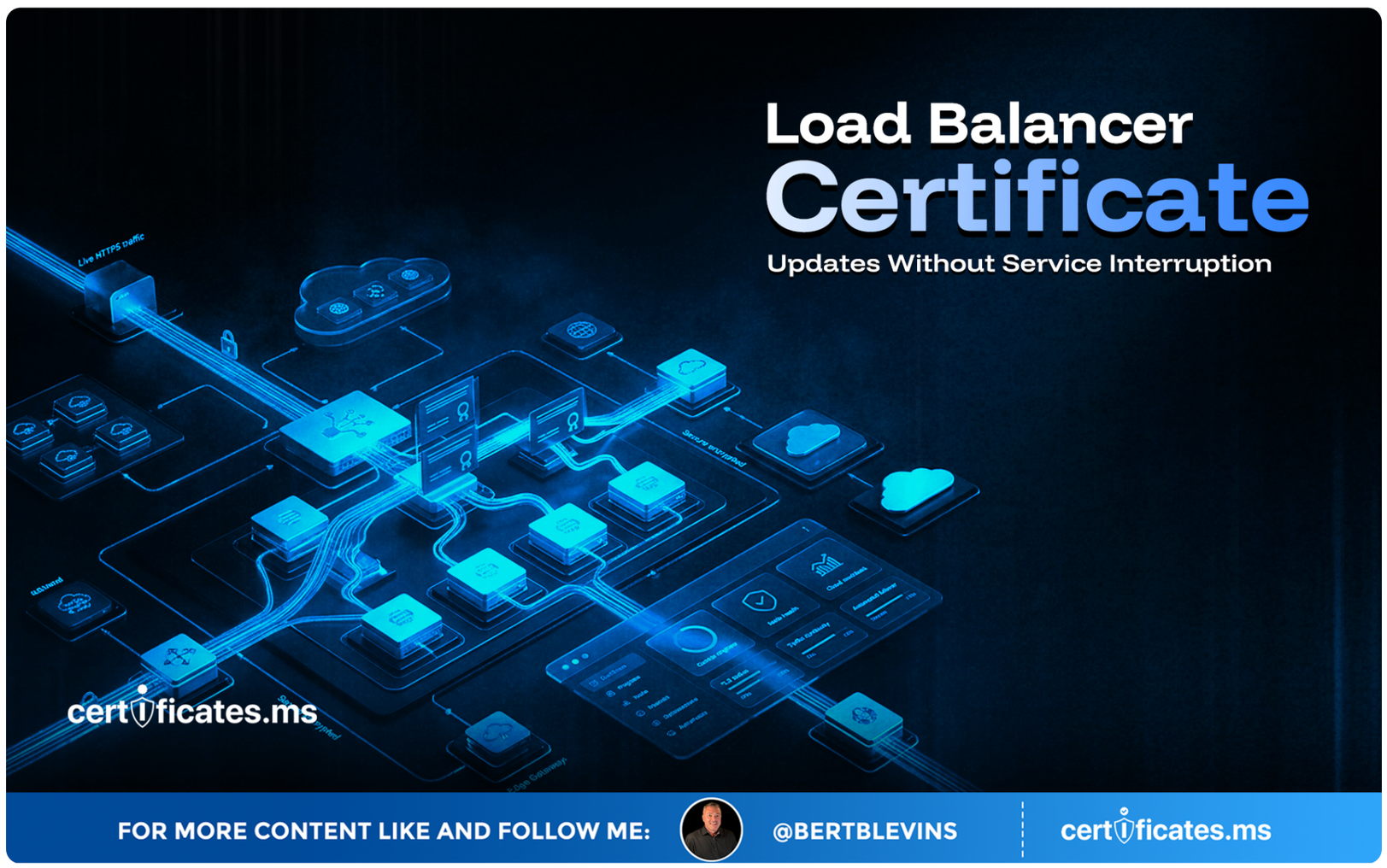
Load Balancer SSL Termination:

Certificates Done Right



Load balancers occupy a unique position in most modern application architectures: they are frequently the single place where encrypted traffic from the outside world gets decrypted before being distributed to backend servers. That makes their certificate configuration disproportionately important. Get it wrong, and every application behind the load balancer inherits the weakness. This article covers what SSL termination at the load balancer actually involves and the practices that keep it secure.

What SSL Termination Actually Means

SSL termination, more accurately TLS termination given that SSL itself has been deprecated for years, refers to the load balancer decrypting incoming encrypted traffic and then either passing it to backend servers as plain, unencrypted traffic over a trusted internal network, or re-encrypting it for the trip to the backend, a pattern known as re-encryption or TLS bridging. Terminating TLS at the load balancer centralizes certificate management in one place rather than requiring every backend server to hold and manage its own certificate, which considerably simplifies operations at scale.

Terminate, Passthrough, or Re-Encrypt

Organizations generally choose among three models. Full termination decrypts at the load balancer and sends plain traffic to the backend, which is simplest but assumes the internal network between load balancer and backend is genuinely trusted. TLS passthrough forwards encrypted traffic straight to the backend without the load balancer ever decrypting it, useful when end-to-end encryption is a strict requirement but limiting the load balancer's ability to inspect or route based on content. Re-encryption terminates at the load balancer and then establishes a fresh encrypted connection to the backend, giving the best of both worlds at the cost of additional certificate management on the backend side and some performance overhead. Zero trust architectures increasingly favor re-encryption or passthrough, since assuming an internal network is automatically safe has become a much riskier bet than it once was.

Certificate Placement and SNI

Modern load balancers typically handle certificates for many domains simultaneously using Server Name Indication, which allows the client to specify which hostname it is connecting to during the TLS handshake itself, before encryption is even established, so the load balancer can present the correct certificate for that specific domain. Properly configuring SNI-based routing avoids the older, more limited practice of dedicating a separate IP address to every certificate, and is standard practice for any load balancer serving multiple domains or a multi-tenant application.

Automating Certificate Deployment Across the Fleet

A single load balancer might be simple to manage manually, but most production environments run redundant load balancers across multiple availability zones or regions, all needing the same certificate deployed consistently and simultaneously. Manual deployment across a fleet invites drift, where one instance ends up running a stale certificate while others have renewed. Automated deployment pipelines, integrated with the same certificate issuance and renewal automation covered elsewhere in this series, should push updated certificates to every load balancer instance atomically, verifying successful deployment across the whole fleet rather than assuming a single successful update means the job is done everywhere.

Health Checks and Graceful Certificate Rotation

Rotating a certificate on a live load balancer serving production traffic needs to happen without dropping active connections. Most modern load balancers support hot certificate reloading, swapping the certificate in memory without restarting the service or interrupting in-flight requests. Deployment automation should include a verification step confirming the new certificate is actually being served correctly, catching a failed or partial rotation before it becomes a customer-facing outage rather than after.

Load Balancers Fronting AI Inference Endpoints

A growing number of load balancer deployments today exist specifically to distribute traffic across AI model inference endpoints, routing requests from applications and agents to whichever backend instance has capacity. These endpoints often see far higher request volumes and more automated, machine-generated traffic than a typical human-facing web application, which puts additional pressure on certificate infrastructure that was originally designed around more predictable, human-paced traffic patterns. Ensuring TLS termination scales cleanly with this kind of bursty, AI-driven request volume, without becoming a bottleneck or a point of certificate-related failure, has become a real design consideration for teams building AI-serving infrastructure.

The Countdown Is Already Running: 200 Days, 100 Days, 47 Days

Every certificate conversation in 2026 eventually arrives at the same clock, and it is worth closing on it here. The CA/Browser Forum's Ballot SC-081v3 is not a proposal under discussion; it is an approved, already-in-motion schedule. Maximum public TLS certificate lifetimes fall from 398 days to 200 days on March 15, 2026. They fall again to 100 days on March 15, 2027. By March 15, 2029, they drop to just 47 days, with domain validation itself needing to be re-proven roughly every 10 days.

01

Translate that into operational terms and the picture gets stark quickly. An organization currently renewing certificates a few times a year will be handling renewal events on the order of every couple of weeks by the end of this countdown, across every endpoint it operates. Manual tracking, calendar reminders, and a spreadsheet somebody checks once a month will not survive contact with that cadence. What has always been an occasional chore is becoming a continuous, automated operation, whether an organization plans for it or not.

02

Because a single load balancer certificate often protects traffic for an entire fleet of backend applications, keeping that certificate's renewal fully automated matters even more once the industry-wide schedule below compresses renewal cycles down toward every 47 days.

03

The 200-day, 100-day, and 47-day milestones are not distant hypotheticals; the first has already arrived. Organizations that build the automation loop now, generating keys, vaulting them securely, brokering issuance across Certificate Authorities through APIs, and rebinding certificates to live endpoints without manual intervention, will meet each deadline without disruption. Organizations that wait will be rebuilding their certificate operations under deadline pressure, with far less room for error and far less time to get it right. The countdown is the call to action. The only real decision left is whether to automate on your own schedule, or on the CA/Browser Forum's.