The Everlasting Cloud: When You Don't, and Infrastructure Becomes Immortal

Nigel Dsouza

Consider the following scenario: 2040. Ten years have passed since you left. Despite this, your cloud infrastructure—designed for a significant financial platform—remains operational.

Your Terraform modules are still being deployed. Releases are still being triggered by your CI/CD processes. Not only has your system outlived you—it has outlived your job.

Imagine that the engineers who are keeping it up to date are unaware of your identity. They are unaware of the reasons behind your trade-offs.

However, your code—your logic—continues to function, rejecting incorrectly configured builds, enforcing policies, and making judgments.

The ghost in the machine is now you.

Infrastructure's Immortality

We discuss legacy code a lot. But what if that phrase is no longer metaphorical?

What if cloud architecture is becoming a durable, executable record of our professional thoughts—a kind of digital afterlife?

Today's tools already facilitate this:

- Git keeps a journal-like record of your commits.
- Terraform brings your intention to life.
- Monitoring tools track score like a heartbeat monitor.

The commit logs contain your name. IAM roles, module patterns, and naming standards are where your logic is found.

You might not survive.

Your infrastructure, however, won't die.

The Moral Accounting

That should sound dystopian—because it is.

What happens if an autoscaling policy you wrote in 2022 is blindly trusted by a future engineer who is unaware that it was a workaround for a vendor bug that has since been fixed?

What happens if your once-state-of-the-art disaster recovery logic becomes a liability—but no one is willing to remove it because it's still functional?

What happens if AI systems start automatically creating infrastructure based on your historical choices, trained on your past preferences?

How does it feel to be the model?

Digital Permanence's Curse

Software used to be transitory. You wrote it. It ran. It was gone.

Now, Infrastructure-as-Code immortalizes your reasoning. It is permanently auditable, deployed, forked, and cloned.

Yet we write it as though it were disposable YAML.

We must pose difficult questions:

- Do we want our choices to be permanent?
- Should there be a deadline for infrastructure?
- Are digital wills necessary for our architecture?

Sunset Engineering: A Novel Field

What if the ability to shut things down is more crucial in the coming ten years than scaling up?

Sunset engineers are experts in developing departure routes, recording the logic, integrating warnings, and gracefully retiring systems.

We need metadata that goes beyond setup to explain intent.

We need tools that enable future teams to ask, "What was this doing here?"—and actually get an answer.

We must acknowledge and plan for the possibility that the things we create will outlive us.

In Summary: The Legacy Is You

This is not an infrastructure tale.

It's an identity narrative.

You believed that you were deploying code.

But you were deploying yourself—pattern by pattern, policy by policy, line by line.

A portion of you was still running in the future you were creating.

Ask yourself, "Do I want this to be my ghost?" the next time you develop a module.

About the Author

Nigel Dsouza is a Principal Software Engineer and Technical Lead at Fidelity Investments, where he builds robust cloud-native systems that—hopefully—won't outlive him. He writes about the peculiar immortality of software, ethics, and the intersection of infrastructure.