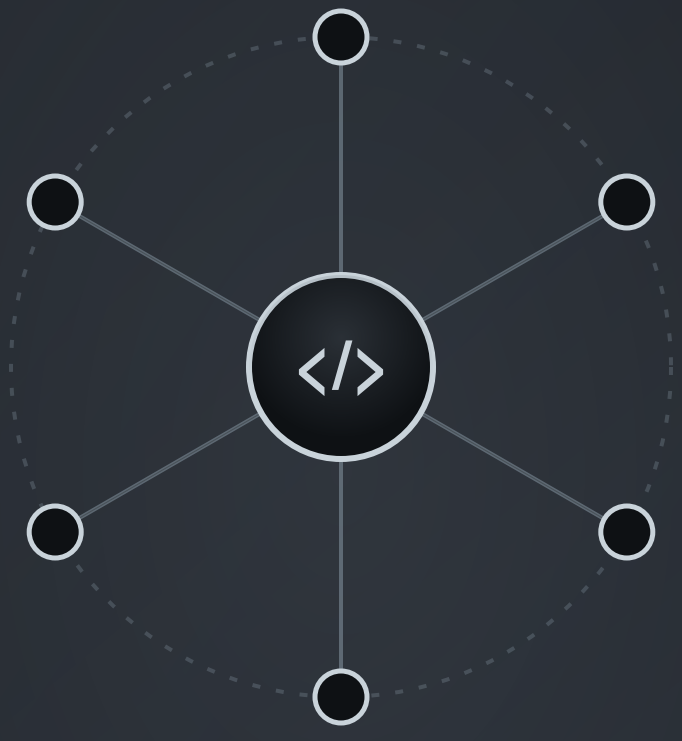

SD-WAN Fundamentals

A Practitioner’s Field Guide

JOZEF BAROŠ

SD-WAN
FUNDAMENTALS

A Practical Guide to Designing, Building, and Operating Cisco Catalyst SD-WAN (Viptela)

An implementation companion — from the first design meeting to steady-state operations

JOZEF BAROŠ
ProDigitalJ

SD-WAN Fundamentals

A Practical Guide to Cisco Catalyst SD-WAN (Viptela)

First edition.

Trademarks and independence

Cisco, Cisco Systems, Catalyst, IOS XE, Meraki, and Viptela, together with their respective product names, are trademarks or registered trademarks of Cisco Systems, Inc. and/or its affiliates. Microsoft 365 and Azure are trademarks of Microsoft Corporation. AWS is a trademark of Amazon.com, Inc. All other trademarks are the property of their respective owners.

This book is an independent publication and is not affiliated with, authorised by, sponsored by, or otherwise endorsed by Cisco Systems, Inc. It is an educational work written to help engineers understand and implement SD-WAN concepts on the Cisco Catalyst SD-WAN platform.

Disclaimer

The information in this book is provided “as is,” without warranty of any kind. The company “Meridian,” its sites, addresses, and configurations are fictional and used purely for illustration; any resemblance to a real organisation is coincidental. All configuration snippets, IP addresses (drawn from documentation ranges such as those reserved in RFC 5737), device identifiers, and command output are illustrative and simplified for teaching. Command syntax, feature names, licensing, and platform behaviour change between software releases — always verify against current Cisco documentation and test in a lab before applying anything to a production network. The author accepts no liability for any loss or damage arising from the use of this material.

Contents

Chapter 1 — Understanding SD-WAN

Welcome to your first SD-WAN project. Over the course of this book we will take a single, realistic enterprise from its aging MPLS network all the way to a fully operational Cisco Catalyst SD-WAN fabric — designing it, building it, migrating live sites onto it, layering on the policies that make it worth the effort, securing it, and finally operating and troubleshooting it day to day. You will finish with the mental models, the configuration patterns, and the hard-won "watch out for this" knowledge to run a project like it yourself.

Before anyone touches a controller, though, you need a solid grasp of *what SD-WAN is and why it works the way it does*. This first chapter builds that foundation: the problem SD-WAN solves, the handful of ideas at its core, and a tour of the Cisco Catalyst SD-WAN components you will spend the rest of the book configuring. Skip it and every later design decision will feel arbitrary. Read it and those decisions will feel obvious.

Meet the project: Meridian

Throughout the book we will follow one company. **Meridian** is a mid-size retailer and logistics operator: one headquarters with an attached data center, around **40 branch sites** spread across several countries, and a growing dependence on cloud applications — Microsoft 365 for the office staff, a SaaS warehouse-management system, and workloads starting to move into a public cloud. Today Meridian runs a traditional dual-MPLS WAN, and it hurts: cloud apps feel slow, the second MPLS circuit costs a fortune to sit mostly idle, opening a new store takes months, and the WAN team spends its days hand-editing router configs.

Meridian's goals will drive every chapter: cut WAN cost by blending broadband with MPLS, give branches fast direct access to Microsoft 365, keep voice quality solid, isolate the guest Wi-Fi and payment systems from each other, and make opening a new store a same-week affair. Whenever a concept could feel abstract, we will ground it in a concrete Meridian decision.

> 🌍 **From the Field — Your real project will rhyme with Meridian's**
>
> The specifics change — bank, hospital, manufacturer, government — but the shape repeats: too much MPLS cost, cloud traffic taking the scenic route, slow site turn-ups, and a team drowning in per-box configuration. If you recognize your own environment in Meridian's, the design patterns in this book will map almost directly onto it.

1.1 The Problem You Are Being Asked to Solve

How the traditional WAN was built

To understand why Meridian hired you, you have to understand the network you are replacing. For most of the last two decades the standard enterprise WAN followed one recipe. Each branch connected to a carrier-provided **MPLS** circuit — a private, SLA-backed transport bought from a service provider — and all traffic between sites flowed across that MPLS cloud, usually through one or two central data centers in a **hub-and-spoke** pattern. Branch routers ran a routing protocol such as **BGP** or **OSPF** toward the provider, and every device was configured by hand, one CLI session at a time.

MPLS earned its place. It delivered predictable latency and jitter, carrier-backed guarantees, and clean separation between customers. In an era when the important applications lived in the corporate data center and the Internet was an afterthought, funnelling everything through the DC was the *right* design. The trouble is that era ended, and the network didn't.

The five pressures that broke the model

No single event killed the traditional WAN. Five pressures piled up until the network became the bottleneck of the whole business. You will see all five in Meridian.

1. **Cloud and SaaS inverted traffic patterns.** Applications moved out of the data center and into Microsoft 365, Salesforce, and public cloud. Yet the WAN still hauls every branch user's cloud traffic back to the data center for centralized security before sending it out to the Internet — the dreaded *hairpin* or *tromboning*. For Meridian's store staff opening Microsoft 365, that means a slow round-trip through headquarters for traffic that was Internet-bound all along.
2. **Bandwidth demand outran MPLS economics.** Video, collaboration, and cloud backups multiplied bandwidth needs, but MPLS costs many times more per megabit than ordinary broadband. Enterprises ended up paying premium telco prices to move Netflix-era volumes.
3. **Active/standby wasted half the spend.** The classic design used MPLS as primary and a second circuit as idle backup. A link you pay for every month but use only during a failure is money sitting on a shelf — exactly Meridian's complaint about its second MPLS circuit.
4. **Per-box configuration didn't scale.** Every router was configured individually. A single policy change across 40 branches meant 40 change windows, 40 chances for a typo, and configuration drift as a permanent state of nature.
5. **Site turn-up took months.** Ordering an MPLS circuit, staging a router, shipping it, and dispatching an engineer turned "open a new store" into a quarter-long project. The business had learned to move faster than its own network.

The daily reality you are replacing: the same change, re-applied by hand, per device

```
! Store 17 of 40 — typed into one more SSH session
class-map match-any VOICE
 match dscp ef
policy-map WAN-EDGE
 class VOICE
  priority percent 20
!
```

```
interface GigabitEthernet0/0/0
 service-policy output WAN-EDGE
!
! ...now do the same on 39 more routers, flawlessly, this weekend.
```

Notice what is wrong here beyond the tedium. There is no single source of truth for "what should the voice policy be everywhere?", no automatic check that the change actually took effect, and no safe rollback tied to whether the site is still reachable. Scripting the CLI with Python or Ansible speeds up the *typing*, but it does not give you *network state* — which is the real problem SD-WAN sets out to fix.

> ⚠️ **Common Problem — "We already script our configs, so we're basically doing SD-WAN"**
>
> This is one of the most common misconceptions you will hear from an incumbent team. Config-push automation delivers text to devices; it maintains no central model of intended state, offers no reachability-aware rollback, and gives no unified view of whether policy is actually in force. SD-WAN's controller model addresses the *state* problem, not merely the *typing* problem. When you hear this objection, that distinction is your answer.

The quiet costs: visibility and security

Two subtler problems complete the picture, and both will shape your design later. First, **visibility**: with intelligence scattered across dozens of independent boxes, answering "why was voice bad in the Brno store at 10:14 this morning?" means logging into devices one by one and correlating by hand. Second, **security posture**: the moment branches gain local Internet links (for backup, or for direct cloud access), each one becomes a potential exposure point — but deploying and managing a full security stack at every small store is operationally out of reach in the old model.

> 💡 **Did you know? — MPLS is not the villain — and it isn't going away**
>
> SD-WAN does not require you to rip out MPLS. In most real migrations, Meridian's included, MPLS survives as *one transport among several*, carrying latency-sensitive traffic like voice while broadband handles bulk and cloud. What SD-WAN removes is MPLS's monopoly — and with it, the carrier's pricing leverage over you.

> **Chapter Recap**
>
> The traditional WAN paired carrier MPLS with per-device, CLI-driven operations — a design tuned for data-center-centric applications. Cloud and SaaS inverted traffic patterns, bandwidth demand collided with MPLS economics, active/standby designs idled paid capacity, box-by-box configuration made every change slow and risky, and site turn-ups took months. Add fragmented visibility and an unmanageable branch security posture, and you have the exact pain Meridian — and your real customer — is paying you to fix. Every SD-WAN feature in this book exists to answer one of these problems; keep the list in mind and the technology will always have a reason.

1.2 The Ideas That Make SD-WAN Work

SD-WAN can look like a pile of new acronyms until you see that it rests on just three ideas. Get these three and everything else in the book is detail hung on the frame: **separate the planes**, **build an overlay so transport stops mattering**, and **drive it all from centralized policy.**

Idea 1: Separate the planes

Software-Defined Networking (SDN) — the parent idea behind SD-WAN — makes one architectural move: split the *control plane* (the logic that decides where traffic should go) from the *data plane* (the hardware that actually forwards packets), and centralize that control logic so the network can be programmed as one system instead of administered as a heap of boxes.

SD-WAN applies this to the wide-area network. Instead of every branch router independently working out its own view of the world, a small set of central **controllers** learns the whole topology, computes reachability, applies company-wide policy, and hands the results to every edge router. The edges keep the job they are good at — forwarding packets fast — while the thinking moves to software you can scale, upgrade, and reason about in one place.

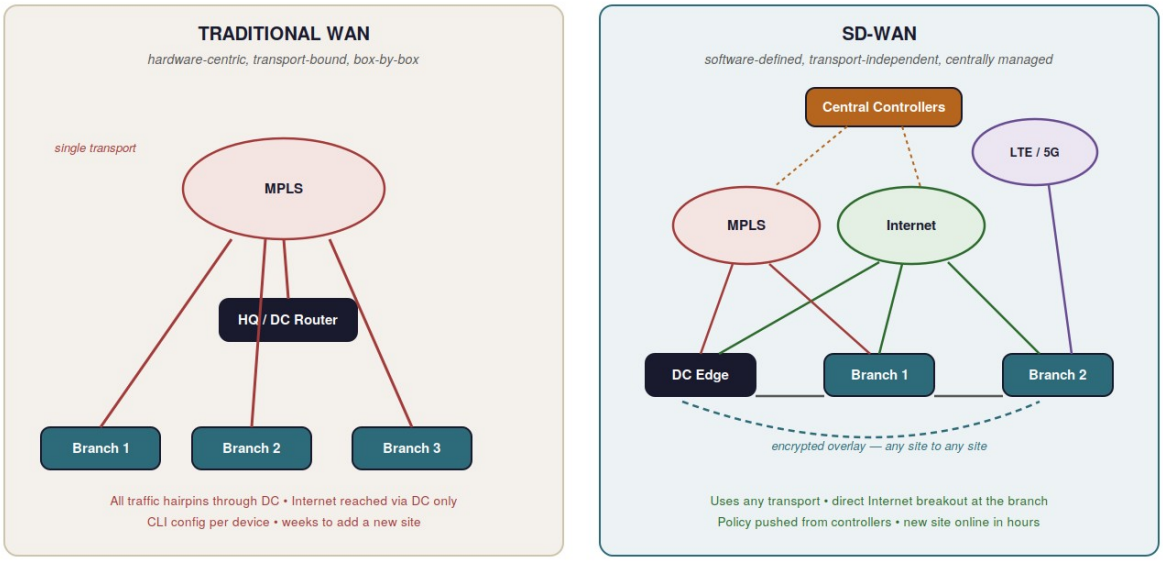

Figure 1.1 — Meridian today (left) versus after migration (right): from a transport-bound hub-and-spoke MPLS network to a transport-independent, centrally controlled overlay that uses every circuit at once.

Idea 2: The overlay — make transport stop mattering

The second idea is the **overlay network**, and it is the one that saves Meridian money. SD-WAN edge routers build encrypted **IPsec** tunnels to one another *over whatever transports happen to exist* — MPLS, broadband Internet, LTE/5G, satellite. Those physical circuits become the **underlay**: interchangeable plumbing whose only job is to move packets between tunnel endpoints. The network your users actually experience — the **overlay** — is the mesh of encrypted tunnels riding on top.

- **Underlay** — the physical transports (MPLS, Internet, cellular) providing raw IP reachability between sites. The plumbing.

- **Overlay** — the encrypted tunnel fabric built on top, where routing, segmentation, and policy actually live. The network you design and operate.
- **Transport independence** — the property that the overlay behaves the same no matter which underlays carry it, so you can mix and swap circuits freely.

This is what unlocks the economics. Because the overlay treats every transport as just another path, Meridian can keep one modest MPLS circuit for voice, add cheap broadband for everything else, and use **both at the same time** — steering each application onto the path that currently serves it best. The idle-backup problem simply disappears: every link you pay for carries production traffic.

> 💡 **Did you know? — You have already used an overlay**
>
> If you have ever run a personal VPN over hotel Wi-Fi, you used an overlay: the tunnel behaved the same whether the underlay was fiber, DSL, or a phone hotspot. SD-WAN industrializes that exact idea — thousands of tunnels, several underlays in use at once, and automatic per-application path selection on top.

Idea 3: Centralized, intent-based policy

The third idea is where SD-WAN earns its keep operationally. In the old world, a business rule like "voice gets priority everywhere; guest Wi-Fi never touches the data center" had to be hand-translated into configuration on every router. In SD-WAN you express that **intent once, centrally**, and the controllers compile it and push it to exactly the sites it applies to. Change the intent, and the whole fabric converges on the new rule — no per-box editing, no drift.

Combined with the overlay, centralized policy is what makes the headline features possible: **application-aware routing** that steers Microsoft 365 onto the healthiest path in real time, **direct Internet access** from the branch under policy control, and **segmentation** that keeps guest, payment, and corporate traffic in separate virtual networks end to end. Each gets its own chapter later. For now, the point is that none of them would be possible if control and policy hadn't been centralized first.

> 🌍 **From the Field — What actually changes on day two**
>
> Teams that finish an SD-WAN migration consistently say the biggest change is *operational rhythm*. A QoS or topology change that used to eat a weekend of change windows becomes a single policy edit — reviewed in the morning, activated after lunch, with the previous version one click away as rollback. That shift, more than any datasheet feature, is what people mean when they say SD-WAN changed how they work.

> ⚠️ **Common Problem — Transport independence is not free bandwidth**
>
> A tempting misread of "overlay" is that it conjures capacity out of thin air. It does not. An IPsec tunnel over a congested 20 Mbps DSL line is still a congested 20 Mbps path — SD-WAN can *detect* the congestion and steer around it, but it cannot create bandwidth. Do not let the elegance of the overlay talk you out of honest capacity planning for each site.

> **Chapter Recap**
>
> SD-WAN stands on three ideas. **Separate the planes**: centralize the control logic, leave fast forwarding at the edges. **Build an overlay**: encrypted tunnels over any transport make MPLS, broadband, and

cellular interchangeable and simultaneously usable — the source of the cost savings. **Drive it with centralized policy**: express intent once and let the controllers compile it out to the fabric, replacing per-device configuration with system-level programming. Everything else in this book — OMP, TLOCs, colors, application-aware routing, segmentation — is machinery in service of these three ideas.

1.3 Meet the Platform: Cisco Catalyst SD-WAN

From Viptela to Catalyst SD-WAN (and why you'll see both names)

The platform you are about to deploy began at **Viptela**, a startup founded in 2012 by former Cisco engineers who built a clean-sheet SD-WAN architecture around the plane-separation idea from the previous section. Cisco acquired Viptela in 2017 and made its architecture the basis of Cisco's flagship SD-WAN offering — first sold as *Cisco SD-WAN*, and since 2023 as **Cisco Catalyst SD-WAN**.

That history hands you a quirk you will live with every day on the project: **almost everything has two names.** The original Viptela names (vManage, vSmart, vBond, vEdge) still appear in CLI output, log messages, API paths, and a decade of documentation and forum posts, while Cisco's current branding uses descriptive names (SD-WAN Manager, Controller, Validator, WAN Edge). You have to be fluent in both. This book gives the current name first with the classic name in parentheses, then uses them interchangeably — exactly as you will encounter them in tickets and on calls.

> ⚠️ **Common Problem — Old tutorials will teach you the wrong CLI**
>
> A lot of freely available Viptela material shows Viptela-OS commands like `show control connections`. On today's Cisco IOS XE edge routers the equivalent is `show sdwan control connections` — same concept, different syntax. When you follow an online guide and a command "doesn't exist," the usual cause is that the guide targets vEdge/Viptela-OS while you are on a cEdge/IOS XE box. Always check which OS a resource assumes.

The four components you will spend the book configuring

Cisco Catalyst SD-WAN splits the architecture into **four planes**, each implemented by a dedicated component. This is the single most important picture in the platform — every later chapter maps onto it — so it is worth committing to memory now.

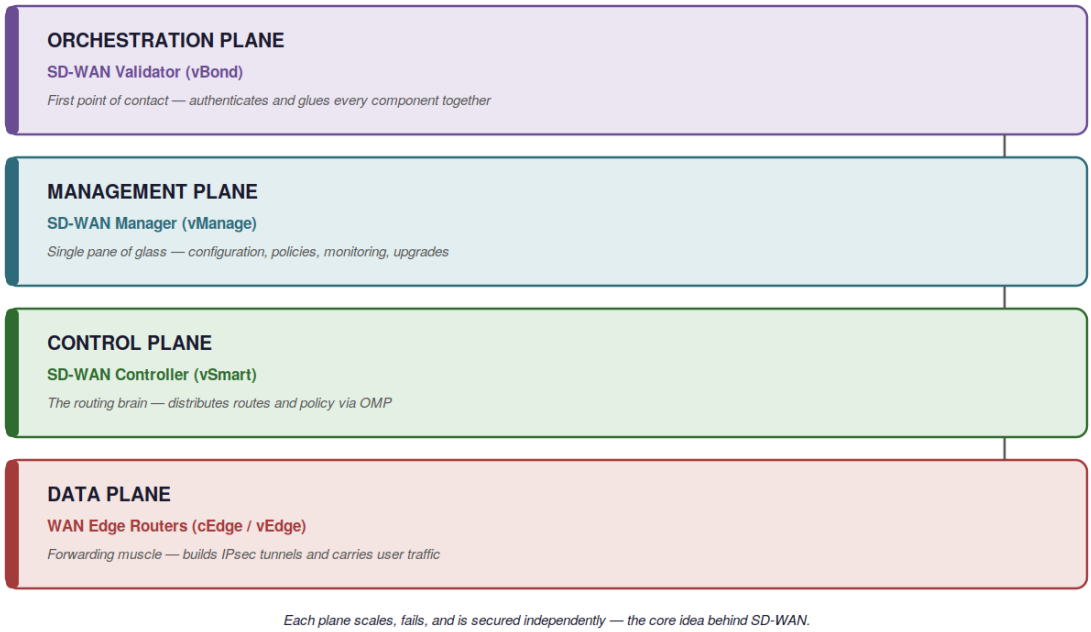

Figure 1.2 — The four planes of Cisco Catalyst SD-WAN and the component that implements each. Three are software; only the WAN Edge is normally a physical device at your sites.

Current name	Classic name	Plane	What it does for you
SD-WAN Manager	vManage	Management	Your GUI and API: build configuration, author policy, monitor, and upgrade the fabric from one place.
SD-WAN Controller	vSmart	Control	The routing brain: learns routes from every edge, applies your policy, advertises best paths over OMP. Never touches user traffic.
SD-WAN Validator	vBond	Orchestration	The bouncer at the door: authenticates every device joining the fabric and tells each one how to find the others. Also helps traverse NAT.
WAN Edge router	cEdge / vEdge	Data	The muscle at each site: builds the encrypted tunnels and forwards every user packet.

Three of the four — Manager, Controller, and Validator — are **software**, run as virtual machines. For most projects today, including Meridian's, these are **hosted by Cisco in the cloud**, so your team never racks a controller; you consume them as a service. They can also run in a public cloud you own or on-premises if a requirement demands it. Only the **WAN Edge** is normally a physical box, sitting at each branch and in the data center (with virtual edges available for cloud sites and labs).

One more naming pair matters at the edge. **vEdge** means the original Viptela hardware running Viptela OS; **cEdge** means a Cisco router — ISR, ASR, or today's **Catalyst 8000 family** — running IOS XE in controller-managed SD-WAN mode. New builds are overwhelmingly cEdge on IOS XE, so that is what every configuration example in this book targets. The “c” simply stands for *Cisco*: a Cisco router doing the vEdge's job.

How the pieces come together: the join sequence

Here is the whole system in one story — each step becomes a full chapter later. When Meridian's new store router powers on, it uses zero-touch provisioning to reach the **Validator (vBond)**, which checks it against a signed list of allowed devices and points it at the other controllers. The edge then registers with the **Manager (vManage)**, which pushes its complete configuration from a template. Next it forms a secure control session with the **Controller (vSmart)** and starts exchanging routing information over **OMP** — the Overlay Management Protocol, the control-plane protocol you will meet properly in Chapter 2.

Through OMP the edge learns about every other site and each site's tunnel endpoints (called **TLOCs**), then builds encrypted IPsec tunnels across every available transport, each one watched continuously by **BFD** probes measuring loss, latency, and jitter. From that moment your centralized policies decide which applications take which paths, which sites may talk to which, and what happens when a path degrades. The clerk at the store just sees applications that work. When something *doesn't* work, this same sequence is your troubleshooting map — you walk it in order to find where the join broke.

Your first health check on any edge: are the three control sessions up? (IOS XE / cEdge)

```
Router# show sdwan control connections
                                  PEER                      PEER
PEER    PEER   SYSTEM-IP   SITE   PUBLIC                    PUBLIC
TYPE    PROT               ID     IP                        PORT   STATE
-------------------------------------------------------------------------
vsmart  dtls   10.0.0.11   100    203.0.113.11              12346  up
vbond   dtls   0.0.0.0       0    203.0.113.10              12346  up
vmanage dtls   10.0.0.1    100    203.0.113.5               12346  up

! Validator (vbond), Controller (vsmart), Manager (vmanage) all "up"
! = this edge has fully joined the fabric. Any session "down" or missing
! points you straight to the stage that failed.
```

Catalyst SD-WAN versus Meraki — don't confuse the two

Cisco sells **two** distinct SD-WAN product lines, and mixing them up is a classic early mistake. **Catalyst SD-WAN (Viptela)** — the subject of this book — targets medium-to-large enterprises that need rich routing (full BGP/OSPF interoperability, complex custom topologies, service chaining) and deep policy control; Meridian fits here squarely. **Meraki SD-WAN** targets lean-IT organizations that value simplicity, managing security appliances and SD-WAN together from the Meraki cloud dashboard with far fewer knobs. Both are legitimate — they suit different operating models. When someone says “Cisco SD-WAN” with no qualifier, they almost always mean the Catalyst/Viptela platform you are learning here.

📐 **Design Decision — Cloud-hosted controllers are the default — and usually the right call**

For a company like Meridian, letting Cisco host the Manager, Controller, and Validator removes a whole category of work: no controller VMs to size, patch, back up, or make highly available yourselves. Choose self-hosted controllers only when a concrete requirement forces it — strict data-residency rules, an air-gapped environment, or a policy against cloud-hosted management. We will weigh this properly in Chapter 3; for now, assume cloud-hosted unless something rules it out.

🌍 **From the Field — The two-names problem bites during incidents**

On a live network you will get tickets that mix eras in a single thread: a monitor alert about “vSmart unreachable,” a change request that says “SD-WAN Controller policy,” and a colleague on the bridge calling the same VM “the smart.” Being fluent in both vocabularies isn't trivia — it is how you avoid routing a P1 to the wrong team at three in the morning.

Chapter Recap

Cisco Catalyst SD-WAN is Viptela's four-plane architecture, acquired by Cisco in 2017 and rebranded in 2023 — which is why every component carries two names you must both know. Its four parts map one-to-one onto the planes: **Manager (vManage)** for management, **Controller (vSmart)** for control, **Validator (vBond)** for orchestration, and **WAN Edge (cEdge/vEdge)** for the data plane. The three controllers are software, usually Cisco-cloud-hosted; the edges are IOS XE routers such as the Catalyst 8000 family. A joining edge is authenticated by the Validator, configured by the Manager, fed routes by the Controller over OMP, and then builds BFD-monitored IPsec tunnels to its peers — a sequence that doubles as your troubleshooting checklist. Keep Catalyst SD-WAN separate in your mind from the simpler Meraki SD-WAN line. With this foundation in place, Chapter 2 opens the hood on how the fabric actually moves a packet.

End of Free Sample

You've just read the opening of SD-WAN Fundamentals — the first chapter of a complete, end-to-end SD-WAN project.

The full book takes you the rest of the way: designing, building, migrating, steering, securing, extending, and operating a real Cisco Catalyst SD-WAN fabric — 10 chapters, 60+ diagrams, real configurations, and a full troubleshooting playbook.

Get the complete book on LeanPub

Search: SD-WAN Fundamentals · Jozef Baroš