Installing Cisco VIM

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

- Cisco VIM Installation Overview, page 1
- Installing Cisco VIM, page 2
- Cisco VIM Client Details, page 3
- Cisco VIM Configuration Overview, page 6
- Updating Cisco NFVI Software, page 36
- Upgrading Cisco NFVI Software, page 37

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 Accessprocedure. 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.

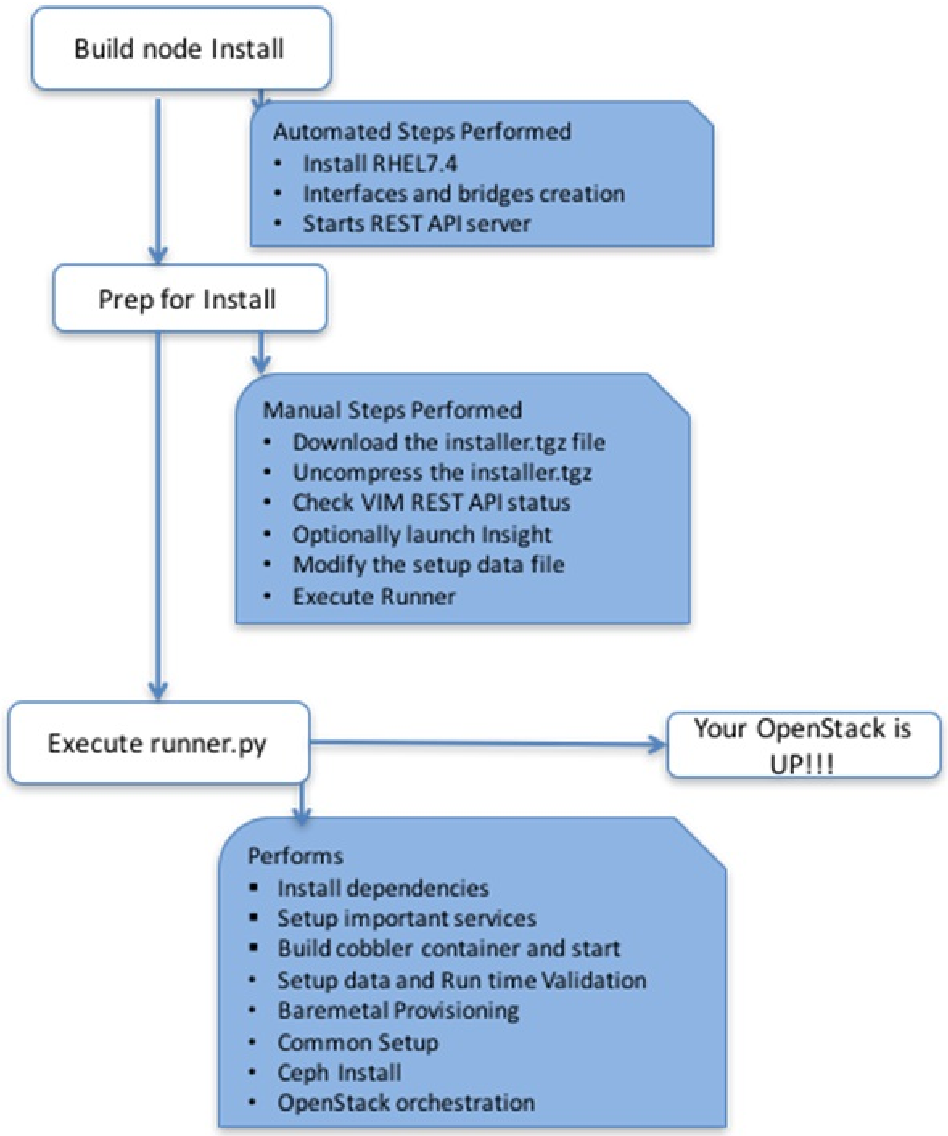
Completing these procedures ensures the Cisco NFVI network infrastructure is set up before the Cisco VIM installation. The bootstrap script is then kicked off, which downloads installer repository, installs Docker and dependencies and starts installer web service,

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 Preboot Execution Environment (PXE) boot of RHEL7.2 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 RHEL7.4
• 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 Insight, you will 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 Insight to complete the configuration and setup of Cisco VIM using the Cisco VIM Insight. If you do not have Cisco VIM Insight, you will 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 Cisco VIM on a Cisco NFVI management node:

Before You Begin

- You need Cisco NFVI installation files download site credentials from your Cisco account representative.
- For Management nodes with NO Internet access, you need USB stick containing the Cisco NFVI installation files. To prepare the USB stick, see the Preparing to Install Cisco NFVI on Management Nodes Without Internet Access.

Step 1 If your management node does not have Internet access and you prepared a USB stick in Preparing to Install Cisco NFVI on Management Nodes Without Internet Access, 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 ~/insight-<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 Insight 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 ~/insight-<tag_id>
```

Step 4 Create a dir (for example, ~/Save/) to contain a copy of the setup_data.yaml file, the file that will configure 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
```

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. For details, see Cisco VIM Configuration Overview, on page 6.

Step 7 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** will show 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> --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
    period-rotate-fernet-keys    Set the frequency of fernet keys rotation on
                                 keystone
    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
3	Validation (software and hardware)	30 seconds
4	Bare metal install	60 minutes
5	Common setup	10 minutes
6	Ceph	5 minutes
7	Orchestration	25 minutes
8	VMTP (external and provider networks)	14 minutes

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 must be performed prior to running the Cisco VIM installation. If you are installing Cisco Insight, you will complete the Cisco VIM data and OpenStack configurations using VIM Insight as described in Installing Cisco VIM through Cisco VIM Insight .

Configuring ToR Automatically

Cisco VIM 2.2, provides a complete automation of the cloud deployment. Cisco VIM, of this feature is to automate day-0 configuration of N9xxx series Top of Rack(ToR) switches. The feature is optional and only applies to Pods that are running without ACI. For ToR switch details related to ACI, refer to the section, "Enabling ACI in Cisco VIM".The purpose is to automate Power-On Auto Provisioning (post-POAP) configuration on ToR offering of Cisco VIM, which constitutes of one or more pair of identical Cisco N9300 series switches. The day-0 ToR automation configures the interfaces 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. The 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, the management interface, vrf, and admin user needs to be pre-provisioned on each of the ToR switch. Also, ssh needs to be enabled in each ToRs. 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. The installer API interface (br_api) on the management node needs to be up and running, and the ssh to the management node through SSH must be working. You should be able to access each of the ToRs through its management interface from the Cisco VIM management node using SSH.

Setting Up the Cisco VIM Data Configurations

The Cisco VIM configuration file, setup_data.yaml, installs and configures the VIM deployment. When creating this file, take extreme care. Any change to this configuration after deployment, with the exception (example: NFVIMON, of adding and removing nodes etc) will cause a stack redeployment. Pay particular attention to the pod networking layout plan configured in setup_data.yaml because any future changes to it will require the pod to be reinstalled.

If your configurations are correct, the installation will go 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, because your browser might render the characters differently. If you are using the Cisco VIM installer, you will not be able to 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 later software updates and upgrades. This ensures a consistent and repeatable installation, which is important. Key setup file parts are shown in the following sections.

Setting Up the 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 entire TORSWITCHINFO section is not provided or CONFIGURE_TORS attribute under TORSWITCHINFO then all the ToR provisioning related steps will be skipped. The general 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 will derive 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:
    ENABLE_QOS_FOR_PORT_PROFILE: true,
    ENABLE_QOS_POLICY: true,
    ENABLE_UCSM_PLUGIN: true,
     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 SERVERES 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.

Setting Up Server Level information for C-series with Intel NIC

When the C-series pod is configured to run in a complete Intel NIC environment, the ToR configurations 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 control and data interfaces facing each for 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

You can use the Auto-ToR configuration feature to setup NCS-5500. The mercury 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 for the pod.

The following code 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>
      vpc_peer_port_address:  <local address with mask for vpc_peer_port_info>
      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
    -
      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>
      vpc_peer_port_address:  <local address with mask for vpc_peer_port_info>
      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


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}

      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}
      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
```

Intel NIC Support

Cisco VIM supports C-series pod running with either all Intel 710X NICs or Cisco VICs. In the case of Intel NIC, each server needs to have 2 of 4 port 710X cards. 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>
```

A C-series pod, running Intel NIC, also supports SRIOV. By Default, SRIOV is not supported. To enable, 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 Intel 710, 1 port (port #c) from each of the Intel NICs are used for SRIOV, but there might be some VNFs which need additional SRIOV ports to function. To meet this requirement, in VIM 2.2, an additional variable has been introduced in setup_data.yaml by which user can include a 2nd 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

```
#INTEL_SRIOV_PHYS_PORTS: <2 or 4>
```

The parameter, INTEL_SRIOV_PHYS_PORTS is optional, and if nothing is defined a value of 2 will be used. The only 2 integer values parameter takes is 2 or 4.

Intel NIC Support for SRIOV only

In Cisco VIM 2.2, we also support C-series pod running with 1 2-port Cisco VIC for control plane and 2 2-port Intel 520s for SRIOV (called VIC/NIC deployment). 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 pod, running Cisco VIC/Intel NIC (520), also supports SRIO V on the Intel 520 Nics . By Default, SRIOV is not supported. To enable, define a value in the range 1-63 (63 is maximum) number of INTEL_SRIOV_VFS: <integer>

By default in the C-series pod running with Cisco VIC and Intel 520, the control plane runs on the Cisco VIC ports, and all the 4 ports from the 2 Intel 520 NICs are used for SRIOV. In the pods running with CISCO_VIC_INTEL_SRIOV enabled, some computes can run only with Cisco VIC without SRIOV option present on it.

Remote Registry Credentials

```
REGISTRY_USERNAME: '<username>'
REGISTRY_PASSWORD: '<password>'
REGISTRY_EMAIL: '<email@address.com>'
```

Common CIMC Access Information for C-series POD

```
CIMC-COMMON:
cimc_username: "admin"
cimc_password: <"cisco123">
```

UCSM Common Access Information for B-series POD

```
UCSMCOMMON:
ucsm_username: "admin"
ucsm_password: <"cisco123">
ucsm_ip: <"a.b.c.d">
ucsm_resource_prefix: <"skull"> # max of 6 chars
ENABLE_UCSM_PLUGIN: <True> #optional; if True, Cisco-UCSM will be used, if not defined,
default is False
MRAID_CARD: <True or False>
ENABLE_QOS_POLICY: True or False # only allowed if ENABLE_UCSM_PLUGIN is True
ENABLE_QOS_FOR_PORT_PROFILE: <True or False>
```

Note When you use Cisco UCS Manager to enable QOS Policy, remember that in certain NFV solutions guest VM (SRIOV) traffic must have heartbeat messages moving across the VMs at a higher priority. In this case the UCS Manager plugin uses a predefined QOS policy name, created by the installer, to attach to the port profile. Cisco VIM does not change the QOS flags that UCS Manager provides by default. You can configure two types of QOS profiles: nfvi (default) or media. For NFV, VM heartbeat messages will have a higher priority. For media, multicast traffic is prioritized on the tenant/provider network over other types of traffic such as SSH and HTTP. The QOS policy with UCS Manager is an optional feature. By default this feature 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 will be 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 should 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
    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
    - 10.0.0.0/8
    - 172.16.0.0/12
```

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: <107>
   subnet: <10.30.115.192/28> # true routable network
   gateway: <10.30.115.193>
   pool:
       - 10.30.115.194 to 10.30.115.206
   segments:
       - cimc
vlan_id: <108>
   subnet: <10.30.116.192/28> # true routable network
   gateway: <10.30.116.193>

ipv6_gateway: 2001:c5c0:1234:5678:1003::1    <== require if IPv6 OpenStack public API is
enabled
ipv6_subnet: 2001:c5c0:1234:5678:1003::/80
   segments:
       - api
-
   vlan_id: 3000
   subnet: 13.13.1.0/24
   gateway: 13.13.1.1
   pool:
     # specify the pool range in form of <start_ip> to <end_ip>, IPs without the "to"
     # will be treated as an individual IP and will be used for configuring
       - 13.13.1.11 to 13.13.1.200

# optional, required if managemen_ipv6 is defined at server level
ipv6_gateway: 2001:c5c0:1234:5678:1002::1
ipv6_subnet: 2001:c5c0:1234:5678:1002::/80
ipv6_pool: ['2001:c5c0:1234:5678:1002::11 to 2001:c5c0:1234:5678:1002::20']

   segments: #management and provisioning will always be the same
      - management
      - provision

# OVS-VLAN requires VLAN-id as "None"
# LinuxBridge-VXLAN requires valid VLAN-id
-
   vlan_id: <vlan_id or None>
   subnet: 14.13.1.0/24
   gateway: 14.13.1.1
   pool:
     - 14.13.1.11 to 14.13.1.254
   segments:
    - tenant
-
   vlan_id: 3005
   subnet: 15.13.1.0/24
   gateway: 15.13.1.1
   pool:
     - 15.13.1.11 to 15.13.1.254
   segments:
     - storage

# optional network "external"
-
vlan_id: <108>
```

```
   segments:
     - external

# optional network "provider"; None for C-series, vlan range for B-series
-
vlan_id: "<None or 3200-3210>"
   segments:
    - provider
```

Define Server Roles

In the Roles section, add the hostname of the servers and their corresponding roles. In the case of micro-pod, specify the same server names under control, compute, and ceph. Also, the number of servers under each role has to be three for micro-pod. One can optionally expand the micro-pod, to include additional computes. In the case of HC (Hyperconverged deployment), all storage nodes will act as compute nodes, but not vice-versa.

```
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
object_storage:
networker:
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:

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

Note The maximum length of non-FQDN hostname is 32 characters. In this example, the length of Your_Controller_Server-1_HostName hostname is 33 characters. So, change the hostname length to 32 or less characters in both the ROLES and SERVERS section.

Cisco VIM introduces a new topology type called micro-pod 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. Starting Cisco VIM 2.2.2, we support the expansion of the micro-pod to accommodate more number of compute nodes. Each cloud application can decide the type of pod needed based on their resource (mem, storage consumption) requirements. In Cisco VIM Release2.2, the micro-pod option supports only OVS/VLAN or VPP/VLAN with Cisco-VIC or Intel 710 NICon a specific BOM. Also, ACI/VLAN is supported on micro-pod with Cisco-VIC.

To enable the micro-pod option, update the setup_data as follows:

```
PODTYPE: micro
```

In the same vein, if customers want to converge compute and storage nodes together, in Cisco VIM Release2.2, the hyper-convergence (UMHC) option is supported as well. The UMHC option supports only OVS/VLANwith a combination of Cisco-VIC and Intel 520 NIC on a specific BOM.

To enable the hyper convergence (UMHC) option, update the setup_data as follows:

PODTYPE: UMHC

Define Servers - C-Series Pod Example

Note The UCS C-Series maximum host name length is 32 characters.

```
SERVERS:
Your_Controller_Server-1_HostName:
cimc_info: {'cimc_ip': '172.22.191.36'}
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
#management_ipv6: 2001:c5c0:1234:5678:1002::12        <== optional, allow manual static
      IPv6 addressing
#cimc username, password at a server level is only needed if it is different from the one
defined in the CIMC-COMMON section
Your_Controller_Server-2_HostName:
cimc_info: {'cimc_ip': '172.22.191.37', 'cimc_username': 'admin','cimc_password': 'abc123'}
rack_info: {'rack_id': 'RackB'}

Your_Controller_Server-3_HostName:
cimc_info: {'cimc_ip': '172.22.191.38'}
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': '172.22.191.40'}
rack_info: {'rack_id': 'RackA'}
hardware_info: {'VIC_slot': '3'} # optional only if the user wants a specific VNIC to be
chosen

.. .. similarly add more computes and 3 storage info
```

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.

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,
has to be defined for all nodes
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 that controller Rack IDs 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 controllers to provide for future rack redundancy. Compute and storage nodes have no Rack ID restrictions.

Multiple VLAN Trunking with SRIOV using UCSM for UCS B-Series Pods

Some NFV solutions require the guest VM single root I/O virtualization (SRIOV) to send and receive VLAN tagged packets. Because the UCSM plugin in Cisco VIM creates the SR-IOV ports and attaches them to the guest VM, the port must be brought up in trunk mode. To support this, special network names are provided to the UCSM plugin at initialization. Each network supports a different set of application VLANs, which are included in the Cisco VIM configuration. When the port profile is created in UCSM, it checks to see if the port is created on one of the special neutron networks. If so, it adds the VLANs provided in the setup_data.yaml to the UCSM port profile. In effect, this allows the VM-FEX port to trunk all of the VLANs. A typical configuration example in setup_data is shown below. This is an optional feature which, by default, is not enabled. If it is not enabled, the section shown below is absent. SRIOV with Multi-VLAN trunking is only available in the UCS B-Series pod enabled with UCSM plugin.

```
SRIOV_MULTIVLAN_TRUNK:
     - network_name1: 124, 2:3,9:13
     - network_name2: 4, 5:7, 8
#all the vlans listed are unique in the entire setup_data.yaml
```

Setting Up the Cisco VIM OpenStack Configurations

The following sections provide examples of Cisco VIM OpenStack configurations 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 ip address in API nework
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: 2001:c5c0:1234:5678:1002::10   <== optional, enable IPv6 for
 OpenStack admin endpoint
```

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
#external_lb_vip_fqdn: <host or DNS name matching external_lb_vip_address>
#external_lb_vip_ipv6_address: 2001:c5c0:1234:5678:1003::10   <== optional, enable IPv6 for
 OpenStack public endpoint
VIRTUAL_ROUTER_ID: <vrrp router id, eg:49>
```

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 should be 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 will create 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)
external_lb_vip_ipv6_address: 2001:c5c0:1234:5678:1003::10   <== optional, enable IPv6 for
 OpenStack public endpoint
```

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

For OpenStack Glance, the 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 the setup_data.yaml file.

```
STORE_BACKEND: ceph #supported as 'ceph' for ceph backend store; don't change
```

OpenStack Cinder Configuration with Dedicated Ceph

For OpenStack Cinder, the OpenStack storage service, the dedicated Ceph object storage configuration is show 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
```

OpenStack Nova Configuration

To reduce the boot time, the NOVA_BOOT_FROM parameter is set to local for Cisco VIM in the OpenStack Newton release. While this reduces the boot time, it does not provide Ceph back end redundancy. To overwrite it, you can set NOVA_BOOT_FROM to **ceph**.

```
# Nova boot from CEPH
NOVA_BOOT_FROM: <ceph> #optional
```

OpenStack Neutron Configuration

OpenStack Neutron configuration is shown below.

```
# ML2 Conf – choose from either option 1 or option 2
# option 1: LinuxBridge-VXLAN
MECHANISM_DRIVERS: linuxbridge
TENANT_NETWORK_TYPES: "VXLAN"
Or
## option 2: OVS VLAN
MECHANISM_DRIVERS: openvswitch
TENANT_NETWORK_TYPES: "VLAN"
# VLAN ranges can be one continuous range or comma separated discontinuous ranges
TENANT_VLAN_RANGES: 3001:3100,3350:3400
# Jumbo MTU functionality. Only in B series, OVS-VLAN
# more info here [Mercury] Jumbo MTU feature in Mercury (B Series)
# ENABLE_JUMBO_FRAMES: True

# for Provider networks, just specifying the provider in the segments under
# the NETWORKING section is enough.
# Note : use phys_prov as physical_network name when creating a provider network
```

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 will be 9000 and VXLAN will be 8950.

Cisco VIM also supports the installation of a handful of optional services, namely, Keystone v3 and Heat. 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 Optional Services section in the setup_data.yaml file:

```
# 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, release 2.2.

To enhance the security portfolio and multi-tenancy with the use of domains, the Keystone v3 support is added in Cisco VIM release 2.2 from an authentication end-point. Keystone v2 and Keystone v3 are mutually exclusive; an administrator has to decide the authentication end-point during installation. By default, the VIM orchestrator picks keystone v2 as the authentication end-point.

To enable Keystone v3, add the following line under the optional services section.

```
# Optional Services:
OPTIONAL_SERVICE_LIST:
- keystonev3
```

LDAP/AD support with Keystone v3

With the introduction of Keystone v3, the OpenStack service authentication can now be delegated to an external LDAP/AD server. In Cisco VIM 2.2, this feature has been introduced optionally if the authorization is done by Keystone v3.

The pre-requisite for enabling LDAP/AD integration is that the LDAP/AD endpoint should be reachable from all the Controller nodes that run OpenStack Keystone Identity Service.

To avail the LDAP/AD support with Keystone v3 feature, add the following section to the setup_data 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: '<ldaps|ldap>://<fqdn|ip-address>:[port]'
            e.g. 'ldap://172.26.233.104:389'
            e.g. 'ldap://172.26.233.104'
            e.g. 'ldaps://172.26.233.104'
            e.g. 'ldaps://172.26.233.104:636'
            e.g. 'ldap://fqdn:389'
            e.g. 'ldap://fqdn'
            e.g. 'ldaps://fqdn'
            e.g. 'ldaps://fqdn:636'
         '<ldaps|ldap>://[<ip6-address>]:[port]'
           e.g. ldap://[2001:420:293:2487:d1ca:67dc:94b1:7e6c]:389 ---> note the mandatory
```

```
"[.. ]" around the ipv6 address

   user: '<DN of bind user>' # e.g. 'dc=admin,dc=cisco,dc=com'
   password: <password> # e.g. password of bind user
   user_filter = (memberOf=CN=os-users,OU=OS-Groups,DC=mercury,DC=local)
   user_id_attribute = sAMAccountName
   user_name_attribute = sAMAccountName
   user_mail_attribute = mail             # Optional
   group_tree_dn = ou=OS-Groups,dc=mercury,dc=local
   group_name_attribute = sAMAccountName
```

Note The parameter values differ based on the Directory Service provider. For Example, OpenLDAP or Microsoft Active Directory.

Integrating identity with LDAP/AD 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*.

The url parameter supports the following format:

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

The protocol can be one of the following: ldap for non-ssl and ldaps when TLS has to be enabled.

The ldap host can be a fully-qualified domain name (FQDN) or an IPv4 or v6 address depending on how the SSL certificates are generated.

The port number is optional. If the port number is not provided, the ldap services are assumed to be 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.

OpenStack Object Storage integration with Cisco VIM

Cisco VIM supports automated integration with a customer-managed object storage solution. The integration points reside primarily in the OpenStack Identity (Keystone) component of Cisco VIM. In the Cisco VIM 2.2, this integration is restricted to Keystone v2 only. It currently integrates with SwiftStack as the choice of object storage solution. The deployment assumes a customer-managed SwiftStack solution. Installation of the SwiftStack Controller/PACO cluster is out of scope of this document and customer should reach out to the SwiftStack team for license and installation details. While OpenStack can support multiple endpoints for a given object-store service, the current setup in the context of automation supports a single Keystone object-store service per SwiftStack PACO cluster endpoint.

The current automation uses the admin role for authentication and authorization of SwiftStack users between the Keystone SwiftStack tenant and SwiftStack account.

Pre-requisites

For a customer-managed deployment model, the minimum pre-requisites are:

- You must have a SwiftStack controller, Cluster deployed with appropriate PAC (Proxy/Account/Container) and Object configured ahead of time.
- You must know the Swift endpoint of the PAC outward facing IP address, the corresponding admin user, password and service tenant information at the time of configuring Keystone integration.
- The networking should be configured in such a way that the PAC outward facing IP address and the pod API network can talk to each other.The Keystone Auth and Keystone Auth Token middleware must be pre-configured in SwiftStack (see Keystone Configuration Requirements in SwiftStack, on page 22)

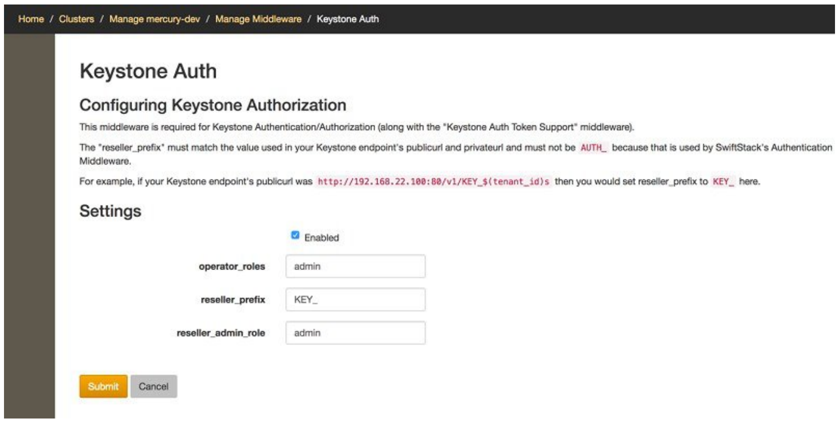
The OpenStack controllers must have network reachability to the SwiftStack API endpoints, so that the Horizon and Cinder Backup service can talk to the SwiftStack endpoints.

Keystone Configuration Requirements in SwiftStack

To configure Keystone authorization, from the SwiftStack controller, choose the **Cluster** > **Manage** > **Middleware** > **Keystone Auth** option.

Note The reseller_prefix setting enables the Keystone Auth middleware invocation at the time of authentication.

Figure 2: Configuring Keystone


Home / Clusters / Manage mercury-dev / Manage Middleware / Keystone Auth
Submit
Cancel

Keystone Auth

Configuring Keystone Authorization

This middleware is required for Keystone Authentication/Authorization (along with the "Keystone Auth Token Support" middleware).

The "reseller_prefix" must match the value used in your Keystone endpoint's publicurl and privateurl and must not be `AUTH_` because that is used by SwiftStack's Authentication Middleware.

For example, if your Keystone endpoint's publicurl was `http://192.168.22.100:80/v1/KEY_$(tenant_id)s` then you would set reseller_prefix to `KEY_` here.

Settings

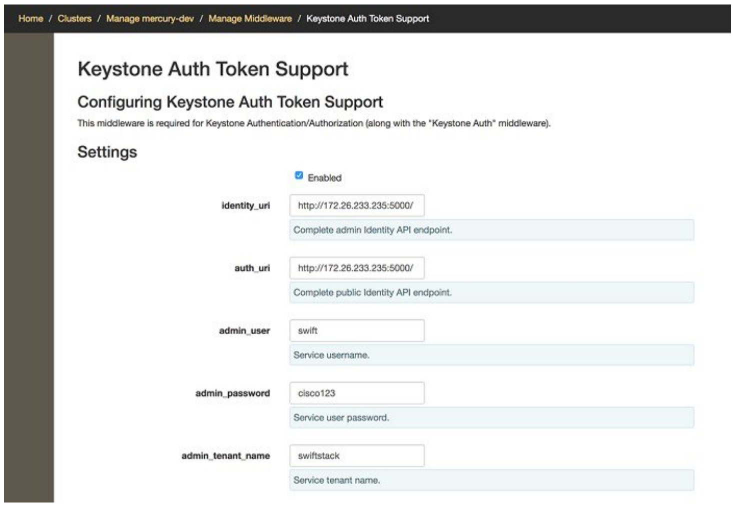
[✓] Enabled

operator_roles	admin
reseller_prefix	KEY_
reseller_admin_role	admin

To configure Keystone Auth Token Support, from the SwiftStack controller, choose the **Cluster** > **Manage** > **Middleware** > **Keystone Auth Token Support** option.

Note auth_uri is deprecated.

Figure 3: Keystone Auth


Home / Clusters / Manage mercury-dev / Manage Middleware / Keystone Auth Token Support

Keystone Auth Token Support

Configuring Keystone Auth Token Support

This middleware is required for Keystone Authentication/Authorization (along with the "Keystone Auth" middleware).

Settings

[✓] Enabled

identity_uri	http://172.26.233.235:5000/
	Complete admin Identity API endpoint.
auth_uri	http://172.26.233.235:5000/
	Complete public Identity API endpoint.
admin_user	swift
	Service username.
admin_password	cisco123
	Service user password.
admin_tenant_name	swiftstack
	Service tenant name.

Usage in Cisco VIM

In order to support SwiftStack endpoint configuration, the following section needs to be configured in the setup_data.yaml file.

```
##########################################
# Optional Swift configuration section
##########################################
# SWIFTSTACK: # Identifies the objectstore provider by name
#   cluster_api_endpoint: <IP address of PAC (proxy-account-container) endpoint>
#   reseller_prefix: <Reseller_prefix configured in Swiftstack Keystone middleware E.g KEY_>
#   admin_user: <admin user for swift to authenticate in keystone>
#   admin_password: <swiftstack_admin_password>
#   admin_tenant: <The service tenant corresponding to the Account-Container used by
Swiftstack>
#   protocol: <http or https> # protocol that swiftstack is running on top
```

The automation supports two modes of Integration with SwiftStack- Integration during fresh installation of the pod and a reconfigure option to add a SwiftStack endpoint to an existing pod running Cisco VIM 2.2.

In the fresh installation mode, the addition of the Optional Swift configuration section in the setup_data.yaml file will automatically provision the following in Keystone:

- Keystone service for Object Store.
- Keystone endpoints for the Object Store service.
- A SwiftStack admin user with admin role in a SwiftStack tenant.

Integration Testing: In order to test if the Keystone integration has been successful, request a token for the configured swift user and tenant.

The output must contain a properly generated endpoint for the object-store service that points to the SwiftStack PAC cluster endpoint with the expected "reseller_prefix".

For example:

KEY_curl -d '{"auth":{"passwordCredentials":{"username": "<username>", "password": "<password>"},"tenantName":"<swift-tenant>"}}' -H "Content-type: application/json" < OS_AUTH_URL >/tokens

The output should list endpoints generated by Keystone for the object-store cluster endpoint of SwiftStack for the user tenant (SwiftStack account).

A sample output snippet (all IP and Keys are just examples, they will vary from pod to pod):

```
  {
      "access": {
          "metadata": {
              "is_admin": 0,
              "roles": [
                  "33f4479e42eb43529ec14d3d744159e7"
              ]
          },
          "serviceCatalog": [
          {
                          "endpoints": [
                              {
                                  "adminURL": "http://10.30.116.252/v1",
                                  "id": "3ca0f1fee75d4e2091c5a8e15138f78a",
                                  "internalURL":
"http://10.30.116.252/v1/KEY_8cc56cbe99ae40b7b1eaeabb7984c77d",
                                  "publicURL":
"http://10.30.116.252/v1/KEY_8cc56cbe99ae40b7b1eaeabb7984c77d",
                                  "region": "RegionOne"
                              }
                          ],
                          "endpoints_links": [],
                          "name": "object-store",
                          "type": "object-store"
                      },
          ......
```

Verify that the Keystone user has access to the SwiftStack cluster. Using the token generated preceding for the swiftstack user and tenant, make a request to the SwiftStack cluster:

```
curl -v -H  "x-auth-token: <auth-token>"
http://10.30.116.252/v1/KEY_8cc56cbe99ae40b7b1eaeabb7984c77d
```

This command displays all the containers (if present) for the SwiftStack tenant (account).

Integrating SwiftStack over TLS

Integrating SwiftStack over TLS: The automation supports SwiftStack integration over TLS. To enable TLS, the CA root certificate must be presented as part of the /root/openstack-configs/haproxy-ca.crt file. The **protocol** parameter within the SWIFTSTACK stanza must be set to **https.** As a pre-requisite, the SwiftStack cluster has to be configured to enable HTTPS connections for the SwiftStack APIs with termination at the proxy servers.

Cinder Volume Backup on SwiftStack

Cisco VIM, enables cinder service to be configured to backup its block storage volumes to the SwiftStack object store. This feature is automatically configured if the SWIFTSTACK stanza is present in the setup_data.yaml file. The mechanism to authenticate against SwiftStack during volume backups leverages the same keystone SwiftStack endpoint configured for use to manage objects. The default SwiftStack container to manage cinder volumes within the Account (Keystone Tenant as specified by "admin_tenant") is currently defaulted to **volumebackups**
Once configured, cinder backup service is enabled automatically as follows:

```
cinder service-list
+------------------+----------------+------+---------+-------+----------------------------+-----------------+
| Binary           | Host           | Zone | Status  | State | Updated_at
| Disabled Reason     |
+------------------+----------------+------+---------+-------+----------------------------+-----------------+
| cinder-backup    | c43b-control-1 | nova | enabled | up    | 2017-03-27T18:42:29.000000
| -                |
| cinder-backup    | c43b-control-2 | nova | enabled | up    | 2017-03-27T18:42:35.000000
| -                |
| cinder-backup    | c43b-control-3 | nova | enabled | up    | 2017-03-27T18:42:33.000000
| -                |
| cinder-scheduler | c43b-control-1 | nova | enabled | up    | 2017-03-27T18:42:32.000000
| -                |
| cinder-scheduler | c43b-control-2 | nova | enabled | up    | 2017-03-27T18:42:32.000000
| -                |
| cinder-scheduler | c43b-control-3 | nova | enabled | up    | 2017-03-27T18:42:31.000000
| -                |
| cinder-volume    | c43b-control-1 | nova | enabled | up    | 2017-03-27T18:42:35.000000
| -                |
| cinder-volume    | c43b-control-2 | nova | enabled | up    | 2017-03-27T18:42:30.000000
| -                |
| cinder-volume    | c43b-control-3 | nova | enabled | up    | 2017-03-27T18:42:32.000000
| -                |
+------------------+----------------+------+---------+-------+----------------------------+-----------------+
```

Backing up of an existing cinder volume.

```
openstack volume list
+--------------------------------------+--------------+-----------+------+-------------+
| ID                                   | Display Name | Status    | Size | Attached to |
+--------------------------------------+--------------+-----------+------+-------------+
| f046ed43-7f5e-49df-bc5d-66de6822d48d | ss-vol-1     | available |    1 |             |
+--------------------------------------+--------------+-----------+------+-------------+

openstack volume backup create f046ed43-7f5e-49df-bc5d-66de6822d48d
+-------+--------------------------------------+
| Field | Value                                |
+-------+--------------------------------------+
| id    | 42a20bd1-4019-4571-a2c0-06b0cd6a56fc |
| name  | None                                 |
+-------+--------------------------------------+

openstack container show volumebackups
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
| account      | KEY_9d00fa19a8864db1a5e609772a008e94 |
| bytes_used   | 3443944                              |
| container    | volumebackups                        |
| object_count | 23                                   |
+--------------+--------------------------------------+

swift list volumebackups

volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00001
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00002
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00003
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00004
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00005
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00006
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00007
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00008
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00009
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00010
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00011
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00012
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00013
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00014
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00015
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00016
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00017
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00018
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00019
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00020
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc-00021
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc_metadata
volume_f046ed43-7f5e-49df-bc5d-66de6822d48d/20170327185518/az_nova_backup_42a20bd1-4019-4571-a2c0-06b0cd6a56fc_sha256file
```

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
```

Cisco VIM Configurations 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 the 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 has an info
#########################################
## Only 2 Values allowed is: 2M or 1G
#VM_HUGEPAGE_SIZE: 2M or 1G

## Percentagae 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 will set the ratio. By default, all VM memories (100%)
## will be huge pages.
## Only input of type integer is allowed, in the range of 0-100 (including 0 and 100)
#VM_HUGEPAGE_PERCENTAGE: 100
```

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 needs 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)
VTC_SSH_USERNAME: '<vtc_ssh_username>'   # Required parameter when VTS Day0 is enabled or
 running NFVbench and/or VMTP
VTC_SSH_PASSWORD: '<vtc_ssh_password>'   # Required parameter when VTS Day0 is enabled or
running NFVbench and/or VMTP

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

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 2.2 integrates the Opflex ML2 plugin (in Unified mode) to manage the tenant VLANs dynamically, as VMs come and go in the cloud. In addition, Cisco VIM 2.2 supports the administrator driven automated workflow to provison the provider networks. In VIM 2.2, this is supported on a C-series based Fullon or micropod running with Cisco VIC 1227.

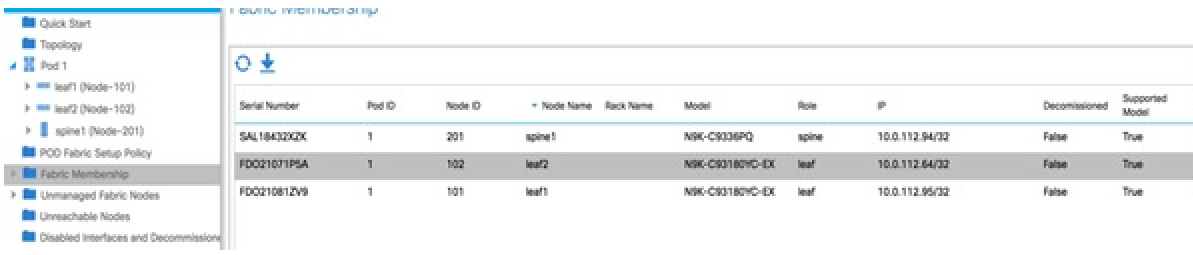
VIM orchestrator configures the day-0 aspects of the ACI fabric, along with the Opflex ML2 plugin integration. The only exception is the manual configuration of L3 out.

Before You Begin

As VIM 2.2, does the day-0 configuration of the ACI, following are the assumptions that VIM makes for the integration to happen.

- Before the VIM installation the APIC 3.0 controllers running in a cluster of three should 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
SAL18432XZK	1	201	spine1		N9K-C9336PQ	spine	10.0.112.94/32	False	True
FDO21071P5A	1	102	leaf2		N9K-C93180YC-EX	leaf	10.0.112.64/32	False	True
FDO21081ZV9	1	101	leaf1		N9K-C93180YC-EX	leaf	10.0.112.95/32	False	True

- Network should be designed such that the management node and controllers are reachable to APIC controllers.
- ACIINFRA a new networking segment is introduced for ACI policy management; ACIINFRA segment should not overlap with the VLANID across the infrastructure
- Tunnel end point address pool (TEP_ADDRESS_POOL) is set to ACI default at 10.0.0.0/16; care should be taken not to assign this address space anywhere else in the cloud.
- Multicast address pool is set to ACI default at 225.0.0.0/15; care should be taken not to assign this address space anywhere else in the cloud.
- ACIINFRA VLANID, the TEP_ADDRESS_POOL, and the multicast address pool are immutable for the lifecycle of the infrastructure.
- Pre-provision of L3 out API network is done before the VIM install as listed:
    - Create installer tenant and VRF and provide the name of it in setup_data
    - Create L3out routed outside object and provide its name in the setup_data
    - Ensure, that this api-l3out must be associated to the tenant VRF.

Note The L3-out object for OpenStack API network needs to be consistently named tha is Name of the L3 Outside object must be the same as the name provided for its corresponding External Network Instance Profile. Example: if you provide api_l3out_network: api-l3out in setup_data, then your dn for the api network should resolve to something like the following: cvim-installer-tenant|uni/tn-cvim-installer-tenant/out-api-l3out/instP-api-l3out.

Note By default optimised DHCP and optimised metadata services are deployed with ACI integration.

Note The plugin automation configures DHCP and Metadata agents in optimized mode. There is no option provided in setup_data to change that setting in the current implementation.

Run the following setup_data in the VIM to add a new APICINFO:

```
APICINFO:
    apic_hosts: '<ip1|host1>:[port], <ip2|host2>:[port], <ip3|host3>:[port]'   # max of 3, min of
1, not 2; reconfigurable
    apic_username:        # common across the 3;
    apic_password:        # common across the 3;
    apic_system_id:       # string max length of 8
    apic_resource_prefix: string    e.g. cvim-1   # max length of 6
    apic_tep_address_pool: 10.0.0.0/16   # static today
    multicast_address_pool: 225.0.0.0/15   # static, today
    apic_pod_id: <int>    #All(int, Range(min=1, max=65535)),
    apic_installer_tenant:  # String, max length 32
    apic_installer_vrf:    # string (max length32) this is the VRF which is associated with the
pre-provisioned API L3out
    api_l3out_network:       #  String, max length 32
# mgmt_l3out_network: # String, max length 32 (optional)
NOTE: mgmt_l3out_network and mgmt_l3out_vrf MUST coexist together if defined
# mgmt_l3out_vrf: # String, max length 32 (optional)
 NOTE: mgmt_l3out_network and mgmt_l3out_vrf MUST coexist together if defined
```

As the APIC manages the Leaf switches, its mandatory to define the Leaf switches in the following format:

```
TORSWITCHINFO:  (mandatory)

  SWITCHDETAILS:
   -
   hostname: <leaf-hostname-1>
   vpc_peer_keepalive: <leaf-hostname-2>
   vpc_domain: 1  # Must be unique across pairs
   br_mgmt_port_info: 'eth1/27'  # br_mgmt_* attributes must exist on at least one pair
   br_mgmt_vlan_info: '3401'
   node_id: <int> # unique across switches

   -
   hostname: <leaf-hostname-2>
   vpc_peer_keepalive: <leaf-hostname-1>
   vpc_domain: 1
   br_mgmt_port_info: 'eth1/27'  # br_mgmt_* attributes must exist on at least one pair
   br_mgmt_vlan_info: '3401'
   node_id: <int> # unique across switches
   -
   hostname: <leaf-hostname-3>
   vpc_peer_keepalive: <leaf-hostname-4>
   vpc_domain: 2  # Must be unique across pairs
   node_id: <int> # unique across switches
   -
   hostname: <leaf-hostname-4>
   vpc_peer_keepalive: <leaf-hostname-3>
   vpc_domain: 2
   node_id: <int> # unique across switches
  -
   hostname: <leaf-hostname-5>
   node_id: <int> # unique across switches
   br_mgmt_port_info: 'eth1/27, eth1/30'  # br_mgmt_* attributes must exist on at least one pair,
only if info is not in peer
   br_mgmt_vlan_info: '3401'
```

Also as CVIM 2.2 orchestrator is doing the day-0 configuration of the ACI, the SERVERS section of the setup_data needs to be augmented to include the server and the switch port associations as listed below:

```
  c32-control-1.cisco.com:
    cimc_info: {cimc_ip: 172.26.229.67}
    management_ip: 192.168.37.17
    rack_info: {rack_id: RackC}
    tor_info: {<leaf-hostname-1>: eth1/15, <leaf-hostname-2>: eth1/15}
  c32-control-2.cisco.com:
    cimc_info: {cimc_ip: 172.26.229.68}
    management_ip: 192.168.37.18
    rack_info: {rack_id: RackC}
    tor_info: {<leaf-hostname-1>: eth1/16, <leaf-hostname-2>: eth1/16}
  c32-control-3.cisco.com:
    cimc_info: {cimc_ip: 172.26.229.69}
    management_ip: 192.168.37.19
    rack_info: {rack_id: RackC}
    tor_info: {<leaf-hostname-1>: eth1/17, <leaf-hostname-2>: eth1/17}
  c32-compute-1.cisco.com:
    cimc_info: {cimc_ip: 172.26.229.70}
    management_ip: 192.168.37.20
    rack_info: {rack_id: RackC}
    tor_info: {<leaf-hostname-3>: eth1/15, <leaf-hostname-4>: eth1/15}

Addtionally the mechanism_driver needs to be “aci” and ACINFRA section needs to be defined in the
networks section.

MECHANISM_DRIVERS: aci TENANT_NETWORK_TYPES: "VLAN"
TENANT_VLAN_RANGES: <START>:<END> # arbitrary VLAN range*** NFV
```

Networking Parameters

```
NETWORKING:
  networks:
  - segments: [aciinfra]
    vlan_id: user_defined_unique_vlan_id. This vlan should not overlap with any of the vlans defined
 in setup data; new item
other segments same as OVS/VLAN.
```

Note Refer to the ACI documentation for usage of L3out external network that is consumed by VMTP below. Also, ensure that the L3 out routed configuration is provisioned in the ACI "common" tenant.

We support execution of VMTP for external network with ACI in place. For the VMTP the NET_NAME key for EXT_NET needs to match the name of the L3out for external network

```
  VMTP_VALIDATION:
EXT_NET:
NET_NAME: <name of L3out for the external network>
```

Support for Provider Networks in ACI

OpFlex plugin integration (3.0) does not currently support a fully automated workflow to provision Provider Networks in neutron. CVIM has provided a utility that will support provisioning neutron provider networks.

- After the installer has completed deployment, ensure that Fabric Access policies for the external link from the border leaf switches have been created manually. This is the link that will carry the L2 traffic between the external ToRs and the border leaf switches. These may be configured as desired (direct PC, PC or VPC). This is typically a one-time admin setup.
- Create a neutron network and subnet in the OpenStack tenant as usual. This is the provider subnet that will be carried through L2 into the fabric. Do not provide segmentation_id. Enable DHCP.
- Run the following command to provision the provider network in ACI:

  ```
  cd installer-<tagid>/tools
  ./apic_create_provider_net.py –netid <neutron-provider-net-id> --staticpath
  <path-to-external-interface-on-borderleaf>
   --segmentationid <vlan-id> --tenantid <openstack-tenant-id>
  ```

Enabling NFVBench on Cisco VIM

This section describes how to setup and use NFVBench with Cisco VIM.

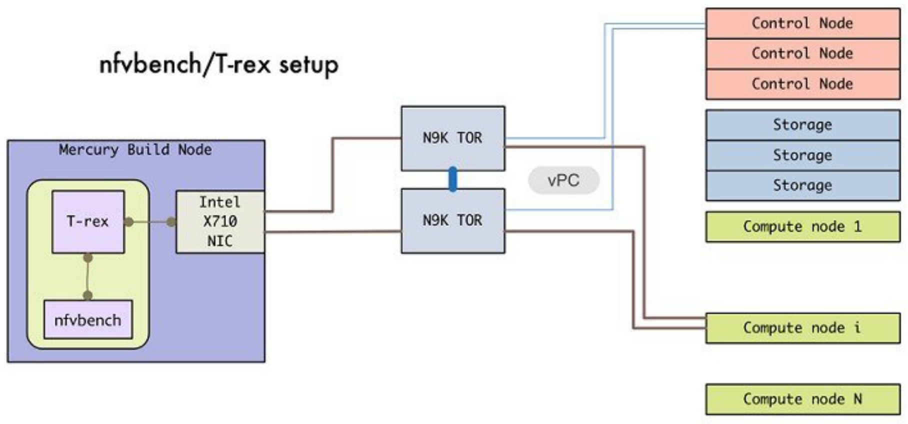
Once the pre-requisites for the management node hardware (Intel NIC) are met, add the NFVBench configurations in the setup_data.yaml. By default, NFVBench configuration is not enabled in Cisco VIM 2.0.

Before You Begin

- NFVBench offering in Cisco VIM, requires a 10GE Intel NIC (Intel X710 NIC (4 x 10G) or Intel-520 (2x10G)) 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 4: 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
   vtep_vlans: vlan_id1,vlan_id2 # mandatory only when mechanism driver is VTS, or tenant type is
VXLAN
#  nic_ports: int1,int2             # Optional input, indicates which 2 of the 4 ports in the 10 G
intel NIC ports on the on the management node is the    NFVBENCH tool using to send and receive
```

```
traffic. If nothing is specified, the tool assumes its Port 1,2 i.e. the first 2 ports that will be
 us
# Please refer to the VTS_PARAMETERS and TORSWITCHINFO if NFVbench is enabled
# Required when mechanism driver is VTS
VTS_PARAMETERS:
   …
   VTS_NCS_IP: '<vtc_ssh_username>'   # Required parameter when VTS enabled
   VTC_SSH_USERNAME: '<vtc_ssh_username>'   # mandatory for NFVbench
   VTC_SSH_PASSWORD: '<vtc_ssh_password>'   # mandatory for NFVbench
# 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 <port-channela>
  switchport mode trunk
  switchport trunk allowed vlan <3000-3049>
interface Ethernetx/y
  switchport mode trunk
  switchport trunk allowed vlan <3000-3049>
  channel-group <a>
```

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 2.2 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.
```

Note NFV Host configuration does not support micro-pod.

Install Mode

Cisco VIM can be deployed 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. Dis-connected: 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 one 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 also the OpenStack instances. The monitoring system is designed such that it can monitor single or multiple pods from a single management system. NFVIMON is enabled by extending the setup_data.yaml file with relevant information. Also, NFVIMON can be enabled on an existing pod, through the reconfigure option. Then, the pod is added as a VIM resource to be monitored in a Control Center.

NFVIMON consists of four components: dispatcher, 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 minimal software piece (dispatcher) 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 current 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.

In terms of networking, the collectors VMs need to have two interfaces: an interface in br_mgmt of the VIM, and another interface that is routable, which can reach the VIM Installer REST API and the RM VMs. As the collector VM is in an independent node, four IPs from the management network of the pod should be pre-planned and reserved. Install steps of the collector, resource manager (RM) and control-center with Cisco Zenpacks (CZ) are Cisco advance services activities.

Installation of NFVIMON Dispatcher

The dispatcher is the only component in NFVIMON that is managed by VIM orchestrator. While the dispatcher acts as a conduit to pass OpenStack information of the pod to the collectors, it is the Cisco Zenpack sitting in the controller node, that gathers the node level information.

To enable dispatcher as part of the VIM Install, 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
  DISPATCHER:
    rabbitmq_username: admin  # Pod specific user for dispatcher module in
ceilometer-collector
```

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
      ....
```

Note TORSWITCH monitoring is disabled when running Cisco VIM with ACI plugin enabled.

Enabling or Disabling Autobackup of Management Node

Cisco VIM 2.2 introduces 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>
```

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 an external syslog server.

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
protocol  : udp # defaults to udp
facility  : <string> # required; possible values local[0-7]or user
severity  : <string; suggested value: debug>
port      : <int>; # defaults, to 514
clients   : 'ELK' # defaults to ELK;
# Please note other than the remote host info, most of the other info is not needed; Also
the client list in 2.2 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.

Configuring Additional VIM Administrators

Cisco VIM supports management of VIM Administrators. VIM administrator has the permissions to login to the management through SSH or the console using the configured password . Administrators have there own accounts.

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 will have access to a shell with system privileges on the management node.

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 then 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.

To prevent double faults, a cloud sanity check is done before the update is started, and another cloud sanity check is performed at the end of the update.

To complete the software update, perform the Installing Cisco VIM, on page 2. If your management node does not have Internet, complete the Preparing to Install Cisco NFVI on Management Nodes Without Internet Access 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 the software update, you must commit it before you can 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 about updating the Cisco NFVI software, see the Managing Cisco NFVI chapter in the Cisco NFV Infrastructure Administrator Guide, Release 2.2

Upgrading Cisco NFVI Software

Cisco VIM's design allows the graceful upgrade of a cloud from version 1.0 (liberty based) to 2.2 (newton based). The seamless process upgrades both OpenStack and infrastructure services to the newer version. As the upgrade involves moving the kernel version (from RHEL 7.2 to 7.4), proper down-time should be planned to upgrade the VIM cloud. The upgrade cause limited service impact, critical components such as controller and storage nodes are upgrade serially, whereas compute nodes are upgraded in a bulk-and-batch manner.

As the OpenStack does not support the skipping of major releases during upgrade from liberty to newton, the VIM upgrade orchestrator internally moves the stack to mitaka as an intermediate step. As part of the upgrade, the REST API server managing the VIM orchestrator also gets upgraded. A script called vim_upgrade_orchestrator.py is used to upgrade the cloud. Also, as part of the upgrade, automatic translation (from Liberty to Newton) of the setup_data.yaml happens so that it is compatible to the target release (2.2.0) version.

Note After you complete the software upgrade you will not be able to roll back to the prior release. During software upgrade all pod management operations are blocked.

For information about upgrading the Cisco NFVI software, see the*"Managing Cisco NFVI" chapter in the Cisco NFV Infrastructure Administrator Guide, Release2.2.*