Cisco Virtualized Infrastructure Manager Administrator Guide, Release 2.2.21

First Published: 2018-04-17

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

CONTENTS

CHAPTER 1

Managing Cisco NFVI

The following topics provide general management procedures that you can perform if your implementation is Cisco VIM by itself or is Cisco VIM and Cisco VIM Insight.

Managing Cisco NFVI Pods

You can perform OpenStack management operations on Cisco NFVI pods including addition and removal of Cisco NFVI compute and Ceph nodes, and replacement of controller nodes. Each action is mutually exclusive. Only one pod management action can be performed at any time. Before you perform a pod action, verify that the following requirements are met:

- The node is part of an existing pod.

- The node information exists in the setup_data.yaml file, if the pod management task is removal or replacement of a node.
- The node information does not exist in the setup_data.yaml file, if the pod management task is to add a node.

  To perform pod actions, see the Managing Hosts in Cisco VIM or NFVI Pods , on page 4section.

General Guidelines for Pod Management

The setup_data.yaml file is the only user-generated configuration file that is used to install and manage the cloud. While many instances of pod management indicates that the setup_data.yaml file is modified, the administrator does not update the system generated setup_data.yaml file directly.

Note To avoid translation errors, we recommend that you avoid copying and pasting commands from the documents to the Linux CLI.

To update the setup_data.yaml file, do the following:

1. Copy the setup data into a local directory:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
```

2. Update the setup data manually:

```
[root@mgmt1 ~]# vi my_setup_data.yaml (update the targeted fields for the setup_data)
```

3. Run the reconfiguration command:

```
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml>
<pod_management_action>
```

In Cisco VIM, you can edit and enable a selected set of options in the setup_data.yaml file using the reconfigure option. After installation, you can change the values of the feature parameters. Unless specified, Cisco VIM does not support unconfiguring of the feature.

The following table summarizes the list of features that can be reconfigured after the installation of the pod.

Features enabled after post-pod deployment	**Comments**
Optional OpenStack Services	• Heat: OpenStack Orchestration Program • Keystone v3: Pod running Keystone v2 can be migrated to Keystone v3 • LDAP: Works only with Keystone v3. Full or partial reconfiguration can be done. All attributes except domain is reconfigurable.

Features enabled after post-pod deployment	Comments
Pod Monitoring	• NFVIMON: third party monitoring from host to service level; needs involvement and planning with Cisco Advance Services. It supports only Keystone v2 in VIM.
Export of EFK logs to External Syslog Server	Reduces single point of failure on management node and provides data aggregation.
NFS for Elasticsearch Snapshot	NFS mount point for Elastic-search snapshot is used so that the disk on management node does not get full.
Admin Source Networks	White list filter for accessing management node admin service.
NFVBench	Tool to help measure cloud performance. Management node needs a 10G Intel NIC (4x10G 710, or 2x10G 520 Intel NIC).
EFK settings	Enavles you to set EFK rotation frequency and size.
OpenStack service password	Implemented for security reasons, so that OpenStack passwords can be reset on-demand.
CIMC Password Reconfigure Post Install	Implemented for security reasons, so that CIMC passwords for C-series pod, can be reset on-demand.
SwiftStack Post Install	Integration with third-party Object-Store. The SwiftStack Post Install feature works only with Keystone v2.
TENANT_VLAN_RANGES and PROVIDER_VLAN_RANGES	Ability to increase the tenant and provider VLAN ranges on a pod that is up and running. It gives customers flexibility in network planning.
Support of Multiple External Syslog Servers	Ability to offload the OpenStack logs to an external Syslog server post-install.
Replace of Failed APIC Hosts and add more leaf nodes	Ability to replace Failed APIC Hosts, and add more leaf nodes to increase the fabric influence.
Make Netapp block storage end point secure	Ability to move the Netapp block storage endpoint from Clear to TLS post-deployment
Auto-backup of Management Node	Ability to enable/disable auto-backup of Management Node. It is possible to unconfigure the Management Node.
VIM Admins	Ability to configure non-root VIM Administrators.
EXTERNAL_LB_VIP_FQDN	Ability to enable TLS on external_vip through FQDN.
EXTERNAL_LB_VIP_TLS	Ability to enable TLS on external_vip through an IP address.

Features enabled after post-pod deployment	Comments
http_proxy and/or https_proxy	Ability to reconfigure http and/or https proxy servers.
Admin Privileges for VNF Manager (ESC) from a tenant domain	Ability to enable admin privileges for VNF Manager (ESC) from a tenant domain.
SRIOV_CARD_TYPE	Mechanism to go back and forth between 2-X520 and 2-XL710 as an SRIOV option in Cisco VIC NIC settings through reconfiguration.
NETAPP	Migrate NETAPP transport protocol from http to https.

Identifying the Install Directory

If the administrator is using CLI to manage the pod, the administrator must know the directory where the pod is installed from(refer to installer directory). To identify the installer directory of a pod, execute the following commands:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# ls –lrt | grep openstack-configs
lrwxrwxrwx.  1 root root      38 Mar 12 21:33 openstack-configs ->
/root/installer-<tagid>/openstack-configs
```

From the output, you can understand that the OpenStack-configs is a symbolic link to the installer directory.

Verify that the REST API server is running from the same installer directory location, by executing the following command:

```
# cd installer-<tagid>/tools
#./restapi.py -a status
Status of the REST API Server:   active (running) since Thu 2016-08-18 09:15:39 UTC; 9h ago
REST API launch directory: /root/installer-<tagid>/
```

Managing Hosts in Cisco VIM or NFVI Pods

To perform actions on the pod, run the commands specified in the following table. If you log in as root, manually change the directory to /root/installer-xxx to get to the correct working directory for these Cisco NFVI pod commands.

Table 1: Cisco NFVI Pod Management

Action	Steps	Restrictions

Remove block_storage or compute node	1. Remove the node information from the ROLES and SERVERS section of the setup_data.yaml file for the specific node.
	2. Enter one of the following commands.
	For compute nodes:
	`ciscovim remove-computes --setupfile ~/MyDir/my_setup_data.yaml <"compute-1,compute-2">`
	For storage nodes:
	`ciscovim remove-storage --setupfile ~/MyDir/my_setup_data.yaml <"storage-1">`
		You can remove multiple compute nodes and only one storage at a time;
		The pod must have a minimum of one compute and two storage nodes after the removal action.
		In Cisco VIM the number of ceph OSD nodes can vary from 3 to 20. You can remove one OSD node at a time as part of the pod management.
		Note On a micro-pod expanded with standalone computes, only the standalone compute nodes can be removed. Pod management operation for storage node is not supported for micro-pod. In hyper-converged mode, compute management operations are not supported for hyper-converged nodes.

Add block_storage or compute node	1. Add the node information from the ROLES and SERVERS section of the setup_data.yaml file for the specific node.
	2. Enter one of the following commands.
	For compute nodes:
	`ciscovim add-computes --setupfile ~/MyDir/my_setup_data.yaml <"compute-1,compute-2">`
	For storage nodes:
	`ciscovim add-storage --setupfile ~/MyDir/my_setup_data.yaml <"storage-1">`
		You can add multiple compute nodes and only one storage node at a time.
		The pod must have a minimum of one compute, and two storage nodes before the addition action.
		In Cisco VIM the number of ceph OSD nodes can vary from 3 to 20. You can add one OSD node at a time as part of the pod management.
		Note On a micro-pod expanded with standalone computes, only the standalone compute nodes can be added. Pod management operation for storage node is not supported for micro-pod. In hyper-converged mode, compute management operations are not supported for hyper-converged nodes.

Action	Steps	Restrictions
Replace controller node	1. If the controller node is in a UCS C-Series pod, update the CIMC info node in the SERVERS section of the setup_data.yaml file for the specific node 2. For B-series only update the blade and chassis info 3. Enter the following command: `ciscovim replace-controller --setupfile ~/MyDir/my_setup_data.yaml <"control-1">`	You can replace only one controller node at a time. The pod can have a maximum of three controller nodes. In Cisco VIM the replace controller node operation is supported in micro-pod. **Note** While replacing the controller node, the IP address and hostname are reused. So, do not update any other controller information other than CIMC access for C-series, and blade and chassis information for B-series. For micro-pod, this operation is supported on the AIO (all in one) nodes.

When you add a compute or storage node to a UCS C-Series pod, you can increase the management/provision address pool. Similarly, for a UCS B-Series pod, you can increase the Cisco IMC pool to provide routing space flexibility for pod networking. Along with server information, these are the only items you can change in the setup_data.yaml file after the pod is deployed. To make changes to the management or provisioning sections and/or CIMC (for UCS B-Series pods) network section, you must not change the existing address block as defined on day 0. You can add only to the existing information by adding new address pool block(s) of address pool as shown in the following example:

```
NETWORKING:
  :
  :

  networks:
  -
    vlan_id: 99
    subnet: 172.31.231.0/25
    gateway: 172.31.231.1
    ## 'pool' can be defined with single ip or a range of ip
    pool:
      - 172.31.231.2, 172.31.231.5 -→ IP address pool on Day-0
      - 172.31.231.7 to 172.31.231.12 -→ IP address pool ext. on Day-n
      - 172.31.231.20
    segments:
    ## CIMC IP allocation. Needs to be an external routable network
      - cimc
  -
    vlan_id: 2001
    subnet: 192.168.11.0/25
    gateway: 192.168.11.1
    ## 'pool' can be defined with single ip or a range of ip
    pool:
      - 192.168.11.2 to 192.168.11.5   -→ IP address pool on Day-0
      - 192.168.11.7 to 192.168.11.12  → IP address pool on day-n
      - 192.168.11.20 → IP address pool on day-n
    segments:
    ## management and provision goes together
      - management
```

```
   - provision
  :
  :
```

The IP address pool is the only change allowed in the networking space of the specified networks management/provision and/or CIMC (for B-series). The overall network must have enough address space to accommodate for future enhancement on day-0. After making the changes to servers, roles, and the corresponding address pool, you can execute the add compute/storage CLI shown above to add new nodes to the pod.

Recovering Cisco NFVI Pods

This section describes the recovery processes for Cisco NFVI control node and the pod that is installed through Cisco VIM. For recovery to succeed, a full Cisco VIM installation must have occurred in the past, and recovery is caused by a failure of one or more of the controller services such as Rabbit MQ, MariaDB, and other services. The management node must be up and running and all the nodes must be accessible through SSH without passwords from the management node. You can also use this procedure to recover from a planned shutdown or accidental power outage.

Cisco VIM supports the following control node recovery command:

```
# ciscovim cluster-recovery
```

The control node recovers after the network partition is resolved.

Note It may be possible that database sync between controller nodes takes time, which can result in cluster-recovery failure. In that case, wait for some time for the database sync to complete and then re-run cluster-recovery.

To make sure Nova services are good across compute nodes, execute the following command:

```
# source /root/openstack-configs/openrc
# nova service-list
```

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

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

In case of a complete pod outage, you must follow a sequence of steps to bring the pod back. The first step is to bring up the management node, and check that the management node containers are up and running using the docker ps –a command. After you bring up the management node, bring up all the other pod nodes. Make sure every node is reachable through password-less SSH from the management node. Verify that no network IP changes have occurred. You can get the node SSH IP access information from /root/openstack-config/mercury_servers_info.

Execute the following command sequence:

- Check the setup_data.yaml file and runtime consistency on the management node:

  ```
  # cd /root/installer-<tagid>/tools
  # ciscovim run --perform 1,3 -y
  ```

- Execute the cloud sanity command:

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

- Check the status of the REST API server and the corresponding directory where it is running:

  ```
  # cd/root/installer-<tagid>/tools
  #./restapi.py -a status
  Status of the REST API Server:  active (running) since Thu 2016-08-18 09:15:39 UTC; 9h
   ago
  REST API launch directory: /root/installer-<tagid>/
  ```

- If the REST API server is not running from the right installer directory, execute the following to get it running from the correct directory:

  ```
  # cd/root/installer-<tagid>/tools
  #./restapi.py -a setup

  Check if the REST API server is running from the correct target directory
  #./restapi.py -a status
  Status of the REST API Server:  active (running) since Thu 2016-08-18 09:15:39 UTC; 9h
   ago
  REST API launch directory: /root/new-installer-<tagid>/
  ```

- Verify Nova services are good across the compute nodes by executing the following command:

  ```
  # source /root/openstack-configs/openrc
  # nova service-list
  ```

If cloud-sanity fails, execute cluster-recovery (ciscovim cluster-recovery), then re-execute the cloud-sanity and nova service-list steps as listed above.

Recovery of compute and OSD nodes requires network connectivity and reboot so that they can be accessed using SSH without password from the management node.

To shutdown, bring the pod down in the following sequence:

1. Shut down all VMs, then all the compute nodes
2. Shut down all storage nodes serially
3. Shut down all controllers one at a time
4. Shut down the management node
5. Shut down the networking gears

   Bring the nodes up in reverse order, that is, start with networking gears, then the management node, storage nodes, control nodes, and compute nodes. Make sure that each node type is completely booted up before you move on to the next node type.

   Validate the Cisco API server by running the following command:

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

Run the cluster recovery command to bring up the POD post power-outage

```
# help on sub-command
ciscovim help cluster-recovery

# execute cluster-recovery
ciscovim cluster-recovery
```

```
# execute docker cloudpulse check
# ensure all containers are up
cloudpulse run --name docker_check
```

Validate if all the VMs are up (not in shutdown state). If any of the VMs are in down state, bring them up using the Horizon dashboard.

Managing Nova Compute Scheduler Filters and User Data

OpenStack Nova is an OpenStack component that provides on-demand access to compute resources by provisioning large networks of virtual machines (VMs). In addition to the standard Nova filters, Cisco VIM supports the following additional scheduler filters:

- ServerGroupAffinityFilter—Ensures that an instance is scheduled onto a host from a set of group hosts. To use this filter, you must create a server group with an affinity policy and pass a scheduler hint using group as the key and the server group UUID as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova server-group-create --policy affinity group-1
  $ nova boot --image IMAGE_ID --flavor 1 --hint group=SERVER_GROUP_UUID server-1
  ```

- ServerGroupAntiAffinityFilter—Ensures that each group instance is on a different host. To use this filter, you must create a server group with an anti-affinity policy and pass a scheduler hint, using group as the key and the server group UUID as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova server-group-create --policy anti-affinity group-1
  $ nova boot --image IMAGE_ID --flavor 1 --hint group=SERVER_GROUP_UUID server-1
  ```

- SameHostFilter—Within an instance set, schedules one instance on the same host as another instance. To use this filter, pass a scheduler hint using **same_host** as the key and a list of instance UUIDs as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova boot --image IMAGE_ID --flavor 1 --hint same_host=INSTANCE_ID server-1
  ```

- DifferentHostFilter—Within an instance set, schedules one instance on a different host than another instance. To use this filter, pass a scheduler hint using **different_host** as the key and a list of instance UUIDs as the value. The filter is the opposite of SameHostFilter. Use the **nova**command-line tool and the **--hint** flag. For example:

  ```
  $ nova boot --image IMAGE_ID --flavor 1 --hint different_host=INSTANCE_ID server-1
  ```

In addition to scheduler filters, you can set up user data files for cloud application initializations. A user data file is a special key in the metadata service that holds a file that cloud-aware applications in the guest instance can access. For example, one application that uses user data is the cloud-init system, an open-source package that is available on various Linux distributions. The cloud-init system handles early cloud instance initializations. The typical use case is to pass a shell script or a configuration file as user data during the Nova boot, for example:

```
$ nova boot --image IMAGE_ID --flavor 1 --hint user-data FILE_LOC server-1
```

Managing Nova Compute Scheduler Filters and User Data

OpenStack Nova is an OpenStack component that provides on-demand access to compute resources by provisioning large networks of virtual machines (VMs). In addition to the standard Nova filters, Cisco VIM supports the following additional scheduler filters:

- ServerGroupAffinityFilter—Ensures that an instance is scheduled onto a host from a set of group hosts. To use this filter, you must create a server group with an affinity policy and pass a scheduler hint using group as the key and the server group UUID as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova server-group-create --policy affinity group-1
  $ nova boot --image IMAGE_ID --flavor 1 --hint group=SERVER_GROUP_UUID server-1
  ```

- ServerGroupAntiAffinityFilter—Ensures that each group instance is on a different host. To use this filter, you must create a server group with an anti-affinity policy and pass a scheduler hint, using group as the key and the server group UUID as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova server-group-create --policy anti-affinity group-1
  $ nova boot --image IMAGE_ID --flavor 1 --hint group=SERVER_GROUP_UUID server-1
  ```

- SameHostFilter—Within an instance set, schedules one instance on the same host as another instance. To use this filter, pass a scheduler hint using **same_host** as the key and a list of instance UUIDs as the value. Use the **nova** command-line tool and the **--hint** flag. For example:

  ```
  $ nova boot --image IMAGE_ID --flavor 1 --hint same_host=INSTANCE_ID server-1
  ```

- DifferentHostFilter—Within an instance set, schedules one instance on a different host than another instance. To use this filter, pass a scheduler hint using **different_host** as the key and a list of instance UUIDs as the value. The filter is the opposite of SameHostFilter. Use the **nova**command-line tool and the **--hint** flag. For example:

  ```
  $ nova boot --image IMAGE_ID --flavor 1 --hint different_host=INSTANCE_ID server-1
  ```

In addition to scheduler filters, you can set up user data files for cloud application initializations. A user data file is a special key in the metadata service that holds a file that cloud-aware applications in the guest instance can access. For example, one application that uses user data is the cloud-init system, an open-source package that is available on various Linux distributions. The cloud-init system handles early cloud instance initializations. The typical use case is to pass a shell script or a configuration file as user data during the Nova boot, for example:

```
$ nova boot --image IMAGE_ID --flavor 1 --hint user-data FILE_LOC server-1
```

Monitoring Cisco NFVI Health with CloudPulse

You can query the state of various Cisco NFVI OpenStack endpoints using CloudPulse, an OpenStack health-checking tool. By default, the tool automatically polls OpenStack Cinder, Glance, Nova, Neutron, Keystone, Rabbit, Mariadb, and Ceph every four minutes. However, you can use a CLI REST API call from the management node to get the status of these services in real time. You can integrate the CloudPulse API into your applications and get the health of the OpenStack services on demand. You can find additional information about using CloudPulse in the following OpenStack sites:

- https://wiki.openstack.org/wiki/Cloudpulse

- https://wiki.openstack.org/wiki/Cloudpulseclient
- https://wiki.openstack.org/wiki/Cloudpulse/DeveloperNotes
- https://wiki.openstack.org/wiki/Cloudpulse/OperatorTests
- https://wiki.openstack.org/wiki/Cloudpulse/APIDocs

CloudPulse has two set of tests: endpoint_scenario (runs as a cron or manually) and operator test (run manually). The supported Cloudpulse tests groups include:

- nova_endpoint
- neutron_endpoint
- keystone_endpoint
- glance_endpoint
- cinder_endpoint

Operator tests include:

- ceph_check—Executes the command, "ceph -f json status" on the Ceph-mon nodes and parses the output. If the result of the output is not “HEALTH_OK” ceph_check the reports for an error.
- docker_check—Finds out if all the Docker containers are in the running state in all the nodes. It the report for an error if any containers are in the Exited state. It runs the command “docker ps -aq --filter 'status=exited'”.
- galera_check—Executes the command, "mysql 'SHOW STATUS;” on the controller nodes and displays the status.
- node_check—Checks if all the nodes in the system are up and online. It also compares the result of “nova hypervisor list” and finds out if all the computes are available.
- rabbitmq_check—Runs the command, “rabbitmqctl cluster_status” on the controller nodes and finds out if the rabbitmq cluster is in quorum. If nodes are offline in the cluster rabbitmq_check the report is considered as failed.

CloudPulse servers are installed in containers on all control nodes. The CloudPulse client is installed on the management node by the Cisco VIM installer. To execute CloudPulse, source the openrc file in the openstack-configs directory and execute the following:

```
[root@MercRegTB1 openstack-configs]# cloudpulse --help
usage: cloudpulse [--version] [--debug] [--os-cache]
                  [--os-region-name <region-name>]
                  [--os-tenant-id <auth-tenant-id>]
                  [--service-type <service-type>]
                  [--endpoint-type <endpoint-type>]
                  [--cloudpulse-api-version <cloudpulse-api-ver>]
                  [--os-cacert <ca-certificate>] [--insecure]
                  [--bypass-url <bypass-url>] [--os-auth-system <auth-system>]
                  [--os-username <username>] [--os-password <password>]
                  [--os-tenant-name <tenant-name>] [--os-token <token>]
                  [--os-auth-url <auth-url>]
                  <subcommand> ...
```

To check results of periodic CloudPulse runs:

```
# cd /root/openstack-configs
# source openrc
# cloudpulse result
+--------------------------------------+------+-------------------+----------+---------+
| uuid | id | name | testtype | state |
+--------------------------------------+------+-------------------+----------+---------+
| bf7fac70-7e46-4577-b339-b1535b6237e8 | 3788 | glance_endpoint | periodic | success |
| 1f575ad6-0679-4e5d-bc15-952bade09f19 | 3791 | nova_endpoint | periodic | success |
```

To view all CloudPulse tests:

```
# cd /root/openstack-configs
# source openrc
# cloudpulse result
+--------------------------------------+------+-------------------+----------+---------+
| uuid | id | name | testtype | state |
+--------------------------------------+------+-------------------+----------+---------+
| bf7fac70-7e46-4577-b339-b1535b6237e8 | 3788 | glance_endpoint | periodic | success |
| 1f575ad6-0679-4e5d-bc15-952bade09f19 | 3791 | nova_endpoint | periodic | success |




| 765083d0-e000-4146-8235-ca106fa89864 | 3794 | neutron_endpoint | periodic | success |
| c1c8e3ea-29bf-4fa8-91dd-c13a31042114 | 3797 | cinder_endpoint | periodic | success |
| 04b0cb48-16a3-40d3-aa18-582b8d25e105 | 3800 | keystone_endpoint | periodic | success |
| db42185f-12d9-47ff-b2f9-4337744bf7e5 | 3803 | glance_endpoint | periodic | success |
| 90aa9e7c-99ea-4410-8516-1c08beb4144e | 3806 | nova_endpoint | periodic | success |
| d393a959-c727-4b5e-9893-e229efb88893 | 3809 | neutron_endpoint | periodic | success |
| 50c31b57-d4e6-4cf1-a461-8228fa7a9be1 | 3812 | cinder_endpoint | periodic | success |
| d1245146-2683-40da-b0e6-dbf56e5f4379 | 3815 | keystone_endpoint | periodic | success |
| ce8b9165-5f26-4610-963c-3ff12062a10a | 3818 | glance_endpoint | periodic | success |
| 6a727168-8d47-4a1d-8aa0-65b942898214 | 3821 | nova_endpoint | periodic | success |
| 6fbf48ad-d97f-4a41-be39-e04668a328fd | 3824 | neutron_endpoint | periodic | success |
+--------------------------------------+------+-------------------+----------+---------+
```

To run a CloudPulse test on demand:

```
# cd /root/openstack-configs
# source openrc
# cloudpulse run --name <test_name>
# cloudpulse run  --all-tests
# cloudpulse run --all-endpoint-tests
# cloudpulse run --all-operator-tests
```

To run a specific CloudPulse test on demand:

```
# cloudpulse run --name neutron_endpoint
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| name       | neutron_endpoint                     |
| created_at | 2016-03-29T02:20:16.840581+00:00     |
| updated_at | None                                 |
| state      | scheduled                            |
| result     | NotYetRun                            |
| testtype   | manual                               |
| id         | 3827                                 |
| uuid       | 5cc39fa8-826c-4a91-9514-6c6de050e503 |
+------------+--------------------------------------+
```

To show detailed results of a specific CloudPulse run:

```
#cloudpulse show 5cc39fa8-826c-4a91-9514-6c6de050e503
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| name       | neutron_endpoint                     |
| created_at | 2016-03-29T02:20:16+00:00            |
| updated_at | 2016-03-29T02:20:41+00:00            |
| state      | success                              |
| result     | success                              |
| testtype   | manual                               |
| id         | 3827                                 |
| uuid       | 5cc39fa8-826c-4a91-9514-6c6de050e503 |
+------------+--------------------------------------+
```

To see the CloudPulse options, source the openrc file in openstack-configs dir and execute:

```
#cloudpulse --help
```

The CloudPulse project has a RESTful Http service called the Openstack Health API. Through this API cloudpulse allows the user to list the cloudpulse tests, create new cloudpulse tests and see the results of the cloudpulse results.

The API calls described in this documentation require keystone authentication. We can use the keystone v2 or v3 version for the authentication. The corresponding configuration must be configured properly in the cloudpulse config in order that the cloudpulse can reach the v2 or the v3 keystone API.

The Identity service generates authentication tokens that permit access to the Cloudpulse REST APIs. Clients obtain this token and the URL endpoints for other service APIs by supplying their valid credentials to the authentication service. Each time you make a REST API request to Cloudpulse, you need to supply your authentication token in the X-Auth-Token request header.

Assessing Cisco NFVI Status with Cloud-Sanity

The cloud-sanity tool is designed to give you a quick overall status of the Pods health. Cloud-sanity can run tests on all node types in the Pod: management, control, compute, and ceph storage.

The following are test areas that are supported in cloud-sanity:

1. RAID Disk health checks.
2. Basic network connectivity between the management node and all other nodes in the Pod.
3. Mariadb cluster size.
4. RabbitMQ operation and status.
5. Nova service and hypervisor list.
6. CEPHMon operation and status.
7. CEPHOSD operation and status.

To run the cloud-sanity tool, logi n to the management node and navigate to the tools folder for the installation currently running. Following are the cloud-sanity run options. Cloud-sanity can be run on all nodes or a particular target role.

Step 1 To run the cloud sanity, complete the following steps:

```
# cd /root/installer-<tag>/tools
# ./cloud_sanity.py –h
usage: cloud_sanity.py [-h] [--check CHECK] [--list] [--verbose]

cloud sanity helper

optional arguments:
  -h, --help            show this help message and exit
  --check CHECK, -c CHECK
                        all - Run all sanity checks. [default action]
                        control - Run controller sanity checks.
                        compute - Run compute sanity checks.
                        cephmon - Run cephmon sanity checks.
                        cephosd - Run cephosd sanity checks.
                        management - Run Management node sanity checks
  --list, -l            List all the available sanity checks.
  --verbose, -v         Run the sanity in verbose mode.
```

Step 2 To run all the cloud-sanity tests, select *all* as the check option.

```
./cloud_sanity.py --check all
Executing All Cloud Sanity in quiet mode. This takes some time.
Cloud Sanity Results
+-------------+------------------------------------------------------------+---------+
| Role        | Task                                                       | Result  |
+-------------+------------------------------------------------------------+---------+
| Management  | Management - Disk maintenance RAID Health **************** | PASSED  |
|             |                                                            |         |
| Management  | Management - Disk maintenance VD Health ****************** | PASSED  |
|             |                                                            |         |
| Control     | Control - Ping All Controller Nodes ********************** | PASSED  |
|             |                                                            |         |
| Control     | Control - Ping internal VIP ****************************** | PASSED  |
|             |                                                            |         |
| Control     | Control - Check Mariadb cluster size ********************* | PASSED  |
|             |                                                            |         |
| Control     | Control - Check RabbitMQ is running ********************** | PASSED  |
|             |                                                            |         |
| Control     | Control - Check RabbitMQ cluster status ****************** | PASSED  |
|             |                                                            |         |
| Control     | Control - Check Nova service list ************************ | PASSED  |
|             |                                                            |         |
| Control     | Control - Disk maintenance RAID Health ******************* | PASSED  |
|             |                                                            |         |
| Control     | Control - Disk maintenance VD Health ********************* | PASSED  |
|             |                                                            |         |
| Compute     | Compute - Ping All Compute Nodes ************************* | PASSED  |
|             |                                                            |         |
| Compute     | Compute - Check Nova Hypervisor list ********************* | PASSED  |
|             |                                                            |         |
| Compute     | Compute - Disk maintenance RAID Health ******************* | PASSED  |
|             |                                                            |         |
| Compute     | Compute - Disk maintenance VD Health ********************* | PASSED  |
|             |                                                            |         |
| CephMon     | CephMon - Check cephmon is running *********************** | PASSED  |
|             |                                                            |         |
| CephMon     | CephMon - CEPH cluster check ***************************** | PASSED  |
|             |                                                            |         |
| CephMon     | CephMon - Check Ceph Mon status ************************** | PASSED  |
|             |                                                            |         |
| CephMon     | CephMon - Check Ceph Mon results ************************* | PASSED  |
|             |                                                            |         |
| CephOSD     | CephOSD - Ping All Storage Nodes ************************* | PASSED  |
|             |                                                            |         |
| CephOSD     | CephOSD - Check OSD result with osdinfo ****************** | PASSED  |
|             |                                                            |         |
| CephOSD     | CephOSD - Check OSD result without osdinfo *************** | PASSED  |
|             |                                                            |         |
+-------------+------------------------------------------------------------+---------+
[PASSED] Cloud Sanity All Checks Passed
```

The cloud-sanity tests use the disk-maintenance and osd-maintenance tools to assess overall health and status of RAID disks and OSD status.

Note Failures that are detected in RAID disk health and CEPHOSD operational status is evaluated with the disk-maintenance and osd-maintenance tools.

Service Catalog URL

The OpenStack Keystone service catalog allows API clients to dynamically discover and navigate to cloud services. Cloudpulse has its own service URL which is added to the Keystone service catalog. You need to send a token request to Keystone to find the service URL of cloudpulse. The token request lists all the catalog of services available.

Get Token from Keystone

To get the token from keystone run the following commands:

Resource URI

Verb	URI
POST	http://<controller_lb_ip>:5000/v2.0/tokens

Example

```
JSON Request
POST / v2.0/tokens
Accept: application/json
{
    "auth": {
        "passwordCredentials":{
                  "username": "admin",
                  "password": "iVP1YciVKoMGId1O"
        }
    }
}

JSON Response
200 OK
Content-Type: application/json
{
  "access": {
    "token": {
      "issued_at": "2017-03-29T09:54:01.000000Z",
      "expires": "2017-03-29T10:54:01Z",
      "id":
"gAAAAABY24Q5TDIqizuGmhOXakV2rIzSvSPQpMAmC7SA2UzUXZQXSH-ME98d3Fp4Fsj16G561a420B4BK0fylcykL22EcO9",
…………
……..
}
```

Get Service Catalog URL for Cloudpulse

Resource URI

Verb	URI
GET	http://<controller_ip>:35357/v2.0/endpoints

Example

```
JSON Request
GET /v2.0/endpoints
Accept: application/json

JSON Response
200 OK
Content-Type: application/json
{"endpoints": [
     {"internalurl": "http://<controller>:9999",
      "adminurl": "http://<controller>:9999",
      "publicurl":"http://<controller>:9999"
}]}
}
```

Cloudpulse APIs

The following are a list of APIs and the corresponding functions that the API performs. The cloudpulse APIs is accessed with the X-Auth-Token which contains the token which is received from the Keystone token generation API mentioned in the preceding panel.

List of Cloudpulse Tests

To get the list of cloudpulse tests:

Resource URI

Verb	URI
GET	http://<controller_ip>:9999/cpulse

Example

```
JSON Request
GET /cpulse
Accept: application/json

JSON Response
200 OK
Content-Type: application/json
{
  "cpulses": [
    {
      "name": "galera_check",
      "state": "success",
      "result":"ActiveNodes:16.0.0.37,16.0.0.17,16.0.0.27",
      "testtype": "periodic",
      "id": 4122,
      "uuid": "a1b52d0a-ca72-448a-8cc0-5bf210438d89"
    }]
}
```

Get detailed result of 1 test

To get detailed result of the test.

Resource URI

Verb	URI
GET	http://<controller_ip>:9999/cpulse/<uuid>

Uuid : uuid of the test

Example

```
JSON Request
GET /cpulse/e6d4de91-8311-4343-973b-c507d8806e94
Accept: application/json

JSON Response
200 OK
Content-Type: application/json
{
      "name": "galera_check",
      "state": "success",
      "result":"ActiveNodes:16.0.0.37,16.0.0.17,16.0.0.27",
      "testtype": "periodic",
      "id": 4122,
      "uuid": " e6d4de91-8311-4343-973b-c507d8806e94"
}
```

Get List of Tests Available

To get a list of available cloudpulse tests:

Resource URI

Verb	URI
GET	http://<controller_ip>:9999/cpulse/list_tests

Example

```
JSON Request
GET /cpulse/list_tests
Accept: application/json

JSON Response
```

```
200 OK
Content-Type: application/json
{
  "endpoint_scenario":
"all_endpoint_tests\ncinder_endpoint\nglance_endpoint\nkeystone_endpoint\nneutron_endpoint\nnova_endpoint",

  "operator_scenario":
"all_operator_tests\nceph_check\ndocker_check\ngalera_check\nnode_check\nrabbitmq_check"
}
```

Schedule a manual cloudpulse test:

To schedule a manual test of cloudpulse run the following commands:

Resource URI

Verb	URI
POST	http://<controller_ip>:9999/cpulse

Example

```
JSON Request
POST /cpulse
Accept: application/json
{
"name": "galera_check"
}

JSON Response
200 OK
Content-Type: application/json
{
      "name": "galera_check",
      "state": "scheduled",
      "result":"NotYetRun",
      "testtype": "manual",
      "id": 4122,
      "uuid": " e6d4de91-8311-4343-973b-c507d8806e94"
}
```

Remove the results of a test

To remove the results of a test.

Resource URI

Verb	URI
DELETE	http://<controller_ip>:9999/cpulse/<uuid>

Uuid : uuid of the test

Example

```
JSON Request
DELETE /cpulse/68ffaae3-9274-46fd-b52f-ba2d039c8654
```

```
Accept: application/json

JSON Response
204 No Content
```

Checking Network Connections

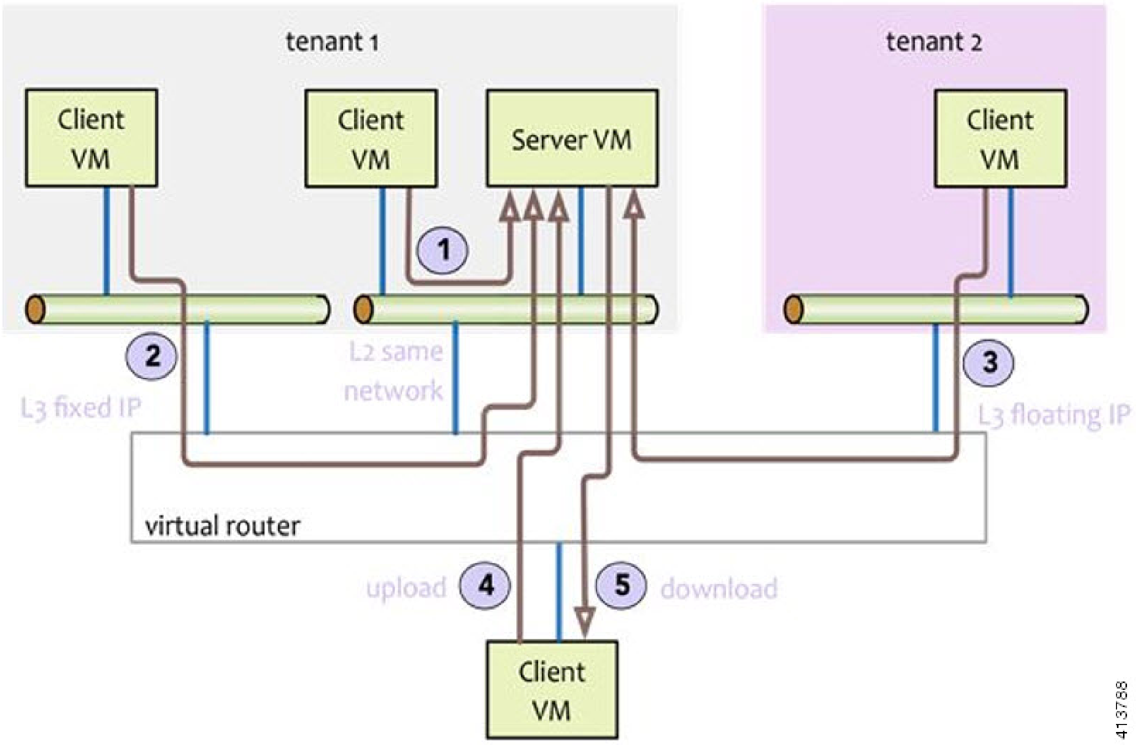
You can use Virtual Machine Through Put (VMTP) to check Layer 2 and Layer 3 data plane traffic between Cisco NFVI compute nodes. VMTP performs ping connectivity, round trip time measurement (latency), and TCP/UDP throughput measurement for the following Cisco NFVI east to west VM-to-VM flows:

- Same network (private fixed IP, flow number 1).
- Different network using fixed IP (same as intra-tenant L3 fixed IP, flow number 2).
- Different network using floating IP and NAT (same as floating IP inter-tenant L3, flow number 3.)
- When an external Linux host is available for testing north to south flows, external host to VM download and upload throughput and latency (L3/floating IP, flow numbers 4 and 5).

The following figure shows the traffic flows VMTP measures. Cloud traffic flows are checked during Cisco VIM installation and can be checked at any later time by entering the following command:

```
$ ciscovim run --perform 8 -y
```

Figure 1: VMTP Cloud Traffic Monitoring



Enabling NFVBench Post Deployment

NFVBench is a data plane performance benchmark tool for NFVI that can be optionally installed after the pod deployment.

NFVBench is used to:

- Verify that the data plane is working properly and efficiently when using well defined packet paths that are typical of NFV service chains.
- Measure the actual performance of your data plane so that you can estimate what VNFs can expect from the infrastructure when it comes to receiving and sending packets.

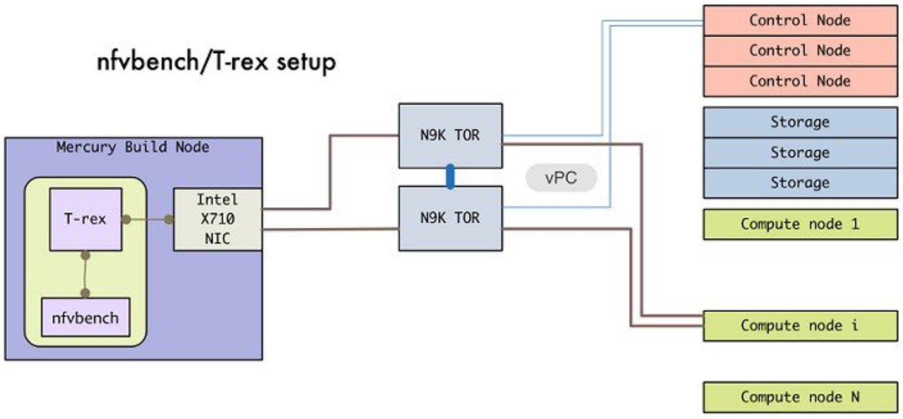
While VMTP only measures VM to VM traffic, NFVBench measures traffic flowing from an integrated software traffic generator (TRex) running on the management node to the ToR switches to test VMs running in compute nodes.

In Cisco VIM, the NFVBench (performance benchmark) is an optional tool. You can deploy NFVBench after the installation of the pod.

Before you begin

- A 10GE Intel NIC (Intel X710 NIC (4 x 10G)) must be installed on a management node.
- A TRex traffic generator which uses DPDK interface to interact with Intel NIC and makes use of hardware, instead of software to generate packets. This approach is more scalable and enables NFVBench to perform tests without software limitations.
- Wire two physical interfaces of the Intel NIC to the TOR switches (as shown in the following figure).

Figure 2: NFVBench topology setup



Procedure

	Command or Action	Purpose
Step 1	Enable the NFVBench configuration in the setup_data.yaml file.	Sample configuration files for OVS/VLAN or VPP mechanism driver:

	Command or Action	Purpose

```
NFVBENCH:
   enabled: True    # True or False
   tor_info: {TORa: eth1/42, TORb: eth1/42} #
mandatory
#  tor_info: {TOR: 'eth1/42,eth1/43'} # use if
there is only one TOR switch
#  nic_ports: 3,4    # Optional input, indicates
which 2 of the 4 available ports
                     # of 10G Intel NIC on the
management node is NFVbench tool using
                    # to send and receive traffic.

                     # Defaults to the first 2
ports of NIC (ports 1 and 2) if not specified.
                    # Port number must be between
 1 and 4, one port cannot be used twice.

                     # Example:
                     # nic_ports: 1,4    # the
first and the last port of Intel NIC are used
                    # nic_slot: 2        # #
Optional, defaults to 1st set of unbonded pair of
 NIC ports in an
                  Intel 710 or 520 card the code
finds; Via this option, one can choose to run
NFVbench
                  via XL710, 520 or X710 card
```

Sample configuration for VTS mechanism driver:

```
NFVBENCH:
   enabled: True    # True or False
   tor_info: {TORa: eth1/42, TORb: eth1/42} #
mandatory
   vtep_vlans: 1500,1501    # Mandatory and needed
 only for VTS/VXLAN.
                            # Specify any pair of
 unused VLAN ids to be used
                            # for VLAN to VxLAN
encapsulation in TOR switch.
#  tor_info: {TOR: 'eth1/42,eth1/43'} # Use if
there is only one TOR switch.
#  nic_ports: 3,4    # Optional input, indicates
which 2 of the 4 available ports
                     # of 10G Intel NIC on the
management node is NFVbench tool using
                    # to send and receive traffic.

                     # Defaults to the first 2
ports of NIC (ports 1 and 2) if not specified.
                    # Port number must be between
 1 and 4, one port cannot be used twice.
                     # Example:
                     # nic_ports: 1,4    # the
first and the last port of Intel NIC are used

VTS_PARAMETERS:
   …
VTS_DAY0: '<True|False>’# Required parameter when
 VTS enabled
VTS_USERNAME: '<vts_username>'# Required parameter
 when VTS enabled
VTS_PASSWORD: '<vts_password>'# Required parameter
```

	Command or Action	Purpose

		` when VTS enabled`
		`VTS_NCS_IP: '11.11.11.111'# '<vts_ncs_ip>',`
		`mandatory when VTS enabled`
		`VTC_SSH_USERNAME: 'admin'# '<vtc_ssh_username>',`
		`mandatory for NFVbench`
		`VTC_SSH_PASSWORD: 'my_password'#`
		`'<vtc_ssh_password>', mandatory for NFVbench`

Step 2	Configuring minimal settings of NFVBench:	`# Minimal settings required for NFVbench`
		`TORSWITCHINFO:`
		`   CONFIGURE_TORS: <True or False> # True if`
		`switches should be configured to support NFVbench`
		`   …`
		`   SWITCHDETAILS:`
		`   - hostname: 'TORa'   # Hostname matching`
		`'tor_info' switch name.`
		`     username: 'admin'   # Login username for`
		`switch user.`
		`     password: 'my_password'  # Login password for`
		` switch user.`
		`     ssh_ip: '172.31.230.123' # SSH IP for switch.`
		
		`   - hostname: 'TORb'`
		`     username: 'admin'`
		`     password: 'my_password'`
		`     ssh_ip: '172.31.230.124'`
		
		TOR switches will be configured based on information provided in tor_info. Two ports specified by interfaces are configured in trunk mode. In order to access them and retrieve TX/RX counters you need the Login details for TOR switches. It is not required to set 'CONFIGURE_TORS' to 'True', but then manual configuration is necessary.
		
		With VTS as mechanism driver additional settings are needed. NFVBench needs access to VTS NCS to perform cleanup after it detaches the traffic generator port from VTS. Also a pair of VTEP VLANs is required for VLAN to VxLAN mapping. Value can be any pair of unused VLAN ID.

Step 3	Reconfigure Cisco VIM to create a NFVBench container. To reconfigure add necessary configuration to the setup_data.yaml file, run the reconfigure command as follows.	`[root@mgmt1 ~]# cd /root/`
		`[root@mgmt1 ~]# mkdir MyDir`
		`[root@mgmt1 ~]# cp`
		`/root/openstack-configs/setup_data.yaml`
		`/root/MyDir/`
		`[root@mgmt1 ~]# cd /root/`
		`[root@mgmt1 ~]# # update the setup_data to include`
		` NFVBENCH section`
		`[root@mgmt1 ~]# cd /root/MyDir/`
		`[root@mgmt1 ~]# vi setup_data.yaml`
		`[root@mgmt1 ~]# cd ~/installer-xxxx`
		`[root@mgmt1 ~]# ciscovim --setupfile`
		`/root/MyDir/setup_data.yaml reconfigure`
		
		After reconfiguration is done, you can see NFVBench container up and is ready to use.

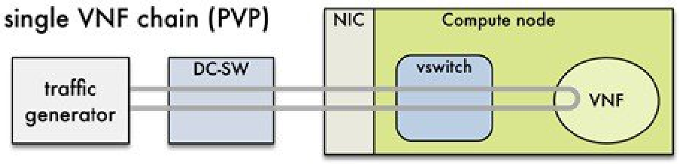
NFVBench Usage

Built-in packet paths

NFVBench can setup and stage three different packet paths.

The default packet path is called **PVP** (Physical - VM - Physical) and represents a typical service chain made of 1 VNF/VM:

Figure 3: Single VNF chain (PVP)



The traffic generator runs inside the NFVBench container on the management node. DC-SW represents the top of rack switch(es). The VNF is a test VM that contains a fast L3 router based on FD.io VPP. This VNF image can also be configured to run an L2 forwarder based on DPDK testpmd (both options generally yield roughly similar throughput results).

Traffic is made of UDP packets generated on the 2 physical interfaces (making it a bi-directional traffic). Packets are forwarded by the switch to the appropriate compute node before arriving to the virtual switch, then to the VNF before looping back to the traffic generator on the other interface. Proper stitching of the traffic on the switch is performed by NFVbench by using the appropriate mechanism (VLAN tagging for VLAN based deployments, VxLAN VTEP in the case of VTS deployments).

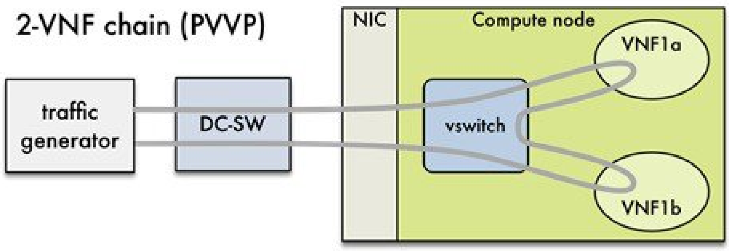
The performance of the PVP packet path provides a very good indication of the capabilities and efficiency of the NFVi data plane in the case of a single service chain made of 1 VNF/VM.

NFVBench also supports more complex service chains made of 2 VM in sequence and called PVVP (Physical-VM-VM-Physical).

In a PVVP packet path, the 2 VMs can reside on the same compute node (PVVP intra-node) or on different compute nodes (PVVP inter-node).

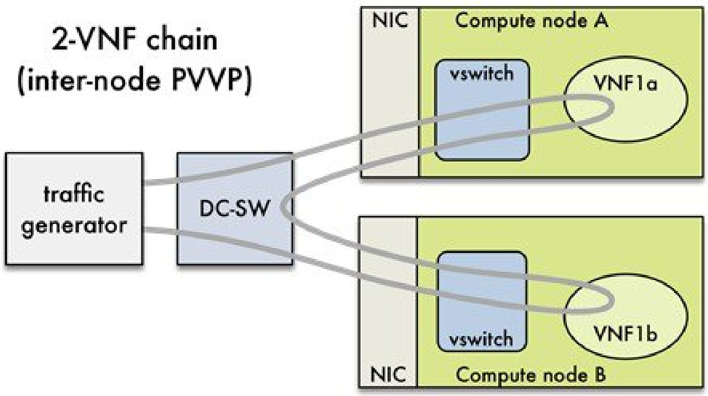
PVVP intra-node is more efficient when a virtual switch is used as packets do not have to go through the switch between the 2 VMs:

Figure 4: 2-VNF chain (PVVP)



PVVP inter-node requires packets to go through the switch and back between the 2 VMs.

Figure 5: 2-VNF chain(inter-node PVVP)



NFVBench Command-Line Options

The common NFVBench command-line options are displayed using the --help option:

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

Control Plane Verification

If you are trying NFVBench for the first time, verify that the tool can stage the default packet path properly without sending any traffic.

The --no-traffic option exercises the control plane by creating a single test service chain with one VM, but does not send any traffic.

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

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

Fixed Rate Run Test

The data plane traffic test is to generate traffic at a fixed rate for a fixed duration. For example, to generate a total of 10,000 packets per second (which is 5,000 packets per second per direction) for the default duration (60 seconds) and using the default frame size of 64 bytes:

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

Packet Sizes

You can specify any list of frame sizes using the -frame-size option (pass as many as desired), including IMIX.

Following is an example, to run a fixed rate with IMIX and 1518 byte frames:

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

NDR and PDR Test

NDR and PDR test is used to determine the performance of the data plane in terms of throughput at a given drop rate.

- No Drop Rate(NDR)- It is the highest throughput achieved while allowing zero packet drop (allows a very low drop rate usually lesser than 0.001%).
- Partial Drop Rate (PDR)-It is the highest throughput achieved while allowing most at a given drop rate (typically less than 0.1%).

NDR is always less or equal to PDR.

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

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

Multi-chain Test

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

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

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

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

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

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

Multi-Flow Test

In Multi-flow test, one flow is defined by a source and destination MAC/IP/port tuple in the generated packets. It is possible to have many flows per chain. The maximum number of flows that are supported is in the order of 1 million flows per direction.

The following command runs three chains with a total of 100K flows per direction (for all chains):

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

External Chain Test

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

This test is used when you want to use NFVBench for only traffic generation. In this case, NFVBench sends traffic from traffic generator and reports results without performing any configuration.

Do necessary configurations such as creating networks and VMs with a configuration that allows generated traffic to pass. NFVBench has to know the 2 edge networks to which the traffic generators are attached.

If the external chains only support L2 forwarding, the NFVBench configuration must:

- Enable VLAN tagging and define the VLAN IDs to use - if applicable (or disable vlan tagging if it is not required).

- The destination MAC to use in each direction (depends on the L2 forwarding mode in place in the service chain).

If the external chains support IPv4 routing, the NFVBench configuration must:

- Define the public IP addresses of the service chain end points (gateway IP) that used to discover destination MAC using ARP.
- Set the vlan tagging appropriately.

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

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

Note NFVBench cannot access ToR switches or v-switch in compute node.

NFVBench Result Generation and Storage

NFVBench detailed results can be stored in JSON format if you pass the --json option with a destination file name or the --std-json option with a destination folder pathname (if you want to use a standard file name generated by NFVBench). It is also possible to use both methods to generate the output into two different files at the same time:

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

The above command creates two JSON files in /tmp/nfvbench container directory, which is mapped to the host directory. The first file is named my.json.

With the--std-json option, the standard NFVBench filename format follows this pattern:

<service-chain-type>-<service-chain-count>-<flow-count>-<frame-sizes>.json

Default chain is PVP and flag *-fs* was used to override traffic profile in the configuration file. With three chains and 10 flows specified file, the *PVP-3-10-64.json* is created.

Interpretation of Results

NFVBench prints data to the command line prompt in a table form. The data includes the current configuration, test devices details, and results computed based on traffic statistics.

Fixed Rate

Run the following command on NFVBench to view the traffic generated at fixed rate at different components of the packet path:

```
[root@mgmt1 ~]# nfvbench --rate 5kpps -fs IMIX
```

NFVBench summary consists of multiple blocks. In some cases, NFVBench displays lesser data. For example, in the output, the EXT chain does not access some path components (like switch). Therefore, the summary does not display.

```
============== NFVBench Summary ========== Date: 2017-03-28 19:59:53
NFVBench version 0.3.5 Openstack Neutron:
vSwitch: VTS Encapsulation: VxLAN
Benchmarks:
> Networks:
> Components:
```

```
> TOR:
Type: N9K Version:
10.28.108.249:
BIOS: 07.34
NXOS: 7.0(3)I2(2b) 10.28.108.248:
BIOS: 07.34
NXOS: 7.0(3)I2(2b)
> VTC:
Version:
build_date: 2017-03-03-05-41
build_number: 14
git_revision: 0983910
vts_version: 2.3.0.40 git_branch: vts231newton
job_name: vts231newton_gerrit_nightly
> Traffic Generator: Profile: trex-local Tool: TRex≠ Version:
build_date: Feb 16 2017 version: v2.18 built_by: hhaim build_time: 18:59:02
> Service chain:
> PVP:
> Traffic:
VPP version:
sjc04-pod3-compute-4: v17.04-rc0~98-g8bf68e8 Profile:  custom_traffic_profile Bidirectional:
 True
Flow count: 1
Service chains count: 1
Compute nodes: [u'nova:sjc04-pod3-compute-4'] Run Summary:

+---------------+------------+--------------------+-------------------+-------------------+
|  L2 Frame Size  |  Drop Rate | Avg Latency (usec) | Min Latency (usec)
| Max Latency (usec) |
+===============+============+====================+===================+===================+
| IMIX | 0.0000% | 16.50 | 10.00
| 241.00 |
+---------------+------------+--------------------+-------------------+-------------------+

L2 frame size: IMIX
Chain analysis duration: 73 seconds




Run Config:
+-------------+------------------+--------------+-----------+
| Direction | Duration (sec) | Rate | Rate |
+=============+==================+==============+===========+
| Forward | 60 | 7.6367 Mbps  | 2,500 pps |
+-------------+------------------+--------------+-----------+
| Reverse | 60 | 7.6367 Mbps  | 2,500 pps |
+-------------+------------------+--------------+-----------+
| Total | 60 | 15.2733 Mbps | 5,000 pps |
+-------------+------------------+--------------+-----------+
Chain Analysis:

+---------------+------+-----------+----------+----------+-----------+----------+----------+

| Interface | Device | Packets (fwd) | Drops (fwd)  |  Drop% (fwd) | Packets (rev) | Drops
 (rev) | Drop% (rev) |
+===============+======+===========+==========+==========+===========+==========+==========+

| traffic-generator | trex | 150,042 | |
| 150,042 | 0 | 0.0000% |
+---------------+------+-----------+----------+----------+-----------+----------+----------+
 | vni-5098 | n9k | 150,042 | 0 |
0.0000% | 150,042 | 0 |  0.0000% |
┼───────────────┼──────┼───────────┼──────────┼──────────┼───────────┼──────────┼──────────┼

 | vxlan_tunnel0 | vpp | 150,042 | 0 |
0.0000% | 150,042 | 0 |  0.0000% |
┼───────────────┼──────┼───────────┼──────────┼──────────┼───────────┼──────────┼──────────┼

 | VirtualEthernet0/0/1 | vpp | 150,042 | 0 |
0.0000% | 150,042 | 0 |  0.0000% |
┼───────────────┼──────┼───────────┼──────────┼──────────┼───────────┼──────────┼──────────┼

 | VirtualEthernet0/0/0 | vpp | 150,042 | 0 |
0.0000% | 150,042 | 0 |  0.0000% |
┼───────────────┼──────┼───────────┼──────────┼──────────┼───────────┼──────────┼──────────┼

 | vxlan_tunnel1 | vpp | 150,042 | 0 |
0.0000% | 150,042 | 0 |  0.0000% |
┼───────────────┼──────┼───────────┼──────────┼──────────┼───────────┼──────────┼──────────┼

 | vni-5099 | n9k | 150,042 | 0 |
0.0000% | 150,042 | 0 |  0.0000% |
┼───────────────┼──────┼───────────┼──────────┼──────────┼───────────┼──────────┼──────────┼

 | traffic-generator | trex | 150,042 | 0 |
0.0000% | 150,042 |  | |
┼───────────────┼──────┼───────────┼──────────┼──────────┼───────────┼──────────┼──────────┼
Run as:
nfvbench -c /tmp/nfvbench/nfvbench.cfg --rate 5kpps -fs IMIX
```

Summary Interpretation:

Lines 1-34: General information about host system and used components.

Lines 35-45: Test-specific information about service chain, traffic profile, and compute nodes.

Lines 46-53: Summary of traffic profile run with results. A new row is added to the table for every packet size in test. The output displays the run summary for the IMIX packet size, but lines for 64B and 1518B can also be present. The Table contains following columns:

The run summary table includes the following columns:

- Drop Rate: The percentage of total drop rate for all chains and flows from the total traffic sent.
- Avg Latency: Average latency of average chain latencies in microseconds.
- Min Latency: Minimum latency of all chain latencies in microseconds.
- Max Latency: Maximum latency of all chain latencies in microseconds.

Lines 54-68: Length of the

Lines 69-89: Detailed analysis of test. Each row represents one interface on packet path in order they are visited. Left side of the table is for forward direction (from traffic generator port 0 to port 1), and the right side is for reverse direction (from traffic generator port 1 to port 0).

The chain analysis table has following columns:

- Interface: Interface name on devices in packet path.

- Device: Device name on which the interface is displayed in the first column is available.
- Packets (fwd): RX counter on given interface, only the first row is TX counter (it is the beginning of packet path).
- Drops (fwd): Amount of packets being dropped between current and previous interface.
- Drop% (fwd): Percentage of dropped packets on this interface to the total packet drops.
- Packets (rev): Similar to Packets (fwd) but for the reverse direction.
- Drops (rev): Similar to Drops (fwd) but for the reverse direction.
- Drop% (rev): Similar to Drop% (fwd) but for reverse direction.

This type of summary is very useful for finding bottlenecks or to verify if the system can handle certain fixed rate of traffic.

NDR/PDR

The test result shows throughput values in different units for both NDR and PDR with latency statistics (minimum, maximum, average) for each test.

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

```
========== NFVBench Summary ==========
Date: 2017-03-28 20:20:46
NFVBench version 0.3.5 Openstack Neutron:
vSwitch: VTS Encapsulation: VxLAN
Benchmarks:
> Networks:
> Components:
> TOR:
Type: N9K Version:
10.28.108.249:
BIOS: 07.34
NXOS: 7.0(3)I2(2b) 10.28.108.248:
BIOS: 07.34







> VTC:


NXOS: 7.0(3)I2(2b)

Version:
build_date: 2017-03-03-05-41
build_number: 14
git_revision: 0983910
vts_version: 2.3.0.40 git_branch: vts231newton
job_name: vts231newton_gerrit_nightly
> Traffic Generator: Profile: trex-local Tool:  TRex Version:
build_date: Feb 16 2017 version: v2.18 built_by: hhaim build_time: 18:59:02
> Measurement Parameters: NDR: 0.001
PDR: 0.1
> Service chain:
> PVP:
> Traffic:
VPP version:
sjc04-pod3-compute-4: v17.04-rc0~98-g8bf68e8 Profile:  custom_traffic_profile Bidirectional:
 True
Flow count: 1
Service chains count: 1
```

```
Compute nodes: [u'nova:sjc04-pod3-compute-4'] Run Summary:
+---+-----------+-----------+-----------+----------+-------------+-------------+-------------+

| - | L2 Frame Size |  Rate (fwd+rev)  |  Rate (fwd+rev)  |  Avg Drop Rate | Avg Latency
(usec) | Min Latency (usec) | Max Latency (usec) |
+===+===========+===========+===========+==========+=============+=============+=============+


 | NDR | IMIX | 4.5703 Gbps | 1,496,173 pps | 0.0006%
| 131.33 |  10.00 | 404.00 |
+---+-----------+-----------+-----------+----------+-------------+-------------+-------------+


 | PDR | IMIX | 4.7168 Gbps | 1,544,128 pps | 0.0553%
| 205.72 |  20.00 | 733.00 |
+---+-----------+-----------+-----------+----------+-------------+-------------+-------------+

L2 frame size: IMIX Chain analysis duration: 961 seconds NDR search duration: 661 seconds
 PDR search duration: 300 seconds
Run Config:
+-------------+-----------------+------------+---------------+
| Direction | Duration (sec) | Rate | Rate |
+=============+=================+============+===============+
| Forward | 60 | 2.3584 Gbps |  772,064 pps |
+-------------+-----------------+------------+---------------+
| Reverse | 60 | 2.3584 Gbps |  772,064 pps |
+-------------+-----------------+------------+---------------+
| Total | 60 | 4.7168 Gbps | 1,544,128 pps |
+-------------+-----------------+------------+---------------+
```

Lines 1-48: Similar to the fixed rate run output explained above.

Lines 49-58: Summary of the test run with benchmark data. For each packet size, there is a row with NDR/PDR or both values depending on chosen rate. The output displays the run summary for the IMIX packet size.

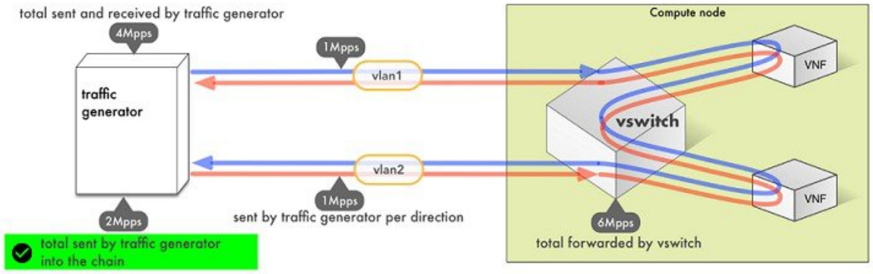
The table includes the following columns:

- L2 Frame Size: Packet size used in the test, can be one of 64B, IMIX, 1518B
- Rate (fwd+rev): Total rate satisfying NDR/PDR condition in a unit bps/pps
- Avg Drop Rate: Average drop rate of test iteration which satisfied NDR/PDR condition
- Avg Latency: Average packet latency of test iteration which satisfied NDR/PDR condition in microseconds
- Min Latency: Minimum latency of test iteration which satisfied NDR/PDR condition in microseconds
- Max Latency: Maximum latency of test iteration which satisfied NDR/PDR condition in microseconds

The NDR and PDR values along with the latency information are good indicators of NFVI solution.

The following figure explains different approaches to interpret the same results.

Figure 6: Measured rates form the traffic generator


total sent and received by traffic generator
4Mpps
1Mpps
vlan1
traffic generator
Compute node
VNF
vswitch
vlan2
1Mpps
2Mpps
sent by traffic generator per direction
VNF
6Mpps
total forwarded by vswitch
total sent by traffic generator into the chain

NFVBench always reports measured rate from the traffic generator perspective as total amount sent into the chain.

Advanced Configuration

More advanced use-cases require customization of the NFVbench configuration file. The default NFVbench configuration file is obtained by using the -show-default-config option.

For example, go to the host folder mapped to a container (/root/nfvbench) and copy default NFV Bench configuration and runthe following command:

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

You can then edit the nfvbench.cfg using any Linux text editor (read and follow nested comments to do a custom configuration) and pass the new configuration file to NFVBench using the -c option.

Cisco VIM CLI

An alternate way to NFVBench CLI is to use ciscovimclient. Ciscovimclient is meant to provide an interface that is more consistent with the CiscoVIM CLI and can run remotely while the NFVBench CLI is executed on the management node.

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

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

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

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

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

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

NFVBench REST Interface

When enabled, the NFVBench container can also take benchmark request from a local REST interface. Access is only local to the management node in the current Cisco VIM version (that is the REST client must run on the management node).

Details on the REST interface calls can be found in Chapter 2, Cisco VIM REST API Resources.

Enabling or Disabling Autobackup of Management Node

Cisco VIM supports the backup and recovery of the management node. By default, the feature is enabled. Auto snapshot of the management node happens during pod management operation. You can disable the auto backup 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>
```

Take a backup of **setupdata** file and update it manually with the configuration details by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# # update the setup_data to change autobackup
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile /root/MyDir/setup_data.yaml reconfigure
```

Forwarding ELK logs to External Syslog Server

Cisco VIM supports backup and recovery of the management node. To keep the process predictable and to avoid loss of logs, the software supports the capability of forwarding the ELK logs to multiple external syslog servers (Minimum 1 and Maximum 3). The capability is introduced to enable this feature after the pod is up and running, with Cisco VIM, through the reconfigure option.

The Syslog Export reconfigure option supports the following options:

- Enable forwarding of ELK logs to External Syslog Server on a pod that is already up and running.

- Reconfigure existing External Syslog Setting to point to a different syslog cluster.

The following section needs to be configured in the setup_data.yaml file.

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

Take a backup of the setupdata file and update the file manually with the configs listed in the preceding section, then run the reconfigure command as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/
[root@mgmt1 ~]# # update the setup_data to include Syslog Export info
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile /root/MyDir/setup_data.yaml reconfigure
```

With this configuration, you should now be able to use export ELK logs to an external syslog server. On the remote host, verify if the logs are forwarded from the management node.

Adding and Reconfiguring VIM Administrators

Cisco VIM supports management of the VIM Administrators.VIM administrator has the permission to log in to the management node through SSH or the console using the configured password. By configuring to one VIM admin account, administrators do not have to share credentials. Administrators have individual accountability.

To enable one or more VIM administrators, perform the following steps:

Step 1 Take a backup of the setupdata file and update the file manually with the configurations listed as,

```
vim_admins:
- vim_admin_username: <username>
  vim_admin_password_hash: <sha512-password-hash>
- vim_admin_username: <username>
  vim_admin_password_hash: <sha512-password-hash>
- vim_admin_username: <username>
  vim_admin_password_hash: <sha512-password-hash>

The value of password hash must be in the standard sha512 format. # To generate the hash
admin_password_hash should be the output from on the management node
#  python -c "import crypt; print crypt.crypt('<plaintext password>')"
```

Step 2 Run the reconfigure commands as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/

# update the setup_data to include vim_admin info
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile /root/MyDir/setup_data.yaml reconfigure
```

Reconfigure of Proxy Post Install

During post-install you can update the http/https proxy server information that is listed in NETWORKING section of the setup_data.yaml.

To update the proxy in the post-VIM install follow these steps:

Step 1 Take a backup of the setupdata file and update the file manually with the configs listed as,

```
http_proxy_server: <a.b.c.d:port> # optional, needed if install is through internet, and the pod is
 behind a proxy
and/or
https_proxy_server: <a.b.c.d:port> # optional, needed if install is through internet, and the pod is
 behind a proxy
```

Step 2 Run the following command to reconfigure:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/

# update the setup_data to update the proxy info
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile /root/MyDir/setup_data.yaml reconfigure
```

Enabling Custom Policy for VNF Manager Post Install

During the post-installation of a cloud, Cisco VIM helps to enable a VNF Manager (such as ESC) to operate and manage tenant VMs in the OpenStack cloud, with additional privileged features.

Following are the steps to enable the custom policy for VNF Manager:

Step 1 Take a backup of the setupdata file and update the file manually with the configurations listed as,

```
ENABLE_ESC_PROV: True
```

Step 2 Run the following commands to reconfigure:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml /root/MyDir/

# update the setup_data to update the proxy info
[root@mgmt1 ~]# cd /root/MyDir/
[root@mgmt1 ~]# vi setup_data.yaml
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile /root/MyDir/setup_data.yaml reconfigure
```

Updating Containers in a Running Cisco VIM Cloud

Cisco VIM allows you to update all OpenStack and infrastructure services such as RabbitMQ, MariaDB, HAProxy, and management node containers such as Cobbler, ELK, VMTP and repo containers with almost no impact to the Cisco NFVI implementation. Updates allows you to integrate Cisco VIM patch releases without redeploying the Cisco NFVI stack from the beginning. Updates have minimal service impact because they run serially component by component one node at a time. If an error occurs during an update, auto-rollback is triggered to return the cloud to its pre-update state. After an update you can check for any functional impacts on the cloud. If everything is fine you can commit the update, which deletes the old containers and old images from the nodes. Should you see any functional cloud impact you can perform a manual rollback to start the old containers again.

Before you begin a container update, keep the following in mind:

- Updates are not supported for registry-related containers and authorized_keys.
- You cannot roll back the repo containers on the management node to an older version after they are updated because rollbacks will delete node packages and might cause the cloud to destabilize.
- To prevent double-faults, a cloud sanity check is performed before the update is started. A cloud sanity check is performed as the last step of the update.

The following table provides an overview to the methods to start the OpenStack update using Cisco VIM. The Internet options refer to management node connectivity to the Internet. If your management server lacks Internet access, you must have a staging server with Internet access to download the Cisco VIM installation artifacts to a USB stick. Cisco recommends selecting one method and staying with it for the full pod lifecycle.

Table 2: OpenStack Update Options

	Without Cisco VIM Insight	**With Cisco VIM Insight**
Without Internet	• Prepare the USB on a staging server • Plug the USB into the management node. • Follow the update steps in the update without Internet procedure.	• Prepare the USB on a staging server • Plug the USB into the management node. • Follow the update steps in the update without Internet procedure.

	Without Cisco VIM Insight	With Cisco VIM Insight
With Internet	• Download the .tgz file from the registry. • Follow the update steps in the update with Internet procedure.	• Download the .tgz file from the registry. • Follow the update steps in the update with Internet procedure.

Updating Cisco VIM Software Using a USB

The following procedure tells you how to load the Cisco VIM installation files onto a Cisco NFVI management node that does not have Internet access. Installation files include: buildnode-K9.iso, mercury-installer.tar.gz, nova-libvirt.tar, registry-2.3.1.tar.gz, and respective checksums..

Before you begin

This procedure requires a CentOS 7 staging server (VM, laptop, or UCS server) with a 64 GB USB 2.0 stick. The staging server must have Internet access (wired access is recommended) to download the Cisco VIM installation files, which you will load onto the USB stick. You then use the USB stick to load the installation files onto the management node. The installation files are around 24 GB in size, downloading them to the USB stick might take several hours, depending on the speed of your Internet connection, so plan accordingly. Before you begin, disable the CentOS sleep mode.

Step 1 On the staging server, use yum to install the following packages:

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

Step 2 Connect to the Cisco VIM software download site using a web browser and login credentials provided by your account representative and download the **getartifacts.py** script from external registry.

```
# download the new getartifacts.py file (see example below)
curl -o getartifacts.py
https://<username>:<password>@cvim-registry.com/mercury-releases/mercury-rhel7-osp8/releases/<1.0.1>/getartifacts.py

curl -o getartifacts.py-checksum.txt
https://<username>:<password>@cvim-registry.com/mercury-releases/mercury-rhel7-osp8/releases/1.0.1/getartifacts.py-checksum.txt

# calculate the checksum and verify that with one in getartifacts.py-checksum.txt
sha512sum getartifacts.py

# Change the permission of getartificats.py
chmod +x getartifacts.py
```

Step 3 Run the **getartifacts.py** script. The script formats the USB 2.0 stick and downloads the installation artifacts. You will need to provide the registry username and password, the tag ID, and the USB partition on the staging server. For example:

To identify the USB drive, execute the **lsblk** command before and after inserting the USB stick. (The command displays a list of available block devices.) The output delta will help find the USB drive location. Provide the entire drive path in the –d option, instead of any partition.

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

Note Do not remove the USB stick while the synchronization is under way.

Step 4 Verify the integrity of the downloaded artifacts and the container images:

```
# create a directory
sudo mkdir -p /mnt/Cisco

# /dev/sdc is the USB drive, same as supplied in get artifacts.py python script
sudo mount /dev/sdc1 /mnt/Cisco
cd /mnt/Cisco

# execute the verification script
./test-usb

# failures will be explicitly displayed on screen, sample success output below
# sample output of ./test-usb execution with 2.2.x release
[root@mgmtnode Cisco]# ./test-usb
INFO: Checking the integrity of this USB stick
INFO: Checking artifact buildnode-K9.iso
INFO: Checking artifact mercury-version.txt
INFO: Checking artifact registry-2.3.1.tar.gz
INFO: Checking artifact nova-libvirt-K9.tar.gz
INFO: Checking required layers:
INFO: 395 layer files passed checksum.
[root@mgmtnode Cisco]#
```

Step 5 To resolve download artifact failures, unmount the USB and run the getartifacts command again with the --retry option:

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

Step 6 Mount the USB and then run the test-usb command to validate all the files are downloaded:

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

# execute the verification script
./test-usb

# In case of failures the out of the above command will explicitly display the same on the screen
```

Step 7 After the synchronization finishes, unmount the USB stick:

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

Step 8 After the synchronization finishes, remove the USB stick from the staging server then insert it into the management node.

Step 9 Complete the following steps to import the Cisco NFVI installation artifacts onto the management node:

a) Identify the USB on the management node:

```
blkid -L Cisco-VIM
```

b) Mount the USB device on the management node:

```
 mount < /dev/sdc > /mnt/
  cd /tmp/
```

c) Extract the import_artifacts.py script:

```
tar --no-same-owner -xvzf /mnt/mercury-installer.tar.gz
```

d) Unmount the USB device:

```
umount /mnt/
```

e) Import the artifacts:

```
cd /tmp/installer-< xxxx >/tools/
./import_artifacts.sh
```

f) Change directory and remove /tmp/installer-< xxxx >

```
cd /root/
rm -fr /tmp/installer-< xxxx >
```

Step 10 Verify the image version and change ID for the software update.

```
cat /var/cisco/artifacts/mercury-version.txt
```

Step 11 Execute the update from the old working directory:

```
cd $old_workspace/installer;
ciscovim update --file /var/cisco/artifacts/mercury-installer.tar.gz
```

After the update is complete, use the newly created directory from here onwards (unless a rollback is planned).

Step 12 Commit the update by running the following command:

```
ciscovim commit # from the new workspace
```

Step 13 To revert the update changes before entering the commit command, enter:

```
ciscovim rollback # and then use older workspace
```

Note Do not run any other Cisco VIM actions while the update is underway.

In Cisco VIM 2.2.12, if updates bring in Kernel chnages, then the reboot of the compute node with VNFs in ACTIVE state is postponed. This is done to mitigate the unpredictability of data plane outage when compute nodes go for a reboot for the kernel changes to take effect, during the rolling upgrade process.

At the end of ciscovim update, the CVIM orchestrator displays the following message on the console and logs:

```
Compute nodes require reboot Kernel updated
<compute_1_with_VM_running>
<compute_3_with_VM_running>
<compute_4_with_VM_running>
<compute_12_with_VM_running>
```

After the Kernel update on Management node, reboot the compute node before proceeding. The logs for this run are available in <mgmt._ip_address>:/var/log/mercury/<UUID>

Note As the redundancy in controller, and storage nodes are built into the product, the reboot of those nodes are automatic during the software update. Also, computes that does not have any VNFs in ACTIVE state, gets automatically rebooted during software update

Updating Cisco VIM Software Using Network Installation

Step 1 From the download site that is provided by your Cisco account representative, download the mercury-installer.gz

```
curl -o mercury-installer.tar.gz
https://{username}:{password}@cvim-registry.cisco.com/
mercury-releases/mercury-rhel7-osp10/releases/{release number}/
mercury-installer.tar.gz
```

The link to the tar ball preceding is an example.

Step 2 Execute the update from the old working directory:

Note Do not run any other Cisco VIM actions while the update is underway.

```
cd /root/installer-<tagid>
ciscovim update –file /root/mercury-installer.tar.gz
```

After the update is complete, use the newly created directory from here onwards (unless a rollback is planned).

Step 3 Commit the update by running the following command:

```
 ciscovim commit
```

Step 4 To revert the update changes before entering the commit command, enter:

```
 ciscovim rollback # and then use older workspace
```

In Cisco VIM, if updates bring in Kernel changes, then the reboot of the compute node with VNFs in ACTIVE state is postponed. This is done to mitigate the unpredictability of data plane outage when compute nodes go for a reboot for the kernel changes to take effect, during the rolling upgrade process.

At the end of ciscovim update, the CVIM orchestrator displays the following message on the console and logs:

```
Compute nodes require reboot Kernel updated
<compute_1_with_VM_running>
<compute_3_with_VM_running>
<compute_4_with_VM_running>
<compute_12_with_VM_running>
```

After the Kernel update on the Management node, reboot the compute node before proceeding

The logs for this run are available in <mgmt._ip_address>:/var/log/mercury/<UUID>

Note The redundancy in controller, and storage nodes are built into the product, the reboot of those nodes are automatic during the software update. Also, computes that does not have any VNFs in ACTIVE state, gets automatically rebooted during the software update. To monitor and reboot the compute nodes through ciscovim cli, refer to the sections titled “Managing Reboot of Cisco VIM Nodes: and “Managing Reboot Status of Cisco VIM Nodes”, in the later part of this guide. It should be noted no pod management operation is allowed till reboot of all CVIM nodes are successful.

Upgrading Containers in a Running Cisco VIM Cloud

Cisco VIM 2.2 allows you to upgrade all OpenStack services, infrastructure services such as RabbitMQ, MariaDB, HAProxy, and management node containers such as Cobbler, ELK, VMTP and repo containers. You can upgrade to new releases of OpenStack without redeploying the Cisco NFVI stack from the beginning. During upgrade, you can expect limited service impact as the upgrade is run serially on component by component (one node at a time).

Cisco VIM 2.2 supports upgrade from a known version of VIM running Liberty (1.0.43) to the current version of Newton (2.2.24). As OpenStack does not support the skipping of major releases during upgrade, the VIM upgrade internally moves the stack to Mitaka and then to Newton release of OpenStack.

As part of the VIM cloud upgrade,

- The runner.py script is used to automatically upgrade the REST API server managing the VIM orchestrator.
- The setup_data.yaml is automatically translated so that the setup_data.yaml file is compatible to the target release version.

Before you begin a container update, consider the following points:

- Plan for the downtime as the upgrade involves moving the Kernel version.
- Updates are not supported for registry-related containers and authorized_keys.
- The repo containers on the management node cannot be rolled back to an older version after the upgrade, as the rollbacks will delete the node packages and destabilize the cloud.
- A cloud sanity check is performed before the update is started to prevent double-faults,. A cloud sanity check is performed as the last step of the update.

Before you begin a Pod upgrade, keep the following in mind:

- There is no roll-back in Upgrades, so it is better to stage it in the lab, and test it few times as issues related to specific customer environment might surface.
- The upgrade script, vim_upgrade_orchestrator.py, is available as part of the 2.2 artifacts and needs to be copied to /root/ location before starting the execution.
- For disconnected upgrade, 2 USBs 2.0 (64GB) should be pre-populated with artifacts from 1.5.19 and 2.2.x.
- Upgrade from 1.0.43 to 2.2.x is restricted to specific starting and end point.
- Upgrade of the cloud is supported in both connected and disconnected mode.
- In 2.2.x, UCSD is no longer supported, instead the UI has been replaced by VIM Insight; Post Upgrade customer should bring up the Insight service on its own and register the pod to it
- We recommended you not to change the upgrade mode and the install mode.
- Upgrade is a one-way operation (there no rollback); so planning should be done before executing the upgrade. In the off chance, if one faces an issue, reach out to Cisco TAC/BU to recover the cloud.
- Prior to upgrade, ensure that the size of the storage network is greater than the total number of nodes (control, compute and ceph) the pod has. This is extremely important; failure to meet this criterion can cause the failure of the pre-upgrade check, and can prevent the upgrade from moving forward.

  Following are the steps on how to get this conversion done before the upgrade commences.

  - Make a copy of the golden ~/openstack-configs/setup_data.yaml from the installer as a different file name For Example:. ~/setup_data-reconfigure-storage-net.yaml
  - Edit the storage network segment in ~/setup_data-reconfigure-storage-net.yaml copy to the desired subnet

    ```
    NETWORKING:
    .......
    - gateway: 17.16.99.1
    pool: [17.16.99.2 to 17.16.99.6]
    segments: [storage]
    ```

```
subnet: 17.16.99.0/24
vlan_id: '3005'
```

Note Do not expand the same storage subnet use the different subnet like:

- # cd ~/installer-1.0.43/tools
- ./reconfigure_storage_network.sh --setup_file /root/setup_data-reconfigure-storage-net.yaml

It takes 10 minutes to complete on a 3 controller, 3 compute, 3 storage node approximately.

At a high level, the upgrade script, vim_upgrade_orchestrator.py, is broken into three steps with logic to abort on fail. In case of failure, it is important to call Cisco support and not recover the cloud on your own.

The following are the three high level steps into which the vim_upgrade_orchestrator.py is broken into:

- **Pre-Upgrade Check**
  - Registry connectivity (if connected install).
  - Setup_data pre check: No UCSM_PLUGIN, sufficient storage pool size.
  - Backup setup_data.yaml, before performing translation.
  - Check and Update INSTALL_MODE in setup_data.yaml (connected or disconnected);
  - Check the storage network size.
  - run cloud - sanity from stable/liberty.
  - Check for reachability to all nodes including compute, controller and storage.
- **Upgrade to 1.5.18 (Mitaka):**
  - Delete UCSD instance if its running on the management node.
  - Delete old version of nova-libvirt on the management node.
  - Auto-translation of setup_data for Mitaka.
  - Upgrade to Mitaka.
  - Check for reachability to all nodes (compute, controller and storage).
- **Upgrade to 2.2.0 (Newton).**
  - Upgrade to Newton.
  - Backup of Management Node.
  - run cloud- sanity from stable/newton.
  - Check for reachability to all nodes (compute, controller and storage)

For a cloud running with VTS; additional steps need to be taken at this point, as VTC is not managed by Cisco VIM.

At a high level listed following are the manual steps:

- Power down the VTS 2.3 master and slave instances.
- Keep the VTS 2.3 XRNC/XRVR master and slave instance running.
- Bring up new VTS 2.5 master and slave instance.
- Login to master and slave VTS 2.5 UI and change admin password.
- Enable the VTS High availability with same assigned VIP IP address in VTS 2.3 deployment.
- Bring up new VTSR 2.5 master and slave instance.
- Login to VTS HA VIP IP address and execute the Underlay Loopback and OSPF Template.
- Power down the VTS 2.3 XRNC/XRVR Master and slave instance before performing vtc upgrade only.

After executing the manual steps, excute the vim_upgrade_orchestrator.py with vtcupgradeonly option to have the VIM cloud working with VTS 2.5.

- Connect to the CIMC of the management node and validate the boot-order list SDCARD as the first choice.
- Power-cycle the management node to complete the management node upgrade.
- Manually move the Pod from Software Raid to Hardware Raid.

The following table provides an overview to the methods to start the OpenStack update using Cisco VIM. The Internet options refer to management node connectivity to the Internet. If your management server lacks Internet access, you must have a staging server with Internet access to download the Cisco VIM installation artifacts to a USB stick. We recommend you to select one method and stay with it for the full pod lifecycle.

Upgrade Method	**Without Cisco VIM Insight**
Without Internet	• Prepare 2 USB 2.0 (64G) on a staging server and populate them with 1.5.x and 2.2.0 artifacts. • Plug both the USB into the management node. • Copy the vim_upgrade_orchestrator.py and follow the upgrade steps in the upgrade without Internet procedure.
With Internet	Copy the vim_upgrade_orchestrator.py and follow the upgrade steps in the upgrade without Internet procedure

Upgrading VIM Software Using a USB

The following procedure tells you how to load the Cisco VIM installation files onto a Cisco NFVI management node that does not have Internet access. Installation files include: build node-K9.iso, mercury-installer.tar.gz, nova-libvirt.tar, registry-2.3.1.tar.gz, and respective checksums.

Before you begin

This procedure requires a CentOS 7 staging server (VM, laptop, or UCS server) with two 64 GB USB 2.0 stick. The staging server must have Internet access(wired access is recommended) to download the Cisco VIM installation files, which you will load onto the USB stick. You then use the USB stick to load the installation files onto the management node. The installation files are around 24 GB in size, downloading them to the USB stick might take several hours, depending on the speed of your Internet connection, so plan accordingly. Before you start, disable the CentOS sleep mode.

Step 1 On the staging server, use yum to install the following packages:

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

Step 2 Connect to the Cisco VIM software download site using a web browser and log in to the username and the password that are provided by your account representative and download the **getartifacts.py** script from external registry.

```
# download the new getartifacts.py file (see example below) curl -o getartifacts.py
https://<username>:<password>@cvim-registry.com/mercury-releases/mercury-rhel7-osp9/releases/<1.5.2>/getartifacts.py

curl -o getartifacts.py-checksum.txt
https://<username>:<password>@cvim-registry.com/mercury-releases/mercury-rhel7-osp9/releases/1.5.2/getartifacts.py-checksum.txt

# calculate the checksum and verify that with one in getartifacts.py-checksum.txt sha512sum
getartifacts.py

# Change the permission of getartificats.py
chmod +x getartifacts.py
```

Step 3 Run the **getartifacts.py** script. The script formats the USB 2.0 stick and downloads the installation artifacts. You will need to provide the registry username and password, the tag ID, and the USB partition on the staging server. For example:

To identify the USB drive, execute the **lsblk** command before and after inserting the USB stick. (The command displays a list of available block devices.) The output data helps find the USB drive location. Provide the entire drive path in the –d option, instead of any partition.

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

Note Do not remove the USB stick while the synchronization is going on.

Step 4 Verify the integrity of the downloaded artifacts and the container images:

```
# create a directory sudo mkdir -p /mnt/Cisco

# /dev/sdc is the USB drive, same as supplied in get artifacts.py python script sudo mount /dev/sdc1
 /mnt/Cisco
cd /mnt/Cisco

# execute the verification script
./test-usb

# failures will be explicitly displayed on screen, sample success output below
# sample output of ./test-usb execution with 2.2.x release [root@mgmtnode Cisco]# ./test-usb
INFO: Checking the integrity of this USB stick INFO: Checking artifact buildnode-K9.iso
INFO: Checking artifact mercury-version.txt INFO: Checking artifact registry-2.3.1.tar.gz INFO:
Checking artifact nova-libvirt-K9.tar.gz INFO: Checking required layers:
INFO: 395 layer files passed checksum. [root@mgmtnode Cisco]#
```

Step 5 To resolve download artifact failures, unmount the USB and run the getartifacts command again with the --retry option:

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

Step 6 Mount the USB and then run the test-usb command to validate all the files are downloaded:

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

# execute the verification script
./test-usb

# In case of failures the out of the above command will explicitly display the same on the screen
```

Step 7 After the synchronization finishes, unmount the USB stick:

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

Step 8 After the synchronization finishes, remove the USB stick from the staging server then insert it into the management node.

Step 9 Repeat step 2 to step 8 for pre-population of CVIM 2.2.x artifacts onto the second USB

Step 10 Insert the 2 pre-populated USBs into the management node of the pod running 1.0.41.

Step 11 Copy the vim_upgrade_orchestrator.py script available in CVIM 2.2 artifacts in the /root/ folder of the management node of the pod running 1.0.41

Step 12 Execute the update from the /root/ location:

```
# cd /root/
# ./vim_upgrade_orchestrator.py –i disconnected [-y]  # -y if you don’t want any interactive mode

After the upgrade is complete, use the newly created directory from here onwards.
```

Note Upgrade process takes several hours (> 6 hours), so excute this process in a VNC. Do not run any other Cisco VIM actions while the upgrade is underway.

Step 13 Copy the management node backup that is created during the upgrade, into a separate server through rsync (see chapter 12 for details).

Step 14 Check that the SDCARD is state as priority 1 for the boot order, from the CIMC of the management node. If not, set it accordingly. Reboot the management node, and wait for it to come all the way up.

Step 15 If VTS is running on the pod, manually transfer the VTC to 2.5

Step 16 Execute the update from the /root/ location:

```
# cd /root/
# ./vim_upgrade_orchestrator.py –i disconnected -s VTCSSHUSERNAME -p VTCSSHPASSWORD –vtsupgradeonly
  [-y]
```

Note "-y" is if you don't want any interactive mode

Step 17 Manually switch the management node of the pod to Hardware Raid. At a high level following steps need to be followed on the management node:

- Go to CIMC and set HDD as the top boot order.
- Enable the HBA on Bios setting from the CIMC. For procedure details, refer to Setting up the UCS C-Series Pod section in the 2.2 Install Guide.

- Initiate backup and restore of the management node with the current upgraded image version. For details on how to do it, refer to Chapter 12, where the backup and restore of the management node is listed.

Step 18 Move the pod to Hardware Raid, execute the following:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>

    Update the kickstart line in setup_data in /root/Save/:
  kickstart:
   block_storage: storage-flexflash-c240m4.ks
   compute: compute-flexflash-c220m4.ks
   control: control-flexflash-c220m4.ks
to
   kickstart:
     control: ucs-b-and-c-series.ks
     compute: ucs-b-and-c-series.ks
block_storage: storage-flexflash-c240m4.ks

 [root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> run -perform 1
```

- Then for each compute node, do a remove, followed by add of the same node; (don't forget to power on the server after the remove); you can do remove followed by add of more than 1 compute node at a time.
- For every controller node, execute the replace controller, one at a time

Note To verify that the systems (compute and controllers) have moved to harware raid, execute the following in each server after the above steps

```
# /opt/MegaRAID/storcli/storcli64 /c0 /v0 show | grep Status
 and verify that "Status = Success" is displayed.
```

Upgrading Cisco VIM Software Using Network Installation

Step 1 From the download site that is provided by your Cisco account representative, download the vim_upgrade_orchestrator.py curl -o vim_upgrade_orchestrator.py https://{username}:{password}@cvim-registry.cisco.com/mercury-releases/mercury-rhel7-osp10/releases/{release number}/vim_upgrade_orchestrator.py. The link to the tar ball preceding is an example.

Step 2 Execute the upgrade from /root/ directory:

```
 $ cd /root/
$ ./vim_upgrade_orchestrator.py -i connected
```

Note Do not run any other Cisco VIM actions while the update is going on.

After the upgrades are complete, use the newly created folder.

VM Resizing

VM resize is the process of changing the flavor of an existing VM. Thus, using VM resize you can upscale a VM according to your needs. The size of a VM is indicated by the flavor based on which the VM is launched.

Resizing an instance means using a different flavor for the instance.

By default, the resizing process creates the newly sized instance on a new node, if more than one compute node exists and the resources are available. By default, the software, allows you to change the RAM size, VDISK size, or VCPU count of an OpenStack instance using **nova resize**. Simultaneous or individual adjustment of properties for the target VM is allowed. If there is no suitable flavor for the new properties of the VM, you can create a new one.

```
nova resize [--poll] <server> <flavor>
```

The resize process takes some time as the VM boots up with the new specifications. For example, the Deploying a Cisco CSR (size in MB) would take approximately 60mins. After the resize process, execute `nova resize-confirm <server>` to overwrite the old VM image with the new one. If you face any issue, you can revert to the old VM using the `nova-resize-revert <server>` command. At this point, you can access the VM through SSH and verify the correct image is configured.

Note The OpenStack **shutdown** the VM before the resize, so you have to plan for a **downtime.**

Note We recommend you not to resize a vdisk to a smaller value, as there is the risk of losing data.

Nova Migrate

The nova migrate command is used to move an instance from one compute host to another compute host. The scheduler chooses the destination compute host based on the availability of the zone settings. This process does not assume that the instance has shared storage available on the target host.

To initiate the cold migration of the VM, you can execute the following command:

```
nova migrate [--poll] <server>
```

The VM migration can take a while, as the VM boots up with the new specifications. After the VM migration process, you can execute `nova resize-confirm <server>` --to overwrite the old VM image with the new one. If you encounter an problem, use the `nova-resize-revert <server>` command to revert to the old VM image. At this point, access the VM through SSH and verify the correct image is configured.

Note The OpenStack **shutdown** the VM before the migrate, so plan for a **downtime**.

CHAPTER 2

Cisco VIM REST API

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

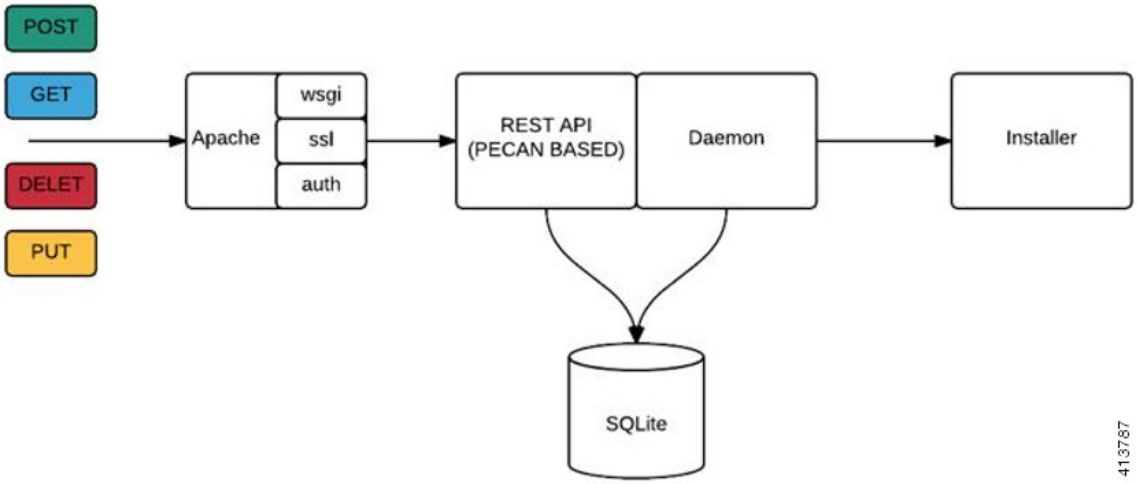
Overview to Cisco VIM REST API

Cisco VIM provides a Representational State Transfer (REST) API that is used to install, expand, and update Cisco VIM. Actions performed using the REST APIs are:

- Install Cisco VIM on Cisco NFVI pods
- Add and delete pods to and from Cisco NFVI installations
- Update Cisco VIM software
- Replace controller nodes
- Perform cloud maintenance operations
- Run cloud validations using Virtual Machine ThroughPut (VMTP), a data path performance measurement tool for OpenStack clouds

The following figure shows the Cisco VIM REST API flow.

Figure 7: Cisco VIM REST API Flow



The Cisco VIM REST API security is provided by the Secure Sockets Layer (SSL) included on the Apache webserver. The Pecan-based web application is called by mod_wsgi, which runs the Rest API server. The Pecan REST API server requires a username and password to authorize the REST API server requests. Apache handles the authorization process, which authorizes the request to access the Pecan web application. Use the Cisco VIM API to upload a new setup_data.yaml file, and start, stop, and query the state of the installation. You can use it to manage the cloud, add and remove compute and Ceph nodes, and replace the controller nodes. A REST API to launch VMTP (L2/L3 data plane testing) and CloudPulse is also provided.

The Cisco VIM REST API is enabled by default in the management node, if you are using the supplied Cisco VIM buildnode.iso. You can access API server on the br_api interface on port 8445. Authentication is enabled by default in the web service.

The API endpoints can be reached with the following URL format:

https://<Management_node_api_ip>:8445

The API endpoint expects a basic authentication which is enabled by default in the management node. The authentication credentials are found in /opt/cisco/ui_config.json in the management node.

The following contents show the Sample ui_config.json:

```
{
  "Kibana-Url": "http://10.10.10.10:5601",
  "RestAPI-Url": "https:// 10.10.10.10:8445",
  "RestAPI-Username": "admin",
  "RestAPI-Password": "a96e86ccb28d92ceb1df",
  "RestDB-Password": "e32de2263336446e0f57",
  "BuildNodeIP": "10.10.10.10"
}
```

Cisco VIM REST API Resources

Setupdata

REST wrapper for setupdata. Provides methods for listing, creating, modifying and deleting setupdata.

Retrieving the setupdata

Resource URI

Verb	URI
GET	/v1/setupdata

Example

JSON Request

```
GET /v1/setupdata
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"setupdatas": [{
     "status": "Active",
     "name":"GG34",
     "uuid": "123"
     "meta":{
           "user":"root"
     },
     "jsondata":{
       .......
     }
 }]}
```

Creating the setupdata

Resource URI

Verb	URI
POST	/v1/setupdata

Example

JSON Request

```
POST /v1/setupdata
Accept: application/json

{
    "name":"GG34",
    "uuid": "123"
    "meta":{
         "user":"root"
  },
  "jsondata":{
    .......
  }
}
```

JSON Response

```
201 OK
Content-Type: application/json
{
     "status": "Active",
     "name":"GG34",
     "uuid": "123"
     "meta":{
         "user":"root"
  },
  "jsondata":{
    .......
  }
}

400 Bad Request
Content-Type: application/json
{
    "debuginfo": null
    "faultcode": "Client"
    "faultstring": "Error"
}

409 CONFLICT
Content-Type: application/json
{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Error"
}
```

Retrieving a single setupdata

Resource URI

Verb	URI
GET	/v1/setupdata/(id)

Property:

id - the id of the setupdata to be queried.

Example

JSON Request

```
GET /v1/setupdata/123
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
     "status": "Active",
     "name":"GG34",
     "uuid": "123"
     "meta":{
         "user":"root"
  },
  "jsondata":{
    .......
  }
}

404 NOT FOUND
Content-Type: application/json
```

```
{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Setupdata could not be found."
}
```

Updating a setupdata

Resource URI

Verb	URI
PUT	/v1/setupdata/(id)

Property:

id - the id of the setupdata to be updated.

Example

JSON Request

```
PUT /v1/setupdata/123
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
     "status": "Active",
     "name":"GG34",
     "uuid": "123"
     "meta":{
         "user":"root"
  },
  "jsondata":{
    .......
  }
}

404 NOT FOUND
Content-Type: application/json
{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Setupdata could not be found."
}
```

Deleting a setupdata

Resource URI

Verb	URI
DELETE	/v1/setupdata/(id)

Property:

id - the id of the setupdata to be deleted.

Example

JSON Request

```
DELETE /v1/setupdata/123
Accept: application/json
```

JSON Response

```
204 NO CONTENT
Returned on success

404 NOT FOUND
Content-Type: application/json
{
    "debuginfo": null
    "faultcode": "Client"
    "faultstring": "Setupdata could not be found."
}
400 BAD REQUEST
Content-Type: application/json

{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Setupdata cannot be deleted when it is being used by an installation"
}
```

Install resource

REST wrapper for install. Provides methods for starting, stopping, and viewing the status of the installation process.

Return a list of installation

Resource URI

Verb	URI
GET	/v1/install

Example

JSON Request

```
GET /v1/install
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"installs": [{
      "ceph": "Skipped",
      "uuid": "123",
      "setupdata": "345",
      "vmtpresult": "{
         "status": "PASS",
         "EXT_NET": []
      }",
      "baremetal": "Success",
      "orchestration": "Success",
      "validationstatus": "{
         "status": "PASS",
         "Software_Validation": [],
         "Hardware_Validation": []
      }",
      "currentstatus": "Completed",
      "validation": "Success",
      "hostsetup": "Success",
      "vmtp": "Skipped"
    }]
}
```

Create an installation

Resource URI

Verb	URI
POST	/v1/install

Example

JSON Request

```
GET /v1/install
Accept: application/js
{
     "setupdata": "123",
     "stages": [
       "validation",
       "bootstrap",
       "runtimevalidation",
       "baremetal",
       "orchestration",
       "hostsetup",
       "ceph",
       "vmtp"
     ]
}
```

JSON Response

```
201  CREATED
Content-Type: application/json
{
    "ceph": "Skipped",
    "uuid": "123",
    "setupdata": "345",
    "vmtpresult": "{
       "status": "PASS",
       "EXT_NET": []
    }",
    "baremetal": "Success",
    "orchestration": "Success",
    "validationstatus": "{
       "status": "PASS",
       "Software_Validation": [],
       "Hardware_Validation": []
    }",
     "currentstatus": "Completed",
     "validation": "Success",
     "hostsetup": "Success",
     "vmtp": "Skipped"
 }


409 CONFLICT
Content-Type: application/json
{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Install already exists"
}
```

Retrieve the installation

Resource URI

Verb	URI
GET	/v1/install/{id}

Property:

id - the id of the install to be queried.

Example

JSON Request

```
GET /v1/install/345
Accept: application/js
```

JSON Response

```
200  OK
Content-Type: application/json
{
    "ceph": "Skipped",
    "uuid": "123",
    "setupdata": "345",
    "vmtpresult": "{
       "status": "PASS",
       "EXT_NET": []
    }",
    "baremetal": "Success",
    "orchestration": "Success",
    "validationstatus": "{
      "status": "PASS",
      "Software_Validation": [],
      "Hardware_Validation": []
    }",
    "currentstatus": "Completed",
    "validation": "Success",
    "hostsetup": "Success",
    "vmtp": "Skipped"
}


404 NOT FOUND
Content-Type: application/json
{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Install doesn’t exists"
}
```

Stop the installation

Resource URI

Verb	URI
DELETE	/v1/install/{id}

Property:

id - the id of the install to be stopped.

Example

JSON Request

```
DELETE /v1/install/345
Accept: application/js
```

JSON Response

```
204 NO CONTENT
Content-Type: application/json

404 NOT FOUND
Content-Type: application/json
{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Install doesn’t exists"
}
```

Nodes

Getting a list of nodes

Resource URI

Verb	URI
GET	/v1/nodes

Example

JSON Request

```
Get /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "nodes": [
        [
           "status": "Active",
           "uuid": "456",
           "setupdata": "123",
           "node_data": "{
             "rack_info": {
                "rack_id": "RackA"
             },
             "cimc_info": {
               "cimc_ip": "10.10.10.10"
             },
             "management_ip": "7.7.7.10"
             }",
             "updated_at": null,
             "mtype": "compute",
             "install": "345",
             "install_logs": "logurl",
             "created_at":"2016-0710T06:17:03.761152",
             "name": " compute-1"
             }
          ]
}
```

Add new nodes

The nodes are in compute or block_storage type. Before adding the nodes to the system, the name of the nodes and other necessary information like cimc_ip and rackid must be updated in the setupdata object. If the setupdata object is not updated, the post call will not allow you to add the node.

Resource URI

Verb	URI
POST	/v1/nodes

Example

JSON Request

```
POST /v1/nodes
Accept: application/js
{
    "name" : "compute-5"
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
     "status": "ToAdd",
     "uuid": "456",
     "setupdata": "123",
     "node_data": "{
        "rack_info": {
        "rack_id": "RackA"
        },
        "cimc_info": {
         "cimc_ip": "10.10.10.10"
        },
        "management_ip": "7.7.7.10"
        }",
        "updated_at": null,
        "mtype": "compute",
        "install": "345",
        "install_logs": "logurl",
        "created_at":"2016-0710T06:17:03.761152",
        "name": " compute-1"
}
```

Retrieve information about a particular node

Resource URI

Verb	URI
GET	/v1/nodes{id}

Property:

id - the id of the node to be queried.

Example

JSON Request

```
POST /v1/nodes
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json
{
     "status": "Active",
     "uuid": "456",
     "setupdata": "123",
     "node_data": "{
       "rack_info": {
        "rack_id": "RackA"
       },
       "cimc_info": {
         "cimc_ip": "10.10.10.10"
       },
       "management_ip": "7.7.7.10"
       }",
       "updated_at": null,
       "mtype": "compute",
       "install": "345",
       "install_logs": "logurl",
       "created_at":"2016-0710T06:17:03.761152",
       "name": " compute-1"
}

404 NOT FOUND
Content-Type: application/json
{
    "debuginfo": null
    "faultcode": "Client"
    "faultstring": "Node doesn’t exists"
}
```

Remove a node

The node that must be deleted must be removed from the setupdata object. Once the setupdata object is updated, you can safely delete of the node. The node object will not be deleted until it calls the remove node backend and succeeds.

Resource URI

Verb	URI
DELETE	/v1/nodes{id}

Property:

id - the id of the node to be removed.

Example

JSON Request

```
DELETE /v1/nodes/456
Accept: application/js
```

JSON Response

```
204 ACCEPTED
Content-Type: application/json

404 NOT FOUND
Content-Type: application/json
{
     "debuginfo": null
     "faultcode": "Client"
     "faultstring": "Node doesn’t exists"
}
```

For clearing the database and deleting the entries in the nodes, the delete api is called with special parameters that are passed along with the delete request. The JSON parameters are in the following format.

JSON Request

```
DELETE /v1/nodes/456
Accept: application/js
{
    "clear_db_entry":"True"\
}
```

JSON Response

```
204 ACCEPTED
Content-Type: application/json

404 NOT FOUND
Content-Type: application/json
{
    "debuginfo": null
    "faultcode": "Client"
    "faultstring": "Node doesn’t exists"
}
```

Note This is done only if the node is deleted from the REST API database. The failure reason of the node must be rectified manually apart from the API. True is a string and not a boolean in the above line.

Replace a controller

Resource URI

Verb	URI
PUT	/v1/nodes{id}

Property:

id - the id of the controller to be replaced.

Example

JSON Request

```
PUT /v1/nodes/456
Accept: application/js
```

JSON Response

```
200 OK
Content-Type: application/json

404 NOT FOUND
Content-Type: application/json
{
    "debuginfo": null
    "faultcode": "Client"
    "faultstring": "Node doesn't exists"
}
```

Offline validation

REST wrapper does the offline validation of setupdata. This will only do S/W Validation of the input setupdata.

Create an offline validation operation

Resource URI

Verb	URI
POST	/v1/offlinevalidation

Example

JSON Request

```
POST /v1/offlinevalidation
Accept: application/json
{
      "jsondata": ".. .. .."
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
     "status": "NotValidated",
     "uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69",
     "created_at": "2016-02-03T02:05:28.384274",
     "updated_at": "2016-02-03T02:05:51.880785",
     "jsondata": "{}",
     "validationstatus": {
        "status": "PASS",
        "Software_Validation": [],
        "Hardware_Validation": []
      }
}
```

Retrieve the results of offline validation

Resource URI

Verb	URI
GET	/v1/offlinevalidation

Property:

id - the id of the node to be queried.

Example

JSON Request

```
GET /v1/offlinevalidation/789
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "status": " ValidationSuccess",
    "uuid": "bb42e4ba-c8b7-4a5c-98b3-1f384aae2b69",
    "created_at": "2016-02-03T02:05:28.384274",
    "updated_at": "2016-02-03T02:05:51.880785",
    "jsondata": "{}",
    "validationstatus": {
       "status": "PASS",
       "Software_Validation": [],
       "Hardware_Validation": []
     }
}
```

Update

Start an update process

Resource URI

Verb	URI

POST	/v1/update

Parameters:

- fileupload - "tar file to upload"
- filename - "Filename being uploaded"

Example

JSON Request

```
curl -sS -X POST --form
"fileupload=@Test/installer.good.tgz" --form
"filename=installer.good.tgz"
https://10.10.10.8445/v1/update
```

Note This curl request is done as a form request.

JSON Response

```
200 OK
Content-Type: application/json
{
    "update_logs": "logurl",
    "update_status": "UpdateSuccess",
    "update_filename": "installer-4579.tgz",
    "created_at": "2016-07-10T18:33:52.698656",
    "updated_at": "2016-07-10T18:54:56.885083"
}

409 CONFLICT
Content-Type: application/json
{
    "debuginfo": null
    "faultcode": "Client"
    "faultstring": "Uploaded file is not in tar format"
 }
```

Rollback an update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
PUT /v1/update
Accept: application/json
{
     "action":"rollback"
}
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "update_logs": "logurl",
    "update_status": "ToRollback",
    "update_filename": "installer-4579.tgz",
    "created_at": "2016-07-10T18:33:52.698656",
    "updated_at": "2016-07-10T18:54:56.885083"
}
```

Commit an update

Resource URI

Verb	URI
PUT	/v1/update

Example

JSON Request

```
PUT /v1/update
Accept: application/json
{
"action":"commit"
}
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "update_logs": "logurl",
    "update_status": "ToCommit",
    "update_filename": "installer-4579.tgz",
    "created_at": "2016-07-10T18:33:52.698656",
    "updated_at": "2016-07-10T18:54:56.885083"
}
```

Retrieve the details of an update

Resource URI

Verb	URI
GET	/v1/update

Example

JSON Request

```
GET /v1/update
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "update_logs": "logurl",
    "update_status": "UpdateSuccess",
    "update_filename": "installer-4579.tgz",
    "created_at": "2016-07-10T18:33:52.698656",
    "updated_at": "2016-07-10T18:54:56.885083"
}
```

Secrets

Retrieve the list of secrets associated with the OpenStack Setup

You can retrieve the set of secret password associated with the OpenStack setup using the above api. This gives the list of secrets for each service in OpenStack.

Resource URI

Verb	URI
GET	/v1/secrets

Example

JSON Request

```
GET /v1/secrets
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"HEAT_KEYSTONE_PASSWORD": "xxxx",
"CINDER_KEYSTONE_PASSWORD": "xxxxx",
….
….
"RABBITMQ_PASSWORD": "xxxxx"
}
```

OpenStack Configs

Retrieve the list of OpenStack configs associated with the OpenStack Setup

You can retrieve the set of OpenStack configs associated with the OpenStack setup using the above api. This gives the current settings of different configs like verbose logging, debug logging for different OpenStack services.

Resource URI

Verb	URI
GET	/v1/openstack_config

Example

JSON Request

```
GET /v1/openstack_config
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
"CINDER_DEBUG_LOGGING": false,
"KEYSTONE_DEBUG_LOGGING": false,
….
….
"NOVA_VERBOSE_LOGGING": true
}
```

Version

Retrieve the version of the Cisco Virtualized Infrastructure Manager.

Resource URI

Verb	URI
GET	/v1/version

Example

JSON Request

```
GET /v1/version
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{"version": "1.9.1"}
```

Health of the Management Node

Retrieve the health of the Management node

This api can be used to retrieve the health of the management node. It checks various parameters like partitions, space and so on.

Resource URI

Verb	URI
GET	/v1/health

Example

JSON Request

```
GET /v1/health
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "status": "PASS",
    "BuildNode Validation": {
        "Check Docker Pool Settings": {"status": "Pass", "reason": "None"}
        ….
        ….
    }
}
```

Hardware Information

REST wrapper to do hardware information of setupdata. This will return the hardware information of all hardware available in the setupdata.

Create a HWinfo operation

Resource URI

Verb	URI
GET	/v1/hwinfo

Example

JSON Request

```
POST /v1/hwinfo
Accept: application/json
{
       "setupdata":"c94d7973-2fcc-4cd1-832d-453d66e6b3bf"
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
  "status": "hwinfoscheduled",
  "uuid": "928216dd-9828-407b-9739-8a7162bd0676",
  "setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf",
  "created_at": "2017-03-19T13:41:25.488524",
  "updated_at": null,
  "hwinforesult": ""
}
```

Retrieve the results of Hwinfo Operation

Resource URI

Verb	URI
GET	/v1/hwinfo/{id}

Property:

id - the id of the node to be queried.

Example

JSON Request

```
GET /v1/hwinfo/789
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
  "status": "hwinfosuccess",
  "uuid": "928216dd-9828-407b-9739-8a7162bd0676",
  "setupdata": "c94d7973-2fcc-4cd1-832d-453d66e6b3bf",
  "created_at": "2017-03-19T13:41:25.488524",
  "updated_at": "2017-03-19T13:42:05.087491",
  "hwinforesult": "{\"172.29.172.73\": {\"firmware\": …………..
  …………
  ……………..
}
```

Release mapping Information

This api is used to see the list of Features included and list of options which can be reconfigured in the Openstack Setup.

Retrieve the release mapping information

Resource URI

Verb	URI
GET	/v1/releasemapping

JSON Request

```
GET /v1/releasemapping
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
[
  {
    "SWIFTSTACK": {
      "feature_status": true,
      ],
      "desc": "swift stack feature"
    }
  },……..
  …………..
}
```

POST Install operations

The following are the post install operations that can be carried on once the OpenStack installation is carried out successfully. It uses a common api. So only one operation is given as an example below:

1. reconfigure,
2. reconfigure -regenerate passwords
3. reconfigure -setpasswords,setopenstack_configs,
4. check-fernet-keys
5. period-rotate-fernet-keys
6. resync-fernet-keys
7. rotate-fernet-keys

Create a post install operation

Resource URI

Verb	URI
POST	/v1/misc

Example

JSON Request

```
POST /v1/misc
Accept: application/json
{"action": {"reconfigure": true}}
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
  "uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733",
  "created_at": "2017-03-19T14:03:39.723914",
  "updated_at": null,
  "operation_status": "OperationScheduled",
  "operation_logs": "",
  "operation_name": "{"reconfigure": true}"
}
```

Retrieve a status of the post install operation

Resource URI

Verb	URI
GET	/v1/misc

Example

JSON Request

```
GET /v1/misc
Accept: application/json
```

JSON Response

```
201 CREATED
Content-Type: application/json
{
  "uuid": "7e30a671-bacf-4e3b-9a8f-5a1fd8a46733",
  "created_at": "2017-03-19T14:03:39.723914",
  "updated_at": "2017-03-19T14:03:42.181180",
  "operation_status": "OperationRunning",
  "operation_logs": "xxxxxxxxxxxxxxxxx",
  "operation_name": "{\"reconfigure\": true}"
}
```

In VIM 2.2, additional Rest APIs are introduced to support NFVBench, query hardware information and to get a list of optional and mandatory features that the pod supports.

Listed below are the details of the API.

NFVBench Network Performance Testing

Create NFVBench Run

Starts network performance test with provided configuration.

REST API To Create Fixed Rate Test

Verb	URI
Post	v1/nfvbench/ create_ndr_pdr_test

Example

JSON Request

```
POST Request URL
/v1/nfvbench/create_fixed_rate_test
JSON Request:
{“nfvbench_request”:
{
    "duration_sec": 20,
    "traffic_profile":  [
         {
            "name": "custom",
            "l2frame_size": [
               "64",
               "IMIX",
               "1518"
            ]
         }
   ],
   "traffic": {
      "bidirectional": true,
      "profile": "custom"
   },
   "flow_count": 1000
}
}
```

JSON Response

```
201 CREATED
Content-Type: application/json
 {
      "status": "not_run",
"nfvbench_request":
‘{
    "duration_sec": 20,
    "traffic_profile":  [
         {
            "name": "custom",
            "l2frame_size": [
               "64",
               "IMIX",
               "1518"
            ]
         }
   ],
   "traffic": {
      "bidirectional": true,
      "profile": "custom"
   },
   "flow_count": 1000
}’,
"created_at": "2017-08-16T06:14:54.219106",
"updated_at": null,
"nfvbench_result": "",
"test_name": "Fixed_Rate_Test"
}
```

Status Polling

Polling of NFVbench run status which is one of nfvbench_running, nfvbench_failed, nfvbench_completed.

Resource URI

Verb	URI
GET	v1/nfvbench/<test_name>

REST API To Get Fixed Rate Test Result

```
GET Request URL
/v1/upgrade/get_fixed_rate_test_result
JSON Response:
 Check If NFVbench Test is running
  200 OK
  Content-Type: application/json
{
     "status": "nfvbench_running",
     "nfvbench_request": '{"traffic": {"bidirectional": true, "profile": "custom"},
"rate": "1000000pps",
"traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}], "duration_sec": 60,
"flow_count": 1000}',
"nfvbench_result": ""
     "created_at": "2017-05-30T21:40:40.394274",
        "updated_at": "2017-05-30T21:40:41.367279",
}

Check If NFVbench Test is completed
  200 OK
  Content-Type: application/json
   {
"status": "nfvbench_completed",
"nfvbench_request": '{"traffic": {"bidirectional": true, "profile": "custom"},
 "rate": "1000000pps",
"traffic_profile": [{"l2frame_size": ["1518"], "name": "custom"}], "duration_sec": 60,
"flow_count": 1000}',
"nfvbench_result": '{"status": "PROCESSED", "message": {"date": "2017-08-15 23:15:04",
"nfvbench_version": "0.9.3.dev2", ….}
"created_at": "2017-05-30T21:40:40.394274",
"updated_at": "2017-05-30T22:29:56.970779",
   }
```

REST API to create NDR/PDR Test

```
POST Request URL
/v1/nfvbench/create_ndr_pdr_test

Accept: application/json
{“nfvbench_request”:
{
    "duration_sec": 20,
    "traffic_profile":  [
         {
            "name": "custom",
            "l2frame_size": [
               "64",
               "IMIX",
               "1518"
            ]
         }
   ],
   "traffic": {
      "bidirectional": true,
      "profile": "custom"
   },
   "flow_count": 1000
}
}

JSON Response
201 CREATED
Content-Type: application/json
 {
    "status": "not_run",
“nfvbench_request”:
‘{
    "duration_sec": 20,
    "traffic_profile":  [
         {
            "name": "custom",
            "l2frame_size": [
               "64",
               "IMIX",
               "1518"
            ]
         }
   ],
   "traffic": {
      "bidirectional": true,
      "profile": "custom"
   },
   "flow_count": 1000
}’

"created_at": "2017-08-16T07:18:41.652891",
"updated_at": null,
     "nfvbench_result": "",
     "test_name": "NDR_PDR_Test"
}
```

REST API To Get NDR/PDR Test Results

```
GET Request URL
/v1/ nfvbench/get_ndr_pdr_test_result

JSON Response:
 If NFVbench NDR/PDR test is running
    200 OK
    Content-Type: application/json
{
     "status": "nfvbench_running",
 "nfvbench_request": '{"duration_sec": 20,
 "traffic": {"bidirectional": true, "profile": "custom"},
 "traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name": "custom"}],
"flow_count": 1000}',
 "nfvbench_result": ""
"created_at": "2017-08-16T07:18:41.652891",
"updated_at": "2017-09-30T22:29:56.970779",

}

If NFVbench NDR/PDR test is completed
  200 OK
  Content-Type: application/json
{
 "status": "nfvbench_completed",
 "nfvbench_request": '{"duration_sec": 20,
"traffic": {"bidirectional": true, "profile": "custom"},
"traffic_profile": [{"l2frame_size": ["64", "IMIX", "1518"], "name": "custom"}], "flow_count":
 1000}',
     "nfvbench_result": '{"status": "PROCESSED",...}'
"created_at": "2017-08-16T07:18:41.652891",
"updated_at": "2017-09-30T22:29:56.970779",

}
```

REST API to Get Node Hardware Information

Rest API helps you to get the hardware information of all the nodes in the POD through CIMC/UCSM.

- Total Memory
- Firmware Info (Model, Serial Number)
- CIMC IP

```
GET Request URL
/v1/hwinfo
Output Response
{
    "hwinforesult": "{"control-server-2": {"memory": {"total_memory": "131072"},
    "firmware": {"serial_number": "FCH1905V16Q", "fw_model": "UCSC-C220-M4S"},
    "cimc_ip": "172.31.230.100", "storage": {"num_storage": 4},
    "cisco_vic_adapters": {"product_name": "UCS VIC 1225"},
     "cpu": {"number_of_cores": "24"}, "power_supply": {"power_state": "on"}}
    …
 }
```

REST API to Get Mandatory Features Mapping

```
POST Request URL
/v1/releasemapping/mandatory_features_mapping

JSON Response:
{
    "mandatory": {
        "networkType": {
            "C": {
                "feature_status": true,
               "values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],

                "insight_label": "Tenant Network",
                "desc": "Tenant Network"
            },
            "B": {
                "feature_status": true,
               "values": [{"name": "VXLAN/Linux Bridge", "value": "VXLAN/Linux Bridge"},],

                "insight_label": "Tenant Network",
                "desc": "Tenant Network"
            }
        },
        "cephMode": {
            "all": {
                "feature_status": true,
                "values": [{"name": "Central", "value": "Central"},],
                "insight_label": "Ceph Mode",
                "desc": "Ceph Mode"
            }
        },
        "podType": {
            "C": {
                "feature_status": true,
                "values": [{"name": "Fullon", "value": "fullon"},],
                "insight_label": "POD Type",
                "desc": "POD Type"
            },
            "B": {
                "feature_status": true,
                "values": [{"name": "Fullon", "value": "fullon"},],
                "insight_label": "POD Type",
                "desc": "POD Type"
            }
        },
        "installMode": {
            "all": {
                "feature_status": true,
                "values": [{"name": "Connected", "value": "connected"}, ],
                "insight_label": "Install Mode",
                "desc": "Install Mode"
            }
        }
    },
    "platformType": [{"name": "B-series", "value": "B"}, {"name": "C-series", "value":
"C"}],
    "postinstalllinks": {
       "view_cloudpulse": {"alwayson": true, "feature_status": true, "platformtype": "all",
 "insight_label": "Run VMTP", "desc": "Cloudpluse"},
        "password_reconfigure": {"alwayson": true, "feature_status": true, "platformtype":
 "all", "insight_label": "Reconfigure Passwords", "desc": "Reconfigure Passwords"}
    }
}
```

REST API to Get Optional Features Mapping

```
POST Request URL
/v1/releasemapping/optional_features_mapping

JSON Response:
 [
    {
        "SWIFTSTACK": {
            "feature_status": true,
            "insight_label": "Swiftstack",
            "repeated_redeployment": true,
           "reconfigurable": ["cluster_api_endpoint", "reseller_prefix", "admin_password",
 "protocol"],
            "desc": "swift stack feature"
        }
    },
    {
        "heat": {
            "feature_status": true,
            "insight_label": "Heat",
            "repeated_redeployment": false,
            "reconfigurable": ["all"],
            "desc": "Openstack HEAT service"
        }
    },
….. other features
]
```

Disk Maintenance information

REST wrapper to query information about RAID disks on Pod nodes. This will return the RAID disk information of all or a selection of RAID disks available in the Pod.

The disk management extension to the VIM REST API enables support for Disk Management actions

End Point	Type	Valid Args	Valid Arg Values	Example
/diskmgmt	GET	None	None	/v1/diskmgmt/
/diskmgmt/ check_disks	GET	None, args	All, control, compute	/v1/diskmgmt/check_disks/ /v1/diskmgmt/check_disks/? args=control,compute
/diskmgmt/ replace_disks	GET	None, args	All, control, compute	/v1/diskmgmt/replace_disks/ /v1/diskmgmt/replace_disks/? args=control,compute
/diskmgmt/server	GET	server_list, action	List of valid server names, check_disks, replace_disks	/v1/diskmgmt/server/?server_list= srv1,srv2&action=check_disks /v1/diskmgmt/server/?server_list= srv1&action=replace_disks

Get a Check disk operation

Resource URI

Verb	URI
GET	/v1/diskmgmt

Example

JSON Request

```
GET /v1/diskmgmt Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [],
    "bad_disks_results_list": [],
    "fcfg_disks_results_list": [],
    "raid_results_list": [
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "EMC-Testbed",
            "role": "management",
            "server": "localhost"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-20",
            "role": "control",
            "server": "15.0.0.7"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-21",
            "role": "control",
            "server": "15.0.0.8"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-22",
            "role": "control",
            "server": "15.0.0.5"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-23",
            "role": "compute",
            "server": "15.0.0.6"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-24",
            "role": "compute",
            "server": "15.0.0.10"
        }
    ],
    "rbld_disks_results_list": [],
    "spare_disks_results_list": []
}
```

Get a Check disk operation for compute nodes

Resource URI

Verb	URI
GET	/v1/diskmgmt/check_disks/?args={all,control,compute}

Example

JSON Request

```
GET /v1/diskmgmt/check_disks/?args=compute
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [],
    "bad_disks_results_list": [],
    "fcfg_disks_results_list": [],
    "raid_results_list": [
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-23",
            "role": "compute",
            "server": "15.0.0.6"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-24",
            "role": "compute",
            "server": "15.0.0.10"
        }
    ],
    "rbld_disks_results_list": [],
    "spare_disks_results_list": []
}
```

Post a replace disk operation

Resource URI

Verb	URI
GET	/v1/diskmgmt/replace_disks/?args={all,control,compute}

Example

JSON Request

```
Get /v1/diskmgmt/replace_disks/?args=compute Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [
        {
            "disk slot": "1",
            "host": "i13-21",
            "replace status": "Success",
            "role": "control",
            "server": "15.0.0.8"
        }
    ]
}
```

Get a check disk operation for a particular server

Resource URI

Verb	URI
GET	v1/diskmgmt/server/?server_list={server_list}&action=check_disks

Example

JSON Request

```
GET
/v1/diskmgmt/server/?server_list=i13-21,i13-23&action=check_disks
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [
        {
            "disk slot": "1",
            "disk state": " UGood",
            "host": "i13-21",
            "role": "control",
            "server": "15.0.0.8"
        }
    ],
    "bad_disks_results_list": [],
    "fcfg_disks_results_list": [],
    "raid_results_list": [
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "NdAtn",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Dgrd",
            "host": "i13-21",
            "role": "control",
            "server": "15.0.0.8"
        },
        {
            "Num PDs": 4,
            "Num VDs": 1,
            "RAID health": "Opt",
            "RAID level": "RAID10",
            "RAID type": "HW",
            "VD health": "Optl",
            "host": "i13-23",
            "role": "compute",
            "server": "15.0.0.6"
        }
    ],
    "rbld_disks_results_list": [],
    "spare_disks_results_list": []
}
```

Perform a replace disk operation for a particular server

Resource URI

Verb	URI
GET	v1/diskmgmt/server/?server_list={server_list}&action={replace_disks}

Example

JSON Request

```
GET
/v1/diskmgmt/server?server_list=i13-21&action=replace_disks
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "add_as_spares_disks_results_list": [
        {
            "disk slot": "1",
            "host": "i13-21",
            "replace status": "Success",
            "role": "control",
            "server": "15.0.0.8"
        }
    ]
}
```

OSD Maintenance information

REST wrapper to query information about OSD on Pod storage nodes. This will return the OSD status information of all or a selection of OSDs available in the Pod.

End Point	Type	Valid Args	Valid Arg Values	Example
/osdmgmt	GET	None	None	/v1/osdmgmt/
/osdmgmt/check_osds	GET	None	Detail	/v1/osdmgmt/check_osds/
osdmgmt/server	GET	server_list, action	List of valid server names, check_osds, replace_osd, osd_name	/v1/osdmgmt/server/?server_list =svr1,svr2&action=check_osds /v1/osdmgmt/server/server_list=svr1&action =replace_osd&osd_name=osd_name

Get a OSD disk operation

Resource URI

Verb	URI
GET	/v1/osdmgmt

Example

JSON Request

```
GET
/v1/osdmgmt
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json
{
    "bad_osds_results_list": [],
    "osd_details_results_list": [
        {
            "All OSD status": "All Good",
            "Num OSDs": 5,
            "OSD_detail": [
                {
                    "OSD_id": 0,
                    "OSD_journal": "/dev/sda4",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-0",
                    "OSD_name": "osd.0",
                    "OSD_path": "/dev/sdb1",
                    "OSD_status": "up",
                    "slot_id": 2
                },
                {
                    "OSD_id": 3,
                    "OSD_journal": "/dev/sda5",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-3",
                    "OSD_name": "osd.3",
                    "OSD_path": "/dev/sdc1",
                    "OSD_status": "up",
                    "slot_id": 3
                },
                {
                    "OSD_id": 6,
                    "OSD_journal": "/dev/sda6",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-6",
                    "OSD_name": "osd.6",
                    "OSD_path": "/dev/sdd1",
                    "OSD_status": "up",
                    "slot_id": 4
                },
                {
                    "OSD_id": 9,
                    "OSD_journal": "/dev/sda7",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-9",
                    "OSD_name": "osd.9",
                    "OSD_path": "/dev/sde1",
                    "OSD_status": "up",
                    "slot_id": 5
                },
                {
                    "OSD_id": 12,
                    "OSD_journal": "/dev/sda8",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-12",
                    "OSD_name": "osd.12",
                    "OSD_path": "/dev/sdf1",
                    "OSD_status": "up",
                    "slot_id": 6
                }
            ],
            "host": "i13-27-test",
            "role": "block_storage",
            "server": "15.0.0.4"
        },
        {
            "All OSD status": "All Good",
            "Num OSDs": 5,
            "OSD_detail": [
                {
                    "OSD_id": 1,
                    "OSD_journal": "/dev/sda4",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-1",
                    "OSD_name": "osd.1",
                    "OSD_path": "/dev/sdb1",
                    "OSD_status": "up",
                    "slot_id": 2
                },
                {
                    "OSD_id": 4,
                    "OSD_journal": "/dev/sda5",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-4",
                    "OSD_name": "osd.4",
                    "OSD_path": "/dev/sdc1",
                    "OSD_status": "up",
                    "slot_id": 3
                },
                {
                    "OSD_id": 7,
                    "OSD_journal": "/dev/sda6",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-7",
                    "OSD_name": "osd.7",
                    "OSD_path": "/dev/sdd1",
                    "OSD_status": "up",
                    "slot_id": 4
                },
                {
                    "OSD_id": 10,
                    "OSD_journal": "/dev/sda7",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-10",
                    "OSD_name": "osd.10",
                    "OSD_path": "/dev/sde1",
                    "OSD_status": "up",
                    "slot_id": 5
                },
                {
                    "OSD_id": 13,
                    "OSD_journal": "/dev/sda8",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-13",
                    "OSD_name": "osd.13",
                    "OSD_path": "/dev/sdf1",
                    "OSD_status": "up",
                    "slot_id": 6
                }
            ],
            "host": "i13-25",
            "role": "block_storage",
            "server": "15.0.0.11"
        },
        {
            "All OSD status": "All Good",
            "Num OSDs": 5,
            "OSD_detail": [
                {
                    "OSD_id": 2,
                    "OSD_journal": "/dev/sda4",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-2",
                    "OSD_name": "osd.2",
                    "OSD_path": "/dev/sdb1",
                    "OSD_status": "up",
                    "slot_id": 2
                },
                {
                    "OSD_id": 5,
                    "OSD_journal": "/dev/sda5",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-5",
                    "OSD_name": "osd.5",
                    "OSD_path": "/dev/sdc1",
                    "OSD_status": "up",
                    "slot_id": 3
                },
                {
                    "OSD_id": 8,
                    "OSD_journal": "/dev/sda6",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-8",
                    "OSD_name": "osd.8",
                    "OSD_path": "/dev/sdd1",
                    "OSD_status": "up",
                    "slot_id": 4
                },
                {
                    "OSD_id": 11,
                    "OSD_journal": "/dev/sda7",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-11",
                    "OSD_name": "osd.11",
                    "OSD_path": "/dev/sde1",
                    "OSD_status": "up",
                    "slot_id": 5
                },
                {
                    "OSD_id": 14,
                    "OSD_journal": "/dev/sda8",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-14",
                    "OSD_name": "osd.14",
                    "OSD_path": "/dev/sdf1",
                    "OSD_status": "up",
                    "slot_id": 6
                }
            ],
            "host": "i13-26",
            "role": "block_storage",
            "server": "15.0.0.9"
        }
    ]
}
```

Perform a check OSD operation for a particular server

Resource URI

Verb	URI
GET	/v1/osdmgmt/server/?server_list={server_list}&action={check_osds}

Example

JSON Request

```
GET
/v1/diskmgmt/server/?server_list=i13-26&action=check_osds
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json

{
    "bad_osds_results_list": [],
    "osd_details_results_list": [
        {
            "All OSD status": "All Good",
            "Num OSDs": 5,
            "OSD_detail": [
                {
                    "OSD_id": 2,
                    "OSD_journal": "/dev/sda4",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-2",
                    "OSD_name": "osd.2",
                    "OSD_path": "/dev/sdb1",
                    "OSD_status": "up",
                    "slot_id": 2
                },
                {
                    "OSD_id": 5,
                    "OSD_journal": "/dev/sda5",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-5",
                    "OSD_name": "osd.5",
                    "OSD_path": "/dev/sdc1",
                    "OSD_status": "up",
                    "slot_id": 3
                },
                {
                    "OSD_id": 8,
                    "OSD_journal": "/dev/sda6",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-8",
                    "OSD_name": "osd.8",
                    "OSD_path": "/dev/sdd1",
                    "OSD_status": "up",
                    "slot_id": 4
                },
                {
                    "OSD_id": 11,
                    "OSD_journal": "/dev/sda7",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-11",
                    "OSD_name": "osd.11",
                    "OSD_path": "/dev/sde1",
                    "OSD_status": "up",
                    "slot_id": 5
                },
                {
                    "OSD_id": 14,
                    "OSD_journal": "/dev/sda8",
                    "OSD_mount": "/var/lib/ceph/osd/ceph-14",
                    "OSD_name": "osd.14",
                    "OSD_path": "/dev/sdf1",
                    "OSD_status": "up",
                    "slot_id": 6
                }
            ],
            "host": "i13-26",
            "role": "block_storage",
            "server": "15.0.0.9"
        }
    ]
}
```

Perform a replace OSD operation for a particular server

Resource URI

Verb	URI
GET	/v1/osdmgmt/server/?server_list={server_name}&action=replace_osd&osd_name={osd_name}

Example

JSON Request

```
GET
/v1/diskmgmt/server/?server_list=i13-25&action=replace_osd&osd_name=osd.10
Accept: application/json
```

JSON Response

```
200 OK
Content-Type: application/json

{
'osd_replace_details_results_list': [
        {   'hdd_slot': 5,
                          'host_name': 'i13-25',
                          'journal_mnt': '/dev/sda4',
                          'new_dev_uuid': 'UUID=94480b3e-5698-4d6a-b715-0613be41cff5',
                          'new_mount': '/var/lib/ceph/osd/ceph-10',
                          'new_osd_id': 10,
                          'new_path': '/dev/sde1',
                          'old_osd_id': 10,
                          'status_msg': 'Successfully deleted, removed and replaced OSD
osd.10 from server i13-25'
                    }
              ]
}
```

CHAPTER 3

Monitoring Cisco NFVI Performance

The following topics tell you how to display logs to monitor Cisco VIM performance.

Logging and Monitoring in Cisco NFVI

Cisco VIM uses a combination of open source tools to collect and monitor the Cisco OpenStack services including Elasticsearch, Fluentd, and the Kibana dashboard (EFK).

In VIM, we have moved our platform to use Fluentd, instead of logstash. However, to maintain backwards compatibility, the code, and documentation refers to ELK, instead of EFK at various places. In VIM, these two acronyms are interchangeable, however it refers to the presence of EFK in the offering. OpenStack services that followed by EFK include:

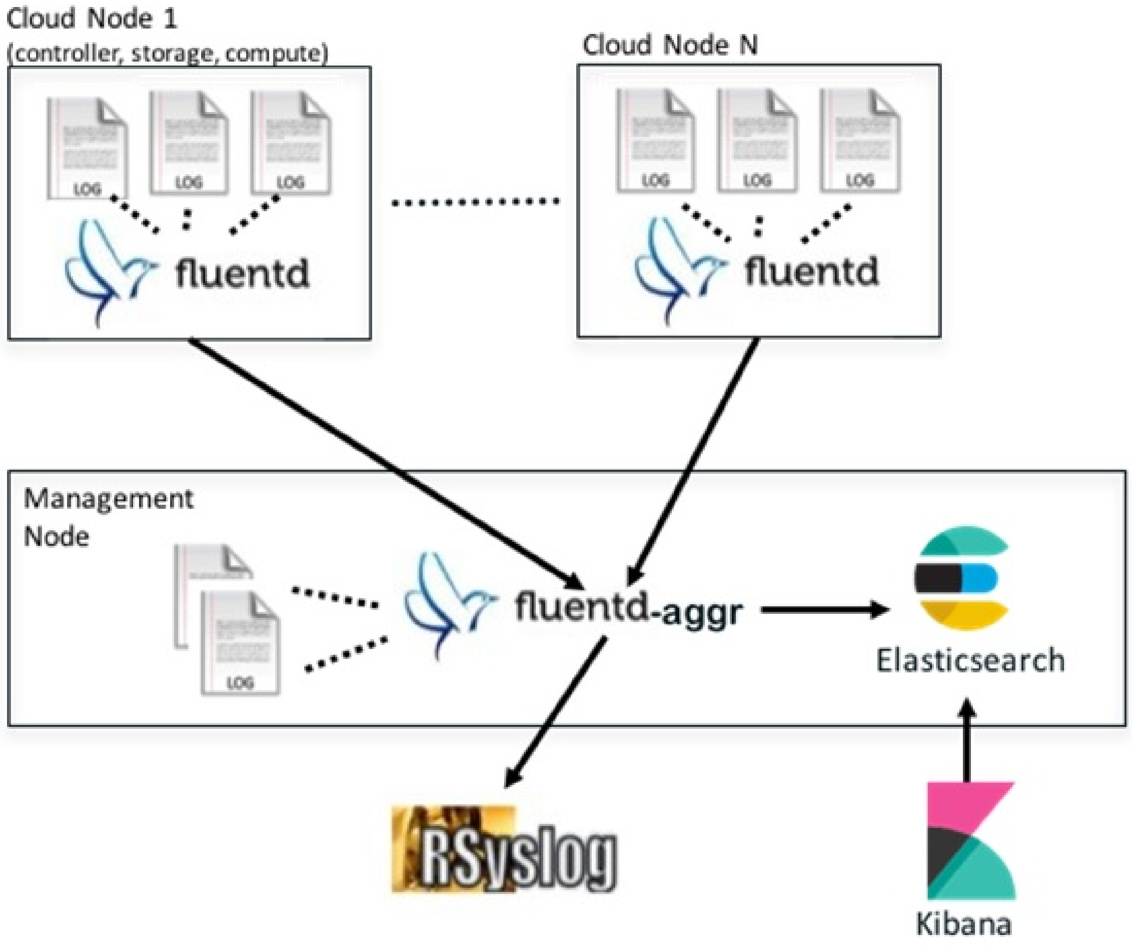
- MariaDB—A relational database management system which is based on MySQL. All the OpenStack components store their data in MariaDB.
- HAProxy—HAProxy is a free open source software that provides a high-availability load balancer, and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers.
- Keystone—Keystone is an OpenStack project that provides identity, token, catalog, and policy services for use specifically by projects in the OpenStack.
- Glance—An OpenStack project that allows you to upload and discover data assets that are meant for use with other services.
- Neutron—An OpenStack project that provides the network connectivity between interface devices, such as vNICs, managed by other OpenStack services, such as Nova.
- Nova—An OpenStack project that is designed to provide massively scalable, on demand, self-service access to compute resources.
- HTTP—The Apache HTTP server Project, an effort to develop and maintain an open-source HTTP server.

- Cinder—An OpenStack block storage service that is designed to present storage resources to the users that are consumed by the OpenStack compute project (Nova).
- Memcached—A general purpose distributed memory caching system.
- CloudPulse—Is an OpenStack tool that checks the health of the cloud. CloudPulse includes operator and end-point tests.
- Heat—The main OpenStack Orchestration program. Heat implements an orchestration engine to launch multiple composite cloud applications that is based on text file templates.
- Other OpenStack services—RabbitMQ, Ceph, Open vSwitch, Linux bridge, Neutron VTS (optional), and others.
- VMTP—Integrated control and data plane log for testing the cloud.
- NFVBench—Network performance benchmarking tool.

A Fluentd container resides on each control, compute, and storage nodes. They forward log to the Fluentd-aggr server residing on the management node.

The following figure shows a high-level schematic of the Fluent service assurance architecture.

Figure 8: EFK Service Assurance Architecture



The EFK flow includes:

- Fluentd extracts the relevant data from the logs and tags them so that Kibana can use it later to display useful information about those logs.
- Fluentd sends the logs from all the compute, controller, and storage nodes to the Fluentd-aggr server on the management node.
- Fluentd-aggr in the management node sends the structured logs into the Elasticsearch database.
- Elasticsearch stores the data, indexes it, and supports fast queries against a large amount of log data.
- Kibana visualizes the data that is stored in Elasticsearch using a custom dashboard. You can also add filters to the data to visualize interesting fragments of the log data.

Displaying Cisco VIM Log Files Using the CLI

Cisco VIM log file location depends on the node and log type. Installer logs are found in the management node under the /var/log/mercury/<install_uuid>/ directory. The last 20 log directories are tarred and kept in this directory. These files contain logs related to bootstrap, build orchestration, baremetal, common setup, and OpenStack orchestration.

If the installer fails, look at the last tar.gz file for logs, for example:

```
[root@mgmtnode mercury]# ls -lrt
total 20
drwxr-xr-x. 2 root root   80 Jul 19 23:42 573f2b7f-4463-4bfa-b57f-98a4a769aced
drwxr-xr-x. 2 root root 4096 Jul 20 03:29 installer
drwxr-xr-x. 2 root root   79 Jul 20 03:29 e9117bc5-544c-4bda-98d5-65bffa56a18f
drwxr-xr-x. 2 root root   79 Jul 20 04:54 36cdf8b5-7a35-4e7e-bb79-0cfb1987f550
drwxr-xr-x. 2 root root   79 Jul 20 04:55 bd739014-fdf1-494e-adc0-98b1fba510bc
drwxr-xr-x. 2 root root   79 Jul 20 04:55 e91c4a6c-ae92-4fef-8f7c-cafa9f5dc1a3
drwxr-xr-x. 2 root root   79 Jul 20 04:58 1962b2ba-ff15-47a6-b292-25b7fb84cd28
drwxr-xr-x. 2 root root   79 Jul 20 04:59 d881d453-f6a0-448e-8873-a7c51d8cc442
drwxr-xr-x. 2 root root   78 Jul 20 05:04 187a15b6-d425-46a8-a4a2-e78b65e008b6
drwxr-xr-x. 2 root root 4096 Jul 20 06:47 d0346cdd-5af6-4058-be86-1330f7ae09d1
drwxr-xr-x. 2 root root   79 Jul 20 17:09 f85c8c6c-32c9-44a8-b649-b63fdb11a79a
drwxr-xr-x. 2 root root   67 Jul 20 18:09 179ed182-17e4-4f1f-a44d-a3b6c16cf323
drwxr-xr-x. 2 root root   68 Jul 20 18:13 426cb05f-b1ee-43ce-862d-5bb4049cc957
drwxr-xr-x. 2 root root   68 Jul 20 18:13 1d2eec9d-f4d8-4325-9eb1-7d96d23e30fc
drwxr-xr-x. 2 root root   68 Jul 20 18:13 02f62a2f-3f59-46a7-9f5f-1656b8721512
drwxr-xr-x. 2 root root   68 Jul 20 18:14 c7417be9-473e-49da-b6d0-d1ab8fb4b1fc
drwxr-xr-x. 2 root root   68 Jul 20 18:17 b4d2077b-c7a9-46e7-9d39-d1281fba9baf
drwxr-xr-x. 2 root root   68 Jul 20 18:35 21972890-3d45-4642-b41d-c5fadfeba21a
drwxr-xr-x. 2 root root   80 Jul 20 19:17 d8b1b54c-7fc1-4ea6-83a5-0e56ff3b67a8
drwxr-xr-x. 2 root root   80 Jul 20 19:17 23a3cc35-4392-40bf-91e6-65c62d973753
drwxr-xr-x. 2 root root   80 Jul 20 19:17 7e831ef9-c932-4b89-8c81-33a45ad82b89
drwxr-xr-x. 2 root root   80 Jul 20 19:18 49ea0917-f9f4-4f5d-82d9-b86570a02dad
drwxr-xr-x. 2 root root   80 Jul 20 19:18 21589a61-5893-4e30-a70e-55ad0dc2e93f
drwxr-xr-x. 2 root root   80 Jul 20 19:22 6ae6d136-7f87-4fc8-92b8-64cd542495bf
drwxr-xr-x. 2 root root 4096 Jul 20 19:46 1c6f4547-c57d-4dcc-a405-ec509306ee25
drwxr-xr-x. 2 root root   68 Jul 20 21:20 c6dcc98d-b45b-4904-a217-d25001275c85
drwxr-xr-x. 2 root root   68 Jul 20 21:40 ee58d5d6-8b61-4431-9f7f-8cab2c331637
drwxr-xr-x. 2 root root 4096 Jul 20 22:06 243cb0f8-5169-430d-a5d8-48008a00d5c7
drwxr-xr-x. 2 root root 4096 Jul 20 22:16 188d53da-f129-46d9-87b7-c876b1aea70c
```

Cisco VIM autobackup logs are found in the following location:

```
# CVIM autobackup logs (auto-backup enabled by default)
/var/log/mercury/autobackup_2.2.x_2018-03-19_15-11-10.log

# cobbler apache log (may be needed for PXE troubleshooting)
/var/log/cobblerhttpd/access_log
/var/log/cobblerhttpd/error_log

# VMTP logs
/var/log/vmtp/vmtp.log
```

Cisco VIM RestAPI log location

```
# CVIM RestAPI logs
/var/log/mercury_restapi/restapi.log

# CIM RestAPI apache logs (TCP port 8445)
/var/log/httpd/mercury_access.log
/var/log/httpd/mercury_error.log

# CIM RestAPI log-directory logs (TCP port 8008)
/var/log/httpd/access_log
/var/log/httpd/error_log
```

EFK log location

```
# Elasticsearch-fluentd-Kibana
/var/log/elasticsearch/
/var/log/fluentd-aggr/
/var/log/kibana/
/var/log/curator/
```

```
# HAProxy TLS certificate expiration check
/var/log/curator/certchecker.log
```

Viewing Cisco VIM Logs

```
# list logs sorted reverse on time
ls -lrt /var/log/mercury/
# untar logs
tar xvzf /var/log/mercury/<UUID>/mercury_install_2018-3-20_10-2.tar.gz -C /tmp/
```

Cisco VIM Configuration Files

```
# example configuration files
/root/openstack-configs/setup_data.yaml.B_Series_EXAMPLE
/root/openstack-configs/setup_data.yaml.C_Series_EXAMPLE

# system maintained setup files - do not modify directly
# always supply user copy of setup_data.yaml
# when using ciscovim client
/root/openstack-configs/setup_data.yaml

# system inventory in pretty format
/root/openstack-configs/mercury_servers_info

# passwords store
/root/openstack-configs/secrets.yaml

# openstack configuration file
/root/openstack-configs/openstack_config.yaml

# RestAPI password
/opt/cisco/ui_config.json

# Insight password
/opt/cisco/insight/secrets.yaml
```

Enabling debug logs for certain OpenStack Services

```
# openstack config file
/root/openstack-configs/openstack_config.yaml

# help
ciscovim help

# list openstack keys
ciscovim list-openstack-configs

# help on reconfigure sub-command
ciscovim help reconfigure

# how to execute subcommand, example below
# important note: reconfigure requires a maintenance window
ciscovim reconfigure --setopenstackconfig KEYSTONE_DEBUG_LOGGING,CINDER_DEBUG_LOGGING
```

On controller and compute nodes, all services are run within their respective Docker™ containers.

To list the Docker containers in the node, execute the following:

```
[root@control-server-2 ~]# docker ps -a
CONTAINER ID        IMAGE                                                          COMMAND
                  CREATED          STATUS          PORTS      NAMES
258b2ca1d46a        172.31.228.164:5000/mercury-rhel7-osp8/nova-scheduler:4780
"/usr/bin/my_init /no"   25 minutes ago    Up 25 minutes             novascheduler_4780
ffe70809bbe0        172.31.228.164:5000/mercury-rhel7-osp8/nova-novncproxy:4780
"/usr/bin/my_init /st"   25 minutes ago    Up 25 minutes             novanovncproxy_4780
```

```
12b92bcb9dc0        172.31.228.164:5000/mercury-rhel7-osp8/nova-consoleauth:4780
"/usr/bin/my_init /st"    26 minutes ago   Up 26 minutes


……
novaconsoleauth_4780
7295596f5167        172.31.228.164:5000/mercury-rhel7-osp8/nova-api:4780
"/usr/bin/my_init /no"    27 minutes ago   Up 27 minutes              novaapi_4780
```

To view the Docker logs of any container, execute the following on the corresponding host:

```
ls –l /var/log/<service_name>/<log_filename>
e.g. ls -l /var/log/keystone/keystone.log
```

To get into a specific container, execute the following commands:

```
[root@control-server-2 ~]# alias | grep container
     root@control-server-2 ~]# source /root/.bashrc
#execute the alias:
  [root@control-server-2 ~]# novaapi
novaapi_4761 [nova@control-server-2 /]$
 novaapi_4761 [nova@control-server-2 /]$ exit
exit
```

If the Docker status indicates a container is down (based on output of "docker ps –a"), collect the Docker service logs as well:

```
cd /etc/systemd/system/multi-user.target.wants/
ls docker* # get the corresponding service name from the output
systemctl status <service_name> -n 1000 > /root/filename # redirects the output to the file
```

For storage nodes running Ceph, execute the following to check the cluster status:

```
ceph –v # on monitor nodes (controller), show's ceph version

ceph –s # on monitor nodes (controller), show cluster status

ceph osd lspools #on monitor nodes (controller),list pools

ceph mon stat # summarize monitor status

ceph-disk list # on OSD / storage nodes; List disks, partitions, and Ceph OSDs

rbd list images # on monitor nodes (controller); dump list of image snapshots

rbd list volumes # on monitor nodes (controller); dump list of volumes
```

Logging Into the Kibana Dashboard

Kibana is an open source data visualization platform that you can use to explore Cisco VIM logs.

To log into the Kibana dashboard:

Step 1 Using a terminal client, use SSH to log into your management node and enter the password to login.

The following command shows the management node has an IP address of 17.0.0.2:

```
# ssh root@17.0.0.2
root@17.0.0.2's password
```

Step 2 In the SSH terminal session, locate the line containing ELK_PASSWORD in /root/installer-{tag id}/openstack-configs/secrets.yaml. Note the value of the ELK_PASSWORD. It will be used in Step 4.

```
cat /root/installer-{tag-id}/openstack-configs/secrets.yaml
...
ELK_PASSWORD: <note this value>
...
```

Step 3 Using your web browser, navigate to http://<management_node_ip_address>:5601.

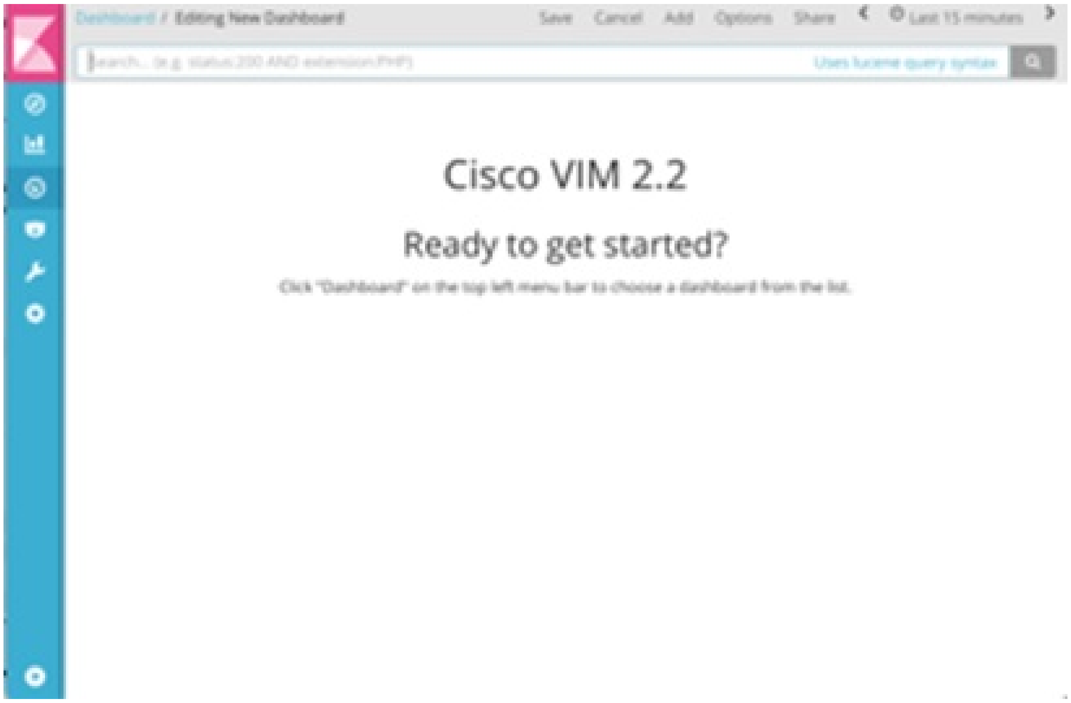
Step 4 When prompted, log in with the following credentials:

User Name: admin

Password: <value of ELK_PASSWORD from Step 2>

The Kibana dashboard appears allowing you to display the Cisco VIM service and installer logs.

Figure 9: Editing New Dashboard


Dashboard / Editing New Dashboard
Save
Cancel
Add
Options
Share
Last 15 minutes
Search... (e.g. status:200 AND extension:PHP)
Uses lucene query syntax

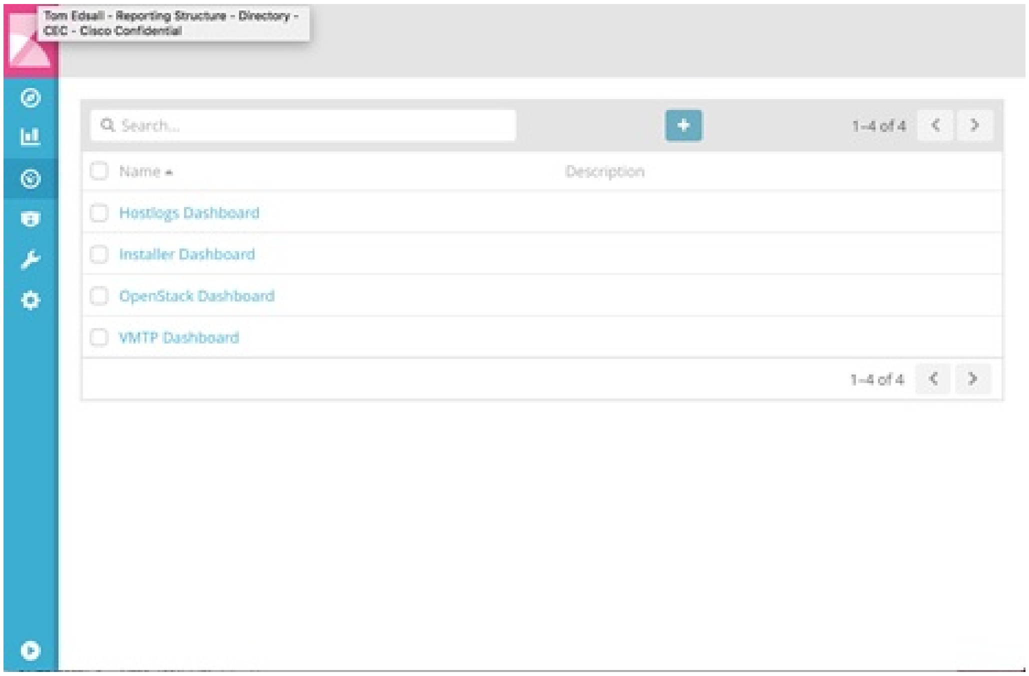
Cisco VIM 2.2

Ready to get started?

Click "Dashboard" on the top left menu bar to choose a dashboard from the list.

Step 5 Navigate to the dashboards by clicking **Dashboard** menu bar and choose the desired dashboard. It is not recommended to use visualize/Timelion/DevTools or Management options on the left side.

Figure 10: Lists of Dashboards


Tom Edsall - Reporting Structure - Directory - CEC - Cisco Confidential
Search...
1-4 of 4
Name
Description
1-4 of 4

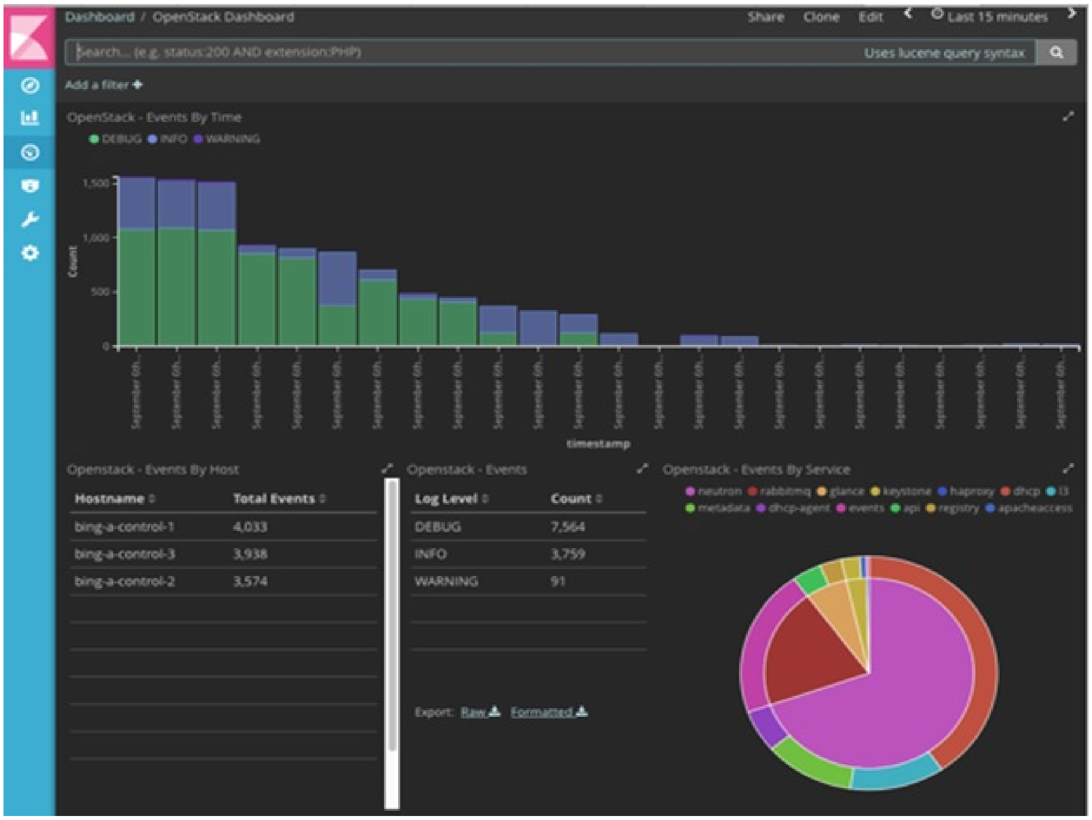
□ Hostlogs Dashboard
□ Installer Dashboard
□ OpenStack Dashboard
□ VMTP Dashboard

The following are the list of dashboards supported:

- Hostlogs Dashboard: Provides log information of the system for the cloud nodes. This displays entries from the host logs-* index in Elasticsearch. It contains the log from /var/log/messages file on each server.
- Installer Dashboard: Provides information about the management node and the installation process. It can only read uncompressed files. Hence, it reads the files prior to the cloud installation. This displays entries from the installer-* index in Elasticsearch.
- OpenStack Dashboard: (openstack-* index) Provides log information about all the OpenStack processes. This displays entries from the openstack-* index in Elasticsearch.
- VMTP Dashboard: Provides log information about the VMTP runs performed against the cloud. It displays entries from the vmtp-* index in Elasticsearch

For Example: if you click on the OpenStack dashboard link the following screen appears.

Figure 11: OpenStack Dashboard



Hostname	Total Events
bing-a-control-1	4,033
bing-a-control-3	3,938
bing-a-control-2	3,574

Log Level	Count
DEBUG	7,564
INFO	3,759
WARNING	91

You can switch on from one dashboard to another by selecting the appropriate dashboard from the right top bar menu.

All dashboards have generic and specific fields.

The generic ones are:

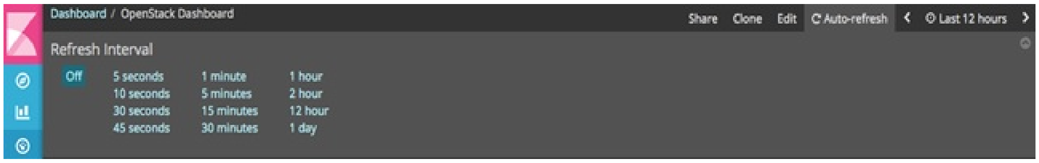
- Title: Title is seen at the top left of the page. Title shows which dashboard is being displayed. For Example: OpenStack Dashboard.
- Time: Time is seen at the top right of the page. Time indicates the time schedule for the log information. You can modify the time to indicate absolute, relative time in the past or specify automatically refresh rates.
- Search bar: Search bar is an input field where you can enter a query in the Lucene syntax format to filter the logs by specific fields (which depend on the fields for the index being selected)
- Add a filter tab: Use this tab to introduce filters graphically.

For more information on using Kibana, see the *Kibana documentation* (Version 5.5.1).

Cisco VIM stores the OpenStack logs in Elasticsearch. The Elasticsearch snapshots all the indices (where the data is stored) which are rotated on a periodic basis. You may not see the older data in Kibana if the data is rotated out and/or deleted.

Logs keep being visualized in Kibana as they are being updated in Elasticsearch on the Discover tab. To debug something on kibana, you can program the Kibana dashboard to auto-refresh at specific intervals (by default is off). To enable auto-refresh, click the date at the top right corner of the dashboard and click Auto-refresh to configure the desired value.

Figure 12: Auto-Refresh



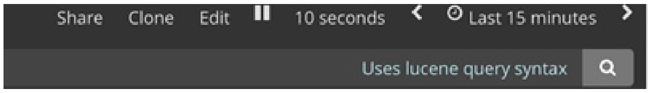
Dashboard / OpenStack Dashboard

Refresh Interval

Off

Off	5 seconds	1 minute	1 hour
	10 seconds	5 minutes	2 hour
	30 seconds	15 minutes	12 hour
	45 seconds	30 minutes	1 day

User can click the play/pause button on the top navigator bar to continue/pause the refreshing of logs events:

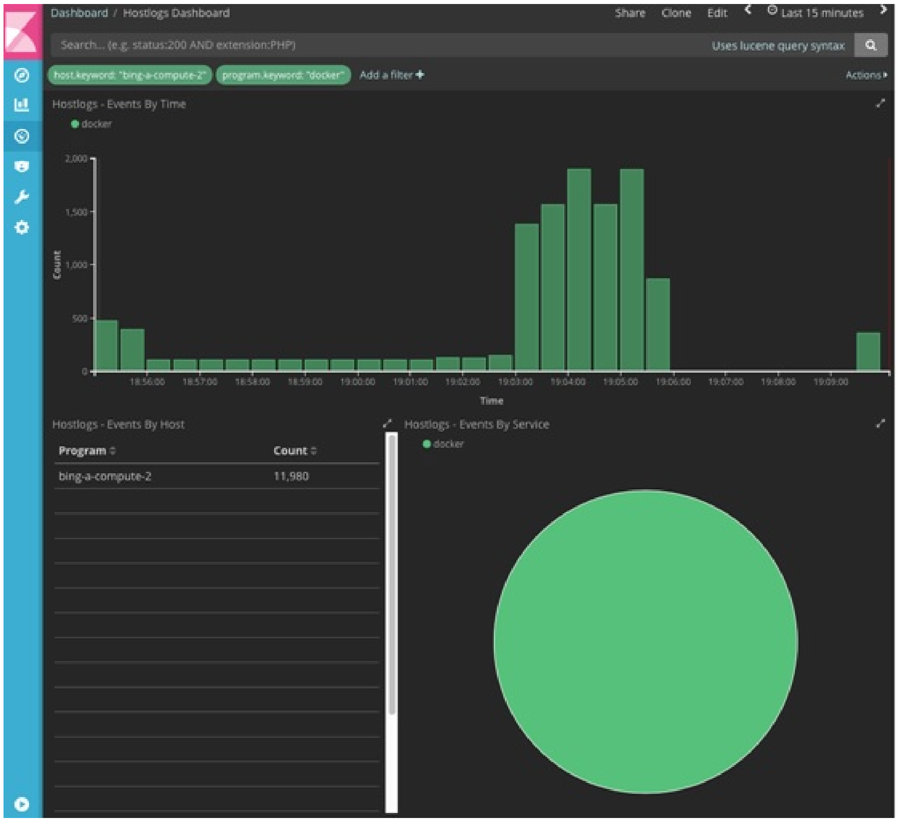


a) Few examples on usage of filters in Openstack dashboard to gather useful information

- • On the Hostlogs Dashboard, in the Events by Host panel, choose a hostname and click the + or - symbol that appears close to the hostname to include or exclude that server from the filter. Then, click the desired slice on the Events By Service panel to add the docker service to the section.

  • Under the Search field, you will see the included sections in green and excluded sections in red.

Figure 13: Hostlogs Dashboard



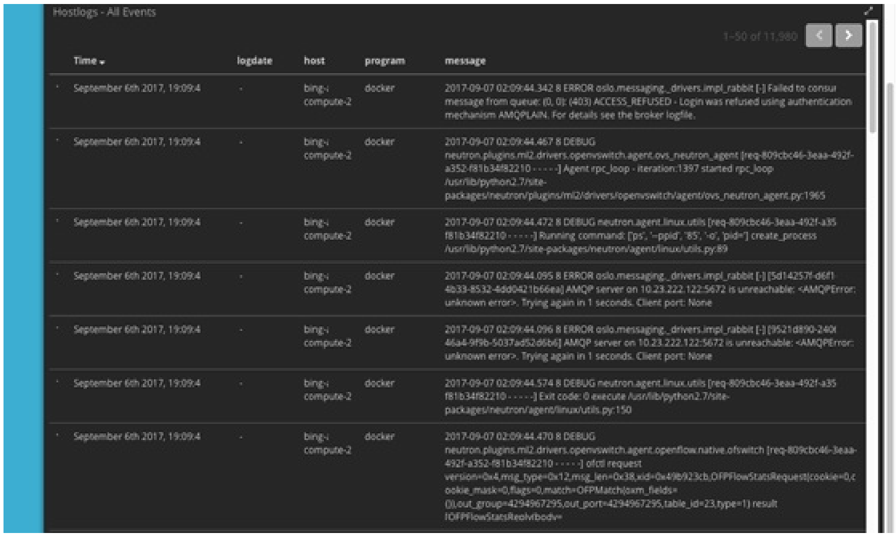
Dashboard / Hostlogs Dashboard

Search... (e.g. status:200 AND extension:PHP)

host.keyword: "bing-a-compute-2"   program.keyword: "docker"

Hostlogs - Events By Host

Program	Count
bing-a-compute-2	11,980



Hostlogs - All Events

1–50 of 11,980

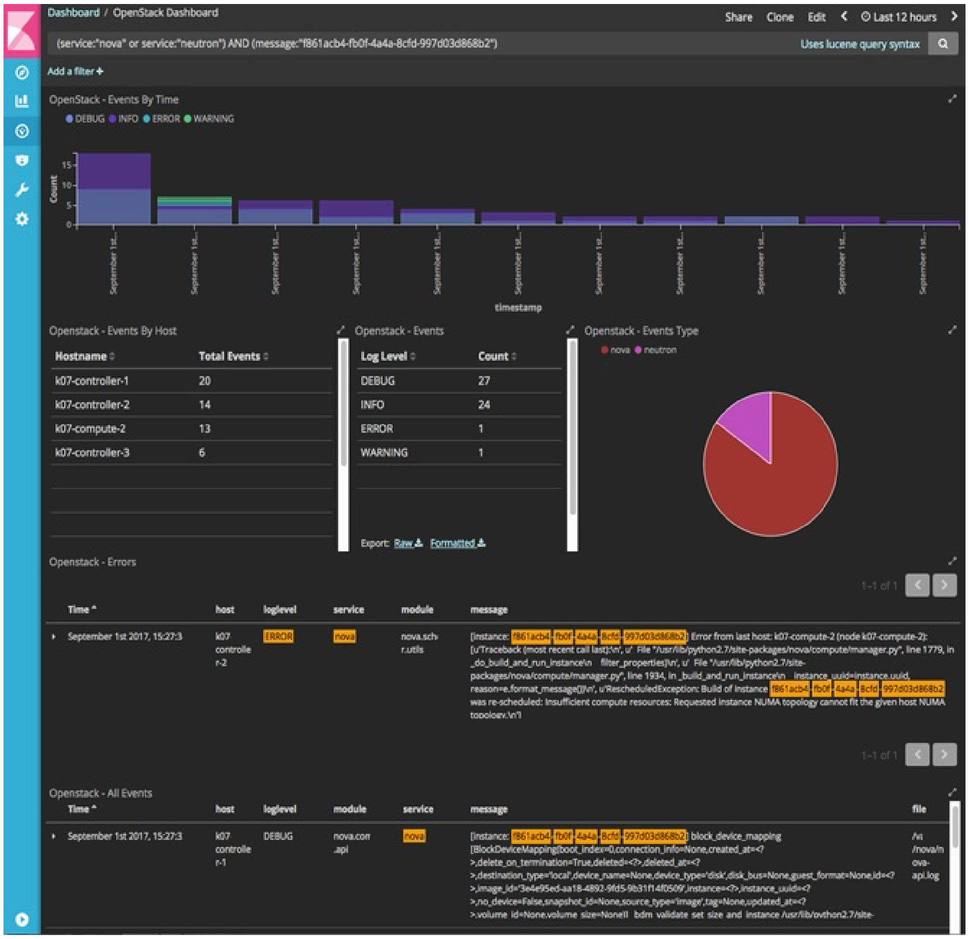
Time	logdate	host	program	message
September 6th 2017, 19:09:4	-	bing-i compute-2	docker	2017-09-07 02:09:44.342 8 ERROR oslo.messaging._drivers.impl_rabbit [-] Failed to consu message from queue: (0, 0): (403) ACCESS_REFUSED - Login was refused using authentication mechanism AMQPLAIN. For details see the broker logfile.
September 6th 2017, 19:09:4	-	bing-i compute-2	docker	2017-09-07 02:09:44.467 8 DEBUG neutron.plugins.ml2.drivers.openvswitch.agent.ovs_neutron_agent [req-809cbc46-3eaa-492f-a352-f81b34f82210 - - - - -] Agent rpc_loop - iteration:1397 started rpc_loop /usr/lib/python2.7/site-packages/neutron/plugins/ml2/drivers/openvswitch/agent/ovs_neutron_agent.py:1965
September 6th 2017, 19:09:4	-	bing-i compute-2	docker	2017-09-07 02:09:44.472 8 DEBUG neutron.agent.linux.utils [req-809cbc46-3eaa-492f-a35 f81b34f82210 - - - - -] Running command: ['ps', '--ppid', '85', '-o', 'pid='] create_process /usr/lib/python2.7/site-packages/neutron/agent/linux/utils.py:89
September 6th 2017, 19:09:4	-	bing-i compute-2	docker	2017-09-07 02:09:44.095 8 ERROR oslo.messaging._drivers.impl_rabbit [-] [5d14257f-d6f1 4b33-8532-4dd0421b66ea] AMQP server on 10.23.222.122:5672 is unreachable: <AMQPError: unknown error>. Trying again in 1 seconds. Client port: None
September 6th 2017, 19:09:4	-	bing-i compute-2	docker	2017-09-07 02:09:44.096 8 ERROR oslo.messaging._drivers.impl_rabbit [-] [9521d890-240 46a4-9f9b-5037ad52d6b6] AMQP server on 10.23.222.122:5672 is unreachable: <AMQPError: unknown error>. Trying again in 1 seconds. Client port: None
September 6th 2017, 19:09:4	-	bing-i compute-2	docker	2017-09-07 02:09:44.574 8 DEBUG neutron.agent.linux.utils [req-809cbc46-3eaa-492f-a35 f81b34f82210 - - - - -] Exit code: 0 execute /usr/lib/python2.7/site-packages/neutron/agent/linux/utils.py:150
September 6th 2017, 19:09:4	-	bing-i compute-2	docker	2017-09-07 02:09:44.470 8 DEBUG neutron.plugins.ml2.drivers.openvswitch.agent.openflow.native.ofswitch [req-809cbc46-3eaa-492f-a352-f81b34f82210 - - - - -] ofctl request version=0x4,msg_type=0x12,msg_len=0x38,xid=0x49b923cb,OFPFlowStatsRequest(cookie=0,cookie_mask=0,flags=0,match=OFPMatch(oxm_fields={}),out_group=4294967295,out_port=4294967295,table_id=23,type=1) result [OFPFlowStatsReply(body=

b) To know the log events in the Openstack for a given VM by writing the filter directly on the Search field:

Note The uuid of the VM can be identified by executing openstack nova list or looking at the horizon website.

- In the Search field which is on top of the Dashboard, write the following Lucene query: (service: nova and service: neutron) AND (message:<uuid>) here, <uuid> is the number got from Horizon or nova list for the identifier of the instance VM.

Figure 14: Search Query Page



(service:"nova" or service:"neutron") AND (message:"f861acb4-fb0f-4a4a-8cfd-997d03d868b2")

Openstack - Events By Host

Hostname	Total Events
k07-controller-1	20
k07-controller-2	14
k07-compute-2	13
k07-controller-3	6

Openstack - Events

Log Level	Count
DEBUG	27
INFO	24
ERROR	1
WARNING	1

Openstack - Errors

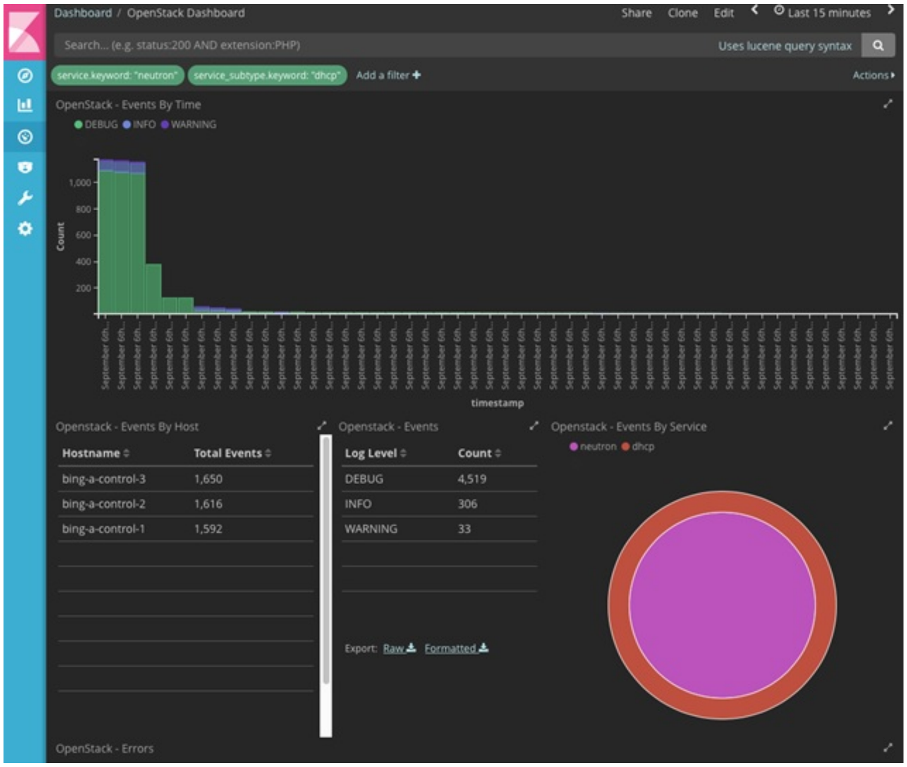
Time	host	loglevel	service	module	message
September 1st 2017, 15:27:3	k07 controlle r-2	ERROR	nova	nova.sch r.utils	[instance: f861acb4-fb0f-4a4a-8cfd-997d03d868b2] Error from last host: k07-compute-2 (node k07-compute-2): [u'Traceback (most recent call last):\n', u' File "/usr/lib/python2.7/site-packages/nova/compute/manager.py", line 1779, in _do_build_and_run_instance\n filter_properties)\n', u' File "/usr/lib/python2.7/site-packages/nova/compute/manager.py", line 1934, in _build_and_run_instance\n instance_uuid=instance.uuid, reason=e.format_message())\n', u'RescheduledException: Build of instance f861acb4-fb0f-4a4a-8cfd-997d03d868b2 was re-scheduled: Insufficient compute resources: Requested instance NUMA topology cannot fit the given host NUMA topology.\n']

Openstack - All Events

Time	host	loglevel	module	service	message	file
September 1st 2017, 15:27:3	k07 controlle r-1	DEBUG	nova.com .api	nova	[instance: f861acb4-fb0f-4a4a-8cfd-997d03d868b2] block_device_mapping [BlockDeviceMapping(boot_index=0,connection_info=None,created_at=<?>,delete_on_termination=True,deleted=<?>,deleted_at=<?>,destination_type='local',device_name=None,device_type='disk',disk_bus=None,guest_format=None,id=<?>,image_id='3e4e95ed-aa18-4892-9fd5-9b31f14f0509',instance=<?>,instance_uuid=<?>,no_device=False,snapshot_id=None,source_type='image',tag=None,updated_at=<?>,volume_id=None,volume_size=None)] _bdm_validate_set_size_and_instance /usr/lib/python2.7/site-	/vi /nova/n ova-api.log

- For example, if the user wants to know the DHCP events of the Openstack Neutron add filters by clicking outer circle of pie chart::
  - On the OpenStack Dashboard, the Openstack - Events By Service panel has a pie chart with the inner section for the services and the outer sections for the service_subtypes. To add filters for selecting all the events in a service (for example, neutron), click on the inner section of the pie. To add filters for selecting the service_subtypes (for example, dhcp), click on the outer circle of the pie.

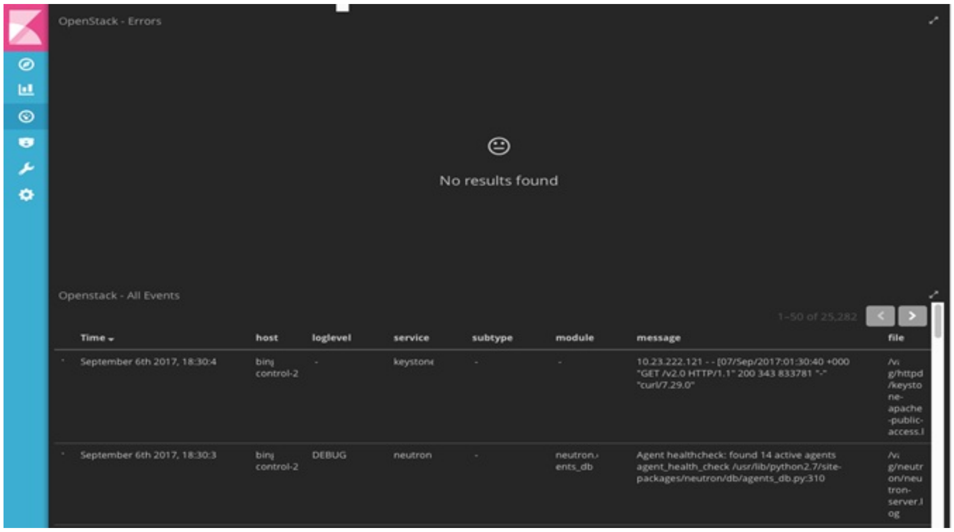
Figure 15: Events by Service



Hostname	Total Events
bing-a-control-3	1,650
bing-a-control-2	1,616
bing-a-control-1	1,592

Log Level	Count
DEBUG	4,519
INFO	306
WARNING	33

- Note: You can scroll down the OpenStack Dashboard to see the OpenStack - Errors and the OpenStack - Events panel.. The OpenStack - Errors panel displays the error messages. If there are no errors, the **No results found** message is displayed.



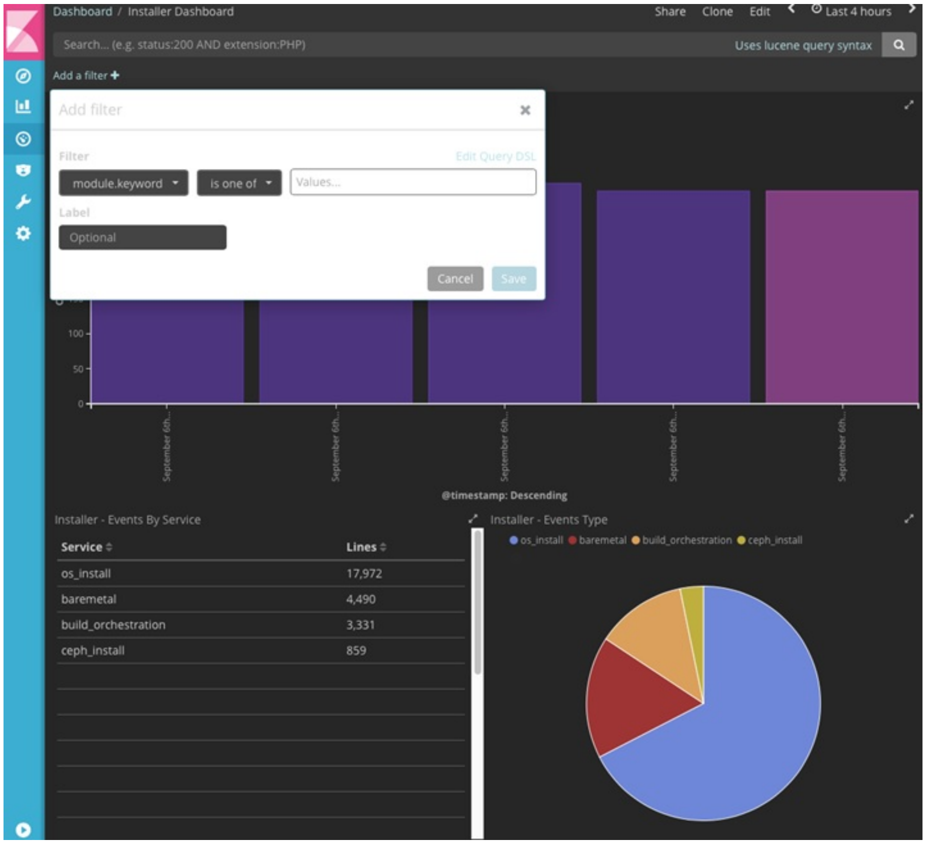
Time	host	loglevel	service	subtype	module	message	file
September 6th 2017, 18:30:4	binj control-2	-	keystone	-	-	10.23.222.121 - - [07/Sep/2017:01:30:40 +000 "GET /v2.0 HTTP/1.1" 200 343 833781 "-" "curl/7.29.0"	/v: g/httpd /keystone-apache -public-access.l
September 6th 2017, 18:30:3	binj control-2	DEBUG	neutron	-	neutron.ents_db	Agent healthcheck: found 14 active agents agent_health_check /usr/lib/python2.7/site-packages/neutron/db/agents_db.py:310	/v: g/neutron/neutron-server.log

- Without knowing the Lucene Syntax, you can set the filter criteria in the **Search** field using the **Add a filter** + option.

Following are the steps to add a filter:

- Click Add a filter (+).
- Set the filter criteria by choosing appropriate label and operators from the drop-down lists, and entering keywords and click Save.

Figure 16: Add Filters Page

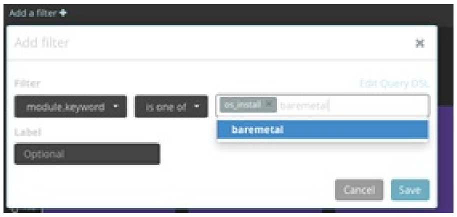


Installer - Events By Service

Service	Lines
os_install	17,972
baremetal	4,490
build_orchestration	3,331
ceph_install	859

Set the filter criteria by choosing appropriate label and operators from the drop-down lists, and entering keywords.

Figure 17: Choosing Appropriate Labels



Rotation of the Cisco VIM Logs

Cisco VIM stores all logs in Elasticsearch. Elasticsearch indices are rotated on a periodic basis to prevent the disk space overflow by creating snapshots. The following lists show the Snapshots that are defined in openstack_config.yaml:

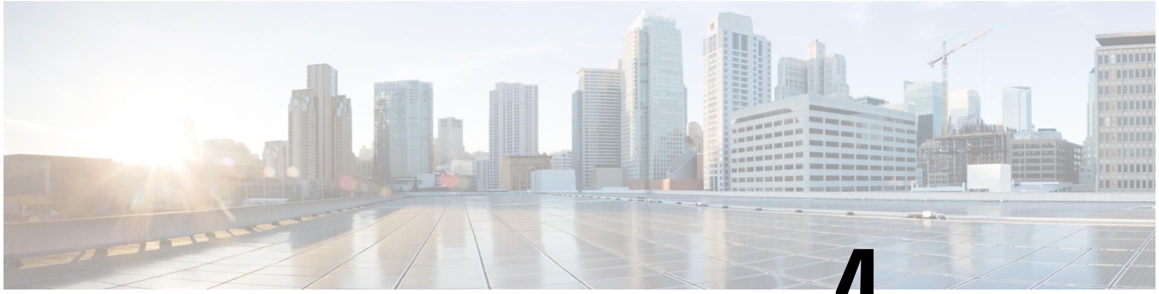
```
# vi ~/openstack-configs/openstack_config.yaml
…
# Elk rotation parameters
elk_rotation_frequency: "monthly"  # Available: "daily", "weekly", "fortnightly", "monthly"
elk_rotation_size: 2               # Unit is in Gigabytes (float is allowed)
elk_rotation_del_older: 10         # Delete older than 10 units (where units depends on the
 value set on elk_rotation_frequency)
…
```

You can change the frequency of the rotation by changing the values. For more information on how to set the Elasticsearch parameters through VIM API or CLI, refer to the section *Reconfiguring Passwords and OpenStack Configurations*.

Cisco VIM uses the open source Elasticsearch Curator tool to manage the Elasticsearch indices and snapshots. For more information about Elasticsearch handles snapshots, look at the official information on Elastic.co (Version 5.4) https://www.elastic.co/guide/en/elasticsearch/client/curator/5.4/index.html.

Network Performance Test with NFVBench

NFVBench is a network performance benchmarking tool integrated with Cisco VIM. For more details, refer to NFVBench section of *Chapter 1* in the admin guide for details.



CHAPTER 4

Managing Cisco NFVI Security

The following topics describe Cisco NFVI network and application security and best practices.

Verifying Management Node Network Permissions

The Cisco NFVI management node stores sensitive information related to Cisco NFVI operations. Access to the management node can be restricted to requests coming from IP addresses known to be used by administrators. The administrator source networks is configured in the setup file, under **[NETWORKING]** using the **admin_source_networks** parameter.

To verify this host based firewall setting, log into the management node as an admin user and list the rules currently enforces by iptables. Verify that the source networks match the values configured. If no source networks have been configured, then all source traffic is allowed. However, note that only traffic destined to ports with known admin services is allowed to pass. The **admin_source_networks** value can be set at install time or changed through a reconfigure.

```
[root@control-server-1 ~]# iptables –list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     tcp  --  10.0.0.0/8           anywhere             tcp dpt:ssh
ACCEPT     tcp  --  172.16.0.0/12        anywhere             tcp dpt:ssh
ACCEPT     tcp  --  10.0.0.0/8           anywhere             tcp dpt:https
ACCEPT     tcp  --  172.16.0.0/12        anywhere             tcp dpt:https
ACCEPT     tcp  --  10.0.0.0/8           anywhere             tcp dpt:4979
ACCEPT     tcp  --  172.16.0.0/12        anywhere             tcp dpt:4979
ACCEPT     tcp  --  10.0.0.0/8           anywhere             tcp dpt:esmagent
ACCEPT     tcp  --  172.16.0.0/12        anywhere             tcp dpt:esmagent
ACCEPT     tcp  --  10.0.0.0/8           anywhere             tcp dpt:8008
ACCEPT     tcp  --  172.16.0.0/12        anywhere             tcp dpt:8008
ACCEPT     tcp  --  10.0.0.0/8           anywhere             tcp dpt:copy
ACCEPT     tcp  --  172.16.0.0/12        anywhere             tcp dpt:copy
ACCEPT     tcp  --  10.0.0.0/8           anywhere             tcp dpt:22250
ACCEPT     tcp  --  172.16.0.0/12        anywhere             tcp dpt:22250
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
DROP       all  --  anywhere             anywhere
```

Verifying Management Node File Permissions

The Cisco NFVI management node stores sensitive information related to Cisco NFVI operations. These files are secured by strict file permissions. Sensitive files include secrets.yaml, openrc, *.key, and *.pem. To verify the file permissions, log into the management node as an admin user and list all of the files in the ***~/openstack-configs/*** directory. Verify that only the owner has read and write access to these files. For example:

```
[root@control-server-1 ~]# ls -l ~/openstack-configs
total 172
-rw-------. 1 root root  3272 Jun 21 17:57 haproxy.key
-rw-------. 1 root root  5167 Jun 21 17:57 haproxy.pem
-rw-------. 1 root root   223 Aug  8 18:09 openrc
-rw-------. 1 root root   942 Jul  6 19:44 secrets.yaml

[…]
```

Viewing Administrator Access Attempts

As the UCS servers are part of the critical Cisco NFVI infrastructure, Cisco recommends monitoring administrator login access periodically.

To view the access attempts, use the **journalctl** command to view the log created by ssh. For example:

```
[root@control-server-1 ~]# journalctl -u sshd
-- Logs begin at Tue 2016-06-21 17:39:35 UTC, end at Mon 2016-08-08 17:25:06 UTC. --
Jun 21 17:40:03 hh23-12 systemd[1]: Started OpenSSH server daemon.
Jun 21 17:40:03 hh23-12 systemd[1]: Starting OpenSSH server daemon...
Jun 21 17:40:03 hh23-12 sshd[2393]: Server listening on 0.0.0.0 port 22.
Jun 21 17:40:03 hh23-12 sshd[2393]: Server listening on :: port 22.
Jun 21 17:40:43 hh23-12 sshd[12657]: Connection closed by 171.70.163.201 [preauth]
Jun 21 17:41:13 hh23-12 sshd[12659]: Accepted password for root from 171.70.163.201 port
40499
Jun 21 17:46:41 hh23-12 systemd[1]: Stopping OpenSSH server daemon...
Jun 21 17:46:41 hh23-12 sshd[2393]: Received signal 15; terminating.
Jun 21 17:46:41 hh23-12 systemd[1]: Started OpenSSH server daemon.
Jun 21 17:46:41 hh23-12 systemd[1]: Starting OpenSSH server daemon...
Jun 21 17:46:41 hh23-12 sshd[13930]: Server listening on 0.0.0.0 port 22.
Jun 21 17:46:41 hh23-12 sshd[13930]: Server listening on :: port 22.
Jun 21 17:50:45 hh23-12 sshd[33964]: Accepted password for root from 171.70.163.201 port
40545
Jun 21 17:56:36 hh23-12 sshd[34028]: Connection closed by 192.168.212.20 [preauth]
Jun 21 17:57:08 hh23-12 sshd[34030]: Accepted publickey for root from 10.117.212.20 port
```

```
62819
Jun 22 16:42:40 hh23-12 sshd[8485]: Invalid user user1 from 10.117.212.20
Jun 22 16:42:40 hh23-12 sshd[8485]: input_userauth_request: invalid user user1 [preauth]
s
```

Verifying SELinux

To minimize the impact of a security breach on a Cisco NFVI server, the Cisco VM enables SELinux (Security Enhanced Linux) to protect the server resources. To validate that SELinux is configured and running in enforcing mode, use the **sestatus** command to view the status of SELinux and verify that its status is enabled and in enforcing mode. For example:

```
[root@mgmt1 ~]# /usr/sbin/sestatus -v
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

Validating Port Listening Services

To prevent access by unauthorized users and processes, Cisco NFVI has no extra services listening on network ports. To verify this, use the netstat -plnt command to get a list of all services listening on the node and verify that no unauthorized services are listening. For example:

```
[root@-control-server-1 ~]# netstat -plnt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address     State       PID/Program
name
tcp        0      0 23.23.4.101:8776        0.0.0.0:*           LISTEN      24468/python2
tcp        0      0 23.23.4.101:5000        0.0.0.0:*           LISTEN      19874/httpd
tcp        0      0 23.23.4.101:5672        0.0.0.0:*           LISTEN      18878/beam.smp

tcp        0      0 23.23.4.101:3306        0.0.0.0:*           LISTEN      18337/mysqld
tcp        0      0 127.0.0.1:11211         0.0.0.0:*           LISTEN      16563/memcached
tcp        0      0 23.23.4.101:11211       0.0.0.0:*           LISTEN      16563/memcached
tcp        0      0 23.23.4.101:9292        0.0.0.0:*           LISTEN      21175/python2
tcp        0      0 23.23.4.101:9999        0.0.0.0:*           LISTEN      28555/python
tcp        0      0 23.23.4.101:80          0.0.0.0:*           LISTEN      28943/httpd
tcp        0      0 0.0.0.0:4369            0.0.0.0:*           LISTEN      18897/epmd

tcp        0      0 127.0.0.1:4243          0.0.0.0:*           LISTEN      14673/docker

tcp        0      0 0.0.0.0:22              0.0.0.0:*           LISTEN      2909/sshd

tcp        0      0 23.23.4.101:4567        0.0.0.0:*           LISTEN      18337/mysqld

tcp        0      0 23.23.4.101:15672       0.0.0.0:*           LISTEN      18878/beam.smp

tcp        0      0 0.0.0.0:35672           0.0.0.0:*           LISTEN      18878/beam.smp

tcp        0      0 127.0.0.1:25            0.0.0.0:*           LISTEN      4531/master

tcp        0      0 23.23.4.101:35357       0.0.0.0:*           LISTEN      19874/httpd
tcp        0      0 23.23.4.101:8000        0.0.0.0:*           LISTEN      30505/python
tcp        0      0 23.23.4.101:6080        0.0.0.0:*           LISTEN      27996/python2
tcp        0      0 23.23.4.101:9696        0.0.0.0:*           LISTEN      22396/python2
tcp        0      0 23.23.4.101:8004        0.0.0.0:*           LISTEN      30134/python
tcp        0      0 23.23.4.101:8773        0.0.0.0:*           LISTEN      27194/python2
tcp        0      0 23.23.4.101:8774        0.0.0.0:*           LISTEN      27194/python2
tcp        0      0 23.23.4.101:8775        0.0.0.0:*           LISTEN      27194/python2
tcp        0      0 23.23.4.101:9191        0.0.0.0:*           LISTEN      20752/python2
tcp6       0      0 :::9200                 :::*                LISTEN      18439/xinetd
tcp6       0      0 :::4369                 :::*                LISTEN      18897/epmd
tcp6       0      0 :::22                   :::*                LISTEN      2909/sshd
tcp6       0      0 ::1:25                  :::*                LISTEN      4531/master
```

Validating Non-Root Users for OpenStack Services

To prevent unauthorized access, Cisco NFVI runs OpenStack processes as a non-root user. To verify OpenStack processes are not running as root, use the **ps** command to get a list of all node processes. In the following example the user is 162:

```
[root@control-server-1 ~]# ps -aux | grep nova-api
162      27194  0.6  0.0 360924 132996 ?       S    Aug08  76:58 /usr/bin/python2
/usr/bin/nova-api
162      27231  0.0  0.0 332192 98988 ?        S    Aug08   0:01 /usr/bin/python2
/usr/bin/nova-api
162      27232  0.0  0.0 332192 98988 ?        S    Aug08   0:01 /usr/bin/python2
/usr/bin/nova-api
162      27233  0.0  0.0 332192 98988 ?        S    Aug08   0:01 /usr/bin/python2
/usr/bin/nova-api
```

Verifying Password Strength

Cisco NFVI passwords can be generated in two ways during installation:

- The Cisco NFVI installer generates unique passwords automatically for each protected service.
- You can provide an input file containing the passwords you prefer.

Cisco-generated passwords are unique, long, and contain a mixture of uppercase, lowercase, and numbers. If you provide the passwords, password strength is your responsibility.

You can view the passwords by displaying the secrets.yaml file. For example:

```
[root@mgmt1 ~]# cat ~/openstack-configs/secrets.yaml
ADMIN_USER_PASSWORD: QaZ12n13wvvNY7AH
CINDER_DB_PASSWORD: buJL8pAfytoJ0Icm
```

```
CINDER_KEYSTONE_PASSWORD: AYbcB8mx6a5Ot549
CLOUDPULSE_KEYSTONE_PASSWORD: HAT6vbl7Z56yZLtN
COBBLER_PASSWORD: bax8leYFyyDon0ps
CPULSE_DB_PASSWORD: aYGSzURpGChztbMv
DB_ROOT_PASSWORD: bjb3Uvwus6cvaNe5
KIBANA_PASSWORD: c50e57Dbm7LF0dRV
[…]
```

Reconfiguring Passwords and OpenStack Configurations

Note This topic does not apply if you have installed the optional Cisco Virtual Topology System. For information about use of passwords when VTS is installed, see the *Installing Cisco VTS* section in the *Cisco NFV Infrastructure 2.2 Installation Guide*.

You can reset some configurations after installation including the OpenStack service password and debugs, TLS certificates, and ELK configurations. Two files, secrets.yaml and openstack_config.yaml which are located in : /root/installer-{tag id}/openstack-configs/, contain the passwords, debugs, TLS file location, and ELK configurations. Also, Elasticsearch uses disk space for the data that is sent to it. These files can grow in size, and Cisco VIM has configuration variables that establishes the frequency and file size under which they are rotated.

Cisco VIM installer generates the OpenStack service and database passwords with 16 alphanumeric characters and stores those in /root/openstack-configs/secrets.yaml. You can change the OpenStack service and database passwords using the password reconfigure command on the deployed cloud. The command identifies the containers affected by the password change and restarts them so the new password can take effect.

Note Always schedule password reconfiguration in a maintenance window because container restarts might disrupt the control plane

.

Run the following command to view the list of passwords and configurations :

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 installer-xxxx]# ciscovim help reconfigure
usage: ciscovim reconfigure [--regenerate_secrets] [--setpassword <secretkey>]
                            [--setopenstackconfig <option>]

Reconfigure the openstack cloud
Optional arguments:
  --regenerate_secrets           Regenerate All Secrets
  --setpassword <secretkey>      Set of secret keys to be changed.
  --setopenstackconfig <option>  Set of Openstack config to be changed.
[root@mgmt1 ~]# ciscovim list-openstack-configs
+-------------------------------+-----------------------------------------+
|              Name             |                 Option                  |
+-------------------------------+-----------------------------------------+
|       CINDER_DEBUG_LOGGING     |                 False                   |
|      KEYSTONE_DEBUG_LOGGING    |                 False                   |
|    CLOUDPULSE_VERBOSE_LOGGING  |                  True                   |
|      MAGNUM_VERBOSE_LOGGING    |                  True                   |
|        NOVA_DEBUG_LOGGING      |                  True                   |
|     NEUTRON_VERBOSE_LOGGING    |                  True                   |
|      external_lb_vip_cert     |  /root/openstack-configs/haproxy.pem   |
|     GLANCE_VERBOSE_LOGGING    |                  True                  |
                      |
|     elk_rotation_frequency    |                monthly                 |
|   CEILOMETER_VERBOSE_LOGGING  |                  True                  |
|     elk_rotation_del_older    |                   10                   |
|       HEAT_DEBUG_LOGGING      |                 False                  |
|    KEYSTONE_VERBOSE_LOGGING   |                  True                  |
|     external_lb_vip_cacert    | /root/openstack-configs/haproxy-ca.crt |
|      MAGNUM_DEBUG_LOGGING     |                  True                  |
|     CINDER_VERBOSE_LOGGING    |                  True                  |
|       elk_rotation_size       |                   2                    |
|    CLOUDPULSE_DEBUG_LOGGING   |                 False                  |
|     NEUTRON_DEBUG_LOGGING     |                  True                  |
|      HEAT_VERBOSE_LOGGING     |                  True                  |
|    CEILOMETER_DEBUG_LOGGING   |                 False                  |
|      GLANCE_DEBUG_LOGGING     |                 False                  |
|      NOVA_VERBOSE_LOGGING     |                  True                  |
+-------------------------------+----------------------------------------+
[root@mgmt1 installer-xxxx]#
[root@mgmt1 installer-xxxx]# ciscovim list-password-keys
+----------------------------------+
| Password Keys                    |
+----------------------------------+
| COBBLER_PASSWORD                 |
| CPULSE_DB_PASSWORD               |
| DB_ROOT_PASSWORD                 |
| KIBANA_PASSWORD                     |
| GLANCE_DB_PASSWORD               |
| GLANCE_KEYSTONE_PASSWORD         |
| HAPROXY_PASSWORD                 |
| HEAT_DB_PASSWORD                 |
| HEAT_KEYSTONE_PASSWORD           |
| HEAT_STACK_DOMAIN_ADMIN_PASSWORD |
| HORIZON_SECRET_KEY               |
| KEYSTONE_ADMIN_TOKEN             |
| KEYSTONE_DB_PASSWORD             |
| METADATA_PROXY_SHARED_SECRET     |
| NEUTRON_DB_PASSWORD              |
| NEUTRON_KEYSTONE_PASSWORD        |
| NOVA_DB_PASSWORD                 |
| NOVA_KEYSTONE_PASSWORD           |
| RABBITMQ_ERLANG_COOKIE           |
| RABBITMQ_PASSWORD                |
| WSREP_PASSWORD                   |
+----------------------------------+
[root@mgmt1 installer-xxxx]#
```

You can change specific password and configuration identified from the available list.

Run the reconfiguration command as follows:

```
[root@mgmt1 ~]# ciscovim help reconfigure
usage: ciscovim reconfigure [--regenerate_secrets] [--setpassword <secretkey>]
                            [--setopenstackconfig <option>]

Reconfigure the Openstack cloud

Optional arguments:
  --regenerate_secrets           Regenerate All Secrets
  --setpassword <secretkey>      Set of secret keys to be changed.
  --setopenstackconfig <option>  Set of Openstack config to be changed.
[root@mgmt1 ~]# ciscovim reconfigure --setpassword ADMIN_USER_PASSWORD,NOVA_DB_PASSWORD
--setopenstackconfig HEAT_DEBUG_LOGGING,HEAT_VERBOSE_LOGGING
```

```
Password for ADMIN_USER_PASSWORD:
Password for NOVA_DB_PASSWORD:
Enter T/F for option HEAT_DEBUG_LOGGING:T
Enter T/F for option HEAT_VERBOSE_LOGGING:T
```

The password must be alphanumeric and can be maximum 32 characters in length.

Following are the configuration parameters for OpenStack:

Configuration Parameter	Allowed Values
CEILOMETER_DEBUG_LOGGING	T/F (True or False)
CEILOMETER_VERBOSE_LOGGING	T/F (True or False)
CINDER_DEBUG_LOGGING	T/F (True or False)
CINDER_VERBOSE_LOGGING	T/F (True or False)
CLOUDPULSE_DEBUG_LOGGING	T/F (True or False)
CLOUDPULSE_VERBOSE_LOGGING	T/F (True or False)
GLANCE_DEBUG_LOGGING	T/F (True or False)
GLANCE_VERBOSE_LOGGING	T/F (True or False)
HEAT_DEBUG_LOGGING	T/F (True or False)
HEAT_VERBOSE_LOGGING	T/F (True or False)
KEYSTONE_DEBUG_LOGGING	T/F (True or False)
KEYSTONE_VERBOSE_LOGGING	T/F (True or False)
MAGNUM_DEBUG_LOGGING	T/F (True or False)
MAGNUM_VERBOSE_LOGGING	T/F (True or False)
NEUTRON_DEBUG_LOGGING	T/F (True or False)
NEUTRON_VERBOSE_LOGGING	T/F (True or False)
NOVA_DEBUG_LOGGING	T/F (True or False)
NOVA_VERBOSE_LOGGING	T/F (True or False)
elk_rotation_del_older	Days after which older logs are purged
elk_rotation_frequency	Available options: "daily", "weekly", "fortnightly", "monthly"
elk_rotation_size	Gigabytes (entry of type float/int is allowed)
external_lb_vip_cacert	Location of HAProxy CA certificate
external_lb_vip_cert	Location of HAProxy certificate

NOVA_RAM_ALLOCATION_RATIO	Mem oversubscription ratio (from 1.0 to 4.0)
NOVA_CPU_ALLOCATION_RATIO	CPU allocation ratio (from 1.0 to 16.0)
ES_SNAPSHOT_AUTODELETE	Elastic search auto-delete configuration, can manage the following: period: ["hourly", "daily", "weekly", "monthly"] # Frequency of cronjob to check for disk space threshold_warning: <1-99> # % of disk space occupied to display warning message threshold_low: <1-99> # % of disk space occupied after cleaning up snapshots threshold_high: <1-99> # % of disk space when starting to delete snapshots

Alternatively, you can regenerate all passwords using regenerate_secrets command option as follows:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim reconfigure --regenerate_secrets
```

In addition to the services passwords, you can change the debug and verbose options for Heat, Glance, Cinder, Nova, Neutron, Keystone and Cloudpulse in /root/openstack-configs/openstack_config.yaml. You can modify the other configurations including the ELK configuration parameters, API and Horizon TLS certificates, Root CA, NOVA_EAMALLOCATION_RATIO and ES_SNAPSHOT_AUTODELETE. When reconfiguring these options (For Example API and TLS), some control plane downtime will occur, so plan the changes during maintenance windows.

The command to reconfigure these elements are:

```
ciscovim reconfigure
```

The command includes a built-in validation to ensure you do not enter typos in the secrets.yaml or openstack_config.yaml files.

When reconfiguration of password or enabling of openstack-services fails, all subsequent pod management operations are blocked. In such casees, we recommend that you contact Cisco TAC to resolve the situation.

Enabling NFVIMON Post Pod Install

The dispatcher is the only component in NFVIMON offering 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 NFVI Zenpack sitting in the CC/RM 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 part of 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 part of 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.

```
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
      ....
```

To initiate the integration of NFVIMON on an existing pod, copy the setupdata into a local dir and update it manually with information listed above, and then run reconfiguration command as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include NFVIMON related
info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

It should be noted that un-configuration of this feature is not supported today. Additionally, NFVIMON is supported only on a pod running with Keystone v2.

Reconfiguring CIMC Password on an Existing Install

Cisco VIM, allows you to reconfigure the CIMC password on an existing install along with OpenStack services.

Note You must have a C-series pod, up and running with Cisco to reconfigure the CIMC password.

Step 1 Update the cimc_password in the CIMC-COMMON section, and/or the individual cimc_password for each server and then run the reconfigure option provided by Ciscovimclient.

```
CIMC-COMMON:
  cimc_username: "admin"
  cimc_password: <"new password">
:
:
SERVERS:
:
control-server-2:
  cimc_info: {'cimc_ip': '<ip_addr>',
             'cimc_username': 'admin',
            'cimc_password': <'update with new passowrd'>} # only needed if each server has specific
 password
```

Step 2 To change the CIMC password for the pod, copy the setupdata into a local location and update it manually with the CIMC password as shown in the snippet above. The new password must satisfy atleast three of the following conditions:

Note Do not change CIMC password directly into the exiting /root/openstack-configs/setup_data.yaml file.

- Must contain at least one lower case letter.
- Must contain at least one upper case letter.
- Must contain at least one digit between 0 to 9.
- One of these special characters !$#@%^-_+=*&
- Your password has to be 8 to 14 characters long.

Step 3 Run the vim reconfiguration command to post update the setup_data as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# cp <my_setup_data.yaml> <my_setup_data_original.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the relevant CIMC  setup_data to include LDAP info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

Note After successful completion of the CIMC Password, reconfigure operation triggers an auto-back when the management node auto-back recovery feature is enabled. If the CIMC Password reconfigure fails, contact Cisco TAC to recover from the failure.

Increasing Provider and Tenant VLAN Ranges

Cisco VIM, provides the flexibility of increasing the provider and tenant VLAN ranges after the post pod installation. Increasing provider and tenant VLAN ranges applies to C-series and B-series pod that is enabled with Cisco UCS Manager plugin. B-series pod running without Cisco UCS Manager plugin, cannot use this feature because of the inherent day-0 networking configuration to be done in FI.

Note You should have the tenant and provider networks enabled on the pod from day-0.

To increase provider and tenant VLAN ranges enter the TENANT_VLAN_RANGES and/or PROVIDER_VLAN_RANGES in the setup_data.yaml file and run the reconfigure command through Ciscovimclient as follows:

```
TENANT_VLAN_RANGES: old_vlan_info, new_vlan_info
or/and
PROVIDER_VLAN_RANGES: old_vlan_info, new_vlan_info
```

To change the pod, copy the setupdata into a local dir and update it manually by running the following command:

```
[root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir [root@mgmt1 ~]# cd MyDir
```

Update the setup_data, by running the following command:

```
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml> [root@mgmt1
 ~]# vi my_setup_data.yaml (update the setup_data with the right info)
```

Run the re-configuration command as follows:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ./ciscovimclient/ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

Fernet Key Operations

Keystone fernet token format is based on the cryptographic authentication method - Fernet. Fernet is an implementation of Symmetric Key Encryption. Symmetric key encryption is a cryptographic mechanism that uses the same cryptographic key to encrypt plaintext and the same cryptographic key to decrypt ciphertext. Fernet authentication method also supports multiple keys where it takes a list of symmetric keys, performs all encryption using the first key in a list and attempts to decrypt using all the keys from that list.

The Cisco NFVI pods uses Fernet keys by default. The following operations can be carried out in Cisco NFVI pods.

To check if the fernet keys are successfully synchronized across the keystone nodes.

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim help check-fernet-keys
usage: ciscovim check-fernet-keys

Check whether the fernet keys are successfully synchronized across keystone nodes.
```

To set the fernet key frequency:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim help period-rotate-fernet-keys
usage: ciscovim period-rotate-fernet-keys <SET_PERIOD_ROTATION_FERNET_KEYS>
```

```
Set the frequency of fernet keys rotation on keystone
Positional arguments:
  <SET_PERIOD_ROTATION_FERNET_KEYS>
Frequency to set for period rotation
```

To forcefully rotate the fernet keys:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim help rotate-fernet-keys
usage: ciscovim rotate-fernet-keys
Trigger rotation of the fernet keys on keystone
```

To resync the fernet keys across the keystone nodes:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim help resync-fernet-keys
usage: ciscovim resync-fernet-keys
Resynchronize the fernet keys across all the keystone nodes
```

Managing Certificates

When TLS protection is configured for the OpenStack APIs, the two certificate files, haproxy.pem and haproxy-ca.crt, are stored in the /root/openstack-configs/ directory. Clients running on servers outside of the deployed cloud to verify cloud authenticity need a copy of the root certificate (haproxy-ca.crt). If a well-known certificate authority has signed the installed certificate, no additional configuration is needed on client servers. However, if a self-signed or local CA is used, copy haproxy-ca.crt to each client. Following instructions specific to the client operating system or browser to install the certificate as a trusted certificate.

Alternatively, you can explicitly reference the certificate when using the OpenStack CLI by using the environment variable OS_CACERT or command line parameter –cacert.

While Cisco NFVI is operational, a daily check is made to monitor the expiration dates of the installed certificates. If certificates are not nearing expiration, an informational message is logged. As the certificate approaches expiration, an appropriate warning or critical message is logged.

```
2017-04-24T13:56:01 INFO Certificate for OpenStack Endpoints at 192.168.0.2:5000 expires
in 500 days
```

It is important to replace the certificates before they expire. After Cisco NFVI is installed, you can update the certificates by replacing the haproxy.pem and haproxy-ca.crt files and running the reconfigure command:

```
 cd ~/installer-xxxx; ciscovim reconfigure
```

Reconfiguring TLS Certificates

Cisco VIM provides a way to configure TLS certificates on-demand for any reason. For Example: certificate expiration policies governing certificate management.

Reconfiguration of certificates in general is supported in the following components:

- Cisco VIM Rest API endpoints:

  Steps to be performed to reconfigure certificate files are as follows:

  - Copy the new key, CA root and certificate files into the ~/openstack-configs folder under the following filenames

```
 cp <new-ca-root-cert> ~/openstack-configs/mercury-ca.crt
 cp <new-key-file> ~/openstack-configs/mercury.key
 cp <new-cert-file> ~/openstack-configs/mercury.crt
```

  - Once copied run the reconfigure steps as under:

    ```
     cd ~/installer-xxxx/tools
     ./restapi.py -a reconfigure-tls
    ```

- OpenStack API endpoints

  Steps to be performed to reconfigure certificate files are as follows:

  - Copy the new key, CA root and certificate files into the ~/openstack-configs folder under the following filenames

    ```
    cp <new-ca-root-cert> ~/openstack-configs/haproxy-ca.crt
    cp <new-cert-file> ~/openstack-configs/haproxy.pem
    ```

  - Once copied run the reconfigure steps as follows:

    ```
     cd ~/installer-xxxx; ciscovim reconfigure
    ```

- SwiftStack Service through Horizon and CinderBackup Service.

  - Reconfiguring TLS certificates for SwiftStack mainly involves client side certificate updates. The CA root certificate in both these cases is updated for components within OpenStack that are clients of the SwiftStack service in general.

  - Copy the new CA root certificate to the ~/openstack-configs folder and run reconfigure.

    ```
    cp <new-ca-root-cert> ~/openstack-configs/haproxy-ca.crt
     cd ~/installer-xxxx; ciscovim reconfigure
    ```

- Logstash service and Fluentd (client-side certificates).

  - For the Logstash service on the management node, both the key and certificate file are reconfigured as part of the reconfigure operation.

  - For the Fluentd service on the controllers, compute and storage nodes, the certificate file are reconfigured as part of the reconfigure operation.

  - Copy of the key and certificate files to the ~/openstack-configs folder on the management node and run reconfigure operation.

    ```
    cp <new-key-file> ~/openstack-configs/logstash-forwarder.key
     cp <new-cert-file> ~/openstack-configs/logstash-forwarder.crt
    cd ~/installer-xxxx; ciscovim reconfigure
    ```

Enabling Keystone v3 on an Existing Install

To continue enhancing our security portfolio, and multi-tenancy with the use of domains, Keystone v3 support has been added in Cisco VIM from an authentication end-point. It should be noted that Keystone v2 and v3 are mutually exclusive. The administrator has to decide during install time the authentication end-point version to go with. By default, VIM orchestrator picks keystone v2 as the authentication end-point. So one can enable Keystonev3 as an install option on day-0 (see 2.2 CiscoVIM install guide), or enable it as a reconfigure option after the pod is installed. To enable Keystone v3 after the pod is installed, one needs to define the following under the optional service section in the setup_data.yam filel.

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

To initiate the integration of Keystone v3 on an existing pod, copy the setupdata into a local dir and update it manually, then run reconfiguration command as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include keystone v3 info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

It should be noted that un-configuration of this feature is not supported today. Additionally, re-versioning Keystone API from v3 to v2 is also not supported.

LDAP/AD support with Keystone v3

With the introduction of KeystoneV3, the openstack service authentication can now we delegated to an external LDAP/AD server. In Cisco VIM, this feature has been introduced optionally if the authorization is done by Keystone v3. Just like Keystonev3, this feature can be enabled on an existing pod running Cisco VIM. To avail of this feature post pod deployment, the setup_data needs to be augmented with the following information during the installation of the pod.

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

```
LDAP:
  domain: <Domain specific name>
  user_objectclass: <objectClass for Users> # e.g organizationalPerson
  group_objectclass: <objectClass for Groups> # e.g. groupOfNames
  user_tree_dn: '<DN tree for Users>' # e.g. 'ou=Users,dc=cisco,dc=com'
  group_tree_dn: '<DN tree for Groups>' # e.g. 'ou=Groups,dc=cisco,dc=com'
  suffix: '<suffix for DN>' # e.g. 'dc=cisco,dc=com'
  url: '<ldap:// host:port>' # e.g. 'ldap://172.26.233.104:389'
or
url: '<ldaps|ldap>://[<ip6-address>]:[port]'
e.g.ldap://[2001:420:293:2487:d1ca:67dc:94b1:7e6c]:389 ---> note the mandatory "[.. ]"
around the ipv6 address
  user: '<DN of bind user>' # e.g. 'dc=admin,dc=cisco,dc=com'
  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
```

To initiate the integration of LDAP with Keystone v3 on an existing pod, copy the setupdata into a local dir and update it manually with the relevant LDAP and Keystone v3 (if absent from before) configuration, then run reconfiguration command as follows:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
```

```
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include LDAP info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

The reconfigure feature supports a full or partial reconfiguration of the LDAP integration service.

Note All the parameters within the LDAP stanza are configurable with the exception of the domain parameter.

Integrating identity with LDAP over TLS: The automation supports keystone integration with LDAP over TLS. In order to enable TLS, the CA root certificate must be presented as part of the /root/openstack-configs/haproxy-ca.crt file. The url parameter within the LDAP stanza must be set to ldaps.

Additionally, the url parameter supportsthe following format: url: '<ldaps | ldap>://<FQDN | IP-Address>:[port]'

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

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

The port number is optional and if not provided assumes that the ldap services are running on the default ports. For Example:. 389 for non-ssl and 636 for ssl. However, if these are not the defaults, then the non-standard port numbers must be provided. Except for the domain, all other item values can be changed via the 'reconfigure' option.

Moving Netapp transport from http to https

For deployements, with NETAPP running over http protocol you can migrate it to https, post-deployment.

Step 1 To initiate the change, copy the setupdata into a local dir and update it manually the name/value pair in the netapp section:

```
NETAPP:
   …
   ….
  server_port: 443
  transport_type: https
  ….
  netapp_cert_file: <root ca path for netapp cluster only if protocol is https>
```

Step 2 Excute the following commands to update the netapp section:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir [root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to netapp section as listed above)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

Enabling ACI in Cisco VIM

With Cisco VIM, we have integrated the Opflex ML2 plugin (in Unified mode) to manage the tenant VLANs dynamically, as VMs come and go in the cloud. In addition, we support an administrator driven automated workflow to provison the provider networks. In VIM, this is supported on a C-series based Fullon or micropod running with Cisco VIC 1227. While the integration of ACI into Cisco VIM is a day-0 activity, VIM supports the replacement of the ACI controller in the ACI cluster, and the expansion of the leaf switches to increase the fabric.

Step 1 To update the setup_data with one or both follow the below steps:

```
APICINFO:
apic_hosts: '<ip1|host1>:[port], <ip2|host2>:[port], <ip3|host3>:[port]'
# max of 3, min of 1, not 2; reconfigurable

Since the APIC manages the Leaf switches, its mandatory to define the new Leaf switches (in pairs)
in the following format:

TORSWITCHINFO:  (mandatory)

  SWITCHDETAILS:
 :
 :
   -
   hostname: <leaf-hostname-1>
   vpc_peer_keepalive: <leaf-hostname-2>
   vpc_domain: 1  # Must be unique across 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
```

Step 2 T o initiate the change in ACI config on an existing pod, copy the setupdata into a local dir and update it manually with the relevantapic_hosts and/or new TORSWITCH information, then run reconfiguration commands follows:

```
[root@mgmt1 ~]# cd /root/ [root@mgmt1 ~]# mkdir MyDir [root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml> [root@mgmt1 ~]# vi
my_setup_data.yaml (update the setup_data to include ACI info)
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

Hardening Cisco VIM Deployment

If you want to harden the Cisco VIM deployment, set up the firewalls ahead of the External Interfaces.

The following tables provide information about the expected traffic from the management interfaces of Cisco VIM.

Table 3: Management Nodes

Interface	Direction	Protocol	Port	Application	Note
br_api	incoming	HTTPS	8445	Insight	

Interface	Direction	Protocol	Port	Application	Note
br_api	incoming	HTTPS	8008	Insight API	
br_api	incoming	HTTPS	5601	Kibana	
br_api	incoming	SSH	22	SSH	
br_api	incoming	HTTPS	3000	CVIM MON	New in v2.4.1
br_api	outgoing	NTP	123	NTP	
br_api	outgoing	DNS	53	DNS	
br_api	outgoing	Syslog	514	Syslog	

Table 4: Controller Nodes

Interface	Direction	Protocol	Port	Application	Note
external_lb_vip	incoming	HTTP	80	Redirects to 443	
external_lb_vip	incoming	HTTPS	443	Horizon	
external_lb_vip	incoming	HTTPS	8774	Nova	
external_lb_vip	incoming	HTTPS	6080	Nova NoVNC Proxy	
external_lb_vip	incoming	HTTPS	9696	Neutron	
external_lb_vip	incoming	HTTPS	8776	Cinder	
external_lb_vip	incoming	HTTPS	9292	Galance	
external_lb_vip	incoming	HTTPS	8000	Heat	
external_lb_vip	incoming	HTTPS	8004	Heat	

CHAPTER 5

Managing Cisco NFVI Storage

This chapter describes basic architectural concepts that will help you understand the Cisco NFVI data storage architecture and data flow. It also provides techniques you can use to monitor the storage cluster health and the health of all systems that depend on it

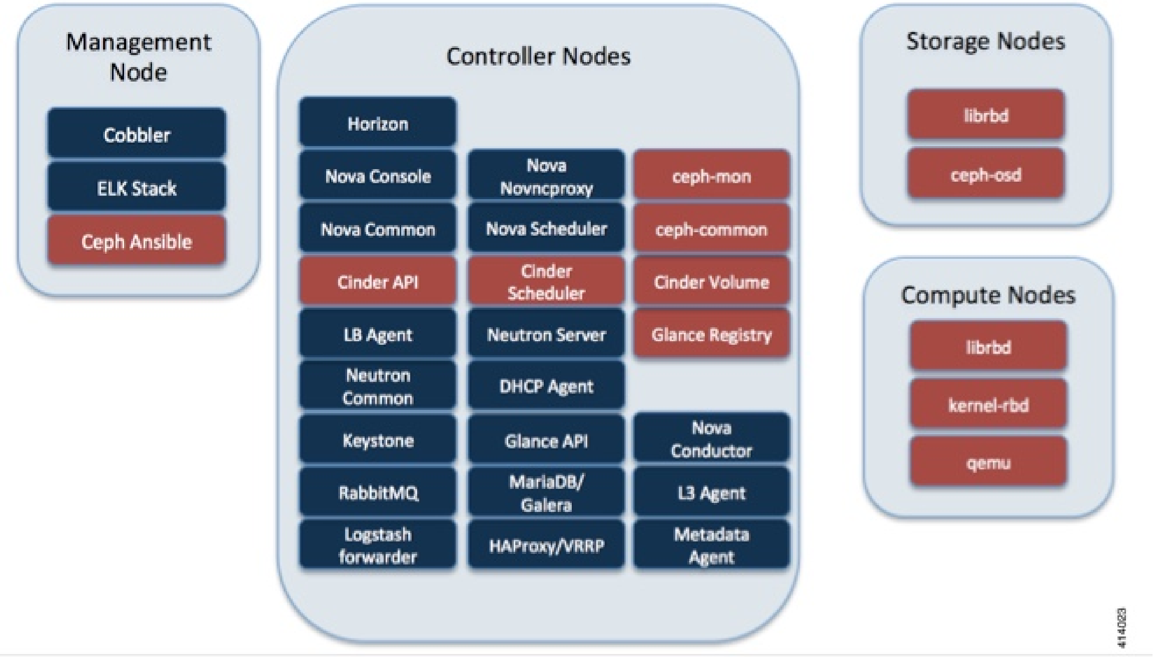
Cisco NFVI Storage Architecture

OpenStack has multiple storage back ends. Cisco NFVI uses the Ceph back end. Ceph supports both block and object storage and is therefore used to store VM images and volumes that can be attached to VMs. Multiple OpenStack services that depend on the storage backend include:

- Glance (OpenStack image service)—Uses Ceph to store images.
- Cinder (OpenStack storage service)—Uses Ceph to create volumes that can be attached to VMs.
- Nova (OpenStack compute service)—Uses Ceph to connect to the volumes created by Cinder.

The following figure shows the Cisco NFVI storage architecture component model.

Figure 18: Cisco NFVI Storage Architecture



Verifying and Displaying Ceph Storage Pools

Ceph is configured with four independent pools: images, volumes, vms, and backups. (A default rbd pool is used internally.) Each Ceph pool is mapped to an OpenStack service. The Glance service stores data in the images pool, and the Cinder service stores data in the volumes pool. The Nova service can use the vms pool to boot ephemeral disks directly from the Ceph cluster depending on how the NOVA_BOOT_FROM option in the ~/openstack-configs/setup_data.yaml was configured prior to Cisco NFVI installation. If NOVA_BOOT_FROM is set to ceph before you run the Cisco NFVI installation, the Nova service boot up from the Ceph vms pool. By default, NOVA_BOOT_FROM is set to local, which means that all VM ephemeral disks are stored as files in the compute nodes. Changing this option after installation does not affect the use of the vms pool for ephemeral disks.

The Glance, Cinder, and Nova OpenStack services depend on the Ceph cluster for backend storage. Therefore, they need IP connectivity to the controller nodes. The default port used to connect Glance, Cinder, and Nova to the Ceph cluster is 6789. Authentication through cephx is required, which means authentication tokens, called keyrings, must be deployed to the OpenStack components for authentication.

To verify and display the Cisco NFVI Ceph storage pools:

Step 1 Launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

Step 3 List the Ceph pools:

```
cephmon_4612 [root@controller_server-1 ~]# ceph osd lspools
0 rbd,1 images,2 volumes,3 vms,4 backups,
```

Step 4 List the images pool content:

```
cephmon_4612 [ceph@controller_server-1 /]$ rbd list images
a4963d51-d3b7-4b17-bf1e-2ebac07e1593
```

Checking the Storage Cluster Health

Cisco recommends that you perform a few verifications to determine whether the Ceph cluster is healthy and is connected to the Glance, Cinder, and Nova OpenStack services, which have Ceph cluster dependencies. The first task to check the health of the cluster itself by completing the following steps:

Step 1 From the Cisco NFVI management node, launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

Step 3 Check the Ceph cluster status:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ ceph status
```

Sample response:

```
cluster dbc29438-d3e0-4e0c-852b-170aaf4bd935
     health HEALTH_OK
     monmap e1: 3 mons at {ceph-controller_server-1=20.0.0.7:6789/0,
    ceph-controller_server-2=20.0.0.6:6789/0,ceph-controller_server-3=20.0.0.5:6789/0}
            election epoch 8, quorum 0,1,2 ceph-controller_server-3,
    ceph-controller_server-2,ceph-controller_server-1
     osdmap e252: 25 osds: 25 up, 25 in
      pgmap v593: 1024 pgs, 5 pools, 406 MB data, 57 objects
            2341 MB used, 61525 GB / 61527 GB avail
                1024 active+clean
```

This example displays three monitors, all in good health, and 25 object storage devices (OSDs). All OSDs show as up and in the cluster.

Step 4 To see a full listing of all OSDs sorted by storage node, enter:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ ceph osd tree
```

Sample response:

```
ID WEIGHT   TYPE NAME               UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 60.18979 root default
-2 18.96994     host controller_server-2
 1  2.70999         osd.1                up  1.00000          1.00000
 5  2.70999         osd.5                up  1.00000          1.00000
 6  2.70999         osd.6                up  1.00000          1.00000
```

```
11  2.70999         osd.11              up  1.00000          1.00000
12  2.70999         osd.12              up  1.00000          1.00000
17  2.70999         osd.17              up  1.00000          1.00000
20  2.70999         osd.20              up  1.00000          1.00000
-3 18.96994     host controller_server-1
 0  2.70999         osd.0               up  1.00000          1.00000
 4  2.70999         osd.4               up  1.00000          1.00000
 8  2.70999         osd.8               up  1.00000          1.00000
10  2.70999         osd.10              up  1.00000          1.00000
13  2.70999         osd.13              up  1.00000          1.00000
16  2.70999         osd.16              up  1.00000          1.00000
18  2.70999         osd.18              up  1.00000          1.00000
-4 18.96994     host controller_server-3
 2  2.70999         osd.2               up  1.00000          1.00000
 3  2.70999         osd.3               up  1.00000          1.00000
 7  2.70999         osd.7               up  1.00000          1.00000
 9  2.70999         osd.9               up  1.00000          1.00000
14  2.70999         osd.14              up  1.00000          1.00000
15  2.70999         osd.15              up  1.00000          1.00000
19  2.70999         osd.19              up  1.00000          1.00000
-5  3.27997     host controller_server-4
21  0.81999         osd.21              up  1.00000          1.00000
22  0.81999         osd.22              up  1.00000          1.00000
23  0.81999         osd.23              up  1.00000          1.00000
24  0.81999         osd.24              up  1.00000          1.00000
```

What to do next

After you verify the Ceph cluster is in good health, check that the individual OpenStack components have connectivity and their authentication tokens—keyrings—match the Ceph Monitor keyrings. The following procedures show how to check the connectivity and authentication between Ceph and Glance, Ceph and Cinder, and Ceph and Nova.

Checking Glance Connectivity

The Glance API container must be connected to the Cisco NFVI controller nodes. Complete the following steps to verify the Glance to controller node connectivity:

Step 1 From the management node, launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Navigate to the Glance API container:

```
[root@controller_server-1 ~]# glanceapi
```

Step 3 Check the Glance API container connectivity to a controller node different from the one entered in Step 1, in this case, controller_server 2:

```
glanceapi_4612 [glance@controller_server-1 /]$ curl controller_server-2:6789
```

If the connection is successful, you see a message like the following:

```
glanceapi_4612 [glance@controller_server-1 /]$ curl controller_server-2:6789
ceph v027?
```

If the connection is not successful, you see a message like the following:

```
glanceapi_4612 [glance@controller_server-1 /]$ curl controller_server-2:6789
curl: (7) Failed connect to controller_server-2:6789; Connection refused
```

A message like the one above means the Ceph monitor running on the target controller node controller_server-2 is not listening on the specified port or there is no route to it from the Glance API container.

Checking one controller node should be enough to ensure one connection path available for the Glance API. However, because Cisco NFVI controller nodes run as part of an HA cluster, you should run Step 3 above targeting all the controller nodes in the Cisco NFVI pod.

What to do next

After you verify the Glance API connectivity to all Cisco NFVI controller nodes, check the Glance keyring to ensure it matches the Ceph monitor keyring.

Verifying Glance and Ceph Monitor Keyrings

Complete the following steps to verify the Glance API keyring matches the Ceph Monitor keyring.

Step 1 Launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Navigate to the Glance API container:

```
[root@controller_server-1 ~]# glanceapi
```

Step 3 Check the Glance keyring content, for example:

```
glanceapi_4612 [glance@controller_server-1 /]$ cat /etc/ceph/client.glance.keyring
[client.glance]
key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

Step 4 Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

Step 5 Display the Ceph Monitor keyring content:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ cat /etc/ceph/ceph.client.glance.keyring
[client.glance]

key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

Verify the keyring matches the Glance API keyring displayed in Step 3.

What to do next

A final check to ensure that Ceph and Glance are connected is to actually import a Glance image using Horizon or the Glance CLI. After you import an image, compare the IDs seen by Glance and by Ceph. They should match, indicating Ceph is handling the backend for Glance.

Verifying Glance Image ID on Ceph

The following steps verify Ceph is properly handling new Glance images by checking that the image ID for a new Glance image is the same as the image ID displayed in Ceph.

Step 1 From the management node, load the OpenStack authentication variables:

```
[root@management-server-cisco ~]# source ~/openstack-configs/openrc
```

Step 2 Import any Glance image. In the example below, a RHEL 7.1 qcow2 image is used.

```
[root@management-server-cisco images]# glance image-create
--name "rhel" --disk-format qcow2  --container-format bare --file
rhel-guest-image-7.1-20150224.0.x86_64.qcow2
```

Step 3 List the Glance images:

```
[root@management-server-cisco images]# glance image-list | grep rhel
| a4963d51-d3b7-4b17-bf1e-2ebac07e1593 | rhel
```

Step 4 Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

Step 5 Display the contents of the Ceph images pool:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ rbd list images | grep
a4963d51-d3b7-4b17-bf1e-2ebac07e1593
a4963d51-d3b7-4b17-bf1e-2ebac07e1593
```

Step 6 Verify that the Glance image ID displayed in Step 3 matches the image ID displayed by Ceph.

Checking Cinder Connectivity

The Cinder volume container must have connectivity to the Cisco NFVI controller nodes. Complete the following steps to verify Cinder volume has connectivity to the controller nodes:

Step 1 From the management node, launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Navigate to the Cinder volume container:

```
[root@controller_server-1 ~]# cindervolume
```

Step 3 Check the Cinder volume container connectivity to a controller node different from the one entered in Step 1, in this case, controller_server-2:

```
cindervolume_4612 [cinder@controller_server-1 /]$ curl controller_server-2:6789
```

If the connection is successful, you see a message like the following:

```
cindervolume_4612 [cinder@controller_server-1 /]$ curl controller_server-2:6789
ceph v027?
```

If the connection is not successful, you see a message like the following:

```
cindervolume_4612 [cinder@controller_server-1 /]$ curl controller_server-2:6789
curl: (7) Failed connect to controller_server-2:6789; Connection refused
```

A message like the one above means the Ceph monitor running on the target controller node controller_server-2 is not listening on the specified port or there is no route to it from the Cinder volume container.

Checking one controller node should be enough to ensure one connection path is available for the Cinder volume. However, because Cisco NFVI controller nodes run as part of an HA cluster, repeat Step 3 targeting all the controller nodes in the Cisco NFVI pod.

What to do next

After you verify the Cinder volume connectivity to all Cisco NFVI controller nodes, check the Cinder keyring to ensure it matches the Ceph monitor keyring.

Verifying the Cinder and Ceph Monitor Keyrings

Complete the following steps to verify the Cinder volume keyring matches the Ceph Monitor keyring.

Step 1 From the management node, launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Navigate to the Cinder volume container:

```
[root@controller_server-1 ~]# cindervolume
```

Step 3 Check the Cinder keyring content, for example:

```
cindervolume_4612 [cinder@controller_server-1 /]$ cat /etc/ceph/client.cinder.keyring
[client.cinder]
key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

Step 4 Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

Step 5 Display the Ceph Monitor keyring content:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ cat /etc/ceph/ceph.client.cinder.keyring
[client.cinder]

key = AQA/pY1XBAnHMBAAeS+0Wmh9PLZe1XqkIW/p0A==
```

Verify the keyring matches the Cinder volume keyring displayed in Step 3.

What to do next

As a final Ceph and Cinder connectivity verification, import a Cinder image using Horizon or the Cinder CLI. After you import the image, compare the IDs seen by Cinder and by Ceph. They should match, indicating Ceph is handling the backend for Cinder.

Verifying the Cinder Volume ID on Ceph

The following steps verify Ceph is properly handling new Cinder volumes by checking that the volume ID for a new Cinder volume is the same as the volume ID displayed in Ceph.

Step 1 From the management node, load the OpenStack authentication variables:

```
[root@management-server-cisco ~]# source ~/openstack-configs/openrc
```

Step 2 Create an empty volume:

```
[root@management-server-cisco ~]# cinder create --name ciscovol1 5
```

The preceding command creates a new 5 GB Cinder volume named ciscovol1.

Step 3 List the Cinder volumes:

```
[[root@management-server-cisco ~]# cinder list
+--------------------------------------+--------+------------------+--
|                 ID                   | Status | Migration Status |…

| dd188a5d-f822-4769-8a57-c16694841a23 | in-use |        -         |…
+----------------------------------+------+
```

Step 4 Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

Step 5 Display the contents of the Ceph volumes pool:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ rbd list volumes
volume-dd188a5d-f822-4769-8a57-c16694841a23
```

Step 6 Verify that the Cinder volume ID displayed in Step 3 matches the volume ID displayed by Ceph, excluding the "volume-" prefix.

Checking Nova Connectivity

The Nova libvirt container must have connectivity to the Cisco NFVI controller nodes. Complete the following steps to verify Nova has connectivity to the controller nodes:

Step 1 From the management node, launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@Computenode_server-1
```

Step 2 Navigate to the Nova libvirt container:

```
[root@compute_server-1 ~]# libvirt
```

Step 3 Check the Nova libvirt container connectivity to a controller node, in this case, controller_server 1:

```
novalibvirt_4612 [root@compute_server-1 /]$ curl controller_server-2:6789
```

If the connection is successful, you see a message like the following:

```
novalibvirt_4612 [root@compute_server-1 /]$ curl controller_server-1:6789
ceph v027?
```

If the connection is not successful, you see a message like the following:

```
novalibvirt_4612 [root@compute_server-1 /]$ curl controller_server-1:6789
curl: (7) Failed connect to controller_server-1:6789; Connection refused
```

A message like the one above means the Ceph monitor running on the target controller node controller_server-1 is not listening on the specified port or there is no route to it from the Nova libvirt container.

Checking one controller node should be enough to ensure one connection path available for the Nova libvirt. However, because Cisco NFVI controller nodes run as part of an HA cluster, you should run Step 3 above targeting all the controller nodes in the Cisco NFVI pod.

What to do next

After you verify the Nova libvirt connectivity to all Cisco NFVI controller nodes, check the Nova keyring to ensure it matches the Ceph monitor keyring.

Verifying the Nova and Ceph Monitor Keyrings

Complete the following steps to verify the Nova libvert keyring matches the Ceph Monitor keyring.

Step 1 From the management node, launch a SSH session to a controller node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Navigate to the Nova libvert container:

```
[root@compute_server-1 ~]# libvirt
```

Step 3 Extract the libvirt secret that contains the Nova libvirt keyring:

```
novalibvirt_4612 [root@compute_server-1 /]# virsh secret-list
UUID                                      Usage …
------------------------------------------------------------------
b5769938-e09f-47cb-bdb6-25b15b557e84  ceph client.cinder …
```

Step 4 Get the keyring from the libvert secret:

```
novalibvirt_4612 [root@controller_server-1 /]# virsh secret-get-value
b5769938-e09f-47cb-bdb6-25b15b557e84
AQBApY1XQCBBEBAAroXvmiwmlSMEyEoXKl/sQA==
```

Step 5 Navigate to the Ceph Monitor container:

```
[root@controller_server-1 ~]# cephmon
```

Step 6 Display the Ceph Monitor keyring content:

```
cephmon_4612 [ceph@controller_server-1 ceph]$ cat /etc/ceph/ceph.client.cinder.keyring
[client.cinder]

key = AQBApY1XQCBBEBAAroXvmiwmlSMEyEoXKl/sQA==
```

Verify the keyring matches the Nova libvert keyring displayed in Step 3. Notice that in the above example the Cinder keyring is checked even though this procedure is for the Nova libvert keyring. This occurs because the Nova services need access to the Cinder volumes and so authentication to Ceph uses the Cinder keyring.

What to do next

Complete a final check to ensure that Ceph and Nova are connected by attaching a Nova volume using Horizon or the Nova CLI. After you attach the Nova volume, check the libvert domain.

Verifying Nova Instance ID

From the management node, complete the following steps to verify the Nova instance ID:

Step 1 Load the OpenStack authentication variables:

```
[root@management-server-cisco installer]# source ~/openstack-configs/openrc
```

Step 2 List the Nova instances:

```
[root@management-server-cisco images]# nova list
+--------------------------------------+------------+--------+--------
| ID                                   | Name       | Status | Task
+--------------------------------------+------------+--------+--------
| 77ea3918-793b-4fa7-9961-10fbdc15c6e5 | cisco-vm   | ACTIVE | -
+----------------------------------+-----------+-------+-
```

Step 3 Show the Nova instance ID for one of the instances:

```
[root@management-server-cisco images]# nova show
77ea3918-793b-4fa7-9961-10fbdc15c6e5 | grep instance_name
| OS-EXT-SRV-ATTR:instance_name        | instance-00000003
```

The Nova instance ID in this example is instance-00000003. This ID will be used later with the virsh command. Nova instance IDs are actually the libvirt IDs of the libvirt domain associated with the Nova instance.

Step 4 Identify the compute node where the VM was deployed:

```
[root@management-server-cisco images]# nova show 77ea3918-793b-4fa7-9961-10fbdc15c6e5 | grep
hypervisor
| OS-EXT-SRV-ATTR:hypervisor_hostname  | compute_server-1
```

The compute node in this case is compute_server-1. You will connect to this compute node to call the virsh commands. Next, you get the volume ID from the libvirt domain in the Nova libvirt container.

Step 5 Launch a SSH session to the identified compute node, compute_server-1:

```
[root@management-server-cisco ~]# ssh root@compute_server-1
```

Step 6 Navigate to the Nova libvirt container:

```
[root@compute_server-1 ~]# libvirt
```

Step 7 Get the instance libvirt domain volume ID:

```
novalibvirt_4612 [root@compute_server-1 /]# virsh dumpxml instance-00000003 | grep rbd
 <source protocol='rbd' name='volumes/volume-dd188a5d-f822-4769-8a57-c16694841a23'>
```

Step 8 Launch a SSH session to a controller node:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 9 Navigate to the Ceph Monitor container:

```
[root@compute_server-1 ~]# cephmon
```

Step 10 Verify volume ID matches the ID in Step 7:

```
cephmon_4612 [ceph@controller_server-1 ceph]
$ rbd list volumes | grep volume-dd188a5d-f822-4769-8a57-c16694841a23
volume-dd188a5d-f822-4769-8a57-c16694841a23
```

Displaying Docker Disk Space Usage

Docker supports multiple storage back ends such as Device Mapper, thin pool, overlay, and AUFS. Cisco VIM uses the devicemapper storage driver because it provides strong performance and thin provisioning. Device Mapper is a kernel-based framework that supports advanced volume management capability. Complete the following steps to display the disk space used by Docker containers.

Step 1 Launch a SSH session to a controller or compute node, for example:

```
[root@management-server-cisco ~]# ssh root@controller_server-1
```

Step 2 Enter the docker info command to display the disk space used by Docker containers:

```
[root@controller_server_1 ~]# docker info
Containers: 24
Images: 186
Storage Driver: devicemapper
 Pool Name: vg_var-docker--pool
 Pool Blocksize: 524.3 kB
 Backing Filesystem: xfs
 Data file:
 Metadata file:
 Data Space Used: 17.51 GB
 Data Space Total: 274.9 GB
 Data Space Available: 257.4 GB…
```

Reconfiguring SwiftStack Integration

Cisco VIM provides integration with SwiftStack, an object storage solution. The key aspect of the SwiftStack integration is to add a SwfitStack endpoint to an existing pod running on Cisco VIM through the reconfigure option. In this case the SwiftStack is installed and managed outside the Cisco VIM ahead of time, and the

VIM orchestrator adds the relevant Keystone configuration details to access the SwiftStack endpoint (see the Cisco VIM install guide for more details of SwiftStack).

The following options support the SwiftStack reconfiguration:

- Enable SwiftStack integration if it is not present.
- Reconfigure the existing SwiftStack PAC endpoint to point to a different cluster (cluster_api_endpoint).
- Reconfigure the Reseller_prefix of the existing SwiftStack installation.
- Reconfigure the admin password (admin_password) of an existing SwiftStack Install.

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 needs to be configured to enable HTTPS connections for the SwiftStack APIs with termination at the proxy servers.

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 as configured for Keysone Auth,AuthToken support in
Swiftstack 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
```

Note The operator should pay attention while updating the settings to ensure that SwiftStack over TLS are appropriately pre-configured in the customer-managed SwiftStack controller as specified in the Install guide.

To initiate the integration, copy the `setupdata` into a local directory by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
```

Update the `setupdata` by running the following command:

```
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include SwiftStack info)
```

Run the reconfiguration command as follows:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

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 is to authenticate against SwiftStack during volume backups leverages. The same keystone SwiftStack endpoint is configured to manage objects. The default SwiftStack container that manages cinder volumes within the account (Keystone Tenant as specified by admin_tenant) is currently defaulted to volumebackups.

Reconfiguring Administrator Source Networks

To access the administrator services, Cisco VIM provides source IP based filtering of network requests on the management node. These services include SSH and Kibana dashboard access. When the services are configured all admin network requests made to the management node are dropped, except the white listed addresses in the configuration.

Reconfiguring administrator source network supports the following options:

- Set administrator source network list: Network addresses can be added or deleted from the configuration; the list is replaced in whole during a reconfigure operation.
- Remove administrator source network list: If the **admin_source_networks** option is removed, then the source address does not filter the incoming admin service requests.

The following section needs to be configured in the Setup_data.yaml file:

```
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 The operator has to be careful while updating the source networks. If the list is misconfigured, operators may lock themselves out of access to the management node through SSH. If it is locked, an operator must log into the management node through the console port to repair the configuration.

To initiate the integration, copy the `setupdata` into a local directory by running the following command:

```
[root@mgmt1 ~]# cd /root/
[root@mgmt1 ~]# mkdir MyDir
[root@mgmt1 ~]# cd MyDir
[root@mgmt1 ~]# cp /root/openstack-configs/setup_data.yaml <my_setup_data.yaml>
```

Update the `setupdata` by running the following command:

```
[root@mgmt1 ~]# vi my_setup_data.yaml (update the setup_data to include SwiftStack info)
```

Run the reconfiguration command as follows:

```
[root@mgmt1 ~]# cd ~/installer-xxxx
[root@mgmt1 ~]# ciscovim --setupfile ~/MyDir/<my_setup_data.yaml> reconfigure
```

Password Reset for Cisco VIM Management Node

Run the following command to reset the Root Password of Cisco VIM management node **RHEL-7 / systemd** .

1. Boot your system and wait until the **GRUB2** menu appears.
2. In the **boot loader** menu, highlight any entry and press **e**.
3. Find the line beginning with linux. At the end of this line, append the following:

   ```
    init=/bin/sh
   ```

   Or if you face any alarm, instead of **ro** change **rw** to **sysroot** as shown in the following example:

   ```
   rw init=/sysroot/bin/sh
   ```

4. Press **Ctrl+X** to boot the system using the options you edited.

   Once the system boots, you can see the shell prompt without having to enter any user name or password:

   ```
   sh-4.2#
   ```

5. Load the installed SELinux policy by running the following command:

   ```
   sh-4.2# /usr/sbin/load_policy -i
   ```

6. Execute the following command to remount your root partition:

   ```
   sh4.2#
       mount -o remount,rw /
   ```

7. Reset the root password by running the following command:

   ```
   sh4.2# passwd root
   ```

   When prompted, enter your new root password and click **Enter** key to confirm. Enter the password for the second time to make sure you typed it correctly and confirm with **Enter** again. If both the passwords match, a confirmation message appears.

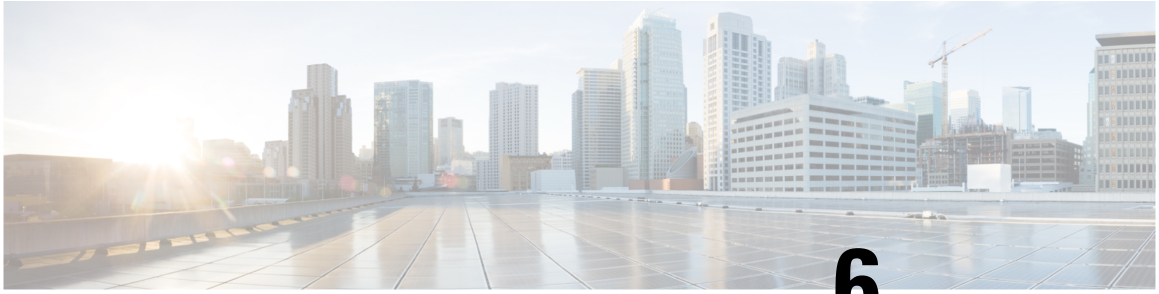
8. Execute the following command to remount the root partition again, this time as read-only:

   ```
   sh4.2#
       mount -o remount,ro /
   ```

9. Reboot the system. Now you can log in as the root user using the new password set up during this procedure.

   To reboot the system, enter **`exit`** and **`exit`** again to leave the environment and reboot the system.

References: https://access.redhat.com/solutions/918283.



CHAPTER 6

Overview to Cisco VIM Unified Management

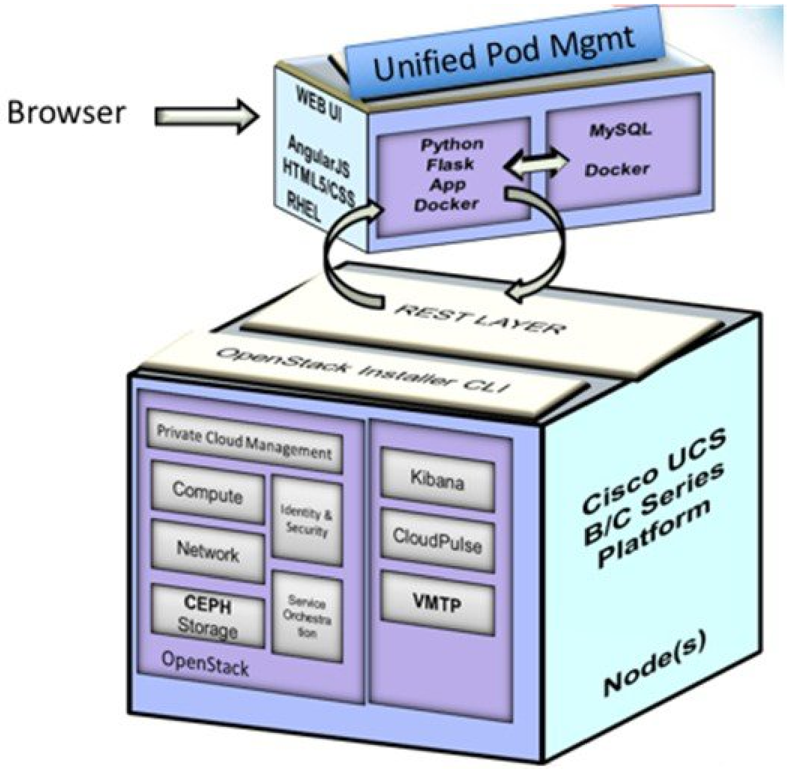
Cisco VIM Insight is an optional application, which acts as a single point of management for the Cisco VIM. Inclusive of your Cisco NFVI package, you can use Cisco VIM Insight to manage Cisco NFVI for day-0 and day-n and for multi-site and multi-pod management features.

Cisco VIM Unified Management Overview

Cisco VIM provides an Intuitive and easy way to deploy and manage the NFVI platform, reducing user-error and providing visualization deployment to manage multiple Cisco VIM Pods from a single portal. In Cisco VIM 2.2 and higher releases, a light-weight UI which is a dockerized application, supports multi-tenancy with local RBAC support and CiscoVIM Rest layer are integrated. The container-based UI platform manages multiple CiscoVIM pods from day-0, or above in the lifecycle of the cloud.

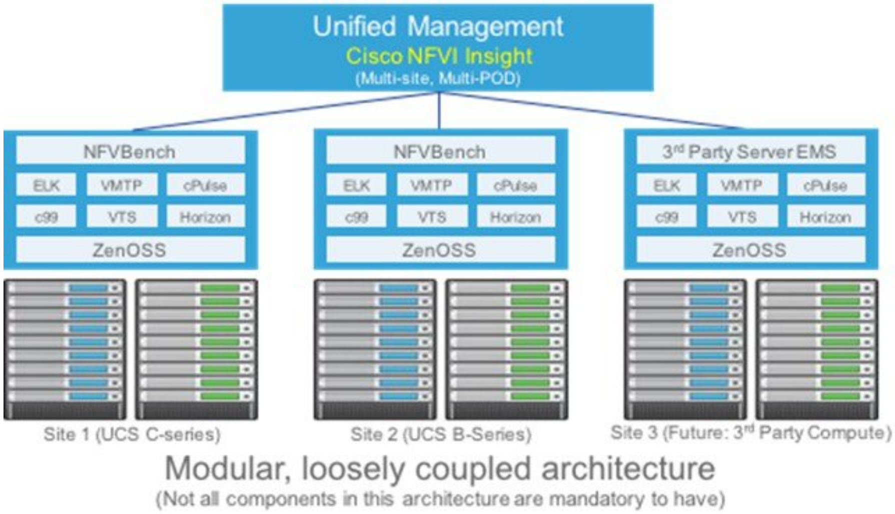
The following figure shows the architecture of the CiscoVIM UM's interaction with a Pod:

Figure 19: Cisco VIM UM's Interaction with a Pod



The architecture of the CiscoVIM UM is light-weight, hierarchical, and scalable. Each local site is autonomous with localized toolsets. Global Unified Management UI, provides ease of management with multisite and multi-pod capability for distributed NFV deployment at scale. This facility can be used through browsers such as IE, Firefox, Safari, and Chrome. Cisco VIM UM by itself, is designed to operate in HA. The platform is a modular, loosely coupled architecture, that provides the capability to manage multiple pods, with RBAC support as depicted in the following figure:

Figure 20: Cisco VIM UM Architecture



Cisco VIM UM can be installed in Standalone or non-HA mode: You can Install in a Standalone or non-HA mode (on the management node of the pod) or a standalone (BOM same as the management node) server. Migrating from one install mode to another can be done effectively as the UI interacts with each Pod through REST API and little RBAC information of the Admin and user is kept in the DB.

The UI has two types of views:

- UI Admin: UI Admin can add users as UI Admin or Pod Admin.
- Pod Admin: Pod Admin has the privilege only at the Pod level, unless Pod Admin is also a UI Admin.

Cisco VIM Unified Management Admin UI Overview

Admin UI is responsible for managing the UI and Pod admin, which includes adding and revoking user privileges. Also, the UI Admin can delete an existing Pod from the management pane.

Cisco VIM Unified Management Pod UI Overview

The Pod UI, is responsible for managing each Pod. VIM UM gives easy access to switch between multiple Pods. Through the Pod UI, a Pod Admin can manage users and their respective roles and responsibilities. Also, the Pod UI provides the user to execute day-0 (install) and day-n (Pod management, software update, and so on.) activities seamlessly. ELK, Horizon Web UI, and so on, are also cross-launched and visible for each Pod through the Pod UI.