OSM Hackfest – Guidelines for VNF builders

Adrián Candel (Altran)
Guillermo Calviño (Altran)
VNF configurations for lifecycle stages

• Basic Instantiation (Day 0): the VNF is instantiated and the management access is established.

• Service Initialization (Day 1): configure the VNF so it starts providing the expected service.

• Runtime Operations (Day 2): re-configure the VNF so its behaviour can be modified during runtime, as well as be able to monitor its main KPIs and run scaling actions over it.
VNF configurations for lifecycle stages

• Lifecycle stages

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")
Day 0 - Basic Instantiation

- Description of each VNF component
- Definition of NFVI requirements
  - Compute performance attributes:
    - CPU Pinning
    - NUMA Topology Awareness
    - Memory Page Size
  - Data plane performance attributes:
    - PCI-Passthrough
    - SR-IOV

More information: OSM Day 0 Guidelines
## Day 0 – Basic instantiation

<table>
<thead>
<tr>
<th>VNF name</th>
<th>VNF Description</th>
<th>VDU name</th>
<th>Image name</th>
<th>Flavor</th>
<th>Nº ifaces</th>
<th>GUEST EPA INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCPU</td>
<td>vMem</td>
<td>vDisk</td>
<td>Mempage size</td>
<td>Dedicated CPUs (YES/NO)</td>
<td>Cores or HW threads</td>
<td>Strict NUMA Mem Policy (YES/NO)</td>
</tr>
<tr>
<td>Dedicated</td>
<td>Core or HW threads</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This table is a reference table. Other parameters could be added, such as the existence of a cloud-init file for each VDU, if the VDU has a charm, etc.

More information: [OSM Day 0 Guidelines](#)
### vEPC VNF Day 0 - collected information Example

<table>
<thead>
<tr>
<th>VNF name</th>
<th>VNF Description</th>
<th>VDU name</th>
<th>Image name</th>
<th>Flavor</th>
<th>GUEST EPA INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vCPU</td>
<td>vMem</td>
</tr>
<tr>
<td>vEPC</td>
<td>Single VDU containing SGW, PGW and MME</td>
<td>spgwmme</td>
<td>nextepc-spgwmm e-base</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>vEPC</td>
<td>HSS VDU</td>
<td>hss</td>
<td>nextepc-hss-base</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Build your VNF diagram

VNF: <name>

ECP: <name>

VDU: <name>

Iface name

ICP: <name>

VL: <name>

ICP: <name>

VDU: <name>

Iface name

Iface name
Build your NS diagram

NS: <name>

VNF: <name>

VL: <name>

CP: <name>

VNF: <name>

VL: <name>

CP: <name>

CP: <name>
Day 0 - Basic Instantiation - configurations

• Minimal configuration of the VNFs can be injected via cloud-init
• Example:

```yaml
#cloud-config
hostname: my_first_vnf
password: osm4u
chpasswd: { expire: False }
ssh_pwauth: True
```

• Identifying the instantiation parameters
• Associate cloud-init-file in corresponding VDU

Cloud-init documentation [here](#)
The goal of Day 1 is the automatic initialization of VNF services right after the instantiation
Day 1 - Service Initialization

• Identifying dependencies between components
  • IP address for connectivity

• Defining the required configuration for service initialization
  • Start some interfaces
  • Replace values in configuration files
  • Start services inside the VNF

• Identifying the need for instantiation parameters
  • External endpoints to configure
Day 1 - Service Initialization

The process after identification comprises:

- Building a Proxy Charm
  - Method 1: Building a Proxy Charm the traditional way
  - Method 2: Using Proxy Charm Generators

More Information: Adding day 1 Primitives / VNF Onboarding Walkthrough
Day 1 - Service Initialization

Example:

**Collection of commands**

```
sudo ip link set ens4 up && sudo dhclient ens4
sudo ip link set ens5 up && sudo dhclient ens5
sudo ip link set ens6 up && sudo dhclient ens6
```

**Charm definition**

```
@when('actions.configure-spgw')
def configure_spgw():
    hss_ip = action_get('hss-ip')
    spgw_ip = action_get('spgw-ip')

    cmd1 = "sudo ip link set ens4 up && sudo dhclient ens4"
    cmd1 = charms.sshproxy.run(cmd1)

    cmd2 = "sudo ip link set ens5 up && sudo dhclient ens5"
    cmd2 = charms.sshproxy.run(cmd2)

    cmd3 = "sudo ip link set ens6 up && sudo dhclient ens6"
    cmd3 = charms.sshproxy.run(cmd3)

    cmd4 = "sudo sed -i \"s/$hss_ip/{}g\" /etc/nextepc/freeDiameter/mme.conf"
    cmd4 = charms.sshproxy.run(cmd4)
    remove_flag('actions.configure-spgw')
```
Day 1 - Service Initialization

Example:

Charm definition

```python
@when('actions.configure-spgw')
def configure_spgw():
    hss_ip = action_get('hss-ip')
    spgw_ip = action_get('spgw-ip')
    cmd1 = "sudo ip link set ens4 up && sudo dhcclient ens4"
    charms.sshproxy._run(cmd1)
    cmd2 = "sudo ip link set ens5 up && sudo dhcclient ens5"
    charms.sshproxy._run(cmd2)
    cmd3 = "sudo ip link set ens6 up && sudo dhcclient ens6"
    charms.sshproxy._run(cmd3)
    cmd3 = 'sudo sed -i "\s/$hss_ip/{}/g" /etc/nextepc/freeDiameter/mme.conf'.format(hss_ip)
    charms.sshproxy._run(cmd3)
    cmd4 = 'sudo sed -i "\s/$spgw_ip/{}/g" /etc/nextepc/freeDiameter/mme.conf'.format(spgw_ip)
    charms.sshproxy._run(cmd4)
    remove_flag('actions.configure-spgw')
```

Day 1 = initial-config-primitives

VNF Descriptor definition

```yaml
vnf-configuration:
  initial-config-primitive:
    seq: 1
    name: config
    parameter:
      - name: ssh-hostname
        value: crw_mgmt_ip>
      - name: ssh-username
        value: ubuntu
      - name: ssh-password
        value: <password>
    seq: 2
    name: configure-spgw
    parameter:
      - name: spgw-ip
        data-type: STRING
        value: <spgw_ip>
      - name: hss-ip
        data-type: STRING
        value: <hss_ip>
```
The goal of Day 2 is the reconfiguration of the services and service monitoring.
Day 2 – Runtime Operations

• Adding Day-2 primitives to the descriptor
  • Used to operate the service for example:
    • Clean a cache
    • Install a route
    • Restart a service
    • Create and restore a backup
Day 2 – Runtime Operations

• Example

Collection of commands

```
sudo route add -net $prefix gw $next_hop
```

Charm definition

```
@when('actions.add-route')
def add_route():
    prefix = action_get('external-prefix')
    next_hop = action_get('next-hop')
    cmd = 'sudo route add -net "' + prefix + '" gw "' + next_hop + '"

    charms.sshproxy.run(cmd)
    remove_flag('actions.add-route')
```

Day 2 = config-primitives

VNF Descriptor definition

```
vnf-configuration:
  config-primitive:
    name: add-route
    parameter:
      - name: external-prefix
data-type: STRING
default-value: '8.8.8.8/32'
      name: next-hop
data-type: STRING
default-value: '192.168.2.1'
```

More Information: VNF Onboarding Walkthrough
Day 2 – Runtime Operations

- Monitoring metrics definition
  - NFVI metrics
  - VNF Indicators - proxy charms with metrics layer
  - Scaling Operators

Example: nfvi metrics definition

```yaml
vdu:
...
  id: spgwmm
...
  monitoring-param:
    id: "spgw_cpu_util"
    nfvi-metric: "cpu_utilization"
    id: "spgw_memory_util"
    nfvi-metric: "average_memory_utilization"
...
  monitoring-param:
    id: "spgw_cpu_util"
    name: "spgw_cpu_util"
    aggregation-type: AVERAGE
    vdu-monitoring-param:
      vdu-ref: "spgwmm"
      vdu-monitoring-param-ref: "spgw_cpu_util"
    id: "spgw_memory_util"
    name: "spgw_memory_util"
    aggregation-type: AVERAGE
    vdu-monitoring-param:
      vdu-ref: "spgwmm"
      vdu-monitoring-param-ref: "spgw_memory_util"
```