VNF Orchestration: Virtual Machines, Public and Private Cloud

Preethika P. (Tata Elxsi)

October 18, 2022
Welcome to the Hackfest
Hackfest Environment

SSID: OSM_Hackfest
Password: WIFI4hackfest!

You

Remote Desktop

Virtual Desktop

OpenStack

OSM

Kubernetes

Hub for Interoperability and Validation at ETSI (HIVE)
Your Openstack Tenant
Agenda

- Overview of OSM
- VNF Onboarding Workflow
- NextEPC Modelling
- NextEPC Deployment
Overview Of OSM
The ETSI NFV Architecture

NFVI (NFV Infrastructure)
- Computing Hardware
- Storage Hardware
- Network Hardware

VNFs (Virtual Network Functions)
- EM 1
- EM 2
- EM 3

VNF Manager(s)
- VNF Manager(s)

VIM (Virtual Infrastructure Manager)
- Officially part of MANO, but usually bundled with NFVI
- Focus on VM lifecycle

Management & Orchestration
- Focus on VNF/NS lifecycle
- NFV Orchestration
- NFV Orchestrator

Oss/BSS
ETSI NFV Architecture and Specifications

All you need is a map

Source: ETSI. Web: https://www.etsi.org/images/articles/NFV%20Architecture.svg
Network Functions

• One or many nodes in a Network Infrastructure that has well defined interfaces and functional networking capability.

• Examples: Firewall, Router, EPC, IMS, etc

• Different Flavours of Network Functions
  ◦ Virtual Network Function
  ◦ Cloud Native/Container Network Function
  ◦ Physical Network Function
Network Services

• The Network Service (NS) is a topology of interconnected NF.

  • ABSTRACTION (NS definition): The topology is agnostic from the place where NF will be deployed.

  • PARTICULARIZATION (NS instance): When instantiating it, parameters are provided specific for those NF instances.

• It is deployed and operated as a whole.

• Examples: LTE, VPN, LAN internet, etc.
OSM IM

• Information model (IM) to define the different descriptor templates:
  • Network Function (NF)
  • Network Service (NS)
  • Network Slice (NST)

• OSM IM is based on YANG*.

• OSM IM aligned with ETSI NFV, derived from SOL006 (which, in turn, derives from IFA011 and IFA014).

• IFA011 describes the VNF descriptor specification whereas IFA014 on NS descriptor.
Modeling NF
NF Package vs NF Descriptor

VNF package

- VNFD
- VNF artifacts
- Additional metadata?

Resource description aspects
- VNF resource orchestration info (EPA resources and internal connectivity)

Management procedures
- VNF primitives
  - Day-1
  - Day-2
- Charms/EE

Additional info
- Icon, README, etc.
Modeling NS
NS Package vs NS Descriptor

- **NS package**
  - NSD
  - NS artifacts
  - Additional metadata?

- **Resource description aspects**
  - NS topology

- **Management procedures**
  - NS primitives
    - Day-1
    - Day-2
  - Charms

- **Additional info**
  - Icon, README, etc.
Different Stages

Design Time

• Building NF and NS descriptors by referencing the OSM Information Model.
• Building Day-1/Day-2 logic to complete the packages
• Testing your packages

Provisioning Time

• Onboarding: package validation and uploading

Run Time (see your VNF and NS in action)

• Instantiation (Deployment + Day 0 + Day 1). Here is where the operator decides:
• In which VIM to deploy
• What instantiation parameters to provide (specific IP addresses, configuration params)
• Operation phase (Day 2)
Multi Cloud Orchestration
Multi Site Orchestration
VNF Onboarding workflow

- Network Functions Virtualisation will only scale if all of the functions can be automated.

...specially true for 5G!

1. Instantiate Network Services/Slices, making VNFs manageable (“Day 0”)
2. Initialize VNFs so they provide the expected service (“Day 1”)
3. Operate the service: monitoring, reconfigurations and (closed-loop) actions (“Day 2”)

© ETSI
Deployment of NextEPC
Next EPC Overview

• MME | Mobility Management Entity
  • Keeps track of User Equipment registered on LTE network
  • Handles requests for network access - setting up and tearing down data sessions

• SGW | Serving Gateway
  • IP router with GTP support and charging functionality
  • Module for signalling between PGW and MME

• PGW | Packet Data Network Gateway
  • Provides access to external Packet Data Networks (i.e: internet)

• HSS | Home Subscriber Service
  • Stores subscriber keys and permitted services
Day 0 Requirements

• Objectives of Day 0
  • Provide the guidelines for including all the necessary elements in the VNF Package for its successful instantiation and management setup, so it can be further configured at later stages.

• Requirements
  • Images for OpenStack
  • Networks (already existing in VIM, or need to be created)
  • Hostnames and IP addresses if needed
NextEPC VNF Model

VDU: S/PGW + MME

eth0
eth1
eth2
eth3

S1
SGi

VDU: HSS

eth0
eth1

S6a

Management
NextEPC VNF Model

Management

VNF: NextEPC

VDU: S/PGW + MME
eth0
eth1
eth2
eth3

VDU: HSS
eth0

S1

Existing VIM Networks

SGi

Internal network, created by OSM

S6a

Management

© ETSI
Day1 Requirements

• Objectives of Day 1
  • Provide the guidelines to include all necessary elements in the VNF Package.
  • Allows the exposed services inside the VNF to be automatically initialized right after the VNF instantiation.

• Requirements
  • Identify dependencies between components
  • Define required configuration for service initialization
  • Identify the need for instantiation parameters
Day1 Requirements

- **S/PGW + MME requires:**
  - 4 network interfaces and addresses
  - SSH keys for management
  - Update of MME configuration with IP address of HSS
- **HSS requires:**
  - 2 network interfaces and addresses
  - SSH keys for management
  - Update of HSS configuration with IP address of SPGW
Day2 Requirements

- Objectives of Day 2:
  - Re-configure the VNF so its behavior can be modified at runtime
  - Monitor its main KPIs and raise alarms

- Requirements:
  - Identify dependencies between components
  - Define possible configurations for runtime operations
  - Define key performance indicators
    - VIM: CPU, memory, network or storage usage
    - VNF/EM: Active sessions, users, application status
  - Define closed loop operations - Scaling or healing
Day2 Requirements

- Add or remove subscribers
- Collect CPU and Memory metrics of the VDU
Charms in Action

- LXD/K8s Operators
  - Proxy Charm
  - Ops Code

- VIM
  - Native Charm
  - VNF

- Kubernetes
  - Native Charm
  - KNF

- Racks
  - PNF
  - PNF

- OSM

© ETSI
Charms in Action - Integration

“I can use a MySQL database”

“I can use LDAP”

“I can send my logs to a syslog”
Charms in Action - Integration

“I can use LDAP”

“I can provide LDAP”
Lines of integration between matching integration points on different charms
Charms in Action

Proxy Charm Relation

VNF: NextEPC
VDU: S/PGW + MME

eth0
eth1
eth2
eth3

VDU: HSS

eth0

Management

S1

SGi

Existing VIM Networks

Internal network, created by OSM

S6a

© ETSI
Hands On

• Ensure om-packages repo is cloned

```
$ git clone --recursive https://osm.etsi.org/gitlab/vnf-onboarding/osm-packages.git
```

• Move to Hackfest MR#13

```
$ cd osm-packages/Hackfest_Demos/OSM-MR13/
```

• Package and Onboard the packages

```
$ osm nfpkg-create vEPC_vnf
$ osm nspkg-create vEPC_ns
```
Verify Onboarding

$ osm nspkg-list

```
osm@osm:~$ osm-packages/Hackfest_Demos/OSM-MR13/NextEPC$ osm nsd-list
<table>
<thead>
<tr>
<th>nsd name</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>VEPC-ns</td>
<td>Sa5a8954-4848-478d-9599-db1cfd115dc0</td>
</tr>
</tbody>
</table>
```

$ osm vnfpkg-list

```
osm@osm:~$ osm-packages/Hackfest_Demos/OSM-MR13/NextEPC$ osm nfpkg-list
<table>
<thead>
<tr>
<th>nfpkg name</th>
<th>id</th>
<th>desc type</th>
</tr>
</thead>
<tbody>
<tr>
<td>VEPC-vnf</td>
<td>8f898aad-998-4c4e-b2a9-e1ec6782a5d21</td>
<td>so1086</td>
</tr>
</tbody>
</table>
```
Instantiate NextEPC Service

- Instantiate the wiki network service

```bash
$ osm ns-create --ns_name EPC --nsd_name vEPC-ns --config_file params.yaml --vim_account openstack
```

- Verify the network service is deployed properly. Same can be checked via UI

```bash
$ osm ns-list
```

<table>
<thead>
<tr>
<th>ns instance name</th>
<th>id</th>
<th>date</th>
<th>ns state</th>
<th>current operation</th>
<th>error details</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPC_1</td>
<td>7004ac93-cf81-4c7d-88ca-99de477bb6a5</td>
<td>2022-10-13T05:44:37</td>
<td>READY</td>
<td>IDLE (None)</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Verification of Service

• Check the juju status

\$ juju status --relations -m <NS_ID>

• List VMs deployed in the Openstack VIM. There should be

\$ openstack server list

• NS reaches the ‘configured’ state, ssh to the HSS and check if the HSS - MME session is established.

\$ sudo netstat -anlp | grep 10.0.6
Day 2 Action

- Test Day-2 reconfiguration operations by running the primitive that adds routes to the SPGW (no host routes allowed in this example). This can be done through the OSM UI or the CLI,

\[
\text{
$ \text{osm ns-action --v nf\_name vnf-epc --vdu\_id spgwmme --action\_name add-route --params \{}\text{\{external\_prefix: \"8.8.8.0/24\", next-hop: \"192.168.2.1\"\\}}\text{\}}\text{ EPC}
$ \]

- Verify the status of action from OSM

\[
\text{
$ \text{osm ns-op-list <NS\_Name>}
$ \text{}
$ \text{osm ns-op-show <day2\_action\_id>}
$ \]

Agenda

- Re-visiting VNF Onboarding Workflow
- Real World Use Case
- Hands on –Wiki Webserver
- Monitoring
- Auto Healing
- Alerts
- NS Policy Update
- Auto Scaling
The ETSI NFV Architecture
Re-visiting VNF Onboarding workflow

- Network Functions Virtualisation will only scale if all of the functions can be automated.

...specially true for 5G!

1. Instantiate Network Services/Slices, making VNFs manageable ("Day 0")
2. Initialize VNFs so they provide the expected service ("Day 1")
3. Operate the service: monitoring, reconfigurations and (closed-loop) actions ("Day 2")
Real World Use Case

- **5G Access Network**: Massive 5G throughput
- **5G Network Edge**: Low latency, ultra-reliable connections. Can co-locate with edge data center
- **5G Core**: Latency-tolerant applications. Can co-locate with HQ data center

Other Destinations
Real World Use Case

5G Access Network

Massive 5G throughput

5G Network Edge

Low latency, ultra-reliable connections. Can co-locate with edge data center

Edge Data Center

5G Core

Latency-tolerant applications. Can co-locate with HQ data center

HQ Data Center

Other Destinations

© ETSI
Real World Use Case

5G Access Network

5G Network

Edge Data Center

5G Core

HQ Data Center

Other Destinations

Massive 5G throughput

Low latency, ultra-reliable connections. Can co-locate with edge data center

Latency-tolerant applications. Can co-locate with HQ data center

Scaling Policy update required!!
NS update feature to the rescue
Real World Use Case
Service Assurance

Main Components

**MON**
- Covers the basic uses 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.

**PLA**
- Provides computation of optimal placement of NFs over VIMs.
- Considers cost of compute/network.

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

**Grafana**
- Integrates seamlessly with Prometheus.
- Great tool for enhancing usability of the system’s Service Assurance.

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

**Auxiliary/Optional**

**Grafana**
- Integrates seamlessly with Prometheus.
- Great tool for enhancing usability of the system’s Service Assurance.

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

• Ensure om-packages repo is cloned

  $ git clone --recursive https://osm.etsi.org/gitlab/vnf-onboarding/osm-packages.git

• Move to Hackfest MR#13

  $ cd osm-packages/Hackfest_Demos/OSM-MR13/1.2-VNF

• Package and Onboard the packages

  $ osm nfpkg-create wiki_webserver_autoscale_vnfd
  $ osm nspkg-create wiki_webserver_autoscale_nsd
Wiki Webserver

VDU: Apache-1
VDU: Apache-0
VDU: Haproxy

eth0
eth1
eth0
eth0

Management
Internal
Verify Onboarding

$ osm nspkg-list

$ osm vnfpkg-list
Instantiate Wiki Webservice Service

- Instantiate the wiki network service

```bash
$ osm ns-create --ns_name wiki_web_service --nsd_name wiki_webserver_autoscale_ns --vim_account openstack
```

- Verify the network service is deployed properly. Same can be checked via UI

```bash
$ osm ns-list
```

<table>
<thead>
<tr>
<th>ns_instance_name</th>
<th>id</th>
<th>date</th>
<th>ns_state</th>
<th>current_operation</th>
<th>error_details</th>
</tr>
</thead>
<tbody>
<tr>
<td>wiki_service_1</td>
<td>c98fa54-3b3f-46a9-af61-672dce35324</td>
<td>2022-10-08T11:52:55</td>
<td>READY</td>
<td>IDLE (None)</td>
<td>N/A</td>
</tr>
</tbody>
</table>

To get the history of all operations over a NS, run "osm ns-op-list NS_ID" 
For more details on the current operation, run "osm ns-op-show OPERATION_ID"
Verification of Service

- List VMs deployed in the Openstack VIM. There should be
  
  ```
  $ openstack server list
  ```

- Check if haproxy is working by trying to access the URL

  ```
  http://<HA Proxy External IP>/
  http://<HA Proxy External IP>:32700/
  ```
Verification of Service

- The Following page should be visible

Thank you for trying OSM!
Data Serving From 192.168.28.10

Thank you for trying OSM!
Data Serving From 192.168.28.8

Note: **Troubleshooting Tips** If HA proxy is not started automatically, play the following commands from haproxy VM

```
$ service haproxy status
$ sudo service haproxy restart
```
## Verification of Service

### Table 1: Service Verification

<table>
<thead>
<tr>
<th>Server</th>
<th>Network</th>
<th>Service</th>
<th>Type</th>
<th>Log</th>
<th>Status</th>
<th>Error</th>
<th>Time</th>
<th>Location</th>
<th>Result</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Diagram 1: Service Overview

![Service Overview Diagram](image)

### Query 1: OSM_VM_Status

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovm_vm_status</td>
<td></td>
</tr>
</tbody>
</table>
```

### Graph 1: Evaluation Time

```
<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>
```

---

© ETSI
Auto Healing

- Healing descriptors can be included and be tied to automatic reaction to VM metric thresholds.
- An internal alarm manager has been added to MON through the 'mon-evaluator' module, so VM metrics can trigger threshold-violation alarms when VM is in `ERROR/DELETE` state and perform healing actions.
Auto Healing – Hands on

- To replicate auto heal scenario, remove apache-VDU 1 from Openstack VIM

  $ openstack server list
  $ openstack server delete <Name of apache vdu 1>

- Check the status of Apache1 VM in Prometheus, the state would have moved to ‘0’

- Check from OSM for healing action taken

  $ osm ns-op-list <NS_Name>
Alert Functions

Alerts

- Alarm section defined within descriptor helps to send notification to external system based on metric monitored.

- An internal alarm manager has been added to MON through the 'mon-evaluator' module, so that both VIM and VNF metrics can also trigger threshold-violation alarms and scaling actions.
NS Policy Update (Lifecycle Management)

• Policy update is a part of NS Update CHANGE_VNFPKG operation and it can be used to update all the parameters related to policies like scaling-aspect, healing-aspect, etc after a NS instance is deployed and running.

• Policy update changes are performed on a running VNF instance unless `software-version` is changed in the new revision of VNFD.
### Scaling Group Descriptor – Hands on

The scaling descriptor is part of a VNFD. Like the example shows, it mainly specifies:

- An existing metric to be monitored, which should be pre-defined in the monitoring-param list (monitoring-parameter).
- The thresholds to monitor (scale-in/out threshold)
- The minimum amount of scaled instances to produce (max-scale-level).
- The minimum time it should pass between scaling operations (cooldown-time).
- The VDU to be scaled and the amount of instances to scale per event.

```
scaling-aspect:
-   aspect-delta-details:
      deltas:
-         id: apache_vdu_autoscale-delta
      vdu-delta:
          -         id: apache_vdu
      number-of-instances: 1
      id: apache_vdu_autoscale
      max-scale-level: 10
      name: apache_vdu_autoscale
      scaling-policy:
          -         cooldown-time: 180
      name: apache_cpu_util_above_threshold
      scaling-criteria:
          -         name: apache_cpu_util_above_threshold
            scale-in-relational-operation: LT
            scale-in-threshold: 20
            scale-out-relational-operation: GT
            scale-out-threshold: 40
            vnf-monitoring-param-ref: apache_vnf_cpu_util
      scaling-type: automatic
      threshold-time: 10
```
Auto Scaling

- Auto scaling allows to automatically scale VNFs with a VDU granularity and based on any available metric.
- Scaling descriptors can be included and be tied to automatic reaction to VIM/VNF metric thresholds.
- Supported metrics are both VIM and VNF metrics.
Let’s play with wiki service!

- Increase http traffic to the Wiki Web service
  
  ```bash
  $ sudo apt install apache2-utils
  $ ab -n 10000000 -c 50 http://<Haproxy-IP>/test.php
  ```

- Monitor traffic in Prometheus and Grafana

- Check the autoscale status from OSM

  ```bash
  $ osm ns-op-list <NS_Name>
  ```
Verification of Auto Scaling – OSM & VIM

• Check the scaled out instances in Openstack VIM

$ openstack server list

• Stop the traffic generator to see the scale in trigger

$ osm ns-op-list <NS_Name>
MON Architecture

Formal documentation: https://osm.etsi.org/gitlab/osm-architecture/osm-arch-doc/blob/master/04-mon.md
POL Architecture

Formal documentation: https://osm.etsi.org/gitlab/osm-architecture/osm-arch-doc/blob/master/05-pol.md
When configuring alarms associated to scaling actions or just webhook notifications (through the VNFD), the following components interact.

1. NBI continuously looks for configured alarms at VNF record.
2. Queries for metric values from `tsdb (prometheus)`.
3. When triggered, puts alarm in bus for `pol` to take actions.
4a. If action is to scale: send to bus for `lcm` to proceed and store action to commonDB.
4b. If action is to notify: send notification to webhook service.
4c. If action is to heal: sends to bus to `lcm` to heal the service.

- NBI
- commonDB (mongo)
- `lcm`
- `mon-evaluator`
- `tsdb (prometheus)`
- `pol`
- `Policy DB (MySQL)`
- `webhook service (external)`
Community installer

wget
chmod +x install_osm.sh
./install_osm.sh

Charmed installer

wget
chmod +x install_osm.sh
./install_osm.sh --charmed
Join us!

ETSI members, non-members, individual developers and users.

Learn [how to join](#)
Thank You!