White Paper

Communications Service Providers
Cloud-Native Network Functions

Understanding How The Core Network Is Evolving to Cloud-Native

How can communications service providers manage the co-existence of virtual machine-based virtualized network functions and cloud-native network functions as they prepare for 5G?

Author

Muthurajan Jayakumar (M Jay)
Cloud-Native Platform Software Engineer,
Network Product Group, Intel

Table of Contents

Introduction

When the telecommunications industry migrated from physical purpose-built appliances toward network functions virtualization (NFV), there was a period of time where networks consisted of a mix of both approaches—creating both opportunity and complexity. We are at a similar inflection point today, where the promise of 5G hinges on adoption of cloud-native technologies—notably cloud-native network functions (CNFs) and microservices—but there will still be plenty of virtualized network functions (VNFs) deployed on legacy virtual machines (VMs). This mixed landscape can present communications service providers (CoSPs) with migration and integration challenges associated with network infrastructure, management and orchestration, and testing/integration.

With ample experience in both NFV and more recent cloud-native core network developments, Intel can offer some perspective on these challenges, through its deep relationships with the ecosystem, such as:

- Being an active member of the Cloud Native Computing Foundation (CNCF)
- Collaborating with the ecosystem to enhance technology solutions, such as contributing to the Cloud Infrastructure Telco Task Force
- Working with many CoSPs to develop commercially viable 5G platforms through the Intel® Network Builders program.

Intel is working with the open source communities to develop Kubernetes features that fill some of the gaps between native Kubernetes capabilities and telecom requirements. This paper provides useful information about how CoSPs can navigate this transitional period as they move toward a cloud-native future.

Prioritizing Cloud-Native Network Function (CNF) Deployment

Some of the first questions CoSPs must answer are which applications should migrate to cloud-native and meet cloud-native principles, which applications should be implemented using containers and microservices, and which applications may be implemented using a VM. To answer these questions, let's look at some of the characteristics of VMs versus containers, and how these characteristics may affect each of the different network planes.

- **VNFs are implemented on VMs.** Each has its own operating system (OS) instance. While this can make startup time slow compared to a containers environment, application functionality is kept completely within the VM and different threads can communicate with each other with little overhead.

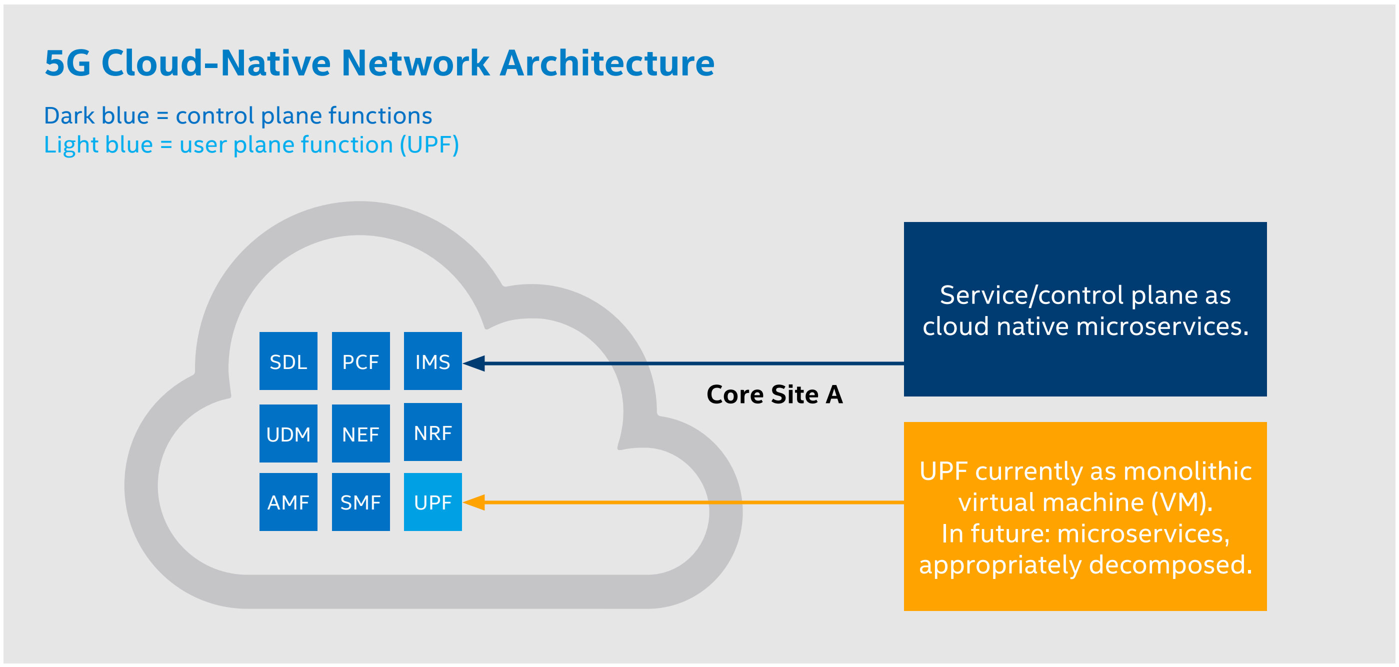
5G Cloud-Native Network Architecture
Dark blue = control plane functions
Light blue = user plane function (UPF)
SDL
PCF
IMS
UDM
NEF
NRF
AMF
SMF
UPF
Core Site A
Service/control plane as cloud native microservices.
UPF currently as monolithic virtual machine (VM). In future: microservices, appropriately decomposed.

Figure 1. A proposed transitional 5G network architecture keeps the user plane function (UPF) on virtual machines (VMs), but will transition to appropriately decomposed microservices over time.

- **CNFs are implemented on a microservices architecture, meaning that they are decomposed into "small building blocks" and then packaged into containers.** Containers share an OS kernel, reducing startup time overhead. However, communication between containers may introduce latency, which can in turn decrease throughput.

Adopting a microservices architecture lets CoSPs take full advantage of the operational benefits of the cloud computing model. When successfully implemented, this transformation can improve the speed, agility and resilience of service development and management processes. In the telecommunications context, a cloud-native microservices architecture delivers tangible benefits in terms of reduced capital expenditure (CapEx) and operational expenditure (OpEx), faster time-to-market service development and deployment, and scalability in both east-west and north-south traffic.

To get started on the path to cloud-native and its operational advantages, CoSPs should look for simple services that are not particularly sensitive to performance metrics (such as latency). This type of service is fairly easy to deploy as a CNF with microservices architecture, and lets CoSPs develop the necessary cloud-native skills over time.

But some aspects of the network are highly sensitive to performance, such as the user plane/data plane. The current deployment as a monolithic application makes sense because it helps reduce communication latency and thereby increase throughput. But on the other hand, a monolithic deployment means that when scaling is required, the entire function (all threads) must scale in concert—which can lead to inefficiency. In contrast to the user plane, the control plane and service plane are less sensitive to latency and are therefore more suited for containerized microservices (see Figure 1).

Beyond the distinction between VNF and CNF deployments based on performance requirements, operational advantages of CNFs may drive a combined CNF/VNF network. For example, an operator may use CNFs with a "managed service." With a managed service, the operator is responsible for supplying and maintaining the infrastructure and processes for a particular service, such as those associated with the Internet of Things (IoT), big data analytics, and security. This enables the enterprise customer to more easily and cost-effectively take advantage of new technology and operational transformation.

The Path to the Future

Kubernetes was not designed with high-performance networking as the focus. For example, a network router needs at least two ports. But native Kubernetes has only one network interface per pod. This is like an airplane, automobile, bicycle, and pedestrian all sharing a single-lane road. Intel has been working with its partners and the CNCF community to develop Multus (Multiple Network Interface) and other Kubernetes advancements that can assist with implementation of network applications, help reduce I/O latency and support the evolution of even the user plane function (UPF) to a distributed UPF (dUPF). A dUPF (see Figure 2) can scale a single thread (or two or three) without having to scale the entire function, helping CoSPs deliver new apps and services more quickly compared to a monolith UPF.

CoSPs can achieve a balance between performance and scalability by keeping the data path of the UPF as a single monolith microservice and decomposing other less performance-sensitive UPF modules like logging and tracing into multiple microservices. That is, functionalities in the UPF that are highly sensitive to performance (called the "fast path" or "critical path") should not be decomposed into multiple

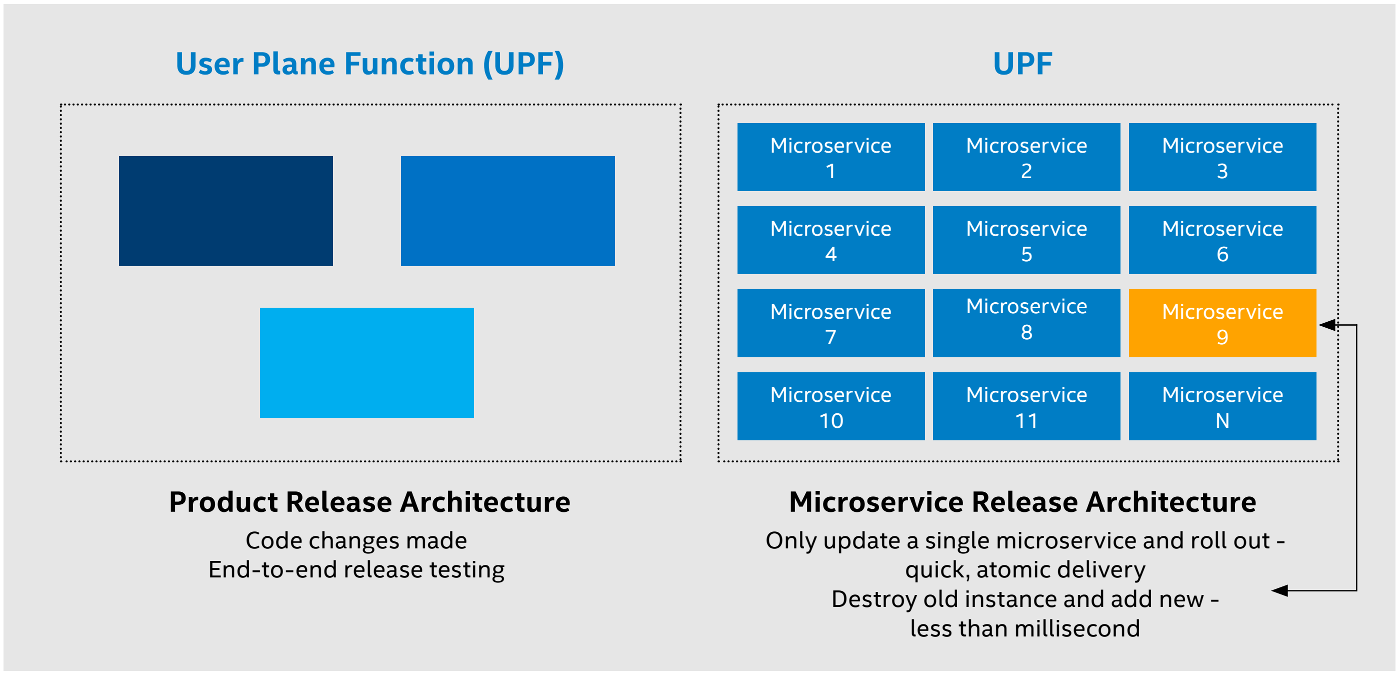
User Plane Function (UPF)
UPF
Microservice 1
Microservice 2
Microservice 3
Microservice 4
Microservice 5
Microservice 6
Microservice 7
Microservice 8
Microservice 9
Microservice 10
Microservice 11
Microservice N
Product Release Architecture
Code changes made
End-to-end release testing
Microservice Release Architecture
Only update a single microservice and roll out - quick, atomic delivery
Destroy old instance and add new - less than millisecond

Figure 2. A distributed user plane function (dUPF) can lead to rapid development, time to market, and deployment.

microservices—they should only be a monolithic image. But functionalities in the UPF that are less performance-sensitive (such as UPF elements interfacing to the access and mobility function (AMF) or session management function (SMF)) can be decomposed into multiple microservices. As an analogy, consider an Intel® processor: the performance-sensitive L1/L2 cache is on the "core" part of the CPU, while last-level cache and direct memory access are considered "un-core."

Combining Kubernetes and OpenStack for Control and Communications in a Mixed Virtualized Network Functions (VNF)/Cloud-Native Network Functions (CNF) Environment

Kubernetes (containers) and OpenStack (VMs) are not mutually exclusive; they can complement each other. As an analogy, think of two groups of engineers, each working under a different manager. While one manager cannot tell the other group of engineers what to do, the engineers themselves can communicate with ease. Similarly, in current hybrid deployments, OpenStack is used for VM management. Kubernetes is used for managing the entire cloud-native deployment, including containers. Communication between the applications and microservices is not affected by which manager is giving the orders about spin-up, spin-down, etc. Because user plane routing is done with external plug-ins rather than default Kubernetes, user plane routing in hybrid cases can be handled by the plug-ins.

For containers (Kubernetes), microservices record their names, services, and IP addresses in a service registry, in a decentralized manner. When one microservice wants to communicate with other microservices, it can detect the presence of those microservices using the registry. When a microservice is no longer needed, it tells the registry to remove the entry. When an operator wants to scale the number of instances, it can use the registry to see what is available. To deal with the possibility that a microservice fails suddenly without a chance to tell the registry to remove it, the registry performs a periodic "health check". If there is no response to the health check, the registry removes that entry.

In contrast, OpenStack uses a centralized registry service (let's call it REGISTRAR). Each microservice does not need to register and unregister in a distributed way because the centralized function REGISTRAR handles this. Consider load balancers: Kubernetes clusters include load balancers, but Kubernetes does not orchestrate this type of resource. Instead, Kubernetes consults a centralized cloud controller for provisioning such resources. In the same way, Kubernetes can interact with the OpenStack controller (REGISTRAR) to find and use microservices on VMs.

Figure 3, developed by the CNCF, suggests one migration path from today's reliance on OpenStack and VNFs to a future state that phases out OpenStack.

One Possible Path to a Cloud-Native Core

In the telco world, 5-9s reliability is a primary concern. While many options are available to increase network reliability and availability, Wind River Titanium Cloud is one example of how carrier-grade reliability can be combined with a complete cloud and virtualization platform solution that features an integrated control and data plane. This platform is based on open source (the OpenStack StarlingX project) and industry standards, with carrier-grade security features.

Titanium Cloud extends OpenStack by adding reliability and availability extensions suitable for a carrier network. According to Wind River, these extensions include the ability to migrate virtual machines (VMs) in hundreds of milliseconds instead of minutes, faster detection of failed VMs, auto-recovery of failed VMs, VM resource management, and faster failover for host and control nodes.

Filling in the Gaps: Kubernetes Networking Innovations

Besides the default single network interface per pod already mentioned, another drawback to Kubernetes is lack of support for special-purpose hardware features, such as Intel® Advanced Vector Extensions 512 (Intel® AVX-512), Intel® FPGAs, and SmartNICs. These features can accelerate certain workloads, but only if they are exposed through the control plane to the underlying applications. And finally, nodes with multiple sockets have non-uniform latency. But by default, cloud-native systems allocate resources with no knowledge of these variations in latency.

In response to the various capabilities that native Kubernetes lacks in the telecom space, Intel is collaborating with the Kubernetes community to advance Kubernetes for networking. You can read about the latest developments by visiting the Intel Network Builders' Container Experience Kits website. Specific links are provided for several projects in the following lists.

- **Bonding common network interface (CNI).** Enables network redundancy of containerized applications through the creation of bonded interfaces.
- **Multus CNI**. Provides high-performance container networking and data plane acceleration for NFV environments.
- **Userspace CNI.** Enables acceleration of east-west traffic (that is, service chaining within a single server) through the creation of multiple network interfaces for pods in Kubernetes to facilitate NFV use cases in container environments.
- **Single-root I/O virtualization (SR-IOV) device plug-in.** Provides north-south traffic acceleration through high-performance network I/O with SR-IOV, orchestrated through Kubernetes.
- **CPU Manager for Kubernetes.** Delivers a performance boost to high-priority applications through core pinning and isolation. You can read the CPU Pinning and Isolation in Kubernetes Technology Guide, and watch a training video.
- **Additional device plug-ins.** Expose special-purpose hardware features such as field programmable gate arrays (FPGAs), GPUs, Intel® QuickAssist Technology (Intel® QAT) cards, and Intel® Optane™ persistent memory, to Kubernetes pods.

Other Kubernetes developments advancing the adoption of CNFs in Kubernetes include the following:

- **Topology Manager.** Addresses performance needs through non-uniform memory access (NUMA) resource allocation (integrated into Kubernetes v1.16). Watch the video here and/or read the Technology Guide.
- **Telemetry-Aware Scheduler** (TAS). Uses telemetry-based policies to schedule workloads. To learn more about this feature watch a demo, view a training video, read a white paper and explore the Technology Guide.

Intel has created a reference architecture, packaging these advancements with the appropriate Intel® architecture (see Figure 4). CoSPs can use this reference architecture to enable an easier transition to a 5G-ready core network. Real use cases are being co-developed by Intel with several ecosystem players, such as Affirmed Networks, Ericsson and Metaswitch.

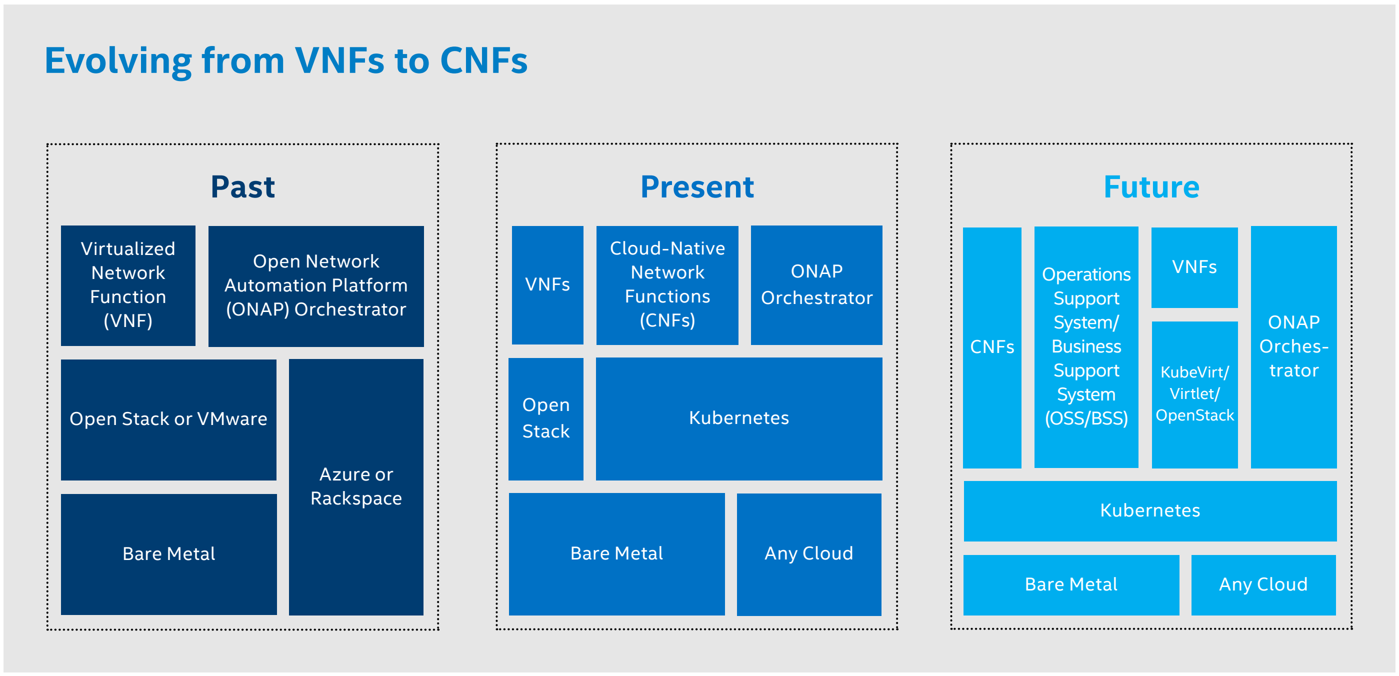
Evolving from VNFs to CNFs
Past
Virtualized Network Function (VNF)
Open Network Automation Platform (ONAP) Orchestrator
Open Stack or VMware
Azure or Rackspace
Bare Metal
Present
VNFs
Cloud-Native Network Functions (CNFs)
ONAP Orchestrator
Open Stack
Kubernetes
Bare Metal
Any Cloud
Future
CNFs
Operations Support System/ Business Support System (OSS/BSS)
VNFs
KubeVirt/ Virtlet/ OpenStack
ONAP Orches-trator
Kubernetes
Bare Metal
Any Cloud

Figure 3. The evolution of management will migrate away from OpenStack toward a truly cloud-native deployment even though virtualized network functions (VNFs) may still be used in certain cases. (source: CNCF Telecom User Group Kickoff presentation).

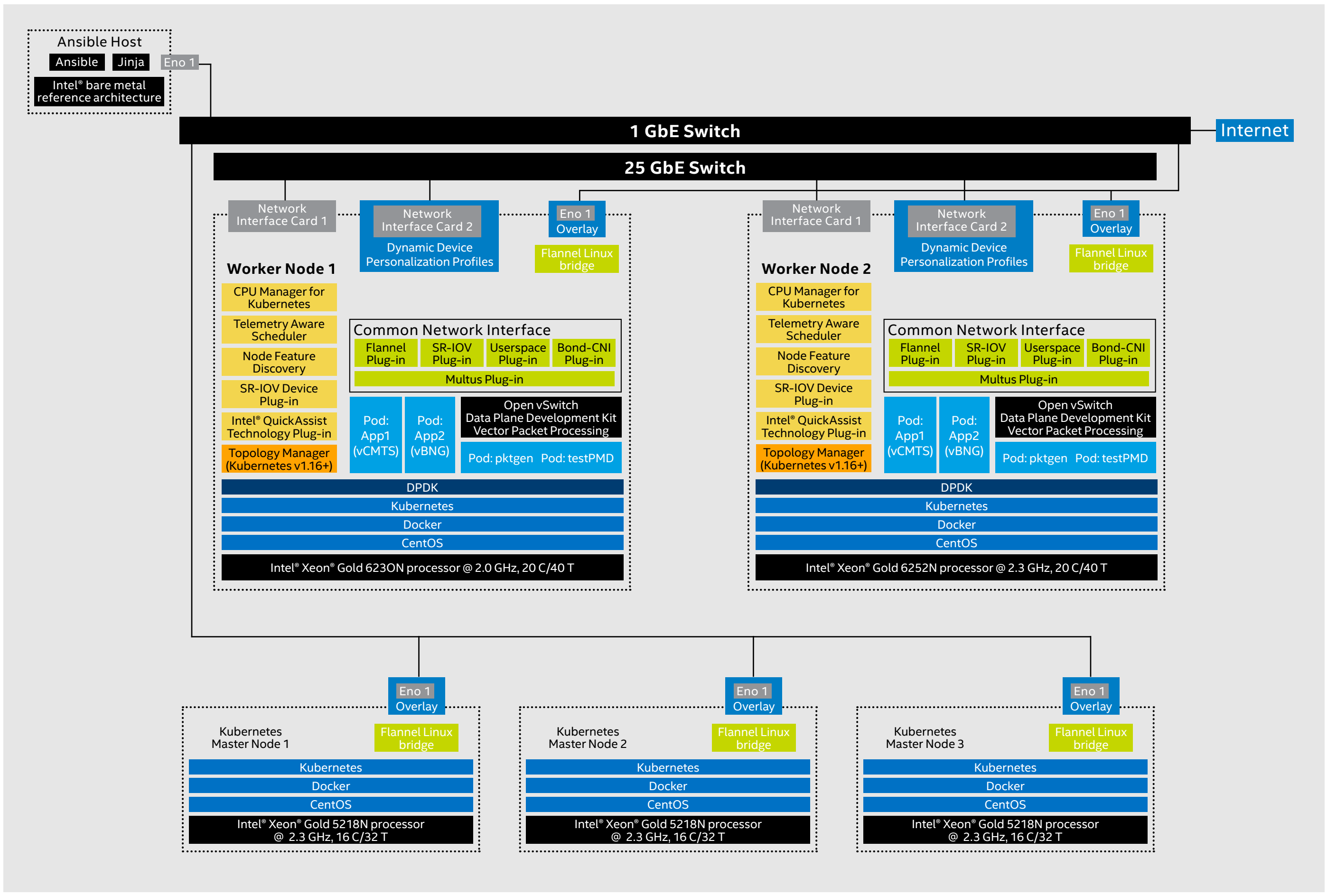

Figure 4. This reference architecture from Intel combines networking-specific Kubernetes innovations and networking-optimized processors to enable cloud-native networks.

Accepting the Culture Shift

Integrating, testing, and deploying network functions across two very different development cultures is possibly more challenging than simply becoming familiar with new technologies. This is probably the biggest motivator for CoSPs to adopt a cloud-native infrastructure, even for the UPF (taking into consideration the aspects already discussed above). By appropriately microservicing the UPF as the "final outpost" of cloud-native 5G core, CoSPs can then gain the cloud-native DevOps benefits, such as continuous integration/continuous delivery (CI/CD). With changes to just a few lines of code, for example, a CoSP could modify how the UPF talks to the SMF, without forklift upgrading the entire UPF. This leads to quicker releases and lower risk due to the use of canary testing.

When the transformation from hardware-based physical appliances to VNFs occurred, CoSPs that didn't make the shift did not survive the market forces. Similarly, operational efficiency requirements and competition from market disrupters like cloud service providers provide a powerful motivation for CoSPs to begin making the culture shift now, so that in two years' time they will be ready to thrive in a 5G world.

Common Acronyms

AMF	Access and Mobility Function
AUSF	Authentication Server Function
CNF	cloud-native network function
CNI	common network interface
dUPF	distributed UPF
PCF	Policy Control Function
RAN	Radio Access Network
RAS	reliability, availability and serviceability
RTT	round-trip time
SBA	service-based architecture
SMF	Session Management Function
SR-IOV	single-root I/O virtualization
UE	User equipment
UDM	User Data Management
UPF	User Plane Function
VM	virtual machine
VNF	virtualized network function

Learn More

You may find the following resources helpful as you explore the world of cloud-native 5G networks:

- Intel's Cloud-Native Transition Reference Material, available under NDA from your Intel representative
- **Intel® Network Builders Container Experience Kits**
- **Intel Network Builders Network Transformation Experience Kits**
- **Closed Loop Automation—Telemetry-Aware Scheduler for Service Healing and Platform Resilience Demo (this is one of many videos on the experience kit site)**
- **Intel Network Builders Program**
- **An Introduction to Cloud-Native 5G Concepts**
- ETSI GR NFV-IFA 029 V0.8.0, **Report on the Enhancements of the NFV Architecture Towards "Cloud-Native" and "PaaS"** (work in progress)
- **ETSI GS NVF-EVE 011 V3.1.1. Specification of the Classification of Cloud-Native VNF Implementations** (work in progress)
- **3GPP TR 23.799, Study on Architecture for Next-Generation System**, V14, Dec 2016
- **Cloud Native Computing Foundation Tools Overview**
- **Network Operator Perspectives on NFV Priorities for 5G**

[1] https://www.windriver.com/products/product-overviews/Titanium-Cloud-Telco-Product-Overview/