anoBytes | TECHNOLOGY SOLUTIONS

Replacing per-store AWS Site-to-Site VPN with Tailscale subnet routers

How a multinational specialty tea retailer cut its monitoring connectivity cost by 91%.

CASE STUDY · NETWORKING · COST OPTIMIZATION · RETAIL

Costs use published list prices as of 2026-08-12, except hardware, which is at the price paid at the time. Client details are anonymized. Published 2026-08-13.

91% lower recurring cost

$4,380 a year down to $384

3.6 months to pay back the hardware

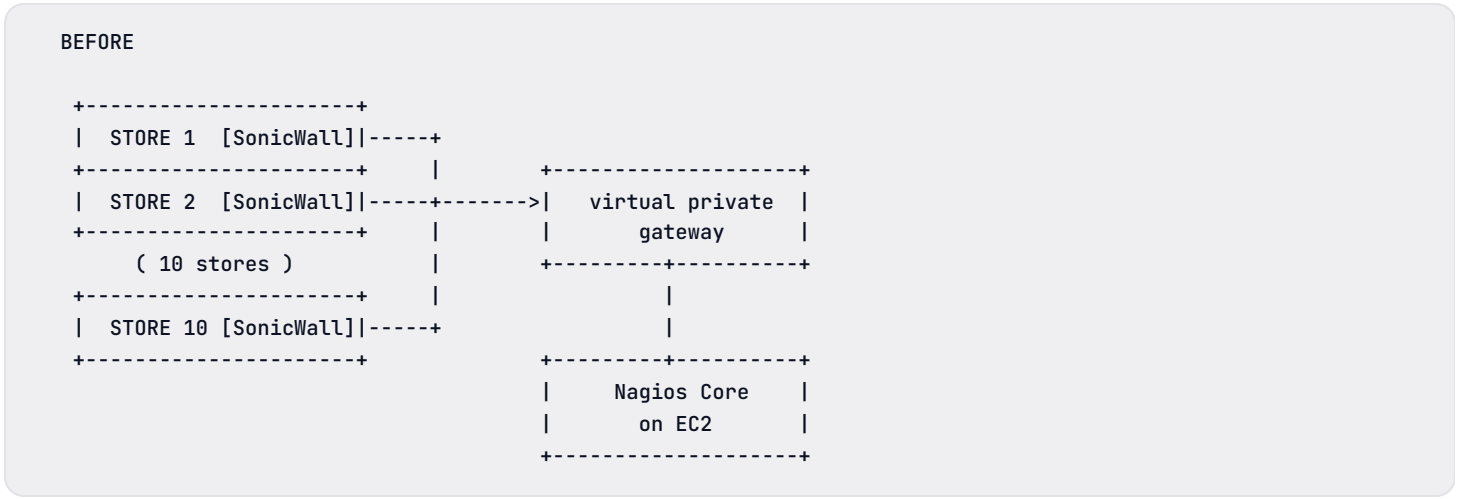
$1,200 across 10 sites

A multinational specialty tea retailer was paying for 10 AWS Site-to-Site VPN connections that carried monitoring traffic and nothing else, and it had already reached AWS's default limit on those connections just as the business prepared to double its store count. Replacing them with a Tailscale tailnet, one Raspberry Pi per store acting as a subnet router, cut the recurring cost by 91.2%, and a new store adds a one-time $120 device and nothing to the monthly bill.

Scale	10 live retail sites, 20 planned
Devices monitored	20 to 30 per store
Retired	10 AWS Site-to-Site VPN connections and their customer gateways
Retained	SonicWall firewalls, the Nagios server, store internet circuits

CHALLENGE Paying for 10 VPN connections to carry pings

The retailer ran Nagios Core on an EC2 instance inside an AWS VPC, polling 20 to 30 devices in every store: IP cameras, point-of-sale (POS) terminals, switches, wireless access points, receipt printers, and the store firewall itself. Each store's SonicWall held an IPSec connection to AWS, and all 10 terminated on a virtual private gateway attached to that VPC. Each connection ran over the store's main internet line only; the firewalls could fail over to a cellular backup for the store's own traffic, but the VPN was not configured on that path, so a store that lost its main line also dropped out of monitoring until the line came back.



Those connections carried only monitoring checks, ICMP pings and SNMP status queries, and nothing else. Each one is built from two IPSec tunnels on the AWS side for redundancy (AWS delivers every connection as a pair of endpoints, so it can service one without dropping the link), so the SonicWalls held 20 tunnel configurations between them. AWS Site-to-Site VPN is priced by the hour at $0.05 per connection-hour, $365 a month across the 10 stores, and the price is the same whether a connection carries heavy traffic or, as these did, kilobytes per minute.

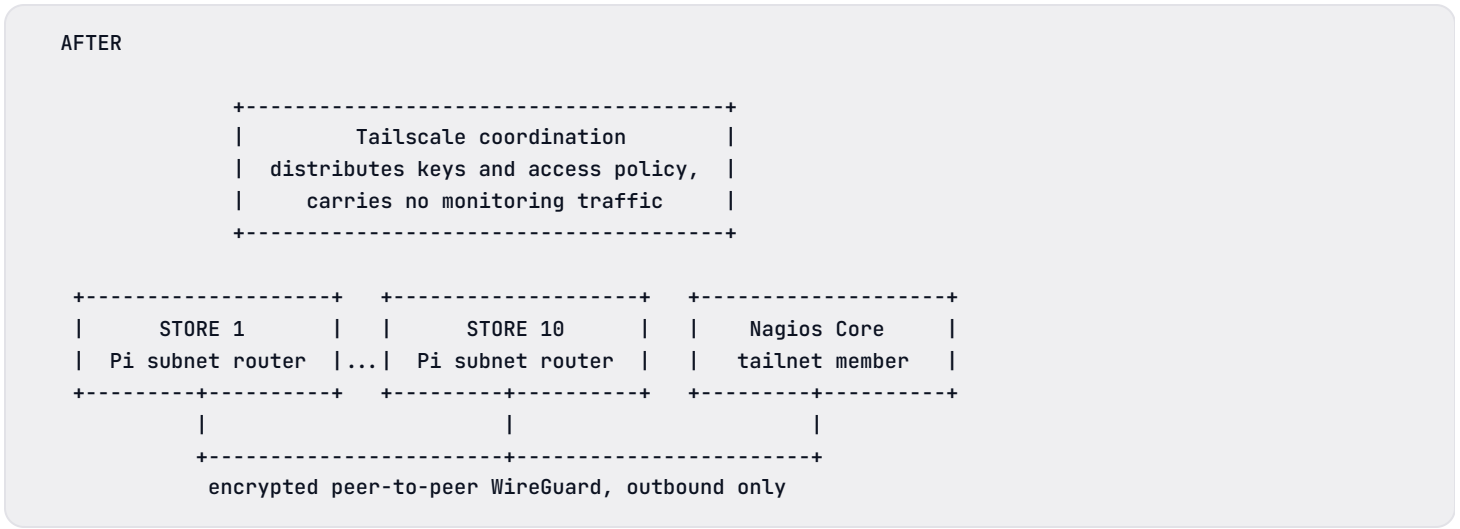
- **A limit that did not show up on the bill**

  AWS allows 10 Site-to-Site VPN connections per virtual private gateway by default, and a gateway attaches to one VPC at a time. With 10 stores connected, the retailer was already at that limit while the business opened toward 20. Store 11 could not join without asking AWS to raise the limit, adding a second gateway and VPC, or moving to a Transit Gateway, which adds its own per-attachment hourly charge on top of the VPN charge already being paid. The stores already planned were going to force a networking change and raise the per-site cost at the same time.

SOLUTION One subnet router per store

The solution had to give Nagios a path to every device without changing settings on any of them, open nothing new to the internet at any store, work with no technical staff on site, and keep monitoring running through the migration.

Each store received a Raspberry Pi running Tailscale as a subnet router: a device that makes everything on the store's network reachable through Tailscale, with nothing installed on the devices themselves. The Nagios host joined the same tailnet (the private network Tailscale creates) as an ordinary member and polled store devices at their local addresses.



Traffic arriving through a subnet router appears by default to come from the router itself, so every camera, terminal, and printer saw polls coming from a local address it already answered. The project was not allowed to change settings on those devices, two to three hundred of them, and it never had to.

Access runs one way. The tailnet policy lets the monitoring server reach the store networks and lets nothing reach back, so a compromised camera cannot reach the monitoring server or another store. And because each Pi dials out to join the mesh, no store needs a static public address or an inbound firewall rule. Dialing out also means the connection follows whichever internet line the firewall is using, so a store that fails over to its cellular backup stays visible in Nagios; the retired tunnels ran only on the main line. The retired connections had each required an inbound IPSec listener on both ends, so the new design has fewer ways in than the old one.

COMPONENT	ROLE	STATUS
Nagios Core on EC2	Monitoring server	Unchanged
AWS Site-to-Site VPN, virtual private gateway	Store-to-cloud transport	Retired
SonicWall firewall, per store	Store perimeter firewalling, dual WAN	Retained
Tailscale	Overlay network, key distribution, access policy	Added
Raspberry Pi, per store	Subnet router advertising the store LAN	Added
Ansible	Fleet build, configuration, and ongoing state	Added

BUILD Implementation

Ansible reached each Pi over SSH with key-based authentication, and one Ansible role did the full setup for a store router: install Tailscale, enable IP forwarding, enroll the device, and advertise that store's subnet. The role describes the finished state rather than a list of steps, so running it again on a misbehaving store is safe: it either changes nothing or fixes what drifted and reports it. That is what keeps the store network maintainable long after the rollout.

A store's devices sit across several VLANs (virtual LANs, the separate network segments a switch keeps apart), and a subnet router can only advertise what it can reach, so each Pi went on the management VLAN, where the firewall routes to every other segment.

Sites migrated one at a time, each cutover taking minutes. Every Pi was built and verified before it went near a store, so the visit was just placing the device, confirming Nagios could poll through the tailnet, and removing the VPN connection. A network engineer on the client side handled the physical work while the cutover ran remotely. Adding a store still takes hardware, shipping, and someone on site; what went away is the networking work: the firewall policy, the customer gateway and VPN connection in AWS, and the routing to match on both ends.

The architecture is smaller as well as cheaper. Store connectivity had been 10 customer gateways, 10 VPN connections, 20 tunnel configurations, and routing kept in step on both ends of every link, each one a thing to document, monitor, and get right during a change. It is now one tailnet policy, one Ansible role, and an inventory. A new store adds a line to that inventory, with no VPN configuration anywhere and no change window on either side.

ISSUE The relay problem

Some months in, Nagios began losing contact with every device in every store at once, recovering, and doing it again. Every store failing in the same moment meant they all depended on something shared, and the shared piece turned out to be a relay.

The store firewalls were performing a form of address translation that stopped the Pis negotiating direct encrypted paths. Every one of them fell back to a shared relay, so the monitoring server was reaching all 10 stores through a single connection, and a brief interruption on that path took all of them down at once.

Two Tailscale commands confirmed it. The status output showed every store router connected through a relay instead of

directly, and the network check showed the firewall assigning a different external port for each destination, the translation behavior that keeps two peers from finding each other. On one store, that port changed eight times in a single hour. After the fix, the same two commands showed direct connections and stable ports.

- **The fix was on the firewalls**
  Enabling consistent source-port translation let each Pi build its own direct tunnel, with the UDP timeout raised to 300 seconds so those tunnels stay stable and an inspection exemption for the encrypted traffic. Consistent NAT opens no inbound ports and weakens no firewall rule; it only makes the source port predictable. UPnP and NAT-PMP, which let internal devices open inbound holes on demand, stayed off, as they should on any retail network. A mesh that falls back to a relay keeps working, so nothing warns you it happened; the connection status has to be checked.

RESULTS What it cost, before and after

Recurring costs use published list prices as of 2026-08-12, so this is a reconstruction at current rates rather than a copy of the client's invoices. Hardware is at the price paid at the time.

LINE ITEM	BEFORE	AFTER
VPN connections	10 x $0.05/hr x 8,760 hr = $4,380/yr	Retired
Overlay devices	n/a	11 devices (one per store plus the monitoring server), inside the included 50 = $0
Subscription seats	n/a	4 x $8/month = $384/yr
Hardware	n/a	10 x $120 one time = $1,200
Year one	$4,380	$1,584
Year two onward	$4,380	$384

Year one cost 64% less and every year after that 91.2% less, a recurring saving of $3,996, with hardware paid back in 3.6 months.

Most of the saving was still ahead of them. The business was opening toward 20 stores, and the recurring cost does not grow with the store count: every plan includes 50 of these devices, and even the doubled fleet, one Pi per store plus the monitoring server, would use 21 of them.

ESTATE SIZE	OLD MODEL, PER YEAR	NEW MODEL, PER YEAR
10 stores	$4,380	$384
20 stores	$8,760	$384

MEASURE	BEFORE	AFTER
Config to add a site	Firewall policy, customer gateway, VPN connection, routing both ends	One inventory entry
Change windows to add a site	One at the store, one in AWS	None
Cutover time per site	n/a	Minutes, with the build done in advance
Inbound listeners per store	One IPSec endpoint	None
Connection limit	10 per gateway, 50 per region by default	50 devices included; 21 used at 20 stores

The Nagios EC2 instance costs the same as before, the SonicWall firewalls stayed in place doing store firewalling, and monitoring traffic was small enough that data transfer never counted either way. None of those appear as savings above.

LESSONS What to take from this

An hourly-billed link costs the same regardless of what crosses it, which makes monitoring-only links worth auditing in any business with more than a handful of sites. The limits a provider sets, its service quotas, deserve the same attention as the invoice: this retailer was already at one with plans to double, and that appeared on no report anyone was reading. And check that each site's connection is direct. A network like Tailscale keeps working when direct paths fail by sending traffic through a shared relay server, and it does this with no warning; when every site is riding the same relay, one interruption there takes them all down at once.

If you suspect you are paying for connectivity that is oversized for what it carries, we can help you find out. Get in touch at anobytes.com.

REF References

AWS: Site-to-Site VPN pricing (connection-hour and data transfer rates) · `aws.amazon.com/vpn/pricing`

AWS: Site-to-Site VPN quotas (connections per virtual private gateway, gateways per Region) · `docs.aws.amazon.com`

AWS: Transit Gateway pricing (per-attachment hour and data processing) · `aws.amazon.com/transit-gateway/pricing`

Tailscale: pricing, plans, and included resource allowances · `tailscale.com/pricing`

Tailscale: subnet routers, route advertisement, and source NAT behavior · `tailscale.com/kb`