



THE LINUX FOUNDATION



NORTH AMERICA

Being a Maintainer in the Age of LLM Mania

Kevin Hannon



Who Am I

- Co-chair of wg-batch in Kubernetes
- Approver of Kubernetes enhancements
- Love Open Source communities
- Skeptic of agentic development

Who Am I

- Want to figure out how to use AI to reduce maintainer burnout and provide stable software for users

What is a maintainer?

- Think “staff” engineer for project
- Reviewing features
- Setting direction
- Coaching new contributors
- Cleaning up work after feature engineers peace out
 - Sad but true :(

Kelsey Hightower at KubeCon 2026:
“Everyone is a ~~junior~~ engineer
when it comes to AI”
toddler

— Kelsey Hightower

First phase of AI as a Maintainer



WELCOME LIL ONE

PUBLISH AN AI CONTRIBUTION POLICY

Three types of AI policies

YOLO

- <https://openai.com/index/harness-engineering/>
- <https://github.com/openclaw/openclaw/blob/main/CONTRIBUTING.md#aivibe-coded-prs-welcome->

AI discouraged

- Llama.cpp : encourage AI to aide in development
- Gentoo:
https://wiki.gentoo.org/wiki/Project:Council/AI_policy

Human in the loop

- [Kubernetes](#)
- [Linux Kernel](#)

Kubernetes AI Policy

- Humans need to understand their code
- Discourage use of AI to respond to maintainer feedback
- Avoid attributing AI in commit messages
- Disclose AI usage

Ref:

<https://www.kubernetes.dev/docs/guide/pull-requests/#ai-guidance>

Second phase of AI as a maintainer



Create an AGENTS.md

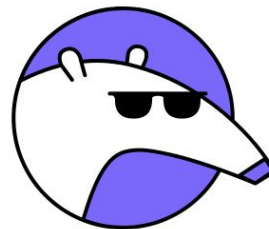
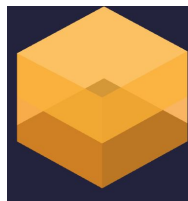
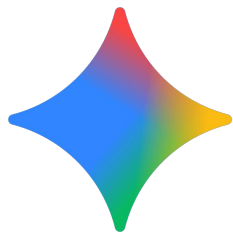
Think of [AGENTS.md](#) as a README for agents; a dedicated place the context and instructions to help AI coding agents

- Many ways to set these up

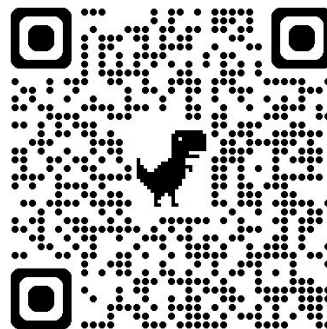
Getting your feet wet

- Use AI tools to be a “peer reviewer”.
- Workflow for me
 - Checkout PR
 - Ask an Assistant to do a first pass review

AI Code Review Tools



Now Available: Free GitHub Copilot Enterprise for CNCF Maintainers



Third phase of AI as a maintainer





Skills for maintainers

- Think of AI as an automation tool
- Automate time consuming tasks!

Time consuming task - triaging issues

- Take a github issue
 - Find duplicates
 - Figure out what subproject or owners should actually look at it
 - If its a bug report / regression, do a regression analysis
 - Write a detailed response and tag folks

example

kueue_cohort_subtree_quota reports CPU as 1000x the actual value #10746

Closed #10747

baoclvn1 opened 3 weeks ago

What happened:
When exporting the cpu metric from kueue_cohort_subtree_quota, it shows up as 1000x the actual value

What you expected to happen:
i.e. 96 cpu in a resource flavor in a cohort should report as 96 cpu not 96000

How to reproduce it (as minimally and precisely as possible):
curl metrics endpoint and grep for subtree
`curl https://localhost:8080/metrics | grep subtree`

Anything else we need to know?:

Environment:

- Kubernetes version (use `kubectl version`): 1.33
- Kueue version (use `git describe --tags --dirty --always`): 0.17.1
- Cloud provider or hardware configuration: EKS
- OS (e.g. `cat /etc/os-release`):
- Kernel (e.g. `uname -a`):
- Install tools:
- Others:

<https://github.com/kubernetes-sigs/kueue/issues/10746>

Thank you for the bug report. This doesn't seem too bad to fix. Would you like to attempt a fix?

AI Analysis

Root Cause: Missing millicPU-to-CPU conversion

CPU resources are stored internally as `millicPU` (`int64`) via `resources.ResourceValue()` which calls `q.MilliValue()` for CPU (`pkg/resources/requests.go:185`). When reporting the cohort metrics, these raw millicPU values are cast directly to `float64` without converting back to CPU units.

Buggy path (`pkg/cache/scheduler/cohort_metrics.go:133`):

```
p.quotaQty (int64, millicPU) → ReportCohortSubtreeQuota() → float64(quota) → NO conversion
```

How ClusterQueue metrics handle this correctly (`pkg/cache/scheduler/clusterqueue.go:518`):

```
int64 millicPU → resourceFloat() → ResourceQuantity() → QuantityToFloat() → proper float64
```

The helper `resourceFloat()` at `pkg/cache/scheduler/cache.go:1859` correctly round-trips the `int64` through `ResourceQuantity()` + `QuantityToFloat()`, which divides millicPU by 1000. The cohort metrics skip this conversion entirely.

The following ClusterQueue metrics use `resourceFloat()` and do not have this problem:

- `kueue_cluster_queue_nominal_quota`
- `kueue_cluster_queue_borrowing_limit`
- `kueue_cluster_queue_lending_limit`
- `kueue_cluster_queue_resource_reservations`
- `kueue_cluster_queue_resource_usage`

Affected Metrics (2)

Both cohort resource metrics have the same bug:

1. `kueue_cohort_subtree_quota` — `pkg/metrics/metrics.go:1854` — `float64(quota)`
2. `kueue_cohort_subtree_resource_reservations` — `pkg/metrics/metrics.go:1987` — `float64(usage)`

Fix

In `pkg/cache/scheduler/cohort_metrics.go`, the `applyCohortMetricPoint` function should use `resourceFloat()` (already available in the same package) to convert the `int64` values before passing them to the metric reporting functions. This requires updating `ReportCohortSubtreeQuota` and `ReportCohortSubtreeResourceReservations` to accept `float64` instead of `int64`, matching the pattern used by `ReportClusterQueueQuotas`.

Time consuming task - CI failures

- Dev branch - open
 - Many PRs merging
- Check CI to see a failing job but you were on vacation for a week being really AFK
- What PR broke the CI?

* maintainer reality is once features merge, it is usually on maintainers to keep project healthy

Triage CI example

[Failing-Test]: Failing static pod tests #137737

Closed #137870



kannon92 opened on Mar 14

Member

Failure cluster [295be8c191b924301d81](#)

Error text:

```
[FAILED] Timed out after 120.001s.
The function passed to Eventually failed at k8s.io/kubernetes/test/e2e_node/static_pod_test.go:106 with:
Expected
  <*v1.Time | 0x4a28e8ce4090>: {
    Time: 2026-03-13T08:38:35Z,
  }
not to equal
  <*v1.Time | 0x4a28e8be7260>: {
    Time: 2026-03-13T08:38:35Z,
  }
In [It] at: k8s.io/kubernetes/test/e2e_node/static_pod_test.go:114 @ 03/13/26 08:40:46.122
```

Recent failures:

[3/13/2026, 12:59:47 PM ci-containerd-node-e2e-serial-ec2](#)
[3/13/2026, 11:37:02 AM ci-kubernetes-node-swap-ubuntu-serial](#)
[3/13/2026, 11:28:59 AM ci-kubernetes-node-kubelet-serial-containerd](#)
[3/13/2026, 10:51:59 AM ci-kubernetes-node-arm64-ubuntu-serial](#)
[3/13/2026, 8:42:54 AM ci-node-crio-swap-serial](#)



kannon92 on Mar 14 · edited by kannon92

Edits Member Author

Note: This analysis was generated by AI (Claude) and should be verified by a human reviewer.

Edit: Updated with the correct date range — the original analysis used the wrong year for git log queries.

Analysis of Failing Static Pod Test

What the test does

The test at `test/e2e_node/static_pod_test.go:106` creates a static pod with an init container, waits for the init container to complete and records `pod.status.startTime`, then removes the init container and restarts kubelet. It expects the mirror pod to get a **new** `startTime` after restart (indicating the pod was recreated). The failure shows both timestamps are identical — the pod is not being recreated as expected.

Recent commits that could impact this (Mar 7–14, 2026)

1. PR #137362 — [on-demand-reList](#) (@tallclair)

- Major changes to PLEG (`pleg/generic.go`, `+270 lines`), `pod_workers.go`, `kubelet.go`, and `pod cache`
- Introduces on-demand PLEG pod relisting, per-pod cache refresh timestamps, and changes to sync result handling
- Modifies how pod workers request relisting, which directly affects pod lifecycle detection after kubelet restart

2. PR #137566 — [fixgetpods](#) (@yangjunmyfm192085)

- Changes container sorting in `kubeGenericRuntimeManager.GetPods()` and `GetPod()` to sort by create time and ID
- Merged March 13 — the same day the failures started
- Could affect which container state is returned for static pods after kubelet restart

3. PR #134768 — [pod-level-resource-managers](#) (@KevinTMtz)

- Pod-level CPU & Memory managers support, topology manager support, pod-level state
- Touches `kubelet.go` and resource manager integration

4. PR #136681 — [restart-pod](#) (@yuanwang04)

- Graduates `RestartAllContainers` feature to beta (now enabled by default)
- Also includes `cf33daace34` — "Fix LastTerminationStatus for RestartAllContainers" which modifies `kubelet_pods.go`
- A newly-default-enabled feature affecting container restart behavior could change how static pods behave after kubelet restart

5. PR #137248 — [propagate-context-cri-client](#) (@SergeyKanzhelev)

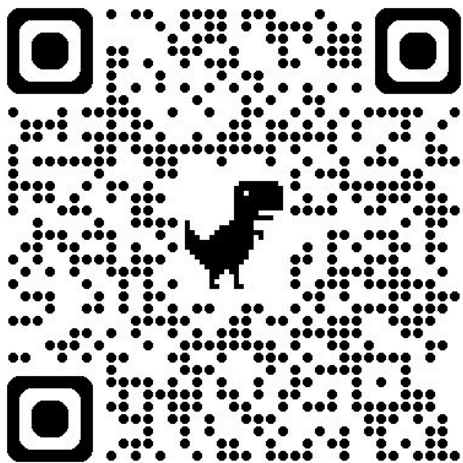
- Propagates context to CRI client `close` and uses contextual logging

6. PR #136185 — [ndf-bitmap](#) (@tallclair)

- Switches `NodeDeclaredFeatures` to bitmapped FeatureSet implementation, adds global default NDF registry

cc @tallclair @yangjunmyfm192085 @KevinTMtz @yuanwang04 @SergeyKanzhelev — could you take a look and see if any of these changes might be related to the static pod test failures?

Kubernetes Maintainer Skills



- a) Build commit messages following AI policy: `/commit-from-changes`
- b) Create a PR following pull-request template: `/create-pr-from-changes`
- c) Triage a github issue: `/triage-issue`
- d) Summarize CI failures: `/triage-prow-job`
- e) Summarize github notifications: `/triage-github-notifications`

Q&A

- Anyone using AI to aide OSS maintainership?
- How are you fighting “AI slop”?
- Any tools you find useful?