Installing Cisco VIM

The following topics tell you how to configure and install Cisco VIM:

Cisco VIM Installation Overview

Before you can install Cisco Virtual Infrastructure Manager, complete the procedures in *Preparing for Cisco NFVI Installation.* If your management node does not have Internet access, complete the *Preparing to Install Cisco NFVI on Management Nodes Without Internet Access* procedure. The Cisco VIM installation procedure provides two methods for downloading and installing the Cisco VIM installation files, from USB stick prepared for installation, or from the Internet.

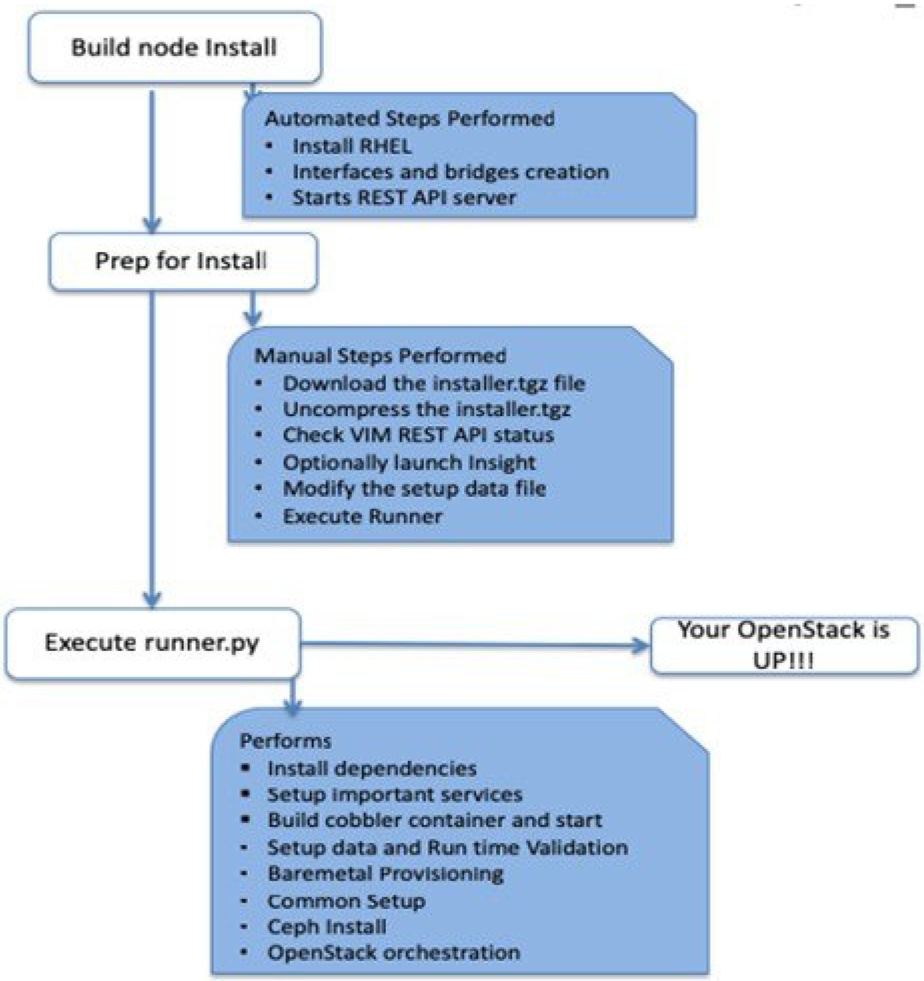
Completing these procedures ensures the Cisco NFVI network infrastructure is set up before the Cisco VIM installation. The bootstrap script is then kicked off, which downloads installer repository, installs Docker and dependencies and starts installer web service,

The Cisco VIM installer can then be launched. It validates the testbed configuration file (setup_data.yaml), creates new vNICs 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. After the installation, the Cisco VIM installer performs common steps across all the Cisco NFVI nodes.

Next, Ceph related packages required for managing the cluster and creating OSD and monitor nodes are installed on the control and storage nodes. By default, the minimum 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 Object Storage Daemon (OSD) to a physical hard drive with a write journal on a separate SSD to support small block random I/O.

The following illustration provides an overview to the Cisco VIM installation.

Figure 1: Cisco VIM Installation Flow


Build node Install
Automated Steps Performed
• Install RHEL
• Interfaces and bridges creation
• Starts REST API server
Prep for Install
Manual Steps Performed
• Download the installer.tgz file
• Uncompress the installer.tgz
• Check VIM REST API status
• Optionally launch Insight
• Modify the setup data file
• Execute Runner
Execute runner.py
Your OpenStack is UP!!!
Performs
• Install dependencies
• Setup important services
• Build cobbler container and start
• Setup data and Run time Validation
• Baremetal Provisioning
• Common Setup
• Ceph Install
• OpenStack orchestration

If you have Cisco Unified Management, complete only part of the Cisco VIM installation procedure and proceed to the Installing Cisco VIM Insight on page procedure followed by Installing Cisco VIM through Cisco VIM Unified Management to complete the configuration and setup of Cisco VIM using the Cisco VIM Insight. If you do not have Cisco VIM UM, configure Cisco VIM by editing the data_setup.yaml as described in the Cisco VIM installation.

Installing Cisco VIM

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

Before you begin

- You need to get Cisco NFVI installation file download site credentials from your Cisco account representative.

- For management nodes with no Internet access, you need a USB stick containing the Cisco NFVI installation files. To prepare the USB stick, see Preparing to Install Cisco NFVI on Management Nodes 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. Cisco recommends 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 the 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 self-signed certificate, see Setting Up Cisco VIM OpenStack Configuration, on page 30

Step 1 If your management node does not have Internet access, use the prepared USB stick and complete the following steps:

a) Insert the USB stick into the management node drive.

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

Step 2 If you are installing Cisco VIM Insight, navigate to Installing Cisco VIM Unified Management and complete the Cisco VIM Insight installation.

If you are not installing Cisco VIM Insight, complete the following steps.

Step 3 Change to the installer directory by running the following command:

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

Step 4 Create a dir (for example, ~/Save/) to contain a copy of the setup_data.yaml file, the file that configures the Cisco NFVI for your particular implementation.

Step 5 Change to the openstack-configs directory and copy the example Cisco VIM setup_data.yaml file into the directory you just created:

```
cd openstack-configs/
cp setup_data.yaml.<C_or_B>_Series_EXAMPLE setup_data.yaml
~/Save/setup_data.yaml
```

Note Only the CPU and MEM allocation ratio needs to be changed for the target pod. Update the following 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

Step 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.

Step 7 If you intend to run the cloud over TLS, see Setting Up Cisco VIM OpenStack Configuration, on page 30 for TLS certificate generation.

Step 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.

Cisco VIM Client Details

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

Note For a complete list of Cisco VIM REST API commands, see the *Cisco NFVI Administrator Guide*.

Before you use the Cisco VIM CLI, check that the API server is up and pointing to the right installer directory. You can execute the following command to validate the state of the API server and the installer directory it is referencing:

```
# 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>/
```

Verify the server status is active and the restapi launch directory is the same the directory from where the installation is launched. If the installer directory, or the REST API state is not correct, go to the target installer directory and execute the following:

```
# 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 tool also provides the options to restart, tear down and reset password for the REST API server as listed:

```
# ./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 running, 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 it is pointing to an incorrect REST API launch directory, go to the installer-<tagid>/tools dir and execute:

```
# ./restapi.py --action setup
```

To confirm that the Rest API server state and launch directory is correct, execute:

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

If you ran the REST API recovery step 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> run --perform 7 -y
```

For an overview to the commands you can execute from the CLI, enter the following command:

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

Command-line interface to the Cisco Virtualized 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
    list-nodes                  List the nodes in the Openstack cloud
    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 configs that can be changed
                                using reconfigure
    list-password-keys          List of password keys that can be changed
                                using reconfigure
    reconfigure                 Reconfigure the Openstack cloud
    cluster-recovery            Recover the Openstack cluster after a network
                                partition or power outage
    mgmtnode-health             Show health of the Management node
    commit                      Commit an update
    rollback                    Rollback an update
    update                      Update the Openstack cloud
    update-status               Status of the update operation
    upgrade                     Upgrade the Openstack cloud
    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
    client-version              Show Virtualized Infrastructure Manager
                                Version
    version                     Show Virtualized Infrastructure Manager
                                Version
    help                        Display help about this program or one of its
```

```
                                subcommands.

Optional arguments:
  --setupfile <setupdata_file>

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 [--join] [--perform <perform>] [--skip <skip>] [-y] Perform a install
operation
Optional arguments:
--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 [root@MercRegTB1 installer]#
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
```

Similarly, you can execute the installation using the skip option, where you explicitly indicate which options to skip. For example

```
$ ciscovim run --skip 1,3 –y
```

Note When using the step-by-step installation, keep a track of what steps are already completed, or unpredictable results might occur.

While the install time varies from pod to pod, typical installation times through the Internet for a UCS C-series with three controller, nine compute, and three storage are listed in the following table.

Table 1:

Operation ID	Operation	Estimated Time
1	Input validation	6 minutes
2	Management node orchestration	40 minutes

Operation ID	Operation	Estimated Time
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

Re-installing Pod with same Image version

In unforeseen circumstances there might be a need to reinstall the pod with the same image version. To alleviate the need of a reimaging of the management node, followed by re-install, you can take the following steps to re-install the pod on assuming that the management node is compatible to the same tag. Ensure that you use the same servers for re-installation. If a different set of servers are used for the re-installation, the servers from the previous install which are no longer participating in the new install must be powered off to avoid the duplicate IP floating in the network.

Listed below are the steps to reinstall the pod without reimaging the management node.

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

```
cd ~/installer-<3.2.0>
./unbootstrap.sh -k
```

Step 2 Verify that no docker containers are running

```
docker ps -a
```

Step 3 Verify that no docker images are present

```
docker images
```

Step 4 Setup RestAPI

```
cd ~/installer-3.2.0/tools
./restapi -a setup
```

Step 5 Regenerate TLS certificate, if needed or TLS is enabled.

```
cd ~/installer-3.2.0
tools/tls_cert_gen.sh -f ~/Save/setup_data.yaml
```

Step 6 Re-run Cisco VIM installation

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

Cisco VIM Configuration Overview

The following topics provide a list of Cisco NFVI configurations you must enter in setup_data.yaml with a yaml editor. These configurations has to be performed prior to running the Cisco VIM installation. If you are installing Cisco Insight, you have to complete the Cisco VIM data and OpenStack configurations using VIM Insight as described in Installing Cisco VIM through Cisco VIM Unified Management .

Configuring ToR Automatically

Cisco VIM provides a complete automation of the cloud deployment. It automates day-0 configuration of N9xxx series Top of Rack (ToR) switches. This feature is optional and applicable only to the Pods that are running with ACI. For ToR switch details related to ACI, see Enabling ACI in Cisco VIM, on page 48.

It automates Power-On Auto Provisioning (post-POAP) configuration on ToR with one or more pair of identical Cisco N9300 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 both B and C-series pods. The automation includes configuration of the edge ports in the leaf switches off which the hosts hang-out and the VPC peer link between the switches.

Auto-configuration feature does not include the configuration of the spine switches and the connectivity between the leaf and the spine; that is 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 in each ToR.

The recommended N9K switch software versions are 7.0(3)I4(6) and 7.0(3)I6(1). Bootstraping 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. You can access each ToR through its management interface from the Cisco VIM management node using SSH.

Setting Up Cisco VIM Data Configuration

You can install and configure the Cisco VIM deployment using the Cisco VIM configuration file (setup_data.yaml). Ensure that you take extreme care while creating the configuration file, as any change in the configuration after deployment, with the exception (example: NFVIMON, of adding and removing nodes and so on) causes a stack redeployment.

Note 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 goes smoothly. Cisco recommends using a YAML editor on Linux (PyCharm, Komodo or vi/vim with YAML plugin) to edit this file. Items shown in brown must be changed to your specific testbed. Do not copy the examples shown below into your YAML file, as your browser might render the characters differently.

If you are using the Cisco VIM installer, you cannot update the OpenStack config files (for example, ml2_conf.ini, and other files) directly. All OpenStack configurations must be in the setup_data.yaml file. This ensures that the installer has a view of the OpenStack deployment, so that it can reliably perform software updates and upgrades. This ensures a consistent and repeatable installation. Key setup file parts are shown in the following sections.

Setting up ToR Configurations for B-series and C-series

The ToR configuration is driven through the mercury setup_data.yaml configuration. The information for automated TOR configuration is provided in two parts in the setup_data.yaml file. The common information is in the TORSWITCHINFO section, whereas the information on individual switch ports connected to specific nodes are under SERVERS section for C-seires, and UCSM-COMMON section for B-series. If the TORSWITCHINFO section is not provided or CONFIGURE_TORS attribute under TORSWITCHINFO then all the ToR provisioning related steps are skipped. The ToR section contains attributes related to ToR connection, configuration for the management interface for the management node, and vPC peer details in case of ToR pairs.

Note 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 for NFVbench
    username: admin  # mandatory for NFVbench
    password: <redacted> # mandatory for NFVbench
    ssh_ip: <a.b.c.d>  # mandatory for NFVbench
    ssn_num: <xyz>
    vpc_peer_keepalive: <f.g.h.i>
    vpc_domain: <int>
    vpc_peer_port_info: <'eth1/45,eth1/46,eth1/47’>
    vpc_peer_vlan_info: <’NNNN,NNNN-NNNN’>
    br_mgmt_port_info: 'eth1/19'
    br_mgmt_po_info: <'NN'>
    -
    hostname: K09-n9k-b # mandatory for NFVbench
    username: admin  # mandatory for NFVbench
    password: <redacted> # mandatory for NFVbench
    ssh_ip: <f.g.h.i>  # mandatory for NFVbench
    ssn_num: < xyz>
    vpc_peer_keepalive: < a.b.c.d>
    vpc_domain: <int>
    vpc_peer_port_info: <'eth1/45,eth1/46,eth1/47’>
    vpc_peer_vlan_info: <’NNNN,NNNN-NNNN’>
    br_mgmt_port_info: 'eth1/19'
    br_mgmt_po_info: <'NN'>
```

The attributes for vpc peer vlan info, vpc domain and br_mgmt_po_info have to match across the ToRs, and should only be defined in only two of the TORs, where the management node is hanging off. The attribute for vpc_peer_vlan_info is optional. If it is not specified, it derives a list of VLAN ids from the host/FI facing interfaces and br_mgmt interface. Also, the attribute for ssn_num which represents the chassis serial number is optional.

The chassis serial number can be obtained by executing the following command on each of the ToRs:

```
show license host-id
```

In the case of B-series, Cisco VIM configures the UCSMCOMMON section to declare the interface configuration under **tor_info_fi** and **tor_info_fi_redundant** for the FI.

Note ToR names need to match with names provided in the TORSWITCHINFO section.

```
UCSMCOMMON:
     ucsm_ip: <p.q.r.s>,
     ucsm_password: <redacted>,
     ucsm_resource_prefix: c43b,
     ucsm_username: admin,
           tor_info_fi: {po: 18,  K09-n9k-a: eth1/17, K09-n9k-b: eth1/17}
           tor_info_fi_redundant: {po: 19,  K09-n9k-a: eth1/19, K09-n9k-b: eth1/19}
```

In this example of B-Series, tor_info is not declared in the SERVERS section as all connectivity is through the FI (controller, compute, and storage) declared in the UCSCOMMON section. VLANs for the FI facing interfaces are derived from the NETWORK segment ROLES for controller, compute, and storage nodes.

The SERVERS section declares the interface configurations for each of the controller, compute, and storage nodes under **tor_info.**

```
SERVERS:
  controller-1:
    rack_info: {rack_id: rack43X}
    cimc_info: {cimc_ip: <ip_addr>}
    tor_info: {po: 5, B9-TOR-9K-1: eth1/5, B9-TOR-9K-2: eth1/5}
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 of the servers and their corresponding network profile roles assigned for each of the segments.

Server Level Setup_data info for C-series with Intel NIC

When the C-series pod is configured to run in a complete Intel NIC environment, the ToR have an additional configuration that is dp_tor_info section. Control plane and data plane traffic are broken out into two separate interfaces with VLAN limiting 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 info for C-series with Intel NIC with SRIOV

When the C-series pod is configured to support SRIOV with Intel NIC, a third interface is configured to allow SRIOV traffic for the compute nodes. Switchports configured for SRIOV are not placed in a port-channel. VLAN limiting is applied to this interface for all the data plane related VLAN IDs.

```
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}
```

Support for Custom Configuration

Custom Configuration is an optional procedure. The setup_data.yaml file has a section called CUSTOM_CONFIG to support custom configuration. Under the CUSTOM_CONFIG section, raw CLI commands can be provided at the global, port channel, and switchport level. CUSTOM_CONFIG is applied at the time of bootstrap and add-interfaces workflow steps.

For example: setup_data.yaml

```
TORSWITCHINFO:
  CONFIGURE_TORS: true
  CUSTOM_CONFIG:
    GLOBAL:
      [<’cli line 1’>,
       <’cli line 2’>,]
    PORTCHANNEL:
      [<’cli line 1’>]
    SWITCHPORT:
      [<’cli line 1’>,
       <’cli line 2’>,]
```

Setting Up ToR Configurations for NCS-5500

Note In Cisco VIM, the following caveats apply to a Cisco VIM deployment with NCS:

- BGP: For a fresh install of Cisco VIM, assure no BGP configuration is present on the NCS, otherwise the peering between the two NCS does not come up properly. Un-configure any existing BGP configuration. If additional BGP complimentary configuration is needed, add it after a successful Cisco VIM install.
- Segment-Routing: The global block of Segment Routing IDs have to be pre-defined by the admin. Make sure that the prefix defined within the setup_data.yaml is within the Segment Routing global block range.
- NCS Interface Naming: There are a set of different Interface naming variations. We support 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, L2VPN sections (on top of the configuration provided by the CVIM automation) causes subsequent Cisco VIM installs to fail.

For a Cisco VIM with NCS-5500 Auto-ToR is a must-have. You can use the Auto-ToR configuration feature to setup NCS-5500. The mercury Cisco VIM setup_data.yaml configuration file is used as an input file for the configuration.

The setup_data.yaml file contains the following three sections:

- **TORSWITCHINFO:** This section provides the general information.
- **SERVERS section for C-series:** This section provides the information on the switch ports that are connected to the specific nodes. When the micro pod is configured to run in a complete Intel NIC environment with NCS-5500 as the ToR, the SERVER level configurations include tor_info (for control plane) and dp_tor_info (data plane) section. 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.
- **MULTI_SEGMENT_ROUTING_INFO:** This section provides the information related to routing.

  NCS-5500 supports a micro-pod with additional computes running on Intel 710 NICs with no SR-IOV with mechanism driver of VPP.

Note The current release supports the use of two NCS-5500 within a single pod.

The following snippet shows an example of the mercury setup_data.yaml configuration file for NCS-5500

```
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> # has to be unique in the ISIS domain
and depends on the
 global segment routing block define 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>,"," seperated,
 max of two entries
     vpc_peer_port_address:  <local address with mask for vpc_peer_port_info>,"," seperated,
 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> has to be unique in the ISIS domain and
 depends on the global segment routing block defined by the 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 maynot
have the entry)

SERVER SECTION FOR C SERIES:
      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, when present
on a given server, the number of SRIOV port need to be 4 and consistent across the servers;
 Also, please set the INTEL_SRIOV_PHYS_PORTS to 4, when using SRIOV with NCS-5500 as ToR.
Please set the value of INTEL_SRIOV_VFS as per the settings of your VNF (see details later
 for the default values, etc)

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_id are mutually exclusive
      ext_bridge_domain: <string> # user pre-provisions physical, bundle interface,
subinterface and external BD" for external uplink and provides
external BD info in the setup_data
```

Customization of Cisco NCS 5500 Configurations 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 to tenant vlans are generated by appending each vlan id from TENANT_VLAN_RANGES to the rt_suffix and rt_prefix as defined in the network segments.

Resulting route-targets (“rt_prefix”, plus “:”, plus “rt_suffix”, plus the VLAN ID) must not exceed the 6 octets as per RFC 4360 for the Extended Communities. The maximum value is 8 octets with first 2 being reserved for type information.

NCS Day-0 Configuration (Prior to starting Cisco VIM install)

The following snippets have to be defined on the NCS before starting Cisco VIM installation:

```
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>
```

Note For SSH to work generate a key using *crypto key generate rsa*.

Pre-requisites for Segment Routing Global Block and ISIS Prefix

The segment routing configuration has to be predefined by the admin.

The following snippet provides an 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
```

Pre-requisites for API and External Network Segments with 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 need to 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 of the NCS-5500 should 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 the deployment of NCS-5500 as TOR, we also support the workloads off the provider network along with the tenant network.

Listed below are some of the assumptions under which this combination works.

- Provider network segment has to be in scope from day-0. Few of the PROVIDER_VLAN_RANGES has to be defined.
- You can always expand the PROVIDER_VLAN_RANGES with additional VLAN range (minimum starting VLAN range is 2)
- 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. They are created manually on the NCS-5500, before the VIM deployment begins; and upstream interfaces are stitched in.

Support and pre-requisites for Provider Network with NCS-Concept

In a deployment of NCS-5500 as TOR, along with the tenant network, we also support provider networks. The following points are key to use provider_networks with a NCS TOR:

- Provider network segment has to be defined on day-0; also, a handful of PROVIDER_VLAN_RANGES has to be defined in the setup_data.yaml.

Note You cannot add it after a Cisco VIM deployment!

- The PROVIDER_VLAN_RANGES can be extended after a Cisco VIM install by running reconfigure with a updated setup_data.yaml (min starting VLAN range is 2, 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 should not exceed 200.
- Bridge domain for provider starts with prefix: provider<VLANId> and are created manually on the NCS-5500 before VIM deployment begins with necessary upstream interfaces configured accordingly.

Pre-requisites for Provider Network with NCS-5500 as TOR

Provider network support requires the following pre-requisites:

Step 1 Define the network and provider vlan ranges sections in setup_data.yaml.

```
NETWORKING:
  - segments: [provider]
    vlan_id: None
PROVIDER_VLAN_RANGES: 127,3406:3409
```

Step 2 Pre-provisioning the NCS with bridge-domains for corresponding VLANs and plumbing the uplink configuration into these bridge-domains.

```
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
```

Note The Cisco VIM Automation will then configure all the host facing subinterfaces for these provider vlans, EVIs and plumb them into each of the pre-provisioned provider bridge-domains.

Note When pre-provisioning bridge-domain, ensure that the BD names follow the naming convention of "provider<vlan-id>".

Intel NIC Support

Cisco VIM supports C-series pod running with either all Intel 710X NICs or Cisco VICs for control and data plane. In the Intel NIC setup, M4 and M5 (Micropod) based pods need to have 2-4 port and 1 or 2 4 port X710 respectively, for control and data plane connectivity. 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, run the following command

```
# 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. See *Figure 7: UCS C-Series Intel NIC Details* in UCS C-Series Network Topologies. To bring in NIC redundancy across the X710s for M5-based Intel NIC systems, define the following global parameter in the setup_data.

```
# NIC_LEVEL_REDUNDANCY: <True or False> # optional and only applies when INTEL_NIC_SUPPORT
 is set to True
```

A C-series pod, running 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 1-32 (32 is maximum number of INTEL_SRIOV_VFS: <integer>.

By default, in the C-series pod running with 4 port Intel 710 card, 1 port (port #c) from each of the Intel NICs are used for SRIOV. However, some VNFs needs additional SRIOV ports to function. To meet the requirement, an additional variable has been introduced in the setup_data.yaml file by which you can 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 if nothing is defined a value of 2 is used. The only values the parameter takes is 2 or 4. For NCS-5500, the only value supported for INTEL_SRIOV_PHYS_PORTS is 4, and has to be defined for SRIOV support on NCS-5500. As the M5 Micropod environment is based on X710 for control and data plane and an additional XL710 or 2 port X710 for SRIOV only INTEL_SRIOV_PHYS_PORTS of 2 is supported.

SRIOV support on a Cisco VIC POD

Cisco VIM supports M4 based C-series pod running with one 2-port Cisco VIC for control plane and two 2-port Intel 520s or two 2-port XL710 for SRIOV (called VIC/NIC deployment). We also support M5 based C-series pod running with one 2-port 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, run the following command:

```
# CISCO_VIC_INTEL_SRIOV: <True or False>
```

A C-series M4 pod, running Cisco VIC/Intel NIC (2x520 or 2xXL710), also supports SRIOV on the Intel NIC. To enable,SRIOV define a value in the range 1-63 (63 is maximum) (for X520) or 1-32 (32 is maximum for XL710) number of INTEL_SRIOV_VFS: <integer>

By default in the C-series M4 pod running with Cisco VIC and Intel 520/XL710, the control plane runs on the Cisco VIC ports, and all the 4 ports from the 2 Intel 520 NICs or 2 intel XL710 are used for SRIOV.

In C-Series M5 pods running with Cisco VIC and Intel XL710, the control plane runs on the Cisco VIC ports and all the 4 or 8 ports from the 2 intel XL710 are used for SRIOV.

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 to setup the card type, in SRIOV.

```
#SRIOV_CARD_TYPE: <X520 or XL710># for M4 based computes
# SRIOV_CARD_TYPE: <XXV710 or XL710> # for M5 based computes
```

Compute supports different types of the card. If SRIOV_CARD_TYPE is not provided, Cisco VIM chooses the first 2 slots from all SRIOV compute nodes. If SRIOV_CARD_TYPE is provided, Cisco VIM chooses the first 2 slots matching the target card type from each of the SRIOV compute nodes, so that a match between intent and reality exist.

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
```

Note From release Cisco VIM 2.4.4 onwards, some computes have XL710 while others have X520 for SRIOV in an 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 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
```

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 setup_data:

```
SERVER_COMMON:
  # 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.
```

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 setup_data 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 VTS 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 Third-party Compute in Hybrid Mode (HP DL360 Gen9)

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 for 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 the changes done to the existing orchestration workflow and Insight UI, you can reuse the existing Cisco VIC+NIC combo deployment scenario. This minimizes the changes needed for the hardware topology and the "setup_data.yaml" configuration file. For NIC settings that need to be passed to enable HPE ProLiant DL360 Gen9 third-party compute, see "Intel NIC Support for SRIOV only".

In case of Quanta servers, the support of third-party has been extended to all nodes (servers in control, compute, storage and management role.

The following table shows the port type mapping between Cisco UCS C-series, HPE ProLiant DL360, and Quanta computes:

Port Type	Cisco UCS c220/c240Compute	HPE ProLiant DL360 Gen9 Compute	Quanta Server
Control and Data Plane	M4: MLOM - VIC 1227 M5: MLOM - VIC 1387 M5 MLOM – VIC 1457	FlexLOM - HP Ethernet 10Gb 2-port 560FLR-SFP+ Adapter	OCP 25G 2 port xxv710 based card
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	PCIe - HP Ethernet 10Gb 2-port 560SFP+ Adapter	PCIe - Intel xxv710 DA2 25 Gbps 2 port NIC
SRIOV	PCIe - Intel X520-DA2 10 Gbps or Intel XL710 DA2 40 Gbps 2 port NIC or Intel XXV710 25Gbps 2 port NIC		

As this deployment do not support Auto-ToR configuration, the TOR switch needs to 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 top-of-rack switches and the switch ports' configuration have been completed, enable/add the following additional variables in the VIM's "setup_data.yaml" configuration file:

```
CISCO_VIC_INTEL_SRIOV: True
INTEL_SRIOV_VFS: 63
```

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">
```

UCSM Common Access Information for B-series POD

```
UCSMCOMMON:
ucsm_username: "admin"
ucsm_password: <"password">
ucsm_ip: <"a.b.c.d">
ucsm_resource_prefix: <"skull"> # max of 6 chars
ENABLE_UCSM_PLUGIN: <True> #optional; if True, Cisco-UCSM is used, if not defined, default
 is False
MRAID_CARD: <True or False>
```

Note In Cisco VIM 3.x, UCSM plugin support is not enabled.

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: <Please 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/oLKs6n3wqKnn
7rQuLGEZDvXnyLbqMoxHdc4PDFWiGXdlg5DIVGigO9KUncPK 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 white list admin's source IP
(v4 or v6)
    - 10.0.0.0/8
    - 172.16.0.0/12
    - <"2001:xxxx::/64">
```

Note 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, the optional **admin_source_networks** parameter is provided. When specified, access to administrator services is only allowed from the IP addresses specified on this list. Use this setting with care, since a misconfiguration can lock out an administrator from accessing the management node through the network. Recovery can be made by logging in through the console and reconfiguring this setting.

Define Network Segments

```
networks:
- # CIMC network section is applicable only for B-series vlan_id: <int> # between 1 and
4096
subnet: <cidr with mask> # true routable network, e.g. 10.30.115.192/28
gateway: <ip address>
pool:
```

```
- ip_address_1 to ip_address_2 in the current network segment
segments:
- cimc
-
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 range for B-series
-
vlan_id: "<None or 3200-3210>" segments:
- provider
```

Note For PODTYPE: ceph, the storage segment needs to be replaced with segment named “cluster”. Also, for central ceph pod, the only other segment allowed is management/provision.

Define Server Roles

In the Roles section, add the hostname 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 HC (Hyperconverged deployment), all storage nodes acts as compute nodes, but not vice-versa.

In the case of edge pod (to support low latency workloads without persistent storage), specify the same server names under control (total of 3), and compute role (there is no server with storage role). 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 pod is 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 3), 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
object_storage:
networker:

ROLES:   -> for PODTYPE: UMHC
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_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
   object_storage:
```

```
 networker:

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)
object_storage: networker:

# Server common
# Provide the username (default: root)
SERVER_COMMON:
   server_username: root

# Allow static override value for platform vendor instead of dynamic
# discovery at runtime, optional value.
#
# Allowed values
# CSCO - Cisco Systems Inc
# HPE  - Hewlett Packard Enterprise
# QCT  - Quanta Cloud 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)

#  vendor: <CSCO or QCT>    <= Global level override, all servers
#  control:
#    hardware_info:
#      vendor: <CSCO or QCT> <= Role level override, all controls
#      root_drive_type: <HDD, SSD, or M.2_SATA>
#  compute:
#    hardware_info:
#      vendor: <CSCO, HPE, or QCT> <= Role level override, all computes
#      root_drive_type: <HDD, SSD, or M.2_SATA>
#  block_storage:
#    hardware_info:
#      vendor: <CSCO or QCT> <= Role level override, all storages
```

Note The maximum length of non-FQDN hostname is 32 characters. The 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.

Cisco VIM introduces a new topology type called Micropod to address solutions that have requirements of high availability, but 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 (mem, 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 2-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 workloards, for example, vRAN that does not need persistent storage. To enable such deployment, update the setup_data as follows:

```
PODTYPE: edge
```

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
```

Define Servers - Rack (C-Series, Quanta) Pod Example

Note The maximum host name length is 32 characters.

```
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, e.g. slot 7
#management_ip: <static_ip from management pool> #optional, if defined for one server, has
 to 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, but if defined for one server,
then it must be defined for all, also if defined management_ip has to be defined

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 the user wants a specific VNIC to be
chosen

Your_Storage_or_Compute-1_HostName:
cimc_info: {'cimc_ip': ‘<v4 or v6>}
rack_info: {'rack_id': 'RackA'}
hardware_info: {'VIC_slot': '3'} # optional only if the user wants a specific VNIC 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
rx_tx_queue_size: <512 or 1024> # optional, only for compute nodes

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_Slot is optional,
 defined for location of Cisco VIC,
VIC_port_channel_enable is optional and applicable to 1457 based VIC where one wants to 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 ofroot_drive_type can be skipped;
However, the defining the option of M.2 is mandatory on a per server basis if it is used
as a booting media for a
given server. Please note the option of M.2 is only valid for UCS M5 systems. NUM_GPU_CARDS
 is optional and can vary on a per server basis.

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 the pod is multi-backend ceph,
 and a minimum of three storage servers should be available for each backend type.
```

Note SRIOV_CARD_TYPE option is valid only when CISCO_VIC_INTEL_SRIOV is True; and can be defined at per compute level in 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 nor at the global level, the default of X520 is set. The compute can be standalone or hyper-converged node.

Note Cisco VIM installation requires that controller node Rack IDs be unique. The intent it to indicates the physical rack location so that physical redundancy is provided within the controllers. If controller nodes are installed all in the same rack, you must assign a unique rack ID to prepare for future Cisco NFVI releases that include rack redundancy. However, compute and storage nodes does not have rack ID restrictions.

Note For Central Ceph cluster, swap the “storage_ip” with “cluster_ip”.

Define Servers - B-Series Pod Example

Note For UCS B-Series servers, the maximum host name length is 16 characters.

```
SERVERS:
Your_Controller_Server-1_HostName:
rack_info: {'rack_id': 'rack2'}
ucsm_info: {'server_type': 'blade',
'chassis_id': 1,
'blade_id' : 1}
Your_Controller_Server-2_HostName:
rack_info: {'rack_id': 'rack3'}
ucsm_info: {'server_type': 'blade',
'chassis_id': 2,
'blade_id' : 1}
Your_Controller_Server-3_HostName:
rack_info: {'rack_id': 'rack4'}
ucsm_info: {'server_type': 'blade',
'chassis_id': 2,
'blade_id' : 4}
#management_ip: <static_ip from management pool> #optional, if defined for one server, it
must be defined for all nodes
#storage_ip: <Fixed ip from the storage pool> # optional, but if defined for one server,
then it must be defined for all,
also if defined management_ip has to be defined
Your_Compute-1_HostName:
rack_info: {'rack_id': 'rack2'}
ucsm_info: {'server_type': 'blade',
'chassis_id': 2,
'blade_id' : 2}
.. add more computes as needed

Your_Storage-1_HostName:
rack_info: {'rack_id': 'rack2'}
ucsm_info: {'server_type': 'rack',
'rack-unit_id': 1}
Your_Storage-2_HostName:
rack_info: {'rack_id': 'rack3'}
ucsm_info: {'server_type': 'rack',
'rack-unit_id': 2}
Your_Storage-3_HostName:
rack_info: {'rack_id': 'rack4'}
ucsm_info: {'server_type': 'rack',
'rack-unit_id': 3}

# max # of chassis id: 24
# max # of blade id: 8
#max # of rack-unit_id: 96
```

Note 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.

Enabling P-GPU

From CVIM 3.4.1, 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, CVIM 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 CVIM hardware validation will fail.

To use GPU in a given VM:

Before you begin

- 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 that has Tesla T4 cards, you must include the following information about 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.
```

Step 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.

Step 2 Modify the openstack flavor to add gpu alias:

Example:

```
openstack flavor set gpuflavor —property “pci_passthrough:alias”=“t4gpu:x”
```

Where 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.

Step 3 Hide the KVM on the guest VM to bypass the Nvidia driver limitation with the KVM hypervisor.

Example:

```
openstack image set gpuimage —property img_hide_hypervisor_id=true
```

Step 4 Launch a VM with the above flavor to see the card passthrough.

What to do next

Verify the card passthrough by using the **nvidia-smi** command inside the guest VM.

Setting Up Cisco VIM OpenStack Configuration

The following sections provide examples of Cisco VIM OpenStack configuration in the setup_data.yaml file.

OpenStack Admin Credentials

```
ADMIN_USER: <admin>
ADMIN_TENANT_NAME: <admin tenant>
```

OpenStack HAProxy and Virtual Router Redundancy Protocol Configuration

```
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 #eg: 49 (range of 1-255)
internal_lb_vip_address: <Internal IP address on mgmt network>
internal_lb_vip_ipv6_address: <Internal IPv6 address on mgmt 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. You must configure the services to ensure the name and address match. 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 the resolution can be made from any desired host outside the pod.

```
external_lb_vip_fqdn: host or DNS name matching external_lb_vip_address
```

OpenStack TLS and HTTPS Configuration

Enabling TLS is important to ensure 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 certificate authority that signed the server side.

For production clouds, these certificates are provided by a trusted third-party CA according to your company IT policy. For test or evaluation clouds, self-signed certificates can be used quickly enable TLS. For convenience, the installer includes a script that creates and install self-signed certificates

Note Do not use the certificates generated by this tool for production. They are for test purposes only.

To use this tool, make the following changes to the setup data file, then 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 `#./tools/tls_cert_gen.sh -f openstack-configs/setup_data.yaml`.

OpenStack Glance Configuration with Dedicated Ceph/Netapp

For OpenStack Glance (OpenStack image service), the dedicated Ceph object storage configuration is show below. Do not change it. The Ceph and Glance keys are generated during the Ceph installation step, so you do not need to specify the keys in setup_data.yaml file.

```
STORE_BACKEND: ceph/netapp #supported as ‘ceph’ for ceph backend store;and netapp for netapp
 backend
```

CPU Allocation for Controllers in Micro and Edge Pods

As the controller node is shared with other types of nodes in micropod (storage and compute) or edge pod (compute), there are deployments where the number of CPU cores allocated to the controller must be higher than the default value of two. From Cisco VIM 3.4.1, NR_RESERVED_ HOST_PCORES option is available on fresh installation only, for micropod and edge pods. The range of the value is from 2 to 12.

You can configure this option by updating the `setup_data.yaml` file as given below:

```
# Number of cores associated to controllers in a micro or edge deployment, # default value
 if not defined is 2
#NR_RESERVED_HOST_PCORES: <2 - 12>
```

CPU Allocation for Ceph in Hyper-converged or Micropod systems

As the storage node is shared with other node types (e.g. 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 2. From release Cisco VIM 2.4.2 onwards, the option CEPH_OSD_RESEREVED_PCORES is available only on fresh installation for Micropod and hyper-converged pods.

This option is set using the following commands in setup_data, where the value can range between 2 and 12.

```
# 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 - 12>
```

CEPH Placement Group Info (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, the code defaults to 60% cinder and 40% glance for NOVA_BOOT_FROM local. Similarly, if no information is provided the code defaults to 40% cinder, 30% glance and 30% nova ephemeral for NOVA_BOOT_FROM ceph. You cannnot change these values after deployment via 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 (OpenStack storage service), the dedicated Ceph object storage configuration is shown below. Do not change it. The Ceph and Cinder keys are generated during the Ceph installation step, so you do not need to 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.

```
VOLUME_DRIVER: ceph/netapp
```

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 Settings on PODTYPE: Edge for Glance Image service

For the edge pod installation to be successful, the central Ceph cluster with which it will communicate 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

OpenStack Neutron configuration is shown below.

```
# 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: 3001:3100,3350:3400
# 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.

Note 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.

The JUMBO_MTU functionality is available only for OVS over VLAN in a UCS B-Series pod. 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.

Control and Data Plane Testing in Cisco VIM

Cisco VIM offers an integrated test to validate the control and data plane sanity of the cloud. Virtual Machine Through Put (VMTP), an optional test is 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/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.)

To enable VMTP for basic Cisco VIM installation, update the setup_data with the following commands:

```
VMTP_VALIDATION:
  EXT_NET:     # Only for V4 with External network with floating IP, min of 5 cont. IP
    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
```

Optional Services in Cisco VIM

Cisco VIM supports the installation of optional services, namely, ceilometer, ironic, and load balance as a service (lbass). 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. The optional services support provides an infrastructure to support additional services in the future.

Note Auto-scaling is not supported in Cisco VIM.

Ceilometer Support in Cisco VIM

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 with the following:

```
# Optional Services:
OPTIONAL_SERVICE_LIST:
    - ceilometer
```

Note Ceilometer is enabled when the pod type is fullon.

LBASS Support

The reference implementation of LBASS is available from Cisco VIM 3.2.2 onwards. The **lbass** service can be brought in as a day-0 option. To enable this service, update the setup_data with the following:

```
# Optional Services:
OPTIONAL_SERVICE_LIST:
- lbass
```

TAAS Support

Cisco VIM 3.4.1 supports Tap As a Server (TAAS) with Open vSwitch (OVS) as the mechanism driver. You can enable the TAAS service in CVIM as a day 0 or day 2 option. To enable this service, update the setup_data.yaml file with the following information:

```
# Optional Services:
OPTIONAL_SERVICE_LIST:
- taas
```

There are additional configurations that need to be set at the infrastructure level to enable TAAS. The gateway information of the tenant network should be set as the virtual IP in the HSRP configuration on the ToR.

Setting up Remote TAAS OVS

You must perform the following steps to setup a remote TAAS OVS, where traffic is mirrored from the pod VM to a destination outside the pod:

1. Create a GRE network and subnet in OpenStack with the destination address of the mirrored traffic. The network provider segment (GRE ID) must be more than 5000.

   ```
   # openstack network create --provider-network-type gre --provider-segment 5001
   mirror_gre_network
   # openstack subnet create --network mirror_gre_network  --subnet-range 14.14.14.0/24
   mirror_gre_subnet
   ```

2. Create a port for the GRE network with the IP address of the destination of the mirrored traffic. The configured IP address must be reachable from the TORs default gateway address of the tenant network.

   ```
   # openstack port create --network mirror_gre_network --fixed-ip subnet=mirror_gre_subnet,
   ip-address=14.14.14.14 mirror_gre__remote_port
   ```

3. Check the route towards the GRE termination point on both TORs:

   ```
   # sh ip route 14.14.14.14
   IP Route Table for VRF "default"
   '*' denotes best ucast next-hop
   '**' denotes best mcast next-hop
   '[x/y]' denotes [preference/metric]
   '%<string>' in via output denotes VRF <string>
   14.14.14.0/24, ubest/mbest: 1/0
   *via 11.11.11.6, [1/0], 1d01h, static
   ```

4. Check ping towards GRE termination point on both TORs:

   ```
   # ping 14.14.14.14
   PING 14.14.14.14 (14.14.14.14): 56 data bytes
   64 bytes from 14.14.14.14: icmp_seq=0 ttl=63 time=0.7 ms
   ```

5. Ensure that the source VM and its ports are active:

   ```
   # openstack server list -f value c33c38b6-48da-42eb-aeb9-9fac58f0a4c0
   server_dst ACTIVE network_destination=20.20.20.7 RHEL-guest-image m1.small
   # openstack port list -f value 0b7c731c-9193-4c58-aadf-102cc5b67cd6 fa:16:3e:9e:6d:d9
   ip_address='10.10.10.4', subnet_id='fca8955f-7de4-46d7-8cc2-89e1cafb530b' ACTIVE
   ```

6. Create the tap-service using the **neutron tap-service-create --port remote_gre_port_id** command.

   ```
   neutron tap-service-create --port c0f4d618-6f09-40aa-9667-0eeeea1a710d --name
   remote_destination
   ```

7. Create tap-flow using the **neutron tap-flow-create --port mirrored_VM_port_id --tap-service tap_service_id --direction both** command.

   ```
   neutron tap-flow-create --port 0b7c731c-9193-4c58-aadf-102cc5b67cd6
   --tap-service 7e0f7976-b31c-459e-a2b8-2bc698961eb2 --direction both --name
   source1_to_remote_destination1
   ```

8. Check ping to or from the source VM.

Note VM should be up and running before traffic mirroring starts.

On mirrored VM, use the **ping** command:

```
ping 192.168.1.1 -I ethX
```

9. Start traffic capture on the mirroring destination point and check if the mirrored traffic has been received.

   On remote gre termination point:

```
tcpdump -enni eth0 proto 47
```

Setting up Local TaaS OVS

You must perform the following steps to setup a local TAAS OVS, where traffic is mirrored from one VM to another VM inside the pod:

1. Ensure that the mirroring traffic source and destination VMs and their ports are active.

```
# openstack server list -f value c33c38b6-48da-42eb-aeb9-9fac58f0a4c0 server_dst ACTIVE

network_destination=20.20.20.7 RHEL-guest-image m1.small
491d11b3-a204-4b90-b843-87d76c33656f server_src ACTIVE network_source=10.10.10.4
RHEL-guest-image m1.small
# openstack port list -f value 0b7c731c-9193-4c58-aadf-102cc5b67cd6 fa:16:3e:9e:6d:d9
ip_address='10.10.10.4',
subnet_id='fca8955f-7de4-46d7-8cc2-89e1cafb530b' ACTIVE
f46be971-3ac3-49f0-bcbf-5018c74280f2 fa:16:3e:43:46:da
ip_address='20.20.20.7', subnet_id='a5044731-b9b5-44c8-8f6f-ae4937e49d52' ACTIVE
```

Note Mirroring traffic source and destination VMs can belong to different networks.

2. Create a tap-service using the **neutron tap-service-create --port destination_VM_port_id** command.

```
neutron tap-service-create --port c0f4d618-6f09-40aa-9667-0eeeea1a710d --name
remote_destination
```

3. Create a tap-flow using the **neutron tap-flow-create --port source_VM_port_id --tap-service tap_service_id --direction both** command.

```
neutron tap-flow-create --port 0b7c731c-9193-4c58-aadf-102cc5b67cd6
--tap-service 7e0f7976-b31c-459e-a2b8-2bc698961eb2 --direction both --name
source1_to_remote_destination1
```

4. Check ping to or from source VM. On a mirrored VM, use the **ping** command.

```
ping 192.168.1.1 -I ethX
```

5. Start traffic capture on the mirroring destination VM and check if the mirrored traffic received.

Ironic Support in Cisco VIM

The reference implementation of ironic is available in 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 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.

Note It is mandatory to include the ironic management/ironic inspector VLANs on the ToR interfaces connected to all the mercury controller servers. This must be manually configured at present.

While deploying ironic, follow the below steps before installing Cisco VIM:

- 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`
- 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.
- 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.

- 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.
- Verify whether the following 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. If deploying on an Intel NIC or the onboard LOM interface, this step is not needed.
  - Set the VLAN mode on the VNIC being used as TRUNK, if VNICs are used as the interface for ironic. This step is not required for deployment on an Intel NIC or the onboard LOM interface.
  - Turn ON the baremetal node, to have access to 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 doing the following and then reboot 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 with the following:

```
# 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
```

Container Support in Cisco VIM

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.

Assumptions to enable Cisco Container Platform on Cisco VIM:

- Cisco VIM is up and running on the pod.
- Cisco Container Platform is deployed during day-0 or day-2 as part of reconfiguration.
- Cisco Container Platform workload runs as an OpenStack Tenant.
- Cisco Container Platform control plane VMs and all tenant cluster VMs are up and runing.
- A user must pre-exist for the OpenStack tenant where Cisco Container Platform can run.
- The virtual IP (VIP) for the Cisco Container Platform Installer is either a provider or floating IP address in the tenant.
- SSH key in tenant has SSH access to Cisco Container Platform control plane VMs.
- Cisco Container Platform is run on a floating IP address-based or a provider-based environment:
    - If Cisco Container Platform is running on a floating IP address-based environment, managed L3 connectivity is used for networking. A public external network is used to host a minimum of 10 floating IP addresses.
    - If Cisco Container Platform is running on a provider network, managed L3 connectivity is not used for networking. A public provider network is used for connecting to hosts and load balancers. A minimum of 20 IP addresses is required.

Prerequisites for Cisco Container Platform installation:

- Use the installer in the VM of Cisco VIM to create a new OpenStack tenant and to deploy the Cisco Container Platform control plane. You can use the control plane API to create multiple tenant clusters.
- Use Calico network plugin as an overlay.
- For persistent storage, use Cinder as the storage provider. When a Kubernetes-based workload request is received for a persistent volume, dynamic persistent storage is automatically created.
- The external TLS must be enabled in the cloud via external_lb_vip_tls being set to True.
- In a dual stack environment, Cisco Container Platform over IPv6 is only supported.

- For Cisco Container Platform over IPv6, ensure that ENABLE_ESC_PRIV is set to True on Day 0 or via reconfiguration option.
- Ensure that LBaaS is enabled in the optional_service_list. If LBaaS is not enabled, follow these steps:
    - Update the setup file to include LBaaS.

      ```
      OPTIONAL_SERVICE_LIST: [heat, lbaas]
      ```

    - Run the following command:

      ```
      ciscovm reconfigure --setupfile <path to new setupfile containing lbass>;
      ```

Cisco Container Platform Installation:

Follow these steps to install Cisco Container Platform on Cisco VIM:

1. Generate an SSH key of ecdsa type.

   ```
   # ssh-keygen -f /root/ecdsa-key -t ecdsa -b 521
   ```

   Press the enter key to continue until the SSH key generation process is completed.

2. Download the tenant and installer images from the following link:

   https://software.cisco.com/download/specialrelease/1fa104c05177043c5ad141ee0e9157e8

3. Configure the tenant or provider networking type on the basis of the CCP_DEPLOYMENT section defined in setup_data.

   ```
   CCP_DEPLOYMENT: # Parameters for CCP Deployment Optional services LBAAS mandatory
   CCP_CONTROL: # Installer creates a new tenant in Openstack based on below information
   and set all quotas in that tenant
   UI_PASSWORD: <UI_PASSWORD> # password for CCP UI (required)
   ccp_subnet_cidr: <ip_address/mask> # subnet to create to deploy CCP control plane
   (required for tenant network should be removed for provider network)
   installer_subnet_cidr: <ip_address/mask> # subnet to create for bootstrap installer
   (required for tenant network should be removed for provider network)
   installer_subnet_gw: <ip_address> # gateway to use for bootstrap installer (required
   for tenant network should be removed for provider network)
   password: <password> # password for the Openstack tenant (required)
   private_key: <absolute path for ed25519 based key> # private key to be used to SSH
   to VM must be ed25519 (required)
   project_name: <tenant_name> # Tenant name to create for CCP control Plane installer
   will create this Openstack tenant (required)
   public_key: <absolute path for ed25519 based public key> # Public key for CCP VMs,
   e.g. /root/ecdsa-key.pub
   username: <string> # username for the CCP control plane tenant (required)
   CCP_INSTALLER_IMAGE: <qcow2 absolute image path> # Pointer to the CCP Installer image
   (required)
   CCP_TENANT: # Test only option not supported in production to create demo tenant cluster
    using CCP API (Optional NA in production)
   password: <password> # password for tenant (required)
   project_name: <project_name> # tenant name to create in Openstack to host tenant cluster
    (required)
   username: <username> # username for openstack tenant (required)
   workers: 1 # no of kubernetes workers in tenant cluster (required)
   subnet_cidr: <ip_address/mask> # tenant subnet CIDR
   CCP_TENANT_IMAGE: <qcow2 based abs path of tenant cluster image> # Pointer to CCP tenant
    cluster image (required)
   DNS_SERVER: [list of length 1 for DNS servers] # DNS server to be reachable from
   cloud(required)); Based on CCP over IPv4 or IPv6, DNS_SERVER input has to be of the same
    type
   KUBE_VERSION: <x.y.z> # Version of Kubernetes to install (required) normally can be
   deciphered from tenant image name; e.g. 2.3.4
   ```

```
NETWORK_TYPE: <tenant or provider> # Network Type valid values provider or tenant network
 (required)
POD_CIDR: <ip_address/mask> # POD CIDR for calico network, optional; Mandatory for IPv6,
 and is of IPv6 address if CCP needs to run over IPv6.
    For CCP over IPv6, only IPv6 based CCP cluster is allowed; CCP over IPv6 is only
allowed for mechanism driver of openvswitch;
    Also, for CCP over IPv6, DNS_SERVER is of type IPv6 and ENABLE_ESC_PRIV must be true
PUBLIC_NETWORK_UUID: <UUID of Openstack external network or provider network; Fake UUID
 incase of Day-0 Install > (optional initially but mandatory when ccp is being run)
CCP_FLAVOR: <flavor> optional initially, but mandatory when NFV_HOSTS is enabled during
 ccp installation.
```

Note CCP install with NO DHCP on provider network is not supported.

4. To enable Cisco Container Platform on Cisco VIM on day-0, update setup_data to include the CCP_DEPLOYMENT section and execute the standard ciscovim install command. After the installation, update the PUBLIC_NETWORK_UUID from the output of "neutron net-list" after sourcing openrc from `/root/openstack-configs` and CCP_FLAVOR

   ```
   [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 update the PUBLIC_NETWORK_UUID and CCP_FLAVOR
   information
   [root@mgmt1 ~]# cd /root/MyDir/
   [root@mgmt1 ~]# vi setup_data.yaml
   [root@mgmt1 ~]# cd ~/installer-xxxx
   [root@mgmt1 ~]# ccp install --setupfile /root/MyDir/setup_data.yaml
   ```

5. Install Cisco Container Platform:

   ```
   # ciscovim ccp install --setupfile <path_to_setup_file>
   ```

6. Get the URL for the Cisco Container Platform control plane:

   ```
   # ciscovim ccp show
   ```

For more information on installing Cisco Container Platform, see https://www.cisco.com/c/en/us/support/cloud-systems-management/container-platform/products-installation-guides-list.html..

For more information on verification and management of Cisco Container Platform cluster, see *Cisco Virtualized Infrastructure Manager Administrator Guide*

LDAP support in Cisco VIM

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.

With the introduction of Keystone v3, the OpenStack service authentication can now be delegated to an external LDAP server. In Cisco VIM, this feature has been introduced optionally if the authorization is done by Keystone v3.

An important pre-requisite for enabling LDAP integration is that the LDAP endpoint has to be reachable from all the Controller nodes that run OpenStack Keystone Identity Service.

To benefit LDAP support with Keystone v3 feature, the setup_data needs to be augmented with the following information during the installation of the pod.

```
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'
  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.

Note The values for the parameters 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. In order 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 formats

```
url:  '<ldaps | ldap>://<FQDN | IP-Address>:[port]’
```

The protocol can be ldap for non-ssl OR ldaps 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 the non-standard port numbers must be provided.

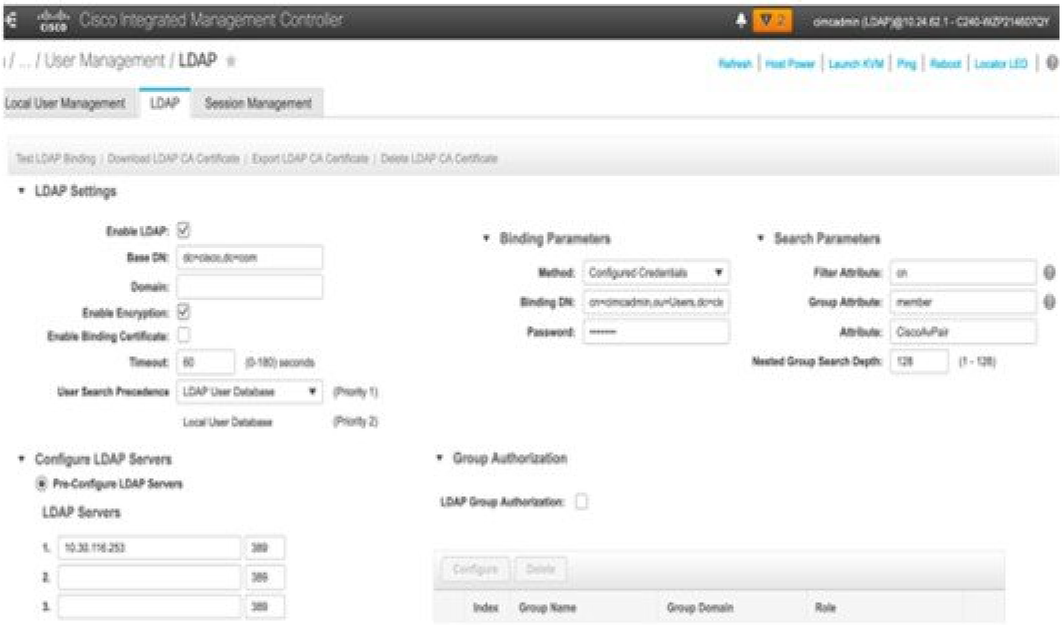
Support for Anonymous LDAP Bind

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.

Note 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. Listed below is a snapshot of the CIMC configuration to authenticate via LDAP.


Cisco Integrated Management Controller
cimcadmin (LDAP)@10.24.62.1 - C240-WZP21460707QY
/ ... / User Management / LDAP
Refresh | Host Power | Launch KVM | Ping | Reboot | Locator LED
Local User Management
LDAP
Session Management
Test LDAP Binding | Download LDAP CA Certificate | Export LDAP CA Certificate | Delete LDAP CA Certificate
Configure
Delete

LDAP Settings

Enable LDAP: ☑
Base DN: dc=cisco,dc=com
Domain:
Enable Encryption: ☑
Enable Binding Certificate: □
Timeout: 60 (0-180) seconds
User Search Precedence: LDAP User Database (Priority 1)
Local User Database (Priority 2)

Binding Parameters

Method: Configured Credentials
Binding DN: cn=cimcadmin,ou=Users,dc=cis
Password: ••••••

Search Parameters

Filter Attribute: cn
Group Attribute: member
Attribute: CiscoAvPair
Nested Group Search Depth: 128 (1 - 128)

Configure LDAP Servers

● Pre-Configure LDAP Servers

LDAP Servers

1. 10.30.116.253 389
2. 389
3. 389

Group Authorization

LDAP Group Authorization: □

Index	Group Name	Group Domain	Role

Red Hat Identity Management System

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 dy 0 or day 1 activity. Multiple IPA servers with IPv4 and IPv6 end points are allowed for redundancy. When IPA is enabled, the features of TTY_LOGGING and vim_ldap_admins cannot be enabled. For more information on IDM, see https://www.redhat.com/archives/rh-community-de-berlin/2012-November/pdfOlwXB8dm7U.pdf.

Assumptions for implementing the IPA solution are listed below:

- 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 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.
- Certificates between the IPA server and hosts over the management network are not included.

- Hostname of all the servers in the pod are in FQDN with only lower cases.

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 can break Cisco VIM operations such as cluster recovery where hostnames defined in the setup_data.yaml file are used. Hence, it is recommended that the hostname of all servers in the pod, including the management node are in FQDN with only lower cases.

To enable IDM, update the `setup_data.yaml` file with the following configuration during installation:

```
IPA_INFO:
  ipa_servers:   # list of ipa_servers
    - hostname: <fqdn hostname in lower case>
      ipaddresses:
        - <ipv4_address or ipv6_address>
    - hostname: <hostname with fqdn in lower case>
      ipaddresses:
        - <ipv4_address>
        - <ipv6_address>
```

Cinder Volume Multi-attach with Cisco VIM

Attaching a volume to multiple hosts or servers simultaneously is used for active/active or active/standby scenarios. For more information about this feature, see the OpenStack documentation at https://docs.openstack.org/cinder/latest/admin/blockstorage-volume-multiattach.html.

This feature is available in Cisco VIM 3.4.1 because of upstream support. Following are the assumptions and caveats associated with this feature:

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, enter the following commands:

1. To attach a volume to multiple hosts or servers, you need to have the multiattach 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>
   ```

   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
   ```

   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:

1. `cinder type-create cephssd`
2. `cinder type-key cephssd set volume_backend_name=ceph-ssd`
3. `cinder type-key cephssd set multiattach="<is> True"`

SolidFire Integration with Cisco VIM

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 to be reachable from Cisco VIM control nodes.

SolidFire is available only as a day-0 configuration. To enable SolidFire, update the setup_data.yam 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 CVIM> # 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 CVIM>
  admin_password: <password for admin user defined above; password criteria is:
        "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."
```

Cisco VIM Configurations for VPP/VLAN Installation

If you are installing Cisco VIM with VPP/VLAN, the mechanism driver in the setup_yaml file should reflect the same.

Cisco VPP/VLAN Mechanism Driver Configuration

```
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; takes values in the range 2
to 4, in order to increase performance by
allocating more cores to VPP
```

Cisco VIM Configuration for Cisco VTS Installation

If you are installing Cisco VIM with Cisco Virtual Topology Systems, you must enter the Cisco VTS parameters in Cisco VIM setup_yaml file.

Cisco VTS Mechanism Driver Configuration

```
MECHANISM_DRIVERS: vts
TENANT_NETWORK_TYPES: "VLAN"
TENANT_VLAN_RANGES: <START>:<END> # arbitrary VLAN range***
ENABLE_JUMBO_FRAMES: True
```

Note VLAN range overlap on the physical network could occur if a hardware VTEP is configured on a top of rack (ToR) switch. (VTEPs are Virtual Extensible Local Area Network (VXLAN) tunnel end points.)

NFV Parameters

```
NFV_HOSTS: ALL
# Only enabled when NFV_HOSTS is set to ALL
#########################################
## Only 2 Values allowed is: 2M or 1G (defaults to 2M)
#VM_HUGEPAGE_SIZE: 2M or 1G

Along with supporting it globally, Cisco VIM also supports VM_HUGEPAGE_SIZE on a per server
 basis with OVS/VTS/VPP as mechanism driver.
SERVERS:
  compute-server-1:
    VM_HUGEPAGE_SIZE: <2M or 1G> # <== optional

## Percentage of huge pages assigned to VM
## On NFV_HOSTS enabled hosts, VM memory can be a mix of regular pages and huge pages. This
 setting sets the ratio. By default, all VM memories (100%)
## has huge pages.
## Only input of type integer is allowed, in the range of 0-100 (including 0 and 100)
# values < 100 is only supported for mechanism driver of openvswitch
#VM_HUGEPAGE_PERCENTAGE: 100

Along with supporting it globally, Cisco VIM also supports VM_HUGEPAGE_PERCENTAGE on a per
 server basis with openvswitch as mechanism driver.

SERVERS:
  compute-server-1:
   VM_HUGEPAGE_PERCENTAGE: <0 to 100> # <== optional, only for mechanism driver openvswitch
```

Note If huge page is used, the memory used in the flavor must be exact multiples of the huge page sizes. For example,memory must be multiple of 2 if 2M huge page is used, and multiple of 1024 if 1G huge page is used.

VMTP Parameters

```
VMTP_VALIDATION parameters: #Required if vmtp is enabled
VMTP_VALIDATION:
    VTS_NET:             #Required if VMTP is enabled for VTS (for VTS only this block is
needed)
    ENABLED: <true or false>
```

Networking Parameters

```
NETWORKING:
      ...
networks:
...        -
vlan_id: <VLAN to carry VTS tenant traffic>    # required for VTS
subnet: <subnet IP cidr>
gateway: <tenant GW IP>
pool:
- "<begin tenant IP> to <end tenant IP>"     # ***
segments:
- tenant
```

Note The tenant network pool size has to take into account the IP addresses that are statically assigned through the VTS VTSR VM bootstrap configuration. For more information, see the Installing Cisco VTS

Cisco VTS Parameters

```
VTS_PARAMETERS:
VTS_USERNAME: 'admin'                  # Required to be 'admin'
VTS_PASSWORD: <VTC UI password>
VTS_NCS_IP:   <VTC mx-net IP>         # VTC mx-net VIP for VTC HA (cannot be in mx-net pool
 range)
VTS_SITE_UUID: <VTS site uuid>     # VTS SITE UUID mandatory VTS parameter (Unique Pod UUID
 to indicate
                                        which pod the VTS is controlling)
VTC_SSH_USERNAME: '<vtc_ssh_username>' # Required parameter when VTS Day0 is enabled or
running VMTP
VTC_SSH_PASSWORD: '<vtc_ssh_password>' # Required parameter when VTS Day0 is enabled or
running VMTP

VTS_Day0_PARAMETERS:
VTS_2.5 mandates the VTC inventory generation and day0 configuration for VTF's to register.
without VTS_DAY0 the cloud is not operational as VTF does not register to VTC. Hence all
cloud operations fail.
This is a boolean variable set as True or False. If set True, VTC day0 can be configured
by the Cisco VIM Installer.
By default values is 'False', i.e. if VTS_DAY0 is not set, the orchestrator sets it internally
 to 'False'
VTS_DAY0: '<True|False>'
```

```
## Optional, BGP_ASN:
   BGP_ASN: int  # Optional, min=1, max=65535;   if it is not defined, the default to 23
## Optional, MANAGED:
  MANAGED : <TRUE OR FALSE> #Optional; if it is true, tor_info in SERVERS becomes mandatory,
 CONFIGURE_TORS under
                                TORSWITCHINFO should be false and VTS deployment mode is
 managed.
```

Note The mx-net IP pool configuration must take into account the IP addresses that are allocated to the VTC (VTS_NCS_IP). For more information, see the Installing Cisco VTS

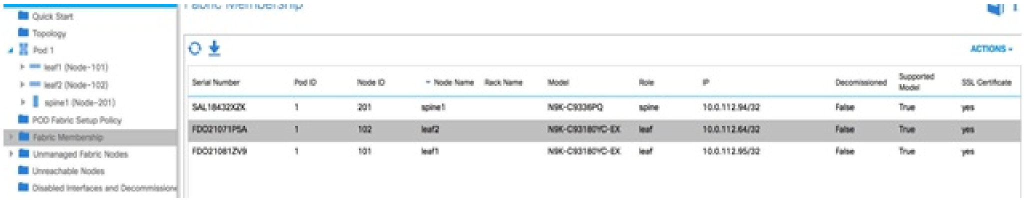
Enabling ACI in Cisco VIM

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.

As Cisco VIM does the day-0 configuration of the ACI, following are the assumptions that Cisco VIM makes for the integration to happen.

- Before the Cisco VIM installation, the APIC 3.2 controllers running in a cluster of three must be installed and active.
- 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 install.

  The IP address should be assigned to each device from the TEP_ADDRESS_POOL.



Serial Number	Pod ID	Node ID	Node Name	Rack Name	Model	Role	IP	Decomissioned	Supported Model	SSL Certificate
SAL18432XZK	1	201	spine1		N9K-C9336PQ	spine	10.0.112.94/32	False	True	yes
FDO21071P5A	1	102	leaf2		N9K-C93180YC-EX	leaf	10.0.112.64/32	False	True	yes
FDO21081ZV9	1	101	leaf1		N9K-C93180YC-EX	leaf	10.0.112.95/32	False	True	yes

- Network should be designed such that the management node and controllers are reachable to APIC controllers.
- 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.
- 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.
- TEP_ADDRESS_POOL, and multicast address pool are immutable for the lifecycle of the infrastructure.

Note Using Auto-ToR provisioning via 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 which 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:
  global:           # Global policies can be overridden per server role
    tor_info:       # <list of global fabric policies for control plane>
    - <str>         # string, E.g. hintfpol-25G-On-RS-FEC.  Must be pre-provisioned.
    - <str>         # string E.g cdpIfP-CdpDisable
    dp_tor_info:    # <list of policies for data plane for Intel NIC>
    - <str>         # string, E.g. lacplagp-LacpActive.  Must be pre-provisioned.
    - <str>         # string E.g lldpIfP-LldpEnable
    sriov_tor_info: # <list of policies for sriov interfaces>
    - <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>
    - <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>
    - <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
        - <path_to_epg_policy1>    # Must be pre-provisioned.
        - <path_to_epg_policy2>    # Must be pre-provisioned.
        tenant:                    # Optional, list of policy for tenant segment
        - <path_to_epg_policy1>    # Must be pre-provisioned.
        - <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
    - <str-1>            # string E.g. 'Server-VlanPool'
    - <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
    - 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'
  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:
    - '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
    - 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 -> provider section
```

Note Ensure that you update right VLAN information when using this option in context of APIC during reconfiguration of VLANs.

Setting of Memory Oversubscription Usage

Cloud allows you for over-subscription of resources (CPU, Memory, storage). The memory oversubscription value is set to 1.5. Cisco VIM gives the flexiblility to change the default values at the beginning of the installation. You can adjust the memory oversubscription value between 1.0 to 4.0.

Run the following command to set the NOVA_RAM_ALLOCATION_RATIO, on fresh install:

```
# cd installer-<tagid>/openstack-configs/
# update NOVA_RAM_ALLOCATION_RATIO value in openstack_config.yaml
```

What to do next

Once the NOVA_RAM_ALLOCATION_RATIO is set, continue with the rest of the steps as planned for installation.

Setting of CPU Oversubscription Usage

Cloud allows you for over-subscription of CPU, storage and memory. The CPU oversubscription value is set to16.0. Cisco VIM gives the flexiblility to change the default values before the installation begins. You can adjust the CPU oversubscription value in the range of 1.0 to 16.0.

Run the following command to set the NOVA_CPU_ALLOCATION_RATIO on fresh install:

```
# cd installer-<tagid>/openstack-configs/
# update NOVA_CPU_ALLOCATION_RATIO value in openstack_config.yaml
```

What to do next

Once the NOVA_CPU_ALLOCATION_RATIO is done, continue with the rest of the steps as planned for installation.

Setting up CPU and RAM Allocation Ratio per Server

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.

To enable this feature, update each of the target compute nodes with the following configuration in the setup_data.yaml file:

```
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
```

You can enable this feature on day 0 or enable it on a per compute basis on day 2 by removing it and adding it back into the cloud after updating the setup_data.yaml file with the configuration.

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.

Note

- The MGMTNODE_EXTAPI_REACH variable must be set during the initial install, 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 True, features such as VMTP and NFVbench are no longer accessible from the management node.

Enabling NFVbench on Cisco VIM

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 mechanism driver is OVS or VPP.

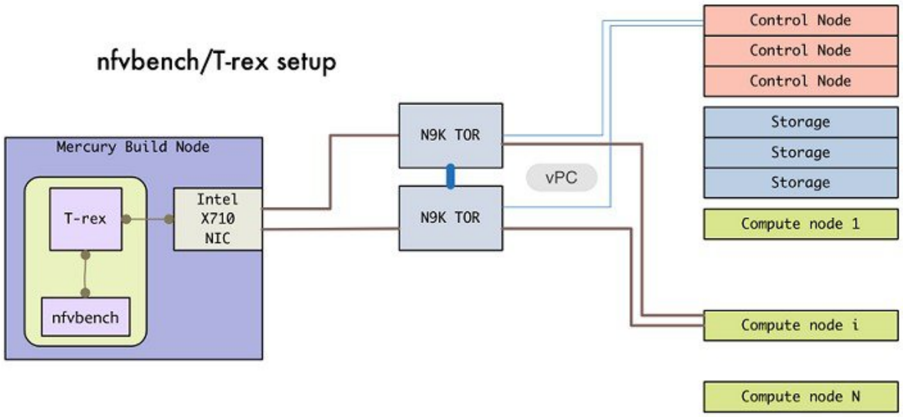
Before you begin

- If you are using Quanta servers, see Installing the Management Node on the Quanta Servers, for the day-0 BIOS setting of the management node.
- NFVbench offering in Cisco VIM, requires an extra Intel NIC (Intel X710 NIC (4 x 10G) or Intel XL710 (2x40G)) or Intel xxv710 (25G) to be installed on the management node.

- To interact with Intel NIC, TRex traffic generator uses DPDK interface, and makes use of hardware instead of just software to generate packets. This approach is more scalable and enables NFVbench to perform tests without software limitations.

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. In case of only one ToR switch connect the first two ports to it as shown in the NFVbench topology figure.

Figure 2: NFVbench topology setup



Step 1 To enable the NFVbench, set the following command:

```
NFVBENCH:
   enabled: True    # True or False
   tor_info: {switch_a_hostname: ethx/y, switch_b_hostname: ethx/y} # mandatory
#  tor_info: {switch_c_hostname: 'etha/b,ethx/y'} # use if there is only one TOR switch
#  nic_ports: int1,int2             # Optional input, indicates which two of the four ports in the
10G intel NIC ports on the
                                      management node is used by NFVBENCH tool to send and receive
traffic. If nothing is
                                      specified, the tool assumes it is port 1,2 i.e. the first two
 ports are used

# nic_slot: <int> # Optional, defaults to 1st set of unbonded pair of NIC ports in an Intel 710 or
520 card the code finds; Via this option, one can choose to run NFVbench via XL710, 520 or X710 card
#
# For VTS/VXLAN, enable the following:
# vteps: "vtep_ip1,vtep_ip2"        # Mandatory and needed only for VTS/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 reconfigurable and separated IP pairs in
 vxlan-tenant network and not in the
                                        vxlan-tenant pool
#  vnis: "vni_id1, vni_id2"          # Mandatory, specify reconfigurable and separated vnid_id pairs
#


 # Note: if nic_ports are defined, then nic_slot has to be defined and vice-versa
   # Minimal settings always required with NFVbench
   TORSWITCHINFO:
   CONFIGURE_TORS: True
   …
```

```
SWITCHDETAILS:
- hostname: <switch_a_hostname>
  username: admin
  password: <password>
  ssh_ip: <ssh access to the switch a

- hostname: <switch_b_hostname>
  username: admin
  password: <password>
  ssh_ip: <ssh access to the switch b
```

The tor_info provides the information to configure the TOR switches. Two ports specified by interfaces will be 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 'True'.

With VTS as mechanism driver additional settings are needed. NFVbench needs access to VTS NCS to perform cleanup after it detaches traffic generator port from VTS. Also a pair of VTEP VLANs is required for VLAN to VxLAN mapping. Value can be any random VLAN ID. Note that vtep_vlans field is required if VxLAN is used as encapsulation without VTS.

Step 2 To do manual configuration on the ToRs, we recommend you to perform the following configurations:

```
interface Ethernetx/y
  switchport mode trunk
  switchport trunk allowed vlan <3000-3049>
  spanning-tree bpdufilter enable
```

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 sets 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 setup_data 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, and if undefined default 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:

```
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 (less than 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>
```

Note You can enable single or multiple options listed above on a per server basis.

NFV Host Configuration

NFV Host configuration describes how to configure NFV hosts and Cisco VIM monitoring.

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 VTS and 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 and VTS, only NFV_HOSTS: ALL is allowed
NFV_HOSTS: ALL
or
NFV_HOSTS: ['compute-server-1']
```

By default, hyper-threading is enabled across compute nodes in Cisco VIM. Based on certain VNF characteristics, Cisco VIM offers user the capability to disable hyper-threading across the pod on day-0. You can also disable it on a single compute node on day-n, updating the setup_data and doing remove or add of compute nodes (see Utilizing NUMA features in Cisco NFV Infrastructure section in the Cisco VIM Admin Guide for details on day-n operation). To disable hyper-threading, update the setup_data with the following name or value pair before starting the installation.

```
DISABLE_HYPERTHREADING: True or False; this is optional and default value is false.
```

Install Mode

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

1. Connected: In this mode, the setup must be connected to Internet to fetch artifacts and docker images.
2. 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.

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

Enabling NFVIMON on Cisco VIM

The Cisco VIM solution uses Cisco NFVI Monitor (NFVIMON) to monitor the health and performance of the NFVI. This includes monitoring both the physical and logical components of single or multiple NFVI pods. The NFVIMON feature enables 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 such that it can monitor single or multiple pods from a single management system. NFVIMON is enabled by extending the setup_data.yaml file with relevant information. You can enable NFVIMON on an existing pod through the reconfigure option. Then, add the pod as the VIM resource to be monitored in a Control Center.

NFVIMON consists of four components: ceilometer service (for data collection), collector, resource manager (RM), and control-center with Cisco Zenpacks (CZ). Integration of NFVIMON into VIM is loosely coupled and the VIM automation only deals with installing the ceilometer service software needed to monitor the pod. The installing of the other NFVIMON components (collector, resource manager (RM) and control-center with Cisco Zenpacks (CZ), are outside the scope of the install guide.

Before you Begin

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) is available only through Cisco Advance Services. At a high level, have a node designated to host a pair of collector VM for each pod, and a common node to host CC and RM VMs, which can aggregate and display monitoring information from multiple pods.

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.

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 services activities.

Installation of 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 Zenpack available in the controller node gathers the node level information.

To enable NFVIMON as part of the VIM installation, update the setup_data 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> #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> # 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> # 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}

# 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> # 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> # 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
```

Note If NFVIMON HA is enabled, ensure that all the admin IPs are on the same subnet for NFVIMON VMs and deployed servers.

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
      ....
```

Enabling CVIM-MON on Cisco VIM

The Cisco VIM solution offers the use of Cisco VIM Monitor (CVIM-MON) to monitor the health and performance of NFVI. This includes monitoring both the physical and logical (openstack services) components at each NFVI pod level.

The CVIM-MON feature enables extensive monitoring and collection of performance data for various components of the cloud infrastructure, and also the OpenStack instances. The monitoring system is designed at a single pod level.

CVIM-MON is enabled by extending the setup_data.yaml file with relevant information.

You can enable CVIM-MON on an existing pod that is installed with Cisco VIM 2.4.3 or later, through the reconfigure option.

The components of CVIM-MON are as follows:

- **CVIM_MON:** It provides the base functionality of monitoring and KPIs.

- **SNMP**: It is enabled for SNMP traps and available only if CVIM_MON is enabled.
- **SERVER-MON:** If SNMP is enabled, you can enable SERVER_MON to use SNMP from the Cisco IMC of Cisco UCS C-series server. This component is available only if the SNMP option is enabled and only on Cisco UCS C-series servers.

Install the CVIM-MON using the standard Cisco VIM installer after enabling it in the setup_data configuration file. It is assumed that the pod is newly installed with Cisco VIM 2.4.3 or later. To install CVIM-MON, CVIM_MON and PODNAME keys must be added to the setup_data.yaml file.

The CVIM_MON key has:

- enabled: A boolean value indicating whether CVIM_MON is enabled.
- polling_intervals: It is a dictionary having three different levels of data collection frequencies. Defining polling_intervals is optional and a default value is used if the polling_interval is not defined.
- ui_access: A boolean indicating whether CVIM-MON UI access is enabled or not.

PODNAME is mandatory for CVIM-MON.

CVIM-MON, SNMP, and SERVER-MON can be installed by the standard Cisco VIM installer, if they are enabled in the setup_data configuration file.

Following are aspects of the SNMP key:

- If SNMP is enabled, CVIM-MON must be enabled.
- List of SNMP managers to send the SNMP traps. This list contains SNMPv2 or SNMPv3 managers. For SNMPv2, community and port field can be set. For SNMPv3, the engine_id and list of users must be specified, where the Engine_id is the EngineContextID which is used to send trap of the SNMP Manager.

Note SNMP-Traps are sent without setting any authentication or security engine_id for the user.

Property Group and Name	Values	Default Value	Description
PODNAME:	<string>	(required)	Must be provided for identifying each pod if 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 acces is enabled or not.

Property Group and Name	Values	Default Value	Description
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, on page 61 .
CVIM_MON: polling_intervals:	-	-	Metric collection frequency 10s <= low frequency <med frequency < high frequency <=1 hour
low_frequency	-	deprecated	Must be higher than med_frequency integer following with time sign (m/h)
medium_frequency	-	deprecated	Must be higher than high_frequency integer following with time sign (s/m/h)
high_frequency	10s to 30s	15s	Integer following with time sign (s/m/h)
SNMP:enabled	true|false	false	A Boolean indicating whether SNMP trap is enabled or not. If true, CVIM_MON:enabled must also be set to true.
SNMP:managers:	-	-	A list of up to 3 SNMP managers to send traps
address	<ipv4 or ipv6>	(required)	IPv4 or IPv6 address of the SNMP manager
port	1-65535	162	Optional, port to send traps
version	v2c|v3	v2c	SNMP manager version
community	<string>	public	Used for SNMPv2c
SNMP:managers:users:			Required for SNMPv3, up to 3 users.

Property Group and Name	Values	Default Value	Description
engine_id	<hexadecimal string>	(required v3)	ContextEngineId (unique across all managers) Minimum length is 5 and max length is 32 Cannot be all 00s or FFs; and cannot start with 0x
name	<string>	(required v3)	User name
auth_key	<string>	(required v3)	Authorization password, must be eight characters at least
authentication	SHA|MD5	SHA	Authentication protocol
privacy_key	<str>	(auth_key)	Encryption key
encryption	'AES128'|'AES192'|'AES256'	'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	‘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

Note 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.

Monitoring External Servers Using CVIM MON

From Cisco VIM 3.4.1, CVIM MON can monitor external non-CVIM servers running RHEL or CentOS 7.6. CVIM MON monitors these servers by installing a Telegraf agent on them. 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.

The following telegraf plugins are enabled:

- cpu
- disk and diskio
- net

- mem
- ipmi_sensor
- kernel
- processes
- swap
- system
- conntrack
- internal: Monitors the health of the Telegraf agent and enabled input plugins
- libvirt: Collects metrics related to any virtual machines running on the external server
- prometheus_client: Provides scraping access to any Prometheus compatible scraper

Impact of Prometheus TSDB

- Metrics collected from external servers are distinguished from CVIM pod metrics by a node_type label value of “external”.
- Metrics for all CPUs have the label “tag” set to “host” by default (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).
- External servers are considered as part of the attached CVIM pod.
- For central monitoring, external servers are assigned the same region or metro as the attached CVIM pod.

In the Grafana dashboard, all counters such as node count per type, alert counters, power consumption aggregates and so on will include external servers as applicable.

Assumptions

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 CVIM management node ISO or CentOS 7.7.
- The external nodes must run in the same site as the monitoring Cisco VIM pod. The scraping of metrics occurs over unauthenticated and unencrypted HTTP connections on port 9273.

Installation Procedure

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. As part of the fresh installation or reconfiguration operation, an `external-monitoring-<telegraf version>.tar.gz` tar file is created in the management node at `/var/cisco/artifacts`.

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 to any directory in the external server.
2. Open port 9273 on the respective external server 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 three files:

File	Description
`extern.conf`	Provides the telegraf configuration
`telegraf<version>.x86_64.rpm`	Installs telegraf on the server
`monitor_external.sh`	Deploys telegraf on the external server

4. To deploy telegraf on the external server, run the monitor_external bash script in privileged mode:

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

   If the script runs successfully, TELEGRAF INSTALL SUCCESS message appears on the console screen. You can now access either the local or central prometheus or grafana, and monitor the external server.

Support of CVIM-MON Grafana with LDAP Backend

The CVIM-MON Grafana 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 with your LDAP credentials. Once the filter is set, it is possible to map the users groups with specific roles of permission in Grafana.

CVIM-MON supports the roles of:

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

To enable LDAP for Grafana, 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:
  enabled: true
  ldap:
    group_mappings:
    - {group_dn: 'CN=Employees,OU=Openstack Users,DC=mercury,DC=local', org_role: Admin}
    - {group_dn: 'CN=OS-ReadOnlyGrp,OU=Openstack Users,DC=mercury,DC=local', org_role:
Viewer} # Optional, Can have only 1 Admin and 1 Viewer User
    domain_mappings: # Max of one domain name for now
    - domain_name: corp_ldap1
     attributes: {email: email, member_of: memberOf, name: givenName, surname: sn, username:
 cn}
      bind_dn: cn=osadmin,cn=users,dc=mercury,dc=local
     bind_password: Lab1234! # optional, this is possible when bind_dn matches all possible
 users
      ldap_uri: "ldap://172.29.68.196:389"  # , separated one or more ldap end points
      search_base_dns: ['dc=mercury,dc=local']
      search_filter: (cn=%s)
```

Property	Field Required	Description
search_filter	Mandatory	Filter for the queries.
search_base_dns	Mandatory	The base dns name used for all queries
ldap_uri	Mandatory	URI used to connect to the LDAP servers (at least one, multiple URIs can be configured - separated by a comma)
group_mappings	Mandatory	Must contain at least one group with org_role **Admin** Optionally, you can add a second group with org_role **Viewer**
domain_name	Optional	Any non empty name is acceptable
domain_mappings	Mandatory	Must contain one domain exactly.
bind_password	Conditional	This is the password of the bind_dn user. When the bind_dn is a group, this field must be omitted.
bind_dn	Mandatory	Refers to the users who can connect to the LDAP sever to check credentials. It can be a read-only user or a group that matches all possible users.
attributes	Mandatory	All subkeys are mandatory.

Enabling Inventory Discovery with CVIM-MON

CVIM_MON is a pre-requisite to enable the feature of inventory discovery. You can enable this feature by setting the following flag in the setup_data.yaml file:

```
INVENTORY_DISCOVERY: {enabled: true}
```

Use the following commands to verify whether the feature is operational:

```
docker ps | grep calipso  - expected results: 3 containers are in ‘UP’ state-
calipso_scan_<tag>, calipso_api_<tag> and calipso_mongo_<tag>
 where calipso_client, calipso_replication_client and calipso_csv_tool are available in
bash and provides –help guidance.
```

To get the API secret, use the command:

```
ciscovim list-secrets --getpassword CALIPSO_API_SERVICE_PWD
```

Note For more information on usage of inventory discovery, see *Cisco Virtualized Infrastructure Manager Administrator Guide*

Enabling or Disabling Autobackup of Management Node

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.

To enable or disable the management node, update the setup_data.yaml file as follows:

```
# AutoBackup Configuration
# Default is True
#autobackup: <True or False>
```

Enabling Custom Policy for VNF Manager

Some of the VNF managers operates, using specific OpenStack features that require the admin role. Cisco VIM introduces a feature to enable non-admin role for VNF managers (such as Cisco ESC). VNF manager is used to operate and manage tenant VMs in the OpenStack cloud, with minimally enhanced privileges.

To enable this option, the administrator needs to add the following line in the setup_data.yaml:

```
ENABLE_ESC_PRIV: True # optional; default is false
```

Forwarding ELK 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 ELK logs to multiple external syslog server.It supports minimum of one and maximum of four external syslog servers.

Before launching the installation, update the setup_data.yaml file with the following information:

```
###################################
## SYSLOG EXPORT SETTINGS
###################################
SYSLOG_EXPORT_SETTINGS:
  -
    remote_host: <Syslog_ipv4_or_v6_addr> # required IP address of the remote syslog
    server protocol : udp # defaults to udp
    facility : <string> # required; possible values local[0-7]or user
    severity : <string; suggested value: 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 #2 (optional)
 server protocol : udp # defaults to udp
 facility : <string> # required; possible values local[0-7]or user severity : <string;
suggested value: debug>
 port : <int>; # defaults, port number to 514 clients : 'ELK' # defaults and restricted to
 ELK;

# Please note other than the remote host info, most of the other info is not needed; Also
the client list is restricted to ELK only
```

With this configuration, the ELK logs are exported to an external syslog server. You can add this configuration to a pod that is already up and running. For more details, refer to Forwarding ELK logs to External Syslog Server section in the admin guide.

Support of NFS for ELK Snapshot

Cisco VIM optionally supports NFS for ELK snapshots. In this configuration, the remote location specified in the configuration has to allow user elasticsearch (2020) and group mercury (500) to read/write into the path specified in remote_path of the remote_host server.

Before launching the installation, update the setup_data.yaml file with the following information:

```
###################################
## ES_REMOTE_BACKUP
###################################
#ES_REMOTE_BACKUP:          # Set if Elasticsearch backups will use 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 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. For more details, refer to Support of NFS for ELK Snapshot section in the admin guide.

Support for TTY Logging

Cisco VIM supports enabling of TTY logging on the management node and all of the cluster hosts through the option in the setup_data.yaml file. By default, the TTY logging feature is not enabled. The feature is made available only at the time of 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, the following message is displayed: `Management node needs to be rebooted for TTY Logging to take effect.`

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
```

Configuring Additional VIM Administrators

Cisco VIM supports management of VIM administrators. VIM administrator can login to the management node or Unified Management node 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 reconfiguation, while the passwords for existing accounts remain unchanged.

Before launching the installation, update the setup_data.yaml file with the following information:

```
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.
```

With the preceding configuration, administrators have access to a shell with system privileges on the management node. To go hand in hand with the management of VIM administrator, Cisco VIM offers the option of disabling “root login”. Listed below are the available options:

```
# 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: True
```

Support of LDAP for Management Node

Cisco VIM supports enabling of LDAP for admin access to the management node. It can be added as a 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_uris is secured over ldaps, and the TLS is enabled for the external api (external_lb_vip_tls: True).

To obtain sudo access to the management node and execute ciscovim commands, you must manually add the user with root privileges to the wheel group in the corresponding LDAP domain, for example, usermode -aG wheel user1.

To enable this feature, update the setup_data with the following during installation.

```
vim_ldap_admins:
  - domain_name: corp_ldap1
    ldap_uri: "ldaps://10.30.116.253:636,ldaps://10.30.116.254:636"
    ldap_search_base: "dc=cisco,dc=com"    # Optional
    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
  - …
```

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.

An optional parameter HORIZON_ALLOWED_HOSTS added in the setup_data 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.

Before launching the installation, update the setup_data.yaml file with the following information:

```
HORIZON_ALLOWED_HOSTS:
  - <NAT-IP>
  - <NAT-IP>
```

With the preceding configuration, administrator can access the Horizon dashboard through aliases or NAT IPs.

DHCP Reservations for VM's MAC Addresses

From release Cisco VIM 3.2.0, 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

For example, the MAC address entry can be [a-f][2,6,a,e]:yz:uv:ws:00:00.

To enable this feature, add the following entry in the setup_data file:

```
BASE_MACADDRESS: <[a-f][2,6,a,e]:[a-f0-9][a-f0-9]:[a-f0-9][a-f0-9]:[a-f0-9][a-f0-9]:00:00>
```

Note 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>

Customizing SSH Login Banner

From release Cisco VIM 3.0.0, you can provide a customized banner that will be 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.

Before launching the installation, update the setup_data.yaml file with the following information:

```
ssh_banner: |
   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.
```

Cinder Volume Encryption

From release Cisco VIM 3.0.0, you can encrypt Cinder volumes using Linux Unified Key Setup (LUKS). This encryption is enabled by default. and does not require any installation. For more information on creating encrypted Cinder volumes, see *Cisco VIM Admin Guide* .

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 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 cleartext 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 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”.
```

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.

Configuring Support for Read-only OpenStack Role

By default, Cisco VIM deployment of OpenStack supports two user roles: admin and user. Admin have privilege to view and change all OpenStack resources including system and project resources. Users have privileges to view and change only project resources.

Optionally, Cisco VIM provides OpenStack user role which is read-only or readonly. Read-only users can view the project resources, but cannot make any changes. Use the optional parameter ENABLE_READONLY_ROLE to enable this feature.

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.

Note 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".

- **All Projects** 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.

Before launching the installation, update the setup_data.yaml file with the following information:

```
ENABLE_READONLY_ROLE: True
```

With the preceding configuration, the readonly role is created in OpenStack. After deployment, the administrators have the privilege to create new users assigned with this role.

Note If the ENABLE_READONLY_ROLE is False (by default), the readonly role will not have special permissions or restrictions, but have 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.

VPP Port Mirroring Support

The VPP Port Mirror feature 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 can be saved as pcap files, which can be used for further analysis by tools like Wireshark and so on.

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 current mirrored port

In addition, the VPP port mirror tools perform the following tasks:

- Checks if the port specified 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

VPP Port Mirroring Usage

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

Step 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' |
```

Step 3 ssh to the target compute node on which the VM is running and join 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.
```

Step 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
```

Note Use the –debug flag to troubleshoot the Linux/VPP commands that are used to set up the port mirror.

Step 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
```

Step 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
```

Step 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
```

Setting up VXLAN/EVPN in Cisco VIM

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.

Step 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
-
```

Step 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' ---> 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' ---> 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)
```

Note 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.

Step 3 In the **SERVERS** section, define vxlan_bgp_speaker_ip for each controller node.

Note The vxlan_bgp_speaker_ip belongs to the vxlan network, however, it is not part of the IP pool defined in the vxlan segment.

```
SERVERS:
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
```

Note 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;
                the ip addresses are unique and are from management network, but are not part of the
 pool
```

Setting up Trusted Virtual Functions

The kernel feature allows the Virtual Functions 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.

Setting up Reception/Transmission Buffer Size

By default, the transmission and reception buffer for the interfaces on each server is set to 1024. This feature allows you to set the rx_tz_queue_size to 256, 512, or 1024 on a per server basis based on the requirement for some VNFs. Also, along with setting the queue size, you can disable the seccomp syscall sandbox in QEMU to avail this feature.

To avail this feature, enable the following under each of the target compute nodes.

```
SERVERS:
compute-server-1:
rx_tx_queue_size: <256 or 512 or 1024> # optional only for compute nodes, default if not
defined is 1024
seccomp_sandbox: <0 or 1> # optional, Set to 1 by default, if not defined.
```

Updating Cisco NFVI 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. Updating Cisco NFVI software has minimal service impact because the update runs serially, component-by-component, one node at a time. If errors occur during an update, an automatic rollback will bring 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.

Note Cisco NFVI software updates are not supported for registry related containers and authorized_keys. Also, after the management node repo containers are updated, they cannot be rolled back to the older versions because this requires node packages to be deleted, which might destabilize the cloud.

Note Update of Cisco NFVI software is within the same major version, that is from 3.2.0 to 3.2.1, and not from 2.4 to 3.0.

To prevent double faults, a cloud sanity check is done both before and after the update.

To complete the software update, perform the Installing Cisco VIM m_install_vim.ditamap#id_33373. If your management node does not have Internet, complete the m_preparing_usb_stick.ditamap#id_38540 procedure 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.

Note After you complete a software update, you must commit it before you perform any pod management operations. During software updates, the following operations are locked: add/remove compute/storage node, replace controllers, and rotate fernet key. Before you commit, you can roll back the update to return the node to its previous software version.

For information on updating the Cisco NFVI software, see *Managing Cisco NFVI* of the corresponding *Cisco VIM Administrator Guide*