

Cisco Virtualized Infrastructure Manager Documentation, 4.2.3

First Published: 2022-04-21

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

1. CVIM 4.2.3 Home . . . 5
1.1 Release Notes . . . 6
1.1.1 Cisco VIM 4.2.3 Release Notes . . . 7
1.2 Cisco NFVI Architecture . . . 15
1.2.1 Cisco NFVI Overview . . . 16
1.2.2 Cisco VIM Overview . . . 20
1.2.3 Networking Overview . . . 21
1.2.4 UCS C-Series Network Topologies . . . 26
1.2.5 Management Node Networking . . . 33
1.2.6 IPv6 Support . . . 35
1.2.7 UCS C-Series . . . 36
1.2.8 High Availability . . . 38
1.2.9 Storage Node Overview . . . 40
1.2.10 OpenStack Telemetry Service . . . 42
1.2.11 NFVBench . . . 45
1.2.11.1 Setting up NFVbench . . . 46
1.2.11.2 Encapsulation . . . 55
1.2.11.3 Cisco VIM CLI . . . 57
1.2.12 Auto-ToR Configuration via ACI API . . . 58
1.2.13 NCS-5500 as ToR Option . . . 59
1.2.14 Disk Management . . . 60
1.2.15 OSD Maintenance . . . 61
1.2.16 Power Management . . . 62
1.2.17 Physical Cores and Reserved Memory . . . 63
1.2.18 Software Hub . . . 64
1.2.19 VXLAN EVPN Design . . . 65
1.2.20 VPP Port Mirroring Support . . . 68
1.2.21 Segment Routing EVPN . . . 69
1.2.22 Container Workload . . . 73
1.2.23 Management Node Centralization . . . 74
1.2.24 L3 Fabric Deployment . . . 75
1.2.25 OpenStack Barbican . . . 76
1.2.26 Management Network Over TLS . . . 78
1.2.27 Support of Pre-encrypted Image . . . 79
1.2.28 ACI Plugin Integration . . . 80
1.3 Installation . . . 81
1.3.1 Cisco NFVI Installation Overview . . . 82
1.3.2 Installation Preparation Without Internet Access . . . 83
1.3.2.1 Air-gapped Installation Approach . . . 84
1.3.2.2 Prerequisites for Air-Gapped Installation . . . 85
1.3.2.3 NFVI Installation Setup via USB . . . 86
1.3.2.4 NFVI Installation File-Based Image . . . 90
1.3.3 Preparing for Cisco NFVI Installation . . . 92
1.3.3.1 Cisco NFVI Hardware Installation . . . 93
1.3.3.2 ToR Switch Configuration for C-Series Pods . . . 94
1.3.3.3 Preparing Cisco IMC . . . 97
1.3.3.4 Management Node on UCS C-series (M4/M5) . . . 98
1.3.3.5 Management Node on Quanta Servers . . . 100
1.3.3.6 Cisco VIM Software Hub . . . 101
1.3.3.7 UCS C-Series Pod . . . 109
1.3.3.8 Out-of-Band Management Switch . . . 113
1.3.3.9 Third-Party Compute Support . . . 114
1.3.4 Centralizing Management Node . . . 115
1.3.5 CVIM Monitor . . . 124
1.3.6 Highly Available CVIM Monitor . . . 131
1.3.6.1 Overview of HA CVIM-MON . . . 132
1.3.6.2 Hardware Requirements for HA CVIM MON . . . 133
1.3.6.3 Networking Layout . . . 134
1.3.6.4 Network Topologies . . . 135
1.3.6.5 Architecture . . . 139
1.3.6.6 Installation Modes . . . 140
1.3.6.7 Setup File . . . 143
1.3.6.8 HA CVIM-MON Installer . . . 148
1.3.6.9 Resources . . . 154
1.3.6.10 Pod Operations . . . 157
1.3.6.11 Stack Operations . . . 171
1.3.6.11.1 Overview of Stacks . . . 174
1.3.6.12 External Servers . . . 175
1.3.6.13 Updating Nodes . . . 176
1.3.6.14 Custom Grafana Dashboards . . . 177
1.3.6.15 Alert Rules . . . 181
1.3.6.16 Alert Manager . . . 185
1.3.6.17 Backup . . . 189
1.3.6.18 Restore . . . 191
1.3.6.19 Reinstallation of CVIM-MON HA Pod . . . 192
1.3.6.20 CVIM MON HA Cluster Monitoring . . . 193
1.3.6.21 HA CVIM Mon Upgrade . . . 196
1.3.6.22 SERVER MON . . . 199
1.3.7 Cisco VIM . . . 200

1.3.7.1 Installation Sequence . . . 201
1.3.7.2 Deployment of Management Node . . . 202
1.3.7.3 Cisco VIM Client Details . . . 203
1.3.7.4 Pod Reinstallation . . . 207
1.3.7.5 Control and Data Plane Testing . . . 208
1.3.7.6 Updating Cisco VIM Software . . . 209
1.3.8 Unified Management . . . 210
1.3.8.1 UM Overview . . . 211
1.3.8.2 UM with Internet Access . . . 213
1.3.8.3 UM with Cisco VIM Software Hub . . . 222
1.3.8.4 UM UI User Authentication and Authorization with LDAP . . . 223
1.3.8.5 UM Without SMTP . . . 225
1.3.8.6 UM Without Internet Access . . . 227
1.3.8.7 UM Optional Services . . . 233
1.3.8.8 UM Post Bootstrap Validation Checks . . . 234
1.3.8.9 UM Admin Login for Standalone Setup . . . 238
1.3.8.10 UM Pod Admin Login for Standalone Setup . . . 239
1.3.8.11 UM Host Administrators . . . 240
1.3.8.12 Reinstallation of UM Node . . . 245
1.4 Verifying Installation . . . 246
1.4.1 Displaying IP Addresses . . . 247
1.4.2 Cisco VIM Client CLI Availability . . . 248
1.4.3 Displaying Cisco NFVI Logs . . . 249
1.4.4 Accessing OpenStack API Endpoints . . . 250
1.4.5 Assessing Cisco NFVI Health . . . 251
1.4.6 HA Proxy Dashboard/ELK Stack Logs . . . 253
1.4.7 Testing Pod/Cloud Infrastructure . . . 254
1.5 Cisco VIM REST API . . . 255
1.5.1 Management Node VMs Lifecycle . . . 256
1.5.2 Cisco VIM REST API Using curl for IPv6 . . . 263
1.5.3 Cisco VIM REST API Using curl for IPv4 . . . 272
1.5.4 API Resources . . . 277
1.5.4.1 Hardware Management Utility . . . 311
1.5.4.2 Disk and OSD Maintenance . . . 314
1.5.4.3 Cloud Sanity . . . 321
1.5.4.4 Mandatory/Optional Feature Mapping . . . 325
1.5.4.5 Testing and Polling . . . 328
1.5.4.6 Post-Installation Operations . . . 332
1.5.4.7 Version and Hardware Information . . . 334
1.5.4.8 OpenStack Setup . . . 336
1.5.4.9 Update . . . 338
1.5.4.10 Install Resource . . . 341
1.5.4.11 Nodes and Replace Controller . . . 344
1.5.4.12 Setupdata and Offline Validation . . . 349
1.5.5 REST API Overview . . . 354
1.6 Monitoring Performance . . . 356
1.6.1 Infrastructure Log Management . . . 357
1.6.2 Displaying Log Files . . . 359
1.6.3 Kibana Dashboard Login . . . 363
1.6.4 Rotation of Logs . . . 370
1.6.5 Elasticsearch . . . 371
1.6.6 CVIM-MON . . . 374
1.6.7 Network Performance Test . . . 382
1.7 Unified Management NG . . . 383
1.7.1 NG UM Blueprints . . . 384
1.7.1.1 NG Overview of UM Blueprint . . . 385
1.7.1.2 NG Create Blueprint . . . 386
1.7.1.3 NG Activating Blueprint in Existing Pod . . . 387
1.7.1.4 NG Validating Blueprint . . . 388
1.7.1.5 NG Viewing and Downloading Blueprint . . . 389
1.7.2 Using NG UM . . . 390
1.7.2.1 NG Naming Conventions . . . 391
1.7.2.2 NG UM Administrator . . . 392
1.7.2.3 NG Pod Administrator . . . 394
1.7.2.4 NG Pod Users . . . 395
1.7.2.5 NG Pod User Administration . . . 396
1.7.2.6 NG UM Dashboard . . . 399
1.7.2.7 NG Pod Registration/Management . . . 402
1.7.2.8 NG Monitoring Pod Status . . . 404
1.7.2.9 NG Context Switching Between Pods . . . 405
1.7.2.10 NG Cisco VIM Pod Software Update . . . 406
1.7.2.11 NG Managing Passwords . . . 407
1.7.2.12 NG Day 2 Reconfigure/Enablement . . . 409
1.7.2.13 NG CVIM RMA for Auto-TOR . . . 413
1.7.2.14 NG Cisco VIM Validation Tools . . . 414
1.7.2.15 NG Cisco VIM Operational Tools . . . 418
1.7.2.16 NG Managing Hardware . . . 421
1.7.2.17 NG Managing Power . . . 429
1.7.2.18 NG Uploading Glance Images . . . 432

1.8 Day 2 Operations of UM . . . 434
1.8.1 Restarting UM . . . 435
1.8.2 Reconfiguring UM . . . 436
1.8.3 Update/Upgrade UM . . . 444
1.8.4 Rollback UM . . . 447
1.8.5 Commit UM . . . 448
1.8.6 Migrating of UM to NG Instance . . . 449
1.8.7 Shutting Down UM . . . 451
1.8.8 UM Tech Support . . . 452
1.9 BIOS/BMC/Firmware Update . . . 453
1.9.1 BIOS/BMC/Firmware Update Overview . . . 454
1.9.2 Cisco UCS Firmware Upgrade . . . 455
1.9.3 Quanta Firmware Upgrade . . . 458
1.9.4 Intel FPGA PAC N3000 Firmware Update Support . . . 461
1.10 Configuration . . . 464
1.10.1 Setup Configuration File . . . 465
1.10.2 ToR Management . . . 466
1.10.3 Servers and Network Option . . . 478
1.10.4 Compute Level Options . . . 494
1.10.5 OpenStack Configuration . . . 497
1.10.6 VPP VLAN . . . 506
1.10.7 L3 Fabric Deployment Details . . . 508
1.11 Supporting RMA for Auto-ToR . . . 512
1.12 Optional Services . . . 513
1.12.1 Heat and Ceilometer . . . 514
1.12.2 Ironic Support . . . 515
1.12.3 LDAP Support . . . 517
1.13 Baremetal Instances . . . 519
1.14 VM Resizing and Migration . . . 521
1.15 Supported Integration . . . 523
1.15.1 NetApp Integration . . . 524
1.15.2 Enabling SolidFire . . . 525
1.15.3 Enabling Zadara . . . 526
1.15.4 Enabling Auto-ToR via ACI API . . . 528
1.15.5 Enabling Cisco VIM with ACI Plugin . . . 532
1.15.6 Updating APIC Parameters . . . 536
1.15.7 Red Hat IDM . . . 538
1.15.8 Swift Integration . . . 540
1.16 Supported Features . . . 543
1.16.1 Platform Security . . . 544
1.16.2 Enabling NFVBench . . . 550
1.16.3 Customization of Edge . . . 553
1.16.4 Installation Mode . . . 555
1.16.5 OpenStack Features . . . 556
1.16.5.1 Memory/CPU Usage . . . 557
1.16.5.2 DHCP Reservations . . . 559
1.16.5.3 Trusted Virtual Functions . . . 560
1.16.5.4 Buffer Size Setup . . . 561
1.16.6 VPP Port Mirroring Usage . . . 562
1.16.7 VXLAN-EVPN Setup . . . 565
1.16.8 Head-End Replication Option . . . 567
1.16.9 Enabling BGP Adjacency . . . 568
1.16.10 Neutron Port Rebinding . . . 569
1.16.11 Managing Provider/Tenant VLAN Ranges . . . 571
1.16.12 Migrate SRIOV . . . 573
1.16.13 Augmenting VIC/NIC Pods . . . 574
1.16.14 P-GPU . . . 575
1.16.15 SR EVPN . . . 576
1.16.16 Cinder Volume Multi-attach . . . 578
1.16.17 Virtual GPU Support . . . 580
1.16.18 Forwarding EFK Logs . . . 581
1.16.19 Network File System . . . 582
1.16.20 TTY Logging . . . 583
1.16.21 Branding VM Workload . . . 584
1.16.22 NFVIMON . . . 585
1.17 Cisco VIM Upgrade . . . 591
1.17.1 Upgrading Cisco VIM in Cloud . . . 592
1.17.2 Upgrading Cisco VIM Using Network Installation (from CVIM 3.4.x) . . . 597
1.17.3 Upgrading Cisco VIM Using Network Installation (from CVIM 4.0.0) . . . 599
1.17.4 Upgrading Cisco VIM Using File-Based Image (from CVIM 3.4.x) . . . 600
1.17.5 Upgrading Cisco VIM Using File-Based Image (from CVIM 4.0.0) . . . 602
1.17.6 Upgrading Cisco VIM Using USB (from CVIM 3.4.x) . . . 603
1.17.7 Upgrading Cisco VIM Using USB (from CVIM 4.0.0) . . . 606
1.18 Cisco VIM Update . . . 608
1.18.1 Prerequisites and Assumptions . . . 609
1.18.2 Updating Cisco VIM in Running Cloud . . . 610
1.18.3 Updating Cisco VIM Using USB . . . 611
1.18.4 Updating Cisco VIM Using Network Installation . . . 614
1.19 Administration . . . 615

1.19.1 Managing Pods . . . 616
1.19.1.1 NUMA Pinning . . . 617
1.19.1.2 Management Storage IP . . . 618
1.19.1.3 Pod Recovery . . . 619
1.19.1.4 Managing Hosts . . . 622
1.19.1.5 Identifying Installer Directory . . . 625
1.19.1.6 General Guidelines . . . 626
1.19.2 Managing Scheduler Filters . . . 629
1.19.3 Monitoring Cisco NFVI Health . . . 630
1.19.4 Assessing Cisco NFVI Status . . . 633
1.19.5 Service Catalog URL . . . 637
1.19.6 Checking Network Connections . . . 640
1.19.7 General Scheme of Enabling Optional Services . . . 641
1.19.8 Managing VIM Administrators . . . 642
1.19.9 Managing Power and Reboot . . . 645
1.19.10 Read-only OpenStack Role . . . 649
1.19.11 Cisco VIM Advanced OpenStack Policy . . . 651
1.19.12 Security . . . 653
1.19.12.1 Verification . . . 654
1.19.12.2 Reconfiguration of CVIM Parameters . . . 658
1.19.12.3 Cloud Settings . . . 666
1.19.12.4 Fernet Key Operations . . . 667
1.19.12.5 Certificates . . . 668
1.19.12.6 LDAP AD Support with Keystone v3 . . . 671
1.19.12.7 NetApp from http to https . . . 673
1.19.12.8 Hardening Cisco VIM Deployment . . . 674
1.19.12.9 Securing Management Node . . . 676
1.19.12.10 LDAP Authentication for Kibana . . . 680
1.19.12.11 Enabling Barbican . . . 682
1.19.12.12 Enabling Management Network over TLS . . . 684
1.19.12.13 Usage of Pre-encrypted Images . . . 686
1.19.13 Storage . . . 690
1.19.13.1 Storage Architecture . . . 691
1.19.13.2 Ceph Storage . . . 692
1.19.13.3 Glance . . . 698
1.19.13.4 Cinder . . . 700
1.19.13.5 Nova . . . 702
1.19.13.6 Docker Disk Space Usage . . . 705
1.19.14 Monitoring with CVIM-MON . . . 706
1.19.14.1 Alerting Rules Customization . . . 707
1.19.14.2 Alert Manager and Receiver Customization . . . 711
1.19.14.3 Silencing Alerts . . . 715
1.19.15 CVIM Pod Migration from Micro to Full . . . 720
1.20 Backup and Restore . . . 722
1.20.1 Backing Up Management Node . . . 723
1.20.2 Restoring Management Node . . . 726
1.20.3 Management Node Autobackup . . . 729
1.20.4 Management Node Migration . . . 730
1.20.5 Backing Up UM Node . . . 731
1.20.6 Restoring UM Node . . . 735
1.21 Managing Cisco VIM Software Hub . . . 737
1.22 Troubleshooting . . . 741
1.22.1 Cisco NFVI Node . . . 742
1.22.2 Pre-checks for Storage Removal . . . 746
1.22.3 General Troubleshooting Procedures . . . 748
1.22.4 Connection/Installation Problems . . . 751
1.22.5 Management Node Recovery Scenarios . . . 753
1.22.6 Compute Node Recovery Scenario . . . 762
1.22.7 Technical Support Tools . . . 765
1.22.8 Disk and OSD Maintenance Tools . . . 769
1.22.9 Utility Tool . . . 775
1.22.10 Cisco VIM Client Debug Option . . . 777
1.23 Wirring Diagrams . . . 781

Cisco VIM 4.2.3 Release Notes

Cisco VIM 4.2.3 Release Notes

- Release Date
- Introduction
- Features of Cisco VIM
- Known Caveats

Release Date

First Published: April 21, 2022

Introduction

Cisco Network Function Virtualization Infrastructure (Cisco NFVI) provides the virtual layer and hardware environment in which virtual network functions (VNFs) operate. VNFs provide a well-defined network function that offers routing, intrusion, detection, Domain Name Service (DNS), caching, Network Address Translation (NAT), and other network functions. While the network functions required a tight integration between a network software and hardware in the past, VNFs decouple the software from the underlying hardware.

Cisco NFVI 4.2.3 is based on the Train release of OpenStack, an open source cloud operating system that controls large pools of compute, storage, and networking resources. The Cisco version of OpenStack is Cisco Virtualized Infrastructure Manager (Cisco VIM). Cisco VIM manages the OpenStack compute, network, and storage services, and all Cisco NFVI build and control functions.

Key roles of Cisco NFVI pods are:

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

Hardware used to create the Cisco NFVI pods include:

- Cisco Unified Computing System (UCS) C240 M4 or C240 M5 or C220 M5: Performs management and storage functions, and services. Includes dedicated Ceph (UCS 240-M4 or UCS 240-M5) distributed object store and the file system. Only Red Hat Ceph is supported.
- Cisco UCS C220/240 M4 or M5: Performs control and compute services.
- HP DL 360 Gen9: Supports as a third-party compute where the control plane is Cisco UCS server.
- Dell PowerEdge R740: Supports as a third-party compute with Cisco UCS server as control and ceph node (on Intel XXV710 NIC).
- Combination of M5 series servers for VIC or NIC (40G) based hyper-converged and Micropod offering.
- Quanta servers: Used as an alternate for Cisco UCS servers for the cloud installation at the core and edge. Supports automated installation of the Central Ceph cluster for providing Glance image services to the edge pod.
- Cisco VIM UCS based pod to act as a Central Ceph cluster for Glance image services.

The UCS C240 and C220 servers are M4/M5 Small Form Factor (SFF) models, where the operating systems boot from Hard Disk Drive (HDD)/Solid State Drive (SDD) for control/compute nodes, and boot from internal SSD for Ceph nodes. Cisco supports pure Intel NIC configuration and Cisco 40G VIC with Intel NIC configuration.

Software applications that manage Cisco NFVI hosts and services include:

- Red Hat Enterprise Linux (RHEL) 8.2 with OpenStack Platform 16.1: Provides the core Operating system with OpenStack capability. RHEL 8.2 and OSP 16.1 are installed on all Cisco NFVI UCS servers.
- Cisco VIM: An OpenStack orchestration system that helps to deploy and manage an OpenStack cloud offering from bare metal installation to OpenStack services, considering the hardware and software redundancy, security, and monitoring. OpenStack release with more features and usability enhancements are tested for functionality, scale, and performance.
- Cisco Unified Management (UM): Deploys, provisions, and manages Cisco VIM on Cisco UCS servers. It provides UI to manage multiple pods, when installed on a dedicated server Unified Management node.
- Cisco VIM Monitor: Used to provide integrated monitoring and alerting of the NFV Infrastructure layer.
- Cisco UCS Manager: Used to perform certain management functions, when UCS B200 blades are installed.
- Cisco Integrated Management Controller (IMC): When installing Cisco VIM, Cisco IMC 4.x or later is recommended.

  Following is the list of Cisco IMC versions with which Cisco VIM has been tested in the lab:

UCS-M4 servers	• Cisco recommends Cisco IMC 4.0(2L). • Extended support of 4.0(1a), 4.0(1b), 4.0(1c), 4.0(2L) is available.

UCS-M5 servers	• Supports CIMC 3.1(2b), 4.0(1a), 4.0(1c), 4.0(2f), 4.0(4d), 4.0(4L), 4.0(4m)*, and 4.1(2a)*, 4.1(3d)*. • Recommended to use 4.0(4L). • Do not use 3.1(3c) to 3.1(3h), 3.0(4a), 4.0(2c), or 4.0(2d). • A minimum bundle version of CIMC 4.0(4d) is required for Cascade Lake support. • For GPU support, you must ensure that the server has CIMC 4.0(2f). For UCS-C240, use 4.1(2d) or higher. ⚠ * Indicates that it is tested with Intel NIC BOM. Other versions of CIMC may work, but you must test Cisco VIM deployment with those versions before rolling it into production. We recommend the customers to move forward with the new releases of CIMC versions after testing and validating it in a non production environment, if old releases are deprecated or if new fixes solves their problem.

Layer 2 networking protocols include:

- VLAN supported using Open vSwitch (OVS).
- VLAN supported using ML2/VPP. It is supported only on Intel NIC.

For pods based on C-series with Intel NIC Single Root I/O Virtualization (SRIOV), the 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 over the same physical NIC.

If you use only VIC, VPP as a mechanism driver is not supported.

Features of Cisco VIM

Cisco VIM is the only standalone fully automated cloud lifecycle manager offered by Cisco for the private cloud. It integrates with Cisco UCS C-series servers, Cisco VIC and Intel NIC. It helps cloud administrators set up and manage private clouds.

The following are the features of Cisco VIM:

Feature Name	Comments

Infrastructure, OpenStack & Ceph version	• RedHat 8.2 EUS Kernel (12/13/2021) and OSP 16.1.7 (12/13/2021)
	  ◦ RHEL 8.2 Version: 4.18.0-193.70.1.el8_2.x86_64
	  ◦ RHEL 8.2 Real Time Version: 4.18.0-193.70.1.rt13.120.el8_2.x86_64
	• Ceph 4.1
	• Python 3.6.8
	• Docker 20.10 for management node only. Podman 1.6.4-19 for control and compute nodes.

Hardware support matrix	• UCS C220 M4 controller or compute with Intel V3 (Haswell) • UCS C240/220 M4 controller or compute with Intel V4 (Broadwell) • UCS C240/220 M5 controller or compute with Intel Skylake or Cascade Lake • HP DL360 Gen 9 with control plane on Cisco UCS M4 servers • Dell PowerEdge R740 with control plane on Cisco UCS M5 servers (with XXV710) • UCS C220/240 M5 in a Micropod environment, with an option to add up to 16 UCS C220/240 M5 computes • UCS C240/220 M5 controller or compute with Intel X710 NIC, SR-IOV, and Cisco Nexus 9000 or Cisco NCS 5500 series switch as ToR • UCS C240/220 M5 servers with Cisco 1457 (for control plane) and Intel XXV710 NIC (for data plane with VPP) and SR-IOV • Support of physical GPU in M5 • Support of UCS-C240 for edge deployment of the cloud • Support of vGPU in M5 (Tech-preview) • Quanta servers as an alternative to Cisco UCS servers for fullon, micro (D52BQ-2U 3UPI), and edge (D52BE-2U) deployment of the cloud • Quanta sever support with Skylake or Cascade Lake Intel CPUs • Quanta servers for Central Glance (D52BQ-2U 3UPI) server to offer glance image services to edge pod. Also, offers equivalent BOM with UCS-M5 servers. • SATA M.2 (960G) as an option for a boot drive • Support of both HDD and SSD based management node or UM node • UCS 220/240 M5 in a nano-pod environment with 2-X710 Intel NIC or 25G/40G VIC/NIC BOM
NIC support	• Cisco VIC: VIC 1227, 1240, 1340, 1380, 1387 (for M5) in 40G VIC/NIC offering, 1457 • Intel NIC: X710, 520, XL710, xxv710 (25G)

Pod type	• Dedicated control, compute, and storage (C-series) node running on Cisco VIC (M4) or Intel X710 (for M4 or M5) (full on) with Cisco Nexus 9000 or Cisco NCS 5500 series switch (only for Intel NIC and VPP as mechanism driver) as ToR. For fullon pods based on Quanta (D52BE-2U) servers, the NIC is xxv710 (25G) with Cisco Nexus 9000 as ToR. Support of UCS-M4 compute (10G VIC with 2-XL710) with UCS-M5 (Cisco VIC 1457 with 2-XL710). • Dedicated control, compute, and storage (C-series) node running on Cisco VIC and Intel NIC (full on) with Cisco Nexus 9000 as ToR. Only SRIOV is supported on Intel NIC. Support of Intel X520 (with 2 NIC cards or compute) on M4 pods or XL710 (2 or 4 NIC cards or compute) on M4/M5 pods for SRIOV cards in the VIC/NIC combination. For M4 pods, VIC or NIC computes running XL710 and X520 can reside in the same pod. Few computes can run with or without SRIOV in a given pod. • Dedicated control, compute, and storage (UCS M5 SFF C-series) node running on Cisco VIC 1457 and Intel xxv710 NIC (full on) with Cisco Nexus 9000 as ToR. Only SRIOV is supported on Intel NIC. With VPP and OVS as the mechanism driver, the number of SRIOV ports are 2 or 4, respectively. • Micropod: Integrated (AIO) control, compute, and storage (C-series) node running on Cisco VIC, Intel X710X, or VIC and NIC combinations. Micropod can be optionally expanded to accommodate more computes (up to 16) running with the same NIC type. This can be done as a Day 0 or Day 1 activity. The computes can boot off HDD or SSD. From Cisco VIM 3.4.1, the Micropod option has been extended to Quanta (D52BE-2U) servers with Intel XXV710 NIC (25G) and Cisco Nexus 9000 (-FX series) as ToR. • Hyper-converged on M4 (UMHC): Dedicated control and compute nodes with all storage acting as compute nodes (M4 C-series) and running on a combination of 1xCisco VIC (1227) and 2x10GE 520 or 2x40GE 710XL Intel NIC with an option to migrate from one to another. You can extend the pod to M5-based computes with 40G Cisco VIC along with 2x40GE 710XL NIC (optionally). • NGENA Hyper-Converged (NGENAHC): Dedicated control and compute nodes, with all storage acting as compute (C-series) nodes. All nodes have a combination of 1xCisco VIC (1227) for control plane, and 1x10GE 710X Intel NIC for data plane over VPP. • Support of M5 as a controller and hyper-converged nodes (with 1457 for the control plane, and 1x10GE X710 (2 port) Intel NIC for data plane) in an existing M4 based pod. • Hyper-converged on M5: Dedicated control and compute nodes with all storage acting as compute (C-series) nodes, running on a combination of 1xCisco VIC (40G) and 2x40GE 710XL Intel NIC. Support of M5 as controller and hyper-converged nodes (with 1457 for control plane, and 1x10GE X710 (2 port) Intel NIC for data plane) in an existing M4-based pod. • Edge: Accommodates restricted power and limited rack space. Quanta (D52BQ-2U 3UPI) or UCS-C240 servers with three converged control and compute nodes, expandable to 16 additional compute nodes. The edge cloud communicates with Quanta/UCS server based Central Ceph cluster for glance service. Persistent storage is not available. • Ceph: Designed to provide glance image services to edge cloud. Quanta (D52BE-2U)/UCS-M5 servers with three converged cephcontrol and cephosd nodes, expandable to additional cephosd nodes for additional storage. Support of UCS based pod for glance image services ⚠ • In a full-on (VIC based), or hyper-converged pod, computes can either have a combination of 1-Cisco VIC (1227) and (2x10GE 520/2x40GE 710XL Intel NIC) or 1-CiscoVIC (1227). The compute running pure Cisco VIC does not run SR-IOV. In 2.4, Cisco supports HP DL360 Gen9 and Dell PowerEdge R740 as a third-party compute. • Cisco VIM does not support a combination of computes from different vendors.
ToR support	• Support of Cisco NCS 5500 (with recommended Cisco IOS XR version 6.1.33.02I or 6.5.1) with splitter cable. Day 0 configuration can support user-defined route-target and ethernet segment ID (ESI). • Supports two or more Cisco NCS 5500 with ships in the night. In this case, the Cisco NCS 5500 configuration is set ahead of time before the pod installation commences. • NXOS support (preferably N9K)
IPV6 support for management network	• Static IPv6 management assignment for servers. • Support of IPv6 for NTP, DNS, LDAP, external syslog server, and AD. • Support of IPv6 for the cloud API endpoint. • Support of CIMC over IPv6. • RestAPI over IPv6. • Support for IPv6 filters for administration source networks. • Support of UM over IPv6.
Centralization of management node	Value Engineering to reduce the overhead of one physical server per pod. Supports management node, Unified Management node, and Cisco VIC software hub.

Mechanism drivers	OVS/VLAN, VPP (20.09)/VLAN (Fast Networking, Fast Data FD.io VPP/VLAN, based on the FD.io VPP fast virtual switch over intel NIC).
SDN controller integration	Integration of ACI Plugin is supported with OSP 16.1. ACI (ships in the night or ToR automation using APIC API) with Cisco VIC/Intel NIC on the UCS C-series M4 platform, and with Intel NIC on Quanta (D52BQ-2U 3UPI) servers.
Installation or update method	• Fully automated online or offline installation. • Support of offline installation via USB or file-based image. • Support of Cisco VIM Software Hub to mitigate the problem associated with the logistics of USB distribution for air-gapped installation. • Support of USB 3.0 64GB for M5 and Quanta-based management node. Support of UCS 2.0 64GB for M4-based management node.
Scale	• Full Pod: Total of 128 nodes (compute and OSD) with Ceph OSD max at 20. ⚠ The 128 node limit is based on the scale testing done with OVS and based on customer deployments. Ensure that you deploy a maximum of 60 nodes at a time. Also, after Day 0, you can add only/remove one ceph node at a time. • Micro/Edge pod: Supports a maximum of 16 standalone compute nodes. • Hyper-converged (HC): Total of 64 nodes including three controllers is available. Maximum of 15 HC (compute and OSD) nodes is supported. ⚠ Ceph OSDs can be HDD or SSD based, but must be uniform across the pod. Computes can boot off 2x1.2TB HDD or 2x960 GB SSD). In the same pod, some computes have SSD, while others can have HDD. Contact Cisco VIM product management team for specific use case and BOM details applicable for each type of pod.
Automated pod life cycle management	• Add or remove compute and Ceph nodes and replace the controller node. • Static IP management for storage network. • Reduction of tenant or provider VLAN through reconfiguration to a minimum of two. • Reconfiguration of passwords and selected optional services. • Automated software update.

Platform security	• Secure OS, RBAC, network isolation, TLS, source IP filtering (v4 and v6), Keystone v3, Bandit, CSDL-compliant, hardened OS, and SELinux. • Change CIMC password post-installation for maintenance and security. • Non-root login for administrators. • Read-only role is available for OpenStack users. • Enabling custom policy for VNF Manager. • Option to disable the management node reachability to the cloud API network. • Hosting of Horizon behind NAT or with a DNS alias. • Cinder volume encryption using Linux Unified Key Setup (LUKS). • Support of configurable login banner for SSH sessions. • Access to management node using Open LDAP or via MS AD • Support for IPv6 filters for administration source networks. • Introduction of Vault to encrypt secrets with reconfigure option. • Enablement of Vault as an option on Day 2. • Extended *permit_root_login* to Unified Management node. • CIMC authentication using LDAP. • Support of out of band and inband RedHat identity, policy and audit (IPA) system. • Support of Horizon and Keystone login settings. • Support of LDAP on Unified Management node. • SSH and password vulnerabilities for management node. • Kernel changes to address vulnerabilities. • Support of FPGA 1.1 image for Quanta GC SKU (2RU1N). • FQDN support for Cisco VIM management API. • LDAP support for Kibana. • Support of TLS over management network. • Support of Barbican using HSM. • Ability to disable VM live migration, snapshot, crash dump, and console. • Cisco VIM advanced OpenStack policy • Disable IPA Cache Credentials • Lock Docker Registry when not used in Management Node
Enhanced Platform Awareness (EPA)	• Supports NUMA, CPU pinning, huge pages, and SRIOV with Intel NIC. • Ability to set HYPERTHREADING at a global level or per compute basis. • Ability to set VM_HUGEPAGE_SIZE and VM_HUGEPAGE_PERCENTAGE at a global level or per compute basis. • Ability to allocate user-defined CPU (up to 6) cores to VPP. • Ability to bring in trusted_vf as a reconfigure option on a per server basis. • Ability to allocate user-defined CPU (up to 12) cores to Ceph for Micropod and hyper-converged nodes. • Ability to allocate user-defined CPU (up to 30) cores to compute nodes. • Ability to set LIBVIRT_WRITETHROUGH_CACHE at global or per compute basis. • Ability to set ISOLCPU at a global level or per compute basis. • Ability to set VIC_link_training at global level or per compute basis. • Improve OSD failure detection time • Ability to allocate user-defined memory (16 to 32 GB) to Ceph for Micropod and hyper-converged nodes. • Ability to allocate user-defined memory (25 to 500 GB) for compute nodes. • Ability to support custom mapping between OpenStack physnet to SRIOV PF.
HA and Reliability	• Redundancy at hardware and software level. • Automated backup and restore of the management node. • Encryption of management node backup snapshot • Relaxation of Security Enhanced Linux (SELinux) requirement for backup and restore of the management node. • Optimization of automated backup and restore of the management node in a connected installation.
Unified Management (UM) support	Single pane of glass in a single mode. Supports multi-tenancy and manages multiple pods from one instance.
Cisco VIM monitor	Collects the metrics from the entire pod. Supports customizing alerts, sending SNMP traps, and exporting to external metric collectors.
Central logging	EFK integrated with external Syslog (over v4 or v6) for a log offload, with optional support of NFS with EFK snapshot.
External Syslog servers	Supports multiple external Syslog servers over IPv4 or IPv6. The minimum and maximum number of supported external Syslog servers are 1 and 3, respectively.
VM migration	• Cold migration and resizing. • NUMA-aware live migration with Virtio and SRIOV.
Pod Migration	Seamless migration of a Micropod to a full pod.

Storage	• Block storage with Ceph or NetApp. • Option to use Ceph for Glance and SolidFire for Cinder. • Option to have multi-backend (HDD and SSD based) Ceph in the same cluster to support various I/O requirements and latency. • Integration of Zadara as an alternate to Ceph. • Support of ObjectStore (via Swift) with SolidFire as backend. • Support for multiple NFS Shares
Monitoring	• Monitor the Cisco VIM pods centrally using the Highly Available Cisco VIM Monitor (HA CVIM-MON) over v4 and v6. • Monitor the Cisco VIM pods individually using the local Cisco VIM Monitor (CVIM-MON) over v4 and v6. • CVIM MON local LDAP support with Grafana, Prometheus, and Alert Manager. • Support of non Cisco VIM managed external servers running RHEL or CentOS. • Ceilometer for resource tracking and alarming capabilities across core OpenStack components is applicable only for fullon pod. • Third-party integration with Zenoss (called NFVIMON)
Optional OpenStack features	• Enable trusted virtual function on a per-server basis. • DHCP reservation for virtual MAC addresses. • Enable VM_HUGE_PAGE_SIZE and VM_HUGE_PAGE percentage on a per-server basis. • Enable CPU and RAM allocation ratio on a per-server basis via add/remove compute or reconfigure. • Flatten volume from snapshot as an option in cinder.
Support of external authentication system	• LDAP with anonymous bind option. • Active Directory (AD).
Software update	Update of cloud software for bug fixes on the same release. Support for VMs associated with more than 8vCPUs with MultiQ enabled Metadata and VPP plugin.*
CIMC upgrade capability	Central management tool to upgrade the CIMC bundle image of one or more servers.
VPP port mirroring	• Ability to trace or capture packets for debugging and other administrative purposes. • Automated update of BMC or BIOS and firmware of Quanta server.
VXLAN extension into the cloud	• Extended native external VXLAN network into VNFs in the cloud. • Support of Layer 3 adjacency for BGP. • Support of single VXLAN network or multi-VXLAN network (with head-end-replication option) terminating on the same compute node. • Support of re-binding of Neutron port to another port. • Support of L3 fabric via VXLAN. ⚠ Only two-VXLAN network is supported.
Technical support for CIMC	Collection of technical support for CIMC.
Enable TTY logging as an option	Enables TTY logging and forwards the log to an external syslog server and EFK stack running on the management node.
Unified Management authentication	Authentication support through local and LDAP.
CIMC authentication using LDAP	Authentication support through LDAP.
Automated enablement of Intel X710/XL710 NIC's PXE configuration on Cisco UCS-C series	Utility to update Intel X710/XL710 NIC's PXE configuration on Cisco UCS C-series.

Power management of computes	Option to selectively turn OFF or ON the power of computes to conserve energy.
Fan policy for servers	Option to set fan policy globally for UCS C-series based pod.
Disk maintenance for pod nodes	Ability to replace faulty disks on the pod nodes without the need to add, remove, or replace node operation.
Branding of VM workload	Ability to check whether the VMs are running on Cisco VIM platform.
Support of workload types	• Supports baremetal (ironic-based) based workloads. • Supports bonding on the Ironic network.
Cloud adaptation for low latency workload	• Enable real-time kernel to support edge pod. • Automated BIOS configuration. • Introduction of custom flavor. • Support of Intel N3000 card on selected servers to handle vRAN workloads. • Support of Cache Allocation Technology (CAT) to handle vRAN workloads. • Support of INTEL_SRIOV_VFS (SRIOV support for Intel NIC) and INTEL_FPGA_VFS (support for Intel N3000 FPGA card) at a per-server level.
Integrated test tools	• Open-source data plane performance benchmarking: VMTP (an open-source data plane VM to VM performance benchmarking tool) and NFVbench (NFVI data plane and a service chain performance benchmarking tool). • Extending VMTP to support v6 over the provider network. • NFVbench support for VXLAN. • Services Health Checks Integration: Cloudpulse and Cloudsanity.

⚠ * Indicates the features introduced in Cisco VIM 4.2.3.

⚠
- For supported BOM details, reach out to *nfvi-plm@cisco.com* or your *Account Team.*
- Configure LACP on the data plane ports of the Cisco Nexus 9000 ToR when Cisco VIM is running on Intel NIC for data plane with VPP as the mechanism driver. When Cisco NCS 5500 is ToR (with VPP), the LACP configuration on the data plane is done through auto-ToR configuration feature of Cisco VIM.

Known Caveats

The following list describes the known caveats in Cisco VIM 4.2.3:

- **CSCvw34928**

  Cisco VIM central-vm is out of sync with RestAPI database, post removal of compute on base deployment.

- **CSCvw20577**

  Nova console historical logs are not transferred after live-migration of instances.

- **CSCvt33521**

  Reconfiguration fails during validation, after the renewal of TLS certificate.

- **CSCvj87681**

  Validation intermittently fails during configuration of auto-tor with Cisco NCS 5500.

- **CSCvw84323**

  HA CVIMMON: The password regeneration for Minio is skipped.
- **CSCvz68190**

  CVIMMON: Custom Alerts file missing post upgrade.

- **CSCvz76137**

CVIMMON: LDAP integration failed when space is there in CN used in bind dn.

- **CSCvz78460**

NFVIMon: Zenoss to provide support for podman to display the containers.

- **CSCvz78454**

NFVIMon: Provide Zenoss zenpack support for Nautilus/Octopus.

- **CSCvz93570**

CVIM Upgrade to 4.2.3 fails with permit_root_login: false configured in setup data.

- **CSCwb34390**

Telegraf: net_stats plugin to get VNIC MAC Address from libvirt instead of VPP

- **CSCwb66397**

VPP: Qemu capability exchange broken

You can use the Bug Search Tool to search for a specific bug or to search for all bugs in a release.

Procedure

1. Go to the Cisco Bug Search Tool.
2. In the **Log In** screen, enter your registered Cisco.com username and password, and then click **Log In**. The Bug Search page opens.
3. To search for a specific bug, enter the bug ID in the Search For field and press Enter.
4. To search for bugs in the current release:

    a. In the **Search For** field, enter **Cisco Network Function Virtualization Infrastructure** with release version and press **Enter**. (Leave the other fields empty.)
    b. When the search results are displayed, use the filter tools to find the types of bugs you are looking for. You can search for bugs by status, severity, modified date, and so forth.

To export the results to a spreadsheet, click the **Export Results to Excel** link.

Cisco NFVI Architecture

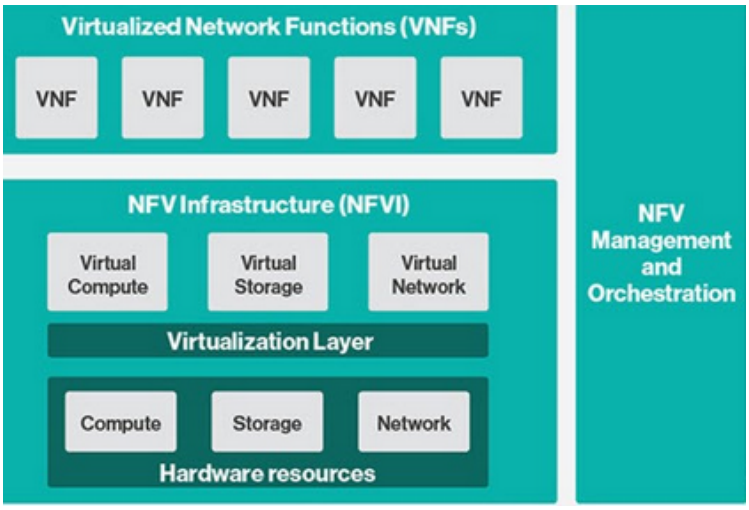
Cisco NFVI Architecture

- Cisco NFVI Overview
- Cisco VIM Overview
- Networking Overview
- UCS C-Series Network Topologies
- Management Node Networking
- IPv6 Support
- UCS C-Series
- High Availability
- Storage Node Overview
- OpenStack Telemetry Service
- NFVBench
- Auto-ToR Configuration via ACI API
- NCS-5500 as ToR Option
- Disk Management
- OSD Maintenance
- Power Management
- Physical Cores and Reserved Memory
- Software Hub
- VXLAN EVPN Design
- VPP Port Mirroring Support
- Segment Routing EVPN
- Container Workload
- Management Node Centralization
- L3 Fabric Deployment
- OpenStack Barbican
- Management Network Over TLS
- Support of Pre-encrypted Image
- ACI Plugin Integration

Cisco NFVI Overview

Cisco Network Function Virtualization Infrastructure Overview

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



Cisco NFVI includes a virtual infrastructure layer (Cisco VIM) that embeds the Red Hat OpenStack Platform (OSP 16.1). Cisco VIM includes the Train release of OpenStack, which is an open source cloud operating system that controls large pools of compute, storage, and networking resources. Cisco VIM manages the OpenStack compute, network, and storage services, and all NFVI management and control functions. Cisco 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 Cisco VIM, are examples of such violations. For clarifications, reach out to Cisco support.

Key Cisco NFVI roles include:

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

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

- Cisco UCS® C240 M4/M5: Performs management and storage functions and services. Includes dedicated Ceph (UCS 240-M4 or UCS 240-M5) distributed object store and file system. (Only Red Hat Ceph is supported).
- Cisco UCS C220/240 M4/M5: Performs control and compute services.
- HP DL360 Gen9: It is a third-party compute where the control plane is Cisco UCS servers.
- Cisco UCS 220/240 M4/M5 (SFF): In a Micropod environment, it is expandable to a maximum of 16 computes.
- Cisco UCS B200 M4 blades: It can be used instead of the UCS C220 for compute and control services. The B200 blades and C240 Ceph server are connected with redundant Cisco Fabric Interconnects managed by UCS manager.
- A combination of M5 series servers are 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 M4 or M5 Small Form Factor (SFF) models where the nodes can boot off a pair of HDDs or SSD as specified in BOM.

The Cisco VIM installer performs bare metal installation and deploys OpenStack services using Docker™ containers to allow for OpenStack services and pod management software updates.
The following table shows the functions, hardware, and services managed by Cisco NFVI nodes.

Function	Number	Hardware	Services

Management	1	• UCS C240 M4 SFF with 8, 16, or 24 1.2 TB HDDs (24 is recommended) • UCS C240 M5 SFF with 8, 16, or 24 1.2 TB HDDs (24 is recommended) • UCS C220 M5 SFF with 8x1.2 TB HDDs • Quanta server (D52BE-2U) with 2x1.2TB HDD • Quanta server (D52BQ-2U 3UPI) with 2x.3.8TB HDD	• Cisco VIM Installer • Cobbler server • Docker Registry • ELK server • CVIM MON components: Prometheus and TSDB
Control	3	• UCS C220/C240 M4/M5 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 • Fluentd
Compute	2+	• UCS C220/C240 M4/M5 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 M4/M5 with two internal SSDs, 1-4 external SSD, 4-20x- 1.2 TB HDDs • SSD-based Ceph: UCS C240 M4/M5 with 2 internal SSDs, minimum of 4 external SSDs, expandable to 24 SSDs • Quanta server (D52BE-2U) HDD Based: 4 SSD 960GB for Journal + 16 SAS HDD (16x2.4 TB) for OSD + 2 (2x2.4 TB SAS 10krpm HDD) for OS • Quanta server (D52BE-2U) SSD Based: 20 SSD (3.8 TB) OSD + 2 OSBoot (2x3.8TB SSD) Micropod/UMHC/NGENAHC environment: • UCS C240 M4/M5 with two 1.2TB HDD for OS boot, one/2 SSDs and 5/10x1.2 TB HDDs • UCS C240 M4/M5 with 2x960GB SSD for OS boot and 4 or 8 x960 GB SSDs	Storage service
Top of Rack (ToR)	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 Cisco 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
- You can use any ToR that supports virtual port channel. Cisco recommends to use Cisco Nexus 9000 SKUs as ToR, which is released as part of Cisco 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 Cisco NFVI hosts and services include:

- Red Hat Enterprise Linux 8.2 with OpenStack Platform 16.1: Provides the core operating system with OpenStack capability. RHEL 8.2 and OPS 16.1 are installed on all target Cisco NFVI nodes.
- Cisco Virtual Infrastructure Manager (VIM): An OpenStack orchestration system that helps to deploy and manage an OpenStack cloud offering from bare metal installation to OpenStack services, taking into account hardware and software redundancy, security and monitoring. Cisco VIM includes the OpenStack Train release with more features and usability enhancements that are tested for functionality, scale, and performance.
- Cisco Unified Management: Deploys, provisions, and manages Cisco VIM on Cisco UCS servers.
- Cisco UCS Manager: Supported UCS manager firmware versions are 2.2(5a) and above.
- Cisco Integrated Management Controller (IMC): Cisco IMC 4.0(4i) or later is supported.

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

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

UCS-M5 servers	• Do not use 3.1(3c) to 3.1(3h), 3.0(4a), 4.0(2c), or 4.0(2d). • Cisco recommends you to use CIMC 4.0(4L). • For Cascade Lake support, you need a bundle version of a minimum of CIMC 4.0(4d). • For GPU support, you must ensure that the server is running with CIMC 4.0(2f). • For UCS-C240, use 4.1(2d) or higher

Enables embedded server management for Cisco UCS C-Series rack servers. Supports Cisco IMC firmware versions of 2.0(13i) or greater for the fresh install of Cisco VIM. Because of recent security fixes, we recommend you to upgrade Cisco IMC to 2.0(13n) or higher. Similarly, Cisco IMC version of 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:

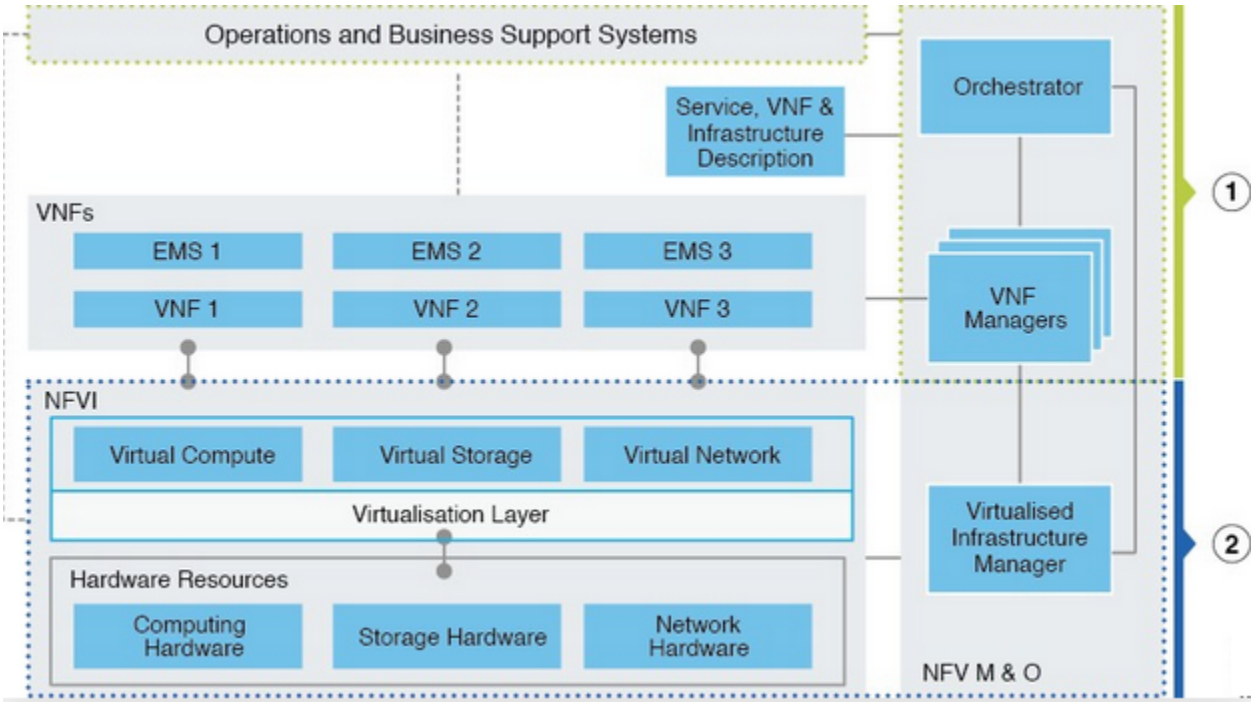
SKU Type	BMC Version	BIOS Version
D52BQ-2U 3UPI (CDC SKU)	4.68.22	3A11.BT17
D52BE-2U (GC SKU)	4.68.22	3A11.BT17

- Cisco Virtual Topology Forwarder (VTF)—VTF leverages Vector Packet Processing (VPP) to provide high-performance Layer 2 and Layer 3 VXLAN packet forwarding.

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

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

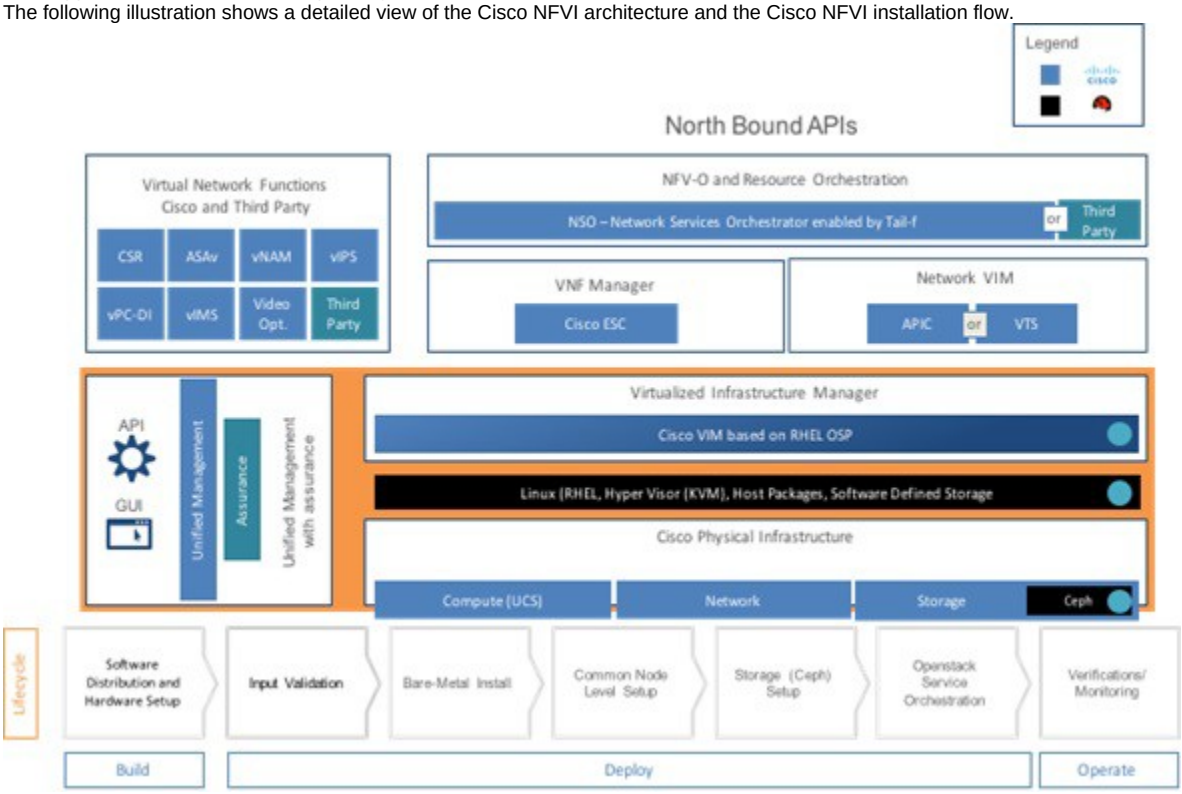
1	• Cisco Network Services Orchestrator • Cisco Elastic Services Controller

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

For cloud networking, Cisco NFVI supports Open vSwitch over VLAN as the cloud network solution for 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 augmented performance mechanism by replacing OVS/VLAN with VPP/VLAN (for Intel NIC).

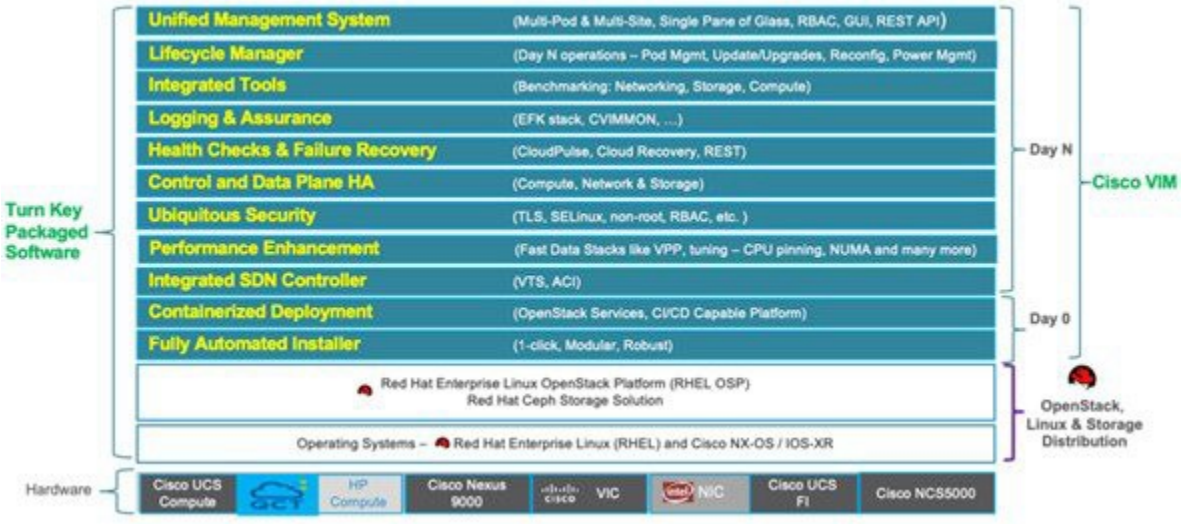
The Cisco NFVI uses OpenStack services running inside containers with HAProxy load balancing and providing high availability to API and management network messaging. Transport Layer Security (TLS) protects the API network from external users to the HAProxy. Cisco VIM installation also includes service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



Cisco VIM Overview

Cisco Virtualized Infrastructure Manager Overview

Cisco Virtualized Infrastructure Manager (VIM) is a fully automated cloud lifecycle management system. Cisco VIM helps to bring up a fully functional cloud in hours, with integrated end-to-end control and data plane verification in place. Cisco VIM offers fully automated Day 1 to Day n cloud lifecycle management. These include capabilities such as pod scaling (expansion), software update, upgrade, or 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 the high-level overview of all Day 0 and Day n items of Cisco VIM.



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

Cisco 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, Cisco VIM segments the various OpenStack networks. The Cisco NFVI network includes six different segments in the physical infrastructure (underlay). These segments are presented as VLANs on the Top-of-Rack (ToR) Nexus switches (except for the provider network) and as vNIC VLANs on Cisco UCS servers. You must allocate subnets and IP addresses to each segment. Cisco NFVI network segments include API, external, management and provisioning, storage, tenant, and provider.

API Segment

The API segment needs one VLAN in an externally accessible subnet that is different from the subnets assigned to other Cisco 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 Cisco NFVI *setup _data.yaml* file, but you must configure the actual subnet using the OpenStack API after the installation. Use the external network to assign OpenStack floating IP addresses to VMs running on Cisco NFVI.

Management and Provisioning Segment

The management and provisioning segment needs one VLAN and one subnet with an address pool 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 Cisco 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 Cisco NFVI (that is, in a lab, it can be a non-routable subnet limited to Cisco NFVI only for C-series pods), or an externally accessible and routable subnet. All Cisco NFVI nodes (including the Cisco VTC node) need an IP address from this subnet.

Storage Segment

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

Tenant Segment

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

Provider Segment

Provider networks are optional for Cisco NFVI operations but are often used for real VNF traffic. You can allocate one or more VLANs for provider networks after installation is completed from OpenStack.
Cisco NFVI renames interfaces based on the network type it serves. The segment Virtual IP (VIP) name is the first letter of the segment name. Combined segments use the first character from each segment for the VIP, with the exception of provisioning whose interface VIP name is "mx" instead of "mp" to avoid ambiguity with the provider network. The following table shows Cisco NFVI network segments, usage, and network and VIP names.

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

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

The Cisco NFVI installer creates the required vNICs, host interfaces, bonds, and bridges with mappings created between all elements. The number and type of created vNICs, interfaces, bonds, and bridges depend on the Cisco NFVI role assigned to the UCS server. For example, the controller node has more interfaces than the compute or storage nodes. The following table shows the networks that are associated with each Cisco NFVI server role.

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

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

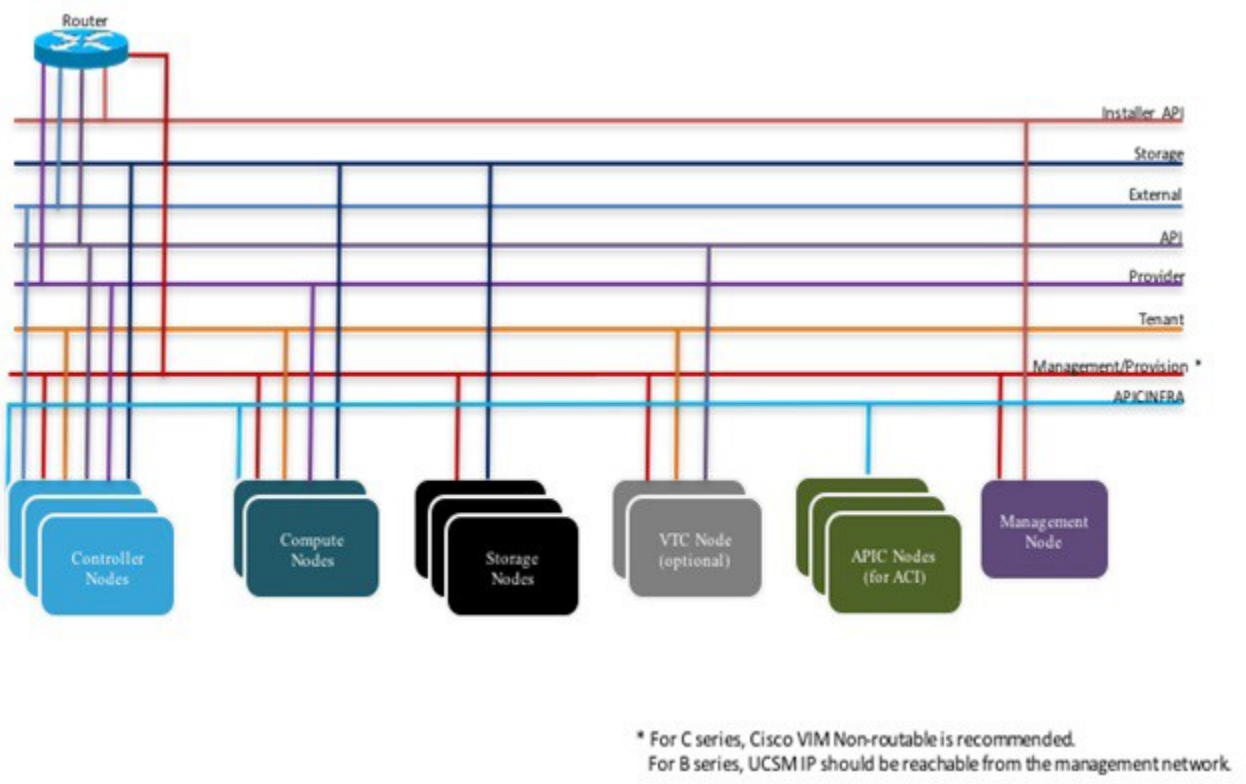
The network arrangement on third-party HP compute is slightly different from that of Cisco compute running with Intel NIC, because the HP computes have 2 less NIC ports than that are available in the Cisco Intel NIC BOM.
Following table lists the differences in the network arrangement between the Cisco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

Network Interface	Cisco UCS Ce220/Ce240M4 /M5 Compute	HPE ProLiant DL360 Gen9 and Quanta Compute	Dell PowerEdge R740
samx	Management control plane network	N/A	Management control plane network

samxpet	Converged control and data plane network based on NIC configuration	Control and data plane network for all other than SRIOV: 1. Management network on *br_mgmt* bridge interface with *samxpet* main interface as one of the member interface (native VLAN configuration required on the top-of-rack switches) 2. Storage network on the sub-interface *samxpet.<storage VLAN>* 3. Tenant and provider networks on veth interface *pet/pet-out* as one of the member interface with *br_mgmt* bridge interface	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 Cisco NFVI deployment, two bridges are created on the controller nodes and linked with interfaces and bonds. The *br_api* bridge connects the API (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 Cisco 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 Cisco VIM has continued to evolve over multiple releases to support customer needs that can vary across multiple dimensions such as cloud capacity, power, physical space, and affordability. The following table shows the available Cisco NFVI hardware and data path deployment combinations.

Pod Type	NIC Type	Hardware Vendor	Mechanism Driver	ToR Type

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

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. Lets call the two NIC cards as NIC_1 and NIC_2 and call their four ports as A, B, C, D. Unlike Cisco VIC based pod, the traffic here is classified as follows:

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

Control Plane

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

1. Management/Provision.
2. Storage.
3. API.

The control plane interface is created by bonding the NIC_1 A port with NIC_2 A port. The bonded interface name is called as samx, indicating that it is carrying Storage, API, Management/Provision traffic (naming convention is similar to Cisco VIC pod). The 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 *samx* 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.
In case of OVS/VLAN, the *pet* interface is used as it is (trunked to carry all the data VLANs) to the Openstack cloud, as all the tagging and untagging happens at the Openstack level. In case of Linux Bridge/VXLAN, sub-interface exists for tenant VLAN to act as the VXLAN tunnel endpoint.

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 *setup_data.yaml* parameter. Unlike the control and data plane interfaces, these interfaces are not bonded and hence there is no redundancy. Each SRIOV port can have maximum of 32 virtual functions and the number of virtual function to be created are configurable through the *setup_data.yaml.* The interface names of the SRIOV 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			+	

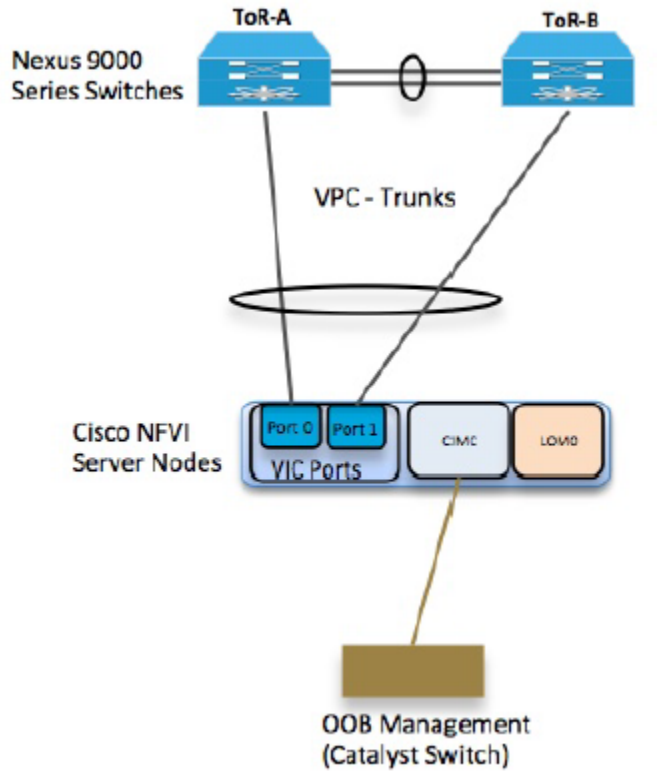
On Intel pod, all kinds of OpenStack networks are created using *physnet1* as the physnet name.

UCS C-Series Network Topologies

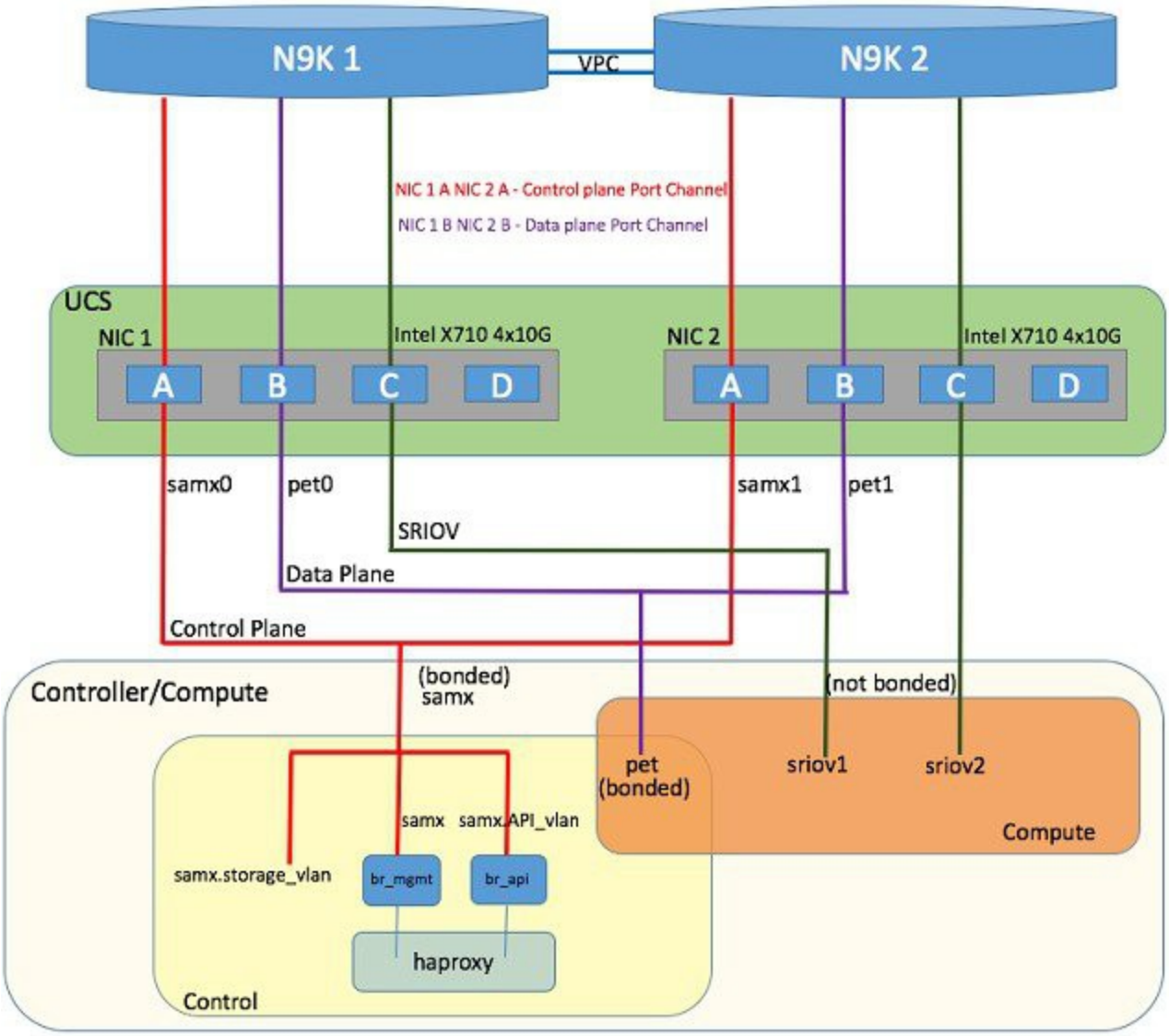
UCS C-Series Network Topologies

Cisco 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 Cisco VIM networks. The UCS Cisco IMC port is connected to an out-of-band (OOB) Cisco management switch.

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



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.

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 Cisco VIM release 2.4.2 onwards, this port option is available for both M4 and M5 based systems or pods.
The following table provides the default link aggregation member pairing support for the pods based on server type:

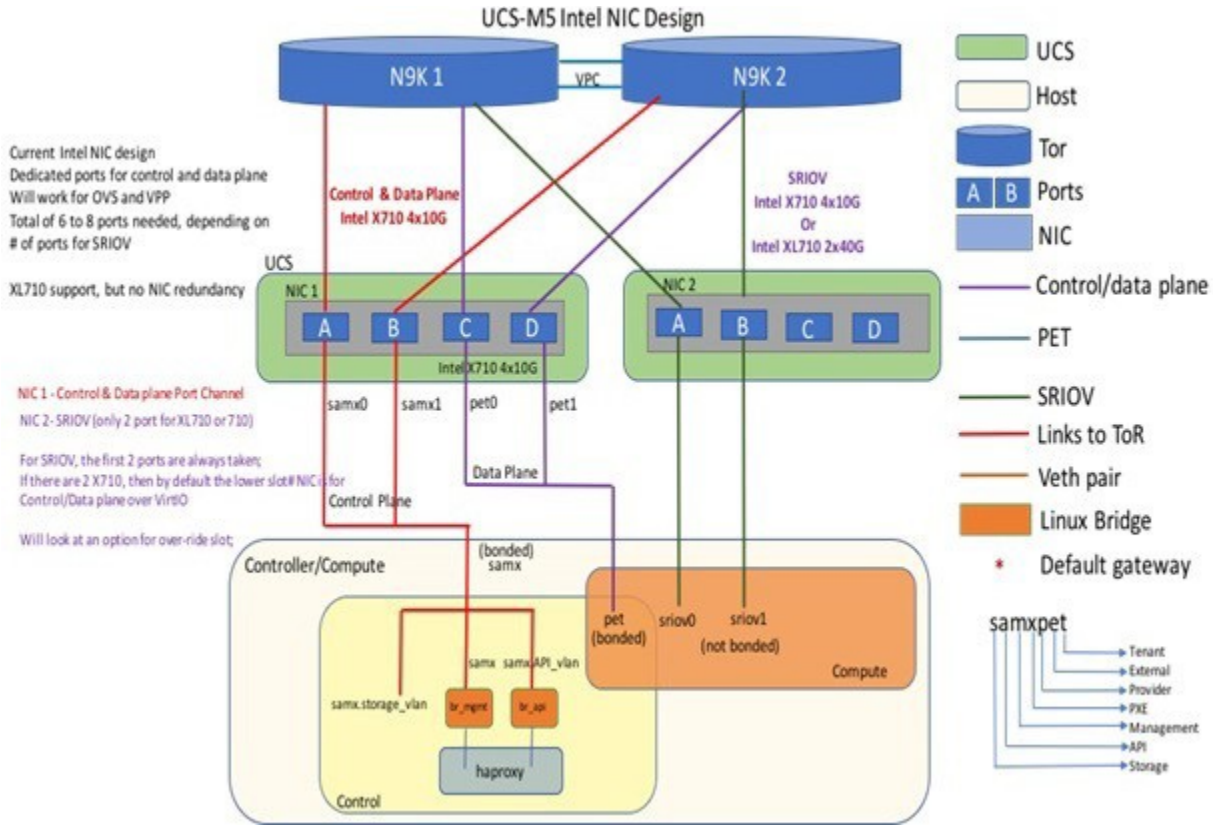
Server/POD Type	Target Functions	Default NIC Layout
M4 Intel NIC based	Control Plane	NIC-1 A + NIC-2 A
	Data Plane	NIC-1 B + NIC-2 B
	SRIOV 0/1	NIC-1 C + NIC-2 C
	SRIOV 2/3	NIC-1 D + NIC-2 D
M5 Intel NIC based	Control Plane	NIC-1 A + NIC-1 B
	Data Plane	NIC-1 C + NIC-1 D
	SRIOV 0/1	NIC-2 A + NIC-2 B
	SRIOV 2/3	NIC-2 C + NIC-2 D

⚠ In M5 pod, a NIC_LEVEL_REDUNDANCY option is available to support the M4 default option for link aggregation settings.

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

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

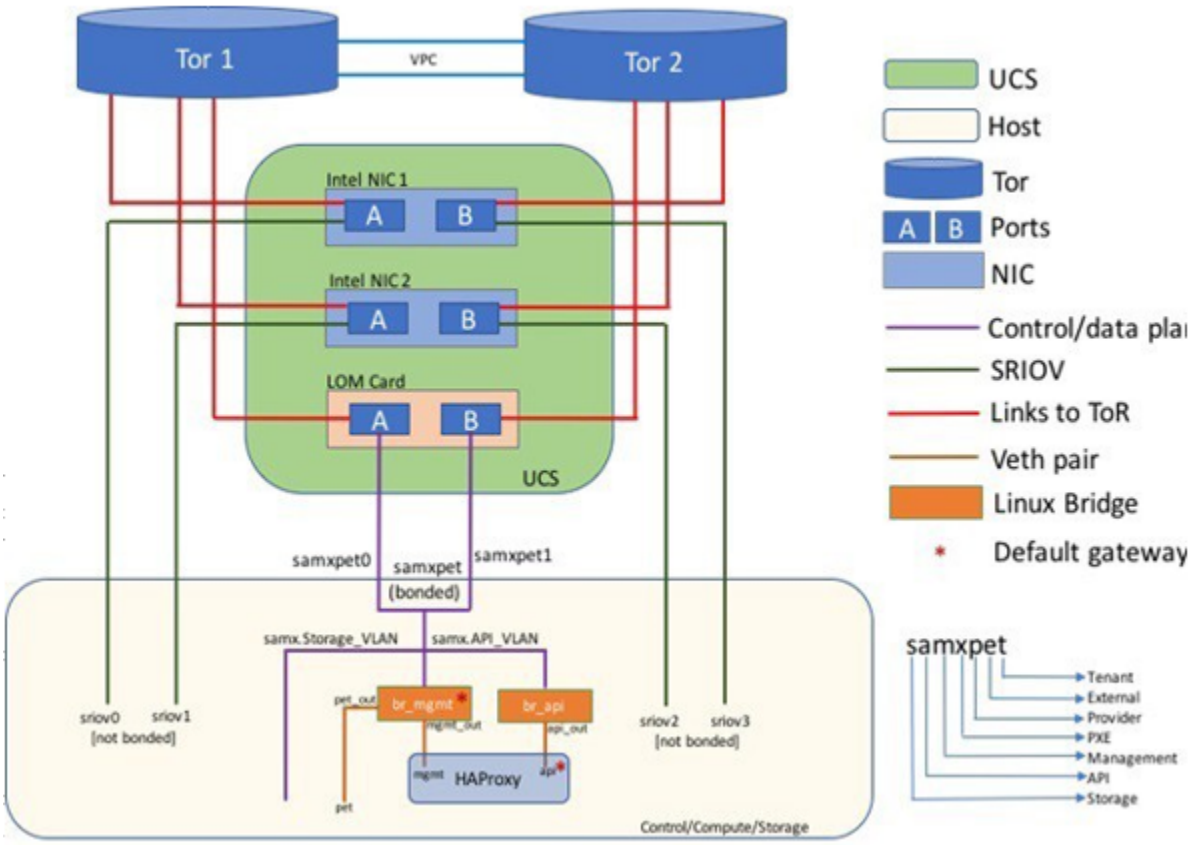
Cisco VIM 2.4 introduces the support of C220/C240 M5 servers in a micropod configuration with an option to augment the pod with additional computes (up to a max of 16). The M5 micropod environment is based on X710 for control and data plane and an additional XL710 or 2xX710 for SRIOV. The SRIOV card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



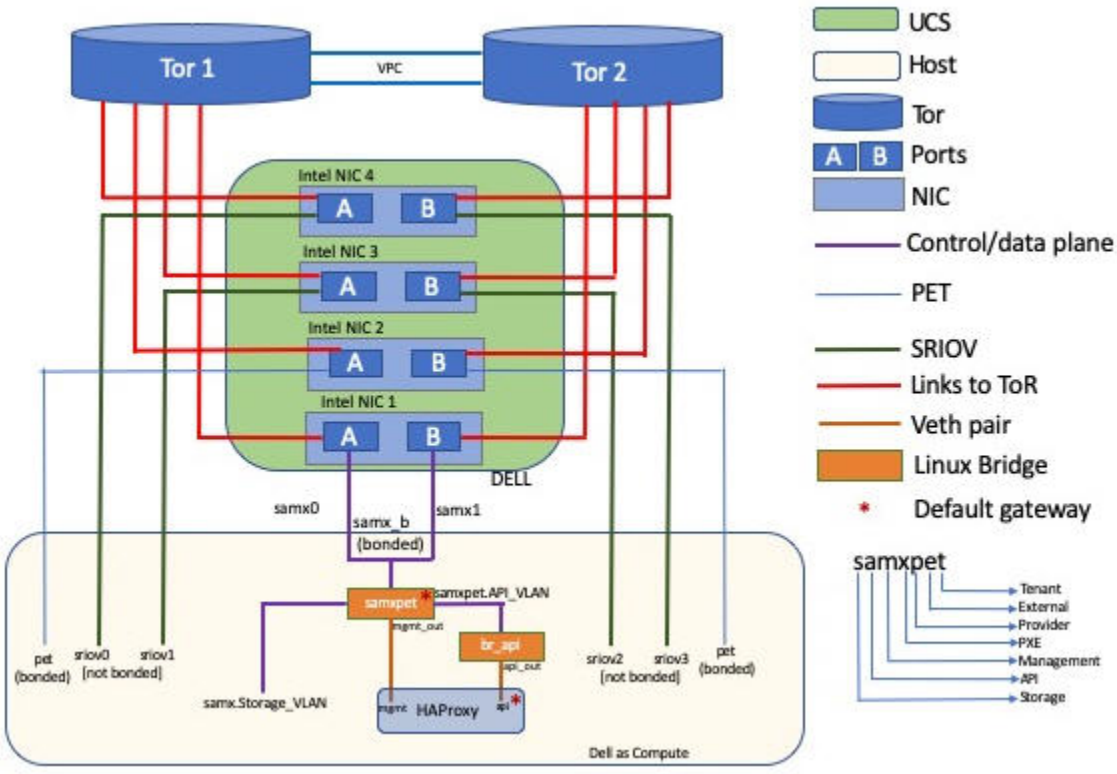
Cisco VIM 2.4 introduces the first third-party compute. The first SKU chosen is HPE ProLiant DL360 Gen9. In Cisco VIM 2.4, the supported deployment is a full-on pod, with OVS as the mechanism driver, where the management, control, and storage nodes are based on existing Cisco UCS c220/240M4 BOM, and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

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

For HP computes, the FlexLOM HP Ethernet 10Gb interface is used for management and tenant network, while the two additional HP Ethernet 10Gb 2-port 560SFP+ adapters are used for SRIOV for the provider network. Listed below is network schematic of the HP Compute node (HP DL360GEN9).



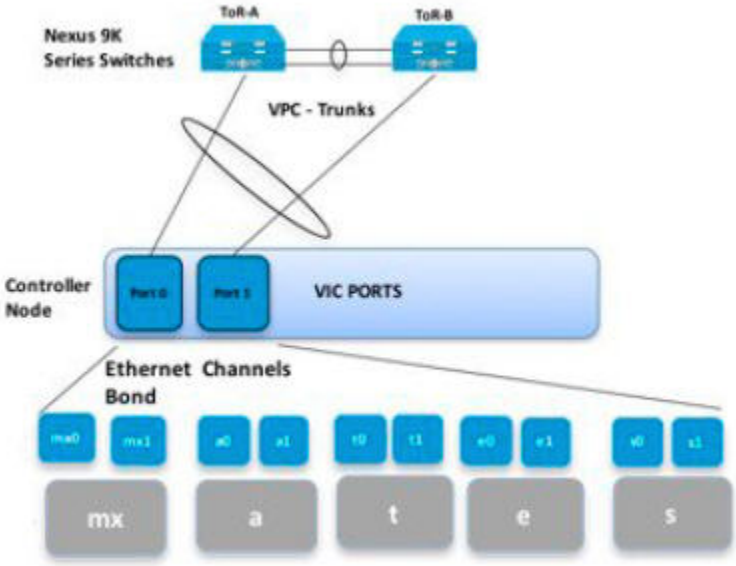
Cisco VIM 3.4.6 uses Dell PowerEdge 740 (with 2x1.2TB HDD) as compute with UCS M5 as control and ceph nodes in a full pod environment. In this configuration, four Intel XXV710 NIC cards are present in the compute with VPP/VLAN network. The cloud control and data plane are running on dedicated NIC cards, while the SRIOV is running on two additional NICs as depicted below:



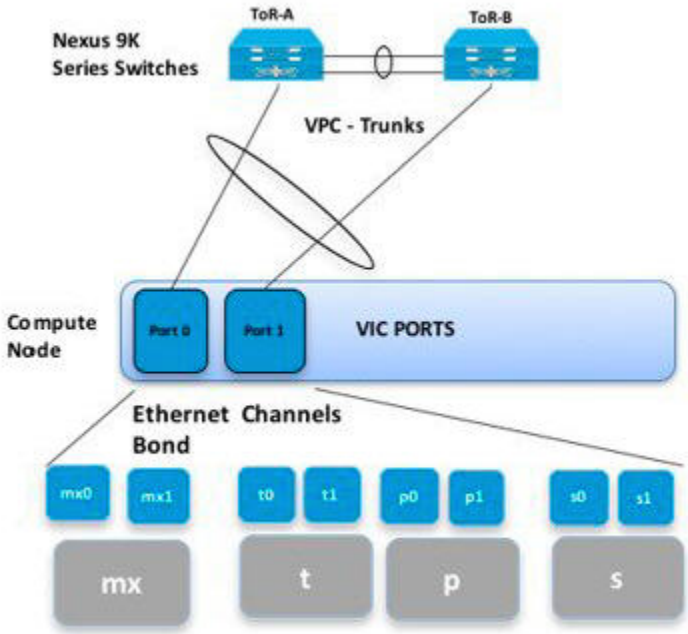
The Cisco NFVI controller node has four bonds: mx, a, t, and e. Each of them has a underlying interface that is named with the network name association and a mapped number. For example, the management and provisioning network, mx, maps to mx0 and mx1, the API network, a, to a0 and a1, and so on. The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

Cisco VIM manages a third-party infrastructure based on Quanta servers, thereby bringing in true software abstraction. In the implementation, the supported deployment is a full-on or edge pod, with OVS as the mechanism driver. With the power limitation and rack restrictions on the edge pod, it cannot support hard-drives for the Ceph service. As the Edge pod does not need persistent storage, it is designed to communicate with a central ceph cluster for providing glance image services only.
The installation and management of the Central Ceph cluster is fully automated and it is assumed that the management network of the edge cloud is routable to that of the central Ceph cluster.
In the case of Quanta servers, the networking is similar to that of the HP computes except for the two port 25G (xxv710) Intel NICs. The 2x25GE OCP card is used for control and data plane network over virtio, and the two additional 25GE 2-port xxv710 based Intel NIC Adapters are used for SRIOV via the provider network.

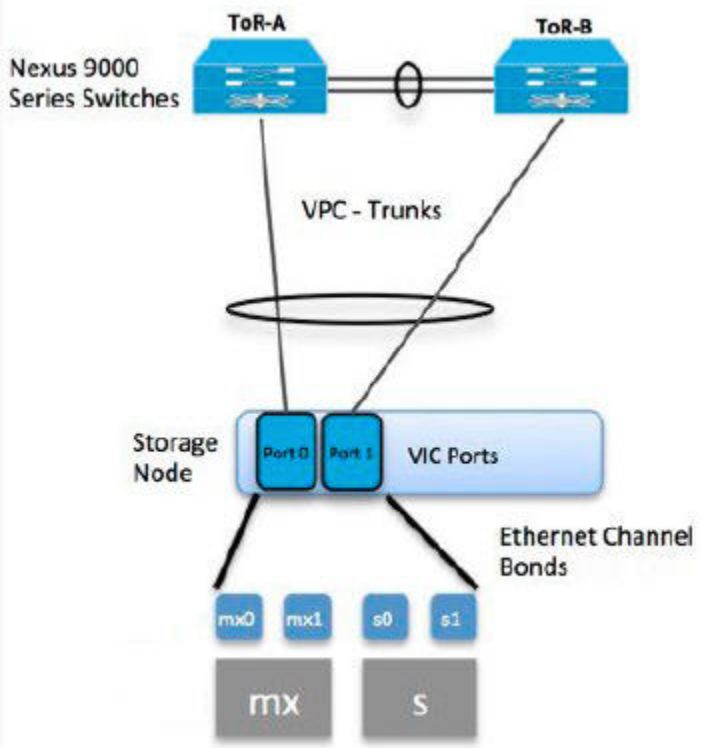
The following figure shows the controller node network-to-bond-to-vNIC interface mapping.



The Cisco 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.

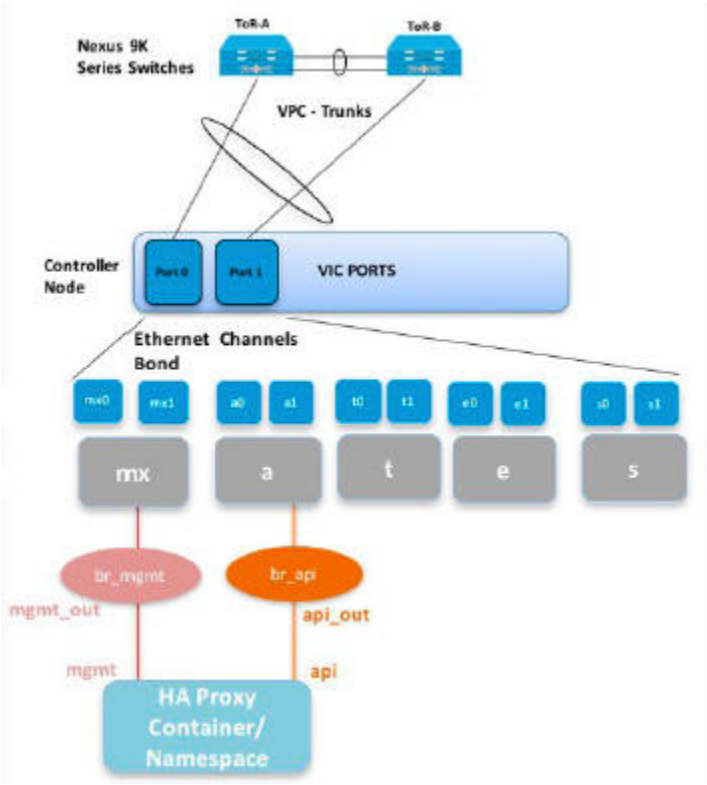


The Cisco NFVI storage node has two bonds: mx and s. Each has an underlying interface that is named with the network name association and a mapped number. For example, the storage network, s, maps to s0 and s1. Storage nodes communicate with other storage nodes over the mx network. The storage network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



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

The following figure shows the connectivity between the mx interface and the br_mgmt bridge, 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.



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

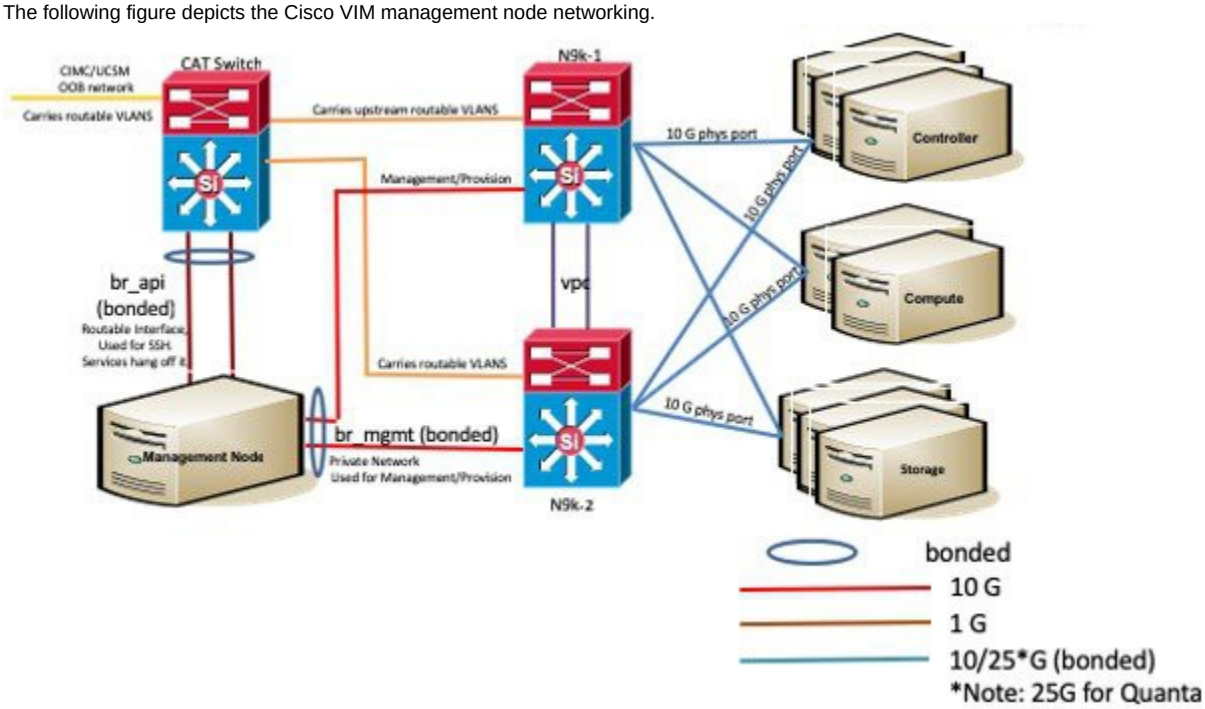
[root@c43-bot-mgmt ~]# ip addr show br_mgmt
7: br_mgmt: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
link/ether 58:ac:78:5c:e4:95 brd ff:ff:ff:ff:ff:ff
inet 172.26.233.104/25 brd 172.26.233.127 scope global br_mgmt
valid_lft forever preferred_lft forever
inet6 fe80::403:14ff:fef4:10c5/64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

Management Node Networking

Management Node Networking

In Cisco VIM, the management node has an interface for API and another interface for provisioning. This is primarily done for security reasons, so that internal pod management or control plane messages (RabbitMQ, Maria DB, and so on) 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 Cisco VIM management node networking.



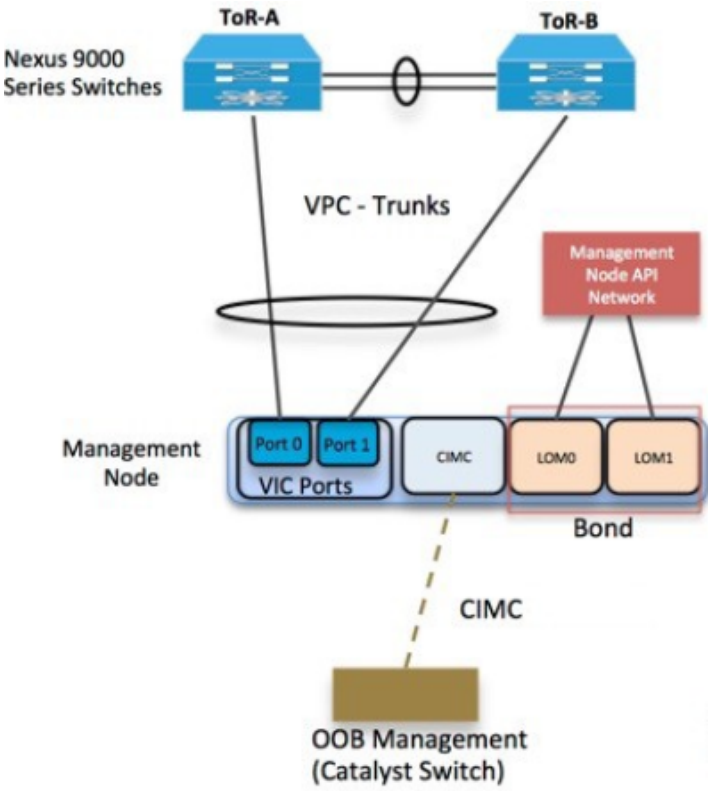
Cisco NFVI UCS C-series management node physically connects to the network. Unlike other nodes, the management node does not use multiple vNICs corresponding to specific Cisco NFVI networks. Instead, it connects to the management and API networks using two different physical connections. The management node connects to the management network using a Cisco two-port VIC or first two ports of intel X710-DA4, with each port connecting to a different ToR switch in a VPC configuration. The Cisco VIC card utilizes the default vNICs, but requires the vNICs to be in trunk mode and the default VLAN set to the management network VLAN.

The management node connects to the API network using both 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 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 Cisco NFVI management node (UCS C-series) API network connections.



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

- CIMC/ILO/BMC of the individual servers from the management node is available through the br_api network.
- Cloud API, external network (for ssh to floating IPs), and provider network from the management node is available, as the VMTP and NFVbench are typically run from the management node.

> ⚠ You can enable or disable the default behavior of the management node reachability from cloud API, external network, and provider network as part of their Day 0 configuration.

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

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

IPv6 Support

IPv6 Support on Management Network

As the number of available routable IPv4 networks is limited, Cisco VIM supports dual-stack environment. In a dual-stack environment, Cisco 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.

> ⚠ 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.

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 Cisco VIM accessibility to the external services over IPv6.

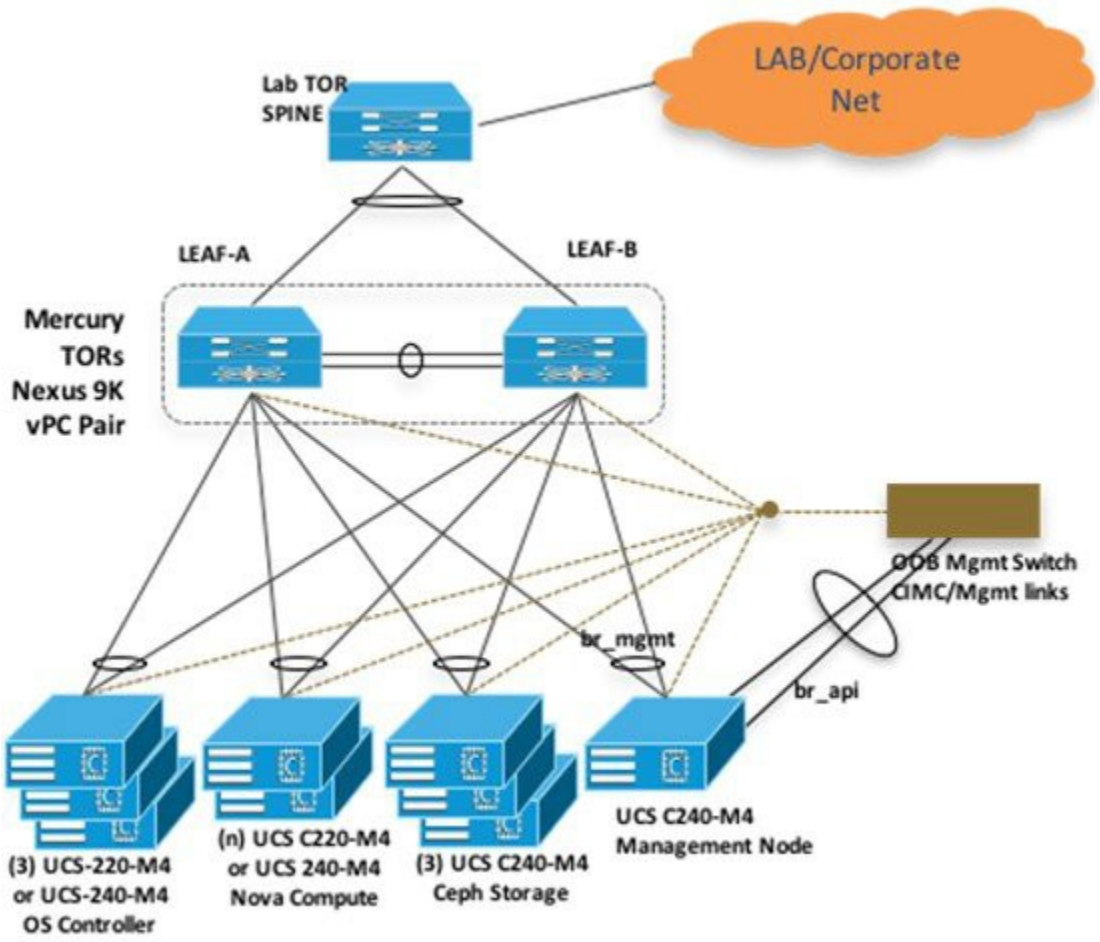
In Cisco VIM, the management network supports IPv6 addresses for servers, while the management node is statically allocated from a given pool. The external services that support both IPv4 and IPv6 addresses are DNS, NTP, and AD or LDAP. You can run IPv4+IPv6 (optionally) as the cloud API endpoint. CIMC/BMC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

UCS C-Series

UCS C-Series

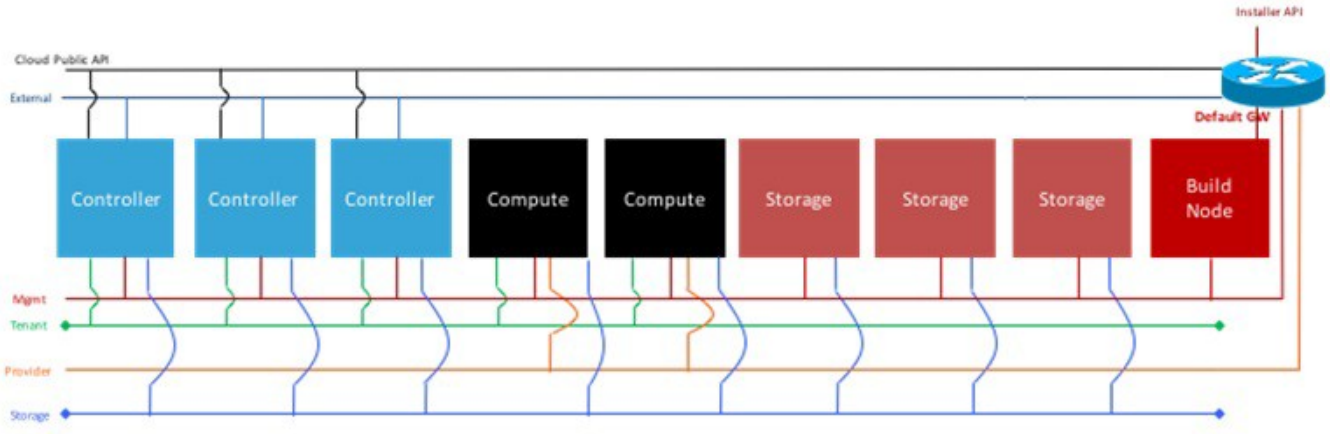
You can deploy Cisco NFVI using a combination of Cisco UCS C-series and or selected third-party rack servers. The C-series management node is connected to the Cisco Nexus 9000 Series ToRs through the Cisco VIC in a VPC configuration. The servers are connected to the ToRs as well. For C-series implementation,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



⚠ Here, UCS 220 M4 is used as the control/compute, but UCS 240 M4 can also be used as control and compute nodes.

Cisco NFVI uses multiple networks and VLANs to isolate network segments. For UCS C-series management and storage nodes, VLANs are trunked between the ToR switches and 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

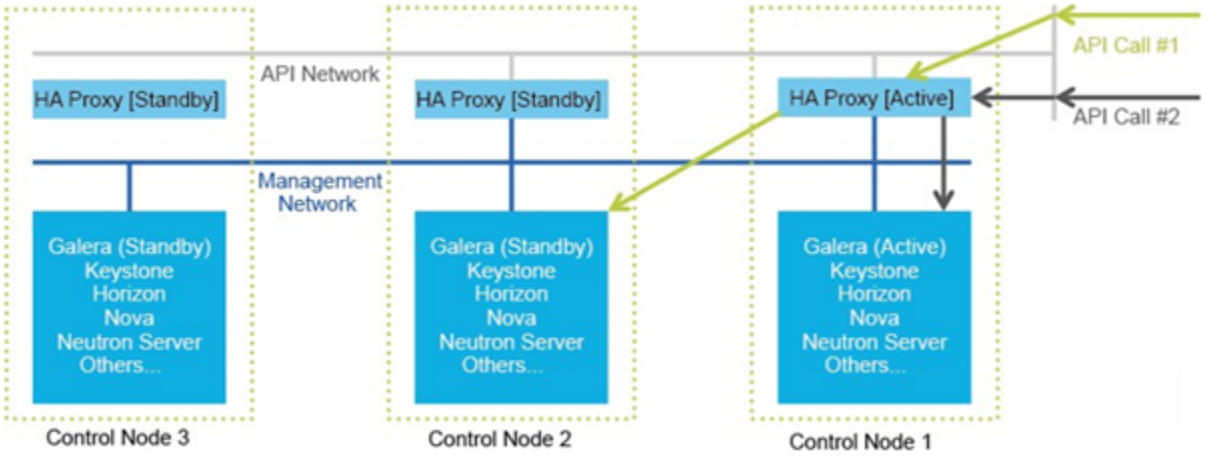
For B series, UCSM IP should be reachable from the management network.

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



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

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

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

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

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

While manual intervention is not needed, some specific failure scenarios (for example, Mariadb, rabbit) can cause problems that require manual intervention. For example, if a complete network failure occurs, the Galera and RabbitMQ clusters can go into three-way partition. While the Cisco NFVI cluster is resilient to single-point failures, two switches failing simultaneously—something highly unlikely in long-running systems—can sometimes happen due to administrative error, in which case, manual intervention is needed.

To repair the pod, the management node must be up and running and all the nodes accessible through password-less SSH from the management node. From the *installer<tagid>* directory, execute:

```
ciscovim cluster-recovery
```

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

```
nova service-list*
```

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

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

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

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

Storage Node Overview

Storage Node Overview

- Block Storage
- ObjectStore
- Third-Party Integration
    - NetApp
    - SolidFire
    - Zadara

Block Storage

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

Cisco NFVI storage nodes include object storage devices (OSDs): 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. The objects reside in a flat address space where they all exist at the same level and cannot be placed inside one another. Each OSD has a unique object identifier (OID) that allows the Cisco NFVI control node to retrieve it without knowing the physical location of the data it contains.

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

SSDs are solid-state storage devices that use integrated circuit assemblies as memory to store data persistently. SSDs primarily use electronic interfaces compatible with traditional block input/output (I/O) hard disk drives, which permit simple replacements in common applications.
Cisco NFVI storage nodes are managed by the control node applications including Ceph monitoring dashboard, Glance, and Cinder. The Ceph monitoring dashboard provides a view into the overall storage node health. Glance virtualizes pools of block storage devices and provides a self-storage API to request and consume those resources. Cinder is an OpenStack block storage service designed to present storage resources to the OpenStack compute node.

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

ObjectStore

Cisco VIM provides an integration with the reference object storage solution of swift. In this case, the SolidFire deployment is a prerequisite where the iSCSI cluster is setup and its storage VIP (svip) is accessible from controller nodes (iSCSI TCP port 3260). Also, the volumes are pre-created and the authentication for volume access is captured. For more details, see Swift Integration.

Third-Party Integration

NetApp

Cisco VIM supports NetApp devices running ONTAP 9.X or higher. NetApp devices are alternative to Ceph for block storage. Cisco 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 Cisco VIM.

SolidFire

In Cisco VIM, you can choose SolidFire as an option for block storage along with Ceph. In this scenario, the backend for Glance is Ceph. The Cinder block storage service manages the creation, attachment, and detachment of these volumes between a storage system, such as SolidFire and different host servers. The SolidFire cluster is pre-deployed with two networks: management and storage. It is recommended that:

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

For details on enabling SolidFire, see Enabling SolidFire

Zadara

The Zadara Virtual Private Storage Array (VPSA) is a software-defined solution, that is available as a Storage-as-a-Service with the storage servers residing in the customer premise. It is an elastic system that provides Enterprise-grade data protection and data management storage services. Cisco VIM provides Day 0 seamless integration with Zadara (pre-deployed ahead of time), and hence enables the following value-added services associated with it:

- Enterprise quality, resilient, highly available, and consistent performance storage for the most demanding data center application workloads.
- Consumed as a service - flexible, dynamic and billable.
- Scale out - to hundreds of storage nodes, thousands of drives, and multi-petabyte storage.
- True multi-tenancy - End-user controlled privacy and security. Separate workloads, resource allocation, and management per tenant such that each tenant truly experiences secure storage with no noisy neighbor.
- Universal storage - Supports all data services on one common infrastructure: Block, File, Object.

For details on enabling Zadara, see Enabling Zadara

OpenStack Telemetry Service

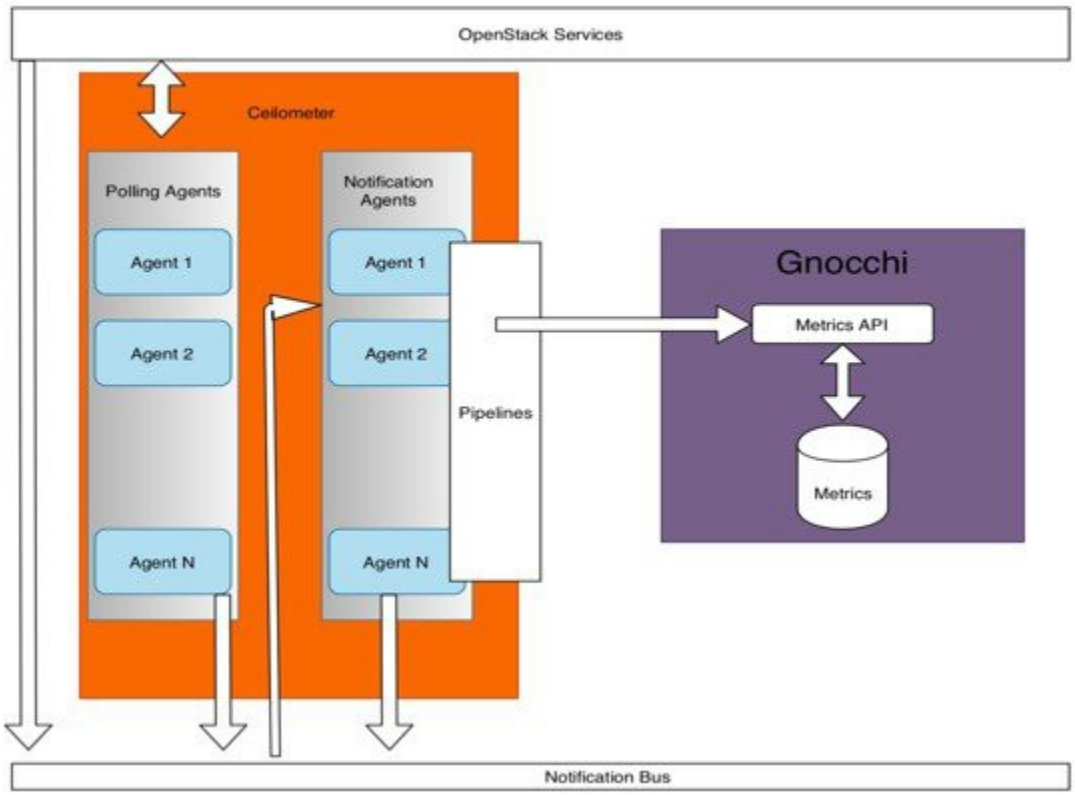
OpenStack Telemetry Service

- Overview
- OpenStack Compute
- OpenStack Image
- OpenStack Block Storage
- Metrics Polling and Retention Policy

Overview

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

The following figure illustrates the high-level architecture of the telemetry services.



If ceilometer is enabled in *setup_data.yaml* file, the Telemetry service in Cisco VIM collects the metrics within the OpenStack deployment.

- Once ceilometer is added as Day 0 configuration on a fullon pod, you can only deploy the pod but cannot be uninstalled.
- If ceilometer is deployed and metrics are generated, CEPH status indicates HEALTH_WARN. This is normal as ceilometer metric pool have too many objects per placement group than average value. However, this specific warning does not block any pod management operation.

The following sections provide a summary of the metrics that are collected with the Cisco VIM using ceilometer/gnocchi OpenStack REST API.

OpenStack Compute

The following metrics are collected for OpenStack Compute:

Name	Type	Unit	Resource	Origin	Note
memory	Gauge	MB	instance ID	Notification	Volume of RAM allocated to the instance

memory.usage	Gauge	MB	instance ID	Pollster	Volume of RAM used by the instance from the amount of its allocated memory
cpu	Cumulative	ns	instance ID	Pollster	CPU time used
cpu.delta	Delta	ns	instance ID	Pollster	CPU time used since previous datapoint
cpu_util	Gauge	%	instance ID	Pollster	Average CPU utilization
vcpus	Gauge	vcpu	instance ID	Notification	Number of virtual CPUs allocated to the instance
disk.read.requests	Cumulative	request	instance ID	Pollster	Number of read requests
disk.read.requests.rate	Gauge	request/s	instance ID	Pollster	Average rate of read requests
disk.write.requests	Cumulative	request	instance ID	Pollster	Number of write requests
disk.write.requests.rate	Gauge	request/s	instance ID	Pollster	Average rate of write requests
disk.read.bytes	Cumulative	B	instance ID	Pollster	Volume of reads
disk.read.bytes.rate	Gauge	B/s	instance ID	Pollster	Average rate of reads
disk.write.bytes	Cumulative	B	instance ID	Pollster	Volume of writes
disk.write.bytes.rate	Gauge	B/s	instance ID	Pollster	Average rate of writes
disk.device.read.requests	Cumulative	request	disk ID	Pollster	Number of read requests
disk.device.read.requests.rate	Gauge	request/s	disk ID	Pollster	Average rate of read requests
disk.device.write.requests	Cumulative	request	disk ID	Pollster	Number of write requests
disk.device.write.requests.rate	Gauge	request/s	disk ID	Pollster	Average rate of write requests
disk.device.read.bytes	Cumulative	B	disk ID	Pollster	Volume of reads
disk.device.read.bytes .rate	Gauge	B/s	disk ID	Pollster	Average rate of reads
disk.device.write.bytes	Cumulative	B	disk ID	Pollster	Volume of writes
disk.device.write.bytes.rate	Gauge	B/s	disk ID	Pollster	Average rate of writes
disk.root.size	Gauge	GB	instance ID	Notification	Size of root disk
disk.ephemeral.size	Gauge	GB	instance ID	Notification	Size of ephemeral disk
disk.capacity	Gauge	B	instance ID	Pollster	The amount of disk that the instance can see
disk.allocation	Gauge	B	instance ID	Pollster	The amount of disk occupied by the instance on the host machine
disk.usage	Gauge	B	instance ID	Pollster	The physical size in bytes of the image container on the host
disk.device.capacity	Gauge	B	disk ID	Pollster	The amount of disk per device that the instance can see
disk.device.allocation	Gauge	B	disk ID	Pollster	The amount of disk per device occupied by the instance on the host machine
disk.device.usage	Gauge	B	disk ID	Pollster	The physical size in bytes of the image container on the host per device
network.incoming.bytes	Cumulative	B	interface ID	Pollster	Number of incoming bytes
network.incoming.bytes.rate	Gauge	B/s	interface ID	Pollster	Average rate of incoming bytes
network.outgoing.bytes	Cumulative	B	interface ID	Pollster	Number of outgoing bytes
network.outgoing.bytes.rate	Gauge	B/s	interface ID	Pollster	Average rate of outgoing bytes
network.incoming.packets	Cumulative	packet	interface ID	Pollster	Number of incoming packets
network.incoming.packets.rate	Gauge	packet/s	interface ID	Pollster	Average rate of incoming packets
network.outgoing.packets	Cumulative	packet	interface ID	Pollster	Number of outgoing packets
network.outgoing.packets.rate	Gauge	packet/s	interface ID	Pollster	Average rate of outgoing packets
network.incoming.packets.drop	Cumulative	packet	interface ID	Pollster	Number of incoming dropped packets
network.outgoing.packets.drop	Cumulative	packet	interface ID	Pollster	Number of outgoing dropped packets
network.incoming.packets.error	Cumulative	packet	interface ID	Pollster	Number of incoming error packets
network.outgoing.packets.error	Cumulative	packet	interface ID	Pollster	Number of outgoing error packets
memory.swap.in	Cumulative	MB	interface ID	Pollster	Memory swap in
memory.swap.out	Cumulative	MB	interface ID	Pollster	Memory swap out
disk.device.read.latency	Cumulative	ns	Disk ID	Pollster	Total time read operations have taken

OpenStack Image

The following metrics are collected for OpenStack Image service:

Name	Type	Unit	Resource	Origin	Note
image.size	Gauge	B	image ID	Notification, Pollster	Size of the uploaded image
image.download	Delta	B	image ID	Notification	Image is downloaded
image.serve	Delta	B	image ID	Notification	Image is served out

OpenStack Block Storage

The following metrics are collected for OpenStack Block Storage:

Name	Type	Unit	Resource	Origin	Note
volume.size	Gauge	GB	Voulume ID	Notification	Size of the volume

Metrics Polling and Retention Policy

Cisco VIM telemetry service polls metrics every 5 minutes and retains the metrics for 48 hours.

For more details, see Heat and Ceilometer

NFVBench

NFVBench

- Setting up NFVbench
- Encapsulation
- Cisco VIM CLI

Setting up NFVbench

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
- VXLAN Fixed Rate

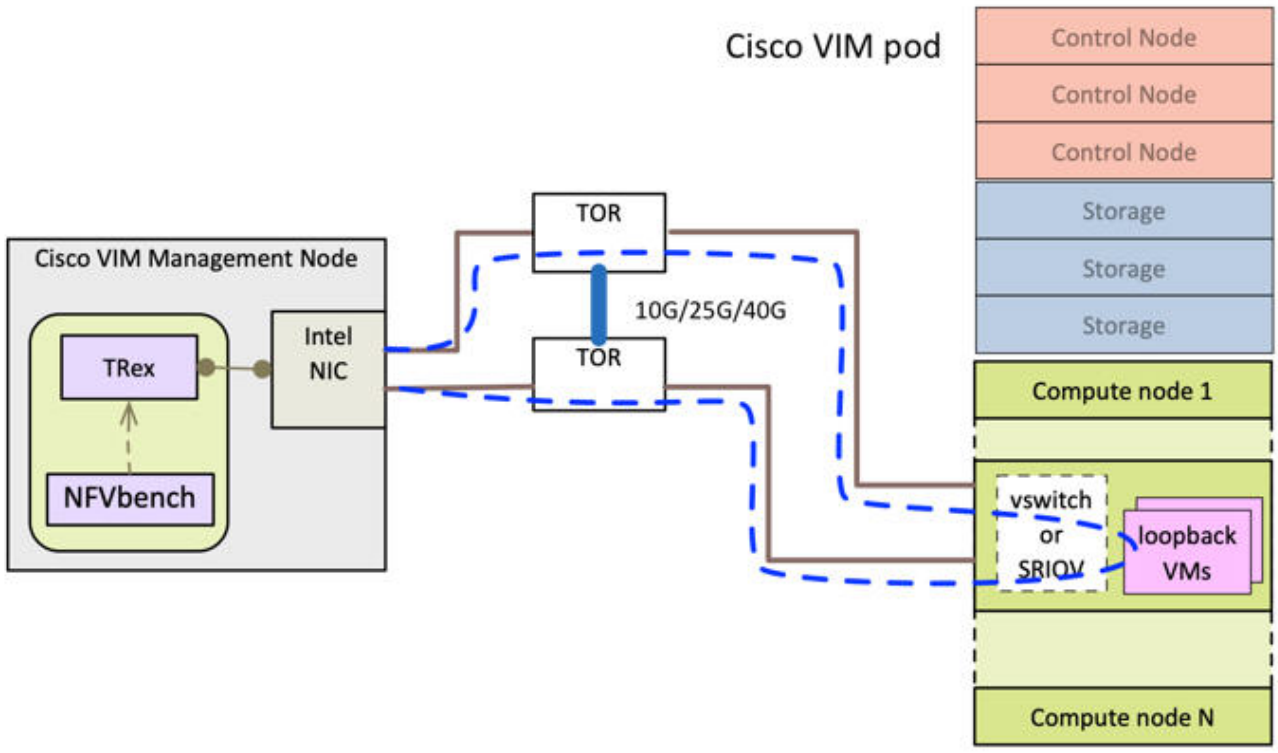
Overview

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:

- 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.

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 Cisco VIM data plane network options:

- OVS with VLAN
- VPP virtual switch with VLAN
- VPP virtual switch with VxLAN overlay
- 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 Cisco VIM configuration file using the following command:

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

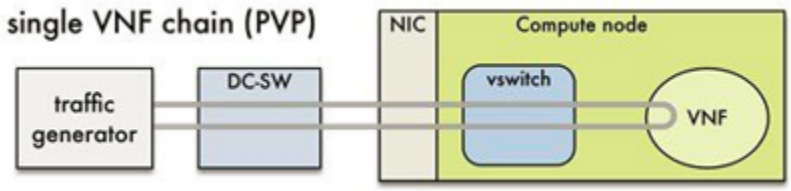
- The tor_info option must specify the interface names on the ToRs where the two Intel NIC ports reserved for NFVbench are connected.
- NFVbench can be enabled post deployment using a reconfigure operation.

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.

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

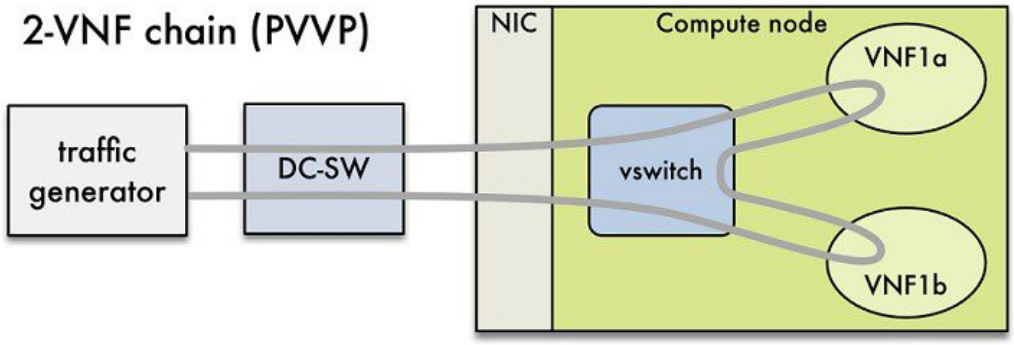
The performance of the PVP packet path provides a very good indication of the capabilities and efficiency of the NFVI data plane in the case of a single service chain made of one VNF/VM.

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 packet path having 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
```

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 passed 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.

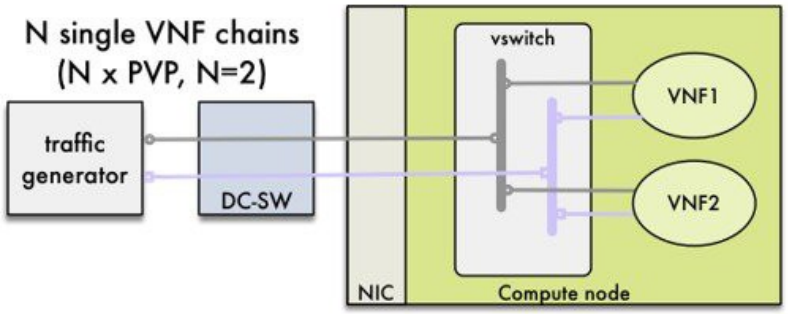
The following command stages only the default PVP packet path, but does not generate any traffic:

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

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 a 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.

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 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 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.

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 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
```

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

VLAN Fixed Rate

The following example shows 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: 2018-12-19 21:26:26
NFVBench version 3.0.4.dev2
Openstack Neutron:
vSwitch: VPP
Encapsulation: VLAN
Benchmarks:
> Networks:
> Components:
> Traffic Generator:
Profile: trex-local
Tool: TRex
> Versions:
> Traffic_Generator:
build_date: Nov 13 2017
version: v2.32
built_by: hhaim
mode: STL
build_time: 10:58:17
> VPP: 18.07
> CiscoVIM: 2.4.3-15536
> Service chain:
> PVP:
> Traffic:
Profile: custom_traffic_profile
Bidirectional: True
Flow count: 10000
Service chains count: 1
Compute nodes: [u'nova:c45-compute-2']
```

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	0.0000%	28	20	330

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

1. Requested TX Rate is the rate that is requested in bps and pps.
2. 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.
3. 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	152.7344 Mbps	50,000 pps	50,000 pps	50,000 pps
Reverse	152.7333 Mbps	152.7334 Mbps	152.7344 Mbps	50,000 pps	50,000 pps	50,000 pps
Total	305.4667 Mbps	305.4668 Mbps	305.4688 Mbps	100,000 pps	100,000 pps	100,000 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 shows the number of packets sent from each port by the traffic generator.
- Vpp.RX.vlan.<id> shows the number of packets received on the VLAN sub-interface with VLAN id <id> in the VPP vswitch.
- Vpp.TX.veth/<id> shows the number of packets sent to the VM.
- Vpp.RX.veth/<id> shows the number of packets received from the VM.

The following table shows 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	3,000,001	=>	=>	=>	=>	3,000,001	28 usec	20 usec	320 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	3,000,001	=>	=>	=>	=>	3,000,001	28 usec	20 usec	330 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 NDR and PDR for IMIX, with the default chain type (PVP), default chain count (1) and default flow count (10k):

```
# nfvbench -fs IMIX
```

The summary of the NFVbench result execution is shown below:

```
Date: 2018-12-20 23:11:01
NFVBench version 3.0.5.dev2
Openstack Neutron:
vSwitch: VPP
Encapsulation: VLAN
Benchmarks:
> Networks:
        > Components:
                > Traffic Generator:
                        Profile: trex-local
                        Tool: TRex
        > Versions:
                > Traffic_Generator:
                        build_date: Nov 13 2017
                        version: v2.32
                        built_by: hhaim
                        mode: STL
                        build_time: 10:58:17
        > VPP: 18.07
        > CiscoVIM: 2.3.46-17358
        > 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: [u'nova:a22-mchester-micro-3']
```

The NFVBench result execution summary table shows 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	8.5352	2,794,136	0.0000%	124	10	245
PDR	IMIX	9.5703	3,133,012	0.0680%	167	10	259

VXLAN Fixed Rate

It is applicable for platforms that support VxLAN only.

Example 1:

In this example, the default frame size of 64B is sent over 1Mpps on two chains using VxLAN with flow count of 10k:

```
# nfvbench --duration 10 -scc 2 --rate 1Mpps --vxlan
```

The summary of the NFVBench result is shown below:

```
2018-12-20 23:28:24,715 INFO --duration 10 -scc 2 --rate 1Mpps --vxlan
2018-12-20 23:28:24,716 INFO VxLAN: vlan_tagging forced to False (inner VLAN tagging must be disabled)
2018-12-20 23:28:24,716 INFO Using default VxLAN segmentation_id 5034 for middle internal network
2018-12-20 23:28:24,716 INFO Using default VxLAN segmentation_id 5017 for right internal network
2018-12-20 23:28:24,716 INFO Using default VxLAN segmentation_id 5000 for left internal network
```

Example 2:

In this example, VxLAN benchmark is run and 64B frames are sent over 100kpps for the default duration.

```
# nfvbench -rate 100kpps --vxlan
2018-12-18 19:25:31,056 INFO VxLAN: vlan_tagging forced to False (inner VLAN tagging must be disabled)
2018-12-18 19:25:31,056 INFO Using default VxLAN segmentation_id 5034 for middle internal network
2018-12-18 19:25:31,056 INFO Using default VxLAN segmentation_id 5017 for right internal network
2018-12-18 19:25:31,056 INFO Using default VxLAN segmentation_id 5000 for left internal network
```

The NFVBench result summary is as follows:

```
Date: 2018-12-18 19:26:40
NFVBench version 3.0.5.dev2
Openstack Neutron:
vSwitch: VPP
Encapsulation: VxLAN
Benchmarks:
> Networks:
        > Components:
                > Traffic Generator:
                        Profile: trex-local
                        Tool: TRex
                > Versions:
                        > Traffic_Generator:
                                build_date: Nov 13 2017
                                version: v2.32
                                built_by: hhaim
                                mode: STL
                                build_time: 10:58:17
                > VPP: 18.07
                > CiscoVIM: 2.3.46-17358
        > Service chain:
                > PVP:
                        > Traffic:
                                Profile: traffic_profile_64B
                                Bidirectional: True
                                Flow count: 10000
                                Service chains count: 1
                                Compute nodes: [u'nova:a22-mchester-micro-1']
```

The following table shows the NFVBench result summary:

L2 Frame Size	Drop Rate	Avg Latency (usec)	Min Latency (usec)	Max Latency (usec)
64	0.0000%	0	nan	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)

Forward	33.6000 Mbps	33.6000 Mbps	33.6000 Mbps	50,000 pps	50,000 pps	50,000 pps
Reverse	33.6000 Mbps	33.6000 Mbps	33.6000 Mbps	50,000 pps	50,000 pps	50,000 pps
Total	67.2000 Mbps	67.2000 Mbps	67.2000 Mbps	100,000 pps	100,000 pps	100,000 pps

The following table shows forward chain packet counters and latency:

Chain	TRex.TX.p0	vpp.RX.vxlan_tunnel0	vpp.TX.veth/0	vpp.RX.veth/1	vpp.TX.vxlan_tunnel1	TRex.RX.p1
0	50,000	=>	=>	=>	=>	50,000

The following table shows reverse chain packet counters and latency:

Chain	TRex.TX.p1	vpp.RX.vxlan_tunnel1	vpp.TX.veth/1	vpp.RX.veth/0	vpp.TX.vxlan_tunnel0	TRex.RX.p0
0	50,000	=>	=>	=>	=>	50,000

Encapsulation

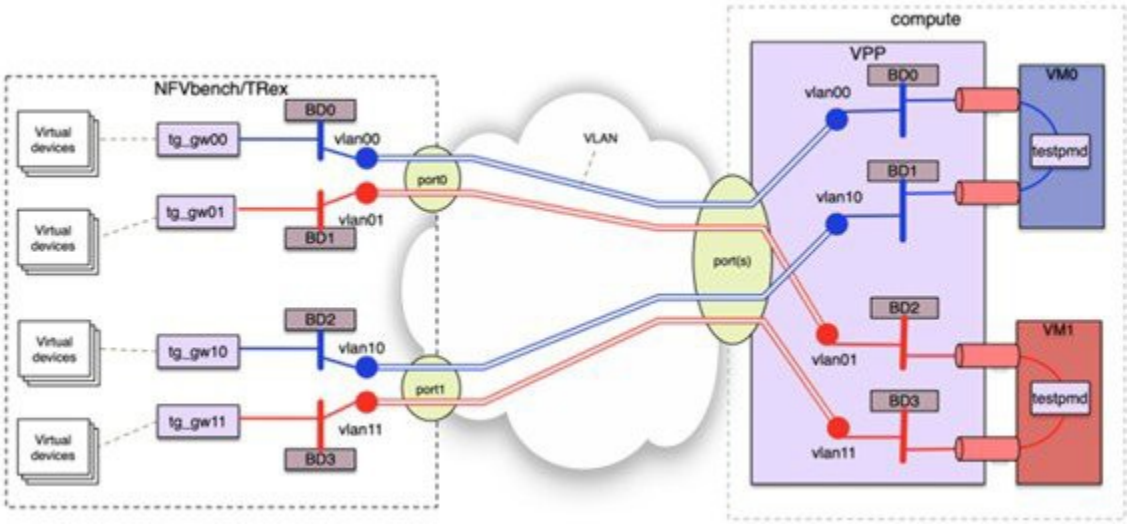
Encapsulation

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

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

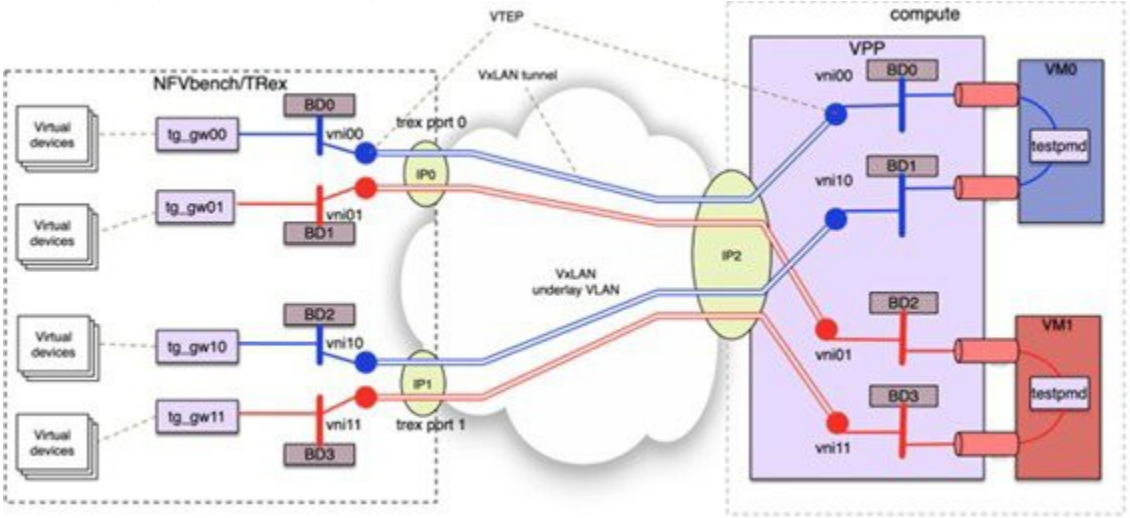
By default, NFVBench uses VLAN tagging for the generated traffic and directs the traffic to the vswitch in the target compute node (OVS or VPP).

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



If VxLAN is enabled, it is possible to force the use of VxLAN using the –vxlan CLI option.
The provision of custom configuration allows you to specify more VxLAN options such as specific VNIs to use. For more details, check the default configuration file.

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



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 knows 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 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 generally 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_sriov1` using respective segmentation_id 2000 and 2001:

```
sriov: true
internal_networks:
left:
segmentation_id: 2000
physical_network: phys_sriov0
right:
segmentation_id: 2001
physical_network: phys_sriov1
```

The segmentation ID fields must be different.
In 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.

Cisco VIM CLI

Cisco VIM CLI

An alternate way to NFVBench CLI is to use ciscovimclient. Ciscovimclient provides an interface that is more consistent with the Cisco 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:

```
[root@mgmt1 ~]# ciscovim 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:

```
[root@mgmt1 ~]# ciscovim 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                          |
+------------------+--------------------------------------+
```

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

```
[root@mgmt1 ~]# ciscovim 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 Cisco VIM (that is the REST client must run on the management node). For more details on the REST interface calls, see Cisco VIM REST API > *Cisco VIM REST API Resources.*

Auto-ToR Configuration via ACI API

Auto-ToR Configuration via ACI API

While the use of ACI plugin brings in the flexibility of dynamic allocation of tenant and provider VLANs on demand, it also ties the OVS version to the ACI plugin. This leads to an extreme tight coupling of Cisco VIM and ACI.

With an APIC plugin, there are might be gaps to cover certain use-cases, for example, where there is a need to have flexibility of different access types (tagged or non-tagged) for the same VLAN but for different servers.

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

With this option, the *setup_data* for each Cisco VIM instance is the single source for the server to switch port mappings. This solution can handle switch provisioning with the correct VLANs during the addition or removal of server and provider/tenant VLAN range expansion via reconfiguration option. The solution is based on the fact that the PV (port VLAN) count in a given ACI fabric domain is under the scale limits 10000 PV/ToR and 450000 PV/fabric.

NCS-5500 as ToR Option

NCS-5500 as ToR Option

Cisco VIM supports NCS-5500 as an alternate to a Nexus ToR. NCS-5500 is an IOS XR-based router which is similar to Nexus switches. You can use the 48 10/25G ports or the 6 40/100G uplink ports model to implement NCS-5500 (port-numbers depend on NCS version). Also, other SKUs of NCS-5500 are supported as long as the NCS-5500 software supports the EVLAG feature.

NCS-5500 uses the technology of bridge domain to connect to the server. 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

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

- Management node.
- Specific or all controller servers.
- Specific or all compute servers.

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 Cisco 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 computes.

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 ceph cluster from operation and then add it back through remove-storage and add-storage options in ciscovim. 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 Cisco 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

Cisco 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 RAM: 41+Vr GB	n/a	n/a
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 Cisco VIM:

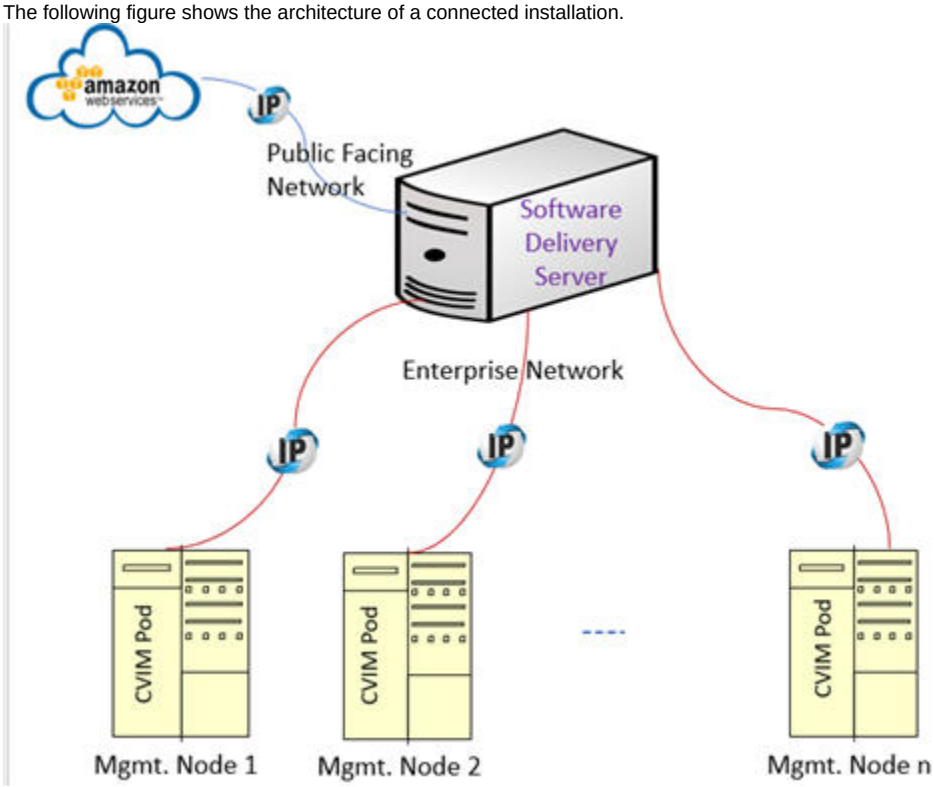
Variables	Usage	Valid range	Default
Q	Cores reserved for (host system and OpenStack Services)	2 to 12	2
C	Cores reserved for CEPH (AIO and HC)	2 to 12	2
V	Cores reserved for VPP vswitch	2 to 6	2
Vr	RAM reserved for VPP		2GB

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

Cisco VIM Software Hub

Cisco VIM is supported in an air-gapped (disconnected mode) environment. You can use a USB or Cisco VIM Software Hub for an air-gapped install. When the number of pods is more, shipping USBs for an air-gapped install and update is not scalable. In such scenarios, we recommend that you use Cisco VIM Software Hub.
Cisco VIM Software Hub contains the Cisco VIM release artifacts such as buildnode ISO, Cisco VIM code, docker registry, and docker images. Using the management node, you can access the release artifacts from the Cisco VIM Software Hub.
You can install the artifacts available on the Cisco VIM Software Hub server through a connected or a disconnected installation procedure. For a connected installation, one end of the Cisco 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.



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

VXLAN EVPN Design

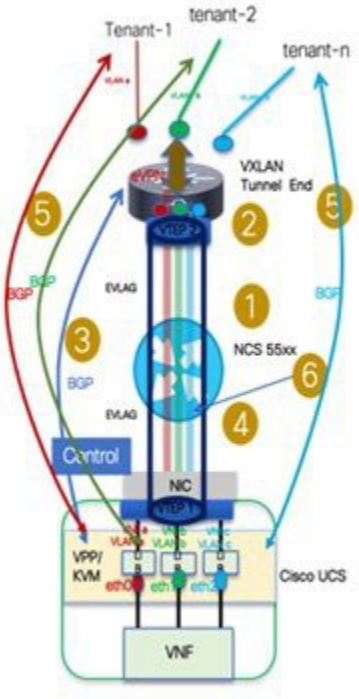
VXLAN EVPN Design

- Overview
- Multi-VXLAN EVPN Design

Overview

From release Cisco VIM 2.4.3 seamless connectivity from VNFs of the private cloud to the customer premise private cloud is enabled.

The architecture of the Cisco VIM Tenant L2 connectivity is shown below:

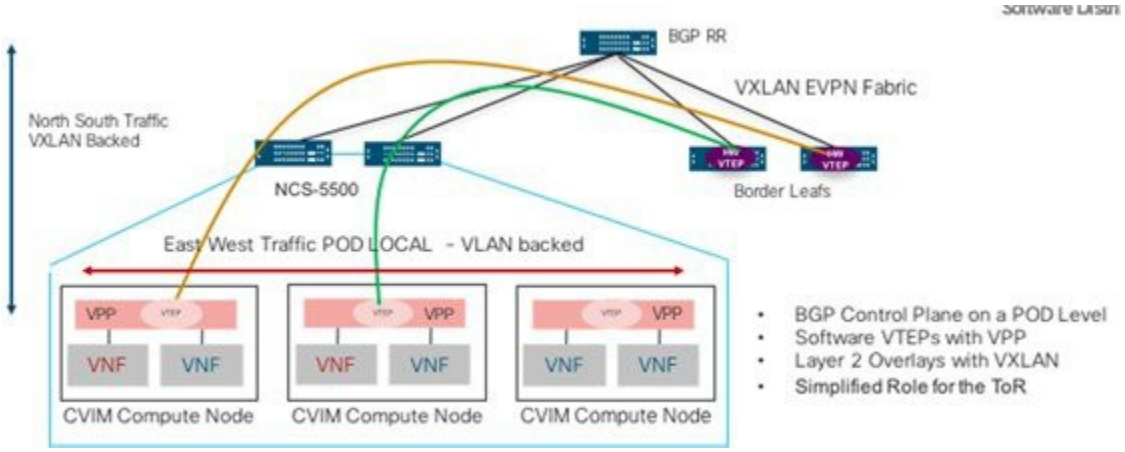


- **Segment 1: Compute to Layer 2 Aggregation (Layer 2 EVLAG)**
  - VPP will establish EVPN based EVLAG with aggregation layer
- **Segment 2: Layer 2 Aggregation switch to Fabric**
  - Fabric node will establish redundancy with aggregation layer
- **Segment 3: Infrastructure Control Plane**
  - EVPN 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

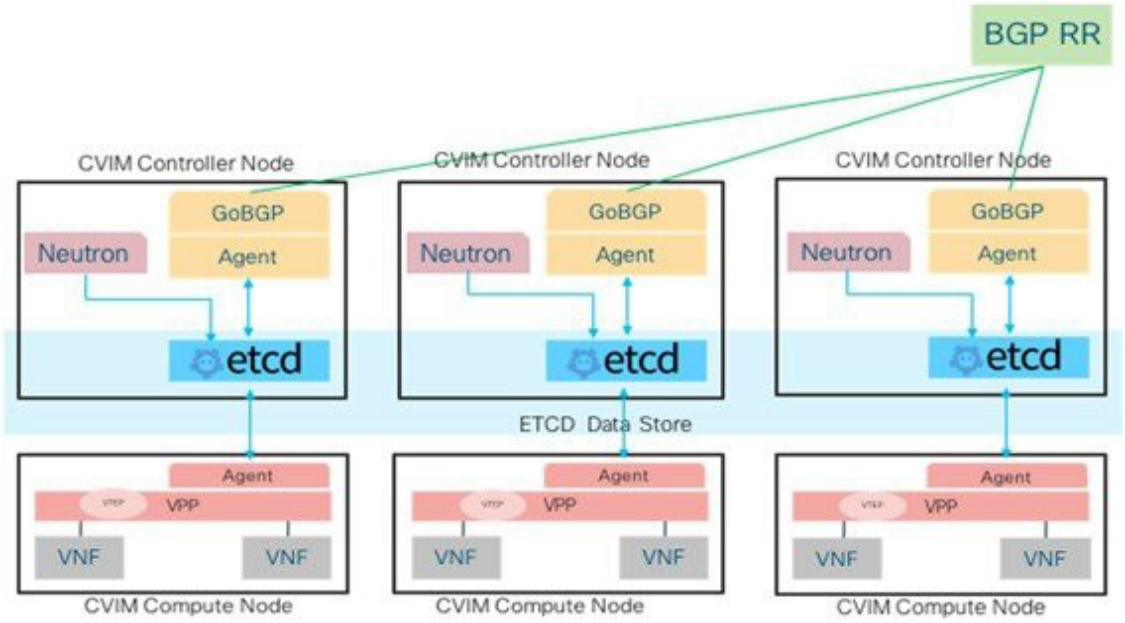
To set up Cisco VIM tenant L2 connectivity architecture, the following assumptions are made:

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

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

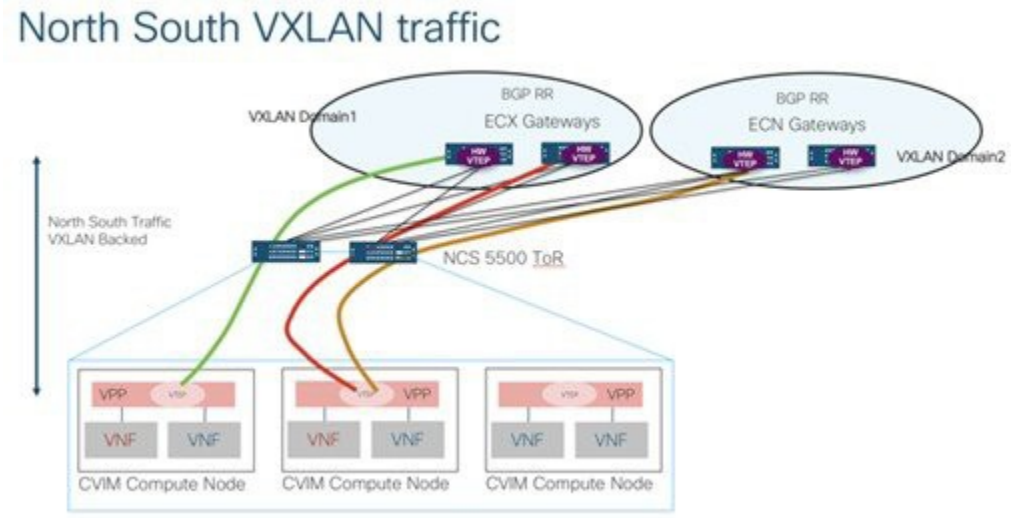


From a control plane point of view, three instances of GoBGP (in Active-Active-Active mode) run on the controller nodes to establish L3 peering with the external BGP RR for importing or exporting VxLAN routes into or from Cisco VIM respectively. The imported information is then pushed into etcd, to maintain a single source of the information within Cisco VIM.
VPP agents create and program VTEP on VPP, and also create a VXLAN tunnel interface for the VM based on the VNI information from Neutron. VPP updates VNF IP/MAC mapping in etcd, which gets exported out through EVPN to the BGP RR. The following figure shows the design of Cisco VIM VXLAN EVPN Control Plan



Multi-VXLAN EVPN Design

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



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

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

The multi-VXLAN EVPN based design optionally supports a static implementation of VXLAN technology through head-end replication (HER). HER helps leverage the VXLAN technology, regardless of the hardware/software limitation in the VXLAN feature set at the remote end of the VTEP tunnel.
With the static information defined in the setup_data, VPP performs the HER to all defined remote VTEPs and updates L2FIB (MAC-IP) table based on flood and learn. If EVPN co-exists with HER, Cisco VIM treats it as if two different sets of BGP speakers exist and provides information from each speaker in the same etcd FIB table.

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

VPP Port Mirroring Support

VPP Port Mirroring Support

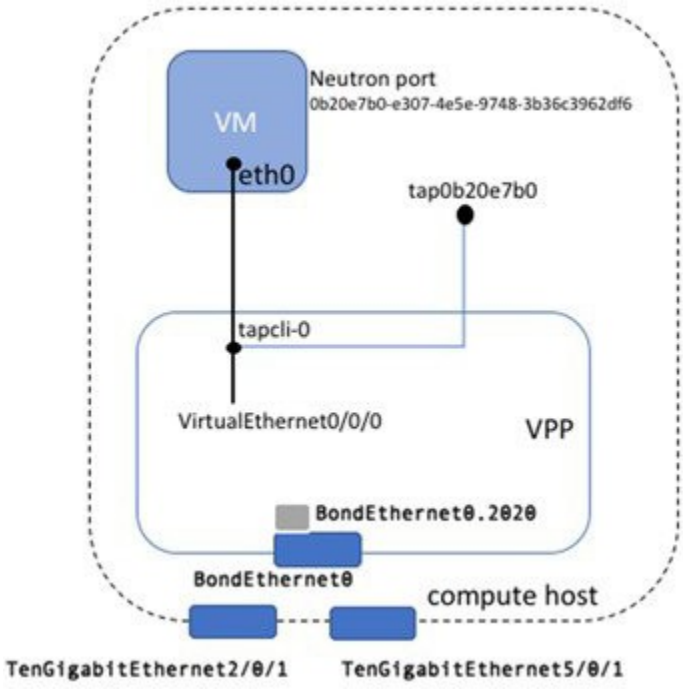
- Architecture
- Limitations

From release Cisco 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:

1. A span port on vpp to mirror the Virtual Ethernet interface corresponding to the VMs vhost interface. This is a tap interface in VPP.
2. A tap device (tap0b20e7b0) on the compute host side is set as a kernel interface. A veth pair is created between the tap device on the VPP side (tapcli-0) and kernel side tap device (tap0b20e7b0) as shown in the below figure.



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.

Segment Routing EVPN

Cisco VIM Segment Routing EVPN Design

- Overview of Segment Routing EVPN
- Overview of Cisco VIM VPP Architecture
- Overview of Cisco 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 VXLAN EVPN designs, it is possible to connect Cisco VIM with an existing Segment Routing (SR) Ethernet VPN (EVPN). You can connect Cisco 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 Cisco VIM pod to an SR EVPN, the existing VPP forwarding architecture is enhanced.

Overview of Cisco VIM VPP Architecture

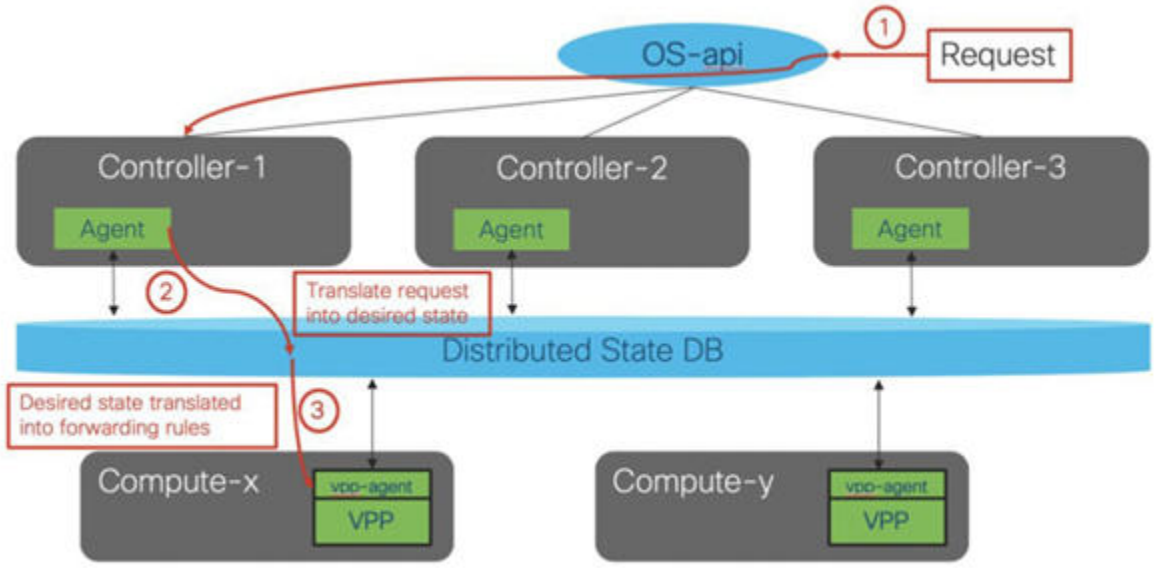
Networking-vpp is the Vector Packet Processing (VPP) based software accelerated virtual switch that is part of Cisco VIM. The architecture of networking-vpp is similar to a distributed SDN controller. However, it is not a separate controller and is integrated with Cisco VIM as a core component. Networking-vpp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



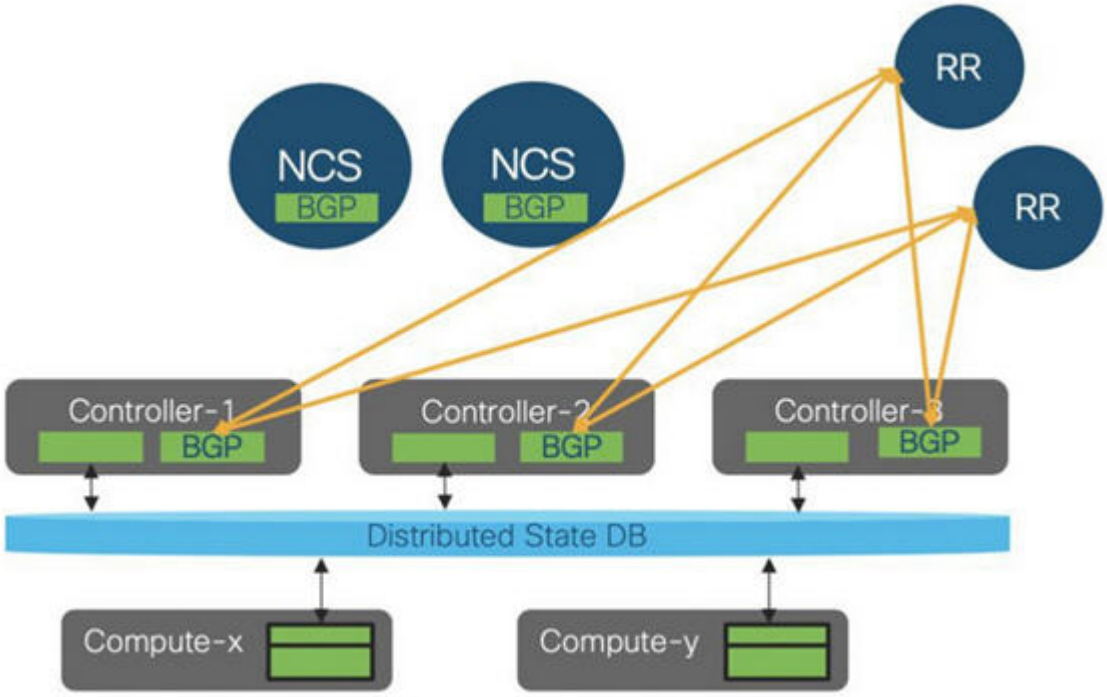
The following steps explain the figure give above:

1. 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.
2. The agent translates the request into the desired state of the network data plane and publishes this state into a distributed state database.
3. The vpp-agents running on the compute nodes watch the distributed state database. When they see a change relevant to the compute node, they translate the desired state and update the VPP forwarding on the compute node.

Overview of Cisco VIM SR EVPN Architecture

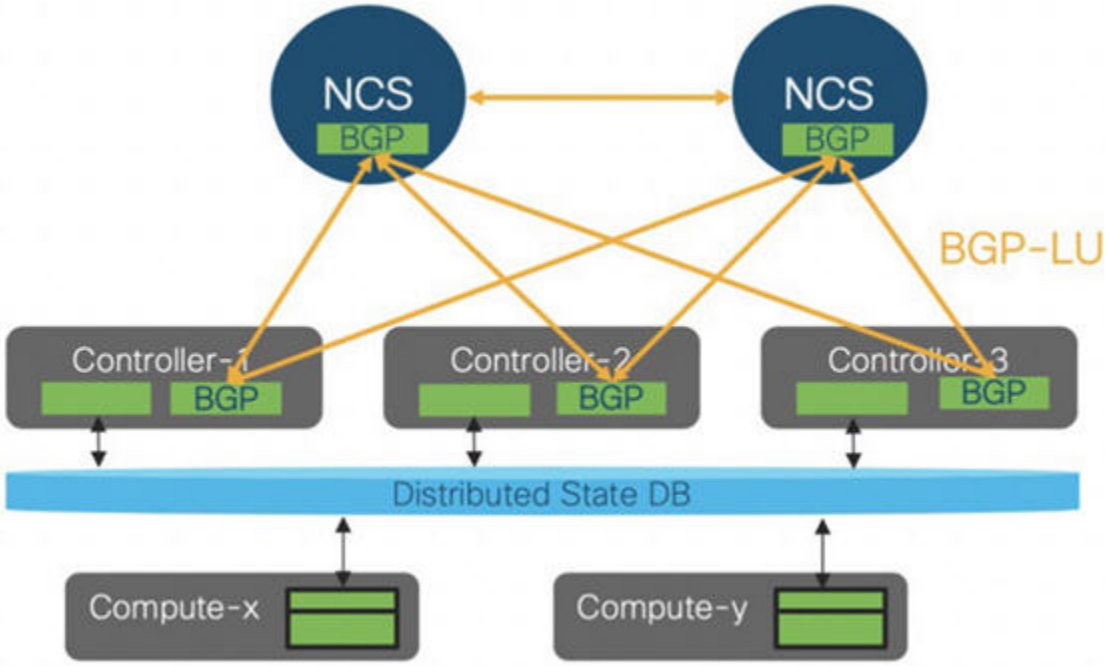
To extend the networking-vpp architecture to support Segment Routing over MPLS, Cisco VIM 3.4.1 adds several additional components.

The figure below depicts the SR EVPN RR connections.



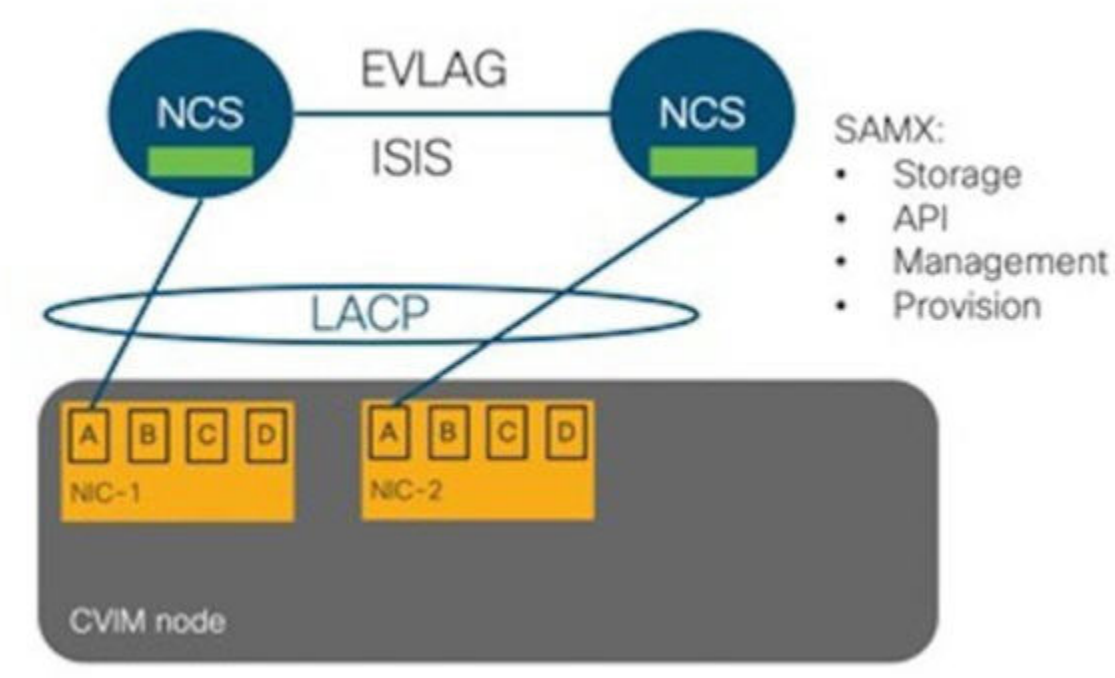
For the control plane, a BGP process runs on each of the three controller nodes. The controllers peer to the EVPN route reflectors. This peering allows the Cisco VIM pod to exchange reachability information with the EVPN. Cisco VIM does an L2 stretch with only single-homed L2 routes. The peering with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

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



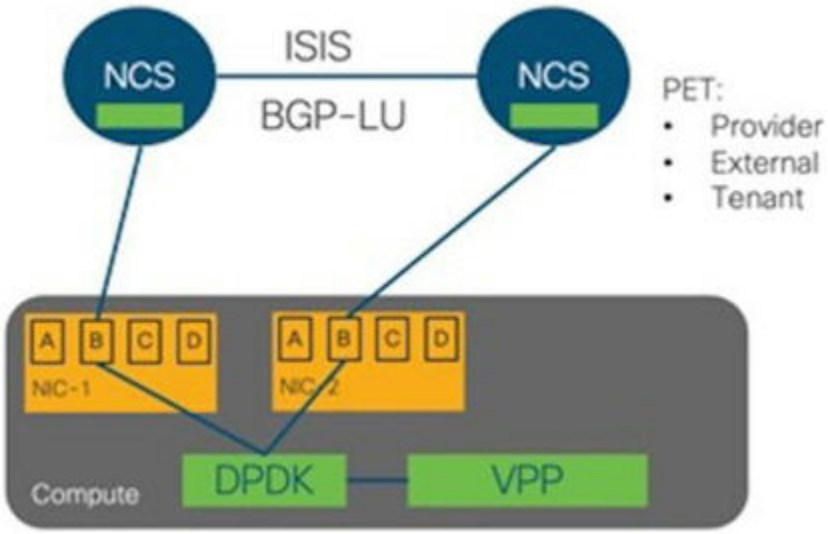
Cisco VIM does not process any SR policies coming from BGP or an SDN controller to program any explicit static or dynamic paths into VPP. BGP-LU and EVPN address family routes are processed by the BGP process running on each of the controllers. The BGP process publishes the SR policy into the distributed state database so that the VPP agents can configure VPP forwarding on the compute nodes to apply the MPLS label to the outbound traffic. The label between the compute node and the NCS ToR represents the path to node SID that identifies the VPP node. Bridge domain is identified by the Openstack Segmentation ID that is used for tenant isolation.
The data plane ports in the Cisco VIM pod are connected to the NCS with an L3 link and EVLAG provides link redundancy to the nodes. The VPP agents encapsulate the traffic leaving the compute node with an MPLS label representing the Segment Routing segment ID. The virtual machines are unaware that traffic will be sent over an SR-enabled network. For traffic arriving on the compute node from the NCS, the MPLS label represents the bridge domain, and the resulting L2 lookup determines the virtual machine to which to forward the traffic after stripping the MPLS header.

The figure below depicts the control plane port connectivity.



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

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 Cisco VIM node and an IP address on the NCS port. These addresses come from a /30 subnet. ISIS runs between the NCS and BGP-LU. On the Cisco VIM nodes, ECMP load balancing occurs over the two links to the NCS. On VPP, there is a loopback interface configured that is used as the SR node SID.
Ports C and D are SR-IOV ports that give the virtual machines direct access to the NIC through the VF interfaces. These are out of scope for the SR EVPN implementation on Cisco VIM.

Container Workload

Container Workload Support

Cisco VIM supports VM, baremetal, or container-based workloads. To support the container-based workloads, Cisco VIM hosts Cisco Container Platform as an application. The orchestrator creates a common OpenStack tenant and deploys the Cisco Container Platform control plane on it. The orchestrator can also create a tenant cluster if needed.
The Kubernetes clusters deployed are multi-master clusters with three master nodes and N worker nodes. The Cisco Container Platform control plane consists of three masters and three workers. The master and worker nodes run as VMs on OpenStack.

Management Node Centralization

Management Node Centralization

- Overview
- Terminology Used
- Architecture

Overview

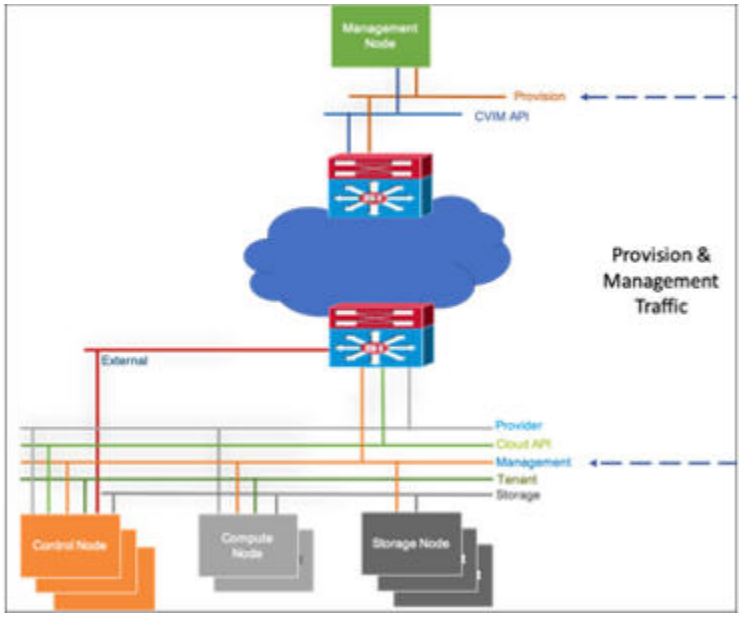
As dedicated management node is available for each pod, an additional cost of a server is involved with respect to space, power, and capital expenditure. As the cloud is moving to the edge, justifying the cost of a server on a per cloud basis becomes challenging. Hence, it is important that Cisco VIM evolves as a platform to support centralization of the management node. To support centralization, the management node is virtualized into a VM, which is then shipped as a QCOW2 image as part of the release. This concept has also been extended to Unified Management and Cisco VIM software hub.

Terminology Used

- Management node VM - A VM managing a pod.
- Management node - A baremetal managing a pod.
- Cisco VIM management pod - A pod hosting management node VMs. It can be a shared pod with other workloads.
- Locally managed pod - A pod that is managed by a local management node.
- Remotely managed pod - A pod that is managed remotely by a management node.

Architecture

The management node VM is Layer 3 adjacent to the pod as depicted in the figure below:



To support the architecture of management node centralization, the following prerequisites must be met:

• The cloud management network is routable from the management node VM.
• The management node VM must have access to BMC/CIMC.
• The pod servers must support remote-presence or equivalent tool, for example, VMCLI for Quanta.
• Solution uses iPXE to fetch kernel, initrd, and kickstart over httpd.

For detailed information and usage, see Centralizing Management Node

L3 Fabric Deployment

Cisco VIM in Layer 3 Fabric Deployment

To avoid pitfalls of traditional switching, modern data center network fabrics are built using Layer 3 (L3) routing like Equal-cost Multi-path Routing (ECMP), leaf-spine architecture, and overlay technologies like VxLAN.

From Cisco VIM 3.4.6, the VxLAN fabric is deployed to allow a single Cisco VIM pod to span across multiple pairs of Nexus 9000 ToR leaf switches that are attached to one L3 routed fabric (not ACI).

To support thousands of unique provider (p)/tenant (t) networks within a single Cisco VIM pod, use VPP as the virtual switch that encapsulates provider /tenant network traffic into VxLAN packets. The packets are then sent over the VLAN associated with Cisco VIM provider/tenant network segments. Thus, Cisco VIM compute nodes send VxLAN-over-VLAN frames to the ToR switches.

The ToR switches:

- Remove the VLAN tag and route the resulting VxLAN packet through the L3 fabric for handing North_South traffic.
- Direct the East-West traffic towards the destination compute node within the VLAN, when the compute nodes are attached to same ToR. The destination compute node receives VxLAN-over-VLAN frames and removes the VLAN tag before passing the VxLAN packet to VPP which in turn removes the VxLAN header and sends the packet to the VM.
- Encapsulate the incoming VxLAN-over-VLAN frame into a VxLAN packet corresponding to the provider/tenant network segment and then routes that VxLAN-over-VxLAN packet towards the destination ToR, when the compute nodes are attached to different pairs of ToRs connected via L3 fabric. The destination ToR removes the outer VxLAN header (corresponding to provider/tenant segment), and then switches the traffic towards the destination compute node within the VLAN. The destination compute node receives VxLAN-over-VLAN frames and removes the VLAN tag before passing the VxLAN packet to VPP which in turn removes the VxLAN header and sends the packet to the VM.

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:

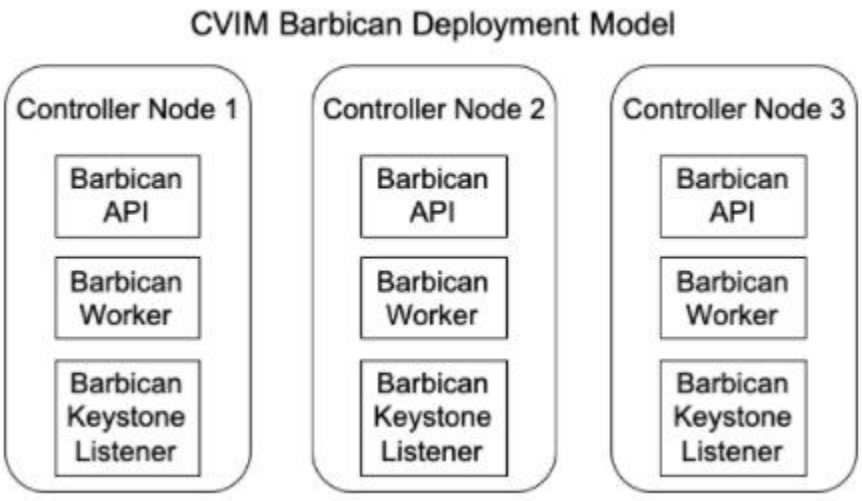
- Key Manager service overview
- OpenStack Barbican Architecture
- Configuring Secret Store Back-end
- Barbican API Documentation
- Volume encryption supported by the key manager

Barbican Architecture in Cisco VIM

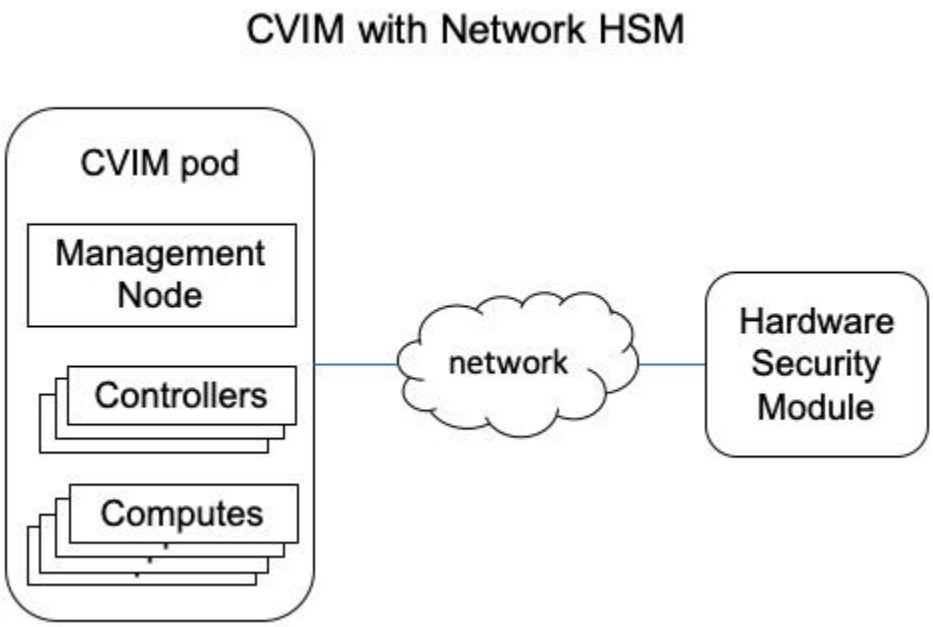
The following are the assumptions in Barbican architecture:

- Hardware Security Modules (HSM) exists in the network that is reachable by the Cisco VIM pod.
- Installation and management of the HSM are outside the scope of Cisco 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 Cisco VIM controller nodes via the management network of Cisco 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, Cisco VIM allows only one secret store back end to be configured at a time.

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



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



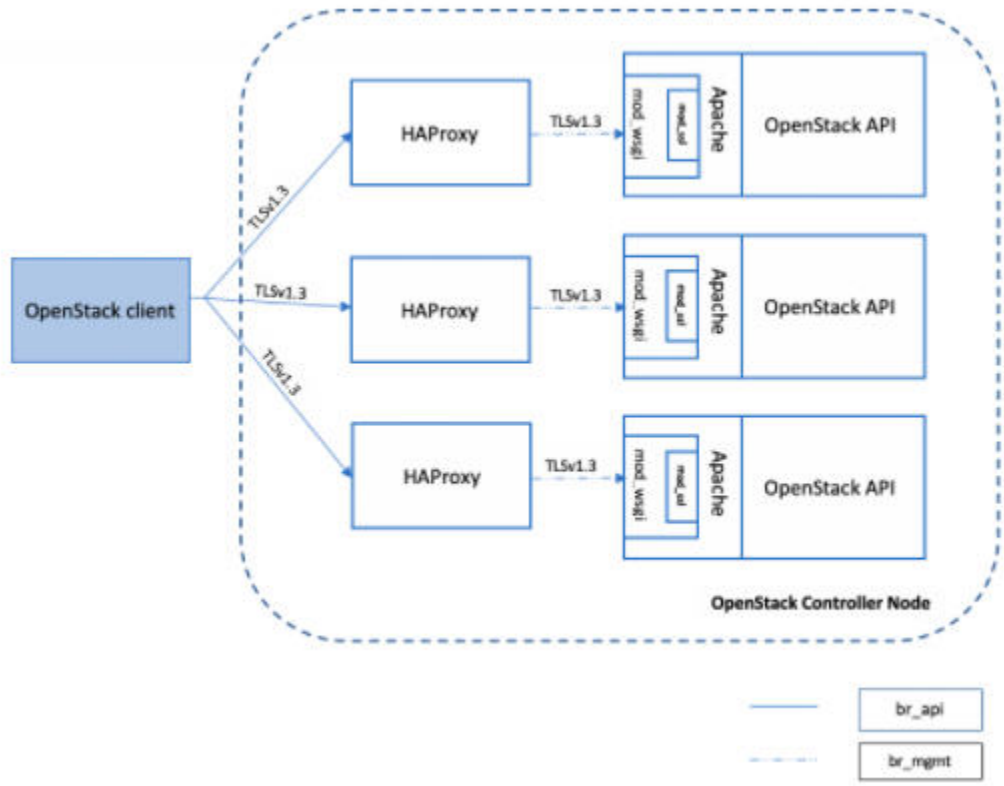
When Barbican is installed on a Cisco VIM pod, Cinder and Nova uses Barbican to store and retrieve volume encryption keys. Barbican is an optional service that can be included in the initial installation or added with a reconfigure operation. For details on how to enable Barbican see Enabling Barbican

Management Network Over TLS

Management Network Over TLS

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

From Cisco 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 Cisco 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 Cisco 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.

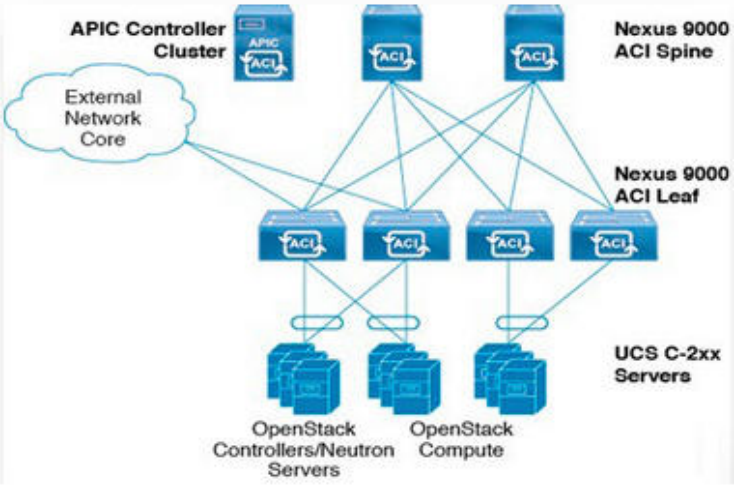
To build encrypted images, you must have your own infrastructure. Planning and implementation of mechanism to build encrypted images is outside the scope of Cisco VIM. As Cisco VIM is an embedded solution, Cisco neither recommends nor supports the installation of additional software on the servers managed by Cisco VIM. For more details, see Usage of Pre-encrypted Images.

ACI Plugin Integration

ACI Plugin Integration

In Cisco VIM 4.2.0, integration of ACI Plugin is supported with OSP 16.1. An ACI with OpenStack deployment consists of a Nexus 9000 Spine/Leaf topology, an APIC cluster, a minimum of three-node cluster of controllers that can act as the Neutron network node, and two or more compute nodes to host Virtual Machine (VM) instances. ACI external routed network connection is a Layer 3 connection outside the fabric. It is used to provide connectivity outside the OpenStack cloud.

The following figure shows an overview of a typical architecture of an ACI fabric with an OpenStack deployment.



The Opflex ML2 plugin integrated with Cisco VIM in Unified mode manages the tenant VLANs dynamically, as and when the VMs operate in the cloud. By utilizing OpFlex, the policy model native to ACI can be extended to virtual switches running on OpenStack Nova compute hosts. OpFlex extension to the compute host allows ACI to use Open vSwitch (OVS) to support common OpenStack features such as Source NAT (SNAT) and floating IP in a distributed manner.

Cisco VIM automates the Day 0 ToR level configuration with ACI, except for L3 out. The exception for L3 out is made as you can configure their upstream infrastructure in different ways.
The network model of how an SRIOV network is extended off the fabric is out of the scope of automation. For supported hardware matrix and *setup_data.yaml* changes, see Enabling Cisco VIM with ACI Plugin

> ⚠ Ensure that Cisco VIM is validated against APIC 4.2(3i) or APIC 5.0 with the APIC plugin version of 5.0.

Installation

Installation

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

Cisco NFVI Installation Overview

Cisco NFVI Installation Overview

Cisco NFVI installation is divided into two processes:

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

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

- **Internet Access**: Internet access is required to download the Cisco NFVI installation files from cvim-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.
- **Cisco NFVI Configurations**: Cisco NFVI configurations are included in the setup_data.yaml file. If you are installing Cisco VIM and not Cisco 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 Cisco NFVI data and OpenStack parameters, see OpenStack Configuration. If you are installing Cisco VIM UM, run Cisco NFVI using Insight UI wizard. For more information, see Unified Management.

Following are the license options for installing Cisco NFVI:

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

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

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 Cisco VIM.

- Approach 1: Getting the artifacts onto USB from a staging server running CentOS/RHEL 8.2
- Approach 2: Creating a file based image on the staging server running CentOS/RHEL 8.2

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

Prerequisites for Air-gapped Installation via USB

1. 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 a Quanta server, you can optionally use USB 3.0 64GB to increase the installation speed significantly.
2. Copy the files to the management node.

Prerequisites for Air-gapped Installation via File-based image

1. The CentOS/RHEL 8.2 staging server (VM, laptop, or UCS server) is connected to the docker registry.
2. Each management node must be connected to this staging server.
3. Ensure that the packages truncate and parted with kpartx installed on the CentOS/RHEL 8.2 staging server.

NFVI Installation Setup via USB

NFVI Installation Setup via USB

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

Cisco recommends 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 8.2 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. The staging server must have wired Internet connection to download the Cisco 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 Cisco 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://cvim-registry.com/mercury-releases/cvim40-
rhel8-osp16/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/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 -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 CVIM 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
```

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 Cisco 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

      - buildnode-K9.iso
      - buildnode-K9.qcow2
      - mercury-installer.tar.gz
      - ironic-images-K9.tar.gz
      - registry-2.6.2.tar.gz
      - docker images and manifests
      - python-docker-py-1.10.6-4.el7.noarch.rpm
      - python-docker-pycreds-1.10.6-4.el7.noarch.rpm
      - python-websocket-client-0.32.0-116.el7.noarch.rpm
      - vim_upgrade_orchestrator.py

        > ⚠ For upgrade, the *python-docker-py-1.10.6-4.el7.noarch.rpm, python-docker-pycreds-1.10.6-4.el7.noarch.rpm, python-websocket-client-0.32.0-116.el7.noarch.rpm,* and *vim_upgrade_orchestrator.py* packages are 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
                  [-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 with 3.4.1 release
#./test-usb
           INFO: Checking the integrity of artifacts on this USB drive
           INFO: Checking artifact python-docker-py-1.10.6-4.el7.noarch.rpm
           INFO: Checking artifact python-docker-pycreds-1.10.6-4.el7.noarch.rpm
           INFO: Checking artifact python-websocket-client-0.32.0-116.el7.noarch.rpm
           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.iso
           INFO: Checking artifact buildnode-K9.qcow2
                   INFO: Checking artifact registry-2.6.2.tar.gz
                   INFO: Checking required layers:
           INFO: 764 layer files passed checksum.

# ./test-usb -a
           INFO: Checking the integrity of artifacts on this USB drive
           INFO: Checking artifact python-docker-py-1.10.6-4.el7.noarch.rpm
           INFO: Checking artifact python-docker-pycreds-1.10.6-4.el7.noarch.rpm
           INFO: Checking artifact python-websocket-client-0.32.0-116.el7.noarch.rpm
           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.iso
           INFO: Checking artifact builnode-K9.qcow2
                   INFO: Checking artifact registry-2.6.2.tar.gz
                   INFO: Checking artifact mariadb-app-K9.tar.gz
                   INFO: Checking artifact insight-K9.tar.gz
                   INFO: Checking required layers:
           INFO: 764 layer files passed checksum.
```

If the download fails, an error message is displayed.

For example:

```
# ./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) =
96ec62a0932a0d69daf60acc6b8af2dc4e5eca132cd3781fc17a494592feb52a7f171eda25e59c0d326fbb09194eeda66036cbdc3
870dafe74f59cf1f2dce225'
!= 'SHA512 (buildnode-K9.iso) =
a6a9e79fa08254e720a80868555679baeea2dd8f26a0360ad47540eda831617bea0514a117b12ee5f36415b7540afa112a1c904cd
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.
```

6. 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
```

7. 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
```

8. When the USB integrity test is done, unmount the USB drive by running the following command:

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

NFVI Installation File-Based Image

NFVI Installation via File-Based Image

- Installation Overview
- Assumption
- Prerequisites
- Procedure

Installation Overview

To download Cisco 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 8.2 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 Cisco VIM artifacts to a file-based image on the staging server (running CentOS/RHEL 8.2) 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 fresh installation of Cisco VIM 3.4.3 and later versions. Also, you can use it to execute update starting Cisco VIM 3.4.3.
- In the *setup_data.yaml,* INSTALL_MODE must be set to *disconnected.*

Prerequisites

Ensure that the packages truncate and gets parted with *kpartx* installed on the CentOS/RHEL 8.2 staging server, that is connected to the docker registry.

Procedure

Following are the steps that outline the process:

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

   ```
   curl -o getartifacts.py -u '<username>:<password>' https://cvim-registry.com/mercury-releases/cvim40-rhel8-osp16/releases/<releaseid>/getartifacts.py
   # Change the permission of getartificats.py
   chmod +x getartifacts.py
   ```

3. 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 4.2.0 --file /var/artifacts-4-2-0.img
   ```

   > As this process cannot be aborted in the middle, it is recommended to use either a VNC session/screen/KVM or serial console.

   For upgrade from Cisco VIM 3.4.x (x = 4, 5, or 6) to Cisco VIM 4.2.0, the above command gets appended with *-U* or *--upgrade:*

```
#./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.*

4. On the staging server, verify the integrity and consistency of the downloaded artifacts in the file-based image:

```
 # curl -o mercury-installer.tar.gz https://username:password@cvim-registry.com/mercury-releases/cvim34-
rhel7-osp13/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
```

5. Copy the file-based image */var/artifacts-<release_tag>.img* from the staging server to */var/* of the target management node:

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

Repeat the test-usb on the management node (from the installer-xxx/tools work space) for consistency
check
# cd installer-3.4.3/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 python-docker-py-1.10.6-4.el7.noarch.rpm
INFO: Checking artifact python-docker-pycreds-1.10.6-4.el7.noarch.rpm
INFO: Checking artifact python-websocket-client-0.32.0-116.el7.noarch.rpm
INFO: Checking artifact vim_upgrade_orchestrator.py
INFO: Checking artifact mercury-installer-internal.tar.gz
INFO: Checking artifact ironic-images-internal.tar.gz
INFO: Checking artifact buildnode-internal-24641.iso
INFO: Checking artifact registry-2.6.2-internal-24641.tar.gz
INFO: Checking artifact buildnode-internal-24641.qcow2
INFO: Checking required layers:
INFO: 782 layer files passed checksum
```

6. 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:

```
# cd /root/installer-<release_tag>/tools/
#  ./import_artifacts.sh -f /var/artifacts-<release_tag>.img
#  ./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 a argument to import artifacts
-h              : To display this help
```

Preparing for Cisco NFVI Installation

Preparing for Cisco NFVI Installation

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

Cisco NFVI Hardware Installation

Cisco NFVI Hardware Installation

Switch on the servers, before you install the Cisco VIM. Set up the CIMC connection so that the servers are reachable from the management node via *br_api.* The following table lists the UCS hardware options and network connectivity protocol used with VLAN over OVS, or VLAN over VPP.

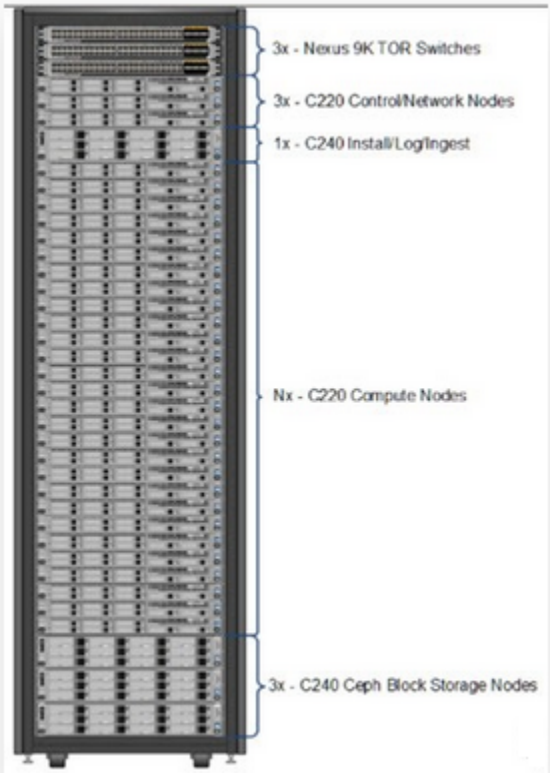
UCS Pod Type	Compute and Controller Node	Storage Node	Network Connectivity Protocol
Rack Type	UCS C220/240 M4/M5	UCS C240 M4/M5 (SFF)	OVS/VLAN or VPP/VLAN (only on intel NIC) or ACI/VLAN (selected HW option)
Rack Type	Controller: UCS C220/240 Compute: HP DL360 Gen9 Compute: Dell PowerEdge 740 Quanta servers for fullon or edge pod	UCS C240 M4 (SFF) Quanta Server	OVS/VLAN
Micropod	UCS 240 M4/M5 Quanta Servers	Not applicable as it is integrated with compute and controller	OVS/VLAN or VPP/VLAN (on intel NIC) OVS/VLAN for Quanta
C-series Hyperconverged	UCS 240 M4/M5	UCS C240 M4/M5 (SFF)	OVS/VLAN

⚠ For BOM details, contact *Cisco VIM Product Management team*

For more information on Cisco vNICs, see LAN and SAN Connectivity for a Cisco UCS Blade.

In addition, the Cisco Nexus 9372 or 93180YC, or 9396PX is also available to serve the Cisco 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 Cisco 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 Cisco NFVI C-series pod.



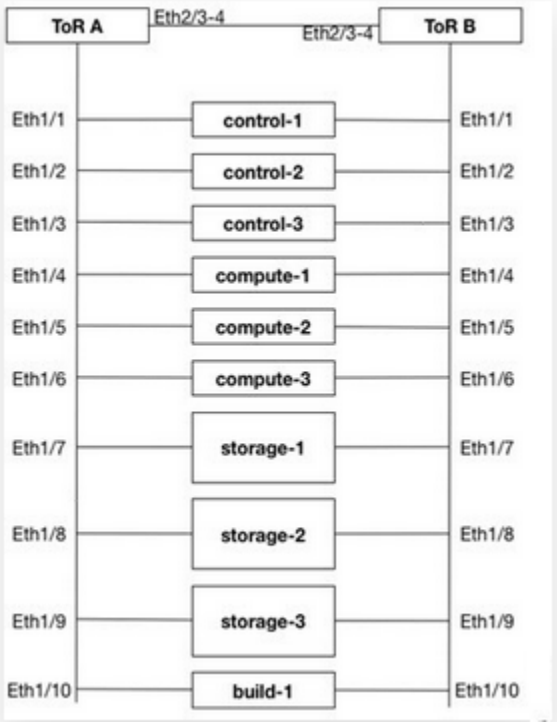
For more information on wiring schematic of various pod configuration, see Wiring Diagrams.

ToR Switch Configuration for C-Series Pods

ToR Switch Configuration for C-Series Pods

During installation, the Cisco 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 to 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.



Cisco 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:

1. 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.

   For the preceding example, the following is the configuration:

   ```
   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
   ```

   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
   ```

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

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

3. 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.

   a. For deployment with any mechanism driver on Cisco VIC:

      There is 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
      ```

   b. 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.

4. 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 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.

   ```
   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
   ```

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

```
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
```

6. 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.

```
gen-leaf-1# show port-channel summary
Flags: D - Down P - Up in port-channel (members)
I - Individual H - Hot-standby (LACP only)
s - Suspended r - Module-removed
S - Switched R - Routed
U - Up (port-channel)
M - Not in use. Min-links not met
--------------------------------------------------------------------------------
Group Port- Type Protocol Member Ports
Channel
--------------------------------------------------------------------------------
1 Po1(SD) Eth LACP Eth1/1(I)
2 Po2(SD) Eth LACP Eth1/2(I)
3 Po3(SD) Eth LACP Eth1/3(I)
4 Po4(SD) Eth LACP Eth1/4(I)
5 Po5(SD) Eth LACP Eth1/5(I)
6 Po6(SD) Eth LACP Eth1/6(I)
7 Po7(SD) Eth LACP Eth1/7(I)
8 Po8(SD) Eth LACP Eth1/8(I)
9 Po9(SD) Eth LACP Eth1/9(I)
10 Po10(SD) Eth LACP Eth1/10(I)
116 Po116(SD) Eth LACP Eth1/116(I)
```

7. 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 Cisco 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
```

8. 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
```

Preparing Cisco IMC

Preparing Cisco IMC

Cisco NFVI requires specific Cisco Integrated Management Controller (IMC) and firmware versions and parameters. The Cisco 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 Cisco 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.
2. For Cisco UCS C-series pods, verify whether the following Cisco IMC information is added: IP address, username, and password.
3. 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.
4. 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 (M4/M5)

Installing Management Node on UCS C-series (M4/M5)

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 Cisco NFVI management node where you plan to install the Red Hat for Enterprise Linux (RHEL) operating system is a Cisco UCS C240 M4/M5 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.

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

   For management node based on UCS M4 boxes, set the following for BIOS parameters:

   ```
   PCI ROM CLP–Disabled
   PCH SATA Mode–AHCI
   All Onboard LOM Ports–Enabled
   LOM Port 1 OptionROM–Disabled
   LOM Port 2 OptionROM–Disabled
   All PCIe Slots OptionROM–Enabled
   PCIe Slot:1 OptionROM–Enabled

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

   a. In the CIMC navigation area, click the **Server** tab and select **Inventory**.
   b. 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.

6. Download the ISO on a local server that can serve HTTP and use CIMC GUI to mount the ISO under *Cisco IMC Mapped vMedia.*
7. In CIMC, launch the KVM console.
8. Mount the Cisco VIM Build node ISO image as a virtual DVD.
9. Reboot the UCS server, then press **F6** to enter the boot menu.
10. Select the CIMC mapped DVD to boot the Cisco VIM Build node ISO image provided with the install artifacts.
11. From the boot menu, select **Install Cisco VIM Management Node**. This is default selection and it gets automatically selected after the timeout.
12. 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 as 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, 172.29.86.62/26
    - API Gateway IPv4 address—Enter the API network default gateway IPv4 address.
    - MGMT IPv4 address—Enter the management node MGMT IPv4 address in CIDR format. For example, 10.30.118.69/26

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

    - Prompt to enable static IPv6 address configuration—Enter **Yes** to continue input similar IPv6 address configuration 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:c5c0:1234:5678:1001::5/8.
    - 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:c5c0:1234:5678:1002::5/80
    - 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 as 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.

1. From the Installation menu, select option **8** to enter the root password.
2. From the Installation menu, select option **2** to enter the time zone.
3. Under the Timezone settings, select option **1,** as option **2** is not supported.
4. Enter the number corresponding to your time zone.
5. Enter the number for your region.
6. 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.

7. After confirming your time zone settings, enter **b** to start the installation.
8. After the installation is complete, press **Return** to reboot the server.
9. 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
```

Management Node on Quanta Servers

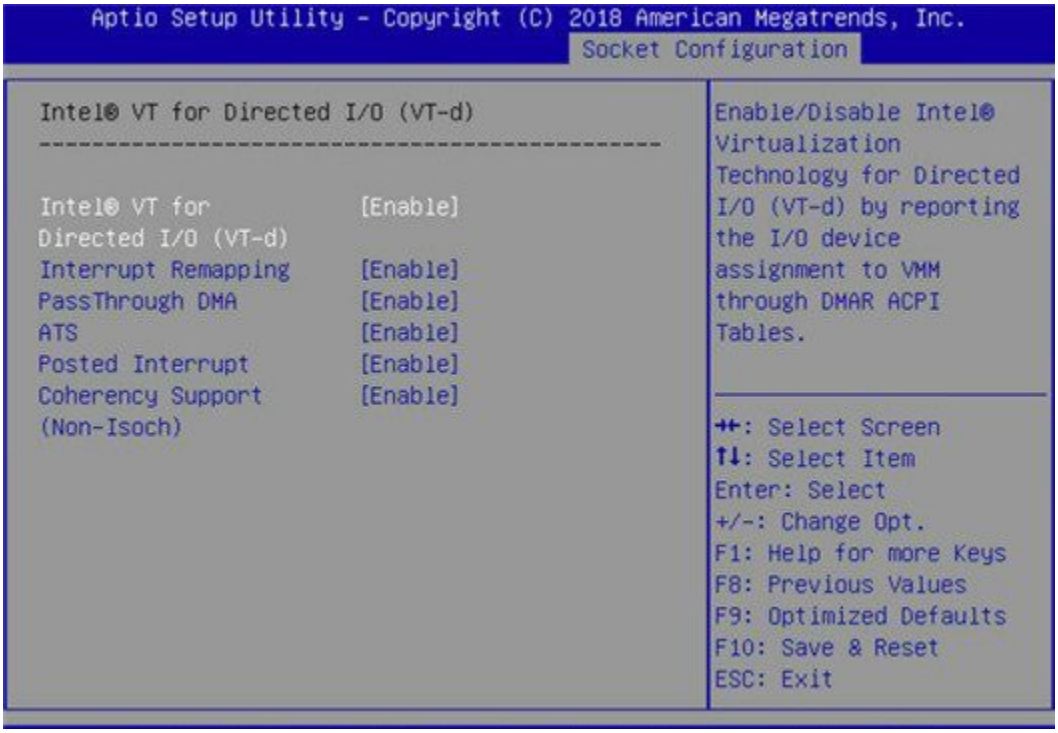
Installing Management Node on Quanta Servers

Most of the settings in the server remain default.

> To run NFVbench, you must enable the Intel VT for Directed I/O (VT-d) on the Management node.

To enable the Intel VT for Directed I/O, navigate to the following path:

BIOS Setup > Socket Configuration > IIO Configuration > Intel VT for Directed I/O (VT-d) > Intel VT for Directed I/O (VT-d) > Enable



To enable NFVbench on a Quanta management node:

- Reboot the MGMT node, press F2 or DEL to enter BIOS:
- Choose **Socket Configuration > IIO Configuration > Intel(R) VT for Directed I/O (VT-d)**.
- Set **Intel(R) VT for Directed I/O (VT-d)** to Enable
- Press F10 to save and exit.

Cisco VIM Software Hub

Installing Cisco VIM Software Hub

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

Cisco VIM Software Hub alleviates the need for Cisco 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.

Cisco 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 Cisco VIM Software Hub Nodes

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

Prerequisites for Cisco VIM Software Hub Server

1. TLS certificate (For production environment)
   On the Cisco 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 Cisco 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 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
     ```

2. Self-signed certificate (For internal use)

   Cisco recommends you to use a trusted CA-signed certificate, when a Cisco VIM Software Hub node is used in a production environment. For internal testing and POC, Cisco supports Cisco VIM Software Hub node with a self-signed certificate.
   Follow the below steps to generate the self-signed certificate:

   ```
   # openssl genrsa -des3 -out https_reverse_proxy.key 2048
   # openssl req -new -key https_reverse_proxy.key -out https_reverse_proxy.csr
   # cp https_reverse_proxy.key https_reverse_proxy.key.org
   # openssl rsa -in https_reverse_proxy.key.org -out https_reverse_proxy.key
   # openssl x509 -req -days 365 -in https_reverse_proxy.csr -signkey
   https_reverse_proxy.key -out https_reverse_proxy.cer
   ```

   Generate the certificate with the same FQDN as specified in the sds_setup_data.yaml. 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, use the same x509 certificate for both cert file and cert chain file. You need to manually trust the self-signed certificate. The operator needs to execute the commands below on both Cisco VIM Software Hub server and Cisco VIM pod management node:

   ```
   # cp <x509 cert> /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 <x509 cert> /etc/docker/certs.d/<fqdn>/ca.crt
```

3. DNS server:

   Ensure that the DNS server can reach the pods and the Cisco VIM Software Hub server. The DNS server must be able to resolve the Cisco 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 Cisco VIM Software Hub Node

The steps to install a Cisco VIM Software Hub node are similar to the steps in Management Node on UCS C-series (M4/M5). The only difference being, in Step 11 of the task, you need to choose the option to configure the server as a Cisco VIM Software Hub server. In the subsequent prompts, you can enter information such as the hostname, ipv4 or ipv6 addresses for br_public and br_private interfaces, and gateway addresses, similar to the Management Node on UCS C-series (M4/M5). In CVIM 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 8.2 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 Cisco 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 Cisco VIM pod must be reachable to the br_private interface of Cisco 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 Cisco 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.
- Cisco 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 Cisco 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/cvim_sds-<tag>
```

Setting up Cisco VIM Software Hub for Cisco VIM Artifact Distribution

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

1. Copy the example file from the *openstack-configs* directory and save it as *sds_setup_data.yaml*.
2. If you want to install a release tag on a Cisco 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 CVIM 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>
############################################################
Cisco Virtualized Infrastructure Manager Installation Guide, 3.0.0
94
Preparing for Cisco NFVI Installation
Installing Cisco VIM Software Hub Node
# Registry credentials to access the CVIM registry (Cisco Supplied)
############################################################
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_USERNAME: <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>
```

3. Save the *sds_setup_data.yaml* file in the following path:

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

Installing Cisco VIM Software Hub in Connected Mode

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

1. Download the *mercury-installer.tar.gz* file of the release that you want.
2. Unzip the zip file manually and rename the unzipped file as *cvim_sds-<release>*.
3. Perform the following steps:

   a) Place a valid TLS certificate in the directory:

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

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

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

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

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

5. Run the following command:

```
# cd to /root/cvim_sds-<target-tag>
# ./sds_runner/runner.py
```

The command validates the Cisco 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 Cisco VIM release registry and populates the Cisco VIM Software Hub server.

Installing Cisco VIM Software Hub in Air-Gapped Mode

Cisco VIM Software Hub is installed in the air-gapped mode when the Cisco 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 Cisco 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.

Before you begin

- Ensure that you have set up a CentOS/RHEL 8.2 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 Cisco VIM installation files, which you want to load onto the USB drive.
- Ensure that you have disabled the CentOS sleep mode.

1. On the staging server, use yum to install the following packages:
   a) PyYAML

   b) python-requests

   Check whether python 3.6.x version is installed. If not, perform *yum install python3*.
2. Access the Cisco VIM software download web site using a web browser.
3. 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://cvim-registry.com/mercury-releases/cvim40-
rhel8-osp16/releases/<releaseid>/getartifacts.py
# Change the permission of getartificats.py
chmod +x getartifacts.py
```

4. 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 CVIM 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.

5. 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]
   ```

6. 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 3.0.0 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 mariadb-app-K9.tar.gz
INFO: Checking artifact haproxy-K9.tar.gz
INFO: Checking artifact insight-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
Cisco Virtualized Infrastructure Manager Installation Guide, 3.4.x
97
Preparing for Cisco NFVI Installation
Installing Cisco VIM Software Hub in Air-Gapped Mode
ERROR: Layer file sha256:ea55b2fc29b95d835d16d7eeac42fa82f17e985161ca94a0f61846defff1a9c8 has a bad
checksum
INFO: 544 layer files passed checksum.
```

7. 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
```

8. 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
```

9. Execute the verification script:

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

   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
```

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

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

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

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

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

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

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

    c. Place a valid TLS certificate in */root/cvim_sds-<tag>/openstack-configs* directory.
    d. Configure the Cisco VIM Software Hub setup data file with all the fields and placed the file in the */root/cvim_sds-<tag>/openstack-configs* directory.
    e. Install the release on the Cisco VIM Software Hub server.
    f. Navigate to the *cvim_sds* directory on the Cisco VIM Software Hub server and execute the following command:

```
# cd /root/cvim_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 Cisco VIM Software Hub Server

When you want to install a Cisco VIM pod using the artifacts obtained from the Cisco 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 Cisco 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 Cisco VIM Software Hub. For example, your domain.com.

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

Cisco 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 Cisco 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 Cisco VIM pod.

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

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

Supported Day 2 Operations

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

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

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:

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

   For servers based on UCS M4 boxes, set the following for BIOS parameters:

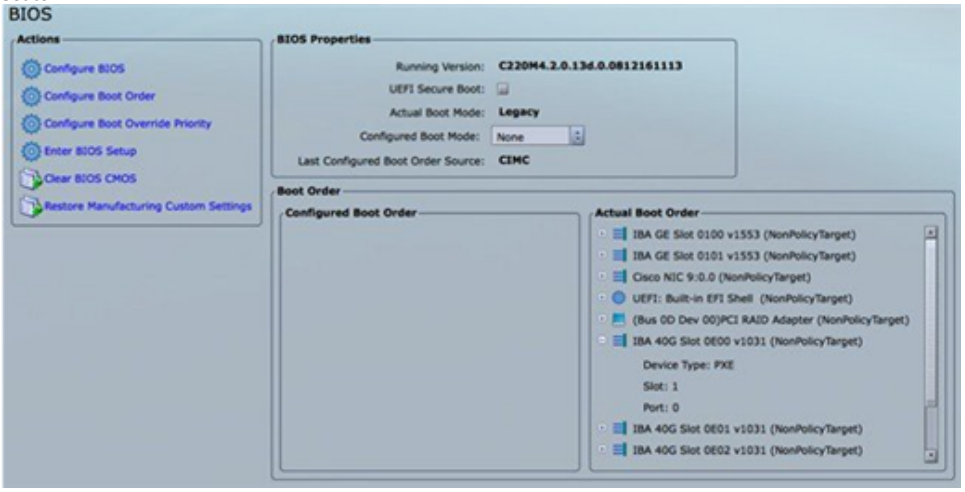
   - CDN Support for VIC—Disabled
   - PCI ROM CLP—Disabled
   - PCH SATA Mode—AHCI
   - All Onboard LOM Ports—Enabled
   - LOM Port 1 OptionROM—Disabled
   - LOM Port 2 OptionROM—Disabled
   - All PCIe Slots OptionROM—Enabled
   - PCIe Slot:1 OptionROM—Enabled
   - PCIe Slot:2 OptionROM—Enabled
   - PCIe Slot: MLOM OptionROM—Enabled
   - PCIe Slot:HBA OptionROM—Enabled
   - PCIe Slot:N1 OptionROM—Enabled
   - PCIe Slot:N2 OptionROM—Enabled
   - PCIe Slot:HBA Link Speed—GEN3

   For servers based on UCS M5 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 their default values.
4. To setup UCS M4 C-series pod with Intel 710 NIC:

   a. Each C-series server must have two 4-port Intel 710 NIC cards.
   b. Connect the ports A, B, and C for each Intel 710 NIC card to the respective ToR.
   c. Enable the PCI slot with Intel NIC cards through the BIOS setting (**BIOS > Configure BIOS > Advanced > LOM and PCI Slot Configuration > All PCIe Slots OptionROM**-Enabled and enable respective slots).
   d. Identify the slots by checking the *slot-id* information under the **Network-Adapter** tab listed under the Inventory link on the **CIMC** pane.
   e. All the Intel NIC ports must be indicated in the BIOS summary page under the **Actual Boot Order** pane, as IBA 40G Slot xyza with Device Type is
   set to PXE.



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



If the boot order for the Intel NICs is not listed as above, enable the PXE boot setting for each UCS-C M4 series server by using either Intel's BootUtil tool on a pre-installed Linux system or boot a special ISO image. This is time consuming especially on a large pod with many nodes. Hence, an automated tool is developed to help with this painstaking process.

> ⚠ From Cisco VIM 3.4.0, the above context is applicable only to UCS M4 series servers, as UCS-M5 is based on UEFI for boot.

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://downloadcenter.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 Cisco VIM Management node to UCS-C series server's CIMC over following ports:

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

Ensure that there is unrestricted network access from UCS-C series server's CIMC to Cisco 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 Cisco VIM Installer's tools directory are:

- *create-bootutil-img.sh*
- *intel-bootutil-update.py*

Usage

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

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

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
```

Example to disable all port C:

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

Workflow:

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

1. Download PREBOOT.exe version (listed above) from Intel's website.
2. Go to Cisco VIM Installer's tools directory.
3. Run *create-bootutil.img* script to create a CIMC-KVM mountable USB image.
4. 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
...
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 Cisco VIM installer API and SSH bonded interface, use 1-GB Intel NICs that connect the Cisco 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
```

Third-Party Compute Support

Support of Third-Party Compute

- Overview
- HP DL 360 Gen9/Gen10
- Dell PowerEdge R740/R640

Overview

Cisco VIM supports selected third-party computes from HPE and Dell in an environment where the controller and Ceph nodes are based out of Cisco UCS. Though Cisco VIM supports full automation, it does not have the redistribution license of utility tool to manage the third-party hardware.

HP DL 360 Gen9/Gen10

For Cisco VIM to manage HP DL360 Gen9/Gen10 as computes, download the Smart Storage Administrator CLI (ssacli) tool directly from HPE website and place the RPM file in */root/installer-<tagid>/openstack-configs/* directory.

> ⚠ Cisco VIM supports *ssacli-4.17-6.0.x86_64.rpm*

The location of the target RPM is available at: https://downloads.linux.hpe.com/SDR/repo/mcp/centos/7/x86_64/current/ssacli-4.17-6.0.x86_64.rpm

The checksum of the target RPM is available at:

SHA1 checksum: d30b81277671179e2c3ea8fe5c1aaeaaed2efb0e

Dell PowerEdge R740/R640

For Cisco VIM to manage Dell PowerEdge R740/R640 as computes, download the PowerEdge RAID Controller (PERC) CLI utility directly from Dell website and place the RPM file in */root/installer-<tagid>/openstack-configs/* directory.

> ⚠ Cisco VIM supports *perccli-007.1020.0000.0000-1.noarch.rpm*

The location of the target RPM is available at: https://www.dell.com/support/home/en-us/drivers/driversdetails?driverid=wd0r5

The checksum of the target RPM is available at:

SHA1 checksum: d36124979d144dfc800f6be5726b74c6451acd6c

Centralizing Management Node

Centralizing Management Node

- Assumptions/Best practice
- Support Matrix
- DHCP Relay Configuration for UCS-based Pods
- Add Target Compute(s) to AZ for Management Node VMs (Optional)
- Workflow/Usage
- Typical VM Flavor
- Execution of Deployment Orchestration
- Tooling Usage
- Launching Resources
- Management Node VM Removal
- Management Node VM and Resource Status
- Central VM Management Result
- Accessing Management Node VMs
- Setup Data Changes for Management Node VM
- Additional Setting for Quanta BMC
- Management Node VM optimization for Nano Pod

For architectural information, see Management Node Centralization

Assumptions/Best practice

To achieve centralization of the management node, the following assumptions are made:

1. The QCOW2 generated by Cisco VIM release must be used as a management node VM.
2. There is a one-to-one relationship between the management node VM and target pod.
3. As the management node VMs are curated to manage Cisco VIM pods, a fully automated orchestration (backed with RestAPI) is provided to manage the same.
4. The management node VMs can only be hosted on a Cisco VIM based cloud with pod type as micro, fullon, UMHC, or NGENAHC. Careful planning of storage must be made on the pod that hosts the management node VMs. This hosting pod typically requires a physical management node to be managed.
5. The management node VMs can manage micro, fullon, UMHC, NGENAHC, or edge pod. For micro, UMHC, or NGENAHC pod hosting the virtual Cisco VIM management node, the following restrictions apply:

   a. The setup_data must run with CEPH_OSD_RESERVED_PCORES: 6

   b. The setup_data must run with *cinder_percentage_data* of 80 or more in CEPH_PG_INFO. This is a Day 0 configuration.
6. The management node VMs can also be hosted on pods where Cinder is supported via Zadara, SolidFire, or multi-backend ceph. For multi-backend ceph, by default, the VMs are booted off volumes created on SSDs. For SolidFire, SSD, and HDD, the volume types with the names *central_mgmt_solidfire, central_mgmt_ceph_ssd*, and *central_mgmt_ceph_hdd* respectively are reserved by the system, and not to be used elsewhere in the cloud.
7. For all types of pods hosting the virtual Cisco VIM management nodes, NOVA_RAM_ALLOCATION_RATIO must be 1.0 globally or on a per compute basis for all the computes belonging to Availability Zone (AZ) defined in CENTRAL_MGMT_AZ of the *setup_data.CentralMgmt.yaml*
8. Host the VMs on compute where NOVA_CPU_ALLOCATION_RATIO is set to 1.0, to avoid resource contention.
9. For the pod hosting the virtual Cisco VIM management nodes, the corresponding VMs are hosted in a project called *central_mgmt,* so that CVIM MON can monitor these nodes as special resources. The project *central_mgmt* is created automatically via the Cisco VIM orchestrator that handles the management node VMs.
10. If the cloud is sharing workloads with other VNFs, it is recommended to add at least one compute node to the host aggregate created by the cloud admin. This is an optional manual step which helps to streamline the operation. The entry of CENTRAL_MGMT_AGGREGATE in the *setup_data.CentralMgmt.yaml* reflects the name of the host aggregate which hosts the management node VMs. With this approach, the cloud administrator can add more computes to the AZ as needed. For details on adding computes, see *Add Target Compute(s) to AZ for Management Node VMs*
11. Admin must allocate a minimum of two routable networks for the management network *(br_mgmt)* and API network *(br_api)* of the VMs. When the VM is a Cisco VIM software hub, the two routable networks are *br_public* and *br_private*. For UM, only the API network (br_api) is used by the VM.
12. File system access to the public key that is imported during VM deployment is needed.
13. For managing the lifecycle of the management node VMs, all associated cloud resources are defined in a *setup_data,* example of which is available at */root/openstack-configs/setup_data.CentralMgmt.yaml.*
14. If cloud is running with HugePages, the management node VMs are automatically launched using HugePages. It is assumed that all computes hosting the management node VMs have the same HugePage size and percentage.
15. Given the criticality of the management node, ensure that CVIMMON is enabled on the cloud hosting the management node VMs and pods managed by the management node VMs.
16. The pod managed by centralized VM, needs to run off a central CVIM-MON infrastructure. Only local CVIM-MON must be used as a transition phase to ensure optimal usage of system resources.
17. The NFVBench is not supported in the deployment of central management VMs.
18. Provider networks belonging to VMTP section must not be used for creating central management VMs networks.
19. After the launch of central management VMs, the administrator must not forget to take a manual backup of the management node and copy the snapshot to another server.
20. The VM usage is supported for management node, Unified Management node, and Cisco VIM software hub.

Support Matrix

In Cisco VIM 3.4.3, management node VMs can be used to drive the following:

Pod Hardware Type	Management Network	BMC User and Network Requirements
Quanta servers	v6/v4	BMC user needs to have Admin, KVM, and VMedia access privileges. Layer 3 connectivity from management node VM to pod management network. v6 network must be routable.
Cisco UCS C-series	v6/v4 or v4	Layer 3 connectivity from management node VM to pod management network. v4 network must be routable. Need to identify the management subnets and configure the DHCP relays on the ToRs attached to the target Cisco VIM OpenStack cloud. The DHCP IP relay address points to the management VM's br_mgmt interface (over v4), which serves the DHCP requests for the cloud hosts via cobbler.

DHCP Relay Configuration for UCS-based Pods

```
F13_N93180_1# show run dhcp

!Command: show running-config dhcp
!Running configuration last done at: Tue Sep 3 12:17:01 2019
!Time: Tue Sep 3 12:17:18 2019

version 9.2(1) Bios:version 07.64
feature dhcp

service dhcp
ip dhcp relay
ipv6 dhcp relay


interface Vlan<id>                                                  --> Target Pod's management network
  ip address <a.b.c.d
/mask>                                                                                  --> SVI address
for VLAN 1228
  ip dhcp relay address <br_mgmt_v4_addr of Management node VM>       --> Management node VM's br_mgmt IPv4
address
```

Add the necessary routing on the ToRs, so that the management node VM's gateway is reachable from the ToR.

Add Target Compute(s) to AZ for Management Node VMs (Optional)

If the cloud is sharing workloads with other VNFs, it is recommended to have target compute nodes for management node VMs. To achieve that, the cloud administrator must create a host aggregate. This is a manual step which helps to streamline the operation. The entry of CENTRAL_MGMT_AGGREGATE in the *setup_data.CentralMgmt.yaml* reflects the name of the host aggregate which hosts the management node VMs. With this approach, cloud administrator can add more computes to the AZ as needed.

```
# cd /root/installer-xxx/openstack-configs
# source openrc

Fetch Hypervisor List
# openstack hypervisor list

Create Availability Zone for Central Management VM
# openstack aggregate create <target_az_name>               # Should match the value of CENTRAL_MGMT_AZ defined
in setup_data for Central VM management
# openstack aggregate set --zone <target_az_name>_az <target_az_name>  # while the Zone name can be different
from the availability_zone setting, it is recommended to append "_az" to the name to set the AZ for operation
ease

Add hypervisor to the AZ, one at a time
# openstack aggregate add host <target_az_name> <hypervisor_name1>
# openstack aggregate add host <target_az_name> <hypervisor_name2>
.
.

Check output of Availability Zone for Central Management VM
# openstack aggregate show <target_az_name>
```

Workflow/Usage

1. You must download the QCOW2 image from the release link onto the right directory of the management node of the cloud hosting the management node VMs. For example,

   ```
   cd to <dir_for_qcow2_image>
   curl -o cvim-image-NNNN.qcow2 https://cvim-registry.com/mercury-releases/cvim34-rhel7-osp13/releases
   /<target-release>/buildnode-<release>.qcow2
   ```

2. For Cisco VIM Software Hub based installation, use the following:

   ```
   cd to <dir_for_qcow2_image>
   curl -o cvim-image-NNNN.qcow2 https://<sds-registry-host>/mercury-releases/cvim34-rhel7-osp13/releases
   /<target-release>/buildnode-<release>.qcow2
   ```

3. Create the *setup_data.CentralMgmt.yaml* by defining required resources to be created on the host pod.

   The following resources must be defined based on the information in *setup_data.CentralMgmt.yaml*:

   - Provide username/password for the OpenStack project of *central_mgmt*.
   - Provider network and subnets for API and management network segments.
   - Security group and rules.
   - Glance image for the CVIM QCOW2 image.
   - Flavor specification.
   - Keypair to import.
   - VOLUME_BACKEND: To indicate the target device type (SSD or HDD) to boot off in a multi-backend ceph. If not defined, system defaults to SSD for creation of volumes.
   - Servers and the corresponding network ports and bootable volume for the source image.

   > ⚠ Following are the recommended best practices:
   >
   > - Use a host aggregate, so that computes dedicated for management node VMs are placed in it.
   > - Define the timezone at a global or per sever level.
   > - Define the cvimadmin_password_hash for console access to the management node at a global or per sever level.

Typical VM Flavor

With the characterization done so far, ensure that the management node VM contain 8 vCPUs, 25G RAM, and 1 TB HDD. Ensure that the management node VMs are run without oversubscription of CPU and memory, so that the management node resources are available all the time. In this regard, the choice of CPU and memory of the compute hosting the management node VM must be made, after calculating the number of management node VMs (consuming 8 vCPUs, 25G memory) that needs to be hosted without oversubscription.

Listed below is the corresponding flavor name:

```
- name: 'cvim-mgmt.8x24-nonlocal'
  ram: 24576
  vcpus: 8
  disk: 1024
```

Careful planning of memory is required, as the servers hosting the management node VMs cannot run with memory oversubscription, to ensure system stability.

Listed below is a sample of the *setup_data.CentralMgmt.yaml*, where all the target resources are defined:

```
#
# This is an example file for Centralizing the management node. It assumes that each of the management node is
running a VM in a cloud powered by Cisco VIM. There is a cloud # image that is released as part of the
artifacts used to get it going
#
PODTYPE: MGMT_CENTRAL

CENTRAL_MGMT_USER_INFO:
  username: <userrname>
  password: <password>


TIMEZONE: <entry as defined in TZ database name or notes of https://en.wikipedia.org/wiki
/List_of_tz_database_time_zones>
                           # Optional; the local value at a per server level overrides the global value;
                           # if local or global is not defined, it will take the timezone of the management
node that is hosting the VMs.

# List of network segments, min of 2
NETWORKS:
- segment: management   # can be either management or api
  vlan_id: <unique vlan_id part of provider network>
  name: <unique name of the network>
  subnets:
  - name: <unique name of the v4 subnet>
    network_cidr: a.b.c.d/mask
    gateway: x.y.x.w
    range: ['e.f.g.h', 'i.j.k.l']     # Start and End IP range to launch VM off
    ip_version: 4
    dns_nameservers: ['v4_dns_server']   # list of v4 DNS servers; max of 3
  - name: <unique name of the v6 subnet>  # Optional only needed if v6 is there
    network_cidr: '<cidr_in_v6 with mask>'  # cidr in v6
    gateway: '<gateway_in_v6>'         # gateway in v6
    range: ['start_ipv6_address', 'end_ipv6_address']   # Start and End IP range to launch VM off
    ip_version: 6
    dns_nameservers: ['v6_dns_server_list'] # list of v6 DNS servers; max of 3
- segment: api  # can be either management or api
  vlan_id: <unique vlan_id part of provider network>
  name: prov-net-api-2001
  subnets:
  - name: <unique name of the network>
    network_cidr: e.g.f.h/mask     # Unique CIDR
    gateway: p.q.r.s             # Corresponding Gateway
    range: ['e.f.g.d', 'i.t.k.l']  # Start and End IP range to launch VM off
    ip_version: 4
    dns_nameservers: [v4_dns_server]   # list of v4 DNS servers; max of 3
  - name: <unique name of the v6 subnet>  # Optional only needed if v6 is there
    network_cidr: '<cidr_in_v6 with mask>'  # cidr in v6
    gateway: '<gateway_in_v6>'         # gateway in v6
    range: ['start_ipv6_address', 'end_ipv6_address']   # Start and End IP range to launch VM off
    ip_version: 6
    dns_nameservers: ['v6_dns_server_list'] # list of v6 DNS servers; max of 3


 # List of Images for Management Node, min of 1
IMAGES:
- name: <unique_image_name>       # Unique name of images
  file_location: '<qcow_image_path>'
```

```
# Flavor info for Management Node, min of 1 info
FLAVORS:
- name: '<unique_flavor_name>'      # Unique name of flavor
  ram: <4096 to 200000>
  vcpus: <4 to 20>
  disk: <512 to 8192>          # Optional, but must be mutually exclusive for disk_vol_size on a per server
basis, between disk and disk_vol_size one of them has to be defined

# List of Keypairs for Management Node, min of 1
KEYPAIRS:
- name: mgmt_vm_keypair         # Unique Name of KEYPAIRS
  public_key_file: '/root/.ssh/id_rsa.pub'


#CVIMADMIN_PASSWORD_HASH should be the output from:

#  python -c 'import crypt; print crypt.crypt("<plaintext_strong_password>")'
# username is: cvimadmin
#CVIMADMIN_PASSWORD_HASH: <Please generate the admin pwd hash using the step above; verify if the output starts
with $6>



# Optional, mechanism to group computes dedicated for Virtual Management Node into one host aggregate
CENTRAL_MGMT_AGGREGATE: <Name of Host Aggregator> # Admin has to add the computes to the host aggregator named
here


# Optional, only allowed when Ceph is multi-backend (HDD and SSD based); if not defined SSD will be used to
create volume in a multi-backend environment. Also, for the entire system there is an uniform backend
#VOLUME_BACKEND: <SSD or HDD>


# List of Management Node in VM, min of 1
SERVERS_IN_VMS:
- name: <unique name of servers in VM>
  keypair: mgmt_vm_keypair     # Keypair name that is defined under KEYAPAIR/name
  image: <unique_image_name>   # image name that is defined under IMAGES/name
  flavor: '<unique_flavor_name>'  # flavor name that is defined under FLAVORS/name
  disk_vol_size: <512 to 8192> # integer, value in Gig, min of 512, max of 8192 Gig; optional but need to be
mutually exclusive for the relevant flavor, between disk and disk_vol_size one of them has to be defined
  node_type: <management or um or sds>
  domain_name: <your.domain.com> # optional
  timezone: <<entry as defined in TZ database name or notes of https://en.wikipedia.org/wiki
/List_of_tz_database_time_zones>
                            # Optional; the local value at a per server level overrides the global value;
                            # if local or global is not defined it will take the timezone of the
                            # management node that is hosting the VMs

  cvimadmin_password_hash: <Generate the admin pwd hash using the step above; verify if the output starts with
$6>
                            # Optional if defined globally; refer to the CVIMADMIN_PASSWORD_HASH section on how
to generate the password;
                            # username is: cvimadmin

  nics:                     # entry of 2 Items for node_type management or sds, entry of 1 is when node_type is
um

    - name: <unique_nic_name_across_all_vms>  # For br_mgmt
      network_name: prov-net-mgmt-2000        # Should Match Provider Network Name
      fixed_ips:
      - subnet: v4_mgmt_subnet                # IPv4 subnet network name
        ipaddress: <ipv4_address>             # IPV4 address belonging to v4_mgmt_subnet
      - subnet: v6_mgmt_subnet                # IPv6 subnet network name
        ipaddress: '<ipv6_address>'           # IPV6 address belonging to v6_mgmt_subnet
    - name: <unique_nic_name_across_all_vms>  # for br_api
      network_name: prov-net-api-2001         # Should Match Provider Network Name
      fixed_ips:
      - subnet: v4_api_subnet                 # IPv4 subnet network name
        ipaddress: <ipv4_address>             # IPV4 address belonging to v4_api_subnet
```

```
    - subnet: v6_api_subnet                # IPv6 subnet network name
      ipaddress: '<ipv6_address>'          # IPV6 address belonging to v6_api_subnet
```

Execution of Deployment Orchestration

- Create the right host aggregate for the management node VMs and assign the right set of computes (Optional and API driven process).
- Update the target *CentalManagement setup_data* with the relevant VM information to spawn the management node VMs.
- Execute the ciscovim CLI to launch one or more VMs

Tooling Usage

```
[root@cvim-mgmt ~]# ciscovim help central-vm
usage: ciscovim central-vm [--vms <vm_name1,vm_name1,...>]
                           [--file <setup_data_central_vm.yaml>] [-y]

                           <launch-all|clean-all|add-vms|delete-vms|list-vms|result|join>

Central Management VM Commands

Positional arguments:
  <launch-all|clean-all|add-vms|delete-vms|list-vms|result|join>
                                The Command to Perform for Central VM
                                Management

Optional arguments:
  --vms <vm_name1,vm_name1,...>   Comma separated list of VM names
  --file <setup_data_central_vm.yaml>
                                Provide a valid 'setup_data_central_vm.yaml'
                                file
  -y, --yes                     Yes option to perform the action
```

Launching Resources

To launch resources defined in *setup_data.CentralMgmt.yaml*, use the below command:

```
# To launch Central VMs for the first time
ciscovim central-vm launch-all --file <target_setup_data> [-y]

# To add VMs later
ciscovim central-vm add-vms --file <target_setup_data> --vms <, separated vms> [-y]   # In the target
setup_data, the target VMs are present in the SERVERS_IN_VMS section
```

Management Node VM Removal

To remove a specific or all management node VMs, use the following command:

```
# To delete all Central VMs
ciscovim central-vm clean-all --file <target_setup_data> [-y]   # In the target setup_data the section
SERVERS_IN_VMS should be absent

# To delete specific VMs
ciscovim central-vm delete-vms --file <target_setup_data> --vms <, separated vms> [-y] # In the target
setup_data the target VMs are absent in the SERVERS_IN_VMS section
```

Once the VM comes up, in case of management node, it contains all the artifacts for *ciscovim* client and *mercury-restapi* installed and running. Depending on the node_type, the network interfaces are appropriately configured.

As the VM password management is not handled by this workflow deployment, the operator must login using the sshkey and then change the password in the VM.

After launch of central management VMs, the administrator must not forget to take a manual backup of the management node and copy the snapshot to another server.

Management Node VM and Resource Status

To list the status of the management node VMs and the resources available to launch additional management node VMs, use the following command:

```
# To list all Central VMs
ciscovim central-vm list-vms [-y]
```

The output lists the summary of management node VMs, their status and the interface IP addresses. Also, it lists the total number of VMs currently running and the resources such as compute, vCPU, RAM, and storage available to launch additional management node VMs.

Central VM Management Result

To view the last result of the run, execute the following:

```
# To list all Central VMs
ciscovim central-vm result [-y]
```

Accessing Management Node VMs

If the management node VM is in *Active* state, some additional Cisco VIM services are available automatically. SSH to management node is not allowed while availing the additional Cisco VIM services. To check if all the services are up and ssh is allowed, execute the following openstack command as a cloud admin:

```
# source /root/openstack-configs/central_mgmt.openrc

# To check if VM is ready for SSH access
# nova console-log <vm_name> | grep -o 'CVIM RestAPI install completed'
 or
# openstack console log show <vm_name> | grep -o 'CVIM RestAPI install completed'
```

Once the VM is ready for SSH access, you can set up the management node VM login in one of the following ways:

1. From the management node of the cloud hosting the VM, you can SSH to the VM and set the password via Linux passwd mechanism for access.
2. Log into the VM as root user, using the private ssh key corresponding to the public key that is provided as part of the KEYPAIRS entry in */root /openstack-configs/setup_data.CentralMgmt.yaml.*

Only the cvimadmin user whose password is defined by: CVIMADMIN_PASSWORD_HASH, have console access, and not the remote user. Attempting to login with this user with SSH will not work.

After launch of central management VMs, the administrator must take a manual backup of the management node and copy the snapshot to another server.

Setup Data Changes for Management Node VM

To support the installation of management node VM over Layer 3, use the following command:

```
NETWORKING:

  domain_name: <domain_name>
  domain_name_servers: [dns_name_server_list]

# Optional, only allowed when the management node is centralized.
# need info of br_mgmt of the Centralized Management Node, IPV6 info is needed for dual stack environment
  remote_management:
   gateway: <ipv4_gateway>
   ipv6_gateway: '<ipv6_gateway>'
   ipv6_subnet: '<ipv6_subnet_with_cidr>'
   subnet: <ipv4_subnet_with_cidr>

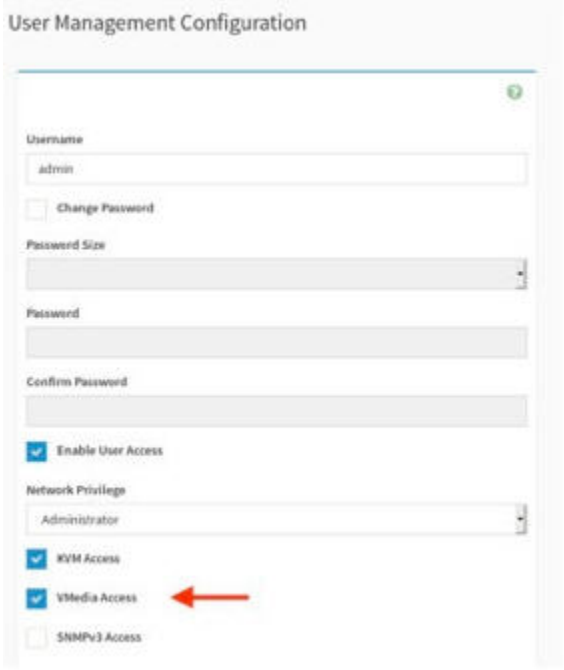
  networks:
  - gateway: <v4_gateway>
    ipv6_gateway: <v6_gateway>
    ipv6_pool: [v6_pool_range]
    ipv6_subnet: <v6_subnet_cidr>
    pool: [v4_pool_range]
    segments: [management, provision]
    subnet: <v4_subnet_cidr>
    vlan_id: <management_vlan_id>
```

The pod's provision and management traffic continue to flow between management node VM's br_mgmt and OpenStack node's br_mgmt/mx interface, but over Layer 3. Since the management node is running in a VM, a faster log rotation within the VM is defaulted. It is recommended to configure the management node VMs with external syslog server, to ensure that you have access to all the necessary logs.

Additional Setting for Quanta BMC

For the central management VM feature to work, the BMC user must have Admin, KVM, and VMedia access privileges as depicted below. To enable it, execute the following:

1. Navigate to each of the Quanta BMC Web UI.
2. In the UI, choose **Settings** > **User Management** > <**User account**>
3. Verify whether **Enable User Access (as Administrator), KVM Access,** and **VMedia Access** options are selected:



Management Node VM optimization for Nano Pod

As the cloud gets deployed at the far-edge, there is an evolution of its footprint towards an offering where a single box hosts the entire workload. The cloud hosted off a single node called nano pod that has both control and compute functionality, with no persistent storage and with no scope for expansion. When a nano pod is managed by a virtual management node that is Layer 3 distance away from itself, an optimization to reduce the persistent storage required by the virtual management node to a minimum of 256G is done. For such design optimizations, following are the assumptions set when using the CENTRAL_REGISTRY option in the setup_data for the nano pod:

- Cisco VIM Software hub is always reachable from the management node and the nano pod continues with Cisco VIM installation, software update, and reconfiguration activities.
- Cisco VIM Software hub has the correct image tags for the pods running off.
- External syslog server is enabled on the management node VM to conserve the storage used.
- Management node VMs (consuming 8 vCPUs, 25G memory and 256G disk space) are running with no CPU or memory oversubscription.
- Since there is no HA in a nano pod deployment, the docker registry gets populated with the right containers, and the control and data plane outage happens during reconfiguration and software update.

To avail this feature, update the *setup_data* of the management node VM driving nano pod installation on Day 0 as given below:

```
#########################################################
# Registry credentials
#########################################################
REGISTRY_NAME: '<registry_name>'     # Mandatory parameter, when SDS is enabled or Central Registry is enabled.
Example registry FQDN name (your.domain.com).
                              When SDS or Central registry is not enabled, this parameter is not needed
REGISTRY_USERNAME: '<username>'
REGISTRY_PASSWORD: '<password>'
REGISTRY_EMAIL: '<email@address.com>'

# Below one is Optional parameter for central registry. This parameter supports only nano pod installation.
CENTRAL_REGISTRY: Enabled
```

CVIM Monitor

CVIM Monitor Configurations

- Overview of CVIM Monitor
- Enabling CVIM-MON on Cisco VIM
- Monitoring External Servers Using CVIM-MON
    - Assumptions for Monitoring External Servers Using CVIM-MON
    - Installation Procedure for Monitoring External Servers Using CVIM-MON Local
    - Uninstallation Procedure for Monitoring External Servers Using CVIM-MON Local
    - Installation Procedure for Monitoring External Servers Using CVIM-MON Central
    - Uninstallation Procedure for Monitoring External Servers Using CVIM-MON Central
- Enabling CVIM-MON Post Pod Installation
- CVIM-MON Grafana/Prometheus and AlertManager with LDAP Backend

Overview of CVIM Monitor

The Cisco VIM Monitor feature (CVIM-MON) provides a comprehensive solution for monitoring the health and for tracking the usage of resources in the Cisco 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 with pre-defined dashboards customized for Cisco 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 CVIM-MON service is called a CVIM-MON stack. The default deployment mode for CVIM-MON is to deploy the CVIM-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 CVIM-MON on Cisco VIM

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

The components of CVIM-MON are as follows:

- CVIM_MON: Provides the base functionality of monitoring and KPIs.
- SNMP: Enables sending CVIM-MON alerts as SNMP traps, only if CVIM-MON is enabled.
- SERVER_MON: Enables collecting UCS-C bare metal alerts into CVIM-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.

To enable CVIM-MON, the **CVIM_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 CVIM-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 CVIM-MON UI access is enabled or not. By default, it is enabled.

PODNAME is mandatory for CVIM-MON and is a name that identifies uniquely each CVIM 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 *Engine_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 whether CVIM-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_servers	List of external server IPs (v4 or v6) that must be monitored by CVIM MON	-	Optional. For more information, see Monitoring External Servers Using CVIM-MON
CVIM_MON: polling_intervals: high_frequency:	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 CVIM-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.
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 user name.
auth_key	<string>	(required v3)	Indicates the authorization password. Must contain at least eight characters.
authentication	SHA|MD5	SHA	Specifies the authentication protocol.
privacy_key	<str>	(auth_key)	Encryption key
encryption	'AES128'|'AES192'|'AES 256'	'AES128'	Encryption protocol
SERVER_MON: 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_SERVERS section	'ALL'	Specifies the UCS-C servers to be monitored.
rsyslog_severity	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
MANAGEMENT_SERVERS:	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

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 CVIM-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>}
  :
  :
```

Monitoring External Servers Using CVIM-MON

CVIM-MON can monitor external non-CVIM servers running RHEL or CentOS 8.2. CVIM-MON monitors these servers by installing a Telegraf agent on them to collect bare metal and libvirt metrics. The telegraf agent runs as 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 CVIM-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 CVIM-MON deployments.
- The external servers must run on UCS M4 or M5 hardware similar to the Cisco VIM management node BOM.
- The external servers must run Cisco VIM management node ISO or CentOS 8.2.
- The external nodes must run on the same site as the monitoring Cisco VIM pod. The scraping of metrics occurs over unauthenticated and unencrypted HTTP connections on port 9273.
- Cisco VIM artifacts of Unified Management and Cisco VIM software hub can be monitored via CVIMMON local or central installation.

Installation Procedure for Monitoring External Servers Using CVIM-MON Local

To enable monitoring of external servers using CVIM-MON, you must update the *setup_data.yaml* file with the details of the external_servers, and run fresh installation or reconfiguration in the Cisco 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/cvim_mon*.

An example of a setup data file with CVIM-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 CVIM-MON, you must execute the following steps on each of the target external servers:

1. Copy the *external-monitoring-<telegraf version>.tar.gz* file from the management node of the Cisco VIM pod to root directory in the external server.

2. Open port 9273 on the respective external server (as per the IP defined in the Cisco VIM pod) to allow communication between Prometheus on the management node and the Telegraf agent on the server.
3. Untar 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

4. To deploy Telegraf on the external server, run the *monitor_external bash* script in privileged mode. This script will be found in the directory *(external-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
```

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 CVIM-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:

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


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 CVIM-MON Central

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

Uninstallation Procedure for Monitoring External Servers Using CVIM-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 CVIM_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 CVIM-MON Post Pod Installation

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

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

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

1. 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
   ```

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

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

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

CVIM-MON Grafana/Prometheus and AlertManager with LDAP Backend

The CVIM-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.

CVIM-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 CVIM_MON section as following (replace example values as appropriate):

```
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 *Viewer.* 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.
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 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 CVIM-MON

Highly Available CVIM Monitor

Overview of HA CVIM-MON

Overview of Highly Available Cisco VIM Monitor

From Cisco VIM 3.4.3, you can monitor Cisco VIM pods either:

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

The local CVIM-MON (introduced in Cisco 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 Cisco VIM pod size (128 nodes).
Local CVIM-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 Cisco 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 CVIM-MON is a complete hardware and software platform that addresses local CVIM-MON limitations with the following features:

- Integrated and highly-available monitoring of a large number of Cisco VIM pods
- Centralized TSDB, alarm and web-based GUI dashboards
- It is based on bare metal Kubernetes platform
- Scalable to hundreds of Cisco VIM pods and thousands of nodes.
- Provides a longer retention time for collected metrics (months instead of 15 days for the local CVIM-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 CVIM-MON supports and requires a limited set of hardware configurations. HA CVIM-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 Cisco VIM 3.4.x and Cisco VIM 4.0.0:

Features	Cisco VIM 3.4.x	Cisco VIM 4.0.0	Comments
Cisco VIM pod local monitoring (CVIM-MON)	yes	yes	
Cisco VIM pod centralized monitoring (HA CVIM-MON)	yes	yes	Local and centralized monitoring are exclusive. Requires an HA CVIM-MON cluster to monitor the Cisco VIM pods.
CVIM pod local to central monitoring reconfiguration	yes	yes	No service impact
CVIM pod central to local monitoring reconfiguration	no	no	Not supported
HA CVIM-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 Cisco VIM pods configured in that domain.
HA CVIM-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 Cisco VIM pods.
HA CVIM-MON max number of monitoring stacks per cluster	min(4, N-1)	min(4, N-2)	N: number of HA CVIM-MON cluster nodes (controller and worker nodes) For example, with HA CVIM-MON 4.0, the minimum configuration to support 4000 sites is three controller nodes and three worker nodes.
HA CVIM-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 CVIM-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 Cisco VIM pod servers. UCS-C bare metal alerts are still not supported in 4.0.0 and planned for a future version.
HA CVIM-MON Grafana built-dashboards organization	flat	1-level folders	In 4.0.0, the built-in dashboards are grouped by folders
HA CVIM-MON CLI integration	partial	complete	In 4.0.0, all cluster management operations are integrated under the *k8s_runner* CLI

Hardware Requirements for HA CVIM MON

Hardware Requirements for HA CVIM-MON

HA CVIM-MON is available for Cisco UCS C-series servers or Quanta servers with:

- One server used as a management node.
- Three or more servers to form a cluster managed by Kubernetes:
    - 384GB 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.
- Each Quanta server must have one Intel XXV 710 card.

Networking Layout

Networking Layout for HA CVIM-MON

- Public Network
- Management and Provisioning Segment

Public Network

The public network (br_api) interfaces with:

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

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 CVIM-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 CVIM-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.

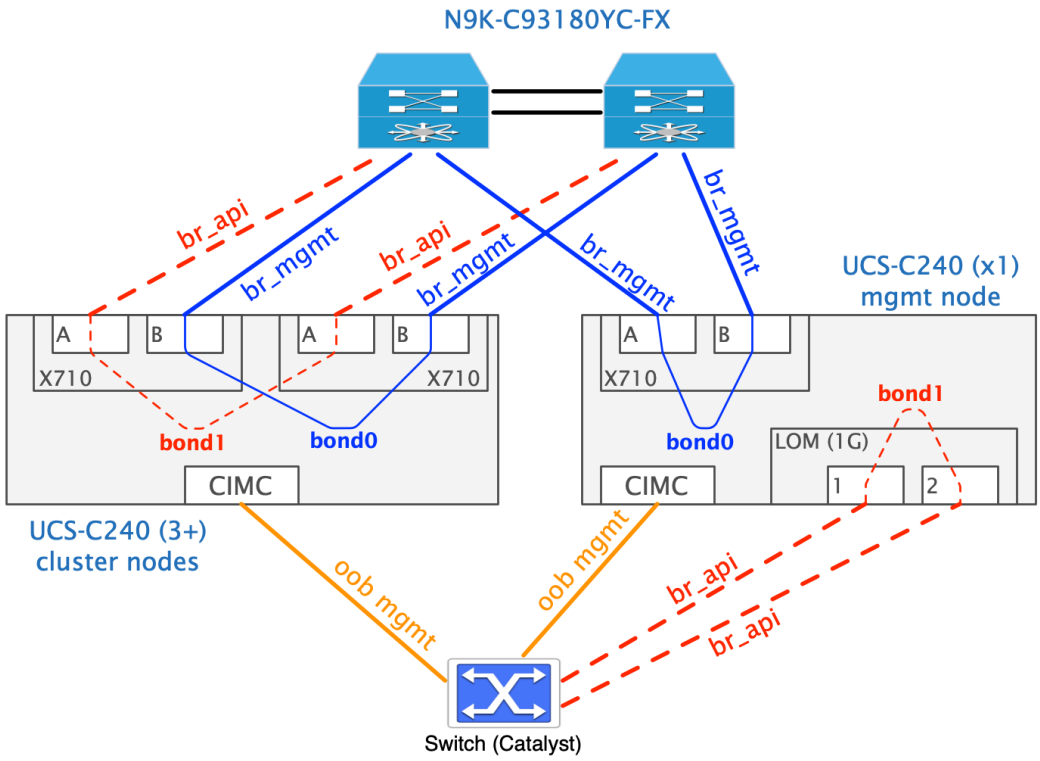
Network Topologies

Network Topologies for HA CVIM-MON

- UCS C-Series Network Topology
  - NXOS Switch Configuration for CVIM MON HA UCS nodes
- Quanta (QCT) Network Topology
  - NXOS Switch Configuration for CVIM MON HA QCT 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.



NXOS Switch Configuration for CVIM 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 correspoding NXOS switch configuration of the of the port A and B links for each of the UCS nodes that make up the CVIM MON HA cluster.

The NXOS Switch configuration connected to Intel NIC port A on the UCS nodes that make up the CVIM 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
```

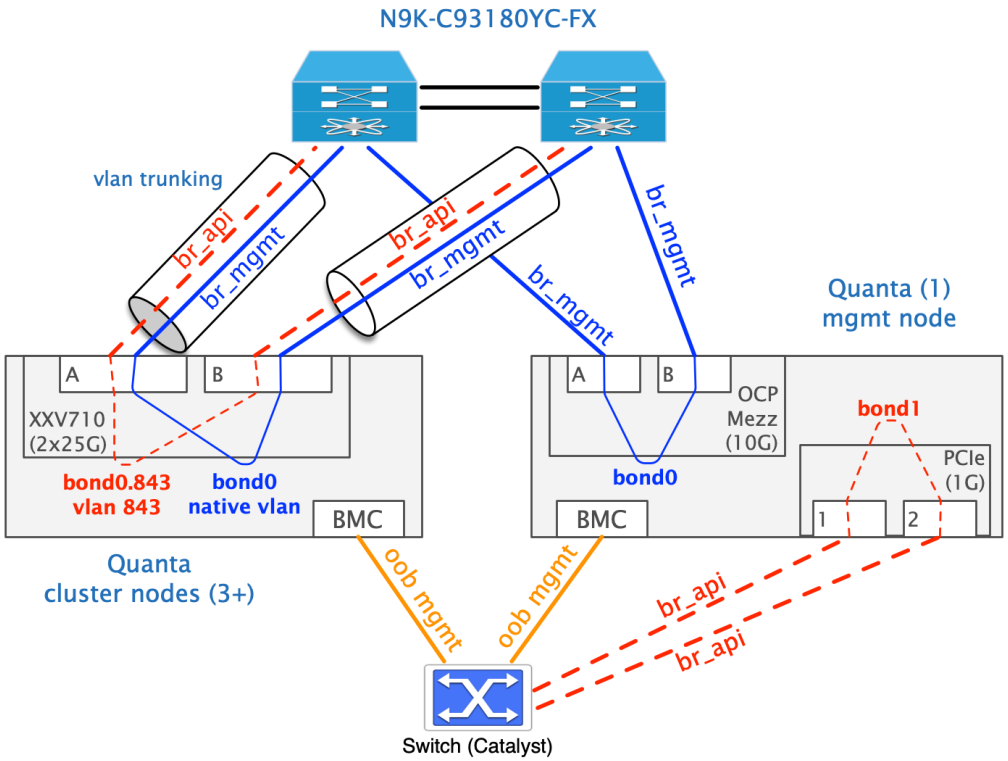
The NXOS Switch configuration connected to Intel NIC port B on the UCS nodes that make up the CVIM MON HA cluster:

```
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
```

Quanta (QCT) Network Topology

The cluster nodes are connected to the ToR switches from the Intel XXV710 card ports as shown below. The *br_api* and *br_mgmt* interfaces are mapped on two different VLANs sharing the same physical links that are connected to the dual N9K TORs using VLAN trunking.



The management node uses the OCP Mezz NIC for *br_mgmt* (2x10G on VLAN 843) and the PCIe NIC for *br_api* (2x1G on native VLAN). The two *br-api* links of the management node are wired to the OOB switch.

NXOS Switch Configuration for CVIM MON HA QCT nodes

It is assumed that switches are preconfigured in virtual port channel mode. Listed below are the specific configuration of the port A and B links for each of the Quanta nodes that make up the CVIMMON HA cluster.

The NXOS switch configuration connected to the 25G Intel NIC of the Quanta nodes that make up the CVIMMON HA cluster:

```
interface port-channel <pc_id_3>
  description mgmt and API interface
  switchport
  switchport mode trunk
  switchport trunk native vlan <mgmt_vlan>
  switchport trunk allowed vlan <mgmt_vlan,api_vlan>
  no lacp suspend-individual
  vpc <pc_id_3>

interface Ethernet1/<id_3>
  description mgmt and API interface ports
  switchport
  switchport mode trunk
  switchport trunk native vlan <mgmt_vlan>
  switchport trunk allowed vlan <mgmt_vlan,api_vlan>
  spanning-tree port type edge trunk
  channel-group <pc_id_3> mode active
  no shutdown
```

NXOS Switch Configuration for Management Node (QCT or 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 or Quanta-based management node of CVIMMON HA cluster.

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

```
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 Guid*e.

Architecture

Overview of HA CVIM-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 CVIM-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 CVIM-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 CVIM-MON 4.0.0 compared to HA CVIM-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 Cisco 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 CVIM-MON , see External Servers

Installation Modes

Installation Modes for HA CVIM-MON

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

- Connected Mode of Install
- Air-Gapped Install using USB
- Air-Gapped Installation Using Software Delivery Server

CVIM-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 Cisco 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 CVIM-MON install. You must provide the following information in the *setup data* file to fetch artifacts from cvim-registry.com:

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

Air-Gapped Install using USB

The following procedure describes how to download the Cisco NFVI installation files onto a USB drive of the staging server with Internet access. You can use the USB to load the Cisco NFVI 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 Cisco VIM installation files onto the USB drive. Once downloaded, you can copy the installation files onto the management node from a USB drive.

1. Fetching artifacts to the staging server:
   a. 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/.

   b. Log into Cisco 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://cvim-registry.com/mercury-releases
/cvim34-rhel7-osp13/releases/<releaseid>/getartifacts.py
Change the permission of getartificats.py chmod +x getartifacts.py
```

   c. 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>
```

d. 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
```

e. 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
```

f. 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
```

g. 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 CVIM MON HA management node, use the prepared USB stick and complete the following steps:

   a. Insert the USB stick into the management node drive, after you install the *buildnode.iso.*
   b. 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 install HA CVIM MON pod remain the same.

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

3. Configure the setup data file:

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

Configuration of Setupdata File

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

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

1. Install the management node with build node ISO.
2. Add the following fields in the HA CVIM MON *setup data* file.

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

HA CVIM 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 Cisco 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_USERNAME** and **SDS_REGISTRY_PASSWORD**. Hence, it is recommended that you have a read-only user on the Cisco VIM pod.

Setup File

Configuring Setup Data File for HA CVIM-MON

- Overview
- Argus Bare Metal Configuration Parameters
- General HA CVIM-MON Cluster Parameters
- HA CVIM-MON Stack Parameters
    - Regions, Metros, and Pods
    - SNMP
    - Syslog Export
    - LDAP Support for Grafana (CVIM MON Stacks)
    - LDAP Support for Grafana (CVIM 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 CVIM-MON management node.

Use the setup data file that describes all the parameters of HA CVIM-MON cluster and the list of all monitored Cisco VIM pods, for the installation of the HA CVIM-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 Cisco VIM target pods.

HA CVIM-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 CVIM-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:

- Unique cluster name to identify the current cluster
- IPv4 or dual stack IPv4+IPv6 selection for the cluster deployment
- List of all cluster nodes and for each node:
    - 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 CVIM-MON Cluster Parameters

Configure the general parameters of the HA CVIM-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 CVIM-MON services
- Virtual router ID for Virtual Router Redundancy Protocol (VRRP)
- 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 CVIM-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 Cisco VIM pods

You must configure each Cisco 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 CVIM-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 CVIM-MON alerts logs to be sent to one or up to 4 remote syslog servers. When SYSLOG_EXPORT is configured, you must configure SNMP.
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)

Example:

```
cvim_mon_stacks:
- name: mystack
  SYSLOG_EXPORT:
    remote_hosts:
      - host: 10.10.10.10
        port: 514
        facility: local5
      - host: 10.10.10.11
        port: 614
        facility: user
```

LDAP Support for Grafana (CVIM 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:

1. Domain mappings
2. 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:

```
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 (CVIM MON HA Cluster Monitor)

CVIM MON HA cluster itself is monitored by default on all CVIM MON HA pods. Grafana, Prometheus, and Alertmanager are installed in an isolated cvimmon-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 cvimmon-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.

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 CVIMMON > *Monitoring External Servers Using CVIM MON*

HA CVIM-MON Installer

Installation and Installer CLI

- Overview
- General installer CLI
- HA CVIM-MON Operational State
- Operations and Operational State History
- Installation Procedure
- Installing HA CVIM-MON
- Installation Failure

Before you Begin

Before using the HA CVIM-MON installer, see the following:

- Installation Modes for HA CVIM-MON
- Configuring Setup File for HA CVIM-MON

Overview

Following are the high-level steps for installing HA CVIM-MON cluster:

- Perform hardware installation and network wiring (described earlier)
- Install the management node
- Install the HA CVIM-MON cluster

The procedure to install the HA CVIM-MON management node is similar to the Cisco VIM management node installation. To install the management node, see Management Node on Quanta Servers and Management Node on UCS Servers. At the prompt, you must select HA CVIM-MON management node to indicate that you are installing a management node for the HA CVIM-MON cluster.

This section describes the procedure to install an HA CVIM-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 CVIM-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.

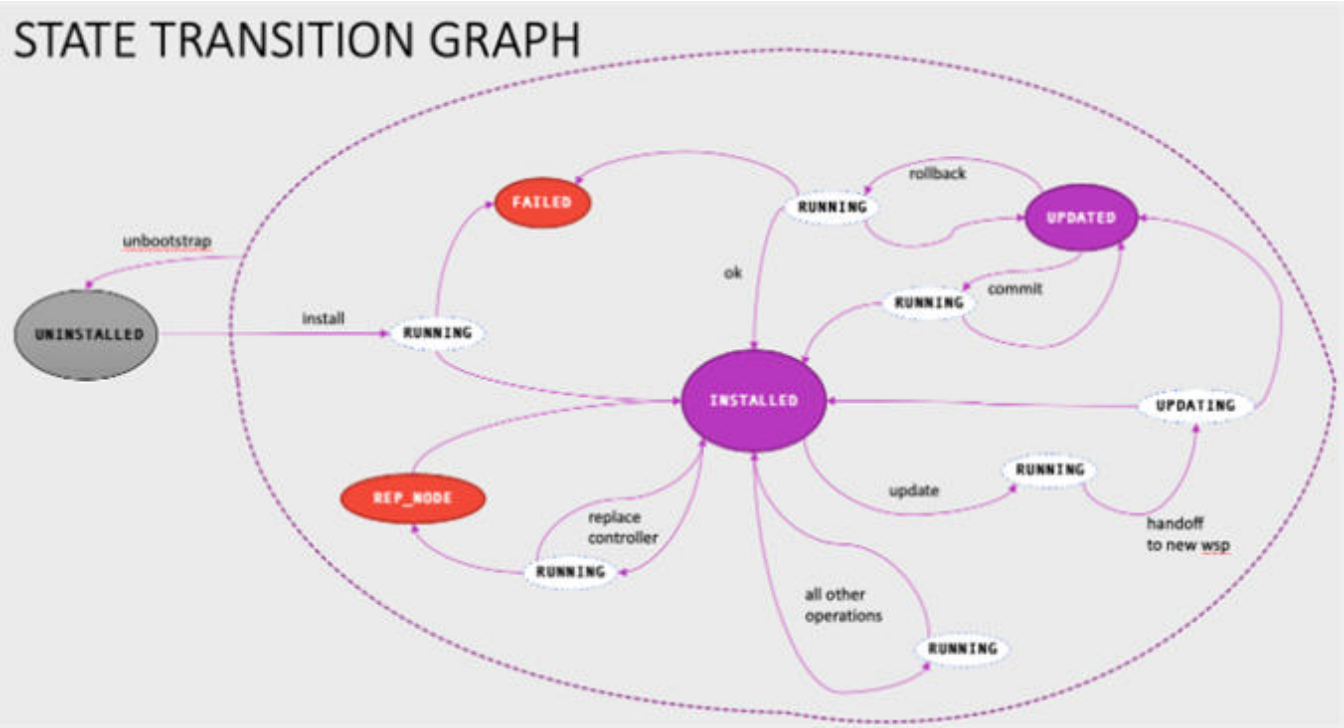
The *k8s_runner* CLI can be executed from any location on the management node.

To see all available arguments, simply type k8s_runner *--help.*

HA CVIM-MON Operational State

A state machine tracks the current state of the HA CVIM-MON cluster on the management node, and determines the operations that can be performed at any time.

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 Cisco 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 new updated version of HA CVIM-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 CVIM-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:

- 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 anytime 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                          |
|  46 | 2021-02-01 14:30:04 |    10   | 0:00:35 | INSTALLED | DELETE_CVIM_POD |  1   | Success
|                                          |
|  45 | 2021-02-01 14:29:29 |    10   | 0:00:00 | INSTALLED | DELETE_CVIM_POD |  1   | Running |
step=VALIDATION                          |
|  44 | 2021-02-01 14:29:29 |    10   |         | INSTALLED | DELETE_CVIM_POD |      | Running | args=--delete-
cvim-pod --setupfile ... |
|     |                     |         |         |           |                 |      |
|                                          |
|  43 | 2021-02-01 14:29:28 |    9    | 0:00:00 | INSTALLED |   GET_ENDPOINT  |      | Success | log=/var/log
/cvimmonha/...              |
|  42 | 2021-02-01 14:29:28 |    9    |         | INSTALLED |   GET_ENDPOINT  |      | Running | args=--get-
endpoint                    |
+-----+---------------------+---------+---------+-----------+-----------------+------+---------
+----------------------------------------+
```

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 CVIM-MON cluster with a candidate *setup_data.yaml* file, and run the HA CVIM-MON installation operation using the *k8s_runner* CLI. This install operation executes the following steps:

1. Validation
   Verifies the hardware and software configuration.
2. Bootstrap Infra
   Prepares the management node for CVIM MON HA installation.
3. Setup Argus
   Prepares bare metal installation.
4. Argus bare metal
   Installs and configures the operating system on all configured cluster nodes.
5. CVIM-MON Infra
   Prepares the management node for Kubernetes and application installation.
6. Kubernetes Provisioner
   Installs the Kubernetes infrastructure.
7. Helm Infra
   Installs the HA CVIM-MON application.

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 CVIM-MON

The normal procedure to install HA CVIM-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>

                        ############################
                         CVIM 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 Cvim-Mon Target Nomenclature  |  PASS  | None  |
| Check duplicate Cvim-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 CVIM-MON-HA      |  PASS  | None  |
+-----------------------------------+--------+-------+
[1/5][VALIDATION: INIT]                                                  [   \   ]    0min 50secs
...
<snip>
```

The installation is considered complete, when the HA CVIM-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.

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 *Installed* state, fix the errors causing the failure and rerun 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.

Resources

Resources for Managing HA CVIM-MON Cluster

- Configuration File and Secrets
- List Service Endpoint URLs (--get-endpoint)
- Kubernetes Resources

You can manage your HA CVIM-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:

```
Prometheus-Password-cvimmon-monitor(Username:admin):YyZM5f3DdyCKxklw1vIN4i0l0M2EoRbkjb+UKm0Sa5Y=
Grafana-Password-cvimmon-monitor(Username:admin):59QRzzo+PEedz8MDfdX26+DoaMJ/OgVgoqGWdUhDS78=
Grafana-Password-cvimmon-monitor(Username:cvimon):Hhp2TZJNZCu28pIzW3F6huBKtIdH1VedtZxaC7P9dFs=
Grafana-Password-stack1(Username:admin):h72DhjEnVS/Rr4nFCZmmxKRmuK/t7qjyZJJrFbTyCtM=
Grafana-Password-stack1(Username:cvimmon):J72ChjJnVS/gh3yFCZmmxKRmuK/t9qjyZJJrFbJYCtN=
Prometheus-Password-stack1(Username:admin):1Ph5AI8JUhiHgX0vjHB3W0Wzgjy2jWfiC5egAQJuuIs=
Grafana-Password-stacktsdb(Username:admin):N/ABGdX0ym5VhJ7Q/k8TO1oeqRXuzvbOmU9JeunA1As=
Grafana-Password-stacktsdb(Username:cvimmon):X/Sjcjasa9hsda23/k8TO1keqRXuzvb9mU9JfunA1Cs=
Prometheus-Password-stacktsdb(Username:admin):F8SPq+1qUSKM08EvlL+bTbL6RU8BI8Qcz/Yjzi0s7gw=
```


From Cisco 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 Cisco VIM, 3.4.6 execute *--regenerate-secrets* option with k8s_runner.
For new stacks, use *--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-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://cvimmon-prometheus-scalestack.cisco.com        | 2001:420:293:
2422::168 |
| prometheus-stackresync-server     | https://cvimmon-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
thanos-alertmanager	View alerts, silence alerts	common / browser or REST API
prometheus-cvimmon-monitor-server	Access prometheus server in charge of the HA CVIM-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

All URLs are assigned the same VIP address. The HTTP server at this IP address reroutes the traffic to the appropriate service container, based on the target FQDN provided in the URL.

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 one 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)
- 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 CVIM-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 CVIM-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-master cvmonq1 --setupfile ~/Save/new_setup_data.yaml
```

Adding a Worker Node

You can add a worker node to the HA CVIM-MON cluster in the following two ways:

- Pre-define in the inital *setup data* file so that the worker node is installed when you install the HA CVIM-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 CVIM-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 CVIM-MON cluster.
- The worker node hardware and the network hardware must conform to the same BOM defined by the HA CVIM-MON master nodes.
- You must apply the same networking schema of the HA CVIM-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 CVIM-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

                        ############################
                         CVIM 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!                                                [   -   ]    0min 14secs
+-----------------------------------+--------+-------+
| Rule                              | Status | Error |
+-----------------------------------+--------+-------+
| Schema Validation of Input File   |  PASS  | None  |
| ...                               |  ...   | ...   |
| 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...   [ DONE! ]    0min 0sec
[3/5][ARGUS_BAREMETAL: INIT]                                                 [ DONE! ]    0min 1sec
[3/5][ARGUS_BAREMETAL: Validating Argus Configs..]                           [ DONE! ]    0min 2secs
[3/5][ARGUS_BAREMETAL: Initiating Argus Baremetal node operation..]          [ DONE! ]    0min 3secs
[3/5][ARGUS_BAREMETAL: Initiating Node deploy: matrix04..]                   [ DONE! ]    0min 3secs
[3/5][ARGUS_BAREMETAL: Servers are pxe-booting..Takes Time!!]                [ DONE! ]    10mins 26secs
[3/5][ARGUS_BAREMETAL: Server(s) deploy operation finished: Success]         [ DONE! ]    10mins 33secs

ADD_WORKER [ARGUS_BAREMETAL] step completed [Success]

[4/5][COMMON_CVIM_MON_INFRA: Server(s) deploy operation finished: Success]    [ DONE! ]    0min 0sec
[WARNING]: Found variable using reserved name: name                          [   |   ]    0min 1sec
[4/5][COMMON_CVIM_MON_INFRA: INIT]                                           [ DONE! ]    0min 2secs
[4/5][COMMON_CVIM_MON_INFRA: update-known-hosts-Set backup_name fact of ...   [ 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
[...]
[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
```

Removing a Worker Node (--remove-worker)

You can remove a worker node from the cluster if required. An HA CVIM-MON cluster can operate without any worker nodes. You can also remove all the worker nodes. After this operation, Cisco VIM deletes the node from the HA CVIM-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:

```
# 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

                        ############################
                         CVIM MON HA ORCHESTRATOR
                        ############################

[1/4][VALIDATION: INIT]                                                      [   -   ]    0min 2secs
```

```
  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                       |        |       |
+-----------------------------------+--------+-------+

  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           |
+-----------------------------------+--------+----------------+
[1/4][VALIDATION: INIT]                                                [ DONE! ]    0min 58secs

REMOVE_WORKER [VALIDATION] step completed [Success]

[2/4][CLEANUP: INIT]                                                   [ DONE! ]    0min 3secs
[2/4][CLEANUP: check-kubernetes-node-Check if the node is present]     [ DONE! ]    0min 3secs
[2/4][CLEANUP: check-kubernetes-node-Get the list of masters]          [ DONE! ]    0min 6secs
[2/4][CLEANUP: delete-kubernetes-node-Check if the node is present]    [ DONE! ]    0min 17secs
[2/4][CLEANUP: delete-kubernetes-node-drain nodes]                     [ DONE! ]    0min 18secs
[WARNING]: Could not match supplied host pattern, ignoring: ha_master  [   \   ]    0min 18secs
[2/4][CLEANUP: delete-kubernetes-node-delete nodes]                    [ DONE! ]
[2/4][CLEANUP: delete-kubernetes-node-kubectl get nodes must be free of ...   [ DONE! ]    0min 25secs

REMOVE_WORKER [CLEANUP] step completed [Success]

[3/4][ARGUS_BAREMETAL: delete-kubernetes-node-kubectl get nodes must be ...   [ DONE! ]    0min 0sec
[3/4][ARGUS_BAREMETAL: INIT]                                           [ DONE! ]    0min 1sec
[3/4][ARGUS_BAREMETAL: Validating Argus Configs..]                     [ DONE! ]    0min 2secs
[3/4][ARGUS_BAREMETAL: Initiating Argus Baremetal node operation..]    [ DONE! ]    0min 3secs
[3/4][ARGUS_BAREMETAL: Initiating Node delete: matrix04..]             [ DONE! ]    0min 14secs
[3/4][ARGUS_BAREMETAL: Server(s) delete operation finished: Success]   [ DONE! ]    0min 21secs

REMOVE_WORKER [ARGUS_BAREMETAL] step completed [Success]

[4/4][GENERATE_INVENTORY: Server(s) delete operation finished: Success]  [ DONE! ]    0min 0sec
[WARNING]: Found variable using reserved name: name                    [   \   ]    0min 0sec
[4/4][GENERATE_INVENTORY: INIT]                                        [ DONE! ]    0min 1sec
[4/4][GENERATE_INVENTORY: generate-inventory-Check if Argus Site File is...   [ DONE! ]    0min 2secs

REMOVE_WORKER [GENERATE_INVENTORY] step completed [Success]

2021-01-12 16:53:06,771 INFO Runner - Backing up setup_data.yaml
2021-01-12 16:53:06,799 INFO Runner - Operation REMOVE_WORKER completed successfully
2021-01-12 16:53:06,800 INFO Runner - Performing autobackup...
2021-01-12 16:53:06,828 INFO Runner - Executing autobackup to /var/cisco/cvimmonha_autobackup
/cvimmonha_autobackup_3.9.8_2021-01-12_16:53:06, disconnected install=False
2021-01-12 16:53:06,828 INFO Runner - Validating the current workspace...
2021-01-12 16:53:06,828 INFO Runner - Creating the backup...
2021-01-12 16:53:06,911 INFO Runner - Compressing the backup...
2021-01-12 16:53:07,114 INFO Runner - Backup completed successfully: /var/cisco/cvimmonha_autobackup
/cvimmonha_autobackup_3.9.8_2021-01-12_16:53:06.tgz
```

```
The logs for this run are available at /var/log/cvimmonha/remove_worker_2021-01-12_16:51:17
```

Regenerating Certificates

You can regenerate Kubernetes, ETCD, and application certificates using HA CVIM-MON.

Kubernetes Certificates (--renew-k8s-certs)

To regenerate Kubernetes certificates, use the following command:

```
[root@queensland ~]# k8s_runner --renew-k8s-certs

2021-01-12 15:35:59,070 INFO Runner - Executing: RENEW_K8S_CERTS (args: --renew-k8s-certs)
2021-01-12 15:36:01,873 INFO Runner - Running generic handler for RENEW_K8S_CERTS

                       ############################
                        CVIM 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

RENEW_K8S_CERTS [VALIDATION] step completed [Success]

[2/2][KUBERNETES_PROVISIONER: INIT]                                      [ DONE! ]    0min 11secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check Cluster State]    [ DONE! ]    0min 14secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period...  [ DONE! ]    0min 39secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Renew Validity Peri...  [ DONE! ]    1min 5secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Get Validity Period...  [ DONE! ]    1min 30secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check if all Pods i...  [ DONE! ]    1min 39secs
[2/2][KUBERNETES_PROVISIONER: kubernetes-renew-certs-Check Cluster State]    [ DONE! ]    1min 43secs
[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
[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

                       ############################
                        CVIM 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

RENEW_ETCD_CERTS [VALIDATION] step completed [Success]

[2/2][KUBERNETES_PROVISIONER: INIT]                                      [ DONE! ]    0min 19secs
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Configure | Check if etcd clu...   [ DONE! ]    0min 22secs
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Get Validity Period for all k...   [ DONE! ]    0min 33secs
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Renew Validity Period for all...   [   |   ]    0min 44secs
[2/2][KUBERNETES_PROVISIONER: etcd_upgrade-Get Validity Period for all k...   [ DONE! ]    0min 56secs

RENEW_ETCD_CERTS [KUBERNETES_PROVISIONER] step completed [Success]

2021-01-12 15:42:49,068 INFO Runner - Operation RENEW_ETCD_CERTS completed successfully
The logs for this run are available at /var/log/cvimmonha/renew_etcd_certs_2021-01-12_15:41:37
```

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_CERTS

                        ############################
                         CVIM MON HA ORCHESTRATOR
                        ############################

[1/2][VALIDATION: INIT]                                                    [   -   ]    0min 2secs

  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

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-MON Ingre...   [ DONE! ]    3mins 21secs
[2/2][HELM_INFRA: prometheus-CVIMMON-stackresync->Replace CVIM-MON Ingre...   [ DONE! ]    3mins 21secs
[2/2][HELM_INFRA: prometheus-CVIMMON-stackresync->Delete all Cert Sensit...   [ DONE! ]    3mins 28secs

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
/cvimmonha_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...
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 CVIM-MON, you can provide trusted CA signed x509 certificates which are used to access the Prometheus, Grafana, and Alertmanager applications in the HA CVIM-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://cvimmon-prometheus-scalestack.cisco.com        | 2001:420:293:
2422::168 |
| prometheus-stackresync-server     | https://cvimmon-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
```

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 CVIM-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 CVIM-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 CVIM 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:

1. Regenerate Secrets
2. Custom Secrets

View Secrets (--list-secrets)

To view passwords for each application, first execute *list-secrets* command to fetch the secret keys.

```
[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) | 4n6AVQ85wbWgDwqQlRSp/XqpEASAN02ieoICDGL+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

                        ############################
                         CVIM MON HA ORCHESTRATOR
                        ############################

[1/2][VALIDATION: INIT]                                                  [   |   ]    0min 3secs

  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

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>"
```


Ensure that you use the same key name as shown in the output including *Username:admin,* otherwise the command is not executed.

Password Policy

1. Only alphanumeric and special characters are allowed for secrets.
2. Passwords must contain at least one letter, one special character, a digit, without blank spaces.
3. Password length must be >=8 and <= 44.
4. 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)

                        ############################
                         CVIM 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
[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 CVIM-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:

1. Copy the CVIM-MON HA setupdata from */root/openstack-configs/* into a local location.
2. 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.
- 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/*

3. 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>
```

Stack Operations

Stack Operations

- Overview of HA CVIM-MON Stacks
- Adding a Stack (--add-stack)
- Deleting a Stack (--delete-stack)
- Adding, Updating, and Deleting Cisco VIM Pod Targets in a Stack
    - Adding a Target Cisco VIM OpenStack Pod (--add-cvim-pod)
    - Reconfiguring a Cisco VIM OpenStack Pod (--reconfigure-cvim-pod)
    - Deleting a Target Cisco VIM OpenStack Pod (--delete-cvim-pod)

Overview of HA CVIM-MON Stacks

An HA CVIM-MON stack is a set of containers responsible for collecting metrics from a set of Cisco 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 Cisco VIM pods. In most deployments, this single stack is sufficient to monitor all Cisco 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 Cisco 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 Cisco VIM pods.

Each stack runs two independent instances of Prometheus servers that collect metrics from the same set of Cisco 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 Cisco 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 Cisco 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 Cisco 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.

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 Cisco 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 CVIM pod targets as CVIM 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 Cisco 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 stackss 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 Cisco VIM Pod Targets in a Stack

The Prometheus servers in each stack are configured to pull metrics from the list of configured target Cisco 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 Cisco 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 Cisco VIM pods as targets once they are configured or reconfigured with CVIM-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 CVIM-MON cluster. Following are the details of *setup_data.yaml* for a Cisco 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.

The following Cisco VIM pod information are required to monitor the Cisco VIM pod from an HA CVIM-MON cluster:

Parameter Name	Description	Pod Management Node Information Location
ip	Cisco 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:* *ciscovim list-secrets* --getpassword *CVIM_MON_PROXY_PASSWORD*
cert	Indicates the pod CA certificate file	*/root/openstack-configs/haproxy-ca.crt*

In case of an IPv6 HA CVIM-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 CVIM-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 Cisco VIM pod to the */root/cvimha_certs/* directory on the HA CVIM-MON cluster management node and with a unique name.

In case where all the Cisco 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 Cisco VIM OpenStack Pod (--add-cvim-pod)

You can execute the *add-cvim-pod* operation after adding 1 or more new Cisco 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 Cisco 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-cvim-pod* operation using the following command:

```
# k8s_runner --add-cvim-pod --setupfile <path_to_candidate_setup_data_file>
```

Reconfiguring a Cisco VIM OpenStack Pod (--reconfigure-cvim-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 Cisco VIM OpenStack Pod (--delete-cvim-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>
```

Overview of Stacks

External Servers

Prerequisites for Monitoring External Servers Using HA CVIM MON

You can monitor the external servers such as standalone Linux servers that are not managed by any Cisco VIM pod, using an HA CVIM-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:

- Servers must be reachable from the Prometheus server in the HA CVIM-MON stack.
- Servers must run on hardware that is similar to the CVIM Management Node BOM.
- Servers must run the CVIM Management node ISO or CentOS 7.7.
- Servers must have the Telegraf agent provided by the HA CVIM-MON stack installer.
- Servers must run in the same site as the HA CVIM-MON cluster.

Prometheus collects the server metrics over unauthenticated and unencrypted HTTP connections on port 9273.
Cisco VIM distinguishes the metrics collected from external servers from the Cisco VIM pod metrics by a node_type label value of external. By default, Cisco 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 CVIM-MON stack monitors external servers, see CVIM Monitor

Updating Nodes

Updating Software of HA CVIM MON Nodes

- Updating to New Version
- Rolling Back to Previous Version
- Committing Updated Version

You can update the software of HA CVIM-MON nodes using the following three actions:

Action	Description
Update	Gets the new software version and updates the HA CVIM-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 CVIM-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 CVIM-MON controller and worker nodes.
- Updates the HA CVIM-MON stacks.

Following are the steps to update the software of HA CVIM-MON nodes:

1. Get the *mercury-installer.tar.gz"* file corresponding to the new version, based on the installation type.
2. 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 rollback to the previous software version, if there are problems after the software update.

To rollback the update, execute the following command from any location:

```
k8s_runner --rollback
```

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 CVIM-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 CVIM 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

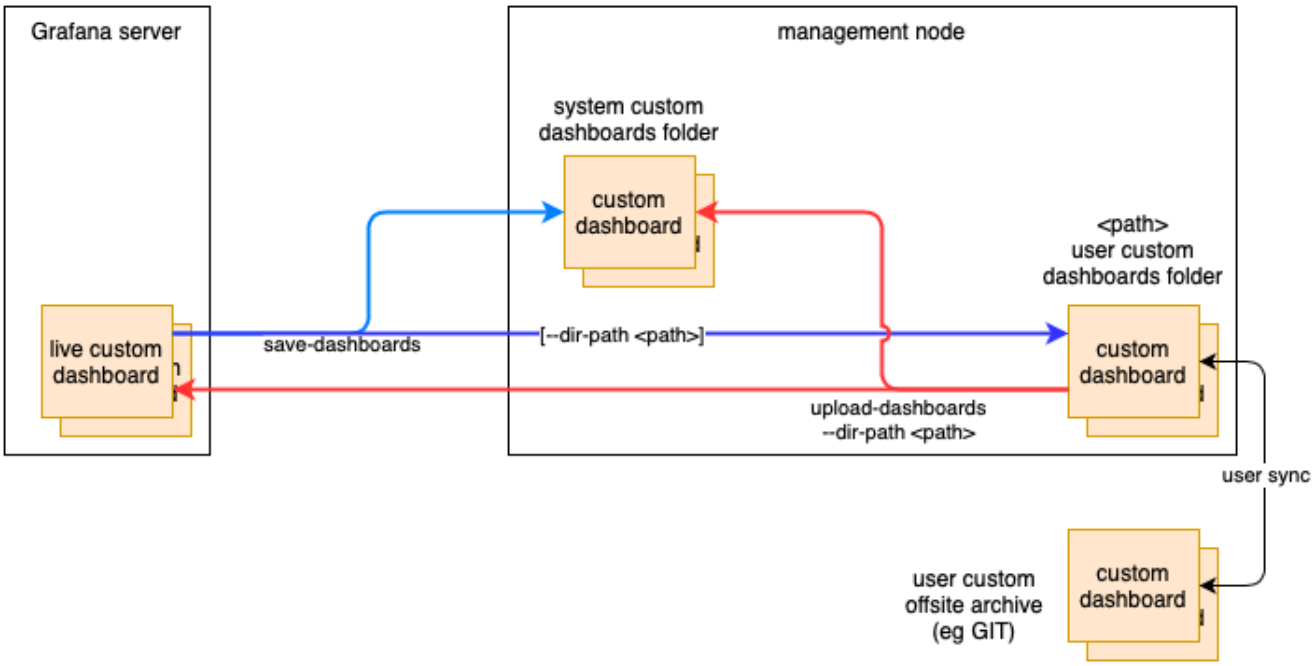
Overview

HA CVIM-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 CVIM-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.
```

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
Unchanged	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.

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
- 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 CVIM-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

1. Create a version controlled environment for saving and managing versions of your custom dashboards, for example, create a git repository.
2. Create a new custom dashboard from the Grafana UI with the admin user.
3. Save the new dashboard in Grafana UI.
4. List the custom dashboards and verify whether a new one is detected:

   ```
   k8s_runner --cvimmon-custom-dashboards --list-dashboards
   ```

5. Create an empty directory on the management node, for example, */root/cdash.* This directory must exist but must be empty.
6. 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
   ```

7. Save a copy of the dashboard in the */root/cdash* folder:

   ```
   k8s_runner --cvimmon-custom-dashboards --save-dashboards --dir-path /root/cdash
   ```

8. Save and commit the content of */root/cdash* in the git repository.

Adding or Modifying Dashboards

1. Add dashboards or modify dashboards in Grafana (admin user).
2. Synchronize the Grafana server and management node by saving the new custom dashboard to the management node:

   ```
   k8s_runner --cvimmon-custom-dashboards --save-dashboards
   ```

3. Empty the user dashboard folder or create a new empty folder.
4. Save a copy of the dashboard in the */root/cdash* folder:

   ```
   k8s_runner --cvimmon-custom-dashboards --save-dashboards --dir-path /root/cdash
   ```

5. Finally save and commit the content of */root/cdash* in the git repository.

Uploading Custom Dashboards on Freshly Installed Cluster

1. Copy the custom dashboards from the git repository to the management node.
2. 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 CVIM-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, that 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.
- 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:

- 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 change-rules, but does not have value: remove the property from built-in alerting rules
- severity
- snmp_fault_code
- snmp_fault_source
- 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.*
```

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 CVIM-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, CVIM-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:

1. Create a proper custom alert manager configuration file:
    a. Create a custom alert manager configuration file.
    b. Edit the content using your favorite editor (see format below).
    c. Verify that the custom alert manager configuration file is valid using the provided tool.
2. 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, that is common for all stacks. It is not possible to apply different custom alert manager configurations for each stack.

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).

```
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:

1. Generic route for all alerts to the SNMP agent running on the management node.
2. 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 Cisco VIM configuration file setup_data.yaml.

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 the another alert having 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
```

Backup

Backing Up HA CVIM-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 CVIM-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 *--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
#
```

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 CVIM-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 CVIM-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-CVIM-MON) and set in HA CVIM-MON UNINSTALLED state.
- You must copy from your backup file storage the appropriate backup compressed file that was created from the same HA CVIM-MON cluster.
- 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

1. After ISO installation, copy the backup compressed file from the remote backup server to the management node, preferably under */var/cisco /cvimmon_backup* directory.
2. Execute the *k8s_runner* restore operation from any location:

   ```
   # k8s_runner --restore <path to the backup compressed file>
   ```

3. 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            |
   ```

4. 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 CVIM-MON HA Pod

Reinstallation of CVIM-MON HA Pod

Due to unforeseen circumstances, there might be a need to reinstall the CVIM MON HA pod with the same image version. To alleviate the need for reimaging the management node followed by re-installation of the CVIM 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 CVIM-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 CVIM 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:

   ```
   docker images
   ```

4. To re-run the Cisco VIM MON HA installation, use the command:

   ```
   k8s_runner --install --setupfile ~/Save/setup_data.yaml
   ```

CVIM MON HA Cluster Monitoring

CVIM MON HA Cluster Monitoring

- Overview
- Components of HA CVIM-MON Cluster Monitoring
- Cluster Monitoring Endpoint URL's
- Secrets for HA CVIM-MON Cluster Monitoring
- Grafana Dashboards
- Alerts

Overview

HA CVIM-MON has an inbuilt mechanism to monitor the cluster itself apart from monitoring the Cisco VIM pods. The HA CVIM-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.

1. Health of the cluster.
2. Resource utilization of the nodes.
3. Status of various Kubernetes objects.
4. Portworx (Storage) monitoring
5. Alerts for various conditions.

Components of HA CVIM-MON Cluster Monitoring

The HA CVIM-MON cluster monitor uses the following components to monitor the kubernetes cluster:

1. Kube-state-metrics.
2. cAdvisor
3. Node-exporter
4. Prometheus to collect metrics .
5. 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
```

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 CVIM-MON Cluster Monitoring

To get the secrets for the HA CVIM-MON cluster monitor, use the following commands:

```
Grafana Dashboard
# ./bootstrap/k8s-infra/k8s_runner.py --get-password "Grafana-Password-cvimmon-monitor(Username:admin)"

+-------------------------------------------------+-------------------------------------------+
| Secret Key                                      | Password                                  |
+-------------------------------------------------+-------------------------------------------+
| Grafana-Password-cvimmon-monitor(Username:admin) | vpUphAfQF6EMcZeWfGIA7fnXk8Gs7zqqFKo8npWjWLk= |
+-------------------------------------------------+-------------------------------------------+

Prometheus Dashboard
# ./bootstrap/k8s-infra/k8s_runner.py --get-password "Prometheus-Password-cvimmon-monitor(Username:admin)"

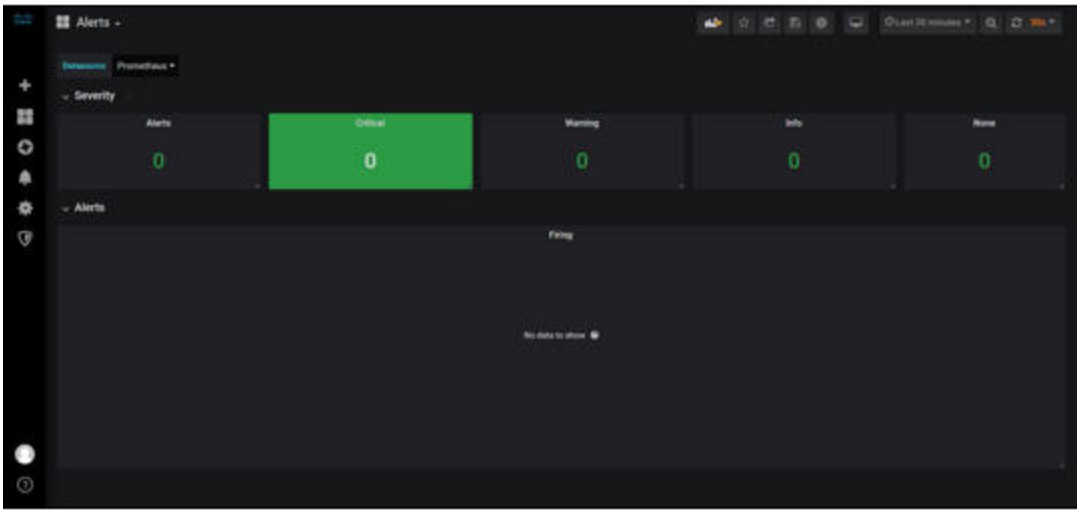
+-------------------------------------------------+-------------------------------------------+
| Secret Key                                      | Password                                  |
+-------------------------------------------------+-------------------------------------------+
| Prometheus-Password-cvimmon-monitor(Username:admin) | w7NTKssuaeOgKT/RB+VInZVR9vHGtn8BzpA8GDr8vow= |
+-------------------------------------------------+-------------------------------------------+
```

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:







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 CVIM Mon Upgrade

Cisco HA CVIM Monitor(HA CVIM-MON) supports upgrade to new releases. HA CVIM MON 3.4.6 release is the starting point for any HA CVIM MON upgrade to 4.2.x lineup. The HA CVIM 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 CVIM MON system. The upgraded to HA CVIM-MON system in 4.x provides enhanced cluster self-monitoring, unified grafana interface, unified quering/alerting across stacks and enhanced *k8s_runner cli*.

Comparison of HA CVIM-MON Component Versions

Component	3.4.x	4.2.x
Kubernetes Version	v1.15.2	v1.17.8
Helm Version	v2.8.0	v3.5.3
Prometheus	2.16.0	2.24.1
Alert manager	v0.20.0	v0.21.0
Grafana	6.7.4	7.2.0
Storage	Portworx	Minio
Thanos	-	v0.17.2
Stacks	Multiple Grafana for each Stack	Single Grafana for visualizing all stacks
Ldap, SNMP, SYSLOG	Per stack	Common

Upgrade Procedure

Pre-upgrade check

1. Plan for proper maintenance window and upgrade downtime. The upgrade downtime is proportional to the HA CVIM MON installation time (time may varies depending on HA cluster size).
2. Copy the *autobackup* directory from the management node to a safe location.
3. The management node should have sufficient disk space for prometheus metrics data storage.
4. The setup data for installation has to be manually updated to be compatible with target release version .
    a. Refer to the new setup data format *setup_data.yaml.CVIMMONHA.EXAMPLE* under */installer-xx/openstack* directory.
5. 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 CVIM MON from 3.4.x (3.4.6)

Perform the following steps to upgrade the HA CVIM MON pod from 3.4.x (x=6) to the current release of HA CVIM MON POD:

1. Copy the latest autobackup from the management node.

```
## Auto backup location
# cd /var/cisco/cvimmonha_autobackup
```

2. 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>
```

3. Re-image the HA management node with the ISO of current release version with the same IP address.
4. 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

5. Run the *prepare_cvimmon_upgrade* tool with the backup. The tool performs the snapshot of the prometheus TSDB from the backup files.

```
# prepare_cvimmon_upgrade --help
================================================================
                CVIMMON PREPARE UPGRADE TOOL
================================================================

usage: prepare_cvimmon_upgrade [-h] --storage_location STORAGE_LOCATION
                               --backup_location BACKUP_LOCATION

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

# prepare_cvimmon_upgrade --storage_location /root/data/ --backup_location /root/cvimmonha_autobackup_3.
4.6_2021-03-04_00\:21\:34/


================================================================
                CVIMMON PREPARE UPGRADE TOOL
================================================================

2021-03-04 15:47:33,930 INFO Runner - Preparing for Cvimmon Upgrade from 3.4.x to 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

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
```

6. Run the HA CVIM 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.

7. Once the install is complete, verify that HA CVIM MON cluster is up and running.

```
#  kubectl get nodes
#  kubectl get pods -A
```

8. 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 -s /root/data/snapshot_20210304_154733/


====================================================================
                CVIMMON COMMIT UPGRADE TOOL
====================================================================

2021-03-05 16:23:03,701 INFO Runner - Commit Tool for Cvimmon Upgrade from 3.4.x to 4.2
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

CVIMMON Snapshot Tool Completed Successfully!!

The logs for this run are available at /var/log/cvimmonha/20210305_162303_commit_cvimmon_upgrade
/commit_cvimmon_upgrade.log
```

9. Obtain the grafana endpoint using *k8s_run* command. Login to unified Grafana UI.

```
#  k8s_run --get-endpoint
```

SERVER MON

⚠ SERVER-MON feature support for HA CVIM MON is enabled as tech preview in CVIM 4.2.2

Enabling SERVER-MON on HACVIMMON POD (**Tech-preview only**)

You can enable SERVER-MON on an existing pod that is installed with Cisco VIM 4.2.2 or later, through the reconfigure option by extending the *setup_data.yaml* file with relevant information.

- SERVER_MON: Enables collecting UCS-C bare metal alerts into CVIM-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. SERVER_MON is a prerequisite for this option.

Example of *setup_data.yaml* section

```
SERVER_MON:
  enabled: true
  host_info:
     - ALL



## 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
```

Cisco VIM

Cisco VIM

- Installation Sequence
- Deployment of Management Node
- Cisco VIM Client Details
- Pod Reinstallation
- Control and Data Plane Testing
- Updating Cisco VIM Software

Installation Sequence

Installation Sequence

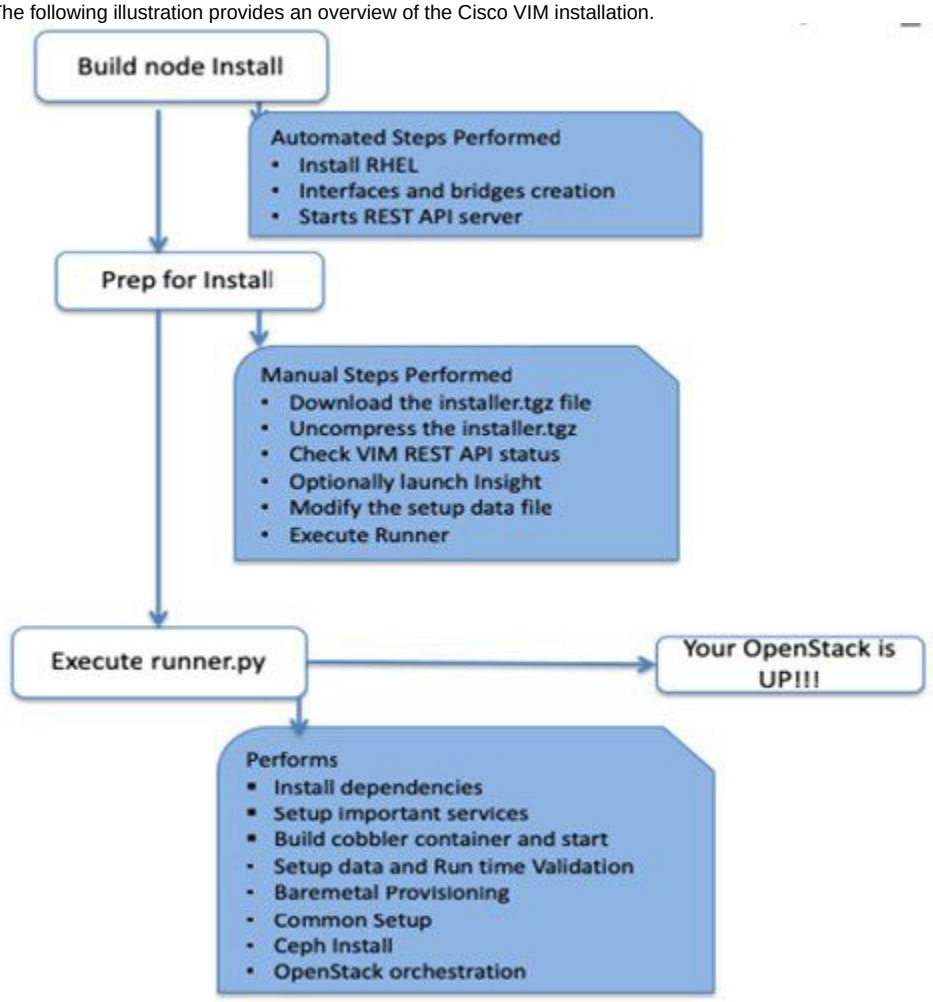
Before you install Cisco VIM, complete the procedures in Preparing for Cisco NFVI Installation and ensure that Cisco NFVI network infrastructure is set up. If your management node does not have internet access, complete the procedure in Installation Preparation Without Internet Access. You can download and install the Cisco VIM installation files from the Internet, Cisco 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.

Ciso 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 Cisco 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 Cisco VIM installer performs common steps across all the Cisco 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.
The following illustration provides an overview of the Cisco VIM installation.



If you have Cisco Unified Management (UM), complete only part of the Cisco VIM installation procedure and proceed to the Installing UM with Internet Access procedure to complete the configuration and setup of Cisco VIM. If you do not have Cisco VIM UM, configure Cisco VIM by editing the *setup_data.yaml* file as described in the Cisco VIM installation.

Deployment of Management Node

Deployment of Management Node

This procedure allows you to install Cisco VIM on a Cisco NFVI management node:

Before you begin

- You must get Cisco NFVI installation file download site credentials from your Cisco account representative.
  For management nodes with no Internet access, you need a USB drive containing the Cisco NFVI installation 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 Cisco 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.

1. If your management node does not have Internet access, use the prepared USB drive and complete the following steps:
   a. Insert the USB drive into the management node.
   b. 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.
2. If you are installing Cisco VIM Insight, navigate to Unified Management and complete the Cisco VIM Insight installation. If you are not installing Cisco VIM Insight, complete the following steps, irrespective of the fact that it is an air-gapped or connected installation.
3. Change to the installer directory by running the following command:

   ```
   cd ~/installer-<tag_id>
   ```

4. Create a directory (for example, ~/Save/) to store a copy of the *setup_data.yaml* file that configures the Cisco NFVI for your particular implementation.
5. Change to openstack-configs directory and copy the example Cisco 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.
6. With a yaml editor, modify the copied example *setup_data.yaml* file as the data setup file for your implementation. This includes both Cisco NFVI data and OpenStack parameters.
7. If you intend to run the cloud over TLS, see OpenStack Configuration for TLS certificate generation.
8. Run the installation:

   ```
   ciscovim --setupfile ~/Save/setup_data.yaml run
   ```

   After the installation is complete, you can view the installation logs at */var/log/mercury/<UUID>.*

Cisco VIM Client Details

Cisco VIM Client Details

Cisco VIM combines the CLI and API so that you can use the CLI or API installer transparently.

> ⚠ For a complete list of Cisco VIM REST API commands, see Cisco VIM REST API

Before using Cisco 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.
```

If the REST API server is not active, executing *ciscovim* shows the following error message:

```
ciscovim -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
```

To check whether the Rest API server state and launch directory are correct, use the following command:

```
./restapi.py --action status
```

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:

```
ciscovim --setup_file <setup_data_file_path> --perform 7 -y
```

From the CLI, enter the following command:

```
#ciscovim --help

usage: ciscovim [--setupfile <setupdata_file>] [--debug] <subcommand> ...

Command-line interface to the Cisco 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
    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                      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
    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
```

```
                              subcommands.

Optional arguments:
  --setupfile <setupdata_file>
  --debug                     Print debugging output

See "ciscovim help COMMAND" for help on a specific command.
To look at the help for a sub-command (e.g. run) execute the following:

# ciscovim help run
usage: ciscovim 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:
# ciscovim list-steps
Virtualized 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:

```
$ ciscovim run --perform 1,3 -y
```

During installation, you can use skip option to indicate which options to skip. For example:

```
$ ciscovim 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. *ciscovim install-status* indicates the current state of the installation process.

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:

1. Copy the *setup_data.yaml* from */root/openstack-configs/* directory to *~/Save/*

```
cd ~/installer-<xxx>
 ./unbootstrap.sh -k
```

> ⚠
> - If you use Auto-TOR with ACI API, execute *./unbootstrap.sh -sk.*
> - 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.

2. To verify that no docker containers are running, use the command:

```
docker ps -a docker ps -a
```

3. To verify that no docker images are present, use the command:

```
docker images
```

4. To setup RestAPI, use the command:

```
cd ~/installer-<xxx>/tools
./restapi -a setup
```

5. 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
```

6. To re-run Cisco VIM installation, use the command:

```
ciscovim run --setupfile ~/Save/setup_data.yaml
```

Control and Data Plane Testing

Control and Data Plane Testing

Cisco 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 Cisco NFVI compute nodes. VMTP performs ping connectivity, round trip time measurement (latency), and TCP throughput measurement for the following Cisco 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 Cisco 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>
   DNS_SERVER: <dns server for provider net>
   SEGMENTATION_ID: <segmentation id for provider net> # Needs to match a vlan defined under
PROVIDER_VLAN_RANGES
   IPV6_MODE: <"slaac" or "dhcpv6-stateless" or "dhcpv6-stateful"> # only for IPv6;
   VNIC_TYPE: <"direct" or normal># use value of direct for SRIOV, default is over virtio (value of normal)
   PHYSNET_NAME: <physnet_name># needed for SRIOV, entry has to be of the name: phys_sriov0, or phys_sriov1,
... phys_sriovn, where n is total num of SRIOV port-1
```


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 both at the same time. Ensure that you do not enable external network for IPv6.

Updating Cisco VIM Software

Updating Cisco VIM Software

The Cisco 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 Cisco NFVI software update runs serially, component-by-component, one node at a time, the service impact is minimal. 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. Cisco recommends 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.

⚠

- Cisco 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 Cisco 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.

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 Cisco VIM installation instructions. Differences between a software update and regular Cisco 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 Cisco 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 ciscovim 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.

Unified Management

Installing Cisco VIM Unified Management

- UM Overview
- UM with Internet Access
- UM with Cisco 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

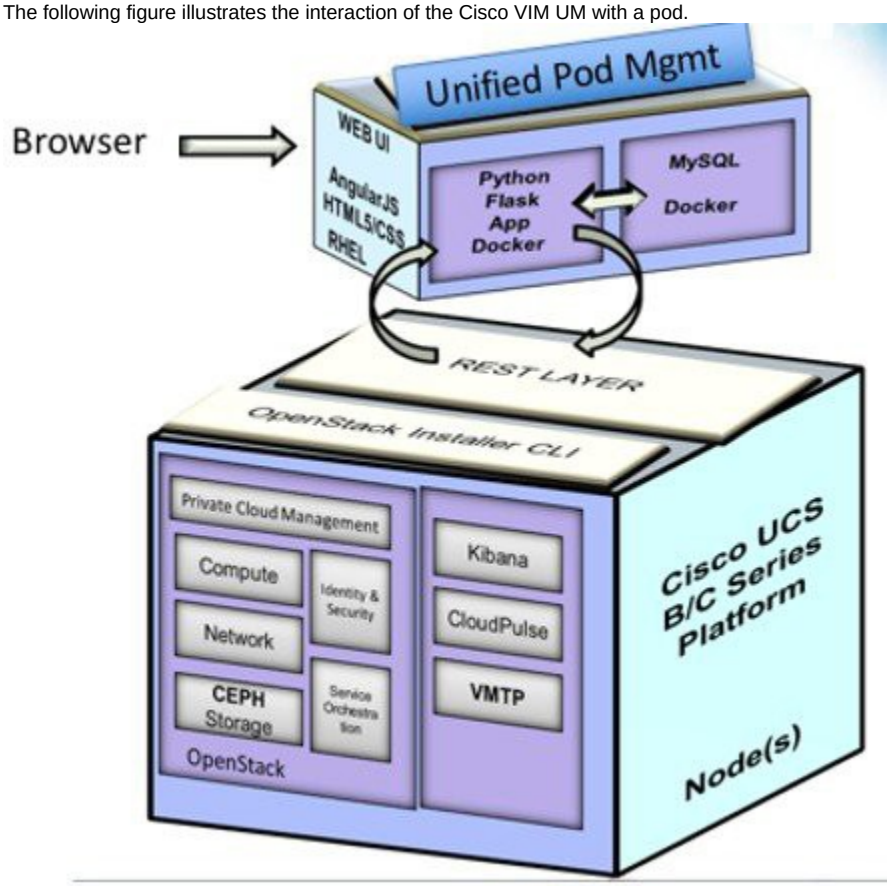
UM Overview

- Admin UI
- Pod UI
- UM OG/NG

Unified Management (UM) also known as Cisco 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. Cisco VIM Insight offers a single pane of glass service to provide deployment visualization and to manage multiple Cisco VIM pods, thereby reducing user-errors. Ensure that the UM is running on a standalone Cisco VIM supported BOM.

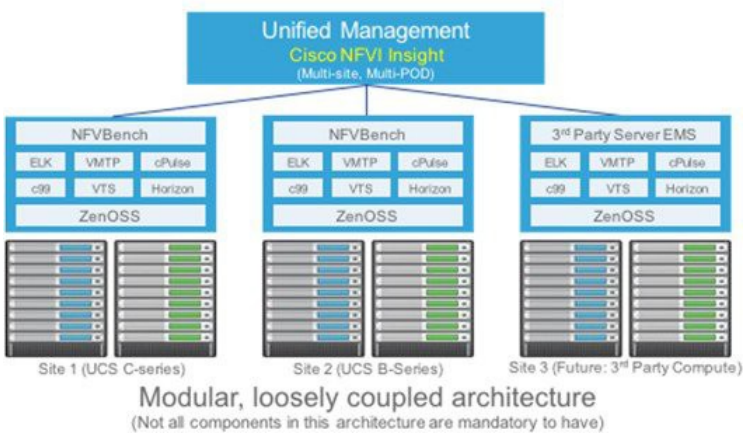
From Cisco VIM 4.0.0, only next-generation Unified Management (UM NG) is supported. Pods upgrading from Cisco VIM 3.4.6, must run with UM NG. For installation of UM NG, ensure that the INSIGHT_NG is defined as True in the *setup_data* of UM. For installation details, see UM with Internet Access

The following figure illustrates the interaction of the Cisco VIM UM with a pod.



The architecture of the Cisco 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. Cisco VIM UM is designed to operate in HA as an option.

UM is integrated with Cisco VIM REST layer, to support multi-tenancy with local RBAC support. The container-based UI platform is loosely coupled, and can help manage multiple Cisco 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 Cisco 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 Cisco VIM 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 Cisco 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 Cisco VIM 4.0.0, only next-generation UM is supported. Ensure that Cisco VIM 4.0.0 pods are managed with Cisco VIM 4.0.0 UM-NG application.

UM with Internet Access

Installing UM with Internet Access

- Overview
- Installation

Overview

Complete the following steps to install Cisco VIM Insight on the Cisco 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 Cisco 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.

1. 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
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 (M4/M5).

2. Run the following commands to copy the installer directory and the standalone *insight_setup_data_yaml.*
    a. 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>
```

b. 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
```

3. 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>'

# 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://cvim-registry.com
#
# b) disconnected: Node has NO connectivity to https://cvim-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]

################################################################################
```

```
#                          INSIGHT SUPER ADMIN                                 #
################################################################################
# User will be the default Super Admin of the system and can grant access to
# all other users getting registered to Cisco 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>

################################################################################
#                        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.>

# 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)
# 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=cvim-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=cvim-um-admin,ou=Roles'
UM_ADMIN_GROUP: <um_admin_group>

# Optional, LDAP Group-DN to be mapped as Insight Pod-Admins
# Defaults to 'cn=cvim-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=cvim-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=cvim-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)
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                            |
##-------------------------------------------------------------------------
## 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.
  -  vim_admin_username: non_root_admin_1
     vim_admin_password_hash: $6.....

     # vim admin with password hash and public key
  -  vim_admin_username: 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
  -  vim_admin_username: 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.
# 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
  - domain_name: corp_ldap2
    ldap_search_base: "dc=cisco,dc=com"
    ldap_cert_path: "<cert_path>"
    # 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>'
    # 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]
#############################################################################
```

4. Save the edited *insight_setup_data.yaml* file.
5. 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 !!
CISCO 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
```

```
The logs for this run are available at /var/log/insight/bootstrap/<date>_<time>

                ################################
                 CISCO 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  |
| CVIM/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 sss daemon]             [ DONE! ]    1min 23secs

.
.

Cisco 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 Cisco 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
```

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 to 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 Cisco VIM Insight Post Bootstrap Validation Checks section, to verify the bootstrap status of Cisco VIM Insight.

UM with Cisco VIM Software Hub

UM with Cisco VIM Software Hub

To reduce the logistics of the artifact distribution during an air-gapped installation, use Cisco VIM Software Hub. To download the artifacts to the Cisco VIM Software Hub server, follow the instructions available at Cisco 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 Cisco 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 Cisco 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 Cisco 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.

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'
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)
# 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=cvim-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=cvim-um-admin,ou=Roles'
UM_ADMIN_GROUP: <um_admin_group>

# Optional, LDAP Group-DN to be mapped as Insight Pod-Admins
# Defaults to 'cn=cvim-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=cvim-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=cvim-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=cvim-um-admin,ou=Group*

UM Without SMTP

Installing UM Without SMTP

By default, a SMTP infrastructure is required for Cisco VIM UM service.
For releases starting from Cisco 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.

To install UM without SMTP, follow the below steps:

1. Modify the *insight_setup_data.yaml* file and add the following key:

```
SMTP_MODE: False
```

2. Remove the following keys from the *insight_setup_data.yaml*:

```
INSIGHT_SMTP_SERVER
INSIGHT_EMAIL_ALIAS
INSIGHT_SMTP_PORT
INSIGHT_EMAIL_ALIAS_PASSWORD
```

3. 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.

4. 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.

5. 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.

UM Without Internet Access

Installing UM without Internet Access

- Overview
- Installation

Overview

Complete the following steps to install Cisco VIM Insight on the Cisco 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 Cisco 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 Cisco VIM installation files on the USB drive.
- Ensure that you have disabled the CentOS sleep mode.

1. On the staging server, use yum to install the following packages:
   a) PyYAML

   b) python-requests

   c) 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/.*
2. Access the Cisco VIM software download website using a web browser.
3. 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://cvim-registry.com/mercury-releases/cvim34-rhel7-osp13/releases/<releaseid>/getartifacts.py
# Change the permission of getartifacts.py
chmod +x getartifacts.py
```

4. 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 CVIM 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.

5. 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 example:

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

6. 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
Cisco Virtualized Infrastructure Manager Installation Guide, 3.4.x
97
Preparing for Cisco NFVI Installation
Installing Cisco VIM Software Hub in Air-Gapped Mode
ERROR: Layer file sha256:ea55b2fc29b95d835d16d7eeac42fa82f17e985161ca94a0f61846defff1a9c8 has a bad
checksum
INFO: 544 layer files passed checksum.
```

7. 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
```

8. 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
```

9. Execute the verification script:

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

10. When the USB integrity test completes, unmount the USB using the following command:

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

11. 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

12. To create *insight_setup_data.yaml*, see *Step 2 and Step 3 of Installation* under UM with Internet Access
13. Save the edited *insight_setup_data.yaml* file.
14. 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 !!
CISCO 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

The logs for this run are available at /var/log/insight/bootstrap/<date>_<time>

                ################################
                 CISCO 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  |
| CVIM/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 sss daemon]             [ DONE! ]    1min 23secs

.
.

Cisco 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 Cisco 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 to
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.

Refer Cisco VIM Insight Post Bootstrap Validation Checks section, to verify the bootstrap status of
Cisco VIM Insight.
```

UM Optional Services

UM Optional Services

For release Cisco VIM 3.2.0 or later, Cisco VIM 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:

1. 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
    ```

2. 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 Cisco VIM Insight bootstrap validation:

1. Check Cisco 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
```

2. After the Cisco VIM Insight bootstrap, you can view the status of Insight installation through *install-status* action using bootstrap:

```
# ./insight/insight_runner.py --install-status
Fetching Insight install status..

Cisco 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' |
+--------------------+--------+-----------------------------------------------+
```

3. You can also verify whether the Insight and MySQL containers are up or not, by running the following command:

```
$ source ~/.bashrc
$ dp

   NAMES             STATUS
insight_ng_26349    Up 8 days
mariadb_26349       Up 8 days
```

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.
```

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>Cisco 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">
<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>
<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>
<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>
<script
src="../static/reconfigureoptionalservices/reconfigureoptionalservices.js"></script>
</body>
```

6. 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

1. Enter the following address on the browser: https://<br_api>:9000.
2. 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.
3. If LDAP mode is True and LDAP user attribute is set to uid, login with LDAP user id credentials.
4. 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:

1. Log in as Insight UM.
2. Navigate to **Manage Pod Admin** and click **Add Pod Admin**.
3. Enter a new Email ID in **Add Pod Admin** pop-up.
4. Enter the username of the Pod Admin.
5. Click **Save**. User Registration mail is sent to a newly added Pod Admin with a token.
6. Click the URL with token and if token is valid then Pod Admin is redirected to Insight-Update Password page.
7. Enter new password and then confirm the same password.
8. Click **Submit**.

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

Cisco 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:

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 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 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>")'


## 2. vim_admin_public_key: ADMIN USER PUBLIC KEY                            |
##---------------------------------------------------------------------------
## 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.
  -  vim_admin_username: non_root_admin_1
     vim_admin_password_hash: $6.....

     # vim admin with password hash and public key
  -  vim_admin_username: 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
  -  vim_admin_username: non_root_admin_3
     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>
```


Cisco VIM administrators can manage their own passwords using the Linux *passwd* command. You can add or remove Cisco VIM administrator through the reconfigure option, while the passwords for their existing accounts remain unchanged.

Enabling UM Administrator with LDAP Authentication

Cisco 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 Cisco 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.

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 Cisco VIM 3.4.6 onwards.

To enable UM administrators with LDAP authentication, perform the following steps:

1. Take a backup of the *setupdata* file and update the file manually with the configuration listed below during installation:

```
UM_LDAP_ADMINS:
- domain_name: corp_ldap1
  ldap_uri: "ldaps://<ip_address_1:[port_1]>,ldaps://<ip_address_2:[port_2]>"
  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
  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.

2. 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)
```

> ⚠ 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 the reconfiguration, you can add a new *domain_name,* but the *domain_name* cannot be changed once configured.

3. To reconfigure the VIM administrator, run the following commands:

```
[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>
```

Enabling Root Login

To complement the management of VIM administrators, Cisco 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
```

3. Run the following command for installation:

```
# For Install
# ./insight/insight_runner.py -f <path_to insight_setup_data.yaml>
```

4. Run the following reconfigration 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

# False: admin cannot use root userid for ssh;  must use vim_admin_username # At least one vim_admin must be
configured, if it is False
  permit_root_login: True
```

Enabling Banner During SSH Login

Cisco 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/
```

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.

1. 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.

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:

   ```
   docker images
   ```

4. Once the uninstallation is complete, proceed with the installation steps mentioned in the Unified Management.

Verifying Installation

Verifying Installation

- Displaying IP Addresses
- Cisco VIM Client CLI Availability
- Displaying Cisco NFVI Logs
- Accessing OpenStack API Endpoints
- Assessing Cisco 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 Cisco 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  |
|               |               |            |            |           |         |
| 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]#
```

Cisco VIM Client CLI Availability

Cisco VIM Client CLI Availability

Cisco VIM client CLI is used for managing Cisco NFVI pods. After the completion of Cisco NFVI installation, verify that the Cisco VIM user is running and pointing to the right management node in the installer directory. Cisco NFVI provides a tool to check the REST API server status and directory where it is running.

1. 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
```

2. 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.
```

3. If the REST API server is not running, executing **ciscovim** displays the following error message:

```
# cd installer-<tagid>/
# ciscovim -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>/
# ciscovim --setupfile <setup_data_file_path> run --perform 7 -y
```


For more information on REST API, see REST API Overview

Displaying Cisco NFVI Logs

Displaying Cisco NFVI Logs

The Cisco 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 Cisco 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

Accessing OpenStack API Endpoints

Accessing OpenStack API Endpoints

The Cisco 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/v2.0 or
https://172.26.233.139:5000/v2.0 (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 Cisco NFVI Health

Assessing Cisco NFVI Health with CloudPulse

You can use the OpenStack CloudPulse tool to verify Cisco NFVI health. CloudPulse servers are installed on all Cisco NFVI control nodes, while the CloudPulse users are connected to the management node. Run the following commands to display Cisco NFVI information. For information about CloudPulse, visit the OpenStack CloudPulse
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 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
```

To run a specific CloudPulse test on demand:

```
[root@vms-line2-build installer-3128.2]# 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-3128.2]#
```

To show detailed results from a specific CloudPulse run:

```
[root@vms-line2-build installer-3128.2]# 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.

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 Kibana 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 Cisco NFVI Pod and Cloud Infrastructure

To test the Cisco NFVI pod and cloud infrastructure (host connectivity, basic mraiadb, 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 Cisco VIM, see Assessing Cisco NFVI Status.

Cisco VIM REST API

Cisco VIM REST API

The following topics explain how to use the Cisco VIM REST API to manage Cisco NFVI.

- Management Node VMs Lifecycle
- Cisco VIM REST API Using curl for IPv6
- Cisco VIM REST API Using curl for IPv4
- API Resources
- REST API Overview

Management Node VMs Lifecycle

Life-cycle Management of Management Node VM via Rest API

- Prerequisites
- Request to launch All Management Node VMs (defined in setup_data)
- Delete All Management Node VMs
- Delete Selected Management Node VMs
- Add New Management Node VMs to Existing Deployment
- Monitoring On-going Deployment
- Request to List All Management Node VMs
- Request to Fetch Status of Last Deployment Request
- Delete Central Management Deployment

To help meet the vision of complete automation, a set of Rest-APIs are developed to manage the deployment of the management node VMs.

Prerequisites

1. Copy the certificates from the management node to local machine from where you launch the APIs.
2. Create a folder in local machine and copy the certificates:

```
# mkdir ~/certificates
```

3. Copy the REST API CA certificates (for mercury commands)

```
# scp root@<Management Node>:/var/www/mercury/mercury-ca.crt ~/certificates
```

   The key information needed are br_api and cloud_api (external_lb_vip_ipv6_address).
4. For the pod hosting the management node VMs, get the REST API credentials:

```
# cat /opt/cisco/ui_config.json
{
    "Kibana-Url": "http://[2001:420:293:2440:b696:91ff:fe22:2dd8]:5601",
    "RestAPI-Username": "admin",
    "RestAPI-Password": "cfb605586d50115333c8",
    "RestDB-Password": "744ebc5feee30b733ac8",
    "RestAPI-Url": "https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445",
    "BuildNodeIP": "2001:420:293:2440:b696:91ff:fe22:2dd8" -> br_api
}
```

Request to launch All Management Node VMs (defined in setup_data)

URL

```
POST /v1/central_vm
```

Request

```
curl -g -i -X POST -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H 'Accept: application
/json' -H 'User-Agent: python-ciscovimclient' --cacert ~/certificates/mercury-ca.crt -d '{"name": "deploy_vms",
"jsondata": {"FLAVORS": {"vcpus": 8, "ram": 24576, "name": "cvim-mgmt.8x24-nonlocal"}, "KEYPAIRS": "****",
"CENTRAL_MGMT_USER_INFO": {"username": "cental_mgmt", "password": "****"}, "PODTYPE": "MGMT_CENTRAL", "IMAGES":
{"file_location": "/root/cm_images/buildnode-internal-24832.qcow2", "name": "cvim-mgmt-24832"}, "TIMEZONE": "US
/Pacific", "SERVERS_IN_VMS": {"name": "shardev1", "cvimadmin_password_hash": "****", "nics": {"fixed_ips":
{"subnet": "v6_api_subnet", "ipaddress": "2001:420:293:243c:10:8:98:4"}, "network_name": "prov-net-api-1230",
"name": "sc-br_api"}, "image": "cvim-mgmt-24832", "node_type": "management", "keypair": "****", "flavor": "cvim-
mgmt.8x24-nonlocal", "disk_vol_size": 512}, "NETWORKS": {"subnets": {"name": "v6_api_subnet",
"dns_nameservers": "2001:420:68d:4001::a", "range": "2001:420:293:243c::500", "ip_version": 6, "network_cidr":
"2001:420:293:243c::/64", "gateway": "2001:420:293:243c::2"}, "segment": "api", "name": "prov-net-api-1230",
"vlan_id": 1230}}, "launch_all": True}' https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/central_vm
```

You can change the name of the deployment. By default, it is *deploy_vms.*

Response

```
 {"status": "ToRun", "delete_vms": "", "add_vms": "", "name": "deploy_vms", "created_at": "2020-03-04T05:59:
00.492525+00:00", u"pdated_at": "2020-03-04T05:59:00.492528+00:00", "jsondata": {"FLAVORS": {"vcpus": 8, "ram":
24576, "name": "cvim-mgmt.8x24-nonlocal"}, "KEYPAIRS": "****", "CENTRAL_MGMT_USER_INFO": {u"sername":
"cental_mgmt", "password": "****"}, "PODTYPE": "MGMT_CENTRAL", "IMAGES": {"file_location": "/root/cm_images
/buildnode-internal-24832.qcow2", "name": "cvim-mgmt-24832"}, "TIMEZONE": u"S/Pacific", "SERVERS_IN_VMS":
{"name": "shardev1", "cvimadmin_password_hash": "****", "nics": {"fixed_ips": {"subnet": "v6_api_subnet",
"ipaddress": "2001:420:293:243c:10:8:98:4"}, "network_name": "prov-net-api-1230", "name": "sc-br_api"},
"image": "cvim-mgmt-24832", "node_type": "management", "keypair": "****", "flavor": "cvim-mgmt.8x24-nonlocal",
"disk_vol_size": 512}, "NETWORKS": {"subnets": {"name": "v6_api_subnet", "dns_nameservers": "2001:420:68d:4001::
a", "range": "2001:420:293:243c::500", "ip_version": 6, "network_cidr": "2001:420:293:243c::/64", "gateway":
"2001:420:293:243c::2"}, "segment": "api", "name": "prov-net-api-1230", "vlan_id": 1230}}, "vms_info": "",
"current_op_logs": "", "list_vms": False, "clean_all": False, "current_op_name": "Launching All VMs",
"launch_all": True}
```

To monitor the status of the current request, see *Monitoring Ongoing Deployment* section below.

Delete All Management Node VMs

URL

```
PUT /v1/central_vm/<Deployment Name>
```

Request

```
curl -g -i -X PUT -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H 'Accept: application
/json' -H 'User-Agent: python-ciscovimclient' --cacert ~/certificates/mercury-ca.crt -d '{"clean_all": True,
"name": "deploy_vms", "jsondata": {"FLAVORS": {"vcpus": 8, "ram": 24576, "name": "cvim-mgmt.8x24-nonlocal"},
"KEYPAIRS": "****", "CENTRAL_MGMT_USER_INFO": {"username": "cental_mgmt", "password": "****"}, "PODTYPE":
"MGMT_CENTRAL", "IMAGES": {"file_location": "/root/cm_images/buildnode-internal-24832.qcow2", "name": "cvim-
mgmt-24832"}, "TIMEZONE": "US/Pacific", "NETWORKS": {"subnets": {"name": "v6_api_subnet", "dns_nameservers":
"2001:420:68d:4001::a", "range": "2001:420:293:243c::500", "ip_version": 6, "network_cidr": "2001:420:293:243c::
/64", "gateway": "2001:420:293:243c::2"}, "segment": "api", "name": "prov-net-api-1230", "vlan_id": 1230}}}'
https://[2001:420:293:2433:172:29:85:106]:8445/central_vm/deploy_vms
```

Response

```
{"status": "ToRun", "delete_vms": "", "add_vms": "", "name": "deploy_vms", "created_at": "2020-03-04T05:59:
00+00:00", "updated_at": "2020-03-04T05:59:27+00:00", "jsondata": {"FLAVORS": {"vcpus": 8, "ram": 24576,
"name": "cvim-mgmt.8x24-nonlocal"}, "KEYPAIRS": "****", "CENTRAL_MGMT_USER_INFO": {"username": "cental_mgmt",
"password": "****"}, "PODTYPE": "MGMT_CENTRAL", "IMAGES": {"file_location": "/root/cm_images/buildnode-internal-
24832.qcow2", "name": "cvim-mgmt-24832"}, "TIMEZONE": "US/Pacific", "NETWORKS": {"subnets": {"name":
"v6_api_subnet", "dns_nameservers": "2001:420:68d:4001::a", "range": "2001:420:293:243c::500", "ip_version": 6,
"network_cidr": "2001:420:293:243c::/64", "gateway": "2001:420:293:243c::2"}, "segment": "api", "name": "prov-
net-api-1230", "vlan_id": 1230}}, "vms_info": "", "current_op_logs": "", "list_vms": False, "clean_all": True,
"current_op_name": "Clean Up All Resources", "launch_all": False}
```

To monitor the status of the current request, see *Monitoring Ongoing Deployment* section below.

Delete Selected Management Node VMs

URL

```
PUT /v1/central_vm/<Deployment Name>
```

Request

```
curl -g -i -X PUT -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H 'Accept: application
/json' -H 'User-Agent: python-ciscovimclient' --cacert ~/certificates/mercury-ca.crt -d '{"delete_vms":
vm_dev1, vm_dev2", "name": "deploy_vms", "jsondata": {"FLAVORS": {"vcpus": 8, "ram": 24576, "name": "cvim-mgmt.
8x24-nonlocal"}, "KEYPAIRS": "****", "CENTRAL_MGMT_USER_INFO": {"username": "cental_mgmt", "password": "****"},
"PODTYPE": "MGMT_CENTRAL", "IMAGES": {"file_location": "/root/cm_images/buildnode-internal-24832.qcow2",
"name": "cvim-mgmt-24832"}, "TIMEZONE": "US/Pacific", "SERVERS_IN_VMS": {"name": "shardev1",
"cvimadmin_password_hash": "****", "nics": {"fixed_ips": {"subnet": "v6_api_subnet", "ipaddress": "2001:420:293:
243c:10:8:98:4"}, "network_name": "prov-net-api-1230", "name": "sc-br_api"}, "image": "cvim-mgmt-24832",
"node_type": "management", "keypair": "****", "flavor": "cvim-mgmt.8x24-nonlocal", "disk_vol_size": 512},
"NETWORKS": {"subnets": {"name": "v6_api_subnet", "dns_nameservers": "2001:420:68d:4001::a", "range": "2001:420:
293:243c::500", "ip_version": 6, "network_cidr": "2001:420:293:243c::/64", "gateway": "2001:420:293:243c::2"},
"segment": "api", "name": "prov-net-api-1230", "vlan_id": 1230}}}' https://[2001:420:293:2433:172:29:85:106]:
8445/central_vm/deploy_vms
```

Response

```
{"status": "ToRun", "delete_vms": "chandradev1", "add_vms": "", "name": "deploy_vms", "created_at": "2020-03-
04T05:59:00+00:00", "updated_at": "2020-03-04T09:37:45+00:00", "jsondata": {"FLAVORS": {"vcpus": 8, "ram":
24576, "name": "cvim-mgmt.8x24-nonlocal"}, "KEYPAIRS": "****", "CENTRAL_MGMT_USER_INFO": {"username":
"cental_mgmt", "password": "****"}, "PODTYPE": "MGMT_CENTRAL", "IMAGES": {"file_location": "/root/cm_images
/buildnode-internal-24832.qcow2", "name": "cvim-mgmt-24832"}, "TIMEZONE": "US/Pacific", "SERVERS_IN_VMS":
{"name": "shardev1", "cvimadmin_password_hash": "****", "nics": {"fixed_ips": {"subnet": "v6_api_subnet",
"ipaddress": "2001:420:293:243c:10:8:98:4"}, "network_name": "prov-net-api-1230", "name": "sc-br_api"},
"image": "cvim-mgmt-24832", "node_type": "management", "keypair": "****", "flavor": "cvim-mgmt.8x24-nonlocal",
"disk_vol_size": 512}, "NETWORKS": {"subnets": {"name": "v6_api_subnet", "dns_nameservers": "2001:420:68d:4001::
a", "range": "2001:420:293:243c::500", "ip_version": 6, "network_cidr": "2001:420:293:243c::/64", "gateway":
"2001:420:293:243c::2"}, "segment": "api", "name": "prov-net-api-1230", "vlan_id": 1230}}, "vms_info": "",
"current_op_logs": "", "list_vms": False, "clean_all": False, "current_op_name": "Deleting VMs: vm_dev1,
vm_dev2", "launch_all": False}
```

⚠ To monitor the status of the current request, see *Monitoring Ongoing Deployment* section below.

Add New Management Node VMs to Existing Deployment

URL

```
PUT /v1/central_vm/<Deployment Name
```

Request

```
curl -g -i -X PUT -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H 'Accept: application
/json' -H 'User-Agent: python-ciscovimclient' --cacert ~/certificates/mercury-ca.crt -d '{"add_vms": "vm_dev1",
"name": "deploy_vms", "jsondata": {"FLAVORS": [{"vcpus": 8, "ram": 24576, "name": "cvim-mgmt.8x24-nonlocal"}],
"KEYPAIRS": [{"public_key_file": "/root/.ssh/id_rsa.pub", "name": "mgmt_vm_keypair"}],
"CENTRAL_MGMT_USER_INFO": {"username": "cental_mgmt", "password": "Lab1234!"}, "PODTYPE": "MGMT_CENTRAL",
"IMAGES": [{"file_location": "/root/cm_images/buildnode-internal-24832.qcow2", "name": "cvim-mgmt-24832"}],
"TIMEZONE": "US/Pacific", "SERVERS_IN_VMS": [{"name": "cmdev1", "cvimadmin_password_hash": "$6$WCVKuxRf.
LQdBNHT$HfgmirX/DMx6KNlS7VblGKms6UUrq4r8DAdQ/xHqbwMtg9ReHNSKGiap1hR0EPIbhWlYGAbxfhV8OxYRt1sZu1", "nics":
[{"network_name": "prov-net-mgmt-1229", "fixed_ips": [{"subnet": "v4_mgmt_subnet", "ipaddress": "10.8.97.10"},
{"subnet": "v6_mgmt_subnet", "ipaddress": "2001:420:293:243b:10:8:97:10"}], "name": "cmdev1-br_mgmt"},
{"network_name": "prov-net-api-1230", "fixed_ips": [{"subnet": "v4_api_subnet", "ipaddress": "10.8.98.10"},
{"subnet": "v6_api_subnet", "ipaddress": "2001:420:293:243c:10:8:98:2"}], "name": "cmdev1-br_api"}], "image":
"cvim-mgmt-24832", "keypair": "mgmt_vm_keypair", "node_type": "management", "flavor": "cvim-mgmt.8x24-
nonlocal", "disk_vol_size": 512}, {"name": "vm_dev1", "cvimadmin_password_hash": "$6$WCVKuxRf.LQdBNHT$HfgmirX
/DMx6KNlS7VblGKms6UUrq4r8DAdQ/xHqbwMtg9ReHNSKGiap1hR0EPIbhWlYGAbxfhV8OxYRt1sZu1", "nics": [{"network_name":
"prov-net-mgmt-1229", "fixed_ips": [{"subnet": "v4_mgmt_subnet", "ipaddress": "10.8.97.11"}, {"subnet":
"v6_mgmt_subnet", "ipaddress": "2001:420:293:243b:10:8:97:11"}], "name": "cg-br_mgmt"}, {"network_name": "prov-
net-api-1230", "fixed_ips": [{"subnet": "v4_api_subnet", "ipaddress": "10.8.98.11"}, {"subnet":
"v6_api_subnet", "ipaddress": "2001:420:293:243c:10:8:98:3"}], "name": "cg-br_api"}], "image": "cvim-mgmt-
24832", "keypair": "mgmt_vm_keypair", "node_type": "management", "flavor": "cvim-mgmt.8x24-nonlocal",
"disk_vol_size": 512}, {"name": "shardev1", "cvimadmin_password_hash": "$6$WCVKuxRf.LQdBNHT$HfgmirX
/DMx6KNlS7VblGKms6UUrq4r8DAdQ/xHqbwMtg9ReHNSKGiap1hR0EPIbhWlYGAbxfhV8OxYRt1sZu1", "nics": [{"network_name":
"prov-net-mgmt-1229", "fixed_ips": [{"subnet": "v4_mgmt_subnet", "ipaddress": "10.8.97.12"}, {"subnet":
"v6_mgmt_subnet", "ipaddress": "2001:420:293:243b:10:8:97:12"}], "name": "sc-br_mgmt"}, {"network_name": "prov-
net-api-1230", "fixed_ips": [{"subnet": "v4_api_subnet", "ipaddress": "10.8.98.12"}, {"subnet":
"v6_api_subnet", "ipaddress": "2001:420:293:243c:10:8:98:4"}], "name": "sc-br_api"}], "image": "cvim-mgmt-
24832", "keypair": "mgmt_vm_keypair", "node_type": "management", "flavor": "cvim-mgmt.8x24-nonlocal",
"disk_vol_size": 512}], "NETWORKS": [{"subnets": [{"range": ["10.8.97.5", "10.8.97.199"], "name":
"v4_mgmt_subnet", "ip_version": 4, "network_cidr": "10.8.97.0/24", "dns_nameservers": ["171.70.168.183"],
"gateway": "10.8.97.2"}, {"range": ["2001:420:293:243b::100", "2001:420:293:243b::500"], "name":
"v6_mgmt_subnet", "ip_version": 6, "network_cidr": "2001:420:293:243b::/64", "dns_nameservers": ["2001:420:68d:
4001::a"], "gateway": "2001:420:293:243b::2"}], "segment": "management", "name": "prov-net-mgmt-1229",
"vlan_id": 1229}, {"subnets": [{"range": ["10.8.98.5", "10.8.98.199"], "name": "v4_api_subnet", "ip_version":
4, "network_cidr": "10.8.98.0/24", "dns_nameservers": ["171.70.168.183"], "gateway": "10.8.98.2"}, {"range":
["2001:420:293:243c::100", "2001:420:293:243c::500"], "name": "v6_api_subnet", "ip_version": 6, "network_cidr":
"2001:420:293:243c::/64", "dns_nameservers": ["2001:420:68d:4001::a"], "gateway": "2001:420:293:243c::2"}],
"segment": "api", "name": "prov-net-api-1230", "vlan_id": 1230}]}
```

Response

```
{"status": "ToRun", "delete_vms": "", "add_vms": "vm_dev1", "name": "deploy_vms", "created_at": "2020-03-04T05:
59:00+00:00", "updated_at": "2020-03-04T09:51:40+00:00", "jsondata": {"FLAVORS": {"vcpus": 8, "ram": 24576,
"name": "cvim-mgmt.8x24-nonlocal"}, "KEYPAIRS": "****", "CENTRAL_MGMT_USER_INFO": {"username": "cental_mgmt",
"password": "****"}, "PODTYPE": "MGMT_CENTRAL", "IMAGES": {"file_location": "/root/cm_images/buildnode-internal-
24832.qcow2", "name": "cvim-mgmt-24832"}, "TIMEZONE": "US/Pacific", "SERVERS_IN_VMS": {"name": "shardev1",
"cvimadmin_password_hash": "****", "nics": {"fixed_ips": {"subnet": "v6_api_subnet", "ipaddress": "2001:420:293:
243c:10:8:98:4"}, "network_name": "prov-net-api-1230", "name": "sc-br_api"}, "image": "cvim-mgmt-24832",
"node_type": "management", "keypair": "****", "flavor": "cvim-mgmt.8x24-nonlocal", "disk_vol_size": 512},
"NETWORKS": {"subnets": {"name": "v6_api_subnet", "dns_nameservers": "2001:420:68d:4001::a", "range": "2001:420:
293:243c::500", "ip_version": 6, "network_cidr": "2001:420:293:243c::/64", "gateway": "2001:420:293:243c::2"},
"segment": "api", "name": "prov-net-api-1230", "vlan_id": 1230}}, "vms_info": "", "current_op_logs": "",
"list_vms": False, "clean_all": False, "current_op_name": "Adding VMs: chandradev1", "launch_all": False}
```


To monitor the status of the current request, see *Monitoring Ongoing Deployment* section below.

Monitoring On-going Deployment

Use the following method to monitor any ongoing operation related to a given deployment.

URL

```
GET /v1/central_vm/status?name=<Deployment Name>
```

Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u  admin:46d13357ef15e5482b52 -H 'Accept: application
/json' -H 'User-Agent: python-ciscovimclient' --cacert ~/certificates/mercury-ca.crt  https://[2001:420:293:
2440:b696:91ff:fe22:2dd8]:8445/ central_vm/status?name=deploy_vms
```

Response

```
{"status": "ToRun", "current_op_name": "Launching All VMs", "current_op_logs": "", "name": "deploy_vms"}
```

Response with Log Files

```
{"status": "Running", "current_op_name": "Launching All VMs", "current_op_logs": "https://[2001:420:293:2433:
172:29:85:106]:8008/mercury/central_vm_deploy2020-03-03_21-59-04.log", "name": "deploy_vms"}
```

Possible values for *Status*

- ToRun
- Running
- Success
- Failed

Request to List All Management Node VMs

URL

```
GET /v1/central_vm/vms_info?name=deploy_vms
```

Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u  admin:46d13357ef15e5482b52 -H 'Accept: application
/json' -H 'User-Agent: python-ciscovimclient' --cacert ~/certificates/mercury-ca.crt https://[2001:420:293:2433:
172:29:85:106]:8445/central_vm/vms_info?name=deploy_vms
```

Response

```
{"vms_info": [{"Status": "ACTIVE", "Name": "shardev1", "Networks": "prov-net-mgmt-1229=10.8.97.12, 2001:420:293:
243b:10:8:97:12; prov-net-api-1230=10.8.98.12, 2001:420:293:243c:10:8:98:4"}, {"Status": "ACTIVE", "Name":
"vmdev1", "Networks": "prov-net-mgmt-1229=10.8.97.11, 2001:420:293:243b:10:8:97:11; prov-net-api-1230=10.
8.98.11, 2001:420:293:243c:10:8:98:3"}, {"Status": "ACTIVE", "Name": "cmdev1", "Networks": "prov-net-mgmt-
1229=10.8.97.10, 2001:420:293:243b:10:8:97:10; prov-net-api-1230=10.8.98.10, 2001:420:293:243c:10:8:98:2"}]}
```

Request to Fetch Status of Last Deployment Request

URL

```
GET /v1/central_vm
```

Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u  admin:46d13357ef15e5482b52 -H 'Accept: application
/json' -H 'User-Agent: python-ciscovimclient' --cacert ~/certificates/mercury-ca.crt https://[2001:420:293:2433:
172:29:85:106]:8445/central_vm
```

Response


```
{
  "centralVMs": [
    {
      "add_vms": "chandradev1",
      "current_op_name": "Adding VMs: chandradev1",
      "launch_all": false,
      "clean_all": false,
      "list_vms": false,
      "status": "Failed",
      "delete_vms": "",
      "name": "deploy_vms",
      "created_at": "2020-03-04T05:59:00+00:00",
      "updated_at": "2020-03-05T06:32:40+00:00",
      "jsondata": "{\"FLAVORS\": [{\"vcpus\": 8, \"ram\": 24576, \"name\": \"cvim-mgmt.8x24-nonlocal\"}], \"
KEYPAIRS\": [{\"public_key_file\": \"/root/.ssh/id_rsa.pub\", \"name\": \"mgmt_vm_keypair\"}], \"
CENTRAL_MGMT_USER_INFO\": {\"username\": \"cental_mgmt\", \"password\": \"Lab1234!\"}, \"PODTYPE\": \"
MGMT_CENTRAL\", \"IMAGES\": [{\"file_location\": \"/root/cm_images/buildnode-internal-24832.qcow2\", \"name\":
\"cvim-mgmt-24832\"}], \"TIMEZONE\": \"US/Pacific\", \"SERVERS_IN_VMS\": [{\"name\": \"cmdev1\", \"
cvimadmin_password_hash\": \"$6$WCVKuxRf.LQdBNHT$HfgmirX/DMx6KNlS7VblGKms6UUrq4r8DAdQ
/xHqbwMtg9ReHNSKGiap1hR0EPIbhWlYGAbxfhV8OxYRt1sZu1\", \"nics\": [{\"network_name\": \"prov-net-mgmt-1229\", \"
fixed_ips\": [{\"subnet\": \"v4_mgmt_subnet\", \"ipaddress\": \"10.8.97.10\"}, {\"subnet\": \"v6_mgmt_subnet\",
\"ipaddress\": \"2001:420:293:243b:10:8:97:10\"}], \"name\": \"cmdev1-br_mgmt\"}, {\"network_name\": \"prov-net-
api-1230\", \"fixed_ips\": [{\"subnet\": \"v4_api_subnet\", \"ipaddress\": \"10.8.98.10\"}, {\"subnet\": \"
v6_api_subnet\", \"ipaddress\": \"2001:420:293:243c:10:8:98:2\"}], \"name\": \"cmdev1-br_api\"}], \"image\": \"
cvim-mgmt-24832\", \"keypair\": \"mgmt_vm_keypair\", \"node_type\": \"management\", \"flavor\": \"cvim-mgmt.
8x24-nonlocal\", \"disk_vol_size\": 512}, {\"name\": \"chandradev1\", \"cvimadmin_password_hash\":
\"$6$WCVKuxRf.LQdBNHT$HfgmirX/DMx6KNlS7VblGKms6UUrq4r8DAdQ/xHqbwMtg9ReHNSKGiap1hR0EPIbhWlYGAbxfhV8OxYRt1sZu1\",
\"nics\": [{\"network_name\": \"prov-net-mgmt-1229\", \"fixed_ips\": [{\"subnet\": \"v4_mgmt_subnet\", \"
ipaddress\": \"10.8.97.11\"}, {\"subnet\": \"v6_mgmt_subnet\", \"ipaddress\": \"2001:420:293:243b:10:8:97:
11\"}], \"name\": \"cg-br_mgmt\"}, {\"network_name\": \"prov-net-api-1230\", \"fixed_ips\": [{\"subnet\": \"
v4_api_subnet\", \"ipaddress\": \"10.8.98.11\"}, {\"subnet\": \"v6_api_subnet\", \"ipaddress\": \"2001:420:293:
243c:10:8:98:3\"}], \"name\": \"cg-br_api\"}], \"image\": \"cvim-mgmt-24832\", \"keypair\": \"
mgmt_vm_keypair\", \"node_type\": \"management\", \"flavor\": \"cvim-mgmt.8x24-nonlocal\", \"disk_vol_size\":
512}, {\"name\": \"shardev1\", \"cvimadmin_password_hash\": \"$6$WCVKuxRf.LQdBNHT$HfgmirX
/DMx6KNlS7VblGKms6UUrq4r8DAdQ/xHqbwMtg9ReHNSKGiap1hR0EPIbhWlYGAbxfhV8OxYRt1sZu1\", \"nics\": [{\"
network_name\": \"prov-net-mgmt-1229\", \"fixed_ips\": [{\"subnet\": \"v4_mgmt_subnet\", \"ipaddress\": \"
10.8.97.12\"}, {\"subnet\": \"v6_mgmt_subnet\", \"ipaddress\": \"2001:420:293:243b:10:8:97:12\"}], \"name\": \"
sc-br_mgmt\"}, {\"network_name\": \"prov-net-api-1230\", \"fixed_ips\": [{\"subnet\": \"v4_api_subnet\", \"
ipaddress\": \"10.8.98.12\"}, {\"subnet\": \"v6_api_subnet\", \"ipaddress\": \"2001:420:293:243c:10:8:98:4\"}],
\"name\": \"sc-br_api\"}], \"image\": \"cvim-mgmt-24832\", \"keypair\": \"mgmt_vm_keypair\", \"node_type\": \"
management\", \"flavor\": \"cvim-mgmt.8x24-nonlocal\", \"disk_vol_size\": 512}], \"NETWORKS\": [{\"subnets\":
[{\"range\": [\"10.8.97.5\", \"10.8.97.199\"], \"name\": \"v4_mgmt_subnet\", \"ip_version\": 4, \"
network_cidr\": \"10.8.97.0/24\", \"dns_nameservers\": [\"171.70.168.183\"], \"gateway\": \"10.8.97.2\"}, {\"
range\": [\"2001:420:293:243b::100\", \"2001:420:293:243b::500\"], \"name\": \"v6_mgmt_subnet\", \"
ip_version\": 6, \"network_cidr\": \"2001:420:293:243b::/64\", \"dns_nameservers\": [\"2001:420:68d:4001::a\"],
\"gateway\": \"2001:420:293:243b::2\"}], \"segment\": \"management\", \"name\": \"prov-net-mgmt-1229\", \"
vlan_id\": 1229}, {\"subnets\": [{\"range\": [\"10.8.98.5\", \"10.8.98.199\"], \"name\": \"v4_api_subnet\", \"
ip_version\": 4, \"network_cidr\": \"10.8.98.0/24\", \"dns_nameservers\": [\"171.70.168.183\"], \"gateway\": \"
10.8.98.2\"}, {\"range\": [\"2001:420:293:243c::100\", \"2001:420:293:243c::500\"], \"name\": \"
v6_api_subnet\", \"ip_version\": 6, \"network_cidr\": \"2001:420:293:243c::/64\", \"dns_nameservers\": [\"2001:
420:68d:4001::a\"], \"gateway\": \"2001:420:293:243c::2\"}], \"segment\": \"api\", \"name\": \"prov-net-api-
1230\", \"vlan_id\": 1230}]}",
      "vms_info": "",
      "current_op_logs": "https://[2001:420:293:2433:172:29:85:106]:8008/mercury/central_vm_deploy2020-03-04_22-
32-25.log"
    }
  ]
}
```


Delete Central Management Deployment

URL

```
DELETE /v1/central_vm/<Deployment Name>
```

Request

```
# curl -i -X DELETE -s -u admin:10e897312b8c606d3899 --cacert /var/www/mercury/mercury-ca.crt https://172.
29.85.78:8445/v1/central_vm/gg34_vms
```

Response

```
HTTP/1.1 204 No Content
Date: Fri, 16 Oct 2020 10:08:15 GMT
Server: WSGIServer/0.2 CPython/3.6.8
Cache-Control: no-cache, no-store, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1
CVIM-API-Minimum-Version: 2.4.0
CVIM-API-Maximum-Version: 4.0.0
CVIM-API-Version: 2.4.0
vary: CVIM-API-Version
```

Cisco VIM REST API Using curl for IPv6

Cisco VIM REST API Using curl for IPv6

- Prerequisites
- Offline Validation using curl
- Start New Installation
- Pod Management Operations
  - Prerequisites
  - Update setup data
  - Add Compute
  - Add Storage
  - Remove Compute
  - Remove Storage
  - Replace controller
  - Fetch Hardware Inventory
  - Glance Image Upload

Prerequisites

1. You need to copy the certificates from the management node to local machine from where you would launch the APIs.
2. Create a folder in local machine and copy the certificates:

```
# mkdir ~/certificates
```

3. Copy REST API CA Certificates (for mercury commands)

```
# scp root@<Management Node>:/var/www/mercury/mercury-ca.crt ~/certificates
```

> ⚠ The key information that you need are br_api and cloud_api (external_lb_vip_ipv6_address).

4. For each POD, get the REST API credentials:

```
# cat /opt/cisco/ui_config.json
{
        "Kibana-Url": "http://[2001:420:293:2440:b696:91ff:fe22:2dd8]:5601",
        "RestAPI-Username": "admin",
        "RestAPI-Password": "cfb605586d50115333c8",
        "RestDB-Password": "744ebc5feee30b733ac8",
        "RestAPI-Url": "https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445",
        "BuildNodeIP": "2001:420:293:2440:b696:91ff:fe22:2dd8" -> br_api
}
```

Offline Validation using curl

1. Create offline validation test

Request

```
curl -g -i -X POST -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{"jsondata" : {<SetupData in JSON Format>}}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/v1/offlinevalidation
UUID is returned from request
```

Response

```
{"status": "NotValidated", "uuid": "2b8253f4-ad9f-4fbf-b224-a65bd210392a", "created_at":
"2019-02-28T18:02:36.808740+00:00", "updated_at": null, "jsondata": "{}"}
```

2. Get the offline validation test result:

Request

```
Curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/v1/offlinevalidation/2b8253f4-ad9f-4fbf-b224-
a65bd210392a
```

Response

```
{"status": "ValidationFailed", "uuid": "2b8253f4-ad9f-4fbf-b224-a65bd210392a",
"created_at": "2019-02-28T18:02:36+00:00", "updated_at": "2019-02-28T18:02:57+00:00",
"jsondata": ""}
```

Start New Installation

1. Create new setup date before starting new installation, for example:

```
curl -g -i -X POST -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'meta': {}, u'name': u'NEWSETUPDATA', u'jsondata': {<SetupData in
JSON Format>}}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/setupdata
```

2. To start the installation:
Request

```
Curl -g -i -X POST -H 'Content-Type: application/json' admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'stages': u'vmtp', u'setupdata': u'8b0d4a46-c67f-4121-99af-
32fde52a82eb'}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/install
```

Response

```
{u'uuid': u'6b02c2ab-441e-471a-9dcc-e771136186e1', u'setupdata':
u'8b0d4a46-c67f-4121-99af-32fde52a82eb', u'vmtpresult': u'', u'updated_at': None,
u'validationstatus': u'', u'currentstatus': u'Not Available', u'install_logs': u'',
u'stages': {u'baremetal': u'Scheduled', u'bootstrap': u'Scheduled', u'runtimevalidation':
u'Scheduled', u'ceph': u'Scheduled', u'orchestration': u'Scheduled', u'validation':
u'Scheduled', u'hostsetup': u'Scheduled', u'vmtp': u'Scheduled'}, u'created_at':
u'2019-03-05T05:22:30.986823+00:00'}
```

3. Get active setupdata with UUID after installation is started:

Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/setupdata
```

Response

It will return in the list format. You must check the status. The status can be Active, Installation Failed,or Installing.

```
{"setupdatas": [{"status": "Active", "uuid": "c5bc5fd9-6f4b-43e7-a61a-a9d409569943",
"jsondata": " {<Setupdata JSON>}", "meta": "{}", "name": "NEWSETUPDATA"}]}
```

4. Monitoring the installation using OP-information (current operation information):

   Request

   ```
   curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
   'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
   ~/certificates/mercury-ca.crt
   https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/op_info
   ```

   Response

   Check for the value of key insight_monitor_status. If it is **Running**, it indicates that the last operation is still in running state. Once the operation is completed, the value is either **Success/Failed** based on the result.

   ```
   {u'created_at': u'2019-02-25 18:15:00+00:00', u'updated_at': u'2019-02-25 18:15:00+00:00',
   u'reboot_required': False, u'update_status': False, u'current_op_logs':
   u'https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8008/mercury/ae3ed699-2ffe-4ae0-a8ab-83ef7fdce008',
   u'current_op_status': u'Running', u'insight_monitor_status': u'Running',
   u'current_op_name': u'install_op Orchestration', u'current_op_monitor':
   u'Install_Op_orchestration'}
   ```

   Sample output information after successful completion is given below:

   ```
   {"created_at": "2019-03-04 21:35:00+00:00", "updated_at": "2019-03-04 21:36:24+00:00",
   "reboot_required": false, "update_status": false, "current_op_logs": "",
   "current_op_status": "diskmgmt_completed", "insight_monitor_status": "Success",
   "current_op_name": "DiskMgmt", "current_op_monitor": ""}
   ```

Pod Management Operations

Prerequisites

Before performing any pod management operation, you need to update the setup data using PUT method.

Update setup data

1. Get the active setup data UUID using the install API

   Request

   ```
   curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
   'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
   ~/certificates/mercury-ca.crt
   https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/install
   ```

   Response

   ```
   {u'installs': {u'uuid': u'6b02c2ab-441e-471a-9dcc-e771136186e1', u'setupdata':
   u'8b0d4a46-c67f-4121-99af-32fde52a82eb', . . .}}
   ```

2. Send PUT request on setup data UUID

   ```
   curl -g -i -X PUT -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
   'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
   ~/certificates/mercury-ca.crt -d '{u'meta': {}, u'name': u'NEWSETUPDATA', u'jsondata':
   {<Setupdata JSON>}}'
   https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/setupdata/8b0d4a46-c67f-4121-99af-32fde52a82eb
   ```

   You can perform the following pod management operations:

- Add compute
- Add storage
- Remove compute
- Remove storage
- Replace controller

Add Compute

1. Add the node entry in setup data and update the setup data by following the steps given under prerequisites.
2. POST to nodes to add entry:

```
curl -g -i -X POST -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52
-H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'name': u'Compute-4'}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes/add_compute
```

Add Storage

1. Add the node entry in setup data and update the setup data by following the steps given under prerequisites.
2. POST to nodes to add entry:

```
curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'name': u'Store-4'}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes/add_storage
```

Remove Compute

1. List the nodes:

   Request

   ```
   curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
   'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
   ~/certificates/mercury-ca.crt https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes
   ```

   Response

   ```
   {"nodes": [{"status": "Active", "uuid": "1929776f-8b77-4b35-b55c-0abd6433b989",
   "setupdata": "8b0d4a46-c67f-4121-99af-32fde52a82eb", "node_data": "{\"rack_info\":
   {\"rack_id\": \"RackC\"}, \"cimc_info\": {\"cimc_ip\": \"172.29.172.81\"},
   \"management_ip\": \"21.0.0.13\"}", "updated_at": "2019-03-04T21:42:38+00:00",
   "reboot_required": "No", "mtype": " block_storage", "install":
   "6b02c2ab-441e-471a-9dcc-e771136186e1", "power_status": "PowerOnSuccess", "install_logs":
   "https://172.31.231.17:8008/mercury/071e79a5-b279-4628-bcf0-df168152cc42", "created_at":
   "2019-03-05T05:42:38+00:00", "name": "compute-3"}, . . . ]}
   ```

2. Remove the node entry in setup data and update the setup data by following the steps given under prerequisites.
3. Send delete request on nodes, to remove the storage node for that UUID:

```
curl -g -i -X DELETE -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52
-H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'force_op': False, u'name':
u'1929776f-8b77-4b35-b55c-0abd6433b989'}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes/remove_compute
```

Remove Storage

1. Get the UUID of the node to be removed by getting the list of nodes
   Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes
```

Response

```
{"nodes": [{"status": "Active", "uuid": "0b7b2b6e-305c-48e0-b9f3-0ddb72bd3b3f",
"setupdata": "8b0d4a46-c67f-4121-99af-32fde52a82eb", "node_data": "{\"rack_info\":
{\"rack_id\": \"RackC\"}, \"cimc_info\": {\"cimc_ip\": \"172.29.172.81\"},
\"management_ip\": \"21.0.0.13\"}", "updated_at": "2019-03-04T21:42:38+00:00",
"reboot_required": "No", "mtype": " block_storage", "install":
"6b02c2ab-441e-471a-9dcc-e771136186e1", "power_status": "PowerOnSuccess", "install_logs":
"https://172.31.231.17:8008/mercury/071e79a5-b279-4628-bcf0-df168152cc42", "created_at":
"2019-03-05T05:42:38+00:00", "name": "Store-3"}, . . . ]}
```

2. Remove the node entry in setup data and update the setup data using steps mentioned in the prerequisites.
3. Send delete request on nodes, to remove the storage node for that UUID.

```
curl -g -i -X DELETE -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52
-H 'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'force_op': False, u'name':
u'0b7b2b6e-305c-48e0-b9f3-0ddb72bd3b3f'}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes/remove_storage
```

Replace controller

1. Get the UUID of the node to be removed by getting the list of nodes:

Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes
```

Response

```
{"nodes": [{"status": "Active", "uuid": "79e43c4c-8cbd-4c81-8c22-3aec717298e9",
"setupdata": "8b0d4a46-c67f-4121-99af-32fde52a82eb", "node_data": "{\"rack_info\":
{\"rack_id\": \"RackC\"}, \"cimc_info\": {\"cimc_ip\": \"172.29.172.81\"},
\"management_ip\": \"21.0.0.13\"}", "updated_at": "2019-03-04T21:42:38+00:00",
"reboot_required": "No", "mtype": " control", "install":
"6b02c2ab-441e-471a-9dcc-e771136186e1", "power_status": "PowerOnSuccess", "install_logs":
"https://172.31.231.17:8008/mercury/071e79a5-b279-4628-bcf0-df168152cc42", "created_at":
"2019-03-05T05:42:38+00:00", "name": "gg34-10"}, . . . ]}
```

2. Remove the node entry in setup data and update the setup data using steps mentioned in the prerequisites.
3. Put nodes to replace entry:

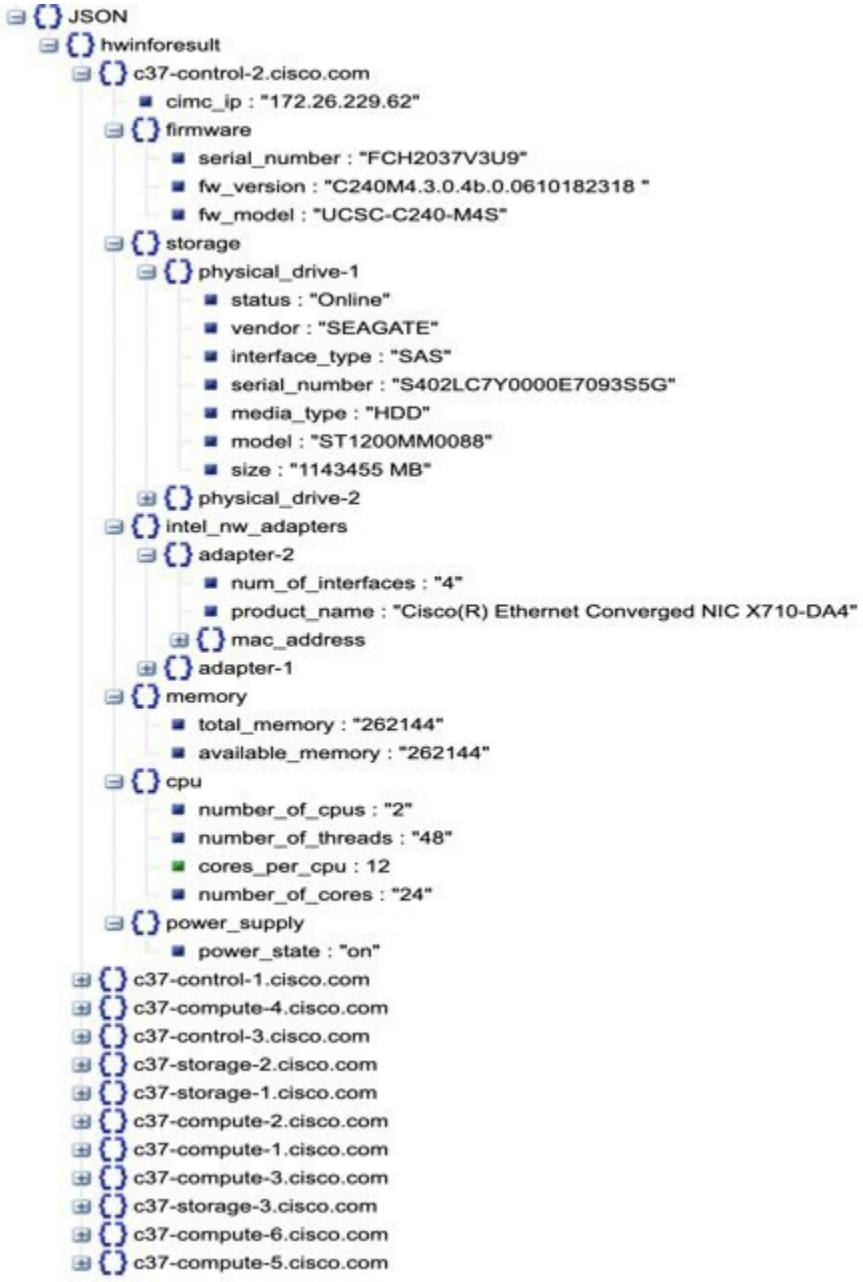
```
curl -g -i -X PUT -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt -d '{u'status': u'ToReplace', u'force_op': False, u'name':
u'gg34-10'}'
https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/nodes/79e43c4c-8cbd-4c81-8c22-3aec717298e9
```

Fetch Hardware Inventory

Request

```
curl -g -i -X GET -H 'Content-Type: application/json' -u admin:46d13357ef15e5482b52 -H
'Accept: application/json' -H 'User-Agent: python-ciscovimclient' --cacert
~/certificates/mercury-ca.crt https://[2001:420:293:2440:b696:91ff:fe22:2dd8]:8445/v1/hwinfo
```

Response



Glance Image Upload

Use RestAPIs to upload and delete multiple images to/from the cloud.

Following are the REST API that are available for usage.

POST /upload

This API is responsible for uploading the image to respective Openstack Cloud.

JSON Payload

```
{
"podsip":[
"172.31.231.17",
"10.30.116.244",
],
"images":[
"Vodafone.iso",
"Rakuten.qcow2",
]
}
Response
{"Upload":true}
```

CURL Request

Following is an example Curl request

```
curl -s -k -X POST -d '{"upload": {"podsip":["172.23.105.218",
"172.29.85.78"],"images":["buildnode-internal-20606.iso","CentOS-7-x86_64-GenericCloud-1503.qcow2"]}}'
-H "Auth: <Token>" https://172.29.85.78:9001/upload
```

Delete /upload

This API is responsible for deleting the image from respective Openstack Cloud.

JSON Payload

```
{
"podsip":[
"172.31.231.17",
"10.30.116.244",
],
"images":[
"Vodafone.iso",
"Rakuten.qcow2",
]
}
```

CURL Request
Following is the example Curl request:

```
curl -s -k -X DELETE -d '{"upload": {"podsip":["172.23.105.218",
"172.29.85.78"],"images":["buildnode-internal-20606.iso","CentOS-7-x86_64-GenericCloud-1503.qcow2"]}}'
-H "Auth: <Token>" https://172.29.85.78:9001/upload
```

Response

```
{"Delete":true}
```

GET /upload

This API is responsible to get the image list from respective Openstack Cloud.

Following are the query string parameters to be passed with GET URL

1. odsip: It is a comma separated string which represents pod IPs, whose Openstack image list needs to be fetched.
2. images: It is a comma separated string which represents Openstack images whose status needs to be fetched.
3. refresh: Takes the value true or false. Used to get updated Openstack images list.

Following are the CURL request examples:

Example 1:

```
curl -s -k -H "Auth: <Token>" https://172.29.85.78:9001/upload
This gives the result of pods on which upload/get/delete operation are performed.
{
        "uploaded": {
                "172.29.85.78": {
                "opsinprogress": 0,
                "images": null,
                "error": ""
                                },
                "172.23.105.218": {
                "opsinprogress": 0,
                "images": null,
                "error": ""
                }
        }
}
```

Example 2:

```
curl -s -k -H "Auth: <Token>" https://172.29.85.78:9001/upload?"podsip=172.29.85.78"
{
        "uploaded": {
                "172.29.85.78": {
                "opsinprogress": 0,
                "images": [
                {
                        "OSStatus": "active",
                        "UploadStatus": "UploadSuccess",
                        "ErrStatus": "",
                        "ID": "c50284d7-191a-42ed-a289-9b52d19b9fd5",
                        "Name": "buildnode-internal-20606.iso"
                },
        {
                        "OSStatus": "active",
                        "UploadStatus": "UploadSuccess",
                        "ErrStatus": "",
                        "ID": "fee44efc-684e-46ac-aa89-b6e785faf1b4",
                        "Name": "CentOS-7-x86_64-GenericCloud-1503.qcow2"
                }
        ],
                "error": ""
                }
                }
}
```

Example 3:

```
curl -s -k -H "Auth: <Token>"
https://172.29.85.78:9001/upload?"podsip=172.29.85.78&refresh=true"
{
        "uploaded": {
                "172.29.85.78": {
                "opsinprogress": 1,
                "images": null,
                "error": ""
},
}
```

Example 4:

```
curl -s -k -H "Auth: <Token>" https://172.29.85.78:9001/upload?"podsip=172.29.85.78&
images=buildnode-internal-20606.iso"
{
        "uploaded": {
        "172.29.85.78": {
        "opsinprogress": 0,
        "images": [
        {
                "OSStatus": "active",
                "UploadStatus": "UploadSuccess",
                "ErrStatus": "",
                "ID": "c50284d7-191a-42ed-a289-9b52d19b9fd5",
                "Name": "buildnode-internal-20606.iso"
        }
        ],
        "error": ""
        }
}
}
```

Cisco VIM REST API Using curl for IPv4

Cisco VIM REST API Using curl for IPv4

- Getting REST API Username & Password
- Nodes APIs and Commands
- List Openstack Configuration Command
- List Password Secrets
- Cluster Recovery
- Last-Run-Status
- Command
- Reconfigure Regenerate Secrets
- Reconfigure Set Password
- Reconfigure Set Openstack Configuration
- Reconfigure CIMC Password

Getting REST API Username & Password

Use the following configuration to get REST API Username and Password:

```
cat /opt/cisco/ui_config.json
 {
        "Kibana-Url": "http://172.31.231.17:5601",
        "RestAPI-Username": "admin",
        "RestAPI-Password": "****",
        "RestDB-Password": "****",
        "RestAPI-Url": "https://172.31.231.17:8445",
        "BuildNodeIP": "172.31.231.17"
 }
```

Nodes APIs and Commands

List Nodes

Use the following curl command to get the node's status, power status, reboot status, and mtype information:

```
curl -i -X GET -u admin:**** -H 'Content-Type: application/json' -H 'Accept: application/json'
 --cacert /var/www/mercury/mercury-ca.crt https://172.31.231.17:8445/v1/nodes
```



Response

```
{"nodes": [{"status": ". . . . "name": "Store-2"}]}
```

Power OFF Nodes

To get the power off status of the nodes, use the below command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H 'Accept:
application/json' --cacert /var/www/mercury/mercury-ca.crt -d '{'status': 'PowerOff',
'force_op': False, 'name': '<Node UUID>'}'
 https://172.31.231.17:8445/v1/nodes/node_power_status
```

⚠ You can find the UUID of the node from the list nodes command.

Power ON Nodes

To get the power ON status of the nodes, use the following command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H 'Accept:
 application/json' --cacert /var/www/mercury/mercury-ca.crt -d '{'status': 'PowerOn', 'force_op': False,
'name': '<Node UUID>'}'
 https://172.31.231.17:8445/v1/nodes/node_power_status
```

⚠ You can find the UUID of the node from the list nodes command.

Power Status of Nodes

To get the Live status of the nodes, first send POST request to /v1/hwinfoAPI, and then place the GET request on v1/hwinfo/get_nodes_power_status after a minute approximately.
Run the below commands to send the POST request and get the power status:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H 'Accept:
application/json' --cacert /var/www/mercury/mercury-ca.crt -d '{}'
https://172.31.231.17:8445/v1/hwinfo
curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H 'Accept: application/json'
 --cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/v1/hwinfo/get_nodes_power_status
```

Response

```
{'Store-3': {'intended_power_state': 'PowerOnSuccess', 'actual_power_state': 'on'},}}
```

Reboot Node

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt -d
'{'status': 'Reboot', 'force_op': False, 'name': '<Node UUID>'}' https://172.31.231.17:8445/v1/nodes
/node_power_status
```

⚠ You can find the UUID of the node from the list nodes command.

Reboot Status

Use the following two commands, to get the reboot status of the node:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt
 -d 'None' https://172.31.231.17:8445/v1/nodes/reboot_status
 curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H 'Accept: application/json'
 --cacert /var/www/mercury/mercury-ca.crt https://172.31.231.17:8445/v1/nodes
```

List Openstack Configuration Command

```
curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H 'Accept: application/json'
 --cacert /var/www/mercury/mercury-ca.crt https://172.31.231.17:8445/v1/openstack_config
```

Response

```
{"KEYSTONE_VERBOSE_LOGGING": true, "GNOCCHI_VERBOSE_LOGGING": true, . . }
```

List Password Secrets

Command

```
curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H 'Accept: application/json'
 --cacert /var/www/mercury/mercury-ca.crt https://172.31.231.17:8445/v1/secrets
```

Response

```
{'HEAT_KEYSTONE_PASSWORD': '***', 'CINDER_KEYSTONE_PASSWORD': '***' . . }
```

Cluster Recovery

Command

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H 'Accept:
application/json' --cacert /var/www/mercury/mercury-ca.crt -d '{'action': {'cluster-recovery': {'run-disk-
checks': False}}}'
https://172.31.231.17:8445/v1/misc
```

Response

```
{'uuid': 'ae3be813-4fae-4510-8467-fab09ac60d2b', 'created_at':
'2019-01-07T08:17:01.229976+00:00', 'updated_at': None, 'operation_status':
'OperationScheduled', 'operation_logs': '', 'operation_name': {'cluster-recovery':
{'run-disk-checks': False}}}
```

Last-Run-Status

Command

```
curl -i -X GET -H 'Content-Type: application/json' -H 'Authorization: ****' -H 'Accept: application/json' -H
'User-Agent: python-ciscovimclient' --cacert
 /var/www/mercury/mercury-ca.crt https://172.31.231.17:8445/v1/op_info
```

Response

```
{'created_at': '2019-01-07 08:27:56+00:00', 'updated_at': '2019-01-07 08:28:03+00:00',
 'reboot_required': False, 'update_status': False, 'current_op_logs':
 'https://172.31.231.17:8008/mercury/79c402d2-f156-4ba2-8f17-ec109401a538',
 'current_op_status': 'OperationRunning', 'insight_monitor_status':
 'Running', 'current_op_name': 'Generate_ssh_keys', 'current_op_monitor':
 'Runner_Op_Generate_ssh_keys'}
```

Reconfigure Regenerate Secrets

Command

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H 'Accept: application/json' --cacert /var
/www/mercury/mercury-ca.crt -d
'{'action': {'regenerate_secrets': '****', 'reconfigure': True}}' https://172.31.231.17:8445/v1/misc
```

Response

```
{'uuid': '83cf2700-275f-4c18-a900-96c36c4987aa', 'created_at': '2019-01-07T08:36:19.279425+00:00',
'updated_at': None, 'operation_status': 'OperationScheduled',
 'operation_logs': '',
 'operation_name': {'regenerate_secrets': '****',
 'reconfigure': True}}
```

Reconfigure Set Password

Command

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H 'Accept: application/json' --cacert /var
/www/mercury/mercury-ca.crt -d '{"action": {"setpassword": {"HAPROXY_PASSWORD": "******"}, "reconfigure":
true}}' https://172.31.231.17:8445/v1/misc
```

Response

```
{"uuid": "16d89b9e-cadc-4467-b1d8-5a8a60171d90", "created_at": "2020-06-30T16:51:17.316126+00:00",
"updated_at": null, "operation_status": "OperationScheduled", "operation_logs": "", "operation_name": "{\"
setpassword\": {\"HAPROXY_PASSWORD\": \"****\"}, \"reconfigure\": true}"}
```

Reconfigure Set Openstack Configuration

Command

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H 'Accept:
 application/json' --cacert /var/www/mercury/mercury-ca.crt -d '{'action': {'reconfigure': True,
'setopenstackconfigs':{'GNOCCHI_VERBOSE_LOGGING': True}}}'
https://172.31.231.17:8445/v1/misc
```

Response

```
{'uuid': '5bbbeff7-76df-4444-a38a-8819a8b579e4', 'created_at':
 '2019-01-07T08:54:13.733254+00:00', 'updated_at': None,
 'operation_status': 'OperationScheduled', 'operation_logs': '',
 'operation_name': {'setopenstackconfigs':
 {'GNOCCHI_VERBOSE_LOGGING': True}, 'reconfigure': True}}
```

Reconfigure CIMC Password

1. List down the setupdata and find UUID of active setupdata using the following command:

```
curl -i -X GET -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt https://172.31.231.17:8445/v1
/setupdata
```

Response



2. Put the content of setupdata with new CIMC Password using the following command:

```
curl -i -X PUT -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt -d
'{'meta': {}, 'name': 'NEWSETUPDATA', 'jsondata':
{'external_lb_vip_address': '172.29.86.9' . . .}, 'uuid':
'3e97381e-4b1c-41a2-9af4-f970a1f1493a'}' https://172.31.231.17:8445/v1/setupdata/3e97381e-4b1c-41a2-9af4-
f970a1f1493a
```

3. Post on Misc API using the below command:

```
curl -i -X POST -H 'Content-Type: application/json' -u admin:**** -H
'Accept: application/json' --cacert /var/www/mercury/mercury-ca.crt -d
'{'action':
 {'reconfigure_cimc_password': True, 'reconfigure': True}}' https://172.31.231.17:8445/misc
```

Response

```
{'uuid': 'f00e1ae0-5674-4218-b1de-8995c9f9c546', 'created_at':
'2019-01-07T09:19:40.210121+00:00', 'updated_at': None, 'operation_status':
'OperationScheduled', 'operation_logs': '', 'operation_name':
{'reconfigure_cimc_password': '****', 'reconfigure': True}}
```

API Resources

API Resources

- Setupdata
- Install resource
- Nodes
- Replace a controller
- Offline validation
- Update
- Secrets
- OpenStack Configs
- Version
- Health of the Management Node
- Hardware Information
- Release mapping Information

Setupdata

REST wrapper for setupdata. Provides methods for listing, creating, modifying, and deleting setupdata.

Retrieving the setupdata

Resource URI

Verb	URI
GET	/v1/setupdata

Example

JSON Request

```
GET /v1/setupdata
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"setupdatas": [{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}]}
```

Creating setupdata

Resource URI

Verb	URI
POST	/v1/setupdata

Example

JSON Request

```
POST /v1/setupdata
Accept: application/json
{
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
```

JSON Response

```
201 OK
Content-Type: application/json
{
        "status": "Active",
        "name":"GG34",
        "uuid": "123"
         "meta":{
         "user":"root"
},
"jsondata":{
.......
 }
}

400 Bad Request
Content-Type: application/json
{
       "debuginfo": null
       "faultcode":"Client"
       "faultstring": "Error"
}
409 CONFLICT
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Error"
}
```

Retrieving a single setupdata

Resource URI

Verb	URI
GET	/v1/setupdata/(id)

Property:
id—The ID of the setupdata that you want to retrieve.

Example

JSON Request

```
GET /v1/setupdata/123
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "status": "Active",
        "name":"GG34",
        "uuid": "123"
        "meta":{
        "user":"root"
},
"jsondata":{
.......
        }
}
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
"faultcode": "Client"
"faultstring": "Setupdata could not be found."
}
```

Updating a setupdata

Resource URI

Verb	URI
PUT	/v1/setupdata/(id)

Property:
id—The ID of the setupdata that you want to update. Example

JSON Request

```
PUT /v1/setupdata/123 Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
 "status": "Active",
 "name":"GG34",
 "uuid": "123"
 "meta":{
 "user":"root"
  },
 "jsondata":{
 .......
 }
}
404 NOT FOUND
Content-Type: application/json
{
 "debuginfo": null
 "faultcode": "Client"
 "faultstring": "Setupdata could not be found."
}
```

Deleting a setupdata

Resource URI

Verb	URI
DELETE	/v1/setupdata/(id)

Property:
id—The ID of the setupdata that you want to delete. Example

JSON Request

```
DELETE /v1/setupdata/123 Accept: application/json
```

JSON Response

```
204 NO CONTENT Returned on success
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Setupdata could not be found."
}
400 BAD REQUEST
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Setupdata cannot be deleted when it is being used by an installation"
}
```

Install resource

REST wrapper for install. Provides methods for starting, stopping, and viewing the status of the installation process.

Return a list of installation

Resource URI

Verb	URI
GET	/v1/install

Example

JSON Request

```
GET /v1/install
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
  {"installs": [{
        "ceph": "Skipped",
        "uuid": "123",
        "setupdata": "345",
        "vmtpresult": "{
        "status": "PASS",
        "EXT_NET": []
  }",
        "baremetal": "Success",
        "orchestration": "Success",
        "validationstatus": "{
        "status": "PASS",
        "Software_Validation": [],
        "Hardware_Validation": []
  }",
    "currentstatus": "Completed",
        "validation": "Success",
        "hostsetup": "Success",
        "vmtp": "Skipped"
  }]
}
```

Create an installation

Resource URI

Verb	URI
POST	/v1/install

Example

JSON Request

```
GET /v1/install
Accept: application/js
{
        "setupdata": "123",
        "stages": [
        "validation",
        "bootstrap",
        "runtimevalidation",
        "baremetal",
        "orchestration",
        "hostsetup",
        "ceph",
        "vmtp"
        ]
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
        "ceph": "Skipped",
        "uuid": "123",
        "setupdata": "345",
        "vmtpresult": "{
        "status": "PASS",
        "EXT_NET": []
}",
        "baremetal": "Success",
        "orchestration": "Success",
        "validationstatus": "{
        "status": "PASS",
        "Software_Validation": [],
        "Hardware_Validation": []
}",
        "currentstatus": "Completed",
        "validation": "Success",
        "hostsetup": "Success",
        "vmtp": "Skipped"
}
409 CONFLICT
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Install already exists"
}
```

Retrieve the installation

Resource URI

Verb	URI
GET	/v1/install/{id}

Property:
id—The ID of the installation that you want to retrieve. Example

JSON Request

```
GET /v1/install/345
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "ceph": "Skipped",
        "uuid": "123",
        "setupdata": "345",
        "vmtpresult": "{
        "status": "PASS",
        "EXT_NET": []
}",
        "baremetal": "Success",
        "orchestration": "Success",
        "validationstatus": "{
        "status": "PASS",
        "Software_Validation": [],
        "Hardware_Validation": []
}",
        "currentstatus": "Completed",
        "validation": "Success",
        "hostsetup": "Success",
        "vmtp": "Skipped"
}
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Install doesn't exists"
}
```

Stop the installation

Resource URI

Verb	URI
DELETE	/v1/install/{id}

Property:
id—The ID of the installation that you want to stop. Example

JSON Request

```
DELETE /v1/install/345
Accept: application/js
```

JSON Response

```
204 NO CONTENT
 Content-Type: application/json

404 NOT FOUND
 Content-Type: application/json
 {
         "debuginfo": null "faultcode": "Client"
         "faultstring": "Install doesn't exists"
 }
```

Nodes

Getting a list of nodes
Resource URI

Verb	URI

GET	/v1/nodes

Example

JSON Request

```
Get /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "nodes": [
        [
                "status": "Active",
                "uuid": "456",
                "setupdata": "123",
                "node_data": "{
                "rack_info": {
                "rack_id": "RackA"
                },
                "cimc_info": {
                  "cimc_ip": "10.10.10.10"
                },
        "management_ip": "7.7.7.10"
          }",
        "updated_at": null,
        "mtype": "compute",
        "install": "345",
        "install_logs": "logurl",
        "created_at":"2016-0710T06:17:03.761152",
        "name": " compute-1"
        }
   ]
}
```

Add New Nodes

The nodes are in compute or block_storage type. Before adding the nodes to the system, the name of the nodes and other necessary information like cimc_ip and rackid must be updated in the setupdata object. If the setupdata object is not updated, the post call does not allow you to add the node.
Resource URI

Verb	**URI**
POST	/v1/nodes

Example

JSON Request

```
POST /v1/nodes
Accept: application/js
 {
 "name" : "compute-5"
 }
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
        "status": "ToAdd",
        "uuid": "456",
        "setupdata": "123",
        "node_data": "{
        "rack_info": {
        "rack_id": "RackA"
        },
        "cimc_info": {
        "cimc_ip": "10.10.10.10"
        },
        "management_ip": "7.7.7.10"
        }",
        "updated_at": null,
        "mtype": "compute",
        "install": "345",
        "install_logs": "logurl",
        "created_at":"2016-0710T06:17:03.761152",
        "name": " compute-1"
}
```

Retrieve information about a particular node

Resource URI

Verb	URI
GET	/v1/nodes{id}

Property:
id—The ID of the node that you want to retrieve. Example

JSON Request

```
POST /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "status": "Active",
        "uuid": "456",
        "setupdata": "123",
        "node_data": "{
        "rack_info": {
        "rack_id": "RackA"
        },
        "cimc_info": {
        "cimc_ip": "10.10.10.10"
        },
        "management_ip": "7.7.7.10"
        }",
        "updated_at": null,
        "mtype": "compute",
        "install": "345",
        "install_logs": "logurl",
        "created_at":"2016-0710T06:17:03.761152",
        "name": " compute-1"
}
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Node doesn't exists"
}
```

Remove a Node

The node to be deleted must be removed from the setupdata object. Once the setupdata object is updated, you can safely delete the node. The node object cannot be deleted until it calls the remove node backend and succeeds.
Resource URI

Verb	URI
DELETE	/v1/nodes{id}

Property:
id—The ID of the node that you want to remove. Example

JSON Request

```
DELETE /v1/nodes/456
Accept: application/js
```

JSON Response

```
204 ACCEPTED
Content-Type: application/json
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Node doesn't exists"
}
```

To clear the database and delete the entries in the nodes, the delete API is called with special parameters that are passed along with the delete request. The JSON parameters are in the following format.

JSON Request

```
DELETE /v1/nodes/456
Accept: application/js
 {
 "clear_db_entry":"True"\
 }
```

JSON Response

```
204 ACCEPTED
 Content-Type: application/json

 404 NOT FOUND
 Content-Type: application/json
 {
         "debuginfo": null
         "faultcode": "Client"
         "faultstring": "Node doesn't exists"
 }
```


This is done only if the node is deleted from the REST API database. The failure reason of the node must be rectified manually apart from the API. True is a string and not a boolean in the preceding line.

Replace a controller

Resource URI

Verb	URI
PUT	/v1/nodes{id}

Property:
id—The ID of the controller that you want to replace. Example

JSON Request

```
PUT /v1/nodes/456 Accept: application/js
```

JSON Response

```
200 OK
 Content-Type: application/json
 404 NOT FOUND
 Content-Type: application/json
 {
         "debuginfo": null
         "faultcode": "Client"
          "faultstring": "Node doesn't exists"
 }
```

Offline validation

REST wrapper does the offline validation of setupdata. Rest Wrapper does only the Software Validation of the input setupdata.

Create an offline validation operation

Resource URI

Verb	URI
POST	/v1/offlinevalidation

Example

JSON Request

```
POST /v1/offlinevalidation
Accept: application/json
 {
 "jsondata": "."
 }
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "NotValidated",
"uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69",
"created_at": "2016-02-03T02:05:28.384274",
"updated_at": "2016-02-03T02:05:51.880785",
"jsondata": "{}",
"validationstatus": {
"status": "PASS",
"Software_Validation": [],
"Hardware_Validation": []
}
}
```

Retrieve the results of offline validation

Resource URI

Verb	URI
GET	/v1/offlinevalidation

Property:
id—The ID of the node you want to retrieve. Example

JSON Request

```
GET /v1/offlinevalidation/789
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": " ValidationSuccess",
"uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69",
"created_at": "2016-02-03T02:05:28.384274",
"updated_at": "2016-02-03T02:05:51.880785",
"jsondata": "{}",
"validationstatus": {
"status": "PASS",
"Software_Validation": [],
"Hardware_Validation": []
}
}
```

Update

Start an Update Process
Resource URI

Verb	URI

POST	/v1/update

Parameters:

- fileupload - "tar file to upload"
- filename - "Filename being uploaded"

Example

JSON Request

```
curl -sS -X POST --form
"fileupload=@Test/installer.good.tgz" --form
"filename=installer.good.tgz"
https://10.10.10.8445/v1/update
```


This curl request is done as a form request.

JSON Response

```
200 OK
 Content-Type: application/json
 {
        "update_logs": "logurl",
         "update_status": "UpdateSuccess",
        "update_filename": "installer-4579.tgz",
        "created_at": "2016-07-10T18:33:52.698656",
        "updated_at": "2016-07-10T18:54:56.885083"
 }
 409 CONFLICT
 Content-Type: application/json
 {
 "debuginfo": null "faultcode": "Client"
 "faultstring": "Uploaded file is not in tar format"
 }
```

Roll back an update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
PUT /v1/update
 Accept: application/json
 {
 "action":"rollback"
 }
```

JSON Response

```
200 OK
 Content-Type: application/json
 {

 "update_logs": "logurl", "update_status": "ToRollback", "update_filename": "installer-4579.tgz",
 "created_at": "2016-07-10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
 }
```

Commit an update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
PUT /v1/update
 Accept: application/json
 {
 "action":"commit"
 }
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 "update_logs": "logurl", "update_status": "ToCommit", "update_filename": "installer-4579.tgz",
 "created_at": "2016-07-10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
 }
```

Retrieve the details of an update

Resource URI

Verb	URI
GET	/v1/update

Example

JSON Request

```
GET /v1/update
 Accept: application/json
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
         "update_logs": "logurl",
        "update_status": "UpdateSuccess",
        "update_filename": "installer-4579.tgz",
    "created_at": "2016-07-10T18:33:52.698656",
        "updated_at": "2016-07-10T18:54:56.885083"
 }
```

Secrets

Retrieve the list of secrets that are associated with the OpenStack Setup
You can retrieve the set of secret password that are associated with the OpenStack setup using the preceding api. This gives the list of secrets for each service in OpenStack.
Resource URI

Verb	URI
GET	/v1/secrets

Example

JSON Request

```
GET /v1/secrets
 Accept: application/json
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 "HEAT_KEYSTONE_PASSWORD": "xxxx", "CINDER_KEYSTONE_PASSWORD": "xxxxx",
 ....
 ....
 "RABBITMQ_PASSWORD": "xxxxx"
 }
```

OpenStack Configs

Retrieve the list of OpenStack configs associated with the OpenStack Setup
You can retrieve the set of OpenStack configs associated with the OpenStack setup using the preceding api. This gives the current settings of different configurations such as verbose logging and debug logging for different OpenStack services.

Verb	URI
GET	/v1/secrets

JSON Request

```
GET /v1/openstack_config
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"CINDER_DEBUG_LOGGING": false,
"KEYSTONE_DEBUG_LOGGING": false,
....
....
"NOVA_VERBOSE_LOGGING": true
}
```

Version

Retrieve the version of the Cisco Virtualized Infrastructure Manager. Resource URI

Verb	URI
GET	/v1/version

Example

JSON Request

GET /v1/version
Accept: application/json

JSON Response

200 OK
Content-Type: application/json
{"version": "1.9.1"}

Health of the Management Node

Retrieve the health of the Management node
This API is used to retrieve the health of the management node. It checks various parameters such as partitions, space and so on.
Resource URI

Verb	URI
GET	/v1/health

Example

JSON Request

GET /v1/health
Accept: application/json

JSON Response

200 OK
Content-Type: application/json
{
"status": "PASS", "pod_status": { "color": "BLUE",
"version": "<VERSION_NO.>"
},
"insight_version": "<VERSION_NO.>"
}
Color signifies the health of the pod for Insight:

- Grey signifies that no installation is kicked off on the pod.
- Green signifies that everything is in Good state and cloud installation is active.
- Blue signifies that some operation is running on the pod.
- Red signifies that the pod is in critical state and you might need TAC support to recover the pod.
- Amber indicates a warning if a pod management (Add/Remove/Replace) operation failed.

Hardware Information

REST wrapper to do hardware information of setupdata. This returns the hardware information of all hardware available in the setupdata.

Create a HWinfo operation

Resource URI

Verb	URI
GET	/v1/hwinfo

Example

JSON Request

POST /v1/hwinfo
Accept: application/json
{
"setupdata":"c94d7973-2fcc-4cd1-832d-453d66e6b3bf"
}

JSON Response

201 CREATED
Content-Type: application/json
{
"status": "hwinfoscheduled",
"uuid": "928216dd-9828-407b-9739-8a7162bd0676",
"setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at": "2017-03-19T13:41:25.488524",
"updated_at": null, "hwinforesult": ""
}

Retrieve the results of Hwinfo Operation

Resource URI

Verb	URI
GET	/v1/hwinfo/{id}

Property:
id—The ID of the node you want to query. Example

JSON Request

GET /v1/hwinfo/789 Accept: application/json

JSON Response

200 OK
Content-Type: application/json
{
"status": "hwinfosuccess",
"uuid": "928216dd-9828-407b-9739-8a7162bd0676",
"setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at": "2017-03-19T13:41:25.488524", "updated_at": "2017-03-19T13:42:05.087491",
"hwinforesult": "{\"172.29.172.73\": {\"firmware\": …….
…….
……..
}

Release mapping Information

This api is used to see the list of Features included and list of options which can be reconfigured in the Openstack Setup.

Retrieve the release mapping information

Resource URI

Verb	URI
GET	/v1/releasemapping

JSON Request

GET /v1/releasemapping Accept: application/json

JSON Response

200 OK
Content-Type: application/json [
{
"SWIFTSTACK": {
"feature_status": true,
],
"desc": "swift stack feature"
}
},.....
.....
}

POST Install operations

The following are the post install operations that can be performed, after the successful installation of OpenStack. It uses a common api. Following is an Example:

1. reconfigure
2. reconfigure -regenerate passwords
3. reconfigure -setpasswords,setopenstack_configs
4. reconfigure -alertmanager_config, -alerting_rules_config
5. check-fernet-keys
6. resync-fernet-keys
7. rotate-fernet-keys

Create a post install operation

Resource URI

Verb	URI
POST	/v1/misc

Examples:

JSON Request
POST /v1/misc
Accept: application/json
{"action": {"reconfigure": true}}

JSON Response

201 CREATED
Content-Type: application/json
{
"uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733", "created_at": "2017-03-19T14:03:39.723914",
"updated_at": null,
"operation_status": "OperationScheduled", "operation_logs": "",
"operation_name": "{"reconfigure": true}"
}

JSON Request

POST /v1/misc
Accept: application/json
{"action": {"reconfigure": true, "alertmanager_config": <json_config>}}

JSON Response

201 CREATED
Content-Type: application/json
{
"uuid": "68b67265-8f09-480e-8608-b8aff77e0ec7", "created_at": "2019-01-09T16:42:11.484604+00:00",
"updated_at": null,

"operation_status": "OperationScheduled", "operation_logs": "",
"operation_name": "{"alertmanager_config": <json_config>, "reconfigure": true}"
}

Retrieve a status of the post install operation

Resource URI

Verb	URI
GET	/v1/misc

Example

JSON Request

GET /v1/misc
Accept: application/json

JSON Response

201 CREATED
Content-Type: application/json
{
"uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733", "created_at": "2017-03-19T14:03:39.723914", "updated_at": "2017-03-19T14:03:42.181180", "operation_status": "OperationRunning", "operation_logs": "xxxxxxxxxxxxxxxx", "operation_name": "{\"reconfigure\": true}"
}
In VIM Rest APIs exist to support NFVBench, query hardware information and to get a list of optional and mandatory features that the pod supports. Following are the API details:

NFVBench Network Performance Testing Create NFVBench Run

Starts the network performance test with provided configuration. REST API To Create Fixed Rate Test

Verb	URI
Post	v1/nfvbench/ create_ndr_pdr_test

Example

JSON Request

POST Request URL
/v1/nfvbench/create_fixed_rate_test JSON Request:
{"nfvbench_request":
{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [

"64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}
}

JSON Response

201 CREATED
Content-Type: application/json
{
"status": "not_run", "nfvbench_request":
'{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [
"64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}',
"created_at": "2017-08-16T06:14:54.219106",
"updated_at": null, "nfvbench_result": "", "test_name": "Fixed_Rate_Test"
}

Status Polling

Polling of NFVbench run status which is one of nfvbench_running, nfvbench_failed, nfvbench_completed.

Resource URI

Verb	URI
GET	v1/nfvbench/<test_name>

REST API To Get Fixed Rate Test Result
GET Request URL
/v1/upgrade/get_fixed_rate_test_result JSON Response:

Check If NFVbench Test is running
200 OK
Content-Type: application/json

{
"status": "nfvbench_running",
"nfvbench_request": '{"traffic": {"bidirectional": true, "profile": "custom"},
"rate": "1000000pps",
"traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}], "duration_sec": 60, "flow_count": 1000}',
"nfvbench_result": ""
"created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-30T21:40:41.367279",
}
Check If NFVbench Test is completed
200 OK
Content-Type: application/json
{
"status": "nfvbench_completed",
"nfvbench_request": '{"traffic": {"bidirectional": true, "profile": "custom"}, "rate": "1000000pps",
"traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}], "duration_sec": 60, "flow_count": 1000}',
"nfvbench_result": '{"status": "PROCESSED", "message": {"date": "2017-08-15 23:15:04", "nfvbench_version": "0.9.3.dev2", ...}
"created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-30T22:29:56.970779",
}
REST API to create NDR/PDR Test
POST Request URL
/v1/nfvbench/create_ndr_pdr_test
Accept: application/json
{"nfvbench_request":
{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [
"64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}
}
JSON Response
201 CREATED
Content-Type: application/json
{
"status": "not_run", "nfvbench_request":
'{
"duration_sec": 20, "traffic_profile": [
{
"name": "custom", "l2frame_size": [

"64",
"IMIX", "1518"
]
}
],
"traffic": { "bidirectional": true, "profile": "custom"
},
"flow_count": 1000
}'
"created_at": "2017-08-16T07:18:41.652891",
"updated_at": null, "nfvbench_result": "", "test_name": "NDR_PDR_Test"
}
REST API To Get NDR/PDR Test Results
GET Request URL
/v1/ nfvbench/get_ndr_pdr_test_result
JSON Response:
If NFVbench NDR/PDR test is running
200 OK
Content-Type: application/json
{
"status": "nfvbench_running",
"nfvbench_request": '{"duration_sec": 20,
"traffic": {"bidirectional": true, "profile": "custom"},
"traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name": "custom"}], "flow_count": 1000}',
"nfvbench_result": ""
"created_at": "2017-08-16T07:18:41.652891", "updated_at": "2017-09-30T22:29:56.970779",
}
If NFVbench NDR/PDR test is completed

200 OK
Content-Type: application/json
{
"status": "nfvbench_completed", "nfvbench_request": '{"duration_sec": 20,
"traffic": {"bidirectional": true, "profile": "custom"},
"traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name": "custom"}], "flow_count": 1000}',
"nfvbench_result": '{"status": "PROCESSED",...}' "created_at": "2017-08-16T07:18:41.652891", "updated_at": "2017-09-30T22:29:56.970779",
}

REST API to Get Node Hardware Information

Rest API helps you to get the hardware information of all the nodes in the POD through CIMC/UCSM.

- Total Memory
- Firmware Info (Model, Serial Number)
- CIMC IP

GET Request URL
/v1/hwinfo Output Response
{
"hwinforesult": "{"control-server-2": {"memory": {"total_memory": "131072"}, "firmware": {"serial_number": "FCH1905V16Q", "fw_model": "UCSC-C220-M4S"}, "cimc_ip": "172.31.230.100", "storage": {"num_storage": 4}, "cisco_vic_adapters": {"product_name": "UCS VIC 1225"},
"cpu": {"number_of_cores": "24"}, "power_supply": {"power_state": "on"}}
…
}

REST API to Get Mandatory Features Mapping

POST Request URL
/v1/releasemapping/mandatory_features_mapping
JSON Response:

{
"mandatory": { "networkType": {
"C": {
"feature_status": true,
"values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],
"insight_label": "Tenant Network", "desc": "Tenant Network"
},
"B": {
"feature_status": true,
"values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],
"insight_label": "Tenant Network", "desc": "Tenant Network"
}
},
"cephMode": {
"all": {
"feature_status": true,
"values": [{"name": "Central", "value": "Central"},], "insight_label": "Ceph Mode",
"desc": "Ceph Mode"
}
},
"podType": {
"C": {
"feature_status": true,
"values": [{"name": "Fullon", "value": "fullon"},], "insight_label": "POD Type",
"desc": "POD Type"
}, "B": {
"feature_status": true,
"values": [{"name": "Fullon", "value": "fullon"},], "insight_label": "POD Type",
"desc": "POD Type"
}
},
"installMode": { "all": {
"feature_status": true,
"values": [{"name": "Connected", "value": "connected"},], "insight_label": "Install Mode",
"desc": "Install Mode"
}

}
},
"platformType": [{"name": "B-series", "value": "B"}, {"name": "C-series", "value": "C"}],
"postinstalllinks": {
"view_cloudpulse": {"alwayson": true, "feature_status": true, "platformtype": "all", "insight_label": "Run VMTP", "desc": "Cloudpluse"},
"password_reconfigure": {"alwayson": true, "feature_status": true, "platformtype": "all", "insight_label": "Reconfigure Passwords", "desc": "Reconfigure Passwords"}
}
}

REST API to Get Optional Features Mapping

POST Request URL
/v1/releasemapping/optional_features_mapping
JSON Response: [
{
"SWIFTSTACK": {
"feature_status": true, "insight_label": "Swiftstack", "repeated_redeployment": true,
"reconfigurable": ["cluster_api_endpoint", "reseller_prefix", "admin_password",
"protocol"],
"desc": "swift stack feature"
}
},
{
"heat": {
"feature_status": true, "insight_label": "Heat", "repeated_redeployment": false, "reconfigurable": ["all"], "desc": "Openstack HEAT service"
}
},
..... other features
]

Cloud sanity information

REST wrapper to run cloud-sanity test suites. The cloud-sanity extension to the VIM REST API enables support for managing cloud-sanity test actions

Create a cloud-sanity test

Verb	URI
Post	/v1/cloud-sanity/create

Example

JSON Request

POST /v1/cloudsanity/create Accept: application/json
'{"cloudsanity_request": {"command": "create",
"action": "test", "test_name": "cephmon", "uuid": ""}}'
test_name can be all,management,control,compute,cephmon,cephosd

JSON Response

201 Created
{
'cloudsanity_request': "{u'action': u'test', u'command': u'create', u'uuid': '5dff1662-3d33-4901-808d-479927c01dde',
u'test_name': u'cephmon'}", 'cloudsanity_result': '',
'created_at': '2018-01-26T20:32:20.436445',
'status': 'not_run', 'test_name': 'cephmon', 'updated_at': ''
}

List cloud-sanity test results

Verb	URI
GET	/v1/cloud-sanity

JSON Request
GET /v1/cloudsanity

JSON Response

200 OK
{ '0b91746f-90b4-4355-a748-727c2e5c59c5': { 'action': 'test',
'created_at': '2018-01-25 12:08:22',
'status': 'cloudsanity_completed', 'test_name': 'management',
'uuid': '0b91746f-90b4-4355-a748-727c2e5c59c5'},
'5695cb31-39e4-4be2-9dee-09e7daffc2e7': { 'action': 'test',
'created_at': '2018-01-25 12:03:06',
'status': 'cloudsanity_completed', 'test_name': 'compute',
'uuid': '5695cb31-39e4-4be2-9dee-09e7daffc2e7'},
'5dff1662-3d33-4901-808d-479927c01dde': { 'action': 'test',
'created_at': '2018-01-26 20:32:20',
'status': 'cloudsanity_completed', 'test_name': 'cephmon',
'uuid': '5dff1662-3d33-4901-808d-479927c01dde'},
'7946255d-df58-4432-b729-20cf16eb5ba5': { 'action': 'test',
'created_at': '2018-01-25 12:05:56',
'status': 'cloudsanity_completed', 'test_name': 'cephosd',
'uuid': '7946255d-df58-4432-b729-20cf16eb5ba5'},
'797d79ba-9ee0-4e11-9d9e-47791dd05e07': { 'action': 'test',
'created_at': '2018-01-25 12:05:11',
'status': 'cloudsanity_completed', 'test_name': 'cephmon',
'uuid': '797d79ba-9ee0-4e11-9d9e-47791dd05e07'},
'962e2c8e-c7b0-4e24-87c1-528cad84002c': { 'action': 'test',
'created_at': '2018-01-26 18:52:31',
'status': 'cloudsanity_completed', 'test_name': 'control',
'uuid': '962e2c8e-c7b0-4e24-87c1-528cad84002c'}, 'd0111530-ee3b-45df-994c-a0917fd18e11': { 'action': 'test',

'created_at': '2018-01-26 18:46:23',
'status': 'cloudsanity_completed', 'test_name': 'control',
'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11'}}

List specific cloud-sanity test results

Verb	URI
GET	/v1/cloud-sanity/list/?test_name={all, management, control,compute,cephmon,cephosd}

JSON Request
GET /v1/cloudsanity/list/?test_name=cephmon Accept: application/json

JSON Response

200 OK
{ '5dff1662-3d33-4901-808d-479927c01dde': { 'action': 'test',
'created_at': '2018-01-26 20:32:20',
'status': 'cloudsanity_completed', 'test_name': 'cephmon',
'uuid': '5dff1662-3d33-4901-808d-479927c01dde'},
'797d79ba-9ee0-4e11-9d9e-47791dd05e07': { 'action': 'test',
'created_at': '2018-01-25 12:05:11',
'status': 'cloudsanity_completed', 'test_name': 'cephmon',
'uuid': '797d79ba-9ee0-4e11-9d9e-47791dd05e07'}}

Show cloud-sanity test results

Verb	URI
GET	/v1/cloud-sanity/show/?uuid=<uuid>

JSON Request
GET /v1/cloudsanity/show/?uuid=d0111530-ee3b-45df-994c-a0917fd18e11

JSON Response

200 OK
{ 'action': 'test', 'cloudsanity_request':
"{u'action': u'test', u'command': u'create',
u'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11', u'test_name': u'control'}",
'cloudsanity_result': '{"status": "PROCESSED",
"message": {"status": "Pass",
"message": "[PASSED] Cloud Sanity Control Checks Passed", "results": {"control": {"ping_all_controller_nodes": "PASSED",
"check_rabbitmq_is_running": "PASSED", "check_rabbitmq_cluster_status": "PASSED", "check_nova_service_list": "PASSED", "ping_internal_vip":
"PASSED",

'created_at': '2018-01-26 18:46:23',
'status': 'cloudsanity_completed', 'test_name': 'control', 'updated_at': '2018-01-26 18:47:58',
"disk_maintenance_raid_health": "PASSED", "check_mariadb_cluster_size": "PASSED", "disk_maintenance_vd_health": "PASSED"}}}}',
'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11'}

Delete cloud-sanity test results

Verb	URI
DELETE	/v1/cloud-sanity/delete/?uuid=<uuid>

JSON Request
GET /v1/cloudsanity/delete/?uuid=444aa4c8-d2ba-4379-b035-0f47c686d1c4

JSON Response

200 OK
{
"status": "deleted",
"message": "UUID 444aa4c8-d2ba-4379-b035-0f47c686d1c4 deleted from database", "uuid": "444aa4c8-d2ba-4379-b035-0f47c686d1c4",
"error": "None"
}

Disk Maintenance information

REST wrapper to query information about RAID disks on Pod nodes. This returns the RAID disk information of all or a selection of RAID disks available in the Pod.
The disk management extension to the VIM REST API enables support for Disk Management actions

Create a Check disk operation

Resource URI

Verb	URI
POST	/v1/diskmgmt/check_disks

Example

JSON Request

POST /v1/diskmgmt/check_disks Accept: application/json '{"diskmgmt_request": {"command": "create",
"action": "check-disks",
"role": "control",
"locator": "False", "json_display": "False", "servers": "", "uuid": ""}}'

JSON Response

201 Created
Content-Type: application/json
{
'action': 'check-disks',
'created_at': '2018-03-08T02:03:18.170849+00:00',
'diskmgmt_request': "{u'uuid': '0729bdea-cc19-440f-8339-ab21e76be84b',

u'json_display': u'False',
u'servers': u'',
u'locator': u'False', u'role': u'control', u'action': u'check-disks', u'command': u'create'}",
'diskmgmt_result': '', 'status': 'not_run', 'updated_at': 'None'
}

Create a replace disk operation

Verb	URI
POST	/v1/diskmgmt/replace_disks

Example

JSON Request

POST /v1/diskmgmt/replace_disks Accept: application/json
'{"diskmgmt_request": {"command": "create",
"action": "replace-disks", "role": "control",
"locator": "False", "json_display": "False", "servers": "", "uuid": ""}}'

JSON Response

201 Created
Content-Type: application/json
{
"status": "not_run",
"diskmgmt_request": "{u'uuid': 'cb353f41-6d25-4190-9386-330e971603c9', u'json_display': u'False',
u'servers': u'', u'locator': u'False', u'role': u'control',
u'action': u'replace-disks', u'command': u'create'}",
"created_at": "2018-03-09T12:43:41.289531+00:00",
"updated_at": "", "diskmgmt_result": "", "action": "replace-disks"}

List check disk operation

Verb	URI
GET	/v1/diskmgmt/list/?action={check-disks,replace-disks\&role={all,management,control,compute}

Example

JSON Request

GET /v1/diskmgmt/list/?action=check-disks\&role=all

JSON Response

200 OK
Content-Type: application/json
{
'0be7a55a-37fe-43a1-a975-cbf93ac78893': {'action': 'check-disks',
'created_at': '2018-03-05 14:45:45+00:00',
'role': 'compute',
'status': 'diskmgmt_completed', 'uuid':
'0be7a55a-37fe-43a1-a975-cbf93ac78893'},
'861d4d73-ffee-40bf-9348-13afc697ee3d': {'action': 'check-disks',
'created_at': '2018-03-05 14:44:47+00:00',
'role': 'control',
'status': 'diskmgmt_completed', 'uuid':
'861d4d73-ffee-40bf-9348-13afc697ee3d'},
'cdfd18c1-6346-47a2-b0f5-661305b5d160': {'action': 'check-disks',
'created_at': '2018-03-05 14:43:50+00:00',
'role': 'all',
'status': 'diskmgmt_completed', 'uuid':
'cdfd18c1-6346-47a2-b0f5-661305b5d160'}
}
}

Show a completed diskmgmt operation

Verb	URI
GET	v1/diskmgmt/show/?uuid=<uuid>

Example

JSON Request

GET /v1/diskmgmt/show/?uuid=d24036c6-4557-4c12-8695-a92f6f9315ed

JSON Response

200 OK
Content-Type: application/json
{'action': 'check-disks',
'created_at': '2018-03-07 21:46:41+00:00',
'diskmgmt_request': "{u'uuid': 'd24036c6-4557-4c12-8695-a92f6f9315ed',
u'json_display': False, u'servers': u'f24-michigan-micro-2', u'locator': False,
u'role': u'compute', u'action': u'check-disks', u'command': u'create'}",
'diskmgmt_result': '{"status": "PROCESSED", "message": ["{\'Overall_Status\': \'PASS\',
\'Result\': {\'fcfg_disks_results_list\': [], \'spare_disks_results_list\': [],
\'raid_results_list\': [{\'RAID level\': \'RAID1\', \'Disk Med\': \'HDD\', \'server\':
\'7.7.7.6\', \'RAID type\': \'HW\', \'host\': \'f24-michigan-micro-2\', \'role\':
\'block_storage control compute\', \'VD health\': \'Optl\', \'Num VDs\': 1, \'Num PDs\': 8, \'RAID health\': \'Opt\'}], \'bad_disks_results_list\': [],
\'rbld_disks_results_list\':
[], \'add_as_spares_disks_results_list\': []}}"]}', 'role': 'compute',
'status': 'diskmgmt_completed', 'updated_at': '2018-03-07 21:47:35+00:00',
'uuid': 'd24036c6-4557-4c12-8695-a92f6f9315ed'
}

Delete a completed diskmgmt operation

Verb	URI
DELETE	v1/diskmgmt/delete/?uuid=<uuid>

Example

JSON Request

DELETE /v1/diskmgmt/delete/?uuid=d24036c6-4557-4c12-8695-a92f6f9315ed

JSON Response

200 OK
Content-Type: application/json
{
"status": "deleted",
"message": "UUID d24036c6-4557-4c12-8695-a92f6f9315ed deleted from database", "uuid": "d24036c6-4557-4c12-8695-a92f6f9315ed",
"error": "None"
}

OSD Maintenance information

REST wrapper to query information about OSD on Pod storage nodes. This returns to the OSD status information of all or a selection of OSDs available in the Pod.

Create a OSD disk operation

Verb	URI
POST	/v1/osdmgmt/check_osds

Example

JSON Request

POST /v1/osdmgmt/osdmgmt/check_osds '{"osdmgmt_request": {"command": "create",
"action": "check-osds",
"locator": "False", "json_display": "False", "servers": "",
"osd": "None",
"uuid": ""}}'

JSON Response

201 Created
Content-Type: application/json
{
'action': 'check-osds',
'created_at': '2018-03-08T21:26:15.329195+00:00',
'osdmgmt_request': "{u'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223', u'json_display': u'False', u'servers': u'', u'locator': u'False', u'command':
u'create', u'action':
u'check-osds', u'osd': u'None'}", 'osdmgmt_result': '', 'status': 'not_run', 'updated_at': 'None'
}

Create a replace OSD operation

Verb	URI
POST	v1/osdmgmt/replace_osd

Example

JSON Request

POST /v1/osdmgmt/replace_osd Accept: application/json
'{"osdmgmt_request": {"command": "create",
"action": "replace-osd",
"locator": "False", "json_display": "False", "servers": "f24-michigan-micro-1", "osd": "osd.9",
"uuid": ""}}'

JSON Response

201 Created
Content-Type: application/json
{
"status": "not_run",
"osdmgmt_request": "{u'uuid': '5140f6fb-dca3-4801-8c44-89b293405310', u'json_display': u'False', u'servers': u'f24-michigan-micro-1', u'locator': u'False',
u'command': u'create',
u'action': u'replace-osd', u'osd': u'osd.9'}", "created_at": "2018-03-09T15:07:10.731220+00:00",
"updated_at": null, "action": "replace-osd", "osdmgmt_result": ""
}
}

List check OSD operation

Verb	URI
GET	v1/osdmgmt/list/? action= {check-osds,replace-osd}

Example

JSON Request

GET /v1/osdmgmt/list/?action=check-osds

JSON Response

200 OK
Content-Type: application/json
{
'4efd0be8-a76c-4bc3-89ce-142de458d844': {'action': 'check-osds',
'created_at': '2018-03-08 21:31:01+00:00',
'status': 'osdmgmt_running', 'uuid':
'4efd0be8-a76c-4bc3-89ce-142de458d844'},
'5fd4f9b5-786a-4a21-a70f-bffac70a3f3f': {'action': 'check-osds',
'created_at': '2018-03-08 21:11:13+00:00',
'status': 'osdmgmt_completed', 'uuid':
'5fd4f9b5-786a-4a21-a70f-bffac70a3f3f'},

'9c64ee52-bed5-4b69-91a2-d589411dd223': {'action': 'check-osds',
'created_at': '2018-03-08 21:26:15+00:00',
'status': 'osdmgmt_completed', 'uuid':
'9c64ee52-bed5-4b69-91a2-d589411dd223'}
}
}
Show a completed osdmgmt operation

Verb	URI
GET	v1/osdmgmt/show/?uuid=<uuid>

Example

JSON Request

GET /v1/osdmgmt/show/?uuid=9c64ee52-bed5-4b69-91a2-d589411dd223

JSON Response

200 OK
Content-Type: application/json
{
'action': 'check-osds',
'created_at': '2018-03-08 21:26:15+00:00',
'osdmgmt_request': "{u'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223', u'json_display': u'False', u'servers': u'', u'locator': u'False', u'command':
u'create', u'action':
u'check-osds', u'osd': u'None'}",
'osdmgmt_result': '{"status": "PROCESSED", "message": ["{\'Overall_Status\': \'PASS\',
\'Result\': { ommitted for doc }}]}', 'status': 'osdmgmt_completed', 'updated_at': '2018-03-08 21:27:16+00:00',
'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
}
}

Delete a completed osdmgmt operation

Verb	URI
DELETE	v1/osdmgmt/delete/?uuid=<uuid>

Example

JSON Request

DELETE /v1/osdmgmt/delete/?uuid=9c64ee52-bed5-4b69-91a2-d589411dd223

JSON Response

200 OK
Content-Type: application/json
{
'error': 'None',
'message': 'UUID 9c64ee52-bed5-4b69-91a2-d589411dd223 deleted from database', 'status': 'deleted',
'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
}

}

Hardware Management Utility

REST wrapper to control the execution of or query information from the hardware validation utility.

Create a Validate Operation

Verb	URI
POST	/v1/hardwaremgmt/validate

JSON Request
POST /v1/hardwaremgmt/validate '{"hwmgmt_request": {"command": "create",
"action": "validate", "hosts": "None",
"file": "None", "feature_list": "all", "uuid": ""}}'
feature_list is a comma separated list of valid features for the given POD

JSON Reponse

201 Created
Content-Type: application/json
{
'action': 'validate',
'created_at': '2018-03-08T22:01:22.195232+00:00',
'hwmgmt_request': "{u'feature_list': u'all', u'command': u'create', u'file': None, u'action': u'validate', u'hosts': None, u'uuid': '89e094d8-b246-4620-afca-ba3529385cac'}",
'hwmgmt_result': '',
'status': 'not_run', 'updated_at': 'None'
}

Create a Validate Operation for Failure

Verb	URI
GET	/v1/hardwaremgmt/resolve_failures

JSON Request
POST /v1/hardwaremgmt/resolve_failures
{
"hwmgmt_request": { "command": "create",
"action": "resolve-failures", "hosts": "None",
"file": "None", "feature_list": "all", "uuid": ""}
}
feature_list is a comma separated list of valid features for the given POD

JSON Response

201 Created
Content-Type: application/json
{

"status": "not_run",
"created_at": "2018-03-09T15:47:36.503712+00:00",
"hwmgmt_request": "{u'feature_list': u'all', u'command': u'create', u'file': None, u'action': u'resolve-failures', u'hosts': None, u'uuid':
'49dc1dc9-3170-4f68-b152-0f99bd19f7b1'}",
"updated_at": "",
"action": "resolve-failures", "hwmgmt_result": ""
}

Create a Validate Operation

Verb	URI
GET	v1/hardwaremgmt/list

JSON Request
GET /v1/hardwaremgmt/list

JSON Response

200 OK
Content-Type: application/json
{'89e094d8-b246-4620-afca-ba3529385cac': {'action': 'validate',
'created_at': '2018-03-08 22:01:22+00:00',
'feature_list': 'all',
'status': 'hardwaremgmt_completed', 'uuid':
'89e094d8-b246-4620-afca-ba3529385cac'},
'9f70e872-a888-439a-8661-2d2f36a4f4b1': {'action': 'validate',
'created_at': '2018-03-08 20:34:32+00:00',
'feature_list': 'all',
'status': 'hardwaremgmt_completed', 'uuid':
'9f70e872-a888-439a-8661-2d2f36a4f4b1'}
}

Show a completed hardwaremgmt operation

Verb	URI
GET	/v1/hardwaremgmt /show /?uuid=<uuid>

JSON Request
GET /v1/hardwaremgmt/show/?uuid=9f70e872-a888-439a-8661-2d2f36a4f4b

JSON Response

200 OK
Content-Type: application/json
{
'action': 'validate',
'created_at': '2018-03-08 20:34:32+00:00',
'feature_list': 'all',
'hwmgmt_request': "{u'feature_list': u'all', u'hosts': None, u'file': None, u'action': u'validate', u'command': u'create', u'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'}",
'hwmgmt_result': '{"status": "PROCESSED", "message": "Validate of all completed", "results": {"status": "PASS", "results": [{"status": "PASS", "name": "CIMC Firmware Version
Check", "err": null}, {"status": "PASS", "name": "All Onboard LOM Ports Check", "err":
null}, {"status": "PASS", "name": "PCIe Slot: HBA Status Check", "err": null}, {"status":

"PASS", "name": "Server Power Status Check", "err": null}, {"status": "PASS", "name": "NFV Config Check", "err": null}, {"status": "PASS", "name": "Physical Drives Check", "err":
null}, {"status": "PASS", "name": "PCIe Slot(s) OptionROM Check", "err": null}, {"status": "PASS", "name": "Intel Network Adapter Check", "err": null}]}}',
'status': 'hardwaremgmt_completed', 'updated_at': '2018-03-08 20:38:02+00:00', 'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'

Delete a completed hardwaremgmt operation

Verb	URI
DELETE	/v1/hardwaremgmt/delete/?uuid=<uuid>

JSON Request
DELETE /v1/hardwaremgmt/delete/?uuid=9f70e872-a888-439a-8661-2d2f36a4f4b1

JSON Response

200 OK
Content-Type: application/json
{
'error': 'None',
'message': 'UUID 9f70e872-a888-439a-8661-2d2f36a4f4b1 deleted from database', 'status': 'deleted',
'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'
}

Hardware Management Utility

Hardware Management Utility

- Create a Validate Operation
- Create a Validate Operation for Failure
- Create a Validate Operation
- Show a completed hardwaremgmt operation
- Delete a completed hardwaremgmt operation

REST wrapper to control the execution of or query information from the hardware validation utility.

Create a Validate Operation

Verb	URI
POST	/v1/hardwaremgmt/validate

JSON Request

```
POST /v1/hardwaremgmt/validate
'{"hwmgmt_request": {"command": "create",
                                    "action": "validate", "hosts": "None",
                                        "file": "None",
                                        "feature_list": "all",
                                        "uuid": ""}}'
                                        Feature_list is a comma separated list of valid features for the given
pod.
```

JSON Reponse

```
201 Created
Content-Type: application/json
{
 'action': 'validate',
 'created_at': '2018-03-08T22:01:22.195232+00:00',
 'hwmgmt_request': "{u'feature_list': u'all', u'command': u'create', u'file': None, u'action': u'validate',
u'hosts': None,
  u'uuid': '89e094d8-b246-4620-afca-ba3529385c'}",
 'hwmgmt_result': '',
 'status': 'not_run',
 'updated_at': 'None'
 }
```

Create a Validate Operation for Failure

Verb	URI
GET	/v1/hardwaremgmt/resolve_failures

JSON Request

```
POST /v1/hardwaremgmt/resolve_failures
 {
 "hwmgmt_request": { "command": "create",
                                          "action": "resolve-failures",
                                         "hosts": "None",
                     "file": "None",
                                         "feature_list": "all",
                                         "uuid": ""}
 }
 feature_list is a comma separated list of valid features for the given POD
```

JSON Response

```
201 Created
 Content-Type: application/json
 {

 "status": "not_run",
 "created_at": "2018-03-09T15:47:36.503712+00:00",

 "hwmgmt_request": "{u'feature_list': u'all', u'command': u'create',
 u'file': None, u'action': u'resolve-failures', u'hosts': None, u'uuid':
 '49dc1dc9-3170-4f68-b152-0f99bd19f7b1'}",
 "updated_at": "",
 "action": "resolve-failures",
 "hwmgmt_result": ""
 }
```

Create a Validate Operation

Verb	URI
GET	v1/hardwaremgmt/list

JSON Request

```
GET /v1/hardwaremgmt/list
```

JSON Response

```
200 OK
Content-Type: application/json
{'89e094d8-b246-4620-afca-ba3529385cac': {'action': 'validate',
                                                                                'created_at': '2018-03-08 22:
01:22+00:00',
                                                                                'feature_list': 'all',
                                                                                 'status':
'hardwaremgmt_completed',
                                                                                'uuid': '89e094d8-b246-4620-
afca-ba3529385cac'},
 '9f70e872-a888-439a-8661-2d2f36a4f4b1': {'action': 'validate',
                                                                                'created_at': '2018-03-08 20:
34:32+00:00',
                                                                                'feature_list': 'all',
                                                                                 'status':
'hardwaremgmt_completed',
                                         'uuid':'9f70e872-a888-439a-8661-2d2f36a4f4b1'}
 }
```

Show a completed hardwaremgmt operation

Verb	URI
GET	/v1/hardwaremgmt /show /?uuid=<uuid>

JSON Request

```
GET /v1/hardwaremgmt/show/?uuid=9f70e872-a888-439a-8661-2d2f36a4f4b
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 'action': 'validate',
 'created_at': '2018-03-08 20:34:32+00:00',
 'feature_list': 'all',

 'hwmgmt_request': "{u'feature_list': u'all', u'hosts': None, u'file':
None, u'action': u'validate', u'command': u'create', u'uuid':
'9f70e872-a888-439a-8661-2d2f36a4f4b1'}",
 'hwmgmt_result':
'{"status": "PROCESSED", "message": "Validate of all completed",
"results": {"status": "PASS", "results": [{"status": "PASS", "name":
"CIMC Firmware Version
 Check", "err": null}, {"status": "PASS", "name": "All Onboard LOM Ports Check", "err":
 null}, {"status": "PASS", "name": "PCIe Slot: HBA Status Check", "err": null}, {"status":
"PASS", "name": "Server Power Status Check", "err": null}, {"status":
"PASS", "name": "NFV Config Check", "err": null}, {"status": "PASS",
"name": "Physical Drives Check", "err":
 null}, {"status": "PASS",
"name": "PCIe Slot(s) OptionROM Check", "err": null}, {"status": "PASS",
 "name": "Intel Network Adapter Check", "err": null}]}}',
 'status': 'hardwaremgmt_completed',
 'updated_at': '2018-03-08 20:38:02+00:00',
 'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'
```

Delete a completed hardwaremgmt operation

Verb	URI
DELETE	/v1/hardwaremgmt/delete/?uuid=<uuid>

JSON Request

```
DELETE /v1/hardwaremgmt/delete/?uuid=9f70e872-a888-439a-8661-2d2f36a4f4b1
```

JSON Response

```
200 OK
Content-Type: application/json
{
 'error': 'None',
 'message': 'UUID 9f70e872-a888-439a-8661-2d2f36a4f4b1 deleted from database',
 'status': 'deleted',
 'uuid': '9f70e872-a888-439a-8661-2d2f36a4f4b1'
 }
```

Disk and OSD Maintenance

Disk and OSD Maintenance

- Disk Maintenance information
  - Create a Check disk operation
  - Create a replace disk operation
  - List check disk operation
  - Show a completed diskmgmt operation
  - Delete a completed diskmgmt operation
- OSD Maintenance information
  - Create a OSD disk operation
  - Create a replace OSD operation
  - List check OSD operation
  - Show a completed osdmgmt operation
  - Delete a completed osdmgmt operation

Disk Maintenance information

REST wrapper to query information about RAID disks on Pod nodes. This returns the RAID disk information of all or a selection of RAID disks available in the Pod.
The disk management extension to the VIM REST API enables support for Disk Management actions

Create a Check disk operation

Resource URI

Verb	URI
POST	/v1/diskmgmt/check_disks

Example

JSON Request

```
POST /v1/diskmgmt/check_disks
Accept: application/json
'{"diskmgmt_request": {"command": "create",
                                          "action": "check-disks",
                                          "role": "control",
                                          "locator": "False",
                                          "json_display": "False",
                                          "servers": "",
                                          "uuid": ""}}'
```

JSON Response

```
201 Created
Content-Type: application/json
{
        'action': 'check-disks',
        'created_at': '2018-03-08T02:03:18.170849+00:00',
        'diskmgmt_request': "{u'uuid': '0729bdea-cc19-440f-8339-ab21e76be84b',
                                                u'json_display': u'False',
    u'servers': u'',
        u'locator': u'False',
        u'role': u'control',
        u'action': u'check-disks',
        u'command': u'create'}",
        'diskmgmt_result': '',
        'status': 'not_run',
        'updated_at': 'None'
}
```

Create a replace disk operation

Verb	URI
POST	/v1/diskmgmt/replace_disks

Example

JSON Request

```
POST /v1/diskmgmt/replace_disks
Accept: application/json
 '{"diskmgmt_request": {"command": "create",
                                                  "action": "replace-disks",
                                                 "role": "control",
                                                 "locator": "False",
                                                 "json_display": "False",
                                                 "servers": "", "uuid": ""}}'
```

JSON Response

```
201 Created
Content-Type: application/json
{
 "status": "not_run",
 "diskmgmt_request": "{u'uuid': 'cb353f41-6d25-4190-9386-330e971603c9',
                                          u'json_display': u'False',
                                          u'servers': u'',
                                          u'locator': u'False',
                                          u'role': u'control',
                                          u'action': u'replace-disks',
                                          u'command': u'create'}",
 "created_at": "2018-03-09T12:43:41.289531+00:00",
 "updated_at": "",
 "diskmgmt_result": "",
 "action": "replace-disks"}
```

List check disk operation

Verb	URI
GET	/v1/diskmgmt/list/?action={check-disks,replace-disks \&role={all,management,control,compute}

Example

JSON Request

```
GET /v1/diskmgmt/list/?action=check-disks\&role=all
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 '0be7a55a-37fe-43a1-a975-cbf93ac78893': {'action': 'check-disks',
                                                                                'created_at': '2018-03-05 14:
45:45+00:00',
                                                                                'role': 'compute',
                                                                                'status':
'diskmgmt_completed',
                                                                                'uuid': '0be7a55a-37fe-43a1-
a975-cbf93ac78893'},
 '861d4d73-ffee-40bf-9348-13afc697ee3d': {'action': 'check-disks',
                                                                                 'created_at': '2018-03-05 14:
44:47+00:00',
                                                                                'role': 'control',
                                                                                 'status':
'diskmgmt_completed',
                                                                                'uuid': '861d4d73-ffee-40bf-
9348-13afc697ee3d'},
 'cdfd18c1-6346-47a2-b0f5-661305b5d160': {'action': 'check-disks',
                                                                                'created_at': '2018-03-05 14:
43:50+00:00',
                                          'role': 'all',
                                          'status': 'diskmgmt_completed',
                                          'uuid': 'cdfd18c1-6346-47a2-b0f5-661305b5d160'}}
 }
```

Show a completed diskmgmt operation

Verb	URI
GET	v1/diskmgmt/show/?uuid=<uuid>

Example

JSON Request

```
GET /v1/diskmgmt/show/?uuid=d24036c6-4557-4c12-8695-a92f6f9315ed
```

JSON Response

```
200 OK
Content-Type: application/json
{'action': 'check-disks',
 'created_at': '2018-03-07 21:46:41+00:00',
 'diskmgmt_request': "{u'uuid': 'd24036c6-4557-4c12-8695-a92f6f9315ed',
 u'json_display': False,
 u'servers': u'f24-michigan-micro-2',
 u'locator': False,
 u'role': u'compute',
 u'action': u'check-disks',
 u'command': u'create'}",
 'diskmgmt_result': '{"status": "PROCESSED", "message": ["{\'Overall_Status\': \'PASS\',
 \'Result\': {\'fcfg_disks_results_list\': [], \'spare_disks_results_list\': [],
 \'raid_results_list\': [{\'RAID level\': \'RAID1\', \'Disk Med\': \'HDD\', \'server\':
 \'7.7.7.6\', \'RAID type\': \'HW\', \'host\': \'f24-michigan-micro-2\', \'role\':

 \'block_storage control compute\', \'VD health\': \'Optl\', \'Num
VDs\': 1, \'Num PDs\': 8, \'RAID health\': \'Opt\'}],
\'bad_disks_results_list\': [], \'rbld_disks_results_list\':
 [], \'add_as_spares_disks_results_list\': []}}"]}',
 'role': 'compute',
 'status': 'diskmgmt_completed',
 'updated_at': '2018-03-07 21:47:35+00:00',
 'uuid': 'd24036c6-4557-4c12-8695-a92f6f9315ed'
 }
```

Delete a completed diskmgmt operation

Verb	URI
DELETE	v1/diskmgmt/delete/?uuid=<uuid>

Example

JSON Request

```
DELETE /v1/diskmgmt/delete/?uuid=d24036c6-4557-4c12-8695-a92f6f9315ed
```

JSON Response

```
200 OK
Content-Type: application/json
{
 "status": "deleted",
 "message": "UUID d24036c6-4557-4c12-8695-a92f6f9315ed deleted from database",
 "uuid": "d24036c6-4557-4c12-8695-a92f6f9315ed",
 "error": "None"
}
```

OSD Maintenance information

REST wrapper to query information about OSD on Pod storage nodes. This returns to the OSD status information of all or a selection of OSDs available in the Pod.

Create a OSD disk operation

Verb	URI
POST	/v1/osdmgmt/check_osds

Example

JSON Request

```
POST /v1/osdmgmt/osdmgmt/check_osds
'{"osdmgmt_request": {"command": "create",
                                          "action": "check-osds",
                                          "locator": "False",
                                      "json_display": "False",
                                          "servers": "",
                                      "osd": "None",
                                          "uuid": ""}}'
```

JSON Response

```
201 Created
Content-Type: application/json
{
 'action': 'check-osds',
 'created_at': '2018-03-08T21:26:15.329195+00:00',
 'osdmgmt_request': "{u'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223',
 u'json_display': u'False', u'servers': u'', u'locator': u'False',
 u'command': u'create', u'action':
 u'check-osds', u'osd': u'None'}",
 'osdmgmt_result': '',
 'status': 'not_run',
 'updated_at': 'None'
}
```

Create a replace OSD operation

Verb	URI
POST	v1/osdmgmt/replace_osd

Example

JSON Request

```
POST /v1/osdmgmt/replace_osd
Accept: application/json
 '{"osdmgmt_request": {"command": "create",
                                           "action": "replace-osd",
                                           "locator": "False",
                                           "json_display": "False",
                                           "servers": "f24-michigan-micro-1",
                                           "osd": "osd.9",
                       "uuid": ""}}'
```

JSON Response

```
201 Created
Content-Type: application/json
{
 "status": "not_run",
 "osdmgmt_request": "{u'uuid': '5140f6fb-dca3-4801-8c44-89b293405310',
 u'json_display': u'False',
 u'servers': u'f24-michigan-micro-1',
 u'locator': u'False',
 u'command': u'create',
 u'action': u'replace-osd',
 u'osd': u'osd.9'}",
 "created_at": "2018-03-09T15:07:10.731220+00:00",
 "updated_at": null,
 "action": "replace-osd",
 "osdmgmt_result": ""
 }
 }
```

List check OSD operation

Verb	URI
GET	v1/osdmgmt/list/?action={check-osds,replace-osd}

Example

JSON Request

```
GET /v1/osdmgmt/list/?action=check-osds
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 '4efd0be8-a76c-4bc3-89ce-142de458d844': {'action': 'check-osds',
                                                                                'created_at': '2018-03-08 21:
31:01+00:00',
                                                                                'status': 'osdmgmt_running',
                                                                                'uuid': '4efd0be8-a76c-4bc3-

89ce-142de458d844'},
 '5fd4f9b5-786a-4a21-a70f-bffac70a3f3f': {'action': 'check-osds',
                                                                                 'created_at': '2018-03-08 21:
11:13+00:00',
                                                                                 'status':
'osdmgmt_completed',
                                                                                'uuid':'5fd4f9b5-786a-4a21-
a70f-bffac70a3f3f'},

 '9c64ee52-bed5-4b69-91a2-d589411dd223': {'action': 'check-osds',
                                                                                 'created_at': '2018-03-08 21:
26:15+00:00',
                                                                                 'status':
'osdmgmt_completed',
                                          'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'}
 }
 }
```

Show a completed osdmgmt operation

Verb	URI

GET	v1/osdmgmt/show/?uuid=<uuid>

Example

JSON Request

```
GET /v1/osdmgmt/show/?uuid=9c64ee52-bed5-4b69-91a2-d589411dd223
```

JSON Response

```
200 OK
Content-Type: application/json
{
 'action': 'check-osds',
 'created_at': '2018-03-08 21:26:15+00:00',

 'osdmgmt_request': "
 {u'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223',
 u'json_display': u'False',
 u'servers': u'',
 u'locator': u'False',
 u'command': u'create', u'action':
 u'check-osds', u'osd': u'None'}",
 'osdmgmt_result': '{"status": "PROCESSED", "message": ["{\'Overall_Status\': \'PASS\',
 \'Result\': { ommitted for doc }}]}',
 'status': 'osdmgmt_completed',
 'updated_at': '2018-03-08 21:27:16+00:00',
 'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
 }
 }
```

Delete a completed osdmgmt operation

Verb	URI
DELETE	v1/osdmgmt/delete/?uuid=<uuid>

Example

JSON Request

```
DELETE /v1/osdmgmt/delete/?uuid=9c64ee52-bed5-4b69-91a2-d589411dd223
```

JSON Response

```
200 OK
Content-Type: application/json
{
 'error': 'None',
 'message': 'UUID 9c64ee52-bed5-4b69-91a2-d589411dd223 deleted from database',
 'status': 'deleted',
 'uuid': '9c64ee52-bed5-4b69-91a2-d589411dd223'
 }

}
```

Cloud Sanity

Cloud Sanity

- Create a cloud-sanity test
- List cloud-sanity test results
- List specific cloud-sanity test results
- Show cloud-sanity test results
- Delete cloud-sanity test results

REST wrapper to run cloud-sanity test suites. The cloud-sanity extension to the VIM REST API enables support for managing cloud-sanity test actions.

Create a cloud-sanity test

Verb	URI
Post	/v1/cloud-sanity/create

Example

JSON Request

```
POST /v1/cloudsanity/create
Accept: application/json
 '{"cloudsanity_request": {"command": "create",
                                                  "action": "test",
                                                  "test_name": "cephmon",
                                                  "uuid": ""}}'
 test_name can be all, management, control, compute, cephmon, cephosd
```

JSON Response

```
201 Created
{
'cloudsanity_request': "{u'action': u'test', u'command': u'create', u'uuid':
'5dff1662-3d33-4901-808d-479927c01dde',
 u'test_name': u'cephmon'}",
'cloudsanity_result': '',
'created_at': '2018-01-26T20:32:20.436445',
'status': 'not_run',
'test_name': 'cephmon',
'updated_at': ''
}
```

List cloud-sanity test results

Verb	URI
GET	/v1/cloud-sanity

JSON Request

```
GET /v1/cloudsanity
```

JSON Response

```
200 OK
{ '0b91746f-90b4-4355-a748-727c2e5c59c5': { 'action': 'test',
                                                
```

GET	/v1/cloud-sanity/list/?test_name={all, management, control,compute,cephmon,cephosd}

JSON Request

```
GET /v1/cloudsanity/list/?test_name=cephmon
Accept: application/json
```

JSON Response

```
200 OK
 { '5dff1662-3d33-4901-808d-479927c01dde': { 'action': 'test',
                                                                                'created_at': '2018-
01-26 20:32:20',
                                                                               'status':
'cloudsanity_completed',
                                                                               'test_name': 'cephmon',
                                                                             'uuid': '5dff1662-3d33-
4901-808d-479927c01dde'},
   '797d79ba-9ee0-4e11-9d9e-47791dd05e07': { 'action': 'test',
                                                                               'created_at': '2018-01-
25 12:05:11',
                                                                               'status':
'cloudsanity_completed',
                                                                               'test_name': 'cephmon',
                                                                                'uuid': '797d79ba-
9ee0-4e11-9d9e-47791dd05e07'}}
```

Show cloud-sanity test results

Verb	URI
GET	/v1/cloud-sanity/show/?uuid=<uuid>

JSON Request

```
GET /v1/cloudsanity/show/?uuid=d0111530-ee3b-45df-994c-a0917fd18e11
```

JSON Response

```
200 OK
 { 'action': 'test',
   'cloudsanity_request':
         "{u'action': u'test',
           u'command': u'create',
           u'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11',
           u'test_name': u'control'}",
   'cloudsanity_result':
          '{"status": "PROCESSED",
                "message": {"status": "Pass",
               "message": "[PASSED] Cloud Sanity Control Checks Passed",
               "results": {"control": {"ping_all_controller_nodes": "PASSED",
                                                                    "check_rabbitmq_is_running": "PASSED",
                                                                   "check_rabbitmq_cluster_status": "PASSED",
                                                                   "check_nova_service_list": "PASSED",
                                                                   "ping_internal_vip": "PASSED",
                                                                   "disk_maintenance_raid_health": "PASSED",
                                                                   "check_mariadb_cluster_size": "PASSED",
                                                                   "disk_maintenance_vd_health": "PASSED"}}}}',
        'created_at': '2018-01-26 18:46:23',
         'status': 'cloudsanity_completed',
        'test_name': 'control',
    'updated_at': '2018-01-26 18:47:58',
        'uuid': 'd0111530-ee3b-45df-994c-a0917fd18e11'}
```

Delete cloud-sanity test results

Verb	URI
DELETE	/v1/cloud-sanity/delete/?uuid=<uuid>

JSON Request

```
GET /v1/cloudsanity/delete/?uuid=444aa4c8-d2ba-4379-b035-0f47c686d1c4
```

JSON Response

```
200 OK
{
 "status": "deleted",
 "message": "UUID 444aa4c8-d2ba-4379-b035-0f47c686d1c4 deleted from database",
 "uuid": "444aa4c8-d2ba-4379-b035-0f47c686d1c4",
 "error": "None"
 }
```

Mandatory/Optional Feature Mapping

Mandatory/Optional Feature Mapping

- Mandatory Feature Mapping
- Optional Feature Mapping

Mandatory Feature Mapping

```
POST Request URL
/v1/releasemapping/mandatory_features_mapping

JSON Response:
{
        "mandatory": {
        "networkType": {
        "C": {
                "feature_status": true,
                "values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],
                "insight_label": "Tenant Network",
                "desc": "Tenant Network"
                },
        "B": {
                "feature_status": true,
                "values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],
                "insight_label": "Tenant Network",
                "desc": "Tenant Network"
                }
},
        "cephMode": {
                "all": {
                        "feature_status": true,
                        "values": [{"name": "Central", "value": "Central"},],
                        "insight_label": "Ceph Mode",
                        "desc": "Ceph Mode"
                                }
                        },
        "podType": {
                "C": {
                        "feature_status": true,
                        "values": [{"name": "Fullon", "value": "fullon"},],
                        "insight_label": "POD Type",
                        "desc": "POD Type"
                        },
                "B": {
                        "feature_status": true,
                        "values": [{"name": "Fullon", "value": "fullon"},],
                        "insight_label": "POD Type",
                        "desc": "POD Type"
                        }
},
        "installMode": {
                        "all": {
                                "feature_status": true,
                                "values": [{"name": "Connected", "value": "connected"}, ],
                                "insight_label": "Install Mode",
                                "desc": "Install Mode"
                                }
        }
},
        "platformType": [{"name": "B-series", "value": "B"}, {"name": "C-series", "value":
                        "C"}],
        "postinstalllinks": {
                                "view_cloudpulse": {"alwayson": true, "feature_status": true, "platformtype":
"all",
                                "insight_label": "Run VMTP", "desc": "Cloudpluse"},
                                "password_reconfigure": {"alwayson": true, "feature_status": true,
"platformtype":
                                "all", "insight_label": "Reconfigure Passwords", "desc": "Reconfigure
Passwords"}
                        }
}
```

Optional Feature Mapping

```
POST Request URL
/v1/releasemapping/optional_features_mapping

JSON Response:
[
{
        "heat": {
                "feature_status": true,
                "insight_label": "Heat",
                "repeated_redeployment": false,
                "reconfigurable": ["all"],
                "desc": "Openstack HEAT service"
                        }
                },
..... other features
]
```

Testing and Polling

Testing and Polling

- NFVBench Network Performance Testing
  - Create NFVBench Run
- Status Polling
  - Get fixed rate test result
  - Execute NDR/PDR test
  - Get NDR/PDR test results

NFVBench Network Performance Testing

Create NFVBench Run

Starts the network performance test with provided configuration.

REST API to create fixed rate test

Verb	URI
Post	v1/nfvbench/ create_ndr_pdr_test

Example

JSON Request

```
POST Request URL
 /v1/nfvbench/create_fixed_rate_test JSON Request:
 {"nfvbench_request":
         {
         "duration_sec": 20,
         "traffic_profile": [
                 {
                   "name": "custom",
                   "l2frame_size": ["64","IMIX", "1518"]
                   }
         ],
         "traffic": { "bidirectional": true,
                          "profile": "custom"
                                 },
         "flow_count": 1000
 }
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
        "status": "not_run",
        "nfvbench_request":
  '{
        "duration_sec": 20,
        "traffic_profile": [
                {
                "name": "custom",
                "l2frame_size": ["64","IMIX","1518"]
                }
        ],
        "traffic": {"bidirectional": true,
                                "profile": "custom"
                                },
        "flow_count": 1000
        }',
"created_at": "2017-08-16T06:14:54.219106",
"updated_at": null,
"nfvbench_result": "",
"test_name": "Fixed_Rate_Test"
}
```

Status Polling

The polling status of NFVbench status can be nfvbench_running, nfvbench_failed, or nfvbench_completed.

Resource URI

Verb	URI
GET	v1/nfvbench/<test_name>

Get fixed rate test result

```
GET Request URL
/v1/upgrade/get_fixed_rate_test_result
JSON Response:
Check If NFVbench Test is running
200 OK
Content-Type: application/json
{
 "status": "nfvbench_running",
 "nfvbench_request": '{"traffic": {"bidirectional": true, "profile": "custom"},
 "rate": "1000000pps",
 "traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}], "duration_sec": 60, "flow_count": 1000}',
 "nfvbench_result": ""
 "created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-30T21:40:41.367279",
 }
Check If NFVbench Test is completed
200 OK
Content-Type: application/json
{
 "status": "nfvbench_completed",
 "nfvbench_request": '{"traffic": {"bidirectional": true, "profile": "custom"}, "rate": "1000000pps",
 "traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}], "duration_sec": 60, "flow_count": 1000}',
 "nfvbench_result": '{"status": "PROCESSED", "message": {"date": "2017-08-15 23:15:04", "nfvbench_version":
"0.9.3.dev2", ....}
 "created_at": "2017-05-30T21:40:40.394274", "updated_at": "2017-05-30T22:29:56.970779",
 }
```

Execute NDR/PDR test

```
POST Request URL
/v1/nfvbench/create_ndr_pdr_test

Accept: application/json
{"nfvbench_request":
{
        "duration_sec": 20,
        "traffic_profile": [
         {
                "name": "custom",
                "l2frame_size": ["64","IMIX","1518"]
                }
        ],
        "traffic": {"bidirectional": true, "profile": "custom"}, "flow_count": 1000}}

JSON Response
201 CREATED
Content-Type: application/json
{
        "status": "not_run",
        "nfvbench_request":
        '{
                "duration_sec": 20,
                "traffic_profile": [{"name": "custom",
        "l2frame_size": ["64","IMIX","1518"]}],
        "traffic": {"bidirectional": true, "profile": "custom"}, "flow_count": 1000}'
        "created_at": "2017-08-16T07:18:41.652891",
        "updated_at": null,
        "nfvbench_result": "",
        "test_name": "NDR_PDR_Test"
}
```

Get NDR/PDR test results

```
GET Request URL
/v1/ nfvbench/get_ndr_pdr_test_result
JSON Response:
If NFVbench NDR/PDR test is running
200 OK
Content-Type: application/json
{
        "status": "nfvbench_running",
        "nfvbench_request": '{"duration_sec": 20,
        "traffic": {"bidirectional": true, "profile": "custom"},
        "traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name": "custom"}],
        "flow_count": 1000}',
        "nfvbench_result": ""
        "created_at": "2017-08-16T07:18:41.652891",
        "updated_at": "2017-09-30T22:29:56.970779",
}
If NFVbench NDR/PDR test is completed
200 OK
Content-Type: application/json
{
        "status": "nfvbench_completed",
        "nfvbench_request": '{"duration_sec": 20,
        "traffic": {"bidirectional": true, "profile": "custom"},
        "traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name": "custom"}], "flow_count":1000}',
        "nfvbench_result": '{"status": "PROCESSED",...}'
        "created_at": "2017-08-16T07:18:41.652891",
        "updated_at": "2017-09-30T22:29:56.970779",
}
```

Post-Installation Operations

Post-Installation Operations

- Create a post install operation
- Retrieve post install operation status

The following are the post install operations that can be performed, after the successful installation of OpenStack. It uses a common api. Following is an Example:

1. reconfigure
2. reconfigure -regenerate passwords
3. reconfigure -setpasswords,setopenstack_configs
4. reconfigure -alertmanager_config, -alerting_rules_config
5. check-fernet-keys
6. resync-fernet-keys
7. rotate-fernet-keys

Create a post install operation

Resource URI

Verb	URI
POST	/v1/misc

Examples:

JSON Request

```
POST /v1/misc
 Accept: application/json
 {"action": {"reconfigure": true}}
```

JSON Response

```
201 CREATED
 Content-Type: application/json
 {
        "uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733",
        "created_at": "2017-03-19T14:03:39.723914",
         "updated_at": null,
         "operation_status": "OperationScheduled",
        "operation_logs": "",
    "operation_name": "{"reconfigure": true}"
 }
```

JSON Request

```
POST /v1/misc
 Accept: application/json
 {"action": {"reconfigure": true, "alertmanager_config": <json_config>}}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
 "uuid": "68b67265-8f09-480e-8608-b8aff77e0ec7",
 "created_at": "2019-01-09T16:42:11.484604+00:00",
 "updated_at": null,
 "operation_status": "OperationScheduled",
 "operation_logs": "",
 "operation_name": "{"alertmanager_config": <json_config>, "reconfigure": true}"
 }
```

Retrieve post install operation status

Resource URI

Verb	URI
GET	/v1/misc

Example

JSON Request

```
GET /v1/misc
 Accept: application/json
```

JSON Response

```
201 CREATED
 Content-Type: application/json
 {
        "uuid":
        "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733",
        "created_at": "2017-03-19T14:03:39.723914",
        "updated_at": "2017-03-19T14:03:42.181180",
        "operation_status": "OperationRunning",
        "operation_logs": "xxxxxxxxxxxxxxxx",
        "operation_name": "{\"reconfigure\": true}"
 }
```

Version and Hardware Information

Version and Hardware Information

- Version
- Hardware Information
    - Create a HWinfo Operation
    - Retrieve Hwinfo Operation Results
    - Get Node Hardware Information

Version

Retrieve the version of the Cisco VIM.

Resource URI

Verb	URI
GET	/v1/version

Example

JSON Request

```
GET /v1/version
 Accept: application/json
```

JSON Response

```
200 OK
 Content-Type: application/json
 {"version": "1.9.1"}
```

Hardware Information

REST wrapper returns the hardware information available in the setupdata.

Create a HWinfo Operation

Resource URI

Verb	URI
GET	/v1/hwinfo

Example

JSON Request

```
POST /v1/hwinfo
 Accept: application/json
 {
 "setupdata":"c94d7973-2fcc-4cd1-832d-453d66e6b3bf"
 }
```

JSON Response

```
201 CREATED
 Content-Type: application/json
 {
 "status": "hwinfoscheduled",
 "uuid": "928216dd-9828-407b-9739-8a7162bd0676",
 "setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at": "2017-03-19T13:41:25.488524",
 "updated_at": null, "hwinforesult": ""
 }
```

Retrieve Hwinfo Operation Results

Resource URI

Verb	URI
GET	/v1/hwinfo/{id}

Property:
id—The ID of the node you want to query. Example

JSON Request

```
GET /v1/hwinfo/789 Accept: application/json
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 "status": "hwinfosuccess",
 "uuid": "928216dd-9828-407b-9739-8a7162bd0676",

 "setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf", "created_at":
"2017-03-19T13:41:25.488524", "updated_at":
"2017-03-19T13:42:05.087491",
 "hwinforesult": "{\"172.29.172.73\": {\"firmware\": ..............
 ............
 ..................
 }
```

Get Node Hardware Information

Rest API helps you to get the hardware information of all the nodes in the pod through CIMC/UCSM.

- Total Memory
- Firmware Info (Model, Serial Number)
- CIMC IP

```
GET Request URL
/v1/hwinfo
Output Response
{
        "hwinforesult": "{"control-server-2": {"memory": {"total_memory": "131072"},
        "firmware": {"serial_number": "FCH1905V16Q", "fw_model": "UCSC-C220-M4S"},
        "cimc_ip": "172.31.230.100", "storage": {"num_storage": 4},
        "cisco_vic_adapters": {"product_name": "UCS VIC 1225"},
        "cpu": {"number_of_cores": "24"}, "power_supply": {"power_state": "on"}}
...
}
```

OpenStack Setup

OpenStack Setup

- Secrets
  - Retrieve the list of secrets that are associated with the OpenStack Setup
- OpenStack Configuration
  - Retrieve the list of configurations associated with the OpenStack Setup
  - Release mapping Information

Secrets

Retrieve the list of secrets that are associated with the OpenStack Setup

You can retrieve the set of secret password that are associated with the OpenStack setup using the preceding api. This gives the list of secrets for each service in OpenStack.
Resource URI

Verb	URI
GET	/v1/secrets

Example

JSON Request

```
GET /v1/secrets
 Accept: application/json
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 "HEAT_KEYSTONE_PASSWORD": "xxxx", "CINDER_KEYSTONE_PASSWORD": "xxxxx",
 ....
 ....
 "RABBITMQ_PASSWORD": "xxxxx"
 }
```

OpenStack Configuration

Retrieve the list of configurations associated with the OpenStack Setup

You can retrieve the set of OpenStack configurations associated with the OpenStack setup using the preceding api. This gives the current settings of different configurations such as verbose logging and debug logging for different OpenStack services.

Verb	URI
GET	/v1/secrets

JSON Request

```
GET /v1/openstack_config
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "CINDER_DEBUG_LOGGING": false,
        "KEYSTONE_DEBUG_LOGGING": false,
        ....
        ....
        "NOVA_VERBOSE_LOGGING": true
}
```

Release mapping Information

This api is used to see the list of features included and list of options which can be reconfigured in the Openstack Setup.

Retrieve the release mapping information

Resource URI

Verb	URI
GET	/v1/releasemapping

JSON Request

```
GET /v1/releasemapping
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
[
   {
        "SWIFTSTACK": {
                "feature_status": true,
                 ],
        "desc": "swift stack feature"
    }
   },........
..............
}
```

Update

Update

- Start an Update Process
- Roll back an update
- Commit an update
- Retrieve the details of an update

Start an Update Process

Resource URI

Verb	URI
POST	/v1/update

Parameters:

- fileupload - tar file to be uploaded.
- filename - Name of the uploaded file.

Example

JSON Request

```
curl -sS -X POST --form
"fileupload=@Test/installer.good.tgz" --form
"filename=installer.good.tgz"
https://10.10.10.8445/v1/update
```

⚠ This curl request is done as a form request.

JSON Response

```
200 OK
 Content-Type: application/json
 {
        "update_logs": "logurl",
         "update_status": "UpdateSuccess",
        "update_filename": "installer-4579.tgz",
        "created_at": "2016-07-10T18:33:52.698656",
        "updated_at": "2016-07-10T18:54:56.885083"
 }
 409 CONFLICT
 Content-Type: application/json
 {
 "debuginfo": null "faultcode": "Client"
 "faultstring": "Uploaded file is not in tar format"
 }
```

Roll back an update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
PUT /v1/update
 Accept: application/json
 {
 "action":"rollback"
 }
```

JSON Response

```
200 OK
 Content-Type: application/json
 {

 "update_logs": "logurl", "update_status": "ToRollback", "update_filename": "installer-4579.tgz",
 "created_at": "2016-07-10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
 }
```

Commit an update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
PUT /v1/update
Accept: application/json
{
 "action":"commit"
}
```

JSON Response

```
200 OK
Content-Type: application/json
{
 "update_logs": "logurl", "update_status": "ToCommit", "update_filename": "installer-4579.tgz",
 "created_at": "2016-07-10T18:33:52.698656", "updated_at": "2016-07-10T18:54:56.885083"
}
```

Retrieve the details of an update

Resource URI

Verb	URI
GET	/v1/update

Example

JSON Request

```
GET /v1/update
 Accept: application/json
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
         "update_logs": "logurl",
        "update_status": "UpdateSuccess",
        "update_filename": "installer-4579.tgz",
    "created_at": "2016-07-10T18:33:52.698656",
        "updated_at": "2016-07-10T18:54:56.885083"
 }
```

Install Resource

Install Resource

- Return a list of installation
- Create an installation
- Retrieve the installation
- Stop the installation

REST wrapper for install. Provides methods for starting, stopping, and viewing the status of the installation process.

Return a list of installation

Resource URI

Verb	URI
GET	/v1/install

Example

JSON Request

```
GET /v1/install
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
  {"installs": [{
        "ceph": "Skipped",
        "uuid": "123",
        "setupdata": "345",
        "vmtpresult": "{
        "status": "PASS",
        "EXT_NET": []
  }",
        "baremetal": "Success",
        "orchestration": "Success",
        "validationstatus": "{
        "status": "PASS",
        "Software_Validation": [],
        "Hardware_Validation": []
  }",
    "currentstatus": "Completed",
        "validation": "Success",
        "hostsetup": "Success",
        "vmtp": "Skipped"
  }]
}
```

Create an installation

Resource URI

Verb	URI
POST	/v1/install

Example

JSON Request

```
GET /v1/install
Accept: application/js
{
        "setupdata": "123",
        "stages": [
        "validation",
        "bootstrap",
        "runtimevalidation",
        "baremetal",
        "orchestration",
        "hostsetup",
        "ceph",
        "vmtp"
        ]
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
        "ceph": "Skipped",
        "uuid": "123",
        "setupdata": "345",
        "vmtpresult": "{
        "status": "PASS",
        "EXT_NET": []
}",
        "baremetal": "Success",
        "orchestration": "Success",
        "validationstatus": "{
        "status": "PASS",
        "Software_Validation": [],
        "Hardware_Validation": []
}",
        "currentstatus": "Completed",
        "validation": "Success",
        "hostsetup": "Success",
        "vmtp": "Skipped"
}
409 CONFLICT
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Install already exists"
}
```

Retrieve the installation

Resource URI

Verb	URI
GET	/v1/install/{id}

Property:
id—The ID of the installation that you want to retrieve.

Example

JSON Request

```
GET /v1/install/345
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "ceph": "Skipped",
        "uuid": "123",
        "setupdata": "345",
        "vmtpresult": "{
        "status": "PASS",
        "EXT_NET": []
}",
        "baremetal": "Success",
        "orchestration": "Success",
        "validationstatus": "{
        "status": "PASS",
        "Software_Validation": [],
        "Hardware_Validation": []
}",
        "currentstatus": "Completed",
        "validation": "Success",
        "hostsetup": "Success",
        "vmtp": "Skipped"
}
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Install doesn't exists"
}
```

Stop the installation

Resource URI

Verb	URI
DELETE	/v1/install/{id}

Property:
id—The ID of the installation that you want to stop. Example

JSON Request

```
DELETE /v1/install/345
Accept: application/js
```

JSON Response

```
204 NO CONTENT
 Content-Type: application/json

404 NOT FOUND
 Content-Type: application/json
 {
        "debuginfo": null "faultcode": "Client"
        "faultstring": "Install doesn't exists"
 }
```

Nodes and Replace Controller

Nodes and Replace Controller

- Nodes
  - Getting a list of nodes
  - Add New Nodes
  - Retrieve information about a particular node
  - Remove a Node
  - Health of the Management Node
- Replace a controller

Nodes

Getting a list of nodes

Resource URI

Verb	URI
GET	/v1/nodes

Example

JSON Request

```
Get /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "nodes": [
        [
                "status": "Active",
                "uuid": "456",
                "setupdata": "123",
                "node_data": "{
                "rack_info": {
                "rack_id": "RackA"
                },
                "cimc_info": {
                  "cimc_ip": "10.10.10.10"
                },
        "management_ip": "7.7.7.10"
          }",
        "updated_at": null,
        "mtype": "compute",
        "install": "345",
        "install_logs": "logurl",
        "created_at":"2016-0710T06:17:03.761152",
        "name": " compute-1"
        }
   ]
}
```

Add New Nodes

The nodes are in compute or block_storage type. Before adding the nodes to the system, the name of the nodes and other necessary information like cimc_ip and rackid must be updated in the setupdata object. If the setupdata object is not updated, the post call does not allow you to add the node.
Resource URI

Verb	URI

POST	/v1/nodes

Example

JSON Request

```
POST /v1/nodes
Accept: application/js
 {
 "name" : "compute-5"
 }
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
        "status": "ToAdd",
        "uuid": "456",
        "setupdata": "123",
        "node_data": "{
        "rack_info": {
        "rack_id": "RackA"
        },
        "cimc_info": {
        "cimc_ip": "10.10.10.10"
        },
        "management_ip": "7.7.7.10"
        }",
        "updated_at": null,
        "mtype": "compute",
        "install": "345",
        "install_logs": "logurl",
        "created_at":"2016-0710T06:17:03.761152",
        "name": " compute-1"
}
```

Retrieve information about a particular node

Resource URI

Verb	URI
GET	/v1/nodes{id}

Property:
id—The ID of the node that you want to retrieve. Example

JSON Request

```
POST /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "status": "Active",
        "uuid": "456",
        "setupdata": "123",
        "node_data": "{
        "rack_info": {
        "rack_id": "RackA"
        },
        "cimc_info": {
        "cimc_ip": "10.10.10.10"
        },
        "management_ip": "7.7.7.10"
        }",
        "updated_at": null,
        "mtype": "compute",
        "install": "345",
        "install_logs": "logurl",
        "created_at":"2016-0710T06:17:03.761152",
        "name": " compute-1"
}
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Node doesn't exists"
}
```

Remove a Node

The node to be deleted must be removed from the setupdata object. Once the setupdata object is updated, you can safely delete the node. The node object cannot be deleted until it calls the remove node backend and succeeds.
Resource URI

Verb	URI
DELETE	/v1/nodes{id}

Property:
id—The ID of the node that you want to remove. Example

JSON Request

```
DELETE /v1/nodes/456
Accept: application/js
```

JSON Response

```
204 ACCEPTED
Content-Type: application/json
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Node doesn't exists"
}
```

To clear the database and delete the entries in the nodes, the delete API is called with special parameters that are passed along with the delete request. The JSON parameters are in the following format.

JSON Request

```
DELETE /v1/nodes/456
Accept: application/js
 {
 "clear_db_entry":"True"\
 }
```

JSON Response

```
204 ACCEPTED
 Content-Type: application/json

 404 NOT FOUND
 Content-Type: application/json
 {
         "debuginfo": null
         "faultcode": "Client"
         "faultstring": "Node doesn't exists"
 }
```


This is done only if the node is deleted from the REST API database. The failure reason of the node must be rectified manually apart from the API. True is a string and not a boolean in the preceding line.

Health of the Management Node

This API is used to retrieve the health of the management node. It checks various parameters such as partitions, space and so on.
Resource URI

Verb	URI
GET	/v1/health

Example

JSON Request

```
GET /v1/health
 Accept: application/json
```

JSON Response

```
200 OK
 Content-Type: application/json
 {
 "status": "PASS", "pod_status": { "color": "BLUE",
 "version": "<VERSION_NO.>"
 },
 "insight_version": "<VERSION_NO.>"
 }
```

Color signifies the health of the pod for Insight:

- Grey signifies that no installation is kicked off on the pod.
- Green signifies that everything is in Good state and cloud installation is active.
- Blue signifies that some operation is running on the pod.
- Red signifies that the pod is in critical state and you might need TAC support to recover the pod.
- Amber indicates a warning if a pod management (Add/Remove/Replace) operation failed.

Replace a controller

Resource URI

Verb	URI
PUT	/v1/nodes{id}

Property:
id—The ID of the controller that you want to replace. Example

JSON Request

```
PUT /v1/nodes/456 Accept: application/js
```

JSON Response

```
200 OK
 Content-Type: application/json
 404 NOT FOUND
 Content-Type: application/json
 {
        "debuginfo": null
        "faultcode": "Client"
         "faultstring": "Node doesn't exists"
 }
```

Setupdata and Offline Validation

Setupdata and Offline Validation

- Setupdata
  - Retrieving the setupdata
  - Creating setupdata
  - Retrieving a single setupdata
  - Updating a setupdata
  - Deleting a setupdata
- Offline validation
  - Create an offline validation operation
  - Retrieve the results of offline validation

Setupdata

REST wrapper for setupdata. Provides methods for listing, creating, modifying, and deleting setupdata.

Retrieving the setupdata

Resource URI

Verb	URI
GET	/v1/setupdata

Example

JSON Request

```
GET /v1/setupdata
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"setupdatas": [{
"status": "Active",
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}]}
```

Creating setupdata

Resource URI

Verb	URI
POST	/v1/setupdata

Example

JSON Request

```
POST /v1/setupdata
Accept: application/json
{
"name":"GG34",
"uuid": "123"
"meta":{
"user":"root"
},
"jsondata":{
.......
}
}
```

JSON Response

```
201 OK
Content-Type: application/json
{
        "status": "Active",
        "name":"GG34",
        "uuid": "123"
         "meta":{
         "user":"root"
},
"jsondata":{
.......
 }
}

400 Bad Request
Content-Type: application/json
{
       "debuginfo": null
       "faultcode":"Client"
       "faultstring": "Error"
}
409 CONFLICT
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Error"
}
```

Retrieving a single setupdata

Resource URI

Verb	URI
GET	/v1/setupdata/(id)

Property:
id—The ID of the setupdata that you want to retrieve.

Example

JSON Request

```
GET /v1/setupdata/123
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
        "status": "Active",
        "name":"GG34",
        "uuid": "123"
        "meta":{
        "user":"root"
},
"jsondata":{
.......
        }
}
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
"faultcode": "Client"
"faultstring": "Setupdata could not be found."
}
```

Updating a setupdata

Resource URI

Verb	URI
PUT	/v1/setupdata/(id)

Property:
id—The ID of the setupdata that you want to update. Example

JSON Request

```
PUT /v1/setupdata/123 Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
 "status": "Active",
 "name":"GG34",
 "uuid": "123"
 "meta":{
 "user":"root"
  },
 "jsondata":{
 .......
 }
}
404 NOT FOUND
Content-Type: application/json
{
 "debuginfo": null
 "faultcode": "Client"
 "faultstring": "Setupdata could not be found."
}
```

Deleting a setupdata

Resource URI

Verb	URI

DELETE	/v1/setupdata/(id)

Property:
id—The ID of the setupdata that you want to delete. Example

JSON Request

```
DELETE /v1/setupdata/123 Accept: application/json
```

JSON Response

```
204 NO CONTENT Returned on success
404 NOT FOUND
Content-Type: application/json
{
        "debuginfo": null
        "faultcode": "Client"
        "faultstring": "Setupdata could not be found."
}
400 BAD REQUEST
Content-Type: application/json
{
"debuginfo": null "faultcode": "Client"
"faultstring": "Setupdata cannot be deleted when it is being used by an installation"
}
```

Offline validation

REST wrapper does the offline validation of setupdata. Rest wrapper does only the software Validation of the input setupdata.

Create an offline validation operation

Resource URI

Verb	URI
POST	/v1/offlinevalidation

Example

JSON Request

```
POST /v1/offlinevalidation
Accept: application/json
 {
 "jsondata": "."
 }
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
"status": "NotValidated",
"uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69",
"created_at": "2016-02-03T02:05:28.384274",
"updated_at": "2016-02-03T02:05:51.880785",
"jsondata": "{}",
"validationstatus": {
"status": "PASS",
"Software_Validation": [],
"Hardware_Validation": []
}
}
```

Retrieve the results of offline validation

Resource URI

Verb	URI
GET	/v1/offlinevalidation

Property:
id—The ID of the node you want to retrieve. Example

JSON Request

```
GET /v1/offlinevalidation/789
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"status": " ValidationSuccess",
"uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69",
"created_at": "2016-02-03T02:05:28.384274",
"updated_at": "2016-02-03T02:05:51.880785",
"jsondata": "{}",
"validationstatus": {
"status": "PASS",
"Software_Validation": [],
"Hardware_Validation": []
}
}
```

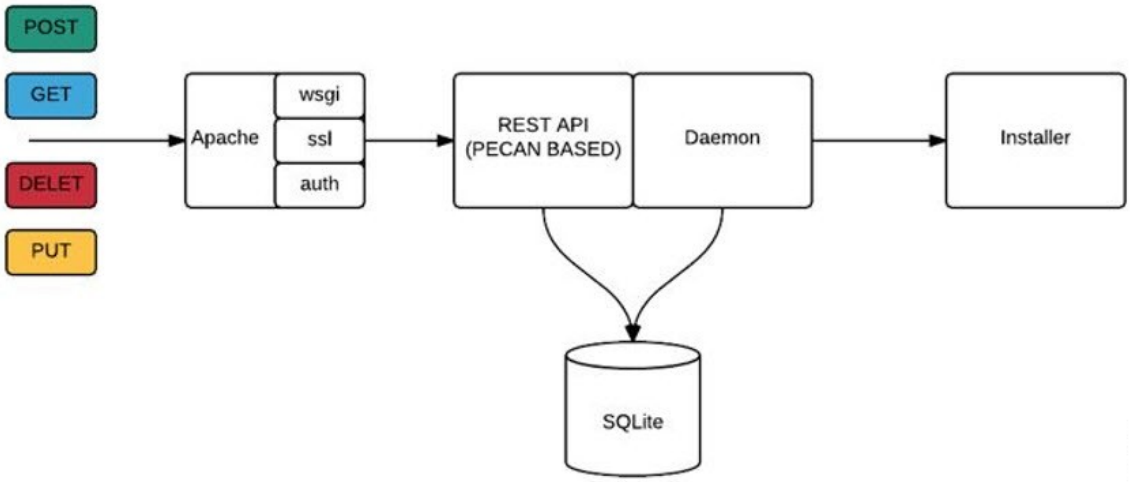
REST API Overview

REST API Overview

A Representational State Transfer (REST) API is used to install, expand, and update Cisco VIM. Actions performed using the REST APIs are:

- Install Cisco VIM on Cisco NFVI pods.
- Add and delete pods to and from Cisco NFVI during installation.
- Update Cisco VIM software.
- Replace controller nodes.
- Perform cloud maintenance operations.
- Run cloud validations using Virtual Machine ThroughPut (VMTP). VMTP is a data path performance measurement tool for OpenStack clouds.

The following figure shows the workflow of Cisco VIM REST API.



The Cisco VIM REST API security is provided by the Secure Sockets Layer (SSL) on the Apache web server. The *mod_wsgi* running on the Rest API server calls the Pecan-based web application. The Pecan REST API server requires a username and password to authorize the REST API server requests.

Apache handles the authorization of the request to access the Pecan web application.

Use the Cisco VIM API to:

- Upload a new setup_data.yaml file to start, stop, or query the state of the installation.
- Manage the cloud.
- Add/remove compute and Ceph nodes, and replace the controller nodes.
- Launch VMTP (L2/L3 data plane testing) and CloudPulse.

The Cisco VIM REST API is enabled by default in the management node, if you are using the supplied Cisco VIM buildnode.iso. You can access API server on the br_api interface on port 8445. The authentication is enabled by default in the web service.

You can access the API end points using the following URL format:

```
https://<Management_node_api_ip>:8445
```

By default, the basic authentication is enabled for the API endpoints in the management node. You can find the authentication credentials in the following file in the management node:

```
/opt/cisco/ui_config.json
```

The following code shows a sample ui_config.json file.

```
{
 "Kibana-Url": "http://10.10.10.10:5601",
 "RestAPI-Url": "https:// 10.10.10.10:8445",
 "RestAPI-Username": "admin",
 "RestAPI-Password": "a96e86ccb28d92ceb1df",
 "RestDB-Password": "e32de2263336446e0f57",
 "BuildNodeIP": "10.10.10.10"
}
```

Monitoring Performance

Monitoring Cisco VIM Performance

The following topics tell you how to display logs to monitor Cisco VIM performance.

- Infrastructure Log Management
- Displaying Log Files
- Kibana Dashboard Login
- Rotation of Logs
- Elasticsearch
- CVIM-MON
- Network Performance Test

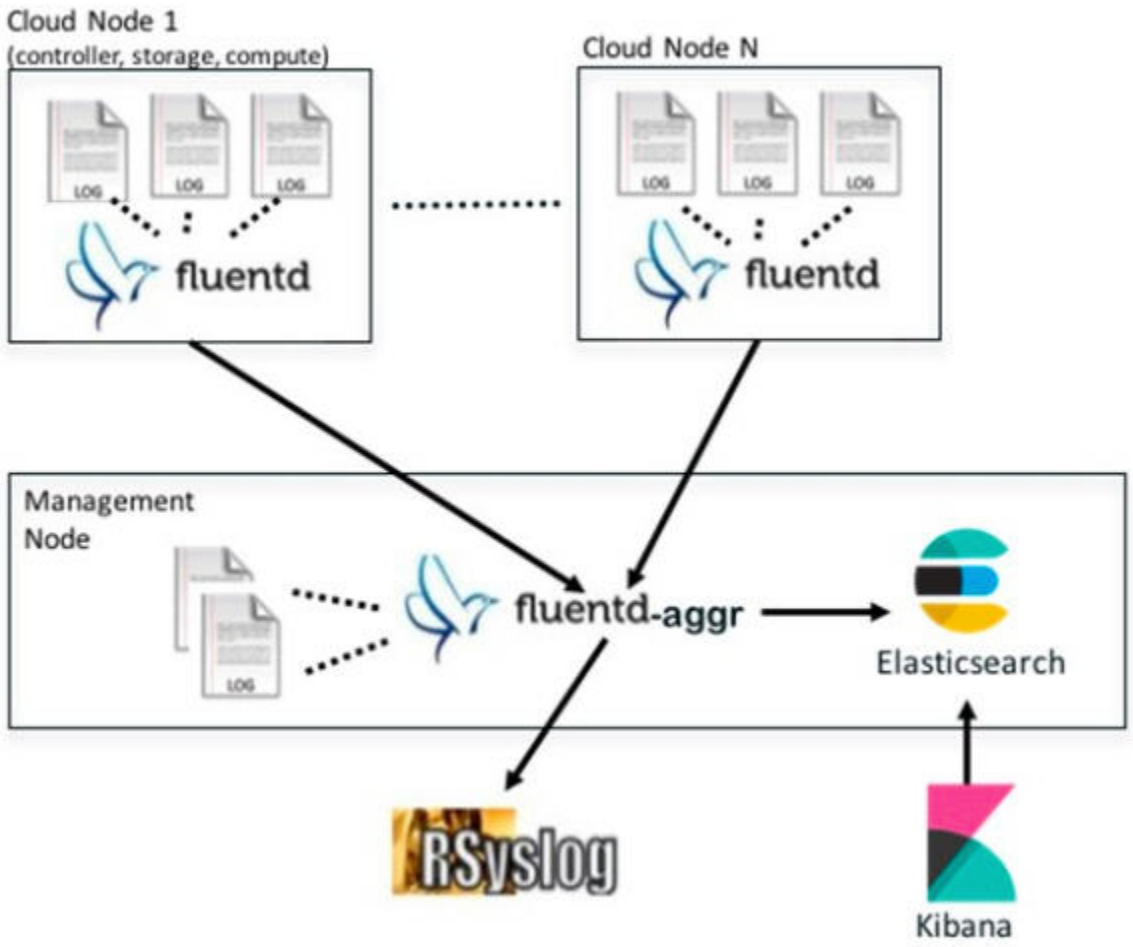
Infrastructure Log Management

Infrastructure Log Management

Cisco VIM uses a combination of open source tools to collect and monitor the Cisco OpenStack services including Elasticsearch, Fluentd, and the Kibana dashboard (EFK). Cisco VIM uses Fluentd instead of logstash. However, to maintain backward compatibility, the code and documentation refers to ELK, instead of EFK at various places. Hence, both ELK and EFK are interchangeably used. OpenStack services followed by EFK include:

- MariaDB: A relational database management system which is based on MySQL. All the OpenStack components store their data in MariaDB.
- HAProxy: HAProxy is a free open source software that provides a high-availability load balancer, and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers.
- Keystone: Keystone is an OpenStack project that provides identity, token, catalog, and policy services for specific use by projects in the OpenStack.
- Glance: An OpenStack project that allows you to upload and discover data assets that are meant for use with other services.
- Neutron: An OpenStack project that provides the network connectivity between interface devices such as vNICs managed by other OpenStack services, for example, Nova.
- Nova: An OpenStack project that is designed to provide on demand massively scalable self-service access to compute resources.
- HTTP: An Apache HTTP server project that is used to develop and maintain an open-source HTTP server.
- Cinder: An OpenStack block storage service that is designed to present storage resources to the users that are consumed by the OpenStack compute project (Nova).
- Memcached: A general-purpose distributed memory caching system.
- CloudPulse: An OpenStack tool that checks the health of the cloud. CloudPulse includes operator and end-point tests.
- Heat: It is the main OpenStack Orchestration program. Heat implements an orchestration engine, to launch multiple composite cloud applications that is based on text file templates.
- Other OpenStack services: RabbitMQ, Ceph, Open vSwitch, and Linux bridge (optional).

A Fluentd container resides on each control, compute, and storage nodes and forwards the logs to the Fluentd-aggr server residing on the management node. The following figure shows a high-level schematic view of the EFK service assurance architecture.



The EFK flow includes:

- Fluentd extracts the relevant data from the logs and tags them, so that Kibana can use it later to display useful information about those logs.
- Fluentd sends the logs from all the compute, controller, and storage nodes to the Fluentd-aggr server on the management node.
- Fluentd-aggr in the management node sends the structured logs into the Elasticsearch database.
- Elasticsearch stores the data, indexes it, and supports fast queries against a large amount of log data.
- Kibana visualizes the data that is stored in Elasticsearch using a custom dashboard. You can also add filters to the data to visualize interesting fragments of the log data.

Displaying Log Files

Displaying Cisco VIM Log Files Using CLI

- Cisco VIM RestAPI log location
- EFK log location
- Viewing Cisco VIM Logs
- Cisco VIM Configuration Files
- Enabling debug logs for certain OpenStack Services

Cisco VIM log file location depends on the node and log type. Installer logs for the last 20 directories are tarred and kept in the management node under the */var/log/mercury/<install_uuid>/* directory. These files contain logs related to bootstrap, build orchestration, baremetal, common setup, and OpenStack orchestration.

If the installer fails, look at the last tar.gz file for logs, for example:

```
[root@mgmtnode mercury]# ls -lrt
total 20
drwxr-xr-x. 2 root root 80 Jul 19 23:42 573f2b7f-4463-4bfa-b57f-98a4a769aced
drwxr-xr-x. 2 root root 4096 Jul 20 03:29 installer
drwxr-xr-x. 2 root root 79 Jul 20 03:29 e9117bc5-544c-4bda-98d5-65bffa56a18f
drwxr-xr-x. 2 root root 79 Jul 20 04:54 36cdf8b5-7a35-4e7e-bb79-0cfb1987f550
drwxr-xr-x. 2 root root 79 Jul 20 04:55 bd739014-fdf1-494e-adc0-98b1fba510bc
drwxr-xr-x. 2 root root 79 Jul 20 04:55 e91c4a6c-ae92-4fef-8f7c-cafa9f5dc1a3
drwxr-xr-x. 2 root root 79 Jul 20 04:58 1962b2ba-ff15-47a6-b292-25b7fb84cd28
drwxr-xr-x. 2 root root 79 Jul 20 04:59 d881d453-f6a0-448e-8873-a7c51d8cc442
drwxr-xr-x. 2 root root 78 Jul 20 05:04 187a15b6-d425-46a8-a4a2-e78b65e008b6
drwxr-xr-x. 2 root root 4096 Jul 20 06:47 d0346cdd-5af6-4058-be86-1330f7ae09d1
drwxr-xr-x. 2 root root 79 Jul 20 17:09 f85c8c6c-32c9-44a8-b649-b63fdb11a79a
drwxr-xr-x. 2 root root 67 Jul 20 18:09 179ed182-17e4-4f1f-a44d-a3b6c16cf323
drwxr-xr-x. 2 root root 68 Jul 20 18:13 426cb05f-b1ee-43ce-862d-5bb4049cc957
drwxr-xr-x. 2 root root 68 Jul 20 18:13 1d2eec9d-f4d8-4325-9eb1-7d96d23e30fc
drwxr-xr-x. 2 root root 68 Jul 20 18:13 02f62a2f-3f59-46a7-9f5f-1656b8721512
drwxr-xr-x. 2 root root 68 Jul 20 18:14 c7417be9-473e-49da-b6d0-d1ab8fb4b1fc
drwxr-xr-x. 2 root root 68 Jul 20 18:17 b4d2077b-c7a9-46e7-9d39-d1281fba9baf
drwxr-xr-x. 2 root root 68 Jul 20 18:35 21972890-3d45-4642-b41d-c5fadfeba21a
drwxr-xr-x. 2 root root 80 Jul 20 19:17 d8b1b54c-7fc1-4ea6-83a5-0e56ff3b67a8
drwxr-xr-x. 2 root root 80 Jul 20 19:17 23a3cc35-4392-40bf-91e6-65c62d973753
drwxr-xr-x. 2 root root 80 Jul 20 19:17 7e831ef9-c932-4b89-8c81-33a45ad82b89
drwxr-xr-x. 2 root root 80 Jul 20 19:18 49ea0917-f9f4-4f5d-82d9-b86570a02dad
drwxr-xr-x. 2 root root 80 Jul 20 19:18 21589a61-5893-4e30-a70e-55ad0dc2e93f
drwxr-xr-x. 2 root root 80 Jul 20 19:22 6ae6d136-7f87-4fc8-92b8-64cd542495bf
drwxr-xr-x. 2 root root 4096 Jul 20 19:46 1c6f4547-c57d-4dcc-a405-ec509306ee25
drwxr-xr-x. 2 root root 68 Jul 20 21:20 c6dcc98d-b45b-4904-a217-d25001275c85
drwxr-xr-x. 2 root root 68 Jul 20 21:40 ee58d5d6-8b61-4431-9f7f-8cab2c331637
drwxr-xr-x. 2 root root 4096 Jul 20 22:06 243cb0f8-5169-430d-a5d8-48008a00d5c7
drwxr-xr-x. 2 root root 4096 Jul 20 22:16 188d53da-f129-46d9-87b7-c876b1aea70c
```

Cisco VIM autobackup logs are found in the following location:

```
# CVIM autobackup logs (auto-backup enabled by default)
/var/log/mercury/autobackup_3.2.x_2019-03-19_15-11-10.log

# cobbler apache log (may be needed for PXE troubleshooting)
/var/log/cobblerhttpd/access_log
/var/log/cobblerhttpd/error_log

# VMTP logs
/var/log/vmtp/vmtp.log
```

Cisco VIM RestAPI log location

```
# CVIM RestAPI logs
/var/log/mercury_restapi/restapi.log

# CIM RestAPI apache logs (TCP port 8445)
/var/log/httpd/mercury_access.log
/var/log/httpd/mercury_error.log

# CIM RestAPI log-directory logs (TCP port 8008)
/var/log/httpd/access_log
/var/log/httpd/error_log
```

EFK log location

```
# Elasticsearch-fluentd-Kibana
/var/log/elasticsearch/
/var/log/fluentd-aggr/
/var/log/kibana/
/var/log/curator/
# HAProxy TLS certificate expiration check
/var/log/curator/certchecker.lo
```

Viewing Cisco VIM Logs

```
# list logs sorted reverse on time
ls -lrt /var/log/mercury/
# untar logs
tar xvzf /var/log/mercury/<UUID>/mercury_install_2018-3-20_10-2.tar.gz -C /tmp/
```

Cisco VIM Configuration Files

```
# example configuration files
/root/openstack-configs/setup_data.yaml.B_Series_EXAMPLE
/root/openstack-configs/setup_data.yaml.C_Series_EXAMPLE

# system maintained setup files - do not modify directly
# always supply user copy of setup_data.yaml
# when using ciscovim client
/root/openstack-configs/setup_data.yaml

# system inventory in pretty format
/root/openstack-configs/mercury_servers_info

# passwords store
/root/openstack-configs/secrets.yaml

# openstack configuration file
/root/openstack-configs/openstack_config.yaml

# RestAPI password
/opt/cisco/ui_config.json

# Insight password
/opt/cisco/insight/secrets.yaml
```

Enabling debug logs for certain OpenStack Services

```
# openstack config file
/root/openstack-configs/openstack_config.yaml

# help
ciscovim help

# list openstack keys
ciscovim list-openstack-configs

# help on reconfigure sub-command
ciscovim help reconfigure

# how to execute subcommand, example below
# important note: reconfigure requires a maintenance window
ciscovim reconfigure --setopenstackconfig KEYSTONE_DEBUG_LOGGING,CINDER_DEBUG_LOGGING
```

On controller and compute nodes, all services are run within their respective Docker™ containers.

To list the Docker containers in the node, execute the following:

```
[root@control-server-2 ~]# docker ps -a
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
258b2ca1d46a 172.31.228.164:5000/mercury-rhel7-osp8/nova-scheduler:4780
"/usr/bin/my_init /no" 25 minutes ago Up 25 minutes novascheduler_4780
ffe70809bbe0 172.31.228.164:5000/mercury-rhel7-osp8/nova-novncproxy:4780
"/usr/bin/my_init /st" 25 minutes ago Up 25 minutes novanovncproxy_4780
12b92bcb9dc0 172.31.228.164:5000/mercury-rhel7-osp8/nova-consoleauth:4780
"/usr/bin/my_init /st" 26 minutes ago Up 26 minutes
......
novaconsoleauth_4780
7295596f5167 172.31.228.164:5000/mercury-rhel7-osp8/nova-api:4780
"/usr/bin/my_init /no" 27 minutes ago Up 27 minutes novaapi_478
```

To view the Docker logs of any container, execute the following on the corresponding host:

```
ls -l /var/log/<service_name>/<log_filename>
e.g. ls -l /var/log/keystone/keystone.log
```

To get into a specific container, execute the following commands:

```
[root@control-server-2 ~]# alias | grep container
root@control-server-2 ~]# source /root/.bashrc
#execute the alias:
[root@control-server-2 ~]# novaapi
novaapi_4761 [nova@control-server-2 /]$
novaapi_4761 [nova@control-server-2 /]$ exit
exit
```

If the Docker status indicates a container is down (based on output of “docker ps –a”), collect the Docker service logs as well:

```
cd /etc/systemd/system/multi-user.target.wants/
ls docker* # get the corresponding service name from the output
systemctl status <service_name> -n 1000 > /root/filename # redirects the output to the file
```

For storage nodes running Ceph, execute the following to check the cluster status:

```
ceph -v # on monitor nodes (controller), show's ceph version
ceph -s # on monitor nodes (controller), show cluster status
ceph osd lspools #on monitor nodes (controller),list pools
ceph mon stat # summarize monitor status
ceph-disk list # on OSD / storage nodes; List disks, partitions, and Ceph OSDs
rbd list images # on monitor nodes (controller); dump list of image snapshots
rbd list volumes # on monitor nodes (controller); dump list of volumes
```

Kibana Dashboard Login

Kibana Dashboard Login

- Logging into Kibana Dashboard
- Using Dashboard Filters

Kibana is an open source data visualization platform that is used to explore Cisco VIM logs.

Logging into Kibana Dashboard

To log into the Kibana dashboard, follow the below steps:

1. With a terminal client, use SSH to log into your management node and enter the password to login. The following command shows that the IP address of the management node is 17.0.0.2:

   ```
   # ssh root@17.0.0.2
   root@17.0.0.2's password
   ```

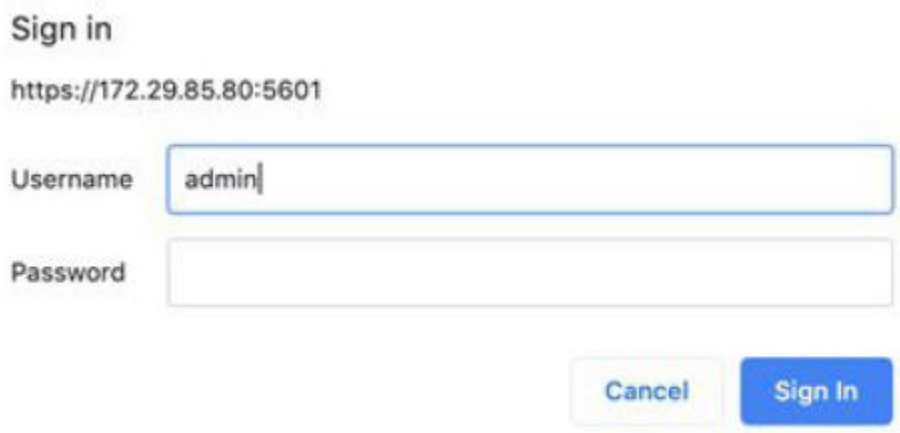
2. To obtain the password, check whether VAULT feature is enabled. If it is enabled, refer to the Vault section, otherwise locate the line containing KIBANA_PASSWORD in */root/installer-{tag id}/openstack-configs/secrets.yaml* during SSH terminal session. Note the value of the KIBANA_PASSWORD as it is used in Step 4.

   ```
   cat /root/installer-{tag-id}/openstack-configs/secrets.yaml
   ...
   KIBANA_PASSWORD: <note this value>
   ...
   ```

3. Navigate to the http://<management_node_ip_address>:5601.

   - Kibana uses the HTTPS + TLS to provide a secure connection between the browser and the Kibana service.
   - By default, Kibana uses the certificate located at */var/www/mercury/mercury.<crt|key>* or you can provide your own certificates in */root/openstack-configs/* directory (using the same mercury.<crt|key> file names).
   - If you are accessing Kibana for the first time, by default it shows self-signed certificate. Some browsers display the warning message *Yourconnectionisnotprivate*. Click **Proceed** to access the Kibana link. A window dialog box appears

4. Enter the **Username** and **Password**:

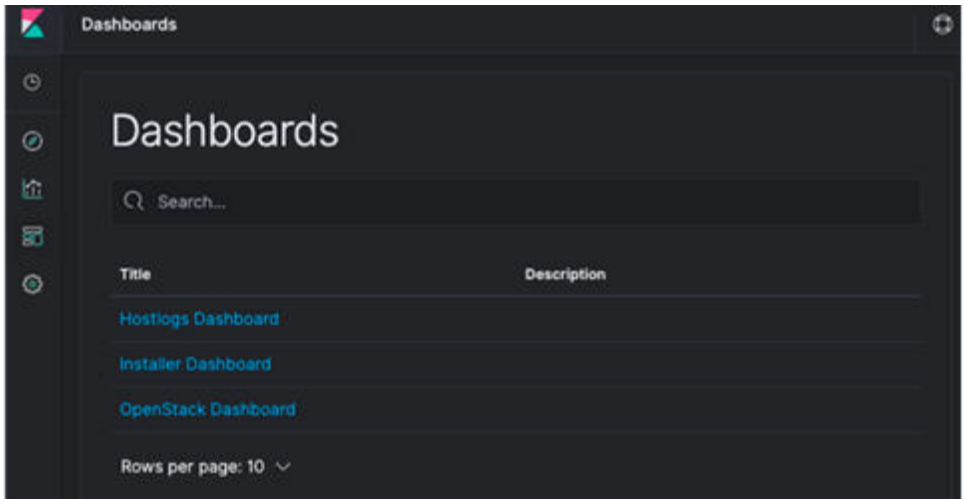
   User Name: admin

   Password: <value of KIBANA_PASSWORD from Step 2>. The Kibana dashboard displays the Cisco VIM service and installer logs.

5. Choose the desired dashboard from the list.

   Ensure that you do not use management options available on the left pane.

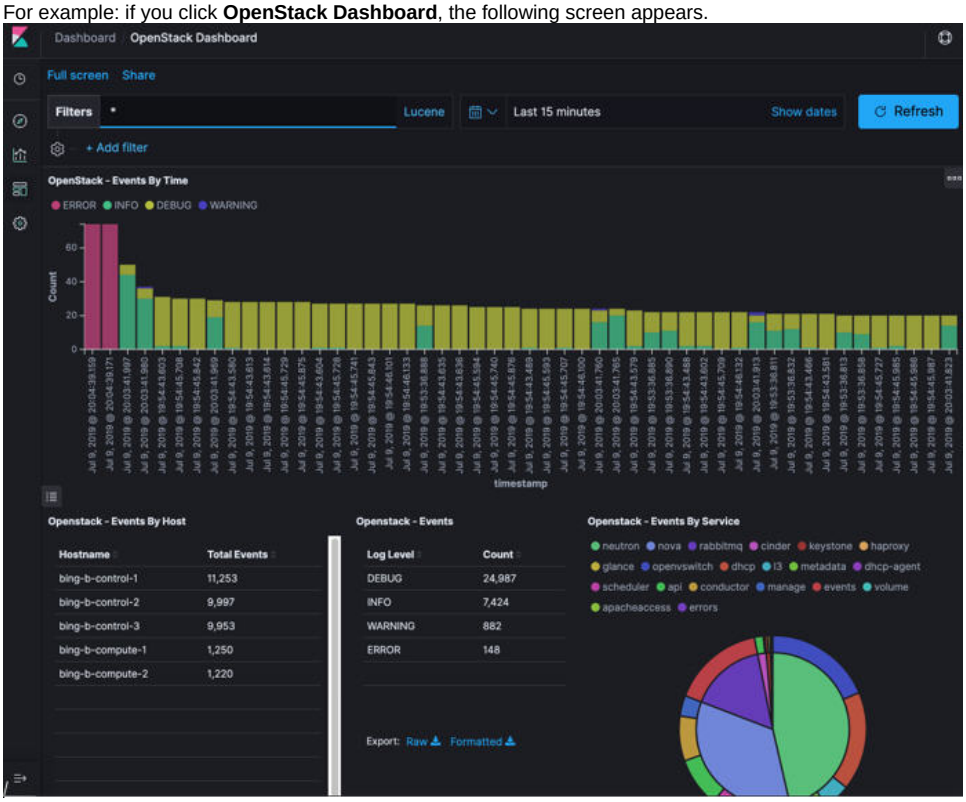
   The following figure shows the lists of Dashboards.

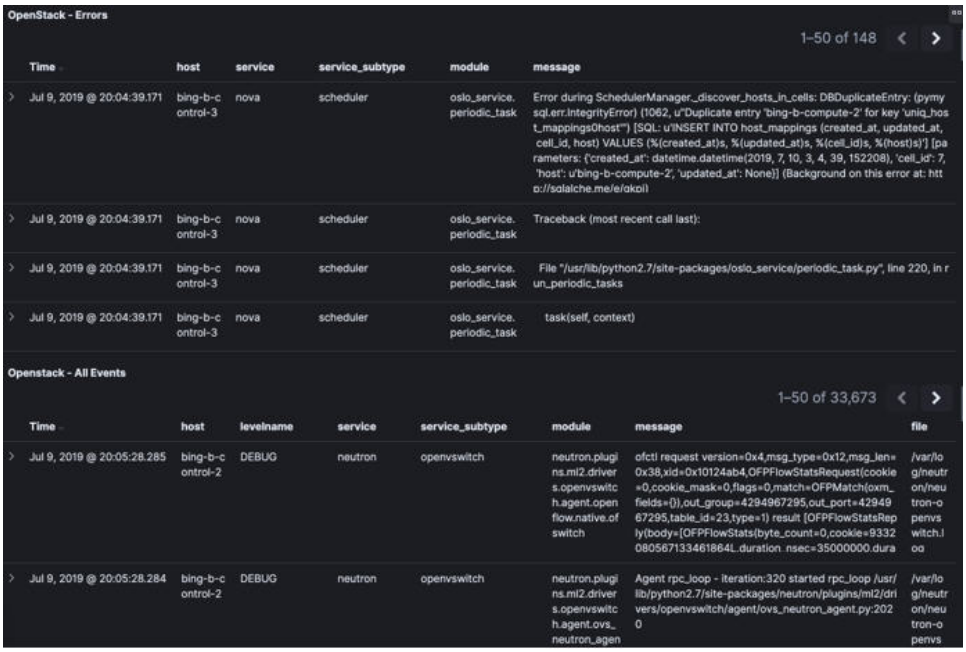


The following are the list of dashboards:

- Hostlogs Dashboard: Provides log information of the system for the cloud nodes. This displays the entries from the host logs-* index in Elasticsearch. It contains the log from */var/log/messages* file on each server.
- Installer Dashboard: Provides information about the management node and the installation process. It can only read uncompressed files. Hence, it reads the files prior to the cloud installation. This displays the entries from the installer-* index in Elasticsearch.
- OpenStack Dashboard: (openstack-* index) Provides log information about all the OpenStack processes. This displays the entries from the openstack-* index in Elasticsearch.
- VMTP Dashboard: Provides log information about the VMTP runs performed against the cloud. It displays the entries from the vmtp-* index in Elasticsearch.

For example: if you click **OpenStack Dashboard**, the following screen appears.





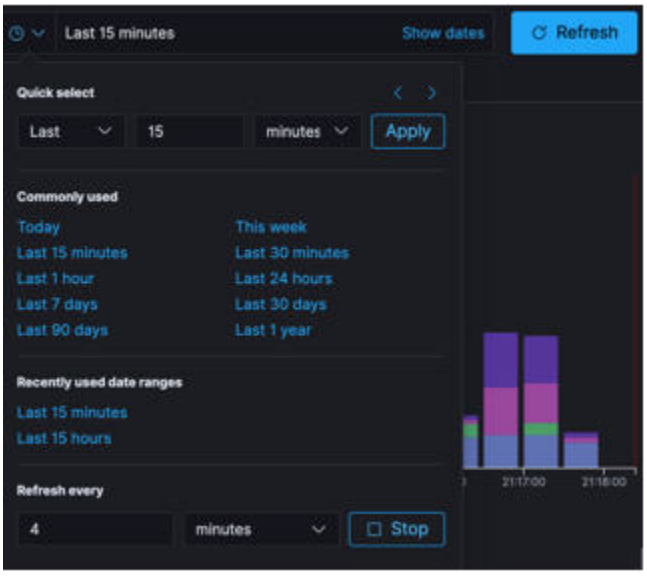
You can switch from one dashboard to another by selecting the appropriate dashboard from the right top bar menu. All dashboards have generic and specific fields.
The generic ones are:

- Title: Title is seen at the top left of the page. Title shows the dashboard that is displayed. For example: OpenStack Dashboard.
- Time bar: Time is seen at the top right of the page. Time indicates the time schedule for the log information. You can modify the time to indicate absolute, relative time in the past or specify automatically refresh rates.
- Filter bar: Search bar is an input field where you can enter a query in the Lucene syntax format to filter the logs by specific fields (which depend on the fields for the index being selected)
- Add a filter: Use this tab to introduce filters graphically.

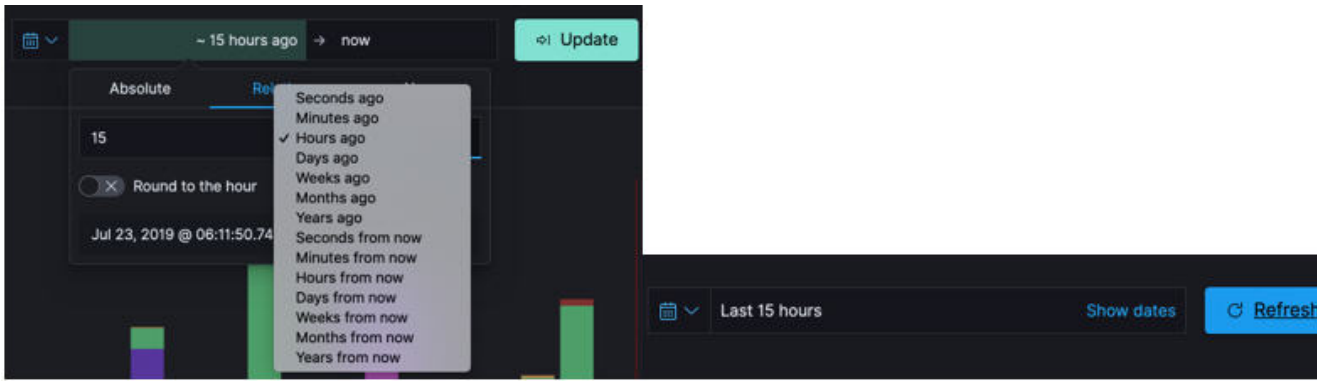
For more information on using Kibana, see the *Kibana documentation* (Version 5.5.1).
Cisco VIM stores the OpenStack logs in Elasticsearch. The Elasticsearch snapshots all the indices (where the data is stored) which are rotated on a periodic basis. You may not see the older data in Kibana, if the data is rotated out and/or deleted.
You can visualize the logs in Kibana when they are updated in Elasticsearch on the **Discover**. To debug something on Kibana, you can program the Kibana dashboard to auto-refresh at specific intervals (by default it is off).

To enable auto-refresh, click the Calendar drawing at the top right corner of the dashboard and program on **Refresh every** with desired value. Configure the desired value by clicking the **Start** and **Auto-refresh**.

Once you program a **Auto-refresh**, the Calendar drawing is replaced by a Clock. Then you can click **Stop** on the top navigator bar to pause the refreshing of logs events. You can also select intervals that you want to see the logs from.



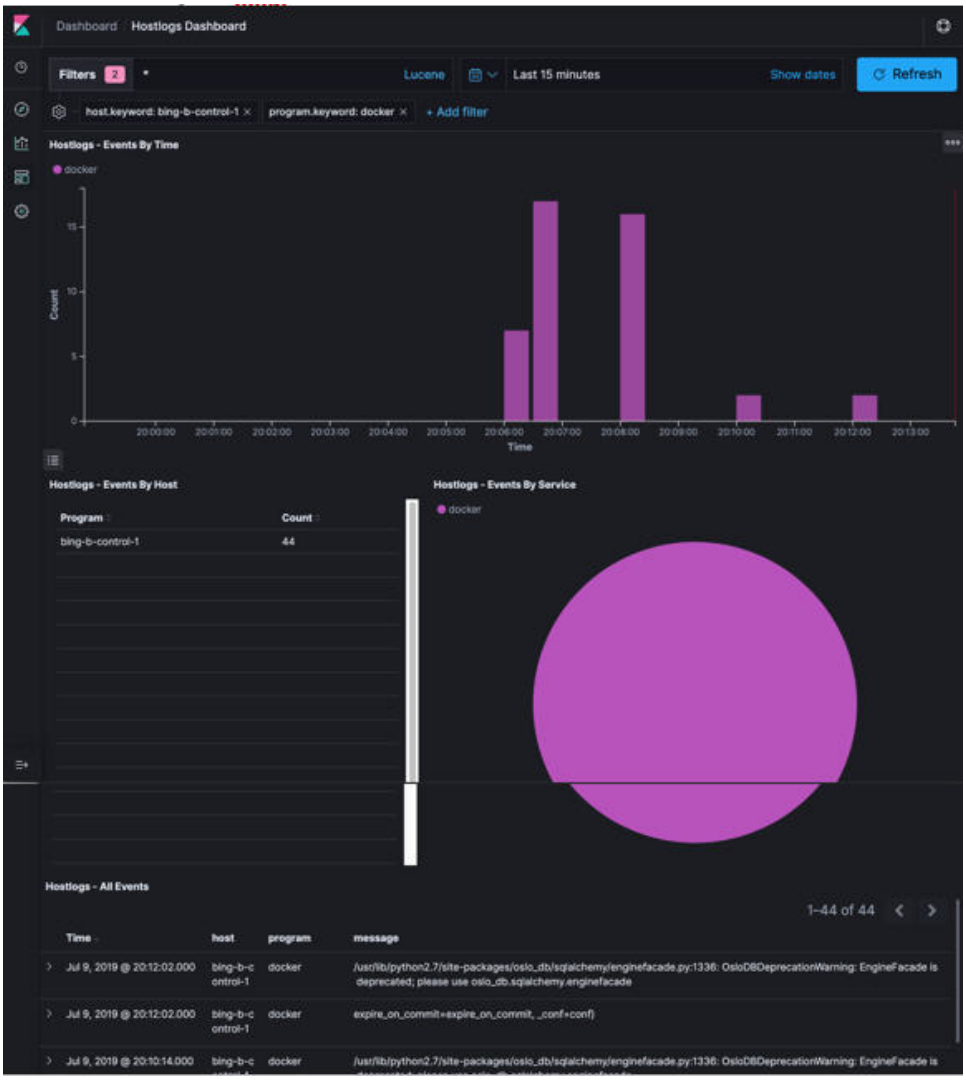
You can also select an absolute or relative interval:



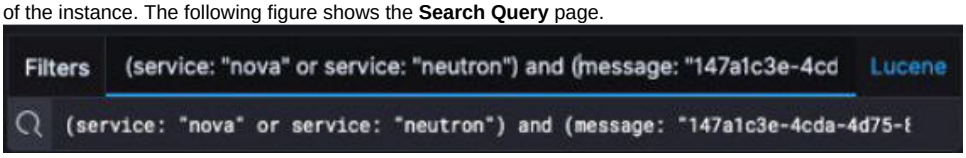
Using Dashboard Filters

1. On the **Hostlogs** Dashboard, in the **Events by Host** panel, choose a hostname and click the **+** or **-** symbol that appears close to the hostname to include or exclude that server from the filter. Then, click the desired slice on the **Events By Service** panel to add the docker service to the section.

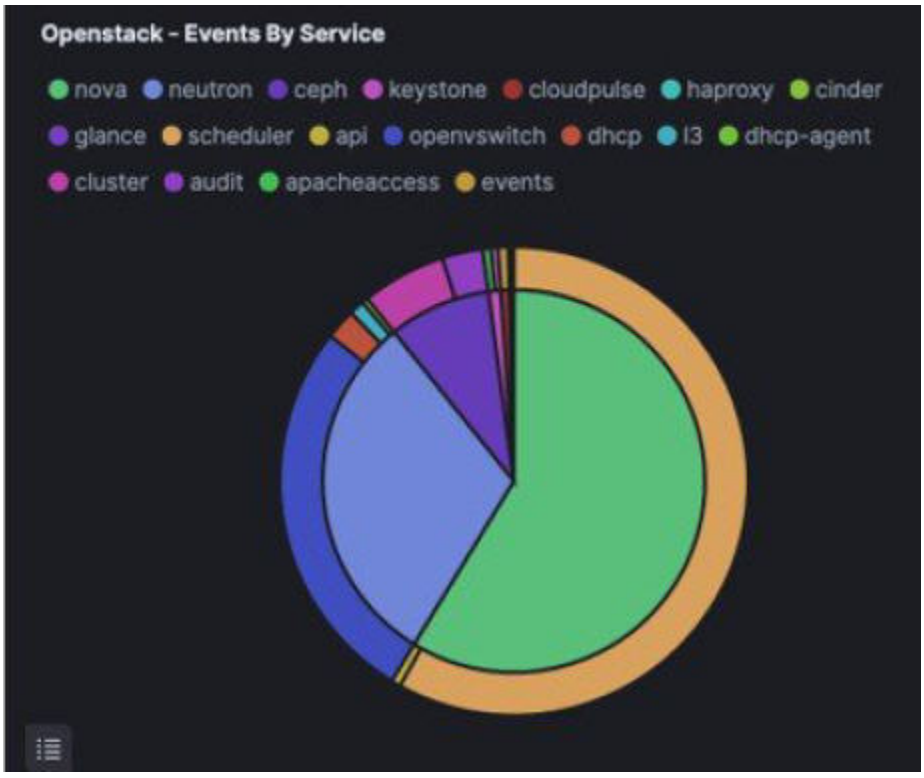
   Under **Filter**, you can see the included sections in green and excluded sections in red. The following figure shows the **Hostlogs** Dashboard:



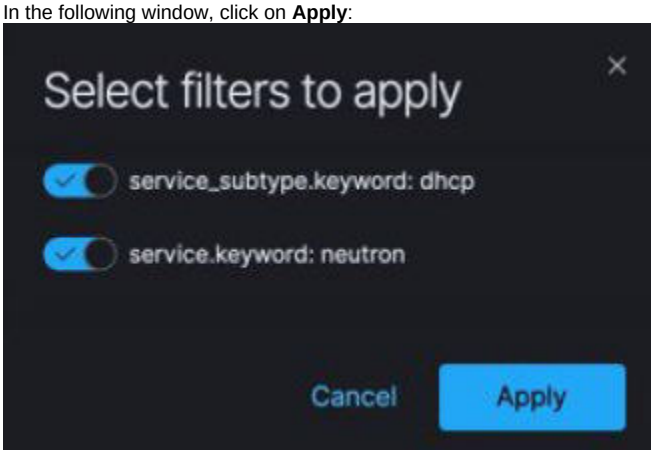
2. To know the log events in the Openstack for a given VM, enter the Lucene query (service: nova or service: neutron and message:<uuid>) in the **Filters** field which is on top of the Dashboard. where <uuid> is obtained from Horizon website or by executing openstack nova list for the identifier of the instance. The following figure shows the **Search Query** page.



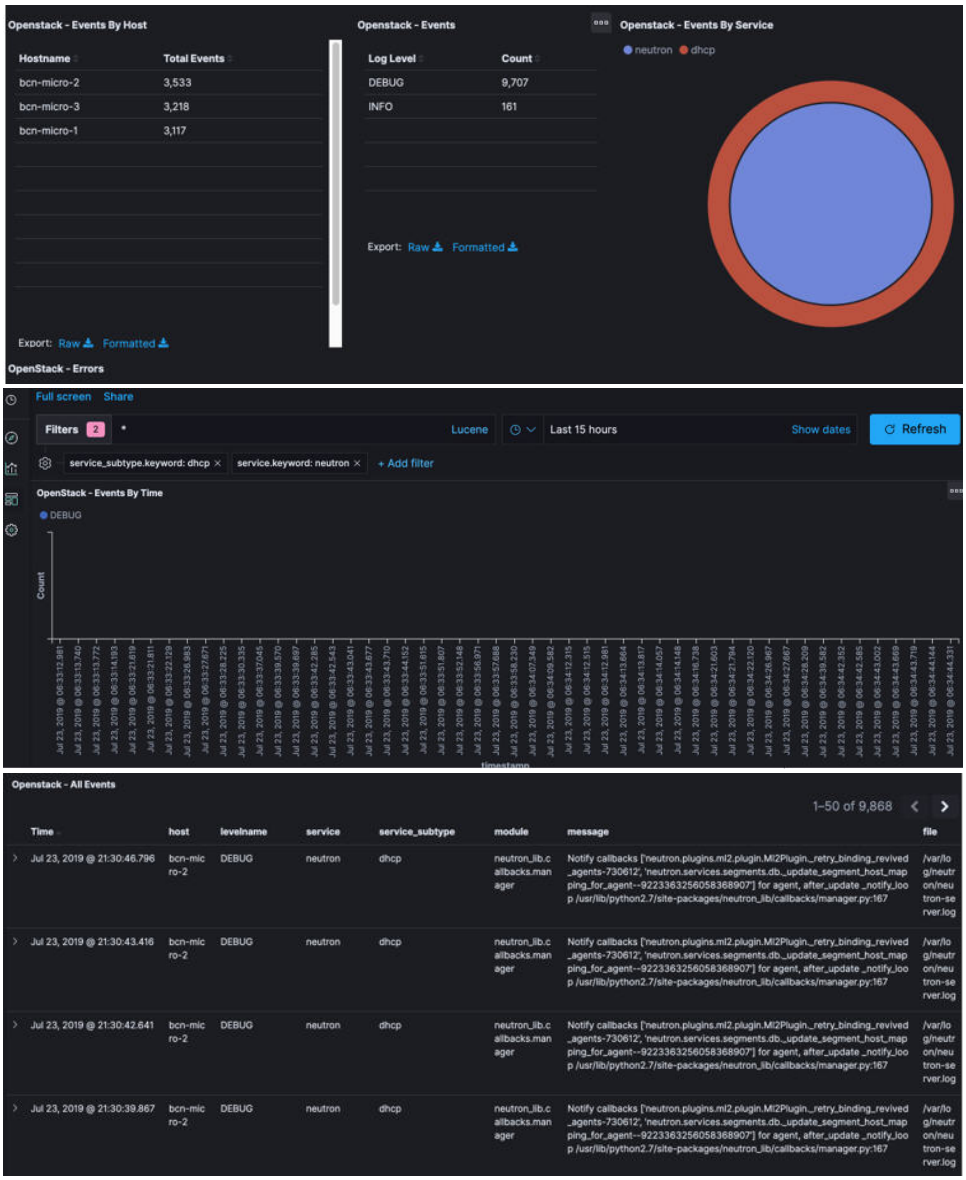
3. If you want to know about the DHCP events of the Openstack Neutron, select the filters by clicking outer circle of pie chart:
On the OpenStack Dashboard, the Openstack - Events By Service panel has a pie chart with the inner section for the services and the outer sections for the service_subtypes. To add filters for selecting all the events in a service (for example, neutron), click on the inner section of the pie. To add filters for selecting the service_subtypes (for example, dhcp), click on the outer circle of the pie. The following figure shows the **Events by Service** Panel.



In the following window, click on **Apply**:



4. You can scroll down the OpenStack Dashboard to see the OpenStack - Errors and the OpenStack - Events panel. The OpenStack - Errors panel displays the error messages. If there are no errors, the **Noresults found** message is displayed.

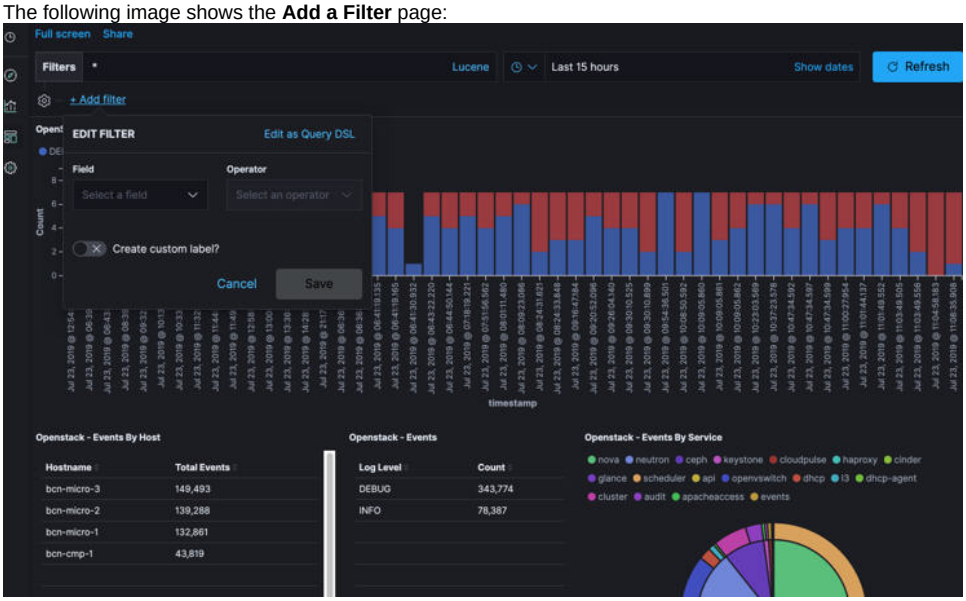


5. Without knowing the Lucene Syntax, you can set the filter criteria in the **Search** field using the **Add a filter +** option.

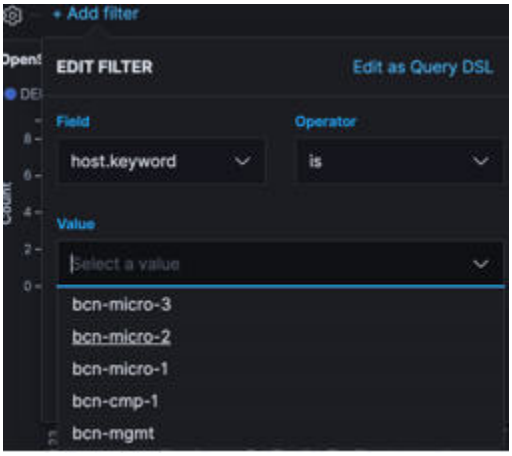
   To add a filter:

   - Click **Add a filter**.
   - Set the filter criteria by choosing appropriate label and operators from the drop-down lists, and entering keywords and click **Save**.

The following image shows the **Add a Filter** page:



6. To set the filter criteria, choose appropriate label and operators from the drop-down lists and enter the keywords in the following figure:



Rotation of Logs

Rotation of Cisco VIM Logs

Cisco VIM stores all logs in Elasticsearch. Elasticsearch indices are rotated on a periodic basis to prevent the disk space overflow by creating snapshots. The following lists show the snapshots that are defined in *openstack_config.yaml:*

```
# vi ~/openstack-configs/openstack_config.yaml
...#
Elk rotation parameters
elk_rotation_frequency: "monthly"         # Available: "daily", "weekly", "fortnightly", "monthly"
elk_rotation_size: 2                              # Unit is in Gigabytes (float is allowed)
elk_rotation_del_older: 10                       # Delete older than 10 units (where units depend on the
value set on elk_rotation_frequency)
```

You can change the frequency of the rotation by changing the values. For more information on how to set the Elasticsearch parameters through VIM API or CLI, see Reconfiguration of CVIM Parameters > *Reconfiguring Passwords and OpenStack Configurations*

Cisco VIM uses the open source Elasticsearch Curator tool to manage the Elasticsearch indices and snapshots. For more information about Elasticsearch handles snapshots, look at the official information on Elastic.co (Version 5.4) https://www.elastic.co/guide/en/elasticsearch/client/curator/5.4/index.html.

Elasticsearch

Elasticsearch

- Snapshot Manager Tool
- Remote NFS Backup for Elasticsearch Snapshots

Snapshot Manager Tool

The snapshot_mgr.sh tool wraps up the Elasticsearch Curator APIs. This tool helps you to access the snapshots of the logs that are maintained by the Elasticsearch.
Run the following command to view the snapshot logs available in the tools directory of the installer:

```
# ./tools/snapshot_mgr.py --help
usage: snapshot_mgr.py [options]

Snapshot Manager handles snapshot logs maintained by Elasticsearch

optional arguments:
-h, --help                                                  show this help message and exit
--list                                                          display all snapshots in Elasticsearch
--display                                                  GET_SS get details of the snapshot called <GET_SS>
--create                                                   create a snapshot
--restore RESTORE_SS                          restore snapshot named <RESTORE_SS>
--delete DELETE_SS                                  delete the snapshot called <DELETE_SS>
--autodelete threshold_warning threshold_low threshold_high autodelete snapshots until reach a disk space
threshold
```

Snapshot list gives you the details of the snapshot performed on the system like the UUID, the name of the snapshot, end time of the snapshot, the state and the indices where it was snapshotted:

```
# ./snapshot_mgr.py --list
+------------------------+------------------------+--------------------+---------
+-------------------------------------------------------------------------------------------------------------------
--------------------+----------+
| uuid                                          | snapshot_name                  | time_snapshot_ended |
state
|indices_snapshoted
| failures |
+------------------------+------------------------+--------------------+---------
+-------------------------------------------------------------------------------------------------------------------
--------------------+----------+
| 6WGVUnKjQbGtZYzfC0yeEg | curator-20180304140002 | 2018-03-04 14:00:04 | SUCCESS |hostlogs-
2018.03.02
| -        |
| U4IVWJNnQW6PdFWxpRUc-A | curator-20180304150001 | 2018-03-04 15:00:04 | SUCCESS |hostlogs-
2018.03.03
| -        |
| 5RxDuhnETC6TW4XSPDNZlw | curator-20180304160001 | 2018-03-04 16:00:24 | SUCCESS |installer-2018.03.03,
installer-2018.03.01, installer-2018.03.02, openstack-2018.03.02,hostlogs-2018.03.04, installer-2018.03.04   |
-        |
| k2gZYwLeRPO98bJZslI2pw | curator-20180305040002 | 2018-03-05 04:00:32 | SUCCESS |openstack-2018.03.03,
hostlogs-2018.03.04, installer-2018.03.04                                                                    |
-        |
```

To view the details of the individual snapshots, run the display option command:

```
# ./tools/snapshot_mgr.py --display curator-20180304140002
{ 'duration_in_millis': 1944,
  'end_time': '2018-03-04T14:00:04.019Z',
  'end_time_in_millis': 1520172004019,
  'failures': [],
  'indices': ['hostlogs-2018.03.02'],
  'shards': { 'failed': 0, 'successful': 5, 'total': 5},
  'snapshot': 'curator-20180304140002',
  'start_time': '2018-03-04T14:00:02.075Z',
  'start_time_in_millis': 1520172002075,
  'state': 'SUCCESS',
  'uuid': '6WGVUnKjQbGtZYzfC0yeEg',
  'version': '6.0.0',
  'version_id': 6000099}
```

To create a snapshot, run the following command:

```
# ./tools/snapshot_mgr.py --create
Executing: curl PUT
http://localhost:9200/_snapshot/es_backup/3a9b90c2979b46bf9c7b3f9223074d5d?wait_for_completion=true
-d
{'indices': 'installer-*,hostlogs-*,openstack-*,vmtp-*', 'ignore_unavailable': 'true',
'include_global_state': 'false'}
Response: {u'snapshot': {u'uuid': u'BSznQj1SQ9mjxxk9swTirQ', u'duration_in_millis': 46496,
u'start_time':
u'2018-03-06T16:37:49.774Z', u'shards': {u'successful': 35, u'failed': 0, u'total': 35},
u'version_id': 6000099,
u'end_time_in_millis': 1520354316270, u'state': u'SUCCESS', u'version': u'6.0.0',
u'snapshot': u'3a9b90c2979b46bf9c7b3f9223074d5d', u'end_time': u'2018-03-06T16:38:36.270Z',
u'indices': [u'installer-2018.03.06', u'vmtp-2018.03.02', u'hostlogs-2018.03.06',
u'hostlogs-2018.03.05',
u'installer-2018.03.05', u'openstack-2018.03.05', u'openstack-2018.03.06'],
u'failures': [], u'start_time_in_millis': 1520354269774}}
```

Run the following command to delete a snapshot:

```
# ./tools/snapshot_mgr.py --delete 3a9b90c2979b46bf9c7b3f9223074d5d
Executing: curl DELETE
http://localhost:9200/_snapshot/es_backup/3a9b90c2979b46bf9c7b3f9223074d5d -d None
Response: {u'acknowledged': True}
```

Restore the indices of a snapshot back to the Elasticsearch database by using the restore option. Run the following command to restore:

```
# ./snapshot_mgr.py --restore curator-20180306050001
Executing: curl POST
http://localhost:9200/hostlogs-2018.03.04,installer-2018.03.05,installer-2018.03.04,
openstack-2018.03.04,hostlogs-2018.03.05,openstack-2018.03.02/_close -d None
```

Remote NFS Backup for Elasticsearch Snapshots

Cisco VIM supports remote NFS backup of the Elasticsearch snapshots. This allows you to empty the disk space in the Elasticsearch snapshots. You can use the snapshot manager tool to manually create, list, show, and delete snapshots.
You can configure remote NFS backup by adding the following section to the setup_data.yaml configuration file:

```
ES_REMOTE_BACKUP: # Set if Elasticsearch backups can use a remote host
service: 'NFS' # Set if an remote NFS server is used
remote_host: <ip_addr> # IP of the NFS server
remote_path: /root/es_remote # Path to location of the backups in the remote server
```

Important considerations about the remote NFS directory on the remote server (specified by the remote_path config option):

- This directory allows the elasticsearch user (pid number 2020) and group mercury (pid 500) to read and write. Otherwise, Curator cannot copy the snapshots to the remote NFS directory.

- It is good if the folder is empty and is used only by Cisco VIM.
- Cisco VIM does not delete the information in this directory after unbootstrap.

You can enable or disable this feature by running reconfiguration commands. During reconfiguration, the remote_host ip or the remote_path can be changed.

CVIM-MON

CVIM Monitor (CVIM-MON)

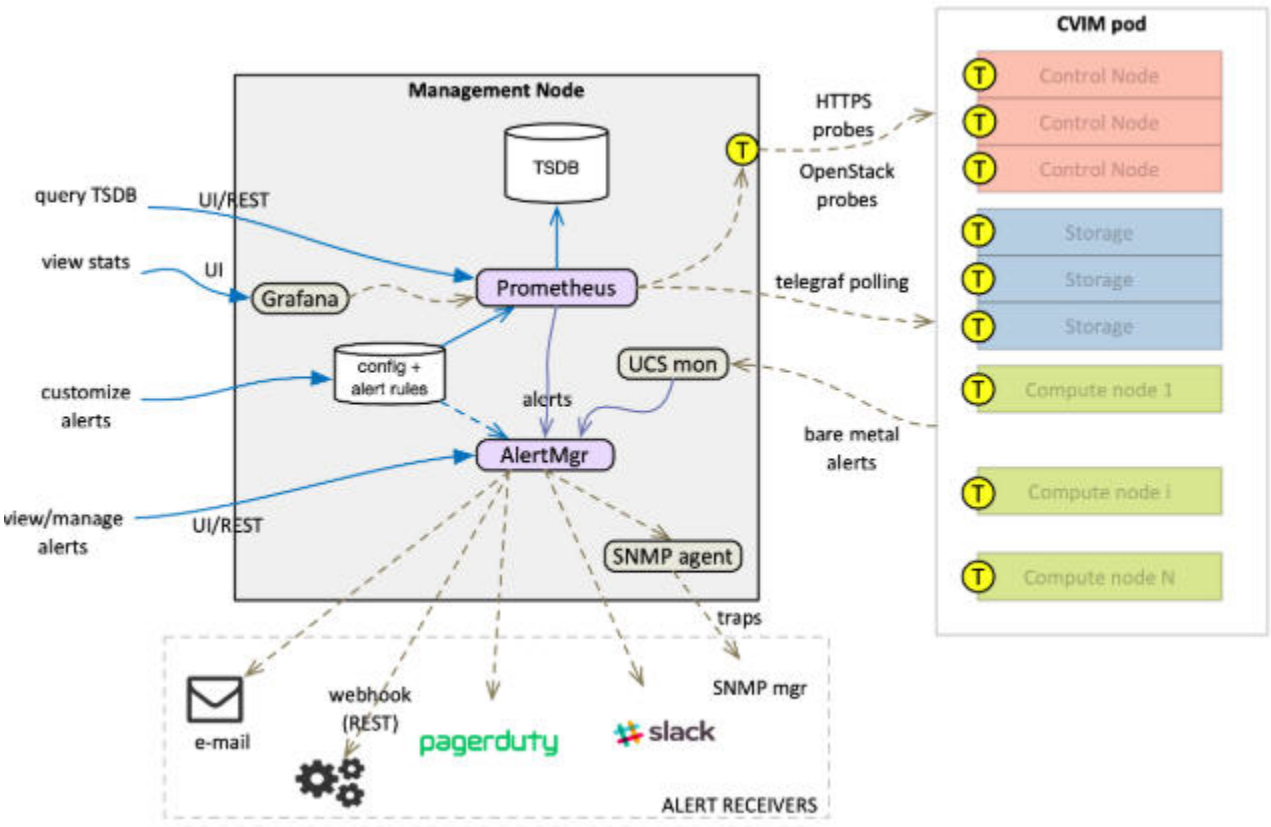
- Overview of CVIM Monitor
- Smart Metrics
    - Node Type Label
    - CPU Role Label
- Metrics Collection
    - Telegraf Metrics
- Alerting Rules
- CVIM-MON Web User Interface
    - Access Login
    - Pod <pod-name> Dashboard
    - Node Level Metrics Dashboard
    - Pod Level Metrics Dataplane Statistics Dashboard
    - Node Dataplane Statistics Dashboard
    - Specialized Dashboards
- SNMP for Monitoring
- Monitoring External Servers Using CVIM-MON

Overview of CVIM Monitor

The Cisco VIM Monitor (CVIM-MON) feature provides a comprehensive solution for monitoring the health and for tracking the usage of resources in the Cisco VIM pod infrastructure, when integrated with Cisco VIM installer.
This solution is available as a configurable 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 with pre-defined dashboards customized for Cisco 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 following figure shows the architecture of CVIM-MON:



A CVIM-MON stack includes the following components:

- a metric collection and metric query server based on the Prometheus open source component.
- an alert manager that is in charge of routing alerts based on the Prometheus alert manager.
- a WEB UI server based on the Grafana open source component.
- an SNMP agent in charge of forwarding alerts to SNMP receivers
- a bare metal alert collector (*UCS mon* in the figure - only available for Cisco UCS servers)

The metrics are collected using a small process running on each node in the pod and is based on the Telegraf open source component (marked *T* in the above figure).

The alerts are sent to a configurable number of receivers.

UI and REST services are provided to external applications or users.

Smart Metrics

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

Node Type Label

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

The following table shows the node types and their metric sources.

Node Type	Source of Metric
mgmt	Management node
controller	Controller node
compute	Compute node
storage	Storage node

aio	all-in-one node(micro-pod deployment)
hc	hyper-converged node (hyper-converged deployment)
edge	edge node (same as aio but without local storage)
external	external node (UM and Cisco VIM Software Hub)

CPU Role Label

CPUs in a Cisco VIM pod are statically categorized to perform specific functions. This partitioning is critical to guarantee a proper level of service for each subsystem independent of the load in the other subsystem. For example, it is imperative to isolate the CPUs reserved for the VPP virtual switch, from any other activity on the same compute node, to guarantee the virtual switch forwarding performance. The CPU metrics are labeled with a role (label name = *role)* to indicate the function of each CPU. This allows to aggregate CPU metrics based on category, which is a lot more useful than aggregating all CPUs. This categorization cannot be done with unlabeled metrics (by reading CPU time series from a TSDB), due to the following reasons:

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

The following table shows the roles used by the CVIM-MON to label CPU metrics and their static CPU assignment:

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

Metrics Collection

Telegraf Metrics

CVIM-MON collects hundreds of different metrics from each node through the Telegraf plugin. The metrics range from low-level kernel to infrastructure services. The interval between metrics collections is configurable between 10 seconds to 5 minutes.
The following table shows the list of Telegraf plugins installed as part of the CVIM-MON deployment:

Plug-in	Notes
ceph	Collects performance metrics from the MON and OSD nodes in a Ceph storage cluster
cpu	Detailed stats for every CPU (with role label)
conntrack	Collects stats from Netfilter's conntrack-tools
net_stats	Detailed metrics for physical and virtual network interfaces in Cisco VIM environment
disk	Detailed stats for every disk
diskio	Disk activity
docker	Detailed metrics on running docker containers
exec	Monitor EFK and Prometheus own storage usage
haproxy	HA proxy/ load balancer metrics for all HTTP services
http_response	Monitor HTTP services availability
hugepages	Monitors huge pages usage per NUMA node
internal	Collects metrics about the telegraf agent itself
ipmi_sensor	Bare metal metrics, including power usage, fan speeds, temperatures, and voltage
kernel	Linux kernel counters such as boot time and context switches, interrupt the node
libvirt	Nova and libvirt data and metrics from VMs running on compute or aio nodes

linkstate	Monitoring LACP, SRIOV links status
mem	Host level memory stats
net	Metrics about network interface and protocol usage (only for interfaces used by Cisco VIM)
chrony	Chrony NTP metrics
openstack	OpenStack related metrics (openstack services, hypervisors, servers)
processes	bare metal process metrics
rabbitmq	RabbitMQ metrics, currently disabled by default
swap	bare metal swap metrics
system	bare metal system load, uptime, and number of users logged in
x509	Monitors certificates

Alerting Rules

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

- Pending: Rule is triggered but the grace period has not expired.
- Fired: Rule is triggered for a period longer than the grace period.

You can monitor the alerts using the web user interface or API and can optionally convert the alerts into SNMP traps. You can configure CVIM-MON to send alerts as SNMP traps to any registered SNMP managers. The maximum number of SNMP managers supported is three, and a combination of SNMPv2 or v3 managers in different servers is supported. The following table shows the list of alerts, fault code and severity.

Alert Name	Fault Code	Severity	Description
instance_down	serviceFailure	critical	The node is not reachable or is down when the Prometheus server tries to scrape a target to retrieve its metrics. An instance down means that metrics from that target cannot be retrieved.
disk_used_percent	resourceThreshold	major	The storage device is used at over 90% capacity.
disk_filling_up_in_4h	resourceUsage	critical	The storage device is likely to run out of space in less than 4h.
docker_container_down	serviceFailure	critical	The docker container running a CiscoVIM infrastructure service is down. This event should never happen and indicates that an infrastructure container is failed or could not start.
link_down_lacp	hardwareFailure	warning	The LACP bonded link is in an error state if one of the two bonded links is no longer operating properly. For example, the error could be caused by the defective cable connection with the NIC, ToR, or a ToR port misconfiguration. The connectivity may still allow traffic to pass but at half the usual throughput. The defective link must be repaired quickly, to reinstate full bandwidth
link_down_sriov	hardwareFailure	warning	The SRIOV link is in down state. This usually indicates an issue with the physical cable wiring or a misconfiguration of the corresponding port on the ToR.
mem_available_percent	resourceThreshold	informational	There is less than 10% of available system memory. Regular 4K pages memory is used by both the system and OpenStack infrastructure services and does not include huge pages. This alert can indicate either an insufficient amount of RAM or abnormal memory usage by the system or infrastructure
memory_running_out_in_4h	resourceUsage	critical	This node is likely to run out of system memory in less than 4h. Based on the historical memory usage, this alert predicts that all the system memory will be used up in less than 4h. This condition should never happen and requires immediate troubleshooting by TAC before the system memory runs out.
swap_used_percent	resourceThreshold	warning	The node is using over 80% of the available swap space. Nodes should normally use only very little swap space. More than that the nodes will not use any swapping at all
conntrack_percent	resourceThreshold	warning	The node is using more than 80% of the available conntrack objects. This is mostly useful for OVS deployments. This indicates an abnormal use of host kernel conntrack resources.
reboot	hardwareFailure	warning	The node is rebooted in less than 10 minutes. Node reboots should be infrequent and be triggered only by the administrator when the node can safely be rebooted. Spontaneous and spurious node reboots should never happen.
system_n_users	resourceThreshold	warning	The node has more than 10 logged-in users.
ceph_error	serviceFailure	critical	The CEPH cluster is in error state and needs to be repaired immediately.
ceph_warning	serviceFailure	warning	The CEPH cluster is in a warning state. It requires attention for the repair to be done.
ceph_osdmap_num_in_osds	resourceThreshold	critical	The CEPH cluster has at least 1 OSD in the OUT state.

ceph_osdmap_num_up_osds	resourceThreshold	critical	The CEPH cluster has at least 1 OSD in the DOWN state.
ceph_pgmap_state_count	resourceUsage	critical	The CEPH cluster has at least 1 placement group that is not in active+clean state.
ceph_pgmap_bytes_avail_filling_up_in_4h	resourceUsage	critical	CEPH may run out of space within 4 hours.
ceph_pgmap_bytes_used_percent	resourceThreshold	warning	CEPH used capacity is over 70%.
ceph_pgmap_bytes_used_percent	resourceThreshold	critical	CEPH used capacity is over 80%.
haproxy_plugin_data_absent	other	informational	Not receiving any metrics from HAproxy for 10 minutes or more (should never happen).
haproxy_active_servers_down	serviceFailure	critical	Indicates that one or more HAProxy active server is not in the UP state.
haproxy_active_servers_backend	serviceFailure	critical	The number of haproxy active server backends is not three.
haproxy_active_servers_galera	serviceFailure	critical	The number of haproxy active galera servers is not one.
haproxy_backup_servers_galera	serviceFailure	critical	The number of haproxy backup galera servers is not two.
http_service_unavailable	serviceFailure	warning	The infrastructure HTTP service at given URL is not responding or is not reachable. This should never happen and may indicate an issue with the availability of the infrastructure service.
rabbitmq_node_running	serviceFailure	critical	At least one of the three rabbitMQ nodes is not running.
rabbitmq_node_mem_used_percent	resourceThreshold	critical	Memory used by rabbitMQ is at 90% of its configured maximum limit.
rabbitmq_queue_consumers	resourceThreshold	critical	One or more rabbitMQ queues have no consumer.
rabbitmq_queue_messages	resourceUsage	critical	The number of queued or unread ready and unacknowledged messages is over 300.
ntpq_offset	resourceThreshold	warning	The mean offset (phase) in the times reported between the local host and remote peer or server is over 2500 milliseconds.
cp_openstack_service_down	serviceFailure	critical	The indicated openstack service is not reachable and likely to be down.
cp_hypervisor_down	serviceFailure	critical	The Nova hypervisor is down.
certificate_expiring_5d	other	critical	The certificate will expire in less than 5 days and must be replaced.
certificate_expiring_10d	other	warning	The certificate will expire in less than 10 days
certificate_expiring_45d	other	informational	The certificate will expire in less than 45 days

- The default alert rules can be fetched using the built-in tool from the management node,

```
[root@milpitas ~]# get_alerting_rules
```

CVIM-MON Web User Interface

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

Access Login

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

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

- The **Forgot your password?** option in the Grafana login page is not functional.
- New password can be generated for Grafana, by running Cisco VIM reconfiguration with the regenerate secrets option.

Pod <pod-name> Dashboard

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

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

Node Level Metrics Dashboard

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

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

The dashboard provides the utilization charts for the following parameters:

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

Pod Level Metrics Dataplane Statistics Dashboard

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

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

Node Dataplane Statistics Dashboard

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

- Two gauges with aggregated (all TX+RX) throughput in PPS and bps across physical interfaces on the specific node.
- One gauge with total virtual interfaces (attached to VMs) running on the specific node.

Statistics	Description
Specific VM drop rate	Specific VMs virtual interfaces TX/RX drops rate out of all TX/RX packets on that VM in a 20m timeslot.
Node throughput in packet-per-second (pps)	Total throughput in pps on all physical interfaces on that specific node.
Node throughput in bits-per-second (bps)	Total throughput in bps on all physical interfaces on that specific node.
Average Node packet size	Size calculated from total per-interface bytes divided by total packets on all node's physical interfaces.
VM throughput in packet-per-second (pps)	Total throughput in pps on all physical interfaces on that specific VM and per VM interface.
VM throughput in bits-per-second (bps)	Total throughput in bps on all physical interfaces on that specific VM and per VM interface.
Average VM packet size	Size calculated from total per-interface bytes divided by total packets on all VM's virtual interfaces.
VM error rate	Error rate on virtual interfaces TX/RX out of all TX/RX packets in a 20m timeslot.

Specialized Dashboards

The following table provides the list of specialized dashboards:

Dashboard Name	Description
VM Inventory	Shows all the VM running in the pod or on external servers
Capacity Usage	Shows overall resources used by OpenStack projects, by hypervisors and overall CEPH resource usage
Project Dashboard	Shows detailed resources used by each OpenStack project
OpenStack services	Chart shows the state of all OpenStack services, infrastructure containers, and hypervisors.
Alerts	Alerts that are triggered passed the grace period or pending (triggered but still within their grace period).
HTTP Servers Stats	Chart to monitor all internal and external HTTP services in the pod.
CEPH	CEPH storage chart, for example, overall OSD CPU load.
NTP	Chart to monitor NTP on the pod.
RabbitMQ	Chart related to rabbitMQ
Etcd	Chart related to etcd. Only available for ML2/VPP deployments.
Advanced Metrics	Chart that monitors the management node activity such as: • Prometheus and Elasticsearch disk usage • Prometheus scraping stats
IPMI	Chart that present bare metal sensor and counters: • Temperature • Voltage • Fan Speed • Power • Error counters

SNMP for Monitoring

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

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

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

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

The SNMP trap contains the following information:

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

For more details, see CISCO-VIM-MIB.my.4.0 definition of the MIB at ftp://ftp.cisco.com/pub/mibs/v2/.
CVIM-MON is integrated into Cisco VIM as an optional component and offered as an add-on with an additional license. CVIM-MON is enabled by extending the *setup_data.yaml* file with relevant information. To enable CVIMON, see Enabling CVIM-MON on Cisco VIM

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

Monitoring External Servers Using CVIM-MON

For details on how to install the Telegraf agent on the external servers, for example, UM and Cisco VIM software hub nodes, see CVIM Monitor. The Telegraf agent that is installed on the external servers have the following plugins enabled:

- cpu
- disk and diskio
- net and net_stats
- mem
- ipmi_sensor
- kernel
- processes
- swap
- system
- conntrack
- internal: Monitors the health of the Telegraf agent and enabled input plugins
- prometheus_client: Provides scraping access to any Prometheus compatible scraper

Based on what the external server contains, the following plugins are also enabled:

- libvirt: Collects metrics related to any virtual machines running on the external server
- x509_cert: Collects metrics for external servers with certificates to monitor
- docker: Collects metrics on external servers with running docker containers
- http_response: Collects metrics on external servers if endpoints exist for monitoring

These metrics are integrated in the Prometheus TSDB with the following arrangement:

- Metrics collected from external servers are distinguished from Cisco VIM pod metrics by a *node_type* label value of *external* (external host name being different)
- Metrics for all CPUs have the label *tag* set to *host* by default. You can customize these metric labels 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.
- External servers are considered as part of the attached Csico VIM pod.
- For central monitoring, external servers are assigned the same region or metro as the attached Cisco VIM pod.

In the Grafana dashboard, all counters such as node count per type, alert counters, power consumption aggregates and so on, include external servers as applicable.

Network Performance Test

Network Performance Test

NFVBench is a network performance benchmarking tool integrated with Cisco VIM. For more details, refer to Setting up NFVbench.

Unified Management NG

Unified Management Next Generation (NG)

- NG UM Blueprints
- Using NG UM

NG UM Blueprints

NG UM Blueprints

- NG Overview of UM Blueprint
- NG Create Blueprint
- NG Activating Blueprint in Existing Pod
- NG Validating Blueprint
- NG Viewing and Downloading Blueprint

NG Overview of UM Blueprint

Overview of UM Blueprint

Blueprints contain the configuration metadata required to deploy an OpenStack system through a Cisco VIM pod in Cisco VIM Unified Management.

In next-generation (NG) UM, you can upload a *yaml* file that contains the metadata for a blueprint. You can also create a blueprint from an existing OpenStack system that you are configuring as a Cisco VIM pod. The configuration in the blueprint is specific to the type of Cisco UCS server in the OpenStack system.

Using blueprint, you can quickly change the configuration of an OpenStack system. You can create or upload multiple blueprints for a Cisco VIM pod, but only one blueprint can be active. If you change the active blueprint for a pod, you must update the configuration of the OpenStack system to match the new blueprint.

You can modify and validate an existing blueprint, or delete the blueprint. However, you cannot modify any of the configuration metadata in the active blueprint for a Cisco VIM pod.

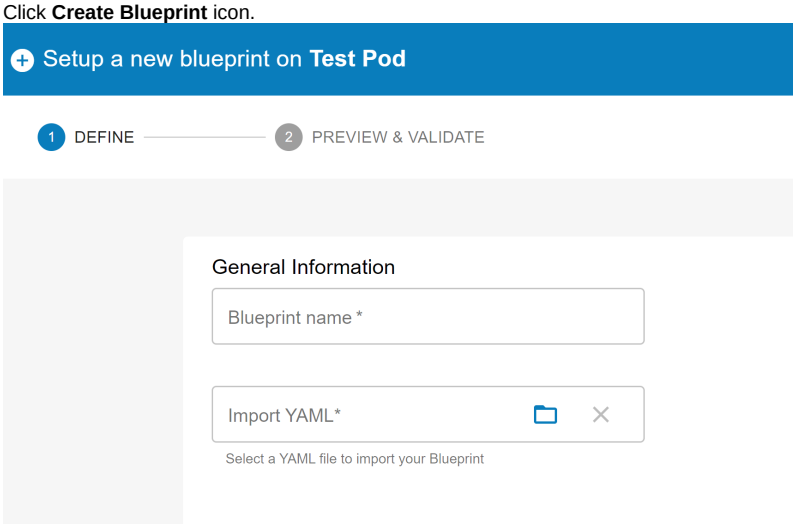
NG Create Blueprint

Creating a Blueprint Using Upload Functionality

Before you begin

- You must prepare *setup_data.yaml* for your pod beforehand. To create *setup_data.yaml,* see *setup_data.yaml.C_Series_EXAMPLE* file.
- You can upload only one blueprint at a time.

1. Log into UM as pod user and select the pod on the landing page. If you have not registered the pod on UM node, see Registering New Pod to UM
2. From the navigation pane, choose **Pod Setup** > **Blueprint Management.**
3. Click **Create Blueprint** icon.



4. Enter a unique **Blueprint Name**
5. Browse the *setup_data.yaml* created beforehand and click **Next**. On this page, a YAML editor is integrated on UI to make changes in *setup_data.yaml.*
6. Once *setup_data.yaml* is finalized, click **Offline Validation.** Based on the *setup_data.yaml,* the management node validation, software validation and hardware validation are done.
7. If offline validation is successful, click **Save** to save the *setup_data.yaml.* You will be redirected to the **Blueprint Management** page.
8. If offline validation fails, click **Cancel** to redirect to **Blueprint Management** page. Follow from *Step 1,* after modifying *setup_data.yaml*
9. Once the blueprint is *Active,* click **Install Setup** icon to begin Cisco VIM deployment on the registered pod.

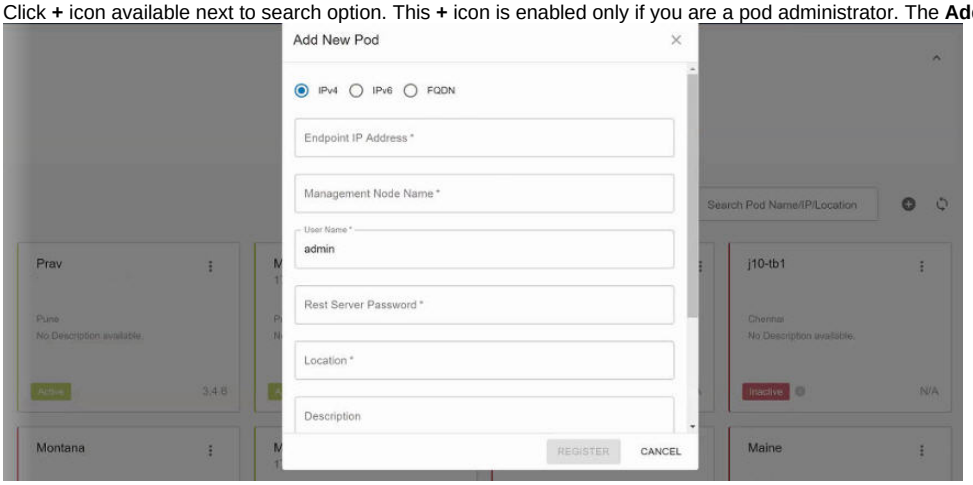
NG Activating Blueprint in Existing Pod

Activating Blueprint in Existing Pod with OpenStack

To activate blueprint for an existing pod installed with OpenStack, follow the below steps:

> **Before you begin**
>
> You must have a pod with OpenStack installation in *active* state. If the OpenStack installation is in failed state, the UM UI cannot fetch the blueprint.

1. Log into UM as pod user**.**
2. Click **+** icon available next to search option. This **+** icon is enabled only if you are a pod administrator. The **Add New Pod** window is displayed.



3. Enter the following:
    a. Management Node IP Address.
    b. Management Node Name (Any friendly name).
    c. REST Server Password. You can get the password from */opt/cisco/ui_config.json* file on Cisco VIM management node.
    d. Description about the management node.
    e. Location.
    f. Root CA Certificate. You can get this file from */var/www/mercury/mercury-ca.crt* on Cisco VIM management node.
4. Once the pod status is *Active,* click on that pod to view pod dashboard.
5. From the navigation pane, choose **Pod Setup** > **Blueprint Management**.
6. Choose the **NEWSETUPDATA**. This is the same setup data that is used by *ciscovimclient* to run the installation on the management node.

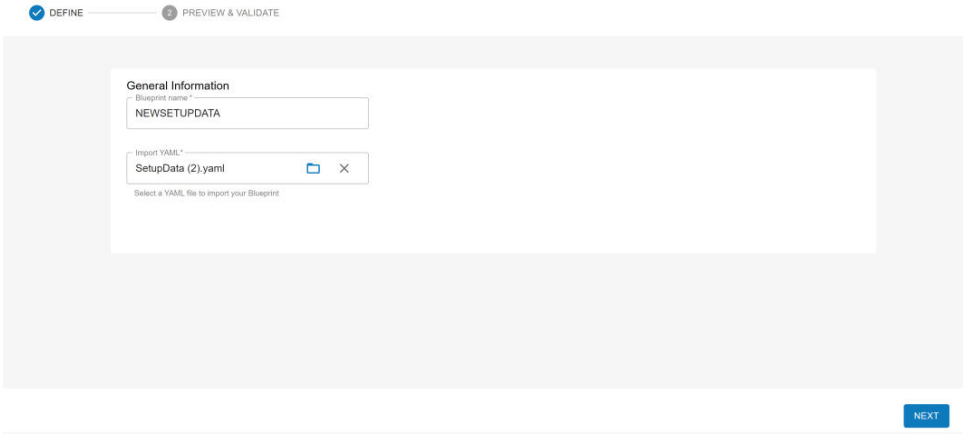
NG Validating Blueprint

Validating Blueprint

You can upload multiple blueprints for the registered pod and perform offline validation for that blueprint. Once the offline validation is completed successfully, the blueprint is marked as *Active* and can be used for Cisco VIM pod deployment. Once the Cisco VIM deployment is completed successfully, the blueprint is marked as *Deployed.*

To validate your *setup_data.yaml,* follow the below steps:

1. Log into UM as pod user and select the pod from the landing page.
2. From the left navigation pane, choose **Pod Setup**.
3. Click **Create Blueprint** icon.
4. Provide a unique name for blueprint and upload an existing *setup_data.yaml* file.



5. Click **Next** and edit the *yaml* file as required.
6. Click **Offline Validation**.

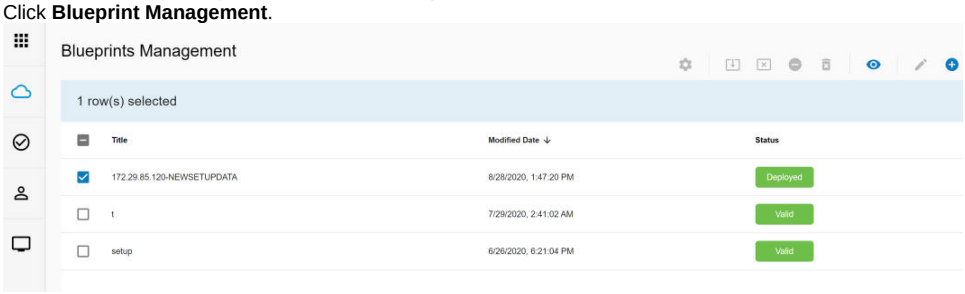
    a. If the validation is successful, the UM allows you to save the blueprint.
    b. If the validation fails, a popup shows all errors and warning in the provided *yaml* file.

NG Viewing and Downloading Blueprint

Viewing and Downloading Blueprint

On the **Blueprint Management** page, you must have at least one blueprint in *Active, In-Active,* or *In-progress* state.

1. Log into UM as pod user and select the pod on the landing page.
2. From the left navigation pane, choose **Pod Setup**.
3. Click **Blueprint Management**.



4. Select a blueprint from the table.
5. To view the blueprint in the *yaml* format, click **Preview and Download YAML** icon. A pop-up window with **Download** button is displayed.
6. Click the **Download** button. The *yaml* is saved locally with the same name of the blueprint.

Using NG UM

Using NG Cisco VIM Unified Management (UM)

- NG Naming Conventions
- NG UM Administrator
- NG Pod Administrator
- NG Pod Users
- NG Pod User Administration
- NG UM Dashboard
- NG Pod Registration/Management
- NG Monitoring Pod Status
- NG Context Switching Between Pods
- NG Cisco VIM Pod Software Update
- NG Managing Passwords
- NG Day 2 Reconfigure/Enablement
- NG CVIM RMA for Auto-TOR
- NG Cisco VIM Validation Tools
- NG Cisco VIM Operational Tools
- NG Managing Hardware
- NG Managing Power
- NG Uploading Glance Images

NG Naming Conventions

Naming Conventions

The following are the naming conventions used in Cisco VIM UM

1. Super Administrator (UM Admin): User having access to UM Admin profile.
2. Pod Administrator: User having access to register a pod in the system. Only UM admin can add new Pod Admin in the system.
3. Pod users (Normal users): Denotes all the users associated with the pod with either Admin or Viewer Role. Admin is the role assigned to user to give full access to a specific pod.

- User who are UM admin or Pod admin but not associated with any pod are not counted in UM admin dashboard user count section.
- Only Pod Admins can register a new pod.
- Every pod must have a user with *Admin* role.
- You cannot revoke/delete user if the user is the last user on the pod with *Admin* role.
- You cannot delete a user, if the user is a Pod admin or UM admin.

NG UM Administrator

UM Administrator

- Adding UM Admin
- Revoking UM Admin

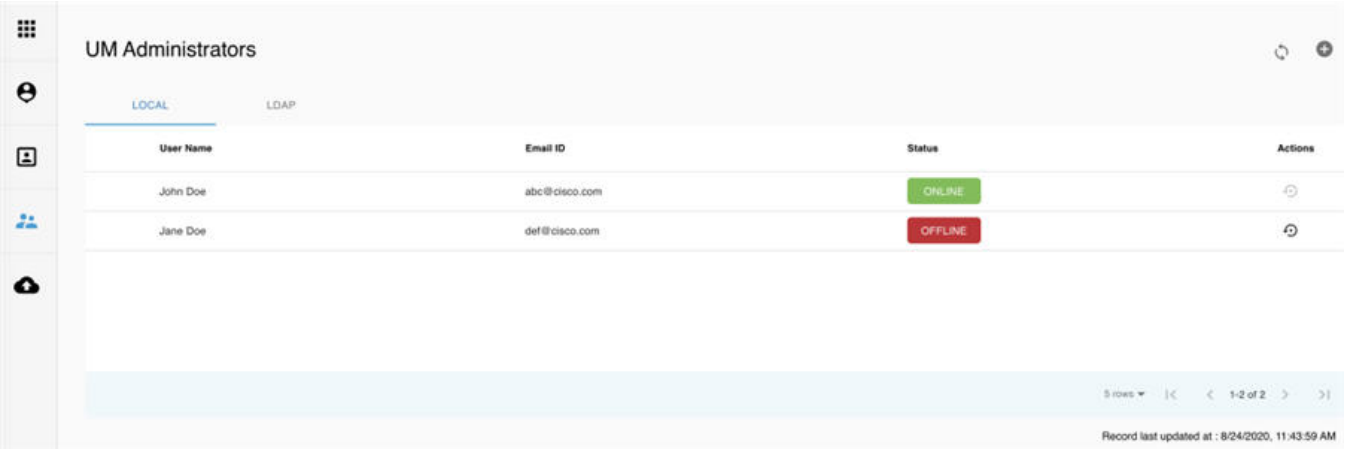
UM Admin have privileges to perform all administrative tasks like creating additional UM admins, Pod Admins, and Pod users, and other operations on registered pods. When you log in as UM Admin, you are redirected to UM Dashboard. You can then switch to pod profile to view registered pods and perform any operation on the pods.

> ⚠ After installing Unified Management node, a default UM is pre-created.

Login credentials for default UM admin are as follows:

Username: UI_ADMIN_EMAIL_ID from *insight_setup_data.yaml*

Password: UI_ADMIN_PASSWORD from */opt/cisco/insight/secrets.yaml*



Fields	Description
Username	Specifies the username of registered UM administrator.
Email Address	Specifies the email ID of the registered UM administrator.
Status	Indicates whether UM admin is currently logged in.
Action	Enables the UM admins to delete other UM administrators registered on the UM node.

> ⚠ Ensure that there must be at least one UM admin in the system.

Adding UM Admin

To add a UM admin perform the following steps:

1. Log into UM as UM Admin and navigate to **UM Administrators** page.
2. Click **+** on top right corner, to add a UM Administrator.
3. Select User auth for the new user. This option is enabled only if LDAP mode is true.
4. Enter the email ID/ LDAP user id (if LDAP user attribute is set to uid).
    - If email is already registered, the **Username** gets populated automatically.
    - If email is not registered, an email is sent to the user with the verification token. If User auth is set as LDAP, no verification token email is sent.
5. Navigate to **Login** page *https://<br_api>:9000*
6. Enter the **Email ID** and **Password** of new UM Admin**.**
7. Click **LOGIN AS UM ADMIN** to view the UM dashboard.

Revoking UM Admin

UM admin can revoke another UM admin's permission. To revoke UM Admin permission for any user, click **Revoke** icon.

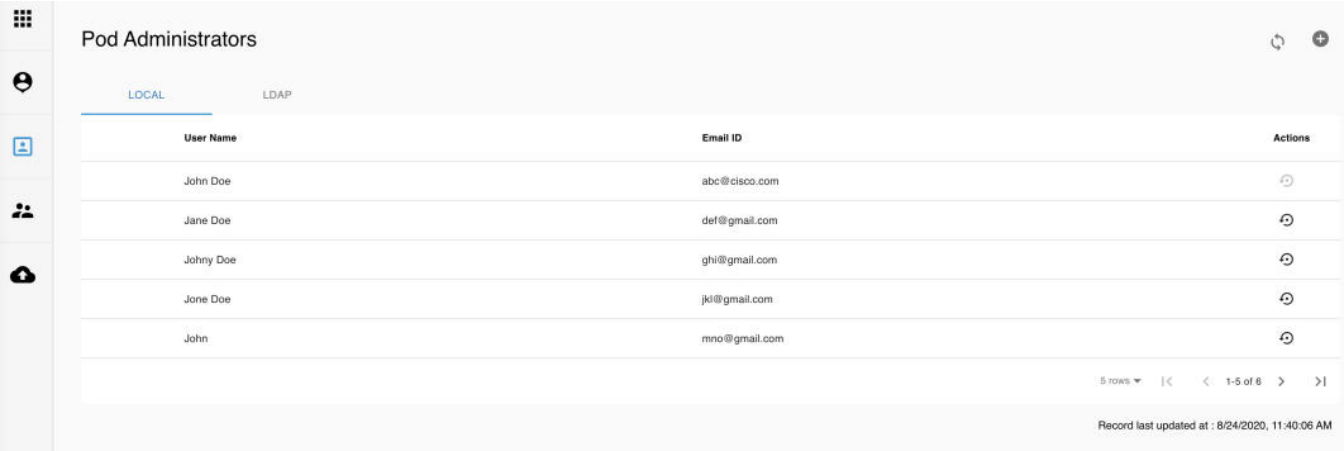
> ⚠ You cannot revoke the permission for yourself. You can revoke user permission, if the user is not associated with any pod. After revoking, the user is deleted from the system.

NG Pod Administrator

Pod Administrator

- Adding Pod Admin
- Revoking Pod Admin

Pod admins have permission to register new pods in the system. UM admin can add any number of Pod admins in the system.



Fields	Description
Username	Indicates the username of registered pod administrator.
Email Address	Specifies the email ID of registered pod administrator.
Action	Enables the UM admins to revoke the permission of pod administrators registered on the UM node.

Adding Pod Admin

1. Log into UM as UM Admin and navigate to **POD Administrators** page.
2. Click **+** icon on the top right of **Pod Administrator** page.
3. Select User auth for the new user. This option is enabled only if LDAP mode is true.
4. Enter the Email ID/LDAP user id (if LDAP user attribute is set to uid) of the user.
    - If the email is already registered, the **Username** gets populated automatically.
    - If the email is not registered, an email is sent to the user with the verification token. If User auth is set as LDAP, no verification token email is sent.
5. Navigate to **UM Login** Page *https://<br_api>:9000*
6. Enter the **Email ID** and **Password** of new pod admin.
7. Click **LOGIN as POD USER**. It redirects to the landing page where the Pod admin can register a new pod.

Revoking Pod Admin

UM admin can revoke pod admin's permission anytime. To revoke pod admin permission for a user, click **Revoke** icon.

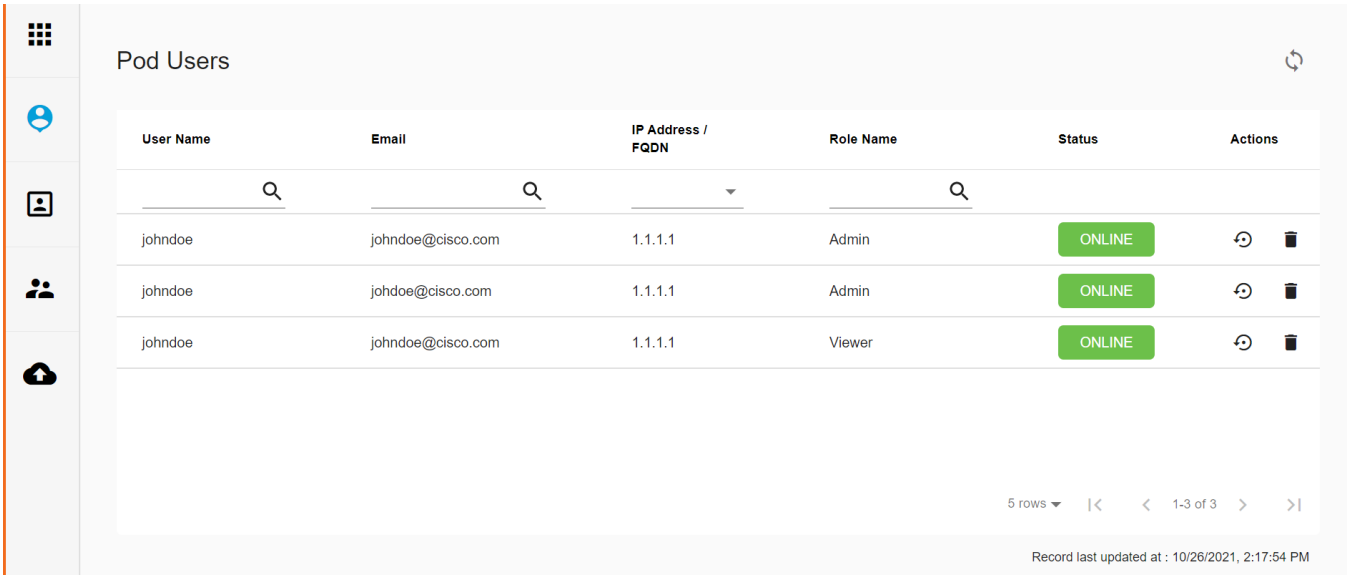
> ⚠ You cannot revoke self permission.

NG Pod Users

NG Pod Users

- Revoking Pod User
- Deleting Pod Users

The **Pod Users** page gives you detailed information of all the pod users registered on the UM node.



- Username - Specifies the username of registered pod user.
- Email Address - Specifies the email ID of registered pod user.
- IP/FQDN - Indicates the pod with which the registered user is associated.
- Role Name - Specifies whether user has Admin or Viewer Role for that pod.
- Status - Indicates whether the user is currently logged in.

A user with UM admin privileges can perform the following actions:

- Manages all the pod users.
- Revoke pod user's permission for a specific pod.
- Delete a pod user.

Revoking Pod User

To revoke a user's permission for a particular pod, click **Revoke** icon under **Action** column of the required pod user. This **Revoke** icon is disabled, if the registered pod user is the only user with a *Admin* role for that particular pod. In this case, you can grant another user with *Admin* role for that pod and then revoke permission of the old user.

> ⚠ If the user is revoked from the last associated pod, the user is automatically deleted from the system.

Deleting Pod Users

UM admin can delete any user from the system by clicking **Delete** icon under the **Action** column of the required pod user. The delete operation is not permitted, if the user has *Admin* role. In this case, you can grant another user with *Admin* role for that pod and then proceed with deletion of the old user details. UM admin must revoke respective permission first and then proceed further.

NG Pod User Administration

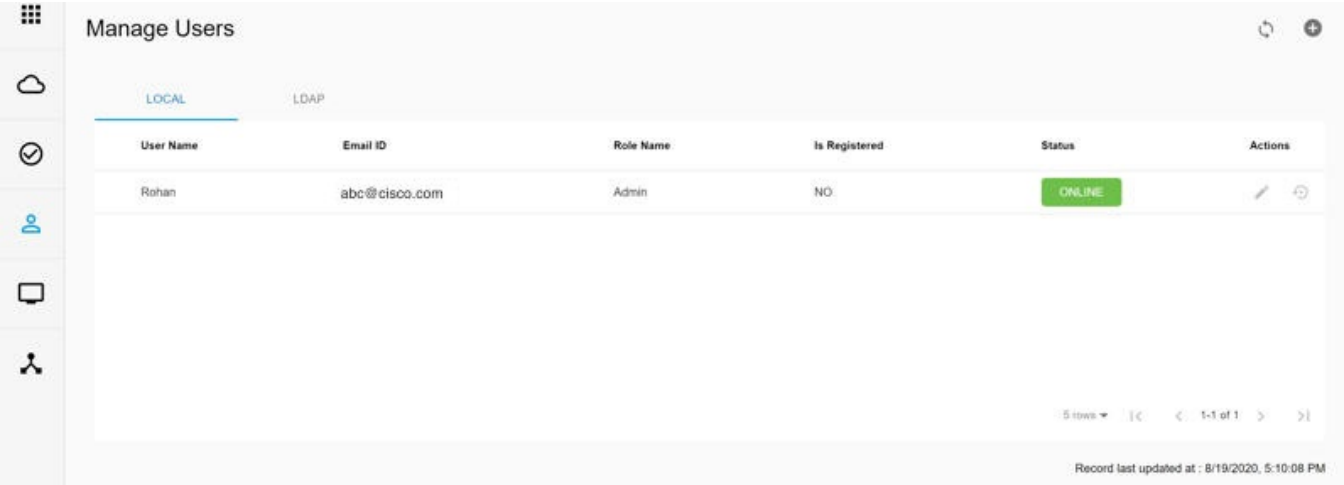
NG Pod User Administration

- Managing Users
  - Adding Pod Users
  - Editing User Privileges
  - Revoking User Privileges

Cisco VIM UM offers users such as Pod Admin(s) and Pod Users to manage users and roles associated with them.

Managing Users

Allows you to add the users and view all the users associated with the pod. To check the online status of all the users, click **Refresh** icon on upper right corner.

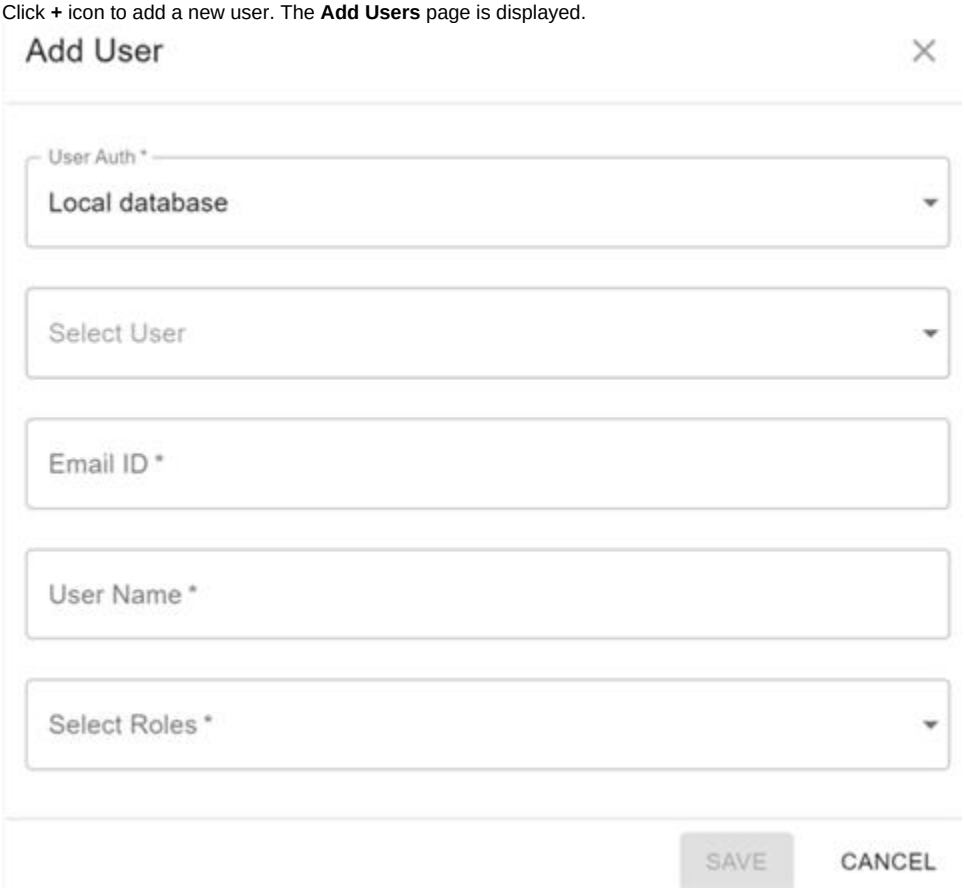


Adding Pod Users

To add a new user, follow the below steps:

1. Log into UM as pod user.
2. Click on the pod, to view the pod dashboard.
3. From the left navigation pane, choose **User Management** >**Manage Users**.

4. Click **+** icon to add a new user. The **Add Users** page is displayed.



5. Enter the following:

Field Name	Field Description
User auth	Select the database from the drop-down for the new user. This option is enabled only if LDAP mode is True.
Select User	While adding new pod-user, a list of users having pod-user permissions is displayed in the user-registration form. This list is available only, when DISPLAY_ALL_POD_USERS is set as True.
Registration Type	The registration type can be user/group, only when User Auth is LDAP. Following fields are available when the Registration Type is *Group*: • Group Dn – Enter the distinguished name of the LDAP group. • Group Name – Enter the name of the LDAP group.
Email ID	Enter the email ID of the user.
User Name	Enter the username if the user is not registered. For registered users, the username gets auto-populated.
Role	Select the role from the drop-down list.

6. Click **Save**.

⚠ NG UI supports only two types of roles such as *Admin* and *Viewer.* A user with *Admin* role have full pod access and can perform all operations on the pod. A user with *Viewer* role can only view the pod information and operation logs.

Editing User Privileges

User with *Admin* role can edit other user's permission for a specific pod.

To edit user's permission:

1. On the **Manage Users** page, click **Edit** icon under **Action** column for the required user.
2. Update the permission.
3. Click **Save**. The grid gets refreshed automatically.

Revoking User Privileges

User with *Admin* role can revoke other user's permission for a specific pod.

To revoke users:

1. On the **Manage Users** page, click **Revoke** icon under **Action** column for the required user. A confirmation pop up appears.
2. Click **Proceed** to continue.

> ⚠ Self-revoke is not permitted. After revoking the user, if the user is not associated with any other pod, then the revoked user is auto-deleted from the system.

NG UM Dashboard

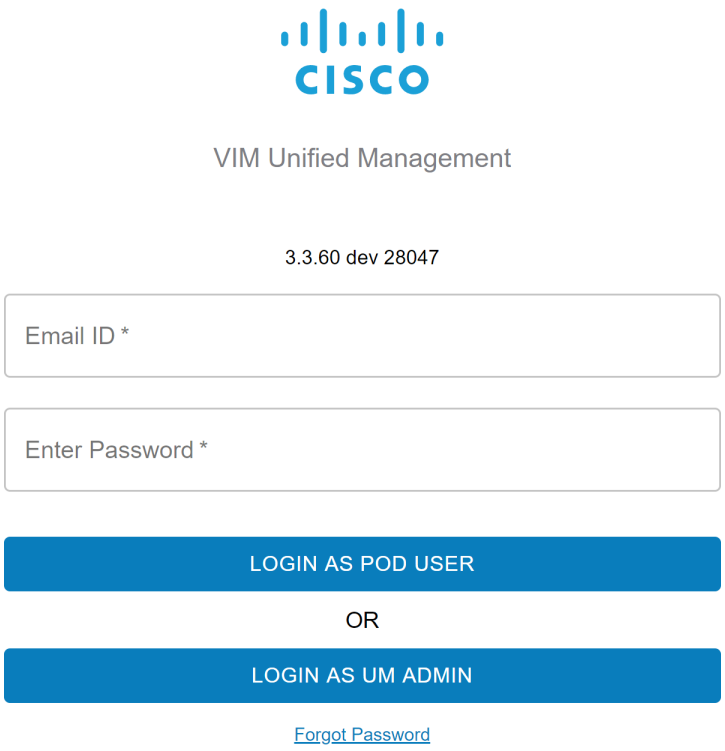
UM Dashboard

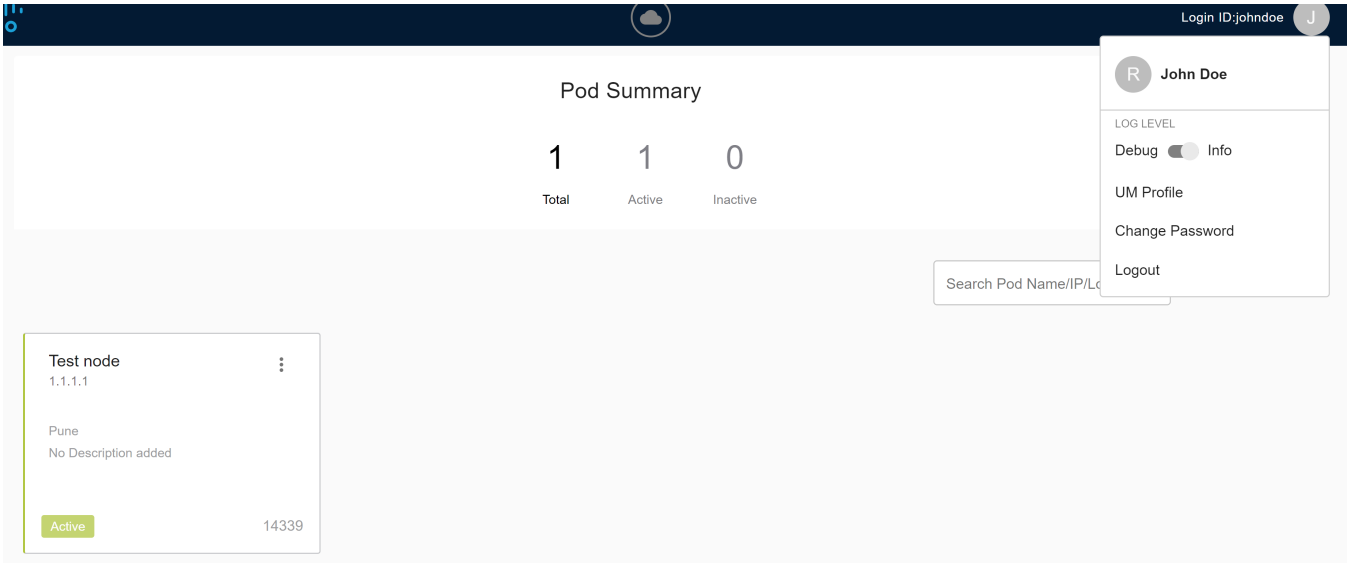
- Overview
- UM Dashboard
- Blueprint
    - Deployed Cloud Status
    - Deployed Blueprint Details
    - Pod Operation Details

Overview

UM Admin also known as Super Admin is created by default, after the UM node installation. The UM admin have access to UM Dashboard and can add additional Pod Admins and UM Admins. Two ways to access UM Dashboard are:

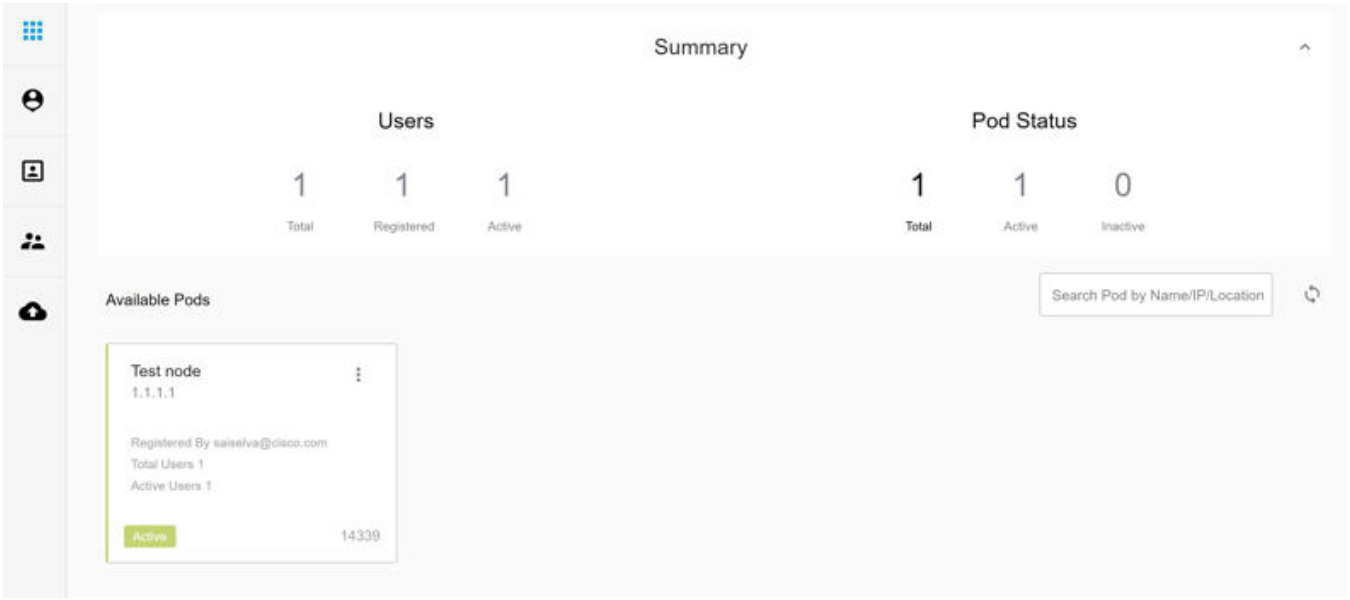
- On the **Login** page, enter **Email ID** and **Password** of the UM Admin and click **LOGIN AS UM Admin**. On successful authentication, UM dashboard page is displayed.
- If you have logged in as pod user, but have privileged access to UM dashboard, you can switch to UM profile dashboard, by clicking **Login ID** on the top right corner of the dashboard page and choosing **UM Profile**.





UM Dashboard

The UM Dashboard displays a list of pods with its pod name, description, IP address, location, status, Cisco VIM pod release version, registration details, and number of registered/active users. You can search for a pod using name, IP and location in the search option.



The **Summary** pane at the top displays the following information:

Users

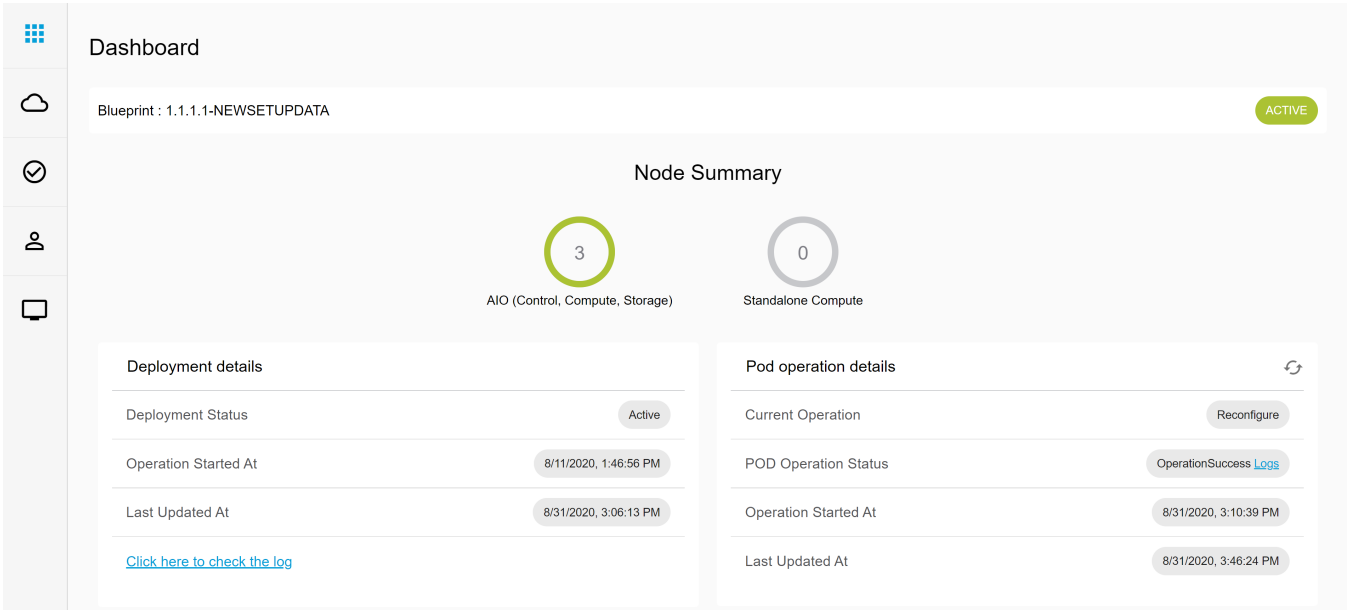
- Total – Total number of registered users who are associated with at least one pod.
- Registered – Number of users who have completed the registration and are associated with at least one pod.
- Active – Number of online users who are currently logged in.

Pod Status

- Active - Number of pods with health status OK (Example: pod management node health is good).
- Inactive - Number of pods whose health status is not good (Example: management node health of the pod is not good).
- Total - Total number of pods registered in the system.

To check the health of the pod, click **Get health of current nodes icon (spin)**.

After selecting a pod, the **Dashboard** of that particular pod is displayed.



Blueprint

Blueprint pane provides the blueprint name and the statuses of the blueprints and various installation stages. The status is *Success*, *Failed,* or *Not Run*. To navigate between various installation stages, click **Next and Previous**.

Deployed Cloud Status

Following are the different types of cloud status on the pod.

- Active (Green): Indicates that the cloud is deployed without any failure.
- Failed (Red): Indicates the failure of the cloud deployment.
- Not Available (Gray): Indicates that the cloud is not deployed on the pod.

Deployed Blueprint Details

Provides information about the deployed blueprint including deployment status, operation start time, operation update time, and a link to the last operation log. During cloud installation failure, the name with keyword regarding component failure is visible as **Deployment Status**.

Pod Operation Details

Provides information about the current operation details, pod operation status, and operation start/update time. **Refresh** icon at the top right corner of **Pod Operation** pane facilitates you to fetch the latest operation status from the pod.

NG Pod Registration/Management

NG Registering/Managing Pods on UM

- Registering Pod
- Editing Pod
- Deleting Pod
- Uploading Root Certificate
- Updating REST Password

Before you begin

Only Pod Admins are eligible to register a new pod. UM Admin can register Pod Admins from the **UM Pod Administrator** page.

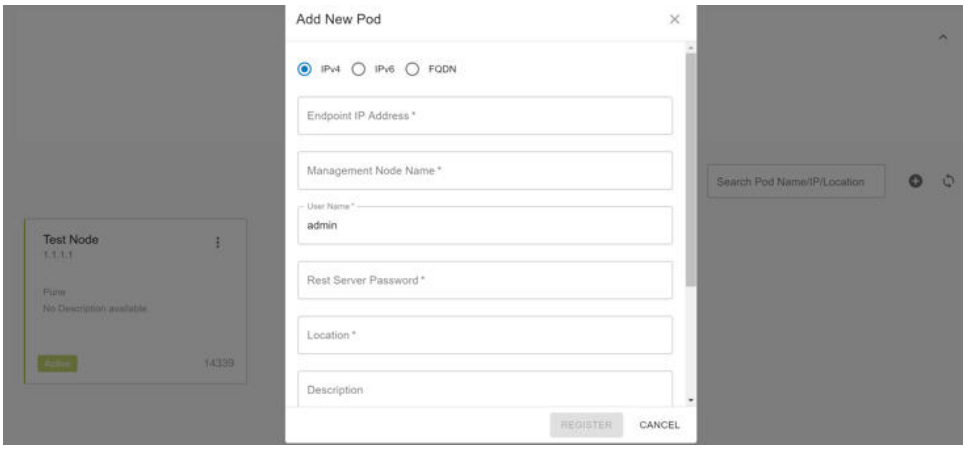
For a UM Admin to register a Pod Admin, follow the below steps:

1. Log into UM as UM Administrator.
2. Navigate to Pod Administrator and click **Add Pod Admin**. For more details, see NG Pod Administrator.

Registering Pod

To register new pod, follow the below steps:

1. On the login page, enter the **Email ID** and **Password** of the Pod Admin.
2. Click **Login as Pod User**. You will be redirected to the landing page.
3. Click **Add New Pod** to register a pod. The **Add New Pod** window is displayed.



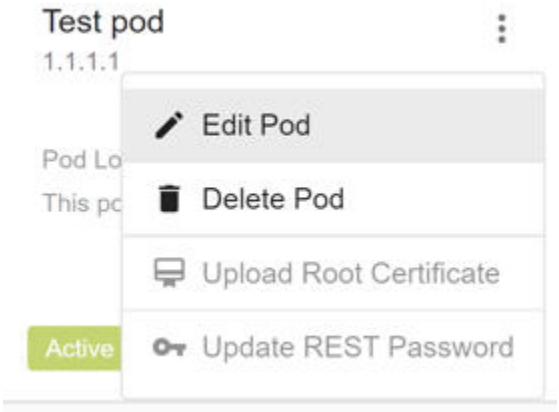
4. Click **IPv4**, **IPv6**, or **FQDN** radio button to choose the management node endpoint type.
5. Enter **Endpoint IP Address** and **Rest Server Password.** You can get the password from */opt/cisco/ui_config.json* file on Cisco VIM management node.
6. Enter **Management Node Name**, **Location** and **Description** as needed.
7. Click **Browse** to select the root CA certificate.
8. Click **Upload Certificate** to upload the selected root CA certificate.
9. Click **Register** to start the pod registration. The newly created pod appears on the landing page.

UM Admins and Pod Admins can edit or delete pods, update Rest API password, and Rest API certificate from the landing page.

Editing Pod

To edit the registered pod details, follow the below steps:

1. On the login page, enter the **Email ID** and **Password** of the Pod Admin.
2. Click **Login as Pod User**. You will be redirected to the landing page.
3. Select the required pod.
4. On the top right of the selected pod, click **More Options** icon represented by three vertical dots.



5. Click **Edit Pod**. The **Edit Pod** window is displayed.
6. Change the **Pod Name** and **Description** of the registered pod.
7. Click **Proceed** to save the changes.

Deleting Pod

To delete the registered pod details, follow the below steps:

1. On the login page, enter the **Email ID** and **Password** of the Pod Admin.
2. Click **Login as Pod User**. You will be redirected to the landing page.
3. Select the pod to be deleted.
4. On the top right of the selected pod, click **More Options** icon represented by three vertical dots.
5. Click **Delete Pod**. A confirmation pop-up with **Proceed** button is displayed.
6. Click **Proceed** to delete the pod.

> ⚠ The deletion of a pod is allowed only for the user who has registered that particular pod.

Uploading Root Certificate

Unified Management node communicates with Rest API service running on the management node. All the clients require Rest API password and certificate for authenticating and sending requests to Rest API over a secured channel. If Rest API certificate is renewed on the Cisco VIM management node, follow the steps mentioned below to upload root certificate for a registered pod:

1. On the login page, enter the **Email ID** and **Password** of the Pod Admin.
2. Click **Login as Pod User**. You will be redirected to the landing page.
3. Select the pod.
4. On the top right of the selected pod, click **More Options** icon represented by three vertical dots.
5. Click **Upload Root Certificate**.
6. Browse and upload the root CA certificate.
7. Click **Save**.

Updating REST Password

If Cisco VIM pod Rest API password is modified, follow the steps mentioned below to update REST password for a registered pod.

1. On the login page, enter the **Email ID** and **Password** of the Pod Admin.
2. Click **Login as Pod User**. You will be redirected to the landing page.
3. Select the pod.
4. On the top right of the selected pod, click **More Options** icon represented by three vertical dots.
5. Click **Update REST Password**.
6. Enter the new password and click **Save.**

NG Monitoring Pod Status

Monitoring Pod Status

The Unified Management application manages the pods and displays the pod management action status with a cloud icon.

The following table displays a summary of the pod operation, the corresponding cloud-icon color, and the pod status.

Pod Operation	UM Icon-Color	Pod Status
Active cloud with no failures	Green	Active
Cloud installation or pod management operation is in progress	Blue	In-progress
Software update (auto) rollback is failed	Red	Critical Warnings
Pending commit post software update	Amber	Warning
Reconfigure failed (for any operation)	Red	Critical Warning
Update, commit, or Rollback failed	Red	Critical Warning
Power management operation fails	Amber	Warning
Management not reachable	Red	Not Reachable

NG Context Switching Between Pods

Context Switching Between Pods

Cisco VIM UM permits you to switch between two or more pods for a particular node. You can be an admin for one or more pods, and a user for some other pods simultaneously. The ability to access multiple pods helps you to maintain context and yet scale from a pod management point of view.

Following are the two ways to switch from one pod to another.

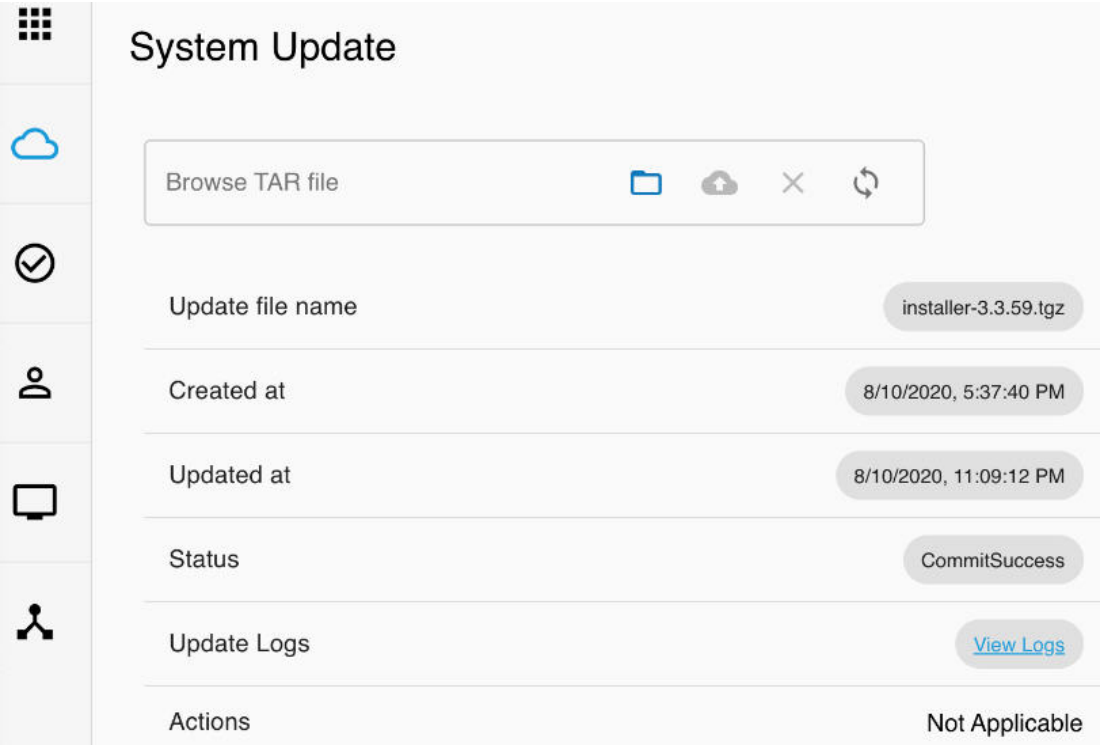
- Context Switching icon: **Context Switching** Icon is situated on the top right corner from the tool tip of the UI. Click **Context Switching** icon to view all the pods that you can access. The pod with a red dot indicates that the REST password entered during management node registration does not match with the current REST password for that particular node. In this case, the Pod admin or user must reach out to UI admin to update the password for that node. UI admin updates the password from **Manage Pods** in **Unified Management UI admin** portal.
- Switch between management nodes: Use dashboard to switch between management nodes. You can see all the pods in the table and navigate to any pod using a single click. If mouse pointer changes from hand or cursor to a red dot sign, it indicates that the REST Password entered during management node registration does not match with the current REST Password for that particular node.

NG Cisco VIM Pod Software Update

Cisco VIM Pod Software Update via NG UM

The software management of Cisco VIM pods includes software update, reconfiguration of OpenStack services and password, and so on.

As part of the cloud lifecycle management, Cisco VIM can bring in patches like bug fixes related to code, security, and so on. The software update of the cloud is done by uploading a valid tar file, following the initiation of a system update from the Unified Management.



To update Cisco VIM software, follow the below steps:

1. Log into UM as pod user and select the pod from the landing page.
2. From the left navigation pane, choose **Pod Setup > System Update**.
3. Click **Browse** icon and select the valid tar file.
4. Click **Upload and Update** icon. A message **System Update has been initiated** is displayed.
5. To view the progress of update, check **Update Logs**.

> ⚠ During software update, the following operations are not permitted:
>
> - Reconfigure
> - System update
> - Pod management

6. Once the update is completed successfully, choose one of the following options:

    a. Commit: To proceed with the update.
    b. Rollback: To cancel the update.

> ⚠ The **Cloud** icon turns amber, after the successful completion of a software update.

7. If update fails**,** *Auto-RollBack* is initiated automatically and Cisco VIM pod is reverted to original version.
8. If auto-rollback fails during software update initiated through Unified Management UI, and the **Cloud** icon turns red, the administrator must contact *Cisco TAC* for support. Do not re-try to update or delete the new or the old installer workspace.

    When commit or rollback is completed successfully, all links are enabled in the UI.

> ⚠ If VMs are running on the compute nodes, those nodes are not rebooted during update. Ensure that you reboot the compute nodes after a successful commit operation.

NG Managing Passwords

Reconfigure Openstack and CIMC Passwords

- Reconfiguration of OpenStack Passwords
  - Regenerate All Passwords
  - Custom Password
- Reconfiguration of CIMC Password

You can change the OpenStack password and CIMC password on Day 1 through reconfiguration, only for the pods that are successfully deployed.

Reconfiguration of OpenStack Passwords

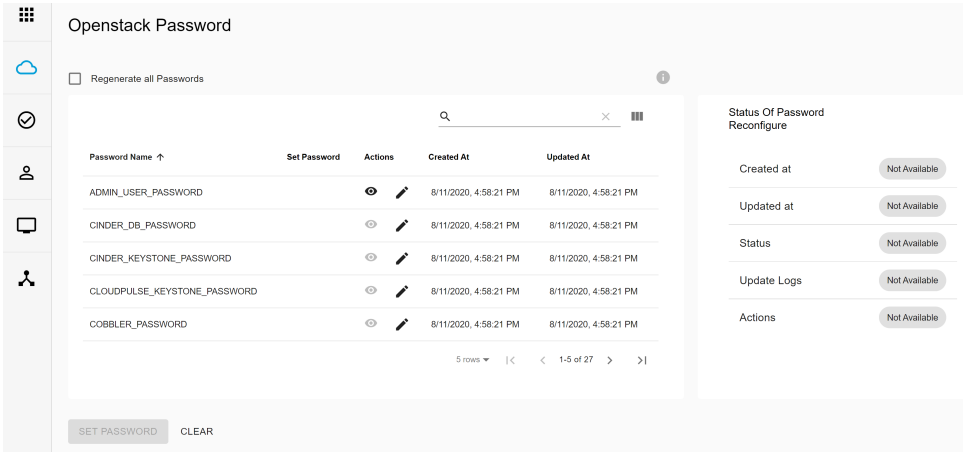
The OpenStack services deployed on control/compute/storage nodes, infrastructure services, and logging and monitoring services deployed on management node are installed on Day 0 with auto-generated passwords. You can set custom passwords or change the passwords of all these services if required.

Regenerate All Passwords

With this option, new passwords are auto-generated for each service and reconfigure operation is performed on the cloud.

To regenerate password, follow the below steps:

1. Log into UM as pod user and select the pod.
2. From the left navigation pane, choose **Pod Setup > OpenStack Password**.



3. Select **Regenerate All Password**.
4. Click **Set Password** to start the reconfiguration. The logs and status of the operation are displayed.

Custom Password

With this option, user can choose custom passwords for some or all Openstack Services. Custom passwords must conform to certain criterias laid out for the product. Following are the steps from UI:

1. Log into UM as pod user and select the pod.
2. From the left navigation bar, choose **Pod Setup > OpenStack Password**
3. Click **Edit** icon and set custom passwords.
4. Click **Set Password** to start the reconfigure operation. The logs and status of the operation are displayed.

Reconfiguration of CIMC Password

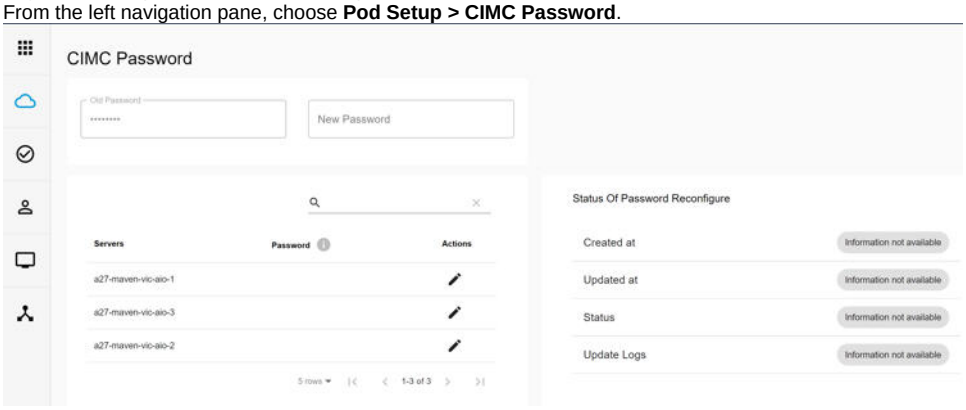
Cisco VIM allows you to update the *cimc_password* in the CIMC-COMMON section, and/or the individual *cimc_password* for each server and then run the update password option.

You need to match the following password rule to update the password:

- Must contain at least one lowercase letter.
- Must contain at least one uppercase letter.
- Must contain at least one digit between 0 to 9.
- One of these special characters !$#@%^-_+=*&
- Your password must be 8 to 14 characters long.

To change CIMC password, follow the below steps:

1. Log into UM as pod user and select the pod.
2. From the left navigation pane, choose **Pod Setup > CIMC Password**.



3. You can reconfigure the CIMC password at global level by adding new CIMC_COMMON password. To reconfigure CIMC password of individual servers, click **Edit** icon under **Actions** column for respective servers.
4. Click **Update Password** to initiate reconfiguration.

NG Day 2 Reconfigure/Enablement

Reconfiguration/Enablement of Features on Day 2

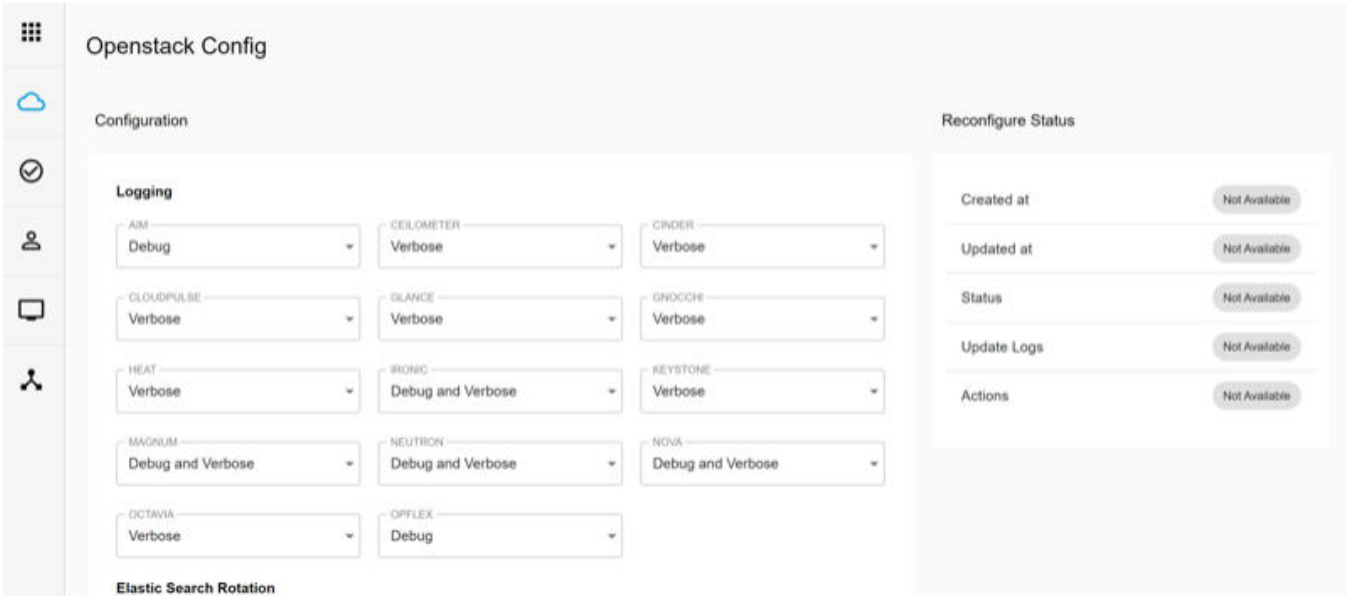
- Overview
- Reconfigure OpenStack Service Log Level, External LB VIP TLS Certificates, and ELK Configurations
- Reconfigure Management TLS Certificates
- Reconfiguration of Optional Services

Overview

To manage Day n operations of the successfully deployed pod, you can enable or reconfigure the OpenStack service log level, ELK configuration, and other optional features. However, often UM cross-launches the relevant services, by delegating the actual rendering to the individual services.

Reconfigure OpenStack Service Log Level, External LB VIP TLS Certificates, and ELK Configurations

You can reconfigure the OpenStack services log level, external load balance virtual IP (LB VIP) TLS certificates, ELK stack configuration, log rotation frequency, and Nova CPU/RAM allocation ratio, after the cloud is deployed.



Following are the steps to reconfigure above mentioned:

1. Log into UM as pod user and select the pod on the landing page.
2. From the navigation pane, choose **Pod Setup > OpenStack Config**.
3. Edit the form shown on the page, as needed. For example, to update the TLS certificate, enter path of new TLS certificate.
4. Click **Set Config** to commence the reconfiguration.

During reconfiguration, all other pod management activities are disabled. On successful reconfiguration, all links are enabled again. If the reconfiguration of OpenStack services fails, all subsequent pod management operations are blocked.

To resolve the situation through CLI, contact Cisco TAC.

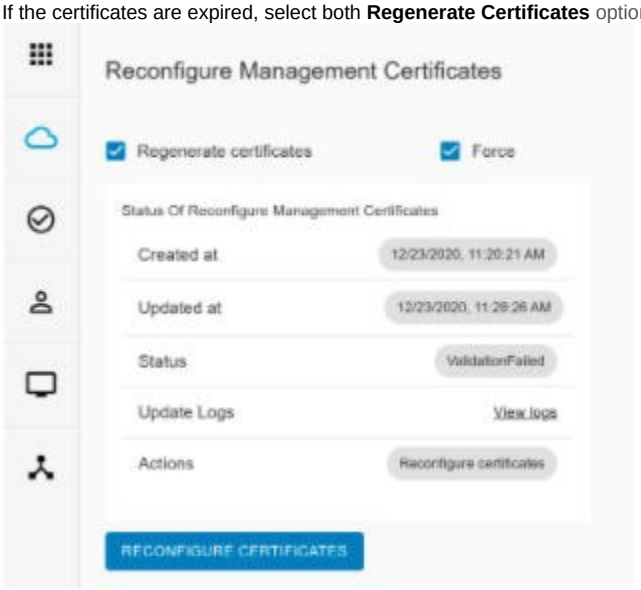
Reconfigure Management TLS Certificates

From Cisco VIM 4.0.0 release, all the traffic flowing on the management network is secured with the SSL/TLS protocol, when MGMT_TLS is enabled in the *setup_data.yaml.* With this configuration, self-signed x509 certificates are created for every server in the pod and signed by a self-signed Root CA authority. Since these certificates are valid only for few months, there is a need to rotate these certificates just before they expire.

Following are the steps to reconfigure management TLS certificates:

1. Log into UM as pod user and select the pod on the landing page.
2. From the navigation pane, choose **Pod Setup > Reconfigure Management Certificates**.

3. If the certificates are expired, select both **Regenerate Certificates** option and **Force** option, otherwise select only **Regenerate Certificate** option.

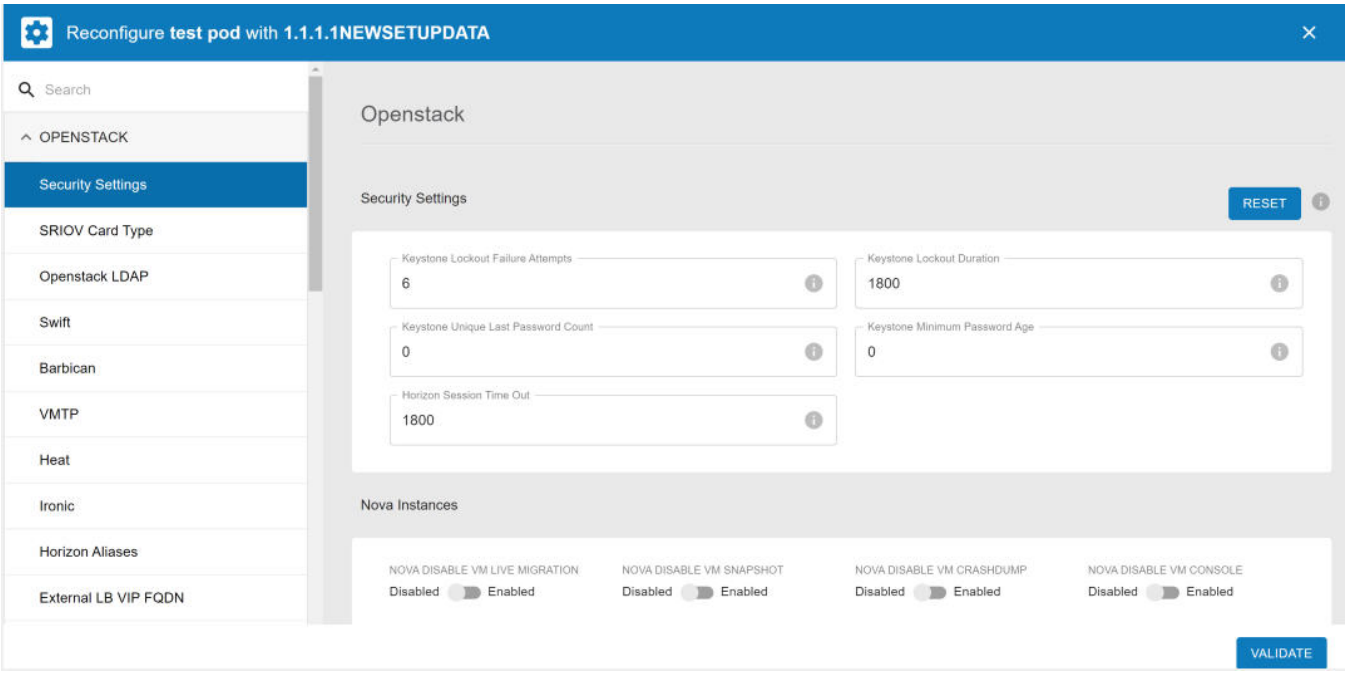


4. Click **Reconfigure Certificates** to start the operation.

When the management TLS certificates expire, Cisco VIM cloud control plane becomes non-functional. Hence, ensure that you rotate the certificates before they expire. In rare scenario if certificates expire, Cisco VIM still provides a mechanism to rotate the certificates and recover the cloud.

Reconfiguration of Optional Services

Cisco VIM offers optional services such as Heat, NFVBench, CVIM MON, and so on, that can be enabled after the deployment of pod. You can enable selected services or all the services in one-shot.



To enable optional services, follow the below steps:

1. Log into UM as pod user and select the pod from the landing page.
2. From the left navigation pane, choose **Pod Setup** > **Blueprint Management.**
3. Select the blueprint that is deployed.
4. Click **Reconfigure** icon. All reconfigurable parameters of Cisco VIM are displayed.
5. Choose the right services and update the fields with the right values.
6. Click **Validate** to begin offline validation.
7. Once offline validation is successful, click **Initiate Reconfigure** to commence the reconfiguration.

During reconfiguration, all other pod management activities are disabled. On successful reconfiguration, all links are enabled again. If reconfiguration fails, all the subsequent pod management operations are blocked.

To resolve the situation through CLI, contact Cisco TAC.

> ⚠ All reconfiguration features contain repeated re-deployment option set to True or False. If repeated re-deployment is set to True, you can change the values associated with the feature. If repeated re-deployment is set to False, you can deploy the feature only once. Un-installation of the feature is only supported as an exception.

List of reconfigurable optional services is given below:

Category	Name
OpenStack	Security Settings (Cloud Settings)
	SRIOV card type
	OpenStack LDAP
	VLAN Ranges
	Swift
	Barbican
	VMTP
	Heat
	Ironic
	Horizon Aliases
	External LB VIP FQDN
	Neutron Base MAC Address
	Enable Read-Only OpenStack Admin
	External LB VIP TLS
	Ceph Tuning
	Custom Manufacturer Tag
	Advanced OpenStack Roles
	Cinder RBD Flatten Volume from Snapshot
Network Settings	IP proxy and servers
	Network
Logging and Monitoring	ES remote backup
	System log export
	Kibana
	CVIM monitoring
Infrastructure	SSH banner
	Installation mode
	Cobbler
	NFVBench
	Auto-backup
	ESC Priv
	Management node ExtAPI FQDN
	Pod name
	Registry Setup
	Management network over TLS

Infra Security	Password management
	SSH access options
	VIM admins
	VIM LDAP admins
	Vault
	IPA info

To enable the reconfigurable optional features, see the respective sections

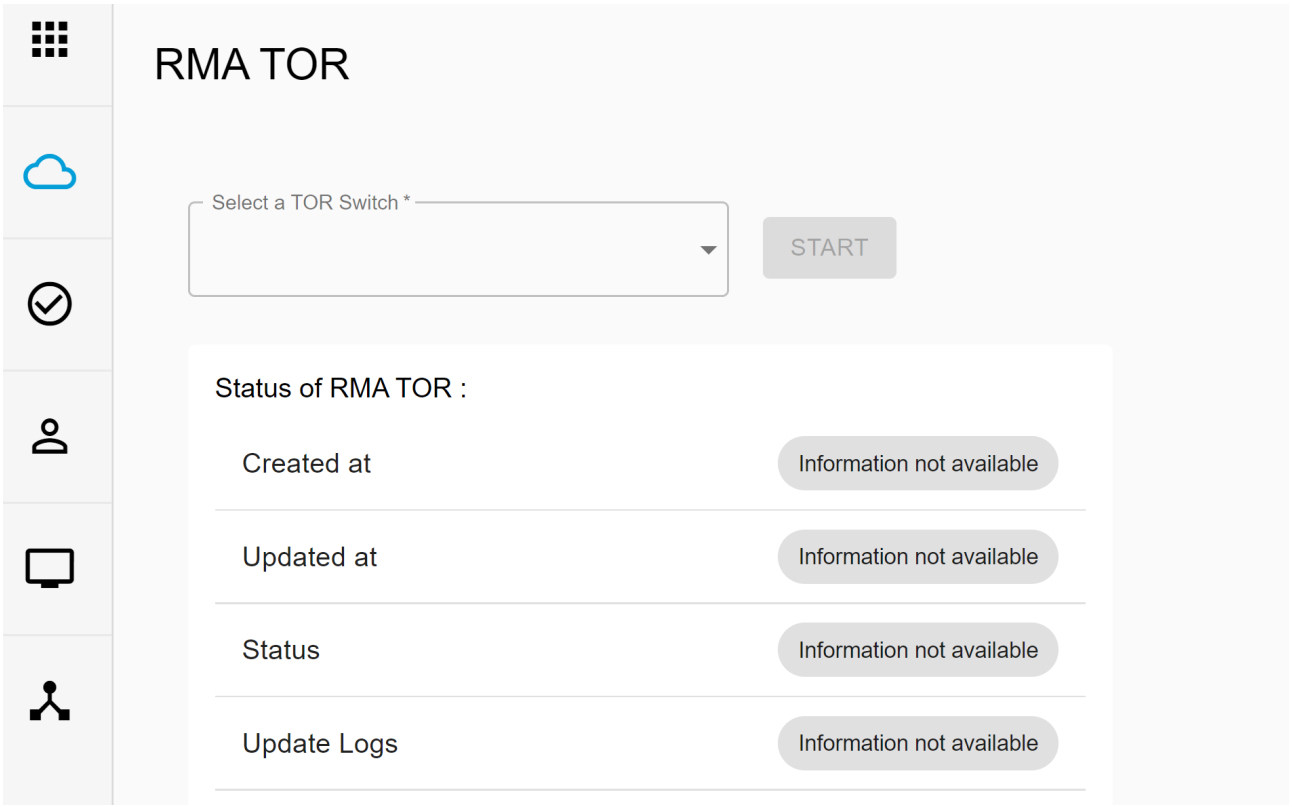
NG CVIM RMA for Auto-TOR

Return Merchandise Authorization for Auto-ToR from UM

When Cisco VIM cloud uses auto-ToR configuration to manage switch ports, you need to replace the existing switches if one malfunctions.

Consider the following assumptions made during RMA of ToR with auto-ToR configuration:

- When a switch is getting RMAed, it is in a virtual port-channel (vPC) mode with another switch to support full switch redundancy.
- Administrator is responsible for manual configuration of the spine connection and L3 Out for the ToR.
- On the new ToR, ensure that you use same server ports.
- Ensure that new ToR name and login credentials are same as the one that you are replacing.



To perform RMA of ToR from UM Node, follow the steps mentioned below:

1. Log into UM as pod user and select the pod from the landing page.
2. From the left navigation pane, choose **Pod Setup > RMA TOR**
3. Select a **ToR Switch** from the dropdown.
4. To initiate RMA ToR operation, Click **Start**. You can view the operation status and logs on the same page.

NG Cisco VIM Validation Tools

Validation Tools

- Overview
- Cloud Sanity
- CloudPulse
- VMTP
- NFVBench
    - NDR/PDR Test
    - Fixed Rate Test

Overview

The validation tools are applicable only to a pod, which is successfully deployed. You can use these validation tools for :

- Testing control plane and data plane sanity
- Finding data plane throughput
- Checking the cloud health

Cloud Sanity

In Cisco VIM, Cloud Sanity is available to run on-demand health checks for all servers in the OpenStack cloud. Cloud sanity testing is divided into 5 categories namely:

Control Tests:

1. Controller server uptime and network interface reachability
2. Docker container status and versions
3. OpenStack control plane services health
4. MariaDB, RabbitMQ cluster health

Compute Tests:

1. Compute servers uptime and network interface reachability
2. Docker container status and versions
3. Neutron agents and Nova components health check

Ceph-mon Tests:

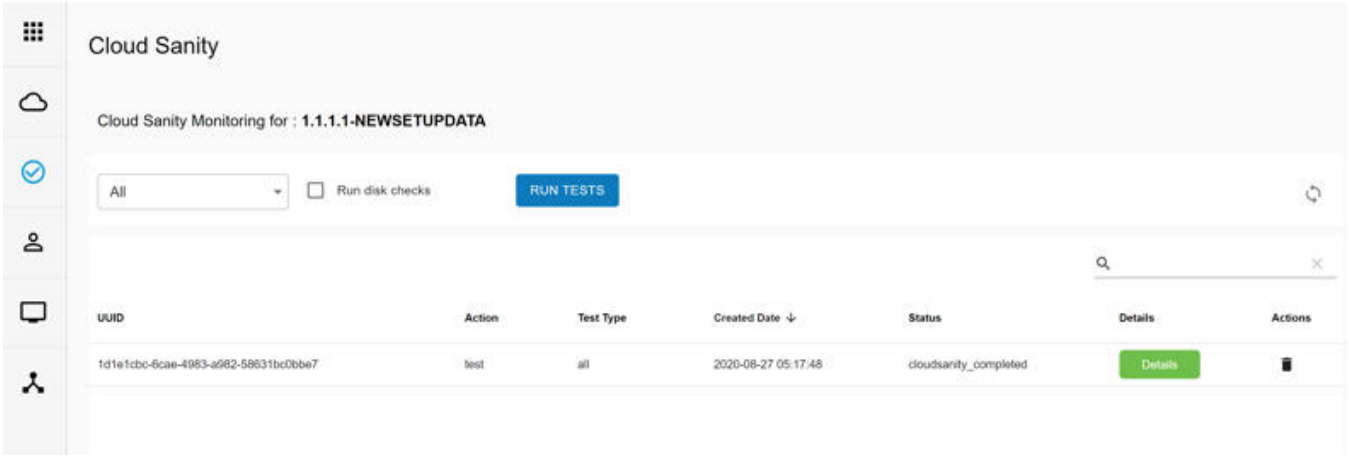
1. Check Ceph-mon container health and version
2. Ceph cluster health
3. Ceph OSD statistics

Ceph OSD Tests:

1. Storage server uptime and network interface reachability
2. Ceph OSD health and other metrics

Management Tests: Management node container status and version check.

There is an additional option to run all tests at once. Also by default, disk checks are disabled, as they may cause performance impact on the cloud. You can enable disk check flag and run cloud sanity tests when needed.



CloudPulse

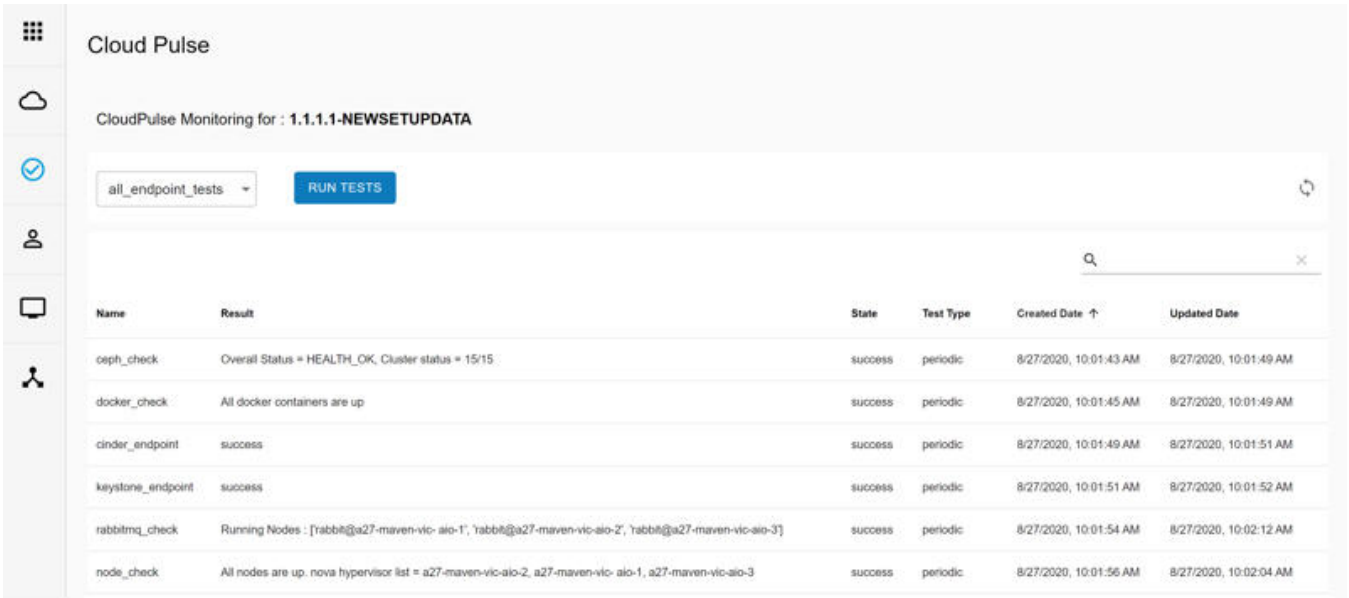
In Cisco VIM, an integrated tool called CloudPulse periodically checks the cloud services endpoint. You can get the test results from **CloudPulse** link. Alternatively, you can run these API endpoint tests on-demand and fetch the test results by clicking **Refresh** icon present above the search option.

Endpoints Tests:

1. cinder_endpoint
2. glance_endpoint
3. keystone_endpoint
4. nova_endpoint
5. neutron_endpoint
6. all_endpoint_tests

Operator Tests:

1. rabbitmq_check
2. galera_check
3. ceph_check
4. node_check
5. docker_check
6. all_operator_tests

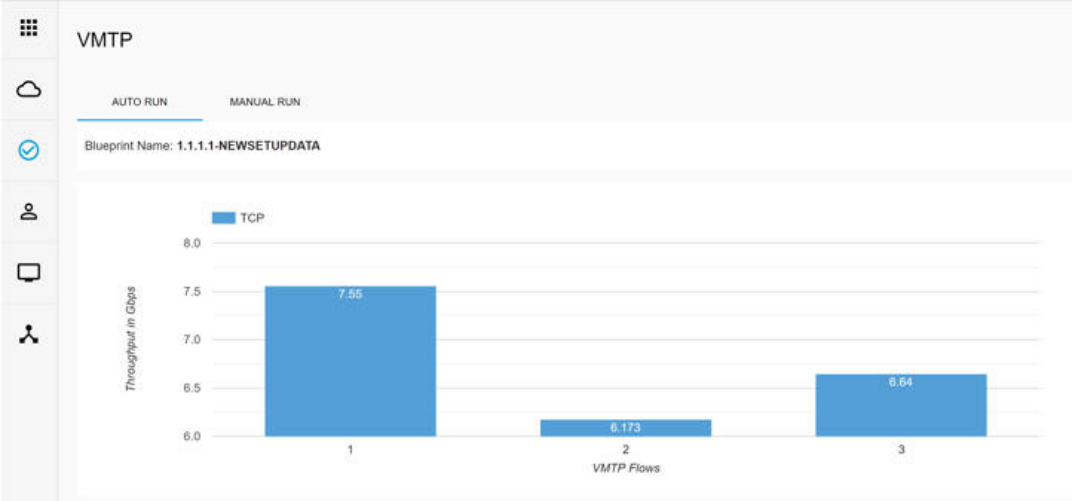


VMTP

Cisco VIM provides an integrated data and control plane test tool called VMTP. VMTP helps you to test the cloud at any given time.
VMTP is divided into two sections:

- **Auto Run**: Shows the results of VMTP which is run during the cloud deployment (Blueprint installation).
- **Manual Run**: To run VMTP on-demand, click **Run Test**.

> If VMTP stage is skipped or not run during blueprint installation, VMTP link is disabled.

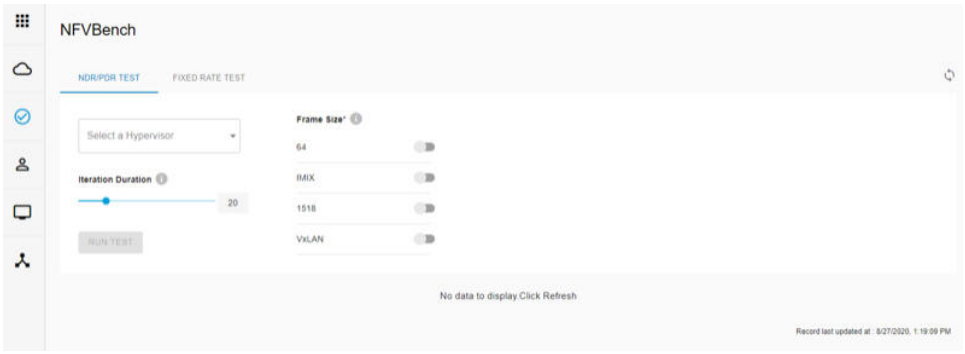


NFVBench

You can execute **NFV Bench** test through Cisco VIM Unified Management. On a pod running with Cisco VIM, you can run either fixed rate test or NDR /PDR test. As the settings and results for the test types differ, the options to run these tests are presented in two tabs, with its own settings and results.

NDR/PDR Test

1. Log into UM as pod user and click the required pod.
2. From the left navigation pane, choose **Validation Tools** > **NFVBench.**



3. Click **NDR/PDR Test** tab.
4. Enter the following:

Name	Description
Iteration Duration	Select duration from 10 to 60 sec. Default is 20 sec.
Frame Size	Select the correct frame size to run.
Run NDR/PDR test	Displays each type of test with its own settings and results once NDR/PDR test is finished.

Fixed Rate Test

1. Log into UM as pod user and select the pod.
2. From the left navigation pane, choose **Validation Tools > NFVBench.**

3. Click **Fixed Rate Test** tab.



4. Enter the following:

Name	Description
Rate	Select right configuration pps or bps from drop-down and enter values: For pps: minimum: 2500pps; maximum: 14500000pps (=14.5Mpps); default: 1000000pps (=1Mpps) For bps: minimum: 1400000bps; maximum: 10000000000bps (=10Gbps); default: 1000000000 (=1Gbps)
Iteration Duration	Select duration in the range of 10-60Sec. Default is 20sec.
Frame Size	Select the right frame size (64,IMIX,1518) to run.
Run Fixed Rate Test	Displays each type of test with its own settings and results, once fixed rate test is finished,

NG Cisco VIM Operational Tools

Operational Tools

- Pod Management
- Cisco VIM Logging
- CVIM Monitoring
- NFVI Monitoring
- Horizon
- Cluster Recovery

You can use the following operation tools to manage Day n operations of the pod that is successfully deployed.

Pod Management

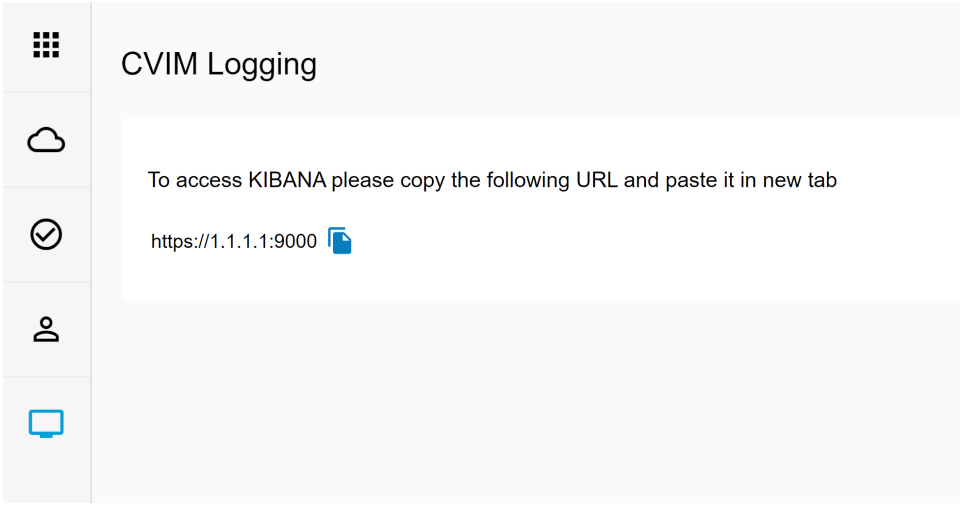
One of the key aspects of Cisco VIM is that it allows the admin to perform pod life-cycle management from a hardware and software perspective. The nodes of a given pod corrupt at times and Cisco VIM provides the ability to add, remove, or replace nodes, based on the respective roles with some restrictions. The following operations are allowed on a running pod:

1. Add or remove storage nodes: You can add one node at a time, when Ceph is run as a distributed storage.
2. Add or remove computes nodes: You can replace N-compute nodes simultaneously. However, at any given point, at least one compute node must be *Active.*
3. Replace control nodes: Double-fault scenarios are not supported, while the replacement of one controller at a time is supported.

For more information on Cisco VIM pod management operations, see NG Managing Hardware and NG Managing Power

Cisco VIM Logging

Cisco VIM uses EFK stack (Elasticsearch, Fluentd and Kibana) for collecting OpenStack service logs from all nodes and storing it in Elasticsearch on the management node. You can view these logs on Kibana Dashboard.

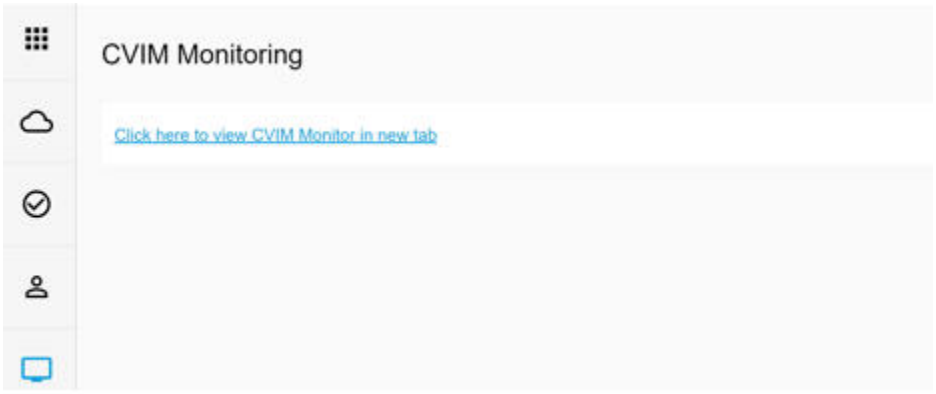


To launch Kibana Dashboard from UI, do the following:

1. Log into UM as pod user.
2. Select the pod**.**
3. From the navigation pane, choose **Operational Tools**
4. Click **CVIM Logging**. Kibana URL is displayed.
5. Use C*opy to clipboard* functionality to copy Kibana URL and open it in another tab. The **Authentication Required** browser pop-up is displayed.
6. Enter the **Username** as *admin.*
7. Enter the **KIBANA_PASSWORD** password obtained from **OpenStack Password** page.

CVIM Monitoring

Cisco VIM provides a monitoring solution comprising Prometheus, Alert Manager, and Grafana Dashboard. The stack components are installed on Cisco VIM management node, when local CVIMMON is installed. Unified Management node UI has a link to cross-launch Grafana Dashboard in a different browser. Cisco VIM monitoring link is enabled only, if the deployed blueprint has CVIMMON section enabled.



To launch Grafana Dashboard from UI, do the following:

1. Log into UM as pod user.
2. Select the pod.
3. From the left navigation pane, choose **Operational Tools**.
4. Click **CVIM Monitoring**.
5. Click the link **Click here to view CVIM Monitor in new tab.** You will be redirected to Grafana Dashboard.
6. Enter the **username** as admin.
7. Enter the **CVIM_MON_PASSWORD** password obtained from **OpenStack Password** page.

NFVI Monitoring

Uses Cisco VIM blueprint file configuration to monitor Cisco VIM cloud using a third-party monitoring solution called *Zenoss*.

The NFVI monitoring link is enabled only, if deployed blueprint contains NFVI monitoring configuration.

1. Log into UM as pod user**.**
2. Select the pod.
3. From the left navigation pane, choose **Operational Tools**.
4. Click **NFVI Monitoring**.
5. Click the link **Click here to view NFVI monitoring**. You will be redirected to **NFVI monitoring** page.

Horizon

Horizon is the canonical implementation of OpenStack dashboard, which provides a web-based user interface for OpenStack services including Nova, Swift, and Keystone.



1. From the navigation pane, choose **Operational Tools** > **Horizon**.
2. Click **Click here to view Horizon in new tab**. You will be redirected to Horizon landing page in a new tab.

Cluster Recovery

To recover the OpenStack cluster after an outage, follow the below steps:

1. From the left navigation pane, choose **Operational Tools** > **Cluster Recovery**.



2. If you want to run disk checks, select **with run-disk-checks ?** option, otherwise proceed with Step 3.

> ⚠ The cluster recovery execution time is more, when run-disk checks are opted.

3. Click **RUN CLUSTER RECOVERY**. A confirmation message **The execution of cluster recovery will have an impact on cloud control plane** is displayed with **Proceed** and **Cancel**.
4. Click **Proceed**. The **Status of Cluster Recovery** is displayed.

> ⚠
>
> a. **RUN CLUSTER RECOVERY** button is disabled when the cluster recovery operation is running. It will be enabled once operation is completed with either *Success* or *Failed* status.
> b. During cluster recovery, other cloud operations like CIMC Password, Cloud Sanity, Cloud Pulse, and CVIM Monitoring are disabled.
> c. The **Cluster Recovery** under the **Operational Tools** is disabled when any other cloud operation is running.

NG Managing Hardware

Managing Hardware

- Overview
- Searching Compute and Storage Nodes
- Pod Management Operations
- Managing Storage Nodes
  - Adding Storage Node
  - Deleting Storage Node
- Managing Compute Nodes
  - Adding Compute Node
  - Deleting Compute Node
- Managing Control Nodes
  - Replacing Control Node

Overview

You can perform the pod management operations such as add/remove computes, add/remove storage, and replace controller on Cisco VIM cloud through UI. You can add or remove multiple compute nodes to or from Cisco VIM cloud in parallel. However, operations like replace controller and add/remove storage are restricted to one server at a time.

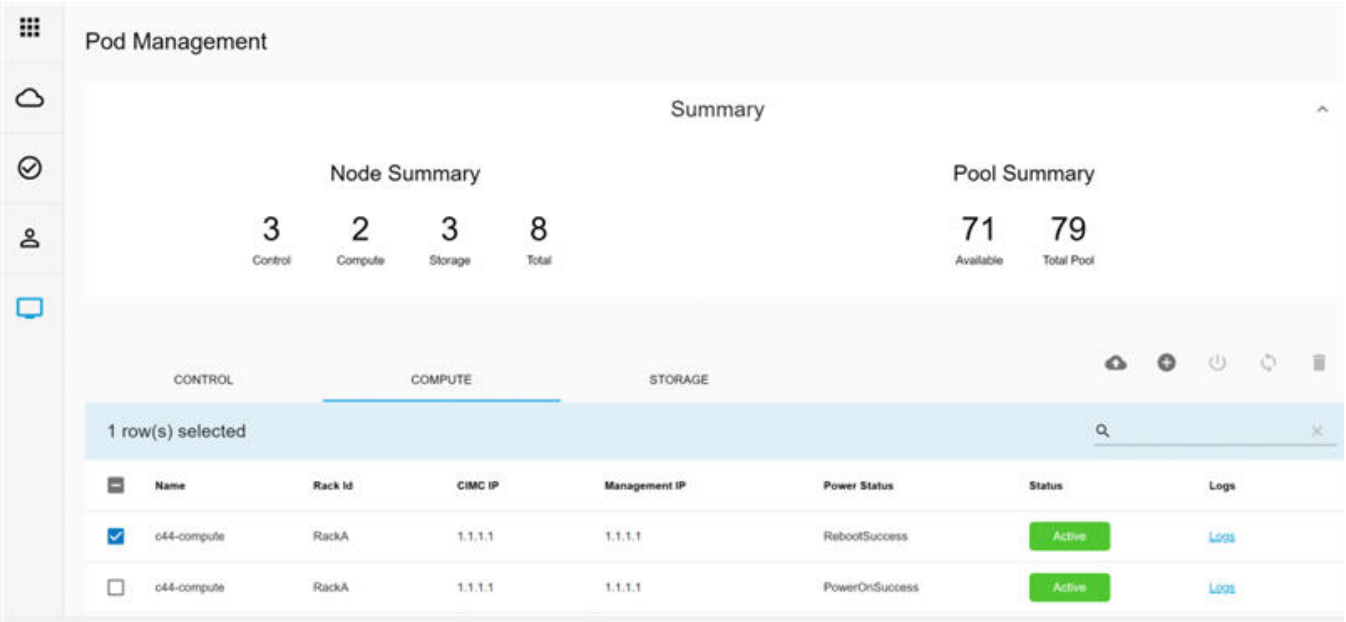
- When you change a control, storage, or compute node in a Cisco VIM pod using Unified Management, it automatically updates the server and role in the active blueprint. Thus, the OpenStack deployment gets changed.
- When a node is removed from Cisco VIM, sensitive data may remain on the drives of the server.
- Administrator recommends you to use Linux tools to wipe the storage server, before using the same server for another purpose. The drives that are used by other application server must be wiped out before adding to Cisco VIM.

Searching Compute and Storage Nodes

On the **Pod Management** page, the search functionality allows you to search the compute and storage nodes based on server names. This **Pod Management** page displays the search results or shows an empty grid if there are no results.

Pod Management Operations

Cisco VIM allows the admin to perform pod life-cycle management from a hardware and software perspective. Cisco VIM provides the ability to power on /off compute node, and add/remove/replace nodes based on the respective roles, when the nodes of a given pod corrupt at times.



The **Pod Management** page has two sections:

1. **Node Summary:** This section shows the number of available nodes and the detailed count of **Control, Compute,** and **Storage** nodes.

2. **IP Pool Summary:** This section shows the total and available IP addresses on the management network.

You can perform the following operations on the running pod:

1. Replace control nodes: Double-fault scenario is not supported. You can replace only one controller at a time.

   ⚠ If the ToR type is Cisco NCS 5500, an additional popup is displayed to enable you to update splitter configuration before replacing the control node

2. Add computes/storage nodes: You can replace N-computes nodes simultaneously. However at any given point, at least one compute node must be active.

   ⚠ If the ToR type is Cisco NCS 5500, an option is available to update the splitter cable configuration.

3. Power On/Off compute nodes: You can power ON or power OFF compute node. At least one compute node must be powered on.
4. Remove compute/storage nodes: You can add one node at a time, when Ceph is run as a distributed storage.

   ⚠ If ToR type is Cisco NCS 5500, an additional popup is displayed to enable the user to update the splitter cable configuration, before the removal of compute or storage node.

5. Add pool: You can increase the pool size at any time.

Managing Storage Nodes

Before you add or remove a storage node, review the following guidelines for managing storage nodes.

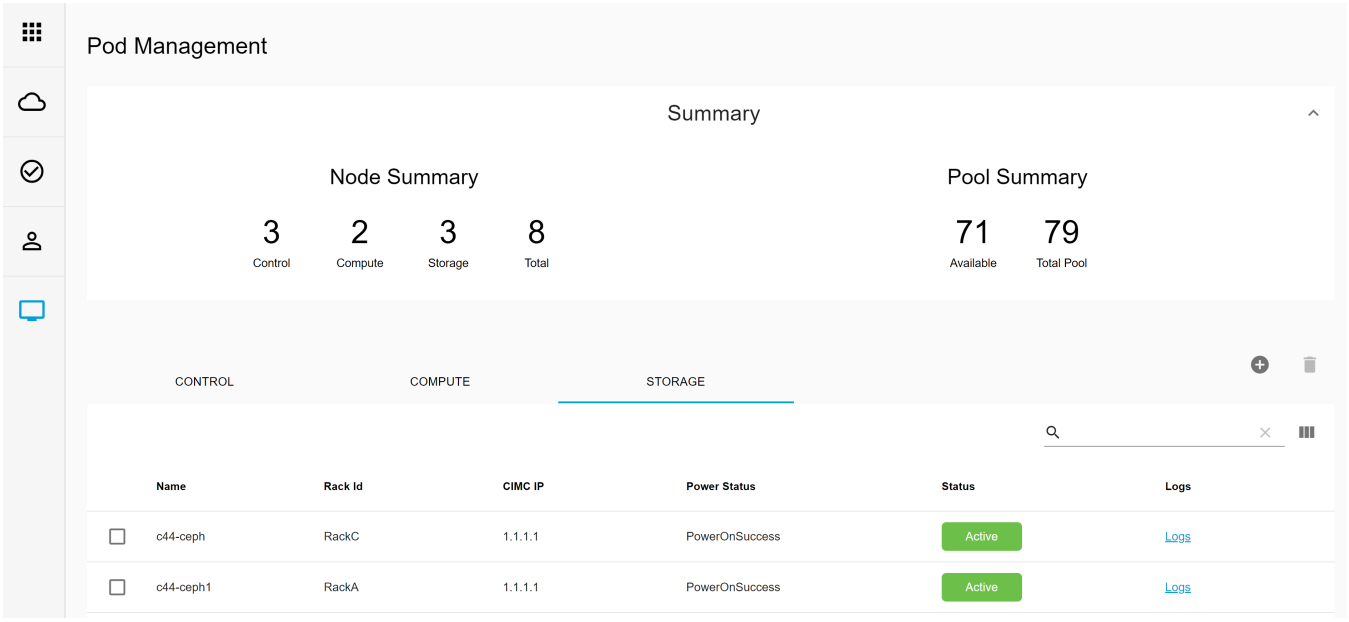
- Required number of storage nodes: A Cisco VIM pod must have a minimum of three and a maximum of 20 storage nodes. If your pod has only two storage nodes, you cannot delete a storage node until you add another storage node. If you have less than three storage nodes, you can add one node at a time, until you get to 20 storage nodes.
- Validation of nodes: When you add a storage node to a pod, Cisco VIM Unified Management validates that all the nodes in the pod meet the minimum requirements and are in active state. If you have a control or compute node in a faulty state, you must either correct, delete, or replace that node before you add a storage node.
- Update blueprint: When you add or delete a storage node, Unified Management updates the blueprint for the Cisco VIM pod.
- Storage node logs: You can access the logs for each storage node from the link in the **Log** column on the storage nodes.

Adding Storage Node

⚠ You cannot add more than one storage node at a time.

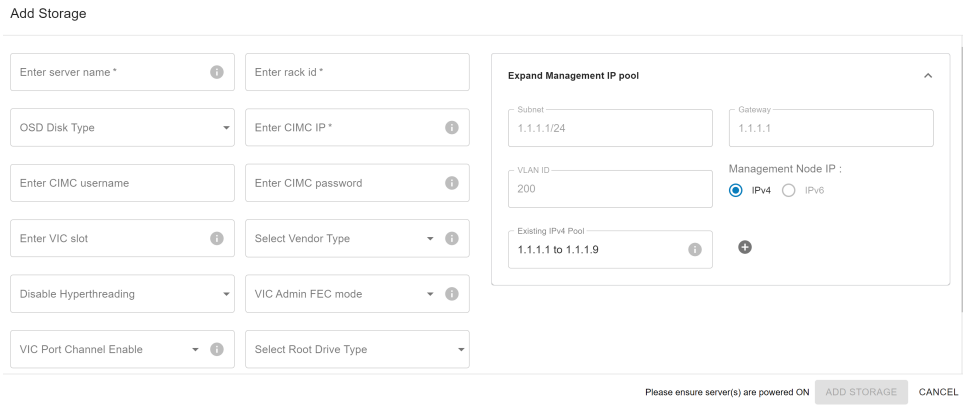
ⓘ **Before you Begin**

- Remove the non-functional storage node from the pod. You can have maximum 20 storage nodes in a Cisco VIM pod.
- Ensure that the server for the new storage node is in powered ON state.



To add a storage node, follow the below steps:

1. Log into UM as pod user and select the pod on the landing page.
2. From the left navigation pane, choose **Operational Tools > Pod Management**.
3. Click the **Storage** tab
4. Click **+** icon. The **Add Storage** window is displayed.



5. For C-series, add the following details:

   - **Server Name**: Enter the name of the storage server to be added.
   - **Rack ID**: Enter the Rack ID in string format.
   - **CIMC IP**: Enter the CIMC IP.
   - **CIMC User Name**: Enter the username for CIMC.
   - **CIMC Password**: Enter the password for CIMC.
   - **VIC Slot**: (Optional). Enter the VIC slot.
   - **ToR switch info**: Mandatory if ToR is configured as True.
   - **Management IPv6**: Enter the IPv6 address of the management network.
   - **Add new TOR**: If ToR is configured as True and all switch ports are exhausted, you can add new pair of ToRs.
   - **Expand Management IP Pool**: If all management IP's from the pool are exhausted, you can expand the management IP pool.

   > ⚠ If all mandatory fields are entered correctly, the **ADD STORAGE** button is enabled.

6. Click **ADD STORAGE** button. An Add node initialization message is displayed and the status of the pod is set to *Active*.
7. To view logs, click **Logs**.

   Two kinds of failure may occur:

   a. **Add Node Pre-Failed**: When addition of node fails in the *Validation* stage, the active blueprint is reverted to the original state and the Unified Management clears the storage node information from the **Storage** tab.

b. **Add Node Post-Failed**: When addition of node fails after the V*alidation* stage, the active blueprint remains modified and the node is visible in the **Storage** tab. At this point, all other links are disabled in the UI and you are required to press **Delete** icon. Unified Management will first delete the node from the blueprint and then initiates the node removal from cloud.

Deleting Storage Node

You cannot remove more than one storage node at a time.

Following are the steps to remove a storage node:

1. Log into UM as pod user and select the pod on the landing page.
2. From the left navigation pane, choose **Operational Tools > POD Management**.
3. Click **Storage** tab.
4. Select the storage node to be deleted and click **Delete** icon. A message **Node Removal Initiated successfully** is displayed. You can remove a storage node with **force** option for hyper-converged pod. The **force** option is useful when VM's are running on the node.

   If the storage node is removed successfully, the storage node is removed from the table.
5. If removal of storage node fails, a pop-up window with **Clear Failed Nodes** button is displayed. Click **Clear Failed Nodes** to remove the node from cloud and blueprint.
6. To view logs, click **View logs**.

Managing Compute Nodes

Before you add or remove a compute node, review the following guidelines:

- Required number of compute nodes: Cisco VIM pod must have a minimum of one compute node and a maximum of 128 nodes. Out of 128 nodes, three nodes are control nodes and the remaining 125 nodes are between compute and Ceph nodes with a maximum of 25 Ceph nodes. If your pod has only one compute node, you cannot delete that node until you add another compute node.

- Update blueprint: When you add or remove a compute node, Unified Management updates the blueprint for the Cisco VIM pod.
- Compute node logs: You can access the logs for each compute node from the link in the respective **Log** column on the compute tab.

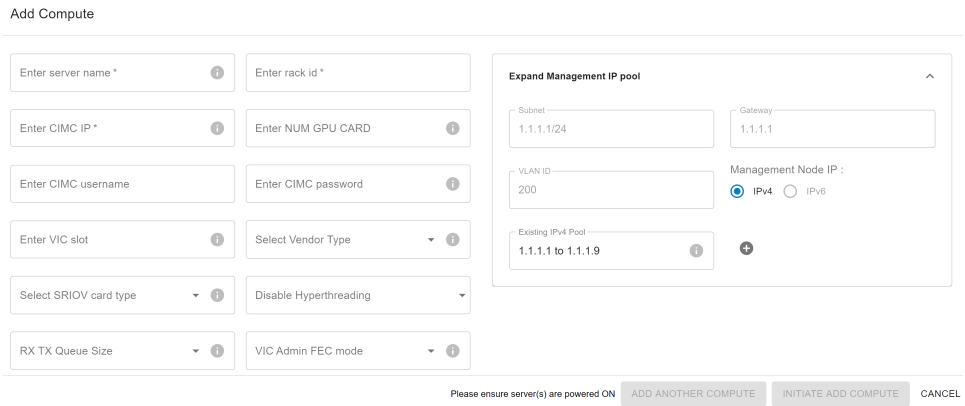
Adding Compute Node

To add a compute node, follow the below steps:

> **Before you begin**
>
> Ensure that the server for the new compute node is in powered state in OpenStack. You can add more than one compute node at a time.

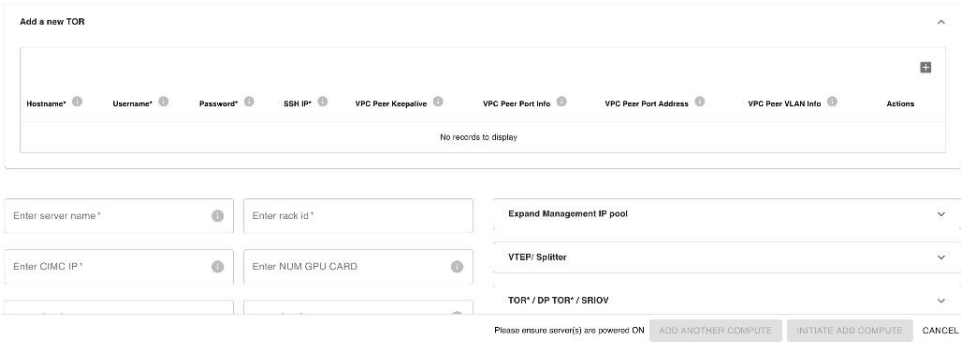
1. Log into UM as pod user and select the pod from the landing page.
2. From the left navigation pane, choose **Operational Tools > Pod Management**.
3. Click **Compute** tab.
4. If you want to create multiple compute nodes at a time, skip to Step 6, otherwise proceed with Step 5.
5. To add compute nodes successively:

   a. Click **+** icon. The **Add Compute** window is displayed.



> If auto-ToR is enabled on Cisco VIM pod, provide ToR port information for each server, so that Cisco VIM automation can provision the switch ports with appropriate configuration. When adding new computes to the cloud, it is possible that all ports in existing switch are exhausted.

   b. To add new pair of ToR switches, click **Add a New ToR** and add new ToR switches information.



c. Enter the required information.

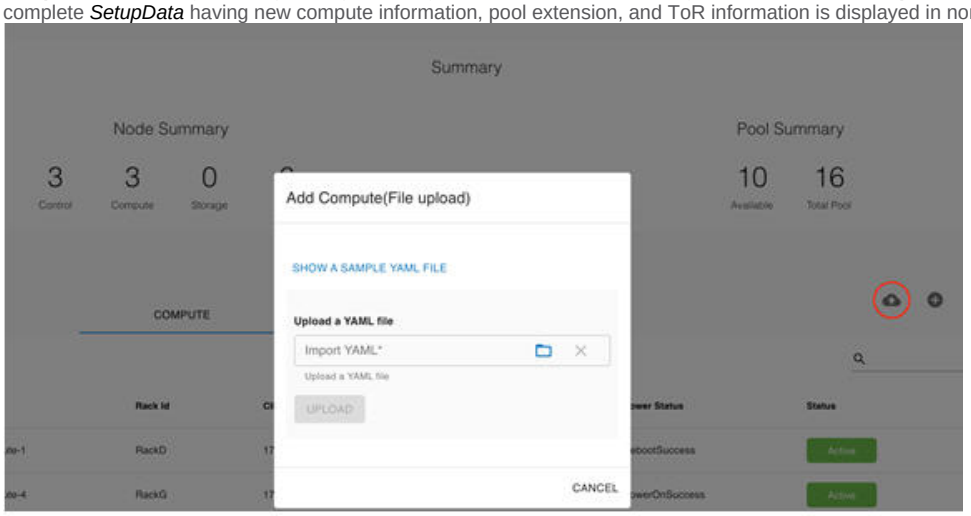
   For C-series, add the following details:

   - **Server Name**: Name of the compute server.
   - **Rack ID**: Enter the Rack ID in string format.
   - **CIMC IP**: Enter the CIMC IP.
   - **CIMC User Name**: Specify the username for CIMC.
   - **CIMC Password**: Enter the password for CIMC.
   - **VIC Slot**: Optional. Enter the VIC slot.
   - **ToR switch info**: Mandatory if configured ToR is set to True.
   - **DP ToR switch info**: Enter DP ToR switch information in string format.
   - **SRIOV ToR info** : Enter the SRIOV ToR information in string format.
   - **Management IPv6** : Enter the IPv6 address of the management network.
   - **Trusted_vf**: Optional and not reconfigurable. Applicable only for SRIOV node with compute role for C-series pod.
   - **Vtep IPs**: IP address from vxlan-tenant and vxlan-tenant.
   - **INTEL_SRIOV_VFS** :Value ranges from 1 to 32.
   - **INTEL_FPGA_VFS**: Value ranges from 1 to 8.
   - **INTEL_VC_SRIOV_VFS**: Value ranges from 1 to 32.
   - **Vendor**: Optional. It can be CISCO - Cisco Systems Inc, QCT - Quanta Cloud Technology Inc, or HPE - Hewlett Packard Enterprise or DELL.
   - **VM Hugepage Size**: Optional. It can be 2M or 1G. Only applicable with NFV HOSTS.
   - **RX TX Queue Size**: Optional. It can be 256, 512, or 1024.
   - **SECCOMP_SANDBOX** : Optional. If not defined, set to 1.
   - **NOVA_CPU_ALLOCATION_RATIO**: Optional, overrides the NOVA_CPU_ALLOCATION_RATIO defined in *openstack_config.yaml.* Values lie in the range of 0.958 to 16.0
   - **NOVA_RAM_ALLOCATION_RATIO:** Optional, overrides the NOVA_RAM_ALLOCATION_RATIO defined in *openstack_config.yaml.* Values lie in the range of 1.0 to 4.0
   - **NUM GPU CARDS**: Optional, for server with GPU. Value lies in the range from 0 to 6.
   - **root_drive_type: <HDD or SSD or M.2_SATA, NUM_GPU_CARDS**: 0 to 6.
   - **VIC Port Channel Enable**: Optional. It can be True or False. By default, it is set to True.
   - **VIC Admin FEC mode**: Optional. It can be auto, off, cl74, or cl91.
   - **COMBINE CPDP:** Optional. It can be True or False. This field is enabled on **Add Compute** window only if INTEL_NIC_SUPPORT is set to True. If COMBINE CPDP is set to True, the following property is added in the *setup_data.yaml* file.

     ```
     Hardware_info: { INTEL_SRIOV_PHYS_PORTS: 2 }
     ```

d. If the existing management network pool is exhausted, click **Expand Management IP Pool** to add a new pool.
e. Click **Initiate Add Compute**. A message **Add compute initiated successfully** is displayed.
f. If you wish to add another compute node, click **Add Another Compute** button. The data in the existing form is saved. You need to fill new compute node information and then click **Initiate Add Compute**. You may use **Previous** and **Next** buttons to navigate among the added node information. Then, proceed with Step 8.

6. Click **Upload** icon above the search functionality. An **Add Compute(File upload)** popup appears. To view a sample *yaml* file, click **Show a sample YAML file**. By referring to the *yaml* file, you can create a customized *yaml* file and click the **Upload** button to upload the *yaml* file. The complete *SetupData* having new compute information, pool extension, and ToR information is displayed in non-editable format.



7. Verify the changes in *SetupData* and click **Initiate** to start the add compute operation.
8. To view logs, click **View logs.**

   Two kinds of failure may occur:

   - **Add Node Pre-Failed**: When addition of node fails in the *Validation* stage, the active blueprint is reverted to the original state and the Unified Management clears the compute node information from the **Compute** tab.
   - **Add Node Post-Failed**: When addition of node fails after the *Validation* stage, the active blueprint remains modified and the node is visible in the **Compute** tab. At this point, all other links are disabled in the UI and you are required to press **Delete** icon. Unified Management first deletes the node from the blueprint and then initiates the node removal from cloud.

Deleting Compute Node

You can remove single or multiple compute nodes simultaneously in case of a hardware failure.

> ⚠ If your pod has only one compute node, you cannot remove that node until you add another compute node.

1. Log into UM as pod user and select the pod from the landing page.
2. From the left navigation pane, choose **Operational Tools** > **Pod Management.**
3. Click **Compute** tab.
4. Select the compute node(s) to be deleted and click **Delete** icon. You can remove a compute node with **Force** option, when VM's are running on the node. A message **Node removal initiated successfully** message is displayed.

   If compute nodes are removed successfully, you cannot view the compute node in the table.
5. If removal of compute node fails, the **Clear Failed Nodes** button is displayed. Click **Clear Failed Nodes** to remove the node from cloud and blueprint.
6. To view the Logs, click **View logs**

Managing Control Nodes

Before you replace a control node, review the following guidelines:

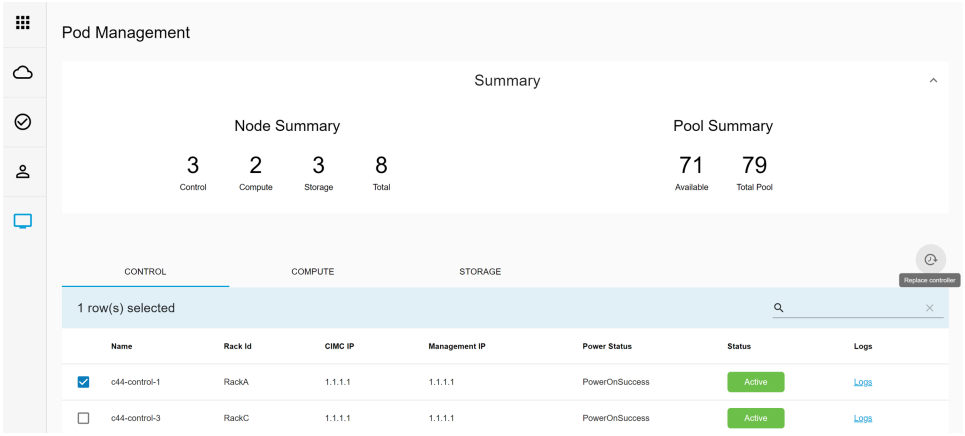
- Required number of control nodes: A Cisco VIM pod must have three control nodes and you can only replace one node at a time.
- Validation of nodes: When you replace a control node, Cisco VIM Unified Management validates if all the other nodes in the pod meet the minimum requirements and are in active state. If you have a storage or a compute node in a faulty state, you must correct the faulty state or delete or replace that node before you can replace the control node.
- Update blueprint: When you replace a control node, Unified Management updates the active blueprint for the Cisco VIM pod.
- Control node logs: You can access the logs for each control node from the link in the respective **Logs** column.

Replacing Control Node

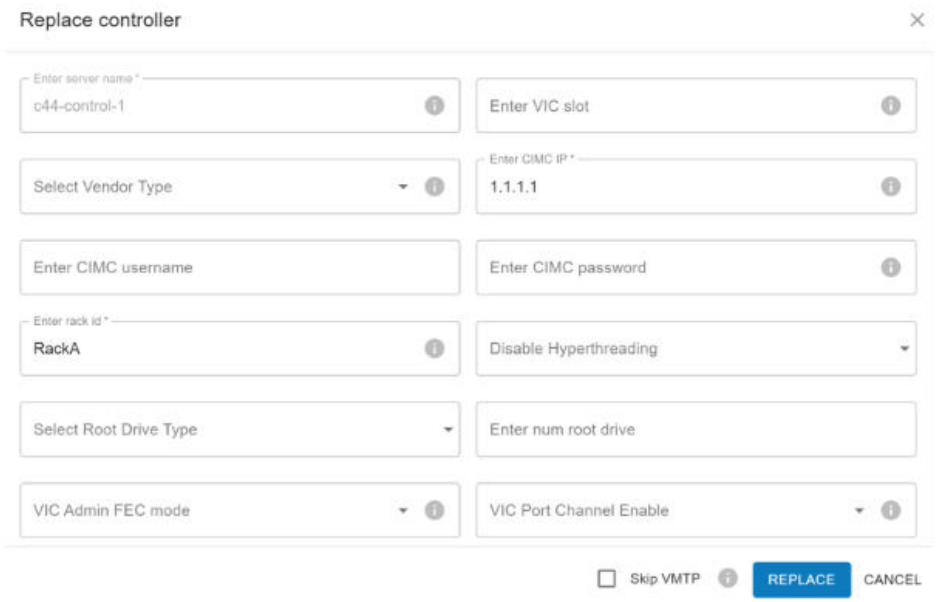
You can replace only one control node at a time.

1. From the navigation pane, choose **Operational Tools > Pod Management**.

2. Click **Control** tab.



3. Select the control node to be replaced and click **Spin** icon. The **Replace controller** window is displayed. When VMs are running on the node, a confirmation pop-up is shown with **Force** option and **Proceed** button. Click **Proceed** to continue. You can replace a control node with **Force** option for Micropod.



4. If you want to edit a specific control node before replacement, make the updates and then click **Replace**. On success, a message **Replace Node Initiated successfully** is displayed.
5. If the replacement of the control node fails, do the following:

    a. Click the link in the **Logs** column to view the logs.
    b. Check the logs to determine the cause of the failure.
    c. Correct the issue and attempt to replace the control node again.

- For replace controller, you can change only a subset of the server information.
- For C-series, you can change the server information such as CIMC IP, CIMC Username, CIMC password, rack_id, and tor_info.

For remove-compute, remove-storage, and replace-controller operations, Cisco VIM attempts to change the host OS boot mode to single user and power-off those server(s) to avoid the creation of duplicate IP address on the management and storage networks, and to prevent the remove/replace server from connecting back to the deployment.

If the remove/replace server is powered-off before triggering Cisco VIM remove/replace operation, the server must not be brought back on the network as it may create duplicate IP address when connected to deployment with valid credentials.

To power up those removed/replaced server(s):
- isolate the server on the network (shutdown of corresponding ToR switchports or unplug the network cables)
- power up the server
- delete the virtual drive from CIMC
- power-cycle the server
- bring the server on the network (no shutdown of corresponding ToR switchports or plug back the network cables)

NG Managing Power

Managing Power

- Power Management of Compute Nodes
- Powering ON a Compute Node
- Powering OFF a Compute Node
- Reboot Compute Node
- Power Status

Power Management of Compute Nodes

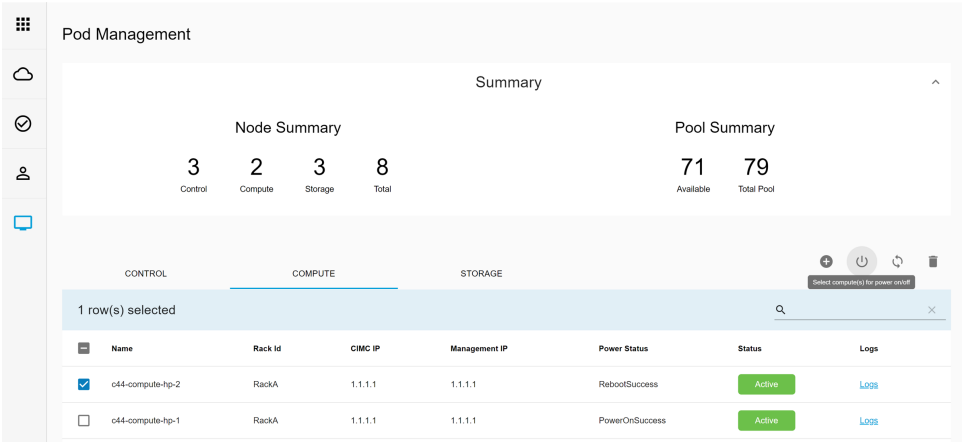
If no active workloads are running on compute nodes but are still part of Cisco VIM cloud, you can simply power OFF them and turn ON them when required. This optimizes the overall power consumption of the data center. After a software update, when Kernel and other system packages are updated, you may need to reboot compute nodes.

Powering ON a Compute Node

Following are the steps to power ON the compute nodes:

1. Log into UM as pod user and select the pod on the landing page.
2. From the left pane, choose **Operational Tools > Pod Management**
3. On the **Pod Management** page, click **Compute** tab.
4. Select the compute nodes that you want to power ON.

> ⚠ The **Power** icon is enabled only after you select the compute nodes.



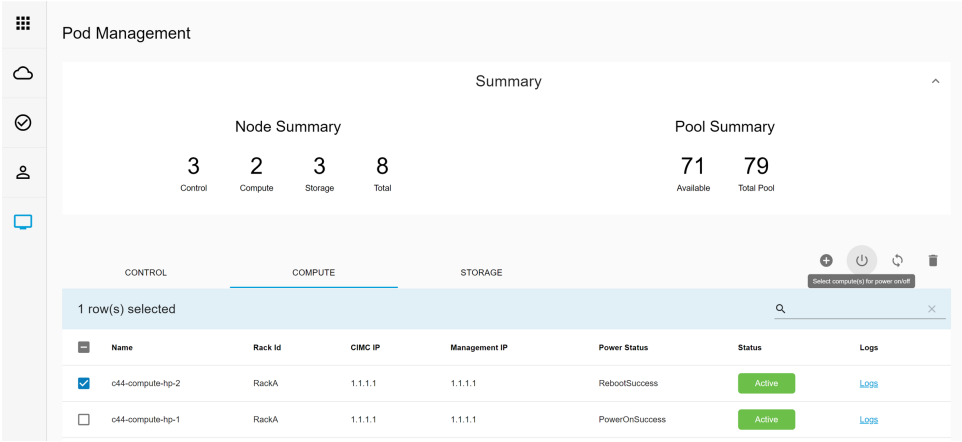
5. Click the **Power** icon available above the search option. A pop-up window with **Proceed** and **Cancel** buttons is displayed.
6. Click **Proceed** to power ON the compute nodes,

Powering OFF a Compute Node

> ⚠ You cannot power OFF all the compute nodes. There must be at least one compute node in ON state.

Following are the steps to power OFF the compute nodes:

1. Log into UM as pod user and select the pod on the landing page.
2. Fromthe left pane, choose **Operational Tools > Pod Management.**
3. On the **Pod Management** page, click **Compute** tab.
4. Select compute nodes that you want to power OFF. A pop-up window with **Proceed** and **Cancel** buttons is displayed.
5. Click **Proceed** to power-off the selected nodes.

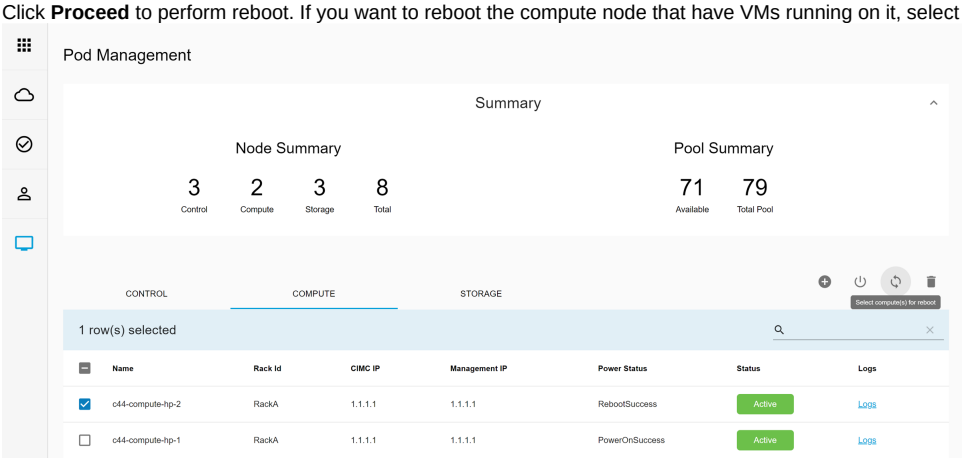


If there is only one compute node and you try to power OFF it, a message **Last compute node can't be powered off** is displayed. Also, when you power OFF the last available compute node in the list of nodes, the message **At least one compute node should be powered on** is displayed.

Reboot Compute Node

To reboot the compute node, follow the below steps:

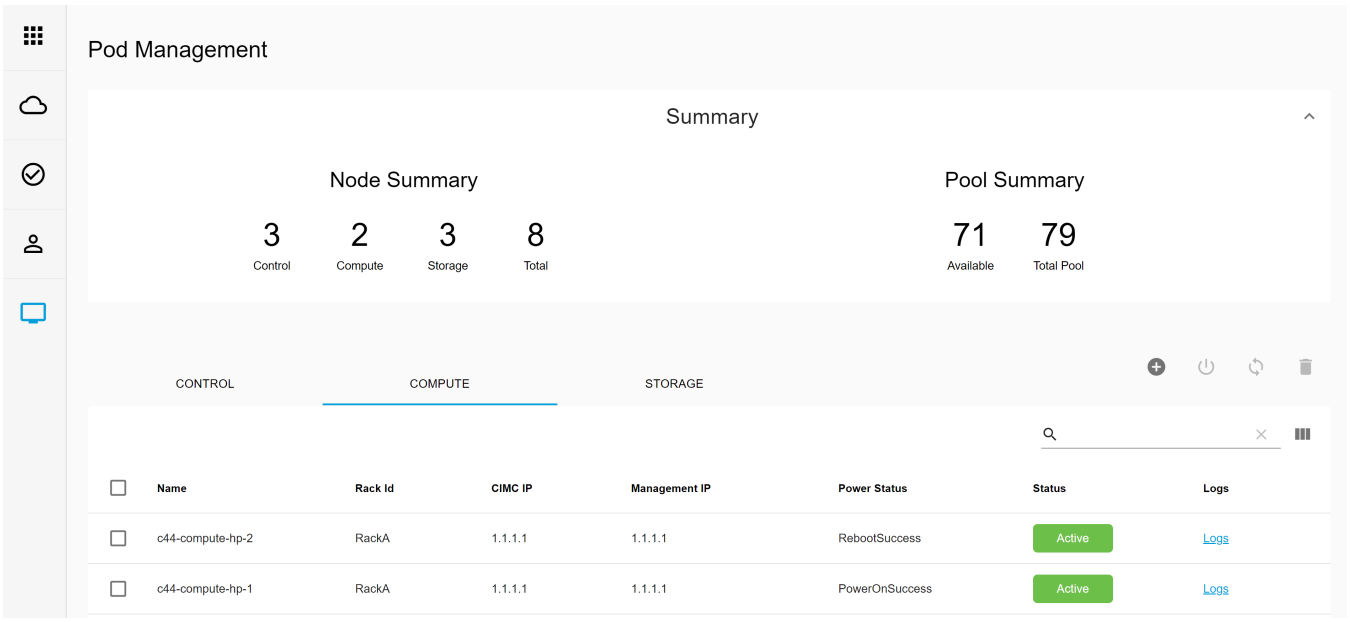
1. Log into UM as pod user and select the pod on landing page.
2. On the left pane, choose **Operational Tools > Pod Management.**
3. On the **Pod Management** page, click **Compute** tab.
4. Select compute nodes that you want to reboot.
5. Click **Reboot** icon available above the search option. A message **Do you really want to reboot the selected compute node** is displayed with a **Force** option and **Proceed** and **Cancel** buttons.
6. Click **Proceed** to perform reboot. If you want to reboot the compute node that have VMs running on it, select **Force** option.



All compute nodes cannot be rebooted simultaneously. There should be at least one compute in power ON state.

Power Status

To view the last power management operation status for each compute node, click **Compute** tab.


Pod Management
Summary
Node Summary
3
Control
2
Compute
3
Storage
8
Total
Pool Summary
71
Available
79
Total Pool
CONTROL
COMPUTE
STORAGE
Name
Rack Id
CIMC IP
Management IP
Power Status
Status
Logs
c44-compute-hp-2
RackA
1.1.1.1
1.1.1.1
RebootSuccess
Active
Logs
c44-compute-hp-1
RackA
1.1.1.1
1.1.1.1
PowerOnSuccess
Active
Logs

NG Uploading Glance Images

Uploading Glance Images

Allows UM admins to upload VM Images to OpenStack pods registered on the UM node. You can upload images to Glance from the UM node without logging into Horizon dashboard or executing OpenStack CLIs.

To upload Glance images, follow the below steps:

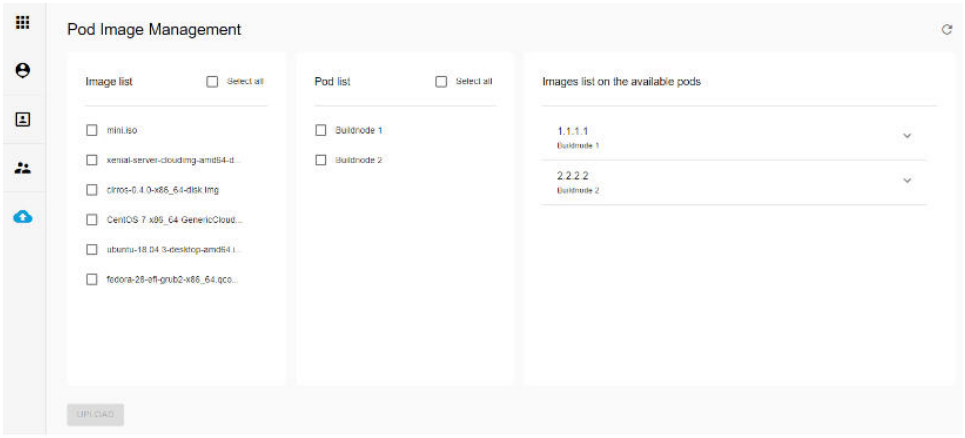
1. On Day 0, you need to upload all the Glance Images in a folder on UM node. To copy those images, use the *glance_image_copy.sh script* available in the insight folder. Once the images are uploaded, UM Admins can view those images on **Pod Image Management** page in UM Profile.

```
# cd /root/Insight-<tag_id>/insight
# ./glance_image_copy.sh -h
glance_image_copy.sh : Cisco Glance Images Validator
-----------------------------------------------------
-d : Input Glance Images Directory.
Allowed Images Formats:
- 'raw', 'qcow2', 'iso', 'ami', 'ari', 'aki', 'vhd', 'vhdx', 'vmdk', 'vdi','ploop'
-h : To display this help
```

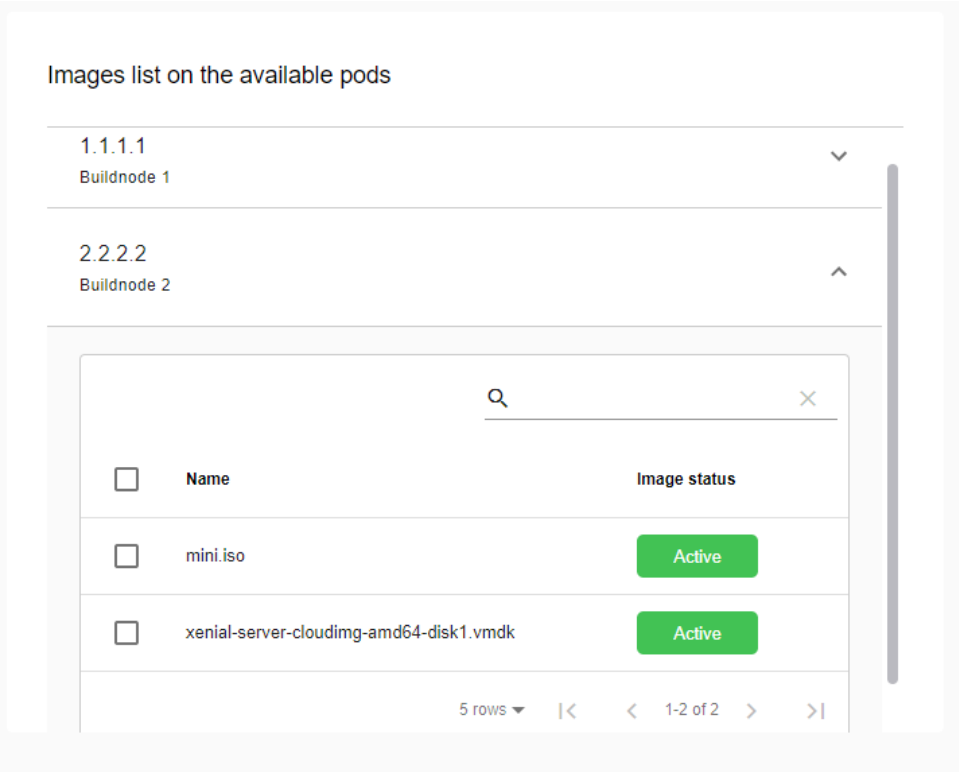
2. Use *-d* option which represents the directory path of Glance Images, and copy the Glance Images by running the following command:

```
# ./glance_image_copy.sh -d /root/ImagesDir/
Copying /root/ImagesDir/mini.iso ...
Copying /root/ImagesDir/xenial-server-cloudimg-amd64-disk1.vmdk ...
Copying /root/ImagesDir/cirros-0.4.0-x86_64-disk.img ...
Copying /root/ImagesDir/CentOS-7-x86_64-GenericCloud-1905.raw ...
```

3. After copying images using the *glance_image_copy.sh* script, refresh the Cisco VIM Unified Management appliance to view the Glance Images under **Pod Image Management**.



4. To upload single/multiple images, select the images to be uploaded and single/multiple pods, and click **Upload**. The status of uploaded/available image(s) are shown under the expansion panel for each pod.



5. To delete a single/multiple image(s), select the image(s) to be deleted from the particular pod.

> ⚠ You can perform image upload or delete operations only on the pods that are in the *Active* state.

6. To view the image status for a particular pod after an upload or delete operation, click the **Refresh** icon.

Day 2 Operations of UM

Day 2 Operations of Cisco VIM Unified Management

- Restarting UM
- Reconfiguring UM
- Update/Upgrade UM
- Rollback UM
- Commit UM
- Migrating of UM to NG Instance
- Shutting Down UM
- UM Tech Support

Restarting UM

Restarting Cisco VIM Unified Management

1. Log into the server in which the Unified Management container is stopped.
2. Restart the Unified Management service by running the following command from the shell window:

```
systemctl restart docker-insight
```

   a. Check the status of Unified Management container by running the following command: *docker ps -a* | *grep* insight.

```
STATUS
 Up 6 seconds
```

   b. Check the status of the service by running the following command:

```
systemctl status docker-insight
```

   The following output is displayed:

```
Docker-insight.service - Insight Docker Service
Loaded: loaded (/usr/lib/systemd/system/docker-insight.service; enabled; vendor preset: disabled)
Active: active (running) since <Date and Time when it got active.
```

Reconfiguring UM

Reconfiguring Cisco VIM Unified Management

- Reconfigurable Keys
- Reconfiguring Unified Management TLS Certificate
    - Reconfiguring Third-party TLS Certificate
    - Reconfiguring Self-signed TLS Certificate
    - Switching from Self-signed TLS Certificate to Third-party TLS Certificate
- Reconfiguring Unified Management MySQL Database Password
    - System-generated Unified Management DB Password
    - User-supplied Unified Management DB Password
- Reconfiguring Unified Management SMTP Server
- Reconfiguring Unified Management LDAP Server
- Reconfiguring Unified Management Optional Features

UM reconfigure action provides you with three major functionalities:

1. Reconfigure Unified Management TLS certificate.
2. Switch from Self-signed TLS certificate to third-party TLS certificate.
3. Reconfigure Unified Management MySQL database password.

An autobackup is triggered after each successful reconfiguration.

> ⚠ Unified Management reconfiguration is not allowed after an update, as the update is an intermediate stage between rollback and commit

Reconfigurable Keys

List down all the reconfigurable keys from Insight setup data.

```
# cd Insight-<tag_id>
# ./insight/insight_runner.py --list-reconfigure-keys

Insight setup data reconfigurable keys:
+--------+-------------------------------+-----------------------------------------------------------------+
| SL.NO. | Name                          | Description                                                     |
+--------+-------------------------------+-----------------------------------------------------------------+
|   1    | DISPLAY_ALL_POD_USERS         | Display/Hide all pod users at its registration                  |
|   2    | INSIGHT_EMAIL_ALIAS           | Email Alias to send notifications from Insight                  |
|   3    | INSIGHT_EMAIL_ALIAS_PASSWORD  | Password to Insight email alias                                 |
|   4    | INSIGHT_SMTP_PORT             | SMTP Port                                                       |
|   5    | INSIGHT_SMTP_SERVER           | SMTP Server for mailing service                                 |
|   6    | LDAP_ADMIN                    | Full DN of LDAP admin user                                      |
|   7    | LDAP_ADMIN_PASSWORD           | Password of LDAP admin user                                     |
|   8    | LDAP_BASE_DN                  | DN tree for LDAP groups                                         |
|   9    | LDAP_CERT_PATH                | Path to self-signed certificate for ldaps                       |
|   10   | LDAP_GROUP_MEMBER             | Attribute on LDAP server used to define group members           |
|   11   | LDAP_GROUP_SEARCH_FILTER      | Search filter to search groups on LDAP                          |
|   12   | LDAP_GROUP_USER_SEARCH_FILTER | Search filter to search group members on LDAP                   |
|   13   | LDAP_MODE                     | Switch to enable LDAP Mode                                      |
|   14   | LDAP_PORT                     | LDAP Server Port                                                |
|   15   | LDAP_ROLE_MEMBER              | Attribute on LDAP server used to define role associated members |
|   16   | LDAP_SECURE                   | Switch to control LDAP protocol ldap/ldaps                      |
|   17   | LDAP_SERVER                   | IP Address or hostname of LDAP server                           |
|   18   | LDAP_USER_ID_ATTRIBUTE        | Attribute to be used as LDAP user ID                            |
|   19   | POD_ADMIN_GROUP               | LDAP group to be mapped as Insight pod admins                   |
|   20   | POD_USER_GROUP                | LDAP group to be mapped as read-write pod users                 |
|   21   | READ_ONLY_POD_USER_GROUP      | LDAP group to be mapped as read-only pod users                  |
|   22   | SSL_CERT_CHAIN_FILE           | SSL chain file of x509 certificate                              |
|   23   | SYSLOG_EXPORT                 | Support for remote syslog export                                |
|   24   | UM_ADMIN_GROUP                | LDAP group to be mapped as Insight UM admins                    |
|   25   | UM_ADMIN_WITH_FULL_POD_ACCESS | Allow/Deny pod user permissions to UM Admin                     |
|   26   | UM_LDAP_ADMINS                | Support LDAP for admin access to UM management node             |
|        |                               | All keys other than domain_name are reconfigurable.           |
|   27   | ntp_servers                   | Clock synchronization for the UM management node                |
|   28   | permit_root_login             | Option to enable/disable root SSH access                        |
|   29   | ssh_banner                    | Display banner during SSH login                                 |
|   30   | vim_admins                    | Support the management of VIM administrators                    |
+--------+-------------------------------+-----------------------------------------------------------------+
```

Reconfiguring Unified Management TLS Certificate

As the UM web-service is protected by TLS, reconfigure action provides flexibility to change the existing TLS certificate. As there were two approaches to configure it, there are also two approaches to change it.

Reconfiguring Third-party TLS Certificate

If you had provided your own TLS Certificate before Insight Installation through PEM_PATH key in insight_setup_data.yaml, then perform the following steps to reconfigure it.

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. Edit the *insight_setup_data.yaml* to change the value of PEM_PATH and/or SSL_CERT_CHAIN_FILE key to point to the path of your new valid TLS/Cert chain file certificate. Then, save the file.
   For example:

   ```
   PEM_PATH: "/root/new_tls.pem"
   SSL_CERT_CHAIN_FILE: "/root/new_ssl.crt"
   ```

3. Start the reconfiguration using the following commands:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure -f </root/insight_setup_data.yaml>

Perform the action. Continue (Y/N)y
The logs for this run are available at /var/log/insight/bootstrap/<date>_<time>

                ################################
                 CISCO 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  |
| CVIM/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 sss daemon]             [ DONE! ]    1min 23secs

.
.

Cisco VIM Insight already Installed!
+------------------+--------+-------------------------------------------------------+
| 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 Cisco 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>

As the summary table describes Insight gets autobacked up after reconfigure at /var/cisco/insight_backup
to preserve the latest state of Insight.
```

Reconfiguring Self-signed TLS Certificate

If you have created a new TLS certificate through *tls_insight_cert_gen.py* before UM installation, follow the below steps to reconfigure it.

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. Run the following commands to reconfigure the self-signed TLS certificate:

```
# cd Insight-<tag_id>
# ./tls_insight_cert_gen.py -h
usage: tls_insight_cert_gen.py [-h] [--overwrite] --file INSIGHTSETUPDATA

TLS cert generator Insight

optional arguments:
  -h, --help            show this help message and exit
  --overwrite, -o       Overwrite Insight certificates if already present in openstack config directory
  --file INSIGHTSETUPDATA, -f INSIGHTSETUPDATA
                        Location of insight_setup_data.yaml

# ./tls_insight_cert_gen.py -f <path insight_setup_data.yaml> --overwrite
This will overwrite the existing TLS certificate.
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  |
| CVIM/Insight workspace conflict check     | PASS   | None  |
| Check registry connectivity               | PASS   | None  |
| Check Email server for Insight            | PASS   | None  |
+-----------------------------------------+--------+-------+
Generating a 4096 bit RSA private key
..............................................................................++
.......++
writing new private key to '../openstack-configs/insight.key'
```

3. Start reconfiguration:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure -f </root/insight_setup_data.yaml>
```

Switching from Self-signed TLS Certificate to Third-party TLS Certificate

If you have created a new TLS certificate through *tls_insight_cert_gen.py* before Insight Installation and want to switch to your own TLS certificate, perform the following steps.

⚠ You cannot switch from third-party TLS certificate to self-signed TLS certificate.

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. Edit the *insight_setup_data.yaml* to add PEM_PATH and SSL_CERT_CHAIN_FILE key to point to path of your new valid TLS and SSL_CERT_CHAIN certificate. Save the file after editing.
   For example:

```
PEM_PATH: "/root/new_tls.pem"
SSL_CERT_CHAIN_FILE: "/root/new_ssl.crt"
```

3. Run the bootstrap:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure -f </root/insight_setup_data.yaml>
```

Reconfiguring Unified Management MySQL Database Password

You can use one of the below methods to reconfigure the MySQL DB password:

1. System generated Unified Management DB password.
2. User-supplied Unified Management DB password.

System-generated Unified Management DB Password

Following are the steps to generate MySQL Unified Management DB password:

1. To generate the Unified Management DB Password run the following command:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure --regenerate_secrets
```

2. Verify the password change by running the following command:

```
# cat /opt/cisco/insight/secrets.yaml
DB_ROOT_PASSWORD: <new_db_password>
```

User-supplied Unified Management DB Password

1. To provide your own MYSQL DB Password, follow the below steps:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure --setpassword
```

   The password must satisfy at least three of the following. Ensure that there is no space and the password length is less than or equal to 16.

   - at least a letter between a to z
   - at least a letter between A to Z
   - at least one number between 0 to 9
   - at least a character from !@-_+=

2. Verify the password change by running the following command:

```
#cat /opt/cisco/insight/secrets.yaml
DB_ROOT_PASSWORD: <new_db_password>
```

Reconfiguring Unified Management SMTP Server

UM requires a valid SMTP server to send emails to Pod-Admin, UI-Admin, and regular users. If the SMTP server is down, you can reconfigure it. Following values can be reconfigured:

- INSIGHT_SMTP_SERVER
- INSIGHT_EMAIL_ALIAS_PASSWORD (only needed for Authenticated SMTP server)
- INSIGHT_EMAIL_ALIAS
- INSIGHT_SMTP_PORT (optional, defaults to 25)

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. Edit the *insight_setup_data.yaml* to add the desired values.
3. Save the file after editing.
4. Run the bootstrap command as follows:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure -f </root/insight_setup_data.yaml>
```

Reconfiguring Unified Management LDAP Server

UM supports both LDAP and LDAP over SSL (LDAPS) for an Active Directory (AD) environment. If the LDAP server is down or if you need to change any of its configuration, execute Unified Management reconfigure acion.

1. Reconfigure the LDAP(s) server:
    a. Fetch the latest *insight_setup_data.yaml* file

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

    b. Edit the *insight_setup_data.yaml* to change the value of LDAP keys.

    Following is the list of LDAP keys with their reconfiguration rules:

LDAP Key	Reconfigure Rules
LDAP_MODE	Can be reconfigured only to True, to switch from No-LDAP to LDAP only, but not vice-versa.
LDAP_SERVER	Reconfigurable to switch to a new LDAP server.
LDAP_PORT	Reconfiguration of this key is allowed.
LDAP_ADMIN	Reconfiguration of this key is allowed.
LDAP_ADMIN_PASSWORD	Reconfiguration of this key is allowed.
LDAP_SECURE	Can be reconfigured only to True, to switch from non-secure LDAP to secure LDAP connection, but not vice-versa.
LDAP_CERT_PATH	Can be reconfigured to switch from self-signed certificate to CA-signed certificate, but not vice-versa.
LDAP_USER_ID_ATTRIBUTE	Can be reconfigured to point to the new LDAP user id attribute.
LDAP_GROUP_SEARCH_FILTER	Can be reconfigured to set a new group search filter.
LDAP_GROUP_MEMBER	Can be reconfigured to set a new attribute that i s used on LDAP server for defining user-group association.
LDAP_GROUP_USER_SEARCH_FILTER	Can be reconfigured to set a new group-user search filter.
UM_ADMIN_GROUP	Can be reconfigured to map LDAP role/group to Insight UM-Admin(s). This key is used, when Insight authorization is via LDAP server.
POD_ADMIN_GROUP	Can be reconfigured to map LDAP role/group to Insight Pod-Admin(s). This key is used, when Insight authorization is via LDAP server.

POD_USER_GROUP	Can be reconfigured to map LDAP role/group to Insight pod users with *Full-pod-access*. This key is used, when Insight authorization is via LDAP server.
READ_ONLY_POD_USER_GROUP	Can be reconfigured to map LDAP role/group to Insight pod users with *Read-only-access.* This key is used, when Insight authorization is via LDAP server.
LDAP_ROLE_MEMBER	Can be reconfigured to set a new attribute that is used on LDAP server for defining user-role/group-role association.

c. Save the edited file.

2. Run the bootstrap command:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure -f </root/insight_setup_data.yaml>
```

Reconfiguring Unified Management Optional Features

Unified Management supports the reconfiguration of optional features.

1. Reconfigure the UM optional feature:

   a. Fetch the latest *insight_setup_data.yaml* file

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

   b. Edit the *insight_setup_data.yaml* to change the value of optional feature keys below:

   - UM_ADMIN_AS_POD_ADMIN: If set to True, all UM-Admins are added as pod-users with *Full-Pod-Access* during pod registration.
   - DISPLAY_ALL_POD_USERS: If set to True, a drop-down appears in the user-registration form with a list of all the users with pod-user permissions while adding a new pod-user.

2. Run the bootstrap command:

```
# cd Insight-<tag_id>
$ ./insight/insight_runner.py --reconfigure -f </root/insight_setup_data.yaml>
```

Update/Upgrade UM

Update/Upgrade Unified Management

- Update Scenarios
- Update VIM UM with Internet Access from 3.2.x (x<=4) or 3.4.y
- Update VIM UM without Internet Access from 3.2.x (x<=4) or 3.4.y
- Upgrade Scenario
- Upgrade Cisco VIM UM from 3.4.6 or later (in 3.4.x lineup)

Cisco VIM Unified Management update allows you to switch to a new UM release. To move the UM node from Cisco VIM 3.2.x or Cisco VIM 3.4.y where x<=4 and y is less than the maintenance release version of the current release, to the current Cisco VIM release, execute the *update* action.

The ***update*** action makes the old docker containers of UM and MariaDB in exit state and brings up new ones with the new tag. The old containers and images are restored until you perform the ***Commit*** action.
The update is an intermediate action and allows you to do either a ***Commit*** action to settle for the current version or do a ***Rollback*** to revert back to the old version.
The old workspace is preserved if you want to do a rollback to the previous version.

Old and new workspaces are preserved after commit and rollback respectively in */opt/cisco/insight/insight_archive*

After an update:

- Your UM workspace is set as the new workspace that you just extracted out of the tarball.
- Backup and reconfigure action are not allowed either from old or new UM workspace.

Update Scenarios

Following are the update scenarios:

- Insight and MariaDB containers get updated to the new tag.
- Either Insight or MariaDB container gets updated to the new tag.
- Data containers get updated to the new tag and not reverted to old tag after rollback. This is done to preserve potential security and kernel fixes.

Update VIM UM with Internet Access from 3.2.x (x<=4) or 3.4.y

The value of *y* is less than the maintenance release version of the current Cisco VIM release.

Following are the steps to update VIM UM:

1. Get the new installer tarball, which is available after each release. Extract the tarball to get the new UM workspace by running the following command:

```
tar --no-same-owner -xvzf mercury-installer.tar.gz
```

2. Create an empty directory for Insight workspace: *Insight-<tag_id>*
3. Copy the contents of the extracted directory *installer-<tag_ig>* within *Insight-<tag_id>*

```
# cp -r installer-<tag_ig>/* Insight-<tag_id>
```

4. Update the Cisco VIM UM by running the following commands:

```
# cd /root/Insight-<tag_id>

Check the containers lined up for update
# ./insight/insight_runner.py --update --dry-run

Start the update
# ./insight/insight_runner.py --update
```

5. Verify the UM update:

```
# ./insight/insight_runner.py --update-status

Cisco VIM Insight Update Status!
+-----------------------------------------+--------+-------------------------------+
| Description                             | Status | Details                       |
+-----------------------------------------+--------+-------------------------------+
| VIM Insight Container: insight_<new_tag> | PASS   | Updated from insight_<old_tag> |
| VIM Mariadb Container: insight_<new_tag> | PASS   | Updated from mariadb_<old_tag> |
+-----------------------------------------+--------+-------------------------------+
```

Update VIM UM without Internet Access from 3.2.x (x<=4) or 3.4.y

> ⚠ The value of *y* is less than the maintenance release version of the current Cisco VIM release.

1. Copy the new installer tarball to the UM node. Extract the tarball to get the new UM workspace by running the following command:

```
tar --no-same-owner -xvzf mercury-installer.tar.gz
```

2. Create an empty directory for Insight workspace: *Insight-<tag_id>*
3. Copy the contents of the extracted directory *installer-<tag_ig>* within *Insight-<tag_id>*

```
# cp -r installer-<tag_ig>/* Insight-<tag_id>
```

4. To download, test and import the new UM artifacts, follow the steps given in UM Without Internet Access only till Step 11.
5. Update the UM by running the following command:

```
# cd /root/Insight-<tag_id>

Check the containers lined up for update
# ./insight/insight_runner.py --update --dry-run

Start the update
# ./insight/insight_runner.py --update
```

6. Verify UM update by running the following command:

```
# ./insight/insight_runner.py --update-status

Cisco VIM Insight Update Status!
+-----------------------------------------+--------+---------------------------------+
|              Description                | Status |           Details               |
+-----------------------------------------+--------+---------------------------------+
| VIM Insight Container: insight_<new_tag> | PASS   |  Updated from insight_<old_tag>  |
| VIM Mariadb Container: insight_<new_tag> | PASS   |  Updated from mariadb_<old_tag>  |
+-----------------------------------------+--------+---------------------------------+
```

Upgrade Scenario

For any CVIM UM upgrade to 4.x line up from 3.4.y, CVIM UM version of 3.4.6 is the starting point. The changes to the underlying infrastructure such as docker, RHEL version, and database schema, exist in Cisco VIM UM code when upgrading from versions of 3.4.6 and beyond (in the 3.4.x lineup) to the current release.

Upgrade Cisco VIM UM from 3.4.6 or later (in 3.4.x lineup)

To address the upgrade of major infrastructure change, upgrade the Cisco VIM UM node from 3.4.6 or later (in the 3.4.x lineup) to the current release of Cisco VIM UM node.

To upgrade Cisco VIM Unified Management to the current release, follow the below steps:

1. Get the installer tarball for the current release. Extract the tarball to get the new UM workspace by running the following command:

```
tar --no-same-owner -xvzf mercury-installer.tar.gz
```

2. Create an empty directory to take a backup of Cisco VIM UM node.
3. Run Insight upgrade script with *-b [backup]* option:

```
# cd /root/installer-<tag_id>/tools
# ./insight_upgrade.sh -b -d <path of backup directory>/
```

4. Copy the backup directory to the remote server.
   For example, to copy the backup directory */var/cisco/insight_upgrade_backup/* from the management node to the remote host 20.0.0.5, execute the following command sequence:

```
rsync -e ssh -go -rtvpX --numeric-ids /var/cisco/insight_upgrade_backup/ root@20.0.0.5:/var/cisco
/insight_upgrade_backup
```

> ⚠ Ensure that the path *root@20.0.0.5:/var/cisco/insight_upgrade_backup/* is available at the remote location.

5. Re-image the UM node with the ISO of the current release version and with the same IP address.
6. Copy the backup directory from the remote server to the management node. For example, to copy the backup directory */var/cisco /insight_upgrade_backup/* from remote host 20.0.0.5 to the management node, execute the following command sequence.

```
rsync -e ssh -go -rtvpX --numeric-ids root@20.0.0.5:/var/cisco/insight_upgrade_backup/ /var/cisco
/insight_upgrade_backup/.
```

> ⚠
> - Ensure that the path */var/cisco/insight_upgrade_backup/* is present on UM node.
> - For installation with Internet access, skip Step 9.

7. Create an empty directory for Insight workspace: *Insight-<tag_id>*
8. Copy the contents of the extracted directory *installer-<tag_ig>* within *Insight-<tag_id>*

```
# cp -r installer-<tag_ig>/* Insight-<tag_id>
```

9. To download, test and import the new UM artifacts, follow the steps given in UM Without Internet Access only till Step 11.
10. Run Insight upgrade with *-r [restore]* option:

```
# cd /root/Insight-<tag_id>/tools
# ./insight_upgrade.sh -r -d <path of backup directory>'
```

11. After successful execution of upgrade, use *reconfigure* option of insight_runner to enable new Insight features or *install-status* to check Insight installation status:

```
# cd /root/Insight-<tag_id>

# ./insight/insight_runner.py --install-status
# ./insight/insight_runner.py --reconfigure -f <insight_setup_data.yaml>
```

Rollback UM

Rollback VIM Unified Management

Cisco VIM Unified Management rollback feature allows you to revert to the old UM release which is used before the update.
Following are some of the key points:

- The rollback action removes the new docker containers of Unified Management and mariaDB which is created after an update and bring up old ones with the old tag.
- After rollback, your UM workspace becomes the old workspace that is used before the update.

Following are the steps to perform UM rollback:

1. Run the following command from the new workspace to rollback Cisco VIM UM:

```
# cd /root/Insight-<tag_id>
# ./insight/insight_runner.py --rollback
```

2. Verify the rollback status by running the following command:

```
# ./insight/insight_runner.py --install-status

Cisco VIM Insight Install Status!
+-----------------------+--------+----------------------------------+
| Description           | Status | Details                          |
+-----------------------+--------+----------------------------------+
| VIM Insight Version   | PASS   | <release_tag>                    |
| VIM Insight UI URL    | PASS   | https://<br_api:9000>            |
| VIM Insight Container | PASS   | insight_<tag_id>                 |
| VIM Mariadb Container | PASS   | mariadb_<tag_id>                 |
+-----------------------+--------+----------------------------------+
```

Commit UM

Commit Unified Management

VIM Insight commit supports for a new Insight release after an update. Following are some of the key points:

- The old workspace is not deleted and retained as it is.
- After the commit, your Unified Management workspace which is used for the update becomes the new workspace.

1. Run the following command to commit Cisco VIM Insight:

```
# cd /root/Insight-<tag_id>
# ./insight/insight_runner.py --commit
```

> ⚠ If update is done from Cisco VIM 3.0.x to Cisco VIM 3.4.2, the following warning message to reconfigure Insight is displayed.
>
> **WARNING: Insight setup-data key 'UM_ADMIN_AS_POD_ADMIN' is deprecated from version 3.4.2**
> **Please reconfigure using 'UM_ADMIN_WITH_FULL_POD_ACCESS' as a new name of the key**

2. Verify the commit status by running the following command:

```
# ./insight/insight_runner.py --install-status
Cisco VIM Insight Install Status!
+-----------------------+--------+----------------------------------+
| Description           | Status | Details                          |
+-----------------------+--------+----------------------------------+
| VIM Insight Version   | PASS   | <release_tag>                    |
| VIM Insight UI URL    | PASS   | https://<br_api:9000>            |
| VIM Insight Container | PASS   | insight_<tag_id>                 |
| VIM Mariadb Container | PASS   | mariadb_<tag_id>                 |
+-----------------------+--------+----------------------------------+
```

Migrating of UM to NG Instance

Migrating of UM to NG Instance

- Overview of Next-Generation (NG) Unified Management
- Switching from Old Generation (OG) UM to NG UM
  - Options with UM-NG
  - Switch to UM-NG via Insight Runner
  - Verify Switch to UM-NG

Overview of Next-Generation (NG) Unified Management

Cisco VIM 3.4.6 release comes with a revamped version of Unified Management called Next-Generation (NG) UI with improved usability, performance and efficiency. The newer UI/UX technologies packaged in the application provide you a seamless experience.

Switching from Old Generation (OG) UM to NG UM

Unified Management orchestrator provides you the functionality to migrate over to the Next-Generation (NG) UM from the OG-UM instance. This is a one-way migration procedure. A summary of the options that are allowed as part of the migration with NG UM is listed below:

Options with UM-NG

Case 1: Install UM-NG

- Fresh installation of UM-NG is allowed
- You cannot switch back to UM-OG

Case 2: Update from UM-OG to UM-NG

- Update from UM-OG to UM-NG is supported.
- Switching back from UM-NG to UM-OG is not allowed.

Case 3: Update with UM-OG and UM-NG

- UM update always take place from OG to OG or NG to NG.
- After UM update and commit, Case 1 and Case 2 will apply.

Switch to UM-NG via Insight Runner

On a system running with UM-OG, execute the following commands to switch to UM-NG.

```
# cd /root/Insight-<tag_id>
# ./insight/insight_runner.py --switch NG
```

Verify Switch to UM-NG

To verify whether the UM is switched from OG to NG, use the following commands:

```
# cd /root/Insight-<tag_id>
# ./insight/insight_runner.py --install-status

Cisco VIM Insight Install Status!
+--------------------+--------+-----------------------------------------------+
| Description        | Status | Details                                       |
+--------------------+--------+-----------------------------------------------+
| Insight Setup      | PASS   | Success                                       |
| Insight Version    | PASS   | <NG|OG>: <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' |
+--------------------+--------+-----------------------------------------------+

Check the 'Insight Version' column in above generated output to verify UM 'NG/OG' and its version.

# Verify the Insight container status

# docker ps -a | grep insight
# 65cd9bbde4c6   127.0.0.1:5000/cvim34-rhel7-osp13/insight-ng:<tag_id>    "/start.sh"     3 days ago     Up 34
hours    insight_ng_<tag_id>
```

Shutting Down UM

Shutting Down Cisco VIM Unified Management

To stop the Cisco VIM Unified Management Container services, shut down Cisco UCS VIM Unified Management by running the *systemctl stop service* command.

1. Log into a server in which the Unified Management container is running.
2. Stop the Unified Management service by running the following command from the shell window:

```
systemctl stop docker-insight
```

   a. Check the status of Unified Management Container by running the following command: *docker ps -a* | *grep* insight.

```
STATUS
Exited (137) 1 second ago
```

   b. Check the status of the service by running the following command:

```
systemctl staus docker-insight
```

   The following information is displayed:

```
Docker-insight.service - Insight Docker Service
Loaded: loaded (/usr/lib/systemd/system/docker-insight.service; enabled; vendor preset: disabled)
Active: inactive (dead) since <Date and Time since it was last active>
```

UM Tech Support

Tech Support

Cisco VIM Unified Management node includes a tech-support tool to gather sosreport, logs and state files from the server. The tech-support tool creates a single tarball of gathered information which can then be shared with Cisco Technical Support to debug and solve issues found in production. This tool uses a configuration file that specifies the information to be collected. You can add custom scripts or commands in this configuration file, to gather more information as needed.

The configuration file is available at:

```
/root/insight-openstack-configs/tech-support/tech_support_um.yaml
```

This tool checks the point where the Cisco VIM UM is executed, and collects the output of files or commands indicated by the configuration file. The technical support script is available at the following location:

```
/root/insight-{tag-id}/tech-support/ directory.
```

To run the script, enter the following command from */root/insight-{tag-id}* directory:

```
[root@santacruz1 insight-28232]# ./tech-support/tech_support um
Going to collect the info about this UM server. Please wait!


Reading config file '/root/insight-28232/openstack-configs/tech-support/tech_support_um.yaml'
Created tar file '/var/log/cisco/tech-support/santacruz1/tech-support-f1S1np/tech_support-santacruz1-20200831-
095000.tar'
Compressed file /var/log/cisco/tech-support/santacruz1/tech-support-f1S1np/tech_support-santacruz1-20200831-
095000.tar.xz created
The compressed file is at: /var/log/cisco/tech-support/santacruz1/tech_support-santacruz1-20200831-095000.tar.xz


Successfully created tech-support compressed file at /var/log/cisco/tech-support/tech_support-20200831-095000.
tar.xz
```

BIOS/BMC/Firmware Update

Update of BIOS/BMC/Firmware

- BIOS/BMC/Firmware Update Overview
- Cisco UCS Firmware Upgrade
- Quanta Firmware Upgrade
- Intel FPGA PAC N3000 Firmware Update Support

BIOS/BMC/Firmware Update Overview

Overview of BIOS/BMC/Firmware Update

To provide a seamless operation experience, tools to upgrade/update the BIOS/BMC/firmware for Cisco UCS C-series and Quanta servers are provided.

For Cisco UCS-C series, the tool uses the HUU image provided by Cisco to update the CIMC and firmware.

For Quanta, the support for automated BIOS/BMC/firmware upgrade is available. This utility is not available for HP or Dell computes.

Cisco UCS Firmware Upgrade

Cisco UCS Firmware Upgrade

- Limitations
- Tools Usage

In Cisco VIM 2.4.2, the Cisco Host Upgrade Utility (HUU) tool developed using Cisco Integrated Management Controller (IMC) Python SDK module (imcsdk-0.9.2.0) is leveraged automatically to upgrade all firmware components running on Cisco UCS C-Series servers.

The wrapper tool only updates the CIMC bundle packages, as the entire Cisco IMC Software bundle (that includes CIMC, BIOS, adapter and storage controller firmware images through HUU images) is updated by default. Adequate planning is required for CIMC upgrade, as it causes the server to get rebooted.

For Cisco VIM 2.4, the CIMC upgrade tool supports the:

- Upgrade of CIMC bundle images for C-series only.
- Concurrent upgrade of CIMC bundle images on multiple C-series servers.
- Pre-validation check for server type and available HUU images.
- Support of the external http server, Cisco VIM Software Hub, or Cisco VIM Management node for the target CIMC upgrade image mounts.
- Checks if the cloud is deployed successfully, and notifies the user with a proper message.
- Checks if selected hosts have any active VMs running on them and notifies the user with a proper message.
- Generation of consolidated summary on firmware version status, on completing the pre-upgrade and post-upgrade.

- Firmware upgrade is supported only on UCS C-series platform and not on Dell or HP platforms.
- If you upgrade CIMC firmware on an existing cloud deployment, it might impact the cloud functionality as the firmware upgrade reboots the host. Hence, ensure that the cloud is operational, post CIMC firmware upgrade.

To check if the cloud is operational, execute the following steps:

- Run the cloud sanity.
- If cloud sanity failure occurs, run cluster recovery and then re-run cloud sanity.

Also for the upgrade operation to work, ensure that the image has the following syntax:

```
ucs-<server_type>huu<version_number>.iso; for example ucs-c220m4-huu-2.0.13n.iso or ucs-c240m4-huu-2.0.13n.iso;
```


Running the UCS Firmware upgrade on host(s) running active VMs results in downtime on those VMs.

Limitations

The following Cisco VIM management operations are not allowed when the firmware upgrade is in progress:

- POD management operations: Addition, removal or replacement of nodes.
- Cisco VIM software update.
- Cisco VIM software upgrade.
- Reconfiguration of Cisco VIM features.

Tools Usage

The CIMC upgrade utility is a standalone python file (ucsc_host_upgrade_utility) which is located under `<cvim_install_dir>/tools/` directory.

To use the tool, execute the following command:

```
[root@hiccup-mgmt-228 tools]# python ucsc_host_upgrade_utility.py -h
usage: ucsc_host_upgrade_utility.py [-h] [--file SETUPFILELOCATION]
                                                                    [--http-server HTTP_SERVER_IP]
                                                                    [--sds-server SDS_SERVER_NAME]
                                                                    [--server-uname UNAME]
                                                                    [--server-pwd PASSWD]
                                                                    [--huu-image-path HUU_IMAGE_PATH]
                                                                    [--host HOSTS] [--exclude-hosts E_HOSTS]
                                                                    [-y]
Script to perform firmware upgrade
optional arguments:
-h, --help                                  show this help message and exit
--file SETUPFILELOCATION, -f SETUPFILELOCATION
                                                    Optional, if not defined will read the setup_data.yaml in /root
/openstack-configs dir for CIMC information of servers;
                                                     To override, provide a valid YAML file with the CIMC
Credentials.
--http-server HTTP_SERVER_IP, -hs HTTP_SERVER_IP
                                                    Optional, only needed if a http server is used to host the
target CIMC bundle image(s).
--sds-server SDS_SERVER_NAME, -sds SDS_SERVER_NAME
                                                    Optional, only needed if a Software Distribution Server (SDS)
is used to host the target CIMC bundle image(s).
--server-uname UNAME, -u UNAME
                                                    Optional, only needed if a http server is used to host the
target CIMC bundle image(s).
--server-pwd PASSWD, -p PASSWD
                                                    Optional, only needed if a http server is used to host the
target CIMC bundle image(s).
--huu-image-path HUU_IMAGE_PATH, -path HUU_IMAGE_PATH
                                                    Comma separated absolute path of the HUU ISO file(s);
                                                    In the case of a web server hosting the files, provide the
absolute path of the URL that includes the file names; that is, exclude
                                                    the scheme://host/ part

--host HOSTS            Comma separated list of host names targeted for CIMC bundle upgrade defined in the
target setup_data.yaml
--exclude-hosts E_HOSTS, -e E_HOSTS
                                                    Comma separated list of hostnames excluded for CIMC bundle
upgrade defined in the target setup_data.yaml
-y, -yes
[root@hiccup-mgmt-228 tools]#
```

If the target CIMC upgrade images are available on Cisco VIM Management node, use the below command:

```
python ucsc_host_upgrade_utility.py [--file <setup_data_test.yaml/cimc_servers.yaml>] -path <huu_image_paths>
```

If the target CIMC upgrade images are hosted in an external http server that is reachable from the Cisco VIM Management node and CIMC of the servers, use the below command:

```
python ucsc_host_upgrade_utility.py [--file <setup_data_test.yaml/cimc_servers.yaml>]
-hs <http_server_ip/hostname> -u
<https_server_un> -path <http_server_pwd> -path <huu_image_paths>
```

If the target CIMC upgrade images are hosted in Ciso VIM Software Hub, use the below command:

```
python ucsc_host_upgrade_utility.py --file [setup_data_test.yaml/cimc_servers.yaml] -sds
[Ciso VIM Software Hub_server_ip/hostname] -u
[sds_server_un] -path [sds_server_pwd] -path [huu_image_paths]
```

Prerequisites to use Ciso VIM Software Hub for hosting the target CIMC bundle image(s) are:

- Ciso VIM Software Hub server must be reachable from the management node over HTTPS.
- Ciso VIM Software Hub server TLS certificate must be trusted by the management node to make TLS connection in verified context

If setup_data.yaml file is not available, you can create it using the below command:

```
# UCSC (C-series) sample format of yaml file to specify the CIMC details
SERVERS:
server-1:
cimc_info:
cimc_ip: "cimc-ip-address"
cimc_username: "cimc-user-name"
cimc_password: "cimc-password"
server-2:
cimc_info:
cimc_ip: "cimc-ip-address"
cimc_username: "cimc-user-name"
cimc_password: "cimc-password"
server-3:
cimc_info:
cimc_ip: "cimc-ip-address"
cimc_username: "cimc-user-name"
cimc_password: "cimc-password"
```

As the upgrade of CIMC takes more than an hour, execute this process in a VNC. If you do not have a VNC environment, execute the same from KVM console of the management node. Do not run this command in background or with nohup option.

Quanta Firmware Upgrade

Quanta Firmware Upgrade

- Limitations
- Tools Usage

From release Cisco VIM 3.2.1, the Quanta firmware upgrade utility tool upgrades the BMC, BIOS, and NVM firmware versions on Quanta servers.

> ⚠ The wrapper tool only updates the BMC, BIOS, and NVM versions based on the provided firmware packages. Adequate planning is required for Quanta firmware upgrade, as it causes the server to get rebooted.

For Cisco VIM 3.2.1, the Quanta firmware upgrade tool supports:

- Upgrade of BMC, BIOS, and NVM for Quanta servers only.
- Concurrent upgrade of firmware on multiple Quanta servers.
- Pre-validation if another firmware flashing is in progress.
- Generation of consolidated summary on firmware version status, on completing the pre-upgrade and post-upgrade.

> ⚠ If you upgrade Quanta firmware on an existing cloud deployment, it might impact the cloud functionality as the firmware upgrade reboots the host. Hence, ensure that the cloud is operational, post Quanta firmware upgrade.

To check if the cloud is operational, do the following:

1. Run the cloud sanity.
2. If cloud sanity fails, run cluster recovery and then re-run the cloud sanity.

For the upgrade to work, ensure that you specify the proper firmware-package bundle with proper name and version of the components (BMC/BIOS/NVM). Few examples given below:

- BIOS firmware package file name must start with a prefix BIOS_XXXX.zip.
- BMC firmware package file name must start with a prefix BMC_XXX.zip.
- NVM firmware package file name must contain NVMXXXXX.zip.

> ⚠ Running the BMC/BIOS/NVM firmware upgrade on host(s) running active VM results in downtime on those VMs.

Limitations

The following Cisco VIM management operations are not allowed when the firmware upgrade is in progress:

- Pod management operations: Addition, removal or replacement of nodes.
- Cisco VIM software update.
- Cisco VIM software upgrade.
- Reconfiguration of Cisco VIM features.

Tools Usage

The Quanta upgrade utility is a standalone python file `quanta_firmware_upgrade.py` which is located under `<cvim_install_dir>/tools/` directory.

To use the tool, execute the following command:

```
[root@mgmt-node tools]# ./quanta_firmware_upgrade.py -h
usage: quanta_firmware_upgrade.py [-h] [--hosts HOSTS]
                                  [--exclude-hosts EXCLUDE_HOSTS]
                                  [--ignore-flash-status] [--no-preserve] [-v]
                                  [-y] [--bmc-only | --bios-only | --nvm-only]
                                  --setupfile SETUPFILE --firmware-package
                                  FIRMWARE_PACKAGE

Script to update BMC and BIOS firmware on Quanta server

optional arguments:
  -h, --help            show this help message and exit
  --hosts HOSTS         comma separated list of servers defined in
                        setup_data.yaml file target for firmware update
  --exclude-hosts EXCLUDE_HOSTS
                        comma separated list of servers defined in
                        setup_data.yaml file to exclude for firmware update
  --ignore-flash-status
                        ignore flash status, needed for older BMC firmware
                        where it lacks the support and continue with BMC/BIOS
                        firmware update
  --no-preserve         do not preserve BMC or BIOS setting during firmware
                        update
  -v, --verbose         enable verbose output
  -y, --yes             skip prompt
  --bmc-only            update BMC firmware only
  --bios-only           update BIOS firmware only
  --nvm-only            update Intel Ethernet NVM only

required arguments:
  --setupfile SETUPFILE
                        setup_data.yaml file location
  --firmware-package FIRMWARE_PACKAGE
                        Firmware package ZIP file location
```

For upgrade BMC only, use the below command:

```
./quanta_firmware_upgrade.py --setupfile <setup-data.yaml> --firmware-package <bmc/bios_update_package.zip> --
bmc-only
```

For upgrade BIOS only, use the below command:

```
./quanta_firmware_upgrade.py --setupfile <setup-data.yaml> --firmware-package <bmc/bios_update_package.zip> --
bios-only
```

For upgrade NVM only, use the below command:

```
./quanta_firmware_upgrade.py --setupfile <setup-data.yaml> --firmware-package <nvm_update_package.zip> --nvm-
only
```


- If you do not specify any option for particular component with specified BIOS/BMC firmware package, the utility upgrades in the order of BMC followed by BIOS but not NVM.
- If you are planning to update both the BMC and the BIOS separately, always update the BMC first and then the BIOS.
- To upgrade NVM, you need to specify --nvm-only option with NVM firmware package, otherwise upgrade fails.
- NVM firmware upgrade does not reboot the servers by default. You need to reboot the nodes manually to complete the upgrade process.

As some of the servers running old firmware are upgrading to latest firmware, password might be reset to default.
You need to manually reset the password by following the below steps:

1. Login to the UI using the factory default **admin** and **cmb9.admin**
2. Once logged-in, you are prompted with a message **First Login or Password Expired. You need to change your password**.

You need `--ignore-flash-status` option for older BMC firmware as it lacks support, and continue with BMC firmware update.
Example

```
./quanta_firmware_upgrade.py --setupfile <setup-data.yaml> --firmware-package <bmc_update_package.zip> --bmc-only --ignore-flash-status
```


As the upgrade of BIOS/BMC takes more than 30 minutes, it is imperative to execute this process in a VNC. If you do not have a VNC environment, execute the same from KVM console of the management node. Do not run this command in background or with nohup option.

Intel FPGA PAC N3000 Firmware Update Support

Intel FPGA Programmable Acceleration Card (PAC) N3000 Firmware Update Support

- Overview
  - Method 1: Fresh installation of Cisco VIM 3.4.3
  - Method 2: Using Script packaged in Cisco VIM's tools directory
- Script Usage
- Sample Configuration
  - Configuration for User-provided Image and XL710
  - Configuration for Image Files
- Sample Output

Overview

Intel FPGA PAC N3000 firmware update script updates all the accelerated compute nodes installed with Intel N3000 card.

The tool checks and performs the following:

- Upgrade from version 1.0 non-secure mode to 1.1 secure mode support.
- Program the unsigned user image into FPGA's user flash region.
- Update the onboard XL710 chips to user provided firmware.

Two ways to update the firmware are:

- Fresh installation of Cisco VIM 3.4.3.
- Manual update using the script packaged in Cisco VIM's tools directory.

Method 1: Fresh installation of Cisco VIM 3.4.3

1. Add the following configuration to *setup_data.yaml* configuration file before pod deployment:

```
setup_data.yaml
===============
<..snip..>
# Optional section, Intel FPGA N3000 firmware version control configuration:
# Specify particular unsigned user image to flash N3000's FPGA and/or update
# N3000's XL710 firmware during install process. As part of unsigned user
# image flashing process, N3000 card will also be upgrade to version 1.1 to
# support secure mode.
# Can specify 1) just "user_image_bitstream_id" and "user_image_file" or
#             2) just "xl710_config_file" and "xl710_image_file" or
#             3) both 1 and 2 options
#
# IMPORTANT: User will have to supply the firmware files and have it placed in
# /root/openstack-configs/ directory.
INTEL_N3000_FIRMWARE:
  user_image_bitstream_id: '<user image bitstream id, starting with 0x>'
  user_image_file: <unsigned user image file; filename only, no path>
  xl710_config_file: <XL710 nvmupdate config file; filename only, no path>
  xl710_image_file: <XL710 nvamupdate image file; filename only, no path>
<..snip..>
```

2. Copy the unsigned user image obtained from the vendor to */root/openstack-configs/* directory.
3. Copy the XL710 configuration and image files to */root/openstack-configs/* directory.
4. Perform pod deployment and update the Intel N3000 firmware during installation.

> ⚠ To complete an update, an additional time of two hours is taken.

Method 2: Using Script packaged in Cisco VIM's tools directory

1. Create an one-time user configuration file containing information about the firmware update:

```
config_file.yaml
================
# Can specify 1) just "user_image_bitstream_id" and "user_image_file" or
#             2) just "xl710_config_file" and "xl710_image_file" or
#             3) both 1 and 2 options
#
# IMPORTANT: You must supply the firmware files and have it placed in
# /root/openstack-configs/ directory.
INTEL_N3000_FIRMWARE:
  user_image_bitstream_id: '<user image bitstream id, starting with 0x>'
  user_image_file: <unsigned user image file; filename only, no path>
  xl710_config_file: <XL710 nvmupdate config file; filename only, no path>
  xl710_image_file: <XL710 nvamupdate image file; filename only, no path>
```

2. Copy the XL710 configuration and image files to */root/openstack-configs/* directory
3. On the already deployed pod, run following script in the Cisco VIM installer's tools directory:

```
cd /root/installer-3.4.3/tools
./intel-fpga-n3000-firmware-update.py -c /root/config_file.yaml
```

⚠ Depending on the number of cards and firmwares to flash, it may take up to two hours to complete.

Script Usage

```
[root@mgmt-node tools]# ./intel-fpga-n3000-firmware-update.py -h
usage: intel-fpga-n3000-firmware-update.py [-h] [-c CONFIG_FILE]
                                           [--check-only] [--hosts HOSTS] [-y]
                                           [-v]

Script to update Intel FPGA N3000 firmware on all Compute node

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_FILE, --config-file CONFIG_FILE
                        one-time use config file containing just
  INTEL_N3000_FIRMWARE  section relevant to Intel FPGA
                        N3000 firmware update, please refer to
                        setup_data.yaml.C_Series_EXAMPLE for more info
  --check-only          perform firmware version check only
  --hosts HOSTS         comma separated list of Compute node defined in
                        setup_data.yaml file target for Intel FPGA N3000
                        firmware update
  -y, --yes             assumed yes, skip prompt
  -v, --verbose enable  verbose output
```

Sample Configuration

Configuration for User-provided Image and XL710

```
INTEL_N3000_FIRMWARE:
  user_image_bitstream_id: '0x2392920A010501'
  user_image_file: phase1_turbo4g_2x1x25g_1fvl_raw_20ww02_unsigned.bin
  xl710_config_file: nvmupdate_25G_0D58.cfg
  xl710_image_file: PSG_XL710_7p00_CFGID2p61_XLAUI_DID_0D58_K32246_800052B0.bin
```

Configuration for Image Files

```
[root@mgmt-node ~]# ls -l /root/openstack-configs/* | grep -E "\.bin|\.cfg"
-rw-r--r--. 1 root root      435 Feb 27 18:53 /root/openstack-configs/nvmupdate_25G_0D58.cfg
-rw-r--r--. 1 root root 45089792 Feb 20 12:20 /root/openstack-configs
/phase1_turbo4g_2x1x25g_1fvl_raw_20ww02_unsigned.bin
-rw-r--r--. 1 root root  4194304 Feb 27 18:54 /root/openstack-configs
/PSG_XL710_7p00_CFGID2p61_XLAUI_DID_0D58_K32246_800052B0.bin
```

Sample Output

```
[root@mgmt-node tools]# ./intel-fpga-n3000-firmware-update.py -c /root/phase1_firmware.yaml --hosts quincy-
compute-1,quincy-compute-2


Starting Intel FPGA N3000 update script
Full log can be found at /var/log/mercury/n3000_firmware_update-20200228151100673979.log
Checking Intel FPGA N3000 firmware on accelerated Compute node(s)
+------------------+--------------+---------------------+------------------+
|      Server      |   Management  | Accelerated Compute | Update Available |
+------------------+--------------+---------------------+------------------+
| quincy-compute-1 | 10.11.216.101 |         Yes         |        Yes       |
| quincy-compute-2 | 10.11.216.102 |         Yes         |        Yes       |
+------------------+--------------+---------------------+------------------+
Following accelerated Compute node(s) will be impacted as part of firmware update:

quincy-compute-1,quincy-compute-2


Any Nova, Neutron, and Telegraf services running on accelerated Compute node(s) will be stopped!!!
At the end of firmware update, accelerated Compute node(s) will be power cycle!!!
Would you like to continue? <y|n> y


Starting Intel FPGA N3000 firmware update process, this may take up to 2+ hours!!!
Successfully executed Intel FPGA N3000 firmware update script
```

Configuration

Configuration

- Setup Configuration File
- ToR Management
- Servers and Network Option
- Compute Level Options
- OpenStack Configuration
- VPP VLAN
- L3 Fabric Deployment Details

Setup Configuration File

Setup Configuration File

To make the Cisco VIM installation simple, repeatable, and predictable for the end user, the *setup_data,yaml* is used to drive the cloud configuration. Depending on its nature, some of the parameters must be set on Day 0, while others can be set and enabled in the cloud via a reconfigure option. You can install and configure the Cisco VIM deployment using the *setup_data.yaml.*

If you are installing Cisco Insight, see Unified Management

Ensure that you take extreme care while creating the configuration file, as certain changes (for example, network, API) in the configuration after deployment, causes a stack redeployment with the exception of adding and removing nodes.

Any change done to the pod networking layout plan configured in *setup_data.yaml* requires the pod to be reinstalled.

If your configuration is correct, the installation is done smoothly.

Recommendations

- Use a YAML editor on Linux (PyCharm, Komodo or vi/vim with YAML plugin) to edit this file. Do not copy the examples shown below into your YAML file, as your browser might render the characters differently.
- Generate the file in a Linux environment.

If you are using the Cisco VIM installer, you cannot update the OpenStack configuration files (for example, *ml2_conf.ini,* and other files) directly. All OpenStack configurations must be available in the *setup_data.yaml* file. This ensures that the installer with a view of the OpenStack deployment can reliably perform pod management, software updates, and upgrades. This ensures a consistent and repeatable installation.

ToR Management

ToR Management

- Overview
- Auto-configuration
- Configuration Setup for C-series with N9K as ToR
    - Server-level setup-data Information for C-series
    - Server-level setup_data Information for C-series with Intel NIC
    - Server-level setup_data Information for C-series with Intel NIC using SRIOV
- Custom N9K Configuration Support
- ToR Configuration for Cisco NCS 5500
    - Auto-ToR of Cisco NCS 5500 for LACP Management
- Customization of Cisco NCS 5500 Configuration for Ethernet Segment ID and Route-Target
- NCS Day 0 Configuration (Prior to Cisco VIM installation)
- Prerequisites for Segment Routing Global Block and ISIS Prefix
- Prerequisites for API and External Network Segments with Cisco NCS 5500 as ToR
- Support and Pre-requisites for Provider Network with NCS
- Prerequisites for Provider Network with NCS 5500 as ToR
- Cisco VIM with NCS-5000 as ToR in "ships in the night" model
- ToR Configuration with ACI Fabric

Overview

For the servers to participate in the cloud, they must be connected to Top of Rack (ToR). Cisco VIM is agnostic to the ToR type, as long as you configure the right VLANs (OpenStack control, data plane) on the relevant ports of the ToR. Also, there exists a Layer 2 domain for the servers to PxE boot. However, if you choose to have Nexus as the ToR with or without ACI as the fabric, auto-configuration of the ToRs is possible. If the ACI plugin or Cisco NCS is used as ToR, the auto-ToR configuration is mandatory.

Auto-configuration

Cisco VIM provides complete automation of the cloud deployment. It automates Day 0 configuration of N9xxx series ToR switches. This feature is optional and applicable to the pods running with or without ACI, while using N9K as ToR. The auto-ToR feature simplifies the deployment, but has restrictions on the configuration of the switch ports connected to the servers. Also, L3-out and spine level configurations are outside the scope of Cisco VIM's automation. For ToR switch details related to ACI, see Enabling Auto-ToR via ACI API.

Cisco VIM automates post Power-On Auto Provisioning (post-POAP) configuration on ToR with one or more pairs of identical Cisco N9K series switches. The Day 0 ToR automation configures the interfaces that are connected to the management *(br_mgmt),* control, compute, and storage nodes of the pod. In addition, it configures the VPC peer link interfaces for ToR pairs. The automation handles C-series pods and manages the configuration of the edge ports in the leaf switches of which the hosts hang-out and the VPC peer link between the switches.
The auto-configuration feature does not include the configuration of the spine switches and the connectivity between the leaf and the spine or the upstream link of the spine switches that carry the external VLAN connectivity.

As the feature is a post-POAP automation provisioning, ensure that the management interface, vrf, and admin user are pre-provisioned on each ToR switch. Also, you must enable SSH to each ToR from the management node.

The minimum N9K switch software versions is 9.3(3).

Bootstrapping the ToR image is still a manual process. Ensure that the installer API interface *(br_api)* is up and running on the management node with SSH enabled. You can access each ToR through its management interface from the Cisco VIM management node using SSH.

Configuration Setup for C-series with N9K as ToR

The automated ToR configuration details are provided in two parts in the mercury *setup_data.yaml* file. The common information is available in the TORSWITCHINFO section, whereas the information on individual switch ports connected to specific nodes is available under SERVERS section for Rack based servers.

If the TORSWITCHINFO section is not provided or CONFIGURE_TORS attribute under TORSWITCHINFO is set to False or not provided, all the ToR provisioning related steps are skipped. The ToR section contains the attributes related to ToR connection, configuration for the management node interface, and vPC peer details in case of ToR pairs.

The port-channel number for the vPC peer link interfaces is derived from the vpc domain. The ToRs are paired with each other based on their corresponding *vpc_peer_link* addresses.

```
TORSWITCHINFO:
 CONFIGURE_TORS: True
 SWITCHDETAILS:
 -
   hostname: K09-n9k-a # mandatory
   username: admin # mandatory
   password: <ssh password> # mandatory
   ssh_ip: <a.b.c.d> # mandatory
   ssn_num: <xyz>. # optional, output of show license host-id
   vpc_peer_keepalive: <f.g.h.i> # peer switch ssh ip
   vpc_domain: <int>
   vpc_peer_port_info: <'eth1/45,eth1/46,eth1/47'>
   vpc_peer_vlan_info: <'NNNN,NNNN-NNNN'> # optional
   br_mgmt_port_info: 'eth1/19'. # only in 1 switch pair
   br_mgmt_po_info: <'NN'> # only in 1 switch pair
 -
   hostname: K09-n9k-a # mandatory
   username: admin # mandatory
   password: <ssh password> # mandatory
   ssh_ip: <a.b.c.d> # mandatory
   ssn_num: <xyz>. # optional, output of show license host-id
   vpc_peer_keepalive: <f.g.h.i> # peer switch ssh ip
   vpc_domain: <int>
   vpc_peer_port_info: <'eth1/45,eth1/46,eth1/47'>
   vpc_peer_vlan_info: <'NNNN,NNNN-NNNN'> # optional
   br_mgmt_port_info: 'eth1/19'. # only in 1 switch pair
   br_mgmt_po_info: <'NN'> # only in 1 switch pair
```


- Attributes for vpc peer vlan info and vpc domain have to match across each ToR pair.
- Attributes br_mgmt_po_info and br_mgmt_port_info have to match across ToR pairs. However, they must be defined only in ToR pairs where the management node is hanging off.
- Attribute for vpc_peer_vlan_info is optional.If it is not specified, it derives a list of VLAN ids from the host or FI facing interfaces and br_mgmt interface.
- Attribute for ssn_num which represents the chassis serial number is optional.

You can obtain the chassis serial number by executing the following command on each of the ToRs:

```
show license host-id
```


ToR names need to match with names provided in the TORSWITCHINFO section.

Server-level setup-data Information for C-series

For C-series based pod, the switch port interface mapping for each of the controller, compute, and storage nodes are defined in the SERVERS > tor_info section. Cisco VIM driven cloud has a notion control (samx) and data (pet). Information present under tor_info always reflects the switch ports participating in OpenStack control (samx) pet. For deployments where the control and data plane are collapsed (for example, pods running on Cisco VIC), the tor_info represents the switch ports that participate in both the roles.

```
SERVERS:
 controller-1:
   rack_info: {rack_id: rack43X} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: <int>, B9-TOR-9K-1: eth1/<int>, B9-TOR-9K-2: eth1/<int>}
 controller-2:
   rack_info: {rack_id: rack43Y} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: 7, B9-TOR-9K-1: eth1/7, B9-TOR-9K-2: eth1/7}
 controller-3:
   rack_info: {rack_id: rack43Z} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: 9, B9-TOR-9K-1: eth1/9, B9-TOR-9K-2: eth1/9}
 compute-1:
   rack_info: {rack_id: rack43} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: 11, B9-TOR-9K-1: eth1/11, B9-TOR-9K-2: eth1/11}
 compute-2:
   rack_info: {rack_id: rack43} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: 13, B9-TOR-9K-1: eth1/13, B9-TOR-9K-2: eth1/13}
 storage-1:
   rack_info: {rack_id: rack43} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: 14, B9-TOR-9K-1: eth1/14, B9-TOR-9K-2: eth1/14}
 storage-2:
   rack_info: {rack_id: rack43} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: 15, B9-TOR-9K-1: eth1/15, B9-TOR-9K-2: eth1/15}
 storage-3:
   rack_info: {rack_id: rack43} cimc_info: {cimc_ip: <ip_addr>}
   tor_info: {po: 16, B9-TOR-9K-1: eth1/16, B9-TOR-9K-2: eth1/16}
```

VLANs for host facing interfaces are derived from NETWORK section, based on the server ROLES definition of each server and their corresponding network profile roles assigned for each segment.

Server-level setup_data Information for C-series with Intel NIC

When you configure a C-series pod to run in a complete Intel NIC environment, the OpenStack control (samx) and data (pet) plane are on different switch ports. As a result, an additional section called *dp_tor_info* is introduced to reflect the switch relevant port to server mapping, that is, control plane and data plane traffic are broken out into two separate interfaces with relevant VLANs (control plane VLANs for tor_info and data plane VLAN for dp_tor_info) applied on each of the interfaces facing the controller and compute nodes.

```
c43b-control-1:
 rack_info: {rack_id: rack43}
 cimc_info: {cimc_ip: <ip_addr>}
 tor_info: {po: 9, K09-n9k-a: 'eth1/9, eth1/12'}
 dp_tor_info: {po: 12, K09-n9k-a: 'eth1/12, eth1/12'}
c43b-compute-1:
 rack_info: {rack_id: rack43} cimc_info: {cimc_ip: <ip_addr>}
 tor_info: {po: 10, K09-n9k-a: 'eth1/10, eth1/13'}
 dp_tor_info: {po: 13, K09-n9k-a: 'eth1/13, eth1/13'}
```

Server-level setup_data Information for C-series with Intel NIC using SRIOV

When you configure a C-series pod to support SRIOV with Intel NIC, a third construct is introduced to allow SRIOV traffic onto the compute nodes. Switch ports configured for SRIOV are not placed in a port-channel. VLAN applied to the SRIOV interface is limited to that of the provider VLANs that are defined under the PROVIDER_VLAN_RANGES in the *setup_data.yaml* file.

```
c43b-compute-1:
 rack_info: {rack_id: rack43}
 cimc_info: {cimc_ip: <ip_addr>}
 tor_info: {po: 10, K09-n9k-a: 'eth1/10, eth1/13'}
 dp_tor_info: {po: 13, K09-n9k-a: 'eth1/13, eth1/13'}
 sriov_tor_info: { K09-n9k-a: eth1/33, K09-n9k-b: eth1/33}
```


For a dedicated storage node, only tor_info is applicable. In case of micro or hyper-converged pod, it is the controllers and computes that define the storage node switch mapping, respectively.

Custom N9K Configuration Support

Custom configuration is an optional procedure. The *setup_data.yaml* file contains an optional CUSTOM_CONFIG section to support custom configuration. Under the CUSTOM_CONFIG section, you can add raw CLI commands at the global, port-channel, and switch port level. CUSTOM_CONFIG is applied at the time of bootstrapping and add-interface workflow steps. You must provide the right switch configuration as the orchestration blindly applies it to the switch.

A sample snippet of the *setup_data.yaml* file with custom configuration is given below:

```
TORSWITCHINFO:
CONFIGURE_TORS: true CUSTOM_CONFIG:

GLOBAL:

[<'cli line 1'>,

<'cli
line 2'>,] PORTCHANNEL:

[<'cli
line 1'>] SWITCHPORT:

[<'cli
line 1'>,

<'cli line 2'>,]
```

ToR Configuration for Cisco NCS 5500

The following caveats apply to a Cisco VIM deployment with Cisco NCS as ToR:

- BGP: For a fresh installation of Cisco VIM, ensure that no BGP configuration is present on the Cisco NCS, otherwise the peering between the two Cisco NCS does not come up properly. Remove the existing BGP configuration, if any. If additional BGP complementary configuration is needed, add it after a successful Cisco VIM installation.
- Segment-Routing: The global block of segment routing IDs must be pre-defined by the admin. Ensure that the prefix defined within the *setup_data.yaml* is within the segment routing global block range.
- Cisco NCS Interface Naming: A set of interface naming variations exist, but supports only the following: [Te0/0/0/0, TenGigE0/0/0/0, Gi0/0/0/0, Hu0/0/1/0, HundredGigE 0/0/1/0, FortyGigE0/0/0/0].
- Any manual adjustments to the ISIS or L2VPN sections (on top of the configuration provided by the Cisco VIM automation) cause subsequent Cisco VIM installation to fail.
- As only a pair of Cisco NCS 5500 is supported, ensure that you choose the right Cisco NCS model.
- With Cisco NCS as ToR, only Cisco C-series servers on Intel NIC are supported.
- The ssh time-out in each of the Cisco NCS must be set to maximum possible value.

  ```
  RP/0/RP0/CPU0:tor-2(config)#ssh server rate-limit 600
  RP/0/RP0/CPU0:tor-2(config)#commit
  ```


From release Cisco VIM 3.4.6, auto-ToR is no longer a must when running with Cisco NCS 5500.

To drive the configuration of Cisco NCS 5500, add the following three sections in *setup_data.yaml* file:

Section	Description
TORSWITCHINFO	This section provides general information.
SERVERS	(For C-series) This section provides information on the switch ports that are connected to the specific nodes. When the Micropod is configured to run in a complete Intel NIC environment with Cisco NCS 5500 as the ToR, the SERVER level configurations include *tor_info* (for control plane) and *dp_tor_info* (data plane) sections. Control plane and data plane traffic are broken out into two separate interfaces with bridge domains applied on each of the control and data interfaces facing each for the controller and compute nodes. *sriov_tor_info* is defined for computes participating in SRIOV.
MULTI_SEGMENT_ROUTING_INFO	This section provides information related to routing.

Cisco NCS 5500 supports a Full or Micropod with additional computes running Intel 710 NICs with VPP as mechanism driver.

Cisco VIM supports only the use of two Cisco NCS 5500 within a single pod, for managing the configuration of Cisco NCS 5500.

The following snippet is an example of the *setup_data.yaml* file which is used to configure Cisco NCS 5500 that act as ToRs for the pod.

```
TORSWITCHINFO:

  CONFIGURE_TORS: true # Mandatory
  TOR_TYPE: NCS-5500   #Mandatory

  SWITCHDETAILS:

 -
   hostname: <NCS-5500-1>   # hostname of NCS-5500-1
   username: admin
   password: <ssh_password of NCS-5500-1>
   ssh_ip: <ssh_ip_address of NCS-5500-1>
   vpc_peer_keepalive: <ssh IP address of the peer NCS-5500-2>
   br_mgmt_port_info: <interface of which br_mgmt of management node is hanging of NCS-5500-1>
   br_mgmt_po_info: <int; bundle Ethernet interface to pxe the management node>
   vpc_peer_port_info: <local interface to which peer NCS-5500 is connected, "," separated, max of 2 entries>' >
   vpc_peer_port_address: <local address with mask for vpc_peer_port_info, "," separated, max of 2 entries>'
can have a mask of /31>
   isis_loopback_addr: <local isis loopback interface address without mask> # assumes /32
   isis_net_entity_title: <isis network_entity_title>
   isis_prefix_sid: <int between 16000-1048575> # must be unique in ISIS domain based on the global segment
routing block defined by the admin

 -

  hostname: <NCS-5500-2> # hostname of NCS-5500-2
  username: admin
  password: <ssh_password of NCS-5500-2>
  ssh_ip: <ssh_ip_address of NCS-5500-2>
  vpc_peer_keepalive: <ssh IP address of the peer NCS-5500-1>
  br_mgmt_port_info: <interface of which br_mgmt of management node is hanging of NCS-5500-2>
  br_mgmt_po_info: <int; bundle Ethernet interface to pxe the management node>
  vpc_peer_port_info: <local interface to which peer NCS-5500 is connected>,"," separated, max of two entries
  vpc_peer_port_address: <local address with mask for vpc_peer_port_info>,"," separated, max of two entries
  isis_loopback_addr: <local isis loopback interface address without mask> # assumes /32
  isis_net_entity_title: <isis network_entity_title>
  isis_prefix_sid: <int between 16000-1048575> Must be unique in ISIS domain based on the global segment
routing block defined by admin.
                        Not allowed, when ESI_PREFIX is defined.

  splitter_opt_4_10: 'FortyGigE<C/D/X/Y>,HundredGigE<E/F/A/B>' # Optional for NCS-5500, only when splitter is
needed on per switch basis (that is, the peer switch may or may
                                                              not have the entry)

SERVER SECTION FOR C SERIES:

SERVERS:
  a27-fretta-micro-1:
    cimc_info: {cimc_ip: 172.28.121.172}
    dp_tor_info: {NCS-5500-1: TenGigE0/0/0/1, NCS-5500-2: TenGigE0/0/0/1, po: 1}
    hardware_info: {VIC_slot: MLOM}
    rack_info:{rack_id: RackA}
    tor_info: {NCS-5500-1: TenGigE0/0/0/0, NCS-5500-2: TenGigE0/0/0/0, po: 2} # Optional
    sriov_tor_info: {NCS-5500-1: TenGigE0/0/0/6, NCS-5500-2: TenGigE0/0/0/6} or
    sriov_tor_info: {NCS-5500-1: 'TenGigE0/0/0/6, TenGigE0/0/0/7', NCS-5500-2: 'TenGigE0/0/0/6,TenGigE0/0/0/7'}


  a27-fretta-micro-2:
    cimc_info: {cimc_ip: 172.28.121.174}
    dp_tor_info: {NCS-5500-1: TenGigE0/0/0/3, NCS-5500-2: TenGigE0/0/0/3, po: 3}
    hardware_info: {VIC_slot: MLOM}
    rack_info: {rack_id: RackB}
    tor_info: {NCS-5500-1: TenGigE0/0/0/2, NCS-5500-2: TenGigE0/0/0/2, po: 4}
```

```
 a27-fretta-micro-3:
   cimc_info: {cimc_ip: 172.28.121.175}
   dp_tor_info: {NCS-5500-1:TenGigE0/0/0/5, NCS-5500-2: TenGigE0/0/0/5, po: 5}
   hardware_info: {VIC_slot: MLOM}
   rack_info:{rack_id: RackC} # optional
   sriov_tor_info: {NCS-5500-1: 'TenGigE0/0/0/8, TenGigE0/0/0/9', NCS-5500-2: 'TenGigE0/0/0/8, TenGigE0/0/0/9'}

#Note: if sriov is defined, it need not be present on all servers; However, if present on a given server, the
number of SRIOV port must be four and consistent across servers. Also, set the INTEL_SRIOV_PHYS_PORTS to four,
when using SRIOV with NCS-5500 as ToR. Set the value of INTEL_SRIOV_VFS as per the settings of your VNF (see
details later for
the default values)

   tor_info: {NCS-5500-1: TenGigE0/0/0/4, NCS-5500-2: TenGigE0/0/0/4, po: 6}

MULTI_SEGMENT_ROUTING_INFO:
  bgp_as_num: <1 to 65535>
  isis_area_tag: <string>
  loopback_name: <loopback<0-2147483647>>
  api_bundle_id: <1 to 65535>
  api_bridge_domain: <string> #Optional, only needed when br_api of mgmt node is also going via NCS-5500; #this
item and api_bundle_ids are mutually exclusive

  ext_bridge_domain: <string> # user pre-provisions physical, bundle interface, sub-interface and external BD
for external uplink and provides external BD info in setup_data
```

Auto-ToR of Cisco NCS 5500 for LACP Management

From Cisco VIM 3.4.6, an option is available to provide an equivalent of no Link Aggregation Control Protocol (LACP) suspend individual feature in Cisco NCS 5500. With this option, Cisco VIM manages Cisco NCS to only pxe boot the servers by shutting and un-shutting the redundant link on one of the ToRs during installation and other pod management operations where pxe boot of server is required. All other required configurations on Cisco NCS like interface, BD, and VLAN are not managed by Cisco VIM. It is expected that the administrator pre-provisions the Cisco NCS ToRs with all required configuration prior to installation and continue to manage the NCS switches independently.

You can have multiple Cisco NCS 5500 pairs managing the ToR, as long as the Layer 2 connectivity between the ToR pairs is maintained.

To avail this feature as a Day 0 option for Cisco NCS 5500, update the setup_data as follows:

```
TORSWITCHINFO:

  CONFIGURE_TORS: False # Mandatory
  MANAGE_LACP: True
  TOR_TYPE: NCS-5500   #Mandatory

  SWITCHDETAILS:

  -
    hostname: <NCS-5500-1>   # hostname of NCS-5500-1
    username: admin
    password: <ssh_password of NCS-5500-1>
    ssh_ip: <ssh_ip_address of NCS-5500-1>
    vpc_peer_keepalive: <ssh IP address of the peer NCS-5500-2>

  -
   hostname: <NCS-5500-2> # hostname of NCS-5500-2
   username: admin
   password: <ssh_password of NCS-5500-2>
   ssh_ip: <ssh_ip_address of NCS-5500-2>
   vpc_peer_keepalive: <ssh IP address of the peer NCS-5500-1>

SERVER SECTION FOR C SERIES:

SERVERS:
  a27-fretta-micro-1:
    cimc_info: {cimc_ip: 172.28.121.172}
    dp_tor_info: {NCS-5500-1: TenGigE0/0/0/1, NCS-5500-2: TenGigE0/0/0/1, po: 1}                          #
Optional
    hardware_info: {VIC_slot: MLOM}
    rack_info:{rack_id: RackA}
    tor_info: {NCS-5500-1: TenGigE0/0/0/0, NCS-5500-2: TenGigE0/0/0/0, po: 2} # Optional
    sriov_tor_info: {NCS-5500-1: TenGigE0/0/0/6, NCS-5500-2: TenGigE0/0/0/6} or
    sriov_tor_info: {NCS-5500-1: 'TenGigE0/0/0/6, TenGigE0/0/0/7', NCS-5500-2: 'TenGigE0/0/0/6,TenGigE0/0/0
/7'}        # Optional


  a27-fretta-micro-2:
    cimc_info: {cimc_ip: 172.28.121.174}
    dp_tor_info: {NCS-5500-1: TenGigE0/0/0/3, NCS-5500-2: TenGigE0/0/0/3, po:
3}                                # Optional
    hardware_info: {VIC_slot: MLOM}
    rack_info: {rack_id: RackB}
    tor_info: {NCS-5500-1: TenGigE0/0/0/2, NCS-5500-2: TenGigE0/0/0/2, po: 4}

 a27-fretta-micro-3:
   cimc_info: {cimc_ip: 172.28.121.175}
   dp_tor_info: {NCS-5500-1:TenGigE0/0/0/5, NCS-5500-2: TenGigE0/0/0/5, po:
5}                                       # Optional
   hardware_info: {VIC_slot: MLOM}
   rack_info:{rack_id: RackC} # optional
   sriov_tor_info: {NCS-5500-1: 'TenGigE0/0/0/8, TenGigE0/0/0/9', NCS-5500-2: 'TenGigE0/0/0/8, TenGigE0/0/0/9'}

#Note: if sriov is defined, it need not be present on all servers; However, when present on a given server, the
number of SRIOV port need to be four and consistent across the servers. Also, set the INTEL_SRIOV_PHYS_PORTS to
four, when using SRIOV with NCS-5500 as ToR. Set the value of INTEL_SRIOV_VFS as per the settings of your VNF
(see details
later for the default values)

   tor_info: {NCS-5500-1: TenGigE0/0/0/4, NCS-5500-2: TenGigE0/0/0/4, po: 6}
```


The base configuration (server and NCS level) listed in the *setup_data* is required. However, you can add additional configuration in the Cisco NCS to customize the configuration.

Customization of Cisco NCS 5500 Configuration for Ethernet Segment ID and Route-Target

Cisco VIM automatically generates the Ethernet Segment Identifier (ESI) for EVPN segments (as defined under each Bundle-Ether interface) and route-targets during Cisco NCS 5500 ToR configuration.
You can set the ESI for EVPN segments only during Day 0 configuration. To customize the configuration, define the following in the setup_data as part of the Day 0 configuration:

```
ESI_PREFIX: 91.<Pod_number>.<pod_region_number>.00.00.00.00
```

Sample ESI

```
evpn
 interface Bundle-Ether<BE#> ethernet-segment
 ethernet-segment identifier type 0 91.<Pod_number>.<pod_region_number>.00.00.00.00.00.00.<BE#_in_hex>

Example:
evpn
 interface Bundle-Ether10
 ethernet-segment
 ethernet-segment identifier type 0 91.05.02.00.00.00.00.00.0a
```

If ESI defined in RFC 7432 is appended with the Bundle ID in hex, it will add up to a total of 9 octects, that is, the ESI_PREFIX must have a max length of 7 octects.
Similar to ESI_PREFIX, Cisco VIM supports custom-defined route-targets for management, storage, and tenant network segment, when Cisco NCS 5500 is set as ToR switch. This configuration is optional on per network segment basis, but Cisco VIM generates route-target automatically if not defined. To avail this configuration, the pod administrator must define a *rt_suffix* and *rt_prefix* in each network segment as listed below:

```
NETWORKING:
  networks:
    - gateway: 5.0.0.1
      pool: [5.0.0.11 to 5.0.0.50]
      segments: [management, provision]
      subnet: 5.0.0.0/24
      vlan_id: 200
      rt_prefix: <Local to POD>
      rt_suffix: < Region>:< pod_region_number >
   -
     gateway: 172.25.34.161
     segments: [storage]
     subnet: 172.25.34.160/28
     vlan_id: 2438
     rt_prefix: <Local to POD>
     rt_suffix: < Region>:< pod_region_number >
```

Resultant Route-Target

```
<Local to POD>:<Region>< POD number in the region><vlan_id>
```

Example:

```
3000:10100214
```

Each route-target is unique with its respective VLAN-id. Route targets associated with tenant VLANs are generated by appending each VLAN id from TENANT_VLAN_RANGES to the *rt_suffix* and *rt_prefix* defined in the network segments.

Resulting route-targets (*rt_prefix*", plus :, plus *rt_suffix*, plus the VLAN ID) must not exceed the six octets as per RFC 4360 for the Extended Communities. The maximum value is eight octets with the first two being reserved for type information.

NCS Day 0 Configuration (Prior to Cisco VIM installation)

You must define the following configuration on the NCS before commencing the Cisco VIM installation:

```
line default
exec-timeout 0 0

SSH:
ssh server v2
ssh server vrf default
ssh server netconf port 831
ssh server netconf vrf default ssh timeout 60
ssh
server rate-limit 600

USERNAME:

username admin
group root-lr
group cisco-support
secret 0 <password>
```

⚠ For SSH to work, generate a key using *crypto key generate rsa*.

Prerequisites for Segment Routing Global Block and ISIS Prefix

An admin must be predefine the segment routing configuration as given in the following example:

```
segment-routing
  global-block 16000 20000

The prefix within the ISIS setup_data.yaml configuration has to be within the global-block IDs.
Example:

TORSWITCHINFO:
  CONFIGURE_TORS: true

  SWITCHDETAILS:
    -  {br_mgmt_po_info:1, br_mgmt_port_info: TenGigE0/0/0/10, hostname: a25-ncs5500-1-ru30,
isis_loopback_addr: 10.10.10.10,
    isis_net_entity_title: 49.0001.1720.1625.5011.00, isis_prefix_sid: 16001, password:
    CTO1234!, ssh_ip: 172.28.123.176, username: admin, vpc_peer_keepalive:
    172.28.123.177, vpc_peer_port_address: '100.100.100.2/29,100.100.101.2/29',vpc_peer_port_info:'HundredGigE0
/0/1/4,HundredGigE0/0/1/5'}

   -  {br_mgmt_po_info: 1, br_mgmt_port_info: TenGigE0/0/0/10, hostname: a25-ncs5500-2-ru29,
isis_loopback_addr: 20.20.20.20,
   isis_net_entity_title: 49.0001.1720.1625.4022.00, isis_prefix_sid: 16002, password: CTO1234!, ssh_ip:
172.28.123.177, username: admin,
   vpc_peer_keepalive: 172.28.123.176, vpc_peer_port_address: '100.100.100.3/29,100.100.101.3/29',
vpc_peer_port_info:
'   HundredGigE0/0/1/2,HundredGigE0/0/1/3'}

 TOR_TYPE: NCS-5500
```

Prerequisites for API and External Network Segments with Cisco NCS 5500 as ToR

Pre-provision the NCS-5500 with the bridge domains for API and external network segments. The configured bridge domain names for API and external must be the same as those defined in *setup_data.yaml* (api_bridge_domain and ext_bridge_domain) under the MULTI_SEGMENT_ROUTING_INFO section defined above.
A check on each NCS-5500 must show the following:

```
RP/0/RP0/CPU0:NCS-5500-2#sh run l2vpn bridge group cvim
l2vpn
 bridge group cvim
          bridge-domain api
 l2vpn bridge group cvim
        bridge-domain external
```

During deployment of Cisco NCS 5500 as ToR, Cisco supports the workloads of the provider network along with the tenant network.

Listed below are some of the assumptions under which this combination works.

- Provider network segment must be available from Day 0.
- Few of the PROVIDER_VLAN_RANGES must be defined. You can always expand the PROVIDER_VLAN_RANGES with an additional VLAN range (minimum starting VLAN range is 2) so that the maximum number of PROVIDER_VLAN_RANGES and TENANT_VLAN_RANGES should add up to 200.
- Bridge domain for provider starts with prefix: provider VLANId. It is created manually on the Cisco NCS 5500 before the Cisco VIM deployment begins and upstream interfaces are configured.

Support and Pre-requisites for Provider Network with NCS

In a deployment of Cisco NCS 5500 as ToR, along with the tenant network, Cisco supports provider networks. The following points are key to use provider_networks with NCS as ToR:

- Provider network segment has to be defined on Day 0. Also, a handful of PROVIDER_VLAN_RANGES must be defined in the *setup_data.yaml.*

- The PROVIDER_VLAN_RANGES can be extended after the Cisco VIM installation, by running reconfiguration with an updated *setup_data.yaml* (minimum starting VLAN range is two, for example, PROVIDER_VLAN_RANGES: 3200:3202 (existing range), 3204:3206 (newly added range)).
- The maximum number of PROVIDER_VLAN_RANGES and TENANT_VLAN_RANGES must not exceed 600.
- Bridge domain for provider starts with prefix: provider<VLANId> and are created manually on the Cisco NCS 5500 before the Cisco VIM deployment begins with necessary upstream interfaces configured.

Prerequisites for Provider Network with NCS 5500 as ToR

Provider network support requires the following prerequisites:

1. Define the network and provider VLAN ranges sections in *setup_data.yaml* as given below:

   ```
   NETWORKING:
   -
     segments: [provider] vlan_id: None
     PROVIDER_VLAN_RANGES: 127,3406:3409
   ```

2. Pre-provision the Cisco NCS with bridge-domains for corresponding VLANs and configure the bridge-domain uplinks:

   ```
   RP/0/RP0/CPU0:NCS-5500-2#sh run l2vpn bridge group cvim
   l2vpn
     bridge group cvim
      bridge-domain provider127

   l2vpn
    bridge group cvim
     bridge-domain provider3406

   l2vpn
    bridge group cvim
     bridge-domain provider3407
   ```

- Cisco VIM configures all the host facing subinterfaces for these provider VLANs and EVIs, and plumbs them into each of the pre-provisioned provider bridge-domains.
- When pre-provisioning bridge-domain, ensure that the BD names follow the naming convention of provider<*vlan-id*>.

Cisco VIM with NCS-5000 as ToR in "ships in the night" model

From Cisco VIM 3.4.6, you can optionally use the NCS-5500 as a router along with the ToR functionality. In this case, the administrator must pre-provision the Cisco NCS ToRs with all required configuration (like interface, BD, and VLAN configuration) prior to Cisco VIM installation and continue to manage the NCS switches independently. This option provides the flexibility of having multiple Cisco NCS 5500 pairs managing the pod, as long as the Layer 2 connectivity between the ToR pairs is maintained.

To avail this feature as a Day 0 option for Cisco NCS 5500, update the *setup_data* as follows:

```
TORSWITCHINFO:
  CONFIGURE_TORS: False # Mandatory
  TOR_TYPE: NCS-5500    # Mandatory
```

To ensure that the servers can pxe boot, you must identify each of the ToR pairs as primary and secondary. For each of the NCS ports belonging to the primary ToR which are participating in PXE boot, configure the correspnding bundle-ether interface with *bundle lacp-fallback timeout 1*. Do not configure the secondary ToR with this configuration (see below for example). This assymetric configuration is sufficient for PXE to work under the following conditions:

- If both ToRs are up, PXE boot of servers will work
- If secondary ToR is down, PXE boot of servers will work
- If primary ToR is down, PXE boot of servers will not work

A failed ToR indicates a pod that is in a severely degraded state. In case of failed ToR, ensure that you fix/replace the impacted ToR before any lifecycle management action is taken. The corresponding configuration in each of the NCS ports participating in PXE boot is listed below:

```
Primary TOR
PXE boot interface for Server

#sh run interface Bundle-ether 10
Thu May 21 22:58:56.786 UTC
interface Bundle-Ether10
 mtu 9216
 lacp mode active
 lacp system mac 0010.0010.0010
 lacp switchover suppress-flaps 300
 bundle lacp-fallback timeout 1        ----> Configration needed on Primary ToR for PXE with NCS as "ships in
the night"
 load-interval 30
!

#sh run interface TenGigE 0/0/0/4
Thu May 21 22:59:07.075 UTC
interface TenGigE0/0/0/4
 description CONTROL interface for server-1
 bundle id 10 mode inherit
 load-interval 30
!


Secondary TOR
PXE boot interface for Server

#sh run interface Bundle-ether 10
Thu May 21 22:40:04.587 UTC
interface Bundle-Ether10
 mtu 9216
 lacp mode active
 lacp system mac 0010.0010.0010
 lacp switchover suppress-flaps 300
 load-interval 30
!

#sh run interface TenGigE 0/0/0/4
Thu May 21 22:40:13.149 UTC
interface TenGigE0/0/0/4
 description CONTROL interface for server-1
 bundle id 10 mode inherit
 load-interval 30
!
```

ToR Configuration with ACI Fabric

Cisco VIM integrates with ACI in three different ways:

1. *ships in the night* where the switch ports are pre-configured with relevant VLANs ahead of time.
2. ToR configuration with and without the APIC plugin using ACI API.

In all these cases, openvswitch is used as the mechanism driver.

In the case of *ships in the night* model, all the relevant configurations are done static and done ahead of time.

If ACI plugin is not used, Cisco VIM invokes the APIC APIs to pre-provision the right set of VLANs (along with the Day 0 aspects) on the corresponding server ports ahead of time.

For details on integration with and without the APIC plugin and the relevant ToR configuration, see Enabling Auto-ToR via ACI API.

Servers and Network Option

Servers and Network Option

- Overview
- SRIOV Support on Cisco VIC Pod
- Support of 25G VIC and NIC
    - Support of Pod Expansion with 4 x 25G XXV710 Computes
    - Support of Pod Expansion with 4 x 25G XXV710 Computes with evenly distributed NUMA Mapping.
- Support of M4/M5 BOM in a Pod
- Remote Registry Credentials
- Common CIMC Access Information for C-series Pod
- Configure Cobbler
- Configure Network
- Define Network Segments
- Define Server Roles
- Define Servers - Rack (C-Series, Quanta) Pod
- Server Common
- Fan Policy for Cisco UCS

Overview

Cisco VIM supports C-series pod running with either all Intel X710 NICs or Cisco VICs for control and data plane. In the Intel NIC setup, M4 and M5 (Micropod) based pods must have 2-4 port and 1 or 2-4 port X710 respectively, for control and data plane connectivity. For BOM details and its variations, reach out to *Cisco VIM Product Management* team. The orchestrator identifies the NIC support, based on the following INTEL_NIC_SUPPORT values:

- False -This is the default value. The orchestrator assumes that all the servers have Cisco VIC.
- True - The orchestrator assumes that all the servers have Intel NIC.

To define the value, set the following configuration in the *setup_data.yaml* file:

```
# INTEL_NIC_SUPPORT: <True or False>
```

The X710 based NIC redundancy is enabled by default for M4-based Intel NIC system, but not for M5-based Intel NIC system. For more information, see UCS C-Series Network Topologies.

To bring in NIC redundancy across the X710s for M5-based Intel NIC systems, define the following global parameters in the *setup_data.yaml* file:

```
NIC_LEVEL_REDUNDANCY: <True or False> # optional and only applies when INTEL_NIC_SUPPORT is set to True
```

A C-series pod with Intel NIC also supports SRIOV as an option, when defined in a setup_data.
To enable SRIOV as an option, define a value in the range of 1 to 32 where 32 is the maximum number of INTEL_SRIOV_VFS: <integer>.

In the C-series pod running with 4-port Intel 710 card, 1 port (port #c) from each of the Intel NICs is used for SRIOV by default. However, some VNFs need additional SRIOV ports to function. To meet the requirement, an additional variable is introduced in the *setup_data.yaml* file to include a second port (port d) of the Intel NIC for SRIOV.
To adjust the number of SRIOV ports, set the following option in the *setup_data.yaml* file:

```
#INTEL_SRIOV_PHYS_PORTS: <2 or 4>
```

The parameter INTEL_SRIOV_PHYS_PORTS is optional and takes the value 2, 4, or 8 (only in M5 based pods). If not defined, value of 2 is used. For SRIOV support on NCS-5500, the value must be 4 for INTEL_SRIOV_PHYS_PORTS.
In M5 Micropod environment with NIC_LEVEL_REDUNDANCY as false, a single X710 is used for control/data plane and an additional XL710 or 2-port X710 is used for SRIOV, where INTEL_SRIOV_PHYS_PORTS takes a value of 2.

From Cisco VIM 3.4.4 with pure Intel NIC pod, you can combine control plane and date plane on the same pair of ports (use total of 2 ports) or individually separated one pair of ports for control plane and another pair of ports for data plane (use total of 4 ports).

- True - control plane and date plane on the same pair of ports (Default for third-party platform)

    samxpet
      |-samxpet0
      |-samxpet1

- False - individually separated one pair of ports for control plane and another pair of ports for data plane (Default for UCS C-series platform)

```
samx
  |-samx0
  |-samx1
pet
  |-pet0
  |-pet1
```

```
# COMBINE_CPDP: <True or False>   # optional and only applies when
#                                 # INTEL_NIC_SUPPORT is set to True
```

SRIOV Support on Cisco VIC Pod

Cisco VIM supports M4-based C-series pod running with one two-port Cisco VIC for control plane and two two-port Intel 520s or two two-port XL710 for SRIOV (called VIC/NIC deployment). It also supports M5-based C-series pod running with one 2-port 40G Cisco VIC for control plane and two 2-port XL710 for SRIOV.
The orchestrator identifies the VIC/NIC support based on the following CISCO_VIC_INTEL_SRIOV values:

- False - This is the default value. The orchestrator assumes that all the servers have Cisco VIC.
- True - The orchestrator assumes that all the servers have Intel NIC.

To define the value, use the command:

```
CISCO_VIC_INTEL_SRIOV: <True or False>
```

A C-series M4 pod running Cisco VIC/Intel NIC (2x520 or 2xXL710) supports SRIOV on the Intel NIC.

A C-series M5 pod running Cisco VIC/Intel NIC (2xXXV710 or 2xXL710) supports SRIOV on the Intel NIC. In the case of 25G and 40G BOM, the Cisco VIC is 1457 or 1387, respectively.

To enable,SRIOV define the below parameter with a value in the range 1-63 (63 is maximum) for X520 or 1-32 (32 is maximum) for XL710 or XXV710: INTEL_SRIOV_VFS: <integer>.

By default in the M4 C-series pods running with Cisco VIC and Intel 520/XL710, the control plane runs on the Cisco VIC ports and all the four ports in the Intel NIC are used for SRIOV.

In C-Series M5 pods running with 40G Cisco VIC and Intel XL710, the control plane runs on the Cisco VIC ports and all the four or eight ports from the two intel XL710 are used for SRIOV. Similarly in the case of 25G M5 pod (25G Cisco VIC and Intel XXV710), the same layout applies.

In M5-based VIC/NIC pods, define INTEL_SRIOV_PHYS_PORTS: <4 or 8>, with default value as 4, to indicate the number of ports participating in SRIOV.

In the pods running with CISCO_VIC_INTEL_SRIOV option, some computes can run only with Cisco VIC without SRIOV option if they do not have Intel NIC cards.
Define the following parameter in the *setup_data.yaml* file to setup the SRIOV card type:

```
#SRIOV_CARD_TYPE: <X520 or XL710>   # for M4 based computes
#SRIOV_CARD_TYPE: <XXV710 or XL710> # for M5 based computes
```

M4 compute supports different types of the card (X520 and XL710), with a pod that supports the computes having different SRIOV card types.

1. If SRIOV_CARD_TYPE is not provided, Cisco VIM chooses the first two slots from all SRIOV compute nodes.
2. If SRIOV_CARD_TYPE is provided, Cisco VIM chooses the first two slots matching the target card type from each of the SRIOV compute nodes, so that a match between intent and reality exists.

For Quanta-based pods, the SRIOV slot order starts from the higher slot number, that is, for NUMA, NIC at higher slot has value 0, 2. You can override this, by defining the following as ascending, in which case NIC at higher slot has value of 1, 3.

```
# SRIOV_SLOT_ORDER: <ascending or descending> # Optional, applicable for Quanta-based pods
```


From Cisco VIM 2.4.4 onwards, some computes have XL710 while others have X520 for SRIOV in M4 settings. This is achieved by defining the SRIOV_CARD_TYPE at a per compute level (see the SERVERS section of the setup_data in example file).

From Cisco VIM 2.4.9 onwards, 40G based M5 computes are supported.

From Cisco VIM 2.4.15, 40G based M5 controller and Ceph nodes can be mixed with 10G based M4 VIC/NIC pods.

Support of 25G VIC and NIC

From Cisco VIM 3.4.0, Cisco VIC 1457 and Intel XXV710 are supported in some specific BOM configuration. The first one is a combination where the control plane is running on two ports (A and C by default) of Cisco 1457 and data plane is running over VPP on the two ports of the Intel XXV710. In this configuration, there is a second XXV710 NIC for SRIOV. To realize this configuration of Cisco VIC and Intel NIC baremetal combination without creating any Cisco vNICs, the option of INTEL_NIC_SUPPORT must be set to true.

```
CISCO_VIC_SUPPORT: true
INTEL_NIC_SUPPORT: true
INTEL_SRIOV_PHYS_PORTS: 2
INTEL_SRIOV_VFS: 16
```

The above option can also be used with openvswitch as the mechanism driver. If the above option is chosen for Cisco 1457 VIC, by default port A and C of the VIC are used for the control plane. To use the port A and B of the Cisco VIC for samx, you can define an optional variable globally or at a per-server level.

Following is the snippet of how to define the configuration in the *setup_data.yaml* file:

```
SERVER_COMMON:
  # Optional global configuration to change VIC's port channel enable configuration
  # option, from default True to False.  Applicable only for Cisco VIC that support Port Channel like UCS VIC
1457 25Gbps network adapter.
  # VIC_port_channel_enable: <True or False>
  # This can also be specified or overridden at per server level under server's hardware_info section.
```

A global configuration option is available to change VIC's admin FEC mode from default *Auto* to either *Off, cl74,* or *cl91'*mode and to adapt to different types of switches. This is applicable only for Cisco VIC that supports changing the admin FEC mode like UCS VIC 1457 25Gbps network adapter.

The following is the snippet in the *setup_data.yaml* file to realize this configuration for the pod.

```
SERVER_COMMON:
  ....
  #VIC_admin_fec_mode: <Auto, Off, cl74, or cl91>
# This can be specified or overridden at per server level under server's hardware_info section.
```

Cisco M4 (VIC 1227) based pods support additional M5 computes running on Cisco 1457 VIC. Additionally, OVS based Cisco M5 VIC (1457) with XXV710 NIC pods is supported. In this combination, the control and data plane run on Cisco 1457 VIC, with four ports of XXV710 Intel NIC dedicated for SRIOV.

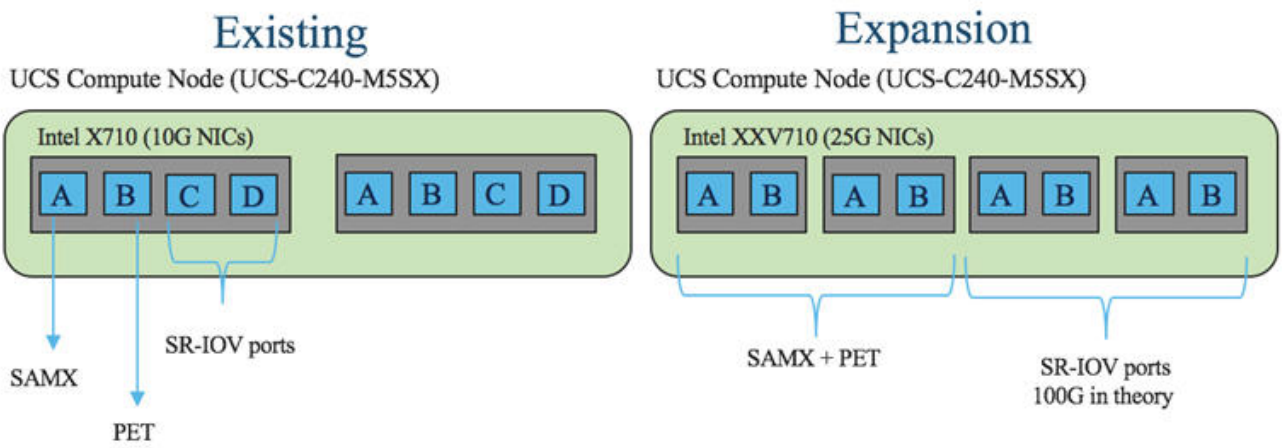
Support of Pod Expansion with 4 x 25G XXV710 Computes

For 10G X710 based Intel pods, typically all computes have two X710-DA4 NICs. From Cisco VM 4.2.0, the expansion of pod with computes having four XXV710-DA2/XL710-DA2 NICs is supported.

Following is the port configuration matrix supported:

```
  # Two X710-DA4 (4 ports card)
  #   control plane: 1-A and 2-A
  #   data plane:    1-B and 2-B
  #   sriov0/2:      1-C and 1-D
  #   sriov1/3:      2-C and 2-D

  # Four XL710-DA2 (2 ports cards)
  #   control plane: 1-A and 2-A
  #   data plane:    1-B and 2-B
  #   sriov0/2:      3-A and 3-B
  #   sriov1/3:      4-A and 4-B
```



For computes with 25G NICs, the lower two NIC slots are used for control and data plane. Hence, care must be taken in placing the NICs across the NUMA node. In an UCS 240 M5 node, use slot 1 for the first 25G NIC, and slot 4, 5, and 6 for next 3 25G NIC cards to yield a balanced virtual network across NUMA nodes. However, SRIOV is dedicated to a single NUMA.

Support of Pod Expansion with 4 x 25G XXV710 Computes with evenly distributed NUMA Mapping.

Cisco VIM support for expansion of POD with computes having four(4) XXV710-DA2 NICs (25Gig NICs) on Cisco UCS C240 M5SX servers with custom NUMA mapping configuration. This features allows to evenly distribute CPDP and SRIOV ports across NUMA on a given compute.

To support this feature, new configuration parameters are introduced at per compute level. Using this parameter user has the flexibility to select PCIe slots for CPDP and SRIOV ports. However care must be taken when assigning PCIe slot numbers and provide PCIe slot numbers which are mapped across NUMA. User has to consult hardware specification for servers/server vendor to get accurate information on this. This feature was validated and supported on Cisco UCS M5 Servers.

Example:

To support configuration displayed in ***Figure-1*** user has to update "setup_data.yaml" with below configuration in the compute node section. Here we are asking installer to pick

PCIe 1 and PCIe 4 for CPDP ports and PCIe 3 and PCIe 6 for SRIOV ports.

```
hardware_info: {NIC_LEVEL_REDUNDANCY: true, cpdp_nic_slot: [1,4], sriov_nic_slot: [3,6]}
```

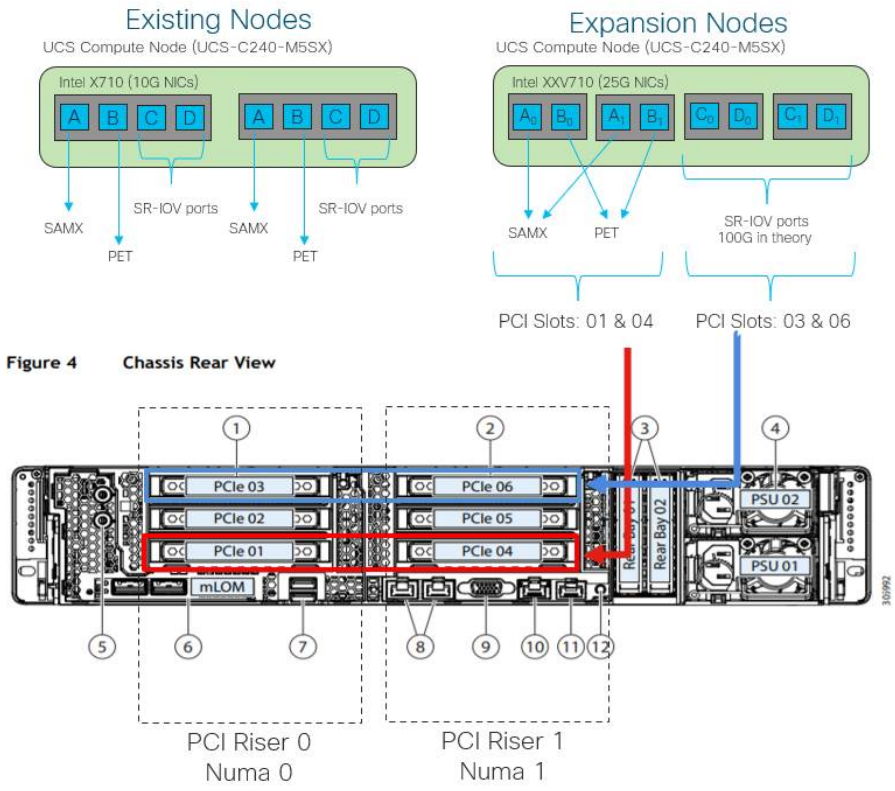


Figure-1

If user want to pick different configuration shown in ***Figure-2*** they have to update "setup_data.yaml" as follow:

```
hardware_info: {NIC_LEVEL_REDUNDANCY: true, cpdp_nic_slot: [1,4], sriov_nic_slot: [2,5]}
```

Here we are asking installer to pick PCIe 1 and PCIe 4 for CPDP ports and PCIe 2 and PCIe 5 for SRIOV ports.

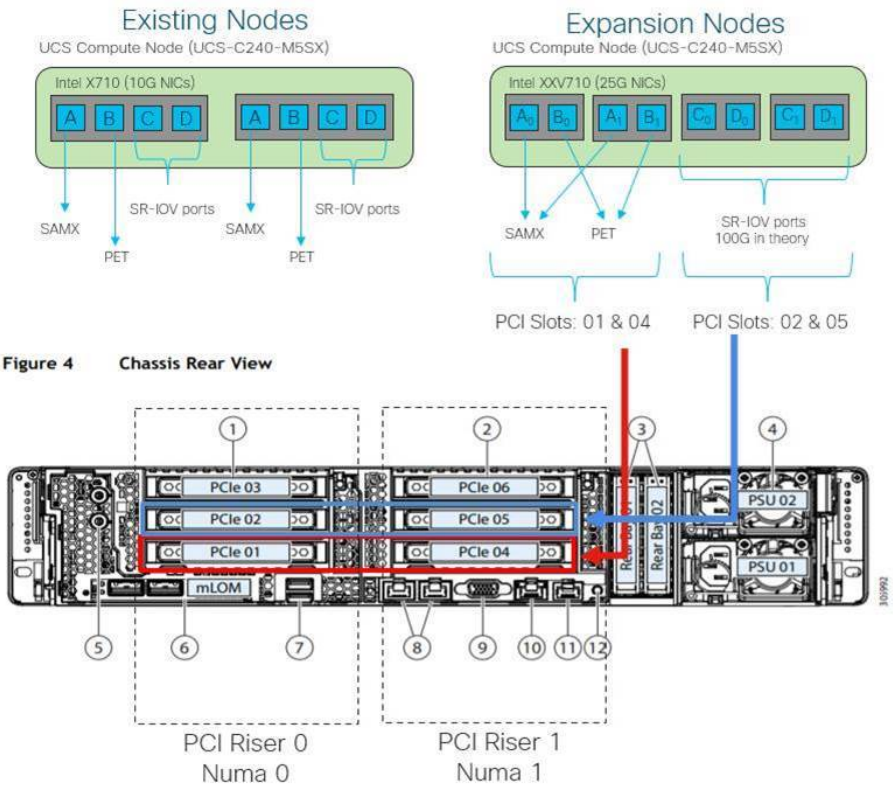


Figure-2

Both configurations displayed in ***Figure-1*** and ***Figure-2*** on UCS C240 M5SX server are supported Configurations.

NOTE: Riser-1 card in the PCI-slots may alter the NUMA mapping. It is always better to refer to the UCS Manual to check how the PCI slots are mapped to the NUMA node and then make the decision to plug in the NIC cards for configuring with CVIM.

Support of Third-party Compute in Hybrid Mode

Cisco VIM 2.4 introduces the first third-party compute. The first SKU chosen is HPE ProLiant DL360 Gen9. With this support, the Cisco VIM software is flexible enough to accommodate other SKUs. In Cisco VIM 2.4, the supported deployment is a full-on pod, with OVS as the mechanism driver, where the management, control, and storage nodes are based on existing Cisco UCS c220/240M4 BOM, and the compute nodes are on HPE ProLiant DL360 Gen9 hardware. From Cisco VIM 2.4.5 onwards, Cisco VIM supports the same HP SKU with both HP and HPE brand.

To minimize changes to the existing orchestration workflow and Insight UI, the existing Cisco VIC+NIC combo deployment scenario is reused to handle HP as a third-party compute. This reduces the changes needed for the hardware topology and *setup_data.yaml* configuration file. For details on NIC settings to enable HPE ProLiant DL360 Gen9 third-party compute, see **Intel NIC Support for SRIOV**

For Quanta servers, the support of third-party is available for all nodes (servers in control, compute, storage, and management role).

Cisco VIM 3.4.6 supports Dell PowerEdge R740 series as compute nodes with Cisco UCS M5 nodes acting as management, control, and storage nodes in a full pod system.

The following table shows the port type mapping between Cisco UCS C-series, HPE ProLiant DL360, Quanta, and Dell as compute:

Port Type	Cisco UCS c220/c240 Compute	HPEProLiantDL360Gen9 as Compute	Quanta Server as Full Pod	Dell PowerEdge 740 as Compute
Control and Data Plane	M4: MLOM - 1 x VIC 1227 M5: MLOM - 1 x VIC 1387 M5 MLOM - 1 x VIC 1457 M5: Intel - 1 or 2 x X710 DA4	FlexLOM - HP Ethernet 10Gb 2-port 560FLR-SFP+ Adapter	OCP 25G 2 port xxv710 based card	PCIe - 2xIntel xxv710 DA2 25 Gbps 2 port NIC
SRIOV	M4: PCIe - Intel X520-DA2 10 Gbps or Intel XL710 DA2 40 Gbps 2 port NIC M5: PCIe - Intel XL710 DA2 40 Gbps 2 port NIC or Intel XXV710 DA2 25 Gbps 2 port NIC	PCIe - HP Ethernet 10Gb 2-port 560SFP+ Adapter	PCIe - 2xIntel xxv710 DA2 25 Gbps 2 port NIC	PCIe - 2xIntel xxv710 DA2 25 Gbps 2 port NIC

As this deployment does not support auto-ToR configuration, the ToR switch must have trunk configuration with native VLAN, jumbo MTU, and no LACP suspend-individual on the control and data plane switch ports.
Sample Nexus 9000 port-channel configuration is as follows:

```
interface port-channel30

description compute-server-hp-1 control and data plane
switchport mode trunk

switchport trunk native vlan 201
spanning-tree port type edge trunk
spanning-tree bpdufilter enable
mtu 9216

no lacp suspend-individual
vpc 30

!
interface Ethernet1/30

description compute-server-hp-1 flexlom port 1
switchport mode trunk

switchport trunk native vlan 201
mtu 9216

channel-group 30 mode active
```

Once the physical connection to the ToR switches and switch ports configuration is completed, enable/add the following additional variables in the *setup_data.yaml* configuration file:

```
CISCO_VIC_INTEL_SRIOV: True
INTEL_SRIOV_VFS: 63
```

Support of M4/M5 BOM in a Pod

With the transition of UCS from M4 to M5, there is a need to support M5 hardware in an existing M4 pods, so that the lifecycle management of the nodes through the EOL/EOS process of M4 can also be supported. Assuming that the additional M5 nodes have the same number of CPU cores, memory, and disk size (generation of the components can vary), the following table lists the NIC level compatibility matrix between M4 and M5 node.

Scenarios	Compute Type	1 x Cisco VIC 1227 (10G)	1 x Cisco VIC 1387 (40G)	1 x Cisco VIC 1457 (25G)	2 x Intel NIC X520 (10G)	2 x Intel NIC XL710 (40G)	2 x Intel NIC XXV710 (25G)	1 x Intel NIC X710-DA2 (10G)	2 x Intel NIC X710-DA4 (10G)	4 x Intel NIC XXV710 (25G)
Scenario 1	Existing M4	✓			✓					
	New M5		✓			✓				
Scenario 2	Existing M4	✓			✓					
	New M5			✓			✓			
Scenario 3*	Existing M4	✓						✓		
	New M5			✓				✓		
Scenario 4	Existing M4								✓	
	New M5								✓	
Scenario 5	Existing M4				✓					
	New M5									✓

- Scenario 3 is applicable only to NGENA BOM
- The M5 BOM for control and ceph nodes is also supported as mentioned in the table above, but you must drop the SRIOV and pet interface aspects of the NIC card from the target node.

Remote Registry Credentials

```
REGISTRY_USERNAME: '<username>'
REGISTRY_PASSWORD: '<password>'
REGISTRY_EMAIL: '<email@address.com>'
REGISTRY_NAME: <hostname of Cisco VIM software Hub'> # optional only if Cisco VIM software Hub is used
```

Common CIMC Access Information for C-series Pod

```
CIMC-COMMON:
cimc_username: "admin"
cimc_password: <"password">
```

Configure Cobbler

```
## Cobbler specific information.
## kickstart:    static values as listed below
## cobbler_username: cobbler #username to access cobbler server; static value of Cobbler; not user configurable
## admin_username: root # static value of root; not user configurable
## admin_ssh_keys: This is a generated key which is put on the hosts.

## This is needed for the next install step, using Ansible.

COBBLER:

pxe_timeout: 45. # Optional parameter (in minutes); min of 30 and max of 120, defaults to 45 mins
cobbler_username: cobbler   #cobbler UI user; currently statically mapped to cobbler; not user configurable
admin_username: root       # cobbler admin user; currently statically mapped to root; not user configurable
#admin_password_hash has be the output from:

# python -c "import crypt; print crypt.crypt('<plaintext password>')"
admin_password_hash: <Generate the admin pwd hash using the step above; verify the output starts with $6>

admin_ssh_keys:  # Optional parameter

- ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAoMrVHLwpDJX8j2DiE55WtJ5NWdiryP5+FjvPEZcjLdtdWaWA7W dP6EBaeskmyyU9B8ZJr1uClIN
/sT6yD3gw6IkQ73Y6bl1kZxu/ZlcUUSNY4RVjSAz52/oLKs6n3wqKnn7rQuLGEZDvXnyLbqMoxHdc4PDFWiGXdlg5DIVGigO9KUncPK
cisco@cisco-server

kickstart:               # not user configurable, optional
control: ucs-b-and-c-series.ks
compute: ucs-b-and-c-series.ks
block_storage: ucs-b-and-c-series.ks
```

Configure Network

```
NETWORKING:

domain_name:domain.example.com

#max of 4 NTP servers
ntp_servers:
-  <1.ntp.example.com>
-  <2.ntp.example2.com >
or
ntp_servers: ['2001:c5c0:1234:5678:1002::1', 15.0.0.254]    <== support for IPv6 address


#max of 3 DNS servers
domain_name_servers:
- <a.b.c.d>
or
domain_name_servers: ['2001:c5c0:1234:5678:1002::5', 15.0.0.1]    <== support for IPv6 address


http_proxy_server: <a.b.c.d:port> # optional, needed if install is through internet, and the pod is behind a
proxy
https_proxy_server: <a.b.c.d:port> # optional, needed if install is through internet, and the pod is behind a
proxy
admin_source_networks: # optional, host based firewall to maintain the admin's source IP (v4 or v6) in allowed
list.

- 10.0.0.0/8

- 172.16.0.0/12

- <"2001:xxxx::/64">
```

External access to the management node is made through the IP address configured on the br_api interface. To provide additional security for this connection, an optional *admin_source_networks* parameter is provided. When specified, the access to administrator services is allowed only from the IP addresses specified on this list.

Use this setting with care, as a misconfiguration can lock out an administrator from accessing the management node through the network. Recovery can be done by logging in through the console and reconfiguring this setting.

Define Network Segments

```
networks:
-
vlan_id: <int>

subnet: <cidr with mask> # true routable network
gateway: <ipv4_address >
ipv6_gateway: <ipv6_address>  <== required if IPv6 based OpenStack public API is enabled
ipv6_subnet: < v6 cidr with mask>

segments:

- api
-
vlan_id: <int>
subnet: <cidr/mask>
gateway: <ipaddress>
pool:
# specify the pool range in form of <start_ip> to <end_ip>, IPs without the "to"
# is treated as an individual IP and is used for configuring

-
ip_address_1 to ip_address_2 in the current network segment

# optional, required if managemen_ipv6 is defined at server level
ipv6_gateway: <ipv6_address>
ipv6_subnet: <v6 cidr with mask>
ipv6_pool: ['ipv6_address_1 to ipv6_address_2']

segments: #management and provisioning are always the same
-  management
-  provision


# OVS-VLAN requires VLAN-id as "None"

# LinuxBridge-VXLAN requires valid VLAN-id

-
vlan_id: <vlan_id or None>
subnet: <v4_cidr w/ mask>
gateway: <v4 ip address>
pool:
- ip_address_1 to ip_address_2 in the current network segment

segments:
- tenant
-
vlan_id: <vlan_id>
subnet: <v4_cidr w/mask>
gateway: <ipv4_addr>

pool:
-
ip_address_1 to ip_address_2 in the current network segment

segments:
- storage
# optional network "external"
-
vlan_id: <int>
segments:
-
external
# optional network "provider"; None for C-series
-
vlan_id: "<None>"
segments:

-
provider
```

For ceph pod, the storage segment must be replaced with segment named *cluster.* Also, for central ceph pod, the only other segment allowed is management/provision.

For ceph pod, if management network is dual stack type, the cluster network must also be a dual-stack network.

Define Server Roles

Under the Roles section, add the host name of the servers and their corresponding roles. In case of Micropod, specify the same server names under control, compute, and ceph. Ensure that the number of servers under each role must be three for Micropod. You can optionally expand the Micropod to include additional computes. In the case of Hyperconverged deployment (HC), all storage nodes act as compute nodes, but not vice-versa.

In case of edge pod, specify the same server names under control role (total of three), and compute role (there is no server with storage role) to support low latency workloads without persistent storage. You can optionally expand the edge pod, to include additional computes. The edge pod can connect to a central Ceph cluster via its management network, so that the Ceph cluster offers glance image service.

The central Ceph cluster to which the edge pods are communicating to for the glance image service is called the Ceph pod-type. For the pod-type *ceph*, specify the same server names under *cephcontrol* (total of three), and *cephosd* role. You can optionally expand the ceph pod to include additional cephosd nodes.

```
ROLES:-> for PODTYPE: fullon
  control:
    Your-Controller-Server-1-HostName
    Your-Controller-Server-2-HostName
    Your-Controller-Server-3-HostName

  compute:
    Your-Compute-Server-1-HostName
    Your-Compute-Server-2-HostName
    Your-Compute-Server-n-HostName

  block_storage:
    Your-Ceph-Server-1-HostName
    Your-Ceph-Server-2-HostName
    Your-Ceph-Server-3-HostName

ROLES:-> for PODTYPE: micro
  control:
    Your-Server-1-HostName
    Your-Server-2-HostName
    Your-Server-3-HostName

  compute:
    Your-Server-1-HostName
    Your-Server-2-HostName
    Your-Server-3-HostName
    Your-Server-4-HostName (optional expansion of computes)
    Your-Server-5-HostName (optional expansion of computes)

  block_storage:
    Your-Server-1-HostName
    Your-Server-2-HostName
    Your-Server-3-HostName

ROLES:-> for PODTYPE: UMHC
  control:
    Your-Controller-Server-1-HostName
    Your-Controller-Server-2-HostName
    Your-Contoller-Server-3-HostName

  compute:
    Your-Compute-Server-1-HostName
    Your-Compute-Server-2-HostName
    Your_HC_Server-1_HostName
    Your_HC_Server-2_HostName
    Your_HC_Server-3_HostName

  block_storage:
    Your_HC_Server-1_HostName
```

```
    Your_HC_Server-2_HostName
    Your_HC_Server-3_HostName

ROLES: -> for PODTYPE: edge
  control:
    Your-Server-1-HostName
    Your-Server-2-HostName
    Your-Server-3-HostName

  compute:
    Your-Server-1-HostName
    Your-Server-2-HostName
    Your-Server-3-HostName
    Your-Server-4-HostName (optional expansion of computes)
    Your-Server-5-HostName (optional expansion of computes)


ROLES: -> for PODTYPE: ceph
  cephcontrol:
     Your-Server-1-HostName
     Your-Server-2-HostName
     Your-Server-3-HostName

  cephosd:
     Your-Server-1-HostName
     Your-Server-2-HostName
     Your-Server-3-HostName
     Your-Server-4-HostName (optional expansion of Ceph OSD Nodes)
     Your-Server-5-HostName (optional expansion of Ceph OSD Nodes)

ROLES:-> for PODTYPE: nano
  control:
    Your-Server-1-HostName

  compute:
    Your-Server-1-HostName
```


- The maximum length of non-FQDN hostname is 32 characters. The maximum length of Your-Controller-Server-1-HostName hostname is 32 characters in both the ROLES and SERVERS section.
- The maximum length including the FQDN is 64 characters, where the hostname can only have characters that are in any combination of "A-Za-z0-9-.", and the TLD is not all numeric.
- Cisco VIM does not allow "_" in the hostnames.
- For enabling IPA, the hostname for all servers including the management node must be in lower case FQDN (RFC4120) and must belong to *ip_domain_name.*

Cisco VIM introduces the notion of Micropod to address solutions that have requirements of high availability, with limited compute and storage needs. In this deployment model, the control, compute, and storage services reside on each of the three nodes that constitute the pod. Cisco VIM also supports the expansion of the Micropod to accommodate additional compute nodes. Each cloud application can decide the type of pod needed based on their resource (memory, storage consumption) requirements. The Micropod option supports only OVS/VLAN (with Cisco-VIC or Intel 710 NIC) or VPP/VLAN (only on Intel NIC) on a specific BOM.

To enable the Micropod option, update the setup_data as follows:

```
PODTYPE: micro
```

Cisco VIM supports the hyper-convergence (UMHC) option of UMHC and NGENAHC. The UMHC option supports only OVS/VLAN with a combination of Cisco-VIC and Intel 520 NIC on a specific BOM, while the NGENAHC option supports only VPP/VLAN with control plane over Cisco VIC and data plane over two-port Intel X-710.

To enable the hyper-convergence with (UMHC) option, update the *setup_data* as follows:

```
PODTYPE: UMHC
```

To enable the hyper-convergence with NGENAHC option, update the setup_data as follows:

```
PODTYPE: NENAHC
```

On Quanta server, you can also enable edge cloud functionality for low-latency workloads, for example, vRAN that does not need persistent storage. To enable such deployment, update the setup_data as follows:

```
PODTYPE: edge
```

From Cisco VIM 3.4.6, a single node non HA cloud with no persistent storage is supported. In this deployment model, the control and compute services reside on a single node that constitute the pod. In case of nano pod, no expansion of the cloud is possible and the management node can be a physical box or one in a VM running off a Cisco VIM cloud, that is Layer 3 adjacent to the nano pod. The nano pod option supports only OVS/VLAN (with Cisco-VIC or Intel 710 NIC) on a specific BOM (for BOM details, reach out to *CVIM Product Management team.*

To enable the nanopod option, update the setup_data as follows:

```
PODTYPE: nano
```

If the edge pod is communicating with a central Ceph cluster that is managed by Cisco VIM, update the *setup_data* for the respective central-ceph cluster as follows:

```
PODTYPE: ceph
```

From Cisco VIM 3.4.6, an option is available to extend the usage of central glance image functionality onto a full/micro/hyper-converged pod for the ceph mon interface on management network. To avail this change, define MON_INTERFACE as *management* as part of Day 0 installation of the pod that is targeted to provide the central glance image functionality. From Cisco VIM 4.0.0, MON_INTERFACE option works for a single stack pod only.

```
MON_INTERFACE: <management or storage> # Optional, default is storage; Value of management is needed as part of
Day 0 installation for the pod
                                       # to provide Central glance image functionality; this option is not
allowed in a dual stack environment.
```

For information on the connectivity of edge and Central ceph pods, see OpenStack Configuration

Define Servers - Rack (C-Series, Quanta) Pod

The maximum length of host name is 32 characters.

```
# For enabling IPA, hostname for all servers including the management node must be in lower case FQDN (RFC4120)
and belongs to ipa_domain_name

SERVERS:

 Your_Controller_Server-1_HostName:
   cimc_info: {'cimc_ip': <IPv4 or IPv6>}
   rack_info: {'rack_id': 'RackA'}

   #hardware_info: {'VIC_slot': '7'} # optional; only needed if vNICs need to be created on a specific slot,
for example, slot 7
   #management_ip: <static_ip from management pool> #optional, if defined for one server, must be defined for
all nodes
   #cimc username, password at a server level is only needed if it is different from the one defined in the
CIMC-COMMON section
   # management_ipv6: <Fixed ipv6 from the management_ipv6 pool> # <== optional, allow manual static IPv6
addressing, also if defined management_ip has to be defined

   #storage_ip: <Fixed ip from the storage pool> # optional. If defined for one server or if management_ip has
to be defined, then it must be defined for all.
   #DISABLE_HYPERTHREADING: <True or False> # <== optional, override the global Hyper-Threading configuration


 Your_Controller_Server-2_HostName:
  cimc_info: {'cimc_ip': '<v4 or v6>', 'cimc_username': 'admin','cimc_password': 'abc123'}
```

```
  rack_info: {'rack_id': 'RackB'}

Your_Controller_Server-3_HostName:

  cimc_info: {'cimc_ip': '<v4 or v6>'}
  rack_info: {'rack_id': 'RackC'}
  hardware_info: {'VIC_slot':'7'} # optional only if a specific VNIC needs to be chosen

Your_Compute-1_HostName:
  cimc_info: {'cimc_ip': '<v4 or v6>}
  rack_info: {'rack_id': 'RackA'}
  hardware_info: {'VIC_slot': '3'} # optional only if specific VNIC needs to be chosen
  VM_HUGHPAGE_PERCENTAGE: <0 - 100>  # optional only for compute nodes and when NFV_HOSTS: ALL and
MECHANISM_DRIVER: openvswitch
  VM_HUGHPAGE_SIZE: <2M or 1G> # optional, only for compute nodes and when NFV_HOSTS is ALL and
MECHANISM_DRIVER is openvswitch
  trusted_vf: <True or False> # optional, only for compute nodes which have in SRIOV, can be enabled via
reconfigure
  rx_tx_queue_size: <512 or 1024> # optional, only for compute nodes
  NOVA_CPU_ALLOCATION_RATIO: 1.0 <float, range: 0.958-16.0>   # <== optional, override the
NOVA_CPU_ALLOCATION_RATIO configuration defined in openstack_config.yaml
  NOVA_RAM_ALLOCATION_RATIO: 1.0 <float, range: 1.0-4.0>      # <== optional, override the
NOVA_RAM_ALLOCATION_RATIO configuration defined in openstack_config.yaml

  hardware_info: {'VIC_slot': '<7>', SRIOV_CARD_TYPE: <XL710 or X520 or XXV710>,
  VIC_port_channel_enable: <True or False>, VIC_admin_fec_mode: <Auto, Off, cl74, or cl91>,
  root_drive_type: <HDD or SSD or M.2_SATA, NUM_GPU_CARDS: < 0 to 6>}, VIC_link_training: '<OFF or ON>'}

  # VIC_Slot is optional, defined for location of Cisco VIC,
  # VIC_port_channel_enable is optional and applicable to 1457 based VIC where you can use port A and B to
connect
                  to ToR (instead of port A and C),
  # VIC_admin_fec_mode is optional and used when the ToR needs explicit configuration of fec mode.
                          Unless deviating from the default BOM, the option of root_drive_type can be skipped.
                          However, it is mandatory to define the option M.2 on a per server basis, if it is
used as a booting media for a given server
  # LIBVIRT_WRITETHROUGH_CACHE: <True or False>  # <== optional, Switches the option from writeback to
writethrough. Note that there might be a performance hit.
  # ENABLE_ISOLCPU: <True or False> # <== optional, default is False. If it is enabled, all VNFs must be
running with CPU pinning flavor, otherwise VNFs will hit significant
                                performance degradations

Note that the option of M.2 is valid only for UCS M5 systems.

  NUM_GPU_CARDS is optional and can vary on a per server basis.
  VIC_link_training: < is optional and provided to have better compatibility between Cisco VIC 1387 (40Gbps)
and Cisco ACI

 Your_Storage HostName:
  cimc_info: {'cimc_ip': 'v4 or v6'} rack_info: {'rack_id': 'RackA'}
  hardware_info: {osd_disk_type: <HDD or SSD>} # optional only when the pod is multi-backend ceph, and a
minimum of three
                     storage servers must be available for each backend type. It is recommended to have four
nodes for each storage type.
```

SRIOV_CARD_TYPE option is valid only if CISCO_VIC_INTEL_SRIOV is True, and can be defined at per compute level for a pod.

If it is not defined at a per compute level, the global value is taken for that compute.

If not defined at the compute level nor at the global level, the default of X520 is set. Here, the compute can be standalone or hyper-converged node.

Cisco VIM installation requires the controller node Rack IDs to be unique, to indicate the physical rack location, so that physical redundancy is built within the controllers. If all the controller nodes are installed in the same rack, you must assign a unique rack ID to prepare for future releases of Cisco VIM that include rack redundancy. However, compute and storage nodes do not have rack ID restrictions.

For Central Ceph cluster, swap the *storage_ip* with *cluster_ip*

- Cisco VIM requires the controller rack IDs to be unique, to indicate the physical rack location and provide physical redundancy for controllers.
- If your controllers are all in the same rack, you must still assign a unique rack ID to the controllers to provide for future rack redundancy. Compute and storage nodes have no rack ID restrictions.

Server Common

Under the SERVER_COMMON section, the options to define parameters at a global or per role type are available. Options include platform vendor, root_drive_type, VIC_admin_fec_mode, and so on. From Cisco VIM 3.4.6, you can turn on/off VIC Link Training via the *setup_data.yaml* for better compatiblity between Cisco VIC 1387 (40Gbps) and Cisco ACI.

```
SERVER_COMMON:
  server_username: root   # not user configurable

  # Optional global config to change VIC's Port Channel Enable configuration
  # option, from default True to False.  Applicable only for Cisco VIC that support Port Channel like UCS VIC
1457 25Gbps network adapter.
  # VIC_port_channel_enable: <True or False>  # This can also be specified or overridden at per server level
under server's hardware_info section.
  # Optional global config to change VIC's admin FEC mode, from default 'Auto'to either 'Off', 'cl74', or
'cl91' mode.  Applicable only for Cisco VIC
  # that support changing the admin FEC mode, like UCS VIC 1457 25Gbps network adapter.
  #VIC_admin_fec_mode: <Auto, Off, cl74, or cl91>  # This can also be specified
                                                   # or overridden at per server level under server's
hardware_info section.

  # Optional value, allow static override value for platform vendor and/or root
  # drive type for Operating System install instead of dynamic discovery at
  # runtime.
  #
  # Optional global config to specify the control plane's interfaces link
  # aggregration mode, default active-active (802.3ad) mode or active-backup
  # mode to support CIMC in-band shared mode with control plane interfaces
  #control_bond_mode: <802.3ad or active-backup>  # This can also be specified
                                                  # or overridden at per server level under server's
hardware_info section.
  #
  # Optional value, allow static override value for platform vendor and/or root
  # drive type for Operating System installation instead of dynamic discovery at runtime.
  #
  # vendor, allowed values:
  #   CSCO - Cisco Systems Inc
  #   HPE  - Hewlett Packard Enterprise
  #   QCT  - Quanta Cloud Technology Inc
  #   DELL - Dell Technology Inc
  # root_drive_type, allowed values:
  #   HDD or SSD - drive locate in front or rear drive bay
  #   M.2_SATA   - internal SSD drive (mandatory config if booting off M.2 SATA
  #                                    SSD, not valid for M4 platform)
  #   NVMe       - internal NVMe drive (mandatory config if booting off M.2
  #                                     NVMe)
  # num_root_drive: Optional field, number of drives to use for root/boot
  #                 virtual drive, valid range from 0 to 26
  #                 - 0 for all available drives
  #                 - 1 to 26 (this value includes dedicated hot spare count)
  # root_drive_raid_level: Optional field, RAID level to use for creating the
  #                        root/boot virtual drive
  #                        - raid0 (require at least 1 drive)
  #                        - raid1 (require at least 2 drives)
  #                        - raid5 (require at least 3 drives)
  #                        - raid6/raid10 (require at least 4 drives)
```

```
  # root_drive_raid_spare: Optional field, number of drives to reserve as
  #                        dedicated hot spare for the RAIDed root/boot virtual
  #                        drive, valid range from 0 to 4
  #
  #  vendor: <CSCO or QCT>    <= Global level vendor override, all servers
  #  control:
  #    hardware_info:         <= Role level override, all controls
  #      vendor: <CSCO or QCT>
  #      root_drive_type: <HDD, SSD, M.2_SATA, or NVMe>
  #  compute:
  #    hardware_info:         <= Role level override, all computes
  #      vendor: <CSCO, HPE, DELL, or QCT>
  #      root_drive_type: <HDD, SSD, M.2_SATA, or NVMe>
  #      num_root_drive: <0 to 26>
  #      root_drive_raid_level: <raid0, raid1, raid5, raid6, or raid10>
  #      root_drive_raid_spare: <0 to 4>
  #  block_storage:
  #    hardware_info:         <= Role level override, all storages
  #      vendor: <CSCO or QCT>
  #

  # VIC_link_training: '<OFF or ON>'            #Default is ON; optionally can be defined at a per server
level
```

Fan Policy for Cisco UCS

From Cisco VIM 3.4.6, you can set the fan policy on a Cisco UCS C-series based pod as part of Day 0 configuration. Based on the fan policy, you can determine the right fan speed for the servers which in turn can help address the problems related to CPU performance and low power consumption due to over-heating. Define the FAN_POLICY as part of Day 0 installation of the Cisco UCS C-series based pod.

> ⚠ Cisco VIM does not manage or default to any fan policy, when the fan policy is not specified in the *setup_data.*

```
FAN_POLICY: <'Balanced' or 'Maximum_Power' or 'High_Power' or 'Low_Power'> # Optional
```

The available settings for fan policy are:

- Balanced: Not suitable for servers with Peripheral Component Interconnect Express (PCIe) cards as they overheat easily.
- Low_Power: Default setting for systems with no PCIe cards.
- High_Power: For fan speeds ranging from 60% to 85%
- Maximum_Power: For fan speeds ranging from 70% to 100%.

Compute Level Options

Compute Level Options

- NFV Hosts Configuration
- VM HugePage Size
- VM HugePage Percentage
- Hyper-threading
- VIC Link Training
- Isolating CPUs Using Tuned Profile
- libvirt writethrough Configuration

NFV Hosts Configuration

Cisco VIM supports CPU pinning and huge page on the compute nodes. To enable non-uniform memory access (NUMA), you can use ALL (case insensitive) to configure all compute nodes. For VPP/VLAN, only the value of ALL is allowed. For OVS/VLAN, alternatively, you can list the compute nodes where NUMA must be enabled.

```
# For VPP, only
NFV_HOSTS: ALL --> Preferred option
or
NFV_HOSTS: "compute-server-1", ..." --> "," separated server name
```

VM HugePage Size

You can enable HugePages and set values for HugePage size at a global level or at a per sever level as part of Day 0 configuration. This guarantees the availability of memory resources, when VMs are launched. The supported values are 2M and 1G. You can change the value on a per compute basis on Day n, by updating the *setup_data* and removing or adding compute nodes. When using this parameter, ensure that NFV_HOSTS is set in the *setup_data.*

```
# Optional, Day 0 parameter at global level

VM_HUGEPAGE_SIZE: <2M or 1G> #by default, it is not enabled

# or at a per server level:
..
SERVERS:
  server_name_1:
    VM_HUGEPAGE_SIZE: <2M or 1G>
...
```

VM HugePage Percentage

You can set HugePage percentage at a global level or at a per sever level as part of Day 0 configuration, when open vSwitch (OvS) is used as the mechanism driver. When VPP is used as the mechanism driver, this value is set to 100% and cannot be changed. However, this value can be set in the range of 0-100% for OVS at a global level or per compute level. To change the value on a per compute basis on Day n, update the *setup_data* and remove or add compute nodes. When using this parameter, ensure that NFV_HOSTS is set in the *setup_data.*

```
# Optional, Day 0 parameter at global level for mechanism driver of open vSwitch

VM_HUGEPAGE_PERCENTAGE: <0 to 100>  # default value, if not defined is 0

# or at a per server level:..
SERVERS:
  server_name_1:
    VM_HUGEPAGE_PERCENTAGE: <True or False>
...
```

If the pod is enabled with HugePages, care must be taken to use the right flavor, to ensure that the system memory is not used to launch the VMs. The usage of system memory for VMs, can lead to Cisco VIM instability, as both the workload and the infrastructure are competing for the resources reserved for the infrastructure.

Hyper-threading

By default, hyper-threading is enabled across compute nodes in Cisco VIM. Based on certain VNF characteristics, you can disable hyper-threading across the pod on Day 0. You can also disable it on a single compute node on Day n, by updating the *setup_data* and removing or adding compute nodes. To disable hyper-threading, update the *setup_data* with the following name or value pair before starting the installation.

```
# Optional, Day 0 parameter at global level

DISABLE_HYPERTHREADING: <True or False>   # default is False

# or at a per server level:
..
SERVERS:
  server_name_1:
    DISABLE_HYPERTHREADING: <True or False>
...
```

VIC Link Training

In Cisco VIM, you can enable Cisco VIC link training on Day 0 for VIC card (for example, VIC 1387 40Gbps) that supports changing of link training option. To enable this option, update the *setup_data* with the following name or value pair before starting the installation. To change the value on a per compute basis on Day n, update the *setup_data* and remove or add compute nodes.

```
# Optional

SERVER_COMMON:
  VIC_link_training: '<OFF or ON>'

#or at a per server level:
..
SERVERS:
  server_name_1:
    hardware_info: {VIC_link_training: '<OFF or ON>'}
...
```

Isolating CPUs Using Tuned Profile

Typically in Cisco VIM, cpu-partitioning via tuned profile is used for allocating reserved CPUs to VNFs. However, in cases where a few additional percentage of performance is critical, you can use isolcpu kernel as a Day 0 parameter. To change the value on a per compute basis on Day n, update the setup_data and remove or add compute nodes. Extra care needs to be taken to ensure that the VNFs are pinned when using this parameter, which otherwise results in performance degradation. As all VNFs are not CPU pinned in reality, an option is available to enable this parameter at a global level or at a per compute level. When using this parameter, ensure that NFV_HOSTS is set in the *setup_data.*

```
# Optional, Day 0 parameter at global level

ENABLE_ISOLCPU: <True or False> # <== optional, day 0 parameter

# or at a per server level:
..
SERVERS:
  server_name_1:
    ENABLE_ISOLCPU: <True or False>
...
```

libvirt writethrough Configuration

In Cisco VIM, libvirt is configured with writeback option. However, in certain cases, VNFs expect the option of libvirt to be in write-through mode. Extra care must be taken when enabling this option, as it leads to performance degradation. As all VNFS does not need write-through option in reality, an option is available to enable this parameter at a global level or at a per compute level.

```
# Optional, parameter at global level

LIBVIRT_WRITETHROUGH_CACHE: <True or False> # <== optional

# or at a per server level:
..
SERVERS:
  server_name_1:
    LIBVIRT_WRITETHROUGH_CACHE: <True or False>
...
```

You can also bring in the write-through option through reconfiguration. For details on how to enable LIBVIRT_WRITETHROUGH_CACHE via reconfiguration, see General Guidelines.

OpenStack Configuration

OpenStack Configuration

- OpenStack Admin Credentials
- OpenStack HAProxy and Virtual Router Redundancy Protocol Configuration
- OpenStack DNS Name Configuration
- OpenStack TLS and HTTPS Configuration
- OpenStack Glance Configuration with Dedicated Ceph or NetApp
- OpenStack Keystone Configuration
- CPU Allocation for Compute in Pods
- CPU Allocation for Ceph in Hyper-converged or Micropod Systems
- Memory Allocation for Ceph in Hyper-converged or Micropod Systems
- Infrastructure Memory Allocation for Computes in Pods
- Ceph Placement Group Information (Optional)
- OpenStack Glance Configuration
- OpenStack Cinder Configuration with Dedicated Ceph/NetApp
- OpenStack Settings on PODTYPE: Ceph for Glance Image service
- OpenStack Cinder Option: Flatten Volume from Snapshot
- OpenStack Settings on PODTYPE: Edge for Glance Image service
- OpenStack Nova Configuration
- OpenStack Neutron Configuration
- Connectivity between Edge and Central Ceph pods
- Ceph Tuning
- Support Custom Mapping Between OpenStack physnet to SRIOV PF
- Evacuating Neutron Networks from a Controller Before Replacing it

OpenStack Admin Credentials

```
ADMIN_USER: <admin>
ADMIN_TENANT_NAME: <admin tenant>
```

OpenStack HAProxy and Virtual Router Redundancy Protocol Configuration

Following are the configuration parameters:

- external_lb_vip_address: An externally routable ipv4 address in API network.
- external_lb_vip_ipv6_address: An externally routable ipv6 address in API network.
- VIRTUAL_ROUTER_ID: vrrp_router_id #for example: 49 (range of 1-255).
- internal_lb_vip_address: <Internal IP address on management network>.
- internal_lb_vip_ipv6_address: <Internal IPv6 address on management network> # optional, only for dual stack environment.

OpenStack DNS Name Configuration

For web and REST interfaces, names are commonly used instead of IP addresses. You can set the optional *external_lb_vip_fqdn* parameter to assign a name that resolves to the *external_lb_vip_address* as given below:

```
external_lb_vip_fqdn: host or DNS name matching external_lb_vip_address
```

You must configure the services to ensure that the name and address matches. Resolution can be made through DNS and the Linux /etc/hosts files, or through other options supported on your hosts. The Cisco VIM installer adds an entry to */etc/hosts* on the management and other Cisco NFVI nodes to ensure that this resolution can be made from within the pod. You must ensure that the resolution can be made from any desired host outside the pod.

OpenStack TLS and HTTPS Configuration

Enabling TLS is important to ensure that the Cisco VIM network is secure. TLS encrypts and authenticates communication to the cloud endpoints. When TLS is enabled, two additional pieces of information must be provided to the installer: *haproxy.pem* and haproxy-ca-crt. These must be placed in the ~ */installer-xxxx/openstack-configs* directory.

- *haproxy.pem* is the server side certificate file in PEM format. It must include the server certificate, any intermediate certificates, and the private key for the server. The common name of the certificate must match the external_lb_vip_address and/or the external_lb_vip_fqdn as configured in the *setup_data.yaml* file.
- *haproxy-ca.crt* is the certificate of the trusted authority that signed the server-side.

For production cloud, these certificates are provided by a trusted third-party CA according to your company IT policy. For cloud test or evaluation, self-signed certificates are used quickly enable TLS. For convenience, the installer includes a script that creates and install self-signed certificates.

As these certificates are generated by this tool for testing purposes, do not use these certificates for production.

To use this tool, make the following changes to the setup data file and run the tool:

```
external_lb_vip_address: <IP address on external network>
external_lb_vip_tls: True
external_lb_vip_fqdn: host or DNS name matching external_lb_vip_address (if FQDN is needed)
```

To run the tool, from the */working_dir/* directory, execute the command:

```
#./tools/tls_cert_gen.sh -f openstack-configs/setup_data.yaml
```

OpenStack Glance Configuration with Dedicated Ceph or NetApp

For OpenStack Glance (the OpenStack image service), the dedicated Ceph object storage configuration is given below. As the Ceph and Glance keys are generated during the Ceph installation, you do not need to specify the keys in *setup_data.yaml* file.

```
STORE_BACKEND: ceph/netapp/file #supported as ceph for ceph backend storage netapp for netapp backend; file
option is valid only for nanopod as no persistent storage exists
```

OpenStack Keystone Configuration

For OpenStack Keystone, the number of workers servicing requests on port 5000 (public) and port 35357 (admin) can be changed from the default values of 5 for each kind. Also the cache backend used by keystone may be changed to memcached. The change is made to the /root/openstack-configs /openstack_config.yaml file, KEYSTONE_CONFIG group of options. For example, to configure keystone to have 12 workers serving public requests and 10 servicing admin requests and use memcached as the caching backend:

KEYSTONE_CONFIG section in openstack_config.yaml

```
KEYSTONE_CONFIG:
   public_workers: 12
   admin_workers: 10
   use_memcached: True
```

After changing the file, run ciscovim reconfigure to modify the keystone configuration files in each controller. The reconfiguration results in keystone containers restarting to have the new configuration take effect. The new configuration results in 12 worker processes servicing the requests from port 5000 and 10 for port 35357.

The aim of increasing the workers servicing keystone requests is to improve the response time for requests. Adding more workers reduces the occurrences of timeout errors (HTTPD 503).

Using memcached as the cache backend enables keystone to process more requests with the same number of workers. With KEYSTONE_CONFIG. use_memcached set to True, the observed request rate processed is 3.8 requests per keystone public worker. When the parameter is False, the observed rate is 2.8 per public worker.

Increase the worker size only if there are free CPUs available to run the additional workers. For controller in full-on pods, CPUs should be available to run new workers. This may not hold true if a controller also has the compute role.

CPU Allocation for Compute in Pods

For some deployments, the number of CPU cores allocated to the computes (for infrastructure) must be more than the default value of two. From Cisco VIM 3.4.6, the option NR_RESERVED_HOST_PCORES is available only on fresh installation, with the values lying in the range of 2 to 30. One can set this value globally or on a per compute basis as well, but ensure that the number of cores chosen for the infrastructure is adequate for the application. Administrator must plan and use the right number of CPU cores for compute infrastructure, to ensure that enough CPU resources are available for the rest of the infrastructure and applications.

You can configure this option by updating the *setup_data.yaml* file as given below:

```
# Number of cores associated to computes, # default value is 2, if not defined
# Optional, Day 0 parameter at global level

NR_RESERVED_HOST_PCORES: <2 - 30>

# or at a per server level:
..
SERVERS:
  server_name_1:
    NR_RESERVED_HOST_PCORES: <2 - 30>
```

CPU Allocation for Ceph in Hyper-converged or Micropod Systems

As the storage node is shared with other node types (for example, compute for Hyper-converged and control and compute for Micropod), there are deployments where the number of CPU cores allocated to the Ceph role needs to be higher than the default value of two. From release Cisco VIM 2.4.2, the option CEPH_OSD_RESEREVED_PCORES is available only on fresh installation for Micropod and hyper-converged pods. One can configure this option by using the following commands in *setup_data* where the value can range between 2 and 30. Administrator must plan on using the right number of CPU cores for Ceph, to ensure that enough CPU resources are available for the rest of the infrastructure and applications.

```
# Number of cores associated to CEPH-OSD in a micro, UMHC or NGNENAHC deployment.
# default value if not defined is 2
# CEPH_OSD_RESERVED_PCORES: <2 - 30>
```

Memory Allocation for Ceph in Hyper-converged or Micropod Systems

As the storage node is shared with other types of nodes (for example, compute for hyper-converged, and control and compute for Micropod), there are deployments where the amount of memory allocated to the Ceph role needs to be different than the default value of 24G. From release Cisco VIM 3.4.6, the option of CEPH_OSD_MEMORY_LIMIT is available only on fresh installation for Micropod and hyper-converged pods. You can configure this option by using the following commands in *setup_data* where the value can range between 16 and 32 GB. Administrator must plan on using the right number of memory (in GB) for Ceph, to ensure that enough memory resources are available for the rest of the infrastructure and applications.

```
# Amount of memory associated to CEPH-OSD services in a micro, UMHC or NGNENAHC deployment.
# default value if not defined is 24
#CEPH_OSD_MEMORY_LIMIT: <16 - 32>
```

Infrastructure Memory Allocation for Computes in Pods

For some deployments, the amount of memory allocated to the computes (for infrastructure) must be more than the default value of 25GB. From Cisco VIM 3.4.6, the option RESERVED_SYSTEM_MEMORY is available only on fresh installation, with the values lying in the range of 25 to 500. One can set this value globally or on a per compute basis, when NFV_HOSTS option is set. Administrator must plan and use the right amount of memory for compute infrastructure, to ensure that enough memory resources are available for the rest of the infrastructure and applications.

You can configure this option by updating the *setup_data.yaml* file as given below:

```
# Reserved memory for running system services in GB
# Applicable when NFV_HOSTS is enabled, default value if not defined is 25 GB

# Optional, Day 0 parameter at global level
RESERVED_SYSTEM_MEMORY: <25 - 500>


# or at a per server level:
..
SERVERS:
  server_name_1:
    RESERVED_SYSTEM_MEMORY: <25 - 500>
```

Ceph Placement Group Information (Optional)

If you need to change the default percentages for placement group calculation, use this section to indicate the amount of data you expect in cinder/glance /nova. For NOVA_BOOT_FROM local, provide the values for cinder and glance. Additionally, for NOVA_BOOT_FROM ceph, provide nova_percentage_data for ephemeral data. All percentages need to add up to 100.

If no information is provided,

- code defaults to 60% cinder and 40% glance for NOVA_BOOT_FROM local.
- code defaults to 40% cinder, 30% glance and 30% nova ephemeral for NOVA_BOOT_FROM ceph.

You cannot change these values after deployment using update or reconfiguration.

```
For NOVA_BOOT_FROM local
CEPH_PG_INFO: {cinder_percentage_data: x, glance_percentage_data: y} # where x and y are integers and must add
up to 100

For NOVA_BOOT_FROM Ceph
CEPH_PG_INFO: {cinder_percentage_data: x, glance_percentage_data: y, nova_percentage_data: z}
where x, y and z are integers and must add up to 100
```

OpenStack Glance Configuration

```
STORE_BACKEND: <ceph or netapp based on backend storage>
```

OpenStack Cinder Configuration with Dedicated Ceph/NetApp

For OpenStack Cinder (the OpenStack storage service), the dedicated Ceph object storage configuration is given below.

```
VOLUME_DRIVER: ceph/netapp
```

Ensure that you do not change it. As the Ceph and Cinder keys are generated during the Ceph installation, you need not specify the keys in *setup_data.yaml* file. Use the *vgs* command to check your volume groups available on your controller nodes. The controller nodes run the Cinder volume containers and hold the volume groups for use by Cinder. If you have available disks and want to create a new volume group for Cinder use the *vgcreate* command.

OpenStack Settings on PODTYPE: Ceph for Glance Image service

Following are the examples for *central_ceph setup_data* details:

```
STORE_BACKEND: 'ceph'
VOLUME_DRIVER: 'ceph'
```

OpenStack Cinder Option: Flatten Volume from Snapshot

To flatten volumes created from snapshots with Ceph, use the below command:

```
CINDER_RBD_FLATTEN_VOLUME_FROM_SNAPSHOT: <True or False> # Optional, default is False, and is reconfigurable.
```

Cinder performs a CEPH rbd flatten operation directly, after the snapshot creation. This results in copying of the data from the parent to the new volume rather than using references to it.

- As full copies of the parent volume are created, additional space is required from the CEPH pool as compared to the regular snapshot.
- The time taken to create this volume clone in Cinder is long, as the flatten operation that requires data copy is additionally done after the initial snapshot.
- For larger volumes, calculate the additional time required as the copying time is non-trivial and dependent on the OSD server and storage hardware designs. Ensure that considerations (including testing in the lab with real NFs) are taken carefully, when it comes to CEPH storage space use and volume creation time.



This Cinder RBD flatten volume is not supported in edge or nano pods.

OpenStack Settings on PODTYPE: Edge for Glance Image service

For the edge pod installation to be successful, the central Ceph cluster with which it communicates for glance image service must be up and running. For the edge pod to communicate with the central Ceph cluster, the following configurations are needed:

```
MON_HOSTS: <3 IPv4 or IPv6 addresses, of the cephcontrol servers in the central ceph cluster>
MON_MEMBERS: <3 IPv4 or IPv6 addresses of the cephcontrol servers in the central ceph cluster>
CLUSTER_ID: <ceph_cluster_id> to fetch the CLUSTER_ID of the central ceph cluster, ssh to the management node
of the "ceph" pod, and execute the following:
#cat /root/openstack-configs/ceph/fetch/ceph_cluster_uuid.conf to get the CLUSTER_ID GLANCE_RBD_POOL: images

GLANCE_CLIENT_KEY: <key_info> to fetch the GLANCE_CLIENT_KEY, ssh to the management node of the "ceph" pod, and
execute the following:
# cd /root/openstack-configs/ceph/fetch/
# ls to get the UUID
# cd /root/openstack-configs/ceph/fetch/<UUID>/
# cat etc/ceph/ceph.client.glance.keyring
```

OpenStack Nova Configuration

To reduce the boot time, the NOVA_BOOT_FROM parameter is set to local for Cisco VIM. While this reduces the boot time, it does not provide Ceph back end redundancy. For typical NFVI workloads, you must not enable this option (it will default to local). To overwrite it, you can set NOVA_BOOT_FROM to Ceph . This is applicable only when the backend is ceph. For Netapp, no entry for this parameter is allowed.

```
#Nova boot from CEPH/local
 NOVA_BOOT_FROM: <ceph or local> #optional, if not defined will default to local
```

OpenStack Neutron Configuration

Following is the OpenStack Neutron configuration:

```
# ML2 Conf - reference implementation of OVS/VLAN

MECHANISM_DRIVERS:
openvswitch TENANT_NETWORK_TYPES: "VLAN"

# VLAN ranges can be a single continuous range or comma separated discontinuous range
TENANT_VLAN_RANGES: <a:b,c:d>  # with a minimum of 2 VLANs

# Jumbo MTU functionality.
# ENABLE_JUMBO_FRAMES: True for provider networks, just specifying the provider in the segments under the
NETWORKING section is enough.
# Use phys_prov as physical_network name when creating a provider network
```

Ensure that you include the PROVIDER_VLAN_RANGES information in the *setup_data* as given in the following syntax:

```
PROVIDER_VLAN_RANGES: <a,b:c,d:e>, where the VLAN ranges can be a continuous range or comma separated
discontinuous range.
```


When creating an external or provider network, use physical_network=phys_ext (need to be specified) or physical_network=phys_prov (need to be specified), respectively.

In a VLAN setup, by default, the MTU size is set to 1500 (1450 for VXLAN) and 8972 bytes. When JUMBO_MTU is enabled (with 28 bytes left for the header), the VLAN MTU is 9000 and VXLAN is 8950.

Connectivity between Edge and Central Ceph pods

Consider the following to design the edge pods:

1. If the edge pod is dual-stack, then the central Ceph pod must be dual-stack. All communications between these two pods happen over IPv6.
2. In case of dual-stack deployments, if one pod is running Jumbo frame, the IP infrastructure between the edge and central Ceph pods must support large MTU on an end-to-end basis, as fragmentation is not supported over IPv6.
3. For an IPv4 based deployment of edge and central Ceph, the large MTU requirement between the pods can be relaxed if:

- One or neither of the pods are running with Jumbo frame enabled.
- The edge pod is enabled with Jumbo frame. In this case, the central Ceph pod can run without Jumbo frame to maintain end-to-end connectivity.
- Both the edge and central Ceph pod are running with Jumbo frame. In this case, the management network of the central Ceph can be run without Jumbo frame. Also, the central pod is running off Intel NICs or Cisco VIC where the control and data plane are separated (not applicable to quanta based pods). To enable this option, install the central Ceph pod with the option of MTU_BR_MGMT set to 1500 as part of Day 0 configuration:

```
MTU_BR_MGMT: <1500 or 9000> # Optional, defaults to the pod's setting.
```

Ceph Tuning

From Cisco VIM 3.4.6, a global parameter CEPH_TUNING is defined to improve overall OSD failure detection time. You can enable this parameter on Day 0 or Day n via a reconfiguration option. For reconfiguration details, see General Guidelines

```
CEPH_TUNING: <True or False> # Optional, default is False
```

⚠ The changes made via reconfiguration must be done in a maintenance window, as it takes time for Ceph to go to HEALTH_OK state.

The following table shows the details of the ceph parameters managed via the global knob of CEPH_TUNING and their corresponding values:

Parameter Name	Description	Default Value	New Value via CEPH_TUNING
mon_osd_adjust_heartbeat_grace	If the parameter is set to True, will calculate laggy interval logic when OSD disks are flapping multiple times. Due to that grace timer (osd_heartbeat_grace), range varies from 20 sec to 300 sec. If the parameter is set to False, marks the disk as failed exactly at 20 sec.	True	False
osd_heartbeat_grace (sec)	Each Ceph OSD daemon checks the heartbeat of other Ceph OSD daemons at random intervals less than every 6 seconds. If a neighboring Ceph OSD daemon does not indicate a heartbeat within a 20 second grace period, the Ceph OSD daemon may consider the neighboring Ceph OSD daemon down and report it back to a Ceph monitor, which updates the Ceph Cluster Map. You may change this grace period by adding an osd heartbeat grace setting under the [mon] and [osd] or [global] section of your Ceph configuration file, or by setting the value at runtime.	20	10
osd_max_markdown_count (sec)	If Ceph monitor is marking an OSD down and the OSD maximum markdown count is more than 5 within osd_max_markdown_period value (which is 600 sec by default), the OSD will shutdown itself.	5	1
osd_client_watch_timeout (sec)	A watch operation registers the client as being interested in notifications on an object. As OSDs keep track of watches on persistent storage, they are preserved across cluster changes by the normal recovery process. If the client loses its connection to the primary OSD for a watched object, the watch is removed after a timeout configured with osd_client_watch_timeout. Watches are automatically reestablished, when a new connection is made or a placement group switches the OSDs.	30	10

⚠

- The above settings are useful when VNF is I/O intensive. Also helps to avoid application crash, when Ceph becomes unavailable (for example, HEALTH_ERR state) for a shorter time window.
- After applying Ceph tuning option, if any OSD disk is marked as down more than one time within 600 sec timeframe, the OSD service is stopped completely. Manual intervention is needed to start the Ceph OSD service again.

Support Custom Mapping Between OpenStack physnet to SRIOV PF

Choose the physnet information to create networks and subsequently create SRIOV ports that are bind to VM. These SRIOV port(s) in turn determines the NUMA node off which the VM gets CPU and memory resources. In Cisco VIM, you can map the physnet to multiple SRIOV physical functions (PFs) as a non-reconfigurable Day 0 option. With this flexibility, OpenStack physnet can map to multiple PFs across NUMAs. In such environment, the decision is left for Nova scheduler to select the right PF based on resource availability during VM launch time.

For this option, the following assumptions need to hold:

- Pod must be enabled with SRIOV.
- The physnet name must start with the prefix: *phys_sriov_*.
- All SRIOV PFs must be mapped into the CUSTOM_SRIOV_PHYSNET_MAPPING section.
- Every SRIOV PF can be mapped to one and only one physnet.
- This configuration overrides the default 1:1 conventional mapping.

To avail this option, enable the following in your *setup_data* as part of Day 0 installation of the pod.

```
# Optional, specify the custom mapping between OpenStack physnet to SRIOV PF.

CUSTOM_SRIOV_PHYSNET_MAPPING:
 sriov:
   phys_sriov_port_a: [<int1>, <int3>]
   phys_sriov_port_b: [<int2>, <int4>]

<int1> to <int4>: indicates the SRIOV PFs

# In the example below, "Port A Physnet" includes three SRIOV PFs
# when each SRIOV NIC has two PFs (e.g. XL710, XXV710, etc.)
#
#CUSTOM_SRIOV_PHYSNET_MAPPING:
# sriov:
#   phys_sriov_port_a: [0, 1, 3]
#   phys_sriov_port_b: [2]
#
# "sriov" section controls the mapping of regular SRIOV interfaces. The name
# "phys_sriov_port_a" will be the physnet to use when creating Neutron provider
# network. "[0, 1, 3]" means to use 1st, 2nd and 4th SRIOV PF ports, i.e. sriov0,
# sriov1, sriov3. "phys_sriov_port_b" will consume 3rd SRIOV PF port i.e. sriov2.
#
```

Ensure that the VM launch command includes *user_defined_physnet* as listed in the example below. Similarly in VMTP section, the correct PHYSNET_NAME must be provided.

```
# Example to launch a VM:

openstack network create prov-net-sriov --provider-physical-network=<user_defined*> --provider-network-
type=vlan --provider-segment <seg_id> --share --enable
openstack subnet create prov-subnet --subnet-range <cidr/mask> --network prov-net-sriov --allocation-pool
start=<start_ip_address>,end=<end_ip_address> --gateway <network_gateway> --no-dhcp --dns-nameserver <dns_1> --
dns-nameserver <dns_2>

*user_defined: is the name defined in CUSTOM_SRIOV_PHYSNET_MAPPING/sriov section of the setup_data.
```

Evacuating Neutron Networks from a Controller Before Replacing it

When a controller is being replaced, all neutron agents on the controller will be down. Consequently the dnsmasq servers running on the controllers will be down as well. Before the controller goes down, the neutron networks managed by the DHCP-agent should be managed by the DHCP agent running on another controller. Each neutron network is managed by two DHCP-agents. So there is only one remaining controller whose DHCP-agent is able manage the network. The procedure is:

1. Find the dhcp-agent IDs on all 3 controllers

Find IDs of all DHCP Agents

```
[root@mgmt ~]# openstack network agent list --agent-type dhcp
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
| ID                                   | Agent Type | Host         | Availability Zone | Alive | State |
Binary             |
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
| c1d8ff4a-50c9-4cc6-83a1-8960dcbc273e | DHCP agent | controller-1 | nova              | :-)   | UP    |
neutron-dhcp-agent |
| 6f3489d9-85af-429a-b8fe-3eba44f1dde8 | DHCP agent | controller-2 | nova              | :-)   | UP    |
neutron-dhcp-agent |
| 6e1bc616-d620-4ca3-a090-88fa78893e1a | DHCP agent | controller-3 | nova              | :-)   | UP    |
neutron-dhcp-agent |
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
[root@mgmt ~]#
```

2. Suppose we want to replace controller-1. We have to evacuate the neutron networks from it. Find the neutron networks managed by controller-1 (dhcp-agent ID is c1d8ff4a-50c9-4cc6-83a1-8960dcbc273e)

Find all networks managed by a controller

```
[root@mgmt ~]# neutron net-list-on-dhcp-agent c1d8ff4a-50c9-4cc6-83a1-8960dcbc273e
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.
+--------------------------------------+-----------+---------------------------------
+--------------------------------------------------+
| id                                   | name      | tenant_id                       |
subnets                                            |
+--------------------------------------+-----------+---------------------------------
+--------------------------------------------------+
| 7b808101-26ed-44ea-9bca-bd9014713318 | demo-net1 | 8550215dea4b455381251ca5183b2f55 | e074b963-b34d-4400-88e4-
b6ae7c0cab6b 10.22.44.0/24 |
+--------------------------------------+-----------+---------------------------------
+--------------------------------------------------+
[root@mgmt ~]#
```

When there are no networks listed, no networks needs to be moved. So we are done.

However, in the case here, the only network managed by the dhcp-agent is "demo-net1" (ID=7b808101-26ed-44ea-9bca-bd9014713318). So it has to be moved out of this controller.

3. For each network in the list from (2). Find the dhcp-agent IDs managing this network. Always use the network UUID, instead of the the name, since multiple networks may have the same name.

Move a network

```
[root@mgmt ~]# openstack network agent list --network 7b808101-26ed-44ea-9bca-bd9014713318
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
| ID                                   | Agent Type | Host         | Availability Zone | Alive | State |
Binary             |
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
| 6f3489d9-85af-429a-b8fe-3eba44f1dde8 | DHCP agent | controller-2 | nova              | :-)   | UP    |
neutron-dhcp-agent |
| c1d8ff4a-50c9-4cc6-83a1-8960dcbc273e | DHCP agent | controller-1 | nova              | :-)   | UP    |
neutron-dhcp-agent |
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
[root@mgmt ~]#
```

In this case, demo-net1 is being managed by dhcp agents on controller-1 and controller-2. As controller-1 will be down, only dhcp-agent on controller-3 (6e1bc616-d620-4ca3-a090-88fa78893e1a) is available to manage this network.

4. Doing the move. Remove demo-net1 from controller-1's dhcp-agent (c1d8ff4a-50c9-4cc6-83a1-8960dcbc273e). Then add the network to controller-3's DHCP-agent (6e1bc616-d620-4ca3-a090-88fa78893e1a)

Moving network from controller-1 to controller-2

```
[root@mgmt ~]# openstack network agent remove network --dhcp c1d8ff4a-50c9-4cc6-83a1-8960dcbc273e 7b808101-26ed-
44ea-9bca-bd9014713318
[root@mgmt ~]# openstack network agent add network --dhcp 6e1bc616-d620-4ca3-a090-88fa78893e1a 7b808101-26ed-
44ea-9bca-bd9014713318
[root@mgmt ~]#
```

5. Verify the network demo-net1 is no longer managed by dhcp-agent on controller-1:

```
[root@mgmt ~]# openstack network agent list --network  7b808101-26ed-44ea-9bca-bd9014713318
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
| ID                                   | Agent Type | Host         | Availability Zone | Alive | State |
Binary              |
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
| 6e1bc616-d620-4ca3-a090-88fa78893e1a | DHCP agent | controller-3 | nova              | :-)   | UP    |
neutron-dhcp-agent |
| 6f3489d9-85af-429a-b8fe-3eba44f1dde8 | DHCP agent | controller-2 | nova              | :-)   | UP    |
neutron-dhcp-agent |
+--------------------------------------+------------+--------------+-------------------+-------+-------
+--------------------+
[root@mgmt ~]#
```

6. Repeat (2) to see if there are other networks needs to be moved. If there are networks listed, repeat steps 3 - 5 on a network in the list. In this case, no more networks are listed. So we are done.

```
[root@mgmt ~]# neutron net-list-on-dhcp-agent c1d8ff4a-50c9-4cc6-83a1-8960dcbc273e
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.
[root@mgmt ~]#
```

7. Done.

VPP VLAN

VPP VLAN

- Cisco VPP/VLAN Mechanism Driver Configuration
- VPP with Generic Segmentation Offload (GSO)
  - Fresh Installation
  - Disabling GSO

Cisco VPP/VLAN Mechanism Driver Configuration

If you are installing Cisco VIM with VPP/VLAN, update the *setup_yaml* file as follows:

```
MECHANISM_DRIVERS: vpp
TENANT_NETWORK_TYPES: "VLAN"
TENANT_VLAN_RANGES: <START>:<END># arbitrary VLAN range***
NFV_HOSTS: ALL
NR_RESERVED_VSWITCH_PCORES: <int> #Optional, defaults to 2. Values in the range 2 to 6 to increase performance
by allocating more cores to VPP
```

VPP with Generic Segmentation Offload (GSO)

GSO is enabled by default, for improving performance, when enterprise workload (over Layer 3) is on-boarded with VPP.

As GSO is a tech-preview feature, ensure that you do not enable it in production.

Fresh Installation

For a freshly installed pod, enable the following option in your *setup_data* as part of the installation:

```
# Optional, Enable/Disable GSO with VPP, default is True
# VPP_ENABLE_GSO: <True or False>
```

To check if a pod is running with GSO or not, SSH to any of the computes and execute the following command:

```
# vpp vppctl show bond details | grep gso
```

Disabling GSO

To disable GSO on a pod that is running with VPP and GSO:

1. Log into the management node.
2. Choose tools directory from the current installer directory using the command:

   ```
   # cd /root/installer-xxx/tools/
   ```

3. Check if all the nodes come back with status *Patched* using the below command:

   ```
   #./toggle_gso_in_vpp.py --action check
   ```

4. Execute the following command:

   ```
   #./toggle_gso_in_vpp.py --action revert

   Wait for the prompt to come back successfully. Note that this will change all your compute nodes.
   ```

If you just want to try one compute first, add -s <SERVER_NAME> to the command. Post experimentation, all nodes must run with or without GSO in a single pod.

5. If the target compute has no existing VM running, no further action is needed. If you have existing VMs running on the target compute node, execute one of the following steps to manage your existing workload to run with the new GSO option:

    a. On the management node, do a *openstack server reboot --hard <VM_NAME_OR_ID>* to hard-reboot the VM on the target compute nodes.
    b. Reboot the target compute node in baremetal, that is, run *reboot* on the compute node.

6. To make the disablement of GSO persistent, do the following steps very carefully, so that subsequent pod management operations do not change the baseline configuration:

    a. Log into the management node.
    b. Copy the *setup_data* to a local directory and update it:

```
# mkdir /root/Save
# cd /root/Save
# cp /root/openstack-configs/setup_data.yaml /root/Save/setup_data_pre_gso_diable.yaml
# cp /root/Save/setup_data_pre_gso_diable.yaml /root/Save/setup_data_gso_diable.yaml
```

    c. Update the file */root/Save/setup_data_gso_diable.yaml* to have the following entry:

```
VPP_ENABLE_GSO: False
```

    d. Copy the *setup_data* back to the systems directory and update the *backup_setup_data*:

```
# cp /root/Save/setup_data_gso_diable.yaml /root/openstack-configs/setup_data.yaml
# cp /root/openstack-configs/setup_data.yaml  /root/openstack-configs/.backup_setup_data.yaml
```

    e. Restart the restapi server and execute the following commands:

```
# cd /root/installer-xxx/tools/
#./restapi.py -a restart
```

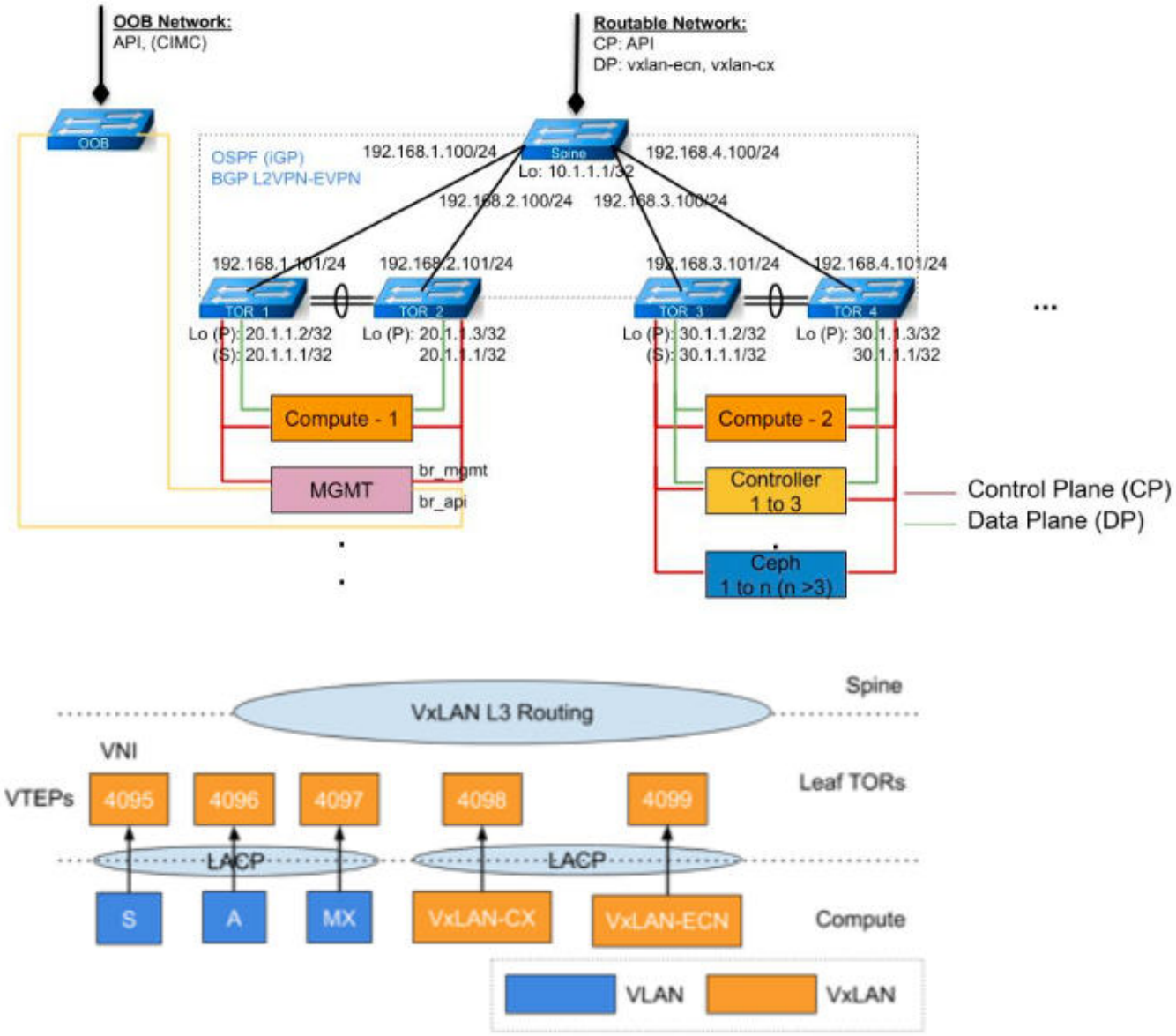
L3 Fabric Deployment Details

L3 Fabric Deployment Details

- High Level Depiction
- Highlights of Spine Switch
- Highlights of ToR Switch in a Leaf Pair
- Setup_Data to Support L3 Fabric Environment

High Level Depiction

The following figure is a high-level depiction of the Cisco VIM deployment in a Layer 3 environment.



In the above figure:

- LACP bonding exists between server and leaves.
- VxLAN fabric exists between spine and leaf.
- For control plane (samx) verified with PXE boot:
    - Packets leave the server with VLAN tag
    - Leaf encapsulates with VxLAN
- For data plane (vxlan-ecn, vxlan-cx):

- Packets leave the server as VxLAN
- Leaf encapsulates with VxLAN
- Double VxLAN encapsulation exists within fabric

Here are few assumptions:

- ToR can support MTU accounted for double encapsulation, that is, 9000 (VM) + 50 * 2 (per VxLAN) < 9216 switch capability.
- Computes with SRIOV are excluded.
- All configuration in the spine and north-bound configurations of the ToRs towards the spine are outside the scope of Cisco VIM.
- Cisco VIM's auto-ToR capability supports southbound configuration from the server to the ToRs.
- You can choose your own automation to configure the VPC domain, as long as the domain id for each ToR pair matches with that defined in setup_data.

Highlights of Spine Switch

Following are the highlights associated with the spine switch in the fabric:

- All configurations associated to the spine switch are outside the scope of Cisco VIM and must be in place before the Cisco VIM installation commences.
- API, vxlan-ecn, and vxlan-cx networks must be mapped to certain VNI for VxLAN encapsulation in the spine.
- VxLAN NVE interface must be configured to encapsulate API, vxlan-ecn, and vxlan-cx VNIs in the spine switch
- The underlay physical interfaces between spine and leaf must be configured in L3 mode, that is, *no switchport*.
- Loopback interface must be configured so that it is vPC compatible if more than one ToR is used.
- OSPF (or other iGP) must be configured to include underlay physical L3 interfaces between spine and leaf, and loopback interface for VxLAN encapsulation.
- BGP L2VPN EVPN must be configured between the spine and leaf as neighbors of each other.

Highlights of ToR Switch in a Leaf Pair

Following are the highlights associated with each ToR switch in a leaf pair:

1. Leaf configuration must be done ahead of Cisco VIM installation.
    - Underlay physical interfaces between spine and leaf must be configured in L3 mode, that is, *no switchport.*
    - Loopback interface must be configured to be vPC compatible if more than one ToR is used.
    - OSPF (or other iGP) must be configured to include underlay physical L3 interfaces between spine and leaf, and loopback interface for VxLAN encapsulation.
    - BGP L2VPN EVPN must be configured between the spine and leaf as neighbors of each other.
2. Leaf configuration that is done as part of Cisco VIM installation
    - Mapping of storage, API, management/provision, vxlan-ecn, and vxlan-cx networks to certain VNI for VxLAN encapsulation is defined in the setup_data and is automated.
    - Connections between servers and leaf ToRs are configured in LACP bonding.
    - VxLAN NVE interface is configured to encapsulate storage, provider network VLANs, management/provision, API, vxlan-ecn, and vxlan-cx VNIs.

Setup_Data to Support L3 Fabric Environment

To support the Cisco VIM deployment in a L3 fabric environment, update the setup_data as follows:

```
NETWORKING:
  domain_name: cisco.com
  domain_name_servers: [173.36.131.10, 171.70.168.183]
  networks:
  - gateway: 10.11.183.1
    pool: [10.11.183.20 to 10.11.183.100]
    segments: [management, provision]
    subnet: 10.11.183.0/24
    vlan_id: 1462
    l3_fabric_vni: 4996   # -------------> VXLAN vnid to be used for mapping to VLAN
  - gateway: 172.29.85.97
    segments: [api]
    subnet: 172.29.85.96/27
    vlan_id: 853
    l3_fabric_vni: 4997  #  -------------> VXLAN vnid to be used for mapping to VLAN
  - gateway: 25.0.0.1
    pool: [25.0.0.11 to 25.0.0.50]
    segments: [tenant]
    subnet: 25.0.0.0/24
    vlan_id: None
  - segments: [external]
    vlan_id: 1463
```

```
  - segments: [provider]
    vlan_id: None
  - gateway: 10.1.1.1
    pool: [10.1.1.100 to 10.1.1.199]
    segments: [vxlan-tenant]
    subnet: 10.1.1.0/24
    vlan_id: 2998
    l3_fabric_vni: 4998  #  -------------> VXLAN vnid to be used for mapping to VLAN
- gateway: 10.2.2.1
    pool: [10.2.2.100 to 10.2.2.199]
    segments: [vxlan-ecn]
    subnet: 10.2.2.0/24
    vlan_id: 2999
    l3_fabric_vni: 4999  #  -------------> VXLAN vnid to be used for mapping to VLAN

- gateway: 11.2.2.1
    pool: [11.2.2.100 to 11.2.2.199]
    segments: [storage]
    subnet: 11.2.2.0/24
    vlan_id: 3000
    l3_fabric_vni: 5999  #  -------------> VXLAN vnid to be used for mapping to VLAN

L3_PROVIDER_VNI_RANGES: "a:b,c:d"   ----> 4100 < a < 2^24-1  ---> will map to PROVIDER_VLAN_RANGES

TORSWITCHINFO:
  CONFIGURE_TORS: true
  SWITCHDETAILS:
    - br_mgmt_port_info: <ethx/y> # management interface of management node
      br_mgmt_po_info: <int> # port channel number for management interface of management node
      vpc_peer_port_info: '<ethx/y,etha/b>' # interface for vpc peer ports
      vpc_peer_vlan_info: '<vlanid1-vlanid5,vlanid10>' # vlan ids for vpc peer ports (optional)
      hostname: <switch_a_hostname>
      password: <switch_a_password>
      ssh_ip: <switch_a_ssh_ip>
      username: <switch_a_username>
      vpc_domain: <int>                          # must match with that in peer switch
      vpc_peer_keepalive: <switch_b_ssh_ip>
      l3_fabric_loopback: loopback0  # ---------> Loopback Interface name, must match with peer switch loopback
number

    - br_mgmt_port_info: <ethx/y> # management interface of management node
      br_mgmt_po_info: <int> # port channel number for management interface of management node
      vpc_peer_port_info: '<ethx/y,etha/b>' # interface for vpc peer ports
      vpc_peer_vlan_info: '<vlanid1-vlanid5,vlanid10>' # vlan ids for vpc peer ports (optional)
      hostname: <switch_b_hostname>
      password: <switch_b_password>
      ssh_ip: <switch_b_ssh_ip>
      username: <switch_b_username>
      vpc_domain: <int>                          # must match with that in peer switch
      vpc_peer_keepalive: <switch_a_ssh_ip>
      l3_fabric_loopback: loopback0  # ---------> Loopback Interface name, must match with peer switch loopback
number

#Optional, needed when additional leaf pairs help build the cloud
    - hostname: <switch_c_hostname>
      password: <switch_c_password>
      ssh_ip: <switch_c_ssh_ip>
      username: <switch_c_username>
      vpc_domain: <int>                          # must match with that in peer switch
      vpc_peer_keepalive: <switch_d_ssh_ip>
      vpc_peer_vlan_info: '<vlanid1-vlanid5,vlanid10>' # vlan ids for vpc peer ports (optional)
      l3_fabric_loopback: loopback0  # ---------> Loopback Interface name, must match with peer switch loopback
number

    - hostname: <switch_d_hostname>
      password: <switch_d_password>
      ssh_ip: <switch_d_ssh_ip>
      username: <switch_d_username>
      vpc_domain: <int>                          # must match with that in peer switch
      vpc_peer_keepalive: <switch_c_ssh_ip>
      vpc_peer_vlan_info: '<vlanid1-vlanid5,vlanid10>' # vlan ids for vpc peer ports (optional)
```

```
      l3_fabric_loopback: loopback0  # ---------> Loopback Interface name, must match with Peer switch loopback
number where
```

- The L3 fabric VNI is a unique value between 4100 and 2^24-1.
- The loopback number must match between ToR pairs and the L3_PROVIDER_VNI_RANGES must include vni values.
- One-to-one mapping exists between PROVIDER_VLAN_RANGES and L3_PROVIDER_VNI_RANGES.

Along with Day 0 installation, Cisco VIM supports RMA of leaf ToR (one at a time). For more details, see Supporting RMA for Auto-ToR. To expand the pod, you can add ToR in pairs, as part of add compute/storage or replace controller.

The expansion of ToR can be done but one ToR pair at a time.

Supporting RMA for Auto-ToR

Supporting Return Merchandise Authorization for Auto-ToR

When Cisco VIM cloud uses auto-ToR configuration to manage switch ports, you need to replace the existing switches if one malfunctions.

Consider the following assumptions made during RMA of ToR with auto-ToR configuration:

- When a switch is getting RMAed, it is in a virtual port-channel (vPC) mode with another switch to support full switch redundancy.
- You can replace multiple switches through Cisco VIM CLI, but only one switch from each pair.
- Administrator is responsible for manual configuration of the spine connection and L3Out for the ToR.

> ⚠ When replacing the ToR, ensure that you use the same server ports to have minimal changes in the setup_data. Also, ensure that the new ToR name is same as the one that you are replacing.

To initiate ToR RMA, take a backup of the setupdata file and update it manually with the configuration details by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/  update the setup_data to include the
changes associated to the ToR that needs to be RMAs
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml -rma_tors <"," separated target
ToRs>
```

Optional Services

Optional Services

- Heat and Ceilometer
- Ironic Support
- LDAP Support

Heat and Ceilometer

Heat and Ceilometer Support

- Heat Support
- Ceilometer Support

Heat Support

OpenStack Heat is an orchestration service that allows you to spin up multiple instances, logical networks, and other cloud services in an automated fashion. To enable Heat, add the following in the *setup_data.yaml*.

```
# Optional Services:
OPTIONAL_SERVICE_LIST:
-  heat
```

To disable heat, remove the Optional Services section from the *setup_data.yaml* file.

Auto-scaling is not supported in Cisco VIM.

Ceilometer Support

The reference implementation of ceilometer is available from Cisco VIM 3.0.0 onwards. The ceilometer service can be brought in as a Day 0 option for fullon pod. To enable this service, update the *setup_data.yaml* with the following:

```
#Optional Services:
OPTIONAL_SERVICE_LIST:
- ceilometer
```


Ceilometer is enabled, when the pod type is fullon.

Ironic Support

Ironic Support

- Overview
- Deploying Ironic

Overview

The reference implementation of ironic is available for Cisco VIM. You can implement the ironic service as a Day 0 or a reconfigure option. You can not disable this service after you enable it.

Ironic support is available only on Cisco UCS C M4 and M5 baremetal servers when Cisco VIM is deployed with OVS or VPP as the mechanism driver. You can use only Nexus 9K as the ToR for Cisco VIM with the ironic service. The ironic interface on the bare metal servers for OpenStack can either be an MLOM interface or the onboard LOM port. Cisco VIM with the ironic service supports only the configuration of a single interface on the baremetal server. This interface can be one of the above-mentioned ports, or if you want to bond ports, Cisco VIM configures an available bond 0 interface on the user deployed image.

Cisco VIM supports the bonding of MLOM or onboard LOM ports from Release 3.4.1. You must deploy the ToR used for Ironic service in the Nexus mode.

You must have one separate network segment that is used for ironic management and ironic inspector. The inspector is a service used to automate the creation of the OpenStack baremetal port with a switch interface, for example, eth 1/39 and MAC address information of both the switch MAC and server interface MAC apart from automatically adding the deploy image information to the ironic node.
You must ensure that the ironic management, ironic inspector, Cisco VIM management, and ironic CIMC networks are routed to each other.

The Cisco VIM management must be able to reach:

- Ironic management network and vice-versa
- CIMC network of the ironic nodes so that the Cisco VIM controller servers can directly reach the CIMC IP of the ironic servers.

To enable network reachability:

- All three networks such as Cisco VIM management, Ironic management, and CIMC must be private networks with SVI interfaces on the ToR.

You must deploy a routed network for all three network segments. In this case, Cisco VIM does not require SVI interfaces on the ToR.

It is mandatory to include the ironic management or ironic inspector VLANs on the ToR interfaces connected to all the mercury controller servers. You must manually configure this setting if the Cisco VIM auto-ToR feature is not enabled.

Deploying Ironic

Follow the below steps for ironic deployment, before installing Cisco VIM:

1. Create a separate *ironic_inventory.yaml* with CIMC or IPMI details of the servers to be used as ironic baremetals. If you want to bond interfaces on the baremetal server, ensure that you configure the corresponding portgroups field with information about the interfaces. For example, you can refer:

   ```
   /root/installer-XXX/openstack-configs/ironic_inventory.yaml
   ```

   Save this file with your ironic server details in */root/installer-XXX/openstack-configs/ironic_inventory.yaml.*
2. Specify the ironic management or ironic inspector VLAN in all control interfaces of the mercury controller servers, if Cisco VIM auto-ToR feature is not enabled. This configuration is required to perform ironic introspection to transfer the images from the controller to the baremetal server.
3. If you deploy ironic in the Nexus mode of ToR and you do not want to bond interfaces, you must manually configure the ToR interface connected to the server. Ensure that no existing configuration exists on the interface of the ToR connected to the baremetal. The interface must be in ACCESS mode. You must set only the ironic inspector VLAN as the access VLAN. If you configure portgroups in the *ironic_inventory.yaml* file to enable bonding, you do not have to manually configure the ToR as Cisco VIM configures it.
4. If you deploy ironic in an ACI mode testbed, you must ensure that ironic management network VLAN and all the tenant VLANs from setup_data are configured on the interface of the ToR connected to the baremetal the ironic inspector VLAN. The interface is in TRUNK mode. You need to set the ironic inspector network as the native VLAN.
5. Verify if the following configurations are done in the baremetal server CIMC before proceeding:

   - Check if IPMI connections are allowed over LAN.
   - In the BIOS configured boot order, only PXE boot is present and available as the first option. If the deployment is over MLOM, you must set the bootorder slot as MLOM. If you use the onboard NIC, you must set the slot as L.
   - PXE is enabled in VNIC adapters if VNICs are used as the interface for ironic. You can skip this step if you deploy ironic on the onboard LOM interface.
   - Set the VLAN mode on the VNIC used as TRUNK, if VNICs are used as the interface for ironic. You can skip this step if you deploy ironic on the onboard LOM interface.

- Turn on the baremetal node to access all parameters of CIMC. Cisco VIM installer verifies the node at step 1.
- Ensure that the Cisco UCS servers used as baremetal nodes have a RAID controller card and that a virtual raid has been created. All hard disks must be set in online or JBOD mode.

- Disable LLDP on Cisco VIC adaptor of all the servers used for ironic by executing the following steps and rebooting the server:

```
sh admin@X.X.X.X (CIMC IP)
C240-FCH1832V1HW# scope chassis
C240-FCH1832V1HW /chassis # show adapter
C240-FCH1832V1HW /chassis # scope adapter <PCI slot>
C240-FCH1832V1HW /chassis/adapter # set lldp disabled
C240-FCH1832V1HW*# commit
C240-FCH1832V1HW /chassis/adapter # show detail <To Verify LLDP is disabled>
```

To enable this service, update the *setup_data.yaml* file with the following configuration:

```
# Optional Services:
OPTIONAL_SERVICE_LIST:
-        ironic

IRONIC:
IRONIC_SWITCHDETAILS: # list of switches off which the ironic servers are hanging. This is mainly
used to provide ironic switch details to neutron
-        {hostname: <switch_name>, password: <password>, ssh_ip: <ssh_ip>, username:
<switch_admin_username>, switch_type: <"Nexus", "ACI", or "BypassNeutron">}


NETWORKING:
......
-        gateway: <gateway_information> # Mandatory if ironic is present pool: [<ip_add_start_1 to
ip_add_end_2>]
segments:  [ironic] subnet: <subnet with/mask>
vlan_id: <unique vlan id across the pod>
inspector_pool: [ip_add_3 to ip_add_4, ip_add_5 to ip_add_6, ip_add_7 to ip_add_8] (# of entry
pool : 3, same network as ironic but doesn't overlap with the pool of IPs defined
in the ironic segment)
# alternate format for pool (# of entry pool : 3)
-        ip_add_3 to ip_add_4
-        ip_add_5 to ip_add_6
-        ip_add_7 to ip_add_8
```

LDAP Support

LDAP Support

- Overview
- Integrating Identity with LDAP over TLS
- Support for Anonymous LDAP Bind
- CIMC Authentication Using LDAP

Overview

To continue enhancing the security portfolio and multi-tenancy with the use of domains, Keystone v3 support is now default in Cisco VIM 3.0.0. The OpenStack service authentication can be delegated to an external LDAP server, if the authorization is done by Keystone v3. To enable LDAP integration, the LDAP endpoint must be reachable from all the controller nodes that run OpenStack Keystone Identity Service. To benefit LDAP support with Keystone v3 feature, the setup_data must be augmented with the following information during the pod installation.

```
LDAP:

domain: <Domain specific name>

user_objectclass: <objectClass for Users> # e.g organizational Person
group_objectclass: <objectClass for Groups> # e.g. groupOfNames
user_tree_dn: '<DN tree for Users>' # e.g. 'ou=Users,dc=cisco,dc=com'
group_tree_dn: '<DN tree for Groups>' # e.g. 'ou=Groups,dc=cisco,dc=com'
suffix: '<suffix for DN>' # e.g. 'dc=cisco,dc=com'
url: '<ldap:// host:port>' # e.g. 'ldap://172.26.233.104:389'
user: '<DN of bind user>' # e.g.'dc=admin,dc=cisco,dc=com'
password: <password> # e.g. password of bind user
user_filter: '(memberOf=CN=os-users,OU=OS-Groups,DC=mercury,DC=local)' # Optional
user_id_attribute: sAMAccountName
user_name_attribute: sAMAccountName
user_mail_attribute: mail            #
Optional group_name_attribute: sAMAccountName
group_filter: '(&(objectClass=group)(|(cn=server-ops)(cn=admins)))' # Optional
group_member_attribute: memberUid  # Optional
group_id_attribute: gidNumber   # Optional
group_members_are_ids: True     # Optional
chase_referrals: <True or False>   # Optional
```

Conditions for LDAP user and password parameters:

- 1 – Can be optional (for group option).
- 2 – It must be mutually inclusive.
- 3 – If defined, it cannot be empty.

> ⚠ The parameter values may differ based on the directory service provider. For example: OpenLDAP or Microsoft Active Directory.

Integrating Identity with LDAP over TLS

The automation supports keystone integration with LDAP over TLS. To enable TLS, the CA root certificate must be presented as part of the **/root /openstack-configs/haproxy-ca.crt** file. The url parameter within the LDAP stanza must be set to *ldaps*.

url parameter supports the following format:

```
url: '<ldaps | ldap>://<FQDN | IP-Address>:[port]'
```

The protocol can be LDAP for non-SSL or LDAP if TLS is to be enabled.

The LDAP host can be a fully-qualified domain name (FQDN) or an IP address depending on how the SSL certificates are generated.

The port number is optional and if it is not provided it is assumed that the LDAP services are running on the default ports. For example: 389 for non-SSL and 636 for SSL. However, if these ports are not the default ports, then non-standard port numbers must be provided.

Support for Anonymous LDAP Bind

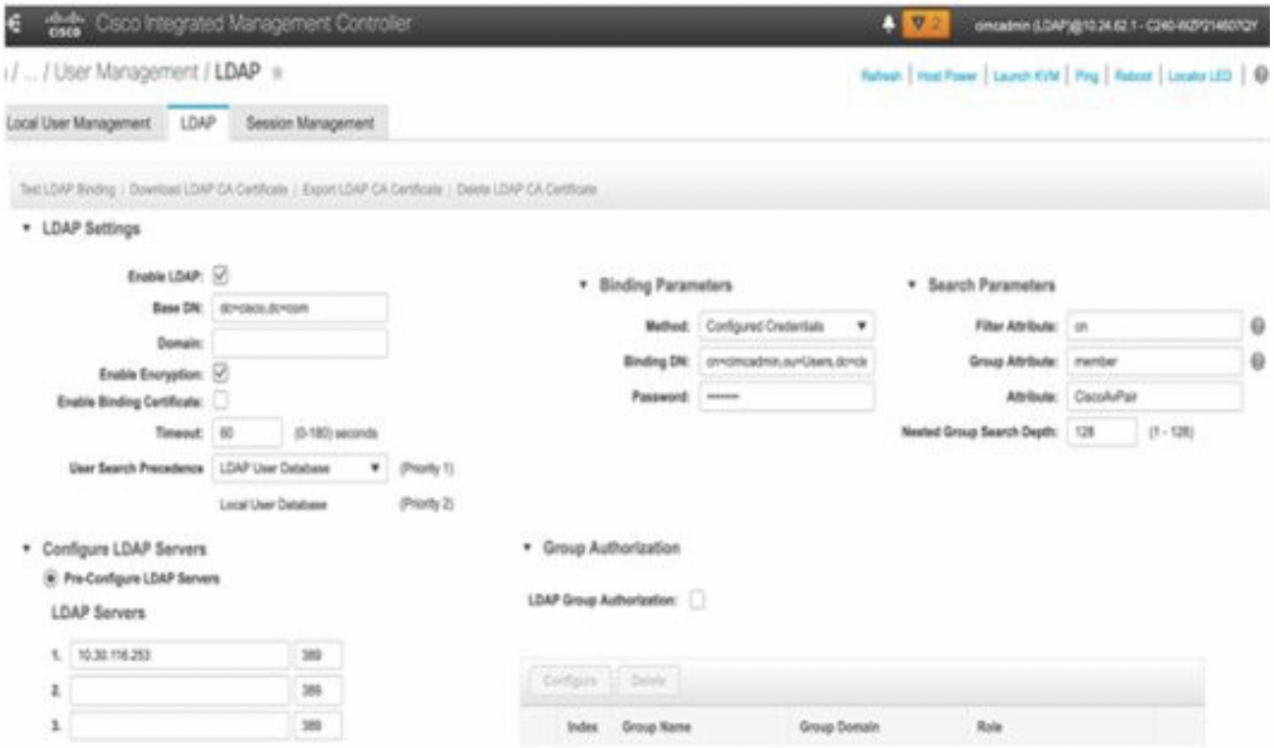
The automation provides support for anonymous simple bind where the LDAP configuration for a user representing the **bindDN** and **password** is optional and may not be provided.

> ⚠ Ensure that the LDAP server allows the clients to bind and search anonymously.

CIMC Authentication Using LDAP

Cisco VIM optionally supports the login of designated users into the CIMC using LDAP authentication. Enabling LDAP authentication for CIMC is a manual day-0 process and outside the scope of Cisco VIM automation. Once the LDAP authentication is setup, you must update the setup_data with the CIMC administration information that authenticates against LDAP, so that Cisco VIM works seamlessly.

The figure below is a snapshot of the CIMC configuration to authenticate using LDAP.



Baremetal Instances

Baremetal Instances

- Launching OpenStack Baremetal Instances
- Deploying Baremetal Instances

Launching OpenStack Baremetal Instances

You can use OpenStack Ironic service to launch baremetal instances once it is enabled through Cisco VIM.

1. Follow the pre-VIM deployment steps as given in the section Enabling Ironic Post Installation
2. Deploy Cisco VIM.
3. After the deployment, run the following commands:

```
# openstack baremetal node list --> Once Cisco VIM installation is completed, the ironic node is switched
to available state and is ready for use.
```

4. If you want to add more ironic nodes after VIM installation, execute the pre-VIM deployment steps and then add nodes. Use correct cimc credentials for $USER, $PASS, and $ADDRESS.

```
# openstack baremetal node create
--network-interface neutron --driver pxe_ipmitool
--name new-node-$ADDRESS \
--driver-info ipmi_username=$USER \
--driver-info ipmi_password=$PASS \
--driver-info ipmi_address=$ADDRESS
# openstack baremetal node manage $NODE_UUID
# openstack baremetal node inspect $NODE_UUID
```

5. Once inspection is completed, run the below commands:

```
openstack baremetal node show $NODE_UUID  to verify node's resources and capabilities set during
inspection process
openstack baremetal port list --> to verify ports. It should have one port of a baremetal node.
openstack baremetal node provide $NODE_UUID --> This PXE boots the deploy image via cleaning and sets
node to available
once done. Do this for all of the ironic nodes.
```

Deploying Baremetal Instances

Once the ironic nodes are available for use, you can launch an instance on the baremetal using the following steps:

1. Create the Nova baremetal flavor:

```
openstack flavor create --ram 1024 --disk 20 --vcpus 2 my-baremetal-flavor
openstack flavor set --property capabilities:boot_option="local" --property capabilities:disk_label="gpt"
my-baremetal-flavor
openstack flavor set --property resources:CUSTOM_BAREMETAL_DEFAULT=1 my-baremetal-flavor
openstack flavor set my-baremetal-flavor \
    --property resources:VCPU=0 \
    --property resources:MEMORY_MB=0 \
    --property resources:DISK_GB=0
```

   The trait name must match with the baremetal node. Here, it is CUSTOM_BAREMETAL.

```
openstack flavor set --property trait:CUSTOM_BAREMETAL=required my-baremetal-flavor
```

2. Add user images into glance. For example, if you are using Ubuntu image, use the following commands:

```
IMAGE_OS=ubuntu
MY_VMLINUZ_UUID=$(openstack image create --disk-format aki --container-format aki --public --file user-
image-${IMAGE_OS}.vmlinuz user-image-${IMAGE_OS}.vmlinuz | awk '/ id /{print $4}')

MY_INITRD_UUID=$(openstack image create --disk-format ari --container-format ari  --public --file user-
image-${IMAGE_OS}.initrd user-image-${IMAGE_OS}.initrd | awk '/ id /{print $4}')

openstack image create --disk-format qcow2  --public --container-format bare --property
kernel_id=$MY_VMLINUZ_UUID --property ramdisk_id=$MY_INITRD_UUID --file user-image-${IMAGE_OS}.qcow2 user-
image-${IMAGE_OS}
```

3. Use the following commands, to create the neutron tenant network to be used with the baremetal node:

```
# openstack network create my-tenant-net-name
# openstack subnet create --network my-tenant-net-name --subnet-range X.X.X.X/XX --ip-version 4
  my_tenant_subnet_name
```

4. Create a simple *cloud-init* script to log into the node after the instance is up:

   Example

```
#cloud-config
password: Lab1234!
chpasswd: { expire: False }
ssh_pwauth: True
```

   If this is a BypassNeutron mode testbed, add the following to the *cloud-init* script:
   Example

```
#cloud-config
password: Lab1234!
chpasswd: { expire: False }
ssh_pwauth: True
runcmd:
- ip link add link enp6s0 name enp6s0.3086 type vlan id <3086 - this will be the segment_id of your
tenant net>
- ip link set dev enp6s0.<3086> up
- dhclient enp6s0.<3086>
```

5. Boot the baremetal node using the below command:

```
# openstack server create --flavor <baremetal_flavor uuid> --image <centos/ubuntu image uuid> --nic
net-id=<tenant network uuid> --config-drive true --user-data user-data.txt <instance-name>
```

   For the creation of host aggregates, do not include controller servers as part of the aggregate.

   When ironic services are deployed, create a separate nova flavor for your VM instances and do not use the above referenced baremetal flavor. When launching a VM instance, use the VM flavor. Additionally, update your VM flavor to contain the following condition :

```
# openstack flavor set <vm-flavor> --property capabilities:hypervisor_type="s!= ironic"
```

VM Resizing and Migration

VM Resizing and Migration

- VM Resizing
- VM Migration
  - Cold Migrate
  - Live Migrate

VM Resizing

VM resizing is the process of changing the flavor of an existing VM. Using VM resize, you can upscale a VM based on your needs. The size of a VM is indicated by the flavor, based on which the VM is launched.

Resizing an instance means using a different flavor for the instance. Based on resources available, OpenStack creates the newly sized instance on the existing or new node, if more than one compute node exists. By default, the software allows you to change the RAM size, VDISK size, or VCPU count of an OpenStack instance using nova resize. The change of one or more properties of the target VM is allowed. If there is no suitable flavor for the new properties of the VM, you can create a new one.

```
VM resize with new flavor
# openstack server resize --flavor <new_flavor>  <vm_name>

Confirm server resize is complete
# openstack server resize confirm <vm_name>

restore server state before resize
# openstack server resize revert  <vm_name>
```

The resize process takes some time as the VM boots up with the new specifications. For example, deployment of a Cisco CSR (size in MB) takes approximately 60 mins. After the resize process, execute *nova resize confirm* command to overwrite the old VM image with the new one. You can revert to the old VM using the *revert* command, if required. At this point, you can access the VM through SSH and verify whether the correct image is configured.

Ensure that you plan for a downtime as the VM shutdown happens before resizing. Cisco recommends you to not resize a vdisk to a smaller value, as there is the risk of losing data.

VM Migration

Cold Migrate

Use the nova migrate or OpenStack server migrate command to move an instance from one compute host to another compute host. The scheduler chooses the destination compute host, based on the availability of the zone settings. The scheduler does not assume that the instance has shared storage available on the target host.

To initiate the cold migration of the VM, you can execute the following command:

```
#openstack server migrate <server>
```

The VM migration can take a while, as the VM boots up with the new specifications. After the VM migration, you can execute `openstack server resize confirm <server> --` to overwrite the old VM image with the new one. If you encounter an issue, use the openstack server resize revert `<server>` command to revert to the old VM image. At this point, access the VM through SSH and verify if the correct image is configured.

```
#openstack server resize confirm <server>
```

Ensure that you plan for a downtime as the VM is shut down by OpenStack before the migration.

Live Migrate

From Cisco VIM 4.0.0, NUMA aware live-migration is supported over both virtio and SRIOV. Live-migrating an instance means moving the virtual machine to a different OpenStack compute server while the instance is running. You can select the host to live migrate the instance. If the destination host is not selected, the nova scheduler chooses the destination compute based on the availability of the zone settings. You cannot use live-migration without shared storage except a booted from volume VM which does not have a local disk.

To initiate the live migration of the VM, you can execute the following command:

VM with boot-from-volume and no local disk

```
#openstack server migrate --live-migration <server>

Select the destination host
#openstack server migrate --live <host> <server>
```

Use the below command to migrate a VM if it has local disk and config-drive:

```
#openstack server migrate --live-migration <server> --block-migration

Select the destination host to migrate to
#openstack server migrate --live <host> <server> --block-migration
```

To check the virtual machine status, use the following command:

```
#openstack server show < server>
```

1. With NFV_HOST enabled, you must ensure that the vCPUs are available on the destination host to avoid collision. With cold migration, the vCPUs available on the destination host are automatically selected and assigned to the VM.
2. Few seconds of data plane disruption is expected during live migration.
3. If you are trying to live-migrate a VM with config drive, it is always considered as cold-migration.
4. Live migration is supported only for computes with same CPU architecture, same CPU supported function sets and same CPU options. Make sure the CIMC versions of the source and destinations are the same before attempting a Live migration.

Supported Integration

Supported Integration

- NetApp Integration
- Enabling SolidFire
- Enabling Zadara
- Enabling Auto-ToR via ACI API
- Enabling Cisco VIM with ACI Plugin
- Updating APIC Parameters
- Red Hat IDM
- Swift Integration

NetApp Integration

NetApp Integration

Cisco VIM supports the integration of NetApp devices running ONTAP 9.X or higher. NetApp devices are used as an alternative to Ceph for block storage. Cisco VIM is integrated and tested with FAS2650 SKU of NetApp. Ensure that block storage information is not present while using NetApp devices.

> ⚠ NetApp is the backend for Nova, Cinder, and Glance services.

The integration of NetApp with Cisco VIM is based on the following assumptions:

- To avoid UUID collision, one-one mapping between a Cisco VIM pod and Storage Virtual Machine (SVM) is required.
- The installation and management of NetApp is outside the scope of Cisco VIM.
- For tiered storage with NetApp, define multiple NFS mount paths.
- Set the *cinder_nfs_path* option in *setup_data,* with NFS mount paths separated by comma in string format.
- The NetApp endpoint(s) defined in the *setup_data* must be reachable from the management network of Cisco VIM.
- As the Cinder volume, Nova VMs, and Glance images come from NetApp storage, create three distinct NFS volumes in NetApp.
- Create and set the respective export policies so that it is available to all clients.
- The volumes for Nova, Cinder, and Glance, must be created with user and group id of 2012, 2007, and 2008, respectively.
- The size of the volumes must be appropriate as per the usage.

To enable NetApp, update the *setup_data.yaml* file with the following code prior to the installation:

```
STORE_BACKEND: netapp   # for glance
VOLUME_DRIVER: netapp   # for cinder

######################################################
New NETAPP config section; no dedicated Ceph Allowed
######################################################

NETAPP:
  server_hostname: <ip of netapp management/API server>
  server_port: <port of netapp management/API server> 80 for HTTP 443 for https
  transport_type: http/https
  username: <username of netapp API server>
  password: <password of netapp API server>
  vserver: <SVM for cinder nfs volume>
  cinder_nfs_server: <data path ip of nfs server>
  cinder_nfs_path: <"comma separated mount_paths"> Ex: "/shared_cinder1,/shared_cinder2,/shared_cinder3"
  nova_nfs_server: <data path ip of nova nfs server>
  nova_nfs_path: <mount path nova nfs>
  glance_nfs_server: <data path of glance nfs server>
  glance_nfs_path: <mount path of glance nfs>
  netapp_cert_file: <root ca path for netapp cluster only if protocol is https>
```

For details on how to secure NetApp transport on Day 2, see NetApp from http to https.

Enabling SolidFire

Enabling SolidFire

Cisco VIM supports the automated integration with a customer-managed SolidFire cluster for a block-storage option. SolidFire supports Cinder service for backup of block-storage. The pre-deployed SolidFire cluster has two HA networks such as management network and storage network. The management network is on 1G interface with active/passive configuration for two ports, while the storage network is on 10G interface with active/active Link Aggregation Control Protocol (LACP) configuration.
It is recommended that the:

- Storage network of Cisco VIM is same as that of SolidFire.
- Management network of Solidfire is reachable from Cisco VIM control nodes.

SolidFire is available only as a Day 0 configuration.

To enable SolidFire, update the *setup_data.yaml* file with the following code prior to the installation.

```
SOLIDFIRE:
  cluster_mvip: <management IP of SolidFire cluster> # must be reachable from the controller nodes
  cluster_svip: <storage VIP on SolidFire cluster to be used by Cisco VIM> # must be in Cisco VIM storage
/management network; recommended to have it in storage network for
 better performance
  admin_username: <admin user on SolidFire cluster to be used by Cisco VIM>
  admin_password: <password> # password criteria:
                             satisfy at least 3 of the following conditions:
                             - at least 1 letter between a to z
                             - at least 1 letter between A to Z
                             - at least 1 number between 0 to 9
                             - at least 1 character from !$#@%^-_+=
                             - AND password length is between 8 and 20 characters
```

Enabling Zadara

Using Zadara

- Overview
- Prerequisites
- Enabling Zadara

Overview

The Zadara Virtual Private Storage Array (VPSA) is a software-defined solution that is available as a Storage-as-a-Service with the storage servers residing in the customer premise and is also Layer 2 adjacent to the cloud. It is an elastic system that provides Enterprise-grade data protection and data management storage services.

Following are the key attributes:

- Enterprise quality, resilient, highly available, and consistent performance storage for the most demanding data center application workloads.
- Consumed as a Service - flexible, dynamic and billable.
- Scale out - to hundreds of storage nodes, thousands of drives, and multi-petabyte storage.
- True multi-tenancy - End-user controlled privacy and security. Separate workloads, resource allocation, and management per tenant such that each tenant truly experiences secure storage with no noisy neighbor.
- Universal storage - Supports all data services on one common infrastructure: Block, File, Object.

You can select Zadara as a backend for Cinder (block storage) and Glance (NFS), instead of Ceph. The use of Zadara helps in handling enterprise workloads, and provides a common infrastructure for all three storage types: Block, File, and Object.

Enabling Zadara in Cisco VIM is a Day 0 fresh Installation activity. In this case, the Cisco VIM cloud is a full-on cloud with Zadara as the backend for Glance and Cinder.

Prerequisites

- Zadara installation must be done ahead of time and is not the scope of Cisco VIM.
- Zadara pools for block storage and Glance images must be created ahead of time.
- For Glance image, a NAS volume must be pre-provisioned and a corresponding NFS export path must be provided in the *setup_data.yaml*
- The management network of Cisco VIM must reach the Zadara endpoint.
- The Fully Qualified Domain Name (FQDN) of the Virtual Private Storage Array (VPSA) *host_name* must be resolved to an IP address in the management network, but cannot be part of the management network pool.

Enabling Zadara

To enable Zadara in Cisco VIM:

1. Ensure that the block_storage section of Cisco VIM is absent.
2. Update the following in Cisco VIM setup_data:

```
ZADARA:
  access_key: <key to access vpsa end point in Zadara>
  vpsa_host: <fqdn of the vpsa host_name> # will resolve to an IP address in the management network;
cannot be part of the management network pool
  vpsa_poolname: <pool_name for vpsa>
  glance_nfs_name: <nfs name for glance>
  glance_nfs_path: <nfs path for glance>
```


Eliminates the need to configure a CA certificate in Cisco VIM, as the standard CA certificate bundle provided with the Linux distribution is sufficient and the matching certificate is pre-seeded to the Virtual Private Storage Array (VPSA).



Though Zadara needs to be enabled as a Day 0 option for security reason, you can change the Zadara *access_key* as a reconfigure option.

3. To update the *access_key,* the administrator must update the *setup_data.yaml* file with the commands listed below:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp/root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# # update the setup_data with right access_key in the Zadara section
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

After the reconfiguration, you can see that the Zadara is back in use with the new *access_key.*

Enabling Auto-ToR via ACI API

Enabling Auto-ToR via ACI API

- Overview
- Assumptions
- Additional Settings for Auto-ToR via ACI API on Day 0

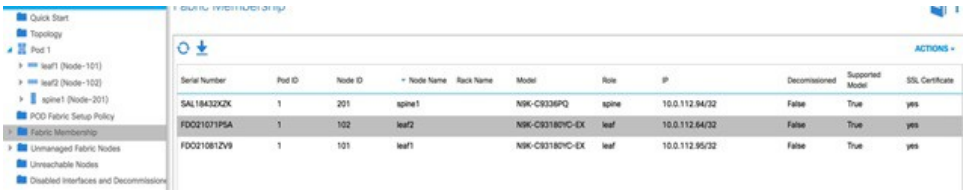
Overview

Cisco VIM integrates with ACI without the APIC plugin. Cisco VIM invokes the APIC APIs to pre-provision the right set of VLANs (along with the Day 0 aspects) on the corresponding server ports ahead of time, while supporting pod management operations.

Assumptions

As Cisco VIM does the Day 0 configuration of the ACI, following are the assumptions that Cisco VIM makes for the integration to happen.

1. Before the Cisco VIM installation, the APIC 3.2 controllers running in a cluster of three must be installed and active.
2. All spine and leaf switches are booted in ACI mode and discovered under Fabric Inventory. The number of leaf switches cannot be changed after the initial installation.
3. The IP address should be assigned to each device from the TEP_ADDRESS_POOL.



4. Network must be designed such that the management node and controllers are reachable to APIC controllers.
5. Tunnel end point address pool (TEP_ADDRESS_POOL) is set to ACI default at 10.0.0.0/16. Ensure that this address space is not assigned in the cloud.
6. Multicast address pool is set to ACI default at 225.0.0.0/15. Ensure that this address space is not assigned anywhere in the cloud.
7. TEP_ADDRESS_POOL and multicast address pool are immutable for the lifecycle of the infrastructure.

> ⚠ Using Auto-ToR provisioning using APIC API, the port PV (port*VLAN) count in a given ACI Fabric domain is under the scale limits 10000 PV/ToR and 450000 PV/Fabric.

Additional Settings for Auto-ToR via ACI API on Day 0

When using the option of ToR automation via ACI API on Day 0 without the APIC plugin, FABRIC_INTERFACE_POLICIES (under SERVER_COMMON section) and vim_apic_network section are required. The FABRIC_INTERFACE_POLICIES include global fabric policies that are defined in APIC and to be applied to the relevant switch ports. Listed below is the definition of the same:

```
SERVER_COMMON:
...
FABRIC_INTERFACE_POLICIES:
#When configure_fabric is False
global:    # Global policies can be overridden per server role sriov_tor_info: # <only 1 policy for sriov
interface>
- <accportgrp-...>    # string, E.g. policy name starting with accportgrp-. Must be pre-provisioned and
mandatory.


#When configure_fabric is True
global:    # Global policies can be overridden per server role tor_info:    # <list of global fabric policies
for control plane>
1.     -<str>    # string, E.g. hintfpol-25G-On-RS-FEC. Must be pre-provisioned.
2.     -<str>    # string E.g cdpIfP-CdpDisable
dp_tor_info:    # <list of policies for data plane for Intel NIC>
3.     -<str>    # string, E.g. lacplagp-LacpActive. Must be pre-provisioned.
4.     -<str>    # string E.g lldpIfP-LldpEnable sriov_tor_info: # <list of policies for sriov interfaces>
5.    -<str>    # string, E.g. hintfpol-25G-On-RS-FEC. Must be pre-provisioned. control:    # Optional in case
needs to over-ridden. Must be one of the SERVER roles.
```

```
tor_info:    # <list of policies for control plane>
6.    -<str>    # string, E.g. hintfpol-25G-On-RS-FEC. Must be pre-provisioned. compute:    # Optional in case
needs to over-ridden. Must be one of the SERVER
roles.
dp_tor_info:    # <list of policies for data plane>
7.    -<str>    # string, E.g. hintfpol-25G-On-RS-FEC. Must be pre-provisioned.

# Pre-provision EPG/BD policies to be configured for management and tenant/provider EPGs (FHS policy)
EPG_POLICIES: # Goes by Network Segments and is entirely Optional
management:    # Optional, list of policy for management segment
       - <path_to_epg_policy>    # Must be pre-provisioned. tn-cvim-installer-tenant/trustctrlpol-Trust-DHCP-Sv.
provider:    # Optional, list of policy for provider segment
1.    - <path_to_epg_policy1>    # Must be pre-provisioned.
2.    - <path_to_epg_policy2>    # Must be pre-provisioned.
tenant:    # Optional, list of policy for tenant segment
3.    - <path_to_epg_policy1>    # Must be pre-provisioned.
4.    - <path_to_epg_policy2>    # Must be pre-provisioned.
In the vim_apic_networks section, the provider and tenant VLAN definitions are listed as below:

vim_apic_networks:
EPG_NAME: 'VL-%s-EPG' # required; pattern substituted with vlan_id

BD_NAME: 'VL-%s-BD'    # required; pattern substituted with vlan_id PROVIDER:
# Support static vlans with the following attributes defined (these vlans will only be referred to bind and
unbind Static Ports)
- app_profile: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core-AP'
EPG_NAME: <str>    # <optional string. Will prefix the pattern in the global EPG_NAME definition>
mode: <trunk|access>    # string, default trunk
tenant: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core' vlan_ids: '<3550>'    # Can be a
only a single id
config_type: pre-provisioned vlan_pools:
- <str-1>    # string E.g. 'Server-VlanPool'

# The "vlan_ids' can be a VLAN Range only for L2 networks provided they belong #to the same VLAN pool and EPGs
map to the same Phydom, App Profile, VRF
- vlan_ids: '<3550>'    # <Can be a single id or range and/or comma separated list>
EPG_NAME: <str>    # <optional string. Will prefix the pattern in the global EPG_NAME definition>
BD_NAME: <str>    # <optional string. Will prefix the pattern in the global BD_NAME definition>
vlan_pools:    # List of vlan pool names. Must be pre-provisioned in ACI POD
1.    <str-1>    # string E.g. 'Server-VlanPool'
2.    <str-2>    # string E.g. 'Ext-VlanPool'
phys_dom: <str>    # string. Must be pre-provisioned in ACI POD. E.g. Server-PhysDom
description: <str>    # optional; string. Must be pre-provisioned in ACI POD. E.g. 'provider net 3550'
tenant: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core' app_profile: <str>    # string, E.g.
Must be pre-provisioned in ACI POD. 'Core-AP' vrf: <str>    # string, E.g. Must be pre-provisioned in ACI POD.
'Core' subnets:    # List of subnets to be configured for the Bridge Domain
3.    scope: <str>    # string. Can be <'private'|'public'|'private,shared'>
4.    gateway_cidr:    # IPv4 or IPv6 network gateway with cidr E.g.
'240b:c010:101:2839::ffff/64'
      ctrl:  <no-default-gateway" or "nd" or "nd,no-default-gateway" or "no-default-gateway,nd" or "unspecified"
> # when gateway cidr is of type IPv6
        or
  ctrl:  <no-default-gateway" or "querier" or "querier,no-default-gateway" or "no-default-gateway,querier" or
"unspecified"> # when gateway cidr is of type IPv4

l3-out:    # optional, List of L3out External Routed Network Instances. Must be pre-provisioned
    -<External Routed Network/Instance Profile>    # E.g. Core-Ba-Ma-L3out/Core-Ba-Ma-ExtEPG
      -<External Routed Network/Instance Profile>    # E.g. ce1-epc-CP-L3out/ce1-epc-CP-ExtEPG mode:
trunk|access    # string, default trunk
       l2_unknown_unicast: <flood or proxy>
       limit_ip_learning: <true or false>
       preferred_group_member: <include or exclude>    # Optional, default is exclude
       arp_flood: <true or false>
       unicast_routing: <true or false>
       nd_policy: <true or false>      # When true, ensure that path/ndifpol is defined under SERVER_COMMON ->
EPG_POLICIES -> provider section

TENANT:
# Does not contain l3out
# Can be a VLAN Range only for L2 networks provided they belong to the # same VLAN pool and EPGs map to the
same Phydom, App Profile, VRF
```

```
- vlan_ids: '<2251:2260,2280,2290>'
EPG_NAME: <str>    # <optional string. Will prefix the pattern in the global EPG_NAME definition>
BD_NAME: <str>    # <optional string. Will prefix the pattern in the global BD_NAME definition>
vlan_pools:
5.    'Server-VlanPool' phys_dom: Server-PhysDom tenant: 'Core'
app_profile: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core-AP' vrf: Nokia-LI
mode: trunk
subnets:    # May contain a subnet in which case only valid value is a single vlan id
1.    scope: <str>    # string. Can be <'private'|'public'|'private,shared'> gateway_cidr:    # IPv4 or IPv6
network gateway with cidr E.g.
'240b:c010:101:2839::ffff/64'

    l2_unknown_unicast: <flood or proxy>
    limit_ip_learning: <true or false>
    preferred_group_member: <include or exclude>    # Optional, default is exclude
    arp_flood: <true or false>
    unicast_routing: <true or false>
    nd_policy: <true or false>  # When true, ensure that path/ndifpol is defined under SERVER_COMMON ->
EPG_POLICIES -> tenant section
```

In the vim_apic_networks section, the provider and tenant VLAN definitions are listed as below:

```
vim_apic_networks:
EPG_NAME: 'VL-%s-EPG' # required; pattern substituted with vlan_id

BD_NAME: 'VL-%s-BD'    # required; pattern substituted with vlan_id PROVIDER:
# Support static vlans with the following attributes defined (these vlans will only be referred to bind and
unbind Static Ports)
- app_profile: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core-AP'
EPG_NAME: <str>    # <optional string. Will prefix the pattern in the global EPG_NAME definition>
mode: <trunk|access>    # string, default trunk
tenant: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core' vlan_ids: '<3550>'    # Can be a
only a single id
config_type: pre-provisioned vlan_pools:
- <str-1>    # string E.g. 'Server-VlanPool'

# The "vlan_ids' can be a VLAN Range only for L2 networks provided they belong #to the same VLAN pool and EPGs
map to the same Phydom, App Profile, VRF
- vlan_ids: '<3550>'    # <Can be a single id or range and/or comma separated list>
EPG_NAME: <str>    # <optional string. Will prefix the pattern in the global EPG_NAME definition>
BD_NAME: <str>    # <optional string. Will prefix the pattern in the global BD_NAME definition>
vlan_pools:    # List of vlan pool names. Must be pre-provisioned in ACI POD
1.    <str-1>    # string E.g. 'Server-VlanPool'
2.    <str-2>    # string E.g. 'Ext-VlanPool'
phys_dom: <str>    # string. Must be pre-provisioned in ACI POD. E.g. Server-PhysDom
description: <str>    # optional; string. Must be pre-provisioned in ACI POD. E.g. 'provider net 3550'
tenant: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core' app_profile: <str>    # string, E.g.
Must be pre-provisioned in ACI POD. 'Core-AP' vrf: <str>    # string, E.g. Must be pre-provisioned in ACI POD.
'Core' subnets:    # List of subnets to be configured for the Bridge Domain
3.    scope: <str>    # string. Can be <'private'|'public'|'private,shared'>
4.    gateway_cidr:    # IPv4 or IPv6 network gateway with cidr E.g.
'240b:c010:101:2839::ffff/64'
      ctrl:  <no-default-gateway" or "nd" or "nd,no-default-gateway" or "no-default-gateway,nd" or "unspecified"
> # when gateway cidr is of type IPv6
        or
  ctrl:  <no-default-gateway" or "querier" or "querier,no-default-gateway" or "no-default-gateway,querier" or
"unspecified"> # when gateway cidr is of type IPv4

l3-out:    # optional, List of L3out External Routed Network Instances. Must be pre-provisioned
    -<External Routed Network/Instance Profile>    # E.g. Core-Ba-Ma-L3out/Core-Ba-Ma-ExtEPG
      -<External Routed Network/Instance Profile>    # E.g. ce1-epc-CP-L3out/ce1-epc-CP-ExtEPG mode:
trunk|access    # string, default trunk
       l2_unknown_unicast: <flood or proxy>
       limit_ip_learning: <true or false>
       preferred_group_member: <include or exclude>    # Optional, default is exclude
       arp_flood: <true or false>
       unicast_routing: <true or false>
       nd_policy: <true or false>      # When true, ensure that path/ndifpol is defined under SERVER_COMMON ->
EPG_POLICIES -> provider section
```

```
TENANT:
# Does not contain l3out
# Can be a VLAN Range only for L2 networks provided they belong to the # same VLAN pool and EPGs map to the
same Phydom, App Profile, VRF
- vlan_ids: '<2251:2260,2280,2290>'
EPG_NAME: <str>    # <optional string. Will prefix the pattern in the global EPG_NAME definition>
BD_NAME: <str>    # <optional string. Will prefix the pattern in the global BD_NAME definition>
vlan_pools:
5.    'Server-VlanPool' phys_dom: Server-PhysDom tenant: 'Core'
app_profile: <str>    # string, E.g. Must be pre-provisioned in ACI POD. 'Core-AP' vrf: Nokia-LI
mode: trunk
subnets:    # May contain a subnet in which case only valid value is a single vlan id
1.    scope: <str>    # string. Can be <'private'|'public'|'private,shared'> gateway_cidr:    # IPv4 or IPv6
network gateway with cidr E.g.
'240b:c010:101:2839::ffff/64'

    l2_unknown_unicast: <flood or proxy>
    limit_ip_learning: <true or false>
    preferred_group_member: <include or exclude>    # Optional, default is exclude
    arp_flood: <true or false>
    unicast_routing: <true or false>
    nd_policy: <true or false>  # When true, ensure that path/ndifpol is defined under SERVER_COMMON ->
EPG_POLICIES -> tenant section
```


Ensure that you update right VLAN information when using this option in context of APIC, during reconfiguration of VLANs.

Enabling Cisco VIM with ACI Plugin

Enabling Cisco VIM with ACI Plugin

Cisco VIM integrates with ACI in two fashion: with and without the APIC plugin. When Cisco VIM integrates with the Opflex ML2 plugin (in Unified mode), the tenant and provider VLANs are managed dynamically as the VMs come and go in the cloud. Cisco VIM orchestrator configures the Day 0 aspects of the ACI fabric along with Opflex ML2 plugin integration, but with an exception of the manual configuration of L3 out.

For the following pods, the management operations are supported via the plugin:

- UCS M5: 40G VIC (samx/pet) / NIC (SRIOV) Fullon pod
- UCS M4/M5: 10G Intel NIC Fullon/Micro pod

Before you begin

As Cisco VIM does the Day 0 configuration of the ACI, following are the assumptions that Cisco VIM makes for the integration to happen. You need to pay attention and take ownership of any exceptions to the assumptions.

- Before the Cisco VIM installation, the APIC controllers must be running in a cluster of one, three or five with a minimum version of 5.0.
- All spine and leaf switches are booted in ACI mode and discovered under fabric inventory.
- Cisco VIM assumes that the ToR switches which are part of the Cisco VIM pod are dedicated, to ensure that there is no conflicting configuration pushed outside of Cisco VIM. If the created vPC domains exist, ensure that the domain ID matches with the information provided in the *setup_data* on a per switch pair basis.
- OpFlex authentication is disabled.
- Network must be designed such that the management node and controllers (via the management network) are reachable to the IP of the APIC controllers.
- For ACI policy management, a networking segment ACIINFRA is provided such that it does not overlap with the VLANID across the infrastructure. This corresponds to ACI infrastructure VLAN defined during fabric setup.
- Tunnel endpoint address pool (TEP_ADDRESS_POOL) is set to ACI default at 10.0.0.0/16. Ensure that this address space is not assigned in the cloud.
- Multicast address pool is set to ACI default at 225.0.0.0/15. Ensure that this address space is not assigned anywhere in the cloud.
- ACIINFRA VLANID, TEP_ADDRESS_POOL, and multicast address pool are immutable for the lifecycle of the infrastructure.
- Pre-provision of Layer 3 out API network must be done before the installation of Cisco VIM and listed below:
    - Create installer tenant and VRF and provide their name in setup_data.
    - Create L3out routed outside object and provide its name in the setup_data.
    - Ensure that this api-l3out must be associated to the tenant VRF.
- VAULT and NFVBench features are not supported with the ACI plugin option.
- If VMTP is used, the Layer 3 Out that matches with the value of the *net_name* defined in the VMTP section, must exists in the tenant *common* so that it can be used across all the user tenants.

- The L3-out object for OpenStack API network needs to be named consistently, that is, the name of the L3 Outside object must be same as the name provided for its corresponding external network instance profile. For example, if you provide *api_l3out_network* as your api-l3out in the *setup_data,* your distinguished name (dn) of the api network must be *cvim-installer-tenant|uni/tn-cvim-installer-tenant/out-api-l3out/instP-api-l3out*

- For SRIOV networks, since the physical networks are tied to isolated sriov nic interfaces, the VMs attached to sriov ports on provider networks cannot have their IP addresses on the primary interface acquired via the neutron DHCP agent. Hence, it is important to create the SRIOV subnets with the *–no-dhcp* option.

For example, openstack subnet creates *--gateway <> --allocation-pool start=<>,end=<> --network sriov --subnet-range <> --no-dhcp sriov-subnet*

- Network configurations for such VMs are expected to be customized using *config-drive* options. This is true for non-ACI plugin integrated deployments as well, when it comes to SRIOV networks.

1. To run with the plugin, update the APICINFO section of *setup_data*:

```
APICINFO:
  apic_hosts: '<ip1|host1>:[port], <ip2|host2>:[port], <ip3|host3>:[port]'   # max of 5, min of 1;
reconfigurable
  apic_username: <admin_username>        # common across the APIC controllers
  apic_password: <password>              # common across the APIC controllers
  apic_system_id: <string_max_length_8>  # string max length of 8
  apic_resource_prefix: <string>         # e.g. cvim-1; max length of 6, helps identify CVIM created
resources
  apic_tep_address_pool: 10.0.0.0/16     # static today and optional
  multicast_address_pool: 225.0.0.0/15   # static, today and optional
  apic_pod_id: <int>                     # All(int, Range(min=1, max=65535)),
  apic_installer_tenant:  <string>       # String, max length of 32
  apic_installer_vrf: <string>           # max length of 32, this is the VRF which is associated with
the pre-provisioned API L3out
  api_l3out_network:  <string>           # max length 32

NOTE: mgmt_l3out_network and mgmt_l3out_vrf MUST coexist together if defined   # Required to have
external connectivity to management network
# mgmt_l3out_network: <string>           # max length 32 (optional)
# mgmt_l3out_vrf:     <string>           # max length 32 (optional)
```

2. Define the mechanism driver and a new network segment called ACIINFRA in the *setup_data* as given below:

```
.
NETWORKING:
  .
  .
  networks:
  - segments: [aciinfra]
    vlan_id: <vlan_id> # This vlan should not overlap with any of the vlans defined in setup data; Needs
to be provided by ACI admin.
.
.
MECHANISM_DRIVERS: aci

TENANT_NETWORK_TYPES: VLAN
TENANT_VLAN_RANGES: <vlan_id1>:<vlan_id2>
PROVIDER_VLAN_RANGES: <vlan_id3>:<vlan_id4>
```

3. As the APIC manages the Leaf switches, it is mandatory to define the Leaf switch pairs in the following format:

```
TORSWITCHINFO: (mandatory)
  SWITCHDETAILS:
    -
      hostname: <leaf-hostname-1>
      vpc_peer_keepalive: <leaf-hostname-2>
      vpc_domain: 1                       # Must be unique across switch pairs
      br_mgmt_port_info: <switch port from which management node hangs off> # br_mgmt_* attributes must
exist on at least one pair, e.g. 'eth1/27'
      br_mgmt_vlan_info: <vlan_id>         # Management network vlan_id
      node_id: <int>                       # unique across switches

    -
      hostname: <leaf-hostname-2>
      vpc_peer_keepalive: <leaf-hostname-1>
      vpc_domain: 1                       # Must be unique across switch pairs
      br_mgmt_port_info: <switch port from which management node hangs off> # br_mgmt_* attributes must
exist on at least one pair, e.g. 'eth1/27'
      br_mgmt_vlan_info: <vlan_id>         # Management network vlan_id
      node_id: <int>                       # unique across switches

    -
      hostname: <leaf-hostname-3>
      vpc_peer_keepalive: <leaf-hostname-4>
      vpc_domain: 2                       # Must be unique across switch pairs
      node_id: <int>                      # unique across switches

    -
      hostname: <leaf-hostname-4>
      vpc_peer_keepalive: <leaf-hostname-3>
      vpc_domain: 2                       # Must be unique across switch pairs
      node_id: <int>                      # unique across switches
```

4. Since Cisco VIM orchestrator does the Day 0 configuration of the ACI, the SERVERS section of the *setup_data* must be augmented to include the server, switch port associations, *sriov_tor_info* and *dp_tor_info* (for intel NIC testbed):

```
SERVERS:

  <controller-hostname-1>:
    cimc_info: {cimc_ip: <cimc_ip_address>}
    rack_info: {rack_id: <rack_info>}
    tor_info: {<leaf-hostname-1>: ethx/y, <leaf-hostname-2>: ethx/y}
    dp_tor_info: {<leaf-hostname-1>: ethi/j, <leaf-hostname-2>: ethi/j}          # Only for Intel NIC
testbed
   :
   :

  <compute-hostname-1>:
    cimc_info: {cimc_ip: <cimc_ip_address>}
    rack_info: {rack_id: <rack_info>}
    tor_info: {<leaf-hostname-1>: etha/b, <leaf-hostname-2>: etha/b}
    dp_tor_info: {<leaf-hostname-1>: ethe/f, <leaf-hostname-2>: ethe/f}          # Only for Intel NIC
testbed
    sriov_tor_info: {<leaf-hostname-1>: ethg/h, <leaf-hostname-2>: ethg/h}
   :
   :
  <storage-hostname-1>:
    cimc_info: {cimc_ip: <cimc_ip_address>}
    rack_info: {rack_id: <rack_info>}
    tor_info: {<leaf-hostname-1>: ethc/d, <leaf-hostname-2>: ethc/d}
```

5. To enable the plugin in a VIC/NIC setup, where the SRIOV is running on the Intel NIC, add the following option:

```
CISCO_VIC_INTEL_SRIOV: True
INTEL_SRIOV_PHYS_PORTS: 4
INTEL_SRIOV_VFS: <int>       # Max of 32
```

6. To enable the plugin in a Intel NIC setup with 2-X710, where the SRIOV is running on two of the ports (C and D) of the Intel NIC, add the following option:

```
INTEL_NIC_SUPPORT: True
INTEL_SRIOV_PHYS_PORTS: 4
INTEL_SRIOV_VFS: <int>       # Max of 32
```

7. Post installation, you can update the parameters associated with APIC. For more information, see Updating APIC Parameters. For VLAN reconfiguration using ACI plugin, see Managing Provider/Tenant VLAN Ranges

Updating APIC Parameters

Updating Cisco Application Policy Infrastructure Controller (APIC) Parameters

- Updating APIC Hosts, Username, Password
- Updating Switch managed by APIC

Updating APIC Hosts, Username, Password

With the ACI integration on Day 0, Cisco VIM supports the replacement of the APIC controller IP, username, and/or password of the APIC cluster. To initiate this change, update the relevant APIC attributes in *setup_data* as listed below:

```
APICINFO:
  apic_hosts: '<ip1|host1>:[port], <ip2|host2>:[port], <ip3|host3>:[port]' # max of 5, min of 1; reconfigurable
  apic_username: <apic_username>
  apic_password: <apic_password>
```

To update APIC configuration on an existing pod, copy the *setup_data* into a local directory and update it manually with the relevant apic_hosts IP, username, or password, and then run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include new APIC info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Updating Switch managed by APIC

With the ACI integration on Day 0, Cisco VIM supports the expansion of the leaf switches to increase the fabric in pairs. To initiate this change, update the relevant APIC attributes in *setup_data* as listed below:

```
Since the APIC manages the leaf switches, it is mandatory to define the new leaf switches (in pairs)
in the following format:
TORSWITCHINFO: (mandatory)
  SWITCHDETAILS:
  :         # Do not touch the existing ToR Pairs
  :
  :
  -
    hostname: <leaf-hostname-new1>
    vpc_peer_keepalive: <leaf-hostname-new2>
    vpc_domain: <int> # Must be unique across pairs
    node_id: <int> # unique across switches
  -
    hostname: <leaf-hostname-new2>
    vpc_peer_keepalive: <leaf-hostname-new1>
    vpc_domain: 1
    node_id: <int> # unique across switches
```

To add new ToR pairs in the TORSWITCH information section of an existing pod installed with ACI via Cisco VIM, copy the *setup_data* into a local directory, update it manually with the relevant SWITCHDETAILS in the TORSWITCHINFO section, and then run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include new SWITCHDETAILS in pair)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Red Hat IDM

Red Hat Identity Management System

- Overview
- Assumptions
- Enabling IDM with reachability over br_api and br_mgmt
- Enabling IDM with reachability over br_mgmt only
- Enabling IDM on Day 2
- Enabling/Disabling IPA Credential Caching

Overview

Cisco VIM supports integration with RedHat Identity Management System (IDM) for Identity, Policy, Audit (IPA). The IPA solution helps in managing users and hosts and applying group-based policies using a centrally managed identity and authentication solution for Linux hosts. You can enable this feature as a Day 0 or Day 1 activity. Multiple IPA servers with IPv4 and IPv6 endpoints are allowed for redundancy. When you enable IPA, you cannot enable the features of TTY Logging and *vim_ldap_admins* as listed in Managing VIM Administrators. In the same vein, IDM and the features of Password Management and SSH_ACCESS_OPTION listed in Securing Management Node are mutually exclusive.

For more information on IDM, see https://www.redhat.com/archives/rh-community-de-berlin/2012-November/pdfOlwXB8dm7U.pdf.

Assumptions

- IDM integration includes all hosts in a pod: management and cloud hosts in the cluster for any pod type.
- All operations are available only for an IPA client on the IPA client host.
- Any host management against the IPA server outside of enrollment and unenrollment from the IPA client is out of the scope of the automation.
- IPA client does not manage any users.
- Local root user access must not change.
- IPA servers can access all hosts through the management network.
- If the IPA server does not reside on the management network, the management node must access it via *br_api*
- The certificates between the IPA server and hosts over the management network are not included.
- Hostname of all the servers (including the management node) in the pod are in FQDN with lower cases (RFC4120) only.
- The servers' hostname including the management node must belong to the *ipa_domain_name* listed in the IPA_INFO section.

Integration of IPA for pods that use short hostnames (non-FQDN) is challenging when it is enabled as part of a reconfiguration operation. During the IPA registration of host clients, the IPA server updates the system hostname of the client to a hostname with FQDN. This breaks Cisco VIM operations such as cluster recovery where the hostnames defined in the *setup_data.yaml* file is used. Also, breakage exists in SRIOV mappings, and so on. Hence, it is recommended that the hostname of all servers in the pod, including the management node are in FQDN with only lower cases.

Enabling IDM with reachability over br_api and br_mgmt

To enable IDM, complete the following steps:

Update the *setup_data.yaml* file with the following commands and execute *ciscovim run:*

```
IPA_INFO:
  ipa_servers:
        - hostname: <fqdn_hostname in lower case belonging to ipa_domain_name>
          ipaddresses: # --- Optional (if defined minimally has to have an IPv4 or IPv6 address)
            - '<ipv4_address>'
            - '<ipv6_address>'
        - hostname: <fqdn hostname in lower case belonging to ipa_domain_name>
          ipaddresses: --- Optional (if defined minimally has to have an IPv4 or IPv6 address)
            - '<ipv4_address>'
            - '<ipv6_address>'
  enroller_user: <enroller_username>
  enroller_password: <enroller_password>
  ipa_domain_name: <ipa_domain_name>
```

Enabling IDM with reachability over br_mgmt only

In Cisco VIM, ensure that *br_api* and *br_mgmt* are on different VRFs and OAM or fabric devices. In Cisco VIM management node, the default route is *br_api.* By default, the Infrastructure services like NTP, DNS, CIMC, and IPA are reachable via *br_api* from the management node, unless they have an IP that belongs to the management network.

In some cases, the access of Infrastructure endpoints such as IDM is available only via *br_mgmt.* To maintain backwards compatibility and bring in additional flexibility in Cisco VIM 4.2.0, you can assign targeted static route for targeted *infra* networks to go over *br_mgmt.* From Cisco VIM 4.2.0, the service of IPA is available with this option. On top of the IDM assumptions listed above, additional assumptions need to hold to support IDM reachability over *br_mgmt.*

Assumptions to Enable IDM with reachability over br_mgmt only

- All hosts and the management node are reachable via *br_mgmt*
- The *setup_data* is augmented with a list static routes that are reconfigurable on Day 2
- The IPA section in the *setup_data* contains *inband* option.
- None of the static route entries in new option of STATIC_ROUTE_VIA_BR_MGMT are overlapping with each other or other existing Cisco VIM networks.

Update the *setup_data.yaml* file with the following command and execute *ciscovim run:*

```
STATIC_ROUTE_VIA_BR_MGMT: <list of v4 or v6network with mask that route via br_mgmt>. # for example, [3456:2345:
ac34::/64', 2.3.4.0/24]

IPA_INFO:
  ipa_servers:
        - hostname: <fqdn_hostname in lower case belonging to ipa_domain_name>
          ipaddresses: # --- Optional (if defined minimally has to have an IPv4 or IPv6 address)
            - '<ipv4_address>'
            - '<ipv6_address>'
        - hostname: <fqdn hostname in lower case belonging to ipa_domain_name>
          ipaddresses: --- Optional (if defined minimally has to have an IPv4 or IPv6 address)
            - '<ipv4_address>'
            - '<ipv6_address>'
  enroller_user: <enroller_username>
  enroller_password: <enroller_password>
  ipa_domain_name: <ipa_domain_name>

  inband_ipa: <True or False>.  # Optional, reconfigurable
```

Enabling IDM on Day 2

To enable IDM on Day 2, see Platform Security.

Enabling/Disabling IPA Credential Caching

To control the cache_credentials setting for IPA, use the following command:

```
Example:

# For IPA, hostname for all servers including the management node must be in
# lower case FQDN (RFC4120) and belongs to ipa_domain_name
#
#IPA_INFO:
# ipa_servers: # list of ipa_servers
# - hostname: <hostname with fqdn belonging to the ipa_domain_name>
# ipaddresses: # ---------> Optional
# - <ipv4_address or ipv6_address>
# - hostname: <hostname with fqdn belonging to the ipa_domain_name>
# ipaddresses: # ---------> Optional
# - <ipv4_address>
# - <ipv6_address>
# enroller_password: <admin_password>
# enroller_user: <admin_username>
# ipa_domain_name: <ipa_domain_name>
# inband_ipa: <True or False> # optional, default is false; can be true when STATIC_ROUTE_VIA_BR_MGMT is defined

 ipa_cache_credentials: <True or False> # optional, default is true; enable/disable credential caching
```

By default, it is enabled for 24 hours to cache the credentials before its expiry. If disabled, the credentials are not cached.

Swift Integration

Swift Integration with CVIM

- Overview
- Prerequisites
- Assumptions
- Enabling Swift
- Potential Pitfalls

Overview

Object Storage (swift) is a HTTP accessible storage service for managing large amount of data, such as video, images, files, VM images, and unstructured data, as Binary Large OBbjects (BLOBs). The BLOBs is a collection of binaries on the underlying file system along with the metadata. The Object Storage distributed architecture supports both horizontal scaling and redundancy with software-based data replication.

Prerequisites

For integration with Cisco VIM, the swift services must be deployed on controller nodes which have access to Storage cluster's storage Virtual IP (sVIP), so that the controller nodes act as iSCSI (Internet Small Computer System Interface) initiators with cluster storage VIP as iSCSI target. The following ports are opened by Cisco VIM orchestrator for swift service interaction:

Interface Name	Direction from Controller POV	Protocol	Port number	Service Name	Port Location
br_api	incoming	TCP	6000	Object server	Deployed on controller node
br_api	incoming	TCP	6001	Account server	Deployed on controller node
br_api	incoming	TCP	6001	Container server	Deployed on controller node
br_api	bi-directional	TCP	10873	RSYNC	Deployed on controller node
br_api	incoming	TCP /HTTP	8080	Proxy server	Deployed on controller node
storage network	outgoing	TCP	3260	Storage VIP	Access to Storage cluster from controller node
storage network	outgoing	TCP	3260	Storage nodes	Access to each storage node, which is part of storage cluster, from controller node

Assumptions

- As swift is an optional service, you must add *swift* value in optional service list and swift related configuration in the setup data configuration file to enable swift.
- iSCSI cluster (SolidFire) is setup and its storage VIP (svip) is accessible from controller nodes (iSCSI TCP port 3260).
- Ensure that the volumes are pre-created in SolidFire and the authentication information for volume access is captured. This is a pre-requisite for swift setup and deployment.
- During Cisco VIM swift deployment, the volumes are discovered and mounted with swift. Also, the data can be accessed only via iSCSI (no fibre channel or other protocol support).
- On the storage cluster server side, you must create an account to represent a volume owner and to receive its own set of Challenge Handshake Authentication Protocol (CHAP) credentials. Use the same account while creating swift specific volumes to share the CHAP credentials.

  > ⚠ Ensure that you create account first before creating the volumes and then use that account for authentication access.

- Create new volumes with a distinct name, so that their unique name can be used as volume identifier in swift setup data. For example, volume names can be given as *vol-controller-1-1a*, *vol-controller-2-1a*, *vol-controller-3-1a* which indicates that volume with name *vol-controller-1-1a* is meant for controller node 1, volume with name *vol-controller-2-1a* is meant for controller node 2 and so on. For swift usage, create a separate dedicated volume for each controller node in your deployment. One volume per controller node is sufficient. If needed, you can create more than one volume per controller node.

  > ⚠ Ensure that volume names are unique and are not part (substring) of other volume names. For example, *vol-controller-1-1* is part of name *vol-controller-1-10. So* the volume name, *vol-controller-1-10* is not valid and cannot be used if volume name *vol-controller-1-1* is a lready used.

- The iscsiadm discovery command always have volume name as part of discovered iSCSI target names in SolidFire environment.
- For Cisco VIM**,** only CHAP based authentication mechanism is supported. Volume access group based authentication is not supported for fresh installation.

- Volumes available from iSCSI cluster are used as swift backstore.
- As number of docker containers are deployed on controller nodes, only Full pod types are supported for swift Integration.
- The storage network for Cisco VIM is same as that for SolidFire.
- The management network for SolidFire is reachable from Cisco VIM controller nodes.
- Swift service is not integrated with any other OpenStack services such as Cinder for backups, Glance, and so on.
- Read-only role support is provided via Swift ACL. For more details, see Swift ACLs
- Do the following only during installation/reconfiguration:
    - Use the account name and respective CHAP credentials to access volumes defined in server, and check for the correctness of CHAP auth information in setup data.
    - Check whether the volume identifiers provided in setup data are specific to each controller node, and ensure those volumes are created using CHAP initiator_username.

Enabling Swift

To enable swift as Day 0 or Day n option via reconfigure (see General Guidelines), add the following in the *setup_data* file:

SWIFT_SETUP with same CHAP Auth across volumes

```
SERVERS:
  control-server-1:
        ....
  control-server-2:
        ....
  control-server-3:
        ....

OPTIONAL_SERVICE_LIST:
  - swift                                    # list swift as a service
  - ....

###############################################################################
# OpenStack Swift (Object-Store) setting: Optional
###############################################################################
SWIFT_SETUP:
  STORAGE_BACKSTORE: iscsi_cluster # Required Input.
  server_role: control   #Mechanism to identify servers where swift component must be installed. Value must be
a valid SERVER ROLE which maps to set of servers/hosts.
Use 'control'
  iscsi_cluster:
    cluster_svip:   <ipv4_address> # cluster storage VIP / target portal
    chap_auth:                        # specified at cluster level
      initiator_username: <username>   # cannot be modified later via reconfigure
      initiator_secret:  <password>      # can be modified later via reconfigure

 # For 2-way CHAP (initiator also validating target server)
      target_username: <username>   # cannot be modified later via reconfigure
      target_secret: <password>      # can be modified later via reconfigure

    volumes:  # server reference must match with key used to identify server under SERVERS section. Once added,
cannot be removed.You can add more volumes later if required.
     - server_reference: <hostname_controller_1>
        volume_identifiers:                                          # Is a list of iscsi volumes used
per node. Must be unique name to match only one related iscsi target during discovery
          - control-server-1-vol-1                    # naming convention: hostname_controller_1-vol-
<index_no>
     - server_reference: hostname_controller_2>
        volume_identifiers:
          - control-server-2-vol-1             # naming convention: hostname_controller_2-vol-<index_no>
     - server_reference: hostname_controller_3>
        volume_identifiers:
          - control-server-3-vol-1             # naming convention: hostname_controller_3-vol-<index_no>
```

Swift rings determine where the data must reside in the cluster. The ring with the container name cvim_swift_rings is created by Cisco VIM as part of swift on-boarding, which can be accessed and viewed via Horizon and Swift APIs.

You must treat these artifacts as part of Cisco VIM system level resources, and must not modify or delete *cvim_swift_rings* container.

Potential Pitfalls

Swift Service Recovery

When swift service cloud sanity check fails, swift service recovery can be done as follows:

Swift ISCSI error Recovery

```
cd <installer_dir>/tools/
./swift_tool.py --action recover
```

ISCSI Target Discovery Error

During installation and/or reconfiguration of swift, an error may arise if ISCSI target discovery fails. Listed below is the snippet of the failure logs:

Target discovery failure from ciscovim logs

```
zgrep -a  "Error: Unable to discover ISCSI Target with CLUSTER SVIP" /var/log/mercury/a2ce2fee-d085-4f4e-be89-
00ae49b87dcf/*.gz

# Sample error output
2020-08-19 02:17:27,487 p=32187 u=root |  fatal: [x.x.x.x]: FAILED! => {"changed": true, "cmd": ["bash", "/tmp
/tmp.Aab1rIw6JH/manage_iscsi_devices.sh", "-a", "setup", "-s", "x.x.x.x", "-f", "control-server-1-vol-1 control-
server-1-vol-2"], "delta": "0:00:00.046186", "end": "2020-08-19 02:17:27.364337", "failed": true, "rc": 1,
"start": "2020-08-19 02:17:27.318151", "stderr": "Error: Unable to discover ISCSI Target with CLUSTER SVIP*= <x.
x.x.x> and volume filter= <control-server-1-vol-1 control-server-1-vol-2>", "stdout": "", "stdout_lines": [],
"warnings": []}
```

Discovery Error Correction

- Check if CHAP credentials are correct, that is, check for typos in username and secret (password) provided in setup data.
- Check if the provided *volume_identifiers* information is correct and then use the provided CHAP account (username) to authenticate those volumes.

If any error is identified in the setup data, correct the error, start again and engage with Cisco Support for clarifications.

Updating Volume Authentication

Once swift service is installed, changes in CHAP authentication *(chap_auth* in setup data) require the swift service to be taken offline. Only secret information can be updated. Swift must be offline when password (initiator or target secret) is changed in storage cluster side. Following sequence must be followed for CHAP secret change:

1. Take swift service offline:

   Swift Offline

   ```
   cd <installer_dir>/tools/
   ./swift_tool.py --action offline
   ```

2. Change secret in storage cluster server, for example, through SolidFire management web UI.
3. Update the secret information in *chap_auth* subsection of swift setup data.
4. Reconfigure for setup data changes:

   Reconfigure

   ```
   ciscovim reconfigure --setupfile <updated_setup_data_file>
   ```

Reinstallation of Swift

For swift deployment, it's recommended to always define new volumes and use them in swift setup data. If you are using volumes from previous swift installation, stale swift container/objects might be present on volumes which results in various replication errors in swift log files.

Supported Features

Supported Features

- Platform Security
- Enabling NFVBench
- Customization of Edge
- Installation Mode
- OpenStack Features
- VPP Port Mirroring Usage
- VXLAN-EVPN Setup
- Head-End Replication Option
- Enabling BGP Adjacency
- Neutron Port Rebinding
- Managing Provider/Tenant VLAN Ranges
- Migrate SRIOV
- Augmenting VIC/NIC Pods
- P-GPU
- SR EVPN
- Cinder Volume Multi-attach
- Virtual GPU Support
- Forwarding EFK Logs
- Network File System
- TTY Logging
- Branding VM Workload
- NFVIMON

Platform Security

Platform Security

- Enabling Custom Policy for VNF Manager Post Installation
- Disabling Management Node Accessibility to Cloud API Network
- Secure local Docker Container Registry
- Horizon Hosting through NAT or DNS Aliases
- Cinder Volume Encryption
- Encryption of Secrets
- Customizing SSH Login Banner
- Enabling Red Hat Identify Management (IDM) System
- Enabling Vault on Day 2 in Cisco VIM Pod
- Enabling Trusted Virtual Function on Day 2
- FQDN Support for Cisco VIM Management API

Enabling Custom Policy for VNF Manager Post Installation

During post-installation of a cloud, Cisco VIM helps to enable a VNF manager (such as ESC) to operate and manage tenant VMs in the OpenStack cloud, with additional privileged features.

Some of the VNF managers operate using specific OpenStack features that require the admin role.

Following are the steps to enable the custom policy for VNF Manager:

1. Take a backup of the setupdata file and update the file manually with the configuration below:

   ```
   ENABLE_ESC_PROV: True #Optional; By default, it is False.
   ```

2. Run the following reconfiguration commands:

   ```
   [root@mgmt1 ~]# cd /root/
   [root@mgmt1 ~]# mkdir MyDir
   [root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
   # update the setup_data to update the proxy info
   [root@mgmt1 ~]# cd /root/MyDir/
   [root@mgmt1 ~]# vi setup_data.yaml
   [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
   ```

Disabling Management Node Accessibility to Cloud API Network

Cisco VIM provides cloud connectivity verification from the data and control plane point of view using tools like cloud-sanity, VMTP, and NFVbench, which are typically run from the management node. For these tools to work, reachability to the Cloud API, external, and provider network is a must.

From release Cisco VIM 2.4.3 onwards, you can set the MGMTNODE_EXTAPI_REACH variable to True in the setup_data file to override the need to ensure reachability of management node from Cloud API, external, and provider network.

For example:

```
MGMTNODE_EXTAPI_REACH: True
```

By default, the MGMTNODE_EXTAPI_REACH variable is set to True. If you do not want to use the MGMTNODE_EXTAPI_REACH variable, you can set it to False as part of the Day 0 settings.

- The MGMTNODE_EXTAPI_REACH variable must be set during the initial installation, and cannot be changed later.
- You must ensure that the Cloud API, external, and provider network are properly routable, as Cisco VIM cannot automatically validate the same.

When MGMTNODE_EXTAPI_REACH is set to False, features such as VMTP and NFVbench are no longer accessible from the management node.

Secure local Docker Container Registry

Starting form CVIM 4.2.2, the Local Docker Container Registry will be down during idle conditions and will be brought up only during CVIM operations. There will not be any alerts generated for the local docker container registry. This will be the default behavior.

When the local docker container registry is up under certain conditions during idle state, for security reasons it can be brought down by running the following playbook.

```
#ansible-playbook /root/installer-4.2.2/bootstrap/playbooks/registry-service-stop.yaml
```

Similarly the local docker container registry can be brought up when required by running the following playbook.

```
#ansible-playbook /root/installer-4.2.2/bootstrap/playbooks/registry-service-start.yaml
```

Horizon Hosting through NAT or DNS Aliases

From release Cisco VIM 3.0.0, you can deploy the Horizon portal through NAT or DNS alias. As a security measure, Horizon accepts a list of host addresses (IP or DNS) that are accessible. By default, this list includes the external_lib_vip_addr, the external_lb_vip_fqdn, and the ipv6 address (if applicable) only.

To host Horizon, perform the following steps:

1. Before launching the installation, update the setup_data.yaml file with the following information.

   ```
   HORIZON_ALLOWED_HOSTS:
   -  <NAT-IP>
   -  <NAT-IP>
   ```

   The parameter HORIZON_ALLOWED_HOSTS is optional and accepts the list of IP addresses and/or DNS names that you want to add as allowed hosts. Mostly, this IP address list match with the NAT address used for the deployment.
2. Run the following commands for reconfiguration:

   ```
   [root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir
   [root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
   # update/include the HORIZON_ALLOWED_HOSTS in the setup_data
   [root@mgmt1 ~]# cd /root/MyDir/
   [root@mgmt1 ~]# vi setup_data.yaml [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
   ```

   The administrator can access the Horizon dashboard through aliases or NAT IPs.

Cinder Volume Encryption

Cisco VIM supports the configuration and creation of encrypted volumes managed by Cinder. The encryption is done natively using Linux Unified Key Setup (LUKS). From release Cisco VIM 3.0.0 onwards, this encryption is enabled by default and does not require any installation. Administrators can use the standard OpenStack APIs to create and mount the volumes. No configuration parameters are available in the setup data.

The following are the steps to create an encrypted volume:

1. From the management node, load the OpenStack authentication variables:

   ```
   [root@management-server-cisco~]# source ~/openstack-configs/openrc
   ```

2. Create a volume type that defines the desired encryption parameters using the below command:

   ```
   [root@management-server-cisco images]# openstack volume type create \
   --encryption-provider nova.volume.encryptors.luks.LuksEncryptor \
   --encryption-cipher aes-xts-plain64 \
   --encryption-key-size 256 \
   --encryption-control-location front-end LUKS
   ```

3. Create an encrypted volume using the following command:

```
[root@management-server-cisco images]# openstack volume create --size 1 --type LUKS encrypted_volume
```

The overall storage Input Output Operations Per Second (IOPS) performance penalty is ~35% for Advanced Encryption Standard Cipher Block Chaining (AES-CBC) and ~60% for AES-XTS, when LUKS is enabled. Also, AES-CBC (keysize='128') encryption provides better performance as compared to AES-XTS (key_size='256') encryption mode.

Encryption of Secrets

Cisco VIM installation dynamically generates passwords for each Openstack service and for services running on the management node. By default, these passwords are system generated and are stored in the *secrets.yaml* file on the management node and then subsequently being read by various steps during the installation.

The *secrets.yaml* file is currently protected by Linux file permissions as well as SELinux mandatory access control. A clear text copy of this file is required during installation, reconfiguration, update, and upgrade. Therefore, the *secrets.yaml* file stores the passwords in cleartext, where a hashed version of these passwords is not sufficient.

From release Cisco VIM 3.4.0, Vault is used. Vault is a tool specifically designed to store and access the passwords securely. Vault encrypts the secrets prior to writing them to persistent storage. Hence, gaining access to the raw storage is not enough to access the secrets. To take advantage of this additional hardening option, you can optionally enable Vault in *setup_data.yaml* as a Day 0 option (reconfigure option will be available in the future). With vault enabled, all the passwords used by Cisco VIM services are stored in Vault with Consul as storage backend. To enable Vault, update the setup_data, with the following information as part of Day 0 installation.

```
VAULT: {enabled: True}
```

Once Vault is enabled, the contents of *secrets.yaml* are no longer visible.

To get the following user-relevant secrets, CLI and the corresponding Rest API are provided:

```
"CVIM_MON_PASSWORD", "CVIM_MON_READ_ONLY_PASSWORD", "CVIM_MON_SERVER_PASSWORD",
"ADMIN_USER_PASSWORD", "KIBANA_PASSWORD", "CVIM_MON_PROXY_PASSWORD", "HAPROXY_PASSWORD".
```

Listed below is an example of how to fetch the secrets:

```
# ciscovim list-secrets --getpassword ADMIN_USER_PASSWORD
+---------------------+------------------+
| Secret Key          | Secret Value     |
+---------------------+------------------+
| ADMIN_USER_PASSWORD | D1g8O6Ws2Woav7Ye |
+---------------------+------------------+
```

The command ciscovim *list-secrets* can list all the secrets that are encrypted. TAC/services are trained on how to fetch any of the non-user facing secrets.

Vault is not supported with *aci* as the mechanism driver.

Customizing SSH Login Banner

From release Cisco VIM 3.0.0, you can provide a customized banner that is displayed when an administrator attempts to login to the management node or Unified Management node. An optional parameter *ssh_banner* in the *setup_data* accepts a string or message to be displayed before the login prompt. This message indicates a warning consistent with a company's IT policies.

1. Before launching the installation, take a backup of the *setupdata.yaml* file and update the file manually with the configuration listed below:

```
ssh_banner:
<your Banner Text>

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.
```

2. Run the following commands for reconfiguration:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
# update/include the ssh_banner in the setup_data
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Enabling Red Hat Identify Management (IDM) System

Cisco VIM supports integration with Red Hat Identity Management System which is based on Identity, Policy, Audit (IPA) technology as a reconfigure option. Before you enable this feature, ensure that hostname of all servers including the management node are in lower case FQDN (RRFC4120), and belongs to ip_domain_name defined in IPA_INFO section. For assumptions associated to IDM, see Red Hat IDM. To enable this feature, follow the below steps:

1. Take a backup of the *setupdata* file and update the file manually with the configuration listed below:

```
# STATIC_ROUTE_VIA_BR_MGMT: <list of v4 or v6 network with mask that route via br_mgmt>. # for
inband_ipa; example, [3456:2345:ac34::/64', 2.3.4.0/24]

IPA_INFO:
ipa_servers:
  - hostname: <hostname with fqdn belonging to the ipa_domain_name>
    ipaddresses: # --- Optional
      - '<ipv4_address>'
      - '<ipv6_address>'
  - hostname: <hostname with fqdn belonging to the ipa_domain_name>
    ipaddresses: --- Optional
      - '<ipv4_address>'
      - '<ipv6_address>'
enroller_user: <enroller_username>
enroller_password: <enroller_password>
ipa_domain_name: <ipa_domain_name>

inband_ipa: <True or False>.  # Optional only for IPA over br_mgmt, reconfigurable
```

2. Run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/  # update/include the IPA_INFO
in the setup_data
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

> ⚠ You can change all the parameters other than *ipa_domain_name* via reconfiguration post installation. Also, you cannot reconfigure IPA in conjunction with any other reconfiguration operation.

Enabling Vault on Day 2 in Cisco VIM Pod

Cisco VIM supports Vault as a reconfiguration option.

1. To enable Vault on a pod running Cisco VIM 3.4.1 or later, update the *setup_data.yaml* file as follows:

```
#Vault:
enabled: True # optional, default if not defined is false
```

2. Take a backup of *setup_data* file and update it manually with the configuration details by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
# update the setup_data to enable vault
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Enabling Trusted Virtual Function on Day 2

For releases earlier to Cisco VIM 3.4.2, you must delete the existing VMs on the target compute to enable the trusted virtual function (VF) on a server after the initial installation of the cloud. To avoid the disruption caused to the existing VMs, a reconfigure option is introduced. You can enable trusted VF on the SRIOV ports as a reconfiguration option, on a per server basis using the below steps:

1. To enable trusted_vf on a pod running Cisco VIM 3.4.2 or later, update the SERVERS section for the target computes where trusted_vf is enabled in the *setup_data.yaml* file as given below:

```
SERVERS:
  <target_compute_name>:
    trusted_vf: True
```

2. Take a backup of *setupdata* file and update it manually with the configuration details by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# # update the setup_data to enable trusted_vf on the target computes
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

FQDN Support for Cisco VIM Management API

By default, Cisco VIM uses self-signed IP based TLS certificates for its management services that include REST API, Kibana, Prometheus, and CVIMMON. Optionally, the administrator can use third-party TLS certificates for Cisco VIM management services.

To enable FQDN support for Cisco VIM management API, follow the below steps:

> ⚠ FQDN must resolve to the management node IP address by configured DNS servers. For dual stack deployment, FQDN must resolve to both IPv4 & IPv6 address.

1. Copy the new key, CA root and certificate files into the *~/openstack-configs* folder under the following filenames:

```
# cp <new-ca-root-cert> /installer-xxx/openstack-configs/mercury-ca.crt
# cp <new-key-file> /installer-xxx/openstack-configs/mercury.key
# cp <new-cert-file> /installer-xxx/openstack-configs/mercury.crt
```

2. Obtain the FQDN name from the TLS certificate:

```
# openssl x509 -in /root/installer-xxx/openstack-configs/mercury-ca.crt -text -noout

Sample output:
. . .

Subject: C=US, ST=California, L=San Jose, O=IT, CN=cvim_management.domain.com
```

3. Once copied, run the following Rest API reconfiguration commands:

```
#  cd /root/installer-xxxx/tools

# ./restapi.py -a reconfigure-tls -d <FQDN>    # For example FQDN is cvim_management.domain.com
```

4. For Day 0 deployment, update the *setup_data* with the MGMTNODE_EXTAPI_FQDN set to <FQDN> value, which is used in Step 3 with *reconfigure-tls* option and run *ciscovim install:*

```
MGMTNODE_EXTAPI_FQDN: <FQDN>
```

5. For Day 2 deployment, update the *setup_data* with MGMTNODE_EXTAPI_FQDN set to <FQDN> and run *ciscovim* reconfigure command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# # update the setup_data with MGMTNODE_EXTAPI_FQDN: <FQDN>
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Enabling NFVBench

Enabling NFVBench Post Deployment

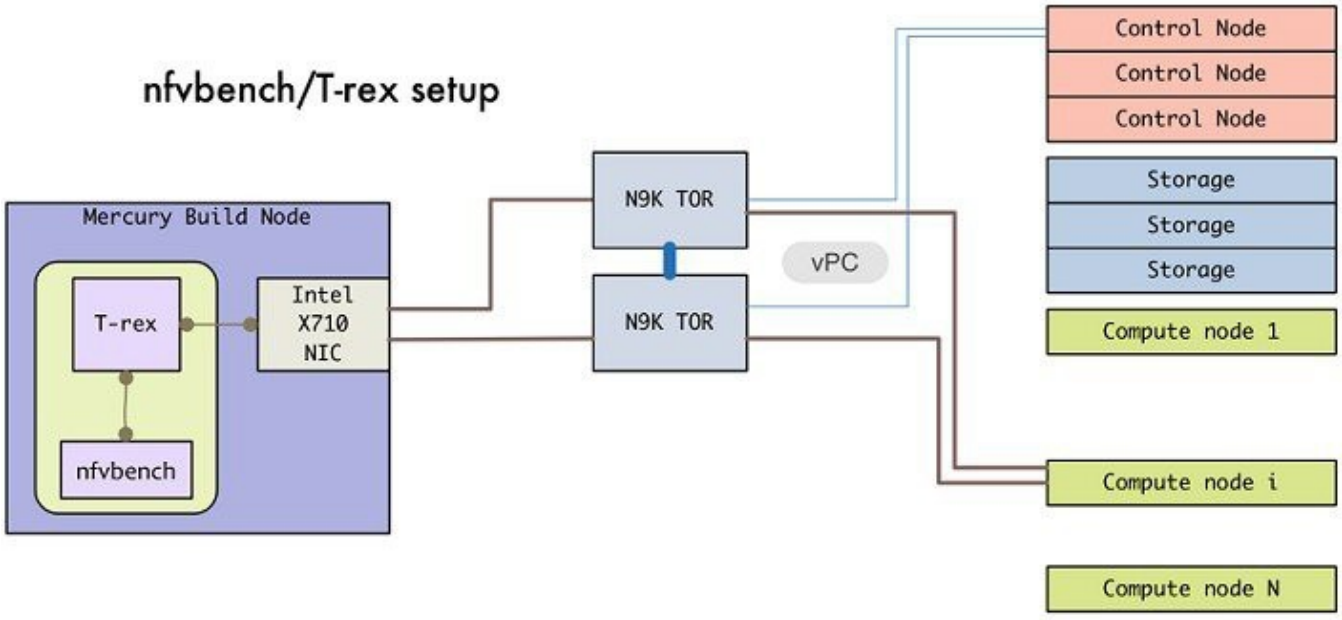
The NFVBench (performance benchmark) is an optional tool. You can deploy NFVBench after the installation of the pod. This section describes how to setup and use NFVbench with Cisco VIM.

Once the pre-requisites for the management node hardware (Intel NIC) are met, add the NFVbench configuration in the setup_data.yaml. By default, NFVbench configuration is not enabled in Cisco VIM as it needs additional hardware. NFVbench also works, when the mechanism driver is OVS or VPP.

Before you begin

- If you are using Quanta servers for the Day 0 BIOS setting of the management node, see Installing Management Node on Quanta Servers.
- An extra 10GE (Intel X710 NIC) or 40GE (Intel XL710) or 25G (Intel xxv710) must be installed on the management node.
- To interact with Intel NIC, the TRex traffic generator uses DPDK interface and uses hardware instead of just software to generate packets. This approach is more scalable and enables NFVbench to perform tests without software limitations.
- Wire two physical interfaces of the Intel NIC to the ToR switches (as shown in the following figure).

If your NIC has more than two ports, use the first two ports only. Connect the first port to the first ToR switch (order is given by setup_data.yaml) and the second port to the second TOR switch. If there is only one ToR switch, connect the first two ports to it as shown in the NFVbench topology figure.



1. Enable the NFVBench configuration in the *setup_data.yaml* file:

   Sample configuration for OVS/VLAN or VPP mechanism driver

```
NFVBENCH:
  enabled: True # True or False
  tor_info: {TORa: eth1/42, TORb: eth1/42} # mandatory, only if AutoTor is Enabled
# nic_ports: 3,4 # Optional input, indicates which two of the four available ports of 10G Intel NIC are
used by NFVbench tool to send and receive traffic.
# Defaults to the first 2 ports of NIC (ports 1 and 2) if not specified.
# Port number must be between 1 and 4, one port cannot be used twice.
# nic_slot: <int> # Optional, defaults to first set of unbonded pair of NIC ports in an Intel 710 or 520
card the code finds; you can run NFVbench via XL710, 520
or X710 card
# Example:
# nic_ports: 1,4 # the first and the last port of Intel NIC are used
# nic_slot: 2 # # Optional, defaults to 1st set of unbonded pair of NIC ports in an Intel 710 or 520
card the code
finds; you can run NFVbench via XL710, 520 or X710 card
# nic_slot: Management node slot on which the NFVbench NIC card is anchored off
# For VXLAN
# vteps: "vtep_ip1,vtep_ip2" # Mandatory and needed only for VXLAN. Specify separated IP pairs in tenant
network and not in the tenant
pool, reconfigurable
##

For VXLAN over vxlan-tenant network
# vteps: "vtep_ip1,vtep_ip2" #Mandatory, specify separated IP pairs in vxlan-tenant network but not in
the vxlan-tenant pool, reconfigurable
# vnis: "vni_id1, vni_id2" # Mandatory, specify the VNI range to be used for all vxlan networks created
by NFVbench for benchmarking
```


vtep_vlans is required if VxLAN is used as encapsulation.

2. Configure minimal settings for NFVBench:

```
# Minimal settings required for NFVbench (mandatory only if tor_info is defined for NFVBench)
TORSWITCHINFO:
CONFIGURE_TORS: <True or False> # True if switches should be configured to support NFVbench
...
SWITCHDETAILS:
- hostname: 'TORa' # Hostname matching 'tor_info' switch name.
  username: 'admin' # Login username for switch user.
  password: 'my_password' # Login password for switch user.
  ssh_ip: <switch_ssh_ip> # SSH IP for switch.

- hostname: 'TORb'
  username: 'admin'
  password: 'my_password'
  ssh_ip: <switch_ssh_ip>
```

The tor_info provides the information to configure the ToR switches. Two ports specified by interfaces are configured in trunk mode in the same *port-channel po*. NFVbench needs the login details to access ToR details and retrieve TX/RX counters. Manual configuration is required if the CONFIGURE_TORS is set to False.

Also, a pair of VTEP VLANs is required for VLAN to VxLAN mapping. Value can be any pair of unused VLAN IDs.

3. To do manual configuration on the ToRs, ensure that you perform the following configuration:

```
interface Ethernetx/y
  switchport mode trunk
  switchport trunk allowed vlan <3000-3049>
  spanning-tree bpdufilter enable
```

4. Reconfigure Cisco VIM to start or restart the NFVBench container. To reconfigure, add necessary configuration to the *setup_data.yaml* file and run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp/root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# # update the setup_data to include NFVBENCH section
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

After the reconfiguration, you can see that the NFVBench container is up and ready for use.

Use the command *~/openstack-configs/openrc* to source the openrc *source* before using the nfvbench.

Customization of Edge

Customization of Edge

From release Cisco VIM 3.0.0 onwards, you need to specify a flavor metadata key *hw:vcpu0_pin_to_shared* to use the optional flavor in OpenStack, that can be set only at Day 0.

When a VM is spawned with the flavor that contains the above metadata set to **Yes**, NOVA allocates additional vCPU on top of the vCPU count specified in the flavor and pin vCPU0 to the pCPU that is reserved in the pool. The pinning of vCPU to pCPU is load-balanced, if hyper-threading is enabled in the host level.

To enable this configuration, set hw:cpu_policy to **dedicated**. And it is often used together with **hw:emulator_threads_policy** being set to **share**, so that the VM emulator threads are also pinned to the same dedicated pool to enable better real-time processing for latency and performance-sensitive VNFs.

To enable this feature, set the following command in the **setup_data.yaml** file on Day 0:

```
ENABLE_VM_EMULATOR_PIN:
<True or False> # optional, default is false
```

The number of cores reserved is determined by VM_EMULATOR_PCORES_PER_SOCKET, which is also pre-defined at the day-0 configuration.

```
VM_EMULATOR_PCORES_PER_SOCKET: < 1 to 4> # Optional,takes effect only when ENABLE_VM_EMULATOR_PIN is true.
If undefined, defaults to value of 1.
```

You can set the NOVA_OPT_LOW_LATENCY flag to enable further optimization on nova libvirt, to achieve lower latency for VM applications. To be specific, it will set **cpu_mode** to **host-passthrough** and **cpu_model_extra_flags** to **tsc-deadline** in nova.conf.

```
NOVA_OPT_FOR_LOW_LATENCY:
True or False # Optional, default to False
```

From release Cisco VIM 3.2.1 onwards, an option to enable Intel's Resource Director Technology (RDT) by Cache Allocation Technology (CAT) is available.

To enable CAT, you must enable NFV_HOSTS option. You can enable the CAT option only as a Day 0 option with the following option in the **setup_data.yaml** file:

```
INTEL_RDT:

ENABLE_CAT: false # Enable Intel CAT, optional and default to False
#Reserved cachelines per socket for sockets, allowed value of 1 to 32.
#Only valid when ENABLE_CAT is sets to True.
RESERVED_L3_CACHELINES_PER_SOCKET: 3
```

The cachelines reserved for hosts are not immediately applied. When first VM with the cacheline requirements lands on the any NUMA node of one compute node, Cisco VIM performs the cacheline partitioning on the host. If VM with no cacheline requirements are spawned (as defined via flavor) on one compute node, all VMs are allowed to use all cachelines available in the CPU. When the last VM with cacheline requirements is deleted from any NUMA node of one compute node, Cisco VIM resets the cacheline masks so that all new and existing VMs are allowed to use all available cachelines again.

To support extreme low latency (< 50 micro-seconds) requirements for vRAN workload, Cisco VIM integrates with Intel N3000 FPGA card for both hardware offload and I/Os. The option of N3000 Intel card is only allowed with Quanta servers, and the following item in the setup_data enables the cards on the servers. These configurations have effect only on computes where the N3000 cards are installed.

```
# Intel FPGA N3000 NIC (for QCT now)
# By default, FPGA VF is not enabled.
# To enable, define a value in the range from 1 to 8.
# INTEL_FPGA_VFS: <integer value from 1 to 8>
# By default, FPGA VF is not enabled.
# VFS support for Intel FPGA N3000 NIC (for QCT now) for SRIOV
# INTEL_VC_SRIOV_VFS: <integer value from 1 to 32>
```

You can enable the virtual function (VFS) values optionally at a per-server level, however, the global configuration is needed as listed below:

```
SERVERS:
compute-server-1:
INTEL_FPGA_VFS: <integer value from 1 to 8>
INTEL_SRIOV_VFS: <integer value from 1 to 32>
INTEL_VC_SRIOV_VFS: <integer value from 1 to 32>
```

⚠ You can enable single or multiple options listed above on a per server basis

Installation Mode

Installation Mode

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

- Connected: In this mode, the setup must be connected to Internet or Cisco VIM software hub to fetch artifacts and docker images.
- Disconnected: In this mode, Cisco VIM is not connected to 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.

OpenStack Features

Memory/CPU Usage

Memory/CPU Usage

- Memory Over-Subscription Ratio
- RAM Over-Subscription Ratio
- CPU and RAM Allocation Ratio

Memory Over-Subscription Ratio

Cloud allows you for over-subscription of resources such as CPU and memory. By default, the memory over-subscription value is set to 1.5. You can adjust the global memory over-subscription value in the range of 1.0 to 4.0.

> ⚠ You can change the default values only at the beginning of the installation

Run the following command to set the NOVA_RAM_ALLOCATION_RATIO, on fresh installation:

```
# cd installer-<tagid>/openstack-configs/
# update NOVA_RAM_ALLOCATION_RATIO value in openstack_config.yaml
```

Once the NOVA_RAM_ALLOCATION_RATIO is set, continue with the rest of the steps as planned for installation.

RAM Over-Subscription Ratio

Cloud allows you for over-subscription of CPU and memory. By default, the CPU over-subscription value is set to 16.0. You can adjust the global CPU over-subscription value in the range of 1.0 to 16.0.

> ⚠ You can change the default value before the installation begins.

Run the following command to set the NOVA_CPU_ALLOCATION_RATIO on fresh installation:

```
# cd installer-<tagid>/openstack-configs/
# update NOVA_CPU_ALLOCATION_RATIO value in openstack_config.yaml
```

Once the NOVA_CPU_ALLOCATION_RATIO is done, continue with the rest of the steps as planned for installation.

CPU and RAM Allocation Ratio

By default, OpenStack sets the CPU and RAM allocation ratio globally. Cisco VIM allows you to change the default CPU and RAM allocation ratio as a Day 0 or Day 1 operation. From Cisco VIM 3.4.1, you can set these parameters on a per compute basis via addition/removal of the target computes. From Cisco VIM 3.4.3, the CPU and RAM allocation ratio can be set on Day 2 via reconfigure option on a per compute basis.

> ⚠ Setting the parameters on a per compute basis via reconfigure option enforces the settings only for the new VMs that get launched on the target compute, but not for the existing VMs.

To enable CPU and RAM allocation ratio on a per compute basis, update each of the target compute nodes with the following configuration in the *setup_data.yaml* file, after taking a backup of the *setup_data.*

```
SERVERS:
   compute-server-1:
   NOVA_CPU_ALLOCATION_RATIO: 1.0 <float, range: 0.958-16.0>   # <== optional, override the
NOVA_CPU_ALLOCATION_RATIO configuration defined in openstack_config.yaml
   NOVA_RAM_ALLOCATION_RATIO: 1.0 <float, range: 1.0-4.0>      # <== optional, override the
NOVA_RAM_ALLOCATION_RATIO configuration defined in openstack_config.yaml
```

Run the following reconfiguration command after updating the target *setup_data:*

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/ # update the RAM and/or CPU Allocation
Ratio on a per compute basis
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```


For a given compute server, if NOVA_CPU_ALLOCATION_RATIO and/or NOVA_RAM_ALLOCATION_RATIO entries are added, you cannot drop it via reconfigure. To get to the global value, it is recommended to set the target compute's NOVA_CPU_ALLOCATION_RATIO and/or NOVA_RAM_ALLOCATION_RATIO value to that of global and run re-configure.

DHCP Reservations

DHCP Reservations for VM’s MAC Addresses

From release Cisco VIM 3.2.0 onwards, you can have DHCP reservations for virtual machine MAC addresses, to get the same IP address always regardless of the host hypervisor or operating system they are running. To avail this optional feature, few restrictions exist.
If the MAC address ends with 00:00, then

- First entry of the first octect must be a Hex
- Second entry of the first octect must be 2, 6, a or e
- No entry can start with *fe*

For example, the MAC address entry can be [a-f][2,6,a,e]:yz:uv:ws:00:00, but not starting with *fe*

1. To enable this feature, take the backup of the of the setupdata file and update the file manually with the configuration listed below:

```
BASE_MACADDRESS: <[a-f][2,6,a,e]:yz:uv:ws:00:00>        # Entry cannot starting with 'fe'
```

> ⚠ To avoid mac-address collision, ensure that a minimum of last three octects is 00. For example: BASE_MACADDRESS: <[a-f][2,6,a,e]:[a-f0-9][a-f0-9]:[a-f0-9][a-f0-9]:00:00:00>

2. Run the following commands for reconfiguration:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
# update/include the BASE_MACADDRESS in the setup_data
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Trusted Virtual Functions

Setting up Trusted Virtual Functions

The kernel feature allows the virtual functions (VF) to become trusted by the physical function and perform some privileged operations such as enabling VF promiscuous mode and changing VF MAC address within the guest. The inability to modify MAC addresses in the guest prevents the users from being able to easily setup up two VFs in a fail-over bond in a guest. To avail this feature, enable the following under each of the target compute nodes that are enabled with SRIOV:

```
SERVERS:
 compute-server-1:
 trusted_vf: <True or False> # <== optional, only applicable if its SRIOV node
```

You can avail this feature on Day 0, or enable in a compute on Day 2 by removing it and adding it back into the cloud after updating the setup_data with the configuration intent. From Cisco VIM 3.4.2, the trusted_vf can be set on Day 2 via reconfigure option on a per compute basis that are running with SRIOV. To enable trusted_vf on a per compute basis, update each of the target compute nodes with the following configuration in the *setup_data.yaml* file, after taking a backup of the *setup_data.* Then run the reconfiguration command after updating the target *setup_data:*

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/ # update the trusted_vf option on a per
compute basis which are running SRIOV
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Buffer Size Setup

Setting up Transmit and Receive Buffer Size

By default, the transmit and receive buffer for the interfaces on each of the servers is set to 1024. This feature allows you to set the rx_tz_queue_size to 512 on a per server basis, which is a requirement for some VNFs. To avail this feature, enable the following under each of the target compute nodes:

```
SERVERS:
 compute-server-1:
 rx_tx_queue_size: <512 or 1024> # optional, default if not defined is 1024
```

You can avail this feature on Day 0, or enable in a compute on Day 2 by removing it and adding it back into the cloud after updating the setup_data with the configuration intent.

VPP Port Mirroring Usage

VPP Port Mirroring Usage

- Overview
- Port Mirroring

Overview

The VPP port mirroring enables you to selectively create a mirror port to a VM. This mirror port detects all the packets sent and received by the VM without having access to the VM. The packets captured in this manner are saved as pcap files which are then used by tools like Wireshark and so on for further analysis.

The following CLIs are available in Cisco VIM:

- vpp-portmirror-create: Tool to create mirrored ports corresponding to Openstack ports.
- vpp-portmirror-delete: Tool to delete mirrored ports.
- vpp-portmirror-list: Tool to get a list of currently mirrored port.

The VPP port mirror tools perform the following:

- Checks if the specified port is a valid neutron port with valid UUID pattern.
- Checks if there is a corresponding Vhost interface in the VPP instance for the neutron port specified.
- Checks if the port has already mirrored.

Port Mirroring

1. Identify the VM that you want to monitor and the compute host on which it runs.

```
From the Management node, execute the following:
#cd /root/openstack-configs
# source openrc
# openstack server show vm-7
+--------------------------------------+------------------------------------------------------------+
| Field                                | Value                                                      |
+--------------------------------------+------------------------------------------------------------+
| OS-DCF:diskConfig                    | AUTO                                                       |
| OS-EXT-AZ:availability_zone          | nova                                                       |
| OS-EXT-SRV-ATTR:host                 | k07-compute-1                                              |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | k07-compute-1                                              |
| OS-EXT-SRV-ATTR:instance_name        | instance-0000004d                                          |
| OS-EXT-STS:power_state               | Running                                                    |
| OS-EXT-STS:task_state                | None                                                       |
| OS-EXT-STS:vm_state                  | active                                                     |
| OS-SRV-USG:launched_at               | 2018-05-10T02:40:58.000000                                 |
| OS-SRV-USG:terminated_at             | None                                                       |
| accessIPv4                           |                                                            |
| accessIPv6                           |                                                            |
| addresses                            | net1=10.0.1.4                                              |
| config_drive                         |                                                            |
| created                              | 2018-05-10T02:40:37Z                                       |
| flavor                               | m1.medium (ac4bdd7f-ff05-4f0d-90a5-d7376e5e4c75)           |
| hostId                               | 8e7f752ab34153d99b17429857f86e30ecc24c830844e9348936bafc   |
| id                                   | 46e576c1-539b-419d-a7d3-9bdde3f58e35                       |
| image                                | cirros (e5e7e9d8-9585-46e3-90d5-4ead5c2a94c2)              |
| key_name                             | None                                                       |
| name                                 | vm-7                                                       |
| os-extended-volumes:volumes_attached | []                                                         |
| progress                             | 0                                                          |
| project_id                           | 434cf25d4b214398a7445b4fafa8956a                           |
| properties                           |                                                            |
| security_groups                      | [{u'name': u'my_sec_group'}]                               |
| status                               | ACTIVE                                                     |
| updated                              | 2018-05-10T02:40:58Z                                       |
| user_id                              | 57e3f11eaf2b4541b2371c83c70c2686                           |
+--------------------------------------+------------------------------------------------------------+
```

2. Identify the neutron port that corresponds to the interface that you want to mirror.

```
# openstack port list | grep 10.0.1.4
| ed8caee2-f56c-4156-8611-55dde24f742a | | fa:16:3e:6a:d3:e8 | ip_address='10.0.1.4',
subnet_id='6d780f2c-0eeb-4c6c-a26c-c03f47f37a45' |
```

3. SSH to the target compute node on which the VM is running and connect the VPP docker container.

```
# vpp
neutron_vpp_13881 [root@k07-compute-1 /]#
The syntax of the Port mirror create tool is as follows:
neutron_vpp_13881 [root@k07-compute-1 /]# vpp-portmirror-create
Option -p (--port) requires an argument
-p --port [arg] Port in openstack port uuid format. Required.
-d --debug Enables debug mode
-h --help This page
-n --no-color Disable color output
VPP port mirror utility.
```

4. Create a port mirror using the neutron port ID identified in Step 2. The CLI tool displays the mirrored interface name:

```
neutron_vpp_13881 [root@k07-compute-1 /]# vpp-portmirror-create -p ed8caee2-f56c-4156-8611-55dde24f742a
===============[ Port Mirroring ]==========================
2018-05-14 22:48:26 UTC [ info] Interface inside vpp is VirtualEthernet0/0/1 for Openstack port:
ed8caee2-f56c-4156-8611-
55dde24f742a
2018-05-14 22:48:26 UTC [ info] Port:ed8caee2-f56c-4156-8611-55dde24f742a is now mirrored at taped8caee2
2018-05-14 22:48:26 UTC [ notice] Note! Please ensure to delete the mirrored port when you are done
with debugging
```

.

> ⚠ Use the –debug flag to troubleshoot the Linux/VPP commands that are used to set up the port mirror.

5. Use the tap device as a standard Linux interface and use tools such as tcpdump to perform packet capture.

```
neutron_vpp_13881 [root@k07-compute-1 /]# tcpdump -leni taped8caee2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on taped8caee2, link-type EN10MB (Ethernet), capture size 262144 bytes
16:10:31.489392 fa:16:3e:6a:d3:e8 > fa:16:3e:0e:58:7b, ethertype IPv4 (0x0800), length 98: 10.0.1.4
> 10.0.1.10: ICMP echo
request, id 32513, seq 25752, length 64
16:10:31.489480 fa:16:3e:0e:58:7b > fa:16:3e:6a:d3:e8, ethertype IPv4 (0x0800), length 98: 10.0.1.10
> 10.0.1.4: ICMP echo
reply, id 32513, seq 25752, length 64
16:10:32.489560 fa:16:3e:6a:d3:e8 > fa:16:3e:0e:58:7b, ethertype IPv4 (0x0800), length 98: 10.0.1.4
> 10.0.1.10: ICMP echo
request, id 32513, seq 25753, length 64
16:10:32.489644 fa:16:3e:0e:58:7b > fa:16:3e:6a:d3:e8, ethertype IPv4 (0x0800), length 98: 10.0.1.10
> 10.0.1.4: ICMP echo
reply, id 32513, seq 25753, length 64
16:10:33.489685 fa:16:3e:6a:d3:e8 > fa:16:3e:0e:58:7b, ethertype IPv4 (0x0800), length 98: 10.0.1.4
> 10.0.1.10: ICMP echo
request, id 32513, seq 25754, length 64
16:10:33.489800 fa:16:3e:0e:58:7b > fa:16:3e:6a:d3:e8, ethertype IPv4 (0x0800), length 98: 10.0.1.10
> 10.0.1.4: ICMP echo
reply, id 32513, seq 25754, length 64
^C
```

6. Obtain a list of all the mirrored ports.

```
neutron_vpp_13881 [root@k07-compute-1 /]# vpp-portmirror-list
VPP interface VPP-side span port Kernel-side span port Neutron port
------------- ----------------- -------------------- ------------
VirtualEthernet0/0/0 tapcli-0 tap88b637e4 net-vpp.port:88b637e4-43cc-4ea2-8a86-2c9b940408ec
VirtualEthernet0/0/1 tapcli-1 taped8caee2 net-vpp.port:ed8caee2-f56c-4156-8611-55dde24f742a
```

7. Remove the mirrored port.

```
neutron_vpp_13881 [root@k07-compute-1 /]# vpp-portmirror-delete -p ed8caee2-f56c-4156-8611-55dde24f742a
===============[ Port Mirroring Operation]=========================
2018-05-14 23:18:49 UTC [ info] Interface inside vpp is VirtualEthernet0/0/1 for Openstack
port:ed8caee2-f56c-4156-8611-
55dde24f742a
Deleted.
2018-05-14 23:18:49 UTC [ info] Port:ed8caee2-f56c-4156-8611-55dde24f742a is now un-mirrored
```

VXLAN-EVPN Setup

VXLAN-EVPN Setup

Choose single VXLAN or multi-VXLAN (multi refers to 2) network terminating on the same box on day-0. Two vxlan segments such as vxlan-tenant and vxlan-ecn are defined.

For single VXLAN network, define only the vxlan-tenant. For two-VXLAN network, define vxlan-ecn segment along with vxlan-tenant network.

To enable VXLAN/EVPN in Cisco VIM, define the following in the setup-data file during the Day-0 deployment. Optionally, you can overload the configuration with that of head-end-replication for static VXLAN configuration.

1. In the **Networking** section, define the segment vxlan-tenant:

```
NETWORKING:
... networks:
.....
- # only needed when NETWORK_OPTIONS is vxlan, and TOR is Cisco NCS5500
vlan_id: <2003>
subnet: <191.168.11.0/25>
gateway: <191.168.11.1>
## 'pool' can be defined with single ip or a range of ip pool:
- <191.168.11.2,191.168.11.5>
- <191.168.11.7 to 191.168.11.12>
- <191.168.11.20>
segments:
  vxlan-tenant
   # only needed when NETWORK_OPTIONS is vxlan, and TOR is Cisco NCS5500, and second VXLAN segment is
required
vlan_id: <2005>
subnet: <191.165.11.0/25>
gateway: <191.165.11.1>
## 'pool' can be defined with single ip or a range of ip pool:
- <191.165.11.2,191.165.11.5>
- <191.165.11.7 to 191.165.11.12>
- <191.165.11.20>
segments:
vxlan-ecn
```

2. Define the vxlan section under NETWORK_OPTIONS, only allowed for Cisco NCS 5500 as ToR:

```
# Optional, only allowed for NCS-5500 as tor NETWORK_OPTIONS:
vxlan:

vxlan-tenant:

provider_network_name: <name of provider network>
bgp_as_num: <int value between 1 and 232-1>
bgp_peers: ['ip1', 'ip2'] ---> list of min length 1, Peer Route Reflector IPs
bgp_router_id: 'ip3' ---> Indicates the router ID to use for local GoBGP cluster, part of vxlan-tenant
network but not in the pool
head_end_replication: # Optional, can be brought in as reconfigure
-    vtep_ips: vni_id1:vni_id2, vni_id3, ... (upto as many Remote POD vteps, as required)

vxlan-ecn:

provider_network_name: <name of provider network> bgp_as_num: <int value between 1 and 232-1>
bgp_peers: ['ip1', 'ip2'] ---> list of min length 1, Peer Route Reflector IPs
bgp_router_id: 'ip3' ---> Indicates the router ID to use for local GoBGP cluster, part of vxlan-ecn
network but not in the pool
head_end_replication:    # Optional and reconfigurable
- vtep_ips: vni_id1:vni_id2, vni_id3, ... (upto as Remote POD many vteps, as required)
```

Following are the assumptions for the HER feature:

- VNIs can repeat across two or more remote POD VTEPs for HA.
- VNIs cannot repeat for the same remote POD VTEP.
- Within the same network segment, no remote POD VTEPs IP adddress can repeat.

3. In the **SERVERS** section, define vxlan_bgp_speaker_ip for each controller node.

The vxlan_bgp_speaker_ip belongs to the vxlan network, but not part of the IP pool defined in the vxlan segment.

```
control-server-1:
.....
# bgp_speaker_addresses: {vxlan-tenant: <ip address> # <== optional, only when NETWORK_OPTIONS is vxlan
network, for
controller node only; IP belongs to the vxlan-tenant network, but not part of the pool as defined in the
network section

vxlan-ecn: <ip address>} # <== optional, only needed for multi-vxlan scenario and only when
NETWORK_OPTIONS is vxlan network,
for controller nodes only; IP belongs to the vxlan-ecn network, but not part of the pool as defined in
the network section
```

Setting up the BGP route-reflector and accessing it over the VXLAN network from the three controllers is outside the scope of Cisco VIM automation.

For head-end-replication option, define Local POD vtep_ips on all servers that act as compute nodes:

```
# vtep_ips:{vxlan-tenant: <ip address>, vxlan-ecn: <ip address>} #IPs must belong to the associated IP
pool of vxlan-tenant and vxlan-ecn
  networks
```

From release Cisco VIM 2.4.9, the BGP session between the controllers and route-reflector is set to be Layer 3 adjacent. By default, it is L2 adjacent. To support Layer 3 adjacency, define bgp_mgmt_address for each controller.

```
# bgp_mgmt_addresses: {vxlan-tenant: <ip address >, vxlan-ecn: <ip address>} # <== optional, only when
NETWORK_OPTIONS
is vxlan network, for contoller node only, needed when BGP peer is  over L3. IP addresses are unique and
are from management network,
but not part of pool.
```

Head-End Replication Option

Enabling Head-End Replication (HER) Option

For the releases Cisco VIM 2.4.9 and later, the multi-VXLAN EVPN based design optionally supports the static implementation of VXLAN technology using head-end replication. HER helps leverage the VXLAN technology, regardless of the hardware or software limitation of the VXLAN feature set at the remote end of the VTEP tunnel.

With the static information defined in the HER setup_data, VPP performs the head-end replication to all defined remote VTEPs and the Layer-2 Forwarding Information Base (L2FIB) MAC-IP table is populated based on flood and learn. When EVPN coexists with HER, Cisco VIM considers them as two different sets of BGP speakers each giving the information which ends up in the same etcd FIB table.

In Cisco VIM, the EVPN acts as the primary mechanism and HER as the fallback methodology. You can add or remove HER to or from an existing EVPN pod through Cisco VIM reconfigure option.

Following are the assumptions for the HER feature:

- VNIs can be allowed in the range of 1 to 65535.
- VNIs can be repeated across two or more remote POD VTEPs for HA.
- VNIs cannot be repeated for the same remote POD VTEP.
- Within the same network segment, no remote POD VTEPs IP address can be repeated.

To enable HER, follow the below steps:

1. Ensure that multi-VXLAN feature exists in day-0 configuration of the setup_data. Add a new section called head-end-replication under the NETWORK_OPTIONS -> vxlan -> vxlan-ecn and vxlan-tenant sections.

```
NETWORK_OPTIONS:
vxlan:
vxlan-tenant:
head_end_replication: # Optional and reconfigurable
- vtep_ips: vni_id1:vni_id2, vni_id3, ... (upto as many remote POD vteps as required)
vxlan-ecn:
head_end_replication: # Optional and reconfigurable
- vtep_ips: vni_id1:vni_id2, vni_id3, ... (upto as many remote POD vteps as required)

Update all compute nodes with vtep_ip information under the SERVERS section:
SERVERS:
Compute1:
...

For head-end-replication option, define vtep_ips on all servers that act as control and compute node

# vtep_ips: {vxlan-tenant: <ip address>, vxlan-ecn: <ip address>} # These IPs must belong to the
associated IP pool of
vxlan-tenant and vxlan-ecn networks, and must match the existing assigned vtep_ip for EVPN as they are
brought in as
part of reconfiguration.
```

2. To determine the respective vtep_ip on a per segment and server basis, run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# cd /root/installer-<x.y.z>/tools
[root@mgmt1 ~]#./vtep_ip_server_mapping.py
# Update the setup_data to include the HER section and vtep_ip corresponding to the network segment
for the respective servers
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Enabling BGP Adjacency

Enabling BGP Adjacency

From release Cisco VIM 2.4.9 onwards, the Layer 2 or Layer 3 BGP adjacency with the peering route-reflector is supported.

> ⚠ For releases prior to Cisco VIM 2.4.9, only Layer 2 BGP adjacency is supported.

Following are the assumptions made to move a pod from a Layer 2 BGP adjacency to that of Layer 3:

- The controllers with the bgp_speaker_addresses peer with the route-reflector over Layer 3.
- This option is only available when vxlan is enabled as NETWORK_OPTIONS.
- Every vxlan segment (vxlan-ecn and vxlan-tenant) will have its own IPs.
- IPs are picked up from management subnet, but they do not belong in the management pool.
- Switching from Layer 2 to Layer 3 peering is only supported, but not vice-versa.
- Once enabled, the only way to change the bgm_mgmt_address is through a replace controller.

To enable layer BGP adjacency between controllers and peering route reflector, follow the below steps:

1. Update all controller nodes with bgp_mgmt_address where the IPs reside in the management subnet, but not in the management IP pool:

   > ⚠
   > - VXLAN feature must exists in day-0 configuration of the setup_data.
   > - One unique IP must be available per VXLAN segment.

   ```
   SERVERS:
   Control1:
   ...
   # bgp_mgmt_address: {vxlan-tenant: <ip address>, vxlan-ecn: <ip address>} # These IPs must belong to
   the
   management segment, but not in the management IP pool.
   ```

2. Run the following reconfiguration commands:

   ```
   [root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir
   [root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
   # update the setup_data to include HER section and vtep_ips info
   [root@mgmt1 ~]# cd /root/MyDir/
   [root@mgmt1 ~]# vi setup_data.yaml [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
   ```

Neutron Port Rebinding

Rebinding of Neutron Port to Another Network

- Overview
- Limitations
- Prerequisites
- Neutron Rebinding

Overview

In OpenStack implementation, once a Neutron port is created from a Neutron network, the ownership of the port cannot be changed. In some cases, the administrator needs to dynamically change the VM port ownership even if it is already bound to a VM. This is particularly useful in case of VXLAN, where it is required to change the VNI associated with an existing port, to introduce a new VNF-VM in an existing service chain.

For example, you can start with:

Net1 – VM1 – Net2

And the goal is to change to:

Net1 – VM1 – Net3 – VM2 – Net2

The new network that owns the port must be in the same project as the old network. If the new network and old network belongs to a different project, ensure that the new network is accessible in the current project. You can achieve that by utilizing Role-Based Access Control (RBAC) feature in OpenStack. For more details on RBAC configuration, see https://docs.openstack.org/neutron/queens/admin/config-rbac.html.

Only admin or network_owner can grant access to RBAC.

Limitations

1. This is only tested and supported in ML2/VPP context. Needs further development to have consistent behavior across ML2 drivers.
2. The support for OVS may not work.
3. When moving ports from one network to another:
    - The security group rules may not work correctly if the move is across different projects honoring RBAC.
    - The vpp-agent does not track the new IP assigned by OpenStack. Hence, the VM cannot ping other VMs in the same Neutron network (L2). The workaround is to edit the security group and add explicit ICMP permit rules or any other traffic you want to. Note that this caveat is not applicable if the port security is turned OFF for Neutron ports.

Prerequisites

The MAC_address must be unique, so that no mac-address duplication exists in the newly migrated network.



The Neutron server validates the above limitations and throws error if they are not satisfied.

Neutron Rebinding

Following are the steps to perform neutron rebinding:

1. Remove the fixed_ip on the port. This clears the IP allocation DBs in Neutron and configures the DHCP agent to have the right information:

```
openstack port set --no-fixed-ip <PORT_ID>
```

2. Migrate the port to the new network:

```
openstack port set <PORT_ID> --network-id <NEW_NETWORK_ID>
```

3. Add the *fixed_ip* of the new subnet back to the port:

```
openstack port set <PORT_ID> --fixed-ip subnet=<NEW_SUBNET_ID>
```

Managing Provider/Tenant VLAN Ranges

Managing Provider/Tenant VLAN Ranges

- Increasing Provider and Tenant VLAN Ranges
- Decreasing Provider and Tenant VLAN Ranges
- Changing the Pod

You can increase or decrease the provider and tenant VLAN ranges post pod installation.

Increasing Provider and Tenant VLAN Ranges

Increasing provider/tenant VLAN ranges is available as a Day 2 option.

> ⚠ You must have the tenant and provider networks enabled on the pod from Day 0.

To increase provider and tenant VLAN ranges, enter the TENANT_VLAN_RANGES and/or PROVIDER_VLAN_RANGES in the *setup_data.yaml* file and run the reconfigure command through *Ciscovimclient* as follows:

```
TENANT_VLAN_RANGES: old_vlan_info, new_vlan_info
or/and
PROVIDER_VLAN_RANGES: old_vlan_info, new_vlan_info
```

Decreasing Provider and Tenant VLAN Ranges

To decrease provider and tenant VLAN ranges, update the TENANT_VLAN_RANGES and/or PROVIDER_VLAN_RANGES to be a subset of the original one in the *setup_data.yaml* file and run the reconfiguration command through *ciscovimclient* as follows:

```
TENANT_VLAN_RANGES: subset_old_vlan_info
PROVIDER_VLAN_RANGES: subset_old_vlan_info
```

In the following example, VLAN from tenant or provider VLAN ranges can be reduced only in blocks, that is, you can reduce vlans by a, b, c:f or any other blocks of entry, but cannot reduce it by a subset of the block (for example: c:d).

```
Pre-reduction of VLAN:
TENANT_VLAN_RANGES: a,b,c:f
PROVIDER_VLAN_RANGES: g,h,k:n
```

> ⚠ You cannot remove and add new VLANs at the same time. Also if you remove VLAN, they cannot be reduced to less than two VLAN for tenant network, and less than one VLAN for provider network.

Changing the Pod

To change the pod, copy the setupdata into a local directory and update it manually by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
```

Update the setup_data, by running the following command:

```
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1~]# vi my_setup_data.yaml (update the setup_data with the right info)
```

Run the reconfiguration command as follows:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ./ciscovimclient/ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```


While using auto-ToR via ACI APIs without the APIC plugin, ensure that you update the vim_apic_networks section with the right VLAN information as part of the reconfiguration option.

Migrate SRIOV

Migrate SRIOV in VIC/NIC Pod

To use this feature, ensure that both the card types are available on the SRIOV compute nodes of the pod and with one of the card type participating in SRIOV as part of installation, and then execute the following steps:

You can redeploy the SRIOV ports between 2-X520 and 2-XL710 in a Cisco VIM pod where the control plane and data plane are running OFF Cisco VIC. This is driven through an optional parameter SRIOV_CARD_TYPE listed in the *setup_data.yaml*.

It is assumed that all computes participating in SRIOV has two sets of card types. Reconfiguration fails if the card type with a total of four ports is not available. Cisco recommends you to have two of each of the card type inserted on a per-compute basis, so that the correct network ports from the target network cards are picked by the orchestrator. However, if the SRIOV_CARD_TYPE is present during the fresh installation or during add compute operation, the SRIOV_CARD_TYPE parameter is given preference for the target/configured card type.

You can define the SRIOV_CARD_TYPE at a per-compute level, to override the global definition. This option allows some computes to run with XL-710 and other computes to run with X-520 for SRIOV ports. It should be noted that computes without SRIOV can co-exist in this pod.

1. Take a backup of the setupdata file and update the file manually with the configuration listed below:

```
SRIOV_CARD_TYPE: <X520 or XL710>
and/or update the hardware_info at a per compute level (see example below)
compute-xx:
hardware_info: {SRIOV_CARD_TYPE: <XL710 or X520>}
```

2. Run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
# update the setup_data to include the target SRIOV card type
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Augmenting VIC/NIC Pods

Augmenting VIC/NIC Pods

From release Cisco VIM 2.4.9 onwards, the augmentation of an existing M4 VIC/NIC based pod (some computes have X520, while others have XL710 for SRIOV), with the M5-based VIC/NIC (40G) computes is supported. To use this augmentation feature, you must define the SRIOV_CARD_TYPE at a per compute level (default is X520).

You can add M5-based 40G VIC/NIC computes into the pod in the following scenarios:

Use Case 1: If you run a pod with M4-based computes having only X520 cards, execute the reconfiguration operation and define the SRIOV_CARD_TYPE as XL710 under the hardware_info section of the target compute, to add the compute of M5 with 40G Cisco VIC and two XL710 cards,.

Use Case 2: If you run the pod with M4-based VIC/NIC computes having XL710 cards, execute the add compute operation and define the SRIOV_CARD_TYPE as XL710 for the target compute, to add M5-based compute nodes with XL710 cards.

The following steps 1 through 3 are not applicable for Use Case 2, and you can directly add/remove compute when required.



Identify if the pod has M4 computes running with two XL710 or not, that is, whether the pod is running with Use Case 1 or Use Case 2.

1. If the pod is running with Use Case 1, execute the following command:

   ```
   # ciscovim reconfigure
   ```

2. Take a backup of the setupdata file and update the file manually with the configuration listed below:

   ```
   Update the hardware_info at a per compute level (see example below)
   compute-xx:
   hardware_info: {SRIOV_CARD_TYPE: <XL710 or X520>}
   ```

3. Run the following reconfiguration commands:

   ```
   [root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir
   [root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/ # update the setup_data to
   include the target SRIOV card type [root@mgmt1 ~]# cd /root/MyDir/
   [root@mgmt1 ~]# vi setup_data.yaml [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ciscovim --setupfile /root/MyDir/setup_data.yaml add-computes <m5compute1, ...>
   ```

P-GPU

P-GPU Support

From Cisco VIM 3.4.1, GPU is supported in a passthrough mode. In this mode, the entire physical GPU card is available to the VM.

You can enable GPU in passthrough mode on a per UCS server basis. You can also enable GPU on a per-server basis (post install) through an add compute operation for the given server. As part of the validation, Cisco VIM checks for the presence of the right number of GPU cards. Hence, you must remove the GPU card from the server when you do not intend to use it, else the Cisco VIM hardware validation will fail.

Before you Begin

- The target server must have a minimum CIMC version of 4.0(2f).
- Only Tesla T4 GPU card is supported.
- To enable GPU on a compute node with Tesla T4 cards, you must include the following information about the target server in the *hardware_info* section of the *setup_data.yaml* file:

```
SERVERS:
  target_server_name:
    hardware_info: {NUM_GPU_CARDS: x} # where x is either 1 or 2 depending on if we have 1 or 2
Tesla T4 cards plugged in.
```

To use P-GPU in a given VM:

1. You must use a guest image with the right Nvidia drivers for the Tesla T4 card while spawning a VM with GPU, else the VM will crash on bootup.
2. Modify the OpenStack flavor to add the GPU alias:

   ```
   openstack flavor set gpuflavor –property "pci_passthrough:alias"="t4gpu:x"
   ```

   In the above example, x is 1 for one T4 card and 2 for two T4 cards. Depending on the value of x, the VM will pass through either one or two physical GPU cards.
3. Hide the KVM on the guest VM to bypass the Nvidia driver limitation with the KVM hypervisor.

   ```
   openstack image set gpuimage –property img_hide_hypervisor_id=true
   ```

4. Launch a VM with the above flavor to see the above card passthrough.

   After you perform the above steps, verify the card passthrough by using the *nvidia-smi* command inside the guest VM.

SR EVPN

Deploying Segment Routing EVPN

Before you Begin

- A pair of NCS 5500 acts as the ToR. As Cisco VIM supports only a single pair of NCS, you must properly scale your NCS and choose the right NCS-5500 SKU, so that all ports of all the nodes can be connected to this single pair of NCS. Ensure that you can increase the port count when selecting the NCS model.
- You cannot enable both SR EVPN and VXLAN EVPN on the same Cisco VIM pod. Only one forwarding mechanism can exist within a single pod.
- Cisco VIM administrators must provide the SR network and subnet information for overlays, and EVI or BD label. Cisco VIM does not manage the SR EVPN, it only attaches to an existing SR EVPN.
- You must provide the BGP configuration to the Cisco VIM installer during deployment.
- Cisco VIM 3.4.1 does not support traffic engineering policies for dynamic path selection through BGP SR extensions, or any explicit path selection through CLI or any interaction with an SDN controller.
- Only single homing L2 routes are supported. Only EVPN Type 2 and Type 3 are exchanged with the EVPN Route Reflectors (RR).
- You can enable this feature in Cisco VIM as a Day 0 option using either *setup_data.yaml* file or Unified Management when you install the pod.
- You cannot convert an existing pod to this SR-enabled forwarding mechanism, even if it is set up to use VPP for VLAN or VXLAN based forwarding.

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

For SR EVPN deployment, you must update the *setup_data.yaml* file as given below :

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

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

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


NETWORKING:
A new networking section called sr-mpls-tenant has been introduced to support SR MPLS.
NETWORKING:
  Networks:
 -
    vlan_id: <vlan_id>
    subnet: <subnet in cidr>
    gateway: <gateway address>
    pool:
      - <a.b.c.d,e.f.g.h>
      - <i.j.k.l to m.n.o.p>
      - <q.r.s.t>

    segments:
      - sr-mpls-tenant

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

Cinder Volume Multi-attach

Cinder Volume Multi-attach with Cisco VIM

- Overview
- Assumptions and Caveats
- Creating Multi-attach Volume
- Configuring Multiple-storage Backends in Ceph

Overview

Attaching a volume to multiple hosts or servers simultaneously is used for active/active or active/standby scenarios. This feature is available in Cisco VIM 3.4.1 because of upstream support.

For more information about this feature, see the OpenStack documentation at https://docs.openstack.org/cinder/latest/admin/blockstorage-volume-multiattach.html.

Assumptions and Caveats

1. Ensure that a multi-attach or clustered file system is used on the volumes, else there is a high probability of data corruption. If two VMs attach to the same volume, you must ensure that only one of the two VMs mounts the file system at a time.
2. By default, secondary volume attachments are made in read or write mode. This setup can cause problems for operations such as volume migration. This happens even if you create the volume with read-only permission.
3. Boot from volume is not supported with multi-attach volume.
4. Horizon does not work with volume attach to instance.
5. Use the nova client instead of OpenStack client for multi-attach volume.

Creating Multi-attach Volume

To attach a volume to multiple hosts or servers:

1. To attach a volume to multiple hosts or servers, you need to have the multi-attach flag set to true in the volume details.

   ```
   $ cinder type-create multiattach
   $ cinder type-key multiattach set multiattach="<is> True"
   ```

2. To create the volume you need to use the volume type you created earlier.

   ```
   $ cinder create <volume_size> --name <volume_name> --volume-type <volume_type_uuid>
   ```

   For example:

   ```
   $ cinder create 10 --name multi --volume-type bb5f35a1-bdfd-4744-b9f9-c0752598881d
   ```

3. Use the nova client to attach the volume to the instance.

   ```
   $ nova volume-attach INSTANCE_ID VOLUME_ID auto
   ```

   For example,

   ```
   $ nova volume-attach testvm3 785959c4-889d-421e-985e-074d488ec823
   ```

Configuring Multiple-storage Backends in Ceph

To configure multiple storage backends, enter the following commands:

```
cinder type-create cephssd

cinder type-key cephssd set volume_backend_name=ceph-ssd

cinder type-key cephssd set multiattach="<is> True"
```

Virtual GPU Support

Virtual GPU (vGPU) Support

- Overview
- Enabling GPU

Overview

From Cisco VIM 3.4.3, vGPU is supported in tech-preview mode. In this mode, the NVIDIA virtual GPU software is installed at the virtualization layer along with the hypervisor. The NVIDIA virtual GPU software creates virtual GPUs that enable each virtual machine (VM) to share a physical GPU installed on the server or allocate multiple GPUs to a single VM to power the most demanding workloads. As the CPU workload is offloaded to GPU, better user experience and demanding engineering/creative applications are supported in a virtualized cloud environment.

You can also enable vGPU on a per-server basis via fresh installation or through an add compute operation for the given server. As part of the validation, Cisco VIM checks for the presence of the right number of GPU cards. You must remove the GPU card from the server when it is not required, as it can cause Cisco VIM hardware validation failure.

Enabling GPU

Before you begin

- The target server must have a minimum CIMC version of 4.0(2f).
- Only Tesla T4 GPU card is supported.
- Since Cisco does not have license to redistribute the NVIDIA rpm, obtain the *NVIDIA-vGPU-rhel-7.6_3.10.0_957.27.2-440.43.x86_64.rpm* from NVIDIA and copy it to */root/installer-xxx/openstack-configs* directory.
- Ensure that the sha1 checksum of the *NVIDIA-vGPU-rhel-7.6_3.10.0_957.27.2-440.43.x86_64.rpm* file is: `9a9c862547b153a32ec98ff092cc4a01a5ec7cc1`.
- All guest VGPU drivers need to be based on NVIDIA 10.X GRID software. This is a required as the compute host and guest major versions need to match up for NVIDIA support. For the guest image, obtain a valid license from NVIDIA and ensure that the version of NVIDIA GRID driver is 10.x.

To enable GPU on a compute node with Tesla T4 cards, you must include the following information about the target server in the *hardware_info* section of the *setup_data.yaml* file:

```
SERVERS:
  target_server_name:
    hardware_info: {NUM_GPU_CARDS: x, VGPU_TYPE: <vgpu_type>} # where x is either 1 or 2 depending on whether
the Tesla T4 cards plugged in is 1 or 2.

#Valid VGPU types: T4-1B, T4-2B, T4-2B4, T4-1Q, T4-2Q, T4-4Q, T4-8Q, T4-16Q, T4-1B4, T4-4C, T4-8C, T4-16C
```

After installation, create a flavor that supports VGPU via the following command:

```
# openstack flavor set vgpu_1 --property "resources:VGPU=1"
```

In OSP13 release, all hypervisors that support virtual GPUs accepts only a single virtual GPU per instance. Hence, ensure that you do not set more than one VGPU.

Forwarding EFK Logs

Forwarding EFK logs to External Syslog Server

Cisco VIM supports backup and recovery of the management node. To keep the process predictable and avoid loss of logs, the software supports the capability of forwarding the EFK logs to multiple external syslog servers. It supports minimum of one and maximum of four external syslog servers. You can enable EFK forwarding feature, after the pod is up and running with Cisco VIM, through the reconfigure option.

The Syslog export reconfigure option supports the following:

- Forwarding of EFK logs to external Syslog server on a pod that is already up and running.
- Reconfiguration of the existing external Syslog settings to point to a different syslog cluster.

1. Before starting the installation, update the *setup_data.yaml* file with the following:

```
###################################
# SYSLOG EXPORT SETTINGS
###################################
SYSLOG_EXPORT_SETTINGS:
-
  remote_host: <Syslog_ipv4_or_v6_addr> # required IP address of the remote syslog server
  protocol : udp # required; only value allowed is udp
  facility : <string> # required; possible values local[0-7] or user
  severity : debug # required; only value allowed is debug
  port : <int>; # defaults, port number to 514
  clients : 'ELK' # defaults and restricted to ELK;
-
  remote_host: <Syslog_ipv4_or_v6_addr> # IP address of the remote syslog server #2 (optional)
  protocol : udp # required; only value allowed is udp
  facility : <string> # required; possible values local[0-7]or user
  severity : debug # required; only value allowed is debug
  port : <int>; # defaults, port number to 514
  clients : 'ELK' # defaults and restricted to ELK;
```

2. Take a backup of the *setupdata* file and update the file manually with the configuration listed in the preceding section. Then, run the reconfiguration command as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# # update the setup_data to include Syslog Export info
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

With this configuration, you can export ELK logs to an external syslog server. On the remote host, verify if the logs are forwarded from the management node.

Most of the information other than the remote host information is not needed. Also, the client list is restricted to ELK only.

Network File System

Network File System (NFS) for ELK Snapshot

Cisco VIM optionally supports NFS for ELK snapshots. The remote location specified in the configuration must allow user elasticsearch (2020) and group mercury (500) to read from or write to the path specified in remote_path of the remote_host server.
Before launching the installation, update the *setup_data.yaml* file with the following:

```
###################################
 ## ES_REMOTE_BACKUP
###################################
#ES_REMOTE_BACKUP:    # Set if Elasticsearch backups uses a remote host.
# service: 'NFS'    # Only value supported is NFS.
# remote_host: <ip_addr>    # IP of the NFS server.
# remote_path: </root/es_remote> # Path to location of the backups in the remote server
```

With this configuration, the ELK snapshots are hosted at the remote NFS location. Thereby, ensuring that the management node does not run out of disk space. You can add this configuration to a pod that is already up and running.

TTY Logging

TTY Logging

Cisco VIM supports TTY logging on the management node and cluster hosts through the option in the *setup_data.yaml* file. By default, the TTY logging feature is not enabled, but available only during installation. If SYSLOG_EXPORT_SETTINGS is configured, the TTY audit messages are available in local syslog, Kibana dashboard, and remote syslog.

For the TTY logging to take effect in the management node, reboot the management node based on the customer downtime window.

At the end of the installation, a message **Management node needs to be rebooted for TTY logging to take effect** is displayed.

Before launching the installation, update the *setup_data.yaml* file with the following information:

```
# TTY Logging with pam.d and auditd. Events available in Kibana and remote syslog, if syslog export is enabled
ENABLE_TTY_LOGGING: <True or False>    # default value is False
```

Branding VM Workload

Branding VM Workload

You can customize the Manufacturer output as observed in the output of `sudo dmidecode -t system`. This feature is available as a global option and can be enabled either at Day 0 or Day 2.

To enable this option on Day 0, update the *setup_data* with the following name or value pair before starting the installation.

To change the value on a per compute basis on Day n, update the *setup_data* and remove or add compute nodes:

```
# Optional
CUSTOM_MANUFACTURER_TAG: <string with maximum length of 64 characters in "">
```

You can also bring in the option through CUSTOM_MANUFACTURER_TAG via reconfiguration. For details on how to enable CUSTOM_MANUFACTURER_TAG via reconfiguration, see General Guidelines

You must validate each VNF against the custom tag, to ensure that lifecycle of the same is not impacted.

NFVIMON

NFVIMON

- NFVIMON Overview
- NFVIMON Architecture
- Enabling NFVIMON
- Overview of NFVIMON High Availability

NFVIMON Overview

You can optionally install Cisco VIM with a third-party software known as NFVI Monitor (NFVIMON) to monitor the health and performance of the NFVI. This includes monitoring the physical and logical components of single or multiple NFVI pods. The NFVIMON feature is enabled by the Zenoss. NFVIMON provides extensive monitoring and collection of performance data for various components of the cloud infrastructure including Cisco UCS blade and rack servers, service profiles, Nexus top of rack switches, fabric connections, and OpenStack instances.

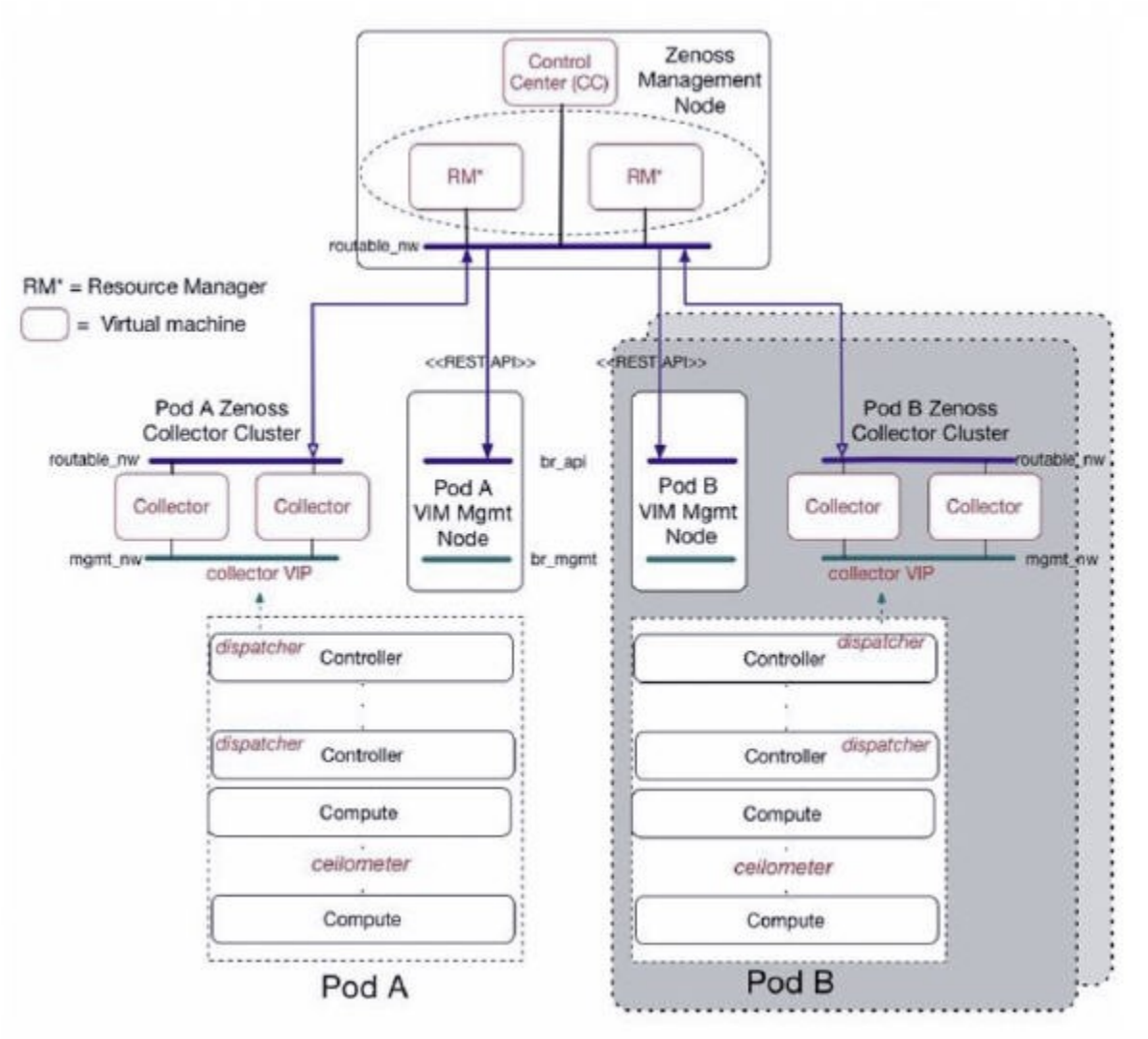
The monitoring system is designed to monitor single or multiple pods from a single management system.

You can enable NFVIMON on an existing pod, by extending the *setup_data.yaml* file with relevant information through the reconfigure option. After this configuration, add the pod as the VIM resource to be monitored in a Control Center.

NFVIMON Architecture

NFVIMON consists of four components:

- ceilometer service for data collection
- collector
- resource manager (RM)
- control-center with Cisco Zenpacks (CZ)



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

As NFVIMON is a third-party software, ensure that the integration of NFVIMON with Cisco VIM is loosely coupled and the VIM automation only deals with installation of the ceilometer service required to monitor the pod. The installation of other NFVIMON components like collector, resource manager (RM) and control-center with Cisco Zenpacks (CZ) are out of scope as they are part of Cisco advanced services.

Before you Begin

1. Ensure that you have engaged with the account team for services engagement on the planning and installation of the NFVIMON accessories along with its network requirements. The image information of collector, Resource Manager (RM) and control-center with Cisco Zenpacks (CZ) can be obtained only through Cisco Advance Services.
2. At a high level, you must have a designated node to host a pair of collector VMs for each pod, and a common node to host CC and RM VMs that can aggregate and display monitoring information from multiple pods.
3. The collector VMs must have two interfaces:
   - Interface with br_mgmt of the VIM.
   - Interface that is routable and reachable to the VIM Installer REST API and RM VMs.
4. As the collector VM is in an independent node, four IPs from the management network of the pod must be pre-planned and reserved. The installation steps of the collector, resource manager (RM) and control-center with Cisco Zenpacks (CZ) are part of Cisco advance service activities.

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

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

- From Cisco VIM 3.2.0, you can use non-root admin keys for monitoring purposes.
- From Cisco VIM 3.4.1, you can run NFVIMON along with CVIM MON. This capability eases the transition from NFVIMON to CVIM MON and allows operators to evaluate CVIM MON while using NFVIMON.
- From Cisco VIM 4.2.1 we only support 'podman' and not 'Docker'. Also we have moved to Ceph 14.2 (Nautilus). Both these changes may impact the NFVIMon to display the information about Docker Containers and Ceph Data. Zenoss team is aware about this and working towards a fix for it.

Enabling NFVIMON

The ceilometer service is the only component in NFVIMON that is managed by Cisco VIM orchestrator. While the ceilometer service collects the metrics to pass OpenStack information of the pod to the collectors, the Cisco NFVI Zenpack available in the controller node gathers the node level information.

1. To enable NFVIMON as part of the Cisco VIM installation, update the *setup_data.yaml* file with the following information:

```
#Define the PODNAME
PODNAME: <PODNAME with no space>; ensure that this is unique across all the pods
NFVIMON:
  MASTER:    # Master Section
    admin_ip: <IP address of Control Centre VM>
  COLLECTOR:    # Collector Section
        management_vip: <VIP for ceilometer/dispatcher to use> #Must be unique across the VIM pod and be
part of br_mgmt network.
        Collector_VM_Info:
-
                hostname: <hostname of Collector VM 1>
                password: <password_for_collector_vm1> # max length of 32
                ccuser_password: <password from master for 'ccuser' (to be used for self monitoring)> #
max length of 32
                admin_ip: <ssh_ip_collector_vm1> # Should be reachable from br_api network
                management_ip: <mgmt_ip_collector_vm1> # Should be part of br_mgmt network
                -
                hostname: <hostname of Collector VM 2>
                password: <password_for_collector_vm2> # max length of 32
                ccuser_password: <password from master for 'ccuser' (to be used for self monitoring)> #
max length of 32
                admin_ip: <ssh_ip_collector_vm2> # Must be reachable from br_api network
                management_ip: <mgmt_ip_collector_vm2> # Must be part of br_mgmt network
        COLLECTOR_TORCONNECTIONS: # Optional. Indicates the port where the collector is hanging off.
Recommended when Cisco NCS 5500 is used as ToR
          - tor_info: {po: <int>, switch_a_hostname: ethx/y, switch_b_hostname: ethx/y}

# Section of MASTER_2 and COLLECTOR_2 are optional and only needed to support NFVIMON in HA
  MASTER_2:    # Master Section
        admin_ip: <IP address of Control Centre VM>
  COLLECTOR_2:    # Collector Section
        management_vip: <VIP for ceilometer/dispatcher to use> #Should be unique across the VIM Pod;
Should be part of br_mgmt network
        Collector_VM_Info:
        -
                hostname: <hostname of Collector VM 1>
                password: <password_for_collector_vm1> # max length of 32
                ccuser_password: <password from master for 'ccuser' (to be used for self monitoring)> #
max length of 32
                admin_ip: <ssh_ip_collector_vm1> # Must be reachable from br_api network
                management_ip: <mgmt_ip_collector_vm1> # Must be part of br_mgmt network
                -
                hostname: <hostname of Collector VM 2>
                password: <password_for_collector_vm2> # max length of 32
                ccuser_password: <password from master for 'ccuser' (to be used for self monitoring)> #
max length of 32
                admin_ip: <ssh_ip_collector_vm2> # Should be reachable from br_api network
                management_ip: <mgmt_ip_collector_vm2> # Should be part of br_mgmt network
        COLLECTOR_TORCONNECTIONS: # Optional. Indicates the port where the collector is hanging off.
Recommended when Cisco NCS 5500 is used as ToR
          - tor_info: {po: <int>, switch_a_hostname: ethx/y, switch_b_hostname: ethx/y}

  DISPATCHER:
        rabbitmq_username: admin # Pod specific user for dispatcher module

  NFVIMON_ADMIN: admin_name # Optional, once enabled, you need to have only one admin that is
reconfigurable to add/update non-root user id.
```


If NFVIMON HA is enabled, ensure that all the admin IPs are on the same subnet for NFVIMON VMs and deployed servers.

2. To monitor ToR, ensure that the following TORSWITCHINFO sections are defined in the *setup_data.yaml* file:

```
TORSWITHCINFO:
SWITCHDETAILS:
-
hostname: <switch_a_hostname>:    # Mandatory for NFVIMON if switch monitoring is needed
username: <TOR switch username>    # Mandatory for NFVIMON if switch monitoring is needed
password: <TOR switch password>    # Mandatory for NFVBENCH; Mandatory for NFVIMON if switch monitoring
is needed
ssh_ip: <TOR switch ssh ip>    # Mandatory for NFVIMON if switch monitoring is needed
....
-
hostname: <switch_b_hostname>:    # Mandatory for NFVIMON if switch monitoring is needed
username: <TOR switch username>    # Mandatory for NFVIMON if switch monitoring is needed
password: <TOR switch password>    # Mandatory for NFVIMON if switch monitoring is needed
ssh_ip: <TOR switch ssh ip>    # Mandatory for NFVIMON if switch monitoring is needed
```

⚠ TORSWITCH monitoring is disabled, when running Cisco VIM with ACI plugin enabled.

3. To initiate the integration of NFVIMON on an existing pod, copy the setupdata into a local directory and update it manually with information listed above, and then run the reconfiguration command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include NFVIMON related info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

4. To initiate the uninstallation of NFVIMON on an existing pod, copy the setupdata into a local directory and remove the entire NFVIMON section listed above, and then run the reconfiguration command as follows:

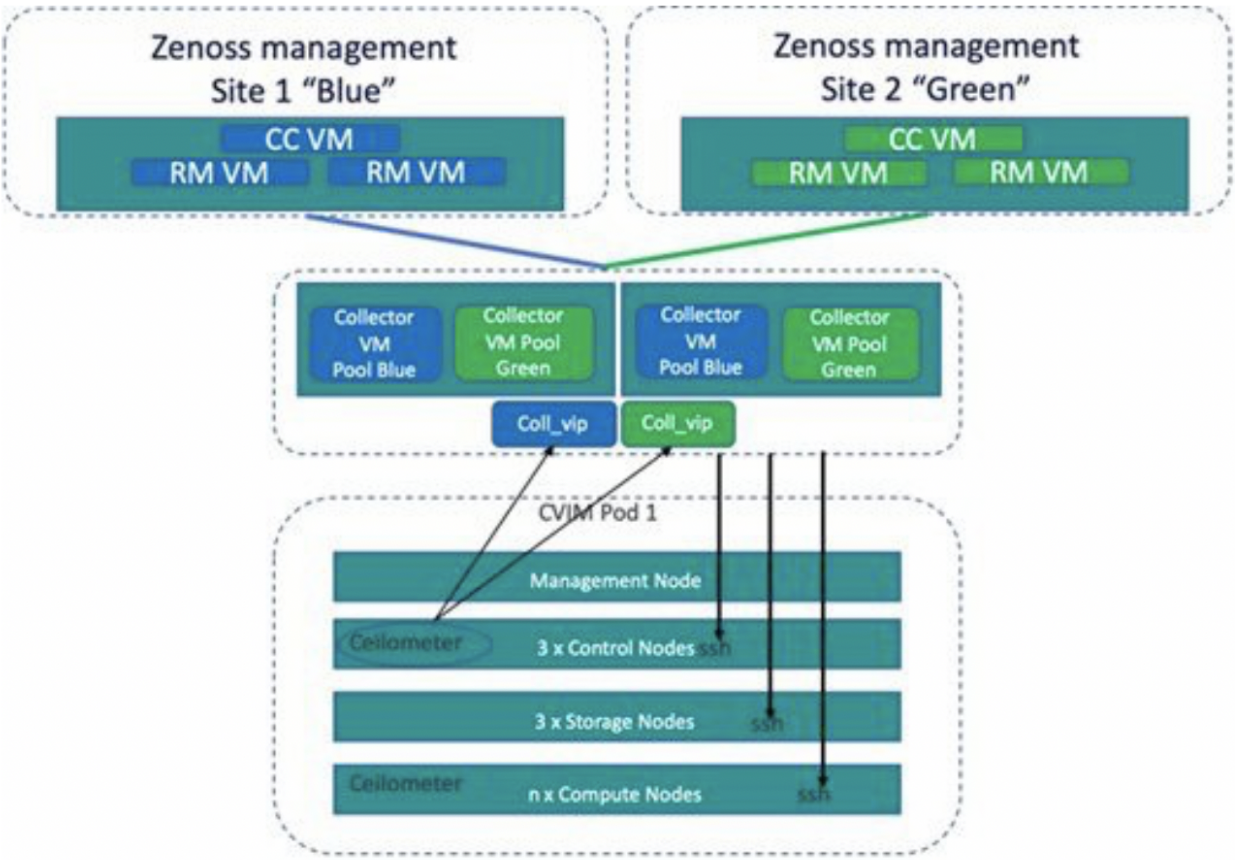
```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to exclude NFVIMON related info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

⚠
- NFVIMON is supported only on a pod running with Keystone v3.
- NFVIMON can run with either root or non-root admin keys for monitoring.

Overview of NFVIMON High Availability

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

The NFVIMON HA architecture is depicted in the figure below:



⚠ You cannot move pods running with NFVIMON in standalone mode to HA mode through reconfiguration.

You can enable NFVIMON HA on Day 0 or Day 1 when the pod is not running with NFVIMON.

Cisco VIM Upgrade

Cisco VIM Upgrade

- Upgrading Cisco VIM in Cloud
- Upgrading Cisco VIM Using Network Installation (from CVIM 3.4.x)
- Upgrading Cisco VIM Using Network Installation (from CVIM 4.0.0)
- Upgrading Cisco VIM Using File-Based Image (from CVIM 3.4.x)
- Upgrading Cisco VIM Using File-Based Image (from CVIM 4.0.0)
- Upgrading Cisco VIM Using USB (from CVIM 3.4.x)
- Upgrading Cisco VIM Using USB (from CVIM 4.0.0)

Upgrading Cisco VIM in Cloud

Upgrading Cisco VIM in Cloud

Cisco VIM allows you to upgrade all OpenStack services, infrastructure services such as RabbitMQ, MariaDB, and HAProxy, and management node containers like Cobbler, ELK, VMTP, and repo. You can upgrade to the latest releases of OpenStack, without redeploying the Cisco NFVI stack from the beginning. During upgrade, you can expect limited control and data plane outage.

Cisco VIM supports an upgrade from Cisco VIM 3.4.y/4.0.0 to its latest version by automatically:

- Upgrading the REST API server that manages the VIM orchestrator using *vim_upgrade_orchestrator.py* script.
- Translating the *setup_data.yaml* file compatible with the target release version.

Ensure that the upgrade is completed in an environment like VNC session.

The following table provides an overview of upgrade methods available to upgrade OpenStack using Cisco VIM.

Upgrade Method	Without Cisco VIM Unified Management	With Cisco VIM Unified Management
Without internet	• Prepare the USB on a staging server. • Plug the USB into the management node. • Follow the update procedure without internet.	• Prepare the USB on a staging server. • Plug the USB into the management node. • Follow the update procedure without internet.
With internet	• Download the .tgz file from the registry. • Follow the update steps with internet	• Download the .tgz file from the registry. • Follow the update steps with internet procedure.

The Internet options refer to management node connectivity to the Internet. If the management server lacks Internet access, a staging server with Internet access is required to download the Cisco VIM installation artifacts to a USB drive, or you can use the file-based image options. If you are using Cisco Software Hub, ensure that it is populated with the target upgrade artifacts. Ensure that you select one upgrade method and stay with it for the full pod lifecycle.

As upgrading Cisco VIM 3.4.x to Cisco VIM 4.2.3 is a multi-stage process, follow the MOP closely and plan to upgrade the Unified Management, independent of the Cisco VIM upgrade.

Cisco VIM upgrade is supported with pods that are in connected or air-gapped mode. Ensure that you select one method and stay with it for the entire upgrade MOP.

- Upgrading of Cisco VIM 3.2.x to Cisco VIM 4.2.3 is not supported.
- In case of upgrade failure, do not try to recover the cloud by yourself. For more information on recovery, contact Cisco Support.

Consider the following points before upgrading:

Note

If the *setup_data.yaml* is configured with *permit_root_login: False* as shown below, do not proceed with Upgrade. Contact CVIM Support for a MOP.

```
# Permit Root Login (optional, default True)
#   True: admin can ssh to management node with root userid and password
#   False: admin can not use root userid for ssh; must use vim_admin_username
#          At least one vim_admin must be configured if this is False
permit_root_login: False
```

1. For multi-stage (Type-1) upgrade:
    a. The upgrade from Cisco VIM 3.4.x to 4.2.3 is supported.
    b. Plan for two maintenance windows with a maximum of 2-3 days gap.
    c. Do not run the cloud in Cisco VIM 3.6.2 for an extended duration, also do not run any pod operation that alters the state of the pod. (Exa mples of pod operations that alter the state of the pod include: add compute, remove compute, add storage, remove storage, replace controller, regenerate secrets, reconfigure, update, rollback, and commit).
    d. Plan for control plane downtime, as the upgrade involves changing the Kernel version.
2. For Type-2 upgrade, plan for control plane downtime, as the upgrade involves changing the Kernel version. The upgrade from Cisco VIM 4.0.0 to Cisco VIM 4.2.3 is supported.
3. Perform a cloud sanity check before initiating the upgrade, to prevent double faults.
4. Before initiating the upgrade, check CIMC logs for any hardware errors including disk errors.

5. If *vim_admin* users are defined in *setup_data.yaml*. Check for *vim_admin users* UID and GID (in /etc/passwd, and /etc/group, respectively). If their UID or GID values are less than 3000, then run the patch for CSCvy10790.

   > ⚠ Do not run the CSCvy10790 patch as non-root vimadmin user

   This patch needs to be run as user root.

   a. If the user root is not permitted to ssh to the management node, change PermitRootLogin parameter in */etc/ssh/sshd_config* from "PermitRootLogin no" to "PermitRootLogin yes" and restart sshd (sudo systemctl restart sshd). This allows root to ssh to the management node.
   b. Before running the CSCvy10790 script, ensure that the all *vim_admin* users whose uid is less than 3000 logged out of the management node. Also any remaining processes owned by these users needs to be stopped.
   c. Login to the management node as root and run CSCvy10790 patch script.
   d. Once CSCvy10790 patch script run is successful, change the PermitRootLogin parameter in */etc/ssh/sshd_config* from "PermitRootLogin yes" back to "PermitRootLogin no". Then restart sshd (systemctl restart sshd).
6. Ensure no custom (non-CVIM) docker containers are running on the management node or control/compute/storage nodes.
7. If Cisco UCS is used, ensure that CIMC of all the servers is running the minimum recommended version.
8. Use the *vim_upgrade_orchestrator.py* script available as part of the target upgrade version. Save a copy of the upgrade script at the /root/ location, before the upgrade.
9. For disconnected upgrades, the USB must be prepopulated with target artifacts.
10. Upgrade to Cisco VIM 4.2.3 is restricted to specific start and endpoints.
11. Upgrading cloud is supported in both connected and air-gapped modes.
12. Upgrading Unified Management on its standalone node is supported.
13. Ensure that you do not change the installation mode during the upgrade.
14. No rollback option is available once an upgrade is initiated. Hence, planning must be done before the upgrade. If you face any issue after the upgrade, reach out to Cisco TAC to recover the cloud.
15. See Prerequisites and Assumptions for Cisco VIM update/upgrade and ensure that all conditions are met.
16. At a high level, the *vim_upgrade_orchestrator.py* script is broken into logical steps to abort in case of failure. In case of failure, reach out to Cisco support for recovery. Cisco does not recommend you to recover the cloud on your own.
17. Upgrade to Cisco VIM 4.2.3 will fail if tech-preview features like LBaaS and CCP in 3.4.x is enabled.
18. If NFVIMON is enabled, the upgrade may fail with pre-check error. Request for the latest *vim_upgrade_orchestrator.py* and use it to overcome the error.
19. Upgrades for Dual-Stack based (were the public_network and cluster_network have different IP versions) Central Ceph Pods are not supported and the workaround is to re-install the Central Ceph pod with CVIM 4.2.3 and to re-configure the Edge pod to use it with a single stack.
20. Ensure that there are no Ceph warnings or Ceph errors before proceeding with the upgrade. Please run the command shown below manually in one of the storage nodes inside the ceph mon container to see if **HEALTH_OK**.

Ceph-Health-Check

```
[root@aio1 ~]# # ON A CONTROLLER NODE, THIS COMMAND BELOW WILL RUN 'ceph -s' INSIDE THE CEPH MON CONTAINER
[root@aio1 ~]# cephmon ceph -s
  cluster:
    id:     6d62abd3-24f5-4dd1-998c-8d88baedc00f
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum ceph-aio1,ceph-aio2,ceph-aio3 (age 6d)
    mgr: aio1(active, since 6d), standbys: aio2, aio3
    osd: 12 osds: 12 up (since 2d), 12 in (since 2d)

  data:
    pools:   2 pools, 768 pgs
    objects: 89 objects, 658 MiB
    usage:   494 GiB used, 13 TiB / 14 TiB avail
    pgs:     768 active+clean

[root@aio1 ~]# # ON A CONTROLLER NODE, THIS COMMAND BELOW WILL RUN 'ceph health' INSIDE THE CEPH MON CONTAINER
[root@aio1 ~]# cephmon ceph health
HEALTH_OK
[root@aio1 ~]#
```

If you find any errors (**HEALTH_WARN** or **HEALTH_ERR**) fix them before proceeding with the Upgrade.

Based on the Cisco VIM image version, the administrator must refer to the appropriate upgrade section (Type 1 or Type 2).

Pre-upgrade Check

Use *vim_upgrade_orchestrator.py* script to performs pre-upgrade checks before initiating the actual upgrade. You can not initiate the actual upgrade until the pre-upgrade check is complete. Note that, the pre-upgrade checks are running at the start of the actual upgrade. Find the following example:

- Before upgrading from 3.4.x to 3.6.2: *./vim_upgrade_orchestrator.py* -i connected -s 3.4.x -n 3.6.2 *--precheck_only*

- Before upgrading from 3.6.2 to 4.2.3: *./vim_upgrade_orchestrator.py* -i connected -s 3.6.2 -n 4.2.3 --*precheck_only*
- Before upgrading from 4.0.0 to 4.2.3: *./vim_upgrade_orchestrator.py* -i connected -s 4.0.0 -n 4.2.3 --*precheck_only*

The pre-upgrade checks include the following:

- Registry connectivity (if connected, installation is initiated).
- Back up the *setup_data.yaml* file, before translation.
- Check and update *INSTALL_MODE* in the *setup_data.yaml* file (connected or disconnected).
- Run cloud sanity on a cloud from the current workspace using Cisco VIM.
- Check for reachability to all nodes including compute, controller, and storage.
- No in-progress or failed operation exists in the Rest API database, as part of pre-validation (check via ciscovim last-run-status).
- Clouds running with VPP must run with GSO disabled. The pre-check relaxes the check for micropod based controller nodes with compute role that does not host any VMs.
- During leap upgrade of RHEL from 7.9 to 8.2, Software RAID is not supported. Convert the nodes that are using software RAID to hardware RAID before the upgrade.
- Checks if the public_network and cluster_network have different IP versions for ceph when upgrading to 3.6.2.
- Check for Ceph HEALTH_OK.
- Check to ensure root login is allowed (permit_root_login:true).
- Check for Ceph Health and Ceilometer.

Type 1 Upgrade

Upgrade from Cisco VIM 3.4.x to Cisco VIM 4.2.3 (x=4, 5, or 6) requires the following:

1. Upgrade the RHEL version from 7.6 to 8.2.
2. Upgrade the OSP version from 13.0 (Queens) to 16.1 (Train) via Rocky and Stein.
3. Upgrade of Python from 2.7 to 3.6.2.
4. Migrate from docker to podman on the servers constituting the cloud.
5. Upgrade of the REST API of the Cisco VIM installer.
6. Upgrade the Red Hat Ceph 3.2 (luminous) to Ceph 4.1 (Nautilus).

Perform the following two-stage upgrade:

1. Stage 1: Upgrades Cisco VIM from 3.4.x to 3.6.2 version running with RHEL 7.9.
    a. Upgrade to Cisco VIM 3.6.2 (in a VNC session). For example: *./vim_upgrade_orchestrator.py* -i connected -s 3.4.x -n 3.6.2.
    b. Reboot the management node.
    c. Reboot compute nodes on which VMs are running.
2. Stage 2: Upgrades Cisco VIM from 3.6.2 to 4.2.3, which requires the leapp upgrade of RHEL from 7.9 to 8.2.
    a. Upgrade to Cisco VIM 4.2.3 (in a VNC session). For example: *./vim_upgrade_orchestrator.py* -i connected -s 3.6.2 -n 4.2.3.
    b. Back up and restore the management node.
    c. Execute the reboot of computes on which VMs are running.
    d. Migrate Ceph volume from Simple to LVM.
    e. Run sanity test on the cloud from 4.2.3.

Before initiating each upgrade, reboot the servers hosting the active VMs. Plan for multiple maintenance windows (MWs) for the full upgrade. When the upgrade to Cisco VIM 3.6.2 and 4.2.3 are completed in MW-1 and MW-2 respectively, rebooting compute with active VMs is performed in between the upgrades, so that all nodes have the right operating system version. Use *vim_upgrade_orchestrator.py* (with in-built checks) available in Cisco VIM 3.6.2 and Cisco VIM 4.2.3 to upgrade to the target version.

Control nodes are rebooted during the upgrade from 3.6.2 to 4.2.3, irrespective of the VMs running on the host. You can expect those VMs to be down for some time while the leapp upgrade reboots take place, or move the VMs off the controllers before the upgrade.

The following table lists the high-level summary of the upgrade steps from Cisco VIM 3.4.x to Cisco VIM 4.2.3:

High-Level Steps	Maintenance Window	Comments
Preparation of 3.6.2 artifacts	Planned activity before maintenance window 1	Only needed for air-gapped installation. Use the *getartifacts.py* provided in Cisco VIM 3.6.2 release. For more details, see NFVI Installation File-Based Image.
Upgrade to Cisco VIM 3.6.2 from Cisco VIM 3.4.x	Maintenance window 1	• Upgrade to RHEL 7.9 from 7.6 • Upgrade of Red Hat CEPH to CEPH 4.1 • Migrate from ceph-disk to ceph- volume simple • Final outcome: OSP13 + RHEL 7.9 + CEPH 4.1
Reboot of management node and computes with active VM	Maintenance window 1	Ensures that the pod is running with RHEL 7.9 with OSP 13.

Preparation of 4.2.3 artifacts	Planned activity before maintenance window 2	Only needed for air-gapped installation. For more details, see NFVI Installation File-Based Image. Use the *getartifacts.py* in Cisco VIM 4.2.3 release with *-U* option.
Upgrade to Cisco VIM 4.2.3 from Cisco VIM 3.6.2	maintenance window 2	• DB migration: Queens-Rocky, Rocky-Stein • Upgrade of Openstack and Infra containers to 4.2.3 • Leapp upgrade from RHEL 7.9 to RHEL 8.2 of all controllers and of compute nodes with no active VMS
Copy the management node backup to a remote system	Maintenance window 2	Copy the management node backup for future restoration. The backup is created automatically during the previous step.
Restoring management node to Cisco VIM 4.2.3	Maintenance window 2	For restoration procedure, see Restoring Management Node.
Planned reboot of compute nodes with active VM	Maintenance window 2	Complete leapp upgrades on compute nodes by rebooting target compute nodes. Use the *cvim_reboot_status.py* provided in */installer-4.2.3/tools/* directory to identity the compute nodes that need reboot. Copy *vim_upgrade_orchestration.py* from the tools dir and to */root/* and initiate the *--finish_upgrade_with_reboot* option or *-fuwr* provide in *vim_upgrade_orchestrator.py* for targeted compute reboots. ⚠ The nodes must be rebooted using the *vim_upgrade_orchestration.py* tool. **Do not** reboot the nodes using 'reboot' or 'shutdown -r'.
Move Ceph Volume from Simple to LVM	Maintenance window 3	Use the *ceph_migration_status.py* provided in */installer-4.2.3/tools/* directory to get the status of Ceph Volume type, and do remove/add storage or replace controller for full/HC and micro pod, to move Ceph volume from simple to LVM.

Type 2 Upgrade

Upgrade from Cisco VIM 4.0.0 to Cisco VIM 4.2.3 constitutes the following:

1. Upgrade within OSP 16.1 (OSP is updated as part of maintenance release).
2. Upgrade within RHEL 8.2 (kernel is updated).
3. Upgrade of the REST API of the Cisco VIM installer.
4. Migrate from docker to podman on the servers constituting the cloud.

Upgrading to Cisco VIM 4.2.3 from Cisco VIM 4.0.0 is a single-stage upgrade requiring a single maintenance window.

The following table lists the high-level summary of the upgrade steps from Cisco VIM 4.0.0 to Cisco VIM 4.2.3:

High-Level Steps	Maintenance Window	Comments
Preparation of 4.2.3 artifacts	Planned activity before maintenance window 1	Only needed for air-gapped installation. For more details, see NFVI Installation File-Based Image. Use the *getartifacts.py* in Cisco VIM 4.2.3 release with *-U* option.
Upgrade to Cisco VIM 4.2.3 from Cisco VIM 4.0.0	Maintenance window 1	Upgrade of Openstack and Infra containers to 4.2.3.
Reboot of management node.	Maintenance window 1	Complete upgrade of management node by rebooting.
Planned reboot of compute nodes with active VM	Maintenance window 1	Complete upgrades on compute nodes by rebooting target compute nodes. Use the *cvim_reboot_status.py* provided in */installer-4.2.3/tools/* directory to identity the compute nodes that need reboot. Copy *vim_upgrade_orchestration.py* from the tools dir and to */root/* and initiate the *--finish_upgrade_with_reboot* option or *-fuwr* provide in *vim_upgrade_orchestrator.py* for targeted compute reboots.

Upgrade to 4.2.3 from 3.4.6 using Cisco Software Hub

It is assumed that the customer has a working SDS Server which is used for 3.4.6 installation.

Also the below mentioned procedure will only work when you start from (3.4.4/3.4.6 version on SDS server).

Perform the following steps to upgrade Cisco VIM using Cisco Software Hub.

1. Install Cisco VIM 3.6.2 tag on existing SDS server with (3.4.4 or 3.4.6) (follow the same procedure used in the 'Cisco VIM Software Hub' section)
2. Install Cisco VIM Rocky tag (https://cvim-registry.com/mercury-releases/cvim-rhel7-osp14/releases/3.810.6/) on existing SDS Server (follow the same procedure used in the 'Cisco VIM Software Hub' section)
3. Install Cisco VIM Stein tag (https://cvim-registry.com/mercury-releases/cvim-rhel8-osp15/releases/3.820.5/) on existing SDS Server (follow the same procedure used in the 'Cisco VIM Software Hub' section)
4. Upgrade Cisco VIM Management node to 3.6.2 using SDS Registry.
5. Once the 3.6.2 upgrade is successful, take the SDS Backup.
6. Install CVIM 4.2.3 ISO on SDS Server (choose SDS Option during ISO Installation)
7. Install CVIM 4.2.3 tag on SDS Server (*./sds_runner/runner.py* command)
8. Once 4.2.3 tag installation is successful, perform SDS backup and restore.
9. Once restore is successful, trigger 4.2.3 upgrade in CVIM management node.

Upgrading Cisco VIM Using Network Installation (from CVIM 3.4.x)

Upgrading Cisco VIM Using Network Installation (from CVIM 3.4.x)

To upgrade from Cisco VIM 3.4.x (x= 4, 5, or 6) to Cisco VIM 4.2.3:

- Upgrade Cisco VIM 3.4.x to Cisco VIM 3.6.2.
- Upgrade Cisco VIM 3.6.2 to Cisco VIM 4.2.3.

Ensure that you plan for multiple maintenance windows to get the upgrade to 4.2.3. For more details, see Upgrading Cisco VIM in Cloud.

1. From the software download site provided by your Cisco account representative, download the *vim_upgrade_orchestrator.py* from Cisco VIM 3.6.2 artifacts:

   ```
   https://{username}:{password}@cvim-registry.com/mercury-releases/cvim36-rhel7-osp13/releases/{release number}/vim_upgrade_orchestrator.py
   ```

2. Execute the upgrade from the */root/* directory:

   ```
   $ cd /root/
   # chmod +x vim_upgrade_orchestrator.py
   #./vim_upgrade_orchestrator.py -i connected -s start_image_tag -n 3.6.2 [-y] # -y if no interactive mode is required.
   # start_image_tag value can be: 3.4.x (x = 4 or 5 or 6)
   ```

   ⚠
   - As the upgrade takes more than 6 hours, execute the upgrade in a VNC. 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 of upgrade.
   - During the upgrade, do not run other Cisco VIM actions.
   - Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

3. After upgrading to Cisco VIM 3.6.2, identify the computes that have active VMs running on them, and plan for its reboot to ensure that built-in redundancy exists for controller and storage nodes. The reboot of these nodes is automated during software upgrade to Cisco VIM 3.6.2.

   The computes that does not have any VNFs in ACTIVE state are automatically rebooted during software upgrade.

   - No pod management operation is allowed until the reboot of all Cisco VIM nodes is successful.
   - In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

   To monitor and reboot the compute nodes through *ciscovim* CLI, see the sections *Managing Reboot of Cisco VIM Nodes* and *Managing Reboot Status of Cisco VIM Nodes* under Managing Power and Reboot.

4. Reboot the management node and then plan for another maintenance window to upgrade to Cisco VIM 4.2.3.
5. From the software download site provided by your Cisco account representative, download the *vim_upgrade_orchestrator.py* from Cisco VIM 4.2.3 artifacts.

   For example:

   ```
   https://{username}:{password}@cvim-registry.com/mercury-releases/cvim42-rhel8-osp16/releases/{release number}/vim_upgrade_orchestrator.py
   ```

6. Execute the upgrade from the */root/* directory:

   ```
   $ cd /root/
   # chmod +x vim_upgrade_orchestrator.py
   #./vim_upgrade_orchestrator.py -i connected -s 3.6.2 -n 4.2.3 [-y] # -y if no interactive mode is required
   ```

   ⚠
   - As the upgrade takes more than 6 hours, execute the upgrade in a VNC. 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 of upgrade.
   - During the upgrade, do not run other Cisco VIM actions.
   - Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

> ⚠ Do not perform any pod operation after upgrading to 3.6.2 or in 3.6.2 that alters the state of the pod until the following steps are completed. Examples of pod operations that alter the state of the pod include: add compute, remove compute, add storage, remove storage, replace controller, regenerate secrets, reconfigure, update, rollback, and commit.

7. Copy the management node backup taken as part of the upgrade in Step 6, to a remote server.
8. Restore the management node using the backup.
9. Plan on VNF downtime to complete leap upgrade on compute nodes via *--finish_upgrade_with_reboot* option provided in *vim_upgrade_orchestrator.py:*

```
$ cd /root/
# chmod +x vim_upgrade_orchestrator.py
#./vim_upgrade_orchestrator.py -i connected -s 3.6.2 -n 4.2.3 --finish_upgrade_with_reboot <", separated
compute nodes in quotes"> [-y] # -y if no interactive mode
is required
```

10. For pods getting upgraded from Cisco VIM 3.x, Ceph volume continues to run in simple mode. Migrate the Ceph volume type to LVM.
    a. Use the tool provided in the *installer-4.2.3/tools/* directory to find out the Ceph nodes that are pending for migration to LVM as Ceph volume. To execute the tool, use the command:

    ```
    # cd /root/installer-4.2.3/tools
    # python ceph_migration_status.py
    ```

    b. Move each Ceph node from simple to LVM mode by performing a remove storage operation followed by an add storage operation for each storage node. Ensure that Ceph is in HEALTH_OK state before initiating each add/remove storage operation. For Micropod, you can migrate the Ceph volume type by performing a replace controller operation for each controller node.

    > ⚠ As Ceph migration takes long time, take subsequent maintenance window for this activity.

    c. After the node has been migrated, use the *ceph_migration_status.py* tool to verify Ceph volume type is LVM.
11. Execute *cloud-sanity* and ceph health check, to ensure that the cloud is completely up and running.

> ⚠ In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

Upgrading Cisco VIM Using Network Installation (from CVIM 4.0.0)

Upgrading Cisco VIM Using Network Installation (from CVIM 4.0.0)

1. From the software download site provided by your Cisco account representative, download the *vim_upgrade_orchestrator.py curl -o vim_upgrade_orchestrator.py* file.

   For example:

   ```
   https://{username}:{password}@cvim-registry.com/mercury-releases/cvim40-rhel8-osp16/releases/{release number}/vim_upgrade_orchestrator.py
   ```

2. Execute the upgrade from the */root/* directory:

   ```
   $ cd /root/
   # chmod +x vim_upgrade_orchestrator.py
   #./vim_upgrade_orchestrator.py -i connected -s start_image_tag -n 4.2.3 [-y] # -y if no interactive mode is required.
   # start_image_tag value can be: 4.0.0
   ```


   - As the upgrade takes more than 6 hours, execute the upgrade in a VNC. 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 of upgrade.
   - During upgrade, do not run other Cisco VIM actions.
   - Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

3. After the upgrade, reboot the management node.
4. Plan on VNF downtime to complete the upgrade on compute nodes with VNFs via *--finish_upgrade_with_reboot* option provided in *vim_upgrade_orchestrator.py:*

   ```
   $ cd /root/
   # chmod +x vim_upgrade_orchestrator.py
   #./vim_upgrade_orchestrator.py -i connected -s 3.6.2 -n 4.2.3 --finish_upgrade_with_reboot <", separated compute nodes in quotes"> [-y] # -y if no interactive mode
   is required
   ```

   The computes that do not have any VNFs in ACTIVE state are automatically rebooted during software upgrade. Ensure that Ceph is HEALTH_OK state after each Hyper-converged or AIO node reboot.

   - No pod management operation is allowed ,until the reboot of all Cisco VIM nodes is successful.
   - In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

5. Run cloud sanity to ensure the cloud is up and running.

Upgrading Cisco VIM Using File-Based Image (from CVIM 3.4.x)

Upgrading Cisco VIM Using File-Based Image (from CVIM 3.4.x)

The following steps describes how to upgrade Cisco VIM 3.4.x to Cisco VIM 4.2.3 in an air-gapped mode without using USB:

1. Create a file-based image, for example, *artifacts-3-6-2.img,* for Cisco VIM 3.6.2 as per the steps listed in NFVI Installation File-Based Image using the *getartifacts.py* from the 3.6.2 release artifacts.
2. Copy the 3.6.2 file-based image to */var/* of the management node running Cisco VIM 3.4.x (x= 4, 5, or 6) from where the upgrade is planned.
3. Copy the *vim_upgrade_orchestrator.py* script available in Cisco VIM 3.4.6 artifacts in the `/root/` folder, to the management node of the pod running running Cisco VIM 3.4.x (x<=4, 5, or 6).
4. Execute the update from the */root/ location*:

```
# cd /root/
# chmod +x vim_upgrade_orchestrator.py
# ./vim_upgrade_orchestrator.py -f <file_based_image_absolute_path> -s start_image_tag -n 3.6.2 [-y]
# Use -y, if you don't want any interactive mode
# start_image_tag value can be: 3.4.x (x<=4 or 5 or 6)
```

- Upgrading Cisco VIM takes more than 6 hours, execute this upgrade in a VNC environment. If you do not have a VNC environment, execute the upgrade from KVM console of the management node.
- Do not run this command in background or with *nohup* option, as it results in upgrade failure.
- Do not run any other Cisco VIM actions during the upgrade. After the upgrade, start using the newly-created directory.
- Ensure that, *REQUESTS_CA_BUNDLE* is not set as an environment variable on the management node.

5. After the upgrade to Cisco VIM 3.6.2, identify the computes that have active VMs running on them, and plan for its reboot. Ensure that the built-in redundancy for controller and storage nodes exists. Rebooting these nodes is automated during software upgrade to Cisco VIM 3.6.2.

   The computes that does not have any VNFs in ACTIVE state are automatically rebooted during software upgrade.

   - Pod management operation is not allowed until the reboot of all Cisco VIM nodes is successful.
   - In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For more information on recovery, contact Cisco Support.

   To monitor and reboot the compute nodes through *ciscovim* CLI, refer the sections *Managing Reboot of Cisco VIM Nodes* and *Managing Reboot Status of Cisco VIM Nodes* under Managing Power and Reboot.
6. Reboot the Management Node before proceeding to the Next Step.
7. Prepare a Cisco VIM 4.2.3 file-based image with the corresponding 4.2.3 artifacts, in a RHEL/CentOS 7 staging server.
8. Run the *getartifacts.py* script provided in Cisco VIM 4.2.3 artifacts with - *U* option. Provide the registry username and password, the tag ID, and the location on the staging server running RHEL/CentOS 7. The following example shows how to create the file-based image for Cisco VIM 4.2.3:

```
chmod +x getartifacts.py
sudo ./getartifacts.py -t <tag_id> -u <username> -p <password> --file <file_based_image_abs_path> -U
```

Use *-U* option to download Rocky and Stein artifacts. This is applicable only for upgrade from 3.6.2 to 4.2.3.

9. Copy the 4.2.1 file-based image to */var/* of the management node, running in Cisco VIM 3.6.2.
10. Copy the *vim_upgrade_orchestrator.py* script available in Cisco VIM 4.2.3 artifacts in the `/root/` folder, to the management node of the pod running the 3.6.2 version. Execute the upgrade from the */root/* location:

```
# cd /root/
# chmod +x vim_upgrade_orchestrator.py
# ./vim_upgrade_orchestrator.py -f <file_based_image_absolute_path> -s 3.6.2 -n 4.2.3 [-y]
# Use -y, if you don't want any interactive mode
```

- As the upgrade takes more than 6 hours, execute the upgrade in a VNC. 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 of upgrade.
- During the upgrade, do not run other Cisco VIM actions.
- Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

> ⚠ Do not perform any pod operation that alters the state of the pod until the following steps are completed. Examples of pod operations that alter the state of the pod include: add compute, remove compute, add storage, remove storage, replace controller, regenerate secrets, reconfigure, update, rollback, and commit.

11. Copy the management node backup generated during the previous step to a remote server.
12. Restore the management node based on the backup saved in the previous step.
13. Plan for VNF downtime to complete leapp upgrade on the compute nodes. Use the *cvim_reboot_status.py* provided in */installer-4.2.3/tools/* directory to identify which compute nodes need reboot. Please copy *vim_upgrade_orchestrator.py* from the tools directory to /root/ and initiate the upgrade using the *--finish_upgrade_with_reboot* option or *-fuwr* provided in *vim_upgrade_orchestrator.py* for targeted compute reboots.

```
$ cd /root/
# chmod +x vim_upgrade_orchestrator.py
#./vim_upgrade_orchestrator.py -f <file_based_image_absolute_path> -s 3.6.2 -n 4.2.3 --
finish_upgrade_with_reboot option <", separated compute nodes in quotes"> [-y]
# Use -y, if no interactive mode is required.
```

14. For pods getting upgraded from Cisco VIM 3.x, Ceph volume continues to run in simple mode. Migrate the Ceph volume type to LVM.
    a. Use the tool provided in the *installer-4.2.3/tools/* directory to find out the Ceph nodes that are pending for migration to LVM as Ceph volume. To execute the tool, use the command:

    ```
    # cd /root/installer-4.2.3/tools
    # python ceph_migration_status.py
    ```

    b. Move each Ceph node from simple to LVM mode by performing a remove storage operation followed by an add storage operation for each storage node. Ensure that Ceph is in HEALTH_OK state before initiating each add/remove storage operation. For Micropod, you can migrate the Ceph volume type by performing a replace controller operation for each controller node.

    > ⚠ As Ceph migration takes long time, take subsequent maintenance window for this activity.

    c. After the node has been migrated, use the *ceph_migration_status.py* tool to verify Ceph volume type is LVM.
15. Execute cloud-sanity and Ceph health check to ensure that the cloud is completely up and running.

> ⚠ In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For more information on recovery, contact Cisco Support.

Upgrading Cisco VIM Using File-Based Image (from CVIM 4.0.0)

Upgrading Cisco VIM Using File-Based Image (from CVIM 4.0.0)

The following procedure describes how to upgrade to Cisco VIM 4.2.3 in an air-gapped mode without using USB.

1. Create a file-based image, for example, *artifacts-4-2-3.img,* for Cisco VIM 4.2.3 as per the steps listed in NFVI Installation File-Based Image using the *getartifacts.py* from the 4.2.3 release artifacts.
2. Copy the 4.2.3 file-based image to */var/* of the management node running Cisco VIM 4.0.0 from where the upgrade is planned.
3. Copy the *vim_upgrade_orchestrator.py* script available in Cisco VIM 4.2.3 artifacts in the `/root/` folder, to the management node of the pod running running 4.0.0.
4. Execute the update from the `/root/ location`:

```
# cd /root/
# chmod +x vim_upgrade_orchestrator.py
# ./vim_upgrade_orchestrator.py -f <file_based_image_absolute_path> -s 4.0.0 -n 4.2.3 [-y] # -y if you
don't want any interactive mode
```

After upgrade, start using the newly created directory.

- As the upgrade takes more than 6 hours, execute this update in a VNC. If you do not have a VNC environment, execute the same from KVM console of the management node.
- Do not run this command in background or with *nohup* option, as the upgrade will fail.
- Do not run any other Cisco VIM actions during upgrade. After upgrade, start using the newly created directory.
- Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

5. After the upgrade, reboot the management node.
6. Plan on VNF downtime to complete the upgrade on compute nodes with VNFs via *--finish_upgrade_with_reboot* option provided in *vim_upgrade_orchestrator.py:*

```
$ cd /root/
# chmod +x vim_upgrade_orchestrator.py
#./vim_upgrade_orchestrator.py -i connected -s 4.0.0 -n 4.2.3 --finish_upgrade_with_reboot <", separated
compute nodes in quotes"> [-y] # -y if no interactive mode
is required
```

The computes that do not have any VNFs in ACTIVE state are also automatically rebooted during software upgrade.

Ensure that Ceph is HEALTH_OK state after each Hyper-converged or AIO node reboot.

- No pod management operation is allowed ,until the reboot of all Cisco VIM nodes is successful.
- In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

7. Run cloud sanity to ensure the cloud is up and running.

Upgrading Cisco VIM Using USB (from CVIM 3.4.x)

Upgrading Cisco VIM Using USB (from CVIM 3.4.x)

To upgrade to Cisco VIM 4.2.3 in an air-gapped mode via USB(s), you need multiple USBs. Hence, it is recommended to use the file-based approach instead of USB-based approach.

The following procedure describes how to prepare the USB with 3.6.2 artifacts in a staging server:

> **Before you begin**
>
> This procedure requires a CentOS 7 staging server (VM, laptop, or UCS server) with a 128 GB USB 2.0 stick. You can download the VIM installation files using the staging server with Internet access (wired access is recommended), and save the files to a USB stick. You can use the USB stick to load the installation files onto the management node. As the size of the installation files comes to 24 GB approximately, downloading them to the USB stick might take several hours, depending on the speed of your Internet connection. Hence, plan accordingly and disable the CentOS sleep mode.

1. On the staging server, use yum to install the following packages:
   - PyYAML (yum install PyYAML)
   - python-requests (yum install python-requests)
2. Connect to Cisco VIM software download site using a web browser and login credentials provided by Cisco account representative, and download the *getartifacts.py* script from the external registry:

```
# download the new getartifacts.py file from CVIM 3.6.2 artifacts (see example below)
curl -o getartifacts.py -u '<username>:<password>'
https//cvim-registry.com/mercury-releases/cvim40-rhel7-osp16/releases/3.6.2/getartifacts.py

# Change the permission of getartificats.py
chmod +x getartifacts.py
```

3. Run the *getartifacts.py* script. The script formats the USB 2.0 drive and downloads the installation artifacts. Provide the registry username and password, the tag ID, and the USB partition on the staging server.

   For example, to identify the USB stick, execute the *lsblk* command before and after inserting the USB stick. The command displays a list of available block devices. You can find the USB stick location from the delta output, and provide the entire drive path in the *–d* option instead of any partition.

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

   > ⚠ Do not remove the USB stick during synchronization.

4. Verify the integrity of the downloaded artifacts and container images:

```
# create a directory
sudo mkdir -p /mnt/Cisco
# /dev/sdc is the USB stick, same as supplied in get artifacts.py python script
sudo mount /dev/sdc1 /mnt/Cisco
cd /mnt/Cisco
# execute the verification script
./test-usb
# failures will be explicitly displayed on screen. A sample success output is shown
# sample output of ./test-usb execution with 3.4.6 release
[root@mgmtnode Cisco]# ./test-usb
INFO: Checking the integrity of this USB stick
INFO: Checking artifact buildnode-K9-13401.iso
INFO: Checking artifact registry-2.6.2-13401.tar.gz
INFO: Checking required layers:
INFO: 605 layer files passed checksum.
[root@mgmtnode Cisco]#
```

5. 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
```

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

```
# /dev/sdc is the USB stick, 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 output of the above command will explicitly display the failure on the screen
```

7. After synchronization, unmount the USB stick:

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

8. Remove the USB stick from the staging server and insert the USB stick into the management node.
9. Insert the pre-populated USBs into the management node of the pod running 3.4.x (x=4 or 5 or 6).
10. Copy the *vim_upgrade_orchestrator.py* script from Cisco VIM 3.6.1 artifacts in the `/root/` folder, to the management node of the pod running running 3.4.x (x<=4 or 5 or 6).
11. Execute the upgrade from the */root/* location:

```
# cd /root/
# chmod +x vim_upgrade_orchestrator.py
# ./vim_upgrade_orchestrator.py -i disconnected -s start_image_tag -n 3.6.2 [-y] # -y if you don't want
any interactive mode
# start_image_tag value can be: 3.4.x (x<=4 or 5 or 6)
```


- As the upgrade takes more than 6 hours, execute this upgrade in a VNC. If you do not have a VNC environment, execute the same from KVM console of the management node.
- Do not run this command in background or with *nohup* option, as the upgrade will fail.
- Do not run any other Cisco VIM actions during upgrade. After upgrade, start using the newly created directory.
- Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

12. After the upgrade to Cisco VIM 3.6.2, reboot the management node.
13. After the upgrade to Cisco VIM 3.6.2, identify the computes that have active VMs running on them, and plan for its reboot. Ensures built-in redundancy for controller and storage nodes exists. The reboot of these nodes are automatic during software upgrade to Cisco VIM 3.6.2.

    The computes that does not have any VNFs in ACTIVE state are automatically rebooted during the software upgrade.

    - No pod management operation is allowed until the reboot of all Cisco VIM nodes is successful.
    - In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

    To monitor and reboot the compute nodes through *ciscovim* CLI, see the sections *Managing Reboot of Cisco VIM Nodes* and *Managing Reboot Status of Cisco VIM Nodes* under Managing Power and Reboot.

14. Prepare another USB with Cisco VIM 4.2.3 artifacts, by repeating Steps 1 through 9.
15. Run the Cisco VIM 4.2.3 *getartifacts.py* script with *-U* option. The script formats the USB drive and downloads the installation artifacts. Provide the registry username and password, the tag ID, and the USB partition on the staging server.

    For example, to identify the USB stick, execute the *lsblk* command before and after inserting the USB stick. The command displays a list of available block devices. You can find the USB stick location from the delta output, and provide the entire drive path in the *–d* option instead of any partition.

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


Do not remove the USB stick during synchronization. Use *-U* option to download Rocky and Stein artifacts. This is applicable only for 3.6.2 to 4.2.3 upgrade.

16. Copy the *vim_upgrade_orchestrator.py* script from Cisco VIM 4.2.3 artifacts in the `/root/` folder, to the management node of the pod running running 3.6.2. Execute the update from the */root/* location:

```
# cd /root/
# chmod +x vim_upgrade_orchestrator.py
# ./vim_upgrade_orchestrator.py -i disconnected -s 3.6.2 -n 4.2.3 [-y] # -y if you don't want any
interactive mode
```

- As the upgrade takes more than 6 hours, execute this upgrade in a VNC. If you do not have a VNC environment, execute the same from KVM console of the management node.
- Do not run this command in background or with *nohup* option, as the upgrade will fail.
- Do not run any other Cisco VIM actions during upgrade. After upgrade, start using the newly created directory.
- Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

17. Copy the management node backup being generated as part of the upgrade, to a remote server.
18. Restore the management node based on the backup saved in the previous step.
19. Plan for VNF downtime to complete leapp upgrade on compute nodes. Use the *cvim_reboot_status.py* provided in *installer-4.2.3/tools/* directory, to identify the compute nodes that need reboot. Copy *vim_upgrade_orchestrator.py* from the tools directory to */root/* and use the *--finish_upgrade_with_reboot* option or *-fuwr* provided in *vim_upgrade_orchestrator.py* for targeted compute reboots.

```
$ cd /root/
# chmod +x vim_upgrade_orchestrator.py
#./vim_upgrade_orchestrator.py -i disconnected -s 3.6.2 -n 4.2.3 --finish_upgrade_with_reboot <",
separated compute nodes in quotes"> [-y] # -y if no interactive mode is required.
```

20. For pods getting upgraded from Cisco VIM 3.x, Ceph volume continues to run in simple mode. Migrate the Ceph volume type to LVM.
    a. Use the tool provided in the *installer-4.2.3/tools/* directory to find out the Ceph nodes that are pending for migration to LVM as Ceph volume. To execute the tool, use the command:

    ```
    # cd /root/installer-4.2.3/tools
    # python ceph_migration_status.py
    ```

    b. Move each Ceph node from simple to LVM mode by performing a remove storage operation followed by an add storage operation for each storage node. Ensure that Ceph is in HEALTH_OK state before initiating each add/remove storage operation. For Micropod, you can migrate the Ceph volume type by performing a replace controller operation for each controller node.

    As Ceph migration takes long time, take subsequent maintenance window for this activity.

    c. After the node has been migrated, use the *ceph_migration_status.py* tool to verify Ceph volume type is LVM.
21. Execute cloud-sanity and ceph health check, to ensure that the cloud is completely up and running.

In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

Upgrading Cisco VIM Using USB (from CVIM 4.0.0)

Upgrading Cisco VIM Using USB (from CVIM 4.0.0)

The following procedure describes how to upgrade to Cisco VIM 4.2.3 in an air-gapped mode using USB.

Before you begin

This procedure requires a RHEL/CentOS 8.2 staging server (VM, laptop, or UCS server) with a 128 GB USB 2.0 stick. You can download the VIM installation files using the staging server with Internet access (wired access is recommended), and save the files to a USB stick. You can use the USB stick to load the installation files onto the management node. As the size of the installation files comes to 24 GB approximately, downloading them to the USB stick might take several hours, depending on the speed of your Internet connection. Hence, plan accordingly and disable the CentOS sleep mode.

1. On the staging server, use yum to install the following packages:
   - PyYAML (yum install PyYAML)
   - python-requests (yum install python-requests)
2. Connect to the Cisco VIM software download site using a web browser and login credentials provided by Cisco account representative, 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//cvim-registry.com/mercury-releases/cvim42-rhel8-osp16/releases/4.2.3/getartifacts.py
# Change the permission of getartificats.py
chmod +x getartifacts.py
```

3. Run the *getartifacts.py* script. The script formats the USB 2.0 drive and downloads the installation artifacts. Provide the registry username and password, the tag ID, and the USB partition on the staging server.

   For example, to identify the USB stick, execute the *lsblk* command before and after inserting the USB stick. The command displays a list of available block devices. You can find the USB stick location from the delta output, and provide the entire drive path in the *–d* option instead of any partition.

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

   Do not remove the USB stick during synchronization.

4. Verify the integrity of the downloaded artifacts and container images:

```
# create a directory
sudo mkdir -p /mnt/Cisco
# /dev/sdc is the USB stick, same as supplied in get artifacts.py python script
sudo mount /dev/sdc1 /mnt/Cisco
cd /mnt/Cisco
# execute the verification script
./test-usb
# failures will be explicitly displayed on screen. A sample success output is shown
# sample output of ./test-usb execution with 3.4.6 release
[root@mgmtnode Cisco]# /test-usb
INFO: Checking artifact vim_upgrade_orchestrator.py
INFO: Checking artifact mercury-installer-internal.tar.gz
INFO: Checking artifact ironic-images-internal.tar.gz
INFO: Checking artifact buildnode-internal-xxxxx.iso
INFO: Checking artifact registry-2.6.2-internal-xxxxx.tar.gz
INFO: Checking artifact buildnode-internal-30606.qcow2
INFO: Checking required layers:
INFO: 858 layer files passed checksum.
[root@mgmtnode Cisco]#
```

5. 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
```

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

```
# /dev/sdc is the USB stick, 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 output of the above command will explicitly display the failure on the screen
```

7. After synchronization, unmount the USB stick.

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

8. After synchronization, remove the USB stick from the staging server and insert the USB stick into the management node.
9. Insert the prepopulated USBs into the management node of the pod running Cisco VIM 4.0.0.
10. Copy the *vim_upgrade_orchestrator.py* script available in Cisco VIM 4.2.3 artifacts in the `/root/` folder, to the management node of the pod running 4.0.0.
11. Execute the upgrade from the `/root/` location*:*

```
# cd /root/
# chmod +x vim_upgrade_orchestrator.py
# ./vim_upgrade_orchestrator.py -i disconnected -s 4.0.0 -n 4.2.3 [-y] # -y if you don't want any
interactive mode
```

After upgrade, start using the newly created directory.

- As the upgrade takes more than 6 hours, execute this upgrade in a VNC. If you do not have a VNC environment, execute the same from KVM console of the management node.
- Do not run this command in background or with *nohup* option, as the upgrade will fail.
- Do not run any other Cisco VIM actions during upgrade. After upgrade, start using the newly created directory.
- Ensure that REQUESTS_CA_BUNDLE is not set as an environment variable on the management node.

12. After the upgrade, reboot the management node.
13. Plan on VNF downtime to complete the upgrade on compute nodes with VNFs via *--finish_upgrade_with_reboot* option provided in *vim_upgrade_orchestrator.py:*

```
$ cd /root/
# chmod +x vim_upgrade_orchestrator.py
#./vim_upgrade_orchestrator.py -i connected -s 3.6.2 -n 4.2.3 --finish_upgrade_with_reboot <", separated
compute nodes in quotes"> [-y] # -y if no interactive mode
is required
```

The computes that do not have any VNFs in ACTIVE state are also automatically rebooted during software upgrade.

Ensure that Ceph is HEALTH_OK state after each Hyper-converged or AIO node reboot.

- No pod management operation is allowed ,until the reboot of all Cisco VIM nodes is successful.
- In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

14. Run cloud sanity to ensure the cloud is up and running.

Cisco VIM Update

Cisco VIM Update

- Prerequisites and Assumptions
- Updating Cisco VIM in Running Cloud
- Updating Cisco VIM Using USB
- Updating Cisco VIM Using Network Installation

Prerequisites and Assumptions

Prerequisites/Assumptions for Update/Upgrade

During Cisco VIM update/upgrade, ensure that no interference exists with external monitoring services.
To handle the potential of docker corruption post repo/kernel update, you must follow the below steps during Cisco VIM update irrespective of the update methodology (connected or disconnected).

- If ACI fabric is integrated with NFVI, you must disable switch policy enforcement that can shutdown the ToR ports to which the NFVI server interfaces are connected, during software update. This is required as the MAC move for floating IP addresses happens during software update with L3 agent failover due to host package update and/or L3 agent container update. Multiple L3 agent failover can result in several MAC moves in a short period of time. Fast MAC moves over a five minute duration can potentially trigger N9K/ACI Mac-move policy violation, thereby causing an immediate shutdown of the server switch port.
- If HP servers are available in the pod targeted for Cisco VIM update, ensure that the rpm for the Integrated Lights Out (iLO) tool *(ssacli-4.17-6.0.x86_64.rpm)* is copied over to */root/installer-xxx/openstack-configs/* directory. For details on where to download this tool, see Support of Third-Party Compute (HP DL 360 Gen9)
- In case of upgrade failure, ensure that you do not try to recover the cloud by yourself. For help and recovery, contact Cisco Support.

Updating Cisco VIM in Running Cloud

Updating Cisco VIM in Running Cloud

Cisco VIM allows you to update all OpenStack and infrastructure services such as RabbitMQ, MariaDB, and HAProxy, and management node containers such as Cobbler, ELK, VMTP, and repo containers with almost no impact to the Cisco NFVI implementation.

Updates allows you to integrate Cisco VIM patch releases without redeploying the Cisco NFVI stack from the beginning. Updates have minimal service impact because they run serially component-by-component one node at a time. If an error occurs during an update, auto-rollback is triggered to return the cloud to the state that existed before the update is performed. After an update, you can check for any functional impacts on the cloud. If everything is fine, you can commit the update, to delete the old containers and old images from the nodes. If you see any functional cloud impact, you can perform a manual rollback to start the old containers again.

Before you begin a container update, ensure that:

- Updates are not supported for registry-related containers and authorized_keys.
- You cannot rollback the repo containers on the management node to an older version once updated, as the rollback deletes the node packages and causes the cloud to destabilize.
- Cloud sanity check is performed before the update is started, to prevent double-faults.

The following table provides various options to update OpenStack using Cisco VIM. The Internet options refer to management node connectivity to the Internet. If your management server lacks Internet access, you must have a staging server with Internet access to download the Cisco VIM installation artifacts to a USB stick. Ensure that you select either of the two options and stay with it for the entire lifecycle of the pod.

	Without Cisco VIM Unified Management	**With Cisco VIM Unified Management**
Without Internet	• Prepare the USB on a staging server • Plug the USB into the management node. • Follow the update procedure without Internet.	• Prepare the USB on a staging server • Plug the USB into the management node. • Follow the update procedure without Internet.
With Internet	• Download the .tgz file from the registry. • Follow the update steps with Internet.	• Download the .tgz file from the registry. • Follow the update steps with Internet procedure.

Updating Cisco VIM Using USB

Updating Cisco VIM Software Using USB

The following procedure describes how to load the Cisco VIM installation files onto a Cisco NFVI management node that does not have Internet access. Installation files include: *buildnode-K9.iso, buildnode-K9.qcow2, mercury-installer.tar.gz*, *ironic-images-K9.tar.gz, registry-2.6.2-<xxxxx>.tar.gz,* vim_upgrade_orchestrator.py and *all_check_sum_file.sign* file.

Before you begin

This procedure requires a CentOS 8.2 staging server (VM, laptop, or UCS server) with a 64 GB USB drive. You can save the VIM installation files on a USB stick and then use the USB stick to load the installation files onto the management node. The installation files are around 24 GB in size. Downloading them to the USB drive might take several hours depending on the speed of your Internet connection. Before you begin, ensure that you disable the CentOS sleep mode

1. On the staging server, use yum to install the following packages:
   - PyYAML (yum install PyYAML)
   - python-requests (yum install python-requests
2. Connect to the Cisco VIM software download site using a web browser and login credentials provided by your account representative 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//cvim-registry.com/mercury-releases/cvim42-rhel8-osp16/releases/<4.2.2>/getartifacts.py
# Change file permission of getartifacts.py
chmod +x getartifacts.py
```

3. Run the *getartifacts.py* script. The script formats the USB 2.0 drive and downloads the installation artifacts. Provide the registry username and password, the tag ID, and the USB partition on the staging server. For example:

   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 delta helps to find the USB drive location. Provide the entire drive path in the –d option, instead of any partition.

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

   Do not remove the USB drive during synchronization.

4. Verify the integrity of the downloaded artifacts and the container images:

```
# create a directory
sudo mkdir -p /mnt/Cisco
# /dev/sdc is the USB drive, same as supplied in get artifacts.py python script
sudo mount /dev/sdc1 /mnt/Cisco
cd /mnt/Cisco
# 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.x release
[root@mgmtnode Cisco]# /test-usb
INFO: Checking the integrity of this USB stick
INFO: Checking artifact buildnode-K9-13401.iso
INFO: Checking artifact registry-2.6.2-13401.tar.gz
INFO: Checking required layers:
INFO: 605 layer files passed checksum.
[root@mgmtnode Cisco]#
```

5. 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
```

6. Mount the USB and then run the *test-usb* command to validate whether 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 is explicitly displayed the same on the screen
```

7. After the synchronization is complete, unmount the USB drive:

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

8. Remove the USB stick from the staging server and then insert it into the management node.
9. Complete the following steps to import the Cisco NFVI installation artifacts onto the management node:
    a. Identify the USB on the management node:

    ```
    blkid -L Cisco-VIM
    ```

    b. Mount the USB device on the management node:

    ```
    mount < /dev/sdc > /mnt/
    mkdir /root/cvim-update-media/
    cd /root/cvim-update-media/
    ```

    c. Extract the *import_artifacts.py* script:

    ```
    tar --no-same-owner -xvzf /mnt/Cisco/mercury-installer.tar.gz
    ```

    d. Unmount the USB device:

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

    e. Import the artifacts:

    ```
    cd /root/cvim-update-media/installer-<4.x.x>/tools/
    ./import_artifacts.sh
    ```

    f. Change directory and remove `/root/cvim-update-media`:

    ```
    cd /root/
    rm -fr /root/cvim-update-media/
    ```

10. Execute Cisco VIM software update:

```
cd /root/
ciscovim update --file /var/cisco/artifacts/mercury-installer.tar.gz
```

After the update is complete, use only the newly created directory unless a rollback is planned.

11. Commit the update by running the following command:

```
ciscovim commit # once commit is done, use a new workspace
```

12. To revert the update changes before entering the *commit* command, use the following command:

```
ciscovim rollback # once rollback is done, use the older workspace
```

> ⚠ Do not run any other Cisco VIM actions when the update is in progress.

In Cisco VIM, if updates bring in Kernel changes, the reboot of the compute node with VNFs in ACTIVE state is postponed. This is done to mitigate the unpredictability of data plane outage when the compute nodes go for a reboot for the kernel changes to take effect, during rolling upgrade.

At the end of *ciscovim* update, the Cisco VIM orchestrator displays the following message on the console and logs:

13. After the kernel update on management node, reboot the management node for the kernel changes to take effect.

Management Node reboot

```
[root@test ~]# reboot
login as: root
root@10.84.66.50's password:
Last login: Tue Sep 21 10:06:14 2021 from 10.24.91.185
```

14. At the end of *ciscovim* update, the Cisco VIM orchestrator displays the following message on the console and logs:

```
Compute nodes require reboot Kernel updated
<compute_1_with_VM_running>
<compute_3_with_VM_running>
<compute_4_with_VM_running>
<compute_12_with_VM_running>
```

15. After the Kernel update on management node, reboot the compute nodes before proceeding, based on the *'ciscovim reboot-status'*.

Reboot-status

```
[root@test ~]# ciscovim reboot-status
Fetching Nodes that require a Reboot
+--------------------+-----------------+
| Node Name                | Reboot Required |
+--------------------+-----------------+
| NFVI-compute-node1| Yes        |
| NFVI-compute-node2| Yes         |
| NFVI-control-node1| NO        |
| NFVI-control-node2| NO        |
| NFVI-control-node3| NO        |
| NFVI-storage-node1| NO        |
| NFVI-storage-node2| NO        |
| NFVI-storage-node3| NO        |
+--------------------+-----------------+
```

16. The logs for this run are available in,

```
<mgmt._ip_address>:/var/log/mercury/<UUID>
```

For Micropod, if the VM landed on the server has dual roles (control and compute), reboot the server and run ciscovim *cluster-recovery*

> ⚠ As the redundancy in controller and storage nodes are built into the product, the reboot of those nodes are automated during the software update.
>
> The computes that does not have any VNFs in ACTIVE state are automatically rebooted during software update. To monitor and reboot the compute nodes through ciscovim cli, see the sections Managing Reboot of Cisco VIM Nodes and Managing Reboot Status of Cisco VIM Nodes.
>
> No pod management operation is allowed until the reboot of all Cisco VIM nodes are successful.

Updating Cisco VIM Using Network Installation

Updating Cisco VIM Software Using Network Installation

1. From the download site that is provided by your Cisco account representative, download the *mercury-installer.gz*:

   ```
   curl -o mercury-installer.tar.gz -u '<username>:<password>' https://cvim-registry.com/mercury-releases
   /cvim34-rhel8-osp16/releases/{release number}/mercury-installer.tar.gz
   ```

   The link to the tar ball preceding is an example.
2. Execute the update from the old working directory:

   ⚠ Do not run any other Cisco VIM actions during an update.

   ```
   cd /root/installer-<tagid>
   ciscovim update --file /root/mercury-installer.tar.gz
   ```

   After the update is complete, use the newly created directory unless a rollback is planned.
3. Commit the update by running the following command:

   ```
   ciscovim commit
   ```

4. To revert the update changes before entering the commit command, enter:

   ```
   ciscovim rollback # and then use older workspace
   ```

   If the updates bring in Kernel changes, the reboot of the compute node with VNFs in ACTIVE state is postponed. This is done to mitigate the unpredictability of data plane outage when the compute nodes go for a reboot for the kernel changes to take effect, during rolling update.
   At the end of *ciscovim* update, the Cisco VIM orchestrator displays the following message on the console and logs:

   ```
   Compute nodes require reboot Kernel updated
   <compute_1_with_VM_running>
   <compute_3_with_VM_running>
   <compute_4_with_VM_running>
   <compute_12_with_VM_running>
   ```

   After the Kernel update on the management node, reboot the compute node before proceeding further. The logs for this run are available in *<mgmt._ip_address>:/var/log/mercury/<UUID>*

   ⚠
   - The redundancy in controller and storage nodes are built into the product. The reboot of those nodes are automated during software update.
   - The computes that does not have any VNFs in ACTIVE state, are automatically rebooted during software update. To monitor and reboot the compute nodes through *ciscovim* CLI, see Managing Power and Reboot > *Managing Reboot of Cisco VIM Nodes* and *Managing Reboot Status of Cisco VIM Nodes*
   - No pod management operation is allowed until the reboot of all Cisco VIM nodes is successful.

Administration

Administration

- Managing Pods
- Managing Scheduler Filters
- Monitoring Cisco NFVI Health
- Assessing Cisco NFVI Status
- Service Catalog URL
- Checking Network Connections
- General Scheme of Enabling Optional Services
- Managing VIM Administrators
- Managing Power and Reboot
- Read-only OpenStack Role
- Cisco VIM Advanced OpenStack Policy
- Security
- Storage
- Monitoring with CVIM-MON
- CVIM Pod Migration from Micro to Full

Managing Pods

Managing Cisco NFVI Pods

- NUMA Pinning
- Management Storage IP
- Pod Recovery
- Managing Hosts
- Identifying Installer Directory
- General Guidelines

NUMA Pinning

NUMA Pinning

From release Cisco VIM 3.4.0, NUMA pinning of VMs is supported. To make use of this feature, you must add *hw:pin_to_numa* in their VM's flavor, and set its value to 0 or 1. When you spawn VM with that flavor, the VM uses only the host CPUs from the NUMA that is specified in the flavor.

Management Storage IP

Management Storage IP Support

From release Cisco VIM 3.2.2 onwards, along with server management IP (v4 and v6), you can also statically define the server storage IP during pod installation. To help in the transition, Cisco created a tool that helps to update the setup_data with the server storage IP information of a pod that is already up and running.
To run the utility, ensure that the pod is up and running Cisco VIM 3.2.2 or later:

```
#cd installer-xxx/tools/
#./update_static_addrs.sh
```

On success, a message *Static Address updates in setup_data.yaml complet*e is displayed at the end of the run.

Pod Recovery

Pod Recovery

This section describes the recovery processes for Cisco NFVI control node and the pod that is installed through Cisco VIM. For recovery to succeed, a full Cisco VIM installation must be done in the past. The recovery is caused by a failure of one or more controller services such as Rabbit MQ, MariaDB, and other services. The management node must be up and running, and all the nodes must be accessible through SSH without passwords from the management node. You can also use this procedure to recover from a planned shutdown or accidental power outage.

For control node recovery, use the following command:

```
ciscovim cluster-recovery
```

The control node recovers after the network partition is resolved.

> ⚠ The database sync between controller nodes might take time, which can result in cluster recovery failure. In that case, wait for some time for the database sync to complete and then rerun cluster-recovery.

To make sure that Nova services are good across compute nodes, execute the following command:

```
# source /root/openstack-configs/openrc
# nova service-list
```

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

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

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

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

In case of a complete pod outage, you must follow a sequence of steps to bring the pod back. The first step is to bring up the management node, and check if the management node containers are up and running using the *docker ps –a* command. After you bring up the management node, bring up all the other pod nodes. Ensure that each node is reachable through *password-less SSH* from the management node. Verify that no network IP changes have occurred. You can get the node SSH IP access information from */root/openstack-config/mercury_servers_info*.

Execute the following command sequence:

1. Check the setup_data.yaml file and runtime consistency on the management node:

   ```
   # cd /root/installer-<tagid>/tools
   # ciscovim run --perform 1,3 -y
   ```

2. Execute the cloud sanity using *ciscovim* command:

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

3. To view the results of *cloud-sanity,* use the command:

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

4. Check the status of the REST API server and the corresponding directory where it is running:

   ```
   # cd/root/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>/
   ```

5. If the REST API server is not running from the right installer directory, execute the following to get it running from the correct directory:

```
cd/root/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>/
```

6. Verify whether the Nova services are good across the compute nodes by executing the following command:

```
# source /root/openstack-configs/openrc
# nova service-list
```

7. If *cloud-sanity* fails, execute *cluster-recovery* (*ciscovim cluster-recovery*), and then re-execute the *cloud-sanity* and nova *service-list* steps as listed above. The recovery of compute and OSD nodes requires network connectivity and reboot, so that they can be accessed using SSH without password from the management node.
8. To shut down, bring the pod down in the following sequence:

    a. Shut down all VMs, then all the compute nodes. It should be noted that graceful shut down of VMs is important. Check the VM status from the output of o*penstack server list --all-projects*, which must show that all VMs are in SHUTOFF State before you proceed.
    b. Shut down all compute node(s).
    c. Shut down all storage nodes serially. Before proceeding to next step, ensure that you wait until the storage node shutdown is completed.
    d. Shut down all controllers, but one at a time. Before proceeding to next step, wait for the controller node shutdown to complete.
    e. Shut down the management node.
    f. Shut down the networking gears.

    ⚠ To shut down a node, SSH to the node or connect to CIMC KVM console and issue the shutdown command *#shutdown -h now*

9. Bring the nodes up in reverse order, that is,

    a. Bring up the networking gears.
    b. Bring up or power ON the management node via CIMC/BMC/ILO as part of the vendor type.
    c. Bring up or power ON the control nodes via CIMC/BMC/ILO as part of the vendor type.
    d. Bring up or power ON the storage nodes and control nodes via CIMC/BMC/ILO as part the vendor type.
    e. Wait until the Ceph health reports are fine and then proceed to next step.
    f. Bring up or power ON the compute nodes via CIMC/BMC/ILO as part of the vendor type.

    ⚠ Ensure that each node type is completely booted up, before you move on to the next node type.

10. Run the cluster recovery command, to bring up the pod post power-outage:

```
# ciscovim cluster-recovery
```

11. Run the cloud sanity using the command:

```
# ciscovim cloud-sanity
```

12. Execute docker cloudpulse check to ensure that all containers are up:

```
cloudpulse run --name docker_check
```

13. Validate the Cisco API server by running the following command:

```
# ciscovim run --perform 1,3 -y
```

14. Bring up all VMs and validate if they are all up (not in shutdown state). If any of the VMs are in down state, bring them up using the Horizon dashboard.

    ⚠ ciscovim *cluster-recovery* programmatically attempts to recover a pod.

If *cluster-recovery* fails to recover for some reason, use *force* option as given below*:*

```
[root@hiccup ~]# ciscovim help cluster-recovery
usage: ciscovim cluster-recovery [--run-disk-checks] [-f] [-y]
Recover the Openstack cluster after a network partition or power outage
Optional arguments:
  --run-disk-checks  Flag to run disk-checks during cluster-recovery
  -f, --force        Option to perform force cluster-recovery
  -y, --yes          Yes option to perform the action
```

Managing Hosts

Managing Hosts

In Cisco VIM, a node can participate in multiple roles based on the pod type. The following rules apply for hardware management of a node:

1. If a node is a Micropod node that acts as controller, compute, and Ceph, the node can only go through the action of replace controller for its swap. You can perform this action on one node at a time.
2. If a node is a hyper-converged node (that is, acting as both compute and Ceph), the node is treated as a ceph node from hardware management point of view and the node can only go through the action of addition or removal of Ceph. This action can be done only on one node at a time.
3. If a node is a standalone compute node, the node can only go through the action of addition or removal of compute. You can add or remove multiple nodes at a time, but you cannot operate the pod with zero compute at any given time.
4. If a node is a dedicated controller node, the node can only go through the action of replace controller for its swap. This action can be done only on one node at a time.
5. If a node is a dedicated Ceph node, the node can only go through the action of addition or removal of Ceph. This action can be done only on one node at a time and you cannot have a pod with less than two node Ceph at a time.
6. For nano pod, host management is not possible.

For pod management actions, see the following table. If you log in as root, manually change the directory to */root/installer-xxx* to get to the correct working directory for these Cisco NFVI pod commands.

Action	Steps	Restrictions

Remove block_storage or compute node	1. Remove the node information from the ROLES and SERVERS section of the setup_data.yaml file for the specific node.
	2. Enter one of the following commands:
	For compute nodes:
	`ciscovim remove-computes --setupfile ~/MyDir/my_setup_data.yaml <"compute-1,compute-2"> [--force]`
	For storage nodes:
	`ciscovim remove-storage --setupfile ~/MyDir/my_setup_data.yaml <"storage-1"> [--force]`
		• You can remove multiple compute nodes, but only one storage at a time.
		• The pod must have a minimum of one compute and two storage nodes after the removal action.
		• In Cisco VIM, the number of Ceph OSD nodes vary from 3 to 20. You can remove one OSD node at a time as part of the pod management.
		⚠
		1. On a Micropod or edge pod expanded with standalone computes, only the standalone compute nodes can be removed. Pod management operation for storage node is not supported for Micro or edge pod.
		2. Compute management operations are not supported for hyper-converged nodes.
		3. In UMHC or NGENAHC pod, if a VM is running on the storage node, remove-storage operation fails in pre-validation and gives a warning to you about the running VM's. Use force option to forcefully remove the storage node.
		4. In Ceph pod, pod management operations for compute is not supported. Removal of storage node is only allowed for servers that are exclusively available with cephosd roles.

Add block_storage or compute node	1. Add the node information from the ROLES and SERVERS section of the setup_data.yaml file for the specific node.
	2. Enter one of the following commands:
	For compute nodes:
	`ciscovim add-computes --setupfile ~/MyDir/my_setup_data.yaml <"compute-1,compute-2"> [--skip_vmtp]`
	For storage nodes:
	`ciscovim add-storage --setupfile ~/MyDir/my_setup_data.yaml <"storage-1"> [--skip_vmtp]`
		• You can add multiple compute nodes and only one storage node at a time.
		• The pod must have a minimum of one compute and two storage nodes before the addition action.
		• The number of ceph OSD nodes can vary from 3 to 20. You can add one OSD node at a time as part of the pod management.
		⚠
		1. On a Micro or edge pod expanded with standalone computes, you can add only the standalone compute nodes. Pod management operation for storage node is not supported.
		2. In hyper-converged mode, compute management operations are not supported for hyper-converged nodes.

Replace controller node	1. If the controller node is in a rack-based deployment (UCS C-series or Quanta-based pod), update the CIMC information node in the SERVERS section of the *setup_data.yaml* file for the specific node. 2. Enter the following commands: `ciscovim replace-controller --setupfile ~/MyDir/my_setup_data.yaml <"control-1"> [--force] [--skip_vmtp]`	• You can replace only one controller node at a time. The pod can have a maximum of three controller nodes. • The replace controller node operation is supported in Micropod. ⚠ 1. While replacing the controller node, the IP address and hostname are reused. So, do not update any controller information other than CIMC access and hardware information for C-series. 2. For micro, edge and Ceph pods, this operation is supported on the AIO (all in one), compute-control, and ceph control nodes, respectively. In a micro or edge pod, if a VM is running on the controller node, the replace controller operation fails during prevalidation and gives a warning about the running VM's. Use *force* option to forcefully replace the controller.

⚠ For remove-compute, remove-storage, and replace-controller operations, Cisco VIM attempts to change the host OS boot mode to single user and power-off those server(s) to avoid the creation of duplicate IP address on the management and storage networks, and to prevent the remove/replace server from connecting back to the deployment.

If the remove/replace server is powered-off before triggering Cisco VIM remove/replace operation, then the server must not be brought back on the network as it may create duplicate IP address when connected to deployment with valid credentials.

To power up those removed/replaced server(s):
- isolate the server on the network (shutdown of corresponding ToR switchports or unplug the network cables)
- power up the server
- delete the virtual drive from CIMC
- power-cycle the server
- bring the server on the network (no shutdown of corresponding ToR switchports or plug back the network cables)

When you add a compute or storage node to a rack-based pod (UCS C-series or Quanta), you can increase the management/provision address pool. Along with server information, these are the only items you can change in the *setup_data.yaml* file after the pod is deployed.

To make changes to the management or provisioning sections, you must not change the existing address block as defined on Day 0. You can add only to the existing information by adding new address pool block(s) of address pool as shown in the following example:

```
NETWORKING:
::
networks:
-
vlan_id: 99
subnet: 172.31.231.0/25
gateway: 172.31.231.1
## 'pool' can be defined with single ip or a range of ip pool:
- 172.31.231.2, 172.31.231.5 - IP address pool on Day-0
- 172.31.231.7 to 172.31.231.12 - IP address pool ext. on Day-n
- 172.31.231.20
segments:
## CIMC IP allocation. Needs to be an external routable network
- cimc
-
vlan_id: 2001
subnet: 192.168.11.0/25
gateway: 192.168.11.1
rt_prefix: < Local to POD > #optional, only for segment management/provision, storage,
tenant and ToR-type NCS-5500
rt_suffix: < Region>:< pod_region_number > #optional, only for segement
management/provision, storage, tenant and ToR-type NCS-5500
## 'pool' can be defined with single ip or a range of ip
pool:
- 192.168.11.2 to 192.168.11.5 - IP address pool on Day-0
- 192.168.11.7 to 192.168.11.12  IP address pool on day-n
- 192.168.11.20  IP address pool on day-n
segments:
## management and provision goes together
- management
- provision
```

You can change only the IP address pool in the networking space of the specified networks management/provision. The overall network must have enough address space to accommodate for future enhancement on Day 0. After making the changes to servers, roles, and the corresponding address pool, you can execute the add compute/storage CLI shown above to add new nodes to the pod.

For C-series M5 pods, with Cisco NCS 5500 as ToR with splitter cable connection onto the server, you have to adjust the entry for the splitter_opt_4_10 in respective SWITCHDETAILS for the Cisco NCS 5500 ToR pairs with the server (cimc_ip), and connection (tor_info, dp_tor_info, sriov_tor_info) details.

For example, to add compute or storage with Cisco NCS 5500 as ToR with splitter cable, add the following entry to the respective Cisco NCS 5500:

```
TORSWITCHINFO:
CONFIGURE_TORS: true # Mandatory
TOR_TYPE: NCS-5500 # Mandatory
ESI_PREFIX:91.<Pod_number>.<podregion_number>.00.00.00.00 #optional - only for NCS-5500
SWITCHDETAILS: -
hostname: <NCS-5500-1> # hostname of NCS-5500-1
username: admin
password: <ssh_password of NCS-5500-1>
...
splitter_opt_4_10: 'FortyGigE<C/D/X/Y>,HundredGigE<E/F/A/B>, ...' # Optional for NCS-5500,
only when
splitter is needed on per switch basis (i.e. the peer switch may or may not have the
entry)
ESI_PREFIX:91.<Pod_number>.<podregion_number>.00.00.00.00 #optional for NCS-5500 only
```

To remove a compute or a storage, delete the respective information. To replace the controller, swap the relevant port information from which the splitter cables originate.

For replace controller, you can change only a subset of the server information. For C-series, you can change the server information such as CIMC IP, CIMC Username, CIMC password, rack_id, and tor_info.

Identifying Installer Directory

Identifying Installer Directory

If you are an administrator and want to use CLI to manage the pods, you must know the location of the installer directory. To identify the installer directory of a pod, execute the following commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# ls -lrt | grep openstack-configs
lrwxrwxrwx. 1 root root 38 Mar 12 21:33 openstack-configs ->
/root/installer-<tagid>/openstack-configs
```

From the output, you can understand that the OpenStack-configs is a symbolic link to the installer directory. Verify if the REST API server is running from the same installer directory location, by executing the following commands:

```
# 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>/
```

General Guidelines

General Guidelines

You can perform OpenStack management operations on Cisco NFVI pods including addition and removal of Cisco NFVI compute and Ceph nodes, and replacement of controller nodes. Each action is mutually exclusive. You can perform only one pod management action at a time. Before you perform a pod action, ensure that the following requirements are met:

- The node is part of an existing pod.
- The node information exists in the *setup_data.yaml* file, if the pod management task is removal or replacement of a node.
- The node information does not exist in the *setup_data.yaml* file, if the pod management task is to add a node.

For more information on operations that can be performed on pods, see Managing Hosts.

The *setup_data.yaml* file is the only user-generated configuration file that is used to install and manage the cloud. While many instances of pod management indicate that the *setup_data.yaml* file is modified, the administrator does not update the system generated *setup_data.yaml* file directly.

To avoid translation errors, ensure that you do not copy and paste the commands from the documents to the Linux CLI.

To update the *setup_data.yaml* file, follow the below steps:

1. Copy the setup data into a local directory:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
```

2. Update the setup data manually:

```
[root@mgmt1 ~]# vi my_setup_data.yaml (update the targeted fields for the setup_data)
```

3. Run the reconfiguration command:

```
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

   You can edit and enable a selected set of options in the *setup_data.yaml* file using the reconfigure option. After installation, you can change the values of the feature parameters. Unless specified, Cisco VIM does not allow you to undo the feature configuration. During reconfiguration, all other pod management activities are disabled. Post-update, the normal cloud management operations commence. If reconfigured OpenStack services fail, all the subsequent pod management operations are blocked. To resolve the situation through CLI, contact Cisco TAC:

   The following table shows the list of features that can be enabled or whose values can be changed post pod deployment:

Features Enabled Post Pod Deployment	**Comment (Repeated Redeployment Value)**
Optional OpenStack services	• Heat: OpenStack Orchestration (False) • LDAP: Works only with Keystone v3. Full or partial reconfiguration can be done. Except for domain, all attributes are reconfigurable. (Partial) • Ironic: Baremetal workload post installation (False) • Container: Cloud-native workload (False) • Barbican (True)
Pod monitoring	CVIM-MON: Monitoring host and service level with or without ui_access (False).
Export of Elasticsearch, Fluentd, and Kibana (EFK) logs to external syslog server	Reduces single point of failure on management node and provides data aggregation .
NFS for Elasticsearch snapshot	Uses NFS mount point for Elasticsearch snapshot, so that the management node disk is not full.
Admin source networks	Allowed list filter for accessing management node admin service over IPv4 or IPv6 (True).
NFVBench	Tool to help measure cloud performance. The management node needs a dedicated 10G/40G Intel NIC (4x10G 710 or 2x40G XL710 Intel NIC) (False)
EFK settings	Enables you to set EFK rotation frequency and size (True).

OpenStack service password	Implemented for security reasons, so that OpenStack passwords can be reset on-demand (True).
CIMC Password Reconfigure post installation	Implemented for security reasons, so that CIMC passwords for C-series pod can be reset on-demand (True).
TENANT_VLAN_RANGES and PROVIDER_VLAN_RANGES	You can increase or decrease the tenant and provider VLAN ranges on a pod that is up and running. It gives flexibility in network planning (True).
DHCP reservation for VM's MAC addresses	Allow DHCP reservation for virtual machine MAC addresses, to get the same IP address always regardless of the host hypervisor or operating system they are running (True).
Enable TRUSTED_VF on a per (SR-IOV) compute basis	Allows virtual functions to become trusted by the physical function and to perform some privileged operations such as enabling VF promiscuous mode and changing VF MAC address of the guest (True).
Support of multiple external syslog servers	Ability to offload the OpenStack logs to a maximum of four external syslog servers post-installation (True).
Replacement of failed APIC hosts and addition of more leaf nodes	Ability to replace failed APIC hosts, and add more leaf nodes to increase the fabric influence (True).
Make NetApp block storage endpoint secure	Ability to move the NetApp block storage endpoint from clear to TLS post deployment (False).
Auto-backup of management node	Ability to enable/disable auto-backup of management node (True).
VIM Admins	• Ability to configure non-root Cisco VIM administrators (True). • Ability to configure Cisco VIM admins authenticated by LDAP (True).
EXTERNAL_LB_VIP_FQDN	Ability to enable TLS on external_vip through FQDN (False).
EXTERNAL_LB_VIP_TLS	Ability to enable TLS on external_vip through an IP address (False).
http_proxy and/or https_proxy	Ability to reconfigure http and/or https proxy servers (True).
Admin privileges for VNF Manager (ESC) from a tenant domain	Ability to enable admin privileges for VNF manager (ESC) from a tenant domain (False).
SRIOV_CARD_TYPE	Mechanism to switch between 2-X520 and 2-XL710 as an SRIOV option for Cisco VIC NIC settings at a global level and per compute level through reconfiguration. In the absence of per compute and global level, X520 card type is set by default (False).
Reset of KVM console passwords for servers	Aids to recover the KVM console passwords for servers (True).
Horizon behind NAT or with DNS alias(es)	Ability to host Horizon behind NAT or with DNS aliases (False).
Login banner for SSH sessions	Supports configurable login banner for SSH sessions (True).
Ability to add Layer 3 BGP session	Ability to switch BGP sessions from Layer 2 to Layer 3 in the presence of VXLAN configuration (False).
Add/remove of head-end-replication option	Ability to add or remove head-end-replication option, in the presence of VXLAN configuration (True).
Enabling Cloud Settings	Ability to set horizon and keystone settings as a reconfigurable options (True).
Vault	Ability to enable vault on Day 2 (False).
Identity, Policy and Audit (IPA) enablement	Ability to enable IPA as a Day 2 option (False).
Enable RAM and CPU Allocation Ratio	Ability to enable RAM and CPU allocation ratio on a per server basis as a Day 2 option (True).
NTP configuration change	NTP information update listed in NETWORKING section (maximum of 4 entries) (True).
DNS configuration change	Domain Name Server (DNS) information update listed in NETWORKING section (True).
CEPH_TUNING	Ability to configure Ceph parameters for faster OSD failure detection on Day 2 (True).
libvirt write-through	Ability to change libvirt from write-back to write-through and vice versa on Day 2 (True).
Enabling ObjectStore via Swift	Ability to enable Object Store via swift on Day 2 (False).
CUSTOM_MANUFACTURER_TAG	Ability to enable CUSTOM_MANUFACTURER_TAG on Day 2 (True).

Disable certain Nova supported actions	Ability to enable/disable VM live migration, snapshot, crash dump, and console (True).
CINDER_RBD_FLATTEN_VOLUME_FROM_SNAPSHOT	Ability to flatten volumes created from snapshots to remove dependency from volume to snapshot (True).

All reconfiguration features contain repeated re-deployment option set to True or False. If repeated re-deployment is set to True, the values associated to the feature can be changed, otherwise, the feature can be deployed only once. Un-installation of the feature is only supported as an exception.

Managing Scheduler Filters

Managing Scheduler Filters and User Data

OpenStack Nova is an OpenStack component that provides on-demand access to compute resources by provisioning large networks of virtual machines (VMs). In addition to the standard Nova filters, Cisco VIM supports the following additional scheduler filters:

- ServerGroupAffinity Filter—Ensures that an instance is scheduled onto a host from a set of group hosts. To use this filter, you must create a server group with an affinity policy and pass a scheduler hint using group as the key and the server group UUID as the value. Use the **nova** comm and-line tool and the **--hint** flag. For example:

  ```
  $ nova server-group-create --policy affinity group-1
  $ nova boot --image IMAGE_ID --flavor 1 --hint group=SERVER_GROUP_UUID server-1
  ```

- ServerGroupAntiAffinityFilter—Ensures that each group instance is on a different host. To use this filter, you must create a server group with an anti-affinity policy and pass a scheduler hint, using group as the key and the server group UUID as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova server-group-create --policy anti-affinity group-1
  $ nova boot --image IMAGE_ID --flavor 1 --hint group=SERVER_GROUP_UUID server-1
  ```

- SameHostFilter—Within an instance set, it schedules one instance on the same host as another instance. To use this filter, pass a scheduler hint using **same_host** as the key and a list of instance UUIDs as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova boot --image IMAGE_ID --flavor 1 --hint same_host=INSTANCE_ID server-1
  ```

- DifferentHostFilter—Within an instance set, schedules one instance on a different host than another instance. To use this filter, pass a scheduler hint using **different_host** as the key and a list of instance UUIDs as the value. The filter is the opposite of SameHostFilter. Use the **nova**comman d-line tool and the **--hint** flag. For example:

  ```
  $ nova boot --image IMAGE_ID --flavor 1 --hint different_host=INSTANCE_ID server-1
  ```

In addition to scheduler filters, you can set up user data files for cloud application initialization. A user data file is a special key in the metadata service that holds a file that cloud-aware applications in the guest instance can access. For example, one application that uses user data is the cloud-init system which is an open-source package that is available on various Linux distributions to handle early cloud instance initialization. The typical use case is to pass a shell script or a configuration file as the user data during the Nova boot, for example:

```
$ nova boot --image IMAGE_ID --flavor 1 --hint user-data FILE_LOC server-1
```

Monitoring Cisco NFVI Health

Monitoring Cisco NFVI Health with CloudPulse

You can query the state of various Cisco NFVI OpenStack endpoints using an OpenStack health-checking tool called CloudPulse. By default, the tool automatically polls OpenStack Cinder, Glance, Nova, Neutron, Keystone, Rabbit, Mariadb, and Ceph every four minutes. However, you can use a CLI REST API call from the management node to get the status of these services in real time. You can integrate the CloudPulse API into your applications and get the health of the OpenStack services on demand. You can find additional information about using CloudPulse in the following OpenStack sites:

- https://wiki.openstack.org/wiki/Cloudpulse
- https://wiki.openstack.org/wiki/Cloudpulseclient
- https://wiki.openstack.org/wiki/Cloudpulse/DeveloperNotes
- https://wiki.openstack.org/wiki/Cloudpulse/OperatorTests
- https://wiki.openstack.org/wiki/Cloudpulse/APIDocs

CloudPulse supports two set of tests: endpoint based test (runs as a cron or manually) and operator test (run manually).

The endpoint based test group include:

- nova_endpoint
- neutron_endpoint
- keystone_endpoint
- glance_endpoint
- cinder_endpoint

Operator tests include:

- ceph_check: Executes the command *ceph -f json status* on the Ceph-mon nodes and parses the output. If the output is not HEALTH_OK, the ceph_check reports an error.
- docker_check: Finds out if all the Docker containers are in the running state in all the nodes. It reports an error, if any container is in the exited state. It 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 nova hypervisor list and finds out if all the computes 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 in the cluster rabbitmq_check, the report is considered as failed.

CloudPulse servers are installed in containers on all control nodes. The CloudPulse client is installed on the management node by the Cisco VIM installer. To execute CloudPulse, source the openrc file in the openstack-configs directory and execute the following:

```
[root@MercRegTB1 openstack-configs]# cloudpulse --help
usage: cloudpulse [--version] [--debug] [--os-cache]
[--os-region-name <region-name>]
[--os-tenant-id <auth-tenant-id>]
[--service-type <service-type>]
[--endpoint-type <endpoint-type>]
[--cloudpulse-api-version <cloudpulse-api-ver>]
[--os-cacert <ca-certificate>] [--insecure]
[--bypass-url <bypass-url>] [--os-auth-system <auth-system>]
[--os-username <username>] [--os-password <password>]
[--os-tenant-name <tenant-name>] [--os-token <token>]
[--os-auth-url <auth-url>]
<subcommand> ..
```

To check the results of periodic CloudPulse, enter the following command:

```
[root@MercRegTB1 openstack-configs]# cloudpulse result
+--------------------------------------+------+-------------------+----------+---------+
| uuid                                                                  | id   | name              | testtype
| state   |
+--------------------------------------+------+-------------------+----------+---------+
| 4f4c619a-1ba1-44a7-b6f8-3a06b5903260 | 7394 | ceph_check                   | periodic | success |
| 68b984fa-2edb-4d66-9d9b-7c1b77d2322e | 7397 | keystone_endpoint | periodic | success |
| c53d5f0f-a710-4612-866d-caa896e2d135 | 7400 | docker_check      | periodic | success |
| 988d387c-1160-4601-b2ff-9dbb98a3cd08 | 7403 | cinder_endpoint   | periodic | success |
| 5d702219-eacc-47b7-ae35-582bb8e9b970 | 7406 | glance_endpoint   | periodic | success |
| 033ca2fc-41c9-40d6-b007-16e06dda812c | 7409 | rabbitmq_check    | periodic | success |
| 8476b21e-7111-4b1a-8343-afd634010b07 | 7412 | galera_check      | periodic | success |
| a06f8d6e-7b68-4e14-9b03-bc4408b55b48 | 7415 | neutron_endpoint  | periodic | success |
| ef56b26e-234d-4c33-aee1-ffc99de079a8 | 7418 | nova_endpoint     | periodic | success |
| f60021c7-f70a-44fb-b6bd-03804e5b7bf3 | 7421 | node_check        | periodic | success |
+--------------------------------------+------+-------------------+----------+---------+
```

By default, 25 results are displayed. Use –number argument to get desired number (up to 240) of results. For example,

```
[root@MercRegTB1 openstack-configs]# cloudpulse result -number 100
```

To view all CloudPulse tests, use the following command:

```
# cd /root/openstack-configs
# source openrc
# cloudpulse test-list
```

To run a 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
```

To run a specific CloudPulse test on demand:

```
# 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 |
+------------+--------------------------------------+
```

To show detailed results of a specific CloudPulse run:

```
#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 |
+------------+--------------------------------------+
```

To see the CloudPulse options, source the openrc file in openstack-configs directory and execute:

```
#cloudpulse --help
```

The CloudPulse uses a RESTful http service called the Openstack Health API to allow you to list the CloudPulse tests, create new CloudPulse tests and see the CloudPulse results.
The API calls require keystone authentication. From release Cisco VIM 3.0.0 onwards, only keystone v3 is supported.
The identity service generates authentication tokens that permit access to the CloudPulse REST APIs. Clients obtain this token and the URL endpoints for other service APIs, by supplying their valid credentials to the authentication service. Whenever you make a REST API request to CloudPulse, you must provide your authentication token in the X-Auth-Token request header.

CloudPulse is not applicable for Ceph pod.

Assessing Cisco NFVI Status

Assessing Cisco NFVI Status with Cloud-Sanity

The cloud-sanity tool is designed to give you a quick overall status of the pod health. Cloud-sanity can run tests on all node types in the pod: management, control, compute, and ceph storage.
The following are test areas supported in cloud-sanity:

1. RAID Disk health checks.
2. Basic network connectivity between the management node and all other nodes in the pod.
3. Mariadb cluster size.
4. RabbitMQ operation and status.
5. Nova service and hypervisor list.
6. CEPHMON operation and status.
7. CEPHOSD operation and status.

To run the cloud-sanity tool, login to the management node and run the ciscovim command with the cloud-sanity option.

Cloud-sanity user workflow:

1. Use "ciscovim cloud-sanity create …" command to initiate a test.
2. Use "ciscovim cloud-sanity list …" command to view summary/status of current test jobs.
3. Use "ciscovim cloud-sanity show … --id <ID>" command to view detail test results.
4. Use "ciscovim cloud-sanity delete … --id <ID>" to delete test results no longer needed.

The results are maintained so that you can view them any time.

Delete the results which are no longer needed.

Following are the steps to assess the pod status:

1. To run the cloud sanity, complete the following steps:

```
# ciscovim help cloud-sanity
usage: ciscovim cloud-sanity [--id <id>] [--skip-disk-checks] [-y]
create|delete|list|show test|result
all|control|compute|cephmon|cephosd|management
Run cloud-sanity test suite
Positional arguments:
create|delete|list|show The control command to perform
test|result The identity of the task/action
all|control|compute|cephmon|cephosd|management
The sanity check
Optional arguments:
--id <id> ID used to identify specific item to
show/delete.
--skip-disk-checks Flag to skip running disk-checks during
cloud-sanity test
-y, --yes Yes option to perform the action
```

2. To run the cloud sanity test, you need to create a test job. Once the test job is created, the system displays a message with the time and the ID when the test job is created.

   Run the following command to create a test job:

```
# ciscovim cloud-sanity create test all
+------------+--------------------------------------+
| Field          | Value                                                                  |
+------------+--------------------------------------+
| command          | create                                |
| created_at | 2018-03-07T15:37:41.727739           |
| id                  | c000ca20-34f0-4579-a997-975535d51dda |
| result         |                                                                                  |
| status         | not_run                                                                  |
| test_name  | all                                                                      |
| updated_at | None                                                                   |
+------------+--------------------------------------+
The user can create different test suites based on target roles. All, management, control, compute,
cephmon and cephosd. Only one test will be run at any time.
Example test create commands:
• ciscovim cloud-sanity create test control
o Runs control node tests only
• ciscovim cloud-sanity create test compute
o Runs compute nodes tests only
• ciscovim cloud-sanity create test management
o Runs management node tests only
• ciscovim cloud-sanity create test cephmon
o Runs cephmon tests only
• ciscovim cloud-sanity create test cephosd
o Runs cephosd tests only
```

The cloud-sanity tests use the disk-maintenance and osd-maintenance tools, to assess overall health and status of the RAID disks and OSD status.

> ⚠ Failures detected in RAID disk health and CEPHOSD operational status can be evaluated with the disk-maintenance and osd-maintenance tools.

3. Use the ciscovim cloud-sanity list command to monitor a currently running test or just view all the tests that have been run/completed in the past:

```
# ciscovim cloud-sanity list test all
+--------------------------------------+--------------+----------+--------------------+
| ID                                                                  | Sanity Check | Status   |
Created                             |
+--------------------------------------+--------------+----------+--------------------+
| c000ca20-34f0-4579-a997-975535d51dda | all                  | Complete | 2018-03-07 15:37:41 |
| 83405cf0-e75a-4ce2-a438-0790cf0a196a | cephmon      | Complete | 2018-03-07 15:52:27 |
| 6beceb00-4029-423b-87d6-5aaf0ce087ff | cephmon          | Complete | 2018-03-07 15:55:01 |
| 2707a2e1-d1b5-4176-8715-8664a86bbf7d | cephosd           | Complete | 2018-03-07 16:11:07 |
| b30e1f49-a9aa-4f90-978a-88ba1f0b5629 | control      | Complete | 2018-03-07 16:14:29 |
| f024ff94-ac3e-4745-ba57-626b58ca766b | compute      | Running  | 2018-03-07 16:16:44 |
+--------------------------------------+--------------+----------+--------------------+
We can filter on cephmon if needed
# ciscovim cloud-sanity list test cephmon
+--------------------------------------+--------------+----------+--------------------+
| ID                                                                  | Sanity Check | Status   |
Created            |
+--------------------------------------+--------------+----------+--------------------+
| 83405cf0-e75a-4ce2-a438-0790cf0a196a | cephmon           | Complete | 2018-03-07 15:52:27 |
| 6beceb00-4029-423b-87d6-5aaf0ce087ff | cephmon      | Complete | 2018-03-07 15:55:01 |
+--------------------------------------+--------------+----------+--------------------+
Example cloud-sanity list commands:
• ciscovim cloud-sanity list control
• ciscovim cloud-sanity list compute
• ciscovim cloud-sanity list management
• ciscovim cloud-sanity list cephmon
• ciscovim cloud-sanity list cephosd
```

4. Use the following commands to view the test-sanity results. Cloud-sanity test results can be passed, failed, or skipped. A skipped test is one that is not supported on this particular POD (for example, RAID test is only supported with Hardware RAID.) A skipped test does not count to the overall pass/fail status.

```
# ciscovim cloud-sanity show test all --id c000ca20-34f0-4579-a997-975535d51dda
Cloud sanity Results
+------------+-------------------------------------------------------------+---------+
| Role                |
Task
| Result  |
+------------+-------------------------------------------------------------+---------+
| Management | Management - Disk Maintenance RAID Health *************** | PASSED  |
|                          |                                                                |         |
| Management | Management - Container Version Check ******************** | PASSED  |
|                          |                                                                |         |
| Management | Management - Disk Maintenance VD Health ***************** | PASSED  |
|            |                                                                |         |
| Control    | Control - Check RabbitMQ is Running ********************** | PASSED  |
|                          |                                                                |         |
| Control    | Control - Check RabbitMQ Cluster Status ***************** | PASSED  |
|                          |                                                                |         |
| Control    | Control - Container Version Check ********************** | PASSED  |
|
|
|         |
| Control    | Control - Check MariaDB Cluster Size ******************** | PASSED  |
|
| Control    | Control - Ping All Controller Nodes ********************** | PASSED  |
|                          |                                                                |         |
| Control    | Control - Check Nova Service List ********************** | PASSED  |
|            |                                                                |         |
| Control    | Control - Ping Internal VIP ***************************** | PASSED  |
|            |                                                                |         |
| Control    | Control - Disk Maintenance RAID Health ****************** | PASSED  |
|            |                                                                |         |
| Control    | Control - Disk Maintenance VD Health ******************** | PASSED  |
|            |                                                                |         |
| Compute    | Compute - Check Nova Hypervisor List ******************** | PASSED  |
|            |                                                                |         |
| Compute    | Compute - Disk Maintenance RAID Health ****************** | PASSED  |
|            |                                                                |         |
| Compute    | Compute - Ping All Compute Nodes ************************ | PASSED  |
|            |                                                                |         |
| Compute    | Compute - Container Version Check ********************** | PASSED  |
|            |                                                                |         |
| Compute    | Compute - Disk Maintenance VD Health ******************** | PASSED  |
|            |                                                                |         |
| CephOSD    | CephOSD - Ping All Storage Nodes ************************ | PASSED  |
|            |                                                                |         |
| CephOSD    | CephOSD - Check OSD Result Without OSDinfo *************** | PASSED  |
|            |                                                                |         |
| CephOSD    | CephOSD - OSD Overall Status **************************** | PASSED  |
|            |                                                                |         |
| CephOSD    | CephOSD - Check OSD Result With OSDinfo ***************** | PASSED  |
|            |                                                                |         |
| CephMon    | CephMon - Check Cephmon Status ************************** | PASSED  |
|            |                                                                |         |
| CephMon    | CephMon - Ceph Cluster Check **************************** | PASSED  |
|            |                                                                |         |
| CephMon    | CephMon - Check Cephmon Results ************************* | PASSED  |
|                          |                                                                |         |
| CephMon    | CephMon - Check Cephmon is Running ********************** | PASSED  |
|                          |                                                                |         |
+------------+-------------------------------------------------------------+---------+
[PASSED] Cloud Sanity All Checks Passed
```

5. To delete the cloud sanity test results, run the following command:

```
# ciscovim cloud-sanity delete test all --id c000ca20-34f0-4579-a997-975535d51dda
Perform the action. Continue (Y/N)Y
Delete of UUID c000ca20-34f0-4579-a997-975535d51dda Successful
# ciscovim cloud-sanity list test all
+--------------------------------------+--------------+----------+---------------------+
| ID                                                                | Sanity Check | Status   |
Created                          |
+--------------------------------------+--------------+----------+---------------------+
| 83405cf0-e75a-4ce2-a438-0790cf0a196a | cephmon          | Complete | 2018-03-07 15:52:27 |
| 6beceb00-4029-423b-87d6-5aaf0ce087ff | cephmon           | Complete | 2018-03-07 15:55:01 |
| 2707a2e1-d1b5-4176-8715-8664a86bbf7d | cephosd      | Complete | 2018-03-07 16:11:07 |
| b30e1f49-a9aa-4f90-978a-88ba1f0b5629 | control      | Complete | 2018-03-07 16:14:29 |
| f024ff94-ac3e-4745-ba57-626b58ca766b | compute      | Complete | 2018-03-07 16:16:44 |
+--------------------------------------+--------------+----------+---------------------+
```

The cloud-sanity tests use the disk-maintenance and osd-maintenance tools to assess overall health and status of RAID disks and OSD status.

Failures detected in RAID disk health and CEPHOSD operational status can be evaluated with the disk-maintenance and osd-maintenance tools.

Service Catalog URL

Service Catalog URL

- Get Token from Keystone
- Get Service Catalog URL for CloudPulse
- CloudPulse APIs
    - List of CloudPulse Tests
    - Get Detailed Test Result
    - Get List of Tests Available
    - Schedule a manual CloudPulse test:
    - Remove Test Results

The OpenStack Keystone service catalog allows API clients to dynamically discover and navigate to cloud services. CloudPulse has its own service URL which is added to the Keystone service catalog. You need to send a token request to Keystone, to find the service URL of CloudPulse. The token request lists the catalog of available services.

Get Token from Keystone

To get the token from keystone, run the following commands:

Resource URI

Verb	URI
POST	http://<controller_lb_ip>:5000/v2.0/tokens

Example

```
JSON Request
POST / v2.0/tokens
Accept: application/json
{
        "auth": {
        "passwordCredentials":{
                "username": "admin",
                "password": "iVP1YciVKoMGId1O"}
                        }
}
JSON Response
200 OK
Content-Type: application/json
{
"access": {
        "token": {
        "issued_at": "2017-03-29T09:54:01.000000Z",
        "expires": "2017-03-29T10:54:01Z",
        "id":"gAAAAABY24Q5TDIqizuGmhOXakV2rIzSvSPQpMAmC7SA2UzUXZQXSH-ME98d3Fp4Fsj16G561a420B4BK0fylcykL22EcO9",
............
........}
}
```

Get Service Catalog URL for CloudPulse

Resource URI

Verb	URI
GET	http://<controller_ip>:35357/v2.0/endpoints

Example:

```
JSON Request
GET /v2.0/endpoints
Accept: application/json
JSON Response
200 OK
Content-Type: application/json
{"endpoints": [
{"internalurl": "http://<controller>:9999",
"adminurl": "http://<controller>:9999",
"publicurl":"http://<controller>:9999"
}]}
}
```

CloudPulse APIs

The following is the list of APIs and their corresponding functions. The CloudPulse API is accessed with the X-Auth-token which is received from the Keystone token generation API .

List of CloudPulse Tests

To get the list of cloudpulse tests:

Resource URI

Verb	URI
GET	http://<controller_ip>:9999/cpulse

Get Detailed Test Result

To get detailed test result:

Resource URI

Verb	URI
GET	http://<controller_ip>:9999/cpulse/<uuid>

Uuid : uuid of the test

Example

```
JSON Request
 GET /cpulse/e6d4de91-8311-4343-973b-c507d8806e94
 Accept: application/json
 JSON Response
 200 OK
 Content-Type: application/json
 {
        "name": "galera_check", "state": "success",
        "result":"ActiveNodes:16.0.0.37,16.0.0.17,16.0.0.27",
        "testtype": "periodic", "id": 4122,
        "uuid": " e6d4de91-8311-4343-973b-c507d8806e94"
 }
```

Get List of Tests Available

To get a list of available cloudpulse tests:

Resource URI

Verb	URI
GET	http://<controller_ip>:9999/cpulse/list_tests

Example

```
JSON Request
 GET /cpulse/list_tests
 Accept: application/json
 JSON Response
 200 OK
 Content-Type: application/json
 {
         "endpoint_scenario":

"all_endpoint_tests\ncinder_endpoint\nglance_endpoint\nkeystone_endpoint\nneutron_endpoint\nnova_endpoint",
        "operator_scenario":
"all_operator_tests\nceph_check\ndocker_check\ngalera_check\nnode_check\nrabbitmq_check"
 }
```

Schedule a manual CloudPulse test:

To schedule a manual test of CloudPulse, run the following commands:

Resource URI

Verb	URI
POST	http://<controller_ip>:9999/cpulse

Example

```
JSON Request POST /cpulse
 Accept: application/json
 {
 "name": "galera_check"
 }
 JSON Response
 200 OK
 Content-Type: application/json
 {
 "name": "galera_check",
 "state": "scheduled",
 "result":"NotYetRun",
 "testtype": "manual",
 "id": 4122,
 "uuid": " e6d4de91-8311-4343-973b-c507d8806e94"
 }
```

Remove Test Results

To remove the test results:

Resource URI

Verb	URI
DELETE	http://<controller_ip>:9999/cpulse/<uuid>

Uuid : uuid of the test

Example

```
JSON Request
 DELETE /cpulse/68ffaae3-9274-46fd-b52f-ba2d039c8654
 Accept: application/json
 JSON Response
 204 No Content
```

Checking Network Connections

Checking Network Connections

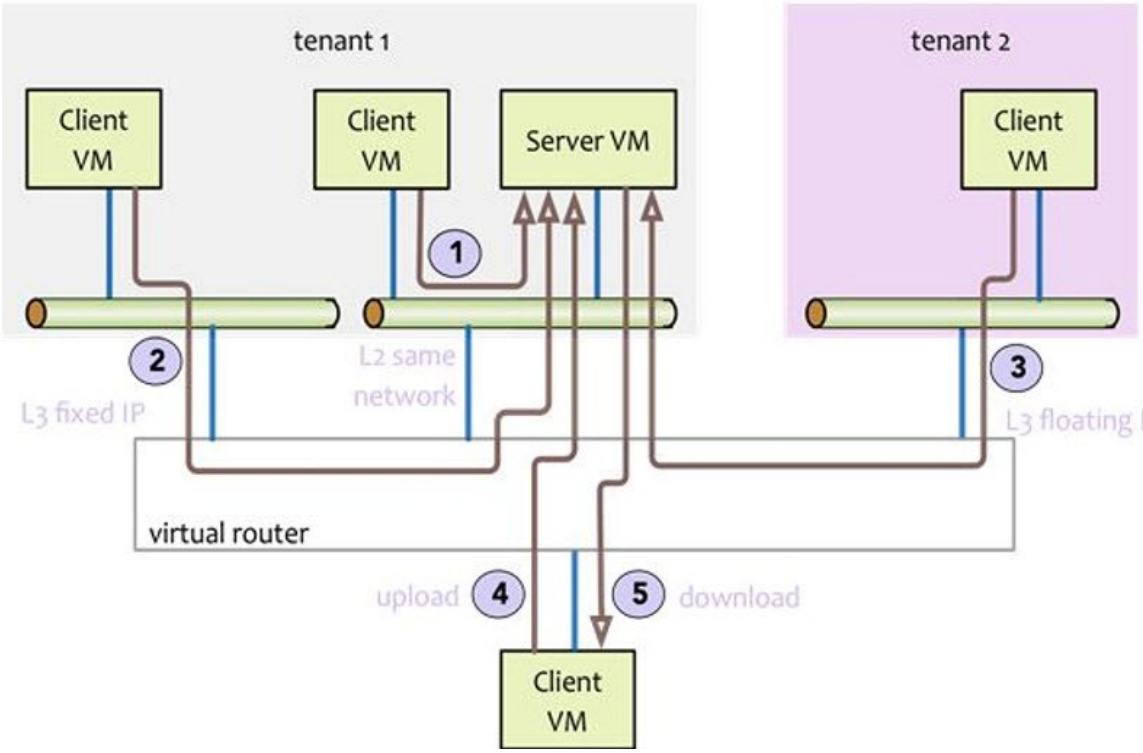
You can use Virtual Machine Through Put (VMTP) to check Layer-2 and Layer-3 data plane traffic between Cisco NFVI compute nodes. VMTP performs ping connectivity, round trip time measurement (latency), and TCP/UDP throughput measurement for the following Cisco NFVI east to west VM-to-VM flows:

- 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).
- When an external Linux host is available for testing north to south flows, external host to VM download and upload throughput and latency (L3 /floating IP, flow numbers 4 and 5).

The following figure shows the traffic flows VMTP measures. Cloud traffic flows are checked during Cisco VIM installation and can be checked at any later time by entering the following command:

```
$ ciscovim run --perform 8 -y
```

The following figure depicts the VMTP cloud traffic monitoring.



General Scheme of Enabling Optional Services

General Scheme of Enabling Optional Services Post Cisco VIM Deployment

Before running the reconfigure option, it is recommended to run the cloud sanity to ensure that the NFVI is up and running and no faults exists. After the successful execution of cloud sanity, take a backup of the setup_data file and update it manually with the configuration details by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml/root/MyDir/ # update the setup_data to for the
targeted change
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Managing VIM Administrators

Managing VIM Administrators

- Additional VIM Administrators
- Enabling VIM Administrator with LDAP Authentication
- Enabling Root Login Post Cisco VIM Installation
- Disabling Root Login

Additional VIM Administrators

The VIM administrator can log into the management through SSH or the console using the configured password. Administrators have their own accounts. After the VIM administrator account creation, the administrator can manage their own password using the Linux *passwd* command. You can change the *vim_admins* parameter to add and remove VIM administrators during reconfiguration, while the passwords for existing accounts remain unchanged.

1. Before launching the installation, take a backup of the *setup_data.yaml* file and update the file manually with the configurations listed below:

```
vim_admins:
- vim_admin_username: <username>
  vim_admin_password_hash: <sha512-password-hash>

- vim_admin_username: <username>
  vim_admin_password_hash: <sha512-password-hash>

- vim_admin_username: <username>
  vim_admin_password_hash: <sha512-password-hash>
  The value of password hash must be in the standard sha512 format.

# To generate the hash admin_password_hash should be the output from on the management node

# python -c "import crypt; print(crypt.crypt('<plaintext password>'))"
```

2. Run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
# update the setup_data to include vim_admin info
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Enabling VIM Administrator with LDAP Authentication

Cisco VIM supports management of the VIM administrators whose access to the management node can be authenticated through an external LDAP server (Open LDAP or Microsoft Active Directory (AD) based). Since Cisco VIM 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 part of Day 0 or Day 1 activity. Multiple LDAP entries are allowed, as only the *domain_name* and *ldap_uri* in each entry are mandatory. Ensure that the *ldap_uri*s 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.*

To obtain sudo access to the management node and execute ciscovim 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 Cisco VIM 3.4.6 onwards. Also, ensure that you enable the pod with external TLS before using this parameter.

To enable VIM administrators with LDAP authentication, perform the following steps:

1. Take a backup of the setupdata file and update the file manually with the configuration listed below during installation:

```
vim_ldap_admins:
- domain_name: corp_ldap1
  ldap_uri: "ldaps://<ip_address_1:[port_1]>,ldaps://<ip_address_2:[port_2]>"
  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

- 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_default_authtok is mandatory, if the LDAP server does not support anonymous bindings.

2. 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)
```

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.

3. To reconfigure the VIM administrator, run the following commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
# update/include the vim_ldap_admin in the setup_data
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Enabling Root Login Post Cisco VIM Installation

Cisco VIM supports an option to enable/disable root access at login. By default, this option is set to True. You can optionally disable this facility through reconfiguration.

Following are the steps to enable the root login:

1. Take a backup of the *setupdata* file and update the file manually with the configurations listed below:

   ```
   permit_root_login: <True or False> # if set to false, use su to drop down to root and execute
   administrator functionalities.
   ```

2. Run the following reconfiguration commands:

   ```
   [root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir
   [root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
   # update the setup_data to include vim_admin info
   [root@mgmt1 ~]# cd /root/MyDir
   [root@mgmt1 ~]# vi setup_data.yaml [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ciscovim --setupfile /root/MyDir/setup_data.yaml reconfigure
   ```

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

# False: admin cannot use root userid for ssh;  must use vim_admin_username # At least one vim_admin must be
configured, if it is False
  permit_root_login: True
```

Managing Power and Reboot

Managing Power and Reboot

- Power Management of C-Series Computes
- Power-on Compute Nodes
- Managing Reboot of Cisco VIM Compute Nodes
- Cisco VIM Client Reboot and Remove Compute Using Force Option
- Managing Reboot Status of Cisco VIM Nodes
- Managing Reboot of Cisco VIM Management Node and Standalone Control and Ceph Nodes

Power Management of C-Series Computes

Before you begin

The power management function of computes optimize the overall power consumption of the data center. Powering down the server through an API/CLI helps you to have a power backup. This procedure is applicable to stand-alone computes only and no VM must be running on that node before power-down is initiated.

1. To power-off one or more compute nodes, run the following commands:

```
Run ciscovim power-off help command
# ciscovim help power-off
usage: ciscovim power-off --setupfile SETUPFILE [-y] <node1,node2,...>
Power Off compute-nodes
Positional arguments:
<node1,node2,...> Power off Compute Nodes
Optional arguments:
--setupfile SETUPFILE <setupdata_file>. Mandatory for any POD management
operation.
-y, --yes Yes option to perform the action
```

2. To list all the nodes in the Openstack cloud, run the following command:

```
# ciscovim list-nodes
```

3. Choose one or more active compute node to be powered off
4. Run the following command:

```
# ciscovim power-off <compute-server-1, compute-server-2, .... > --setupfile <path_setup_data.yaml>
```

5. Run the following command to verify that the computes are power off:

```
#ciscovim list-nodes
```

- The status of the compute nodes that are powered off must be *InActive.*
- To prevent cloud destabilization, you must ensure that at least one compute node is in *Active* state.
- You cannot perform the pod management operations such as update, reconfigure, and so on, to the entire pod, if one or more compute nodes are powered off.
- You cannot turn OFF the power of computes which run VMs or which provide other roles such as All-in-one (AIO) nodes in a Micropod, using this API. Power error-handling methods are added to handle such cases.
- As part of the power-off action, cloud-sanity is run internally. If the cloud-sanity fails, the power-off action is aborted.

Power-on Compute Nodes

Following are the steps to power-on the compute nodes:

1. Run the following command to power-on one or more compute nodes:

```
Run ciscovim power-on help command
# ciscovim help power-on
usage: ciscovim power-on --setupfile SETUPFILE [-y] <node1,node2,...>
Power On compute-nodes
Positional arguments:
<node1,node2,...> Power on Compute Nodes
Optional arguments:
--setupfile SETUPFILE <setupdata_file>. Mandatory for any POD management
operation.
-y, --yes Yes option to perform the action
```

2. To list all the nodes in the OpenStack cloud, enter the following command:

```
# ciscovim list-nodes
```

3. Choose one or more active compute nodes to be powered on.
4. Run the following command:

```
# ciscovim power-on <compute-server-1, compute-server-2, .... > --setupfile <path_setup_data.yaml>
```

5. Run the following command to verify whether the compute(s) are powered on:

```
# ciscovim list-nodes
```


The status of the compute nodes that were powered on must be *Active*.

Managing Reboot of Cisco VIM Compute Nodes

Use ciscovim CLI to reboot the Cisco VIM nodes. During software update, the core libraries like *kernel, glibc, systemd* and so on, require rebooting of the system, to run the latest version. With Cisco VIM, you can reboot nodes (if needed) during an update, but Cisco defers the update of compute nodes which are running application VM's.

Reboot the nodes using the CLI before migrating the VMs on another computes as given below:

1. Run the following command to reboot one or more compute nodes:

```
Run ciscovim reboot help command
# ciscovim help reboot
usage: ciscovim reboot [-y] <node1,node2,...>
Reboot compute-nodes
Positional arguments:
<node1,node2,...> Reboot Compute Nodes
Optional arguments:
-y, --yes Yes option to perform the action
```

2. Run the following command to select one or more compute nodes:

```
# ciscovim reboot<compute-server-1, compute-server-2, .... >
```


- You cannot reboot all the compute nodes simultaneously. Ensure that at least one node is *Active* to prevent the cloud destabilization.
- You cannot reboot the computes on which VMs are running. The nodes which are associated with multiple roles, for example, All-in-one (AIO) nodes in a Micropod or Hyper-converged can be rebooted one at a time.

Cisco VIM Client Reboot and Remove Compute Using Force Option

When VM's are running on a particular compute node, you cannot reboot or remove that compute node. Cisco VIM installer internally checks for the presence of VM's and aborts the operation, if VM is running on the target compute node.
To execute remove-compute operation without any failure, migrate or terminate the VMs running on compute nodes and then execute remove or reboot operations using the below option in Cisco VIM client:

```
-f/--force
```

Before executing reboot or remove compute operations with *force* option:

- If a remove compute operation is executed with a force option, the VMs running on that compute node are deleted.
- If a reboot compute operation is executed with a force option, the VMs are restored to last running status post successful reboot of that compute node.

Example of Remove Compute

```
# ciscovim help remove-computes
usage: ciscovim remove-computes --setupfile SETUPFILE [-y] [-f] <node1,node2,...>
Remove compute-nodes from the Openstack cloud
Positional arguments:
<node1,node2,...> Remove compute nodes
Optional arguments:
--setupfile SETUPFILE <setupdata_file>. Mandatory for any POD management
operation.
-y, --yes Yes option to perform the action
-f, --force Force option to remove or reboot
# ciscovim remove-computes --setupfile /tmp/remove_computes_setup_data.yaml gg34-4 -y --force
monitoring remove_compute (gg34-4) operation
.......................
Cisco VIM Runner logs
.......................
```

Example of Removing Multiple Computes

```
# ciscovim remove-computes --setupfile /tmp/remove_computes_setup_data.yaml gg34-1, gg34-2
-y -force
```

For remove-compute operation, Cisco VIM attempts to change the host OS boot mode to single user and power-off those compute node(s) to avoid the creation of duplicate IP address on the management and storage networks, and to prevent the compute node(s) from connecting back to the deployment.

If the compute node(s) to be removed are powered-off before triggering Cisco VIM remove-compute operation, the server must not be brought back on the network as it may create duplicate IP address when connected to deployment with valid credentials.

To power up the removed compute node(s):
- network isolate the compute node(s) (shutdown of corresponding ToR switchports or unplug the network cables)
- power up the server from CIMC
- delete the virtual drive from CIMC
- power-cycle the server from CIMC
- bring the compute node on the network (no shutdown of corresponding ToR switchports or plug back the network cables)

If ToR_TYPE is Cisco NCS 5500, you must manually remove all the sub-interfaces that are manually configured on the NCS switch, as the Cisco VIM automation does not unconfigure/configure the sub-interfaces for which the VLANs are not defined in the *setup_data.yaml.* If sub-interfaces are not removed, it results in remove-compute operation.

Example of Reboot Compute

```
# ciscovim help reboot
usage: ciscovim reboot [-y] [-f] <node1,node2,...>
Reboot compute-nodes
Positional arguments:
<node1,node2,...> Reboot Compute Nodes
Optional arguments:
-y, --yes Yes option to perform the action
-f, --force Force option to perform the action
# ciscovim reboot gg34-4 -y --force
monitoring reboot (gg34-4) operation
.......................
Cisco VIM Runner logs
.......................
```

Example of Rebooting Multiple Computes

```
# ciscovim reboot compute-server-hostname1,compute-server-hostname2 -y --force
```

Managing Reboot Status of Cisco VIM Nodes

You can find which Cisco VIM nodes require a reboot after an update, using the CLI. Reboot the nodes after an update, so that the cloud is running the latest host packages.

It is mandatory for the operator to reboot nodes to be able to perform next update or pod management operation.

Run the following command to check the reboot pending status for nodes in the pod, post update.

```
Run ciscovim reboot-status help command
# ciscovim help reboot-status
usage: ciscovim reboot-status
List of Openstack Nodes that require a reboot
Sample command execution:
# ciscovim reboot-status
Fetching Nodes that require a Reboot
+--------------------------+-----------------+
| Node Name                | Reboot Required |
+--------------------------+-----------------+
| compute-server-hostname1 | No              |
| compute-server-hostname2 | No              |
| compute-server-hostname3 | No              |
+--------------------------+-----------------+
```

Managing Reboot of Cisco VIM Management Node and Standalone Control and Ceph Nodes

You must reboot the nodes that constitute the control plane of Cisco VIM with utmost care, only during a maintenance window. Execute the *ciscovim cluster recovery,* if any of the controllers are rebooted. Check the ceph status post reboot of the Ceph nodes and wait until Ceph reaches HEALTH_OK state.

To reboot the nodes one at a time, execute the following command:

```
# shutdown -r now
```


The power down of nodes must be done gracefully but not via CIMC/BMC/ILO to avoid potential disk corruption.

Read-only OpenStack Role

Read-only OpenStack Role

By default, Cisco VIM deployment of OpenStack supports two user roles: admin and user. Admin can view and change all OpenStack resources including system and project resources. Users can view and change only project resources.
Optionally, Cisco VIM provides OpenStack user role which is read-only administrator or readonly. Use the optional parameter ENABLE_READONLY_ROLE to enable this feature.

The OpenStack read-only user can:

- Access the project and identity dashboards, but not the admin dashboard.
- View all the project resources, but cannot make any changes to them.

The admin can only assign the readonly role using the Horizon dashboard or OpenStack CLI, to the target user for accessing each project. A user can be given the readonly role to multiple projects.

> ⚠ Ensure that the admin role is not given for the user having only readonly access, as the conflict of access will not constrain the user to read-only operations.

Enabling this feature provides the following enhancements to the Cisco VIM pod.

- *readonly* role is added to the OpenStack deployment.
- OpenStack service policies are adjusted to grant read permissions such as *list* and *show*, but not *create, update*, or *delete.*
- **AllProjects** tab is added to the Horizon interface. This allows the readonly user to see all instances for which the user have access. Under the **Project** tab, you can see the resources for a single project. You can change the projects using the Project pulldown in the header.

To enable read-only OpenStack role and create read-only OpenStack administrator, perform the following steps:

1. Before launching the installation, take a backup of the setupdata file and update the setup_data.yaml file with the following information:

```
ENABLE_READONLY_ROLE: True
```

2. If the OpenStack user role is not enabled on the Day 0, you can enable it by executing the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
# update the setup_data to include ENABLE_READONLY_ROLE: True
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

When the feature is enabled, an OpenStack administrator can create new user accounts that will have the special privileges of a Read-Only user.

3. From the management node, load the OpenStack authentication variables:

```
[root@management-server-cisco ~]# source ~/openstack-configs/openrc
```

4. Create a new user account with a strong password:

```
[root@management-server-cisco images]# openstack user create --password-prompt reader
User Password:
Repeat User Password:
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | e2f484de1e7d4faa9c9de2446ba8c3de |
| name     | reader                           |
| username | reader                           |
+----------+----------------------------------+
```

5. Assign the project and role to that user account:

```
[root@management-server-cisco images]# openstack role add --project admin --user reader
readonly
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id            | ed2fb5b2c88e42419185736240901 74b |
| name             | readonly                                  |
+-----------+----------------------------------+
```

Alternatively, the OpenStack admin logged into the Horizon dashboard can perform the above steps. You can perform the actions corresponding to the CLI commands on the Identity/Users panel in the dashboard.

After deployment, the administrators can create new users assigned with this role.

If the ENABLE_READONLY_ROLE is False (by default), the readonly role will not have special permissions or restrictions, but have the create, update, and delete permissions to project resources similar to that of project member. You need to assign the users with readonly role, when ENABLE_READONLY_ROLE is set to True.

Cisco VIM Advanced OpenStack Policy

Cisco VIM Advanced OpenStack Policy

Cisco VIM advanced OpenStack supports the following roles for Cisco VIM pod :

- Admin: View and change all OpenStack resources including system and project resources.
- Member: View and change only the project resources.
- Readonly: Optionally available to view the project resources regardless of ENABLE_READONLY_ROLE setting.
- Operator: Available as a tech preview option from Cisco VIM 4.2.0, to start, stop, and reboot servers (instances) with read only privileges. Only the admin can assign this operator role using the Horizon dashboard or OpenStack CLI, to the target user for accessing one or more projects.

Ensure that the admin role is not given for the people having only operator role, as the conflict of access will not constrain the user to operator role.

If advanced roles is enabled, **All Projects** tab is visible in the Horizon interface. This allows the user with operator role, to see all instances for which the user have access. Under the **Project** tab, you can see the resources for a single project. You can change the projects using the Project pulldown in the header.

To enable operator OpenStack role and create operator OpenStack administrator, perform the following:

1. Before launching the installation, take a backup of the *setupdata* file and update the *setup_data.yaml* file with the following information:

   ```
   ENABLE_ADVANCED_ROLES: <True or False> # default is False
   ```

2. If the operator OpenStack role is not enabled on the Day 0, you can enable it by executing the following reconfiguration commands:

   ```
   [root@mgmt1 ~]# cd /root/
   [root@mgmt1 ~]# mkdir MyDir
   [root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
   # update the setup_data to include ENABLE_ADVANCED_ROLES: True
   [root@mgmt1 ~]# cd /root/MyDir/
   [root@mgmt1 ~]# vi setup_data.yaml
   [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
   ```

   When the advanced OpenStack roles is enabled, the OpenStack administrator can create new user accounts having special privileges of a operator role.
3. From the management node, load the OpenStack authentication variables for creating user accounts:

   ```
   [root@management-server-cisco ~]# source ~/openstack-configs/openrc
   ```

4. Create a new user account with a strong password:

   ```
   [root@management-server-cisco images]# openstack user create --password-prompt operate
   User Password:
   Repeat User Password:
   +----------+----------------------------------+
   | Field    | Value                            |
   +----------+----------------------------------+
   | email    | None                             |
   | enabled  | True                             |
   | id       | e2f484de1e7d4faa9c9de2446ba8c3de |
   | name     | operate                          |
   | username | operate                          |
   +----------+----------------------------------+
   ```

5. Assign the project and role to that created user account:

```
[root@management-server-cisco images]# openstack role add --project admin --user operate operator
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id            | ed2fb5b2c88e42419185736240901 74b |
| name              | operator                                 |
+-----------+----------------------------------+
```

Alternatively, the OpenStack admin logged into the Horizon dashboard can perform the above steps. You can perform the actions corresponding to the CLI commands on the Identity/Users panel in the dashboard.

After deployment, the administrators can create new users assigned with this role.

If the ENABLE_ADVANCED_ROLES is False (by default), the operator role will not have special permissions or restrictions, but have the create, update, and delete permissions to project resources similar to that of project member. You need to assign the users with operator role, when ENABLE_ADVANCED_ROLES is set to True.

Security

Security

The following topics describe Cisco NFVI network and application security and best practices.

- Verification
- Reconfiguration of CVIM Parameters
- Cloud Settings
- Fernet Key Operations
- Certificates
- LDAP AD Support with Keystone v3
- NetApp from http to https
- Hardening Cisco VIM Deployment
- Securing Management Node
- LDAP Authentication for Kibana
- Enabling Barbican
- Enabling Management Network over TLS
- Usage of Pre-encrypted Images

Verification

Verification

- Verifying Management Node Network Permissions
- Verifying Management Node File Permissions
- Viewing Administrator Access Attempts
- Verifying SELinux
- Validating Port Listening Services
- Validating Non-Root Users for OpenStack Services
- Verifying Password Strength

Verifying Management Node Network Permissions

The Cisco NFVI management node stores sensitive information related to Cisco NFVI operations. Access to the management node can be restricted to requests coming from IP addresses known to be used by administrators. The administrator source networks is configured in the setup file, under **[NETWORKING]** using the **admin_source_networks** parameter.

To verify this host based firewall setting, log into the management node as an admin user and list the rules currently enforces by iptables. Verify that the source networks match the values configured. If no source networks have been configured, then all source traffic is allowed. However, note that only traffic destined to ports with known admin services is allowed to pass. The **admin_source_networks** value can be set at install time or changed through reconfiguration.

```
[root@control-server-1 ~]# iptables -list
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT icmp -- anywhere anywhere
ACCEPT tcp -- 10.0.0.0/8 anywhere tcp dpt:ssh
ACCEPT tcp -- 172.16.0.0/12 anywhere tcp dpt:ssh
ACCEPT tcp -- 10.0.0.0/8 anywhere tcp dpt:https
ACCEPT tcp -- 172.16.0.0/12 anywhere tcp dpt:https
ACCEPT tcp -- 10.0.0.0/8 anywhere tcp dpt:4979
ACCEPT tcp -- 172.16.0.0/12 anywhere tcp dpt:4979
ACCEPT tcp -- 10.0.0.0/8 anywhere tcp dpt:esmagent
ACCEPT tcp -- 172.16.0.0/12 anywhere tcp dpt:esmagent
ACCEPT tcp -- 10.0.0.0/8 anywhere tcp dpt:8008
ACCEPT tcp -- 172.16.0.0/12 anywhere tcp dpt:8008
ACCEPT tcp -- 10.0.0.0/8 anywhere tcp dpt:copy
ACCEPT tcp -- 172.16.0.0/12 anywhere tcp dpt:copy
ACCEPT tcp -- 10.0.0.0/8 anywhere tcp dpt:22250
ACCEPT tcp -- 172.16.0.0/12 anywhere tcp dpt:22250
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED
DROP   all -- anywhere anywhere
```

Verifying Management Node File Permissions

The Cisco NFVI management node stores sensitive information related to Cisco NFVI operations. These files are secured by strict file permissions. Sensitive files include secrets.yaml, openrc, *.key, and *.pem. To verify the file permissions, log into the management node as an admin user and list all of the files in the *~/openstack-configs/* directory. Verify that only the owner has read and write access to these files. For example:

```
[root@control-server-1 ~]# ls -l ~/openstack-configs
total 172
-rw-------. 1 root root 3272 Jun 21 17:57 haproxy.key
-rw-------. 1 root root 5167 Jun 21 17:57 haproxy.pem
-rw-------. 1 root root 223 Aug 8 18:09 openrc
-rw-------. 1 root root 942 Jul 6 19:44 secrets.yaml
[...]
```

Viewing Administrator Access Attempts

As the UCS servers are part of the critical Cisco NFVI infrastructure, Cisco recommends monitoring administrator login access periodically.

To view the access attempts, use the *journalctl* command to view the log created by ssh. For example:

```
[root@control-server-1 ~]# journalctl -u sshd
-- Logs begin at Tue 2016-06-21 17:39:35 UTC, end at Mon 2016-08-08 17:25:06 UTC. --
Jun 21 17:40:03 hh23-12 systemd[1]: Started OpenSSH server daemon.
Jun 21 17:40:03 hh23-12 systemd[1]: Starting OpenSSH server daemon...
Jun 21 17:40:03 hh23-12 sshd[2393]: Server listening on 0.0.0.0 port 22.
Jun 21 17:40:03 hh23-12 sshd[2393]: Server listening on :: port 22.
Jun 21 17:40:43 hh23-12 sshd[12657]: Connection closed by 171.70.163.201 [preauth]
Jun 21 17:41:13 hh23-12 sshd[12659]: Accepted password for root from 171.70.163.201 port 40499
Jun 21 17:46:41 hh23-12 systemd[1]: Stopping OpenSSH server daemon...
Jun 21 17:46:41 hh23-12 sshd[2393]: Received signal 15; terminating.
Jun 21 17:46:41 hh23-12 systemd[1]: Started OpenSSH server daemon.
Jun 21 17:46:41 hh23-12 systemd[1]: Starting OpenSSH server daemon...
Jun 21 17:46:41 hh23-12 sshd[13930]: Server listening on 0.0.0.0 port 22.
Jun 21 17:46:41 hh23-12 sshd[13930]: Server listening on :: port 22.
Jun 21 17:50:45 hh23-12 sshd[33964]: Accepted password for root from 171.70.163.201 port 40545
Jun 21 17:56:36 hh23-12 sshd[34028]: Connection closed by 192.168.212.20 [preauth]
Jun 21 17:57:08 hh23-12 sshd[34030]: Accepted publickey for root from 10.117.212.20 port 62819
Jun 22 16:42:40 hh23-12 sshd[8485]: Invalid user user1 from 10.117.212.20
Jun 22 16:42:40 hh23-12 sshd[8485]: input_userauth_request: invalid user user1 [preauth]
```

Verifying SELinux

To minimize the impact of a security breach on a Cisco NFVI server, the Cisco VM enables SELinux (Security Enhanced Linux) to protect the server resources. To validate that SELinux is configured and running in enforcing mode, use the sestatus command to view the status of SELinux and verify that its status is enabled and in enforcing mode. For example:

```
[root@mgmt1 ~]# /usr/sbin/sestatus -v
SELinux status: enabled
SELinuxfs mount: /sys/fs/selinux
SELinux root directory: /etc/selinux
Loaded policy name: targeted
Current mode: enforcing
Mode from config file: permissive
Policy MLS status: enabled
Policy deny_unknown status: allowed
Max kernel policy version: 28
```

Validating Port Listening Services

To prevent access by unauthorized users and processes, Cisco NFVI has no extra services listening on network ports. To verify this, use the netstat -plnt command to get a list of all services listening on the node and verif that no unauthorized services are listening. For example:

```
[root@-control-server-1 ~]# netstat -plnt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program
name
tcp 0 0 23.23.4.101:8776 0.0.0.0:* LISTEN 24468/python2
tcp 0 0 23.23.4.101:5000 0.0.0.0:* LISTEN 19874/httpd
tcp 0 0 23.23.4.101:5672 0.0.0.0:* LISTEN 18878/beam.smp
tcp 0 0 23.23.4.101:3306 0.0.0.0:* LISTEN 18337/mysqld
tcp 0 0 127.0.0.1:11211 0.0.0.0:* LISTEN 16563/memcached
tcp 0 0 23.23.4.101:11211 0.0.0.0:* LISTEN 16563/memcached
tcp 0 0 23.23.4.101:9292 0.0.0.0:* LISTEN 21175/python2
tcp 0 0 23.23.4.101:9999 0.0.0.0:* LISTEN 28555/python
tcp 0 0 23.23.4.101:80 0.0.0.0:* LISTEN 28943/httpd
tcp 0 0 0.0.0.0:4369 0.0.0.0:* LISTEN 18897/epmd
tcp 0 0 127.0.0.1:4243 0.0.0.0:* LISTEN 14673/docker
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 2909/sshd
tcp 0 0 23.23.4.101:4567 0.0.0.0:* LISTEN 18337/mysqld
tcp 0 0 23.23.4.101:15672 0.0.0.0:* LISTEN 18878/beam.smp
tcp 0 0 0.0.0.0:35672 0.0.0.0:* LISTEN 18878/beam.smp
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 4531/master
tcp 0 0 23.23.4.101:35357 0.0.0.0:* LISTEN 19874/httpd
tcp 0 0 23.23.4.101:8000 0.0.0.0:* LISTEN 30505/python
tcp 0 0 23.23.4.101:6080 0.0.0.0:* LISTEN 27996/python2
tcp 0 0 23.23.4.101:9696 0.0.0.0:* LISTEN 22396/python2
tcp 0 0 23.23.4.101:8004 0.0.0.0:* LISTEN 30134/python
tcp 0 0 23.23.4.101:8773 0.0.0.0:* LISTEN 27194/python2
tcp 0 0 23.23.4.101:8774 0.0.0.0:* LISTEN 27194/python2
tcp 0 0 23.23.4.101:8775 0.0.0.0:* LISTEN 27194/python2
tcp 0 0 23.23.4.101:9191 0.0.0.0:* LISTEN 20752/python2
tcp6 0 0 :::9200 :::* LISTEN 18439/xinetd
tcp6 0 0 :::4369 :::* LISTEN 18897/epmd
tcp6 0 0 :::22 :::* LISTEN 2909/sshd
tcp6 0 0 ::1:25 :::* LISTEN 4531/master
```

Validating Non-Root Users for OpenStack Services

To prevent unauthorized access, Cisco NFVI runs OpenStack processes as a non-root user. To verify OpenStack processes are not running as root, use the ps command to get a list of all node processes. In the following example the user is 162:

```
[root@control-server-1 ~]# ps -aux | grep nova-api
162 27194 0.6 0.0 360924 132996 ? S Aug08 76:58 /usr/bin/python2
/usr/bin/nova-api
162 27231 0.0 0.0 332192 98988 ? S Aug08 0:01 /usr/bin/python2
/usr/bin/nova-api
162 27232 0.0 0.0 332192 98988 ? S Aug08 0:01 /usr/bin/python2
/usr/bin/nova-api
162 27233 0.0 0.0 332192 98988 ? S Aug08 0:01 /usr/bin/python2
/usr/bin/nova-api
```

Verifying Password Strength

Cisco NFVI passwords can be generated in two ways during installation:

- The Cisco NFVI installer generates unique passwords automatically for each protected service.
- You can provide an input file containing the passwords you prefer.

Cisco-generated passwords are unique, long, and contain a mixture of uppercase, lowercase, and numbers. If you provide the passwords, password strength is your responsibility.

You can view the passwords by displaying the secrets.yaml file. For example

```
[root@mgmt1 ~]# cat ~/openstack-configs/secrets.yaml
ADMIN_USER_PASSWORD: QOZGSjVQzgu7ejv1
CINDER_DB_PASSWORD: TP2h7OAfa0VHZBb2
CINDER_KEYSTONE_PASSWORD: 0jko2Vc76h005eP9
CLOUDPULSE_KEYSTONE_PASSWORD: Vuov6wdPe5jc5kGp
COBBLER_PASSWORD: 8bhVOeciqw5jUyY5
CPULSE_DB_PASSWORD: 2DwLE0IsavQWEfMn
CVIM_MON_PASSWORD: t4qf4ORVRTtce4E0
CVIM_MON_READ_ONLY_PASSWORD: UTicXzdxn0krFplS
CVIM_MON_SERVER_PASSWORD: 1qcASpt2bRuDWbi7
DB_ROOT_PASSWORD: 5a4pQjTpCZDO1sE5
ETCD_ROOT_PASSWORD: 43yluJNsNBhv8kTp
GLANCE_DB_PASSWORD: U1HdRc7lkZslW2nD
GLANCE_KEYSTONE_PASSWORD: FpQfFnqg0AtcJbVa
HAPROXY_PASSWORD: dQzIKoi9WbCxwHGz
```

When Vault is used, it provides information about the following password only:

"CVIM_MON_PASSWORD", "CVIM_MON_READ_ONLY_PASSWORD",

"CVIM_MON_SERVER_PASSWORD", "ADMIN_USER_PASSWORD", "KIBANA_PASSWORD",

“CVIM_MON_PROXY_PASSWORD", "HAPROXY_PASSWORD".

```
# ciscovim list-secrets --getpassword <PASSWORD_KEY>
For example,
ciscovim list-secrets --getpassword ADMIN_USER_PASSWORD
+---------------------+-----------------------------+
|Secret Key                         | Secret Value                                |
+---------------------+-----------------------------+
| ADMIN_USER_PASSWORD| D1g8O6Ws2Woav7Ye             |
|+--------------------+-----------------------------+
```

Reconfiguration of CVIM Parameters

Reconfiguration of CVIM Parameters

- Reconfiguring Passwords and OpenStack Configurations
- Reconfiguring Glance Client Key for Central Ceph Cluster
- Reconfiguring CIMC/BMC Password on Existing Installation
- Reconfiguring Administrator Source Networks
- Password Reset for Cisco VIM Management Node

Reconfiguring Passwords and OpenStack Configurations

You can reset some configurations after installation including the OpenStack service password and debugs, TLS certificates, and log rotation configurations. The *secrets.yaml* and *openstack_config.yaml* which are located in : */root/installer-{tag id}/openstack-configs/* contain the passwords, debugs, TLS file location, and ELK configurations. Also, Elasticsearch uses disk space for the data that is sent to it. Cisco VIM has configuration variables that establishes the frequency and file size.

Cisco VIM installer generates the OpenStack service and database passwords with 16 alphanumeric characters and stores those in */root/openstack-configs /secrets.yaml.* You can change the OpenStack service and database passwords using the password reconfigure command on the deployed cloud. The command identifies the containers affected by the password change and restarts them so the new password can take effect.

Always schedule the password reconfiguration in a maintenance window as the container restart might disrupt the control plane.

Run the following command to view the list of passwords and configurations:

```
[root@mgmt1 installer-xxxx]# ciscovim help reconfigure
usage: ciscovim reconfigure [--regenerate_secrets] [--setpassword <secretkey>]
[--setopenstackconfig <option>]
Reconfigure the openstack cloud
Optional arguments:
--regenerate_secrets Regenerate All Secrets
--setpassword <secretkey> Set of secret keys to be changed.
--setopenstackconfig <option> Set of Openstack config to be changed.
[root@mgmt1 ~]# ciscovim list-openstack-configs
+----------------------------+-----------------------------------------------------------------------------------------
+
|            Name            |                                      Option
|
+----------------------------+-----------------------------------------------------------------------------------------
+
|    IRONIC_DEBUG_LOGGING    |                                       True
|
|    OPFLEX_DEBUG_LOGGING    |                                       True
|
|  GNOCCHI_VERBOSE_LOGGING   |                                       True
|
|     AIM_DEBUG_LOGGING      |                                       True
|
|    CINDER_DEBUG_LOGGING    |                                      False
|
|   KEYSTONE_DEBUG_LOGGING   |                                      False
|
|     log_rotation_size      |                                       100M
|
| CLOUDPULSE_VERBOSE_LOGGING |                                       True
|
|  OCTAVIA_VERBOSE_LOGGING   |                                       True
|
|   MAGNUM_VERBOSE_LOGGING   |                                       True
|
|     NOVA_DEBUG_LOGGING     |                                       True
|
|  NEUTRON_VERBOSE_LOGGING   |                                       True
|
|    external_lb_vip_cert    |                      /root/openstack-configs/haproxy.pem
|
|   GLANCE_VERBOSE_LOGGING   |                                       True
```

```
|
|   elk_rotation_frequency   |                                        weekly
|
| CEILOMETER_VERBOSE_LOGGING |                                         True
|
| NOVA_CPU_ALLOCATION_RATIO  |                                         16.0
|
|  CLOUDPULSE_DEBUG_LOGGING  |                                        False
|
|   log_rotation_frequency   |                                        weekly
|
|     HEAT_DEBUG_LOGGING     |                                        False
|
|  KEYSTONE_VERBOSE_LOGGING  |                                         True
|
|   external_lb_vip_cacert   |                      /root/openstack-configs/haproxy-ca.crt
|
|   GNOCCHI_DEBUG_LOGGING    |                                        False
|
|    MAGNUM_DEBUG_LOGGING    |                                         True
|
|   OCTAVIA_DEBUG_LOGGING    |                                        False
|
|   log_rotation_del_older   |                                          8
|
|   CINDER_VERBOSE_LOGGING   |                                         True
|
|     elk_rotation_size      |                                          2
|
|   IRONIC_VERBOSE_LOGGING   |                                         True
|
|   elk_rotation_del_older   |                                          8
|
|   NEUTRON_DEBUG_LOGGING    |                                         True
|
|    HEAT_VERBOSE_LOGGING    |                                         True
|
|  CEILOMETER_DEBUG_LOGGING  |                                        False
|
|   ES_SNAPSHOT_AUTODELETE   |         {u'threshold_warning': 60, u'enabled': True, u'period': 'hourly',
|
|                            |                    u'threshold_high': 80, u'threshold_low': 50}
|
|    GLANCE_DEBUG_LOGGING    |                                        False
|
|    NOVA_VERBOSE_LOGGING    |                                         True
|
| NOVA_RAM_ALLOCATION_RATIO  |                                         1.5
|
+----------------------------+-------------------------------------------------------------------------------
+
The command in to get the list of all passwords:
[root@mgmt1 installer-xxxx]# ciscovim list-secrets
+---------------------------------+---------------------------+---------------------------+
| Secret Key                      | Created At                | Updated At                |
+---------------------------------+---------------------------+---------------------------+
| ADMIN_USER_PASSWORD             | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| CEILOMETER_DB_PASSWORD          | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| CEILOMETER_KEYSTONE_PASSWORD    | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| CINDER_DB_PASSWORD              | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| CINDER_KEYSTONE_PASSWORD        | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| CLOUDPULSE_KEYSTONE_PASSWORD    | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| COBBLER_PASSWORD                | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| CPULSE_DB_PASSWORD              | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| CVIM_MON_SERVER_PASSWORD        | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| DB_ROOT_PASSWORD                | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| ETCD_ROOT_PASSWORD              | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| GLANCE_DB_PASSWORD              | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| GLANCE_KEYSTONE_PASSWORD        | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| HAPROXY_PASSWORD                | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| GNOCCHI_DB_PASSWORD             | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
```

```
| GNOCCHI_KEYSTONE_PASSWORD        | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| HEAT_DB_PASSWORD                 | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| HEAT_KEYSTONE_PASSWORD           | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| HEAT_STACK_DOMAIN_ADMIN_PASSWORD | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| HORIZON_SECRET_KEY               | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| IRONIC_DB_PASSWORD               | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| IRONIC_KEYSTONE_PASSWORD         | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| KEYSTONE_DB_PASSWORD             | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| KIBANA_PASSWORD                  | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| METADATA_PROXY_SHARED_SECRET     | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| NEUTRON_DB_PASSWORD              | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| NEUTRON_KEYSTONE_PASSWORD        | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| NOVA_DB_PASSWORD                 | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| NOVA_KEYSTONE_PASSWORD           | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| RABBITMQ_ERLANG_COOKIE           | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| RABBITMQ_PASSWORD                | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| VOLUME_ENCRYPTION_KEY            | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| VPP_ETCD_PASSWORD                | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
| WSREP_PASSWORD                   | 2020-04-19 00:52:22+00:00 | 2020-04-19 00:52:22+00:00 |
+----------------------------------+---------------------------+---------------------------+

[root@mgmt1 installer-xxxx]#
When using Vault, you can fetch information about the following user facing password only:
"CVIM_MON_PASSWORD", "CVIM_MON_READ_ONLY_PASSWORD",
"CVIM_MON_SERVER_PASSWORD", "ADMIN_USER_PASSWORD", "KIBANA_PASSWORD",
"CVIM_MON_PROXY_PASSWORD", "HAPROXY_PASSWORD".
```

You can change specific password and configuration identified from the available list.
Run the reconfiguration command as follows:

```
# ciscovim help reconfigure
usage: ciscovim reconfigure [--regenerate_secrets]
                            [--setupfile <setupdata_file>]
                            [--alertmanager_config <alertmanager_config_file>]
                            [--alerting_rules_config <alerting_rules_config_file>]
                            [--setpassword <secretkey>]
                            [--setopenstackconfig <option>]
                            [--setopenstackconfig_file <config_file>]
                            [--cimc_password] [--rma_tors <tor1,tor3,...>]
                            [--regenerate_ceph_keyring] [-y]
Reconfigure the Openstack cloud

Optional arguments:
  --regenerate_secrets            Regenerate All Secrets
  --setupfile <setupdata_file>    User setup_data.yaml
  --alertmanager_config <alertmanager_config_file>
                                  User alertmanager_config.yaml
  --alerting_rules_config <alerting_rules_config_file>
                                  User alerting_rules_config.yaml
  --setpassword <secretkey>       Set of secret keys to be changed.
  --setopenstackconfig <option>   Set of Openstack config to be changed.
  --setopenstackconfig_file <config_file>
                                  Set of Openstack configs to be changed from
                                  file.
  --cimc_password                 Reconfigure CIMC password
  --rma_tors <tor1,tor3,...>      Comma separated list of ToRs
  --regenerate_ceph_keyring       Regenerate Ceph Keyring
  -y, --yes                       Yes option to perform the action to reconfigure the Openstack cloud

[root@mgmt1 ~]# ciscovim reconfigure --setpassword ADMIN_USER_PASSWORD,NOVA_DB_PASSWORD --setopenstackconfig
HEAT_DEBUG_LOGGING,HEAT_VERBOSE_LOGGING
Password for ADMIN_USER_PASSWORD:
Password for NOVA_DB_PASSWORD:
Enter T/F for option HEAT_DEBUG_LOGGING:T
Enter T/F for option HEAT_VERBOSE_LOGGING:T
```

The password must be alphanumeric and can be maximum 32 characters in length.

Following are the configuration parameters for OpenStack:

Configuration Parameter	Allowed Values
AIM_DEBUG_LOGGING	T/F (True or False)
CEILOMETER_DEBUG_LOGGING	T/F (True or False)
CEILOMETER_VERBOSE_LOGGING	T/F (True or False)
CINDER_DEBUG_LOGGING	T/F (True or False)
CINDER_VERBOSE_LOGGING	T/F (True or False)
CLOUDPULSE_DEBUG_LOGGING	T/F (True or False)
CLOUDPULSE_VERBOSE_LOGGING	T/F (True or False)
GLANCE_DEBUG_LOGGING	T/F (True or False)
GLANCE_VERBOSE_LOGGING	T/F (True or False)
GNOCCHI_DEBUG_LOGGING	T/F (True or False)
GNOCCHI_VERBOSE_LOGGING	T/F (True or False)
HEAT_DEBUG_LOGGING	T/F (True or False)
HEAT_VERBOSE_LOGGING	T/F (True or False)
IRONIC_DEBUG_LOGGING	T/F (True or False)
IRONIC_VERBOSE_LOGGING	T/F (True or False)
KEYSTONE_DEBUG_LOGGING	T/F (True or False)
KEYSTONE_VERBOSE_LOGGING	T/F (True or False)
MAGNUM_DEBUG_LOGGING	T/F (True or False)
MAGNUM_VERBOSE_LOGGING	T/F (True or False)
NEUTRON_DEBUG_LOGGING	T/F (True or False)
NEUTRON_VERBOSE_LOGGING	T/F (True or False)
NOVA_DEBUG_LOGGING	T/F (True or False)
NOVA_VERBOSE_LOGGING	T/F (True or False)
OPFLEX_DEBUG_LOGGING	T/F (True or False)
elk_rotation_del_older	Days after which older logs are purged
elk_rotation_frequency	Available options: daily, weekly, fortnightly, monthly
elk_rotation_size	Gigabytes (entry of type float/int is allowed)

log_rotation_frequency	Frequency to rotate the logs on all the servers. Available options: daily, weekly, monthly, yearly
log_rotation_size	Indicates the maximum file size to start the rotation of the log files. Value is a float number + the unit (available options: k, M, G) Examples: 100M indicates that rotation happens when the log file size exceeds 100 Megabytes. 5.1G indicates that rotation happens when the log file size exceeds 5.1 Gigabytes.
log_rotation_del_older	Number of files (already rotated) to keep before deleting the old files
external_lb_vip_cacert	Location of the HAProxy CA certificate
external_lb_vip_cert	Location of the HAProxy certificate
NOVA_RAM_ALLOCATION_RATIO	Mem oversubscription ratio (from 1.0 to 4.0)
NOVA_CPU_ALLOCATION_RATIO	CPU allocation ratio (from 1.0 to 16.0)
ES_SNAPSHOT_AUTODELETE	The auto-deletion of the old snapshots done to to the elasticsearch database can be managed by the following parameters: enabled: [True|False] # Enable/Disable the cronjob in the management node. If the cronjob is disabled, disk space in the management can be consumed by the periodic snapshots made by Curator period: [hourly, daily, weekly, monthly] # Frequency of cron job to check for disk space consumed by the Elasticsearch snapshots threshold_warning: <1-99> # % of disk space occupied to start displaying a warning message. threshold_low: <1-99> # % of disk space occupied after cleaning up snapshots. threshold_high: <1-99> # % of disk space when starting to delete snapshots.

Alternatively, you can regenerate all passwords using regenerate_secrets command option as follows:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --regenerate_secrets
```

In addition to the services passwords, you can change the debug and verbose options for Heat, Glance, Cinder, Nova, Neutron, Keystone and Cloudpulse in */root/openstack-configs/openstack_config.yaml*. You can modify the other configurations including the ELK configuration parameters, API and Horizon TLS certificates, Root CA, NOVA_RAM ALLOCATION_RATIO, NOVA_CPU_ALLOCATION_RATIO, and ES_SNAPSHOT_AUTODELETE. When reconfiguring these options (For Example API and TLS), some control plane downtime will occur, so plan the changes during maintenance windows.

The command to reconfigure these elements are:

```
ciscovim reconfigure
```

The command includes a built-in validation to ensure that you do not enter typos in the secrets.yaml or openstack_config.yaml files.

When reconfiguration of password or enabling of openstack-services fails, all subsequent pod management operations are blocked. In such case, you can contact Cisco TAC to resolve the situation.

From Cisco VIM 3.4.1, you can enable NOVA_RAM_ALLOCATION_RATIO and NOVA_CPU_ALLOCATION_RATIO on a per server basis during Day 0 installation or Day 2 as part of pod management.

- For pod operations, OpenStack uses the service accounts such as admin, cinder, glance, heat, heat_domain_admin, neutron, nova, placement, and cloudpulse. These accounts use passwords to authenticate each other for standard operations. You must not change the password used by these accounts, other than using the ciscovim reconfigure operation. To enforce this behavior, starting Cisco VIM 2.4.5, the "change password" panel is disabled on the Horizon dashboard for these accounts.
- You should create personal OpenStack user accounts for those who need OpenStack admin or member access. You can change the passwords for these accounts through the Horizon dashboard, OpenStack CLI, or OpenStack client interface.

Reconfiguring Glance Client Key for Central Ceph Cluster

From release Cisco VIM 3.0.0, the installation of a central ceph cluster is automated to serve the images to edge pods through Glance service. No local ceph cluster for edge pods as they have constraints on power and space. The edge pods do not need any persistent storage and provide services via a central ceph cluster for glance. For the edge pods to communicate with central ceph cluster using a cluster id, a GLANCE_CLIENT_KEY is required.

Follow the below steps to reset the GLANCE_CLIENT_KEY:

1. Regenerate the client keyring for glance.

   a. On the central ceph cluster, ssh to the management node and execute the following:

   ```
   # ciscovim reconfigure --regenerate_ceph_keyring --setupfile /root/Save<setup_data.yaml> -y
   ```

   Alternatively, you can regenerate the key via the corresponding RestAPI call:

   ```
   # curl -u <user>:<password> -X POST https://<ip|host>:8445/v1/misc --header "Content-Type:
   application/json" -d '{"action": {"reconfigure": "true", "regenerate_ceph_keyring": true}}'
   ```

2. Retrieve the generated client keyring for glance. From the management node, execute the following command to get the cluster UUID:

   ```
   # cat /root/openstack-configs/ceph/fetch/ceph_cluster_uuid.conf
   <cluster_uuid>
   # cat /root/openstack-configs/ceph/fetch/<cluster_uuid>/etc/ceph/ceph.client.glance.keyring
   [client.glance]
   key = <GLANCE_CLIENT_KEY>
   caps mon = "allow r"
   caps osd = "allow class-read object_prefix rbd_children, allow rwx pool=images"
   ```

3. Reconfigure the edge pod to use the new keyring generated on central ceph. SSH to the management node of each edge pod and execute the following:

   ```
   [root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir [root@mgmt1 ~]# cd MyDir
   [root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
   [root@mgmt1 ~]# cp<my_setup_data.yaml> <my_setup_data_original.yaml>
   [root@mgmt1 ~]# vi my_setup_data.yaml (update the GLANCE_CLIENT_KEY with the new info)
   [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
   ```

   4 Optional, but recommended to do it on a handful of pods. Once reconfiguration is done, test if the keyring works by creating and deleting glance images.

Reconfiguring CIMC/BMC Password on Existing Installation

Cisco VIM allows you to reconfigure the CIMC/BMC password on an existing installation. By default, the CIMC/BMC passwords of the servers defined in SERVERS and MANAGEMENT_SERVERS section are updated together.

> ⚠ You must have a Cisco C-series or Quanta-based pod up and running with Cisco to reconfigure the CIMC password.

1. Update the cimc_password in the CIMC-COMMON section, and/or the individual cimc_password for each server and then run the reconfigure option provided by ciscovimclient:

   ```
   CIMC-COMMON:
     cimc_username: "admin"
     cimc_password: <"new password">
     ::
   SERVERS:
     :
     control-server-2:
       cimc_info: {'cimc_ip': '<ip_addr>', 'cimc_username': 'admin', 'cimc_password': <'update with new
   password'>} # only needed if each server has specific password
     :
   ```

   Optionally, update the CIMC password for the servers in the MANAGEMENT_SERVERS section, when CIMC password for those servers are different:

```
MANAGEMENT_SERVERS:
  <hostname>:
    cimc_info: {'cimc_ip': '<ip_addr>', 'cimc_username': 'admin', 'cimc_password': <'update with new
password'>} # only needed if each server has specific password
   :
   :
```

2. To change the CIMC password for the pod, copy the setupdata into a local location and update it manually with the CIMC password as shown in the snippet above. The new password must satisfy atleast 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.
   - One of these special characters !$#@%^-_+=*&
   - Your password has to be 8 to 14 characters long.

   Do not change CIMC password directly into the exiting */root/openstack-configs/setup_data.yaml* file.

3. Run the vim reconfiguration command, to post 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 <my_setup_data.yaml>
[root@mgmt1 ~]# cp <my_setup_data.yaml> <my_setup_data_original.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the relevant CIMC setup_data to include LDAP info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~1]# ciscovim reconfigure --cimc_password --setupfile /root/MyDir/<my_setup_data.yaml>
```

   After successful completion of the CIMC Password, reconfigure operation triggers an auto-back when the management node auto-back recovery feature is enabled. If the CIMC Password reconfigure fails, contact Cisco TAC to recover from the failure.

Reconfiguring Administrator Source Networks

To access the administrator services, Cisco VIM provides source IP based filtering of network requests on the management node. These services include SSH and Kibana dashboard access. When the services are configured, all admin network requests made to the management node are dropped, except the addresses in the allowed list.
Reconfiguration of administrator source network supports the following options:

- Set administrator source network list: You can add or delete the network addresses from the configuration. The entire list is replaced during reconfiguration.
- Remove administrator source network list: If the **admin_source_networks** option is removed, the source address does not filter the incoming admin service requests.

1. Configure the following commands in the setup_data.yaml file:

```
admin_source_networks: # optional, host based firewall to include the allowed list of admin's source IP
- 10.0.0.0/8
- 172.16.0.0/12
```

   Care must be taken while updating the source networks. If the list is misconfigured, operators are locked out of access to the management node through SSH. If it is locked, the operator must log into the management node through the console port to repair the configuration.

2. To initiate the integration, copy the setupdata into a local directory by running the following command:

```
[root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
```

3. Update the setupdata by running the following command:

```
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include Admin Source Network information
```

4. Run the reconfiguration command as follows:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Password Reset for Cisco VIM Management Node

Run the following command to reset the root password of Cisco VIM management node *RHEL-7 / system*

1. Boot your system and wait until the **GRUB2** menu appears.
2. In the **boot loader** menu, highlight any entry and press *e*.
3. Find the line beginning with linux. At the end of this line, append the following:

```
init=/bin/sh
```

   If you face any alarm, instead of *ro,* change *rw* to *sysroot* as shown in the following example:

```
rw init=/sysroot/bin/sh
```

4. Press *Ctrl+X* to boot the system using the options you edited. Once the system boots, you can see the shell prompt without having to enter any user name or password:

```
sh-4.2#
```

5. Load the installed SELinux policy by running the following command:

```
sh-4.2# /usr/sbin/load_policy -i
```

6. Execute the following command to remount your root partition:

```
sh4.2#
mount -o remount,rw /
```

7. Reset the root password by running the following command:

```
sh4.2# passwd root
```

   When prompted, enter your new root password and click *Enter* key to confirm. Enter the password for the second time to ensure that you typed it correctly and confirm with *Enter* again. If both the passwords match, a confirmation message appears.
8. Execute the following command to remount the root partition again, this time as read-only:

```
sh4.2#
 mount -o remount,ro /
```

9. Reboot the system. Now, you can login as the root user using the new password set up during this procedure. To reboot the system, enter *exit*, leave the environment, and reboot the system. For more details on changing root password, see https://access.redhat.com/solutions/918283.

Cloud Settings

Cloud Settings

You can enable specific cloud security hardening settings as a Day 0 or Day n option via reconfiguration. Listed below are the settings that are allowed in Cisco VIM.

```
cloud_settings:

  # Optional, number of incorrect password attempts before the user is locked out.
  # Default is 0 for no lockout.
  # Type: Integer, Allowed range: 0 to 100
  keystone_lockout_failure_attempts: 6

  # Optional, number of seconds for which a user is locked out after exceeding
  # keystone_lockout_failure_attempts.
  # Default is 1800 (30 minutes).
  # Type: Integer, Allowed range: 300 (5 minutes) to 86400 (24 hours).
  keystone_lockout_duration: 1800

  # Optional, number of last passwords to compare with the new one.
  # Default is 0 for no history check.
  # Type: Integer, Allowed range: 0 to 10
  keystone_unique_last_password_count: 5

  # Optional, minimum number of days allowed between password changes.
  # Default is 0 for no limit.
  # Type: Integer, Allowed range: 0 to 7
  keystone_minimum_password_age: 1

  # Optional, number of seconds of inactivity before the Horizon dashboard session
  # is logged out.
  # Default is 1800 (30 minutes).
  # Type: Integer, Allowed range: 300 (5 minutes) to 86400 (24 hours).
  horizon_session_timeout: 1800


  # Optional, ability to disable certain actions typically supported by Nova for instances
  # Default is False for each
  # Type: Boolean, Allowed values: True or False

  # Ability to disable live migration
  nova_disable_vm_live_migration: <True or False> # default is False

  # Ability to disable VM snapshot
  nova_disable_vm_snapshot: <True or False> # default is False

  # Ability to disable VM crashdump
  nova_disable_vm_crashdump: <True or False> # default is False

  # Ability to disable VM console
  nova_disable_vm_console: <True or False> # default is False
```

Open source documentation of cloud settings is available at: https://docs.openstack.org/keystone/latest/admin/configuration.html#security-compliance-and-pci-dss

To initiate the integration of cloud settings on an existing pod, copy the setup data into a local directory and update it manually with the information listed above, and then run the *reconfigure* command as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include cloud_settings related info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Fernet Key Operations

Fernet Key Operations

Keystone fernet token format is based on the cryptographic authentication method - Fernet. Fernet is an implementation of Symmetric Key Encryption. Symmetric key encryption is a cryptographic mechanism that uses the same cryptographic key to encrypt plaintext and the same cryptographic key to decrypt ciphertext. Fernet authentication method also supports multiple keys where it takes a list of symmetric keys, performs all encryption using the first key in a list and attempts to decrypt using all the keys from that list.

The Cisco NFVI pods uses fernet keys by default. The following operations can be carried out in Cisco NFVI pods.

To check if the fernet keys are successfully synchronized across the keystone nodes.

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim help check-fernet-keys
usage: ciscovim check-fernet-keys
Check whether the fernet keys are successfully synchronized across keystone nodes.
```

To forcefully rotate the fernet keys:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim help rotate-fernet-keys
usage: ciscovim rotate-fernet-keys
Trigger rotation of the fernet keys on keystone
```

To resync the fernet keys across the keystone nodes:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim help resync-fernet-keys
usage: ciscovim resync-fernet-keys
Resynchronize the fernet keys across all the keystone nodes
```

Certificates

Certificates

- Managing Certificates
- Reconfiguring TLS Certificates
    - Cisco VIM Rest API Endpoints
    - OpenStack API Endpoints (haproxy)
    - Fluentd Services (client-side certificates)
- Verifying TLS Certificates

Managing Certificates

When TLS protection is configured for the OpenStack APIs, the two certificate files namely *haproxy.pem* and *haproxy-ca.crt,* are stored in the */root /openstack-configs/* directory. To verify cloud authenticity, the clients running on servers outside of the deployed cloud need a copy of the root certificate *(haproxy-ca.crt).* If the installed certificate is signed by a well-known certificate authority , no additional configuration is needed on client servers. However, if a self-signed or local CA is used, copy the *haproxy-ca.crt* to each client. Following instructions are specific to the client operating system or browser, to install the certificate as a trusted certificate.

Alternatively, you can explicitly reference the certificate using the environment variable OS_CACERT or command line parameter –cacert, if OpenStack CLI is used.
While Cisco NFVI is operational, a daily check is made to monitor the expiration dates of the installed certificates. If certificates are not nearing expiration, an informational message is logged. As the certificate approaches expiration, an appropriate warning or critical message is logged.

```
2017-04-24T13:56:01 INFO Certificate for OpenStack Endpoints at 192.168.0.2:5000 expires in 500 day
```

Reconfiguring TLS Certificates

Cisco VIM provides a way to configure TLS certificates on-demand for any reason. For example: certificate expiration policies governing certificate management.
Reconfiguration of certificates in general is supported in the following components:

Cisco VIM Rest API Endpoints

To reconfigure certificate files, follow the below steps:

1. Copy the new key, CA root, and certificate files into the /root/*openstack-configs* folder under the following filenames:

   ⚠ Cisco VIM RestAPI endpoint supports IP based CN (common name) TLS certificate.

   ```
   cp <new-ca-root-cert> /root/openstack-configs/mercury-ca.crt
   cp <new-key-file> /root/openstack-configs/mercury.key
   cp <new-cert-file> /root/openstack-configs/mercury.crt
   ```

   mercury-ca.crt must contain entire trust chain including RootCA and any intermediate CA. *mercury.pem* must contain the TLS certificate and the private key. The private key must not have any passphrase (non-encrypted key).

2. Once copied, run the below reconfiguration command:

   ```
   cd /root/installer-xxxx/tools/
   ./restapi.py -a reconfigure-tls
   ```

3. If the copied certificate in Step 1 is FQDN based, use the following option to reconfigure:

   ```
   cd /root/installer-xxxx/tools/
   ./restapi.py -a reconfigure-tls -d <FQDN>
   For e.g.
   ./restapi.py -a reconfigure-tls -d gg34-bn.cisco.com
   ```

4. To get the FQDN name from certificate, use following command:

```
# openssl x509 -in /root/openstack-configs/mercury-ca.crt -text -noout
. . .
Subject: C=US, ST=California, L=San Jose, O=IT, CN=gg34-bn.cisco.com
```

5. Execute ciscovim reconfigure to deploy TLS certificate to other services like Kibana, and so on.

OpenStack API Endpoints (haproxy)

It is important to replace the certificates before they expire. After Cisco NFVI is installed, you can update the certificates by replacing the haproxy.pem and haproxy-ca.crt files and running the reconfigure command. To reconfigure certificate files, follow the below steps:

1. Copy the new key, CA root, and certificate files into the *~/openstack-configs* folder under the following filenames:

   > ⚠ *haproxy-ca.crt* must contain entire trust chain including RootCA and any intermediate CA. If LDAP certificate for keystone authentication is issued by a different authority than that of haproxy certificate, the entire trust chain (RootCA and any intermediate CA) for LDAP certificate must be appended to the file.
   > *haproxy.pem* must contain the TLS certificate and the private key. The private key must not have any passphrase (non-encrypted key)

   ```
   cp <new-ca-root-cert> /root/openstack-configs/haproxy-ca.crt
   cp <new-cert-file> /root/openstack-configs/haproxy.pem
   ```

2. Once copied, run the below reconfiguration steps:

   ```
   mv /root/openstack-configs/openrc /root/<your_dir>/      # workaround as listed in CSCvt33521
   cd /root/installer-xxxx
   ciscovim reconfigure
   ```

3. For generating third-party certificates, follow the below:

   a. If *external_lb_vip_fqdn* is defined in the *setup_data,* generate the certificate using the *external_lb_vip_fqdn* as DNS.
   b. If *external_lb_vip_ipv6_address* is defined but not *external_lb_vip_fqdn*, generate the certificate using the *external_lb_vip_ipv6_address* as IP Address.
   c. If the *setup_data* contains only *external_lb_vip_address*, but neither *external_lb_vip_ipv6_address* nor *external_lb_vip_fqdn*, generate the certificate using the *external_lb_vip_address* as IP Address.

Fluentd Services (client-side certificates)

This is applicable only when MGMT_TLS is set to True.

1. For the Fluentd-aggregator service on the management node, both the key and certificate file are reconfigured as part of the reconfigure operation.
2. For the Fluentd service on the controllers, compute and storage nodes, the certificate file is reconfigured as part of the reconfigure operation.
3. Copy the key and certificate files to the *~/openstack-configs* folder on the management node, and run the reconfiguration command:

   ```
   cp <new-key-file> /root/openstack-configs/logstash-forwarder.key
   cp <new-cert-file> /root/openstack-configs/logstash-forwarder.crt
   cd ~/installer-xxxx
   ciscovim reconfigure
   ```

Verifying TLS Certificates

Cisco VIM provides a tool to check the expiration date of the installed certificates. If a certificate is expired, you may not be able to access the HTTPS endpoints. Checks are run daily and a syslog message is created if a certificate is nearing expiration.
In addition, a tool is provided to check certificate expiration on demand using the following command:

```
# cd /root/installer-xxxx/tools
# python3 certificate-check.py -help
```

To check all certificates, run the following commands:

```
# cd /root/installer-xxxx/tools
# python3 certificate-check.py
```

To check a single certificate, run the following commands:

```
# cd /root/installer-xxxx/tools
# python3 certificate-check.py -s openstack
```

LDAP AD Support with Keystone v3

LDAP AD Support with Keystone v3

With the introduction of KeystoneV3, the openstack service authentication can be delegated to an external LDAP/AD server. In Cisco VIM, this feature has been introduced optionally if the authorization is done by Keystone v3. Just like Keystonev3, this feature can be enabled on an existing pod running Cisco VIM. To avail this feature post pod deployment, the setup_data needs to be augmented with the following information during the pod installation.

An important pre-requisite for enabling AD/LDAP integration is that the AD/LDAP endpoint MUST be reachable from all the Controller nodes that run OpenStack Keystone Identity Service.

```
LDAP:
domain: <Domain specific name>
user_objectclass: <objectClass for Users> # e.g organizationalPerson
group_objectclass: <objectClass for Groups> # e.g. groupOfNames
user_tree_dn: '<DN tree for Users>' # e.g. 'ou=Users,dc=cisco,dc=com'
group_tree_dn: '<DN tree for Groups>' # e.g. 'ou=Groups,dc=cisco,dc=com'
suffix: '<suffix for DN>' # e.g. 'dc=cisco,dc=com'
url: '<ldap:// host:port>' # e.g. 'ldap://172.26.233.104:389'
or
url: '<ldaps|ldap>://[<ip6-address>]:[port]'
e.g.ldap://[2001:420:293:2487:d1ca:67dc:94b1:7e6c]:389 ---> note the mandatory "[.. ]"
around the ipv6 address
user: '<DN of bind user>' # e.g. 'dc=admin,dc=cisco,dc=com', Optional but need to added
along with password.
password: <password> # e.g. password of bind user, Optional but need to be added along
with DN of bind user.
user_filter = (memberOf=CN=os-users,OU=OS-Groups,DC=mercury,DC=local)
user_id_attribute = sAMAccountName
user_name_attribute = sAMAccountName
user_mail_attribute = mail # Optional
group_tree_dn = ou=OS-Groups,dc=mercury,dc=local
group_name_attribute = sAMAccountName
group_filter: '(&(objectClass=group)(|(cn=server-ops)(cn=admins)))' # Optional
group_member_attribute: memberUid # Optional
group_id_attribute: gidNumber # Optional
group_members_are_ids: True # Optional
chase_referrals: <True or False> # Optional
```

Condition for LDAP user and password parameters are as follows:

- 1 – Can be optional
- 2 – Should be mutually inclusive
- 3 – If defined, it cannot be empty

To initiate the integration of LDAP with Keystone v3 on an existing pod, copy the setupdata into a local directory, update it manually with the relevant LDAP configuration, and then run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include LDAP info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

The reconfigure feature supports a full or partial reconfiguration of the LDAP integration service.

⚠ All the parameters within the LDAP are configurable with the exception of the domain parameter.

Integrating identity with LDAP over TLS: The automation supports keystone integration with LDAP over TLS. To enable TLS, the CA root certificate must be presented as part of the */root/openstack-configs/haproxy-ca.crt* file. The url parameter within the LDAP stanza must be set to ldaps. Additionally, the url parameter supports the following format: url: '<ldaps | ldap>://<FQDN | IP-Address>:[port]' The protocol can be one of the following: ldap for non-ssland ldaps when TLS needs to be enabled.

The ldap host can be a fully-qualified domain name (FQDN) or an IPv4 or v6 address depending on how the SSL certificates are generated.

The port number is optional and if not provided assumes that the ldap services are running on the default ports. For example: 389 for non-ssl and 636 for ssl. However, if these are not the defaults, the non-standard port numbers must be provided. Except for the domain, all other item values can be changed via the 'reconfigure' option.

NetApp from http to https

Moving Netapp transport from http to https

For deployments with NetApp running over http protocol, you can migrate it to https post-deployment.

1. To initiate the change, copy the *setupdata* into a local directory and manually update the name/value pair in the NetApp section:

```
NETAPP:
 ...
 ....
 server_port: 443
 transport_type: https
 ....
 netapp_cert_file: <root ca path for netapp cluster only if protocol is https>
```

2. Execute the following commands, to update the NetApp section:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to netapp section as listed above)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Hardening Cisco VIM Deployment

Hardening Cisco VIM Deployment

If you want to harden the Cisco VIM deployment, setup the firewalls ahead of the external interfaces.

The following table provide information about the expected traffic from the management node interfaces of Cisco VIM.

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
br_api	incoming	HTTPS	TCP	8445	RestAPI	
br_api	incoming	HTTPS	TCP	8008	RestAPI logs	
br_api	incoming	HTTPS	TCP	9000	Unified Management UI	
br_api	incoming	HTTPS	TCP	5601	Kibana	
br_api	incoming	SSH	TCP	22	SSH	
br_api	incoming	HTTPS	TCP	3000	Grafana	
br_api	outgoing	NTP	UDP	123	NTP	
br_api	outgoing	DNS	UDP	53	DNS	
br_api	outgoing	Syslog	UDP	514	Syslog	User configurable. Default value is 514.
br_api	outgoing	HTTPS	TCP	443	Cisco VIM artifacts	Download artifacts from SDS
br_mgmt	incoming	HTTP	TCP	7081	Fluentd-aggr	From all nodes to mgmt node
br_api	outgoing	HTTP	TCP	9090	Prometheus	
br_api	outgoing	HTTP	TCP	9093	Alertmanager	
localhost	incoming / outgoing	HTTP	TCP	1162	SNMP / CVIM_MON	Internal communication between processes
br_api	outgoing	SNMP	UDP	162	SNMP	Userdefined. Default value is 162
br_api	incoming	HTTP	TCP	22	SERVER_MON	From CIMC of the UCS servers.
br_api	incoming	Syslog	UDP	5140	SERVER_MON + Syslog	From CIMC of the UCS servers to the management node
br_api	outgoing	LDAP	TCP	389	LDAP	Default: 389 or defined in setup_data
br_api	outgoing	LDAPS	TCP	636	LDAPS	Default: 636 or defined in setup_data
br_api	outgoing	SSH	TCP	22	CIMC	SSH Access to CIMC
br_api	outgoing	HTTP	TCP	80	CIMC	
br_api	outgoing	HTTPS	TCP	443	CIMC	
br_mgmt	incoming	NFS	UDP	111	Remote NFS backup for ELK	Portmapper to get the port number for nfsd

The following table provide information about the expected traffic from the control node interfaces of Cisco VIM.

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
external_lb_vip	incoming	HTTP	TCP	80	Redirects to 443	
external_lb_vip	incoming	HTTPS	TCP	443	Horizon	
external_lb_vip	incoming	HTTPS	TCP	8774	Nova	
external_lb_vip	incoming	HTTPS	TCP	6080	Nova NoVNC Proxy	
external_lb_vip	incoming	HTTPS	TCP	9696	Neutron	
external_lb_vip	incoming	HTTPS	TCP	8776	Cinder	
external_lb_vip	incoming	HTTPS	TCP	9292	Glance	
external_lb_vip	incoming	HTTPS	TCP	8000	Heat	
external_lb_vip	incoming	HTTPS	TCP	8004	Heat	
external_lb_vip	incoming	HTTPS	TCP	9999	Cloudpulse	
external_lb_vip	incoming	HTTPS	TCP	8777	Ceilometer	
external_lb_vip	incoming	HTTPS	TCP	8041	Gnocchi	
external_lb_vip	incoming	HTTPS	TCP	8778	Placement	

external_lb_vip	incoming	HTTPS	TCP	5000	Keystone	
br_mgmt	outgoing	HTTP	TCP	15672	RabbitMQ monitoring	From management node only
br_mgmt	outgoing	LDAP	TCP	389	LDAP	Default: 389 or defined in setup_data
br_mgmt	incoming	LDAPS	TCP	636	LDAPS	Default: 636 or defined in setup_data
br_mgmt	incoming	HTTP	TCP	7081	Fluentd	To management node

If swift is enabled, see Swift Integration for additional ports that get opened in the controllers to communicate with the iSCSI target.

The following table provide information about the expected traffic from the Cisco VIM Software Hub Server node interfaces of Cisco VIM.

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
br_public	outgoing	NTP	UDP	123	NTP	
br_private	incoming	HTTPS	TCP	443	HTTPD	Browsing artifacts on a web browser and using reverse proxy for docker registry
br_private	incoming	HTTPS	TCP	8441	HTTPD	Cisco VIM pod registration on SDS Software Hub
br_public	incoming	SSH	TCP	22	SSH	For SDS servers without an ADMIN (br_admin) network.
br_admin	incoming	SSH	TCP	22	SSH	For SDS servers with an ADMIN (br_admin) network.

The following table provide information about the expected traffic from the Unified Management node interfaces of Cisco VIM.

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
br_api	outgoing	HTTPS	TCP	8445	Unified Management	Connect to Cisco VIM management node RestAPI
br_api	incoming	HTTPS	TCP	9000	HTTPD	UI
br_api	incoming	SSH	TCP	22	SSH	

Securing Management Node

Securing Cisco VIM Management Node

- Overview
- Constraints
- Enabling Management Node Security
- Operation Details
  - Check Password Age Policies
  - View/Reset Authentication Failure Records
  - View/Unlock Inactive Accounts

After an update to Cisco VIM 3.4.6, ensure that you run a plain reconfiguration if PASSWORD_MANAGEMENT or SSH_ACCESS_OPTIONS settings are defined in setup data prior to update.
Reconfiguration ensures that all the related settings are mapped to Cisco VIM 3.4.6 standards and also fixes the minor caveats reported in previous releases.

Overview

You can match the management node with customer's IT policies from SSH and password protocols point of view. The following section describes various options to track user activity and ways to get out of a block or failed state. Listed below are the optional *setup_entries* used to further harden the management node security.

```
PASSWORD_MANAGEMENT:

  ## Password strength checks:                                                |
  ##-------------------------------------------------------------------------
  #   a. Minimum length of the password must be 8.
  #   b. 5 characters in the new password must NOT be present in the old password.
  #   c. Passwords should satisfy at least three of the following conditions:
  #        - at least 1 letter between a to z
  #        - at least 1 letter between A to Z
  #        - at least 1 number between 0 to 9
  #        - at least 1 other character
  #      Each of the mentioned condition can be called as 'class'
  #   d. Max number of 3 allowed consecutive same characters ('aaa..').
  #   e. Max number of 4 allowed consecutive characters of the same class ('1234..').
  #   f. Username should NOT be a part of the password.

  # Optional, switch to enable/disable Cisco VIM password strength checks.
  # Type: Boolean, Options: True and False, Not enabled by default.
  strength_check: <True or False>


  ## Password expiration policies:
  ##-------------------------------------------------------------------------
  #   a. Not applicable to root user.
  #   b. Force users to change password on first login.
  #   c. max_days can only be used in conjunction with warning_age and vice versa.

  # Optional, max days after which the user is forced to change self password.
  # Type: Integer, Allowed range: 90 to 99999 (default).
  maximum_days: 90

  # Optional, number of days before the expiry of password; users will get expiry
  # notification upon login.
  # Should be less than the value of maximum days.
  # Type: Integer, Allowed range: 1 to 99998, Default: 7.
  warning_age: 7

  # Optional, number of last passwords to compare with new one.
  # Type: Integer, Allowed range: 2 to 12, Not enabled by default.
  history_check: 2
```

```
SSH_ACCESS_OPTIONS:

  # Optional, defaults to 3600 secs
  # Type: Integer, Allowed range: 300 to 3600.
  session_idle_timeout: 3600

  # Optional, only one active SSH session or KVM/serial console.
  # Type: Boolean, Options: True and False. Not enabled by default.
  enforce_single_session: <True or False>

  # Optional, number of login attempts.
  # Type: Integer, Allowed range: 3 to 6, By default not enabled.
  session_login_attempt: 3

  # Optional, lockout session for a duration in seconds for the user who
  # exceeded the session_login_attempt using incorrect passwords.
  # Defaults to 600 secs, when session_root_lockout_duration is defined,
  # otherwise not enabled.
  # Type: Integer, Allowed range: 300 to 86400.
  session_lockout_duration: 300

  # Optional, duration in seconds up to which root user is locked out.
  # if session_login_attempt is exceeded.
  # Type: Integer, Allowed range: 300 to 1800, By default not enabled.
  session_root_lockout_duration: 300

  # Optional, days after which inactive users are locked out.
  # Type: Integer, Allowed range: 90 to 99999, By default not enabled.
  # The accounts that are never logged in are not affected.
  lockout_inactive_users: 90
```

Constraints

Though both PASSWORD_MANAGEMENT and SSH_ACCESS_OPTIONS are optional, the following constraints are applicable:

1. IPA and PASSWORD_MANAGEMENT keys are mutually exclusive.
2. Under the PASSWORD_MANAGEMENT section, the attribute of *maximum_days* can only be used in conjunction with *warning_age* and vice versa.
3. IPA and SSH_ACCESS_OPTIONS keys are mutually exclusive.
4. For SSH_ACCESS_OPTIONS, the attribute of *session_lockout_duration* and *session_root_lockout_duration* can only exist, if the session login attempts exist. The *session_login_attempt* cannot exist, if *session_lockout_duration* or *session_root_lockout_duration* is not defined.
5. If permit_root_login is False and SSH_ACCESS_OPTIONS is enabled, the administrator must take care of using it to avoid lockout from the system for a long duration. If permit_root_login is set to True, the Admin (root) can intervene whenever a user (vim admin) is blocked from the system either due to invalid access or inactivity.
6. The attribute of enforce_single_session under SSH_ACCESS_OPTIONS is applicable when accessing the management node via SSH session or KVM/serial console.
7. The password expiration policy does not apply to the root user. However, the root user needs to abide by the password strength checks while setting/resetting the password for self or others.
8. Enforce single session does not apply to the root user.
9. Careful planning must be done to enforce single session for non-root users, to avoid lockout due to networking issues such as the disconnection of the VPN access in the middle of the SSH session.
10. Password management forces the vim admins to change their password on the first login. This is not applicable when the feature is enabled via reconfiguration, as the vim admins would already be using the system. The vim admins must also change their password, after the expiry of maximum_days from the date on which their password is set.
11. The *lockout_inactive_users* setting is applicable only to users who have logged into the system at least once.
12. Following SSH_ACCESS_OPTIONS subkeys are not allowed when vim_ldap_admins is enabled:

    a. *session_log_attempt*
    b. *session_lockout_duration*
    c. *session_root_lockout_duration*
    d. *lockout_inactive_users*
    e. *enforce_single_session*

Enabling Management Node Security

You can enable this feature on Day 0 as part of the fresh installation or later on via reconfiguration. Once enabled, you cannot disable the feature/attributes.

To enable this feature via reconfiguration, follow the below steps:

1. Take a backup of the setup data file and update the file manually with the configuration listed above.
2. Run the following reconfiguration command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# # update the setup_data to include PASSWORD_MANAGEMENT and/or SSH_ACCESS_OPTIONS
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

Operation Details

As the VIM admins can enter a blocked state due to invalid password usage or inactivity, the root user can track and rectify such activities.

Listed below are some of the operations the root administration can do to get the vim admins out of the bind.

Check Password Age Policies

Admin (root) user can check the password policy for a particular vim admin or vim admins can also check theirs via the below 'chage' utility.

```
# chage -l <vim_admin_user_name>
Last password change                                    : Jan 13, 2020
Password expires                                        : Apr 22, 2020
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 100
Number of days of warning before password expires       : 3
```

View/Reset Authentication Failure Records

The vim_admins cannot access their account for session_lockout_duration if the session login attempts are exceeded. Admin (root) user can intervene if required.

```
Display authentication failure records:
# faillock --user <vim_admin_user_name>
<vim_admin_user_name>:
When                Type  Source                                         Valid
2020-01-15 15:25:17 RHOST 10.65.252.219                                      V
2020-01-15 13:41:31 RHOST 10.65.252.219                                      I
2020-01-15 13:41:34 RHOST 10.65.252.219                                      I
2020-01-15 13:41:53 RHOST 10.65.252.219                                      I
2020-01-15 13:41:58 RHOST 10.65.252.219                                      I
2020-01-15 14:15:11 RHOST 10.65.252.219                                      I
2020-01-15 14:15:16 RHOST 10.65.252.219                                      I
2020-01-15 15:25:20 RHOST 10.65.252.219                                      V


Reset authentication failure records for a vim_admin user:
# faillock --user <vim_admin_user_name> --reset

Reset all authentication failure records:
# faillock --reset
```

View/Unlock Inactive Accounts

The vim_admin account is locked out if the login to the management node is not performed for a predetermined number of days specified against lockout_inactive_users.

```
View user account status:
# passwd -S <vim_admin_user_name>

Reset user passwords:
# passwd <vim_admin_user_name>

Unlock user account:
# passwd -u <vim_admin_user_name>
# su - <vim_admin_user_name>
```

⚠ Usage of any other commands outside the context of *ciscovim* (Cisco VIM CLI) is strictly not allowed.

LDAP Authentication for Kibana

LDAP Authentication for Kibana

- Enabling LDAP Authentication for Kibana
- Enabling LDAP on Kibana Post Cisco VIM Installation

Enabling LDAP Authentication for Kibana

From Cisco VIM 3.4.6, you can access Kibana (in the EFK solution) on the management node using LDAP credentials, if LDAP is enabled on Day 0 as a reconfigurable option. To enable LDAP, configure the connection to the LDAP server and set a valid filter to access Kibana with the relevant LDAP credentials.

The Kibana supports the role of *Viewer* only, as there is no option to modify the existing dashboards in Kibana.

- 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 for Kibana, modify the *setup_data.yaml* file to include a *ldap* section under the KIBANA as given below (replace example values as appropriate):

```
KIBANA:
  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

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

Following are the list of properties used to enable LDAP:

Property	Field Required	Description
search_filter	Mandatory	It is the filter 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 *Viewer.* Optionally, you can add a second group with org_role *Viewer.* You can add multiple LDAP groups mapped to org_role.
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.
root_ca_cert	Optional	Indicates the path to your root CA certificate.
group_search_filter	Optional	Used to search 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 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 member Uid.

Enabling LDAP on Kibana Post Cisco VIM Installation

To enable LDAP on Kibana post Cisco VIM installation via the reconfiguration option, follow the below steps:

1. To initiate the integration, copy the setupdata into a local directory by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
```

2. Update the setupdata by running the following command:

```
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include ladp section under KIBANA
```

3. Run the following reconfiguration command:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Enabling Barbican

Enabling Barbican

- Overview
- Configuring Secret Store Back end
  - Simple Crypto Plugin
  - PKCS#11 Crypto Plugin
- Protecting HSM PIN

Overview

You can optionally enable Barbican during initial installation of Cisco VIM or later on as part of reconfiguration operation. You cannot disable once enabled.

To enable Barbican, add *barbican* to the OPTIONAL_SERVICE_LIST in your setup data file. For example:

```
OPTIONAL_SERVICE_LIST: [heat, barbican]
```

Configuring Secret Store Back end

Barbican stores secrets in a secret store. Cisco VIM supports the following secret store back ends:

- Simple Crypto Plugin
- PKCS#11 (Public-Key Cryptography Standard) Crypto Plugin via HSM

Simple Crypto Plugin

The Simple Crypto Plugin implements encryption in software and stores the encrypted secrets in the Barbican database. To use the Simple Crypto Plugin, simply add *barbican* to the OPTIONAL_SERVICE_LIST. No additional Cisco VIM configuration is required. Cisco VIM automatically generates the encryption key.

PKCS#11 Crypto Plugin

The PKCS#11 Crypto Plugin enables use of a Hardware Security Module (HSM) via the PKCS#11 Cryptographic Token Interface. Cisco VIM supports the ATOS HSM. To use ATOS HSM, place the HSM certificate, key, and software *iso* files in the *installer-xxxxx/openstack-configs/* directory on the management node, and update the information in the setup data file:

```
OPTIONAL_SERVICE_LIST: [barbican]

BARBICAN:
  p11_crypto_plugin:
    p11_hsm_type: atos <required: must be set to 'atos'>
    p11_login: <optional: your HSM PIN, you must provide the value in the BARBICAN_PKCS11_PASSWORD secret if it
is not provided here>
    p11_mkek_label: <required: your mkek label>
    p11_hmac_label: <required: your hmac label>
    p11_slot_id: <deprecated: use p11_token_labels>
    p11_token_labels: <required for p11_hsm_type atos>
      - <required: label of first HSM's token>
      - <required: label of second HSM's token if second HSM is used>
    atos_info:
      atos_iso_file: <required: your HSM software iso file (e.g. Proteccio3.00.03.iso)>
      atos_client_cert_file: <optional: your HSM client certificate file (e.g. hsm_client.crt)>
      atos_client_key_file: <optional: your HSM client key file (e.g. hsm_client.key)>
      atos_server_info:
        - atos_server_ip_address: <required: your HSM server-1 IP address>
          atos_server_cert_file: <optional: your HSM server-1 certificate file (e.g. hsm_server_1.crt)>
        - atos_server_ip_address: <optional: your HSM server-2 IP address>
          atos_server_cert_file: <optional: your HSM server-2 certificate file (e.g. hsm_server_2.crt)>
```

Protecting HSM PIN

When using Barbican with a PKCS#11 HSM back end, you can provide the HSM PIN either in the setup data file attribute *BARBICAN.p11_crypto_plugin.p11_login* or as the user-provided secret BARBICAN_PKCS11_PASSWORD.

To avoid storing the HSM PIN in cleartext, first enable Vault for Cisco VIM secret storage, then prior to Barbican installation, initialize BARBICAN_PKCS11_PASSWORD by running the following command from the installer/tools directory:

```
# cd /root/installer-xxx/tools/
# ./init_secret.py --setupfile=/path/to/setup_data.yaml --password_name=BARBICAN_PKCS11_PASSWORD
```

To change the HSM PIN after installing Barbican, run the following command:

```
# ciscovim reconfigure --setpassword BARBICAN_PKCS11_PASSWORD
```

It is possible to bring in the option of Barbican via reconfiguration. For details on how to enable Barbican via reconfiguration, see General Guidelines.

Enabling Management Network over TLS

Enabling Management Network Over TLS

- Overview
- Assumptions
- Enabling TLS Management Network on Day 0
- Enabling TLS Management Network on Day 2
- Monitoring TLS Certificates
- Regenerating Management TLS Certificates
- Regenerating Management TLS Certificates after Expiry

Overview

From Cisco VIM 4.0.0, you can optionally enable TLS over management network on Day 0 or Day 2. If enabled, you cannot disable TLS. To balance between content and time, you can enable it for the required set of features.

Assumptions

The following assumptions are made when enabling management network over TLS:

- Applicable only for a fullon pod.
- The *external_lb_vip_tls* must be enabled to avail TLS over management network
- The mechanism drivers other than OVS are not supported with management network over TLS.
- All Cisco VIM nodes must be part of Layer 2 adjacent from the management node and not supported with central management VM.
- OpenStack services such as Swift, Ceilometer, SolidFire, Netapp, and Ironic are not supported with management network over TLS.
- The central ceph or nano pod type is not supported with management network over TLS.
- Cisco VIM manages all the certificates for the management network. It creates a pod specific root certificate which is used to sign and verify each certificate on the network. It also creates a certificate for each interface and node on the network, distribute them, and ensure that they are up to date.
- TLS version 1.2 is used for protocol, and the cipher suite is TLS_AES_256_GCM_SHA384.
- Some services like memcache might not run over TLS.
- As the storage network for data replication is not encrypted, use the storage network as a private non-routable network.

Enabling TLS Management Network on Day 0

To enable management network over TLS as part of fresh installation, update the *setup_data* with the following:

```
##########################################################
# TLS on MGMT Network
# When enabled management network will be secured over TLS
# If not defined it will default to False.
##########################################################
#MGMT_TLS: <true or false>
```

Enabling TLS Management Network on Day 2

To initiate the configuration of management network over TLS on an existing installation, copy the setupdata into a local directory, update it manually with the relevant configuration as listed above, and then run the following reconfiguration commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include MGMT_TLS: True)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Monitoring TLS Certificates

Cisco VIM provides a tool that checks the TLS certificates and reports the time remaining until expiration. To perform this check via CLI, run the following command:

```
[root@mgmt1 ~]# cd ~/installer-xxxx/tools
[root@mgmt1 ~]# python certificate-check.py [-s MGMT_TLS]
```

The certificate expiry is monitored automatically by CVIMMON if it is enabled.

Regenerating Management TLS Certificates

The TLS certificates on the management network are generated with validity of three years. An admin can regenerate the TLS certificates, when the certificates are close to the expiry date or for some reasons beyond Cisco VIM control. To regenerate the management network TLS certificates for all the nodes including the self-signing authority, enter the following commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# ciscovim reconfigure --reconfigure_certificates management_certs
```

You cannot reconfigure the external and management TLS certificates simultaneously.

Regenerating Management TLS Certificates after Expiry

If TLS certificates over management network expires, regenerate the certificates using the below command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# ciscovim reconfigure --reconfigure_certificates management_certs --force
```

Usage of Pre-encrypted Images

Usage of Pre-encrypted Images

- Overview
- Generating LUKS Passphrase
- Creating Encrypted Image
- Storing Encrypted Image
- Defining Encrypted Volume Type
- Creating Encrypted Boot Volume
- Booting VM from Encrypted Volume

Overview

From Cisco VIM 4.0.0, the usage of pre-encrypted images is supported by default. To use pre-encrypted images, you must enable Barbican, the OpenStack secret manager. For more details on Barbican, see OpenStack Barbican

Generating LUKS Passphrase

To use a pre-encrypted image, the encryption key or LUKS passphrase for that image must be available in Barbican. Use the Barbican to create and store the encryption key. Ensure that you maintain an one-to-one relationship between encrypted images and keys, that is, use one key to encrypt one image. To create additional key for each image that you want to encrypt, use the following command:

```
# source /root/openstack-configs/openrc

# openstack secret order create key --name 'passphrase' --algorithm aes --bit-length 256
+----------------+------------------------------------------------------------------------------------------------
+
| Field          | Value
|
+----------------+------------------------------------------------------------------------------------------------
+
| Order href     | https://[2001:420:293:2569:10:30:116:24]:9311/v1/orders/52e24671-b679-4dc1-9f36-17611f5be727
|
| Type           | Key
|
| Container href | N/A
|
| Secret href    | None
|
| Created        | None
|
| Status         | None
|
| Error code     | None
|
| Error message  | None
|
+----------------+------------------------------------------------------------------------------------------------
+

# openstack secret order get https://[2001:420:293:2569:10:30:116:24]:9311/v1/orders/52e24671-b679-4dc1-9f36-
17611f5be727
+----------------
+-------------------------------------------------------------------------------------------+
| Field          |
Value                                                                                       |
+----------------
+-------------------------------------------------------------------------------------------+
| Order href     | https://[2001:420:293:2569:10:30:116:24]:9311/v1/orders/52e24671-b679-4dc1-9f36-
17611f5be727  |
| Type           |
Key                                                                                         |
| Container href | N
/A                                                                                         |
| Secret href    | https://[2001:420:293:2569:10:30:116:24]:9311/v1/secrets/9e98b94f-8725-4a45-b375-
67a926513545 |
| Created        | 2020-11-16T03:21:54+00:
00                                                                  |
| Status         |
ACTIVE                                                                                      |
| Error code     |
None                                                                                        |
| Error message  |
None                                                                                        |
+----------------
+-------------------------------------------------------------------------------------------+

# openstack secret get \
  --payload_content_type 'application/octet-stream' \
  --file luks1.key \
  https://[2001:420:293:2569:10:30:116:24]:9311/v1/secrets/9e98b94f-8725-4a45-b375-67a926513545

# xxd -p -c 32 luks1.key
71f3ef71cc10baf3d48caa7a77c043bc549ad1f5e4d7520aa8cabb1604b53eda
```

⚠ Ensure that the length of encryption key is exactly 256 bit or 32 bytes.

Creating Encrypted Image

Use the 32 byte LUKS passphrase to create an encrypted image on a Linux server that is separate from Cisco VIM nodes. It requires a software that is not included in Cisco VIM.

Following is an example of creating an encrypted image based on the CirrOS image.

1. Download the image and convert it from QCOW2 to raw format:

```
# yum install cryptsetup qemu-img
# wget http://download.cirros-cloud.net/0.5.1/cirros-0.5.1-x86_64-disk.img
# qemu-img convert -f qcow2 -O raw \
    cirros-0.5.1-x86_64-disk.img  cirros-0.5.1.raw
```

2. Use *losetup* to find an available loop file, probably */dev/loop0*. When prompted, enter the 32 byte LIKS passphrase to create a LUKS encrypted container:

```
# dd if=/dev/zero of=cirros-luks-container bs=1M count=2000
# losetup /dev/loop0 cirros-luks-container
# cryptsetup luksFormat --type luks1 /dev/loop0

WARNING!
========
This will overwrite data on /dev/loop0 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter passphrase for /root/fordocs/cirros-luks-container:
Verify passphrase:
```

3. Mount the container and copy the CirrOS image into it. Use the following file command to verify the resulting file, *cirros-luks-container*:

```
# cryptsetup luksOpen /dev/loop0 byov
# sudo dd if=cirros-0.5.1.raw of=/dev/mapper/byov
# cryptsetup luksClose byov
# losetup -d /dev/loop0
# file cirros-luks-container
```

Storing Encrypted Image

To store an encrypted Glance image based on the secret created by UUID and the created LUKS container, for example, *cirros-luks-container*, use the following command:

```
# source /root/openstack-configs/openrc
# openstack image create --disk-format raw --container-format bare \
  --file cirros-luks-container --private \
  --property cinder_encryption_key_id=9e98b94f-8725-4a45-b375-67a926513545 \
  --property cinder_encryption_key_deletion_policy=on_image_deletion \
  cirros-encrypted
```

⚠ The encrypted image is stored with a specified name *cirros-encrypted.*

Defining Encrypted Volume Type

You must define a volume type corresponding to LUKS encryption for the creation of encrypted boot volume. This is a one-time setting for a given pod.

```
# source /root/openstack-configs/openrc
# openstack volume type create \
  --encryption-provider nova.volume.encryptors.luks.LuksEncryptor \
  --encryption-cipher aes-xts-plain64 \
  --encryption-key-size 256 \
  --encryption-control-location front-end \
  LUKS
```

Creating Encrypted Boot Volume

You need to create an encrypted volume as you cannot boot the image directly.

1. Get the list of images and the UUID of the encrypted image you wish to use.
2. Create an encrypted volume using the below commands and then boot that volume:

```
# source /root/openstack-configs/openrc
# openstack image list
+--------------------------------------+------------------+--------+
| ID                                   | Name             | Status |
+--------------------------------------+------------------+--------+
| dac93186-2793-47e8-97c1-e8d54a91a9b0 | RHEL-guest-image | active |
| efc0773a-e156-4e41-9b64-11363d43b359 | cirros-encrypted | active |
+--------------------------------------+------------------+--------+

# openstack volume create --image efc0773a-e156-4e41-9b64-11363d43b359 --size 2 --type LUKS cirros-
bootable
```

Booting VM from Encrypted Volume

To boot the VM from encrypted volume, use the following command:

```
source /root/openstack-configs/openrc

openstack server create --flavor <flavor> --nic net-id=<ext-net-name> --volume cirros-bootable cirros-encrypted-
boot
```

Storage

Storage

- Storage Architecture
- Ceph Storage
- Glance
- Cinder
- Nova
- Docker Disk Space Usage

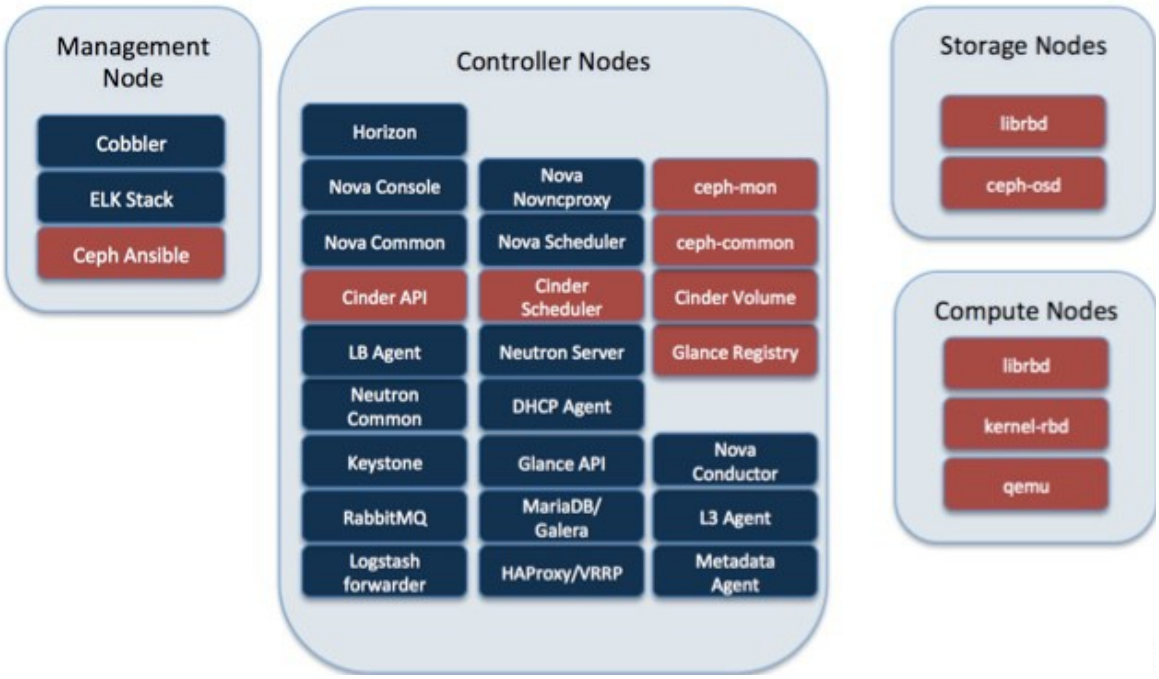
Storage Architecture

Storage Architecture

OpenStack has multiple storage back ends. Cisco NFVI uses the Ceph backend. As Ceph supports both block and object storage, it is used to store VM images and volumes that can be attached to VMs. Multiple OpenStack services that depend on the storage backend include:

- Glance (OpenStack image service): Uses Ceph to store images.
- Cinder (OpenStack storage service): Uses Ceph to create volumes that can be attached to VMs.
- Nova (OpenStack compute service): Uses Ceph to connect to the volumes created by Cinder.

The following figure shows the Cisco NFVI storage architecture component model.



Ceph Storage

Ceph Storage

- Verifying and Displaying Ceph Storage Pools
- Checking the Storage Cluster Health
- Working with Multi-Backend Ceph
- Ceph Storage Expansion

Verifying and Displaying Ceph Storage Pools

Ceph is configured with four independent pools: images, volumes, vms, and backups. (A default rbd pool is used internally.) Each Ceph pool is mapped to an OpenStack service. The Glance service stores data in the images pool, and the Cinder service stores data in the volumes pool. The Nova service can use the vms pool to boot ephemeral disks directly from the Ceph cluster depending on how the NOVA_BOOT_FROM option in the *~/openstack-configs /setup_data.yaml* is configured prior to Cisco NFVI installation.

If NOVA_BOOT_FROM is set to ceph before you run the Cisco NFVI installation, the Nova service boots up from the Ceph vms pool. By default, NOVA_BOOT_FROM is set to local, which means that all VM ephemeral disks are stored as files in the compute nodes. Changing this option after installation does not affect the use of the vms pool for ephemeral disks.

The Glance, Cinder, and Nova OpenStack services depend on the Ceph cluster for backend storage. Therefore, they need IP connectivity to the controller nodes. The default port used to connect Glance, Cinder, and Nova to the Ceph cluster is 6789. Authentication through cephx is required, which means authentication tokens called keyrings, must be deployed to the OpenStack components for authentication.

To verify and display the Cisco NFVI Ceph storage pools:

1. Launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

2. Navigate to the Ceph Monitor container

```
[root@controller_server-1 ~]# cephmon
```

3. List the Ceph pools:

```
cephmon_4612 [root@controller_server-1 ~]# ceph osd lspools
0 rbd,1 images,2 volumes,3 vms,4 backups
```

4. List the images pool content:

```
cephmon_4612 [ceph@controller_server-1 /]$ rbd list images
a4963d51-d3b7-4b17-bf1e-2ebac07e1593
```

Checking the Storage Cluster Health

Cisco recommends that you perform a few verification to determine whether the Ceph cluster is healthy and is connected to the Glance, Cinder, and Nova OpenStack services that have Ceph cluster dependencies. The first task to check the health of the cluster itself by completing the following steps:

1. From the Cisco NFVI management node, launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

2. Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

3. Check the Ceph cluster status:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ ceph status
Sample response:
cluster dbc29438-d3e0-4e0c-852b-170aaf4bd935
health HEALTH_OK
monmap e1: 3 mons at {ceph-controller_server-1=20.0.0.7:6789/0,
ceph-controller_server-2=20.0.0.6:6789/0,ceph-controller_server-3=20.0.0.5:6789/0}
election epoch 8, quorum 0,1,2 ceph-controller_server-3,
ceph-controller_server-2,ceph-controller_server-1
osdmap e252: 25 osds: 25 up, 25 in
pgmap v593: 1024 pgs, 5 pools, 406 MB data, 57 objects
2341 MB used, 61525 GB / 61527 GB avail
1024 active+clean
```

This example displays three monitors, all in good health, and 25 object storage devices (OSDs). All OSDs show as up and in the cluster.

4. To see a full listing of all OSDs sorted by storage node, enter:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ ceph osd tree
Sample response:
ID WEIGHT TYPE NAME UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 60.18979 root default
-2 18.96994 host controller_server-2
1 2.70999 osd.1 up 1.00000 1.00000
5 2.70999 osd.5 up 1.00000 1.00000
6 2.70999 osd.6 up 1.00000 1.00000
11 2.70999 osd.11 up 1.00000 1.00000
12 2.70999 osd.12 up 1.00000 1.00000
17 2.70999 osd.17 up 1.00000 1.00000
20 2.70999 osd.20 up 1.00000 1.00000
-3 18.96994 host controller_server-1
0 2.70999 osd.0 up 1.00000 1.00000
4 2.70999 osd.4 up 1.00000 1.00000
8 2.70999 osd.8 up 1.00000 1.00000
10 2.70999 osd.10 up 1.00000 1.00000
13 2.70999 osd.13 up 1.00000 1.00000
16 2.70999 osd.16 up 1.00000 1.00000
18 2.70999 osd.18 up 1.00000 1.00000
-4 18.96994 host controller_server-3
2 2.70999 osd.2 up 1.00000 1.00000
3 2.70999 osd.3 up 1.00000 1.00000
7 2.70999 osd.7 up 1.00000 1.00000
9 2.70999 osd.9 up 1.00000 1.00000
14 2.70999 osd.14 up 1.00000 1.00000
15 2.70999 osd.15 up 1.00000 1.00000
19 2.70999 osd.19 up 1.00000 1.00000
-5 3.27997 host controller_server-4
21 0.81999 osd.21 up 1.00000 1.00000
22 0.81999 osd.22 up 1.00000 1.00000
23 0.81999 osd.23 up 1.00000 1.00000
24 0.81999 osd.24 up 1.00000 1.00000
```

5. After you verify that the Ceph cluster is in good health, check whether the individual OpenStack components have connectivity and their authentication tokens or keyrings match the Ceph Monitor keyrings. That is, check the connectivity and authentication between Ceph and Glance, Ceph and Cinder, and Ceph and Nova.

Working with Multi-Backend Ceph

The OpenStack component that provides an API to create block storage for cloud is called OpenStack Block Storage service or Cinder. Cinder requires you to configure single backend (by default) or multiple backends.
Cisco VIM supports the following Cinder backends either configured in isolation or parallel.

- Storage nodes full of SSDs disks
- Storage nodes full of HDDs disks

Choosing multi-backend ceph is currently a day-0 option, that is, the cloud administrator must choose single backend or multi-backend storage option at the beginning. It is recommended to have four nodes with a minimum being three nodes for each storage type (HDD or SSD).
To enable support for Cinder multi-backends, update the setup_data.yaml to include the *osd_disk_type* as HDD/SSD under *hardware_info* of the storage server as given below:

```
storage-hdd-server-1: ---I Need minimum of 3; recommended to have 4
 cimc_info: {cimc_ip: <cimc_ip}
 hardware_info: {osd_disk_type: HDD}
 rack_info: {rack_id: RackA}
 storage-ssd-server-1: --I ---I Need minimum of 3; recommended to have 4
 cimc_info: {cimc_ip: <cimc_ip>}
 hardware_info: {osd_disk_type: SSD}
 rack_info: {rack_id: RackB}
```

After successful deployment of Cisco VIM, follow the below steps to create Cinder multi-backends:

1. Log into Horizon of the cloud and navigate to **Admin > Volume > Volume Types.**

    a. Specify the **Name** and **Description** (Optional). The Name can be *volume-ssd* or *volume-hdd* for SSD or HDD, respectively.
    b. Select **View extraspecs** from the drop-down.
    c. Click on **Create** and update the **Key** with **volume_backend_name**.
    d. Enter the **Value** as given below:

    For SSD, the Value is **ceph-ssd**. For HDD, the Value is **ceph**.
2. Create volume from Horizon for each backend type:

    a. Choose **Project/Volumes/Volumes**
    b. Click **CreateVolume**

    Volume name : SSD volume (example)

    Description: Optional

    Volume Source : use the default

    Type : Choose the volume type from the drop-down. It can be volume-ssd/volume-hdd.
3. Attach the volume to instance.

    a. Navigate to **Project/compute/instance**.
    b. Select the instance to be attached the volume.
    c. Select Volume ID from the **Attach volume** drop-down.

Ceph Storage Expansion

From release Cisco VIM 3.0.0, a command expand-storage is available to add disks to expand an already deployed Ceph storage cluster. You can deploy the storage nodes in the Openstack PoD in one of the two ways given below:

- Combination of HDD and SSD drives with Ceph deployed with dedicated journals.
- All SSD drives with Ceph deployed with collocated journals.

The expand-storage command is supported only on storage nodes with a combination of HDD and SSD drives.

You must install disk drives based on how the node is originally deployed. If you have a storage node with a 1 SSD/4 HDDs ratio, insert the disks with the same SSD/HDD ratio for expanding the storage. The expand-storage command looks for the blocks of disks with that ratio during the expansion, and installs one block at a time.

Workflow

- Use ciscovim list-nodes to find a node with a role of *block-storage*.
- Insert a block of disks, based on your storage deployment into the target node.
- Log into the CIMC of the target node and navigate to the storage panel.
- Verify that no disks are reporting errors.
- If any disk reports foreign configuration, clear the configuration.
- Enable JBOD, if RAID controller is used.
- Run cloud-sanity and ensure that no failure occurs.
- Run osdmgmt check-osds to check the state and number of the OSDs currently installed.
- Run expand-storage with the name of the target storage node.
- Run osdmgmt check-osds to check the state and number of the OSDs .
- Compare the outputs of the two osdmgmt check-osd commands and verify whether the new disks areadded as additional OSDs.

The expand storage command runs in the same manner as other ciscovim commands but with various steps executed at a time. The command execution is stopped in case of any failure. You can view the logs once the command execution is complete. The steps for the expand-storage command are:

- Hardware validations
- Baremetal
- CEPH for expansion
- VMTP

Command Help

```
$ ciscovim help expand-storage
usage: ciscovim expand-storage --setupfile SETUPFILE [-y] <node>
Expand storage node capacity
Positional arguments:
<node> Expand Storage capacity of a storage node
Optional arguments:
--setupfile SETUPFILE <setupdata_file>. Mandatory for any POD management
operation.
-y, --yes Yes option to perform the action
```

Workflow command examples:

To expand the storage of node i13-27-test, get the current number/state of the OSDs in the cluster.

```
$ ciscovim list-nodes
+-------------+--------+---------------+---------------+
| Node Name   | Status | Type          | Management IP |
+-------------+--------+---------------+---------------+
| i13-20      | Active | control       | 15.0.0.7      |
| i13-21      | Active | control       | 15.0.0.8      |
| i13-22      | Active | control       | 15.0.0.5      |
| i13-23      | Active | compute       | 15.0.0.6      |
| i13-24      | Active | compute       | 15.0.0.10     |
| i13-25      | Active | block_storage | 15.0.0.11     |
| i13-26      | Active | block_storage | 15.0.0.9      |
| i13-27-test | Active | block_storage | 15.0.0.4      |
+-------------+--------+---------------+---------------+
ciscovim osdmgmt show check-osds --id <id>
+--------------------+-------------+---------------+-----------+---------+
| Message                  | Host        | Role          | Server    | State   |
+--------------------+-------------+---------------+-----------+---------+
| Overall OSD Status | i13-25           | block_storage | 15.0.0.11 | Optimal |
|                            | i13-26      | block_storage | 15.0.0.9  | Optimal |
|                    | i13-27-test | block_storage | 15.0.0.4  | Optimal |
|                            |             |               |           |         |
| Number of OSDs     | i13-25      | block_storage | 15.0.0.11 | 10      |
|                    | i13-26      | block_storage | 15.0.0.9  | 10      |
|                    | i13-27-test | block_storage | 15.0.0.4  | 12      |
+--------------------+-------------+---------------+-----------+---------+
+-------------+--------+--------+----+------------+-----------+-----------
| Host              | OSDs   | Status | ID | HDD Slot   | Path      | Mount
+-------------+--------+--------+----+------------+-----------+-----------
.....
| i13-27-test | osd.2  | up     | 2  | 4 (JBOD)   | /dev/sda1 |
|                  | osd.5  | up     | 5  | 3 (JBOD)   | /dev/sdb1 |
|                   | osd.8  | up     | 8  | 6 (JBOD)   | /dev/sdc1 |
|             | osd.11 | up     | 11 | 2 (JBOD)   | /dev/sdd1 |
|             | osd.14 | up     | 14 | 5 (JBOD)   | /dev/sde1 |
|                  | osd.19 | up     | 19 | 9 (JBOD)   | /dev/sdi1 |
|                  | osd.24 | up     | 24 | 10 (JBOD)  | /dev/sdj1 |
|             | osd.27 | up     | 27 | 8 (JBOD)   | /dev/sdl1 |
|             | osd.28 | up     | 28 | 12 (JBOD)  | /dev/sdm1 |
|             | osd.29 | up     | 29 | 11 (JBOD)  | /dev/sdn1 |
|             | osd.30 | up     | 30 | 13 (JBOD)  | /dev/sdo1 |
|             | osd.31 | up     | 31 | 17 (JBOD)  | /dev/sdp1 |
+-------------+--------+--------+----+------------+-----------+
```

Run the expand-storage command:

```
# ciscovim expand-storage i13-27-test --setupfile setup_data.yaml
Perform the action. Continue (Y/N)Y
Monitoring StorageMgmt Operation
. . . . Cisco VIM Runner logs
The logs for this run are available in
<ip>:/var/log/mercury/05f068de-86fd-479c-afda-c54b14ffdd3e
```

```
##########################################
Cisco Virtualized Infrastructure Manager
##########################################
[1/3][VALIDATION: INIT] [ / ]
0min 0sec
Management Node Validations!
..
Omitted for doc
..
[1/3][VALIDATION: Starting HW Validation, takes time!!!] [ DONE! ]
Ended Installation [VALIDATION] [Success]
[2/3][CEPH: Checking for Storage Nodes] [ DONE! ]
[2/3][CEPH: Creating Ansible Inventory] [ DONE! ]
..
Omitted for doc
..
[2/3][CEPH: Waiting for server to come back first try] [ DONE! ]
Ended Installation [CEPH] [Success]
VMTP Starts
/home/vmtp/.ssh/id_rsa already exists.
Omitted for doc
..
[3/3][VMTP: INIT] [ DONE! ]
Ended Installation [VMTP] [Success]
The logs for this run are available in
<ip>:/var/log/mercury/05f068de-86fd-479c-afda-c54b14ffdd3e
===========
Check the OSDs
ciscovim osdmgmt create check-osds
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| action     | check-osds                           |
| command    | create                               |
| created_at | 2019-01-07T19:00:23.575530+00:00     |
| id              | adb56a08-fdc5-4810-ac50-4ea6c6b38e3f |
| locator    | False                                |
| osd             | None                                                  |
| result     |                                      |
| servers    | None                                 |
| status     | not_run                              |
| updated_at | None                                 |
+------------+--------------------------------------+
ciscovim osdmgmt list check-osds
+--------------------------------------+------------+----------+----------
| ID                                                | Action     | Status   |  Created|
+--------------------------------------+------------+----------+----------
| cd108b85-2678-4aac-b01e-ee05dcd6fd02 | check-osds | Complete | 2019-01-|
| adb56a08-fdc5-4810-ac50-4ea6c6b38e3f | check-osds | Complete | 2019-01-|
+--------------------------------------+------------+----------+----------
ciscovim osdmgmt show check-osds --id <id>
+--------------------+-------------+---------------+-----------+---------+
| Message                | Host            | Role                | Server    | State   |
+--------------------+-------------+---------------+-----------+---------+
| Overall OSD Status | i13-25           | block_storage | 15.0.0.11 | Optimal |
|                            | i13-26           | block_storage | 15.0.0.9  | Optimal |
|                          | i13-27-test | block_storage | 15.0.0.4  | Optimal |
|                            |                       |                    |                |             |
| Number of OSDs     | i13-25           | block_storage | 15.0.0.11 | 10          |
|                             | i13-26            | block_storage | 15.0.0.9  | 10            |
|                             | i13-27-test | block_storage | 15.0.0.4  | 16          |
+--------------------+-------------+---------------+-----------+---------+
+-------------+--------+--------+----+------------+-----------+-----------
| Host              | OSDs   | Status | ID | HDD Slot   | Path      | Mount
+-------------+--------+--------+----+------------+-----------+-----------
....
|i13-27-test  | osd.2  | up     | 2  | 4 (JBOD)   | /dev/sda1 |
|                  | osd.5  | up     | 5  | 3 (JBOD)   | /dev/sdb1 |
|                  | osd.8  | up     | 8  | 6 (JBOD)   | /dev/sdc1 |
|                   | osd.11 | up     | 11 | 2 (JBOD)   | /dev/sdd1 |
|                   | osd.14 | up         | 14 | 5 (JBOD)   | /dev/sde1 |
```

```
|             | osd.19 | up            | 19 | 9 (JBOD)   | /dev/sdi1 |
|             | osd.24 | up     | 24 | 10 (JBOD)  | /dev/sdj1 |
|             | osd.27 | up         | 27 | 8 (JBOD)   | /dev/sdl1 |
|             | osd.28 | up         | 28 | 12 (JBOD)  | /dev/sdm1 |
|             | osd.29 | up         | 29 | 11 (JBOD)  | /dev/sdn1 |
|             | osd.30 | up           | 30 | 13 (JBOD)  | /dev/sdo1 |
|             | osd.31 | up         | 31 | 17 (JBOD)  | /dev/sdp1 |
|             | osd.32 | up         | 32 | 15 (JBOD)  | /dev/sdq1 |
|             | osd.33 | up         | 33 | 14 (JBOD)  | /dev/sdr1 |
|             | osd.34 | up         | 34 | 16 (JBOD)  | /dev/sds1 |
|             | osd.35 | up         | 35 | 7 (JBOD)   | /dev/sdt1 |
+-------------+--------+--------+----+------------+-----------+------------
```

Glance

Glance

- Checking Glance Connectivity
- Verifying Glance and Ceph Monitor Keyrings
- Verifying Glance Image ID on Ceph

Checking Glance Connectivity

The Glance API container must be connected to the Cisco NFVI controller nodes. Complete the following steps to verify the Glance connectivity to controller node:

1. From the management node, examine the IP addresses of controller node:

   ```
   [root@management-server-cisco ~]# cat /root/openstack-configs/mercury_servers_info
   ```

2. From the management node, launch a SSH session to a controller node, for example:

   ```
   [root@management-server-cisco ~]# ssh root@controller_server-1
   ```

3. Navigate to the Glance API container:

   ```
   [root@controller_server-1 ~]# glanceapi
   ```

4. Check the Glance API container connectivity to the storage IP address of the controller node different from the one entered in Step 2:

   ```
   glanceapi_4612 [glance@controller_server-1 /]$ curl <storage_ip_of_another_controller>:6789
   ```

   If the connection is successful, no message is displayed, but you need to do Ctrl+C to terminate the connection:

   ```
   glanceapi_4612 [glance@controller_server-1 /]$ curl 7.0.0.16:6789
   ```

   If the connection is not successful, the following message is shown:

   ```
   glanceapi_4612 [glance@controller_server-1 /]$ curl 7.0.0.16:6789 curl: (7)
   Failed connect to controller_server-2:6789; Connection refused
   ```

   The above message indicates that the Ceph monitor running on the target controller node controller_server-2 is not listening on the specified port or there is no route to it from the Glance API container.
5. To ensure that one connection path is available for the Glance API, checking one controller node is sufficient. As Cisco NFVI controller nodes run as part of an HA cluster, you should run Step 3 above targeting all the controller nodes in the Cisco NFVI pod.
6. After you verify the Glance API connectivity to all Cisco NFVI controller nodes, check the Glance keyring to ensure that it matches with the Ceph monitor keyring.

Verifying Glance and Ceph Monitor Keyrings

Complete the following steps to verify the Glance API keyring matches the Ceph Monitor keyring.

1. Launch a SSH session to a controller node, for example:

   ```
   [root@management-server-cisco ~]# ssh root@controller_server-1
   ```

2. Navigate to the Glance API container:

   ```
   [root@controller_server-1 ~]# glanceapi
   ```

3. Check the Glance keyring content, for example:

```
glanceapi_4612 [glance@controller_server-1 /]$ cat /etc/ceph/client.glance.keyring [client.glance]
key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

4. On the management node, check the CEPH cluster UUID:

```
[root@management-server-cisco ~]# cat /root/openstack-configs/ceph/fetch/ceph_cluster_uuid.conf
0e96e7f2-8175-44b3-ac1a-4f62de12ab9e
```

5. Display the Ceph Glance keyring content:

```
[root@management-server-cisco ~]# cat
/root/openstack-configs/ceph/fetch/0e96e7f2-8175-44b3-ac1a-4f62de12ab9e/etc/ceph/ceph.client.glance.
keyring
[mon.]
key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

6. Verify whether the keyring matches the Glance API keyring displayed in Step 3.
7. To ensure that Ceph and Glance are connected, import a Glance image using Horizon or the Glance CLI. After you import an image, compare the IDs seen by Glance and by Ceph. If the IDs match, it indicates that Ceph is handling the backend for Glance.

Verifying Glance Image ID on Ceph

The following steps verify whether Ceph is properly handling new Glance images, by checking that the image ID for a new Glance image is the same as the image ID displayed in Ceph.

1. From the management node, load the OpenStack authentication variables:

```
[root@management-server-cisco ~]# source ~/openstack-configs/openrc
```

2. Import any Glance image. In the example below, a RHEL 7.1 qcow2 image is used.

```
[root@management-server-cisco images]# openstack image create
"rhel" --disk-format qcow2 --container-format bare --file rhel-guest-image-7.1-20150224.0.x86_64.qcow2
```

3. List the Glance images:

```
[root@management-server-cisco images]# openstack image list | grep rhel
| a4963d51-d3b7-4b17-bf1e-2ebac07e1593 | rhel
```

4. Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

5. Display the contents of the Ceph images pool:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ rbd list images | grep
a4963d51-d3b7-4b17-bf1e-2ebac07e1593
a4963d51-d3b7-4b17-bf1e-2ebac07e1593
```

6. Verify that the Glance image ID displayed in Step 3 matches with the image ID displayed by Ceph.

Cinder

Cinder

- Checking Cinder Connectivity
- Verifying Cinder and Ceph Monitor Keyrings
- Verifying Cinder Volume ID on Ceph

Checking Cinder Connectivity

The Cinder volume container must have connectivity to the Cisco NFVI controller nodes. Complete the following steps to verify that Cinder volume has connectivity to the controller nodes:

1. From the management node, examine the IP addresses of controller node:

   ```
   [root@management-server-cisco ~]# cat /root/openstack-configs/mercury_servers_info
   ```

2. From the management node, launch a SSH session to a controller node, for example:

   ```
   [root@management-server-cisco ~]# ssh root@controller_server-1
   ```

3. Navigate to the Cinder volume container:

   ```
   [root@controller_server-1 ~]# cindervolume
   ```

4. Check the Cinder volume container connectivity to the storage IP address of controller node different from the one entered in Step 1:

   ```
   cindervolume_4612 [cinder@controller_server-1 /]$ curl 7.0.0.16:6789
   ```

   If the connection is successful, no message is displayed, but you need to do a Ctrl-C to terminate the connection:

   ```
   cindervolume_4612 [cinder@controller_server-1 /]$ curl 7.0.0.16:6789
   ```

   If the connection is not successful, the following is displayed:

   ```
   cindervolume_4612 [cinder@controller_server-1 /]$ curl controller_server-2:6789
   curl: (7) Failed connect to controller_server-2:6789; Connection refused
   ```

   The above message indicates that the Ceph monitor running on the target controller node controller_server-2 is not listening on the specified port or there is no route to it from the Cinder volume container.
5. To ensure that one connection path is available for the Cinder volume, it is enough to check one controller node. As the Cisco NFVI controller nodes run as part of an HA cluster, repeat Step 3 targeting all the controller nodes in the Cisco NFVI pod.
6. After you verify the Cinder volume connectivity to all Cisco NFVI controller nodes, check whether the Cinder keyring matches the Ceph monitor keyring.Verifying Cinder and Ceph Monitor Keyrings.

Verifying Cinder and Ceph Monitor Keyrings

Complete the following steps to verify whether the Cinder volume keyring matches with the Ceph Monitor keyring.

1. From the management node, launch a SSH session to a controller node, for example:

   ```
   [root@management-server-cisco ~]# ssh root@controller_server-1
   ```

2. Navigate to the Cinder volume container:

   ```
   [root@controller_server-1 ~]# cindervolume
   ```

3. Check the Cinder keyring content, for example:

```
cindervolume_4612 [cinder@controller_server-1 /]$ cat /etc/ceph/client.cinder.keyring
[client.cinder]
key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

4. On management node, check the CEPH cluster UUID:

```
[root@management-server-cisco ~]# cat /root/openstack-configs/ceph/fetch/ceph_cluster_uuid.conf
0e96e7f2-8175-44b3-ac1a-4f62de12ab9e
```

5. Display the Ceph Cinder keyring content:

```
[root@management-server-cisco ~]# cat
/root/openstack-configs/ceph/fetch/0e96e7f2-8175-44b3-ac1a-4f62de12ab9e/etc/ceph/ceph.client.cinder.
keyring
[client.cinder]
key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

6. Verify whether the keyring matches with the Cinder volume keyring displayed in Step 3.
7. For final verification of Ceph and Cinder connectivity, import a Cinder image using Horizon or the Cinder CLI. After you import the image, compare the IDs seen by Cinder and by Ceph. If the IDs match, it indicates that Ceph is handling the backend for Cinder.

Verifying Cinder Volume ID on Ceph

The following steps verify whether Ceph is properly handling new Cinder volumes by checking that the volume ID for a new Cinder volume is the same as the volume ID displayed in Ceph.

1. From the management node, load the OpenStack authentication variables:

```
[root@management-server-cisco ~]# source ~/openstack-configs/openrc
```

2. Create an empty volume:

```
[root@management-server-cisco ~]# openstack volume create --size 5 ciscovol1
```

The preceding command creates a new 5 GB Cinder volume named ciscovol1.

3. List the Cinder volumes:

```
[[root@management-server-cisco ~]# openstack volume list
+--------------------------------------+-----------+-----------+------+-------------+
| ID                                   | Name      | Status    |
Size | Attached to |
+--------------------------------------+-----------+-----------+------+-------------+
| 3017473b-6db3-4937-9cb2-bd0ba1bf079d | ciscovol1 | available | 5    |                      |
+--------------------------------------+-----------+-----------+------+-------------+
```

4. Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

5. Display the contents of the Ceph volumes pool:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ rbd list volumes
volume-3017473b-6db3-4937-9cb2-bd0ba1bf079d
```

6. Verify whether the Cinder volume ID displayed in Step 3 matches with the volume ID displayed by Ceph, excluding the *volume-* prefix.

Nova

Nova

- Checking Nova Connectivity
- Verifying Nova and Ceph Monitor Keyrings
- Verifying Nova Instance ID

Checking Nova Connectivity

The Nova libvirt container must have connectivity to the Cisco NFVI controller nodes. Complete the following steps to verify Nova has connectivity to the controller nodes:

1. From the management node, examine the IP addresses of controller node:

   ```
   [root@management-server-cisco ~]# cat /root/openstack-configs/mercury_servers_info
   ```

2. From the management node, launch a SSH session to a controller node, for example:

   ```
   [root@management-server-cisco ~]# ssh root@Computenode_server-1
   ```

3. Navigate to the Nova libvirt container:

   ```
   [root@compute_server-1 ~]# libvirt
   ```

4. Check the Nova libvirt container connectivity to the storage address of the controller node different from the one entered in Step 1:

   ```
   novalibvirt_4612 [root@compute_server-1 /]$ curl 7.0.0.16:6789
   ```

   If the connection is successful, no message is displayed, but you need to do a Ctrl-C to terminate the connection:

   If the connection is not successful, a message is displayed as follows:

   ```
   novalibvirt_4612 [root@compute_server-1 /]$ curl 7.0.0.16:6789
   curl: (7) Failed connect to controller_server-1:6789; Connection refused
   ```

   The above message indicates that the Ceph monitor running on the target controller node controller_server-1 is not listening on the specified port or there is no route to it from the Nova libvirt container.
5. To ensure that one connection path is available for the Nova libvirt, checking one controller node is sufficient. As Cisco NFVI controller nodes run as part of an HA cluster, you should run Step 3 above targeting all the controller nodes in the Cisco NFVI pod.
6. After you verify the Nova libvirt connectivity to all Cisco NFVI controller nodes, check the Nova keyring to ensure that it matches with the Ceph monitor keyring.

Verifying Nova and Ceph Monitor Keyrings

Complete the following steps to verify whether the Nova libvirt keyring matches with the Ceph Monitor keyring.

1. From the management node, launch a SSH session to a controller node, for example:

   ```
   [root@management-server-cisco ~]# ssh root@compute_server-1
   ```

2. Navigate to the Nova libvert container:

   ```
   [root@compute_server-1 ~]# libvirt
   ```

3. Extract the libvirt secret that contains the Nova libvirt keyring:

```
novalibvirt_4612 [root@compute_server-1 /]# virsh secret-list
UUID                                                                  Usage ...
------------------------------------------------------------------
b5769938-e09f-47cb-bdb6-25b15b557e84 ceph client.cinder
```

4. Get the keyring from the libvert secret:

```
novalibvirt_4612 [root@controller_server-1 /]# virsh secret-get-value
b5769938-e09f-47cb-bdb6-25b15b557e84 AQBApY1XQCBBEBAAroXvmiwmlSMEyEoXKl/sQA==
```

5. On management node, check the CEPH cluster UUID:

```
[root@management-server-cisco ~]# cat /root/openstack-configs/ceph/fetch/ceph_cluster_uuid.conf
0e96e7f2-8175-44b3-ac1a-4f62de12ab9e
```

6. Display the Ceph Cinder keyring content:

```
[root@management-server-cisco ~]# cat
/root/openstack-configs/ceph/fetch/0e96e7f2-8175-44b3-ac1a-4f62de12ab9e/etc/ceph/ceph.client.cinder.
keyring
[client.cinder]
key = AQBApY1XQCBBEBAAroXvmiwmlSMEyEoXKl/sQA==
```

7. Verify whether the keyring matches with the Nova libvirt keyring displayed in Step 3.

> ⚠ In the above example, the Cinder keyring is checked even though this procedure is for the Nova libvirt keyring. This occurs because the Nova services need access to the Cinder volumes and hence authentication to Ceph uses the Cinder keyring.

8. Complete a final check to ensure that Ceph and Nova are connected, by attaching a Nova volume using Horizon or the Nova CLI. After you attach the Nova volume, check the libvirt domain.

Verifying Nova Instance ID

From the management node, complete the following steps to verify the Nova instance ID of a guest VM having a cinder volume attached:

1. Load the OpenStack authentication variables:

```
[root@management-server-cisco installer]# source ~/openstack-configs/openrc
```

2. List the Nova instances:

```
[root@management-server-cisco images]# nova list
+--------------------------------------+------------+--------+--------
| ID                                                                    | Name                | Status
| Task
+--------------------------------------+------------+--------+--------
| 77ea3918-793b-4fa7-9961-10fbdc15c6e5 | cisco-vm         | ACTIVE | -
+--------------------------------------+------------+--------+-------
```

3. Show the Nova instance ID for one of the instances:

```
[root@management-server-cisco images]# nova show
77ea3918-793b-4fa7-9961-10fbdc15c6e5 | grep instance_name
| OS-EXT-SRV-ATTR:instance_name | instance-00000003
```

The Nova instance ID in this example is instance-00000003. This ID is used later with the virsh command. Nova instance IDs are actually the libvirt IDs of the libvirt domain associated with the Nova instance.

4. Identify the compute node where the VM is deployed:

```
[root@management-server-cisco images]# nova show 77ea3918-793b-4fa7-9961-10fbdc15c6e5 | grep
hypervisor
| OS-EXT-SRV-ATTR:hypervisor_hostname | compute_server-1
```

The compute node in this case is compute_server-1. Connect to this compute node to call the virsh commands.

Next, you can get the volume ID from the libvirt domain in the Nova libvirt container.

5. Launch a SSH session to the identified compute node, compute_server-1:

```
[root@management-server-cisco ~]# ssh root@compute_server-1
```

6. Navigate to the Nova libvirt container:

```
[root@compute_server-1 ~]# libvirt
```

7. Get the instance libvirt domain volume ID:

```
novalibvirt_4612 [root@compute_server-1 /]# virsh dumpxml instance-00000003 | grep rbd
<source protocol='rbd' name='volumes/volume-dd188a5d-f822-4769-8a57-c16694841a23'>
```

8. Launch a SSH session to a controller node:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

9. Navigate to the Ceph monitor container:

```
[root@compute_server-1 ~]# cephmon
```

10. Verify whether the volume ID matches the ID in Step 7:

```
cephmon_4612 [ceph@controller_server-1 ceph]
$ rbd list volumes | grep volume-dd188a5d-f822-4769-8a57-c16694841a23
volume-dd188a5d-f822-4769-8a57-c16694841a23
```

Docker Disk Space Usage

Displaying Docker Disk Space Usage

Docker supports multiple storage backends such as Device Mapper, thin pool, overlay, and a Union File System (AUFS). Cisco VIM uses the devicemapper storage driver as it provides strong performance and thin provisioning. Device Mapper is a kernel-based framework that supports advanced volume management capability.

Complete the following steps to display the disk space used by Docker containers.

1. Launch a SSH session to a controller or compute node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

2. Enter the *docker info* command to display the disk space used by Docker containers:

```
[root@controller_server_1 ~]# docker info
Containers: 24
Images: 186
Storage Driver: devicemapper
Pool Name: vg_var-docker--pool
Pool Blocksize: 524.3 kB
Backing Filesystem: xfs
Data file:
Metadata file:
Data Space Used: 17.51 GB
Data Space Total: 274.9 GB
Data Space Available: 257.4 GB...
```

Monitoring with CVIM-MON

Monitoring with CVIM-MON

- Alerting Rules Customization
- Alert Manager and Receiver Customization
- Silencing Alerts

Alerting Rules Customization

Alerting Rules Customization

- Managing Alerting Rules
- Alerting Rules Customization Workflow
- Custom Alerting Rule File Format
- Addition of Alerting Rules
- Modification of Alerting Rules
- Deletion of Alerting Rule

Cisco VIM monitor is deployed with a set of built-in alerting rules that cover the most important error conditions that can occur in the pod. 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.

Managing Alerting Rules

The alerting rules define how alerts should be triggered based on conditional expressions on any available metric. For example, you can trigger an alert when any performance metric such as CPU usage, network throughput or disk usage reaches certain threshold.
You can add new alerting rules and modify or delete the pre-built existing alerting rules by following the below steps:

1. Create a proper custom alerting rules configuration file:
    a. Create a custom alerting rule configuration file named alerting_custom_rules.yml under the management node openstack-configs directory.
    b. Add the new rules, modified rules and deleted rules in that file using your favorite editor (see the file format below)
    c. Verify that the custom alerting rule file is valid using a provided tool.
2. Once validated, if needed, you can rename it and issue a standard reconfiguration using the ciscovim cli.

Alerting Rules Customization Workflow

Once the custom alerting rules file is validated, you must merge the two files such as default pre-existing alerting rules and custom alerting rules. Whenever a reconfigure operation with --alerting_rules_config option is performed, a merge tool starts with the default alerting rules as a base for all customizations. Any previous modifications are overwritten based on the current content of alerting_custom_rules.yml file. The merge tool output file consists of:

1. All rules from alerting_custom_rules.yml that do not belong to group change-rules or delete-rules.
2. Rules from default_alerting_rules.yml:
    - that do not duplicate from custom file rules.
    - that are not supposed to be deleted.
    - that can be modified based on change-rules input.

Custom Alerting Rule File Format

The *alerting_custom_rules.yml* file must follow the format defined in this section. This format is identical to the one used by the Prometheus configuration file, with a few additional semantic extensions to support deletion and modification of pre-built existing rules.

General Format

The group entry contains a list of groups identified by (group_name), where each group can include one or more rules. Use the labels to determine the severity and other snmp trap attributes.
Following are the limitations to set labels:

- severity, snmp_fault_code, and snmp_fault_severity must be set to one of the values specified in the example below.
- snmp_fault_source should indicate the metric used in the alert expression
- snmp_node must not be changed.
- snmp_podid must be same as the pod name specified insetup_data.yaml

```
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}
```

Addition of Alerting Rules

Any alert rule specified under a group other than **change-rules** group or **delete-rules** group is populated to the merged output file. You can prioritize the custom rules over the pre-existing rules if there are two alerts with the same name in both the files, such that only the one from custom file is kept as a result of the merge.

Modification of Alerting Rules

You can modify any pre-existing rule using the following syntax:

```
groups:
- name: change-rules
rules:
- alert: {alert_name}
expr: {new_alert_expression}
annotations:
summary: {new_alert_summary}
```

The above merge script finds only the group named **change-rules** and modifies the expression and/or summary of the corresponding alert.
If the alert to be changed does not exist, it will not be created and no changes are performed.

Deletion of Alerting Rule

You can delete any built-in rule using the following construct:

```
groups:
        name: delete-rules
        rules:
        alert: {alert_name/regular_expression}
```

The above merge script finds only the group named **delete-rules** and deletes the pre-existing rules that match the alert name or regular expressions.

If the alert to be deleted does not exist, no changes are performed.

Example

The following custom configuration file includes examples of new alerting rule, modified alerting rule and deleted alerting rules:

```
groups:
- name: cpu
rules:
- alert: cpu_idle
annotations:
description: CPU idle usage is too high - resources under-utilized
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.*
```

Validation Script

Validate any custom configuration file prior to reconfiguration, by executing the following CLI command from any location on the management node:

```
check_alerting_rules (no additional parameters are required)
```

The validation script uses the prometheus "promtool", but skips some of its checks to allow the modification and deletion of rules. It also checks if the provided SNMP severities and fault codes are supported. When no custom file is present, the expected location is mentioned in the output.

Ouput of validation script in case of success

```
check_alerting_rules
check_promtool.py: checking /prometheus/alerting_custom_rules.yml
check_promtool.py: success:
check_promtool.py: regular expressions for rules to be deleted: 2
check_promtool.py: rules to be changed: 3
check_promtool.py: rules to be added: 2
```

Output of validation script in case of failure

```
# check_alerting_rules
check_promtool.py: checking custom-rules.yml
check_promtool.py: failure:
check_promtool.py: group "new_group", rule 0, "new_alert": could not parse expression:
parse error at char 8:
could not parse remaining input "@$"...
check_promtool.py: group "new_group2", rule 0, "new_alert_3": could not parse expression:
parse error at char 7:
bad number or duration syntax: "1"
# check_alerting_rules
check_promtool.py: checking /prometheus/alerting_custom_rules.yml
check_promtool.py: failure:
check_promtool.py: line 36: field custom_field not found in type rulefmt.Rule
```

Alert Manager and Receiver Customization

Alert Manager and Receiver Customization

- Supported Receivers
- Alert Manager Custom Configuration File Format
- SNMP Trap Receivers

The alert manager component in Cisco VIM monitor controls the routing, grouping, and inhibiting alerts that are sent by the Prometheus alert rules engine to the appropriate receivers.
The default configuration in Cisco VIM monitor allows every alert to be forwarded as SNMP traps to the SNMP managers through SNMP agent if enabled in the Cisco VIM configuration file.
After deployment, you can add custom alert routes, alert grouping, alert inhibitions and receivers by following the below steps:

1. Create a proper custom alerting rules configuration file:
    a. Create a custom alert manager rule configuration file named alertmanager_custom_config.yml.
    b. Edit the content using your favorite editor (see format below).
    c. Copy that file to the management node openstack-configs directory
    d. Verify that the custom alerting rule file is valid using a provided tool.
2. Once the file is validated, if needed, you can either leave it in *openstack-configs* directory or move it to your preferred location. Then use a **ciscovim reconfigure** command with an additional parameter:

```
[root@mgmt1 ~]# ciscovim reconfigure --alertmanager_config <alertmanager_config_file>
```

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 should only include 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 should contain three main keys (global, route, 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 can be 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).

```
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:

1. Generic route for all alerts to the SNMP agent running on the management node.
2. 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 Cisco VIM configuration file setup_data.yaml.

Example: inhibit (mute) alerts matching a set of labels

Inhibit alerts is a tool that prevents certain alerts to be trigged if other alert/alerts is/are trigged. If one alert having the target attribute matches with the another alert having 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 (alert docker_container_down).

```
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 the check_alertmanager_config from anywhere in the management node and ensure that you get a **SUCCESS** in the output from the configuration POV.

```
> check_alertmanager_config
Checking '/var/lib/cvim_mon/alertmanager_custom_config.yml' SUCCESS
Found:
- global config
- route
- 0 inhibit rules
- 1 receivers
- 0 templates
```

Silencing Alerts

Silencing Alerts

- Silences
- Credentials and Certificates
- Managing Silences Using Web UI
    - Create a Silence
    - View, Edit, or Expire a Silence
- Managing Silences using REST API
    - Create a Silence Using HTTP
    - Retrieve Silences
    - Delete a Silence

Silences

During administrative pod operations, it is sometimes necessary to temporarily disable alerting as these operations can trigger alerts. The alerts caused by these operations can be silenced to avoid unnecessary escalation in applications that receive and handle alerts.

CVIM-MON provides the ability to silence alerts based on arbitrary filters on the alert labels by creating silences. A silence is a filter programmed in the alert manager component with the following attributes:

- start: indicates when the silence starts to be active
- end: indicates when the silence is no longer active
- duration: an alternate way to specify the duration of the silence (might be easier than programming an end date)
- matchers: a list of one or more filters on alert labels. Each matcher has a label name, a label value, and a regex flag which indicates whether the label value is a regular expression or an exact match.
- creator: arbitrary text identifying the silence creator
- comment: arbitrary text describing the purpose of the silence

When a silence is active, any alert that matches the filters are effectively silenced such that:

- these alerts should not be counted in the Grafana UI alert counters
- they should not be visible in the Grafana UI alert table
- no receiver should receive these alerts (SNMP traps, webhooks)

The administrator can create as many silences as required, list silences, expire them, or delete them using two different interfaces: web UI or REST API.

Credentials and Certificates

The username and password can be:

- *admin* and the static auto-generated password available on the management node in the *openstack-configs/secrets.yaml* with the property name CVIM_MON_SERVER_PASSWORD
- LDAP user credentials associated to the Prometheus/Alertmanager role

In the case of HTTP access, a certificate file is required to certify that you are connecting to the right server.
Depending on the way you generate and manage the certificates, you may or may not need to do anything specific to use the right certificate.
If you use self-signed certificates, you need to download it on the server that will connect to the alert manager.
You can download the self-signed certificates from the target management node at the following location:

```
/var/www/mercury/mercury.crt
```

Managing Silences Using Web UI

Point your browser to the CVIM-MON Alert Manager UI which is available at the following URL:

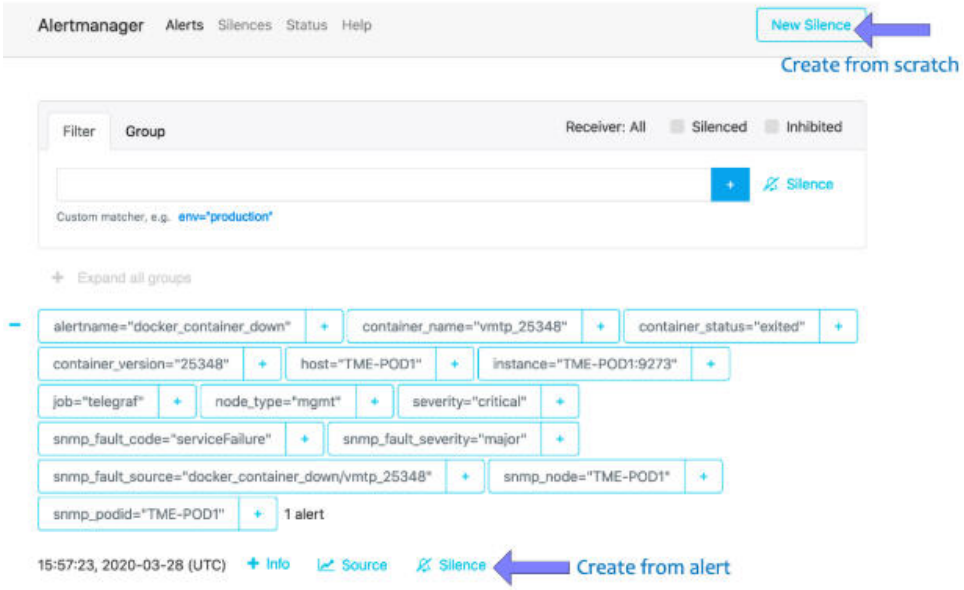
```
https://<mgmt_node_ip>:9093
```

The username is *admin* and the password is the same password as for accessing the Prometheus service.

The landing page shows the list of all standing alerts.

Create a Silence

1. To create a new silence, click **New Silence** on the top right corner of the dashboard or click on **Silence** on the **Filter** pane for the existing standing alert. The **Silence Create** page is displayed.

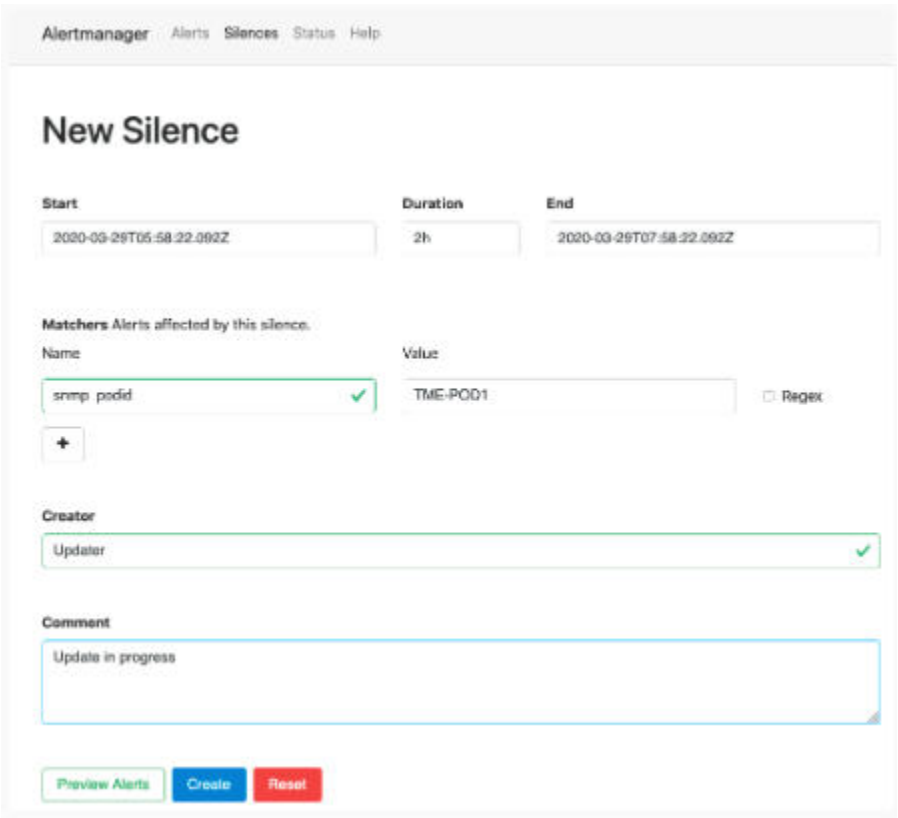


2. Enter the mandatory fields such as start time, duration or end time.
3. Edit the filters if it is created from an alert (mostly removing unneeded filters) or add the necessary filters, a creator and comment, and then click **C reate**.
4. You can view, edit, or set expiry for the silences from the **Silences** tab.

Example: Silencing all alerts from the current pod

To silence all alerts from the local pod preceding a software upgrade, for example, you can define a silence with just one filter which matches on the label *s nmp_podid* and set the value to the local pod name, where the pod name is defined in the *openstack-configs/setup_data.yaml*. You can set the duration to any arbitrary value or a high value and expire the silence later at the right time.

Example of silence for the pod named TME_POD1:



View, Edit, or Expire a Silence

You can view, edit, reactivate or set expiry for silences from the **Silences** tab.

Each selected silence shows the count and list of affected alerts.

Managing Silences using REST API

The alert manager service can also handle silence operations using the REST API. This interface allows easier implementation of any alert silencing policy through script.

The HTTP requests must be sent from an external server with the appropriate credentials and certificates.

Create a Silence Using HTTP

To create an alert silence, send an HTTP POST request with the following URL:

```
https://<mgmt_node_ip>:9093/api/v2/silences
```

The body of the POST request is a JSON string with the following content:

```
{
   "comment": <silence description>,
   "createdBy": <silence creator>,
   "startsAt": <date_time>,
   "endsAt":   <date_time>,
   "matchers": [{"isRegex": <true|false>,"name": <label name>,"value": <label value>},
                {"isRegex": <true|false>,"name": <label name>,"value": <label value>},
                ...]
}
```

The *startsAT* and *endsAT <date_time>* values must comply to RFC-3339 (Date and Time on the Internet: Timestamps).

For example "`2019-07-20T18:01:30.00Z`" represents July 20, 2019 1:30am in UTC.
The label name must correspond to a label associated to the alert that needs to be silenced.
The *isRegex* field indicates whether a regular expression is used in the label value.
You can provide as many matchers as needed, but the alerts that are silenced must match all matchers (that is, they are AND not OR).

Each silence is uniquely identified by its silence ID, which is a UUID assigned by the alert manager and returned by the POST operation.

Example of curl request to create a silence that disables all alerts from the pod TME-POD1 and have a critical severity:

```
$ curl -sS -H "Content-Type: application/json" -u admin:8EavKsAK4IdHZrnO --cacert ./certificates.crt
https://172.23.105.218:9093/api/v2/silences -X POST -d '{"comment": "test sample silence","createdBy":
"tester","startsAt": "2019-07-20T18:00:00.00000000Z", "endsAt": "2019-07-20T18:00:59.46418637Z","matchers":
[{"isRegex": false,"name": "snmp_podid","value": "TNE-POD1"},{"isRegex": false,"name": "severity","value":
"critical"}]}'


{"silenceID":"6ae6e8c5-9de8-44c1-8cd7-2e3da825a786"}
$
```

Retrieve Silences

You can retrieve silences using an HTTP GET operation.

Example of silence retrieval

```
$ curl -sS -u admin:8EavKsAK4IdHZrnO --cacert ./certificates.crt https://172.23.105.218:9093/api/v2/silences |
python -m json.tool
[
    {
        "comment": "test",
        "createdBy": "devops",
        "endsAt": "2019-06-09T17:45:40.814Z",
        "id": "95ba1628-4d24-462b-9a68-a325980f8ef7",
        "matchers": [
            {
                "isRegex": false,
                "name": "alertname",
                "value": "diskmon_physical_drive_state"
            },
            {
                "isRegex": false,
                "name": "snmp_podid",
                "value": "bn-champagne"
            }
        ],
        "startsAt": "2019-06-09T15:46:09.350Z",
        "status": {
            "state": "active"
        },
        "updatedAt": "2019-06-09T15:46:09.350Z"
    }
]
```

You can also retrieve a silence from its silence ID by appending the ID to the URL:

```
https://<mgmt_nod_ip>:9093/api/v2/silence/<silence_ID>
```

Example of silence retrieval from its ID

```
$ curl -sS -u admin:8EavKsAK4IdHZrnO --cacert ./certificates.crt https://172.23.105.218:9093/api/v2/silence
/95ba1628-4d24-462b-9a68-a325980f8ef7 | python -m json.tool
{
    "comment": "test",
    "createdBy": "devops",
    "endsAt": "2019-06-09T17:45:40.814Z",
    "id": "95ba1628-4d24-462b-9a68-a325980f8ef7",
    "matchers": [
        {
            "isRegex": false,
            "name": "alertname",
            "value": "diskmon_physical_drive_state"
        },
        {
            "isRegex": false,
            "name": "check",
(---SNIP---)
```

Delete a Silence

To delete a silence, use the delete HTTP request on the following URL:

```
https://<mgmt_nod_ip>:9093/api/v2/silence/<silence_ID>
```

Example of silence deletion using curl

```
$ curl -sS -u admin:8EavKsAK4IdHZrnO --cacert ./certificates.crt -X DELETE https://172.23.105.218:9093/api/v2
/silence/95ba1628-4d24-462b-9a68-a325980f8ef7
```

CVIM Pod Migration from Micro to Full

Seamless Migration of Cisco VIM Pod from Micro to Full

- Overview
- Assumptions
- High-Level Method of Procedure (MOP)
  - Pre-migration Check
  - Migration from MicroPod to Full Pod
  - Post-Migration check

Overview

You can switch the cloud installation from Micropod to a full pod, as it is more economical. For the transition to happen, VMs must continue to run on the dedicated computes.

Assumptions

Following assumptions need to hold before migrating Cisco VIM from a Micropod to a full pod:

- Depending on the amount of storage used, the migration may take time and must be planned over multiple back-to-back maintenance releases.
- None of the AIO nodes can have any active VMs running on them. Adequate pre-planning is required to proactively move the VMs on the AIO nodes to standalone compute nodes.
- Three standalone Ceph nodes are needed, with its BOM matching that of the pod in terms of NICs and OSDs.
- No other pod management activity must happen during the migration.

High-Level Method of Procedure (MOP)

Pre-migration Check

1. Adequate pre-checks must be done to ensure that the pod is not in *single-fault* state.
    - Re-run Cloud Sanity and confirm that it is passing.
    - Confirm that *ciscovim last-run-status* is good.
    - Confirm that cloud-pulse is indicating that all tests are passing. For more details on cloud-pulse, see Monitoring Cisco NFVI Health.
    - Confirm that *ciscovim run --perform 1* is passing.
    - Confirm that *ciscovim run --perform 3* is passing.
2. Ensure that management and storage ip are statically defined. If not statistically defined, run the following from the */root/intaller-xxx/tools /update_static_addrs.sh*
3. Ensure that all VMs are running off the standalone nodes only.
4. Ensure that all the three standalone Ceph nodes are in place and connected to the switches.
5. Verify if BOM of the three standalone Ceph node matches with that of the pod in terms of NICs and OSDs.

Migration from MicroPod to Full Pod

1. Repeat the following three times:
    a. Handle the addition of one standalone Ceph node followed by replacement of one controller from AIO to standalone node.
    b. Create the *setup_data* with one additional Ceph node and run offline validation.
    c. Assuming that the offline validation passes, run the add storage with *skip vmtp* option:

```
ciscovim add-storage --setupfile <setup_data_with_add_1_storage> <hostname_of_new_ceph_node> --
skip_vmtp
```

    d. Post add storage, check and wait until *ceph* -s is in HEALTH_OK state and all ceph OSD are in *up* status (via ceph osd tree).
    e. Update the BOM of the controller to that of a standalone controller (2x1.2TB HDD), and remove the drives participating in Ceph: 1 SSD and OSD drives.
    f. Create the *setup_data* to replace one of the controllers as a standalone controller, and run the offline validation.
    g. Execute replace controller via *ciscovim* on the target controller with *skip vmtp* option:

```
ciscovim replace-controller --setupfile <setup_data_with_standalone_controller> --skip_vmtp
<hostname_of_target_controller>
```

    h. Check and wait until the *ceph -s* is in HEALTH_OK state and all ceph OSD are in up status (via ceph osd tree).
2. After the migration is complete, update the *setup_data* as follows:
    a. Change PODTYPE to *fullon* from *Micro*

b. Remove CEPH_OSD_RESERVED_PCORES entry if it exists.
c. Copy the the *setup_data.yaml* and *.backup_setup_data.yaml,* and restart *rest_api.py.*

3. Update internal database for nodes to standalone:

   a. List the nodes and its corrsponding roles:

   ```
   ciscovim list-nodes
   ```

   b. Get the controller hostnames via:

   ```
   mysql -u root -p<RestDB-Password> cvimdb -e 'select name,mtype from node;'

   # The RestDB-Password information can be obtained from /opt/cisco/ui_config.json
   ```

   c. Update the role of each controller to that of standalone control via:

   ```
   mysql -u root -p<RestDB-Password> cvimdb -e "update node set mtype='control' where
   name='<controller_hostname>'"

   # The RestDB-Password information can be obtained from /opt/cisco/ui_config.json
   ```

   d. Restart Cisco VIM *rest_api* server:

   ```
   systemctl status mercury-restapi
   systemctl restart mercury-restapi
   systemctl status mercury-restapi
   ```

   e. List the nodes and its corrsponding roles:

   ```
   ciscovim list-nodes
   ```

Post-Migration check

1. Re-run Cloud Sanity and confirm it is passing.
2. Confirm if *ciscovim last-run-status* is good.
3. Confirm if *cloud-pulse* is indicating that all tests are passing. For more details on cloud-pulse, see Monitoring Cisco NFVI Health.
4. Confirm that *ciscovim run --perform 1* is passing.
5. Confirm that *ciscovim run --perform 3* is passing.

Backup and Restore

Backup and Restore of Management/UM Node

- Backing Up Management Node
- Restoring Management Node
- Management Node Autobackup
- Management Node Migration
- Backing Up UM Node
- Restoring UM Node

Backing Up Management Node

Backing Up Management Node

- Overview
- Management Node Backup
- Backup with Cisco VIM Service Logs
- Encryption of Management Node Backup
- Pods Hosting Central Management VMs

Overview

The management node hosts critical services such as Cisco VIM REST API, Cobbler for PXE, ELK for logging/Kibana dashboard, and VMTP for the cloud validation in Cisco VIM.
As the management node is not redundant, ensure that you take the backup of the management node. If you are facing any issues with the platform, you can restore the management node using the saved management node information.

Management Node Backup

An administrator must maintain the number of backup snapshots on the management node. The backup of the management node is possible only after the complete deployment of at least one Cisco VIM. Two copies of backup, either manual or autobackup is maintained at the management node itself and the older copy is overwritten when a next backup is performed.

Following are some of the activities that cannot be performed during backup:

- Pod management.
- Software update or upgrade.
- Addition, deletion or replacement of nodes.

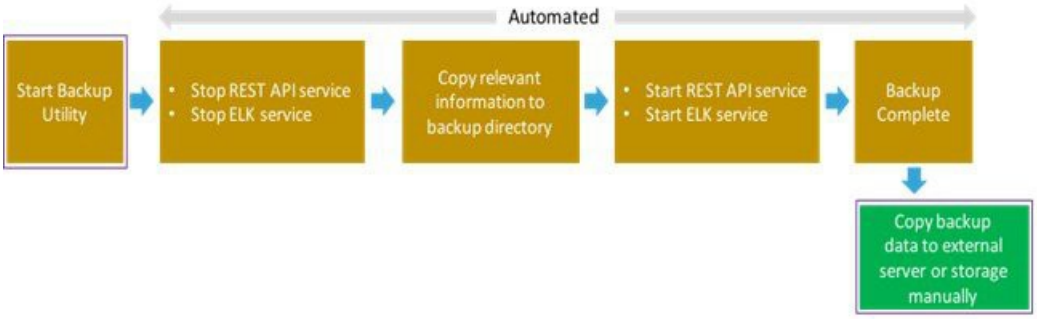
During backup, the REST API service is stopped and the OpenStack logs are cached on the control, compute, and storage nodes until the restoration of the management node is complete.
As part of the backup operation, two files are created:

- .backup_files
- .backup_hash

where *.backup_files* is a list of files that are backed up and the second one is the hash. These two files are also placed at the */var/cisco/ folder* of all the controllers and are used during the restore validation. However, these files are not visible to users and are compressed under *cvim_<RELEASE_TAG>.tar.gz* in the backup directory. Modification of *cvim_<RELEASE_TAG>.tar.gz* and its files is strictly not permitted.

When you attempt to restore from a particular backup, these two files within backup are compared with the files kept in the controllers. If there is any discrepancy, the restore validation fails and you are prompted to either terminate the restore operation or continue despite the validation failure.

Only one copy of the .backup_files and .backup_hash are kept at the controllers. That is, whenever a new backup is created, these two files are overwritten with the most recent ones. Hence the restore validation passes only when the latest backup is used for restore. The following figure illustrates the management node backup operation.



Before you begin

- Save the management node information, for example, IP address of the management node, for use during the restore operation.
- Ensure that you have the br_mgmt and br_api IP addresses and the respective network information.

1. Launch a SSH session with the Cisco NFVI management node.
2. Navigate to *<installer-ws>/tools/mgmt/ directory*.

3. Execute *mgmt_node_backup.py*. The backup operation takes approximately 30 minutes and creates *backup_<tag>_<date-time>* directory in the */var/cisco/ path* for disconnected installation. From Cisco VIM 3.4.3, an optimization to reduce the backup size is available for connected installation. This reduces the overhead needed to copy the backup from the pod during restoration. However, ensure that the management node is connected to the registry during restoration, to download the artifacts for which additional time is required.
4. Copy the directory to a remote server using *rsync* or *scp,* to recover the management node later via restore.
   For example, to copy the backup directory to the remote server *<target_ip>:/var/cisco/directory*, execute the following command sequence:

```
# autobackup triggered as part of pod management operation
rsync -e ssh -go -rtvpX --numeric-ids /var/cisco/autobackup_<version>_<date-time>
root@<remote_ip_address>:/var/cisco/

# manual backup triggered by admin
rsync -e ssh -go -rtvpX --numeric-ids /var/cisco/backup_<version>_<date-time> root@<remote_ip_address>:
/var/cisco/

# scp command can also be used for remote copy (starting CVIM 3.4.3)
# Note that scp command can only be used when copying Cisco VIM or Cisco UM backup directories.
scp -r /var/cisco/autobackup_<version>_<date-time> <username>@<remote_ip_address>:/var/cisco/
```

> ⚠ It is mandatory to copy the backup directory as a whole. Do not perform a bare copy of only the backup directory contents. For instance, *autobackup|backup_<version>_<date-time>* directory must be present in your remote node, after executing the mentioned *rsync* or *scp* command.

> ⚠ On the remote server, protect the backup directory from unauthorized access as the backup files may contain sensitive information. To preserve the file ownership and Linux markings, run as *root* to sync the remote server. The remote server must run RHEL or CentOS 7.x, so that no permission or markings are lost.

5. At the remote server, change the directory to where the backup directory is copied to. In this example, */var/cisco/backup_<version>_<date-time>/.*
6. To verify whether the backup is not corrupted or modified, execute *./check_integrity* and install the packages on the server where *check_integrity* is executed:

```
python-prettytable
python-jinja2
python-babel
python-markupsafe
python-setuptools
pytz
```

Backup with Cisco VIM Service Logs

By default, Cisco VIM service logs are not backed up during autobackup or manual backup. These logs include *REST-API*, *VMTP, SNMP,* and so on. You can take its backup manually, if required.

For manual backup, you can override by appending the *-l* or *--logs* option to the backup command. For optimal performance (reduction of space and time), do not use this option.

```
# cd installer/tools/mgmt
# ./mgmt_node_backup.py --help
Usage: ./mgmt_node_backup.py [options]
Options:

-h, --help  show this help message and exit
-l, --logs  force to also collect Cisco VIM service logs on backup
```

Encryption of Management Node Backup

By default, the management node snapshot contains *setup_data* having sensitive information that is used for restoration during management node failure. This results in an exposure of information about the pod. From Cisco VIM 4.2.0, the management node snapshot is encrypted to protect the leakage of cloud associated information when the snapshot is saved off-premise.

Following are the assumptions for encrypting management node snapshot:

- Since a password is needed to encrypt the management node snapshot, this option is available only as manual option.
- The password used to encrypt the management node, must be used for restoration of the snapshot as well.

- If admin forgets the password, the restoration of the management node from the encrypted snapshot is not possible and mostly leads to reinstallation of the Cisco VIM instance.
- A management node backup must exist, before running the encryption.
- During encryption of the backup, you must enter the password twice.
- Ensure that the encryption and decryption of the management node backup are done in a VNC or KVM console, so that the session does not terminate.

Following are the options to encrypt/decrypt the management node snapshot:

```
# cd installer/tools/
# ./backup_encrypt.py -h
usage: backup_encrypt.py [-h] (-d DECRYPT | -e ENCRYPT)
Script to Encrypt or Decrypt Backup directory/file
optional arguments:
-h, --help show this help message and exit
-d DECRYPT, --decrypt DECRYPT location of the CVIM encrypted backupfile
-e ENCRYPT, --encrypt ENCRYPT location of CVIM backup directory


Example Snippet of Encryption of Backup dir of management node:
[root@razor tools]# ./backup_encrypt.py -e /var/cisco/<path>/autobackup_4.1.0_2021-02-26_19-10-56/

DO NOT press ^c while the Encryption is in progress
Aborting Encrypt operation with a ^c may corrupt the encrypted backup file

*** You Should remember the password to decrypt this backup, otherwise you cannot decrypt this backupfile ***

Password:
Password Again:
Initiating Encryption. Please wait...
Encrypted backup file autobackup_4.1.0_2021-02-26_19-10-56_3650878860.tar.gpg is available in /var/cisco/<path>
```


The encrypted backup file is in gpg format.

To ensure that the encrypted snapshot is not corrupted, during its copy to and from the management node, use the *cksum* command against the *gpg* file and compare the values at the source and destination. Listed below is an example of the same:

```
# cksum autobackup_4.1.0_2021-02-26_19-10-56_3650878860.tar.gpg
3650878860 9518011 autobackup_4.1.0_2021-02-26_19-10-56_3650878860.tar.gpg
```

Pods Hosting Central Management VMs

You need to do a manual backup of QCOW2 images used for hosting the central management VMs, as the automated backup process does not take care of those target images. You can find the location of these images under IMAGES section of /root/openstack-configs/*setup_data.CentralMgmt.yaml*.

Restoring Management Node

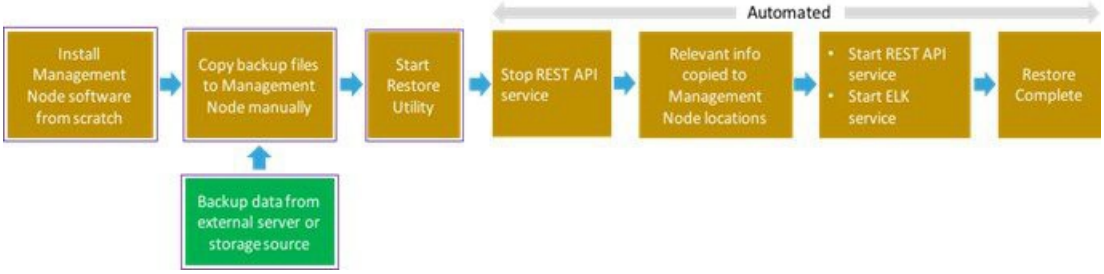
Restoring Management Node

You have to reimage the management node with the running Cisco VIM release ISO version when the backup is performed, before initiating the restore operation. The restoration fails, in case of a version mismatch.

> ⚠ Version checking is available only for offline installation.

During restore operation, the system checks for the management node's IP address information to match the prior configuration. The logs are cached on the control, compute, and storage nodes from the moment of the management node failure until its restoration.

The following figure illustrates Cisco VIM management node restore operation.



> ⓘ Ensure that you have the *hostname, br_mgmt* and *br_api IP* addresses of the failed management node.

> ⚠ Ensure that you execute the following restoration procedure only with *root* user.

1. Reimage the management node with the ISO version with which you want to restore the node, and with the same hostname and IP address that is used before the node failure. For more information on the management node, see Management Node on UCS C-series (M4/M5) or Management Node on Quanta Servers based on the type of management node that is used.
2. Navigate to */var/cisco/directory* at the remote server where the backup folder is copied during the backup operation.
3. Copy the backup file (encrypted or not) to the */var/cisco/directory* of the reimaged management node.
   For example, to copy the backup folder from the remote host *<remote_ip>* to the management node */var/cisco/directory*, execute the following command from the management node:

   ```
   scp <current_backupin_gpg_format> root@<remote_ip>:/var/cisco/<current_backupin_gpg_format>
   ```

   or

   ```
   rsync -e ssh -go -rtvpX --numeric-ids root@<remote_ip>:/var/cisco/backup_2017-01-09_14-04-38 /var/cisco/

   Or
   ## on the management node, create /var/cisco/ directory, if it does not exist
   # mkdir /var/cisco/
   # scp command can also be used (starting Cisco VIM 3.4.3)
   # scp -r <remote_username>@<remote_ip>:/var/cisco/backup_2017-01-09_14-04-38 /var/cisco/
   ```

   > ⚠ It is mandatory to copy the backup directory as a whole. Do not perform a bare copy of only the backup directory contents. For instance, *autobackup|backup_<version>_<date-time>* directory must be present in your management node after executing the mentioned *rsync* or *scp* command.

   > ⚠ Ensure that you preserve the backup directory at the remote node, until the entire process of restore is complete.

4. If step file is in gpg format, execute *cksum*command against the gpg file and compare the values at the source and destination. Listed below is an example of the same:

```
# cksum autobackup_4.1.0_2021-02-26_19-10-56_3650878860.tar.gpg
3650878860 9518011 autobackup_4.1.0_2021-02-26_19-10-56_3650878860.tar.gpg
```

5. If the backup of the management node is encrypted and is in gpg format, execute the below commands, otherwise skip this step:

```
# cd /root/installer-<tag>/tools/
# ./backup_encrypt.py -d /var/cisco/<directory_path>/autobackup_4.1.0_2021-02-26_19-10-56_3650878860.tar.
gpg
```

6. Navigate to the backup folder and execute the following command, to verify if the backup is not corrupted or modified:

```
# cd /var/cisco/backup_<date-time>
# ./check_integrity
```

7. To restore a baremetal management node that is Layer 2 adjacent to the pod or a management node VM that is Layer 3 adjacent to the pod, execute the following command in */var/cisco/backup_<date-time>* folder. In both conditions, the API and management network does not change before and after the restoration.

```
/var/cisco/backup_<date-time> # ./restore
```

8. To migrate the management node that is on Layer 2 or Layer 3 adjacent to the pod, to the one in a VM that is on Layer 3 adjacent to the pod, execute the following command in */var/cisco/backup_<date-time>* folder:

```
/var/cisco/backup_<date-time> # ./restore --remote
```


The restore operation takes around 45 to 60 minutes. As the central management of VMs do not support NFVBench, the corresponding information gets deleted from the *setup_data.yaml* in the central management VM.

In both conditions, the API and management network changes before and after the restoration. However, the hostname of the management node cannot be changed.

Before restoration, the restore script validates the backup folder.

.

9. If the validation fails, the restoration is halted and an error message is displayed. The script also verifies the last performed backup folder in the management node. If any defects are detected, you must confirm to proceed with restore operation.

```
....
2017-02-02 21:25:23 INFO Starting Cisco VIM restore...
2017-02-02 21:25:23 INFO Cisco VIM restore: estimated run time is approx. 45 mins...
2017-02-02 21:25:23 INFO Please see progress log for restore at
/var/log/mercury/installer/restore_2017-02-02_21:25:23.log
2017-02-02 21:25:27 ERROR Error: Backup id is not the one expected
Error: Found hashID file only in controller(s): controller-2, controller-3
Management backup files are ok (as per controller-2)
Management backup files are ok (as per controller-3)
The management node changed after the last backup was stored. Do you still want to proceed restoring
this management node? [Y/n] y
2017-02-02 22:17:55 INFO Workspace restored to /root/installer-6518
2017-02-02 22:17:55 INFO Cisco VIM restore: Executing restore playbook ...
2017-02-02 22:18:47 INFO Cisco VIM restore: Executing bootstrap playbook ...
```

> ⚠ The default behavior is to continue by keying **Return** or **Y**. Keying **N** terminates the restore operation.

```
...
2017-02-02 21:25:23 INFO Starting Cisco VIM restore...
2017-02-02 21:25:23 INFO Cisco VIM restore: estimated run time is approx. 45 mins...
2017-02-02 21:25:23 INFO Please see progress log for restore at
/var/log/mercury/installer/restore_2017-02-02_21:25:23.log
2017-02-02 21:25:27 ERROR Error: Backup id is not the one expected
Error: Found hashID file only in controller(s): controller-2, controller-3
Management backup files are ok (as per controller-2)
Management backup files are ok (as per controller-3)
The management node changed after the last backup was stored. Do you still want to proceed restoring
this management node? [Y/n] n
Aborting the restore operation as per user request
```

10. Once the restoration is done, several health check points are automatically executed and the summary of results for that particular cloud availability is displayed:
11. Run the following checks manually to verify the restore status:

    a. Check the status of the REST API server:

    ```
    # 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>/
    ```

    b. Check the *setup_data* and runtime consistency of the management node:

    ```
    # cd installer-<tagid>/; ciscovim run --perform 1,3 -y
    ```

    c. Execute the cloud sanity using *ciscovim* command:

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

    d. To view the results of cloud sanity, use the command:

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

    > ⚠ If the pod is hosting VMs for central management, after the management node restoration, copy the QCOW2 images to directory location as defined in the IMAGES section of */root/openstack-configs/setup_data.CentralMgmt.yaml.*

Management Node Autobackup

Management Node Autobackup

Cisco VIM supports the backup and recovery of the management node. By default, the feature is enabled. Auto-snapshots of the management node happens during pod management operation. You can disable the autobackup of the management node. After the successful completion of certain pod management operations, a backup of the management node is performed automatically. Only two copies of the backup, either manual or automatic is kept at */var/cisco/* at any given time.

The directory format is autobackup_<tag>_<timestamp>.

Following are the list of operations:

- Fresh installation of Cisco VIM.
- Commit an update.
- Replace controller.
- Add or remove compute nodes.
- Add or remove the storage node.
- Reconfiguration.
- CVIM-MON.

1. To enable or disable the management node, update the *setup_data.yaml* file as follows:

```
AutoBackup Configuration

# Default is True
#autobackup: <True or False>
```

2. Take a backup of setupdata file and update it manually with the configuration details by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# # update the setup_data to change autobackup
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --setupfile /root/MyDir/setup_data.yaml
```

The following table shows the scenarios of when autobackup is performed:

Pod operation	Autobackup performed
Update	No
Rollback	No
Commit	Yes
Update fail with auto rollback	No

After the successful creation of the autobackup folder, you can copy it to an external server for later restoration as mentioned in Restoring Management Node.

Management Node Migration

Management Node Migration: Baremetal to VM

With the advent of management node in VMs, Cisco VIM provides a second level of optimization for migrating the management node from baremetal (that is Layer 2 adjacent to the Cisco VIM pod) to VMs (that is Layer 3 adjacent to the Cisco VIM pod).

To get the migration, follow the below steps:

1. Use the standard steps to backup the management node preferably without the logs. For details, see Backing Up Management Node
2. SSH into the baremetal management node, and shut down all the services running on the management node only via the following:

```
# cd installer-XXXXX
# ./tools/mgmt/mgmt_node_teardown.sh
```

3. Launch the central management VM in the Cisco VIM cloud. For more details, see Centralizing Management Node
4. Restore the snapshot onto the management node VM. For more details, see Restoring Management Node
5. Continue using the VM based management node for any new pod management operation.

Backing Up UM Node

Backing Up Unified Management (UM) Node

- Overview
- Backing up Cisco VIM UM
  - Autobackup UM
  - Manual Backup
  - Preserving Backup Directory on Remote Node

Overview

The UM node hosts critical services such as certificates, mariadb, and so on for the UI solution of Cisco VIM. As the UM node is not redundant, ensure that you take the backup of the same. This way if the UM node goes through a non-recoverable failure, you can restore the UM node using the saved UM node information.

Backing up Cisco VIM UM

The administrator maintains the backup for UM on the management node. The UM backup is done only after the complete deployment of the UM bootstrap. Only two copies of the backup directory are maintained at the management node. The older copy is overwritten, when a next UM backup or autobackup takes place.

UM backup is stored at the default backup location */var/cisco/insight_backup/* with backup dir in the *insight_autobackup_<release_tag>_<date>_<time>* for mat.

UM UI triggers an autobackup whenever it detects any operation related to MySQL database entry to preserve the latest state of UM.

> ⚠ UM backup is not allowed after an update. The update is an intermediate stage between rollback and commit. Any change relevant to MySQL database entry is not backed up, after an update from UM UI.

Autobackup UM

If there is a change, UM Installation automatically runs a daemon process to take the autobackup.
Get the live status of the process by checking the log located at */var/log/insight/insight_autobackup/insight_autobackup.logs* or *systemctl status insight-autobackup*.

> ⚠ A maximum of 10 log files of size 1024*1024 is maintained in the directory.

Following are the scenarios where autobackup is initiated:

Unified Management Operation	Auto-backup Performed
Adding or deleting pod	Yes
Changing pod REST password and certificate	Yes
Add/Edit/Delete all types of users	Yes
Add/Edit/Delete roles	Yes
Modify user and role association	Yes
Revoking or adding user permission	Yes
Log in or Logout	No
Context switching	No
Change user password	Yes

1. To check the status of UM, perform the following steps:

```
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 Wed 2017-08-30 01:17:18 PDT; 19s ago
Main PID: 19949 (python)
Memory: 12.4M
CGroup: /system.slice/insight-autobackup.service
19949 /usr/bin/python /root/<installer-tag>/insight/playbooks/../insight_autobackup.py
```

2. To stop UM autobackup, do the following:

```
systemctl stop insight-autobackup

insight-autobackup.service - Insight Autobackup Service
Loaded: loaded (/usr/lib/systemd/system/insight-autobackup.service; enabled; vendor preset:
disabled)
Active: inactive (dead) since Mon 2017-09-04 00:43:43 PDT; 5s ago
Process: 19993 ExecStop=/bin/kill ${MAINPID} (code=exited, status=0/SUCCESS)
Main PID: 19984
Memory: 56.0K
CGroup: /system.slice/insight-autobackup.service
```

3. To start UM autobackup, use the following command:

```
systemctl start 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 Wed 2017-08-30 01:17:18 PDT; 19s ago
Main PID: 19949 (python)
Memory: 12.4M
CGroup: /system.slice/insight-autobackup.service
19949 /usr/bin/python /root/<installer-tag>/insight/playbooks/../insight_autobackup.py
```

4. UM autobackup performs the following operations.
    a. Installation
        - Invokes the script when the Galera DB and UM containers are up.
        - Stores the log in */var/log/insight/insight_autobackup_logs/insight_autobackup.log.*
        - Notifies if the service is up or not with 10-seconds pulse.

        ```
        [2017-09-04 00:49:01,504] INFO [Insight Autobackup] Insight Autobackup Service Running.
        [2017-09-04 00:49:11,514] INFO [Insight Autobackup] Insight Autobackup Service Running.
        [2017-09-04 00:49:21,525] INFO [Insight Autobackup] Insight Autobackup Service Running.
        ```

        - If there is any change, it takes a backup (time to check if SQL difference is 30 seconds).
        - Creates *insight_db.tar.gz* and *ui_snapshot.tar.gz* and dumps them in the latest backup directory.
    b. Update
        - During the update, *insight_runner* does not trigger autobackup.
        - Terminates autobackup service and does not maintain a backup in the intermediate state.
    c. Rollback: Re-invokes the script from the previous workspace.
    d. Commit: Invokes the script again from the new workspace.
    e. Uninstall: Deletes the service files, but retains the log directory.

Manual Backup

Launch an SSH session with Cisco UM node and execute the following commands:

```
# cd Insight-<tag_id>/insight/backup_restore/
#./um_node_backup.py -h

Usage:  ./um_node_backup.py [options]
Options:
    -h, --help   show this help message and exit


# ./um_node_backup.py

2020-05-29 03:14:42 INFO     Backup Validation Passed
2020-05-29 03:14:42 INFO     Starting Cisco UM backup ...
2020-05-29 03:14:42 INFO     Cisco UM backup: estimated run time is approx. 30 mins...
2020-05-29 03:14:42 INFO     Please see backup progress log at /var/log/insight_backup
/insight_backup_<version>_<date>_<time>.log

2020-05-29 03:14:42 INFO     Cisco UM prebackup: executing prepbackup playbook ...
2020-05-29 03:14:45 INFO     Cisco UM backup: executing backup playbook ...
2020-05-29 03:15:00 INFO     Cisco UM states backed up /var/cisco/insight_backup
/insight_backup_<version>_<date>_<time>
2020-05-29 03:15:00 INFO     Executing Cisco UM postbackup .../var/cisco/insight_backup
/insight_backup_<version>_<date>_<time>
2020-05-29 03:15:00 INFO     Insight DB backed up ...
2020-05-29 03:15:06 INFO     Cisco UM backup completed: backupdir at /var/cisco/insight_backup
/insight_backup_<version>_<date>_<time>
```

From Cisco VIM 3.4.6, an optimization to reduce the backup size is available for connected installation. This reduces the need to copy the backup from the pod during restoration. However, ensure that the management node is connected to the registry during restoration, to download the artifacts for which additional time is required.

Preserving Backup Directory on Remote Node

1. Copy the directory to a remote server using *rsync* or *scp,* to recover the management node later via restore.

   For example, to copy the backup directory to the remote server *<target_ip>:/var/cisco/insight_backup/directory*, execute the following commands:

   ```
   # autobackup triggered as part of UM management operation
   rsync -e ssh -go -rtvpX --numeric-ids /var/cisco/insight_backup/insight_backup_<version>_<date-time>
   root@<remote_ip_address>:/var/cisco/insight_backup/

   # manual backup triggered by admin
   rsync -e ssh -go -rtvpX --numeric-ids /var/cisco/insight_backup/insight_backup_<version>_<date-time>
   root@<remote_ip_address>:/var/cisco/insight_backup/

   # scp command can also be used for remote copy (starting CVIM 3.4.6)
   scp -r /var/cisco/insight_backup/insight_backup_<version>_<date-time> <username>@<remote_ip_address>:/var
   /cisco/insight_backup/
   ```

   ⚠ It is mandatory to copy the backup directory as a whole. Do not perform a bare copy of only the backup directory contents. For instance, *insight_autobackup|backup_<version>_<date-time>* directory must be present in your remote node, after executing the mentioned *rsync* or *scp* command.

   ⚠ On the remote server, protect the backup directory from unauthorized access as the backup files may contain sensitive information. To preserve the file ownership and Linux markings, run as *root* to sync the remote server. The remote server must run RHEL or CentOS 7. x, so that no permission or markings are lost.

2. At the remote server, change the directory to where the backup directory is copied to. In this example, it is */var/cisco/insight_backup /insight_backup_<version>_<date-time>/.*
3. To verify whether the backup is not corrupted or modified, execute *./check_integrity* and install the packages on the server where *check_integrity* is executed:

```
python-prettytable
python-jinja2
python-babel
python-markupsafe
python-setuptools
pytz
```

Restoring UM Node

Restoring Unified Management (UM) Node

You can restore the Cisco VIM UM to its previous running state that existed at the time of backup.

> ⚠ Ensure that you do not run the UM on the node on which restore operation is performed.

1. Reimage the UM node with the ISO version with which you want to restore the node, and with the same IP address that is used before the failure of the node.
2. Navigate to */var/cisco/insight_backup/* directory at the remote server where the backup directory is copied during backup. Execute *./check_integrity* , to verify if the backup is not corrupted or modified.
3. Copy the backup directory to the */var/cisco/insight_backup/* directory of the re-imaged UM node. For example, to copy the backup directory from the remote host <remote_ip> to the management node */var/cisco/insight_backup/*directory, execute the following command sequence:

```
rsync -e ssh -go -rtvpX --numeric-ids <remote_username>@<remote_ip>:/var/cisco/insight_backup
/insight_backup_2020-01-09_14-04-38 /var/cisco/insight_backup

Or
# On the UM node, create /var/cisco/ directory, if it does not exist
# mkdir /var/cisco/
# scp command can also be used (starting Cisco VIM 3.4.6)
# scp -r <remote_username>@<remote_ip>:/var/cisco/insight_backup/insight_backup_2020-01-09_14-04-38 /var
/cisco/insight_backup
```

> ⚠ It is mandatory to copy the backup directory as a whole. Do not perform a bare copy of only the backup directory contents. For instance, i*nsight_autobackup|backup_<version>_<date-time>* directory must be present in your management node, after executing the mentioned *rsync* or *scp* command.

> ⚠ It is highly recommended to use a KVM session instead of SSH, for restoring the UM node.
> Ensure that you preserve the backup directory at the remote node, until restoration is complete.

4. Navigate to the backup folder and execute the following command to verify if the backup is not corrupted or modified:

```
# cd /var/cisco/insight_backup/insight_backup_<date-time>
# ./check_integrity
```

5. In */var/cisco/insight_backup/insight_backup_<date-time>* directory, execute the following command:

```
# ./restore -h

Usage of um_node_restore script
Options:
 -h --help
     Show this help text
 --verbose
     Set the logging level to debug


#./restore

Running System validation. May take time....
Restore Checksum Validation Passed
Restore Validation Passed
2020-05-29 05:04:15 INFO     Starting Cisco UM restore...
2020-05-29 05:04:15 INFO     Cisco UM restore: estimated run time is approx. 45 mins...
2020-05-29 05:04:15 INFO     Please see progress log for restore at /var/log/insight_restore
/insight_restore_2020-05-29_05:04:15.log
2020-05-29 05:04:15 INFO     Workspace restored to /root/insight-3.4.6
2020-05-29 05:04:15 INFO     Cisco UM restore: Executing restore playbook ...
2020-05-29 05:04:19 INFO     Executing UM node validation and orchestration step....
2020-05-29 05:14:25 INFO     Verifying UM install status..

Fetching Insight install status..


 Cisco VIM Insight Install Status!
+--------------------+--------+-----------------------------------------------+
| Description        | Status | Details                                       |
+--------------------+--------+-----------------------------------------------+
| Insight Setup      | PASS   | Success                                       |
| Insight Version    | PASS   | NG: 4.0.0                                     |
| Insight UI URL     | PASS   | https://<br_api>:9000                         |
| Mariadb Container  | PASS   | mariadb_26349                                 |
| Insight Container  | PASS   | insight_26349                                 |
| Insight Autobackup | PASS   | [ACTIVE]: Running 'insight-autobackup.service' |
+--------------------+--------+-----------------------------------------------+

[WARNING]: UM Management node needs to be rebooted for following to take effect:
         * Updated core libraries and services.


2020-05-29 05:14:25 INFO     Completed Cisco UM Restore successfully ...
```

Managing Cisco VIM Software Hub

Managing Cisco VIM Software Hub

- Updating Cisco VIM Software Hub TLS Certificate and Registry Credentials
- Cisco VIM Software Hub Server Backup and Restore
- Checking Integrity of Autobackup Files
- Restoring Cisco VIM Software Hub from Backup
- Resolving Low Disk Space
- Manually Updating Packages
- Upgrading Cisco VIM Software Hub from Cisco VIM 3.x to Cisco VIM 4.0.0

Cisco VIM Software Hub helps mitigate the need to ship USBs across different pods during Cisco VIM installation or update. To ensure the long-term viability of Cisco VIM Software Hub, it is designed to handle the following Day 2 scenarios:

Updating Cisco VIM Software Hub TLS Certificate and Registry Credentials

Before installing the release artifacts from the Cisco VIM Software Hub server, you must provide a valid TLS certificate and Cisco VIM Software Hub registry credentials in the sds_setup_data.yaml file. Taking into account the security policies of an organization, Cisco VIM Software Hub allows you to update the TLS certificate and registry credentials on the Cisco VIM Software Hub server as required.

1. Navigate to the last installed release workspace using the *ls –lrt* command.
2. Replace the TLS certificate in the *openstack-configs* directory.
3. Modify the credentials in the *sds_setup_data.yaml* file.
4. Run the following command for the changes to take effect:

   ```
   # cd /root/cvim_sds-<last-tag> # directory of last installed release and execute the following command.
   # ./sds_runner/runner.py
   ```

   This operation validates the changes in the *sds_setup_data.yaml* file and new TLS certificate. It reconfigures the Cisco VIM Software Hub server components with this new information.

   The Cisco VIM Software Hub registry credentials of the pods that rely on Cisco VIM Software Hub are also reconfigured.

Cisco VIM Software Hub Server Backup and Restore

Cisco VIM Software Hub triggers an autobackup operation when a new Cisco VIM release is installed on the Cisco VIM Software Hub server. It takes a backup of the relevant files from the Cisco VIM Software Hub server, and saves it in the following location on the Cisco VIM Software Hub server:

```
directory /var/cisco/autobackup_<tag>_<date-time>
```

For manual backup of Cisco VIM Software Hub, use the following commands:

```
# cd cvim_sds-<tag-id>/sds/bkp_rest/
# ./sds_backup.py
```

Cisco VIM Software Hub maintains only the latest two backup directories. The older copy is overwritten when the next autobackup operation is triggered. If you want to use an older backup directory for a restore operation later, you need to save it to another location before it is overwritten. You can use the *rsync* or *scp* commands to save it to an RHEL7/CentOS based system which is external to the Cisco VIM Software Hub server.

Checking Integrity of Autobackup Files

You can use the script provided in the autobackup directory to check the integrity of the autobackup files after using the *rsync* or *scp* commands.

Before you begin

Ensure that the following packages are installed on the backup server using yum:

- python-prettytable
- python-jinja2
- python-babel
- python-markupsafe
- python-setuptools
- pytz

1. Navigate to the autobackup directory.
2. Execute the following command to run the script:

   ```
   # ./check_integrity
   ```

Restoring Cisco VIM Software Hub from Backup

A Cisco VIM Software Hub restore operation is done, when the original Cisco VIM Software Hub server is being replaced by a new one.

1. Reimage the Cisco VIM Software Hub server with the ISO version with which you want to restore the node, and with the same IP address that is used before the failure of the node.
2. Navigate to the location where the backup directory is copied during the backup operation.
3. Verify the integrity of the backup files as described in Checking Integrity of Autobackup Files
4. Copy the backup file to the directory of the reimaged Cisco VIM Software Hub node.

   For example, you can copy the backup directory from the remote host 20.0.0.5 to the Cisco VIM Software Hub node directory */var/cisco/* as follows:

   ```
   rsync -e ssh -go -rtvpX --numeric-ids root@20.0.0.5:/var/cisco/autobackup_2017-01-09_14-04-38
    /var/cisco/
   ```
5. Navigate to the backup directory and execute the following command to verify if the backup is not corrupted or modified.
6. In the */var/cisco/autobackup_<tag>_<date-time>* directory, execute the following commands:

   ```
   # cd /var/cisco/backup_<date-time>
   # ./restore
   ```

   It may take about 45 minutes for the restore operation to complete.

   Before restoring a backup directory, the restore script validates the backup directory. If the validation fails, the restore operation is interrupted and an error message is displayed. The restore script also verifies the latest backup directory in the Cisco VIM Software Hub Node. If defects are detected, you need to confirm whether you want to proceed with the restore operation.

   For example:

   ```
   2017-02-02 21:25:23 INFO Starting Cisco VIM restore...
   2017-02-02 21:25:23 INFO Cisco VIM restore: estimated run time is approx. 45 mins...
   2017-02-02 21:25:23 INFO Please see progress log for restore at
   /var/log/mercury/installer/restore_2017-02-02_21:25:23.log
   2017-02-02 21:25:27 ERROR Error: Backup id is not the one expected
   Error: Found hashID file only in controller(s): controller-2, controller-3 Management backup
   files are ok (as per controller-2)
   Management backup files are ok (as per controller-3)
   The management node changed after the last backup was stored. Do you still want to proceed restoring
   this management node? [Y/n] y
   2017-02-02 22:17:55 INFO Workspace restored to /root/installer-6518
   2017-02-02 22:17:55 INFO Cisco VIM restore: Executing restore playbook ...
   2017-02-02 22:18:47 INFO Cisco VIM restore: Executing bootstrap playbook ...
   ```

   To continue the restore operation, you can press the *Enter* key or *Y* key. If you want to abort the restore operation, you need to press the *N* key.

   ```
   2017-02-02 21:25:23 INFO Starting Cisco VIM restore...
   2017-02-02 21:25:23 INFO Cisco VIM restore: estimated run time is approx. 45 mins...
   2017-02-02 21:25:23 INFO Please see progress log for restore at
   /var/log/mercury/installer/restore_2017-02-02_21:25:23.log
   2017-02-02 21:25:27 ERROR Error: Backup id is not the one expected
   Error: Found hashID file only in controller(s): controller-2, controller-3 Management backup
   files are ok (as per controller-2)
   Management backup files are ok (as per controller-3)
   The management node changed after the last backup was stored. Do you still want to proceed restoring
   this management node? [
   Y/n] n
   Aborting the restore operation as per user request
   ```

Resolving Low Disk Space

Installing releases on Cisco VIM Software Hub server is not allowed, if the free disk space is less than 20%. Hence, use a utility to remove docker images from the container registry running on the Cisco VIM Software Hub server. You can find the cleanup script at the following location:

```
/root/cvim_sds-<last-tag>/sds/registry_cleanup.py
```

Example of running the cleanup script:

```
# ./registry_cleanup.py -h
usage: registry_cleanup.py [-h] (--list | --delete DELETE | --unused_tags)
[-u USERNAME] [-p PASSWORD] [-r REGISTRY]
List/Delete image tags in the registry
optional arguments:
    -h, --help Show this help message and exit
    --list List Image Tags in Registry
    --delete DELETE Delete Images of provided tags from registry
    --unused_tags List unused Tags in SDS registry
    -u USERNAME, --username USERNAME
    Registry Username
    -p PASSWORD, --password PASSWORD
    Registry Password
    -r REGISTRY, --registry REGISTRY
    Registry URL
```

The cleanup script requires three mandatory parameters namely Registry URL, Registry username, and Registry password. The script supports the following three options:

- List Image Tags: The option lists all the images and corresponding tags present in the docker registry.
- Unused Tags: This option lists all the releases present on the Cisco VIM Software Hub server but not used by any Cisco VIM pod. By default, the pods are registered with the Cisco VIM Software Hub server. When a pod is installed, updated, roll backed, or upgraded, the release information is sent to Cisco VIM Software Hub. You can use this command to identify the releases that can be safely removed from the Cisco VIM Software Hub server.
- Delete Tags: You can specify the releases that you want to remove from the docker registry. The script removes these images and frees the disk space.

A sample snippet of the command template is listed below:

```
#./registry_cleanup.py -u <username> -p <password> -r https://<sds_domian_name>/ --list
#./registry_cleanup.py -u <username> -p <password> -r https://<sds_domian_name>/ --delete
3.2.0
```

Manually Updating Packages

Cisco VIM Software Hub installs the repositories within docker containers, so that all the packages to be installed are obtained from those repositories. These repositories are updated when you install a later version of Cisco VIM release on the Cisco VIM Software Hub server. Once the repositories are updated, all the packages except httpd package and its dependencies are updated. When httpd is updated, all downstream connections are disrupted and the Cisco VIM pod installation must be restarted. Hence, updating httpd is deferred.

To update httpd and its dependent packages, you can use the update script found in the tools directory. Ensure that you run this script during the maintenance phase so that none of the Cisco VIM pods are currently attempting to get artifacts from the Cisco VIM Software Hub server.
Run the following command to execute the update script:

```
# cd /root/cvim_sds-<last-tag> # directory of last installed release and execute the following command.
# ./update_httpd.sh
```

Upgrading Cisco VIM Software Hub from Cisco VIM 3.x to Cisco VIM 4.0.0

As Cisco VIM Software Hub instance running 4.0.0 cannot download 3.x or 2.x artifacts, careful planning is essential before upgrading Cisco VIM Software Hub to 4.0.0. However, the artifacts prepopulated on Cisco VIM 3.x Software Hub can be migrated to Cisco VIM 4.x Software Hub.

To upgrade the SDS node from Cisco VIM 3.x lineup to Cisco VIM 4.x, and keep the Cisco VIM 3.x docker images and Cisco VIM artifacts restored on Cisco VIM 4.x node, follow the below steps:

1. Take the backup of Cisco VIM 3.x lineup docker images from Cisco VIM 3.x Software Hub node:

   ```
   [root@sds1-mgmt ~]# du -sh /var/lib/docker/volumes/container_registry
   28G /var/lib/docker/volumes/container_registry

   [root@sds1-mgmt ~]# du -sh /var/www/html/mercury-releases/
   2.6G    /var/www/html/mercury-releases/

   [root@backup-node ~]# mkdir -p /var/container_registry
   [root@sds1-mgmt ~]# rsync --ignore-existing -raz --progress /var/lib/docker/volumes/container_registry/
   root@backup-node:/var/container_registry/

   [root@backup-node ~]# mkdir -p /var/mercury-releases
   [root@sds1-mgmt ~]# rsync --ignore-existing -raz --progress /var/www/html/mercury-releases/ root@backup-
   node:/var/mercury-releases/
   ```

   Ensure that you have enough space on remote server to copy the artifacts from software hub node.

2. Reimage the Software Hub node with Cisco VIM 4.x lineup.
3. Copy the artifacts from backup node to Cisco VIM 4.x Software Hub node:

```
[root@sds1-mgmt ~]# rsync --ignore-existing -raz --progress root@backup-node:/var/container_registry/ /var
/lib/docker/volumes/container_registry/
[root@sds1-mgmt ~]# rsync --ignore-existing -raz --progress root@backup-node:/var/mercury-releases/ /var
/www/html/mercury-releases/
```

4. Verify whether restoration is done successfully by downloading Cisco VIM 3.x and Cisco VIM 4.x images from Docker registry:

```
[root@sds1-mgmt ~]# docker login <SDS_REGISTRY_NAME>
# Enter SDS_REGISTRY_USERNAME and SDS_REGISTRY_PASSWORD for authentication

[root@sds1-mgmt ~]# docker pull <private_registry>/<3.x repository>
```

Troubleshooting

Troubleshooting

- Cisco NFVI Node
- Pre-checks for Storage Removal
- General Troubleshooting Procedures
- Connection/Installation Problems
- Management Node Recovery Scenarios
- Compute Node Recovery Scenario
- Technical Support Tools
- Disk and OSD Maintenance Tools
- Utility Tool
- Cisco VIM Client Debug Option

Cisco NFVI Node

Cisco NFVI Node

- Displaying Cisco NFVI Node Names and IP Addresses
- Verifying Cisco NFVI Node Interface Configurations
- Displaying Cisco NFVI Node Network Configuration Files
- Viewing Cisco NFVI Node Interface Bond Configuration Files
- Viewing Cisco NFVI Node Route Information
- Viewing Linux Network Namespace Route Information

Displaying Cisco NFVI Node Names and IP Addresses

Complete the following steps to display the Cisco NFVI node names and IP addresses.

1. Log into the Cisco NFVI build node.
2. View the *openstack-configs/mercury_servers_info* file to see the node name and address:

```
# more openstack-configs/mercury_servers_info Total nodes: 5
Controller nodes: 3
+------------------+--------------+--------------+--------------+-----------------+---------+
| Server           | CIMC                   | Management          | Provision         |
Tenant                   | Storage |
+------------------+--------------+--------------+--------------+-----------------+---------+
| test-c-control-1 | 10.10.223.13 | 10.11.223.22 | 10.11.223.22 | 169.254.133.102 | None         |
|                                    |                            |              |
|                |         |
| test-c-control-3 | 10.10.223.9  | 10.11.223.23 | 10.11.223.23 | 169.254.133.103 | None         |
|                                    |              |
|                                |                                    |                 |
| test-c-control-2 | 10.10.223.10 | 10.11.223.24 | 10.11.223.24 | 169.254.133.104 | None         |
|                                    |                            |
|                                  |                                      |                   |
+------------------+--------------+--------------+--------------+-----------------+---------+
Compute nodes: 2
+------------------+--------------+--------------+--------------+-----------------+---------+
| Server                    | CIMC                   | Management          | Provision         |
Tenant                   | Storage |
+------------------+--------------+--------------+--------------+-----------------+---------+
| test-c-compute-1 | 10.10.223.11 | 10.11.223.25 | 10.11.223.25 | 169.254.133.105 | None         |
|                                    |                            |
|                                  |                                      |                 |
| test-c-compute-2 | 10.10.223.12 | 10.11.223.26 | 10.11.223.26 | 169.254.133.106 | None         |
|                                    |                            |
|                                  |                                      |                      |
+
```

> During Cisco NFVI deployment, SSH public keys for each node are added to *.../.ssh/authorized_keys*. Hence, from the build node you can log into each of the Cisco NFVI nodes without passwords. For some reason, if you need account information, see the *openstack-configs/secrets.yaml* file on the build node.

Verifying Cisco NFVI Node Interface Configurations

Complete the following steps to verify the interface configuration of Cisco NFVI nodes:

1. SSH into the target node, for example, one of the Cisco VIM controllers:

```
[root@mgmt-node~]# ssh root@control-server-1
[root@control-server-1 ~]#
```

2. Enter the *ip a* command to get a list of all interfaces on the node:

```
[root@control-server-1 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN link/loopback 00:00:00:00:00:00
brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
2: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000 link/ether
54:a2:74:7d:42:1d brd ff:ff:ff:ff:ff:ff
3: enp9s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000 link/ether
54:a2:74:7d:42:1e brd ff:ff:ff:ff:ff:ff
.
.
.
```

Displaying Cisco NFVI Node Network Configuration Files

Complete the following steps to view the Cisco NFVI node network configuration files:

1. SSH into the target node, for example, one of the Cisco VIM controllers:

```
[root@mgmt-node~]# ssh root@control-server-1
[root@control-server-1 ~]#
```

2. List all the network configuration files in the */etc/sysconfig/network-scripts* directory, for example:

```
[root@control-server-1 ~]# ls /etc/sysconfig/network-scripts/
ifcfg-a                 ifcfg-enp15s0                ifcfg-mx0                ifdown-
ib                ifup                         ifup-ppp
ifcfg-a0                ifcfg-enp16s0                ifcfg-mx1                ifdown-ippp
ifup-aliases         ifup-routes
ifcfg-a1                ifcfg-enp17s0                ifcfg-p                ifdown-ipv6         ifup-
bnep                ifup-sit
ifcfg-br_api         ifcfg-enp18s0                ifcfg-p0                ifdown-isdn         ifup-
eth                ifup-Team
ifcfg-br_mgmt         ifcfg-enp19s0                ifcfg-p1                ifdown-post         ifup-
ib                ifup-TeamPort
ifcfg-e                 ifcfg-enp20s0                ifcfg-t                ifdown-ppp
ifup-ippp                ifup-tunnel
ifcfg-e0                ifcfg-enp21s0                ifcfg-t0                ifdown-routes
ifup-ipv6                ifup-wireless
ifcfg-e1                ifcfg-enp8s0                ifcfg-t1                ifdown-
sit                ifup-isdn                init.ipv6-global
ifcfg-enp12s0         ifcfg-enp9s0                ifdown                ifdown-Team
ifup-plip                network-functions
ifcfg-enp13s0         ifcfg-lo                ifdown-bnep         ifdown-TeamPort ifup-
plusb                network-functions-ipv6
ifcfg-enp14s0         ifcfg-mx                ifdown-eth                ifdown-tunnel
ifup-post
```

Viewing Cisco NFVI Node Interface Bond Configuration Files

Complete the following steps to view the Cisco NFVI node interface bond configuration files:

1. SSH into the target node, for example, one of the Cisco VIM controllers:

```
[root@mgmt-node~]# ssh root@control-server-1
[root@control-server-1 ~]#
```

2. List all of the network bond configuration files in the */proc/net/bonding/* directory:

```
[root@control-server-1 ~]# ls
/proc/net/bonding/ a bond0 e mx p t
```

3. To view more information about a particular bond configuration, use the following command:

```
[root@control-server-1 ~]# more /proc/net/bonding/a
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)
Bonding Mode: load balancing (xor)
Transmit Hash Policy: layer3+4 (1)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
.
.
.
```

Viewing Cisco NFVI Node Route Information

Complete the following steps to view Cisco NFVI node route information. Note that this is not the HAProxy container running on the controller. The default gateway must point to the gateway on the management network using the *br_mgmt* bridge.

1. SSH into the target node, for example, one of the Cisco VIM controllers:

```
[root@mgmt-node~]# ssh root@control-server-1
[root@control-server-1 ~]#
```

2. View the routing table (default gateway) of the Cisco NFVI node:

```
server-1 ~]# route -n
Kernel IP routing table
Destination                Gateway                 Genmask                 Flags         Metric
Ref                 Use Iface
0.0.0.0                         10.23.221.33         0.0.0.0                 UG
0          0                                  0 br_mgmt
10.23.221.32                0.0.0.0                 255.255.255.240 U                 0
0                                0 br_mgmt
17.16.3.0                        0.0.0.0                 255.255.255.0         U
0          0                                  0 t
169.254.0.0                0.0.0.0                 255.255.0.0         U
1016          0                                  0 br_api
169.254.0.0                0.0.0.0                 255.255.0.0         U                 1017
0                           0 e
169.254.0.0                   0.0.0.0                 255.255.0.0         U
1019          0                                  0 br_mgmt
169.254.0.0                0.0.0.0                 255.255.0.0         U
1020          0                                  0 p
169.254.0.0                0.0.0.0                 255.255.0.0         U
1021          0                                  0 t
172.17.0.0                      0.0.0.0                 255.255.0.0         U
0          0                                  0 docker0
```

Viewing Linux Network Namespace Route Information

Complete the following steps to view the route information of the Linux network namespace that the HAProxy container uses on a Cisco NFVI controller node. The default gateway must point to the gateway on the API network, using the API interface in the Linux network namespace.

1. SSH into the target node, for example, one of the Cisco VIM controllers:

```
[root@mgmt-node~]# ssh root@control-server-1
[root@control-server-1 ~]#
```

2. Enter the *ip netns* command to find the name of the network namespace:

```
[root@control-server-2 ~]# ip netns 17550 (id: 0)
```

3. Enter the *ip netns exec* command to view the routing table (default gateway) of the Linux network namespace:

```
[root@control-server-2 ~]# ip netns exec 17550 route -n
Kernel IP routing table
Destination                 Gateway                 Genmask                 Flags         Metric
Ref                 Use Iface
0.0.0.0                         172.29.86.1         0.0.0.0                 UG
0                 0                         0 api
10.23.221.32                 0.0.0.0                 255.255.255.240 U                0
0                       0 mgmt
172.29.86.                         0 0.0.0.0                255.255.255.0         U
0                 0                         0 api
```

Pre-checks for Storage Removal

Pre-checks for Storage Removal

Upon completion of the pod management operations like add storage, ensure that any subsequent operation such as remove storage is done on the same storage node after accounting for all the devices, and their corresponding OSDs have been marked in the persistent crush map as shown in the output of the ceph osd crush tree.

1. To get a list of all the devices configured for ceph osds, execute the following command on the storage node where a remove-storage pod operation is performed:

```
[root@storage-3 ~]$ df | grep -oh ceph-[0-9]*
[root@storage-3 ~]$ df | grep -oh ceph-[0-9]*
ceph-1
ceph-5
ceph-7
ceph-10
```

2. Log into any of the controller nodes and run the following commands within the ceph mon container:

```
$ cephmon
$ ceph osd crush tree
```

3. From the json output, locate the storage node to be removed and ensure that all the devices listed for ceph osds have corresponding osd entries for them by running the following command:

```
        {
                                        "id": -3,
                                        "name": "storage-3",
                                        "type": "host",
                                        "type_id": 1,
                                        "items": [
                                                {
                                                        "id": 1,
                                                        "name": "osd.1",
                                                        "type": "osd",
                                                        "type_id": 0,
                                                        "crush_weight": 1.091095,
                                                        "depth": 2
                                                },
                                                {
                                                        "id": 5,
                                                        "name": "osd.5",
                                                        "type": "osd",
                                                        "type_id": 0,
                                                        "crush_weight": 1.091095,
                                                        "depth": 2
                                                },
                                                {
                                                        "id": 7,
                                                        "name": "osd.7",
                                                        "type": "osd",
                                                        "type_id": 0,
                                                        "crush_weight": 1.091095,
                                                        "depth": 2
                                                },
                                                {
                                                        "id": 10,
                                                        "name": "osd.10",
                                                        "type": "osd",
                                                        "type_id": 0,
                                                        "crush_weight": 1.091095,
                                                        "depth": 2
                                                }
                                        ]
                                },
```

- If ToR_TYPE is Cisco NCS 5500, you must manually remove all the sub-interfaces that were manually configured on the NCS switch, as Cisco VIM automation does not unconfigure/configure the sub-interfaces for which the VLANs were not defined in the *setup_data.yaml*.
- If manual removal of sub-interface is not done, remove-compute operation is initiated.

General Troubleshooting Procedures

General Troubleshooting Procedures

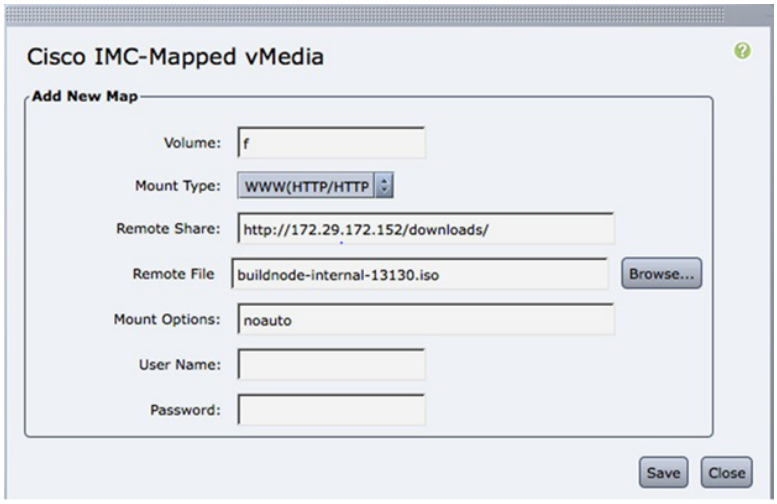
- Managing CIMC and ISO Installation
- Management Node Installation Failure
- Configuring Boot Order
- PXE Failure Issue During Baremetal Step
- Connecting to Docker Container

Managing CIMC and ISO Installation

When you are remote, it is good to map the ISO through the CIMC Mapped vMedia.

To add new mapping:

1. Click **Server > Remote Presence > Virtual Media > Add New Mapping**.

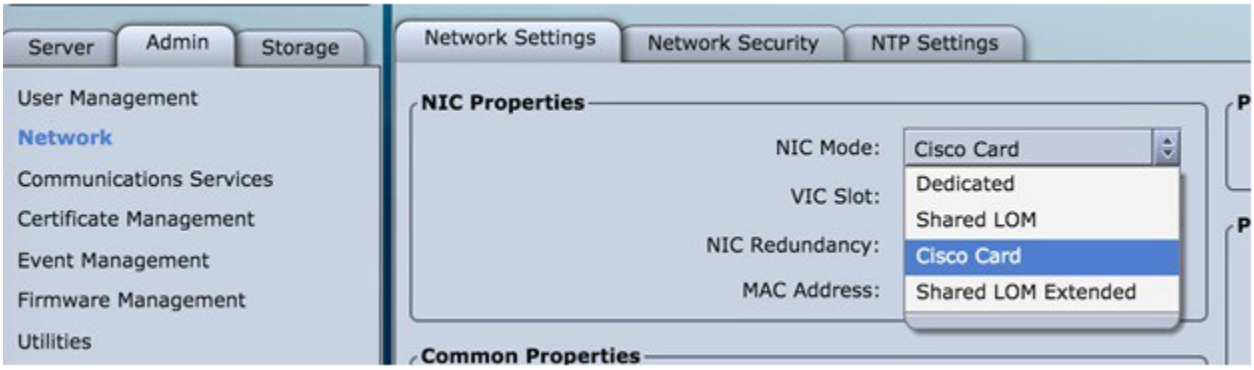


2. Enter the **Volume**, **Mount Type**, **Remote Share**, **Remote File**, **User name**, and **Password**.
3. Click **Save**. The CIMC pulls the ISO directly from the HTTP server.

Management Node Installation Failure

The management node installation fails, if the CIMC is configured for Cisco Card mode.

Choose the dedicated mode in the following screen:

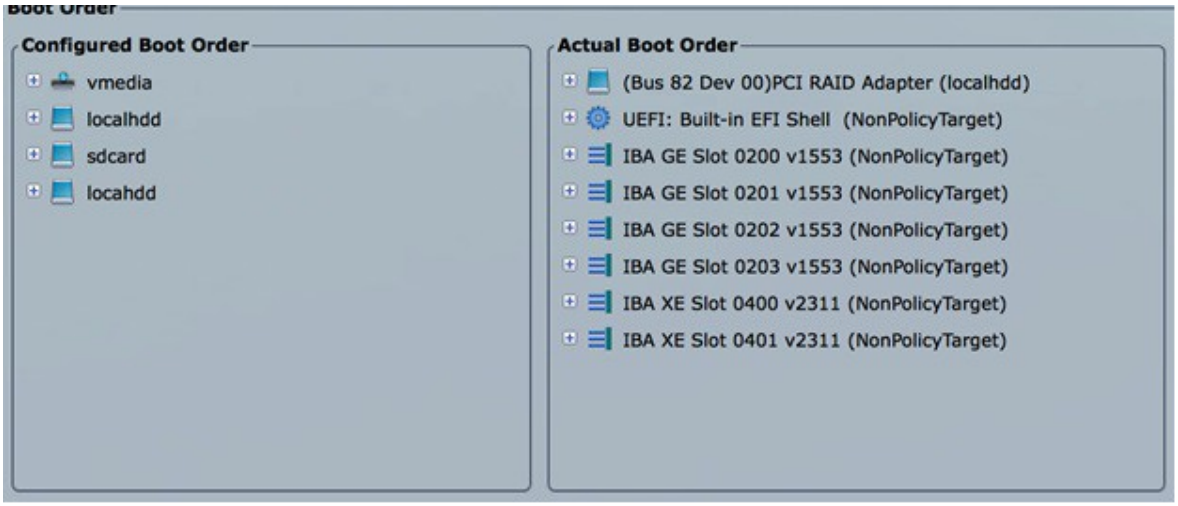


The selected method that is shown in the preceding screen is the incorrect mode.

Configuring Boot Order

The management node does not come up post reboot and must be booted from hard drive.

To check for the actual boot order, choose **Server > BIOS > Configure Boot Order > Boot Order**.



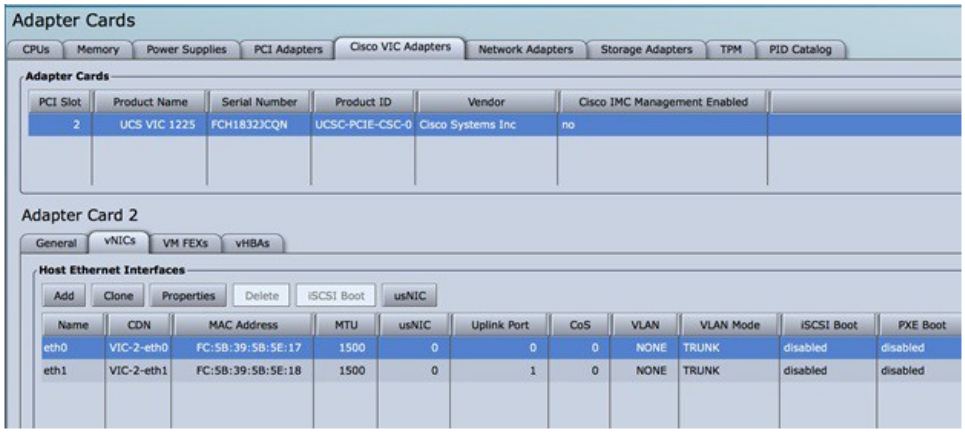
PXE Failure Issue During Baremetal Step

Perform the following steps in case of PXE boot failure:

1. Check the log file */var/log/mercury/mercury_baremetal_install.log* and connect to failing node CIMC KVM console to find out more on PXE boot failure reason.
2. Check log file */var/log/mercury/<UUID>/mercury_baremetal_install.log.*
3. Ensure that all validations in the above steps pass.
4. Connect to KVM console of failing node(s), to find out more on PXE boot failure.
5. Check L2/L3 network connectivity between failing node(s) and management node.
6. Check for VPC configuration and port-channel status of failing node(s) and ensure *no lacp suspend-individual* is configured on the port-channel.
7. Check if the actual PXE boot order does not differ from the boot-order configured.
8. Perform tcpdump on the management node interface br_mgmt, to watch for UDP port 67 (dhcp) or UDP pot 69 (tftp) tcpdump -I br_mgmt port 67 or port 69 # on the management node.
9. Perform tcpdump on the management node management interfacebr_mgmt on TCP 80 tcpdump -I br_mgmt port 80 # on the management node.
10. Check the apache log to watch the management IP address of failing node (if static allocated) tail *-f/var/log/cobblerhttpd/access_log* # on the management node.
11. For authorization required error messages during bare metal (Step 4) with CIMC operations such as hardware validations or cleaning up vNIC, check whether the maximum allowed simultaneous connections (4) are in use.

    All four connections are run when the third-party application monitoring CIMC does not properly close CIMC. This makes Cisco VIM installer not to login using *xmlapi* with valid username and password. Check Cisco IMC logs on CIMC (**Server > Faults and Logs > Cisco IMC Logs**) to know the reason for which the access is denied. The reason may be maximum session or incorrect credentials. The workaround is to disable third-party monitoring, wait at least 10 minutes and then perform Cisco VIM operations.
12. In case none of the nodes are getting DHCP address, DHCP requests arrive at the management node without giving any response. To check CIMC VIC adapter settings, choose **Server > Inventory > Cisco VIC Adapters > vNICs | VLAN & VLAN** mode. Ensure that the VLANs (both id and mode) configured does not match with that of N9K switch.

Option	Description
CIMC	Trunk : None
Switch	Access : vlan_mgmt



Connecting to Docker Container

To connect to the docker container, do the following:

```
# generally, aliases are created for all containers
# use alias to identify those
alias | grep in_container
# checking specific alias by name alias cobbler

# check docker containers
# alias created by CVIM
dp
# list docker containers
docker ps -a
# list docker images
docker images

# connecting to container
docker exec -it my_cobbler_<tag_id> /bin/bash

# connecting to docker container as privileged user
docker exec -it -u root my_cobbler_<tag_id> /bin/bash

# systemctl files
systemctl -a | egrep "docker-.*.service"

# check specific service
systemctl status mercury-restapi -l
systemctl status docker-vmtp

# restart specific service
systemctl restart docker-vmtp
```

Connection/Installation Problems

Connection/Installation Problems

- Container Download Problems
- Cisco IMC Connection Problems during Bare Metal Installation
- API VIP Connection Problems
- HAProxy Services Downtime after Initial Installation or HA Failover
- Management Node Problems

Container Download Problems

- Check installer logs log file */var/log/mercury/mercury_buildorchestration.log* for any build node orchestration failures including stuck *registry-populate local registry*. Downloading the Docker container from your management node can be slow.
- Check the network connectivity between the management node and the remote registry in *defaults.yaml* on the management node (*grep "^registry:" openstack-configs/defaults.yaml*).
- Verify if the valid remote registry credentials are defined in *setup_data.yaml* file.
- A proxy server is required to pull the container images from remote registry. If a proxy is required, exclude all IP addresses for your setup including management node.

Cisco IMC Connection Problems during Bare Metal Installation

If Cisco IMC has too many connections, the installer cannot connect to it.

To clear the connections,

1. Log into your Cisco IMC.
2. Select the **Admin>Sessions** tab
3. Clear the connections.

API VIP Connection Problems

Verify if the active HAProxy container is running in one of the controller nodes. On that controller within the HAProxy container namespace, verify whether the IP address is assigned to the API interface. Also, verify whether your ToR and the network infrastructure connecting your ToR are provisioned with API network segment VLAN.

HAProxy Services Downtime after Initial Installation or HA Failover

The HAProxy web interface is accessible on TCP port 1936.

```
http://<external_lb_vip_address>:1936/
Username: haproxy
Password: <HAPROXY_PASSWORD> from secrets.yaml file
```

After initial installation, the HAProxy web interface can report to several OpenStack services with downtime depending on when that OpenStack service is installed after HAProxy installation. The counters are not synchronized between HAProxy active and standby. After HA proxy failover, the downtime timers vary based on the uptime of new active HAproxy container.

Management Node Problems

Service Commands

To identify all the services that are running, enter the following commands:

```
$ systemctl -a | grep docker | grep service
On controller ignore status of:
docker-neutronlb
On compute ignore status of:
docker-neutronlb, docker-keystone
```

To start a service on a host, execute the below command:

```
$ systemctl start <service_name>
```

To stop a service on a host, run the following command:

```
$ systemctl stop <service_name>
```

To restart a service on a host, run the below command :

```
$ systemctl restart <service_name>
```

To check the service status on a host, run the following command:

```
$ systemctl status <service_name>
```

Management Node Recovery Scenarios

Management Node Recovery Scenarios

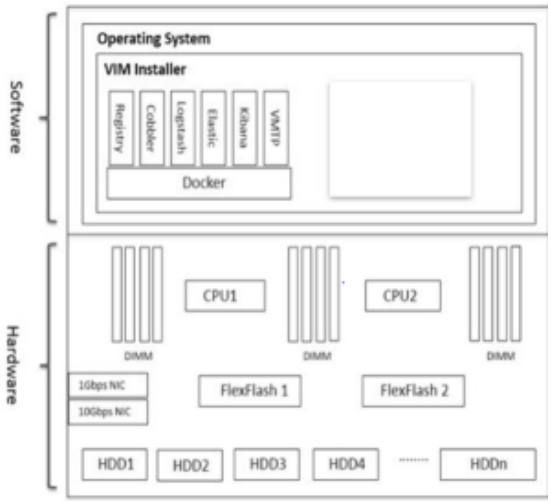
- Management Node Architecture
- Scenario 1: Failure of One or Two Active HDDs
- Scenario 2: Simultaneous Failure of More than Two Active HDDs
- Scenario 3: Spare HDD Failure
- Scenario 4: Power Outage or Reboot
- Scenario 5: System Reboot
- Scenario 6: Docker Daemon Start Failure
- Scenario 7: Service Container (Cobbler, ELK) Start Failure
- Scenario 8: One link failure on the bond Interface
- Scenario 9: Two Link Failures on the Bond Interface
- Scenario 10: REST API Service Failure
- Scenario 11: Graceful Reboot with Cisco VIM Unified Management
- Scenario 12: Power Outage or Hard Reboot With VIM Unified Management
- Scenario 13: Cisco VIM Unified Management Reinstallation
- Scenario 14: VIM Unified Management Container reboot
- Scenario 15: Intel (I350) 1Gbps LOM failure
- Scenario 16: Cisco VIC 1227 10Gbps mLOM failure
- Scenario 17: DIMM Memory Failure
- Scenario 18: One CPU Failure

Management Node Architecture

The Cisco NFVI management node hosts the Cisco VIM Rest API service, Cobbler for PXE services, ELK for Logging to Kibana dashboard services, and VMTP for cloud validation. As the maintenance node does not have redundancy, understanding its point of failure and recovery scenarios are important. Managing node recovery scenarios are described in the following steps.

The management node architecture includes a Cisco UCS C240 M4 server with dual CPU socket. It has a 1-Gbps on-board (LOM) NIC and a 10-Gbps Cisco VIC mLOM. HDDs are used in 8,16, or 24 disk configurations.

The following figure shows the high-level maintenance node of the hardware and software architecture.



Different management node hardware or software failures can cause Cisco NFVI service disruptions and outages. You can recover few failed services through manual intervention. If the system is operational during a failure, double faults cannot be recoverable.

The following table lists the management node failure scenarios and their recovery options:

Scenario #	Failure or Trigger	Recoverable	Operational Impact
1	Failure of 1 or 2 active HDD	Yes	No
2	Simultaneous failure of more than 2 active HDD	No	Yes
3	Spare HDD failure: 4 spare for 24 HDD; or 2 spare for 8 HDD	Yes	No
4	Power outage/hard reboot	Yes	Yes
5	Graceful reboot	Yes	Yes
6	Docker daemon start failure	Yes	Yes

7	Service container (Cobbler, ELK) start failure	Yes	Yes
8	One link failure on bond interface	Yes	No
9	Two link failures on bond interface	Yes	Yes
10	REST API service failure	Yes	No
11	Graceful reboot with Cisco VIM Unified Management	Yes	Yes. CLI alternatives exist during reboot.
12	Power outage or hard reboot with Cisco VIM Unified Management	Yes	Yes
13	VIM Unified Management Container reinstallation	Yes	Yes. CLI alternatives exist during reinstall.
14	Cisco VIM Unified Management Container reboot	Yes	Yes. CLI alternatives exist during reboot.
15	Intel 1350 1Gbps LOM failure	Yes	Yes
16	Cisco VIC 1227 10-Gbps mLOM failure	Yes	Yes
17	DIMM memory failure	Yes	No
18	One CPU failure	Yes	No

Scenario 1: Failure of One or Two Active HDDs

The management node uses 8,16, or 24-HDDs. The HDDs are configured with RAID 6, to enable data redundancy and storage performance, and to overcome any unforeseen HDD failures.

- When eight HDDs are installed, seven are active disks and one is spare disk.
- When 16 HDDs are installed, 14 are active disks and two are spare disks.
- When 24 HDDs are installed, 20 are active disks and four are spare disks.

With RAID 6 up, two simultaneous active HDD failures can occur. When an HDD fails, the system begins automatic recovery by moving the spare disk to active state and rebuilding the new active HDD. It takes approximately four hours to rebuild the new disk and move to synchronized state. During this operation, the system is fully functional without causing any impact. However, you must monitor the system to ensure that more failures do not occur to enter into a double fault situation.

You can use the *storcli* commands to check the disk and RAID state as shown in the following commands:

Ensure that the node is running with hardware RAID, by checking the *storcli* output and comparing it to the one preceding.

```
[root@mgmt-node ~]# /opt/MegaRAID/storcli/storcli64 /c0 show
<...snip...>
TOPOLOGY:
========
-----------------------------------------------------------------------------
DG Arr Row EID:Slot DID Type State BT                Size         PDC         PI SED         DS3
FSpace TR
-----------------------------------------------------------------------------
0         -         -         -                 -         RAID6 Optl         N           4.087 TB  dflt
N         N         dflt         N                 N
0         0         -          -      -         RAID6 Optl         N           4.087 TB  dflt         N
N         dflt         N                 N <== RAID
6 in optimal state
0         0         0         252:1         1         DRIVE Onln         N         837.258 GB  dflt
N         N         dflt         -                 N
0         0         1         252:2         2        DRIVE Onln         N         837.258 GB  dflt
N         N         dflt         -                 N
0         0         2         252:3         3        DRIVE Onln         N         930.390 GB  dflt
N         N         dflt         -                 N
0         0         3         252:4         4         DRIVE Onln         N         930.390 GB  dflt
N         N         dflt         -                 N
0         0         4         252:5         5         DRIVE Onln         N         930.390 GB  dflt
N         N         dflt         -                 N
0         0         5         252:6         6         DRIVE Onln         N         930.390 GB  dflt
N         N         dflt         -                 N
0         0         6         252:7         7         DRIVE Onln         N         930.390 GB  dflt
N         N         dflt         -                 N
0         -         -         252:8         8         DRIVE DHS         -         930.390 GB   -
-         -          -           -                 N
-----------------------------------------------------------------------------
<...snip...>
PD LIST:
```

```
=======
------------------------------------------------------------------------------
EID:Slt DID State         DG                Size         Intf         Med          SED PI         SeSz
Model                 Sp
------------------------------------------------------------------------------
252:1         1         Onln         0         837.258 GB         SAS         HDD          N          N
512B ST900MM0006  U <== all disks
functioning
252:2         2         Onln         0         837.258 GB         SAS         HDD          N          N
512B ST900MM0006  U
252:3         3         Onln         0         930.390 GB         SAS                HDD          N
N         512B ST91000640SS U
252:4         4         Onln         0         930.390 GB         SAS                HDD          N
N         512B ST91000640SS U
252:5         5         Onln         0         930.390 GB         SAS         HDD          N          N
512B ST91000640SS U
252:6         6         Onln         0         930.390 GB         SAS                HDD          N
N         512B ST91000640SS U
252:7         7         Onln         0         930.390 GB         SAS                HDD          N
N         512B ST91000640SS U
252:8         8         DHS          0         930.390 GB         SAS         HDD          N          N
512B ST91000640SS D
------------------------------------------------------------------------------
[root@mgmt-node ~]# /opt/MegaRAID/storcli/storcli64 /c0 show
<...snip...>
TOPOLOGY :
========
-----------------------------------------------------------------------------------------
DG         Arr Row EID:Slot DID         Type State BT                 Size         PDC          PI SED
DS3         FSpace TR
-----------------------------------------------------------------------------------------
0         -         -         -                 -                 RAID6 Pdgd N         4.087 TB
dflt         N         N dflt         N                 N <== RAID 6
in degraded state
0         0         -         -                 -                 RAID6 Dgrd N         4.087 TB
dflt         N         N dflt         N                 N
0         0         0         252:8         8                 DRIVE Rbld Y         930.390 GB
dflt         N         N dflt         -                 N
0         0         1         252:2         2                 DRIVE Onln N         837.258 GB
dflt         N         N dflt         -                 N
0         0         2         252:3         3                 DRIVE Onln N         930.390 GB
dflt         N        N dflt         -                N
0         0        3         252:4         4                DRIVE Onln N         930.390 GB
dflt         N         N dflt         -                 N
0         0        4         252:5         5                DRIVE Onln N         930.390 GB
dflt         N         N dflt         -                 N
0         0         5         252:6         6                 DRIVE Onln N         930.390 GB
dflt         N         N dflt         -                 N
0         0         6         252:7         7                 DRIVE Onln N         930.390 GB
dflt         N         N dflt         -                 N
-----------------------------------------------------------------------------------------
<...snip...>
PD LIST :
========
-------------------------------------------------------------------------------
EID:Slt         DID State         DG                Size                Intf         Med SED PI SeSz
Model                 Sp
-------------------------------------------------------------------------------
252:1                 1         UGood         -         837.258 GB                 SAS         HDD N         N
512B ST900MM0006         U <== active disk
in slot 1 disconnected from drive group 0
252:2                 2         Onln         0         837.258 GB                 SAS         HDD N         N
512B ST900MM0006         U
252:3                 3         Onln         0         930.390 GB                 SAS         HDD N         N
512B ST91000640SS U
252:4                 4         Onln         0         930.390 GB                 SAS         HDD N         N
512B ST91000640SS U
252:5                 5         Onln         0         930.390 GB                 SAS         HDD N         N
512B ST91000640SS U
252:6                 6         Onln         0        930.390 GB                SAS         HDD N         N
512B ST91000640SS U
```

```
252:7                 7         Onln         0        930.390 GB                 SAS         HDD N         N
512B ST91000640SS U
252:8                 8         Rbld         0         930.390 GB                 SAS         HDD N         N
512B ST91000640SS U <== spare disk
in slot 8 joined drive group 0 and in rebuilding state
-------------------------------------------------------------------------------
[root@mgmt-node ~]# /opt/MegaRAID/storcli/storcli64 /c0/e252/s8 show rebuild
Controller = 0
Status = Success
Description = Show Drive Rebuild Status Succeeded.
------------------------------------------------------------
Drive-ID                Progress%                Status         Estimated Time Left
------------------------------------------------------------
/c0/e252/s8                   20         In progress 2 Hours   28 Minutes <== spare disk in slot 8 rebuild
status
------------------------------------------------------------
```

To replace the failed disk and add it back as a spare:

```
[root@mgmt-node ~]# /opt/MegaRAID/storcli/storcli64 /c0/e252/s1 add hotsparedrive dg=0
Controller = 0
Status = Success
Description = Add Hot Spare Succeeded.
[root@mgmt-node ~]# /opt/MegaRAID/storcli/storcli64 /c0 show
<...snip...>
TOPOLOGY :
========
-----------------------------------------------------------------------------
DG Arr Row         EID:Slot         DID Type State BT                 Size         PDC PI SED DS3
FSpace         TR
-----------------------------------------------------------------------------
0         -         -         -                         -         RAID6 Pdgd N           4.087 TB dflt
N         N dflt         N                N
0         0         -         -                         -         RAID6 Dgrd N           4.087 TB dflt
N         N dflt         N                N
0        0         0         252:8                8         DRIVE Rbld Y         930.390 GB dflt N         N
dflt         -                N
0         0         1         252:2                2         DRIVE Onln N         837.258 GB dflt N         N
dflt         -                N
0         0         2         252:3                3         DRIVE Onln N         930.390 GB dflt N         N
dflt         -                N
0         0         3         252:4                4         DRIVE Onln N         930.390 GB dflt N         N
dflt         -                N
0         0         4         252:5                5         DRIVE Onln N         930.390 GB dflt N         N
dflt         -                N
0         0         5         252:6                6         DRIVE Onln N         930.390 GB dflt N         N
dflt         -                N
0         0         6         252:7                7         DRIVE Onln N         930.390 GB dflt N         N
dflt         -                N
0         -         -         252:1                1         DRIVE DHS  -         837.258 GB         -
-         -         -         -                N
-----------------------------------------------------------------------------
<...snip...>
PD LIST :
=======
-----------------------------------------------------------------------------
EID:Slt DID State         DG                 Size         Intf         Med SED PI         SeSz
Model                         Sp
-----------------------------------------------------------------------------
252:1         1         DHS         0         837.258 GB         SAS         HDD N         N         512B
ST900MM0006         U <== replacement
disk added back as spare
252:2         2         Onln         0         837.258 GB         SAS         HDD N         N         512B
ST900MM0006         U
252:3         3         Onln         0         930.390 GB         SAS         HDD N         N         512B
ST91000640SS         U
252:4         4         Onln         0         930.390 GB         SAS         HDD N         N         512B
ST91000640SS         U
252:5         5         Onln         0         930.390 GB         SAS         HDD N         N         512B
ST91000640SS         U
252:6         6         Onln         0         930.390 GB         SAS         HDD N         N         512B
ST91000640SS         U
252:7         7         Onln         0         930.390 GB         SAS         HDD N         N         512B
ST91000640SS         U
252:8         8         Rbld         0         930.390 GB         SAS         HDD N         N         512B
ST91000640SS         U
-----------------------------------------------------------------------------
```

Scenario 2: Simultaneous Failure of More than Two Active HDDs

If more than two HDD failures occur at the same time, the management node goes into an unrecoverable failure state as RAID 6 allows for recovery of up to two simultaneous HDD failures. To recover the management node, reinstall the operating system.

Scenario 3: Spare HDD Failure

When the management node has 24 HDDs, four are designated as spares. Failure of any of the disks does not impact the RAID or system functionality. Cisco recommends replacing these disks when they fail (see the steps in Scenario 1) to serve as standby disks and so when an active disk fails, an auto-rebuild is triggered.

Scenario 4: Power Outage or Reboot

If a power outage or hard system reboot occurs, the system boots up and comes back to operational state. Services running on the management node during downtime gets disrupted. See the steps in Scenario 9 for the list of commands to check the services status after recovery.

Scenario 5: System Reboot

If a graceful system reboot occurs, the system boots up and come back to operational state. Services running on the management node during down time gets disrupted. See the steps in Scenario 9 for the list of commands to check the services status after recovery.

Scenario 6: Docker Daemon Start Failure

The management node runs the services using Docker containers. If the Docker daemon fails to come up, it causes services such as ELK, Cobbler, and VMTP to go into down state. You can use the *systemctl* command to check the status of the Docker daemon, for example:

```
# systemctl status docker
docker.service - Docker Application Container Engine
Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2016-08-22 00:33:43 CEST; 21h ago
Docs: http://docs.docker.com
Main PID: 16728 (docker)
```

If the Docker daemon is in down state, use the *systemctl restart docker* command to restart the Docker service. Run the commands that are listed in Scenario 9 to verify that all the Docker services are active.

Scenario 7: Service Container (Cobbler, ELK) Start Failure

As described in Scenario 8, all the services run as Docker containers on the management node. To find all services running as containers, use the *docker ps –a* command. If any services are in *Exit* state, use the *systemctl* command and grep for Docker to find the exact service name, for example:

```
# systemctl | grep docker- | awk '{print $1}'
docker-cobbler-tftp.service
docker-cobbler-web.service
docker-cobbler.service
docker-container-registry.service
docker-elasticsearch.service
docker-kibana.service
docker-logstash.service
docker-vmtp.service
```

If any services need restarting, use the *systemctl* command. For example, to restart a Kibana service:

```
# systemctl restart docker-kibana.service
```

Scenario 8: One link failure on the bond Interface

The management node is set up with two different networks: *br_api* and *br_mgmt.* The *br_api* is the external interface used for accessing external services such as Cisco VIM REST API, Kibana, and Cobbler. The *br_mgmt* is an internal interface used for providing management connectivity to all OpenStack nodes (control, compute and storage). Each network has two ports that are bonded to provide redundancy. If one port fails, the system remains completely functional through the other port. If a port fails, check for physical network connectivity and remote switch configuration to debug the underlying cause of the link failure.

Scenario 9: Two Link Failures on the Bond Interface

As described in Scenario 10, each network is configured with two ports. If both ports are down, the system is not reachable and management node services may be disrupted. After the ports are up, the system is fully operational. Check the physical network connectivity and the remote switch configuration, to debug the underlying link failure cause.

Scenario 10: REST API Service Failure

The management node runs the REST API service for Cisco VIM clients to reach the server. If the REST service is down, Cisco VIM clients cannot reach the server to trigger any server operations. However, with the exception of the REST service, other management node services remain operational.

To verify whether the management node REST services are fully operational, use the following command to check that the httpd and mercury-restapi services are in active and running state:

```
# systemctl status httpd
httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2016-08-22 00:22:10 CEST; 22h ago
# systemctl status mercury-restapi.service
mercury-restapi.service - Mercury Restapi
Loaded: loaded (/usr/lib/systemd/system/mercury-restapi.service; enabled; vendor preset:
disabled)
Active: active (running) since Mon 2016-08-22 00:20:18 CEST; 22h ago
```

Use the *restapi* tool to check the REST API server status and the location of the folder where it is running, by executing 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 whether the server status is active and check that the *restapi* launch folder matches the folder where the installation is launched. The *restapi* tool also provides the options to launch, tear down, and reset password for the *restapi* server as shown in the following command:

```
# ./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.
                                         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.
```

If the REST API server is not running, execute *ciscovim* to show the following error message:

```
# cd installer-<tagid>/
# ciscovim -setupfile ~/Save/<setup_data.yaml> run
```

If the installer directory or the REST API state is not correct or points to an incorrect REST API launch directory, navigate to the installer-*<tagid>/tools* directory and execute:

```
# ./restapi.py -action setup
```

To confirm that the REST API server state and launch directory are correct, run the following command:

```
# ./restapi.py -action status
```

Scenario 11: Graceful Reboot with Cisco VIM Unified Management

Cisco VIM Unified Management runs as a container on the management node. After a graceful reboot of the management node, the Cisco VIM Unified Management and its associated database containers comes up. Hence, there is no impact on recovery.

Scenario 12: Power Outage or Hard Reboot With VIM Unified Management

The Cisco VIM Unified Management container comes up automatically following a power outage or hard reset of the management node.

Scenario 13: Cisco VIM Unified Management Reinstallation

If the management node which is running the Cisco VIM Unified Management fails and cannot come up, you must uninstall and reinstall the Cisco VIM UM. After the Unified Management container comes up, add the relevant bootstrap steps listed as part of installation to register the pod. Cisco VIM UM automatically detects the installer status and reflects the present status appropriately.

To clean up and reinstall Cisco VIM UM, run the following command:

```
# cd /root/installer-<tagid>/insight/
# ./bootstrap_insight.py -a uninstall -o standalone -f </root/insight_setup_data.yaml>
```

Scenario 14: VIM Unified Management Container reboot

On reboot of the VIM Unified Management container, services continue to work as it is.

Scenario 15: Intel (I350) 1Gbps LOM failure

The management node is set up with an Intel (I350) 1-Gbps LOM for API connectivity. Two 1-Gbps ports are bonded to provide connectivity redundancy. No operational impact occurs, if one of these ports goes down. However, if both ports fail, or the LOM network adapter fails, the system cannot be reached through the API IP address. If this occurs, you must replace the server as the LOM is connected to the system motherboard. To recover the management node with a new server, complete the following steps. Ensure that the new management node hardware profile matches the existing server with the Cisco IMC IP address assigned.

1. Shut down the existing management node.
2. Unplug the power from the existing and new management nodes.
3. Remove all HDDs from existing management node and install them in the same slots of the new management node.
4. Plug in the power to the new management node, but do not boot the node.
5. Verify whether the configured boot order is set to boot from local HDD.
6. Verify whether the Cisco NFVI management VLAN is configured on the Cisco VIC interfaces.
7. Boot the management node for the operating system to begin. After the management node is up, the management node bond interface is down due to the incorrect MAC address and it points to old node network card MAC address.
8. Update the MAC address under */etc/sysconfig/*network-scripts.
9. Reboot the management node. It is fully operational. All interfaces must be in an *up* state and be reachable.
10. Verify that the Kibana and Cobbler dashboards are accessible.
11. Verify the Rest API services are up. See Scenario 15 for any recovery steps.

Scenario 16: Cisco VIC 1227 10Gbps mLOM failure

The management node is configured with a Cisco VIC 1227 dual port 10-Gbps mLOM adapter for connectivity to the other Cisco NFVI nodes. Two 10 Gbps ports are bonded to provide connectivity redundancy. If one of the 10-Gbps ports goes down, no operational impact occurs. However, if both Cisco VIC 10 Gbps ports fail, the system goes into an unreachable state on the management network. If this occurs, you must replace the VIC network adapters. Otherwise pod management and Fluentd forwarding service are disrupted.

If you replace a Cisco VIC, update the following:

- Management and provisioning VLAN for the VIC interfaces using Cisco IMC,
- MAC address in the interfaces under */etc/sysconfig/network-scripts* interface configuration file.

Scenario 17: DIMM Memory Failure

The management node is set up with multiple DIMM memory across different slots. Failure of one or memory modules may cause the system to go into unstable state, depending on the number of occurrences of DIMM memory failures . DIMM memory failures are standard system failures like any other Linux system server. If a DIMM memory fails, replace the memory module(s) as soon as possible to keep the system in stable state.

Scenario 18: One CPU Failure

Cisco NFVI management nodes have dual core Intel CPUs (CPU1 and CPU2). If one CPU fails, the system remains operational. However, always replace failed CPU modules immediately. CPU failures are standard system failures such as any other Linux system server. If a CPU fails, replace it immediately to keep the system in stable state.

Compute Node Recovery Scenario

Compute Node Recovery Scenario

The Cisco NFVI compute node hosts the OpenStack services to provide processing, network, and storage resources to run instances. The node architecture includes a Cisco UCS C220 M4/M5 server with dual CPU socket, NIC of different combinations, and SSD/HDDs in RAID 1 configuration.

Recovery from Failure of One Active HDD/SSD

With RAID 1, data is shown but limited to one active HDD/SSD failure. When an HDD/SSD fails, the node is still functional with no impact. However, the data are no longer illustrated. Losing another HDD/SSD results in unrecoverable and operational downtime. You must replace the failed disk soon, as it takes approximately two hours to rebuild the new disk and move to synchronized state.

To check the disk and RAID state, run the *storcli* commands as follows:

Ensure that the node is running with hardware RAID, by checking the *storcli* output and comparing to the one that is shown in the following command.

```
[root@compute-node ~]# /opt/MegaRAID/storcli/storcli64 /c0 show
<...snip...>
TOPOLOGY :
========
-----------------------------------------------------------------------------------
DG         Arr Row  EID:Slot   DID   Type  State BT         Size     PDC         PI SED DS3 FSpace TR
-----------------------------------------------------------------------------------
0         -         -                 -            -      RAID1  Optl  N  837.258 GB dflt         N         N
dflt         N            N <== RAID 1 in
optimal state
0         0         -                 -            -      RAID1  Optl  N  837.258 GB dflt         N         N
dflt         N            N
0         0         0          252:2      9     DRIVE  Onln  N  837.258 GB dflt         N         N
dflt         -      N
0         0         1          252:3      11    DRIVE  Onln  N  837.258 GB dflt         N         N
dflt         -      N
-----------------------------------------------------------------------------------
<...snip...>
Physical Drives = 2
PD LIST :
=======
-------------------------------------------------------------------------
EID:Slt         DID State DG                Size         Intf Med         SED PI SeSz Model                 Sp
-------------------------------------------------------------------------
252:2                9         Onln   0  837.258 GB         SAS  HDD         N         N  512B ST900MM0006 U
<== all disks
functioning
252:3                11         Onln   0  837.258 GB         SAS  HDD         N         N  512B ST900MM0006 U
-------------------------------------------------------------------------
[root@compute-node ~]# /opt/MegaRAID/storcli/storcli64 /c0 show
<...snip...>

TOPOLOGY :
========
-----------------------------------------------------------------------------------------
DG Arr Row  EID:Slot         DID         Type         State         BT                 Size         PDC PI SED
DS3 FSpace         TR
-----------------------------------------------------------------------------------------
0         -         -                 -                -                 RAID1         Dgrd         N
837.258 GB dflt  N  N dflt         N                N <== RAID 1 in
degraded state.
0         0         -                 -                -                 RAID1         Dgrd         N
837.258 GB dflt  N  N dflt         N                N
0         0         0                 -                -                 DRIVE         Msng         -
837.258 GB  -          -  -  -         -                N
0         0         1         252:3               11                DRIVE         Onln         N
837.258 GB dflt  N  N dflt         -                N
-----------------------------------------------------------------------------------------
<...snip...>
PD LIST :
=======
-------------------------------------------------------------------------
EID:Slt         DID         State         DG                Size         Intf Med SED PI SeSz   Model    Sp
-------------------------------------------------------------------------
252:2                9                UGood         -         837.258 GB  SAS  HDD  N  N  512B ST900MM0006 U
<== active disk
in slot 2 disconnected from drive group 0
252:3                11                Onln         0         837.258 GB  SAS  HDD  N  N  512B ST900MM0006 U
-------------------------------------------------------------------------
```

To replace the failed disk and add it back as a spare, run the following command:

```
[root@compute-node ~]# /opt/MegaRAID/storcli/storcli64 /c0/e252/s2 add hotsparedrive dg=0
Controller = 0
Status = Success
Description = Add Hot Spare Succeeded.
[root@compute-node ~]# /opt/MegaRAID/storcli/storcli64 /c0 show
<...snip...>
TOPOLOGY :
========
-----------------------------------------------------------------------------
DG         Arr Row   EID:Slot  DID Type  State BT    Size     PDC PI SED DS3 FSpace TR
-----------------------------------------------------------------------------
0         -         -             -                 -   RAID1 Dgrd  N  837.258 GB dflt N   N dflt  N      N
0         0         -                 -                 -   RAID1 Dgrd  N  837.258 GB dflt N   N dflt  N      N
0         0         0           252:2         9   DRIVE Rbld  Y  837.258 GB dflt N   N dflt  -      N
0         0         1           252:3         11  DRIVE Onln  N  837.258 GB dflt N   N dflt  -      N
-----------------------------------------------------------------------------
<...snip...>
PD LIST :
=======
-------------------------------------------------------------------------
EID:Slt         DID State DG    Size    Intf Med SED PI SeSz    Model   Sp
-------------------------------------------------------------------------
252:2                9         Rbld  0  837.258 GB SAS  HDD  N  N  512B ST900MM0006 U <== replacement
disk in slot 2 joined device group 0 and in rebuilding state
252:3                11  Onln  0  837.258 GB SAS  HDD  N  N  512B ST900MM0006 U
-------------------------------------------------------------------------

[root@compute-node ~]# /opt/MegaRAID/storcli/storcli64 /c0/e252/s2 show rebuild
Controller = 0
Status = Success
Description = Show Drive Rebuild Status Succeeded.

-----------------------------------------------------------
Drive-ID         Progress%         Status                        Estimated Time Left
-----------------------------------------------------------
/c0/e252/s2         10                In progress         1 Hours 9 Minutes <== replacement disk in slot 2
rebuild
status
-----------------------------------------------------------
```

Technical Support Tools

Technical Support Tools

- Running Cisco VIM Technical Support Tool
- Tech-Support Configuration File
- Tech-Support if Servers Are Offline
- Running Cisco VIM Software Hub Technical Support Tool

Running Cisco VIM Technical Support Tool

Cisco VIM includes a tech-support tool that you can use to gather Cisco VIM information to help solve issues working with Cisco Technical Support. With tech-support tool, you can execute custom scripts. You must execute the runner at least once, before using the tech-support tool. This tool uses a configuration file that specifies what information to collect.

The configuration file is located in the location: */root/openstack-configs/tech-support/tech_support_cfg.yaml.*

The tech-support tool checks the point where the Cisco VIM installer is executed, and collects the output of files or commands that are indicated by the configuration file. For example, if the installer fails during validation, the tech-support provides information that is listed in the configuration file up to validation. You can override this default behavior, by adding the *--stage* option to the command.

The tech-support script is located at the *management node /root/installer-{tag-id}/tech-support* directory. To run it after the runner execution, enter the following command:

```
./tech-support/tech_support.py
```

The command creates a compressed tar file containing all the information that is gathered. The file location is displayed on the console once the execution is complete. You need not have to execute the command with any options. However, if you want to override any default behavior, you can use the following options:

```
#/tech_support.py --help
Usage: tech_support.py [options]

Tech-support collects information about your cloud

Options:
-h, --help                                show this help message and exit
--stage=STAGE                          specify the stage where installer left off
--config-file=CFG_FILE
                                          specify alternate configuration file name
--tmp-dir=TMP_DIR                 specify alternate temporary directory name
--file-size=TAIL_SIZE
                                          specify max size (in KB) of each file collected
--host-list=HOST_LIST
                                          List (comma separated) of the hostnames of the servers
                                          to collect info from
--ip-list=IP_LIST                 List (comma separated) of the IPv4 of the hosts to
                                          collect info from
--exclude-mgmt-node         specify if mgmt node info needs to be excluded
--include-cimc                          specify if cimc techsupport needs to be included
--include-hw-diags                specify if hardware diagnostics need to be included
```

Where:

- stage: Indicates the state at which the installer left off. The possible values are: INPUT_VALIDATION, BUILDNODE_ORCHESTRATION, VALIDATION, BAREMETAL_INSTALL, COMMON_SETUP, CEPH, ORCHESTRATION or VMTP.
- config-file: Provides the path for a specific configuration file. Ensure that your syntax is correct. Look at the default */root/tech-support/openstack-configs/tech_support_cfg.yaml* file as an example on how to create a new config-file or modify the default file.
- tmp-dir: Provides the path to a temp directory that tech-support can use to create the compressed tar file. The tech-support tool provides the infrastructure to execute standard Linux commands from packages that are included in the Cisco VIM installation. This infrastructure is extensible and you can add commands, files, or custom bash or Python scripts into the configuration file pane for the tool to collect the output of those commands or scripts. See the *README* pane for more details.
- file-size: Is an integer that specifies (in KB) the maximum file size that tech-support captures and tail the file if needed. By default, this value is set to 10 MB. For example, if no file-size option is provided and the tech-support needs to collect */var/log/mercury/data.log* and the data.log is more than 10 MB, tech-support gets the last 10 MB from */var/log/mercury/data.log.*
- host-list: Provides the list of hosts for which information needs to be collected from the tech-support through hostname. By default, all hosts are selected.
- ip-list: Provides the list of hosts to get tech-support, when their management IPv4 defaults to all hosts.
- exclude-mgmt-node: It is an option to not get tech-support from the management node.

- include-cimc: Only applied for Cisco servers. This option allows to specify the list of hosts to get the CIMC tech-support. You can use this option along with the *–host-list* and *–ip-list* options.
- include-hw-diags: Only applied for Quanta servers. This option allows to specify the list of hosts to get the hardware support information collected on Quanta servers. It also collects the hardware information of the management node, if it is a Quanta server. This option can be used along with the *–host-list* option.

You can avail tech-support for CIMC via this tool. With the given design associated to the CIMC tech-support command, ensure that you do not use *–include-cimc* option by default. It is recommended to use tech-support for CIMC for specific servers where issues are seen. Ensure that you use a maximum of three servers at a time using the below command:

```
#./tech-support.py --include-cimc --host-list=server_hostname_1,server_hostname_2
or
#./tech-support.py -include-cimc --ip-list=cimc_ip_host1,cimc_ip_host2
```

When using the *ip-list* option, provide the list of the management IP addresses. The tech-support can figure out the CIMC IP address from the *setup_data.yaml* file.

Tech-Support Configuration File

Cisco VIM tech-support is a utility tool is designed to collect the VIM pod logs which help users to debug the issues offline. The administrator uses the tech-support configuration files to provide the list of commands or configuration files. The tech support tool of the Cisco VIM gathers list of commands or configuration files for the offline diagnostic or debugging purposes.

By default the tech-support configuration file is located at the */root/openstack-configs/tech-support/tech_support_cfg.yaml* file. Alternatively, you can use a different one by specifying the *-config-file* option. The syntax of this configuration file must be as follows:

The tech-support configuration file section is divided into eight sections which correspond to each installer stage:

- INPUT_VALIDATION
- BUILDNODE_ORCHESTRATION
- VALIDATION
- BAREMETAL_INSTALL
- COMMON_SETUP
- CEPH
- ORCHESTRATION
- VMTP

For each of these eight sections, there are tags divided on hierarchical levels. At the first level, the tag indicates the host(s) or path on which the command (s) run and from where the file(s) can be collected.

The possible tags are as follows:

- - HOSTS_MANAGEMENT : Run in the management node only
- - HOSTS_CONTROL : Run in all the control nodes
- - HOSTS_COMPUTE : Run in all the compute nodes
- - HOSTS_STORAGE : Run in all the storage nodes
- - HOSTS_COMMON : Run in all the compute and control nodes
- - HOSTS_ALL : Run in all the compute, control and storage nodes

If *HOSTS* tag is not specified in any of these eight sections, no information is collected for that stage.

For each of the hosts mentioned above, there is a second level tag which specifies where to run the command. The possible values of those tags are as follows:

- - SERVER_FILES: Path(s) to the file(s) that tech-support needs to collect.
- - SERVER_COMMANDS: Command(s) or script name(s) that needs to be executed directly on the server. The command(s) must be included before in the $PATH. For the scripts, see Custom scripts given below.
- - CONTAINERS: Indicates the tech-support tool about the command(s) to be executed and the files to be gathered from a container. See the following steps for more specific information of what can be added in this section.

In the CONTAINERS section, indicate the path where the commands are to be executed or gathered from container, with a <*container_name*> tag. The following are shown to get the string for the<*container_name*> tag:

- all_containers: Execute inside all containers (regardless of the state).
- <container_name>: The Container Name must be the name of a container. It indicates the container in which the command needs to run for gathering the information. It runs commands inside the container only if the mentioned container is up (as commands cannot be run on dead containers). Examples of how to get the container name:

Execute *docker ps* and get the name (without any numbers) of the last column of output *docker ps -a.*

For example:

```
CONTAINER ID IMAGE COMMAND <snip> NAMES 81bc4e54cbfb <registry>/vmtp:4263 /bin/bash" vmtp_4263
```

The tech-support runs the Linux commands on the server (from packages that is included in RHEL7.3). Add the name of the commands under the *SERVE R_COMMANDS* section of the configuration file to run the commands.

If the administrator wants to add a custom bash or execute python script in some set of servers in the cloud, add the script into the *custom-scripts* directory in the current directory path */root/openstack-configs/tech-support/* and add the script name into the corresponding *SERVER_COMMANDS* section.

The tech-support tool will scp the script(s) in the *custom-scripts* directory, into the appropriate cloud nodes where it will be executed based on the config file. This tool captures the output (stdout and stderr) and adds it to the collection of files. It is assumed that the scripts are self-standing and independent, and needs no external input.

Following is an example of a custom tech-support configuration file. This example shows the information gathered by the tech-support tool for the following configuration file:

```
COMMON_SETUP:
        HOSTS_ALL:                 # All compute, control and storage hosts
                SERVER_FILES:
                        - /usr/lib/docker-storage-setup
                SERVER_COMMANDS:
                        - docker info
                        - my_script.sh
                CONTAINERS:
                        all_containers:  #execute in all containers (even if they are in down state)
                          CONTAINER_COMMANDS:
                                - docker inspect
                                - docker logs
                                        logstash:
                                          CONTAINER_FILES:
                                                - /var/log/
                                          CONTAINER_COMMANDS:
                                                - ls -l
```

If the installation is ended in at least the *COMMON_SETUP* state, the tech-support tool runs under all *OpenStack* nodes (Compute, Control and Storage) and does the following for the given configuration :

- Gather the contents of */usr/lib/docker-storage-setup* file if exists.
- Run *docker info* command and collect the output.
- Run *my_script.sh* and collect the output. The *my_script.sh* is an example of a bash script which is previously added to the */root/openstack-configs /tech-support/custom-scripts* directory.
- Collect the output of docker inspect and docker logs for all containers.
- Collect the files in */var/log* inside the logstash container (if there is container with that name). This is equivalent to running the following command (where */tmp* indicates a temporary location where the tech-support tool gathers all the information): *docker cp logstash_{tag}:/var/log/ /tmp.*
- Collect the output of the command *docker exec logstash_{{tag}}: ls -l.*

Tech-Support if Servers Are Offline

It is difficult to collect the information from the servers, if one or more cloud nodes are not reachable. In this case, you can connect through the **KVM** consol e into those servers and run the local tech-support tool.

1. To run the local tech-support tool run the following command:

   ```
   /root/tech_support_offline
   ```

2. Cisco VIM tech_support _offline collects the Logs and other troubleshooting output from the server and place it in the location of the other server:

   ```
   /root/tech_support
   ```

> ⚠ After the server is reachable, you can use the Cisco VIM tech-support tool to collect all the files under the */root/tech-support/* directory for debugging any offline issues.

Running Cisco VIM Software Hub Technical Support Tool

The Cisco VIM Software Hub technical support tool uses a configuration file that specifies the information to be collected. The configuration file is located in the following location:

```
/root/cvim_sds-{tag-id}/openstack-configs/tech-support/tech_support_sds.yaml
```

This tool checks the point where the Cisco VIM Software Hub is executed, and collects the output of files or commands indicated by the configuration file. The technical support script is available at the Software Hub node in the following location:

```
/root/cvim_sds-{tag-id}/tech-support/ directory.
```

To run the script, enter the following command:

```
./tech-support/tech_support sds
```

This command execution creates a compressed tar file containing all the collected information and displays the file location.

Disk and OSD Maintenance Tools

Disk and OSD Maintenance Tools

- Manage Physical Drives
- Management Node
- Compute/Control Node
- OSD Maintenance Tool (Storage Node)

Manage Physical Drives

You can use the disk-maintenance tool to check the status of all physical drives that are present in running and operational nodes in the following roles (except for third-party servers):

- Management
- Control (all or specific nodes)
- Compute (all or specific nodes)
- Storage

This provides the information about the present status of the physical drives. The status can be Online, Offline, Rebuilding, or Unconfigured Good or JBOD states, if all disks are ok. If not, the disks that have gone bad are displayed with the slot number and server information, for replacement. When multiple disks are to be replaced, it is recommended to remove or add disk for that node.

- Physically remove and insert a new disk, before attempting to replace.
- For smooth operation, wipe out the disk before attempting replace operation.
- Call Cisco TAC, if you face any issue. Do not reattempt.

Management Node

To check the status of the *Diskmgmt* log in to the management node and run the *ciscovim* command with the *diskmgmt* option. The design of the *diskmgmt* user interface follows a test job create, list, show, and delete the workflow.

1. Run the *Help* command to see all available command line options:

```
# ciscovim help diskmgmt
ciscovim help diskmgmt
usage: ciscovim diskmgmt [--server <node1,node2,...>] [--id <id>]
                         [--locator {on,off}] [--json-display] [-y]
                         <create|delete|list|show> <check-disks|replace-disks>
                         <all|management|control|compute|storage>

HDD maintenance helper

Positional arguments:
  <create|delete|list|show>       The control command to perform
  <check-disks|replace-disks>     The identity of the task/action
  <all|management|control|compute|storage>
                                  The role of the target host(s)

Optional arguments:
  --server <node1,node2,...>      List of specific control/compute host names
                                  within the target role.
  --id <id>                       ID used to identify specific item to
                                  show/delete.
  --locator {on,off}              Turn on|off locator LED for server with bad
                                  disks and for the physical drives.
  --json-display                  Show output will be in JSON format.
  -y, --yes                       Yes option to perform the action
```

2. Run the following *check-disk* operation command to create *check-disks* operation for all control nodes in the pod. The system responds with a message indicating the Time, ID, and creation date/time.

```
# ciscovim diskmgmt create check-disks control
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| action          | check-disks                       |
| command          | create                                                                 |
| created_at | 2018-03-07T21:12:20.684648+00:00        |
| id                  | 0c6d27c8-bdac-493b-817e-1ea8640dae57 |
| locator          | False                                                                  |
| result          |                                                                                 |
| role                 | control                                                                |
| servers          | None                                                                  |
| status          | not_run                                                                    |
| updated_at | None                                                                 |
+------------+--------------------------------------+
```

3. Run the *ciscovim diskmgmt list* command to monitor the currently running task and completed tasks. The list command can filter based on the role. Use *all* command to list all the tests implemented in the database.

```
# ciscovim diskmgmt list check-disks control
+--------------------------------------+-------------+---------+----------+--------------------------+
| ID                                                                  | Action          | Role    |
Status   | Created
+--------------------------------------+-------------+---------+----------+--------------------------+
| 861d4d73-ffee-40bf-9348-13afc697ee3d | check-disks | control | Complete | 2018-03-05 14:44:47+00:00
|
| 0c6d27c8-bdac-493b-817e-1ea8640dae57 | check-disks | control | Running  | 2018-03-07 21:12:20+00:00
|
+--------------------------------------+-------------+---------+----------+--------------------------+
[root@F24-Michigan ~]# ciscovim diskmgmt list check-disks compute
+--------------------------------------+-------------+---------+----------+--------------------------+
| ID                                                                  | Action      | Role    |
Status   | Created
|
+--------------------------------------+-------------+---------+----------+--------------------------+
| 0be7a55a-37fe-43a1-a975-cbf93ac78893 | check-disks | compute | Complete | 2018-03-05 14:45:45+00:00
|
+--------------------------------------+-------------+---------+----------+--------------------------+
[root@F24-Michigan ~]# ciscovim diskmgmt list check-disks all
+--------------------------------------+-------------+---------+----------+--------------------------+
| ID                                                                  | Action      | Role    |
Status   | Created
|
+--------------------------------------+-------------+---------+----------+--------------------------+
| cdfd18c1-6346-47a2-b0f5-661305b5d160 | check-disks | all     | Complete | 2018-03-05 14:43:50+00:00
|
| 861d4d73-ffee-40bf-9348-13afc697ee3d | check-disks | control | Complete | 2018-03-05 14:44:47+00:00
|
| 0be7a55a-37fe-43a1-a975-cbf93ac78893 | check-disks | compute | Complete | 2018-03-05 14:45:45+00:00
|
| 0c6d27c8-bdac-493b-817e-1ea8640dae57 | check-disks | control | Complete | 2018-03-07 21:12:20+00:00
|
+--------------------------------------+-------------+---------+----------+--------------------------+
```

4. Run the following command to show the detailed results of *diskmgmt check-disks* operation:

```
# ciscovim diskmgmt show check-disks control --id 0c6d27c8-bdac-493b-817e-1ea8640dae57
+--------------------------+---------------------+------------------------------+---------
+-------------------------+
| Message                  | Host                | Role                         | Server  |
State                   |
+--------------------------+---------------------+------------------------------+---------
+-------------------------+
| Raid Health Status       | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 |
Optimal                 |
|                          | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 |
Optimal                 |
|                          | f24-michigan-micro-3 | block_storage control compute | 7.7.7.5 |
Optimal                 |
|                          |                      |                               |
|                        |
| VD Health Status         | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 |
Optimal                          |
|                          | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 |
Optimal                 |
|                                     | f24-michigan-micro-3 | block_storage control compute |
7.7.7.5 | Optimal                 |
|                                     |
|                                           |         |                          |
| RAID Level and Type      | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 | Type-
HW; Level - RAID1  |
|                          | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 | Type-
HW; Level - RAID1  |
|                          | f24-michigan-micro-3 | block_storage control compute | 7.7.7.5 | Type-
HW; Level - RAID1  |
|                          |                      |                               |
|                          |
| Number of Physical Disks | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 |
8                       |
|                          | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 |
8                       |
|                          | f24-michigan-micro-3 | block_storage control compute | 7.7.7.5 |
8                       |
| Number of Virtual Disks  | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 |
1                       |
|                          | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 |
1                       |
|                          | f24-michigan-micro-3 | block_storage control compute | 7.7.7.5 |
1                       |
| Boot Drive Disk Media-Type | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 |
HDD                     |
|                          | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 |
HDD                     |
|                          | f24-michigan-micro-3 | block_storage control compute | 7.7.7.5 |
SSD                     |
+--------------------------+---------------------+------------------------------+---------
+-------------------------+
State Keys:
DHS-Dedicated Hot Spare|UGood-Unconfigured Good|GHS-Global Hotspare
UBad-Unconfigured Bad|Onln-Online|Offln-Offline
Rbld-Rebuilding|JBOD-Just a Bunch Of Disks
```

5. Run the following command to delete the *diskmgmt check-disks*:

```
Delete a diskmgmt check-disks result:
```

⚠ Cisco recommends you to delete the tests that are not in use.

Compute/Control Node

All the compute/control nodes must run with hardware RAID and typically have two HDDs or SSDs. If the nodes are running with HDDs/SSDs and have one of the HDD/SSD in failed state, see Compute Node Recovery Scenario to recover the node.

OSD Maintenance Tool (Storage Node)

You can use the OSD maintenance tool to check the status of all OSDs that are present in running and operational block storage nodes. This tool gives you the detailed information about the status of the OSDs whether they are Up or Down, along with the information about which HDD/SSD corresponds to which OSD, including the slot number and server hostname.

If a bad OSD/Journal is found, follow the below guidelines:

- If an OSD is down, check the status of the corresponding disk via the CIMC.
- If the CIMC reports that the physical disk health is good, run the *cluster-recovery* and redo *check-osds* to see if the problem cleared.
- If the CIMC reports that the physical disk is faulty, use only the *replace-osd* command.
- Physically remove the defective disk and insert a new disk before attempting the replace-osd command.
- For smooth operation, after a new replacement disk is inserted into the server. Check the new disk health via CIMC and clear any foreign configuration. Also, wipe out the disk before attempting the *replace-osd* command.
- You can replace only one OSD at a time. Space out each replacement of OSD by 30 minutes for *ceph-mon* to sync/rebalance.
- In the case of a dedicated SSD journal failure, the replace-journal command can be used.
- A dedicated journal failure occurs, when all OSDs associated with that journal show as down.
- Physically remove and insert a new SSD journal disk before attempting the replace-journal command. Check the new disk health via CIMC and clear any foreign configuration.
- Use the *replace-journal* command that deletes and adds back all OSDs associated with the defective journal to the CEPH cluster.
- The replace-journal command is considered to be a disruptive operation. Depending on the size of the cluster and amount of data stored, the time to rebalance the data can take a long time. Therefore, the replace-journal must be done during a maintenance window.
- Call TAC in case of issues. Do not reattempt.

To check the status of the *osdmgmt* tool log in the management node and run the *ciscovim* command with the *osdmgmt* option. The *osdmgmt* user interface allows you to create, list, show, and delete a workflow.

- Use *ciscovim osdmgmt create* command to initiate a check and replace OSD operation
- Use *ciscovim osdmgmt list* command to view summary and status of current OSD operations
- Use *ciscovim osdmgmt show ... --id <ID>* command to view detail OSD operation results
- Use '*ciscovim osdmgmt delete … --id <ID>* command to delete the results.

Examples of usage of this tool:

1. Run the *Help* command to see all the options:

```
# ciscovim help osdmgmt
usage: ciscovim osdmgmt [--server <node1,node2,...>] [--detail] [--id <id>]
                        [--osd <osd_name>] [--locator {on,off}]
                        [--json-display] [--tag TAG] [-y]
                        <create|delete|list|show>
                        <check-osds|replace-osd|replace-journal>

OSD maintenance helper

Positional arguments:
  <create|delete|list|show>       The control command to perform
  <check-osds|replace-osd|replace-journal>
                                  The identity of the task/action

Optional arguments:
  --server <node1,node2,...>      List of specific block_storage hostnames
  --detail                        Display full OSD details
  --id <id>                       ID used to identify specific item to
                                  show/delete.
  --osd <osd_name>                Name of down OSD to replace. Eg. 'osd.xx'
  --locator {on,off}              Turn on|off locator LED for server with bad
                                  OSDs and for the physical drives.
  --json-display                  Show output will be in JSON format.
  --tag TAG                       Text to associate with a particular job
  -y, --yes                       Yes option to perform the action
```

2. To check the *osds* run the following command:

```
# ciscovim osdmgmt create check-osds
+------------+--------------------------------------+
| Field          | Value                                                                                    |
+------------+--------------------------------------+
| action     | check-osds                                                                                |
| command    | create                                                                                       |
| created_at | 2018-03-08T21:11:13.611786+00:00         |
| id         | 5fd4f9b5-786a-4a21-a70f-bffac70a3f3f |
| locator    | False                                                                                      |
| osd        | None                                                                                       |
| result     |                                                                                                    |
| servers    | None                                                                                       |
| status     | not_run                                                                                       |
| updated_at | None                                                                                       |
+------------+--------------------------------------+
```

3. Monitor the *osdmgmt* check operations using *te list* command. Use the *ciscovim* o*sdmgmt list* commands to monitor the currently running test and to view the tests that are run/ completed.

```
# ciscovim osdmgmt list check-osds
+--------------------------------------+------------+----------+--------------------------+
| ID                                   | Action     | Status   | Created                  |
+--------------------------------------+------------+----------+--------------------------+
| 5fd4f9b5-786a-4a21-a70f-bffac70a3f3f | check-osds | Complete | 2018-03-08 21:11:13+00:00 |
| 4efd0be8-a76c-4bc3-89ce-142de458d844 | check-osds | Complete | 2018-03-08 21:31:01+00:00 |
+--------------------------------------+------------+----------+--------------------------+
```

4. To show the detailed results of *osdmgmt check-osds operation*, run the following command:

```
# ciscovim osdmgmt show check-osds --id 5fd4f9b5-786a-4a21-a70f-bffac70a3f3f
+--------------------+---------------------+-----------------------------+---------+---------+
| Message            | Host                | Role                        | Server  | State   |
+--------------------+---------------------+-----------------------------+---------+---------+
| Overall OSD Status | f24-michigan-micro-3 | block_storage control compute | 7.7.7.5 | Optimal |
|                    | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 | Optimal |
|                    | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 | Optimal |
| Number of OSDs     | f24-michigan-micro-3 | block_storage control compute | 7.7.7.5 | 5       |
|                    | f24-michigan-micro-1 | block_storage control compute | 7.7.7.7 | 5       |
|                    | f24-michigan-micro-2 | block_storage control compute | 7.7.7.6 | 5       |
+--------------------+---------------------+-----------------------------+---------+---------+
+----------------------+--------+--------+----+-----------+-----------+--------------------------
+-----------+
| Host                 | OSDs   | Status | ID | HDD Slot  | Path      | Mount                     |
Journal |
+----------------------+--------+--------+----+-----------+-----------+--------------------------
+-----------+
| f24-michigan-micro-3 | osd.0         | up          | 0  | 4 (JBOD)  | /dev/sda1 | /var/lib/ceph/osd
/ceph-0  | /dev/sdf1 |
|                      | osd.1  | up     | 1  | 5 (JBOD)  | /dev/sdb1 | /var/lib/ceph/osd/ceph-1  | /dev
/sdf2 |
|                      | osd.3  | up     | 3  | 7 (JBOD)  | /dev/sdc1 | /var/lib/ceph/osd/ceph-3  | /dev
/sdf3 |
|                      | osd.5  | up     | 5  | 8 (JBOD)  | /dev/sdd1 | /var/lib/ceph/osd/ceph-5  | /dev
/sdf4 |
|                      | osd.6  | up     | 6  | 6 (JBOD)  | /dev/sde1 | /var/lib/ceph/osd/ceph-6  | /dev
/sdf5 |
| f24-michigan-micro-1 | osd.2  | up     | 2  | 5 (JBOD)  | /dev/sda1 | /var/lib/ceph/osd/ceph-2  | /dev
/sdf1 |
|                      | osd.7  | up     | 7  | 7 (JBOD)  | /dev/sdb1 | /var/lib/ceph/osd/ceph-7  | /dev
/sdf2 |
|                      | osd.9  | up     | 9  | 8 (JBOD)  | /dev/sdc1 | /var/lib/ceph/osd/ceph-9  | /dev
/sdf3 |
|                      | osd.11 | up     | 11 | 6 (JBOD)  | /dev/sdd1 | /var/lib/ceph/osd/ceph-11 | /dev
/sdf4 |
|                      | osd.13 | up     | 13 | 4 (JBOD)  | /dev/sde1 | /var/lib/ceph/osd/ceph-13 | /dev
/sdf5 |
| f24-michigan-micro-2 | osd.4  | up          | 4  | 8 (JBOD)  | /dev/sda1 | /var/lib/ceph/osd/ceph-4  |
/dev/sdf1 |
|                      | osd.8  | up     | 8  | 5 (JBOD)  | /dev/sdb1 | /var/lib/ceph/osd/ceph-8  | /dev
/sdf2 |
|                      | osd.10 | up          | 10 | 4 (JBOD)  | /dev/sdc1 | /var/lib/ceph/osd/ceph-10 |
/dev/sdf3 |
|                      | osd.12 | up          | 12 | 6 (JBOD)  | /dev/sdd1 | /var/lib/ceph/osd/ceph-12 |
/dev/sdf4 |
|                      | osd.14 | up          | 14 | 7 (JBOD)  | /dev/sde1 | /var/lib/ceph/osd/ceph-14 |
/dev/sdf5 |
+----------------------+--------+--------+----+-----------+-----------+--------------------------
+-----------+
```

5. To delete the *check-disk osds*, run the following command:

```
# ciscovim osdmgmt delete check-osds --id 5fd4f9b5-786a-4a21-a70f-bffac70a3f3f
Perform the action. Continue (Y/N)Y
Delete of UUID 5fd4f9b5-786a-4a21-a70f-bffac70a3f3f Successful
[root@F24-Michigan ~]# ciscovim osdmgmt list check-osds
+--------------------------------------+------------+----------+--------------------------+
| ID                                   | Action     | Status   | Created                  |
+--------------------------------------+------------+----------+--------------------------+
| 4efd0be8-a76c-4bc3-89ce-142de458d844 | check-osds | Complete | 2018-03-08 21:31:01+00:00 |
+--------------------------------------+------------+----------+--------------------------+
```

Utility Tool

Utility Tool

- Utility to Resolve Cisco VIM Hardware Validation Failures
- Resolving Hardware Failures

Utility to Resolve Cisco VIM Hardware Validation Failures

The Cisco VIM Hardware Validation utility tool enables to perform hardware validation and captures the user and environmental hardware validation errors during installation of UCS C-series servers. The tool enables you to fix these errors that are based on the inputs you provide at the Command Line Interface (CLI). It validates the updated configuration to verify if the changes are applied properly. After the error is resolved, you can resume the installation from the point of failure.

The *ciscovim hardware-mgmt* user interface allows you to test the job validate orresolve-failures(create), list, show, and delete workflow.

Hardware-mgmt user workflow:

1. Use "*ciscovim hardware-mgmt validate …*" command to initiate a validation.
2. Use "*ciscovim hardware-mgmt list …*" command to view summary/status of current test jobs.
3. Use "*ciscovim hardware-mgmt show … --id <ID>*" command to view detail test results.
4. Use "*ciscovim hardware-mgmt delete … --id <ID>*" to delete test results.

A database of results is maintained so that the you can keep the results of *multiple hardware-mgmt* operations and view them at any time.

You cannot use the utility for the following tasks:

- Configuring BIOS settings for Dell or HPE compute nodes.
- Upgrading or changing the firmware version.
- Resolving hardware failures other than lom, hba, pcie_slot, power, and vnic_pxe_boot.

Resolving Hardware Failures

To capture the list of failures that can be resolved by using the utility, go to the *install* directory and execute the *help* command:

```
# cd <installer-id>/clouddeploy
# python hw_validations.py -help .
```

The following shows the output of the *help* command:

```
usage: hw_validations.py [-h] [--resolve-failures RESOLVE_FAILURES]
[--validate VALIDATE_OF] [-y] [--host HOSTS]
[--file SETUP_FILE_LOCATION]
UCS Hardware Validations
optional arguments:
-h, --help show this help message and exit
--resolve-failures RESOLVE_FAILURES, -rf RESOLVE_FAILURES
                                            all - Fix all the failures.
                                            lom - Fix LOM port(s) status failures.
                                            hba - Fix HBA port status failures.
                                            pcie_slot - Fix PCIe slot status failures.
                                            power - Fix Power failures.
                                            vnic_pxe_boot - Fix Vnic PXE_Boot status failures
-y, -yes
--host HOSTS Comma separated list of hostnames
--file SETUP_FILE_LOCATION, -f SETUP_FILE_LOCATION
                        Provide a valid 'setup_data.yaml' file
```

Command Syntax

```
hw_validations.py [-h] [--resolve-failures RESOLVE_FAILURES] [--validate VALIDATE_OF] [-y]
[--host HOSTS] [--file SETUP_FILE_LOCATION]
```

The following table provides the description of the parameters of the command:

Optional	Description
[-h], --help	Provides detailed information about the command.
[--resolve-failures RESOLVE_FAILURES], -rf RESOLVE_FAILURES	Enables you to specify the failure that you want to resolve. The optional arguments are as follows:
[-y]	Yes
[--host HOSTS]	Enables you to specify the hostname of the server for which you want to resolve failures. You cannot specify the IP address or CIMC IP address of servers as arguments. You can specify a list of hostnames as comma-separated arguments. If the -host option is not specified, the failures of all the servers that are specified in the *setup_data.yaml* file are resolved.
[--file SETUP_FILE_LOCATION] [-f SETUP_FILE_LOCATION]	Enables you to specify the name of a *setup_data.yaml* file.

Examples of Command Usage

The following table provides the commonly used commands along with their examples.

Purpose	Syntax	Example
To resolve all failures.	python hw_validations.py --resolve-failures all -y	python hw_validations.py --resolve-failures all -y
To simultaneously resolve one or more failures.	python hw_validations.py --resolve-failures <failure-1>,<failure-2> -y	To resolve the lom and hba status failures: python hw_validations.py --resolve-failures lom,hba -y
To resolve the errors by using the *setup_data.yaml* file.	python hw_validations.py --resolve-failures <failure-1>,<failure-2> -y --file <location-of-yaml file>	To resolve the LOM status failures by using ~- file: python hw_validations.py --resolve-failures lom,hba -y --file ~/save/setup_data.yaml
To resolve failures on a particular server as specified in the *setup_data.yaml* file by using the **--** *host* option.	python hw_validations.py --resolve-failures <failure-1> -y --host <name-of-host-server-1>, <name-of-host-server-2>	To resolve the PCIe slot failures on hiccup-controller-1 server as specified in the setup_data.yaml: python hw_validations.py --resolve-failures pcie_slot -y --host hiccup-controller-1

Cisco VIM Client Debug Option

Cisco VIM Client Debug Option

- Overview
- Examples of Using debug Option to get list of passwords
- Examples of Using debug option to get list of nodes
- Example of Getting Response from REST API using Curl Commands
- Examples of Response of REST APIs

Overview

The *--debug* option enables you to get verbose logging on the ciscovim client console. You can use verbose logging to troubleshoot issues with the ciscovim client.

The *debug* option has the following parts:

- Curl command: It can be executed standalone for debugging. It displays the REST API endpoint and Request Payload.
- Response of REST API.

Examples of Using debug Option to get list of passwords

```
# ciscovim --debug list-password-keys
2018-05-28 22:13:21,945 DEBUG [ciscovimclient.common.httpclient][MainThread] curl -i -X GET
-H 'Content-Type: application/json' -H 'Authorization: ****' -H 'Accept: application/json'
-H 'User-Agent: python-ciscovimclient' --cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/secrets
2018-05-28 22:13:21,972 DEBUG [ciscovimclient.common.httpclient][MainThread]
HTTP/1.1 200 OK
content-length: 1284
x-xss-protection: 1
x-content-type-options: nosniff
strict-transport-security: max-age=31536000
server: WSGIServer/0.1 Python/2.7.5
cache-control: no-cache, no-store, must-revalidate, max-age=0
date: Tue, 29 May 2018 05:13:21 GMT
x-frame-options: SAMEORIGIN
content-type: application/json; charset=UTF-8
{u'HEAT_KEYSTONE_PASSWORD': '****', u'CINDER_KEYSTONE_PASSWORD': '****',
u'METADATA_PROXY_SHARED_SECRET': '****', u'WSREP_PASSWORD': '****', u'ETCD_ROOT_PASSWORD':
'****', u'HEAT_DB_PASSWORD': '****', u'CINDER_DB_PASSWORD': '****', u'KEYSTONE_DB_PASSWORD':
'****', u'NOVA_DB_PASSWORD': '****', u'GLANCE_KEYSTONE_PASSWORD': '****',
u'CLOUDPULSE_KEYSTONE_PASSWORD': '****', u'VPP_ETCD_PASSWORD': '****', u'COBBLER_PASSWORD':
'****', u'DB_ROOT_PASSWORD': '****', u'NEUTRON_KEYSTONE_PASSWORD': '****',
u'HEAT_STACK_DOMAIN_ADMIN_PASSWORD': '****', u'KIBANA_PASSWORD': '****',
u'IRONIC_KEYSTONE_PASSWORD': '****', u'ADMIN_USER_PASSWORD': '****', u'HAPROXY_PASSWORD':
'****', u'NEUTRON_DB_PASSWORD': '****', u'IRONIC_DB_PASSWORD': '****', u'GLANCE_DB_PASSWORD':
'****', u'RABBITMQ_ERLANG_COOKIE': '****', u'NOVA_KEYSTONE_PASSWORD': '****',
u'CPULSE_DB_PASSWORD': '****', u'HORIZON_SECRET_KEY': '****', u'RABBITMQ_PASSWORD': '****'}
+----------------------------------+
| Password Keys                    |
+----------------------------------+
| ADMIN_USER_PASSWORD              |
| CINDER_DB_PASSWORD               |
| CINDER_KEYSTONE_PASSWORD         |
| CLOUDPULSE_KEYSTONE_PASSWORD     |
| COBBLER_PASSWORD                 |
| CPULSE_DB_PASSWORD               |
| DB_ROOT_PASSWORD                 |
| ETCD_ROOT_PASSWORD               |
| GLANCE_DB_PASSWORD               |
| GLANCE_KEYSTONE_PASSWORD         |
| HAPROXY_PASSWORD                 |
| HEAT_DB_PASSWORD                 |
| HEAT_KEYSTONE_PASSWORD           |
| HEAT_STACK_DOMAIN_ADMIN_PASSWORD |
| HORIZON_SECRET_KEY               |
| IRONIC_DB_PASSWORD               |
| IRONIC_KEYSTONE_PASSWORD         |
| KEYSTONE_DB_PASSWORD             |
| KIBANA_PASSWORD                  |
| METADATA_PROXY_SHARED_SECRET     |
| NEUTRON_DB_PASSWORD              |
| NEUTRON_KEYSTONE_PASSWORD        |
| NOVA_DB_PASSWORD                 |
| NOVA_KEYSTONE_PASSWORD           |
| RABBITMQ_ERLANG_COOKIE           |
| RABBITMQ_PASSWORD                |
| VPP_ETCD_PASSWORD                |
| WSREP_PASSWORD                   |
+----------------------------------+
```

Examples of Using debug option to get list of nodes

```
# ciscovim --debug list-nodes
2018-05-28 22:13:31,572 DEBUG [ciscovimclient.common.httpclient][MainThread] curl -i -X GET
-H 'Content-Type: application/json' -H 'Authorization: ****' -H 'Accept: application/json'
-H 'User-Agent: python-ciscovimclient' --cacert /var/www/mercury/mercury-ca.crt
https://172.31.231.17:8445/nodes
2018-05-28 22:13:31,599 DEBUG [ciscovimclient.common.httpclient][MainThread]
HTTP/1.1 200 OK
content-length: 2339
x-xss-protection: 1
x-content-type-options: nosniff
strict-transport-security: max-age=31536000
server: WSGIServer/0.1 Python/2.7.5
cache-control: no-cache, no-store, must-revalidate, max-age=0
date: Tue, 29 May 2018 05:13:31 GMT
x-frame-options: SAMEORIGIN
content-type: application/json; charset=UTF-8
{u'nodes': {u'status': u'Active', u'uuid': u'6b1ea6ee-b15b-41ca-9d79-3bb9ec0002bc',
u'setupdata': u'fe78b5f9-5a46-447c-9317-2bf7362c1e81', u'node_data': {u'rack_info':
{u'rack_id': u'RackD'}, u'cimc_info': {u'cimc_ip': u'172.29.172.81'}, u'management_ip':
u'21.0.0.10'}, u'updated_at': u'2018-05-25T11:14:46+00:00', u'reboot_required': u'No',
u'mtype': u'control', u'install': u'372aa3c1-1ab0-4dd0-a8a8-1853a085133c', u'power_status':
u'PowerOnSuccess', u'install_logs':
u'https://172.31.231.17:8008//edd3975c-8b7c-4d3c-93de-a033ae10a6b6', u'created_at':
u'2018-05-21T13:25:50+00:00', u'name': u'gg34-2'}}
+-----------+--------+---------+---------------+
| Node Name | Status | Type    | Management IP |
+-----------+--------+---------+---------------+
| gg34-1    | Active | control | 21.0.0.12           |
| gg34-2    | Active | control | 21.0.0.10     |
| gg34-3    | Active | control | 21.0.0.11     |
| gg34-4    | Active | compute | 21.0.0.13     |
+-----------+--------+---------+---------------+
```

Example of Getting Response from REST API using Curl Commands

```
Get the REST API Password.
# cat /opt/cisco/ui_config.json
{
"Kibana-Url": "http://172.31.231.17:5601",
"RestAPI-Url": "https://172.31.231.17:8445",
"RestAPI-Username": "admin",
"RestAPI-Password": "********************",
"RestDB-Password": "********************",
"BuildNodeIP": "172.31.231.17"
}
Form the Curl Command.
curl -k -u <RestAPI-Username>:<RestAPI-Password> <RestAPI-Url>/<Endpoint>
E.g. To get Nodes Info of Cloud
curl -k -u admin:**** http://172.31.231.17:5601/v1/nodes
```

Examples of Response of REST APIs

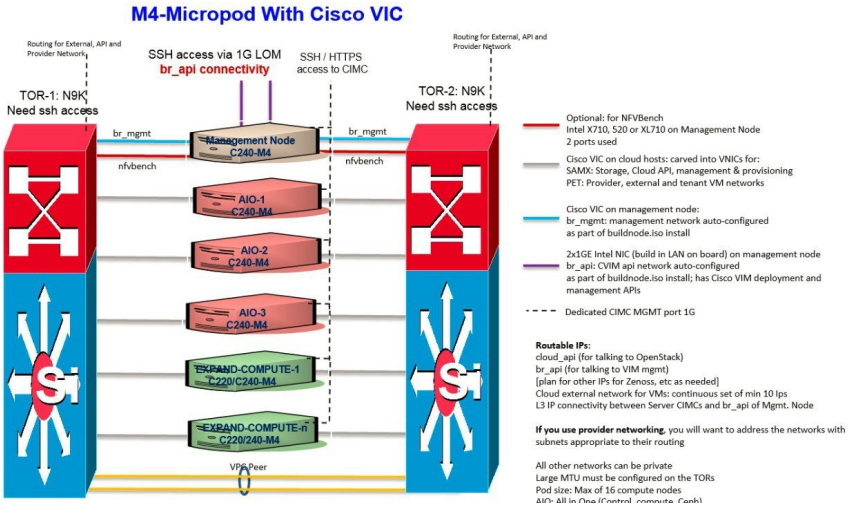
```
API "/"
# curl -k -u admin:**** https://172.31.231.17:8445/
{"default_version": {"id": "v1", "links": [{"href": "http://127.0.0.1:8083/v1/", "rel":
"self"}]}, "versions": [{"id": "v1", "links": [{"href": "http://127.0.0.1:8083/v1/", "rel":
"self"}]}], "name": "Virtualized Infrastructure Manager Rest API", "description":
"Virtualized Infrastructure Manager Rest API is used to invoke installer from API."}
API "/v1/setupdata/"
# curl -k -u admin:**** https://172.31.231.17:8445/v1/setupdata/
{"setupdatas": [. . .]}
API "/v1/nodes"
# curl -k -u admin:**** https://172.31.231.17:8445/v1/nodes
{"nodes": [{"status": "Active", "uuid": "0adabc97-f284-425b-ac63-2d336819fbaf", "setupdata":
"fe78b5f9-5a46-447c-9317-2bf7362c1e81", "node_data": "{\"rack_info\": {\"rack_id\":
\"RackC\"}, \"cimc_info\": {\"cimc_ip\": \"172.29.172.75\"}, \"management_ip\":
\"21.0.0.13\"}", "updated_at": "2018-05-21T15:11:05+00:00", "reboot_required": "No", "mtype":
"compute", "install": "372aa3c1-1ab0-4dd0-a8a8-1853a085133c", "power_status":
"PowerOnSuccess", "install_logs":
"https://172.31.231.17:8008//edd3975c-8b7c-4d3c-93de-a033ae10a6b6", "created_at":
"2018-05-21T13:25:50+00:00", "name": "gg34-4"}, . . . . ]}
API "/v1/secrets"
# curl -k -u admin:**** https://172.31.231.17:8445/v1/secrets
{"HEAT_KEYSTONE_PASSWORD": "5oNff4jWsvAwnWk1", "CINDER_KEYSTONE_PASSWORD": "Hq4i6S5CnfQe7Z2W",
"RABBITMQ_ERLANG_COOKIE": "XRMHBQHTLVJSVWDFKJUX", "METADATA_PROXY_SHARED_SECRET":
"XNzrhosqW4rwiz7c", "WSREP_PASSWORD": "z1oQqhKd1fXDxJTV", "ETCD_ROOT_PASSWORD":
"LMLC8gvi1IA3KiIc", "HEAT_DB_PASSWORD": "J8zt8ldMvdtJxAtG", "CINDER_DB_PASSWORD":
"BVX3y2280DSx2JkY", "KEYSTONE_DB_PASSWORD": "55fVNzxR1VxCNOdh", "NOVA_DB_PASSWORD":
"Rk1MK1OIJgsjGZal", "IRONIC_KEYSTONE_PASSWORD": "9tYZgIw6SZERZ1dZ", "ADMIN_USER_PASSWORD":
"DjDQrk4QT7pgHy94", "GLANCE_KEYSTONE_PASSWORD": "w4REb8uhrHquCfRm", "HAPROXY_PASSWORD":
"oB0v7VJoo2IfB8OW", "CLOUDPULSE_KEYSTONE_PASSWORD": "q6QVvxBQhrqv6Zhx", "NEUTRON_DB_PASSWORD":
"FZVMWgApcZR4us5q", "IRONIC_DB_PASSWORD": "dq3Udmu95DWyX1jy", "GLANCE_DB_PASSWORD":
"O7vQ2emuPDrrvD4x", "KIBANA_PASSWORD": "azHHhP4ewxpZVwcg", "VPP_ETCD_PASSWORD":
"NLyIAvECMW2qI7Bp", "NOVA_KEYSTONE_PASSWORD": "JUfMNGz0BZG7JwXV", "NEUTRON_KEYSTONE_PASSWORD":
"QQ0lo8Q87BjFoAYQ", "CPULSE_DB_PASSWORD": "DaFthNtpX2RvwTSs", "COBBLER_PASSWORD":
"XoIJ9mbWcmVyzvvN", "HORIZON_SECRET_KEY":
"NHkA0qwHIWUSwhPZowJ8Ge3RyRd6oM8XjOT8PHnZdckxgm3kbb1MSltsw0TAQJnx", "DB_ROOT_PASSWORD":
"seqh5DRIKP6ZsKJ8", "HEAT_STACK_DOMAIN_ADMIN_PASSWORD": "Vu6LexEadAxscsvY",
"RABBITMQ_PASSWORD": "LBoYoxuvGsMsl1TX"}
API "/v1/nodes/mgmt._node"
# curl -k -u admin:**** https://172.31.231.17:8445/v1/nodes/mgmt_node
{"api_ip": "172.31.231.17", "mgmt_ip": "21.0.0.2"}
```

Wirring Diagrams

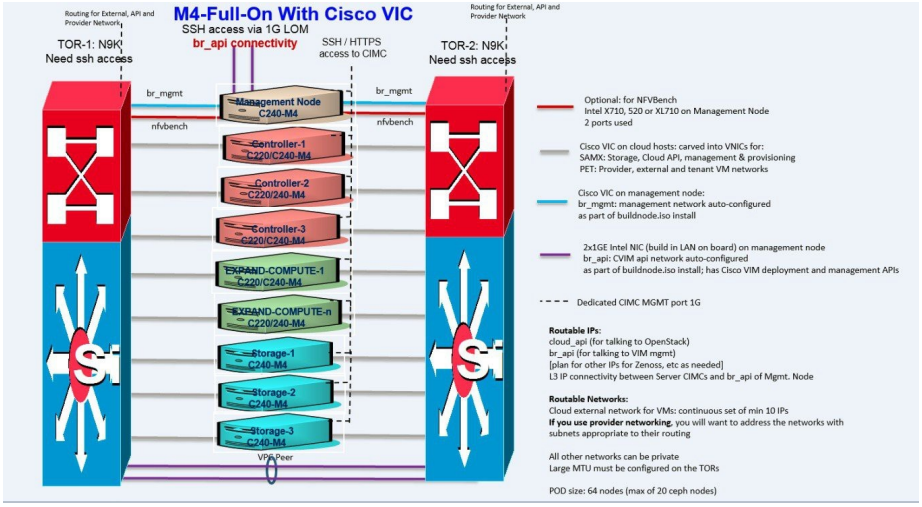
Wiring Diagrams

- M4-Micropod with Cisco VIC
- M4-Full-On with Cisco VIC
- M4/M5 Micropod with Intel NIC (X710) - NIC Redundancy
- M4 Hyperconverged with VPP based Cisco VIC/NIC (1xX710) (no SRIOV)
- M5-Micropod with Intel NIC (X710) - No NIC Redundancy
- M4/M5 Full-On with Intel NIC (X710) and with NIC Redundancy
- M4/M5 Full-On with Cisco VIC/NIC (2xXL710/2x520)
- M4/M5 Micropod with Cisco VIC/NIC (2xXL710/2x520)
- M4/M5-HC with Cisco VIC/NIC (2xXL710/2x520)
- Quanta (D52BQ-2U 3UPI) Fullon Pod with 25GE Intel NIC (xxv710)
- Quanta (D52BE-2U) Edge Pod with 25GE Intel NIC and N3000
- Quanta (D52BQ-2U 3UPI) Ceph Pod with 25GE Intel NIC (xxv710)
- UCS (C240) Edge Pod with 25GE Intel NIC and N3000

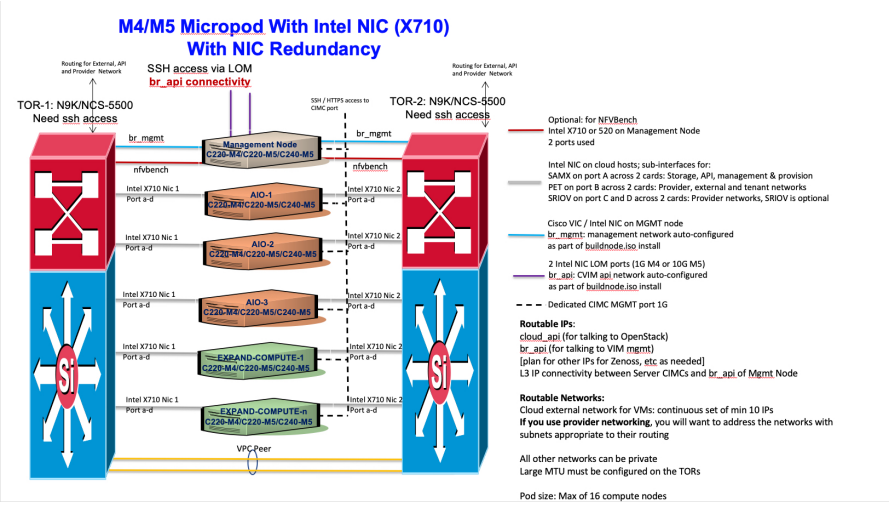
M4-Micropod with Cisco VIC



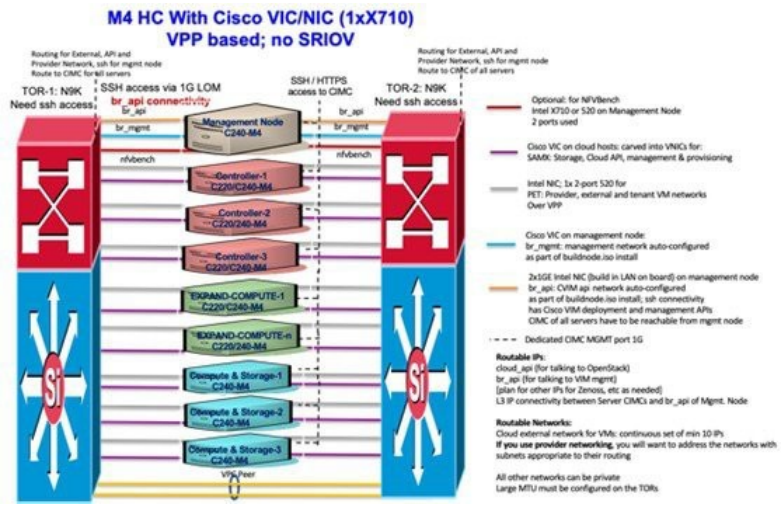
M4-Full-On with Cisco VIC



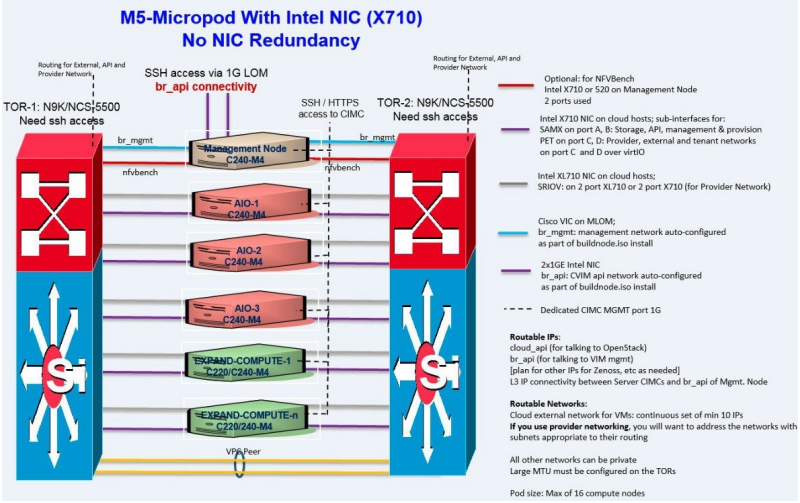
M4/M5 Micropod with Intel NIC (X710) - NIC Redundancy



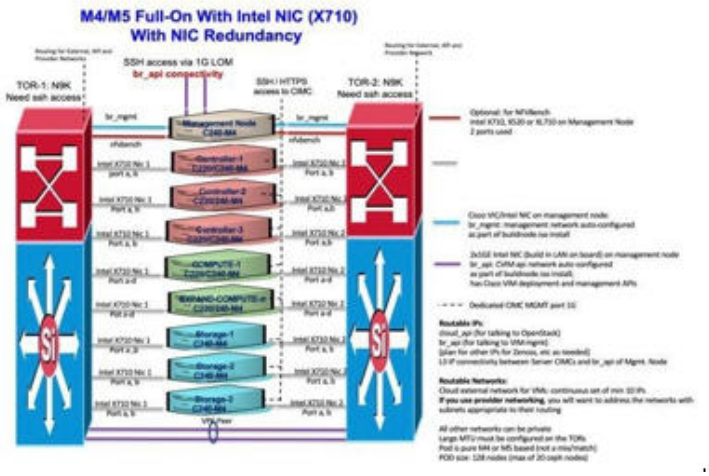
M4 Hyperconverged with VPP based Cisco VIC/NIC (1xX710) (no SRIOV)



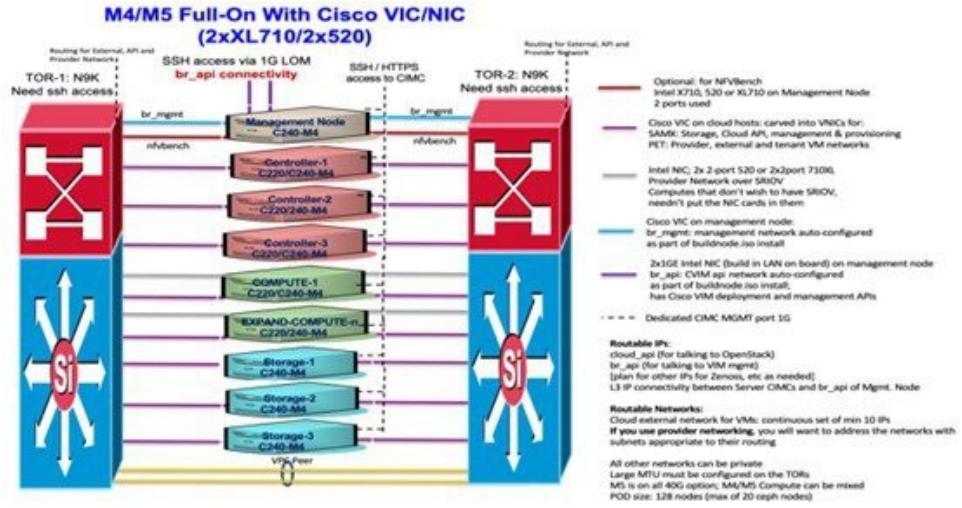
M5-Micropod with Intel NIC (X710) - No NIC Redundancy



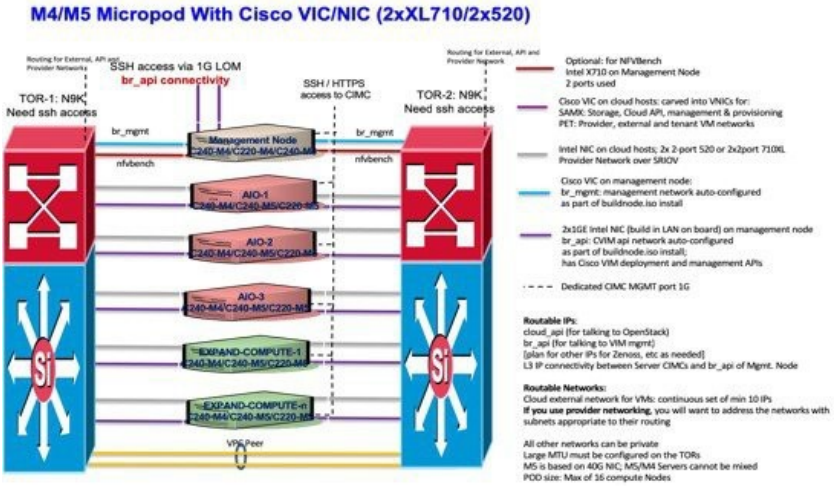
M4/M5 Full-On with Intel NIC (X710) and with NIC Redundancy



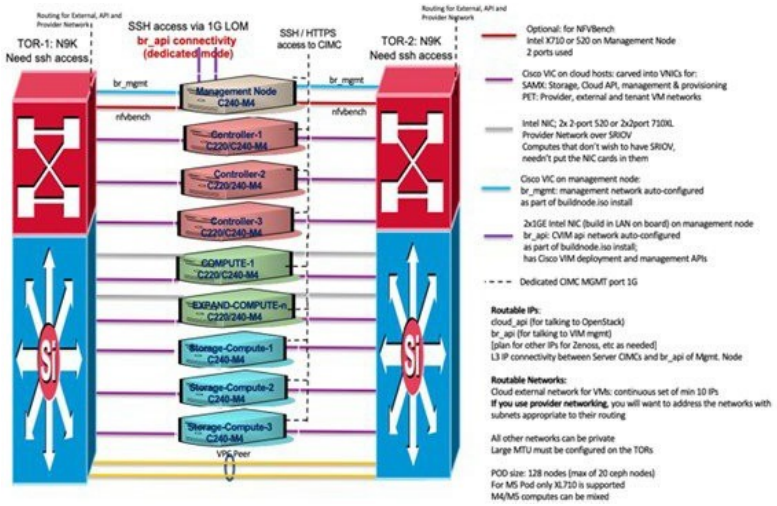
M4/M5 Full-On with Cisco VIC/NIC (2xXL710/2x520)



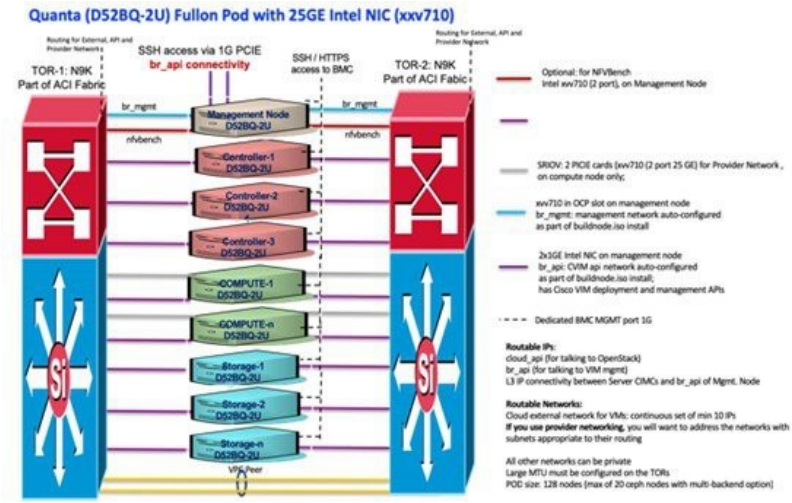
M4/M5 Micropod with Cisco VIC/NIC (2xXL710/2x520)



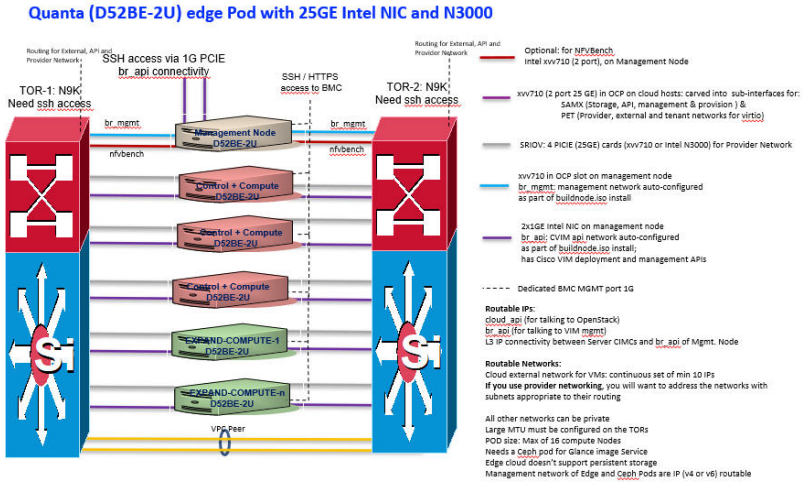
M4/M5-HC with Cisco VIC/NIC (2xXL710/2x520)



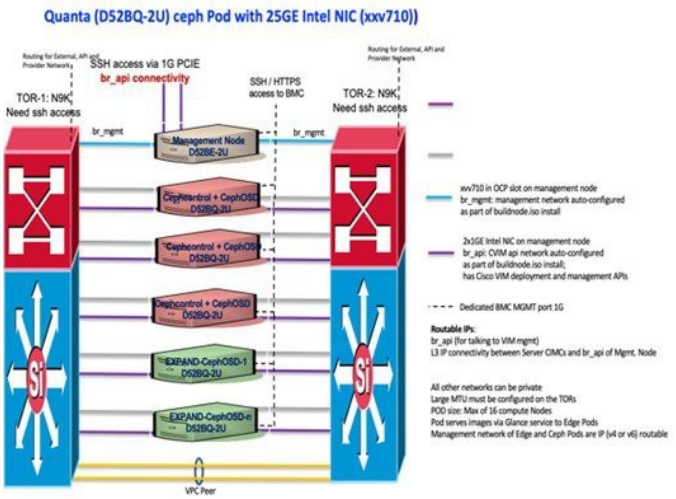
Quanta (D52BQ-2U 3UPI) Fullon Pod with 25GE Intel NIC (xxv710)



Quanta (D52BE-2U) Edge Pod with 25GE Intel NIC and N3000



Quanta (D52BQ-2U 3UPI) Ceph Pod with 25GE Intel NIC (xxv710)



UCS (C240) Edge Pod with 25GE Intel NIC and N3000

