alkira

WHITEPAPER

Beyond Colocation Hubs: Network Modernization in the AI Era

for Network, Infrastructure, and Cloud Teams

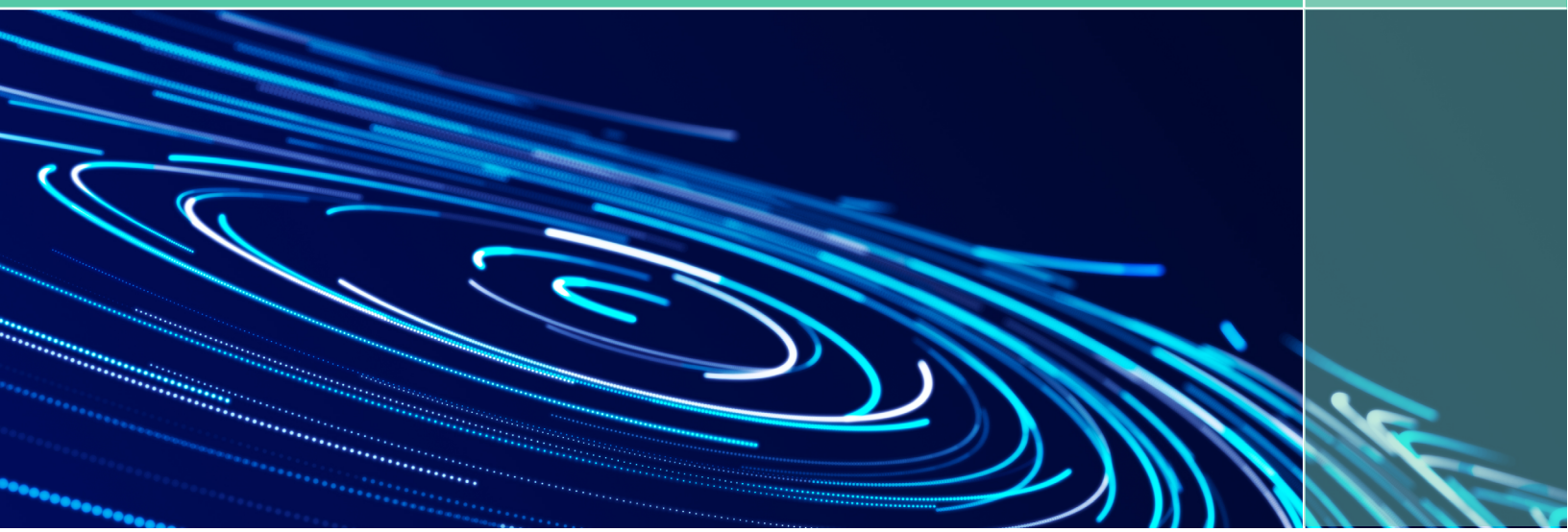


Table of Contents:

Modernization succeeds when connectivity, segmentation, and operations shift from hardware-bound, per-environment designs to a centralized, cloud-delivered operating model that scales with cloud and AI delivery velocity.

At-a-glance: what breaks, what to require, and how to migrate

What changed

- Application architectures create more service-to-service traffic and more continuous data movement.
- Traffic is now many-to-many: on-prem to cloud, cloud to cloud, region to region, site to SaaS, and partner to application.
- Change is more frequent, so networking must support repeatable updates, not long build cycles.

Constraints of traditional models

- Legacy architectures are often built around a few fixed paths, so as traffic spreads they add extra hops and create shared dependency points.
- Control planes fragment across clouds, WAN overlays, and appliances.
- Segmentation drifts, creating security exceptions and audit friction.

New modernization requirements

- Any-to-any routing across clouds, data centers, edge, partners.
- Consistent segmentation and centralized policy enforcement.
- API/IaC automation, change guardrails, and end-to-end observability.

Migration blueprint

- Phase 1: map flows and segmentation intent, define success tests.
- Phase 2: pilot the fabric with one segment, validate visibility and rollback.
- Phase 3: migrate high impact paths first: inter-cloud, partner access, data movement.
- Phase 4: scale with templates, reduce duplicated tunnels and hub dependency.

Executive Summary

Enterprises built colocation-era networks to connect offices and data centers to cloud on-ramps. That model breaks down when applications, data, and users are distributed across multiple clouds, SaaS, edge locations, and partner ecosystems, and when AI programs increase east-west traffic and data movement requirements.

For technical leaders, the modernization problem is no longer limited to bandwidth or routing design. It is an operating model problem: how to deliver consistent connectivity, segmentation, and security across environments with change velocity that matches cloud and AI delivery cycles.

The technical gap

- Cloud teams can provision compute, storage, and identity-driven controls quickly.
- Network and security controls are still implemented as per-environment configurations and device-bound change windows.
- The result is segmentation drift, duplicated intent, and higher change risk as distribution increases.

What Changed: Workloads, Traffic, and Operating Constraints

Topology mismatch and backhaul traffic

Older hub-and-spoke designs were built for a world where most traffic went in and out of a central data center. In a distributed world, that same design often forces traffic to detour through a central hub even when the two endpoints are close to each other.

What this looks like in practice

- **Cloud-to-cloud traffic detours through a hub** (sometimes a colo or a central security stack), even when workloads are in the same metro or region.
- **Security becomes a bottleneck** when inspection is centralized, because more traffic is forced through the same choke point.
- **The blast radius grows** because a single hub outage or misconfiguration impacts many unrelated apps and teams.

Why it matters

This affects latency, congestion, and operational risk, not just bandwidth.

High lifecycle overhead

A large portion of legacy networking is still tied to physical infrastructure. That means scaling, redesigning, or expanding the network often requires more than configuration work. It requires physical steps and lifecycle management that slow execution and add operational load.

What this can look like

- New capacity or new services require purchasing hardware, waiting on lead times, and aligning budgets.
- Teams must rack and stack, cable, and stage devices, then coordinate maintenance windows for cutover.
- Ongoing operations include patching, upgrades, RMA swaps, and spares management.
- When architectures change, teams have to unrack, decommission, and manage asset disposition.

Why it matters

This introduces a different pace than cloud delivery. Even when designs are sound, hardware-bound workflows can extend timelines, increase coordination overhead, and make it harder to keep up with fast-moving cloud and AI programs.

Fragmented control planes

In hyper-distributed environments, networking and security policy tends to be defined in multiple systems. Each environment (especially clouds) has its own primitives, and the WAN/security stack adds more layers. Over time, "what we intended" and "what is actually enforced" drift apart.

What this looks like in practice

- Each environment, especially cloud environments, uses different constructs for routing and security, so the same intent has to be re-implemented multiple times.
- WAN overlays and security appliances introduce separate operational workflows and tooling.
- Teams end up **copying policies between environments**, leading to inconsistency, exceptions, and debugging overhead.

Why it matters

Policy becomes harder to standardize and easier to break. Small differences accumulate into real security and reliability issues.

Segmentation drift

Segmentation is frequently built using environment-specific constructs (subnets, ACLs, VRFs, firewall rule sets). As environments evolve, segmentation can remain correct locally, but vary across domains.

What this looks like in practice

- Segmentation rules vary by cloud, by site, or by team, so "prod vs non-prod" or "regulated vs non-regulated" is not enforced the same way everywhere.
- Exceptions and special cases grow naturally as new apps and partners are onboarded.
- Teams need more time to validate access paths across domains during troubleshooting or audits.

Why it matters

Segmentation remains valuable, but consistency and evidence can require more effort as the number of environments and stakeholders increases.

AI-era requirements (more data movement + tighter boundaries)

AI initiatives often increase cross-domain data movement and expand the number of systems that need controlled access to sensitive datasets and model artifacts. This raises the bar on consistency and predictability.

What this can look like

- Data pipelines require predictable throughput across cloud and on-prem boundaries.
- Some inference and real-time paths benefit from stable latency and jitter characteristics.
- AI datasets and model artifacts create new "sensitive zones" that need consistent isolation and least-privilege access wherever workloads run.

Why it matters

AI programs tend to highlight where connectivity, segmentation, and policy consistency need to be repeatable across domains, because data and workflows are distributed by design.

Requirements for a Modern Enterprise Network in the AI Era

If you are looking to reduce reliance on colocation hubs, or you are feeling constrained by colo lead times and change cycles, focus on the operating model, not a feature checklist. For a modern enterprise network, the baseline requirement should be that your team can deliver connectivity, segmentation, and required security controls without expanding your colo footprint or standing up new hub stacks. That means fewer hardware purchases, fewer rack and stack cycles, and less per site appliance sprawl. At the same time, the model should integrate cleanly with your existing data centers, WAN edges, and cloud environments so you can transition in phases without a redesign.

Recommended: -aaS Delivery Model

- **No infrastructure build-out to consume the service:** adoption should not require procuring hardware, staging software images, or deploying and maintaining a new appliance fleet to stand up the network fabric.
- **Provider-operated lifecycle** for the underlying platform: capacity expansion, resiliency, and upgrades are managed as part of the service so teams are not planning modernization around refresh cycles.
- **Fast time-to-attach** new clouds, sites, and partners: onboarding should not be gated by long infrastructure lead times (for example, building out a new colocation footprint or waiting on a complex hub expansion).

Hyper-agile connectivity and routing

- **Any-to-any connectivity** across data centers, clouds, edge sites, and partners without redesigning topology per environment.
- **Direct east-west routing** that minimizes hub detours and supports multi-region architectures (cloud-to-cloud, site-to-cloud, cloud-to-partner).
- **Standards-based routing integration** (for example BGP where appropriate) so existing WAN and data center routing can interoperate during phased migrations.

Consistent and end-to-end segmentation and security

- **Consistent segmentation model** that spans cloud and on-prem environments (segments or VRFs mapped to intent), so "prod vs non-prod" and sensitive domains are enforced the same way across locations.
- **Centralized policy definition with uniform enforcement** across attachment points, reducing per-environment policy variance.
- **Flexible security insertion** that supports required inspection controls without forcing all traffic through a single centralized enforcement point (which often sits in a colo hub in legacy designs).

Simplified operations, automation, and observability

- **Unified control plane** for topology, segmentation, and policy, with API and Infrastructure-as-Code support to make changes repeatable.
- **Change safety built in:** RBAC, guardrails, and auditability that reduce misconfiguration risk as change velocity increases.
- **End-to-end telemetry** across clouds and sites, including flow visibility by segment and enough context to support troubleshooting and audit evidence.

High resilience

- **Multi-region resilience patterns** with clear failure domains and recovery options aligned to application availability goals.
- **Elastic scale** for bandwidth and services aligned to workload demand, including sustained data movement and burst events.
- **Reduced dependency on refresh-driven modernization:** the model should not require periodic rebuilds of hub infrastructure (often colocated) to expand capacity, insert new services, or keep platforms current.

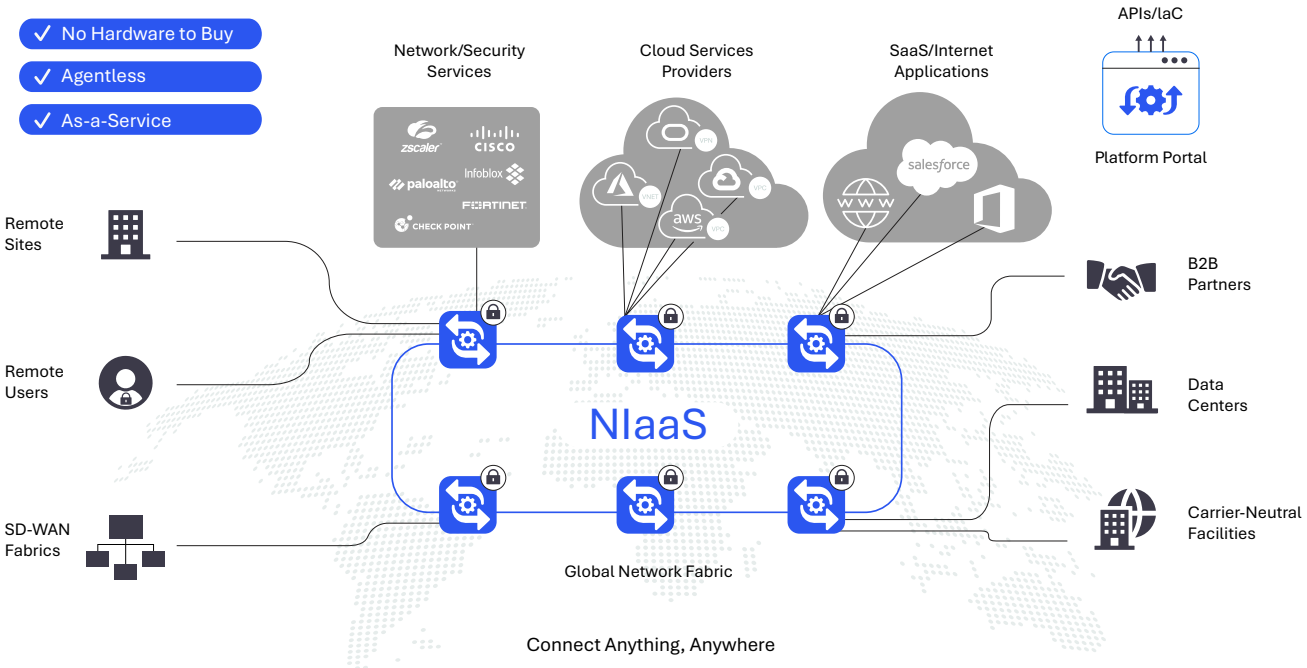
NIaaS as an Architectural Pattern

Network Infrastructure as a Service (NIaaS) applies cloud operating principles to the enterprise network. Instead of building connectivity by assembling devices, tunnels, and per environment designs, teams consume a cloud delivered fabric that provides connectivity, segmentation, and policy through a centralized control plane.

In colocation based models, the colo hub often becomes the default place to terminate circuits, host security stacks, and route traffic between clouds, sites, and partners. Over time, that hub becomes both a scaling point and an agility constraint. Expanding capacity, adding a new cloud region, onboarding a partner, or changing segmentation can turn into a colo project with procurement, cross connects, rack and stack, maintenance windows, and ongoing hardware lifecycle work.

NIaaS reduces that dependence by shifting the fabric and core services to a provider operated platform. Colocation can still play a role where you need private interconnect or proximity to specific ecosystems, but it no longer has to be the primary transit and control point for the enterprise network.

Reference Architecture



Core building blocks

- **Global fabric and points of presence:** Distributed network services close to workloads, users, and clouds.
- **Attachment models:** Standardized ways to connect public clouds, data centers, sites, and partner environments.
- **Policy and segmentation engine:** Define segments and routing intent once, then apply consistently across environments.
- **Service insertion:** Integrate inspection and firewalling where required without creating per site architectural variance.
- **Automation interfaces:** API first operations and Infrastructure as Code support for repeatable deployments.
- **Observability:** Centralized monitoring and flow analytics across clouds, sites, and partner connections.

What NIaaS is not

- Not a management dashboard layered on top of device by device operations.
- Not an overlay that still requires the enterprise to scale, upgrade, and maintain the underlying infrastructure.
- Not a single cloud construct that increases lock in when multi cloud and partner connectivity are requirements.

A Practical Migration Path to NIaaS

A successful move away from colocation centric networking is rarely a single cutover. It is a phased transition that shifts the operating model from hardware and hub management to a consumable network fabric. The simplest way to keep risk low is to migrate one traffic class at a time and validate three invariants at every step: **reachability, segmentation,** and **visibility.**

In most enterprises, colocation hubs became the default place to terminate circuits, host shared security stacks, and route traffic between clouds, data centers, sites, and partners. As the environment becomes more distributed, that hub model can turn routine changes into colo work. Adding capacity, onboarding a new partner, or extending segmentation often means cross connects, rack and stack, maintenance windows, and ongoing lifecycle management.

Instead of an enterprise built and maintained colocation stack, the transit and policy layer is delivered as a cloud-native operated fabric with distributed points of presence. Colocation can still be used where private interconnect or proximity matters, but it is no longer the place your team has to build, scale, and maintain in order to keep the network moving.

Step 1: Identify the colo constraints you want to remove first

Start with a use case where your current colo hub model is creating measurable friction. The point is not to "move everything out of colo" on day one. The point is to remove the dependencies that turn routine network changes into colo work.

Good first targets:

- **Inter cloud paths** that are forced through a colo hub for transit or inspection, adding latency and increasing shared dependency.
- **Partner and third party connectivity** that requires bespoke VPNs, one off firewall rules, or manual provisioning tied to hub operations.
- **Shared services access** where consistent segmentation is difficult because policy is implemented differently across clouds, on prem, and the hub stack.
- **High volume data movement** where throughput and policy consistency are constrained by centralized enforcement and hub capacity planning.

What to look for in a NIaaS-type model:

- A provider operated fabric that can carry inter cloud, site, and partner traffic without requiring your team to scale a colo stack to support it.
- A consistent segmentation model that spans clouds and on prem, so the same segment intent applies everywhere.
- Security insertion options that do not force all flows through a single physical hub footprint you own and operate.

Step 2: Define acceptance criteria before you move traffic

Write success criteria in clear technical terms so the pilot is measurable and repeatable.

Consider these criteria:

- **Connectivity:** which environments must route to which others, and which must not.
- **Segmentation:** which segments exist and what is allowed and denied between them.
- **Inspection:** which traffic requires security insertion, and where enforcement should occur.
- **Visibility:** what telemetry is required for troubleshooting and audit evidence, including flow level context.
- **Rollback:** how to revert without impacting unrelated segments or environments.

Step 3: Use integration first patterns that preserve stability

- Most teams do not replace WAN, data center routing, or security platforms on day one. The goal is to attach, validate, then expand, while keeping production stable.
- Common integration points:
- WAN and data center routing integration at the edge, using standard dynamic routing where appropriate, so traffic can shift away from colo transit without a redesign.
- Cloud attachments per provider with consistent segment mapping so a segment means the same thing in each cloud and on prem.
- Security insertion that applies inspection consistently by segment or traffic class without requiring every flow to traverse a single centralized colo stack.
- IP reality handling, including overlapping CIDRs and legacy addressing, planned early so it does not block the rollout.

Step 4: Run a pilot that proves the operating model, not just connectivity

- A strong pilot should prove repeatability and operational safety. Examples include:
- Two cloud environments plus one data center or WAN attachment.
- Three segments that matter operationally, such as prod, shared services, partner.
- One inspection requirement using service insertion.
- End-to-end telemetry with flow and policy context.
- A rollback procedure you execute once under controlled conditions.

Step 5: Expand with templates and reduce colo dependency where it is acting as the constraint

- Once the first use case is stable, scale by repeating the pattern and standardizing delivery.
- Expand by region, business unit, or site using templates.
- Migrate additional traffic classes in order of friction and value.
- Where colo hubs are primarily serving as transit, policy enforcement, or shared security hosting, shift those functions to the provider operated fabric. Keep colo where it provides specific value, such as private interconnect, ecosystem adjacency, or proximity requirements.

Common pitfalls to avoid

- Recreating a new centralized hub as the default path for east west traffic.
- Allowing segmentation to diverge by environment, which recreates operational variance.
- Treating visibility as optional. Without flow and policy evidence, troubleshooting and audits stay manual and slow, even if the topology changes.

Validation: Tests, Runbooks, and Success Metrics

Once you have selected an NIaaS approach and migrated an initial colo-based use case, the next step is validation. The objective is to confirm the fabric delivers consistent routing, segmentation, and required security insertion through a centralized control plane, with repeatable operations and audit-ready visibility.

Tests for reachability and segmentation

- **Connection map test:** confirm each segment can reach what it should, and cannot reach what it should not, across cloud, data center, and sites
- **Routing sanity check:** confirm routes are correct and traffic takes the expected path, including any overlapping IP ranges
- **Partner access check:** confirm partners can reach only what they need, and access can be limited by time if required

Operational tests

- Build test: create a new connection and segment using your automation tools, and confirm the change is recorded
- Undo test: roll back a change quickly without breaking other segments
- Isolation test: isolate one segment during a drill and confirm other segments keep running, with clear visibility into what changed

Metrics to track

- Time to bring up a new region (cloud or site) with the same segmentation and rules as everywhere else
- Time to onboard a new partner with limited access and a clear record of what was granted
- Change reliability: how often network changes cause issues, and how often you need emergency fixes
- Policy consistency: how often you have to add special rules to make things work, and whether the same rules are applied the same way in every cloud, data center, and site.

The Alkira Platform, Delivering Network Infrastructure On-Demand

Alkira provides Network Infrastructure as a Service through a globally available, secure, cloud-based fabric and centralized control plane, with no network hardware or software to deploy. Teams use Alkira to connect public clouds, data centers, branches, and partner environments into one network model with consistent segmentation and policy. This helps reduce per environment differences and makes network changes easier to repeat, review, and track.

Technical capabilities typically evaluated

- Centralized design and operations: define segments and connectivity intent once and deploy across domains.
- Multi-cloud connectivity: connect and route between AWS, Azure, GCP, OCI, and private environments through one fabric.
- Integrated service insertion: apply security and network controls consistently using existing tools without re-architecting and adding complexity
- API and Infrastructure-as-Code support: integrate network provisioning into cloud and platform delivery workflows.
- End-to-end visibility: observe flows and segmentation posture across clouds, data centers, and partner links.

Fit for common modernization use cases

- Colocation exit and refresh avoidance by shifting to a cloud-delivered fabric for interconnect and policy.
- Multi-cloud routing and segmentation without duplicating constructs and controls per provider.
- Secure partner connectivity with standardized onboarding and segmentation boundaries.
- AI data movement between storage, compute, and inference locations with policy consistency.

Summary

Modern network modernization is an operating model shift. For technical teams, the goal is clear. Deliver consistent connectivity and segmentation across clouds, data centers, sites, and partners without making every change a new build, a new hub dependency, or another round of special-case policy work.

Next steps if considering NIaaS

- **Select the first conversion target** with clear impact, such as inter-cloud routing, partner onboarding, or segmentation consistency.
- **Baseline current performance:** time to deliver common changes, how often changes cause issues, and how many special access rules exist outside your standard model.
- **Run a focused pilot** in one region or business unit, including visibility requirements and a rollback test.
- **Scale with repeatable patterns:** expand by template, then retire legacy tunnels, hub transit paths, and per-environment exceptions as the new model becomes standard.

To learn more about NIaaS and how Alkira can support your transition, visit alkira.com.