OSM usage

Gerardo García (Telefónica, OSM TSC Chair)

OSM Training Seminar - SLICES

13/02/2024
Registering VIM accounts
Registering VIMs/Clouds

- With the GUI
Registering VIMs/Clouds

- With the GUI

![New VIM Account form](image-url)

- Name*
- VIM Project/Tenant Name*
- Type*
- Description
- VIM URL*
- Schema Type
- VIM Username*
- VIM Password*
- VIM Location
- Upload Config

* Mandatory fields are marked with an asterisk (*)

- To add a new TYPE, please enter input above
- Type the Data location name, Latitude & Longitude to show in map view

Please upload file with .yaml or .yml format
Registering VIMs/Clouds

● With the client:

```bash
osm vim-create --name etsi-vim-slicesX --account_type openstack \
   --auth_url http://172.21.247.1:5000/v3 \
   --user slicesX --password "slicesXFeb24!" --tenant slicesX\n   --description "ETSI VIM" \
   --config '{management_network_name: osm-ext,\n              dataplane_physical_net: physnet2}'
```
Exercise

- Register a VIM account
  - Name: etsi-vim-slicesX
  - Type: openstack
  - User, password and tenant/project: the ones in the Google Spreadsheet
  - Description: "ETSI VIM"

- Config params:
  - management_network_name: osm-ext
  - dataplane_physical_net: physnet2
VIM monitoring

- Airflow:
  - Osm-slices-1: http://172.21.249.14:11193 (admin, admin)
  - Osm-slices-2: http://172.21.249.74:17641 (admin, admin)
  - Osm-slices-4: http://172.21.249.60:16206 (admin, admin)

- Prometheus dashboard:
  - Osm-slices-4: http://172.21.249.60:9091
  - Check metric: osm_vim_status
Registering K8s clusters
Exercise

- Register a K8s cluster with the client

```sh
osm k8scluster-add --creds whitemist-kubeconfig.yaml \
   --version "v1.26" \
   --vim etsi-vim-slicesX \
   --k8s-nets "{net1: osm-ext}" \
   --description "K8s cluster" \
   --skip-jujubundle \
   k8scluster-slicesX
```
Create your own Kubernetes cluster

The simplest way is to create a VM and install K3s like this:

curl -sfL https://get.k3s.io | sh -
sudo k3s kubectl get node
mkdir .kube
sudo mv /etc/rancher/k3s/k3s.yaml ./kube/config
sudo chown $USER:$USER .kube/config
Modeling NF and NS
OSM provides a platform to create Networks as a Service and to manage them conveniently later...

OSM manages the low-level setup for Network Functions, so that they are ready for use.

- It covers in 100% the role of a kind of specialized PaaS for Network Functions, with 2 key features:
  1. Complex connectivity setup, including EPA and underlay scenarios.
  2. Solve inter-NF relations.

- Returns: NS/NF ready for its use and properly connected:
  - Exposes the “function” and its lifecycle, not its components.
  - Presented as a whole (i.e., abstracts from low-level details of the NF).
  - Easy (standardized) access to NF's lifecycle operations, via primitives.

- This follows well-known paradigms in IT and public clouds.
OSM provides a platform to create Networks as a Service (NaaS) and to manage them conveniently.
Packages embed resource description and operational procedures

- NFV Packages (VNF, PNF, HNF)
- NS Package
- Network Slice Package

- Mgmt Procedures
- Resource Description
- Topology

- OSM’s NBI
- EE (Charms)
- VIM/SDN Connectors

- Physical units: PNF 1, VNF 2, VNF 3
- Deployment

- VIM’s NBI
Packages embed resource description and operational procedures
All in OSM is model-driven to make VNFs and NS as portable and reusable as possible

- **Provided by the vendor**, fully describe their own product:
  - Topology
  - Parametrized
  - Actions for Day-0, Day-1, and Day-2

- **Doesn’t** need to know any detail about:
  - The target infrastructure
  - Other components that will be part of the scenario
All in OSM is model-driven to make VNFs and NS as portable and reusable as possible

(V)NF PACKAGES:

- Describes how to combine a set of NF packages to create a specific scenario.
- Parametrized.
- Have actions for Day-0, Day-1, and Day-2.

NS PACKAGES / SLICE PACKAGES:

Slice Packages work similarly, but using NS as building blocks

(*) NS instances play the role of Slice Subnets of a given slice. Some of them may be shared by more than one slice instance. This is taken into account by OSM, so a slice is more sophisticated than just a "NS of NS".
All in OSM is model-driven to make VNFs and NS as portable and reusable as possible

NS PACKAGES / SLICE PACKAGES:

Upon instantiation, you just need to decide:
- The target VIM (or VIMs)
- Values for the parameters (IP addresses, keys, etc.)
Modeling NF
VNF package vs VNF 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
- Execution Environments
  - Juju Charms, Helm EE

Additional info
- Icon, README, etc.

Execution Environments
(Juju Charms, Helm EE)
Descriptors are written in YAML and contain:

- **Topology description (VDU, internal VLD, Connection Points)**
- **Scaling-groups**
- **Monitoring params**
- **Reference to day-0 configuration file**
- **Execution environment list (e.g. charms, monitoring environments)**
- **Day-1 primitives (sequence)**
- **Day-2 primitives**

```yaml
vnfd:
  description: Virtual Desktop Computer
ext-cpd:
  - id: virtual-pc-private-ext
    int-cpd:
    cpd: eth0-int
    vdu-id: virtual-pc
    id: hackfest_virtual-pc_vnf
    mgmt-cp: virtual-pc-mgmt-ext
    product-name: hackfest_virtual-pc_vnf
sw-image-desc:
  - id: ubuntu20.04
    image: ubuntu20.04
vdu:
  cloud-init-file: virtual-pc_init
description: virtual-pc
id: virtual-pc
int-cpd:
  - id: eth0-int
  virtual-network-interface-requirement:
    - name: eth0
      virtual-interface:
        type: PARAVIRT
  - id: eth1-int
version: '1.0'
virtual-compute-desc:
  - id: virtual-pc-vdu-compute
  virtual-cpu:
    num-virtual-cpu: 8
  virtual-memory:
    size: 32.0
virtual-storage-desc:
  - id: virtual-pc-vdu-storage
```
Modeling NF
CNF descriptor

CNF descriptors must contain:

• List of KDU (and their associated helm-chart or juju-bundle)
• K8s cluster requirements

```
vnfd:
  description: CNF with single KDU
df:
  - id: default-df
ext-cpd:
  - id: mgmt-ext
    k8s-cluster-net: mgmtnet
id: openldap_knf
k8s-cluster:
  nets:
    - id: mgmtnet
kdu:
  - name: ldap
    helm-chart: stable/openldap
    mgmt-cp: mgmt-ext
    product-name: openldap_knf
    provider: Telefonica
    version: '1.0'
```
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.
Descriptors are written in YAML and contain:

- Topology description (NF, VL)

```yaml
nsd:
  description: Simple NS with a single VNF and a single VL
  df:
    id: default-df
    df-vnf-profile:
      id: vnf
      virtual-link-connectivity:
        constituent-cpd-id:
          constituent-base-element-id: vnf
          constituent-cpd-id: vnf-cp0-ext
          virtual-link-profile-id: mgmtnet
          vnfd-id: hackfest_basic-vnf
    id: hackfest_basic-ns
    name: hackfest_basic-ns
    version: 1.0
    virtual-link-desc:
      id: mgmtnet
      mgmt-network: true
    vnfd-id:
      - hackfest_basic-vnf
```
VNFD for hackfest_basic_vnf – Part 1

vnfd:
  description: A basic VNF descriptor w/ one VDU
  df:
    - id: default-df
      instantiation-level:
        - id: default-instantiation-level
          vdu-level:
            - number-of-instances: 1
              vdu-id: hackfest_basic-VM
              vdu-profile:
                - id: hackfest_basic-VM
                  min-number-of-instances: 1
  ext-cpd:
    - id: vnf-cp0-ext
      int-cpd:
        cpd: vdu-eth0-int
        vdu-id: hackfest_basic-VM
      id: hackfest_basic-vnf
      mgmt-cp: vnf-cp0-ext
      product-name: hackfest_basic-vnf

https://osm.etsi.org/gitlab/vnf-onboarding/osm-packages/-/blob/master/hackfest_basic_vnf/hackfest_basic_vnfd.yaml
VNFD for hackfest_basic_vnf – Part 2

```json
vdu:
  - id: hackfest_basic-VM
    name: hackfest_basic-VM
    sw-image-desc: ubuntu20.04
    virtual-compute-desc: hackfest_basic-VM-compute
    virtual-storage-desc:
      - hackfest_basic-VM-storage
    int-cpd:
      - id: vdu-eth0-int
        virtual-network-interface-requirement:
          - name: vdu-eth0
            virtual-interface:
              type: PARAVIRT

    virtual-compute-desc:
      - id: hackfest_basic-VM-compute
        virtual-cpu:
          num-virtual-cpu: 1
        virtual-memory:
          size: 1.0
      virtual-storage-desc:
        - id: hackfest_basic-VM-storage
          size-of-storage: 10
```

© ETSI
NSD for hackfest_basic_ns

```yaml
nsd:
  nsd:
    - description: Simple NS with a single VNF and a single VL
df:
    - id: default-df
      vnf-profile:
        - id: vnf
          vnfd-id: hackfest_basic-vnf
          virtual-link-connectivity:
            - constituent-cpd-id:
              - constituent-base-element-id: vnf
              - constituent-cpd-id: vnf-cp0-ext
              virtual-link-profile-id: mgmtnet
  id: hackfest_basic-ns
  name: hackfest_basic-ns
  version: 1.0
  virtual-link-desc:
    - id: mgmtnet
      mgmt-network: true
  vnfd-id:
    - hackfest_basic-vnf
```

https://osm.etsi.org/gitlab/vnf-onboarding/osm-packages/-/blob/master/hackfest_basic_ns/hackfest_basic_nsd.yaml
Adding NF and NS packages
Reference: OSM packages

- Gitlab: https://osm.etsi.org/gitlab/vnf-onboarding/osm-packages

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

- Use OSM client to upload hackfest_basic_vnf package
- Use OSM client to upload hackfest_basic_ns package
- Use OSM GUI to upload hackfest_basic_metrics_vnf package
- Use OSM GUI to upload hackfest_basic_metrics_ns package
Solution (with OSM client)

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

cd osm-packages

# Optional validation and build
# osm package-validate hackfest_basic_vnf
# osm package-validate hackfest_basic_ns
# osm package-build hackfest_basic_vnf
# osm package-build hackfest_basic_ns

osm nfpkg-create hackfest_basic_vnf
osm nspkg-create hackfest_basic_ns
NS instantiation and operation
Instantiation and termination
Exercise

- Launch a NS instance of hackfest_basic_ns
- Check VMs created in Openstack
- Check VNF reachability
- Terminate the NS instance
Solution (with OSM client)

# Launch instance
osm ns-create --ns_name hfbasic --nsd_name hackfest_basic-ns
    --vim_account etsi-vim-slicesX
    --ssh_keys ~/.ssh/id_rsa.pub
    --config '{vld: [ {name: mgmtnet, vim-network-name: osm-ext} ]}'

# Check VMs were created in Openstack
openstack server list

# Find mgmt IP address of the VNF
osm vnf-list
osm vnf-show <VNF_ID>

# Check VNF reachability
ssh ubuntu@<IP_ADDRESS>

# Terminate instance
osm ns-delete hfbasic
NS instantiation and operation
Working with CNF
Working with CNF

# Add packages
osm vnfpkg-create openldap_knf
osm nspkg-create openldap_ns

# Launch NS
osm ns-create --ns_name ldap --nsd_name openldap_ns --vim_account etsi-vim-slicesX
--config '{vld: [ {name: mgmtnet, vim-network-name: osm-ext} ],
additionalParamsForVnf: [ {member-vnf-index: openldap, additionalParamsForKdu: [ {kdu_name: ldap, additionalParams: {service: {type: LoadBalancer }, adminPassword: admin}} ] } ] }'

# Check number of replicas
osm vnf-show <VNF_ID> --kdu ldap | yq -r .config.replicaCount
Working with CNF

# Upgrade CNF
```
osm ns-action --action_name upgrade --vnf_name openldap --kdu_name ldap --params
'{"replicaCount":"3",}' ldap
```

# Check number of replicas
```
osm vnf-show <VNF_ID> --kdu ldap | yq -r .config.replicaCount
```

# Rollback CNF
```
osm ns-action --action_name rollback --vnf_name openldap --kdu_name ldap ldap
```

# Check number of replicas
```
osm vnf-show <VNF_ID> --kdu ldap | yq -r .config.replicaCount
```

# Terminate NS
```
osm ns-delete ldap
```
NS instantiation and operation
Scaling
Scaling scenario

# Add packages
osm nfpkg-create hackfest_basic_metrics_vnf
osm nspkg-create '/robot-systest/osm-packages/hackfest_basic_metrics_ns

# Launch NS
osm ns-create --ns_name manual_scaling_test --nsd_name hackfest_basic-ns-metrics --vim_account etsi-vim-slicesX --config '{vld: [ {name: mgmtnet, vim-network-name: osm-ext} ]}' --ssh_keys ~/.ssh/id_rsa.pub

# Scale out
osm vnf-scale --scaling-group vdu_autoscale --scale-out manual_scaling_test vnf

# Scale in
osm vnf-scale --scaling-group vdu_autoscale --scale-in manual_scaling_test vnf

# Terminate NS
osm ns-delete manual_scaling_test
NS instantiation and operation
Auto-healing
Auto-healing scenario

# Add packages
osm vnfpkg-create autoheal_vnf
osm nspkg-create autoheal_ns

# Launch NS
osm ns-create --ns_name heal_test --nsd_name autoheal_nsd --vim_account etsi-vim-slicesX --config '{vld: [ {name: mgmt, vim-network-name: osm-ext} ]}' --ssh_keys ~/.ssh/id_rsa.pub

# Force error in VM
openstack server set --state error <OPENSTACK_VM_ID>

# Wait for VM to be auto-healed
osm ns-op-list

# Terminate NS
osm ns-delete heal_test
Thank You!

osm.etsi.org
osm.etsi.org/docs/user-guide
osm.etsi.org/wikipub