Your Cloud App's Hidden Carbon Cost: Why DevOps Must Go Green

Nigel Dsouza

When you automatically scale a Kubernetes cluster or spin up an EC2 instance, you're not merely installing code.

You're consuming energy.

However, as engineers, we almost never consider the question: *How much carbon will my deployment contribute?*

Having worked on large-scale, high-availability system design for financial platforms for a number of years, I have direct experience with how performance, uptime, and velocity drive architectural choices. But as climate change picks up speed, we need to consider CO_2 in addition to SLA when making infrastructure decisions.

Cloud Isn't Usually Green

It's easy to believe that cloud computing is "greener" than on-premises systems. Providers such as AWS, GCP, and Azure invest in data center efficiency and renewable energy. The catch is that you—not them—are in charge of optimization.

Waste and emissions are caused by underusing resources, provisioning large instances, and leaving staging environments running overnight. According to a 2023 assessment by the International Energy Agency, data centers were responsible for around 1.5% of the world's electricity usage.

That's us. It's not abstract.

Practices in DevOps Require a Sustainability Layer

The use of infrastructure-as-code has simplified deployment. But it has also reduced the friction caused by overprovisioning. We require a new mindset: **Sustainability-as-Code**.

I've implemented the following procedures in my work as a cloud engineer:

- **Tag everything with purpose and TTL:** When the testing time expires, resources are automatically deleted.
- **Use serverless architectures sensibly:** Although Lambdas can be more efficient, the benefits may be offset by concurrency settings and payload bloat.
- **Choose carbon-intelligent regions:** Both AWS and GCP provide cleaner-energy regions; choose the greener option.
- **Measure cloud emissions:** Cost dashboards should be accompanied by tools such as AWS Customer Carbon Reports and Cloud Carbon Footprint.

Sustainability Is a Problem for Engineering

The finest engineers leave a clean imprint in addition to writing clean code.

We are obsessed with cost per request and CPU utilization. Why not measure deployments in grams of CO_2?

What if your pipeline measurements included green KPIs? What if we boasted about reducing carbon emissions in the same way we do latency?

Our work on failover and disaster recovery architecture at Fidelity allowed us to reconsider cross-regional tactics. We decreased our carbon footprint without compromising SLAs by redirecting low-priority traffic to more environmentally friendly regions during off-peak hours.

Leaders in the Cloud Must Establish the Standard

Sustainable architecture is a leadership duty, not just a nice-to-have. We must design for energy efficiency in the same way we invest in automation and uptime.

Ask yourself if your infrastructure is elastic—or simply big by default—if you're a cloud lead.

- Are your developers taught to consider carbon as an expense?
- Do your engineering KPIs include sustainability metrics?

No one else is responsible for the green cloud. We own it.

And we'll be able to construct a sustainable future more quickly if we begin treating carbon as technical debt.

About the Author

Nigel Dsouza works for Fidelity Investments as a Tech Lead and Principal Software Engineer. He specializes in developing scalable, efficient systems in AWS, cloud-native architecture, and DevOps automation. He has a strong interest in climate-conscious cloud development and sustainable engineering.