KernelCI kci-dev: Closing the Developer Feedback Loop

Arisu Tachibana

- KernelCI Infra WG member
- kci-dev creator / maintainer
- Gentoo Kernel leader
- CIP testing member
- Cybertrust Japan Co., Ltd.



20 minutes

- ~10 minutes: quick kci-dev intro & demo-style overview
- ~10 minutes: roadmap discussion and open Q&A



Who this talk is for

- If you are in this MC you are probably interested in Kernel testing
- Make it easier with kci-dev



Motivation: Why the feedback loop isn't closed (yet)

- Massive combination of trees, branches, architectures, and tests
- KernelCI provides tons of data, but primarily via dashboards and APIs
- Developers still rely on email reports or ad-hoc scripts to get results
- Friction remains in getting CI feedback directly into dev workflow



KernelCI in one slide

- Upstream, open testing platform for the Linux kernel
- Builds, boots, and tests across distributed labs (continuous integration at scale)
- Volume: thousands of kernel builds and boot tests each week
- Interaction historically via web UI or raw JSON APIs not developer-friendly
- Need: an ergonomic way to query and trigger tests from the terminal



What kci-dev lets you do today

- Query results: Get build/boot summaries, compare test results between revisions, filter by hardware, etc. — all via one CLI
- **Trigger tests**: Launch KernelCI jobs for your tree or branch on demand and do simple bisection (no waiting for nightly runs)
- Fetch logs quickly: Download and inspect boot logs or test output straight from the terminal
- Integrate easily: Scriptable JSON outputs for CI pipelines, plus shell completions for interactive use

Real-World Context

1. Developer Tree

- Trigger KernelCI jobs for any tree in KernelCI
- Example: using kci-dev checkout with --giturl, --branch, --tipoftree

2. CIP SLTS kernels

- Trigger SLTS kernel builds for CIP trees
- Example:

```
kci-dev checkout --giturl https://git.kernel.org/.../cip/linux-cip.git --branch linux-6.1.y-cip --tipoftree
```

• Validate long-term maintenance CI status directly from the terminal

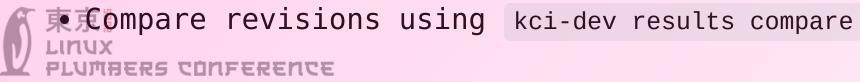


3. ChromiumOS coverage

- Use kci-dev maestro coverage to retrieve kernel coverage info
- Terminal shows function/line coverage and URLs to the reports
- Graph views generated through --graph-output and opened locally

4. Gentoo Kernel (gentoo-sources)

- Validate Gentoo patchset kernels using KernelCI
- Run builds/tests against gentoo-sources trees before publication



Triggering builds with job filter

```
kci-dev checkout \
  --giturl https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git \
  --branch master \
 --tipoftree \
  --job-filter baseline-arm64-foundriesio \
  --watch \
  --test crit
Retrieving latest commit on tree: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git branch: master
Commit to checkout: d358e5254674b70f34c847715ca509e46eb81e6f
OK
treeid: 98a7e00aae69091a4923f00f2f42508b4c2558c962b797bafa661ceadc8063b1
checkout_nodeid: 693af331fee8383e92b6a864
Watching for jobs on treeid: 98a7e00aae69091a4923f00f2f42508b4c2558c962b797bafa661ceadc8063b1
Watching for test result: crit
job_filter: baseline-arm64-foundriesio, checkout
Running job....
PASS branch checkout - node_id:693af331fee8383e92b6a864 (2025-12-11T16:39:06.510000)
Running job......
```



Remaining gaps in developer workflow

- Maintainer adoption: Not all maintainers use kci-dev yet some workflows (e.g. patch series on mailing lists) aren't covered
- Context switching: Still some need to visit dashboards for certain info (e.g. detailed triage or artifact browsing)
- Incomplete features: New KernelCI data (like bisect info)
 need better CLI support or docs
- Performance concerns: Large result sets can be slow to fetch; no offline caching means repeated queries are
- 東京pefficient LINUX PLUMBERS CONFERENCE

Technical debt & under-the-hood improvements

- Refactor and modularize: Clean up duplicate logic (e.g. consolidate Maestro API code) to simplify maintenance
- Improve test coverage: Add more unit and integration tests for kci-dev itself to catch regressions early
- Robustness: Handle API changes gracefully (ensure kci-dev keeps working as KernelCI evolves)
- Library-ready structure: Decouple CLI parsing from core logic so kci-dev functions can be imported and reused easily



Future features on the roadmap

- Ad-hoc patch testing: Trigger KernelCI runs on patch series or unmerged code (e.g. from an mbox or local branch)
- Smarter result diffing: Richer comparisons between test runs (highlight regressions, link to relevant logs, inline dashboard links for context)
- Local caching of data: Cache recent results locally to speed up repeated queries and allow basic offline result browsing
- **Deeper Git integration**: Automatically infer context (current git repo/branch) for commands and potentially integrate with git workflows (e.g., run tests on git push)

Team priorities & milestones

- Make kci-dev maintainer and developers friendly: Ensure kernel tree maintainers can seamlessly use it for their workflows (multiple branches, stable releases, etc.)
- Enable library usage: Provide a supported Python API so kcidev can be imported as a module in other tools or CI systems
- Stability towards v1.0: Focus on fixing bugs and polishing UX for a stable 1.0 release (address known issues, consistent outputs)
- Stay in sync with KernelCI: Align with KernelCI's new pipeline and features, so kci-dev remains fully compatible the backend evolves

Questions for Kernel Developers & Maintainers

- What slows you down today?
- Do you need CI results earlier in your workflow? How early?
- What data from KernelCI do you actually look at during development?
- Would patch-series testing (mbox / git range) help your workflow?
- How often do you compare test results between revisions?
- Do you want CI to auto-run on your tree, or only on-demand?
 Tree registration on KernelCI or local patches? self-service
 東東面estro deploy method?
- what's missing in kci-dev to make you use it daily?

Getting started after this talk

- Give kci-dev a try on your own kernel tree this week (pip install kci-dev and go!)
- Share your feedback: Identify your top pain points in kernel testing and let us know if kci-dev doesn't solve them yet
- Join the community: Come to KernelCI community meetings or chat — help us prioritize what matters to you https://discord.gg/KWbrbWEyqb
- Contribute: Whether it's a bug report, a documentation fix, or a code patch, all contributions are welcome and valued

Thank You!

kci-dev

```
Documentation: https://kci.dev
GitHub: https://github.com/kernelci/kci-dev/
Your feedback will help shape the future of kci-dev
```

• slides

Slides:

https://aliceinwire.github.io/presentations/LPC_2025_TALK/

