

The power of Kubernetes CRDs to automate the underlay network at the Edge

JUNE-2022

Mauricio Rojas (Mau)

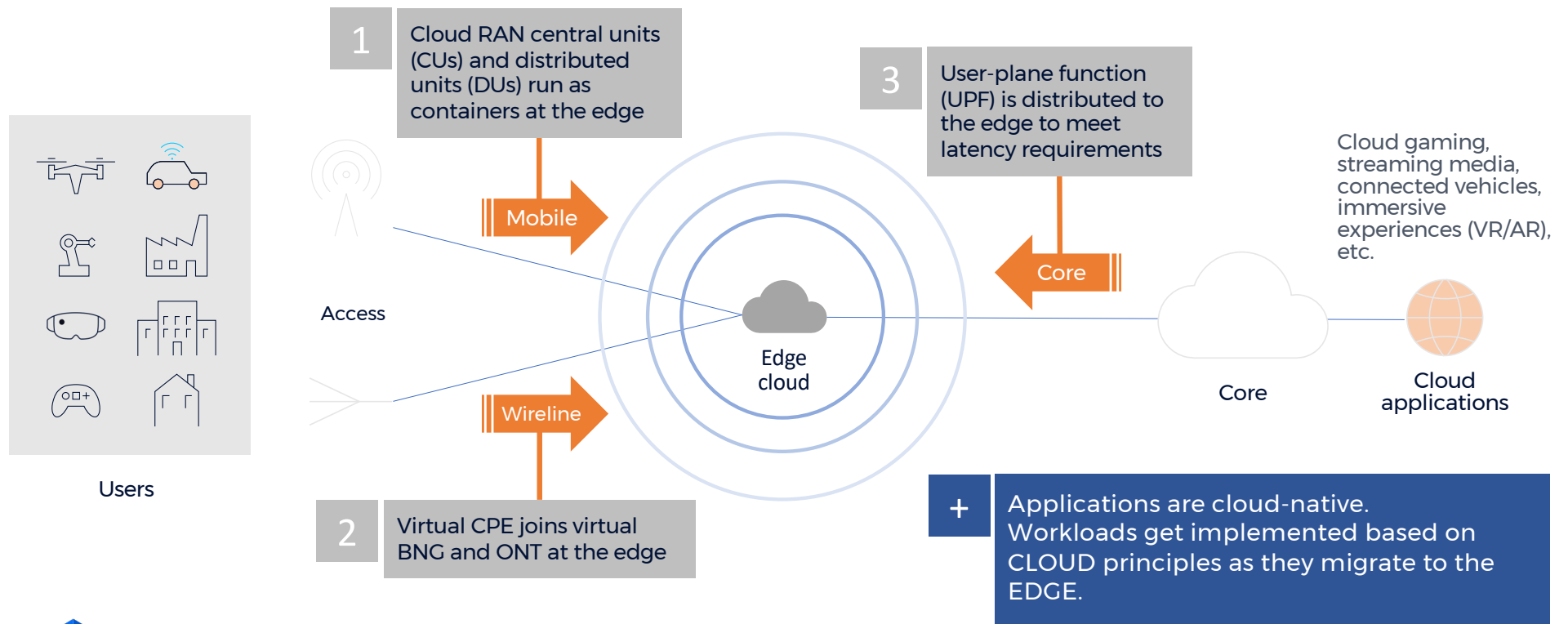
NOKIA



p1nrojas

Let's meet at the edge cloud

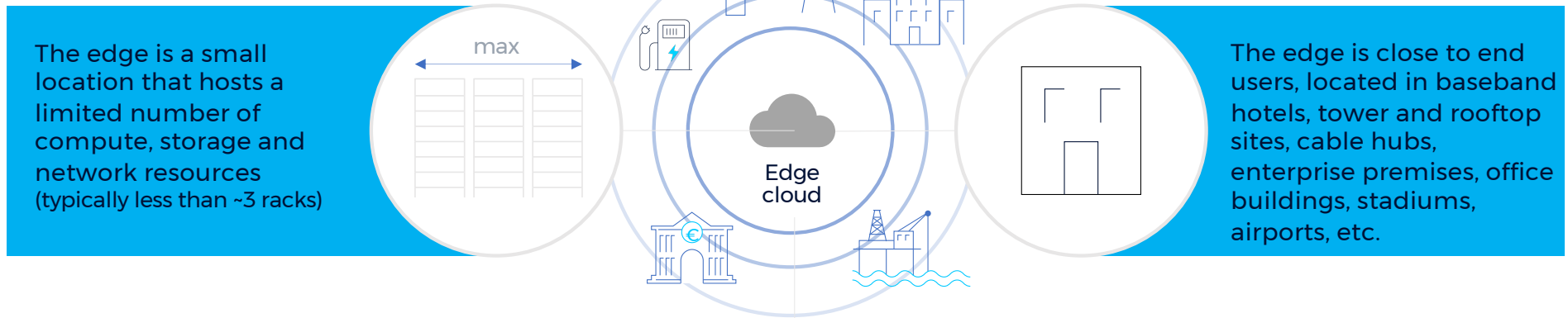
The edge cloud becomes a very critical piece of the infrastructure equation



Wait.. an edge cloud?

What is the edge cloud?

Where is your edge cloud?



How is your edge cloud called?

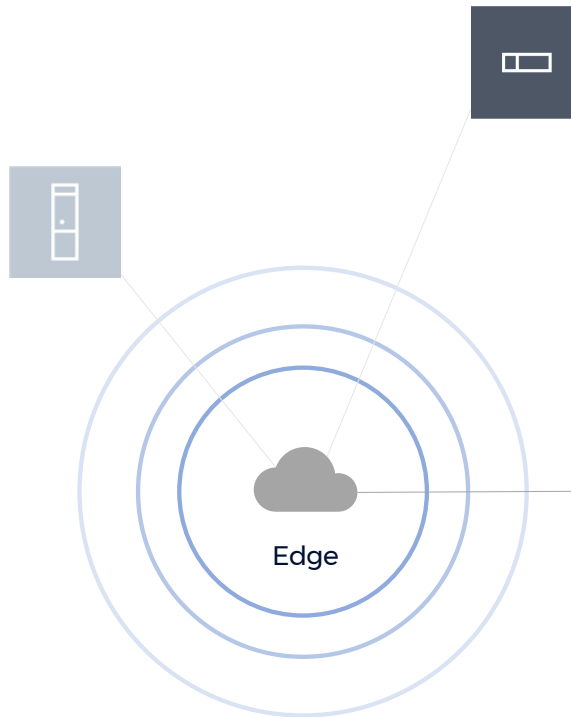
Other names used to describe the edge

- far edge
- local edge
- new edge
- mobile edge
- mini data center
- multi-access edge cloud (MEC)

What makes this edge so special

Compute and storage

- The edge is a local compute environment that builds on a cloud-native architecture (containers)
- Cloud management systems allow applications to consume workloads (compute & storage) resource on-demand
- Kubernetes is the most popular cloud management platform with 77% market share and growing



Networking

- Connect the servers hosting the workloads in the edge **and** connect to other edges and data centers

Key edge constraints & requirements

- Agility - Connections should be established automatically with compute and storage
- Efficiency - The edge is a space- and cost-constrained environment
- Self-contained - The edge should continue to run if the connection to the other data centers is lost
- Performance - Apps have stringent requirements in terms of latency and reliability

Telco CNF Apps at the Edge

Main requirements for CNF Apps at the Edge

Unless you have the Underlay Network covered. You don't have an end-to-end solution

Small Footprint. No room for Management/Automation platforms

Lack of resources to adapt orchestration tools to a separated API framework (i.e. GitOps, Prometheus)

Multitenancy and granular security and control for **multivendor** deployments

Day 2 changes to the Underlay Network, along with the CNF App dynamic

Expose underlay network natively inside **Kubernetes**

multus

Kubernetes custom resource definition (CRD)

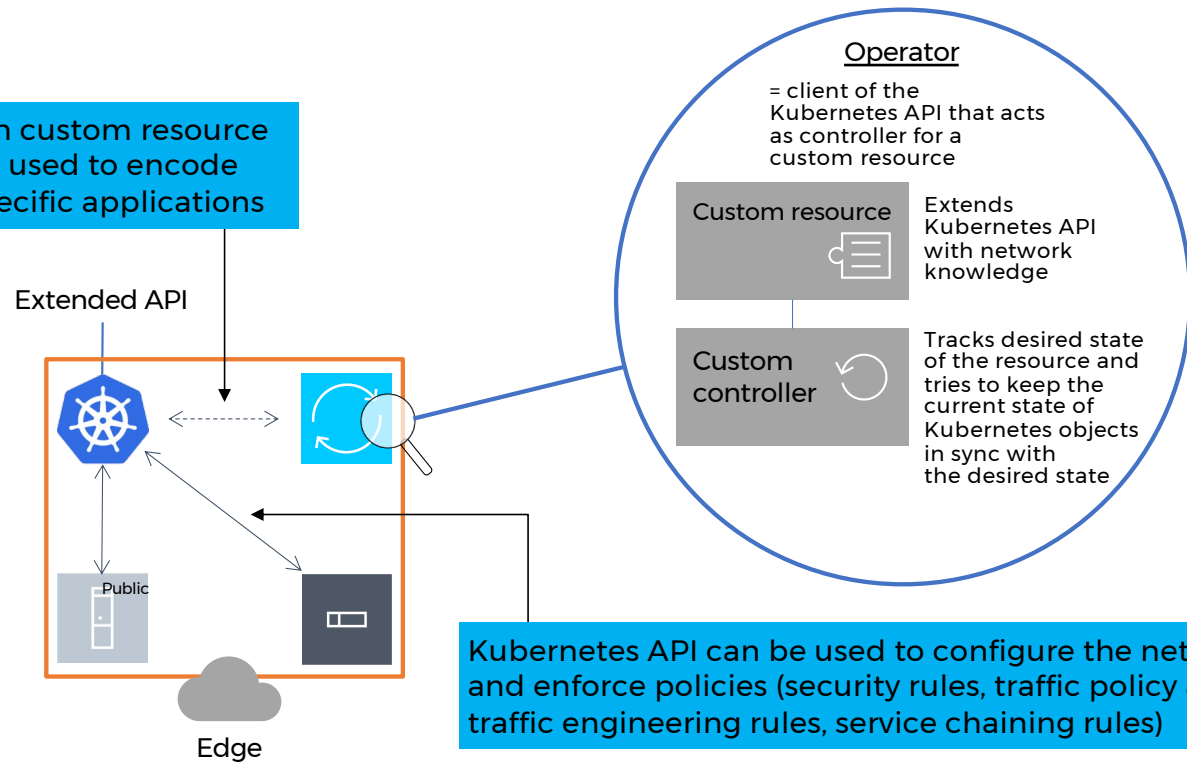
- **A powerful feature introduced in Kubernetes 1.7.**
- Introduce unique objects or types to meet their custom requirements

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: myplatforms.contoso.com
spec:
  scope: Namespaced
  versions:
    - name: v1alpha1
      version:
        storage: true
        schema:
          openAPIV3Schema:
            type: object
            properties:
```

Kubernetes potential

From container orchestration to network control

Kubernetes API can be extended with custom resource and customer controllers that can be used to encode domain knowledge (=network) for specific applications



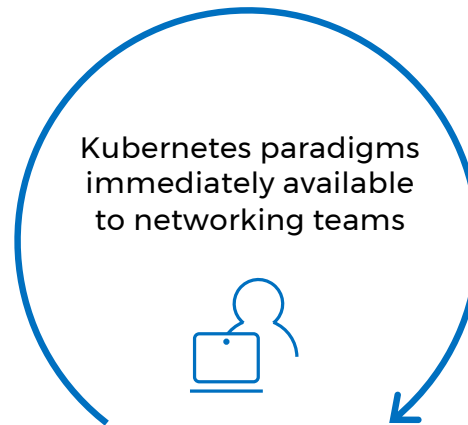
Leveraging the Kubernetes ecosystem

GitOps collaborative across involved teams



Enables organizations to continuously deliver software applications while efficiently managing IT and network infrastructure

- Declarative
- Versioned and immutable
- Pulled automatically
- Continuously reconciled



Prometheus

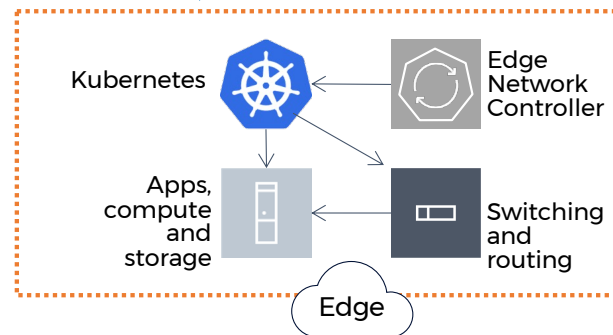


Fluentd

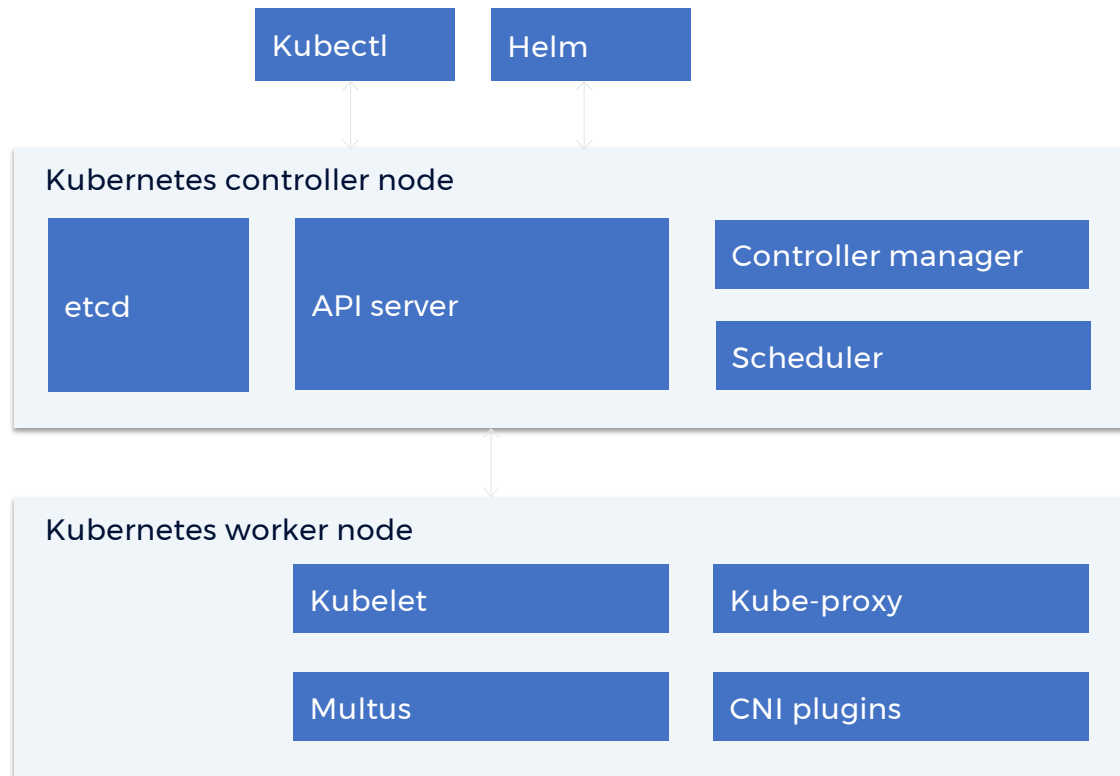


Elastic

Monitoring, logging and assurance through CNCF-based industry proven tools

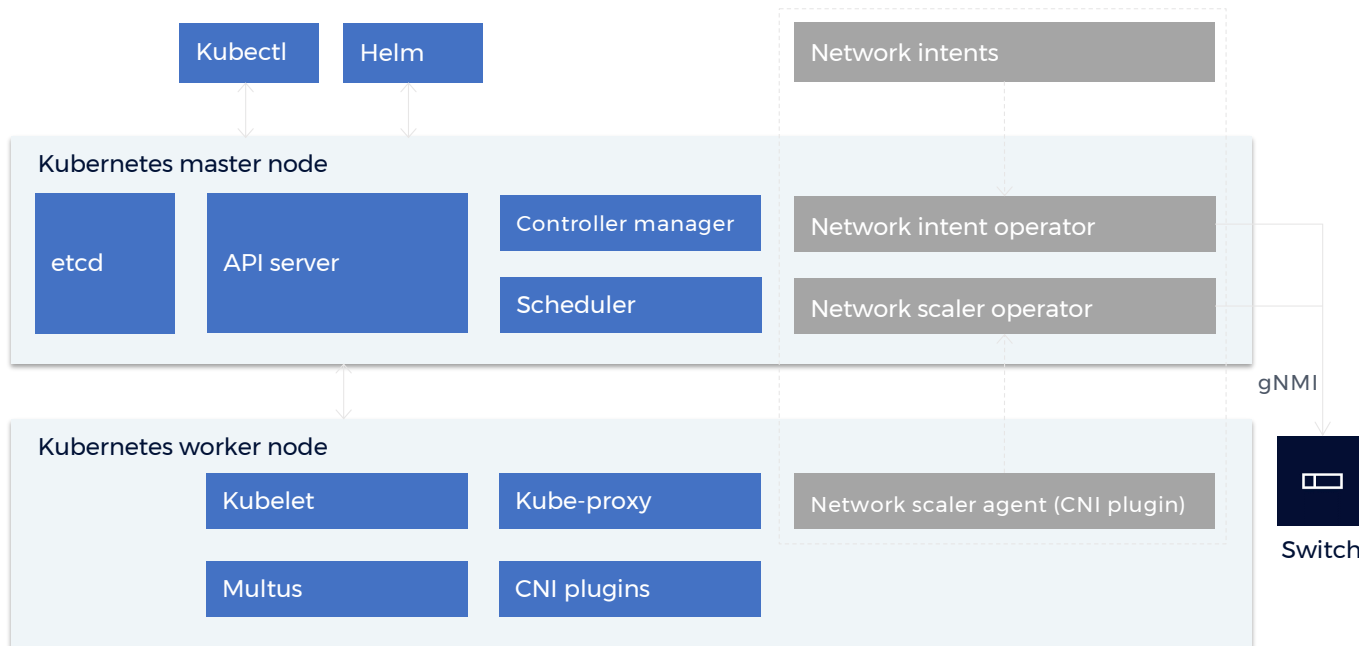


Kubernetes architecture



K8s+CRD+Controller Architecture

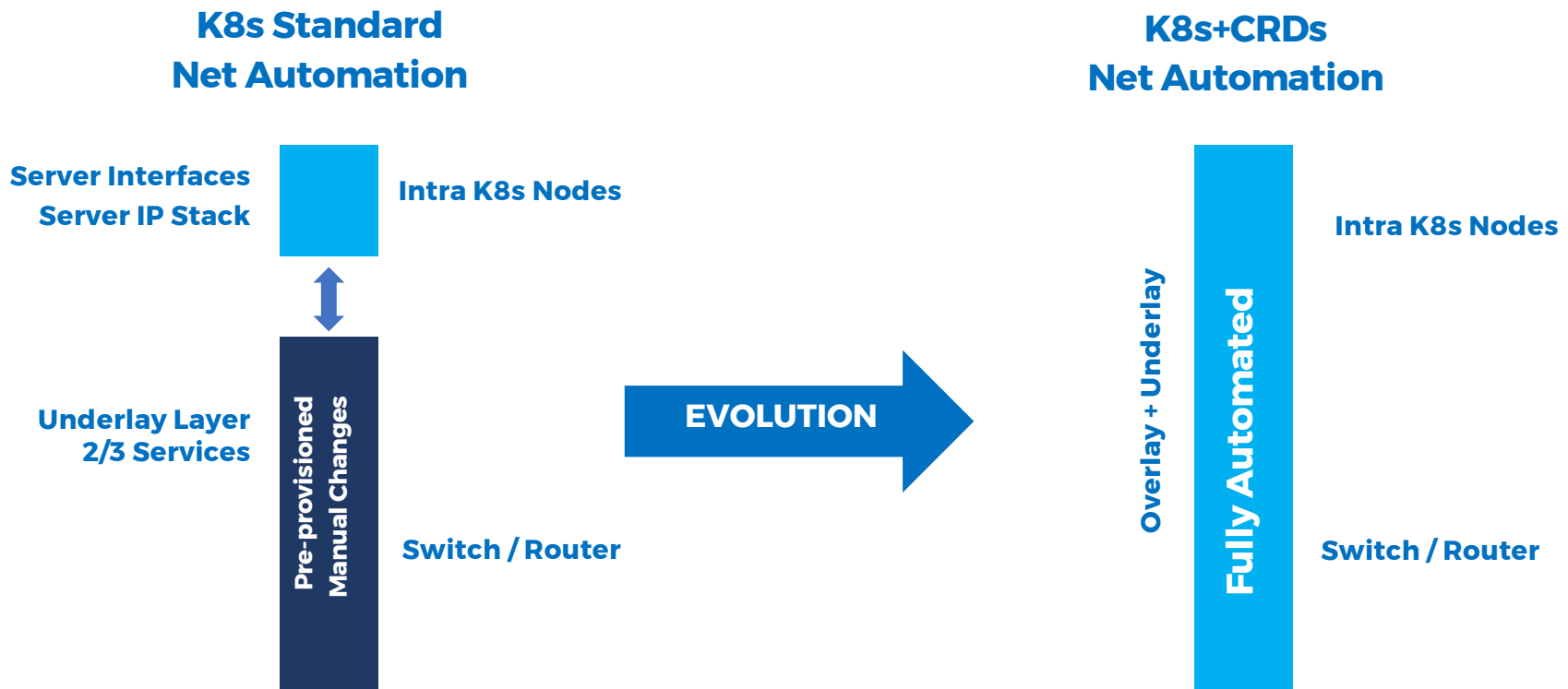
Extending Kubernetes to enable full network control



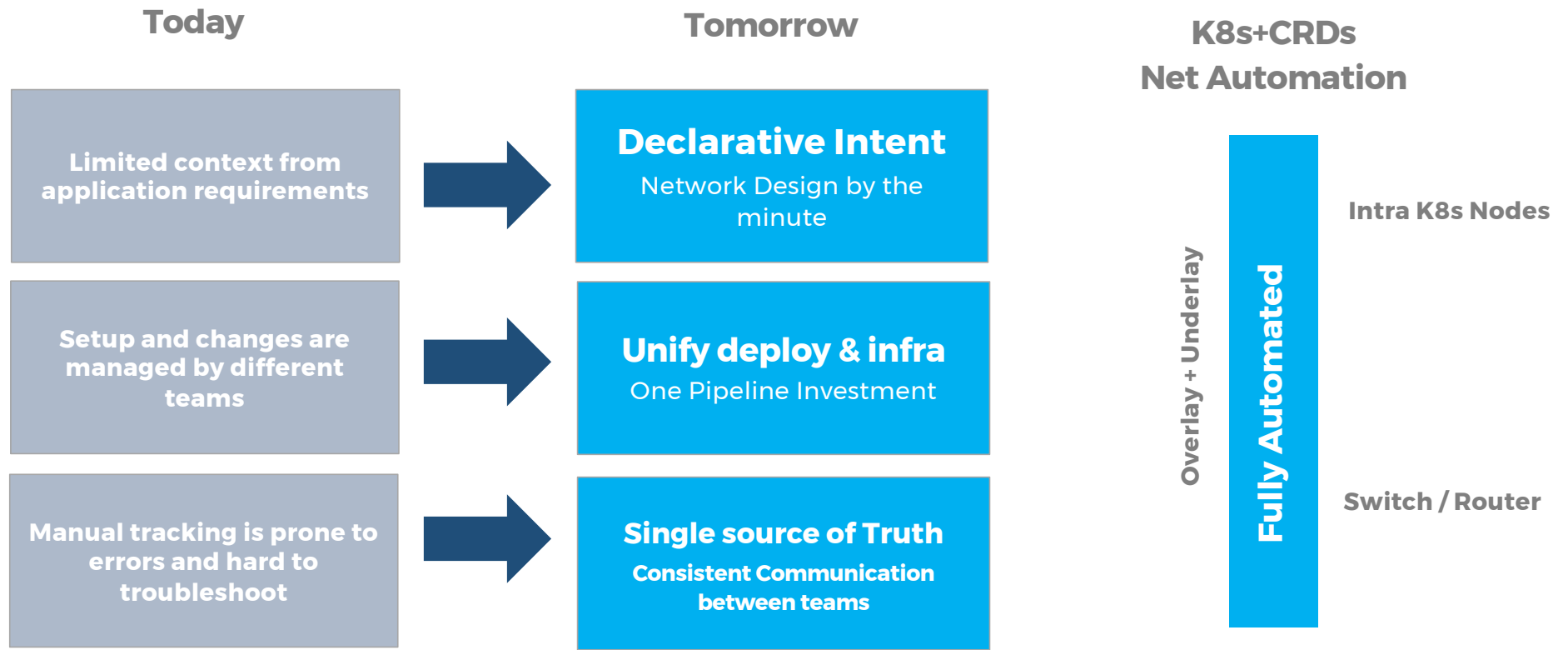
An operator is a client of the Kubernetes API that acts as controller for a custom resource

- Network intent operator allows exposure of the YANG tree of the switch and its configuration using Kubernetes API paradigm
- Network scaler (operator + agent) is a lightweight application designed to react to events and configure the switch appropriately

What is intended?

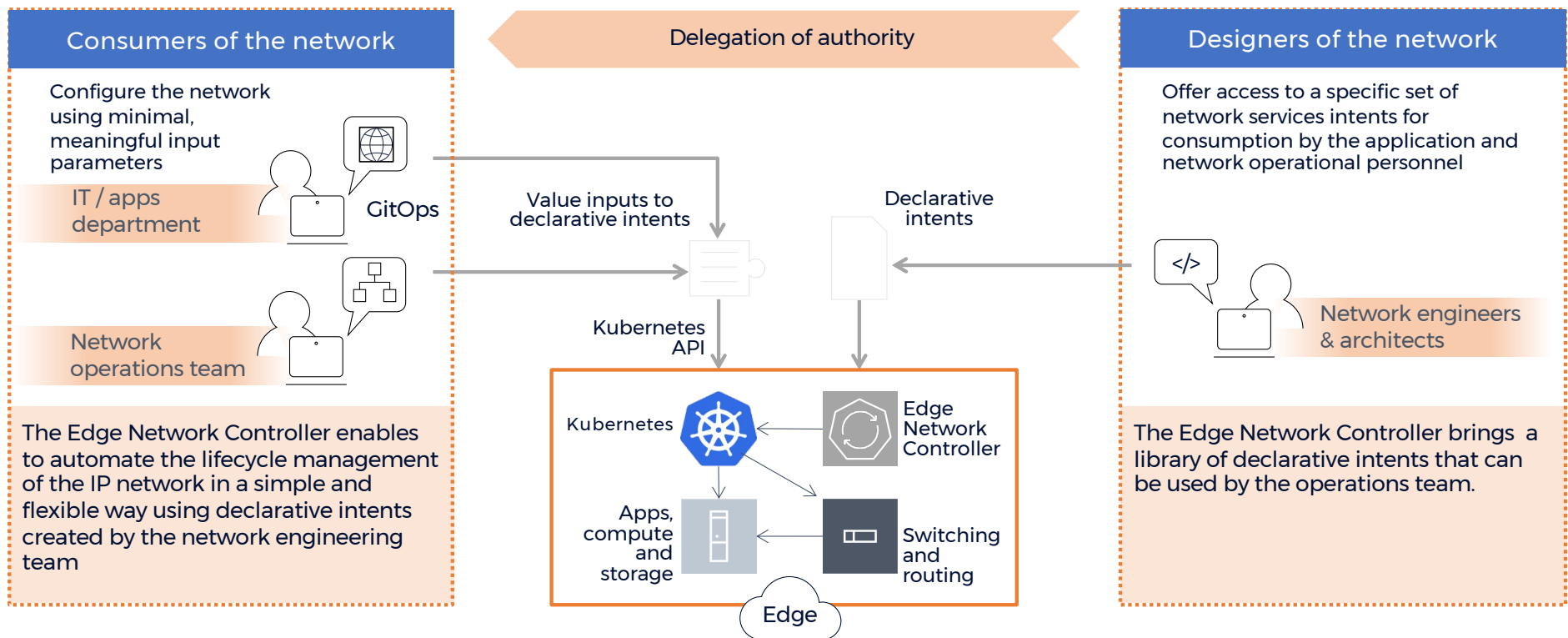


What would be the benefit?




Unify Deployment Apps/Network

Role and responsibility of involved teams




Automated device provisioning

Leverage declarative templating engine to generate k8s resources



```
device_list:
  srl-leaf:
    type: SRLinux
    interfaces:
      - port: 1/1
        sub: [251, 252, 253, 254]
        tagging: true
      - port: 1/2
        sub: []
        tagging: true
      - port: 1/3
        sub: []
        tagging: true
    ...
  srl-border:
    ...
```

values.yaml



```
- range $name, $device := .Values.device_list }}
{{- if eq $device.type "SRLinux" }}
{{- range $ignore, $itf := $device.interfaces }}
---
apiVersion: nwi.enc.nokia.com/v1alpha2
kind: SRLinuxConfig
metadata:
  name: {{ $name }}-e{{ toString $itf.port | replace "/" "-" }}
spec:
  switchID: {{ $device.ip | quote }}
  path: "/interface[name=ethernet-{{ $itf.port }}"
  properties:
    name: "ethernet-{{ $itf.port }}"
    admin-state: enable
    description: "Managed by ENC NwI Operator"
    vlan-tagging: {{ $itf.tagging }}
    mtu: 9412
{{- range $ignore, $sub := $itf.sub }}
---
[...]
```

template/srl-interface-config.yaml

Network resources are managed as native k8s resources

e.g., part of automated deployments to describe application's network SLAs

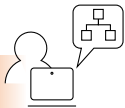
```
$ kubectl get srlinuxconfigs.nwi.enc.nokia.com | awk 'NR==1 || /ethernet/'
```

NAME	SWITCH	PATH	STATUS	AGE
srl-border-e1-1	172.30.0.8	/interface[name=ethernet-1/1]	Ready	43h
srl-border-e1-1-251	172.30.0.8	/interface[name=ethernet-1/1]/subinterface[index=251]	Ready	43h
srl-border-e1-2	172.30.0.8	/interface[name=ethernet-1/2]	Ready	43h
srl-border-e1-2-252	172.30.0.8	/interface[name=ethernet-1/2]/subinterface[index=252]	Ready	43h
srl-border-e1-3	172.30.0.8	/interface[name=ethernet-1/3]	Ready	43h
srl-border-e1-3-253	172.30.0.8	/interface[name=ethernet-1/3]/subinterface[index=253]	Ready	43h
srl-border-e1-4	172.30.0.8	/interface[name=ethernet-1/4]	Ready	43h
srl-border-e1-4-254	172.30.0.8	/interface[name=ethernet-1/4]/subinterface[index=254]	Ready	43h
srl-border-e1-5	172.30.0.8	/interface[name=ethernet-1/5]	Ready	43h
srl-border-e1-5-251	172.30.0.8	/interface[name=ethernet-1/5]/subinterface[index=251]	Ready	43h
srl-border-e1-5-252	172.30.0.8	/interface[name=ethernet-1/5]/subinterface[index=252]	Ready	43h
srl-border-e1-5-253	172.30.0.8	/interface[name=ethernet-1/5]/subinterface[index=253]	Ready	43h
srl-border-e1-5-254	172.30.0.8	/interface[name=ethernet-1/5]/subinterface[index=254]	Ready	43h
srl-leaf-e1-1	172.30.0.11	/interface[name=ethernet-1/1]	Ready	43h
srl-leaf-e1-1-251	172.30.0.11	/interface[name=ethernet-1/1]/subinterface[index=251]	Ready	43h
srl-leaf-e1-1-252	172.30.0.11	/interface[name=ethernet-1/1]/subinterface[index=252]	Ready	43h
srl-leaf-e1-1-253	172.30.0.11	/interface[name=ethernet-1/1]/subinterface[index=253]	Ready	43h
srl-leaf-e1-1-254	172.30.0.11	/interface[name=ethernet-1/1]/subinterface[index=254]	Ready	43h
srl-leaf-e1-2	172.30.0.11	/interface[name=ethernet-1/2]	Ready	43h
srl-leaf-e1-3	172.30.0.11	/interface[name=ethernet-1/3]	Ready	43h

IT / apps
department



Network
operations
team



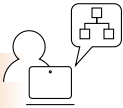
Network resources are exposed as native k8s resources

Can be consumed by applications, e.g., to track network state

IT / apps department



Network operations team



```
$ kubectl get srlinuxconfigs.nwi.enc.nokia.com srl-leaf-e1-1 -o yaml
apiVersion: nwi.enc.nokia.com/v1alpha2
kind: SRLinuxConfig
metadata:
[...]
  name: srl-leaf-e1-1
  namespace: default
spec:
  path: /interface[name=ethernet-1/1]
  properties:
    admin-state: enable
    mtu: 9412
    name: ethernet-1/1
    vlan-tagging: true
    switchID: 172.30.0.11
status:
  conditions:
  - lastTransitionTime: "2022-03-14T14:34:41Z"
    message: ""
    reason: Created
    status: "True"
    type: Ready
```


Use case groups

Not limited to

Targeted access to specific configuration based on user role



A network administrator can expose a subset of the switch configuration through the Kubernetes API for a specific user, based on their role at the edge cloud operations.

5G edge slicing



5G edge slicing allows operators to offer their enterprise customers virtual network services over both public 4G and 5G networks. Sensitive data and latency-sensitive applications can be kept on campus.

The Edge Network Controller allows to dynamically configure networks in the edge, on a per-service basis – as the 5G edge slicing solution enables for the RAN and the datacenter aspects.

MEC site network automation



Multi-access Edge Compute (MEC) site would include servers that run telco workloads for Cloud RAN and 5G Core components. The Edge Network Controller automates the edge switch that connects those servers with strict SLAs from the transport networks and makes it consumable natively leveraging Kubernetes API.

The Edge Network Controller can also automate the edge gateway to the WAN.

Enterprise home network



Secure gateways are deployed on the edge site to terminate the IPSec connections and securely forward the traffic from home workers to the corporate network. The Edge Network Controller is leveraged to program the security gateway workload along with the configuration of the edge site switch and the edge gateway.

Demo

Enterprise Home Network

01-JAN-2020



Workforces split between offices and homes are the new norm

The COVID19 pandemic only acted as accelerant

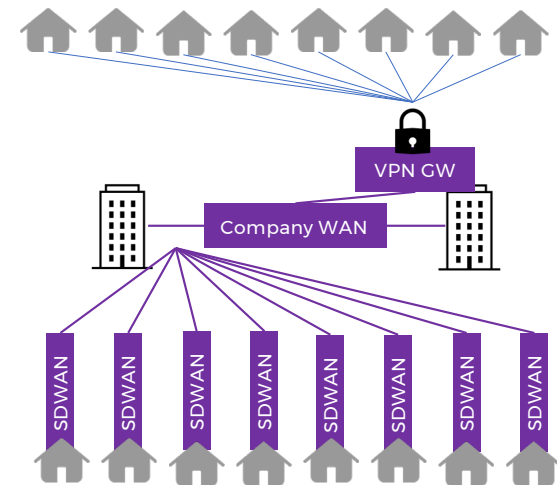
Home are also becoming “micro branch offices”

- Secure access to company resources is required
- Company traffic must coexist with private use

E2E VPNs are challenging

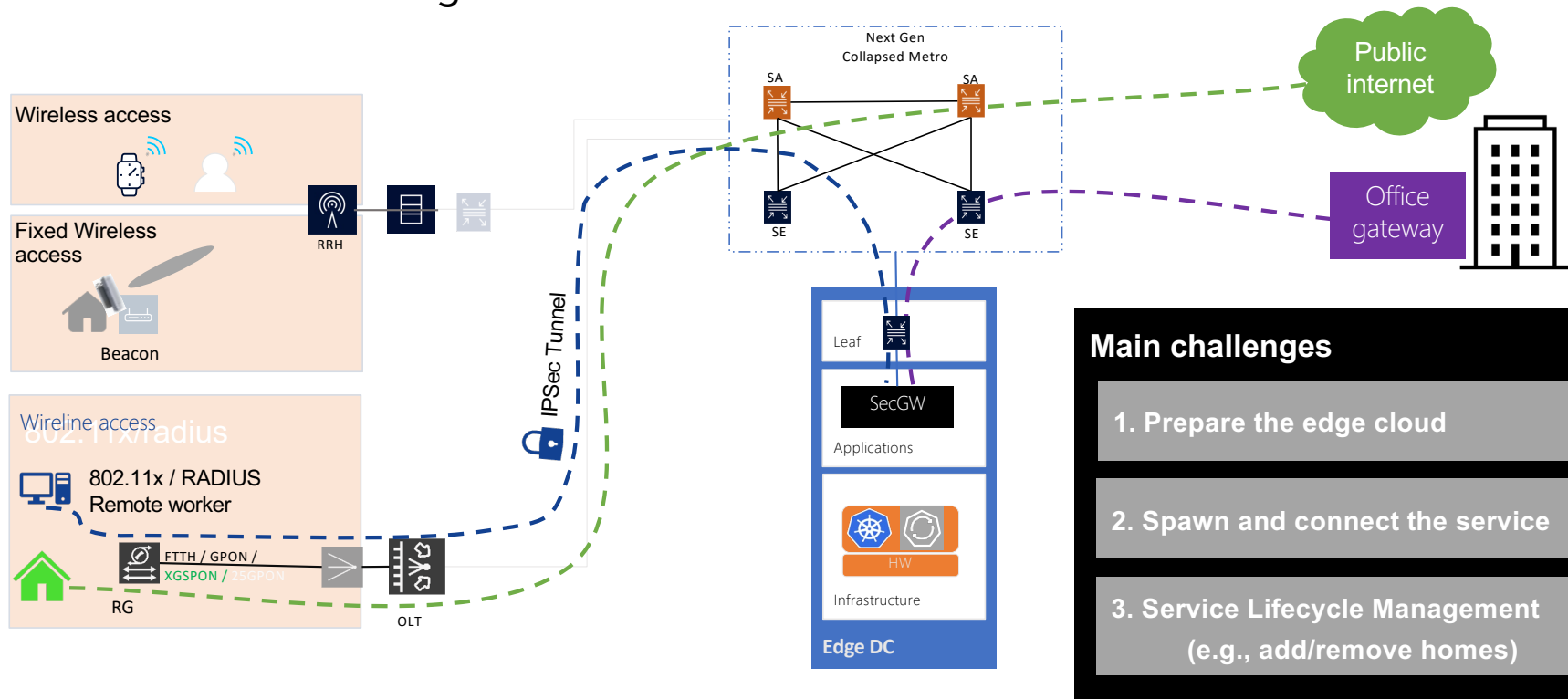
- Expensive, e.g., licenses, support costs
- Complex scalability, e.g., on-premises HW appliances
- Operational complexity, e.g., keys/certificates, active troubleshooting of network-induced issues such as CGNAT/PMTU

SD-WAN gateways in the home are not a solution either...



Architecture overview

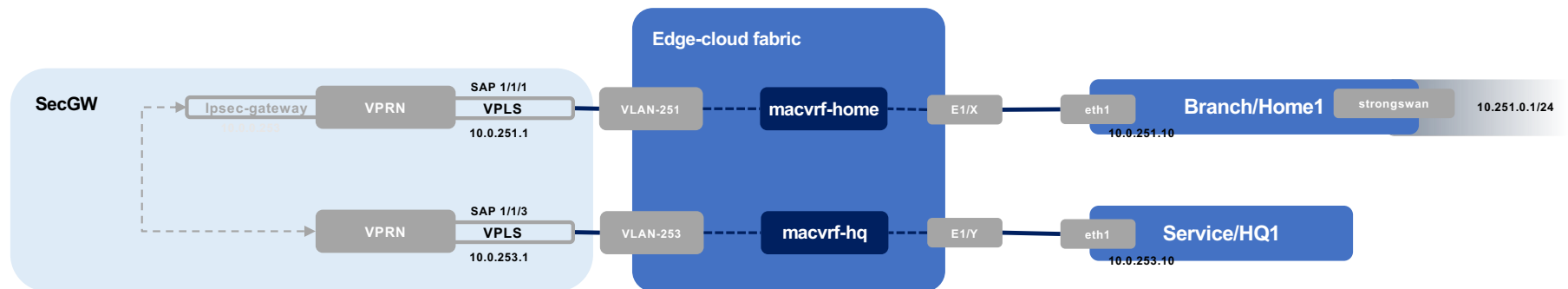
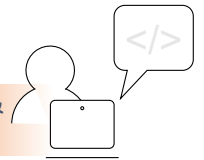
Concentrate the office-WAN interconnects in novel on-demand new edge-cloud service



PoC Network blueprint

- Service design for a single home-office pair

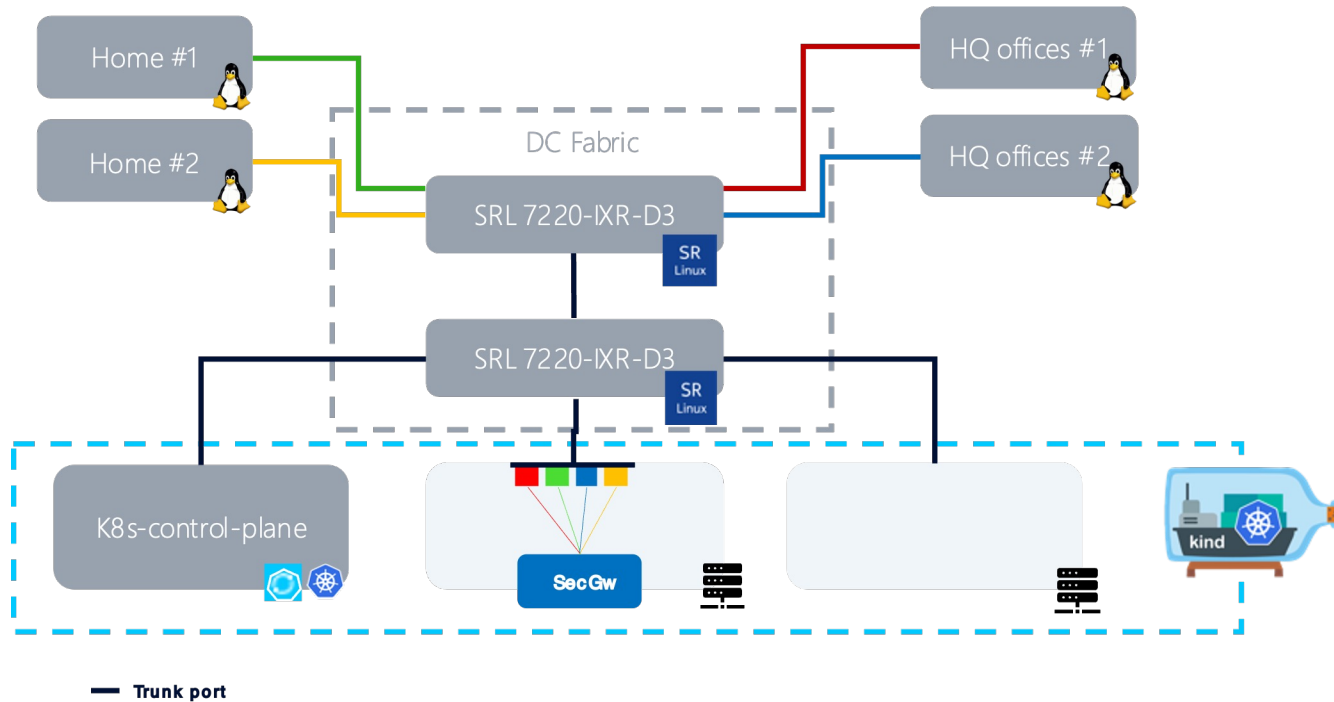
Network engineers & architects



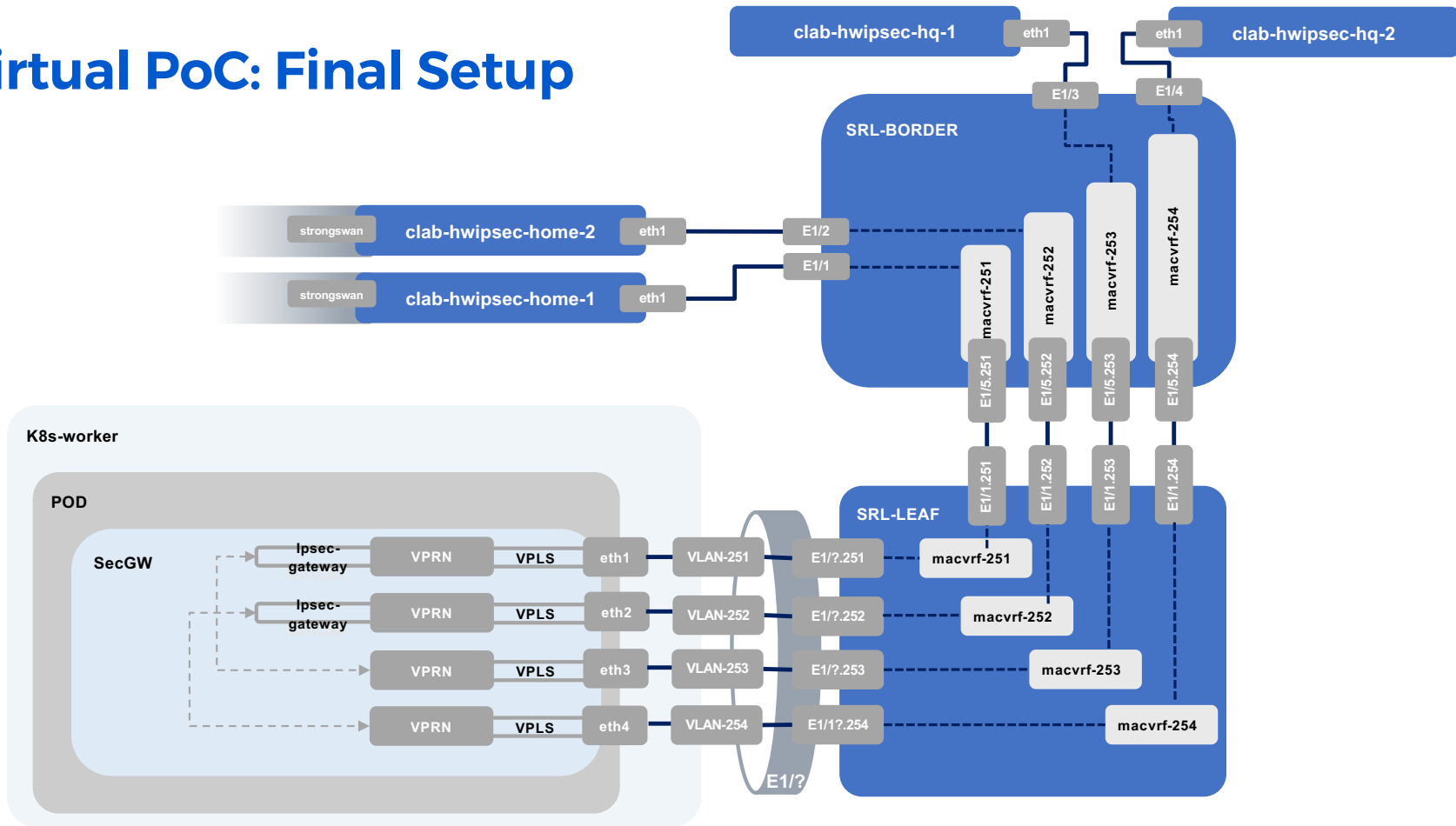
One network design, programmable on every edge, implementing the service lifecycle management operations

Virtual PoC: a representation of the network designers' service

Underlying container lab topology and logical mappings



Virtual PoC: Final Setup



Demo recording

```
deployment.apps/cert-manager-webhook created
mutatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created
validatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created
.....Waiting for k8s object creation
Waiting for pod 'cert-manager-webhook-6d66b4cdd-zwv8' to come up
> kubectl -n cert-manager wait --for=condition=Ready pod --timeout=120s cert-manager-webhook-6d66b4cdd-zwv8
pod/cert-manager-webhook-6d66b4cdd-zwv8 condition met
Deploying ENC
> kubectl apply -f /home/ubuntu/vno-secgw/enc-nwi-operator/selected/config/v1_namespace_enc-nwi-system.yaml
helm install enc-nws-operator /home/ubuntu/vno-secgw/enc-nws-operator/selected/charts/enc-nws-operator-22.6.0-alpha.1-20220313020510.tgz --set replicaCount=1 --set enc_nws_operator_images_enc_nws_controller_registrydocker.io/library --namespace enc-nws-system --create-namespace
helm install enc-nws-cni /home/ubuntu/vno-secgw/enc-nws-cni/selected/charts/enc-nws-cni-22.6.0-alpha.1-20220313083317.tgz --set enc_nws_cni_images_enc_nws_cni_registrydocker.io/library --set enc_nws_cni_images_enc_nws_cni_pullPolicy:IfNotPresent --namespace kube-system
No resources found in enc-nwi-system namespace.
> kubectl create -f /home/ubuntu/vno-secgw/enc-nwi-operator/selected/data/sros-schema-21.10.yaml
namespace/enc-nwi-system created
configmap/sros-isonschema created
> kubectl create -f /home/ubuntu/vno-secgw/enc-nwi-operator/selected/data/srlinux-schema-21.6.3.yaml
configmap/srlinux-isonschema created
> helm install enc-nwi-operator /home/ubuntu/vno-secgw/enc-nwi-operator/selected/charts/enc-nwi-operator-22.6.0-alpha.1-20220313022049.tgz --set replicaCount=1 --set enc_nwi_operator_images_enc_nwi_operator_registrydocker.io/library --namespace enc-nwi-system
NAME: enc-nws-cni
LAST DEPLOYED: Tue Mar 29 18:33:01 2022
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NAME: enc-nws-operator
LAST DEPLOYED: Tue Mar 29 18:33:05 2022
NAMESPACE: enc-nws-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NAME: enc-nwi-operator
LAST DEPLOYED: Tue Mar 29 18:33:09 2022
NAMESPACE: enc-nwi-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
> kubectl apply -f /home/ubuntu/vno-secgw/enc-nwi-operator/selected/workloadinterfaces-template-configmap.yaml
configmap/workloadinterface-templates created
customresourcedefinition.apiextensions.k8s.io/srlinuxconfigs.nwi.enc.nokia.com patched
customresourcedefinition.apiextensions.k8s.io/srosconfigs.nwi.enc.nokia.com patched
configmap/workloadinterface-templates unchanged

real    5m27.851s
user    0m54.713s
sys     0m9.111s
ubuntu@blade-d13:~/vno-secgw$
```

```
A:srl-border#
--[ running ]--[ ]--
A:srl-border# /interface ethernet-1/1
--[ running ]--[ interface ethernet-1/1 ]--
A:srl-border# info
admin-state enable
--[ running ]--[ interface ethernet-1/1 ]--
A:srl-border# show n
--[ running ]--[ interface ethernet-1/1 ]--
A:srl-border# exit
--[ running ]--[ ]--
A:srl-border# show network-instance summary
+-----+-----+-----+-----+-----+-----+
| Name | Type | Admin | Oper | Router id | Description |
+-----+-----+-----+-----+-----+-----+
| mgmt | ip-vrf | enable | up | | Management network instance |
+-----+-----+-----+-----+-----+-----+
--[ running ]--[ ]--
A:srl-border# EOF encountered
ubuntu@blade-d13:~/vno-secgw$
```

PowerPoint Slide Show - ENC-SecGW-PoC.pptx - PowerPoint

3. Lab and cluster day-0 provisioning

36 © 2022 Nokia — Trunk port Nokia Internal use

OT
Tilmans, Olivier (Nokia - BE/Answer)

References

- Edge Network Controller
 - <https://www.nokia.com/networks/products/edge-network-controller/>
 - Stay tuned, **Open APIs to be announced!**
- Containerlab: <https://containerlab.dev/>
- SRLinux: <https://www.nokia.com/networks/data-center/service-router-linux-NOS/>



p1nrojas

Thank you

June 2022



NOKIA