ingress-nginx Retirement: Four Ways Forward

2026-07-18

ingress-nginx is retired, not retiring — the GitHub repo was archived in March 2026. What changed, the CVEs disclosed since, and four honest ways forward.

The `ingress-nginx` Kubernetes controller is not "retiring soon." It already has. The GitHub repository was archived and made read-only on 2026-03-24, and its own description now reads in the past tense: it *was* an Ingress controller. For a piece of software that reportedly still runs in front of roughly half of all cloud-native workloads, that is a completed fact, not a countdown.

This is Part 1 of a field guide to migrating away from it. This part is deliberately non-technical — it is for whoever has to decide *what to do*, not yet for whoever will do the migration work. Parts 2 onward get concrete: inventorying what you actually depend on, an annotation compatibility matrix, migration mechanics, and a reproducible lab.

> **► TL;DR**
>
> `ingress-nginx` is archived, not just deprecated. No more releases, bugfixes, or security patches are coming. Existing clusters keep running unchanged — nothing breaks on its own — but every new CVE from here on goes unpatched. Four honest ways forward exist: keep running it and accept the risk, switch to another Ingress controller, move to Gateway API, or use a maintained fork as a bridge while you decide. None of them is free, and none of them is obviously wrong for every situation.

What's actually retired, and what isn't

The retirement announcement is specific about scope, and the distinction matters — it is easy to overstate what happened here.

Retired?	Item	Meaning
Yes	`ingress-nginx` controller	Archived read-only on GitHub since 2026-03-24. No further releases, bug-fixes, or security patches.
No	Kubernetes Ingress API	Still part of Kubernetes; other controllers keep implementing it.
No	NGINX (the software)	Unaffected — `ingress-nginx` was one controller built on top of it, not NGINX itself.
No	NGINX Gateway Fabric	A separate, actively maintained F5/NGINX project implementing Gateway API. Not the retired controller, despite the similar name.
No (separate project)	Chainguard EmeritOSS fork	A maintained fork for stability and CVE backports, started in response to the retirement — not upstream `ingress-nginx`, and not framed by its maintainers as a permanent substitute.

Existing deployments are not broken by any of this. Helm charts and container images already published remain available. What stops is everything forward-looking: no new releases, no bug-fixes, and — the part that actually matters for anything internet-facing — no fixes for whatever security issues get found next.

Why "it still works" isn't the right test

The Kubernetes Steering Committee and Security Response Committee's statement is unusually direct about why this is happening: `ingress-nginx` had been maintained by one or two people, unpaid, for years, and the accumulated technical debt and security-relevant design choices (the `snippet` annotations chief among them) meant continued maintenance was "no longer reasonable or even possible… even if resources did materialize." That is a statement about the software itself, not just about funding — the committees are explicit that this isn't reversible by simply finding a new maintainer.

The retirement has already produced a concrete example of the risk. On 2026-02-02, four CVEs were disclosed against `ingress-nginx` in a single batch — two of them HIGH severity (CVSS 8.8 each, per NVD): a configuration-injection issue that can lead to remote code execution and secret disclosure (CVE-2026-24512), and an authentication-bypass issue affecting the `auth-url` annotation (CVE-2026-1580); plus CVE-2026-24513 and CVE-2026-24514, rated low (3.1) and medium (6.5) severity respectively. Fixes existed for these because they landed before the archival date; the same disclosure arriving today would not get one.

There is also a compliance angle worth naming plainly, even without a single canonical source to point at: end-of-life software sitting in the request path is exactly the kind of thing SOC 2, PCI-DSS, ISO 27001, and HIPAA audits are designed to flag. If your organization is subject to any of those, "it still works" is unlikely to be the question your auditor asks — see this blog's own compliance

deep dive for how PCI-DSS, HIPAA, NIS2, DORA, and CRA controls actually map onto Kubernetes configuration.

Risk timeline

The sequence is short and, at this point, entirely in the past:

1. **2025-11-11** — Kubernetes announces the retirement (best-effort maintenance to continue until March 2026).
2. **2026-01-29** — Steering Committee and Security Response Committee issue a follow-up statement reinforcing the decision and its finality.
3. **2026-02-02** — Four CVEs disclosed in a single batch, two of them HIGH severity.
4. **2026-03-24** — The `kubernetes/ingress-nginx` repository is archived and made read-only.

Nothing in that list is upcoming. The only future-tense item left is whatever the *next* unpatched vulnerability turns out to be.

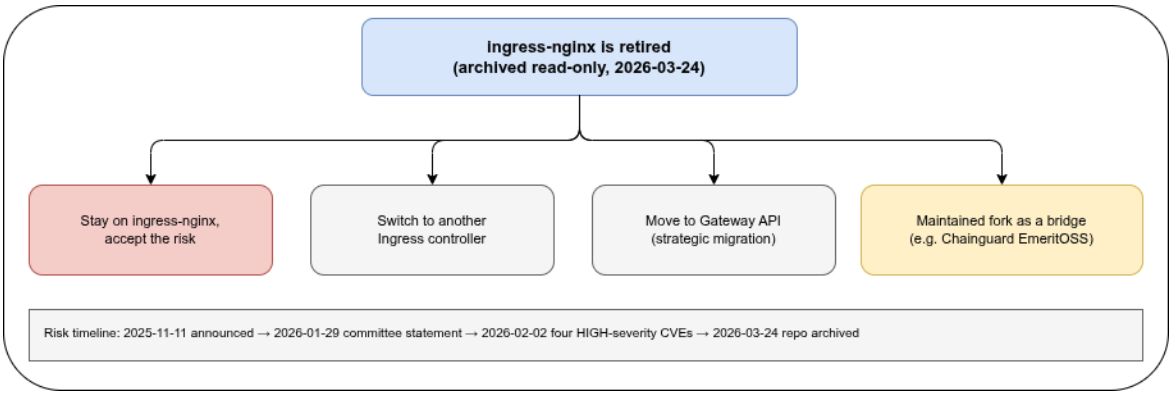

Figure 1: ingress-nginx retired leads to four options: stay and accept risk, switch controller, Gateway API, or a maintained fork

Four honest paths forward

None of these is presented as *the* answer — the right one depends on your annotation usage, your team's capacity, and your risk tolerance, all of which Parts 2 onward help you actually measure instead of guess at.

1. **Stay on `ingress-nginx` and accept the risk.** Valid as a short-term position if you have genuinely assessed the exposure — internal-only traffic, a short runway to a planned migration, or a compensating control like a WAF in front of it. It stops being valid the moment it becomes the default because migrating felt like too much work.
2. **Switch to another Ingress controller.** The lowest-effort move if you are already committed to the Ingress API and just need a maintained implementation. Part 4 of this series looks at the concrete candidates and what actually carries over from your existing `nginx.ingress.kubernetes.io/*` annotations.
3. **Move to Gateway API.** The strategic option — it is where Kubernetes networking is headed, and where every major implementation (Envoy Gateway, Cilium, Istio, the cloud providers' managed offerings) is investing. It is also the option with the most migration mechanics to get right, which is what Part 5 is for.

4. **Use a maintained fork as a bridge.** Chainguard's EmeritOSS program forked `ingress-nginx` specifically for this transition: the free fork on GitHub gets dependency updates and best-effort CVE fixes but no new features and no pre-built images; a commercial tier adds pre-built container images, a remediation SLA, and a FIPS variant. Either way, its own maintainers frame it as buying time to migrate deliberately, not as a permanent substitute.

> ⚠ **What a fork does not solve**
>
> A maintained fork removes the "no more CVE fixes" problem for as long as it's maintained. It does not remove the underlying design issues the retirement statement cited, and it does not move you any closer to Gateway API if that is where you eventually need to land. Treat it as time bought, not risk eliminated.

What the rest of this series covers

This part answered one question: do you need to act. The honest answer for almost everyone running `ingress-nginx` on internet-facing infrastructure is yes, on some timeline that you get to set — but not one you get to ignore indefinitely. The series hub tracks what's published so far and what's still planned; Part 2 starts the practical work with an inventory of what your own clusters actually depend on before you touch anything.

Sources

- Ingress NGINX Retirement: What You Need to Know — kubernetes.io (2025-11-11)
- Ingress NGINX: Statement from the Kubernetes Steering and Security Response Committees — kubernetes.io (2026-01-29)
- kubernetes/ingress-nginx — GitHub (archived 2026-03-24)
- Fork Yeah: We're keeping ingress-nginx alive — Chainguard (2026)
- Introducing Chainguard EmeritOSS: Sustainable stewardship for mature open source — Chainguard (2026)
- Gateway API — Kubernetes SIG Network