White paper
Public

The bridge to possible

How automation is driving network engineer skills transformation

How is enterprise network automation evolving? What are the implications on people and processes? What are the skills that network engineers will require in the future? Get answers to all these questions and more.

Contents

Introduction

Digital transformation is defined as the application of technology to create new business models, processes, and services. It is being driven by a culmination of technologies and trends such as [1]:

- Remote work making a high-quality user experience essential wherever the users are.
- Boost of Software-as-a-Service (SaaS) applications.
- Wireless WiFi 6 and 5G delivering higher-speed connectivity and better performance.
- IoT enabling people to interact with things to create a wide range of new use cases.

According to IDC, enterprises and governments around the world have made digital transformation a strategic initiative, which has helped drive adoption and implementation of new technologies and related services [2]. Digital capabilities transform the way customers buy and consume products and services [3].

As an enterprise digitalizes, a transformation of the way it is organized and operated is required, with particular emphasis on people and processes.

Enterprise network automation is evolving to support digitalization; the technology shift to software-defined networking (SDN), the proliferation of analytics and insights tools to drive better user experience (complemented with machine learning and artificial intelligence), cloud platform adoption, and the introduction of NetDevOps practices are all driving the network engineer skills transformation.

This white paper reviews the evolution of network automation and identifies the skills requirements introduced at every step, including a 3- to 5-year horizon.

Drivers for network automation

Why are enterprises embracing network automation? Nearly every organization relies on continuous connectivity [4]. Users in the campus, branch, or homeworking – as well as devices – need to communicate with applications for the enterprise business to run. These applications reside in on-prem datacenters, colocations, and public clouds. The network provides the connectivity that enables this communication and is expected to be always available. But it is not just about providing connectivity, it is also about optimizing the experience of the users consuming the business applications.

At the same time, there is a greater need for the network to become more dynamic. Think about the demand from business teams; they want new services, and they want them fast, they demand agility to Information Technology (IT) teams. As a result, IT teams must be ready to reconfigure the network at any time, in many cases these are changes at scale, e.g., they need to be able to instantaneously reconfigure every node of the WAN network when a new application in the cloud is deployed, or when required to respond to emerging threat conditions [4]. These changes need to be expedited, while minimizing the errors during the deployment. The network must remain resilient, reliable, and secure.

The adoption of a network 'digital twin' – a digital representation of the production network – is gaining popularity, as it makes operations more efficient. A key use case is the creation of staging test environments to validate network changes before applying them to production, to increase reliability and resilience. In this context, creating test environments by consuming resources from a shared pool and returning them once testing is completed improves the cost efficiency, as it removes the need to keep resources (e.g., compute, network, and storage) allocated when not in use.

Another trend is the increased rate of changes both in the service supply side (applications) as well as in the service demand side (application consumers). Applications are moving to containers and serverless. Consumers often change location and access network type (Wired Ethernet, WiFi, Private 5G). These changes translate into constant consumer-to-supplier network policy changes and make compliance with the organization's security policies more challenging.

These trends and requirements drive the need for network automation, which takes actions based on:

- Business intent translation into network policies.
- User experience insights.
- Network insights mapping to user experience insights.
- Business transaction health mapping to application and connectivity performance.

Network automation evolution

In 1983, the U.S. Advanced Research Projects Agency Network (ARPANET) took the decision to migrate to TCP/IP Internet protocol suite [5]. In 1986, Kirk Lougheed left Stanford University to join Cisco as the company's first engineer, he developed the first version of Cisco Internetworking Operating System (IOS) and began building and testing routers [6]. This was the time when history of networking, as we know it today, started.

Initially, the network relied on manual provisioning and configuration via Command-Line Interface (CLI). This was device-by-device (per box) configuration executed by network engineers using Telnet protocol to connect to devices, and with configuration single source of truth residing in network devices themselves.

During many years, networks were provisioned and operated in this way, IT teams delivered very slow rate of configuration changes, but still enough to satisfy the demand of business teams. It was a rudimentary network operation style where IT teams manually translated business intent into network device configurations, and where Information Technologies Operations Management (ITOM) and Configuration Management Data Base (CMDB) systems were not connected to the network.

What about monitoring? Together with the emergence of Internet Protocol (IP) based networking, monitoring protocols and tools were also introduced. The Simple Network Management Protocol (SNMP) was created as a standard management protocol for devices in IP networks [7], then tools such as MRTG - that builds and displays graphs using data collected with the SNMP protocol - became the first tools to monitor networks [8]. The network engineer was responsible to correlate, analyze, and interpret network events to recognize problem situations, and take actions to remediate them.

Key challenges with this approach to deploy and manage networks were:

- Error prone, manual configuration and analysis.
- Network configuration drift was often introduced, resulting in deployment inconsistencies.
- Low rate of configuration changes.
- Network knowledge relied on a few individuals.
- No configuration was removed from routers, switches, and firewalls, even though they were not used anymore, to avoid the risk of inadvertently blocking some service.

The first scripts

Already in the early days of networking, engineers started to adopt basic automation – using scripts - with focus on very frequent recurring tasks, e.g., basic device configuration automation and service provisioning. These scripts were relatively simple software programs written with languages such as Bash or Expect. Still today, scripting remains a key skill for network engineers to have.

Automating the software upgrade of a network device, deploying an IP Access Control List (ACL) to a group of devices, or collecting network data are examples of use cases for scripting-based automation.

In such legacy networks – which still exist today – scripts interact with the network using CLI or SNMP. Network engineers initiate script execution manually or schedule them in Linux Cron or similar utilities.

Key challenges with this approach to manage networks are:

- Very limited benefits of automation, which focuses only on highly recurring tasks.
- Limited or no-collaboration between IT teams working on different network domains (campus, WAN, datacenter). Each team maintains and executes their own scripts.
- Device-by-device (per box) configuration.
- Most scripts interact with device CLI interfaces, which are designed for human interaction and not for machine-to-machine communication and automation.

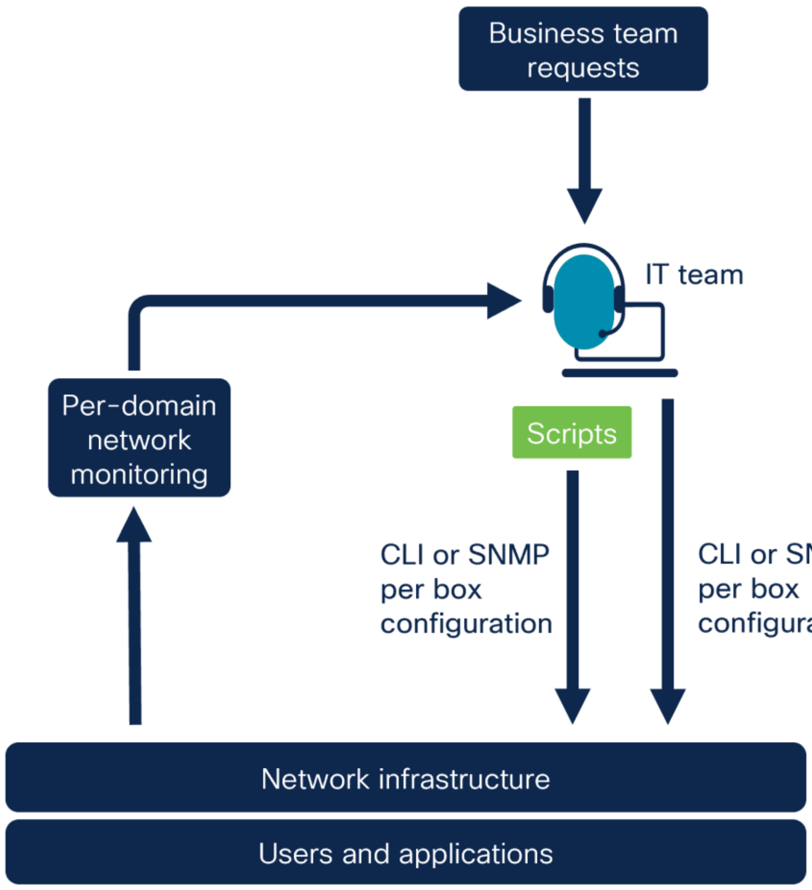

Business team requests
IT team
Per-domain network monitoring
Scripts
CLI or SNMP per box configuration
CLI or SNMP per box configuration
Network infrastructure
Users and applications

Figure 1: Introduction of scripts to automate network configuration tasks

In the early days of networking, the first off-the-shelf network configuration tools were also released, for IT teams to simplify network management. Cisco Prime (Cisco Works prior to 2011) is a good example of such a tool, still being used today by many organizations.

Network controllers and Software-Defined Networking (SDN)

As the demand for network services and their scalability increased, the network architectures themselves started to expose their own limitations: hierarchy of different network domains, higher over-subscription ratios, lack of network topology abstraction (no underlay/overlay separation), or control plane scalability, among other.

To overcome these limitations, networking vendors started the transition towards Software-Defined Networking (SDN) architectures for datacenter, campus, and WAN. As examples of this trend:

- Cisco acquired Meraki in 2012 [9].
- VMware acquired Nicira (NSX) in 2012 [10].
- Juniper Networks acquired Contrail Systems in 2012 [11].
- Alcatel-Lucent venture Nuage Networks launched SDN solution in 2013 [12].
- Cisco acquired Insieme (Application Centric Infrastructure) in 2013 [13].
- Cisco acquired Tail-f Systems in 2014 [14].
- Cisco acquired Viptela (SD-WAN) in 2017 [15].
- Cisco released DNA Center and SD-Access in 2017 [16].

In SDN architectures, per-domain network controllers are introduced, they act as an abstraction layer and point of management for IT teams to provision, configure, and monitor the network.

In many cases, organizations integrate ITOM and CMDB with network controllers, facilitating the design, plan, delivery, operation, and control of information technology services offered to users.

In SDN architectures, the network is treated as a 'fabric', a virtualized, automated lattice of overlay connections on top of the physical network topology, with each device inheriting a specific role and function within the fabric, being managed by a network controller. A network 'fabric' delivers the following benefits to the operator [17]:

- Abstracts the underlaying physical topology to create virtualized planes that simplify deployment and operations.
- Applies configurations fabric-wide rather than device-by-device, which increases the consistency and facilitates large scale deployments.
- Applies services and enforces policies uniformly across wired and wireless users and devices.
- Lowers business risk by limiting traffic propagation, containing threats, and boosting security.
- Creates standardized building blocks such that the network can be easily scaled.

In SDN architectures, network controllers play a key role in the translation of business intent into configurations (Intent-Based Networking), the design, the policy definition, and automation of the network deployment and change. Intent-Based Networking (IBN) transforms a device-centric manually operated network into a controller-led network that captures business intent and translates it into policies that can be automated and applied consistently across the network. The goal is for the network to continuously monitor and adjust performance to help assure desired business outcomes. The network controller acts as a central control point for network activity. The adoption of network controllers and SDN architectures in all network domains (campus, WAN, datacenter, and cloud) extends their benefits throughout the enterprise and helps make digital transformation a reality [18].

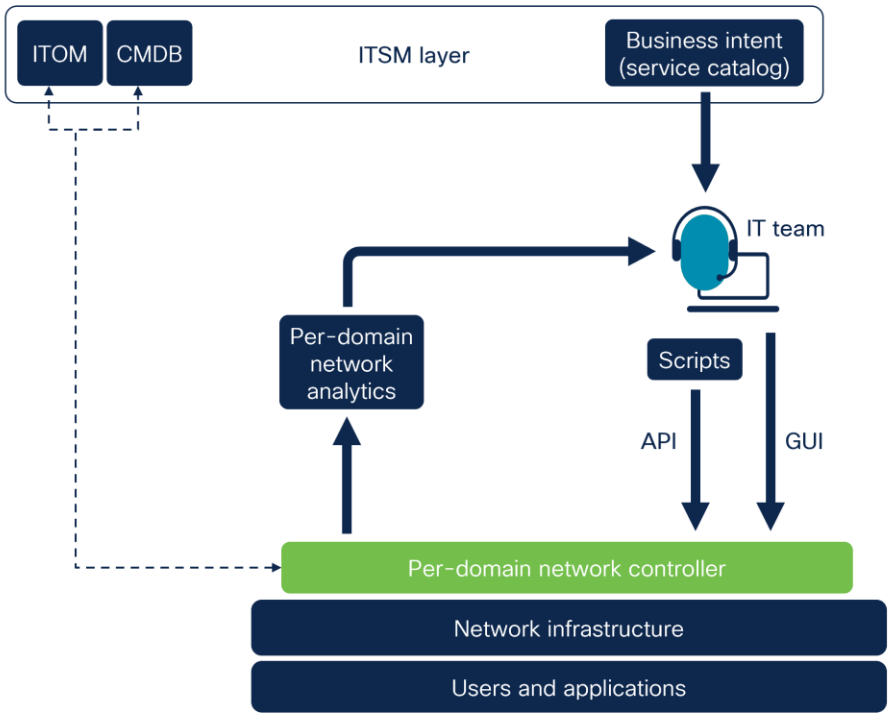

ITOM
CMDB
ITSM layer
Business intent (service catalog)
IT team
Per-domain network analytics
Scripts
API
GUI
Per-domain network controller
Network infrastructure
Users and applications

Figure 2: Per-domain network controllers

Benefits of network controllers and SDN architectures can be summarized as:

- Faster response to new business requirements.
- Higher reliability.
- Reduced operational expenses.
- Consistent and scalable network changes.
- Reduced risk and continuous compliance.
- Improved rate of change through automation.
- Facilitated network programmability.

SDN architectures are being adopted by many organizations. At the time of writing this document – September 2022 - this move from traditional networking towards controller-based networks is still ongoing and is expected to continue over the next years. To support this transition, SDN architectures and network controller knowledge are key skills for network engineers to have.

Network controllers, REST APIs, and programmability

Network controllers are a key enabler of programmability, thanks to their APIs (Application Programming Interfaces) and their SDKs (Software Development Kits) which facilitate the integration of the network controllers with IT systems, exposing their data and functionality through service interfaces enabling automation.

Network engineers leverage network controller APIs to interact with the network and request tasks to be completed (e.g., shutdown an interface), fetch pieces of information (e.g., get the status of an interface), or store a piece of information (e.g., set the description of an interface). In this way, the interaction with the network is automated. In most cases, RESTful APIs are used.

A RESTful API is an architectural style for an API that uses HTTP request methods to access and use data. That data can be accessed via GET, PUT, PATCH, POST, and DELETE methods, which refers to the reading, updating, creating, and deleting programmable resources [19].

Undoubtedly, knowledge of **REST APIs** and **programmability** are key skills for network engineers to acquire.

Network programmability is a generic term, which has different meanings to different people. In this document we are referring to network programmability as the use of software tools to deploy, manage, and troubleshoot network elements [20].

Scripts are used in network programmability. As these scripts started to become more complex and interact with the network using APIs, the use of **Python** has become popular; nowadays many network engineers make use of it. Python is not the only language being used for network automation, but the combination of being an easy to learn language, and availability of many code samples and utilities, has made it a go-to language for network engineers [21]. Python has a built-in standard library that provides complete support for data encoding and decoding, network protocols, and all other networking concepts. Writing the code for network programming in Python is easier than in C++ or Java. Python programming language provides two levels of network service access: low-level and high-level access. The low-level access offers the basic socket support of the operating system, while the high-level access allows the implementing protocols like FTP, HTTP, etc. [22].

Broadened scope of networks

The scope of 'network' has broadened over the years. While in the first days of networking it referred to routers and switches only, extending to firewalls and load-balancers later, today the term encompasses networks in virtualized datacenter environments, VXLAN overlays, public cloud networking, or container service mesh. We could say that the network is the collection of physical and virtual systems that provide connectivity between users, devices, and applications.

This is raising the need for network engineers to become **cross-domain experts**, since IT teams need to manage end-to-end connectivity. Today we still observe many organizations managing network connectivity in silos, in particular the cloud and container environments, but there is a trend towards unifying responsibilities under a single team that manages end-to-end connectivity. The number of network engineers with **public cloud and Docker / Kubernetes** certifications is rapidly increasing, which facilitates this operational transformation.

Automating network security

Enterprises need to add consistent end-to-end security policies to network deployments. Using CLIs or GUIs is not a way to rapidly deploy security policies that have been validated, are consistent, and cover end-to-end traffic flows [23]. Furthermore, administrator errors (fat finger errors) can leave the configuration in an unsecured state.

In traditional network domains, a subset of security features is deployed and managed by network controllers. For example, a campus network controller will typically provide the capability to implement user access control, segmentation, and micro-segmentation policies.

In the case of dedicated appliance-based security systems such as firewalls or IDS/IPS, traditional management methods are no longer sufficient and there is a shift towards leveraging device or controller APIs to provide consistent, scalable methods, to automate and orchestrate configuration changes.

Cross-domain analytics and user experience

Together with the evolution of automation, there is a shift from network monitoring towards analytics and insights. New tools are being introduced to analyze network trends and patterns, to provide IT teams deeper understanding of network behavior and the impact on the applications and services the network enables. This equips IT teams to make data-driven decisions, anticipate and predict failures, as well as accelerate remediation.

In parallel to this, full-stack observability tools are being introduced to the market. They provide users' application experience – as well application performance – insights. They move beyond domain monitoring into full-stack visibility, insights, and actions across the technology stack, so IT operations teams can deliver and optimize the user experience [24]. They deliver real-time observability across the modern technology stack – applications, software defined compute, storage, services, network, and more, i.e., the 'full stack' – to provide enterprises with in-depth visibility into the behavior, performance and health of their applications and supporting infrastructure via high fidelity telemetry (metrics, events, logs and traces) collected from their entire IT estate [25]. This allows to create a common context of the full-stack services provided and supported by the different IT teams, allowing them to take actions prioritizing user experience and business performance.

These trends will lead network engineers to leverage **data gathering, visualization** (i.e., Telegraf, Grafana, Prometheus, etc.), and **full-stack observability tools**.

In the current context of distributed, microservices based applications and workload and workforce mobility, network and user analytics and insights need to be deployed cross-domain. This is where Artificial Intelligence (AI) and Machine Learning (ML) tools are introduced. They collect metadata from multiple network controllers and user experience observability tools to provide cross-domain insights that enable data-driven decisions relevant to the end-to-end service. As AI/ML becomes more relevant in modern networking, we expect to see network engineers that broaden their technology skills, learning the **basics of AI/ML** and the networking related use cases, and working closely together with AI/ML teams.

Infrastructure as Code (IaC)

Another emerging trend in the last years is Infrastructure as Code (IaC), following a common cloud deployment model where cloud infrastructure and services are maintained using tools and processes commonly used by software developers (version control system, peer review, automated testing, deployment pipelines, etc.).

Enterprise IT organizations start to treat network configurations in a similar fashion as code, storing them in structured data files (e.g., YAML) in version control systems such as GIT. An IaC configuration inventory represents the desired state of the network configuration, then IaC tools are leveraged to make certain that the network current state aligns with the desired state. IaC configuration inventory files content is the single source of truth for the desired state of the network.

The predominant tools in IaC are **Ansible** and **Terraform**, but also commercial tools such as Cisco NSO [26]. Networking vendors are developing Ansible collections (a distribution format for Ansible content that can include playbooks, roles, modules, and plugins [27]) and Terraform providers to enable IT organizations with the evolution of network automation towards the next level.

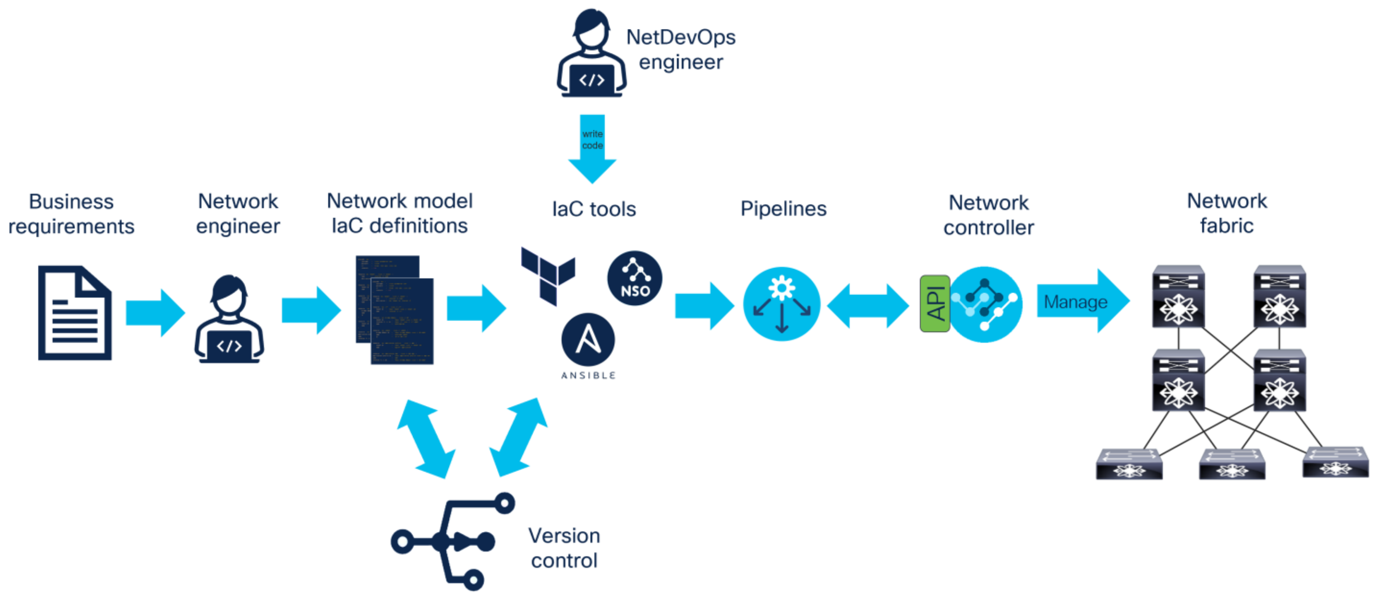
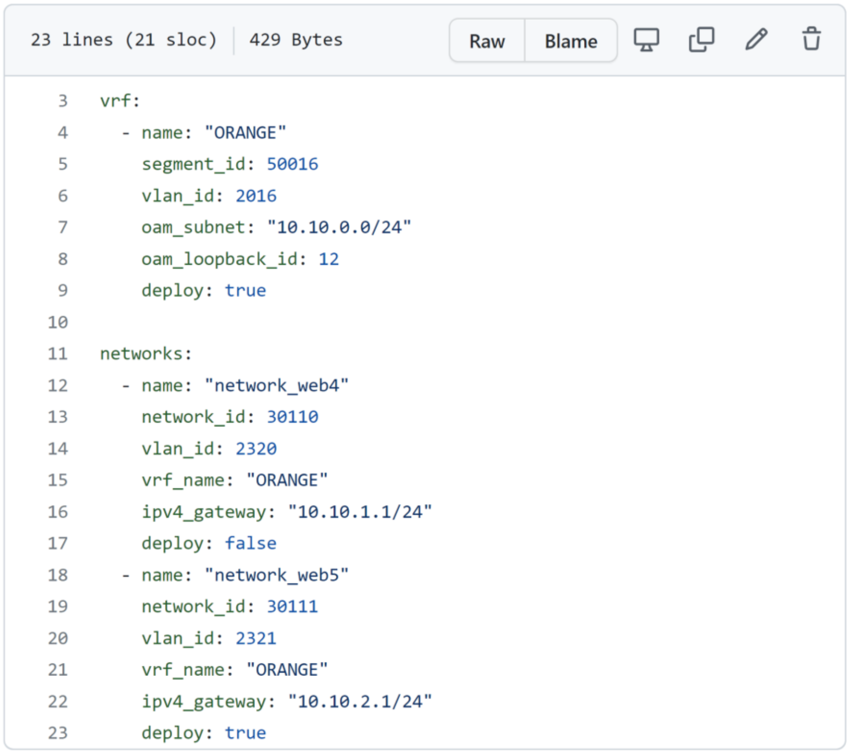


Figure 3: Infrastructure as Code process

There are generally two ways to approach IaC: declarative (functional) vs imperative (procedural). The difference between the declarative and the imperative approach is essentially defining 'what' versus 'how' [28]. A declarative approach keeps a list of the current state of the system objects, which simplifies the management of the infrastructure. An imperative approach instead defines the specific commands needed to achieve the desired configuration, and those commands then need to be executed in the correct order [29]. With an imperative approach, error conditions must be managed explicitly.



```
vrf:
  - name: "ORANGE"
    segment_id: 50016
    vlan_id: 2016
    oam_subnet: "10.10.0.0/24"
    oam_loopback_id: 12
    deploy: true

networks:
  - name: "network_web4"
    network_id: 30110
    vlan_id: 2320
    vrf_name: "ORANGE"
    ipv4_gateway: "10.10.1.1/24"
    deploy: false
  - name: "network_web5"
    network_id: 30111
    vlan_id: 2321
    vrf_name: "ORANGE"
    ipv4_gateway: "10.10.2.1/24"
    deploy: true
```

Figure 4: IaC declarative configuration definition file example

There is currently a trend towards the adoption of IaC declarative approach, where the automation tools own the responsibility of the system consistency.

When choosing the software to be used in IaC environments, the 'network rate of change' is often a key decision driver. Today there are tools that deliver very fast mapping of IaC based configuration inventory (network-model) to underlaying network configurations. This agility cannot be achieved manually or with workflow orchestrator engines.

Policy-as-Code

Once an organization starts to manage the network infrastructure with a declarative IaC approach, they will realize that they need to include policy and procedure enforcement as part of their automation framework. A policy can be thought of as a set of rules. As such, every organization is going to have a number of policies in place, and even an organization without formal policies in place will still need to comply with regulations, agreements, and laws. A policy might describe things like which devices on a network should be considered trusted, or what roles are required to perform different actions in a device [30].

In this context, Policy-as-Code (PaC) is a natural next step in the evolution of the network automation governance. Declarative policy models are very useful tools to this end, an example would be the adoption of Open Policy Agent (OPA) to describe unit tests for Terraform plans, which makes it possible to evaluate the Terraform code against rules and policies defined by the administrator. OPA makes it possible to write policies that test the changes Terraform is about to make before they are actually executed [31]. Such tools ease and foster collaboration between NetDevOps and SecDevOps teams.

NetDevOps and CI/CD pipelines

The IaC approach often goes hand in hand with the adoption of tools and DevOps practices that application developers have been using for years. This approach is commonly referred to as NetDevOps.

In NetDevOps, Continuous Integration (CI) is the process of automating and integrating network IaC changes and updates from many team members during network operations. In CI, automated tools confirm that IaC definitions are valid and error-free and will not introduce errors when applied to a test/validation environment. This helps detecting errors and enables network changes at an unprecedented rate.

Continuous Delivery (CD) is the ability to push network changes into production multiple times a day (or hour or minute). CI/CD is part of NetDevOps, it helps shorten the network change process.

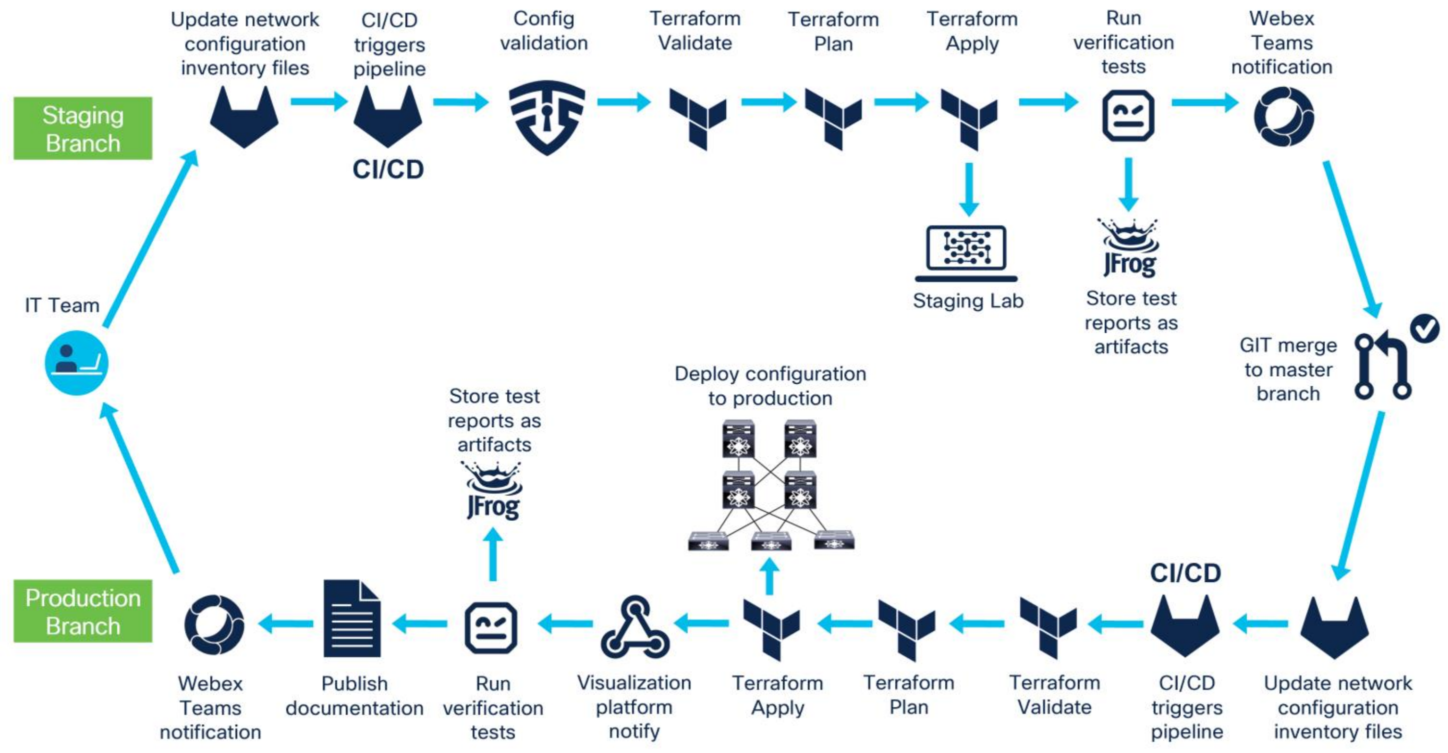


Figure 5: Industrialized network configuration with CI/CD pipeline

A CI/CD pipeline is a series of steps which are executed in a specific order to deploy network changes. These steps typically include IaC configuration inventory syntax validation and security checks using tools like Yamillint and Yamale, testing the network changes in a lab with ROBOT framework-based test automation tools, ChatOps to notify operations teams, deployment of changes in production network, verification of changes in production network, and other steps. Common tools used to build pipelines are Jenkins, Drone, or Github Actions, among others.

Drivers for organizations to adopt IaC and NetDevOps are:

- Highly reliable provisioning and change processes thanks to automated pre and post deployment validations and verifications.
- Network configuration consistency.
- Enablement of changes at scale.
- Reduction of human errors through four-eyes principles/review process.
- Elimination of network device configuration drift.
- Cost savings, thanks to the adoption of open-source tools and automation.
- Agility, faster provisioning of services, and higher rate of change.
- Historical log of network changes (facilitating both troubleshooting and compliance audit).
- Highly skilled architects and engineers can focus on tasks more relevant to the business.

While the tools facilitate the work, the IT team remains responsible to take network change decisions and initiate them by making updates to the IaC configuration inventory file(s).

Where do IaC and CI/CI pipelines fit into the framework that we are using throughout this white paper to represent the evolution of network automation? This is described in Figure 6.

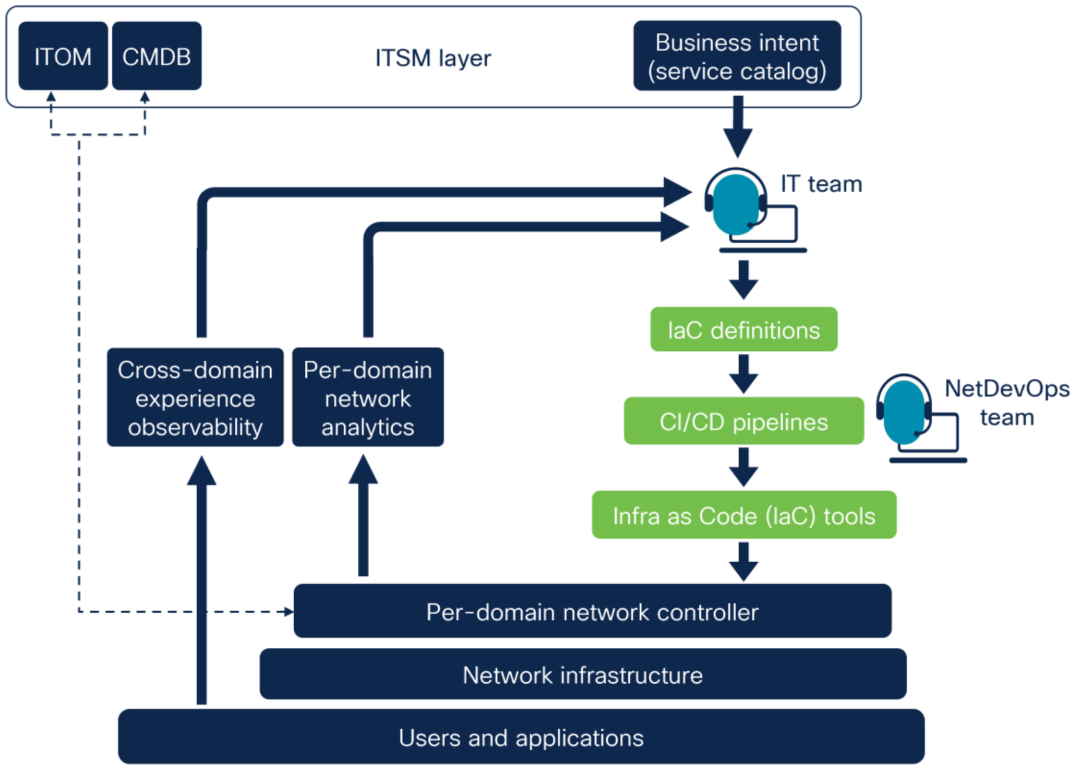

ITOM
CMDB
ITSM layer
Business intent (service catalog)
IT team
IaC definitions
CI/CD pipelines
NetDevOps team
Infra as Code (IaC) tools
Cross-domain experience observability
Per-domain network analytics
Per-domain network controller
Network infrastructure
Users and applications

Figure 6: IaC and CI/CD pipelines in automation framework

IaC and NetDevOps trends are leading many network engineers to learn **human-readable data-serialization languages** (e.g., YAML, JSON), **Ansible**, **Terraform**, as well as **CI/CD tools** and **NetDevOps** practices.

With the move towards NetDevOps, we also start to observe enterprise organizations foster greater collaboration across teams within IT, with shared script repositories, and shared strategy in terms of tools in use.

Digital twin

Testing to validate network and security policy changes in staging environment is a key element of CI/CD pipelines. A network 'digital twin' facilitates the change validation processes, by providing a software representation of the production network infrastructure.

Automation is used to replicate the production network topology and state – as well as network security posture – in a staging environment, where physical components are replaced by a digital representation. Then, network changes are simulated within the 'digital twin' and results are validated. If results are satisfactory, the changes are applied to production network. Finally, the state of the production network is also validated.

Network data model

In the context of automation with IaC, defining a network and network-service data model is a recommended first step. A data model is an abstract model that organizes elements of data and standardizes how they relate to one another, and to the properties/attributes of real-world entities. For instance, a data model may specify that the data element representing a network is composed of a number of VRF elements that represent the name, description, routing settings, and interfaces assigned. YANG is commonly used as data modeling language [32].

Developing knowledge around **network data modeling** and **definition of schemas** that specify the expected structure of YAML files - input value types (String, Enum, IP, etc) and additional constraints (e.g., value ranges, regular expressions, etc) - are valuable skills for network teams to have.

Workflow orchestration

While network controllers and IaC are – in most cases – deployed per network domain, it is common that organizations adopt network cross-domain workflow orchestration tools that link business intent to end-to-end network changes. An orchestration workflow will typically trigger multiple CI/CD pipelines in different network domains (datacenter, WAN, campus, cloud) to deploy/change/remove the different network components that provide connectivity to a business service.

For specific business use cases, a service catalog portal triggers orchestrator workflows that update IaC configuration inventory. IT teams also provide manual inputs to workflow execution based on user experience and network insights.

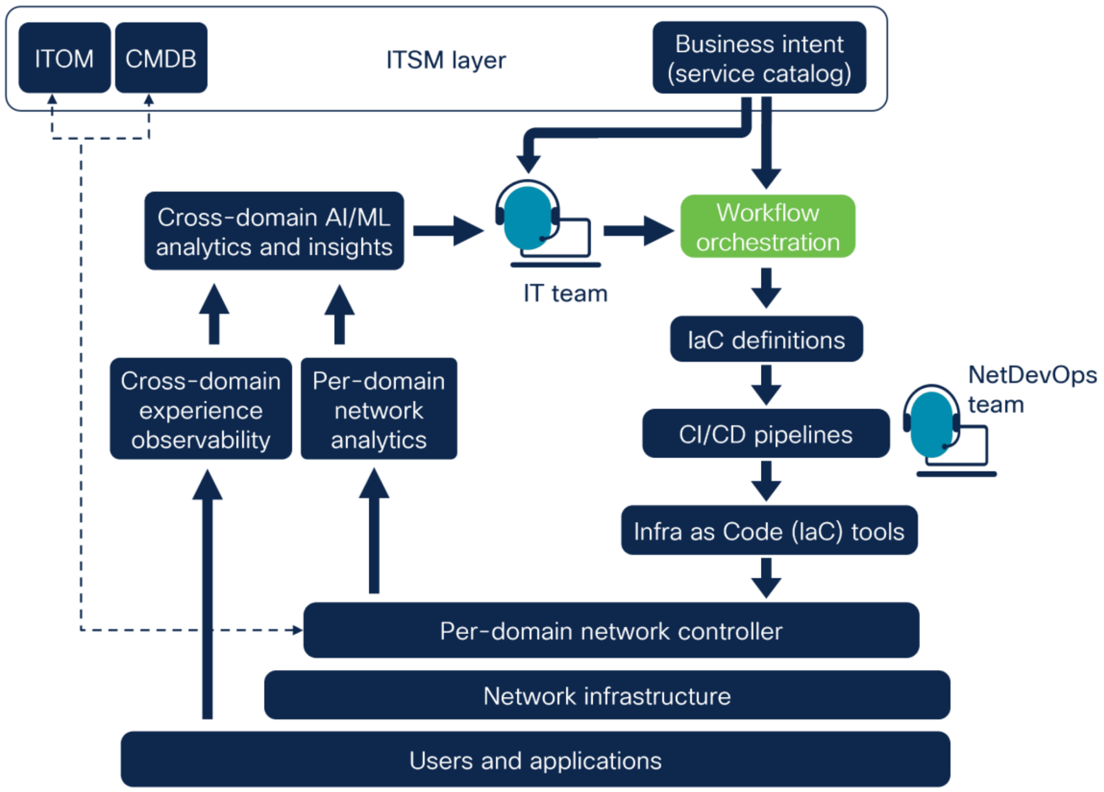

ITOM
CMDB
ITSM layer
Business intent (service catalog)
Cross-domain AI/ML analytics and insights
IT team
Workflow orchestration
IaC definitions
NetDevOps team
CI/CD pipelines
Infra as Code (IaC) tools
Cross-domain experience observability
Per-domain network analytics
Per-domain network controller
Network infrastructure
Users and applications

Figure 7: Cross-domain workflow orchestration

IT teams may either initiate the provisioning through the workflow orchestrator or initiate changes by making updates to the network IaC configuration inventory.

In this context, **workflow orchestration** is another key skill for network teams.

Closed-loop automation

The ultimate goal of automation is closed-loop systems with automated provisioning of self-organized, self-diagnosing, and dynamically updated intent-based network fabrics, where business intent and cross-domain insights drive network changes, and IT team becomes an observer and supervisor while NetDevOps team maintains pipelines.

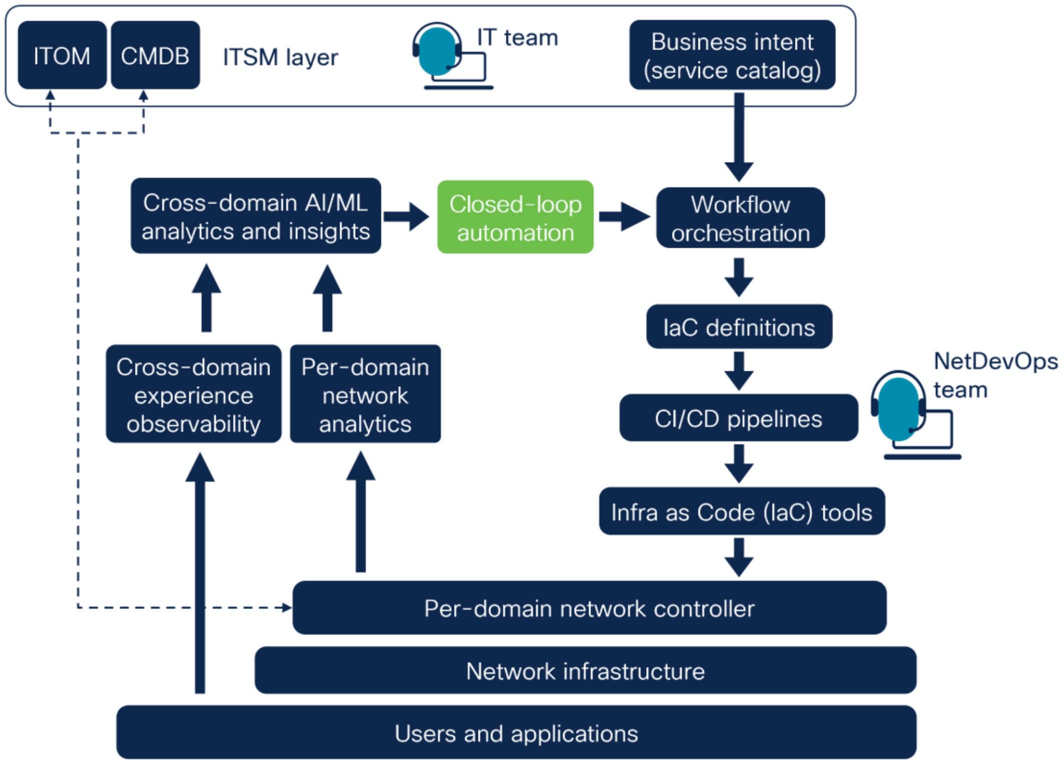

ITOM
CMDB
ITSM layer
IT team
Business intent (service catalog)
Cross-domain AI/ML analytics and insights
Closed-loop automation
Workflow orchestration
IaC definitions
NetDevOps team
CI/CD pipelines
Infra as Code (IaC) tools
Cross-domain experience observability
Per-domain network analytics
Per-domain network controller
Network infrastructure
Users and applications

Figure 8: Closed-loop automation

Note that in typical operational environments, network performance data is abundant, but many toolsets simply do not have a comprehensive and coalesced understanding of the network control state and routing protocol event data. When data is not organized, analyzed, and displayed logically, it becomes difficult to comprehend. This is where an integrated closed-loop system that automates real-time feedback is required. Experience has shown that an integrated solution with full multivendor network programmability and workflow orchestration is key to achieving measurable benefits, including improved capital efficiency, better-operating expense utilization, and reduction in time to deploy a service [33].

Artificial Intelligence for IT Operations (AIOps) is a term coined by Gartner in 2016 as an industry category for machine learning analytics technology that enhances IT operations analytics. Such operation tasks include automation, performance monitoring, and event correlations, among others. The goal is to enable IT transformation, receiving continuous network insights, and providing continuous fixes and improvements via automation [34]. Adoption of AIOps drives network automation towards closed-loop and predictive operations.

Benefits of closed-loop automation are:

- Augmented network reliability through AIOps.
- Automated fault recovery, reducing downtime and business impact of network outages.

- Incident resolution time is reduced, increasing customer experience.
- Reduction of operational cost by reduction of required manual tasks.
- Increases the benefits of IaC and CI/CD.

AIOps and closed-loop automation requires network teams to develop or hire AI/ML (pattern detection, anomaly detection, machine learning) skills, to fine tune the systems.

Protecting automation code

In the first steps of the automation journey, basic scripts are developed. As the team develops network automation scripts, security should always be top of mind. Applying best practices to version control systems, secure storage of the credentials used by scripts (data at rest), and encrypting data communication to network devices (data in transit) are required to comply with enterprise security policies.

IT team must protect the version control system by disabling public repository creation, granting network access only to specific IP addresses, using Multi Factor Authentication (MFA), keeping regular control of the users allowed to access, and never storing credentials in the repository, among other measures.

Best practices to store credentials used by code must also be adopted, to prevent that they are retrieved in the event a system that runs the automation code is compromised. There are solutions such as HashiCorp's Vault that helps you secure, store, and tightly control access to tokens, passwords, certificates, encryption keys for protecting secrets, and other sensitive data using a UI, CLI, or HTTP API [35].

Communication from systems running the scripts to network devices should use a secure channel, like TLS/SSL. Communicating with a TLS certificate protects all access credentials and API data in transit using end-to-end encryption. API keys are another step toward securing a REST API [36].

DevOps and DevSecOps practices in network automation code development

As the enterprise progresses in the network automation journey, DevOps and DevSecOps will play an increasingly important role. Network automation code (e.g., scripts, Ansible modules, other) should be treated as any other software and, therefore, its development needs to follow software development DevOps practices. While earlier in this document we have discussed about CI/CD pipelines in the context of network operations, the network automation code development process will also leverage CI/CD pipelines. Think about network automation with Ansible; automation use cases are first defined, then Ansible modules are developed or customized to implement the use cases. These Ansible modules need to be tested in a lab environment, then deployed to production to be used in the network automation process. The behavior of the modules in production is then monitored to provide continuous feedback to the team developing them.

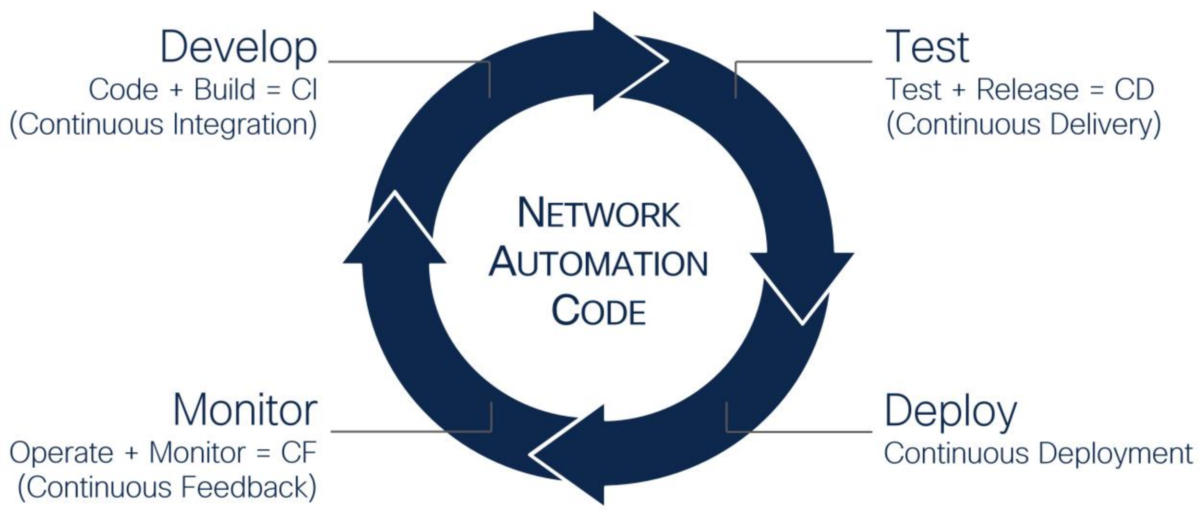


Figure 9: Network automation code development cycle

What about the security of the network automation code? Adding security to the automation code development DevOps pipeline is described as ‘shift left security practice’. The idea is to catch security vulnerabilities earlier in the network automation code lifecycle. This will reduce operational risk downstream in the production environment and the potential for - usually far greater - associated remediation costs and by extension, reputational costs, regulatory fines, etc.

DevSecOps complements DevOps practices and focuses on security. Capabilities like ‘Static Application Security Testing (SAST)’ ensure that security weaknesses such as those listed in the ‘OWASP Top Ten’ [37] and the ‘2022 CWE Top 25 Most Dangerous Software Weaknesses’ [38] are detected in the proprietary code being developed.

Another key capability that is required for effective DevSecOps practice is the feedback loop. Both application observability telemetry as well as automation code runtime data should be leveraged to inform developers of unusual code execution behaviours or sub-performant code.

Network engineer skills of the future

In previous sections we have reviewed the evolution trends in network automation and identified skills that network teams require today or will require in the future.

As organizations define their automation strategies, it is important to understand the transformation of the network engineer's role, as well as the technical skills it will require [39]. We have summarized our view in Table 1.

It is crucial for organizations to enable network teams to confidently rebalance resources, reduce risks, and accelerate innovation [40]. There is also an opportunity for network engineers’ personal satisfaction, evolving from being perceived as the bottleneck and slowing down innovation, to becoming the enabler of IT’s success, and to better service the lines of business (not to mention the greater appeal of their resume).

Network engineer's work evolution	Network engineer's skills of the future
▪ Work on cross-domain projects ▪ Architect SDN based networks ▪ Programming ▪ Create network change prioritization rules ▪ Create CI/CD pipelines ▪ Define automation test cases ▪ Adjust automation based on operational feedback/problems found ▪ Use open-source data gathering and visualization tools for troubleshooting ▪ Apply Site Reliability Engineer (SRE) disciplines to operation [42] ▪ Develop AI/ML close-automation systems ▪ Observe and supervise automation systems ▪ Engineer solutions for sustainability	▪ Public cloud networking ▪ Docker, Kubernetes ▪ Scripting, programmability, Python ▪ Software code protection and integrity ▪ SDN / overlay network architectures ▪ Interacting with APIs / REST ▪ Data modeling ▪ Human-readable data-serialization language (e.g., YAML) ▪ Ansible, Terraform ▪ CI/CD tools, NetDevOps practices ▪ Data gathering, massaging and visualization tools (e.g., Telegraf, Grafana, etc.) ▪ Full stack observability tools ▪ AI/ML (Pattern detection, anomaly detection, machine learning)

Table 1: Network engineer work evolution and skills of the future

Traditional network engineers might be overwhelmed, especially during the transition from provisioning and manually managing static on-premises resources to providing a readily available, dynamic, and highly automated network infrastructure. With the list in Table 1, we hope to help clarifying the road ahead and address the challenges head-on.

The list is extensive, and **not every network engineer will require each of the skills listed** in the table or will be involved in all of the work activities. However, it is unquestionable that the scope of the network engineer is expanding and will continue doing so, therefore, **every network engineer must be open to developing new skills, become more cross-domain knowledgeable, get familiar with new tools, and learn to connect the applications with the network and with the performance of the business**.

As network engineers accelerate their journey to develop their skills, the skills development journey will become increasingly digitalized. The use of digitized learning platforms such as Coursera, Udemy, or Pluralsight, is rapidly growing.

To contribute to network engineer learning, Cisco has created the DevNet community as an extensive repository of information, tutorials, labs, and sandboxes, to learn and practice with the technologies and methodologies mentioned in this paper. The DevNet certification program validates the skills of software developers, DevOps engineers, automation specialists, and other software professionals. The program validates key emerging technical skills for a new kind of IT professional, empowering organizations to embrace the potential of applications, automation, and infrastructure for the network, Internet of Things (IoT) and DevOps [41].

Summary

Enterprise network automation is evolving to support digitalization; the technology shift to software-defined networking (SDN), the proliferation of analytics and insights tools to drive better user experience, cloud platform adoption, and the introduction of NetDevOps practices are all driving the network engineer skills transformation.

This white paper reviews the evolution of network automation, from the introduction of scripts to closed-loop automation systems, and identifies the skills requirements introduced at every step.

We have listed the skills that network teams require today – and in the future – and why every network engineer must be open to developing new skills, become more cross-domain knowledgeable, get familiar with new tools, and learn to connect the applications with the network and with the performance of the business.

Authors

Asier Arlegui, Principal Architect, Cisco Customer Experience

Greg Page, Enterprise Strategist, Cisco Sales

Juan José Fonseca, Solutions Architect, Cisco Customer Experience

Reviewers

Koenraad Bastiaens, VP, Cisco Customer Experience

Oliver Boehmer, Principal Architect, Cisco Customer Experience

Luca Relandini, Principal Architect, Cisco Sales

Héctor Fernández, Technical Solutions Architect, Cisco Sales

Rafael Muller, Principal Engineer, Cisco Customer Experience

Mike Reshetar, Principal Architect, Cisco Customer Experience

Citations and Bibliography

[1] Cisco, *Executive perspectives,* https://www.cisco.com/c/en/us/solutions/executive-perspectives/digital-transformation.html.

[2] IDC, *How HCI Is Powering Hybrid Cloud Optimization to Accelerate,* https://www.cisco.com/c/dam/en/us/products/collateral/hyperconverged-infrastructure/idc-hybrid-cloud-wp.pdf.

[3] J. W. a. T. E. Lah, *Digital Hesitation: Why B2B Companies Aren't Reaching Their Full Digital Transformation Potential,* TSIA, 2022.

[4] S. Harrell, *Cisco Digital Network Architecture (Foreword chapter),* 2019.

[5] Wikipedia, *ARPANET,* https://en.wikipedia.org/wiki/ARPANET.

[6] Cisco, *Newsroom,* https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2020/m11/employee-no-4-talks-cisco-history-and-life-as-the-company-s-first-engineer.html.

[7] IETF, https://datatracker.ietf.org/doc/html/rfc1157.

[8] T. Oetiker, *MRTG,* https://oss.oetiker.ch/mrtg/.

[9] Cisco Press Release, *Cisco Completes Acquisition of Meraki,* https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2012/m12/cisco-completes-acquisition-of-meraki.html, 2012.

[10] VMware, VMware to Acquire Nicira, https://news.vmware.com/releases/vmw-nicira-07-23-12, 2012.

[11] ZDNET, Juniper buys enterprise SDN firm Contrail for $176m, https://www.zdnet.com/home-and-office/networking/juniper-buys-enterprise-sdn-firm-contrail-for-176m/.

[12] D. C. Dynamics, Alcatel-Lucent venture Nuage launches SDN platform, https://www.datacenterdynamics.com/en/news/alcatel-lucent-venture-nuage-launches-sdn-platform/.

[13] Cisco Press Release, *Cisco Completes Acquisition of Insieme Networks,* https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2013/m12/cisco-completes-acquisition-of-insieme-networks.html, 2013.

[14] C. P. R. Cisco Completes Acquisition of Tail-f Systems, https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2014/m07/cisco-completes-acquisition-of-tail-f-systems.html.

[15] Cisco Press Release, *Cisco Completes Acquisition of Viptela,* https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2017/m08/cisco-completes-acquisition-of-viptela.html, 2017.

[16] Cisco Press Release, *Cisco unveils network of the future that can learn, adapt and evolve,* https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2017/m06/cisco-unveils-network-of-the-future-that-can-learn-adapt-and-evolve.html, 2017.

[17] Cisco, What Is a Network Fabric?, https://www.cisco.com/c/en/us/solutions/enterprise-networks/what-is-a-network-fabric.html.

[18] Cisco, *What is intent-based networking (IBN)?,* https://www.cisco.com/c/en/us/solutions/intent-based-networking.html.

[19] Techtarget, *REST API (RESTful API),* https://www.techtarget.com/searchapparchitecture/definition/RESTful-API.

[20] Networkcomputing, Network Programmability Basics, https://www.networkcomputing.com/networking/network-programmability-basics, 2016.

[21] Cisco DevNet, *Python Network Automation,* https://developer.cisco.com/site/python/.

[22] Linuxhint, Python Networking, https://linuxhint.com/python-networking/.

[23] Cisco, Implementing Automation for Cisco Security Solutions (SAUI) v1.0.

[24] Cisco, *Full-Stack Observability from Cisco,* https://www.cisco.com/c/en/us/solutions/full-stack-observability.html.

[25] Cisco AppDynamics, *What Is Full-Stack Observability?,* https://www.appdynamics.com/topics/what-is-full-stack-observability.

[26] Cisco, *Network Services Orchestrator,* https://www.cisco.com/c/en/us/products/cloud-systems-management/network-services-orchestrator/index.html.

[27] Ansible, Using Collections, https://docs.ansible.com/ansible/latest/user_guide/collections_using.html.

[28] Wikipedia, Infrastructure as Code, https://en.wikipedia.org/wiki/Infrastructure_as_code.

[29] RedHat, *What is Infrastructure as Code (IaC)?,* https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac.

[30] A. Eknert, What is Open Policy Agent?, https://www.styra.com/blog/what-is-open-policy-agent/.

[31] Open Policy Agent Terraform, https://www.openpolicyagent.org/docs/latest/terraform.

[32] Wikipedia, *Data_model,* https://en.wikipedia.org/wiki/Data_model.

[33] Tail-f, An integrated closed loop solution enables a fully automated network, https://www.tail-f.com/an-integrated-closed-loop-solution-enables-a-fully-automated-network/.

[34] Wikipedia, Artificial Intelligence for IT Operations, https://en.wikipedia.org/wiki/Artificial_Intelligence_for_IT_Operations.

[35] HashiCorp, Manage Secrets & Protect Sensitive Data, https://www.vaultproject.io/.

[36] Okta, Securing REST APIs, https://developer.okta.com/blog/2019/09/04/securing-rest-apis.

[37] OWASP, OWASP Top Ten, https://owasp.org/www-project-top-ten/.

[38] CWE, 2022 CWE Top 25 Most Dangerous Software Weaknesses, https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html.

[39] I. Pinto and A. Arlegui, Network Automation Trends and Strategy, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/network-automation-strategy-wp.html.

[40] C. Rittler, How Automation Can Reveal IT Blind Spots and Unleash Innovation, https://blogs.cisco.com/customerexperience/how-automation-can-reveal-it-blind-spots-and-unleash-innovation.

[41] Cisco, DevNet Training and Certification Program, https://www.cisco.com/c/en/us/training-events/training-certifications/certifications/devnet.html.

[42] Google, What is Site Reliability Engineering (SRE)?, https://sre.google/.