OSM MR Hackfest – Hack 2
Network Services General Instantiation

Gianpietro Lavado (Whitestack)
Alfonso Tierno (Telefonica)
Felipe Vicens (Atos)
New OSM instance!

This time, we will use a shared OSM instance, leveraging the OSM RBAC capabilities. Don’t forget to create your VIM.

```
osm vim-create --name openstack-site-hackfest-x --user osm_hackfest_x --password osm_hackfest_x --auth_url http://172.21.247.1:5000/v3 --tenant osm_hackfest_x --account_type openstack
```
VNF: hackfest_multivdu_vnfd

VDU: mgmtVM
- Image name: cirros034
- VM Flavor: 1 CPU, 128 RAM, 1 GB disk
- Interfaces:
  - mgmtVM-eth0: VIRTIO
  - mgmtVM-eth1: VIRTIO

ICP: mgmtVM-internal

mgmtVM-eth0

VL: internal

ICP: dataVM-internal

dataVM-eth0
dataVM-xe0

VDU: dataVM
- Image name: cirros034
- VM Flavor: 1 CPU, 128 RAM, 1 GB disk
- Interfaces:
  - dataVM-eth0: VIRTIO
  - dataVM-xe0: VIRTIO

External Connection point: vnf-mgmt

External Connection point: vnf-data
Final Multi-VDU Picture

NS: hackfest_multivdu-ns
VNF: hackfest_multivdu-vnf

VNF 1 / VDU: mgmtVM
VNF 1 / VDU: dataVM

VNF 2 / VDU: mgmtVM
VNF 2 / VDU: dataVM

Internal VLDs & CPs

External VLDs & CPs
VNF/NS Compose

• Compose a VNF or NS graphically.
VNF: hackfest_multivdu_vnfd

VDU: mgmtVM
- Image name: cirros034
- VM Flavor: 1 CPU, 128MB RAM, 1 GB disk
- Interfaces:
  - mgmtVM-eth0: VIRTIO
  - mgmtVM-eth1: VIRTIO

External Connection point: vnf-mgmt

VL: internal

ICP: mgmtVM-internal

mgmtVM-eth0
mgmtVM-eth1

External Connection point: vnf-data

VDU: dataVM
- Image name: cirros034
- VM Flavor: 1 CPU, 128MB RAM, 1 GB disk
- Interfaces:
  - dataVM-eth0: VIRTIO
  - dataVM-xe0: VIRTIO

dataVM-eth0
dataVM-xe0

ICP: dataVM-internal
User Interface

• Steps:
  • Compose a new VNF

• Create new Package

Create new Package

Package name: MultiVDU_envId

[Cancel] [Create]
VNFD Composer

- **Steps**
  - VNFD Composer

- **Keyboard shortcuts**
  - Create edge: Select the first vertex by clicking on it. **Shift + left-click** on another vertex (different than the selected one).
  - Delete edge: Select the vertex by clicking on it. **right-click + Delete**
Creating the new multi-VDU VNF (1/4)

- **Steps**
  - Create VDUs
  - (Drag and drop)

<table>
<thead>
<tr>
<th>VDU</th>
<th>SAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>1</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>mgmtVM</td>
</tr>
<tr>
<td>image</td>
<td>cirros034</td>
</tr>
<tr>
<td>Id</td>
<td>mgmtVM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VDU</th>
<th>SAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>1</td>
</tr>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>dataVM</td>
</tr>
<tr>
<td>image</td>
<td>cirros034</td>
</tr>
<tr>
<td>Id</td>
<td>dataVM</td>
</tr>
</tbody>
</table>

- Edit the descriptor to add the flavor:
  - `{vm-flavor: {memory-mb: '128', storage-gb: '1', vcpu-count: '1'}}`
Creating the new multi-VDU VNF (2/4)

- **Steps**
  - Create Connection Points: vnf-mgmt vnf-data (Drag and drop)

  ![Connection Points](image)

  - Link CPs with VDUs (Shift + Left Click)

  ![VDU Connections](image)
Creating the new multi-VDU VNF (3/4)

• **Steps**
  
  • Create Internal VL: ![internal](Drag and drop)
  
  • Link internal VL with VDUs (Shift + Left Click)
    • VNFD composer automatically create the internal connection points:
      ```yaml
      mgmtVM-internal  dataVM-internal
      ```
  
  • Edit the descriptor to add the CP in mgmt-interface
    ```yaml
    mgmt-interface:
      cp: 'vnfd-mgmt'
    ```
Creating the new multi-VDU VNF (4/4)

- Final Scenario multiVDU_vnfd

And finally, this is the sample file of Hackfest Multi VDU VNF Descriptor
https://osm-download.etsi.org/ftp/osm-6.0-six/7th-hackfest/packages/hackfest_multivdu_vnf.tar.gz
NS diagram

NS: hackfest_multivdu_nsd

VNF: hackfest_multivdu_vnfd
CP: vnf-data

VNF: hackfest_multivdu_vnfd
CP: vnf-mgmt

VL: mgmtnet
VL: datanet
User Interface

• Steps:
  • Compose a new NS

  ![NS Packages](image1.png)

  ![Compose a new NS](image2.png)

• Create new Package

  ![Create new Package](image3.png)
NSD Composer

• Steps
  • NSD Composer

• Keyboard shortcuts
Creating the NSD (1/3)

• Select VNFs: [MultiVDU_vnf:1] [MultiVDU_vnf:2] (Drag and drop)

<table>
<thead>
<tr>
<th>VNF</th>
<th>VNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>member-vnf-index 1</td>
<td>member-vnf-index 2</td>
</tr>
<tr>
<td>vnfd-id-ref MultiVDU_vnfd</td>
<td>vnfd-id-ref MultiVDU_vnfd</td>
</tr>
</tbody>
</table>

• Create VLs: [mgmtnet] [datanet] (Drag and drop)

<table>
<thead>
<tr>
<th>Virtual Link</th>
<th>Virtual Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vim network name</td>
<td>Vim network name</td>
</tr>
<tr>
<td>Name</td>
<td>mgmtnet</td>
</tr>
<tr>
<td>Mgmt network</td>
<td>true</td>
</tr>
<tr>
<td>Type</td>
<td>ELAN</td>
</tr>
<tr>
<td>Id</td>
<td>mgmtnet</td>
</tr>
</tbody>
</table>
Creating the NSD (2/3)

• Steps

  • Link VLs with VNFs (Shift + Left Click)

  • You need to know the name for the CPs (vnf-data and vnf-mgmt)
Creating the NSD (3/3)

- Final Scenario multiVDU_nsd

And finally, this is the sample file of Hackfest Multi VDU VNF Descriptor

https://osm-download.etsi.org/ftp/osm-6.0-six/7th-hackfest/packages/hackfest_multivdu_nsd.tar.gz
Deploying NS in the UI

- Visit the catalogue and confirm elements are created
- Launch the NS from the UI
  - Depending on the VIM, specify a VIM network name to map `mgmtnet`
  - If you need to change the VIM, change the network name using `config`:
    ```
    {vld: [{name: mgmtnet, vim-network-name: osm-ext}]}
    ```
- Click the info button to see the mgmt IP address of each VNF
- Connect to each VNF:
  - `ssh cirros@<IP>`
    - `password: cubswin:`
Advanced Network modelling!

The rich OSM Information Model and instantiation-time parameters allow for:

- **IP Profiles** for modifying the subnet parameters
- Assigning **fixed ip addresses** to connection points
- Assigning **fixed MAC addresses** to connection points
- Multi-site deployments

...and more!

More information here:
https://osm.etsi.org/docs/user-guide/05-osm-usage.html#advanced-instantiation-using-instantiation-parameters

### 5.2. Advanced instantiation: using instantiation parameters

OSM allows the parametrization of NS or NSI upon instantiation (Day-0 and Day-1), so that the user can easily decide on the key parameters of the service without any need of changing the original set of validated packages.

Thus, when creating a NSInstance, it is possible to pass instantiation parameters to OSM using the `--config` option of the client or the `config` parameter of the UI. In this section we will illustrate through some of the existing examples how to specify those parameters using OSM client. Since this is one of the most powerful features of OSM, this section is intended to provide a thorough overview of this functionality with practical use cases.
Presentation & Hands-on: Automating Day-0 configuration through cloud-init
What is cloud-init and what can it be used for?

- It is a Linux package used to automate initial configuration of a VM
- VM requirements:
  - Cloud-init package
  - Cloud-init configuration (data source) via /etc/cloud/cloud.cfg
    - Config drive
    - Openstack metadata server
    - ...
- What can be done?
  - Setting a default locale
  - Setting an instance hostname
  - Generating instance SSH private keys
  - Adding SSH keys to a user’s .ssh/authorized_keys so they can log in
  - Setting up ephemeral mount points
  - Configuring network devices
  - Adding users and groups
  - Adding files

- Cloud-init is available in Linux VMs and might be supported in other OS
- Not all VIMs support cloud-init via a metadata server
Hands-on: Get and explore the packages

• VNF Package
  • https://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest_cloudinit_vnf.tar.gz

• NS Package
  • https://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest_cloudinit_ns.tar.gz
NS: hackfest_cloudinit_nsd

VNF: hackfest_cloudinit_vnfd
CP: vnf-mgmt
CP: vnf-data

VL: mgmtnet
VL: datanet

VNF: hackfest_cloudinit_vnfd
CP: vnf-mgmt
CP: vnf-data
#cloud-config
password: osm4u
chpasswd: { expire: False }
ssh_pwauth: True

write_files:
  - content: |
      # My new helloworld file
      
      owner: root:root
      permissions: '0644'
      path: /root/helloworld.txt

A password is added for the default user (‘ubuntu’).
Put whatever password you want!

A new file ‘/root/helloworld.txt’ will be created at VM creation to illustrate the way this feature works.
Hands on: Deploying NS in the UI

1. Onboard the VNFD and NSD
2. Select `hackfest_cloudinit_nsd` and instantiate it
   If you need to change the VIM, change the network name using config:
   ```
   {vld: [{name: mgmtnet, vim-network-name: osm-ext}]}
   ```
3. Complete the form
   a. Add a name to the NS
   b. Select the Datacenter where the NS will be deployed
4. Go to the dashboard to see the instance and get the mgmt IP address of the VNF
5. **Connect to each VNF:** `ssh ubuntu@<IP>`
6. Check that the cloud-config file was executed by entering your password and checking the file.
Bonus Hands-on: Build the VNF!

VDU: mgmtVM
- Image name: hackfest3-mgmt
- VM Flavor: 1 CPU, 1GB RAM, 10 GB disk
- Interfaces:
  - mgmtVM-eth0: VIRTIO
  - mgmtVM-eth1: VIRTIO
- Cloud init input

VDU: dataVM
- Image name: hackfest3-mgmt
- VM Flavor: 1 CPU, 1GB RAM, 10 GB disk
- Interfaces:
  - dataVM-eth0: VIRTIO
  - dataVM-xe0: VIRTIO

External Connection point: vnf-mgmt
External Connection point: vnf-data

VNF: hackfest_cloudinit_vnfd
VL: internal
ICP: mgmtVM-internal
ICP: dataVM-internal
Bonus Hands-on: Creating the new CloudInit VNF (1/5)

- Use the tool to create a new VNFD called "hackfest_cloudinit_vnfd":
  osm package-create --image hackfest3-mgmt vnf hackfest_cloudinit

- Descriptor located at
  hackfest_cloudinit_vnf/hackfest_cloudinit_vnfd.yaml

- Add 2 Connection Points (external):
  - CONNECTION POINT 1:
    • name: vnf-mgmt
  - CONNECTION POINT 2:
    • name: vnf-data

- Add new VLD ‘internal’ to the VNF:
  - Name: internal
  - TYPE: ELAN
  - Refer to internal CPs we will define later

```
Internal VLD example
...
  mgmt-interface:
    cp: vnf-mgmt
  connection-point:
    - id: vnf-mgmt
      name: vnf-mgmt
      type: VPORT
    - id: vnf-data
      name: vnf-data
      type: VPORT
  internal-vid:
    - id: internal
      name: internal
      short-name: internal
      type: ELAN
      internal-connection-point:
        - id-ref: mgmtVM-internal
        - id-ref: dataVM-internal
  ...
```
Bonus Hands-on: Creating the new CloudInit VNF (2/5)

• Add VDU1 in the VNF
  • Name: mgmtVM
  • Image: hackfest3-mgmt
  • VM Flavor:
    • VCPU COUNT: 1
    • MEMORY MB: 1024
    • STORAGE GB: 10
  • Add 1 internal connection point:
    • ID: mgmtVM-internal
    • Name: mgmtVM-internal
    • Type: VPORT
  • Add 2 interfaces to the VDU:
    • Interface 1:
      • Name: mgmtVM-eth0
      • Position: 1
      • Connection-point-type: EXTERNAL
      • EXTERNAL-CONNECTION-POINT-REF: vnf-mgmt
      • Virtual-interface:
        • Type: VIRTIO
    • Interface 2:
      • Name: mgmtVM-eth1
      • Position: 2
      • Connection-point-type: INTERNAL
      • INTERNAL-CONNECTION-POINT-REF: mgmtVM-internal
      • Virtual-interface:
        • Type: VIRTIO
Bonus Hands-on: Creating the new CloudInit VNF (3/5)

- Add VDU2 in the VNF
  - Name: dataVM
  - Image: hackfest3-mgmt
  - VM Flavor:
    - VCPU COUNT: 1
    - MEMORY MB: 1024
    - STORAGE GB: 10
- Add 1 internal connection point:
  - ID: dataVM-internal
  - Name: dataVM-internal
  - Type: VPORT
- Add 2 interfaces to the VDU:
  - Interface 1:
    - Name: dataVM-eth0
    - Position: 1
    - Connection-point-type: INTERNAL
    - INTERNAL-CONNECTION-POINT-REF: dataVM-internal
    - Virtual-interface:
      - Type: VIRTIO
  - Interface 2:
    - Name: dataVM-xe0
    - Position: 2
    - Connection-point-type: EXTERNAL
    - EXTERNAL-CONNECTION-POINT-REF: vnf-data
    - Virtual-interface:
      - Type: VIRTIO
Bonus Hands-on: Creating the new CloudInit VNF (4/5)

• Download cloud-init file:
  
  https://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/other/cloud-config.txt

• Modify VDU mgmtVM in VNFD:
  • cloud-init-file: cloud-config.txt
  • Add cloud-init file to hackfest_cloudinit_vnf/cloud_init folder
Bonus Hands-on: Creating the new CloudInit VNF (5/5)

- Validate your descriptor using the tool:
  `osm package-validate hackfest_cloudinit_vnf`

- Generate VNF package:
  `osm package-build hackfest_cloudinit_vnf`
NS: hackfest_cloudinit_nsd

VNF: hackfest_cloudinit_vnfd
CP: vnf-data
VL: datanet

VNF: hackfest_cloudinit_vnfd
CP: vnf-data
VL: datanet

VNF: hackfest_cloudinit_vnfd
CP: vnf-mgmt
VL: mgmtnet

VNF: hackfest_cloudinit_vnfd
CP: vnf-mgmt
VL: mgmtnet

Bonus Hands-on: Build the NS!
Bonus Hands-on: Creating the NS (1/3)

- Use the tool to create a new NSD called "hackfest_cloudinit_nsd":
  
  ```
  osm package-create ns hackfest_cloudinit
  ```

- The descriptor is located at `hackfest_cloudinit_ns/hackfest_cloudinit_nsd.yaml`

- Specify constituent VNFs:
  
  ```yaml
  constituent-vnfd:
  - vnfd-id-ref: hackfest_cloudinit-vnf
    member-vnf-index: '1'
  - vnfd-id-ref: hackfest_cloudinit-vnf
    member-vnf-index: '2'
  ```
Bonus Hands-on: Creating the NS (1/3)

- Add first VLD:
  - id: mgmtnet
    name: mgmtnet
    short-name: mgmtnet
    type: ELAN
    mgmt-network: 'true'
    vim-network-name: osm-ext
    vnfd-connection-point-ref:
      - vnfd-id-ref: hackfest_cloudinit-vnf
        member-vnf-index-ref: '1'
        vnfd-connection-point-ref: vnf-mgmt
      - vnfd-id-ref: hackfest_cloudinit-vnf
        member-vnf-index-ref: '2'
        vnfd-connection-point-ref: vnf-mgmt
• Add second VLD:
  - `id: datanet`
    `name: datanet`
    `short-name: datanet`
    `type: ELAN`
    `vnfd-connection-point-ref:
      - vnfd-id-ref: hackfest_cloudinit-vnf`
      `member-vnf-index-ref: '1'
      `vnfd-connection-point-ref: vnf-data`
    - `vnfd-id-ref: hackfest_cloudinit-vnf`
      `member-vnf-index-ref: '2'
      `vnfd-connection-point-ref: vnf-data"
Bonus Hands-on: Creating the NSD (3/3)

- Validate your descriptor using the tool:
  `osm package-validate hackfest_cloudinit_ns`

- Generate NS package:
  `osm package-build hackfest_cloudinit_ns`
Presentation & Hands-on: Enabling high performance on VNFs - SDN-Assist
EPA (Enhanced Platform Awareness)

- **EPA features** like the use of large hugepages memory, dedicated CPUs, strict NUMA node placement, the use of passthrough and SR-IOV interfaces, can be used in OSM's VNF descriptors since Rel ZERO.

- If your VIM supports EPA, then you don't need to do anything extra to use it from OSM. VIM connectors in OSM take advantage of EPA capabilities if the VIM supports it. All you need to do is build your descriptors and deploy.

- Openstack configuration for EPA ([reference guide](#))
EPA examples

- **Huge pages**: allocates big chunks of memory to avoid having too many entries to look up.
- **CPU Pinning**: binds a set of CPUs to a VM, improving the performance by avoiding degrading events such as cache misses.
- **NUMA (Non-Uniform Memory Access)**: it provides separate memory for each processor to avoid the decrease on performance when several processors attempt to address the same memory.
- **SR-IOV (Single Root Input/Output Virtualization)**: specification that allows the isolation of a PCI Express resource (e.g. a network interface) for manageability and performance reasons.
**SR-IOV** allows VNFs to have direct access to a virtualized PCI of a NIC, thus giving it better throughput.

**Note:** Enabling SR-IOV requires a node reload for reconfiguration of the IOMMU virtualization mode. It also requires physical interfaces to be dedicated to this feature.
### Provisioning SR-IOV with OpenStack

1. Create a network that uses the SR-IOV physhnet
   ```
   openstack network create --provider-network-type=vlan --provider-physical-network=physnet2 --provider-segment=110 sriov-vlan110
   ```

2. Create the corresponding subnet
   ```
   openstack subnet create --no-dhcp --network=sriov-vlan110 --subnet-range=11.0.0.0/24 sriov-vlan110-subnet
   ```

3. Create a port with the direct binding at the subnet
   ```
   openstack port create --network sriov-vlan110 --vnic-type=direct --binding-profile trusted=true sriov-vlan110-port01
   ```

4. Launch the VM using that port
   ```
   openstack server create --flavor m1.large --image bionic --nic port-id=sriov-vlan110-port01 vm01
   ```

5. Repeat as needed, then configure the switches

### Automating SR-IOV connectivity for the complete NS with OSM

1. Add the VIM that includes the physhnet (once)
   ```
   osm vim-create --name VIMSRIOV --user user --password password --auth_url http://172.21.7.5:5000/v3 --tenant tenant --account_type openstack --config '{dataplane_physical_net: physnet2, microversion: 2.32}'
   ```

2. Model your VNF(s) to use SR-IOV instead of VIRTIO
   ```
   interface:
   - name: dataVDU
type: EXTERNAL
virtual-interface:
type: SR-IOV
   ```

3. Launch your NS, OSM will take care of everything, including VLD interconnectivity if the fabric was pre-configured with SDN Assist.

**More info on SDN Assist:**

**Note:** In all cases, the image you use should have the driver for supporting the physical NIC
1. OSM orquestrates SR-IOV
   → Proper assignment of I/O physical interfaces to the VM (VFs = Virtual Functions)

2. OSM SDN Assist gives the ability to create L2 connections between VFs
   • Interconnecting VMs
   • Attaching external traffic sources
A basic example of SR-IOV in OSM

You could clone the Hackfest-basic as the baseline for building:

http://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest-basic-vnf-sriov.tar.gz

http://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest-basic-ns-sriov.tar.gz
1. OSM orchestrates two VNFs: Gen & vBNG. Each one with:
   → EPA: 10 cores, 32 GB huge pages, 2 PCIe passthrough interfaces
2. VIM deploys and allocates physical interfaces to the VM
3. OSM SDN Assist creates requested L2 connectivity
Bonus Hands-on: Enabling EPA in a VNF

- Starting from hackfest_basic_vnfd we can build the following VNF:

  VNF: hackfest_epa_vnfd

  External Connection point: vnf-mgmt

  VDU:
  - Name: mgmtVM
  - Resources: 1 CPU, 1GB RAM, 10 GB disk
  - Image name: hackfest3-mgmt
  - Guest EPA
  - Cloud-init

  Interface: mgmtVM-eth0

Note: changes highlighted in yellow
1. Download the nsd and vnfd packages

```
wget http://osm-download.etsi.org/ftp/osm-5.0-five/6th-hackfest/packages/hackfest_basic_vnf.tar.gz
wget http://osm-download.etsi.org/ftp/osm-5.0-five/6th-hackfest/packages/hackfest_basic_ns.tar.gz
```

2. Create the NSD and VNFD in OSM

```
osm vnfd-create hackfest_basic_vnf.tar.gz
osm nsd-create hackfest_basic_ns.tar.gz
```

3. From UI, edit the VNFD to **lower the memory to 256MB** (to reduce exhaustion), and include:

```
vdu:
  - ...
    guest-epa:
      cpu-pinning-policy: DEDICATED
      mempage-size: LARGE
```

4. Launch it!
Bonus Hands-on: Enabling EPA in a VNF

• In the VIM, the flavor of the VM should be a new custom one:

<table>
<thead>
<tr>
<th>Instance Name</th>
<th>Image Name</th>
<th>IP Address</th>
<th>Flavour</th>
<th>Key Pair</th>
<th>Status</th>
<th>Availability Zone</th>
<th>Task</th>
<th>Power State</th>
<th>Time since created</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>epa-1-mgmtVM 1-1</td>
<td>hackfest3-mgmt</td>
<td>172.21.248.111</td>
<td>mgmtVM-flv-1</td>
<td>-</td>
<td>Active</td>
<td>nova</td>
<td>None</td>
<td>Running</td>
<td>9 hours, 5 minutes</td>
<td>Create Snapshot</td>
</tr>
</tbody>
</table>

• Authenticate in your VIM using Openstack CLI to show the EPA flavor specs.

```
oracle@osmf:~$ source osm_hackfest_40.sh
oracle@osmf:~$ openstack flavor show mgmtVM-flv | grep properties
```
Presentation & Hands-on: Monitoring & Auto-scaling VNFs
Service Assurance MDG

Check the Monitoring and autoscaling user guide here!
https://osm.etsi.org/docs/user-guide/05-osm-usage.html#monitoring-and-autoscaling

Main components

- MON
  - Covers the basic use cases, with a solid architecture to expand them easily.
  - Opportunities to enhance usability.

- POL
  - Designed around the autoscaling use case. Starting to cover VNF alarms.
  - Architecture needs a revisit based on expected use cases.

- Prometheus
  - OSM’s TSDB for metrics since REL5
  - Opportunities to enhance multi-tenancy to match new RBAC capabilities.

Auxiliary/Optional Tools

- Grafana
  - Integrates seamlessly with Prometheus.
  - Great tool for enhancing usability of the system’s Service Assurance → should be included by default?

- ELK
  - Proved seamless integration with OSM.
  - Main use case remains at log processing where stack is used.

© ETSI 2020
Prometheus collects the following metrics from “MON Exporter”

<table>
<thead>
<tr>
<th>Metric</th>
<th>Collection type</th>
<th>Behavior</th>
<th>KPI</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIM Status</td>
<td>Infrastructure</td>
<td>By default</td>
<td>status (up/down)</td>
<td>vim_id</td>
</tr>
<tr>
<td>SDNC Status</td>
<td></td>
<td></td>
<td>status (up/down)</td>
<td>sdnc_id</td>
</tr>
<tr>
<td>VM Status</td>
<td></td>
<td></td>
<td>status (up/down)</td>
<td></td>
</tr>
<tr>
<td>VDU CPU Utilization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDU Memory Utilization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDU Packet forwarding</td>
<td>VNF</td>
<td>Enabled by descriptor</td>
<td></td>
<td>nsr_id, vnf_member_index, vdu_name</td>
</tr>
<tr>
<td>VNF Metrics through Juju</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(to be migrated to Prometheus exporters)
Metrics collection

- VDU Metric Collection from VIM

```json
vdu:
  id: hackfest_basic-VM
  ...
  monitoring-param:
    - id: "cpu_util"
      nfvi-metric: "cpu_utilization"
    ...
  monitoring-param:
    - id: "vnf_cpu_util"
      name: "vnf_cpu_util"
      aggregation-type: AVERAGE
      vdu-monitoring-param:
        vdu-ref: "hackfest_basic-VM"
        vdu-monitoring-param-ref: "cpu_util"

nfvi-metric corresponds to a OSM metric name which maps to the corresponding metric in each supported VIM
```
Scaling descriptors can be included and be tied to automatic reaction to VIM/VNF metric thresholds. An internal alarm manager is supported, so that both VIM and VNF metrics can trigger threshold-violation alarms and scaling actions.

```
scaling-group-descriptor:
  - name: "vdu_autoscale"
    min-instance-count: 0
    max-instance-count: 10
    scaling-policy:
      - name: "cpu_util_above_threshold"
        scaling-type: "automatic"
        threshold-time: 10
        cooldown-time: 120
        scaling-criteria:
          - name: "cpu_util_above_threshold"
            scale-in-threshold: 10
            scale-in-relational-operation: "LT"
            scale-out-threshold: 60
            scale-out-relational-operation: "GT"
            vnf-monitoring-param-ref: "vnf_cpu_util"

  vdu:
    - vdu-id-ref: hackfest_basic_metrics-VM
      count: 1
```
VNF Alarms (new)

• Alarms based on metric thresholds can be sent to webhooks

vdu:
  - alarm:
    - alarm-id: alarm-1
    operation: LT
    value: 20
    actions:
      alarm:
        - url: https://webhook.site/1111
      ok:
        - url: https://webhook.site/2222
      insufficient-data:
        - url: https://webhook.site/3333
      vnf-monitoring-param-ref: vnf_cpu_util
Objective: Follow RBAC structure for metric consumption.

- Prometheus does not support multi-tenancy, other projects need to be explored (e.g. Cortex)
- Short-term implementation is a label for `project_id` in all Prometheus metrics

Grafana Dashboard Automation

Objective: adding to the previous feature, a new “MON Dashboarder” component will take care of dashboard “lifecycle”.

<table>
<thead>
<tr>
<th>Updates in...</th>
<th>...automates these dashboards...</th>
<th>...and these Grafana resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSM installation</td>
<td>System Metrics, Admin Project-scoped</td>
<td>Admin-privileges</td>
</tr>
<tr>
<td></td>
<td>$\rightarrow$ beta available</td>
<td></td>
</tr>
<tr>
<td>OSM Projects</td>
<td>Project-scoped (Grafana “team” privileges)</td>
<td>Grafana “team” privileges</td>
</tr>
<tr>
<td></td>
<td>$\rightarrow$ beta available</td>
<td></td>
</tr>
<tr>
<td>OSM Users</td>
<td>-</td>
<td>Grafana users to teams</td>
</tr>
<tr>
<td>OSM Network Services</td>
<td>NS-scoped $\rightarrow$ beta available</td>
<td>-</td>
</tr>
</tbody>
</table>

...Let’s play with this!
1. From your SSH console, download new descriptors and upload them to OSM

```shell
wget http://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest_basic_metrics_vnfd.tar.gz
wget http://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/packages/hackfest_basic_metrics_nsd.tar.gz
osm vnfd-create hackfest_basic_metrics_vnfd.tar.gz
osm nsd-create hackfest_basic_metrics_nsd.tar.gz
```

2. Instantiate your NS

```shell
osm ns-create --ns_name hfmetrics_XX --nsd_name hackfest_basic-ns-metrics --vim_account openstack-site-hackfest-X --config '{vld: [ {name: mgmtnet, vim-network-name: osm-ext} ]}'
```

1. Access your VM and stress it out!

   osm vnf-list  # to find the IP address
   ssh ubuntu@172.21.248.XX  # password: osm4u

   yes > /dev/null &  # 4 or 5 times!

2. Wait for a bit (5 to 10 minutes due to current collection period), and watch it scale!
Presentation & Hands-on: Orchestrating Network Slices with OSM
Network Slicing Overview

In the scope of 5G, network slicing allows the network operator to deploy independent PLMNs with different capabilities.

Different network slices addressing different types of usage requiring different levels of functionality, performance and reliability.

Source: NGMN
ETSI NFV Framework for Network Slicing

Analysis of 3GPP and alignment of NFV architecture in the ETSI GR NFV-EVE 012

Considerations:

- Network Slice Subnet can be considered as an NFV Network Service.

- Network Functions can be described as VNF and PNF
Network Slicing in OSM

- **Network slice instance**: a set of network functions and the resources for these network functions which are arranged and configured, forming a complete logical network to meet certain network characteristics.

- The NSI contains:
  - NFs (e.g. belonging to AN and CN)
  - Information relevant to the interconnections between these NFs like topology of connections.
  - Individual link requirements (e.g. QOS attributes)

  - **The NSI is defined by a Network Slice Template.**
NST diagram
Network Slice Template Requirements

VNFD:
- `slice_hackfest_vnf.tar.gz`
- `slice_hackfest_middle_vnf.tar.gz`

NSD:
- `slice_hackfest_ns.tar.gz`
- `slice_hackfest_middle_ns.tar.gz`

NST:
- `slice_hackfest_nst.yaml`
- `slice_hackfest2_nst.yaml`

• The descriptors for the slice hackfest session are available:
VNFD - slice_hackfest_vnfd

**VNF**: slice_hackfest_vnfd

**VDU**:
- **Name**: ubuntu_slice-VM
- **Resources**: 1vCPU, 1GB RAM, 10GB Disk
- **Image name**: 'US1604'
- **Interfaces**:
  - eth0 VIRTIO
  - eth1 VIRTIO
VNF: slice_hackfest_middle_vnfd

VDU:
- Name: ubuntu_slice-VM
- Resources: 1vCPU, 1GB RAM, 10GB Disk
- Image name: 'US1604'
- Interfaces:
  - eth0 VIRTIO
  - eth1 VIRTIO
  - Eth2 VIRTIO
NSD - slice_hackfest_nsd
NSD: slice_hackfest_middle_nsd

VLD: nsd_vnfd_vld_data1

VLD: nsd_vnfd_vld_data2

VLD: nsd_vnfd_vld_mgmt

VNF: slice_hackfest_middle_vnfd

cp: nsd_cp_mgmt

cp: nsd_cp_data1

cp: nsd_cp_data2

© ETSI 2020
NST - sliceHackfest NST

NST: sliceHackfest_NSTD

VLD: nds_vnfd_vld_mgmt

VLD: nds_vnfd_vld_data1

VLD: nds_vnfd_vld_data2
nst:
  - id: slice_hackfest_nstd
    name: slice_hackfest_nstd
    SNSSAI-identifier:
      - slice-service-type: eMBB
    quality-of-service:
      id: 1

netslice-subnet:
  - id: slice_hackfest_nsd_1
    is-shared-nss: false
    description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
    nsd-ref: slice_hackfest_nsd
  - id: slice_hackfest_nsd_2
    is-shared-nss: true
    description: NetSlice Subnet (service) composed by 1 vnf with 3 cp
    nsd-ref: slice_hackfest_middle_nsd
  - id: slice_hackfest_nsd_3
    is-shared-nss: false
    description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
    nsd-ref: slice_hackfest_nsd

netslice-vld:
  - id: slice_vld_mgmt
    name: slice_vld_mgmt
    type: ELAN
    mgmt-network: true
    nss-connection-point-ref:
      - nss-ref: slice_hackfest_nsd_1
        nsd-connection-point-ref: nsd_cp_mgmt
      - nss-ref: slice_hackfest_nsd_2
        nsd-connection-point-ref: nsd_cp_mgmt
      - nss-ref: slice_hackfest_nsd_3
        nsd-connection-point-ref: nsd_cp_mgmt

  - id: slice_vld_data1
    name: slice_vld_data1
    type: ELAN
    nss-connection-point-ref:
      - nss-ref: slice_hackfest_nsd_1
        nsd-connection-point-ref: nsd_cp_data
      - nss-ref: slice_hackfest_nsd_2
        nsd-connection-point-ref: nsd_cp_data
      - nss-ref: slice_hackfest_nsd_3
        nsd-connection-point-ref: nsd_cp_data

  - id: slice_vld_data2
    name: slice_vld_data2
    type: ELAN
    nss-connection-point-ref:
      - nss-ref: slice_hackfest_nsd_2
        nsd-connection-point-ref: nsd_cp_data1
      - nss-ref: slice_hackfest_nsd_3
        nsd-connection-point-ref: nsd_cp_data
Creating a Network Slice Template (NST)

Information Model

```plaintext
module: nst
   +--rw nst* [id]
      +--rw id string
      +--rw name string
      +--rw SNSSAI-identifier
         | +--rw slice-service-type network-slice-type
         | +--rw slice-differentiator? string
      +--rw quality-of-service
         | +--rw id uint16
         | +--rw resource-type? resource-type
         | +--rw priority-level? uint16
         | +--rw packet-delay-budget? uint16
         | +--rw packet-error-rate? uint16
         | +--rw default-max-data-burst? uint16
```

NST - id, name, and slice parameters section

```plaintext
nst:
   - id: slice_hackfest_nst
     name: slice_hackfest_nst
     SNSSAI-identifier:
       slice-service-type: eMBB
     quality-of-service:
       id: 1
```

Creating a Network Slice Template (NST)

Information Model

```
+--rw netslice-subnet*
    |    +--rw id [id]
    |    +--rw description? string
    |    +--rw is-shared-nss? boolean
    |    +--rw nsd-ref -> /nsd:nsd-catalog/nsd/id
    |    |    +--rw instantiation-parameters
    |    |    |    +--......
```

NST - netslice-subnet section

```
- id: slice_hackfest_nsd_1
  is-shared-nss: false
  description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
  nsd-ref: slice_hackfest_nsd

- id: slice_hackfest_nsd_2
  is-shared-nss: true
  description: NetSlice Subnet (service) composed by 1 vnf with 3 cp
  nsd-ref: slice_hackfest_middle_nsd

- id: slice_hackfest_nsd_3
  is-shared-nss: false
  description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
  nsd-ref: slice_hackfest_nsd
```

*The Information model for the NST is available:*

## NST - netslice-subnet section

<table>
<thead>
<tr>
<th>netslice-subnet:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- id: slice_hackfest_nsd_1</td>
</tr>
<tr>
<td>is-shared-nss: false</td>
</tr>
<tr>
<td>description: NetSlice Subnet (service) composed by 1 vnf with 2 cp</td>
</tr>
<tr>
<td>nsd-ref: slice_hackfest_nsd</td>
</tr>
<tr>
<td>- id: slice_hackfest_nsd_2</td>
</tr>
<tr>
<td>is-shared-nss: true</td>
</tr>
<tr>
<td>description: NetSlice Subnet (service) composed by 1 vnf with 3 cp</td>
</tr>
<tr>
<td>nsd-ref: slice_hackfest_middle_nsd</td>
</tr>
<tr>
<td>- id: slice_hackfest_nsd_3</td>
</tr>
<tr>
<td>is-shared-nss: false</td>
</tr>
<tr>
<td>description: NetSlice Subnet (service) composed by 1 vnf with 2 cp</td>
</tr>
<tr>
<td>nsd-ref: slice_hackfest_nsd</td>
</tr>
</tbody>
</table>

## NSD - id, name, and NS parameters section

<table>
<thead>
<tr>
<th>nsd-catalog:</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsd:</td>
</tr>
<tr>
<td>id: slice_hackfest_nsd</td>
</tr>
<tr>
<td>name: slice_hackfest_nsd</td>
</tr>
<tr>
<td>short-name: slice_hackfest_ns</td>
</tr>
<tr>
<td>description: NetServiceDescriptor with 2 vnfs and 2 vld (mgmt and data networks)</td>
</tr>
<tr>
<td>vendor: OSM</td>
</tr>
<tr>
<td>version: '1.0'</td>
</tr>
<tr>
<td>logo: osm_2x.png</td>
</tr>
</tbody>
</table>

Creating a Network Slice Template (NST)

Information Model

```
|--rw netslice-vld* [id]
  |   |--rw id               string
  |   |--rw name?           string
  |   |--rw short-name?    string
  |   |--rw vendor?        string
  |   |--rw description?   string
  |   |--rw version?       string
  |   |--rw type?          manotypes:virtual-link-type
  |   |--rw root-bandwidth? uint64
  |   |--rw leaf-bandwidth? uint64
  |   |--rw provider-network
  |       |--rw physical-network? string
  |       |--rw segmentation_id? uint32
  |       |--rw mgmt-network? boolean
  |       |--rw nss-connection-point-ref* [nss-ref nsd-connection-point-ref]
  |       |      |--rw nss-ref - [nss-ref nsd-connection-point-ref]
  |       |             |   |--rw nsd-connection-point-ref -> /nst/netslice-subnet/id
  |       |             |       |--rw nsd-connection-point-ref -> /nsd:nsd-catalog/
  |       |             |                         nsd/connection-point/name
  |       |             |--rw ip-address? inet:ip-address

* The information model for the NST is available:
```

NST - netslice-vld section

```
netslice-vld:
  - id: slice_vld_mgmt
    name: slice_vld_mgmt
    type: ELAN
    mgmt-network: true
    nss-connection-point-ref:
      - nss-ref: slice_hackfest_nsd_1
        nsd-connection-point-ref: nsd_cp_mgmt
      - nss-ref: slice_hackfest_nsd_2
        nsd-connection-point-ref: nsd_cp_mgmt
      - nss-ref: slice_hackfest_nsd_3
        nsd-connection-point-ref: nsd_cp_mgmt
  - id: slice_vld_data1
    name: slice_vld_data1
    type: ELAN
    nss-connection-point-ref:
      - nss-ref: slice_hackfest_nsd_1
        nsd-connection-point-ref: nsd_cp_data
      - nss-ref: slice_hackfest_nsd_2
        nsd-connection-point-ref: nsd_cp_data1
```
Creating a Network Slice Template (NST)

**NST - netslice-subnet section**

```yaml
netslice-subnet:
  id: slice_hackfest_ns_1
  nsd-ref: slice_hackfest_nsd
```

**NSD - id, name, and NS parameters section**

```yaml
nsd-catalog:
  nsd:
    - id: slice_hackfest_nsd
      name: slice_hackfest_nsd
      short-name: slice_hackfest_ns
```

**NST - netslice-vld**

```yaml
netslice-vld:
  - id: slice_hackfest_vld_mgmt
    name: slice_hackfest_vld_mgmt
    type: ELAN
    mgmt-network: true
    nss-connection-point-ref:
      - nss-ref: slice_hackfest_nsd_1
        nsd-connection-point-ref: nsd_cp_mgmt
```

**NSD - connection-point section**

```yaml
connection-point:
  - name: nsd_cp_mgmt
    vld-id-ref: nsd_vnfd_vld_mgmt
```

* The information model for the NST is available: [http://osm-download.etsi.org/repository/ontology/ReleaseSIX/docs/osm-int/osm_im_trees/nst.html](http://osm-download.etsi.org/repository/ontology/ReleaseSIX/docs/osm-int/osm_im_trees/nst.html)
Before the netslice instantiation, Adding VNF, NS packages and NST

- **VNF package:**
  - `osm vnfd-list`
  - `osm vnfd-create slice_hackfest_vnf.tar.gz`
  - `osm vnfd-create slice_hackfest_middle_vnf.tar.gz`
  - `osm vnfd-show slice_hackfest_vnf`
  - `osm vnfd-show slice_hackfest_middle_vnf`

- **NS package:**
  - `osm nsd-list`
  - `osm nsd-create slice_hackfest_ns.tar.gz`
  - `osm nsd-create slice_hackfest_middle_ns.tar.gz`
  - `osm nsd-show slice_hackfest_nsd`
  - `osm nsd-show slice_hackfest_middle_nsd`

- **NST:**
  - `osm nst-list`
  - `osm nst-create slice_hackfest_nst.yaml`
  - `osm nst-create slice_hackfest2_nst.yaml`
  - `osm nst-show slice_hackfest_nst`
Creating a Network Slice Instance (CLI)

- osm nsi-create --help

Usage: osm nsi-create [OPTIONS]
creates a new Network Slice Instance (NSI)

Options:
- --nsi_name TEXT  name of the Network Slice Instance
- --nst_name TEXT  name of the Network Slice Template
- --vim_account TEXT  default VIM account id or name for the deployment
- --ssh_keys TEXT  comma separated list of keys to inject to vnfs
- --config TEXT  Netslice specific yaml configuration:
  netslice_subnet: [
    id: TEXT, vim_account: TEXT,
    vnfs: [member-vnf-index:
      TEXT, vim_account: TEXT]
  vlds: [name: TEXT, vim_network:
    name: TEXT or DICT with vim_account, vim_net entries],
  netslice-vld: [name: TEXT, vim_network-name: TEXT or
    DICT with vim_account, vim_net entries]
- --config_file TEXT  nsi specific yaml configuration file
- --help  Show this message and exit.

- osm nsi-create --nsi_name my_first_slice --nst_name slice_hackfest_nst --vim_account <replace_vim_account_name> --config 'netslice-vld: [{ "name": "slice_vld_mgmt", "vim-network-name": <replace_vim_external_network> }]'
Listing and Deleting a Network Slice Instance (CLI)

- List Network Slice Instances
  - `osm nsi-list`
- Delete Network Slice Instance
  - `osm nsi-delete <nsi_name>` or `<nsi_id>`
Managing Network Slice Templates (via CLI)

- Creates a new Network Slice Template
  - `netslice-template-create / nst-create`
- Deletes a Network Slice Template
  - `netslice-template-delete / nst-delete`
- List all Network Slice Templates
  - `netslice-template-list / nst-list`
- Shows the content of a Network Slice Template
  - `netslice-template-show / nst-show`
- Updates a Network Slice Template
  - `netslice-template-update / nst-update`
Managing Network Slice Instances (via CLI)

- Creates a new Network Slice Instance
  - netslice-instance-create / nsi-create
- Deletes a Network Slice Instance
  - netslice-instance-delete / nsi-delete
- List all Network Slice Instances (NSI)
  - netslice-instance-list / nsi-list
- Shows the history of operations over a
  - netslice-instance-op-list / nsi-op-list
- Shows the info of an operation over a Network Slice Instance (NSI)
  - netslice-instance-op-show / nsi-op-show
- Shows the content of a Network Slice Instance (NSI)
  - netslice-instance-show / nsi-show
Shared Network Slices

- Let’s share the middle NS using a new NST

NST: slice_hackfest2_nst
VLD: nsd_vnfd_vld_mgmt

NSD
slice_hackfest_middle_nsd
(shared: TRUE)

VLD: nds_vnfd_vld_data2

NSD
slice_hackfest_nsd
Network Slice Template 2

nst:
- id: slice_hackFest2_nst
  name: slice_hackFest2_nst
  SNSSAI-identifier:
    slice-service-type: eMBB
  quality-of-service:
    id: 1
  netslice-subnet:
    - id: slice_hackfest_nsd_2
      is-shared-nss: true
      description: NetSlice Subnet (service) composed by 1 vnf with 3 cp
      nsd-ref: slice_hackfest_middle_nsd
    - id: slice_hackfest_nsd_3
      is-shared-nss: false
      description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
      nsd-ref: slice_hackfest_nsd
  netslice-vld:
    - id: slice_vld_mgmt
      name: slice_vld_mgmt
      type: ELAN
      mgmt-network: true
      nss-connection-point-ref:
        - nsd-ref: slice_hackfest_nsd_2
          nsd-connection-point-ref: nsd_cp_mgmt
        - nsd-ref: slice_hackfest_nsd_3
          nsd-connection-point-ref: nsd_cp_mgmt

- id: slice_vld_data2
  name: slice_vld_data2
  type: ELAN
  nss-connection-point-ref:
    - nsd-ref: slice_hackfest_nsd_2
      nsd-connection-point-ref: nsd_cp_data2
    - nsd-ref: slice_hackfest_nsd_3
      nsd-connection-point-ref: nsd_cp_data

Creating a Network Slice Instance (UI)

- List Network Slice Templates
  - In the left panel, click on the Netslice Templates icon

![Network Slices Templates](image)
Creating a Network Slice Instance (UI)

New NSI

- Name: my_netslice
- Description: Description
- Nst Id: slice_hackfest_nst
- Vm Account Id: Openstack_7.18
- SSH Key: Paste your key here...
- Config:
  - netslice-vld:
    - name: slice_vld_mgmt
    - vim-network-name: osm-ext

Actions: Create, Cancel
Shared Network Slices

- Final deployment

VLD: nsd_vnfd_vld_data2
VLD: nsd_vnfd_vld_data1
VLD: nsd_vnfd_vld_mgmt
Shared Network Slices

- Result of deleting NSI 1

VLD: nsd_vnfd_vld_data2
VLD: nsd_vnfd_vld_data1
VLD: nsd_vnfd_vld_mgmt

NSD
slice_hackfest_middle_nsd
(SHARED NSS)

NSD
slice_hackfest_nsd

NSI 2
Open Source MANO

Find us at:

osm.etsi.org
osm.etsi.org/wikipub