XGVela Architecture Doc

Tracking info

Chapter	Editor	Latest Version	Date
chapter 0	Qihui Zhao	0.5	11/9/2020
chapter 1	Qihui Zhao	0.5	11/9/2020
chapter 2	Ying Liu	0.01	
chapter 3	Qihui Zhao	0.1	02/24/2021
chapter 4	Sandeep Karkala, Vance Shipley	0.01	
chapter 5	Deepak S	0.5	1/30/2021
chapter 6	Azhar Sayeed	0.8	11/6/2020

Link to previous draft

(Previous comments can be found in the following links)
chapter 0:
https://docs.google.com/document/d/1LE2EbQTY8SF9dUf8oNv8s8PjO6bPz43W_L4Uh1oniZQ/edit?usp=sharing
chapter 1:
https://docs.google.com/document/d/1EKPLjpU0aoZTS7-g1idMp012BxgZZmyvy7FfD7MxhMM/edit?usp=sharing
chapter 2:
https://docs.google.com/document/d/1U8fdzYsjmyIz1kx6PRD66jnf01NCqxRCvrgxTdTpUOE/edit?usp=sharing
chapter 3:
https://docs.google.com/document/d/1VXHoJkGVZNwHnkK3rLL7fchY0zyhf2gygmPmEGK9TOQ/edit?usp=sharing
chapter 4:
https://docs.google.com/document/d/1RxpapmSu5ZpXIpM8dtZTTzMHWVH2snDwjjeOa_3Nong/edit?usp=sharing
chapter 5:
https://github.com/XGVela/XGVela/blob/master/doc/Architecture%20Documents/xGVela_Networking_Requirements.docx
chapter 6:
https://drive.google.com/file/d/1IvcMtxbGXmqjwCvmsnKRi-PR30nbUpwo/view?usp=sharing

Chapter 0: Introduction

0.1 Why need cloud native PaaS in telecom industry

This section describes why we need a cloud native PaaS for telecom industry.

Telecom operators' revenue growth in traditional business areas is weak, and they need to seek new breakthroughs in innovation fields. Flexibility and fast online new services will be the key to achieve new revenue streams. The traditional 2G/3G/4G network mainly serves individual users. The network design is less differentiated and more unified, and its business experience guarantee often relies on the network to do its best. In the era of 5G and the future Internet, users of the 2B industry need differentiated and customized services. The accelerated digital transformation of the industry puts forward higher requirements for network flexibility.

At present, the telecommunication network has the following problems:

1. Inflexible network functions due to monolith architecture: monolithic network functions are inflexible when upgrading, scaling in/out, generating new network functions.
2. Capability of cloud platforms is relatively single, which mostly provides only infrastructure resources: pure infrastructure resources are not able to support convenient innovation of applications/ network functions. It requires NF/App to implement some duplicated functionalities, such as observability, monitoring, which are not related to core NF/App logics and cost a lot of effort.
3. Development and operation processes are complex, costly and slow: the business processes of existing telecommunication network elements strictly follow the sequence of requirements analysis, design, encoding, integration, testing and delivery. Product delivery duration is usually calculated by month. Future network use case/ scenario is changeable. Customer requirements may be unclear at the beginning and would continuously change as service grows. Traditional development and management processes need improvement to be more automatic.

With the rise of cloud native technology, containers, microservices, Devops, low coupling back-end services and other technologies and concepts have some advantages in solving the above problems.

At present, the telecom industry has seen advantages and disadvantages of cloud native. It is generally believed that the telecom network is gradually evolving towards the cloud native, although there are challenges in terms of technology implementation and architecture adjustment. It is a good time for all telco operators, telco vendors, IT cloud providers to work together to explore the method of cloud native evolution within the telecommunication industry. And we think a flexible cloud native PaaS platform specially designed for the telco industry could help to implement cloud native technology and concepts in telecommunication networks.

Within this document, the architecture and telco-featured functionalities of XGVela, which is a telco cloud native PaaS platform, would be explored. This document would cover the platform definition, telco PaaS requirement analysis, open source software selection, gap analysis. We hope to provide XGVela's core architecture and instruction for future platform implementation with this document.

Chapter 1: Definition of XGVela

This document contains the definition of XGVela. It covers a summary of XGVela capabilities, figure of XGVela architecture, definitions of XGVela and project scope.

XGVela is a telecom cloud native PaaS platform designed to provide PaaS capabilities needed in the upper application software, network element development, operation and maintenance.

XGVela could contains the following capabilities:

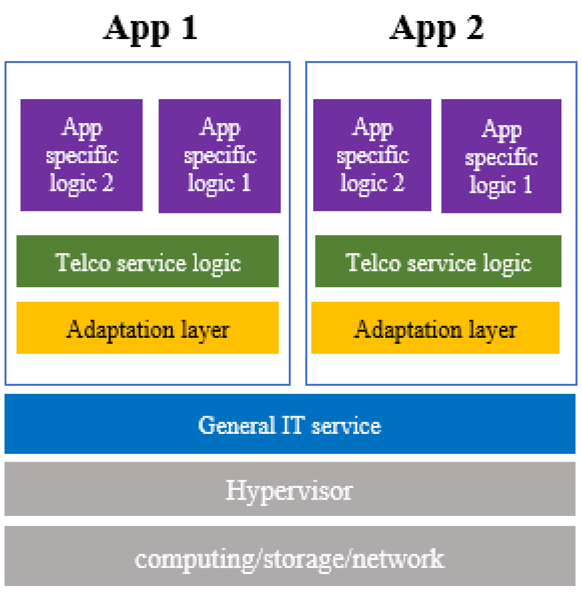


figure 0 XXX (for capability #1)

- NF/application related common microservice functional components: these components are functional units of NF/application, which can implement one or multiple functions of NF/application. The components could be common among different vendors, such as database, load balancer, firewall etc., or could be common among different NF/applications but unique for vendors/operators, such as customized UI, common NF microservices, etc.
- Observability and management capabilities: these capabilities would provide observability and management for resources, PaaS components and NFs/applications, which include performance management, logging, alarm, configuration management, tracing, etc.

- Infrastructure related capabilities: these capabilities could help to enable infrastructure abilities, such as hardware acceleration, and make them available as PaaS capability.
- PaaS management capabilities: these capabilities are related to lifecycle management of PaaS components, which may include image repository, LCM, etc.

1.1 High-level architecture and definition of XGVela

Considering that there are many open source and commercialized implementations of PaaS and cloud native technologies, we category the above capabilities into three categories, which has been shown in the following figure: Telco PaaS (green), Adaptation Layer (yellow), and General PaaS (blue).

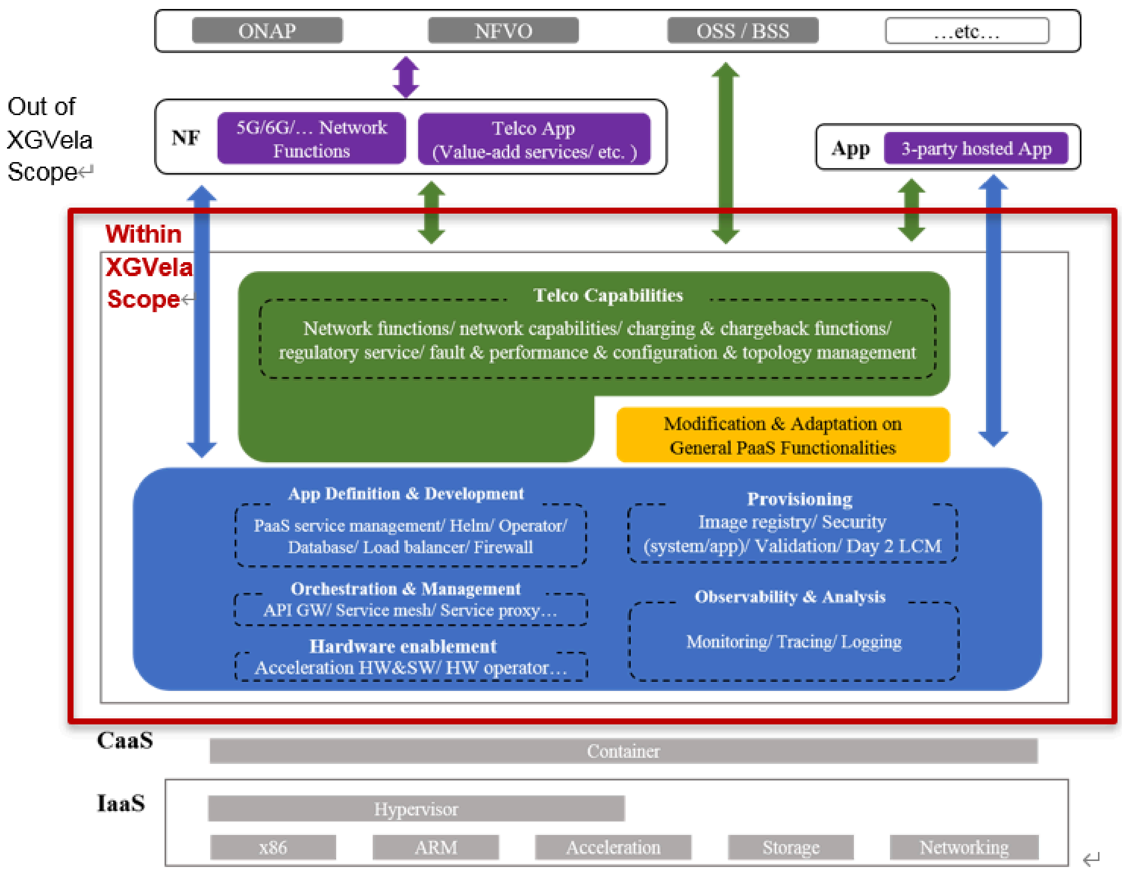

ONAP
NFVO
OSS / BSS
…etc…
Out of XGVela Scope
NF
5G/6G/… Network Functions
Telco App (Value-add services/ etc.)
App
3-party hosted App
Within XGVela Scope
Telco Capabilities
Network functions/ network capabilities/ charging & chargeback functions/ regulatory service/ fault & performance & configuration & topology management
Modification & Adaptation on General PaaS Functionalities
App Definition & Development
PaaS service management/ Helm/ Operator/ Database/ Load balancer/ Firewall
Provisioning
Image registry/ Security (system/app)/ Validation/ Day 2 LCM
Orchestration & Management
API GW/ Service mesh/ Service proxy…
Observability & Analysis
Monitoring/ Tracing/ Logging
Hardware enablement
Acceleration HW&SW/ HW operator…
CaaS
Container
IaaS
Hypervisor
x86
ARM
Acceleration
Storage
Networking

figure 1: High-level architecture diagram of XGVela

Definition of General PaaS (blue block in figure 1):

- A software platform orchestrated by kubernetes
- Provide environment and mechanism to run/manage applications and its associated services
- Provide abstracted capabilities/services to develop, manage and provide add-on values to applications through standard APIs with standard procedures
- General PaaS functionalities comes from existing CNCF and open source projects

List functionalities for General PaaS (not tools to implement)

such as logging

Definition of Adaptation layer (yellow block in figure 1):

- Adaptation/enhancement for General PaaS to be used within telco scenarios
- Prefer to be implemented in the form of plug-ins, drivers and other non-invasive ways
- Be used together with General PaaS

List functionalities for Adaptation layer (not tools to implement)

such as logging model

Definition of Telco PaaS (green block in figure 1):

- A software/service category on top of General PaaS
- Provide capabilities/services to be used to support development and management of telco services including 5G-Core, 5G-RAN, telco value-add service and other telco/5G beyond functionalities

List functionalities for Telco PaaS (not tools to implement)

1.2 Scope f XGVela

- Select the preferred General PaaS, analyze gaps with telecom requirements based on specific use cases
- Take the gap between telecom requirements and General PaaS as input of adaptation layer, implement adaptation layer for General PaaS
- Explore Telco PaaS use cases, functionalities and realize them
- Define APIs, models, workflows of XGVela platform
- Do platform functionality testing, and joint testing with other community and products

Chapter 2: PaaS Management

2.1 Overview

As a cloud-native PaaS platform for the telecom industry, XGVela provides platform capabilities and necessary components for telecom network functions. These capability components run on the underlying infrastructure platform in the form of containers, such as CaaS built on Kubernetes. CaaS can provide containerized infrastructure orchestration of capability components for application running . While XGVela needs to implement application-oriented capability component orchestration on top of CaaS. And this level of abstraction is PaaS management.

2.2 Architecture

Note: XGVela is described from the perspective of PaaS function types in the high-level architecture in Chapter 1. This section will explain XGVela's architecture and the relationship with the north/south system from a management perspective.

From the definition in the previous section, it can be concluded that the XGVela architecture should include various necessary capability components and the management of them and the platform.

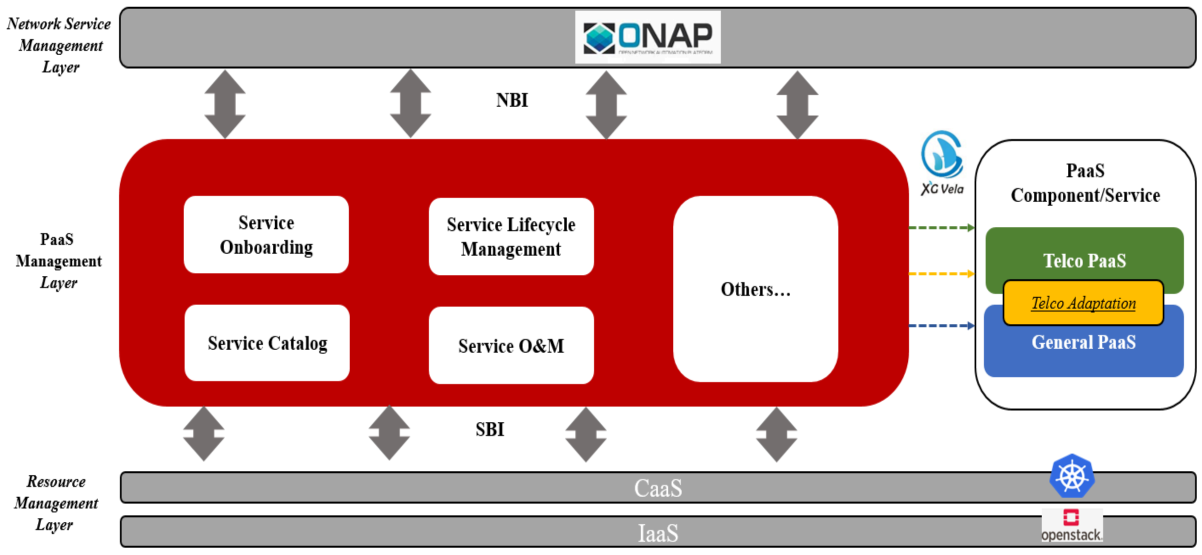


figure 3:PaaS management architecture diagram of XGVela(The red block)

Possible function blocks

Service Onboarding:

The service onboarding module is mainly responsible for bringing new components and services into the platform management, and it includes service onboarding and off-the-shelf. The main functions of this module should include:

- Support access to services of different packaging forms, including virtual machines,bare metal and container, and third-party software running on other servers in the local data center
- Support custom service capabilities, such as through standardized configuration templates and customized new service capabilities based on existing services
- Can customize the running environment of middleware
- ...

Service catalog:

- It is to manage and display the list of capabilities and components that the paas platform can provide. After the service is onboarding, it can be used as a template for the management plane to instantiate, and be bound and used by applications/CNF
- The service catalog can be provided through a web interface, such as a visual service market
- ...

Service lifecycle management

The platform provides full lifecycle management capabilities for service components

- Support the service of components through virtual machines, bare metal or containers, and implement unified lifecycle management
- Develop elastic scaling strategies for service instances based on monitoring KPIs
- Can customize the configuration of service scale and resource specifications according to the needs of the application
- Provide web UI for full lifecycle management
- Provide service instance management, including start, stop, restart, copy, upgrade, repair, delete, etc.
- Provide a unified user management and authentication mechanism, support RBAC model, and users can obtain access to resources through the association of roles and operating permissions
- Provide security certification and protection capabilities for platform services and middleware
- ...

Service operation and maintenance management

Mainly provide operation and maintenance management functions for running services to ensure stable running status of services

- Implement service installation and deployment, health check, log audit, etc.
- Service tracking, recording all operations in the service life cycle for problem retrospection and operation and maintenance management
- Provide operation and maintenance web UI
- Service operation and maintenance operations can be performed based on monitoring data
- Include the metering module.Count and expose the number and frequency of service usage,etc.
- ...

Paas management interface functions

- Northbound interface:
    - Service listing API
    - Service lifecycle management API
    - Platform and service operation and maintenance API
- Southbound interface:
    - Apply for and bind different types of CaaS resources (including containers, virtual machines, networks, storage, acceleration, etc.)

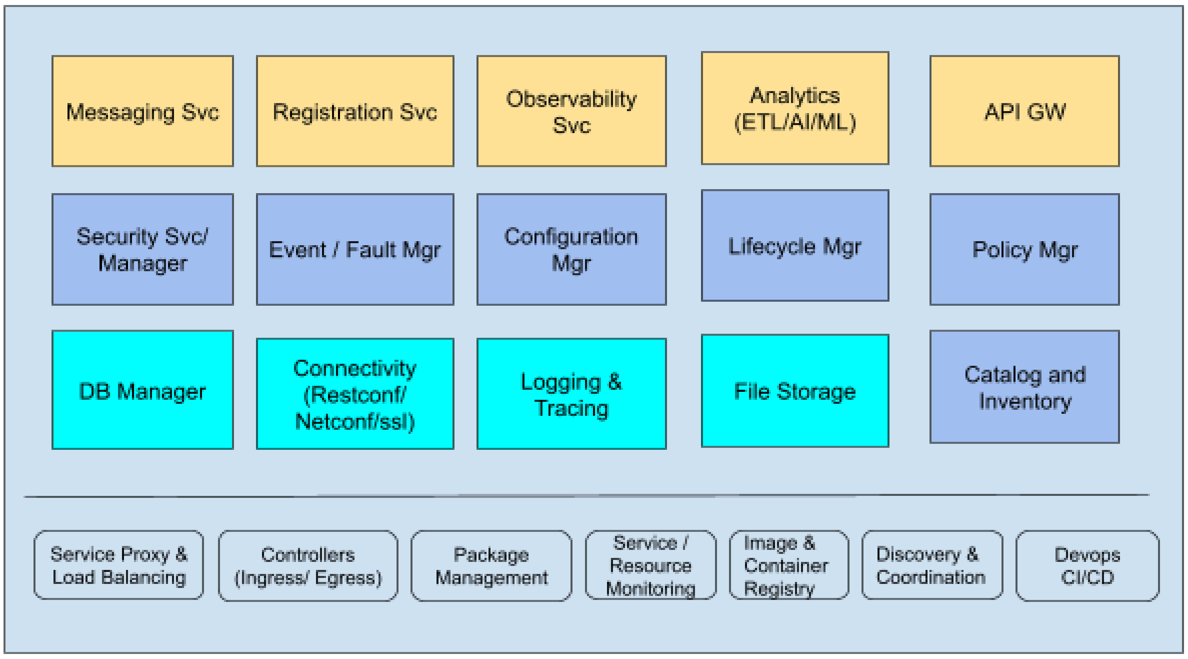
Other possible functions

- Support customize the application operating environment, deploy suitable middleware for different needs, and enable the infrastructure platform(CaaS) through add-ons
- Provide development capabilities for custom services, including development environment, development framework, development language, and access interfaces, etc.
- ...

Chapter 3: General PaaS

3.1 High-level definition

（by Seshu）



(by Qihui)

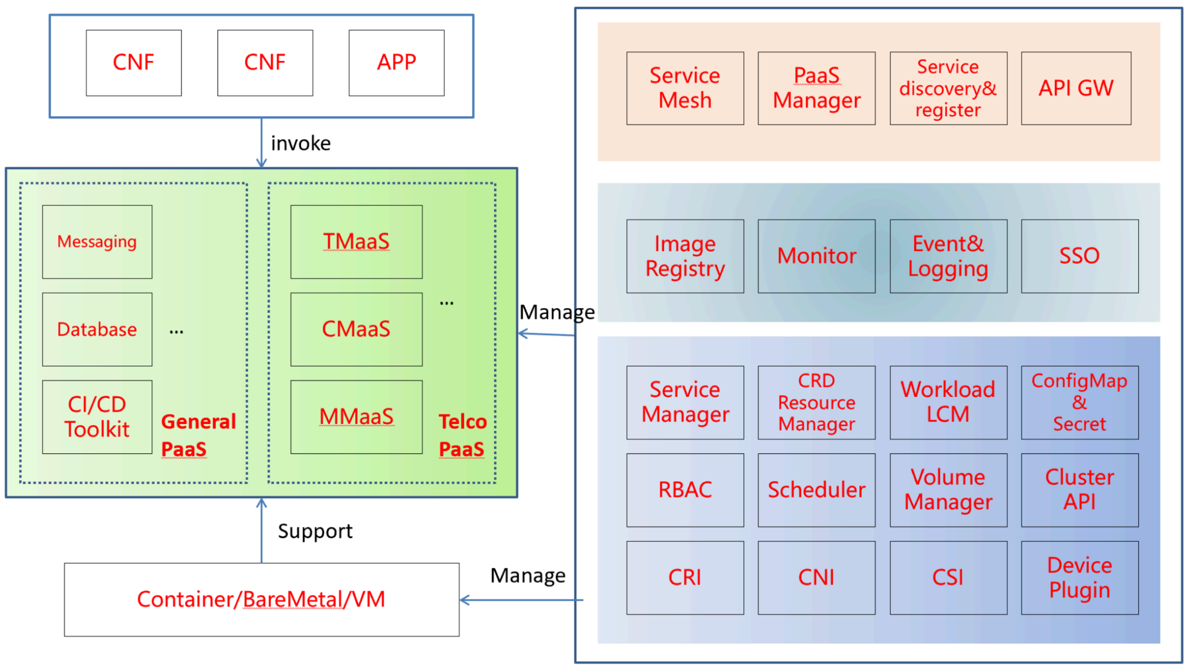


figure 2: Block diagram of General PaaS

Definition of General PaaS:

- *A group of abstracted capabilities/services that can support to develop, manage and provide add-on values for applications through standard APIs with standard procedures*
- *General PaaS capabilities/services can run on container environment and virtual machine environment*
- *Be commonly used by IT industries, and can play as foundation for other industry-specific PaaS capabilities*

D*efinition of XGVela General PaaS*

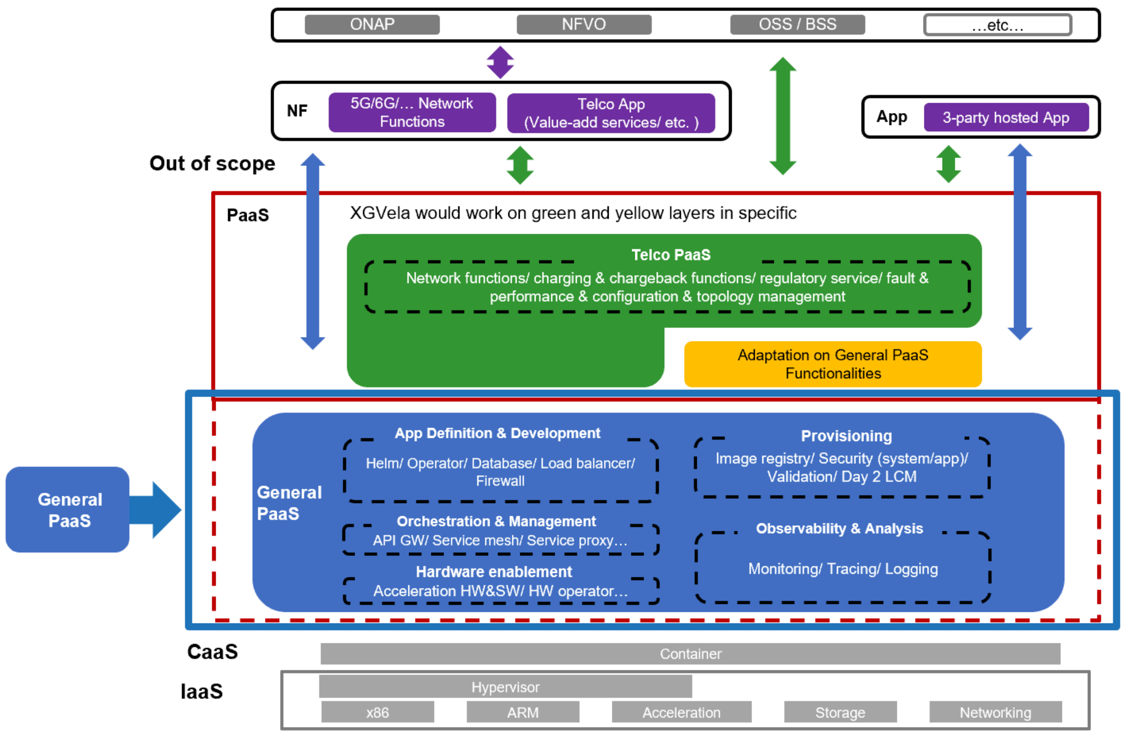
- *XGVela General PaaS run on container environment*
-

3.2 Requirements

Key requirements on General PaaS in XGVela are:

- Under Cloud Native scenario, General PaaS should support to run on container environment
- General PaaS shall not include Kubernetes
- General PaaS should support to be orchestrated by kubernetes, or other orchestration/ PaaS management software/system as long as well packaged and adapted
- Lifecycle management of General PaaS as well as Telco PaaS should follow “Service lifecycle management” in Chapter “PaaS Management”
- General PaaS should provide service to applications and end users through standard NBI
- General PaaS is recommended to refer to commonly used CNCF projects, and prefer to implement industry-specific adaptation through plug-n-play method (Loose coupling)
-

3.3 Architecture



ONAP
NFVO
OSS / BSS
…etc…
NF
5G/6G/… Network Functions
Telco App (Value-add services/ etc.)
App
3-party hosted App
Out of scope
PaaS
XGVela would work on green and yellow layers in specific
Telco PaaS
Network functions/ charging & chargeback functions/ regulatory service/ fault & performance & configuration & topology management
Adaptation on General PaaS Functionalities
General PaaS
General PaaS
App Definition & Development
Helm/ Operator/ Database/ Load balancer/ Firewall
Orchestration & Management
API GW/ Service mesh/ Service proxy…
Hardware enablement
Acceleration HW&SW/ HW operator…
Provisioning
Image registry/ Security (system/app)/ Validation/ Day 2 LCM
Observability & Analysis
Monitoring/ Tracing/ Logging
CaaS
Container
IaaS
Hypervisor
x86
ARM
Acceleration
Storage
Networking

3.4 General PaaS reference functionalities

Operation and Maintenance

- It is required that General PaaS can support real-time observability about the environment and PaaS services.
- General PaaS should monitor running status and performance of Pods. Monitoring data should be displayed through web UI.
- General PaaS should support collecting and storing logs generated by resources (required), PaaS capabilities (required) and applications (optional, based on end users' order). Logs can be displayed through the web UI, and outputted to other log management systems.
- General PaaS should trace all the operations happening within the lifecycle of PaaS services.
- General PaaS should support automatically managing alarms generated by resources and PaaS capabilities, including alarm collection, alarm display, alarm update, history alarm storage.
- **Existing solution:** Prometheus, ElasticSearchelastic search, Kibana, Jaeger ……

Image Repository

- It is required that General PaaS provide image repository and image management

- Image should be stored, managed and transmitted by layers to reduce the network pressure caused by duplicated image transmission
- Image should support to be managed and operated through CLI as well as UI
- Image repository should support multi-user environments and RBAC.
- **Existing solution:** Harbor

DevOps

- It is recommended that General PaaS support CI/CT/CD, code management and development project management.
- **Existing solution:** Jenkins, Maven, Jira, Argo

Microservice Architecture (Service mesh?)

- General PaaS should support microservice architecture
- Microservice architecture should support service breakers to avoid unnecessary waiting when calling a service. Service breakers should support auto breaking and manual breaking, and CRUD of breaking rules.
- Microservice architecture should support traffic control to ensure stable service.
- Microservice architecture should support time-out and retry mechanism to avoid endless waiting when service cannot be reached.
- Microservice should support displaying the topology of all related services, pods, which includes network topology, running status, performance, etc.
- **Existing solution:** ?Istio, Envoy, Kiali

API GW

-

Load Balancing

- General PaaS should provide LB as a PaaS service.
- LB includes east-west LB to manage traffic within a cluster, and south-north LB to manage traffic goes in-and-out a cluster.
- LB should support collaborating with services/microservices and dynamically update service status, access info and LB strategy.
- General PaaS should monitor LB including topology, health check, performance, etc.
- **Existing solution:** ?

Middleware

- General PaaS should support the following middleware:
  - Message queue
  - Cache
- General PaaS should monitor middleware including health check, performance, etc.
- **Existing solution:** RabbitMQ, Kafka, Redis, Memcache

Database

- General PaaS should support relational databases and non-relational databases.
- Database should support to be instantiated and provided to end users as a service.
- General PaaS should monitor database including health check, performance, chart number etc.
- **Existing solution**: MySQL, MongoDB

Chapter 4: Adaptation Layer

Place holder for extension of General PaaS

Chapter 5: Telco PaaS Management

5.1 Overview

Telco network poses certain key challenges for a seamless cloud deployment as it involves multiple network functions from multiple vendors, all designed in their own form and shape, as each vendor tries to bring in their own differentiators.

Migration to the cloud also propagates adoption of open-source software and de-facto standards. In an open-source ecosystem, software that can't adapt are less popular. Adaptability is as important as performance.

Also, in a mature Cloud-Native environment, PaaS layer would be owned by the Operator,

- PaaS stack is mostly made up of open-source software and CNCF helps certify a finite set of mature options.
- In the open-source community, standardisation is what the majority adopt. Any effort to standardise the PaaS stack and API's would be counterproductive.
- PaaS is also productised and delivered to operators as an off the shelf solution. PaaS vendors are trying to fill the gaps left behind after putting together the open-source software. In this effort there is a fair chance they might introduce proprietary variations.
- PaaS usage SLA's across operators might vary, impacting how much of PaaS services vendor applications can consume or rely on to deliver intended behavior and performance.

Cloud-Native platform needs to be segregated as,

- Telco PaaS: Focused on delivering Telco specific features, procedure and adaptations. Services in this layer interwork with Generic PaaS where necessary to deliver the end function. This layer can also be used for multi-tenancy to deliver application and vendor specific isolations and SLA's.
- Generic PaaS: Implements generic cloud native functions using generally available software and tools.

One of the key objectives of Telco PaaS, together with Generic PaaS, is also to provide all the 'platform' needs of a Cloud-Natve telco workload. So the applications are lightweight and contain only code to deliver the intended business logic.

Telco PaaS management provides 3GPP and cloud-native compliant northbound and southbound interfaces for OSS, MANO and PaaS integrations. At the same time, it delivers a high level of reliability, agility and scale expected of distributed and cloud-native systems.

Focus of Telco PaaS is to implement functions necessary to support Telco workloads and procedures in a cloud native environment. It shall not duplicate the functions already available in the Generic PaaS layer/community rather interface, enhance or adapt Generic

PaaS functions where available. As there are generally multiple options available in the Generic PaaS for any given function, interface and adaptations must follow plug-n-play approach to allow choices for the end-user and also ensure faster adaptations to newer and better options in the future. This is critical.

Telco PaaS management implements services, but not limited to, for Configuration Management, Fault Management, Log management, Performance Management, Topology management and High Availability for the managed NFs. There are opportunities to generalise certain other Telco specific functions such as subscriber tracing, Lawful Intercept (LI), CDR (Call Data Records), etc.

- Topology Management as a Service (TMaaS): This service models networks functions, components within each network function and associated resources as Managed Objects and makes it possible to manage the objects individually or as a group of related objects.
- Configuration Management as a Service (CMaaS): This service manages configuration of Network Functions and µServices. Configuration is described using Yang and encoded in JSON. NetConf and CLI are exposed for configuration management.
- Fault Management as a Service (FMaaS): This service implements 3GPP compliant fault and alarm management model. Provides interfaces for application µService to publish and subscribe to various events. It interfaces with the metrics management system (Prometheus) for TCA events. Exposed VES compliant NBI for notifications.
- Metrics Management as a Service (MMaaS): This service implements Prometheus at the core for metrics collection and monitoring and provides necessary correlations to 3GPP managed object model, events and clock aligned measurements.
- High Availability as a Service (HAaaS): Kubernetes does not meet fast HA requirements needed for certain Telco stateful services. HAaaS provides overlay HA capability by supporting a n:m Active-Standby model in a distributed scalable fashion.

4.2 RELATED DOCUMENTS

3GPP SPECIFICATIONS

1. 3GPP TS 28.532: Management and orchestration; Generic management services
2. 3GPP TS 28.533: Management and orchestration; Architecture framework

3. 3GPP TR 28.890: Management and orchestration; Study on integration of Open Network Automation Platform (ONAP) and 3GPP management for 5G networks
4. 3GPP TS 32.102: "Telecommunication management; Architecture"
5. 3GPP TS 32.300: "Telecommunication management; Configuration Management (CM); Name convention for Managed Objects"
6. 3GPP TS 32.352: Telecommunication management; Communication Surveillance (CS) Integration Reference Point (IRP); Information Service (IS)
7. 3GPP TS 28.533: Management and orchestration; Architecture framework
8. 3GPP TS 28.620: Telecommunication management; Fixed Mobile Convergence (FMC) Federated Network Information Model (FNIM) Umbrella Information Model (UIM)
9. 3GPP TS 28.622 Generic Network Resource Model (NRM); Integration Reference Point (IRP); Information Service (IS)
10. 3GPP TS 28.541 5G Network Resource Model (NRM); Stage 2 and stage 3

ETSI SPECIFICATIONS

11. ETSI GS NFV-SWA 001 Network Functions Virtualisation (NFV); Virtual Network Functions Architecture
12. ETSI GS NFV-IFA 011 Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification
13. ETSI GS NFV-IFA 013 Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification
14. ETSI GS NFV-IFA 015 Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on NFV Information Model
15. ETSI GS NFV-IFA 029 Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS"
16. ETSI GS ZSM 002 Zero-touch Network and Service Management (ZSM); Reference Architecture

ITU SPECIFICATIONS

17. ITU-T Recommendation X.710 (1991): Common Management Information Service Definition for CCITT Applications
18. ITU-T Recommendation X.721 Definition of Management Information
19. ITU-T Recommendation X.731 (1992): State Management Function

OTHER REFERENCES

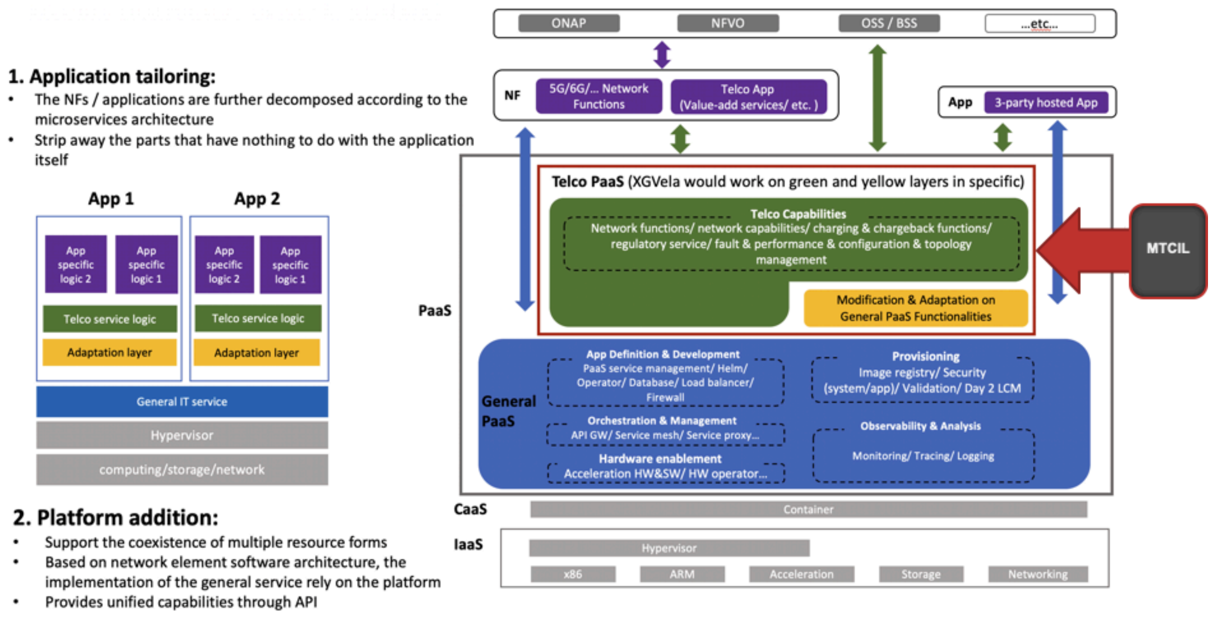
20. VES: https://wiki.onap.org/display/DW/5G+-+Real+Time+PM+and+High+Volume+Stream+Data+Collection
21. VES: https://wiki.onap.org/display/DW/VES+7.1
22. Kubernetes: https://kubernetes.io/

4.3 Requirements

Key requirements are,

- Telco PaaS shall implement functions necessary to support Telco workloads and procedures in a cloud native environment.
- Telco PaaS shall not duplicate the functions already available in the Generic PaaS layer/community rather interface, enhance or adapt Generic PaaS functions where available.
  As there are generally multiple options available in the Generic PaaS for any given function, interface and adaptations must follow plug-n-play approach to allow choices for the end-user and also ensure faster adaptations to newer and better options in the future. This is critical.
- Telco PaaS shall be deployable on any kubernetes based CaaS and Generic PaaS distributions that might be instrumented on Cloud or Bare metal.
- Telco PaaS shall support Telco standards compliant information and object models
- Telco PaaS shall support Telco standards compliant NBIs
  - NetConf for configuration management
  - VES for event notifications
  - TM Forum OpenAPIs
  - 3GPP 5G Service Based Interfaces (SBI)
  - NFV orchestration - Or-Vnfm-Em/Vnf???
- PaaS and CNF Packaging model and NFV alignment

4.4 Architecture



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

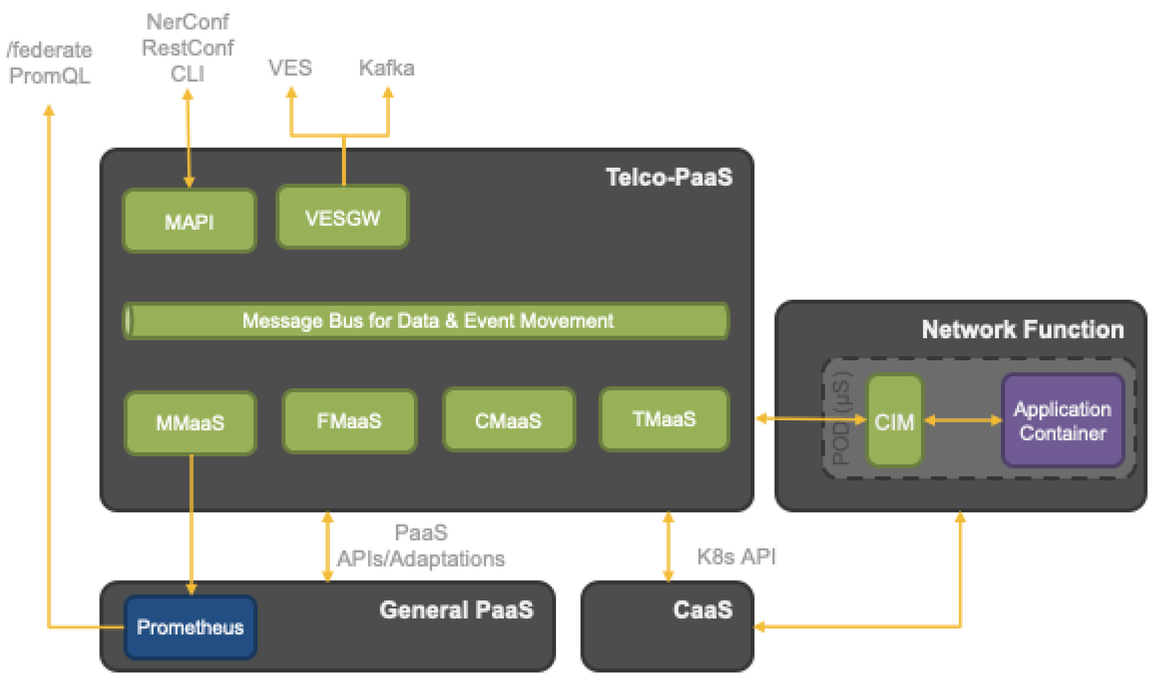
App 1
App 2
App specific logic 2
App specific logic 1
Telco service logic
Adaptation layer
General IT service
Hypervisor
computing/storage/network
ONAP
NFVO
OSS / BSS
...etc...
NF
5G/6G/... Network Functions
Telco App (Value-add services/ etc.)
App
3-party hosted App
Telco PaaS (XGVela would work on green and yellow layers in specific)
Telco Capabilities
Network functions/ network capabilities/ charging & chargeback functions/ regulatory service/ fault & performance & configuration & topology management
Modification & Adaptation on General PaaS Functionalities
MTCIL
PaaS
General PaaS
App Definition & Development
PaaS service management/ Helm/ Operator/ Database/ Load balancer/ Firewall
Provisioning
Image registry/ Security (system/app)/ Validation/ Day 2 LCM
Orchestration & Management
API GW/ Service mesh/ Service proxy...
Observability & Analysis
Monitoring/ Tracing/ Logging
Hardware enablement
Acceleration HW&SW/ HW operator...
CaaS
Container
IaaS
Hypervisor
x86
ARM
Acceleration
Storage
Networking

4.5 Existing Solution

Some of the functions for Telco-PaaS will be seeded from Mavenir MTCIL. For initial seeding, following functions are considered,

- Configuration Management as a Service (CMaaS)
- Topology Management as a Service (TMaaS)
- Fault Management as a Service (FMaaS)
- Core framework that includes the sidecar, CIM (CNF Interface Module). Discussed in the 'Proposed Architecture' section below.
- Helm based packaging framework

4.6 Proposed Architecture - high level



4.6.1 CIM

CNF interface module, a component of Telco-PaaS, provides a 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 applications over REST for APIs and NATS for messaging and events.

4.6.2 Telco-PaaS

- 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 applications via CIM and with Prometheus for TCA. Enriches and exports events VES (version 7.1). Events can also be pushed via Kafka.

4.6.3 Management Model

In a networked environment, the majority of the network functions (NF), especially carrier class functions, consist of many interrelated components that need to be individually managed by management services (MnS). Management network functions (MnF) are NF which implement MnS.

For each function in the network, there are many different components that need to be discovered, monitored and managed. For the management services to manage a set of network functions, the network functions are represented or modeled as managed objects (MO) which may be stored in a database to monitor state and perform management operations.

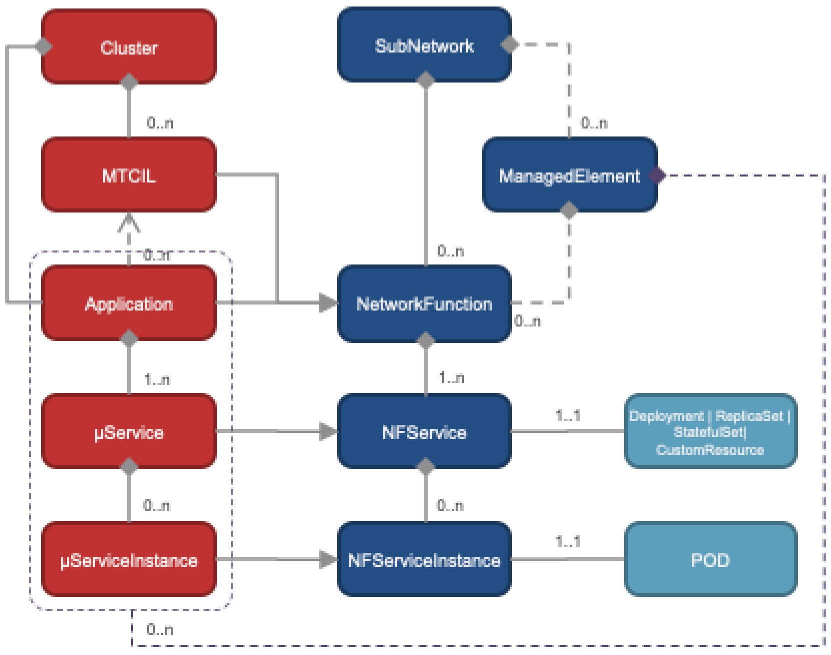
All network functions are componentized, virtualized and grouped into CNF. Management services manage functions based on CNF view.

- Each NF is disaggregated into sub-services (micro-services for containerization) of various types.
- One or more network functions can be deployed on Telco PaaS.
- One or more management functions may be deployed on Telco PaaS.
- Telco PaaS instance is deployed on General PaaS/CaaS and runs in its own namespace
- Each NF instance deployed on Telco PaaS runs in its own namespace.

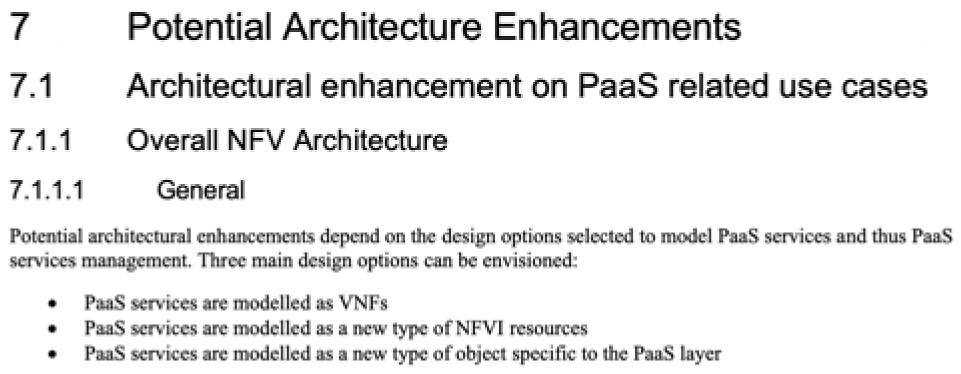
Telco PaaS defines and implements a generic ManagedObject model schema for describing the exporting the cluster and NF topology. Topology will be rendered as a hierarchical structure of ManagedObjects.

A ManagedObject models basic properties and relationships. These are updated with actual NF, µService resources (containers, pod, volumes, configuration, etc) properties and relationships upon discovery of the same via K8s APIs.

Managed Object model is based on and in most part derived from 3GPP NRM (Release 16).



Telco PaaS services are also managed as CNF's. This is an obvious and available option as also discussed under NFV (NFV-IFA-029).



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

4.6.4 State Management

A NF can assume a number of internal states to represent the status of the NF. Transitions between these states provide architectural patterns for some expected NF functionality. Before a NF can start its life cycle, it is a prerequisite that the NF was on-boarded (process of uploading NF package - charts, images, data etc.).

State	Description
Null	A VNF Instance does not exist and is about to be created.
Instantiated Not Configured	VNF Instance does exist but is not configured for service.
Instantiated Configured - Inactive	A VNF Instance is configured for service.
Instantiated Configured - Active	A VNF Instance that participates in service.

Terminated	A VNF Instance has ceased to exist.

Telco PaaS implements NFV states at NetworkFunction as well as NFService. States are correlated from k8s probes and resource events.

- Startup Probe
- Liveness Probe
- Readiness Probe
- POD event

Chapter 6: Infrastructure Networking enablement for Telco CNFs

5.1 Overview

This document describes the NF networking architecture, models and requirements for Telco's consumption of xGVela. The architectures and requirements apply to general PaaS and xGVela components.

These requirements have been identified as MUST, SHOULD and SHALL to show mandatory, expected and optional behavior

Telcos deploy a cloud environment at the far edge, edge including regional locations and the core data centers. The networking requirements and scenarios differ at each location. While internal and external networking within a platform is a MUST HAVE requirement at the network edge or Core Telco aggregation points, it also is a SHALL HAVE requirement as you move towards the far edge site within the Telco infrastructure depending on the end user requirements. Capturing networking requirements for all the use cases and deployment models is a tall order and requires a full workstream in Telco infrastructure design.

The goal of this document is not to revisit the deployment models or define CNF networking for all the model but to highlight additional asks for Telco specific use cases and deployments

Telco CNFs journey

With the emergence of Edge computing and 5G technologies, telecommunication providers are looking at ways to migrate their existing monolithic services to microservices architecture and containers to address the nextgen bandwidth and throughout needs. With Network function virtualization (NFV), these providers started moving from legacy hardware towards virtualized network function (VNF) running on COTS to now moving towards containerized network function (CNF) on cloud infrastructure.

VNFs are the current defacto standard network architecture to help telco providers to move from physical dedicated hardware towards more agile services, but we still have limitation like it is hard to manage and scale without decoupling the virtual appliance from the underlying infrastructure. Additionally, scalability is the desired feature in the NFV being the backbone of 5G or edge computing which demands the large-scale deployment, portability, agility, scalability and lower overhead.

The cloud-native environment is built for scale, fault-tolerance, resilience and agility. The promise of CNFs is to address some of the fundamental limitation of the VNFs by moving many of these functions into containers. Containerization of network architecture components makes it possible to run a variety of services on the same cluster and more easily on-board already decomposed applications, while dynamically directing network traffic to correct pods [3].

The below diagram shows typical VNFs and CNF architecture [4].

CNFs needs to interplay with VNFs and co-exits with legacy systems, are much necessary from a smooth transition from traditional network architecture to cloud-native network architecture. Since network functions are critical and backbone of the 5G network architecture, there is a need for CNFs to satisfy all the requirements of a network function like, networking (configuration, QoS etc), deployment, HA, scalability, telemetry, manageability etc.

This document covers the requirements of a network for a production ready CNFs in telco infrastructure.

5.2 Requirements

Key requirements of CNF networking are,

Networking requirements		
Requirements	Existing Solution	Gaps w.r.t TelcoPaaS
It SHALL have ability to support dynamic virtual Networks & dynamic Network Creation/Termination	Multus CNI, OVN-Multi CNI (OVN4nfv), controller,	
It SHALL have ability for adding provider networks to requested CNF.		

It MUST have ability to add multiple interfaces to a given network function.	OVNSFC, Kube OVN, SRIOV-CNI	
It SHALL have ability to create a Static and dynamic Network function chaining between the CNFs, applications		
It SHALL have ability to load balance across different instance of Network function		
It MUST have ability to dynamically add and remove Network Policy for a given CNFs		
It MUST have ability to route and manage the Co-existences of network functions and application		
It MUST have ability to manage the interoperability of CNFs, VNFs and legacy device.	Virtlet, CRI Proxy, KubeVirt	
It MUST have ability to dynamically configuration of Virtual switches/NICs	CNI – Kube OVN,	Network Operators? NMaaS?
It Shall have ability to configure the underlaying network with SDN support.	Kube OVN, OVN4NFV	

Performance requirements		
Requirements	Existing Solution	Gaps w.r.t TelcoPaaS
It shall configure the platform parameters for Low latency and jitter	SRIOV-CNI, OVS-DPDK CNI (Virtio-User), Smart NICs CNIs, OVN-Multi-CNI	NMaaS and Network Operators
It shall configure the platform parameters High bandwidth and throughput		
It shall configure the platform and software parameters for performance determinism across CNFs		

Security requirements		
Requirements	Existing Solution	Gaps w.r.t TelcoPaaS
It Must support the Multi-tenancy across multiple CNFs	TPM/TXT? SGx, DDP	SMaaS? (CNI Security Management as a Service?)

It Must enable the ability to authenticate and verify of underlaying infrastructure		
It Must enable the ability to Network traffic isolation		

General requirements		
Requirements	Existing Solution	Gaps w.r.t TelcoPaaS
It shall support APIs for orchestrating Network Service on Telco PaaS	Ansible, NFD, Multus, istio, Telemetry (collectd, prometheus), Ceph, CMK, FPGA etc…	
It Shall support Multi-residency support		
It shall support app level and platform level network Telemetry & Monitoring		

5.3 High Level Architecture

XGVela is a telecom cloud native PaaS platform designed to provide PaaS capabilities needed in the upper application software, network element development, operation and maintenance. Below is the High-level architecture diagram of XGVela.

The Green Box is the Telco PaaS layer that abstracts the telco specific capabilities for easy deployment and management of network function using underlaying General PaaS. The Networking requirements of a network function are addressed on both Telco PaaS and General PaaS.

General PaaS layer will be responsible for Provisioning the physical NIC, providing multiple interfaces, telemetry, policy apply, enhanced platform features etc. While the Telco PaaS microservice (Network-Management-as-a-Service (NMaaS)) will be responsible for abstracting the APIs that will be consumed by the NFVO/Orchestration layer.

Proposed Architecture - high level for Networking

Traditionally, multiple network interfaces are employed by network functions to provide separation of control, management and data/user network planes. They are also used to support different protocols or software stacks and different tuning and configuration requirements. Typically, in Kubernetes each pod only has one network interface (apart from a loopback) which is not enough for a production ready telco network function. Multus Container Network Interface (CNI) is the device plugin that can be used to create multiple network interfaces for pods in Kubernetes.

NmaaS: Network Management as a Service:

- Exposes NB APIs to Orchestration to configure the Infrastructure NIC, add/delete interface to NF at runtime, SRIOV configuration etc.
- Talks to Network Operator to setup the required underlaying driver/software etc needed based on the Network function definition

Multus: A container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods

Userspace CNI: A Container Network Interface (CNI) plugin designed to implement userspace networking (as opposed to kernel space networking). An example is any DPDK based applications. It is designed to run with either OVS-DPDK or VPP along with the Multus CNI plugin in Kubernetes deployments

OVN-Multi-CNI: A Container Network Interface (CNI) plugin to manage network traffic flows, network policy support, including ingress and egress rule etc.

SRIOV- NIC CNI: A Container Network Interface (CNI) plugin enables the configuration and usage of SR-IOV VF networks in containers and orchestrators like Kubernetes.

Deployment Models of PaaS Networking.

Make Before Break principle

5.4 Proposed Architecture

5.4.1 Existing Solution:

This section lists the open source projects which satisfy the platform networking enabling requirements for xGVela.

OpenNESS[7] (Open Network Edge Services Software) is an edge computing software toolkit that enables highly optimized and performant edge platforms to onboard and manage applications and network functions with cloud-like agility across any type of network. OpenNESS Enhanced K8s extension makes it suitable to deploy Telco workload (CNFS) including management, control and data plane along with traditional cloud applications. This Features can be base line for the Telco PaaS. Some of the network function specific features like CNFs, multi interface, acceleration support, SFC, telemetry etc that OpenNESS enables will be discussed in this section

Below is the list of the Opensource projects that satisfy the requirements of networking to run the CNFs.

Networking requirements

Requirements: <add explanation for every requirements>

Current Opensource components address the requirements: e.g, ovn-multi CNI, Multus, etc.

<I will expand this further>

Gaps:

Performance Requirements:

Requirements: <add explanation for every requirement>

Current Opensource components address the requirements: e.g, ovn-multi SRIOV CNI etc, ovs-dpdk CNI, smart NICs etc

<I will expand this further>

Gaps:

Generic requirements

Requirements: <add explanation for every requirement>

Current Opensource components address the requirements: e.g, deployment, ONAP, EMCO, kubespray, telemetry etc.

<I will expand this further>

Gaps:

List of CNIs that will enhance the dynamic configuration and multi network performant network for CNFs

Multus-cni:

Multus CNI is a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods. Typically, in Kubernetes each pod only has one network interface (apart from a loopback) -- with Multus you can create a multi-homed pod that has multiple interfaces. This is accomplished by Multus acting as a "meta-plugin", a CNI plugin that can call multiple other CNI plugins.

https://github.com/intel/multus-cni

Ovn4nfv-k8s

OVN Multi CNI and associated network watcher (for K8S) Enables multiple OVN based networks and enables PODs/VNFs to sit on multiple virtual networks. Founded by Intel in OPNFV: https://gerrit.opnfv.org/gerrit/gitweb?p=ovn4nfv-k8s-plugin.git;a=summary

Kube-OVN

Kube-OVN integrates the OVN-based Network Virtualization with Kubernetes. It offers an advanced Container Network Fabric for Enterprises with the most functions and the easiest operation.

ligato.io

An Open Source Go Framework for Building Applications to Control and Manage Cloud Native Network Functions (CNF)

SRIOV-NIC CNI

The SR-IOV CNI plug-in plumbs VF interfaces allocated from the SR-IOV device plug-in directly into a Pod

https://github.com/intel/sriov-cni

OVS-DPDK CNI

The Userspace CNI is a Container Network Interface (CNI) plugin designed to implement userspace networking (as opposed to kernel space networking). An example is any DPDK based applications. It is designed to run with either OVS-DPDK or VPP along with the Multus CNI plugin in Kubernetes deployments. It enhances high performance container Networking solution and Data Plane Acceleration for containers. https://github.com/intel/userspace-cni-network-plugin

5.4.2 Telco PaaS Specific Components:

- **NmaaS: Network Management as a Service:**

<Will add low level details of these new components>

- **Network Operator:**

<Will add low level details of these new components>

5.4.3 Analysis - Gaps, Pros and Cons

5.5 Gaps

List the gaps in the current architecture

Analysis of where these gaps can be addressed

5.6 Summary

5.7 References

[1]https://www.openshift.com/blog/telco-revolution-or-evolution-depends-on-your-perspective-but-your-network-is-changing

[2]https://www.intel.in/content/www/in/en/communications/why-containers-and-cloud-native-functions-paper.html

[3 https://www.redhat.com/en/blog/cnf-and-vnf-certification-red-hat-and-intel

[4]https://wiki.onap.org/download/attachments/79203136/VNF-CNF%20OVP%20-%202%20key%20slides_V3.pptx?version=2&modificationDate=1582822482000&api=v2

[5] https://github.com/XGVela/XGVela/wiki/Architecture-Document

[6]https://events19.linuxfoundation.org/wp-content/uploads/2018/07/ONS2019_Cloud_Native_NFV.pdf

[7] https://www.openvswitch.org/support/ovscon2019/day1/1133-OVNForK8sNetworkFunctions.pdf

[8]https://events19.linuxfoundation.org/wp-content/uploads/2017/11/2018-OSSNA-CNF-Journey-in-Telecom-Seminar.pdf

[9] https://events19.linuxfoundation.org/wp-content/uploads/2018/07/ONS2019_Cloud_Native_NFV.pdf

Chapter 7: High Availability

6.1 Overview

This document describes the high availability architecture, models and requirements for Telco's consumption of xGVela. The requirements and architectures described in this document apply to general PaaS and xGVela components alike. Some of these requirements can be achieved through the tuning/deployment models of the platform and others can be achieved via the orchestration or workloads.

These requirements have been identified as MUST, SHOULD and SHALL to show mandatory, expected and optional behavior

Telcos deploy a cloud environment at the far edge, edge including regional locations and the core data centers. The availability requirements differ at each location. While high availability of the platform is a MUST HAVE requirement at the core data center, it may be a SHALL HAVE requirement as you move towards the far edge site with in the Telco infrastructure. Capturing high availability requirements for all the use cases and deployment models is a tall order and requires a full workstream in Telco infrastructure design.

The goal of this document is not to revisit the deployment models or define high availability for all the model but to highlight additional asks for Telco specific use cases and deployments

6.1.1 High Availability for Telco Networks

High availability is needed to ensure service availability and continuity. Enabling mechanisms are distributed throughout the system, at all levels, and the redundancy of the hardware and software subsystems eliminate single points of failure (SPOFs) that can affect the service. Control mechanisms for these redundant components are distributed to associated control and redundancy management systems. The control processes should be implemented at the lowest level that has sufficient scope to deal with the underlying failure scenarios, with escalation to higher-level mechanisms when failure cannot be handled at lower levels. This ensures that the fastest mechanisms are always used, as the remediation control loop times generally increase when the decision logic is moved up in the stack or into broader control domains.

High availability alone is not sufficient for telco services. In addition to high availability of the service, service continuity after failures and failovers is also required. Service continuity requires associated state protection mechanisms to be implemented for the stateful components, e.g. using state checkpointing. State protection mechanisms are the responsibility of the associated state "owner" processes, whether they are part of the infrastructure or part of an application. In other words, SDN controllers would be responsible for their internal state protection, while applications implemented as network functions would be responsible for their internal state.

For the redundancy, the target is to decrease the amount of redundant resources required. This implies that the direction of the redundancy structures is away from dual-redundant (typically active-standby 1:1 prevalent in the "box" implementations) towards either N+1 or N:1 schemes. This is enabled by the reduction or elimination of the direct physical attachment associations with the use of the cloud native functions, a move towards micro-services types of architectures, and the ability to add, remove and relocate service instances dynamically, based on offered load and resource utilization state. The infrastructure support for such mechanisms is essential for success, and the new, more dynamic configuration of the applications and services has many implications for both network services as well as the whole control, orchestration and assurance software stack.

The availability and continuity mechanisms use "intent"-driven interfaces. The intent specifies the desired state, connectivity or other aspect of the system from the service user's perspective. The orchestration and control systems determine the implementation of the request, using the network resources available at the time of new request, and subsequently ensure that the intent continues to be met while the service is in use.

The associated orchestration and control subsystems are always aware of both the intended configuration and the actual configuration, and can autonomously work to drive the system to intended state should there be deviations (either due to failures or due to intent changes).

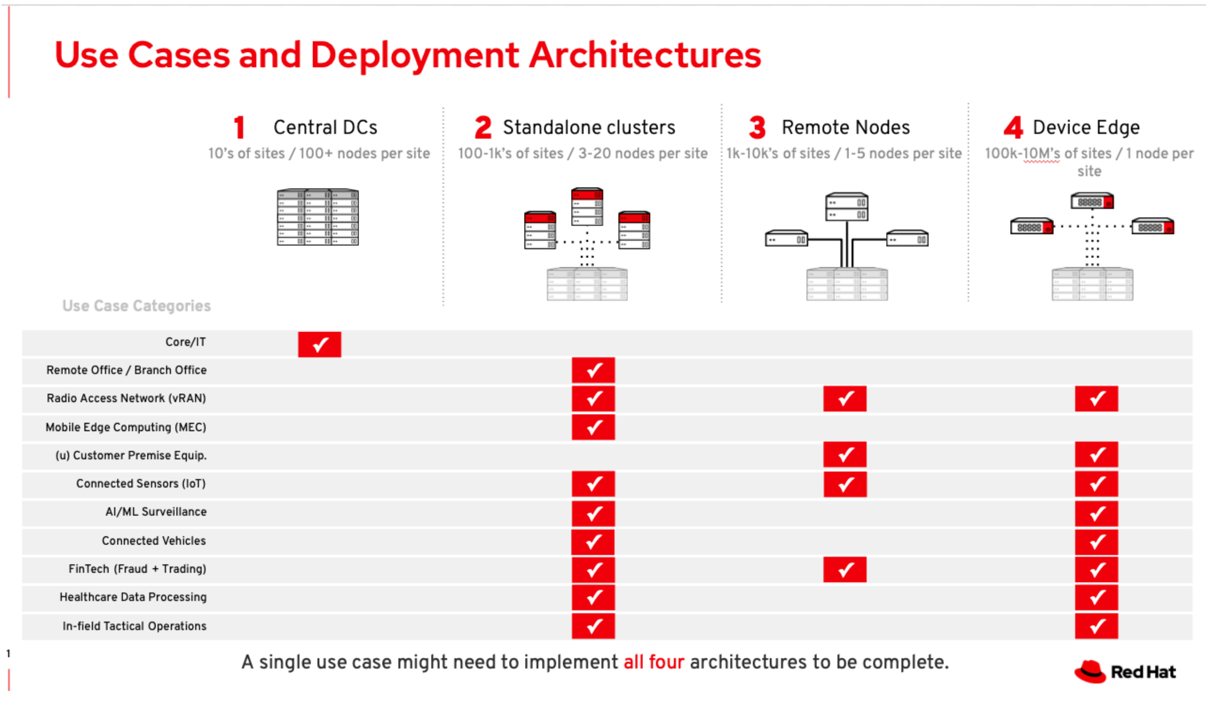
6.1.2 Network Topology HA Considerations

HA mechanisms associated with Network topology are out of scope of this document.

6.2 Architecture

In the access network, redundancy mechanisms are more technology-specific, and also vary by the market or application. For example, redundancy and access connection redundancy are typically not supported for residential customers due to extra cost, while for large and critical business locations, redundancy may be supported all the way to the customer premises. Redundancy mechanisms need to be investigated in the context of the specific access technology, especially for mechanisms at the level of the subscriber interface. While topics like Network redundancy and HA are out of scope of this document, it is expected that Telcos will design the PaaS to be implemented with multiple network interfaces, hardware and software redundancies to be full HA.

Figure below describes the requirements of Telco and Enterprise and various deployments scenarios. Each such deployment scenario results in a HA model that is different. For example, the HA model for Central DCs or stand alone clusters may be full HA while the HA model for remote nodes or device edge may be partial or no HA.



Use Cases and Deployment Architectures

Use Case Categories	1 Central DCs 10's of sites / 100+ nodes per site	2 Standalone clusters 100-1k's of sites / 3-20 nodes per site	3 Remote Nodes 1k-10k's of sites / 1-5 nodes per site	4 Device Edge 100k-10M's of sites / 1 node per site
Core/IT	✓			
Remote Office / Branch Office		✓		
Radio Access Network (vRAN)		✓	✓	✓
Mobile Edge Computing (MEC)		✓		
(u) Customer Premise Equip.			✓	✓
Connected Sensors (IoT)		✓	✓	✓
AI/ML Surveillance		✓		✓
Connected Vehicles		✓		✓
FinTech (Fraud + Trading)		✓	✓	✓
Healthcare Data Processing		✓		✓
In-field Tactical Operations		✓		✓

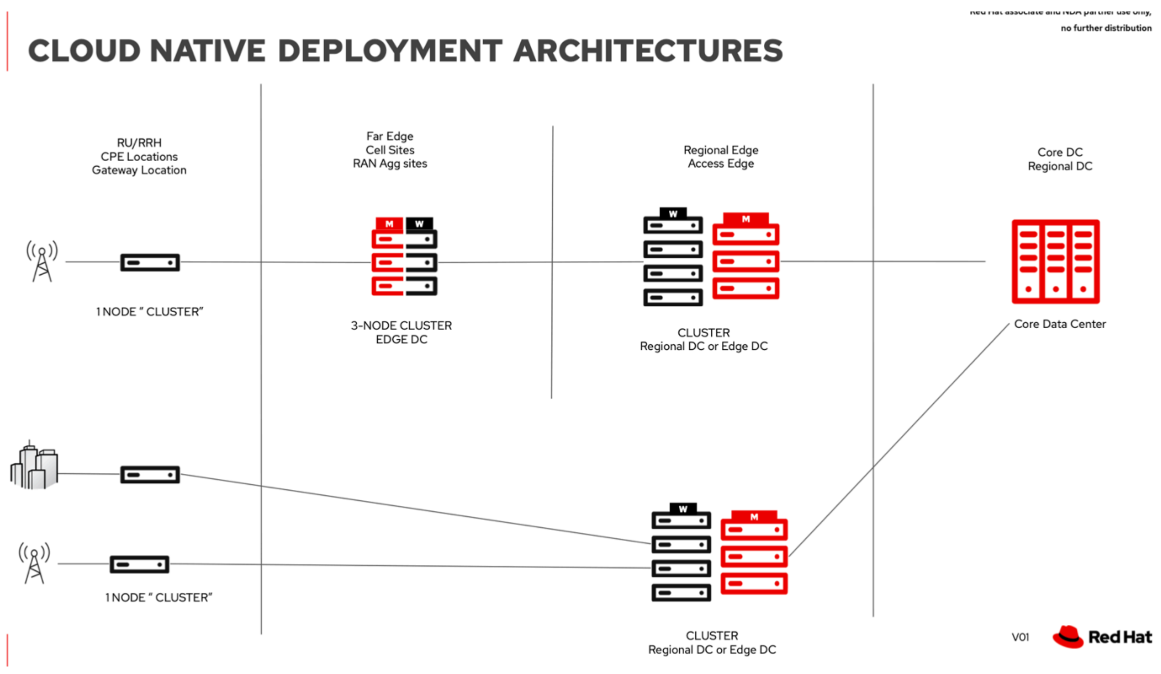
1

A single use case might need to implement all four architectures to be complete.

Red Hat

6.2.1 Deployment Models of PaaS.

xGVela PaaS can be deployed in the core DC, regional DC, in the central office, Metro or regional POPs, In building centers and far edge sites such as cell site aggregation locations, enterprise CPE, ruggedized IoT gateways etc. These are multiple nodes at core or regional locations to single nodes in far edge situations.
Refer to picture below



6.3 Requirements

Telco infrastructure can be divided into three categories for availability of the infrastructure. They are:

- No HA: For the infrastructure components – the high availability for services and applications can be achieved through stateful applications that can reconnect to different upstream or downstream devices and applications in case of failure. However, upon failure of the device, the application or the microservice running in that environment is shut down and no service is available from that instance of the application
- Partial HA: In this context there is high availability of the platform but limited to some failure scenarios such as single failure or double failure where the infrastructure can continue to operate but in a degraded environment
- Full HA: The infrastructure operating the cloud environment is operating in full HA mode where upon failure of any server hardware component or software component is full capable of continued operations as if nothing happened.

With the above scenarios following are the overall requirements for xGVela PaaS on HA

- The PaaS SHALL be capable of being deployed in any footprint – VM or bare metal and HA is equally applicable to both models
- The PaaS MUST support quorum-based redundancy models for controllers. This implies there are n>=3 controllers where n is odd such that (n/2)+1 controllers are required to be operational for full quorum

- The PaaS SHALL support Active-Standby redundancy model for cluster controllers. If a quorum-based model is not supported then the cluster shall support an active standby controller model at a minimum. Active-Standby implies one controller is active while another one is on hot or cold standby and can take over operations upon detection of failure or via manual intervention.
    - Active-Standby model can be a 1+1 or N+M where N and M >= 1. This model while complex in implementation is well known and understood by Telcos in which N active controllers are backed up by M standby controllers
- The PaaS MUST allow deployment of stateful and stateless applications. The HA of the cluster MUST allow for deployment of all types of applications and MUST NOT restrict any application types due to sharing of resources such as registry, databases etc.
- The PaaS MUST be capable of upgrades – see more details below on each operational environment requirements
- The PaaS MUST accomplish graceful shutdown during degraded environment operations
-

6.3.1 Non HA Operational environment for PaaS

This operational environment is a PaaS or its components deployed in a single nodes or a server in a remote location such as far edge. If the server/device fails there is no service – because there are no redundant servers. If it is a single node deployed then the PaaS is operating in a non-redundant environment and is prone to hardware or software failure. In such deployment models, Application HA may be accomplished where applications and network functions are lighted up across multiple single nodes and communicate with each other via a heartbeat function to detect failures or connect with each other upstream or downstream nodes and are activated either automatically or via orchestration during such failure scenarios.

6.3.2 Partial HA

Partial HA can be defined as high availability under limited failures. In this scenario, the system is designed for optimal costs with minimum components to meet the basic HA requirements. So, under single failure or limited failures the PaaS continues to operate with no degradation in service. Once the threshold is crossed with respect to failure scenarios the PaaS goes into graceful shutdown mode or read only mode of operation

Let us examine the above scenario with an example. For example, a single node is deployed either as part of a distributed cluster or as a remotely managed device. In such scenarios this single node is hosting network functions or applications that operate normally. If the connectivity is lost between this single node and its control plane or management plane but the data path exists for user traffic to flow, then this is a limited failure scenario where the node has lost partial connectivity to the control plane but is otherwise functional. In such scenarios the workloads can continue to function. Following are the requirements that must be supported by the PaaS in such scenarios.

- The compute node MUST continue to be operational and the PaaS MUST continue to provide service to the applications and network functions when connectivity is lost to the PaaS control plane
- The compute node when reconnected back with the control plan MUST NOT evacuate the workloads. The control plane may mark the node tainted or stale until it can authenticate the node and verify the workloads running and then decide based on policy if the nodes needs to be reset or refreshed
- If the node reboots while disconnected from the control plane then the node MUST NOT operate or attempt to bring up the workloads as the node is no longer authenticated and is NOT operating under an environment of trust. This scenario is considered as a double or multiple failure and partial HA covers limited failures only.

6.3.3 Full HA

Full HA is defined as the PaaS is operational in a highly available environment. These systems are designed with lots of redundancy in hardware and software components. The PaaS controllers operate in N+M or quorum mode where it can tolerate (n/2)-1, n>=3 node failures and still be fully functional. The system would be designed in such a way that node, link and application failures are tolerated including multiple failures. Full HA designs are more costly and at the discretion of the user or Telco. The amount of redundancy varies from deployment to deployment based on cost and operational comfort.

In full HA mode, multiple nodes are deployed with control plane and enough compute power to run applications in redundant modes such as replica sets. Traffic and sessions are automatically load balanced across replica instances based on defined policy. (Policy requirements are documented in a different section). Following are the requirements for full HA

- PaaS MUST continue to operate under multiple node/server failures until quorum is maintained. See next requirement.

- When controller quorum is lost, PaaS MUST operate in read only mode where applications that are running must remain up. No new applications can be scheduled in this mode of operation. This controller quorum is applicable to any control plane functions be it SDN, Infrastructure or other application controllers. The choice of deploying custom controllers in kubernetes is dependent on the application and the deployment model and replica sets for those custom controllers are defined as part kubernetes deployment

6.3.4 Reporting Requirements

Since high availability overall is system architecture dependent there are many ways of achieving HA. Telemetry and reporting of infrastructure status is necessary to perform HA operations with respect to workload management.
Following are the reporting requirements wrt to HA operating scenarios.

- PaaS shall provide status of Nodes to managers and orchestrators on a configurable interval - see Telemetry and Management requirements for details
- PaaS shall monitor POD health and log necessary alarms and events
- PaaS shall perform remedial actions such as restarting the applications/network functions Pods in case of nodes or Pod failures based on configured policy
- Any failures - hardware, software, system or application software processes, shall be logged on the compute node and reported per the telemetry architecture to the appropriate collectors and management tools regardless of HA environments - partial or full HA.

6.3.5 Service chaining for HA

Service chaining of functions and applications has an impact on the overall availability of the system. Refer to the following presentation for a discussion on availability calculation. Any software components stringed together in series the availability of the overall system is multiple of the availability of each component - hence the overall availability decreases because any one component in the chain fails the overall chain or service is down. For details on this topic and availability calculations see the attached slides.

https://drive.google.com/file/d/1xonAldqEdnBeHElYoH22CFromTJ5D3W_/view?usp=sharing

6.3.6 Lifecycle Management - Upgrades and patch management

Following are the requirements for PaaS lifecycle management.

- The user/administrator must be able to perform lifecycle management of the PaaS without affecting the operations of the PaaS itself
    - Lifecycle management includes - Software upgrades of components, addition of new hardware to the cluster, enablement of the hardware acceleration on a given node, advertisement of new capability or addition of new functionality.
- Under no circumstances should the entire cluster (control, compute and storage nodes) be required to reboot upon upgrade or downgrade of any software. It is ok to reboot one or few nodes at a time without affecting the existing workloads on the cluster. The node that is required to be reboot must be evacuated first and the workload be operational on another node before it reboots.
- Software patches Shall be applied to the system or other software applications/components without affecting the entire cluster. The upgrade/patch process be handled on a group of nodes at a time or in an availability zone at a time
- The PaaS shall support both upgrades and downgrades of software components from version n to version n-1 and n+1.

6.3.7 Make Before Break principle

Make before break principle implies any time a node or infrastructure element is taken out of service for maintenance the workloads on the node or the infrastructure element must be gracefully moved elsewhere first before this node is taken out of service to ensure minimal disruption to the end user service.

This is a normal mode of operation in a Telco environment where the requirement of end to end service is very high and is subject to penalties when the service degrades. Hence any changes in the infrastructure hardware or software are handled with care to ensure minimal or no impact to the service.

6.4 Proposed Architecture

6.4.1 Existing Solution

All the requirements described above are must have for a Telco PaaS. However, they may all be implemented in the general PaaS. Most of the general PaaS available today support such quorum or active standby controller redundancy model. If the requirements defined in this document are supported by general PaaS then with respect to HA there is no distinction between a Telco PaaS and general PaaS. However, the availability model for general PaaS may be less stringent than a Telco PaaS and hence these requirements defined in this document are necessary and sufficient.

An open source platform like OKD supports a quorum based HA model for PaaS controllers that quorum based HA model fits the requirements of Telco PaaS. Additionally a StarlingX based platform may support and N+M or Active Standby or load shared two controller model for the control plane of the PaaS. That also satisfies the requirements of Telco PaaS. This provides a choice to Telco customers in terms of what model is more convenient to their operational environment and they can choose that mode of operation.

Descriptions of operations of the HA environment and things such as the leader election process for controllers and how these controllers communicate with each other is outside the scope of this document. However, xGVela does not pose any additional requirements to the core HA model of the PaaS other than the requirements that it be supported for the operations in a Telco environment.

6.4.2 High Level Proposed Architecture

With the overall architecture of xGVela the Telco PaaS components sit on top of a general PaaS environment. In that context the High availability of Telco PaaS components is dependent on the general PaaS availability. Hence the general PaaS is required to support the controller HA and Kubernetes or OpenStack environment.

The lifecycle management of those components is outside scope of the HA document but is covered in the overall architecture. The lifecycle of Telco PaaS layer or components in the Telco PaaS impacts the availability of the overall services. So features and functions such as In-Service upgrades of components, make-before-break become necessary to ensure the entire PaaS environment operates in an efficient manner.

The proposed HA architecture for Telco PaaS is the same as that of general PaaS with the following attributes

- General PaaS controllers may also host xGVela specific control functions such as Telemetry collectors, log collectors, API gateways, multi-cluster management capabilities etc. These components are specific to telco usage and may be part of the xGVela layer
- XGVela components - be it telemetry functions, infrastructure management functions, Kubernetes custom resource controllers and definitions, registry components can all run as “applications and services” on the general PaaS
- These components defined above then use the PaaS replica sets properties to instantiate multiple replicas of applications and controllers on the general to deliver HA based services

- Any device profiles, service profiles and user profile configurations consumed by xGVela can be stored in local attached persistent storage and retrieved at will for initial or re-deployment.
- Any compute nodes deployed, local or remote, with general PaaS SHALL be available or useable for xGVela components deployment
- xGVela components must follow the same guidelines as that of the underlying PaaS/cloud native infrastructure for deployment of their functionality unless otherwise explicitly specified