The bridge to possible

Cisco NX-OS: The Advanced Network Operating System Designed for the Future

Contents

Kernel: Linux Heritage 3

Network Architecture: Performance From the Foundation 4

Package Management: Flexible Administration 5

Container Support: Take Full Advantage of Your Hardware 7

Reliability: Performing When You Need It Most 8

Programmability: Intent-Based Data Center Networking 8

Network Visibility: See Everything Within Your Network 10

Virtualization: Deploy NX-OS in Any Environment 10

Automation: Nexus Family Coordination 10

Nexus Platform Summary: The Correct Choice for Any Network 11

Automated workflows and virtualization technologies have led to dramatic improvements in data center scale, agility, and efficiency. Application developers, server administrators, cloud teams, and DevOps teams have been utilizing the processes and tools around automation for many years, resulting in streamlined and less error-prone workflows. These teams can keep up with the speed of business requirements and market transitions due to these modern workflows. Leveraging open development frameworks has been essential for innovation.

Why not leverage these concepts for the network, whose management methods are still dominated by human-to-machine interfaces?

Cisco has filled this gap with the Cisco Nexus Operating System (Cisco NX-OS). NX-OS is an extensible, open, and programmable network operating system that runs on the Cisco Nexus platform. This rich software suite is built on a Linux foundation that exposes APIs, data models, and programmatic constructs. Using Application Programmatic Interfaces (APIs) and configuration agents, operators can affect configuration changes more programmatically and efficiently .

Cisco NX-OS is based on a highly resilient, Linux-based software architecture and has been carefully built to ensure performance, security, and reliability in the most demanding data center environments. Our network operating system enables all types of application deployments and is the fulcrum of networking solutions for next-generation data centers and cloud networks. NX-OS is the industry's most deployed data center operating system and can be found across the globe.

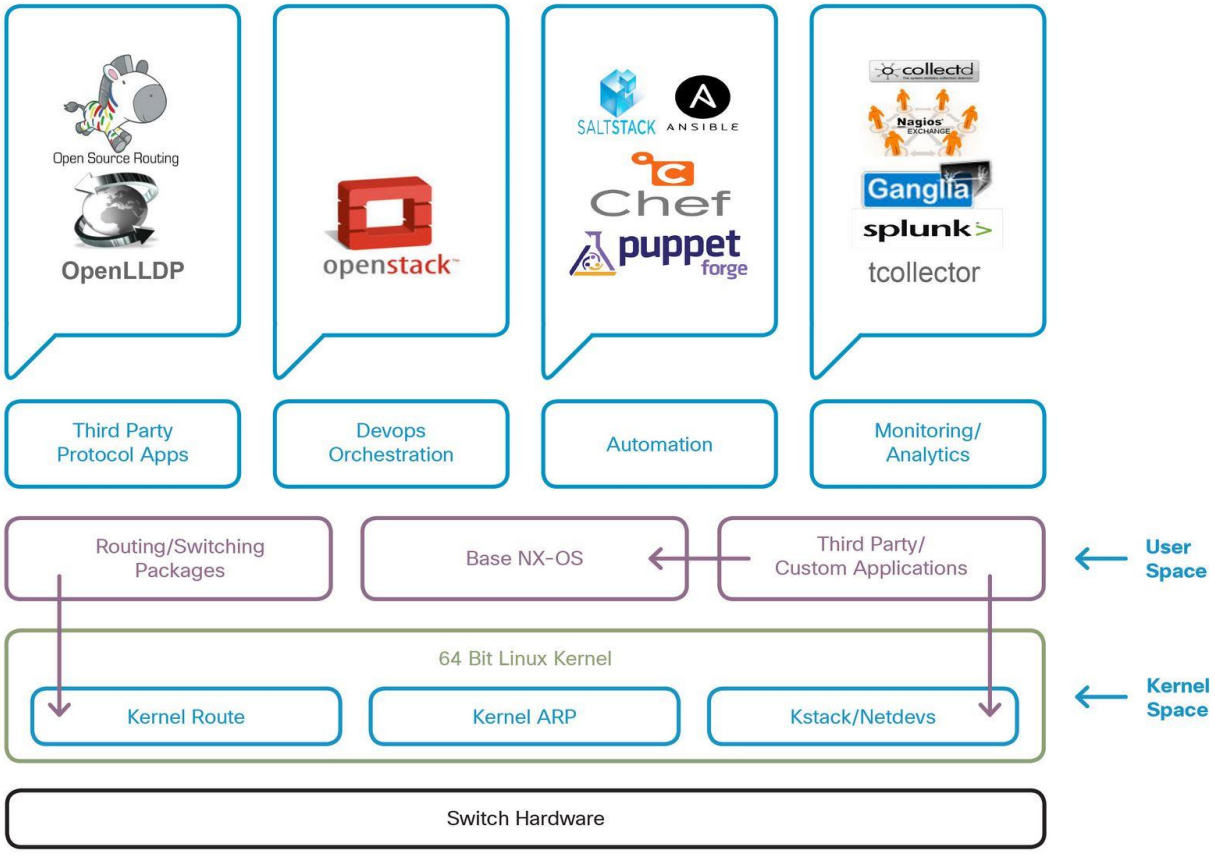
Kernel: Linux Heritage

The core of NX-OS is based on version 4.19 of the 64-bit Linux kernel. Kernel version 4.19 provides the OS with a healthy balance of features, maturity, and stability. This serves as a solid foundation for programmability, observability, and Linux-based management of Nexus switches. Additionally, NX-OS leverages the native Linux networking stack. Switch interfaces, including physical, port-channel, vPC, VLAN, and other logical interfaces, are mapped to the kernel as standard Linux netdevs. The implementation for VRFs also utilizes the kernel by mapping to standard Linux namespaces. This native interface management for all port types allows for open-source and custom tools to be utilized on the Nexus platform.

- Interface Management: Leverage standard Linux utilities such as ifconfig, ethtool, and route for network interfaces, routing, and other associated tasks.
- Tools for Troubleshooting: Utilize tools like tcpdump, ping, and traceroute for your troubleshooting needs.

- VRF Capabilities with Namespaces: Each VRF created within NX-OS will have a corresponding namespace associated with it. This will maintain the VRF isolation between NX-OS and the Linux kernel.

Due to the foundation of the OS, any standard Linux-based application can be run on the platform without changes or wrapper libraries. Users can leverage their standard Linux server management tools and workflows to install their custom-developed applications, or other open-source programs, and have them function out-of-the-box on the Nexus platform. NX-OS also uses the Linux kernel's networking stack to send and receive packets, therefore, applications can communicate with the OS without custom modifications. This also allows for straightforward integration with common third-party configuration management agents like Puppet and Ansible, and telemetry applications such as Ganglia, Splunk, tcollector, and Nagios.



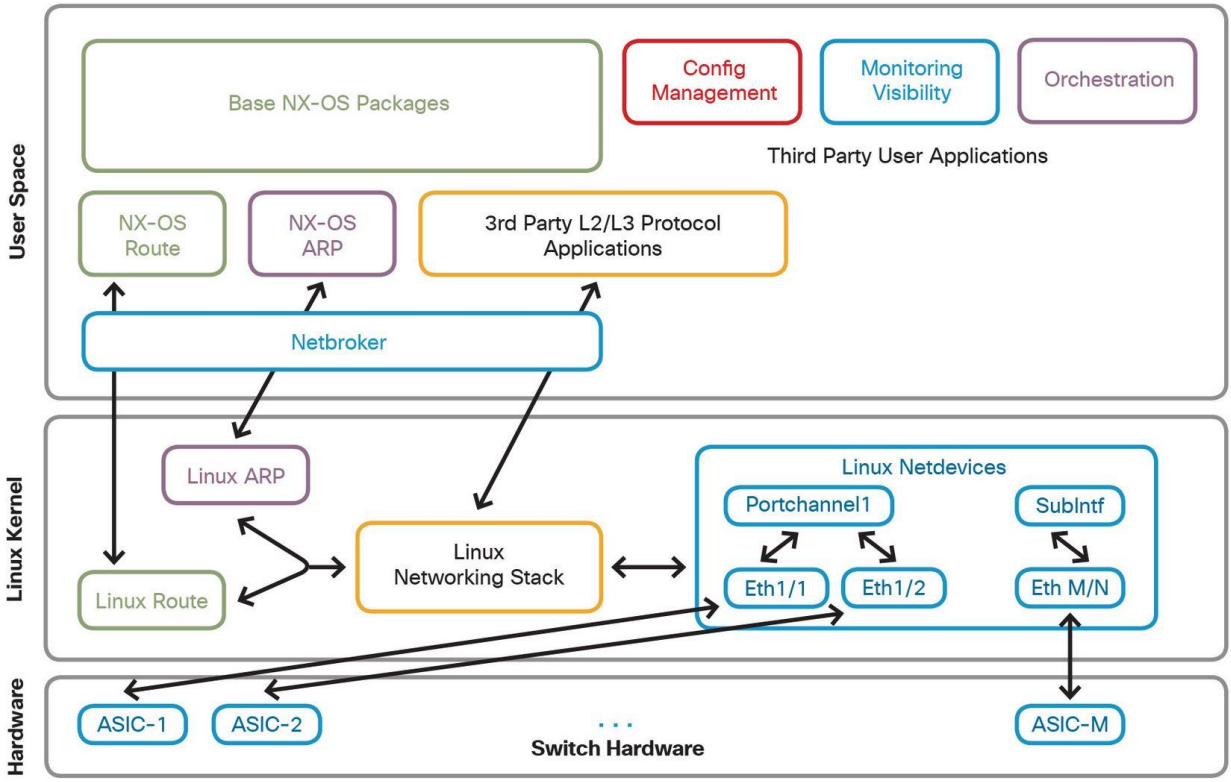
Shell access is also securely provided on the Nexus platform by using the Bourne-Again Shell (Bash). Using Bash enables access to the underlying Linux system on the device. This bash shell is included in the NX-OS image and can be enabled at any time.

Network Architecture: Performance From the Foundation

The NX-OS network architecture is made up of two primary layers:

- User space processes and software comprised of traditional NX-OS software processes (OSPF, vPC, BGP, NXOS, ARP, etc.), and third-party user applications (configuration management, visibility, analytics, custom-built agents/tools).
- 64 Bit Linux 4.19 kernel layer. This interfaces with Linux kernel net devices and the Linux networking stack (route, ARP tables, etc.).

The network architecture gives access to the Linux kernel networking stack, where the switch's physical and logical interfaces are represented as a net device and an IP address at the kernel level. This design opens the door to management of the routing and front panel ports using unmodified Linux-based tools and applications. However, there needs to be a synchronization function between NX-OS and the kernel, to ensure the two layers work effectively in tandem. This synchronization function between user space NX-OS processes and the kernel layer is provided by the netbroker module, which ensures changes implemented to physical and logical interfaces in NX-OS are reflected correctly to the Linux netdevice interfaces.



Package Management: Flexible Administration

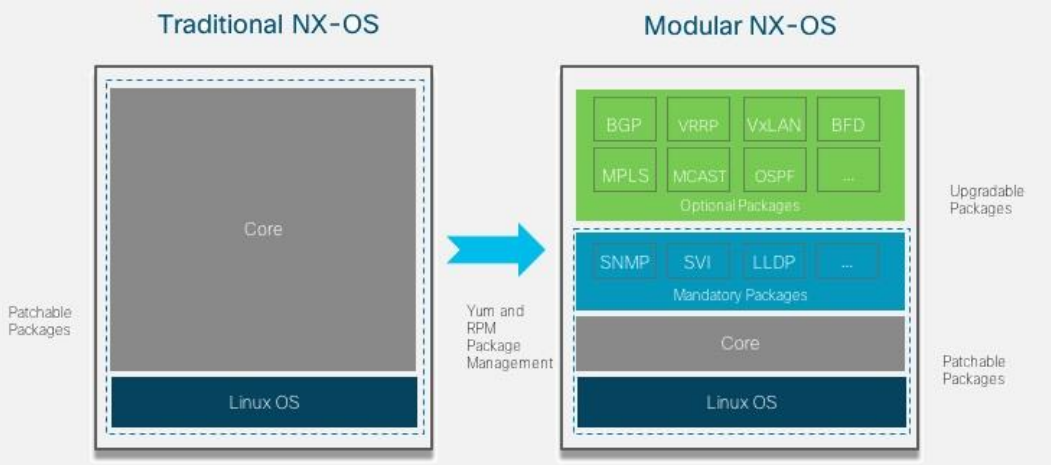
NX-OS leverages standard package management tools, such as RPM and YUM for software management. These tools ease the deployment and administration of external or custom-developed applications in the Bash shell. The package managers provided with NX-OS give you automated and programmable control over your external software packages.

Cisco's NX-OS provides the following two environments for installing packages:

- Bash Shell: The native NX-OS Linux environment.
- Guest Shell: A secure Linux container environment running CentOS.

The modular design of NX-OS gives package managers the ability to control which packages are deployed to your devices. By leveraging these tools, administrators can customize the NX-OS images used in their environments. This customization provides the following advantages over a single monolithic image:

- Reduce the NX-OS image footprint.
- Asynchronously deliver features to your environment.
- Independently fix features by deploying new packages, instead of upgrading the entire system image.
- Ensure features cannot be accidentally enabled in your environment (because they were not deployed to the device).
- Features have clean code boundaries and do not affect the rest of the NX-OS image.



Yellowdog Updater, Modified (YUM) is the default package and repository management tool for many operating systems, including NX-OS. The YUM package management infrastructure provides the following benefits:

- Automatic resolution of software dependencies.
- Easy to use command line interface to install or upgrade software.
- YUM can browse or search multiple software locations simultaneously for a specific package.
- YUM can use either local or remote software repositories to upgrade or install software. RPM package files in these repositories are organized in a hierarchical manner so they can be found by the YUM management tool.

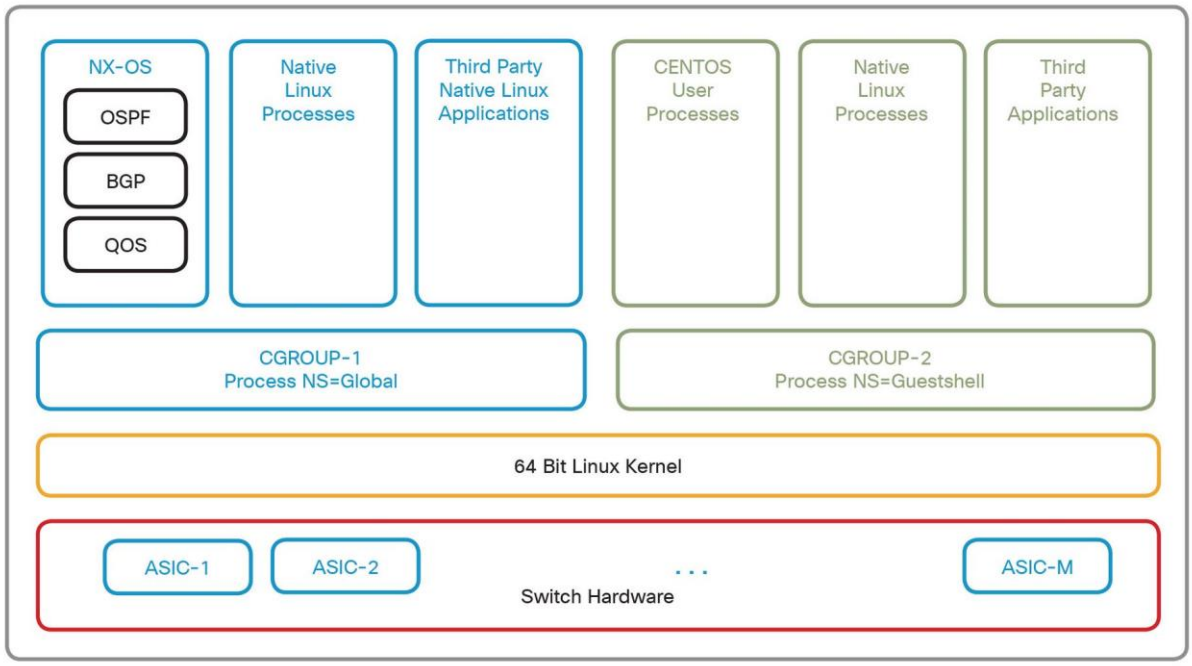
Starting in version 10.x, NX-OS has updated to Dandified YUM (DNF), which is the next-generation version of YUM. This ensures NX-OS will be fully compatible with your environment and will be able to scale with all of your package repositories.

Container Support: Take Full Advantage of Your Hardware

NX-OS supports running Linux Containers (LXCs) and Docker containers directly on the platform and provides access to a CentOS Guest Shell. This allows customers and third-party application developers to add custom functionality directly on the device in a secure, isolated environment. The benefits of utilizing containers include:

- Isolated and secure application execution environments.
    - Resource and process isolation. Resources such as CPU, memory, storage, and network bandwidth can be controlled and limited for each environment.
- Independent software release cycles.
    - Customers and third-party developers can create their own open-source applications, independent of the NX-OS software release cycle.
- Network operation optimization
    - Custom applications running within a container can reduce the need to use separate software solutions or interfaces. For example, a packet capture tool running locally on the device, or an application monitoring the state of the device.

The diagram below shows a high-level overview of the container system architecture included in NX-OS.



The Linux kernel provides built-in resource control mechanisms (cgroups) which allow for the isolation and limiting of system resources. This isolation is transparent and does not create any process-scheduling overhead. The processes within a container are given their own namespace which does not overlap with

the native system processes or any other containers that may be present. This ensures that NX-OS processes are protected and isolated from the containerized guest processes.

Reliability: Performing When You Need It Most

Due to its heritage, NX-OS has always abided by the Linux best practices for OS structure. The following categories are just a few of the many ways NX-OS ensures your network will be ready to perform for you:

- Modularity: Modules are loaded into the kernel only when needed. Modules can be loaded and unloaded on demand.
- Fault Isolation: Provides complete process isolation for NX-OS features, services, and user application processes.
- Resiliency: Gracefully restarts or initializes processes following unexpected exit conditions (segfault, panic, etc.).

NX-OS utilizes a unique multi-process state-sharing architecture that separates an element's state from the parent processes. This reflects Cisco's core software design philosophy and enables fault recovery and software updates to individual components, without affecting the running state of the system. Protocol routing and switching processes, security functions, management processes, and even device drivers are decoupled from the kernel. These modules and processes run in the user space, not the kernel space, which ensures process control and system stability. This modular architecture allows for the updating and restarting of individual processes without requiring a full device reload.

The NX-OS architecture also allows for hitless, zero-packet-loss, In-Service Software Upgrades (ISSU). Additionally, fast reloads are also possible when performing an upgrade. This provides stateless process restarts with minimal impact to both the control and data planes. Hardware flexibility is also a key feature included in the Nexus family. Graceful insertion and removal of devices allows hardware to be seamlessly added, replaced, or removed. Any process without CPU affinity supports stateful process restarts. Additionally, these features are patchable through RPMs or regular software maintenance upgrade packages through the NX-OS CLI or Linux workflows.

Programmability: Intent-Based Data Center Networking

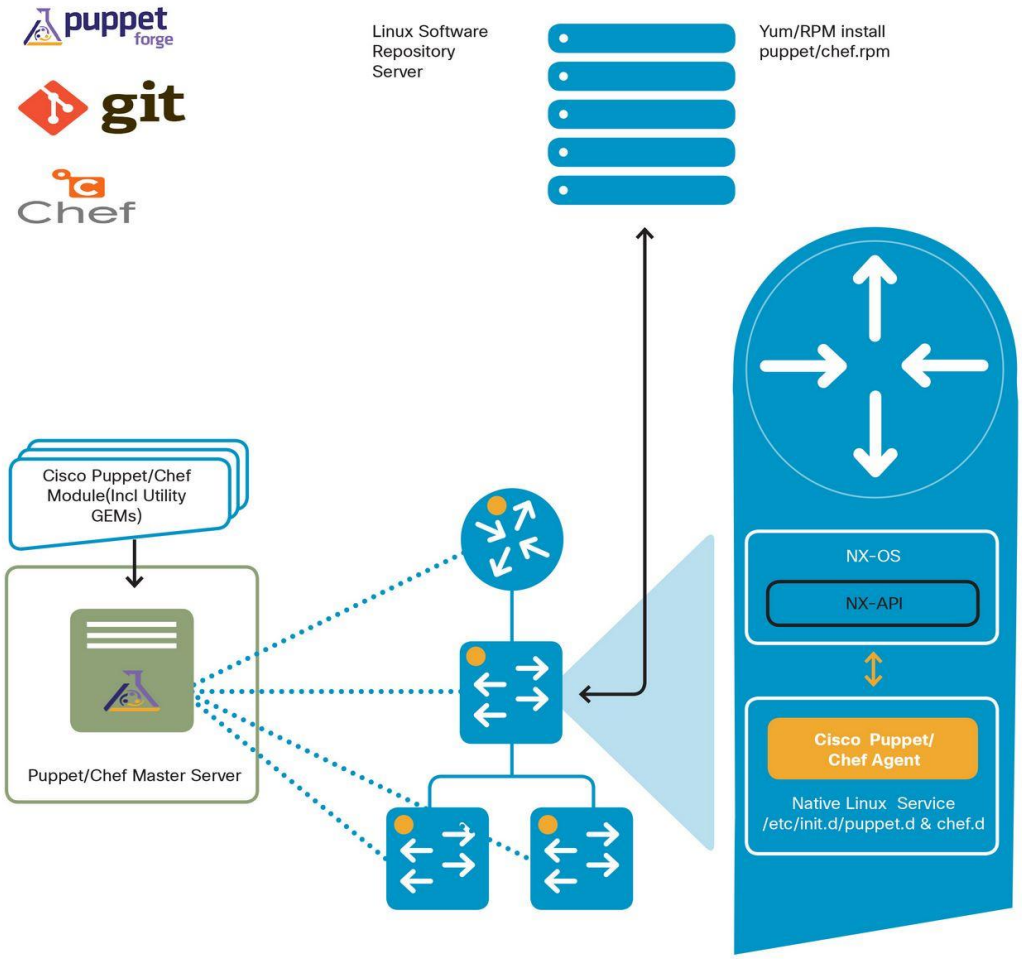
Cisco NX-OS has been designed for easy deployment, development, and management. Zero-touch provisioning is made possible by the Power-On Auto Provisioning (POAP) feature. This feature allows the device to utilize the Preboot eXecution Environment (PXE) to facilitate the boot process and initial configuration of the Nexus device. Zero-touch provisioning is further simplified by the use of the same NX-OS binary image for all Nexus 9000 and Nexus 3000 devices.

NX-OS also provides a toolchain to build custom packages and agents for the platform. Cisco has published an extensive SDK to enable a build environment that can be installed on a Linux server. This provides the ability to download a build agent that will incorporate the source code in the local directory structure. This SDK allows administrators to build and package binaries for use on NX-OS. You can choose to deploy these applications directly on the Linux filesystem or in an LXC container.

The NX-OS platform also provides flexibility in tool choice when it comes to programmability. The platform can be modified in many ways to ensure it integrates smoothly with your current toolset.

- Python Libraries: There is embedded Python shell support with native NX-OS libraries that can be utilized for development.
- NX-API REST: Interact with network elements through RESTful API calls. This allows for a data model-driven approach to network configuration, automation, and management.
- NX-API CLI: This provides the ability to embed NX-OS CLI commands in a structured data format (JSON or XML) for execution on the device via an HTTP or HTTPS transport. The data returned from these calls will also be formatted in JSON or XML, making it easy to parse the data with modern programming languages. A sandbox testing environment (NX-API Developer Sandbox) also exists as a development tool.
- Extensive Native YANG and industry-standard OpenConfig model support through RESTCONF/NETCONF and gRPC.

These flexible interfacing choices give configuration management tools the ability to provide true intent-based automation.



Automated device management also eases lifecycle operations such as firmware management, compliance auditing, and performance monitoring.

Network Visibility: See Everything Within Your Network

Visibility within the network has become an increasingly important element of the data center. Cisco NX-OS provides the visibility required to ensure your network is easy to analyze and within compliance.

- Network software states are exposed through a comprehensive publish/subscribe centralized database (DME) which is accessible through gRPC/Protobufs, HTTP/JSON, and YANG state streaming.
- Cisco cloud-scale ASICs enable flow state awareness and granular sub-second access to utilization metrics for microburst awareness and queue occupancy for large-scale RDMA over Converged Ethernet (RoCE) storage environments.
- Advanced buffer monitoring reports real-time buffer use per port and per queue, which allows organizations to monitor traffic bursts and application traffic patterns.
- Highly configurable network traffic monitoring fabrics can be built with Cisco Nexus Dashboard Data Broker (NDDB). NDDB helps you build simple, scalable, cost-effective, and highly performant Test Access Points (TAP) and Switched Port Analyzer (SPAN) aggregation networks that run at full line rate.
- Cisco Nexus Dashboard Insights gives you the information needed to maintain, troubleshoot, and expand your data center. Gain visibility into how your network is performing, how it's being utilized, and how it will respond to high-load scenarios. Nexus Dashboard Insights will also monitor your network's control plane, data plane, and forwarding plane to aid you in finding the root cause of network anomalies and reduce recovery time.

Virtualization: Deploy NX-OS in Any Environment

As workloads continue to move to the cloud, virtualized deployments continue to be an integral building block of fabric architectures. The Nexus family fits into every element of your data center with the virtual images of NX-OS. N9Kv (Virtual NX-OS) extends automation and operational models for DevOps and NetOps integration with images based on VMware, KVM, and Fusion. Additionally, deployment and management of N9Kv instances are simplified since it utilizes the same NX-OS image as all physical devices in the Nexus family. Cisco provides extensive support for N9Kv as it is an integral piece to any cloud workload.

Automation: Nexus Family Coordination

Cisco provides in-house automation tools to complete the Nexus family. Data Center Network Manager (DCNM) is a network management platform for all Cisco NX-OS-enabled deployments. DCNM can assist you with fully automated deployments of many fabric architectures, such as IP fabric for Media, storage networks, BGP EVPN-based VXLAN, classic core networks, aggregation networks, access deployments, or any other NX-OS-based fabric you may need. DCNM will accelerate your provisioning from days to

minutes and simplify new deployments from day zero through day N. Reduce troubleshooting time with the DCNM web GUI which provides a centralized dashboard to review network topology, network fabric architecture, and current infrastructure status. The DCNM platform ensures configuration errors are never introduced into your fabric. Automate ongoing changes in a closed-loop, with templated deployment models and configuration compliance alerting with automatic remediation. DCNM can also help upgrade devices in your fabric to ensure a smooth, automated upgrade. The web GUI allows for streamlined overviews of both your physical and virtualized infrastructure.

The DCNM GUI also includes a Fabric Builder to ensure you create, modify, or delete your VXLAN BGP EVPN fabrics according to the Cisco recommended best practices. DCNM Fabric Builder can help you with many tasks, including:

- Creating new VXLAN, MSD, and external VXLAN fabrics
- Viewing VXLAN and MSD fabric topologies, including connections between fabrics
- Updating fabric settings
- Saving and deploying fabric changes
- Deleting fabrics
- Adding switches to a fabric
- Provisioning start-up configurations and IP addresses to new switches through POAP
- Updating switch policies
- Creating intra-fabric or inter-fabric connections

Nexus Platform Summary: The Correct Choice for Any Network

Kernel: Leverage the many advantages of Linux through the NX-OS Linux-based design and utilize the tools you're already familiar with.

Network Architecture: The NX-OS network architecture gives it access to the Linux kernel networking stack. This allows the switch to achieve maximum performance on all interfaces.

Package Management: Deploy and maintain all of your favorite tools with built-in package managers. Deploy these tools in the guest shell or native Linux environment. You can also use these package managers to customize your NX-OS image.

Container Support: Deploy your applications in Linux Containers to add custom functionality. NX-OS will ensure they are contained within isolated execution environments so you can run your applications with peace of mind.

Reliability: NX-OS will load modules only when needed, provide complete process isolation, and gracefully restart processes if they experience unexpected exit conditions. NX-OS ensures your network is always ready to perform.

Programmability: Program and automate your network with the many automation abilities of NX-OS. Use Python libraries to customize device behavior, utilize the NX-API for a data model-driven approach to network configuration, and send CLI commands to devices for scalable configuration changes.

Network Visibility: Gain visibility into your network through the detailed data provided by NX-OS. Monitor port, buffer, queue, and hardware utilization. Cisco also provides multiple methods for remotely monitoring your network.

Virtualization: Combine the benefits of NX-OS with the elasticity of the cloud with N9Kv. Utilize the same tools to manage and monitor your physical or virtual NX-OS devices.

Automation: Cisco designed the Nexus family with scale in mind. DCNM can deploy, modify, combine, or decommission an NX-OS-based network. Ensure your network is set up correctly and quickly with Cisco DCNM.

Whether you are deploying a new network fabric or incorporating Cisco Nexus products into your existing network, you can be sure that NX-OS and the Nexus family is the right choice for your environment.

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 07/21