Symphony for the Broken: Coding in the Era of Post-Apocalyptic Horror and Cyberpunk Dreams

Nigel Dsouza

After a vicious mission, all you can do in *Cyberpunk 2077* is stand on a balcony and watch neon spill across a broken city. The soundtrack is amazing; it's eerie and ambient, almost like Hans Zimmer composed the music for your failure.

It's stunning. And very depressing. Because what was once thought of as a bright chrome future is now rusty, jerky, and filled with regret.

I sometimes feel the same way when I look at production logs at two in the morning.

We're living in our own science fiction world.

We've turned into cyberpunks—using shell scripts and coffee to patch outdated systems. Observing AI models that we don't fully comprehend make unpredictable decisions. Composing Terraform as though it were a spellbook.

In a world that is collapsing more quickly than we can debug it, we create systems that are supposed to last forever. We automate until we lose sight of how it functions. Until we forget why we began, we deploy.

The Effects of Scale

The world ends in *Fallout 4* as a result of our unquestioning pursuit of limitless energy.

Although it's fiction, it's not that far off.

We scale systems without considering how they might fail. Until everything is weightless, we strive for serverless, stateless, and trustless. The pipeline then bursts one day, leaving no one left who understands how it operates.

Our cloud architecture is evolving into automated, sealed, and ultimately banned vaults.

Hans Zimmer Would Recognize

Infrastructure wouldn't be happy if it had a soundtrack. Zimmer's deep, vibrating strings would be the source. Small keys. Silences and swellings. The sound of fragility, awe, and tension.

That's how it feels to develop systems at a global scale while facing deadline pressure and an impending incident budget. It is a slow symphony of entropy that is hardly contained.

The composer is you.

Writing code is only one aspect of system architecture. You compose a score.

Every module represents a motif. Every service call is a harmony. A crescendo for each deployment.

And what if you don't leave behind records, trends, and intentions? Only static will be heard by your successors.

Moving Towards a Future That Is More Playable

Perhaps more automation isn't the solution. Perhaps it's increasing the playability of systems.

Similar to games, our infrastructure ought to be:

- **Explorable**: safe to break, observable, and self-documenting.
- **Narrative**: a story is told through commit logs and comments.
- **Adjustable**: similar to sliders for latency, cost, and performance.
- **Immersive**: the tooling and dashboards resemble control panels rather than punishment.

What if engineers eagerly anticipated deployment, much like players anticipate boss battles? What if exploring a codebase was similar to learning about lore? What if our alerts were written by Zimmer?

In Summary: The Code That We Are Due

Rust and chrome are not the future. Both are involved.

The next movement is written in that tension between innovation and decay, between ambition and burnout.

You do more than just code. You are writing a civilization's music.

Make it sing, then.

About the Author

Nigel Dsouza works at Fidelity Investments as a Principal Software Engineer, gamer, and cloud-native system composer. He documents as though someone might need to survive in his architecture and codes as though it were a cyberpunk noir. Minor is his preferred key.