Network Automation using KRM at Swisscom

CNTF, 25.11.2024

Ashan Senevirathne, Pablo Garcia, Fabian Schulz

swisscom

Ashan Senevirathne
Product Owner

ashan.senevirathne@swisscom.com

Fabian Schulz
DevOps Engineer

fabian.schulz1@swisscom.com

Pablo Garcia
DevOps Engineer

pablo.garciamiranda@swisscom.com

Swisscom's Telco to TechCo transformation journey

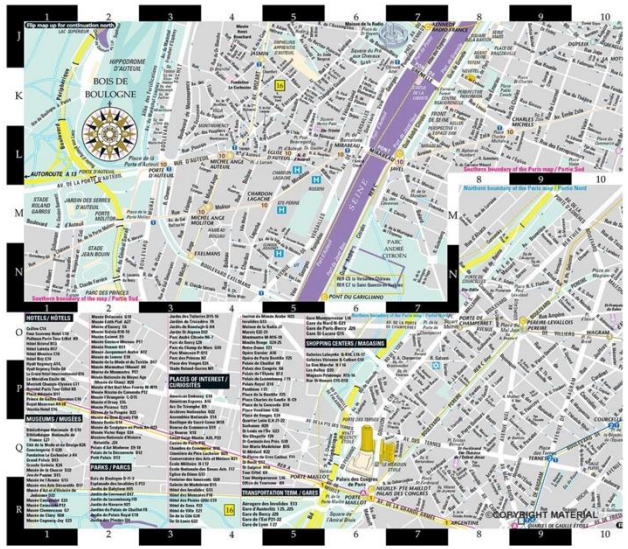
Telco	TechCo	
PNF & VNF Complex NFVI operations	True Cloud nativeness Kubernetes on Baremetal	**The catalyst** of our **industry transformation**
Resource Monitoring Classical Ops	Service observability SRE & Autonomous Networks	Focus on **Services instead Boxes & Failover instead of 99.999%**
Manual scripted changes 1 release per year Changes take weeks Traffic-free / downtime for upgrades	Intent-based orchestration CI/CD/CT Changes take hours In-service upgrades	**Declarative** and **Intent-based** automation on **GitOps** principles

An Analogy…

Static paper map

- Fixed
- Static
- Unchanging
- Overwhelming

This is GitOps today

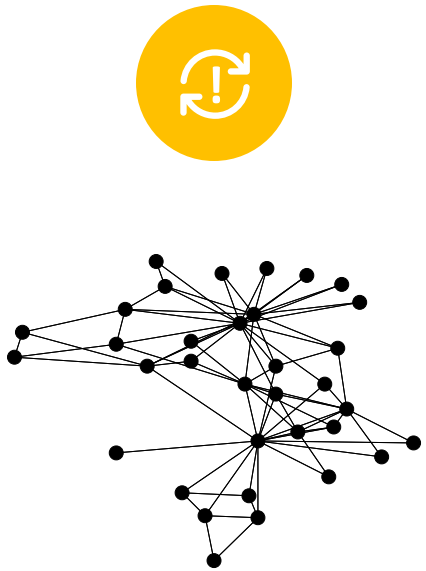


Apple maps

- Dynamic
- Changes based on external conditions
- More focused
- Simple to navigate

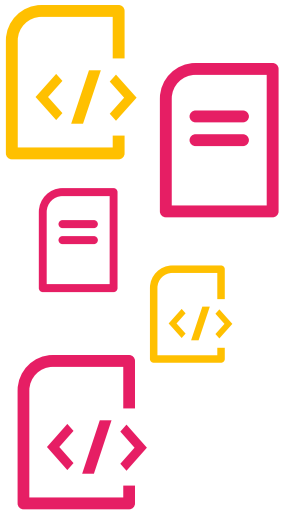
This is GitOps w/ KRM

Key Challenges



Network Automation Complexity

Dual Mode Core adds both 4G and 5G NFs; complexity grows



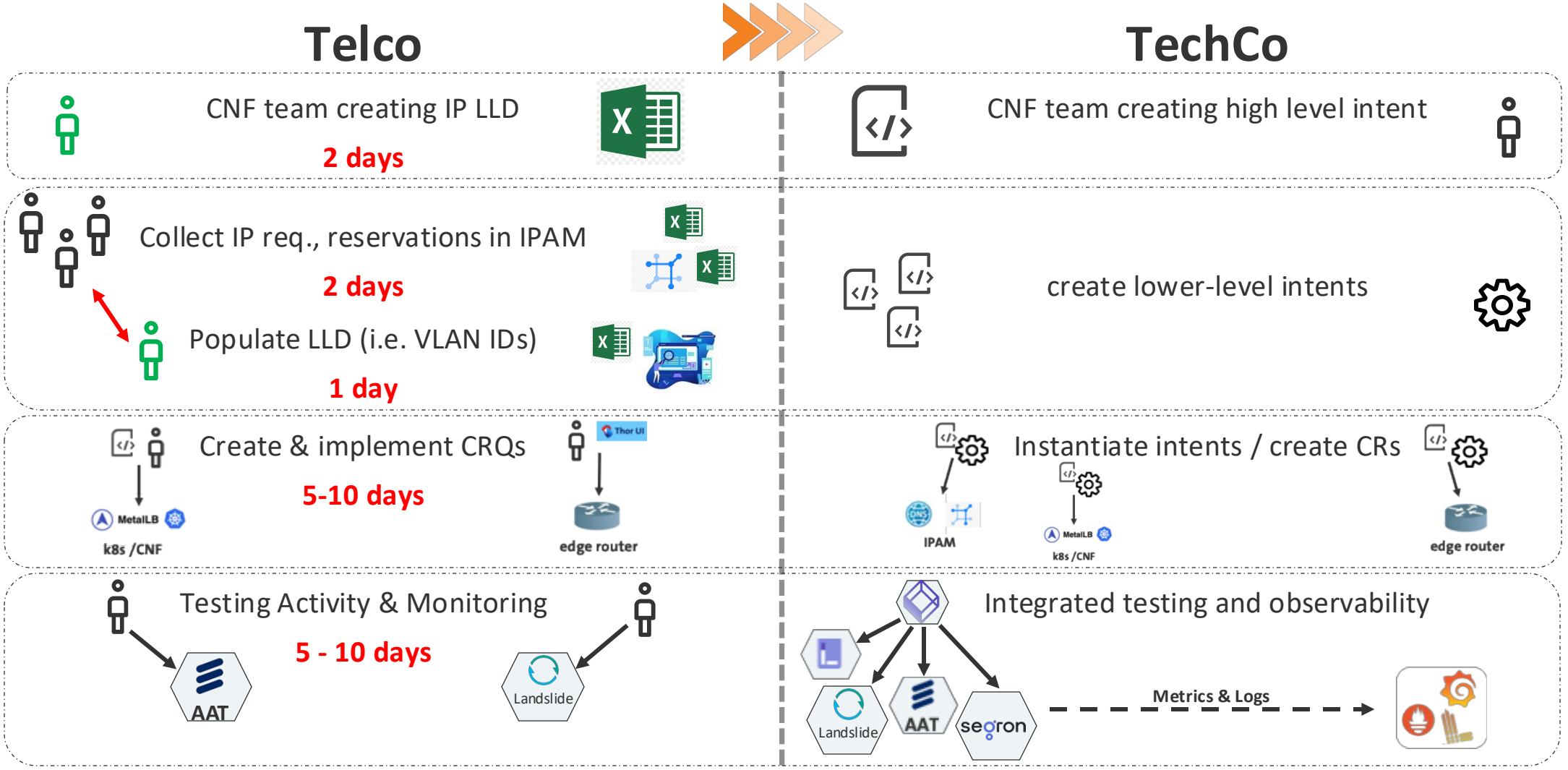
Limited Scalability

Complex NF configurations; many integrations; limits scaling

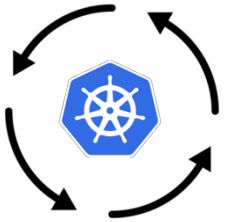
System Lock-in

Proprietary tools block multi-vendor options

Manual vs. Intent Driven Connectivity Provisioning



Key Orchestration Principles



Kubernetes

Kubernetes as a common automation framework

Intent based

Target state description in Kubernetes Resource Model (KRM)

Simplicity

Reduce complexity across systems and processes

GitOps

Anchored in a GitOps management process

Re-usable

Re-use existing solutions within Swisscom

Cloud-Native

Leverage cloud-native technologies and the cloud native ecosystem

What Is a 5G Core?

Each blue object

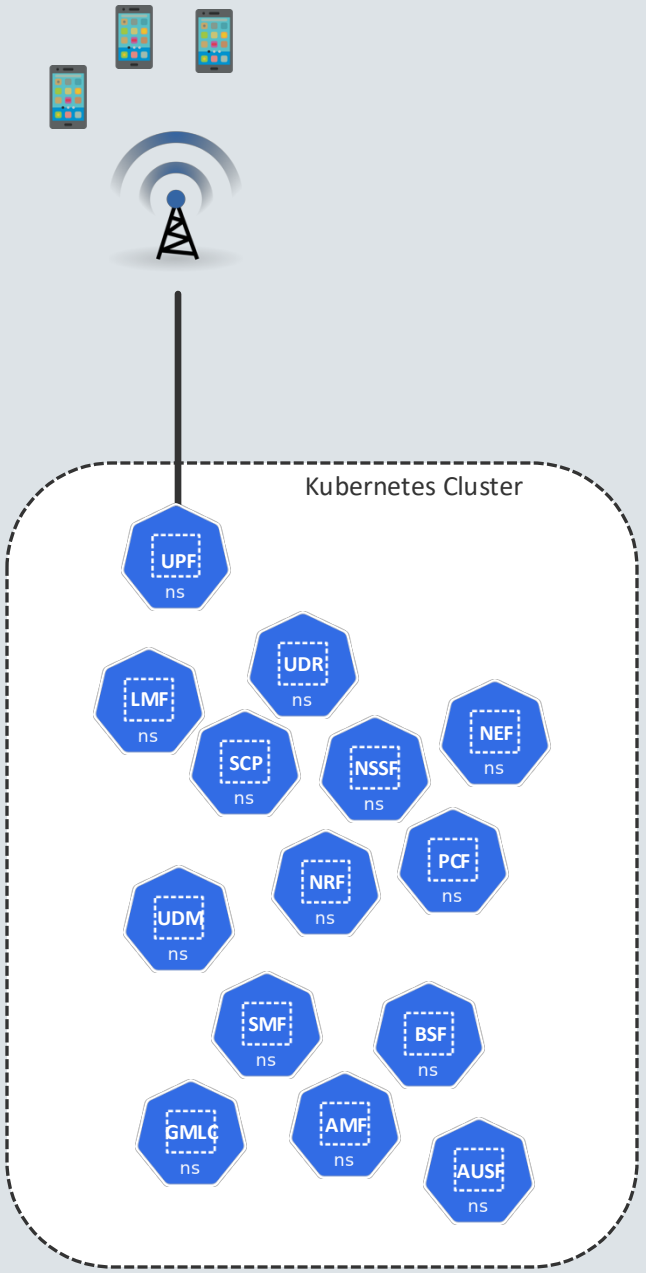
- is a «CNF» aka «Containerized Network Function»
  - e.g. Router (UPF), Authentication Service (AUSF)
- Deployed using Helm

Configuration is done via

- Helm Values
- Other Configuration Interfaces

Scale

- A development environment contains ~2000 pods
- A total of 5000 interdependent configuration parameters



What Is a 5G Core?

IP addresses

Subnets

VLANs

DNS Records

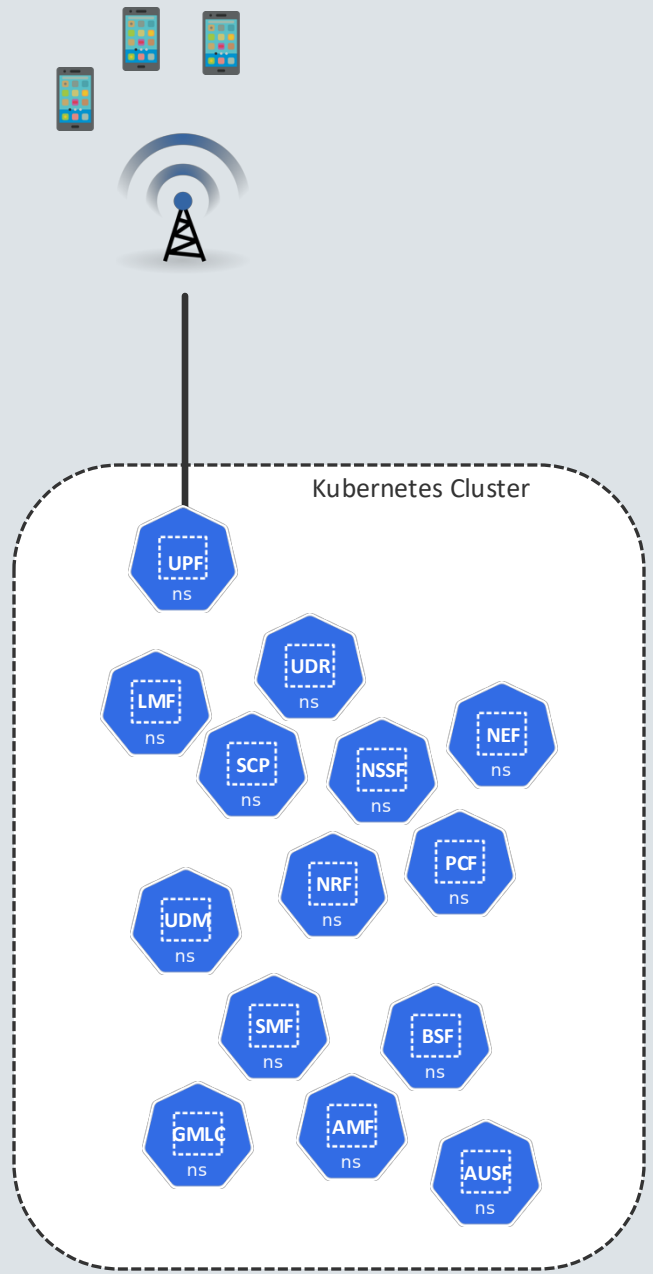
Network function variables

Infrastructure variables

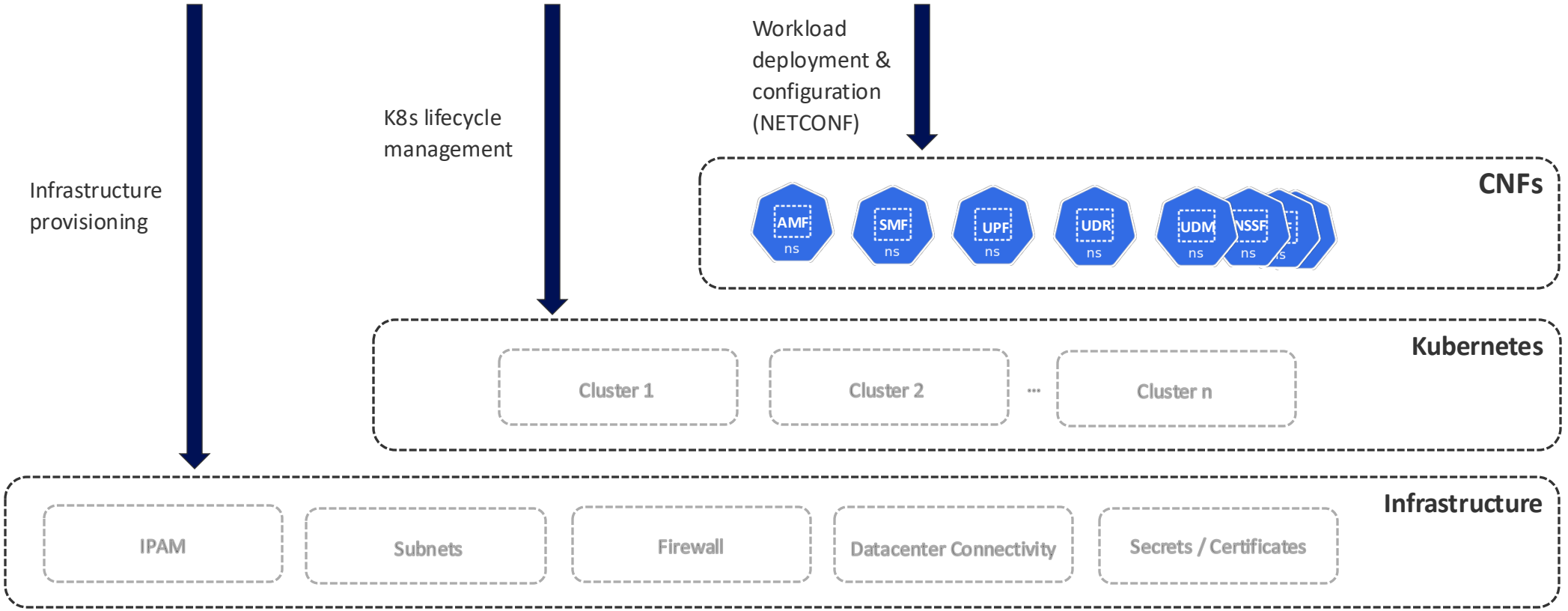
Network function-Network function mapping

Secret references

Certificate references



Cloud Native Resource Orchestration



Kubernetes Operators

Kubernetes Operators: Why?

K8s is well Established

Kubernetes is broadly adopted and Know-How is growing globally.

Reconcile Loop

Self-healing by design.

Community

Kubebuilder has a vivid community.

Kubernetes Resource Model (KRM)

API extensions

Custom Resource Definitions extend the Kubernetes API.

e.g. API definition for NetBox PrefixClaim

CRs as Instances

Custom Resources instantiate a CRD.

e.g. NetBox PrefixClaim resource

Business Logic

Use of Operators or templates to run custom logic

e.g. Assemble a config, Self healing

What Are Our Requirements?

KRM Based

Input and Output can be KRM resources

Garbage Collection

Resources belonging together are lifecycled together

Output Flexibility

Generated resources stored with various backends

Reconciliation

Output is updated as Input is changed

Functions

Ability to define helper functions (e.g. string manipulation)

Gradual Adoption

Migration from static to dynamic taken at own pace.

But What About Helm, Kustomize, Argo and Flux?

Limited Dynamic Assembly

We cannot use live Kubernetes resources as source of information for e.g. a helm release

No Custom Functions

We cannot invoke arbitrary code into Kustomize/Helm templating

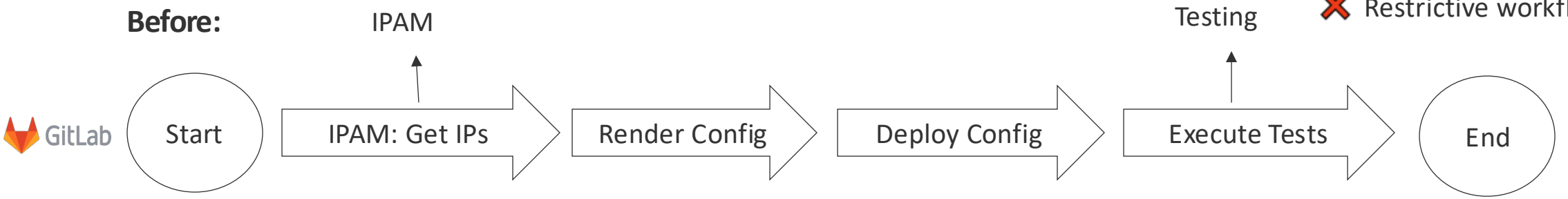
Only Partially KRM

Not all Resources involved follow KRM (e.g. Helm Values)

From the production pipeline to the conditional dance

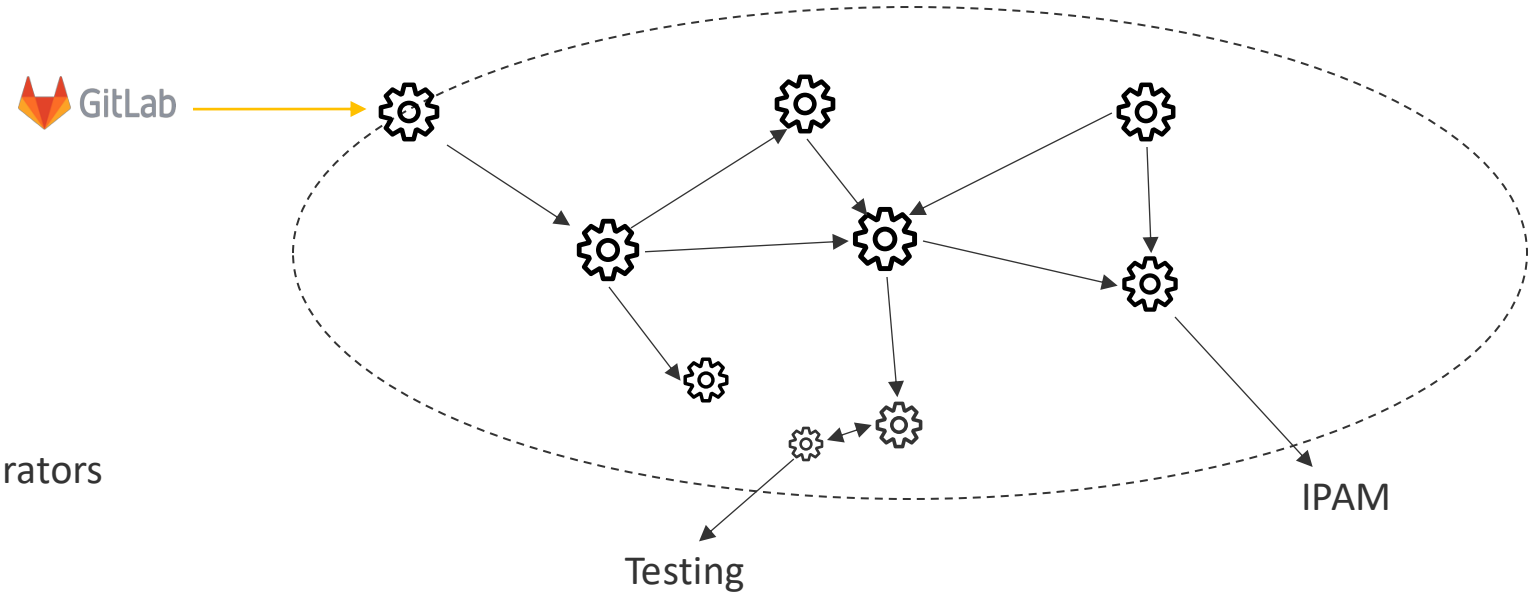
- ✗ No Reconciliation
- ✗ Fire and forget
- ✗ No standard interfaces
- ✗ Restrictive workflow

Before:



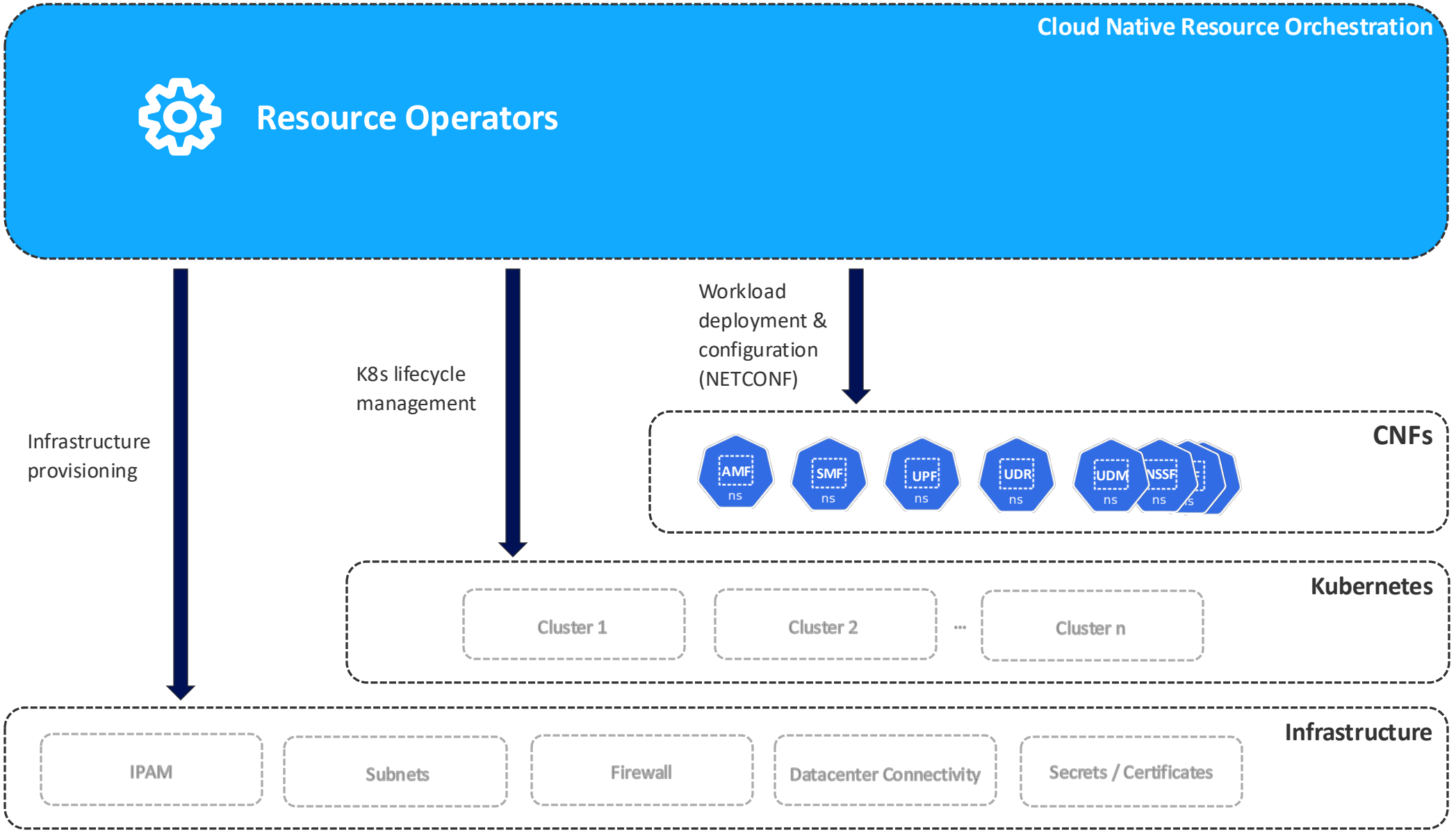
Now:

- ✓ Constant Reconciliation
- ✓ Standard interfaces
- ✓ Complex workflows possible

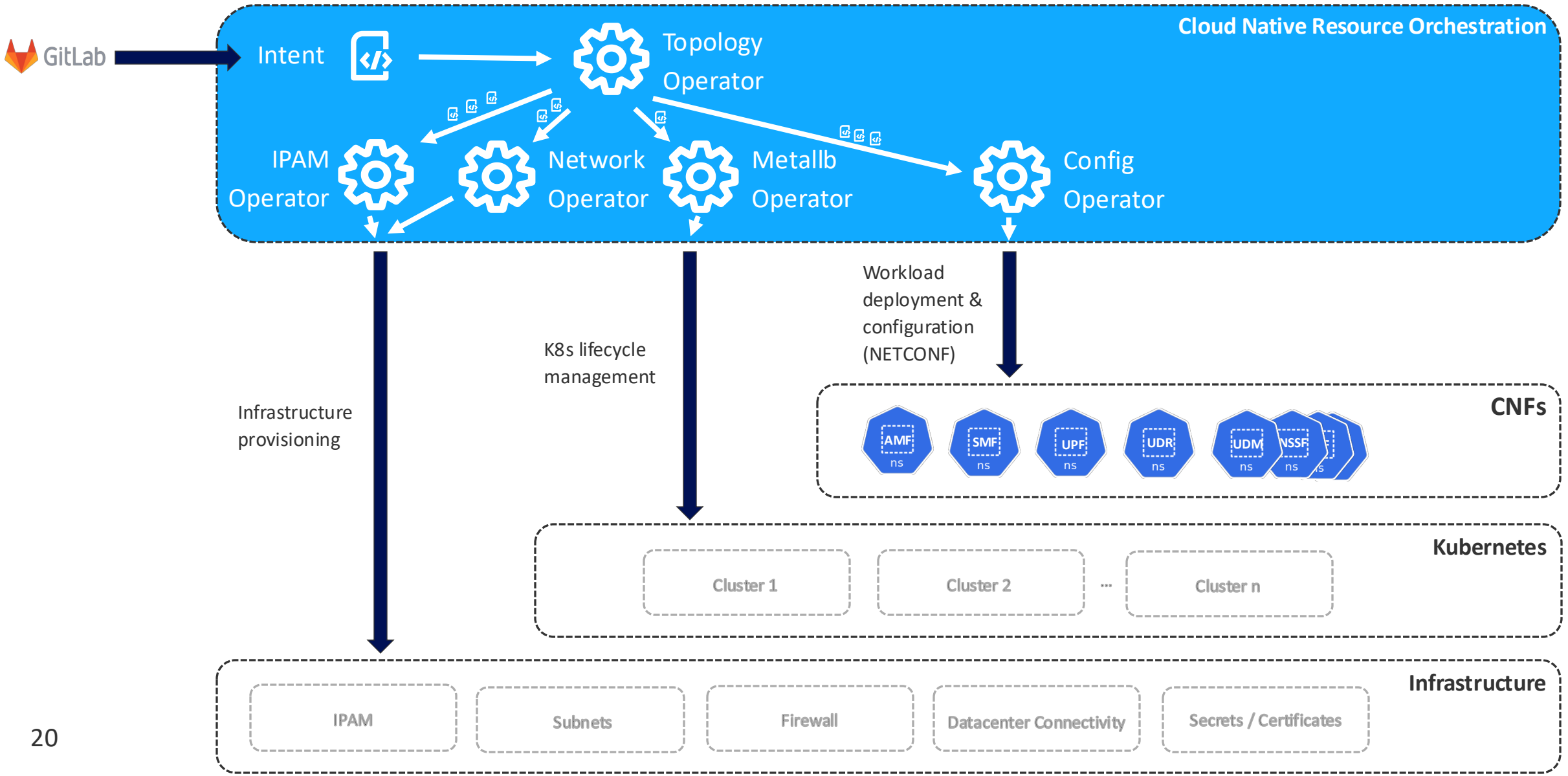


Operators

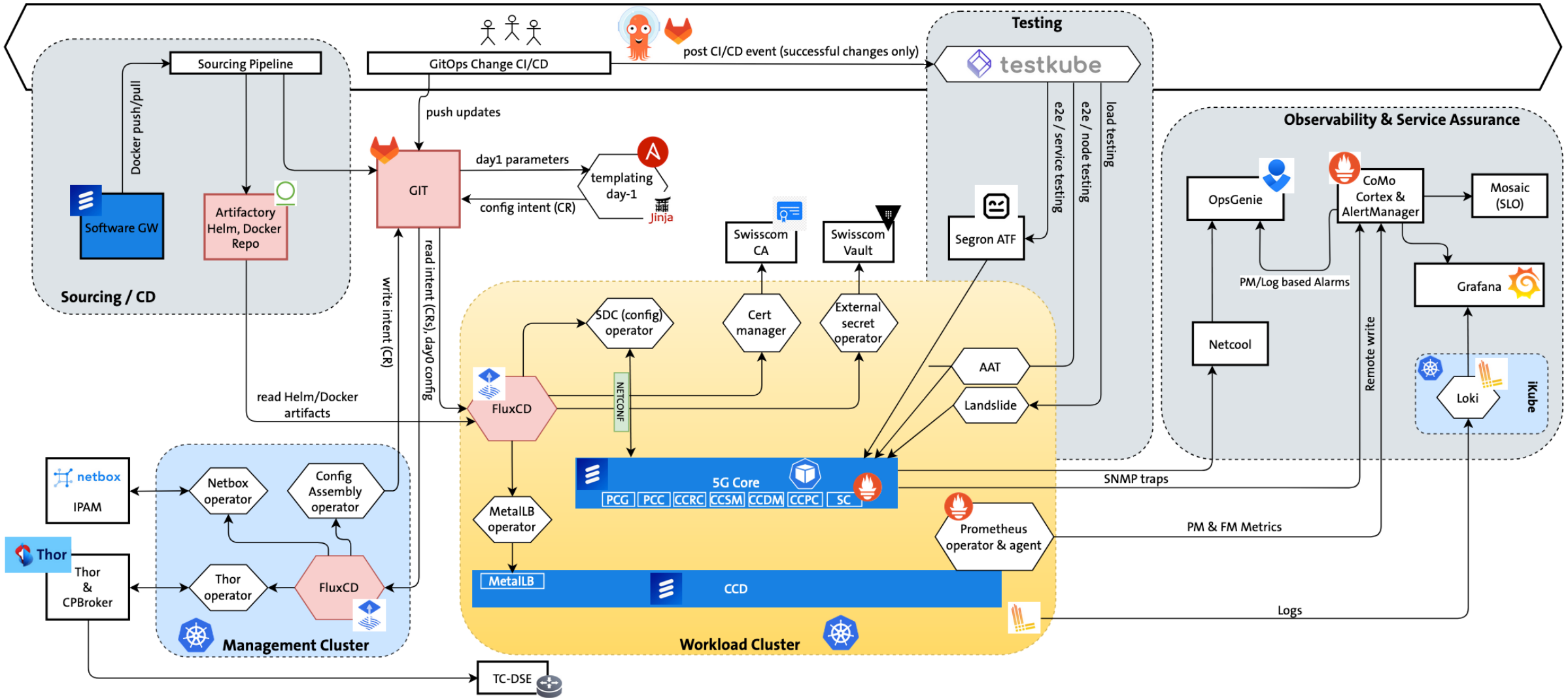
Cloud Native Resource Orchestration



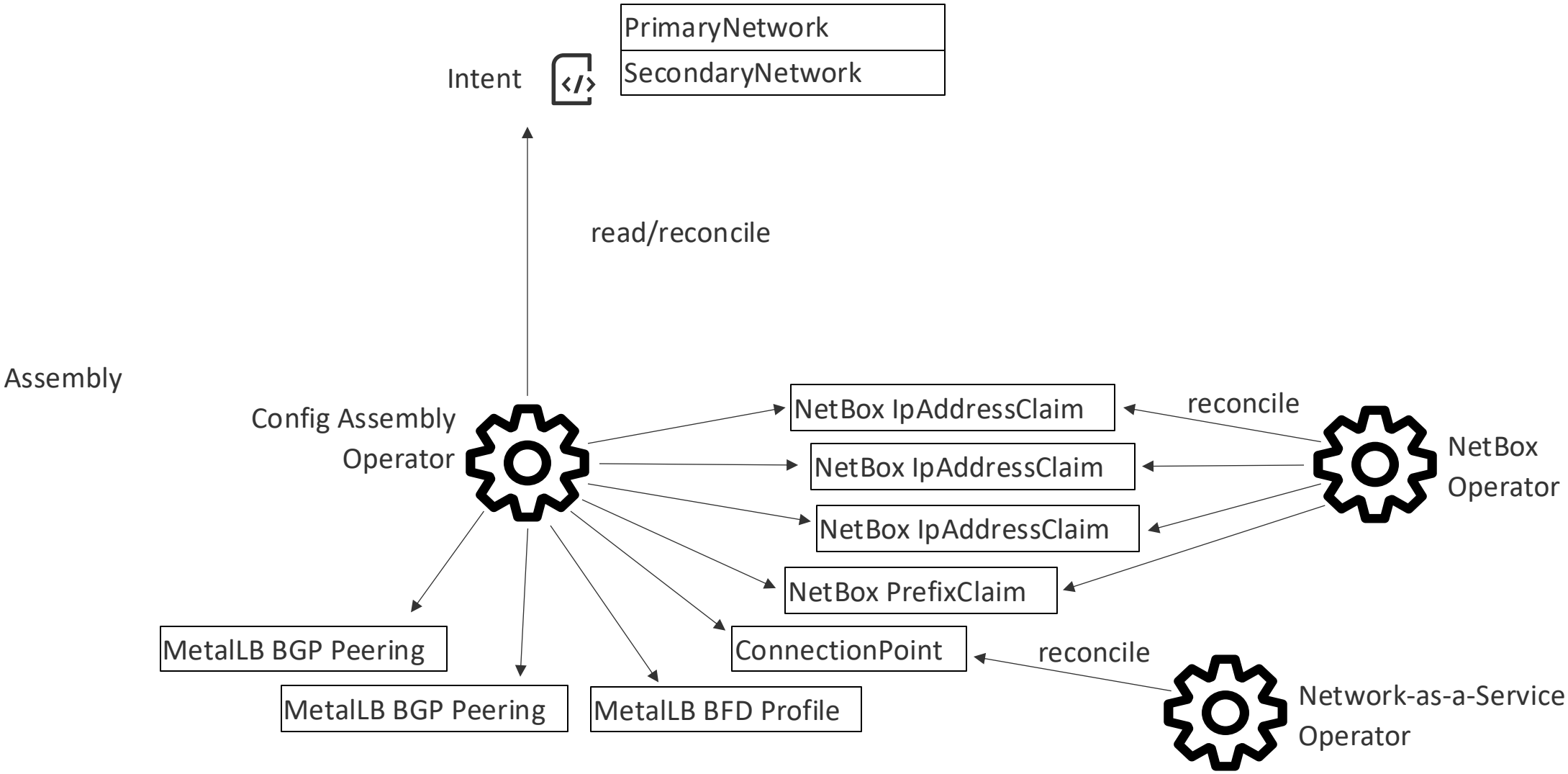
Cloud Native Resource Orchestration



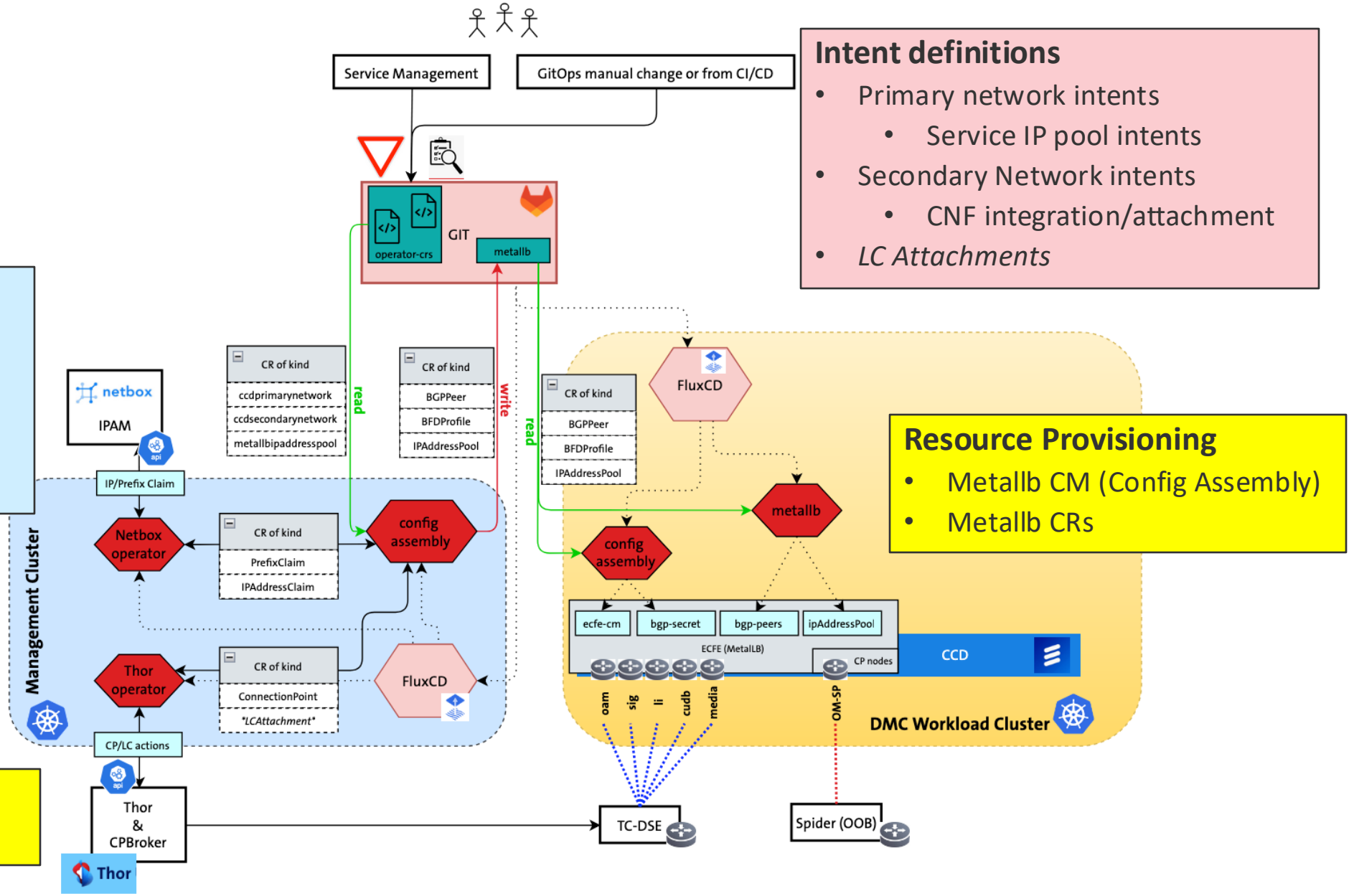
DMC Automation, Big Picture



Configuration Abstraction with Network Topology Operator



Demo: Primary Network Provisioning, including TC-DSE and Metallb config



Intent definitions

- Primary network intents
  - Service IP pool intents
- Secondary Network intents
  - CNF integration/attachment
- *LC Attachments*

Resource Orchestration

- Netbox (IPAM) operator
- Thor (TC-GW) operator
- Config Assembly (Network Topology) operator

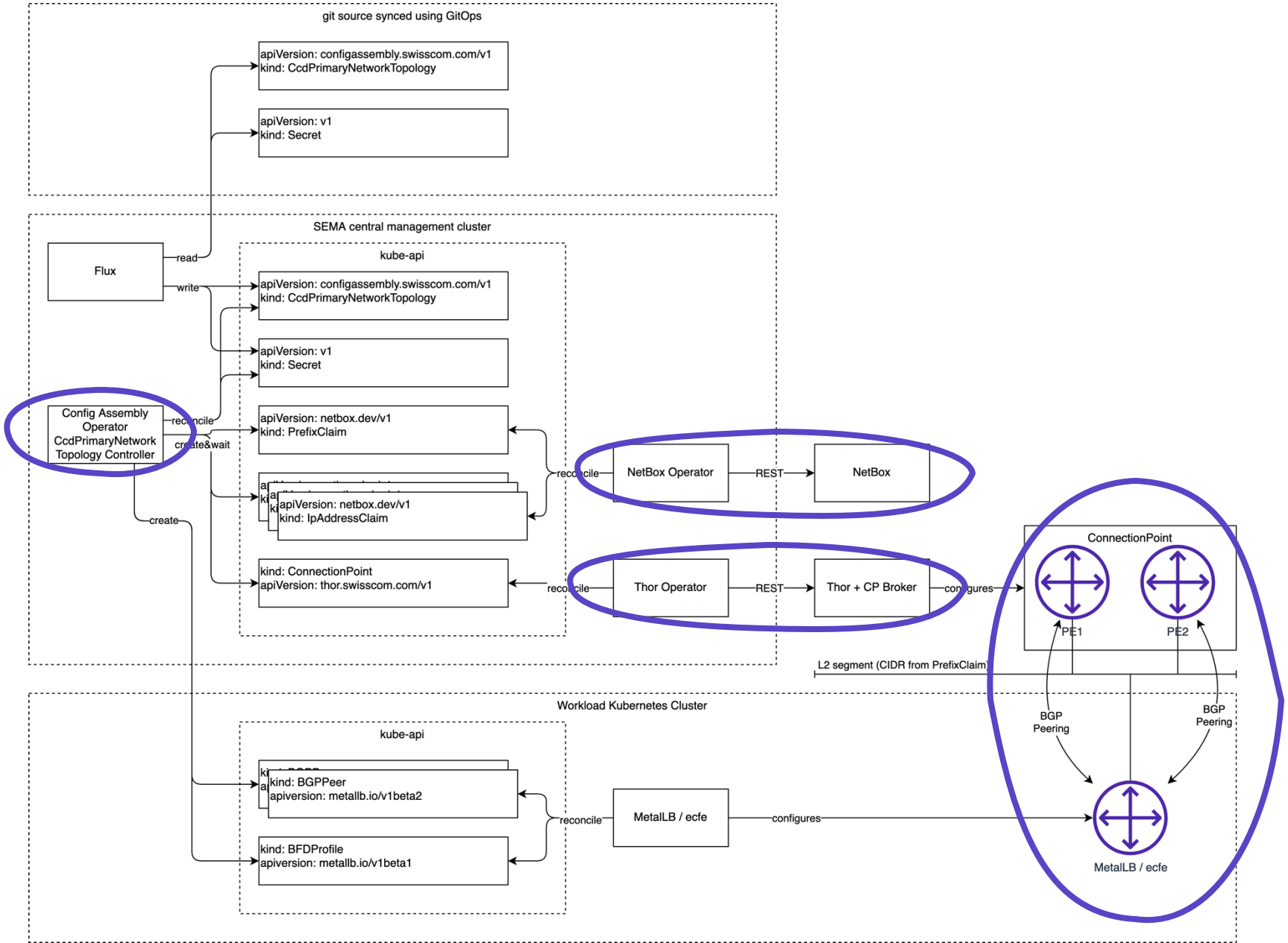
Resource Provisioning

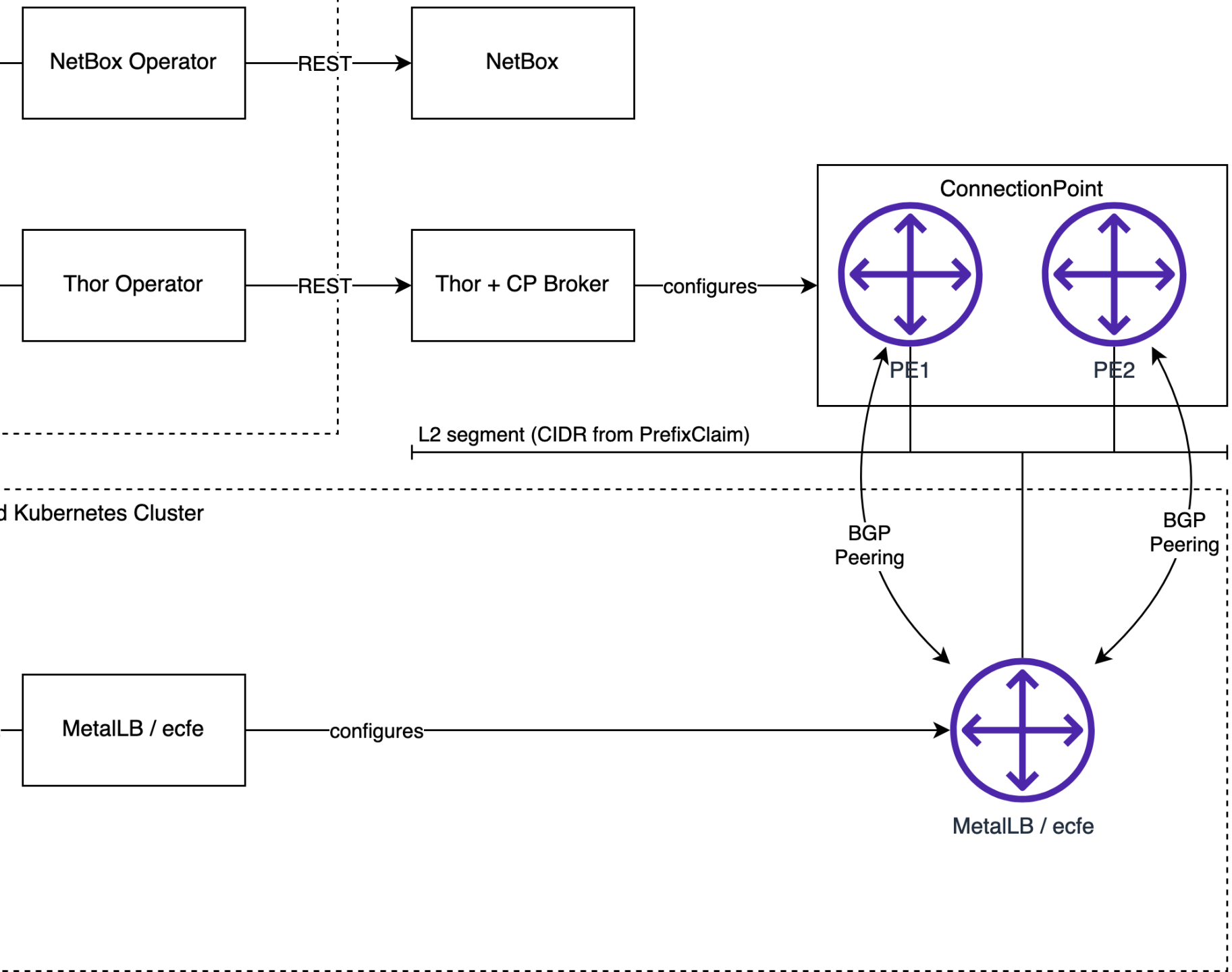
- Metallb CM (Config Assembly)
- Metallb CRs

Thor Provisioning

- TC-DSE configuration

Config Assembly Operator CcdPrimaryNetworkTopology Controller Context



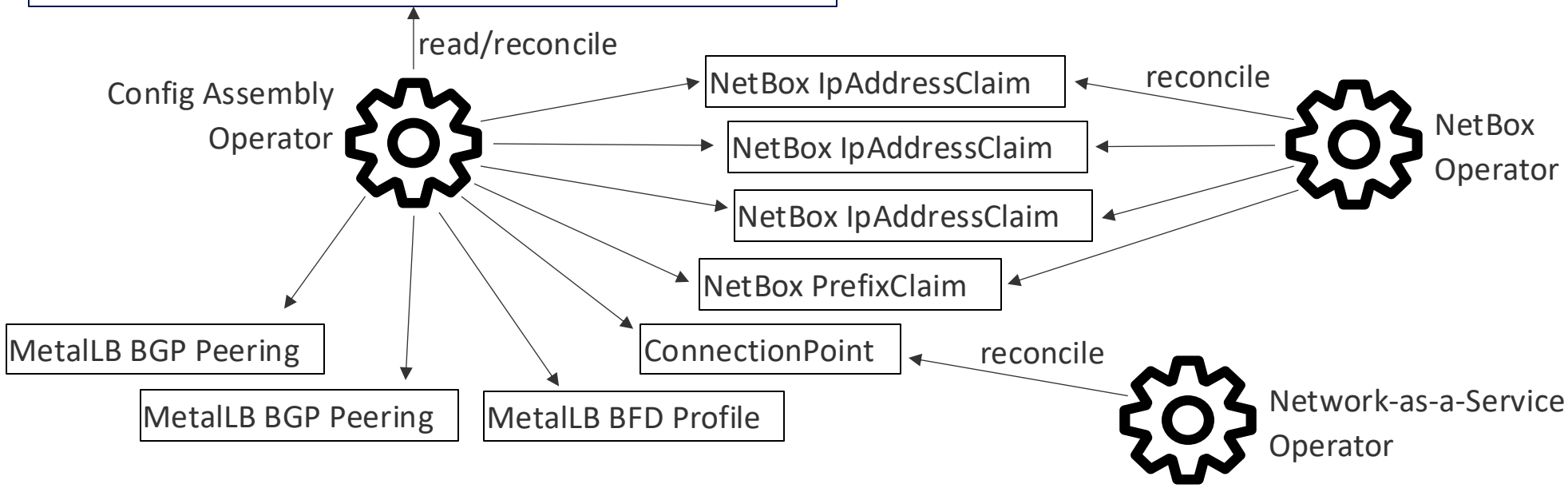

NetBox Operator
REST
NetBox
Thor Operator
REST
Thor + CP Broker
configures
ConnectionPoint
PE1
PE2
L2 segment (CIDR from PrefixClaim)
d Kubernetes Cluster
BGP
Peering
BGP
Peering
MetalLB / ecfe
configures
MetalLB / ecfe

Configuration Abstraction with Network Topology Operator

Intent

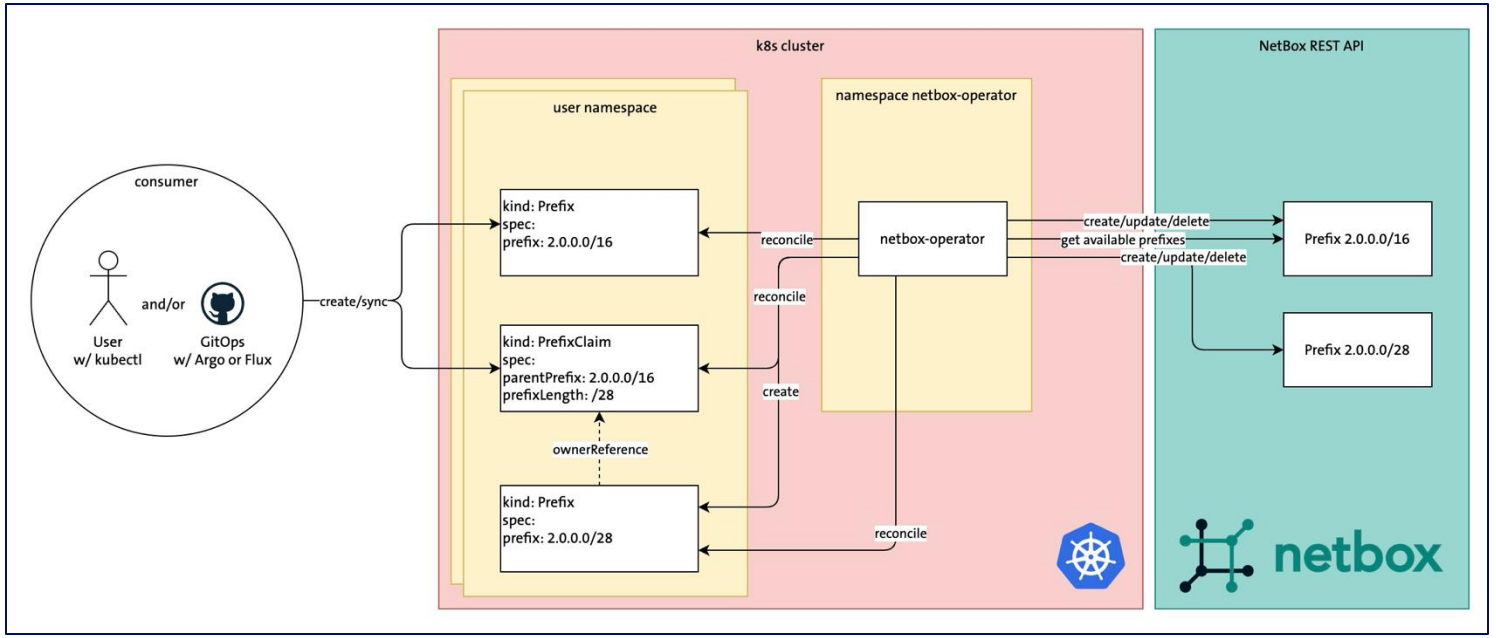
```
apiVersion: configassembly.swisscom.com/v1
kind: PrimaryNetworkTopology
spec:
  netBoxConfig:
    parentPrefixSelector:
      tenant: 5G
    prefixLength: /24
  metalLbAsn: 65000
```

Assembly



NetBox Operator

NetBox Operator, a tool designed to integrate NetBox resource management – for IPAM, DCIM and more – directly into your Kubernetes environment.

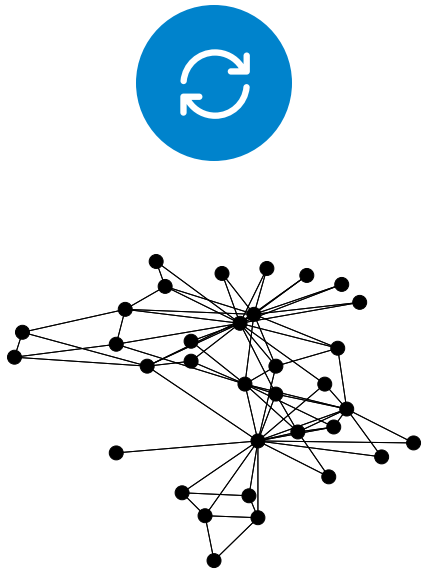


https://github.com/netbox-community/netbox-operator

CNCF Blog Post


Demo

How We've Overcome the Challenges



Network Automation Complexity

Automation of IPAM and Data Center Network

Limited Scalability

Easy to scale since using Kubernetes Operators

System Lock-in

Swisscom in full control of the code

Productivity Gains Using Intent-Driven Automation

Telco

CNF team creating IP LLD

2 days

Collect IP req., reservations in IPAM

2 days

Populate LLD (i.e. VLAN IDs)

1 day

Create & implement CRQs

5-10 days

Testing Activity & Monitoring

5 - 10 days

TechCo

CNF team creating high level intent

2 hrs

create lower-level intents

3 secs

Instantiate intents / create CRs

3-5 mins

Integrated testing and observability

2 hrs and continuous

Metrics & Logs

Our Learnings & Suggestions

... on network automation

Avoid

Checking in low level configurations in Git
Things like IP Addresses, VLAN IDs

Complicated configurations
Avoid unnecessary layers of abstraction

Ignore tooling gaps
Avoid using tools like Ansible or Jinja that aren't designed with Kubernetes in mind

Neglect Industry Relevance
Don't assume KRM challenges and solutions are unique to your sector

Aim to

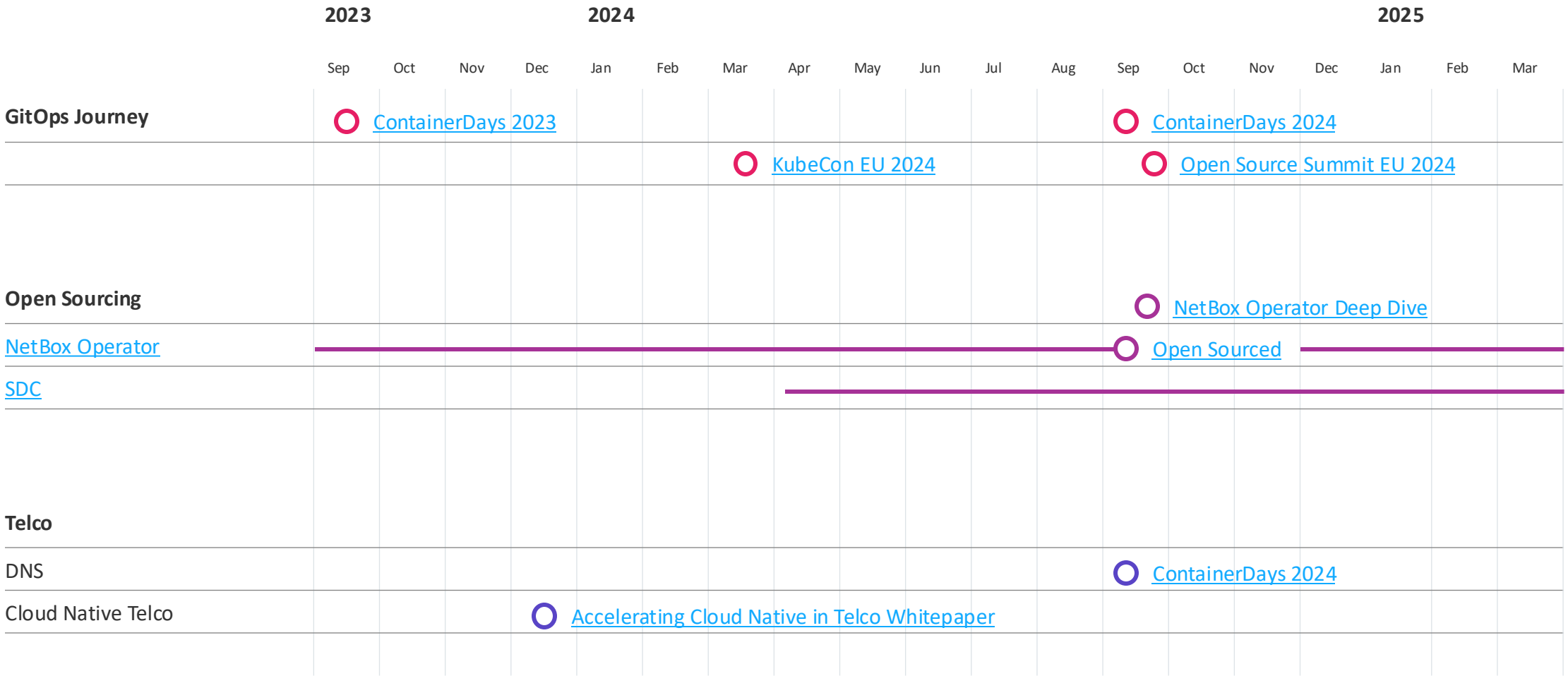
Leverage abstraction
Simplify complex configurations by focusing on essential controls

Reuse cloud native tools to be in-band with K8s
E.g. Flux, Argo, Testkube, cert-manager

Stay Agile
Continuously refine and adapt your KRM practices to evolving needs and technologies

Contribute to the ecosystem
Share your code

Our Publications and Related Talks



Thanks!