Continuous Efficiency: AI as a daily habit for greener software

When was the last time someone asked in a standup: *"How could we do this more sustainably?"* I don't hear it often—not because developers don't care, but because time is scarce and the backlog is loud.

The GitHub Next and GitHub Sustainability teams propose a framing that feels more actionable: treat sustainability as **efficiency**. Faster code, less waste, fewer resources burned. Their name for the practice is *Continuous Efficiency*: incremental, validated improvements that accumulate over time instead of arriving as a big "green refactor" that never gets scheduled.

Why "continuous" is the point

Most teams already have habits that run continuously: tests, linters, formatting, security checks, CI. Continuous Efficiency tries to join that family. The promise is not magic optimization; it's that small improvements can happen regularly, be measured, and be reviewed—so the codebase gently trends toward being cheaper to run and easier to maintain.

Where AI fits

The key enabling idea is combining Continuous AI (automation enriched by modern LLMs inside collaboration workflows) with Green Software practices (building systems that are more energy-efficient and typically more performant and resilient). In the best case, this turns sustainability from "extra work" into "default behavior".

Agentic workflows as a delivery mechanism

One concrete implementation discussed is an experimental approach called Agentic Workflows running in GitHub Actions. The interesting part is authoring: instead of writing traditional automation logic, you describe intent in natural language inside a Markdown file, then compile it into a standard workflow. At runtime an agent can inspect repository context, propose changes, and surface them through familiar collaboration artifacts like comments and pull requests—while staying within the guardrails of the platform.

Standards, rules, and performance work

Two directions stand out. First, turning standards into executable rules: describe what "good" looks like, then apply it across a codebase more broadly than conventional pattern-based tooling. Second, performance improvement in heterogeneous real-world repos: an iterative approach where an agent can discover how to build and benchmark a project, run measurements, and propose targeted optimizations under human guidance.

I like this because it doesn't pretend expertise disappears. Instead, it tries to scale it: making improvements easier to start, easier to validate, and easier to repeat—so teams can keep learning in public, but also keep their software lean.

Contributor: Alessandro Linzi