

# MTCIL Demo

15th Oct 2020

Sandeep Karkala ([sandeep.karkala@mavenir.com](mailto:sandeep.karkala@mavenir.com))  
Sayan Sarkar ([sayan.sarkar@mavenir.com](mailto:sayan.sarkar@mavenir.com))  
Luis Zalce ([luis.zalce@mavenir.com](mailto:luis.zalce@mavenir.com))

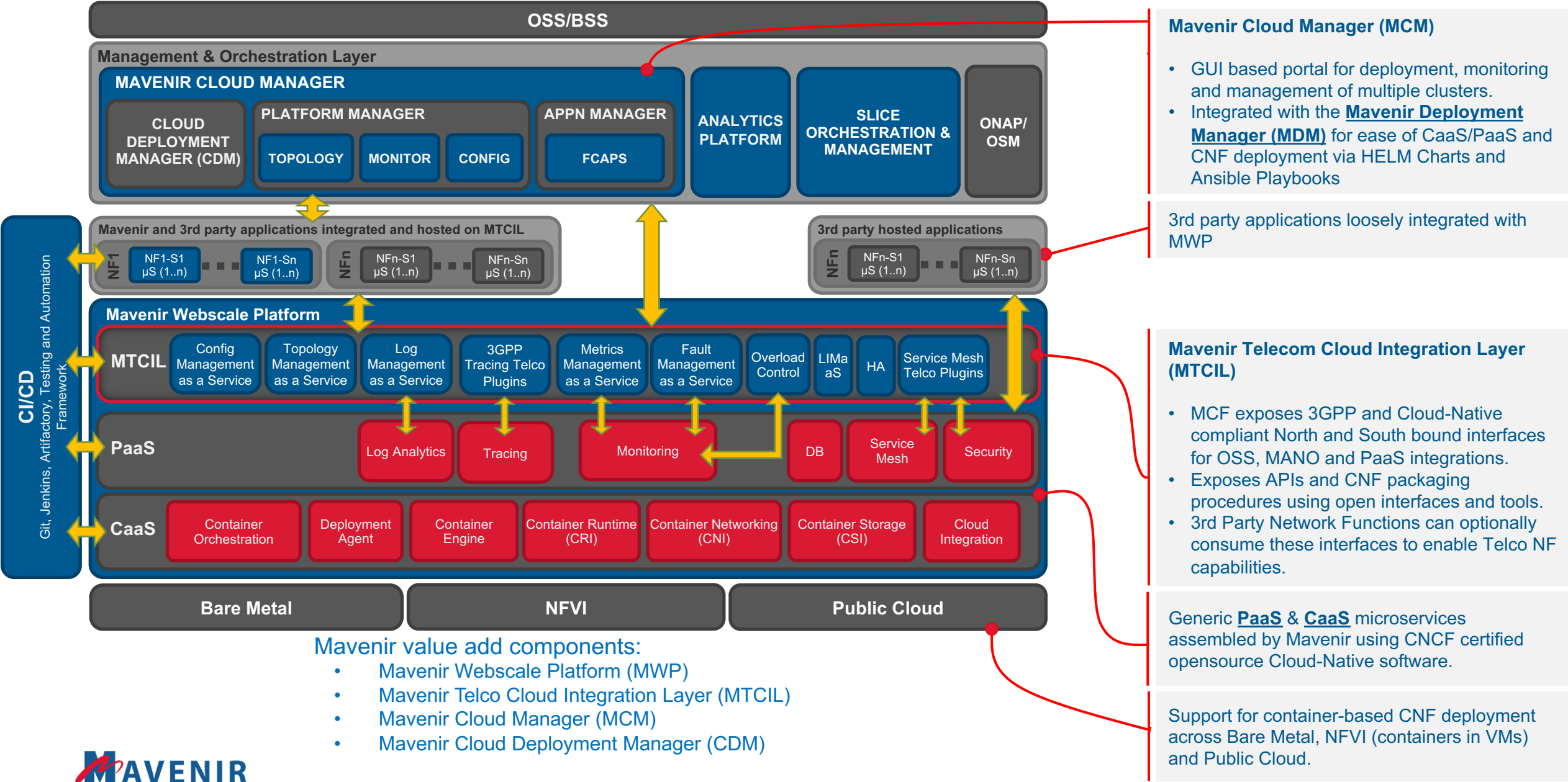
mavenir.com

# Overview

- Architecture Overview
  - Contribution to XGVela
- Design Overview
  - Management Model
  - Interfaces
  - Packaging
- Demo Use cases



# Architecture Overview

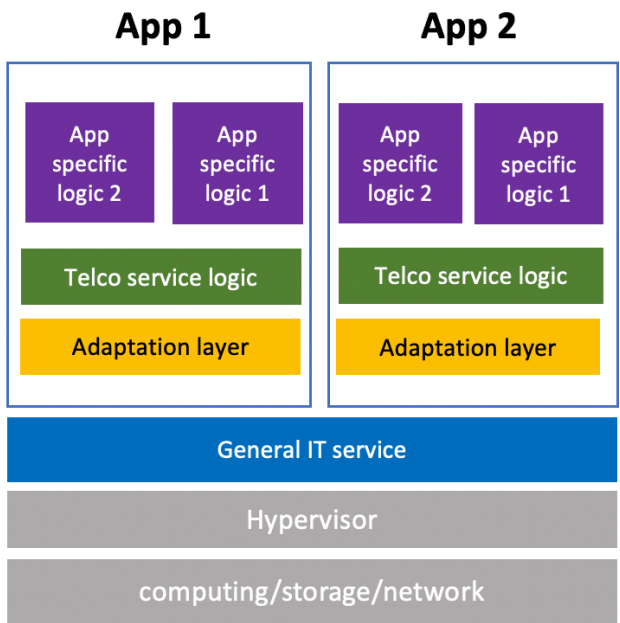


# Contribution to XGVela

## How does XGVela contribute to XG?

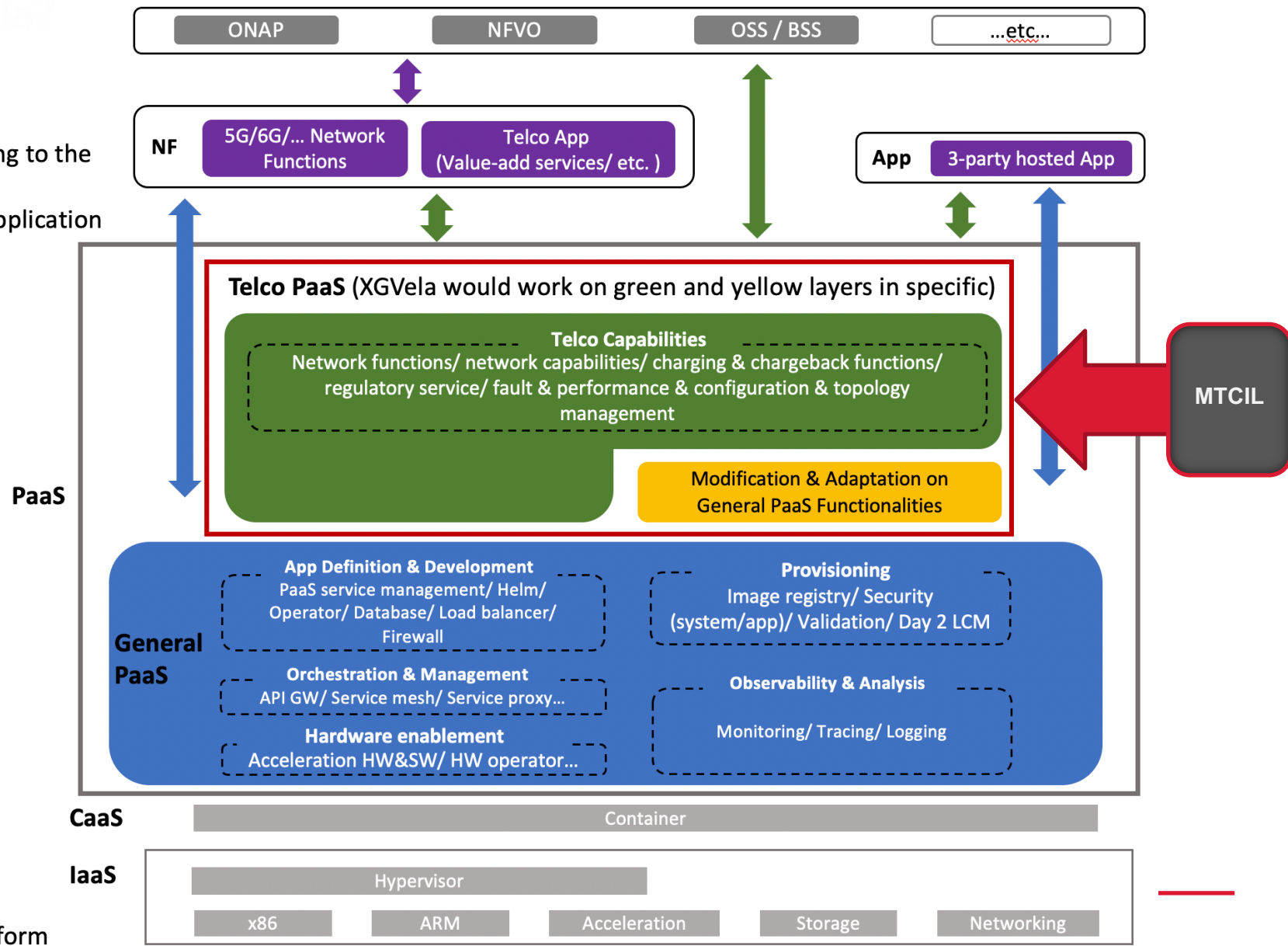
### 1. Application tailoring:

- The NFs / applications are further decomposed according to the microservices architecture
- Strip away the parts that have nothing to do with the application itself



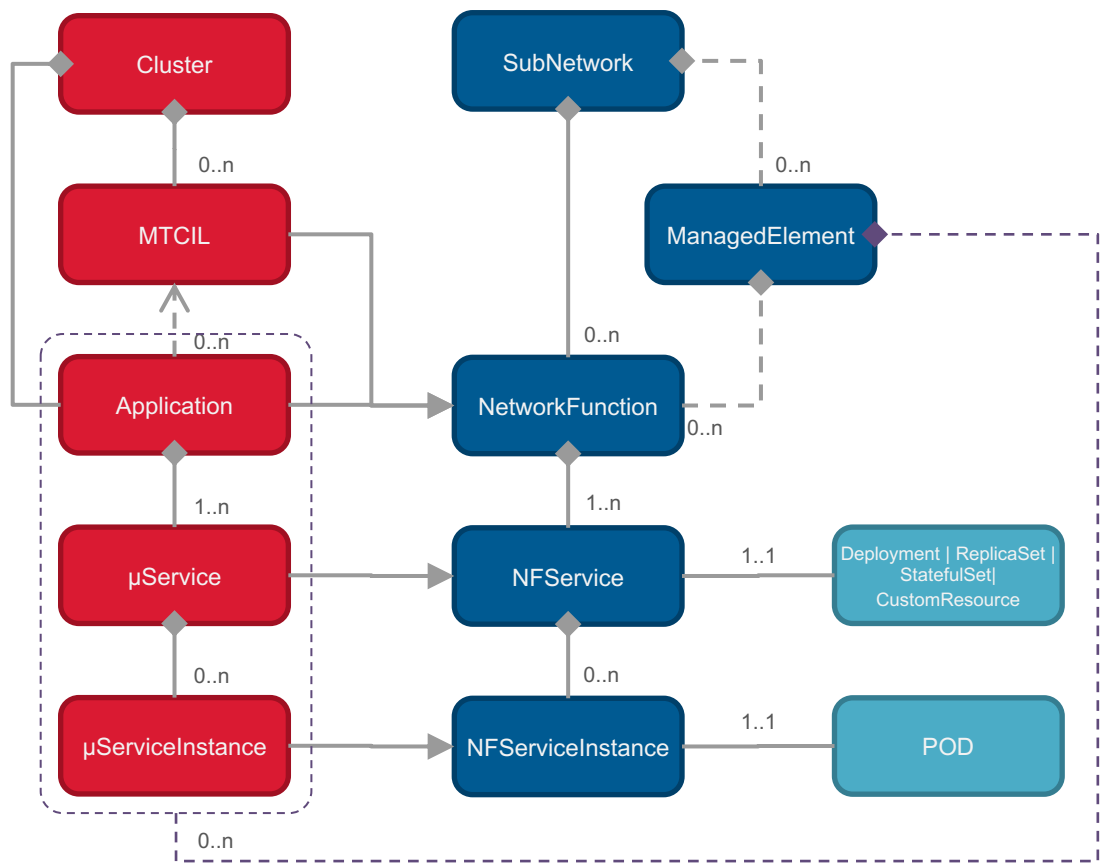
### 2. Platform addition:

- Support the coexistence of multiple resource forms
- Based on network element software architecture, the implementation of the general service rely on the platform
- Provides unified capabilities through API





# Design Overview – Management Model



Modelling and managing PaaS services as VNFs is one of the options discussed under NFV (NFV-IFA029)

## 7 Potential Architecture Enhancements

### 7.1 Architectural enhancement on PaaS related use cases

#### 7.1.1 Overall NFV Architecture

##### 7.1.1.1 General

Potential architectural enhancements depend on the design options selected to model PaaS services and thus PaaS services management. Three main design options can be envisioned:

- PaaS services are modelled as VNFs
- PaaS services are modelled as a new type of NFVI resources
- PaaS services are modelled as a new type of object specific to the PaaS layer

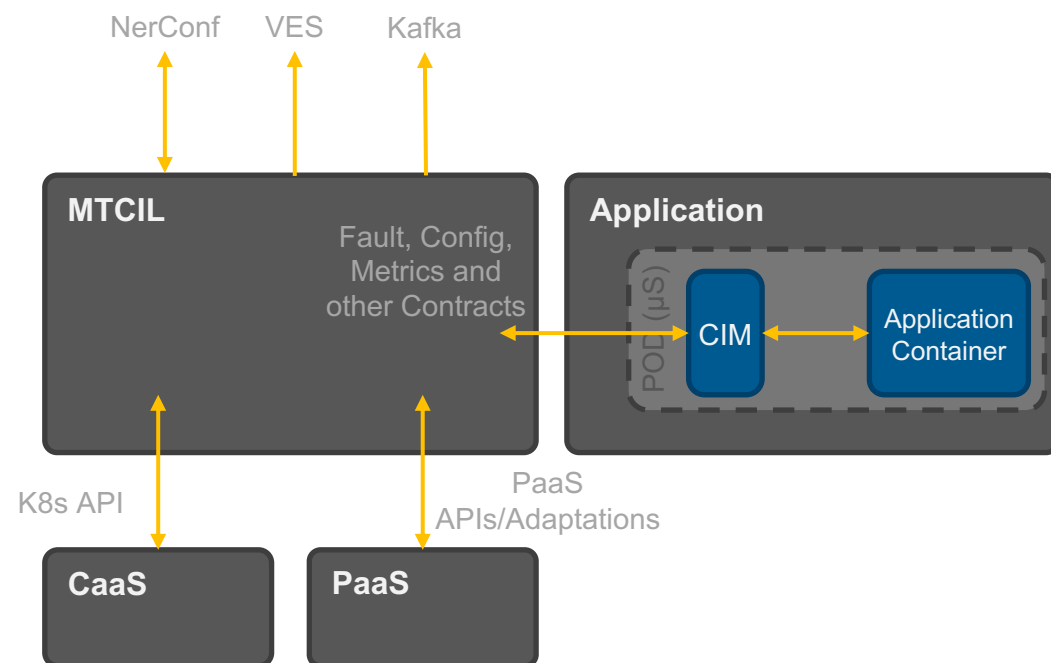
#### ETSI NF States,

- NULL
- INSTANTIATED\_NOT\_CONFIGURED
- INSTANTIATED\_CONFIGURED\_ACTIVE
- INSTANTIATED\_CONFIGURED\_INACTIVE

#### Correlated from k8s probes and resource events,

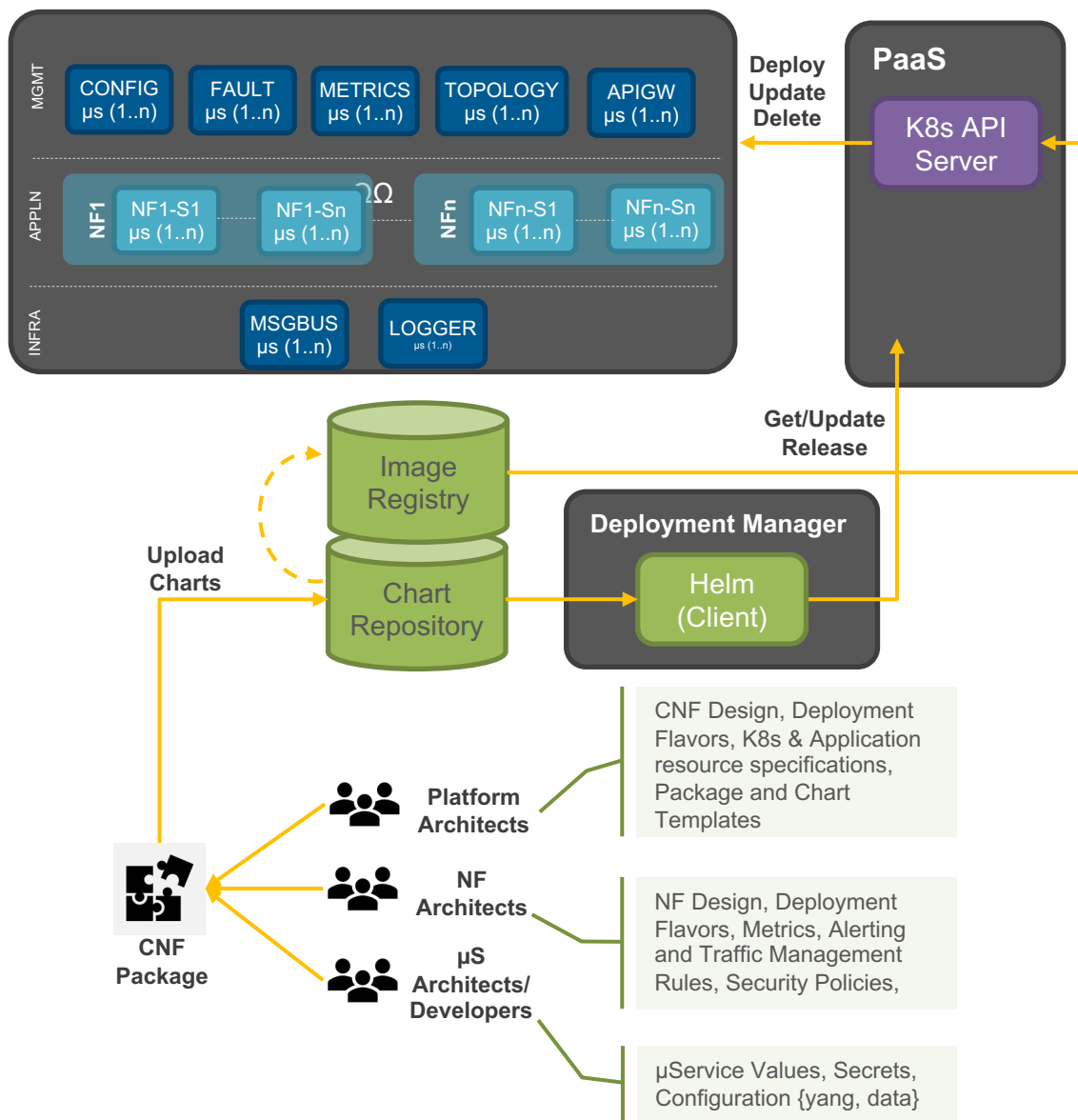
- Startup Probe
- Liveness Probe
- Readiness Probe
- ...

# Design Overview – Interfaces



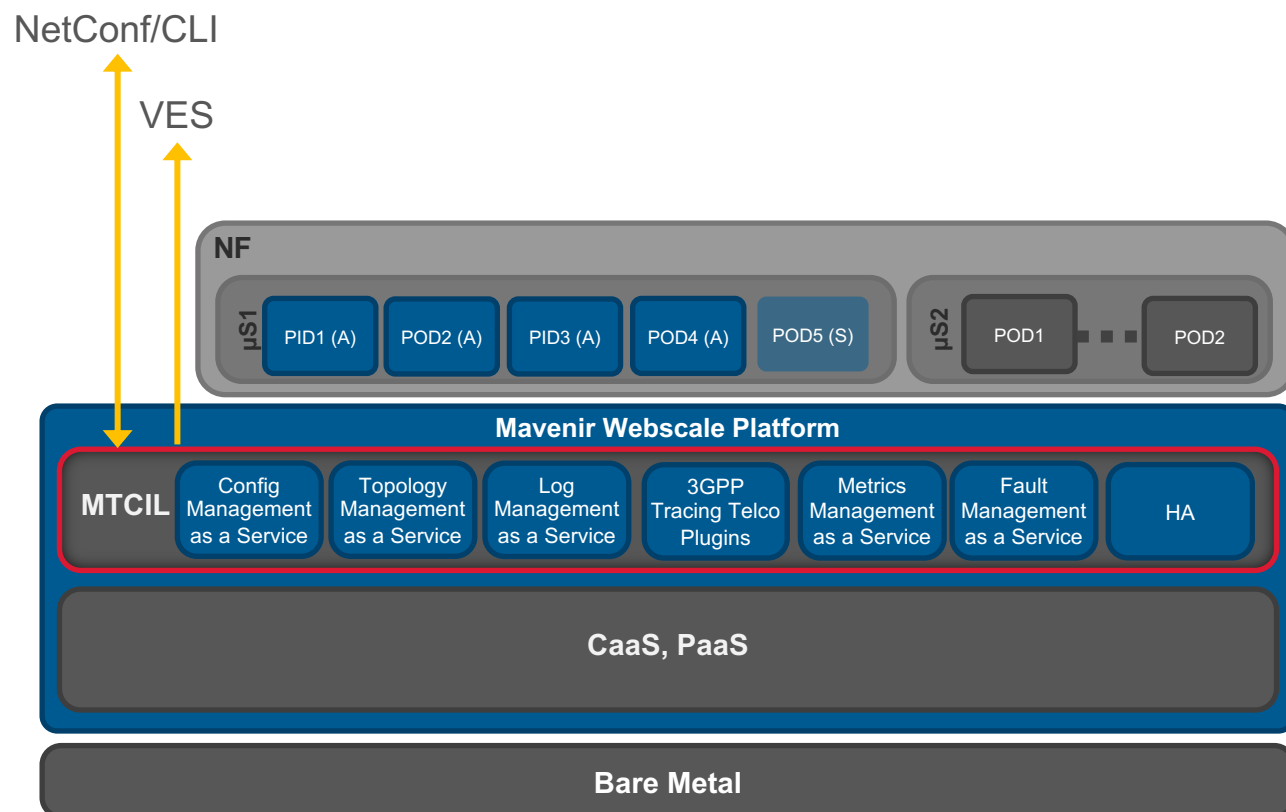
- **CIM** (CNF interface module, a component of MTCIL) provides single integration point API for Mavenir or 3rd party hosted applications
  - Deployed as a sidecar to application containers.
  - Implements various single node design patterns to enable loose coupling of application containers to the infrastructure.
  - Interfaces with application over REST for APIs and NATS for messaging and events.
- **MTCIL**
  - CMaaS exposes NetConf NBI for configuration and topology management. Interfaces with k8s for config discovery and push. JSON/REST interface to push config to application containers.
  - TMaaS interfaces with k8s for auto-discovery of services, builds 3GPP NF models. Exposes REST and also interfaces with CMaaS to expose Topology data over NetConf.
  - MMaaS is primarily Prometheus at the core.
  - FMaaS interfaces with application via CIM and with Prometheus for TCA. Enriches and exports events VES (version 7.1). Events can also be pushed via Kafka.
  - HAaaS provides a distributed HA model for fast switchover for stateful services. Interfaces with application via CIM.

# Design Overview - Packaging



- We take an opinionated (yet flexible) way of packaging CNF's as an essential measure to ensure,
  - Simplicity (by Separation of Concerns): Each team focuses on data and configuration most relevant to them.
    - μService architect/developer only needs to care about μService deployment values (only a subset), configuration schema, data and Secretes.
    - Network Function architect/Developer only needs to care about NF design, deployment flavors, various application specific rules and policies.
    - Platform Architect/Developer provides packaging framework and tools for generating necessary Helm charts, templates and Kubernetes resource specifications, annotation and label injections necessary.
  - Consistency
    - CNF deployment specification and packaging is abstracted and managed through a consistent set of opinionated procedures and design.
    - Packaging framework is not NF specific
    - All services are consistently configured
- CNF Packaging consists of two parts,
  - Package Builder
  - Deployment Manager (It also takes care of C/PaaS deployment along with CNF and is discussed in detail in the next section).
  - Can be encapsulated in a CSAR

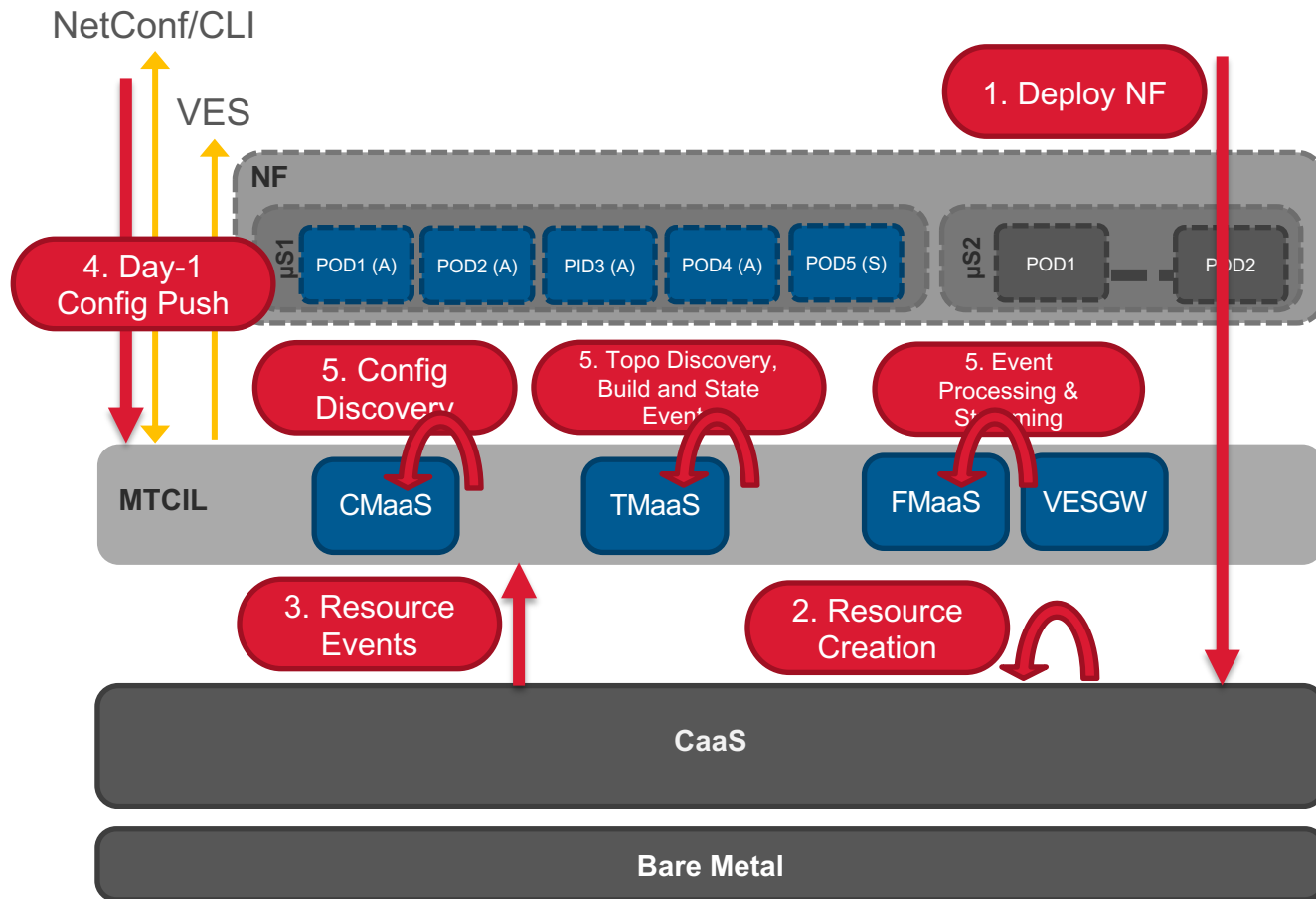
# Demo – Setup & Use cases



- When deployed, NFs are auto-initialised with dynamic Day-1 configuration
- When deployed, NF topology is automatically constructed, NFV states correlated and state events are generated.
- When deployed, notifications and faults are generated and notified northbound over VES
- When deployed, NF with HAaaS preference, is automatically configured to run PODs with active or standby roles as per the policy.
- When a configuration is changed via CLI (or over NetConf), changes are pushed to applicable μServices.
- When an active NFSI/POD fails, active role is switched over to a standby NFSI/POD.

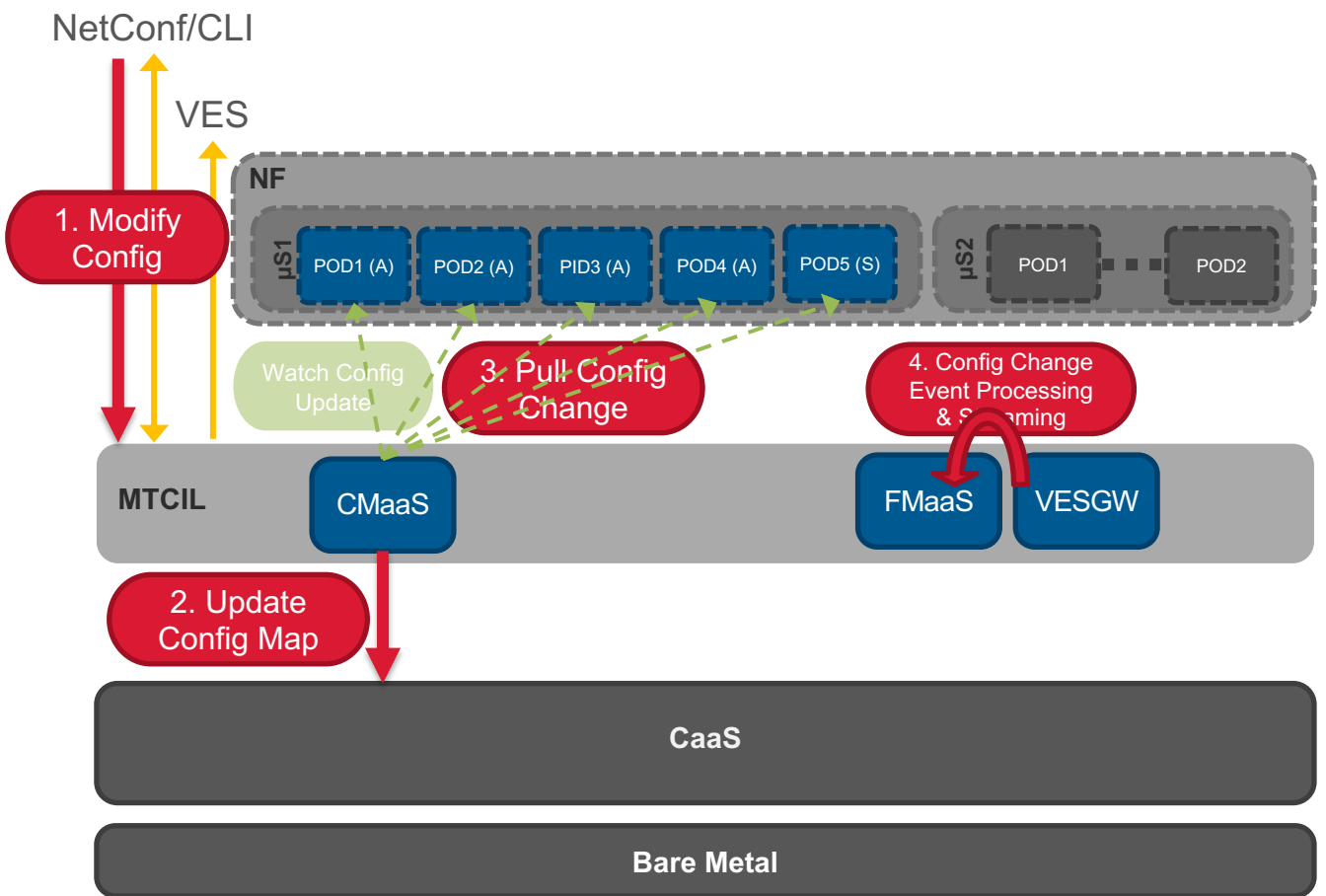


# Demo – Use case: Deployment and Discovery



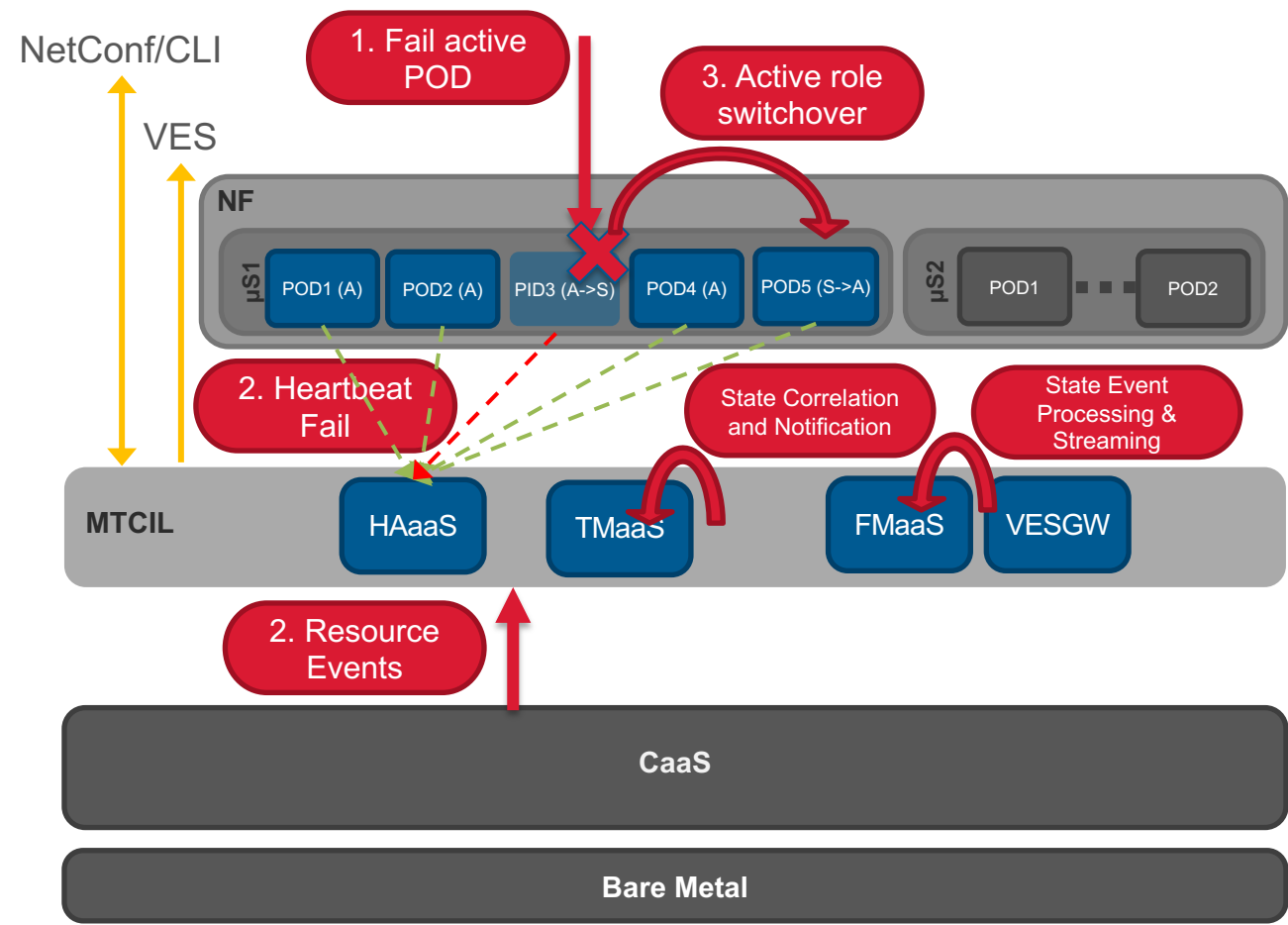
- When deployed, NFs are auto-initialised with dynamic Day-1 configuration
- When deployed, NF topology is automatically constructed, NFV states correlated and state events are generated.
- When deployed, notifications and faults are generated and notified northbound over VES
- When deployed, NF with HAaaS preference, is automatically configured to run PODs with active or standby roles as per the policy.

# Demo – Use case: Dynamic Config Update



- When a configuration is changed via CLI (or over NetConf/RestConf NBI), changes are pushed to applicable μServices.

# Demo – Use case: HA



- When an active NFSI/POD fails, active role is switched over to a standby NFSI/POD.

---

THANK YOU