
LF
NETWORKING
LFN Developer & Testing Forum

LF NETWORKING

LFN Developer & Testing Forum

SFC Automation with Nodus and EMCO

Eric Multanen, eric.w.multanen@intel.com

Kuralamudhan Ramakrishnan, kuralamudhan.ramakrishnan@intel.com

01/13/2022

Agenda

- Cloud Native Networking
- What is Nodus?
- SFC Automation with EMCO
- Call to Action

Cloud Native networking

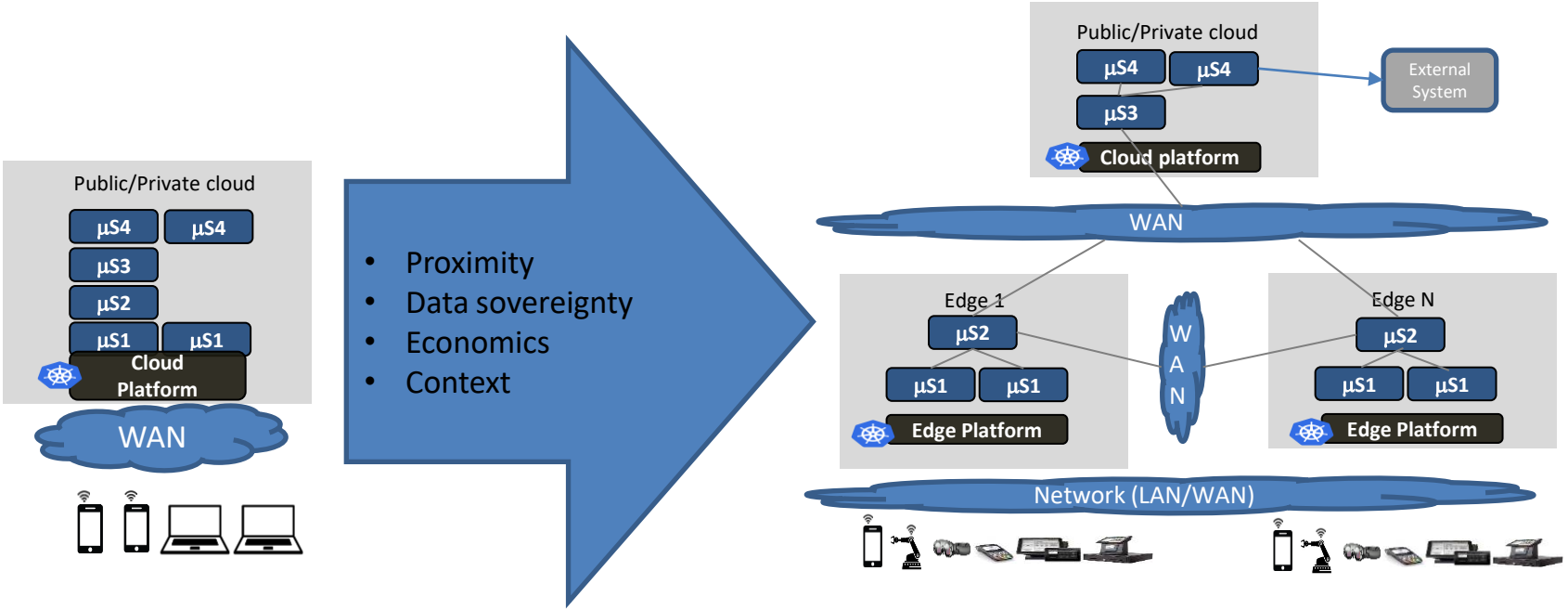
- Kubernetes Network doesn’t handle network at the first place, it call CNI plugin to handle the network.
- CNI stands for Container Networking Interfaces. This community works on the CNI spec. The standard way to communicate to a container Network namespaces
- Networking in Kubernetes remains as the out-of-scope components
- Kubernetes Networking doesn’t address Multiple network Interfaces, Multiple Network managements and identities network as entity, we must depend on the CNI plugins to do heavy lifting here.
- Things not yet discussed are more in Kubernetes – Edge Networking, Network as an entity to be configured by Kubernetes, Virtual and Provider Networks, SDN, Service Function Chaining

What is Nodus ?

- Nodus is Latin word for “a Knot”.
- As the name suggests, the Nodus is the network controller in Kubernetes
- It act like a knot that perfectly converge the NFV networking concept and uses the Kubernetes labels to implement the Service Function Chaining.
- Nodus is the answer for Software Defined Networking in Kubernetes, it takes care of Edge networking solutions, supports containers and VMs Service Function Chaining

Application and Network Transformation in Edges

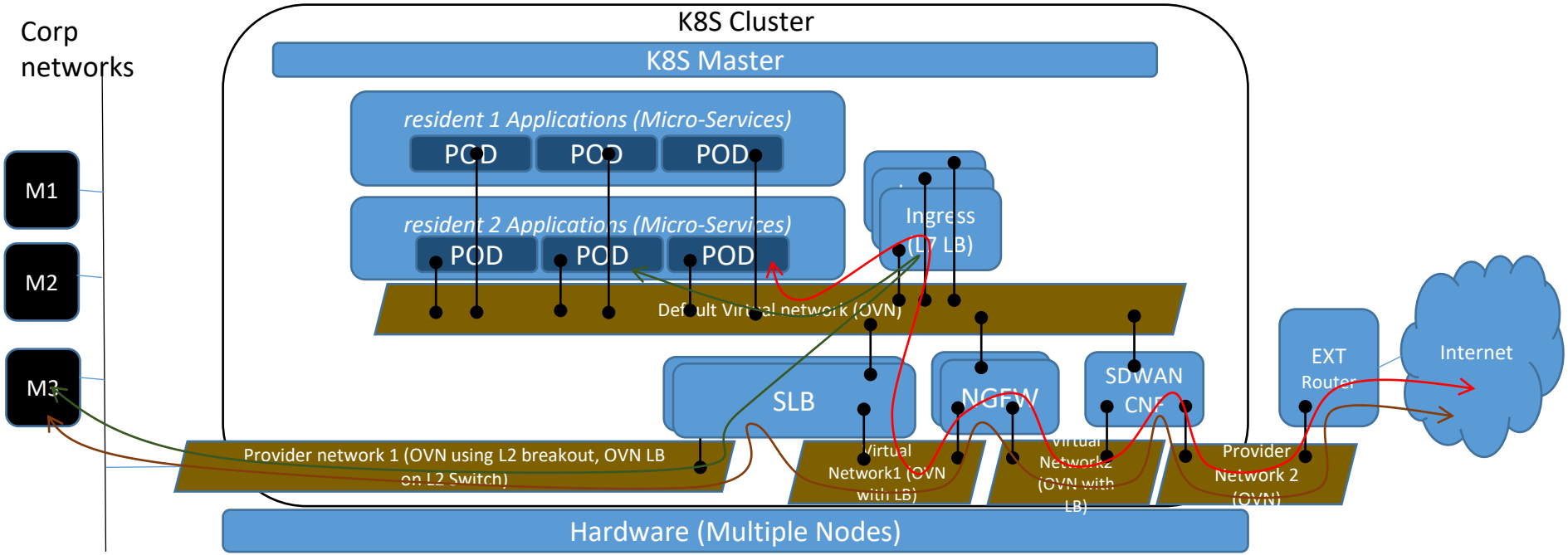
(AR/VR apps, Gaming, Analytics and Even traditional applications due to sovereignty and context)



Centralized computing to Geo distributed computing

An App consisting of four Micro-services
ms1 talks to ms2, ms2 to ms3 and ms3 to ms4
ms1" is user facing service
"ms1", "ms2" are expected to be there together
"ms2" is stateful and hence need to talk to each other

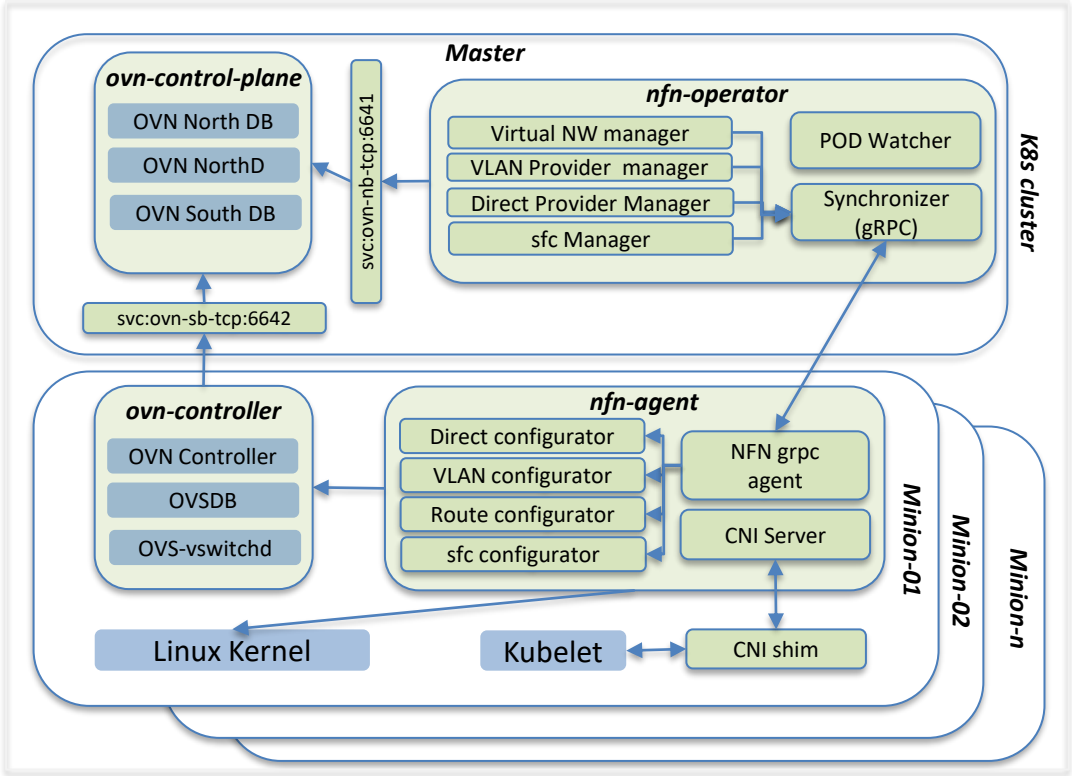
How does NFV based deployment with Cloud-native applications look like (Taking SDWAN with security NFs as an example)



Why did we choose OVN for Nodus?

- One of the best programmable controller
- Hides OVS complexity
- Broader eco-system
- L2 CNI – Support for unicast, multicast, broadcast applications
- One site level IPAM – No IP address restriction with number of nodes
- Possible to implement critical features with table-based pipeline
  (Firewall, Routing, Switching, Load balancing, Network Policy)
- SmartNIC friendly

Nodus Architecture blocks



NFN Operator:

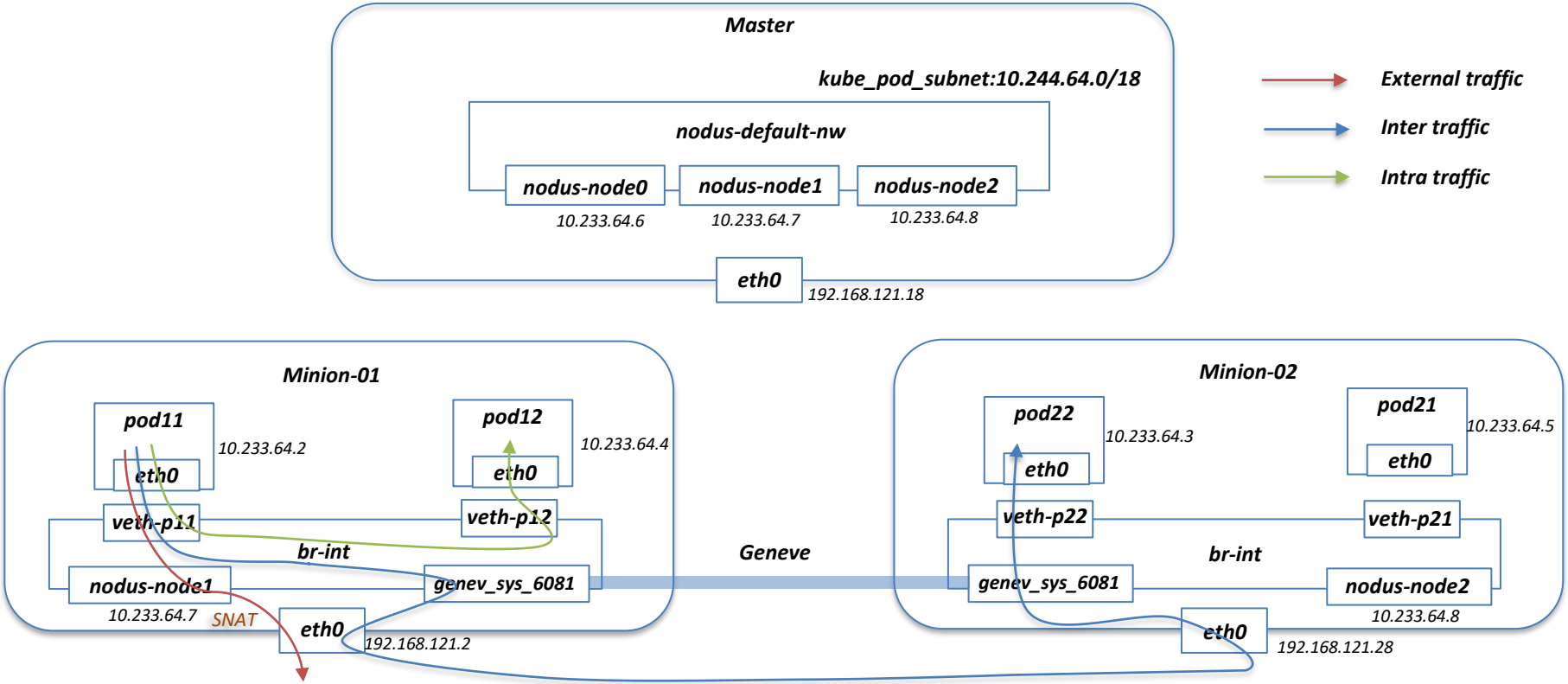
- Exposes virtual, provider, chaining CRDs to external world.
- Programs OVN to create L2 switches.
- Watches for PODs being coming up
  - Assigns IP addresses for every network of the deployment.
  - Looks for replicas and auto create routes for chaining to work.
  - Create LBs for distributing the load across CNF replicas.

NFN agent:

- Performs CNI operations.
- Configures VLAN and Routes in Linux kernel (in case of routes, it could do it in both root and network namespaces)
- Communicates with OVSDB to inform of provider interfaces. (creates ovs bridge and creates external-ids:ovn-bridge-mappings)

https://github.com/akraino-edge-stack/icn-nodus

Nodus – Networking traffic between pods

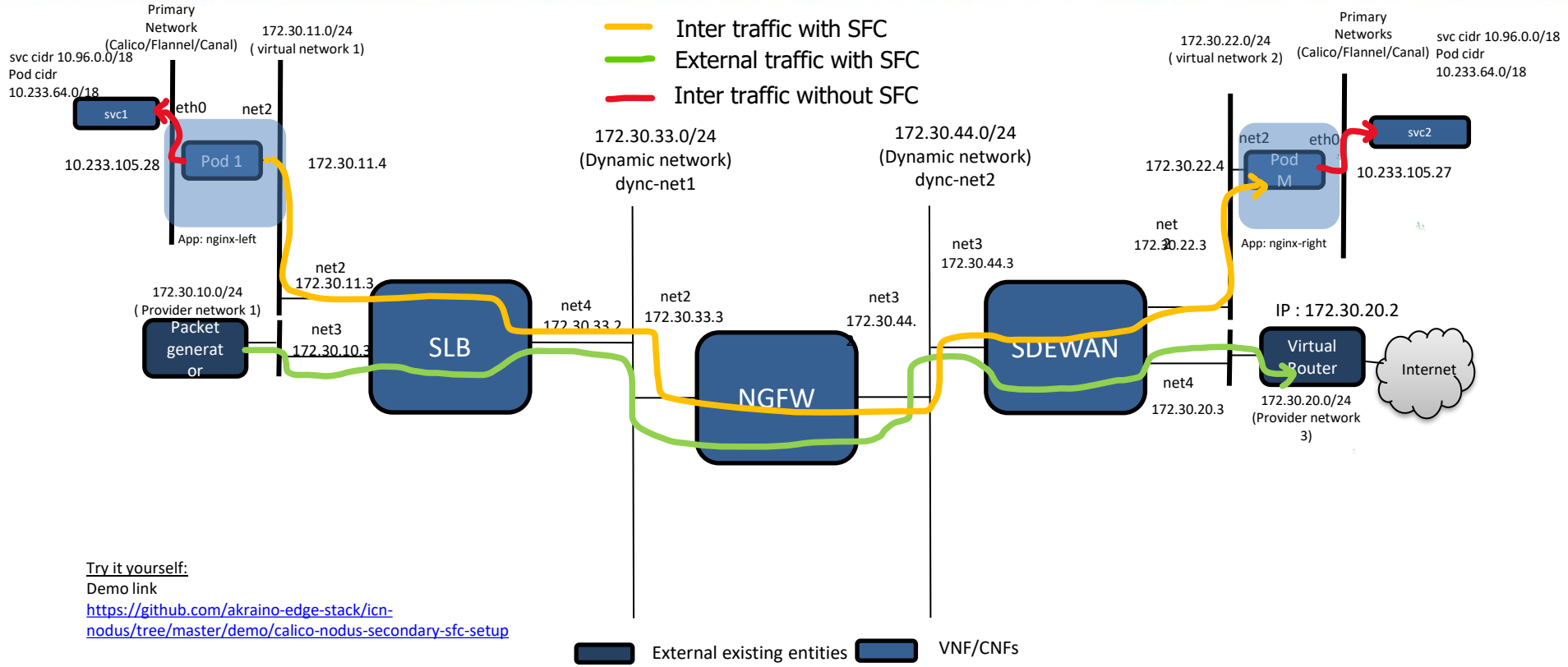


Network Chaining CR

```
apiVersion: k8s.plugin.opnfv.org/v1alpha1
kind: NetworkChaining
metadata:
  name: example-networkchaining
spec:
  # Add fields here
  chainType: "Routing"
  routingSpec:
    namespace: "default"
    networkChain: "net=virutal-net1,app=slb,net=dync-net1,app=ngfw,net=dync-net2,app=sdewan,net=virutal-net2"
    left:
    - networkName: "left-pnetwork"
      gatewayIp: "172.30.10.2"
      subnet: "172.30.10.0/24"
      podSelector:
        matchLabels:
          sfc: head
      namespaceSelector:
        matchLabels:
          sfc: head
    right:
    - networkName: "right-pnetwork"
      gatewayIp: "172.30.20.2"
      subnet: "172.30.20.0/24"
      podSelector:
        matchLabels:
          sfc: tail
      namespaceSelector:
        matchLabels:
          sfc: tail
```

Revisited standards -
Draft_Kubernetes_Software_defined_Network_Custom_Resource_Definition_De
-facto_Standard_out_of_CNI_scope

Nodus Advanced SFC - using provider networks & one Virtual networks with pod labels



Try it yourself:
Demo link
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/calico-nodus-secondary-sfc-setup

Nodus Status

- Current
  - Dynamic Network Creation
  - VLAN Provider Network Support – Controller and Agent
  - Direct Provider Network Support – Controller and Agent
  - SFC feature – Controller and Agent
  - Kubespray default primary network plugin
  - Tested with sdewan CNFs and SDEWAN Controller
  - Multiple SFC Network chaining – Working on 4 SFC models
  - Kubernetes Network policy based on OVN ACLs

Link to Repo:
https://github.com/akraino-edge-stack/icn-nodus
Demo:
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/nodus-primary-sfc-setup
https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/calico-nodus-secondary-sfc-setup

Network Innovation continues…

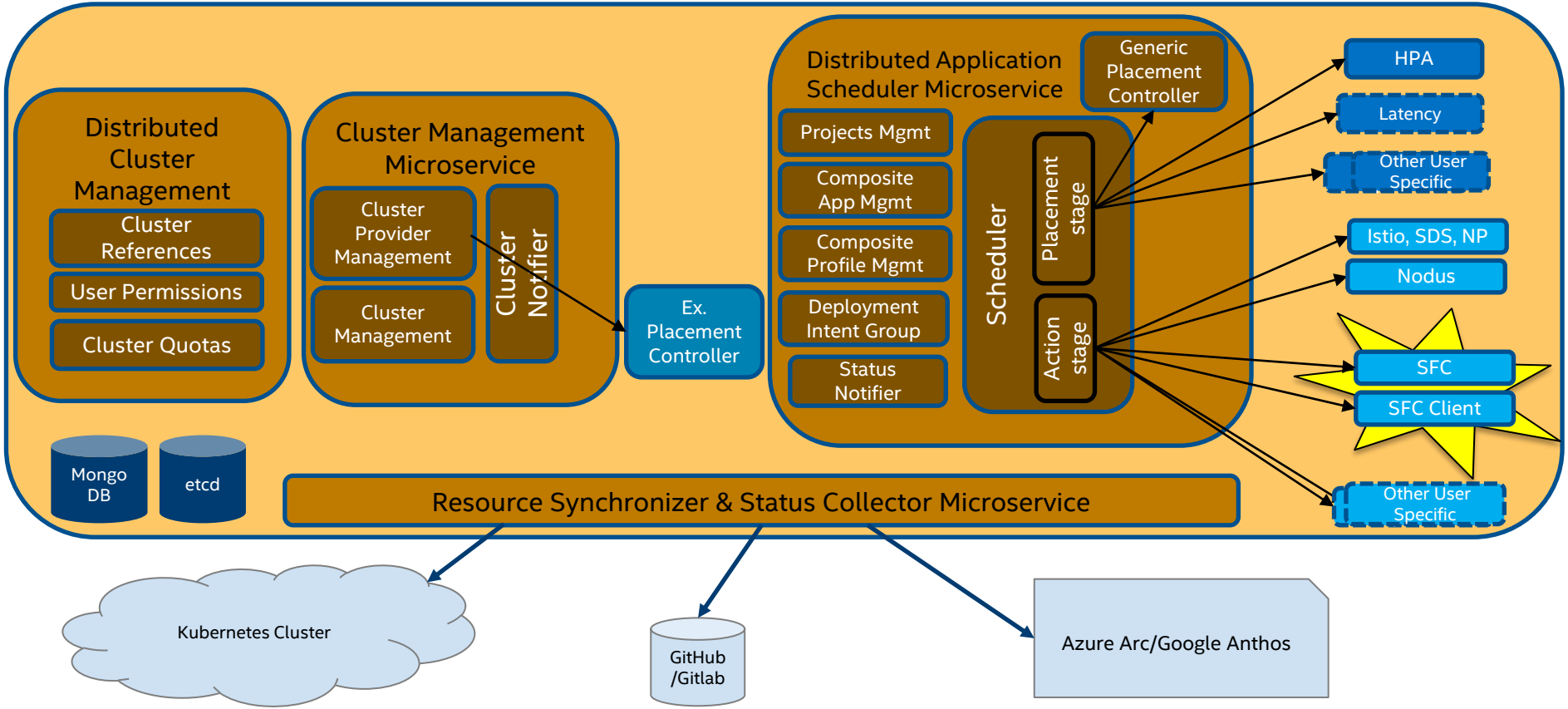
- Work In Progress
  - VM SFC in Kubernetes
  - SRIOV NIC as primary network interfaces
  - Using OVN Load balancer for Kubernetes service(without kube-proxy)
  - SFC support with OVN load balancer support for NF Elasticity
  - Proxy less service mesh with OVN & Ipsec in network namespace
  - IPv6 support
  - Traffic interception method with 5G UPF
  - Kubespray Centos CI/CD, SFC advance testing
  - Standard Software Defined Network Defacto standard in Kubernetes – Google Docs

LF NETWORKING

LFN Developer & Testing Forum

SFC Automation with EMCO

EMCO SFC Action Controllers



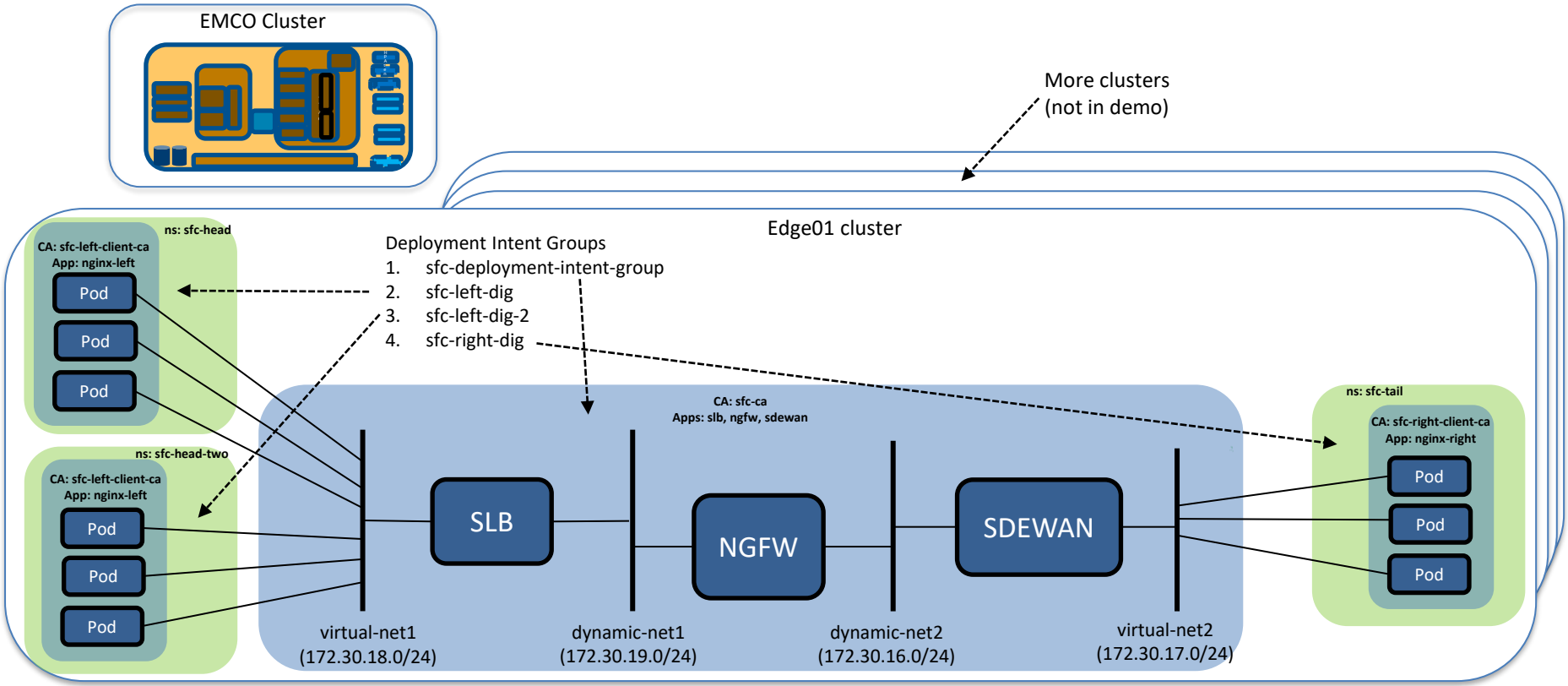
SFC Controller Intents and Action

- SFC Intent
  - Link Intents
    - Identify the SFC Functions in the Composite App (and sequence)
  - Client Selector Intents
    - Identify client Pods which may attach to either end
  - Provider Network Intents
    - Identify Provider Networks attached to either end
- Action
  - Ensure Functions (Pods) are labeled (link labels)
  - Create a Network Chaining CR and deploy to each cluster with the Functions

SFC Client Controller Intents and Action

- SFC Client Intent
  - Specify Pod resource in a Composite App to be attached to a Network Chain
- Action
  - Find the relevant Client Selector Intent (match namespace selector with Logical Cloud)
  - Ensure Pod selector labels are applied to the Pod template

EMCO SFC demo overview



EMCO SFC demo

- <show demo>

Call to Action

- Try out Nodus SFC
  - https://github.com/akraino-edge-stack/icn-nodus
  - https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/calico-nodus-secondary-sfc-setup
- Try deploying SFCs with EMCO
  - https://gitlab.com/project-emco/core/emco-base
  - https://gitlab.com/project-emco/core/emco-base/-/tree/main/examples/sfc
    Documentation and example updates are in Merge Request 51 (until it gets merged):
    https://gitlab.com/project-emco/core/emco-base/-/merge_requests/51
  - https://github.com/akraino-edge-stack/icn-nodus/tree/master/demo/calico-nodus-secondary-sfc-setup-II
- File issues, file feature requests, fix issues, contribute code, etc.