KubeCon CloudNativeCon North America 2026

containerd: Project Update and Deep Dive

Wei Fu, Microsoft
Samuel Karp, Google



KubeCon CloudNativeCon North America 2026

Release Cycle & LTS



Revisiting containerd's Release Strategy

Drivers for Change

Challenges with Prior 6-Month Cadence

- Misalignment with Kubernetes 4-month cadence
- Feature delivery delays from schedules
- Unclear guidance for upgrades

Objectives

- Day-one validation for new Kubernetes Enhancement Proposals (KEPs)
- Clear upgrade horizons
- Well-defined support boundaries

Release Cadence Improvements

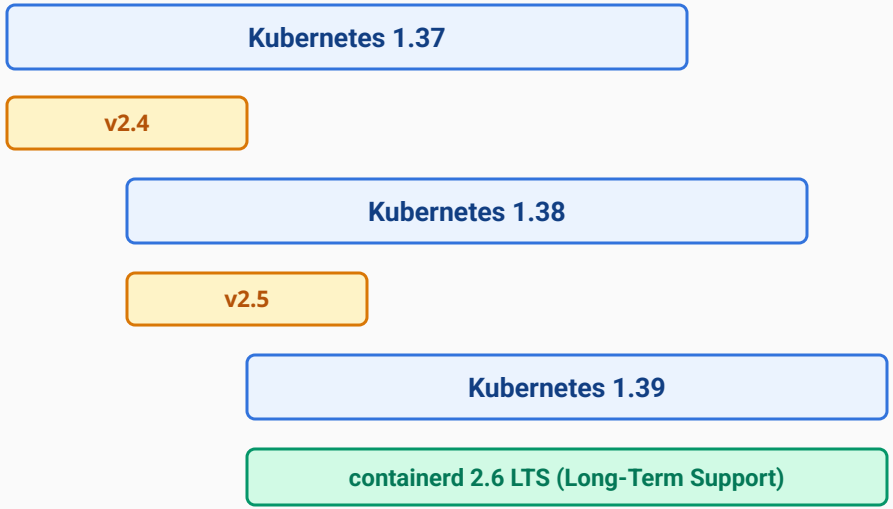
4-Month Cadence

Rapid Feature Delivery

Kubernetes Alignment

Long Term Support

Kubernetes & containerd LTS Alignment



Predictable Timing

Constant sequence:
Regular → Regular → LTS (e.g. 2.3, 2.6, 2.9).

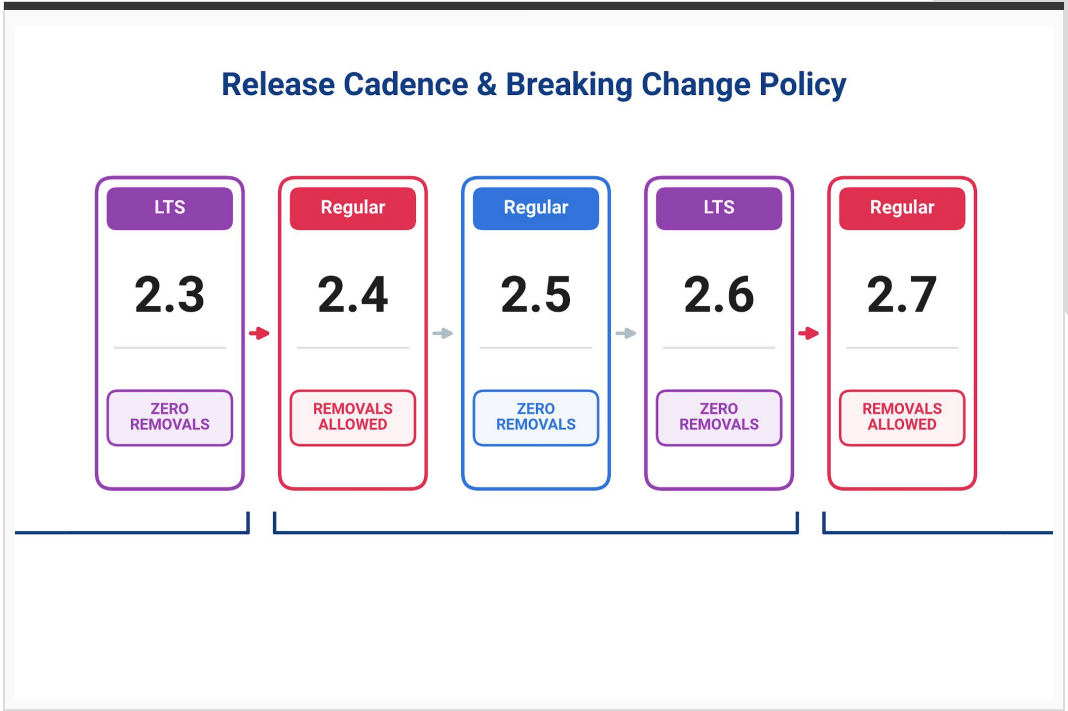
2-Year Support Window

1-year overlap between LTS releases

Kubernetes & Cloud Parity

Matches major cloud provider enterprise support windows

Deprecation & Removal Policy for Safe Upgrades



1-Release Deprecation Minimum

Programmatic deprecation warnings before removal.

Safe Path Into LTS

No removals or breaking changes leading up to an LTS, ensuring a safe upgrade path into it.

Strict Post-LTS Removals

Removals occur only in the release immediately following an LTS (e.g., 2.4 after 2.3).

Consecutive LTS Skip-Upgrades & User Journeys

Dual User Personas

- **Feature-driven users:** Upgrade quickly for access to cutting-edge features.
- **Stability-driven users:** Stay on LTS releases only.

Tested Skip-Upgrade Paths

- **Consecutive LTS jumps:** Direct, CI-tested paths (e.g., 2.0 → 2.3 LTS).
- **Caveat resolution:** Resolve deprecation warnings before jumping.

Up to 32 Months of Upgrade Safety

- **Low-friction adoption:** Smooth transition from regular releases to next LTS.
- **Extended lifecycle:** Up to 32 months of support without breaking changes.

LTS Backporting Guardrails

- **Bug & security fixes only:** No feature backports to active LTS releases.
- **No CRI backport pressure:** Synced cadence eliminates backporting new CRI APIs.



What's new in containerd 2.4

CRI & Kubernetes in containerd 2.4

KubeCon CloudNativeCon North America 2026

Safer pods on any runtime, denser nodes, and data shipped as images.

SHIPPED IN v2.4.0

UserNS + hostNetwork

K8s 1.37 alpha

#13162

BEFORE

hostNetwork pods: pod root = host root

hostNetwork + hostUsers: false

Pod root is unprivileged on the host

Runtime Feature Discovery

#13504

BEFORE

Only runc reported features; other runtimes looked unsupported

Every runtime reports its features

kubelet sees userns / mount support per handler

CRI List Streaming

K8s 1.37 alpha

#13187

BEFORE

One giant List reply hits kubelet's 16 MiB gRPC cap

Streamed in fixed 5,000-item pages

No response-size ceiling on dense nodes

OCI Image Volumes

#13542

BEFORE

Model weights baked into the app image

Weights as a read-only OCI image

Versioned apart from the app · now works on EROFS

Also in 2.4: CDI always on · NRI gets image name + digest · /proc/interrupts masked · CRI info exports sandbox image + CNI dirs

Image distribution improvements

Bounded memory, faster decoding, complete content and reusable layers.

SHIPPED IN v2.4

Bounded Image Fetching

v2.4.2

#14311

BEFORE

Slow early chunks let unread buffers pile up

Memory bounded per layer

A chunk slot frees only after the reader consumes it

Faster Gzip Decoding

#13560

BEFORE

In-process fallback used Go stdlib gzip

1.3–1.5x faster gzip decode

klauspost/compress, no external tools needed

Fetch All Content

#14126

BEFORE

GC'd layer blobs blocked export and push

Opt-in: fetch missing layer blobs

Keep snapshots, become export / push ready

EROFS Layer Reuse

#13813

BEFORE

Every node re-converts tar layers to EROFS

Prebuild once, reuse by diffID

Warm cache skips fetch, convert and layer writes

Bound buffers, not just concurrency

KubeCon CloudNativeCon North America 2026

Parallel downloads. Ordered reads. Slow early chunks stall consumption.

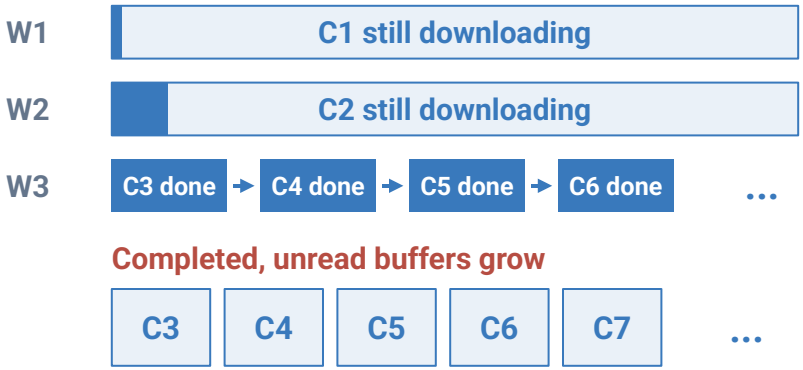
SHIPPED IN v2.4.2

#14311

One layer, 3 workers; C1 and C2 download slowly.

Before: download limits only

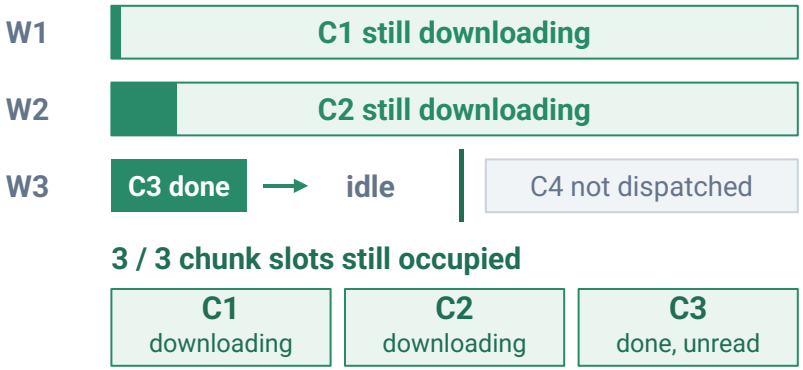
Fast worker keeps fetching the next chunk.



Worker freed. Buffer not consumed.

After: bound outstanding chunks

A slot is released only after reader EOF.



C1 read to EOF -> slot returned -> C4 starts

Bound outstanding chunks, not just active downloads.

Gzip gets a faster in-process decoder

KubeCon CloudNativeCon North America 2026

GZIP

v1.2.0
Optional unpigz
External process
unpigz CLI, when available
PR #2640

v2.0.0
ISA-L igzip
External process
igzip CLI, when available
PR #9200

v2.4.0 IN-PROCESS
Faster in-process gzip
Go stdlib -> klauspost/compress

amd64	arm64
1.29x	1.53x
gzip decode vs Go stdlib	#13560

2018 — 2021 — 2024 — 2025 — 2026

ZSTD
Alternative codec

v1.5.0
zstd support
A faster-codec option
PR #4809

v2.1.0
Decoder tuning
Disable lowmem; less copying
PR #11335

Decode throughput vs Go stdlib: amd64 214 -> 276 MB/s (+29%) · arm64 255 -> 391 MB/s (+53%)

Single-stream decode, not a total image-pull speedup. Default selection unchanged: igzip -> unpigz -> in-process.

Gzip decoder benchmarks: two native platforms

Two independent runs. Ratios compare decoders within each platform, on the same input.

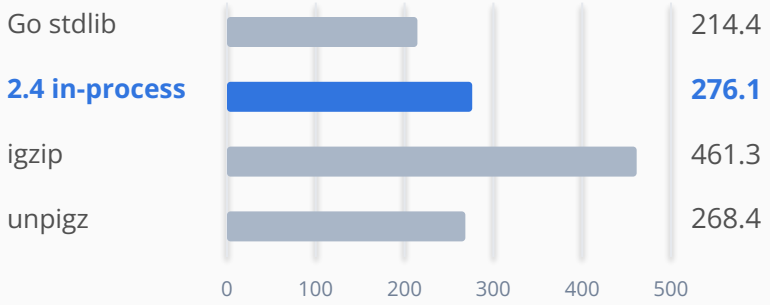
amd64 | Intel Xeon 8573C

1.29x

Standard_D16ds_v6 | 16 vCPU, 8 cores x 2 threads

2.126 GB gzip -> 4.582 GB decoded

Decoded MB/s (higher is better)



Speedup badge: 2.4 in-process / Go stdlib

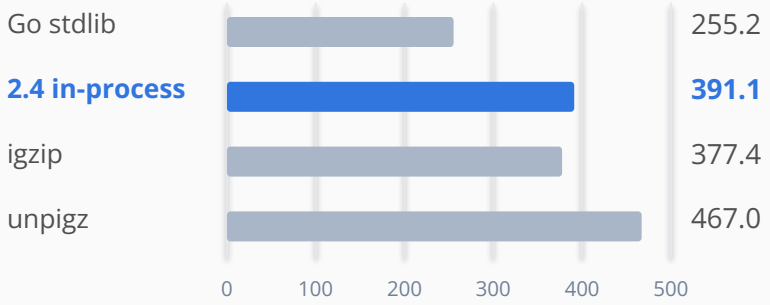
arm64 | Azure Cobalt 100

1.53x

Standard_D16ps_v6 | 16 vCPU, 16 cores x 1 thread

1.932 GB gzip -> 3.651 GB decoded

Decoded MB/s (higher is better)



Speedup badge: 2.4 in-process / Go stdlib

6 samples per case | one stream per decode | compressed input in memory | output discarded

v2.4.0 + supplied benchmark patch; configured Go 1.26.6, GOMAXPROCS=16; igzip 2.30.0, pigz 2.6.

Different input layers and CPU topologies: compare decoders within each panel, not CPUs across panels.

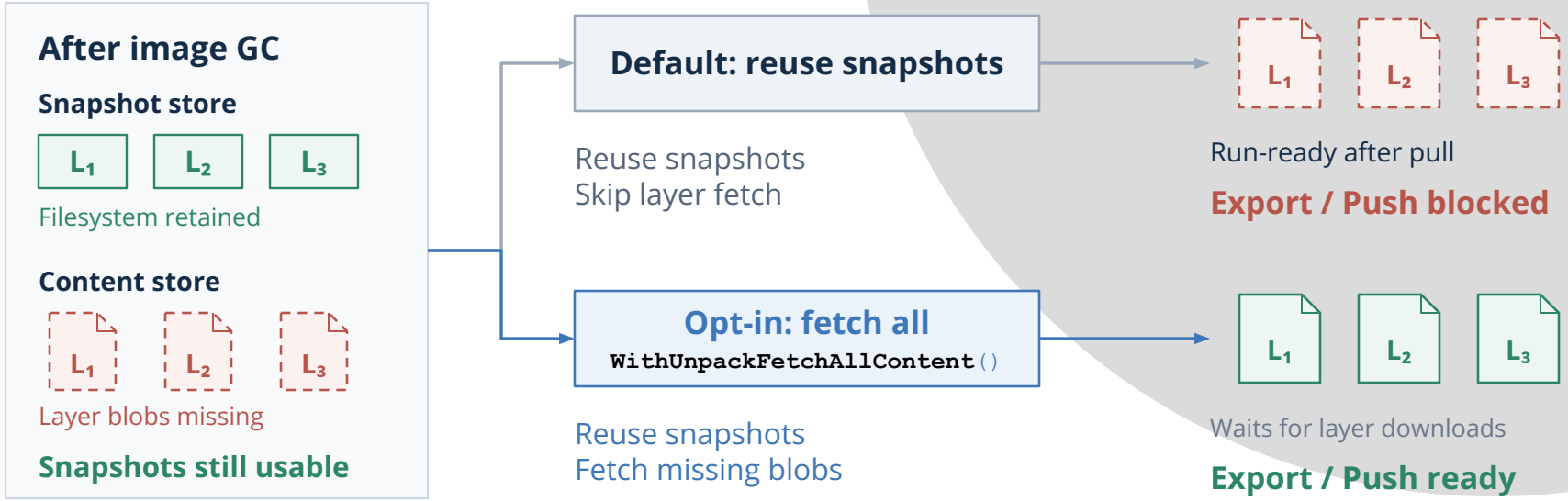
Source & setup: https://gist.github.com/fuweid/9b2de1ec6a4822f2681f56727faeadfb

Run-ready is not content-complete

Pull + unpack: snapshots can outlive their layer blobs after GC.

SHIPPED IN v2.4.0

#14126



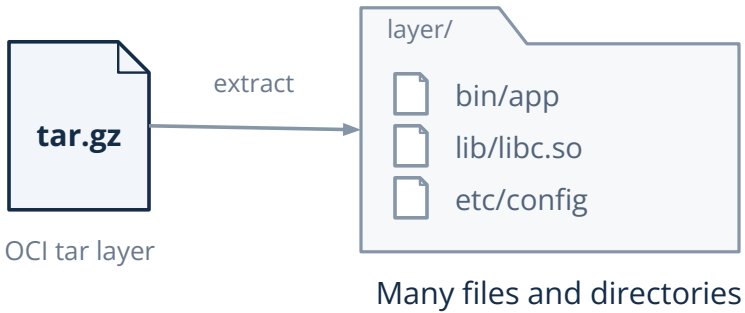
Fetch All Content: same snapshots, complete layer blobs.

Wait for all selected layers before reporting success.

EROFS: one file per layer

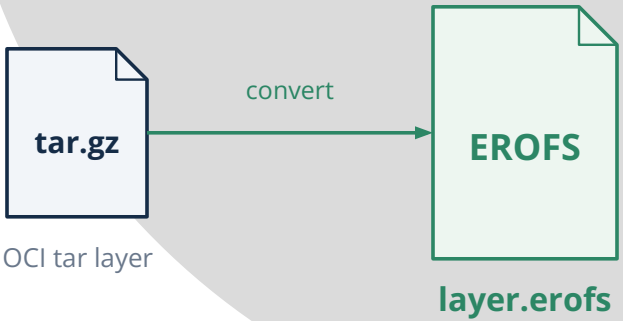
One EROFS file per layer can be prebuilt, persisted and reused by diffID.

Directory-based unpack



More filesystem metadata to write and clean

EROFS differ



Targeted fsync: layer file + directories

One file per layer: targeted fsync, and a layer you can build once and reuse.

Flush to disk: how the boundary changed

KubeCon CloudNativeCon North America 2026

Crash consistency requires data to reach disk before metadata.

Crash reports: committed metadata can outlive layer data — #5854 #14164

Unmount flush

Mount, apply,
then unmount

Flush is implicit

Direct unpack

No temporary mount
No unmount barrier

Writeback can lag

#3528

syncfs (opt-in)

Flush the backing filesystem

#9401 #9769

EROFS fsync

Flush layer files
+ snapshot directories

Before metadata commit

#14334

Before v1.3 — **2019 / v1.3.0** — **2024 / v1.7.14** — **2026 / v2.4.3 (opt-in)**

Same rule, finer boundary: persist data before metadata.

Write layer data → Flush data + directories → Commit metadata

Warmup: reuse layers, skip the rebuild

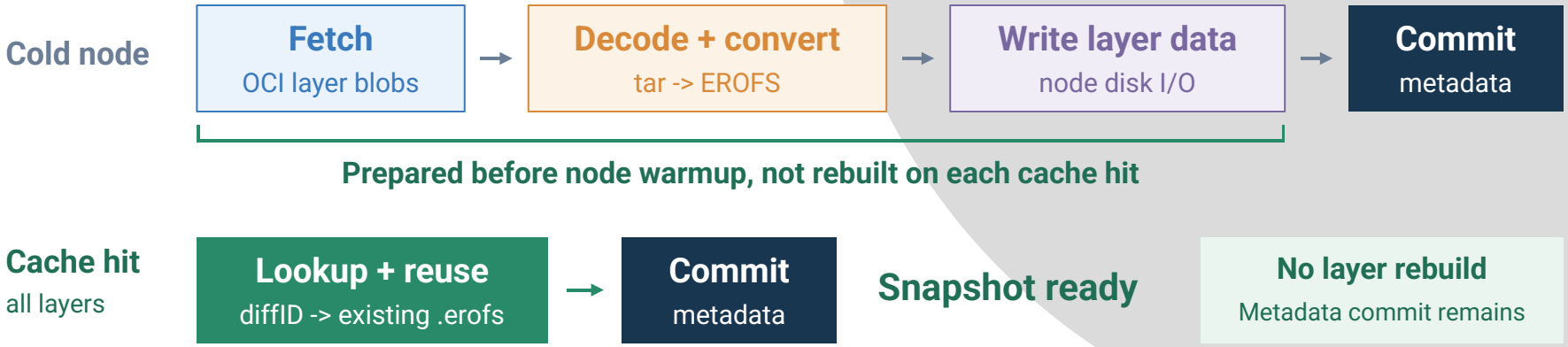
KubeCon CloudNativeCon North America 2026

All requested layers already exist in the prebuilt, read-only cache.

SHIPPED IN v2.4.0

#13813 #13826

Node-side layer preparation



Layer fetch skipped
Network: no OCI blob download

Decode / convert skipped
CPU: no tar-to-EROFS rebuild

Layer-data writes skipped
Disk: reuse existing layer files

Observability in containerd 2.4

KubeCon CloudNativeCon North America 2026

Trace context across processes. Metrics tied to actual image-path work.

SHIPPED IN v2.4.0

Trace context to runc + hooks

#14036

BEFORE

Traces stopped at the shim

W3C trace context reaches runc + hooks
Context propagation, not automatic spans

OpenTelemetry configuration

Breaking

#14166

BEFORE

containerd-specific OTLP and sampling keys

Standard OTEL_* environment variables
Old keys removed: migrate before upgrading

Image-pull throughput

#13245

BEFORE

Cached layers inflated pull MiB/s

image_pulling_throughput_mibps
Fetched bytes only; end-to-end incl. extraction

EROFS warm-cache visibility

#13941

BEFORE

Warm-cache hits were invisible

Prometheus metrics for the layer cache
Lookups, hits, bytes served and errors



Next up: containerd 2.5

Looking ahead to containerd 2.5

KubeCon CloudNativeCon North America 2026

Proposed scope, not a feature guarantee. PR status as of Nov 3, 2026.

TARGET: JAN 6, 2027

Sandboxed runc shim

Open PR

#14135

TODAY

Every pod runs a pause container

Pauseless pods

Sandbox API in the runc shim, no pause image

Runtime config directory

Open PR

#13103

TODAY

All runtime handlers live in one config.toml

One file per runtime handler

runc/runtime.toml, kata/runtime.toml, ...

Mount manager schema v2

Open PR

#13975

TODAY

Persisted mount state is trusted as-is

Shared mounts + state repair

Ref-count identical mounts, verify real mounts

Go EROFS conversion

Draft

#13589

TODAY

Conversion execs mkfs.erofs on the host

Tar -> EROFS in pure Go

Per-layer, flat merged or tar-index images; any OS

Go EROFS builds on tarconv, merged in containerd/continuity #289. Also exploring pidfd-based supervision; NRI 1.0 is tracked separately.

KubeCon CloudNativeCon North America 2026

Security deep dive



The 2026 Security Landscape

KubeCon CloudNativeCon North America 2026

Vulnerability Flood & LLM Capabilities

LLM Capability Shift

- Models build exploits without humans
- Low barrier to create “plausible” security reports

Impact on Maintainers

- High volume of invalid PoCs and hallucinations
- Diversion from core development
- More coordination and release work

Report Influx: 2020-2025 vs. 2026

36 reports / 19 vulns

2020-2025

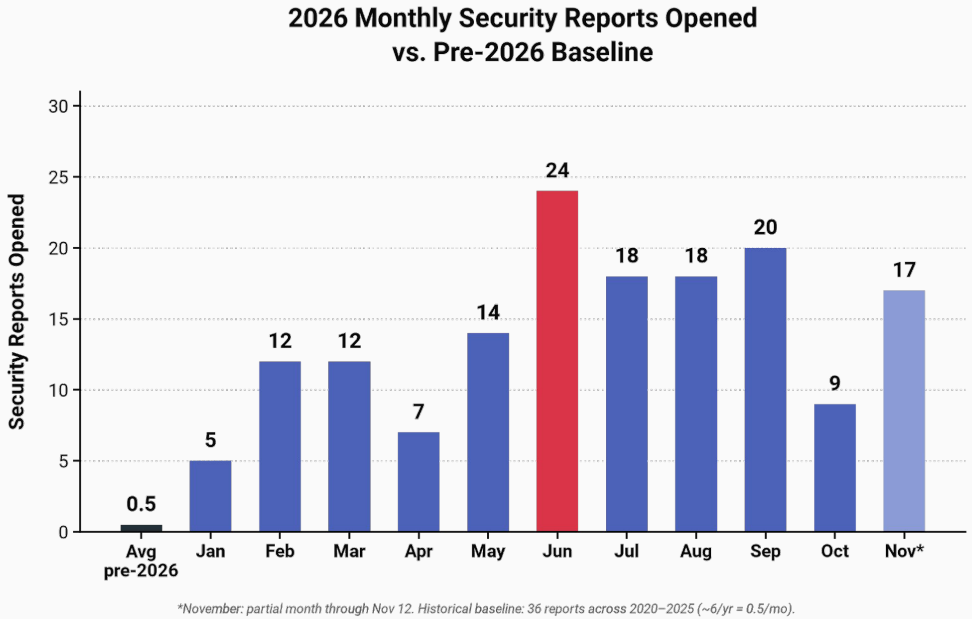
156 reports / 14 vulns

2026 alone (~85% Rejected)

+41%

Rejection Rate Increase

Monthly Influx: Sustained Surge Across 2026



*November: partial month through Nov 12. Historical baseline: 36 reports across 2020–2025 (~6/yr = 0.5/mo).

28.4x Monthly Influx Surge

2020-2025 baseline: **~0.5/month**
2026 average: **14.2/month**

Sustained rate

Feb–Nov influx: **7–24 reports/mo**
Peak: **24 reports** (June 2026)

Operational Triage Burden

Unvetted AI reports **diverting maintainer focus**

2026 Reports: Breakdown of Rejection Reasons

Across 2026, **~85%** of upstream GHSA reports were rejected and closed.

39%

DUPLICATE REPORTS

39%

NOT A VULNERABILITY

- Non-security bugs
- Theoretical flaws

22%

SCOPE & SPECIFICATION

- Out of scope
- Withdrawn
- Full hallucination
- Intended behavior

2026 Security Process Improvements

KubeCon CloudNativeCon North America 2026

Threat Model & Skills

- Published a **threat model** for containerd
- **Report-processing skills** for vulnerability triage phases and preparation
- LLM skill to **reduce release overhead**

Tooling & CI

- **Better GHSA browser extension** to enhance state tracking and coordination
- **Reduced CI friction** in private security forks via conditional GitHub Actions execution

Frontier Models & Programs

- Maintainer access to **frontier cyber models**, including Gemini Cyber, Anthropic CVP, and OpenAI Daybreak Blue
- Participated in the **OpenAI Patch the Planet** program

Security Partnerships

- Partnered with **security engineers** funded by LF Alpha-Omega and OSTIF Project V-LAT

containerd Security Patching Process

01 **Triage:** scope, audit for a valid PoC, reproduce

02 **Patching:** authorship, review, CI, and cherry-picking

03 **Advisory content:** title/description, CVSS scoring, package metadata, CVE ID assignment

04 **Embargo management:** if necessary, typically only for high/critical

05 **Prep patch releases:** up to 5 active releases may need a fix

Join the community!

- **GitHub:** github.com/containerd
- **Slack:** slack.cncf.io (Channels: `#containerd`, `#containerd-dev`)
- **YouTube:** youtube.com/@CNCFcontainerd
- **Bluesky:** @containerd.dev
- **Community meetings:** containerd.io/meetings

KubeCon CloudNativeCon North America 2026

Thank you!

Questions?

github.com/containerd/containerd

#containerd on CNCF Slack (slack.cncf.io)

Containerd: Project Update and Deep Dive

Session Feedback

Scan QR code to submit feedback

