LF NETWORKING

LFN Developer & Testing Forum

LF NETWORKING

LFN Developer & Testing Forum

Cloud Native Orchestration Using ASD

Innovative Cloud Native Modeling and Packaging

November 18, 2022

Presented by: Byung-Woo Jun, byung-woo.jun@est.tech

PoC Team:

Michael Morris, Krupa Nagabhushan, Jeff van Dam, Vasyl Razinkov,

Waqas Ikram, Raviteja Karumuri, Gerard Nugent, Nadeeshani Jayathilake

Anti-Trust Policy Notice

- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrustpolicy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

Agenda

- Application Service Descriptor (ASD)
- ASD Design & Orchestration Overview
- ASD Onboarding to ONAP SDC
- ASD Orchestration By ONAP SO
- Demo - Kubernetes Environment & Components
- Next
- Q&A

Application Service Descriptor (ASD)

- **What?**
  - Deployment descriptor for cloud native applications/ functions
  - Minimum information needed for the Orchestrator
  - References to the cloud native artifacts
  - CSAR package adhering to ETSI SOL004
- **Why?**
  - Minimise duplication
  - Use Kubernetes management capabilities
  - Clean separation between high-level orchestration and cloud native application deployment
- **Business Driver?**
  - Ericsson Cloud RAN
  - ORAN
  - Reference implementation in ONAP

ASD Proof of Concept (PoC) in ONAP*

- The PoC was executed in ONAP framework to demonstrate viability of the ASD concept for NF modelling and packaging.
- Sample applications like {MariaDB, nginx}, {Free5GC and UERANSIM} have been modelled and packaged following the ASD concept.
- The packages have been onboarded in the ONAP management system.
- Using ONAP components, the sample applications modelled and packaged using the ASD concept have been successfully instantiated in the underlaying Kubernetes cluster.

* ONAP – Open Network Automation Platform, https://www.onap.org/

ASD PoC Phase 1 Goals

ASD Package Onboarding & Design to SDC

- Onboarding & validating vendor ASD App packages
- Transforming onboarding ASD models and packages into ONAP internal models and packages
- Adding additional artifacts, such as vendor licensing models
- Adding custom properties as needed

Service CSAR with ASD resources distribution to ONAP Runtime thru DMaaP

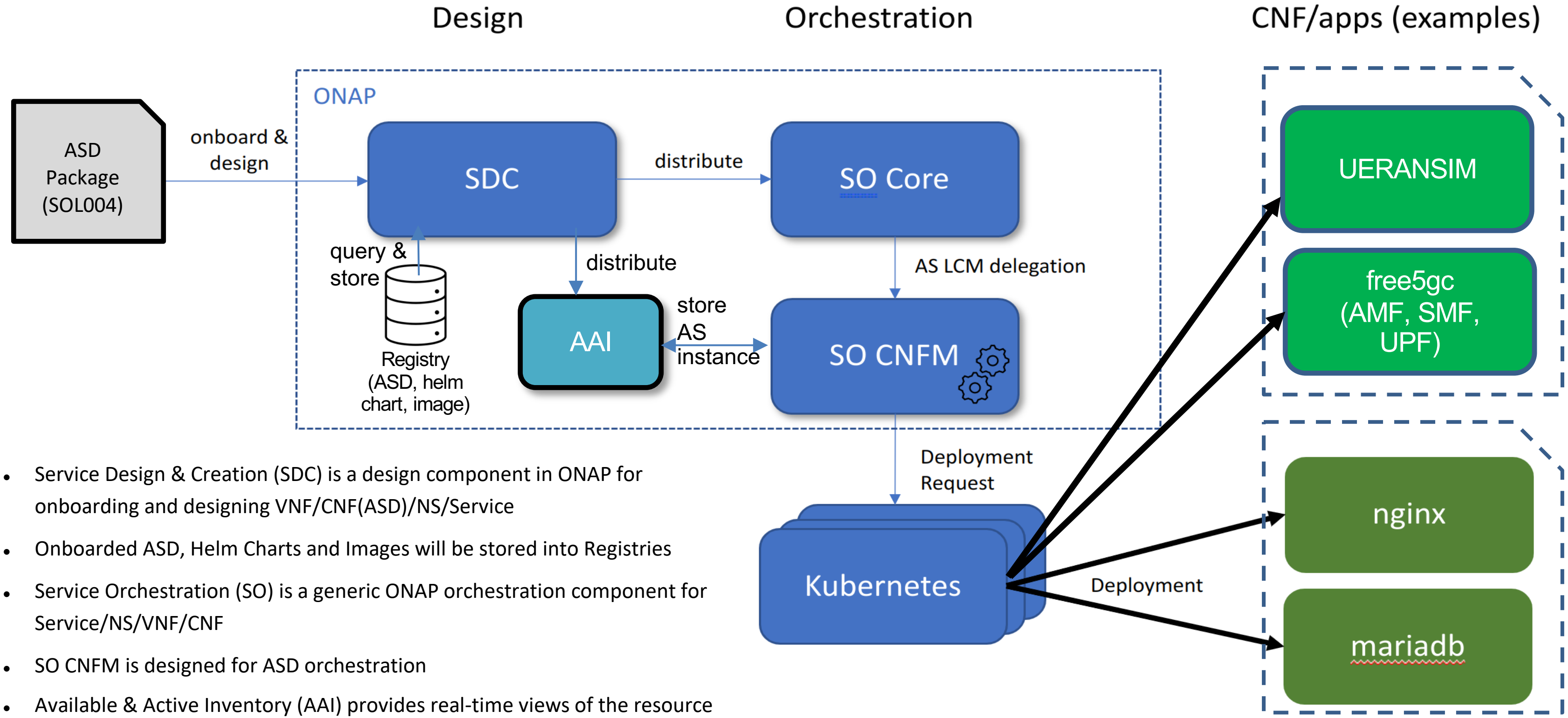
- Preserving the vendor ASD CSAR inside Service CSAR
- Providing Query APIs for ASD, Helm Charts and Image artifacts

SO Orchestration for ASD-Aligned CNFs/Apps

- Create / Instantiate / Terminate / Delete / Query AS operations
- External K8s Clusters registration
- AS topology inventories (Generic VNF / VF-Modules / K8s Resources)
- Deploy associated Helm Charts to a selected K8s cluster, based on ASD definitions and User input

Learning & Feedback

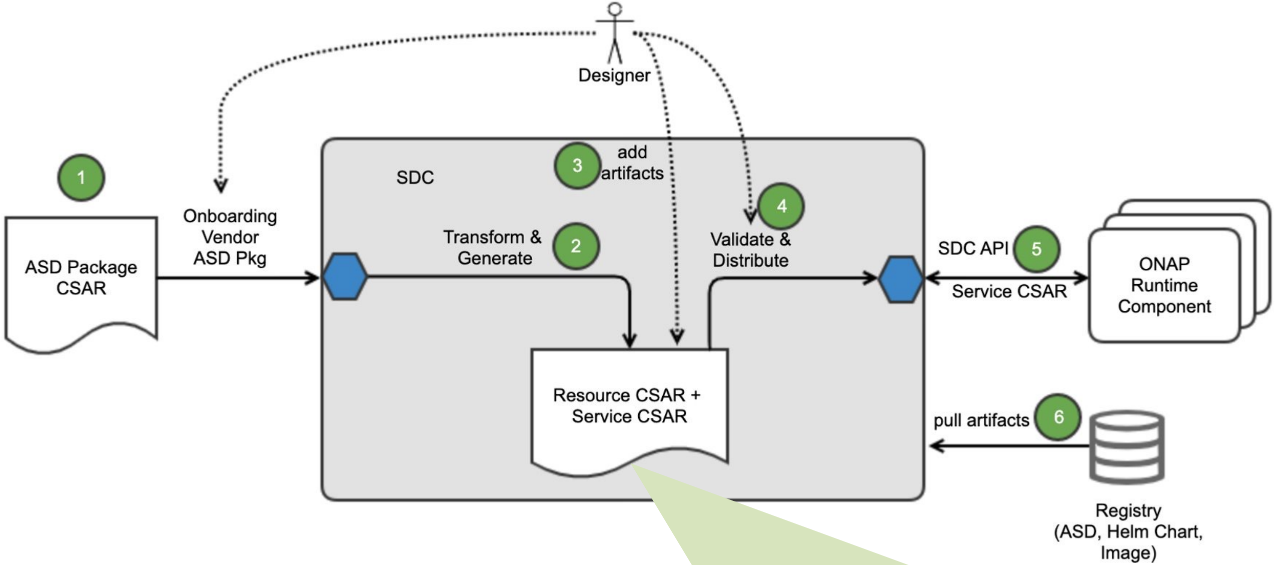
ASD Design & Orchestration Overview



- Service Design & Creation (SDC) is a design component in ONAP for onboarding and designing VNF/CNF(ASD)/NS/Service
- Onboarded ASD, Helm Charts and Images will be stored into Registries
- Service Orchestration (SO) is a generic ONAP orchestration component for Service/NS/VNF/CNF
- SO CNFM is designed for ASD orchestration
- Available & Active Inventory (AAI) provides real-time views of the resource and service instances and relationships

ASD Onboarding to ONAP SDC

1. Onboard vendor ASD packages and package validation
2. Transform onboarding ASD models into ONAP models and generate Resource CSAR and Service CSAR for distribution
3. Add additional artifacts such as vendor licensing models
4. Validate and decide for distributing the Service CSAR to ONAP runtime
5. Distribute the Service CSAR to ONAP runtime
6. Pull & store ASD, Helm Charts and Images from SDC to Registries (future implementation)



SDC generates Resource (VF) CSAR, based on ASD.

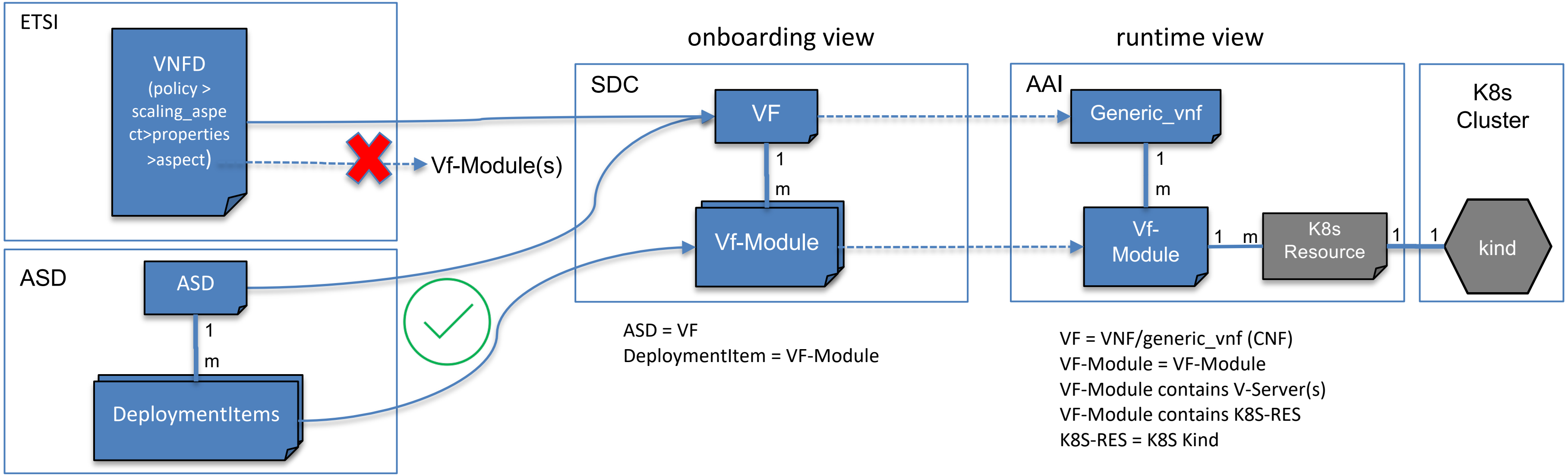
A. ASD-specific metadata is copied into the specific files, which will be used by Runtime components. (entity_definition_type = ‘asd’)

B. Onboarded ASD models are mapped into the ONAP internal models; ASD <-> ONAP VF, DeploymentItems <-> VF-Modules).

C. Add the AS_PACKAGE directory to store the original vendor ASD CSAR

ASD Model Transitions in ONAP

- ETSI VNFD does not have its sub-structure that corresponds to ONAP Vf-module
    - Current ONAP ETSI-Alignment supports VNF and above, no mapping to ONAP Vf-module
- ASD models are mapped into ONAP internal models
- ONAP VF and Vf-module will be stored into AAI as generic-vnf and vf-module
- K8s Cluster resource Kinds will be represented as K8s resources in AAI



ASD = VF
DeploymentItem = VF-Module

VF = VNF/generic_vnf (CNF)
VF-Module = VF-Module
VF-Module contains V-Server(s)
VF-Module contains K8S-RES
K8S-RES = K8S Kind

ASD Orchestration By ONAP SO

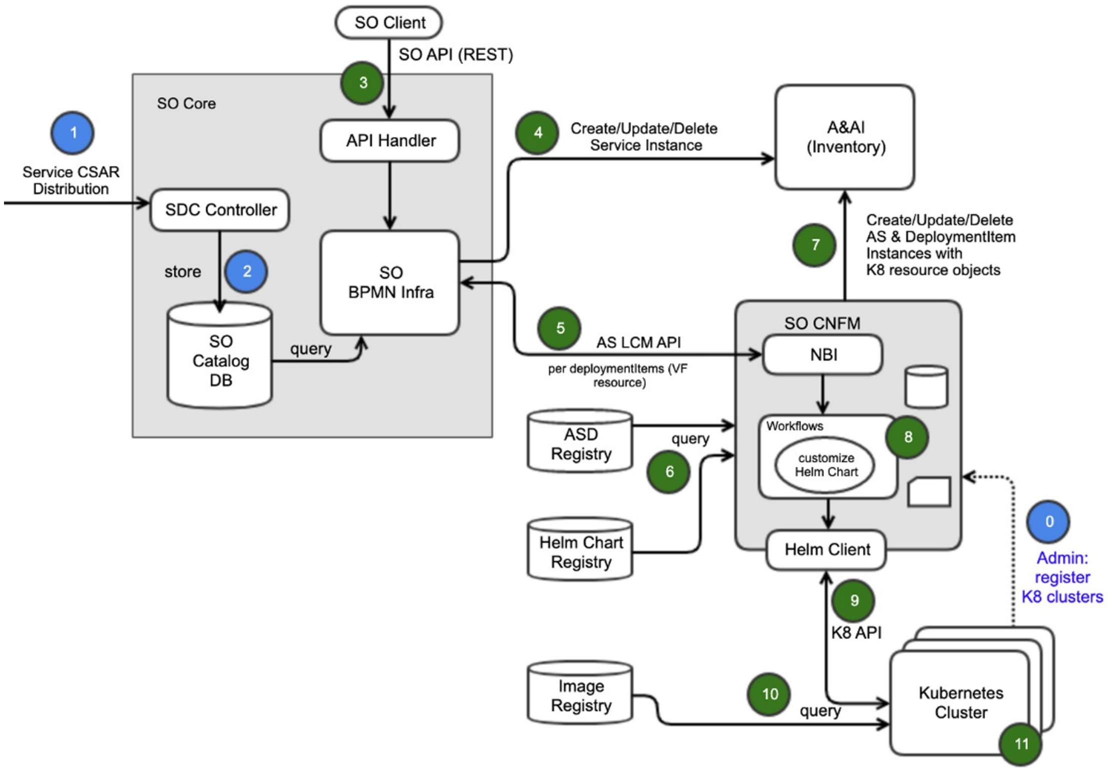
0. Register multiple Kubernetes Clusters

SO Core

1. Onboard ASD Service CSAR into SO runtime
2. Store ASD Metadata into SO Catalog DB
3. Support SO API Endpoint for ASD-CNF orchestration
4. Create/Update/Delete a Service Instance in AAI

SO CNFM

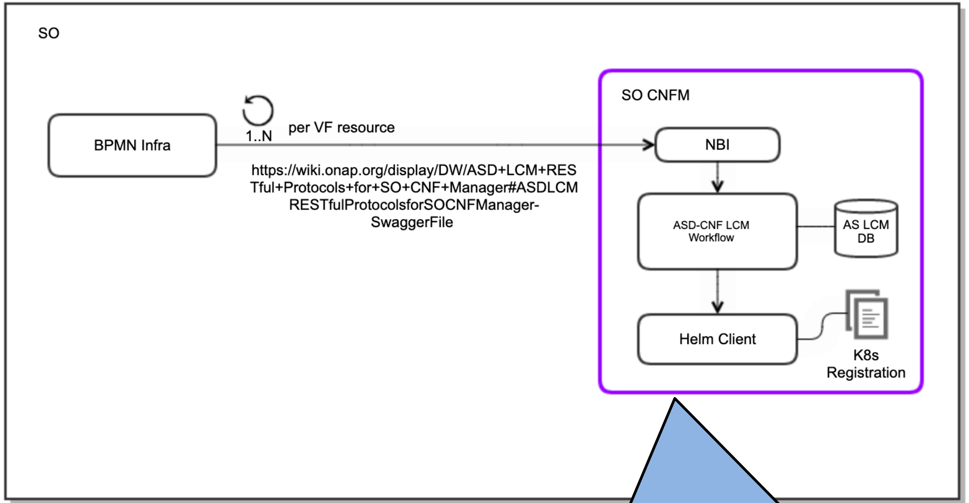
5. Use AS LCM APIs as NBI (AS LCM RESTful Protocols)
6. Query ASD and Helm Charts from the registries
7. Store ASD & DeploymentItems instances with K8 resources (kind list) in AAI
8. Customize instance-level Helm Chart values, based on User Input and ASD
9. Deploy ASD-based CNFs to Kubernetes clusters through Helm Commands and Kubernetes APIs
10. Query Images from the registry
11. Applications deployed in the K8s and operational



In PoC, images will be provided to K8s Clusters manually

SO CNFM Details

- Supports NBI which implements AS LCM RESTful protocols
  - Create/Instantiate/Terminate/Delete AS workflows in BPMN Infra invokes SO CNFM thru AS LCM RESTful protocols
  - AS LCM RESTFul Protocols
- Is a pluggable SO sub-component with NBI, Business Logic (AS-CNF LCM) and SB plugin (Helm client for now, CNF Adapter for EMCO is under consideration)
- Has corresponding workflows for Create/Instantiate/Terminate/Delete/Query AS operation
  - ASD LCM Operations in PoC
- Contains Camunda-based BPMN workflows and associated Java business logics for AS-CNF LCM.
- Handles external K8S cluster registration for creating available K8S cluster pools
- Manages ASD and associated Helm Charts Queries
  - For the initial PoC, SO CNFM retrieves ASD, Helm Charts from SDC, instead of artifact registries.
  - Images will be provided to K8S clusters manually for now
- Customizes instance-level Helm Charts values, based on ASD lifecycleparameters and user input
- Invokes Helm commands to deploy ASD-CNFs
- Invokes Kubernetes Java API to monitor K8s resources

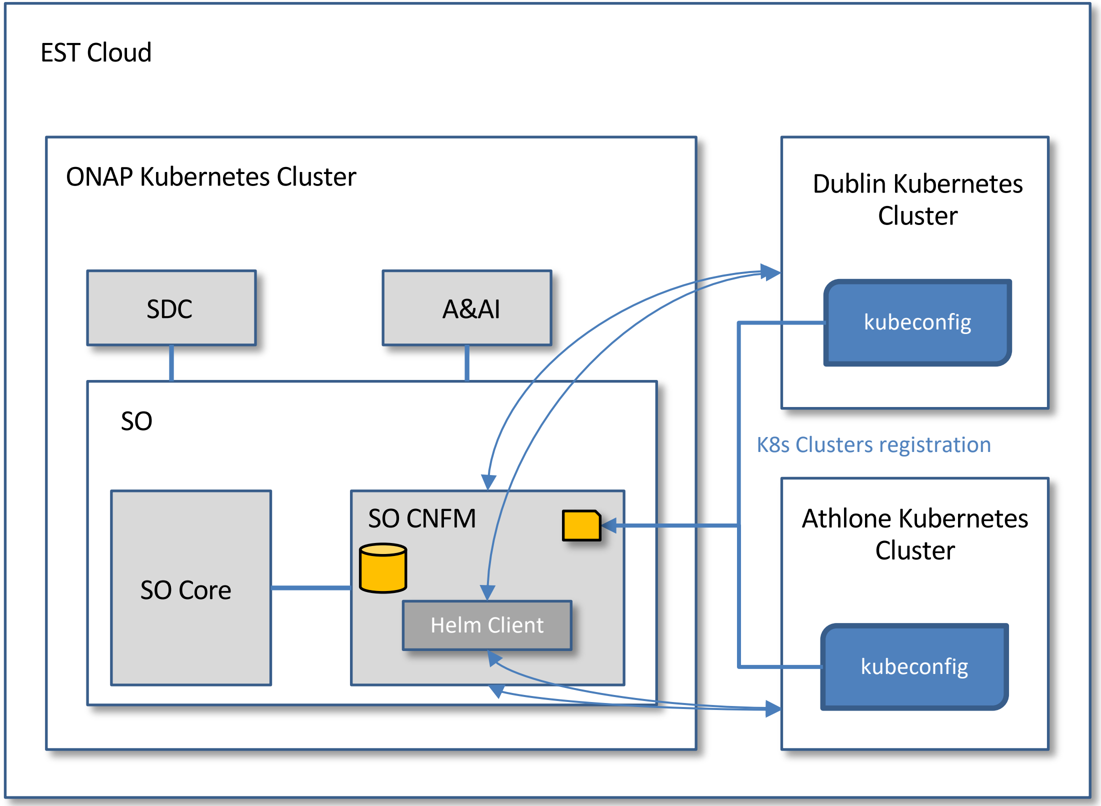

SO
BPMN Infra
1..N
per VF resource
https://wiki.onap.org/display/DW/ASD+LCM+RESTful+Protocols+for+SO+CNF+Manager#ASDLCMRESTfulProtocolsforSOCNFManager-SwaggerFile
SO CNFM
NBI
ASD-CNF LCM Workflow
AS LCM DB
Helm Client
K8s Registration

Note: Helm Client is used as the SO CNFM SB adapter. When EMCO is used, another adapter can be used (e.g., SO CNF Adapter)

Demo – Kubernetes Env & Components

LF NETWORKING
LFN Developer & Testing Forum

- All the ONAP components are deployed to "ONAP" K8s cluster
- To demonstrate multiple external K8s clusters handling, created "Dublin" and "Athlone" K8s clusters
- Multiple K8s Clusters will be registered into SO CNFM thru kubeconfig
  - Copy kubconfig info external K8s Clusters into SO CNFM
- Helm Client chooses one of the registered K8s clusters by using kubeconfig setting
- Helm Client uses Helm commands to interact with external K8s Clusters
- SO CNFM uses Kubernetes Java APIs for monitoring K8s resources

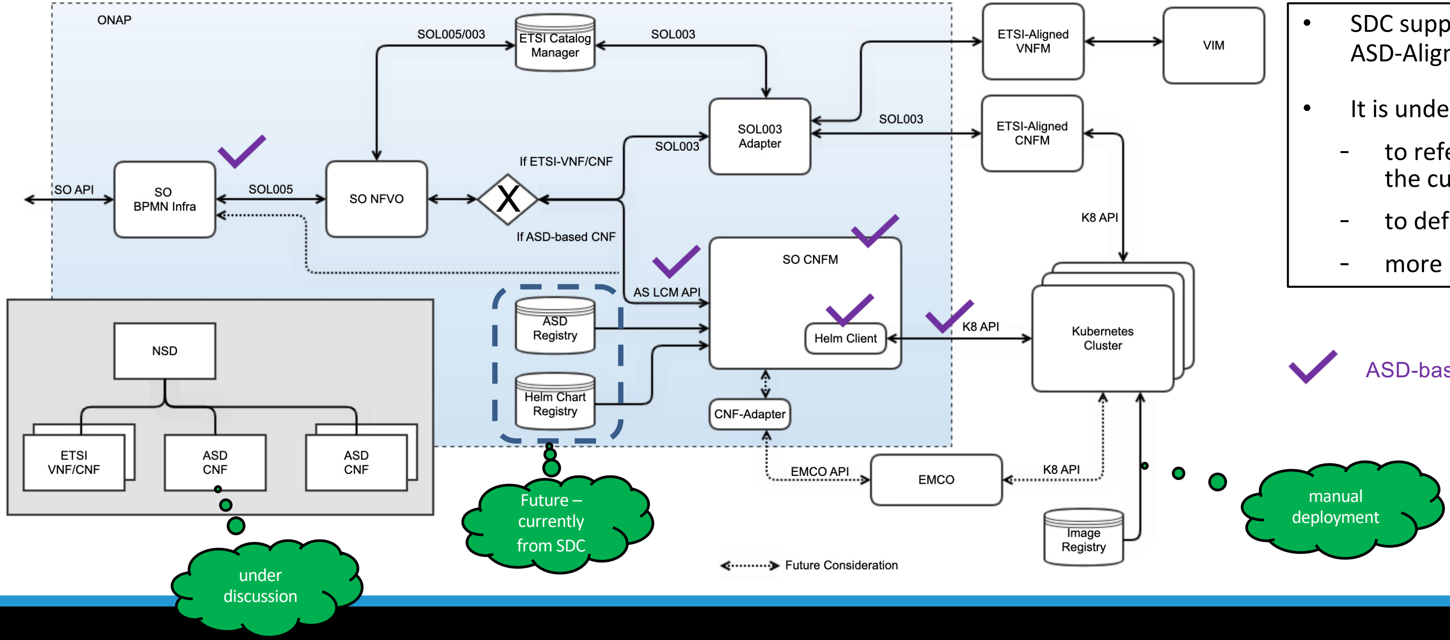


NSD & ASD Orchestration Plan

- NSD can contain both ETSI-Aligned VNF/CNFs and ASD-based CNFs, defining network
- NS LCM Orchestration uses SO NFVO thru SOL005
- SO NFVO decomposes embedded VNFs/CNFs
- If the model is ETSI-VNF/CNF, SO NFVO will invoke SOL003 Adapter thru SOL003
  - Depending on the vendor VNFM / CNFM capabilities, the orchestration path can reach Kubernetes clusters or VIM

- If the model is ASD-based CNF, SO NFVO will invoke SO CNFM thru AS LCM API
  - SO CNFM queries ASD and Helm Charts
  - SO CNFM generates instance-level deployment artifacts (e.g., helm charts), based on the user input parameter

- SDC supports both ETSI-Aligned VNF/NS and ASD-Aligned CNF
- It is under discussion how to enhance NSD:
  - to reference ASD-Aligned CNFs in addition to the current ETSI-Aligned VNFs/CNFs
  - to define Virtual Links for ASD-based CNFs
  - more



✓ ASD-based CNF orchestration PoC

Next & Q & A

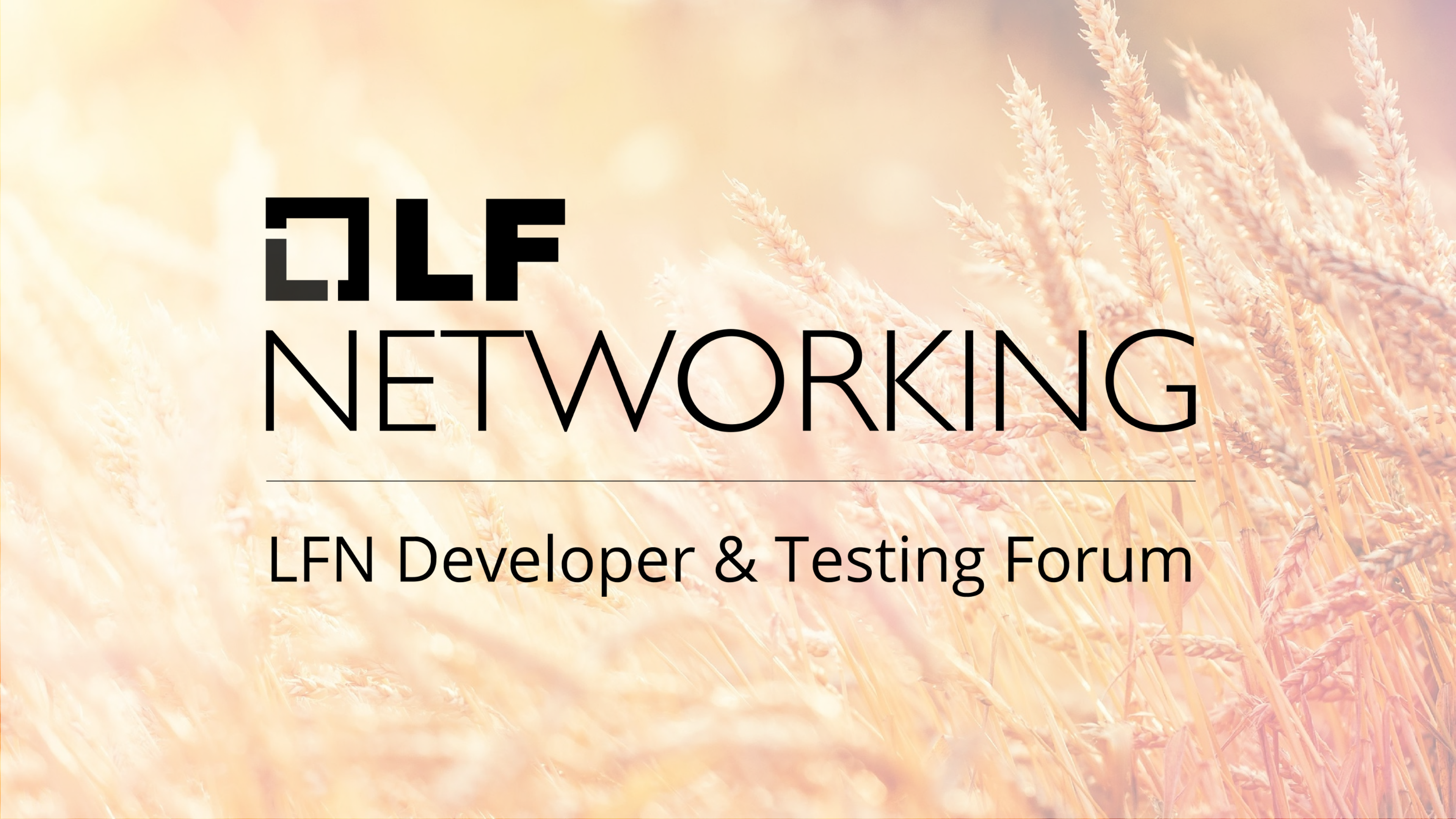
- Helm Charts values customization Enhancements
- Kubernetes Cluster registration Enhancements
- Helm Upgrade
- Use Case Support for the enhancedClusterCapbilities
- NS-ASD Onboarding and Orchestration
    - NSD and ASD support
    - NS with ASD Onboarding to SDC and SO NFVO Enhancements
        - DeploymentItems affinity
        - Multi-Cluster deployment

- Registries Management
    - ASD Registry
    - Helm Chart Registry
    - Image Registry

- EMCO + Adapter

Q & A

Thank you!



LF NETWORKING

LFN Developer & Testing Forum