LF NETWORKING
Developer & Testing Forum

ONAP Architecture Evolution

ONAP Streamlining – The Process

https://wiki.onap.org/display/DW/ONAP+Streamlining+-+The+Process

November 2023

Presenter: Byung-Woo Jun (Ericsson) – ARCCOM Chair, TSC, SECCOM

https://lfnetworking.org

ONAP Benefits to Industry

LFN Developer & Testing Forum

- Previously, ONAP as a platform had shown e2e network automation to the industry.
- Operators, vendors and enterprises have learned how service/network automation (modeling, orchestration, policy-based closed loop, optimization…) works on VM and Cloud-Native environments for VNF, PNF, CNF, NS, Network/RAN Slicing and e2e service thru ONAP.
- In ONAP, there are numerous valuable use cases, that leverage and coordinate clusters of ONAP component functions (e.g., SDC, SO, A&AI, DCAE, Policy, SDNC, SDNR, CPS, CDS…) to achieve objectives, such as:
  - E2E Service
  - Network Slicing
  - RAN slicing
  - Closed Loop
  - ETSI-based NS & VNF orchestration
  - Helm-based CNF orchestration
  - ASD-based (including Helm) CNF orchestration
  - …
- Now, the operators, vendors and enterprises want to select and apply ONAP functions to their portfolio. No one needs to take ONAP as a whole.

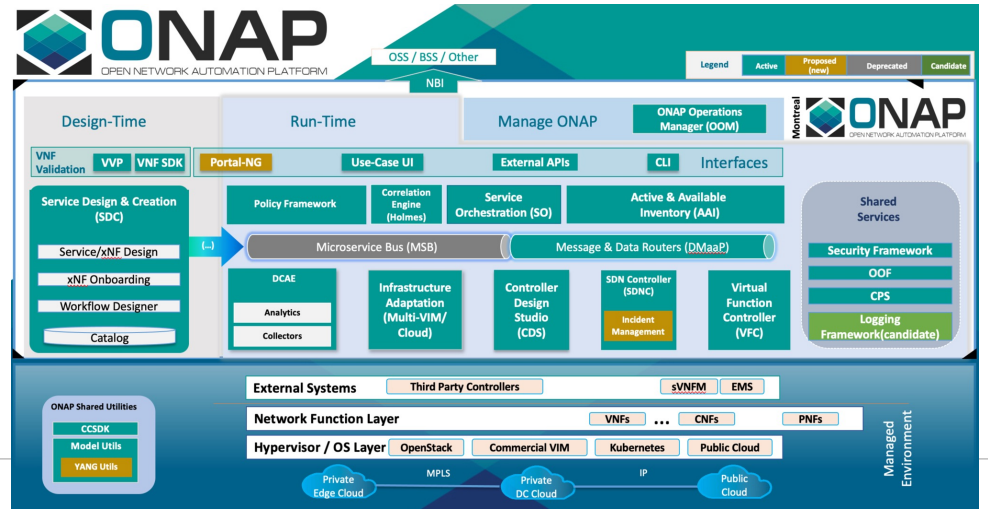
- Our goal is to continue to support those use cases efficiently for use in commercial production environments and portfolios.
- We expect the industry wants to pick and choose desired ONAP component functions, swap some of the ONAP functions, and integrate those functions into their portfolios seamlessly, without bringing in a whole ONAP platform.
- ONAP streamlining, which drives individual components and clusters of components guided by use cases, will enable the flexible and dynamic function adoption by the industry.

- ONAP stakeholders are thinking about connecting ONAP, O-RAN, Nephio, EMCO, and other communities for larger objectives, by
  - Reuse of selected ONAP functions
  - Functional delegations
- ONAP streamlining evolution is started.

ONAP Streamlining – Transformation

LFN Developer & Testing Forum

- Thru ONAP Streamlining, **ONAP is no longer a platform**, rather it provides various network automation functions, and security reference configuration in LFN.
- ONAP enables individual ONAP function build, and component deployment thru CD.
- Build use cases for repository-based E2E service, NS, CNF and CNA onboarding, and CD-based ONAP component triggering mechanisms with abstracted interfaces for choreography.
- Standard-based abstracted interfaces with declarative APIs.
  - Each component will be autonomous and invoked from any level of Network Automation, by leveraging CD mechanisms – e.g., GITOps and CD readiness
- ONAP will become more intent-based and declarative, and bring in more AI, conforming to standards such as 3GPP, TMForum, ETSI, IETF, O-RAN, etc.
  - Extend UUI User Intent support and AI-based natural language translation, on top of that, applying coming 3GPP and TMForum models and APIs.
  - Delegate resource-level orchestration to functions from the external community .
- For Security, ONAP continues to support the Service Mesh, Ingress, OAuth2, IdAM-based authentication and authorization, and considers sidecar-less solutions for NF security.



ONAP focuses on Network Automation Functions

- Modular
- individual
- interface abstraction
- loose coupling
- Extensibility
- Interchangeability
- Autonomous
- Declarative
- CI / CD ONAP components and E2E Service, NS, CNF & CNA handling

ONAP Component Design Requirements 1/2

- **ONAP components should be designed not only for ONAP but also non-ONAP consumption.**
  - Instead of a component being graduated, an ONAP component becomes obsolete or unmaintained if ONAP does not have use cases for it.
  - Some ONAP component-specific features tend to be ignored if they are not used by other ONAP components.
  - ONAP component functions should be used by not only ONAP but also non-ONAP.
    - Component design should be generic and extensible in a way that would enable it to be used in non-ONAP
    - If components are more generally applicable, there is the potential to gain more traction.
- **ONAP component dependencies and couplings to other ONAP components should not be in an ONAP-specific way.**
  - Those dependencies and couplings could be both syntactic and semantic.
  - Numerous intra-ONAP component interfaces and communications are ONAP-specific.
  - Some limited APIs standardization efforts are in place, such ETSI MANO APIs, ASD, 3GPP...
- **Making each ONAP component should be ‘stand-alone’, so potential users can take a single component, without getting involved in the whole of ONAP.**
  - ONAP should support Pick-and-Choose.
  - In the future, each ONAP component will be autonomous, and its interface models and APIs will be more normalized.

ONAP Component Design Requirements 2/2

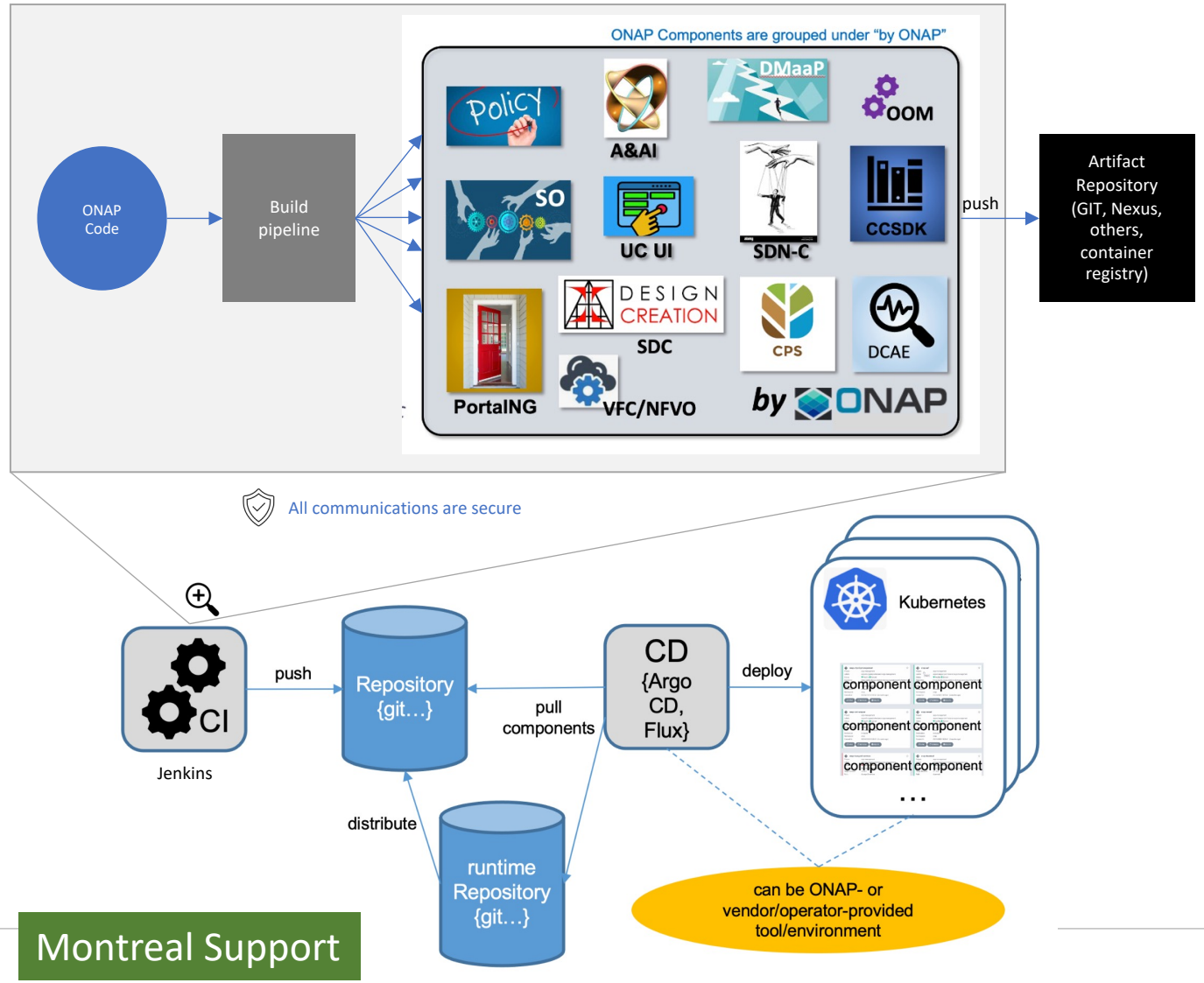
- **ONAP component interactions should be based on standards and extensible to facilitate integration with other systems, especially for non-ONAP.**
  - Aligning with standards where possible should be global requirements.
  - If there must be a deviation, that can be done in an extensible way that enables the standard-based approach.
- **ONAP component Helm charts in OOM should be re-written to build/deploy a component individually.**
  - CI build/integration of a vendor/operator could be less compatible with ONAP one.
  - OOM is not used by some vendors/operators.
  - In some cases, a vendor maintains a completely different set of Helm charts for ONAP components.
- **ONAP Security mechanisms should be industry standard/de facto-based to integrate with vendor/operator security and logging.**
  - The current security based on Service-Mesh, Ingress and Keycloak should be maintained.
  - ONAP components should not handle runtime security and logging collections directly.
- **Timelines and cadence of the ONAP release should be flexible for accommodating different release strategies.**
  - Should support creating a ‘Release’ in JIRA for the component releases.
  - Branching strategies are not aligned with ONAP CMO (Current Mode of Operation).
  - Resulting in an artificial split in functionality between releases.

ONAP Streamlining – Component Design, Build & Deployment

LFN Developer & Testing Forum

- ONAP Components are independently deployable pieces of software, built out of one or more microservices
  - Modular
  - Autonomous
  - Extensible and substitutional
- ONAP Network Automation processes will manage more intent-based operations using AI/ML.
  - Manage user and other Intents and translations
  - Study on TMForum & 3GPP Intent models and APIs
- ONAP components conform to the standards and de facto specifications to enable plug and play and pick-and-choose facilitation.

- ONAP repository-based SW management enables smaller imperative actions that can be triggered by different events in the orchestration and SW LCM flow.
  - Events can trigger different types of deployment automation jobs or chains of automation jobs (pipelines).
- In Jenkins, ONAP OOM build scripts will be used for ONAP component builds and will store built ONAP components into the Artifact Repository (e.g., Nexus). This can be changed.
- CD (e.g., ArgoCD, Flux, others) will be used to pick and choose ONAP components.

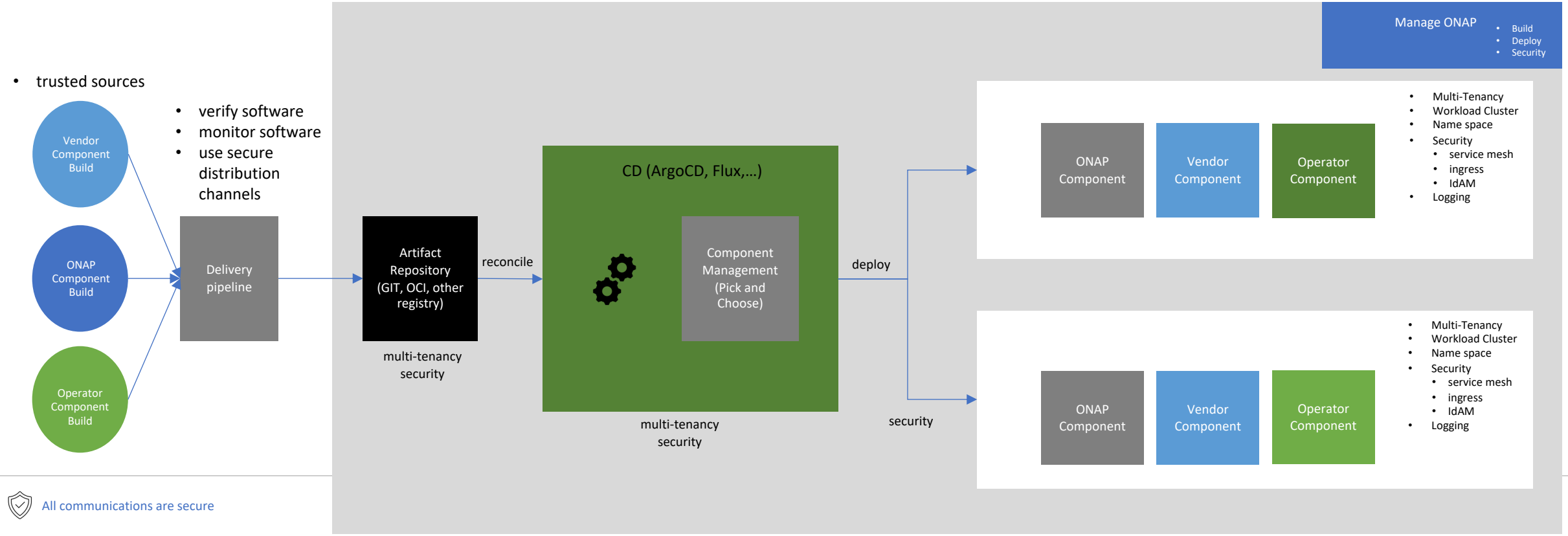


Montreal Support

Generic Repository-Based Component, Build & Deployment Target Architecture

LFN Developer & Testing Forum

- ONAP, Vendor and Operator CI/CD and Repository-based Component Build and Deployment.
- Selected ONAP components can be deployed along with other vendor / operator components.
- Target environment should support multi-tenancy, multi-workload cluster, multi-namespace.
- Secure Software Supply Chain CI/CD (automated framework) will be applied for component deployment from various sources securely
  - ONAP is no longer enclosed platform, so secure software supply chain CI/CD support is a must (see the SSSC CI/CD page).

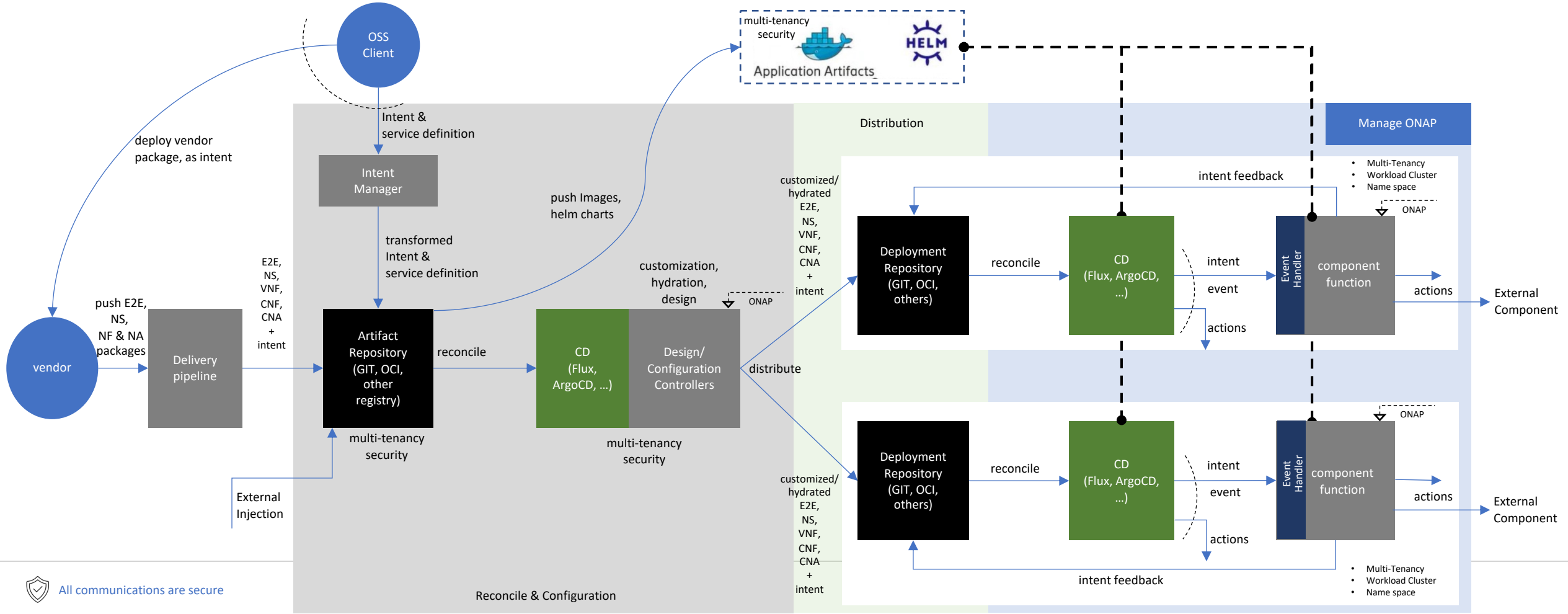


All communications are secure

ONAP Repository-based Software Package + LCM Use Case Study

LFN Developer & Testing Forum

- Package onboarding to ONAP thru the repository can also trigger SW LCM flows (deploying packages as intents).
- Applications, packages and intents are worked in the multi-tenancy, multi-workload cluster and multi-namespace runtime environment.

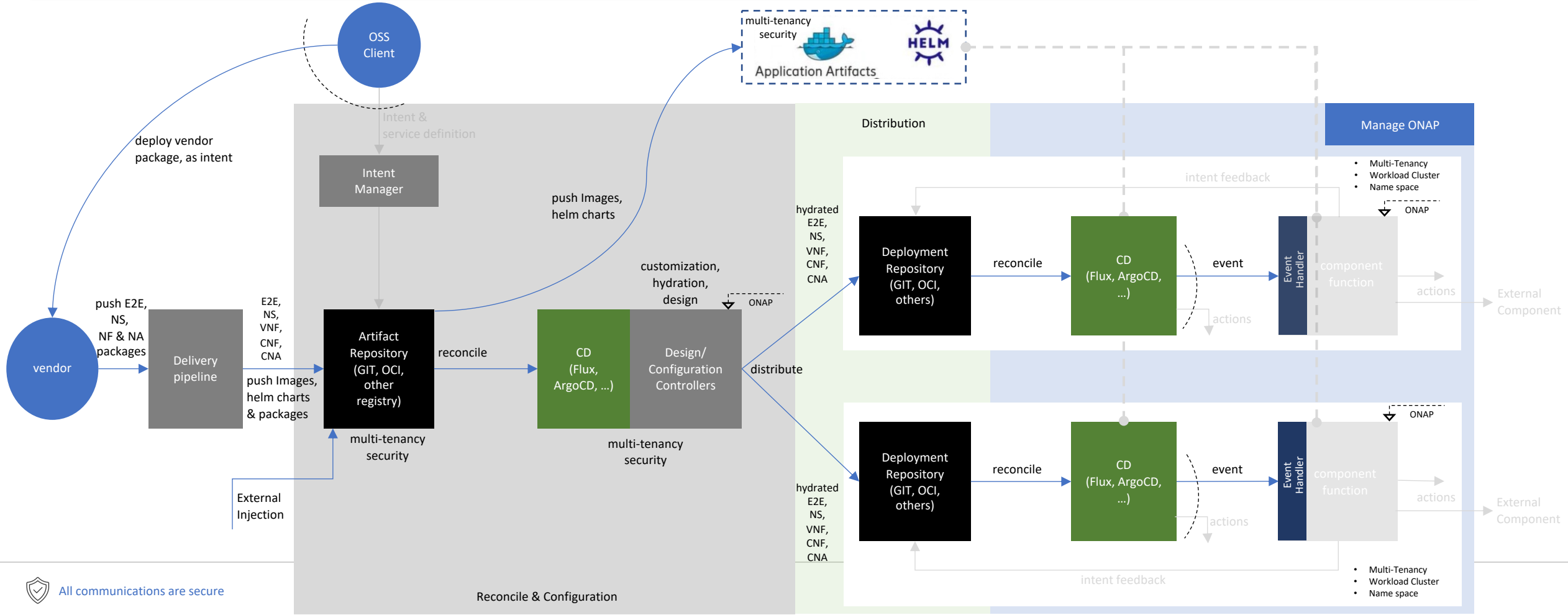


All communications are secure

ONAP Repository-based Software Package Onboarding Use Case Study

LFN Developer & Testing Forum

- ONAP repository-based SW packages will be onboarded by leveraging the delivery pipeline, Artifact repository and reconciliation.
- Packages will be customized/hydrated and delivered to target destinations.
- Package-related events will be generated, and telling packages are ready (current ONAP handles onboarding and LCM with two-step processes).

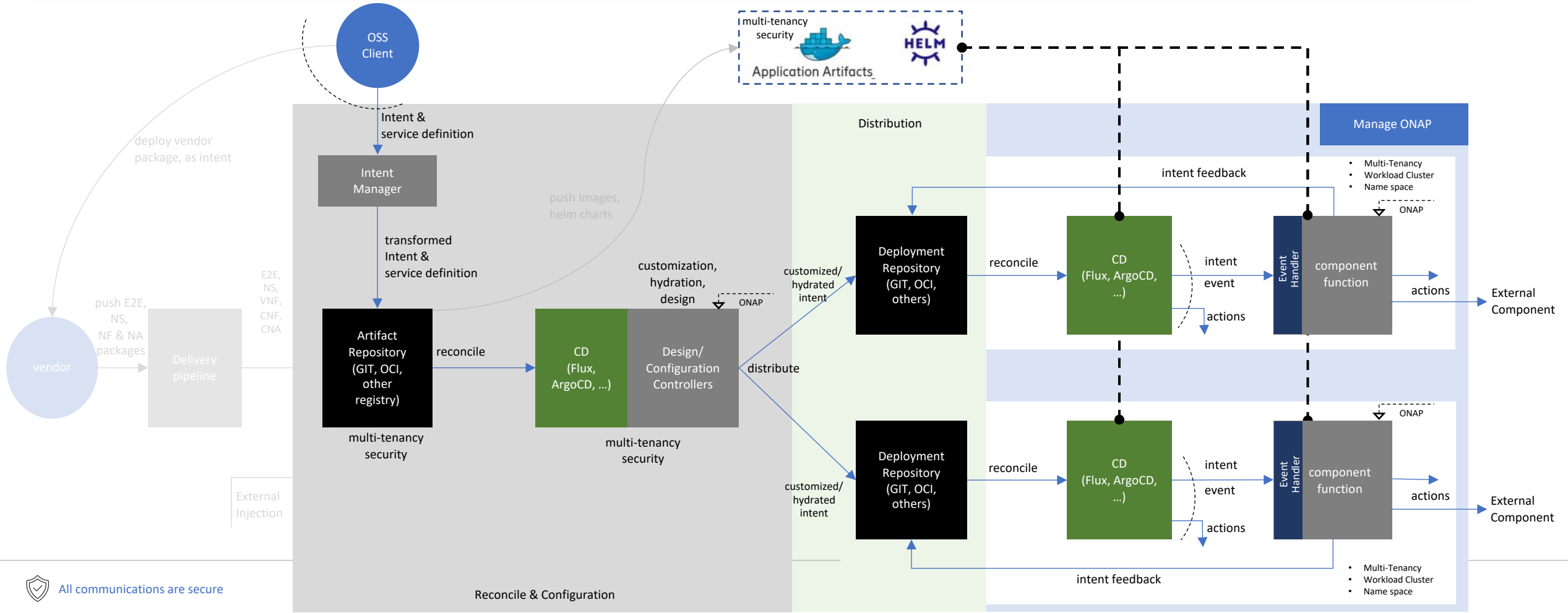


All communications are secure

ONAP Repository-based Software Lifecycle Management Use Case Study

LFN NETWORKING
LFN Developer & Testing Forum

- ONAP repository-based SW management enables smaller imperative actions that can be triggered by different events in the orchestration and SW LCM flow.
- When package onboarding is ready, OSS client can trigger software LCM processes by sending intents.

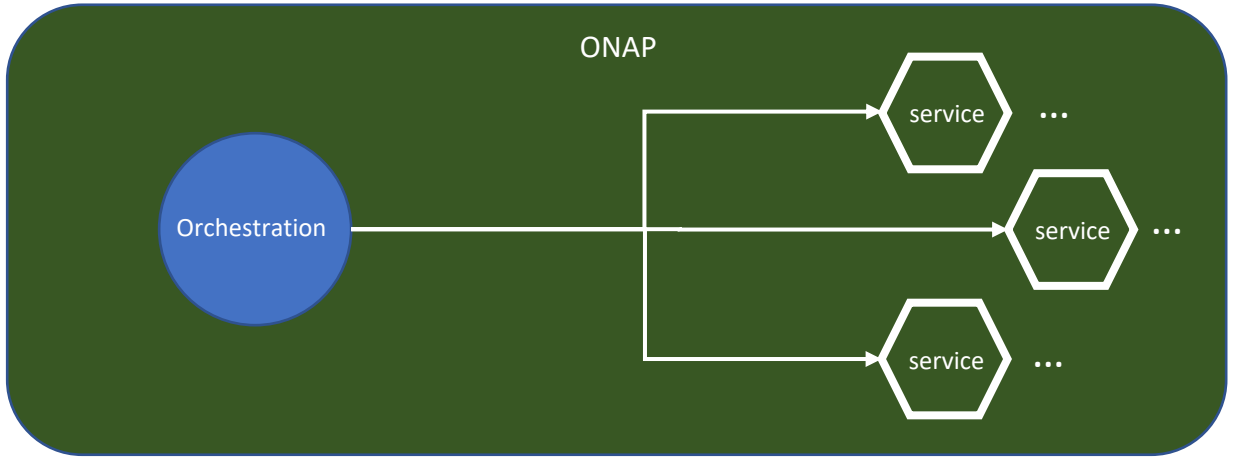


All communications are secure

ONAP Component Interaction Patterns

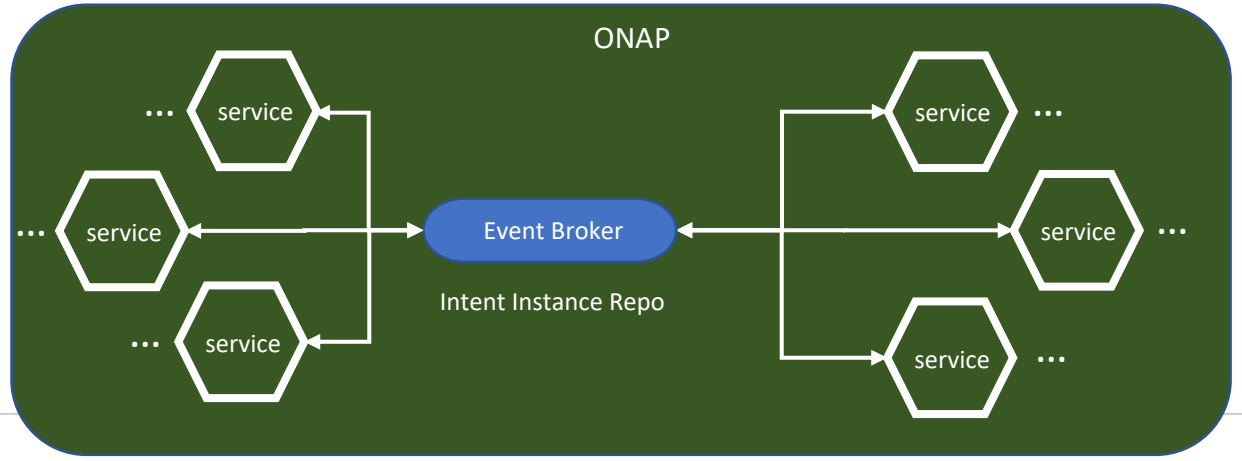
LFN Developer & Testing Forum

Orchestration Pattern



- Centralized control
- Tight coupling
- High dependency
- High-cost service substitution, addition, removal; may rewire portions of the communication path

Choreography Pattern



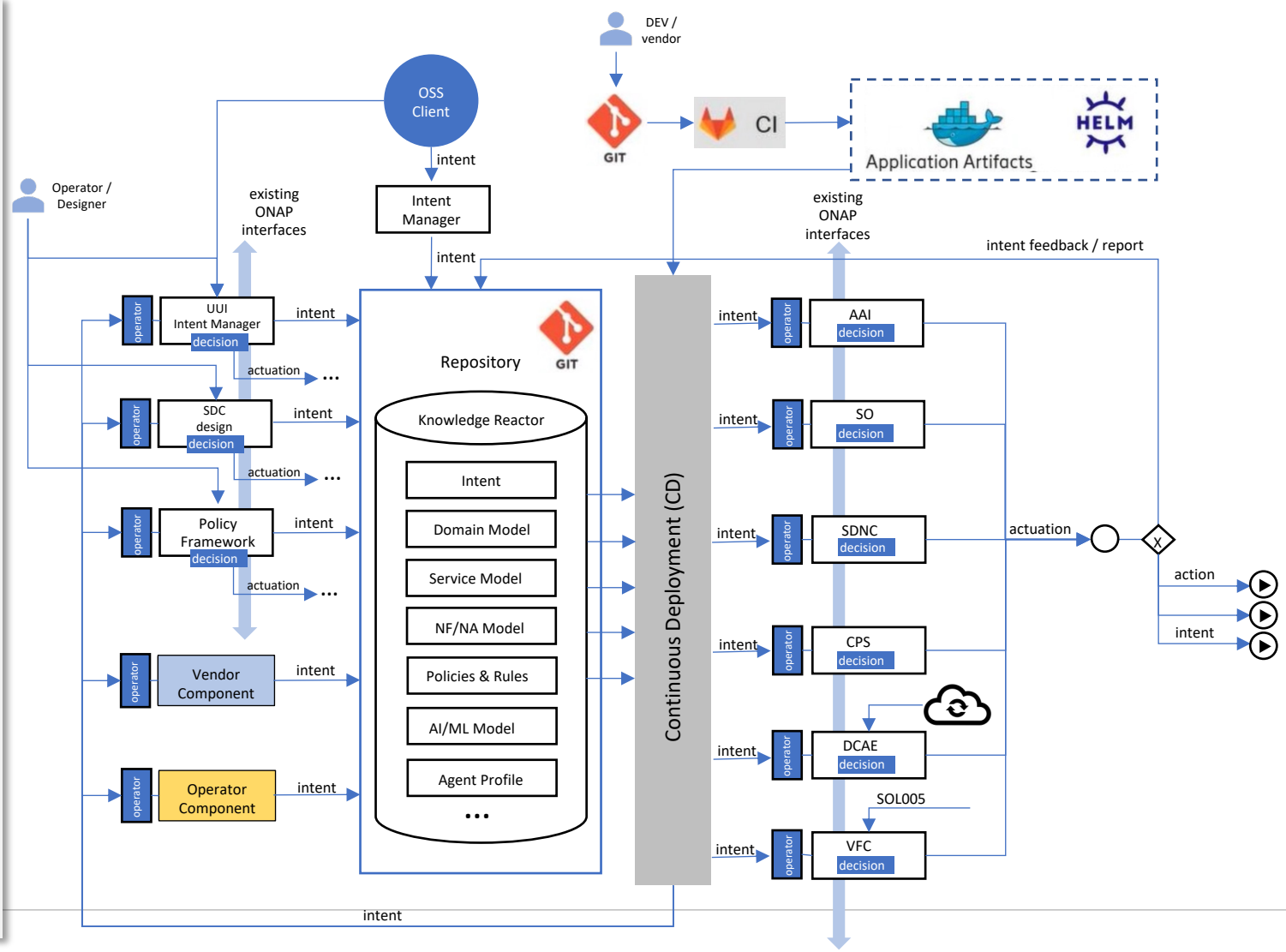
Combinations of choreography- and orchestration-pattern based interactions

- Distributed & autonomous operations
- Loose coupling
- Declarative & Intent-based handling
- Service independence
- Shorter chain of workflows
- Flexible service substitution, addition and removal

ONAP CI/CD Repository-based Network Automation Intent Use Cases – an Idea

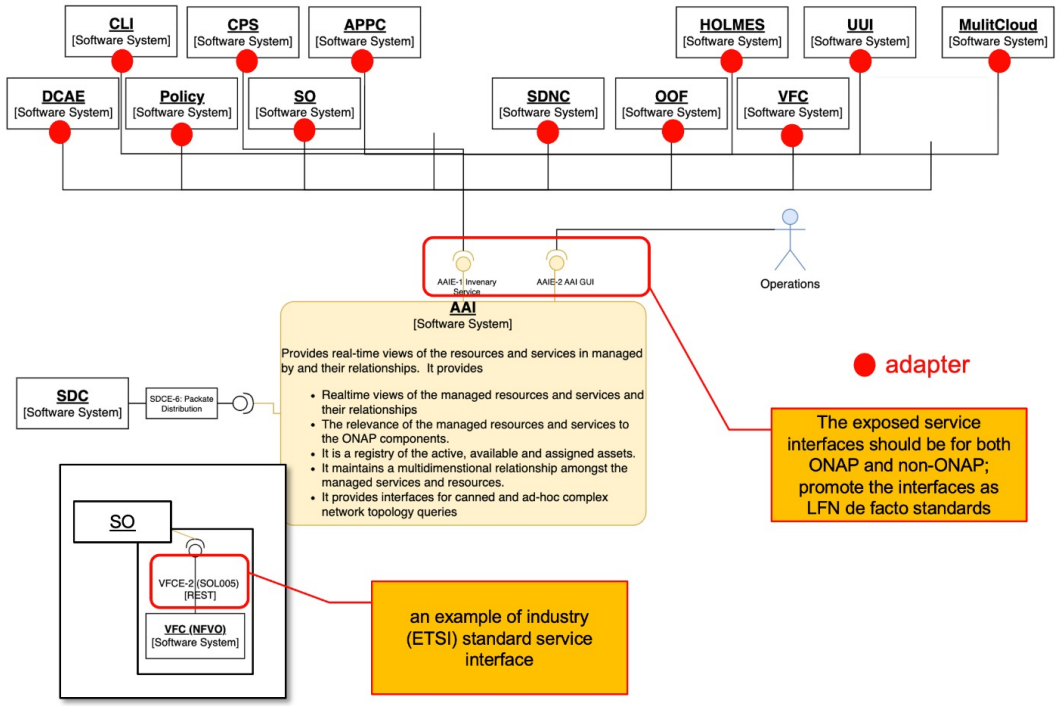
LFN Developer & Testing Forum

- **ONAP CI/CD and Repository mechanisms can be substituted by vendor/operator mechanisms.**
- Repository holds artifacts, intents and other models.
- ONAP components and vendor / operator components send and store their intents to the Repository.
- OSS Client sends their intents to the Repository thru Intent Manager, or UUI or else.
  - User-level Intents will be translated in UUI.
  - System-level Intents will be handled by Intent Manager.
- CD monitors the Repository and reconciles for events
- Thru CD, the corresponding intents will be delivered to the ONAP components thru the operators / event handlers.
- The operators will work as façade to trigger ONAP component functions (it would be an optional path, based on use cases).
- ONAP components do their business decisions and handles actuations.
- The actuations can result in actions and/or another-level of intents, which will be stored back to the Repository for next events. AI/ML can generate smart intents for next iterations.
- Images and/or Helm Charts and others will be stored thru the CI pipelines, which to be used by ONAP components, as needed.
- ONAP components can leverage the existing ONAP interfaces, based on their needs – a hybrid mode between declarative and imperative APIs will be supported.
  - ONAP ends up having smaller imperative actions that can be triggered by different events.



ONAP Component Interface Abstraction

- ONAP component interfaces should be designed/used for/by not only ONAP but also non-ONAP.
- ONAP component functions can be substituted and/or extended by vendors/operators.
- Component dependencies and couplings to other ONAP components should be removed.
  - Those dependencies and couplings could be both syntactic and semantic.
  - Intra-ONAP component interfaces and communications should not be ONAP-specific.
  - Aligning with standards where possible (e.g., ETSI NFV MANO, ASD, 3GPP SA5…) should be global requirements.
  - If there must be a deviation, that can be done in an extensible way that enables the standard-based approach.
- The exposed service interfaces should be for both ONAP and non-ONAP; promote ONAP component interfaces as LFN de facto standards.
  - If exposed service interfaces conform to industry standards (e.g., ETSI SOL005, ETSI SOL003, 3GPP SA5), the interactions between the service provider and consumer would be simplified (e.g., VFC case in this diagram).
- For now, the service consumers can use “adapters” which choose a desired service interface.
- **Action Points:**
  - **Promote ONAP Component API models and interfaces as open-source de facto APIs.**
  - **Event Handler / operator façade can be used trigger ONAP components as the previous slide.**



Plan for component interface abstraction and LFN-level de facto standardization

ONAP Component Interface Catalog Study

LF NETWORKING
LFN Developer & Testing Forum

- Expose the ONAP component service API models and interfaces, maybe as LFN-level de facto APIs for network automation, for example.
  - Create catalog of APIs for each ONAP project (Information Model, Data Model, OpenAPI).

SDC Service Provider Interfaces

- SDCE-1: VF Designer
- SDCE-2: Service Designer
- SDCE-3: DCAE Designer
- SDCE-4: Service Test
- SDCE-5: Service Test Dist
- SDCE-6: Artefact Distribution
- SDCE-7: Service Catalogue Retrieval

SO Service Provider Interfaces

- SO-E-01
- SO-E-02

AAI Service Provider Interfaces

- AAIE-1: Inventory Service
- AAIE-2: AAI GUI

Policy Service Provider Interfaces

- POE-1: Policy Type Design
- POE-2: Policy Design
- POE-3: Policy Management
- POE-4: Data Ingress
- POE-5: Decision Query
- CLPOE-1: Control Loop LCM, Policy LCM

DCAE Service Provider Interfaces

- DCAE-EXT1: VES Collector
- DCAE-EXT2: HV-VES Collector
- DCAE-EXT3: Data File Collector
- DCAE-EXT4: SNMPTrap
- DCAE-EXT5: RESTConf
- DCAE-EXT9: Data Extraction Service (DES)
- DCAE-EXT10: DCAE Openloop/CL Event
- DCAE-EXT11: PNF Registration Handler
- DCAE-EXT13: Slice Analysis MS
- DCAE-EXT14: PM Subscription Handler Service

CPS Service Provider Interfaces

- CPS-E-01: Provides remote clients with model LCM
- CPS-E-02: Generic data mutation interface
- CPS-E-03: Generic read/query interface
- CPS-E-04: Change notifications
- CPS-E-05: xNF data access
- CPS-E-06: Temporal data access
- CPS-E-07: Administration interface

OOF Service Provider Interfaces

- OOFE-1: Homing / Traffic Distribution
- OOFE-2: PCI/ANR Optimization
- OOFE-4: Route Optimization
- OOFE-5: OOF Model Administrator
- OOFE-6: Network Slicing

UUI Service Provider Interfaces

- UU-APIE-1: Operator Portal
- UU-APIE-2: Customer Portal

VFC Service Provider Interfaces

- VFCE-1: Portal / OSS Interface, based on ETSI SOL-005
- VFCE-2: Service Orchestrator / Policy Interface, based on ETSI SOL-005

SO NFVO Service Provider Interfaces

- SOL005: NS LCM interface

SO SOL003 Adapter Service Provider Interfaces

- SOL003: VNFM LCM interface

VNFSDK Service Provider Interfaces

- VNFSDKE-1: VNF Package Manager
- VNFSDKE-2: Market Place GUI
- VNFSDKE-3: Market Place
- VNFSDKE-4: VNF Test Platform

Multi-Cloud Service Provider Interfaces

- MCE-2: Resource LCM
- MCE-3: N/A
- MCE-4: Atomic Resource LCM
- MCE-5: Placement optimization
- MCE-6: Infra Provider Registry
- MCE-7: CNF LCM

Modeling Service Provider Interface

- etsicatalogAPIE-1: Catalog API
- etsicatalogAPIE-2: NSD Management API
- etsicatalogAPIE-3: VNF Management API
- etsicatalogAPIE-4: Parser API

Portal-NG Service Provider Interfaces

- TBD

DMaaP Service Provider Interface

- ~~DMaaP-1: DMaaP Bus Controller~~
- ~~DMaaP-2: DMaaP Message Router Source~~
- ~~DMaaP-3: DMaaP Message Router Consuming Interface~~
- DMaaP-4: DMaaP Data Routing Source
- DMaaP-5: DMaaP Data Routing Consumption Interface

Holmes Service Provider Interfaces

- HOLMESE-1: Rule Management
- HOLMESE-2: Health check

SDNC Service Provider Interface

- ORAN-Policy: A1 policy management updates
- CONE-1: Operations Interface
- CONE-3: Service Order Interface
- CONE-4: Policy Interface

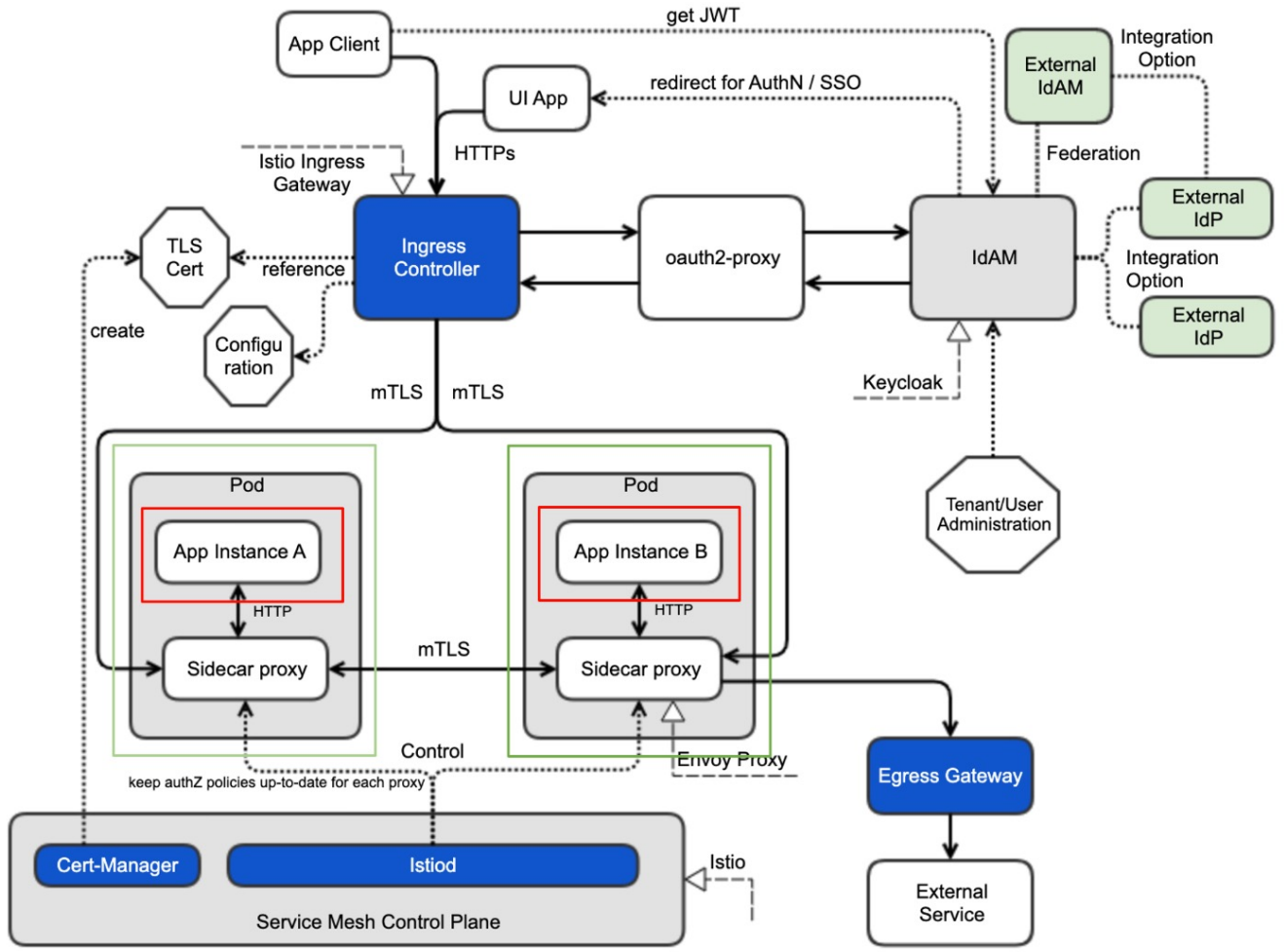
CDS Service Provider Interface

- CDSE-1: CDS interface for Blueprint

ONAP Runtime Security Architecture

LFN Developer & Testing Forum

- ONAP components are protected by Ingress Controller, Keycloak (IdAM) and Istio (Service Mesh), with AuthN/Authz, intra-secure communications, external-secure communications.
- ONAP components themselves do not have their own/ proprietary protection any longer (e.g., removal of HTTP Basic Authentication and HTTPs).
- Current OOM-provided security support as described above will be provided as ONAP reference security mechanism.
- It is assumed that vendors/operators support industry de facto security mechanism like ONAP security and imported ONAP components are protected by the security mechanism.
- ONAP will provide documentation of security architecture, global requirements and best practices, informing how to protect/secure selected ONAP components.
  - For secure external communications, Ingress Controller, aouth2-proxy and IdAM are used
  - For intra-secure communications, Istio is be used with Cert-Manager and policies
  - For user authentication and authorization, KeyCloak is used, with SSO support and OAuth2-based token generation and validation
- ONAP (OOM) provides security reference implementation and configuration by leveraging service mesh (Istio), ingress (Istio GA) and IdAM (Keycloak). The reference implementation and configuration can be replaced by the vendor/operator-provided security mechanism.



- ONAP provides security reference implementation. Vendors / operators can realize and configure with their own security system.
- Since vendor/operator component can be deployed with ONAP components, Secure Software Supply Chain is important.

ONAP SBOM & Secure Supply Chain

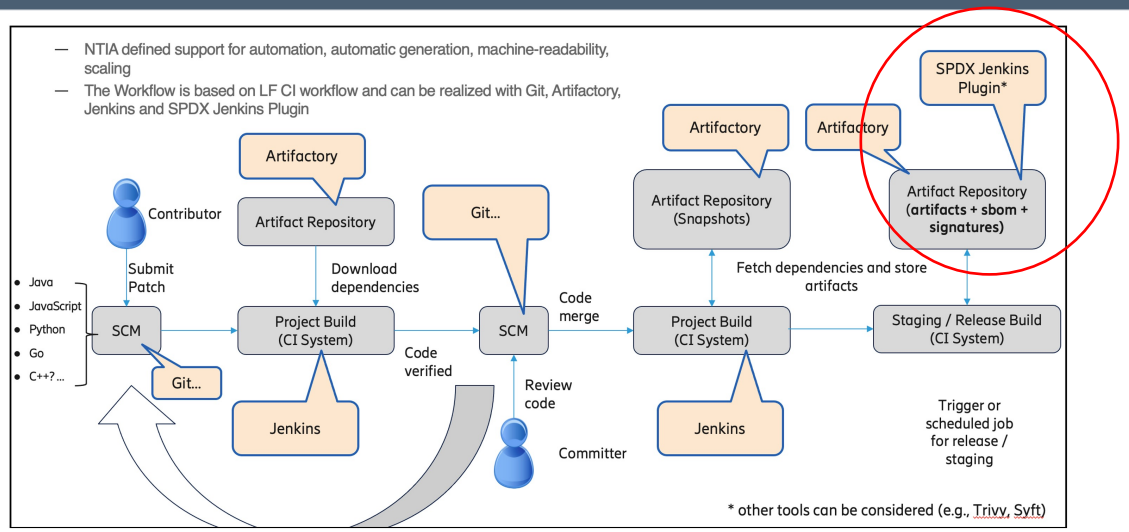
LFN Developer & Testing Forum

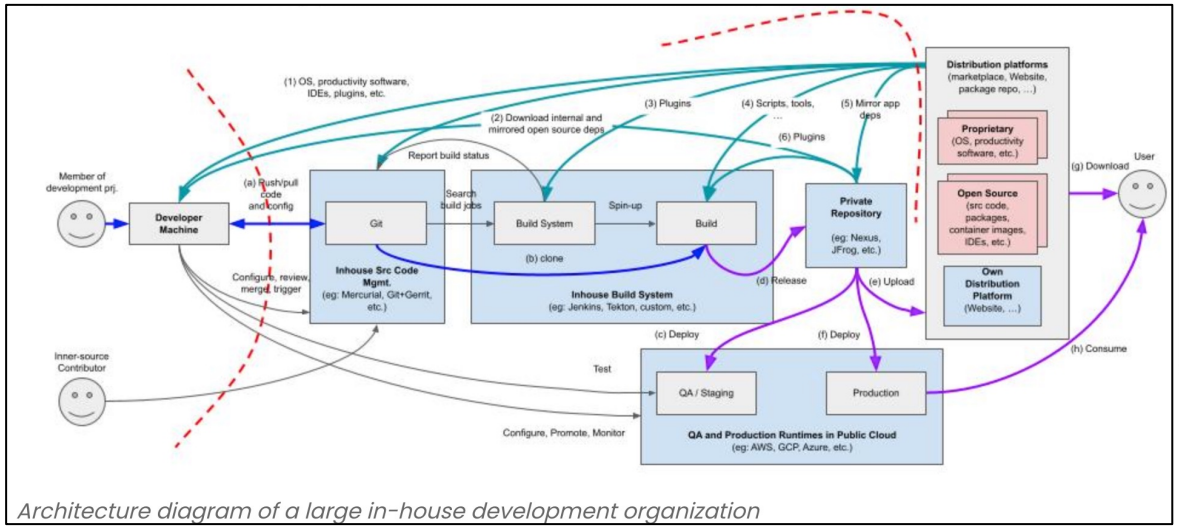
SBOM

- ONAP CI pipeline is based on Linux Foundation CI workflow.
- At the end, the CI pipeline generates signed SPDX-based SBOM files and stores them into a centralized repository.
  - ONAP supports SBOM by leveraging the spdx-sbom-generator, https://github.com/opensbom-generator/spdx-sbom-generator

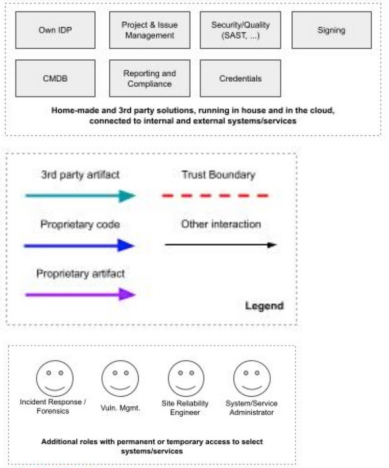
Secure Software Supply Chain

- Software supply chain is a highly targeted attack point for vulnerabilities and exploitability.
- If a software package is injected with malicious code, it may be much difficult to identify the issues at the later stage, by just looking at the generated SBOM.
- Since ONAP Streamlining allows ONAP components are mixed with vendor/operator-supplied components, the software supply chain must be protected and secured.
- With the secure supply chain, software and SBOM which are built, added and generated during the secure pipeline can be trusted.
- **With the secure software supply chain, every step in the software supply chain needs to be verified, validated and tamper-resistant, with provenance proof and trust of delivery.**





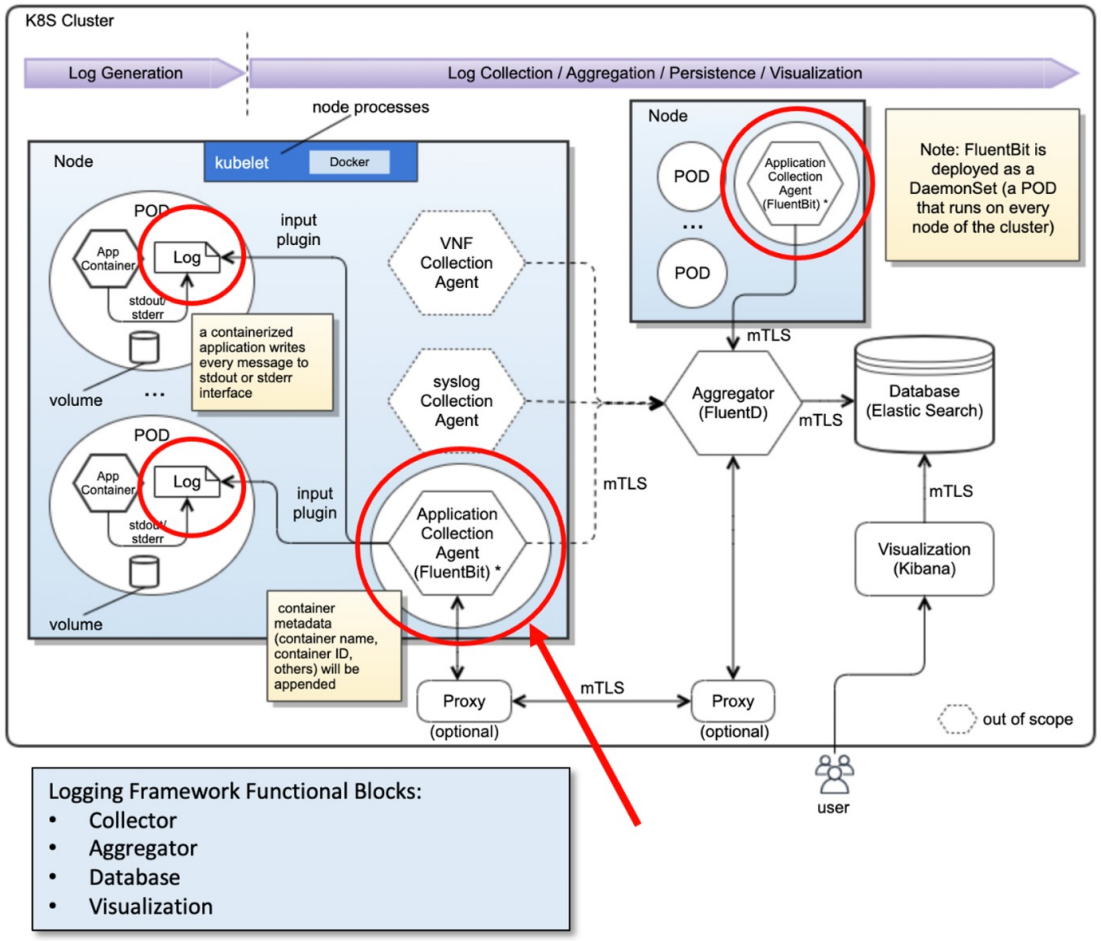
Architecture diagram of a large in-house development organization



https://openssf.org/blog/2023/09/27/threat-modeling-the-supply-chain-for-software-consumers/

ONAP Logging Architecture Readiness

- ONAP supports open-source and standard-based logging.
- ONAP already separates log generation from log collection / aggregation/persistence/visualization/analysis.
  - Each ONAP component handle log generation only thru STDOUT and STDERR, by following ONAP security logging fields – global requirements, https://wiki.onap.org/display/DW/Security+Logging+Fields+-+Global+Requirement
  - The log destination will be configured
  - Log collection agent(s) will be configured; ONAP reference configuration is using FluentBit as the collection agent;
    - ONAP uses a separate privileged namespace to deploy FluentBit for security reasons
    - Vendors/operators can configure it differently, based on their needs
  - Vendors/operators can realize and configure the log collection/ aggregation/persistence/visualization with their own logging ecosystem
- **This ONAP logging architecture is well aligned with the ONAP Streamlining. And, there will be no/minor impact on logging due to individual ONAP components.**



ONAP provides security reference implementation and configuration for logging. Vendors / operators can realize and configure with their own logging ecosystem.

ONAP Unmaintained Project Management

LFN Developer & Testing Forum

- Unmaintained ONAP projects can be archived; i.e., it will not part of Jenkins jobs or delivery.
- There is the ONAP unmaintained project management process: https://wiki.onap.org/display/DW/Project+State%3A+Unmaintained
- Projects that are removed from Montreal release:
  - MSB
  - CPS-Temporal

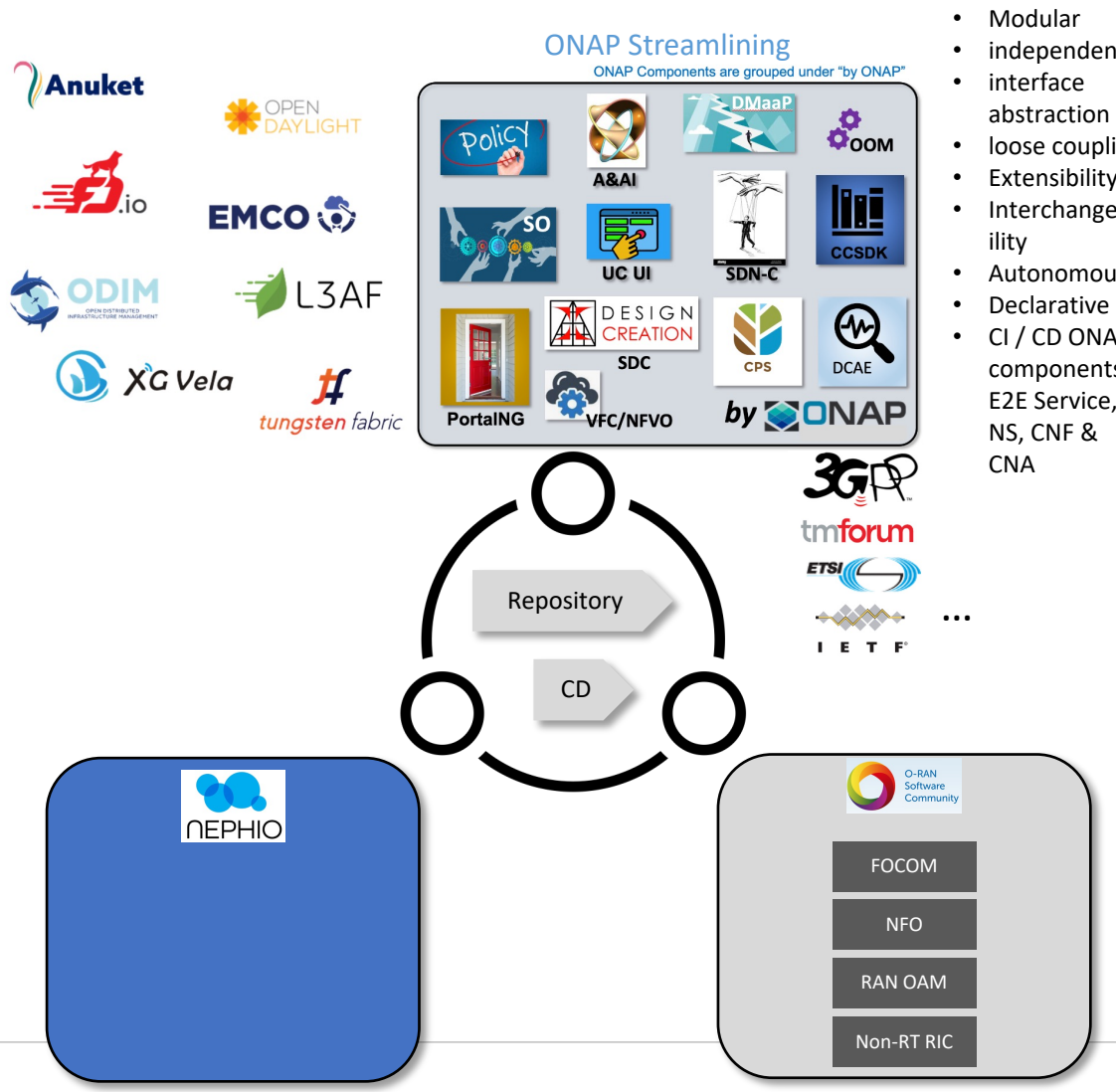
	Action	Responsible	Accountable	Consulted	Informed	Artifact	Tool/process
1	Final call to the ONAP Community raised by the PTL or by the TSC or TSC-Delegate. (Negative Result for volunteer for succession results in following steps).	PTL	TSC	PTLs	ArcSubComm SECCOM	JIRA EPIC(new) Ticket (highest priority/severity)	ONAP TSC JIRA Ticket (TSC on update CC)
2	Review what is used by the Community and the dependencies to other components and maintain the repositories that are necessary for the ONAP Components	ArcSubComm	TSC	PTLs, SECCOM	PTLs	JIRA EPIC-Tasks per project-New (Projects Dependency List)	ONAP JIRA, Nexus, PTL Assigned task updates, Weekly status on TSC Meeting
3	Verify what (if any) impact the change has on OOM/Integration (CIST)/DOC projects and ensure that is communicated	ArcSubComm	TSC	PTLs, SECCOM, DOC	PTLs	JIRA Task update, impact assessment report	ONAP JIRA, Weekly status on TSC Meeting
4	If the repo(s) are still required, then Identify an alternative path (if any)	ArcSubComm	TSC	PTLs	ArcSubComm, SECCOM	JIRA Task update,	Weekly status on TSC Meeting
5	Identify potential remaining committers to maintain the remaining repositories.	TSC	TSC	PTLs	ArcSubComm, SECCOM	Info.yaml to look up, JIRA	TSC delgation, ONAP JIRA, Gerrit Repo.
6	Make functionality retirement decision	TSC	TSC	ArchSubComm, SECCOM	Arc, Sec, PTLs	JIRA update LFN IT Jira issue for updating INFO.yaml	ONAP JIRA LF IT JIRA
7	Update INFO.yaml with **TBD (define fields, etc.)**	PTL, Contributor (if available), LF IT	TSC	None	ArcSubComm, SECCOM, PTLs	JIRA task	PTL, Committer, Super Committer, LF-IT, LF IT JIRA
8	In gerrit set the appropriate repositories that are no longer in use to 'Read Only' access	PTL (make request if available), LF-IT	PTL TSC (if no PTL)	None	ArcSubComm, SECCOM, PTL (if available) TSC	ONAP JIRA, LF IT JIRA	ONAP JIRA, LF IT JIRA, gerrit, github
9	Update the Architecture diagrams and references (Confluence)	ArchSubComm	TSC	None	Doc, ArcSubComm, SECCOM, PTL (if available)	ONAP JIRA, wiki	ONAP JIRA, confluence
10	Update the Architecture diagrams and references (readthedocs)	Doc	TSC	ArcSubComm	None	readthedocs	readthedocs
11	Remove Jenkins jobs (I think Code scanning and report generation needs to continue until closed and archived)	PTL (make request if available), LF-IT	TSC	None	PTL (if available)	JJBs	Jenkins
12	Move the project to Unmaintained State Projects including Clean-up of other wiki pages, RDT, JIRA, mailing lists, calendars, etc.	Project Management & its delegate	TSC	None	None	JIRA Tickets Updates	JIRA ONAP, JIRA LF-IT
13	Indicate in the release note that the project is in Unmaintained state i.e. add a hint in the header.	Documentation Team	TSC	None	None	User Docs, Release Notes	ONAP JIRA Updates, WIKI, ONAP.read the docs

...

ONAP – O-RAN SC – Nephio Collaboration

LFN Developer & Testing Forum

- ONAP sees Nephio is as a good showcase for declarative intent with continuous reconciliation
- ONAP has been looking for its delegation and integration points for Nephio
- Nephio could be a good candidate for Infra and CNF management delegation, but Nephio can be applied to every layer. Nephio is not only implementation but also architecture pattern / framework
- Nephio can be applied to every layer.
  - Some ONAP functions can apply Nephio pattern / framework
  - O-RAN SC SMO (FOCOM, NFO, OAM) can apply Nephio pattern / framework
- Nephio has the possibility to change existing orchestration APIs and layered network automation architecture
- Nephio needs to expose its capabilities to others via well-defined interaction mechanisms. The mechanisms should be declarative and intent-based with active reconciliation
- ONAP Streamlining supports pick-and-choose facilitation to others, e.g., O-RAN SC SMO
  - O-RAN SC SMO can reuse ONAP component functions base on their use cases



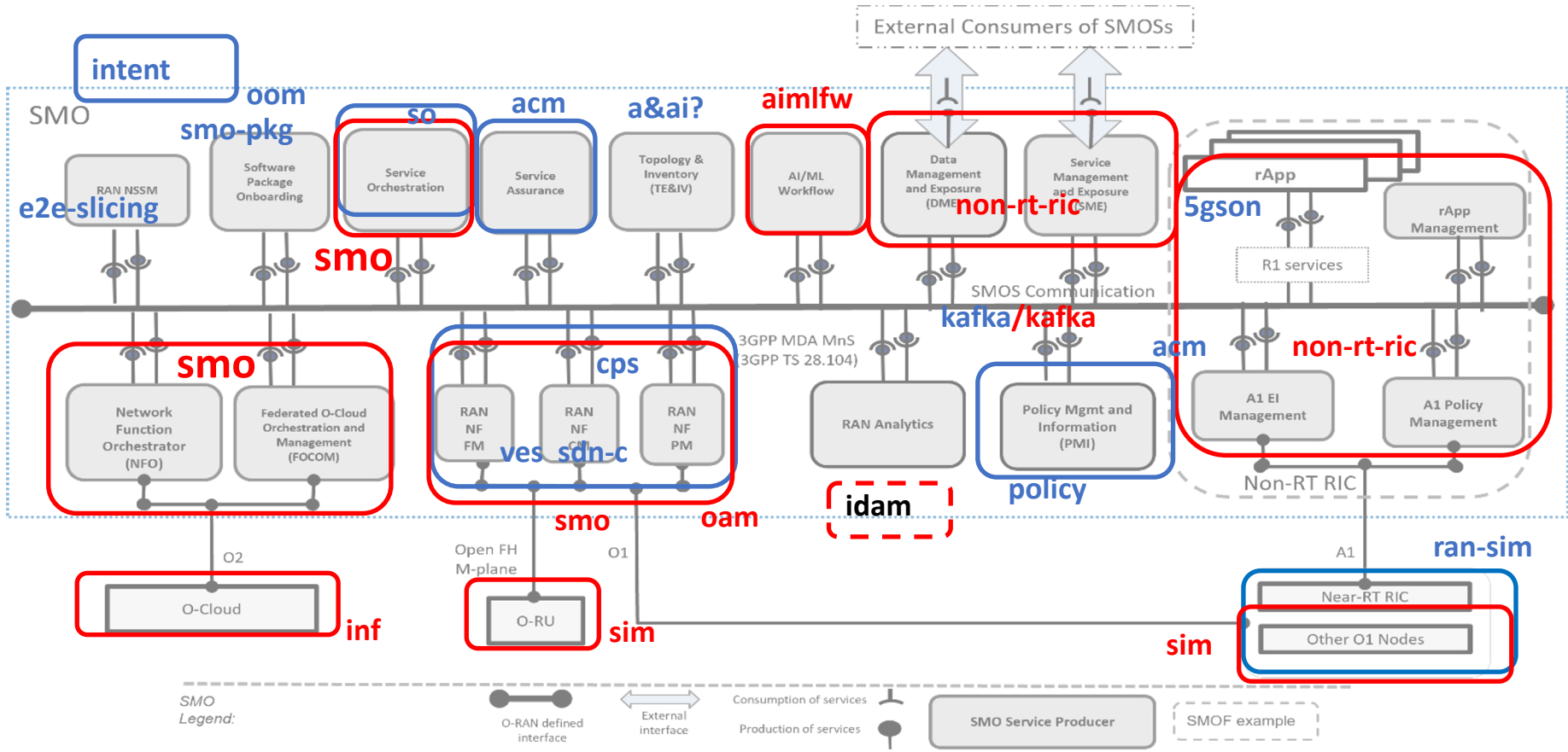
- Modular
- independent
- interface abstraction
- loose coupling
- Extensibility
- Interchangeability
- Autonomous
- Declarative
- CI / CD ONAP components + E2E Service, NS, CNF & CNA

ONAP and O-RAN Collaboration in Process

- Input from N.K. Shankar

LFN Developer & Testing Forum

- ONAP Streamlining provides pick-and-choose facilitation. It will help O-RAN SC choose ONAP functions as needed.
- The O-RAN SC and ONAP projects are related to SMO (expect changes)
- O-RAN SC SMO is considering reuse of ONAP functions.



Other relevant projects

int int

dcae? oof? sdc ?

security-framework

Notes:

- Figure is meant to explore alignment to O-RAN architecture
- Each red/blue label is an osc/onap open source project/component
- We may/do not have or need 1:1 mapping between projects and architecture blocks

Ref: https://wiki.o-ran-sc.org/download/attachments/78217260/dec_smo_flows_v1.4_20231109.pptx?api=v2 , and based on https://oranalliance.atlassian.net/wiki/download/attachments/2217476445/O-RAN-WG1.Decoupled-SMO-Architecture-TR-v01.00.14.zip?api=v2

ONAP and Nephio Collaboration

- Input from China Mobile Use Case

LFN Developer & Testing Forum

For Intent and AI-based analysis management and AI, China Mobile input:

- Nephio can play a significant role in the intent processing within the Network Layer and NE layer.
- Nephio could be a candidate for core network element delivery and assurance intent management.
- Nephio can utilize AI for intelligent diagnosis, analysis and optimization. Using AI, intent conflicts can be resolved.
- Conversion of Intent models between Nephio and other Intent systems.

Enhanced End-to-End Intent Processing via ONAP-Nephio Collaboration, see the https://wiki.lfnetworking.org/display/LN/2023-11+-+Nephio%3A+Enhanced+End-to-End+Intent+Processing+via+ONAP-Nephio+Collaboration session.

Requirements for Nephio:

- Nephio needs to define its NBI for other systems, such as ONAP and O-RAN SC SMO.

example

Intent Model from other system

Attribute	Content	Description
intentId	String	The identifier of this intent.
intentName	String	It describes the name of the intent.
	IntentExpectation	Multiple expectation lists contained in one intent.
intentContexts	Context	It describes the list of IntentContext(s) which represents the constraints and conditions that should apply for the entire intent.
intentFulfilmentInfo	FulfilmentInfo	It describes status of fulfilment of an intent and the related reasons for that status.

Nephio CRD

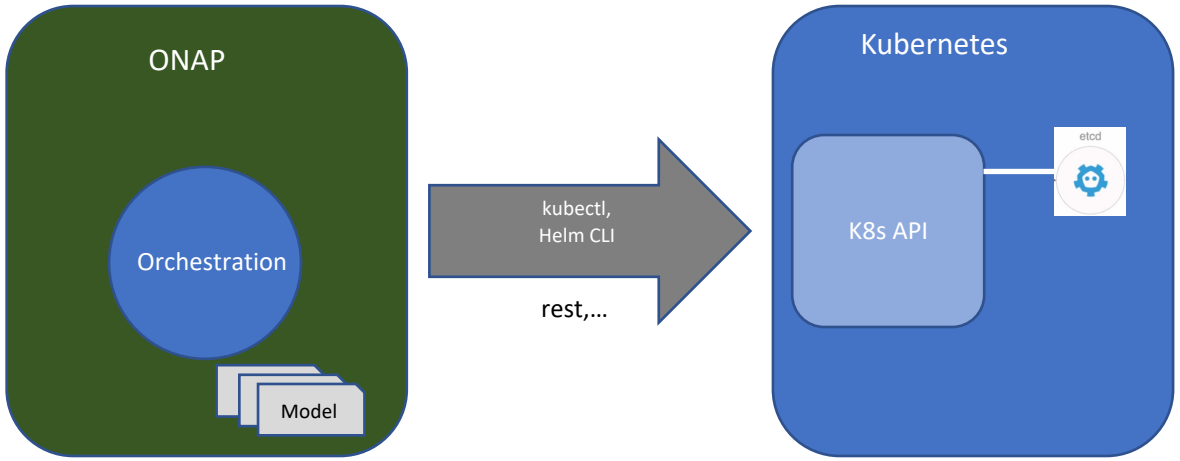
```
apiVersion: nf.nephio.org/v1alpha1
kind: FiveGCoreTopology
metadata:
  name: fivegcoretopology-sample
spec:
  upfs:
    - name: "agg-layer"
      selector:
        matchLabels:
          nephio.org/region: us-central1
          nephio.org/site-type: edge
      namespace: "upf"
      upf:
        upfClassName: "free5gc-upf"
        capacity:
          uplinkThroughput: "1G"
          downlinkThroughput: "10G"
        n3:
          - networkInstance: "sample-vpc"
            networkName: "sample-n3-net"
        n4:
          - networkInstance: "sample-vpc"
            networkName: "sample-n4-net"
        n6:
          - dnn: "internet"
            uePool:
              networkInstance: "sample-vpc"
              networkName: "ue-net"
              prefixSize: "16"
```

- Convert other intent model to the format of Nephio CRDs.
- Operation information of Nephio intent is parsed and added to other intent model.

for more details, see https://wiki.onap.org/display/Meetings/TSC+2023-10-12?preview=/190218511/193527857/Use%20Cases%20Proposal%20for%20Nephio-v2.pdf

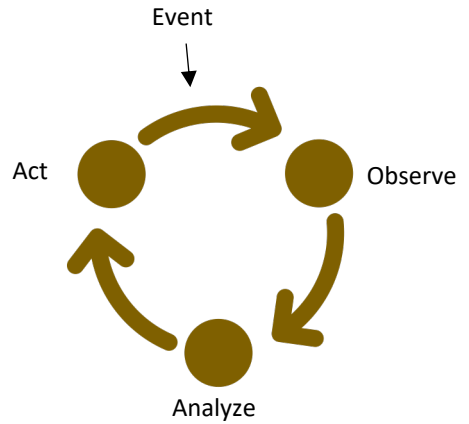
ONAP Southbound Interaction Study (Declarative and Reconciliation-based on)

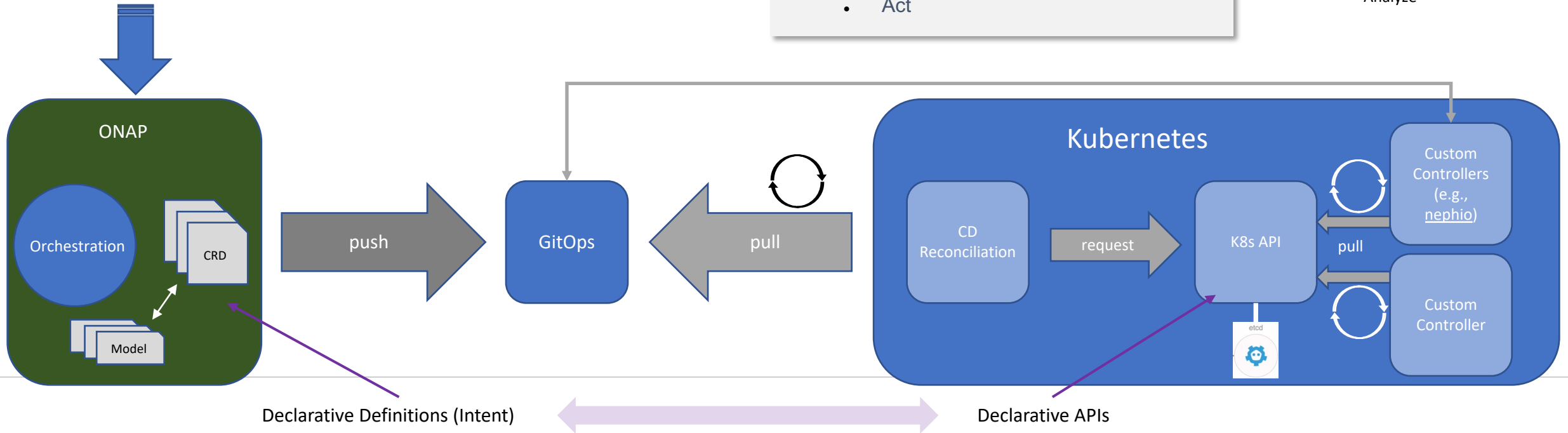
LFN Developer & Testing Forum



ONAP Southbound Interaction paradigm shift – a possibility

- Change from Imperative orchestration to Declarative choreography
- Define declarative CRDs as extensible K8s APIs and send Intents
- Use Kubernetes Operators
  - Observe
  - Analyze
  - Act





Q & A

LFN NETWORKING
LFN Developer & Testing Forum

Thank you !