NEPHIO

https://nephio.org/about/

Agenda

- ❏ Problem Statement (Why)
- ❏ Introduction (What)
- ❏ Community Details
- ❏ NephIO Internal
  - ❏ K8s based Automation Control Plane
  - ❏ Declarative automation framework
- ❏ Reference architecture & NephIO Plan
- ❏ Benefits of Nephio
- ❏ NephIO relation with other existing open source (ONAP, EMCO, Anuket, Terraform)
- ❏ Google Open source (planned to be leveraged in NephIO)
- ❏ Summary

Problem Statement (Why NephIO)

Technologies like distributed cloud enable on-demand, API-driven access to the edge.

Current Orchestration model for multi vendor, distributed cloud options has multiple problems:

1. Imperative.
2. Mainly fire-and-forget orchestration methods struggle to take full advantage of the dynamic capabilities of these new infrastructure platforms.
3. Doesn’t solve end to end provisioning of infrastructure, on demand network function deployment and LCM and configurations.

Nephio uses new approaches that can handle the complexity of provisioning and managing a multi-vendor, multi-site deployment of interconnected network functions across on-demand distributed cloud.

The solution is intended to address the **initial provisioning of the network functions** and **the underlying cloud infrastructure,** and also provide Kubernetes-enabled reconciliation to **ensure the network stays up through failures, scaling events, and changes** to the distributed cloud.

Current solutions are more or less doesn't solve entire set of problems i.e. infrastructure provision (including network, storage etc), network function deployment, network function configuration, and life cycle management.

NephIO - Introduction

Nephio is a Kubernetes-based intent-driven automation of

- Network functions and
- The underlying infrastructure that supports those functions.

It allows users to express high-level intent, and provides intelligent, declarative automation that can set up the **cloud and edge infrastructure**, render initial configurations for the network functions, and then deliver those configurations to the right clusters to get the network up and running.

NOTE : Nephio's goal is to develop open automation configuration in conformance to the O-RAN and 3GPP specifications.

NephIO to provide intent based mechanism to provision multi site infrastructure, deploy network function deployment, configure them and carry out their life cycle management.
Nephio mission is to simplify the deployment and management of multi-vendor **cloud** infrastructure and network functions across large scale **edge** deployments”

https://nephio.org/about/

Community Details

Roadmap: Not yet released
Date: **April 04, 2022**
Status of project: Not Known
Intention of the project: The mission of the Project is to deliver carrier-grade, simple, open, Kubernetes-based cloud-native intent automation, and common automation templates that materially simplify the deployment and management of multi-vendor cloud infrastructure and network functions across large scale edge deployments. The project seeks to enable faster onboarding of network functions to production including provisioning of underlying cloud infrastructure with a true cloud-native approach, and reduces costs of adoption of cloud and network infrastructure.
Probable Release Date: Not Known
License : Apache 2.0

Founding Organizations	Google Cloud, LF
Service Providers	Airtel, Bell, Elisa, Equinix, Jio, Orange, Rakuten Mobile, TIM, TELUS, Vapor IO, Virgin Media O2, WINDTRE
Network Function, Service and Infrastructure Vendors	Aarna Networks, ARM, Casa Systems, DZS, Ericsson, F5, Intel, Juniper, Mavenir, Nokia, Parallel Wireless, VMware

Very Recently Launched by Google & LF.

NephIO Internal

Nephio breaks down the problem into two primary areas:

- Kubernetes as a uniform automation control plane in each site to configure all aspects of the distributed cloud and network functions; and
- An Intent based automation framework that leverages Kubernetes declarative, actively-reconciled methodology along with machine-manipulable configuration to handle the complexity of these configurations.

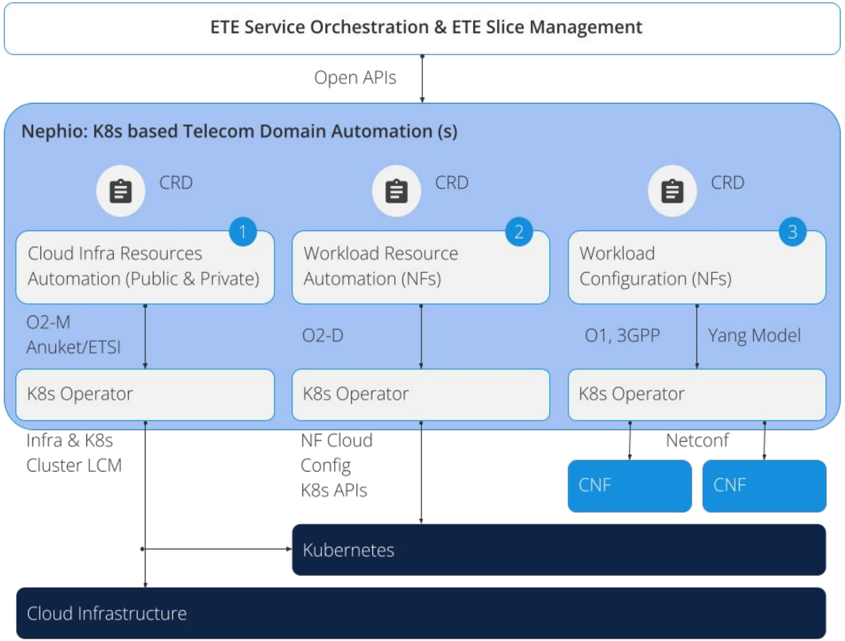
Two Major building blocks of NephIO

1. Kubernetes based control plane
2. Intent driven Declarative Automation Framework

K8s based Automation Control Plane

NEPHIO visualize configuration stack in three layers.

1. **Cloud automation layer:** Nephio publishes Kubernetes-based CRDs and operators for each public and private cloud infrastructure automation that is in conformance to industry standards (e.g., O-RAN O2 interface). These CRDs and operators can make use of existing Kubernetes-based ecosystem projects as pluggable southbound interfaces (e.g., Google Config Connector, AWS Controllers for Kubernetes, and Azure Service Operator), providing an open integration point and more uniform automation across those providers.
2. **Workload resource automation**: covers the configuration for provisioning network function containers and the requirements those functions have for the node and network fabric.
   This includes the native Kubernetes primitives and industry extensions such as multi-network Pods, SR-IOV, and similar technologies. Today, using these effectively requires complex Infrastructure-as-Code templates that are purpose built for specific network functions.(Eg. Multus etc).
3. **Workload configuration:** Nephio initially provides tooling and libraries to assist vendors with integrating existing Yang and other industry models with NephIO, in conformance to the standards (e.g., 01, 3gpp interfaces specs).   NOTE: To fully realize the benefits of cloud native automation, these models will need to migrate to Kubernetes CRDs,.



ETE Service Orchestration & ETE Slice Management
Open APIs
Nephio: K8s based Telecom Domain Automation (s)
CRD
CRD
CRD
1
2
3
Cloud Infra Resources Automation (Public & Private)
Workload Resource Automation (NFs)
Workload Configuration (NFs)
O2-M Anuket/ETSI
O2-D
O1, 3GPP
Yang Model
K8s Operator
K8s Operator
K8s Operator
Infra & K8s Cluster LCM
NF Cloud Config K8s APIs
Netconf
CNF
CNF
Kubernetes
Cloud Infrastructure

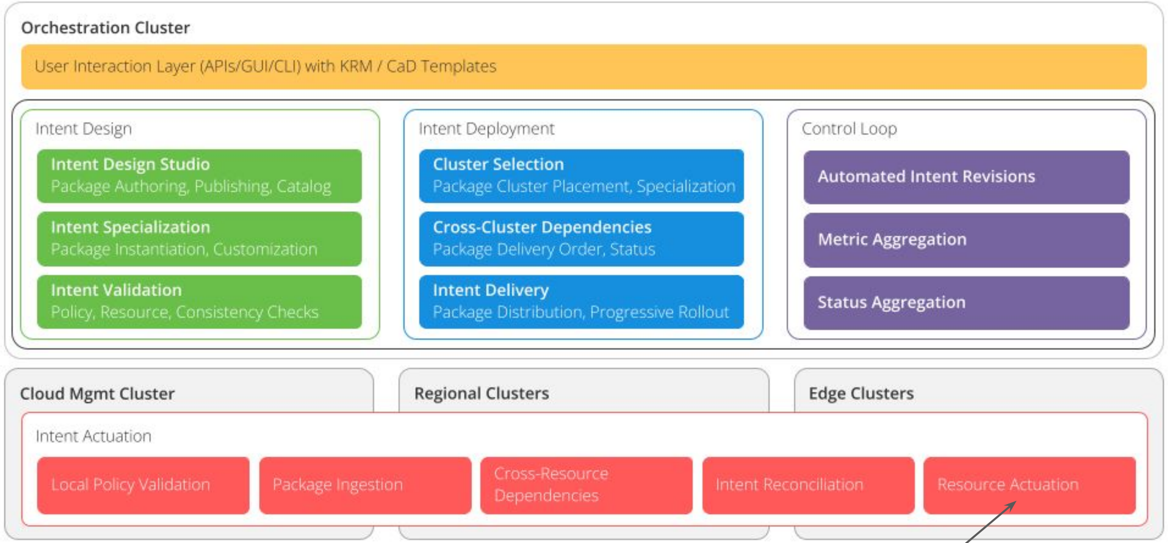
<u>3 Layer configuration stack</u>

- <u>Cloud automation layer</u> provides 3gpp/ORAN conformed CRDs which in turn will call public(AWS, google, azure)/private(Anuket) cloud APIs to provide infrastructures: **It means NephIO is going to provide a wrapper and will be leveraging existing VIM for Infra Provisioning.**
- <u>Workload Resource automation layer</u> provides CRD to deploy network functions and their related requirements eg multi network, storage etc.
- <u>Workload configuration layer</u> provide CRD to configure network function which in turn call 3gpp conformed APIs.

Nephio is establishing open, extensible Kubernetes Custom Resource Definition (CRD) models for each layer of the stack i.e. NephIO control plane will be based on Kubernetes

NephIO plan is to only support CNF and not VNF.

Intent driven Declarative Automation Framework



Orchestration Cluster
User Interaction Layer (APIs/GUI/CLI) with KRM / CaD Templates
Intent Design
Intent Design Studio
Package Authoring, Publishing, Catalog
Intent Specialization
Package Instantiation, Customization
Intent Validation
Policy, Resource, Consistency Checks
Intent Deployment
Cluster Selection
Package Cluster Placement, Specialization
Cross-Cluster Dependencies
Package Delivery Order, Status
Intent Delivery
Package Distribution, Progressive Rollout
Control Loop
Automated Intent Revisions
Metric Aggregation
Status Aggregation
Cloud Mgmt Cluster
Regional Clusters
Edge Clusters
Intent Actuation
Local Policy Validation
Package Ingestion
Cross-Resource Dependencies
Intent Reconciliation
Resource Actuation

Figure 2: Nephio Functional Building Blocks

K8s based Automation Control Plane

Figure 2 provides an overview of Nephio's functional components of declarative automation framework.

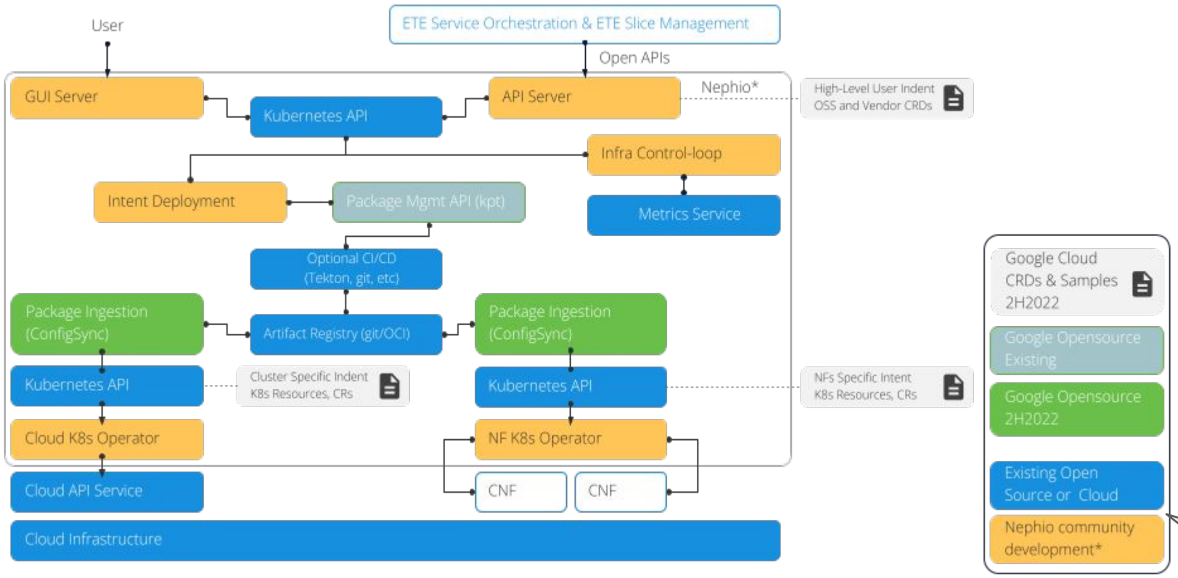
The Nephio automation framework is built on the Google Open Source projects `kpt` and `ConfigSync` and implements the Configuration-as-Data approach to configuration management.

This enables users to author, review and publish configuration packages which may then be cloned and customized to deploy network functions.

This customization can be fully automated, or mix-and-match automated and human-initiated changes without conflicts and without losing the ability to easily upgrade to new versions of the packages.

A GUI will be provided to user to design and deploy intent.
Packaging and distributing of configurations to different site will be managed by underlying layers using config sync and KPT (google open source)
The k8s based automation discussed in previous pages is resource actuation component shown in this picture.

Reference Architecture & Plan



*Reference architecture to evolve in the community

NephIO Plan

1. To realize a service one can use GUI, Open APIs or an high level declarative user intent using CRDs.
2. The API server call intent deployment to validate the intent.
3. Package management service(kpt) will help to package the configurations generated from intent and it will push to artifacts registry.
4. Config sync then push the configurations various clusters using different k8s operators.
5. Cloud k8s operators will call different public/private cloud API to provision infrastructure.
6. NF k8s operator will call APIs to deploy and configuration for a network function.

Project Plan includes:

1. Leverage some existing Google Open source (kpt (already open source by google) & ConfigSync (google planned open source))
2. Develop Intent related components and Operators for infrastructure and NF provisioning as part of NephIO project.

Note: Kubernetes as a basis for all development .i.e all development will be in form of kubernetes custom resource and custom controllers.

Benefits of Nephio?

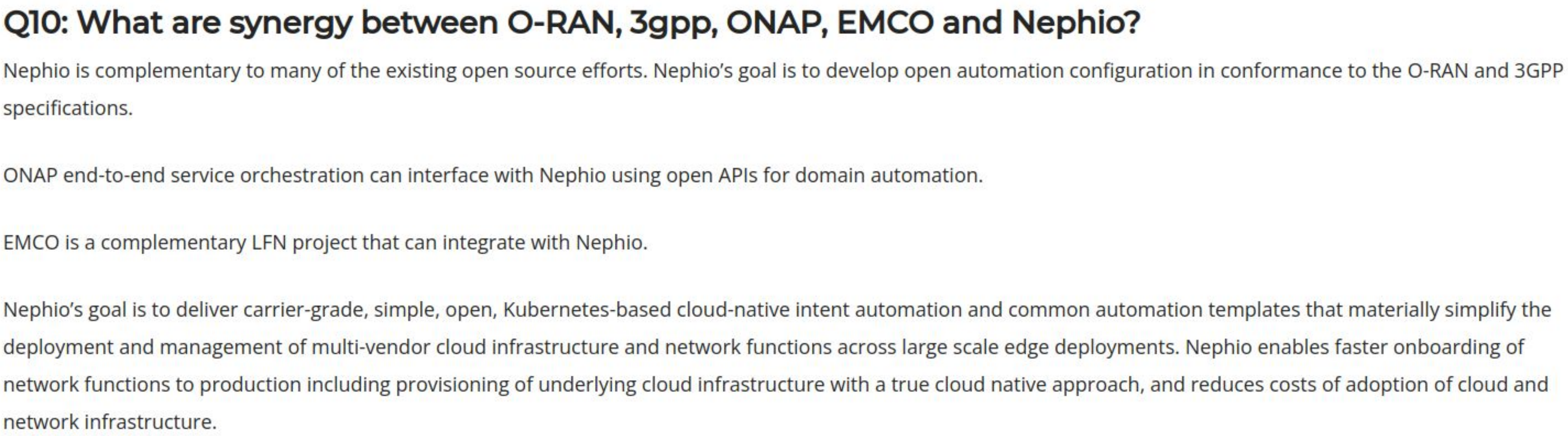
For **telcos**, Nephio provides an **open, simple, widely adopted Kubernetes based cloud-native automation** that enables **multi-vendor support, faster onboarding, easier life-cycle management, embedded control-loop, active reconciliation and service assurance** — reducing cost by efficiency and agility.

For **cloud providers**, Nephio provides a common cloud-based automation framework based on well-proven Kubernetes technology. This minimizes the levels of custom automation solutions needed for each application. Kubernetes based automation enables faster development with known technology and assures network functions will deploy and run reliably on top of the cloud.

For **network function vendors,** Nephio enables easier **multi-vendor integration with cloud providers**, makes **network function onboarding to cloud easier**, and improves the overall customer experience with simple and reliably integrated cloud native automation.

Relation with Other projects

Synergies with other projects



Q10: What are synergy between O-RAN, 3gpp, ONAP, EMCO and Nephio?

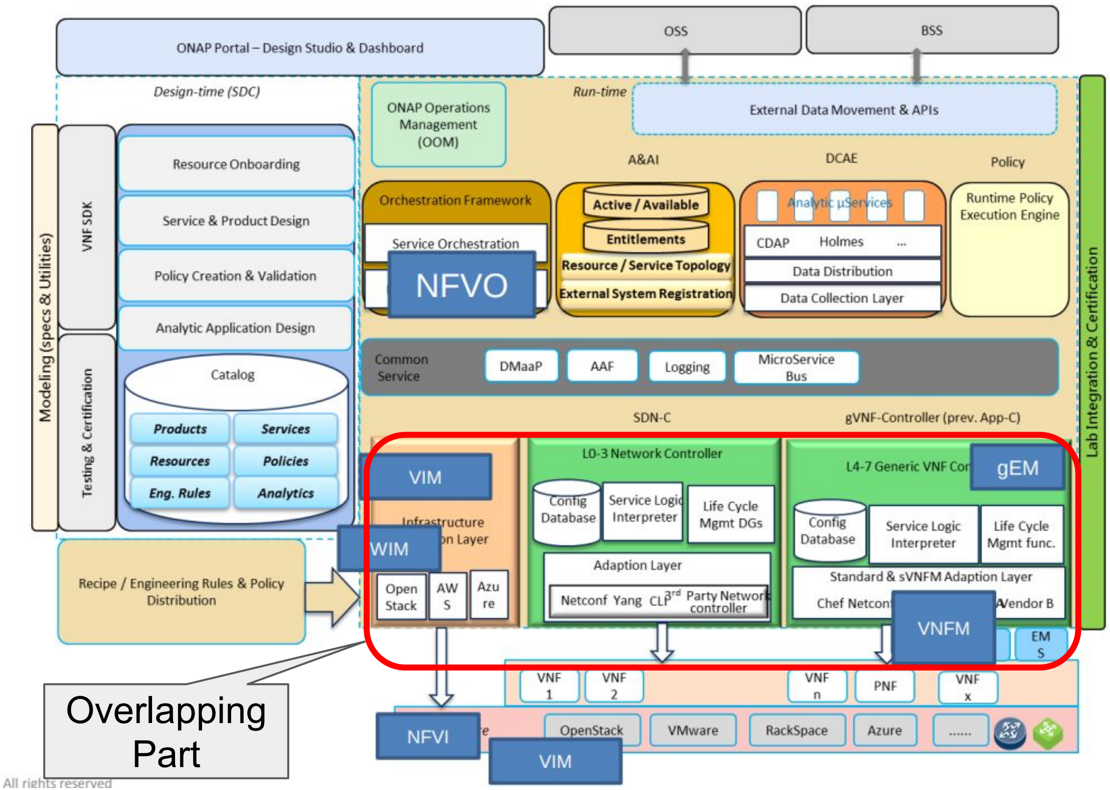
Nephio is complementary to many of the existing open source efforts. Nephio’s goal is to develop open automation configuration in conformance to the O-RAN and 3GPP specifications.

ONAP end-to-end service orchestration can interface with Nephio using open APIs for domain automation.

EMCO is a complementary LFN project that can integrate with Nephio.

Nephio’s goal is to deliver carrier-grade, simple, open, Kubernetes-based cloud-native intent automation and common automation templates that materially simplify the deployment and management of multi-vendor cloud infrastructure and network functions across large scale edge deployments. Nephio enables faster onboarding of network functions to production including provisioning of underlying cloud infrastructure with a true cloud native approach, and reduces costs of adoption of cloud and network infrastructure.

NephIO-ONAP Integration/Relation



Feature/ Function Overlap between ONAP and NephIO:

1. Network Function Deployment
   a. ONAP: Workload deployment done using ONAP Multi-VIM & requirement provisioning using Controllers (SDNC, CDS)
   b. NephIO: New kubernetes operators
2. Network Function Configuration
   a. ONAP: Done using ONAP controllers (APPC, CDS)
   b. NephIO: New kubernetes operators

<u>NephIO vs ONAP</u>

ONAP provides service orchestration, whereas NephIO is at domain level

1. Overlapping component/features among ONAP & NephIO
2. One Possible integration between ONAP & NephIO: ONAP leveraging NephIO for Domain Orchestration as an alternative.

NephIO vs Terraform

Terraform	NEPHIO
Meant for **Infrastructure provisioning** in Multi cloud scenario. Terraform is used for dynamically provision infrastructure across multiple cloud providers. Terraform **not good** for **application deployment or configuration**. https://medium.com/google-cloud/dont-deploy-applications-with-terraform-2f4508a45987	Provides **E2E functionality** including 1. Infrastructure provisioning 2. NF Deployment 3. NF Configurations.
.tf format. (Declarative), executes using scripts.	K8s based declarative configuration. Can be executed using API or yaml files(with CRDs).

1. Terraform is basically meant for Infrastructure Provisioning whereas NephIO is handling E2E.

Others : NephIO-EMCO & NephIO-Anuket

NephIO - Anuket

Overlap Function: Infrastructure Provisioning. Note: Anuket is limited to infrastructure provisioning.

Possible Integration: Anuket provide wrapper around k8s & openstack to provision infrastructure.
NephIO can leverage Anuket for infrastructure provisioning.

NephIO - EMCO

EMCO is a complementary LFN project that can integrate with Nephio. EMCO provides means to orchestrate network functions across multiple k8s cluster. But it doesn't provision infrastructure for it. Nephio can provision infra and use EMCO to deploy/configure network functions.

Google Open Source
Planned to be leveraged in NephIO

Google KPT

Kpt (pronounced “kept”) is an OSS tool for building declarative workflows on top of resource configuration.

kpt is a toolkit to help you manage, manipulate, customize, and apply Kubernetes Resource configuration data files.

- Fetch, update, and sync configuration files using git.
- Examine and modify configuration files.
- Generate, transform, validate configuration files using containerized functions.
- Apply configuration files to clusters.

https://googlecontainertools.github.io/kpt/

https://kpt.dev/book/

https://kpt.dev/book/02-concepts/

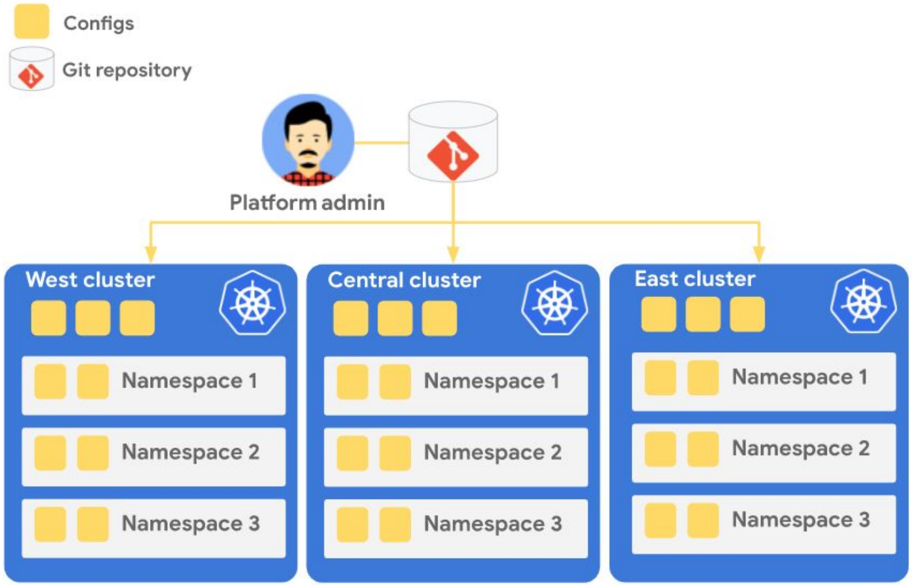
An example usage of KPT, and comparison with helm

https://www.meanpug.com/kubernetes-kpt-in-the-wild/#:~:text=Helm%20is%20a%20full%2Dfeatured,%2C%20not%20competitor%2C%20to%20kpt.

Google Config Sync

Config Sync lets cluster operators and platform administrators deploy consistent configurations and policies. You can deploy these configurations and policies to individual Kubernetes clusters, multiple clusters that can span hybrid and multi-cloud environments, and multiple namespaces within clusters.

Using the same principles as Kubernetes itself, **Config Sync continuously reconciles the state of registered clusters with a central set of Kubernetes declarative configuration files called configs. Configs are stored in one or more Git repositories, and Config Sync keeps them consistent with the configured Kubernetes objects.** This GitOps approach (sometimes also referred to as configuration as code) lets you manage and deploy common configurations with a process that is auditable, transactional, reviewable, and version-controlled.



Summary

Pros:

1. **Intent driven approach** to **handle E2E** from Infrastructure provisioning to NF deployment and their configuration.
   a. Single intent hiding details from user and handling all user needs.

2. NephIO **leverages Kuberenetes** for control plane. If everything is a Kubernetes resource, be it hosting or deployable asset, it makes orchestration of service features much easier.

3. Capable to **support Multi cloud as well as Edge**: It can supports uses cases related to 5G, IOT etc.

4. Founding organization from broad sectors including network vendors, infrastructure vendor, cloud provider, operators

Cons:

1. Integration with existing Service orchestration platform like ONAP is not planned, like considering how to build something designed as a higher layer, the new fourth layer, to the Nephio structure and focus on intent for service orchestration

2. One bad signal, though, is the fact that the majority of those ten operators who launched NFV in 2012 are not, so far, part of the project.

3. Support for VNF is not planned.

Resources

https://nephio.org/about/

https://nephio.org/wp-content/uploads/sites/6/2022/04/Nephio-Project-Technical-Charter-Final-4-11-2022-1.pdf

https://nephio.org/frequently-asked-questions/

https://cloud.google.com/blog/topics/developers-practitioners/build-platform-krm-part-1-whats-platform

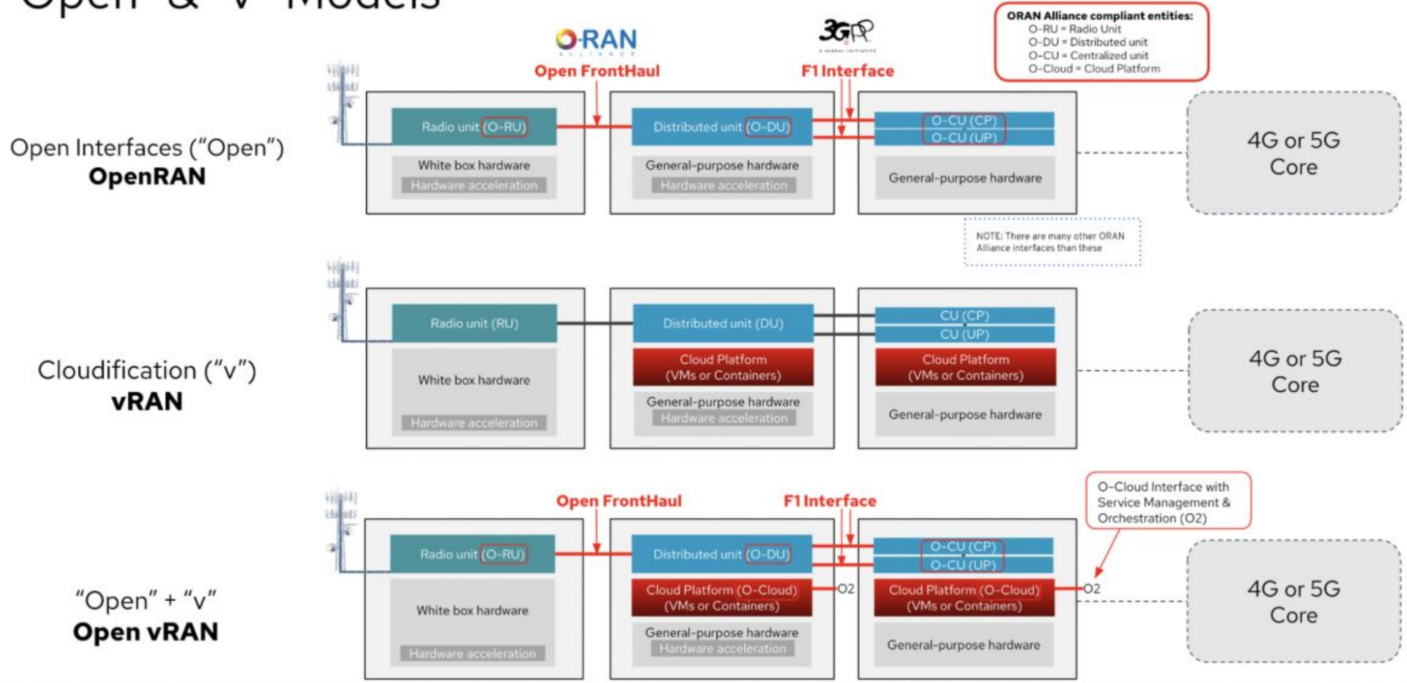
https://cloud.google.com/blog/topics/developers-practitioners/build-platform-krm-part-2-how-kubernetes-resource-model-works

https://www.linuxfoundation.org/press-release/the-linux-foundation-and-google-cloud-launch-nephio-to-enable-and-simplify-cloud-native-automation-of-telecom-network-functions/#:~:text=Nephio's%20goal%20is%20to%20deliver,across%20large%20scale%20edge%20deployments.

THANKS

Open vRAN

"Open" & "v" Models



Open Interfaces ("Open")
OpenRAN

O-RAN

Open FrontHaul

3GPP

F1 Interface

ORAN Alliance compliant entities:
O-RU = Radio Unit
O-DU = Distributed unit
O-CU = Centralized unit
O-Cloud = Cloud Platform

Radio unit (O-RU)

White box hardware

Hardware acceleration

Distributed unit (O-DU)

General-purpose hardware

Hardware acceleration

O-CU (CP)

O-CU (UP)

General-purpose hardware

4G or 5G Core

NOTE: There are many other ORAN Alliance interfaces than these

Cloudification ("v")
vRAN

Radio unit (RU)

White box hardware

Hardware acceleration

Distributed unit (DU)

Cloud Platform (VMs or Containers)

General-purpose hardware

Hardware acceleration

CU (CP)

CU (UP)

Cloud Platform (VMs or Containers)

General-purpose hardware

4G or 5G Core

"Open" + "v"
Open vRAN

Open FrontHaul

F1 Interface

O-Cloud Interface with Service Management & Orchestration (O2)

Radio unit (O-RU)

White box hardware

Hardware acceleration

Distributed unit (O-DU)

Cloud Platform (O-Cloud) (VMs or Containers)

General-purpose hardware

Hardware acceleration

O2

O-CU (CP)

O-CU (UP)

Cloud Platform (O-Cloud) (VMs or Containers)

General-purpose hardware

O2

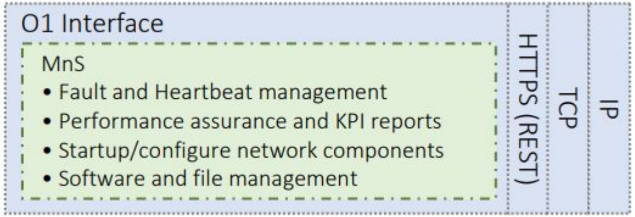
4G or 5G Core

Key	RRH = Remote radio head
	BBU = Baseband unit
		RU = Radio unit	DU = Distributed unit
			CU = Centralized unit
				UP = User plane
				CP = Control plane

https://www.redhat.com/en/blog/open-ran-and-o-ran-brief

ORAN- O1 Interface

The O1 interface connects the SMO to the RAN managed elements. These include the near real-time RIC, O-CU, O-DU, O-RU, and the open evolved NodeB (O-eNB). The management and orchestration functions are received by the managed elements via the O1 interface. The SMO in turn receives data from the managed elements via the O1 interface for AI model training.

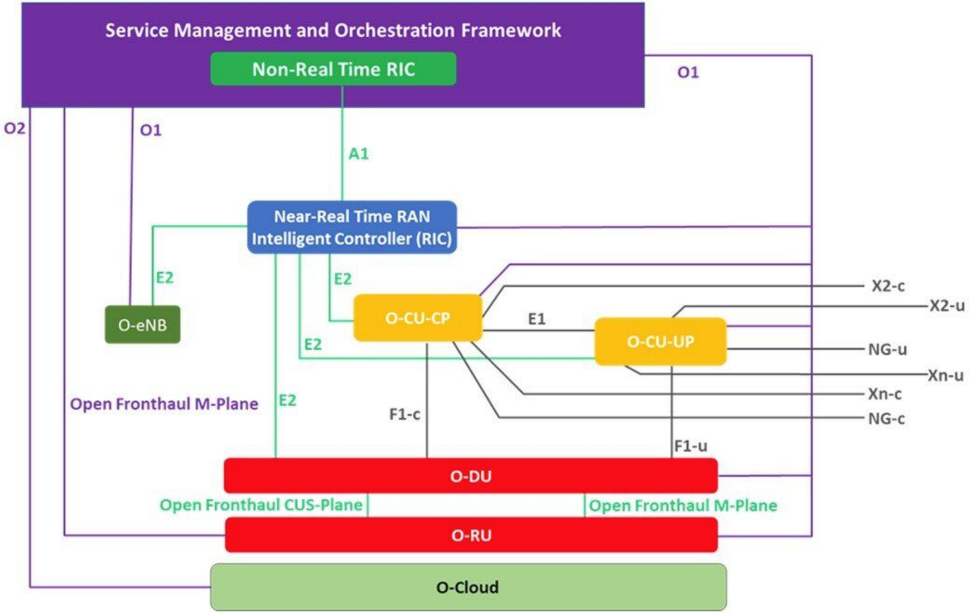


O1 Interface

MnS

- Fault and Heartbeat management
- Performance assurance and KPI reports
- Startup/configure network components
- Software and file management

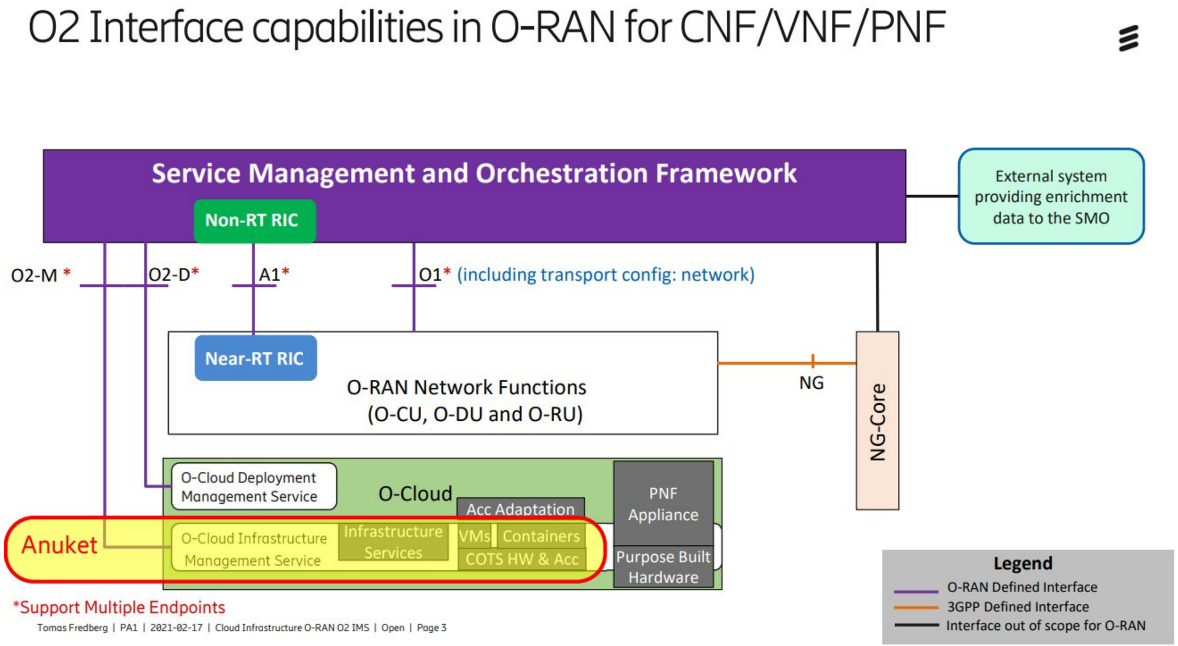
Fig. 7: O-RAN O1 interface and Management Services.



https://arxiv.org/pdf/2202.01032.pdf

ORAN- O2 Interface

The O2 interface is how the SMO communicates with the O-Cloud it resides in. Network operators that are connected to the O-Cloud can then operate and maintain the network with the O1 or O2 interfaces by reconfiguring network elements, updating the system, or upgrading the system.



O2 Interface capabilities in O-RAN for CNF/VNF/PNF

Service Management and Orchestration Framework
Non-RT RIC
External system providing enrichment data to the SMO
O2-M *
O2-D*
A1*
O1* (including transport config: network)
Near-RT RIC
O-RAN Network Functions (O-CU, O-DU and O-RU)
NG
NG-Core
O-Cloud Deployment Management Service
O-Cloud
Acc Adaptation
PNF Appliance
Anuket
O-Cloud Infrastructure Management Service
Infrastructure Services
VMs
Containers
COTS HW & Acc
Purpose Built Hardware

Legend
- O-RAN Defined Interface
- 3GPP Defined Interface
- Interface out of scope for O-RAN

*Support Multiple Endpoints

Tomas Fredberg | PA1 | 2021-02-17 | Cloud Infrastructure O-RAN O2 IMS | Open | Page 3

https://wiki.anuket.io/download/attachments/4391167/ORAN_O2-IMS-alignment.pdf?version=2&modificationDate=1613597403728&api=v2

ANUKET

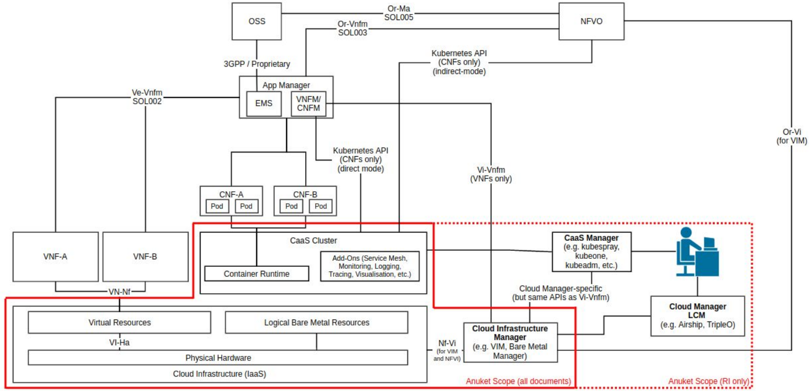
- OPNFV and CNTT merge to form Anuket, enable rapid deployment of cloud native and virtual network functions (CNFs & VNFs), infrastructure, and services

- Anuket unifies global ecosystem of telecom operators, vendors, and systems integrators with over 100 organizations participating to date

- New project facilitates seamless lifecycle management for integrating requirements, architecture, specifications, implementation, testing and deployment of Telecom Network Infrastructure

The Anuket project is focussed on:

- Functional capabilities of the cloud infrastructure and the infrastructure management
- Functional interfaces between infrastructure and infrastructure management
- Functional interfaces between workloads and workload management



Functional Scope of Anuket specifications

- ❑ RA-1: This CNTT Reference Architecture (RA-1) provdies an OpenStack distribution agnostic reference architecture that includes the Network Function Virtualisation Infrastructure (NFVI) and Virtual Infrastructure Manager (VIM).
- ❑ RA-2: This Reference Architecture for Kubernetes describes the high level system components and their interactions, taking the goals and requirements of the Reference Model and mapping them to real-world Kubernetes (and related) components.