OpenSearchCon INDIA

From Code to Carbon: GreenOps for OpenSearch:
Building Cost-Efficient and Sustainable Tech

- Amrutha KH, SDE II

Key Terms & Concepts

OpenSearchCon INDIA

GreenOps is our approach and tooling for greener operations: catching wasteful compute patterns in code on every pull request, before they reach production.

Green = less wasted work, less energy, lower emissions. **Ops** = where teams already work.

OpenSearch

The engine.

The search engine your app uses to find and analyze data.

GreenOps

The early warning on code.

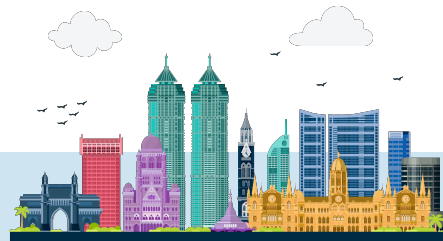
The coach + gate that checks code before merge.

Code → Carbon

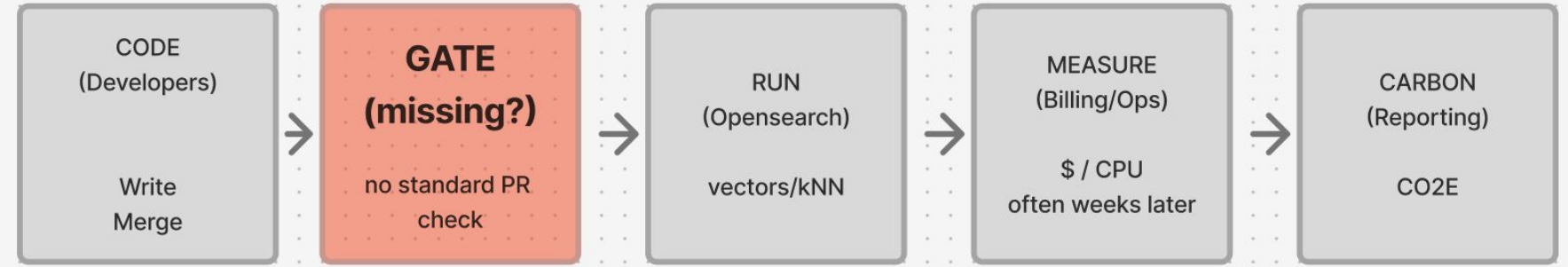
The story we're telling.

Connecting developer decisions to sustainability outcomes through PR checks and post-deploy truth.

"The software you merge causes work on servers; work uses electricity; electricity has a carbon footprint depending on the grid."



The Problem: Where the Chain Breaks



1. Design Cost

OpenSearch + vectors are costly by design

Embeddings, kNN, large indexes → RAM, CPU, ingest. The infrastructure track exists because this is hard.

2. Code Multiplication

Application code can multiply that cost

Serial API calls, huge retrieves, full reindexes. Passes tests, still burns the cluster.

3. Disconnected Impact

Carbon & cost are disconnected from the PR

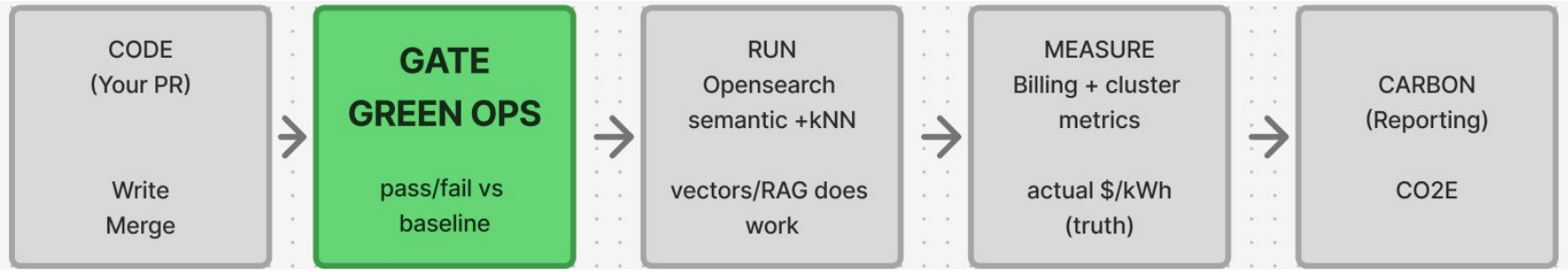
Devs see latency; finance sees kWh later. Rarely linked to the merge that caused it.

"We ship code, → OpenSearch does the work, → the planet and budget pay—but we don't gate 'getting worse' at step 2. "

GreenOps fills the gate - before OpenSearch and RAG pay the price

OpenSearchCon INDIA

From merge decision to cluster truth, GreenOps guards the gate; OpenSearch and billing prove the impact.



What GreenOps does:

- **Reproducible:** Runs on every PR - same rules locally and in CI.
- **Static Analysis:** Scores efficiency risk based on code patterns (not real-time power).
- **Regression Fails:** Compared to a committed baseline on `main`.

PR (GreenOps) Production (OpenSearch)

"Should we merge?" "What actually happened?"

Indicative carbon impact score CPU, RAM, ingest, query latency, bill

Catches bad habits early Proves if fixes worked

"GreenOps: fail the PR, not the planet honestly."

How GreenOps runs in practice

OpenSearchCon INDIA

Process: init once · Gate every PR · Measure after ship on OpenSearch:

`init` → `PR scan` → `ship` → measure

1. INIT (once)

`npx greenops init`

`greenops.config.json`

`.greenops/baseline.json`

Commit both → Baseline for future PRs

2. PR (every change)

`greenops scan --diff` → pass / fail

Only changed .ts/.js

AST → score 0–100

Gate vs baseline → Pass / Fail

INEFFICIENT EXAMPLE:
87 - FAIL (6 findings)
Serial index() in loop, KNN > 200

3. SHIP & MEASURE

App → OpenSearch (vectors, kNN, RAG)

Dashboard: `/metrics`

- Measure: $, CPU, latency
- Carbon: kWh x grid
- Optional FinOps $

EFFICIENT EXAMPLE:
0 - PASS (0 findings)
Bulk index(), filtered kNN

"GreenOps: fail the PR, not the planet. From code AST to cluster metrics truth."

Built for the OpenSearch ecosystem

OpenSearchCon INDIA

Before merge	**GreenOps gate on code**	GreenOps gates application code that indexes, queries, and feeds RAG (v1: JS/TS; more languages next).
After merge	**OpenSearch = truth**	OpenSearch runs vector search & kNN; cluster metrics + billing show if the gate mattered.
Open direction	**Others can extend** `packages/cli/src/rules.ts`	CLI + rules others can extend (LF-style open tooling alongside OpenSearch, not a fork of it).

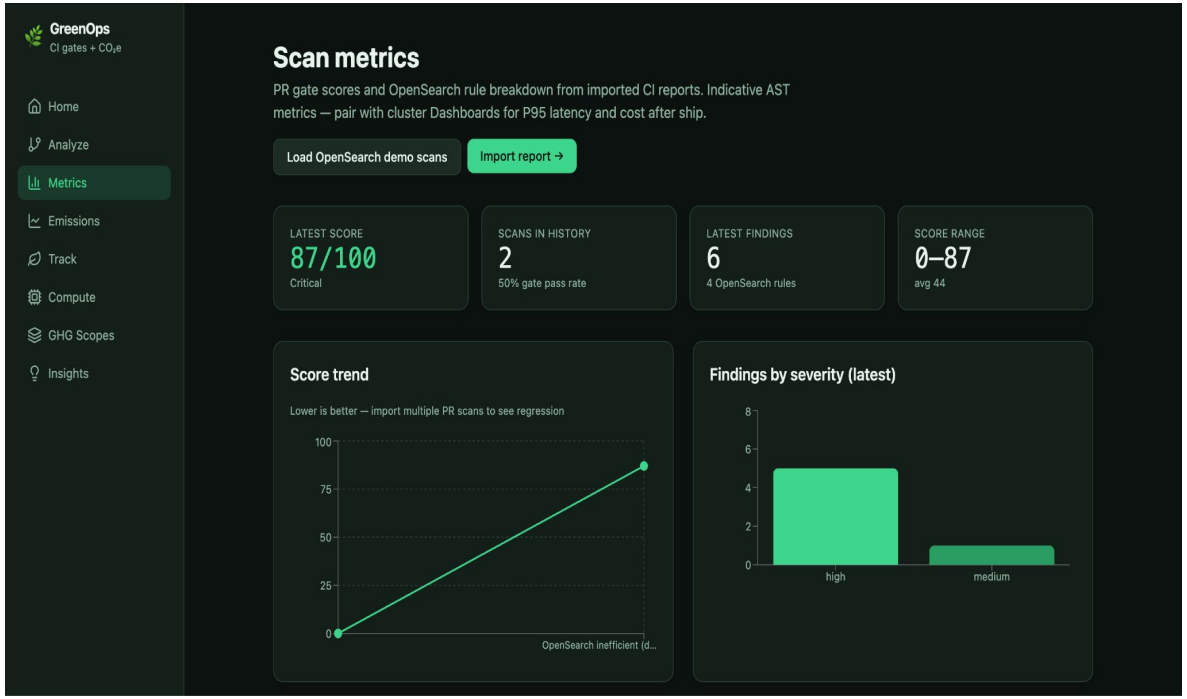
"GreenOps stops bad patterns before the merge. After merging, OpenSearch runs vectors and kNN - Dashboards, CPU, latency, and billing tell you if it worked. We don't fake that number from AST; that's the second half of code-to-carbon."

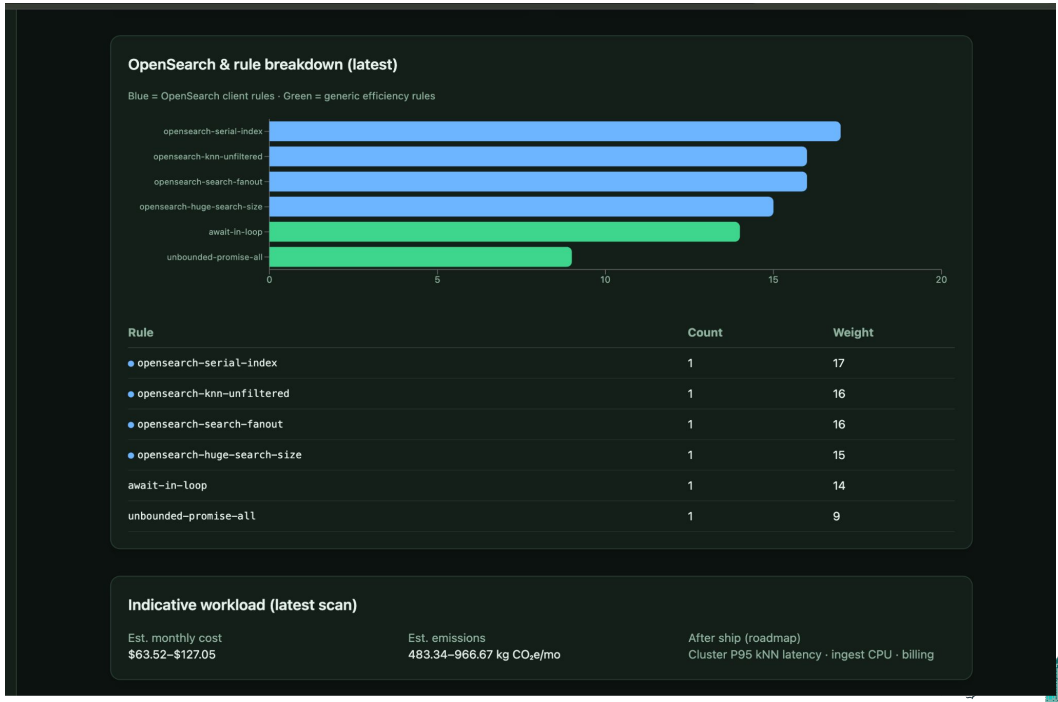
"Our checks live in open TypeScript on GitHub: anyone can add rules for bulk ingest, kNN size, and filters. Teams still run the same ***`scan --diff`*** *in CI. GreenOps doesn't replace OpenSearch; it sits in front of it on the pull request."*

Code smell (demo)	OpenSearch equivalent
Promise.all + many calls OR Huge size / big kNN k	Parallel search() fan out, Over-fetching hits & vectors for RAG

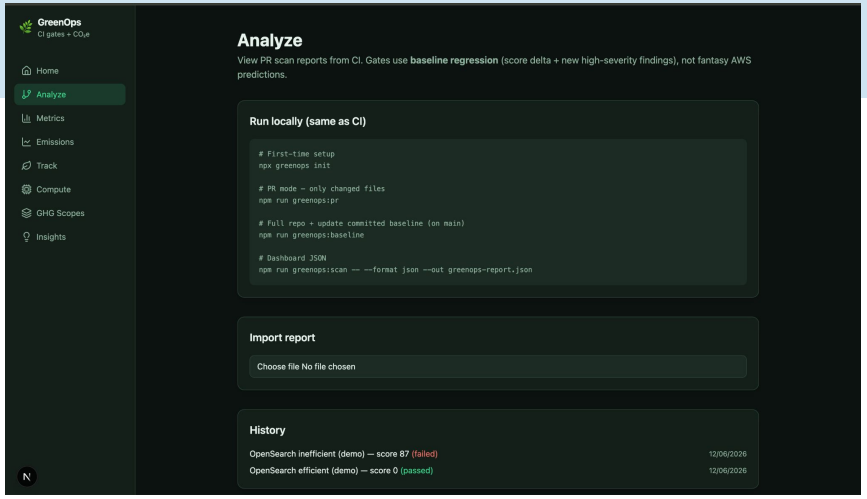
What’s next and closing points

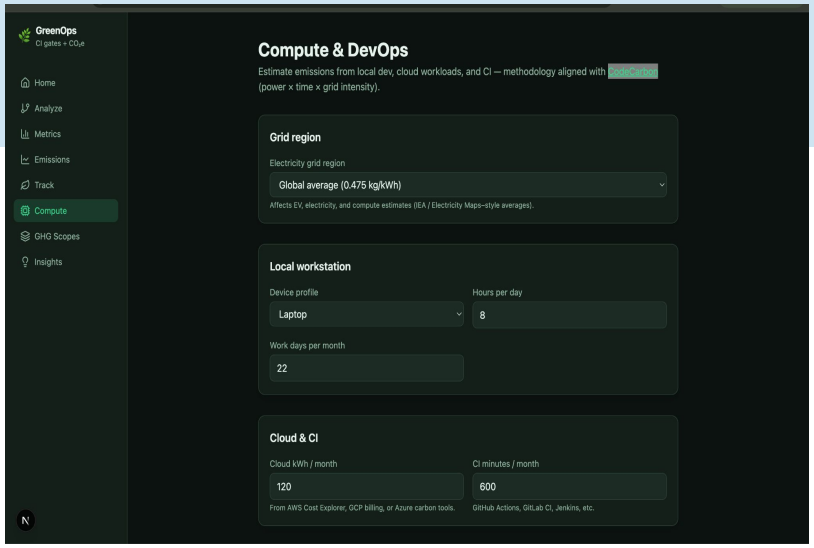
Now	Next
OpenSearch PR rules (bulk, kNN, fanout)	Auto-snapshot on every deploy
Metrics dashboard	AWS Cost Explorer auto-import
Proof timeline - before/after with git commit + score + cluster + $	P95 kNN from Observability

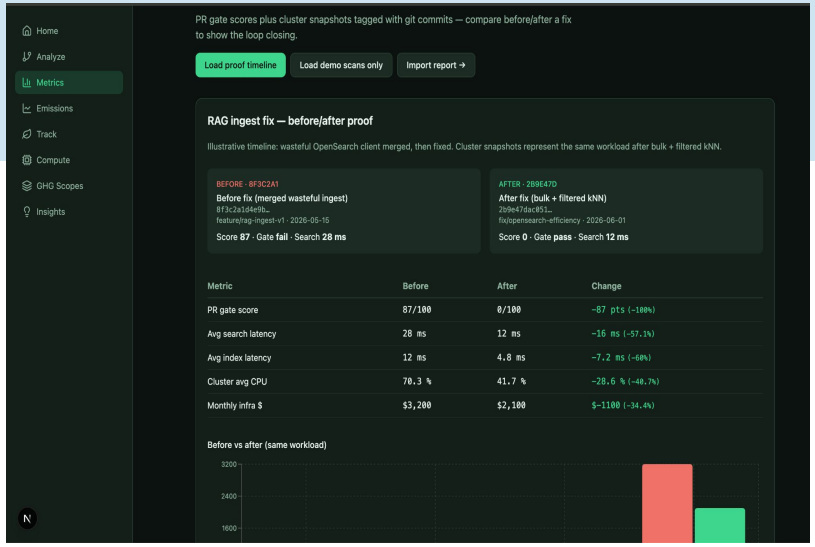




DEMO







CLOSING THOUGHTS: How GreenOps make sense:

1. Gate in the PR - catch serial index, huge kNN, and search fanout before merge
2. Score with proof - 87 → 0 on the same OpenSearch ingest patterns you run in prod
3. Cluster truth after ship - CPU, latency, and $ in one before/after timeline
4. Open alongside OpenSearch - AST in CI today; cluster metrics and FinOps close the loop next

Catch waste in the PR. Measure truth in OpenSearch. Close the loop to carbon.

Demo repo: (REPO LINK)


OpenSearchCon
INDIA
THANK YOU