recommend you change the Password.

Insight autobackup takes place after the installation, and is located at default backup location /var /cisco/insight_backup; details of which is provided below the bootstrap summary table.

To add a new UI Admin in a setup that just got created, login to VIM insight and add a new UI admin user from the Manage UI Admin Users menu. Without doing a fresh install (that is un-bootstrap, followed by bootstrap) of the insight application, the UI admin that was bootstrapped cannot be changed.

© 2026 HCLTech. All rights reserved.

Refer VIM Insight Post Bootstrap Validation Checks section, to verify the bootstrap status of VIM Insight

UM Optional Services

UM Optional Services

For release CVIM 3.2.0 or later, CVIM UM service provides the following optional features:

- Automatically add each UM admin as the default pod user with *Full-Pod-Access* to a pod during pod registration. Display all the pod users as suggested users, while registering a new pod user.

> ⚠ By default, the optional features are not enabled. To enable these optional features, change the value of corresponding keys to *True* in Insight setup data file.

To install UM with these optional features, follow the below steps:

Modify the *insight_setup_data.yaml* file and add following key:

a. To automatically add each UM admin to pod with *Full-Pod-Access* during pod registration, use the below command:

```
UM_ADMIN_WITH_FULL_POD_ACCESS: True
```

b. To display the suggested users during pod user registration, use the following command:

```
DISPLAY_ALL_POD_USERS: True
```

1. Save the yaml file and begin the installation from the insight directory:

```
# cd /root/Insight-<tag_id>/

# ./insight/insight_runner.py -f </root/insight_setup_data.yaml>
```

UM Post Bootstrap Validation Checks

UM Post Bootstrap Validation Checks

Follow the below steps to check the VIM Insight bootstrap validation:

Check Insight installation workspace:

```
# cd $HOME # ll
Look for path pointed by symlink 'insight-openstack-configs'
# lrwxrwxrwx.  1 root root 38 May 20 23:59 insight-openstack-configs -> /root/insight-<tag_id>
/openstack-configs
```

After the VIM Insight bootstrap, you can view the status of Insight installation through *install-status* action using bootstrap:

© 2026 HCLTech. All rights reserved.

```
# ./insight/insight_runner.py --install-status
Fetching Insight install status.

vim Insight Install Status!+--------------------+--------
+ ------------------------------------------- +
| Description        | Status | Details                                          |
+--------------------+--------+--------------------------------------------------+
| Insight Setup      | PASS   | Success                                          |
| Insight Version    | PASS   | NG: <release_tag>                                |
| Insight UI URL     | PASS   | https://<br_api>:9000                            |
| Mariadb Container  | PASS   | mariadb_<tag_id>                                 |
| Insight Container  | PASS   | insight_<tag_id>                                 |
| Insight Autobackup | PASS   | [ACTIVE]: Running 'insight-autobackup.service'   |
+--------------------+--------+--------------------------------------------------+
```

You can also verify whether the Insight and MySQL containers are up or not, by running the following command:

```
$ source ~/.bashrc

$ dp
```

4. Check the status of Insight, by running the following command:

```
$ systemctl status docker-insight
```

```
docker-insight.service - Insight Docker Service
Loaded: loaded (/usr/lib/systemd/system/docker-insight.service; enabled; vendor preset disabled)
Active: active (running) since Fri 2017-04-07 13:09:25 PDT; 36s ago Main PID: 30768 (docker-current)
Memory: 15.2M
CGroup: /system.slice/docker-insight.service
30768 /usr/bin/docker-current start -a insight_ng_<tag-id>

Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:Tables_in_rbac
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:buildnode_master
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:permission_master
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:role_master
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:role_permission
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:user_master
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:user_role
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:user_session
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:Starting the apache httpd
Apr 07 13:09:26 i11-tb2-ins-3 docker[30768]:AH00558: httpd: Could not reliably
                                            determine

the server's fully qualified domain name, using 2.2.2.6. Set the 'ServerName' directive gl... this message
Hint: Some lines were ellipsized, use -l to show in full.
```

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

5. Check if the Insight is up, by running the following command:

```
$ curl https://br_api:9000 -k (or --insecure)


Your response of curl should show the DOCTYPE HTML:
<!DOCTYPE html>
<!--[if lt IE 7]> <html lang="en" ng-app="myApp" class="no-js lt-ie9 lt-ie8 lt-ie7">
<![endif]-->
<!--[if IE 7]> <html lang="en" ng-app="myApp" class="no-js lt-ie9 lt-ie8">
<![endif]-->
<!--[if IE 8]> <html lang="en" ng-app="myApp" class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html lang="en" ng-app="mercuryInstaller" class="no-js">
<!--<![endif]-->
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>VIM Installer</title>
<meta name="description" content="">
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=0"/>
<link rel="stylesheet"
href="../static/lib/html5-boilerplate/dist/css/normalize.css">
<link rel="stylesheet" href="../static/lib/html5-boilerplate/dist/css/main.css">
<link rel="stylesheet" href="../static/lib/bootstrap/bootstrap.min.css">
<link rel="stylesheet" href="../static/lib/font-awesome/font-awesome.min.css">
<!--<link
href="http://maxcdn.bootstrapcdn.com/font-awesome/4.1.0/css/font-awesome.min.css" rel="stylesheet">-->
<link rel="stylesheet" href="../static/lib/bootstrap/bootstrap-theme.min.css">
```

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

```
<link rel="stylesheet" href="../static/lib/uigrid/ui-grid.min.css">
<link rel="stylesheet" href="../static/lib/chart/angular-chart.min.css">
<script
src="../static/lib/html5-boilerplate/dist/js/vendor/modernizr-2.8.3.min.js"></script>
<link rel="stylesheet" href="../static/css/app.css">
<!--new dashboard css starts-->
<link rel="stylesheet" href="../static/css/dashboard.css">
<!--new dashboard css end-->
</head>
<body class="skin-blue sidebar-collapse" ng-controller="DashboardCtrl" id="ToggleNavbar">
<div class="wrapper" id="wrapper">
<div class="content-wrapper" id="contentclass">
<mi-header></mi-header>

<mi-left-side-navbar></mi-left-side-navbar>
<message-box> </message-box>
<div class=" viewheight" ng-view autoscroll="true"></div>
</div>
<mi-footer></mi-footer>
</div>
<!--new dashboard js starts-->
<script src="../static/lib/bootstrap/jquery.min.js"></script>
<script src="../static/lib/jquery/jquery-ui.js"></script>
<script src="../static/lib/bootstrap/progressbar.js"></script>
<!--new dashboard js ends-->
<script src="../static/lib/chart/Chart.min.js"></script>
<script src="../static/lib/bootstrap/bootstrap.min.js"></script>
<script src="../static/lib/angular/angular.js"></script>
<script src="../static/lib/chart/angular-chart.min.js"></script>
```

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

```
<script src="../static/lib/uigrid/angular-touch.js"></script>
<script src="../static/lib/uigrid/angular-animate.js"></script>
<script src="../static/lib/uigrid/csv.js"></script>
<script src="../static/lib/uigrid/pdfmake.js"></script>
<script src="../static/lib/uigrid/vfs_fonts.js"></script>
<script src="../static/lib/uigrid/ui-grid.js"></script>
<script src="../static/lib/angular/smart-table.min.js"></script>
<script src="../static/lib/angular-route/angular-route.js"></script>
<script src="../static/lib/angular-cookies/angular-cookies.js"></script>
<script src="../static/lib/angular/angular-translate.js"></script>
<script
src="../static/lib/angular/angular-translate-loader-static-files.min.js"></script>
<script
src="../static/lib/angular/angular-translate-storage-cookie.min.js"></script>
<script
src="../static/lib/angular/angular-translate-storage-local.min.js"></script>
<script src="../static/lib/yamltojson/yaml.js"></script>
<script src="../static/lib/yaml/js-yaml.min.js"></script>
<script src="../static/lib/d3/d3min.js"></script>
<script src="../static/utility/utility.js"></script>
<script src="../static/widgets/widgets.js"></script>
<script src="../static/app.js"></script>
<script src="../static/layout/layout.js"></script>
<script src="../static/login/login.js"></script>
<script src="../static/globals/globals.js"></script>
<script src="../static/dashboard/dashboard.js"></script>
<script src="../static/cloudpulse/cloudpulse.js"></script>
<script src="../static/blueprintsetup/physicalsetupwizard/ucsmcommon.js"></script>
<script src="../static/blueprintsetup/physicalsetupwizard/cimccommon.js"></script>

<script src="../static/vmtp/runvmtp.js"></script>
<script src="../static/blueprintsetup/physicalsetupwizard/networking.js"></script>
<script src="../static/blueprintsetup/physicalsetupwizard/serverandroles.js"></script>
<script src="../static/blueprintsetup/openstacksetupwizard/cephsetup.js"></script>
<script src="../static/blueprintsetup/openstacksetupwizard/cindersetup.js"></script>
<script src="../static/blueprintsetup/openstacksetupwizard/glancesetup.js"></script>
```

© 2026 HCLTech. All rights reserved.

<script src="../static/blueprintsetup/openstacksetupwizard/haproxy.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/keystonesetup.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/swiftstack.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/neutronsetup.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/vmtpsetup.js"></script>

<script src="../static/blueprintsetup/physicalsetupwizard/physicalsetupwizard.js"></script>

<script src="../static/blueprintsetup/servicesSetupWizard/systemlog.js"></script>

<script src="../static/blueprintsetup/servicesSetupWizard/NFVbench.js"></script>

<script src="../static/blueprintsetup/servicesSetupWizard/servicesSetupWizard.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/openstacksetupwizard.js"></script>

<script src="../static/blueprintsetup/blueprintsetup.js"></script>

<script src="../static/blueprintmanagement/blueprintmanagement.js"></script>

<script src="../static/topology/topology.js"></script>

<script src="../static/monitoring/monitoring.js"></script>

<script src="../static/horizon/horizon.js"></script>

<script src="../static/podmanagement/podmanagement.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/tlssupport.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/elksetup.js"></script>

<script src="../static/systemupdate/systemupdate.js"></script>

<script src="../static/blueprintsetup/physicalsetupwizard/registrysetup.js"></script>

<script src="../static/registertestbed/registertestbed.js"></script>

<script src="../static/registersaas/registersaas.js"></script>

<script src="../static/useradministration/manageusers.js"></script>

<script src="../static/useradministration/rolemanagement.js"></script>

<script src="../static/saasadmindashboard/saasadmindashboard.js"></script>

<script src="../static/saasadmindashboard/buildnodes.js"></script>

<script src="../static/saasadmindashboard/buildnodeusers.js"></script>

<script src="../static/saasadmindashboard/managesaasuser.js"></script>

<script src="../static/saasadminusermanagement/saasadminusermgmt.js"></script>

<script src="../static/blueprintsetup/physicalsetupwizard/nfvisetup.js"></script>

<script src="../static/blueprintsetup/physicalsetupwizard/torswitch.js"></script>

<script src="../static/blueprintsetup/openstacksetupwizard/vtssetup.js"></script>

<script src="../static/rbacutilities/rbacutility.js"></script>

<script src="../static/forgotpassword/forgotpassword.js"></script>

<script src="../static/changepassword/changepassword.js"></script>

<script src="../static/passwordreconfigure/passwordreconfigure.js"></script>

<script src="../static/openstackconfigreconfigure/openstackconfigreconfigure.js"></script>

© 2026 HCLTech. All rights reserved.

<script src="../static/reconfigureoptionalservices/reconfigureoptionalservices.js"></script>

</body>

You can check the status of Insight autobackup service that is invoked as a daemon process. Incremental autobackup is taken from the database and */opt/cisco/insight/mgmt_certs* directory if there is any change. In other words, it triggers a backup whenever there is a change in Insight DB resulting from UI and saves it in the latest backupdir located at /var/cisco/insight_backup

systemctl status insight-autobackup

insight-autobackup.service - Insight Autobackup Service
Loaded: loaded (/usr/lib/systemd/system/insight-autobackup.service; enabled; vendor preset disabled)
Active: active (running) since Mon 2017-09-04 05:53:22 PDT; 19h ago
Process: 21246 ExecStop=/bin/kill ${MAINPID} (code=exited, status=0/SUCCESS) Main PID: 21287 (python)
Memory: 9.2M
CGroup: /system.slice/insight-autobackup.service 21287 /usr/bin/python
/var/cisco/insight_backup/insight_backup_2.1.10_2017-08-31_03:02:06/root
/rohan/installer-10416/insight/playbooks/../insight_autobackup.py
Sep 04 05:53:22 F23-insight-4 systemd[1]: Started Insight Autobackup Service.
Sep 04 05:53:22 F23-insight-4 systemd[1]: Starting Insight Autobackup Service...

UM Admin Login for Standalone Setup

UM Admin Login for Standalone Setup

For security reasons, the Insight Admin logs into the UI with which UM is bootstrapped. Insight Admin needs to add new users as Pod Admin.

Registration of UM Admin to UM

- Enter the following address on the browser: https://<br_api>:9000.
- Enter the **Email ID** and **Password**. The Email ID should be the one specified as UI_ADMIN_EMAIL_ID in *insight_setup_data.yaml* during bootstrap. The Password for UI Admins are generated at:

```
/opt/cisco/insight/secrets.yaml
```

where key is UI_ADMIN_PASSWORD.

- If LDAP mode is True and LDAP user attribute is set to uid, login with LDAP user id credentials.
- Click **Login as UI Admin User**. You will be redirected to Insight UI Admin Dashboard.

UM Pod Admin Login for Standalone Setup

UM Pod Admin Login for Standalone Setup

Follow the below steps:

- Log in as Insight UM.
- Navigate to **Manage Pod Admin** and click **Add Pod Admin**.
- Enter a new Email ID in **Add Pod Admin** pop-up.
- Enter the username of the Pod Admin.
- Click **Save**. User Registration mail is sent to a newly added Pod Admin with a token.
- Click the URL with token and if token is valid then Pod Admin is redirected to Insight-Update Password page.
- Enter new password and then confirm the same password.
- Click **Submit**.

© 2026 HCLTech. All rights reserved.

UM Host Administrators

Managing UM Host Administrators

- Additional VIM UM Administrators
- Enabling UM Administrator with LDAP Authentication
- Enabling Root Login
- Disabling Root Login
- Enabling Banner During SSH Login

Additional VIM UM Administrators

VIM UM supports the management of the VIM administrators. The administrator can log into the Unified Management node through SSH or the console using the configured password. By configuring one VIM admin account, administrators do not have to share credentials. Administrators have individual accountability. This can be included as a part of a fresh installation or can be brought in via reconfiguration.

To enable one or more VIM administrators, perform the following steps:

Fetch the latest insight_setup_data.yaml file:

```
# cd /root/
# mkdir MyDir
# cp /root/insight-openstack-configs/insight_setup_data.yaml /root/MyDir/
```

© 2026 HCLTech. All rights reserved.

2. Modify the insight_setup_data.yaml file manually with the configurations listed as below:

```
# each vim admin must have a vim_admin_password_hash, a vim_admin_public_key,
# or both.

## 1. vim_admin_username: Admin Username                                      |
##-------------------------------------------------------------------------
## vim admin usernames should satisfy following criteria:
##     a. required
##     b. unique
##     c. ASCII chars
##     d. no space allowed
##     e. 1 <= length <=32
##     f. Only lower case letters
##     g. digits, underscores, or dashes
##     h. first character must be a letter or an underscore


## 2. vim_admin_password_hash: Admin Unix Passwd Hash                         |
##-------------------------------------------------------------------------
## Optional, to generate a password hash:
## -> python -c 'import crypt; print crypt.crypt ("<plaintext_strong_pwd>")'


## 3. vim_admin_public_key: Admin Unix Pubkey                                 |
##-------------------------------------------------------------------------
## Optional, vim_admin_public_key is a user's public key.
## It can be generated with 'ssh-keygen'.
## Should be a string that starts with "ssh-rsa AAAA" or "ssh-ed25519 AAAA"


# Optional, must have at least one sub key defined.
vim_admins:
     # vim admin with only password hash.
               1. vim_admin_username:
                  non_root_admin_1
                  vim_admin_password_hash
                  : $6$.....

     # vim admin with password hash and public key
               2. vim_admin_username:
                  non_root_admin_2
                  vim_admin_password_hash
                  : $6$.....
     vim_admin_public_key: ssh-rsa AAAA... or ssh-ed25519 AAAA...
```

3. Run the following command for installation:

```
# For installation

# ./insight/insight_runner.py -f <path_to insight_setup_data.yaml>
```

4. Run the following reconfiguration command:

```
# For reconfigure

# ./insight/insight_runner.py --reconfigure -f <path_to insight_setup_data.yaml>
```

> ⚠ VIM administrators can manage their own passwords using the Linux *passwd* command. You can add or remove VIM administrator through the reconfigure option, while the passwords for their existing accounts remain unchanged.

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

Enabling UM Administrator with LDAP Authentication

VIM UM supports the management of the UM administrators whose access to the management node can be authenticated through an external LDAP server (Open LDAP or Microsoft Active Directory (AD) based). Since Unified Management node is not a general-purpose Linux box, all users accessing the management node have the same administrative privileges, that is, the enforcement is limited to user authentication and not authorization. It can be added as a Day 0 or Day 1 activity. Ensure that the *ldap_uris* is secured over LDAPs.

Starting from VIM 4.4.2 LDAP authentication with both secure and non-secure through an external LDAP server (Open LDAP or Microsoft Active Directory (AD) based) is supported.

To obtain sudo access to the UM node, you can choose one of the following options:

- If *sudoers_group_name* is not enabled, then manually add the user with root privileges to the wheel group in the corresponding LDAP domain, for example, *usermode -aG wheel user1*
- Use the optional parameter *sudoers_group_name*, so that all members listed in *sudoers_group_name* gain root privileges. This optional parameter is available only from VIM 3.4.6 onwards.

To enable UM administrators with LDAP authentication over secure (i.e., **ldap_uri : ldaps**:// <>) or non-secure(.i.e., **ldap_uri : ldap://<>**) perform the following steps:

Take a backup of the *setupdata* file and update the file manually with the configuration listed below during installation:

UM_LDAP_ADMINS:

1. domain_name: corp_ldap1

ldap_uri: "ldaps://<ip_address_1:[port_1]>,ldaps://<ip_address_2:[port_2]>" or ldap_uri: "ldap://<ip_address>

ldap_search_base: "dc=cisco,dc=com"

ldap_cert_path: <LDAP Cert Path> # Optional

ldap_user_object_class: posixAccount # Optional

ldap_user_uid_number: uidNumber # Optional

ldap_user_gid_number: gidNumber # Optional

ldap_group_member: memberUid # Optional

- domain_name: corp_ldap2
  ldap_uri: "ldaps:///<ip_address_3:[port_3]"

  ldap_search_base: "dc=cisco,dc=com"

  ldap_cert_path: <LDAP Cert Path> # Optional

  ldap_user_object_class: posixAccount # Optional

  ldap_user_uid_number: uidNumber #Optional

  ldap_user_gid_number: uidNumber #Optional

  ldap_group_member: memberUid #Optional

  ldap_default_bind_dn: "<string>" #Optional

  ldap_default_authtok: "<string>" # Optional

  ldap_default_authtok_type: "<string>" # Optional (password|obfuscated_password) ldap_group_search_base: "<string>" # Optional

© 2026 HCLTech. All rights reserved.

ldap_user_search_base: "<string>" # Optional access_provider: "<string>" #Optional simple_allow_groups: "<string>" # Optional

ldap_id_use_start_tls: <boolean> # Optional

ldap_tls_reqcert: "<string>" #Optional (never|allow|try|demand) chpass_provider: "<string>" # Optional (ldap|krb5|ad|none)

sudoers_group_name: <list> # Optional (list of group that should be able to execute "sudo")

⚠ ldap_default_authtok is mandatory if the LDAP server does not support anonymous bindings.

To support AD (Windows AD) with or without Kerberos, provide the additional parameters as given below:

UM_LDAP_ADMINS:

- access_provider: simple
  ....
  ....

modified attribute values

ldap_schema: rfc2307 # supported possible values 'rfc2307' or 'ad' ldap_user_object_class: person # supported possible values 'posixAccount', 'person'

new attributes (for AD with LDAP or Kerberos)

realm_name: <string> # optional, mandatory if krb5 is the auth_provider id_provider: ldap # optional, default is ldap

auth_provider: krb5 # optional, possible values ldap | krb5, defaults to ldap ldap_user_name: sAMAccountName # mandatory if ldap_schema is 'ad'

kdc_server: '<fqdn>:<port>' # optional # optional, mandatory if auth_provider is krb5 kerberos_ticket_lifetime: <int> # optional, relevant if krb5

is auth_provider (defaults to 24h in

sec) (0 - 2147483647)

kerberos_renew_lifetime: <int> # optional relevant if krb5 is auth_provider (defaults to 7d in sec) (0 - 2147483647)

Multiple entries of the LDAP domain are allowed. For each entry, only *domain_name* and *ldap_uri* info are mandatory. Ensure that the *l dap_uri* is secured over LDAPs. As part of the reconfiguration, you can add a new *domain_name,* but the *domain_name* cannot be changed once configured.

© 2026 HCLTech. All rights reserved.

4. To reconfigure the VIM administrator, run the following commands:

Enabling Root Login

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/insight-openstack-configs/insight_setup_data.yaml /root/MyDir/
# update/include the vim_ldap_admin in the setup_data
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi insight_setup_data.yaml
[root@mgmt1 ~]# cd ~/insight-xxxx
[root@mgmt1 ~]# cd Insight-<tag_id>
[root@mgmt1 ~]# ./insight/insight_runner.py --reconfigure -f <path_to_insight_setup_data.yaml>
```

To complement the management of VIM administrators, VIM supports an option to enable/disable root access at login. By default, this option is set to True. You can optionally disable this root access through reconfiguration or installation.

To enable root login, follow the below steps:

1. Fetch the latest *insight_setup_data.yaml* file:

```
# cd /root/
# mkdir MyDir
# cp /root/insight-openstack-configs/insight_setup_data.yaml /root/MyDir/
```

2. Modify the *insight_setup_data.yaml* file manually with the configuration listed below:

```
# True: root can SSH to the UM management node.
# False: root cannot SSH to the UM management node.
#       At least one vim_admin must be configured if this is False
#       Use su to drop down to root and execute administrator functionalities permit_root_login: False
```

© 2026 HCLTech. All rights reserved.

3. Run the following command for installation:

```
# For Install

# /insight/insight_runner.py -f <path_to_insight_setup_data.yaml>
```

4. Run the following reconfiguration command:

```
# For reconfigure

# /insight/insight_runner.py --reconfigure -f <path_to_insight_setup_data.yaml>
```

Disabling Root Login

You can disable the root login using the below command:

```
# Permit Root Login (optional, default True)


# True: admin can SSH to management node with root userid and password
```

Enabling Banner During SSH Login

VIM UM supports the enabling of a banner during SSH login to the management node. You can enable the banner during fresh installation or via reconfiguration.

To enable banner during login, perform the following steps:

1. Fetch the latest *insight_setup_data.yaml* file:

```
# cd /root/

# mkdir MyDir

# cp /root/insight-openstack-configs/insight_setup_data.yaml /root/MyDir/
```

© 2026 HCLTech. All rights reserved.

2. Modify the insight_setup_data.yaml file manually with the configuration listed below:

```
ssh_banner:
<your Banner Text>
```

3. Run the following command for installation:

```
# For Install
#./insight/insight_runner.py -f <path_to_insight_setup_data.yaml>
```

4. Run the following reconfiguration command:

```
# For reconfigure
#./insight/insight_runner.py --reconfigure -f <path_to_insight_setup_data.yaml>
```

Reinstallation of UM Node

Reinstallation of UM Node

Due to unforeseen circumstances, there might be a need to reinstall the UM application with the same image version. To alleviate the need for re-imaging the UM node, follow the steps listed below to uninstall and reinstall the UM application with the image tag used for install in the first place.

Copy the setup_data.yaml from /root/openstack-configs/ directory to ~/Save/

```
cd insight-<tag-id>/insight
./insight_unbootstrap.sh -k
```

⚠ Ensure that you pass *-k* option during uninstallation, to preserve the artifacts.

To verify that no docker containers are running, use the command:

```
docker ps -a
```

To verify that no docker images are present, use the command:

```
docker images
```

Once the uninstallation is complete, proceed with the installation steps mentioned in the Unified Management.

© 2026 HCLTech. All rights reserved.

Verifying Installation

Verifying Installation

- Displaying IP Addresses
- VIM Client CLI Availability
- Displaying NFVI Logs Accessing OpenStack API
- Endpoints Assessing NFVI Health
- HA Proxy Dashboard/ELK Stack
- Logs Testing Pod/Cloud Infrastructure

Displaying IP Addresses

Displaying IP Addresses

To display the IP addresses for all NFVI nodes, enter the following command:

```
# cd /root/openstack-configs
[root @nfvi_management_node openstack-configs]# cat
/root/installer/openstack-configs/mercury_servers_info
```

Sample output:

Total nodes: 8

Controller nodes: 3

```
+---------------+---------------+------------+------------+-----------+ ------------------------------------------+
| Server       | CIMC        | Management | Provision  | Tenant      | Storage |
+---------------+---------------+------------+------------+-----------+ ------------------------------------------+
| c44-control-1 | 172.26.233.54 | 10.21.1.25 | 10.21.1.25 | 10.2.2.22 | None    |
|                                    |                       |             |             |             |           |
| c44-control-3 | 172.26.233.56  | 10.21.1.27 | 10.21.1.27 | 10.2.2.24 | None   |
|                                  |                 |             |             |             |           |
| c44-control-2 | 172.26.233.55  | 10.21.1.28 | 10.21.1.28 | 10.2.2.25 | None   |
|                                  |                 |             |             |             |           |
+---------------+---------------+------------+------------+-----------+ ------------------------------------------+
```

Compute nodes: 2

```
+---------------+---------------+------------+------------+-----------+ ------------------------------------------+
|      Server          |   CIMC            | Management | Provision  |       Tenant   | Storage
+---------------+---------------+------------+------------+-----------+ ------------------------------------------+
| c44-compute-1 | 172.26.233.57 | 10.21.1.26 | 10.21.1.26 | 10.2.2.23 |  None   |
|                                  |                 |             |             |             |           |
```

© 2026 HCLTech. All rights reserved.

| c44-compute-2 | 172.26.233.58 | 10.21.1.23 | 10.21.1.23 | 10.2.2.21 | None |

Storage nodes: 3

Server	CIMC	Management	Provision	Tenant	Storage
c44-storage-3	172.26.233.53	10.21.1.22	10.21.1.22	None	10.3.3.22
c44-storage-2	172.26.233.52	10.21.1.24	10.21.1.24	None	10.3.3.23
c44-storage-1	172.26.233.51	10.21.1.21	10.21.1.21	None	10.3.3.21

[root@c44-top-mgmt openstack-configs]#

VIM Client CLI Availability

VIM Client CLI Availability

VIM client CLI is used for managing NFVI pods. After the completion of NFVI installation, verify that the VIM user is running and pointing to the right management node in the installer directory. NFVI provides a tool to check the REST API server status and directory where it is running.

To run the tool, enter the following command:

```
#cd installer-<tagid>/tools #./restapi.py -a status
Status of the REST API Server: active (running) since Thu 2016-08-18 09:15:39 UTC; 9h ago REST API launch directory: /root/installer-<tagid
```

Confirm if the server status is active and check whether the REST API directory matches with the directory where the installation is launched. The REST API command also provides options to start, tear down, and reset the REST API server password. Run the following REST API command to reset the password.

```
#./restapi.py -h
usage: restapi.py
[-h] --action ACTION [--yes] [--verbose]

REST API setup helper
optional arguments:
-h, --help    show this help message and exit
--action ACTION, -a ACTION
                                    setup - Install and start the REST API server.
                                    teardown - Stop and uninstall the REST API server.
                                restart - Restart the REST API server.
                            regenerate-password -Regenerate the password for
                                reset-password - Reset the REST API password with the given password.
                                    status - Check the status of the REST API server

--yes, -y           Skip the dialog. Yes, to the action.
--verbose, -v        Perform the action in verbose mode.
```

© 2026 HCLTech. All rights reserved.

If the REST API server is not running, executing **VIM** displays the following error message:

```
# cd installer-<tagid>/
# hvim --setupfile ~/Save/<setup_data.yaml> run
```

4. If the installer directory or the REST API state is not correct or pointing to an incorrect REST API directory, go to the installer-*<tagid>/tools* directory and execute the following command:

```
# ./restapi.py -action setup
```

5. Confirm that the REST API server state and directory are correct using the below command:

```
./restapi.py -action status
```

6. If the REST API recovery step was run on an existing pod, run the following command to ensure that the REST API server continues to manage it:

```
# cd installer-<tagid>/
# hvim --setupfile <setup_data_file_path> run --perform 7 -y
```

⚠ For more information on REST API, see REST API Overview

Displaying NFVI Logs

Displaying NFVI Logs

The NFVI installation logs are generated and available in */var/log/mercury//<install_uuid>/* directory of the management node. The last 20-log directories are tarred and kept in this directory. The logs are archived (tar.gz file) after each run.

The following table lists the NFVI installation steps and corresponding log files:

Step	Description	Log File
1	INPUT_VALIDATION	mercury_baremetal_install.log
2	MGMTNODE_ORCHESTRATION	mercury_buildorchestration.log
3	VALIDATION	mercury_baremetal_install.log
4	BAREMETAL	mercury_baremetal_install.log
5	COMMONSETUP	mercury_os_install.log
6	CEPH	mercury_ceph_install.log
7	ORCHESTRATION	mercury_os_install.log
8	VMTP	None

© 2026 HCLTech. All rights reserved.

© 2026 HCLTech. All rights reserved.

Accessing OpenStack API Endpoints

Accessing OpenStack API Endpoints

The NFVI installer stores the access credentials in */root/installer-<tag-number>/openstack-configs/openrc* of the management node. The external_lb_vip_address provided in the setup_data.yaml is the IP address where OpenStack APIs are handled.

Example:

```
export OS_AUTH_URL=http://172.26.233.139:5000/v3 or

https://172.26.233.139:5000/v3 (if TLS is enabled)

export OS_USERNAME=admin

export OS_PASSWORD=xyzabcd

export OS_TENANT_NAME=admin

export OS_REGION_NAME=RegionOne

# For TLS, add

export OS_CACERT=/root/openstack-configs/haproxy-ca.crt
```

The corresponding setup_data.yaml entry:

```
 ################################################
# HA Proxy ################################################

external_lb_vip_address: 172.26.233.139
```

Assessing NFVI Health

Assessing NFVI Health with CloudPulse

You can use the OpenStack CloudPulse tool to verify NFVI health. CloudPulse servers are installed on all NFVI control nodes, while the CloudPulse users are connected to the management node. Run the following commands to display NFVI information. For information about CloudPulse, visit the OpenStack CloudPulse

© 2026 HCLTech. All rights reserved.

To check the results of periodic CloudPulse runs:

```
# cd /root/openstack-configs
# source openrc
# cloudpulse result
+--------------------------------------+------+--------------------+----------+---------+
| uuid                                 | id   | name               | testtype | state   |
+--------------------------------------+------+--------------------+----------+---------+
| bf7fac70-7e46-4577-b339-b1535b6237e8 | 3788 | glance_endpoint    | periodic | success |
| 1f575ad6-0679-4e5d-bc15-952bade09f19 | 3791 | nova_endpoint      | periodic | success |
| 765083d0-e000-4146-8235-ca106fa89864 | 3794 | neutron_endpoint   | periodic | success |
| c1c8e3ea-29bf-4fa8-91dd-c13a31042114 | 3797 | cinder_endpoint    | periodic | success |
| 04b0cb48-16a3-40d3-aa18-582b8d25e105 | 3800 | keystone_endpoint  | periodic | success |
| db42185f-12d9-47ff-b2f9-4337744bf7e5 | 3803 | glance_endpoint    | periodic | success |
| 90aa9e7c-99ea-4410-8516-1c08beb4144e | 3806 | nova_endpoint      | periodic | success |
| d393a959-c727-4b5e-9893-e229efb88893 | 3809 | neutron_endpoint   | periodic | success |
| 50c31b57-d4e6-4cf1-a461-8228fa7a9be1 | 3812 | cinder_endpoint    | periodic | success |
| d1245146-2683-40da-b0e6-dbf56e5f4379 | 3815 | keystone_endpoint  | periodic | success |
| ce8b9165-5f26-4610-963c-3ff12062a10a | 3818 | glance_endpoint    | periodic | success |
| 6a727168-8d47-4a1d-8aa0-65b942898214 | 3821 | nova_endpoint      | periodic | success |
| 6fbf48ad-d97f-4a41-be39-e04668a328fd | 3824 | neutron_endpoint   | periodic | success |
```

To run a specific CloudPulse test on demand:

```
# cd /root/openstack-configs # source openrc

# cloudpulse run --name <test_name> # cloudpulse run --all-tests

# cloudpulse run --all-endpoint-tests # cloudpulse run --all-operator-tests
```

© 2026 HCLTech. All rights reserved.

```
[root@vms-line2-build installer-3128.21# cloudpulse run --name neutron_endpoint
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| name       | neutron_endpoint                     |
| created_at | 2016-03-29T02:20:16.840581+00:00     |
| updated_at | None                                 |
| state      | scheduled                            |
| result     | NotYetRun                            |
| testtype   | manual                               |
| id         | 3827                                 |
| uuid       | 5cc39fa8-826c-4a91-9514-6c6de050e503 |
+------------+--------------------------------------+
[root@vms-line2-build installer-3128.21#
```

To show detailed results from a specific CloudPulse run:

```
[root@vms-line2-build installer-3128.21# cloudpulse show 5cc39fa8-826c-4a91-9514-6c6de050e503
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| name       | neutron_endpoint                     |
| created_at | 2016-03-29T02:20:16+00:00            |
| updated_at | 2016-03-29T02:20:41+00:00            |
| state      | success                              |
| result     | success                              |
| testtype   | manual                               |
| id         | 3827                                 |
| uuid       | 5cc39fa8-826c-4a91-9514-6c6de050e503 |
```

CloudPulse has two test sets: endpoint_scenario (runs as a cron or manually) and operator test (run manually). Endpoint tests include:

- nova_endpoint
- neutron_endpoint
- keystone_endpoint
- glance_endpoint
- cinder_endpoint

Operator tests include:

- ceph_check
- docker_check
- galera_check
- node_check
- rabbitmq_check

The following table lists the operator tests that you can perform with CloudPulse.

Test	Description
Ceph Check	Executes the ceph -f json status command on the Ceph-mon nodes and parses the output. If the result of the output is not HEALTH_OK, the ceph_check reports an error.
Docker Check	Finds out if all Docker containers are in running state on all nodes and reports an error if any containers are in the exited state. The Docker check runs the command, docker ps -aq --filter 'status=exited'.
Galera Check	Executes the command *mysql SHOW STATUS* on the controller nodes and displays the status.
Node Check	Checks if all the nodes in the system are up and online. It also compares the results of the Nova hypervisor list and determines whether all the compute nodes are available.
RabbitMQ Check	Runs the command, *rabbitmqctl cluster_status* on the controller nodes and finds out if the RabbitMQ cluster is in quorum. If nodes are offline, the rabbitmq_check reports a failure.

© 2026 HCLTech. All rights reserved.

HA Proxy Dashboard/ELK Stack Logs

Displaying HA Proxy Dashboard and ELK Stack Logs

You can view the HA Proxy dashboard at: http://< *external_lb_vip_address* >:1936 using the following login credentials:

- Username—haproxy
- Password—Value for HAPROXY_PASSWORD in **/root/installer-<tag-number>/openstack-configs/secrets.yaml**

You can use the opensearch dashboard to view logs aggregated by Logstash at: *http://< management_node_IP>:5601* using the following login credentials:

- Username—admin
- Password—Value for KIBANA_PASSWORD in **/root/installer-<tag-number>/openstack-configs/secrets.yaml**

Testing Pod/Cloud Infrastructure

Testing NFVI Pod and Cloud Infrastructure

To test the NFVI pod and cloud infrastructure (host connectivity, basic mariadb, rabbit, ceph cluster check, and RAID disks), you can use the cloud-sanity tool available on the management node.

⚠ For details on the execution of cloud-sanity with vim, see Assessing NFVI Status.

UM Without SMTP

Installing UM Without SMTP

By default, a SMTP infrastructure is required for the VIM UM service.
For releases starting from VIM 2.4.2, the UM service is supported in the absence of SMTP.
To install UM without SMTP, follow the below steps:

⚠ The migration of the UM service to SMTP-enabled mode from the mode which does not require SMTP is not supported.

Modify the *insight_setup_data.yaml* file and add the following key:

```
SMTP_MODE: False
```

Remove the following keys from the *insight_setup_data.yaml*:

```
INSIGHT_SMTP_SERVER INSIGHT_EMAIL_ALIAS INSIGHT_SMTP_PORT INSIGHT_EMAIL_ALIAS_PASSWORD
```

© 2026 HCLTech. All rights reserved.

Save the *insight_setup_data.yaml* file and begin the installation.

```
# cd Insight-<tag-id>

# ./insight/insight_runner.py -p 1,2 -f <path_to_setup_data>
```

With SMTP disabled, insight runner sets both the Super Admin and Pod Admin as the default user. The default user can login and register the pod, but cannot perform the following:

- Add a new user at pod level.
- Add a new Pod Admin.
- Add a new Super Admin.

To add new user or update password for the existing user for Insight without SMTP, use the below script:

```
# cd Insight-<tag-id>/insight
# ./user_populate.py --help
usage: user_populate.py [-h] [--username USERNAME] [--emailid EMAILID]
                        [--usertype USERTYPE] [--updatepass UPDATEPASS]

Add new user or update password for existing user for Insight without SMTP.

optional arguments:
  -h, --help            show this help message and exit
  --username USERNAME, -u USERNAME
                        Name of the user.
  --emailid EMAILID, -e EMAILID
                        Email ID of the user.
  --usertype USERTYPE, -t USERTYPE
                        User Type:
                        super_admin - User is Super User for Insight
                        pod_admin - User allowed to register new pods
                        pod_user - User can only get associated with pods
  --updatepass UPDATEPASS, -p UPDATEPASS
                        Email ID of the user whose password needs to be updated.
```

Following is an example to add a user:

```
# cd Insight-<tag-id>/insight

#./user_populate.py -u abc -e abc@abc.com -t pod_user
```

- **-t** can take one of the following values such as *super_admin*, pod_admin, and *pod_user* as an argument.
- If the user already exists, an error stating "User already exists" is displayed. If the user is new, the script prompts to enter a new password and confirmation password.

To use forgot password functionality, use the below command:

```
#./user_populate.py -p abc@abc.com
```

© 2026 HCLTech. All rights reserved.

If you add a user or change a password using *-p* option, you are redirected to the **Change Password** page on the first login through UM.

The specifications and information regarding the products in this manual are subject to change without notice. All statements, information, and recommendations in this manual are believed to be accurate but are presented without warranty of any kind, express or implied. Users must take full responsibility for their application of any products.

The software license and limited warranty for the accompanying product are set forth in the license agreement signed between the parties.

Notwithstanding any other warranty herein, all document files and software of these suppliers are provided “as is” with all

Faults. Hclsoftware and the above-named suppliers disclaim all warranties, expressed or implied, including, without limitation, those of merchantability, fitness for a particular purpose and noninfringement or arising from a course of dealing, usage, or trade practice.

In no event shall hclsoftware or its suppliers be liable for any indirect, special, consequential, or incidental damages, including, without limitation, lost profits or loss or damage to data arising out of the use or inability to use this manual, even if hclsoftware or its suppliers have been advised of the possibility of such damages.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

HCLSOFTWARE Logo is trademark of HCL Technologies Ltd., and/or its affiliates in the U.S. and other countries.

Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between HCLSOFTWARE and any other company.

Copyrights @HCL Technologies Ltd., 2026 All rights reserved.

© 2026 HCLTech. All rights reserved.

Classification: Internal

HCLTech | Supercharging Progress™

hcltech.com