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 section is not applicable, if you have installed the optional Cisco Virtual Topology System. For information about use of passwords when VTS is installed, see *Installing Cisco VTS* section in the *Cisco NFV Infrastructure 2.4 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 the password reconfiguration in a maintenance window as the container restart might disrupt the control plane.

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

```
[root@mgmt1 ~]# 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
IRONIC_DEBUG_LOGGING	True
OPFLEX_DEBUG_LOGGING	True
GNOCCHI_VERBOSE_LOGGING	True
AIM_DEBUG_LOGGING	True
CINDER_DEBUG_LOGGING	False
KEYSTONE_DEBUG_LOGGING	False
log_rotation_size	100M
CLOUDPULSE_VERBOSE_LOGGING	True
MAGNUM_VERBOSE_LOGGING	True
NOVA_DEBUG_LOGGING	True
NEUTRON_VERBOSE_LOGGING	True
external_lb_vip_cert	/root/openstack-configs/haproxy.pem
GLANCE_VERBOSE_LOGGING	True
elk_rotation_frequency	weekly
CEILOMETER_VERBOSE_LOGGING	True
NOVA_CPU_ALLOCATION_RATIO	16.0
CLOUDPULSE_DEBUG_LOGGING	False
log_rotation_frequency	weekly
HEAT_DEBUG_LOGGING	False
KEYSTONE_VERBOSE_LOGGING	True
external_lb_vip_cacert	/root/openstack-configs/haproxy-ca.crt
GNOCCHI_DEBUG_LOGGING	False
MAGNUM_DEBUG_LOGGING	True
log_rotation_del_older	8
CINDER_VERBOSE_LOGGING	True
elk_rotation_size	2
IRONIC_VERBOSE_LOGGING	True
elk_rotation_del_older	8
NEUTRON_DEBUG_LOGGING	True
HEAT_VERBOSE_LOGGING	True

```
                                                          |
| CEILOMETER_DEBUG_LOGGING   | False
                                                          |
| ES_SNAPSHOT_AUTODELETE     | {u'threshold_warning': 60, u'enabled': True, u'period':
u'hourly', u'threshold_high': 80, u'threshold_low': 50} |
| GLANCE_DEBUG_LOGGING       | False
                                                          |
| NOVA_VERBOSE_LOGGING       | True
                                                          |
| NOVA_RAM_ALLOCATION_RATIO  | 1.5
                                                          |
+--------------------+------------------------------------------------------------------------------------+

[root@mgmt1 installer-xxxx]#
[root@mgmt1 installer-xxxx]# ciscovim list-password-keys
+----------------------------------+
| Password Keys                    |
+----------------------------------+
| ADMIN_USER_PASSWORD              |
| CINDER_DB_PASSWORD               |
| CINDER_KEYSTONE_PASSWORD         |
| CLOUDPULSE_KEYSTONE_PASSWORD     |
| COBBLER_PASSWORD                 |
| CPULSE_DB_PASSWORD               |
| CVIM_MON_PASSWORD                |
| CVIM_MON_READ_ONLY_PASSWORD      |
| CVIM_MON_SERVER_PASSWORD         |
| DB_ROOT_PASSWORD                 |
| ETCD_ROOT_PASSWORD               |
| GLANCE_DB_PASSWORD               |
| GLANCE_KEYSTONE_PASSWORD         |
| HAPROXY_PASSWORD                 |
| HEAT_DB_PASSWORD                 |
| HEAT_KEYSTONE_PASSWORD           |
| HEAT_STACK_DOMAIN_ADMIN_PASSWORD |
| HORIZON_SECRET_KEY               |
| KEYSTONE_DB_PASSWORD             |
| KIBANA_PASSWORD                  |
| METADATA_PROXY_SHARED_SECRET     |
| NEUTRON_DB_PASSWORD              |
| NEUTRON_KEYSTONE_PASSWORD        |
| NOVA_DB_PASSWORD                 |
| NOVA_KEYSTONE_PASSWORD           |
| RABBITMQ_ERLANG_COOKIE           |
| RABBITMQ_PASSWORD                |
| VOLUME_ENCRYPTION_KEY            |
| 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 that you do not enter typos in the secrets.yaml or openstack_config.yaml files.

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

Note

- For pod operations, OpenStack uses the service accounts such as admin, cinder, glance, heat, heat_domain_admin, neutron, nova, placement, and cloudpulse. These accounts use passwords to authenticate each other for standard operations. You must not change the password used by these accounts, other than using the ciscovim reconfigure operation. To enforce this behavior, starting Cisco VIM 2.4.5, the "change password" panel is disabled on the Horizon dashboard for these accounts.
- You should create personal OpenStack user accounts for those who need OpenStack admin or member access. You can change the passwords for these accounts through the Horizon dashboard, OpenStack CLI, or OpenStack client interface.

Reconfiguring Glance Client Key for Central Ceph Cluster

From release Cisco VIM 3.0.0, the installation of a central ceph cluster is automated to serve the images to edge pods through Glance service. No local ceph cluster for edge pods as they have constraints on power and space. The edge pods do not need any persistent storage and provide services via a central ceph cluster for glance. For the edge pods to communicate with central ceph cluster using a cluster id, a GLANCE_CLIENT_KEY is required.

Follow the below steps to reset the GLANCE_CLIENT_KEY:

Step 1 Regenerate the client keyring for glance.

a) On the central ceph cluster, ssh to the management node and execute the following:

```
# ciscovim reconfigure --regenerate_ceph_keyring -y
```

Alternatively, you can regenerate the key via the corresponding RestAPI call:

```
# curl -u <user>:<password> -X POST https://<ip|host>:8445/v1/misc --header "Content-Type:
application/json" -d '{"action": {"reconfigure": "true", "regenerate_ceph_keyring": true}}'
```

Step 2 Retrieve the generated client keyring for glance. From the management node, execute the following command to get the cluster UUID:

```
# cat /root/openstack-configs/ceph/fetch/ceph_cluster_uuid.conf
<cluster_uuid>
# cat /root/openstack-configs/ceph/fetch/<cluster_uuid>/etc/ceph/ceph.client.glance.keyring
[client.glance]
key = <GLANCE_CLIENT_KEY>
caps mon = "allow r"
caps osd = "allow class-read object_prefix rbd_children, allow rwx pool=images"
```

Step 3 Reconfigure the edge pod to use the new keyring generated on central ceph. SSH to the management node of each edge pod and execute the following:

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

Step 4 Optional, but recommended to do it on a handful of pods. Once reconfiguration is done, test if the keyring works by creating and deleting glance images.

Enabling NFVIMON Post Pod Install

You can optionally install Cisco VIM with a third-party software known as NFVIMON which is used to monitor the health and performance of the NFV infrastructure. The NFVIMON feature enables extensive monitoring and performance data for various components of the cloud infrastructure including Cisco UCS blade and rack servers, service profiles, Nexus top of rack switches, fabric connections and also the OpenStack

instances. The monitoring system is designed such that it can monitor single or multiple pods from a single management system. NFVIMON can be enabled by extending the setup_data.yaml with relevant information on an existing pod, through the reconfigure option.

NFVIMON consists of four components: ceilometer for data collection, collector, Resource Manager (RM), and control-center with Cisco Zenpacks (CC). As NVIFMON is a third party software, care has been taken to make sure its integration into VIM is loosely coupled and the VIM automation only deals with installing the ceilometer service software piece needed to monitor the pod. The installing of the other NFVIMON components (collector, Resource Manager (RM) and control-center with Cisco Zenpacks (CC)), are Cisco Advance Services led activity and those steps are outside the scope of the install guide.

Before you Begin

Ensure that you have engaged with Cisco Advance Services on the planning and installation of the NFVIMON accessories along with its network requirements. Also, the image information of collector, Resource Manager (RM) and control-center with Cisco Zenpacks (CC)) is available only through Cisco Advance Services. At a high level, you can have a node designated to host a pair of collector VM for each pod, and a common node to host CC and RM VMs, which can aggregate and display monitoring information from multiple pods.

The collector VMs must have two interfaces: an interface to br_mgmt of the VIM, and another interface that is routabl and reachable to the VIM Installer REST API and the RM VMs. As the collector VM is available in an independent node, four IPs from the management network of the pod must be pre-planned and reserved. The installation steps of the collector, Resource Manager (RM) and control-center with Cisco Zenpacks (CC)) are part of Cisco Advance Services led activity.

Installation of NFVIMON

The ceilometer service is the only component in NFVIMON offering that is managed by VIM orchestrator. While the ceilometric service collects the metrics to pass openstack information of the pod to the collectors, the Cisco NFVI Zenpack available in the CC/RM node gathers the node level information. To enable NFVIMON 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
    COLLECTOR_TORCONNECTIONS: # Optional. Indicates the port where the collector is hanging
 off. Recommended when Cisco NCS 5500 is used as ToR
      - tor_info: {po: <int>, switch_a_hostname: ethx/y, switch_b_hostname: ethx/y}
  DISPATCHER:
    rabbitmq_username: admin  # Pod specific user for dispatcher module.
```

```
  NFVIMON_ADMIN: admin_name  # Optional, once enabled, need to have only 1 admin
reconfigurable to add/update user id
```

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 directory and update it manually with information listed above, and then run the reconfiguration command as follows:

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

To initiate the uninstallation of NFVIMON on an existing pod, copy the setupdata into a local directory and remove the entire NFVIMON section listed above, and then run the reconfiguration command as follows:

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

Note

- NFVIMON is supported only on a pod running with Keystone v3 (Default for release Cisco VIM 3.2.0).
- NFVIMON can run with either root or non-root admin keys for monitoring.

Enabling CVIMMON Post Pod Installation

CVIMMON, an extensive monitoring solution, is designed to monitor a single pod from a single management system. Cisco VIM can be optionally installed with CVIMMON, to monitor the health and performance of the NFV infrastructure. CVIMMON is enabled by extending the setup_data.yaml with relevant information on an existing pod, using the reconfigure option.

Post installation of Cisco VIM 2.4.3 or later versions, you can enable CVIMMON and CVIM-TRAP (SNMP, SERVER_MON) using the reconfigure option.

Note CVIM-TRAP can be enabled, only if CVIMMON exists. Once the CVIMMON or CVIM-TRAP is enabled, it cannot be disabled again.

To enable the CVIMMON and SNMP-Trap features or to change the individual parameters in CVIMMON, SNMP, or SERVER_MON:

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

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

2. Edit the setup data.
3. Save the file and execute the below command. For sample configuration, see *Enabling CVIMMON on Cisco VIM* section of *Cisco Virtualized Infrastructure Manager Installation Guide*

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

Note Migration from SNMPv2 to SNMPv3 is only supported, but not vice-versa.

Reconfiguring CIMC/BMC Password on Existing Installation

Cisco VIM allows you to reconfigure the CIMC/BMC password on an existing installation along with OpenStack services.

Note You must have a C-series or Quanta-based pod up and running with Cisco to reconfigure the CIMC/BMC 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 ~1]# ciscovim reconfigure --cimc_password --setupfile /root/MyDir/<my_setup_data.yaml>
```

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 reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Note While using auto-ToR via ACI APIs without the APIC plugin, ensure that you update the `vim_apic_networks` section with the right VLAN information as part of the reconfiguration option.

Fernet Key Operations

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

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

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

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

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

To forcefully rotate the fernet keys:

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

To resync the fernet keys across the keystone nodes:

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

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
```

Verifying TLS Certificates

Cisco VIM provides a tool to check the expiration date of the installed certificates. If a certificate is expired, you may not be able to access the HTTPS endpoints. Checks are run daily and a syslog message is created if a certificate is nearing expiration.

In addition, a tool is provided to check certificate expiration on demand.

The tool's command line support can be shown as follows:

```
# cd ~/installer-xxxx/tools
# python certificate-check.py -help
```

To check all certificates, run the following commands:

```
#cd ~/installer-xxxx/tools
# python certificate-check.py
```

To check a single certificate, run the following commands:

```
cd ~/installer-xxxx/tools
# python certificate-check.py –s openstack
```

LDAP/AD support with Keystone v3

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

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

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

user_filter = (memberOf=CN=os-users,OU=OS-Groups,DC=mercury,DC=local)
user_id_attribute = sAMAccountName
user_name_attribute = sAMAccountName
user_mail_attribute = mail             # Optional
group_tree_dn = ou=OS-Groups,dc=mercury,dc=local
group_name_attribute = sAMAccountName
group_filter: '(&(objectClass=group)(|(cn=server-ops)(cn=admins)))'  # Optional
group_member_attribute: memberUid # Optional
group_id_attribute: gidNumber    # Optional
group_members_are_ids: True      # Optional
chase_referrals: <True or False> # Optional
```

Condition for LDAP user and password parameters are as follows:

- 1 – Can be optional
- 2 – Should be mutually inclusive
- 3 – If defined, it cannot be empty

To initiate the integration of LDAP with Keystone v3 on an existing pod, copy the setupdata into a local directory, update it manually with the relevant LDAP configuration, and then run the following reconfiguration commands:

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

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

The reconfigure feature supports a full or partial reconfiguration of the LDAP integration service.

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 reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

Enabling Cinder Volume Encryption in Cisco VIM

Cisco VIM supports the configuration and creation of encrypted volumes managed by Cinder. The encryption is done natively using Linux Unified Key Setup (LUKS). Administrators can use the standard OpenStack APIs to create and mount the volumes. From release Cisco VIM 3.0.0, the Cinder volume encryption is supported by default. No configuration parameters are available in the setup data.

The following are the steps to create an encrypted volume:

Step 1 From the management node, load the OpenStack authentication variables:

```
[root@management-server-cisco ~]# source ~/openstack-configs/openrc
```

Step 2 Create a volume type that defines the desired encryption parameters using the below command:

```
[root@management-server-cisco images]# openstack volume type create \
       --encryption-provider nova.volume.encryptors.luks.LuksEncryptor \
       --encryption-cipher aes-xts-plain64 \
       --encryption-key-size 256 \
       --encryption-control-location front-end LUKS
```

Step 3 Create an encrypted volume using the following command:

```
[root@management-server-cisco images]# openstack volume create --size 1 --type LUKS encrypted_volume
```

Replacing ACI Controller in Cisco VIM

The Opflex ML2 plugin (in Unified mode) integrated with Cisco VIM manages the tenant VLANs dynamically, as VMs come and go in the cloud. In addition, we support an administrator driven automated workflow to provision the provider networks. This feature is supported on a C-series based Fullon or Micropod running with Cisco VIC 1227 and Intel NIC x710 with redundancy at NIC level. While the integration of ACI into Cisco VIM is a day-0 activity, Cisco 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, 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 reconfigure --setupfile ~/MyDir/<my_setup_data.yaml>
```

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 1: Management Nodes

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
br_api	incoming	HTTPS	TCP	8445	RestAPI	
br_api	incoming	HTTPS	TCP	8008	RestAPI logs	
br_api	incoming	HTTPS	TCP	9000	Unified Management UI	
br_api	incoming	HTTPS	TCP	5601	Kibana	
br_api	incoming	SSH	TCP	22	SSH	
br_api	incoming	HTTPS	TCP	3000	Grafana	
br_api	outgoing	NTP	UDP	123	NTP	
br_api	outgoing	DNS	UDP	53	DNS	
br_api	outgoing	Syslog	UDP	514	Syslog	User configurable. Default value is 514.

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
br_mgmt	incoming	HTTP	TCP	7081	Fluentd-aggr	From all nodes to mgmt node
br_api	outgoing	HTTP	TCP	9090	Prometheus	
br_api	outgoing	HTTP	TCP	9093	Alertmanager	
localhost	incoming/outgoing	HTTP	TCP	1162	SNMPCVIM_MON	Internal communication between processes
br_api	outgoing	SNMP	UDP	162	SNMP	Userdefined. Default value is 162
br_api	incoming	HTTP	TCP	22	SERVER_MON	From CIMC of the UCS servers.
br_api	incoming	Syslog	UDP	5140	SERVER_MON + Syslog	From CIMC of the UCS servers to the management node
br_api	outgoing	LDAP	TCP	389	LDAP	Default: 389 or defined in setup_data
br_api	outgoing	LDAPS	TCP	636	LDAPS	Default: 636 or defined in setup_data

Table 2: Controller Nodes

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
external_lb_vip	incoming	HTTP	TCP	80	Redirects to 443	
external_lb_vip	incoming	HTTPS	TCP	443	Horizon	
external_lb_vip	incoming	HTTPS	TCP	8774	Nova	
external_lb_vip	incoming	HTTPS	TCP	6080	Nova NoVNC Proxy	
external_lb_vip	incoming	HTTPS	TCP	9696	Neutron	
external_lb_vip	incoming	HTTPS	TCP	8776	Cinder	

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
external_lb_vip	incoming	HTTPS	TCP	9292	Galance	
external_lb_vip	incoming	HTTPS	TCP	8000	Heat	
external_lb_vip	incoming	HTTPS	TCP	8004	Heat	
external_lb_vip	incoming	HTTPS	TCP	9999	Cloudpulse	
external_lb_vip	incoming	HTTPS	TCP	8777	Ceilometer	
external_lb_vip	incoming	HTTPS	TCP	8778	Placement	
external_lb_vip	incoming	HTTPS	TCP	5000	Keystone	
br_mgmt	incoming	HTTP	TCP	156272	RabbitMQ monitoring	From management node only
br_mgmt	outgoing	LDAP	TCP	389	LDAP	Default: 389 or defined in setup_data
br_mgmt	outgoing	LDAPS	TCP	636	LDAPS	Default: 636 or defined in setup_data
br_mgmt	outgoing	HTTP	TCP	7081	Fluentd	To management node

Table 3: Cisco VIM Software Hub Server Node

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
br_public	outgoing	NTP	UDP	123	NTP	
br_public	incoming	HTTPS	TCP	443	HTTPD	Browsing artifacts on a web browser
br_private	incoming	HTTPS	TCP	8441	HTTPD	Reverse proxy for docker registry
br_public	incoming	SSH	TCP	22	SSH	

Table 4: Unified Management Node

Interface	Direction	Protocol	UDP/TCP	Port	Application	Note
br_api	outgoing	HTTPS	TCP	8445	Unified Management	Connect to Cisco VIM management node RestAPI
br_api	incoming	HTTPS	TCP	9000	HTTPD	UI
br_api	incoming	SSH	TCP	22	SSH	

Cisco VIM Monitor Alerting Rules Customization

Cisco VIM monitor is deployed with a set of built-in alerting rules that cover the most important error conditions that can occur in the pod. You can view the alerts from the Grafana user interface or Alerting dashboard or send them optionally to a number of supported receivers.

After deployment, the pod administrators can customize the alerting rules based on their requirements.

Alerting Rules Customization

The alerting rules define how alerts should be triggered based on conditional expressions on any available metric. For example, you can trigger an alert when any performance metric such as CPU usage, network throughput or disk usage reaches certain threshold.

You can add new alerting rules and modify or delete the pre-built existing alerting rules by following the below steps:

1. Create a proper custom alerting rules configuration file:
    1. Create a custom alerting rule configuration file named `alerting_custom_rules.yml` under the management node `openstack-configs` directory.
    2. Add the new rules, modified rules and deleted rules in that file using your favorite editor (see the file format below)
    3. Verify that the custom alerting rule file is valid using a provided tool.
2. Once validated, if needed, you can rename it and issue a standard reconfiguration using the ciscovim cli..

```
[root@mgmt1 ~]# ciscovim reconfigure --alerting_rules_config <alerting_rules_config_file>
```

Custom Alerting Rule File Format

The `alerting_custom_rules.yml` file must follow the format defined in this section. This format is identical to the one used by the Prometheus configuration file, with a few additional semantic extensions to support deletion and modification of pre-built existing rules.

General Format

The group entry contains a list of groups identified by (group_name), where each group can include one or more rules. Use the labels to determine the severity and other snmp trap attributes.

Following are the limitations to set labels:

- `severity`, `snmp_fault_code`, and `snmp_fault_severity` must be set to one of the values specified in the example below.
- `snmp_fault_source` should indicate the metric used in the alert expression
- `snmp_node` must not be changed.
- `snmp_podid` must be same as the pod name specified in`setup_data.yaml`

```
groups:
- name: {group_name}
  rules:
  - alert: {alert_name}
    annotations:
      description: {alert_description}
      summary: {alert_summary}
    expr: {alert_expression}
    for: {pending_time}
    labels:
      severity: {informational/warning/critical}
      snmp_fault_code:
{other/resourceUsage/resourceThreshold/serviceFailure/hardwareFailure/networkConnectivity}
      snmp_fault_severity: {emergency/critical/major/alert/informational}
      snmp_fault_source: {fault_source}
      snmp_node: '{{ $labels.instance }}'
      snmp_podid: {pod_id}
```

Addition of Alerting Rules

Any alert rule specified under a group other than **change-rules** group or **delete-rules** group is populated to the merged output file. You can prioritize the custom rules over the pre-existing rules if there are two alerts with the same name in both the files, such that only the one from custom file is kept as a result of the merge.

Modification of Alerting Rules

You can modify any pre-existing rule using the following syntax:

```
groups:
- name: change-rules
  rules:
  - alert: {alert_name}
    expr: {new_alert_expression}
    annotations:
      summary: {new_alert_summary}
```

The above merge script finds only the group named **change-rules** and modifies the expression and/or summary of the corresponding alert.

If the alert to be changed does not exist, it will not be created and no changes are performed.

Deletion of Alerting Rule

You can delete any built-in rule using the following construct:

```
groups:
- name: delete-rules
```

```
  rules:
  - alert: {alert_name/regular_expression}
```

The above merge script finds only the group named **delete-rules** and deletes the pre-existing rules that match the alert name or regular expressions.

If the alert to be deleted does not exist, no changes are performed.

Example

The following custom configuration file includes examples of new alerting rule, modified alerting rule and deleted alerting rules:

```
groups:
- name: cpu
  rules:
  - alert: cpu_idle
    annotations:
      description: CPU idle usage is too high - resources under-utilized
      summary: CPU idle too high
    expr: cpu_usage_idle > 80
    for: 5m
    labels:
      severity: informational
      snmp_fault_code: resourceUsage
      snmp_fault_severity: informational
      snmp_fault_source: cpu_usage_idle
      snmp_node: '{{ $labels.instance }}'
      snmp_podid: pod7
  - alert: cpu_iowait
    annotations:
      description: CPU iowait usage is too high
      summary: CPU iowait too high
    expr: cpu_usage_iowait > 10
    for: 3m
    labels:
      severity: warning
      snmp_fault_code: resourceUsage
      snmp_fault_severity: alert
      snmp_fault_source: cpu_usage_iowait
      snmp_node: '{{ $labels.instance }}'
      snmp_podid: pod7
- name: change-rules
  rules:
  - alert: disk_used_percent
    expr: disk_used_percent > 99
    annotations:
      summary: Disk used > 99%
  - alert: reboot
    annotations:
      summary: Server rebooted
  - alert: system_n_users
    expr: system_n_users > 10
- name: delete-rules
  rules:
  - alert: disk_filling_up_in_4h
  - alert: mem.*
```

Validation Script

Validate any custom configuration file prior to reconfiguration, by executing the following CLI command from any location on the management node:

```
check_alerting_rules (no additional parameters are required)
```

The validation script uses the prometheus "promtool", but skips some of its checks to allow the modification and deletion of rules. It also checks if the provided SNMP severities and fault codes are supported. When no custom file is present, the expected location is mentioned in the output.

Ouput of validation script in case of success

```
# check_alerting_rules
check_promtool.py: checking /prometheus/alerting_custom_rules.yml
check_promtool.py: success:
check_promtool.py:  regular expressions for rules to be deleted: 2
check_promtool.py:  rules to be changed: 3
check_promtool.py:  rules to be added: 2
```

Output of validation script in case of failure

```
# check_alerting_rules
check_promtool.py: checking custom-rules.yml
check_promtool.py: failure:
check_promtool.py:    group "new_group", rule 0, "new_alert": could not parse expression:
parse error at char 8:
                       could not parse remaining input "@$"...
check_promtool.py:   group "new_group2", rule 0, "new_alert_3": could not parse expression:
 parse error at char 7:
                       bad number or duration syntax: "1"
# check_alerting_rules
check_promtool.py: checking /prometheus/alerting_custom_rules.yml
check_promtool.py: failure:
check_promtool.py:    line 36: field custom_field not found in type rulefmt.Rule
```

Alert Manager and Receiver Customization

The Alert Manager component in Cisco VIM Monitor is in charge of routing, grouping, and inhibiting alerts that are sent by the Prometheus alert rules engine to the appropriate receivers.

The default configuration in Cisco VIM Monitor allows every alert to be forwarded as SNMP traps to the SNMP managers through SNMP agent if enabled in the Cisco VIM configuration file.

After deployment, you can add custom alert routes, alert grouping, alert inhibitions and receivers by following the below steps:

1. Create a proper custom alerting rules configuration file:
    1. Create a custom alert manager rule configuration file named `alertmanager_custom_config.yml`.
    2. Edit the content using your favorite editor (see format below).
    3. Copy that file to the management node `openstack-configs` directory
    4. Verify that the custom alerting rule file is valid using a provided tool.

2. Once the file is validated, if needed, you can either leave it in `openstack-configs` directory or move it to your preferred location. Then use a **ciscovim reconfigure** command with an additional parameter:

```
[root@mgmt1 ~]# ciscovim reconfigure --alertmanager_config <alertmanager_config_file>
```

Supported Receivers

The Alert Manager supports the following list of receivers:

- webhook
- pagerduty
- e-mail
- pushover
- wechat
- opsgenie
- victorops

Alert Manager Custom Configuration File Format

General Format

The following listing shows the general format of the alert manager configuration file. Most custom configuration files should only include a small subset of the available options.

```
global:
# ResolveTimeout is the time after which an alert is declared resolved # if it has not been
 updated.
[ resolve_timeout: <duration> | default = 5m ]

# The default SMTP From header field. [ smtp_from: <tmpl_string> ]
# The default SMTP smarthost used for sending emails, including port number.
# Port number usually is 25, or 587 for SMTP over TLS (sometimes referred to as STARTTLS).

# Example: smtp.example.org:587 [ smtp_smarthost: <string> ]
# The default hostname to identify to the SMTP server. [ smtp_hello: <string> | default =
"localhost" ]
[ smtp_auth_username: <string> ]
# SMTP Auth using LOGIN and PLAIN. [ smtp_auth_password: <secret> ]
# SMTP Auth using PLAIN.
[ smtp_auth_identity: <string> ] # SMTP Auth using CRAM-MD5.
[ smtp_auth_secret: <secret> ]
# The default SMTP TLS requirement.
[ smtp_require_tls: <bool> | default = true ]

# The API URL to use for Slack notifications. [ slack_api_url: <secret> ]
[ victorops_api_key: <secret> ]
[ victorops_api_url: <string> | default =
"https://alert.victorops.com/integrations/generic/20131114/alert/" ]
[ pagerduty_url: <string> | default = "https://events.pagerduty.com/v2/enqueue" ] [
opsgenie_api_key: <secret> ]
[ opsgenie_api_url: <string> | default = "https://api.opsgenie.com/" ] [ hipchat_api_url:
<string> | default = "https://api.hipchat.com/" ]  [ hipchat_auth_token: <secret> ]
[ wechat_api_url: <string> | default = "https://qyapi.weixin.qq.com/cgi-bin/" ] [
wechat_api_secret: <secret> ]
[ wechat_api_corp_id: <string> ]
```

```
# The default HTTP client configuration [ http_config: <http_config> ]

  # Files from which custom notification template definitions are read.



# The last component may use a wildcard matcher, e.g. 'templates/*.tmpl'. templates:
[ - <filepath> ... ]

# The root node of the routing tree. route: <route>

# A list of notification receivers. receivers:
- <receiver> ...

# A list of inhibition rules. inhibit_rules:
[ - <inhibit_rule> ... ]
```

The custom configuration must be a full working config file with the following template. It should contain three main keys (global, route, receiver).

The global configuration must have at least one attribute, for example, resolve_timeout = 5m. Ensure that all new receivers must be part of the route so the alerts can be routed to the proper receivers. The receiver name cannot be snmp.

You can find the configuration details for creating route/receiver in the Prometheus Alert Manager documentation (publicly available online).

```
global: resolve_timeout: 5m

route: <route>

receivers:
- <receiver> ...

The following is a custom config to add a webhook receiver.

global:
  resolve_timeout: 5m

route:
  group_by: ['alertname', 'cluster', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 8737h
  receiver: receiver-webhook

receivers:
- name: 'receiver-webhook'
  webhook_configs:
  - send_resolved: true
    url: 'http://webhook-example:####/xxxx/xxx'
```

Default Built-in Configuration File

Two different default configuration files are available to define the following in order:

1. Generic route for all alerts to the SNMP agent running on the management node.
2. Route to a generic receiver that can be customized to add a channel of notification (webhook, slack and others).

Default configuration file with SNMP enabled

```
:
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname', 'cluster', 'service']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 8737h

  # A default receiver
  receiver: snmp

receivers:
- name: 'snmp'
  webhook_configs:
  - send_resolved: true
    url: 'http://localhost:1161/alarms'
```

Default configuration file with SNMP disabled

```
route:
  receiver: recv
  group_by:
  - alertname
  - cluster
  - service
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 8737h
receivers:
- name: recv
```

SNMP Trap Receivers

You can send the SNMP traps to SNMP managers enabled in the Cisco VIM configuration file `setup_data.yaml`.

Example: inhibit (mute) alerts matching a set of labels

Inhibit alerts is a tool that prevents certain alerts to be trigged if other alert/alerts is/are trigged. If one alert having the target attribute matches with the another alert having source attribute, this tool inhibits the alert with target attribute.

This is the general format for inhibit alerts. You can set a regex to match both the source and target alerts and to filter the alerts per label name.

```
# Matchers that have to be fulfilled in the alerts to be muted.
target_match:
  [ <labelname>: <labelvalue>, ... ]
target_match_re:
  [ <labelname>: <regex>, ... ]

# Matchers for which one or more alerts have to exist for the
# inhibition to take effect.
source_match:
  [ <labelname>: <labelvalue>, ... ]
```

```
source_match_re:
  [ <labelname>: <regex>, ... ]

# Labels that must have an equal value in the source and target
# alert for the inhibition to take effect.
[ equal: '[' <labelname>, ... ']' ]
```

Example: Inhibit alerts if other alerts are active

The following is an example of inhibit rule that inhibits all the warning alerts that are already critical.

```
inhibit_rules:
- source_match:
    severity: 'critical'
  target_match:
    severity: 'warning'
  # Apply inhibition if the alertname is the same.
  equal: ['alertname', 'cluster', 'service']
```

This is an example of inhibit all alerts docker_container in containers that are down (which has the alert docker_container_down on).

```
inhibit_rules:
  - target_match_re:
       alertname: 'docker_container.+'
    source_match:
       alertname: 'docker_container_down'
    equal: ['job', 'instance']
```

Validation Script

When a new configuration is set, execute the `check_alertmanager_config` from anywhere in the management node and ensure that you get a **SUCCESS** in the output from the configuration POV.

```
> check_alertmanager_config
Checking '/var/lib/cvim_mon/alertmanager_custom_config.yml'  SUCCESS
Found:
 - global config
 - route
 - 0 inhibit rules
 - 1 receivers
 - 0 templates
```