Getting Started With New KernelCI CLI Tools

Automating Linux Kernel Testing and Validation

Arisu Tachibana
KernelCI / Gentoo

Motivation: Why Kernel QA Is Hard

- Many trees, branches and configurations
- Multiple architectures, boards and toolchains
- CI dashboards are powerful but not ergonomic
- Developers still click UIs or write ad-hoc scripts
- Command-line tools keep context in the terminal

KernelCI in One Slide

- Upstream, open testing for the Linux kernel
- Builds, boots and tests across distributed labs
- Great data, but interaction is:
  - Mostly via web UI
  - Raw REST APIs
- Need something ergonomic in the terminal

Real-World Context: gentoo-sources

- Fast-moving tree with many patches
- High pressure to avoid regressions
- Need quick, scriptable visibility into CI results
- KernelCI CLI tools are built with this reality in mind

Introducing the KernelCI CLI Tools

- **kci-dev**
  - Developer-focused CLI to interact with KernelCI dashboards and Maestro
  - Installable from PyPI (`pip install kci-dev`) or development snapshots via poetry
  - Ships `results` (dashboard), Maestro, and validation commands in one binary
  - Completions bundled for bash, zsh, and fish
- **kci-deploy** *(work in progress)*
  - Tool for deploying local / internal KernelCI maestro stacks
- Goal: make KernelCI a first-class tool in your terminal

Install & Configure Fast

- `virtualenv .venv && source .venv/bin/activate`
- `pip install kci-dev`
- `kci-dev config` scaffolds `~/.config/kci-dev/kci-dev.toml` for Maestro auth
- Results-only workflows work **without** a config file
- Request API tokens via the KernelCI GitHub issue template when you need Maestro access

Shell Completions Included

- Bash: `source /path/to/kci-dev/completions/kci-dev-completion.bash`
- Zsh: add completions directory to `$fpath` and run `compinit`
- Fish: copy `completions/kci-dev.fish` to `~/.config/fish/completions/`
- Keep completions in sync with the installed kci-dev version

kci-dev: What You Can Do Today

Dashboard (`results`) commands

```
kci-dev results summary --giturl <git url> --branch <branch> --history
kci-dev results compare --giturl <git url> --branch <branch> <older> <newer>
kci-dev results hardware summary --name <vendor,board> --origin maestro --json
kci-dev results tests --giturl <git url> --branch <branch> --commit <sha> --filter filter.yaml
```

Maestro commands (need config/token)

```
kci-dev checkout --branch <branch> --giturl <git url> --commit <sha> --job-filter <regex> --watch
kci-dev testretry --nodeid <node-id>
kci-dev watch --nodeid <node-id> --job-filter <regex>
kci-dev maestro validate builds --all-checkouts --days 7 --table-output
kci-dev maestro results --nodeid <node-id> --json
```

kci-dev: Everyday Workflow

1. **Install & configure**
   - `pip install kci-dev`; optional `virtualenv` + completions
   - `kci-dev config` creates `~/.config/kci-dev/kci-dev.toml` (Maestro only)
   - Request API tokens from KernelCI GitHub issue template
2. **Discover context**
   - `kci-dev results trees` to see tracked repos/branches
   - `kci-dev results summary --history` for recent pass/fail trends
   - `kci-dev results hardware summary` to see per-board coverage
3. **Inspect and compare**
   - `kci-dev results build|boot|test --id … --download-logs`

kci-dev: Quality-of-Life Details

- Rich output modes: table, JSON, quiet for scripts
- `--history` summaries with pass/fail/inconclusive color coding
- `compare` highlights regressions with dashboard/log links
- `results hardware` views boards + per-board summaries
- Filters: `--filter` YAML for hardware/tests, `--arch`, `--status`
- Profiles via `--instance` / `--settings` to switch endpoints
- Designed for piping into `jq`, `fzf`, notebooks

Dashboard Endpoint Reality Check

- `kci-dev results` currently pulls a multi-megabyte dashboard payload
- Initial download may take a few seconds; caching work is in progress
- Known upstream issue is tracked in the KernelCI dashboard repo

kci-deploy: For Lab Owners

- Simplifies standing up a Maestro stack
- Encodes best practices for networking and storage
- Shares the same UX patterns as kci-dev
- Early previews welcome: help shape the roadmap!

Automation Patterns

- **Pre-submit checks**: gate merges on KernelCI signal
- **Nightly reports**: email/Matrix summaries via cron
- **Release readiness**: track blockers for RCs
- **Local sanity tests**: run focused boards before shipping

Demo Script (Idea)

```
# Configure once
virtualenv .venv && source .venv/bin/activate
pip install kci-dev
kci-dev config  # writes ~/.config/kci-dev/kci-dev.toml

# Morning health check
kci-dev results summary --giturl https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git --branch master --history
kci-dev results hardware summary --name mediatek,mt8195 --origin maestro --json

# Investigate a failure
kci-dev results boot --id maestro:<node-id> --download-logs | less

# Nudge infra
kci-dev testretry --nodeid <node-id>

# Validate dashboard vs Maestro for the last week
kci-dev maestro validate builds --all-checkouts --days 7 --table-output
```

Roadmap & Collaboration

- Deeper git integration (auto-pick branch/commit)
- Better diffing between runs
- Inline links back to dashboards
- kci-deploy installer previews in Q2
- Looking for testers, lab partners, and feedback

Documentation & Updates

- Docs: https://kernelci.github.io/kci-dev/
- Source: https://github.com/kernelci/kci-dev
- Token requests and issues: KernelCI GitHub templates

Getting Started After This Talk

- Try **kci-dev** with your tree this week
- Share your top 3 pain points in Kernel QA
- Join the KernelCI community calls / Matrix
- Contribute docs, plugins, and issue repros

Thank You!

Slides: https://aliceinwire.github.io/presentations/OSSJ_2025/
X: https://x.com/arisu_gyaru
Instagram: https://www.instagram.com/gyaru_arisu/
GitHub: https://github.com/aliceinwire
Questions welcome