Onboarding a Complex Network Function
Mark Beierl (Canonical)
Magma Components
Modelling a Network Service

VNF Name: srs-lte-enb-vnfd
- VDU: srsLTE
  - eth0
  - srsLTE-mgmt-ext

VNF Name: magma-agw-pnf
- VDU: Magma Access Gateway
  - eth1
  - vnf_internal
  - gateway_public

VNF Name: magma_orc_cnf
- VDU: Magma Orchestrator
  - mgmt
  - mgmtnet-ext

Management Network (mgmtnet)
S1 Network (mgmtnet)
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 - configurations

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

• Example:

```cloud-config
password: osm4u
chpasswd: { expire: False }
ssh_pwauth: True
```

• Identifying the instantiation parameters
  • Memory, CPU, number of instances, networking, etc

Cloud-init documentation [here](#)
Day 1 - Service Initialization

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

More Information: VNF Onboarding guidelines / VNF Onboarding Walkthrough
The goal of Day 2 is the reconfiguration of the services and service monitoring.
Day 2 - Runtime Operations

- 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

More Information: VNF Onboarding guidelines / VNF Onboarding Walkthrough
Model the srsLTE eNodeB
## 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>
</tr>
</thead>
<tbody>
<tr>
<td>srs-lte-enb_vnfd</td>
<td>An eNB simulator</td>
<td>srsLTE-vdu</td>
<td>ubuntu-focal-20.04-amd64-server-20220606</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 GB</td>
<td>15 GB</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](#)
VNF Descriptor: Sizing

vnfd:
  vdu:
    virtual-compute-desc: srsLTE-vdu-compute
    virtual-storage-desc:
      - srsLTE-vdu-storage
  description: srsLTE VDU
  version: 1.0
virtual-compute-desc:
- id: srsLTE-vdu-compute
  virtual-cpu:
    num-virtual-cpu: 2
  virtual-memory:
    size: 4.0
virtual-storage-desc:
- id: srsLTE-vdu-storage
  size-of-storage: 15
vnfd:
  vdu:
    description: srsLTE-vdu
    id: srsLTE-vdu
    name: srsLTE-vdu
    sw-image-desc: ubuntu20.04
    sw-image-desc:
      - id: ubuntu20.04
        image: ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-20220606
        name: ubuntu20.04
vnfd:
  vdu:
    - description: srsLTE-vdu
      id: srsLTE-vdu
  int-cpd:
    - id: eth0-int
      virtual-network-interface-requirement:
        - name: eth0
          virtual-interface:
            type: PARAVIRT
    - id: eth1-int
      int-virtual-link-desc: internalS1
      virtual-network-interface-requirement:
        - name: eth1
          virtual-interface:
            type: PARAVIRT
Network External Connection Points

vnfd:
  ext-cpd:
    - id: srsLTE-mgmt-ext
  int-cpd:
    cpd: eth0-int
    vdu-id: srsLTE-vdu
vnfd:
  vdu:
    - cloud-init-file: cloud-config
      description: srsLTE-vdu

#cloud-config
password: osm2022
chpasswd: { expire: False }
ssh_pwauth: True
packages:
  - net-tools
  - python-importlib
vnfd:
  df:
    - id: default-df
      lcm-operations-configuration:
        operate-vnf-op-config:
          day1-2:
            execution-environment-list:
              - id: srs-enb-ue-ee
                juju:
                  charm: srs-enb-ue
                  proxy: false
            initial-config-primitive:
              - name: config
                execution-environment-ref: srs-enb-ue-ee
                parameter:
                  - name: bind_address_subnet
                    value: <bind_address_subnet>
                  - name: mme_addr
                    value: <mme_addr>
                seq: 1
vnfd:
  df:
    - id: default-df
      lcm-operations-configuration:
        operate-vnf-op-config:
          day1-2:
            config-primitive:
              - name: attach-ue
                execution-environment-ref: srs-enb-ue-ee
                parameter:
                  - data-type: STRING
                    name: usim-imsi
                  - data-type: STRING
                    name: usim-k
                  - data-type: STRING
                    name: usim-opc
              - name: detach-ue
                execution-environment-ref: srs-enb-ue-ee
attach-ue:
  description: Attach User Emulator to enodeB
params:
  usim-imsi:
    description: "USIM IMSI"
    type: string
  usim-k:
    description: "USIM K"
    type: string
  usim-opc:
    description: "USIM OPC"
    type: string
required:
  - usim-imsi
  - usim-k
  - usim-opc
detach-ue:
  description: "Detach from AGW."
remove-default-gw:
  description: "Remove default gateway"
def _on_attach_ue_action(self, event):
    self._stored.ue_usim_imsi = event.params['usim-imsi']
    self._stored.ue_usim_k = event.params['usim-k']
    self._stored.ue_usim_opc = event.params['usim-opc']
    self._configure_srsue_service()
    service_restart(SRS_UE_SERVICE)
    self._stored.ue_attached = True
    self.unit.status = self._get_current_status()
    event.set_results({"status": "ok", "message": "Attached successfully"})
Model the Magma Orchestrator
Magma Orchestrator

Model
Kubernetes, SOL006, Helm and Juju

- SOL006 defines standard model
- Competing model descriptors for Cloud Native
  - Helm
  - Juju
- ETSI recognizes additional models
  - Descriptor can make reference to a Juju bundle, or Helm chart
- OSM manages bundle or chart as single entity
vnfd:
  product-name: magma_orc_cnf
  version: "1.0"
  provider: Canonical
  description: "K8s container deployment of Magma Orchestrator"
  id: magma_orc_cnf
  kdu:
    - name: magma-orc-kdu
      juju-bundle: bundle.yaml
Open Source MANO

Q&A
Operating “proxy” workloads

Where can we run our own operations code for this workload?
Operating “native” workloads

Workloads optimised for OSM have a charm that drives the workload directly.
OSM VM (Ubuntu 20.04, Microk8s, Juju, LXD)

srsLTE

Magma AGW (PNF)

Proxy Charms

Native Charm