White Paper

Cloud-Native Network Functions (CNFs)

Contents

Introduction

Cloud-native principles and technology have proven to be an effective acceleration technology in building and continuously operating the largest clouds in the world. This new technology has been selected by Cisco and others in the industry to develop next-generation Virtual Network Functions (VNFs) called Cloud-Native Network Functions (CNFs). These CNFs, when running inside telecommunications premises, form a private cloud, and the same public cloud principles can be effectively used. CNFs cover all branches of the service provider market, including cable, mobile, video, security, and network infrastructure.

Virtualization and VNFs helped us in getting started in moving toward cloud-native applications. Virtualization, when done properly, offered software models with increased flexibility with hardware dependencies eliminated. However, there are limitations in that VNFs upgrades are slow, restarts take a long time, CLI is still the main interface, software was typically a lift and shift operation, hypervisors such as OpenStack were hard to install, there was little elasticity, and scaling was problematic.

Cloud-native applications address these limitations. Cloud-native applications in general have these characteristics:

- Developed using microservices architecture (that is, 12-factor apps)
- Managed by Kubernetes-style orchestration
- Built-in microservice discovery mechanisms
- Supports dynamic elasticity and scale
- Resilient services
- Improved feature velocity
- Smaller footprint with fast restart
- Continuous deployment and automation principles
- Consistent lifecycle management across containers
- Modern health and status telemetry

Cloud Native customer benefits are depicted in the figure below.

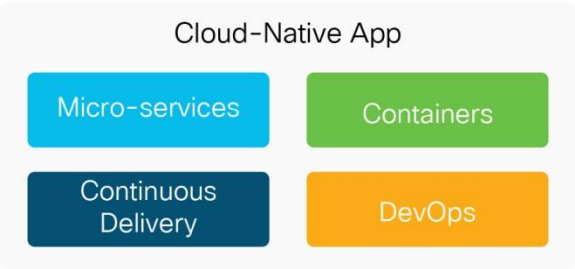
- **Web scale**
  - **Speed:** Lego approach to app creation shortens time to market and lets customer control pace of innovation.
  - **Flexibility:** Strong infrastructure offers allow one to focus on app layers, where they create value/differentiation versus recreating common infrastructure.
  - **Efficiency:** Reusable services across teams and business units translate into lower OpEx cost for developer and customer.
  - **Improved business outcomes:** Speed, scale, and agility lead to increased revenue.
- **Deployment**
  - Standard DevOps principles and tooling are used to allow for increased feature velocity as well as consistent deployments.
- **Security**
  - Cloud-native tooling for security scans and cloud penetration tests provide increased confidence in the security of solutions.
  - Smaller CNFs can independently control subscribers (limit the blast zone) versus the monolithic box approach.
- **Monitoring**
  - Standard tools such as Kubernetes, Prometheus, and Elastic Search provide common health and status of containers.

The cloud-native approach has been proven at web-scale companies and has shown increased speed, flexibility, efficiency, and business outcomes as depicted. Teams can now focus more on the business value in the application versus the infrastructure.

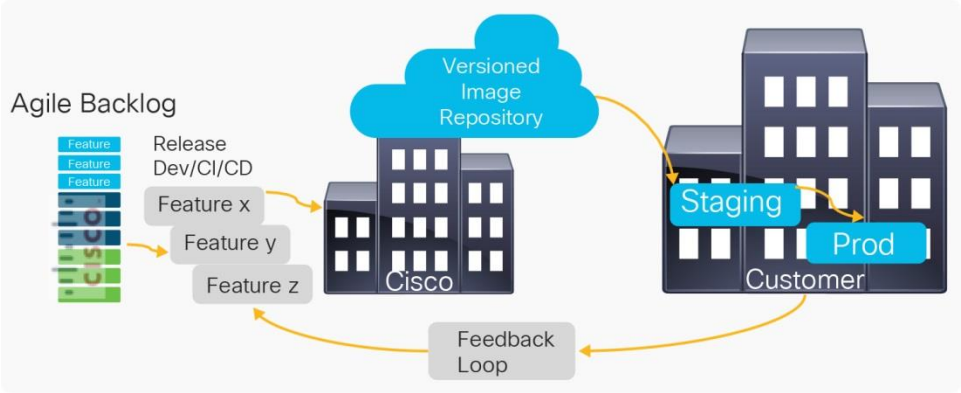
Primary cloud-native constructs

Being cloud native is an approach to building and running applications that fully exploit the advantages of the cloud model. A cloud-native application utilizes a collection of tools that manage and simplify the orchestration of the services that make up the application. These services, each with its own lifecycle, are connected by APIs and are deployed as containers. These containers are orchestrated by a container scheduler, which manages where and when a container should be provisioned into an application and is responsible for lifecycle management. Cloud-native applications are designed to be portable to different deployment environments: for example, in a public, private, or hybrid cloud. Continuous delivery and DevOps are methods used to automate the process of building, validating, and deploying services into a production network.

Cloud Native constructs are depicted in the figure below.



- **Microservices:** An architectural style that structures an application as a collection of loosely coupled services to implement business capabilities. Microservices are commonly deployed in containers and enable the continuous delivery and deployment of large, complex applications. Each microservice can be deployed, upgraded, scaled, and restarted independently of other services in the application as part of an automated system, enabling frequent updates to live applications without affecting end customers.
- **Containers:** Containers are another form of virtualization, using Operating System (OS)–level virtualization. A single OS instance is dynamically divided among one or more isolated containers, each with a unique writable file system and resource quota. Containers can be deployed on both bare metal and virtual machines. Containers deployed on bare metal offer performance benefits to virtual machines by eliminating the hypervisor overhead. Although each microservice is commonly deployed in a separate container, multiple microservices may be deployed per container to address application and performance requirements, for example, when colocation of services logically simplifies the design or when services fork multiple processes in a container.
- **Continuous delivery:** Makes an individual application change ready for release as soon as it is ready, without waiting for bundling with other changes into a release or an event such as a maintenance window. Continuous delivery makes releases easy and reliable, so organizations can deliver frequently, at less risk, and with immediate feedback from end users. The way service providers consume software this frequently will revolutionize speed to market. Eventually, deployment becomes an integral part of the business process and enterprise competitiveness, taking advantage of canary and A/B testing in the real world rather than artificial labs.
- **DevOps:** DevOps is the utilization of lean and agile techniques to combine development and operations into a single IT value stream. DevOps enables organizations to build, test, and release software more rapidly and iteratively by applying continuous integration and delivery. For example, DevOps enables the automation of deploying and validating a new software feature in an isolated production environment, which can then be rolled out more broadly into production after it has been proven. To fully realize DevOps, service providers must adopt cloud-native techniques, establish automated continuous integration, and deliver pipelines with its vendors. (See Figure 3.)



Service providers are looking to reduce OpEx by automating and simplifying their network operations, allowing for faster services time to market, and deploying across a broad range of cloud environments. Cloud-native technologies provide the fundamental building blocks to build applications that achieve these objectives.

Benefits of being cloud native

A cloud-native architecture has many benefits. The following sections capture the primary benefits and best practices for applying cloud-native principles to CNFs to further define Cisco's progress and strategy.

Distributed microservices

Expanding on previous discussion, microservices are componentized, reusable software modules enabling a number of benefits for the customer and the development organization. Microservices expose smaller discrete functions to an application through APIs. APIs are maintained and versioned and promote reuse of microservices in other applications. For example, microservices for control plane are used across a number of different applications. Microservice APIs are typically exposed over a RESTful interface or via a message bus, which allows each service to choose the best technology available for client operations. For example, Java can be used for a control-plane service, and Go can be used for data-plane services. This componentization allows open-source technologies to be more easily integrated into the application or swapped for different technologies as the application evolves.

Lightweight footprint

Containers are a way of virtualizing an application process or set of processes and are inherently lightweight because, unlike a virtual machine, the OS is shared across containers. Significant performance improvements can be realized when starting and upgrading containers during lifecycle operations. Containers may be deployed on bare metal with a basic Linux OS or can be deployed on virtual machines residing on top of a hypervisor. Although some of the benefits of containers are limited when running on virtual machines, a majority of the instances do not require the virtual machine to be upgraded for lifecycle events. For example, upgrading the software containers in a lifecycle event do not require the virtual machines to be upgraded.

Service discovery

Service discovery is one of the primary components of the cloud-native stack and is used to provide a real-time service registry for all available services. The service registry enables new services to be dynamically orchestrated into an application. Services are automatically scaled and recovered via Kubernetes if a service becomes unavailable and the service must be restored.

Lifecycle management

One of the primary benefits of moving to containerized microservices is the ability to orchestrate the containers so that separate lifecycle management processes can be applied to each service. This allows for each service to be versioned and upgraded singularly as opposed to upgrading the entire application or virtual machine image. When upgrading an application, the container scheduler determines which individual services have changed and deploys only those specific services into the broader application. When the application is implemented with the appropriate level of state separation, this process allows for fully automated in-service upgrades and rollback of the containers that make up the application.

State separation

One of the most commonly agreed-on design patterns in cloud-native applications is a clean separation of stateful (also known as backing services) and stateless services. Application services containing functional logic (database, file system, or cache) should be separated from stateful services. For example, a service handling a create session request implements the logic for creating the session but stores the session information in a separate stateful service, which physically stores the session to memory or disk. This allows for the stateless application services to be autonomous, lightweight, upgradable, recoverable, and rapidly scalable.

Stateful services are more challenging because of where and how the state is actually stored: for example, on the file system, in memory, or in a cloud storage file system. Stateful services must address the availability, consistency, and portability of state, which typically requires replication across one or more containers while making sure that consistency of the state is maintained.

Availability and resiliency

Cloud-native applications inherently provide support for high availability and resiliency through service discovery and load-balancing transactions across stateless application containers. In addition, because containers are lightweight, recovery times are far less than when recovering a virtual machine, physical box, or application as a whole. This allows for faster and more granular ways of responding to failure events.

High availability cannot be solved by container orchestration alone, and, in most cases, the application itself has resiliency requirements. Stateful services, such as a resilient database, require resiliency beyond the inherent features of a cloud-native architecture, which requires state synchronization and data integrity. Additionally, protocol services require specific failover and availability mechanisms defined at the protocol level.

Operational benefits

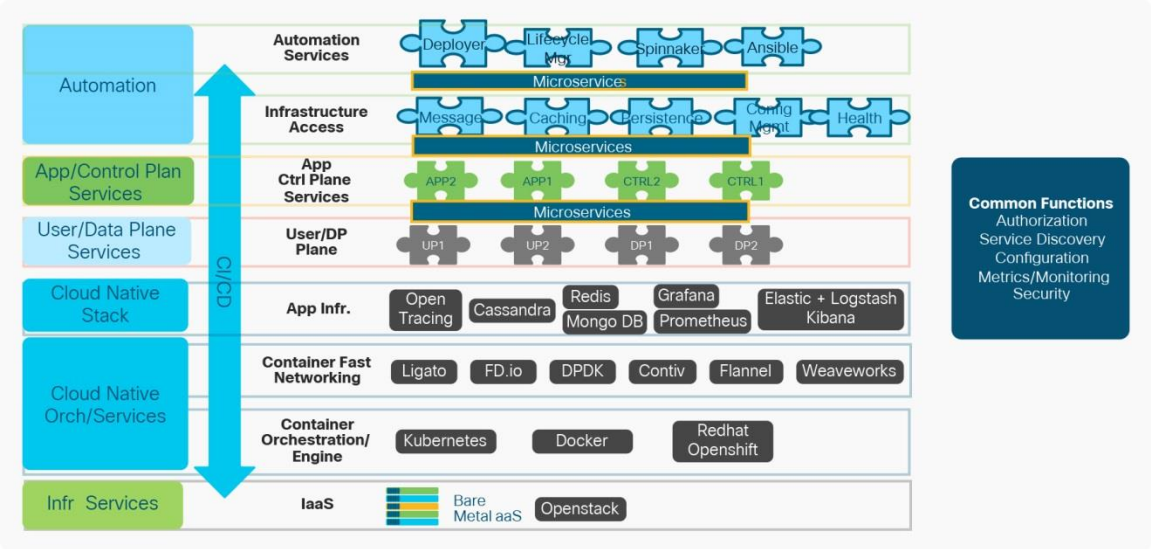
Fundamentally, containerized applications running on bare metal perform better than those running on virtual machines because there is not the overhead of a hypervisor. Because of the lightweight footprint of containers, the speed of instantiating or recovering services is optimized. Because virtual machine instantiation includes an underlying OS and disk resources, the provisioning process can take minutes, whereas a container instantiation can take seconds. When containers are deployed on top of virtual machines—for example, in a CNF architecture—and the hypervisor overhead is still present, there are still a number of operational benefits because containers have a separate lifecycle from virtual machines. For example, a software upgrade or recovery might not require the instantiation of a new virtual machine. Therefore, because containers are lightweight, the benefits of starting, recovering, and upgrading services are substantially faster.

Scalability

A containerized architecture enables the ability to scale each microservice independently. Each container is monitored based on KPI metrics, enabling the orchestration scheduler to scale/descale individual containers. As new containers are started up for scaling, they register themselves in the service discovery layer and are automatically orchestrated into the broader application. Load balancing is used to transparently add new container instances without affecting the containers that depend on that container.

Components technology stack

Figure 4 is a model that Cisco® CNF applications typically follow. Each major area is summarized further in the rest of this section, starting from the top and working down.



Automation and common functions

Customers expect that Cisco CNFs will behave and operate in a consistent manner across applications, causing the need for:

- Consistent open framework for cloud-native container management
- Ability to easily deploy and update cloud-native containers from Cisco
- Visibility into container deployments throughout the entire customer's VNF lifecycle
- Inherent security in deploying Cisco CNF containers in the cloud and on the customer premises
- Standard methodology for CNF deployment across Cisco
- One-click deployment options to make it easy for customers
- Ability to do everything via API

To meet these expectations, Cisco utilizes a common Helm chart format for its container structure. This structure contains all the deployment information, as well as Kubernetes deployment instructions. By having each container follow a consistent chart structure, customers can more easily integrate the containers into their cloud-native CD systems if desired.

For Cisco customers wanting to use Cisco deployment, higher level deployment functionality is provided by a deployment user interface/API. Cisco is using tools such as Spinnaker, which provides a framework to manage a distributed deployment at scale with simple operational tooling. Deployment tools such as Spinnaker and Helm provide benefits, including:

- **Multicloud location:** Applications can reside in multiple locations, including the private data center and in the cloud with multiple cloud providers.
- **Automated releases:** Create deployment pipelines that run integration and systems verifications, spin up and down containers, and monitor rollouts. Pipelines (that is, workflows) can be triggered from events.
- **Ability to build in deployment best practices:** Create and deploy immutable images for faster, easier rollbacks and elimination of hard-to-debug configuration drift issues. Built-in deployment strategies such as red/black and canary.
- **Verification via automation:** Included to make sure of a successful deployment.

Control and user- and data-plane microservices

Cisco provides control- and data-plane microservices. These microservices typically communicate via Kafka messages for control- to data-plane interaction and through specialized interfaces for high-speed data transfer where latency matters. Microservices from Cisco are used for upstream and downstream data processing as well as for networking areas such as diameter routing. Configuration is provided via products such as Etcd.

Later sections detail more about control and data planes and cable and mobility use cases.

Cloud-native services

The cloud-native services offered are from a standard Cloud-Native Compute Foundation (CNCF) common stack. We typically classify the services into the following areas:

- **Messaging:** Kafka consumers and producers are used.
- **Data store:** Teams primarily use Mongo or Cassandra as a backing store.
- **Security:** Vault is used for certificate stores with M-TLS for encryption.
- **Logging:** Typical ELK stack is used with fluentd.
- **Configuration:** etcd.
- **Health and status:** Prometheus in conjunction with Grafana is used.
- **Service mesh:** Istio.
- **Orchestration:** Kubernetes and Docker containers.

Containersss fast networking

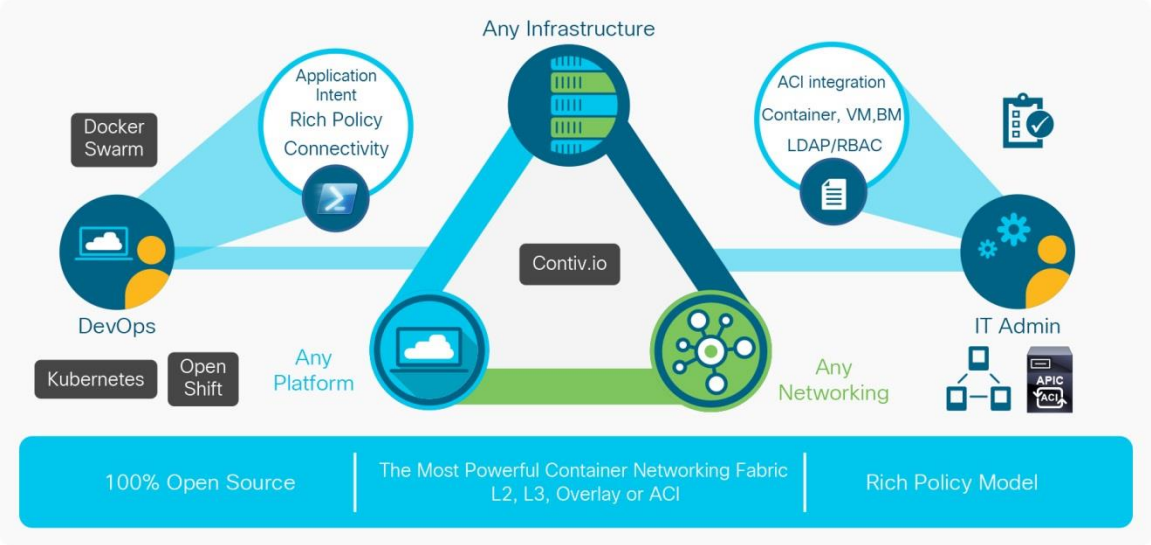
Basic container networking uses CNI implementations such as Weave, Flannel, Cisco Contiv-VPP, and others.

High-speed data-plane interaction between containers and the external network requires multiple networks to implement the CNF functionality. Cisco differentiates in this area with its own high-speed data-plane applications: FD.io and Vector Packet Processing (VPP), for example. Cisco's approach takes advantage of the low overhead of containers to deliver higher performance using cloud-native technologies to build the network functions so they run in the same network and user space as the applications. Network functions become part of the service topology. Network functions are truly just another service and can be developed and deployed using the same tools as the applications with the same velocity. Cisco software uses user space versus the kernel providing benefits such as ease and speed of upgrade, less system call overhead, and decreased dependency on Linux networking. Technology usage is FD.io (VPP data plane), DPDK (network), and SPDK (storage).

The VPP platform is an extensible framework that provides out-of-the-box production quality switch/router functionality, which can run on commodity CPUs. The primary benefits of VPP are its high performance, proven technology, modularity and flexibility, and rich feature set. The framework allows anyone to plug in new graph nodes without the need to change core or kernel code. VPP supports a cloud-native architecture with its ability to be orchestrated as a part of a Docker containerized solution. VPP is proven in many networks today and is the basis for multiple Cisco virtualized network functions.

Additionally, the Cisco data-plane containers use a Go language agent to access VPP. This Go language library was open-sourced by Cisco via the Ligato program. Ligato (github.com/ligato) provides a mechanism for delivering and managing agents for cloud-native network functions to enable them to become part of the application service topology, all in user space.

For integration of CNI interaction and high-speed data-plane interaction, Cisco has developed Contiv integrated with VPP to abstract container connectivity and networking. Contiv is a user space–based, high-performance, high-density networking plug-in for Kubernetes, which then uses FD.io/VPP as the industry's highest performance data plane. Functionalities included are the allocation of IP addresses to networking-related pods (IPAM) and programming of the underlying infrastructure it uses (Linux TCP/IP stack, OVS, VPP, and so on) to connect pods to other pods in the cluster and/or to the external world. It also implements K8s network policies that define which pods can talk to each other. (See Figure 5.)



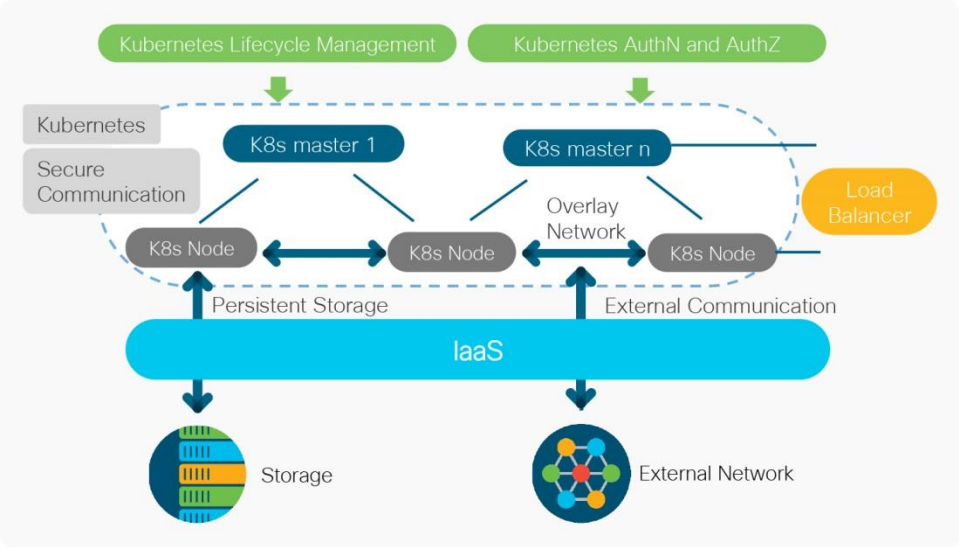
Infrastructure

Cisco cloud-native applications operate on bare metal as well as well as in VM-based environments. Cisco offers the Cisco UCS® bare metal infrastructure as well as cloud management infrastructure provided by the Cisco Container Platform. Cisco has chosen Kubernetes as its common container orchestration platform. Cisco is providing a managed Kubernetes service via the Cisco Container Platform to make sure of secure and reliable platform for CNFs.

The Cisco Container Platform is a fully curated, lightweight container management platform for production-grade environments, powered by Kubernetes, and delivered with Cisco support. It reduces the complexity of configuring, deploying, securing, scaling, and managing containers via automation coupled with Cisco's best practices for security and networking. The Cisco Container Platform (Figure 6) is built with an open architecture using open-source components, so you're not locked in to any single vendor. It works across both on-premises and public cloud environments.

Benefits are:

- The ability to easily manage multiple clusters
- Simple installation and maintenance
- Networking and security consistency
- Transparent application deployment, both on the premises and in public clouds
- Persistent storage



The Cisco Container Platform will offer support for Kubernetes on bare metal as a service. It will perform the bare metal hardware bringup, including the installation of the operating system, configuration of the system, and subsequent deployment of Kubernetes. Additional Kubernetes clusters can be brought up on demand. To support the hybrid world, OpenStack and VMware will be supported.

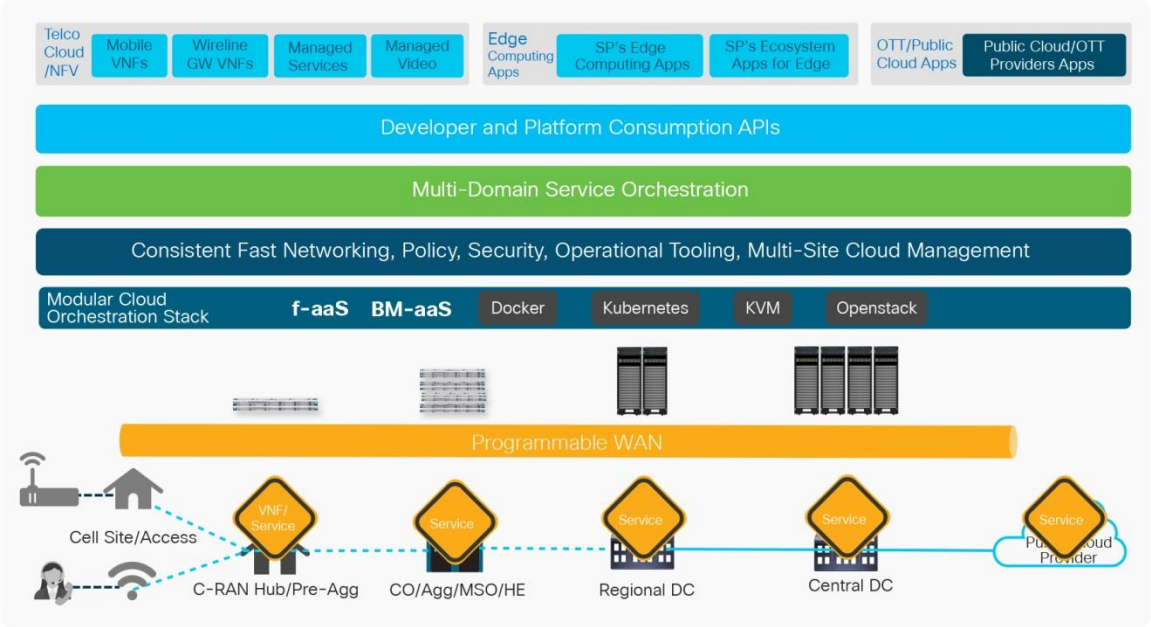
The Cisco Container Platform is offering various modes to support cloud native directly as well as the transition from VMs to containers (hybrid mode) where VMs and containers coexist.

The Cisco Container Platform will support container networking by natively integrating the Contiv-VPP/Ligato work previously mentioned into the Kubernetes deployment. In this manner, typical CNI-based container networking can be supported as well as high-speed networking use cases, all controlled by common policies.

Distributed telecommunications cloud requirements

Container-based CNF systems will need to support a distributed telecommunications cloud from the data center to the edge. The containers will be mapped to the external network via a Contiv-VPP/Ligato framework, which will support external connectivity requirements to areas such as the DC Interconnect (DCI), virtual routers, and virtual switches. Cisco is actively working the architecture and technical aspects of fast networking for containers and how we can make the networking architecture a part of the broader network fabric with service provider WAN. In addition, interworking with SR-IOV is under discussion as well as container-to-VM-based interworking for CNF/VNF chaining use cases. This architecture is considering failure domain considerations in the multisite deployment models as well as connectivity for MPLS/SR to the TOR, MPLS/SR to the host vSwitch, and MPLS/SR to the CNF.

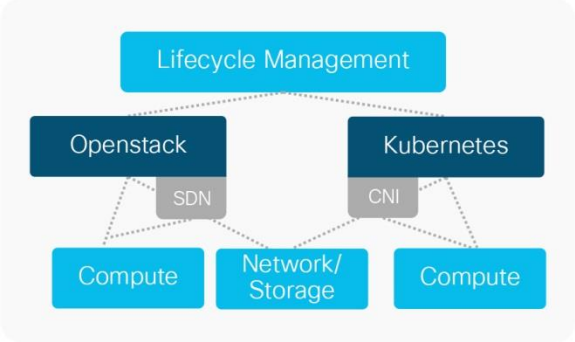
The different areas of the network are important for placement. Figure 7 shows the support of smaller environments at the edge through the data center with the need for cloud-native techniques across all environments.



Hybrid world implications and MANO integration

Cloud-first companies will likely use public cloud for new applications when they can but will use private cloud when they must because of latency requirements and packet rate needs as well as for data sovereignty, compliance, and other needs. Companies will be in a hybrid cloud mode for some time as they transition. The customer will have applications spread across public clouds accessed through private connections to the on-premises systems. Cisco, with its deep networking background, is the right vendor to help one optimize traffic patterns, secure interactions, reduce costs, and provide flexibility.

In a similar analogy, the hybrid world is driving the need for container-based systems to coexist with VM/bare metal workloads while using common orchestration. This includes hybrid VM/container systems, which drive VNF/CNF interworking. Because a majority of service providers are currently deploying NFV cloud solutions based on VNFs running on virtual machines, Cisco's cloud-native strategy can be supported on top of virtual machines in an NFV MANO architecture as previously described and still brings significant advantages because it is using cloud-native automation techniques. When cloud-native techniques such as dynamic discovery and container scheduling are integrated into the CNF, this integration dependency is simplified. (See Figure 8.)



These options will require cloud-native network functions to support a number of deployment pipeline integrations such as NFV management and organization (MANO), Open Network Automation Platform (ONAP), central office rearchitected as a data center (CORD), and public and private clouds running on bare metal and virtual machines.

Cloud-native applications support an enhanced level of portability to multiple deployment pipelines with continuous integration and deployment capabilities required to meet these requirements.

Cisco CNF implementation examples

Cloud-native broadband router CNF

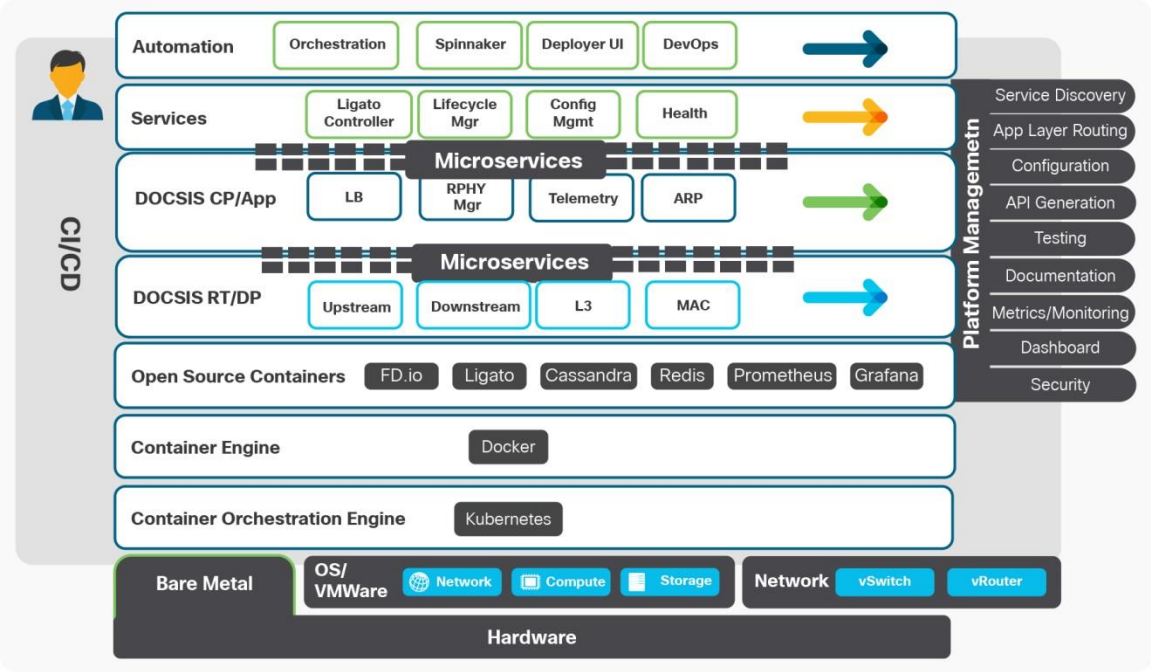
The Cisco cable team is applying cloud-native virtualization to CMTS DOCSIS using cloud-native techniques. This transformative work enables more effective management and deployment of cable networks. The effort is called the cloud-native broadband router.

In contrast to hardware-based systems, the new system will separate control and data plan functionality into microservices that can be separated and eventually run in hybrid clouds. Cable operators and hardware vendors have recognized the potential of cloud-native virtualization and have produced early stage products that are cloud-native virtualized CMTS/CCAP systems. The concept is simple. The CMTS/CCAP processing component that performs the heavy lifting of routing traffic and managing modems is moved into a virtualized environment running on bare metal or on a virtual machine. The cloud-native CNF is fundamentally then a load-sharing distributed system. If any component fails, its load can be moved to a different place via Kubernetes orchestration. Furthermore, scaling up or down can use the same load sharing and distribution system, and in that respect a failure is simply a case of a “forced scale-down.”

The cloud-native environment for cable includes modern deployment techniques such as one button to click, canary testing, and red/black testing, allowing for rolling updates to production software. It additionally has incorporated cloud-native techniques for health and status as well as logging. This breaking of CMTS into a cloud-native stack increases the reliability, scalability, and feature velocity that have been seen in the web and business app space to:

- Revolutionize cable-based SW development and operations
- Enable rapid service development and deployment, with in-production testing (CI/CD)
- Enable resilient and elastic resource scaling and continuous service upgrades
- DevOps and automation movement (large scale with low operational overhead)
- Apply relevant app-level concepts to real-time DOCSIS system software
- Expose modern web-native interfaces, where needed, to simplify overall system and take advantage of modern technologies such as streaming telemetry
- Enable faster time to true DevOps to facilitate operational transformation for cable operators

The architecture abstraction follows. It uses the deep Cisco expertise across important technology areas and DOCSIS leadership as well as expertise in full cloud-native stacks. It includes VPP for software data-plane container networking and orchestration and microservices framework and lifecycle management. All these are made easier to use via the Ligato (ligato.io) infrastructure. (See Figure 9.)



Mobile CNF

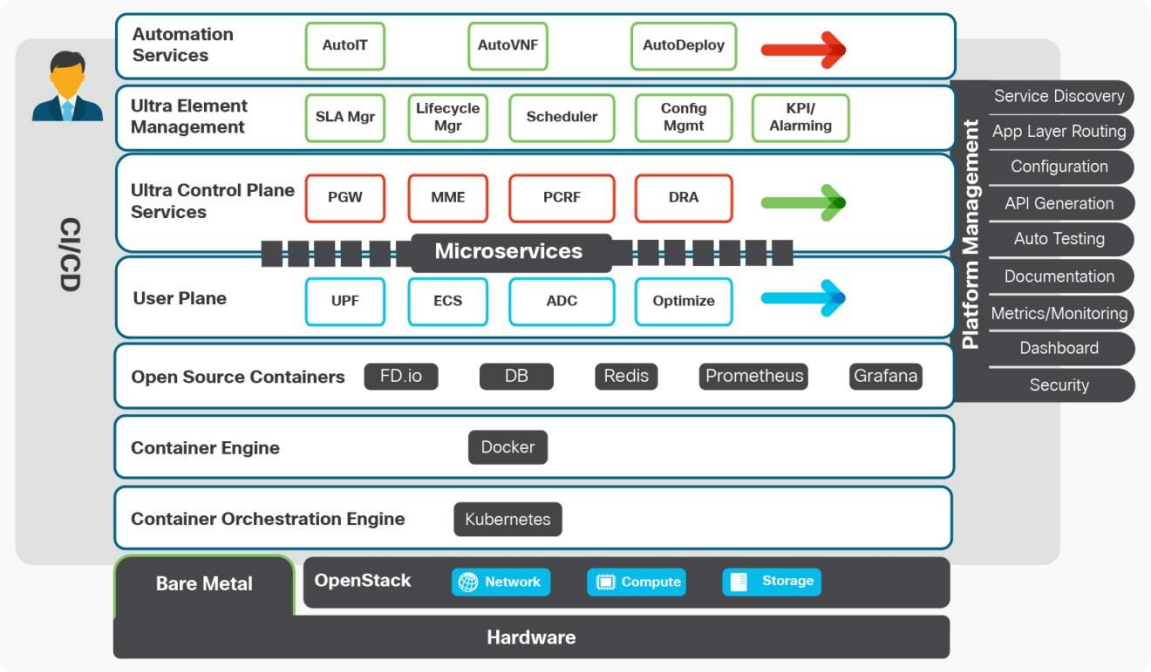
Service providers are looking to disaggregate the network by moving network functions closer to the edge: Mobile Edge Computing (MEC). One example of MEC is in a Control User-Plane Separation (CUPS) architecture, where the user-plane functions of the mobile core are moved to the edge of the network, while the control plane is deployed centrally. As a result of these use cases, service providers require the flexibility to deploy network functions in a number of deployment environments. These environments include public, private, and hybrid clouds. Cloud-native technologies offer a richer set of infrastructure and tools to achieve this level of automation.

The Cisco Ultra Services Platform (USP) provides a common NFV-compliant platform for Cisco mobile core functions, including user-plane and gateway control-plane functions, along with policy, charging, and subscriber data management functions. The Cisco USP is evolving to support cloud-native network functions. As part of this evolution, Cisco VNFs are being decomposed into multiple microservices and deployed as containers, each of which can then be independently scaled, upgraded, and deployed according to the mobile operator's business requirements. USP provides a common cloud-native platform that enables Cisco to deliver its suite of mobile CNFs as individual applications or as an end-to-end mobile core across a wide range of cloud environments, making sure of the automation and simplicity required to reduce service provider OpEx and deliver mobility use cases.

The result of this effort is a suite of microservices deployed as Docker containers and integrated with a common cloud-native management stack, which can be orchestrated as a single CNF or as multiple CNFs running as an integrated mobile core solution.

Similarly, Cisco's evolution to being cloud native is also being realized today with the release of Cisco Policy and Charging Rules Function (PCRF) and Diameter Routing Agent (DRA) applications deployed on a cloud-native architecture utilizing Docker containers with integrated container orchestration and scheduling; service discovery; and lifecycle management such as autoscale, upgrade/rollback, and high availability. Mobility data planes are developed using VPP technology.

The figure below abstracts the general mobile CNF architecture.



Summary

Cloud-native principles and technology will help service providers to achieve web scale. Virtualization and VNFs helped us in getting started in moving toward cloud-native applications, and being cloud native continues this journey. Service providers must fully automate the deployment and operations of the network. There are specific considerations to realizing being cloud native in the network that are not inherent to web-based cloud-native solutions such as user-plane and protocol considerations. Cisco has the underlying technology and experience to make sure of a smooth transition from VNF to CNF. Cisco's strategy for being cloud native introduces a new kind of technology tenants, including microservices, containers, orchestration, continuous integration and deployment, and DevOps. These tenants and their underlying design constructs and benefits enable the CNF to be fully automated and operated to maximize automation and orchestration to achieve new revenue opportunities and use cases. Cisco's approach to realizing cloud-native application transformation is well under way.

More information

For more information, contact your Cisco sales or product representatives.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA C11-740841-00 06/18