Multi-Vendor VNF Onboarding Using ONAP

Nokia – Vara Talari & Timo Perala
ONAP Casablanca Architecture
(High-Level View with Projects)
Nokia’s ONAP Participation/Contribution

• Nokia among the founding platinum members of ONAP
• Nokia interest in ONAP is multifaceted
  • VNF and PNF vendor
  • Automation, Orchestration and Management Solution provider
  • Services business, e.g. integration, Multivendor VNF deployment (Radisys, Clear water, Nokia etc)
• Nokia main focus areas to date in ONAP
  • 5G use case/ PNF deployment
  • BBS use case
  • Alignment with established standards

ONAP contributors, as of April 3 2019
VNF Onboarding on ONAP

Process overview

1. Gather requirements
2. Create HLD & LLD
3. Create VLM
4. Create VSP
5. Create VF
6. Create Service
7. Test VF
8. Approve VF
9. Test Service
10. Approve Service
11. Approve Service for distribution
12. distribute the service
13. Create Service Instance & Virtual Function
14. Preload the VNF topology
15. SDN-C
16. Create/launch VF-Modules
17. Instantiation
Landscape of modeling in ONAP
Different DMs are used for VNF templates. ECOMP uses HEAT, OPEN-O uses TOSCA. We will be working together on supporting HEAT, TOSCA and YANG VNF modeling.

The core implementation engine for LCM and operation in run time should be data model agnostic.
VF-C architecture overview
VNF TOSCA Template Requirements for ONAP

- VNF TOSCA template is aligned to the newest TOSCA protocol, “Working Draft 04-Revision 06”.
- VNF TOSCA template supports HPA features, such as NUMA, Hyper Threading, SRIOV, etc.

- In the ONAP, VNF Package and VNFD template can be designed by manually or via model designer tools.
- The VNF package structure is align to the NFV TOSCA protocol, and supports CSAR
- The VNFD and VNF package are all align to the NFV TOSCA protocol, which supports multiple TOSCA template yaml files, and also supports self-defined node or other extensions.

A Topology Template consists of a set of Node Templates and Relationship Templates that together define the topology model of a service as a (not necessarily connected) directed graph. A node in this graph is represented by a Node Template.

Reference: https://onap.readthedocs.io/en/beijing/submodules/vnfrqts/requirements.git/docs/Chapter5.html
VNF TOSCA Template Requirements for ONAP

- VNF topology: it is modeled in a cloud agnostic way using virtualized containers and their connectivity. Virtual Deployment Units (VDU) describe the capabilities of the virtualized containers, such as virtual CPU, RAM, disks; their connectivity is modeled with VDU Connection Point Descriptors (VduCpd), Virtual Link Descriptors (Vld) and VNF External Connection Point Descriptors (VnfExternalCpd);

- VNF deployment aspects: they are described in one or more deployment flavours, including instantiation levels, supported LCM operations, VNF LCM operation configuration parameters, placement constraints (affinity / anti-affinity), minimum and maximum VDU instance numbers, and Scaling aspect for horizontal scaling.

- SR-IOV Passthrough: Definitions of SRIOV_Port are necessary if VDU supports SR-IOV.

- Hugepages: Definitions of mem_page_size as one property shall be added to Properties and set the value to large if one VDU node supports hugepages.

- NUMA (CPU/Mem): We add definitions of numa to requested_additional_capabilities if we want VUD nodes to support NUMA.

- Hyper-Threading: Definitions of Hyper-Threading are necessary as one of requested_additional_capabilities of one VUD node if that node supports Hyper-Threading.

- OVS+DPDK: Definitions of ovs_dpdk are necessary as one of requested_additional_capabilities of one VUD node if that node supports dpdk.

**tosca.nodes.nfv.VDU.Compute**
The NFV Virtualization Deployment Unit (VDU) compute node type represents a VDU entity which it describes the deployment and operational behavior of a VNF component (VNFC)

**tosca.nodes.nfv.VDU.VirtualStorage**
The NFV VirtualStorage node type represents a virtual storage entity which it describes the deployment and operational behavior of a virtual storage resources

Reference: https://onap.readthedocs.io/en/beijing/submodules/vnfrqts/requirements.git/docs/Chapter5.html
TOSCA Template updates for VNF deployment via VF-C /CBAM (VNFM)

1. The VNF must declare the externalVnfmId and onapCsarId as modifyable attribute in CBAM package. Each should have a default value. (The concrete value will be filled out by CBAM)
2. Each operation must declare a jobId additional parameter in CBAM package (value will be filled out by CBAM)
3. The heal operation must declare the jobId, vmName, vnfcId and action parameters in CBAM package (values will be filled out by CBAM)
4. Each operation (including built-in) must include the following section as the last pre_action (all JS are provided by CBAM)

```javascript
javascript: javascript/cbam.pre.collectConnectionPoints.js include: - javascript/cbam.collectConnectionPoints.js
output: operation_result
```

CBAM supplied JavaScripts
- cbam.post.collectConnectionPoints.js
- cbam.pre.collectConnectionPoints.js
- cbam.collectConnectionPoints.js

Reference:
TOSCA Template updates for VNF deployment via VFC

The ONAP package must be written so that the VDU.Compute, VDU.VirtualStorage, VnfVirtualLinkDesc, VduCpd has exactly the same name as in CBAM package
the metadata section of the ONAP package must be the following:

- the vendor must be the same as in Nokia package vendor field
- the vnfdVersion must be the same as in Nokia package the descriptor_version field
- the name must be the same as in Nokia package the product_info_name field
- the version must be the same as in Nokia package the software_version field
- the vnfmType must be NokiaSVNFM

The complete CBAM package must be placed in the in Artifacts/OTHER/cbam.package.zip file

**Convertor:**
Visit [http://msb.api.simpledemo.onap.org/api/NokiaSVNFM/v1/convert](http://msb.api.simpledemo.onap.org/api/NokiaSVNFM/v1/convert)
Select the CBAM package to be converted into an ONAP package
Click on upload button and the ONAP package will be downloaded

ONAP VNF package

- Images are not handled by ONAP
  - Manually uploaded to the cloud
  - Linked manually to VNFC
- VNFD embedded in ONAP CSAR
  - Certain elements are duplicated
CBAM – ONAP integration high level architecture: component interaction

1. Publish it’s existence & APIs
2. Upload images
3. Trigger operation via portal
4. Trigger operation via VF-C NBI
5. Download package
6. Create cross VNF resources
7. Search for Nokia VNFM
8. Trigger operation & track progress via S-VNF API
9. Trigger operation & track progress via CBAM API
10. Allocate resources, (re)configure VNF
11. Send LCN containing structural changes
12. Modify VNF structure
VNF deployment through ONAP

Heat Template Deployment
Static HOT
Basic VNFD with static HOT
Parametrized VNF
Ansible
Change HOT to rigid vs scalable aspects
Update VNFD and test scaling

ONAP VNF Requirements:
https://onap.readthedocs.io/en/beijing/submodules/vnfrqts/requirements.git/docs/Chapter5.html
Customer Specific Templates to ONAP

- **Vendor Specific Template**
  - Modified to heat template as per ONAP requirements

- **Validation in Open Stack**
  - Validated Heat template in Openstack
  - Corrected all the errors

- **Final Heat template**
  - Deployed in ONAP (VSP)
Updates to Templates for ONAP Compliance

- **Resource Registry**
  - Removed vendor specific resource registry

- **Naming Convention**
  - Follow Naming convention

- **Additional Updates**
  - ResourceGroup does not deal with structured parameters (comma-delimited-list and json) – We replaced json with yaml format.
  - Volumes created during runtime
  - Internal Networks created during runtime
  - External Networks pre-created and specified in env file

- **Nested config**
  - Removed Nested config
  - Nested Heat is supported by ONAP*

- **User Data**
  - Updated User data- Meta Data

- **Scale-In/Scale – Out and other LCMs**
  - Currently removed, Handled seperately
Example of resource group

Example updates

**Example of resource group**
type: OS::Heat::ResourceGroup
LB1:
  type: ALU::VMG::LB
  the above one was for resource registry
MG1:
  type: ALU::VMG::MG
OAMA:
  type: ALU::VMG::OAM
resource_registry:
  ALU::VMG::OAM: VMG_OAM.template.yaml
  ALU::VMG::LB: VMG_LB.template.yaml
  ALU::VMG::MG: VMG_MG.template.yaml
CBAM - OAMA:
  type: ALU::VMG::OAM

To ONAP compliance

oam_server_0:
  type: OS::Nova::Serve
ONAP mg_work_server_0:
  type: OS::Nova::Server

Run time Volume creation

dependent_volume:
  type: OS::Cinder::Volume
  properties:
    name: { list_join: [ "-", get_param: [ cbam, vnfId ] }, OAM-A-CompactFlash1 ]

size:
  get_param: [ cbam, extensions, oamCompactFlash1Size ]
oam_comp_flash_01:
  type: OS::Cinder::Volume
  name:
    list_join: [ "-", get_param: vnf_id, OAM, get_param: oamSlotId_0, CompactFlash1 ]
  size: 2

User Data:

user_data:
  str_replace:
    template: "$oamSmbios"
    params:
      $oamSmbios: get_param: [ resources, smbios ]
  user_data_format: "RAW"

user_data:
  str_replace:
    template: "$smbios1"
    params:
      smbios1: get_param: smbios_1
  user_data_format: "RAW"

Naming Convention:

oamACompactFlash1:
  type: OS::Cinder::Volume
  properties:
    name:
      list_join: [ "-", [ get_param: [ cbam, vnfId ] }, OAM-A-CompactFlash1 ]

<table>
<thead>
<tr>
<th>Nokia</th>
<th>vHSS</th>
<th>vCSCF</th>
<th>vMME</th>
<th>vSAE-GW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radisys</td>
<td>vMRF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clearwater</td>
<td>vIMS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Service instance

<table>
<thead>
<tr>
<th>Action</th>
<th>Service Name</th>
<th>Service Invariant UUID</th>
<th>Service Version</th>
<th>Category</th>
<th>Distribution Status</th>
<th>Last Updated</th>
<th>Tenure Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy</td>
<td>Service</td>
<td>49b1f4348a-7447-405e-8200-1b0227f472de</td>
<td>1.0</td>
<td>Network</td>
<td>DISTRIBUTED</td>
<td>2017-09-20 14:52:18</td>
<td></td>
</tr>
<tr>
<td>Deploy</td>
<td>Service</td>
<td>e2209d-3a93-4b52-9207-9520d0076a34</td>
<td>1.0</td>
<td>Network</td>
<td>DISTRIBUTED</td>
<td>2017-09-20 14:52:18</td>
<td></td>
</tr>
<tr>
<td>Deploy</td>
<td>Service</td>
<td>1271814-7a35-4850-8c00-3d65197f46e3</td>
<td>1.0</td>
<td>Network</td>
<td>DISTRIBUTED</td>
<td>2017-09-20 14:52:18</td>
<td></td>
</tr>
</tbody>
</table>

**Status:** COMPLETE  
Service Instance has been created successfully.

Enter Data and Confirm to Create Service Instance.  
Cancel to Return to Previous Page.  
Data entered will be lost.
VF Deployment

Create VF Module

Service Name: Service
Subscriber Name: Demonstration
Service Instance Name: Demonstration
Model Name: Vsp.pago.vw.module-0
Model Invariant UUID: 0f8ehG98-80c9-460a-8fbb-8e54c44c58d
Model Version: 1
Model UUID: 8acc7a03-1542-4678-9c00-4070686e6e27

User Provided Data (* indicates required field)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Name</td>
<td>DemoVInstance</td>
</tr>
<tr>
<td>LCP Region</td>
<td>Nat</td>
</tr>
<tr>
<td>Tenant</td>
<td>1035199</td>
</tr>
<tr>
<td>Suppress Rollback on</td>
<td></td>
</tr>
<tr>
<td>Failure</td>
<td></td>
</tr>
</tbody>
</table>

Enter data and confirm to
Create VF Module
Cancel to return to previous page.
Data entered will be lost

Create Cancel
VF Deployment
Topology Pre-Load
Post Script
Topology Pre-load
Post Script


X-FromAppId: API client

Authorization: Basic YWRtaW46Y29tOm1hZGluZXJoZWFydaleWb2FkZWEzZjJh
Accept: application/json

X-TransactionId: 0a3f6713-ba96-4971-a6f8-c2da85a3176e

Cache-Control: no-cache

Postman-Token: e1c8d1ec-4cd9-5744-3ac9-f83f0d3c71d4

Content-Type: application/json

{
    "input": {
        "vnf-topology-information": {
            "vnf-topology-identifier": {
                "service-type": "3daf586c-af0a-42b3-8a29-6853bed70186",
                "vnf-name": "DemoModule3",
                "vnf-type": "Vsp..base_vfw..module-0",
                "generic-vnf-name": "DemoVNF",
                "generic-vnf-type": "bd4768a0-90ec-4c3b-b9fd 0"
            }
        },
        "vnf-assignments": {
            "availability-zones": [],
            "vnf-networks": [],
            "vnf-vms": []
        }
    }
}
## Updates to Templates for ONAP Compliance

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resource Registry</strong></td>
<td>• Removed vendor specific resource registry</td>
</tr>
<tr>
<td><strong>Naming Convention</strong></td>
<td>• Follow Naming convention</td>
</tr>
<tr>
<td><strong>Additional Updates</strong></td>
<td>• ResourceGroup does not deal with structured parameters (comma-delimited-list and json) – We replaced json with yaml format.</td>
</tr>
<tr>
<td></td>
<td>• Volumes created during runtime</td>
</tr>
<tr>
<td></td>
<td>• Internal Networks created during runtime</td>
</tr>
<tr>
<td></td>
<td>• External Networks pre-created and specified in env file</td>
</tr>
<tr>
<td><strong>Nested config</strong></td>
<td>• Removed Nested config</td>
</tr>
<tr>
<td></td>
<td>• Nested Heat is supported by ONAP*</td>
</tr>
<tr>
<td><strong>User Data</strong></td>
<td>• Updated User data - Meta Data</td>
</tr>
<tr>
<td><strong>Scale-In/Scale-Out and other LCMs</strong></td>
<td>• Currently removed, Handled separately</td>
</tr>
</tbody>
</table>
Thank you

Please let us know if you have questions