Verifying the Cisco NFVI Installation

The following topics provide quick methods for checking and assessing the Cisco NFVI installation.

Displaying Cisco NFVI Node IP Addresses

To display the IP addresses for all Cisco NFVI nodes, enter the following command:

```
# cd /root/openstack-configs
[root @nfvi_management_node openstack-configs]# cat
/root/installer/openstack-configs/mercury_servers_info
```

Sample output is shown below:

```
Total nodes: 8
Controller nodes: 3
+---------------+---------------+------------+------------+-----------+---------+
| Server        | CIMC          | Management | Provision  | Tenant    | Storage |
+---------------+---------------+------------+------------+-----------+---------+
| c44-control-1 | 172.26.233.54 | 10.21.1.25 | 10.21.1.25 | 10.2.2.22 | None    |
|               |               |            |            |           |         |
| c44-control-3 | 172.26.233.56 | 10.21.1.27 | 10.21.1.27 | 10.2.2.24 | None    |
|               |               |            |            |           |         |
| c44-control-2 | 172.26.233.55 | 10.21.1.28 | 10.21.1.28 | 10.2.2.25 | None    |
|               |               |            |            |           |         |
+---------------+---------------+------------+------------+-----------+---------+
Compute nodes: 2
+---------------+---------------+------------+------------+-----------+---------+
|     Server    |      CIMC     | Management | Provision  |   Tenant  | Storage |
+---------------+---------------+------------+------------+-----------+---------+
| c44-compute-1 | 172.26.233.57 | 10.21.1.26 | 10.21.1.26 | 10.2.2.23 |   None  |
|               |               |            |            |           |         |
| c44-compute-2 | 172.26.233.58 | 10.21.1.23 | 10.21.1.23 | 10.2.2.21 |   None  |
|               |               |            |            |           |         |
+---------------+---------------+------------+------------+-----------+---------+
```

```
Storage nodes: 3
+---------------+---------------+------------+------------+--------+-----------+
|     Server    |      CIMC     | Management | Provision  | Tenant |  Storage  |
+---------------+---------------+------------+------------+--------+-----------+
| c44-storage-3 | 172.26.233.53 | 10.21.1.22 | 10.21.1.22 |  None  | 10.3.3.22 |
|               |               |            |            |        |           |
| c44-storage-2 | 172.26.233.52 | 10.21.1.24 | 10.21.1.24 |  None  | 10.3.3.23 |
|               |               |            |            |        |           |
| c44-storage-1 | 172.26.233.51 | 10.21.1.21 | 10.21.1.21 |  None  | 10.3.3.21 |
|               |               |            |            |        |           |
+---------------+---------------+------------+------------+--------+-----------+
[root@c44-top-mgmt openstack-configs]#
```

Verifying Cisco VIM Client CLI Availability

The Cisco VIM Client CLI is very important for managing Cisco NFVI pods. After the Cisco NFVI installation is complete, verify that the Cisco VIM client is running and pointing to the right management node installer directory. Cisco NFVI provides a tool that you can use to check the REST API server status and directory where it is running.

To launch the tool, enter the following:

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

Confirm that the server status is active and check that the REST API launch directory matches the directory where the installation is launched.

The REST API command also provides options to launch, tear down, and reset the REST API server password as listed below:

```
# ./restapi.py -h
usage: restapi.py [-h] --action ACTION [--yes] [--verbose]

REST API setup helper

optional arguments:
  -h, --help            show this help message and exit
  --action ACTION, -a ACTION
                        setup - Install and Start the REST API server.
                        teardown - Stop and Uninstall the REST API
       server.
                        restart - Restart the REST API server.
                        regenerate-password - Regenerate the password for
       REST API server.
                        reset-password - Reset the REST API password with
      user given password.
                        status - Check the status of the REST API server
  --yes, -y             Skip the dialog. Yes to the action.
  --verbose, -v         Perform the action in verbose mode.
```

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

```
# cd installer-<tagid>/
# ciscovim –setupfile ~/Save/<setup_data.yaml> run
```

If the installer directory or the REST API state is not correct or pointing to an incorrect REST API launch directory, go to the installer-<tagid>/tools dir and execute:

```
# ./restapi.py –action setup
```

Confirm that the REST API server state and launch directory is correct:

```
# ./restapi.py –action status
```

If the REST API recovery step was run on an existing pod, run the following command to ensure that the REST API server continues to manage IT:

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

Note Detailed information about the Cisco NFVI REST API is provided in the Cisco Network Function Virtualization Infrastructure Administrator Guide.

Displaying Cisco NFVI Logs

Cisco NFVI installation logs are generated in the management node /var/log/mercury//<*install_uuid*>/ directory. The last 20 log directories are tarred and kept in this directory. The logs are archived (tar.gz file) after each run. The following table lists the Cisco NFVI installation steps and corresponding log files:

Table 1: Cisco NFVI Installation Logs

Step	Description	Log File
1	INPUT_VALIDATION	mercury_baremetal_install.log
2	MGMTNODE_ORCHESTRATION	mercury_buildorchestration.log
3	VALIDATION	mercury_baremetal_install.log
4	BAREMETAL	mercury_baremetal_install.log
5	COMMONSETUP	mercury_os_install.log
6	CEPH	mercury_ceph_install.log
7	ORCHESTRATION	mercury_os_install.log
8	VMTP	none

Accessing OpenStack API Endpoints

The Cisco NFVI installer stores access credentials in the management node /root/installer-<*tag-number*>/openstack-configs/openrc. The external_lb_vip_address provided in setup_data.yaml is the IP address where OpenStack APIs are handled. An openrc example is shown below:

```
export OS_AUTH_URL=http://172.26.233.139:5000/v2.0 or
 https://172.26.233.139:5000/v2.0 (if TLS is enabled)
export OS_USERNAME=admin
export OS_PASSWORD=xyzabcd
export OS_TENANT_NAME=admin
export OS_REGION_NAME=RegionOne
#  For TLS, add
export OS_CACERT=/root/openstack-configs/haproxy-ca.crt
```

The corresponding setup_data.yaml entry:

```
###############################################
# HA Proxy
###############################################
external_lb_vip_address: 172.26.233.139
```

Assessing Cisco NFVI Health with CloudPulse

You can use the OpenStack CloudPulse tool to verify Cisco NFVI health. CloudPulse servers are installed in containers on all Cisco NFVI control nodes, and CloudPulse clients are installed on the management node. Run the following commands to display Cisco NFVI information. For information about CloudPulse, visit the OpenStack CloudPulse website.

To check the results of periodic CloudPulse runs:

```
# cd /root/openstack-configs
# source openrc
# cloudpulse result
+--------------------------------------+------+-------------------+----------+---------+
| uuid                                 | id   | name              | testtype | state   |
+--------------------------------------+------+-------------------+----------+---------+
| bf7fac70-7e46-4577-b339-b1535b6237e8 | 3788 | glance_endpoint   | periodic | success |
| 1f575ad6-0679-4e5d-bc15-952bade09f19 | 3791 | nova_endpoint     | periodic | success |
| 765083d0-e000-4146-8235-ca106fa89864 | 3794 | neutron_endpoint  | periodic | success |
| c1c8e3ea-29bf-4fa8-91dd-c13a31042114 | 3797 | cinder_endpoint   | periodic | success |
| 04b0cb48-16a3-40d3-aa18-582b8d25e105 | 3800 | keystone_endpoint | periodic | success |
| db42185f-12d9-47ff-b2f9-4337744bf7e5 | 3803 | glance_endpoint   | periodic | success |
| 90aa9e7c-99ea-4410-8516-1c08beb4144e | 3806 | nova_endpoint     | periodic | success |
| d393a959-c727-4b5e-9893-e229efb88893 | 3809 | neutron_endpoint  | periodic | success |
| 50c31b57-d4e6-4cf1-a461-8228fa7a9be1 | 3812 | cinder_endpoint   | periodic | success |
| d1245146-2683-40da-b0e6-dbf56e5f4379 | 3815 | keystone_endpoint | periodic | success |
| ce8b9165-5f26-4610-963c-3ff12062a10a | 3818 | glance_endpoint   | periodic | success |
| 6a727168-8d47-4a1d-8aa0-65b942898214 | 3821 | nova_endpoint     | periodic | success |
| 6fbf48ad-d97f-4a41-be39-e04668a328fd | 3824 | neutron_endpoint  | periodic | success |
```

To run a CloudPulse test on demand:

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

To run a specific CloudPulse test on demand:

```
[root@vms-line2-build installer-3128.2]# cloudpulse run --name neutron_endpoint
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| name       | neutron_endpoint                     |
| created_at | 2016-03-29T02:20:16.840581+00:00     |
| updated_at | None                                 |
| state      | scheduled                            |
| result     | NotYetRun                            |
| testtype   | manual                               |
| id         | 3827                                 |
| uuid       | 5cc39fa8-826c-4a91-9514-6c6de050e503 |
+------------+--------------------------------------+
[root@vms-line2-build installer-3128.2]#
```

To show detailed results from a specific CloudPulse run:

```
[root@vms-line2-build installer-3128.2]# cloudpulse show 5cc39fa8-826c-4a91-9514-6c6de050e503
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
```

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

CloudPulse has two test sets: endpoint_scenario (runs as a cron or manually) and operator test (run manually). Endpoint tests include:

- nova_endpoint
- neutron_endpoint
- keystone_endpoint
- glance_endpoint
- cinder_endpoint

Operator tests include

- ceph_check
- docker_check
- galera_check
- node_check
- rabbitmq_check

The following table lists the operator tests you can perform with CloudPulse.

Table 2: CloudPulse Operator Tests

Test	Description
Ceph Check	Executes the ceph -f json status command on the Ceph-mon nodes and parses the output. If the result of the output is not HEALTH_OK, the ceph_check reports an error.
Docker Check	Finds out if all Docker containers are in running state on all nodes and reports an error if any containers are in the Exited state. The Docker check runs the command, docker ps -aq --filter 'status=exited',
Galera Check	Executes the command, mysql 'SHOW STATUS, on the controller nodes and displays the status.
Node Check	Checks if all the nodes in the system are up and online. It also compares the results of the Nova hypervisor list and determines whether all the compute nodes are available.
RabbitMQ Check	Runs the command, rabbitmqctl cluster_status, on the controller nodes and finds out if the RabbitMQ cluster is in quorum. If nodes are offline, the rabbitmq_check reports a failure.

Displaying HAProxy Dashboard and ELK Stack Logs

You can view the HAProxy dashboard at: http://< *external_lb_vip_address* >:1936 using the following username and password

- Username—haproxy
- Password—Value for HAPROXY_PASSWORD in /root/installer-<tag-number>/openstack-configs/secrets.yaml

You can use the Kibana dashboard to view logs aggregated by Logstash at: http://< *management_node_IP*>:5601 using the following username and password

- Username—admin
- Password—Value for ELK_PASSWORD in /root/installer-<tag-number>/openstack-configs/secrets.yaml

Checking Cisco NFVI Pod and Cloud Infrastructure

To test the Cisco NFVI pod and cloud infrastructure (host connectivity, basic mraiadb, rabbit, ceph cluster check and RAID disks), you can use the cloud-sanity tool available on the management node. To execute, enter:

```
# cd installer-<tagid>/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.
```

To list the available cloud-sanity checks, execute:

```
# ./cloud_sanity.py -l


Available sanity checks
-----------------------

-----------------------
 1 -  cloud-sanity : Management - Disk maintenance RAID Health
 2 -  cloud-sanity : Management - Disk maintenance VD Health
```

```
 3 -  cloud-sanity : Control - Ping All Controller Nodes
 4 -  cloud-sanity : Control - Ping internal VIP
 5 -  cloud-sanity : Control - Check Mariadb cluster size
 6 -  cloud-sanity : Control - Check RabbitMQ is running
 7 -  cloud-sanity : Control - Check RabbitMQ cluster status
 8 -  cloud-sanity : Control - Check Nova service list
 9 -  cloud-sanity : Control - Disk maintenance RAID Health
10 -  cloud-sanity : Control - Disk maintenance VD Health
11 -  cloud-sanity : Compute - Ping All Compute Nodes
12 -  cloud-sanity : Compute - Check Nova Hypervisor list
13 -  cloud-sanity : Compute - Disk maintenance RAID Health
14 -  cloud-sanity : Compute - Disk maintenance VD Health
15 -  cloud-sanity : CephMon - Check cephmon is running
16 -  cloud-sanity : CephMon - CEPH cluster check
17 -  cloud-sanity : CephMon - Check Ceph Mon status
18 -  cloud-sanity : CephMon - Check Ceph Mon results
19 -  cloud-sanity : CephOSD - Ping All Storage Nodes
20 -  cloud-sanity : CephOSD - Check OSD result with osdinfo
21 -  cloud-sanity : CephOSD - Check OSD result without osdinfo

Results for a test can be either passed, failed, or skipped.  A skipped test indicates a
test which couldn't be run on this particalur PoD – i.e. a hardware RAID test is skipped on
 a node which doesn't have hardware RAID.
A successful compute node check is shown below:
#./cloud_sanity.py -c compute
Executing Compute Cloud Sanity in quiet mode. This will take some time.

+---------+-------------------------------------------------------------+---------+
| Role    | Task                                                        | Result |
+---------+-------------------------------------------------------------+---------+
| Compute | Compute - Ping All Compute Nodes ***********************  | PASSED |
|         |                                                             |         |
| Compute | Compute - Check Nova Hypervisor list *******************  | PASSED |
|         |                                                             |         |
+---------+-------------------------------------------------------------+---------+
[PASSED] Cloud Sanity Compute Checks Passed
```

A failure example is shown below:

```
[root@MercTB1 tools]# ./cloud_sanity.py -c control
Executing Control Cloud Sanity in quiet mode. This will take some time.

+---------+-------------------------------------------------------------+---------+
| Role    | Task                                                        | Result |
+---------+-------------------------------------------------------------+---------+
| 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 ********************  | FAILED |
|         |                                                             |         |
+---------+-------------------------------------------------------------+---------+
[FAILED] FATAL ERROR occured when running sanity checks.
[NOTE] One or more testcase[s] skipped. Use --list to see the complete list.
```

To view the details of a failure, use the v option as shown below:

```
# ./cloud_sanity.py -c control -v

PLAY [Executes Cloud Sanity] ***************************************************

GATHERING FACTS ****************************************************************
ok: [7.7.7.15]
ok: [7.7.7.11]
ok: [7.7.7.14]

TASK: [cloud-sanity | Control - Ping All Controller Nodes] *********************
changed: [7.7.7.15 -> localhost] => (item=7.7.7.15)
changed: [7.7.7.15 -> localhost] => (item=7.7.7.11)
changed: [7.7.7.15 -> localhost] => (item=7.7.7.14)
```

```
TASK: [cloud-sanity | Control - Ping internal VIP] ****************************
changed: [7.7.7.15 -> localhost]

TASK: [cloud-sanity | Control - Check Mariadb cluster size] *******************
changed: [7.7.7.11]
changed: [7.7.7.15]
changed: [7.7.7.14]

TASK: [cloud-sanity | Control - Check RabbitMQ is running] ********************
failed: [7.7.7.11] => {"changed": true, "cmd": "docker ps -a | grep rabbit | grep Up | awk
 '{print $NF}' | cut -f2 -d '_'", "delta": "0:00:00.021044", "end": "2016-08-18
23:45:34.838817", "failed": true, "failed_when_result": true, "rc": 0, "start": "2016-08-18
 23:45:34.817773", "stdout_lines": [], "warnings": []}
changed: [7.7.7.15]
changed: [7.7.7.14]

FATAL: all hosts have already failed -- aborting

PLAY RECAP ********************************************************************
7.7.7.11                   : ok=4    changed=3    unreachable=0    failed=1
7.7.7.14                   : ok=5    changed=4    unreachable=0    failed=0
7.7.7.15                   : ok=5    changed=4    unreachable=0    failed=0

[FAILED] FATAL ERROR occured when running sanity checks.
[NOTE] One or more testcase[s] skipped. Use --list to see the complete list.
```