K8S Application Service Descriptor and Packaging

ONAP CNF TF and Modeling Sub-Committee and ETSI NFV workshop (information sharing)

Nokia: Thinh Nguyenphu, Timo Perala,
Ericsson: Marian Darula, Byung-Woo Jun, Zu Qiang

2022-02-22

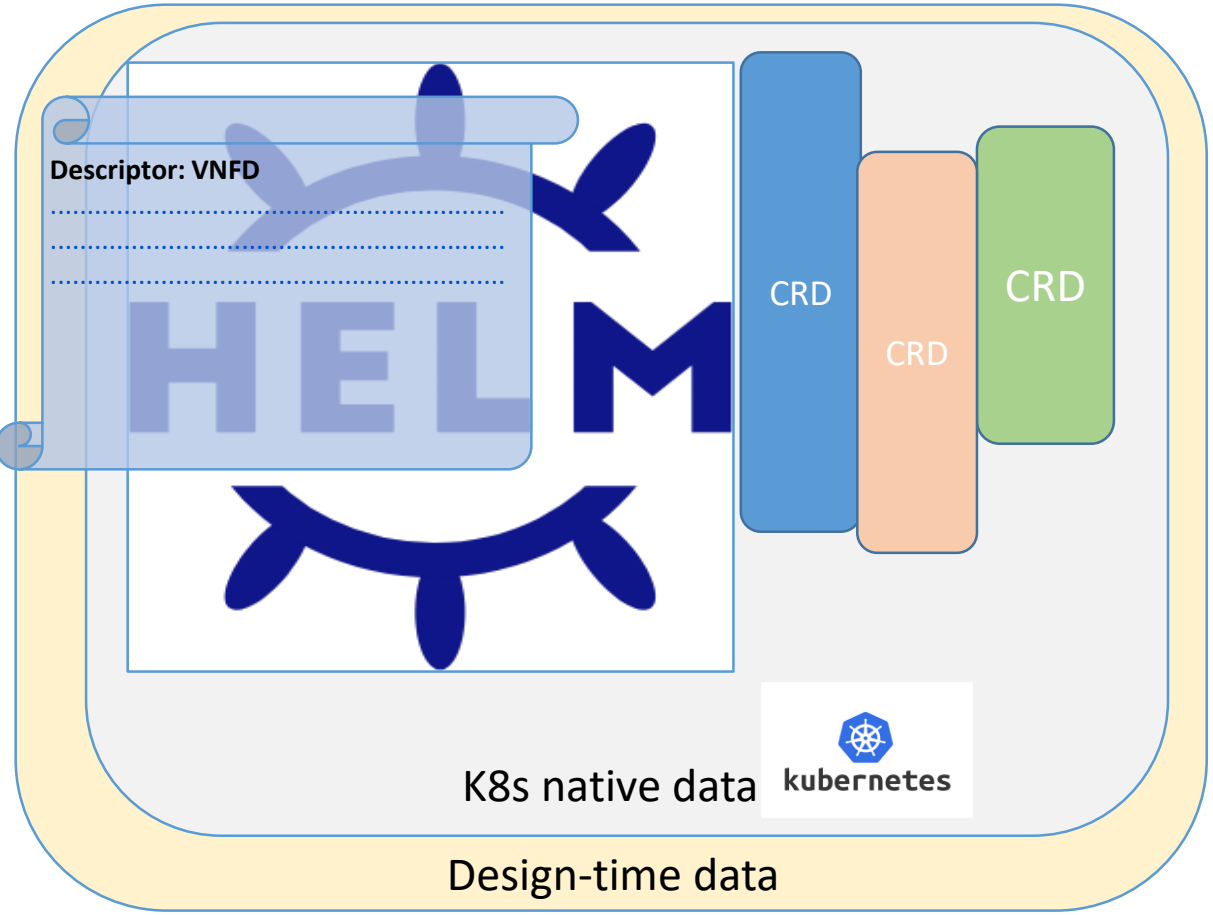
Contents

- Part 1: Why ASD
- Part 2 : Overviews, status, and plans
- Part 3: Application Service Descriptor (ASD)
- Part 4: ASD Packaging
- Part 5: ONAP PoC ASD

Part 1: Why ASD

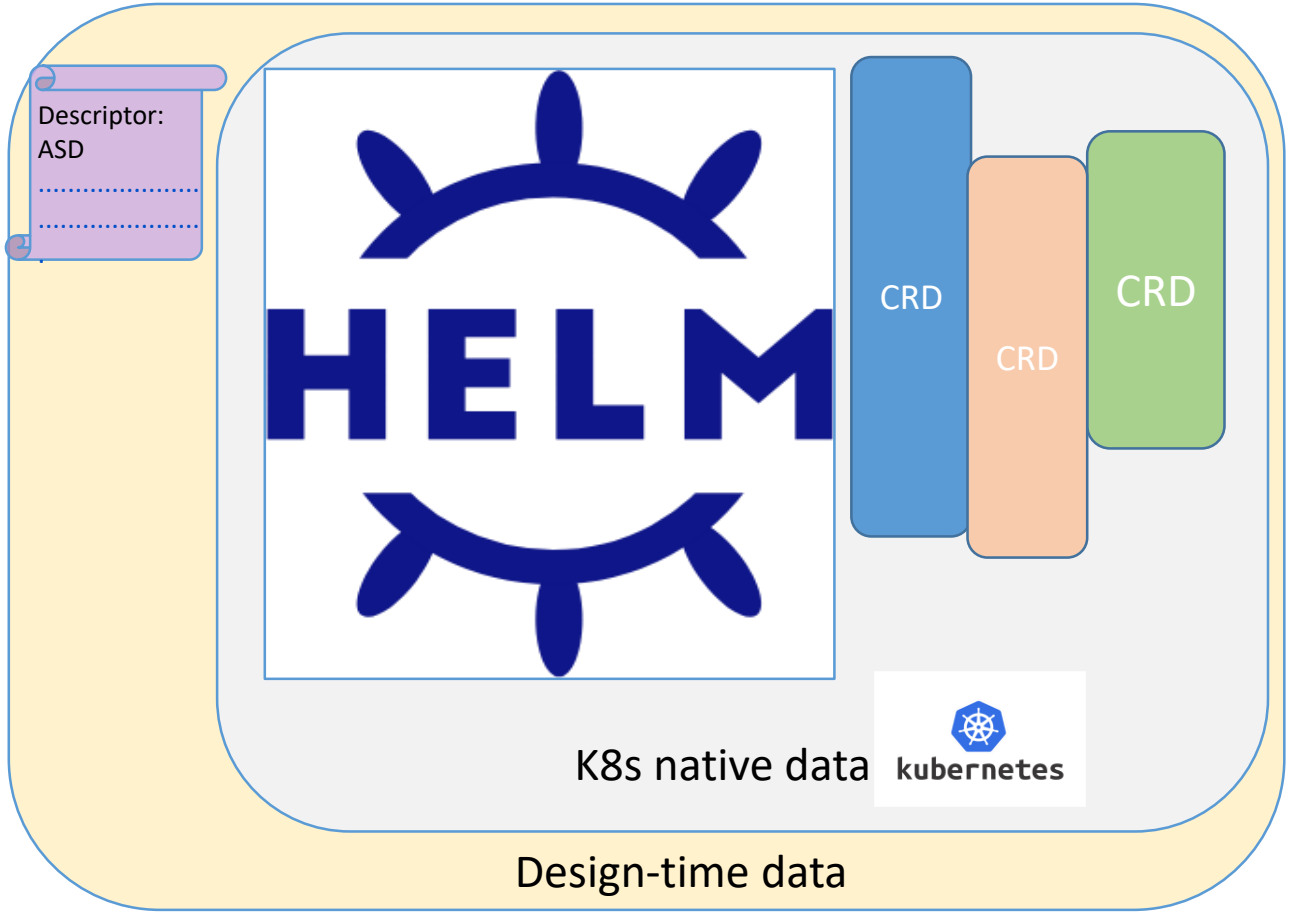
Difference in the modeling approaches (1)
ETSI NFV SOL001 vs ASD

ETSI NFV SOL001 VNFD*



*Requires ETSI NFV-MANO architecture (NFVO, VNFM)

ASD, an alternative**



**ETSI NFV-MANO architecture not needed

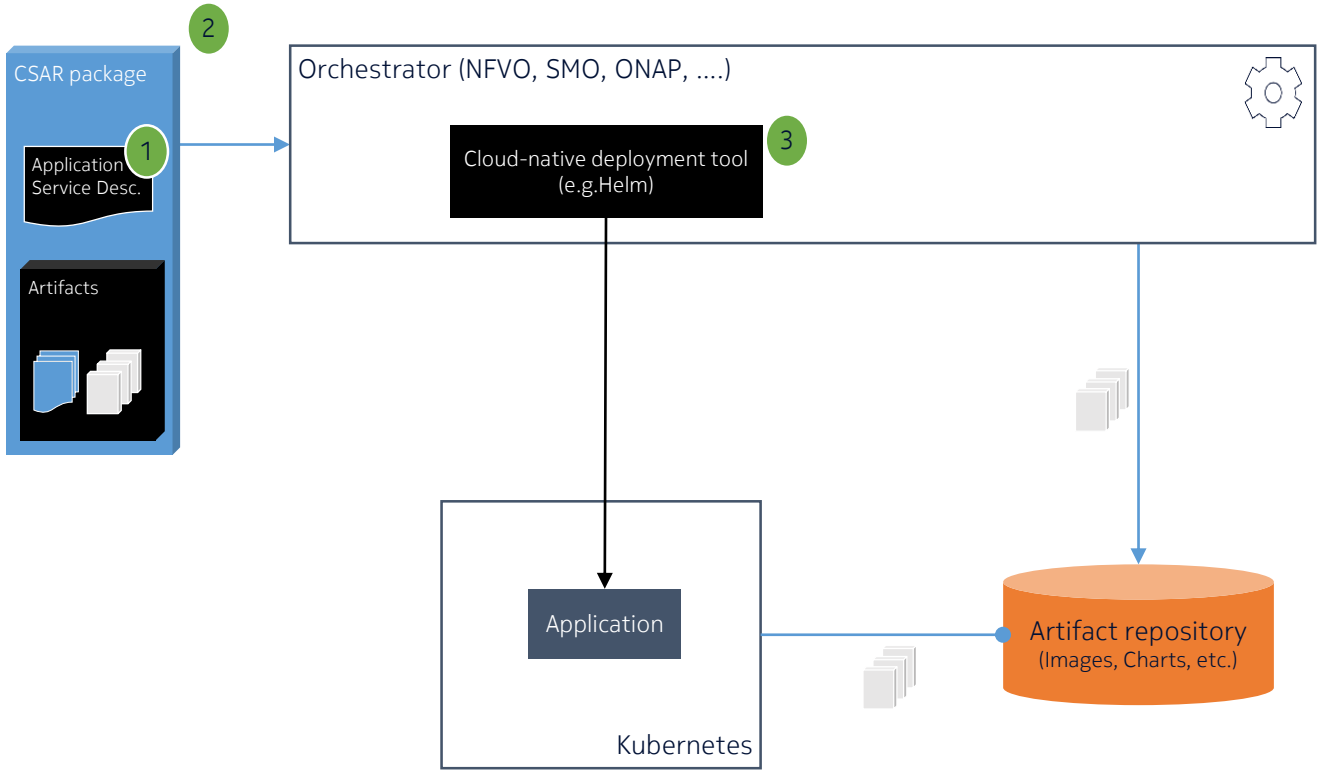
Part 2: Overview and status

ASD & packaging summary

- ASD is a common, simplified deployment descriptor for containerized cloud native deployments aiming to:
  - Quickly leverage enhancements in Kubernetes while minimizing development and integration efforts
  - Avoid duplication of attributes/properties included in Helm Charts
  - Descriptor format is not requiring a particular deployment tool (e.g., Helm)
  - Support CNF Direct Path project
  - Leverage established packaging standards (e.g., SOL004 with ASD as a top-level deployment artifact)
- ONAP: ASD requirement has been presented and has been approved by ONAP Requirement sub-committee
- ONAP: ASD information model and packaging format have been presented on several occasions and have been endorsed by CNF Task Force
  - ASD information model: https://wiki.onap.org/display/DW/Application+Service+Descriptor+%28ASD%29+onboarding+IM,
  - ASD packaging format: https://wiki.onap.org/display/DW/Application+Service+Descriptor+%28ASD%29+Onboarding+Packaging+Format
  - Detail proposal with example: https://wiki.onap.org/display/DW/Application+Service+Descriptor+%28ASD%29+and+packaging+Proposals+for+CNF
  - Jakarta release - functional requirements
  - REQ ticket in jira: https://jira.onap.org/browse/REQ-993
- Status:
  - Jakarta release: ASD information model specification, ASD Orchestration PoC

Part 3: ASD and Package:

ASD Model and Packaging Proposal



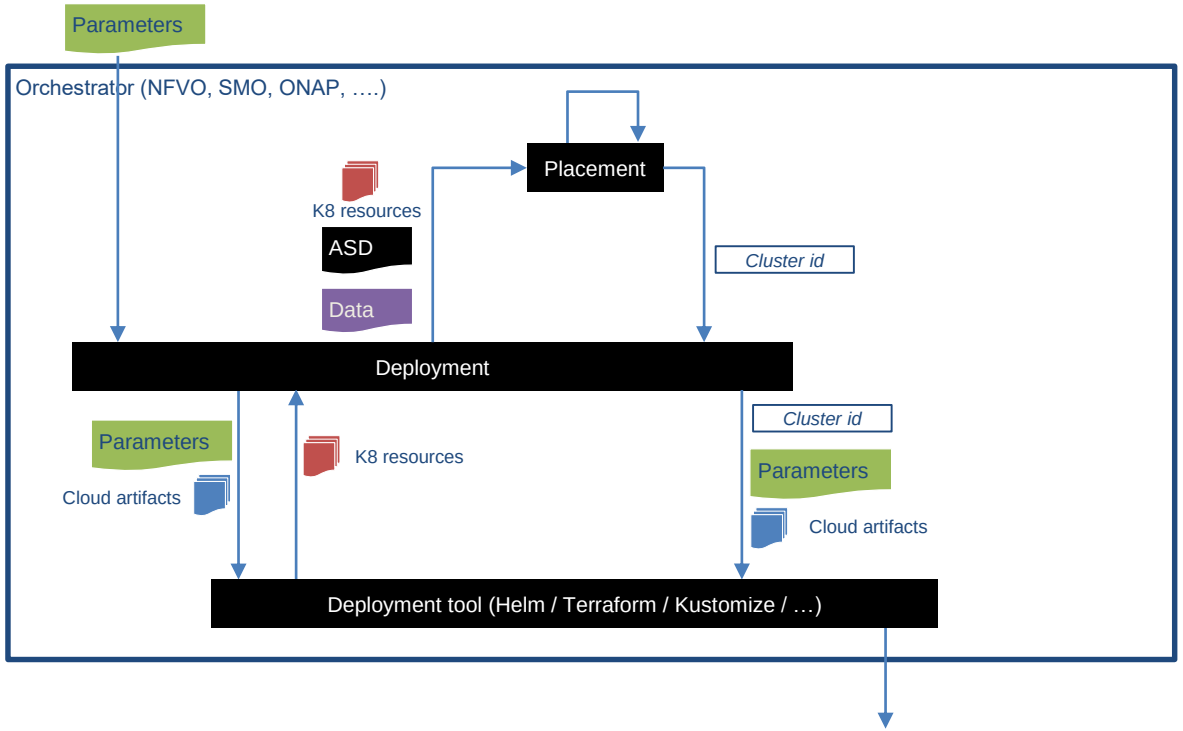
ASD Model proposal :

1. ASD is a common, simplified deployment descriptor for CNFs, xApps and rApps aiming to:
   - Quickly leverage enhancements in Kubernetes while minimizing development and integration efforts
   - **Avoid duplication of attributes/properties included in Helm Charts**
   - **Avoid vendor needs to maintain VNFD and data in native templates in synch; avoid error prone approach**
   - **Build on cloud native tooling**
   - Descriptor format is not requiring a particular cloud-native deployment tool (e.g. Helm)
   - Leverage established packaging standards (e.g., SOL004 with ASD as a deployment template)
2. Use CSAR packaging for bundling metadata, ASD and cloud-native artifacts in a single package. Describe the application using a lightweight, Application Service Descriptor.
3. As an example, Helm v3 is embedded as the initial cloud-native deployment tool in the orchestrator.

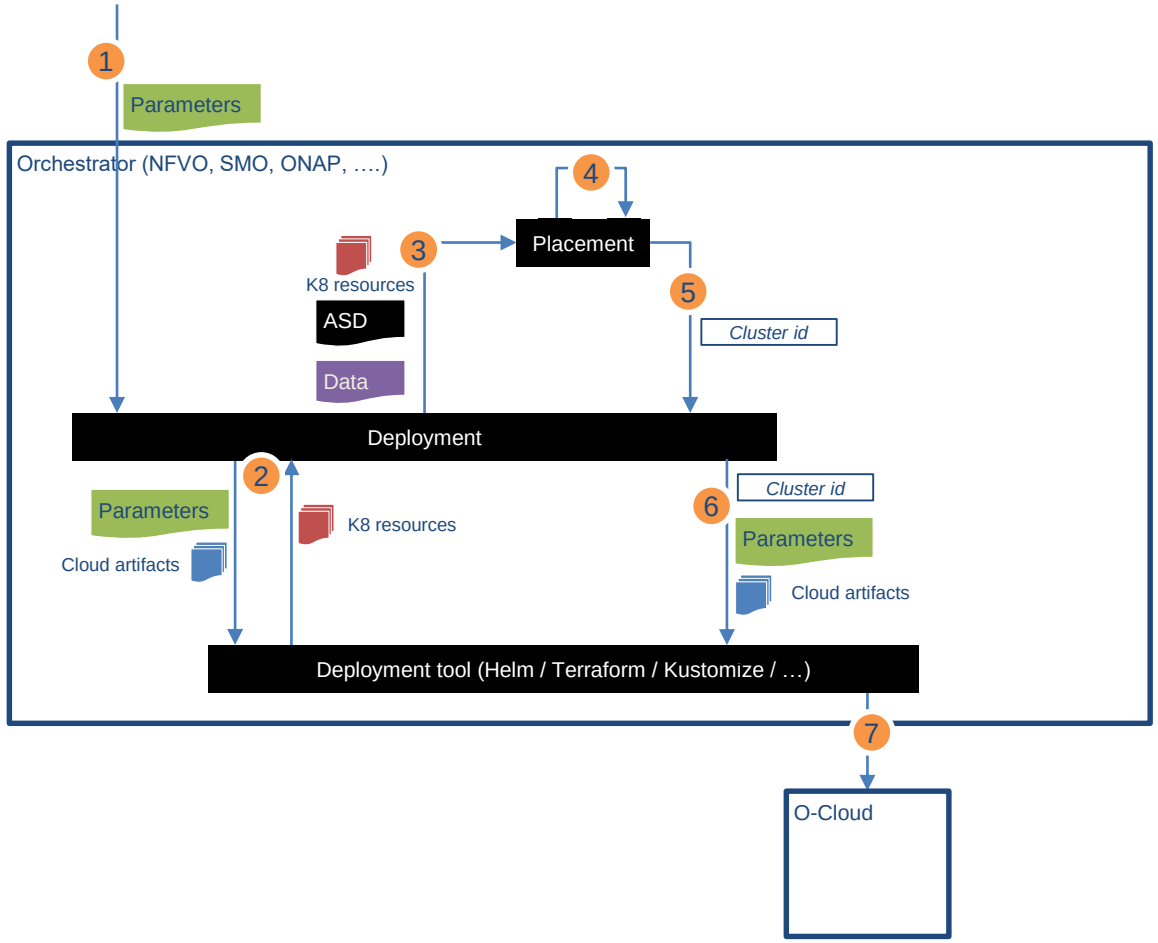
Example: Orchestration Capabilities

Non-Normative example of how SMO capabilities might be structured

- Functional block description:
  - Deployment: this function is in charge of coordinating the deployment of an application through the DMS interface. It receives deployment requests, interprets deployment package descriptors, delegates placement and interacts with cloud deployment tools.
  - Deployment tool: one or more commonly used cloud packaging / deployment tools, like Helm, Terraform, Kustomize, Helmfile…
  - Placement: this function is responsible for selecting an appropriate cluster in the available O-Cloud resource pools. The placement criteria is up to each implementor but has to respect the restrictions indicated in the deployment descriptors.



Example flow of application deployment



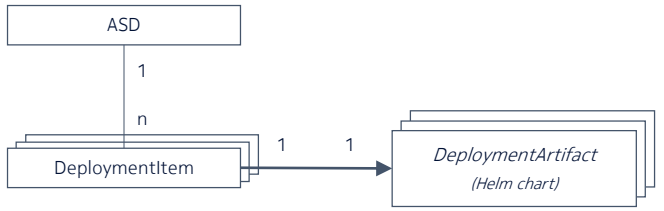
1. A deployment order is received, along with the required lifecycleParameters values
2. The cloud-native deployment tool is invoked with the received parameters to transform the cloud artifacts into K8S resource descriptions.
3. The K8S resource descriptions, ASD and any other relevant data is sent 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
ASD, top level

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

ASD: Deployment Item

As an alternative to “parent” or “umbrella” charts, we propose a means to structure and sequence multiple Helm charts that is easily expressed inside the ASD, using the DeploymentItem construct

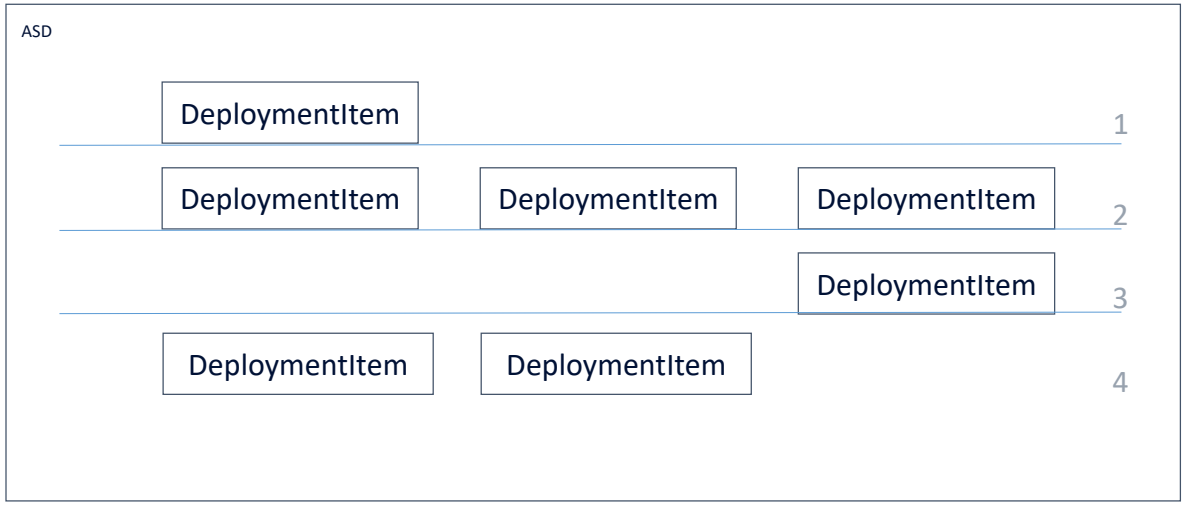


Attribute	Qualifier	Cardinality	Content	Description
deploymentItemId	M	1	Identifier	The identifier of this deployment item
artifactType	M	1	String	Specify artifact type. e.g. Helm chart, helmfile, CRD etc.
artifactId	M	1	String	Reference to a DeploymentArtifact. It can refer to URI or file path.
deploymentOrder	M	0..1	Integer	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.
lifecycleParameters	M	0..N	String	List of parameters that can be overridden at deployment time (e.g. values for values.yaml in the chart this item references)

ASD: deploymentOrder

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: asdExtCpd

Attribute	Qualifier	Cardinality	Content	Description
Id	M	1	String	The identifier of this extCpdData
description	M	1	String	Describes the service exposed.
virtualLinkRequirement	M	1..N	String	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.
networkInterfaceRealizationRequirements	M	0..1	datatype.networkInterfaceRealizationRequirements	Details container implementation specific requirements on the NetworkAttachmentDefinition ~~to~~ . See Note 2 & 3.
inputParamMappings	M	0..1	datatype.extCpd.ParamMappings	Information on what parameters that are required to be provided to the deployment tools for the asdExtCpd instance.
resourceMapping	M	0..1	String	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.

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 ExtCpd may refer to same additional network interface requirements.

ASD: asdExtCpd, networkInterfaceRealizationRequirements IE

Attribute	Qualifier	Cardinality	Content	Description
trunkMode	M	0..1	”false” | ”true”	If not present or set to”false”, means that this interface shall connect to single network. If set to ”true” then the network interface shall be a trunk interface (connects to multiple VLANS).
ipam	M	0..1	"infraProvided", "orchestrated", "userManaged"	The default value ("infraProvided") means that the CNI specifies how IPAM is done and assigns the IP address to the pod interface.
nicOptions	M	0..N	"examples": ["i710", "mlx-cx5v"]	nics a direct user space driver the application is verified to work with. Allowed values from ETSI registry.
interfaceType	M	0..1	"kernel.netdev", "direct.userdriver", "direct.kerneldriver", "direct.bond", "userspace	This attribute is applicable for passthrough and memif interfaces. Value default value is ”kernel.netdev”.
interfaceOptions	M	0..N	"virtio", "memif"	Alternative vNIC configurations the network interface is verified to work with.
interfaceRedundancy	M	0..1	"infraProvided", "activePassiveBond", "activeActiveBond", "activePassiveL3", "activeActiveL3", ”bondCni”, "Left", "Right"	”infraProvided” means that the application sees one vNIC but that the infrastruture provides redundant access to the network via both switch planes. ”Left” 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. "activeActiveBond": The bonded left/right links must be part of a multi-chassis LAG in active-active mode "activePassiveBond": Interfaces bonded in active-passive mode in the application with move of bond MAC address. No specific requirements on DC fabric. "activePassiveL3": Move of application IP address "activeActiveL3": Anycast/ECMP ”bondCni” ; the mated pair network interfaces are used via a 3rd bond cni based network interface.

ASD: asdExtCpd, datatype.ExtCpd.ParamMappings 1/2

Attribute	Qualifier	Cardinality	Content	Description
loadbalancerIP	M	0..1	String	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. Note 2
externalIPs	M	0..N	String	When present, this attribute specifies the name of the deployment artifact input parameter through which the orchestrator can configure the extermalIPs 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 2
ipAddressParameter	M	0..1	String	When present, this attribute specifies the name of the deployment artifact input parameter through which the orchestrator can configure the IP address(es), ipv4 and/or IPv6, for this asdExtCpd. The param name and provided IP address value will be passed to the deployment tool when deploying the DeploymentArtifacts. Note 1
nadNames	M	0..N	String	These attributes specifies, for an ExtCpd respesenting a secondary network interface, the name(s) of the deployment artifact input parameter(s) through which the orchestrator can provide the names 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 1,2,3
nadNamespace	M	0..1	String	Specifies, for an ExtCpd respesenting a secondary network interface, the name of the deployment artifact input parameter through which the orchestrator can provide the namespace where the NADs are located. Attribute may be omitted if the namespace is same as the application namespace. Note 2

ASD: asdExtCpd, datatype.ExtCpd.ParamMappings 2/2

Note 1: When the ExtCpd 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>.]$^{0..N}$<paramname>". Whether the optional parts of the format are present depends on how the parameter is declared in the helm chart. An example is:
"chartName:subChart1.subChart2.subChart3.Parent1.Parent2.Parent3.parameter".

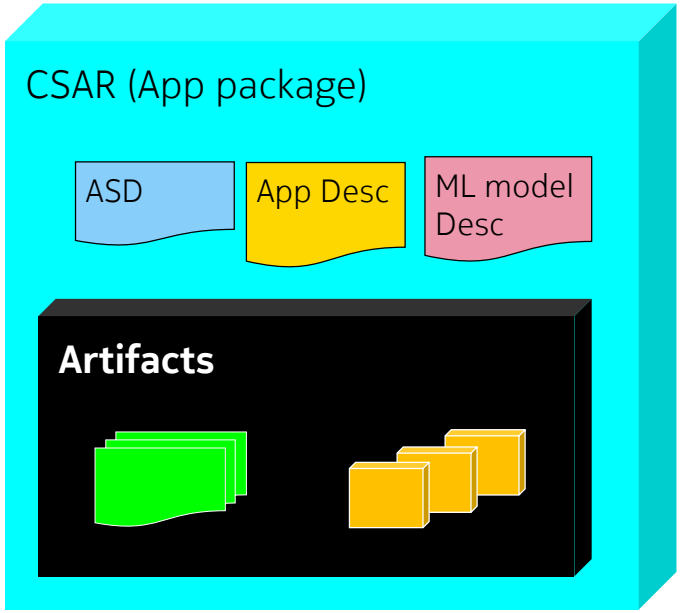
Note 3: A direct attached (passthrough) network interface, such as an sriov interface, attaches to a network via only one of the two switch planes in the infrastructure.
When using a direct attached network interface one therefore commonly in a pod uses a mated pair of sriov network attachments, where each interface attaches same network but via different switchplane.
The application uses the mated pair of network interfaces as a single logical “swith-path-redundant” network interface – and this is represented by a single ExtCpd.
Also there is a case where a third “bond” attachment interface is used in the pod, bonding the two direct interfaces so that the application do not need to handle the redundancy issues – application just uses the bond interface.
In this case all three attachments are together making up a logical “switch-path-redundant” network interface represented by a single ExtCpd. When three NADs are used in the ExtCpd the NAD implementing the bond attachment interface is provided through the parameter indicated in the third place in the nadNames attribute.

ASD: enhancedClusterCapabilities

Attribute	Qualifier	Cardinality	Content	Description
minKernelVersion	M	1	String	Describes the minimal required Kernel version, e.g. 4.15.0. Coded as displayed by linux command uname –r
requiredKernelModules	M	0..N	Strings	Required kernel modules are coded as listed by linux lsmod command, e.g. ip6_tables, cryptd, nf_nat etc.
conflictingKernelModules	M	0..N	String	Kernel modules, which must not be present in the target environment. The kernel modules are coded as listed by linux lsmod command, e.g. ip6_tables, cryptd, nf_nat etc. Example: Linux kernel SCTP module, which would conflict with use of proprietary user space SCTP stack provided by the application.
requiredCustomResources	M	0..N	Structure (inlined)	List the required custom resources types in the target environment, identifying each by the "kind" and "apiVersion" field in the K8S resource manifests and in the application. The list shall include those custom resource types which are not delivered with the application. Example: requiredCustomResources: -{kind: "Redis", apiVersion: "kubedb.com/v1alpha1"}
>kind	M	0..1	String	Kind of the custom resource
>apiVersion	M	0..1	String	apiVersion of the custom resource
clusterLabels	M	0..N	String	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 (https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/). It is recommended that labels follow a standardised meaning e.g. for node features (https://kubernetes-sigs.github.io/node-feature-discovery/v0.9/get-started/features.html#table-of-contents). Example: ClusterLabels - feature.node.kubernetes.io/cpu-cpuid.AESNI: true
requiredPlugin	M	0..N	Structure (inlined)	A list of the names and versions of the required K8s plugin (e.g. multus v3.8)
>requiredPluginName	M	0..1	String	The names of the required K8s plugin (e.g. multus)
>requiredPluginVersion	M	0..1	String	The version of the required plugin (e.g. 3.8)

Part 4: ONAP ASD Package

Leveraging ETSI-compliant packaging



The proposal for packaging is to continue to rely on CSAR.

Recommend to standardize:

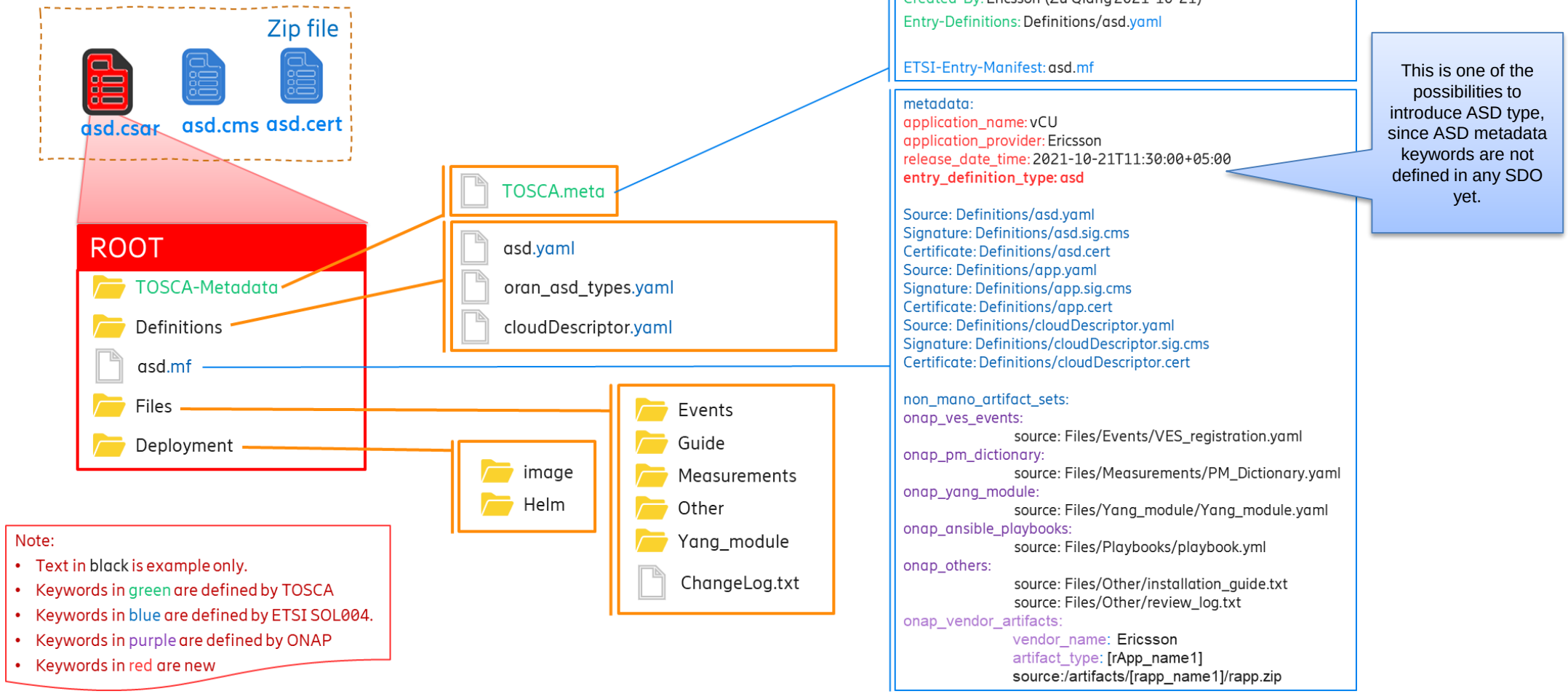
a) The Packaging of containerized applications, the ASD itself, and the usage of ASD as a deployment artifact in a CSAR.
b) The use of Helm as the cloud-native artifact for describing (pieces of) the application
c) Adopt SOL004, adding the possibility to include an ASD instead of a VNFD as a top-level artifact

ASD Onboarding Packaging Format

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.

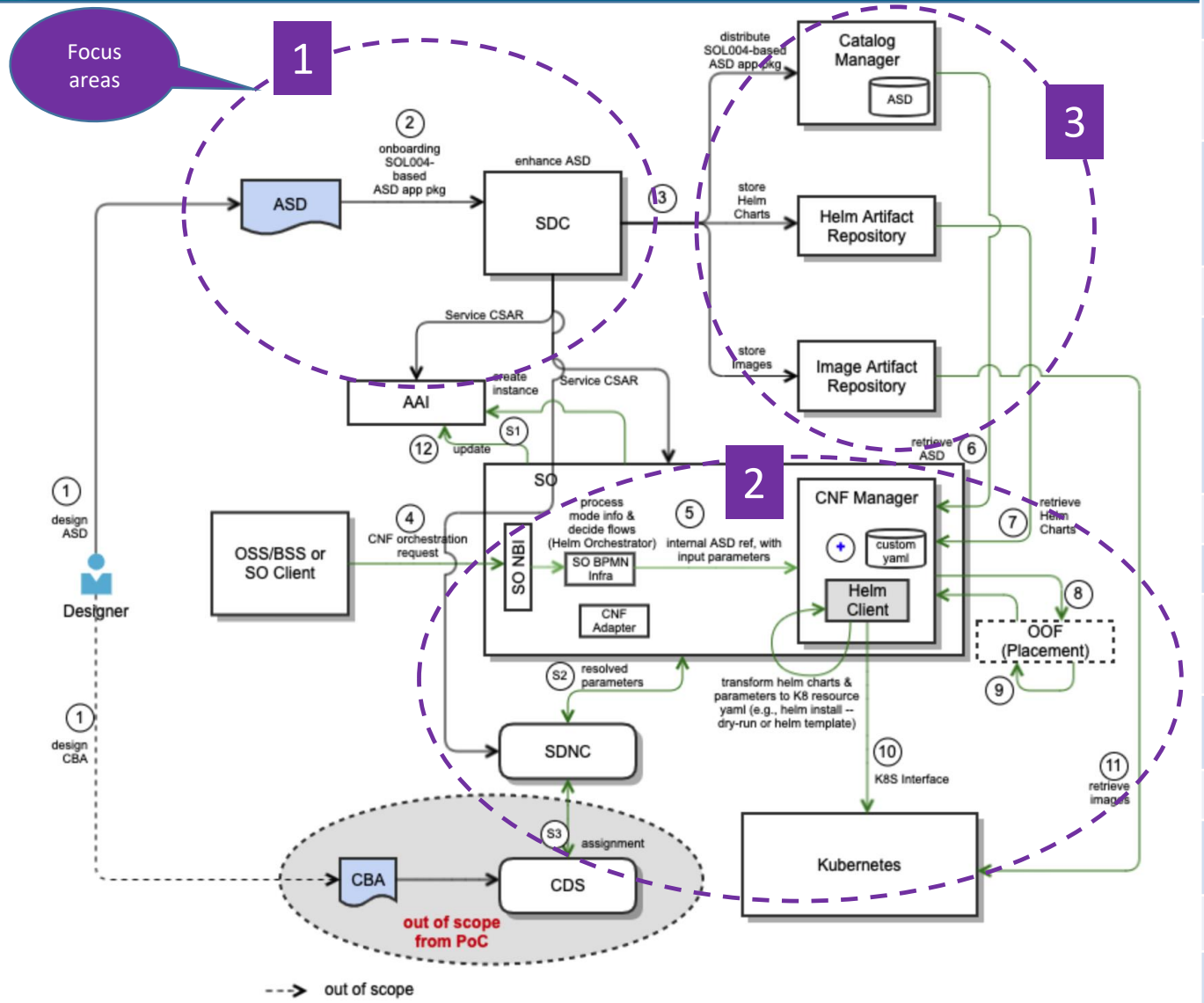
An example of an ASD onboarding package



Part 5: ONAP ASD PoC

ASD Onboarding, Distribution and Orchestration PoC

- Target: Jakarta Release



#	Actor	Action
1	Designer	•Design ASD and/or CBA
2	Designer / SDC	•Onboard SOL004-based ASD package (could include CBA) •Design Resource & Compose Resources into Service •Enhance ASD (ASD')
3	SDC	•Distribute Service CSAR to ONAP runtime components (SO, AAI, SDNC, etc.) •Distribute ASD (ASD') to Catalog Manager •Distribute Helm Charts to Helm Repository (centralized) •Distribute Images to Image Repository (centralized)
4	SO Client	•Start CNF service creation
5	SO	•S1: create instance to AAI •S2: resolve parameters •S3: CDS returns assignment •Process model & decide flows (in this case, Helm Orchestrator) •Delegate Resource-level orchestration to Helm Orchestrator •Pass ASD' or its reference with input (resolved) parameters
6	CNF Manager	•Retrieve ASD (ASD') from Catalog Manager
7	CNF Manager	•Retrieve Helm Charts from Helm Repository
	CNF Manager - internal	•Transform ASD cloud artifacts with parameters to K8S resource description (e.g., helm template or helm install --dry-run) •Get additional data as needed
8	CNF Manager	•Get placement information by passing K8 resources + ASD' + additional data to the Placement
9	Placement	•make a placement decision based on input data & return it
10	CNF Manager	• Set the target K8S cluster and Send CNF creation request (with cluster id, parameter, cloud artifacts) to K8S (e.g., helm install ...)
11	k8S	•Retrieve images from Image Repository and process for CNF
12	SO	•Update instance to AAI

Summary

Application Service Descriptor - Summary

- Application Service Descriptor (ASD) provides simplified way of modelling and packaging of NFs
  - It's an alternative to 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.
- PoC ongoing to proof the concept in ONAP environment *LINK*

Back-up

Difference in the modeling approaches (2)
ETSI NFV SOL001 vs ASD

Modeling approach	A) Data Overlap with cloud native templates	B) Data conflicts risks => out-of-synch templates	C) Building on Cloud native tools	Findings
ETSI NFV SOL001 VNFD	YES	YES	NO	A) Vendor needs to maintain the VNFD and data in native templates in Synch; an error prone approach B) Conceptual and functional differences between VM-based and container-based native technologies C) Incompatible with K8s extension mechanisms (e.g., CRDs not supported) D) VNFD needs to play catch-up with ever-growing data from CNCF ecosystem, otherwise it limits orchestration to its data subset, stripping the benefits of CNCF ecosystem's open extensibility
ASD	NO (Complements)	NO	YES	
Others (CRD based)	NO (Complements)	NO	YES	

NOKIA