Evolving the Mobile Core to Being Cloud Native

Introduction

The most effective business driver for Network Functions Virtualization (NFV), and its evolution to 5G, is for service providers to foster new opportunities to scale their businesses while substantially reducing the operational expenses to deploy and manage their network functions. Service providers are beginning to realize the powerful combination of increased revenue combined with OpEx reductions through the virtualization of network functions to a common NFV cloud architecture. However, in many cases, Virtual Network Functions (VNFs) have essentially been ported from a physical appliance to virtual machines without fully addressing the underlying software architecture, thereby not delivering complete separation from infrastructure, the optimal level of operational automation, or the OpEx reductions promised by NFV. Rapid technological changes create challenges for service providers that cannot be quickly resolved in a solution that relies on manual intervention. Too often the dependency of the underlying infrastructure has impeded the efficient deployment of a VNF, adding to system integration time and costs. As a result, service providers are looking look to cloud-native technologies to fully facilitate new revenue opportunities and realize OpEx savings and network automation.

To fully realize the flexibility that virtualization promises, the underlying application software of network functions must be architected to support any infrastructure and fully automate deployments and lifecycle events such as service creation, transparent software upgrades, dynamic scalability and simple recovery. An API-driven NFV model gives service providers the ability to combine applications from different sources, support new functionality, and install patches quickly, greatly expanding the amount of innovation providers and the telecom community at large can achieve.

Contents

For example, 5G introduces a number of new revenue opportunities that rely on automated orchestration of VNFs running on network slices in the core and on Mobile Edge Computing (MEC). The success of network slicing and MEC use cases is tied to the promise of network automation. Network slicing enables the service provider to slice the network into vertical and/or operational domains; for example, network slicing enables different slices of the network to be allocated for an Internet of Things (IoT) or enterprise customer. In this example, an enterprise slice might be used for deploying network services to employees, while an IoT slice is optimized for a specific set of IoT devices to communicate with the network. To reasonably achieve potentially thousands of network slices, slice provisioning, lifecycle operations and configuration management must be automated to the level of "pushbutton" operations.

For MEC, service providers are looking to disaggregate the network by moving network functions closer to the edge. 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. The MEC use case enables user planes to be distributed to the network edge to achieve lower latencies as well as the potential for locating them in an enterprise data center. The MEC and CUPS use cases require fully automated operations across operational domains to account for the large number of edge computing sites where these workloads must be managed. Cloud-native technologies offer a richer set of infrastructure and tools to achieve this level of automation.

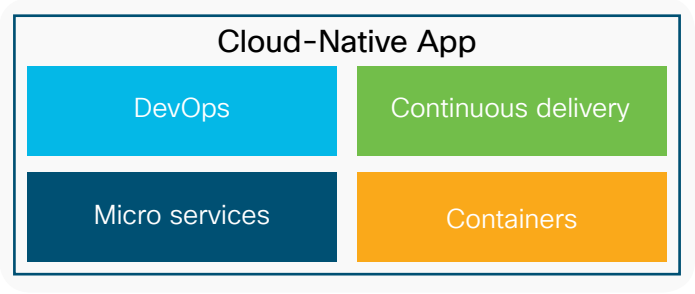
As a result of these use cases, service providers require the flexibility to deploy network functions in a number of deployment environments. These environments include public, private and hybrid clouds. For example, service providers might choose to deploy traditional consumer services in their own private cloud but services such as IoT, Mobile Virtual Network Operator (MVNO), and enterprise services in a public cloud. The public cloud option becomes especially relevant when the service provider is looking to address global deployments, which rely on locations outside of their network footprint. In the MEC and CUPS examples, the user plane could be located in a private cloud, whereas the control plane resides in a public cloud. These options require network functions to support a number of deployment pipelines such as NFV Management and Organization (MANO), Open Network Automation Platform (ONAP), Central Office Re-architected as a Datacenter (CORD), 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.

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, 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 VNFs 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 5G use cases.

Primary cloud-native constructs

Being cloud native is an approach to building and running applications that fully exploit the advantages of the cloud computing 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 the 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. (See Figure 1.)

Figure 1. Cloud native applications



The primary tenets of cloud-native applications are:

- **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/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 within 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 slicing 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. In order to fully realize DevOps, service providers must adopt cloud-native techniques, establish automated continuous integration, and deliver pipelines with its vendors.

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.

Cloud-native considerations for the mobile core

A cloud-native mobile core solution has a unique set of considerations that must be addressed to achieve the full potential of cloud-native technologies. Cloud-native technologies have grown and matured out of technologies and methodologies based on web applications and services. The following include some examples of unique considerations for the cloud-native mobile core.

User plane packet processing

Cloud-native technologies have historically been applied to cloud and data center workloads as opposed to packet processing for complex network functions such as an EPC. Whereas control plane functions in the mobile core allow for applying weblike techniques such as overlay networking, user plane packet processing requires an optimized architecture. Packet processing microservices must be scheduled in the user space and colocated with other packet processing containers on the same host: for example, the equivalent of a pod in Kubernetes. Because of the immense east-west traffic load and low latency requirements on packet processing, standard networking between containers must be optimized to not overload the kernel and virtual I/O. Cisco optimizes the communication between user plane containers on a host using shared memory to bypass the effects of the kernel or virtual networking layer. These optimizations, among others, are required to effectively achieve the benefits of cloud-native user plane functions in the mobile core.

3GPP Protocol Considerations

Cloud-native best practices define using simplified RESTful and/or message bus APIs for the internal communication between containers. Cloud-native technologies have native support for automating the discovery and orchestration of these services into the application. These design patterns enable for simplified lifecycle management of services in a cloud-native architecture.

3GPP protocols such as Diameter and GTP are not natively supported in cloud-native technologies. In a cloud-native mobile core, these protocols must be integrated with the same level of dynamic discovery and orchestration as RESTful and message bus protocols. Cisco solves this problem by componentizing 3GPP protocols as separate microservices; therefore, the core application functionality can be separated from the underlying protocol. This enables the ability to support compatibility between 4G/5G protocols while maintaining a consistent application layer. Additionally, this approach enables the comingling of 3GPP and cloud-native protocols such that APIs can be extended beyond what 3GPP defines to create an enhanced level of flexibility and agility. This approach also supports the ability for a service-based architecture where functions expose service-oriented APIs to the mobile core, which is a critical evolution of 5G.

Mobile Core from 4G to 5G

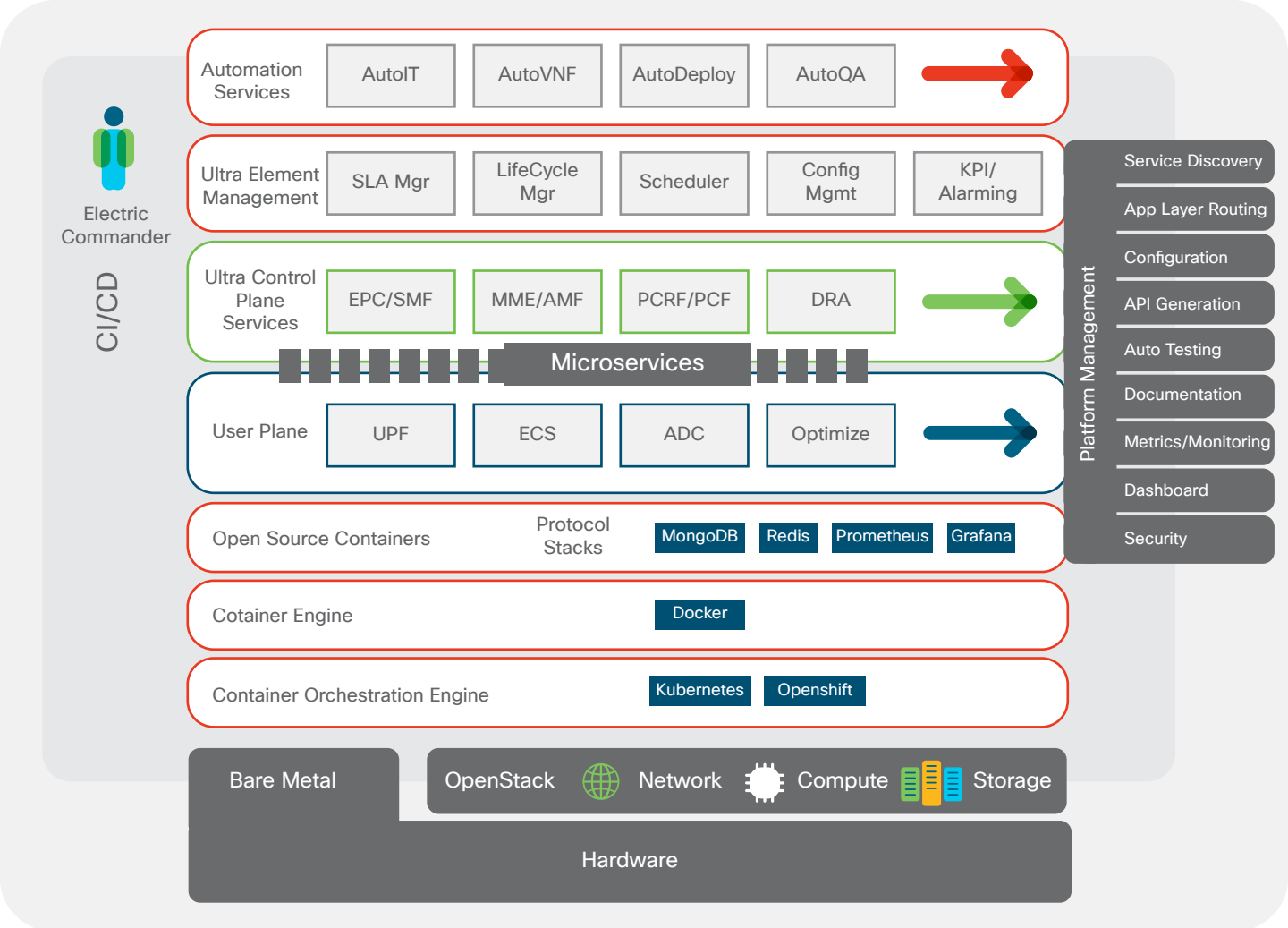
In addition to supporting new 5G specifications, a cloud-native mobile core must support all of the capabilities of 4G to enable the backward compatibility in a common solution. The mobile core must support the functionality, control plane signaling and error handling, and interoperability that have been defined over many years with 4G and carry forward to the new capabilities of 5G. The Cisco mobile core solution brings these proven capabilities of 4G into its cloud-native mobile core and provides a strong foundation for the evolution to 5G.

Ultra services platform evolution to being cloud native

Cisco Ultra Services Platform (USP) is the product umbrella that combines Cisco virtual assets, including packet core, policy and service chaining technologies into a single integrated VNF and the underlying platform that automates and simplifies the onboarding, instantiation and operation of this closely integrated collection of technologies. The critical objective of USP is to provide a fully automated, NFV-compliant solution to deploy and manage VNFs. By integrating more capability within a common mobile core VNF, Cisco can abstract the complexity of integrating these components and remove this responsibility from service providers, relieving them of lengthy and costly integration efforts.

In the next evolution of USP, Cisco is consolidating all mobile core functions into a common cloud-native platform. This consolidation effort includes gateway and user plane applications as well as policy, charging and unified data management. 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 VNF or as multiple VNFs running as an integrated mobile core solution.

Figure 2. Cisco cloud native mobile core architecture



Cisco's evolution to being cloud native is being realized today with the release of Cisco PCRF and 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.

As part of the current Cisco ultra CUPS architecture, the User Plane Function (UPF) now resides in its own containerized microservice and has been separated from the Session Management Function (SMF). In this architecture, the SMF owns the master state repository, and the UPF is able to recover session state from the SMF during a recovery or upgrade event. CUPS state separation follows the cloud-native principle of application state separation and allows for forwarding optimization as well as enhanced availability and upgradability.

Cisco is including Vector Packet Processing (VPP) technology in its user plane function as part of its transition to being cloud native. 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/kernel code. VPP supports a cloud-native architecture with its ability to be orchestrated as a part of a Docker containerized solution. VPP is a primary component of Fast Data input/output (FD.io) and was donated by Cisco into this open-source community. With FD.io, we are able to take advantage of enhancements from the open-source community to bring new features sooner. VPP is proven in many networks today and is the basis for multiple Cisco virtualized network functions aligning the mobile core to this common strategy.

Cisco's cloud-native mobile strategy is realized through USP as a fully automated end-to-end mobile core solution deployable on public/provide clouds, on bare metal or virtual machines, and aligning with broader NFV initiatives such as the Open Network Automation Platform (ONAP) and the Telecom Infra Project (TIP), the Open Platform for NFV (OPNFV), ETSI NFV SIG and OpenMANO.

Being cloud native and nFV

A majority of service providers are currently deploying NFV cloud solutions based on VNFs running on virtual machines. Cisco's cloud-native strategy is fully supported on top of virtual machines in an NFV MANO architecture and brings significant advantages to simplify the automation integration with the management layers such as the VNFM and NFVO network. In this architecture, complex VNFs typically expose application-level dependencies to the management layers, which adds complexity to the VNFM. When cloud-native techniques such as dynamic discovery and container scheduling are integrated into the VNF, this integration dependency is simplified and does not impose container technologies into the management layer. By deploying containers on virtual machines, the solution also addresses the documented security concerns around user plane vulnerabilities with containers running on bare metal.

Benefits of being cloud native for the mobile core

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

Microservices

As explained earlier, microservices are componentized, reusable software modules enabling a number of benefits for the customer and the development organization. Microservices expose discrete functions to an application through APIs. APIs must be maintained and versioned and promote reuse of microservices in other applications. For example, a Diameter protocol service could be used across a number of 3GPP functions. 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 example, Java can be used for a control plane service and C, or 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. For example, a new EPC application service can be dynamically added to an application for scalability or service recovery. Conversely, as a service becomes unavailable, it is updated in the service registry so that the container scheduler, such as Kubernetes, knows that this 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 can allow for fully automated in-service upgrades and rollback of the containers that make up the application.

State separation

One of the most commonly agreed-upon 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 should be separated from stateful services: for example, database, file system, or cache. For example, an EPC 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.

Cisco currently supports a common set of stateful repository services used for managing application state: for example, session and subscriber data. This makes sure that application services are stateless and use these common stateful services for state separation between application containers. As part of its evolution toward being cloud native, Cisco is integrating the EPC technology to internally use these state separation services.

Cisco's broader mobile core evolution includes a set of User Data Repository/Unified Data Mangement (UDR/UDM) stateful services for subscriber and identity management across all mobility functions, including third-party network functions. This solution supports multiple storage engines—for example, in memory or on disk, local and georedundant, transient and long-lived data—and is fully containerized in Docker as a cloud-native stateful service.

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, 3GPP 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 an NFV 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.

Continuous integration and deployment

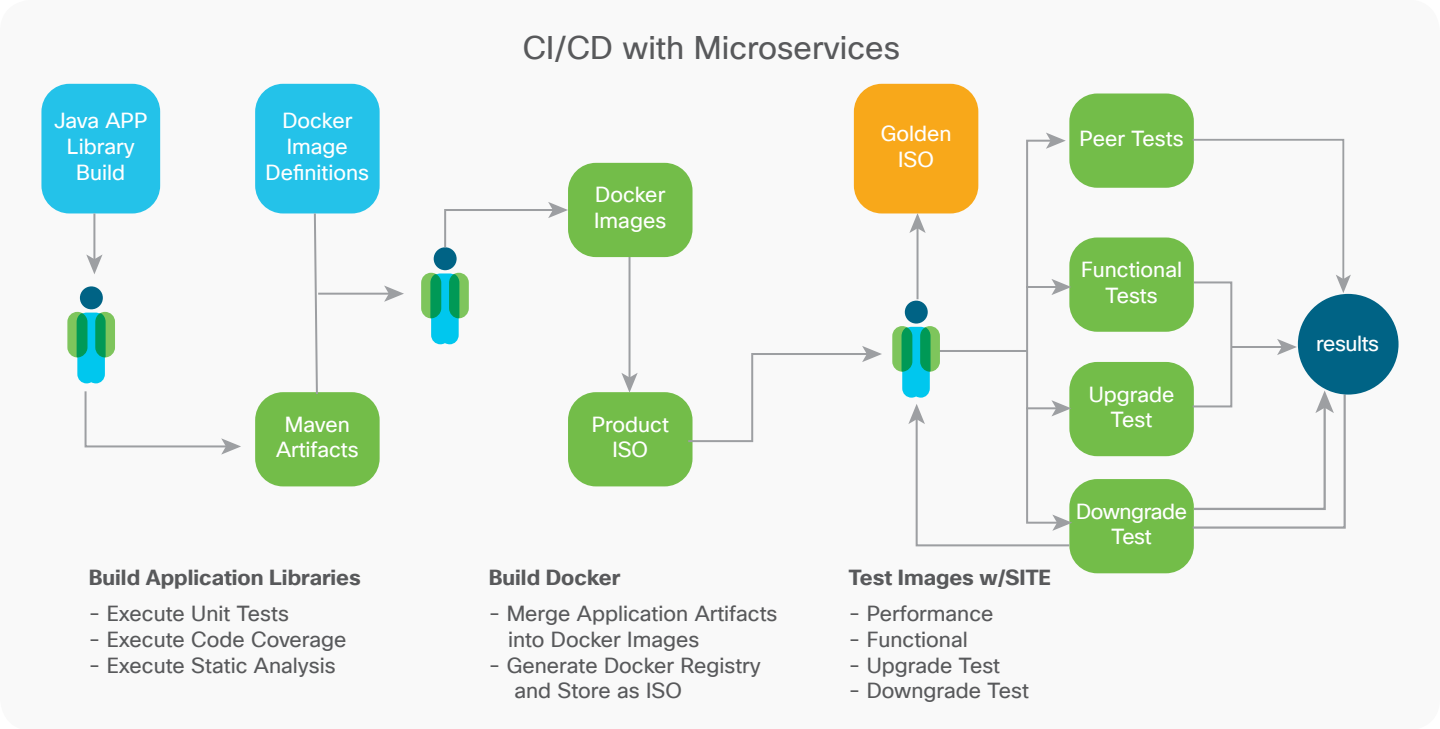
Continuous integration, continuous delivery and continuous deployment are primary pillars for managing a cloud-native application.

Continuous integration encompasses the automated build and unit testing of the software artifacts produced by the engineering team. Developers should be able to run all unit tests locally before merging code into the main integration environment. Successful completion of this process results in an installable set of software images.

Continuous delivery encompasses the automated installation and validation of the continuous integration artifacts into performance and functional test environments. Successful completion of this process results in a production-ready software image. Continuous deployment enables the automation of these production-ready software images into a production environment where they are revalidated and deployed into the live environment.

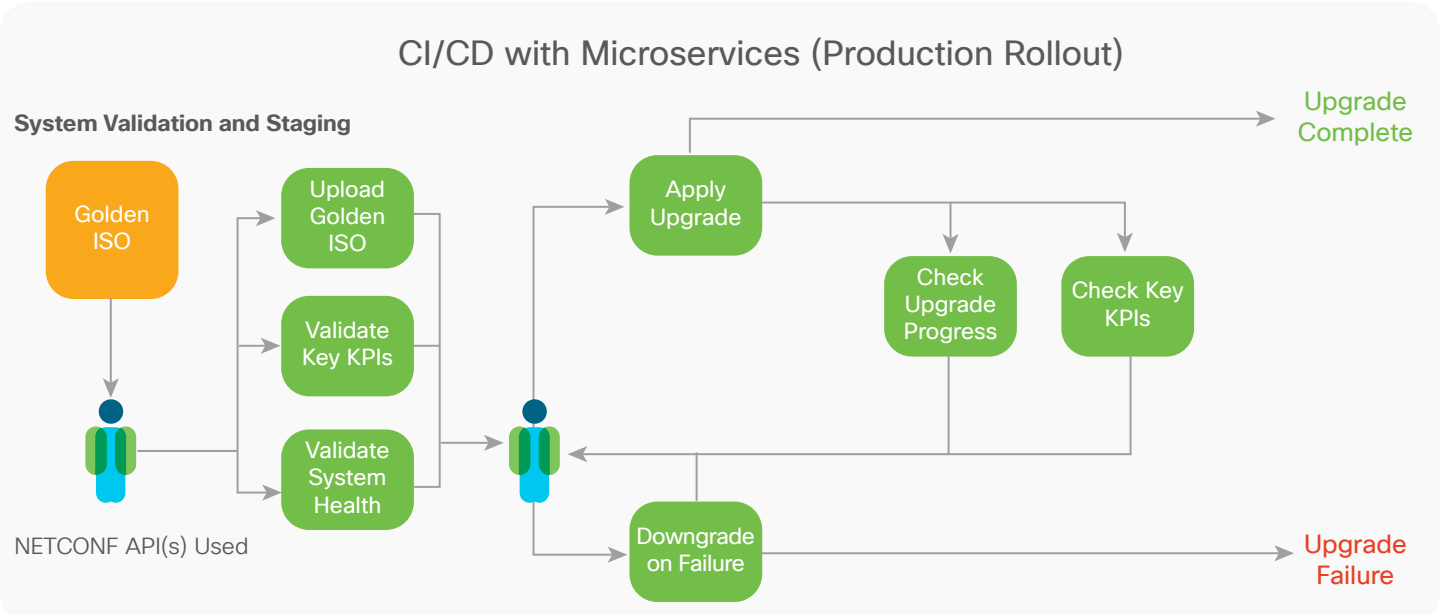
Figure 3 outlines a continuous integration/continuous delivery process to automate the build, packaging, verification and testing of Docker microservices within a development organization.

Figure 3. Continuous Integration/Continuous Delivery



Continuous integration/continuous delivery can then be applied to achieve the automated deployment and validation of a release into a production environment. (See Figure 4.)

Figure 4. Continuous Integration/Continuous Delivery with Microservices (production rollout)



Summary

Service providers must fully automate the deployment and operations of the mobile core to expand revenues realized by offering new services defined in 5G. New 5G use cases such as network slicing and the disaggregation of the mobile core with CUPS and MEC require deploying in the core and at the edge across public, private and hybrid clouds, and NFV-compliant and bare metal architectures. This enables the service provider to offer new vertical solutions and network slices such as IoT, enterprise, MVNO, PMB and consumer networks.

There are specific considerations to realizing being cloud native in the mobile core that are not inherent to web-based cloud-native solutions such as user plane and protocol considerations. Additionally, a cloud-native mobile core must support compatibility between 4G and 5G. This requires the underlying applications of the mobile core architecture to support the application functionality and compatibility of a 3GPP mobile core while introducing the concepts of being cloud native. Cisco has the underlying technology and experience to make sure of a smooth transition from 4G to 5G, maintaining backward compatibility, and addressing the considerations of a cloud-native mobile core.

Cisco's strategy for a cloud-native mobile core 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 mobile core to be fully automated and operated to maximize automation and orchestration to achieve new revenue opportunities and use cases defined in 5G.

Cisco's approach to realizing a cloud-native mobile core is well under way with the realization of policy control and user plane functions and the broader strategy of an end-to-end mobile core solution.

More information

For more information, email mobilecloud@cisco.com.

C11-739215-00 06/17