Will Ruggiano

GitHub LinkedIn Discord: *willruggiano* Email: *crowd.must.cue@usecloaked.com*

My background is actually in architecture (i.e. undergrad) and I have been mostly self-taught in the ways of coding (though I have had several incredible colleagues and mentors along the way). It started with Java - I wrote my first main method with the help of a Dummies Guide :) Years later I learned C++ while working on a real-time streaming protocol at AWS, and this really sparked my interested in the lower levels: the network stack, memory management, the *type system* and compiler. Around that time I also found Nix and neovim which fueled many a late night, side project. Most recently I've been learning a new language, Zig, and becoming increasingly interested in machine learning. I've primarily fueled this interest by (a) reading academic papers like "Attention Is All You Need" and books like "Why Machines Learn", and (b) applying various learnings in the context of a practical syllabus like Karpathy's Zero to Hero lectures.

Experience

Software Engineer, Tendrel 2023–

- The usual web stack: React, Typescript, NextJS/GraphQL, Postgres.
- Led the development and release of our most "successful" application.[1]

Software Engineer, Amazon/AWS 2016–2023

- Joined Amazon out of college.
- Spent four-ish years on the retail side with the Notifications team. More or less: SOA, Java, DynamoDB.
- Worked with two phenomenal engineers to design, build and deploy a new notifications service, and then migrate/deprecate the legacy (Perl) service.[2]
- Stumbled upon Apple's iOS rich-content push notification APIs, and then led its integration into Amazon's iOS mobile application.[3]
- Transitioned to AWS to try something new. Landed on a team building a real-time streaming protocol for AWS's VDI products.
- Worked primarily with one other engineer on the core protocol/shared library. C++(17), UDP, STUN/TURN and a little bit of Python mixed in for sanity.

Undergrad (Architecture), University of Virginia 2012-2016

- I loved Legos as a kid, so architecture seemed like the perfect fit :D
- First coding experience: Python in Grasshopper (Rhino 3D).

---

[1] This was the only application that saw organic sign ups

[2] This had two big effects: (1) During migration, I uncovered a bug in the legacy system that resulted in a substantial amount of high volume notifications being dropped, ~20% or so if I recall correctly, which meant a lot of money down the drain. I got a promotion out of it :) (2) We cut our infrastructure bill by 70% (even with the 20% bump in traffic)

[3] I don't have raw numbers on this one but we targetted the high volume use cases first (enabled by [^2])... so, somewhere on the order of the count of Amazon retail orders.

- Managed to complete a CS minor in my final 3 semesters, ironically *after* I'd already gotten a SWE job.

Code

tendrelhq/graphql
There's some interesting stuff in here, mainly around the so-called "state machine" mechanism. This repo was the backend for the Runtime application during my time at Tendrel. A rather detailed description of how this application was intended to work can be found here.

neovim.drv
This is my personal developer environment, packaged as a Nix flake. You'll find mostly Lua and Nix code in here, all for the configuration of neovim. For yet more Nix code see the library I use to do the heavy lifting.

nix-community/neovim-nightly-overlay
I am a co-maintainer of this repo. Its purpose is to make nightly neovim accessible to the Nix ecosystem. Unsurprisingly you'll find only Nix code in there.

dotfiles
I guess I'll include this one since it demonstrates a modicum of knowledge about Linux and "system administration". It's mostly Nix code (do you see a pattern?) and contains the system configurations for my laptop and desktop.

Reading

Within the last 60 days or so, a random sample:

- A Categorial Theory of Patches
- Attention Is All You Need
- Byzantine Eventual Consistency and the Fundamental Limits of Peer-to-Peer Databases
- Byzantine-Tolerant Causal Broadcast
- ChatDev
- Constructing and Analyzing the LSM Compaction Design Space
- Engineering Emergence
- GSM-Symbolic
- Interaction Combinators
- Lightweight higher-kinded polymorphism
- Ranged Based Set Reconciliation
- Why Machines Learn
- Why Do Multi-Agent Systems Fail?
- The majority of the links in this blog post