Building an efficient private telco cloud with Red Hat OpenShift on Red Hat OpenStack Platform

Red Hat, Inc. and KDDI CORPORATION

Table of contents

1. English Version

1.1 Abstract and Introduction

> **This white paper is released as an early access version**
>
> This white paper is released as an early access version, so there may be cases where the content referenced or announced in the white paper is not included in the current white paper content.
>
> Red Hat and KDDI are preparing to release the white paper with all content included by the fourth quarter of 2024.

1.1.1 Abstract

This white paper provides best practices for the "Shift on Stack" configuration, which integrates Red Hat OpenShift Container Platform into the Red Hat OpenStack Platform within a telecommunications cloud.

The technological landscape surrounding telecommunications operators is undergoing significant changes, especially since the introduction of virtualization with network functions virtualization (NFV) and more recently with containerization with cloud-native network functions (CNF). With the advent of 5G Core network (5GC) defined in 3GPP Release 16 and beyond, Service-based Architecture (SBA) has been specified, accelerating the movement towards cloud-native architectures among telecommunications operators.

In this white paper, we discuss the challenges faced by telecommunications operators due to the complexity of integration and the short product lifecycle within this cloud-native paradigm. We then illustrate how leveraging Red Hat OpenStack Platform for private cloud in conjunction with Red Hat OpenShift, which enables multi-cloud and hybrid cloud environments, can address and resolve these challenges. Furthermore, we introduce best practices on the effective use of these products, particularly focusing on considerations and tuning within the "Shift on Stack" configuration, utilizing OpenShift on OpenStack. We also compare this approach to utilizing OpenShift on bare metal, another option within the telecommunications cloud. Lastly, we present a case study showcasing the utilization of these solutions in a real-world network environment, specifically highlighting the case of KDDI, one of Japan's leading telecommunications operators.

It's worth noting that this white paper targets both telecommunications operators adopting the telecom cloud and telecom application vendors providing applications for these telecom clouds.

1.1.2 Introduction

The trend towards openness and commoditization in telecommunications continues unabated. Digital switches, once operated on dedicated machines, chips, operating system (OS), and applications, transitioned in the 2000s to Advanced TCA hardware and COTS servers with general-purpose CPUs, and dedicated OS evolved into carrier-grade Linux, leveraging Linux technologies. In the 2010s, Enterprise Linux distributions like Red Hat Enterprise Linux became the standard in telecom core networks. Furthermore, the emergence of virtualization and cloud computing, coupled with the rise of public clouds, led to the birth of OpenStack as software for building private clouds, widely adopted by telecommunications operators. In the early 2020s, new technologies like containers, and container orchestration software such as Kubernetes, enabled telecommunication operators to deploy cloud-native technologies within their core networks, facilitating automated operations and rapid application deployment.

However, telecommunications software and operations may not always keep pace with rapid technological changes, especially considering the unique business environment for telecom operators. The societal impact of communication outages, which affect not only direct customers but also citizens reliant on communication-based services, underscores the criticality of robust telecom infrastructure, highlighting the significance of terms like "carrier-grade" and "telco-grade".

In addition to business considerations, telecom software and operations entail specialized requirements beyond typical enterprise needs. Collaboration between operators and telecom application vendors is essential for building complex mobile core networks capable of serving millions of subscribers seamlessly. Standardization efforts, such as those by 3GPP, are indispensable for ensuring interoperability and backward compatibility, necessitating continued support for older protocols and interfaces.

Coexistence of legacy and modern systems remains a critical theme for operators, particularly for established players with long histories. Despite the advent of 5G, many operators still support 3G and even 2G services, demanding cost-effective operation alongside efforts to

integrate and migrate legacy systems. Moreover, telecom applications must handle vast traffic volumes, maintaining performance even during network congestion, requiring rigorous testing and performance optimization.

In this complex and interconnected landscape of mobile core networks, ensuring stable operations amidst frequent system interactions poses significant challenges. While software and hardware migrations are crucial for maintaining stable communication services, the execution of these migrations must also be stable and reliable, necessitating robust automation and rollback mechanisms, especially given the stateful nature of operations within mobile core networks.

To effectively leverage the hybrid cloud and cloud-native capabilities of Red Hat OpenShift in the telecommunications industry, grounded discussions considering both current challenges and historical contexts are essential.

This white paper combines the concepts of cloud computing with Red Hat OpenStack Platform and hybrid cloud with Red Hat OpenShift to offer a solution to these practical operator challenges.

1.2 Background

1.2.1 Virtualization and NFV

Network function virtualization (NFV) is a way to virtualize network services that have traditionally run on proprietary hardware. These services are packaged as VMs on commodity hardware, which allows service providers to run their network on standard servers instead of proprietary ones. NFV improves scalability and agility by allowing service providers to deliver new network services and applications on demand, without requiring additional hardware resources.

Key components within an NFV architecture are virtual network functions (VNFs), which service providers can flexibly run across different servers or move around as needed when demand changes. This flexibility lets service providers deliver services and apps faster. For example, if a customer requests a new network function, they can spin up a new VM to handle that request. If the function is no longer needed, the VM can be decommissioned. This can also be a low-risk way to test the value of a potential new service.

VNFs are built on top of a virtual infrastructure manager (VIM) that abstracts the infrastructure to allocate compute, storage, and networking resources efficiently among the VNFs. The framework for managing NFVI and provisioning new VNFs occurs in the management, automation, and network orchestration (MANO) elements defined by NFV.

More Details

- Understanding virtualization
- What is NFV?

1.2.2 Telco Cloud

Telco cloud is a software-defined, highly resilient cloud infrastructure that allows telecommunications service providers (telcos) to add services more quickly, respond faster to changes in network demand, and manage central and decentralized resources more efficiently. It is one of the key foundational components in a successful digital transformation.

Enterprise clouds run internal, administrative functions and can have customer-facing portals - all delivered in a mix of public, private, and hybrid configurations. Traditionally, telco clouds are focused on running more restrictive network functions and essential business applications that require much higher levels of observability, control, fault tolerance, and availability.

While greenfield operators (those who build infrastructure where none existed before) can build a completely cloud-native environment from the ground up, established operators must build their telco cloud to work with legacy network environments. Legacy and cloud-native networks need to coexist for some period, allowing operators to migrate network functions, services, and applications in a way that makes the most sense for their organization.

When migrating to a cloud-native architecture, a holistic approach is key. The migration might be implemented one network function or service at a time, but the process should start with a comprehensive cloud readiness assessment that encompasses infrastructure, applications and service portfolios, organization, and processes.

More Details

- What is telco cloud?

1.2.3 Containerization and Microservice

With the advent of 5G, telco clouds will utilize newer technologies like containers and microservices, as well as hybrid cloud architectures. The adoption of CNFs can resolve some of the limitations of VNFs by moving many of these functions into containers. Containerization of network functions makes it possible to manage how and where the functions run in the environment.

CNFs are not just the containerization of network functions. To get the full benefit of cloud-native principles requires further rearchitecting of network function software, like decomposing it into microservices, allowing multiple versions during updates, and using available platform services like generic load-balancers or datastores.

As more service providers adopt cloud-native environments, CNFs must co-exist with legacy VNFs during the transition. Service providers must fully automate the development, deployment, maintenance, and operation of the network to effectively handle escalating demand, accelerate deployments, and reduce complexity. Proven methodologies for configuration and deployment, tools matured in open source communities, and rigorous testing and certification are more critical for service providers than ever.

The telco cloud is evolving to rely on an open, horizontal, hybrid approach to cloud infrastructure that increases flexibility and vendor independence. The industry is recognizing that an open, horizontal approach rather than a vertically integrated stack?leads to faster innovation, improved performance, increased agility, and significantly reduces the total cost of ownership.

More Details

- Understanding cloud-native applications

1.2.4 Issues on Fragmentation

Historically, network equipment providers (NEP) have supplied both software and hardware to telecom operators, with their software running on their own hardware. In the cloud-native world, they sell only software, which runs on hardware platform provided by operators or third parties. While the separation of software from hardware brings many advantages to both NEPs and operators, it also introduces some challenges. NEPs needs to ensure their software runs properly and maintains guaranteed performance on various paltforms. This may result in additional cost to NEPs due to required verification testing and potential modification on each platform.

The approach toward cloud-native networks varies among operators and currently appear to be fragmented. This fragmentation incurs extra costs for both NEPs and operators.

We present this white paper to provide one of the best practices for utilizing Red Hat OpenShift Container Platform and Red Hat OpenStack Platform for NFs in operators' networks. We anticipate that this activity will help suppress fragmentation and exploit the advantages from private cloud solutions.

1.3 "Shift on Stack" Solution Overview

1.3.1 Solution Overview

Challenges of Telecom Operators Toward Cloud-Native

Continuing to utilize these cloud-native technologies within telecom applications and telecom clouds is important, and there is no doubt about the value that can be obtained by doing so.

However, as of 2024, this transformation may still be only halfway through.

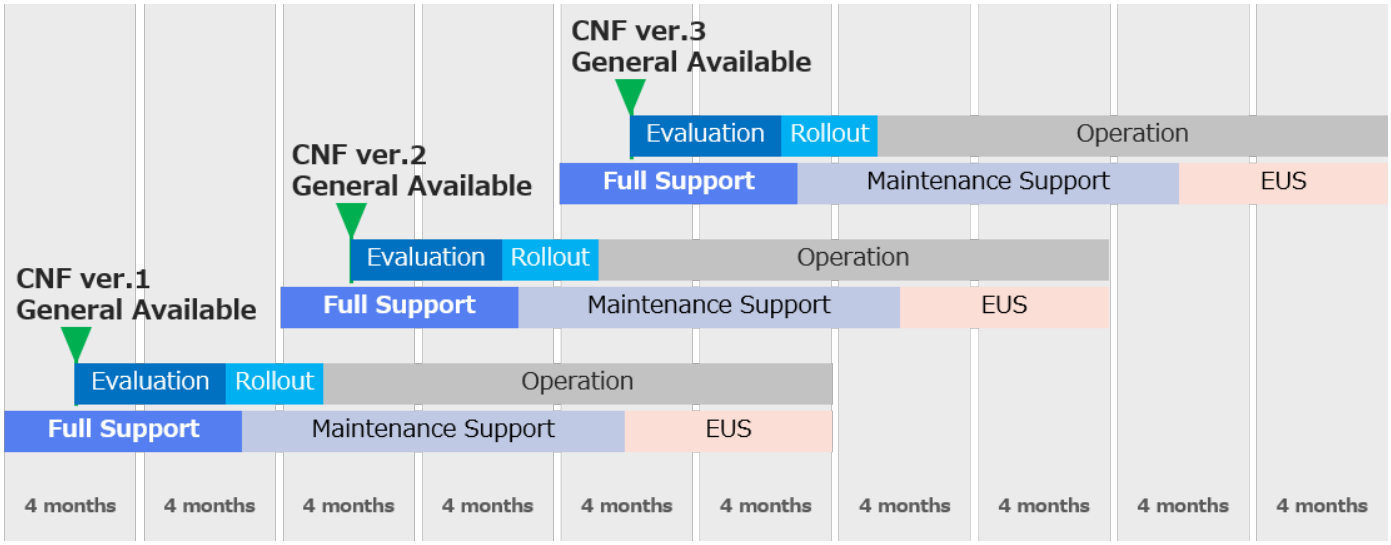
In particular, telecom applications have quickly transitioned from physical appliances to virtualization through NFV in a short period of time, and then adopted cloud-native technologies based on container technologies centered on Kubernetes. At the same time, the core architecture has changed from 4G to 5G. Thus, there has been a drastic paradigm shift in technology from the 2010s to the present.

In the midst of these rapid paradigm shifts, both telecom application vendors and telecom operators are required to experiment and adapt quickly. The telecommunications industry requires very high availability, and the communication protocols used cannot always be replaced freely. Communication protocols need to be retained for compatibility with older communication standards. In addition, there are many assets accumulated from the past regarding these communication protocols and highly reliable redundancy designs. Therefore, it is necessary from an economic rationality perspective for telecom application vendors to reuse past assets even for new generation software such as 5G.

As a result, the use of communication protocols such as Diameter and GTP, which use SCTP and UDP, continues. In Kubernetes, implementations of containers that are aware of VLANs and IP addresses, such as multi-CNI using Multus and macvlan/ipvlan, are being carried out. This allows CNF, which requires immediate switching to standby at the application layer, to continue using traditional implementation methods and concepts, such as L2 redundancy methods using VRRP.

In order to utilize cloud-native technologies in such a situation, operators need to overcome the following specific challenges.

SHORT LIFECYCLE / FREQUENT UPGRADE



Example life cycle between OpenShift and CNF

Kubernetes releases a new major version three times a year, with 14 months of updates provided. Based on this, Red Hat offers the OpenShift Container Platform, achieving up to 24 months of support. However, this is short compared to the long-term support at the enterprise level and the 4-year support provided by the Red Hat OpenStack Platform.

This short lifecycle facilitates the addition of new features and the abolition of old ones, accelerating the turnover of software. While this is effective for environments and content providers where change is intense, it poses challenges in the communications field where stable infrastructure is required.

Mobile systems undergo major changes about every 10 years, and updates are made carefully to avoid service interruptions. Therefore, adapting to the short lifecycle of Kubernetes and updating the software version every two years is a significant challenge for telecom application vendors and telecom operators.

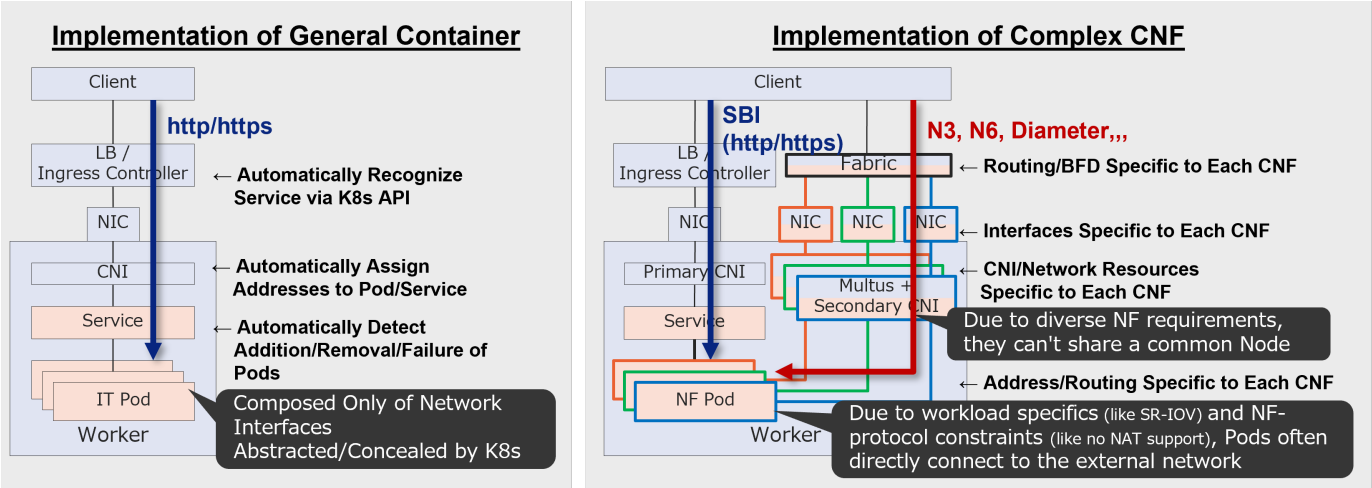
TIGHT COUPLING BETWEEN KUBERNETES AND CNF VERSIONS

Upgrading software versions to cope with Kubernetes' short lifecycle is a significant challenge. Furthermore, network operators demand high performance and availability requirements from telecom applications. To meet these, telecom application vendors need to tune Kubernetes and the operating system.

Telecom application vendors thoroughly verify CNF software in their labs and require the replication of these settings in the operator environment. However, this poses a challenge in constructing a core network with multiple vendors, as the Kubernetes or OpenShift versions and parameters used vary by vendor and product.

Therefore, to standardize the container platform layer and construct a core network with multiple vendors, it is necessary to absorb the differences in versions and settings for each CNF and align the interfaces. However, realizing this requires substantial energy, and it is a practical challenge that not all operators can adopt.

COMPLEX NETWORK REQUIREMENTS



Compared to general containers, CNFs require more complex implementation

The network requirements of Cloud-Native Network Functions (CNFs) used in core networks significantly differ from those of typical web applications. Particularly, CNFs that require rapid application redundancy switching employ L2 redundancy such as VRRP, or L3 redundancy combined with BGP and BFD. Also, multiple IP addresses may be necessary to utilize SCTP's multihoming.

In the case of data plane CNFs, SR-IOV and DPDK might be used. Utilizing these technologies requires handling PCI-E devices directly within the CNF container.

Given these characteristics and operational requirements of telecom applications, when using Kubernetes in telecom, a single Pod needs to be able to use multiple networks, such as VLAN and SR-IOV, in addition to the primary CNI. However, as of now, Kubernetes only manages addresses as IPAM and performs service discovery for the primary CNI.

Therefore, to meet the complex network requirements of telecom applications and telecom operators, other automated or manual operations are necessary.

CNF AND VNF MIXED WORKLOAD

New generation network functions are increasingly being deployed as Cloud-Native Network Functions (CNFs) using cloud-native technologies. However, many network functions, such as those based on older standards or those with little change, are still not being converted into CNFs. This is because containerization requires not just the use of container technology, but also the transformation of the system architecture to a cloud-native one, and converting monolithic applications into microservices is not an easy task. Therefore, for these less dynamic traditional network functions, it may not be necessary to fully adopt cloud-native technologies. Instead, it's important to continue providing services with minimal effort and cost. Consequently, telecom operator platforms are required to accommodate both CNFs and Virtual Network Functions (VNFs).

MORE DETAILS

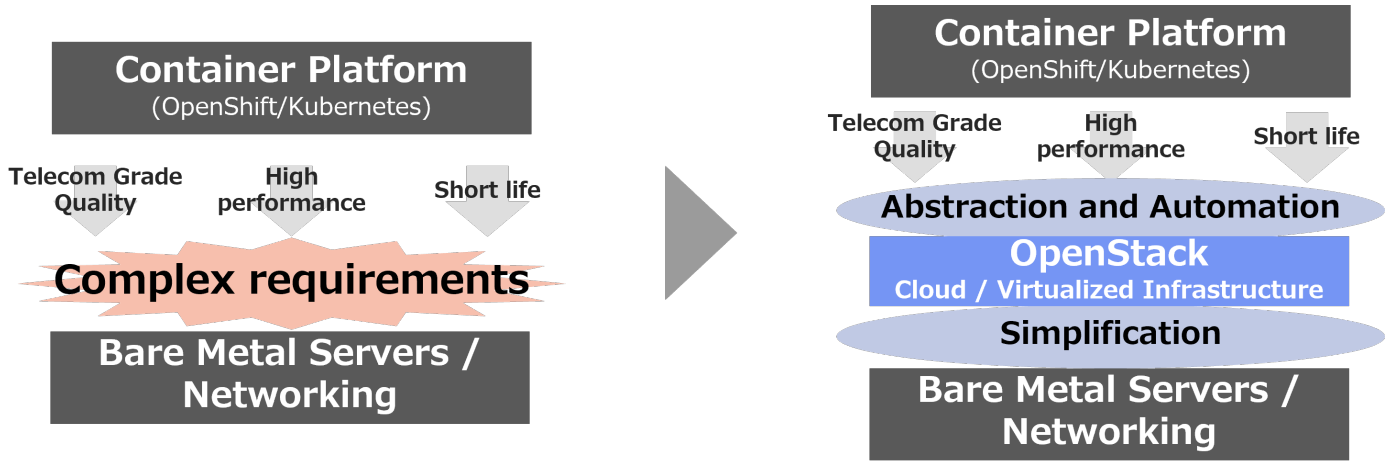
For more detailed information regarding these challenges, please refer to Appendix A.

Solving Our Challenges with "Shift on Stack"

From the next chapter onwards, we will discuss the details of these challenges. We will demonstrate the effectiveness of the solution, "Openshift on OpenStack (Shift on Stack[1])", which combines OpenStack as a private cloud and Kubernetes as a cloud-native platform.

Basic Concept and Strategy

MAXIMIZING THE VALUE OF OPENSHIFT/KUBERNETES WITH OPENSTACK



Benefits of inserting an abstraction layer through OpenStack

In general, having OpenStack reside between OpenShift and bare metal servers can introduce various overheads. These overheads include OpenStack services, hypervisor QEMU/KVM processing, software required to run as a hypervisor such as Open vSwitch, and controller nodes that provide the OpenStack database and APIs. However, these overheads do not merely create waste; they also facilitate a higher rate of bare metal utilization (maximizing return on investment), high degree of automation and simplify management by abstracting the hardware and physical networking layers.

By leveraging the benefits of OpenStack, such as abstraction, we can address the challenges that Telco operators face.

In this white paper elucidates how value can be maximized through the integration of virtualization and private clouds with Red Hat OpenStack Platform, alongside cloud-native platforms utilizing OpenShift.

Using Shift on Stack yields the following benefits.

EFFICIENT INTEGRATION

NEP vendors offering NFs can choose the optimal Kubernetes/OpenShift version and surrounding ecosystem for each NF individually. Additionally, by leveraging virtualization, they can freely select hardware configurations and network setups. This enables them to replicate configurations close to their own validation environments on operators' actual networks without being constrained by specifications imposed by other vendors' NFs or being influenced by hardware and network configurations specific to operator environments.

SIMPLE OPERATIONS

The greatest advantage of Kubernetes/OpenShift lies in the ability to focus on application operation. Therefore, simplifying the construction and upgrade of Kubernetes/OpenShift clusters becomes extremely important as it does not provide intrinsic value from an application perspective. Creating and deleting server and network resources on demand is the most crucial and effective aspect of cloud infrastructure. The existence of infrastructure that allows Kubernetes/OpenShift to be easily rebuilt and reconfigured enables the operation of Kubernetes/OpenShift clusters in an immutable manner, similar to container applications. Consequently, these operations can be greatly simplified.

PROMOTING AUTOMATION

OpenStack enables all operations on the infrastructure via a RESTful API. Moreover, utilizing these APIs, infrastructure automation tools such as OpenStack CLI, OpenStack Heat, Ansible, and Terraform/OpenTofu can automate the management of infrastructure components beyond virtual servers and primary networks for Kubernetes/OpenShift clusters. These components may include secondary networks, DNS records, load balancers, container registries, bastion servers, monitoring servers, and more, which are sometimes required externally by Kubernetes/OpenShift. OpenStack can address these challenges with its services, templates, virtualization including networking and others.

RESOLVING THE ABOVE CHALLENGES WITH SHIFT ON STACK

No.1: Short lifecycle / Frequent upgrade

- OpenStack APIs for virtual servers, virtual storage, and virtual networking enhance the automation of OpenShift setup using Infrastructure as Code (IaC) tools.
- OpenStack's infrastructure abstraction streamlines operations such as the creation and deletion of OpenShift clusters. This makes it much easier to deploy and support different versions of OpenShift clusters as required by each tenant, as ease of life-cycle management of the OpenShift clusters easily redeploying the newer cluster versions and suspending and/or removing the of older version clusters once the change is complete.
- By handling persistent data from OpenShift clusters, OpenStack facilitates resource lifecycle management of persistent and stateful resources (such as Persistent Volumes), simplifying the process of replacing OpenShift clusters.
- OpenStack can offer abstract and consistent infrastructure resources for OpenShift deployments such as compute and networking.
- Furthermore, hardware abstraction via virtualization conceals the presence of physical hardware from virtual machine instances.

This enables the lifecycle of the actual physical hardware to be decoupled from the lifecycle of the virtual machine, except for features like SR-IOV, which utilize PCIe pass-through capability.

No.2: Tight coupling between Kubernetes and CNF version

- With infrastructure isolation and abstract delivery via OpenStack, network application vendors offering Cloud Native Functions (CNF) can deploy their own validated OpenShift versions and configurations in the operator's environment, mitigating concerns about conflicts with other vendors.

No.3: Complex network requirements

- The on-demand creation of independent virtual networks for each tenant is facilitated by Neutron's virtual networking capabilities within OpenStack.
- Neutron's capacity to create and segregate virtual networks ensures the necessary network isolation for each Cloud Native Function (CNF) and OpenShift cluster.
- All Neutron networking can be fully automated through the API.

No.4: CNF and VNF mixed workload

- OpenStack's management capabilities for virtual machines are robust.
- Given OpenStack's comprehensive and proven functionality in managing Virtual Network Functions (VNFs), leveraging this capability enables treating the OpenShift cluster for CNFs akin to a virtual machine.

---

1. In this white paper, we may refer to OpenShift on OpenStack as "Shift on Stack" for brevity. ↩

1.3.2 Efficient Integration

Hardware Abstraction

Generally, the lifecycle of hardware is longer than that of Kubernetes, but they do not necessarily align. Also, to enhance the network to cope with increasing traffic, it is necessary to expand server resources according to demand. However, it is conceivable to procure different generations of hardware depending on the procurement timing. Also, you may want to select better hardware each time, so you may want to change the hardware product by conducting an RFP.

In such situations, if you use bare metal servers directly, Kubernetes/OpenShift and CNF may need to tune each setting or change the design to match these servers, depending on factors such as which side of the NUMA Node the fast NIC PCIe card used is located.

Virtualization by OpenStack can hide this from CNF. This allows you to reduce the verification man-hours from the CNF perspective, such as dealing with hardware replacement and heterogeneous hardware configurations, by using abstracted paravirtualized devices.

Flexible Infrastructure Selection Using Virtualization

Generally, by using Kubernetes/OpenShift, software can be freed from the construction of its operating environment and the management of network connectivity, allowing it to focus more on application development and operation.

However, in operating NF for mobile core networks, it is not enough to just have the vanilla Kubernetes/OpenShift configuration. For these NFs, which require high reliability and high performance, Kubernetes is often required to absorb complex network requirements and tune CNF-specific parameters. Furthermore, these requirements may vary depending on the vendor providing the CNF and the type of CNF. Therefore, it is a significant challenge to absorb these different requirements in Kubernetes on bare metal servers.

Therefore, by utilizing virtualization, you can divide the large bare metal server into clusters divided according to different Kubernetes/OpenShift requirements for each CNF, optimize each cluster for the CNF it carries, and deploy it, allowing you to flexibly choose the Kubernetes configuration.

Absorption of Environmental Differences through Abstraction

Also, the abstraction of server resources, network resources, storage resources, etc. through virtualization can absorb and hide environmental differences.

For example, by using paravirtualized devices such as virtio, you can provide virtual hardware with high-performance software implementation that does not depend on the type of hardware used to the Kubernetes/OpenShift controller node and worker node.

Furthermore, you can match detailed parameters such as the number of vCPUs, the capacity of memory, the number of NICs connected to the worker node, the capacity and number of storage, etc. in each environment.

This allows you to match the NF vendor's verification environment with the operator's production environment and different operator environments, and because it is hardware-independent, you can also reduce the verification patterns when using different hardware.

Economic Benefits

By preparing the optimal Kubernetes/OpenShift application operating environment for each CNF in this way, you can generate the following economic benefits:

- You can shorten the hardware verification man-hours by hiding the hardware replacement and the selection of heterogeneous hardware through virtualization.
- By preparing the optimal integrated Kubernetes/OpenShift for CNF, you can save time and resources for CNF integration.
- You can match the lab environment of the NF vendor and the production environment.

1.3.3 Simple Operations

Unified Operation of CNFs and VNFs

Even with the advent of CNFs, there remains numerous use cases where virtual machines are necessary. This necessity extends not only to NF but also to peripheral equipment such as O&M nodes and OSS/BSS in telecommunications carriers. There are also NF implementations that run on virtual machines, not on containers or Kubernetes. Additionally, many legacy nodes, such as those for 4G and earlier or voice service, continue to operate as VNF.

In such situations where there is a need for virtual machines, you can unify the operation of the infrastructure for both CNFs and VNFs by introducing an OpenStack layer on top of hardware. This also makes it easier to allocate hardware resources between CNF and VNF.

Furthermore, for large-scale VM-based VNFs, there are operations such as server redundancy with HA (high availability) function due to hardware abstraction, as well as live migration at the time of hardware failure. Therefore, network operators can unify the handling of these hardware between CNF/containers and VNF/VMs by using the Shift on Stack configuration.

Definition of the Decomposition Point of Responsibility through Resource Partitioning

There are some limitations to using Kubernetes/OpenShift in a multi-vendor configuration. Generally, partitioning using Namespace is used to take multi-tenancy in Kubernetes. However, there are some limitations to this method.

The first issue relates to a conflict in security and configuration. For CNFs, where high availability and performance are required, there may be a need to specifically tune the Linux Kernel parameters of the operating system where the container operates. To achieve higher performance, there may also be a need to directly leverage hardware technologies such as SR-IOV. In these cases, privileged user rights in Kubernetes might be required, and there is a possibility that the changes in the Linux Kernel parameters used by each CNF vendor may conflict.

The second issue is the division of network resources. Kubernetes has a feature called CNI that dynamically configures networks and services, etc., without being aware of the network infrastructure, but these networks are different from technologies that strictly divide networks, such as virtual networks provided in public cloud VPCs or OpenStack Neutron. Therefore, it is not possible to divide or connect the network for each container request, and the network connecting the container to the outside must be treated flatly.

Against these challenges, by introducing OpenStack as IaaS and actively dividing Kubernetes/OpenShift clusters, responsibility decomposition points can be clearly decomposed.

Simplification of Kubernetes/OpenShift Cluster Upgrade

In the Kubernetes upstream, a new major version is released three times a year, and Red Hat OpenShift Container Platform, a downstream distribution of Kubernetes, follows this. OpenShift enhances this upstream support period, providing 6 months of full support and 12 months of maintenance support, and realizing up to 24 months of support for specific versions as Extended Update Support (EUS).

In this way, Kubernetes needs to be regularly upgraded to new versions, and at the same time, CNF must also follow these Kubernetes versions and upgrade the CNF version. For network operators who are required to provide communication without interruption, it is important in operation to be able to perform these regularly occurring version upgrades with high quality and quickly.

However, in-place upgrading the OpenShift/Kubernetes cluster is very challenging as it requires considering the mixed state of CNF and OpenShift versions during the upgrade process and adjusting the timing of container restarts.

Also, fundamentally, Kubernetes platforms such as OpenShift don't support version downgrades today. Therefore, if a problem occurs during the upgrade, it will also be difficult to revert the cluster back to its original version.

In contrast, the Shift on Stack configuration makes it easier to realize simple version upgrades by facilitating blue-green updates through automation. Generally, creating or deleting virtual servers is simpler than creating or deleting bare metal servers. Since you can use OpenStack, a system that has already been completed, there is no need to create a new bare metal server provisioning system or network orchestrator.

By launching a new version of the OpenShift cluster and a new version of CNF, and switching on the network side, you can realize a very simple version upgrade.

1.3.4 Promoting Automation

Cloud Computing with OpenStack

The essence of the value provided by OpenStack is not just the provision of virtualization. OpenStack is cloud infrastructure software for realizing cloud computing. This allows users to use the resources they need through the RESRful API provided by the cloud infrastructure, not just hardware abstraction through virtualization.

By using these APIs, resources such as virtual machines and virtual networks can be managed by software. This allows complex network requirements needed for network functions to be software-defined using various tools such as Red Hat Ansible Automation Platform, OpenStack Heat, and other automation software.

In particular, these automations are powerfully effective in the following use cases in Telco Cloud.

Automation of Infrastructure Resource Management with IaC

With OpenStack, server resources, network resources, storage resources, and other server information can be defined through OpenStack's APIs. This allows not only the configuration information of CNF (such as Helm and Helm Values) to be manages in Git, but also the configuration information about server, network, and storage resources for Kubernetes/OpenShift clusters to be managed in Git using OpenStack Heat, Ansible, Terraform/OpenTofu, and so on.

Automation of Infrastructure Deployment/Scaling

Infrastructure managed by IaC can easily create copies of the same configuration by changing its parameters. Generally, mobile core networks are deployed on a large scale and multiple times, and are expanded to meet increasing demand.

Containerized CNFs following various Kubernetes practices can easily deploy and scale their network functions. However, this also requires Kubernetes/OpenShift worker nodes to operate the CNF. If you have cloud infrastructure like OpenStack, you can automate the deployment and scaling of these OpenShift clusters.

1.4 Practice of "Shift on Stack"

1.4.1 Technical Overview

In this chapter, we will introduce the practical aspects of OpenShift on OpenStack.

Using OpenShift on top of OpenStack is a long-supported and common configuration in OpenShift. However, in the telecommunications industry, where bare metal OpenShift is gaining attention, there are not many examples of building communication applications with Shift on Stack.

In this white paper, we will describe considerations and precautions for using communication applications.

Consideration from Multiple Perspectives

In practicing Shift on Stack, it is necessary to consider both the perspective of OpenStack, which is the infrastructure, and OpenShift, which is the cloud-native platform.

From the OpenStack perspective, the first thing needed is a general perspective as Network Function Virtualization (NFV). Virtual machine-based NFV systems, called Virtual Network Functions (VNFs), are very popular in terms of tuning and setting perspectives for running VNFs. In addition to this, OpenStack also needs to care about the features of OpenShift/Kubernetes.

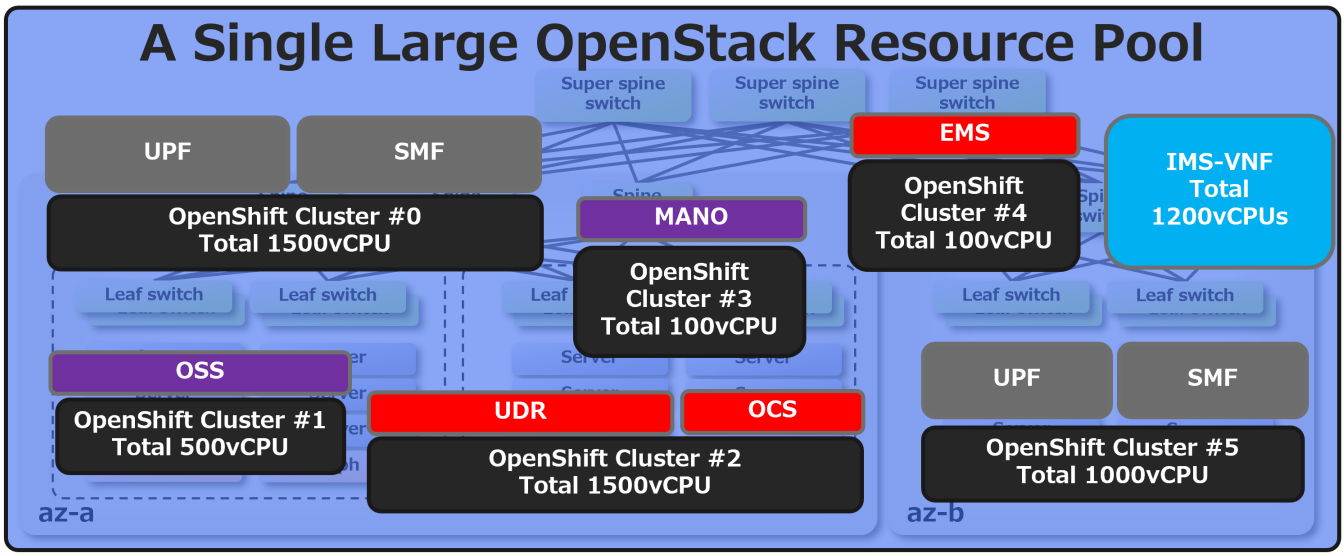
Kubernetes is not just about providing various functions on its own, but it is very good at combining various plugins according to the environment. To make it easy to introduce various plugins, interfaces called Container Network Interface (CNI) for networks and Container Storage Interface (CSI) for storage are defined, and various external components can be used. There is a collection of plugins called Cloud Provider OpenStack that makes the most of OpenStack's features, and you will use this. At this time, it is necessary to consider the characteristics of the communication application, which is CNF, and to consider from the perspective of which components should be prepared on the OpenStack side in addition to when VNF is used.

Of course, there are also tuning and considerations for handling communication workloads in OpenShift. The tuning from the perspective of OpenShift/Kubernetes is basically the same as bare metal OpenShift, so you can use this knowledge as it is. However, there are some considerations for running CNF workloads on OpenShift on OpenStack. For example, in the case of normal workloads, IPI (Installer Provisioned Infrastructure) installation is used, but in the case of CNF workloads that require a large number of network divisions, it may be necessary to choose UPI (User Provisioned Infrastructure). In addition, there is a perspective to effectively use very large bare metal servers in bare metal, but a different way of thinking is required on OpenStack where resources can be cut out by virtualization. Furthermore, in bare metal servers, management such as division of SR-IOV NICs needs to be done by OpenShift, but in Shift on Stack, SR-IOV management is done by OpenStack.

The items mentioned above are just examples, but in this way, it is necessary to consider from the perspective of OpenStack, the perspective of OpenShift, and the perspective of the parts where each intersects.

Basic Strategy

LARGE OPENSTACK CLUSTER AND OPENSTACK MULTI-TENANT OPERATION



Partitioning a Large OpenStack Resource Pool

Creating a dedicated OpenStack cluster for one OpenShift cluster cannot be said to be an effective measure.

As already introduced, the benefits of using OpenStack are as follows:

- Efficient integration
- Simple operation
- Promotion of automation

In order to unify the infrastructure operations of telecommunications carriers, including not only CNFs but also VNFs and IT workloads, and to reduce operating costs, it is necessary to scale the OpenStack cluster to a certain size. Therefore, it is first necessary to realize a unified and consistent private cloud with OpenStack by operating it in multi-tenant, and the efficient deployment and operation of CNFs by OpenShift is based on this foundation.

REALIZE BY DIVIDING THE DEPLOYMENT OF OPENSHIFT CLUSTERS TO ABSORB THE DEMAND DIFFERENCES FOR EACH CNF

As mentioned earlier, the benefits of virtualization are hardware abstraction and the realization of automation by IaC. The hardware requirements and network requirements required by CNFs differ depending on the CNF or CNF vendor. One of the things that OpenStack is most good at is absorbing these differences and flexibly assigning resources. Taking advantage of the advantage of easy resource division and flexible resource assignment, we will divide the OpenShift cluster according to the requirements.

DIVIDE THE RESPONSIBILITY RANGE OF OPENSTACK AND OPENSHIFT

OpenStack and OpenShift clearly separate their roles.

OpenShift/Kubernetes also has functions to configure the infrastructure, but it is effective to clearly define which of these functions to enable in OpenStack and OpenShift.

Details will be described later, but in general, the functions are divided as follows.

Category	Setting	OpenStack	OpenShift	Description
Computing	CPU Pining	V	V	CPU pinning should be done on both OpenStack and OpenShift
Computing	Hugepage	V	V	Hugepage should be provided ob both GuestVM and container
Computing	NUMA management	V	-	OpenShift uses single NUMA isolated for each virtual machine
Computing	Reserve HA Resources	V	-	OpenStack provides compute resources for HA
Networking	High Speed NW	V	-	High speed virtual network with OVS-DPDK or SR-IOV
Networking	NIC partitioning with SR-IOV	V	-	SR-IOV VF is isolated and attached to VM by OpenStack. OpenShift worker node uses a host-device
Networking	Network redundancy	V	(V)[1]	OpenStack provides network redundancy except SR-IOV attached vNIC
Networking	Network QoS setting	V	-	OpenStack provides bandwidth limitation and DSCP marking
Storage	Assign Volume	V	-	Persistent Volume is provided by OpenStack Cinder
Storage	Volume Redundancy	V	-	OpenStack Cinder backend provides Persistent volume
Storage	Volue QoS settin	V	-	IOPS and Bandwidth limitation is provided by OpenStack Cinder
Storage	RWX Volue assign	V	V	Select OpenStack Manilla or OpenShift Data Faundation

Category	Setting	OpenStack	OpenShift	Description
				depends on requirement
Others	DNS record managemen	V	V	Using OpenStack Designate

As such, functions and redundant configurations such as networks and storage are provided by OpenStack as cloud infrastructure, so by effectively using Cloud Provider OpenStack [2], you can reduce the resources managed by OpenShift.

Use and Features of Red Hat Products

Red Hat's telco cloud solutions offer the ability to run VNF and CNF-based network functions side-by-side. Red Hat OpenStack Platform gives service providers the ability to deploy networks at scale and deliver services and applications based on changing customer demand. Telco network functions are placed in the most optimal locations to deliver necessary performance. Red Hat OpenShift helps service providers adopt a cloud-native approach for their telco cloud for accelerated application development and consistency across hybrid cloud environments.

RED HAT OPENSTACK PLATFORM

It's been over a decade since the creation of the OpenStack project. Since then, Red Hat OpenStack Platform has become a leading technology for service provider cloud environments that propels innovation and is one of the top NFV infrastructure for service providers. Red Hat service provider customers running Red Hat OpenStack Platform are estimated to have more than 2.5 billion mobile subscribers in 2022.

- Red Hat OpenStack Platform is the leading commercial distribution
- Red Hat OpenStack Platform provides a scalable, flexible telco cloud environment based on proven, integrated technologies that extend from the core datacenter to edge devices
- Red Hat OpenStack Platform provides a proven foundation for telco cloud's critical workloads

RED HAT OPENSHIFT

Red Hat OpenShift empowers service providers in modernizing their network infrastructure and accelerates their ability to deliver 5G services faster. OpenShift is an enterprise-ready Kubernetes application platform with a choice of deployment options that give service providers complete flexibility to deploy in various locations and environments to build their distributed 5G networks.

- Red Hat OpenShift is the leading commercial Kubernetes solution
- Red Hat OpenShift delivers low and predictable latency, high bandwidth, and distributed architectures needed by telco cloud architectures
- A container-based platform offers a scalable and flexible way to evolve telco cloud infrastructure

OTHER SOFTWARES

- Red Hat Ceph Storage
- Red Hat OpenShift Data Foundation

Practice

In this white paper, we will introduce the details of the practice in Shift on Stack for the following items in the following chapters.

OPENSTACK CLUSTER DESIGN

We will show the design of OpenStack required to practice Shift on Stack. Here is an overview.

- Selection of components to use in OpenStack

OpenStack Components to Use	Purpose
Keystone - Identity Service	Necessary for authentication and authorization
Glance - Image Service	Necessary for image management
Nova - Compute Service	Necessary for running virtual machines
Placement - Placement Service	Necessary for running virtual machines
Neutron(ML2/OVN) - Network Service	Necessary for managing virtual networks
Cinder - Block Storage Service	Used for RWO Persistent Volume
Designate - DNS Service	Used as a DNS server for Ingress
Heat - Orchestration Service	Used for UPI installation/automation
Octavia - Load balancer Service	Used for UPI LB/Service type: LoadBalancer, etc.
Swift/Ceph - Object Storage Service	Used for data backup destinations, etc.

- Compute settings
    - Divide server types that require overcommitment, server types that require CPU Pinning, etc. to increase server utilization efficiency
    - Perform placement design considering the use of NUMA nodes and SR-IOV
- Setting of availability zones
    - Divide the availability zones to increase fault tolerance, and deploy OpenShift with awareness of the availability zones
    - Set availability zones for compute, network, storage, and load balancer respectively
- Network tuning
    - Implement high-speed networking that does not rely solely on SR-IOV by utilizing ML2/OVN and Open vSwitch - DPDK
    - Introduce effective use of SR-IOV
- QoS design
    - Design for storage and network QoS to mitigate the noisy neighbor problem
- Scaling
    - Design for scalability that can manage many OpenShift clusters
- Advance
    - Introduce more effective usage methods, including KDDI's practical examples

OPENSHIFT CLUSTER DESIGN

We will show the design of OpenShift required to practice Shift on Stack. Here is an overview.

- Installation
  - Perform installation by UPI
  - Introduce infrastructure creation by OpenStack Heat and installation automation by Ansible and OpenShift-Installer to simplify UPI installation
  - Introduce the use of Octavia for LBaaS and Designate for DNSaaS
- Virtual machine design pattern
  - Analyze CNF use cases and narrow down the configuration patterns of OpenShift clusters and worker nodes
  - Use Neutron's trunk

---

1. Only for SR-IOV ↩
2. https://github.com/kubernetes/cloud-provider-openstack ↩

1.5 Conclusion

This white paper aims to provide best practices for integrating the Red Hat OpenShift Container Platform with the Red Hat OpenStack Platform in telecom cloud environments, a configuration we refer to as "Shift on Stack".

The advent of open technologies has increased the degree of freedom in choice, leading to a fragmented telecom cloud landscape. From the perspective of Red Hat's products, we introduced various options and released this white paper at an early stage, hoping it would serve as a reference for telecom operators worldwide.

In this white paper, we have:

- Shared the challenges faced by telecom operators, such as the increasing complexity of integration and the shortening product lifecycle in the wave of cloud-native trends.
- Demonstrated that by combining the Red Hat OpenStack Platform for private cloud and Red Hat OpenShift for realizing multi-cloud and hybrid cloud, we can leverage these to solve these challenges.
- Introduced an overview of best practices on how to use these products more specifically, and what tuning considerations exist in the "Shift on Stack" configuration where OpenShift is used on OpenStack.

Moving forward, Red Hat and KDDI plan to expand the content of this white paper around the fourth quarter of 2024, introducing more practical and specific best practices. This will include a comparison that incorporates the use and performance verification of OpenShift on bare metal, another option in the telecom cloud. Furthermore, we plan to introduce actual use cases in a major Japanese telecom operator, KDDI's 5G core system, not just benchmarks and desktop studies.

Red Hat and KDDI will continue to contribute to the acceleration of innovation in the telecommunications industry by utilizing open-source software and keeping information open. If you have interest in our activities, please feel free to contact us at telco-cloud@kddi.com.

1.6 Contributors

1.6.1 Red Hat

- Nick Satsia
    - Global Solution Architect in the Teleco industry
- Kazutoshi Hayakawa
    - Solution Sales Specialist in Teleco industry

1.6.2 KDDI CORPORATION

- Hiroshi Tsuji
    - Seinior Expert of Network and Cloud Platform
- Takuya Sawada
    - Deputy Head of Node Technology Department

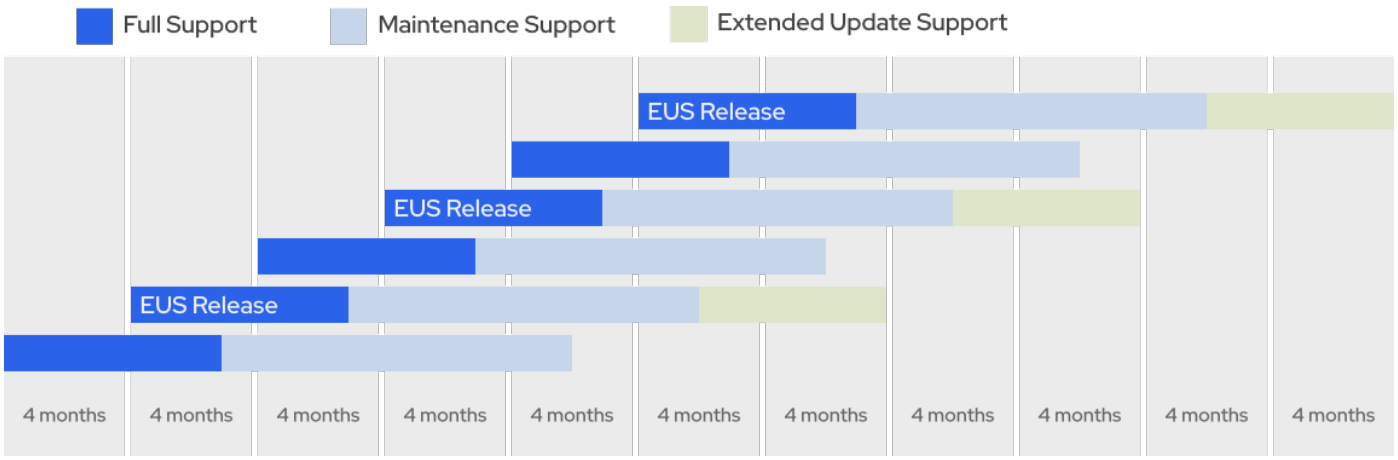
1.7 Changelog

Version	Date	Information about changes
v1.0	2024-05-07	Initial version

1.8 Appendix

1.8.1 Appendix A: Details of Challenges

Short lifecycle / Frequent upgrading



Official OpenShift Life Cycle[†]

In the Kubernetes upstream, a release cycle is adopted where a new major version is released three times a year. This is followed by 12 months of regular support and 2 months of maintenance support, providing a total of 14 months of code fixes. At Red Hat, this support period is enhanced in the Red Hat OpenShift Container Platform, which is based on Kubernetes. It offers 6 months of full support and 12 months of maintenance support, and provides 6 months of support for specific versions as Extended Update Support (EUS). This achieves a maximum of 24 months of support as standard.

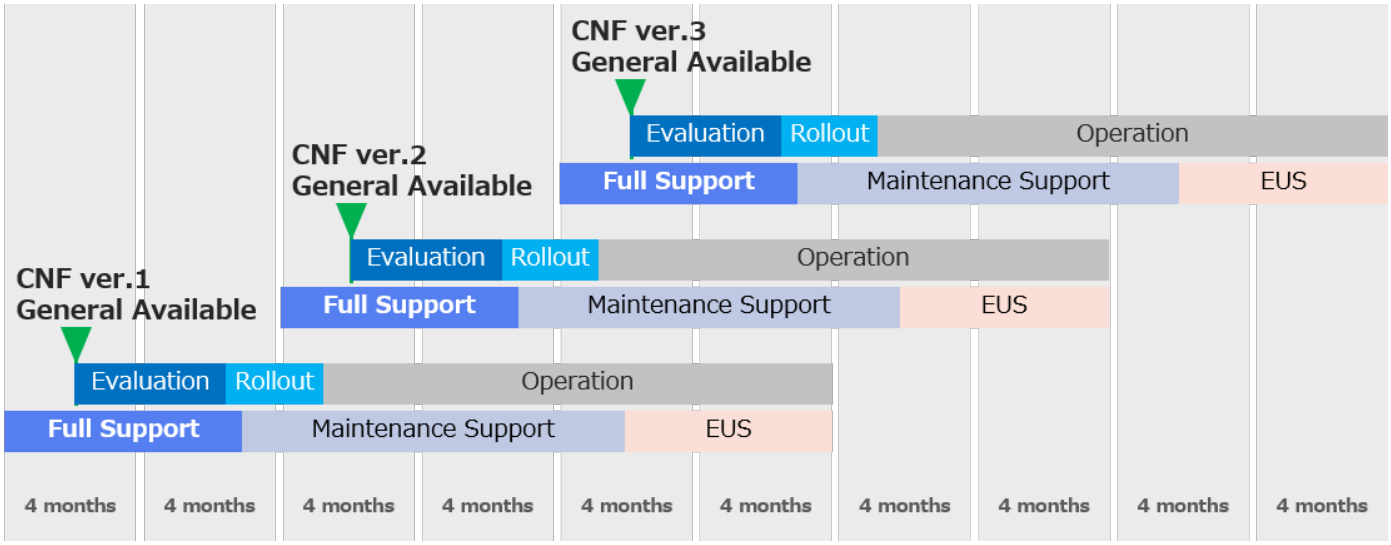
Even with the long-term support provided by Red Hat OpenShift, this is still very short compared to the 10-year long-term support at the enterprise level provided by Red Hat Enterprise Linux, or the 4-year support provided by Red Hat OpenStack Platform.

This short software lifecycle makes it easier to incorporate new features, improvements, and deprecations into Kubernetes and Red Hat OpenShift, which are platform software that run applications, accelerating the metabolism of the Kubernetes software.

This rapid update of platform software is an effective method for web companies, where the lifecycle of the applications providing the service is overwhelmingly faster than the platform software, due to frequent CI/CD execution in environments where business and technology are changing rapidly.

However, this short software lifecycle of Kubernetes also presents challenges in the telecommunications domain, where stable infrastructure is required.

Mobile systems have generations like 4G/5G, and major changes are made about every 10 years. In the mobile core network, which accomodates many subscribers unitgerruptedly, care is taken to avoid service outage caused by upgrade. Therefore, mobile operators have traditionally operated by updating hardware and software for problem resolution and maintenance limits set by the hardware and software vendors they use. Generally, updates were made only a few times a year at most for a single software or system.



Example life cycle between OpenShift and CNF

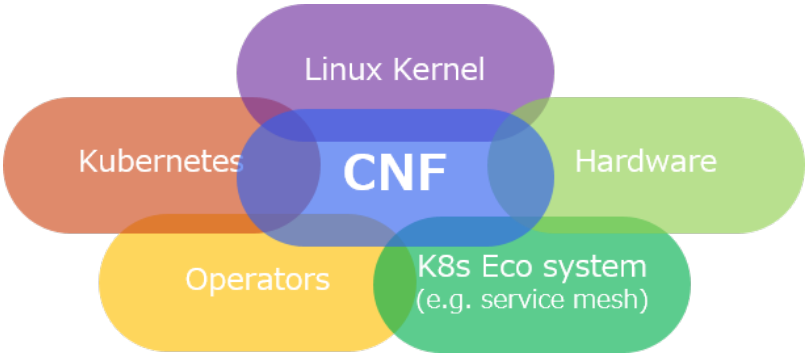
For telecom application vendors and telecom operators, adapting to Kubernetes' short lifecycle and upgrading the software version of both Kubernetes and telecom applications every two years is a very challenging task.

Tight coupling between Kubernetes and CNF versions

As explained in the previous section, upgrading the software version to match Kubernetes' short lifecycle is a significant challenge.

Furthermore, to provide stable communication at a low cost, network operators demand very high performance and availability requirements from telecom applications, which are network functions. To meet these requirements demanded by network operators, telecom application vendors providing CNFs need to meticulously tune Kubernetes and the operating system.

Therefore, telecom application vendors thoroughly verify their CNF software in their own labs. They then require operators to reproduce the configuration content found and verified in the lab as much as possible in the operator environment. Network application vendors often clearly specify detailed parameter settings, including the versions of Kubernetes and OpenShift, to operators, defining clear operating conditions for the CNFs. This kind of tuning is more prominent in data plane CNFs (such as UPF) that handle large volumes of mobile subscriber traffic using DPDK or SR-IOV.



CNF has many strong dependencies

Given this situation, there is a strong dependency between telecom applications and Kubernetes or OpenShift. For operators, adopting these versions and settings in line with the application, Kubernetes, and OpenShift versions and parameter tuning verified by the vendor is the least risky and most reliable method when introducing network functions.

However, this approach presents challenges when constructing a core network with multiple vendors. This is because different vendors and CNF products may use different versions of Kubernetes and OpenShift, and the parameters and tuning know-how of the products used may

differ. Therefore, to standardize the container platform layer such as Kubernetes/OpenShift, ensure competitiveness, and construct a multi-vendor core network to ensure redundancy, it is necessary to absorb differences in versions and settings for each CNF and align interfaces.

In reality, Tier 1 operators leading the telecom industry are strongly promoting these initiatives and achieving results. However, realizing these requires a great deal of energy, and it is also a realistic challenge that not all operators can adopt the same strategy as them.

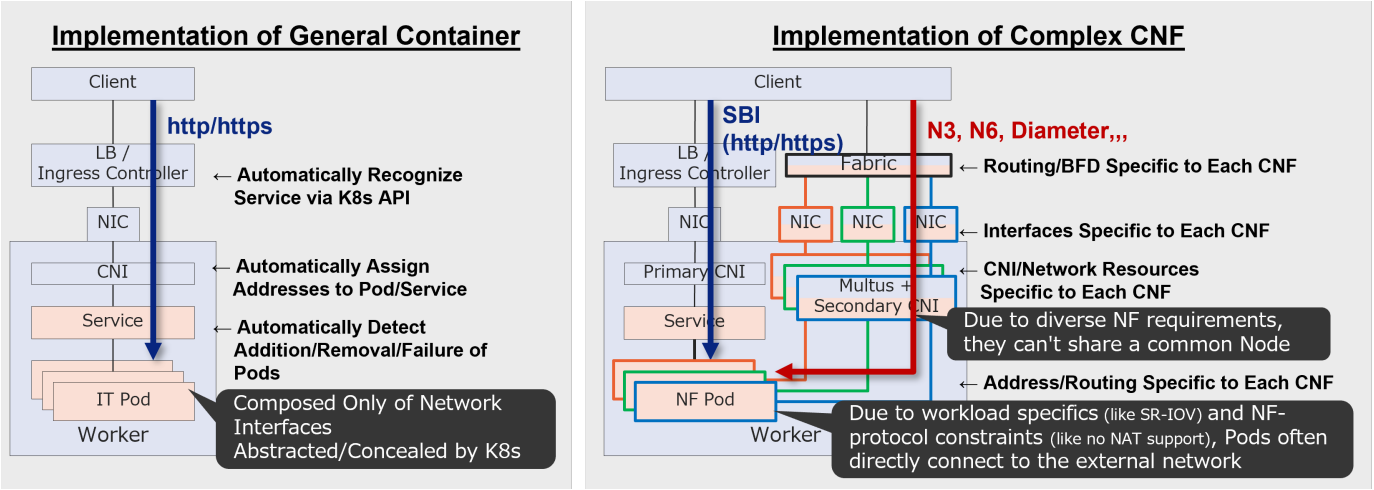
Complex network requirements

The network requirements for CNFs used in the core network significantly differ from those of typical web applications, which cnstitue the most common use cases of Kubernetes.

Traditional network requirements

In particular, for CNFs that terminate user sessions or handle user plane communication, and other CNFs that require quick application redundancy switching, L2 redundancy using VRRP or L3 redundancy combining BGP and BFD is implemented to achieve immediate switching at the application level. In addition, CNFs may need to have multiple IP addresses to use SCTP multihoming, or need to have a VLAN interface due to application design. Furthermore, in the traditional field of communication, signaling and data plane, OAM traffic, and service orders from BSS, etc., interfaces, IP addresses, VRF, and network devices have been separated and created according to the type of communication.

These requirements are significantly different from the cloud-native concept where it is sufficient to connect to one VPC and external traffic can communicate using NAT/NAPT.



Compared to general containers, CNFs require more complex implementation[††]

Furthermore, in data plane CNFs, SR-IOV and DPDK may be used. SR-IOV is a technology that handles network interfaces logically divided as PCI-E devices. DPDK is a technology that handles network interfaces with userland drivers instead of kernel drivers. Therefore, when using SR-IOV and DPDK, it is necessary to handle PCI-E devices directly inside the CNF container.

Due to these characteristics of telecom applications and operational requirements, when using Kubernetes in telecom, there is a need to use multiple networks such as VLAN and SR-IOV in addition to the primary CNI for a single Pod. Therefore, many CNFs use Multus as a CNI and connect multiple network interfaces.

However, as of 2024, Kubernetes only manages addresses as IPAM and performs service discovery for the primary CNI.

Also, the primary CNI basically cannot carve out virtual networks and provides a flat network to all containers, filtering access control between containers using the RBAC mechanism.

In the case of using Kubernetes alone, the networks of the secondary CNI connected by the CNF are provisioned outside of Kubernetes, and Kubernetes operates on the assumption that the provisioning has been correctly performed for the Deployment or StatefulSet manifest for the applied CNF.

Therefore, because the complex network requirements of telecom applications and telecom operators cannot be resolved with only the standard features of Kubernetes, these network requirements need to be met through automated or manual operations.

CNF and VNF mixed workload

The new generation of network functions are often deployed using cloud-native technologies as CNFs. However, there are still many network functions that are not converted to CNFs.

For example, older standard network functions such as 4G, network functions with little change like IMS, core network functions of fixed networks, and functions for operation monitoring such as OSS/BSS, often renew their network functions composed of existing physical servers and virtual servers on stable virtual servers. This is because containerization is not just about using container technology, but also about transforming the system architecture to become cloud-native, such as microservices.

However, these traditional applications originally adopt a monolithic architecture, which necessiates changes to the application itself to enable containerization.

However, converting a monolithic application to microservices is not an easy task. The purpose of containerization, cloud-native transformation, and microservices is to make changes and modifications safer and smaller.

Therefore, for these network functions and systems that have existed from the old days and have little change, it may not be necessary to fully adopt cloud-native technologies.

Rather, it can be important for business to continue to provide services with minimal changes, less effort, and lower costs.

It's important to be aware that cloud-native is not the only best technology choice, but one of the best options we can take.

Therefore, in telecom operators, VNFs are expected to continue being widely utilized in the future. It is essential that the telecom operators' platform can accomodate both CNFs and VNFs concurrently.

---

1. https://access.redhat.com/support/policy/updates/openshift ↩
2. https://www.redhat.com/it/blog/building-cnf-applications-with-openshift-pipelines ↩

2. 日本語版

2.1 はじめに

> **このホワイトペーパは早期公開版としてリリースしています**
>
> このホワイトペーパは早期公開版としてリリースしてるため、ホワイトペーパ内で参照、予告している内容が現時点のホワイトペーパのコンテンツに含まれない場合があります。
>
> ホワイトペーパは2024年第4半期にすべてのコンテンツが含まれる形でリリースできるように、Red HatとKDDIにおいて準備しています。

2.1.1 概要

このホワイトペーパではテレコクラウドにおける Red Hat OpenShift Container Platform をRed Hat OpenStack Platform とインテグレーションして利用する OpenShift on OpenStack, "Shift on Stack"構成のベストプラクティスを提供します。

通信事業者を取り巻く技術的環境は大きく変化しています。 特にNetwork Functions Virtualization (NFV)、における仮想化の導入され、3GPP Release 16 以降で規定された、5G Core network(5GC) では Service-based Architecture (SBA) が規定され、通信事業者におけるクラウドネイティブに向けた動きが加速しています。

このホワイトペーパではこのクラウドネイティブの流れのなかで生じているインテグレーションの複雑化、製品ライフサイクル短期間化といった通信事業者の課題を共有します。次にRed Hat OpenStack Platform によるプライベートクラウドと、マルチクラウド、ハイブリッドクラウドを実現する Red Hat OpenShift を組合せることで、レバレッジを実現しこれらの課題を解決できることを示します。さらに具体的にこれらの製品をどのように利用するのか、OpenShiftをOpenStack上で利用する"Shift on Stack"構成において どのようなチューニングや考慮点があるのかのベストプラクティスをもとに紹介します。 またテレコクラウドにおけるもう一つのオプションであるベアメタル上でのOpenShiftの利用と比較します。 最後に、実際のネットワークでの活用事例として、日本の大手通信事業者であるKDDIでの活用事例を紹介します。

なお、このホワイトペーパはテレコクラウドを導入する通信事業者と、通信事業者のテレコクラウドに対してアプリケーションを提供するテレコムアプリケーションベンダを対象としています。

2.1.2 序文

通信におけるオープン化やコモディティ化の流れは途絶えることなく続いています。 専用マシン、専用チップ、専用OS、専用アプリケーションで動作していたデジタル交換機は 2000年代には専用マシンは汎用CPUを搭載したAdvanced TCAハードウェアやCOTSサーバに変わり、専用OSは台頭してきたLinuxの技術を利用したキャリアグレードLinuxに変化しました。2010年代には Red Hat Enterprise Linux のような エンタープライズ Linux が通信事業者のコアネットワークでも利用することが当たり前になり、さらに、ベアメタルハードウェアをサーバ仮想化、ネットワーク仮想化技術を利用して仮想化し効率よく利用する流れが生じました。また、この仮想化とパブリッククラウドによるクラウドコンピューティングの台頭により、プライベートクラウドを作るソフトウェアとしてOpenStackが誕生し、通信事業者での導入されました。そして2020年代前半の現代においては、新たにコンテナやコンテナオーケストレーションソフトウェアであるKubernetesを活用して、通信事業者のコアネットワーク内部においてもクラウドネイティブ技術を活用し、自動化されたオペレーションと迅速なアプリケーションを展開を容易にしています。

しかしながら、急速なテクノロジーの変化に対して通信のソフトウェアやオペレーションが必ずしも対応しきれているわけではありません。特に、通信事業者の置かれたビジネス環境は特殊です。現代社会においてモバイルネットワークサービスをはじめとする通信サービスが存在しないことはもはや考えられません。社会インフラとなった通信がひとたび長時間にわたりアウテージすれば通信事業者の直接的なお客様だけではなく間接的に通信を利用したサービスを受けている多くの市民に対しても多大なる影響を与え、社会に混乱を招いてしまいます。「キャリアグレード」「テレコムグレード」という言葉は過去から存在しますが、その言葉の持つ意味の重さは年々増しています。

このようなビジネス上の環境からも通信のソフトウェアやオペレーションには、一般的なエンタープライズ要件に比べて特殊な要件が少なからず存在します。特に数千万加入者の通信を滞りなく利用できるためのモバイルコアネットワークは複雑です。その複雑なモバイルコアネットワークを構築するためには、オペレータだけではなく、テレコムソフトウェアを提供するテレコムアプリケーションベンダの協力は欠かせません。さらに、モバイルコアネットワークでは異なるベンダ間や異なるオペレータ間での相互接続性や、過去のシステムとの後方互換性・接続性(例えば4Gと5Gとの相互接続など)が非常に重要です。そのためには3GPP等による標準化策定と、それを元にした実装が欠かせません。さらに、これらの互換性のためには古いプロトコルや古いインタフェースを利用し続ける必要もあります。

特に新興のモバイルネットワークオペレータを除いた場合、古いシステムと新しいシステムを共存させていくことはオペレータにとって非常に重要なテーマです。5Gサービスが始められて久しい2024年の現在においても、多くの通信事業者においては依然3Gサービスが稼働されています。また地域によってはより古い2Gサービスも依然利用されています。さらに新規加入者の獲得が難しいこれらの古いサービスについては低コストでの運用が求められるます。このような古いシステムとの連携や、古いシステムのマイグレーションも発生するため、特に長い歴史を持つ通信事業者においてはインフラに対する要件などが複雑化します。また、テレコムアプリケーションは膨大なトラフィックを処理し、ネットワークの輻輳時にも正しく処理できることが求められます。テレコムアプリケーションベンダやオペレータはこのために高性能要件を求め、それを満たすことを確認するためのに安定的にテストも必要となります。

さらに多くのシステムが複雑に相互作用して動作するモバイルコアではミスなく、安定してオペレーションする難易度も上がっています。ソフトウェアやハードウェアのマイグレーション作業は安定した通信を提供するためには欠かせませんが、このマイグレーションの実施そのものも安定して実行していく必要があります。その為にも自動化は重要なファクタですが、ステートを様々な箇所でを多く持つモバイルコアネットワークにおける自動化の考慮事項は多くなります。さらに問題がおきた場合のロールバック手段の確保も非常に重要です。

このように通信事業者においてRed Hat OpenShift の持つハイブリッドクラウド・クラウドネイティブ技術を有効に活用していくためには、テレコム業界の持つ現実的な課題に対しては、明日の新たな技術だけではなく、現在や過去にまで目を向けた、地に足が付いた議論が必要です。

このホワイトペーパでは、Red Hat OpenStack Platform によるクラウドコンピューティングの概念と、Red Hat OpenShift によるハイブリッドクラウドの概念を組み合わせて、これらの現実的なオペレータの課題に対して一つのソリューションを提供します。

2.2 背景

2.2.1 仮想化とNFV

ネットワーク機能仮想化（NFV）は、従来は専用ハードウェアで実行されていたネットワークサービスを仮想化する方法です。これらのサービスは、コモディティハードウェア上の仮想マシン（VM）としてパッケージ化され、サービスプロバイダーが専用のサーバーではなく標準のサーバーでネットワークを実行できるようにします。NFVは、追加のハードウェアリソースを必要とせずに、サービスプロバイダーが新しいネットワークサービスやアプリケーションをオンデマンドで提供できるようにして、拡張性と俊敏性を向上させます。

NFVアーキテクチャ内の主要なコンポーネントは、仮想ネットワーク機能（VNF）であり、需要が変化すると必要に応じて異なるサーバーで柔軟に実行したり、移動したりすることができます。この柔軟性により、サービスプロバイダーはサービスやアプリケーションをより迅速に提供できます。例えば、顧客が新しいネットワーク機能を要求した場合、そのリクエストを処理するための新しいVMを立ち上げることができます。機能が不要になった場合、VMを廃止することができます。これは、潜在的な新しいサービスの価値をテストするための低リスクな方法でもあります。

VNFは、仮想インフラストラクチャマネージャ（VIM）の上に構築されます。これにより、VNF間で効率的に計算、ストレージ、およびネットワークリソースを割り当てるためのインフラストラクチャが抽象化されます。NFVIを管理し、新しいVNFをプロビジョニングするフレームワークは、NFVによって定義された管理、自動化、およびネットワークオーケストレーション（MANO）要素によって行われます。

詳細情報

- Understanding virtualization
- What is NFV?

2.2.2 テレコクラウド

テレコクラウドは、ソフトウェアで定義された、非常に強靭なクラウドインフラストラクチャであり、通信サービスプロバイダー（テレコ）がサービスを迅速に追加し、ネットワーク需要の変化により迅速に対応し、中央および分散されたリソースをより効率的に管理できるようにします。これは、成功したデジタル変革の中核的な要素の一つです。

エンタープライズクラウドは、内部の管理機能を実行し、顧客向けのポータルを持つことができます。これらはすべて、パブリック、プライベート、およびハイブリッドの構成の組み合わせで提供されます。伝統的に、テレコクラウドは、より制限されたネットワーク機能や必須のビジネスアプリケーションを実行することに焦点を当てています。これらは、観測性、制御、障害耐性、および可用性のレベルがはるかに高いものです。

グリーンフィールド・オペレーター（以前存在しなかったインフラストラクチャを構築する者）は、ゼロから完全にクラウドネイティブな環境を構築することができますが、既存のオペレーターは、レガシー・ネットワーク環境と連携するテレコクラウドを構築する必要があります。レガシーとクラウドネイティブのネットワークは一定期間共存する必要があり、オペレーターが組織にとって最も意味のある方法でネットワーク機能、サービス、およびアプリケーションを移行できるようにします。

クラウドネイティブなアーキテクチャへの移行では、総合的なアプローチが重要です。移行は1つのネットワーク機能やサービスごとに実施されるかもしれませんが、プロセスはインフラストラクチャ、アプリケーションとサービスのポートフォリオ、組織、およびプロセスを包括する包括的なクラウド対応評価から始めるべきです。

詳細情報

- What is telco cloud?

2.2.3 コンテナ化とマイクロサービス

5Gの登場により、テレコクラウドはコンテナやマイクロサービスなどの新しい技術、そしてハイブリッドクラウドアーキテクチャを活用するようになります。CNFの採用は、多くの機能をコンテナに移動することで、VNFのいくつかの制限を解消することができます。ネットワーク機能のコンテナ化により、環境内で機能がどのようにしてどこで実行されるかを管理することが可能になります。

CNFは単なるネットワーク機能のコンテナ化にとどまりません。クラウドネイティブの原則の完全な利点を得るためには、ネットワーク機能ソフトウェアの再アーキテクチャがさらに必要であり、それはマイクロサービスに分解し、更新時に複数のバージョンを許可し、ジェネリックなロードバランサーやデータストアなどの利用可能なプラットフォームサービスを使用することを含みます。

より多くのサービスプロバイダーがクラウドネイティブ環境を採用するにつれて、CNFは移行中にレガシーのVNFと共存しなければなりません。サービスプロバイダーは、エスカレートする需要を効果的に処理し、展開を加速し、複雑さを削減するために、ネットワークの開発、展開、保守、および運用を完全に自動化する必要があります。設定や展開のための実証済みの手法、オープンソースコミュニティで成熟したツール、厳格なテストと認証は、サービスプロバイダーにとってこれまで以上に重要です。

テレコクラウドは、柔軟性とベンダー独立性を高めるオープンで水平なハイブリッドアプローチのクラウドインフラストラクチャに頼るように進化しています。業界は、垂直統合されたスタックではなく、オープンで水平なアプローチがより速いイノベーション、改善されたパフォーマンス、高い俊敏性、および総所有コストの大幅な削減につながることを認識しています。

詳細情報

- Understanding cloud-native applications

2.2.4 フラグメント化の問題

かつてNEPは、彼ら自信のハードウェア上で動くソフトウェアを一体として通信事業者に提供してきました。一方でクラウドネイティブな世界においては、NEPは通信事業者もしくはサードパーティから提供されるハードウェアの上で動作するソフトウェアだけを提供することになります。このソフトとハードの分離はNEPと通信事業者の両者に多くの利点をもたらすと同時にいくつかの問題を生じさせています。NEPは複数のプラットフォーム上でそのソフトウェアが適切に動作し、保証された処理能力を担保しなくてはならなりません。このためNEPは必要な検証や場合によっては修正を行う必要があり、ときに追加コストにもつながることになります。

クラウドネイティブへの取り組みは通信事業者ごとに様々であり、現時点ではフラグメント化していると言えるでしょう。このフラグメント化はNEPと通信事業者に余分なコストを強いています。

このホワイトペーパーでは、通信事業者がRed Hat OpenShift Container Platform を Red Hat OpenStack Plaftform 上で動作させる際のベストプラクティスのひとつを提供します。われわれはこの活動がフラグメントの課題を抑制し、プライベートクラウドソリューションの利点を得ることにつながると期待しています。

2.3 "Shift on Stack"ソリューションの概要

2.3.1 ソリューションの概要

クラウドネイティブに向けた通信事業者の課題

今後もテレコムアプリケーションやテレコクラウドの中においてこれらのクラウドネイティブ技術を活用していくことは重要であり、 そして活用していくことで得られる価値が存在していることににについて疑念の余地はないでしょう。

しかしながら2024年現在においてはその変革はまだ道半ばであるかもしれません。

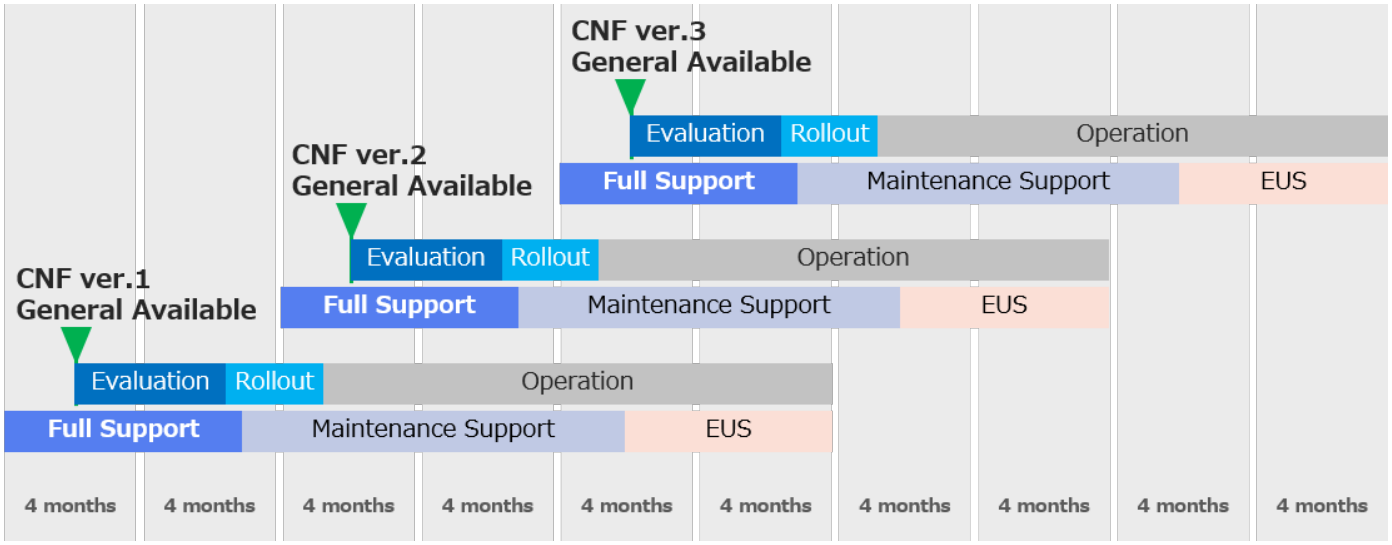
特にテレコムアプリケーションは短期間で物理アプライアンスからNFVによるの仮想化に移行しました。そして次はKubernetesを中心としたコンテナ技術をベースにしたクラウドネイティブ技術を採用しました。 さらにそれらと時を同じくして4Gから5Gへコアアーキテクチャが変化しました。 このように2010年代から現在にかけて技術的なパラダイム変化が激しく起こりました。

これらの目まぐるしいパラダイムの変化の中でテレコムアプリケーションベンダもテレコムオペレータのどちらも素早い変化を求められた結果試行錯誤を繰返しています。 通信業界では非常に高い可用性が求められています。また利用する通信プロトコルも常にあたらしい物に、そして自由に置き換えられるわけではありません。 旧世代の通信規格との互換性のために、旧規格の通信プロトコルを残し続ける必要があります。 また、これらの通信の方式や信頼性の高い冗長化設計などについても過去からの積み上げた資産が多数あります。 このため、テレコムアプリケーションベンダが5Gなどの新しい世代のソフトウェアにおいても過去の資産を流用することは、経済合理性の観点からも必要なことです。

その結果、DiameterやGTPなどといったSCTPやUDPを用いた通信プロトコルの利用は以前続いています。KubernetesにおいてもMultusを用いたマルチCNIやmacvlan/ipvlanといったvlanやIPアドレスを意識したコンテナ実装がされています。これによりアプリケーションレイヤでの即時のスタンバイへの切り替わりが求められる通信ソフトウェアにおいて、VRRP等を利用したL2冗長化方式など旧来からの実装方式や考え方を利用し続けています。

このような中でクラウドネイティブ技術を活用していくためには以下のような具体的な課題をオペレータは克服していく必要があります。

短いライフタイムとアップグレード



Example life cycle between OpenShift and CNF

Kubernetesは年に3回新しいメジャーバージョンをリリースし、14カ月の修正が提供されます。Red Hatはこれを基にOpenShift Container Platformを提供し、最大24カ月のサポートを実現しています。しかし、これはRed Hat Enterprise Linux(RHEL)のようなエンタープライズレベルの長期サポートやRed Hat OpenStack Platformのもつ4年サポートに比べて短いです。

この短いライフサイクルは新機能の追加や古い機能の廃止を容易にし、ソフトウェアの新陳代謝を早めます。これは変化が激しい環境やウェブ企業にとって有効ですが、安定したインフラストラクチャが求められる通信領域には課題があります。

モバイルシステムは約10年ごとに大きな変更があり、サービス停止を避けるために更新は慎重に行われます。そのため、Kubernetesの短いライフサイクルに適応し、2年ごとにソフトウェアバージョンを更新することはテレコムアプリケーションベンダーやテレコムオペレータにとって大きな課題です。

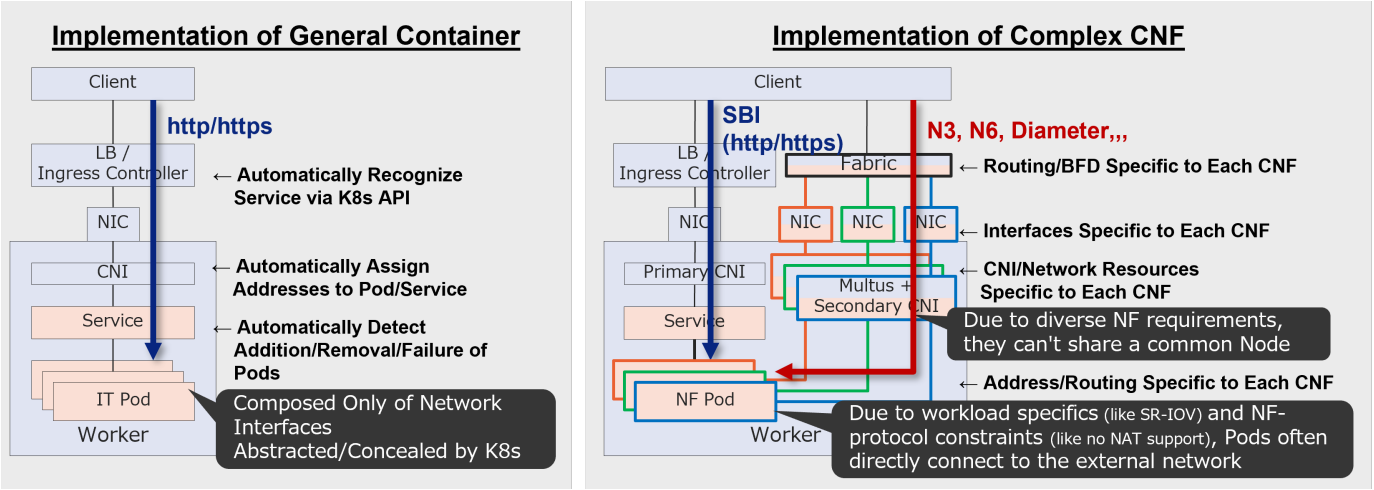
KUBERNETES バージョンとCNFバージョンの強い紐付け

Kubernetesの短いライフサイクルに対応するためのソフトウェアバージョンのアップグレードは大きな課題です。さらに、ネットワークオペレータは、テレコムアプリケーションに高い性能要件や可用性の要件を求めます。これを満たすために、テレコムアプリケーションベンダはKubernetesやオペレーティングシステムのチューニングを行う必要があります。

テレコムアプリケーションベンダは自社のラボでCNFのソフトウェアの検証を行い、その設定をオペレータ環境でも再現することを求めます。しかし、これはマルチベンダでコアネットワークを構成する上で課題があります。それは、ベンダごと、製品ごとに利用するKubernetesやOpenShiftのバージョンやパラメータが異なるためです。

そのため、コンテナプラットフォームのレイヤを共通化し、コアネットワークをマルチベンダで構成するためには、これらのCNFごとのバージョンや設定の差分を吸収し、インタフェースを揃える必要があります。しかし、これを実現するためには多大なエネルギーが必要であり、全てのオペレータが同様の戦略を取ることは現実的な課題です。

複雑なネットワーク要件



Compared to general containers, CNFs require more complex implementation

コアネットワークで使用されるCNFのネットワーク要件は、一般的なウェブアプリケーションなどとは大きく異なります。特に、高速なアプリケーションの冗長化切り替えが求められるCNFでは、VRRPなどのL2での冗長化や、BGPやBFDを組み合わせたL3での冗長化を行います。また、SCTPのマルチホーミングを利用するために複数のIPアドレスを持つことが必要な場合もあります。

データプレーン系のCNFでは、SR-IOVとDPDKを使用する場合もあります。これらを使用すると、CNFのコンテナ内部で直接PCI-Eデバイスを扱う必要があります。

これらのテレコムアプリケーションの特性や運用上の要件から、テレコムでKubernetesを使用する場合、一つのPodに対してプライマリCNIの他に、VLANやSR-IOVなどの複数のネットワークを利用できる必要があります。しかし、現時点ではKubernetesがIPAMとしてアドレス管理を行ったりサービスディスカバリを行うのはプライマリCNIだけです。

したがって、テレコムアプリケーションやテレコムオペレータの複雑なネットワーク要件を満たすためには、その他の自動化または手動のオペレーションが必要となります。

CNFとVNFの混在環境

新世代のネットワークファンクションはクラウドネイティブ技術を用いたCNFでデプロイされることが増えていますが、旧規格のネットワークファンクションや変化が少ないネットワークファンクションなどは依然としてCNF化されないことが多くあります。これは、Containerizationがシステムアーキテクチャのクラウドネイティブ化を必要とするためで、モノリシックなアプリケーションをマイクロサービスにすることは容易ではありま

せん。そのため、変化が少ない旧来のネットワークファンクションでは、クラウドネイティブ技術をフルスケールで採用する必要はなく、少ない労力とコストで持続的に提供することが重要です。したがって、テレコムオペレータのプラットフォームでは、CNFとVNFの共存が求められます。

より詳細な課題

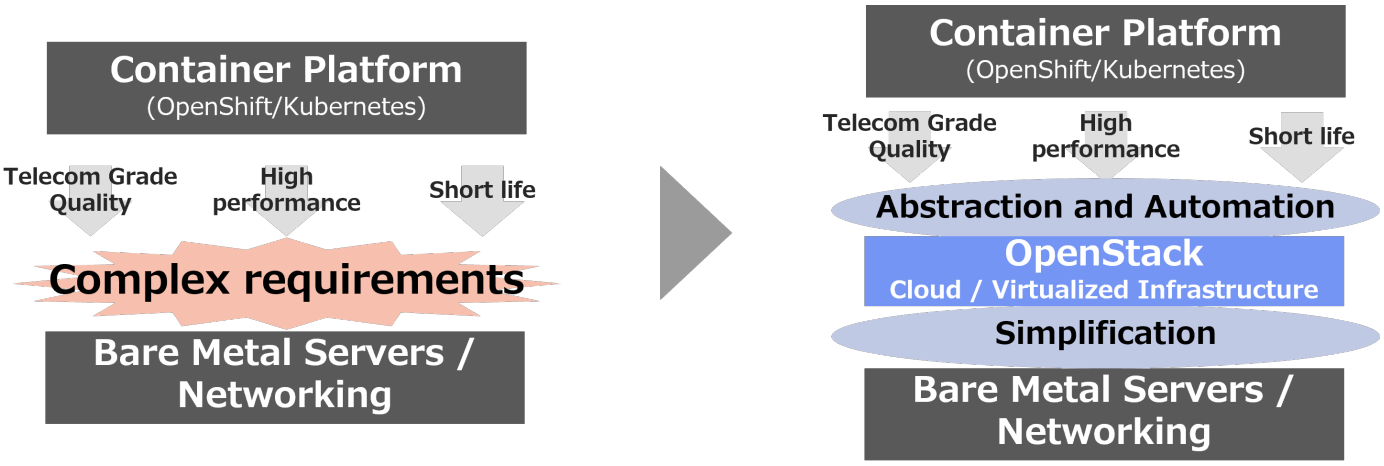
課題に関してのより詳細な情報は付録Aをご参照ください

"Shift on Stack"による課題の解決

次のチャプターからはこれらの課題の詳細を論じます。そして、プライベートクラウドとしてのOpenStackとクラウドネイティブプラットフォームとしてのKubernetesを組み合わせる、Openshift on OpenStack(Shift on Stack[1])によるソリューションの有効性を示します。

基本的なコンセプトと戦略

OPENSTACKによるOPENSHIFT/KUBERNETESの価値の最大化



Benefits of inserting an abstraction layer through OpenStack

一般的にOpenStackをOpenShiftとベアメタルサーバの間に存在することは、様々なオーバヘッドだと考えられることがあります。 これらのオーバヘッドと呼ばれるものの中にはOpenStackのサービスやハイパーバイザのQEMU/KVMの処理、 Open vSwitchといったハイパーバイザとして動作させるために必要なソフトウェアや OpenStackのデータベースやAPIを提供するコントローラノードなどが含まれます。 しかしながら、これらのオーバーヘッドはただ無駄を生むだけではなく、ハードウェアを抽象化することにより高度な自動化や管理の簡素化を生み出します。

このOpenStackによる抽象化などといったメリットを最大限に生かし、レバレッジを効かせることができれば、 テレコムオペレータが持つ課題を解決できるでしょう。

このホワイトペーパではRed Hat OpenStack Platformによる仮想化・プライベートクラウドと、 OpenShiftによるクラウドネイティブプラットフォームを組み合わせることにより、 その価値を最大化し得ることを説明します。

Shift on Stackを用いることで次のような効果を得られます。

効率的なインテグレーション

NFを提供するNEPベンダはNF個々に最適なKubernetes/OpenShiftバージョンや周辺エコシステムを選択できます。また、ハードウェア構成やネットワーク構成についても、仮想化を挟むことで自由に選択することができるようになります。これにより他ベンダのNFと共存するためにオペレータから指定される制約事項や、オペレータ環境固有のハードウェア構成、ネットワーク構成の影響を受けることなく、自社の検証環境に近い構成をオペレータの実際のネットワークで再現することができます。

シンプルなオペレーション

Kubernetes/OpenShiftの最大のメリットはアプリケーションの運用に集中することができることです。そのため、アプリケーション観点からは本質的な価値がない、Kubernetes/OpenShiftのクラスタの構築やアップグレードを簡素化することは非常に重要になります。サーバやネットワークリソースをオンデマンドに作成したり、削除したりすることはクラウドインフラストラクチャの最も重要かつ効果を発揮するポイントです。Kubernetes/OpenShiftを気軽に作り変え・組み換えができるインフラストラクチャが存在することはコンテナアプリケーション同様にKubernetes/OpenShiftクラスタもイミュータブルに運用することができるため、これらの運用をを非常にシンプル化することができます。

自動化の推進

OpenStackはインフラストラクチャに対するすべての操作をWeb APIに実現できます。さらにこれらのAPIを利用したOpenStack CLI、OpenStack HeatやAnsible、Terraform/OpenTofuによるIaCツールによりKubernetes/OpenShiftクラスタの仮想サーバやプライマリネットワークだけではなくセカンダリネットワークやKubernetes/OpenShiftの外部に必要となる場合があるDNSレコード、ロードバランサ、コンテナレジストリ、Bastionサーバ、監視サーバなどといったインフラストラクチャの管理を自動化することができます。OpenStackは、これらの必要とされるネットワークなどの機能の提供と、テンプレートや仮想化機能を使った自動化を提供することが可能です。

解決される課題

課題1 : 短いライフタイムとアップグレード

- OpenStackによる仮想サーバや仮想ストレージ、仮想ネットワークに対するインフラストラクチャのAPIは、 IaCツールによるOpenShiftのセットアップの自動化を補強します
- OpenStackによるインフラストラクチャー抽象化は、OpenShiftクラスタの作成や削除などのオペレーションを効率化します。これにより、テナントごとに必要な異なるバージョンのOpenShiftクラスタを簡単にデプロイおよびサポートできるようになります。ライフサイクル管理の容易さにより、新しいクラスタバージョンを簡単に再展開し、変更が完了したら古いバージョンのクラスターを停止または削除することができます。
- OpenShiftクラスタから永続的なデータを取り除き、永続性やステートのあるリソース(Persistent Volume) のリソースのライフサイクル管理をOpenStackに担わせることが出来でき、OpenShiftクラスタの入替を容易にします
- OpenStackはOpenShiftに対してサーバやネットワークなどを抽象的かつ一貫性のあるインフラストラクチャリソースとして提供できます
- さらに、仮想化によるハードウェアの抽象化は、 仮想マシンインスタンスに対して物理的なハードウェアの存在を隠蔽できます。

これによりSR-IOVなどのPCI-Eパススルー機能を用いるような機能を除き、 実際の物理的なハードウェアのライフサイクルと仮想マシンのライフサイクルを分離することができます。

課題2: Kubernetes バージョンとCNFバージョンの強い紐付け

- インフラストラクチャを分離し抽象的にOpenStackにより提供できるため、 CNFを提供するネットワークアプリケーションベンダは自社でテストしたOpenShiftバージョンや、 OpenShiftの設定値をほかのベンダとの競合を気にすることなく、オペレータの環境で再現できます。

課題3: 複雑なネットワーク要件

- テナント毎、作成されるネットワーク毎に独立した仮想ネットワークをオンデマンドに作成することが、 OpenStackではNeutronが提供する仮想ネットワーク機能により実現できます。
- Neutronの持つ仮想ネットワークの作成と分割機能は、CNF毎やOpenShiftクラスタ毎に必要となる ネットワークの分離を提供できます。
- APIを介しすべてのNeutronによるネットワークオペレーションは完全に自動化できます。

課題4: CNFとVNFの混在環境

- OpenStackによる仮想マシンの管理機能は強力です。
- VNFを管理する機能としてのOpenStackは完成されており、十分な実績を有しているため、 CNFのためのOpenShiftクラスタを仮想マシンとして扱うことでこのOpenStackの機能を活用できます。

---

1. 本ホワイトペーパではOpenShift on OpenStackについて"Shift on Stack"と略して表記する場合があります。 ↩

2.3.2 効率的なインテグレーション

ハードウェアの抽象化

一般的にハードウェアのライフサイクルはKubernetesのライフサイクルに比べて長いですが、必ずしもアラインされるわけではありません。また増大するトラフィックに対応しネットワークを増強するために、デマンドに合わせたサーバリソースの拡張を行う必要があります。しかしながら、調達時期により異なる世代のハードウェアを調達を行うことも考えられます。またRFPを実施しより条件の良いハードウェアをその都度選択し、ハードウェアを変更したいかもしれません。

このような状況において、ベアメタルサーバを直接使う場合、例えば使用する高速なNICのPCIeカードがNUMA Nodeのどちら側に存在するのかといった観点など、Kubernetes/OpenShiftならびに CNFはこれらのサーバに合わせたそれぞれの設定のチューニングを行ったり設計の変更を行う必要があるでしょう。

OpenStackによる仮想化はこれをCNFに対して隠蔽することができます。これによりハードウェアのリプレースやヘテロジニアスハードウェア構成などにおいて、抽象化された準仮想化デバイスを利用することは、CNFの観点での検証工数を減らすことができます。

仮想化を活用した柔軟なインフラの選択

一般的にKubernetes/OpenShiftを用いることで、ソフトウェアはその動作環境の構築やネットワークの接続性の管理から開放され、よりアプリケーションの開発や運用に集中することができるようになります。

しかしながらモバイルコアネットワークのためのNFを運用する上においては、バニラのKubernetes/OpenShiftの構成のままでよいわけではありません。高い信頼性と高いパフォーマンスの発揮を求められるこれらのNFのためには複雑なネットワーク要件の吸収やCNF固有のパラメータチューニングなどがKubernetesに求められる場合が多くあります。さらにこれらの要件はCNFを提供するベンダ毎、CNF種別毎などで異なる場合があります。このためベアメタルサーバ上のKubernetesにおいて、これらの異なる要件を吸収することには課題が多くあります。

そこで仮想化を活用し、大きなベアメタルサーバをCNF毎に異なるKubernetes/OpenShiftの要件毎に分割したクラスタにし、それぞれのクラスタを搭載するCNF毎に最適化しデプロイすることでKubernetesの構成を柔軟に選択することができます。

抽象化による環境差分の吸収

また、仮想化によるサーバリソース、ネットワークリソース、ストレージリソースなどを抽象化は環境差分を吸収し隠蔽することができます。

例えば準仮想化デバイスであるvirtioなどを利用することで利用するハードウェア種別に依存しない高パフォーマンスなソフトウェア実装の仮想ハードウェアをKubernetes/OpenShiftコントローラノード、ワーカノードに提供することができます。

さらにvCPUの数やメモリの容量、ワーカノードに接続するNICの本数、ストレージの容量と数など、細かなパラメータを各環境で合わせることができます。

これにより、NFベンダのベリフィケーション環境とオペレータのプロダクション環境や異なるオペレータ同士の環境も合わせることができるようになるとともに、ハードウェアに非依存となるため異なるハードウェアを用いる場合での検証パターンについても削減することができます。

経済的メリット

このようにCNF毎に最適なKubernetes/OpenShiftアプリケーションの動作環境をCNF毎に用意することで次のような経済的なメリットを生むことができます。

- ハードウェアリプレイスやヘテロジニアスなハードウェアの選択を仮想化による抽象化で隠蔽し、ハードウェア検証工数を短縮できます。
- CNFに最適なインテグレーション済みのKubernetes/OpenShiftを用意することで、CNFのインテグレーションにかかる時間とリソースを節約できます。
- NFベンダのラボ環境とプロダクション環境をそろえることができます。

2.3.3 シンプルなオペレーション

CNFとVNFの運用の統一化

CNFが登場してもなお、NFだけではなく通信事業者におけるO&M装置群やOSS/BSSなどの周辺設備に広めた場合には仮想マシンが必要となるユースケースは多数存在します。NFにおいてもコンテナやKubernetesを使わない仮想マシンでの実装が必要なケースが存在します。また4G以前の設備や音声系の設備などではVNFの実装のまま運用されるものも多くあります。

このように仮想マシンのニーズがある状況において購入するハードウェアやハードウェアを導入した後のプロビジョニングをプライベートクラウドとしてのOpenStackを挟むことでインフラストラクチャの運用を統一化できます。また、これによりCNFとVNFの間でのハードウェアリソースの融通も容易になります。

さらに大規模なVMを利用したVNFに対してはハードウェアの抽象化によるHA機能でのサーバの救済やハードウェア故障時のLive migrationによる負荷の事前の退避などのオペレーションが存在します。そのためネットワークオペレータはShift on Stack構成を用いることでCNF/コンテナとVNF/VMの間でこれらのハードウェアに対する扱いを統一化することができます。

リソースの分割による責任分解点の定義

マルチベンダ構成でKubernetes/OpenShiftを利用することにはいくつか制限事項があります。一般的にKubernetesでマルチテナンシをとるためにはNamespaceを用いた分割が用いられます。しかしながらこの方法にはいくつかの制限事項があります。

1つ目はセキュリティ面および設定面での競合です。高い可用性や性能が求められるCNFではコンテナが動作するオペレーションシステムのLinux Kernelパラメータをそれ専用にチューニングすることが求められる場合があります。またより高い性能を求めるためにSR-IOVなどのハードウェア技術をダイレクトに利用する場合があります。この場合Kubernetesのユーザ権限としてPrevillagedなユーザ権限が必要となってしまい、さらにCNFベンダ毎に利用するLinux Kernelのパラメータの変更が競合する可能性があります。

2つ目はネットワークリソースの分割です。KubernetesにはCNIというネットワークを動的に設定したりServiceなど、ネットワークインフラを意識することなく動作する機能がありますが、これらのネットワークはパブリッククラウドのVPCやOpenStack Neutronなどにおいて提供されている仮想ネットワークのように、ネットワークを厳密に分割する技術とは異なります。そのため、ネットワークをコンテナの要求毎に分割したり接続したりすることはできず、コンテナが外部とつながるネットワークはフラットに扱う必要があります。

これらの課題に対してIaaSとしてのOpenStackを導入し、Kubernetes/OpenShiftクラスタを積極的に分割することで責任分解点を明確に分解することができます。

Kubernetes/OpenShiftクラスタのアップグレードの簡素化

Kubernetes のアップストリームでは年に3回新しいメジャーバージョンがリリースされ、Kubernetesのダウンストリームディストリビューションである Red Hat OpenShift Container Platformもこれに従っています。OpenShiftではこのアップストリームのサポート期間をエンハンスし、6カ月のフルサポートと12カ月のメンテナンスサポートを提供し、Extended Update Support(EUS)として特定バージョンで最大24カ月のサポートを実現しています。

このようにKubernetesは定期的に新しいバージョンにアップグレードしていく必要があり、同時にCNFもこれらのKubernetesのバージョンに追従して、CNFのバージョンアップも行っていく必要があります。通信を途絶えることなく提供することが求められるネットワークオペレータでは、この定期的に発生するバージョンアップを品質高くかつ、素早く行えるようになることが運用の中で重要なファクタです。

しかしながらOpenShiftクラスタをインプレースアップグレードすることは途中のアップグレード過程においてCNF/OpenShiftのバージョンが混在する状態を考慮したり、コンテナの再起動タイミングを調整したりする必要があり、滞りなくバージョンアップするというのは非常にチャレンジングです。

また基本的にOpenShiftはバージョンダウンできないため、バージョンアップにおいて問題が起きた場合に、元のバージョンのCNF/OpenShiftクラスタに切り戻すことも同様に難しくなります。

これに対してOpenStackによるShift on Stack構成は自動化によるブルーグリーンアップデートを実現しやすくシンプルなバージョンアップを容易にします。一般的にベアメタルサーバの作成や削除よりも、仮想サーバの作成や削除の方がより簡易になります。OpenStackというすでに完成された仕組みを利用することができるため、ベアメタルサーバプロビジョニングシステムや、ネットワークのオーケストレータを新たに作る必要もありません。

新たなバージョンのOpenShiftクラスタと新たなバージョンのCNFを立ち上げ、ネットワーク側で切り替えることなどで非常に簡易にバージョンアップを実現することができます。

2.3.4 自動化の推進

OpenStackによるクラウドコンピューティング

OpenStackの提供する価値の本質は仮想化の提供だけではありません。OpenStackはクラウドコンピューティングを実現するための、クラウドインフラストラクチャソフトウェアです。これにより仮想化などによるハードウェアの抽象化だけではなく、ユーザはクラウドインフラストラクチャが提供するWeb APIを通じて、自ら必要なリソースを利用することができるようになります。

これらのAPIを利用することで、仮想マシンや仮想ネットワークといったリソースをソフトウェアで管理することができます。ネットワークファンクションに必要となる複雑なネットワーク要件等をRed Hat Ansible Automation PlatformやOpenStack Heat、その他の自動化ソフトウェアなど、様々なツールを利用してソフトウェアデファインドできます。

特にこれらの自動化は次のようなユースケースでテレコクラウドにおいては強力に効果を発揮します。

IaCでのインフラリソース管理の自動化

OpenStackによりサーバリソースや、ネットワークリソース、ストレージリソースといったサーバ情報をOpenStackの持つAPIで定義が可能です。これによりHelmやHelm VluesなどのCNFの構成情報だけではなく、Kubernetes/OpenShiftクラスタのためのサーバ、ネットワーク、ストレージのリソース情報の構成情報についてもOpenStack HeatやAnsible、Terraform/OpenTofu等を用いてGitなどで管理することができます。

インフラストラクチャのデプロイ/スケールの自動化

IaC管理されたインフラストラクチャはそのパラメータを変更することで同じ構成のコピーを簡単に作ることができます。一般的にモバイルコアネットワークは大規模かつ複数デプロイされ、また、需要の増加に合わせて拡張を行います。

Kubernetesの各種プラクティスに従ってコンテナ化されたCNFはそのネットワークファンクションのデプロイやスケーリングが容易にできます。しかしながらそれを実現するためにはCNFを動作させるためのKubernetes/OpenShiftのワーカノードが事前に必要となります。OpenStackなどのクラウドインフラストラクチャがあれば、これらのOpenShiftクラスタのデプロイやスケーリングを自動化できます。

2.4 "Shift on Stack"の実践

2.4.1 テクニカル概要

本チャプターでは、OpenShift on OpenStack の実践について紹介いたします。

OpenShiftをOpenStackの上に使い利用することはOpenShiftにおいても長らくサポートされる構成であり一般的なものです。しかしながら通信業界においてはベアメタル OpenShiftが注目されていることもあり、通信アプリケーションをShift on Stackで構築した事例の紹介は多くありません。

このホワイトペーパでは通信アプリケーションが利用する上での考慮点や注意点について記載していきます。

複数の視点での検討

Shift on Stack を実践していく上においては当然、インフラストラクチャであるOpenStackの観点とクラウドネイティブプラットフォームであるOpenShiftの両方の観点が必要になります。

OpenStackの観点ではまず一般的な ネットワーク仮想化(NFV) としての観点が必要です。仮想マシンを利用するNFVシステムはVNF(Virtual Network Function)と呼ばれ、そのVNFを動かすためのチューニングや設定の観点は非常にポピュラーなものとなっています。これに加えてOpenShift/Kubernetesの持つ特徴についてもOpenStackとしてのケアが必要となります。

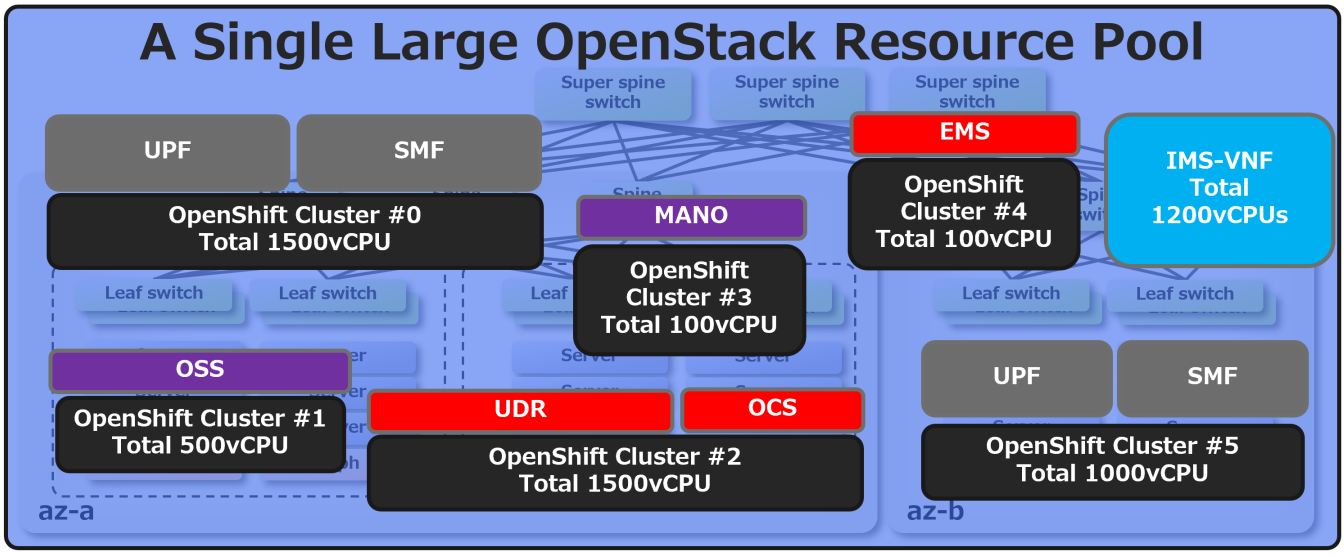
Kubernetesはそれ単体で様々な機能を提供するのではなく、環境に応じてプラガブルにさまざまなプラグインを組み合わせることが非常に得意です。そのために様々なプラグインを導入しやすいように、ネットワークには Container Network Interface(CNI)や、ストレージにはContainer Storage Interface(CSI)と呼ばれるインタフェースが定義されおり、様々な外部コンポーネントを利用することができるようになっています。これらのプラグインの一つにはOpenStackの機能を最大限に活用するCloud Provider OpenStackというプラグインのコレクションがあり、これを利用することになります。この際に通信アプリケーションであるCNFの特性なども考慮し、どのコンポーネントをOpenStack側で準備すべきかといった観点での検討がVNFの時と比べて追加で必要となります。

またOpenShiftに関しても当然通信のワークロードを扱う上でのチューニングや検討事項が存在します。このOpenShift/Kubernetes観点でのチューニングについて基本的にベアメタルOpenShiftと同じため、これらのナレッジをそのまま利用できます。しかしながら、CNFのワークロードにおいてOpenShiftをOpenStack上で動作させる上での特異な考慮事項はいくつかあります。例えばインストール方法についても通常のワークロードであればIPI (Installer Provisioned Infrastructure)インストールが用いられますが、ネットワークを多数分割する必要があるCNFのワークロードではUPI (User Provisioned Infrastructure)の選択が必要な場合があります。また、ベアメタルでは非常に大きなベアメタルサーバを有効に活用するための観点がありますが、仮想化により、リソースの切り出しが可能なOpenStackの上においては異なる考え方が必要になります。さらに、ベアメタルサーバではSR-IOVのNIC分割などの管理はOpenShiftで行う必要がありますが、Shift on StackにおいてはOpenStack側でSR-IOVの管理が行われます。

これら上げた事項はあくまでも一例になりますが、このように OpenStackとしての視点、OpenShiftとしての視点、またそれぞれがまじわる部分についての視点での検討が必要となります。

基本的な戦略

大きなOPENSTACKクラスタとOPENSTACKマルチテナント運用



Partitioning a Large OpenStack Resource Pool

1つのOpenShiftクラスタのために1つの専用のOpenStackクラスタを作ることは有効な手段とは言えないでしょう。

すでに紹介した通り、OpenStackを使うことによるメリットは以下の点です。

- 効率的なインテグレーション
- シンプルなオペレーション
- 自動化の推進

CNFだけではなくVNFやITワークロードを含め、通信事業者のインフラストラクチャのオペレーションを統一化し、運用稼働を削減していくためにはOpenStackクラスタをある一定の大きさまでスケールさせることが必要となってきます。そのために、マルチテナントで運用し統一的で一貫性のあるプライベートクラウドをOpenStackで実現することがまず必要であり、OpenShiftによるCNFの効率的なデプロイや運用はその土台の上に初めて成り立ちます。

CNF毎の要求差分の吸収はOPENSHIFTのクラスタのデプロイを分割し実現する

前述の通り仮想化をしているメリットはハードウェアの抽象化とIaCによる自動化の実現です。ハードウェア構成やネットワーク構成を仮想化により柔軟にカスタマイズできます。CNFが求めるハードウェア要件、ネットワーク要件はCNFごとや、CNFベンダごとに異なります。これらの差分の吸収、柔軟なリソースのアサインはOpenStackが最も得意とする事の一つになります。やみくもにクラスタの構成パターンが増えることへの考慮は必要ですが、リソース分割や重畳が容易で柔軟なリソースアサインができる利点を活かし、OpenShiftクラスタを要件ごとに分割していきます。

OPENSTACKとOPENSHIFTの責任範囲を分割する

OpenStackとOpenShiftはそれぞれの役割を明確に分離します。

OpenShift/Kubernetesにもインフラストラクチャの設定を行う機能はありますが、OpenStack、OpenShiftそれぞれのどちらでそれらのこれらの機能を有効化するのかを明確に定めておくことは有効です。

詳細は以降に記載しますが大まかには以下の通りに機能を分担します。

設定カテゴリ	設定内容	OpenStack	OpenShift	説明
コンピューティング	CPU Pining	V	V	正しくCPU pinningするためには両方で実施する必要があります
コンピューティング	Hugepage利用	V	V	HugepageをGuestVMおよびコンテナ双方で適切に設定する必要があります
コンピューティング	NUMAの管理	V	-	NUMAは仮想マシンで分割しシングルNUMAのVMとしてOpenShiftは利用します
コンピューティング	HAリソースの確保	V	-	HA先のハードウェアリソースをOpenStackで用意します
ネットワーキング	高速なNWの提供	V	-	OVS-DPDKによる高速な仮想ネットワークもしくはSR-IOVを提供します
ネットワーキング	SR-IOVでのNIC分割	V	-	SR-IOVはOpenStackにより分割しVMにアタッチされ、OpenShiftからはhost-deviceとして扱います
ネットワーキング	ネットワークの冗長化	V	(V)[1]	SR-IOVを除くネットワークの冗長化はOpenStack側で実現します
ネットワーキング	ネットワークのQoS設定	V	-	Bandwidth limitやDSCPマークなどのQoSはOpenStack側で設定します
ストレージ	ボリュームの切だし	V	-	Persistent Volumeの実態はOpenStack Cinderにより提供されます
ストレージ	ボリュームの冗長化	V	-	Persistent Volumeの実態はOpenStack Cinderのバックエンドにより冗長化されます
ストレージ	ボリュームのQoS設定	V	-	Persistent VolumeへのIOPS/Bandwidth

設定カテゴリ	設定内容	OpenStack	OpenShift	説明
				の制限はCinderにより実現します
ストレージ	RWXボリュームの切だし	V	V	要件によりOpenStack Manillaか、OpenShift Data Faundationを選択します
その他	DNSレコードの管理	V	V	OpenStack Designate により管理します

このようにネットワークやストレージなどの機能や冗長構成はクラウドインフラストラクチャとしてのOpenStackにより提供されるため、Cloud Provider OpenStack [2] を効果的に利用することで OpenShiftで管理するリソースを減らすことができます。

Red Hat製品の使用と特徴

Red Hatのテレコクラウドソリューションは、VNFとCNFベースのネットワーク機能を並行して実行する能力を提供します。Red Hat OpenStack Platformは、サービスプロバイダーが大規模なネットワークを展開し、変化する顧客の需要に基づいたサービスやアプリケーションを提供する能力を提供します。テレコのネットワーク機能は、必要なパフォーマンスを提供するために最適な場所に配置されます。Red Hat OpenShiftは、サービスプロバイダーがテレコクラウドに対してクラウドネイティブのアプローチを採用し、アプリケーション開発の加速とハイブリッドクラウド環境全体での一貫性を実現するのに役立ちます。

RED HAT OPENSTACK PLATFORM

OpenStackプロジェクトが作られてから10年以上が経過しました。それ以来、Red Hat OpenStack Platform は通信事業者のクラウド環境をリードする技術になり、革新の推進力となり通信事業者の NFV 基盤のトップの一角となることができました。Red Hat の通信事業者のお客様の、およそ25億サブスクライバーの通信を Red Hat OpenStack Platform が支えています (2022年弊社調べ)。

- Red Hat OpenStack Platformは、主要な商用ディストリビューションです
- Red Hat OpenStack Platformは、コアデータセンターからエッジデバイスまでをカバーする、実証済みの統合技術に基づくスケーラブルで柔軟なテレコクラウド環境を提供します
- Red Hat OpenStack Platformは、テレコクラウドの重要なワークロードのための実証済みの基盤を提供します

RED HAT OPENSHIFT

Red Hat OpenShift は、通信事業者の近代的なネットワーク基盤に利用いただくことで、5G のネットワークサービスをより早く提供することを可能にします。OpenShift は企業利用用途の Kubernetes アプリケーション基盤です。OpenShift は複数のデプロイメントモデルを提供することができ、通信事業者が 5G サービス提供のための分散配置に必要となる構成の自由度を提供します。

- Red Hat OpenShiftは、主要な商用Kubernetesソリューションです
- Red Hat OpenShiftは、テレコクラウドアーキテクチャに必要な低遅延、予測可能な遅延、高帯域幅、分散アーキテクチャを提供します
- コンテナベースのプラットフォームは、テレコクラウドインフラストラクチャを進化させるためのスケーラブルで柔軟な方法を提供します

その他のソフトウェアの利用

- Red Hat Ceph Storage
- Red Hat OpenShift Data Foundation

実践

このホワイトペーパでは次に示す項目について、次のチャプター以降においてShift on Stackで実践の詳細を紹介していきます。

OPENSTACK クラスタデザイン

Shift on Stackを実践する上において、必要となるOpenStackの設計を示します。以下に概要を紹介します。

- 利用するOpenStackにコンポーネントの選択

利用するOpenStackコンポーネント	用途
Keystone - Identity Service	認証認可に必要
Glance - Image Service	イメージ管理のために必要
Nova - Compute Service	仮想マシンの稼働に必要
Placement - Placement Service	仮想マシンの稼働に必要
Neutron(ML2/OVN) - Network Service	仮想ネットワークの管理に必要
Cinder - Block Storage Service	RWO Persitent Volumeに利用
Designate - DNS Service	IngressのためのDNSサーバに利用
Heat - Orchestration Service	UPIによるインストール/自動化に利用
Octavia - Load balancer Service	UPIのためのLB/Service type: LoadBalancer等に利用
Swift/Ceph - Object Storage Service	データのバックアップ先等に利用

- コンピュートの設定
  - オーバコミットするサーバタイプ、CPU Pinningが必要なサーバタイプ等を分割しサーバの利用効率を高めます
  - NUMAノードやSR-IOVの利用を意識した配置設計を行います
- アベイラビリティゾーンの設定
  - アベイラビリティゾーンを分割し耐障害性を高めます、OpenShiftのデプロイはアベイラビリティゾーンを意識します
  - コンピュート、ネットワーク、ストレージ、ロードバランサそれぞれでのアベイラビリティゾーンを設定します
- ネットワークのチューニング
  - ML2/OVNおよびOpen vSwitch - DPDKの活用を行い、SR-IOVだけに頼らない高速なネットワーキングを実現します
  - SR-IOVの効果的な利用方法を紹介します
- QoS設計
  - ストレージやネットワークのQoSについて設計し、ノイジーネイバー問題を緩和します
- スケーリング
  - 多数のOpenShiftクラスタを管理することがでいるスケーラビリティに関する設計を行います
- アドバンス
  - より効果的な利用方法をKDDIの実践例などを交えて紹介します

OPENSHIFT クラスタデザイン

Shift on Stackを実践する上において、必要となるOpenShiftの設計を示します。以下に概要を紹介します。

- インストレーション
  - UPIによるインストールを行います
  - UPIのインストールを簡便化するためにOpenStack Heatによるインフラの作成とAnsibleとOpenShift-Installerによるインストレーションの自動化を紹介します
  - OcataviaによるLBaaSやDesignateによるDNSaaSの活用を紹介します
- 仮想マシンの設計パターン
  - CNFのユースケースを分析し、OpenShiftクラスタ、ワーカノードの構成パターンを絞り込みます
  - Neutronのトランクポートを活用し、タグVLANが必要なユースケースに対応します
  - Neutornのトランクポートを活用し、OVS-DPDKによるパケット転送の性能を最適化します
- ネットワークのベストプラクティス
  - 次の機能を利用する場合のベストプラクティスを紹介します
    - SR-IOV
    - MetalLB
    - 複数VRF接続
    - Floating IPによるNATの利用
- ストレージ要件への対応
  - ODFとエフェメラルディスクを活用することで高信頼なRWX PVを実現します
  - 静的プロビジョニングと動的プロビジョニングの使い分けとステートレスワークロードの移動
- CNFアップグレード/OpenShiftアップグレード
  - 新規クラスタを構築してのマイグレーションを実現します
  - ステートレスワークロードの移動を検討します

---

1. SR-IOVのみ ↩
2. https://github.com/kubernetes/cloud-provider-openstack ↩

2.5 まとめ

このホワイトペーパはテレコクラウドにおける Red Hat OpenShift Container Platform をRed Hat OpenStack Platform とインテグレーションして利用する OpenShift on OpenStack, "Shift on Stack"構成のベストプラクティスを提供を目指し執筆を進めています。

今回オープン技術の登場の結果、選択の自由度が増し、結果的にフラグメント化しているテレコクラウドの現状に対し、Red Hatのプロダクトの視点からも様々なオプションを提供できることを紹介し、世界中のテレコムオペレータの参考となることを期待し、本ホワイトペーパを現在の段階において早期に公開しました。

現状のこのホワイトペーパでは次のことを紹介しました。

- クラウドネイティブの流れのなかで生じているインテグレーションの複雑化、製品ライフサイクル短期間化化といった通信事業者の課題を共有しました。
- 次にこの課題に対してRed Hat OpenStack Platform によるプライベートクラウドと、マルチクラウド、ハイブリッドクラウドを実現する Red Hat OpenShift を組合せることで、レバレッジを実現しそれらの課題を解決できることを示しました。
- そしてより具体的にこれらの製品をどのように利用するのか、OpenShiftをOpenStack上で利用する"Shift on Stack"構成において どのようなチューニングや考慮点があるのかのベストプラクティスの概要を紹介しました。

今後、Red HatとKDDIでは2024年第4四半期頃をめどに本ホワイトペーパのコンテンツをを拡充していき、より実践的かつ具体的なベストプラクティスの紹介を行う予定です。その中ではテレコクラウドにおけるもう一つのオプションであるベアメタル上でのOpenShiftの利用と性能検証を含めて比較する予定としています。さらに、ベンチマークや机上検討だけではなく、実際のネットワークでの活用事例として、日本の大手通信事業者であるKDDIでの5Gコアシステムでの具体的な活用事例を紹介する予定です。

Red HatとKDDIでは今後もオープンソースソフトウェアを活用し、情報をオープンにしていくとこで通信業界におけるイノベーションの加速に貢献していきます。本取り組みに関してご興味のある方は、telco-cloud@kddi.com までお問い合わせください。

2.6 寄稿者

2.6.1 Red Hat

- Nick Satsia
    - Global Solution Architect in the Teleco industry
- Kazutoshi Hayakawa
    - Solution Sales Specialist in Teleco industry

2.6.2 KDDI株式会社

- Hiroshi Tsuji
    - Seinior Expert of Network and Cloud Platform
- Takuya Sawada
    - Deputy Head of Node Technology Department

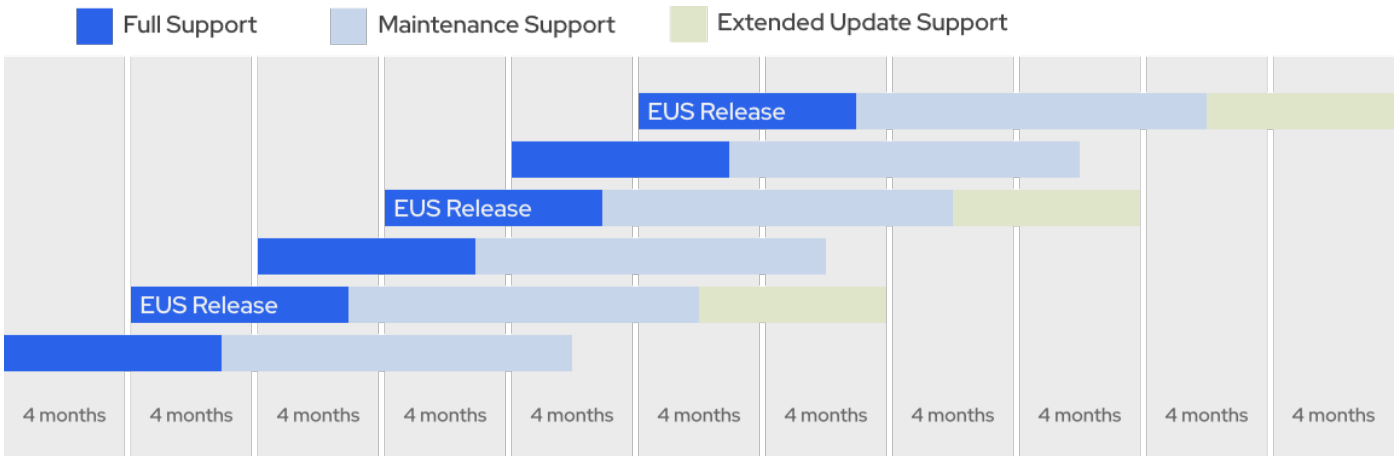
2.7 改版履歴

バージョン	日付	改版内容
v1.0	2024-05-07	初版

2.8 付録

2.8.1 付録A: 課題の詳細

短いライフタイムとアップグレード



Official OpenShift Life Cycle[†]

Kubernetes のアップストリームでは年に3回新しいメジャーバージョンがリリースされるリリースサイクルを採用しています。 そして12カ月の通常サポートと2カ月のメンテナンスサポートの計14カ月のコード修正が提供されます。 Red HatではこのKubernetesをアップストリームとした、Red Hat OpenShift Container Platform において、アップストリームのサポート期間をエンハンスしています。 6カ月のフルサポートと12カ月のメンテナンスサポートを提供し、Extended Update Support(EUS)として特定バージョンを対象に6カ月のサポートを提供しています。 これにより標準で最大24カ月のサポートを実現しています。

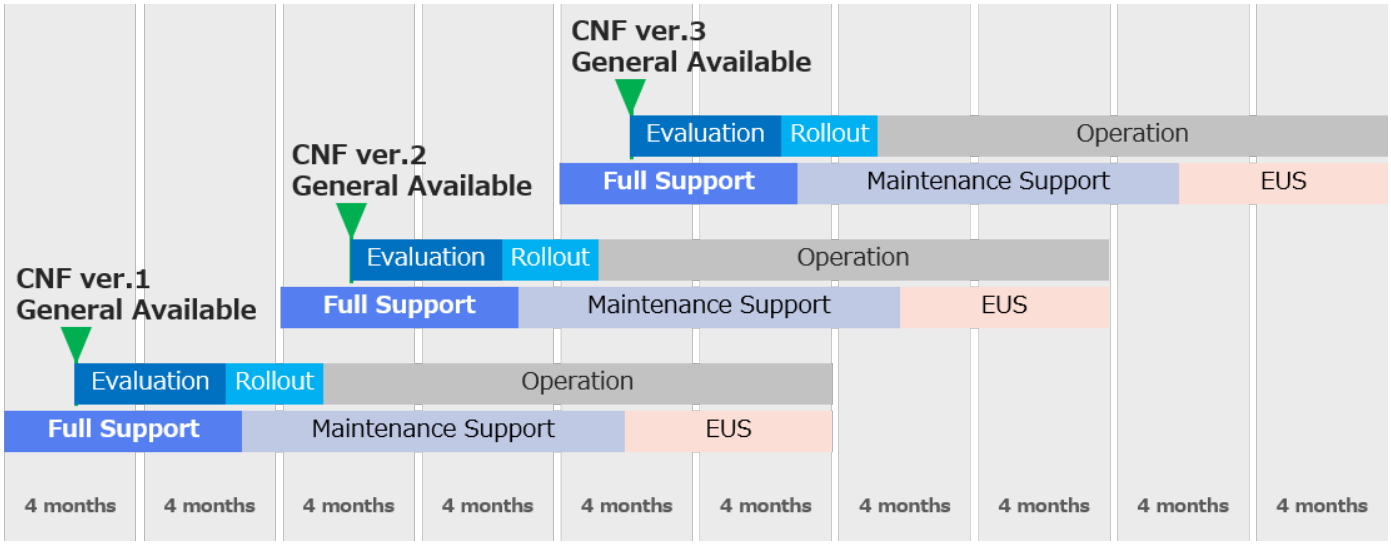
このようなRed Hat OpenShiftによる長期間のサポートの提供でも、 これはこれまでRed Hat Enterprise Linuxが提供しているようなエンタープライズレベルでの10年にわたる長期サポートや、 Red Hat OpenStack Platformが提供しているような4年のサポートに比べても非常に短いものとなっています。

このような短いソフトウェアのライフサイクルは アプリケーションを動作させるプラットフォームソフトウェアである KubernetesやRed Hat OpenShiftへの新しい機能の取込みや改善、古い機能の廃止などを行いやすくなっており、 Kubernetes というソフトウェアの新陳代謝を早めています。

事業、技術ともに変化が激しい環境において頻繁にCI/CDが実行される環境が整っており、サービスを提供するアプリケーションのライフサイクルがプラットフォームソフトウェアより圧倒的に早い、ウェブ企業などではこのようにプラットフォームソフトウェアの更新が早いことは有効な手段です。

しかしながらこのようなKubernetesの短いソフトウェアライフサイクルは 安定したインフラストラクチャが求められる通信領域においては課題もあります。

モバイルシステムには4G/5Gといった世代が存在し、約10年ごとにそのしくみに対して大きな変更がされます。 また多くの加入者が途切れなく利用するモバイルコアネットワークにおいては変更によりサービス停止を生じないようにするために、ネットワークに変更を加えることに対して慎重になります。 そのためモバイルオペレータは利用するハードウェアやソフトウェアのベンダによる保守限界や問題対処のためにのみハードウェアやソフトウェアの更新を行てきたのがこれまでの運用であり、 一般的には一つのソフトウェアやシステムに対しては多くても年に数度の更新しか行われませんでした。



Example life cycle between OpenShift and CNF

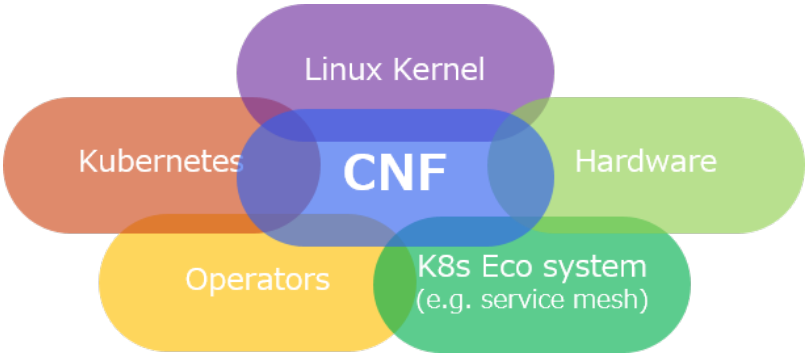
このようなテレコムアプリケーションベンダやテレコムオペレータにとって、 Kubernetesの短いライフサイクルに適合し、 2年毎にそのソフトウェアバージョンをKubernetesとテレコムアプリケーションの両方を同時に上げていくことは非常にチャレンジングな課題です。

Kubernetes バージョンとCNFバージョンの強い紐付け

前セクションで説明した通り、Kubernetesの短いライフサイクルに対してソフトウェアバージョンをアップグレードして行くことは大きな課題です。

さらにネットワークオペレータは安定した通信を低コストに提供していくために、 ネットワークファンクションであるテレコムアプリケーションに非常に高い性能要件や可用性を求めます。 このネットワークオペレータの求める要件を満たすために、 CNFを提供するテレコムアプリケーションベンダは、 Kubernetesやオペレーティングシステムに対するチューニングを入念の行う必要があります。

その為にテレコムアプリケーションベンダは自身のラボにおいてCNFのソフトウェアの検証を入念に行います。 そしてラボで検証され見つけられた設定内容をオペレータ環境でも極力再現することをオペレータに対して求めます。 そのためネットワークアプリケーションベンダは、KubernetesやOpenShiftのバージョンを含めたCNFが動作する、動作条件を明確に定め、細かなパラメータ設定をオペレータに対して明確に指定することが多くあります。 このようなチューニングはDPDKを用いたりSR-IOVを用いたりする大容量のモバイル加入者のトラフィックが通る、 データプレーンCNF（UPFなど）ではより顕著です。



CNF has many strong dependencies

このような状況から、テレコムアプリケーションとKubernetesやOpenShiftには強い依存関係が存在します。 オペレータにとってベンダが検証したプリケーションとKubernetes、OpenShiftのバージョン、パラメータのチューニングに合わせてこれらのバージョンや設定を採用することが、ネットワークファンクションを導入するにあたって最もリスクが少なく、そして確実な方法と言えます。

しかしながらこの方向性はマルチベンダでコアネットワークを構成する上では課題があります。それは、ベンダ毎、CNFの製品ごとに利用するKubernetes, OpenShiftのバージョンが異なったり、 そこで利用する製品のパラメータやチューニングノウハウが違うためです。 その為、Kubernetes/OpenShiftなどのコンテナプラットフォームのレイヤを共通化し、 競争性を確保したり、冗長性を確保するためにコアネットワークをマルチベンダで構成するためには、 これらのCNF毎のバージョンや設定の差分等を吸収し、インタフェースをそろえてていく必要があります。

実際にテレコム業界をリードするようなTier 1オペレータはこれらの取組みを強力に進め、効果を出しています。 しかしながら、これらを実現するために力は多大なエネルギーを必要とするため、 全てのオペレータにおいて彼らと同様の戦略をとれるわけではないこともまた現実的な課題です。

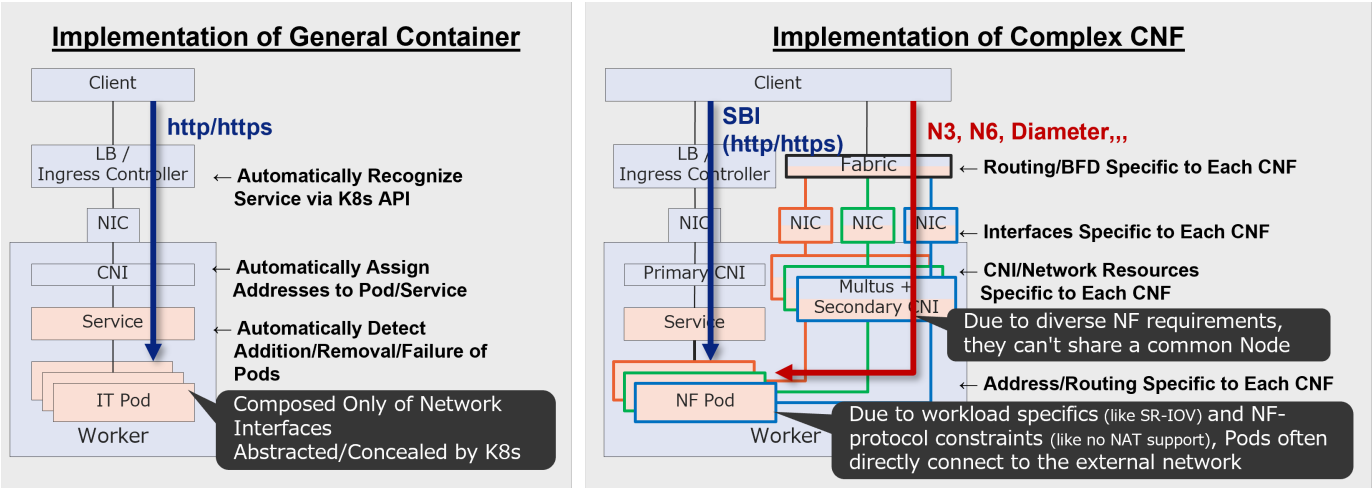
複雑なネットワーク要件

コアネットワークで利用されるCNFにおけるネットワーク要件はKubernetesの最も一般的なユースケースで利用されているような通常のウェブアプリケーション等とは大きく異なります。

Traditional network requirements

特にユーザのセッションを終端するCNFやユーザプレーンの通信を行うCNFなどでは 素早いアプリケーションの冗長化切替が求められます。その為にCNFによってはアプリケーションレベルでの即時での切替を実現するために、 VRRPなどのL2での冗長化を行ったり、BGPやBFDを組み合わせたL3での冗長化を行っています。 その他にもSCTPのマルチホーミングを利用するために複数のIPアドレスを持ったり、アプリケーションの設計上からVLANインタフェースをCNFが持つ必要があったりします。 さらに、従来からの通信の分野では、シグナリングとデータプレーン、OAMトラフィックやBSSから得られるService orderなど、通信の種別により 利用するインタフェースやIPアドレスやVRF、ネットワーク機器を分けて作られてきたこともありました。

このような要件は一つのVPCに接続すればよく、外部向けトラフィックはNAT/NAPTを利用して通信できれば良いクラウドネイティブな考え方とは大きく異なります。



Compared to general containers, CNFs require more complex implementation[††]

更にデータプレーン系のCNFではSR-IOVとDPDKを用いる場合もあります。 SR-IOVはPCI-Eデバイスとして論理分割されたネットワークインタフェースを扱う技術です。 DPDKはネットワークインタフェースをカーネルドライバではなく、ユーザランドドライバで扱う技術です。 そのためSR-IOVとDPDKを用いると、CNFのコンテナ内部で直接PCI-Eデバイスを扱う必要があります。

このようなテレコムアプリケーションの特性や、運用上の要件から、 テレコムにおいてKubernetesを用いる場合、一つのPodに対してプライマリCNIの他に、 VLANやSR-IOV等の複数のネットワークを利用できる必要が出てきます。 そこで多くのCNFはCNIとしてMultusを利用し、複数のネットワークインタフェースを接続します。

しかしながら、2024年の現時点においてKubernetesがIPAMとしてアドレス管理を行ったりサービスディスカバリを行うのはあくまでもプライマリCNIだけです。

また、プライマリCNIについても仮想ネットワークを切り出したりすることは基本的にできず、 全てのコンテナに対してフラットなネットワークを提供し、コンテナ間のアクセスコントロールにはRBACの仕組みを利用してフィルタリングを行います。

このようにKubernetesだけを用いる場合、CNFが接続するセカンダリCNIのネットワークについては、Kubernetesの外部でプロビジョニングされたうえで、 Kubernetesは適用されたCNFのためのDeploymentやStatefulSetのマニュフェストに対して、そのプロビジョニングが正しく行われた前提動作します。

このように テレコムアプリケーション やテレコムオペレータ の複雑なネットワーク要件を Kubernetesの標準機能だけでは解決することはできないため、自動か手動のオペレーションでこれらのネットワーク要件を満たす必要があります。

CNFとVNFの混在環境

新世代のネットワークファンクションからクラウドネイティブ技術を用いるCNFでデプロイされることが多くなりました。 しかしながら、依然としてCNF化されないネットワークファンクションも多くあるでしょう。

例えば、4Gなどの旧規格のネットワークファンクションや IMSといった変化が少ないネットワークファンクション、 固定ネットワークのコアネットワーク機能、 OSS/BSSといった運用監視のための機能などは既存の物理サーバ、仮想サーバで構成されたネットワークファンクションを 変化の少ない仮想サーバでリニューアルすることも多くあるでしょう。 これはContainerizationがただコンテナ技術を利用するだけではなく、システムアーキテクチャも マイクロサービス化するなどクラウドネイティブ化することが重要だからです。

これらの旧来のアプリケーションはアプリケーションがもともとモノリシックなアーキテクチャをとっているため、 Containerizationするためにはアプリケーションそのものの変化が必要になります。 しかしながらモノリシックなアプリケーションをマイクロサービスにすることは簡単なことではありません。 そして、Containerization、クラウドネイティブ化、マイクロサービス化の目的は変化や変更をより安全に、そして小さく実行するためです。 その為、変化が乏しいこれらの旧来から存在するネットワークファンクションやシステムでは 必ずしもクラウドネイティブ技術をフルスケールで採用する必要はないかもしれません。 それよりもむしろ、変化を極力起こさず少ない労力、少ないコストで持続的に提供していくことがビジネス上重要であることはあります。

クラウドネイティブは唯一無二のベストな技術選択ではなく、我々がとりえるベストな選択肢の一つであるということを意識することは重要です。その為、テレコムオペレータにおいては、CNFだけではなくVNFについても今後も広く運用されていくでしょう。 テレコムオペレータのプラットフォームにはCNFとVNFを共存させることができることが求められます。

---

1. https://access.redhat.com/support/policy/updates/openshift ↩
2. https://www.redhat.com/it/blog/building-cnf-applications-with-openshift-pipelines ↩