Cisco Network Services Orchestrator (NSO): The Bridge to Automation

Summary

Whether architecting a service orchestration toolchain or building a DevOps environment, the underlying automation strategies have typically expected developers to understand how infrastructure works and infrastructure owners to be conversant with application development and service creation. Real world experience tells us that this approach is flawed. Cisco® Network Services Orchestrator (NSO) offers a more realistic approach by serving as bridge between application or service owners and infrastructure owners, letting each team operate in their native environment, yet still collaborate effectively together. Years of operational experience have produced a platform with features and operational capabilities that these teams will find valuable as part of any automation initiative, enterprise or service provider.

Contents

Automation matters

Robust automation capabilities are generally recognized as a competitive advantage, since they can help improve customer experience, increase revenues, and lower costs. Market pioneers are pairing automation with concepts such as agile development, continuous integration, and continuous deployments (CI/CD); and reshaping how they operate with demonstrated positive results. One of the keys to success with these initiatives is robust sophisticated tooling to support an automation strategy.

Industry leaders all share certain traits: they move quickly, they strive to deliver a great customer experience, and they have a handle on prices and costs. More often than not, the result is both growth of revenue and profitability, which lets these organizations keep shareholders happy and re-invest in the business. Successfully initiating and maintaining this virtuous cycle requires an on-going commitment to aligning people and skills; processes and culture; and technology that can be challenging to even the most committed company. While all three elements are important, this paper will mostly focus on the last element: technology (although it is impossible to escape discussing the other two). Pioneering work by companies like Google, Netflix, Intuit, and WalMart has shown how automating processes can speed service creation, improve customer experience, and lower costs. Automation is recognized as a competitive advantage, and in the last few years, the question has changed from, “Should I automate?”, to “How do I automate?”

How automation helps and why it often disappoints

Whether you are a mobile operator, a retailer, a media company, or a bank, your business operates based on a number of core processes that you repeat on a regular basis. How well you repeat those processes has a lot to do with how happy your customers are, how happy your employees are, and how long you will stay in business. How quickly can you respond to customer requests? How much friction is there in your order process? How quickly can you roll out new products and services?

Market leaders like the ones mentioned earlier have applied automation through three foundational concepts which help improve the speed and quality of how they get things (software, products, services) from concept to customer:

Agile development

Agile philosophy focuses on accelerating delivery by removing process barriers, fostering collaboration between customers and engineers, and continuous rapid incremental improvement. While “agile” started as a software development methodology, its concepts have come to be successfully applied to everything from manufacturing to marketing to service creation. Automation compresses development and testing timelines.

Continuous integration/continuous delivery (CI/CD)

CI/CD pipelines are concerned with rapid, dependable implementation. Continuous integration is a software practice. While it typically discussed with regards to integration of bug fixes and new features into the production code, it could just as easily reference router configuration changes or firewall rules. Continuous delivery is concerned with deployment of software and associated infrastructure. The concept of "infrastructure as code" is the application of CI and CD to infrastructure software and configuration management. CI/CD is intrinsically dependent on automation as manual processes not only introduce errors, variances, and delay; they also throttle scalability and increase costs.

DevOps

At its core, DevOps is about driving a cultural shift to break down typical organizational stove-piping and cultivating a culture of collaboration and shared ownership. Ironically, a key to DevOps success is tooling and automation that allows greater autonomy between teams.

Theory vs. reality

Automation works its magic because it accomplishes three things:

1. By handling monotonous processes that make up the vast majority of the operational time (i.e., a user needing an IP address for a web server) it frees your teams to work on higher-value tasks like better understanding the needs of customers and stakeholders (for instance, learning how important business apps and services work).
2. Automated processes are executed in a consistent way so downstream processes don't have to deal with errors and variability, and customers get predictability and a better experience.
3. Automated processes run on demand, at machine speed, night and day. This shortens development, testing, and deployment cycles for everyone. For the organization, this translates to faster time to market, increased productivity, and lower costs.

The theory is simple and straightforward. Reality is a bit more complicated.

Automation depends on software being able to assert programmatic control over the physical world—on conceptual goals successfully crossing over into real-word action. Even virtual resources (VMs, containers, etc.) are still provisioned on physical infrastructure. For instance, when a service owner clicks a button on her screen to deploy another instance of her app, that action must eventually translate to VMs being spun up, network ports being brought up, IP addresses being allocated, firewalls and load balancers being updated, amongst other things. Real-world customers can be complicated and demanding, and infrastructure can be equally complicated and demanding, so success depends upon the robustness of the bridge connecting them both (see Figure 1).

This is a role where Cisco NSO shines: ensuring infrastructure intent matches infrastructure reality. Cisco NSO has been deployed for almost a decade in some of the world's largest and most complex multi-vendor production environments. This white paper will examine how that experience has translated into a set of capabilities and tools that both developer and operations teams can use to efficiently and comprehensively implement automation strategy.

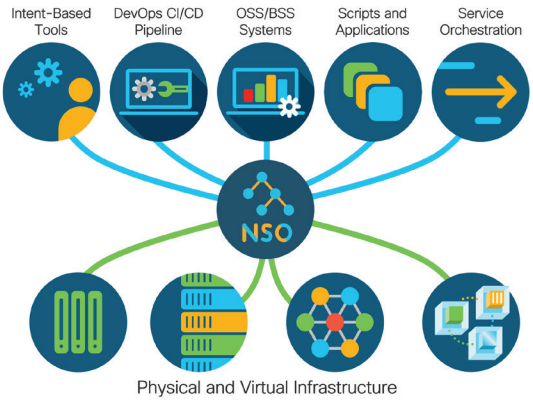


Figure 1 - Bridging worlds

Building a better bridge

Years of experience helping customers automate tier-1 environments has taught us some critical design principles that support the goals of DevOps:

Scalable sophistication

Automation tooling is like math. Elementary school arithmetic is fine if you need to double a pie recipe, but you'll need something like calculus if you are trying to land a rocket on Mars. Similarly, you need tooling that is simple enough to get started easily but powerful enough for your more sophisticated initiatives. As your goals gets more complicated, you need to make sure you have the means to adequately express them. Scale is not just about increasing complexity; it's also about being able to handle an ever-increasing number of services, apps, and devices.

Flexible and adaptable edges

Change is inevitable, so your bridge must be able to easily accommodate new apps and tools on the "north" side of the bridge. Similarly, your bridge must be able to accommodate changes in infrastructure on the "south" side, such as new vendors, virtualization/containerization, and adoption of cloud.

Normalization

The bridge must be able to abstract the view from the other side—hide the complexity and heterogeneity that is reality. App owners should be able to ask for resources without caring about the implementation details. Similarly, infrastructure owners should see requests in a consistent way regardless of the app, tool, or system making the requests. Because of this normalization and abstraction role, the bridge must also serve as the authoritative source of truth for both sides of the bridge as to what is actually occurring.

Developer-centricity

The quality of the programmatic controls available to both sides of the bridge will dictate what can be accomplished and how quickly it can be accomplished.

Together, these four principles help deliver on the promise of DevOps. They allow the entire organization to work more cohesively while allowing each team to function autonomously. Infrastructure owners can change and optimize their resources without breaking apps and services. Similarly, new apps and services can be rolled out more quickly and without creating infrastructure churn. Costs are lowered through increased agility, efficiency, and productivity. Customers are happier with products and services that are more relevant, delivered more quickly, and with more predicable quality.

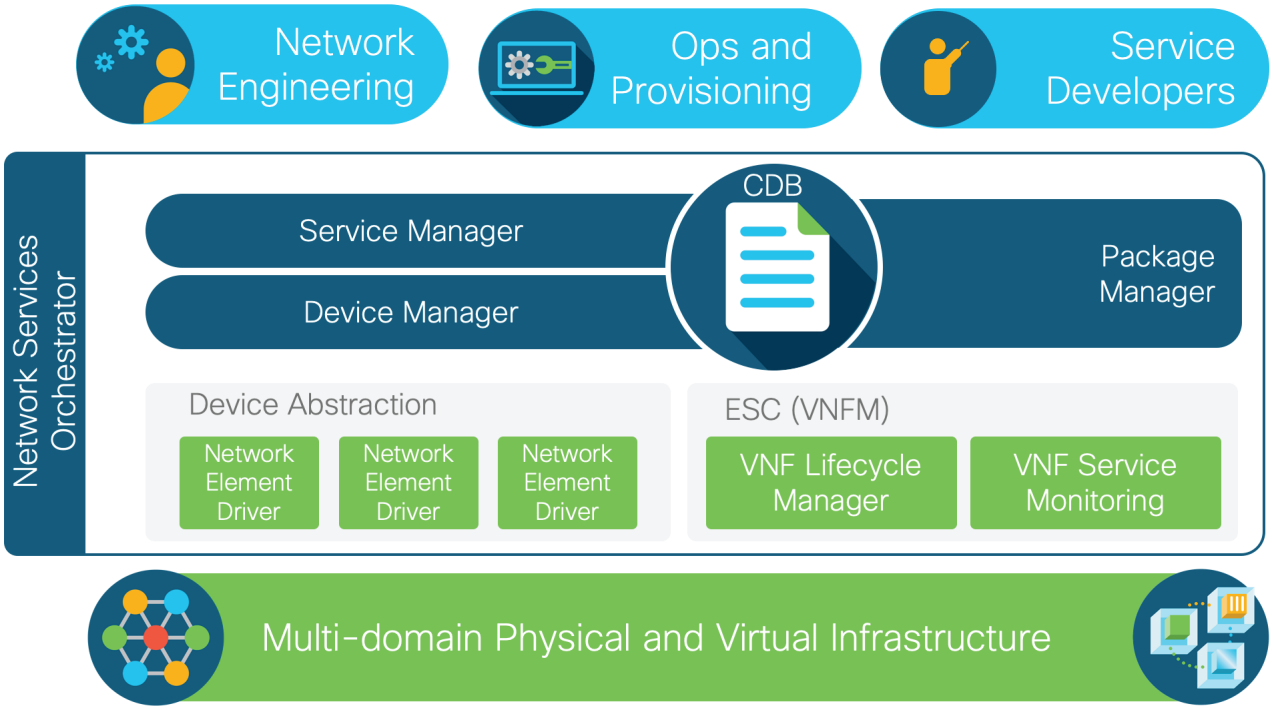
Cisco Network Service Orchestrator (NSO)

Cisco NSO 5 represents nearly a decade of accumulated wisdom in automating large, complex tier-1 environments. While NSO has its roots in the service provider market, in recent years more and more large enterprises are seeing the need for a proven automation solution. Figure 2 provides an overview of the elements of NSO and how they relate.

At a very high level, NSO has three components:

1. A model-based programmatic interface that allows for control of everything from simple device turn-up and configuration management to sophisticated, full lifecycle service management.
2. A fast, highly scalable, highly available configuration data store that serves as a single source of truth.
3. A device abstraction layer that uses network element drivers (NEDs) to mediate access to both Cisco and more than 150 non-Cisco physical and virtual devices.

Figure 2 - NSO architecture



Taken together, these components allow NSO to provide a single, network-wide interface to all network devices and services—both physical and virtual—using a common modeling language and data store. NSO lets development teams define services and hand them off to operations teams that can then automate service activations and changes quickly and simply, moving from high-level intent to granular device configurations in a single transaction.

Looking a little deeper, real-world experience has given NSO some key properties, including:

A true model-driven system

NSO can automatically generate a single, well-defined API into the entire network environment. Using the standardized YANG modeling language you can model and automate any type of device—layers 1 through 7, physical or virtual, addressed traditionally or via software-defined networking (SDN) overlays. And you can model any type of service or policy.

Real-time configuration database (CDB)

NSO captures the real-time configuration state of every device and service in the network. In a world where network provisioning and operations teams often work with data that is as much as 70 percent inaccurate, NSO can provide a single, scalable, continuous source of truth for the network.

Stateful convergence

To achieve end-to-end automation, an orchestrator should be able to receive the "intent" of the service and translate that to real change in the network. Many networking organizations currently rely on workflow definitions to accomplish this—and often find themselves overwhelmed by a constantly growing body of workflows to account for each unique case. NSO takes a different approach, using the concept of state convergence. Using the same common data models and modeling language to describe services and devices, NSO fully automates the creation, deletion, and run-time modification of network services. It maps design-time service definitions to run-time network operations through a single, flexible data model for a service. NSO's stateful convergence algorithm then derives the minimum network changes required and executes them.

Multi-domain orchestration

Automation tools have typically been bound to a technology domain: a tool for the data center network, a tool for the WAN, a tool for the optical network, and perhaps tools to manage firewalls and other L4-7 devices. NSO can span multiple technology domains allowing you to automate cross-domain service chains much more easily and dependably.

For the virtual realm, NSO includes the Cisco Elastic Services Controller (ESC) as part of the core platform. ESC provides the essential capabilities needed to automate the full lifecycle of virtual resources, virtual machines (VMs), or container-based networking elements as part of end-to-end services. As part of a single service model, NSO can trigger the launch, configuration, and ongoing monitoring and license management of virtual network functions (VNFs)—both individually and in complex service chains.

Getting there from here

While NSO provides the tools to support DevOps goals, building the organizational expertise developing processes and shifting organizational culture to take gain maximum benefits from DevOps should be an incremental, collaborative, iterative process. Start small, implement something simple, learn from that experience, and then repeat with something a little more challenging. We recommend the following progression.

Phase 1 - Use NSO as a programmable network interface

Use NSO to provide a single API into the network. Operations gains a network provisioning and configuration power tool, with the ability to perform network-wide command-line interface (CLI) and configuration changes from a single interface, in a single transaction, instead of having to individually touch multiple boxes and use different, device-specific commands.

Phase 2 - Use NSO for service abstraction

NSO draws on device and service models to begin more fully automating service activations and changes. You see an end-to-end view of the service as a whole, instead of just seeing the individual device configurations.

Phase 3 - Use NSO for DevOps infrastructure automation

As you make the people and process changes to support agile development and CI/CD, NSO can support that change by enabling everyone involved in the service—product developers, network engineers, provisioning and operations teams—to work together to design and execute new services and changes, quickly and continuously.

Let's look at each of these phases in greater detail.

Phase 1: Build a programmable network interface

Despite the fact that they may be juggling hundreds or thousands of device configurations, network engineers often rely on manual processes—or largely manual ones, like CLI scripting—that are fragile and labor-intensive. NSO provides a better way using the configuration datastore (CDB) and the abstraction/normalization of the network element drivers (NEDs) to provide a much more robust and resilient way of handling configuration management through two mechanisms:

Transactions

Configuration changes are handled like database transactions: all changes are applied at once, and if any part of the change fails, the entire transaction rolls back.

Synchronization mechanism and diff engine

NSO can compare the configuration in the CDB to a device and highlight differences. NSO can also synchronize in either direction. It can bring the device back in line with what the CDB expects or update with CDB with the device configuration (i.e., to capture out-of-band updates).

These two mechanisms combine to ensure configuration changes are implemented in a trusted fashion:

1. NSO receives intent (what the network should look like).
2. NSO compares the desired state to the current state and presents a "diff" before proceeding.
3. NSO updates the device to match the desired state.
4. NSO then reads back the device configuration to ensure it matches the desired state.

This process is at the heart of NSO and, by itself, it can make life significantly simpler for operations:

Automated device configurations with network-wide CLI and REST interfaces

NSO manages device configurations for the entire network with a single interface and consistent syntax. Network engineers and operations teams can use the same tools they use now—CLI scripting or REST interfaces—to manipulate the configuration lifecycle of hundreds or thousands of devices as a single set. They can put network elements into groups and make template-based configuration changes to large swaths of the network at once.

Golden configurations

Network engineers can use templates to ensure that all devices of a certain type or group comply with a particular configuration. And they can update the golden configuration template and apply it to all devices in that group automatically. Simply having templates to describe the proper configuration of all devices is a huge benefit to network engineers and operations teams currently relying on decentralized, manual processes to try to keep up with sprawling heterogeneous networks.

Configuration compliance reporting

Once golden configurations are applied, network engineers and operations teams can use NSO to poll the network for any element that deviates from the template. They have direct access to all devices and can immediately capture any element that has undergone out-of-band. Engineers can then update the golden configuration to make an exception for a change that's beneficial, or re-run the template to bring the device into compliance.

Phase 2: Service abstraction

This second phase uses NSO to help service owners design, deploy, and modify services while also providing operations teams greater visibility into what is running (or not). Provisioning and operations teams can now make automated changes at a high level, without having to explicitly code each step and address every device and element of a service. Network operations teams now have deeper visibility into services. Rather than examining low-level data from network devices and trying to infer what each service is doing, they can view and trace the services running on the network at the customer-facing perspective.

Developers: Closing the gap between design time and run time

In traditional environments, the people that design and build things (apps, services, products, etc.) rarely talk to the people tasked with operating them. There are often cultural and organizational barriers but these two groups also often lack the tools and language to effectively collaborate. This inability to align becomes a drag on the entire organization.

Without the upfront involvement of infrastructure teams, important requirements and design challenges for a new service may be unaddressed in the service models. What appears like a simple "ask" from a developer might actually be quite operationally complex. Problems may not be revealed until very late in the development cycle or even after release when they are potentially customer

impacting. Attempting to address problems at this point is expensive and the "unplanned work" has a cascading negative impact on all the teams involved.

As organizations start to align around DevOps principles and pursue service abstraction, they begin to bridge the gap between design intent and network execution with tooling like NSO. Now, service designers and developers define new services in human-readable YANG data models. Network teams can then test and deploy them much more quickly, because the services are written in a language they (and their network tools) already speak.

Service provisioning: Time to market

The rule for provisioning is simple: faster is better! Latency in the provisioning process usually translates to lost revenue or diminished customer experience. Fortunately, NSO incorporates a number of capabilities that speeds provisioning:

Full-service lifecycle automation

NSO automates the entire end-to-end service provisioning process. It encompasses all network devices and resources, VNFs, applications, and network services—both at the level of coarse-grained service intent and fine-grained run-time configurations. Furthermore, NSO allows provisioning teams to modify running services as well as create and delete them, so they can make changes much more quickly and accurately.

Transactions

As noted earlier, NSO uses a database-style transaction model for provisioning new services or changing existing ones to ensure that the network—and any customer's service—is never left in an unknown state.

Activation testing

You have the tools to build canary tests for new and changed services by sending active traffic over the new (or newly changed) service, measure customer-facing key performance indicators (KPIs), and verify that the service is performing as expected.

Run-time service modifications

NSO can make changes to active services, as opposed to deleting and re-creating a service to implement a change. NSO's state convergence capabilities automatically generate the minimum configuration changes needed to fulfill the modified intent of the service. For example, a customer may log into their self-service portal to change their service level (i.e. bronze to gold) or modify security rules. Once the changes are requested, NSO makes the fine-grained network changes to fulfill the customer's request.

Network change dry-run capabilities

The NSO network change tool shows how a planned change will affect the network and services before executing it. Before committing to a change, teams can perform a dry run and see the minimum set of changes that will occur in the network as the change is executed.

Decoupling of OSS and the network

Because NSO acts as a bridge, the OSS layer is shielded from the intricacies of the network and vice-versa. Teams can manage the lifecycle of their respective areas independently, using a stable interface to allow each layer to communicate with the other. The OSS can be updated without worrying about dependencies with specific networking equipment. Similarly, network infrastructure can be more agile, since new vendors or devices no longer need to be explicitly integrated into the OSS.

Operations: Owning the customer experience

The operations team is often the first line of defense for managing customer experience and they have often lacked the ability to understand what is actually happening. NSO gives operations teams better tools for meeting that challenge.

Traceability

NSO shows operations teams not only what is happening on the network, but why, by letting them examine network services within the context of the associated customer-facing services. A team can trace a single service across devices and see exactly how each configuration (or change) affects each customer's service. This capability makes it much easier to troubleshoot problems for a customer, understand the impact of software version updates, and perform other customer operations and support functions more efficiently.

Deeper insight into service configuration

NSO gives operations teams visibility into how a service is configured and which resources it is allocated to quickly understand the relationship between the service instance and what's actually running in the network. This allows operations to find a mismatch between how a device is behaving, how it is configured, and what the service expects. Operations can identify which service is responsible for a particular part of a device's configuration (for instance, who needs VLAN 99?).

Finally, resource failures can be linked to impacted services (for example, which services were using the link that keeps flapping).

Service planning

For new services is being provisioned with Reactive FastMap (discussed in the next section), operations can use NSO's Plan tool to immediately see how far the provisioning has progressed, with real-time status and configurations from the network. This capability is essential for automating and tracking services composed of both quickly implemented and more time-consuming operations.

Service health

NSO allows networking organizations to incorporate orchestrated assurance into the service model so they can track service KPIs that reflect the real-world customer experience and, if applicable, verify that the service is meeting service-level agreements (SLAs).

Phase 3: Full DevOps infrastructure automation

Ready to start realizing the competitive advantages of uniting network services with agile development methodologies and CI/CD processes? Let's take a closer look at how NSO DevOps capabilities can help.

Model-based architecture

As discussed earlier, one of the defining features of NSO is that it is entirely model-based. NSO captures every aspect of a service in its models. The YANG service model becomes a precise black box specification for a service. By automatically mapping service intent to device configurations, NSO significantly reduces the amount of manual coding required, as any change in the data model automatically triggers real-time re-rendering of the entire system: the UI, the APIs, the data stores, and southbound abstractions. Developers can make design-time changes to service capabilities with unprecedented agility, and without dependence on, or disruption to, the infrastructure teams.

Stateful convergence

Dynamic service creation and modification is possible because NSO works to continually converge the network towards the desired state through two mechanisms:

FastMap

Developers only need describe the "create" operations for a service. FastMap automatically determines the update, delete, and repair operations needed for any type of run-time service modification, saving the developer the time and effort to define workflows for every conceivable service lifecycle scenario.

Reactive FastMap

Ideal for multi-domain and distributed environments, Reactive FastMap takes a non-linear approach to implementing necessary changes needed to reach a desired state. Some changes (i.e., apply a new firewall rule) might take seconds to implement while others (i.e., spin up a new VM) might take minutes. Instead of getting stuck waiting on changes to complete, Reactive FastMap makes changes where it can and continually re-evaluates what still needs to be done

Package management

NSO gives developers a comprehensive, systematic approach to package management with tools to manage applications on top of the platform through the full lifecycle of installing, updating, and uninstalling packages. The platform applies strict versioning rules and allows developers to capture dependencies between packages.

Northbound integration APIs

To support DevOps processes and serve and an effective bridge, NSO provides a stable, flexible software interface:

A rich set of northbound APIs

NSO supports APIs ranging from programmatic or RPC-based protocols (such as NETCONF/RESTCONF) to language bindings like Erlang, Java, Python, and C. NSO also provides human-to-machine interfaces, such as a web UI and a set of CLIs. All of these interfaces are automatically rendered from the models that developers create.

API mediation

A common impediment service provider developers face is existing OSS/BSS systems with hard-coded southbound calls to infrastructure. With conventional orchestration systems, service providers have to undertake an extensive integration project to change how OSS systems parse parameters to the orchestrator. NSO simply adapts to the existing APIs the OSS uses. Developers can create data models with that API, load them into NSO, and map it to the existing service package. The example is SP-specific, but NSO can provide similar API mediation for enterprise systems.

Transaction-safe operations

As mentioned previously, NSO uses a transactional model by default, which means developers don't have to worry about developing and maintaining one for themselves.

Idempotent operations

A core tenet of DevOps, NSO's diff and convergence operations implicitly deliver idempotency.

Transactionality and idempotency are particularly valuable for integrating with service provider OSS/BSS instances, as it means that when those systems call on NSO, they never have to contend with a change being partially executed. Any new service or change transaction is either fully applied or not applied at all. Additionally, idempotency means that event-based systems in the OSS/BSS layer don't have to be coded with logic to avoid sending too much configuration. OSS systems no longer need to gather and maintain state at all—eliminating a huge amount of complexity in the OSS layer. Upper-layer systems become much more adaptable, and simpler and less expensive to integrate.

Multi-vendor abstraction through NEDs

NSO uses a device abstraction layer built around the concept of NEDs that allow NSO to manipulate every device in the network programmatically. The NED computes the ordered sequence of device-specific commands to take the element from its current configuration state to the desired configuration state. This frees developers from both having to write and maintain device-specific code and the logic to handle multi-vendor environments. NEDs are available for practically any network physical or virtual Cisco element, as well as over 150 non-Cisco devices. For further flexibility, NSO 5 introduces a NED Builder tool that allows customers to create their own NEDs for NETCONF devices.

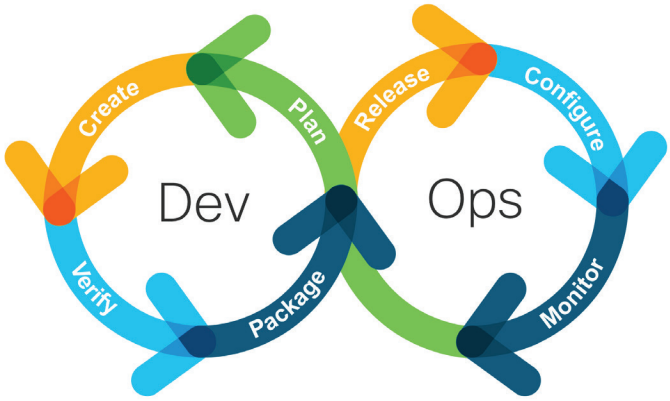
Developer tools and SDK content

NSO includes comprehensive tools to help developers work faster and more efficiently at every stage of the development cycle.

Create

NSO gives developers the ability to run a full production-grade installation of the system in a realistic development environment, so they can get started coding right away. It features a wide set of YANG tools, including YANG validators and compilers. This simplifies work for developers during the "Create" process.

Figure 3 - DevOps cycle



Verify

NSO provides development and production test capabilities through its NetSim tool. This network simulator allows developers to quickly and inexpensively test their code on a realistic simulation of the production environment. NSO also provides offline tools for validating version migration. These tools validate the extent to which clients or consumers of a service (i.e., an orchestrator or a OSS/BSS system) need updates, so developers can avoid introducing unintended disruptive changes.

Package

When developers release new code, NSO provides a self-contained and versioned package format. This means that developers can build and package their work such that the package is the only thing they need to import into the running system. NSO also provides hitless package installation and version migration, so developers can introduce new packages or update existing ones at run time, without impacting the operation of the system.

Configure

NSO can integrate into a CI/CD pipeline so that infrastructure and infrastructure configuration can be seamlessly deployed in concert with the related software packages. Beyond simplifying initial deployment, this capability is helpful with functions like auto-scaling so adding or deleting app instances also automatically includes the associated infrastructure.

Monitor

NSO provides insight to understand how an app or service is interacting with the infrastructure--is there a performance bottleneck or is a service running out of resources. The CDB also provides a single source of truth for performance management, health monitoring, system assurance and similar tools to easily gather operational data on the state of infrastructure.

More information

cisco.com/go/nso

NSO on DevNet

developer.cisco.com/nso

NSO on GitHub

github.com/NSO-developer

Conclusion

NSO makes it easy to bring development tools and methodologies from the software world into the network world. When everyone is speaking the same language and using the same tools, and information flows freely in all directions, the time between coming up with a great idea and getting it into production is radically shortened.

What to do next?

Crossing the bridge will take many steps. Be incremental and be persistent:

- Map the links between technology, people, and processes
- Invest in upgrading the skills of your people
- Take small steps, conduct many pilots, and learn things
- Recognize there will be resistance to change
- Embrace uncertainty
- Let each team move at their own pace
- Keep your customer (internal or external) front and center
- Start with something small but useful: get a win, gain confidence, repeat with something a bit more challenging

By breaking down the walls separating service designers, application developers and infrastructure operations teams, NSO helps organizations accelerate their adoption of DevOps principles and culture with its attendant benefits for both an organization and its customers.