ONAP kubernetes architecture/design proposal

Contributors: Isaku Yamahata, TBD

Introduction

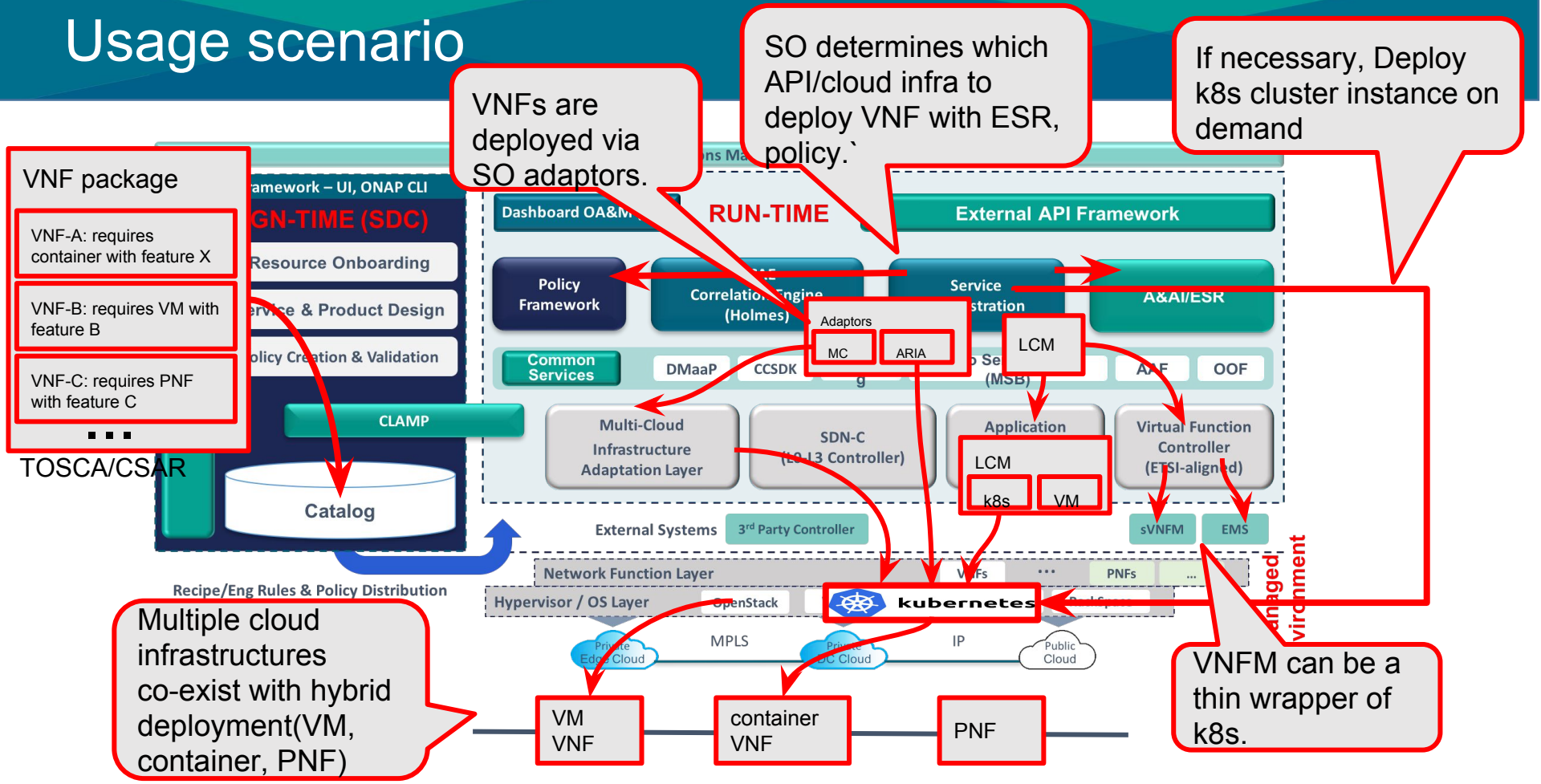
Container/COE is buzzing

Many benefits of container/COE technology

- container/COE has many cloud native features.
  - Easy deployment/version upgrade
  - Autoscaling/autohealing with lower response time
  - E.g. K8s: wisdom of google's past 15+ experience
- Industry trends
  - Take advantage of industry investment
  - Align with industry trends
- For more cloud native VNF

Let's take advantage of container/COE technology for ONAP

Usage scenario



Use Cases

- vCPE: edge cloud
  - Less hardware cost as container technology uses less resources than VM.
  - Simpler management
- 5G

Challenges: Technology gaps

container/coe technology may not be as mature as VM(openstack) technology. Especially

- Network
- Security
- VNF life cycle management with COE with consistency

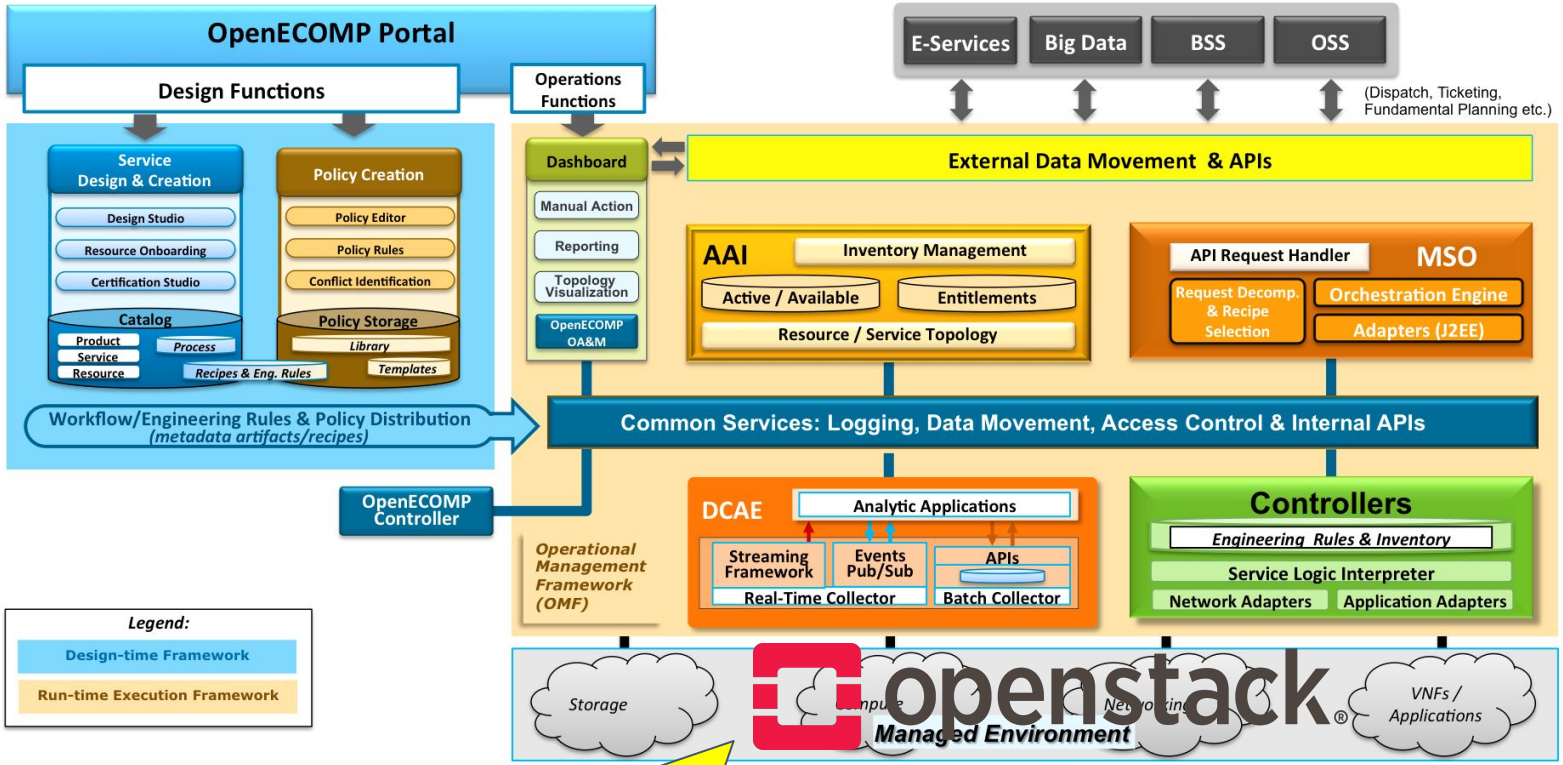
How do you address/guarantee those challenges.

Research activity would go out side of ONAP

Discussed in later slide

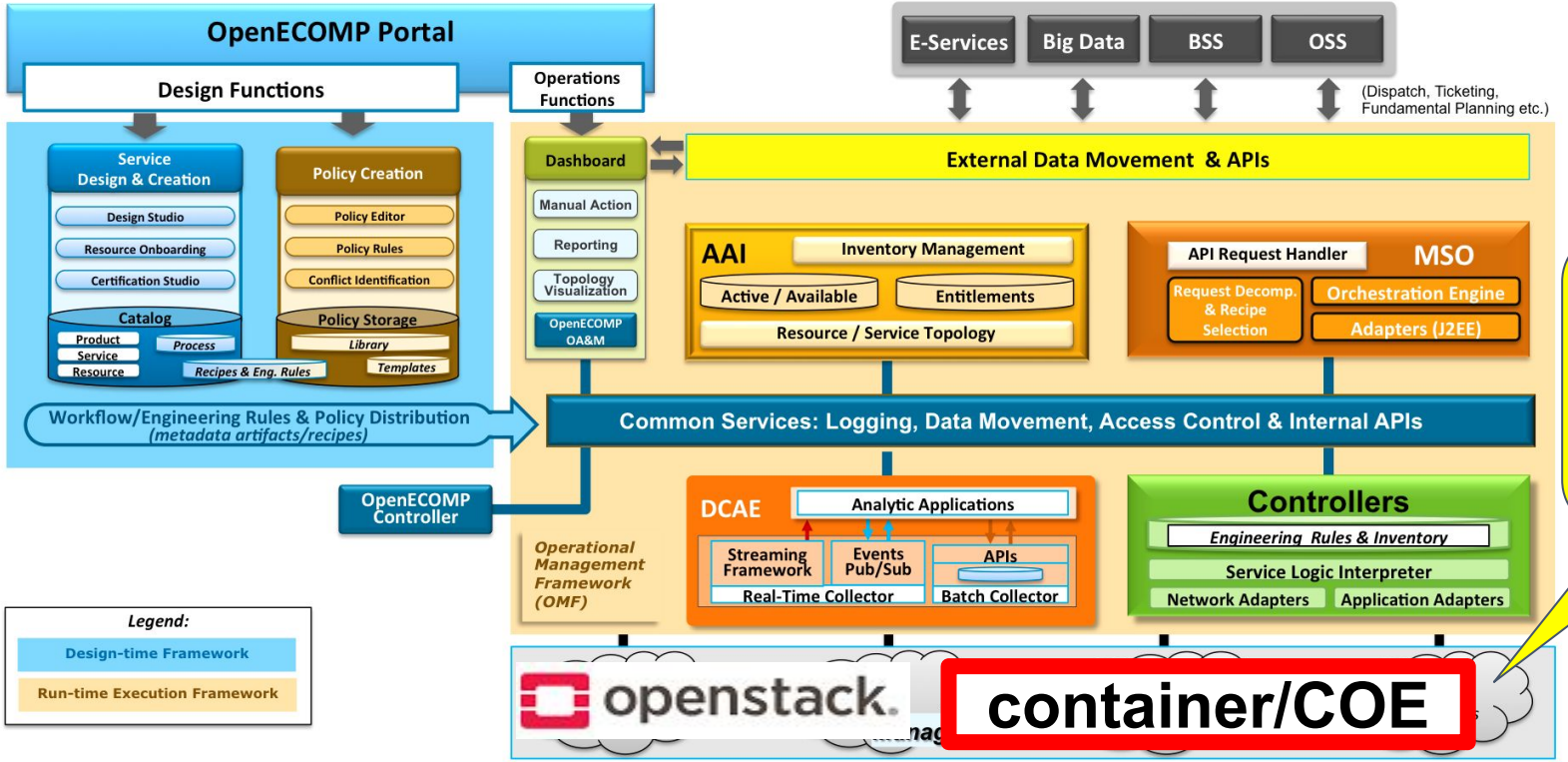
Goal and scope

Current ONAP architecture



Currently ONAP is heavily tied into OpenStack(or IaaS infrastructure)

Goal and Scope



Proposal: teach ONAP container/ COE

influence/ feedback/ contribute

Related projects (TOSCA, container, COE, security, etc...)

Add one more choice for cloud infrastructure

Goal and scope

Goal

- Have ONAP take advantage of container/COE technology for cloud native era
- Utilizing of industry momentum/direction
- Influence/feedback the related technologies(e.g. TOSCA, container/COE)
- Add more choices for VNF deployment/LCM in addition to IaaS(OpenStack)
  - It will co-exist with the existing component(VM cloud, PNF, VNFM under APP-C, VF-C)

Scope

- Teach ONAP container/COE in addition to openstack so that VNFs can be deployed/run/managed over container/COE in cloud native way

Non goal/out-of-scope

- Not installer/deployment. ONAP running over container
  - OOM project ONAP on kubernetes
  - https://wiki.onap.org/pages/viewpage.action?pageId=3247305
  - https://wiki.onap.org/display/DW/ONAP+Operations+Manager+Project
  - Self hosting/management might be possible. But it would be further phase.
- container/COE installer/deployment
- Replace the existing components. E.g. multicloud, APP-C, VF-C, ETSI VNFM/EMS
  - Goal is to give more choices with co-existence with already existing components
  - The result would be more adaptors/drivers/plugins in those projects
  - Some of them can be simplified by utilizing k8s features

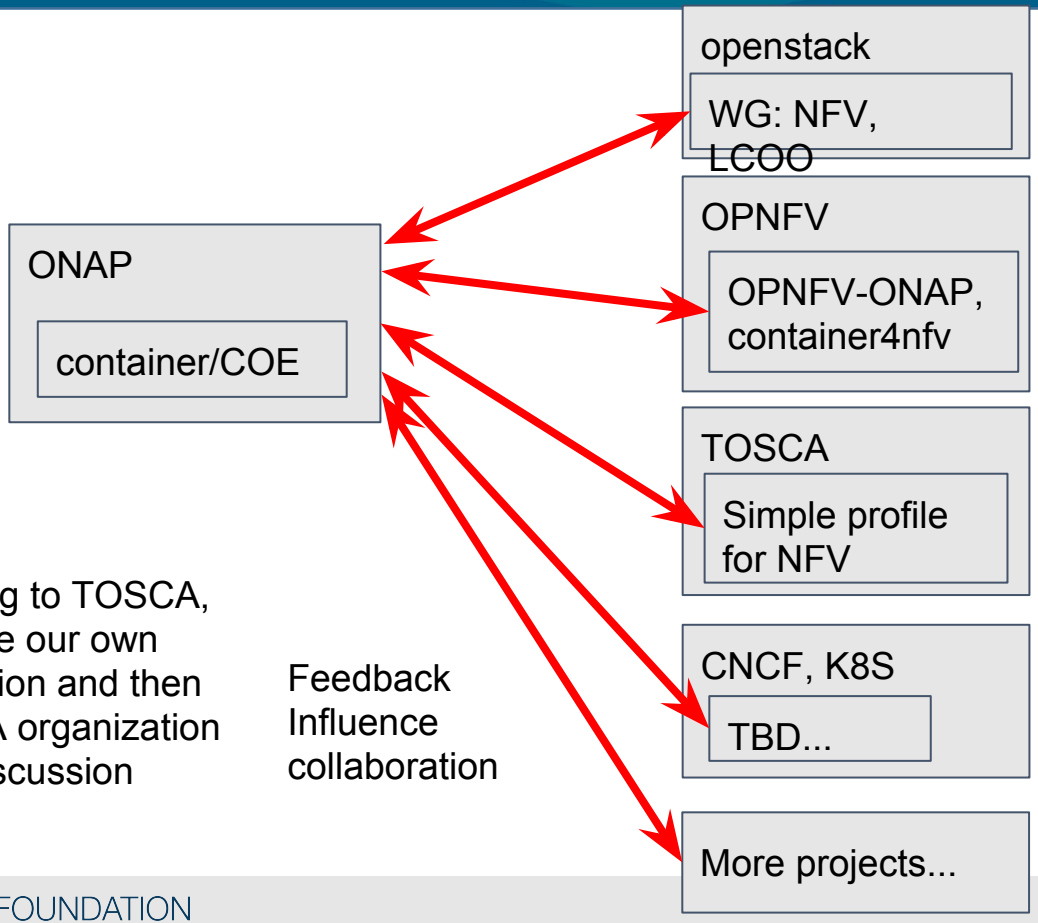
Non goal/out-of-scope(conti.)

- Address tech gaps of each elemental technology.
  - E.g. security, container networking, TOSCA enhancement as SPEC
  - ONAP scope is orchestration to make better use of them.
- Those research/engineering activity may take place outside of ONAP
  - With the collaboration of ONAP community
  - E.g. openstack, OPNFV
  - Potentially k8s, TOSCA
- ONAP would give feedback/influence on those activity

Technical gaps and research activity

- Research activity may take place outside of ONAP, e.g.
- Container networking: CNI
  - there are several technical gaps in container networking. The effort to fill those gaps will be done in their own community. ONAP will give feedback/influence based on our requirements/findings.
  - This would be k8s, CNI effort
- Security
  - There are concerns about container compared to VM. several technologies allows COE to run VM instead of container. Also hybride deployment
- Life cycle magement: TBD

Relationship to other projects



E.g. regarding to TOSCA, we can define our own nodes definition and then go to TOSCA organization to see the discussion outcome.

Feedback
Influence
collaboration

If necessary, engage to new communities and form task forces/projects in their community.

Principles

Design/architecture principle

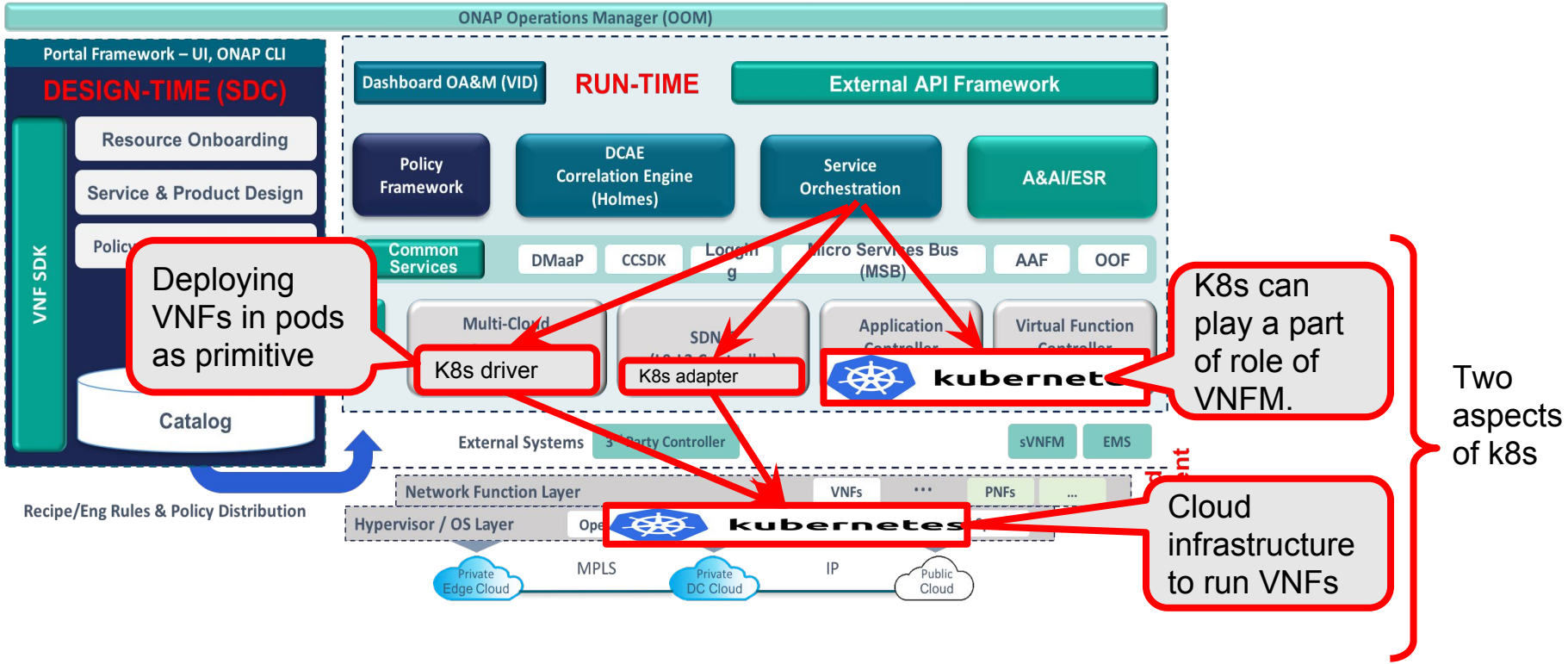
- ONAP should be able to take all the advantage of container/coe technology
- The design should not tie in to any single container technology
- design/architecture should align with other ONAP architecture/projects/future direction
- Should leverage the existing ONAP components
- Balance long term direction and short term achievement
  - E.g. TOSCA -> container API isn't available. It would take long time. Multiple steps will be needed. But at the same time we'd like to have something working/usable.

Design/Architecture principles (Cont.)

- Ideally ONAP components above multicloud, e.g. SO, controllers, should be agnostic to cloud infrastructure and untouched (or no major change) due to the introduction of K8s support. The difference of cloud infrastructure should be absorbed in multicloud
- The basic change should be, new type for cloud infrastructure and new features for it

What does container/COE means for ONAP

Where do COE functionalities fit in ONAP?



Feature overlap between SO/controllers and k8s

There are several overlapping features.

- Run application: (bare) pod
- Lifecycle management
  - VNF configuration: config map
  - Autoscaling/autohealing: replicaset
  - Monitoring: probe
- Rolling upgrade: rolling update
- Networking:
  - CNI, loadbalancer
  - DNS

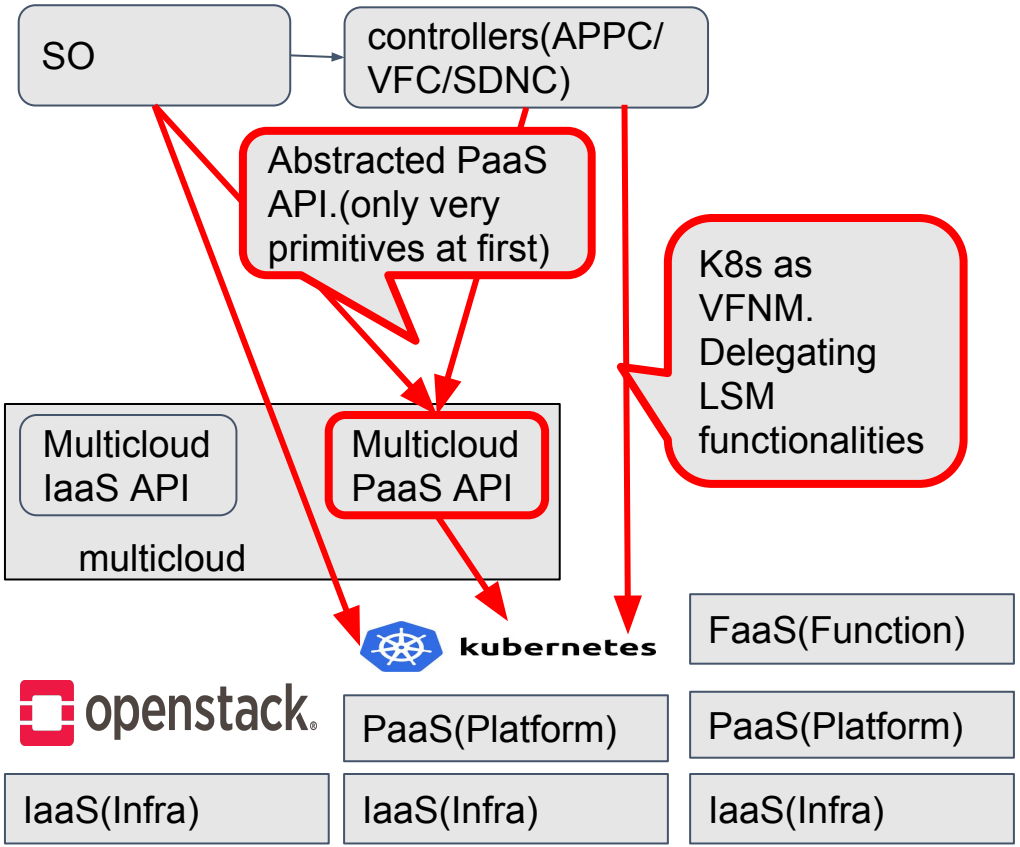
Multi cloud

APP-C/VF-C

SDN-C

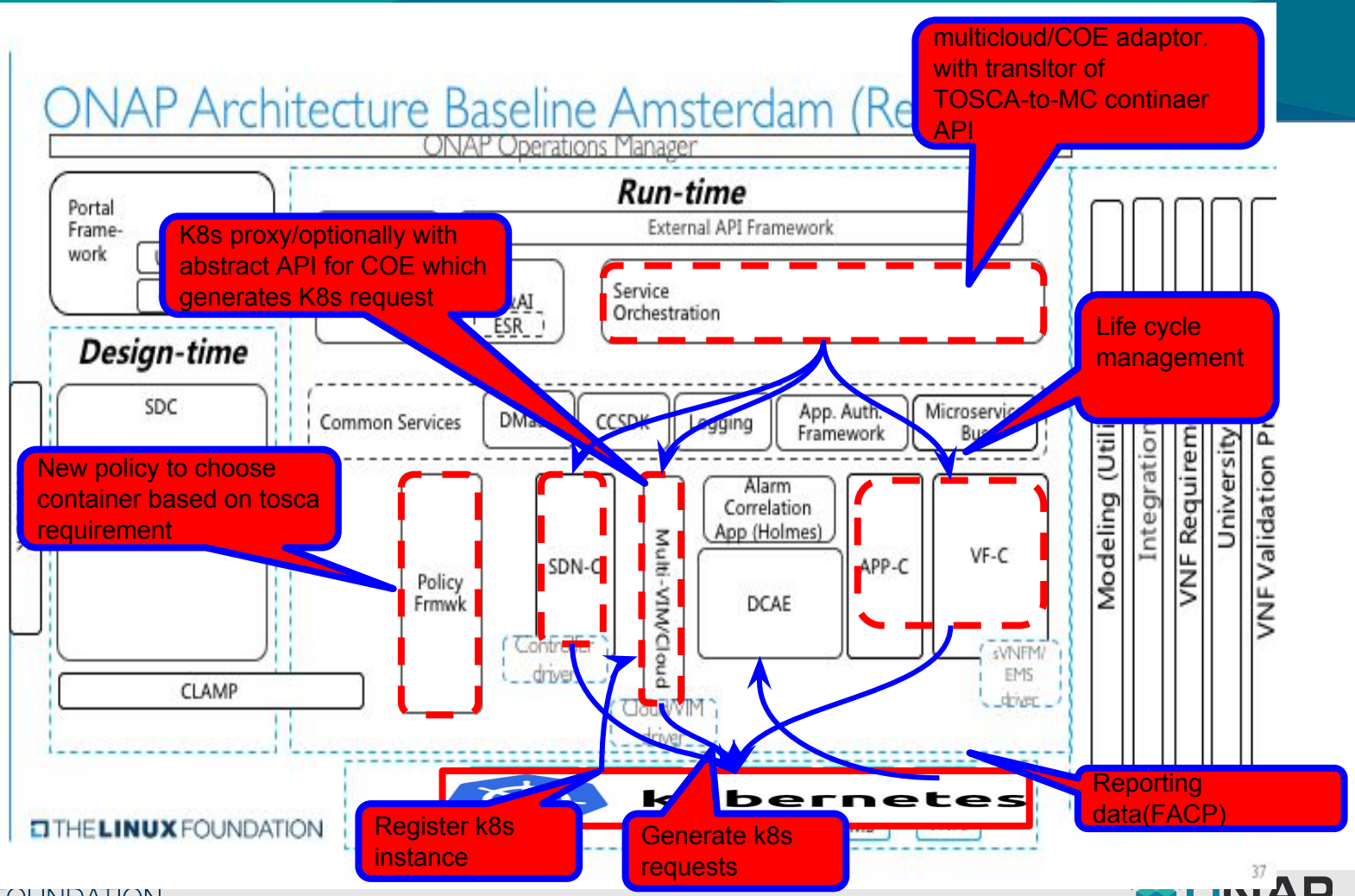
- In principle those features should be delegated to k8s because k8s has good features and many users wants to take advantage of it. At the same time if necessary/vnf desires, SO should be able to override it.
- K8s feature isn't always super set of SO's. In this case SO would implement with the help of k8s feature.
- Life cycle management
- Autoscaling/autohealing
  - replicaset/deployment/statefulset
- container probe
- Rolling upgrade
- Watching events

K8S functionalities and call flow



- Kubernetes is different layer of functionalities with overlap
- K8s on baremetal or VM cloud(openstack)
- K8s has, Node management, Autoscaling, monitoring/autohealing
- => a kind of VNFM
- VNFM would be just a thin wrapper of k8s

Proposed architecture/roadmap



https://wiki.onap.org/display/DW/Architecture

Roadmap proposed

scope: k8s on baremetal

Phase 1: only primitives
- Support primitives to deploy pods
- TOSCA enhancement to represent k8s requirement
- SO enhancement to know k8s.ability to Schedule VNFs onto k8s.

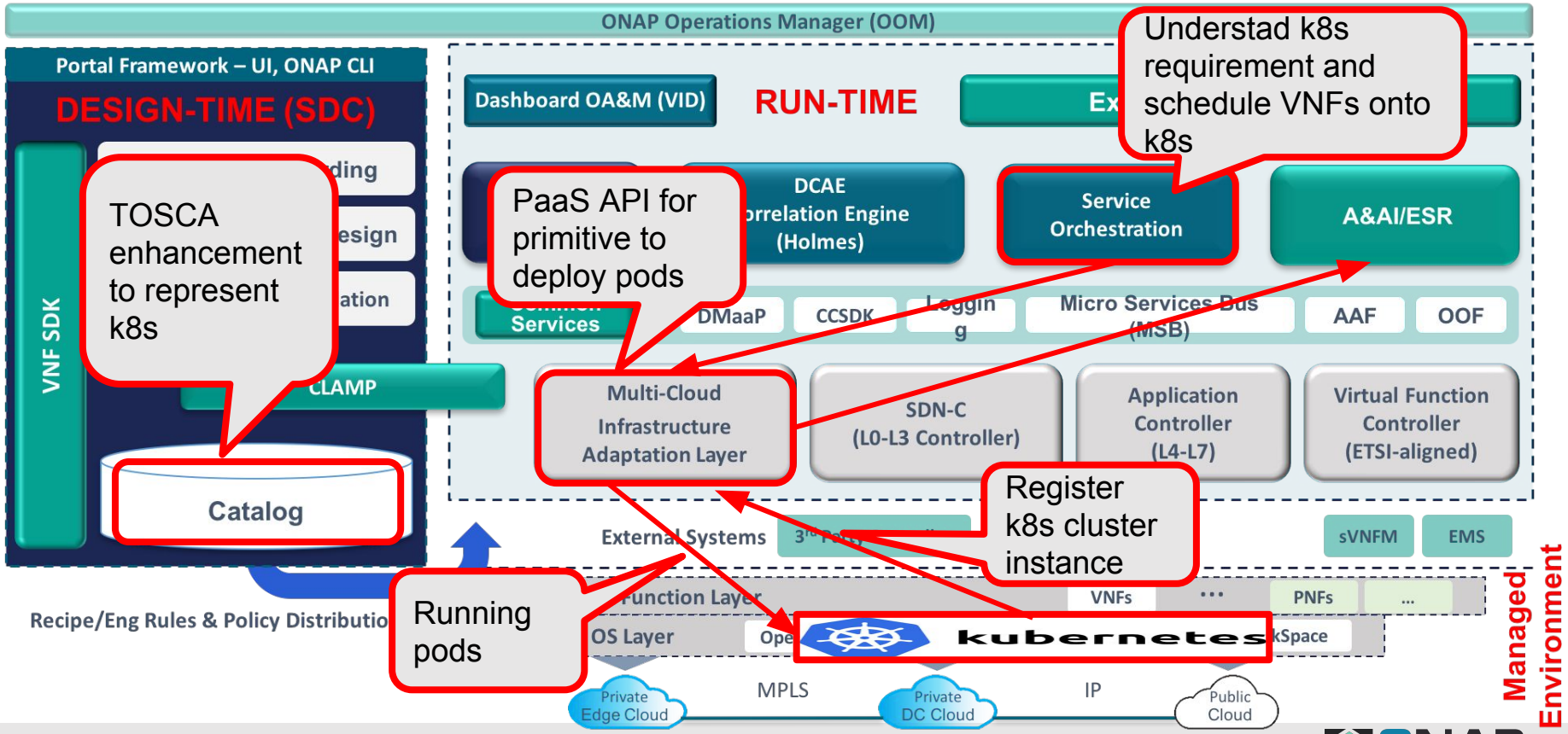
Scope: hyperscale container support

Phase 2: full functionalities
- Life cycle management with k8s as VFNM by controllers or directly SO
- APP-C/VF-C
- SDN-C: address networking
- More multicloud PaaS API
- Data management: closed loop feedback

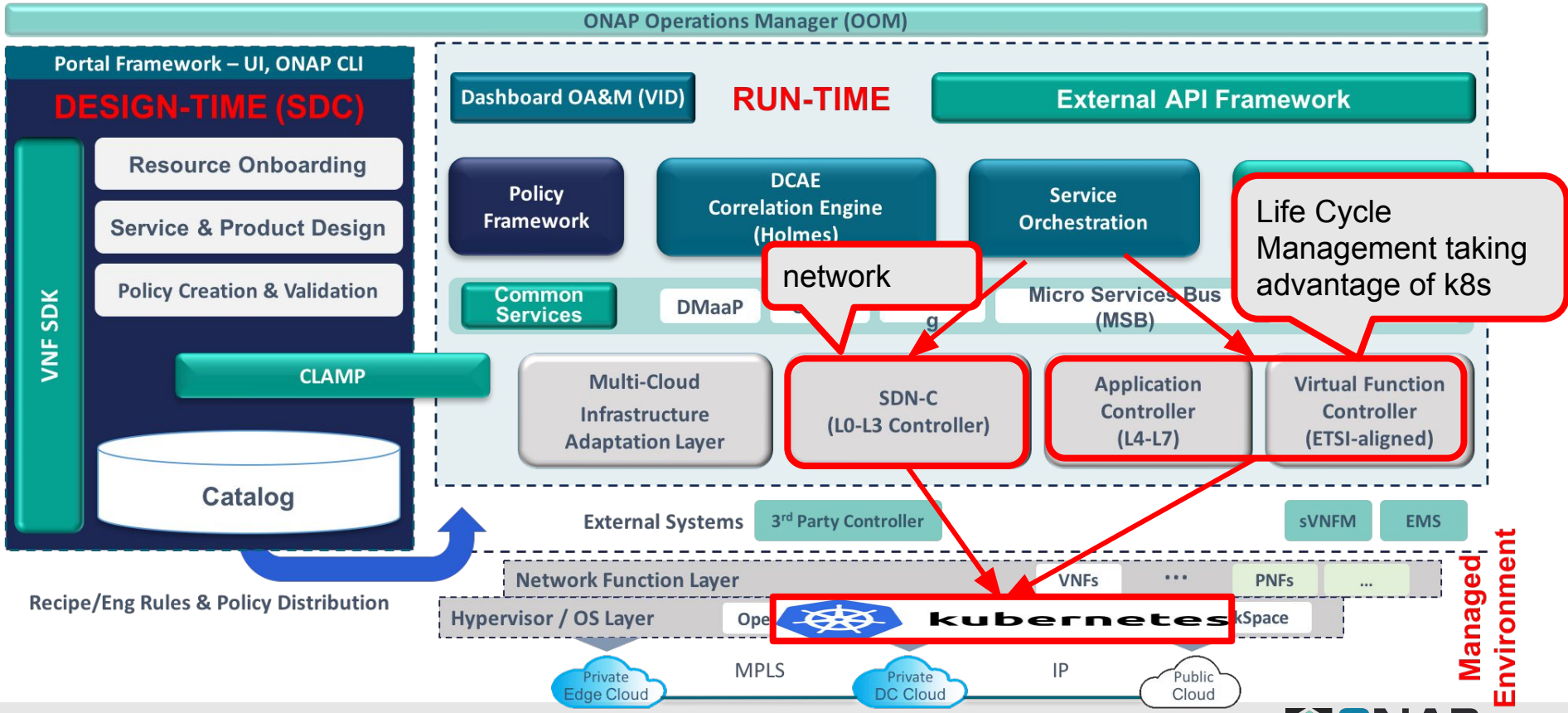
Scope: make use of k8s unique features

Phase 3: advanced functionalities
- Deploy/instantiate k8s cluster instances on demand.
- Scale kubernetes cluster instances
- K8s cluster federation
- (ONAP package and self managed ONAP service)
- Put your fancy features

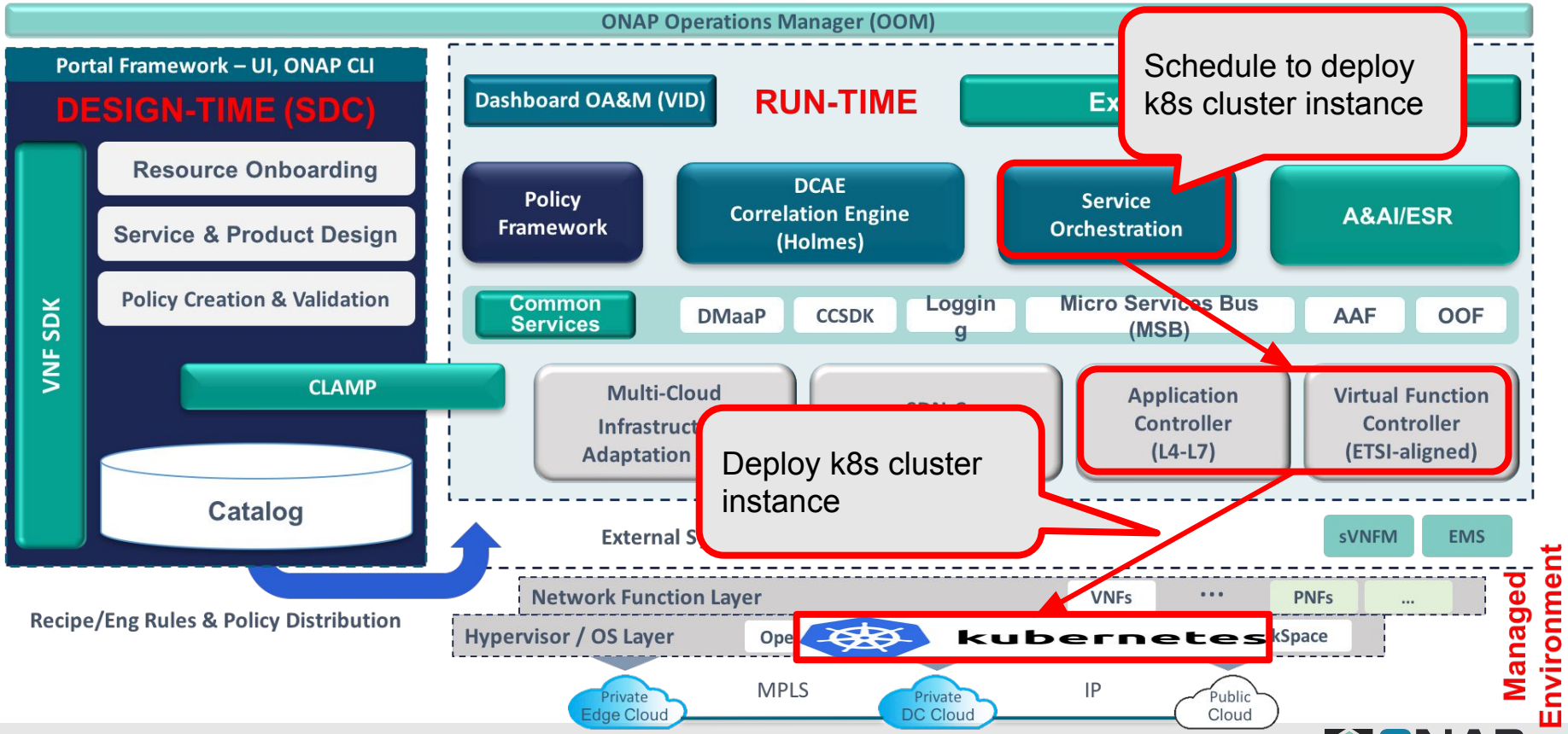
Phase 1: pod primitives



Phase 2: life cycle management(APP-C, VF-C)



Phase 3: k8s cluster mgmt



architecture/design choice discussion

architecture choices/discussion points: TBD

item	recommendation
Project home(new project or subproject of an existing project or task force?)	TBD: how to position this effort. TBD: show minimal projects to touch.
usecase	TBD
How to use container	Use container orchestration engine.e.g. k8s
API design: especially workload/lifecycle, model driven or not?	Model driven. Allow SO to use COE API directly. Multicloud as proxy: Open in multicloud discussion
Where to convert TOSCA to container API (if model driven). SO/controllers or multicloud	SO/controllers (eventually as CCSDK)
Source code repo	Subrepo of multicloud as starter

Project home: new project or existing project?

Project home	pros	cons
Multicloud subproject	Least overhead for project management	architecture/design/implementation would be tracted to the existing design
New project for any container	New design for container is possible. The scope would be much more than multicloud project.	Overhead of project management
New project for each container technology	ditto	There is commonality among container technology. Too much overhead of project management

How to use container

How to use container/coe	comment
Run container under VM based orchestartor. E.g. openstack nova-docker, zun, magnum)	Little benefit to use container
Use container (e.g. docker) and manage other resource by ONAP.	Resource(host/network/etc) orchestration layer is missing. Needs to be implement in ONAP
Use container orchestration engine. E.g. k8s/docker swarm/...	Full advantage of container/coe. And this functionality will be used by multi-cloud and as g-VNFM by APP-C

There are also technologies for COE to use VM. e.g. virtlet, kubevirt. It would make sense as VNF migration from VM to container as transition path.

Api choice	Align with multicloud direction?	New API? API consumer needs to be enhanced	container/coe feature can be easily utilized?	Other comment
Re-use the existing multicloud API: Coerce into the existing VIM API	Align with the existing code. No with future direction.	No	No	
Define new API for container	No with future direction	Yes	Yes	
Expose container/COE API directly	No with future direction	Yes	Yes	heavily depends on container/coe technology
Model driven API: without enhancement: coerce into model driven VM based API	yes	No. no additional changes to API consumer	No	
Model driven API: With enhancement. Probably in long term, contributing to TOSCA	Yes with future direction.	Yes	Yes	Needs to implement/invent conversion logic from TOSCA to container/COE Reasonable abstraction among container/COE technologies

Where to translate TOSCA to K8S: Open

Now under discussion: Fuel for discussion

Eventually follow the community decision

Where to convert	comment
caller(adaptor in SO/CCSDK)	Multiple place to host code/process it
Callee (in multicloud or new project)	Single place to host code/process it

Source code repository

Source code repository	comment
Subdirectory of one of the multicloud repositories	No appropriate repo
New subrepo for any container under multicloud	Doesn't align with the current multicloud practice
New subrepo under Multicloud per container tech	Each container technology support can be evolved independently. Least overhead for repo management
New repo for any container technology under new project	
New repo per container technology under new project	Each container technology support can be evolved independently.

Summary and next steps

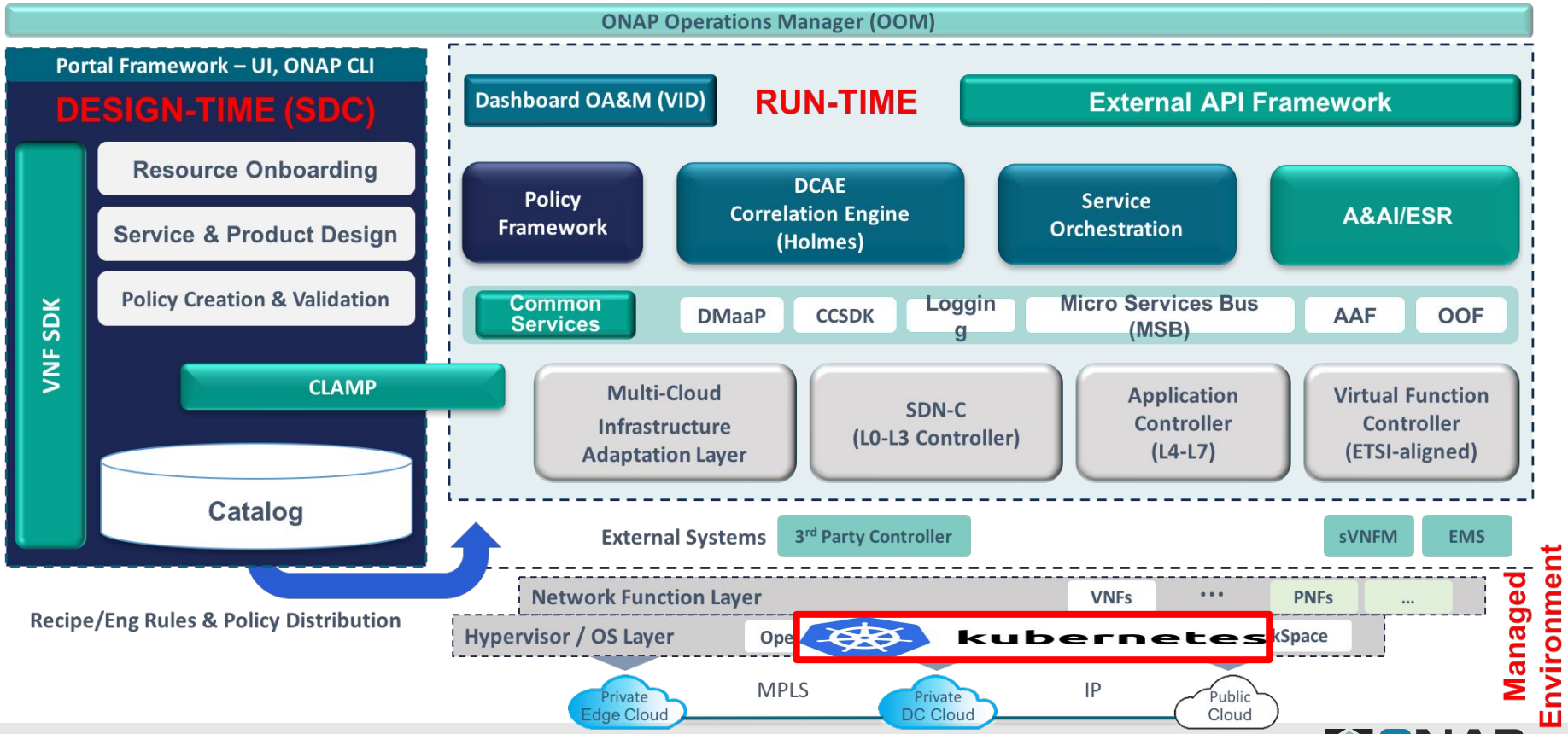
architecture choices

item	recommendation
Project home(new project or subproject of an existing project?)	New projet
How to use container	Use container orchestration engine.e.g. k8s
API design: especially workload/livecycle, model driven or not?	Model driven. Allow SO to use COE API directly. Multicloud as proxy: Open in multicloud discussion
Where to convert TOSCA to container API (if model driven). SO/controllers or multicloud	SO/controllers (eventually as CCSDK)
Source code repo	Subrepo of multicloud as starter

Next steps

- Agree on its design/architecture and project home
  - Usecases, story, integration(containerised VNFs)
    - 5G, edge cloud
- Define the next development scope and propose new project
  - Feature, functionality and task
  - Release-3?
    - activity can start from R-2 and in R-3 the effort can be official.
  - Pick one usecase and contenairize it: vCPE?
- participants start development

backup


ONAP Operations Manager (OOM)
Portal Framework – UI, ONAP CLI
DESIGN-TIME (SDC)
VNF SDK
Resource Onboarding
Service & Product Design
Policy Creation & Validation
CLAMP
Catalog
Dashboard OA&M (VID)
RUN-TIME
External API Framework
Policy Framework
DCAE Correlation Engine (Holmes)
Service Orchestration
A&AI/ESR
Common Services
DMaaP
CCSDK
Loggin g
Micro Services Bus (MSB)
AAF
OOF
Multi-Cloud Infrastructure Adaptation Layer
SDN-C (L0-L3 Controller)
Application Controller (L4-L7)
Virtual Function Controller (ETSI-aligned)
External Systems
3rd Party Controller
sVNFM
EMS
Recipe/Eng Rules & Policy Distribution
Network Function Layer
VNFs
PNFs
Hypervisor / OS Layer
kubernetes
MPLS
IP
Private Edge Cloud
Private DC Cloud
Public Cloud
Managed Environment

ONAP, TOSCA to K8S mapping analysis

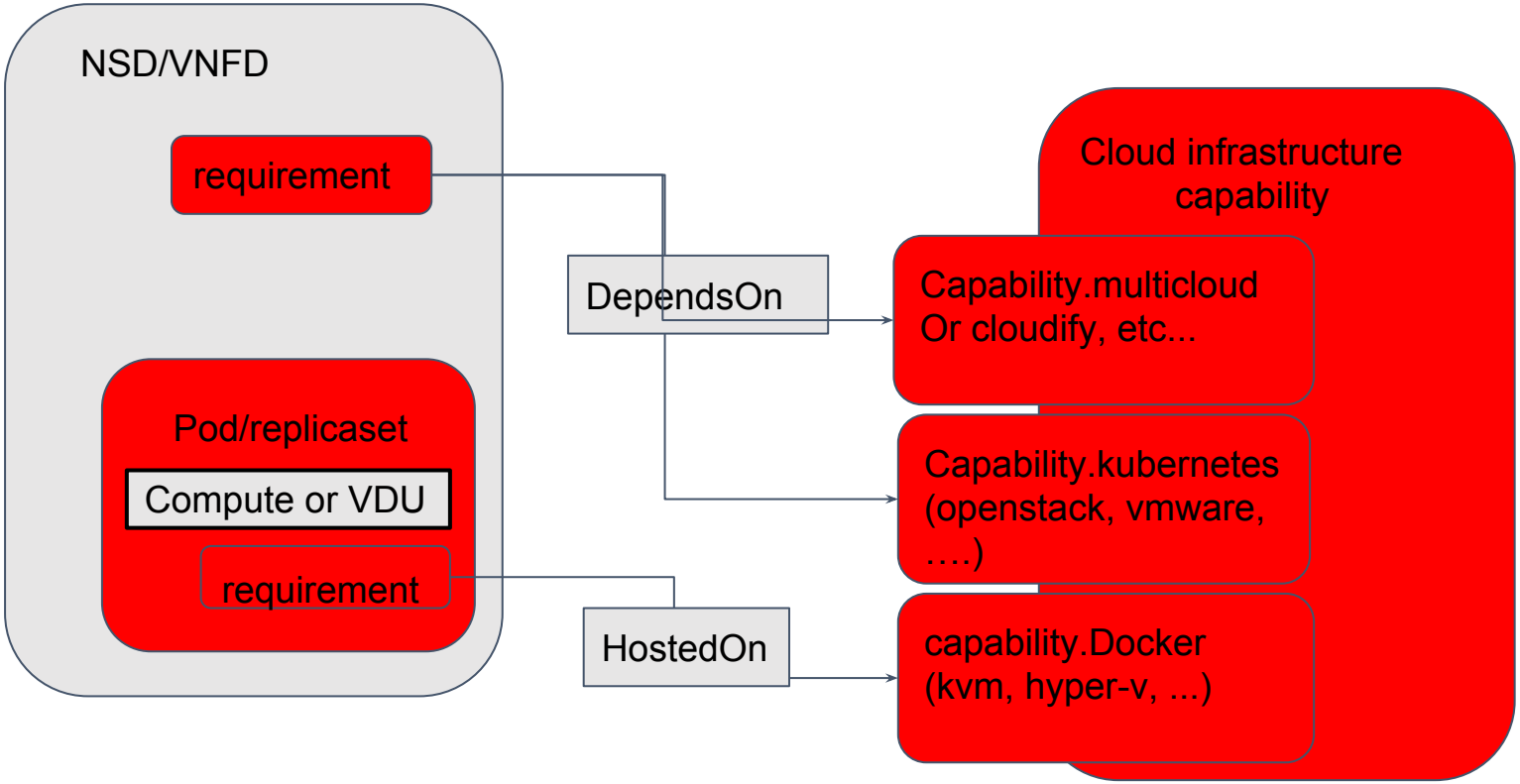
TOSCA

- Current TOSCA is defined for VMs. there are gaps/missings for container

The existing schema

- tosca.nodes.Compute and related: TOSCA simple profile
  tosca.nodes.nfv.VDU.*: TOSCA nfv profile

TOSCA enhancement for k8s



TOSCA: representing k8s cluster

- New node for k8s
  - Includes k8s features
- .compute:
  - relationsship.dependsOn
  - Feature: requirement for k8s nodes
    - -> mapping to nodeSelector

TOSCA

- LCM interface
- tosca.interfaces.node.lifecycle.Standard
  - Create
  - Configure
  - Start
  - Stop
  - Delete
- tosca.interfaces.relationship.Configure
  - pre_configure_source
  - pre_configure_target
  - post_configure_source
  - post_configure_target
  - add_target
  - add_source
  - target_changed
  - remove_target

- Life cycle hooks and local execution
  - Only PostStart, PreStop are available, and PostStart are asynchronous
    - Synchronization needs to be done by ONAP
  - Other hook needs to be triggered by SO or multicloud.
  - Or more life cycle hooks could be added to k8s

TOSCA functions and output

TOSCA functions

- get_attribute, get_operation_output

- Output:
  - Output value will be returned once request is accepted to openstack.
  - E.g. ip address of the port will be assigned when port is created.

K8s

- All the request to k8s is asynchronous. So all the request(e.g. assigned IP address) will be available later.(or error), If output is really needed, synchronization needs to be implemented with k8s watch API.
  - In cloud native way, those actual value(e.g. IP address) should be retrieved dynamically(e.g. By DNS). so that there should be no reference to output.
  - But probably output support would be needed in the transition from VM world to cloud native world

TOSCA NFV simple profile to k8s corresponding

NFV simple profile

- VDU
- VL
- External CP
- Internal CP
- VirtualNetworkInterfaceRequirements
- ...

K8S corresponding

- Pod template in service/deployment
- Network policy + k8s namespace
  - It's about network isolation. So with k8s namespace, pods can be isolated.
- Service ExternalIP or Ingress
  - LoadBalancerIP
- Service ClusterIP
  - K8s Loadbalancer needs to be enhanced for ONAP support
- Multus

VNF scheduling

TOSCA

- affinity/antiaffinity

K8s

- Kube-scheduler
- nodeSelector
- NodeAffinity,
- PodAffinity, PodAntiAffinity

VNF configuration

-

K8s

- PodPreset:
  - Injecting config value bootup time
- ConfigMap

Node status

Tosca Node state

- Initial
- Creating
- Created
- Configuring
- Configured
- Starting
- Started
- Stopping
- Deleting
- error

K8s pod status

- Pending
- Running
- Succeeded
- Failed
- unknown

K8s service

MultiCloud interface

MultiCloud for R-2 or later

- Feature reporting
- VM
  - Only image deployment with tosca.node.Compute
  - Resource limit(memory/cpu)
- Network/port
- Storage
  - Ephemeral volume
  - Persistent volume
    - Block
    - object

K8S corresponding

- Node Feature Discovery
- Bare pod, pod, service or deployment
  - With replicas=1
  - Resource quota
- Network
  - Plainly network/port can be mapped to k8s service. With single IP address/network interface
  - Service + clusterIP/ExternalIP
  - With Multus, it can be mapped to more
  - Network policy + k8s namespace
    - CNI plugin needs to support it
- Storage
  - -> StorageClass, PersistenVolume, PersistentVolumeClaim

APP-C interface

TBD

Technical gaps

More on networking

- Multiple network interface: Multus
  - IP address assignment needs to be addressed
- Tenant networking: k8s network policy + k8s namespace
- Security group: k8s network policy
  - Needs more protocol supports
- External networking: Service ExternalIP/loadbalancer or Ingress
- QoS: bandwidth allocation, dscp marking etc
  - Performance isolation
- L2 networking: TBD
- SFC: TBD
- Hybridge deployment: PNF + VM VNF + container VNF
- Network slicing
- WAN, federation

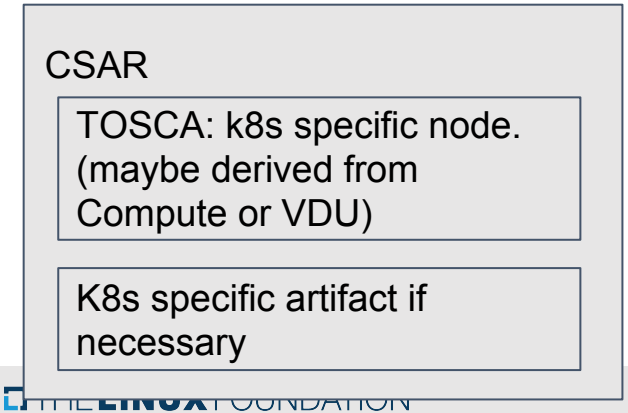
Security

-

detailed architecture/workflow

VNF design/packaging

- There is no (major) difference from VM case
- TOSCA and CASR
- It will includes extra bits for k8s
- TOSCA extension and artifact specific to k8s will be discussed TOSCA section



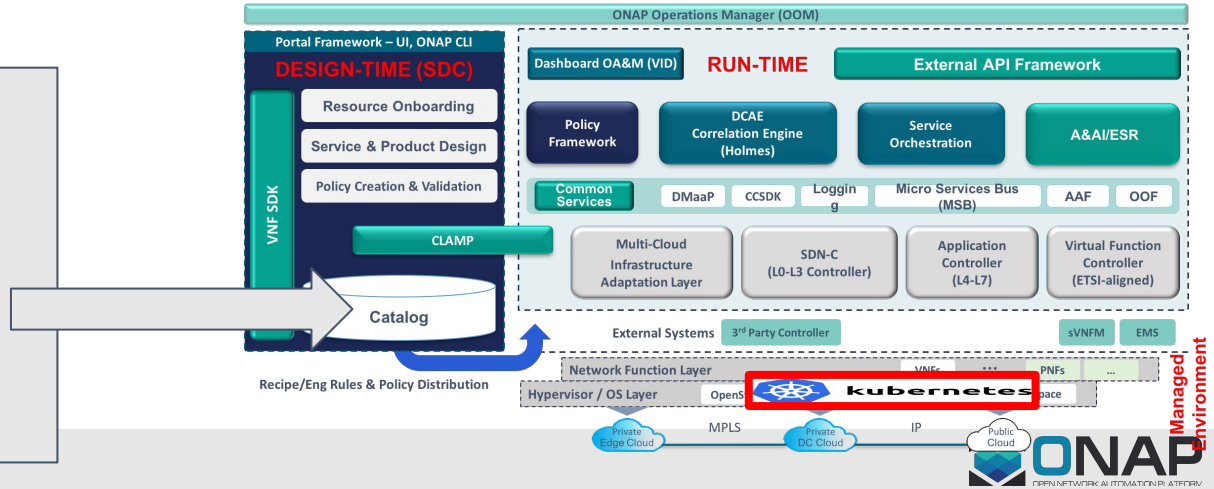
VNF on-boarding

- No (major) change as SO can store NSD/VNFD in CSAR/TOSCA format in catalog

CSAR

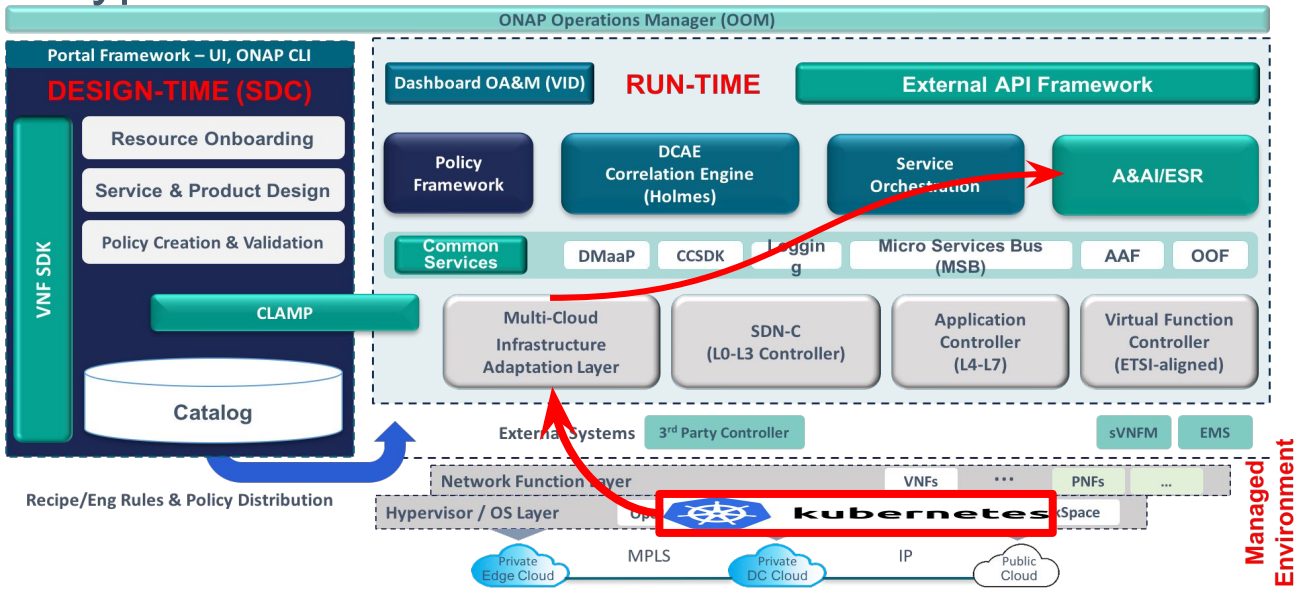
TOSCA: k8s specific node. (maybe derived from Compute or VDU)

K8s specific artifact if necessary



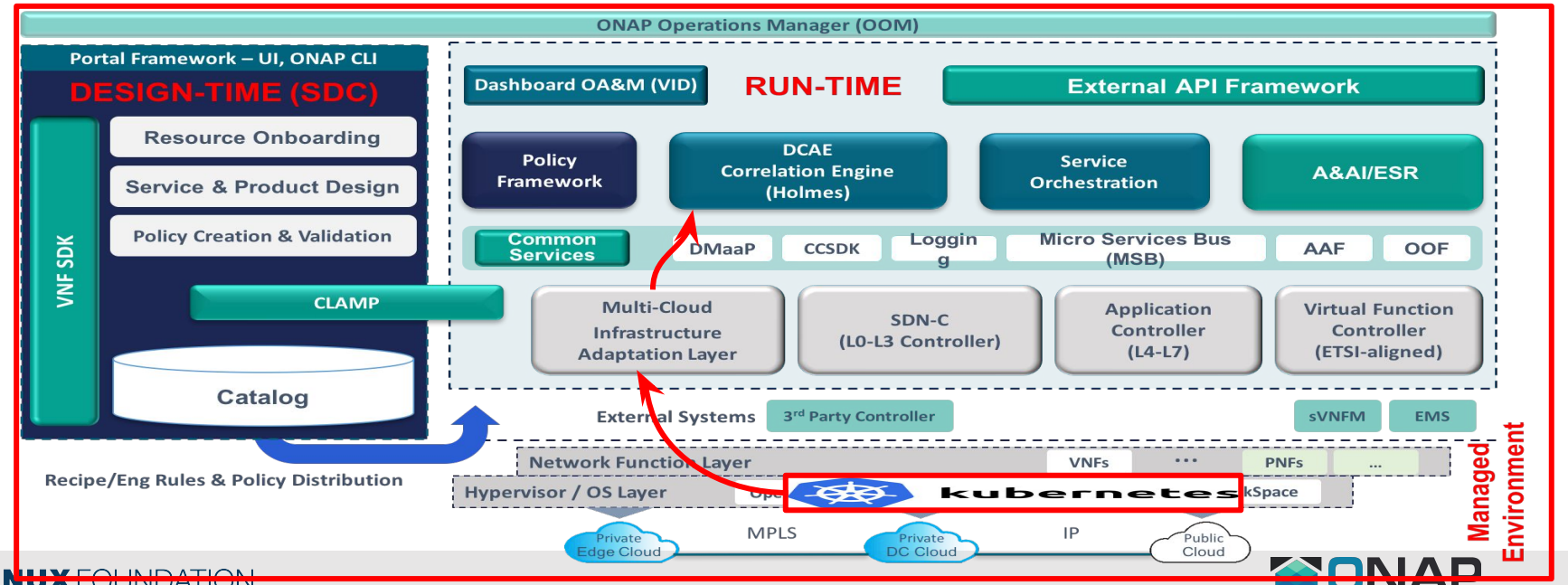
Registering k8s cluster instance

- No (major) change compared to openstack case
- Just new type/feature of cloud infrastructure for k8s



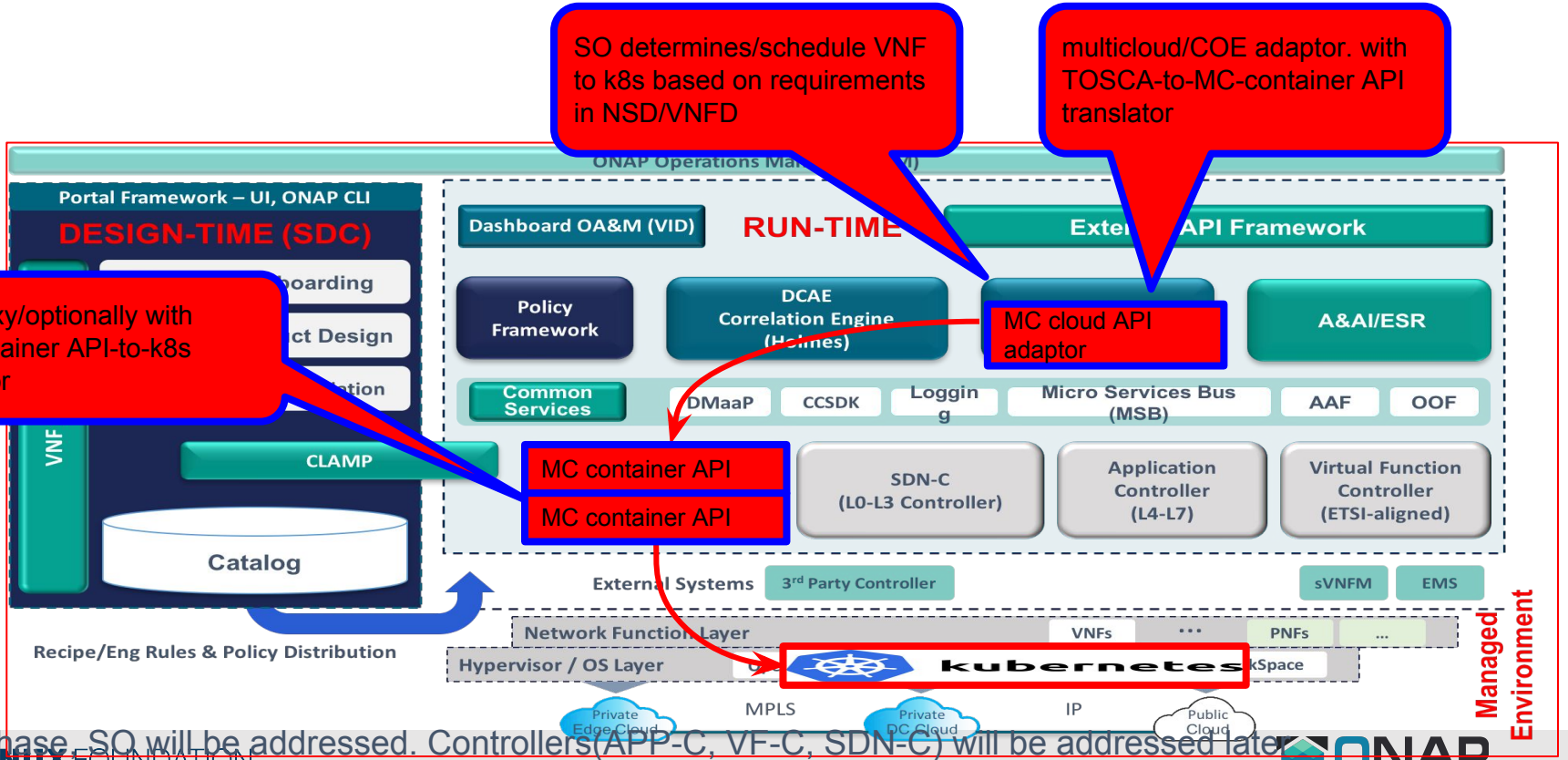
Data management of k8s stats(DCAE, FCAPS)

- No (major) change to the existing framework
- Adding new plugin to collection from k8s to report to DCAE



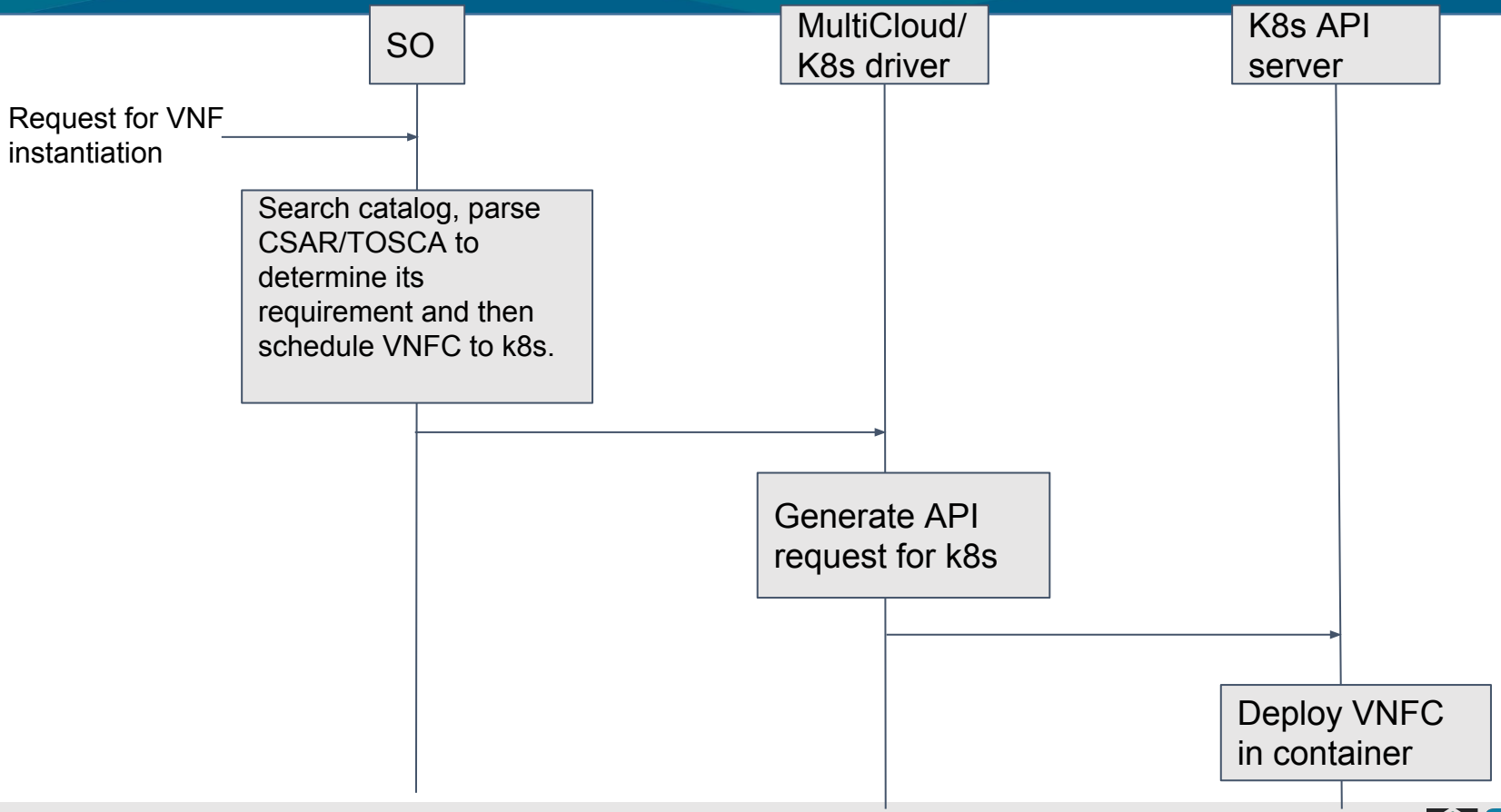
Workload management: instantiating/configuring VNF

New

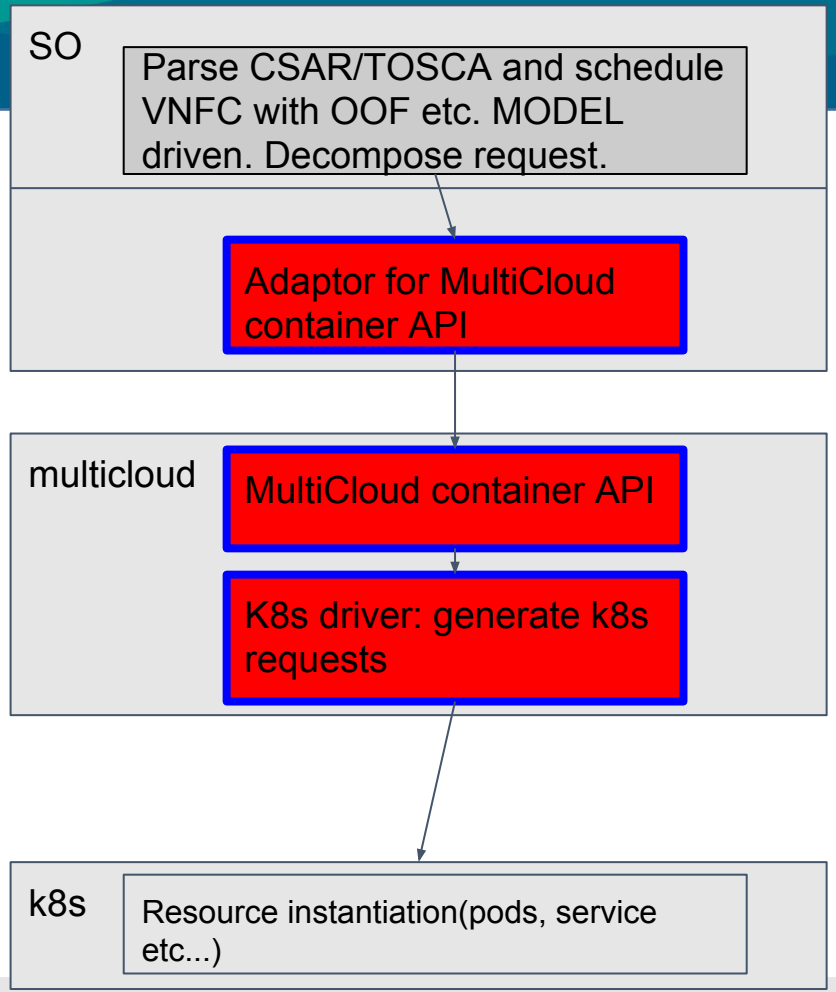


At first phase, SO will be addressed. Controllers(APP-C, VF-C, SDN-C) will be addressed later.

Operation flow



New



SO adaptor interface based on TOSCA

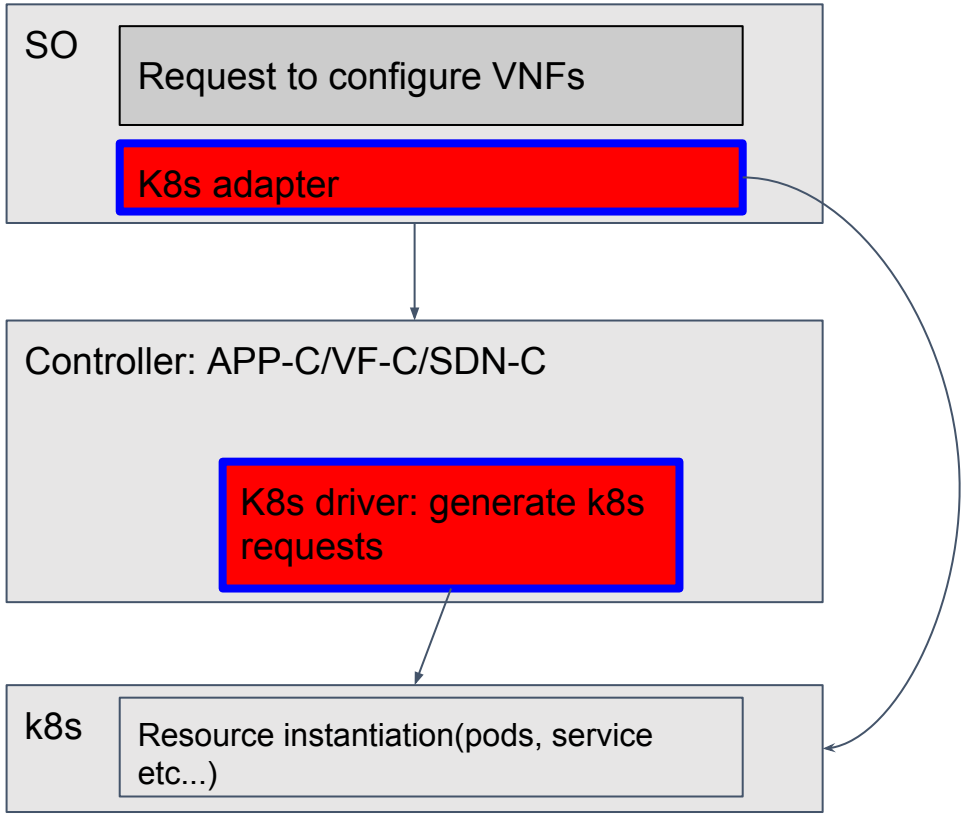
New API in multicloud for container/COE

- New API in multicloud for container/COE workload
  - Registration, data management remains same
  - SDNC api will be future task
- container/COE API is very different from VM's or openstack
- Abstracted API, not specific to any container/COE

- TOSCA based? Schema and VNF interface
  - tosca.nodes.Compute or tosca.nodes.nfv.VDU.* with container enhancement
    - E.g. tosca.nodes.Container drived_from VDU.Compute
  - tosca.interfaces.node.lifecycle.Standard
- Start with k8s passthrough in very short term to have working code

Life cycle management

New



Life cycle management

- K8s has many functionalities for life cycle managemet
- Many functions can be just deligated to k8s directly