ONAP: Application Service Descriptor (ASD) for K8s NFs

Marian Darula, Byung-Woo Jun, Zu Qiang (Ericsson)
Thinh Nguyenphu, Joe Schumacher (Nokia)
Application Service Descriptor (ASD)

- **Motivation**
  - ASD is a common, simplified deployment descriptor for ONAP, O-RAN (NFs, xApps and rApps) aiming to:
    - Quickly leverage enhancements in Kubernetes while minimizing development and integration efforts
    - Avoid duplication of attributes/properties included in cloud native artifacts, e.g., Helm Charts
    - Descriptor format is not requiring a particular deployment tool (e.g., Helm)
    - Leverage established packaging standards (e.g., SOL004 with ASD as a top-level deployment artifact)

- **Status**
  - ONAP: The proposal is being reviewed at ONAP Modelling Subcommittee
    - ASD onboarding information model (IM)
    - ASD Resource Data Model
    - ASD Onboarding Packaging Format
    - Based on above proposal ASD and Application Onboarding and LCM Orchestration PoC is ongoing for ONAP Jakarta release
  - O-RAN: A technical proposal has been submitted to WG10 and WG6.
The proposed solution:

1. Use CSAR packaging for bundling metadata and cloud-native artifacts in a single package. Describe the application using a lightweight Application Service Descriptor.

2. Choose Helm v3 as the initial cloud-native tool to embed in the orchestrator.
1. A deployment order is received, along with the required lifecycleParameters values.
2. The cloud-native deployment tool is invoked with the received parameters to transform the cloud artifacts into K8S resource descriptions.
3. The K8S resource descriptions, ASD and any other relevant data is sent to the placement function.
4. Placement decision is done based on input data.
5. Inform deployment of placement.
6. Request the cloud native deployment tool to deploy on the identified target cluster.
7. Cloud native deployment tool deploys application in the chosen cluster using the K8S API.
## Application Service Descriptor: Top Level

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asdId</td>
<td>M</td>
<td>1</td>
<td>Identifier</td>
<td>Identifier of this ASD information element. This attribute shall be globally unique. The format will be defined in the data model specification phase.</td>
</tr>
<tr>
<td>asdSchemaVersion</td>
<td>M</td>
<td>1</td>
<td>Version</td>
<td>Specifies the version of the ASD’s schema (if we modify an ASD field definition, add/remove field definitions, etc.)</td>
</tr>
<tr>
<td>asdProvider</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Provider of the AS and of the ASD.</td>
</tr>
<tr>
<td>asdApplicationName</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Name to identify the Application Service. Invariant for the AS lifetime.</td>
</tr>
<tr>
<td>asdApplicationVersion</td>
<td>M</td>
<td>1</td>
<td>Version</td>
<td>Specifies the version of the Application (so, if software, deploymentItems, ASD values, ... change, this changes)</td>
</tr>
<tr>
<td>asdApplicationInfoName</td>
<td>M</td>
<td>0..1</td>
<td>String</td>
<td>Human readable name for the Application service. Can change during the AS lifetime.</td>
</tr>
<tr>
<td>asdInfoDescription</td>
<td>M</td>
<td>0..1</td>
<td>String</td>
<td>Human readable description of the AS. Can change during the AS lifetime.</td>
</tr>
<tr>
<td>asdExtCpd</td>
<td>M</td>
<td>0..N</td>
<td>datatype.ExtCpd</td>
<td>Describes the externally exposed connection points of the application.</td>
</tr>
<tr>
<td>enhancedClusterCapabilities</td>
<td>M</td>
<td>0..1</td>
<td>datatype. enhancedClusterCapabilities</td>
<td>A list of expected capabilities of the target Kubernetes cluster to aid placement of the application service on a suitable cluster.</td>
</tr>
<tr>
<td>deploymentItems</td>
<td>M</td>
<td>1..N</td>
<td>DeploymentItem</td>
<td>Deployment artifacts</td>
</tr>
</tbody>
</table>
## ASD: asdExtCpd

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The identifier of this extCpdData</td>
</tr>
<tr>
<td>description</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Describes the service exposed.</td>
</tr>
<tr>
<td>virtualLinkRequirement</td>
<td>M</td>
<td>1..N</td>
<td>String</td>
<td>Refers in an abstract way to the network or multiple networks that the ExtCpd shall be exposed on (ex: OAM, EndUser, backhaul, LI, etc). The intent is to enable a network operator to take decision on to which actual VPN to connect the extCpd to. Note 1.</td>
</tr>
<tr>
<td>networkInterfaceRealization</td>
<td>M</td>
<td>0..1</td>
<td>datatype.networkInterface RealizationRequirements</td>
<td>Details container implementation specific requirements on the NetworkAttachmentDefinition. See Note 2 &amp; 3.</td>
</tr>
<tr>
<td>inputParamMappings</td>
<td>M</td>
<td>0..1</td>
<td>Datatype.extCpd.ParamMappings</td>
<td>Information on what parameters that are required to be provided to the deployment tools for the asdExtCpd instance.</td>
</tr>
<tr>
<td>resourceMapping</td>
<td>M</td>
<td>0..1</td>
<td>String</td>
<td>Kubernetes API resource name for the resource manifest for the service, ingress or pod resource declaring the network interface. Enables, together with knowledge on namespace, the orchestrator to lookup the runtime data related to the extCpd.</td>
</tr>
</tbody>
</table>

Note 1: Corresponds more or less to a virtual_link requirement in ETSI NFV SOL001.  
Note 2: Applies only for ExtCpds representing secondary network interfaces in a pod.  
Note 3: Several ExtCpds may refer to same additional network interface requirements.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trunkMode</td>
<td>M</td>
<td>0..1</td>
<td>&quot;false&quot;</td>
<td>&quot;true&quot;</td>
</tr>
<tr>
<td>ipam</td>
<td>M</td>
<td>0..1</td>
<td>&quot;infraProvided&quot;, &quot;orchestrated&quot;, &quot;userManaged&quot;</td>
<td>The default value (&quot;infraProvided&quot;) means that the CNI specifies how IPAM is done and assigns the IP address to the pod interface.</td>
</tr>
<tr>
<td>interfaceType</td>
<td>M</td>
<td>0..1</td>
<td>&quot;kernel.netdev&quot;, &quot;direct.userdriver&quot;, &quot;direct.kerneldriver&quot;, &quot;direct.bond&quot;, &quot;userspace&quot;</td>
<td>This attribute is applicable for passthrough and memif interfaces. Value default value is &quot;kernel.netdev&quot;.</td>
</tr>
<tr>
<td>interfaceOptions</td>
<td>M</td>
<td>0..N</td>
<td>&quot;virtio&quot;, &quot;memif&quot;</td>
<td>Alternative vNIC configurations the network interface is verified to work with.</td>
</tr>
<tr>
<td>interfaceRedundancy</td>
<td>M</td>
<td>0..1</td>
<td>&quot;infraProvided&quot;, &quot;activePassiveBond&quot;, &quot;activeActiveBond&quot;, &quot;activePassiveL3&quot;, &quot;activeActiveL3&quot;,</td>
<td>&quot;infraProvided” means that the application sees one vNIC but that the infrastruture provides redundant access to the network via both switch planes. &quot;Left&quot; and “right” indicates a vNIC connected non-redundantly to the network via one specific (left or right) switchplane. All other attributes indicates a mated vNIC pair in the Pod, one connecting to the network via left switchplane and the other connecting to the network via the right switchplane, and with application using them together as a redundant network interface using a particular redundancy method that need to be accomodated in the node infrastructure.</td>
</tr>
<tr>
<td>nicOptions</td>
<td>M</td>
<td>0..N</td>
<td>&quot;examples&quot;: [&quot;i710&quot;, &quot;mlx-cx5v&quot;]</td>
<td>nics a direct user space driver the application is verified to work with. Allowed values from ETSI registry.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Qualifier</td>
<td>Cardinality</td>
<td>Content</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>loadbalancerIP</td>
<td>M</td>
<td>0..1</td>
<td>String</td>
<td>When present, this attribute specifies the name of the deployment artifact input parameter through which the orchestrator can configure the loadbalancerIP parameter of the K8s service or ingress controller that the ExtCpd represents.</td>
</tr>
<tr>
<td>externalIPs</td>
<td>M</td>
<td>0..N</td>
<td>String</td>
<td>When present, this attribute specifies the name of the deployment artifact input parameter through which the orchestrator can configure the externalIPs parameter of the K8s service or ingress controller, or the pod network interface annotation, that the ExtCpd represents. The param name and provided IP address(es) value will be passed to the deployment tool when deploying the DeploymentArtifacts. Note 1</td>
</tr>
<tr>
<td>nadName</td>
<td>M</td>
<td>0..N</td>
<td>String</td>
<td>These attributes specifies, for an ExtCpd respesenting a secondary network interface, the name(s) of the network attachment definitions (NADs) the orchestrator has created as base for the network interface the ExtCpd represents. It is expected that the NADs themselves have been created prior to the deployment of the deployment artifacts. Note 2, Note 3</td>
</tr>
<tr>
<td>nadNamespace</td>
<td>M</td>
<td>0..1</td>
<td>String</td>
<td>Specifies, for an asdExtCpd respresenting a secondary network interface, the namespace where the NADs are located. Attribute may be omitted if the namespace is same as the application namespace.</td>
</tr>
</tbody>
</table>

Note 1: When the asdExt Cpd represent a networkRedundant/mated-pair of sriov interfaces, there are references to 2 or 3 related NADs needed to be passed, while for other interface types only one NAD reference is needed to be passed.

Note 2: The format of the Content strings is specific for each different orchestration templating technology used (Helm, Teraform, etc.). Currently only a format for use with Helm charts is suggested: "helmchartname:[subchartname.]0..N[parentparamname.] 0..Nparametername". Whether the optional parts of the format are present depends on how the parameter is declared in the helm chart.

Note 3: A direct attached (passthrough) network interface, such as an sriov interface, attaches to a network via only one of the two switch planes in the infrastructure.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minKernelVersion</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Describes the minimal required Kernel version, e.g. 4.15.0. Coded as displayed by linux command <code>uname -r</code></td>
</tr>
<tr>
<td>requiredKernelModules</td>
<td>M</td>
<td>0..N</td>
<td>String</td>
<td>Required kernel modules are coded as listed by linux <code>lsmod</code> command, e.g. <code>ip6_tables</code>, <code>cryptd</code>, <code>nf_nat</code> etc.</td>
</tr>
<tr>
<td>conflictingKernelModules</td>
<td>M</td>
<td>0..N</td>
<td>String</td>
<td>Kernel modules, which must not be present in the target environment. The kernel modules are coded as listed by linux <code>lsmod</code> command, e.g. <code>ip6_tables</code>, <code>cryptd</code>, <code>nf_nat</code> etc. Example: Linux kernel SCTP module, which may conflict with use of proprietary user space SCTP stack provided by the application.</td>
</tr>
<tr>
<td>clusterLabels</td>
<td>M</td>
<td>0..N</td>
<td>String</td>
<td>This attribute allows to associate arbitrary labels to clusters. These can indicate special infrastructure capabilities (e.g., NW acceleration, GPU compute, etc.). The intent of these labels is to serve as a set of values that can help in application placement decisions. clusterLabels follow the Kubernetes label key-value-nomenclature (<a href="https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/">https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/</a>). It is recommended that labels follow a standardised meaning e.g. for node features (<a href="https://kubernetes-sigs.github.io/node-feature-discovery/v0.9/get-started/features.html#table-of-contents">https://kubernetes-sigs.github.io/node-feature-discovery/v0.9/get-started/features.html#table-of-contents</a>). Example: ClusterLabels - <code>feature.node.kubernetes.io/cpu-puid.AESNI</code>: true</td>
</tr>
<tr>
<td>requiredPlugin</td>
<td>M</td>
<td>0..N</td>
<td>Structure (inlined)</td>
<td>A list of the names and versions of the required K8s plugin (e.g. multus v3.8)</td>
</tr>
<tr>
<td>&gt;requiredPluginName</td>
<td>M</td>
<td>0..1</td>
<td>String</td>
<td>The names of the required K8s plugin (e.g. multus)</td>
</tr>
<tr>
<td>&gt;requiredPluginVersion</td>
<td>M</td>
<td>0..1</td>
<td>String</td>
<td>The version of the required plugin (e.g. 3.8)</td>
</tr>
</tbody>
</table>
### ASD: deploymentItems

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deploymentItemId</td>
<td>M</td>
<td>1</td>
<td>Identifier</td>
<td>The identifier of this deployment item</td>
</tr>
<tr>
<td>artifactType</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Specifies the artifact type. One of following values can be chosen: &quot;helm_chart&quot;, &quot;helmfile&quot;, &quot;crd&quot;, &quot;terraform&quot;</td>
</tr>
<tr>
<td>artifactId</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Reference to a DeploymentArtifact. It can refer to URI or file path.</td>
</tr>
<tr>
<td>deploymentOrder</td>
<td>M</td>
<td>0..1</td>
<td>Integer</td>
<td>Specifies the deployment stage that the DeploymentArtifact belongs to. A lower value specifies that the DeploymentArtifact belongs to an earlier deployment stage, i.e. needs to be installed prior to DeploymentArtifact with higher deploymentOrder values. If not specified, the deployment of the DeploymentArtifact can be done in arbitrary order and decided by the orchestrator.</td>
</tr>
<tr>
<td>lifecycleParameters</td>
<td>M</td>
<td>0..N</td>
<td>String</td>
<td>The list of parameters that can be overridden at deployment time (e.g., the list of parameters in the values.yaml which can be overridden at deployment time)</td>
</tr>
</tbody>
</table>

In order to support complex applications that require multiple artifacts (like Helm charts) to be installed in a particular order, the orchestrator must support an easy method of chaining these artifacts – including dependency relationships.

As shown, items are given a deployment order. Items with the same order are deployed in parallel; items with different orders are deployed in sequence.
ASD Resource Data Model

- The initial **ASD Resource Data Model** proposal is based on TOSCA language specification
  - TOSCA parser available in ONAP
  - Use minimal set of TOSCA language capabilities defining ASD schema
  - Defining **TOSCA node types of ASD**
  - Data model resembling “plain yaml”
  - To be used in the ongoing PoC, the final names for node types can be selected as a part of standardization.
**ASD node_template example**

```yaml
node_templates:
  applicationServiceDescriptor:
    type: tosca.nodes.asd
    description: "Sample Application"
    properties:
      descriptor_id: fdsa-3211-sdfs-dwqqe
      descriptor_invariant_id: dsdx-2324-wers-3234
      version: 1.0
      schema_version: 2.0
      provider: MyCompany
      application_name: SampleApp
      application_version: 2.3
      application_info_name: Sample Application
    extCpds:
      - id: 1
        description: webpage-service
        virtual_link_requirement: endUser
      - id: 2
        description: transactionAPI
        virtual_link_requirement: backhaul
    enhanced_cluster_capabilities: [ ... ]
    artifacts:
      # these are the deployment items:
      sampleapp-db:
        type: tosca.artifacts.asd.deploymentItem
        file: "sampleapp-db-operator-helm.tgz" # or a URI
        properties:
          artifact_type: "helm_chart"
          itemId: 1
          deployment_order: 1
          lifecycle_parameters:
            - "Values.db.fullBackupInterval"
            - "Values.db.walConsolidationInterval"

    tosca.nodes.asd:
      derived_from: tosca.nodes.Root
      description: "The ASD node type"
      version: 0.1
      properties:
        descriptor_id:
          type: string # UUID
          required: true
          description: Identifier of this ASD.
        descriptor_invariant_id:
          type: string # UUID
          required: true
          description: Identifier of this descriptor in a version independent manner.
          It is in UUID format as specified in RFC 4122
        version:
          type: string
          description: Version of the ASD node type.
          required: true
          default: 0.1
      properties:
        virtual_link_properties:
          virtual_link_properties:
            virtual_link_properties:
            virtual_link_properties:
```

**ASD TOSCA service template**

ASD type definition

- A list of expected capabilities of the target Kubernetes cluster to aid placement of the application service on a suitable cluster.
In order to facilitate compatibility with ETSI, ONAP and other telco standards, the CSAR (NFV SOL 004ed421) packaging format is used with following details:

- The structure and format of an ASD package shall conform to the TOSCA Simple Profile YAML v1.1 Specification of the CSAR format. The zip file format shall conform to Document Container Format File.
- CSAR format with TOSCA-Metadata directory, specified in ETSI NFV SOL004ed431 section 4.1.2, with the differences that the following TOSCA.meta file keynames extensions are optional:
  - ETSI-Entry-Change-Log
  - ETSI-Entry-Tests
  - ETSI-Entry-Licenses
- Non-MANO artifact sets, specified in ETSI NFV SOL004ed431 section 4.3.7
- Registered non-MANO artifact keywords can be reused, to avoid duplication
- Package and artifacts security, specified in ETSI NFV SOL004 ed431 section 5 and 4.3.6
- Package manifest file, specified in ETSI NFV SOL004ed431 section 4.3.2, with new manifest metadata proposed in the wiki page
- Additional non-mano-artifact keywords for 5G use cases.
An example of an ASD onboarding package

This is one of the possibilities to introduce ASD type, since ASD metadata keywords are not defined in any STO yet.
ASD Link to network service modelling

• ASD has one K8S cluster scope
• Link to network level modelling
• One option is to use ETSI MANO NSD
  – To avoid any impacts on ETSI NSD standarization and current ONAP implementation
  – A new node type is proposed: asdInNsd
  – The approach can be used for early PoC
• ASD allows to integrate with other options for network service modelling, the solutions are under discussion.
The NSD must associate the virtual link requirements to the correct NS Virtual Link nodes for the networks that each extCpd shall connect to.
Application Service Descriptor - Summary

- Application Service Descriptor (ASD) provides simplified way of modelling and packaging of NFs
  - It’s an alternative to complex ETSI MANO based approach.
  - Relies on cloud native modeling tools (e.g. helm), complemented by slim descriptor layer providing information which cannot be conveyed via native modeling tools (e.g. networking related information)
  - Not repeating information from the native tools.
  - Utilizing established standards where applicable (e.g. TOSCA for the schema definition, the CSAR structure based on SOL004ed431 format).

- PoC ongoing to proof the concept in ONAP environment

[LINK]