XGVela Technical Architecture

Content:

1. Introduction – why telecom industry needs Cloud Native PaaS

Since NFV architecture was firstly proposed as ETSI standard in 2012, telecommunication industry and telecom operators have started the cloud transformation of network, moving network functions from dedicated physical devices to standard virtualized environment. By 2021, majority of global telecom operators have built Network Cloud, which is private cloud carrying 4G/5G network functions, value-added network functions, network management systems, network orchestration systems, etc. Operator representatives include AT&T, Verizon, Telstra, BT, DT, China Mobile, China Telecom, China Unicom, NTT, etc. Take China Mobile for example, by the end of 2020, China Mobile has built Network Cloud in 8 major districts within China to carry 37 types of network functions including 5GC, IMS, EPC. The proportion of network cloud is up to 75%, which keeps increasing with the construction of edge cloud.

Network Cloud now follows NFV (Network Function Virtualization) architecture (refers to ETSI GS NFV 002). Network functions are implemented as VNFs (Virtualized Network Function), which use virtual machine as virtualized infrastructure and use OpenStack to manage virtual resources.

With the maturity and popularity of 5G network and edge computing, the customer of telecommunication network expands from individual users to enterprise/industry users. The relatively fixed functions and configurations of core network, which mainly used by individual users, now becomes changeable due to diverse vertical industry use cases. Take 5G core slicing as example, use cases of different industries needs different 5G core network. Network slice used by NB-IOT applications, such as smart metering, requires mainly signaling functions and bandwidth only up to several Kbps. While network slice used by VR and video broadcast requires mainly data transmission capabilities and bandwidth varying from 10Mbps to 100Mbps. To better meet customers' requirements, Network Cloud needs to improve its agility, flexibility and reliability.

However, existing Network Cloud has the following problems:

- **Inflexible network function architecture:** VNFs are now developed and delivered in coarse size, which is hard to achieve accurate upgrading, scaling in/out, fault location and etc. And the network functions is not able to support customized design, deployment, configuration to meet diverse requirements from vertical industry.
- **Monotonous capability of network cloud:** current network cloud provides only infrastructure resources, which is not able to support convenient innovation of network functions/application. Using pure infrastructure is complicated and requires developers and operators obtaining high ability. Besides, Telco's network cloud involves multiple network functions and infrastructures from multiple vendors, which are all designed in their own form and shape, and these cross-vendor gaps need to be filled by PaaS above infrastructure.
- **Complex, costly and slow delivery process:** the business processes of existing telecommunication network functions strictly follow the sequence of requirements analysis, design, developing, integration, testing and delivery. Product delivery duration is usually calculated by month. Network requirements of future use cases/ scenarios are changeable. Customer requirements on network may continuously change as their service grow. Traditional development and management processes need to be agile and automatic.

Cloud Native, as the best practice in cloud computing in IT industry, can help network cloud achieve agility, flexibility and reliability. Technologies and concepts such as container, microservice and DevOps can effectively alleviate the above problems. Container, as lightweight, flexible, and commonly used infrastructure can increase agility. Micro service architecture is an effective way in implementing complex software stack. It supports best flexibility in function isolation and evolution. Microservice-type network function supports isolated management, fast customized design/configuration and function combination under different use cases. DevOps can build CI/CD pipeline, and establish continuous feedback during the development and construction process of network cloud, which will improve the end-to-end automation and improve response speed to vertical industry speed. Cloud native is the inevitable direction of network cloud evolution.

Cloud native requires the cloud platform to provide the capabilities required by the applications as much as possible, and the applications use the capabilities provided by the cloud as much as possible so that to grow on the cloud. PaaS, as a bearing platform for required capabilities such as microservice architecture, CI/CD pipeline, network management functionalities and network function reusable modules, is acknowledged as enablement platform of cloud native for network cloud and telecommunication industry.

According to the above analysis, cloud native PaaS, which refers to PaaS providing cloud native capabilities such as containers, microservice, automation tools, and cloud native network function components, is necessary and worth researching in telecommunication industry. Exploring cloud native PaaS in open source community can provide reusable capabilities and reference implementation for the industry.

2. PaaS Condition in Network Cloud

Although it can be predicted that cloud native PaaS can improve the flexibility of network cloud, if looking at current situation of telecommunication industry, it is not clear how to introduce PaaS into network cloud, or what PaaS capabilities are required by network. Most of the network clouds, which have been putting into use, follow NFV architecture and do not take PaaS as an independent system or visible object. These operators and vendors use virtual machines and containers directly as infrastructure, and package all required functional modules and software reliance into VNF delivery image. Figure 1 shows a simple diagram of current VNF condition that only resources are reused among VNFs while VNF related software are dedicated to each VNF.

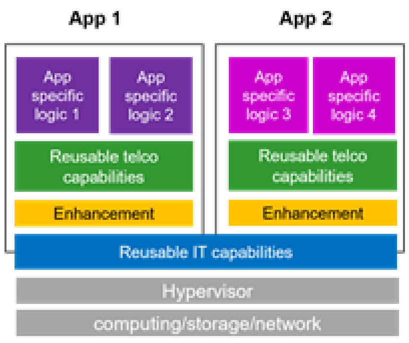


Figure 2-1 VNF Diagram

However, each vendor has its own internal PaaS to support development/maintenance and provide NF (Network Function) required capabilities/modulars; and some operators have designed internal CI/CD pipeline and testing tools. But few of these capabilities are called as PaaS capabilities in network cloud at current stage.

In the field of standardization, ETSI GR NFV-IFA 029 is the representative standard of PaaS for network cloud. It points out the role of PaaS, which includes: 1) eliminating complex operation of NFVI platforms for network function developers and supports auto management/orchestration of NFVI; 2) shielding infrastructure difference of different vendors; 3) providing NFV network service to external customers (e.g. government, bank or enterprises in a vertical industry). ETSI GR NFV-IFA 029 also proposed three potential NFV architectures enhanced with PaaS, but no decision has been made. There is no detailed PaaS capabilities defined in existing standards.

In the field of open source, CNCF provides large amount of cloud native PaaS capabilities that can be used in practical product implementation directly and indirectly (with enhancement). There is no specific indication of which software is necessary or suitable to play as specific PaaS capability in network cloud. As open source can provides reference implementation and is tightly related to practical operation, it is a good start point to explore cloud native PaaS for network cloud.

According to incomplete investigation among industry, standards and open source, possible types of cloud native PaaS capabilities for network cloud are listed as below:

- **PaaS capabilities required to implement NF functions:** This type of PaaS capability is necessary to achieve NF functions and logics, for example database, load balancer, protocol processing capability (PFCP, GTPU…), message bus, possible NF functional modulars, and some infrastructure related capabilities (such as hardware acceleration). User of this type of capabilities is mainly developers. These capabilities could be common among different NFs but unique for different venders/operators.
- **PaaS capabilities required to manage NF functions:** This type of PaaS capability helps to optimize the operation and management of network cloud, but has no direct influence on NF logics. Possible capabilities includes observability capabilities, CI/CD tools, testing tools, FCAPS management tools, etc. User of this type of capabilities is mainly developers and operation staffs. These capabilities can also be used to support NFV management systems such as NFVO, VNFM, EMS, OSS.
- **PaaS capabilities to expose NF service to external customers:** Representative capabilities of this type include bandwidth management capability, user identification capability, mobility management capability, UPF traffic routing function, and other edge computing network functions. User of this type of capabilities is external customers. Currently this type of capabilities is commonly carried and provided by MEC platform.

3. XGVela Overview

According to previous chapters, as the PaaS related standards and research are relatively indolent, it is a good way for telecommunication industry to start with existing open-source PaaS capabilities and build reference implementations. We could explore the enhancement of existing PaaS capabilities in telecom scenarios as well as new PaaS capabilities dedicatedly used telco network cloud. This is the reason that we start XGVela project.

XGVela is a telecom cloud native PaaS platform for 5G and future network cloud. It is targeting on delivering common and reusable PaaS capabilities required in the processes of network function development/running, network cloud management/maintenance, network cloud capability exposure and etc., so that applications are lightweight and contain only code to deliver the intended business logic.

XGVela was firstly launched in April 2020. It joint Linux Foundation Networking as Sandbox project in January 2021. China Mobile, Mavenir, Redhat, Huawei, Sgiscale, Intel, ZTE, STC, Ericsson, China Telecom, China Unicom, Nokia, WindRiver forms the first TSC group. XGVela got its first batch of seed code, which delivers telecom management PaaS functions, from Mavenir in December 2020.

3.1 XGVela High-Level Architecture

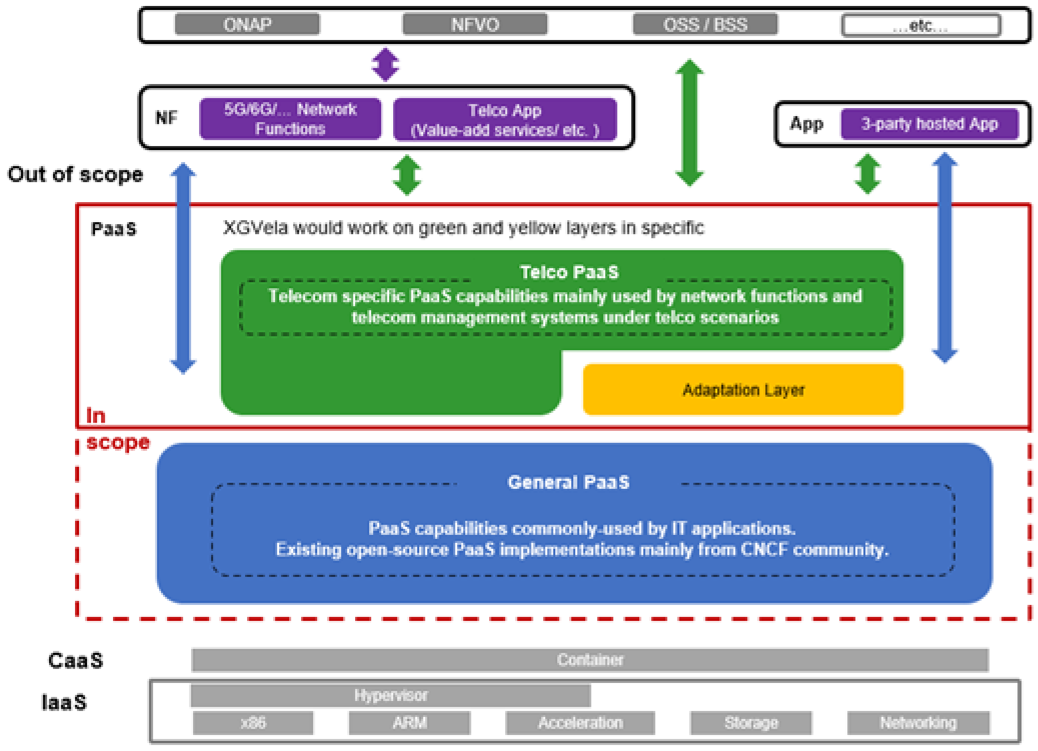

ONAP
NFVO
OSS / BSS
...etc...
NF
5G/6G/... Network Functions
Telco App (Value-add services/ etc.)
App
3-party hosted App
Out of scope
PaaS
XGVela would work on green and yellow layers in specific
Telco PaaS
Telecom specific PaaS capabilities mainly used by network functions and telecom management systems under telco scenarios
Adaptation Layer
In scope
General PaaS
PaaS capabilities commonly-used by IT applications.
Existing open-source PaaS implementations mainly from CNCF community.
CaaS
Container
IaaS
Hypervisor
x86
ARM
Acceleration
Storage
Networking

Figure 3-1 XGVela High-Level Architecture

Figure 3-1 shows the high-level architecture of XGVela.

XGVela, as the cloud native PaaS platform, runs on the container environment by default, and interwork with K8S to realize the orchestration and management of containers. Network Functions (NFs) and applications obtains needed common PaaS capabilities from XGVela. Upper layer telco management systems can select XGVela PaaS capabilities to play as their sub-modules to achieve O&M functions; XGVela PaaS capabilities can also be treated as “platform” resources that support to be orchestrated by management systems.

To distinguish from the existing PaaS implementations in the open-source communities, XGVela divides PaaS capabilities into three categories:

- **Category 1: General PaaS**

  Represented by the blue block in Figure 3-1.

  General PaaS can provide the abstract tools, services and environment required in the process of development, deployment, running, operation and management of applications and their associated services.

  General PaaS capabilities are standard cloud capability which can be offered by any cloud provider, and it will have the capabilities to be shared by a number of industry specific PaaS including Telco PaaS.

  In short, **General PaaS refers to commonly-used PaaS capabilities by all industries**(e.g. service mesh, API GW, LB, observability) **and existing open-source PaaS implementations** (e.g. Istio, envoy, Zookeeper, Grafana). XGVela takes those implementations for integration instead of re-inventing the wheels.

- **Category 2: Adaptation Layer**

  Represented by the yellow block in Figure 3-1.

  Adaptation Layer is unique enhancement of General PaaS capability when applied to telecom scenario. To avoid coupling with General PaaS, the enhancement would be implemented in the form of plug-ins, drivers, and other non-invasive forms.

  In telecom scenario, the General PaaS capability is usually used in combination with corresponding adaptation layer enhancement points. For example, many load balancer software supports HTTP, TCP, UDP, while for 5G network functions, e.g. UPF, protocols like PFCP, GTPU are used in control plane and data plane flow. If a developer wants to use existing open-source load balancer in 5G core, PFCP and GTPU protocol analysis capability should be enhanced for that LB and delivered together with the LB to provide service.

- **Category 3: Telco PaaS**

  Represented by green block in Figure 3-1.

  Telco PaaS focuses on delivering telecom specific PaaS capabilities, which implements telecom features such as multi-tenancy, multiple network plane, network function topology, network function configuration, etc. These capabilities are used to **serve telecom network functions and telecom management systems**. Some Telco PaaS needs to interwork with General PaaS to deliver complete PaaS service.

The above three categories of PaaS capabilities together constitute XGVela, which provides all "platform" services for cloud native Telecom workloads. Developers can make combinations freely of general PAAS, general PAAS + adaptation and telco PAAS based on requirements.

3.2 XGVela Project Scope

- Define PaaS platform architecture, necessary functions / interfaces, processes, common software, etc. for Telecom scenarios. Content will cover general PaaS, adaptation layer and telco PaaS, of which General PaaS capabilities refers to existing implementations.
- Explore requirements of adaptation layer and telco PaaS base on telecom use cases, and implement functionality, interface, etc.
- Build reference implementation of telecom cloud native PaaS platform.

4. XGVela Technical Insight

4.1 Technical Architecture

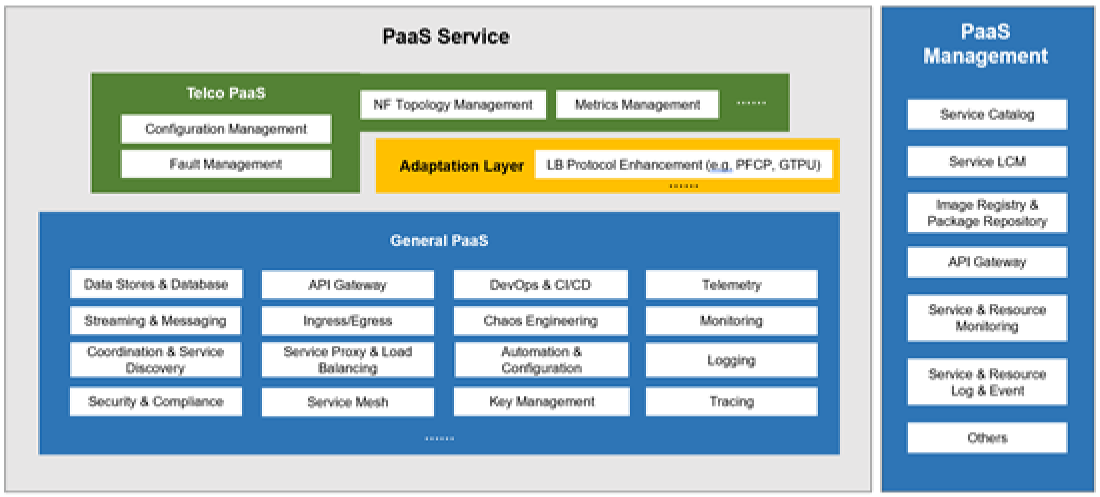


Figure 4-1 XGVela Technical Architecture

XGVela, as PaaS platform for telecom scenarios, its functional framework is basically consistent with most of the commercial PaaS platforms. As shown in figure 4-1, PaaS platform contains two parts: **PaaS Management** and **PaaS Service**.

PaaS Management is responsible for managing PaaS services, which ensures PaaS platform to provide PaaS services to customers. PaaS Management complete the onboarding, orchestration, monitoring and maintenance of PaaS services in the background, make sure that applications and developer/operating stuff can select and use PaaS services, and help to manage operation status of PaaS services. Figure 4-1 shows the most basic functions that PaaS Management should have, which are service catalog, service lifecycle management, Image repository and package repository, API gateway, service & resource monitoring, service & resource log & event. For detailed description and requirements of each function, please refer to chapter 4.2.1

PaaS Service represents the capabilities/functions required by applications, developer and operation stuff, which achieves the core value of PaaS platform. It is managed and orchestrated by PaaS Management. The three types of PaaS capabilities concluded in chapter 2 belong to PaaS Service which are also separated as three categories (General PaaS, General PaaS + Adaptation Layer, Telco PaaS) based on using scenarios. The number of PaaS services keeps increasing with the diverse of use cases. And PaaS services keep upgrading based on customers' needs. Currently, some functions/services have been summarized for different categories. For detailed description and requirements of each service, please refer to chapter 4.2.2 and 4.2.3.

If referring to existing commercial PaaS platform, PaaS Management and General PaaS usually together constitute the PaaS platform of IT industry. Therefore, blue block is used to represent these two together. Corresponding to chapter 3.2, what XGVela is doing is to pick required General PaaS capabilities (blue block) by telco use cases, then find the enhancement point of these General PaaS capabilities as adaptation layer, and explore Telco PaaS capabilities.

4.2 XGVela PaaS Functional Requirements

4.2.1PaaS Management

PaaS Management is responsible for the management of PaaS services. It is required to include at lease the following capabilities:

Service Catalog

Service Catalog is a directory of services, which lists all PaaS services provided by PaaS platform to customers. Specific requirements of service catalog includes:

- It is required that Service Catalog support adding new PaaS service to itself, and deleting, modifying, and querying existing PaaS services. The added PaaS services can exist in multiple forms, which can be local operator or helm package, as well as PaaS services on public cloud that adapted remotely through interfaces.
- It is required that Service Catalog support users/applications/external systems to order PaaS services on demand, and trigger the instantiation and configuration of selected PaaS services.
- It is recommended to achieve Service Catalog with well-designed UI.

Service Lifecycle Management

Service Lifecycle Management (Service LCM) is responsible for managing the lifecycle of PaaS services. Specific requirements include:

- It is required that Service LCM to select images and packages from Image & Package Repository based on user's selection in Service Catalog, and complete instantiation of PaaS services, which are packaged as Operator or Helm Chart. It also requires Service LCM to start, stop, upgrade, delete PaaS services.
- Service LCM uses Kubernetes interface for resource application and orchestration of PaaS services.
- It is required that Service LCM manages scaling in/out of PaaS service instance based on monitoring KPIs and scaling policy set by customers.
- It is recommended to implement UI for service LCM.
- It is required that Service LCM supports access control, which provide unified user management and authentication mechanism, and support RBAC. Users can obtain management permission for PaaS services through role settings.

Image and Package Repository

Image & Package Repository is responsible for the storage and management of PaaS service images and deployment packages. The images and packages can be stored locally in PaaS platform repository, as well was remotely in cloud repositories (GitHub, docker hub, etc.). When user selects PaaS services in Service Catalog and triggers service instantiation process, Service LCM will ask Image & Package Repository for images and packages.

API Gateway

API Gateway is an PaaS service as well as an important PaaS Management function. When using as PaaS Management function, API Gateway is responsible for exposing the service API of PaaS services, and routing user traffic to target processing unit. Specific requirements include:

- It is required that API Gateway to support identity authentication and authorization of API calls, verifying the legitimacy and authority of API calling entity (user, other software, etc.), releasing secure access control, avoiding security threats on PaaS service.
- It is required that API Gateway to support forwarding user request to correct processing unit. The forwarding process supports load balancing, and traffic control actions based on pre-defined traffic management policies, which may include health check, rate limiting, time out & retry, circuit breaker, etc.
- It is recommended that API Gateway to support protocol processing, including HTTP, HTTP2, GRPC, web socket, etc.
- It is required that API Gateway support managing the service API of PaaS services, including API definition (path, parameters, etc.), API publishin g/ suspending/ online/ offline/ withdraw/ etc. This feature is mainly used by developers/operators to provide API-type PaaS services for external usage.
- It is required to support monitoring API usage and API data analysis, which cover performance, usage, alarm, log, etc.

Service & Resource Monitoring

Service & Resource Monitoring is responsible for monitoring PaaS service instances and resources the service used, collecting and displaying monitoring data through well-designed dashboard, and alarm setting. The monitoring KPI and data are determined by PaaS services themselves. This function assists PaaS platform manager and PaaS service consumer to track the status of PaaS services and resources.

Monitoring contents usually include:

- Service-level: running status and performance of PaaS service instance.
- Resource-level: running status and performance of resource used by PaaS Service.

Service & Resource Log & Event

Service & Resource Log & Event is responsible for the log & event management of PaaS services and resource the services used. Log management includes log collection, log storage, log index, etc. Event management records all the changes of PaaS services and related resources. Event usually carries info of event content (what), event object (who), event time (how), event type, event status, etc. This function helps PaaS platform managers and PaaS service users efficiently find, locate, and solve problems.

To conclude, almost every PaaS platform would acquire the above management functions either in a manual way or in an automatic way. **Mature representatives include open source implementation - OpenShift, public cloud - AWS, Azure, Alibaba cloud, Tencent cloud, and many other commercial products.**

4.2.2 General PaaS Requirements

As described in Chapter 3.1, General PaaS represents all PaaS Services that have no industry differences. It can serve a variety of applications that have requirements on common PaaS services. It can also be used as the basis for other industrial PaaS services/platforms.

4.2.2.1 General Requirements on General PaaS

Before detailly analyzing the functions that General PaaS should provide, let's take a look at the general requirements for all General PaaS services:

- General PaaS shall complete service lifecycle management through PaaS Management. There are usually two ways for users to use a PaaS service: call API of API-type PaaS service and create an instance-type PaaS service. API-type PaaS services (like AI service-facial recognition,

image processing, etc.) are managed by *API Gateway* of PaaS Management. Instance-type PaaS Services (like DB/ LB, etc.) are managed by *Ser vice Lifecycle Management*.
- General PaaS services shall be packaged as Operator or Helm Chart. Related image and package can be stored in local Image & Package Repository, as well as in remote repository while pre-configuring automatic access to the remote repo.
- All General PaaS service shall support to general monitoring data, logs, events, alarms, etc., and report to *Service & Resource Monitoring* function and *Service & Resource Log & Event* function in PaaS Management.
- General PaaS services shall support to be deployed on any Kubernetes cased CaaS (Container as a Service) layer.
- General PaaS shall support custom configuration. The configuration parameters are designed by PaaS Service Provider, the configuration contents are provided by users according to application requirements following certain rules.
- It is recommended to select General PaaS software from commonly used CNCF projects.

4.2.2.2 Functional Requirements on General PaaS

The number and type of General PaaS services will keep increasing as more and more use cases and user requirements have been explored. In June 2020, XGVela and Anuket made a joint survey on commonly used General PaaS functions and software. Here we'll provide requirements on mostly used General PaaS functions based on the statistical results.

Data Store/Databases

General PaaS shall provide mainstream relational databases and non-relational databases. The database shall support user management, configuration management and monitoring of itself. Detailed requirements are listed below.

- Data Store/ Database type & software:

Relational DB	Include but not limited to MySQL, MariaDB, PostgreSQL.
Non-relational DB	Include but not limited to MangoDB, Radis, etcd.
Mainstream commercial DB	Include but not limited to Oracle, SQLServer.
Distributed data storage	Include but not limited to Ceph.

- It is recommended database achieve the following monitoring KPIs: number of queries, response time, number of errors, throughput, number of query concurrency, number of tables.
- General PaaS shall support backup of data storage and database manually and automatically. The backup needs to support be downloaded and be used for data recovery.

Streaming & Messaging

General PaaS shall provide Streaming & Messaging functions and software, which can provide services through API or instantiation. Detailed requirements are listed below.

- Commonly used Streaming & Messaging software includes but not limited to AMQP Apache Kafka, Apache Spark, Rabbit MQ, NATS.
- Streaming & Messaging functions provided by General PaaS should support message caching, transmission, replication, distribution, encryption and compression.
- It is recommended that Streaming & Messaging software achieve the following KPIs: total number of messages, message publish rate, message delivery rate, etc.

Service Proxy & Load Balancing

General PaaS shall provide Service Proxy & Load Balancing functions and software. The service can be provided through container instance or service APIs. The Service Proxy & Load Balancing functions should support customized configuration management, monitoring and operation. This function can be used to manage east-west traffic within a K8S cluster, as well as north-south traffic across multiple K8S clusters. Detailed requirements are listed below.

- This functions should dynamically update the status of service backend, including updating service IP address, traffic management like flow forwarding/ flow control/ ACL, etc.
- Commonly used Service Proxy & Load Balancing software includes but not limited to Istio, Contour, MetalLB, NGINX, Envoy, Linkerd.
- It is recommended that this function support statistic and analysis of flows, including number of messages (request/response/error/etc.). the number of sent/received data packets, delay, etc.

Observability

General PaaS shall provide tools to help developers, operators, and users to realize observability of applications running on PaaS platform, which is to obtain metrics/logs/alarms/link status of applications, make these data visualized, so that different group of people can get insight of application running status, locate fault and analyze problems. Observability can help to maintain the stability and reliability of the application. Observability usually includes monitoring, logging, and tracing.

The life cycle of application software and system can be roughly divided into two stages: developing stage and running stage. In developing stage, application software designer/developer should implement the generation of metrics, logs, events, and design the output of these contents according to requirements of General PaaS tools (which includes data format mapping, General PaaS agent integration, etc.). In the running stage, application software will generate metrics, logs, events. These data can be collected by agent and reported to matched General PaaS tool, or General PaaS tool can directly pull data from agent. Data generation and collection at running stage are automatically completed by General PaaS tool.

Detailed requirements are listed below:

- General PaaS shall provide monitoring software:
  - Monitoring software should support metric collection with support to federate and stream metrics across clusters.
  - The collected metrics should support visualization by dashboard.
  - It is recommended that monitoring software can support customized alarm and aggregation rules configuration, and alert subscription.

    - Commonly used monitoring software include Promehteus, Cortex, Thanos, Grafana, Kiali, Zabbix, and Collectd, within which Promehteus and Grafana are most popular. The above software can be used independently or combinedly.
- General PaaS shall provide software for log management, which includes collecting, storing, displaying and other operations of software and system logs.
    - Commonly used logging software include Fluentd, ElasticSearch, Logstash, FluentBit, ELK, among which ElasticSearch, Fluentd and ELK are mostly used.
    - The collected logs should be visualized.
    - Besides collecting, storing, indexing, displaying logs, the log management software can also support log analysis, alarm and alarm subscription.
- General PAAS shall provide tracing functions to record all operations in the whole service life cycle, so that it can provide information for problem analysis and O&M.
    - Commonly used tracing software include Jaeger, OpenTracing, OpenCensus, OpenTelemetry, ZIPkin, among which Jaeger, OpenTelemetry, and Open Tracing are most popular.

DevOps

The lifecycle of software generally includes requirement determination, user experience design, development, testing, deployment, continuous O&M (operation and management). The object of DevOps is to connect these six steps into an automatic workflow, so that developers can only focus on coding and continuously get operation feedbacks, which can finally shorten the product delivery cycle, as well as improve delivery quality.

General PaaS shall provide DevOps tools to help enterprise to build automatic workflow and implement DevOps concepts, which includes:

- **Continuous integration and continuous delivery tools**, which helps to build automatic pipeline of integration, testing, deployment, and upgrade.
- **Project management tools**, which helps developers to create independent workspace to make operations including code change, construction, automatic testing, integration, release, etc.
- **Code management tools**, which provides code repository and code quality management functions. These tools should support maintaining detailed application code changing records, and authorization management of code branch.
- **Automatic testing tools**, which support automatically execute test cases according to user-defined test contents and generate visualized testing results.

Commonly used open-source software include Jenkins, Gitlab, Maven, Argo, Sonar.

Service Mesh

General PaaS shall provide Service Mesh functions and software. Detailed requirements are listed below.

- General PaaS service mesh shall support traffic management (TCP proxying, load balancing, traffic split, mirroring, cricuit breaker, fault injection, filters, external routing, ingress, etc.) Security (mTLS, certificate rotation), proxy injection, CNI plugins. multi cluster support.
- General PaaS service mesh shall support dashboard for visualizing the mesh, various communications, link load conditions, etc.
- Commonly used open-source tools are Linkerd, Consul, Istio, Envoy, among which Istio+Envoy are popular choice.

API Gateway

API Gateway described in this section is General PaaS service, which is mainly responsible for exposing service API of user-developed applications /systems. Requirements of it are basically the same as that of API Gateway as PaaS Management functions.

Commonly used open-source tools are Kong, Tyk, 3-Scale, Istio.

Service Discovery & Registration

General PAAS shall provide Service Discovery & Registration functions and software to help micro services of application/system to obtain each other's access information

- Service Discovery & Registration functions shall maintain real-time microservice access info, which includes adding address of new microservice, update microservice instance address, deleting information of fault microservice, etc.
- Commonly used open -source software include CoreDNSetcdZookeeperNetflixNacos, among which CoreDNS, etcd, Zookeeper are popular choice.

4.2.3 Telco PaaS Requirements

As defined in Chapter 3.1, the key object of Telco PaaS is to implement functions necessary to support telco workloads and procedures in a cloud native environment. It could be used independently or work together with available General PaaS capabilities/functions.

4.2.3.1 General Requirements on Telco PaaS

Like General PaaS, before discussing about detailed functional requirements, there are common requirements applicable to all Telco PaaS functions.

- Telco PaaS shall implement functions necessary to support Telco workloads and procedures in cloud native environment.
- Telco PaaS shall not duplicate the functions already available in General PaaS layer rather than interface, enhance or adapt General PaaS functions where available.
- Telco PaaS shall support Telco standards compliant information and object model.
- Telco PaaS shall support Telco standards compliant NBIs:
    - NetConf for configuration management
    - VES for event notification
    - TM Forum OpenAPIs
    - 3GPP5G Service Based Interfaces (SBI)
    - NFV orchestration – Or-Vnfm-EM/Vnf???
- It is recommended Telco PaaS to use CNF packaging model and align with NFV.

The following are requirements similar to that of General PaaS (detailed description can refer to Chapter 4.2.2.1).

- Telco PaaS shall be deployed on any Kubernetes based CaaS and General PaaS distributions that might be instrumented on Cloud or bare metal.
- Telco PaaS shall complete service lifecycle management through PaaS Management.
- Telco PaaS services shall be packaged as Operator or Helm Chart, stored in Image & Package repository.
- Telco PaaS services shall generate metrics, log, alarm, event and report these data to PaaS Management.
- Telco PaaS shall support custom configuration.

4.2.3.2 Possible Functional Solution for Telco PaaS

Till now, during the development of telecom network functions and related systems, it is able to summarize the General PaaS capabilities/functions, while the type and number of Telco PaaS used is not clear. This doesn't mean that Telco PaaS has not been used, but because there is no open-source reference implementation of Telco PaaS. Therefore, in this chapter, common Telco PaaS functions will be summarized based on the development and management experience. The number and category of Telco PaaS functions will keep increasing as more use cases are explored.

In Chapter 2, three possible types of cloud native PaaS capabilities in network cloud have been concluded, which are PaaS capability required to implement NF functions, PaaS capability required to manage NF functions, PaaS capability to expose NF service to external customers. Among the three types, PaaS capability required to manage NF functions is bypass, which has little impact on the business logic and functions of the NF, and is a good choice for initial exploration. Therefore, XGVela starts from these type of PaaS capability, and explores the management-related Telco PaaS functions.

Management-related Telco PaaS functions 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 generalize certain other Telco specific functions such as subscriber tracing, lawful intercept (LI), call data records (CDR), etc.

- Topology Management Service: 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 Service: 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 Service: 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 Service: 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 Service: 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.

4.2.3.3 Telco PaaS Solution contributed by Mavenir MTCIL

The first batch of Telco PaaS functions of XGVela come from the seed code contributed by Mavenir company's product "MTCIL", which mainly provides management telco PAAS capabilities. Related functions will be introduced below.

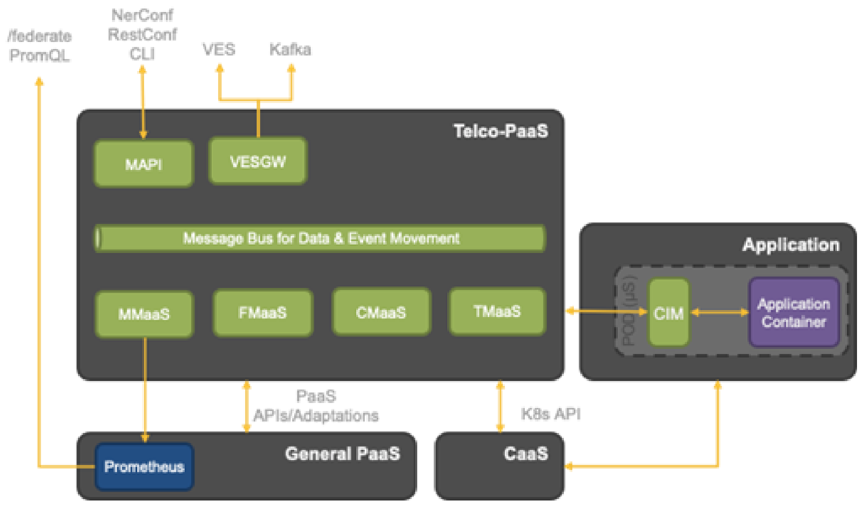


Figure 4-2 Architecture of Mavenir Contribution

Figure 4-2 shows the functional architecture of Mavenir contribution. In the figure, Telco PaaS functions include CMaaS (Configuration Management as a Service), TMaaS (Topology Management as a Service), FMaaS (Fault Management as a Service); And two auxiliary management capabilities, VESGW (ONAP VES Gateway) and CIM (CNF Interface Module).

CIM

CNF interface module, a most important assistant component of Telco-PaaS, provides a single integration point and API for NFs or 3rd party applications. It has the following features:

- 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.

- It is the local agent for other management Telco PaaS capabilities to manage NFs and get NF information.

CMaaS

Configuration Management as a Service manages the configuration of network functions and application, and can be treated as configuration management center for CNF (Cloud Native VNF). It has the following features:

- Exposes NetConf NBI for orchestration and management systems like ONAP to push configuration for NFs/application and related microservices.
- Yang is supported as data model of configuration.
- Translates configuration in Yang model and NetConf protocol into simple JSON/REST, and push translated configuration to NF/application containers.
- Supports two methods to update Day-2 configuration:
    - One method is to deliver configuration via K8S rolling update. This method requires NF/application pod to restart and re-read the ConfigMap, and is mostly adopted by IT applications and stateless applications.
    - The other method is direct API calls to NF/application container via etcd and CIM per application need. This method supports NF /application pod to update configuration without restart. As the structure and configuration of telecom network functions are complex, it is generally do configuration after the deployment of network functions, and the network functions should maintain stable running state. Therefore, this method is more suitable for telecom configuration.

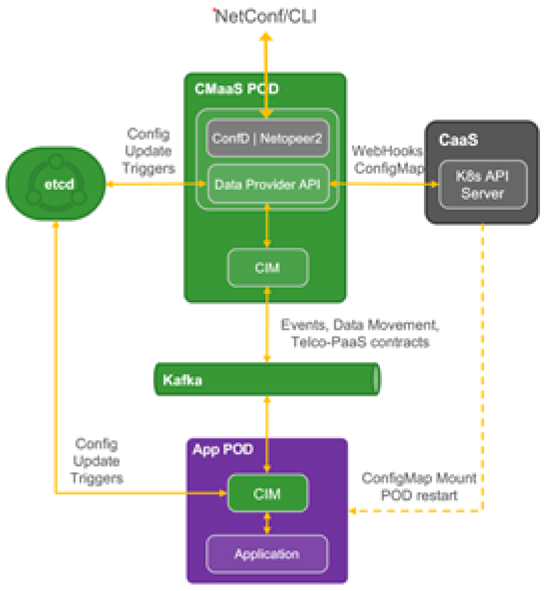


Figure 4-3 CMaaS Diagram

Figure 4-3 shows the working principle of CMaaS. Firstly, CMaaS pod will receive the configuration sent by management system (such as ONAP) through NBI, which is formatted in Yang model and transmitted through NETCONF protocol; Then the CMaaS module will parse the configuration and store it in etcd in the form of key-value pair, where CMaaS will maintain the version of the configuration; After that, CMaaS will notify the CIM module pre-integrated in NF / application that the new configuration needs to be processed, and trigger the application to update the configuration.

TMaaS

Topology Management as a Service constructs a complete 3GPP network function topology, which combines the network function topology, microservice topology as resource pod topology together. This function helps K8S gain NF-level topology. It supports the following features

- Interfaces with K8S for auto-discovery of services, which will also get resource topology from K8S showing the relationship among deploymet /statefulset, services, pod, and containers.
- Builds 3GPP model, and constructs NF topology of NF, NF microservices and NF microservice instances. It can also combine this NF topology with resource topology.
- Exposes REST and also interfaces with CMaaS to expose Topology data over NetConf.

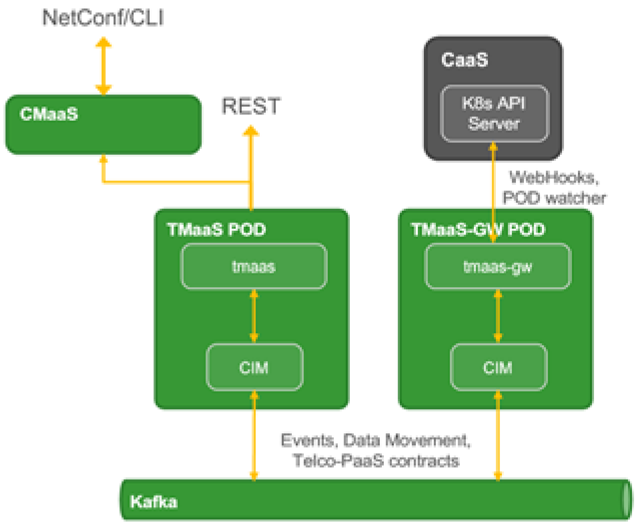


Figure 4-4 TMaaS Diagram

Figure 4-4 shows the basic working principle of TMaaS. In the figure, TMaaS GW pod directly obtains deployment, service, pod and other information from k8s to form the topology and resource topology of micro service instances. The TMaaS pod predefines the NF topology relationship of the network element and the network element micro service, and this topology relationship will be marked in the deployment / statefulset or other files through annotation. TMaaS GW will sent resource topology to TMaaS. TMaaS will combine the resource topology and predefined NF topology through annotation, and output the merged topology through NBI, which is under definition.

TMaaS uses united 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.

TMaaS follows 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).

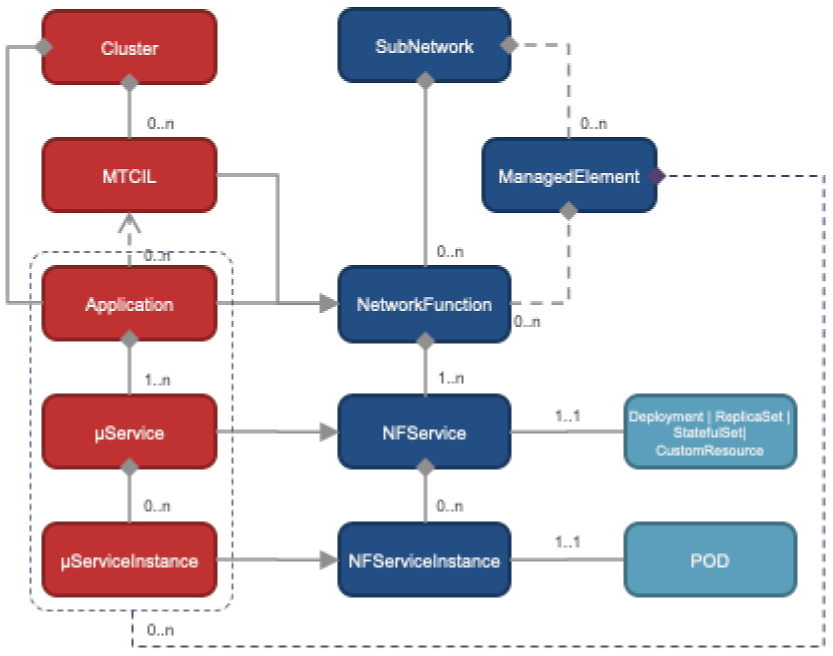


Figure 4-5 Management Model followed by NF to deploy on XGVela

FMaaS

Fault Management as a Service is a telecom event management module, which collects events from the system, translate them into NES format, and report to management systems for information and analysis. It has the following features:

- Implements 3GPP compliant fault and alarm management model.
- Provides interfaces for application µService to publish and subscribe to various events.
- Interfaces with the metrics management system (Prometheus) for TCA events.
- Exposed VES compliant NBI for notifications.

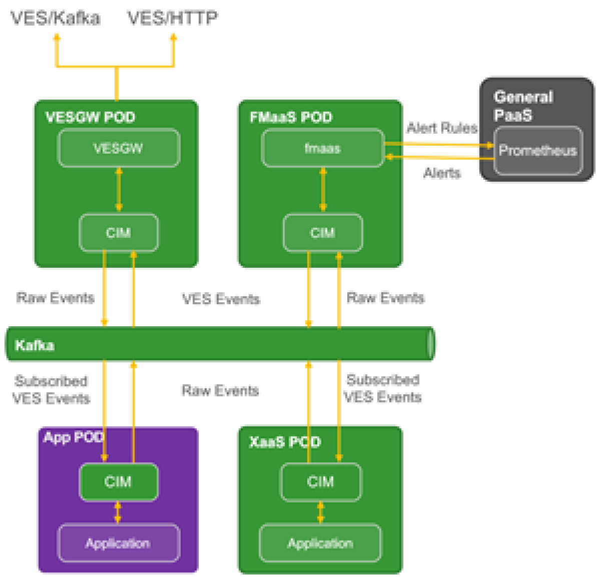


Figure 4-6 FMaaS Diagram

Figure 4-6 shows the basic working principle of FMaaS. FMaaS collects events from Prometheus, application pod, XMaaS pod, translate them into VES format. After the translation, the event can be pushed to management systems (like ONAP and VES collector) through VES GW. During this process, FMaaS can also send alert rules to Prometheus.

4.2.3.4 Telco PaaS Solution Proposed by Intel

Network is the most important and complex infrastructure resource in cloud computing. Compared with IT applications, telecom network functions have more complex requirements on network. Telecom network functions generally need multiple network planes, and each network plane is used to carry traffic flow. For example, in telecom Core Cloud, network functions are usually designed with three network planes, which are control network plane, data /user network plane, storage network plane, to separately carry management traffic flow, data plane traffic flow, and storage traffic flow; In telecom Edge Cloud, network functions are at least designed with two network planes, including data/user network plane (used for high-speed user plane data forwarding on edge side) and merged network plane for management and storage (few flow on these two plane so that merged to save resource).

From the perspective of infrastructure, multiple network plane is generally realized by implementing multiple network interfaces/cards to network functions instance, and each network interface/card is assigned to a network plane to forward relative traffic flows. For virtual machine, multiple vNICs can be setup for multiple network planes. For container, it supposed to the same solution as that of virtual machine.

However, 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. For this problem, there is an open-source solution, Multus, that can solve it. Multus is a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods. With Multus, you can create a multi-homed pod that has multiple interfaces. Multus CNI has already support many network CNIs, including Calico, Flannel, Userspace CNI (ovs-dpdk/vpp), OVN-Multi CNI, SRIOV-NIC CNI and SMartNIC CNI. Developers can specify different network CNI for different network planes to realize container multi network planes. For example, control network plane can chose traditional CNIs with lower performance like Calico, Flannel; while data/user network plane can chose CNIs with better forwarding performance, for example SRIOV-NIC CNI.

According to above contents, we can see that Multus ensures pod to implement multiple virtual network interface/card and implement multiple network plane for containers at infrastructure level. However, for developers of network functions, they still need to understand the working principles and using methods of different CNIs. The difficulty of network development has not been reduced. Therefore, based on Multus solution, a Telco PaaS function named NMaaS is proposed in this Chapter.

NMaaS, **Network Management as a Service**, is proposed to expose the service of container multiple network planes. It has the following features:

- Exposes NB APIs for Orchestration/Management systems or developers to configure the Infrastructure NIC, add/delete interface to NF at runtime, SRIOV configuration etc.
- Shields the different using methods of different CNIs and provides consistent user experience on NIC management. For example, developers can simply specify the number of vNICs and SLA requirements on these vNICs to complete container vNIC configuration.
- Supports customized value settings for vNIC parameters such as latency, jitter, bandwidth, throughput, performance, and trigger CaaS layer to setup the required underlaying driver/software based on these requirements.

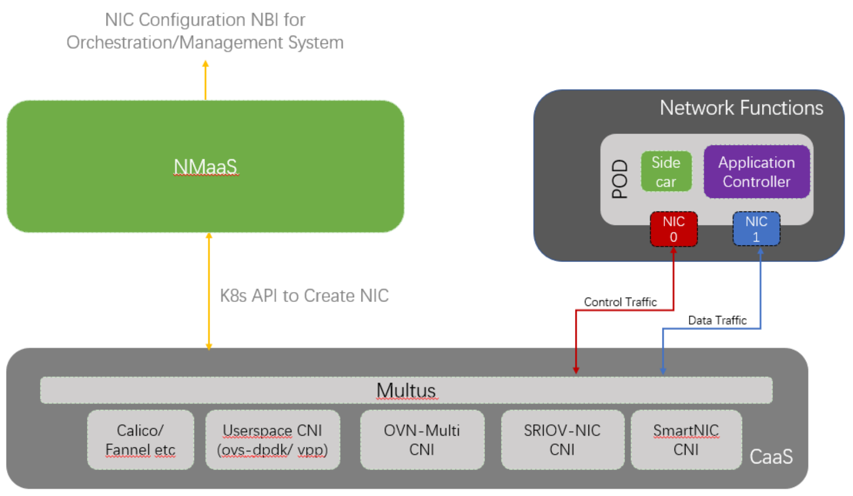


Figure 4-7 Proposed NMaaS Diagram

Figure 4-7 shows the basic working principle of NMaaS. NMaaS pod receives a vNIC request from orchestration/management systems or command line through standard NMaaS NBI. NMaaS will convert vNIC request into contents that can be understand by CaaS layer, including vNIC type, amount, configutation, etc. And the converted information will be sent to Multus to setup required underlaying driver/software. Then, NMaaS will trigger K8S create new vNIC for target Pod, and update the traffic and vNIC configuration in Pod.

4.2.3.5 Telco PaaS Solution Proposed by China Mobile

For telecommunication industry, network functions are designed following a paradigm and in distributed architecture. No matter it is PNF, VNF or CNf, each NF is desgined in distributed architecture with multiple modules, which typically includes interfaces module (which is in charge of south-north communication with other NFs), business processing modules (which deals with NF business logics), data storage module (which stores NF data), and operation andf management module (which is in charge of NF operation and management). Within all these functional modules, operation and management module (OAM module), as it has similar functions to different NFs, can be shared as common module.

The common operation and management functions used by NF usually include but not limited to: configuration management, heartbeat management, performance management, log management, alarm management. If checking those open-sourced network function projects, such as OAI and Free5GC, it is shown that operation and management functions are non-business-logic related functions, which are usually not included in network functions source code. However, these OAM functions are necessary in NF productions, which usually are developed by vendors or operators. Besides, according to cloud native application design principles, observability is one of the most important feature an application should have. With these OAM functions, NF can achieve observability easily.

As OAM functions are common, reusable, non-business-log-related and necessary functions to NF, they can be treated as PaaS capabilities provided by cloud service provider to NF developers. Using platform provided OAM fucntions can help the NF developers focue more on core business design while be reliefed from repetitive OAM implementation. So cloud native OAM, which is a group of operation and management functions for NFs, can be treated as Telco PaaS capabilities.

Typical functions of cloud native OAM:

- **NF configuration management function:** this function manages configurations of NFs. It supports to accept NF configuration command from orchestration systems, manage current and historical configurations of NF, and configuring NF. As NF's configurations are usually stored in configuration files and has vendor-different formats, the management target for this function is only configuration file without defining the file contents. For some newly developed NF strictly following cloud native principle and using Kubernetes/Docker as resources, their configurations can be stored directly in etcd as configmaps.
- **NF heartbeat management function:** this function helps to monitors the NF's health status. This function has different levels of implementation. The easiest-level implementation is collecting NF-level heartbeat outside NF. This relies on the NF itself to generate heartbeat within each microservice instances, monitoring internal heartbeats, determine NF health status internally with overall microservice heartbeat data. The heartbeat collected on this level can only reflect whether the NF is normal or not, while not knowing whether it is healthy inside. The middle-level implementation is collecting NF microservice-level heartbeat. This relies on each NF microservice to report its heartbeat, with which the function can determine whether the NF is healthy based on some pre-defined rules. The hardest-level implementation, which is also the most cloud native one, is monitors the health status of pod through kubernetes liveness probe, and analyze microservices-level and NF-level health status sutomatically. As current NF are mostly self-contained, the easiest-level implementation is mostly used, while the middle-level implementation and hardest-level implementation are prefered to be followed in the future.
- **NF performance management function:** this function helps to monitors the metrics of NF. It supports to collect the metrics data and do analysis based on pre-defined rules. Prometheus is the most commonly selected software.
- **NF log management function:** this function helps to collect and store logs of NF.
- **NF alarm management function**: this functions helps to collect alarms of NF both at NF-level and resource-level. It also supports to clean the alarm data.

For more design and interface details, please go to:

- https://github.com/XGVela/cloud-native-OAM/blob/main/docs/cloud_native_oam_design.md
- https://github.com/XGVela/cloud-native-OAM/blob/main/docs/API_guide-northbound_interfaces.md

4.2.4 XGVela PaaS Workflow

After summarizing the technical architecture and functional requirements, XGVela related workflows will be covered in this Chapter. These workflows are applicable to all PaaS platforms, which includes XGVela. In this Chapter, we only cover the simple and general workflows that have no telecom features, while the interaction with NFVO, VNFM and other telecom systems will be considered in future release.

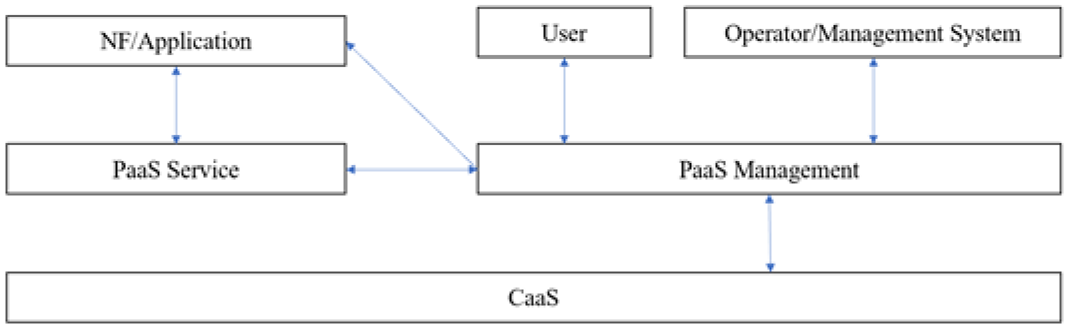


Figure 4-8 PaaS Platform Workflow

We simplify the PaaS technical architecture and its relationship with outer management systems as blocks in figure 4-7, which includes NF/application, User, Operator/Management systems, PaaS Service, PaaS Management, and CaaS. The major workflow are listed below.

Service Order Process

Service Order Process ensure PaaS services to be ordered by Users. Firstly, User will browse all PaaS Services through Service Catalog in PaaS Management and select required PaaS Service. Then, User will configure user-defined parameters of the selected PaaS Service, such as interface, performance parameter, reliability parameter, service model (dedicated or shared), etc.

Service Instantiation Process

Service Instantiation Process instantiates PaaS Service. After ordering required PaaS Service, the User will trigger the service instantiation. The Service Lifecycle Management in PaaS Management will instantiate selected PaaS Service. For shared PaaS Service, instantiation means selecting existing and running PaaS service on PaaS platform, completing configuration including RBAC/access/etc., and return access API to User. For dedicated PaaS Service, instantiation means selecting images/packages/descriptors in Image & Package Repository of PaaS Management, sending these files together with user-defined configurations to CaaS, and triggering K8S to complete PaaS Service instantiation. The CaaS layer will reply access API and monitoring data of PaaS Service to PaaS Management.

Service Unsubscribe Process

Service Unsubscribe Process lets Users to unsubscribe the ordered and instantiated PaaS Service through PaaS Management. For shared PaaS Service, PaaS Management will stop PaaS Service for Users through changing the PaaS Service configuration like RBAC. For dedicated PaaS Service, PaaS Management will delete PaaS Service instance on CaaS.

Service Configuration Updating Process

Service Configuration Updating Process can help User/Operator/ Management Systems to update configuration of PaaS Service through configuration center in PaaS Management. The Service LCM and CaaS will execute new configuration on Service and Pod.

Adding Service Process

Adding Service Process lets Operators to add new PaaS Service to PaaS platform. The process includes uploading images and packages of new PaaS Service into Image & Package Repository, onboarding PaaS Service in Service Catalog, setting the basic configuration (including configuration items and parameters) and user-defined configuration items.

Deleting Service Process

Deleting Service Process lets Operators to offline and remove PaaS Service on PaaS platform. PaaS Management will firstly check the usage condition of PaaS Service selected to be deleted. PaaS Service in use cannot be deleted. If the PaaS Service is not used, PaaS Management will then remove it from Service Catalog, and deleting related images and packages in Image & Package Repository.

Service Operation and Maintenance Process

Service & Resource Monitoring/Log/Event in PaaS Management achieve health check, monitoring, log management, alarm management of PaaS Service through the interfaces with PaaS Service instance.

The O&M interface, data model, data generation, etc. should be pre-developed based on PaaS Management requirements during design and development stage of PaaS Service.

5. High Availability

5.1 Overview of High Availability

High availability is usually 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.

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 scheme. 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-service 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).

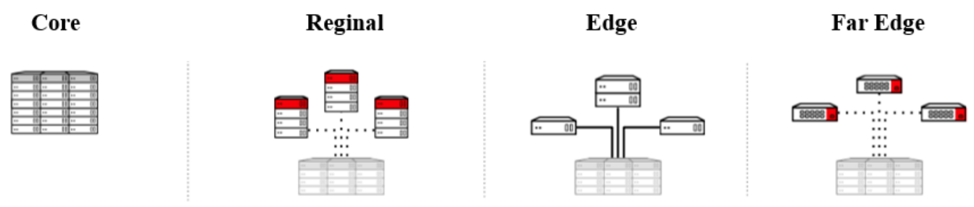
5.2 Telco High Availability Requirements on PaaS

Telecom applications usually requires reliability up to 99.999%. And this high reliability should be achieved by the overall network cloud, whose major components usually include hardware, virtualization layer, PaaS layer, application layer and management systems. All these components construct a chain. Any 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. So before considering the overall reliability, it is necessary to ensure that each component maintains high reliability. As PaaS is an important component for network cloud, for the following contents, we’ll introduce about Telco High Availability requirements on PaaS, which is applicable to XGVela and all other PaaS platforms.

5.2.1 Network Cloud Deployment Scenarios

When talking about network cloud, we usually are talking about a distributed cloud with multiple sites spreading all over the country. Knowing the deployment scenarios of network cloud can help to know the potential HA requirement of each component in network cloud, especially PaaS.

Figure 5-1 describes typical deployment scenarios of telecom network cloud. The telco network cloud usually can be separated into the following type: core cloud, reginal cloud, edge cloud and far edge nodes. The features of each deployment scenario are displayed in figure 5-1.



	Core	Reginal	Edge	Far Edge
Number of Site (Order of Magnitude)	10	100	1000	10000
Number of nodes per site (Order of Magnitude)	1000+	100+	10+	1+
NF carried	Telco Core NF (2/3/4/5G), value-added NF	Lightweight telco core NF, value-added NF, vRAN	User plane of telco NF, vRAN	vRAN
Type of applications carried	IT	IoT, AI/ML surveillance, FinTech, connected vehicles, …	IoT, customer premise equip., Fin Tech, …	IoT, customer premise equip., Fin Tech, connected vehicles, …

Figure 5-1 Typical Deployment Scenarios of Network Cloud

As each deployment scenario has different resource, carries different applications, and may have different architectures, they will have different HA requirements. For example, as core cloud carried the most importance telco core NFs and has sufficient resources, the HA level should be high, while edge mage has lower HA level due to lack of enough resources.

5.2.2 Deployment of PaaS in Network Cloud

XGVela PaaS can be deployed in the core cloud, reginal cloud, as well as edge cloud and far edge nodes. The locations may include data centers, central office, metro or reginal POPs, enterprise CPE, ruggedized IOT gateways etc.

According to PaaS technical architecture described in Chapter 4, PaaS usually includes PaaS Management, which is necessary for PaaS platform to provide PaaS service to users, and PaaS Service, which are a group of service required by users on demand. The resources occupied by PaaS Management is relatively stable as only management related functions are deployed. Only one function in PaaS Management may requires changeable resources – storage capacity of Image and Package Repository, which may increase as more PaaS services are integrated onto PaaS platform. However, the amount of resource used by PaaS Service vary. It depends on user requirements and may vary from no resource occupation (which refers to no PaaS service ordered by user) to very large amount. The influence factors of resources used by PaaS Service at least include the number of PaaS service instantiated on PaaS platform, and the resource configuration of each PaaS service.

As different deployment scenarios in figure 5-1 have different amount of resource, PaaS would be deployed differently in each deployment scenarios.

- Core Cloud & Reginal Cloud usually have sufficient resources for PaaS deployment. In these two deployment scenarios, PaaS Management is fully deployed; PaaS Management and PaaS Service will consider high-level of reliability.
- Edge Cloud usually has limited resources, so PaaS Management is not required to be deployed. For those edge clouds connected to core cloud or reginal cloud, edge cloud can use the PaaS Management in core/reginal cloud remotely. For those independent edge cloud, for example an enterprise edge cloud, PaaS Management can be deployed in a cropped version and can be mixed deployed with other application instances.
- Far Edge only has few nodes, so all resources should be saved to applications and required PaaS service while PaaS Management should be deployed in remote core cloud or reginal cloud.

5.2.3 Infrastructure HA Categories

As infrastructure is the bottom layer of whole network cloud, PaaS, applications, management systems are all deployed on infrastructure, the HA condition of infrastructure will influence the HA strategy of all the other components in network cloud. So, before analyzing the HA requirement of PaaS, it is necessary to look at the different HA solutions of infrastructure.

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

- **No HA**: No HA means there is no redundancy of infrastructure. For this category, the high availability for services and applications can be achieved through stateful applications/services that can reconnect to different upstream or downstream devices and applications/services 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.

5.2.4 PaaS HA Requirements

5.2.4.1 General HA Requirements on XGVela PaaS

Before analyzing HA requirements for XGVela PaaS on different infrastructure HA categories, there are the following general HA requirements:

- 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 PaaS Management. This implies there are n>=3 PaaS Management instances where n is odd such that (n/2)+1 PaaS Management instances are required to be operational for full quorum.
- The PaaS SHALL support Active-Standby redundancy model for PaaS Management. If a quorum-based model is not supported then PaaS Management shall support an active standby model at a minimum. Active-Standby implies one management instance 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 PaaS 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.
- PaaS shall perform remedial actions such as restarting the service Pods if service not working properly, or service pod fails.

5.2.4.2 HA Requirements on PaaS Under No HA scenario

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.

5.2.4.3 HA Requirements on PaaS Under Partial HA scenario

Partial HA can be defined as high availability under limited failures. In this scenario, such as the edge cloud, 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. Following are the requirements that must be supported by the PaaS in such scenario.

- The PaaS Service MUST continue to be operational and continue to provide service to the applications and network functions when connectivity is lost to the PaaS Management.
- The PaaS service instance when reconnected back with the PaaS Management MUST NOT kill the PaaS service or recreate a new one. The PaaS Management may mark the service tainted or stale until it can authenticate the service and verify the running status of service itself and related applications, and then decide based on policy if the service needs to be reset or refreshed.

5.2.4.4 HA Requirements on PaaS Under Full HA scenario

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 Management components 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, platform and application failures are tolerated including multiple failures. Full HA designs are more costly and at the discretion of Telco. The amount of redundancy varies from deployment to deployment based on cost and operational comfort.

In full HA scenario, resources of multiple nodes are used to deploy PaaS Management, and the PaaS services can be deployed in redundant modes such as replica sets to ensure high availability. Traffic and sessions are automatically load balanced across replica PaaS service instances based on defined policy. Following are the requirements must be supported by the PaaS in Full HA scenario.

- PaaS MUST continue to operate under multiple node/server failures until quorum is maintained.
- When PaaS Management quorum is lost, PaaS MUST operate in read only mode where PaaS services that are running must remain up and continue to provide service to applications. No new PaaS Service can be scheduled in this mode of operation.

5.2.5 Existing Solutions

All the above requirements are applicable to PaaS platform, which includes both General PaaS and Telco 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 Management 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.

5.2.6 Proposed HA Architecture for Telco PaaS

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 HA and Kubernetes or OpenStack environment.

The lifecycle management of those components is outside scope of the HA chapter 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 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 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 Telco PaaS layer.
- XGVela 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.
- Telco PaaS components must follow the same guidelines as that of the underlying PaaS/cloud native infrastructure for deployment of their functionality unless otherwise explicitly specified.