Management and Network Orchestration (MANO) Inter working and Feasibility Report

by
5G-MANO Team

Table of Contents

Table of Figures

Index of Tables

1 Introduction

This document contains the report on the tests and study done on the inter working of OSM and other MANO modules.

2 The Network Service and Slice for Testing

To Test the network service on MANO platform the first service/slice was introduced with two Virtual Network Functions (VNFs). If the same network service contains the two VNFs then we say it network service with two VNFs. But if it is implemented in slice then it will contain two network services containing one network each.

2.1 Service/Slice with two VNFs

This network service or network slice will contain two VNFs. One VNF is a apache web server and the other VNF is the database server. The client opens a browser and sends a URL request to the apache web server for a CGI page. The CGI script at apache web server will fetch the data of a table from the database server. The data is finally displayed on the browser of the client.

In the first step the virtual box images (VDI) for the data base server and apache web server were created

2.1.1 VDI Image Creation

The images were created for apache web server and db server as described below.

2.1.1.1 VDI Image Creation for apache web server

Following steps were performed for the vdi image creation of apache web server.

1. Fetch the OSI image of ubuntu 18.04. from ubuntu mirror. The file name for image is ubuntu-18.04.4-desktop-amd64.iso.
2. Create a ubuntu 18.04 virtual machine through virtual box with the following specifications
   - name => webserver
   - CPU => 1
   - RAM => 4096 MB
   - Hard Disk => 20GB
3. Provide administrative user name as '*webserver*'
4. run command "*sudo apt-get update*" to update the package information
5. run command "*sudo apt-get upgrade*" to upgrade the packages
6. run command "*sudo apt-get install vim*" to install the vim editor
7. run command "*sudo apt-get install openssh-server*" to install the open ssh server
8. run command "*sudo apt-get install mariadb-client*" to install the mariadb DB client

9. run command "*sudo apt-get install apache2*" to install the apache web server
10. run command "*sudo apt-get install curl*" to install the curl
11. After running these commands the CGI is to be configured.
12. Check the CGI script path in "*examples/apache/serve-cgi-bin.conf*" file. In this setup it was "*/usr/lib/cgi-bin/*"
13. Go to the directory "*/etc/apache2/mods-available*" and check for "*cgi.load file*". It should be present there. In our case it was present.
14. Run command "cd /etc/apache2/mods-enabled" and check whether the link "*cgi.load*" exists or not. In our case it was not there.
15. Run command " *sudo ln -s ../mods-available/cgi.load* " to create the link.
16. Run command " *sudo service apache2 reload* " to reload the apache web service.
17. Create a file "/usr/lib/cgi-bin/NSTest.cgi" and put the contents in the file as shown in the following table

```
#!/bin/bash
echo "Content-type: text/html"
echo ""
echo "<html><head><title>CDOT Employees List"
echo "</title></head><body>"
dbhostname=196.1.110.72
echo "<h1>List of CDOT Employees</h1>"
echo "list of CDOT employees as on  $(date)"
echo "<br><br>"
echo "<table  border=\"1\">"
mysql -h ${dbhostname} -u admin -ppassword -D MANO -e "select
CONCAT(\"<tr><td>\",STAFFNO, \"</td>\") as \"<tr><td>STAFF
NO</td>\",CONCAT(\"<td>\",NAME, \"</td>\") as \"<td>NAME</td>\",
CONCAT(\"<td>\",CDOTGROUP, \"</td></tr>\") as \"<td>GROUP</td></tr>\" from
EMPLOYEECDOT"
echo "</table>"
echo "<br>"
echo "data has been fetched from @<b>"${dbhostname}"<b>"
echo "</body></html>"
```

18. Create a file "/usr/bin/change-dbserver-ip" and put the contents in the file as shown in the following table

```
sudo sed -i 's/^dbhostname=.*/dbhostname='${1}'/' /usr/lib/cgi-bin/NSTest.cgi
```

19. Run command " *sudo chmod +x /usr/bin/change-dbserver-ip*"

20. Now we can run command "*change-dbserver-ip <DB Server IP >*" whenever we want to change the dbserver IP in the CGI script.
21. The URL for the CGI page which will be accessed through client will be "***http://<webserverIP>/cgi-bin/NSTest.cgi*** "
22. The VDI image will be available at the " *<dault Machine Folder of Virtual Box>/webserver/webserver.vdi* ".
23. In our case the path for the virtual image was

    " */home/jugalk/VirtualBox Vms/webserver/webserver.vdi* "

After creation of the web server image the image of db server is created

2.1.1.2 VDI Image Creation for mariadb server

Following steps were performed for the vdi image creation of apache web server.

1. Fetch the OSI image of ubuntu 18.04. from ubuntu mirror. The file name for image is ubuntu-18.04.4-desktop-amd64.iso.
2. Create a ubuntu 18.04 virtual machine through virtual box with the following specifications
   - name => dbserver
   - CPU => 1
   - RAM => 4096 MB
   - Hard Disk => 20GB
3. Provide administrative user name as 'd*bserver*'
4. run command "*sudo apt-get update*" to update the package information
5. run command "*sudo apt-get upgrade*" to upgrade the packages
6. run command "*sudo apt-get install vim*" to install the vim editor
7. run command "*sudo apt-get install openssh-server*" to install the open ssh server
8. run command "*sudo apt-get install mariadb-server*" to install the mariadb DB server
9. run command "*sudo apt-get install mariadb-client*" to install the mariadb DB client
10. run command "sudo mysql" to go into the mysql server
11. run the following commands shown in the table on mariadb prompt to create and authorize mysql user.

```
> CREATE USER 'admin'@'localhost' IDENTIFIED BY 'password';
> GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;
> CREATE USER 'admin'@'%' IDENTIFIED BY 'password';
> GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%' WITH GRANT OPTION;
> FLUSH PRIVILEGES;
```

12. Re-login to the mysql with user name ‘admin’ and password as ‘password’ and run the following commands, as shown in table below, on mariadb prompt to create and update the database and table.

```
> CREATE TABLE EMPLOYEECDOT (STAFFNO INT(11) NOT NULL , NAME VARCHAR (100)
NOT NULL, CDOTGROUP VARCHAR(100), PRIMARY KEY (STAFFNO));
> INSERT INTO EMPLOYEECDOT (STAFFNO, NAME, CDOTGROUP) VALUES ( '3525', 'Jugal
Kishore Sharma', '5G-MANO')
….. similarly insert more rows to be displayed …….
```

13. The VDI image will be available at the “ *<dault Machine Folder of Virtual Box>/webserver/dbserver.vdi ”*.

14. In our case the path for the virtual image was

    “ */home/jugalk/VirtualBox Vms/dbserver/dbserver.vdi ”*

2.1.2 Virtual Network Function Descriptors (VNFDs) for OSM

The two network function descriptors were created one for apache web server and the other for the mariadb db server. The descriptors were written in the yaml format and for the ETSI Open Source MANO (OSM) platform. The user guise of osm can be referenced for the understanding of the descriptors [1].

2.1.2.1 Virtual Network Function Descriptor(VNFD) for apache web server

Following table described the VNFD for apache web server. The descriptor shown here is in yaml format. The tag “*image*” contains value “*webserver*” here. But this may be different in different VNFI environment depending on the image actually present there.

```
vnfd:vnfd-catalog:
  vnfd:
  - id: webserver-vnf
    name: webserver-vnf
    short-name: webserver-vnf
    version: '1.0'
    description: A
    logo: osm.png
    connection-point:
    - name: vnf-cp0
      type: VPORT
    vdu:
```

```
  - id: webserver-VM
    name: webserver-VM
    image: 'webserver'
    count: 1
    vm-flavor:
      vcpu-count: 2
      memory-mb: 4096
      storage-gb: 20
    interface:
    - name: vdu-eth0
      type: EXTERNAL
      virtual-interface:
        type: PARAVIRT
      external-connection-point-ref: vnf-cp0
  mgmt-interface:
    cp: vnf-cp0
```

2.1.2.2 Virtual Network Function Descriptor(VNFD) for mariadb db server

Following table described the VNFD for mariadb db server. The descriptor shown here is in yaml format. The tag "*image*" contains value "d*bserver*" here. But this may be different in different VNFI environment depending on the image actually present there.

```
vnfd:vnfd-catalog:
  vnfd:
  - id: dbserver-vnf
    name: dbserver-vnf
    short-name: dbserver-vnf
    version: '1.0'
    description: A controller VNF descriptor with one VDU for configuring physical devices
    logo: osm.png
    connection-point:
    - name: vnf-cp0
      type: VPORT
    vdu:
    - id: dbserver-VM
      name: dbserver-VM
      image: dbserver
      count: '1'
      vm-flavor:
        vcpu-count: '2'
        memory-mb: '4096'
        storage-gb: '20'
      interface:
      - name: vdu-eth0
        type: EXTERNAL
        virtual-interface:
          type: PARAVIRT
        external-connection-point-ref: vnf-cp0
    mgmt-interface:
      cp: vnf-cp0
```

2.1.3 Network service Descriptors (NSDs) for OSM

Two network service descriptors were introduced containing one VNF each. The service descriptors were written in yaml format. These were written for the OSM platform. Following sections describe these network service descriptors. The user guise of osm can be referenced for the understanding of the descriptors [1].

2.1.3.1 Network Service Descriptor(NSD) for apache web service

This network service has one virtual network function of web server and a link to external network. Following table contains the network service descriptor. The network name here is "*mgmtnet"* which can be different for different NFVIs.

```
nsd:nsd-catalog:
   nsd:
   -  id: webserver-ns_nsd
      name: webserver-ns_nsd
      short-name: webserver-ns_nsd
      description:  NS for WebServer
      version: '1.0'
      logo: osm.png
      connection-point:
      -  name: nsd_cp_mgmt
         vld-id-ref: mgmtnet
      constituent-vnfd:
      -  vnfd-id-ref: webserver-vnf
         member-vnf-index: '1'
      vld:
      -  id: mgmtnet
         name: mgmtnet
         short-name: mgmtnet
         type: ELAN
         mgmt-network: 'true'
         vnfd-connection-point-ref:
         -  vnfd-id-ref: webserver-vnf
            member-vnf-index-ref: '1'
            vnfd-connection-point-ref: vnf-cp0
```

2.1.3.2 Network Service Descriptor (NSD) for mariadb db service

This network service has one virtual network function of db server and a link to external network. Following table contains the network service descriptor. The network name here is "public*"* which can be different for different NFVIs.

```
vnfd:vnfd-catalog:
   vnfd:
   -  id: dbserver-vnf
      name: dbserver-vnf
      short-name: dbserver-vnf
      version: '1.0'
      description: A controller VNF descriptor with one VDU for configuring physical devices
      logo: osm.png
      connection-point:
```

```
    - name: vnf-cp0
      type: VPORT
    vdu:
    - id: dbserver-VM
      name: dbserver-VM
      image: dbserver
      count: '1'
      vm-flavor:
        vcpu-count: '2'
        memory-mb: '4096'
        storage-gb: '20'
      interface:
      - name: vdu-eth0
        type: EXTERNAL
        virtual-interface:
          type: PARAVIRT
        external-connection-point-ref: vnf-cp0
    mgmt-interface:
      cp: vnf-cp0
```

2.1.4 Network Slice Descriptors

There were different cases for network slice and accordingly the slice descriptors were written.

Case1: One slice containing both services but on different NFVI

The descriptor in this case is written here in the following table. The NSTest1 and NSTest2 are the NFVI VIM accounts for two different VIMs.

```
nst:
- id: nstest_nst
  name: nstest_nst
  SNSSAI-identifier:
    slice-service-type: eMBB
  quality-of-service:
    id: 2

  netslice-subnet:
  - id: dbserver-ns_nsd
    is-shared-nss: 'false'
    description: Service for DB server
    nsd-ref: dbserver-ns_nsd
    instantiation-parameters:
    - vimAccountId: NSTest2


  - id: webserver-ns_nsd
    is-shared-nss: 'false'
    description: Service for web server
    nsd-ref: webserver-ns_nsd
    instantiation-parameters:
    - vimAccountId: NSTest1

  netslice-vld:
```

```
- id: dbserver_vld_mgmt
  name: dbserver_vld_mgmt
  type: ELAN
  mgmt-network: 'true'
  nss-connection-point-ref:
  - nss-ref: dbserver_nsd_1
    nsd-connection-point-ref: nsd_cp_mgmt

- id: webserver_vld_mgmt
  name: webserver_vld_mgmt
  type: ELAN
  mgmt-network: 'true'
  nss-connection-point-ref:
  - nss-ref: webserver_nsd_1
    nsd-connection-point-ref: nsd_cp_mgmt
```

Case2: One slice containing one service

In this case the network slice descriptor is not required because the single service can be instantiated directly.

3 OSM Feasibility Test with VMWare Cloud

OSM supports the VMware's vCloud Director as VIM [1]. To test the scenario it was required to install the VMware's vCloud Director along with compute nodes.

3.1 Vmware's vCloud Director

The architecture of VMware's vCloud Director is shown in the figure below. The details of the architecture can be seen in the the official document of vmvare [2].

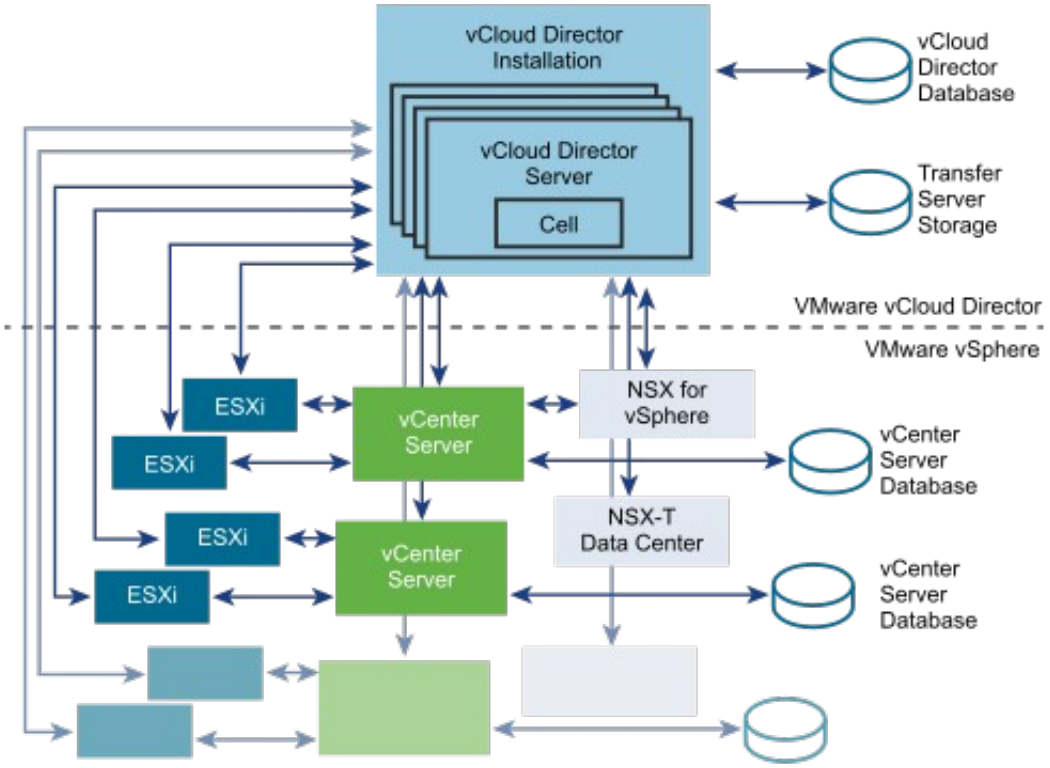

Figure 1: Architecture of VmVare's vCloud Director

The first requirement was to install the ESXi. To install the ESXI following steps were done.

1. The software is the licensed one. So we downloaded the image "Vmware-Vmvisor-Installer-7.0.0-15843807.x86_64.iso" on two months trial version.
2. Prepared the bootable disk (Pen Drive) for this.
3. But the workstation did nor recognized the disk as bootable disk
4. Got CD from the CSG version of version 5.5
5. The installation halted at a point and the screen was blank. No input and output was there. The same result remained in multiple attempts.
6. The compatibility was checked on the internet and it was found that the processor of the workstation (Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz) is not in the compatibility list [4].

7. The compatible servers were not there so the activity was suspended till the compatible resources are available.

3.2 Conclusion

The feasibility of inter operability of OSM and VmVare's vCloud Director could not be checked due to the unavailability of compatible resources to install VMware cloud or NFVI.

4 OSM Feasibility Test with VMware Integrated OpenStack (VIO)

OSM supports the VMware Integrated OpenStack (VIO) [3]. To test the scenario it was required to install the VMware Integrated OpenStack (VIO) along with compute nodes.

4.1 VMware Integrated OpenStack (VIO)

The architecture of VMware Integrated OpenStack (VIO) is shown in figure below. The details can be seen in the document “VMware Integrated OpenStack Installation and Configuration Guide”[3].

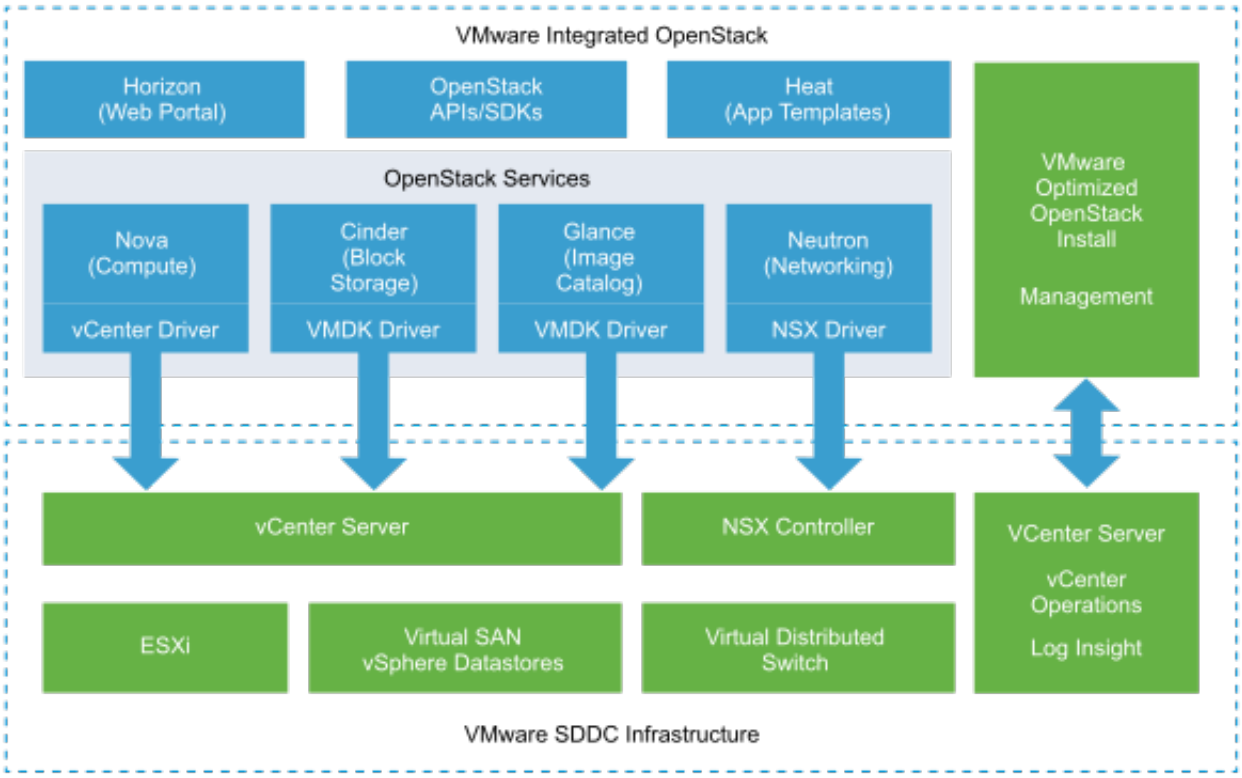

Figure 2: Arcitecture of VMware Integrated OpenStack (VIO)

The basic component in the architecture is the ESXi in VMWare SDDC Infrastructure. But as we have explained in the chapter “The Feasibility Test with VMWare Cloud” that the installation is not possible because of the unavailability of the resources.

4.2 Conclusion

The feasibility of inter operability of OSM and VmVare's vCloud Director could not be checked due to the unavailability of compatible resources to install VMware cloud or NFVI.

5 Feasibility test with OSM when two VNFs are on two different OpenStack VIM/NFVI

The two VNFs webserver and db server are well explained in As explained in chapter 2. The service is also explained in chapter2. There can be following ways to implement the services.

1. Two VNFs running on different NFVIs but are part of same network service.
2. Two services containing one VNF each are part of a single network slice. The services running of different NFVI.
3. The two services containing one VNF each instantiated on different NFVI through same OSM. And the user service involving both web server and db server runs smoothly.

5.1 Common Generic Steps

Following listed are the activities to be performed for the on boarding of the VNFs and Network services

1. Prepare the "*qcow*" image from the "*vdi*" images
2. create project or tenant of openstack on both open stack VIMs
3. allocate resource quota to project or tenant on both open stack VIMs
4. Create non admin user account on both open stack VIMs for the tenant
5. Create the desired Flavours
6. Create Networks and Sub Networks
7. upload the "qcow" images to the clouds (NFVI)
8. Prepare the VNF and NS packages for onboarding
9. Create project and user in OSM
10. create VIM accounts in OSM
11. On boarding the VNFs, NS, and NST in OSM

5.1.1 Prepare "qcow: image from the "vdi" image

It has been already explained that how to create the "*vdi*" images using virtual box for web server and db server, and where to find them in ubuntu. In our case following were the path where these "vdi" images were stored

1. The "*vdi*" image for mariadb db server was stored at -

   " */home/jugalk/VirtualBox Vms/dbserver/dbserver.vdi* "
2. The "*vdi*" image for apache web server was stored at -

   " */home/jugalk/VirtualBox Vms/dbserver/dbserver.vdi* "

Following commands were executed to create the "*qcow*" image. Please note that the commands are there with file name and directory of our case. This directory path may change if the "vdi" files are ekpt at different locations.

1. Run command "*sudo apt-get -y install qemu-kvm libvirt-bin virtinst bridge-utils*" to install the required packages qemu-kvm, libvirt-bin and virtinst bridge-utils.
2. Run command "*cd /home/jugalk/VirtualBox\ Vms/dbserver/*" to go to the directory where "vdi" image for dbserver is stored.
3. Run command "*qemu-img convert -f vdi -O qcow2 dbserver.vdi dbserver.qcow2*" to convert the "*vdi*" image "*dbserver.vdi*" to "*qcow2*" image "*dbserver.qcow2*"
4. Run command "*cd /home/jugalk/VirtualBox\ Vms/webserver/*" to go to the directory where "vdi" image for webserver is stored.
5. Run command "*qemu-img convert -f vdi -O qcow2 webserver.vdi webserver.qcow2*" to convert the "*vdi*" image "web*server.vdi*" to "*qcow2*" image "web*server.qcow2*"

5.1.2 Create project or tenant of openstack on both open stack VIMs

The two different VIMs were configured for this. The user interface "*horizon*" was used to create the project or tenant. The project "*NStest*" was created on both openstack VIMs with the following details. (The details to create project can be seen in open stack user guide). This operation was done through administrative account

Name	NSTest
Domain Name	Default
Domain ID	default
Enabled	Yes
Description	To test the Network Services through OSM

5.1.3 Allocate resource quota to project or tenant on both open stack VIMs

The resource quota was allocated to the tanent NSTest was as under:

Instances	18
VCPUs	20
RAM	50 GB
Floating Ips	50
Security Groups	10
Security Group Rules	100
Networks	100
Ports	500

Routers	10

5.1.4 Create non admin user account on both open stack VIMs

Using the same horizon GUI a non administrative user “osmtest” was created on both VIMs with the following details. This operation was done through an administrative account.

Name	osmtest
Domain Name	Default
Domain ID	default
Description	To test OSM on OpenStack
Email	jugal@cdot.in
Enabled	Yes
Password Expires At	None
Primary Project	NSTest
Role	user

5.1.5 Create the desired Flavors

New flavors for virtual machines were created on both VIMs with following details. This operation was done through administrative account.

On first VIM whose IP address for horizon was 192.168.151.79 following flavor was already present with desired configuration so there was no need to create a new flavor. The details of flavor are:

Name	opensips_flavor
VCPUs	2
RAM	4GB
Root Disk	20GB
Ephemeral Disk	0GB
Swap Disc	0MB
RX/TX Factor	1.0
Public	Yes
Metadata	No

On second VIM whose IP address for horizon was 192.168.151.80 a new flavor with desired configuration was created.

figuration so there was no need to create a new flavor. The details of flavor are:

Name	ubuntu204
VCPUs	2
RAM	4GB
Root Disk	20GB
Ephemeral Disk	0GB
Swap Disc	0MB
RX/TX Factor	1.0
Public	Yes
Metadata	No

5.1.6 Create Networks and Sub-networks

The networks can be created by the non admin user. But in our case we required externally managed networks. These networks were already created in the VIMs and NFVIs. We have used these networks in the experiment. As all the resources were in use we did not try to create a externally managed network through non admin user.

Following are the details of externally managed network used from VIM whose IP address for horizon was 192.168.151.79.

Nerwork Details:-

Name	public
Shared	Yes
External Network	Yes
MTU	1500

Subnet details:-

Name	public-subnet
Network Name	public
Subnet Pool	None
IP Version	IPv4
CIDR	192.168.152.0/24
IP Allocation Pools	Start 192.168.152.32 - End 192.168.152.254
Gateway IP	192.168.152.1
DHCP Enabled	Yes

Additional Routes	None
DNS Name Servers	196.1.105.47

Following are the details of externally managed network used from VIM whose IP address for horizon was 192.168.151.80.

Nerwork Details:-

Name	mgmtnet
Shared	Yes
External Network	Yes
MTU	1500

Subnet details:-

Name	vnf-management-subnet
Network Name	mgmtnet
Subnet Pool	None
IP Version	IPv4
CIDR	192.168.156.0/24
IP Allocation Pools	Start 192.168.156.2 - End 192.168.156.254
Gateway IP	192.168.156.1
DHCP Enabled	Yes
Additional Routes	None
DNS Name Servers	196.1.105.47

5.1.7 upload the “qcow” images to the clouds (NFVI)

This operation was done by the new non administrative user “*osmtest*” created by on both VIMs. The images were loaded on the VIMs with the following details.

Image for webserver :-

Name	dbserver
Owner	NSTest
Type	image
Visibility	Image from Other Project - Non-Public

Protected	no
Disk Format	QCOW2
Size	6.88 GB
Min Disk	20 GB
Min. RAM	4096
Container Format	BARE
Description	webserver

Image for dbserver:-

Name	webserver
Owner	NSTest
Type	image
Visibility	Image from Other Project - Non-Public
Protected	no
Disk Format	QCOW2
Size	8.07GB
Min Disk	20 GB
Min. RAM	4096
Container Format	BARE
Description	webserver

5.1.8 Prepare the VNF and NS packages for onboarding

The yaml files for VNF and NS had already been discussed in chapter 2. Following package and empty directories were created as shown in the table. The directories are shown in red color. All files are empty except the yaml file.

For dbserver vnf	dbserver-vnf |-- README |-- charms |-- checksums.txt |-- cloud_init |-- dbserver-vnf_vnfd.yaml

	|-- icons | `-- osm.png |-- images `-- scripts
For webserver vnf	webserver-vnf |-- README |-- charms |-- checksums.txt |-- cloud_init |-- webserver-vnf_vnfd.yaml |-- icons | `-- osm.png |-- images `-- scripts
For dbserver ns	dbserver-ns |-- README |-- checksums.txt |-- dbserver-ns_nsd.yaml |-- icons | `-- osm.png |-- ns_config |-- scripts `-- vnf_config
For webserver ns	webserver-ns |-- README |-- checksums.txt |-- icons | `-- osm.png |-- ns_config |-- scripts |-- vnf_config `-- webserver-ns_nsd.yaml

Following steps was done to generate packages

1. prepare directories and file as stated in the above table. The yaml file details are given in chapter 2
2. Run command "*git clone https://osm.etsi.org/gerrit/osm/devops.git*" to clone the devops repository of OSM.
3. Install the IM module as per the guidelines [6]
4. go to the directory where the package directories are kept.
5. Run command " *<devops_path>/devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd -N dbserver-vnf*" to generate the dbserver vnf package.

6. Run command " *<devops_path>/devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd -N webserver-vnf*" to generate the weserver vnf package.
7. Run command " *<devops_path>/devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t nsd -N dbserver-ns*" to generate the dbserver ns package.
8. Run command " *<devops_path>/devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t nsd -N webserver-ns*" to generate the webserver ns package.

5.1.9 Create project and user in OSM

A new project and user was created by using the guidelines given in user guide [1]. Following was done.

1. Create project "NSTest"
2. Create use "osmtest"
3. associate user "osmtest" with project "NSTest" with roles as "project_user" and "project_admin"

5.1.10 Create VIM accounts in OSM

The VIM accounts were created after logging in by user "osmtest" with the following details

Details of VIM account NSTest1:-

Name	NSTest1
VIM Username	osmtest
VIM URL	http://mano:5000/v3/
Type	openstack
Tenant name	NSTest
Description	Test for osm NS

Details of VIM Account NSTest2:-

Name	NSTest2
VIM Username	osmtest
VIM URL	http://mano:5000/v3/
Type	openstack
Tenant name	NSTest
Description	Test for osm NS

5.1.11 On boarding of VNFs, NS and NST in OSM

The on boarding of all the VNF and NS packages was successfully done following the guidelines specified in the user guide [1]. The NST package described in chapter2 could not be on boarded and has given error. The problem has been reported on the open source mano slack.

5.2 Two VNFs running on different NFVIs but are part of same network service

To do this the Network Service descriptor should have some mechanism so that we may declare the instantiation VIM of the listed VNF in the NS descriptor. But this is not supported by OSM Information Module. So this case is not feasible to test at present.

5.3 Two services containing one VNF each are part of a single network slice: the services running of different NFVI

To implement this scenario, the network slice template (NST) as defined in chapter 2 case 1 is to be on boarded. But osm did not allow this to be on boarded and gave error. The problem has been reported on the open source mano slack. So this case could not be tested at present.

5.4 Two services containing one VNF each instantiated on different NFVI through same OSM

In this scenario there is no need for building a network slice because one slice containing one network service will no mean too much. The service were instantiated one by one using the OSM NBI. After instantiating the services the IP addresses of the VNFs were recorded. Following steps were done to test.

1. Login to the VM of webserver vnf through ssh
2. Run command "*change-dbserver-ip < ip of dbderver >*" on VM of webserver vnf. In our case the command was "*change-dbserver-ip 192.168.152.227*"
3. open the web browser from client machine outside NFVIs and open URL "http://<ipaddress of webserver>/cgi-bin/NSTest.cgi". In our case the URL was "http://192.168.156.184/cgi-bin/NSTest.cgi"

The output is shown in the following figure:

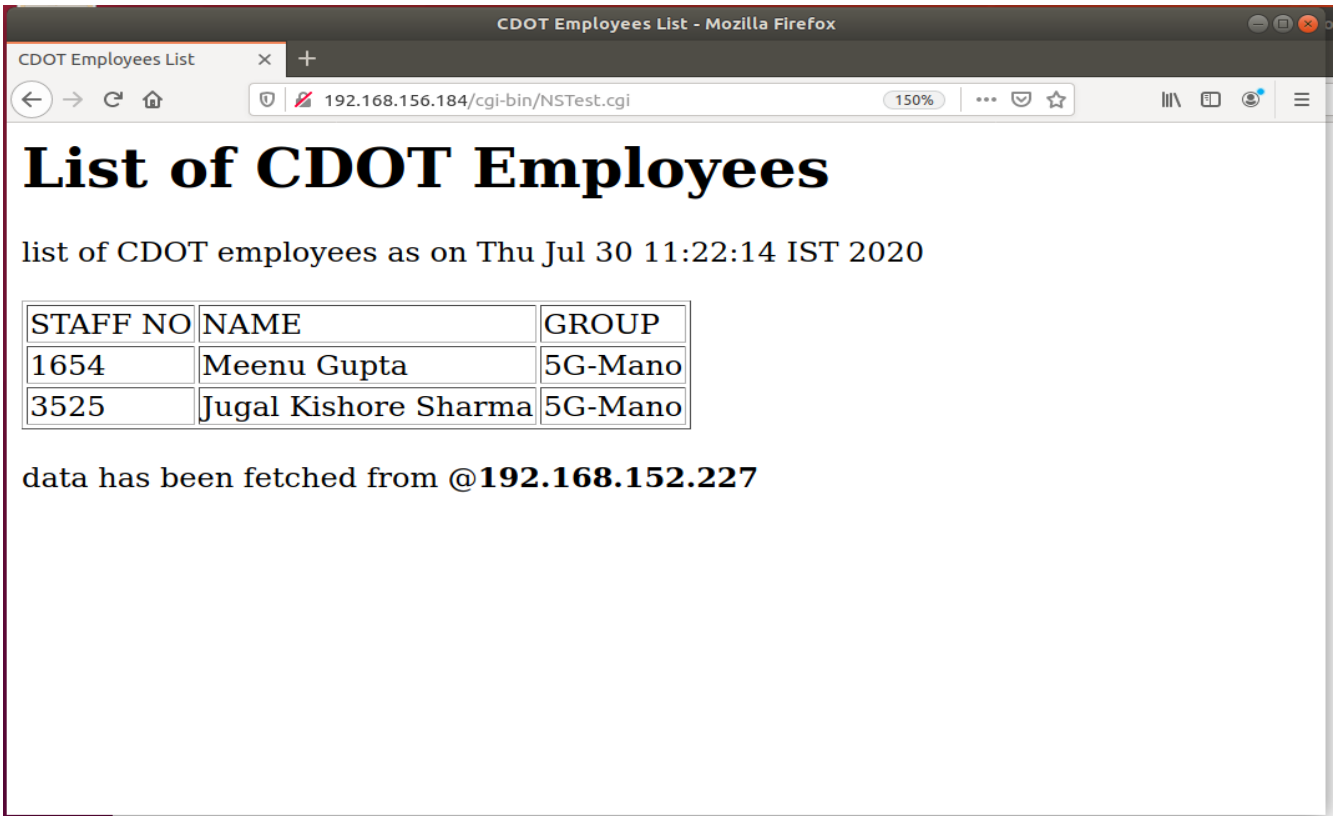

Figure 3: Output of Web Page indicating both VNFs on different NFVIs are working fine

5.5 Conclusion

Following can be concluded regarding openstack VIM

1. The project or tenant account is to be created by administrative user
2. The resource limits to the project are to be assigned by administrative user
3. The and user account is to be created by administrative user.
4. The flavor addition can only be done through administrative user.
5. The image uploading can be done through non administrative user.
6. The non administrative user can add the network in the project but regarding external network the testing is not done due to certain limitations.
7. The non administrative user can instantiate the Virtual Machine
8. The non administrative user can do life cycle management for the virtual machines created in the project associated with the user.
9. The non administrative user can create the security groups and manage rules.

Following can be concluded regarding Open Source Mano

1. Project creation is to be done by administrative user.

2. User creation is to be done by administrative user.
3. The project and role association is to be done by administrative user
4. Multiple VIM account can be created by non administrative user within allowed projects.
5. Package on boarding and deletion can be done by non administrative user within allowed projects.
6. The life cycle management can be done by non administrative user within allowed projects for network services and VNFs.
7. OSM does not support two VNFs belonging to same network service to be instantiated on different VIM/NFVIs.
8. OSM GUI does not allow a network slice to instantiate the network service sub-net on different NFVIs.
9. OSM IM allows to write the instantiation parameter with network service sub-net including the VIM parameters. But the network service template with such details did not get on boarded and throws error. The same has been uploaded on osm slack.

6 OSM feasibility test with FOG O5: Literature Survey

6.1 Eclipse FOG O5

Eclipse fog05 allows the end-to-end management of compute, storage, networking and I/O fabric in the Edge and Fog Environment. Instead of relying on a centralized architecture (like cloud-management-systems) it is based on a decentralized architecture.

Eclipse fog05 allows users to manage and deploy different types of applications, packaged as containers, VMs, binaries and so on. This possibility is achieved by Eclipse fog05 plugin architecture.

More specifically, Eclipse fog05 is a Fog Infrastructure-as-a-Service solution composed by two major components:

- The Fog Orchestration Engine (FOrcE)
- The Fog Infrastructure Manager (FIM)

These components provides the abstractions to deploy your applications in the Fog and Edge environment. [7]

6.2 The Survey

A literature survey was done from the documents available from the internet and is was fond that the Eclipse FOG O5 is OSM compatible. The OSM ETSI user guide [1] section 1.3 states that the FOG O5 is supported by OSM. The section 1.3.1.7 describes in detail the configuration required to add the VIM account for Eclipse FOG O5. Further the section “Eclipse fog05 with ETSI OSM Orchestrator” of the Eclipse FOG O5 getting started guide [8] declares “ *Eclipse fog05 is compatible with ETSI OSM, and can be used as a VIM by this one. Current interoperability is limited to simple NS using LXD containers as VNFs*.”. Further it describes well tested example network service with OSM.

6.3 The Exception

The OSM release 7 GUI does not have the option for Eclipse FOG O5. However we may assume that it may be supported with CLI because in the OSM user guide is is written as supported. The testing with Eclipse FOG O5 could not be done because the Eclipse FOG O5 installation could not be made working when this document was written.

6.4 Conclusion

As per the survey the Eclipse FOG O5 is compatible with the OSM. It has different architecture as compared to openstack and other clouds and is well suited for edge.

7 OSM feasibility test with FOG O5: Eclipse FOG O5 Installation

To test the feasibility with OSM the eclipse FOG O5 was installed by using the following steps with the help of installation steps provided in the getting started guide [7].

7.1 Installation from Debian Packages

Following steps were taken to install Eclipse FOG O5 from Debian packages.

7.1.1 Preparing Environment for Installation

Following steps were taken to prepare environment for installation

1. Install Ubuntu 20.04. If already installed then ignore it. This is because the "***glibc***" version used by Eclipse FOG O5 is supported in by Ubuntu 20.04 or higher.
2. run command "*sudo apt-get update*" to update the package information.
3. run command "*sudo apt-get upgrade*" to upgrade the packages.
4. run command "*sudo apt-get install vim*" to install the vim editor.
5. run command "*sudo apt-get install openssh-server*" to install the open ssh server.
6. run command "*sudo apt-get install git*" to install the git client.
7. run command "*sudo apt install python3-pip*" to install the pip3 for python packages.

7.1.2 Installing the agent

To install the agent we can simply run the following commands

1. run command "*wget https://github.com/eclipse-fog05/fog05/releases/download/v0.2.0/zenoh_0.3.0-1_amd64.deb*" to get the deb package for zenoh server.
2. run command "*sudo apt install ./zenoh_0.3.0-1_amd64.deb*" to install the zenoh server
3. run command "*wget https://github.com/eclipse-fog05/fog05/releases/download/v0.2.0/fog05_0.2.0-1_amd64_ubuntu.bionic.deb*" to get the agent.
4. run command "sudo apt install ./fog05_0.2.0-1_amd64_ubuntu.bionic.deb" to install the agent

It will create the folder "*/etc/fos*" in which we can found the configuration file agent.json and the agent, and configures the systemd service "*fos_agent*" and the Zenoh systemd service "*zenoh*"

7.1.3 Installing the Linux Plugins

To install the linux plugin we can simply run the following commands

1. Run command "*wget https://github.com/eclipse-fog05/fog05/releases/download/v0.2.0/libzenoh-0.3.0-Linux.deb*" to get the zenoh plugin library

2. Run command "*sudo apt install ./libzenoh-0.3.0-Linux.deb*" to install the zenoh plugin library.
3. Run command "*sudo pip3 install fog05-sdk==0.2.0 zenoh ==0.3.0 yaks==0.3.0.post1*" to install the desired python plugins.
4. Run command "*wget https://github.com/eclipse-fog05/fog05/releases/download/v0.2.0/fog05-plugin-os-linux_0.2.0-1_amd64_ubuntu.bionic.deb*" to get the gog o5 plugins.
5. Run command "*sudo apt install ./fog05-plugin-os-linux_0.2.0-1_amd64_ubuntu.bionic.deb*" to install the fog plugins.

After the installation the directory "*/etc/fos/plugins/plugin-os-linux*" is created and the plugin configuration "*/etc/fos/plugins/plugin-os-linux/linux_plugin.json*" is populated with the "*/etc/machine-id*" as "*nodeid*" value. The systemd service "*fos_linux*" is created.

7.1.4 Installing the Linux bridge Plugins

Following steps are taken to install linux bridge plugins

1. Run command wget "https://github.com/eclipse-fog05/fog05/releases/download/v0.2.0/fog05-plugin-net-linuxbridge_0.2.0-1_amd64_ubuntu.bionic.deb" to fetch linux bridge plugins.
2. Run command sudo apt install ./fog05-plugin-net-linuxbridge_0.2.0-1_amd64_ubuntu.bionic.deb to install linux bridge plugins.

After the installation the directory "*/etc/fos/plugins/plugin-net-linuxbridge*" is created and the plugin configuration "*/etc/fos/plugins/plugin-net-linuxbridge/linuxbridge_plugin.json*" is populated with the "*/etc/machine-id*" as "*nodeid*" value. The systemd service "*fos_linuxbridge"* is created.

7.1.5 Installing FDU Plugins

As the test setup the installation was done for LXD plugins. First to install the LXD following steps are required

1. Run command "*sudo snap install lxd*"

To install the LXD plugin following steps are required

1. Run command "*wget https://github.com/eclipse-fog05/fog05/releases/download/v0.2.1/fog05-plugin-fdu-lxd_0.2.1-1_amd64.deb*" to fetch the FDU plugins for LXD.
2. Run command "*sudo apt install ./fog05-plugin-fdu-lxd_0.2.1-1_amd64.deb*" to install the plugins.

After the installation the directory "*/etc/fos/plugins/plugin-fdu-lxd*" is created and the plugin configuration "*/etc/fos/plugins/plugin-fdu-lxd/LXD_plugin.json*" is populated with the "*/etc/machine-id*" as "*nodeid*" value. The systemd service "*fos_lxd*" is created

Note:-

The above command for LXD containers were not sufficient and further configuration was required. At the time of the writing this document the complete configuration could not be done. The following additional configuration was done.

```
$ sudo adduser {your-user-name-here} lxd
$ newgrp lxd
$ id
$ lxc list
$ sudo apt install zfsutils-linux
$ lxc storage create pool1 zfs
$ lxc storage list { Following output will be displayed }

+---------+-------------------+-----------+-----------------------------------------------------+-------------+
| NAME  | DESCRIPTION | DRIVER |                SOURCE                                | USED BY |
+---------+-------------------+-----------+-----------------------------------------------------+-------------+
| pool1-- |                       |    zfs       | /var/snap/lxd/common/lxd/disks/pool1.img  | 0              |
+-------+--------------------+-----------+-----------------------------------------------------+-------------+
$ lxc profile device add default root disk path=/ pool=pool1
```

Now as per guidelines the following services should start by giving the following commands

```
$ sudo systemctl start zenoh
$ sudo systemctl start fos_agent
$ sudo systemctl start fos_linux
$ sudo systemctl start fos_linuxbridge
$ sudo systemctl start fos_lxd
```

But in our case only three services could start. To make the services running following was done

1. Run command "*wget https://files.pythonhosted.org/packages/88/90/54293958e770a6881ed18b293e077b36908550b61f23b08e95736ee3c32d/zenoh-0.3.0.tar.gz*" to get zenoh source of required version.
2. Expend the archive by giving command "*tar -xvf zenoh-0.3.0.tar.gz*"
3. Go to the source directory by giving command "*cd zenoh-0.3.0/*"
4. Install the python APIs by giving command "*sudo python3 setup.py install*"

7.1.6 Installation to run Hello world basic program

Run following commands to install containerd and the containerd plugin for Eclipse fog05.

```
$ wget https://github.com/eclipse-fog05/fog05/releases/download/v0.2.0/fog05-plugin-fdu-containerd_0.2.0-1_amd64_ubuntu.bionic.deb
$ sudo apt install ./fog05-plugin-fdu-containerd_0.2.0-1_amd64_ubuntu.bionic.deb
```

The site "https://eclipse-fog05-python3-client-api.readthedocs.io/en/latest/installation.html" contains the guidelines for the installation of client APIs. Following commands were given to install the same.

```
$ git clone https://github.com/eclipse-fog05/sdk-python
$ cd sdk-python
$ make
```

```
$ sudo make install
$ cd ../
$ git clone https://github.com/eclipse-fog05/api-python
$ cd api-python
$ sudo make install
```

The helloworld was tries as per the instructions given in FOG O5 guide [7] but it could not run because the node was not created. At this stage this research was discontinued.

8 Abbreviations

BSS	Business Support Systems
CLI	Command Line Interface
DWDM	Dense Wavelength Division Multiplexing
EMS	Element Management System
GPON	Gigabit Passive Optical Network
ILA	Inline Amplifier
LoRa	Low Range
LTE	Long Term Evolution
MANO	Management and Orchestration
NFVI	Network Functions Virtualization Infrastructure
OLT	Optical Line Terminal
ONT	Optical Network Terminal
OSM	Open Source MANO
OSS	Operations Support Systems
PNF	Physical Network Function
PoC	Proof of Concept
PON	Passive Optical Network
ROADM	Reconfigurable Optical Add-Drop Multiplexer
SDN	Software Defined Network
SNI	Service Node Interface
SNMP	Simple Network Management Protocol
VIM	Virtual Infrastructure Manager
VNF	Virtual Network Function

9 REFERENCES

[1]. OSM user guide "https://osm.etsi.org/docs/user-guide/" click here .

[2]. vCloud Director Installation, Configuration, and Upgrade Guide. "https://docs.vmware.com/en/VMware-Cloud-Director/10.0/com.vmware.vcloud.install.doc/" click here

[3]. VMware Integrated OpenStack Installation and Configuration Guide. "https://docs.vmware.com/en/VMware-Integrated-OpenStack/4.0/com.vmware.openstack.install.doc/" click here

[4]. Vmware system compatibility guide. "https://www.vmware.com/resources/compatibility/pdf/vi_systems_guide.pdf" click here

[5]. Open Stack user guide "https://docs.openstack.org/mitaka/user-guide/" click here

[6]. Creating your own VNF package "https://osm.etsi.org/wikipub/index.php/Creating_your_own_VNF_package" click here

[7]. FOG O5 guide https://fog05.io/docs/overview/ click here here

[8]. Eclipse fog05 with ETSI OSM Orchestrator "https://fog05.io/docs/getting-started/etsi-osm-interop/" click here

[9].