KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

LTS, EOL, CVEs: Solutions for Stress-Inducing Scenarios

Lachlan Evenson & Bridget Kromhout, Microsoft



Who are you two?



Lachlan Evenson

Product Manager @ Microsoft Azure; Kubernetes Steering Emeritus; CNCF GB

Originally from Australia; now lives in San Francisco. Likes hiking, baseball, cricket and language.

GitHub & socials: lachie83



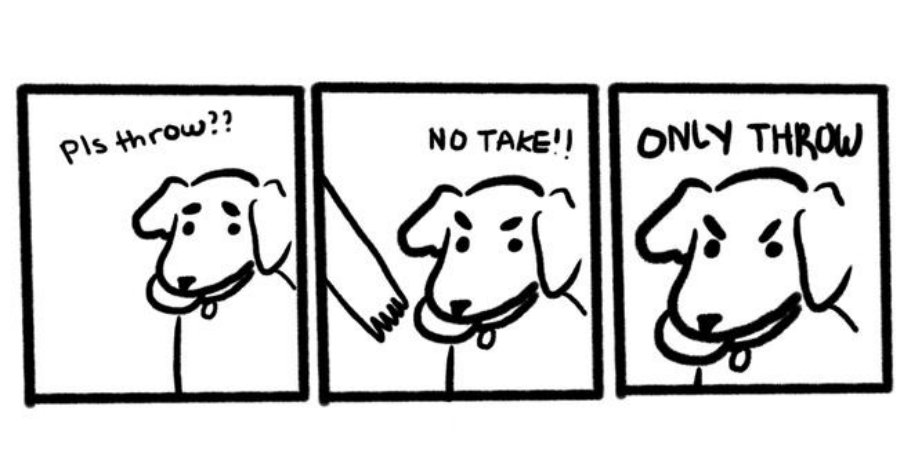
Bridget Kromhout

Product Manager @ Microsoft Azure; SIG Cloud Provider co-chair; CNCF GB alternate

Originally from Saint Paul; now lives in Minneapolis. Likes gardening, bicycles, snow, and cats.

GitHub & socials: bridgetkromhout

Change versus stability: revisiting assumptions



LTS The latest features versus long-term support

EOL Shiny new releases versus shiny new bugs

CVEs Patches for CVEs versus hardened images

LTS: When should you change?

What we once thought

- Why would Kubernetes need an LTS?
- If the APIs are the contract points, why does Kubernetes need versions?

What we came to understand

- Nobody wants disruption, and upgrades have a reputation of being disruptive
- K8s famously had breaking changes in each release!
- LTS is for those trading broad choices for a longer timespan

Actualfax Reality

- Delaying upgrades limits choices (and means version incompatibilities)
- What resiliency can we build in to prevent disruption?
- Kubernetes WG-LTS focused on improving upgrades

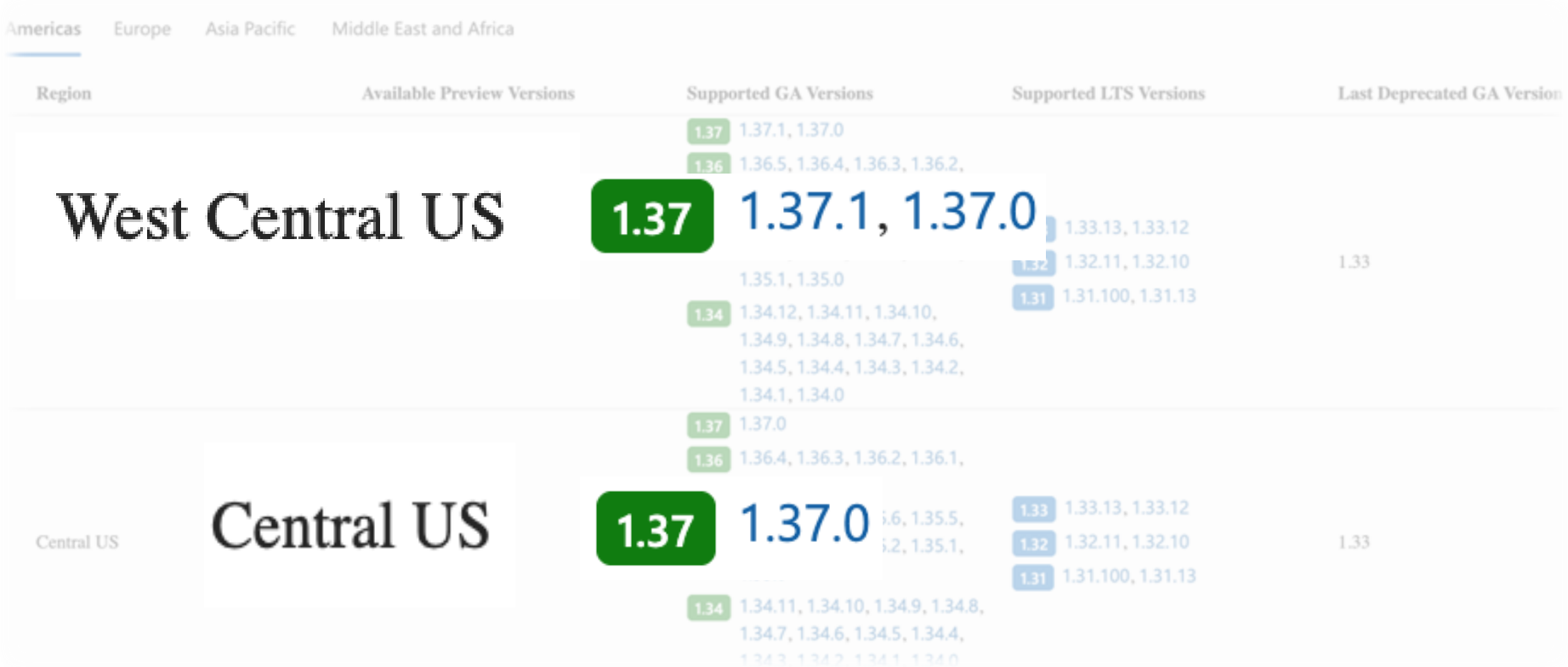
The speed of light, and other non-negotiable realities



Americas | Europe | Asia Pacific | Middle East and Africa

Region	Available Preview Versions	Supported GA Versions	Supported LTS Versions	Last Deprecated GA Version
West Central US	-	1.37 1.37.1, 1.37.0 1.36 1.36.5, 1.36.4, 1.36.3, 1.36.2, 1.36.1, 1.36.0 1.35 1.35.9, 1.35.8, 1.35.7, 1.35.6, 1.35.5, 1.35.4, 1.35.3, 1.35.2, 1.35.1, 1.35.0 1.34 1.34.12, 1.34.11, 1.34.10, 1.34.9, 1.34.8, 1.34.7, 1.34.6, 1.34.5, 1.34.4, 1.34.3, 1.34.2, 1.34.1, 1.34.0	1.33 1.33.13, 1.33.12 1.32 1.32.11, 1.32.10 1.31 1.31.100, 1.31.13	1.33
Central US	-	1.37 1.37.0 1.36 1.36.4, 1.36.3, 1.36.2, 1.36.1, 1.36.0 1.35 1.35.8, 1.35.7, 1.35.6, 1.35.5, 1.35.4, 1.35.3, 1.35.2, 1.35.1, 1.35.0 1.34 1.34.11, 1.34.10, 1.34.9, 1.34.8, 1.34.7, 1.34.6, 1.34.5, 1.34.4, 1.34.3, 1.34.2, 1.34.1, 1.34.0	1.33 1.33.13, 1.33.12 1.32 1.32.11, 1.32.10 1.31 1.31.100, 1.31.13	1.33

The speed of light, and other non-negotiable realities (zoomed in)



Upgrade risks and timelines you didn't choose

Kubernetes upgrade mandates from your managed service could look something like...

"A platform-initiated upgrade doesn't follow your maintenance window, doesn't stop for deprecated API usage, and can override a Pod Disruption Budget that blocks a node drain, so workloads might be disrupted."

What does stable even mean?

Why does software go EOL?

- Planned (Open Service Mesh was archived with a path to replacement with Istio)
- Sudden (ingress-nginx urgently needed replacement while Gateway API was still in progress)

What is stability?

- Stable releases include assumptions about lifecycle
- Period of support, or contract around commitments to APIs

Unexamined risk

- Load-bearing components we don't consider are a source of unexamined risk
- Part of a system might go EOL before the rest can be updated

The trouble with forking

Patch ingestion

- Forks can limit the ability to ingest necessary patches
- Trade-offs between stability and usability
- Incompatibilities in code or licensing

Hard forks

- Hard fork continuity planning
- Evaluate what you lose in a hard fork
- Azure de-forked Cluster Autoscaler after multiple years

Soft forks

- If you can't upstream your changes immediately, can you soft-fork?
- Can you make upstream modular and extensible?
- Instead of forking, Azure team members became upstream etcd maintainers

How true is that vuln report?

CVE zero

- Is the goal to be at CVE zero?
- What does "CVE zero" actually mean?
- Is CVE zero a distraction?

Quieting scanners

- Is the goal a green dashboard?
- Is it an assertion about safety?
- Iterative improvements vs perfection which cannot last

The race against exploits

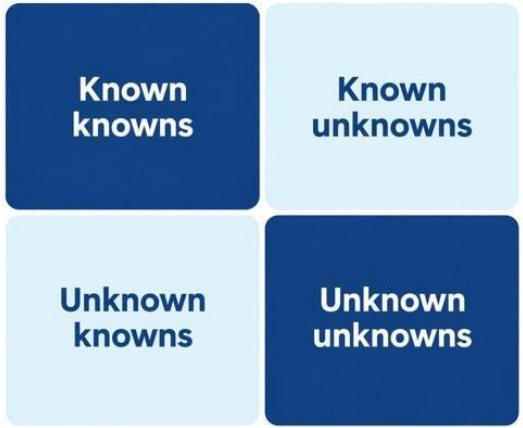
- If not patched, do you assume you're exploitable?
- Can you patch at any time?
- Are you happy with your intake of change and rollout of updates?

Responsible disclosure: what is known?

Embargoes — Cooperation and trust across teams.

Coordinated disclosure — Works best where collaboration is in place.

Unknown unknowns — We don't know what we don't know.



Maintainability concerns

Clean-room rewrites	Re-implementation of open source projects carries risks, much like a hard fork of a project; future patching is uncertain.
No upstream patches	Maintainability risk if upstreams aren't patching anymore; AI-fueled zero-days can appear faster than AI-assisted patches
Code no one understands	Drive-by PRs with generated code that nobody (including the submitter) understands.

Open source tools for build security



- Build packages and containers for multiple Linux distributions and Windows, with a focus on supply chain security
- Minimal image size, fewer vulnerabilities, smaller attack surface

https://github.com/project-dalec/dalec



- Directly patch container images without full rebuilds
- Optionally informed by vulnerability scanning results

https://github.com/project-copacetic/copacetic

Maintainability options

Testing & provenance

- Testing & issue remediation at scale.
- Code provenance disclosure & attribution for responsible AI.

Broaden contributor base

- Broaden pool of contributors to de-risk maintainer churn.
- The value of open source is a broad community with many eyes.

Collaborate in CNCF

- Load-bearing open source exists outside the current zeitgeist.
- Low cost to fork, but we create more value by collaborating.

The case for change

Good decisions

- There is more code than ever.
- We need to make good decisions about that code.

Rubric for choices

- What happens if we do nothing?
- How do we preserve optionality while protecting trust?
- How do we anticipate the consequences of our actions?

Change isn't optional

- Deciding not to change is still a choice.
- Inaction lets someone else choose changes that will affect you.

What's next? You decide!

KubeCon CloudNativeCon North America 2026

Evaluate competing options

Ask for stakeholder feedback

Work with your community

Alpha-Omega

OpenSSF
OPEN SOURCE SECURITY FOUNDATION



KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

Thanks for joining LTS, EOL, CVEs: Solutions for Stress-Inducing Scenarios

Feedback welcome!

LTS, EOL, CVEs: Solutions for Stress-Inducing Scenarios

Lachlan Evenson & Bridget Kromhout, Microsoft

