Developer's Identity Crisis: Microservices vs. Monolith

Nigel Dsouza

We used to be the custodians of monoliths—enormous codebases that contained everything. Yes, they were a mess. However, they were complete. It made sense.

Then we were told to break them by the industry. Break the monolith. Disconnect everything. With the advent of microservices, being a developer was no longer just about knowing the system. It was about realizing your role in it.

The Fragmented Engineer's Birth

You were a city-builder in the monolith world. You are now a plumber for a single street in microservices. You own the pipe—not the domain or the flow—but `service-accounts-api-v2`.

And that is confusing, even though it is scalable.

We transitioned from architects to tenants. From those who create to those who maintain. And whether we acknowledge it or not, a lot of us fail to see the monolith's clarity.

The Autonomy Myth

Autonomy was promised by microservices. Teams could fully own their services. However, in reality?

Cross-functional ownership has been exchanged for cross-team dependencies.

Do you want to alter the onboarding process? Five approvals, three PR merges, and a Slack conversation with `DevOpsDragon420`—who hasn't responded in days—are required.

It turns out that autonomy is a UX issue.

Memory as a Monolith

A monolith is a narrative system. You can view the form of decisions made over time by scrolling through a file. The evolution—good, bad, and the hacks that were shipped on Friday night—is palpable.

What are microservices? They have amnesia. Without a state. Separated. Intended to forget everything but their work.

They scale beautifully. However, they can't recall why.

The Dilemma Faced by the Developer

We need clarity because we are engineers. Storytelling. Command. But our tools are becoming more distributed, modular, and AI-assisted.

We've been instructed to compose symphonies using only trumpet parts.

So we hold onto architecture diagrams as if they were treasure maps. We create too many documents. To recreate a mental image of the system, we build countless tools.

Identity Reclamation in a Dispersed World

Stop acting like this isn't difficult.

The time has come to create systems that are discoverable as well as decomposable.

That narrate their tale. That act as tour guides rather than gatekeepers—welcoming new developers. Systems that make engineers feel whole again.

Because we are still storytellers whether we write microservices or monoliths. Still able to solve problems. Creators nonetheless.

And we deserve systems that not only scale but also restore our sense of development.

About the Author

Nigel Dsouza works for Fidelity Investments as a Principal Software Engineer. He has constructed monoliths, broken them, and sewn them into clouds. He writes not only for the machines but also for the developers who are trying to recall their initial motivation.