

# Cisco Ultra M

## Troubleshooting Guide



Introduction	4
Authors, Contributions and Methodology	5
Who Should Read This Book?	7
The Ultra M Solution	8
Introduction	9
Architecture and Components	13
Virtual Infrastructure Manager (VIM)	17
Networking and Storage	32
Storage	50
OpenStack Infrastructure Deployment with UAS	51
VPC-DI VNF	59
Elastic Services Controller	64
Element Manager	75
Troubleshooting	88
Troubleshooting Theory	89
Health Checks	95
Data Collection	117
Virtual Infrastructure Manager	128
CEPH Storage	153
VNF Recovery	156
StarOS	165

Virtual Network Function Manager & Element Manager	187
VPC-DI Upgrade	202
<b>Troubleshooting Use Cases</b>	<b>203</b>
PXE Boot Failure	204
CEPH Placement Groups Inconsistency	209
OpenStack Cluster Recovery	218
VM in an Unhealthy State	243
VM Unreachable via Floating IP	249
Di Network Congestion	256
CF/SF Unplanned Reboot or Stuck in Boot Process	267
Data Recovery from Failed CF	278
HD RAID Issues	290
DIMM Failures	294
IFTASK Tuning Considerations	309
<b>Acronyms</b>	<b>315</b>
Acronyms	315



# Introduction

## Authors, Contributions and Methodology

The virtualization world is a complex world in terms of deployments and, more importantly, troubleshooting when problems occur. With this context in mind, troubleshooting experts from the business unit and technical services have been brought together to contribute to the creation of this Ultra troubleshooting document. The objective of this document is to help support personnel to troubleshoot the Ultra M and VPC-DI solutions. This document is a complementary “tool” to be used to help recover from issues. It should be noted that this document is not meant to be a replacement for the hands-on training which can only be obtained by getting your hands dirty in the lab.

Fred Carpenito,  
Director of Engineering

.....

This guide represents the result of an intense collaborative effort between Cisco's Engineering and Technical Services experts coming from around the world (India, Belgium and USA) to create, in a single week, this collection of methodologies, procedures and troubleshooting techniques. Each person has been able to contribute to the guide using their strengths in the various disciplines that make up the curriculum, including their own perspectives and experiences drawn from their years of experience in the industry, working in the labs, writing code and knowledge articles, and supporting the customer base. Materials have been drawn from various internal sources including knowledge bases/wikis, product documentation, freely available Internet sources, case notes from customer experiences, personal knowledge storehouses, and of course most importantly one's own personal knowledge and experience.

The Book Sprint ([www.booksprints.net](http://www.booksprints.net)) methodology was used for writing this guide. The Book Sprint methodology is an innovative new style of cooperative and collaborative authorship. Book Sprints are strongly facilitated and leverage team-oriented inspiration and motivation to rapidly deliver large amounts of well-authored

and reviewed content, and incorporate it into a complete narrative in a short amount of time.

Please consider that due to its being put together in a short time, the most important focus has been on amassing the actual content, while adherence to consistent writing approaches across all the writers was considered of less importance. Even if the answer to your question is not listed directly, the material here will hopefully, in many cases, give you a good starting point to solving an issue that you might not otherwise have had, saving you time up front in your search. In other cases, it might give you exactly what you are looking for, and prepare you better for future issues that you may encounter.

This guide was written and produced by:

Dave Damerjian, Technical Leader, Services  
Dennis Lanov, Technical Leader, Services  
Jeff Williams, Technical Leader, Services  
Rama Ramachandran, Technical Leader, Engineering  
Robert West, Technical Leader, Services  
Roshan Warriar, Support Engineer, Services  
Snezana Mitrovic, Support Engineer, Services  
Solomon Ayankulankara Kunjan, Technical Leader, Services  
Sourav Jyoti Das, Support Engineer, Services

The Book Sprint team for the production of this book included: Faith Bosworth (facilitation), Raewyn Whyte (editor), Henrik van Leeuwen (illustrator), Agathe Baëz (book production), Juan Gutierrez (IT support).

## Who Should Read This Book?

This book was created for engineers who need to troubleshoot the Cisco Ultra M solution.

It is expected that the reader has prior knowledge, training and experience with these products and, especially, the related technologies. The book focuses on troubleshooting and also covers theoretical concepts for a better understanding of specific features.

For additional information related to configuration and features, please refer to the Cisco Ultra M Documentation.

# The Ultra M Solution

# Introduction

Ultra M is a pre-packaged and validated virtualized mobile packet core solution designed to simplify the deployment of virtual network functions (VNFs). The solution combines the Cisco Ultra Service Platform (USP) architecture, Cisco Validated OpenStack infrastructure, Cisco networking and computing hardware platforms, into a fully integrated and scalable stack. Ultra M provides the tools to instantiate and provide basic lifecycle management for VNF components on a complete OpenStack virtual infrastructure manager. The VPC-DI VNF currently provides virtualized instances of various 3G and 4G mobile packet core (MPC) gateways that enable mobile operators to offer enhanced mobile data services to their subscribers. VPC-DI addresses the scaling and redundancy limitations of VPC-SI (Single Instance) by extending the StarOS boundaries beyond a single VM. VPC-DI allows multiple VMs to act as a single StarOS instance with shared interfaces, shared service addresses, load balancing, redundancy, and a single point of management.

Multiple Ultra M models are available. These models are differentiated by their scale in terms of the number of active Service Functions (SFs), numbers of VNFs, and architecture:

- Ultra M Small
- Ultra M Medium
- Ultra M Large
- Ultra M Extra Small (XS)

The Hyper-Converged architecture combines the Ceph Storage and Compute node. The converged node is referred to as an OSD compute node. Non-Hyper-Converged Ultra M models implement separate Ceph Storage and Compute nodes.

VM configurations for different Ultra M Models are shown below:

Functions(s)	Non-Hyper-Converged			Hyper-Converged	
	Small	Medium	Large	XS Single VNF	XS Multi VNF
<b>AutoIT-VNF</b>	1	1	1	1	1
<b>AutoDeploy</b>	1	1	1	1	1
<b>AutoVNF</b>	3	3	3	3	2 per VNF
<b>ESC(VNFM)</b>	2	2	2	2	2 per VNF
<b>UEM</b>	3	3	3	3	3 per VNF
<b>CF</b>	2	2	2	2	2 per VNF
<b>SF</b>	9	12	16	8	8 per VNF

For Multi-VNF deployment only the Hyper-Converged XS Multi VNF model is needed. Up to a maximum of 4 VNFs can be deployed.

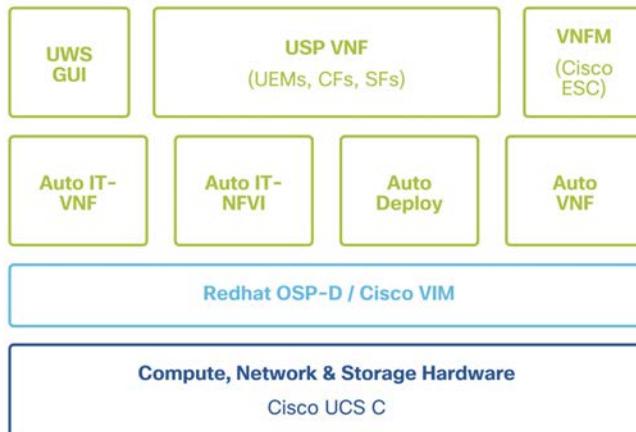
## Functional Components

The Ultra M solution consists of multiple hardware components including servers that function as controller, compute, and storage resources. The various functional software components that comprise the Ultra M solution are:

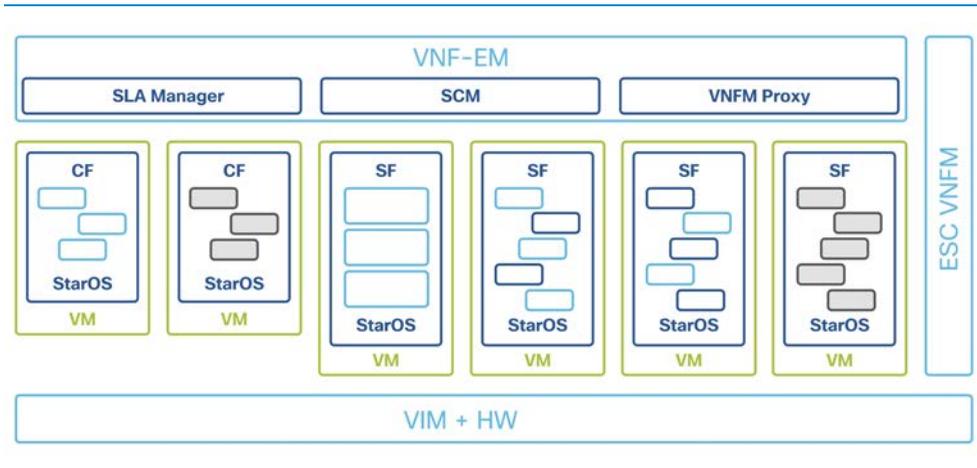
- Virtual Infrastructure Manager (Red Hat OSP-D or Cisco VIM) to manage the OpenStack infrastructure deployment and Day 2 management.
- Ultra Automation Services (UAS): A suite of tools provided to simplify the deployment process:
  - AutoIT-VNF: Provides storage and management for system ISOs.
  - AutoDeploy: Initiates the deployment of the VNFM and VNF components through a single deployment script.

- AutoVNF: Initiated by AutoDeploy, AutoVNF is directly responsible for deploying the VNFM and VNF components based on inputs received from AutoDeploy.
- Cisco Elastic Services Controller (ESC): Serves as the Virtual Network Function Manager (VNFM).
- VNF Components: USP-based VNFs are comprised of multiple components providing different functions:
  - Ultra Element Manager (UEM): Serves as the Element Management System (EMS, also known as the VNF-EM); it manages all of the major components of the USP-based VNF architecture.
  - Control Function (CF): A central sub-system of the VPC-DI VNF, the CF works with the UEM to perform lifecycle events and monitoring for the VPC-DI VNF.
  - Service Function (SF): Provides service context (user I/O ports), handles protocol signaling, session processing tasks, and flow control (demux).

Ultra M Components:



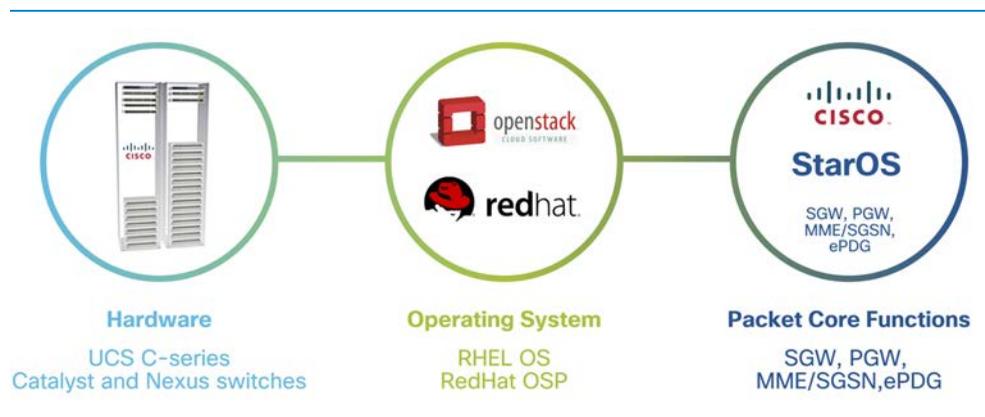
Functional components of Ultra M with various VMs are depicted below. Some of the infrastructure VMs, such as AutoIT and AutoDeploy, are not redundant. Open Stack Controllers and CEPH clusters have redundancy and are part of the infrastructure. VNFM-specific VMs such as EM, ESC, SF, and CFs have redundancy. Each VNF will have specific requirements which can vary: for example, in a StarOS VNF, each CF and SF should be deployed on separate compute nodes.



# Architecture and Components

Ultra Services can be deployed in various models depending upon the software and hardware components that are chosen.

Ultra M is a Cisco Validated Design implementation of the Ultra Services Platform running with fixed hardware and software configurations.



## Cisco Ultra M HW Configurations:

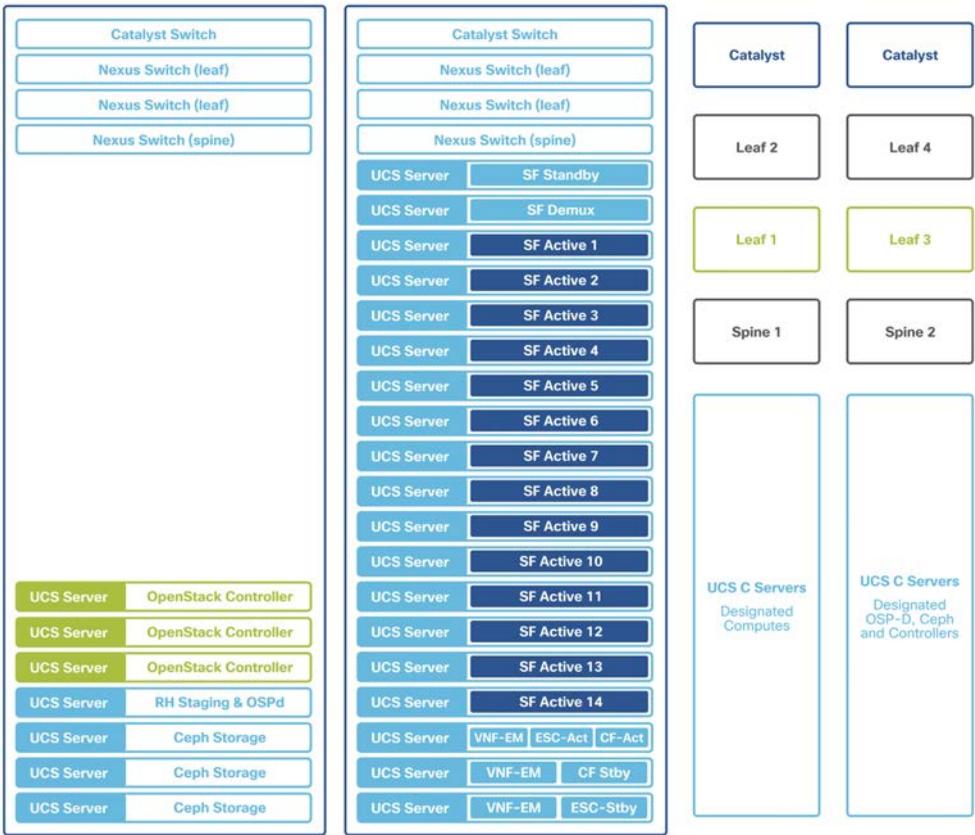
Details of the current Cisco-validated deployment models can be found in Cisco Ultra Gateway Platform Configuration Guides:

<https://www.cisco.com/c/en/us/support/wireless/ultra-gateway-platform/products-installation-and-configuration-guides-list.html>

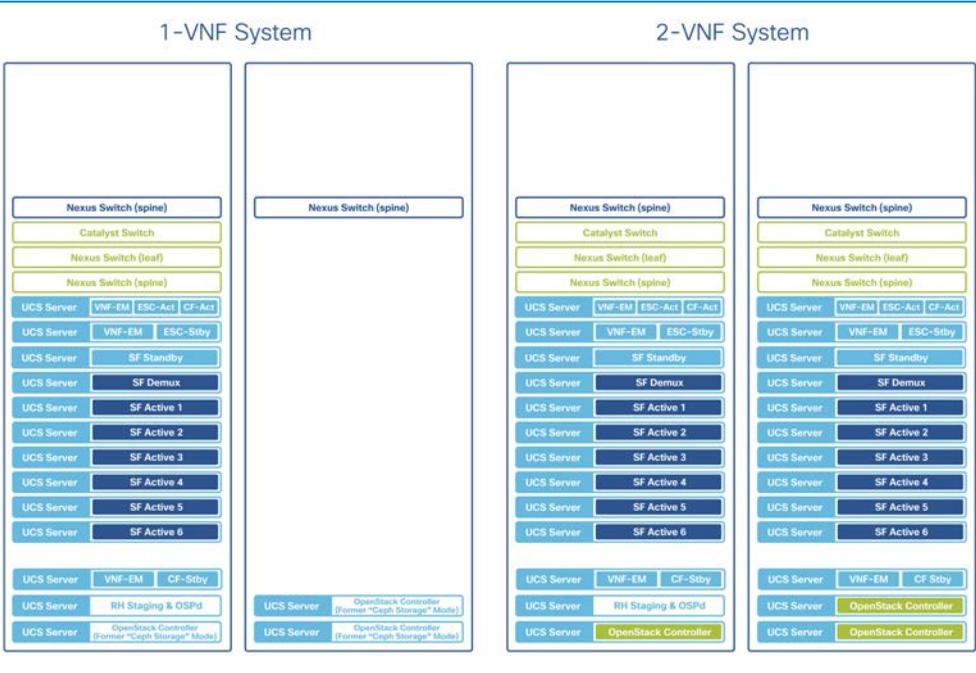
## Ultra M Rack Setup

The diagrams below show the rack setups for an Ultra M Solution for 1-VNF and 2-VNF systems.

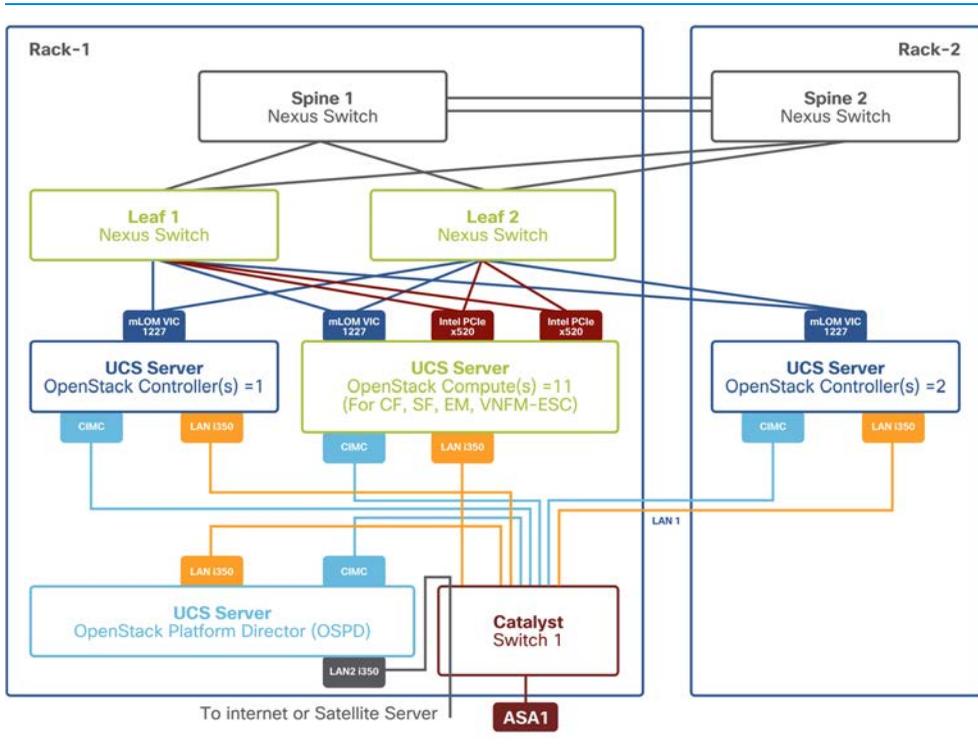
The UCS rack servers host the OpenStack and VPC-DI VMs. The example below shows the rack setup for an Ultra M Large Model:



Below is the rack setup for 1-VNF and 2-VNF systems:



The Ultra M is based on Cisco UCS C servers and Nexus switches. The overall architecture for a single VNF is shown below, which includes UCS servers, Switches (leaf and spine) and OSP-D or Cisco VIM. On a UCS server, various USP components are run, and switches provide connectivity (Di Network and Service Network):



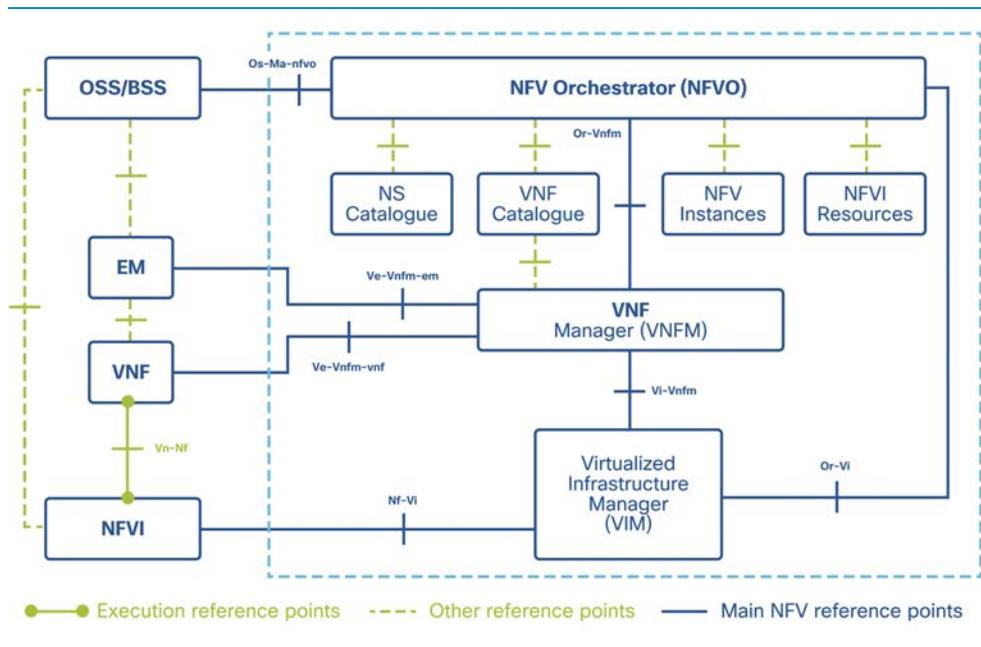
The network connectivity should be made according to the Deployment Guide.

# Virtual Infrastructure Manager (VIM)

The European Telecommunications Standards Institute (ETSI) has defined the network functions virtualization management and orchestration (NFV-MANO) architecture, comprising three major functional blocks:

- Virtual Infrastructure Manager (VIM).
- Virtualized Network Functions (VNFs).
- NFV Orchestrator (NFVO).

The Virtualized Infrastructure Manager (VIM) is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one operator's infrastructure domain. The picture below (image source: <http://www.etsi.org>) shows the VIM in the NFV architecture:



As identified in the ETSI specification, major VIM functions include the following:

- Management of inventory-related information for NFVI hardware resources, software resources and discovery of the capabilities and features of such resources.
- Management of the virtualized resource capacity and forwarding of information related to NFVI resources capacity and usage reporting.
- Orchestrating the allocation/upgrade/release/reclamation of NFVI resources and managing the association of the virtualized resources to the physical compute, storage, and networking resources.

In this document, the focus is the OpenStack VIM which is either RedHat OpenStack or Cisco VIM.

In the Ultra M solution, OpenStack Platform Director (OSP-D) or Cisco VIM functions as the virtual infrastructure manager (VIM). Hyper-Converged Ultra M Single and Multi-VNF Models use Redhat OpenStack Platform 10 (OSP 10-Newton).

RedHat OSP-D is based on OpenStack-On-OpenStack [Triple-O] architecture where the Undercloud OpenStack runs on a UCS server and installs Overcloud OpenStack on other UCS servers. The UCS servers take the role of compute, control and storage node (as per OpenStack terminology).

Cisco VIM is another supported VIM and runs on UCS Servers. The Cisco VIM follows a Kolla-Based OpenStack Model where all the OpenStack services are deployed as containers. Cisco VIM has a management node which contains the container repository and is responsible for deploying the whole OpenStack on UCS servers.

## Generic VIM Concepts

### NUMA

These two technologies aim to deliver improved performance in the VIM layer based on the knowledge of compute host layout. Scheduling can be improved by factoring workloads that are expected to be handled by a guest OS.

In multiprocessor computing environments, the x86 system was designed with equal access for all CPUs in the system, resulting in Uniform Memory Access. This changed when system memory was divided into zones and associated with a particular CPU in modern multi-CPU systems. With more sockets and threads being used, an interconnect bus was designed to handle connections from all CPUs while accessing memory was considered an improvement compared to CPU overclocking. The end result was a new technology called NUMA which facilitates faster access to local zones and slower access to remote zones.

Below is a sample output of `lscpu` from a compute node. There are 2 sockets on this node where each CPU is placed. Each socket (CPU) has 14 cores and 2 threads per core available. Altogether there are  $(2 \times 14 \times 2) = 56$  VCPU available. The numbering order for VCPU would be 0-55:

```
[root@compute-node ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:           Little Endian
CPU(s):               56
On-line CPU(s) list: 0-55
Thread(s) per core:  2
Core(s) per socket:  14
Socket(s):            2
NUMA node(s):        2
<..>
```

To check the operating system of the compute node which supports NUMA features, use the command below:

```
[root@compute-node ~]# lscpu | grep NUMA
NUMA node(s):          2
NUMA node0 CPU(s):    0-13,28-41
NUMA node1 CPU(s):    14-27,42-55
[root@compute-node ~]#
```

In the above output, there are 2 NUMA nodes available. The first 14 CPU (0-13) belongs to the first NUMA Node and next 14 CPU (14-27) belongs to 2nd NUMA Node. The rest of the CPU cores [28-41 and 42-55] are not physical cores, but they are part of hyper-threading.

The output below shows the capability of a hypervisor to support NUMA:

```
novalibvirt_11111 [root@compute-node /]# virsh nodeinfo
CPU model:          x86_64
CPU(s):             56
CPU frequency:      2600 MHz
CPU socket(s):      1
Core(s) per socket: 14
Thread(s) per core: 2
NUMA cell(s):       2
Memory size:        201212936 KiB
```

The NUMA nodes are each represented by a <cell> entry which lists the CPUs available within the node, the memory available within the node – including the page sizes, and the distance between the node and its siblings:

```
novalibvirt_11111 [root@compute-node /]# virsh capabilities
<capabilities>

  <host>
    <uuid>355303ae-b3c4-8145-b608-f56ad4655343</uuid>
    <cpu>
      <arch>x86_64</arch>

  --
```

```

<topology>
  <cells num='2'>
    <cell id='0'>
      <memory unit='KiB'>100549640</memory>
      <pages unit='KiB' size='4'>24630018</pages>
      <pages unit='KiB' size='2048'>41951</pages>
      <pages unit='KiB' size='1048576'>0</pages>
      <distances>
        <sibling id='0' value='10' />
        <sibling id='1' value='21' />
      </distances>
      <cpus num='28'>
        <cpu id='0' socket_id='0' core_id='0' siblings='0,28' />
        <cpu id='1' socket_id='0' core_id='1' siblings='1,29' />
        <cpu id='2' socket_id='0' core_id='2' siblings='2,30' />
        <cpu id='3' socket_id='0' core_id='3' siblings='3,31' />
        <cpu id='4' socket_id='0' core_id='4' siblings='4,32' />
        <cpu id='5' socket_id='0' core_id='5' siblings='5,33' />
        <cpu id='6' socket_id='0' core_id='6' siblings='6,34' />
        <cpu id='7' socket_id='0' core_id='8' siblings='7,35' />
        <cpu id='8' socket_id='0' core_id='9' siblings='8,36' />
        <cpu id='9' socket_id='0' core_id='10' siblings='9,37' />
        <cpu id='10' socket_id='0' core_id='11' siblings='10,38' />
        <cpu id='11' socket_id='0' core_id='12' siblings='11,39' />
        <cpu id='12' socket_id='0' core_id='13' siblings='12,40' />
        <cpu id='13' socket_id='0' core_id='14' siblings='13,41' />
        <cpu id='28' socket_id='0' core_id='0' siblings='0,28' />
        <cpu id='29' socket_id='0' core_id='1' siblings='1,29' />
        <cpu id='30' socket_id='0' core_id='2' siblings='2,30' />
        <cpu id='31' socket_id='0' core_id='3' siblings='3,31' />
        <cpu id='32' socket_id='0' core_id='4' siblings='4,32' />
        <cpu id='33' socket_id='0' core_id='5' siblings='5,33' />
        <cpu id='34' socket_id='0' core_id='6' siblings='6,34' />
        <cpu id='35' socket_id='0' core_id='8' siblings='7,35' />
        <cpu id='36' socket_id='0' core_id='9' siblings='8,36' />
        <cpu id='37' socket_id='0' core_id='10' siblings='9,37' />
        <cpu id='38' socket_id='0' core_id='11' siblings='10,38' />
        <cpu id='39' socket_id='0' core_id='12' siblings='11,39' />
        <cpu id='40' socket_id='0' core_id='13' siblings='12,40' />
        <cpu id='41' socket_id='0' core_id='14' siblings='13,41' />
      </cpus>
    </cell>
    <cell id='1'>
      <memory unit='KiB'>100663296</memory>
      <pages unit='KiB' size='4'>24658944</pages>
      <pages unit='KiB' size='2048'>41950</pages>

```

```

<pages unit='KiB' size='1048576'>0</pages>
<distances>
  <sibling id='0' value='21' />
  <sibling id='1' value='10' />
</distances>
<cpus num='28'>
  <cpu id='14' socket_id='1' core_id='0' siblings='14,42' />
  <cpu id='15' socket_id='1' core_id='1' siblings='15,43' />
  <cpu id='16' socket_id='1' core_id='2' siblings='16,44' />
  <cpu id='17' socket_id='1' core_id='3' siblings='17,45' />
  <cpu id='18' socket_id='1' core_id='4' siblings='18,46' />
  <cpu id='19' socket_id='1' core_id='5' siblings='19,47' />
  <cpu id='20' socket_id='1' core_id='6' siblings='20,48' />
  <cpu id='21' socket_id='1' core_id='8' siblings='21,49' />
  <cpu id='22' socket_id='1' core_id='9' siblings='22,50' />
  <cpu id='23' socket_id='1' core_id='10' siblings='23,51' />
  <cpu id='24' socket_id='1' core_id='11' siblings='24,52' />
  <cpu id='25' socket_id='1' core_id='12' siblings='25,53' />
  <cpu id='26' socket_id='1' core_id='13' siblings='26,54' />
  <cpu id='27' socket_id='1' core_id='14' siblings='27,55' />
  <cpu id='42' socket_id='1' core_id='0' siblings='14,42' />
  <cpu id='43' socket_id='1' core_id='1' siblings='15,43' />
  <cpu id='44' socket_id='1' core_id='2' siblings='16,44' />
  <cpu id='45' socket_id='1' core_id='3' siblings='17,45' />
  <cpu id='46' socket_id='1' core_id='4' siblings='18,46' />
  <cpu id='47' socket_id='1' core_id='5' siblings='19,47' />
  <cpu id='48' socket_id='1' core_id='6' siblings='20,48' />
  <cpu id='49' socket_id='1' core_id='8' siblings='21,49' />
  <cpu id='50' socket_id='1' core_id='9' siblings='22,50' />
  <cpu id='51' socket_id='1' core_id='10' siblings='23,51' />
  <cpu id='52' socket_id='1' core_id='11' siblings='24,52' />
  <cpu id='53' socket_id='1' core_id='12' siblings='25,53' />
  <cpu id='54' socket_id='1' core_id='13' siblings='26,54' />
  <cpu id='55' socket_id='1' core_id='14' siblings='27,55' />
</cpus>
</cell>
</cells>
</topology>

```

--

## CPU Pinning

CPU pinning refers to the mechanism in which the OpenStack compute environment is configured to reserve dedicated CPU cores for compute nodes and guest VM processes so that there is no resource crunch for the same CPU core. This allows for aggregating all hosts configured for CPU pinning and creates a performance-focused environment.

The **virsh vcpuinfo** command gives us information about CPU pinning:

```
novalibvirt_11111 [root@compute-node /]# virsh list --all
Id      Name                               State
-----
5       instance-000000c1                 running

novalibvirt_12614 [root@rcdn-c3-compute8-sriov /]#
novalibvirt_12614 [root@rcdn-c3-compute8-sriov /]# virsh vcpuinfo 5
VCPU:      0
CPU:       9
State:     running
CPU time:  8776.9s
CPU Affinity:  -----y-----
--
```

If the CPU Pinning is not set, the output will appear as shown below:

```
VCPU:      0 CPU:      9 State:      running CPU time:    0.5s
CPU Affinity:  yyyyyyyy
```

In the example above, the VM has its first VCPU currently mapped to physical CPU number 9. The 'CPU Affinity' line shows that VCPU number 0 has CPU pinning enabled and it is mapped to physical CPU 9.

The same output can also be checked from "virsh dumpxml" command for the VM. In the output below, VCPU 0 is mapped to CPU 9 via cpuset attributes.

```
novalibvirt_11111 [root@compute-node /]# virsh dumpxml 5
<domain type='kvm' id='5'>
  <name>instance-000000c1</name>
```

```

--
<vcpu placement='static'>12</vcpu>
<cputune>
  <shares>12288</shares>
  <vcpupin vcpu='0' cpuset='9' />
  <vcpupin vcpu='1' cpuset='11' />
  <vcpupin vcpu='2' cpuset='34' />
  <vcpupin vcpu='3' cpuset='40' />
  <vcpupin vcpu='4' cpuset='33' />
  <vcpupin vcpu='5' cpuset='10' />
  <vcpupin vcpu='6' cpuset='27' />
  <vcpupin vcpu='7' cpuset='50' />
  <vcpupin vcpu='8' cpuset='49' />
  <vcpupin vcpu='9' cpuset='48' />
  <vcpupin vcpu='10' cpuset='25' />
  <vcpupin vcpu='11' cpuset='18' />
  <emulatorpin cpuset='9-11,18,25,27,33-34,40,48-50' />
</cputune>
<numatune>
  <memory mode='strict' nodeset='0-1' />
  <memnode cellid='0' mode='strict' nodeset='0' />
  <memnode cellid='1' mode='strict' nodeset='1' />
</numatune>
--

```

In this example, the VM has strict NUMA mapping with both the NUMA nodes.

```

novalibvirt_11111 [root@compute-node /]# virsh numatune 5
numa_mode      : strict
numa_nodeset   : 0-1

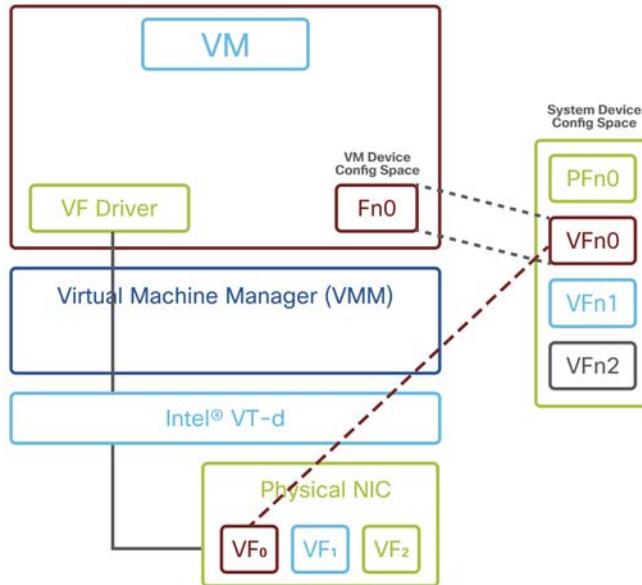
```

## SR-IOV

A Single Root I/O Virtualization card is an Intel technology-based NIC used in virtualization implementations. It is a standardized way of bypassing the VMM's involvement in data movement by providing independent memory space, interrupts, and DMA streams for each virtual machine. SR-IOV provides a mechanism by which a Single Root Function (for example a single Ethernet port) can appear to be multiple separate physical devices.

SR-IOV introduces two new function types:

- Physical Functions (PFs): Full PCIe functions that include the SR-IOV extended capability. The capability is used to configure and manage the SR-IOV functionality.
- Virtual Functions (VFs): Lightweight PCIe functions that contain the resources necessary for data movement and a minimized set of configuration resources.
- Intel NIC SR-IOV functionality is described below:



In the Ultra M Solution, different types of Intel NIC cards are used, such as Intel X520 or Intel X710.

In the Intel X520 Card, Each PF supports 32 VFs and typically has 128 queues.

By default, the SR-IOV VF seems to support only 2 Tx and 2 Rx queues, but even if iftask from StarOS guest is requesting for 13 Tx and 5 Rx queues, only the maximum available per VF will be allocated. Example debug console logs shown below:

```
1237413 Wed Jan 17 04:23:38 2018^@ PID:7719 APP: max rx queues supported 2
1237414 Wed Jan 17 04:23:38 2018^@ PID:7719 APP: max tx queues supported 2
1237415 Wed Jan 17 04:23:38 2018^@ PID:7719 APP: hw rx requested 5
1237416 Wed Jan 17 04:23:38 2018^@ PID:7719 APP: hw tx requested tx 13
```

If too many VFs are created, then each VF will have a fewer number of queues – in this case,  $128/32 = 4$  queues, 2 for transmitting and 2 for receiving.

If 16 VFs are created, the system will allocate  $128/16=8$  queues per VF, 4 for Tx and 4 for Rx. The maximum number of queues used in StarOS is 4 in each direction and the settings mentioned below are the recommended configurations for getting the best performance.

In OSP-D deployment, the following lines in network.yaml file need to be changed:

```
NeutronSriovNumVFs: "enp10s0f0:16,enp10s0f1:16,enp133s0f0:16,enp133s0f1:16"
```

Then Overcloud deployment will take care of setting the parameters correctly for the SR-IOV card.

For Cisco VIM, setup\_data.yaml file needs to be uncommented and the following line changed from:

```
# INTEL_SRIOV_VFS: <integer>

to:

INTEL_SRIOV_VFS: 16
```

This change will require a reboot of UCS.

Even though the data transfer between VF to guest operating system is known, there are two components to the SR-IOV driver - one is on the host operating system and the

other is on the guest operating system. Make sure that the correct OpenStack Version and RedHat Versions are used as mentioned in this document.

### Pause Frames and SR-IOV

Pause Frames (flow-control) are used in Gigabit Ethernet networking as a flow-control mechanism which is normally enabled in both directions so that a switch can signal a server to slow down (Pause) and resume sending traffic (No Pause). Similarly, a server can send a similar signal through the flow control towards the switch, which can result in more traffic being dropped, depending on the buffering capacity of the switches.

With SR-IOV cards, Pause Frames can have a detrimental effect on VPC-DI VMs. If a Pause Frame is sent towards an SR-IOV card, then its treatment by the card can cause heartbeats going between SF and CF cards to be paused, resulting in a reload of the VM. If both CFs reload, then it is going to cause a complete subscriber loss because it is similar to a complete reload of the VNF. Therefore, it is recommended to disable the Pause Frames for SR-IOV ports in both directions.

### NIC Bonding and Snooping

To make NIC bonding failover work correctly, check the NIC port configuration and make sure spoof checking is on. This is done at RedHat Linux level using the following command:

```
ip link show enp10s0f0

enp10s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode
DEFAULT qlen 1000
    link/ether 90:e2:ba:d6:81:44 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 5 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 6 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 7 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 8 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 9 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 10 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
    vf 11 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
```

```

vf 12 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
vf 13 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off
vf 14 MAC fa:16:3e:2a:d0:06, spoof checking on, link-state auto, trust off
vf 15 MAC fa:16:3e:76:b6:a1, spoof checking on, link-state auto, trust off

```

This output shows a correct number of VFs configured with spoof checking on, which is the correct mode.

## Hashing

The hashing mechanism is used for assigning packets and sessions to various components inside the VM. The incoming packets are hashed using a combination of IP Source Address, Destination Address and Transport for UDP traffic (three tuples) and using IP Source Address, Destination Address, Transport, Source port and Destination port for TCP traffic (5 tuples). This is by default in both IPv4 and IPv6 cases. For mobility, since eGTP packets are UDP-based, this leads to lesser granularity and can cause polarization.

The way to identify this issue is using the command as below:

```

[root@ultram-compute-0 ~]# ethtool -n enp10s0f0 rx-flow-hash udp4
UDP over IPV4 flows use these fields for computing Hash flow key:
IP SA
IP DA

```

The above command shows three tuples used. The way to correct the situation is using the commands below:

```

ethtool -N enp10s0f0 rx-flow-hash udp4 sdfn
ethtool -N enp10s0f0 rx-flow-hash udp6 sdfn

```

After the correction, the output will look like:

```

[root@ultram-compute-0 ~]# ethtool -n enp10s0f0 rx-flow-hash udp4
UDP over IPV4 flows use these fields for computing Hash flow key:
IP SA
IP DA

```

```
L4 bytes 0 & 1 [TCP/UDP src port]
L4 bytes 2 & 3 [TCP/UDP dst port]
```

## Configuring SW RSS

The UCS NIC supports hardware-based Receive Side Scaling (RSS); however, RSS is only supported on IP traffic. For other network protocols, such as MPLS, GTP, L2TP, and GRE, all the traffic is routed into a single queue.

The VPC-DI provides a software RSS capability that distributes MPLS traffic to the available vCPU cores for processing. This increases resource utilization and provides improved throughput.

The software RSS capability can be supplemental to the Cisco UCS NIC hardware RSS support, meaning that it distributes some traffic not supported by the hardware NIC (MPLS traffic only in this release). The VPC-DI can also provide comprehensive RSS coverage, meaning that it distributes all traffic. This option is applicable when hardware that does not support RSS is used.

It is recommended to add the following to StarOS configuration (this is a global command):

```
config
  iftask sw-rss supplemental
```

## Fencing

The Ultra M Red Hat-based deployment model uses three (3) OpenStack controller nodes. When fencing is not enabled, one faulty controller node could cause data corruption in a cluster. Fencing is the process of isolating a node to protect a cluster and its resources. Adding fencing to the controller nodes on the Ultra M system will provide high-availability and help mitigate some of the issues.

For a detailed step-by-step guide on how to enable fencing on the Red Hat Controllers, refer to official Red Hat documentation for appropriate release. A summary of the steps is provided below.

The above does not apply to Cisco VIM-based deployments.

Fencing pre-requisites:

1. Make sure all controllers have connectivity to the provisioning network and the IPMI/CIMC network. The following command may be used to check the connectivity:

```
for i in $(nova list | grep control | awk '{print $12}' | sed 's/ctlplane=//g')
do
(ssh -o StrictHostKeyChecking=no heat-admin@$i "hostname; sudo cat /etc/sysconfig/
network-scripts/route-enol")
done

ultram-controller-0.localdomain
169.254.169.254/32 via 192.200.0.1 dev enol
ultram-controller-1.localdomain
169.254.169.254/32 via 192.200.0.1 dev enol
ultram-controller-2.localdomain
169.254.169.254/32 via 192.200.0.1 dev enol
```

2. Obtain the CIMC IP address, username and password for all 3 controllers
3. Login on each controller and configure the following:

It is important to provide the correct IP address for each controller so that the expected node is fenced on failure. Replace the highlighted text with correct IP address, username and password.

```
heat-admin@controller-0 ~]$ sudo pcs stonith create my-ipmilan-for-controller-0
fence_\  
ipmilan pcmk_host_list=controller-0 ipaddr=192.100.0.1 login=admin passwd=admin  
lanplus=1 op moni\  
tor interval=60s
```

4. Enable fencing for the cluster by issuing the following command on any of the controller nodes. This only needs to be executed from one controller node.

```
[heat-admin@controller-2 ~]$ sudo pcs property set stonith-enabled=true
```

Post Operation Checks:

To verify the fencing is enabled/started and no errors reported

```
[heat-admin@controller-0 ~]$ sudo pcs stonith show
my-ipmilan-for-controller-0    (stonith:fence_ipmilan):    Started controller-0
my-ipmilan-for-controller-1    (stonith:fence_ipmilan):    Started controller-1
my-ipmilan-for-controller-2    (stonith:fence_ipmilan):    Started controller-0
```

Further checks:

From the output of the **pcs status** command, verify the following:

- Check that all 3 controllers are listed as Online.
- Check that haproxy-clone is started on all 3 controllers.
- Check that galera-master lists all 3 controllers as Masters.
- Check that rabbitmq-clone is started on all 3 controllers.
- Check that redis-master lists one controller as master and the other 2 controllers as slaves.
- Check that openstack-cinder-volume is started on one node.
- Check that my-ipmilan/stonith is started on all 3 controllers.
- Check that daemons corosync, pacemaker and pcsd are active and enabled.

To verify fencing behavior, the command **sudo pcs stonith fence controller-1** could be done

Use with caution: the above command will reboot the controller-1 node

The master controller will record this transaction in the `var/log/cluster/corosync.log` should further logging information be needed.

# Networking and Storage

## Physical Infrastructure

The physical infrastructure of the Ultra M Solution includes Catalyst and Nexus Switches:

- Cisco Catalyst Switches provide physical layer 1 switching for Ultra M components for the management and provisioning networks. Different models may be used, specifically Catalyst C2960XR-48TD-I or 3850-48T-S.
- Nexus Switches serve as network leafs and spines. Specifically, Nexus 93180-YC-EX typically serves as the leaf switch and Nexus 9236C serves as the spine switch.

UCS servers are always connected to the switches in the following fashion:

- UCS MLOM (Modular Lan on Motherboard) ports 1-2 connect to the Nexus leaf switches. This interface serves for Overcloud provisioning, VNF Management, and Orchestration.
- Every UCS is equipped with 2 PCIe SR-IOV cards that are used for the Service and Di-Internal network. These interfaces connect to the Nexus switches.
- Every UCS has the onboard 1G Intel port used for PXE boot from OSP-D to Compute/Ceph/Controller nodes. This port is connected to the Catalyst switch.
- Finally, every UCS is equipped with the CIMC/IPMI interface. This is server management interface used for accessing the UCS CIMC. It connects to the Catalyst switch.

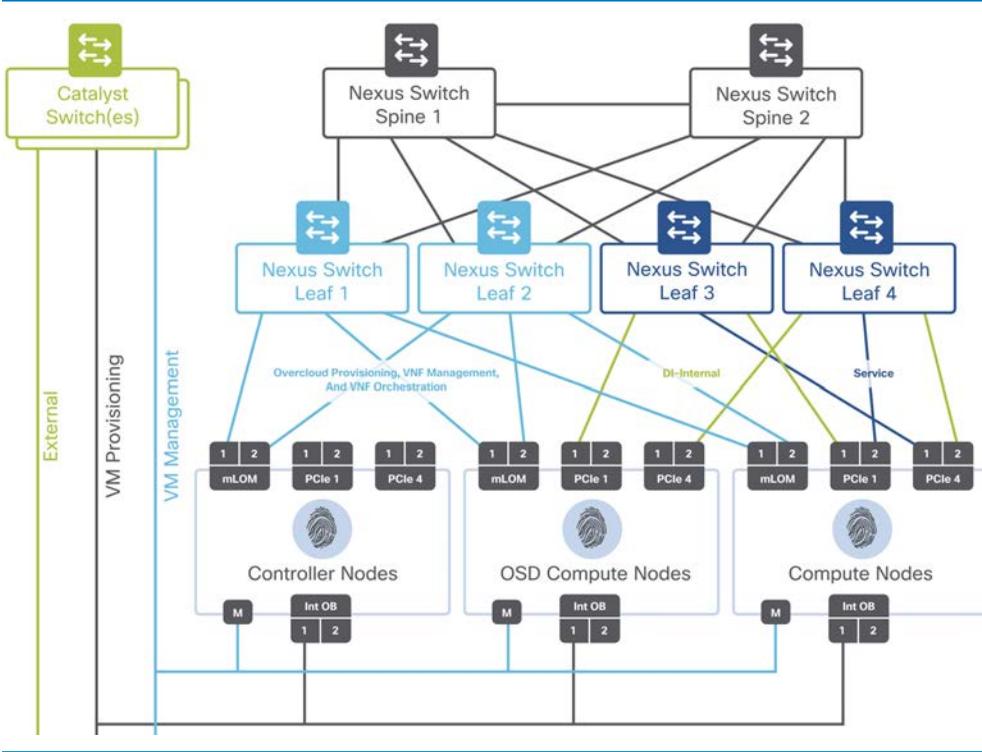
The following table provides a brief summary of interfaces and connectivity:

UCS Interface Type	Connects to	Used For
<b>IPMI</b>	Catalyst	CIMC access to the UCS
<b>Intel onboard</b>	Catalyst	PXE boot from OSP-D to the OpenStack nodes (Compute/OSD/Controller/Ceph)
<b>mLOM</b>	Nexus	Overcloud provisioning, VNF Management and Orchestration
<b>PCIe</b>	Nexus	Di Network and Service Network

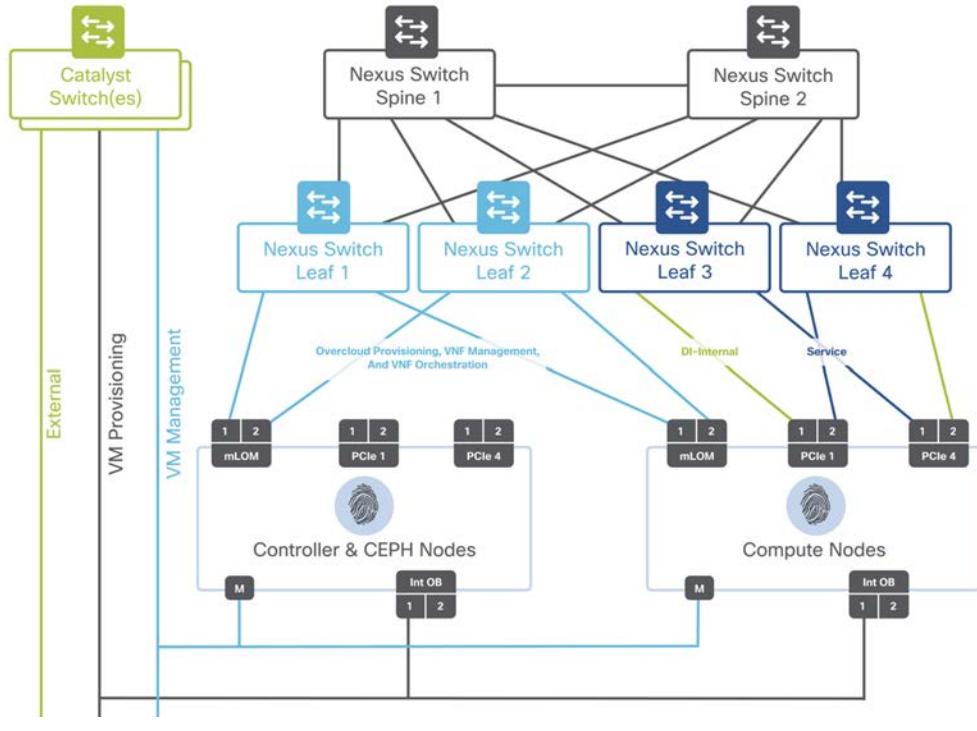
### Spine and Leaf Technology

Ultra M implements a leaf and spine network topology. The topology varies between different Ultra M deployment models, specifically, Hyper-Converged and Non-Hyper-Converged models, and with the number of the compute nodes. Please refer to the topology diagrams below:

Hyper-Converged Ultra M Single and Multi-VNF Leaf and Spine Topology:



**Non-Hyper-Converged Ultra M Model Leaf and Spine Topology:**



The spine switches have high port density and replace the core switch in the traditional three-tier core, distribution and access model. The leaf switches replace the access layer functionality, and the distribution layer is collapsed, resulting in two tiers only. Every leaf switch is attached to every spine switch. The way spine and leaf switches are connected results in every server having an identical number of physical hops to another server within the data center, resulting in predictable network latency. For detailed guidelines for interconnections between leafs and spines as well as UCS servers, refer to the Ultra M deployment guide for the appropriate software versions.

## OSP-D Undercloud and Overcloud Networking

Based on Undercloud and Overcloud networking requirements, see below the details of NIC card placement in controller, compute and storage servers.

The controller server has one MLOM with 2x10GigE ports, 1 LOM with 2x1Gb ports, 1Gb CIMC port.

The compute server has one MLOM with 2x10GigE ports, 1 LOM with 2x1Gb ports, 2 PCIe cards in slot 1 & 4 with 2x10GigE, 1 GigE CIMC

The OSD compute server has one MLOM with 2x10GigE ports, 1 LOM with 2xGb ports and 2 PCIe cards in slot 1 & 4 with 2x10GigE ports 10GigE, 1xGb CIMC port

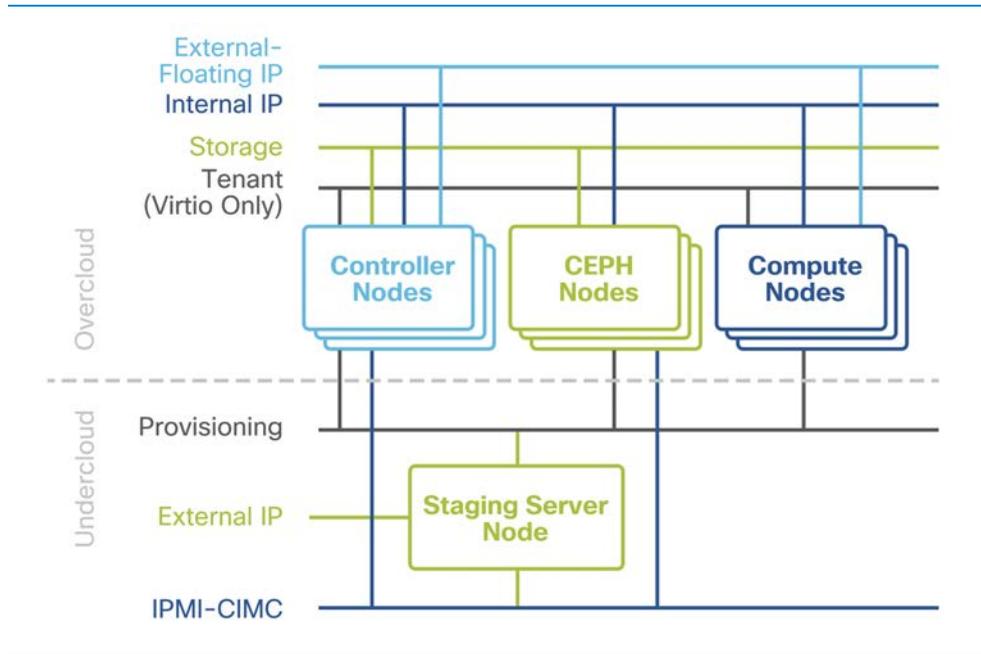
The storage server has one MLOM with 2x10GigE ports, 1 LOM with 2xGb ports, 1Gb CIMC port

### Connection Details

1. CIMC ports - used to connect the Server and IPMI enabled
2. LOM (Lan of Motherboard)
  - 1st port is used to connect to the provisioning network in all node types (controller, compute and storage)
  - 2nd port is used to connect to an external network in the controller
3. MLOM (Modular Lan on Motherboard) - 1st and 2nd ports are bonded together to form Virtio Networks in Overcloud deployment for API, External, Tenant, Storage and Storage management.
4. PCIe Slots in 1 and 4 are used to connect to StarOS, VNF Di Network and Service Network. Port 1 in slot 1 and slot 4 in port 2 are bonded together for Di Network redundancy and port 2 in slot 1 and slot 4 port 1 are bonded for the service network

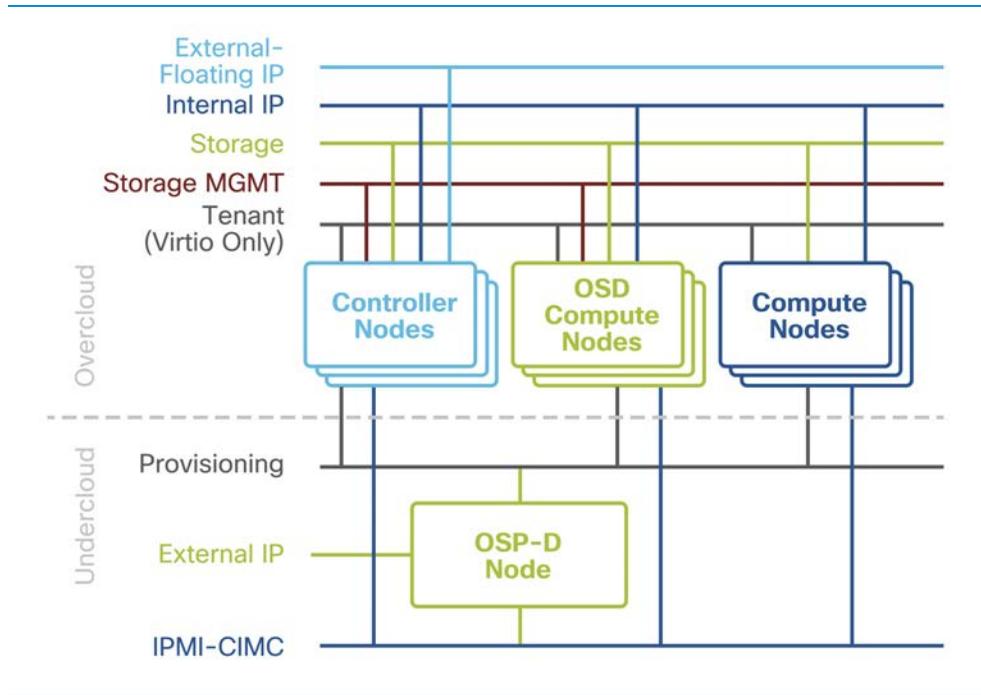
### Non-Hyper-Converged Logical Network Topology

The following diagram represents the logical network diagram for OSP-D networking:



### Hyper-Converged Network Topology

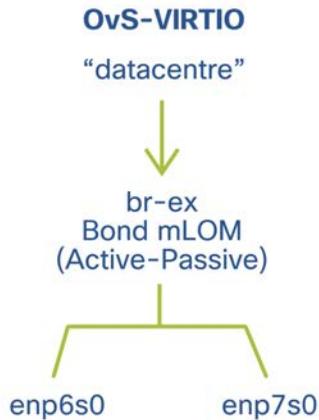
The following diagram represents the logical network diagram for OSP-D networking:



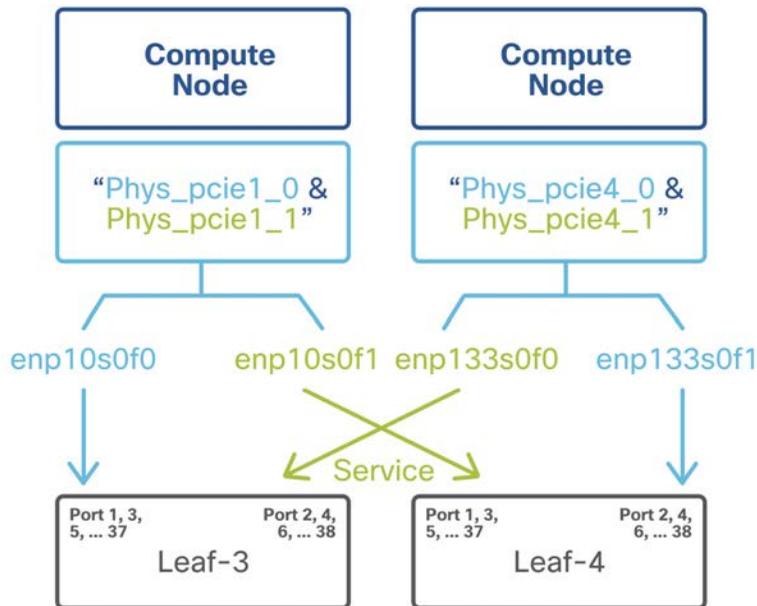
## OpenvSwitch Networking

OpenvSwitch (OVS) NIC bonding is performed in MLOM Ports in ENP6s0 and ENP7s0 using host OVS bond:

---



SR-IOV Networking is network type Flat under OpenStack configuration. NIC Bonding is used to ensure port-level redundancy for PCIe Cards involved in SR-IOV Tenant Networks. SR-IOV NIC Bonding is performed in StarOS VNF:



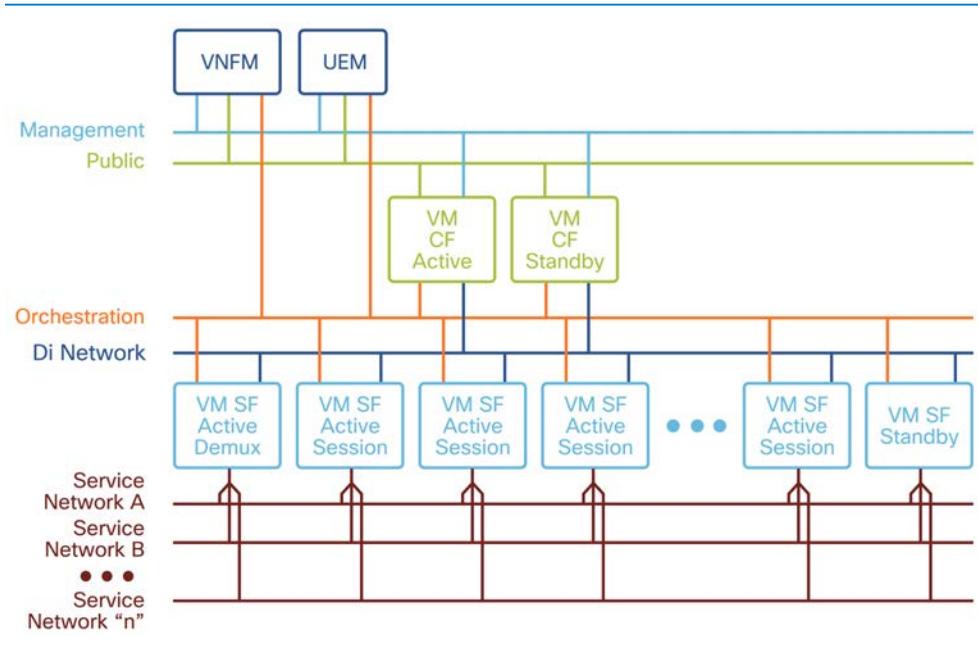
For more about SR-IOV and NIC bonding, refer to the SR-IOV portion in the Virtual Infrastructure Manager section.

### StarOS VNF Network Topology

The USP-based VNF networking requirements and the specific roles are as follows:

- **DI-Internal:** This is the DI-internal network which serves as a 'backplane/fabric' for CF-SF and CF-CF communications-created flat OpenStack network using SR-IOV.
- **Management:** This is the local management network between the CFs and other management elements such as the UEM and VNFM.
- **Orchestration:** This is the network used by VNFM (ESC) to onboard/monitor/recover the StarOS VNF.
- **Public:** The neutron router has an external gateway to the public network.

- Service: The Service Network is created as a flat OpenStack network using SR-IOV for getting traffic in and out of StarOS VNF.

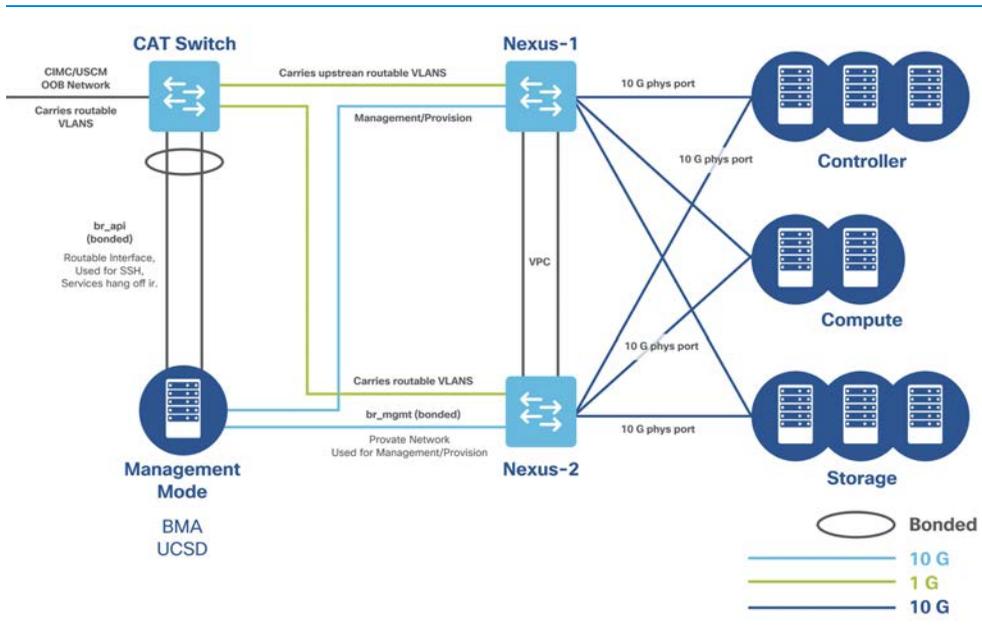


## Cisco VIM Networking

In this section, Cisco VIM networking is explained in terms of physical connectivity and the overlay network (OpenStack networking).

1. With respect to physical connectivity, the management node and all control/compute/storage nodes are connected to N9K ToR switches.

The physical topology of Cisco VIM is given below:



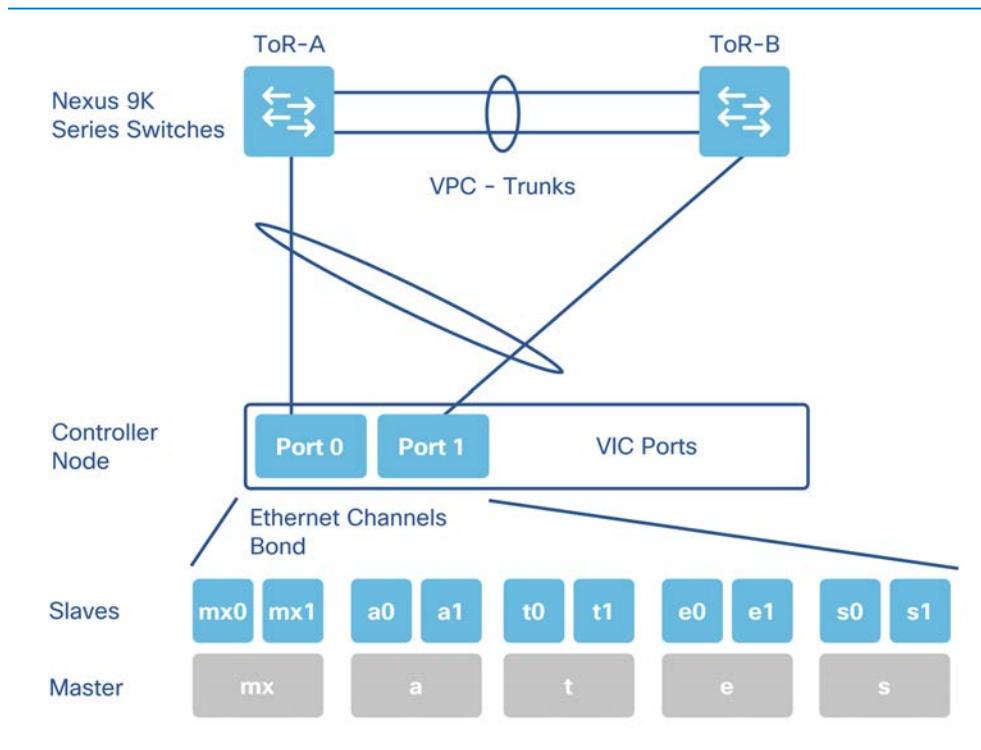
In the above diagram:

- All the 10G ports are connected to N9K switches.
- Both the N9K switches are in a single VPC domain; the 10G ports of each server are connected to both N9K switches and form port-channels.
- In the management node, 2 Linux bridges are created, `br_mgmt` and `br_api`.
  - `br_mgmt` carries all the management traffic.
  - `br_api` exists on management node as well as controller nodes. It is a routable network and provides SSH connectivity to the management node. On the

other hand, the same network is used on controller nodes to provide access to the OpenStack API network to the external world.

- Routable networks are also extended to N9K switches (via OOB switches) to service provider networks.

2. Bond mapping for controller nodes is shown below:



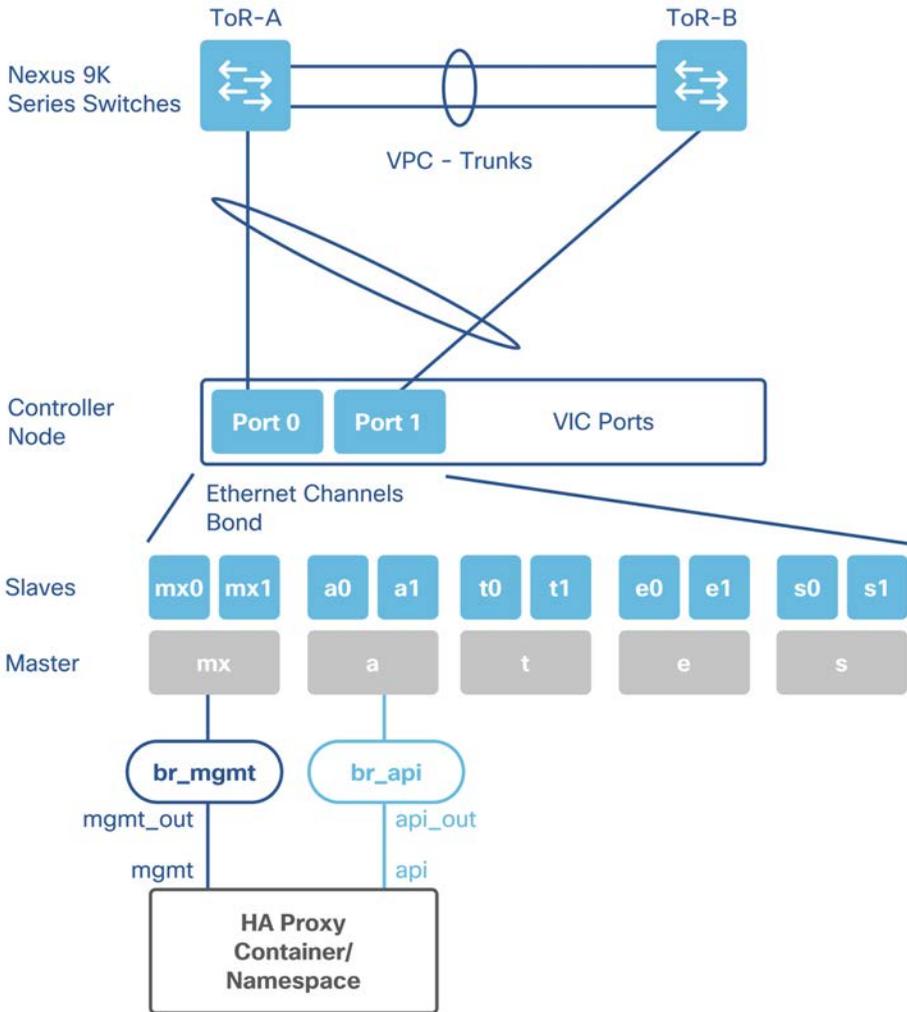
In the above diagram:

- The VIC cards are curved into multiple interfaces such as mx0 and mx1 (management and provisioning), a0 and a1 (api), t0 and t0 (tenant), e0 and e1

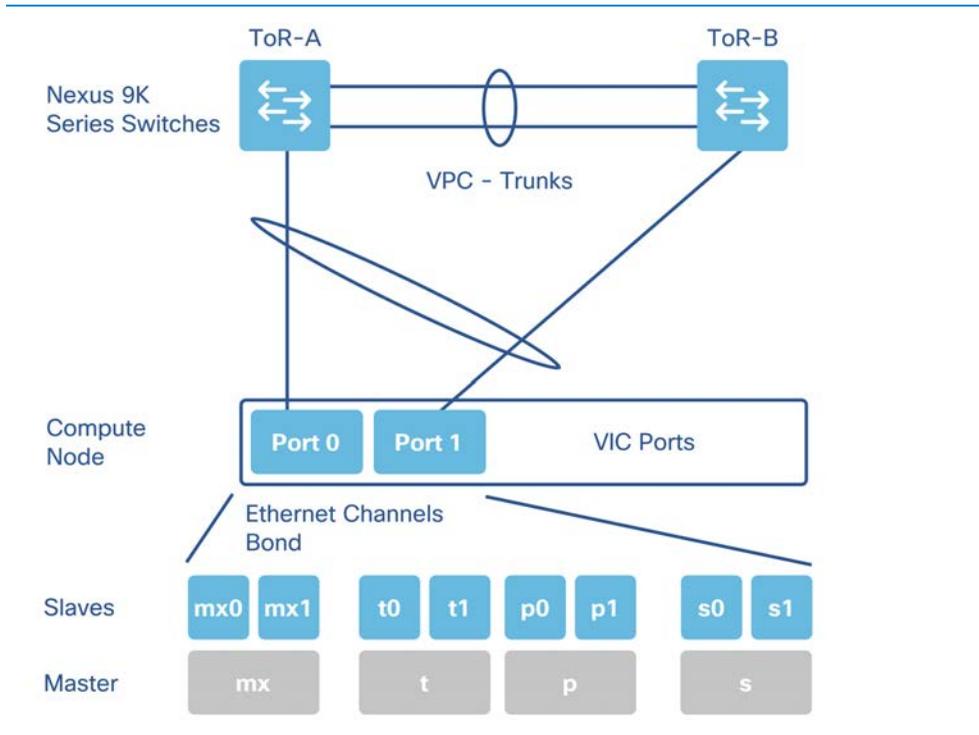
(external) and s0 and s1 (storage). There may also be p0 and p1 interfaces which belong to the service provider network.

- Once curving is done from the UCS level, all the interfaces are represented to the Operating System. From OS, the bonding feature from kernel to bond interfaces from the same category. For example, mx0 and mx1 forms a bonded interface called mx, and so on.

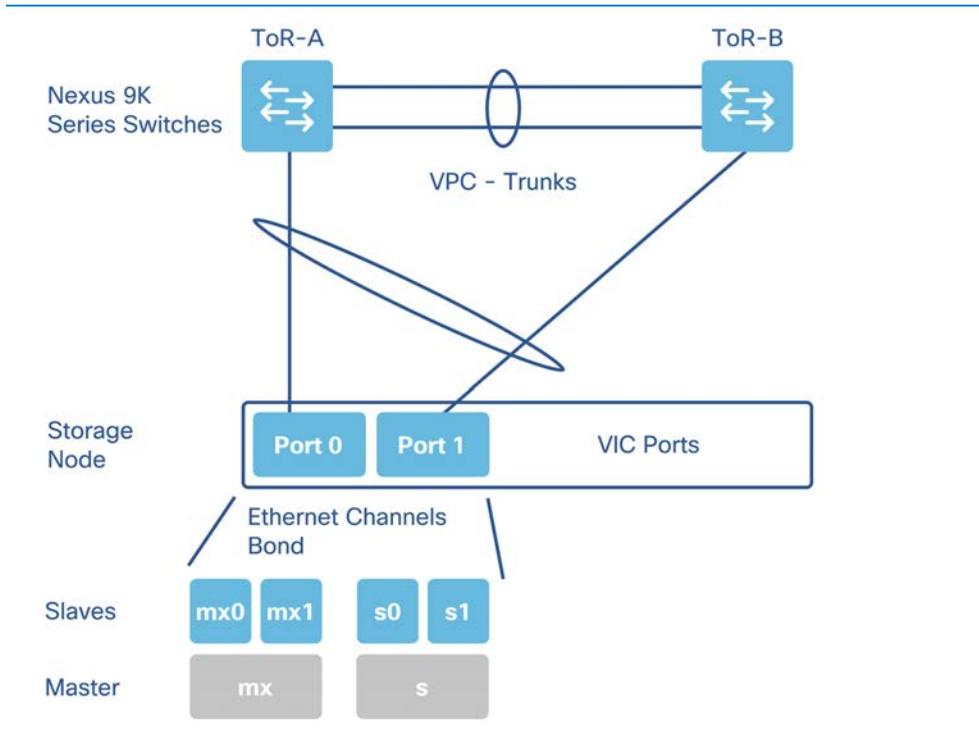
3. In this diagram, mx and a interfaces (on the controller node) are connected to br\_mgmt and br\_api bridge respectively. HA Proxy is a container running inside the controller nodes where the OpenStack API endpoint is running. br\_api bridge connects api (a) interface to the HA Proxy container:



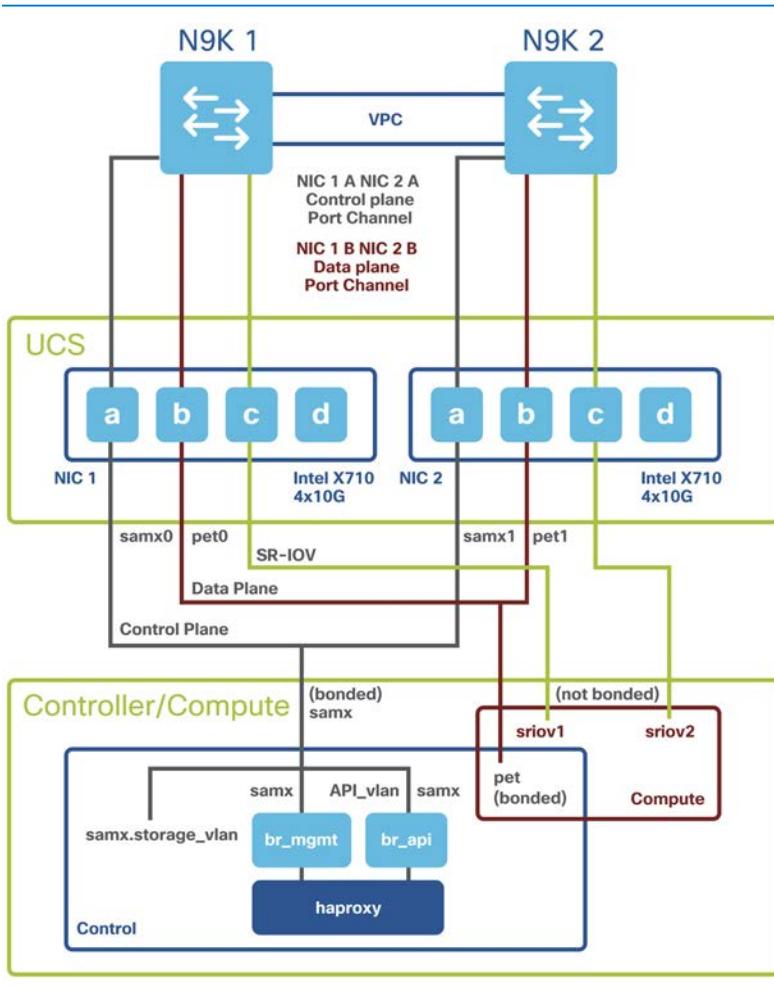
4. The bond mapping for compute nodes is shown below:



5. The bond mapping for storage nodes is shown below:



6. Intel NIC card connectivity details and network considerations:



In the diagram above:

- The Intel NIC card model is X710 with 4x10G ports.

- Port A on each card carries samx traffic ( storage, api, and management).
- Port B on each card carries pet traffic (provider, external and tenant).
- Port C and D on each card will be used for SR-IOV.
- samx interfaces are bonded together and available on controller/compute/storage nodes.
- pet interfaces are bonded together and available on controller/compute nodes.
- SR-IOV interfaces are not bonded and are available on compute nodes only.

7. There are several deployment scenarios where Cisco VIC and Intel NIC are run together.

Cisco VIC cards are used for carrying OpenStack traffic whereas the Intel NIC card is used for carrying DI-LAN traffic.

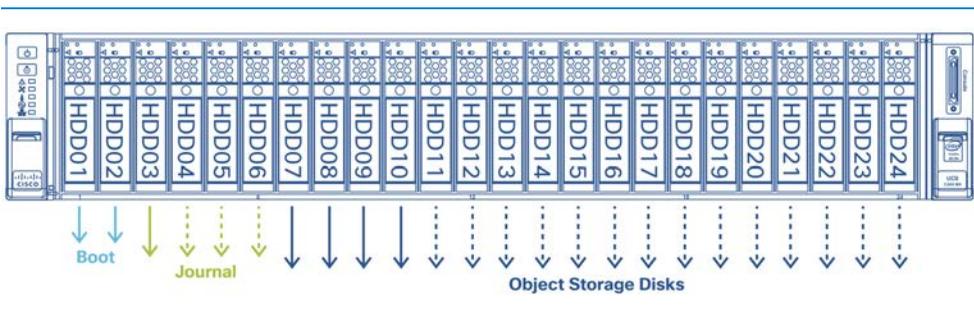
8. To check the SR-IOV networking configuration, refer the SR-IOV Networking section in "OSP-D Undercloud and Overcloud Networking".

# Storage

In the Ultra M Solution, there are specific storage requirements for disks and storage controllers.

For OSP-D and Cisco VIM deployment, please follow the Ultra M Deployment Guide and Cisco VIM installation guide respectively for information about storage requirements.

The figure below displays the storage disk layout for the UCS C240 series servers used in the Ultra M solution.



The following are Ultra M requirements for storage:

- The Boot disks contain the operating system (OS) image with which to boot the server.
- The Journal disks contain the Ceph journal file(s) used to repair any inconsistencies that may occur in the Object Storage Disks.
- The Object Storage Disks store object data for USP-based VNFs.

# OpenStack Infrastructure Deployment with UAS

## AutoDeploy/AutoIT

Ultra Automation Services (UAS) is an automation framework consisting of a set of software roles used to automate the VIM and USP-based VNF deployment as well as related components such as the VNFM. Beyond deployment automation, UAS manages software bundle components within an inventory manager. In addition, it can also be used to automate the deployment of third-party components such as NFVI/VIM, test tools, and USFs that are not part of the distributed USP software bundle.

The UAS consists of:

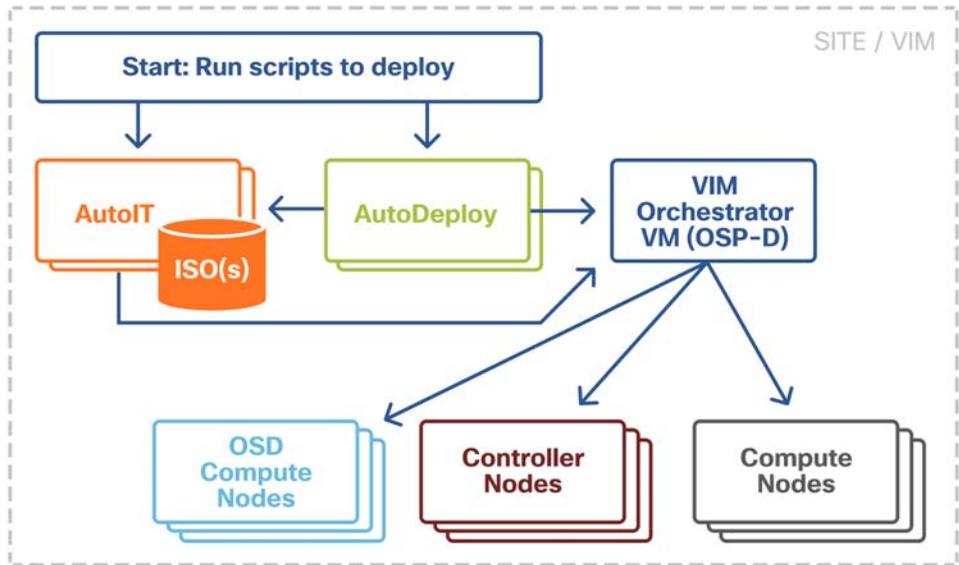
- AutoIT
- AutoDeploy
- AutoVNF

Main functionalities are:

- AutoIT
  - Undercloud deployment
  - Overcloud deployment
  - VIM provisioning
  - Ultra M Manager for system health monitoring
- AutoDeploy
  - Multi-POD deployment & operation
  - VNF software package management

- VNF software upgrade

The figure displays a high-level view of the VIM installation automation process workflow using UAS:



AutoIT is the UAS software role used to automate the process of:

- deploying the VIM Orchestrator (synonymous with the OpenStack Undercloud).
- installing the virtual infrastructure manager (VIM, synonymous with the OpenStack Overcloud) which manages the Network Function Virtualization Infrastructure (NFVI).
- onboarding/upgrading the USP ISO software package onto the Ultra M Manager Node.

AutoIT performs the deployments based on manifests it receives from AutoDeploy. Additionally, it hosts a web server to facilitate VM deployment and delivery of software packages using REST and ConfD APIs for provisioning Overcloud nodes.

AutoIT can be deployed in the following scenarios:

- as a single VM on the Ultra M Manager Node (the same physical server as AutoDeploy and OSP-D VM) during a bare metal installation.
- in high-availability (HA) mode which provides 1:1 redundancy. When deployed in HA mode, two AutoIT VMs are deployed: one active, one standby.
- as a single VM within an existing OpenStack deployment.
- in HA mode within an existing OpenStack deployment.

When supporting VIM installation automation processes, AutoIT:

- sets up AutoIT-NFVI nodes
- sets API endpoint based on ConfD to Auto-Deploy and NSO
- deploys the VIM Orchestrator
- works through the VIM Orchestrator to deploy the VIM
- brings up OSPD as a VM

## **OpenStack Infrastructure deployment with Cisco VIM**

When deploying OpenStack Infrastructure with Cisco VIM, the main functionalities of Cisco VIM are the following:

- OpenStack infrastructure deployment
- OpenStack infrastructure management (add, remove, replace, reconfigure etc.)
- OpenStack logging and monitoring aggregation, presentation
- maintaining OpenStack cloud health and recovery

Please refer to the Cisco Virtual Infrastructure Manager Installation Guide for the Cisco VIM version here: <https://www.cisco.com/c/en/us/support/cloud-systems-management/nfv-infrastructure/products-installation-guides-list.html>

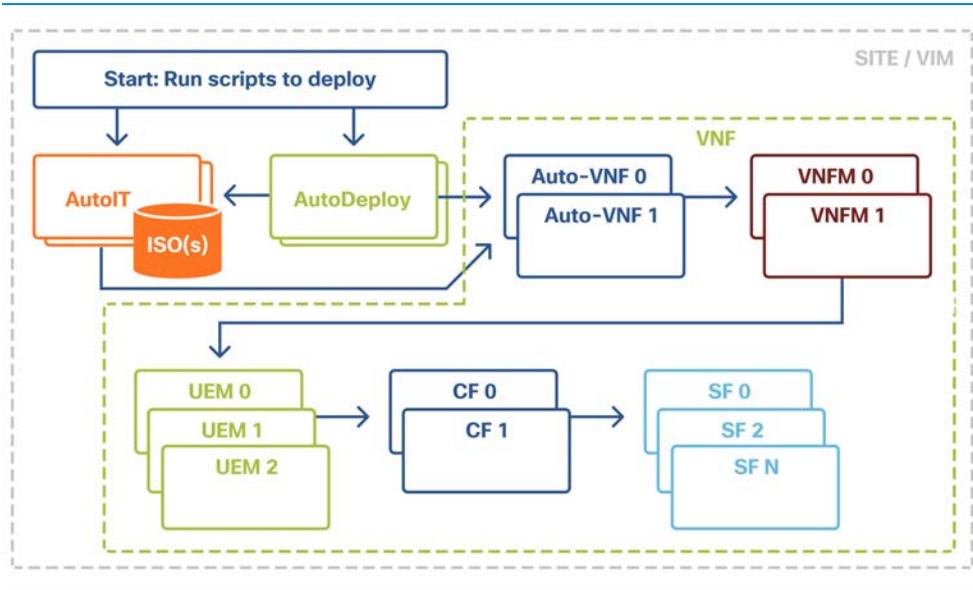
## VNF Deployment with Auto-VNF

AutoVNF performs the following:

- Multi-VNF deployment and operation
- VNF provisioning
- ESC deployment

For information about NFV constructs and how to build the deployment configuration file using NFV constructs, please follow the USP Deploy Automation Guide: <https://www.cisco.com/c/en/us/support/wireless/ultra-gateway-platform/products-installation-and-configuration-guides-list.html>

The figure below displays a high-level view of the deployment automation workflow for a single VNF. In a multi-VNF environment, AutoDeploy can deploy up to four VNFs concurrently. Additional details pertaining to the deployment automation process are provided in the deployment automation documentation.



**Note** Multi-VNF deployments are supported only in the context of the Ultra M solution.

When supporting VNF deployment automation processes, AutoIT has responsibility for:

- onboarding Ultra Automation Services (UAS) VMs.
- VIM provisioning to onboard VNFs.
- managing different version of software packages by hosting into YUM repo.
- APIs to onboard VNF packages.
- bringing up AutoVNF VMs and monitors for failures.
- storing release public key information in the ISO database for RPM signature verification by YUM through the installation process.

## AutoDeploy

AutoDeploy is the UAS software role that provides single and multi-site AutoVNF orchestration. In this context, a “site” is a single VIM instance. As such, a single AutoDeploy instance is capable of deploying the AutoVNF UAS software roles within multiple deployment scenarios:

- Single VIM/Single VNF.
- Single VIM/Multi-VNF.

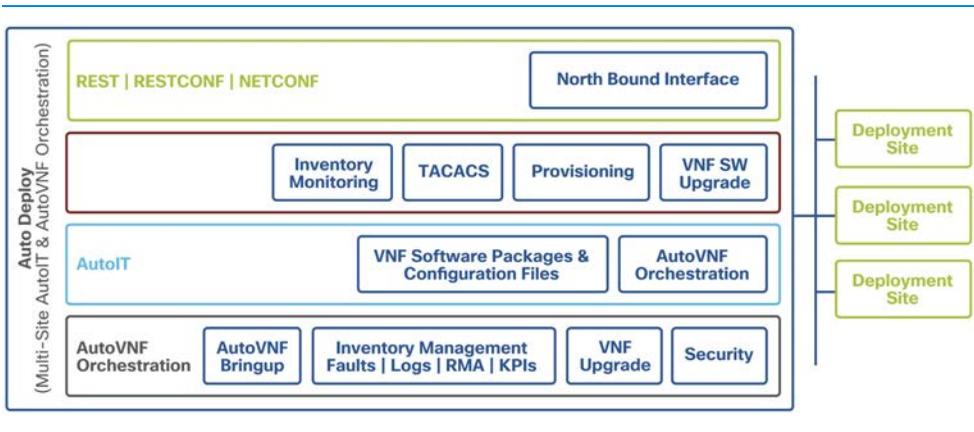
In a multi-VNF environment, AutoDeploy can deploy up to four VNFs concurrently. Additional details pertaining to the deployment automation process are provided in the deployment automation documentation.

AutoDeploy can be deployed in the following scenarios:

- As part of the VIM installation automation process:
  - on bare metal with high-availability (HA) support. HA support provides 1:1 VM redundancy. When deployed in HA mode, two AutoDeploy VMs are deployed on the same physical server: one active, one standby.
  - on bare metal without HA support. In this scenario, a single AutoDeploy VM is deployed.
- As part of an existing deployment:
  - in HA mode within an existing OpenStack deployment. When deployed in HA mode, two AutoDeploy VMs are deployed on the same physical server: one active, one standby.
  - as a single VM within an existing OpenStack deployment.
  - in this release, one AutoDeploy VM is deployed per VIM. The AutoDeploy VM must have network access to the VIM in order to provide orchestration.

Once instantiated, AutoDeploy provides the following functionality. Please refer to the diagram below:

- AutoVNFs bootstrapping and provisioning for deployments (Day-0/Day-1/Day-N).
- AutoVNF Deployments Life-Cycle including start, stop and inventory management (consolidated).
- performs release image signing validation by verifying the certificate and public key provided in the release ISO.



AutoDeploy operations are performed using any of the following methods:

- ConfD CLI and API based transactions
- WebUI based transactions

## AutoVNF

AutoVNF is the software role within UAS that provides deployment orchestration for USP-based VNFs. It does this by emulating an NFVO and VNFM for deployments.

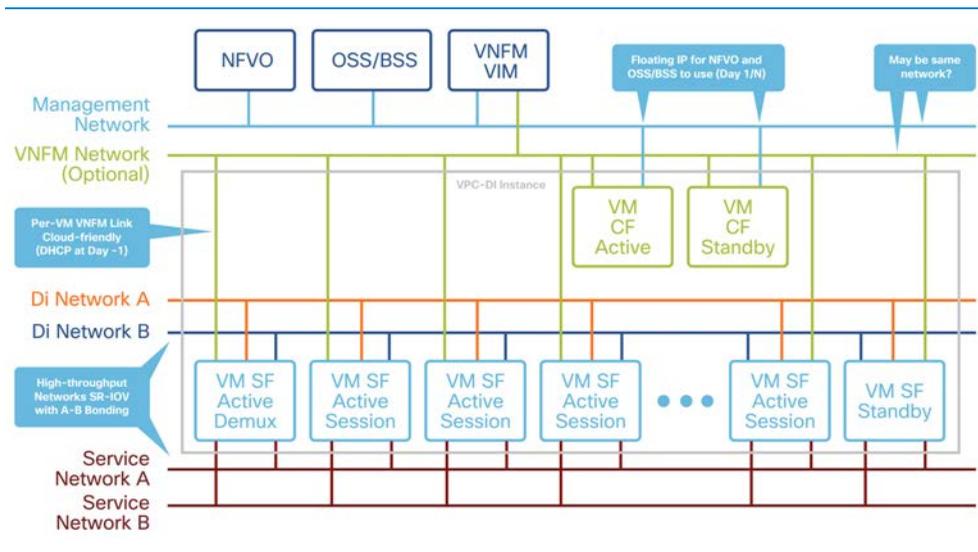
When used in Ultra M solution deployments, AutoVNF is instantiated by the AutoDeploy software role based on configuration data provided. It is deployed with a 1:1 HA redundancy model. Processes across the VMs are monitored and restarted if necessary. ConfD synchronizes the CDB between the active and standby VMs. Each of the VMs are deployed on separate compute nodes within VIM.

For VNF deployments brought up using only AutoVNF (e.g. Stand-alone AutoVNF-based deployments), only a single VM is deployed.

Once operational, AutoVNF deploys ESC which serves as the VNFM, per configurable YANG-based definitions. Please refer to the section on Elastic Service Controller for more details.

# VPC-DI VNF

A VPC-DI instance is a grouping of VMs that act as a single manageable instance of StarOS. VPC-DI consists of the following major components:



## Control Function (CF)

A central sub-system of the VNF, the CF works with the UEM to perform lifecycle events and monitoring for the VNF. Two CF VMs act as an active standby (1:1) redundant pair. The active CF is responsible for the following functions:

- Controller tasks
- Local context VPNMGR
- Local context (MGMT) and Di Network vNICs
- System boot image and configuration storage on vHDD

- Record storage on vHDD
- Out-of-Band (OOB) management (vSerial and vKVM) for CLI and logging

## Service Function (SF)

SF VMs provide service context (user I/O ports), handle protocol signaling, session processing tasks, and flow control (demux). A VPC-DI instance can contain up to 30 SF VMs.

Each SF VM dynamically takes on one of three roles as directed by the CF:

- Demux VM (flow assignments)
- Session VM (traffic handling)
- Standby VM (n+1 redundancy)

A minimum configuration for a VPC-DI instance requires four SFs - two active, one demux and one standby.

## DPDK Internal Forwarder

The Intel Data Plane Development Kit (DPDK) is an integral part of the VPC architecture and is used to enhance system performance. The DPDK Internal Forwarder (IFTASK) is a software component that is responsible for packet input and output operations and provides a fast path for packet processing in the user space by bypassing the Linux kernel. It is required for system operation. Upon CF or SF instantiation, DPDK allocates a certain proportion of the CPU cores to IFTASK depending on the total number of CPU cores. The remaining CPU cores are allocated to applications.

DPDK is a set of libraries and drivers for fast packet processing. DPDK facilitates debugging in kernel mode and this makes it easy to develop applications. DPDK Library consists of two layers:

- Environment Abstraction layer: This layer takes care of assigning cores to threads and associating PCIe devices to CPUs based on CPU Pinning and NUMA configurations explained earlier.
- Poll Mode Driver: Assigning SW-RSS queues and associating them to procllets so that appropriate processes receive packets from the network.

## Di Network

In order for the VMs within a VPC-DI instance to communicate with each other, each instance must have a private L2 network that interconnects the VMs. This network should utilize a VLAN within the IaaS/virtualization infrastructure and be exposed untagged to each VM as the first vNIC.

The Di Network must be for the exclusive use of a single VPC-DI instance. No other devices may be connected to this network.

Note: If more than one instance is instantiated within the same datacenter, each instance must have its own Di Network.

All the VMs within an instance must be physically located in the same site, ideally in the same few racks, with minimal interconnecting devices. The reliability of the Di Network is important for the stability of the VPC-DI instance.

### Network Requirements

The reliability and performance of the Di Network are critical to the reliability and performance of VPC-DI. The Di Network is used for internal control, signaling, and bearer traffic. Bearer traffic may traverse the Di Network multiple times, so any packet loss in the Di Network would impact the perceivable packet loss of VPC-DI as a whole.

Note: The infrastructure connecting the VMs should be 10 Gbps or higher between all VMs and have a redundant configuration. A redundant configuration can be provided in one of these ways:

- on the host using a vSwitch (for Virtio/VMXNET3 interfaces)
- on the hardware, such as the Cisco UCS virtual interface card (VIC)
- in the VPC-DI using network interface bonding

## Packet Flows

SF ports are used to receive and transmit bearer and signaling packets. To simplify network settings and address usage, only VLANs for high-bandwidth (bearer) packets need to be connected to all SFs. Low-bandwidth interfaces (signaling) can be connected to just two SFs. In the diagrams below, the bearer VLANs are connected to all SFs, while signaling and other VLANs are only connected to the first two SFs.

Note: This asymmetric arrangement means that fewer interfaces are needed, however careful consideration should be paid to failures since the loss of two VMs results in loss of services.

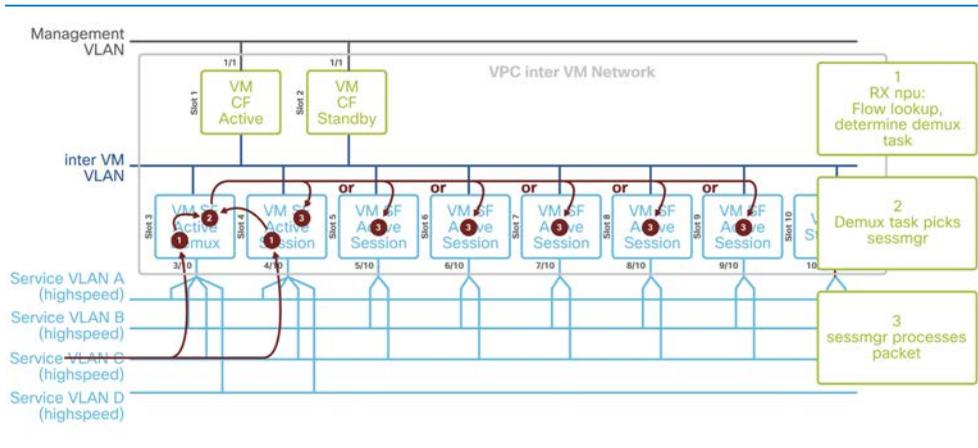
ECMP does hashing based on a hash and can send traffic to any SF VM.

On ingress, the SFs perform flow lookups and direct packets to the specific SESSMGR task on a specific SF. Some of this ingress traffic is processed by local SESSMGR tasks, otherwise, it is relayed via the Di Network to the correct SF. On egress, each SF sends out packets from its local port (provided ECMP is used). In most cases, the number of VMs that packets traverse is less than two. However, ACLs and tunneling may increase the number of hops for specific flows depending on the EPC configuration.

### **Packets Received on SF Demux VM**

On the demux and standby SF, all session traffic received is relayed to another SF for session processing. The figure below shows how ingress packets are distributed via the Demux SF to other session SFs for processing.

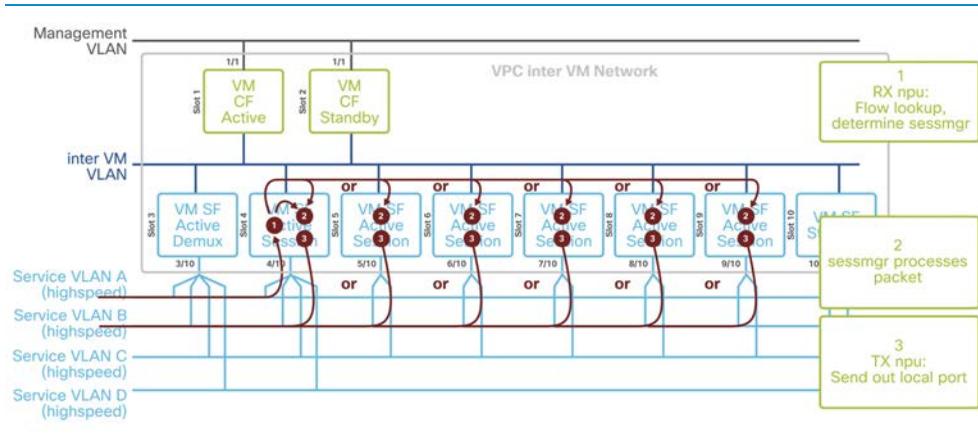
Packet Flows - SF Demux:



**Packets Received on SF Session VM**

The figure below shows how ingress packets received by a session SF are distributed to other sessions SFs for processing.

Packet Flows - SF Session:



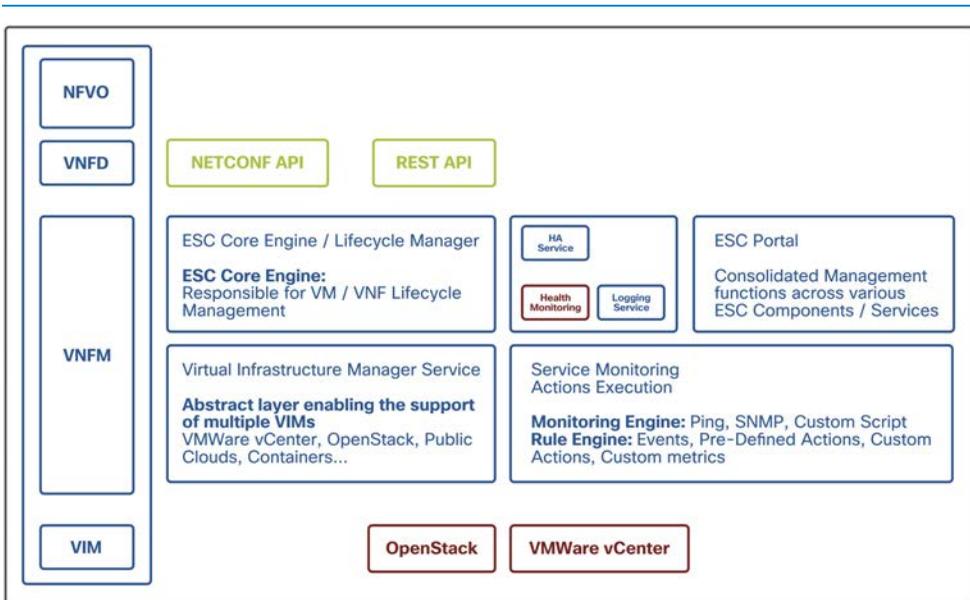
# Elastic Services Controller

## Overview

The Elastic Services Controller (ESC) acts as a VNFM (as per ETSI NFV Model). It is responsible for management of the life cycle of VNFs.

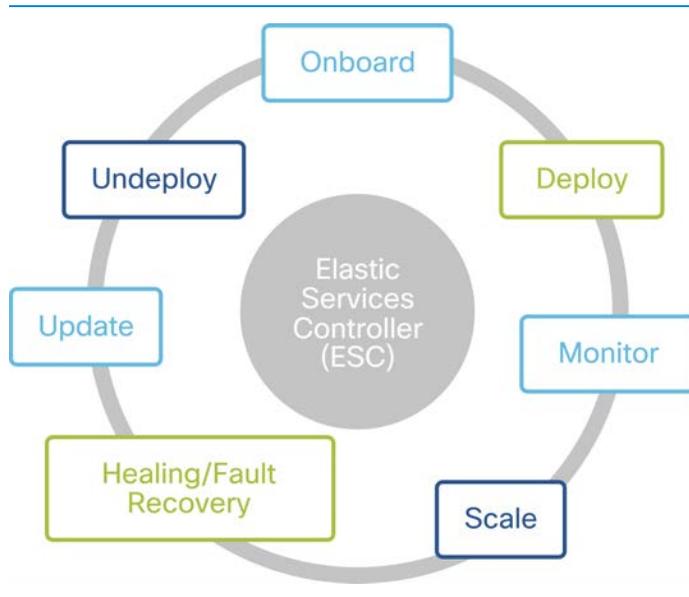
In Ultra M, ESC listens to NETCONF to communicate to Element Manager / UAS on the northbound interface, and on the southbound interface, it talks to the OpenStack API to maintain the life cycle of VNFs.

The diagram below shows the Cisco Elastic Services Controller Architecture



## ESC Life Cycle

ESC initiates the VNF deployment based on the requests received via northbound interfaces. ESC manages the entire lifecycle of the VNFs in different stages as shown below:



## ESC High-Availability Architecture

### High-Availability (HA) Overview

ESC supports high-availability in the form of a primary and standby model. Two ESC instances are deployed in the network to prevent ESC failure and provide ESC service with minimum service interruption. If the primary ESC instance fails, the standby instance automatically takes over the ESC services. ESC HA resolves the following single point failures:

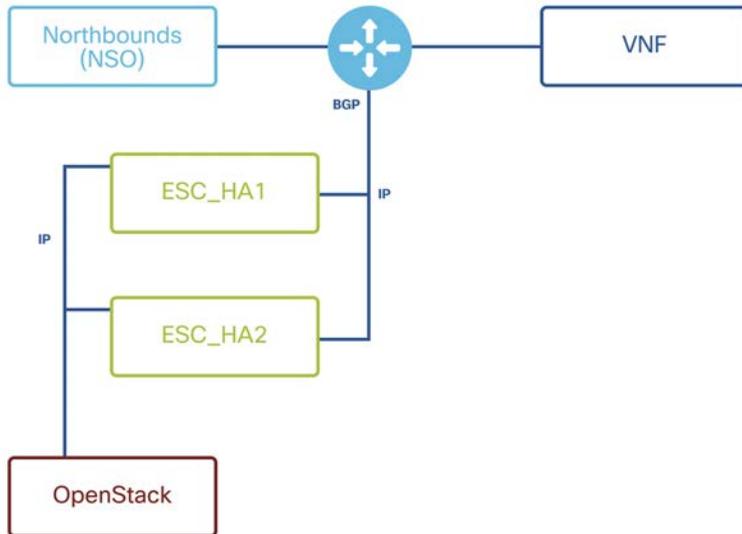
- network failures
- power failures
- dead VM instance
- scheduled downtime
- hardware issues
- internal application failures

ESC has the capability to provide high-availability in a one-to-one configuration. ESC HA deployment has two network variations and two database variations.

### Simple Network HA

The first of the network variations is the Simple Network HA. Please refer to the diagram below. In this scenario, both ESC VMs are on the same network and use keep-alive messages to check on health and determine which ESC instance is Master and which one is Backup.

Keep-alive messages run on the southbound interface of the ESC.



### How High-Availability Works

ESC HA network can be either set up as a single installation of an ESC HA pair or deployed as two standalone ESC nodes that are converted into a HA pair after re-configuring these nodes post-deployment. An HA deployment consists of two ESC instances: a primary and a standby. Under normal circumstances, the primary ESC instance provides the service. The corresponding standby instance is passive. The standby instance is in constant communication with the primary instance and monitors the primary instance status. If the primary ESC instance fails, the standby instance automatically takes over the ESC services to provide ESC service with minimum interruption.

The standby also has a complete copy of the database of the primary, but it does not actively manage the network until the primary instance fails. The KeepAliveD service monitors the activity status of both primary and standby instances. When the primary instance fails, the standby takes over automatically. The standby instance takes over the primary instance to manage the services while primary instance restoration is taking

place. When the failed instance is restored, if required, a manually-initiated switch-over can be established and network management resumed via the primary instance.

Both primary and standby ESC instances are connected to the northbound orchestration system through an IPv4 or IPv6 network. The router uses BGP or VIP routing to route Anycast IP to the primary ESC instance. For the northbound system, a unique virtual IP address is assigned to access the current primary ESC High-Availability instance. The deployed VNFs are connected to both ESC primary and standby instances through another IPv6 network.

ESC HA nodes are managed by KeepAliveD and DRBD (replication tool to keep the ESC database synchronized) sync network services. While the KeepAliveD service monitors both primary and standby instances status, the DRBD service monitors primary instance DB and syncs the changes to the standby instance DB. These two services can be co-located on the same VIP network or in two separate networks. A VM handshake between ESC instances occurs through the KeepAliveD over the IPv4 or IPv6 network.

See Cisco Elastic Services Controller Install and Upgrade Guide for more details: <https://www.cisco.com/c/en/us/support/cloud-systems-management/elastic-services-controller-esc/products-installation-guides-list.html>

## ESC Configuration Data Model

ESC is running ConfD software to maintain Config and the Operational Data Model. Exporting the ESC data model will give us a clear idea about the deployment model.

1. To dump the ESC Config Data Model, use the command below:

```
[admin@PGW-esc-esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get
esc_datamodel/tenants
Operational Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin
--privKeyFile=/home/admin/.ssh/confd_id_dsa --privKeyType=dsa --get -x
"esc_datamodel/tenants"
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:iETF:params:xml:ns:netconf:base:1.0" message-id="1">
```

```
<data>
<snipped>
```

2. In the ESC data model, there are VM placement policies such as those below:

```
<placement>
  <target_vm_group_ref>s10</target_vm_group_ref>
  <type>anti_affinity</type>
  <enforcement>strict</enforcement>
  <vm_group_ref>c1</vm_group_ref>
  <vm_group_ref>c3</vm_group_ref>
  <vm_group_ref>s2</vm_group_ref>
  <vm_group_ref>s4</vm_group_ref>
  <vm_group_ref>s5</vm_group_ref>
  <vm_group_ref>s6</vm_group_ref>
  <vm_group_ref>s7</vm_group_ref>
  <vm_group_ref>s8</vm_group_ref>
  <vm_group_ref>s9</vm_group_ref>
</placement>
```

This output above explains that S10 (SF10) is going to be deployed in a compute node where other VMs (such as CF or REST of SF) will not be deployed, and so on and so forth.

3. Each CF or SF is deployed as vm\_group:

```
<vm_group>
  <name>c1</name>
  <flavor>PGW-DEPLOYMENT-control-function</flavor>
  <bootup_time>1800</bootup_time>
  <recovery_wait_time>1</recovery_wait_time>
```

bootup\_time explains that after ESC initiates the deployment, it will wait for 1800 sec for VM to boot up.

4. The `kpi_data` section on each `vm_group` explains how each VM will be monitored from ESC:

```
<kpi_data>
  <kpi>
    <event_name>VM_ALIVE</event_name>
    <metric_value>1</metric_value>
    <metric_cond>GT</metric_cond>
    <metric_type>UINT32</metric_type>
    <metric_occurrences_true>1</metric_occurrences_true>
    <metric_occurrences_false>30</metric_occurrences_false>
    <metric_collector>
      <type>ICMPPing</type>
      <nicid>2</nicid>
      <poll_frequency>10</poll_frequency>
      <polling_unit>seconds</polling_unit>
      <continuous_alarm>false</continuous_alarm>
    </metric_collector>
  </kpi>
</kpi_data>
```

The output above shows that ESC uses ICMP Ping method on interface 2 every 10 seconds to determine if the VM is alive.

5. In the rule section of each `vm_group`, if there is no recovery policy provided, then by default it takes `REBOOT_THEN_REDEPLOY`:

```
<rules>
  <admin_rules>
    <rule>
      <event_name>VM_ALIVE</event_name>
      <action>ALWAYS log</action>
      <action>FALSE recover autohealing</action>
      <action>TRUE servicebooted.sh</action>
    </rule>
  </admin_rules>
</rules>
```

## ESC Operational Data Model

ESC Operational Data Model provides the live status of all VMs being monitored by ESC.

1. To export the ESC operational data model, use the command below:

```
[admin@pgw-esc-esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get
esc_datamodel/opdata
Operational Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin
--privKeyFile=/home/admin/.ssh/confd_id_dsa --privKeyType=dsa --get -x
"esc_datamodel/opdata"
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
<snipped>
```

2. In the beginning of opdata output, the VIM details to which ESC is communicating on the southbound interface are displayed:

```
<system_config>
  <active_vim>OPENSTACK</active_vim>
  <openstack_config>
    <os_auth_url>http://10.20.30.40:5000/v2.0</os_auth_url>
    <admin_role>admin</admin_role>
    <os_tenant_name>core</os_tenant_name>
    <os_username>core</os_username>
    <member_role>_member_</member_role>
  </openstack_config>
</system_config>
```

3. On the later section, images, flavors and respective status are shown:

```
<images>
  <image>
    <name>PGW-DEPLOYMENT-control-function</name>
    <image_id>9b49c4dc-6efc-4bca-8328-ab95685921dc</image_id>
    <public>true</public>
    <state>IMAGE_ACTIVE_STATE</state>
  </image>
```

```

<flavors>
  <flavor>
    <name>PGW-DEPLOYMENT-control-function</name>
    <flavor_id>e45588e0-1b79-4e4b-b7aa-67b23e44f1df</flavor_id>
    <public>true</public>
    <state>FLAVOR_ACTIVE_STATE</state>
  </flavor>

```

4. As part of ESC opdata model, there are 2 deployment models and respective service states available.

The first deployment model includes all CF and SF VMs and second deployment model consists of EM VMs

- Deployment Model for CF/SF VMs:

```

<deployments>
  <deployment_name>PGW-DEPLOYMENT-1.0.0-1</deployment_name>
  --
  <state_machine>
    <state>SERVICE_ACTIVE_STATE</state>
    <vm_state_machines>
      <vm_state_machine>
        <vm_name>PGW-DEPLOYM_c1_0_7a8e098f-f107-4b67-8363-
c5fe7fa6e442</vm_name>
        <state>VM_ALIVE_STATE</state>
      </vm_state_machine>
      <vm_state_machine>
        <vm_name>PGW-DEPLOYM_s3_0_61c9bfa2-3706-4681-a30e-
c42f7e3f957e</vm_name>
        <state>VM_ALIVE_STATE</state>
      </vm_state_machine>
    --
  --

```

- Deployment Model for EM VMs:

```

<deployments>
  <deployment_name>PGW-DEPLOYMENT-em</deployment_name>

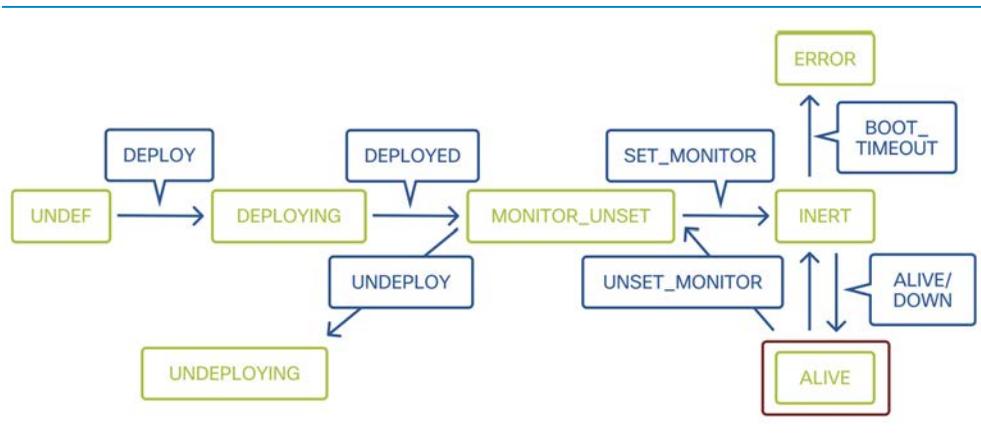
  --

  <state_machine>
    <state>SERVICE_ACTIVE_STATE</state>
    <vm_state_machines>
      <vm_state_machine>
        <vm_name>PGW-DEPLOYM_DAPGW1_0_8697dc79-64ec-4b68-9080-
a54b95470dd9</vm_name>
        <state>VM_ALIVE_STATE</state>
      </vm_state_machine>
    </vm_state_machines>
  </state_machine>

  --
  
```

## VNF Health Monitoring

As part of ESC Life Cycle management, the VNF goes through several states. The diagram below demonstrates different VNF states.



The green boxes are identified as different VM states such as VM\_UNDEF\_STATE or VM\_INERT\_STATE. The blue boxes explain the event triggered with every state change of VMs such as VM\_DEPLOY\_EVENT or VM\_DEPLOYED\_EVENT.

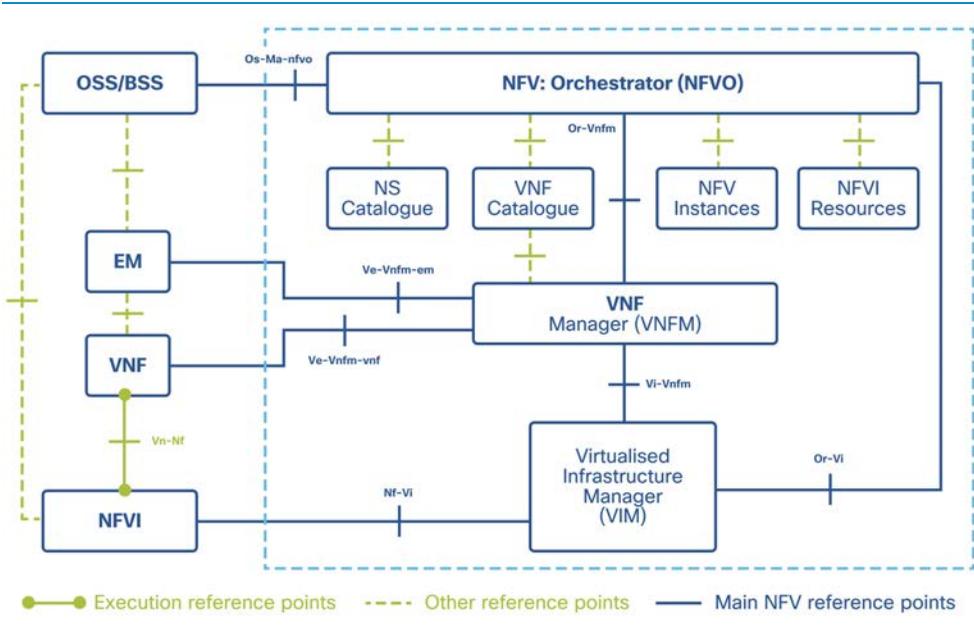
## **ESC Troubleshooting**

Please check the Virtual Network Function Manager and Element Manager section in Troubleshooting chapter for ESC troubleshooting.

# Element Manager

## Overview

Element Manager is one of the functional blocks as defined in the ETSI MANO Specification. EM is not part of the MANO itself, however, it exchanges information with it over the reference point called VE-VNFM-EM. Please refer to the diagram below, image source: <http://www.etsi.org>



VE-VNFM-EM reference point serves as an exchange point between the VNF Element Manager and the VNF Manager. VNF Manager in the Ultra M Solution is Cisco Elastic Services Controller (ESC).

This reference point, along with VE-VNFM-VNF reference point, that provides communication between the VNF and VNF Manager, is used for the lifecycle operations, such as:

- VNF instantiation.
- VNF termination.
- VNF scaling operations.
- VNF configuration.

Additionally, EM performs some of the FCAPS (fault, configuration, accounting, performance and security management) functions.

It is important to note that VNF Manager (Cisco ESC in Ultra M solution) will also be performing lifecycle and FCAPS functions. However, it is to be noted that EM takes care of the functional components while VNFM takes care of the virtual infrastructure part of the VNF.

What does that mean for the VNF?

- 1 If there is an issue with spinning up a VNF - it will be notified by the VNFM (ESC).
- 2 If there is an issue with the Mobile Core function itself - it will be notified by the EM.

## **EM Architecture**

In the ETSI MANO architecture, the EMS interfaces with the network's management systems (northbound), the VNF (southbound), and the VNFM (eastbound).

The UEM provides the following network management functions:

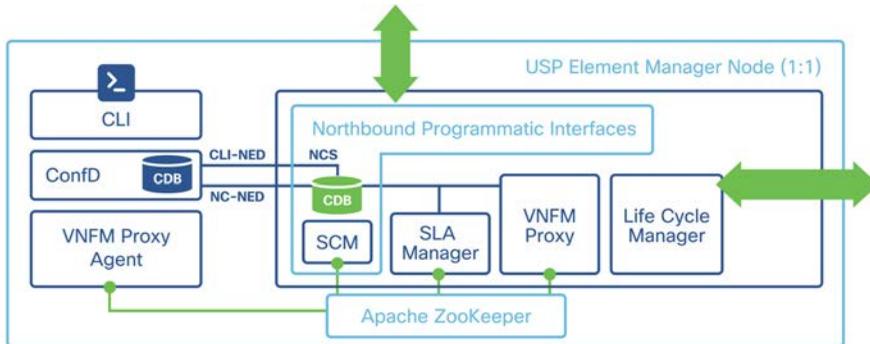
- Configuration
- Fault management

- Usage accounting
- Performance measurement
- Security management
- Operational state of VNF

In the Cisco Ultra Services Platform, the EM is embedded as part of the VNF and is split into two components:

- 1 StarOS-based control function (CF) runs as VM.
- 2 VM to host SCM, SLA manager, and VNFM Proxy.

The diagram below provides an overview of the two functional components:



USP-EM presents the Northbound Interface (NBI) to take care of the configuration and management functions of VNF. Service Configuration Manager (SCM) is used for that. SCM performs the validation and passes the configuration to the ConfD on the CF.

CF runs the ConfD and NETCONF will be used as the communication interface between the CF and EM(SCM).

Lifecycle Manager (LCM): The LCM exposes a single and common interface to the VNF (Ve-Vnfm) that is used for performing life-cycle management procedures on a VNF.

In order to provide Service Level Agreement (SLA) management, the system should notify VNF-specific faults to the EM. The SLA Manager (SLA-M) enables service assurance by building a framework to monitor specific NFV primitives including NFVI resources, application-specific resources and platform specific resources and aggregates all VNF Component logs in a central location. The SLA-M ensures SLAs are met by taking actions on events, KPI thresholds etc. At the time of writing this book, SLA Manager is not yet fully implemented.

The VNFM-Proxy is a component placed between the VNFM and the VNF to abstract the VE-VNFM reference point from the VNFM. The VNFM-Proxy is a VDU inside the VNF-EM.

CF will be taking care of the internal management of all VDUs involved in constituting the VNF.

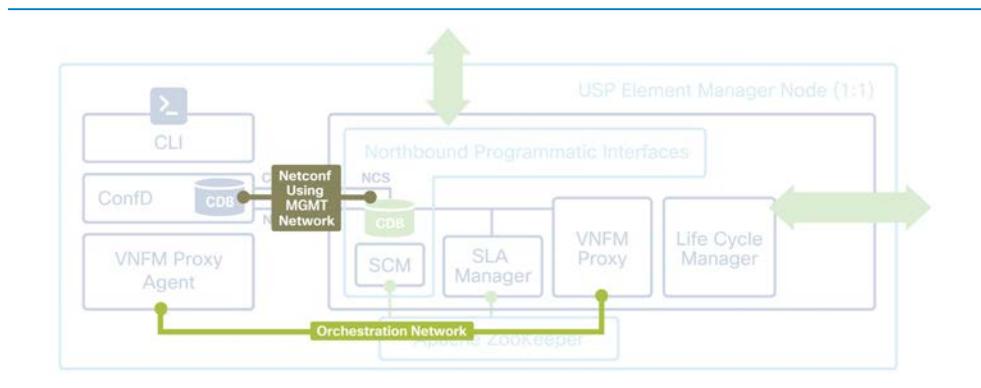
CF will install and configure each VM with predefined artifacts and then start the applications on the VM (same as for VPC-DI solution). EM will manage the full lifecycle of VNF from onboarding, through instantiation, configuration, scaling in/out, starting/stopping, to termination.

## Communication between EM and CF

EM and CF have two different connections:

- 1 NETCONF connection between EM and CF mgmt VIP. This is using the management network.
- 2 Connection over the ZooKeeper framework between VNFM Proxy Agent (CF) and VNFM proxy (EM). This is using the orchestration network.

The diagram below provides an overview of the two functional components:



On the CF side, important functions introduced here are:

- VNFR: VNF records contain the operational state of the different elements from a cloud perspective (networks/vms/ports).
- CONFD manager: this process provides the NETCONF connection to the CDB which contains the configuration. CF can send the VNFR to the EM.

Status of the confd manager must be started and can be checked with the **show confd manager** command:

```
[local]UGP# show confdmgr

State Information
-----
State           Started
Subscriptions   29
Last successful id 1520-5153-17163
Last failed id  None
Autosave url    Not configured
Username         admin
```

To verify the ConfD CDB content is possible, the command **show confdmgr confd cdb** will provide cdb information.

Access to the ConfD is also possible from debug shell in local context (servr 1 bash) by executing **"/usr/confd/bin/confd\_cli -u admin -C"**.

VNFM\_PROXY Agent: this process on CF will speak to the VNFM proxy on the active EM. This process will be informing EM VNFM proxy about events that are to be forwarded to the ESC, such as StarOS-initiated start/stop/reboot for the particular VM. This process will also receive faults from the NFVI which are reported by the ESC (proxied from the EM towards the CF).

The vnfmproxy agent contains a ZooKeeper client, and it will use ZK database transactions using JSON RPCs towards the ZooKeeper framework on EM.

To verify the status of the VNFMPProxy Agent, the following command may be used:

```
[local]UGP# show vnfmp-proxy-agent status
VNFM Proxy Agent Status:
  State       : online
  Connected to : 172.16.180.5:2181
  Bind Address : 172.16.180.23:51539
VNFM Proxy address count: 3
```

To check the state on the EM side:

```
admin@scm# show ems

EM          VNFM
ID  SLA  SCM  PROXY
-----
5   up   up   up
8   up   up   up
```

In the sample below, the Master EM has the IP address 172.16.180.5 (can be easily obtained by **ip add show**).

To check the status from the EM side, verify that operational state is enabled:

```
admin@scm# show devices device state

                OPER
                STATE
NAME           OPER  ERROR  LAST
STATE         TAG   TRANSACTION MODE  TRANSACTION ID
-----
scm-cf-nc     enabled -      lock-reset-candidate 1520-5153-17163
```

EMCTRL: Element Manager control task is in-charge of translating the events received by vnfmp proxy task into actionable StarOs processes (for example, card down).

The following output from the system shows the basic task information within the CLI **show task resources**:

2/0	vnfma	0	0.09%	15%	18.23M	130.0M	14	500	--	--	-	good
2/0	emctrl	0	0.10%	15%	13.72M	40.00M	16	500	--	--	-	good
2/0	confdmgr	0	0.11%	25%	15.98M	40.00M	21	100	--	--	-	good

As noted here, we can see they are hosted on the CF function (represented as card 2 in this case).

The state of the process can be checked by using the following command:

```
[local] UGP# show emctrl status
emctrl status:
emctrl in state: ALIVE
```

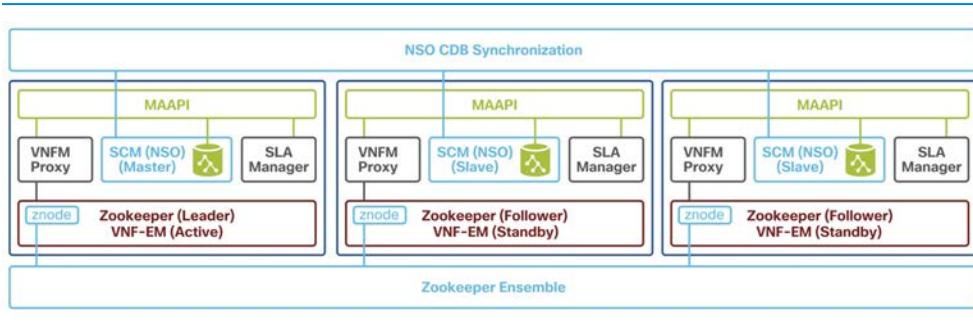
## EM Redundancy

Element Manager requires three (3) VMs hosted on the three different hosts to provide full redundancy.

Apache ZooKeeper is used to form the VNF-EM HA cluster. VNF-Proxy creates and monitors different nodes in the cluster. Please refer to the diagram below.

EM uses CDB (Configuration Datastore) to store and synchronize the configuration. It is the same CDB used in NSO Solution. CDB also manages relationships between services and devices and can handle multiple revisions of device interfaces.

CDB (MAAPI) interface is used internally between processes to communicate configuration transactions. The NETCONF is used to communicate fault states.



The leader election is done by using Lowest IP address (VM with the lowest IP is chosen to be the ZooKeeper Leader).

SCM will be taking Master or Slave role. VNFm proxy sets the SCM role to master or slave and ideally, the SCM Master will be on the same node where ZooKeeper Leader runs.

The third node is not practically used and it is present for ensuring ZooKeeper Ensemble.

## EM Recovery

During the deployment, the EM deployment configuration is set on the AutoDeploy node. ESC will be monitoring the EM VM status. Based on set configuration, the keepalive and recovery mechanism will be set. In the sample below, keepalive is set to 10, which means that the ESC will probe EM every 10 seconds. AutoVNF sets EM VM recovery parameters to REBOOT\_THEN\_DEPLOY in ESC.

```
vnfd ultram-vnfd1
vnf-rackd ultram-vnf-rack1
vnf-package vnf-pkg1
vdu-catalog element-manager
login-credential em-login
ha-type one-to-one
health-check-frequency 10
```

```
health-probe-max-miss 6
recovery-type          recovery-restart
volumes em-volume
```

Hence, if a compute node fails, the following will happen:

- 1 EM HA model is designed across 3 nodes, hence a single fault will not impact functionality. EM will continue to work.
- 2 If the host recovers within the recovery window, Auto-recovery will bring EM-VM using “reboot” policy.
- 3 If not, Auto-Recovery will deploy EM-VM on an available host in the same zone.
- 4 If no host is available, EM-VM will be moved into an error state. At this stage, use ESC-provided API/CLI to manually bring up EM-VM at a later time.

## Deployment ID, UUID and their relationship in EM

There are three major components – ESC, EM, and StarOS VNF - behind OpenStack (in OSP-D and in Cisco VIM). EM, in a way, proxies for the ConfD queries and sends responses on behalf of the StarOS VNF. Each of these components runs as a VM and maintains information. When all three don't match, there is a UUID mismatch alarm in EM.

ESC makes a YANG call to EM to get ConfD data. ConfD has both configuration information and operational data/state. EM translates, as needed, the queries that come from ESC and sends responses as needed.

The following are examples of messages sent from ESC to EM:

```
13:51:14,721 21-Mar-2018 INFO Deployment ID: 3783a2d9-c0dc-43c7-b9f3-5739c5d52402
13:51:14,721 21-Mar-2018 INFO Deployment name: Lab1-epdg-1.0.0-1
13:51:14,722 21-Mar-2018 INFO VM group name: v2
13:51:14,722 21-Mar-2018 INFO VM ID: d00e543b-4a09-4beb-a775-5a9b040a6cca
13:51:14,722 21-Mar-2018 INFO VM Name: Lab1-epdg-1.0._v2_0_694278c6-91f2-4cba-
a9fb-b7bd804acf20
13:52:41,097 21-Mar-2018 INFO Deployment ID: 3783a2d9-c0dc-43c7-b9f3-5739c5d52402
```

```

13:52:41,097 21-Mar-2018 INFO Deployment name: Lab1-epdg-1.0.0-1
13:52:41,098 21-Mar-2018 INFO VM group name: v3
13:52:41,098 21-Mar-2018 INFO VM ID: 0efda5f2-80fd-4469-b233-048a3f032c37
13:52:41,098 21-Mar-2018 INFO VM Name: Lab1-epdg-1.0._v3_0_705643ac-9fa4-45c5-
96a1-16d0d7538245

13:53:21,911 21-Mar-2018 INFO Deployment ID: 3783a2d9-c0dc-43c7-b9f3-5739c5d52402
13:53:21,911 21-Mar-2018 INFO Deployment name: Lab1-epdg-1.0.0-1
13:53:21,911 21-Mar-2018 INFO VM group name: v4
13:53:21,911 21-Mar-2018 INFO VM ID: 8669a231-f7cc-4231-a48c-200b1dbd565d
13:53:21,911 21-Mar-2018 INFO VM Name: Lab1-epdg-1.0._v4_0_83ae77cf-64b0-48e3-
a5c1-e4c5484bafb3

13:54:53,143 21-Mar-2018 INFO Deployment ID: 3783a2d9-c0dc-43c7-b9f3-5739c5d52402
13:54:53,143 21-Mar-2018 INFO Deployment name: Lab1-epdg-1.0.0-1
13:54:53,143 21-Mar-2018 INFO VM group name: v5
13:54:53,143 21-Mar-2018 INFO VM ID: 59b659f9-259b-42fb-a169-4b5d32a1df00
13:54:53,143 21-Mar-2018 INFO VM Name: Lab1-epdg-1.0._v5_0_95a89128-e9f9-442a-
bdb5-fdc3e8c2f6c9

13:56:28,879 21-Mar-2018 INFO Deployment ID: 3783a2d9-c0dc-43c7-b9f3-5739c5d52402
13:56:28,879 21-Mar-2018 INFO Deployment name: Lab1-epdg-1.0.0-1
13:56:28,879 21-Mar-2018 INFO VM group name: v6
13:56:28,879 21-Mar-2018 INFO VM ID: 1312fdd8-e4c3-40ec-a49a-b87d789b4f45
13:56:28,879 21-Mar-2018 INFO VM Name: Lab1-epdg-1.0._v6_0_48f9a878-99b8-4ab8-
8007-31e4ca03b700

13:58:03,384 21-Mar-2018 INFO Deployment ID: 3783a2d9-c0dc-43c7-b9f3-5739c5d52402
13:58:03,384 21-Mar-2018 INFO Deployment name: Lab1-epdg-1.0.0-1
13:58:03,384 21-Mar-2018 INFO VM group name: v7
13:58:03,384 21-Mar-2018 INFO VM ID: 7452d5dd-2ccb-406a-be04-e96fffc9ce6d
13:58:03,384 21-Mar-2018 INFO VM Name: Lab1-epdg-1.0._v7_0_2b9ae851-2406-45bd-
b8b9-73448613bd79

13:58:46,182 21-Mar-2018 INFO Deployment ID: 3783a2d9-c0dc-43c7-b9f3-5739c5d52402
13:58:46,182 21-Mar-2018 INFO Deployment name: Lab1-epdg-1.0.0-1
13:58:46,182 21-Mar-2018 INFO VM group name: v1
13:58:46,182 21-Mar-2018 INFO VM ID: 3031aa32-d667-4922-8d7e-fc5a6739fe60
13:58:46,182 21-Mar-2018 INFO VM Name: Lab1-epdg-1.0._v1_0_9b98416e-95cf-470d-
9ec4-f8fe247cff1f

```

The same information can be seen in StarOS side using commands:

```
[local]EM-LAB# show card hardware | grep -i uuid
UUID/Serial Number      : 6EF5BF1C-25D8-4C7E-862C-3E87A2371848
UUID/Serial Number      : CAF9E049-8739-483F-9C18-9840EE23EA8E
UUID/Serial Number      : 308B0303-F665-4368-9640-679D7FE13189
UUID/Serial Number      : 546A6194-1B99-448F-8FE0-0CC54C42C154
UUID/Serial Number      : CAEB56DA-86D4-41D3-942E-0CE2866A6EBC
UUID/Serial Number      : 085BBF4E-D87D-491D-AFA6-0398DA6E7231

show emctrl vdu list

show vdu summary
Number of Instances: 2
-----
control-function:
    BOOT_generic_di-chassis_CF2_1
    BOOT_generic_di-chassis_CF1_1
session-function:
    BOOT_generic_di-chassis_SF1_1
    BOOT_generic_di-chassis_SF2_1
    BOOT_generic_di-chassis_SF4_1
    BOOT_generic_di-chassis_SF3_1
-----

EM-LAB# show vdu detail type control-function instance BOOT_generic_di-
chassis_CF2_1
vdu-id: control-function, vdu-instance: BOOT_generic_di-chassis_CF2_1, state:
from:Invalid to:Alive
card_number: 2, card_type: 0x40010100, uuid:caf9e049-8739-483f-9c18-9840ee23ea8e
networks:
cp-id: di_intf1, state: Alive, type: unknown
vl: vl-di-internall vnfc: cf-vnfc-di-chassis
mac: fa:16:3e:2a:d0:06, ip: 192.168.1.16
cp-id: orch, state: Alive, type: unknown
vl: vl-orchestration vnfc: cf-vnfc-di-chassis
mac: fa:16:3e:33:15:73, ip: 172.16.180.12
cp-id: mgmt, state: Alive, type: unknown
vl: vl-management vnfc: cf-vnfc-di-chassis
mac: fa:16:3e:09:a9:46, ip: 172.16.181.21

Show vdu detail type session-function instance BOOT_generic_di-chassis_SF1_1
vdu-id: session-function, vdu-instance: BOOT_generic_di-chassis_SF1_1, state:
from:Invalid to:Alive
card_number: 3, card_type: 0x42020100, uuid:308b0303-f665-4368-9640-679d7fe13189
```

```

networks:
  cp-id: di_intf1, state: Alive, type: unknown
    vl: vl-di-internall vnfc: sf-vnfc-di-chassis
    mac: fa:16:3e:76:b6:a1, ip: 192.168.1.12
  cp-id: orch, state: Alive, type: unknown
    vl: vl-orchestration vnfc: sf-vnfc-di-chassis
    mac: fa:16:3e:d3:70:21, ip: 172.16.180.16
  cp-id: svc_intf1, state: Alive, type: unknown
    vl: vl-service-network1 vnfc: sf-vnfc-di-chassis
    mac: fa:16:3e:b9:89:30, ip: 10.10.10.4

```

If the information in StarOS and the information seen in EM/ESC do not match, then there will be a UUID mismatch error. It is therefore important to check the details seen in the CLI outputs to avoid the error.

## Monitoring EM

ESC talks to EM on TCP port 830. This can be seen from the following CLI output:

Port 830 is used by NETCONF/YANG. SSH protocol is used in interaction for securing the communication between ESC and EM, but after the connection is established at SSH level, a Web Services Application will interact with StarOS VM. Any CLI type commands won't work because it is all in the NETCONF/programmability interface. The connection established from the EM VM is shown below:

```

ubuntu@ultramvnm1em-0:/opt/cisco/em$ netstat -an | grep 830
tcp        0      0 172.16.181.7:52257    172.16.181.9:830     ESTABLISHED
tcp6       0      0 172.16.180.3:43534   172.16.180.11:830    ESTABLISHED
tcp6       0      0 172.16.180.3:43533   172.16.180.11:830    ESTABLISHED

```

Typically, a YANG script or other YANG tools are re-used to check whether EM is responding to queries. An Ansible script can be used for monitoring.

# Troubleshooting

# Troubleshooting Theory

## Overview

Many complex problems are consequences of changes in the network. It could be a configuration change, topology change, traffic pattern change, subscriber behavior changes, signaling storms, or flapping of different components. Any problem can result in some KPI degradation and show an anomaly in single or various components or counters.

Prior to troubleshooting, it is most important to get the broadest possible understanding of the problem and to try to evaluate all possible changes in the network, platform, and design. The challenge here is that many production systems are maintained by multiple teams who may not be aware of all the changes made by others. Complex systems also require particular levels of monitoring, and an individual troubleshooter may not have access to all monitoring tools. Even the most complex problems can be resolved by asking simple questions at the right time. Mastering the technology is a matter of building up experience and a repertoire of questions which bring different angles and dimensions to bear on the problems.

When troubleshooting a network device, it is best to first identify possible KPI degradations and initial events or symptoms of the problem from a macro perspective. Once a baseline is established, then focus in on the details. Begin with the data and verifiable facts rather than leaping to a conclusion. Try to identify the preconditions of the issue and clearly define the symptoms. This approach can reduce the time to diagnose and resolve issues.

### Initial questions:

- Which product has the problem?
- What service was running in the product? What other products were affected which were not running the same service?
- What was the symptom noted?

- How many nodes were affected by this problem?
- In which location did the problem occur?
- When was the problem first noticed?
- What was the expected behavior?
- Is there a location with a similar setup which did not experience this problem?
- If the problem is no longer seen, how long did the problem persist? What was done that caused the problem to be resolved?
- What activity was going on in the network during the time the problem was seen? (This activity is not necessarily related to the Ultra M environment but could be related to other activities such as routing changes, switch replacement, etc.)
- Was there a change in network management prior to the incident?
- Has this problem occurred before? If so, what was the history of that occurrence?
- What else was observed?

**More detailed questions specific to Ultra M:**

- How many subscribers are affected? Is there any specific type of subscriber affected?
- How many subscribers are not affected? Is there anything specific shared by those who are not affected?
- What specific KPI degradation is observed? Which values are expected and what are some previous trends? Are there any other KPIs which follow the same trends?
- Confirm if the problem is related to:
  - Single, some or all services
  - Specific VM, a group of VMs or all VMs
  - Specific card/port, group or all cards/ports
  - Specific sessmgr/aaamgr or all sessmgr/aaamgrs
  - Single or multiple services
  - Single or multiple VLANs/interfaces

- Single, multiple or all NPUSIMs. If NPUSIM is not the problem, check the Di Network
  - Type of calls, and if so, what is the duration of those calls or any other factor specific to those calls?
  - A specific interface - how is the call flow affected and in which phase is it affected?
  - A specific peer group or all peers - is there anything specific about the peers affected?
- Is there anything specific to the affected components or objects?
  - Identify any events which occurred close to the same times when a previous example of this issue happened. What is common to these events?
  - When thinking about symptoms or conditions always confirm: Where was the problem experienced? Where was it not experienced? What was not experienced? Where?
  - Where not? When? When not?
  - What percentage of devices or users are affected? What observations can be made about these devices or users?
  - Confirm whether this issue is expected behavior and if not, in what ways it is unexpected. What differences are there between the affected and unaffected? What is the key difference from non-expected behavior?
  - What are the examples of working and non-working case scenarios?
  - What was the duration of the issue? Is this a recurring or one-time event?
  - History of the problem:
    - Issue trigger?
    - Configuration/Recent/Network changes?
    - Change management?
    - Flapping?
    - Timeline of events?
    - Logs?

- Patterns?
- Differences and delta?
- Pre-conditions?
- Micro- and macro-perspectives?
- Confirmed and excluded conditions or triggers?
- Trends?
  
- Relevant configuration details:
  - Working config
  - Non-working config
  - Problematic?
  - Specific details?
  
- Relevant counters, thresholds, limits, return codes, frequency, timeouts, delta max/avg/min/deviation?
- Any specific ranges, oscillations, deviations or differences? Trends/graphs?

**Other involved devices:**

- What is the difference noticed by comparison to other involved devices, both working and nonworking?
- Hardware
- Software version
- Configuration changes
- Are these Cisco devices or third-party devices?
- Any other unknowns: Related info, logs, traces, events or history?

**Topology and design details:**

- Diagrams or graphs
- Design documents
- Previous stability duration
- Previous versions of software/hardware in use?
- Is this a new setup?
- Is this a production, lab, or another type of environment?
- Are other teams involved?

**Reproduction:**

- Issue reproduced? If yes, can it be consistently reproduced or is it an intermittent problem?
- What method was used to reproduce the issue?
- Which tools were used to reproduce the issue?
- For how long have reproduction attempts been made?

**Workarounds:**

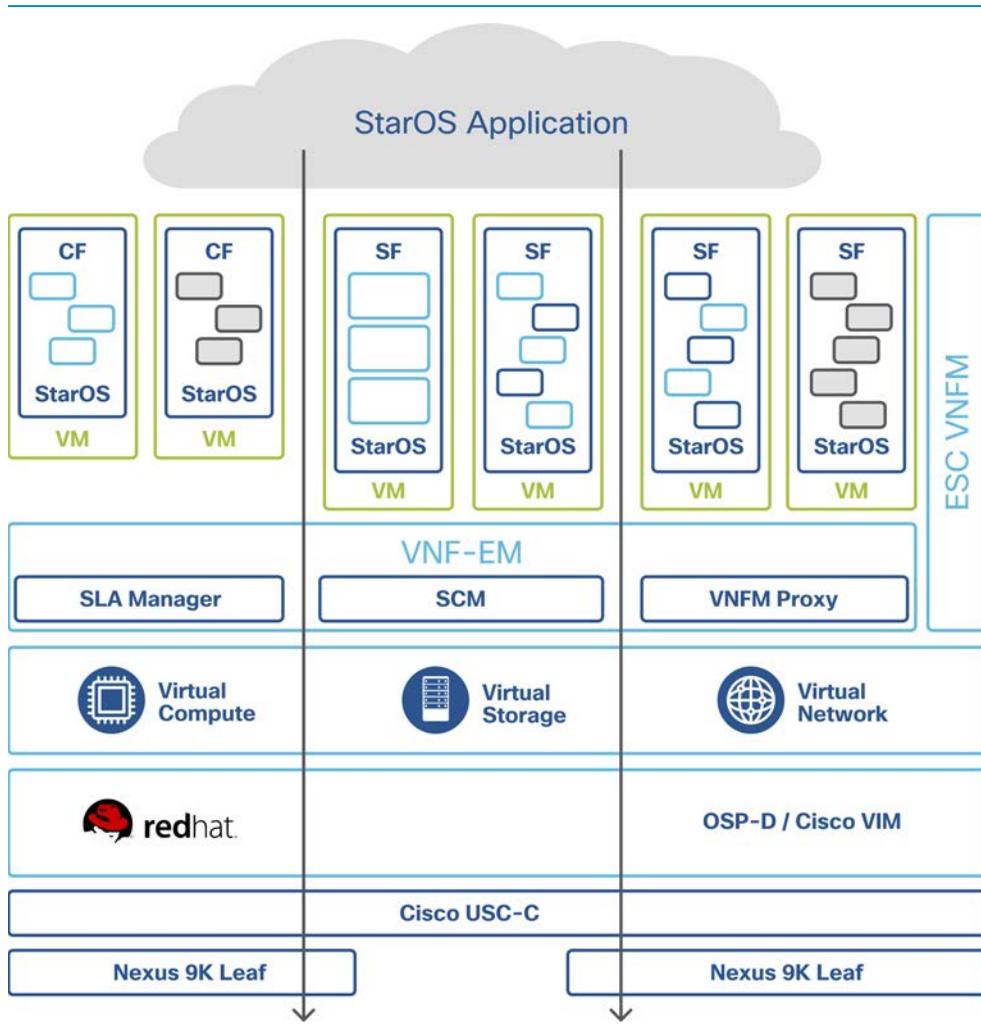
- Is it possible?
- What are the disadvantages of implementing the workaround? Any subscriber impact?
- Is it confirmed, tested and validated in the lab?

**General Methodology - a layered-based approach for troubleshooting Ultra M Components**

In the absence of a specific alarm pointing to the cause of a detected problem, the general troubleshooting approach should be taken.

If the detected problem is not related to the Application(StarOS), check Orchestration Layer.

If the detected problem is not related to the Orchestration Layer, check the Virtual and Physical Infrastructure.



# Health Checks

## Ultra M Manager Health Check Service

The Ultra M Manager Health Check Service is designed to specify which components, domains, services, and severity to monitor in an Ultra M deployment. The service is available for Red Hat OpenStack Based Deployments. It comes in the form of RPM (Red Hat Package Manager) that has to be installed additionally on the Ultra M Openstack Director Node. The service provides a centralized point of monitoring through which different components of the solution may be overseen.

Ultra M Manager Health Check Service provides the following functions when configured:

- Syslog Proxy.
- Aggregation of the events from Cisco UCS, OpenStack, and Ultra Automation Services and sending SNMP traps northbound. Events received from the listed components are mapped against Ultra M SNMP MIB.
- Utilities to ease upgrading UCS firmware.

The detailed list of monitored services is available in the Ultra M Solutions Guide for the respective software version, specifically:

- Ultra M MIB for MIB Information.
- Event and Syslog Management within the Ultra M Solution for install steps, starting/stoppings service, and details about monitored services.
- Using the UCS Utilities within the Ultra M Manager Section provides details on the ways in which the Ultra M Manager Health Check Service can be used to simplify the upgrade of the UCS firmware.

The output of the Ultra M Manager Health Check Service is located in the **`/var/log/cisco/ultram-health`** location. Inside the location, 3 types of files are created:

- report - these are generated on every run and represent the summary of the system status. These files are useful for overall status monitoring.
- log - provide verbose output from debug/log from different components. These files are useful for troubleshooting.
- error - these events cause Ultra M Manager Health Check Service to generate the trap northbound.

## Nexus 9000

Runtime Health Monitoring provides realtime operational status update on the Nexus switch. Please refer to the Nexus 9000 System Management Guide under section Configuring Online Diagnostics.

The Health Monitoring Nondisruptive Diagnostics table gives the frequency of monitoring. The following commands provide useful information:

```
show diagnostic events [error | info]

1) Event:E_DEBUG, length:128, at 363740 usecs after Tue Mar 27 14:21:52 2018
   [543520084] Mar 27 14:21:52 2018(diagclient_handle_bootup_testing):Bootup test
   result handle: no.
   of bootup test results pending 0

2) Event:E_DEBUG, length:103, at 363729 usecs after Tue Mar 27 14:21:52 2018
   [543520084] Mar 27 14:21:52 2018(diagclient_post_fsm_event_pair):setting test id
   in event rid = D
   , slot=0

3) Event:E_DEBUG, length:102, at 363721 usecs after Tue Mar 27 14:21:52 2018
   [543520084] Mar 27 14:21:52 2018(diagclient_test_result_notif):Clearing current
   testing type in t
   est = 4

4) Event:E_DEBUG, length:142, at 363713 usecs after Tue Mar 27 14:21:52 2018
   [543520084] Mar 27 14:21:52 2018(diagclient_test_result_notif):test srv uuid =
   503 test run count
   = 0 test req count = 0 num hm test enabled= 1
```

**show diagnostic result module slot [test [test-name | all]] [detail]**

Example:

```

show diagnostic result module 1

Current bootup diagnostic level: complete
Module 1: 48x1/10G SFP+ 6x40G Ethernet Module (Active)

Test results: (. = Pass, F = Fail, I = Incomplete,
U = Untested, A = Abort, E = Error disabled)

1) USB-----> .
2) NVRAM-----> .
3) RealTimeClock-----> .
4) PrimaryBootROM-----> .
5) SecondaryBootROM-----> .
6) BootFlash-----> .
7) SystemMgmtBus-----> .
8) OBFL-----> .
9) ACT2-----> .
10) Console-----> .
11) FpgaRegTest-----> .
12) Mce-----> .
13) AsicMemory-----> .
14) Pcie-----> .
15) PortLoopback: U

Port  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
-----
      U  U  U  U  U  U  U  U  U  U  U  U  U  U  U  U

Port 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
-----
      U  U  U  U  U  U  U  U  U  U  U  U  U  U  U  U

Port 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
-----
      U  U  U  U  U  U  U  U  U  U  U  U  U  U  U  U

Port 49 50 51 52 53 54
-----
      U  U  U  U  U  U
    
```

```

16) RewriteEngineLoopback: .

Port  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
-----
. . . . .

Port  17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
-----
. . . . .

Port  33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
-----
. . . . .

Port  49 50 51 52 53 54
-----
. . . . .
    
```

**show diagnostic status module slot**

```

show diagnostic status module 1

          <BU>-Bootup Diagnostics, <HM>-Health Monitoring Diagnostics
          <OD>-OnDemand Diagnostics, <SCH>-Scheduled Diagnostics

=====
Card:(1) 48x1/10G SFP+ 6x40G Ethernet Module
=====
Current running test           Run by
-NA-                           -NA-
Currently Enqueued Test       Run by
-NA-                           -NA-
    
```

CLI **show version** provides information on software level and uptime:

```

Leaf-1# show version
Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Copyright (C) 2002-2016, Cisco and/or its affiliates.
All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under their own
licenses, such as open source. This software is provided "as is," and unless
otherwise stated, there is no warranty, express or implied, including but not
    
```

```

limited to warranties of merchantability and fitness for a particular purpose.
Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or
GNU General Public License (GPL) version 3.0 or the GNU
Lesser General Public License (LGPL) Version 2.1 or
Lesser General Public License (LGPL) Version 2.0.
A copy of each such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://opensource.org/licenses/gpl-3.0.html and
http://www.opensource.org/licenses/lgpl-2.1.php and
http://www.gnu.org/licenses/old-licenses/library.txt.

```

Software

```

BIOS: version 07.56
NXOS: version 7.0(3)I4(2)
BIOS compile time: 06/08/2016
NXOS image file is: bootflash:///nxos.7.0.3.I4.2.bin
NXOS compile time: 7/21/2016 8:00:00 [07/21/2016 16:09:32]

```

Hardware

```

cisco Nexus9000 93180YC-EX chassis
Intel(R) Xeon(R) CPU @ 1.80GHz with 24634044 kB of memory.
Processor Board ID FDO00000EM

```

```

Device name: Leaf-1
bootflash: 53298520 kB

```

Kernel uptime is 37 day(s), 18 hour(s), 27 minute(s), 55 second(s)

Last reset

```

Reason: Unknown
System version: 7.0(3)I4(2)
Service:

```

plugin

```

Core Plugin, Ethernet Plugin

```

Active Package(s):

## Catalyst Switch

During every boot cycle, diagnostics tests are run. If there are any problems the booting will stop. The diagnostic test results example is shown:

```

show diagnostic post
Stored system POST messages:

Switch 1
-----

POST: MA BIST : Begin
POST: MA BIST : End, Status Passed

POST: TCAM BIST : Begin
POST: TCAM BIST : End, Status Passed

POST: Thermal Tests : Begin
POST: Thermal Tests : End, Status Passed

POST: PortASIC Stack Port Loopback Tests : Begin
POST: PortASIC Stack Port Loopback Tests : End, Status Passed

POST: PortASIC Port Loopback Tests : Begin
POST: PortASIC Port Loopback Tests : End, Status Passed

POST: EMAC Loopback Tests : Begin
POST: EMAC Loopback Tests : MAC Loopback Passed
POST: EMAC Loopback Tests : PHY Loopback Passed
POST: EMAC Loopback Tests : End, Status Passed

```

**show interfaces** privileged EXEC command to see if the port is in error-disabled, disabled, or shutdown status.

Example:

```

show interfaces GigabitEthernet 1/0/1
GigabitEthernet1/0/1 is up, line protocol is up (connected)
  Hardware is Gigabit Ethernet, address is 2cab.ebac.ab81 (bia 2cab.ebac.ab81)
  MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 1000Mb/s, media type is 10/100/1000BaseTX
  input flow-control is off, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0

```

```

Queueing strategy: fifo
Output queue: 0/40 (size/max)
 5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 2000 bits/sec, 2 packets/sec
   14 packets input, 896 bytes, 0 no buffer
   Received 14 broadcasts (0 multicasts)
   0 runts, 0 giants, 0 throttles
   0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
   0 watchdog, 0 multicast, 0 pause input
   0 input packets with dribble condition detected
 9910277 packets output, 660981565 bytes, 0 underruns
   0 output errors, 0 collisions, 1 interface resets
   0 unknown protocol drops
   0 babbles, 0 late collision, 0 deferred
   0 lost carrier, 0 no carrier, 0 pause output
   0 output buffer failures, 0 output buffers swapped out

```

If a port is in error-disabled state, then **shutdown** of the port is required, followed by a **no shutdown** of the port in configuration mode to come out of error-disabled state.

If the port statistics show excessive FCS, late-collision, or alignment errors, verify that the cable distance from the switch to the connected device meets the recommended guidelines.

CLI **show version** provides information on software level and uptime:

```

C2960X-1> show version
Cisco IOS Software, C2960X Software (C2960X-UNIVERSALK9-M), Version 15.2(2)E5,
RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2016 by Cisco Systems, Inc.
Compiled Thu 02-Jun-16 01:31 by prod_rel_team

ROM: Bootstrap program is C2960X boot loader
BOOTLDR: C2960X Boot Loader (C2960X-HBOOT-M) Version 15.2(3r)E1, RELEASE SOFTWARE
(fc1)

C2960X-1 uptime is 5 weeks, 2 days, 18 hours, 29 minutes
System returned to ROM by power-on
System restarted at 11:19:17 UTC Sun Feb 18 2018
System image file is "flash:/c2960x-universalk9-mz.152
-2.E5/c2960x-universalk9-mz.152-2.E5.bin"
Last reload reason: power-on

```

cisco WS-C2960XR-48TD-I (APM86XXX) processor (revision Q0) with 524288K bytes of memory.

Processor board ID FDO0000B0N0  
 Last reset from power-on  
 2 Virtual Ethernet interfaces  
 1 FastEthernet interface  
 50 Gigabit Ethernet interfaces  
 2 Ten Gigabit Ethernet interfaces  
 The password-recovery mechanism is enabled.

512K bytes of flash-simulated non-volatile configuration memory.

Base ethernet MAC Address : 2C:AB:EB:AC:AB:80  
 Motherboard assembly number : 73-100636-04  
 Power supply part number : 341-0530-03  
 Motherboard serial number : FDO0000KPK  
 Power supply serial number : LIT000000SD  
 Model revision number : Q0  
 Motherboard revision number : A0  
 Model number : WS-C2960XR-48TD-I  
 Daughterboard assembly number : 73-14200-03  
 Daughterboard serial number : FDO0000KBT  
 System serial number : FDO00000N0  
 Top Assembly Part Number : 68-100313-03  
 Top Assembly Revision Number : A0  
 Version ID : V07  
 CLEI Code Number : CMMKL10ARG  
 Daughterboard revision number : A0  
 Hardware Board Revision Number : 0x22<

Switch Ports	Model	SW Version	SW Image
* 1 52	WS-C2960XR-48TD-I	15.2(2)E5	C2960X-UNIVERSALK9-M

Configuration register is 0xF

## UCS

Please refer to the corresponding Server Installation and Service Guide for accessing UCS via CIMC (Cisco Integrated Management Controller) and checking the health status. See below for examples of Cisco UCS C240 M4 outputs.

To display details of UCS use the following:

```
C240-FCH0000V1JJ# show cimc detail
Cisco IMC:
  Firmware Version: 2.0(13i)
  Current Time: Wed Mar 28 01:48:59 2018
  Boot-loader Version: 2.0(13i).36
  Local Time: Wed Mar 28 01:48:59 2018 UTC +0000
  Timezone: UTC
  Reset Reason: ac-cycle

C240-FCH0000V1JJ# show chassis detail
Chassis:
  Power: on
  Serial Number: FCH0000V1JJ
  Product Name: UCS C240 M4SX
  PID : UCSC-C240-M4SX
  UUID: 1BBCB979-E9EF-40D1-83C4-7C751CC06D94
  Software Serial Number:
  Locator LED: off
  Description:
```

To display CPU information:

```
C240-FCH0000V1JJ# scope chassis
C240-FCH0000V1JJ /chassis # show cpu
Name          Cores    Version
-----
CPU1          14      Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz
CPU2          14      Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz
```

To display memory and DIMM information:

```
C240-FCH0000V1JJ /chassis # show dimm
```

Name	Capacity	Channel Speed (MHz)	Channel Type
DIMM_A1	32768 MB	2400	DDR4
DIMM_A2	Not Installed	Unknown	Unknown
DIMM_A3	Not Installed	Unknown	Unknown
DIMM_B1	32768 MB	2400	DDR4
DIMM_B2	Not Installed	Unknown	Unknown
DIMM_B3	Not Installed	Unknown	Unknown
DIMM_C1	Not Installed	Unknown	Unknown
DIMM_C2	Not Installed	Unknown	Unknown
DIMM_C3	Not Installed	Unknown	Unknown
DIMM_D1	Not Installed	Unknown	Unknown
DIMM_D2	Not Installed	Unknown	Unknown
DIMM_D3	Not Installed	Unknown	Unknown
DIMM_E1	32768 MB	2400	DDR4
DIMM_E2	Not Installed	Unknown	Unknown
DIMM_E3	Not Installed	Unknown	Unknown
DIMM_F1	32768 MB	2400	DDR4
DIMM_F2	Not Installed	Unknown	Unknown
DIMM_F3	Not Installed	Unknown	Unknown
DIMM_G1	Not Installed	Unknown	Unknown
DIMM_G2	Not Installed	Unknown	Unknown
DIMM_G3	Not Installed	Unknown	Unknown
DIMM_H1	Not Installed	Unknown	Unknown
DIMM_H2	Not Installed	Unknown	Unknown
DIMM_H3	Not Installed	Unknown	Unknown

```
C240-FCH0000V1JJ /chassis # show dimm-summary
```

```
DIMM Summary:
  Memory Speed: 2400 MHz
  Total Memory: 131072 MB
  Effective Memory: 131072 MB
  Redundant Memory: 0 MB
  Failed Memory: 0 MB
  Ignored Memory: 0 MB
  Number of Ignored Dimms: 0
  Number of Failed Dimms: 0
  Memory RAS possible: Independent Mirroring Lockstep
  Memory Configuration: Independent
```

To display power supply properties:

```
C240-FCH0000V1JJ /chassis # show psu
Name          In. Power (Watts)  Out. Power (Watts)  Firmware  Status  Product ID
-----
-----
PSU1          91                 N/A                 10052030  Present UCSC-PSU2V2-
1...
PSU2          120                N/A                 10052030  Present UCSC-PSU2V2-
1...
```

**show current** command under **scope sensor** will give the current status of all sensors including fans, power supply and temperature:

```
C240-FCH0000V1JJ# scope sensor
C240-FCH0000V1JJ /sensor # show current
Name          Sensor Status  Reading  Units
Min. Warning Max. Warning Min. Failure Max. Failure
-----
-----
PSU1_IOUT     Normal        6.00    AMP
N/A           114.00        N/A     116.00
PSU2_IOUT     Normal        7.00    AMP
N/A           114.00        N/A     116.00
```

## VIM

The status of the cluster in OpenStack is checked with the following command. The output will show current issues with the controllers' cluster and online/offline controllers. Additionally, the status of RabbitMQ is showed when errors are detected. Please refer to the VIM section for further steps. The output below does not display any issues:

```
[heat-admin@controller-0 ~]$ sudo pcs status
Cluster name: tripleo_cluster
Stack: corosync
Current DC: controller-0 (version 1.1.15-11.e17_3.4-e174ec8) - partition with quorum
Last updated: Tue Mar 27 15:05:57 2018      Last change: Tue Mar 27 15:00:31 2018
by root via crm_resource on controller-0
```

```

3 nodes and 19 resources configured

Online: [ controller-0 controller-1 controller-2 ]

Full list of resources:

ip-11.18.0.107    (ocf::heartbeat:IPAddr2):    Started controller-0
ip-11.20.0.106    (ocf::heartbeat:IPAddr2):    Started controller-2
Clone Set: haproxy-clone [haproxy]
  Started: [ controller-0 controller-1 controller-2 ]
Master/Slave Set: galera-master [galera]
  Masters: [ controller-0 controller-1 controller-2 ]
ip-11.19.0.105    (ocf::heartbeat:IPAddr2):    Started controller-0
ip-11.20.0.108    (ocf::heartbeat:IPAddr2):    Started controller-2
Clone Set: rabbitmq-clone [rabbitmq]
  Started: [ controller-0 controller-1 controller-2 ]
Master/Slave Set: redis-master [redis]
  Masters: [ controller-1 ]
  Slaves: [ controller-0 controller-2 ]
ip-192.200.0.111  (ocf::heartbeat:IPAddr2):    Started controller-0
ip-10.201.206.23  (ocf::heartbeat:IPAddr2):    Started controller-2
openstack-cinder-volume (systemd:openstack-cinder-volume):
  Started controller-0

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled

```

### VIM Health status in Health Check Service

When using the Ultra M Health Manager for monitoring the OpenStack services status, the output of the file **ultram\_health\_os.report** should be checked. This gives the status of all essential services for all OpenStack nodes:

```

[stack@bru-ospd-ultram-1 ultram-health]$ more ultram_health_os.report
-----
OpenStack Node           | Service Name                       | Status
| Error Info, if any
-----
-----

```

```
(truncated output)

...
ultram-controller-2 | pcs:status | :-)
|
ultram-controller-2 | pcs:cluster_status | :-)
|
ultram-controller-2 | rabbitmqctl:cluster_status | :-)
|
ultram-controller-2 | ntpdc:loopinfo | :-)
|
ultram-controller-2 | openstack-aodh-evaluator.service | :-)
|
ultram-controller-2 | openstack-aodh-listener.service | :-)
|
....
```

## ESC

In the Ultra M solution, the ESC is deployed in Master-Backup Mode with 2 VMs without volumes.

The following command provides brief information about the role (Master or Backup) and overall health status:

```
[admin@ultram-vnfml-esc-0 ~]$ escadm status
0 ESC status=0 ESC Master Healthy
```

And for the standby node:

```
[admin@ultram-vnfml-esc-1 ~]$ escadm status
0 ESC status=0 ESC Backup Healthy
```

ESC software also provides a health utility that performs a basic health check of the ESC. This script verifies the status of the enabled services and provides final status:

```
[admin@ultram-vnfml-esc-0 ~]$ health.sh
esc ui is disabled -- skipping status check
esc_monitor start/running, process 867
esc_mona is up and running ...
vimmanager start/running, process 2934
vimmanager start/running, process 2934
esc_confd is started
tomcat6 (pid 3159) is running...           [ OK ]

postgresql-9.4 (pid 2853) is running...
ESC service is running...
Active VIM = OPENSTACK
ESC Operation Mode=OPERATION

/opt/cisco/esc/esc_database is a mountpoint
===== ESC HA (MASTER) with DRBD =====
DRBD_ROLE_CHECK=0
MNT_ESC_DATABASE_CHECK=0
VIMMANAGER_RET=0
ESC_CHECK=0
STORAGE_CHECK=0
ESC_SERVICE_RET=0
MONA_RET=0
ESC_MONITOR_RET=0
=====
ESC HEALTH PASSED
```

The backup ESC should also be checked. As the detailed checking can be done while a node is active, the Backup ESC will indicate status as Failed or Passed only (without the detailed breakdown as above for the Master node)

```
[admin@ultram-vnfml-esc-1 ~]$ health.sh

===== ESC HA (BACKUP) =====
=====
ESC HEALTH PASSED
```

## ESC Status Monitored by Health Check Service

Ultra M Manager Health Check Service shows the status of the ESC by displaying the content of the `ultram_health_uas.report` output:

```
root@ospd-ultram-1 ultram-health]# more ultram_health_uas.report
-----
-----
VNF ID          | UAS Node | Status  | Error Info, if any
-----
-----
172.21.201.128 | autovnf  | :-      |
172.21.201.128 | vnf-em   | :-      |
172.21.201.128 | esc      | :-      |
-----
-----
```

## Disk Utilization Proactive Monitoring

It is recommended that the disk is kept in a healthy state by monitoring disk space on both Master and Backup ESC in `/var` and `/opt` partitions. The following commands may be used for proactive monitoring:

```
df -h

[admin@ultram-vnfm1-esc var]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda3       4.7G  1.7G  2.8G  38% /
tmpfs           1.9G  4.0K  1.9G   1% /dev/shm
/dev/vda1       477M   32M  420M   7% /boot
/dev/vda5       4.7G  564M  3.9G  13% /opt
/dev/vda6       11G   351M  11G   4% /var
```

**Note** It is important to perform the periodic backup of the ESC configuration. This is covered in the Elastic Services Controller Install and Upgrade Guide for the appropriate software version.

## EM

Element Manager runs in Active Standby mode and has 3 components:

- VNFM Proxy - creates and monitors nodes in the cluster
- Service Configuration Manager (SCM)
- SLA Manager - polls for critical logs

It is important that all 3 components are up and running, otherwise, the EM is not considered healthy.

### Manual EM Health Check Status

EM is running in Active/Standby pair and requires that all 3 functions are up and running for the health to be in a non-error state.

To check the state of the EM, connect to the EM VIP address and run **show ems** command. Expected output:

```
admin@scm# show ems
EM          VNFM
ID  SLA  SCM  PROXY
-----
5   up  up  up
8   up  up  up
```

From the above, it can be seen that there are two EM nodes, with ID 5 and 8. These IDs are coming from the IP address of the EM VMs. Their status can be verified and is "up" for all services: SLA, SCM, and VNFM Proxy.

To verify the High-Availability state, we can run the **show ncs-state ha** command, which will give us an overview of the High-Availability status. Expected output is:

```
admin@scm# show ncs-state ha
ncs-state ha mode master
ncs-state ha node-id 5-1522100017
ncs-state ha connected-slave [ 8-1522100130 ]
```

Connect to the standby(slave) node to see:

```
admin@scm# show ncs-state ha
ncs-state ha mode slave
ncs-state ha node-id 8-1522100130
ncs-state ha master-node-id 5-1522100017
```

### EM Status Check by Health Check Service

Ultra M Health Manager performs the basic health check of the EM for all 3 components. The status is visible from the **ultram\_health\_uas.report**:

The expected output is :- ) which indicates EM is working fine:

```
[stack@ospd-ultram-1 ultram-health]$ more ultram_health_uas.report
-----
VNF ID          | UAS Node | Status  | Error Info, if any
-----
172.21.201.128 | autovnf  | :- )   |
172.21.201.128 | vnf-em   | :- )   |
172.21.201.128 | esc      | :- )   |
-----
```

In cases where an EM component failure is detected, the health manager will present the status of the individual component:

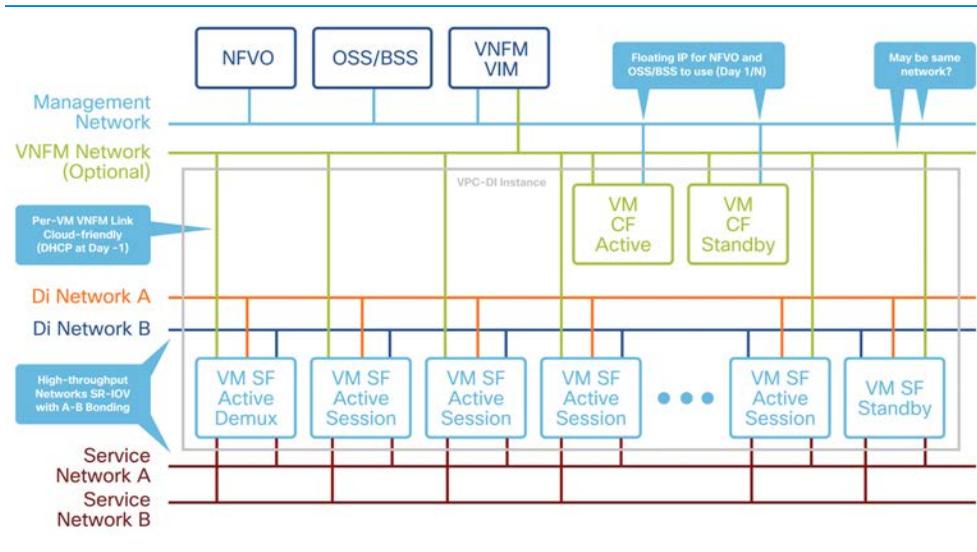
```
[stack@ospd-ultram-1 ultram-health]$ more ultram_health_uas.report
-----
VNF ID          | UAS Node | Status  | Error Info, if any
-----
172.21.201.128 | autovnf  | :)      |
172.21.201.128 | vnf-em   | XXX     | EM : 5 not healthy (SLA: up, SCM: down,
VNFM-PROXY: up)
172.21.201.128 | esc      | :-)     |
-----
```

In the above output, we can note the EM is reported not healthy. Specifically, in the example, we can see the SLA and VNF-PROXY function are running fine (status up), but the SCM status is down. For further troubleshooting of the EM being in any state other than healthy (:)), please refer to the troubleshooting EM section.

## VPC-DI

### Di Network:

This diagram shows a basic Di network layout.



**VPC-DI Component:**

VPC-DI is a distributed VNF consisting of multiple VMs acting as a single system. The Virtual Network Function (VNF) consists of 2 “CF” VMs acting as 1:1 redundant management/supervisor roles and multiple other VMs called “SF” doing session processing.

**Heartbeat CF-CF:**

Standby CF is declared failed after between 2.2 and 3.1 seconds having lost 2 heartbeats with a 1.2 second timeout each and a 3rd heartbeat timeout of 0.2 seconds.

Active CF is declared failed after between 4.2 and 5.1 seconds having lost 4 heartbeats with a 1.2 second timeout each and a 5th heartbeat timeout of 0.2 seconds.

**Heart Beat CF-SF:**

Loss of 2 consecutive heartbeats with a 1.2 second timeout each and a 3rd heartbeat timeout of 0.2 seconds.

CF hasn’t received an IPC advertisement from that SF recently (usually 5-8 seconds).

**SF-CF Reverse Heartbeat:**

If the SF does not receive a heartbeat for 7 seconds it will assume both CFs are failed. The SF will perform a one-time reverse heartbeat SF->CF->SF to both CFs to confirm they are down or unreachable.

**Advanced Fault Detection (AFD):**

The active CF sends ICMP packets at a faster rate (10 per second) using priority 6 to check the health of other VMs. The CF uses a directed broadcast ICMP to the broadcast address of the Di Network. Unicast ICMP responses are received from all VMs back into the CF.

**General Di Network Commands:**

Commands	Explanation
<b>show cloud monitor di-network summary</b>	The Health column shows "Bad" if either 5 minutes or 60 minutes loss is greater than 1 %. It indicates the Di Network is not clean.
<b>show cloud monitor di-network detail</b>	Display more information such as absolute number of drops, latency, and most recent results. It also give an idea of Di Network health and also an indication of issue outside Q-VPC.
<b>show heartbeat stats card x cpu 0</b>	Display the recent heartbeat status. This contains 2 components, the 10 most recent heartbeats and the 16 most recent AFD attempts. This command helps to determine the root cause of communication issues between cards and Di Network.
<b>show card table</b>	Look for any cards in o. ine or booting status.
<b>show heartbeat stats</b>	Display the recent heartbeat status. It contains 2 components, the 10 most recent heartbeats and the 16 most recent AFD attempts. This command helps to determine the root cause of communication issue b/w cards.

Example output:

**show cloud monitor di-network summary**

If the rate is larger than 1%, health status is marked bad:

```
[local]vpc-di# show cloud monitor di-network summary
Card 3 Test Results:
ToCard Health 5MinLoss 60MinLoss
  1      Good    0.00%    0.00%
  2      Good    0.00%    0.00%
  4      Bad     1.92%    2.40%
  5      Bad     3.21%    2.86%
  6      Bad     4.18%    4.88%
  7      Bad     3.53%    6.02%
  8      Bad     5.48%    6.67%
Card 4 Test Results:
ToCard Health 5MinLoss 60MinLoss
  1      Good    0.00%    0.00%
  2      Good    0.00%    0.02%
  3      Bad     4.24%    2.38%
  5      Bad     5.00%    2.13%
  6      Bad     1.96%    3.07%
  7      Bad     2.28%    3.07%
  8      Bad     5.46%    4.55%
```

### show cloud monitor di-network detail

Show cloud command is used to check RTT between SFs and SFs and CFs. It makes sure there are no drops and also measures the RTT. It gives test packet loss rate for the past 5 and 60 minutes. If the rate is larger than 1% health status is marked bad:

```
[local]cf-1# show cloud monitor di-network detail | more
Card 3 Test Results:
ToCard Health 5MinLoss 60MinLoss
  1      Good    0.00%    0.00%
  2      Good    0.00%    0.00%
  4      Good    0.00%    0.00%
  5      Good    0.00%    0.00%
  6      Good    0.00%    0.00%
Dest TotalPkt JumboPkt TotalDrops JumboDrops LongRTT AveragerTT
  1  1559451  779725      1          1      0    0.114ms
  2  1559445  779722      1          1      0    0.131ms
  4  1559451  779725      1          1      0    0.122ms
  5  1559448  779724      1          1      0    0.128ms
  6  1559449  779724      1          1      0    0.150ms
Last 10 RTT in ms (starting from most current):
```

1	0.305	2.262	2.167	2.0	1.39	1.313	1.345	2.313
1.352	1.372							
2	1.313	2.259	2.178	1.8	1.34	1.307	1.350	2.319
1.362	1.378							
4	0.309	3.255	2.174	2.7	1.41	1.299	0.308	2.307
1.358	1.367							
5	1.309	2.254	2.176	1.1	1.30	1.304	1.348	2.316
1.360	1.376							
6	1.315	2.264	2.180	1.10	1.36	1.310	1.352	2.322
1.364	1.381							

### show heartbeat stats hatcpu

Heartbeat stats command displays the heartbeat status. This displays 10 of the most recent heartbeats and 16 of the most recent AFD attempts:

```
[local]cf-1# show heartbeat stats hatcpu
facility hatcpu instance 20 intf CPBOND:
hb 1562146 :51.261, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562147 :52.261, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562148 :53.263, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562149 :54.263, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562150 :55.264, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562151 :56.265, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562152 :57.266, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562153 :58.267, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562154 :59.268, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 1562155 :00.268, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
afd 15544159 :59.583, rtt 0ms
afd 15544160 :59.683, rtt 1ms
afd 15544161 :59.784, rtt 0ms
afd 15544162 :59.885, rtt 1ms
afd 15544163 :59.986, rtt 0ms
afd 15544164 :00.086, rtt 1ms
afd 15544165 :00.186, rtt 0ms
afd 15544166 :00.286, rtt 0ms
afd 15544167 :00.387, rtt 0ms
afd 15544168 :00.488, rtt 1ms
afd 15544169 :00.589, rtt 1ms
afd 15544170 :00.689, rtt 0ms
afd 15544171 :00.790, rtt 0ms
afd 15544172 :00.890, rtt 0ms
afd 15544173 :00.991, rtt 0ms
afd 15544174 :01.091, rtt 0ms
```

# Data Collection

## Catalyst

Collect the output of the following commands for data collection:

```
show log > bootflash://logs-mmddyyyy.txt  
  
show tech-support > bootflash://showtech
```

Sample **show log** , CLI output is shown below:

```
show log  
  
Syslog logging: enabled (0 messages dropped, 1 messages rate-limited, 0 flushes, 0  
overruns, xml disabled, filtering disabled)  
  
No Active Message Discriminator.  
  
No Inactive Message Discriminator.  
  
    Console logging: level debugging, 4589 messages logged, xml disabled,  
                    filtering disabled  
    Monitor logging: level debugging, 0 messages logged, xml disabled,  
                    filtering disabled  
    Buffer logging:  level debugging, 4589 messages logged, xml disabled,  
                    filtering disabled  
    Exception Logging: size (4096 bytes)  
    Count and timestamp logging messages: disabled  
    File logging: disabled  
    Persistent logging: disabled  
  
No active filter modules.  
  
    Trap logging: level informational, 4590 message lines logged  
    Logging Source-Interface:      VRF Name:  
  
Log Buffer (4096 bytes):  
4, changed state to down  
Mar 26 14:31:02.505: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/35, changed state
```

```
to up
Mar 26 14:31:03.509: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/35, changed state to up
Mar 26 14:31:20.946: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/39, changed state
to up
Mar 26 14:31:21.950: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/
0/39, changed state to up
Mar 26 14:31:26.028: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/40, changed state to down
Mar 26 14:31:27.074: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/40, changed state
to down
Mar 26 14:31:28.671: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/36, changed state
to up
Mar 26 14:31:29.433: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/34, changed state
to up
Mar 26 14:31:29.671: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/36, changed state to up
Mar 26 14:31:30.432: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/34, changed state to up
Mar 26 14:31:51.068: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/35, changed state to down
```

## Nexus

On the Nexus Switch, execute the following command:

```
switch# tac-pac bootflash://showtech.switch1
```

## UCS

Collect Tech Support information.

Via CIMC, in admin/utilities, collect show tech (local or remote - depending on the setup).

## Red Hat Linux

Two types of log information can be collected at Red Hat Linux level - Red Hat SOS report and Linux level logs from all components via Cisco script.

- RH SOS report:

```
sosreport --batch --all-logs
```

Copy the log files from the location indicated in the above CLI output. This will be needed by Red Hat for analysis.

## OpenStack

Various components of OpenStack logs and their locations are mentioned below. Depending on the component the appropriate logs will need to be collected - either full sosreport or component level logs:

neutron: /var/log/neutron/\*.log

nova: /var/log/nova/\*.log

heat: /var/log/heat/\*.log

ironic: /var/log/ironic/\*.log

glance: /var/log/glance/\*.log

cinder: /var/log/cinder/\*.log

ovswitch: /var/log/neutron/openvswitch-agent.log

ceph logs: /var/log/ceph/ceph.log

ceph monitoring: /var/log/ceph/ceph-mon.hostname.log

ceph osd: /var/log/ceph/ncc-osd.1701.log

## ESC

ESC provides a built-in script for collecting the logs: **collect\_esc\_logs.sh**. This script will create the tarball file in /tmp with a specific name.

```
[admin@ultram-vnfml-esc-0 ~]$ sudo /usr/bin/collect_esc_log.sh
```

By default, ESCs set a logging level to "true" in the developer log section of /opt/cisco/esc/esc-confd/confd.conf

```
<developerLog>
  <enabled>true</enabled>
  <file>
    <enabled>true</enabled>
    <name>/var/log/esc/confd/devel.log</name>
  </file>
```

This log can grow large and if it does, either enable log rotation or set the flag to false.

ESC logs are stored in the /var/log/esc directory. The most useful logs are:

### Yang ESC Log

This log is stored in the yangesc.log file and contains a brief summary of the ConfD transactions. This log is very useful for getting the basic info on historic ESC events and can be used to easily track them. The sample below shows a ConfD transaction - deployment of the one VM, in this case EM.

```
15:30:07,563 02-Mar-2018 INFO ===== CONF D TRANSACTION STARTED =====
15:30:07,700 02-Mar-2018 INFO ===== DEPLOY SERVICE REQUEST RECEIVED (UNDER TENANT) =====
=====
15:30:07,701 02-Mar-2018 INFO Tenant name: core
```

```

15:30:07,701 02-Mar-2018 INFO Deployment name: vnfd1-deployment-em
15:30:08,282 02-Mar-2018 INFO
15:30:08,282 02-Mar-2018 INFO ===== CONFID TRANSACTION ACCEPTED =====
15:31:37,740 02-Mar-2018 INFO
15:31:37,740 02-Mar-2018 INFO ===== SEND NOTIFICATION STARTS =====
15:31:37,740 02-Mar-2018 INFO Type: VM_DEPLOYED
15:31:37,740 02-Mar-2018 INFO Status: SUCCESS
15:31:37,740 02-Mar-2018 INFO Status Code: 200
15:31:37,740 02-Mar-2018 INFO Status Msg: VM Deployed in a service deployment. VM
name: [vnfd1-deployment_vnfd1-_0_5a5ce949-e17f-4a92-
aeac-4a8c454c14c7]
15:31:37,740 02-Mar-2018 INFO Tenant: core
15:31:37,740 02-Mar-2018 INFO Deployment ID: 9c76873f-6a04-4179-acf6-2fe4af4b0ebf
15:31:37,740 02-Mar-2018 INFO Deployment name: vnfd1-deployment-em
15:31:37,740 02-Mar-2018 INFO VM group name: vnfd1-deployment-em-1
15:31:37,740 02-Mar-2018 INFO User configs: 1
15:31:37,740 02-Mar-2018 INFO VM Source:
15:31:37,740 02-Mar-2018 INFO VM ID: 20461b93-4003-4e4c-8ad2-26500d935da3
15:31:37,740 02-Mar-2018 INFO Host ID:
a4a810031bfcefa0c8621e91dfa2b68edd9be00bfdc910407d596e64
15:31:37,740 02-Mar-2018 INFO Host Name: ultram-compute-2.localdomain
15:31:37,740 02-Mar-2018 INFO ===== SEND NOTIFICATION ENDS =====

```

### ESC Manager Log

This log is stored in the escmanager.log and contains the output of the actions taken by the ESC as well as the ESC state machine changes. Normally, this log is very detailed and after the event has been narrowed down by using yangesc.log first, the specific timeframe can be examined in this log.

### ESC Monitor Log

This log is stored in the esc\_monitor.log and contains the output of the ESC performing checks on the environment.

For troubleshooting purposes, it is often important to get the ESC data model and opdata. This can be achieved by running the following command:

```

/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get esc_datamodel/opdata

```

The output of this command yields the current ESC data model which contains useful information, such as Service Deployment status or individual VM status. For example, the output below shows:

- the EM deployment name "vnfd1-deployment-em"
- one EM deployed, called vnfd1-deployment-em-1
- all the details about the deployed VM, including UUID, host info, ports attached, volumes attached...
- the state of the VM (in the example below it is VM\_ALIVE\_STATE)
- the state of the service

```

<deployment_name>vnfd1-deployment-em</deployment_name>

  <service_name>-</service_name>

    <service_version>-</service_version>

    <deployment_id>SystemAdminTenantIdvnfd1-deployment-
em</deployment_id>

    <vm_group>

      <name>vnfd1-deployment-em-1</name>

      <vm_instance>

        <vm_id>20461b93-4003-4e4c-8ad2-26500d935da3</vm_id>

        <name>vnfd1-deployment_vnfd1-_0_5a5ce949-e17f-4a92-aeac-
4a8c454c14c7</name>

<host_id>a4a810031bfcefa0c8621e91dfa2b68edd9be00bfdc910407d596e64</host_id>

    <hostname>ultram-compute-2.localdomain</hostname>

    <interfaces>

      <interface>

        <nicid>0</nicid>

```

```
<type>virtual</type>

<port_id>7f785715-9ab2-41e4-9b89-317c63af0714</port_id>

<network>98d5f497-788c-4cd6-9c39-a95b345bad61</network>

<subnet>01187ff1-d969-43db-879a-b6b535742b5d</subnet>

<ip_address>172.16.180.5</ip_address>

<mac_address>fa:16:3e:0f:65:0b</mac_address>

<netmask>255.255.255.0</netmask>

<gateway>172.16.180.1</gateway>

</interface>

<interface>

  <nicid>1</nicid>

  <type>virtual</type>

  <port_id>c242ba23-dcb0-4a74-8720-1b3604295733</port_id>

  <network>228691d9-8f14-4396-95ad-c5070a935b6a</network>

  <subnet>ad263275-3564-4e16-84f9-36aca2b2e8ed</subnet>

  <ip_address>172.16.181.9</ip_address>

  <mac_address>fa:16:3e:d6:da:66</mac_address>

  <netmask>255.255.255.0</netmask>

  <gateway>172.16.181.1</gateway>

</interface>

</interfaces>

<volumes>

  <volume>
```

```

                <display_name>vnfd1-deployment_vnfd1-_0_5a5ce949-e17f-4a92-
aeac-4a8c454c14c7</display_name>

                <valid>2</valid>

                <external_id>85c7828d-97c4-4612-b324-
cab94f03a88f</external_id>

                <bus>ide</bus>

                <type>LUKS</type>

                <size>>nullGB</size>

            </volume>

        </volumes>

    </vm_instance>

</vm_group>

<vm_group>

...

    <state_machine>

        <state>SERVICE_ACTIVE_STATE</state>

        <vm_state_machine>

            <vm_name>vnfd1-deployment_vnfd1-_0_5a5ce949-e17f-4a92-aeac-
4a8c454c14c7</vm_name>

            <state>VM_ALIVE_STATE</state>

        </vm_state_machine>

    </vm_state_machines>

</state_machine>

</deployments>

</tenant>

```

## UAS

UAS is often referred to as Auto-VNF.

### UAS Logs

Navigate to `/opt/cisco/usp/uas/scripts` and execute `sudo ./collect-uas-logs.sh`

This script will be collecting log tarball data into a file in the `/tmp` directory

```
ubuntu@autovnf1-uas-1:/opt/cisco/usp/uas/scripts$ sudo ./collect-uas-logs.sh
Collecting Network Data

Collecting zookeeper nodes...

Creating log tarball autovnf1-uas-1-uas-logs-2018-03-26_21.53.19.UTC.tar.bz2 ...

....

===== Tarball available at: /tmp/autovnf1-uas-1-uas-logs-2018-03-
26_21.53.19.UTC.tar.bz2 =====

To extract the tarball, run: "tar jxf /tmp/autovnf1-uas-1-uas-logs-2018-03-
26_21.53.19.UTC.tar.bz2"
```

### Transaction and Event Logs

Transaction and event logs are stored in the following sub-directories under `/var/log/cisco-uas`.

- `/ha` High-availability logs provide information about ZooKeeper and ConfD peers joining the cluster and electing the HA master node. For example:

```
ubuntu@autovnf1-uas-1:/var/log/cisco-uas/ha$ more info.log
2018-03-02 15:23:31,407 - Started ConfD Cluster Manager.
2018-03-02 15:23:31,407 - HA Reboot policy is OFF.
2018-03-02 15:23:31,418 - Trying to acquire election lock.
```

```

2018-03-02 15:23:31,435 - Acquired election lock.
2018-03-02 15:23:31,583 - Detected zookeeper follower on this node.
2018-03-02 15:23:31,583 - Trying to become master.
2018-03-02 15:23:31,600 - Attained master state
2018-03-02 15:23:31,624 - Emitted confd-master event.
2018-03-02 15:23:31,632 - AutoVNF service started successfully
2018-03-02 15:23:31,651 - bind ha vip to ha interface successful
2018-03-02 15:23:31,655 - default route deleted
2018-03-02 15:23:31,657 - Successfully set default gateway to 172.16.181.1
2018-03-02 15:23:31,663 - Setting oper data: ha-active in confd.
2018-03-02 15:23:34,446 - Setting oper data: 172.16.181.101, 1.0.1-1 in confd.
2018-03-02 15:23:54,266 - A slave joined the cluster

```

- **/autovnf** AutoVNF Transaction Logs contain transaction sub-directories, VNFD information and NETCONF traces for the given transaction. Transaction ID in AutoVNF is shown in **show transactions** of ConfD.
- **/uas-manager** UAS-manager logs provide information about UAS Manager itself.
- **/zookeeper** ZooKeeper logs provide information on ZooKeeper-related troubleshooting of the synchronization issues between Active/Standby UAS.

## EM

EM logs are collected by executing the script: **/opt/cisco/em-scripts/collect-em-logs.sh**

EM ZooKeeper log files and snapshots are located at: **/var/lib/zookeeper/data/version-2/**

To collect EM ZooKeeper log files:

```

cd /tmp

tar zcfv zookeeper_data.tgz /var/lib/zookeeper/data/version-2/

ls -las /tmp/zookeeper_data.tgz

```

## CVIM

### Running the Cisco VIM Technical Support Tool

Cisco VIM includes a tech-support tool that can be used to gather Cisco VIM information. The tech-support tool can be extended to execute custom scripts. It can be called after **ciscovim run** is executed at least once. The tech-support tool uses a configuration file that specifies what information to collect. The configuration file is located in the following location:

```
/root/openstack-configs/tech-support/tech_support_cfg.yaml
```

The tool keeps track of the point where the Cisco VIM installer has executed and collects the output of files or commands indicated by the configuration file. For example, if the installer fails at Step 3 (VALIDATION), the tech-support tool will provide the information listed in the configuration file up to Step 3 (included). You can override this default behavior by adding the `--stage` option to the command. The tech-support script is located at the management node `/root/installer-{tag-id}/tech-support` directory.

In order to run it after the runner execution, enter the following command:

```
./tech-support/tech_support.py
```

**Note** The above command will collect tech support data from all controller, compute and storage nodes unless specified in the `tech_support_cfg.yaml` file.

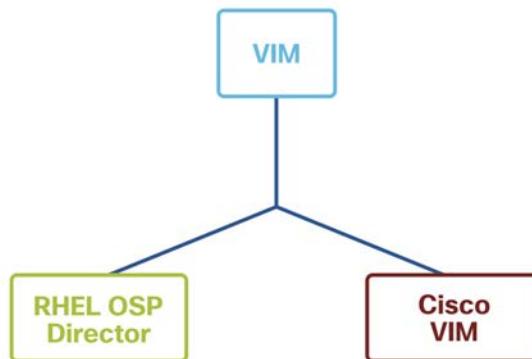
To run the tech support utility for a specific node, run the following command:

```
./tech-support/tech_support.py --host-list=<node ip>
```

It will collect the tech support data from the specific node along with management node.

# Virtual Infrastructure Manager

Ultra M comes with two major different models, based on the Red Hat OSP-D and based on the Cisco VIM. This section will be organized to provide details for each and a third section that will provide common (OpenStack) outputs which can be utilized.



## 1. Red Hat OSP-D Based Deployment

In Red Hat OSP-D Deployment there is a concept of Undercloud and Overcloud.

The Undercloud represents the main director (OSP-D) node that includes components for provisioning and managing the OpenStack nodes. In pre-5.5 Ultra M releases, OSP-D is running on bare metal Red Hat and it is referred to as Undercloud. In post-5.5 Ultra M releases, it is running in the VM and is referred to as VIM-Orch.

Virtualized infrastructure manager is built for full lifecycle management of software and hardware. It allows control and management of NFV infrastructure (NFVI) compute, storage, and network resources – enabling the installation, deployment, and management of a highly secure and high performance cloud.

## 1.1 Undercloud/VIM-ORCH Logs

This section provides the path to collect logs for OpenStack services for Undercloud OpenStack service:

```
neutron: /var/log/neutron/*.log
nova: /var/log/nova/*.log
heat: /var/log/heat/*.log
ironic: /var/log/ironic/*.log
glance: /var/log/glance/*.log
cinder: /var/log/cinder/*.log
ceph: /var/log/ceph/*.log
```

## 2. Cisco VIM Based Deployment

Cisco VIM platform acts as a VIM layer as per ETSI Model. Cisco VIM uses RedHat OpenStack Platform, RedHat Linux Operating System, UCS and Nexus hardware to build the NFV infrastructure and provide virtual resources (compute, network and storage) to the upper layer.

Below are some of the basic Cisco VIM Platform administration commands.

### 2.1 Operating the Cisco VIM - CLI Options

Command line interface to CVIM functions:

run	Perform/terminate an install operation
install-status	Status of installation of the Openstack cloud
list-steps	List steps
add-computes	Add compute-nodes to the Openstack cloud

add-storage	Add a storage-node to the Openstack cloud
list-nodes	List the nodes in the Openstack cloud
power-off	Power Off compute-nodes
power-on	Power On compute-nodes
power-status	Show the power status of nodes
remove-computes	Remove compute-nodes from the Openstack cloud
remove-storage	Remove a storage-node from the Openstack cloud
replace-controller	Replace a controller in the Openstack cloud
list-openstack-configs reconfigure	List of Openstack configs that can be changed using reconfigure
list-password-keys reconfigure	List of password keys that can be changed using reconfigure
reconfigure	Reconfigure the Openstack cloud
cluster-recovery or power outage	Recover the Openstack cluster after a network partition or power outage
mgmtnode-health	Show health of the Management node
commit	Commit an update
rollback	Rollback an update
update	Update the Openstack cloud
update-status	Status of the update operation
upgrade	Upgrade the Openstack cloud
check-fernet-keys	Check whether the fernet keys are successfully synchronized across keystone nodes
nfvbench	Launch NFVBench Flows
nfvimon	NFVI Monitoring / Zenoss management operations

<code>period-rotate-fernet-keys</code>	Set the frequency of fernet keys rotation on keystone
<code>resync-fernet-keys</code>	Resynchronize the fernet keys across all the keystone nodes
<code>rotate-fernet-keys</code>	Trigger rotation of the fernet keys on keystone
<code>client-version</code>	Show Virtualized Infrastructure Manager Version
<code>cloud-sanity</code>	Run cloud-sanity test suite
<code>diskmgmt</code>	HDD maintenance helper
<code>hardware-mgmt</code>	Hardware validation and failure resolution
<code>osdmgmt</code>	OSD maintenance helper
<code>version</code>	Show Virtualized Infrastructure Manager Version
<code>last-run-status</code>	Get Current or Last operation details

When looking for help on the specific command, **ciscovim help <specific command>** option can be used to obtain more information:

```
[root@mgmt ~]# ciscovim help reconfigure

usage: ciscovim reconfigure [--regenerate_secrets]
                               [--setupfile <setupdata_file>]
                               [--setpassword <secretkey>]
                               [--setopenstackconfig <option>] [--cimc_password]
                               [-y]

Reconfigure the Openstack cloud

Optional arguments:

--regenerate_secrets           Regenerate All Secrets

--setupfile <setupdata_file>  User setup_data.yaml

--setpassword <secretkey>     Set of secret keys to be changed.
```

```

--setopenstackconfig <option> Set of Openstack config to be changed.

--cimc_password                Reconfigure CIMC password

-y, --yes                      Yes option to perform the action

```

Examples of some useful Cisco VIM CLIs are provided below:

```

ciscovim install-status
+-----+
| Stages                | Status |
+-----+
| INPUT_VALIDATION      | Success |
| MGMTNODE_ORCHESTRATION | Success |
| VALIDATION            | Success |
| BAREMETAL             | Success |
| COMMON_SETUP          | Success |
| CEPH                  | Success |
| ORCHESTRATION         | Success |
| VMTP                  | Failed  |
+-----+

ciscovim list-nodes
+-----+-----+-----+-----+
| Name                  | Status | Type          | Management IP |
+-----+-----+-----+-----+
| bgl-storage-2-bp     | Active | block_storage | 192.168.11.13 |
| bgl-compute-1-bp     | Active | compute       | 192.168.11.12 |
| bgl-control-1-bp     | Active | control       | 192.168.11.15 |
| bgl-storage-1-bp     | Active | block_storage | 192.168.11.17 |
| bgl-storage-3-bp     | Active | block_storage | 192.168.11.14 |
| bgl-control-3-bp     | Active | control       | 192.168.11.11 |
| bgl-control-2-bp     | Active | control       | 192.168.11.10 |
| bgl-compute-2-bp     | Active | compute       | 192.168.11.16 |
+-----+-----+-----+-----+

ciscovim mgmtnode-health
+-----+-----+-----+
| Rule                  | Status | Error |
+-----+-----+-----+
| Check Docker Pool Settings | PASS  | None  |
| Root Password Check      | PASS  | None  |
| Check Kernel Version     | PASS  | None  |
| Check Root Dir Partition | PASS  | None  |
| Check Docker Version     | PASS  | None  |

```

```

| REST API Server Status          | PASS | None |
| Check Bond Intf. Settings      | PASS | None |
| Check Boot Partition Settings  | PASS | None |
| Check Management Node Tag      | PASS | None |
| Check Ansible Version          | PASS | None |
| Check LVM partition            | PASS | None |
| Check LV Swap Settings         | PASS | None |
| Check Home Dir Partition       | PASS | None |
| Check /var Partition           | PASS | None |
| Check RHEL Pkgs Install State  | PASS | None |
+-----+-----+-----+

```

### 2.2 Operating the Cisco VIM - Platform Checks

Run a sanity check on Cisco VIM platform:

```

[root@mgmt]# ./cloud_sanity.py -c all
Executing All Cloud Sanity in quiet mode. This will take some time.

Cloud Sanity Results
+-----+-----+-----+
+
| Role      | Task                                                                 | Result
|
+-----+-----+-----+
+
| Management | Management - Disk maintenance RAID Health ***** | PASSED
|
|
| Management | Management - Disk maintenance VD Health ***** | PASSED
|
|
| Management | Management - Container version check ***** | 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 *****	SKIPPED
Control	Control - Disk maintenance VD Health *****	SKIPPED
Control	Control - Container version check *****	PASSED
Compute	Compute - Ping All Compute Nodes *****	PASSED
Compute	Compute - Check Nova Hypervisor list *****	PASSED
Compute	Compute - Disk maintenance RAID Health *****	SKIPPED
Compute	Compute - Disk maintenance VD Health *****	SKIPPED

```

| Compute      | Compute - Container version check ***** | PASSED
|
| CephMon      | CephMon - Check Ceph Mon 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
|
| CephOSD      | CephOSD - OSD Overall status ***** | SKIPPED
|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+
[PASSED] Cloud Sanity All Checks Passed

```

Check the REST API server status running inside management node:

```
[root@mgmt tools]# ./restapi.py --action status

Status of the REST API Server: Active: active (running) since Wed 2018-02-28
17:32:32 UTC; 3 weeks 5 days ago
REST API launch directory: /root/installer-12710

----- REST API script execution completed -----
```

Check the syslog files (/var/log/messages) for any errors.

Check the container status of the management node. Containers should be up or created depending on type:

```
[root@mgmt ~]# dp

NAMES                                STATUS

nfvbench_12614                       Up 12 hours
vmtp_12614                           Up 13 hours
vimconfig_12614                      Up 13 hours
fluentd_aggr_12614                   Up 13 hours
curator_12614                        Up 13 hours
kibana_12614                         Up 13 hours
elasticsearch_12614                  Up 13 hours
tftp_server_12614                    Up 13 hours
my_cobbler_12614                     Up 13 hours
repo_mirror_12614                    Up 13 hours
rhel-server-rhceph-osd-rpms_12614    Created
cisco-rhel-server-openstack-hotfix-rpms_12614    Created
```

rhel-server-openstack-devtools-rpms_12614	Created
rhel-server-openstack-optools-rpms_12614	Created
rhel-server-openstack-tools-rpms_12614	Created
rhel-server-openstack-rpms_12614	Created
rhel-ha-for-rhel-server-rpms_12614	Created
rhel-server-rh-common-rpms_12614	Created
rhel-server-extras-rpms_12614	Created
rhel-server-optional-rpms_12614	Created
rhel-server-rpms_12614	Created
mercury-thirdparty-hw-binary-utilities-rpms_12614	Created
mercury-cloudpulse-rpms_12614	Created
mercury-buildnode-rpms_12614	Created
mercury-common-rpms_12614	Created
mercury-repofiles_12614	Created
rhel72_12614	Created
authorized_keys	Created
container_registry	Up 13 hours

To retrieve the supplied alias commands, run **alias** on all Cisco VIM nodes:

```
[root@mgmt]# alias

alias cobbler='in_container my_cobbler_12710'
alias cp='cp -i'
alias curator='in_container curator_12710'
alias dp='docker ps -a --format "table {{.Names}}          {{.Status}}"'
alias egrep='egrep --color=auto'
alias elasticsearch='in_container elasticsearch_12710'
alias fgrep='fgrep --color=auto'
```

```
alias fluentdaggr='in_container fluentd_aggr_12710'
alias grep='grep --color=auto'
alias kibana='in_container kibana_12710'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias nfvbench='in_container nfvbench_12710 nfvbench'
alias registry='in_container container_registry'
alias repomirror='in_container repo_mirror_12710'
alias rm='rm -i'
alias tftpsrv='in_container tftp_server_12710'
alias vimconfig='in_container vimconfig_12710'
alias vmtpl='in_container vmtpl_12710'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
```

Check the container status in Controller/Compute/Storage Nodes:

```
[root@controller1 ~]# dp

NAMES                                STATUS
heatapicfn_12614                     Up 12 hours
heatapi_12614                        Up 12 hours
heatengine_12614                    Up 12 hours
horizon_12614                        Up 12 hours
cloudpulse_server_12614              Up 12 hours
novanovncproxy_12614                Up 12 hours
novaconsoleauth_12614               Up 12 hours
novassh_12614                       Up 12 hours
novacompute_12614                   Up 12 hours
novaapi_12614                       Up 12 hours
novascheduler_12614                 Up 12 hours
```

novaconduct_12614	Up 12 hours
novalibvirt_12614	Up 12 hours
novacommon_12614	Up 12 hours
cindervolume_12614	Up 12 hours
cinderscheduler_12614	Up 12 hours
cinderapi_12614	Up 12 hours
neutron_vpp_12614	Up 12 hours
neutron_metadata_agent_12614	Up 12 hours
neutron_l3_agent_12614	Up 12 hours
neutron_dhcp_agent_12614	Up 12 hours
neutron_server_12614	Up 12 hours
etcd_12614	Up 12 hours
etcddata	Created
neutron_common_12614	Up 12 hours
glanceapi_12614	Up 12 hours
glancer_12614	Up 12 hours
keystone_12614	Up 12 hours
rabbitmq_12614	Up 12 hours
mariadb_12614	Up 12 hours
mariadbdata	Created
haproxy_12614	Up 12 hours
memcached_12614	Up 12 hours
cephmon_12614	Up 12 hours

```

fluentd_12614                Up 12 hours
<...>
    
```

Run the following from the management node to determine the active HAProxy container.

In the example below, the hostnames of the controllers are c1b, c2b and c3b. The hostnames of the controllers in a particular deployment will determine what the syntax for loop would need to be:

```

for f in `seq 1 3`; do ssh 'root@c'$f'b' haproxy 'ip a' ; echo 'c'$f'b' complete ; done
    
```

Look for the output of the HAProxy container that has the API and MGMT HAProxy VIP addresses assigned:

```

[root@mgmt ~]# for f in `seq 1 3`; do ssh 'root@c'$f'b' haproxy 'ip a' ; echo 'c'$f'b' complete ; done
<...>
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWNqlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
11: mgmt@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP qlen 1000
    link/ether 32:43:5d:b9:ae:58 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 169.254.98.118/24 scope global mgmt
        valid_lft forever preferred_lft forever
    inet 192.168.65.41/24 scope global mgmt
        valid_lft forever preferred_lft forever
13: api@if14: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP qlen 1000
    link/ether c6:c7:50:7d:ee:1a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.201.242.203/29 scope global api
        valid_lft forever preferred_lft forever
c3b complete
    
```

Internal and External VIP should be attached to the master controller.

In the above output, 192.168.11.5 is the internal VIP and 10.127.207.31 is the External VIP where OpenStack endpoint APIs are listening.

Please note that the interface number 11 and 13 in HAProxy container is mapped with interface number 12 and 14 in the controller host respectively. They are configured as VETH pair:

```
[root@controller-1]# ip addr show type veth
12: mgmt-out@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
br_mgmt state UP qlen 1000
    link/ether b2:f6:9b:2d:eb:34 brd ff:ff:ff:ff:ff:ff link-netnsid 0
14: api-out@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master
br_api state UP qlen 1000
    link/ether f2:68:3a:ef:8b:bb brd ff:ff:ff:ff:ff:ff link-netnsid 0
```

Check the Ceph Status from ceph-mon container inside controller node:

```
[root@controller ~]# cephmon
cephmon_12710 [ceph@controller /]$ ceph -s
  cluster ecc74b6a-7261-4594-9346-b28c18d5e92b
  health HEALTH_OK
  monmap e1: 3 mons at {controller-1=192.168.12.15:6789/0,controller=
192.168.12.10:6789/0,controller=192.168.12.11:6789/0}
    election epoch 4, quorum 0,1,2 controller-2,controller-3,controller-1
  osdmap e56: 12 osds: 12 up, 12 in
    flags sortbitwise,require_jewel_osds,recovery_deletes
  pgmap v57782: 544 pgs, 5 pools, 530 MB data, 72 objects
    2058 MB used, 13405 GB / 13407 GB avail
    544 active+clean
```

To check **rabbitmq status** in Cisco VIM platform, login to RabbitMQ container (alias name is rabbit):

```
[root@controller]# rabbit
rabbitmq_12710 [rabbitmq@controller/]$
```

Check the commands listed below:

```
rabbitmqctl status
rabbitmqctl node_health_check
rabbitmqctl list_exchanges -p <node_name>
rabbitmqctl list_queues -p <node_name>
```

Check the RabbitMQ logs under **/var/log/rabbitmq** directory.

## 2.2 Cisco VIM Logs

With the OpenStack services running containerized in a Cisco VIM deployment, special care has been taken to log MOST of the container logs externally on the host that the container is running on.

All the OpenStack services (such as Nova, Neutron, Cinder etc.) have related logs available inside **/var/log** directory on the host machine.

```
[root@controller]# ls -ld */
drwxrwx---. 2 root      500    6 Feb 28 19:18 agent-ovs/
drwxrwx---. 2 root      500    6 Feb 28 19:18 aim/
drwxr-xr-x. 2 root    root    4096 Feb 28 19:09 anaconda/
drwx----- 2 root    root      99 Mar 26 21:55 audit/
drwxrwx---. 2 root      500    6 Feb 28 19:18 ceilometer/
drwxrwx---. 2 root      500   86 Feb 28 19:22 ceph/
<..>
```

If any specific log is not available inside **/var/log** directory on the host, then login to the respective service container and fetch the logs.

## 3. Operating OpenStack

This section covers useful troubleshooting commands, followed by some sample outputs.

OpenStack service list will list OpenStack services used in the deployment:

```
openstack service list
```

OpenStack hypervisor list will list the compute resources ID:

```
openstack hypervisor list
```

OpenStack compute service list will list the Nova services status:

```
openstack compute service list
```

OpenStack server list shows the list of host UUID and IP address:

```
openstack server list
```

Neutron network list provides network ID and subnet used in the deployment:

```
neutron net-list  
neutron subnet-list
```

Neutron agent list will list the Neutron plugins used in deployment and the status:

```
openstack network agent list
```

Neutron port list will provide the MAC address of the ports and corresponding network:

```
openstack port list
```

Ironic services will be used to manage/update and build the bare metal nodes:

```
openstack baremetal node list
```

Heat services will be used to deploy the stack and status of resources completed and failed in the stack:

```
openstack stack list  
openstack stack resource list <stack name>
```

Nova list will show virtual machine id, name, host, and instance name:

```
nova list --fields name,host,instance_name
```

OpenStack network-list provides network ID and subnet used in the deployment:

```
openstack network list  
openstack subnet list
```

OpenStack network agent list will list the Neutron plugins used in deployment and the status:

```
openstack network agent list
```

OpenStack router list will list the Neutron routers used in deployment and the status:

```
openstack router list
```

OpenStack l3-agent list will list the Neutron routers placed in the controller and show which active controllers are handling routers in Overcloud deployment and the status:

```
neutron l3-agent-list-hosting-router main
```

OpenStack image list provides the list of images in Overcloud openstack:

```
openstack image list
```

OpenStack service list provides the list of volumes in OpenStack:

```
openstack volume list
```

Ceph status will be checked via login into any controller or OSD compute node:

```
ssh heat-admin@ipadress sudo ceph -s
```

A snapshot of OpenStack CLIs is provided below:

```
openstack service list
+-----+-----+-----+
| ID | Name | Type |
+-----+-----+-----+
| 06c53b63b9254290bcacf2248896c81c4 | glance | image |
| 070d670786314387a3f0ba5ee34e49af | keystone | identity |
| 0a2515eaf04b434499d8d2768f0eef78 | gnocchi | metric |
| 461d78154d6743b0aefa5d0b14a7f299 | heat | orchestration |
| 5932c6f40a95481eaea7cf86d55ad61a | swift | object-store |
| 70c8101728f14fdf93bf8b578c181e3b | aodh | alarming |
| 92acf2db374444d5a473be2317271716 | neutron | network |
| a911d7047e434c32b5056d988db0abfb | heat-cfn | cloudformation |
| b3dbe48b0956471daa282540d125bee5 | cinderv3 | volumev3 |
| daa7747ff08b43c9a48b3406db1f5780 | cinder | volume |
| e3a9deefa73643829d4290ce68a8f5b8 | ceilometer | metering |
| e990cbf3d90f414fa9faffa88a860cf3 | cinderv2 | volumev2 |
| f06c6dce361043e59e267f1ebcca60e5 | nova | compute |
+-----+-----+-----+

openstack hypervisor list
+---+-----+
| ID | Hypervisor Hostname |
+---+-----+
| 2 | osd-compute-1 |
| 5 | compute-0 |
| 14 | osd-compute-0 |
+---+-----+

openstack compute service list
+-----+-----+-----+-----+-----+-----+
| ID | Binary | Host | Zone | Status | State | Updated At |
|
```

```

+-----+-----+-----+-----+-----+-----+-----+
-----+
| 65 | nova-consoleauth | controller-1 | internal | enabled | up | 2018-03-
26T20:16:49.000000 |
| 215 | nova-conductor | controller-2 | internal | enabled | up | 2018-03-
26T20:16:57.000000 |
| 245 | nova-conductor | controller-1 | internal | enabled | up | 2018-03-
26T20:16:58.000000 |
| 275 | nova-scheduler | controller-0 | internal | enabled | up | 2018-03-
26T20:16:48.000000 |
| 278 | nova-compute | osd-compute-1 | nova | enabled | up | 2018-03-
26T20:16:53.000000 |
| 287 | nova-conductor | controller-0 | internal | enabled | up | 2018-03-
26T20:16:58.000000 |
| 317 | nova-compute | osd-compute-0 | nova | enabled | up | 2018-03-
26T20:16:56.000000 |
+-----+-----+-----+-----+-----+-----+-----+
-----+

openstack server list
+-----+-----+-----+-----+-----+-----+-----+
-----+
| ID | Name | Image Name |
Status | Networks |
|
+-----+-----+-----+-----+-----+-----+-----+
-----+
| caf9e049-8739-483f-9c18-9840ee23ea8e | vnf1-deployment_c1_0_9ece3e13-0ab7-45 |
ACTIVE | orchestr=172.16.180.12; ultram-vnfm1 |
|
| | 43-80ed-ce834e1d255e |
| -di-internall1=192.168.1.16; |
| | |
| mgmt=172.16.181.21 |
| 085bbf4e-d87d-491d-afa6-0398da6e7231 | vnf1-deployment_s6_0_eee43d0b-
ACTIVE | ultram-vnfm1-service- | ultram-vnfm1-session-function
|
| | 0ec3-4624-9dfe-d512a381281f |
| network1=10.10.10.2; |
| | |
| orchestr=172.16.180.10; ultram-vnfm1 |
| | |
| -di-internall1=192.168.1.8 |

```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
openstack network list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID | Name |
+-----+-----+-----+-----+
| Subnets |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 102051a7-448e-467e-b04a-feddf830f71 | public |
| 5370f0ae-8952-42f6-8caf-50a10f8544d5 | |
| 2b6d40ab-be89-4f13-8c2d-4177d8420b08 | HA network tenant |
c7ba964cf10049cfb3d34ebde5103a06 | cba7e3c3-3ff6-4e6a-8fa5-fbf53ef818c0 |
| 9357e52d-650c-4380-8938-8a888880b097 | orchestr |
| 4352390f-2d3b-4064-9eab-8a5a976e7d46 | |
| a5431de6-f7ce-4fb7-bc73-d4a0c038d1e3 | mgmt |
| 730533d5-f4f4-4aa9-ae59-647c07097fa1 | |
| ddecae0c-a638-47fa-9923-e2a0030c0ce0 | ultram-vnfm1-di-internall |
| 6b216199-7814-43f2-9084-f84ac5ad3f32 | |
| ff2alcc3-9daa-41f6-9d73-abd8e2954f34 | ultram-vnfm1-service-network1 |
| dadf10bf-ced1-4f2f-a791-ca101345a78e | |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
openstack subnet list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID | Name |
+-----+-----+-----+-----+
| Network | Subnet |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 4352390f-2d3b-4064-9eab-8a5a976e7d46 | orchestr |
| 9357e52d-650c-4380-8938-8a888880b097 | 172.16.180.0/24 |
| 5370f0ae-8952-42f6-8caf-50a10f8544d5 | public-subnet |
| 102051a7-448e-467e-b04a-feddf830f71 | 10.201.206.0/24 |
| 6b216199-7814-43f2-9084-f84ac5ad3f32 | ultram-vnfm1-di-internall-subnet |
| ddecae0c-a638-47fa-9923-e2a0030c0ce0 | 192.168.1.0/24 |
| 730533d5-f4f4-4aa9-ae59-647c07097fa1 | mgmt |
| a5431de6-f7ce-4fb7-bc73-d4a0c038d1e3 | 172.16.181.0/24 |
| cba7e3c3-3ff6-4e6a-8fa5-fbf53ef818c0 | HA subnet tenant |
c7ba964cf10049cfb3d34ebde5103a06 | 2b6d40ab-be89-4f13-8c2d-4177d8420b08 |
169.254.192.0/18 |

```

```

| dadf10bf-ced1-4f2f-a791-ca101345a78e | ultram-vnfm1-service-network1-subnet
| ff2a1cc3-9daa-41f6-9d73-abd8e2954f34 | 10.10.10.0/24 |
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
openstack network agent list
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
| ID | Availability Zone | Alive | State | Binary | Agent Type | Host |
+-----+-----+-----+-----+-----+-----+-----+
| 050c2983-a83e-411b-8d6d-42409e400fea | Open vSwitch agent | osd-compute-1 | None
| True | UP | neutron-openvswitch-agent |
| 3264fa28-087c-48e5-81ea-cdbc26ba7696 | Open vSwitch agent | controller-1 | None
| True | UP | neutron-openvswitch-agent |
| 33b6cec2-3647-41a2-a003-ec663777b21d | Open vSwitch agent | controller-2 | None
| True | UP | neutron-openvswitch-agent |
| fa4b76bd-d54f-43ac-a843-58445c18a7bc | NIC Switch agent | compute-0 | None
| True | UP | neutron-sriov-nic-agent |
| fd8a364d-fbc2-464e-984a-c9caa35531a0 | Open vSwitch agent | controller-0 | None
| True | UP | neutron-openvswitch-agent |
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
openstack router list
+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
| ID | Name | Status | State | Distributed | HA |
Project |
+-----+-----+-----+-----+-----+-----+-----+
| 4ceee33e-55c3-4d1f-b834-c2 | router-1 | ACTIVE | UP | False | True |
7fd84fde844741deb1d62fc5464 |
| 4ead7dfa18abf |
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
openstack port list
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
-----+

```

```

| ID | Name
| MAC Address | Fixed IP Addresses |
+-----+-----+
-----+
| 0136bc69-efe4-4cb6-a8d1-2e8c687951a1 |
| fa:16:3e:81:d7:bc | ip_address='10.201.206.49',
|
| subnet_id='5370f0ae-8952-42f6-8caf-50a10f8544d5'
| 0348005d-ddea-4477-af6c-d4cd131066f7 | autovnf1-uas-0-eth0-port
| fa:16:3e:d1:e6:23 | ip_address='172.16.181.13',
|
| subnet_id='730533d5-f4f4-4aa9-ae59-647c07097fal'
| 0a248633-ce0f-446a-b5af-19bf56a6513a | HA port tenant
c7ba964cf10049cfb3d34ebde5103a06 | fa:16:3e:83:4e:44 |
ip_address='169.254.192.3', subnet_id='cba7e3c3-3ff6 |
|
| -4e6a-8fa5-fbf53ef818c0'
| 0fc3da65-7d47-49a4-9577-2a8c89f3cffe | vnf1-deployment_s6_0_eee43d0b-0ec3-4624-
9dfe- | fa:16:3e:ac:df:12 | ip_address='172.16.180.10', subnet_id='4352390f-
|
| d512a381281f
| 2d3b-4064-9eab-8a5a976e7d46'
| 14da9f21-4f5a-49af-9eaf-e8dd823d62d8 |
| fa:16:3e:2a:2c:52 | ip_address='10.201.206.43',
|
| subnet_id='5370f0ae-8952-42f6-8caf-50a10f8544d5'
+-----+-----+
-----+

openstack baremetal node list
+-----+-----+-----+
--+-----+-----+-----+
| UUID | Name | Instance UUID
| Power State | Provisioning State | Maintenance |
+-----+-----+-----+
--+-----+-----+-----+
| 8ce1e775-1310-4f78-9458-192ca6b3948e | None | 83728dd2-1b2a-4204-94a6-
b81f4d944eba | power on | active | False |
| 832de615-d4c1-4835-866c-0aa3eb4d36b9 | None | 53512dc5-307f-44a3-9fac-
03103c3791ad | power on | active | False |
| 58a979f6-3f29-4c83-9594-6d8a0337870a | None | 702fa698-450f-4acb-8aa6-
d87f3d3b070c | power on | active | False |
| 2266fab0-b871-493d-99ac-06014a338bf7 | None | 4044ab1c-5a96-4402-8fec-

```

```

971a3bfa5feb | power on | active | False |
| 8c4b96b1-05be-4578-8fd2-1392f51978b8 | None | d202b223-7e76-4dba-971f-
8d24c158f980 | power on | active | False |
| 85f58c3c-bf83-4aed-9b42-178a7c36799e | None | 5552fad1-7056-4285-b0f7-
2c7cfb2a0dd9 | power on | active | False |
| d9e561de-6604-4097-952d-a6b0b8c0968b | None | a3a48584-7eee-4455-aala-
39e52ba55e23 | power on | active | False |
| 576ad281-d31e-44b6-9bb6-3b3c55821770 | None | 35b6f193-07e6-44a9-82c1-
116dded3095c | power on | active | False |
+-----+-----+-----+-----+
--+-----+-----+-----+-----+

```

```

nova list --fields host,name
+-----+-----+-----+-----+
-----+
| ID | Host | Name |
|
+-----+-----+-----+
-----+
| 2b9898be-0c5b-4eda-bc75-3b42eafb0cba | compute-0 | autovnf1-uas-0 |
|
| 308b0303-f665-4368-9640-679d7fe13189 | compute-0 | vnfd1-
deployment_s3_0_e7b44362-606a-42fe-a66b-010d8152582e |
| 65db9fe5-2d2b-4892-b292-9b1e15f834fb | compute-0 | auto-it-vnf-ISO-590-uas-0 |
|
+-----+-----+-----+-----+
-----+

```

```

neutron l3-agent-list-hosting-router router-1
+-----+-----+-----+-----+-----+-----+
| id | host | admin_state_up | alive | ha_state |
+-----+-----+-----+-----+-----+-----+
| 41004da8-0580-401c-ab2b-5d3f1785387f | c2b | True | | :- ) | standby |
| d0216e1f-6dcc-4cf8-b07e-61a61486a4e1 | c3b | True | | :- ) | standby |
| c3d6b200-8583-4428-91e5-451167fa2284 | c1b | True | | :- ) | active |
+-----+-----+-----+-----+-----+-----+

```

```

openstack image list
+-----+-----+-----+-----+
--+-----+
| ID | Name |
| Status |
+-----+-----+-----+-----+

```

```
--+-----+
| 6a69ee5c-c7a5-460c-91c3-f2a4f3a4ab32 | ultram-vnfm1-session-function
| active |
| 66ccc9ce-f822-45f6-915a-753001f08c64 | ultram-vnfm1-control-function
| active |
| 1d896280-307c-43ef-8d18-a4471f64ccaf | auto-it-vnf-ISO-590-usp-uas-1.0.0-
632.qcow2 | active |
+-----+
--+-----+

openstack volume list
+-----+
+-----+-----+
| ID | Display Name |
+-----+-----+
| Status | Size | Attached to |
+-----+-----+
+-----+-----+
| ae3b52f7-0d56-407a-bad2-888b480268d5 | autovnf1-uas-vol-2
| in-use | 1 | Attached to autovnf1-uas-2 on /dev/vdb |
| ad7fec33-b42b-47fa-9ff9-a9899fa0d5a3 | autovnf1-uas-vol-1
| in-use | 1 | Attached to autovnf1-uas-1 on /dev/vdb |
| daaf5064-36d7-42b5-b7fc-28eca6c8a2fd | autovnf1-uas-vol-0
| in-use | 1 | Attached to autovnf1-uas-0 on /dev/vdb |
+-----+-----+
+-----+-----+

openstack stack list
+-----+-----+
+-----+-----+
| ID | Stack Name | Stack Status |
Creation Time | Updated Time |
+-----+-----+
+-----+-----+
| b1df4881-7d87-459f-a169-0d97d966b0cc | UltraM-RCDNlab | CREATE_COMPLETE | 2018-
01-06T15:25:08Z | None |
+-----+-----+
+-----+-----+

openstack stack resource list UltraM-RCDNlab
+-----+-----+
+-----+-----+
+-----+
| resource_name | physical_resource_id
```

resource_type	resource_status	updated_time
UpdateWorkflow	fb5fbbbc-58b1-4e92-86dd-d36642350725	
OS::TripleO::Tasks::UpdateWorkflow	CREATE_COMPLETE	2018-01-06T15:25:08Z
CephStorageHostsDeployment	788a2931-67c3-4394-b21d-1dba807c031f	
OS::Heat::StructuredDeployments	CREATE_COMPLETE	2018-01-06T15:25:08Z
OsdComputeAllNodesDeployment	2a9e88cc-777e-4611-8853-7e9fd33aa426	
OS::Heat::StructuredDeployments	CREATE_COMPLETE	2018-01-06T15:25:08Z
BlockStorageHostsDeployment	1be75b9f-bd14-498b-972c-4886757a5c48	
OS::Heat::StructuredDeployments	CREATE_COMPLETE	2018-01-06T15:25:08Z
CephStorage	129cc24f-993c-4d05-8248-e91d9e6db098	
OS::Heat::ResourceGroup	CREATE_COMPLETE	2018-01-06T15:25:08Z
ObjectStorageAllNodesDeployment	ea21ed90-223e-4bd8-8513-b3778812149d	
OS::Heat::StructuredDeployments	CREATE_COMPLETE	2018-01-06T15:25:08Z

openstack baremetal node list

UUID	Name	Instance UUID
Power State	Provisioning State	Maintenance
8ce1e775-1310-4f78-9458-192ca6b3948e	None	83728dd2-1b2a-4204-94a6-b81f4d944eba
power on	active	False
832de615-d4c1-4835-866c-0aa3eb4d36b9	None	53512dc5-307f-44a3-9fac-03103c3791ad
power on	active	False

## CEPH Storage

If there are any problems in Ceph cluster, typically the alarm *Ceph Cluster down* is raised in the OSP-D based deployment.

In case of Cisco VIM-based deployment, the alarm will not be raised.

To verify the Ceph status use the **ceph -w** or **ceph -s** commands to identify the overall status.

```
[heat-admin@osd-compute-0 ~]$ sudo ceph -w
cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_OK
```

Expected output is Health OK.

However, in certain scenarios, we may need to troubleshoot why the CEPH is not healthy and this section will provide some guidelines around CEPH troubleshooting for Ultra M.

- Restarting the CEPH-OSD. The first recommended step is to try to restart the OSD. However, it is advised to disable the scrub before the restart.

```
ceph osd set noscrub
ceph osd set nodeep-scrub
systemctl restart <ceph-osd@x.service>
```

- Ceph Usage Check. If just restarting the ceph does not help, look further into the usage of CEPH. The **ceph df** command will obtain more information

```
[heat-admin@osd-compute-0 ~]$ sudo ceph df
GLOBAL:
  SIZE          AVAIL          RAW USED        %RAW USED
  11161G        10808G         353G            3.16
POOLS:
  NAME          ID      USED          %USED          MAX AVAIL      OBJECTS
  rbd           0         0              0              4213G          0
  metrics      1       1636M          0.04           4213G          96483
  images       2       4766M          0.11           4213G          646
  backups      3         0              0              4213G          0
  volumes      4      32952M         0.76           4213G          8288
  vms          5      81842M         1.86           4213G          20784
```

The output above represents a healthy system. However, from the above, any abnormalities present will be identified.

## Summary of Useful Ceph Troubleshooting Commands:

- Ceph status:
  - `ceph -s`
  - `ceph -w`
  - `ceph df`
  - `ceph osd tree`
  
- Ceph status : Set and unset noscrub and nodeep-scrub:
  - `ceph osd set noscrub`
  - `ceph osd set nodeep-scrub`
  - `ceph osd unset noscrub`
  - `ceph osd unset nodeep-scrub`

- Deep scrub and Repair of a Placement Group (PG):
  - Deep scrub and Repair of a PG:
    - `ceph pg deep-scrub <pg-id>`
    - `ceph pg repair <pg-id>`
    - `ceph -w` to watch the repair status
  - Finding PG details:
    - `ceph health detail`
    - `rados list-inconsistent-obj <> --format=json-pretty`

# VNF Recovery

## SF/CF VM Recovery Prechecks

This section covers prechecks to be performed before starting the VM recovery process through the Element Manager (EM) in VPC.

If StarOS VMs are not in Active or Standby state in the command output resulting from **show card table**, then the VMs can be recovered using the methodology listed in this section.

For the recovery procedure, obtain the OpenStack name of the card (VM). Refer to the EM section on how to obtain and correlate the VM name.

### Step 1:

Log into the EM and check that EM cluster is healthy:

```
admin@scm# show ems

EM VNFM
ID SLA SCM PROXY
-----
5 up up up
9 up up up

admin@scm# show ncs-state ha
ncs-state ha mode master
ncs-state ha node-id 9-1518035669
ncs-state ha connected-slave [ 5-1518043097 ]
```

### Step 2:

On the ESC, validate that the NETCONF connection to EM is established. Port 830 is the assigned port which can be used along with the netstat command to check NETCONF status:

```
[admin@ultram-vnfm1-esc-0 ~]$ netstat -an | grep 830
```

tcp	0	0 0.0.0.0:830	0.0.0.0:*	LISTEN
tcp	0	0 172.16.181.6:830	172.16.181.21:58761	ESTABLISHED
tcp	0	0 172.16.181.6:830	172.16.181.21:58760	ESTABLISHED
tcp	0	0 :::830	:::*	LISTEN

### Step 3:

Check the ESC data model and make sure both StarOS and EM state machines are showing **SERVICE\_ACTIVE\_STATE** and ensure all StarOS and EM VMs are in **VM\_ALIVE\_STATE**. Use the CLI below and search for <state\_machine> for each VM:

```
[admin@esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get esc_datamodel/opdata
```

```
<state_machine>
```

```
<state>SERVICE_ACTIVE_STATE</state>
```

Once all these points are verified, move ahead with recovering the VM.

## CF/SF VM Recovery from EM

### Step 1:

In the VPC CLI, synchronize the filesystem:

```
[local]UltraM# filesystem synchronize all
```

### Step 2:

Check the operation state of the card/VM that needs to be recovered with the following command:

```
[local]UltraM# show card table
```

### Step 3:

Put card/VM that needs recovery into standby state (if possible). If the card is in **booting** state, then proceed to next step:

For Control Function:

```
[local]UltraM# card switch from <> to <>
```

For Service Function:

```
[local]UltraM# card migrate from <> to <>
```

### Step 4:

SSH into Master EM and suspend the target CF or SF VM.

Section Element Manager in chapter "The Ultra M Solution" covers steps to identify vdu and vnfc parameters for the CF or SF VM.

**Note** Please note the numbering under VM's naming does not always match the card numbering under VPC. For example, card 3 under VPC does not always correspond to SF3 VM under OpenStack.

In the example below, SF5 had vdu session-function and vnfc BOOT\_generic\_di-chassis\_SF3\_1:

```
admin@scm# suspend-vnfc vdu session-function vnfc BOOT_generic_di-chassis_SF3_1
```

```
***** The prompt should be in config mode<
admin@scm(config)# resume-vnfc
```

### Step 5:

Monitor logs in EM and wait for "SERVICE\_UPDATED":

```
/var/log/em/vnfm-proxy/vnfm-proxy.log

<depid>9e227257-505c-46cc-8140-ef6195d44c9d</depid>
<event>
<type>SERVICE_UPDATED</type>
</event>
</escEvent>
</notification>
```

### Step 6:

Monitor log in ESC logs below:

```
tail /var/log/esc/yangesc.log
```

**Note** Please do not execute **resume** until service update is finished and SERVICE\_UPDATED is shown in vnfm-proxy.log in EM.

### Step 7:

Resume CLI from EM. Use the correct vdu and vnfc of the affected card.

In the example below **SF5** had vdu session-function and vnfc **BOOT\_generic\_di-chassis\_SF3\_1**:

```
admin@scm# resume-vnfc vdu session-function vnfc BOOT_generic_di-chassis_SF3_1
```

```
***** The prompt should be in config mode
admin@scm(config)# resume-vnfc
```

### Step 8:

Monitor logs in EM and wait for "SERVICE\_UPDATED":

```
tail -f /var/log/em/vnfm-proxy/vnfm-proxy.log

<depid>9e227257-505c-46cc-8140-ef6195d44c9d</depid>
<event>
<type>SERVICE_UPDATED</type>
</event>
</escEvent>
</notification>
```

### Step 9:

Monitor log in ESC logs below:

```
tail -f /var/log/esc/yangesc.log
```

After VM is recovered, perform basic health checks on the VM in VPC, EM and collect **show support details**.

**Note** ESC can be used to recover the CF/SF in specific situations, however, that may cause EM to go out of sync with the ESC and/or CF. This is not the recommended method unless otherwise instructed by Cisco.

## EM VM Recovery from ESC

Reported issues on EM can range from a specific EM not being available in a cluster to EM component failure (SCM, VNFM-PROXY, SLA). Depending on the type of failure, the recovery steps will be different. Recovery steps for specific scenarios can be found in the following section.

## EM VM Recovery Prechecks

Before any of the specific recovery steps, the following pre-checks should be conducted.

### Step 1:

Check EM cluster:

```

ncs --status | more
vsn: 4.1.1
SMP support: yes, using 2 threads
Using epoll: yes
available modules: backplane,netconf,cdb,cli,snmp,webui
running modules: backplane,netconf,cdb,cli,webui
status: started
cluster status:
mode: master
node id: 3-1521861535
connected slaves: 1
    
```

From this output, it can be seen that cluster is up as the connected slaves show 1. The NCS command below provides the same information:

```

@scm# show ncs-state ha
ncs-state ha mode master
ncs-state ha node-id 5-1522100017
ncs-state ha connected-slave [ 8-1522100130 ]
    
```

If no slaves are connected, it means EM is in the standalone state and no slaves will be displayed above. Ultra M Health Manager will report the following

```

[stack@ospd ~]$ cat /var/log/cisco/ultram-health/ultram_health_uas.report
-----
-----
VNF ID          | UAS Node | Status   | Error Info, if any
-----
-----
    
```

```
10.10.10.10 | vnf-em | XXX | EM: 1 is not part of HA-CLUSTER,EM is running
in standalone mode
```

### Step 2:

Log into the EM and check EM health status using the **show ems** command.

This output indicates cluster is up (the two node IDs 3 and 4 are displayed). However, the SLA and SCM service is down in one of the EMs:

```
admin@scm# show ems

EM VNFM
ID SLA SCM PROXY
-----
3 up up up
4 down down up
```

### Step 3:

Collect the operational data from the ESC.

With the ESC **get esc\_datamodel/opdata** command obtain the current status of the EMs. Possible EM VM states are alive or error:

```
[admin@esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get esc_datamodel/opdata

<state_machine>
<state>SERVICE_ACTIVE_STATE</state>
<vm_state_machines>
  <vm_state_machine>
    <vm_name>vnfd1-deployment_em_0_9ece3e13-0ab7-4543-80ed-
ce834e1d255e</vm_name>
    <state>error</state>
  </vm_state_machine>
```

Every Ultra M deployment contains three EM VMs. All three EMs should be in Active state, otherwise the ZooKeeper cluster management process will fail.

**Note** Although the three EM VMs are created, the NCS cluster shows only two VMs and ZooKeeper cluster shows three. The third EM VM is used only for ZooKeeper clustering.

## EM VM Recovery

If one of the EM VMs is showing components in DOWN state, recover the components by rebooting the VM via ESC:

```
[admin@esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli vm-action REBOOT
<em-vm-name>
```

If one of the EM VMs is in the Error state, use the command from ESC to reboot the EM:

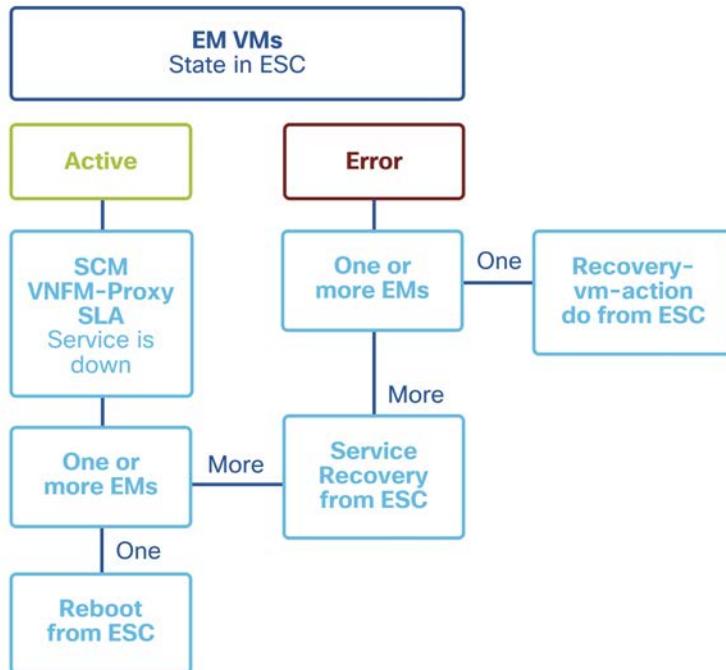
```
[admin@esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli recovery-vm-action DO
<em-vm-name>
```

If more than one of the EM VMs are in Error state, use the service recovery command from ESC to recover the EMs:

```
[admin@esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli svc-action RECOVER
<tenant name> <em-deployment-name>
```

For the details please refer to the VNF and EM section in the Troubleshooting chapter.

The following is a flowchart that shows steps to be taken while recovering the EMs having the failures described above.



# StarOS

## General Information

Each VPC-DI setup comprises of at least 2 CF (Control Function) VMs and the number of SF (Service Function) VMs depending on the size of the deployment.

Virtual Machines (VM) Description :

### CF - Control Function VM:

Two CF VMs act as an active: standby (1:1) redundant pair. The active CF is responsible for the following functions:

- Controller tasks
- Local context VPNMGR
- Local context (MGMT) and Di Network vNICs
- System boot image and configuration storage on vHDD
- Record storage on vHDD
- Out-of-Band (OOB) management (vSerial and vKVM) for CLI and logging

### SF - System Function VM:

SF VMs provide service context (user I/O ports) and handle protocol signaling and session processing tasks. A VPC-DI instance can contain up to 30 SF VMs.

Each SF VM dynamically takes on one of three roles as directed by the CF:

- Demux VM (flow assignments)
- Session VM (traffic handling)
- Standby VM (n+1 redundancy)

## Mapping the CF and SF VMs to the OpenStack Host

This section will explain how to identify OpenStack Host hosting the CF or SF VMs.

From the StarOS, run **show card table** that will list all available CF and SF:

```
[local]UGP# show card table
```

Slot	Card Type	Oper State	SPOF	Attach
1: CFC	Control Function Virtual Card	Active	No	
2: CFC	Control Function Virtual Card	Standby	-	
3: FC	1-Port Service Function Virtual Card	Standby	-	
4: FC	1-Port Service Function Virtual Card	Standby	-	
5: FC	1-Port Service Function Virtual Card	Standby	-	
6: FC	1-Port Service Function Virtual Card	Standby	-	

Choose the card/vm needed to be mapped and found on OpenStack level.

In this example, choose CF2:

Run **show card hardware <cardnumber> | grep -i UUID**

```
[local]UGP# show card hardware 2 | grep -i UUID
UUID/Serial Number      : CAF9E049-8739-483F-9C18-9840EE23EA8E
```

Login to VIM (C-VIM or OSP-D) and grep for the specific nova instance using the above UUID:

```
[stack@UGP]$ source corerc
nova list | grep -i CAF9E049-8739-483F-9C18-9840EE23EA8E
| caf9e049-8739-483f-9c18-9840ee23ea8e | vnfd1-deployment_c1_0_9ece3e13-0ab7-4543-80ed-ce834e1d255e | ACTIVE | - | Running | orchestr=172.16.180.12; ultram-vnfm1-di-internal1=192.168.1.16; mgmt=172.16.181.21 |
```

With a VM Name that corresponds to the UUID, the VM name can be used in Nova to show CLI to find the:

- a) Instance ID
- b) VM\_Host

```
nova show vnf-d1-deployment_c1_0_9ece3e13-0ab7-4543-80ed-ce834e1d255e
.....
| OS-EXT-SRV-ATTR:host                | ultram-compute-0.localdomain
| OS-EXT-SRV-ATTR:instance_name      | instance-000001cd
|
.....
```

From the above output, CF is hosted on the compute-0 and virsh instance is 000001cd. With this information, SSH to the host and verify instance is there, or perform any virsh or operating system checks if needed.

```
[stack@ultram-ospd ~]$ source stackrc
[stack@ultram-ospd ~]$ nova list
+-----+-----+-----+-----+-----+-----+
| ID                                     | Name                                     | Status | Task
State | Power State | Networks                                     |
+-----+-----+-----+-----+-----+-----+
| a3a48584-7eee-4455-a                 | compute-0                               | ACTIVE | -
ctlplane=192.200.0.108 |
| 35b6f193-07e6-44a9-82c1-1           | compute-1                               | ACTIVE | -
ctlplane=192.200.0.107 |
| 83728dd2-1b2a-4204-94a6-b           | controller-0                             | ACTIVE | -
ctlplane=192.200.0.113 |
| 53512dc5-307f-44a3-9fac-0           | controller-1                             | ACTIVE | -
ctlplane=192.200.0.104 |
| 702fa698-450f-4acb-8aa6-d           | controller-2                             | ACTIVE | -
ctlplane=192.200.0.110 |
| 4044ab1c-5a96-4402-8fec-9           | osd-compute-0                           | ACTIVE | -
Running |
```



- `show system uptime`

#### Monitoring Related:

- `show snmp traps history verbose`
- `show alarm outstanding all`
- `show logs`
- `show crash list`

#### Redundancy Related:

- `show rct stats verbose`
- `show session recovery status verbose`

#### VM related:

- `show card table`
- `show card hardware`
- `show port table`

#### HD RAID related:

- `show hd raid verbose`
- `show active-charging edr-udr-file statistics / show cdr statistics`
- `dir /hd-raid/records`
- `show snmp trap history verbose`

## StarOS (VPC-DI specific) Cloud Configuration

This section will concentrate on the Virtual Platform that is not covered by Cisco ASR5000/5500 Mobility Troubleshooting Guide.

The cloud commands below are specific for VPC-DI systems and can be used to obtain more information about the configuration:

```
[local]# show cloud configuration

Card 1:
  Config Disk Params:
  -----
  No config disk available
  Local Params:
  -----
  CARDSLOT=1
  CARDTYPE=0x40010100
  CPUID=0
```

Card type will be CF or SF. This value is hardcoded to:

CF = 0x40010100

SF = 0x42020100

The CPUID field identifies which boxer CPU complex the software is running on. Currently, only CPUID 0 is supported.

The cloud-specific hardware configuration can be displayed by utilizing the **show cloud** commands. To prevent resource allocation issues, it is important that all CF VMs used in VPC-DI are allocated the same vCPU and memory size, same applies for all SF VMs:

```
[local]# show cloud hardware

Card 1:
  CPU Nodes           : 1
  CPU Cores/Threads   : 8
  Memory              : 8192M (qvpc-di-medium)
  Hugepage size       : 2048kB
  cpeth0              :
    Driver             : virtio_net
  loeth0              :
    Driver             : virtio_net
Card 2:
  CPU Nodes           : 1
```

```

CPU Cores/Threads      : 8
Memory                 : 8192M (qvmc-di-medium)
Hugepage size         : 2048kB
cpeth0                 :
  Driver                : virtio_net
loeth0                 :
  Driver                : virtio_net

```

Card 3:

```

CPU Nodes              : 1
CPU Cores/Threads     : 16
Memory                 : 16384M (qvmc-di-medium)
Hugepage size         : 2048kB
cpeth0                 :
  Driver                : enic
port3_10               :
  Driver                : enic

```

[local]# **show cloud hardware optimum**

```

CPU Nodes              : 1
CPU Cores/Threads     : 8
Memory                 : 16384M (qvmc-di-medium)
Hugepage size         : 2048kB
NIC driver             : virtio_net

```

Optimum cloud configuration for SF

```

CPU Nodes              : 1
CPU Cores/Threads     : 16
Memory                 : 32768M (qvmc-di-medium)
Hugepage size         : 2048kB
NIC driver             : virtio_net

```

[local]# **show cloud hardware test card\_number**

```

Card 1:
  CPU Nodes            : 1
* CPU Cores/Threads   : 8           Optimum value is 8
* Memory               : 8192M (qvmc-di-medium Optimum value is 16384
Hugepage size         : 2048kB
cpeth0                 :
  Driver                : virtio_net
loeth0                 :
  Driver                : virtio_net

```

## Di Network verification

SSH into the public IP of the Di system to log into the active CF.

Check network connectivity between the VMs (Di internal network) as below. Use the below with caution.

The command below helps to get into the debug shell and run a hidden command for CF card 2.

```
[Local] cf-1# configure
[Local] cf-1(config) # tech-support test-commands password <>
[Local] cf-1(config) # cli hidden
[Local] cf-1(config) # end
[Local] cf-1# cli test-commands password <>
[Local] cf-1# debug shell card 2 cpu 0
```

Executing the **ip addr** command, obtain the Di internal network IP of the active CF. It will be in the format: 172.16.X.1.

Following this example, if there are 2 CF and 4 SF the IP allocation will be:

CF 1	172.16.0.1
CF 2	172.16.1.1
SF 1	172.16.2.1
SF 2	172.16.3.1
SF 3	172.16.4.1
SF 4	172.16.5.1

For example for CF1:

```
cpeth0: <BROADCAST,MULTICAST,UP,10000> mute 7100 qdisc prio qlen 20000
link/ether fa:16:3e:d6:65:e9 brd ff:ff:ff:ff:ff:ff
inet 172.16.0.1/18 brd 172.16.63.255 scope global cpeth0
```

From the active CF, ping each of these CFs and SFs IPs to check if the VM to VM communication is healthy. There is access to “shell into” any of the cards for SF/CF using the debug shell command above to check connectivity between specific VMs (SFs and CFs):

```

local|cf-1# debug shell card 2 cpu 0
vpc-di:card2-cpu0#
vpc-di:card2-cpu0#
vpc-di:card2-cpu0# ping 172.16.2.1
PING 172.16.2.1 (172.16.2.1) 56(84) bytes of data.
64 bytes from 172.16.2.1: icmp_seq=1 ttl=64 time=0.500 ms dscp=0 [BE]
64 bytes from 172.16.2.1: icmp_seq=2 ttl=64 time=0.998 ms dscp=0 [BE]
64 bytes from 172.16.2.1: icmp_seq=3 ttl=64 time=1.00 ms dscp=0 [BE]
64 bytes from 172.16.2.1: icmp_seq=4 ttl=64 time=0.996 ms dscp=0 [BE]
64 bytes from 172.16.2.1: icmp_seq=5 ttl=64 time=0.997 ms dscp=0 [BE]
64 bytes from 172.16.2.1: icmp_seq=6 ttl=64 time=0.999 ms dscp=0 [BE]
^C
--- 172.16.2.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5004ms
rtt min/avg/max/mdev = 0.500/0.915/1.003/0.188 ms

```

The **show heartbeat stats** command can also be used on SFs to get the round trip times (RTT) between the SFs and between the SF and CF on the Di internal network. This can verify the Di internal network, make sure there are no dropped frames in ‘Drops’ columns, and also measure the RTT:

```

# show heartbeat stats card 3 cpu 0
Card 3 Test Statistics:
Dest  TotalPkt  JumboPkt  TotalDrops  JumboDrops  LongRTT  AverageRTT
2      1654      827       0            0            0         0.566ms
4      1642      821       0            0            0         0.377ms
5      1654      827       0            0            0         0.365ms
6      1649      825       0            0            0         0.358ms

Last 10 RTT in ms (starting from most current):
2      1.414      1.505      0.797      0.488      1.307      0.754      0.585      0.655
1.64      2.785
4      0.938      0.694      0.330      0.421      1.255      0.640      0.542      0.503
0.528      1.437
5      0.652      0.672      0.342      0.427      1.192      0.563      0.524      0.590
0.548      1.497

```

```
6      0.455      0.486      0.530      0.401      1.114      0.473      0.432      0.448
0.556      1.507
```

If BFD protocol is used for the contexts, the command below can be used to verify that all the service port connections to the BFD peer are UP. This is a good command to verify Service Network interfaces:

```
# show bfd neighbor
Tuesday November 17 18:13:54 PST 2015
All the time intervals are in ms
OurAddr      NeighAddr      LD/RD      DcttTime State      Intf
172.23.1.31  172.23.1.1     1/1090519526 900      Up      sgi1_3
172.23.1.41  172.23.1.1     2/1090519527 900      Up      sgi1_4
172.23.1.51  172.23.1.1     3/1090519528 900      Up      sgi1_5
172.23.1.61  172.23.1.1     4/1090519529 900      Up      sgi1_6
172.23.1.71  172.23.1.1     5/1090519530 900      Up      sgi1_7
```

### VM (CF/SF) Boot Process

Bootup logs of CF and SF can be collected in Ultra M Service Platform using the Virtual Shell (virsh) commands. The general process of collecting bootup logs is covered in Troubleshooting Tech Node - How to Collect Logs from the VM Boot:

<https://www.cisco.com/c/en/us/support/docs/wireless/asr-5700/212432-how-to-collect-logs-from-vm-bootup.html>

The following steps cover major commands to collect bootup logs:

- Identify a compute running CF/SF, instance id and management IP.
- SSH to the compute with heat-admin user and switch to super user.
- Attach to serial1 of the VM with **virsh console instance-<number> serial1**

## Integration with EM/ESC

In VPC DI of Ultra M Service Platform, active ESC communicates with active CF via active EM. The communication consists of the following:

- NETCONF connection between EM/NSO and CF mgmt virtual IP. This is using the mgmt network.
- Connection over the ZooKeeper framework between VNFM Proxy Agent (CF) and VNFM proxy (EM). This is using the orchestration network.

The following are components on CF that provide integration among active CF, EM and ESC:

- confdmgr
- vnfmpoxy agent
- emctrl

**Confdmgr** is the process that enables NETCONF on the CF. It is associated with a CDB database which contains the cloud configuration. This is configured by EM through NETCONF. As usual with NETCONF, the EM can in return subscribe to receive VNFRs, records of the operational state of all the elements.

The following command displays the state of Confd manager:

```
[local]UltraM# show confdmgr

State Information
-----
State                Started
Subscriptions          29
Last successful id    1521-151716-630599
Last failed id        None
Autosave url         Not configured
Username              admin

Statistics
-----
Triggers              1
```

```

Notifications          29
Successful notifications 4
Failed notifications   0
Unexpected

```

**Vnfmproxy agent** is the process on CF that communicates to the VNFMPROXY on the active EM. It's used for the following:

- To inform VNFM proxy in EM about certain events that ESC should know about: VM start/stop/reboot initiated by Ultra Service Platform.
- To receive faults from NFVI which are reported by ESC, and are proxied towards CF by EM's VNFM proxy.

The following command displays the state of VNFM proxy manager:

```

local]UltraM# show vnfm-proxy-agent status
VNFM Proxy Agent Status:
  State      : online
  Connected to : 172.16.180.3:2181
  Bind Address : 172.16.180.19:58752
  VNFM Proxy address count: 3

```

**emctrl** is the process on CF that translates NFVI events (received through VNFM proxy agent) into StarOS events (card down, etc).

The following command displays the state of EM controller:

```

[local]UltraM# show emctrl status
emctrl status:
emctrl in state: ALIVE

```

**emctrl** enables a query about the card inventory (VDU) of EM through VNFM, and it will subscribe to notifications for these VDUs through the VNFMa, so it is aware of changes on the EM/ESC/NFVI level.

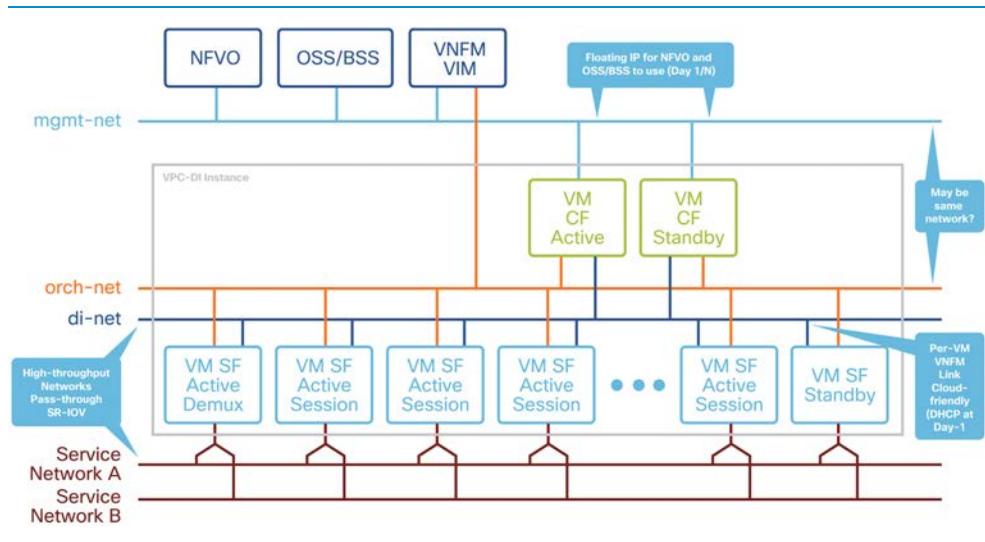
The following command display VDU list:

```
[local]UltraM# show emctrl vdu list
Showing emctrl vdu
card[01]: name[CFC_01 ] uuid[6EF5BF1C-25D8-4C7E-862C-3E87A2371848 ]
card[02]: name[CFC_02 ] uuid[CAF9E049-8739-483F-9C18-9840EE23EA8E ]
card[03]: name[SFC_03 ] uuid[308B0303-F665-4368-9640-679D7FE13189 ]
card[04]: name[SFC_04 ] uuid[546A6194-1B99-448F-8FE0-0CC54C42C154 ]
```

## Di and Service Network Troubleshooting

A minimum configuration for a VPC-DI instance requires four SFs - two active, one demux and one standby.

Here is the reference diagram for the Di Network:



The reliability and performance of the Di network are critical to the reliability and performance of VPC-DI. The Di Network is used for internal control, signaling, and

bearer traffic. Bearer traffic may traverse the Di network multiple times, so any packet loss in the Di network would impact the perceivable packet loss of VPC-DI as a whole.

## Basic Connectivity

Traditional ping tests are probably the most basic troubleshooting that can be performed. Pinging the next hop address(s) from the context that you troubleshooting is a good starting point. In addition, the following commands relate to connectivity:

- ping
- ping6
- show ip route
- show ipv6 route
- show ip arp
- show ipv6 neighbors (ipv6 version of show ip arp)
- show ip interface [summary]
- show ipv6 interface [summary]
- traceroute
- traceroute6

## Monitoring

In general, the active CF monitors for faults of other VMs and will take recovery action if a VM is declared unresponsive. Multiple monitoring methods are used as described below, some are purely for monitoring, alarming, or debugging purposes while others take an active role in determining faults and triggering recovery action. Heartbeats run every second while AFD every 100 milliseconds.

**CF Mastership:**

Both CF VMs run a process called “masterd” which is a daemon for determining active/standby roles of the 2 CFs. Once VM boots up, it will send a broadcast to figure out Master / Standby mode.

**Heartbeats CF-CF:**

Each CF sends heartbeats once per second over the IPC mechanism using UDP packets with priority 7. The IPC mechanism picks the UDP port number which is not fixed. Packets originate and terminate in the “hatsystem” userspace process.

Each heartbeat is a round trip active->standby->active or standby->active->standby. Heartbeats are sent once per second and if a response is not received within approximately 1.2 seconds it is considered a lost heartbeat.

In summary:

- Standby CF is declared failed after between 2.2 and 3.1 seconds having lost 2 heartbeats with a 1.2 second timeout each and a 3rd heartbeat timeout of 0.2 seconds.
- Active CF is declared failed after between 4.2 and 5.1 seconds having lost 4 heartbeats with a 1.2 second timeout each and a 5th heartbeat timeout of 0.2 seconds.

**Reverse Heartbeats SF-CF:**

Each SF tracks received heartbeats from CFs. If the SF does not receive a heartbeat for 7 seconds it will assume both CFs have failed and the SF is no longer under control of a CF.

In summary, SF is declared failed once all these conditions are met:

- Loss of 2 consecutive heartbeats with a 1.2 second timeout each and a 3rd heartbeat timeout of 0.2 seconds.

- CF hasn't received an IPC advertisement from that SF recently (usually 5-8 seconds).

### **Advanced Fault Detection (AFD):**

In addition to normal heartbeats, the active CF sends ICMP packets at a faster rate (10 per second) using priority 6 to check the health of other VMs. The CF uses a directed broadcast ICMP to the Di net's broadcast address. Unicast ICMP responses are received from all VMs back into the CF.

### **Di Network Health**

Specific to VPC-DI, additional monitoring was added to help debug network reliability issues. This monitoring is intended to find issues with the reliability of the infrastructure providing interconnection between VMs over longer time periods (minutes to hours).

#### **show heartbeat stats**

The response contains 2 components, the 10 most recent heartbeats and the 16 most recent AFD attempts.

Code	Description	Cause
<b>RV</b>	Reply, Valid	Normal reply
<b>RO</b>	Reply, Out of order	Response received, but out of order
<b>RI</b>	Reply, Ignored	Response for a destination already declared dead
<b>RD</b>	Reply, Delayed	Response, but hat process has been excessively delayed (scheduling or CPU contention)
<b>BV</b>	Bounce, Valid	Normal timeout
<b>BI</b>	Bounce, Ignored	Timeout, but: <ul style="list-style-type: none"> <li>▪ Response for a newer heartbeat already received</li> <li>▪ Destination that has just started</li> <li>▪ Destination already declared dead</li> </ul>
<b>DN</b>	Dead, Non-critical	Declared dead (non-critical task)
<b>DC</b>	Dead, Critical	Declared dead (critical task or VM)
<b>EE</b>	Error	Internal Error

Example: Standby CF VM stopped

Log messages

```

2017-Feb-02+10:27:52.729 [hat 3067 warning] [1/0/6082 <hatsystem:0>
hatsystem_afd.c:325] [software internal system syslog] Advanced fault detection
missed multiple packets on cpu 2/0.

hb 2198 :43.112, RV, - 00000000 00000000, cp 0, tx 0ms, rx 13ms, rtt 13ms
hb 2199 :44.112, RV, - 00000000 00000000, cp 0, tx 1ms, rx 0ms, rtt 1ms
hb 2200 :45.126, RV, - 00000000 00000000, cp 0, tx 0ms, rx 4ms, rtt 4ms
hb 2201 :46.127, RV, - 00000000 00000000, cp 0, tx 0ms, rx 2ms, rtt 2ms
hb 2202 :47.128, RV, - 00000000 00000000, cp 0, tx 1ms, rx 1ms, rtt 2ms
hb 2203 :48.129, RV, - 00000000 00000000, cp 0, tx 0ms, rx 1ms, rtt 1ms
hb 2204 :49.129, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2205 :50.130, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2206 :51.136, RV, - 00000000 00000000, cp 0, tx 1ms, rx 0ms, rtt 1ms
hb 2207 :52.131, RV, - 00000000 00000000, cp 0, tx 0ms, rx 0ms, rtt 0ms
    
```

```

afd 21738 :51.083, rtt 14ms
afd 21739 :51.184, rtt 3ms
afd 21740 :51.284, rtt 2ms
afd 21741 :51.385, rtt 1ms
afd 21742 :51.486, rtt 0ms
afd 21743 :51.586, rtt 2ms
afd 21744 :51.687, rtt 0ms
afd 21745 :51.787, rtt 1ms
afd 21746 :51.887, rtt 1ms
afd 21747 :51.988, rtt 2ms
afd 21748 :52.104, rtt 3ms
afd 21749 :52.204, rtt 1ms
afd 21750 :52.305, rtt 1ms
afd 21751 :52.405, tout
afd 21752 :52.505, tout
afd 21753 :52.628, tout
2017-Feb-02+10:27:54.059 [hat 3066 info] [1/0/6082 <hatsystem:0> hatsystem_afd.c:373]
[software internal system syslog] Advanced fault detection lost packet sequence 21751
sent :52.405 to cpus: 2/0
2017-Feb-02+10:27:54.160 [hat 3066 info] [1/0/6082 <hatsystem:0> hatsystem_afd.c:373]
[software internal system syslog] Advanced fault detection lost packet sequence 21752
sent :52.505 to cpus: 2/0
2017-Feb-02+10:27:54.261 [hat 3066 info] [1/0/6082 <hatsystem:0> hatsystem_afd.c:373]
[software internal system syslog] Advanced fault detection lost packet sequence 21753
sent :52.628 to cpus: 2/0
.....
2017-Feb-02+10:27:54.273 [hat 3013 warning] [1/0/6082 <hatsystem:0> hat_hb_lib.c:923]
[software internal system syslog] Task hatcpu/20 on cpu 2/0 missed heartbeat sequence
2208 HASM-CNT -1 ns_age 5 bounce_code N/A uptime 2276.
.....
2017-Feb-02+10:27:55.309 [hat 3014 critical] [1/0/6082 <hatsystem:0>
hat_hb_lib.c:875] [software internal system syslog] HAT instance 0 found Critical
task hatsystem/1 failed on cpu 2/0. Missed heartbeat sequence 2149 ns_age 5
bounce_code N/A uptime 2276.

```

After the 2nd consecutive heartbeat loss to hatsystem, the CF is declared dead and the recovery process is started.

### show cloud monitor

This command provides a summary and detailed information for long-term monitoring.

As a note, the “Health” column shows “Bad” if either the 5 minute or 60-minute loss is greater than 1%.

Di Network should be available and healthy all the time in order for CFs and SF communicate to each other. The **show cloud monitor** command has both a summary and detailed information for long-term monitoring.

Below everything is working fine:

```
[local]UGP# show cloud monitor di-network detail
Card 3 Heartbeat Results:
ToCard  Health  5MinLoss  60MinLoss
  1      Good    0.00%    0.00%
  2      Good    0.00%    0.00%
  4      Good    0.00%    0.00%
  5      Good    0.00%    0.00%
  6      Good    0.00%    0.00%

Dest  TotalPkt  JumboPkt  TotalDrops  JumboDrops  LongRTT  AverageRTT
  1    238554   119277      0           0           0        1.373ms
  2    238545   119273      0           0           0        0.913ms
  4    238552   119276      0           0           0        1.223ms
  5    238454   119227      0           0           0        1.334ms
  6    238551   119276      0           0           0        0.965ms

Last 10 RTT in ms (starting from most current):
  1    1.884    1.885    1.826    1.887    1.883    1.840    1.887    1.886
1.891    1.892
  2    0.882    0.882    0.827    0.884    0.891    0.837    0.890    0.888
0.885    0.873
  4    0.886    0.885    0.829    0.886    0.894    0.839    0.892    0.890
0.891    0.875
  5    1.881    1.882    1.823    1.884    1.880    1.837    1.884    1.883
1.883    1.885
  6    0.877    0.878    0.822    0.880    0.885    0.833    0.884    0.884
0.875    0.869

Card 4 Heartbeat Results:
ToCard  Health  5MinLoss  60MinLoss
  1      Good    0.00%    0.00%
  2      Good    0.00%    0.00%
  3      Good    0.00%    0.00%
  5      Good    0.00%    0.00%
  6      Good    0.00%    0.00%

Dest  TotalPkt  JumboPkt  TotalDrops  JumboDrops  LongRTT  AverageRTT
  1    238556   119278      0           0           0        1.085ms
  2    238546   119273      0           0           0        1.062ms
  3    238553   119277      0           0           0        0.959ms
```

```

5    238455    119228        0         0         0    1.070ms
6    238552    119276        0         0         0    1.119ms

```

Last 10 RTT in ms (starting from most current):

```

1    1.378    1.669    1.672    1.612    1.615    1.309    1.550    1.554
1.615    2.666
2    1.383    0.664    0.677    0.605    0.604    1.313    1.555    0.552
0.608    0.671
3    1.370    1.664    1.665    1.602    1.604    1.302    1.539    1.549
1.603    1.664
5    1.376    1.667    1.669    1.607    1.612    1.307    1.545    1.552
1.610    1.667
6    1.381    1.671    0.673    1.617    1.617    1.311    1.552    1.556
1.620    0.667

```

Card 5 Heartbeat Results:

ToCard	Health	5MinLoss	60MinLoss
1	Good	0.00%	0.00%
2	Good	0.00%	0.00%
3	Good	0.00%	0.00%
4	Good	0.00%	0.00%
6	Good	0.00%	0.00%

Dest	TotalPkt	JumboPkt	TotalDrops	JumboDrops	LongRTT	AverageRTT
1	238456	119228	0	0	0	1.526ms
2	238456	119228	0	0	0	0.964ms
3	238456	119228	0	0	0	0.974ms
4	238456	119228	0	0	0	1.454ms
6	238456	119228	0	0	0	1.014ms

Last 10 RTT in ms (starting from most current):

```

1    1.759    1.875    1.815    1.817    1.864    1.763    1.752    1.866
1.745    1.747
2    0.763    0.889    0.815    0.821    0.866    0.767    0.755    0.868
0.757    0.755
3    0.768    0.898    0.823    0.825    0.878    0.771    0.774    0.872
0.767    0.759
4    0.766    0.893    0.817    0.823    0.875    0.769    0.771    0.870
0.762    0.757
6    0.759    0.881    0.810    0.817    0.862    0.764    1.756    0.864
0.748    0.750

```

Card 6 Heartbeat Results:

ToCard	Health	5MinLoss	60MinLoss
1	Good	0.00%	0.00%
2	Good	0.00%	0.00%
3	Good	0.00%	0.00%
4	Good	0.00%	0.00%

```

5      Good      0.00%      0.00%

Dest  TotalPkt  JumboPkt  TotalDrops  JumboDrops  LongRTT  AverageRTT
1     238558   119279    0           0           0        1.136ms
2     238550   119275    0           0           0        1.513ms
3     238557   119279    0           0           0        1.207ms
4     238557   119279    0           0           0        1.143ms
5     238459   119230    0           0           0        1.108ms
Last 10 RTT in ms (starting from most current):
1     0.826   0.823   0.828   1.851   0.780   0.826   0.888   0.887
0.781   0.771
2     1.818   1.818   1.823   1.853   1.769   1.817   1.874   1.881
1.766   1.762
3     0.814   0.819   0.822   1.847   0.775   0.821   0.882   0.882
0.776   0.766
4     0.809   0.815   0.817   1.843   0.770   0.818   0.875   0.878
0.770   0.762
5     0.822   0.821   0.824   1.849   0.777   0.824   0.885   0.884
0.779   0.768

show iftask stats summary

```

PPS numbers per SF for Service and Di Net ports.

Packet drops (and NDR reporting)

## Troubleshooting IFTASK

Di network and service ports are controlled by the DPDK Internal Forwarder (IFTASK) process. IFTASK is a software component that will be responsible for packet IO in the Cisco VPC. IFTASK is responsible for receiving packets from the device or port. It may be a virtual or physical pass-through device and provides a fast path for packet processing in User Space, bypassing the Linux Networking and Linux IP stack completely.

The following command displays statistics on IFTASK

```
show iftask stats summary
```

The output provides statistics per SF for Service and Di Network ports.

### **VPC-DI Packet Flow**

This topic provides a basic overview of how a packet traverses inside a VPC-DI network.

Refer VPC-DI System Administration Guide, Section: Packet Flow:

<https://www.cisco.com/c/en/us/support/wireless/ultra-gateway-platform/products-installation-and-configuration-guides-list.html>

# Virtual Network Function Manager & Element Manager

## Virtual Network Function Manager (VNFM)

In the Ultra M deployments, the VNFM role is played by Cisco Elastic Services Controller (ESC). To ensure that ESC is in a healthy state, execute the **/opt/cisco/esc/esc-scripts/health.sh** script. If the ECS is not healthy, then proceed with the recovery steps below.

**Note** Backup of ESC Database (DB) is a mandatory step in troubleshooting of the ESC.

Below is a command to backup ESC DB into db\_backup.tar file under /home/admin:

```
[admin@ultram-vnfml-esc ~]$ sudo /opt/cisco/esc/esc-scripts/esc_dbtool.py backup --
file scp://admin@127.0.0.1:/home/admin/db_backup.tar
```

For example, a successful DB backup is shown below:

```
admin@127.0.0.1's password:
2018-01-18 19:15:37,272: esc_dbtool.py(542): INFO: backup
/tmp/esc_backup_2018_01_18_19_15_32.tar.bz2 ssh://admin@127.0.0.1:/home/backup.tar
2018-01-18 19:15:37,785: esc_dbtool.py(353): INFO: Status postgresql-9.4, rc= 0
2018-01-18 19:15:37,785: esc_dbtool.py(242): INFO: Dump database to /tmp/.tmp_db_6935
2018-01-18 19:15:37,998: esc_dbtool.py(258): INFO: Copy cdb to /tmp/.tmp_db_6935
2018-01-18 19:15:38,009: esc_dbtool.py(261): INFO: Copy truststore files to
/tmp/.tmp_db_6935
2018-01-18 19:15:38,012: esc_dbtool.py(264): INFO: Copy esc ui application files
to/tmp/.tmp_db_6935
2018-01-18 19:15:38,015: esc_dbtool.py(267): INFO: Create backup file:
/tmp/esc_backup_2018_01_18_19_15_32.tar.bz2
2018-01-18 19:15:38,291: esc_dbtool.py(518): INFO: Copy
/tmp/esc_backup_2018_01_18_19_15_32.tar.bz2 to ssh://admin@127.0.0.1:/home_backup.tar
```

## Recovering ESC from Fault State

There are scenarios where the ESC boots and ends up in a fault state. Some of the possible reasons are listed below.

### Low Disk Space

Check the disk space with the **df -kh** command. Identify the file on the disk which is causing this behavior. Check the top ten files in the **/var directory**. GZIP and remove them as necessary to clear space.

```
du -a /var | sort -n -r | head -n 10

[admin@ultram-vnfm1-esc ~]$ sudo du -a /var | sort -n -r | head -n 10
359500 /var
217720 /var/log
215024 /var/log/esc
120764 /var/lib
95756 /var/log/esc/mona
83168 /var/log/esc/mona/mona.log
72216 /var/log/esc/vimmanager
72212 /var/log/esc/vimmanager/vimmanager.log
60984 /var/lib/tomcat6
60980 /var/lib/tomcat6/webapps
```

Validate if the **logrotate** is working well. Check the size and content of **/var/log/messages**.

Check the directory listing of **/etc/rsyslog.d** and if the following new custom files are present:

00-escmanager.conf, 01-messages.conf and 02-mona.conf

They should look like the following:

```
[esc-esc-0 rsyslog.d]$ cat 01-messages.conf

$ModLoad imfile
$InputFileName /var/log/messages
$InputFileTag messages:
$InputFileStateFile stat-messages
```

```

$InputRunFileMonitor

$template messages_log, "%syslogtag::% %msg%"
if $programname == 'messages' then @@10.248.164.17:514;messages_log
if $programname == 'messages' then ~

```

This is expected output.

```

esc-esc-0 rsyslog.d]$ cat 02-mona.conf
$ModLoad imfile
$InputFileName /var/log/esc/mona/mona.log
$InputFileTag mona:
$InputFileStateFile stat-mona
$InputRunFileMonitor
$template mona_log, "%syslogtag::% %msg%"
if $programname == 'mona' then @@10.248.164.17:514;mona_log
if $programname == 'mona' then ~

```

This is expected output.

In cases when the logrotate is not working, the following change should be made in the above files:

a) In the file 01-messages.conf following line: "*if \$programname == 'messages' then ~*" has to be replaced with:

```
if $programname == 'messages' then stop
```

b) 02-mona.conf following line: "*if \$programname == 'mona' then ~*" has to be replaced with

```
if $programname == 'mona' then stop
```

After the two changes have been made and saved, restart the rsyslog service:

```

[admin@ultram-vnfml-esc rsyslog.d]$ sudo service rsyslog status
rsyslogd (pid 1433) is running...
[admin@ultram-vnfml-esc rsyslog.d]$

```

Check the `/var/log/esc/confd/netconf.trace` file growth and make sure the configuration files `/opt/cisco/esc/esc-confd/confd.conf` do not have trace enabled.

**Note** Please note the ESC DB file located at `/opt/cisco/esc/.DRBD_IMG` is limited to 3Gb Max.

### Network Related Issues

Check if the ESC default gateway is reachable. Identify the default gateway with the `netstat -r`. Ping it.

If ESC is not pingable, make sure to check the external default gateway and OpenStack L3 router. Also verify that ESC is reachable on the management IP.

Check the `/var/log/messages` for any error or fault messages. For example, the following messages could be attributed to network connectivity issues:

```
Dec 3 23:59:52 ESC-ESC-1 Keepalived_vrrp[1645]: VRRP_Script(kad_check_network)
failed
Dec 4 00:00:04 ESC-ESC-1 Keepalived_vrrp[1645]: VRRP_Instance(esc_control_eth_vrrp)
Now in FAULT state
Dec 4 00:00:04 ESC-ESC-1 Keepalived_vrrp[1645]: VRRP_Group(vsoc_esc_vipg) Syncing
instances to FAULT state
```

Check if the ESC is able to reach the OpenStack Keystone server. This can be validated by looking for error messages in:

```
/var/log/esc/esc_monitor.log

/var/log/esc/vimmanager/vimmanager.log

/var/log/esc/escmanager.log
```

When ESC is able to successfully connect to the OpenStack Keystone server, messages similar to the following can be seen in `escmanager.log`:

```
19:48:50,859 04-Dec-2017 VimConnectionMonitor INFO
[OpenStackDriver.java:getOpenStackAuthenticationId():688] [tid=] Status code returned
from Openstack for create token: 200
```

After fixing the network issue, the following steps should be performed:

1. Remove these two files from `/opt/cisco/esc` and reboot the ESC VM:

```
/opt/cisco/esc/esc_haagent_pending.state
/opt/cisco/esc/esc_haagent.lock
```

2. Check if the ESC has allocated IP addresses on the ethernet interfaces.
3. Check the log messages in the file:

```
/var/log/esc/error_escmanager.log
```

4. Finally, to make sure ESC is working correctly, run the **sudo health.sh** script to verify health status is correct.

### ESC Service in SERVICE\_ERROR\_STATE

Ultra M deployment has seen the following behavior on ESC for EM deployment.

All VMs are ALIVE in ESC (and fully operational), but the service is in an ERROR state, as can be seen by the ESC `opdata`:

```
[admin@ultram-vnfm1-esc ~] /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get
esc_datamodel/opdata
```

The relevant output (for EM service) is here:

```
<state_machine>
  <state>SERVICE_ERROR_STATE</state>
  <vm_state_machines>
    <vm_state_machine>
      <vm_name>DEPLOYM_PGWL_0_979e7730-e2d4-42cb-ad6d-d08492ce5158</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
    <vm_state_machine>
      <vm_name>DEPLOYM_PGWL_0_b889ca23-12be-46df-be4f-4a4de7ac65b6</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
    <vm_state_machine>
      <vm_name>DEPLOYM_PGWL_0_f9d5badd-89a5-4cc0-a46e-f88b7d35f6ea</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
  </vm_state_machines>
</state_machine>
</deployments>
```

This is not a good state. It might impact automatic VM recovery by ESC.

To proceed with the recovery, the Service Name from the data model is needed:

```
[admin@ultram-vnfml-esc]/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get
esc_datamodel/opdata | grep -i deployment_name

    <deployment_name>PCF10-DEPLOYMENT-1.0.0-1</deployment_name>

    <deployment_name>vnfd1-deployment-em</deployment_name>
```

### Recovering the Service in SERVICE\_ERROR\_STATE

The following ESC recovery command can be used to recover the service that is in Service Error state:

```
[admin@ultram-vnfml-esc]/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli svc-action
RECOVER core vnfd1-deployment-em
```

This will trigger the service to move to an INERT state:

```

<state_machine>
  <state>SERVICE_INERT_STATE</state>
  <vm_state_machines>
    <vm_state_machine>
      <vm_name>PGW10-DEPLOYM_PGW1_0_979e7730-e2d4-42cb-ad6d-
d08492ce5158</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
    <vm_state_machine>
      <vm_name>PGW10-DEPLOYM_PGW1_0_b889ca23-12be-46df-be4f-
4a4de7ac65b6</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
    <vm_state_machine>
      <vm_name>PGW10-DEPLOYM_PGW1_0_f9d5badd-89a5-4cc0-a46e-
f88b7d35f6ea</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
  </vm_state_machines>
</state_machine>
</deployments>

```

After recovery, deployment service should move to **ACTIVE** status:

```

<state_machine>
  <state>SERVICE_ACTIVE_STATE</state>
  <vm_state_machines>
    <vm_state_machine>
      <vm_name>PGW10-DEPLOYM_PGW1_0_979e7730-e2d4-42cb-ad6d-
d08492ce5158</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
    <vm_state_machine>
      <vm_name>PGW10-DEPLOYM_PGW1_0_b889ca23-12be-46df-be4f-
4a4de7ac65b6</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
    <vm_state_machine>
      <vm_name>PGW10-DEPLOYM_PGW1_0_f9d5badd-89a5-4cc0-a46e-
f88b7d35f6ea</vm_name>
      <state>VM_ALIVE_STATE</state>
    </vm_state_machine>
  </vm_state_machines>
</state_machine>

```

```
</vm_state_machine>  
</vm_state_machines>  
</state_machine>  
</deployments>
```

## ESC Logs

VNF deployment creation/service update/deletion logs can be found at following path:

```
/var/log/esc/yangesc.log
```

VM monitoring logs can be found at following path:

```
/var/log/esc/mona/mona.log
```

ESC related logs can be found at following path:

```
/var/log/esc/escmanager.log
```

## Additional ESC Commands

Check operational data model:

```
/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get esc_datamodel/opdata
```

Push the config or edit deployment model:

```
/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli edit-config xxxx.xml
```

Recovering the service deployment:

```
/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli svc-action RECOVER core vnfdl-deployment  
-em
```

Reboot/Start/stop the VM:

```
/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli vm-action vm-action
<STOP|START|REBOOT|DISABLE_MONITOR|ENABLE_MONITOR> <vm name>
```

Recovery VM action:

```
/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli recovery-vm-action <DO> <vm name>
```

Check Confd configuration:

```
[admin@ultram-vnfml-esc esc-cli]$ /opt/cisco/esc/confd/bin/confd_cli -u admin -C
admin connected from 10.131.32.255 using ssh on ultram-vnfml-esc

ultram-vnfml-esc# show running-config

aaa authentication users user admin

uid          9000

gid          100

password     $6$90hp$/hG0YdAYo2JxX2UVcjQDSATurRK.

ssh_keydir   /var/confd/homes/admin/.ssh

homedir      /var/confd/homes/admin
```

### Obtaining the VNF IPs from the ESC

By utilizing the confd configuration, from the ESC, the EM and CF VIP can be obtained:

```
[admin@esc-esc-0 ~]$ /opt/cisco/esc/confd/bin/confd_cli -u admin -C
admin connected from 10.131.32.255 using ssh on ultram-vnfml-esc
```

## Element Manager IP Address

```
ultram-vnfm1-esc# show running-config | include nha-gateway

data "vnfm-type:esc \nvnm-ip:172.16.180.11 \nvnm-port:830\nvnm-ping-
frequency:10\nvnm-ping-fail-threshold:30\nvnm-tenant:core \nvnm-
vnfd://opt/cisco/em/config/vnfd.xml\nha-vip:172.16.181.7/24\nha-interface:eth1\nha-
gateway:172.16.181.1"
```

## CF IP Address

```
ultram-vnfm1-esc# show running-config | include -a 1 CF_VIP

variable CF_VIP_ADDR
val [ 172.16.181.9 ]
```

In the example above, the IP address of the CF is 172.16.181.9 and EM IP address is 172.16.181.7.

## Element Manager (EM)

The EM (Element Manager) acts as Element Management System as per ETSI NFV architecture. It directly interacts with VNFs as well as VNFM.

EM is deployed as 3 node cluster VMs on top of OpenStack.

```
[stack@ospd ~]$ openstack server list | grep EM
| 3a8df501-fd3e-4699-bcfb-3ad65d917094 | EM1_0_ceef42c5-9a28-40a7-9c5b-b917903ca9ab |
ACTIVE | UAS-uas-management=172.168.10.9; UAS-uas-orchestration=172.168.11.5
| HIPCF100-DEPLOYMENT-element-manager |
| 4acb5a8a-412d-41fc-a5d2-1c6b53083066 | EM2_0_5c9ea1a3-59dd-4183-bc58-acd815735b87 |
ACTIVE | UAS-uas-management=172.168.10.8; UAS-uas-orchestration=172.168.11.4
| HIPCF100-DEPLOYMENT-element-manager |
| 5213c623-3079-4856-8fd0-b15ddf75baf1 | EM3_0_964623c0-ebbf-4dea-a171-39aa698a8507 |
ACTIVE | UAS-uas-management=172.168.10.7; UAS-uas-orchestration=172.168.11.3
| HIPCF100-DEPLOYMENT-element-manager |
```

ESC is responsible for deploying these VMs. All the EM VMs are part of a single service:

```
[admin@esc-esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli get
esc_datamodel/opdata
Operational Data

<...>

    <state_machine>
      <state>SERVICE_ACTIVE_STATE</state>
      <vm_state_machines>
        <vm_state_machine>
          <vm_name>EM2_0_5c9ea1a3-59dd-4183-bc58-acd815735b87</vm_name>
          <state>VM_ALIVE_STATE</state>
        </vm_state_machine>
        <vm_state_machine>
          <vm_name>EM3_0_964623c0-ebbf-4dea-a171-39aa698a8507</vm_name>
          <state>VM_ALIVE_STATE</state>
        </vm_state_machine>
        <vm_state_machine>
          <vm_name>EM1_0_ceef42c5-9a28-40a7-9c5b-b917903ca9ab</vm_name>
          <state>VM_ALIVE_STATE</state>
        </vm_state_machine>
      </vm_state_machines>
    </state_machine>

<...>
```

Check the NCS high-availability state running in EM Machine:

```
ubuntu@EM0:~$ ncs_cli -u admin -C

admin connected from 127.0.0.1 using ssh on EM0
admin@scm#

admin@scm# show ncs-state ha
ncs-state ha mode master
ncs-state ha node-id 3-1520262513
ncs-state ha connected-slave [ 4-1521723948 ]
admin@scm#
```

Check the EM services. All should be up and running:

```
admin@scm# show ems
EM          VNFM
ID  SLA  SCM  PROXY
-----
3   up   up   up
4   up   up   up
```

**Note** The EM ID 3 & 4 denotes the last octet of the IP address of each EM VM. The EM machine with IP address ending with 5 is the ZooKeeper machine.

To check which machine is ZooKeeper master or follower, run the following command:

```
ubuntu@EM0:~$ ip a s eth0; /opt/cisco/usp/packages/zookeeper/current/bin/zkServer.sh
status
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether fa:16:3e:4b:2d:95 brd ff:ff:ff:ff:ff:ff
    inet 172.168.11.3/24 brd 172.168.11.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe4b:2d95/64 scope link
        valid_lft forever preferred_lft forever
ZooKeeper JMX enabled by default
Using config: /opt/cisco/usp/packages/zookeeper/current/bin/./conf/zoo.cfgMode:
leader

ubuntu@EM1:~$ ip a s eth0; /opt/cisco/usp/packages/zookeeper/current/bin/zkServer.sh
status
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether fa:16:3e:ac:e0:73 brd ff:ff:ff:ff:ff:ff
    inet 172.168.11.4/24 brd 172.168.11.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:feac:e073/64 scope link
        valid_lft forever preferred_lft forever
ZooKeeper JMX enabled by default
Using config: /opt/cisco/usp/packages/zookeeper/current/bin/./conf/zoo.cfgMode:
follower
```

Check the ZooKeeper database to verify the CF and SF records:

```

ubuntu@EM0:~$ /opt/cisco/usp/packages/zookeeper/current/bin/zkCli.sh ls
/oper/vdus/control-function
Connecting to localhost:2181
<..>
WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[BOOT_generic_di-chasis_CF2_1, BOOT_generic_di-chasis_CF1_1]
ubuntu@EM0:~$

ubuntu@EM0:~$ /opt/cisco/usp/packages/zookeeper/current/bin/zkCli.sh ls
/oper/vdus/session-function
Connecting to localhost:2181
<..>
WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[BOOT_generic_di-chasis_SF1_1, BOOT_generic_di-chasis_SF3_1, BOOT_generic_di-
chasis_SF2_1, BOOT_generic_di-chasis_SF5_1, BOOT_generic_di-chasis_SF4_1,
BOOT_generic_di-chasis_SF7_1, BOOT_generic_di-chasis_SF6_1,
BOOT_generic_di-chasis_SF8_1]
ubuntu@EM0:~$

```

Check the NCS configuration database and verify the elements with ZooKeeper operational database:

```

admin@scm# show vdus vdu vnfcI

```

				CONSTITUENT		
MEMORY	STORAGE			DEVICE	ELEMENT	
IS				CPU	UTILS	USAGE
ID	ID		NAME	GROUP	GROUP	
INFRA	INITIALIZED	VIM ID		UTILS	BYTES	BYTES
control-function	BOOT_generic_di-chasis_CF1_1	scm-cf-nc	scm-cf-nc	scm-cf-nc	di-chasis	
true	true	6923fd55-cd2d-40d3-8b61-6b19f99d68c7	-	-	-	
		BOOT_generic_di-chasis_CF2_1	scm-cf-nc	scm-cf-nc	di-chasis	
true	true	cc381fa0-06a2-466c-bd27-d719d142ef81	-	-	-	
session-function	BOOT_generic_di-chasis_SF1_1	-	-	-	di-chasis	
true	false	ab667215-42fc-4543-9854-a3852fcf12c0	-	-	-	
		BOOT_generic_di-chasis_SF2_1	-	-	di-chasis	

```

true   false   2598c63c-2620-4077-a0c6-dca8a422638c - - -
BOOT_generic_di-chasis_SF3_1 - - di-chasis
true   false   71133e5f-b6d3-46d2-b422-cc143b3b8b26 - - -
BOOT_generic_di-chasis_SF4_1 - - di-chasis
true   false   0e055b75-a010-48c9-97b0-08edfc0af5db - - -
BOOT_generic_di-chasis_SF5_1 - - di-chasis
true   false   46186e68-1878-4d35-83b0-007e268bc0d8 - - -
BOOT_generic_di-chasis_SF6_1 - - di-chasis
true   false   01d2953a-f08b-4620-888e-7225f0e6c7c9 - - -
BOOT_generic_di-chasis_SF7_1 - - di-chasis
true   false   a9c6b63e-8833-4994-bd11-aa94c160eca8 - - -
BOOT_generic_di-chasis_SF8_1 - - di-chasis
true   false   86aaeb38-a9ff-4ae7-b873-207678d4fc58 - - -

```

The NCS device below should be enabled:

```

admin@scm# show devices device state
                OPER
                STATE
NAME           OPER  ERROR                LAST TRANSACTION
              STATE  TAG   TRANSACTION MODE    ID
-----
scm-cf-nc     enabled -      lock-reset-candidate 1517-221053-160063
admin@scm#

```

Verify that the NETCONF service is running and port is enabled:

```

admin@scm# show ncs-state netconf
NETCONF SSH listen addresses:
IP           PORT
-----
0.0.0.0     2022

OR

ubuntu@EM0:~$ netstat -an | grep 2022
tcp          0          0 0.0.0.0:2022          0.0.0.0:*            LISTEN

```

Check the logs available at `/var/log/em` directory:

```
ubuntu@EM0:~$ ls -lrt /var/log/em
total 16
drwxr-xr-x 2 zk   zk   4096 Dec 20 10:46 zookeeper
drwxr-xr-x 2 sla  sla  4096 Dec 20 10:49 sla
drwxr-xr-x 2 root root 4096 Mar 22 13:05 vnfm-proxy
drwxr-xr-x 2 root root 4096 Mar 27 13:17 ncs
```

# VPC-DI Upgrade

For VPC software updates on Ultra M, please refer to the corresponding Cisco Ultra Services Platform Deployment Automation Guide of your software release:

<https://www.cisco.com/c/en/us/support/wireless/ultra-automation-services/products-installation-and-configuration-guides-list.html>

For example, the Update of VPC for R6.0 is covered at the following link:

Cisco Ultra Services Platform Deployment Automation Guide, Release 6.0 > Chapter: VNF Upgrade/Redeployment Automation

[https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-6\\_N6-0/USP\\_DAG/N6-0\\_USP\\_Deploy\\_Automation\\_Guide/N5-8\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_0101.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-6_N6-0/USP_DAG/N6-0_USP_Deploy_Automation_Guide/N5-8_USP_Deploy_Automation_Guide_chapter_0101.html)

The guide describes the step-by-step procedure to update VPC software, which includes undeploying the old release and deploying the new software release of Virtualized Packet Core.

For upgrade of components of Ultra M including UCS bare metal OSD Computes and Controller, please refer to the corresponding Ultra M Solutions Guide for your software release.

For example, Upgrade using UCS Utilities within Ultra M manager for R6.0 is covered at the following link:

[https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-6\\_N6-0/Ultra-M-Solutions/N6-0-Ultra-M-Solution-Guide/N5-8-Ultra-M-Solution-Guide\\_appendix\\_01101.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-6_N6-0/Ultra-M-Solutions/N6-0-Ultra-M-Solution-Guide/N5-8-Ultra-M-Solution-Guide_appendix_01101.html)

For update and upgrade procedures of Cisco VIM, please refer to the corresponding Cisco Virtual Infrastructure Manager Installation Guide. Cisco Virtual Infrastructure Manager Installation Guide, 2.2.20:

[https://www.cisco.com/c/en/us/td/docs/net\\_mgmt/network\\_function\\_virtualization](https://www.cisco.com/c/en/us/td/docs/net_mgmt/network_function_virtualization)

# Troubleshooting Use Cases

# PXE Boot Failure

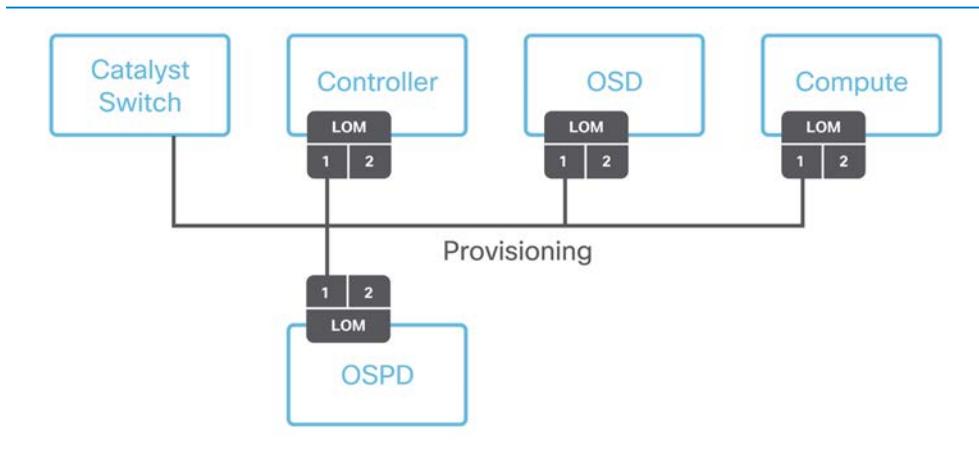
## Introduction

This chapter covers Red Hat OpenStack Platform Director (OSP-D) Based Deployments. Within the Ultra M solution, OpenStack Platform Director (OSP-D) functions as the virtual infrastructure manager (VIM). The method by which the VIM is deployed depends on the Ultra M model architecture. There are two major architectures:

- VIM for Hyper-Converged Ultra M Models
- VIM for Non-Hyper-Converged Ultra M Models

OSP-D acts as the DHCP server and will be assigning the IP addresses to the compute/controller/OSD/Ceph nodes via a provisioning network interface. The interface will be used for PXE boot (during the introspection phase).

Simplified graphical representation of this process is shown below:



This chapter covers troubleshooting steps for PXE boot failure scenario.

## Problem Statement

The symptom seen in a case where the PXE boot fails to complete will be that the UCS server has failed to boot. To identify this, login to the UCS KVM console via CIMC.

```
Intel(R) Boot Agent GE v1.5.53
Copyright (C) 1997-2014, Intel Corporation

Client MAC ADDR: XX:XX:XX:XX:XX:XX GUID:89BE196B 9874 4B74 8352 68634AC552F1
PXE-E51: No DHCP or proxyDHCP offers were received.

PXE-MOF: Exiting Intel Boot Agent.

Intel(R) Boot Agent GE v1.5.53
Copyright (C) 1997 - 2014, Intel Corporation

PXE-E61: Media test failure, check cable
PXE-MOF: Exiting Intel Boot Agent.

Reboot and Select proper Boot device
or Insert Boot Media in selected Boot device and press a key
```

## Recovery Steps

If the UCS is stuck in booting while waiting for the DHCP offer from the DHCP server, take the following troubleshooting steps:

### Step 1:

Identify the Catalyst port to which the UCS is connected and verify the port is up.

```
show int gig x/x/x

GigabitEthernet1/0/20 is up, line protocol is up (connected)
Hardware is Gigabit Ethernet, address is xxxx.xxxx.xxxx.xxxx
Description: "Connected to LAN1 port on COMPUTE_1"
```

```
MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,  
reliability 255/255, txload 1/255, rxload 1/255
```

## Step 2:

Identify the MAC address table of the affected UCS server. Remember the provisioning interface is the onboard Intel port 1 interface. The MAC address can be obtained via CIMC if not otherwise known.

2.a. Execute the **show mac address-table dynamic** in Catalyst and make sure to find the MAC address of the server being troubleshoot is present in the MAC address table.

In case that the MAC address is not present, as a first recovery step, flash the arp table by executing **clear mac address-table**.

2.b. Verify that the UCS is connected to the expected interface and correct VLAN.

2.c. Lastly, make sure the interface is configured with the portfast.

An example of the correct configuration is provided below:

```
interface GigabitEthernet1/0/25  
description "Connected to Mgmt  
switchport access vlan 105  
spanning-tree portfast
```

## Step 3:

While the UCS server is trying to boot (that is, while the OpenStack Introspection process is running), verify that packet count increases in inbound/outbound direction of corresponding port on the Catalyst switch.

```
show int gig x/x/x | grep packet
```

If packets are increasing, the connectivity part has been verified. Next step is to check the DHCP server on OSP-D.

**Step 4:**

Check further in the `/var/log/messages` if there are any failures

For example, if so, something like the string below will be displayed:

```
ultram-ospd dnsmasq-dhcp[5632]: DHCPDISCOVER(tap1234567-11) 48:01:3a:8b:ca:13 no
address available
```

This means that OSP-D failed to allocate the IP to the UCS and it will not complete the boot process.

Confirm that DHCP process is fully operational and running and there are available IPs. The process running DHCP service is called **openstack-ironic-inspector-dnsmasq.service**. To verify the status of the service:

```
[stack@ospd-ultram-1 ~]$ systemctl list-units | grep dns
openstack-ironic-inspector-dnsmasq.service loaded active running PXE
boot
dnsmasq service for Ironic Inspector
```

Check the IP addresses assigned to the specific MACs:

```
[stack@ospd-ultram ~]$ sudo systemctl status openstack-ironic-inspector-
dnsmasq.service
[stack@ospd-ultram-~]$ sudo journalctl -u openstack-ironic-inspector-dnsmasq
```

Lastly, the configuration file that holds the DHCP range configuration varies among releases:

- 5.1 based deployments - `undercloud.conf` configuration file

```
dhcp_start = 192.200.0.10
dhcp_end = 192.200.0.200
```

- 5.5 and above based deployments - vim-orch configuration file

```
provisioning-network dhcp-ip-range start      192.200.0.10  
provisioning-network dhcp-ip-range end       192.200.0.200
```

# CEPH Placement Groups Inconsistency

## Problem Statement

CEPH is not healthy and the whole of the CEPH cluster is down due to CEPH Placement Groups (PGs) - PG scrub errors.

First troubleshooting step is to check the Ceph status by using **ceph -s** command. This command is to be executed on the OSD/Ceph node.

```
[root@osd-compute-0 ~]# ceph -s
cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_ERR .
3 pgs inconsistent
12 scrub errors
monmap e2: 3 mons at {dnucs002-controller-0=11.118.0.10:6789/0,dnucs002-
controller-1=11.118.0.11:6789/0,dnucs002-controller-2=11.118.0.12:6789/0}
election epoch 38, quorum 0,1,2 dnucs002-controller-0,dnucs002-controller-
1,dnucs002-controller-2
osdmap e136: 12 osds: 12 up, 12 in
flags sortbitwise
pgmap v139713: 704 pgs, 6 pools, 525 GB data, 188 kobjects
1573 GB used, 11819 GB / 13393 GB avail
701 active+clean
3 active+clean+inconsistent
client io 98047 B/s wr, 0 op/s rd, 13 op/s wr
[root@dnucs002-osd-compute-0 ~]#
```

As demonstrated in the sample above, we can see the scrub errors and the placement group inconsistencies are very high.

The next step is to isolate the specific OSD that is causing the issue. In this process, the first step is to identify the placement group that has inconsistency and also figure out the mapping OSD using **ceph health detail**.

## Find the Placement groups

A simple command can show use of the PG:

```
[root@osd-compute-0 ~]# ceph health detail
HEALTH_ERR 3 pgs inconsistent; 12 scrub errors
pg 4.5d is active+clean+inconsistent, acting [5,4,3]
pg 4.21 is active+clean+inconsistent, acting [8,3,7]
pg 4.3f is active+clean+inconsistent, acting [2,7,3]
12 scrub errors
```

From the above results, 3 PGs are inconsistent, specifically 4.5d, 4.21 and 4.3f, and his is acting on OSD [ 5,4,3], [8,3,7] & [ 2,7,3]

The first step in recovery is to do the deep scrub on the affected PGs and run **ceph pg repair <PG\_ID>**.

Since scrub errors and placement group inconsistencies are identified, the next step is to set noscrub and nodeep-scrub while working on fixing the issue. This is done in order to keep the scrub queue empty.

## Scrub check

Normally the disk state can be software error or hardware failure where the disk replacement is required. Command **ceph pg <PG\_ID> query** may be used for that

If the placement group is queried, information that helps narrow down the source of the issue can be obtained. Specifically, timelines provide an indication of when the issue started. In the sample presented below, that latest scrub ran at 20:07, however, the last clean scrub came the day before at 9:36. This information can help isolate the start time and find the specific event that happened around that time that triggered the PG inconsistency.

```

> ceph pg 1.4 query

{
  "state": "active+clean+inconsistent",
  ....
  "last_scrub_stamp": "2018-02-17 20:07:52.487123",
  "last_deep_scrub": "112'144",
  "last_deep_scrub_stamp": "2018-02-17 20:07:52.487123",
  "last_clean_scrub_stamp": "2018-02-16 09:36:54.918422"
},
  "stats": {
    ...
    "num_scrub_errors": 17,
    "num_shallow_scrub_errors": 0,
    "num_deep_scrub_errors": 17,
    "num_objects_recovered": 52,
    "num_bytes_recovered": 419692544,
    ...
  }
}

```

## Placement Group Repair

To recover the inconsistent Placement Group, run **ceph pg repair <PG\_ID>**.

To monitor progress of the repair, check messages in `/var/log/ceph/ceph*.log`.

```

$ ceph pg repair 1.4
instructing pg 1.4 on osd.3 to repair

tail -f /var/log/ceph/ceph*.log

2018-02-18 13:00:31.454374 mon.0 6.6.6.8:6789/0 77373 : cluster [INF] pgmap v80542:
896 pgs: 2 active+clean+inconsistent, 894 active+clean; 74902 MB data, 220 GB used,
16538 GB / 16759 GB avail
2018-02-18 13:01:38.042559 osd.5 6.6.6.7:6812/2675 1255 : cluster [INF] 2.11e scrub
starts
2018-02-18 13:01:38.043660 osd.5 6.6.6.7:6812/2675 1256 : cluster [INF] 2.11e
scrub ok

```

Sometimes manual repair, as demonstrated above, does not help, and in that case, further checks are needed.

## Disable Scrub

Disabling scrub comes in handy also in the case when OSD is marked down because it did not reply in time due to being busy scrubbing a placement group.

```
ceph osd set noscrub
ceph osd set nodeep-scrub
```

## Manually running deep scrub

To run the deep scrub manually, run the **ceph pg deep-scrub <PG\_ID>**:

```
ceph pg deep-scrub 4.5d
ceph pg deep-scrub 4.21
ceph pg deep-scrub 4.3f
```

## Identify the impacted OSD

If manual repair and deep scrub did not fix the issue, isolating the OSD causing the issue is required.

Every Placement Group is associated with a primary OSD that replicates the data to the two standby OSDs. In the sample used here, show OSDs with the **ceph health detail** command:

```
[root@osd-compute-0 ~]# ceph health detail
HEALTH_ERR 3 pgs inconsistent; 12 scrub errors
pg 4.5d is active+clean+inconsistent, acting [5,4,3]
```

```
pg 4.21 is active+clean+inconsistent, acting [8,3,7]
pg 4.3f is active+clean+inconsistent, acting [2,7,3]
12 scrub errors
```

The above shows, for example, placement group 4.5d has been associated with OSD 5, 4 and 3.

Further inspection shows that OSD-3 is common to all 3 affected PGs.

These placement groups are active+clean and were able to serve the read/write requests (otherwise their status would be **inactive**). The clean status indicates that placement group has been able to recover from a previous failure (otherwise status would be **unclean**). Finally, the status is not stale, which means that placement group has been updated by OSD - indicating that at least one OSD is up.

## Inspecting Stale Inactive and Unclean Placement Groups

When the Ceph health identifies the PG is not active+clean, the following commands may be used to identify stale inactive and unclean PGs for each OSD node.

```
ceph pg dump_stuck stale
ceph pg dump_stuck inactive
ceph pg dump_stuck unclean
```

## Inspecting the Inconsistencies

Ceph provides a utility to inspect the inconsistencies and it is run by using **rados list-inconsistent-obj** command.

Specifically, this command will locate errors - such as in the sample below where there is a `read_error` for the suspect OSD-3:

```
[root@osd-compute-0 ~]# rados list-inconsistent-obj 4.5d --format=json-pretty
{
  "epoch": 84,
  "inconsistents": [
    {
      "object": {
        "name": "rbd_data.29de96ebafa01.00000000000008cd8",
        "namespace": "",
        "locator": "",
        "snap": "head"
      },
      "errors": [
        "read_error"
      ],
      "shards": [
        {
          "osd": 3,
          "size": 4194304,
          "errors": [
            "read_error"
          ]
        },
        {
          "osd": 4,
          "size": 4194304,
          "omap_digest": "0xffffffff",
          "data_digest": "0x43d61c5d",
          "errors": []
        },
        {
          "osd": 5,
          "size": 4194304,
          "omap_digest": "0xffffffff",
          "data_digest": "0x43d61c5d",
          "errors": []
        }
      ]
    },
    {
      "osd": 3,
      "size": 4194304,
      "omap_digest": "0xffffffff",
      "data_digest": "0x43d61c5d",
      "errors": []
    }
  ]
}
```

## Find the Disk mapping to OSD

The **ceph-disk list** command shows a device that is mapped to OSD-3.

```
[root@osd-compute-1 heat-admin]# ceph-disk list
/dev/sda :
/dev/sda1 other, iso9660
/dev/sda2 other, xfs, mounted on /
/dev/sdb :
/dev/sdb1 ceph journal, for /dev/sdc1
/dev/sdb3 ceph journal, for /dev/sdd1
/dev/sdb2 ceph journal, for /dev/sde1
/dev/sdb4 ceph journal, for /dev/sdf1
/dev/sdc :
/dev/sdc1 ceph data, active, cluster ceph, osd.0, journal /dev/sdb1
/dev/sdd :
/dev/sdd1 ceph data, active, cluster ceph, osd.6, journal /dev/sdb3
/dev/sde :
/dev/sde1 ceph data, prepared, cluster ceph, osd.3, journal /dev/sdb2
/dev/sdf :
/dev/sdf1 ceph data, active, cluster ceph, osd.9, journal /dev/sdb4
```

From the above output, OSD-3 has been mapped to the `/dev/sde1`

## Further troubleshooting utilizing SOS reports

After narrowing down the PG inconsistencies to the OSD-3 `/dev/sde` device, next check if there are any reports of the hardware failures. Normally, that can be done from the Operating System side. Red Hat provides SOS reports that can be used to obtain the data for troubleshooting.

```
grep sde sos_commands/logs/journalctl_--no-pager_--boot >>

Jun 14 12:25:18 osd-compute-1.localdomain kernel: sd 0:2:4:0: [sde] tag#2 FAILED
Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
Jun 14 12:25:18 osd-compute-1.localdomain kernel: sd 0:2:4:0: [sde] tag#2 Sense Key :
Medium Error [current]
Jun 14 12:25:18 osd-compute-1.localdomain kernel: sd 0:2:4:0: [sde] tag#2 Add. Sense:
Unrecovered read error
Jun 14 12:25:18 osd-compute-1.localdomain kernel: sd 0:2:4:0: [sde] tag#2 CDB:
```

```
Read(10) 28 00 2c 28 8b 48 00 02 00 00
Jun 14 12:25:18 osd-compute-1.localdomain kernel: blk_update_request: critical medium
error, dev sde, sector 740854600
```

From the output, it appears there is a medium error on a specific sector, hence this disk shall be replaced.

## Replacing the faulty disk

Before removing the bad disk, please make sure there is enough space in the cluster to backfill this OSD data in other OSDs in the cluster.

### Step 1:

Stop the OSD

```
# systemctl stop ceph-osd@3
```

### Step 2:

Mark OSD out

```
# ceph osd out osd.3
```

### Step 3:

Wait for backfill/recovery(data rebalance) to finish. Once rebalance is completed, the bad disk can be removed from the cluster and replaced.

**Step 4:**

Remove the OSD from the crush map

```
# ceph osd crush remove osd.3
```

**Step 5:**

Remove the authentication key for the OSD.

```
# ceph auth del osd.3
```

**Step 6:**

Unmount the mount point:

```
# umount /var/lib/ceph/osd/ceph-3
```

**Step 7:**

Wait for rebalance to finish and once done, unset the noscrub and nodeep-scrub flags.

```
# ceph osd unset noscrub  
# ceph osd unset nodeep-scrub
```

# OpenStack Cluster Recovery

Cluster stack is a high-availability and load-balancing stack which is used to make OpenStack infrastructure highly available.

Cluster health can be degraded for multiple reasons:

- power outage causing multiple controllers (participants in the cluster) to go down simultaneously.
- database corruption within OpenStack affecting the cluster participants.
- memory issues on controllers.
- connectivity issues between controllers.

This section covers recover steps for Ultra M with OSP-D and Cisco VIM deployments.

## OpenStack cluster recovery with OSP Director

### Step 1:

Check network connectivity.

All IPs for all resources (haproxy-clone, galera-master, rabbitmq-clone, redis-master) should be reachable via all controllers participating in the pcs cluster.

```
[root@ultram-controller-1 heat-admin]# pcs status
Cluster name: tripleo_cluster
Stack: corosync
Current DC: ultram-controller-0 (version 1.1.15-11.e17_3.4-e174ec8) - partition with quorum
Last updated: Wed Mar 28 14:20:22 2018          Last change: Tue Mar 27 15:00:31 2018
by root via crm_resource on ultram-controller-0

3 nodes and 19 resources configured
```

```

Online: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]

Full list of resources:

ip-11.18.0.107 (ocf::heartbeat:IPaddr2):      Started ultram-controller-0
ip-11.20.0.106 (ocf::heartbeat:IPaddr2):      Started ultram-controller-2
Clone Set: haproxy-clone [haproxy]
  Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: galera-master [galera]
  Masters: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
ip-11.19.0.105 (ocf::heartbeat:IPaddr2):      Started ultram-controller-0
ip-11.20.0.108 (ocf::heartbeat:IPaddr2):      Started ultram-controller-2
Clone Set: rabbitmq-clone [rabbitmq]
  Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: redis-master [redis]
  Masters: [ ultram-controller-1 ]
  Slaves: [ ultram-controller-0 ultram-controller-2 ]
ip-192.200.0.111 (ocf::heartbeat:IPaddr2):    Started ultram-rcdnlab-
controller-0
ip-10.201.206.23 (ocf::heartbeat:IPaddr2):    Started ultram-rcdnlab-
controller-2
openstack-cinder-volume (systemd:openstack-cinder-volume): Started
ultram-controller-0

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
[root@ultram-controller-1 heat-admin]#
[root@ultram-controller-1 heat-admin]# ping 11.20.0.108 -c 3    >> Checked Galera-
master connectivity on controller 2
PING 11.20.0.108 (11.20.0.108) 56(84) bytes of data.
64 bytes from 11.20.0.108: icmp_seq=1 ttl=64 time=0.082 ms
64 bytes from 11.20.0.108: icmp_seq=2 ttl=64 time=0.095 ms
64 bytes from 11.20.0.108: icmp_seq=3 ttl=64 time=0.112 ms

--- 11.20.0.108 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.082/0.096/0.112/0.014 ms

```

## Step 2:

### PCS Cluster Resource Identification

In the event that any pcs resource (highlighted in red) is in Stopped State or not part of Cluster "Started" output, then perform the following

Identify the resource that has failed with pcs status CLI:

```
[root@ultram-controller-1 heat-admin]# pcs status
Cluster name: tripleo_cluster
Stack: corosync
Current DC: ultram-controller-0 (version 1.1.15-11.e17_3.4-e174ec8) - partition with
quorum
Last updated: Wed Mar 28 14:20:22 2018          Last change: Tue Mar 27 15:00:31 2018
by root via crm_resource on ultram-controller-0

3 nodes and 19 resources configured

Online: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]

Full list of resources:

ip-11.18.0.107 (ocf::heartbeat:IPaddr2):      Started ultram-rcdnlab-controller-0
ip-11.20.0.106 (ocf::heartbeat:IPaddr2):      Started ultram-rcdnlab-controller-2
Clone Set: haproxy-clone [haproxy]
  Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: galera-master [galera]
  Masters: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
ip-11.19.0.105 (ocf::heartbeat:IPaddr2):      Started ultram-controller-0
ip-11.20.0.108 (ocf::heartbeat:IPaddr2):      Started ultram-controller-2
Clone Set: rabbitmq-clone [rabbitmq]
  Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: redis-master [redis]
  Masters: [ ultram-controller-1 ]
  Slaves: [ ultram-controller-0 ultram-controller-2 ]
ip-192.200.0.111 (ocf::heartbeat:IPaddr2):      Started ultram-controller-0
ip-10.201.206.23 (ocf::heartbeat:IPaddr2):      Started ultram-controller-2
openstack-cinder-volume (systemd:openstack-cinder-volume): Started
ultram-controller-0

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

### Step 3:

#### Recovery

For recovering resources on a specific controller, run the following:

**Note** The steps below are applicable only to all non-Galera resources.

In this example, non-Galera resources are : ha-proxy, ha-proxy-clone, rabbitmq, rabbitmq-clone and redis-master.

This example shows the pcs resource restart for RabbitMQ resource on Controller-0

```
Syntax : pcs resource restart <resource_name> <controller_name>

           pcs resource cleanup <resource_name> <controller_name>

[root@ultram-controller-1 heat-admin]# pcs resource restart rabbitmq ultram-
controller-0
Warning: using rabbitmq-clone... (if a resource is a clone or master/slave you must
use the clone or master/slave name
```

If the resource has to be restarted on all controllers, then use the **-clone** resource in the pcs restart CLI.

```
[root@ultram-controller-1 heat-admin]# pcs resource restart rabbitmq-clone
Warning: using rabbitmq-clone... (if a resource is a clone or master/slave you must
use the clone or master/slave name
```

### Step 4:

#### PCS Resource Cleanup

Cleanup old failed actions in PCS resource with the following:

```
Syntax : pcs resource cleanup <resource_name> <controller_name>

[root@ultram-controller-1 heat-admin]# pcs resource cleanup rabbitmq
Cleaning up rabbitmq:0 on ultram-controller-0, removing fail-count-rabbitmq
```

```
Cleaning up rabbitmq:0 on ultram--controller-1, removing fail-count-rabbitmq
Cleaning up rabbitmq:0 on ultram-controller-2, removing fail-count-rabbitmq
Waiting for 3 replies from the CRMD... OK
```

### Step 5:

Verify the PCS status

After PCS resource clean up, run **pcs status** CLI to confirm if all the resources have started on all controllers in the cluster.

PCS Cluster health can be checked using the following command and below is the output from a healthy cluster.

```
[root@ultram-controller-1 heat-admin]# pcs status
Cluster name: tripleo_cluster
Stack: corosync
Current DC: ultram-controller-0 (version 1.1.15-11.e17_3.4-e174ec8) - partition with
quorum
Last updated: Wed Mar 28 14:20:22 2018          Last change: Tue Mar 27 15:00:31 2018
by root via crm_resource on ultram-controller-0<

3 nodes and 19 resources configured<

Online: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]

Full list of resources:

ip-11.18.0.107 (ocf::heartbeat:IPaddr2):          Started ultram-controller-0
ip-11.20.0.106 (ocf::heartbeat:IPaddr2):          Started ultram-controller-2
Clone Set: haproxy-clone [haproxy]
  Started: [ ultram-rcdnlab-controller-0 ultram-rcdnlab-controller-1 ultram-
controller-2 ]
Master/Slave Set: galera-master [galera]
  Masters: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
ip-11.19.0.105 (ocf::heartbeat:IPaddr2):          Started ultram-controller-0
ip-11.20.0.108 (ocf::heartbeat:IPaddr2):          Started ultram-controller-2
Clone Set: rabbitmq-clone [rabbitmq]
  Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: redis-master [redis]
  Masters: [ ultram-rcdnlab-controller-1 ]
  Slaves: [ ultram-rcdnlab-controller-0 ultram-rcdnlab-controller-2 ]
ip-192.200.0.111 (ocf::heartbeat:IPaddr2):          Started ultram-controller-0
```

```
ip-10.201.206.23      (ocf::heartbeat:IPAddr2):      Started ultram-controller-2
openstack-cinder-volume (systemd:openstack-cinder-volume): Started
ultram-controller-0

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

GALERA-MASTER resource cannot be directly restarted and some additional checks need to be done and verified. It is recommended to contact Cisco for the appropriate steps and recommendations.

## OpenStack cluster recovery with Cisco VIM

With Cisco VIM, OpenStack cluster recovery has been simplified and included as a single CLI option in the `ciscovim` CLI command suite. To conduct Cluster recovery, the command is **`ciscovim cluster-recovery`**. The following is example output from a successful run.

```
[root@mgmtnode-1]# ciscovim -h
usage: ciscovim [--setupfile <setupdata_file>] <subcommand> ...

Command-line interface to the Cisco Virtualized manager

Positional arguments:

  <subcommand>

  ----- snip -----

  cluster-recovery          Recover the Openstack cluster after a network
                             partition or power outage

Optional arguments:

  --setupfile <setupdata_file>
```

```
[root@mgmtnode-1]# ciscovim cluster-recovery

Perform the action. Continue (Y/N)y

Monitoring Cluster-recovery Operation

. .

Cisco VIM Runner logs

The logs for this run are available in 10.201.242.202:/var/log/mercury/fb358e3a-
1d14-4a59-983c-58492275b358

#####

Cisco Virtualized Infrastructure Manager

#####

[1/1][ORCHESTRATION: Generating Inventory]
[ / ] 0min 0sec

[1/1][ORCHESTRATION: Generating Inventory]
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-check rabbitmq nodes status via r...
[ - ] 0min 22secs

[1/1][ORCHESTRATION: rabbitmq_recovery-check rabbitmq nodes status via r...
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-parse rabbitmq nodes status output]
[ | ] 0min 22secs

[1/1][ORCHESTRATION: rabbitmq_recovery-parse rabbitmq nodes status output]
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-check if any rabbitmq node thinks...
[ - ] 0min 22secs

[1/1][ORCHESTRATION: rabbitmq_recovery-check if any rabbitmq node thinks...
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | stop docker-ra...
[ | ] 0min 22secs
```

```

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | stop docker-ra...
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | stop docker-ra...
[ - ] 0min 22secs

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | stop docker-ra...
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-wait for services to stop]
[ | ] 0min 22secs

[1/1][ORCHESTRATION: rabbitmq_recovery-wait for services to stop]
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | remove mnesia ...
[ - ] 0min 22secs

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | remove mnesia ...
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | start docker-r...
[ | ] 0min 22secs

[1/1][ORCHESTRATION: rabbitmq_recovery-rabbitmq restart | start docker-r...
[ DONE! ]

[1/1][ORCHESTRATION: rabbitmq_recovery-wait for sometime before checking...
[ - ] 0min 23secs

[1/1][ORCHESTRATION: rabbitmq_recovery-wait for sometime before checking...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-get wsrep cluster status]
[ \ ] 0min 27secs

[1/1][ORCHESTRATION: galera_recovery-get wsrep cluster status]
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-get wsrep ready]
[ | ] 0min 27secs

[1/1][ORCHESTRATION: galera_recovery-get wsrep ready]
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-get wsrep cluster size]

```

```
[ / ] 0min 27secs

[1/1][ORCHESTRATION: galera_recovery-get wsrep cluster size]
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-create wsrep cluster status list, w...
[ - ] 0min 27secs

[1/1][ORCHESTRATION: galera_recovery-create wsrep cluster status list, w...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-determine if partial failure recove...
[ / ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-determine if partial failure recove...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-get current mariadb container]
[ \ ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-get current mariadb container]
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | get wsre...
[ | ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | get wsre...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | get wsre...
[ - ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | get wsre...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | create b...
[ | ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | create b...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | check if...
[ - ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | check if...
[ DONE! ]
```

```
[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | set pc.b...
[ | ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | set pc.b...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | copy tem...
[ - ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | copy tem...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | reset th...
[ | ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-partial failure recovery | reset th...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-pause for sometime after partial fa...
[ - ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-pause for sometime after partial fa...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-determine if partial failure recove...
[ | ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-determine if partial failure recove...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | stop do...
[ - ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | stop do...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | copy te...
[ | ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | copy te...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | start t...
[ - ] 0min 28secs
```

```

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | start t...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | wait fo...
[ | ] 0min 28secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | wait fo...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | get seq...
[ - ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | get seq...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | create ...
[ | ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | create ...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | copy te...
[ - ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | copy te...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | start d...
[ | ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | start d...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | mariadb...
[ - ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | mariadb...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | reset t...
[ | ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-complete failure recovery | reset t...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-pause for sometime after complete f...

```

```

[ - ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-pause for sometime after complete f...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | check cluster s...
[ | ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | check cluster s...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | check cluster s...
[ / ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | check cluster s...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | get cluster sta...
[ - ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | get cluster sta...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | get cluster con...
[ \ ] 0min 29secs

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | get cluster con...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | create cluster ...
[ \ ] 0min 30secs

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | create cluster ...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | check cluster s...
[ - ] 0min 30secs

[1/1][ORCHESTRATION: galera_recovery-cluster integrity | check cluster s...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-node status | check for wsrep_ready...
[ | ] 0min 30secs

[1/1][ORCHESTRATION: galera_recovery-node status | check for wsrep_ready...
[ DONE! ]

```

```
[1/1][ORCHESTRATION: galera_recovery-node status | check for wsrep_conne...
[ / ] 0min 30secs

[1/1][ORCHESTRATION: galera_recovery-node status | check for wsrep_conne...
[ DONE! ]

[1/1][ORCHESTRATION: galera_recovery-node status | check for wsrep_local...
[ - ] 0min 30secs

[1/1][ORCHESTRATION: galera_recovery-node status | check for wsrep_local...
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ \ ] 0min 43secs

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Pause for sometime before checking again]
[ / ] 0min 43secs

[1/1][ORCHESTRATION: nova-check-Pause for sometime before checking again]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Create expected nova-conductor list]
[ \ ] 0min 43secs

[1/1][ORCHESTRATION: nova-check-Create expected nova-conductor list]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ / ] 0min 43secs

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ | ] 0min 45secs

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...
[ | ] 0min 45secs

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...
```

```

[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Create expected nova-scheduler list]
[ - ] 0min 45secs

[1/1][ORCHESTRATION: nova-check-Create expected nova-scheduler list]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ \ ] 0min 45secs

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...]
[ - ] 0min 46secs

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...]
[ / ] 0min 47secs

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Create expected nova-consoleauth list]
[ \ ] 0min 47secs

[1/1][ORCHESTRATION: nova-check-Create expected nova-consoleauth list]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ / ] 0min 47secs

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...]
[ \ ] 0min 48secs

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Pause for sometime before checking again]
[ - ] 0min 48secs

```

```
[1/1][ORCHESTRATION: nova-check-Pause for sometime before checking again]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...
[ | ] 0min 48secs

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Create expected nova-computes list]
[ - ] 0min 48secs

[1/1][ORCHESTRATION: nova-check-Create expected nova-computes list]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ - ] 0min 48secs

[1/1][ORCHESTRATION: nova-check-Check nova service list for down state]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ | ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Pause for sometime before checking again]
[ \ ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Pause for sometime before checking again]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Create expected nova-conductor list]
[ / ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Create expected nova-conductor list]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ \ ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Create expected nova-scheduler list]
```

```
[ / ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Create expected nova-scheduler list]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ \ ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Create expected nova-consoleauth list]
[ / ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Create expected nova-consoleauth list]
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ \ ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Restart specific nova service on down or...
[ DONE! ]

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...
[ / ] 0min 50secs

[1/1][ORCHESTRATION: nova-check-Check nova service list again for down s...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy disk.check.py to /tmp]
[ - ] 0min 50secs

[1/1][ORCHESTRATION: cloud-sanity-Copy disk.check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Run Disk maintenance for results data]
[ | ] 0min 51secs

[1/1][ORCHESTRATION: cloud-sanity-Run Disk maintenance for results data]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Management - Disk maintenance RAID Hea...
[ | ] 1min 39secs

[1/1][ORCHESTRATION: cloud-sanity-Management - Disk maintenance RAID Hea...
[ DONE! ]
```

```

[1/1][ORCHESTRATION: cloud-sanity-Management - Disk maintenance VD Health]
[ / ] 1min 39secs

[1/1][ORCHESTRATION: cloud-sanity-Management - Disk maintenance VD Health]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove disk.check.py from /tmp]
[ / ] 1min 40secs

[1/1][ORCHESTRATION: cloud-sanity-Remove disk.check.py from /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove diskmgmt check disks results js...]
[ - ] 1min 40secs

[1/1][ORCHESTRATION: cloud-sanity-Remove diskmgmt check disks results js...]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ - ] 1min 40secs

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Run Docker API Check for actual containe...]
[ \ ] 1min 40secs

[1/1][ORCHESTRATION: cloud-sanity-Run Docker API Check for actual containe...]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Save actual docker tag results JSON ou...]
[ / ] 1min 40secs

[1/1][ORCHESTRATION: cloud-sanity-Save actual docker tag results JSON ou...]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Test container names/version]
[ \ ] 1min 40secs

[1/1][ORCHESTRATION: cloud-sanity-Test container names/version]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Management - Container version check]
[ / ] 1min 43secs

```

```
[1/1][ORCHESTRATION: cloud-sanity-Management - Container version check]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Control - Ping All Controller Nodes]
[ - ] 1min 43secs

[1/1][ORCHESTRATION: cloud-sanity-Control - Ping All Controller Nodes]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Control - Ping internal VIP]
[ - ] 1min 46secs

[1/1][ORCHESTRATION: cloud-sanity-Control - Ping internal VIP]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Control - Check Mariadb cluster size]
[ - ] 1min 48secs

[1/1][ORCHESTRATION: cloud-sanity-Control - Check Mariadb cluster size]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Control - Check RabbitMQ is running]
[ \ ] 1min 48secs

[1/1][ORCHESTRATION: cloud-sanity-Control - Check RabbitMQ is running]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Control - Check RabbitMQ cluster status]
[ / ] 1min 48secs

[1/1][ORCHESTRATION: cloud-sanity-Control - Check RabbitMQ cluster status]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy Nova Check File to /tmp/]
[ / ] 1min 52secs

[1/1][ORCHESTRATION: cloud-sanity-Copy Nova Check File to /tmp/]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Execute Nova Service List]
[ \ ] 1min 52secs

[1/1][ORCHESTRATION: cloud-sanity-Execute Nova Service List]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove Nova Service Check file from /t...
```

```
[ \ ] 1min 53secs

[1/1][ORCHESTRATION: cloud-sanity-Remove Nova Service Check file from /t...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy disk.check.py to /tmp]
[ \ ] 1min 53secs

[1/1][ORCHESTRATION: cloud-sanity-Copy disk.check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Run Disk maintenance for results data]
[ / ] 1min 54secs

[1/1][ORCHESTRATION: cloud-sanity-Run Disk maintenance for results data]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Control - Disk maintenance VD Health]
[ / ] 1min 54secs

[1/1][ORCHESTRATION: cloud-sanity-Control - Disk maintenance VD Health]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove disk.check.py from /tmp]
[ \ ] 1min 54secs

[1/1][ORCHESTRATION: cloud-sanity-Remove disk.check.py from /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove diskmgmt check disks results js...
[ - ] 1min 54secs

[1/1][ORCHESTRATION: cloud-sanity-Remove diskmgmt check disks results js...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ - ] 1min 54secs

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ \ ] 1min 55secs

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ DONE! ]
```

```

[1/1][ORCHESTRATION: cloud-sanity-Run Docker API Check for actual contai...
[ \ ]    1min 55secs

[1/1][ORCHESTRATION: cloud-sanity-Run Docker API Check for actual contai...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Save actual docker tag results JSON ou...
[ \ ]    1min 55secs

[1/1][ORCHESTRATION: cloud-sanity-Save actual docker tag results JSON ou...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Test container names/version]
[ / ]    1min 55secs

[1/1][ORCHESTRATION: cloud-sanity-Test container names/version]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Control - Container version check]
[ / ]    2mins 4secs

[1/1][ORCHESTRATION: cloud-sanity-Control - Container version check]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy Nova Check File to /tmp/]
[ / ]    2mins 6secs

[1/1][ORCHESTRATION: cloud-sanity-Copy Nova Check File to /tmp/]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Compute - Ping All Compute Nodes]
[ \ ]    2mins 7secs

[1/1][ORCHESTRATION: cloud-sanity-Compute - Ping All Compute Nodes]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Execute Nova Hypervisor Check]
[ / ]    2mins 7secs

[1/1][ORCHESTRATION: cloud-sanity-Execute Nova Hypervisor Check]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Compute - Check Nova Hypervisor list]
[ \ ]    2mins 10secs

[1/1][ORCHESTRATION: cloud-sanity-Compute - Check Nova Hypervisor list]

```

```
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove Nova Service Check file from /t...
[ / ] 2mins 10secs

[1/1][ORCHESTRATION: cloud-sanity-Remove Nova Service Check file from /t...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy disk.check.py to /tmp]
[ / ] 2mins 10secs

[1/1][ORCHESTRATION: cloud-sanity-Copy disk.check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Run Disk maintenance for results data]
[ - ] 2mins 10secs

[1/1][ORCHESTRATION: cloud-sanity-Run Disk maintenance for results data]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Compute - Disk maintenance RAID Health]
[ - ] 2mins 11secs

[1/1][ORCHESTRATION: cloud-sanity-Compute - Disk maintenance RAID Health]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove disk.check.py from /tmp]
[ | ] 2mins 11secs

[1/1][ORCHESTRATION: cloud-sanity-Remove disk.check.py from /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove diskmgmt check disks results js...
[ | ] 2mins 11secs

[1/1][ORCHESTRATION: cloud-sanity-Remove diskmgmt check disks results js...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ \ ] 2mins 11secs

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ | ] 2mins 11secs
```

```
[1/1][ORCHESTRATION: cloud-sanity-Copy docker api check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Run Docker API Check for actual contai...
[ | ] 2mins 11secs

[1/1][ORCHESTRATION: cloud-sanity-Run Docker API Check for actual contai...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Save actual docker tag results JSON ou...
[ | ] 2mins 12secs

[1/1][ORCHESTRATION: cloud-sanity-Save actual docker tag results JSON ou...
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Test container names/version]
[ \ ] 2mins 12secs

[1/1][ORCHESTRATION: cloud-sanity-Test container names/version]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Compute - Container version check]
[ - ] 2mins 21secs

[1/1][ORCHESTRATION: cloud-sanity-Compute - Container version check]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-CephMon - Check cephmon is running]
[ | ] 2mins 23secs

[1/1][ORCHESTRATION: cloud-sanity-CephMon - Check cephmon is running]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-CephMon - CEPH cluster check]
[ - ] 2mins 23secs

[1/1][ORCHESTRATION: cloud-sanity-CephMon - CEPH cluster check]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-CephMon - Check Ceph Mon status]
[ | ] 2mins 23secs

[1/1][ORCHESTRATION: cloud-sanity-CephMon - Check Ceph Mon status]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-copy ceph check file to /tmp]
```

```

[ - ] 2mins 24secs

[1/1][ORCHESTRATION: cloud-sanity-copy ceph check file to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Ceph mon sanity details]
[ | ] 2mins 24secs

[1/1][ORCHESTRATION: cloud-sanity-Ceph mon sanity details]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Check Ceph Mon Details Output]
[ \ ] 2mins 25secs

[1/1][ORCHESTRATION: cloud-sanity-Check Ceph Mon Details Output]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-remove ceph check file]
[ | ] 2mins 25secs

[1/1][ORCHESTRATION: cloud-sanity-remove ceph check file]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-CephOSD - Ping All Storage Nodes]
[ | ] 2mins 25secs

[1/1][ORCHESTRATION: cloud-sanity-CephOSD - Ping All Storage Nodes]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-copy ceph check file to /tmp]
[ / ] 2mins 28secs

[1/1][ORCHESTRATION: cloud-sanity-copy ceph check file to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Check cephmon is running]
[ \ ] 2mins 29secs

[1/1][ORCHESTRATION: cloud-sanity-Check cephmon is running]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Ceph OSD Sanity]
[ | ] 2mins 29secs

[1/1][ORCHESTRATION: cloud-sanity-Ceph OSD Sanity]
[ DONE! ]

```

```
[1/1][ORCHESTRATION: cloud-sanity-CephOSD - Check OSD result with osdinfo]
[ \ ] 2mins 29secs

[1/1][ORCHESTRATION: cloud-sanity-CephOSD - Check OSD result with osdinfo]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Check Ceph OSD Details Output without ...]
[ / ] 2mins 29secs

[1/1][ORCHESTRATION: cloud-sanity-Check Ceph OSD Details Output without ...]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-CephOSD - Check OSD result without osd...]
[ - ] 2mins 30secs

[1/1][ORCHESTRATION: cloud-sanity-CephOSD - Check OSD result without osd...]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-remove ceph check file]
[ | ] 2mins 30secs

[1/1][ORCHESTRATION: cloud-sanity-remove ceph check file]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Copy osd.check.py to /tmp]
[ | ] 2mins 30secs

[1/1][ORCHESTRATION: cloud-sanity-Copy osd.check.py to /tmp]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Run OSD maintenance for results data]
[ / ] 2mins 30secs

[1/1][ORCHESTRATION: cloud-sanity-Run OSD maintenance for results data]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-CephOSD - OSD Overall status]
[ / ] 2mins 30secs

[1/1][ORCHESTRATION: cloud-sanity-CephOSD - OSD Overall status]
[ DONE! ]

[1/1][ORCHESTRATION: cloud-sanity-Remove osd.check.py from /tmp]
[ \ ] 2mins 30secs

[1/1][ORCHESTRATION: cloud-sanity-Remove osd.check.py from /tmp]
```

```
[ DONE! ]
```

```
[1/1][ORCHESTRATION: cloud-sanity-Remove osdmgmt check OSDs results json...
```

```
[ \ ] 2mins 31secs
```

```
Skipping Kubernetes deployment
```

```
[1/1][ORCHESTRATION: cloud-sanity-Remove osdmgmt check OSDs results json...
```

```
[ DONE! ]
```

```
Ended Installation [ORCHESTRATION] [Success]
```

```
The logs for this run are available in 10.201.242.202:/var/log/mercury/fb358e3a-1d14-4a59-983c-58492275b358
```

# VM in an Unhealthy State

## Troubleshooting OpenStack when VM is down or not responsive.

### Step 1:

Check to see the UUID of the specific VM that is failing in the Overcloud. In the **openstack server list --all-projects** check the project and the specific VM that is failing.

```
[root@mgmt ~]# openstack server list --all-project
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID                | Name                | Status | Networks |
+-----+-----+-----+-----+
| a30ac904-474a-47cc-9735-03e6fc7e7ce2 | saegw-pgw-vdu-cf1-2 | ACTIVE | vpc-  
mgmt=192.168.68.215; global-  
orch=172.20.20.30; di-  
internal1=192.168.145.10  
| b305f5a5-e0e2-4d27-8133-ed540ef8915a | saegw-pgw-vdu-sf1-2 | ACTIVE |  
service2=192.168.142.12,  
192.168.142.13;  
service1=192.168.141.12,  
192.168.141.13; global-  
orch=172.20.20.33; di-  
internal1=192.168.145.13  
| 1d98c1c1-75bc-4966-a358-ccae98aae0b3 | saegw-pgw-vdu-sf1-1 | ACTIVE |  
service2=192.168.142.10,  
|
```

192.168.142.11;				
service1=192.168.141.10,				
192.168.141.11; global-				
orch=172.20.20.32; di-				
internal1=192.168.145.12				
b0db2c35-ac5b-478a-9441-34179c83250f	saegw-pgw-vdu-cf1-1	ACTIVE	vpc-	
mgmt=192.168.68.216; global-				
orch=172.20.20.31; di-				
internal1=192.168.145.11				
8e7ab228-5167-4e14-8eea-61b4ebf083af	saegw-pgw-em-3	ACTIVE	vpc-	
mgmt=192.168.68.207; global-	ultra-em			
orch=172.20.20.23				
e779af84-c097-4cf6-b983-cc22f1197e1c	saegw-pgw-em-2	ACTIVE	vpc-	
mgmt=192.168.68.206; global-	ultra-em			
orch=172.20.20.22				
f9c9d0d2-535b-4dbf-9c31-644deb03a526	saegw-pgw-em-1	ACTIVE	vpc-	
mgmt=192.168.68.205; global-	ultra-em			
orch=172.20.20.21				
ccbd69cf-b79c-4ee5-af5d-0672f877b90a	AUTO-VNF-1	ACTIVE	vpc-	
mgmt=192.168.68.8, 10.201.34.203;	Autovnf-Image			global-
orch=172.20.20.13				
95f248a4-2815-452d-841c-aacafed63dc5	pod4esc02	ACTIVE	vpc-	
mgmt=192.168.68.6; global-	Esc-Image			
orch=172.20.20.12				
047df3a3-869f-41ca-a365-9b669f85b3e2	pod4esc01	ACTIVE	vpc-	
mgmt=192.168.68.5; global-	Esc-Image			
orch=172.20.20.11				
+-----+-----+-----+-----+				
-----+-----+-----+-----+				

**Step 2:**

Access the compute node that is hosting that VM.

**openstack server show <server UUID>** - along with other info, it will report the actual VM state and which compute node it's running on. For example here is a running active SAEGW node.

```
[root@mgmt ~]# openstack server show b305f5a5-e0e2-4d27-8133-ed540ef8915a<
+-----+
| Field                                | Value
|<
+-----+
+-----+
| OS-DCF:diskConfig                    | MANUAL
|<
| OS-EXT-AZ:availability_zone          | nova
|<
| OS-EXT-SRV-ATTR:host                 | rcdn-c3-compute7-sriov
|<
| OS-EXT-SRV-ATTR:hypervisor_hostname | rcdn-c3-compute7-sriov
|<
| OS-EXT-SRV-ATTR:instance_name       | instance-000000c4
|<
| OS-EXT-STS:power_state               | Running
|<
| OS-EXT-STS:task_state                | None
|<
| OS-EXT-STS:vm_state                 | active
|<
| OS-SRV-USG:launched_at              | 2018-03-26T23:17:54.000000
|<
| OS-SRV-USG:terminated_at            | None
|<
| accessIPv4                          |
|<
| accessIPv6                          |
|<
| addresses                            | service2=192.168.142.12, 192.168.142.13;
service1=192.168.141.12, 192.168.141.13; |<
|                                       | global-orch=172.20.20.33; di-
internal1=192.168.145.13                |<
| config_drive                        | True
|<
```

```

| created                | 2018-03-26T23:17:18Z
|<
| flavor                 | sf-flavor (72400dbb-2372-40b4-aba1-
151ae083cdb8)          |<
| hostId                 |
2a0e1a140669f07281ea24bee24fbe3f2026bdaa58884879a0c51008
|<
| id                    | b305f5a5-e0e2-4d27-8133-ed540ef8915a
|<
| image                 | qvpc-sf (95cda377-d270-40f8-837c-
64cf0467231d)        |<
| key_name              | None
|<
| name                  | saegw-pgw-vdu-sf1-2
|<
| os-extended-volumes:volumes_attached | []
|<
| progress              | 0
|<
| project_id            | 67edb12b853345cd8aa29db2288056cf
|<
| properties            |
|<
| security_groups       | [{u'name': u'default'}, {u'name':
u'default'}, {u'name': u'default'}, {u'name':
|<
|                        | u'default'}, {u'name': u'default'},
{u'name': u'default'}]]
|<
| status              | ACTIVE
|<
| updated               | 2018-03-27T20:04:08Z
|<
| user_id               | c14534c3a3024d2b8bc4d0f319d306cc
|<
+-----+-----+
+-----+

```

Once in the compute node, use **virsh list --all** to check status of all VMs that the compute is hosting.

The server could be in the shutdown or error state or it could be destroyed completely. Shutdown means that it has been powered down but it still can be started and is listed by virsh list, whereas if destroyed, then it won't show up in the virsh list at all.

```

vir1@compute3:~$ virsh list --all
Id      Name                               State
-----
5       instance-00001c96                  running
47      instance-00001d46                  running
...
153     instance-00001f1c                  running
154     instance-00001fld                  running
-       instance-00001e64                  shut off
    
```

### Step 3:

Go to the compute node and examine logs for indications of an error in the `/var/log/libvirt/qemu` directory and examine the log of the format `instance-<instance #>.log`:

```
/var/log/nova/nova-compute.log
```

### Step 4:

If VM is created by VNFM, then use the recovery method described in the section VNF recovery in the chapter Troubleshooting.

If VM is created by OpenStack directly, then it can be reset with `virsh` command from the compute node:

```

novalibvirt_12614 [root@compute7-sriov qemu]# virsh reset instance-000000c4

novalibvirt_12614 [root@c3-compute7-sriov qemu]# virsh list --all
Id      Name                               State
-----
5       instance-000000c4                  running
    
```

**Step 5:**

Check the VM is recovered after resetting the VM from virsh.

If performing virsh reset of VM didn't recover, check the memory NIC on compute HW is functioning properly.

# VM Unreachable via Floating IP

## Step 1:

Check in OpenStack that both CF VM are in ACTIVE state

Login to the OSP-D and source the Overcloud corerc file:

```
[stack@ultram-ospd ~]$ nova list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID                                     | Name
| Status | Task State | Power State | Networks
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| e50095d8-5205-4f84-8274-9887448d5f37 | auto-deploy-ISO-590-uas-0
| ACTIVE | -          | Running    | mgmt=172.16.181.8, 10.201.206.42
|
| 65db9fe5-2d2b-4892-b292-9b1e15f834fb | auto-it-vnf-ISO-590-uas-0
| ACTIVE | -          | Running    | mgmt=172.16.181.10, 10.201.206.41
|
| 2b9898be-0c5b-4eda-bc75-3b42eafb0cba | autovnf1-uas-0
| ACTIVE | -          | Running    | orchestr=172.16.180.7; mgmt=172.16.181.13
|
| f7ca6bd3-2e43-46a6-ab90-eeaced58166 | autovnf1-uas-1
| ACTIVE | -          | Running    | orchestr=172.16.180.9; mgmt=172.16.181.3
|
| 60ad49d5-719c-4a22-98e5-b4a5605db952 | autovnf1-uas-2
| ACTIVE | -          | Running    | orchestr=172.16.180.6; mgmt=172.16.181.12
|
| 9775f2a5-5841-4dd4-84c9-70324bf9be09 | ultram-vnfm1-ESC
| ACTIVE | -          | Running    | orchestr=172.16.180.11; mgmt=172.16.181.15,
10.201.206.46
|
| 2bf23264-a237-449d-a8c4-0cac781a3647 | ultram-vnfm1-em_ultram_0_bc981c0f-53d9-4eb0-
89ff-4066ddfc7774 | ACTIVE | -          | Running    | orchestr=172.16.180.3;
mgmt=172.16.181.4
|
| e239b9b2-e836-4cb2-904e-1969749950b1 | ultram-vnfm1-em_ultram_0_d0f0de76-5fa0-431e-
a40e-708fe9077a50 | ACTIVE | -          | Running    | orchestr=172.16.180.5;
mgmt=172.16.181.6
|
```

```
| c2a63649-4d71-4d25-a815-bf27475e4a63 | ultram-vnfm1-em_ultram_0_dec5cdba-facb-44ab-
be6b-7e47c1e60216 | ACTIVE | - | Running | orchestr=172.16.180.4;
mgmt=172.16.181.5 |
| caf9e049-8739-483f-9c18-9840ee23ea8e | vnfd1-deployment_c1_0_9ece3e13-0ab7-4543-
80ed-ce834e1d255e | ACTIVE | - | Running | orchestr=172.16.180.12;
ultram-vnfm1-di-internall=192.168.1.16; mgmt=172.16.181.21 |
| 6ef5bf1c-25d8-4c7e-862c-3e87a2371848 | vnfd1-deployment_c2_0_b0c371ee-57b8-4e95-
9c2e-771150e5ecc5 | ACTIVE | - | Running | orchestr=172.16.180.19;
ultram-vnfm1-di-internall=192.168.1.6; mgmt=172.16.181.14 |
| 308b0303-f665-4368-9640-679d7fe13189 | vnfd1-deployment_s3_0_e7b44362-606a-42fe-
a66b-010d8152582e | ACTIVE | - | Running | ultram-vnfm1-service-
network1=10.10.10.4; orchestr=172.16.180.16; ultram-vnfm1-di-internall=192.168.1.12 |
| 546a6194-1b99-448f-8fe0-0cc54c42c154 | vnfd1-deployment_s4_0_1862bfc9-a0f6-4d16-
9b5f-1c7141b5455a | ACTIVE | - | Running | ultram-vnfm1-service-
network1=10.10.10.8; orchestr=172.16.180.21; ultram-vnfm1-di-internall=192.168.1.9 |
| caeb56da-86d4-41d3-942e-0ce2866a6ebc | vnfd1-deployment_s5_0_cb6dceb2-28f0-4b2f-
a84e-336850f40b1e | ACTIVE | - | Running | ultram-vnfm1-service-
network1=10.10.10.12; orchestr=172.16.180.15; ultram-vnfm1-di-internall=192.168.1.3 |
| 085bbf4e-d87d-491d-afa6-0398da6e7231 | vnfd1-deployment_s6_0_eee43d0b-0ec3-4624-
9dfe-d512a381281f | ACTIVE | - | Running | ultram-vnfm1-service-
network1=10.10.10.2; orchestr=172.16.180.10; ultram-vnfm1-di-internall=192.168.1.8 |
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
[stack@ultram-ospd ~]$
```

**Step 2:**

Check if Floating IP is working.

Check whether only that VM floating IP is not working, or all floating IPs are not working in the Ultra M deployment.

If all floating IPs are not working, then check the Neutron router IP is reachable and check the external connectivity in switch and router.

Issue the following commands, verify the status, and take recovery action as per the scenario below:

```
nova list --fields name,host,instance_name,status
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
```

ID	Name
Host	Instance Name
9775f2a5-5841-4dd4-84c9-70324bf9be09	ultram-vnfm1-ESC
ultram-rcdnlab-compute-0.localdomain	instance-000001b2
2bf23264-a237-449d-a8c4-0cac781a3647	ultram-vnfm1-em_ultram_0_bc981c0f-53d9-4eb0-89ff-4066ddfc7774
ultram-rcdnlab-compute-1.localdomain	instance-000001b5
e239b9b2-e836-4cb2-904e-1969749950b1	ultram-vnfm1-em_ultram_0_d0f0de76-5fa0-431e-a40e-708fe9077a50
ultram-rcdnlab-compute-0.localdomain	instance-000001b8
c2a63649-4d71-4d25-a815-bf27475e4a63	ultram-vnfm1-em_ultram_0_dec5cdba-facb-44ab-be6b-7e47c1e60216
ultram-rcdnlab-osd-compute-2.localdomain	instance-000001bb
caf9e049-8739-483f-9c18-9840ee23ea8e	vnfd1-deployment_c1_0_9ece3e13-0ab7-4543-80ed-ce834e1d255e
ultram-rcdnlab-compute-0.localdomain	instance-000001cd
6ef5bf1c-25d8-4c7e-862c-3e87a2371848	vnfd1-deployment_c2_0_b0c371ee-57b8-4e95-9c2e-771150e5ecc5
ultram-rcdnlab-compute-1.localdomain	instance-000001be
308b0303-f665-4368-9640-679d7fe13189	vnfd1-deployment_s3_0_e7b44362-606a-42fe-a66b-010d8152582e
ultram-rcdnlab-compute-0.localdomain	instance-000001c1
546a6194-1b99-448f-8fe0-0cc54c42c154	vnfd1-deployment_s4_0_1862bfc9-a0f6-4d16-9b5f-1c7141b5455a
ultram-rcdnlab-compute-1.localdomain	instance-000001c4
caeb56da-86d4-41d3-942e-0ce2866a6ebc	vnfd1-deployment_s5_0_cb6dceb2-28f0-4b2f-a84e-336850f40b1e
ultram-rcdnlab-osd-compute-0.localdomain	instance-000001c7
085bbf4e-d87d-491d-afa6-0398da6e7231	vnfd1-deployment_s6_0_eee43d0b-0ec3-4624-9dfe-d512a381281f
ultram-rcdnlab-osd-compute-1.localdomain	instance-000001ca

Check the floating IP state in Neutron:

```
neutron floatingip-list
```

id	fixed_ip_address	floating_ip_address
port_id		
22a06c06-85dd-44a9-8995-4eab6ddab13c	172.16.181.10	10.201.206.41
6da8f093-d839-4990-8dae-85b2b0477922		
79bd0694-d4c1-4b1a-acab-2b5dde2b67cf	172.16.181.15	10.201.206.46
bfeb8b8b-7c00-4d53-a87a-70a1126e7fcb		
c7f52924-ff66-4363-b101-c302d0b049d3	172.16.181.101	10.201.206.43
211f7e35-2d57-4a33-9460-036b403d1d22		
d887d359-2791-4562-afba-8d506dc97c1f	172.16.181.7	10.201.206.47
f3c0ae00-d065-473c-a66e-955a34f1a89d		
de930dd7-d773-4c44-9878-1e642de3d90c	172.16.181.9	10.201.206.45

```

ee15d874-b81b-4a7a-b1f8-ebb0f20eed7d |
| e7281d3a-b6b3-42dd-8222-7c92acb1f703 | 172.16.181.8 | 10.201.206.42 |
39d3ff18-073f-41b8-8846-5814d2d05b00 |
+-----+-----+-----+-----+
-----

[stack@ultram-ospd ~]$ neutron floatingip-show 22a06c06-85dd-44a9-8995-4eab6ddab13c
+-----+-----+
| Field | Value |
+-----+-----+
| created_at | 2018-01-21T02:16:20Z |
| description | |
| fixed_ip_address | 172.16.181.10 |
| floating_ip_address | 10.201.206.41 |
| floating_network_id | 102051a7-448e-467e-b04a-feeddf830f71 |
| id | 22a06c06-85dd-44a9-8995-4eab6ddab13c |
| port_id | 6da8f093-d839-4990-8dae-85b2b0477922 |
| project_id | c7ba964cf10049cfb3d34ebde5103a06 |
| revision_number | 10 |
| router_id | c4ca5644-4be7-4127-8065-5ad8b6ecf51c |
| status | ACTIVE |
| tenant_id | c7ba964cf10049cfb3d34ebde5103a06 |
| updated_at | 2018-03-15T18:07:16Z |
+-----+-----+

```

**Step 3:**

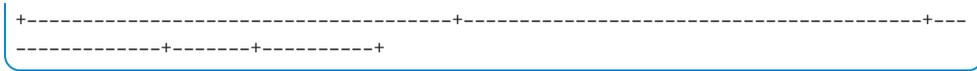
Check Neutron router status.

Confirm that Neutron router is alive:

```

[stack@ultram-ospd ~]$ neutron l3-agent-list-hosting-router main
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
| id | host |
+-----+-----+-----+-----+-----+
admin_state_up | alive | ha_state |
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
| 9a873cd8-dfe8-48f2-b585-c67eddf03460 | ultram-rcdnlab-controller-2.localdomain |
True | :- ) | standby |
| bdd36044-7332-4a5e-8541-e65b90f83f44 | ultram-rcdnlab-controller-1.localdomain |
True | :- ) | standby |
| 41129149-786c-45ad-ac38-a82b92527fe3 | ultram-rcdnlab-controller-0.localdomain |
True | :- ) | active |

```



**Step 4:**

Check the controller l3-agent status is up

Confirm that l3-agent is alive:

```
neutron agent-list | grep -i l3
| 41129149-786c-45ad-ac38-a82b92527fe3 | L3 agent          | ultram-rcdnlab-
controller-0.localdomain | nova              | :-) | True          | neutron-l3-
agent                    |
| 9a873cd8-dfe8-48f2-b585-c67eddf03460 | L3 agent          | ultram-rcdnlab-
controller-2.localdomain | nova              | :-) | True          | neutron-l3-
agent                    |
| bdd36044-7332-4a5e-8541-e65b90f83f44 | L3 agent          | ultram-rcdnlab-
controller-1.localdomain | nova              | :-) | True          | neutron-l3-
agent                    |
```

**Step 5:**

Check the security group relevant configuration.

Confirm that IP addresses are allowed in the security group:

```
[stack@ultram-ospd ~]$ nova secgroup-list-rules default

WARNING: Command secgroup-list-rules is deprecated and will be removed after Nova
15.0.0 is released. Use python-neutronclient or python-openstackclient instead.
+-----+-----+-----+-----+-----+
| IP Protocol | From Port | To Port | IP Range | Source Group |
+-----+-----+-----+-----+-----+
| icmp        | -1        | -1      | 0.0.0.0/0 |               |
| tcp         | 1         | 65535   | 0.0.0.0/0 |               |
|             |           |         |           | default      |
| udp         | 1         | 65535   | 0.0.0.0/0 |               |
|             |           |         |           | default      |
+-----+-----+-----+-----+-----+
```

**Step 6:**

Restart the neutron agent service.

Login into active controller and restart the Neutron-l3-agent.service:

```
systemctl restart neutron-l3-agent.service
```

**Step 7:**

If the floating IP is still not reachable, then try to do a packet capture on the active controller.

Login to the active controller. Go to network namespace and collect the packet capture. See whether packets are reaching the Neutron router and return traffic is coming from VM:

```
ip netns
qdhcp-ff2a1cc3-9daa-41f6-9d73-abd8e2954f34
qdhcp-ddecae0c-a638-47fa-9923-e2a0030c0ce0
qrouter-c4ca5644-4be7-4127-8065-5ad8b6ecf51c
qdhcp-a5431de6-f7ce-4fb7-bc73-d4a0c038d1e3

ip netns exec qrouter-c4ca5644-4be7-4127-8065-5ad8b6ecf51c sudo ip add show
ip netns exec qrouter-c4ca5644-4be7-4127-8065-5ad8b6ecf51c sudo tcpdump -i
<interface>
```

**Step 8:**

If the issue is still present, make sure that return route is installed on the CF VM. To verify or add the route, connect to the CF VM Console (StarOS) via OpenStack Horizon Dashboard.

## vnfd1-deployment\_c2\_0\_b0c371ee-57b8-4e95-9c2e-771150e5ecc5

Overview Log Console Action Log

### Instance Console

If console is not responding to keyboard input: click the grey status bar below. [Click here to show only console](#)  
To exit the fullscreen mode, click the browser's back button.

```

Connected (unencrypted) to: QEMU (instance-00001be)
trator user unknown on device /dev/pts/0 from remote-ip 0.0.0.0 has gained access
s to the StarOS shell
2018-Mar-30*11:34:01.560 [cli 30052 unusual] [1/0/22222 <cli:1022222> ommands_de
bug.c:2283] [software internal system critical-info syslog] CLI Security Adminis
trator user unknown on device /dev/pts/0 from remote-ip 0.0.0.0 has gained acces
s to the StarOS shell
2018-Mar-30*11:34:01.641 [cli 30052 unusual] [1/0/22222 <cli:1022222> ommands_de
bug.c:2283] [software internal system critical-info syslog] CLI Security Adminis
trator user unknown on device /dev/pts/0 from remote-ip 0.0.0.0 has gained acces
s to the StarOS shell

Cisco Systems QvPC-DI Intelligent Mobile Gateway
RCDN-LAB login:
Cisco Systems QvPC-DI Intelligent Mobile Gateway
RCDN-LAB login:
Cisco Systems QvPC-DI Intelligent Mobile Gateway
RCDN-LAB login:
Cisco Systems QvPC-DI Intelligent Mobile Gateway
RCDN-LAB login: admin
Password:
Last login: Fri Mar 30 11:50:11 -0400 2018 on pts/2 from 10.131.33.60.
Cisco Systems QvPC-DI
[local]RCDN-LAB# config
[local]RCDN-LAB(config)# context local
[local]RCDN-LAB(config-ctx)# ip route 0.0.0.0 0.0.0.0 172.16.181.1 _

```

From there proceed to configure

```
ip route 0.0.0.0 0.0.0.0 <GW IP > LOCAL1
```

# Di Network Congestion

Di Network utilizes the heartbeat mechanism to maintain the awareness of the internal Di Network.

High-Availability Task (HAT) is monitoring the heartbeats between the VMs.

During the periods of peak traffic, the Di Network may get congested. This section will provide the troubleshooting guidelines for such event.

In the event that the Di Network is overloaded, the typical consequence presented to the VPC will likely be an SF or CF card reload. This, however, does not mean that in the case of the SF/CF reload the root cause is always related to the Di Network congestion.

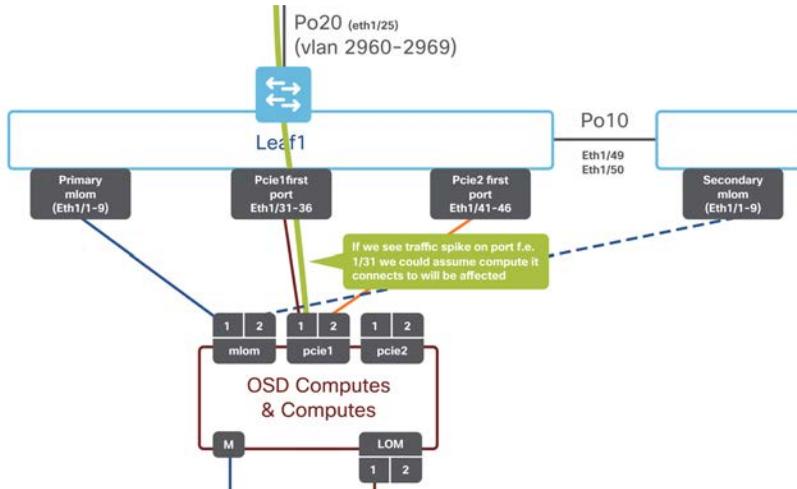
Active CF monitors the health state of the SFs via the heartbeat mechanism. SF/CF will always have links connecting to the two different Leaf Switches - one active at the time. SF will be declared dead and rebooted intentionally when the following conditions are met:

1. Loss of 2 consecutive heartbeats with a 1.2-second timeout each and a 3rd heartbeat timeout of 0.2 seconds.
2. CF hasn't received an IPC advertisement from that SF recently (usually 5-8 seconds)

If there is SF/CF reboot involved, normally during the troubleshooting procedure a check would be made to ensure whether it was caused by the heartbeat loss.

Except for the external factors, Di Network may get higher traffic due to the internal factors such as the network issues on the Nexus switch, the port being down, or one of the SR-IOV interfaces on the UCS going down.

When troubleshooting the Di Network, the first step is to have good networking understanding. Identify the port number on the Nexus switches that provide the Di Network connectivity:



Due to the nature of the SR-IOV, when troubleshooting the Di Network issues, the available troubleshooting points are Nexus switches or the VPC. The following data may be collected/monitored:

## Network Side

1. KPI network loads graphs from the Leaf/Spine switches, monitoring the load on the SGi, S1-U and S5 interfaces. This comes outside of the Ultra M setup and typically has to be set up up front.
2. Nexus 9000 packet statistics. To obtain this data, execute the following commands where x/y represents the affected interface on the Nexus 9000 side:

```
show interface x/y
show interface counters errors
show queuing interface ethernet x/y
show policy-map interface ethernet x/y
```

Module-specific counters, available after executing **attach module** commands

```
show module
attach module <>
show hardware internal ns fabric interface counters
show hardware internal interface counters nz
show hardware internal fabric interface counters nz
```

In case of the Di Network overload, the drop counters will be increasing, for example:

```
Ethernet1/1 is up
admin state is up, Dedicated Interface

RX
 337014428000 unicast packets  8633 multicast packets  122126705 broadcast
packets
 337136563338 input packets  314410270114613 bytes
 6885425392 jumbo packets  0 storm suppression bytes
 0 runts  0 giants  0 CRC  0 no buffer
 0 input error  0 short frame  0 overrun  0 underrun  0 ignored
 0 watchdog  0 bad etype drop  0 bad proto drop  0 if down drop
 0 input with dribble  0 input discard
 0 Rx pause

TX
 341645702819 unicast packets  189318469 multicast packets  2458841623 broadcast
packets
 344293862911 output packets  311921566725116 bytes
 6318341308 jumbo packets
 0 output error  0 collision  0 deferred  0 late collision
 0 lost carrier  0 no carrier  0 babble  11838268 output discard
 0 Tx pause
```

By checking deeper for appropriate interfaces, discards which are due to tail drops can be isolated:

```
LEAF# show interface counters errors
-----
Port          Align-Err  FCS-Err  Xmit-Err  Rcv-Err  UnderSize  OutDiscards
-----
Eth1/11      0          0          0          0          0          30480599

LEAF# show queuing interface ethernet
Egress Queuing for Ethernet1/11 [System]
```

```

+-----+
|                                     |
|                               QOS GROUP 0                               |
|-----+-----+
|                               | Unicast | Multicast |
|-----+-----+
| Tx Pkts | 63627373029 | 58258473 |
| Tx Byts | 58943285501690 | 12841719144 |
| WRED/AFD & Tail Drop Pkts | 30480599 | 0 |
| WRED/AFD & Tail Drop Byts | 38598658618 | 0 |
| Q Depth Byts | 0 | 0 |
| WD & Tail Drop Pkts | 30480599 | 0 |
+-----+-----+

LEAF# show policy-map interface ethernet 1/11
  Class-map (queuing):  c-out-8q-q-default (match-any)
    bandwidth remaining percent 100
    queue dropped pkts : 30480599
    queue depth in bytes : 894

```

For a more detailed explanation about Nexus 9000 queuing mechanisms, please refer to: Platform Buffer and Queueing Architecture Whitepaper <https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-732452.html>

A full root cause analysis would depend on the source of the peak traffic.

The potential solution to the above-demonstrated step is to best discussed with Cisco Technical Assistance Center and they may include:

- Changing the buffer profile to burst, to better handle the microbursts at the hardware level.
- Deploying the flow control mechanisms, to smooth out the traffic received on the interfaces.

## VPC Side

### Cloud Monitor

The internal High-Availability Task (HAT) is always monitoring the heartbeats across the VMs on the internal Di Network. This information can be displayed at any time using the **show cloud monitor di-network summary** exec mode command:

```
[local]UGP# show cloud monitor di-network summary
Card 3 Heartbeat Results:
ToCard  Health    5MinLoss    60MinLoss
  1      Good      0.00%      0.00%
  2      Good      0.00%      0.00%
  4      Good      0.00%      0.00%
  5      Good      0.00%      0.00%
```

To display information about the installed hardware IFTASK info, use **show cloud hardware iftask** command:

```
[local]UGP# show cloud hardware iftask
Card 1:
  Total number of cores on VM:      8
  Number of cores for PMD only:     0
  Number of cores for VNPU only:    0
  Number of cores for PMD and VNPU: 2
  Number of cores for MCDMA:        0
  Number of cores for Crypto:       0
  Hugepage size:                    2048 kB
  Total hugepages:                  3670016 kB
  NPUSHM hugepages:                 0 kB
  CPU flags: avx sse sse2 ssse3 sse4_1 sse4_2
  Poll CPU's: 1 2
  KNI reschedule interval: 5 us
```

To display the performance information about Di Network, use the **show cloud performance dinet pps** command:

```
[local]UGP# show cloud performance dinet pps
----- Average DINet Performance (in Kpps) -----
Card      Current          5min          15min
          Rx      Tx      Rx      Tx      Rx      Tx
-----
```

1	0.194	0.089	0.183	0.087	0.183	0.087
2	0.222	0.172	0.213	0.155	0.214	0.156
3	0.102	0.052	0.102	0.050	0.101	0.050
4	0.107	0.054	0.102	0.049	0.102	0.049
5	0.099	0.040	0.099	0.038	0.099	0.038

## IFTASK

DI-net and service ports are controlled by the DPDK Internal Forwarder (IFTASK) process. This process is responsible for packet processing in the Ultra M Solution. Internal Forwarder would be responsible for receiving packets from the device and would provide a fast path for packet processing in user space, by bypassing the Linux networking and Linux IP stack completely.

The information about IFTASK process will be displayed with internal command **show iftask stats summary**:

```
[local]UGP# show iftask stats summary

/truncated output/

-----
Counter                SF3          SF4          SF5          ___TOTAL___
-----
svc_rx                  23           16           0            39
svc_tx                  15           21           22           58
di_rx                   22           21           22           65
di_tx                   61           16           0            77
__ALL_DROPS__          0            0            0            0
svc_tx_drops            0            0            0            0
di_rx_drops             0            0            0            0
di_tx_drops             0            0            0            0
sw_rss_enq_drops        0            0            0            0
kni_thread_drops        0            0            0            0
kni_drops               0            0            0            0
medma_drops             0            0            0            0
```

## Heartbeat Monitors

The internal command **show heartbeat stats card x cpu** will provide information per card per facility:

```
[local]UGP# show heartbeat stats card 3 cpu 0
facility sitmain instance 30 intf CPBOND:
hb 2025395 :03.488, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025396 :04.489, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025397 :05.489, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025398 :06.490, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025399 :07.491, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025400 :08.492, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025401 :09.493, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025402 :10.493, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025403 :11.493, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
hb 2025404 :12.494, RV, S fffffe430 ffffffff81132809, cp 0, tx 0ms, rx 0ms, rtt 0ms
```

In case of the heartbeat loss, it is important to check the data in the **debug console card x cpu x** to identify the length and level of packet drops.

The traditional command **debug console card x cpu x**, also available from the SSD, will provide information on the current and historic heartbeats from/towards the specific card.

This output is very useful when troubleshooting Di Network congestion issues.

```
***** debug console card 10 cpu 0 tail 10000 only *****
...
card10-cpu0: di-net heartbeat to-card 4 current (5:36/36) and history :
card10-cpu0: 43:0/60 42:0/60 41:0/60 40:0/60 39:0/60 38:0/60 37:0/60 36:0/60 35:0/60
card10-cpu0: 34:0/60 33:0/60 32:0/60 31:0/60 30:0/60 29:0/60 28:0/60 27:0/60 26:0/60
25:0/0
```

The output is to be read as follows:

<b>Current Period</b>	Drops: Heartbeats Sent in Last Minute / Time Offset
<b>Historic Period</b>	Time Offset: Drops/ Heartbeat Sents in Last Minute

Time index indicates the current time moment in Last One Minute Period. For example, 5 indicates that 5th second in the last one-minute interval.

In the sample above:

**di-net heartbeat to-card 4 current (5:36/36)** - This indicates the 36th second of the last one-minute as a reference period and displays that 36 heartbeats have been sent and 5 drops occurred.

In the CLI output **43:0/60 42:0/60 41:0/60** - In minute 43 there were 0 drops, in minute 42 there were 0 drops, out of the 60 heartbeats sent in total.

### Increasing the Di Network heartbeat loss period

In order to make the Di Network less susceptible to missed heartbeats, configure the following command in StarOS:

```
(config)# high-availability fault-detection card hb-loss 10
```

This configuration option is available in StarOS Release 21.8 or 21.7.3 and above.

### Bulkstats Data and System Data Collection during the period of Di Network congestion

When Di Network is congested, collecting SSD adds to the congestion and hence it can pose a challenge in obtaining information for troubleshooting. In order to overcome the limitation of overloading an already congested Di Network, there are two options available for gathering crucial data:

- Gather Bulkstats for Di Network

To collect the Bulkstats the following schema should be enabled upfront:

```
card schema cardSch17 format
PPM,card,cardSch17,%epochtime%,%localdate%,%localtime%,%uptime%,%card%,%dinet-rxpmts-
curr%,%dinet-tpxmts-curr%,%dinet-rxpmts-5minave%,%dinet-tpxmts-5minave%,%dinet-
```

```

rxpkts-15minave%,%dinet-txpks-15minave%,%dinet-txdrops-curr%,%dinet-txdrops-
5minave%,%dinet-txdrops-15minave%,%iftask-errors%,

port schema portSch3 format
PPM,port,portSch3,%epochtime%,%localdate%,%localtime%,%uptime%,%card%,%port%,%util-
rxpkts-curr%,%util-txpks-curr%,%util-rxpks-5min%,%util-txpks-5min%,%util-rxpks-
15min%,%util-txpks-15min%,%util-txdrops-curr%,%util-txdrops-5min%,%util-txdrops-
15min%,

card schema cardSch6 format
PPM,card,cardSch6,%epochtime%,%localdate%,%localtime%,%uptime%,%card%,%task-allmgr-
num%,%task-allmgr-maxcpu%,%task-allmgr-maxmem%,%task-l2tpmgr-num%,%task-l2tpmgr-
maxcpu%,%task-l2tpmgr-maxmem%,%task-famgr-num%,%task-famgr-maxcpu%,%task-famgr-
maxmem%,%task-hamgr-num%,%task-hamgr-maxcpu%,%task-hamgr-maxmem%,%task-acsmgr-
num%,%task-acsmgr-avgcpu%,%task-acsmgr-avgmem%,%task-acsmgr-maxcpu%,%task-acsmgr-
maxmem%,%task-vpnmgr-num%,%task-vpnmgr-maxcpu%,%task-vpnmgr-maxmem%,%npuutil-now%,

card schema cardSch7 format
PPM,card,cardSch7,%epochtime%,%localdate%,%localtime%,%uptime%,%card%,%npuutil-
5minave%,%npuutil-15minave%,%npuutil-rxbytes-5secave%,%npuutil-txbytes-
5secave%,%npuutil-rxbytes-5minave%,%npuutil-txbytes-5minave%,%npuutil-rxbytes-
15minave%,%npuutil-txbytes-15minave%,%npuutil-rxpks-5secave%,%npuutil-txpks-
5secave%,%npuutil-rxpks-5minave%,%npuutil-txpks-5minave%,%npuutil-rxpks-
15minave%,%npuutil-txpks-15minave%,

```

The above schemas collect information related to Di Network, iftask and NPUSIM at periodic intervals and will help in isolating the issue, if any, and allow Cisco Engineering to make suitable recommendations.

- Configure SDR

System Data Collection Report executes a set of predefined commands that can be executed periodically. For virtualization, you need 21.8 or 21.7.3 StarOS release. The following configuration should be added to enable SDR:

```

config
  no support collection
  no support record all
  support record section show_pgw_service_stats command "show pgw-service statistics
all"
  support record section show_sgw_service_stats command "show sgw-service statistics
all"
  support record section show_egtpc_stats command "show egtpc statistics verbose"
  support record section show_egtpc_stats_interface_sgw_ingress command "show egtpc
statistics interface sgw-ingress"
  support record section show_egtpc_stats_interface_sgw_egress command "show egtpc
statistics interface sgw-egress"
  support record section show_egtpc_stats_interface_pgw_ingress command "show egtpc
statistics interface pgw-ingress"
  support record section show_diameter_stats command "show diameter statistics"
  support record section show_acs_nat_stats command "show active-charging nat
statistics"
  support record section show_port_utilization command "show port utilization table"
  support record section show_cpu_info_verbose command "show cpu info verbose"
  support record section show_session_disconnect_reasons command "show session
disconnect-reasons verbose"
  support record section show_apn_stats command "show apn statistics all"
  support record section show_subscriber_data_rate_apn_wildcard command "show
subscriber data-rate apn *"
  support record section show_npu_utilization_table command "show npu utilization
table"
  support record section show_npumgr_dinet_utilization_pps command "show npumgr dinet
utilization pps"
  support record section show_npumgr_utilization_info command "show npumgr
utilization info"
  support record section show_rct_stats command "show rct stats"
  support record section show_iftask_stats_summary command "show iftask stats
summary"
  support record section show_cloud_monitor_di_network_detail command "show cloud
monitor di-network detail"
  support record section show_cloud_monitor_di_network_summary command "show cloud
monitor di-network summary"
  support record section show_cloud_performance_dinet_pps command "show cloud
performance dinet pps"
  support record section show_cloud_performance_port command "show cloud performance

```

```
port"  
  support collection sleep-duration minutes 15  
exit
```

Save config:

```
save configuration
```

Validate record collection:

```
dir /hd-raid/support/records
```

should show records collected. One file every 15 minutes.

# CF/SF Unplanned Reboot or Stuck in Boot Process

The following are various components that may cause VPC CF/SF reboot or SF/CF stuck in booting scenarios:

- VPC (StarOS)
- ESC
- Openstack
- QEMU
- UCS
- Red Hat OS on the compute

This section will provide guidelines to narrow down the source of the issue to the relevant component.

## Symptom: CF/SF reboot or stuck in booting

The initial notification will be typically received from the VPC (StarOS) side as SNMP Trap:

```
Fri Nov 24 05:27:19 2017 Internal trap notification 60 (CardDown) card 4 type 2-Port  
Service Function Virtual Card UUID 151F94FB-F538-4602-ABAB-6B554D4396D7
```

This trap identifies that the card went down.

```
Fri Nov 24 05:33:19 2017 Internal trap notification 9 (CardBootFailed) card 4 type 2-  
Port Service Function Virtual Card UUID 151F94FB-F538-4602-ABAB-6B554D4396D7
```

The trap identifies that the card failed to boot subsequently. There are around 5 minutes difference in between the traps in the record above.

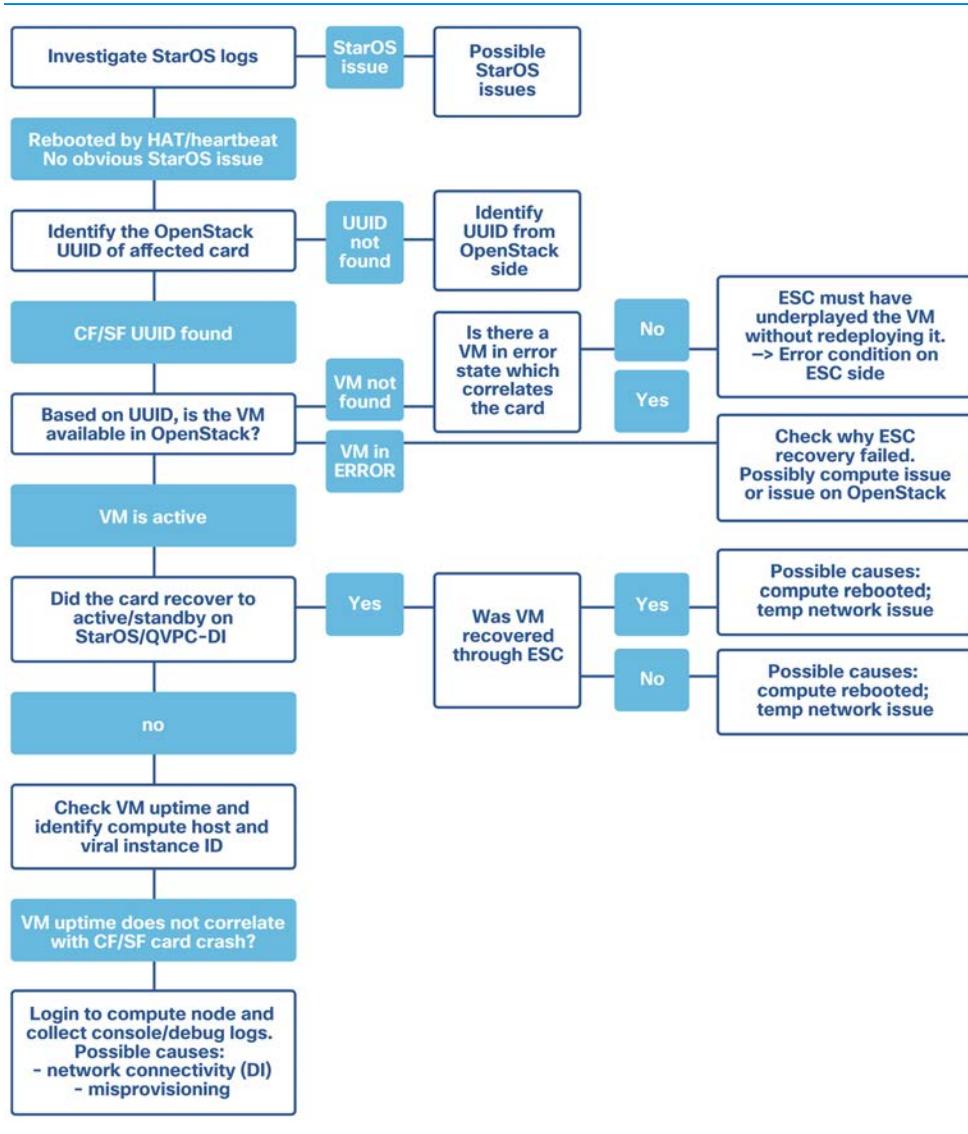
After the event has occurred, a check of **show card table (all)** output, will show a range of symptoms in any of the following:

- Normal Active or Standby state.
- Booting state.
- Not being present at all (card missing from show card table all output totally).

After a certain number of unsuccessful reboots, the card disappears from the inventory and can only be seen as None in 'show card table all'

## Troubleshooting Flow Chart

The following flowchart can be followed to isolate the failed component:



### Step 1:

Investigate StarOS logs

Relevant logs from SSD to be checked:

```
show card table all
show rct stats verbose
show snmp trap history verbose (search for ManagerFailure or CardDown)
show logs
```

Console logs from CFs (1 and 2), from the affected SF, and also from a random working SF

```
debug console card 1 processor 0
debug console card 2 processor 0
debug console card x processor 0
show persistdump list card 1 (followed by show persistdump display)
show persistdump list card 2 (followed by show persistdump display)

[local]UltraM# show persistdump list card 2
20171130-160627-68066-3
[local]UltraM#
[local]UltraM# show persistdump display card 2 name 20171130-160627-68066-3 | more
Thursday November 30 11:43:53 EST 2017
Displaying 20171130-160627-68066-3.gz...
Console logs dump start at 20171130-160627
----- START -----
Chassis      : QvPC-DI
Serial No    :
Failed Card  : 3
SPC Cards    : 1 2
Master SPC   : 2
Build       : 21.1 (68066)

[local]UltraM# show persistdump display card 2 name 20171130-160627-68066-3 | grep --
---
Thursday November 30 11:46:16 EST 2017
----- START -----
----- CONSOLE card 3 cpu 0 -----
2017-Nov-30+06:23:54.750 card 3-cpu0: <4>[ 174.825933] -----> Start of packet
dump (80 bytes) <-----
```

```

2017-Nov-30+06:23:54.754 card 3-cpu0: <4>[ 174.831746] -----> End of packet
dump <-----
2017-Nov-30+11:06:20.802 card 3-cpu0: ----- START -----
2017-Nov-30+11:06:20.804 card 3-cpu0: ----- ARP DUMP VR 0 -----
2017-Nov-30+11:06:20.806 card 3-cpu0: ----- PING -----
2017-Nov-30+11:06:21.215 card 3-cpu0: ----- COLLECT IFTASK DEBUG INFO ----
-----
2017-Nov-30+11:06:21.464 card 3-cpu0: ----- END -----
----- CONSOLE card 3 cpu 1 -----
----- CONSOLE card 3 cpu 2 -----
----- CONSOLE card 1 cpu 0 -----
2017-Nov-30+06:24:06.742 card 1-cpu0: <4>[ 87.450520] -----> Start of packet
dump (80 bytes) <-----
2017-Nov-30+06:24:06.746 card 1-cpu0: <4>[ 87.459295] -----> End of packet
dump <-----
----- CONSOLE card 2 cpu 0 -----
2017-Nov-30+06:22:53.441 card 2-cpu0: <4>[ 59.757505] -----> Start of packet
dump (80 bytes) <-----
2017-Nov-30+06:22:53.445 card 2-cpu0: <4>[ 59.762268] -----> End of packet
dump <-----
----- IFTASK CFG card3 -----
----- IFTASK CARD card3 -----
----- IFTASK PORTS card3 -----
----- BOXER card3 -----
----- MCDMA SUMMARY card3 -----
----- -/-- -----
----- IFCONFIG card3 -----
----- IFTASK STATS card3 -----
----- MCDMA DEV card3 -----
----- IFTASK PORT DETAILS card3 -----
----- IFTASK LOGS card3 -----
APP: ----Start: Dumping mempool mbuf_pool_data_node_0 0x7fff9e009980 ----
APP: -----
APP: ----Start: Dumping mempool mbuf_pool_ctl_node_0 0x7fff901875c0 ----
APP: -----
----- PORT 0 INFO -----
----- PORT 1 INFO -----
----- PORT 2 INFO -----
----- IFTASK.PY LOGS card3 -----
----- DMSG card3 -----
[ 174.825933] -----> Start of packet dump (80 bytes) <-----
[ 174.831746] -----> End of packet dump <-----
----- END -----

```

**Step 2:**

Identify UUID of the affected SF/CF

The UUID refers to the OpenStack UUID of the VM associated with the SF card. If the CF/SF is redeployed through the ESC the UUID may change.

- Option 1: Card present on the system

To identify the UUID from the StarOS, execute the **show card hardware x** command for the corresponding card to:

```
[local]UltraM# show card ha 3
Thursday November 30 10:19:03 EST 2017
Card 3:
  Card Type           : 2-Port Service Function Virtual Card
  CPU Packages        : 8 [#0, #1, #2, #3, #4, #5, #6, #7]
  CPU Nodes           : 1
  CPU Cores/Threads   : 8
  Memory              : 16384M (vpc-di-medium)
  UUID/Serial Number  : 3020A5AB-F0B7-475A-BE78-EE82E2B6DD1
```

- Option 2: Card not present on the system

If the card is not available on the system (not visible in **show card table** output) the above command will not be available. In that case, obtain the UUID from the historical SNMP traps

```
Fri Nov 24 05:27:19 2017 Internal trap notification 60 (CardDown) card 4 type 2-Port
Service Function Virtual Card UUID 151F94FB-F538-4602-ABAB-6B554D4396D7
```

In the absence of the above, hidden **show emctrl vdu list** command can be used. This command will provide the list of the VDUs with the associated UUIDs synchronized with the EM.

```
[local]UltraM# show emctrl vdu list
Thursday November 30 10:27:15 EST 2017
Showing emctrl vdu
card[01]: name[CFC_01] uuid[DC8069B2-0164-446A-B189-2377C8F5FEFA]
```

```
card[02]: name[SFC_02] uuid[3193B425-1FEB-4C1B-AAD9-7B7961CC411D]
card[03]: name[SFC_03] uuid[3020A5AB-F0B7-475A-BE78-EE82E2B66DD1]
card[04]: name[SFC_04] uuid[FB526BEB-924F-4E1B-A97D-9A7FC5C6BE13]
```

### Step 3:

Identify the VM in OpenStack that correlates with the SF/CF UUID

Once the UUID of the affected VM has been identified, proceed to identify the status of the VM on the OpenStack level.

After sourcing the appropriate tenant keystone rc file, check the VMs associated with CF/SF:

```
[stack@ospd]$ source corerc
[stack@ospd]$ nova list | egrep "(_c|_s)" | sort

| 3193b425-1feb-4c1b-aad9-7b7961cc411d | vnf1-deployment_c1_0_8246a5a6-77b7-484d-
8fe5-227505425213 | ACTIVE | - | Running | orchestr=172.16.180.17;
vnf1-deployment-di-internal1=192.168.1.12; vnf1-deployment-di-
internal2=192.168.2.14; mgmt=172.16.181.24
|
| dc8069b2-0164-446a-b189-2377c8f5fefa | vnf1-deployment_c4_0_867689f4-c6e4-48ef-
9a4f-08b9d18b3f39 | ACTIVE | - | Running | orchestr=172.16.180.19;
vnf1-deployment-di-internal1=192.168.1.3; vnf1-deployment-di-internal2=192.168.2.9;
mgmt=172.16.181.18
|
| 3020a5ab-f0b7-475a-be78-ee82e2b66dd1 | vnf1-deployment_s2_0_9a869c33-a817-4b99-
b9f5-d496190433b3 | ACTIVE | - | Running | vnf1-deployment-di-
internal2=192.168.2.10; orchestr=172.16.180.16; vnf1-deployment-di-
internal1=192.168.1.10; vnf1-deployment-service-network1=10.10.10.4, 10.10.10.12;
vnf1-deployment-service-network2=20.20.20.6, 20.20.20.3 |
| fb526beb-924f-4e1b-a97d-9a7fc5c6be13 | vnf1-deployment_s3_0_16af9038-16a4-476a-
8fba-0ab6e915032c | ACTIVE | - | Running | vnf1-deployment-di-
internal2=192.168.2.4; orchestr=172.16.180.20; vnf1-deployment-di-
internal1=192.168.1.5; vnf1-deployment-service-network1=10.10.10.5, 10.10.10.8;
vnf1-deployment-service-network2=20.20.20.7, 20.20.20.13 |
```

If the UUID identified on StarOS is listed in step 2, check the state of the VM.

Possible states are: ACTIVE / ERROR / SHUTOFF / POWERING-OFF

Lastly, if the UUID identified is not listed above, check if there is a different UUID in ERROR that could correlate to the affected card.

**Step 4:**

Check the VM uptime/compute/virsh instance id

If the VM is identified based on the UUID, more details can be retrieved through **nova show** command. Important are:

- compute which hosts the VM
- virsh instance name
- launch/create/updated time

```
[stack@ospd-ultram-1 custom-templates]$ nova show 3020a5ab-f0b7-475a-be78-ee82e2b66dd1 | grep OS-EXT-SRV-ATTR
| OS-EXT-SRV-ATTR:host | ultram-osd-compute-2.localdomain
| OS-EXT-SRV-ATTR:hostname | vnfdd1-deployment-s2-0-9a869c33-a817-4b99-b9f5-d496190433b3
| OS-EXT-SRV-ATTR:hypervisor_hostname | ultram-osd-compute-2.localdomain
| OS-EXT-SRV-ATTR:instance_name | instance-00000050
| OS-EXT-SRV-ATTR:kernel_id |
| OS-EXT-SRV-ATTR:launch_index | 0
| OS-EXT-SRV-ATTR:ramdisk_id |
| OS-EXT-SRV-ATTR:reservation_id | r-h4089hyk
| OS-EXT-SRV-ATTR:root_device_name | /dev/vda
| OS-EXT-SRV-ATTR:user_data | -

[stack@bru-ospd-ultram-1 custom-templates]$ nova show 3020a5ab-f0b7-475a-be78-ee82e2b66dd1 | grep 2017
| OS-SRV-USG:launched_at | 2017-11-30T11:08:27.000000
```

```
| created | 2017-11-30T11:08:14Z
|
| updated | 2017-11-30T11:20:50Z
```

The updated time will change whenever there was OpenStack activity on the VM. for example: through ESC recovery, through Nova reboot, etc.

If there was no activity (for example, the VM got rebooted only from within StarOS/application), the updated time will not change.

### Step 5:

Collecting console/debug logs from compute.

At this point, we know:

- The VM was ACTIVE the whole time. No state change on OpenStack.
- The associated card did not yet recover in StarOS: It's either rebooting, or removed on StarOS **show card table**

Login to the compute identified in the above step.

First check the provisioning IP by sourcing the stackrc file:

```
[stack@ospd-ultram-1 ~]$ source stackrc
[stack@ospd-ultram-1 ~]$ nova list
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| ID | Name | Status | Task State |
+-----+-----+-----+-----+-----+
| d55b33e6-8d55-4f75-a7b9-72cc5275044c | ultram-compute-0 | ACTIVE | - |
Running | ctlplane=192.200.0.103 |
+-----+-----+-----+-----+-----+
| d5315cb3-8dfa-469f-a19d-7736bdd406b7 | ultram-compute-1 | ACTIVE | - |
Running | ctlplane=192.200.0.105 |
+-----+-----+-----+-----+-----+
| 2ad4de49-c733-4d60-97b2-aa0e84e6ce66 | ultram-controller-0 | ACTIVE | - |
Running | ctlplane=192.200.0.102 |
+-----+-----+-----+-----+-----+
| fb5960a6-b127-45be-8ed6-cb14136f0c99 | ultram-osd-compute-0 | ACTIVE | - |
```

```
Running      | ctlplane=192.200.0.110 |
| 3d577eb6-b321-4f24-87e2-38087b6aab79 | ultram-osd-compute-1 | ACTIVE | - |
Running      | ctlplane=192.200.0.111 |
| 7e946f1f-1405-432f-bf9c-1aa79f751537 | ultram-osd-compute-2 | ACTIVE | - |
Running      | ctlplane=192.200.0.109 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
The stack user has predefined ssh keys which allow it to login to this provisioning
IP:
[stack@ospd-ultram-1 ~]$ ssh heat-admin@192.200.0.109
Last login: Thu Nov 30 14:24:44 2017 from 192.200.0.1
[heat-admin@ultram-osd-compute-2 ~]
```

Things to check on this compute:

**uptime** - to verify if there was a recent reboot. There should not have been a recent reboot.

```
[heat-admin@ultram-rcdnlab-compute-1 ~]$ uptime
21:53:41 up 36 days, 23:30,  1 user,  load average: 5.83, 5.42, 5.29
```

- Check Nova console logs of the VM in **/var/lib/nova/instances/<uuid>/console.log**  
These are the regular VM console logs that can also be seen through Horizon GUI.
- Collect virsh serial logs. These are special debug logs from CF/SF which provide additional information during boot. At this stage, we would expect to see in the serial logs why the card keeps rebooting.

If you get a prompt, it means the card has booted fine, and the card should be available in StarOS:

```
[heat-admin@ultram-osd-compute-2 3020a5ab-f0b7-475a-be78-ee82e2b66dd1]$ sudo virsh
list --all
 Id      Name                                     State
-----
 4      instance-00000050                       running

[heat-admin@ultram-osd-compute-2 3020a5ab-f0b7-475a-be78-ee82e2b66dd1]$ sudo virsh
console instance-00000050 serial1
Connected to domain instance-00000050
Escape character is ^]

qvpc-di:card3-cpu0
```

## CF and SF VM Recovery

If the SF or CF VM is not found and is in VM\_ERROR within the ESC Opdata state, please refer to the Chapter: Troubleshooting Section : VNF Recovery

# Data Recovery from Failed CF

This section will explain how to obtain the StarOS core file directly from CEPH when CF VM is in error or failed state. The overview of the steps is as follows:

- Find the CF instance ID and cloud host.
- Obtain the host IP address and get its hmp **info block**.
- Identify the right image ID (typically, will be the one with size 200GB).
- Export CF volume.
- Create a loop device, a block device, and a mount point.
- Mount the image.

## Finding the CF instance ID and cloud host

Login into OpenStack as core user and obtain the VM name and Instance name where CF is running by executing **nova list --fields name,host,instance\_name**

```
nova list --fields name,host,instance_name
```

ID	Name
Host	Instance Name
ba156dfc-a01a-4b73-a552-f60b999da33a	ultram-vnfm1-em_ultram_0_6e14156f-e1ae-4172-9f05-7e5989bf1d75   ultram-rcdnlab-osd-compute-2.localdomain   instance-000000aa
4ea2921c-8760-4554-8ed5-2b538122712f	<b>vnfd1-deployment_c1_0_45358147-dbb6-4b51-9e86-396ce2b54bf5</b>   <b>ultram-rcdnlab-compute-0.localdomain</b>   <b>instance-000000b9</b>
03adf8dc-a6ca-4dee-a735-3ba6c2bf849b	vnfd1-deployment_c2_0_66a61b28-c86e-4213-

```

b24b-7aa1706a83e0 | ultram-rcdnlab-compute-1.localdomain | instance-000000d2 |
| 7dd5709e-4653-46cc-9a7c-2a4076814a11 | vnfd1-deployment_s3_0_e198088e-e82e-46a4-
b32c-8a5adc7c6db2 | ultram-rcdnlab-osd-compute-1.localdomain | instance-000000b0 |
| 594612bc-0493-414b-913d-3afbacc86eabc | vnfd1-deployment_s4_0_23e2307e-114c-4886-
a130-5088a9fd4e5c | ultram-rcdnlab-osd-compute-0.localdomain | instance-000000b3 |
| e17cdf08-0add-43f9-bb01-26ceceec90dd | vnfd1-deployment_s6_0_faaf284f-6dce-4424-
b02c-056e0d846985 | ultram-rcdnlab-osd-compute-2.localdomain | instance-000000bc |
+-----+-----

```

From the above output, note that CF in question is running on compute-0 node instance 000000b9. This information is needed for further steps.

### Obtaining the Compute Node IP and get its hmp info block

**Step 1:**

Cisco VIM Option. If you are troubleshooting the OSP-D based deployment, proceed to step 2.

In Cisco VIM-based deployment, the IP can be obtained by displaying `/openstack-configs/mercury_servers_info`

```

cat openstack-configs/mercury_servers_info

Total nodes: 27

External API IP: 10.84.103.198

Controller nodes: 3

+-----+-----+-----+-----+-----+-----+-----+
-----+
| Server | CIMC | Management | Provision | Tenant |
Storage |

```

-----+					
-----+					
controller-3	192.100.0.4	192.100.0.210	192.100.0.210	192.168.53.14	
192.168.52.14					
controller-2	192.100.0.3	192.100.0.211	192.100.0.211	192.168.53.15	
192.168.52.15					
controller-1	192.100.0.2	192.100.0.212	192.100.0.212	192.168.53.16	
192.168.52.16					
-----+					
-----+					
Compute nodes: 21					
-----+					
-----+					
Server	CIMC	Management	Provision	Tenant	
Storage					
-----+					
-----+					
compute-18	192.100.0.6	192.100.0.201	192.100.0.201	192.168.53.5	
192.168.52.5					
compute-15	192.100.0.22	192.100.0.202	192.100.0.202	192.168.53.6	
192.168.52.6					

```

| osd-compute-1* | 192.100.0.24 | 192.100.0.203 | 192.100.0.203 | 192.168.53.7 |
192.168.52.7 |
|
|
|
|
|
|
|

```

After collecting the IP address for the compute in question, proceed to step 3.

**Step 2:**

Red Hat OSP-D Option

In Red Hat OSP-D based deployment the IP can be obtained by executing **nova list** after sourcing the stackrc file

```

[stack@ultram-ospd ~]$ source stackrc

[stack@ultram-ospd ~]$ nova list

+-----+-----+-----+-----+
-----+-----+-----+-----+
| ID                                     | Name                                     | Status | Task
State | Power State | Networks                                     |
+-----+-----+-----+-----+
-----+-----+-----+-----+
| a3a48584-7eee-4455-a1a1-39e52ba55e23 | UltraM-RCDNlab-compute-0             | ACTIVE | -
| Running      | ctlplane=192.200.0.108 |
|
| 35b6f193-07e6-44a9-82c1-116dded3095c | UltraM-RCDNlab-compute-1             | ACTIVE | -
| Running      | ctlplane=192.200.0.107 |
|
| 83728dd2-1b2a-4204-94a6-b81f4d944eba | UltraM-RCDNlab-controller-0         | ACTIVE | -
| Running      | ctlplane=192.200.0.113 |
|
| 53512dc5-307f-44a3-9fac-03103c3791ad | UltraM-RCDNlab-controller-1         | ACTIVE | -
| Running      | ctlplane=192.200.0.104 |
|
| 702fa698-450f-4acb-8aa6-d87f3d3b070c | UltraM-RCDNlab-controller-2         | ACTIVE | -
| Running      | ctlplane=192.200.0.110 |

```

```
| 4044ab1c-5a96-4402-8fec-971a3bfa5feb | UltraM-RCDNlab-osd-compute-0 | ACTIVE | -
| Running      | ctlplane=192.200.0.105 |
|
| d202b223-7e76-4dba-971f-8d24c158f980 | UltraM-RCDNlab-osd-compute-1 | ACTIVE | -
| Running      | ctlplane=192.200.0.102 |
|
| 5552fad1-7056-4285-b0f7-2c7cfb2a0dd9 | UltraM-RCDNlab-osd-compute-2 | ACTIVE | -
| Running      | ctlplane=192.200.0.106 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```

### Step 3:

Login to the host with **ssh heat-admin@192.200.0.108**

```
ssh heat-admin@192.200.0.108

Last login: Tue Feb  6 02:50:17 2018 from 192.200.0.1

[heat-admin@compute-0 ~]$ sudo su

[root@compute-0 heat-admin]# virsh list --all

Id      Name                                     State
-----
11      instance-0000008f                       running
12      instance-00000095                       running
13      instance-0000009b                       running
14      instance-000000a1                       running
17      instance-000000b9                       running  >> this is VM instance for CF1,
saved from step 1
18      instance-000000a7                       running
```

**Step 4:**

Extract the volume details for the CF HDD by using **virsh dumpxml <ID>** command

```
virsh dumpxml 17 | egrep -i "nova|disk|volume"

<nova:instance xmlns:nova="http://openstack.org/xmlns/libvirt/nova/1.0">

  <nova:package version="14.0.6-2.el7ost"/>

  <nova:name>vnfd1-deployment_c1_0_45358147-dbb6-4b51-9e86-396ce2b54bf5</nova:name>

  <nova:creationTime>2018-01-22 23:31:43</nova:creationTime>

  <nova:flavor name="ultram-vnfm1-control-function">

    <nova:memory>16384</nova:memory>

    <nova:disk>6</nova:disk>

    <nova:swap>0</nova:swap>

    <nova:ephemeral>0</nova:ephemeral>

    <nova:vcpus>8</nova:vcpus>

  </nova:flavor>

  <nova:owner>

    <nova:user uuid="01e7bb20d7c849118eae2a9f19a2550c">core</nova:user>

    <nova:project uuid="c7ba964cf10049cfb3d34ebde5103a06">core</nova:project>

  </nova:owner>

</nova:instance>

<disk type='network' device='disk'>

  <source protocol='rbd' name='volumes/volume-7a41b0d4-3a92-4628-b267-3ff2ec3abde8'>

  </disk>
```

```

<disk type='network' device='disk'>
    <source protocol='rbd' name='volumes/volume-dded7ad0-cc27-458a-923e-
7dd6c441f7d4'>
</disk>

<disk type='network' device='cdrom'>
    <source protocol='rbd' name='vms/4ea2921c-8760-4554-8ed5-
2b538122712f_disk.config'>
</disk>

    <source path='/var/lib/nova/instances/4ea2921c-8760-4554-8ed5-
2b538122712f/console.log' />

    <source path='/var/lib/nova/instances/4ea2921c-8760-4554-8ed5-
2b538122712f/console.log' />

```

The **virsh dumpxml** output provides info from two (2) volumes for CF VM. These are:

- CF boot /flash.
- HD-RAID

While there is no visible indication which of the two identified volumes is /flash and which is /hd-raid, it is expected HD RAID has a bigger volume.

### Step 5:

Verify the writeback cache mode

Before proceeding to the next step, double check that volume cache mode is set to **writeback** (desired mode for the VPC) by utilizing **qemu-monitor-command** facility:

**virsh qemu-monitor-command --hmp instance-0000009b 'info block'**

```

drive-ide0-0-0 (#block138): rbd:volumes/volumes/volume-7a41b0d4-3a92-4628-b267-
3ff2ec3abde8;id=openstack;auth_supported=cephx\;none;mon_host=11.118.0.10\;6789\
;11.118.0.11\;6789\;11.118.0.12\;6789 (raw)

```

```
Cache mode: writeback
```

```
drive-ide0-0-1 (#block395): rbd:volumes/volume-dded7ad0-cc27-458a-923e-7dd6c441f7d4:id=openstack:auth_supported=cephx\none:mon_host=11.118.0.10\:6789\none:mon_host=11.118.0.11\:6789\none:mon_host=11.118.0.12\:6789 (raw)
```

```
Cache mode: writeback
```

```
drive-ide0-1-0 (#block555): rbd:vms/46f85f69-1204-4b93-8970-9cf5a7043fc2_disk.config:id=openstack:auth_supported=cephx\none:mon_host=11.118.0.10\:6789\none:mon_host=11.118.0.11\:6789\none:mon_host=11.118.0.12\:6789 (raw, read-only)
```

```
Removable device: not locked, tray closed
```

```
Cache mode: writeback
```

## Identifying the right image ID

Confirm the size of the volume by executing the following command on the compute node:

```

rbd info volumes/volume-dded7ad0-cc27-458a-923e-7dd6c441f7d4

  rbd image 'volume-dded7ad0-cc27-458a-923e-7dd6c441f7d4':

    size 200 GB in 51200 objects ==> we can see that this volume is 200GB in
size

    order 22 (4096 kB objects)

    block_name_prefix: rbd_data.7fb556e76350d

    format: 2

    features: layering, exclusive-lock, object-map, fast-diff, deep-flatten

    flags:

```

## Exporting CF Volume

The CF volume can be exported to any compute node where the Raid0 will be created and content reviewed. The MDADM utility will be used to create the Raid0 to add this volume. MDAM is a Linux utility used to manage and monitor software RAID devices.

```
rbd export --image volumes/volume-dded7ad0-cc27-458a-923e-7dd6c441f7d4  
volumes/volume-dded7ad0-cc27-458a-923e-7dd6c441f7d4 /tmp/vnf2-cf1.img
```

```
Exporting image: 100% complete...done.
```

## Creating a loop device, a block device, and a mount point

In this step, dummy raid-1 is created using mdtools in the same host compute and the contents of the exported volume in the host are listed.

If the volume is not corrupted, the contents of the volume can be read/viewed.

### Step 1:

Verify the kernel md space

```
[root@compute]# cat /proc/mdstat  
Personalities :  
unused devices: <none>
```

### Step 2:

Create loop0 device with exported Volume Image (step 4)

```
[root@compute] # losetup /dev/loop0 /tmp/vnf2-cf1.img
```

### Step 3:

Verify the kernel md space

```
[root@compute] # cat /proc/mdstat
Personalities :
md127 : inactive loop0[0](S)
        209715132 blocks super 1.1
unused devices: <none>
```

### Step 4:

Stop the mdadm process

```
[root@compute] # mdadm --stop /dev/md127
mdadm: stopped /dev/md127
```

### Step 5:

Verify again no inactive devices created are present

```
[root@compute] # cat /proc/mdstat
Personalities :
unused devices: <none>
```

### Step 6:

Start the Raid-1 (one Volume only)

```
[root@compute] # mdadm -A --run /dev/md0 /dev/loop0
mdadm: /dev/md0 has been started with 1 drive (out of 2).
```

## Mounting the Disk

In this final step, create the directory for the disk and mount it:

```
[root@compute] # mkdir /mnt/md0raid  
[root@compute] # mount /dev/md0 /mnt/md0raid
```

## Verifying

If the volume is not corrupted by listing the contents of the mounted volume, the original data from failed CF VM can be seen.

```
[root@compute] # cd /mnt/md0raid/  
[root@compute] # ls -l  
total 0  
drwxr-xr-x. 7 root root 111 Feb 5 23:14 data  
drwxr-xr-x. 7 root root 63 Jan 30 13:36 meta  
drwxr-xr-x. 2 root root 6 Jan 30 13:36 temp<  
  
[root@compute] # ls -l data  
total 8512  
drwxrwxr-x. 2 root root 4096 Jan 30 13:44 ahm  
drwxrwx---. 4 root root 26 Jan 30 13:45 confd_dir  
-rw-rw-r--. 1 root root 8710947 Feb 5 23:14 crash-01-00-5a78e560-core.gz  
drwxr-xr-x. 2 root root 6 Jan 30 13:36 databases  
drwxr-xr-x. 9 root root 84 Jan 30 13:36 records  
drwxrwxr-x. 3 root root 20 Jan 30 14:45 support
```

Finally, we can access or copy the core file to the desired destination and perform further checks as needed.

```
[root@compute] # file data/crash-01-00-5a78e560-core.gz  
data/crash-01-00-5a78e560-core.gz: gzip compressed data, from Unix, last modified:  
Mon Feb 5 23:14:40 2018, max speed
```

After all needed files have been transferred, unmount the `/dev/md0`

```
umount /dev/md0
```

```
verify with "df -h" /dev/md0 device is removed
```

# HD RAID Issues

HD RAID can get degraded due to different issues. In this section, troubleshooting steps for two different scenarios are provided.

## RAID Degraded due to different valid image

### Problem Statement

RAID is in degraded state with the remote disk containing a different valid image.

```
[local]UGP# show hd raid verbose
HD RAID:
  State                : Available (clean)
  Degraded             : No
  UUID                 : 1af9c6dd:698ba377:f8c3b631:905d6d4f
  Size                 : 34GB (34000000000 bytes)
  Action               : Idle
  Disk                 : hd-locall
    State              : In-sync component
    Created            : Thu Mar 15 18:05:40 2018
    Updated           : Fri Mar 30 13:59:04 2018
    Events            : 393
    Model              : ATA QEMU HARDDISK 2.5+
    Serial Number     : 0950c19a-f596-459e-b
    Location           : CFC1 6EF5BF1C-25D8-4C7E-862C-3E87A2371848
    Size              : 34.3GB (34359738368 bytes)
  Disk                 : hd-remotel
    State              : Valid image of 954baf46:1198a1bb:c085fdd9:0a56c169
    Created            : Thu Mar 15 18:05:40 2018
    Updated           : Fri Mar 30 13:59:04 2018
    Events            : 393
    Model              : ATA QEMU HARDDISK 2.5+
    Serial Number     : 3826c387-1aec-4c22-b
    Location           : CFC2 CAF9E049-8739-483F-9C18-9840EE23EA8E
    Size              : 34.3GB (34359738368 bytes)
```

**Root Cause:**

When CF1 and CF2 have two different valid images, meaning two different RAID UUIDs, RAID will be in degraded state after every switchover until manually overwritten to form a single RAID with both disks.

So whenever CF changes state from standby to active, it forms RAID only with its local disk.

The remote disk could not be added since it has a different valid image compared to local disk, and config **no overwrite valid disk** is configured which stops overwriting any valid image (This is a recommended config to avoid data loss).

**Solution:**

With this state, a manual overwrite can be used to bring the remote disk into RAID.

Please use the below exec CLI to do this.

```
hd raid overwrite remotel
```

Once the above CLI is issued, the remote disk becomes a spare component and starts rebuilding RAID.

## Disks not detected by the CF cards

**Problem Statement:**

The disk was not recognized by Active CF and this stopped it from forming HD RAID.

```
[local]VPC-DI# show hd raid
Tuesday February 20 19:10:28 UTC 2018
HD RAID:

State                : Not available
[local]VPC-DI#
```

**Root Cause:**

Make sure Cinder service is up and running, Ceph is in the healthy state.

```
[root@ultram-controller-0 heat-admin]# pcs status | grep cinder
openstack-cinder-volume          (systemd:openstack-cinder-volume):      Started
ultram-controller-0

[heat-admin@ultram-controller-0 ~]$ ceph -s
2018-03-30 18:09:40.276835 7f8c12ea3700 -1 auth: unable to find a keyring on
/etc/ceph/ceph.client.admin.keyring,/etc/ceph/ceph.keyring,/etc/ceph/keyring,/
etc/ceph/keyring.bin: (2) No such file or directory
2018-03-30 18:09:40.276847 7f8c12ea3700 -1 monclient(hunting): ERROR: missing
keyring, cannot use cephx for authentication
2018-03-30 18:09:40.276848 7f8c12ea3700 0 librados: client.admin initialization
error (2) No such file or directory
Error connecting to cluster: ObjectNotFound
[heat-admin@ultram-controller-0 ~]$ sudo ceph -s
cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_OK
```

Verify the NTP is sync across the cluster. Following command can be used to quickly verify the NTP synchronization

```
[stack@ospd-]$ for i in $(nova list | grep ACTIVE | awk '{print $12}' | sed
's\ctlplane=\g' ) ; do ssh heat-admin@${i} sudo ntpstat | grep sync ; done
synchronised to NTP server (173.38.201.67) at stratum 2
synchronised to NTP server (173.38.201.67) at stratum 2
synchronised to NTP server (173.38.201.67) at stratum 2
```

**Solution:**

NTP out-of-sync issue can be fixed by restarting the NTPD process using the **sudo systemctl restart ntpd** command on the affected node.

After NTP is synchronized, re-scan the Hard Disk and restart the hdctrl task

```
debug hdctrl restart -rescan_local_disk

debug hdctrl restart
```

**Note** To monitor the status of the RAID resync process, **debug hdctrl mdstat** internal command can be used.

# DIMM Failures

One of the most commonly seen issues in Servers is memory failure. Memory failure can cause a VM to go to a PAUSE or ERROR state. This section will focus on important steps for troubleshooting and recovering from DIMM failures.

## Problem Statement

Should the DIMM failure happen, the following alarm will be reported via CIMC:

DIMM 7 is inoperable : Check or replace DIMM

There are two types of DIMM failures: correctable and non-correctable.

The SEL log will contain the record of the failure.

```
UCS# scope sel
UCS /sel # show entries
Time                Severity           Description
-----
2018-03-28 16:52:14 Non-Recoverable "DDR4_P1_C1_ECC: Memory sensor, non-recoverable
event, Upper Non-Recoverable going high (Non-Correctable ECC occurred on this DIMM)
was asserted"
2018-03-28 16:52:09 Critical           "System Software event: Memory sensor,
Uncorrectable ECC error, DIMM socket 1, Channel C, Processor Socket 1. was asserted"
2018-03-28 16:49:16 Informational "DDR4_P1_C1_ECC: Memory sensor, 2 new correctable
ECC errors"
```

The following guide can be used to identify whether the hardware replacement is needed. <https://www.cisco.com/c/dam/en/us/products/collateral/servers-unified-computing/ucs-manager/whitepaper-c11-736116.pdf>

In case that hardware replacement is needed, following sections will provide a detailed step-by-step procedure to be followed.

Before proceeding, verify Ultra M is in a healthy state and has a recent backup of OSP-D and impacted VMs (ESC, auto-deploy, etc).

## DIMM Replacement in OSD Compute

### Step 1:

#### Graceful VM Shutdown

First, determine which VMs are running on the impacted OSD compute server. In the sample below the DIMM on the server hosting `osd-compute-1` will be replaced

**Note** Both ESCs should never be on the same OSD compute server.

```
[stack@ospd ~]$ nova list --field name,host |grep osd-compute-1
| 34cd92ff-04ba-4da7-886a-3281a94c813c | vnf1-autovnf-uas-1
| ultram-osd-compute-1.localdomain |
| 1f2817a8-9d49-443b-b61b-3ee4e2bce75c | vnf1-vnfm-ESC-0
| ultram-osd-compute-1.localdomain |
| e268c6b8-f45d-4e0c-8ae0-3846662be461 | vnf1-vnfd-deploy_c1_0_1e637e17-6fed-45b0-af5e-5772733618b0 | ultram-osd-compute-1.localdomain |
| d611f3dd-e3df-457d-bfd5-9e98b7c634a8 | vnf1-vnfd-deploy_vnf1-v_0_c8e79899-fe30-460d-b455-baf3da91cec3 | ultram-osd-compute-1.localdomain |
```

In case that ESC Master is hosted on the OSD compute, before proceeding, perform the switchover of the ESC so that the Master ESC moves to a different host, and request to stop server `vnf1-vnfm-ESC-0`:

```
[stack@ospd ~]$ source corerc
[stack@ospd ~]$ nova stop vnf1-vnfm-ESC-0
```

Check previous Standby ESC that it has assumed Master role:

```
[admin@vnf1-vnfm-esc-1 ~]$ escadm status
0 ESC status=0 ESC Master Healthy
```

For all ESC-managed VMs (CF, EM) hosted on the affected OSD compute, shut down the VMs from the ESC-Master using `vm-action` option:

```
[admin@vnf1-vnfm-esc-1 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli vm-action STOP
<VM name>
```

For any remaining VMs (ESC, autovnf-uas, auto-deploy or auto-it), shutdown using `nova stop` from OSP-D:

```
[stack@ospd ~]$ source corerc
[stack@ospd ~]$ nova stop vnf1-autovnf-uas-1
Request to stop server vnf1-autovnf-uas-1 has been accepted
```

### Step 2:

Verify all VMs are SHUTOFF

Verify all the VMs hosted on the affected OSD Compute are in SHUTOFF state from the OSP-D:

```
[stack@ospd ~]$ nova list | grep vnf1-autovnf-uas-1
| 34cd92ff-04ba-4da7-886a-3281a94c813c | vnf1-autovnf-uas-1 | SHUTOFF | - |
Shutdown | vnf1-autovnf-uas-management=172.17.181.9; vnf1-autovnf-uas-
orchestration=172.17.180.9
```

### Step 3:

Put the CEPH into maintenance mode

Login to a controller or the OSD compute server and put CEPH into maintenance mode.

```
[heat-admin@ultram-controller-0 ~]$ sudo ceph osd set norebalance
set norebalance
[heat-admin@ultram-controller-0 ~]$ sudo ceph osd set noout
set noout
[heat-admin@ultram-controller-0 ~]$ sudo ceph status
  cluster eb2bb192-b1c9-11e6-9205-525400330666
    health HEALTH_WARN
```

```

noout,norebalance,sortbitwise,require_jewel_osds flag(s) set
monmap e1: 3 mons at {ultram-controller-0=11.118.0.40:6789/0,ultram-controller-
1=11.118.0.41:6789/0,ultram-controller-2=11.118.0.42:6789/0}
election epoch 58, quorum 0,1,2 ultram-controller-0,ultram-controller-
1,ultram-controller-2
osdmap e194: 12 osds: 12 up, 12 in
flags noout,norebalance,sortbitwise,require_jewel_osds
pgmap v584865: 704 pgs, 6 pools, 531 GB data, 344 kobjects
1585 GB used, 11808 GB / 13393 GB avail
704 active+clean
client io 463 kB/s rd, 14903 kB/s wr, 263 op/s rd, 542 op/s wr

```

#### Step 4:

##### Shutoff Server and Perform HW Maintenance

Power off the specified UCS server and proceed with the DIMM replacement. For the details of how to replace the hardware, refer to the CPU Replacement Procedure section in the Official Cisco UCS Server Installation and Service Guide.

**Note** UCS provides locator LED functionality that can help Field Engineers identify the affected server quickly.

#### Step 5:

##### Power On Server

After the DIMM card has been replaced, power on the server and verify that the server comes up.

```

[stack@ospd ~]$ source stackrc
[stack@ospd ~]$ nova list |grep osd-compute-1
| e35ef010-b6c5-4b2a-b245-7d3af6242bb1 | ultram-osd-compute-1 | ACTIVE | - | Running
| ctlplane=192.200.0.102 |

```

**Step 6:**

## VM Recovery

Start ESC-controlled VMs (EM, CF) from ESC Master:

```
[admin@vnf1-vnfm-esc-1 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli vm-action
START <VM name>
```

Start all remaining VMs from OSP-D.

```
[stack@ospd ~]$ nova start vnf1-autovnf-uas-1

Request to start server vnf1-autovnf-uas-1 has been accepted.
```

**Step 7:**

## VM Verification and unsetting CEPH from Maintenance

Check that all VMs are ACTIVE with **nova list --field name,status,host |grep osd-compute-1**:

```
[stack@ospd ~]$ source corerc
[stack@ospd ~]$ nova list --field name,status,host |grep osd-compute-1
| 34cd92ff-04ba-4da7-88 | vnf1-autovnf-uas-1 | ACTIVE | ultram-osd-compute-
1.localdomain |
| 1f2817a8-9d49-443b-b61b-3ee4e2bce75c | vnf1-vnfm-ESC-1
| ACTIVE | ultram-osd-compute-1.localdomain |
| e268c6b8-f45d-4e0c-8ae0-3846662be461 | vnf1-vnfd-deploy_c1_0_1e637e17-6fed-45b0-
af5e-5772733618b0 | ACTIVE | ultram-osd-compute-1.localdomain |
| d611f3dd-e3df-457d-bfd5-9e98b7c634a8 | vnf1-vnfd-deploy_vnf1-v_0_c8e79899-fe30-
460d-b455-baf3da91cec3 | ACTIVE | ultram-osd-compute-1.localdomain |
```

Login to a controller or OSD compute and remove CEPH from maintenance mode.

```
[heat-admin@ultram-controller-0 ~]$ sudo ceph osd unset norebalance
unset norebalance
[heat-admin@ultram-controller-0 ~]$ sudo ceph osd unset noout
unset noout
```

Verify the CEPH status is HEALTH\_OK

```
[heat-admin@ultram-controller-0 ~]$ sudo ceph status
cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_OK
monmap e1: 3 mons at {ultram-controller-0=11.118.0.40:6789/0,ultram-controller-1=11.118.0.41:6789/0,ultram-controller-2=11.118.0.42:6789/0}
election epoch 58, quorum 0,1,2 ultram-controller-0,ultram-controller-1,ultram-controller-2
osdmap e196: 12 osds: 12 up, 12 in
flags sortbitwise,require_jewel_osds
pgmap v584954: 704 pgs, 6 pools, 531 GB data, 344 kobjects
1585 GB used, 11808 GB / 13393 GB avail
704 active+clean
client io 12888 kB/s wr, 0 op/s rd, 81 op/s wr
```

## DIMM Replacement on Compute Node

### Step 1:

#### Graceful VM Shutdown

Identify if there is active SF on the affected compute node. If so, before shutting down the SF VM, migrate the SF to the standby state. Please refer to the "Troubleshooting Overview" chapter to identify which SF VM is running on this compute node.

```
[local]UGP# card migrate from 10 to 6
Sunday December 03 23:46:33 UTC 2017
Are you sure? [Yes|No]: Yes
Sunday December 03 23:46:36 UTC 2017
```

Session recovery will show migration progress.

```
[local]UGP# show session recovery status
Sunday December 03 23:46:45 UTC 2017
Session Recovery Status:
  Overall Status      : Migration in Progress
  Last Status Update  : 15 seconds ago
```

The migration has finished once the session recovery status is set as Ready for Recovery.

```
[local]UGP# show session recovery status
Sunday December 03 23:48:48 UTC 2017
Session Recovery Status:
  Overall Status      : Ready For Recovery
  Last Status Update  : 8 seconds ago
```

The SF will now show as Standby.

```
[local]UGP# show card table
Sunday December 03 23:49:19 UTC 2017
```

Slot	Card Type	Oper State	SPOF	Attach
1: CFC	Control Function Virtual Card	Active	No	
2: CFC	Control Function Virtual Card	Standby	-	
3: FC	4-Port Service Function Virtual Card	Active	No	
4: FC	4-Port Service Function Virtual Card	Active	No	
5: FC	4-Port Service Function Virtual Card	Active	No	
6: FC	4-Port Service Function Virtual Card	Active	No	
7: FC	4-Port Service Function Virtual Card	Active	No	
8: FC	4-Port Service Function Virtual Card	Active	No	
9: FC	4-Port Service Function Virtual Card	Active	No	
10: FC	4-Port Service Function Virtual Card	Standby	-	

Lastly, shut down the SF VM on the compute node from the ESC Master using vm-action.

```
[admin@vnf1-esc-esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli vm-action STOP
vnf1-DEPLOYMENT-_s5_0_e93ed083-b60e-4085-8bb4-63147b14c87c
VM Action
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin --
```

```
privKeyFile=/home/admin/.ssh/confd_id_dsa --privKeyType=dsa --
rpc=/tmp/esc_nc_cli.1zHd2SN01c
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok/>
</rpc-reply>
```

Verify VM is in the SHUTOFF state before continuing.

```
[stack@ospd ~]$ nova list |grep -i 58d9bcc1-741a-4da7-bca2-6c5337bead36
| 58d9bcc1-741a-4da7-bca2-6c5337bead36 | vnf1-DEPLOYMENT-_s5_0_e93ed083-b60e-4085-
8bb4-63147b14c87c | SHUTOFF | - | Shutdown | vnf1-UAS-uas-
orchestration=172.168.11.12; service1=192.168.12.39, 192.168.12.51; di-
internal2=192.168.11.24; di-internal1=192.168.10.20; service2=192.168.13.43,
192.168.13.51 |
```

### Step 2:

Shutoff Server and Perform HW Maintenance

Power off the specified UCS server and proceed with the DIMM replacement. For the details of how to replace the hardware, refer to CPU Replacement Procedure section in the Official Cisco UCS Server Installation and Service Guide.

**Note** UCS provides locator LED functionality that can help Field Engineers identify the affected server quickly.

### Step 3:

Power On Server

Power on the server and verify the server comes up. Check the UCS memory status is healthy:

```
[stack@ospd ~]$ source stackrc
[stack@ospd ~]$ nova list |grep compute-7
| 1021eae3-76a5-4a52-bff4-a94fdc8d25a7 | ultram-compute-7 | ACTIVE | - |
Running | ctlplane=192.200.0.116 |
```

### Step 4:

Start ESC controlled SF VM from ESC Master

Start SF VM on the compute node from the ESC Master using vm-action:

```
[admin@vnf1-esc-esc-0 ~]$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli vm-action START
vnf1-DEPLOYMENT-_s5_0_e93ed083-b60e-4085-8bb4-63147b14c87c

/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin --
privKeyFile=/home/admin/.ssh/confd_id_dsa --privKeyType=dsa --
rpc=/tmp/esc_nc_cli.mdGeUihlpN
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok/>
</rpc-reply>
```

Verify VM is up before continuing:

```
[stack@ospd ~]$ nova list |grep -i 58d9bcc1-741a-4da7-bca2-6c5337bead36
| 58d9bcc1-741a-4da7-bca2-6c5337bead36 | vnf1-DEPLOYMENT-_s5_0_e93ed083-b60e-4085-
8bb4-63147b14c87c | ACTIVE | - | Running | vnf1-UAS-uas-
orchestration=172.168.11.12; service1=192.168.12.39, 192.168.12.51; di-
internal2=192.168.11.24; di-internal1=192.168.10.20; service2=192.168.13.43,
192.168.13.51 |
```

### Step 5:

Verification

Login to VMs on compute and verify the card comes up as Standby and Session Recovery returns to Ready For Recovery:

```
[local]UGP# show card table
Monday December 04 00:10:59 UTC 2017
Slot          Card Type                                     Oper State   SPOF  Attach
-----
1: CFC        Control Function Virtual Card                 Active       No
2: CFC        Control Function Virtual Card                 Standby      -
3: FC         4-Port Service Function Virtual Card         Active       No
4: FC         4-Port Service Function Virtual Card         Active       No
```

5:	FC	4-Port Service Function Virtual Card	Active	No
6:	FC	4-Port Service Function Virtual Card	Active	No
7:	FC	4-Port Service Function Virtual Card	Active	No
8:	FC	4-Port Service Function Virtual Card	Active	No
9:	FC	4-Port Service Function Virtual Card	Active	No
10:	FC	4-Port Service Function Virtual Card	Standby	-

The session recovery status shall be set back to ready for recovery:

```
[local]UGP# show session recovery status
Monday December 04 00:11:03 UTC 2017
Session Recovery Status:
  Overall Status      : Ready For Recovery
  Last Status Update  : 3 seconds ago
```

## DIMM Replacement in Controller

### Prerequisites

From OSP-D, login to the controller and verify pcs is in a good state - all three controllers should be online and Galera showing all three controllers as Master.

A healthy cluster requires 2 active controllers so verify that the remaining two controllers are online and active.

```
[heat-admin@ultram-controller-0 ~]$ sudo pcs status
Cluster name: tripleo_cluster
Stack: corosync
Current DC: ultram-controller-2 (version 1.1.15-11.e17_3.4-e174ec8) - partition with quorum
Last updated: Mon Dec 4 00:46:10 2017 Last change: Wed Nov 29 01:20:52 2017 by hacluster via crmd on ultram-controller-0
3 nodes and 22 resources configured

Online: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]

Full list of resources:

ip-11.118.0.42 (ocf::heartbeat:IPaddr2): Started ultram-controller-1
```

```

ip-11.119.0.47 (ocf::heartbeat:IPaddr2): Started ultram-controller-2
ip-11.120.0.49 (ocf::heartbeat:IPaddr2): Started ultram-controller-1
ip-192.200.0.102 (ocf::heartbeat:IPaddr2): Started ultram-controller-2
Clone Set: haproxy-clone [haproxy]
    Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: galera-master [galera]
    Masters: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
ip-11.120.0.47 (ocf::heartbeat:IPaddr2): Started ultram-controller-2
Clone Set: rabbitmq-clone [rabbitmq]
    Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: redis-master [redis]
    Masters: [ ultram-controller-2 ]
    Slaves: [ ultram-controller-0 ultram-controller-1 ]
ip-10.84.123.35 (ocf::heartbeat:IPaddr2): Started ultram-controller-1
openstack-cinder-volume (systemd:openstack-cinder-volume): Started ultram-
controller-2
my-ipmilan-for-controller-0 (stonith:fence_ipmilan): Started ultram-controller-0
my-ipmilan-for-controller-1 (stonith:fence_ipmilan): Started ultram-controller-0
my-ipmilan-for-controller-2 (stonith:fence_ipmilan): Started ultram-controller-0

Daemon Status:
    corosync: active/enabled
    pacemaker: active/enabled
    pcsd: active/enabled

```

### Step 1:

Move controller cluster to maintenance mode

Put the PCS cluster on the controller to be updated into standby:

```
[heat-admin@ultram-controller-0 ~]$ sudo pcs cluster standby
```

Check **pcs status** again and make sure the pcs cluster stopped on this node. The **pcs status** on the other 2 controllers should show the node as standby.

```

[heat-admin@ultram-controller-0 ~]$ sudo pcs status
Cluster name: tripleo_cluster
Stack: corosync
Current DC: ultram-controller-2 (version 1.1.15-11.e17_3.4-e174ec8) - partition with
quorum
Last updated: Mon Dec  4 00:48:24 2017 Last change: Mon Dec  4 00:48:18 2017 by root

```

```

via crm_attribute on ultram-controller-0

3 nodes and 22 resources configured<

Node ultram-controller-0: standby
Online: [ ultram-controller-1 ultram-controller-2 ]

Full list of resources:

ip-11.118.0.42 (ocf::heartbeat:IPaddr2): Started ultram-controller-1
ip-11.119.0.47 (ocf::heartbeat:IPaddr2): Started ultram-controller-2
ip-11.120.0.49 (ocf::heartbeat:IPaddr2): Started ultram-controller-1
ip-192.200.0.102 (ocf::heartbeat:IPaddr2): Started ultram-controller-2
Clone Set: haproxy-clone [haproxy]
  Started: [ ultram-controller-1 ultram-controller-2 ]
  Stopped: [ ultram-controller-0 ]
Master/Slave Set: galera-master [galera]
  Masters: [ ultram-controller-1 ultram-controller-2 ]
  Slaves: [ ultram-controller-0 ]
ip-11.120.0.47 (ocf::heartbeat:IPaddr2): Started ultram-controller-2
Clone Set: rabbitmq-clone [rabbitmq]
  Started: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
Master/Slave Set: redis-master [redis]
  Masters: [ ultram-controller-2 ]
  Slaves: [ ultram-controller-1 ]
  Stopped: [ ultram-controller-0 ]
ip-10.84.123.35 (ocf::heartbeat:IPaddr2): Started ultram-controller-1
openstack-cinder-volume (systemd:openstack-cinder-volume): Started ultram-
controller-2
my-ipmilan-for-controller-0 (stonith:fence_ipmilan): Started ultram-controller-1
my-ipmilan-for-controller-1 (stonith:fence_ipmilan): Started ultram-controller-1
my-ipmilan-for-controller-2 (stonith:fence_ipmilan): Started ultram-controller-2

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled

```

**Step 2:**

## Shutoff Server and Perform HW Maintenance

Power off the specified server and proceed with the DIMM replacement. For the details about how to replace the hardware, refer to CPU Replacement Procedure section in the Official Cisco UCS Server Installation and Service Guide.

**Note** UCS provides locator led functionality that can help Field Engineers identify the affected server quickly.

**Step 3:**

Power on the server and verify the server comes up into ACTIVE.

```
[stack@ospd ~]$ source stackrc
[stack@ospd ~]$ nova list |grep controller-0
| 1ca946b8-52e5-4add-b94c-4d4b8a15a975 | ultram-controller-0 | ACTIVE | - |
Running | ctlplane=192.200.0.112 |
```

**Step 4:**

Resume PCS status to normal mode

Login to the impacted controller, remove standby mode by setting 'unstandby'.

```
[heat-admin@ultram-controller-0 ~]$ sudo pcs cluster unstandby
```

Verify the controller comes Online with cluster and Galera shows all three controllers as Master. This may take a few minutes.

```
[heat-admin@ultram-controller-0 ~]$ sudo pcs status
Cluster name: tripleo_cluster
Stack: corosync
Current DC: ultram-controller-2 (version 1.1.15-11.e17_3.4-e174ec8) - partition with
quorum
```

```
Last updated: Mon Dec 4 01:08:10 2017 Last change: Mon Dec 4 01:04:21 2017 by root
via crm_attribute on ultram-controller-0
```

```
3 nodes and 22 resources configured
```

```
Online: [ ultram-controller-0 ultram-controller-1 ultram-controller-2 ]
```

## Step 5:

### Post Replacement Health Check

Verify the CEPH is in the HEALTH\_OK state

```
[heat-admin@ultram-controller-0 ~]$ sudo ceph -s
cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_OK
monmap e1: 3 mons at {ultram-controller-0=11.118.0.10:6789/0,ultram-controller-
1=11.118.0.11:6789/0,ultram-controller-2=11.118.0.12:6789/0}
election epoch 70, quorum 0,1,2 ultram-controller-0,ultram-controller-
1,ultram-controller-2
osdmap e218: 12 osds: 12 up, 12 in
flags sortbitwise,require_jewel_osds
pgmap v2080888: 704 pgs, 6 pools, 714 GB data, 237 kobjects
2142 GB used, 11251 GB / 13393 GB avail
704 active+clean
client io 11797 kB/s wr, 0 op/s rd, 57 op/s wr
```

Verify that ironic node did not go into the maintenance mode. You can use the ironic node-list command on OSP-D to verify the status. If in maintenance node, perform the recovery.

```
[stack@ospd-ultram-1 ~]$ ironic node-list
+-----+-----+-----+
+-----+-----+-----+
| UUID                               | Name | Instance UUID |
| Power State | Provisioning State | Maintenance |
+-----+-----+-----+
+-----+-----+-----+
| fcf359f8-b2dd-4279-affc-ee5dbf3c50e7 | None | 7ecdc5af-f629-4d72-bae4-0cc6d8641482 |
| None          | active          | True      |
+-----+-----+-----+
[stack@ospd-ultram-1 ~]$ ironic node-set-maintenance fcf359f8-b2dd-4279-affc-
```

ee5dbf3c50e7 off

```
[stack@ospd-ultram-1 ~]$ ironic node-list
```

```

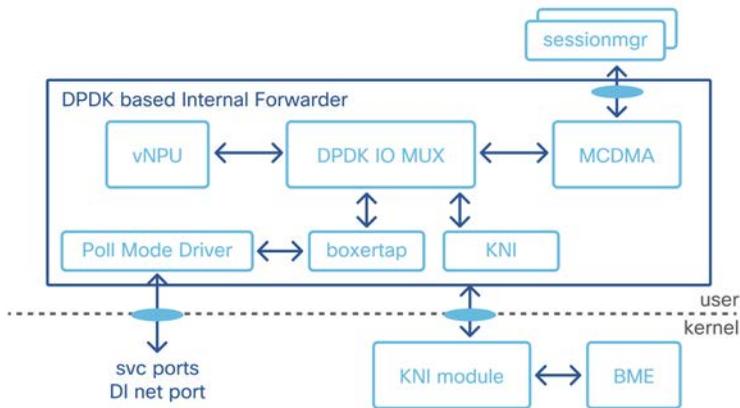
+-----+-----+-----+
| UUID                                     | Name | Instance UUID |
| Power State | Provisioning State | Maintenance |
+-----+-----+-----+
| fcf359f8-b2dd-4279-affc-ee5dbf3c50e7 | None | 7ecdc5af-f629-4d72-bae4-0cc6d8641482 |
| None          | active          | False      |

```

# IFTASK Tuning Considerations

IFTASK is a multi-thread process consisting of main thread, worker threads and monitoring threads. The worker threads are classified as vNPU and MCDMA threads. vNPU threads handle the interaction with the network while MCDMA threads are used for inter-process communication.

The diagram below shows how the various StarOS Internal subsystems interact with DPDK and IFTASK.



Poll Mode Driver (PMD) works directly with the network and interacts with DPDK I/O Multiplexer. If the packet is for a known process for which a flow exists, it will be sent via the MCDMA channel to the appropriate application layer. MCDMA channels are one of the interprocess communication mechanisms used in StarOS. If there are any flows that need to be sent to another VM, they get passed via an internal forwarder to vNPU. Any flow that cannot be handled by PMD falls back to the slow path which is the Kernel NIC Interface (KNI) driver.

When traffic is operating under normal conditions, the **show npumgr utilization info** command is used to show traffic distribution. Here is a sample output when there is uneven distribution of traffic:

```
***** show npumgr utilization information 3/0/0 *****
5-Sec Avg: lcore01| lcore02| lcore03| lcore04| lcore05| lcore06| lcore07| lcore08| lcore09| lcore10| lcore11| lcore12|
Idle: 17%| 33%| 53%| 61%| 38%| 40%| 58%| 69%| 28%| 48%| 35%| 68%|
PortRX: 32%| 49%| 0%| 0%| 0%| 0%| 42%| 31%| 0%| 0%| 0%| 32%|
PortTX: 21%| 18%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%|
KnIRX: 3%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%|
McdmaRX: 0%| 0%| 20%| 18%| 27%| 27%| 0%| 0%| 35%| 25%| 31%| 0%|
Mcdma: 0%| 0%| 2%| 2%| 3%| 3%| 0%| 0%| 7%| 4%| 5%| 0%|
McdmaFlush: 0%| 0%| 25%| 19%| 32%| 30%| 0%| 0%| 30%| 23%| 29%| 0%|
Cipher: 27%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%| 0%|
rx kbits/sec: 292650| 392695| 415491| 269531| 552922| 415016| 665545| 661409| 682682| 429160| 696150| 647119|
rx frames/sec: 50381| 49019| 57708| 41169| 78852| 59744| 108205| 83305| 95242| 61830| 105807| 83902|
tx kbits/sec: 285507| 385621| 423795| 275453| 564216| 423613| 649926| 649383| 696355| 438052| 711376| 635011|
tx frames/sec: 50368| 49005| 57708| 41169| 78852| 59744| 108179| 83277| 95242| 61830| 105807| 83878|
```

The traffic pattern is not uniform, with some cores being more utilized (lcore01) and some cores less utilized (lcore08).

Each of these cores are VCPUs that are allocated to different application usage. In the example above, lcore01, lcore02, lcore07, lcore08 are used for PMD and lcore03, lcore04, lcore05, lcore06, lcore09, lcore10, lcore11 and lcore12 are used for MCDMA channels. PMD is used to interface with the NIC driver to communicate with the external world via IFTASK proctlet. In order to change the allocations, consider how many cores are used by the StarOS SF VM. Then typically allocate between 30% to 50% of cores for use by IFTASK proctlet (total of MCDMA and PMD). Port Speed, Di Network capacity/utilization and the traffic pattern together determine the allocations.

Another useful command to use to determine the network performance is **show iftask stats**.

```
***** show iftask stats *****
Thursday December 14 20:18:15 UTC 2017
CARD 3 STATS
-----
Counters                SF3                SF3_PPS
-----
svc_rx                   282945967          90
svc_tx                   270816001          86
di_rx                    6169859            0
di_tx                    4808575            0
__ALL_DROPS__           5                  0
```

svc_tx_drops	0	0
di_rx_drops	0	0
di_tx_drops	0	0
sw_rss_enq_drops	0	0
kni_thread_drops	0	0
kni_drops	5	0
mcdma_drops	0	0
mux_deliver_hop_drops	0	0
mux_deliver_drops	0	0
mux_xmit_failure_drops	0	0
mc_dma_thread_enq_drops	0	0
sw_tx_egress_enq_drops	0	0
cpeth0_drops	0	0
mcdma_summary_drops	0	0
fragmentation_err	0	0
reassembly_err	0	0
reassembly_ring_enq_err	0	0
__DISCARDS__	39242	0
CARD 4 STATS		
-----		
Counters	SF4	SF4_PPS
-----		
.....		
TOTAL STATS		
-----		
Counters	TOTAL	TOTAL_PPS
-----		
svc_rx	2182482694	719
svc_tx	2101284573	691
di_rx	47104198	4
di_tx	41661796	3
__ALL_DROPS__	34	0
svc_tx_drops	0	0
di_rx_drops	0	0
di_tx_drops	0	0
sw_rss_enq_drops	0	0
kni_thread_drops	0	0
kni_drops	34	0
mcdma_drops	0	0
mux_deliver_hop_drops	0	0
mux_deliver_drops	0	0
mux_xmit_failure_drops	0	0
mc_dma_thread_enq_drops	0	0

```

sw_tx_egress_enq_drops          0          0
cpeth0_drops                    0          0
mcdma_summary_drops             0          0
fragmentation_err               0          0
reassembly_err                  0          0
reassembly_ring_enq_err         0          0
__DISCARDS__                    35715     0
-----
NDR is      100.0000
CONTINUE_TRAFFIC
-----
-----
    
```

Once the NDR in the above output goes below 98, then examine the reasons for discards and packet drops. Based on this, increase the allocation of cores to iftask by changing the config in AutoVNF file and redeploy.

```

vdu vdu-sf1
vdu-type session-function
image url none
flavor vcpus 30
flavor ram 86016
flavor root-disk 16
flavor ephemeral-disk 0
flavor swap-disk 0
upp cores 40
upp crypto-cores 0
upp service-mode vpc
upp disable-mcdma false
upp disable- numa false
upp param DI_INTERFACE_VLANID
value 2145
    
```

In the above configuration example, the number of VCPUs used by SF is 30. Of these, 40% or 12 cores are allocated to IFTASK. The decision to increase the IFTASK cores has implications for tuning the rest of the allocation between MCDMA and PMD cores. The type of traffic influences the allocation between MCDMA and PMD cores. The following are the lines necessary to alter the allocation of cores to MCDMA and PMD.

```

upp param IFTASK_MCDMA_CORES
value 50
    
```

From the previous example of 12 IFTASK cores, 50% or 6 are used for MCDMA and 6 are used by PMD which gives uniform allocation ratio for non-iftask cores (18) and MCDMA cores (6). The number of non-iftask cores should always be a whole number multiple of the number of MCDMA cores allocations. Determining that ratio is very complex and traffic patterns such as User Plane Cipher traffic, NIC utilization and congestion on Di Network all should be considered in determining these values. The MCDMA channels should be balanced between the MCDMA cores across the NUMA nodes. Based on the amount of VNPU and MCDMA packet processing needed, pick the appropriate combination.

# Acronyms

# Acronyms

AFD - Advanced Fault Detection

AutoIT-VNF - Provides storage and management for system ISOs.

CDB - Configuration Datastore

CF - Control Function

CIMC - Cisco Integrated Management Controller

CIMC/IPMI - Cisco Integrated Management Controller / Intelligent Platform Management Interface

Cisco VIM - Cisco Virtualized Infrastructure Manager

DI - Distributed Instance

DPDK - Data Plane Development Kit

DRBD - Distributed Replicated Block Device

EM - Element Manager

EMCTR - Element Manager Control Task

ESC - Elastic Service Controller

ETSI - European Telecommunications Standards Institute

HA mode - High-Availability mode

HD RAID - Hard Drive Redundant Array of Independent Disks

IFTASK - Internal Forwarder

MCDMA - Multi-Channel Direct Memory Access

MDADM - administers or manages multiple RAID devices

MLOM - Modular LAN on Motherboard

MPC - Mobile Packet Core

NETCONF - Network Configuration Protocol

NFV - Network Function Virtualization

NFG - Network Function Group

NFVI - Network Function Virtualization Infrastructure

NFVO - Network Functions Virtualization Orchestrator

NFV-MANO - Network Functions Virtualization Management and Orchestration

NS - Network Service

NSD - Network Service Descriptor

OOB - Out-of-Band

OSC - OpenStack Controller	UAS - Ultra Automation Services
OSP-D - OpenStack Platform Director	UCS - Unified Computing System
OSD - Object Storage Daemon (in this book referred to as the integrated Nova and Ceph service)	UEM - Ultra Element Manager
OVS - Open vSwitch	UGP - Ultra Gateway Platform
PNF - Physical Network Function	USP - Ultra Service Platform
PNFD - PNF Descriptor	VDU - Virtual Deployment Unit
PXE - Preboot eXecution Environment	VIM - Virtualized Infrastructure Manager
REST - REpresentational State Transfer	VM - Virtual Machine
RPM - Red Hat Package Manager	VNF - Virtual Network Function
RSS - Receive Side Scaling	VNFD - VNF Descriptor
SDN - Software Defined Networking	VNFM - Virtual Network Function Manager
SESSMGR - Session Manager	VPC - Virtualized Packet Core.
SF - Service Function	VPC-DI - Virtual Packet Core Distributed Instance
SI - (VPC) Single Instance	VPC-SI - Virtual Packet Core Single Instance
SR-IOV - Single Root I/O Virtualization	YANG - Yet Another Next Generation

Dave Damerjian  
Dennis Lanov  
Jeff Williams  
Rama Ramachandran  
Robert West  
Roshan Warriar  
Snezana Mitrovic  
Solomon Ayankulankara Kunjan  
Sourav Jyoti Das

