

TEST NETWORK

Test before you touch production: the Oktavia test network

CML

Cisco SD-WAN

Lifecycle

11 October 2026 · 11 minute read

Opening a site, moving a router to a new model or closing a branch are changes almost nobody rehearses, because rehearsing them costs as much as doing them. This article explains how Oktavia uses a test network in Cisco Modeling Labs and the Cisco SD-WAN Manager so that each of those changes is designed against intent, tested, approved, applied and verified, always in that order.

The change nobody rehearsed

Small configuration changes have a procedure: a window, a review, a backup. Large changes, oddly, almost never do. Opening a branch, swapping a site's routers for a new model or closing an office happen a few times a year, each time with a different engineer, and what was learned last time lives in an email or in someone's memory.

The outcome is familiar. The new site comes up, but with hand-made addressing and an access list copied from another site that no longer applied. The new router boots, but one port ended up on the wrong card. The closed branch keeps showing on the hub as a down BGP neighbor for months, because nobody removed its side of the tunnel.

A lifecycle change that cannot be rehearsed runs for the first time in production, every time.

The idea: design against intent, test on a twin, and only then apply

Oktavia starts from a source of truth: the inventory and topology the network describes about itself, and on top of them the declared intent of what each device should have configured. Every change, from a VLAN to a whole site, is first expressed as a change to that intent, not as commands.

From there, the same path applies to everything:

- **Design.** The change is proposed from what already exists —the design of the other sites, the equivalent port on the new model, the free addressing— and a person edits it before moving on.
- **Test.** The digital twin renders the configuration that results from that intent and compares it with the device. When the change involves new devices, the test network brings them up so it can be rehearsed

end to end.

- **Approve.** A person reviews the exact diff that will be sent. Nothing is applied without that approval.
- **Apply.** With each vendor's safe primitive.
- **Verify.** A new audit, and connectivity tests when needed, confirm the network ended up as declared. It is what we call closing the loop in [the article on the digital twin](#).

Two design decisions hold this together. First: Oktavia does not improvise configuration. What is sent to a device comes from intent and deterministic templates, and it can be read before it is approved. Second: artificial intelligence is optional and never writes to the network. When used, it explains or diagnoses over a bounded summary of the device or site, never over full configurations.



The test network: what Oktavia does in CML and what it leaves to the operator

Today the Oktavia test network is Cisco Modeling Labs (CML). That is where Oktavia builds the devices of a change before they exist in production, under agreed rules so there is never doubt about who did what:

- **It creates nodes with a minimal day 0.** Only what is needed to log in over SSH: hostname, management interface and the user from the credential vault, the same one production will use. The day-0 configuration, which stays stored in the lab, carries only the password hash, never the clear-text password. Everything else —VLANs, routing, services, hardening— arrives later through the loop, with testing and approval.
- **It wires them.** It connects the new node to its neighbors according to the design, and can add a test PC per access port to check that users get an address and can reach out.
- **It never powers a node on.** Powering on is always the operator's call. Oktavia does power off, but only the device leaving in a replacement or a removal.
- **It records every operation.** The state of each operation is stored per tenant, so an interrupted operation can be resumed where it stopped, or undone if it failed halfway.
- **It tags what it creates.** Each node carries the `oktavia:device=` tag with the name of the device it represents. Even if someone moves or renames nodes in the lab, Oktavia knows which is which, and a removal takes exactly what is its own and nothing else.

Before touching the lab, Oktavia confirms CML is up and ready. It looks like a detail, and it is what prevents half-done operations against a lab that was still booting.



Site and device lifecycle

This is the heart of it. In Oktavia, opening and closing sites, and onboarding, replacing and retiring devices, are not craft projects: they are one-button flows, each with its preview, its approval and its verification. The examples use a lab with IOS branches in CDMX and Monterrey and a hub in Guadalajara (GDL).

Creating a new site

A new site is not designed from scratch: it is cloned from the design of its site type. For example, a «Querétaro» branch with the same design as the existing ones: two WAN routers and one LAN switch, each router with a GRE tunnel and eBGP to its hub.

- **Proposed, editable addressing.** Oktavia proposes free addresses, VLANs, autonomous system and tunnels, translated from the reference site. The person reviews them and changes them if needed; the proposal is checked against what is already in use before it is saved.
- **Intent for the three devices and the hub.** The site's intent is saved and, on the hub side, what belongs to it: its BGP neighbor, its tunnel, its subinterface and the VLAN on the transport switches. That side is applied through the hub's own loop, with its own diff.
- **Nodes in the lab.** Oktavia creates the three nodes with their minimal day 0, wires them and adds test PCs. The operator powers them on, and each device goes through its own onboarding.

Removing a site

Removal mirrors creation, and it is where the difference from doing it by hand shows most:

- **It cleans the hub in order.** It builds the removal lines in the sequence that leaves nothing dangling: first no neighbor in BGP, then routes, tunnel and subinterface; on the layer 2 side, access port, trunk, VLAN interface and VLAN.
- **It cleans Oktavia.** It archives each device and then removes its intent, inventory, compliance results, rules, telemetry and known SSH keys.
- **It cleans the lab.** It removes the site's nodes and their test PCs, identified by their tag.
- **It validates that nothing is left.** At the end it checks Oktavia, the lab and the hub's running configuration, and reports any remnant of the site.

Because creation and removal are exact, the cycle is idempotent: create, remove and create Querétaro again can be repeated before every demo, and the network ends up the same each time.

Onboarding a device

A new device, in a new or an existing site, goes through a series of visible steps: access (ping, SSH port and login with the vault credential), discovery and full audit, telemetry enabled if requested, loop closure with the digital twin, and a final verification with its routing state and compliance. The bring-up configuration comes from intent —banners, access lists and the console line included— and only includes the ports the model actually has.

Replacing a device

Swapping a device for another model under the same name, such as an IOSv router for a Catalyst 8000v or an access switch for a 16-port one, takes three steps:

- **Prepare.** Oktavia reads the current device's ports, suggests the mapping to the new model and lets you edit it. The preview shows which intent ports would be left uncovered and how many references change.

The new node is created with its day 0, unwired, while the old one stays in service.

- **Cut over.** Each link of the old device moves to the equivalent port on the new one, and the old one is powered off. The operator powers on the new one and its configuration arrives through the loop.
- **Finish.** The old device is removed from the lab once the new one is powered on; the practice is to do it after its verification.

A replacement changes the device's SSH identity: same address, different key. Oktavia stores known keys per tenant and rejects a changed key until the operator explicitly confirms it. It is never accepted on its own, because a key that changed can also be an impersonated device.

Retiring a device

Retiring a device starts by archiving it: a downloadable package with its inventory, intent, last backed-up configuration and compliance, without secrets. Then its intent, inventory, compliance results and telemetry are removed, so it stops showing up as a down device on the dashboards.

FLOW	WHAT OKTAVIA PROPOSES	WHAT THE PERSON DECIDES
Site creation	Clone of the site-type design, free addressing, hub side, test nodes	Final addressing, approval of each change, powering on the nodes
Site removal	Ordered removal lines, Oktavia and lab cleanup, final validation	Approval of the hub change
Device onboarding	Minimal day 0, configuration from intent, telemetry	Approval of the loop
Device replacement	Port mapping, new node, link cutover	Final mapping, power-on, confirmation of the new SSH key
Device retirement	Archive package and cleanup	Confirming the retirement

✓ Connectivity tests built from intent

An audit says whether configuration and state match what was declared. For a site that just came up, the user's question also matters: does it work? Oktavia builds the tests from intent, without inventing targets:

- **Per site:** users get an address over DHCP and respond, the default route is present, the other sites are reachable and, for information only, so is the internet.
- **Per device:** each declared interface is up, each tunnel answers at the far end, OSPF and BGP neighbors are established and the default route exists.

They are optional buttons, not a mandatory step of the loop, on purpose: the first pings of a new site fail while routing converges and the PCs request addresses. That is why they can be repeated, or left running until they pass, instead of flagging as broken something that was only starting up.



Cisco SD-WAN: the same loop, through the Manager

In a Cisco SD-WAN network configuration is not written device by device: it lives in the Manager. That is why Oktavia integrates with the API of the lab's Manager instead of logging in to the edges over CLI, and the logic is the same:

- **Overlay state.** Oktavia collects control connections, BFD sessions and OMP peers, and audits them like any other state rule.
- **Onboarding an edge or a site.** The new edge is attached to its Config Group with variables proposed from a reference edge of the same design, and Oktavia generates the bootstrap it boots with. Credentials are taken server side and never appear in the preview.
- **LAN segments.** A new segment is declared as intent and the twin creates it in the Config Group; its removal takes it out of every edge in the group, protecting what is still in use.

Design, test, approve, apply and verify: the primitive changes —the Manager instead of the CLI— and the loop does not.



Why it matters

- **Rehearsals you can repeat.** A lifecycle change can be rehearsed as many times as needed, and undone, without rebuilding the lab by hand.
- **Honest demos.** What is shown is the same flow used in production, with the same diff and the same approval, not a staged presentation.
- **Production rehearsed on its own design.** The test site is a clone of the real design, so what fails in the lab is what would have failed in production.
- **Nothing left halfway.** Validated creations and removals avoid the leftovers that pollute compliance for months.

Rehearsal stops being a separate project and becomes the first step of every change.

When did you last rehearse opening a site?

If the answer is «never» or «we did it live», we can show you the full creation and removal of a branch in a demo. To quote, we only need a rough idea of how many devices you manage.

About Oktanet

Oktanet is a Mexican network automation company focused on controlled, traceable, and scalable operations for enterprise environments across Latin America.

[LinkedIn](#) · [YouTube](#)

© 2026 Oktanet · Oktavia · Public article, with no confidential customer information.