7 Habits Of Highly Effective DC Networkers

A PACKET PUSHERS WHITEPAPER

Table of Contents

Introduction

To be a data center network engineer is to live with complexity and uncertainty. A change in one place could have an unintended effect somewhere else. Performance could suffer, workloads could be exposed, or the network could crash.

To operate successfully in uncertain environments where network requirements constantly evolve, network engineers need to cultivate good habits. Experienced network engineers tend to be careful, thoughtful, and precise. Traditionally, before making a change, they seek out information about the network through SNMP traps, syslogs, device configurations, packet captures, and streaming telemetry. And when something goes wrong, they have a process in place to find, diagnose, and fix the problem.

While those general practices can serve engineers well, this paper proposes 7 specific habits, listed below, that can help network engineers better grapple with the uncertainty in data center management and operations.

These habits can stand on their own as a set of best practices, but they also align with Juniper Apstra, a software platform built on the notion of Intent-Based Networking (IBN). IBN provides reliable automation and orchestration in a data center network. The Apstra software, which works in a multi-vendor environment, automates, orchestrates, and validates changes to ensure that the outcome matches the intent of the business.

The 7 habits are:

1. Design for business outcomes
2. Automate for reliability and repeatability
3. Learn once, use often
4. Pick the right equipment
5. Validate as you go
6. Be proactive, not reactive
7. Document your work

This whitepaper explores these 7 habits of effective data center network operators and examines how Apstra compliments those habits.

The 7 Habits

1. Design For Business Outcomes

The reason a data center network exists is to support the applications and services that drive a business. However, a typical data center design tends to start with vendor selection rather than desired outcomes. That selection is often influenced by factors other than business objectives. An executive who's enjoyed many fine lunches with a sales rep may want one product, while the engineering staff who've invested time and money in vendor certifications might want another.

There may be small but essential differences among products. One vendor's code might be weighed down with bells and whistles that the organization doesn't want or need, but still has to maintain and update. One vendor's hardware specs might be ideal, but its network OS could be buggy. How the software implements a key protocol might require extra effort by the engineers to make it work.

The result is the demands of actual business applications have to be bent to fit within the quirks and constraints of the product. This means more operational complexity and a greater risk of problems. It slows down the pace at which the network can support new applications and services, so the network becomes a bottleneck.

A better habit is to start with business outcomes, and then design the network around those outcomes. By focusing on intent, Apstra prioritizes outcomes. It then operates at the device level to declaratively implement those outcomes. It continuously monitors and validates network state to ensure those outcomes are delivered.

And because Apstra supports a wide range of network hardware and software, organizations can choose what's best for the business, rather than having a design be dictated by a vendor's quirks.

2. Automate For Reliability And Repeatability

The networking industry has been talking about automation for decades, but most enterprise shops have yet to move past home-grown scripts that engineers use like a multi-function pocket-knife: a few handy tools for small jobs.

The truth is, most enterprise data centers are too brittle to support broad, reliable automation that's orchestrated across multiple devices and services. One reason is that an automated process could kick off a sequence of unintended events that brings down the entire network And further, after setup, it's not unusual for configurations to drift away from a baseline. Engineers who write scripts and playbooks have to constantly adjust their code to account for the nuances of the different software versions running on network devices.

Network engineers are also missing key elements to support broad-based automation. That includes deep visibility into network state, a reliable source of truth of the expected device configuration, the ability to test changes before being pushed into production, and a mechanism for validating the outcome of a process.

Apstra enables reliable and repeatable automation because it has guardrails in place to prevent unintended problems and to ensure that changes align with the operator's intent. For example, if an engineer makes a configuration change that doesn't align with intent, whether it be a typo in a command or a configuration change that would violate an existing policy, Apstra will notify the engineer and prevent the change.

Apstra can do this because it holds the entire network state in its graph database (described below), allowing it to identify potential problems before they're pushed into production. In this way, Apstra streamlines change management where common changes can be made in a predictable and validated way, safeguarding against unintended consequences.

And, if a problem does arise with a change, Apstra also includes the ability to roll back devices to a known good state. The result is automation processes that are both repeatable and reliable.

3. Learn Once, Use Often

Anyone who occasionally uses a business application knows that each time you come back to it to complete a task, you inevitably have to waste time re-familiarizing yourself with how it works.

The same applies to management, monitoring, and automation tools. If they aren't a part of your regular workflow, your task gets delayed as you fumble around the interface. It's a good habit to know your tools well so that you can get the most value from them.

Apstra is straightforward to learn and use in a few days' time. This compares favorably to other solutions that represent such an architectural and operational sea change that they take months to learn. And, because it offers a multiplicity of functions including monitoring, configuration, and analytics, network engineers can quickly get up to speed on the software and then return to it regularly, gaining mastery of a useful tool.

At the same time, because Apstra can operate in a multi-vendor environment, it abstracts away the individual quirks of each NOS and vendor protocol implementation. This means network engineers can get their work done without having to be intimately familiar with the nuances of each NOS.

4. Pick The Right Equipment

A smart habit is to choose the right equipment for the job at hand, rather than have to conform operations to fit how a particular piece of equipment works.

Apstra's multi-vendor strategy gives network teams the flexibility to get the right hardware and software to support the company's objectives. This also means that companies can get purchasing leverage, and avoid lock-in. They can also source equipment from different suppliers, which may be necessary given current supply chain constraints.

5. Validate As You Go

Tech social media is full of anecdotes about hasty CLI commands, misconfigurations, and typos that resulted in all sorts of exciting and challenging outcomes. The fact is, humans make mistakes, which is why successful engineers check their work as you operate in production.

Those checks can take a variety of forms. Some network OSs check each word in a command string as it's typed to alert engineers to a typo. Engineers might ask a colleague to sanity-check a configuration, or upload scripts to a repo for others to review. Some organizations rely on ITIL-based change management processes.

Apstra enforces this good habit by continuously validating device changes to make sure those changes align with intent. For example, if you create a new VLAN and accidentally add it to an interface with an existing untagged VLAN, Apstra not only returns an error but also shows you why.

Continuous validation limits human mistakes while also ensuring that new changes produce the desired intent; that is, the new changes fit within business-defined parameters regarding reachability, performance, security, and compliance.

6. Be Proactive, Not Reactive

It's a good habit to deal with small problems before they become big ones. For instance, if a switch port goes down intermittently, it's better to find the root cause and address it before the port fails outright.

If you wait, you're going to be swarmed with alerts and logs (and perhaps angry text messages) while you try to figure out what went wrong. If it's a loose cable, you got lucky. If it's a bad network card or optical module, you might wish you'd kept spares on hand.

Apstra supports proactive operations through its analytics capabilities. It monitors physical device state and can warn of anomalies that may indicate bigger problems to come if not addressed right away. It also monitors bandwidth issues, utilization, and overall capacity, and will signal engineers ahead of time so that they can meet growing demands smoothly and efficiently rather than with their hair on fire.

7. Document Your Work

Like eating your vegetables and exercising regularly, everyone knows that documenting your work--be it updating network diagrams, annotating configuration changes, and so on--is a good habit. Documentation explains what and why something was done. It provides a record of what happened. This record can come in handy later on for new changes, troubleshooting, an audit, etc.

Documentation is important because engineers move into different roles or take a job elsewhere. When they go, they take a lot of operational and institutional knowledge with them. Documentation captures this knowledge so that it can be retained by the organization and shared with operators.

The problem is that documenting your work is tedious and time-consuming. It's often the last thing on an engineer's mind when grappling with a tricky new deployment or responding to a crisis. Of all the habits on this list, regular and reliable documentation might be the hardest to maintain.

Apstra can support this habit because it is designed to be essentially self-documenting. It constantly collects telemetry from individual devices and maintains overall network state. It stores changes and version histories, and can act as a kind of time machine that lets you peer into the past to see what change or update was made, and what the outcome was.

And as mentioned above, Apstra can roll back the network to a known good state if necessary.

A Brief Overview Of Apstra And IBN

Apstra is built around the concept of Intent-Based Networking (IBN). Apstra starts with business-level intentions, or expected outcomes, and then translates those intentions into the device-level configurations necessary to meet those outcomes.

Business intent covers general outcomes, such as ensuring a certain service level for a set of applications. It also covers specific outcomes, such as deploying a fabric, turning up a VLAN and attaching the correct ports, or enforcing access policies.

Apstra continually monitors the overall state of the network as well as individual network devices. This ensures that the network always meets the organization's intent. If an event occurs that violates that intent, such as packet loss, congestion, or interface problems, Apstra can alert an engineer and provide relevant details about the issue. In some cases, Apstra can provide suggested next steps or automatically remediate the problem if desired.

If an engineer tries to make a configuration change that would clash with previous intent, Apstra will alert the engineer by providing pre-change analysis to avoid unintended consequences. As new

network services are added to the data center, Apstra ensures that the network configurations required to support those services don't conflict with existing intent.

Core Principles

To enable IBN, Apstra is built on key principles including reference designs, multi-vendor support, continuous validation and integration.

Reference Designs: A significant reason for data center complexity is that data center networks are often put together without any long-term strategy. Custom, specialized implementations are put in place to address an immediate business requirement. Over time, the result is a web of complex configurations that are impervious to automation and require hand-crafted, time-consuming care and maintenance. There's often little to no documentation for such a network--save the institutional lore passed on from one engineer to the next. Accumulated technical debt may force engineers to rely on quick fixes just to keep the business running.

Apstra starts with a clean slate via the use of a limited number of reference designs. Reference designs describe a template for the physical infrastructure, including switching hardware and cabling. A typical reference design is a leaf-spine or Clos network.

While some engineers might feel constrained by a reference design or template, this approach is guided by industry best practice and dramatically reduces the problems that arise from one-off, stopgap, and snowflake implementations that bedevil a typical data center network. It is this level of discipline and rigor that must be followed to ensure there is a single source of truth and control over changes.

By using a reference blueprint to guide automation across the full design-deploy-operate lifecycle, Apstra is enabling configuration as code, implementing version control, and providing verifications of dependencies. This approach gives enterprises many of the proven automation techniques of cloud providers, without the need for a deep bench of in-house experts.

Multi-Vendor Support: Apstra supports a growing number of hardware vendors, including Cisco, Artista, Dell, and Juniper, as well as whitebox hardware makers. Apstra also supports a variety of network OSs, from commercial NOSs to open-source options such as SONiC.

This multi-vendor support ensures that organizations get the right devices to meet their business and technical needs, and that these devices align with their engineers' knowledge and skill sets. It also provides greater choice and flexibility for customers, who may want to mix and match vendors and NOSs across racks or pods to get competitive pricing, consider supply chain diversity, or to meet operational requirements.

Closed-Loop Validation: Apstra ensures intent through closed-loop validation based on its single source of truth. Closed-loop validation makes sure that a change or update actually occurred, and that the change or update aligns with intent. Contrast this with a simple automation script, which can automate a set of commands, but the script simply runs and then stops; it has no context as to whether a change was made or if the change resulted in the desired outcome.

Closed-loop validation is essential for reliable automation because it checks that the automated process actually produced an outcome in line with business requirements, and that the network continues to operate in a desired state. This validation provides the reliability that engenders a network team's trust in an automated system.

Integration: Apstra tightly integrates with VMware NSX-T and VMware vSphere to streamline operations between the server and networking teams. Apstra not only delivers end-to-end visibility across both the physical underlay and the virtual overlay, it also identifies when changes in the overlay require adjustment in the underlay, such as the fabric's configured MTU size not matching requirements for the overlay. Apstra also detects and remediates anomalies and misconfigurations.

In this way, the underlay is responsive and optimized to the changing dynamics of workload connectivity and volume. Further, when issues arise, the Apstra software can quickly identify whether the root cause lies in the underlay or is a problem in the virtual overlay. By accelerating the time it takes to address issues, Apstra helps organizations significantly reduce mean time to repair and operational costs.

Additionally, open APIs support integration to common workflow tools such as ServiceNow, chatbots, and Slack.

Apstra Components

The Apstra product consists of three software elements: device agents, a data store, and a graph database. These elements work together to deliver intent-based networking. We'll briefly review each one.

Agents: Apstra uses device agents on network devices, physical and virtual, to configure the devices and send telemetry data to Apstra's data store. For network devices that can't run a third-party agent, an off-box agent can be run as a Linux container in the Apstra server. This off-box agent can get state information from the network device via SSH or APIs.

Data Store: The data store runs on a server and collects agent telemetry, network design details, network anomalies, and other data. User intent is also held in the data store, as is the graph database.

Graph Database: Apstra represents every element or object in the data center network within the graph database, as well as all network configurations. This is how Apstra “understands” the network state. The graph database serves as a virtual model of the network and a single source of truth for different teams. This model is kept up to date in real time because it is continuously being fed network and device telemetry. User intent is compared against this model to ensure that network and device configurations will deliver the appropriate outcomes.