REQ-458 – CNFO – Honolulu Extensions

Lukasz Rajewski (Orange)

Seshu Kumar M (Huawei)

Konrad Bańka (Samsung)

09.11.2020

CNFO - Summary for the requirement subcommittee

Executive Summary - Provide CNF orchestration support through integration of K8s adapter in ONAP SO

- Support for provisioning CNFs using an external K8s Manager
- Support the Helm based orchestration
- leverage the existing functionality of Multi cloud in SO
- Bring in the advantages of the K8s orchestrator
- Set stage for the Cloud Native scenarios

REQ-341
Guilin

REQ-458
Honolulu

Owners: Lukasz Rajewski (Orange), Seshu Kumar M (Huawei), Srini Addepalli (Intel)

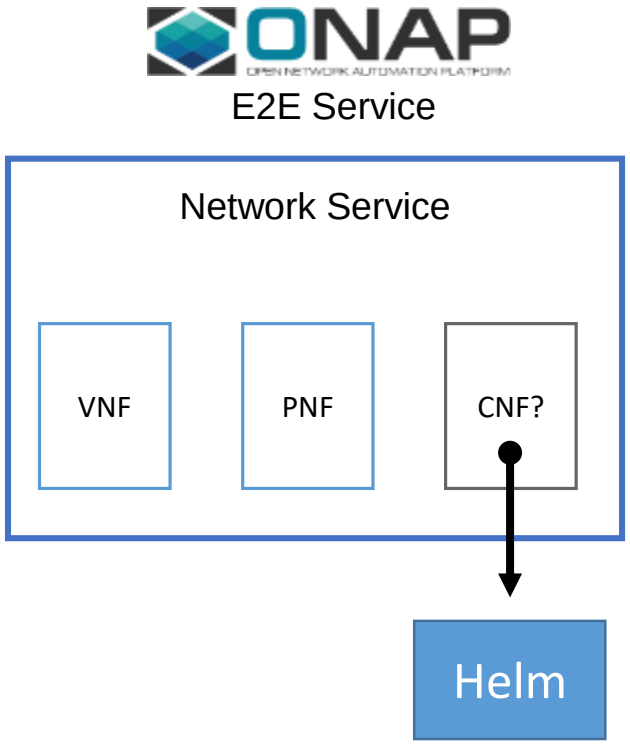
Business Impact - Enables operators and service providers to orchestrate CNFs based services along with the VNFs and PNFs

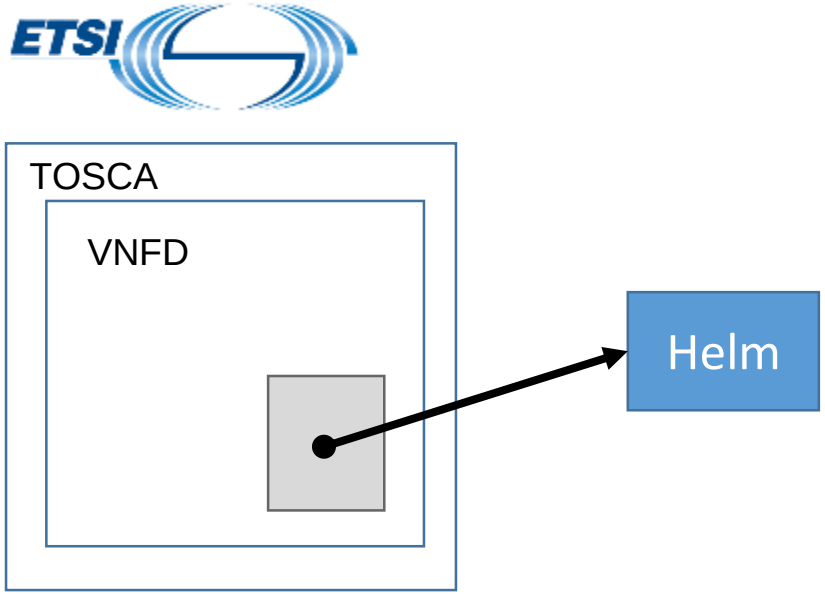
Business Markets - All operators and service providers that are intended to use the CNFs along with PNFs / VNFs

Funding/Financial Impacts - Reduction in the footprint of the ONAP for CNF support.

Organization Mgmt, Sales Strategies - *There is no additional organizational management or sales strategies for this requirement outside of a service providers "normal" ONAP deployment and its attendant organizational resources from a service provider.*

ONAP - ETSI CNF model Alignment



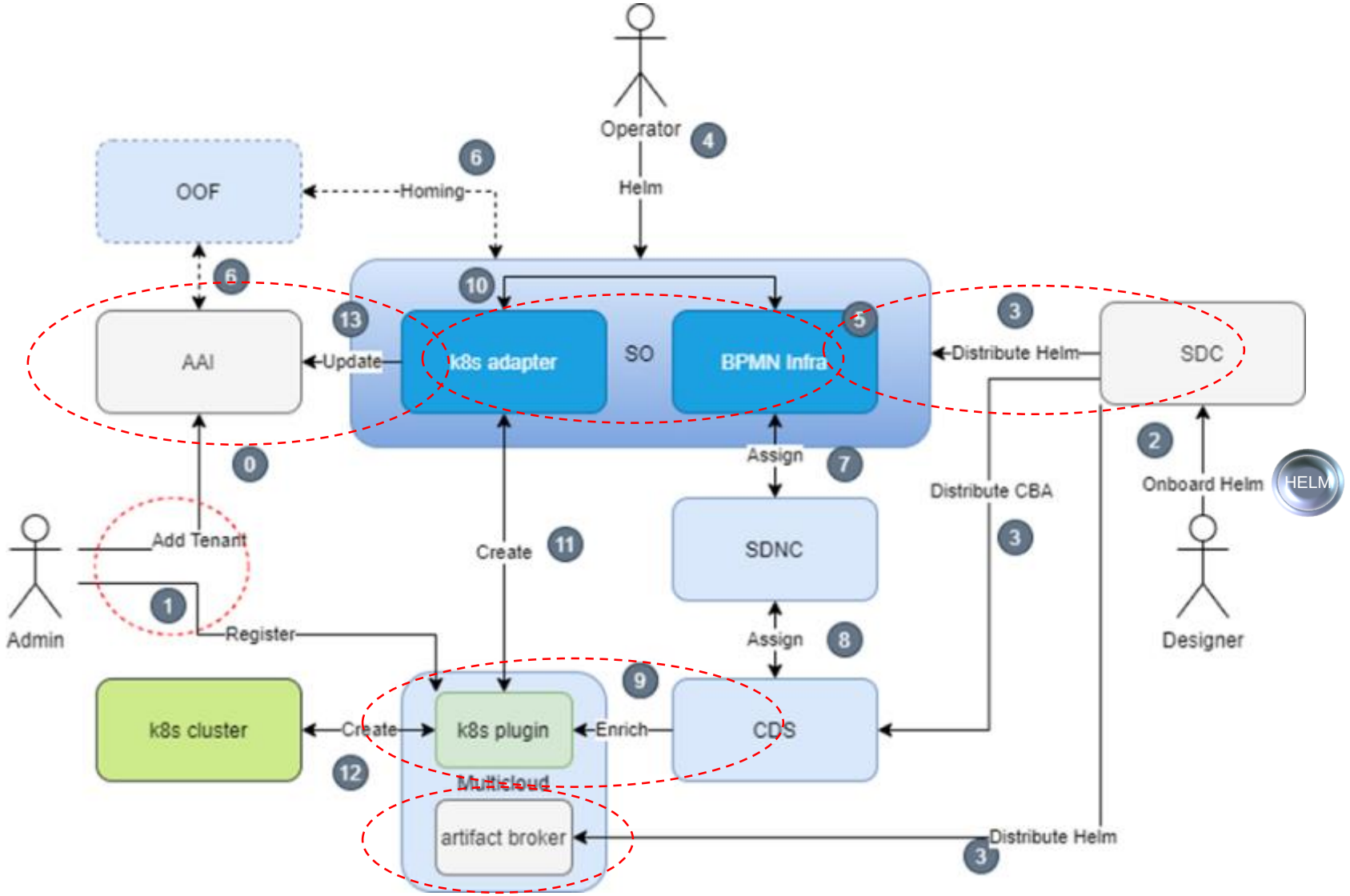


Integration of Native (CNF Adapter) with ETSi (SOL003 Adapter) paths in SO

Design/AAI CNF Model

How the ETSI CNF AAI model will look?

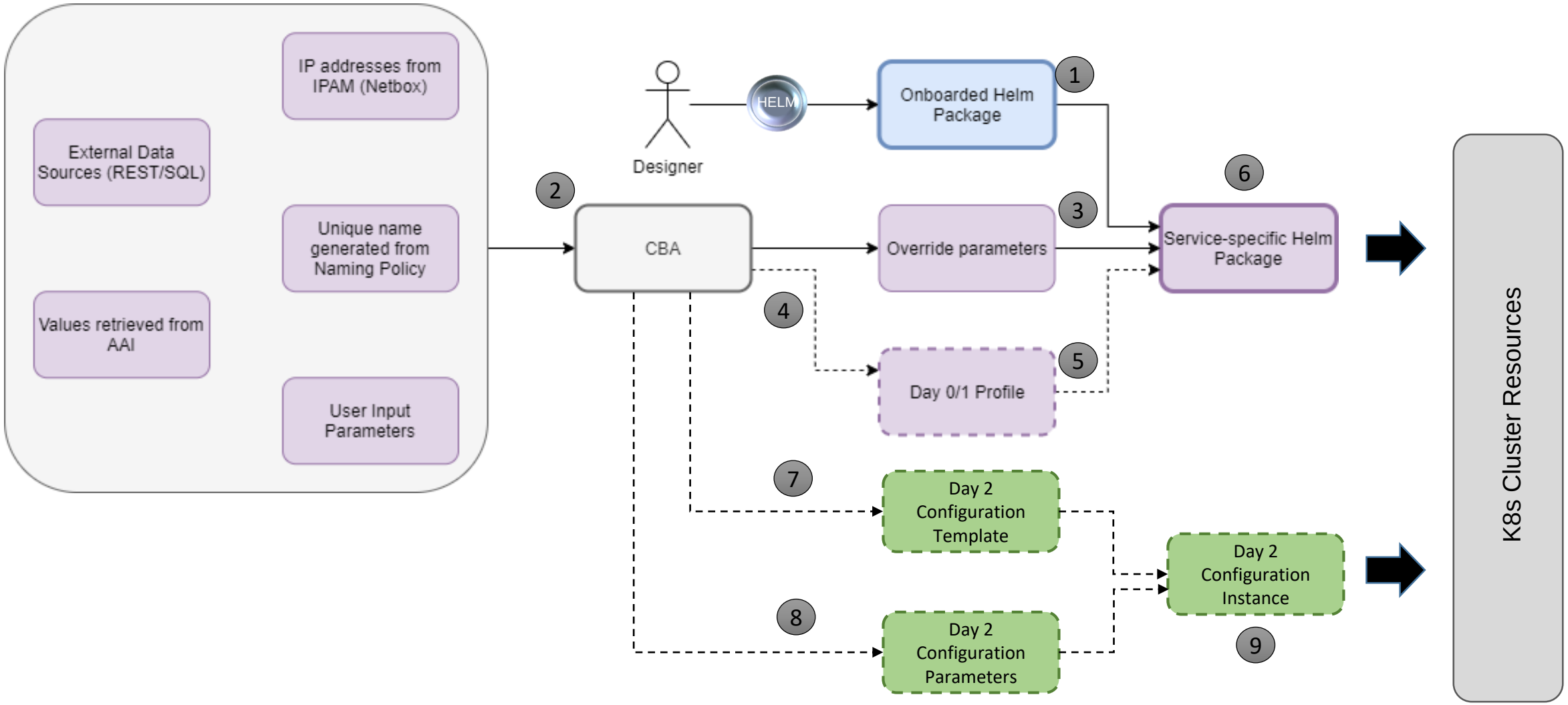
Guilin - K8s Adapter (Helm) Flow Day 0/1



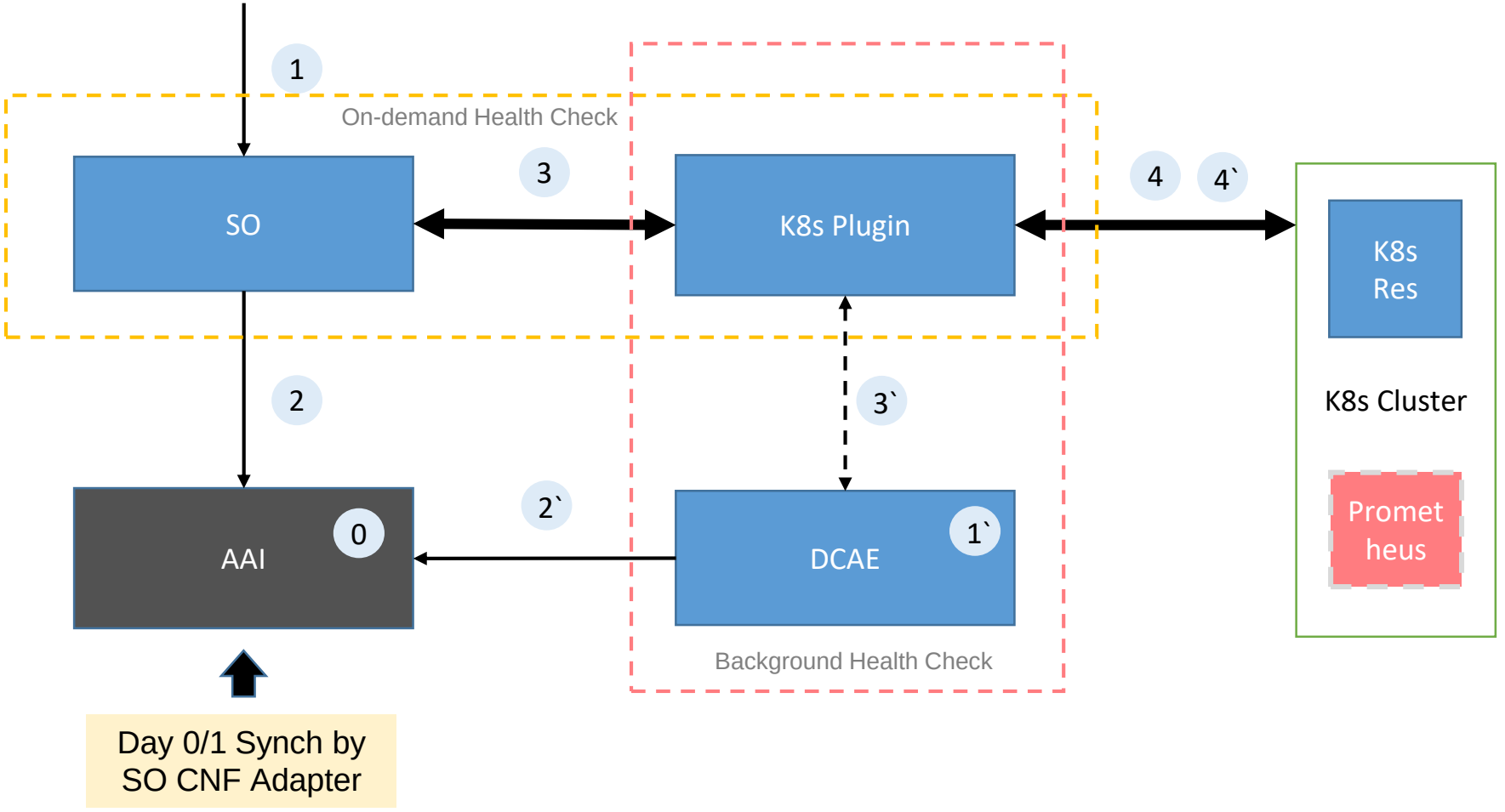
Focus on Native Day2 Operations

- AAI model changes
- SO AAI Data Update
- SO CNF Status
- SDC Distribution
- Helm Validation
- K8s Plugin – v2 APIs
- Native Day2 for CNF in CDS

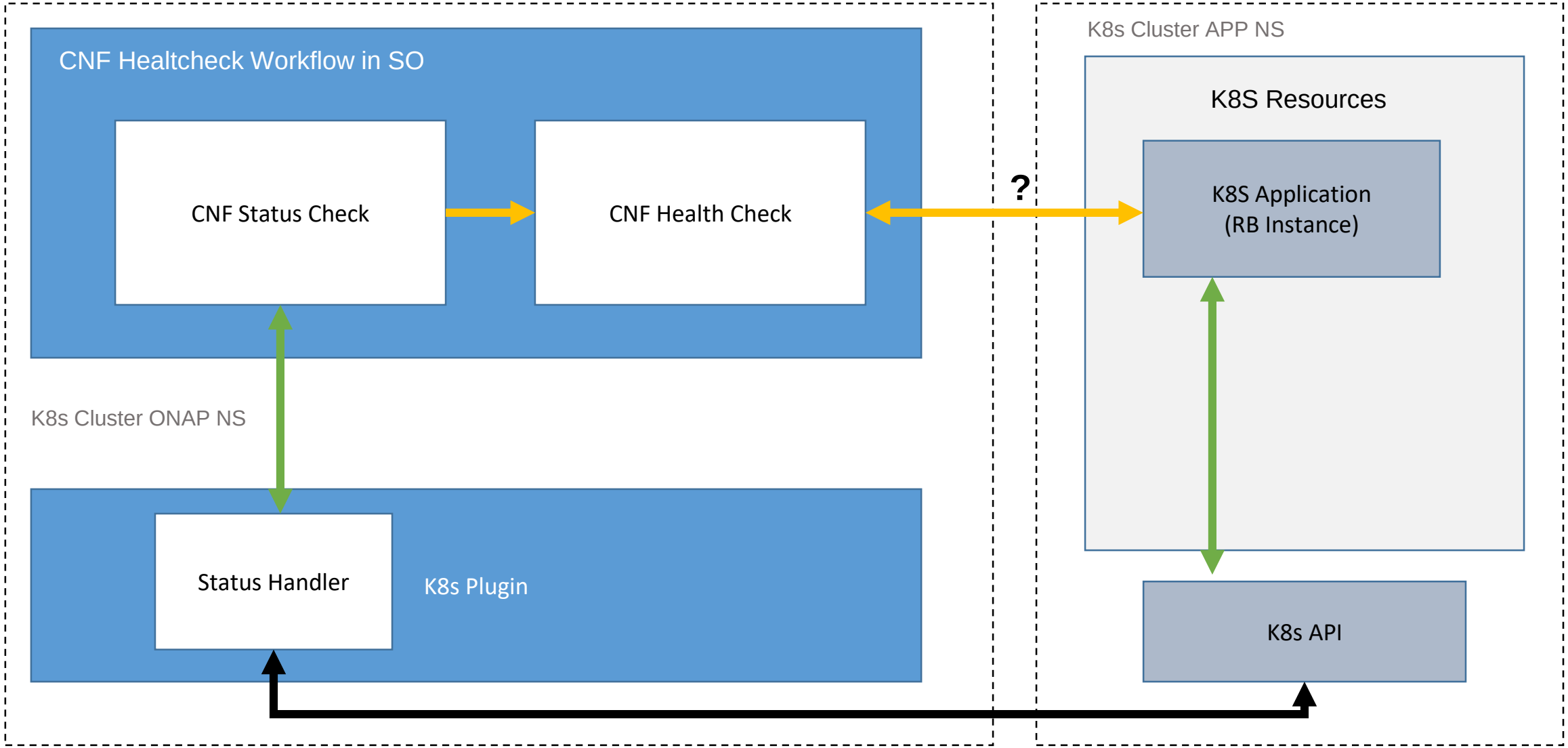
Helm Package Day 0/1 + Day2



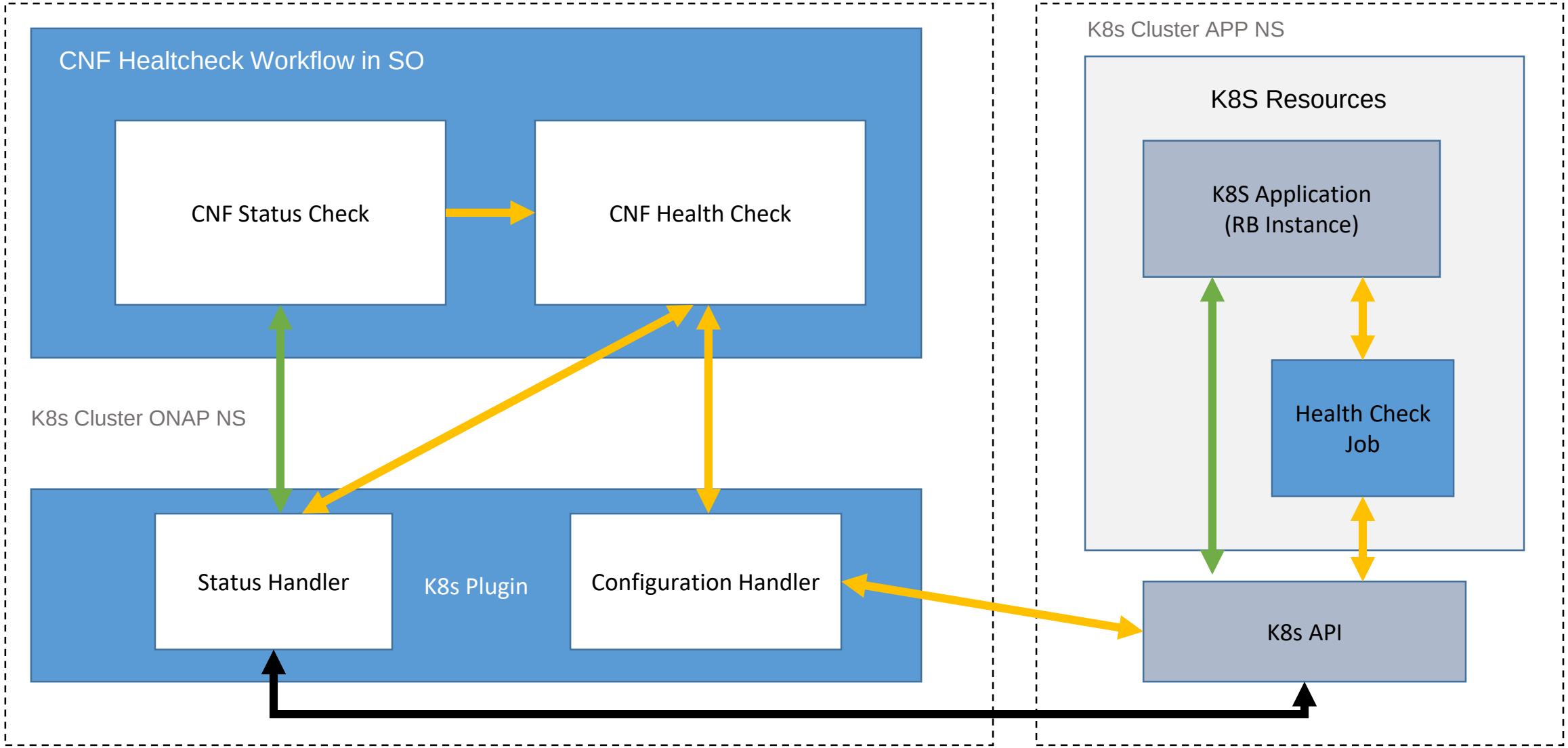
Day2 CNF Health Flow - ONAP



On-Demand CNF Health Check – Challenge



On-Demand CNF Health Check – Proposal



Proposed scope for REQ-458 - Honolulu+ (1)

REQ-458

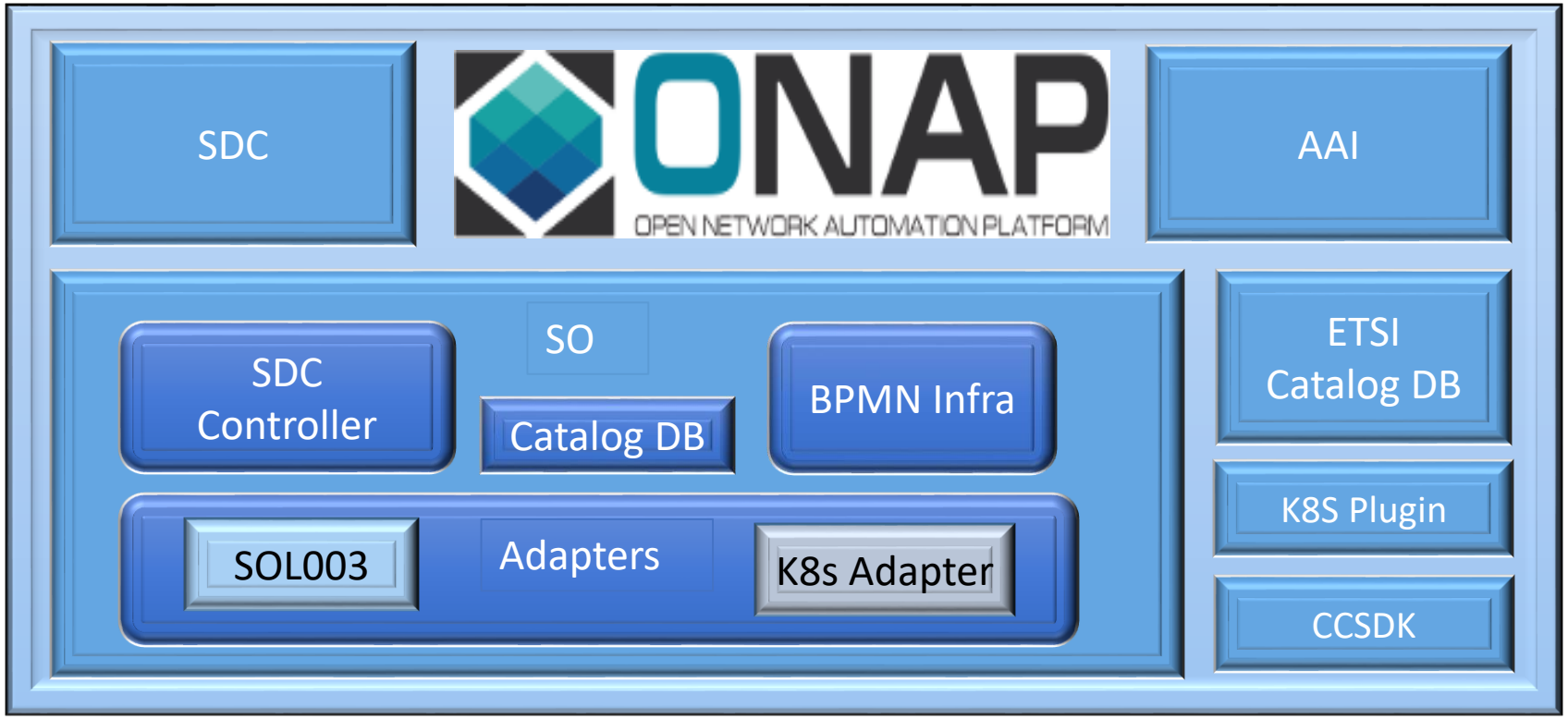
- SDC Enhancements
  - Continuation of native Helm support changes
  - Helm validation [stretch]
  - Artifact type recogniton by manifest type [stretch]
- AAI model changes
  - K8s resource type created from helm package -> similar role to vserver object
  - Snapshot of Status API result in AAI
- AAI API - Exposure of Status API result with conversion to JSON
- SO Changes
  - SO E2E API Improvements
  - SO CNF Adapter
    - Status API in CNF Adapter
    - AAI synchronization after each change -> Notification based
  - SO Integration ETSI Flow <- We need to make sure the flow will coexist with REQ-334

Proposed scope for REQ-458 - Honolulu+ (2)

REQ-458

- Integration of K8s API v2 -> Investment for the future development
  - Configuration API for v2
  - v2 in OOM + adaptation of existing helm charts for NFR
  - SO CNF adapter must be changed in SO
  - ArtifactBroker must be modified for v2 or replaced by CNF adapter distribution
  - Native Profile Handler in CDS must be switched into v2
  - v2 in ONAP python-sdk?
- CCSDK/CDS
  - Native Configuration API Handler for v1 or v2
  - Native Status API Handler for v1 or v2
- Dedicated CNF Health Check Workflow in SO
  - Status Check -> Status API result verification
  - CNF Health Check with Dedicated Health Check Job Execution
- **We may want to switch to another pure CNF use case**
  - CNF use case CBA + Integration scripts
  - Reference Health Check Job Implementation for selected CNF use case
  - **Prometheus for collection of metrics**

ONAP CNF Orchestration – Honolulu Impact



External (S)VNFM

Model to drive the flows:

- ✓ SDC to denote the flow of which VNFM should be used - similar to Orchestration Type
- ✓ Information model – optional
- ✓ Need to investigate the best place to have the meta data, we can perhaps use the existing fields

SDC:
On-board the helm and process it as an artifacts of the CSAR to be distributed

- ✓ On-board Helm Charts (Continuation)
- ✓ Helm Validation in SDC (Stretch)

SO:
Won't consume the Helm by itself but parse it and push it forward to other ONAP components

- ✓ Parse the CSAR, extract helm
- ✓ SOL003 adapter enhancements
- ✓ K8s Adapter – v2 + Status API
- ✓ Distribution of Helm from CNF adapter (Optional)

K8s Plugin:

- ✓ Configuration API in v2
- ✓ Artifact Broker supports v2
- ✓ v2 in OOM

AAI:

- ✓ Persist k8s instance identifier
- ✓ Persist k8s resources ID created from Helm
- ✓ Expose Status API result

CCSDK:

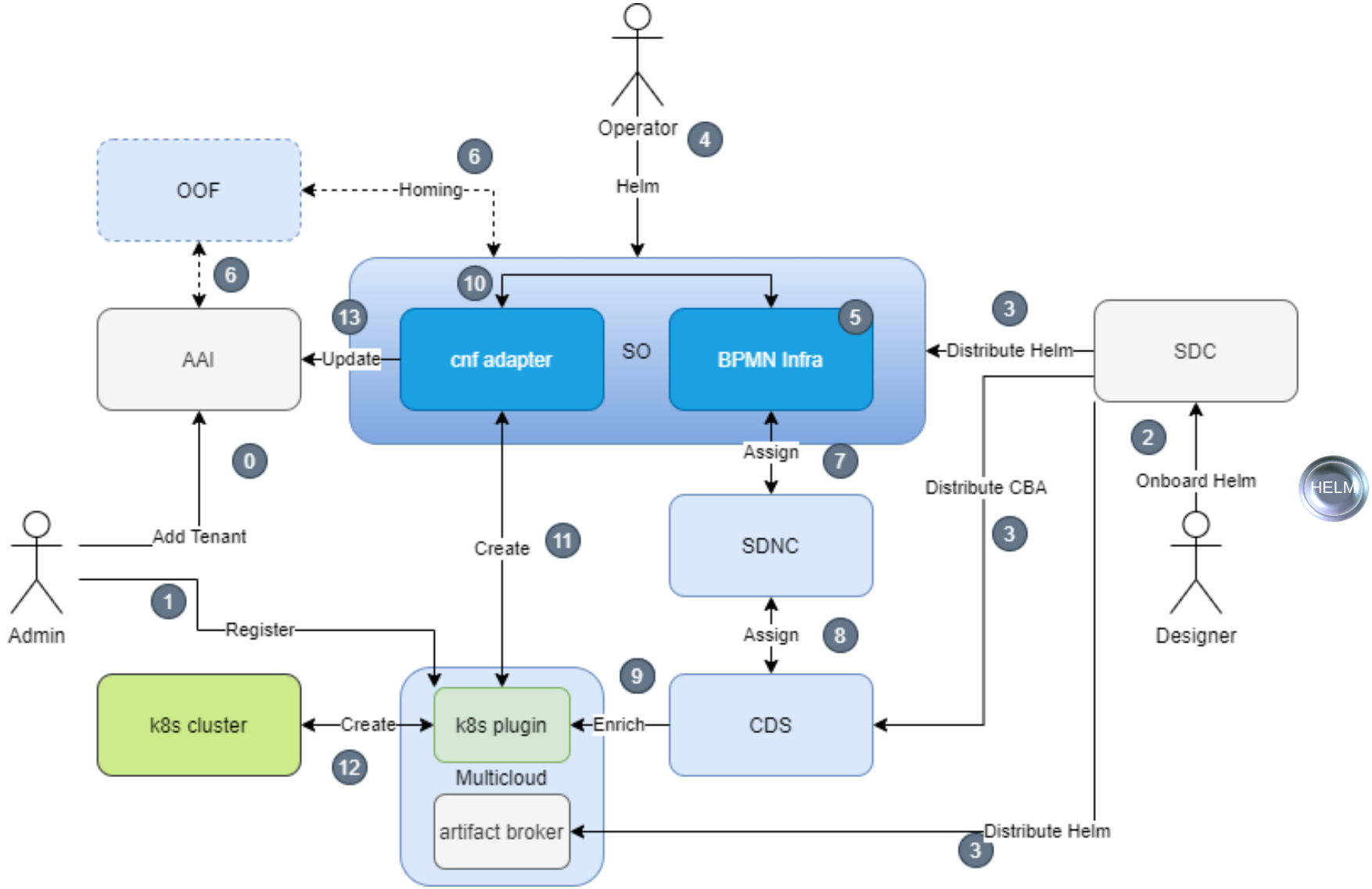
- ✓ Native suport of Configuration API
- ✓ Native Support of Status API
- ✓ Adaptation for v2

ETSI Catalog DB:

- ✓ Persist the ETSi VNFM data (IFA-29 and IFA-40)
- ✓ Persist Images

SDC

Guilin – CNF/Helm Day0/1 Flow



Helm Package – Structure v2 vs v3

Helm v2

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  requirements.yaml   # OPTIONAL: A YAML file listing dependencies for the chart
  values.yaml         # The default configuration values for this chart
  charts/             # A directory containing any charts upon which this chart depends.
  templates/          # A directory of templates that, when combined with values,
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

Helm v3

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values for this chart
  values.schema.json  # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file
  charts/             # A directory containing any charts upon which this chart depends.
  crds/               # Custom Resource Definitions
  templates/          # A directory of templates that, when combined with values,
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

Helm Package – Chart descriptor

Helm v2

```
apiVersion: The chart API version, always "v1" (required)
name: The name of the chart (required)
version: A SemVer 2 version (required)
kubeVersion: A SemVer range of compatible Kubernetes versions (optional)
description: A single-sentence description of this project (optional)
keywords:
  - A list of keywords about this project (optional)
home: The URL of this project's home page (optional)
sources:
  - A list of URLs to source code for this project (optional)
maintainers: # (optional)
  - name: The maintainer's name (required for each maintainer)
    email: The maintainer's email (optional for each maintainer)
    url: A URL for the maintainer (optional for each maintainer)
engine: gotpl # The name of the template engine (optional, defaults to gotpl)
icon: A URL to an SVG or PNG image to be used as an icon (optional).
appVersion: The version of the app that this contains (optional). This needn't be SemVer.
deprecated: Whether this chart is deprecated (optional, boolean)
tillerVersion: The version of Tiller that this chart requires. This should be expressed as a SemVer range: "
```

Helm Package - Structure

```
|-Chart.yaml
|-templates
|   |-network_attachment_definition.yaml
|   |-onap-private-net.yaml
|   |-protected-private-net.yaml
|   |-unprotected-private-net.yaml
|-values.yaml
```

- Helm Validation Required
- Should allow rejection of VSP/Helm package in time of VSP onboarding

```
|-Chart.yaml
|-charts
|   |-packetgen
|   |   |-Chart.yaml
|   |   |-templates
|   |   |   |-deployment.yaml
|   |   |   |-service.yaml
|   |   |   |-_helpers.tpl
|   |   |-values.yaml
|   |-sink
|   |-Chart.yaml
|   |   |-templates
|   |   |   |-configmap.yaml
|   |   |   |-deployment.yaml
|   |   |   |-service.yaml
|   |   |   |-_helpers.tpl
|   |   |-values.yaml
|-templates
|   |-deployment.yaml
|   |-onap-private-net.yaml
|   |-protected-private-net.yaml
|   |-unprotected-private-net.yaml
|   |-_helpers.tpl
|-values.yaml
```

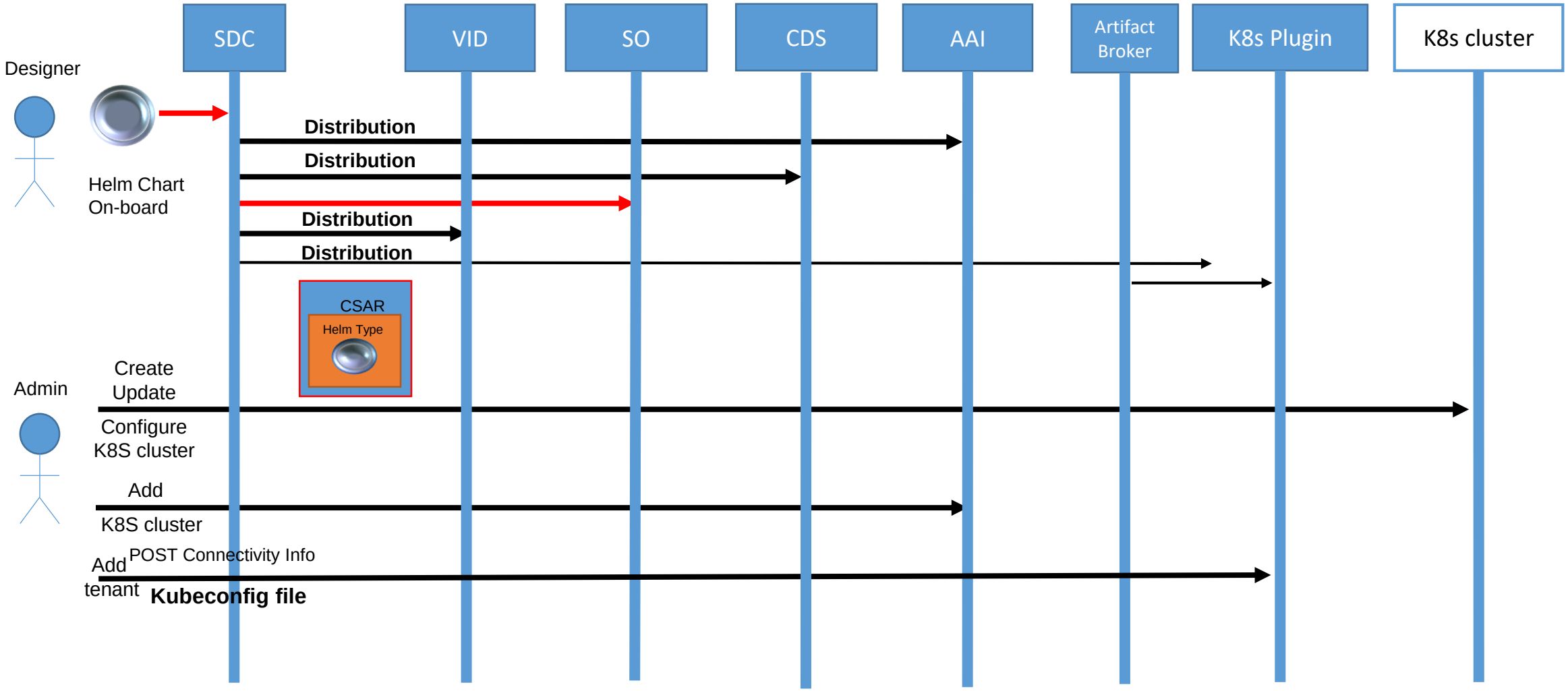
SDC - Onboarding Package with Helm

```
{
    "name": "virtualFirewall",
    "description": "",
    "data": [
        {
            "file": "base_template.yaml",
            "type": "HEAT",
            "isBase": "true",
            "data": [
                {
                    "file": "base_template.env",
                    "type": "HEAT_ENV"
                }
            ]
        },
        {
            "file": "base_template_cloudtech_k8s_charts.tgz",
            "type": "CLOUD_TECHNOLOGY_SPECIFIC_ARTIFACT"
        }

    ]
}
```

```
{
    "name": "virtualFirewall",
    "description": "",
    "data": [
        {
            "file": „helm_base_template.tgz",
            "type": "HELM",
            "isBase": "true"
        }

    ]
}
```

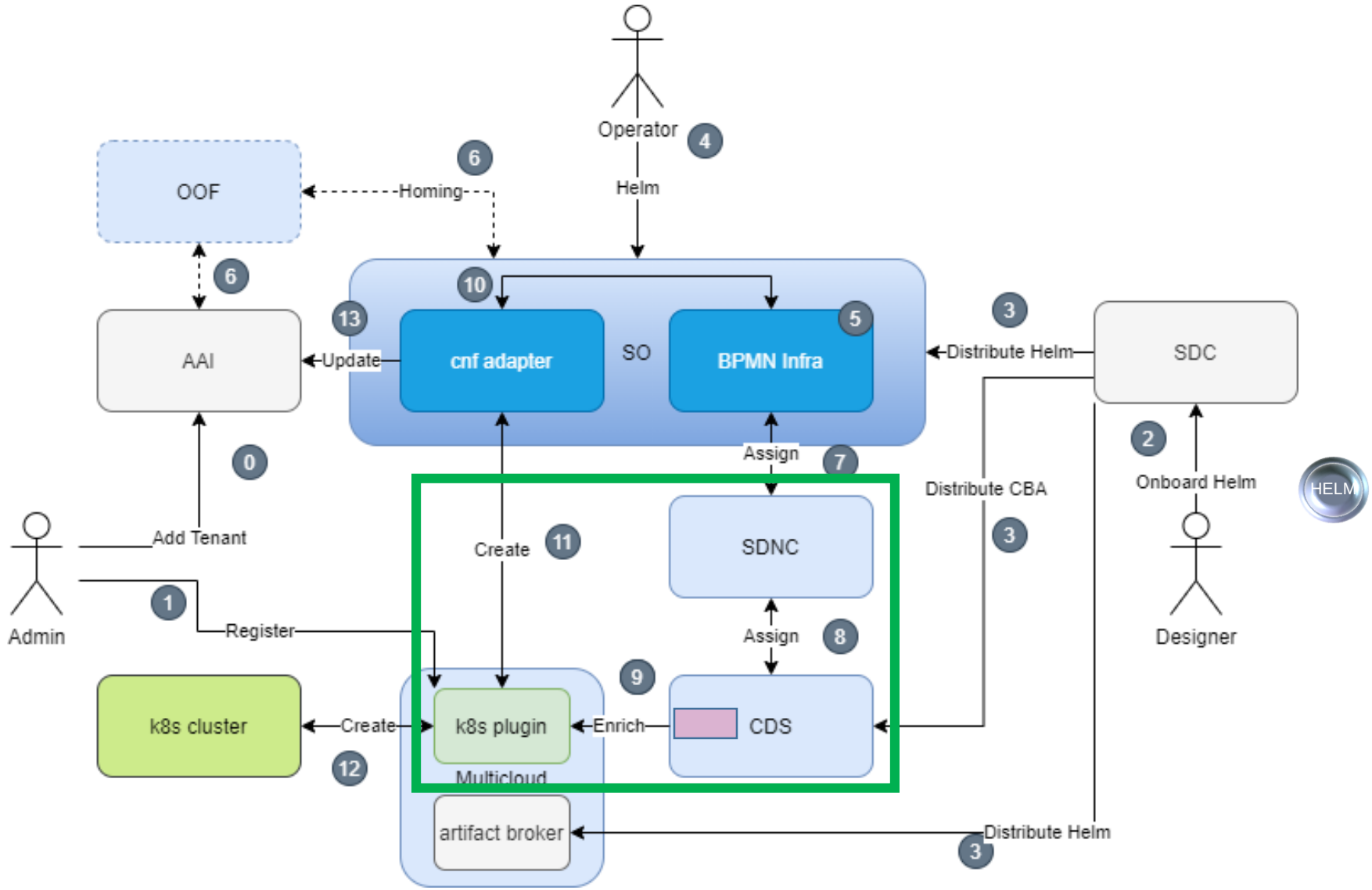
- Recognition of HELM type base on type in the manifest – not the name of file
  - Most probably change to be applied for general reading of type from manifest
- Removal of dummy_heat_ignore base template
  - Distribution of VSP without any heat templates
  - Base vf-module only heat

Guilin - Design and Distribution of the Helm Chart - Day 0

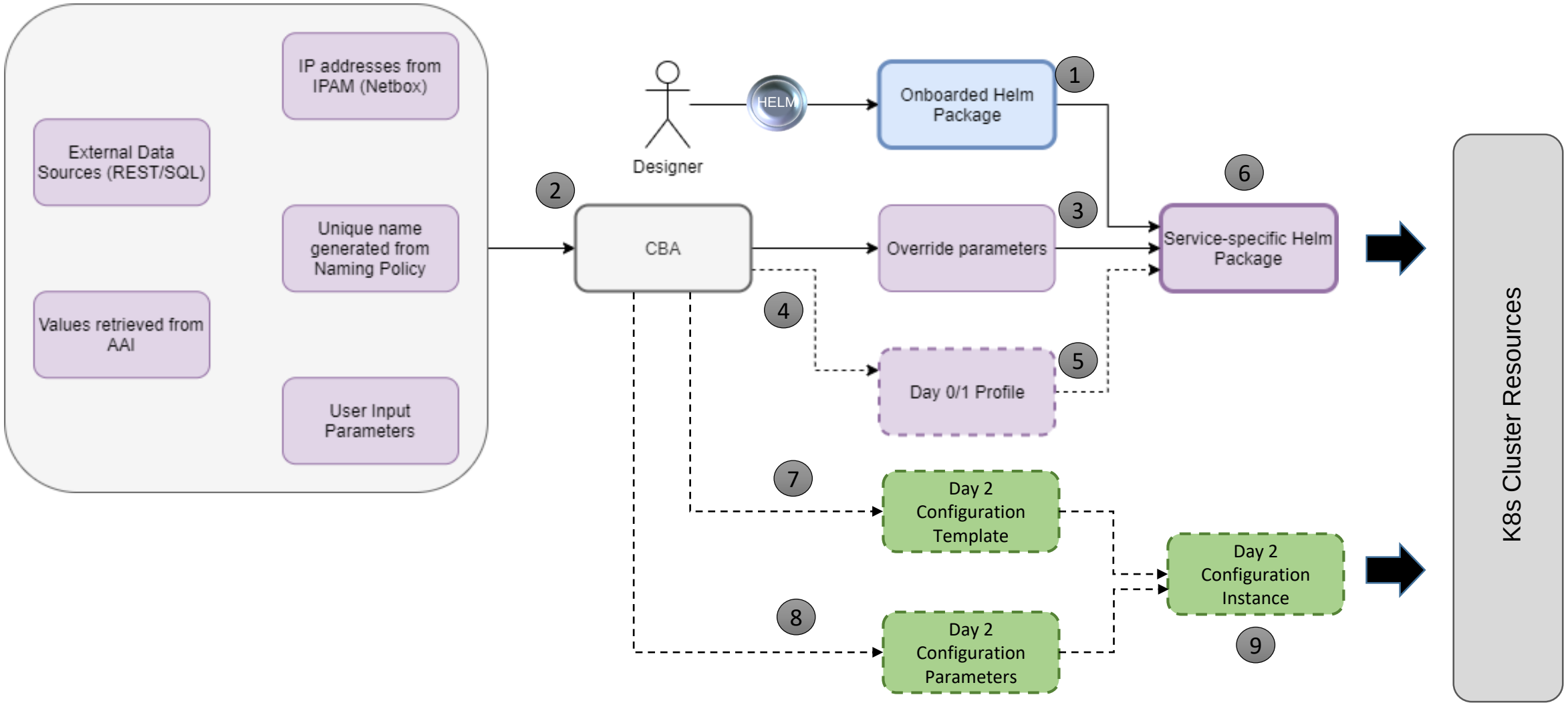


SDNC & CDS

CNF/Helm Day 0/1 Flow [Guilin]

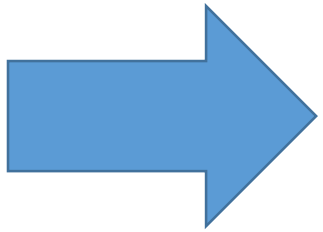


Helm Package Day 0/1 + Day2



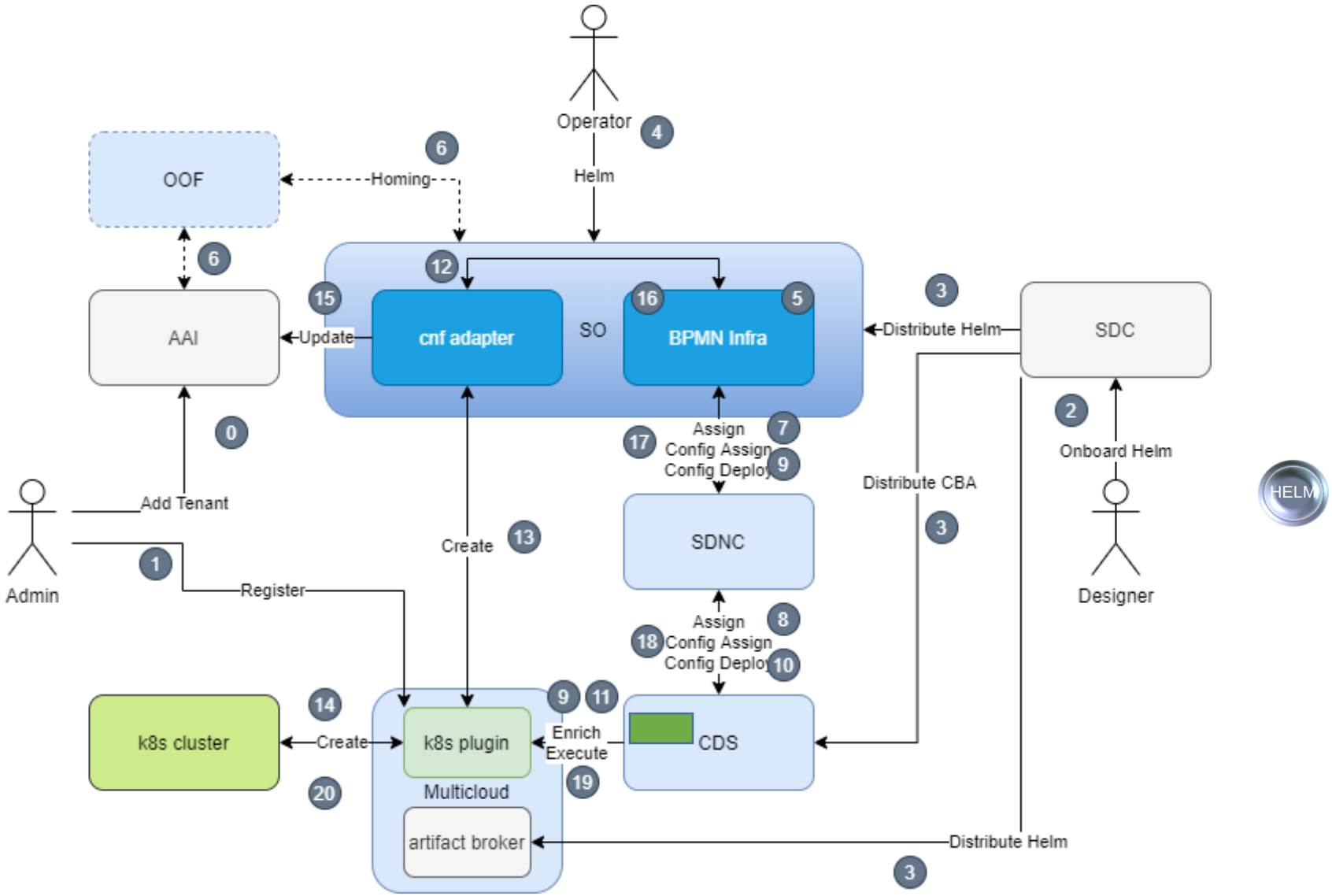
K8s Configuration Template and role of CDS [Honolulu]

```
|-Chart.yaml
|-templates
|   |-configmap.yaml
|-values.yaml
```

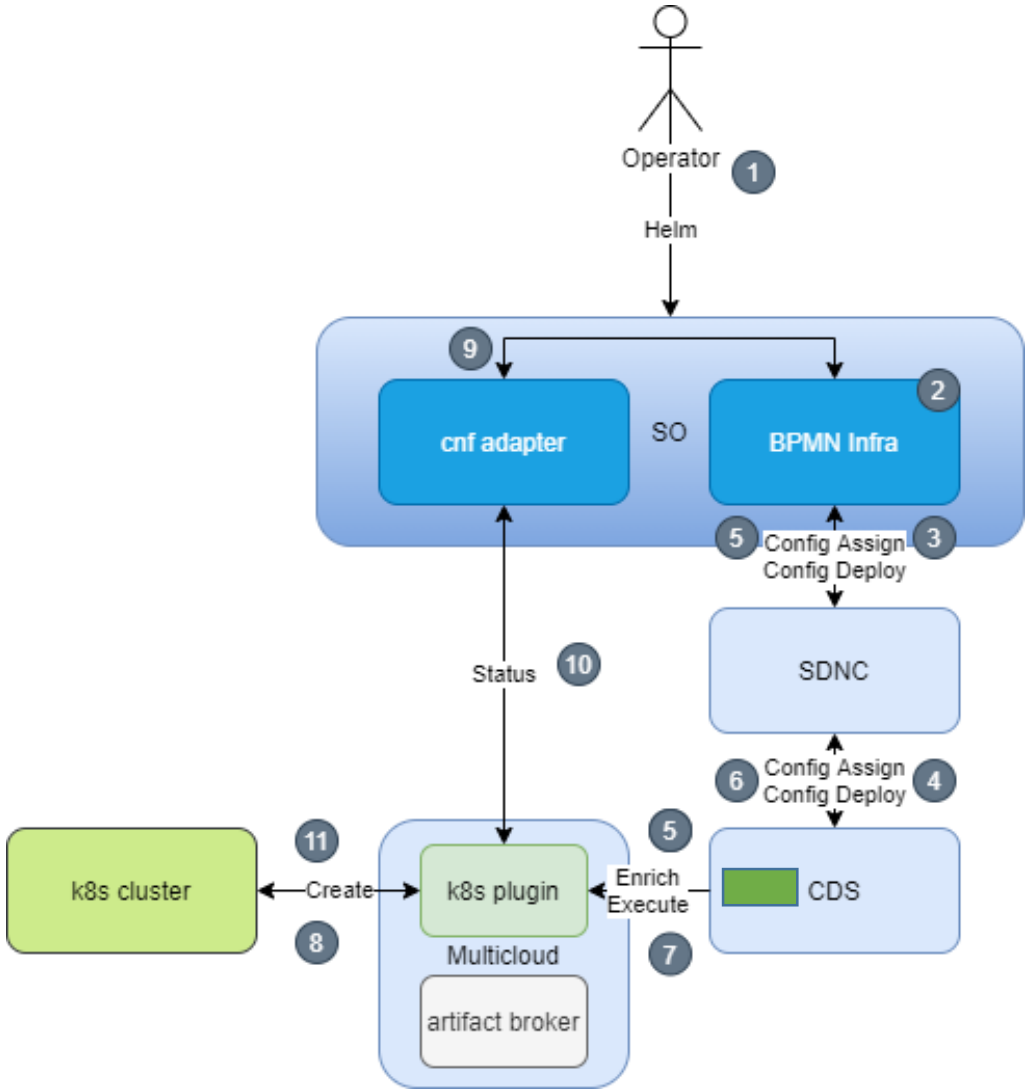


- Supplementary simple Helm Chart associated with Main Helm Chart
- Definition of Configuration Template may be hold in the CBA or may be generated in the CBA
- CDS will upload selected Configuration Template
- CDS will generate values for Configuration Instance and will create it

CNF/Helm Day 0/1 Flow with Config Assign/Deploy [Honolulu]



CNF/Helm Day 2 Flow [Honolulu]



- Separate flow after instantiation of CNF
- CDS Resolves Configuration Template
- CDS Uploads Configuration Template
- CDS Resolves
- CDS Creates Configuration Instance

RB Profile Native Upload [Guilin]

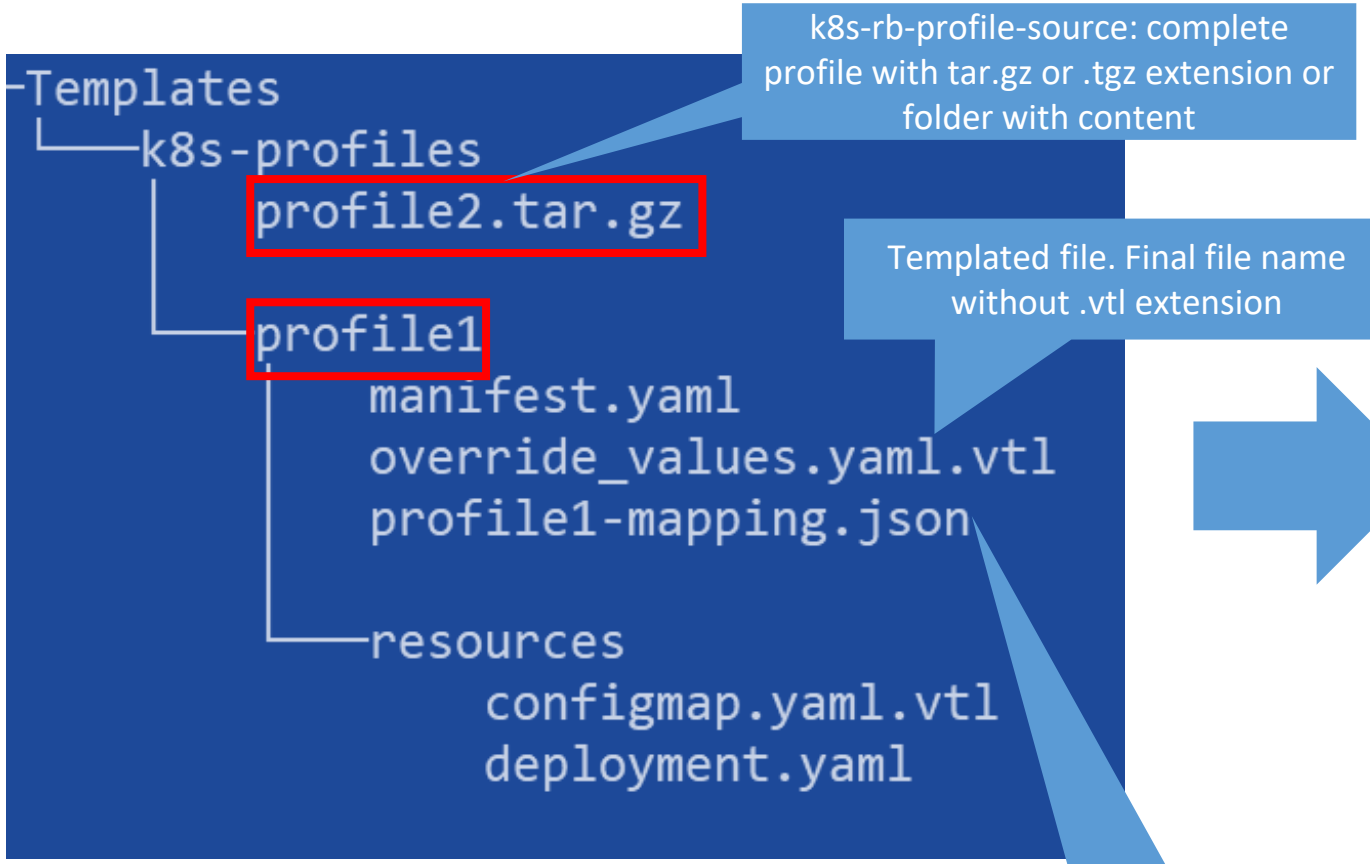
```
"k8s-profile-upload": {
    "type": "component-k8s-profile-upload",
    "interfaces": {
        "K8sProfileUploadComponent": {
            "operations": {
                "process": {
                    "inputs": {
                        "artifact-prefix-names": {
                            "get_input": "template-prefix"
                        },
                        "resource-assignment-map": {
                            "get_attribute": [
                                "resource-assignment",
                                "assignment-map"
                            ]
                        }
                    }
                }
            }
        }
    },
    "artifacts": {
        "vfw-cnf-cds-base-profile": {
            "type": "artifact-k8sprofile-content",
            "file": "Templates/k8s-profiles/vfw-cnf-cds-base-profile.tar.gz"
        },
        "vfw-cnf-cds-vpkg-profile": {
            "type": "artifact-k8sprofile-content",
            "file": "Templates/k8s-profiles/vfw-cnf-cds-vpkg-profile"
        },
        "vfw-cnf-cds-vpkg-profile-mapping": {
            "type": "artifact-mapping-resource",
            "file": "Templates/k8s-profiles/vfw-cnf-cds-vpkg-profile/ssh-service-mapping.json"
        }
    }
},
```

Here the k8s-rb* inputs will be taken from resource-assignment-map

Each profile source must be listed as artifact-k8sprofile-content artifact

If profile source is a folder it needs to have associated a mapping file

RB Profile Native Upload – Profile Manifest [Guilin]



k8s-rb-profile-source: complete profile with tar.gz or .tgz extension or folder with content

Templated file. Final file name without .vtl extension

Profile will contain only files mentioned in the manifest

```
---
version: v1
type:
  values: override_values.yaml
  configresource:
    - filepath: resources/deployment.yaml
      chartpath: templates/deployment.yaml
    - filepath: resources/configmap.yaml
      chartpath: templates/deployment.yaml
```

CBA profiling content

Definitions of values used for Velocity (vtl) templating

Manifest for profile1 source

Day2 Config template and upload [Honolulu]

```
"k8s-profile-upload": {
    "type": "component-k8s-profile-upload",
    "interfaces": {
        "K8sProfileUploadComponent": {
            "operations": {
                "process": {
                    "inputs": {
                        "artifact-prefix-names": {
                            "get_input": "template-prefix"
                        },
                        "resource-assignment-map": {
                            "get_attribute": [
                                "resource-assignment",
                                "assignment-map"
                            ]
                        }
                    }
                }
            }
        }
    },
    "artifacts": {
        "vfw-cnf-cds-base-profile": {
            "type": "artifact-k8sprofile-content",
            "file": "Templates/k8s-profiles/vfw-cnf-cds-base-profile.tar.gz"
        },
        "vfw-cnf-cds-vpkg-profile": {
            "type": "artifact-k8sprofile-content",
            "file": "Templates/k8s-profiles/vfw-cnf-cds-vpkg-profile"
        },
        "vfw-cnf-cds-vpkg-profile-mapping": {
            "type": "artifact-mapping-resource",
            "file": "Templates/k8s-profiles/vfw-cnf-cds-vpkg-profile/ssh-service-mapping.json"
        }
    }
},
```

New node type and modified interfaces structure

Rename artifact type to use common type across profile/day2 k8s nodes

Day2 Config Values apply [Honolulu]

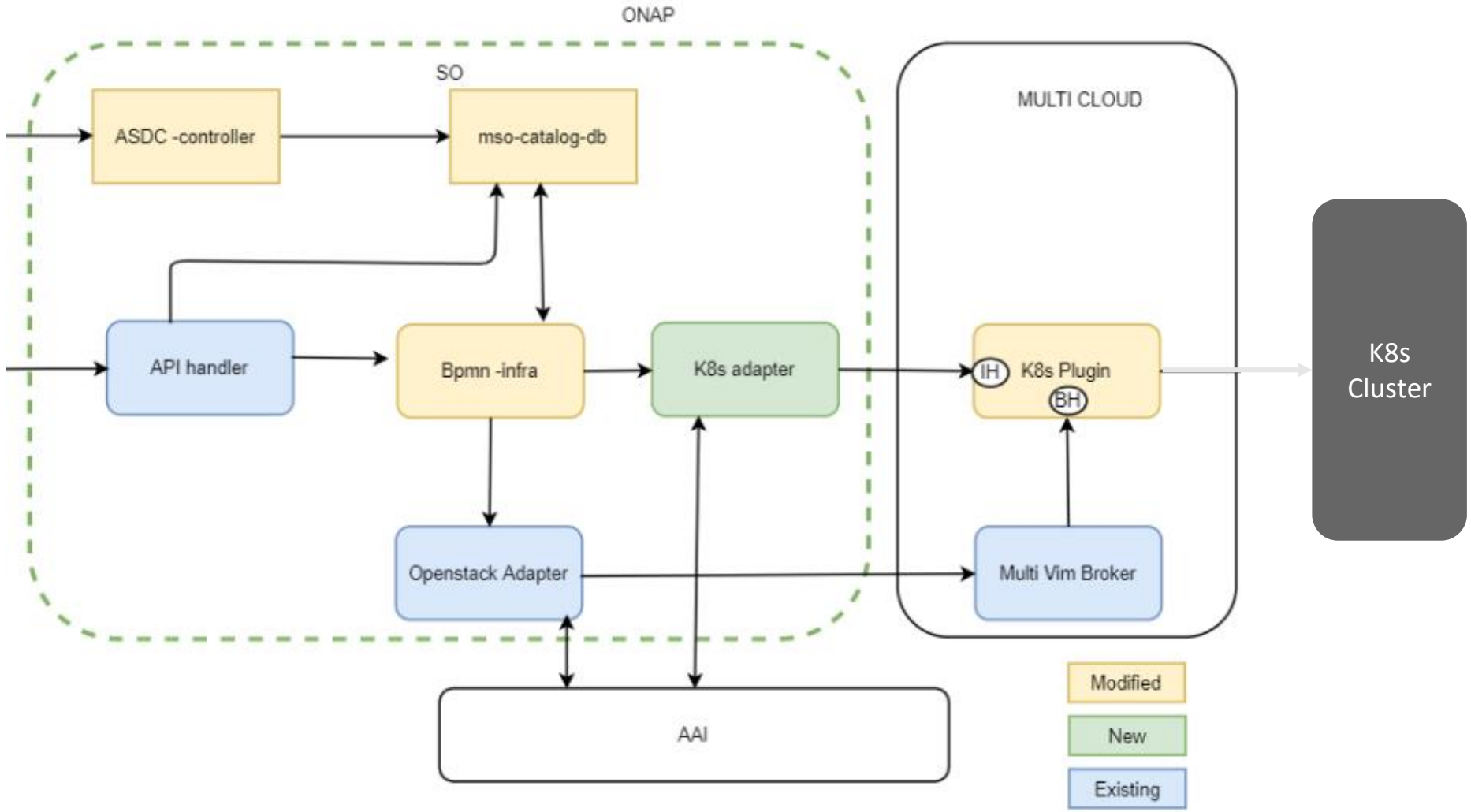
New node type and modified interfaces structure

Configuration values could be derived from assignment map …

… or from mapping file

Multicloud

K8splugin usage by SO (Guilin)



K8splugin REQchanges

- K8splugin changes in Honolulu release target “Instance Handler” API set, that is used by SO’s K8s Adapter for direct communication with k8splugin. They are not expected to impact existing Multicloud-based communication flow.
- Following development is tracked in MULTICLOUD-1233 story

Adaptation for Day2 Configuration API

- Existing Day2 Configuration API manages Day2 configuration objects on profile, not instance basis. This is due to initial assumption of k8splugin, that single CNF profile should be utilized by a single CNF instance.
- In Guilin this assumption has been loosed, enabling reuse of profile across many CNF instances. This, however, requires now to implement proper adjustments in Day2 Configuration API.
- Further adaptation may be implemented to respect CNF Instance instantiation-time parameters when templating Day2 Configuration objects.

Enhancements for Status API functionalities

- Implemented in Guilin, Status API allows status retrieval of (almost) all objects created and related to CNF Instance.
- To simplify user experience and integration with this API, in Honolulu it is expected to enhance it by accepting query parameters to limit response to selected objects. This feature will either expand Status API endpoint or create a new, dedicated one (Query API).
- **[Stretch]** Implementation of Publish/Subscribe capability of Status/Query API is planned to simplify operation of SO/K8s Adapter for further provisioning of this dynamic inventory information to AAI

Implementation of Healthcheck API

- To support implementation of CNF Healthcheck operation in SO, dedicated Healthcheck (Test) API is planned for implementation.
- This API would utilize capability already available for helm-based deployments - `helm test` - to run predefined set of jobs and retrieve their result
- Jobs utilized by this API would need to be defined in Helm package according to official Helm specification.
- Implementation of this API should also fix current K8splugin behavior of not filtering Test-defined jobs out from instantiation of CNF

Thank You!