Mendy Man Software Developer

[cv@mendy.dev] • [+1 (514)-292-9428] • [Montreal, Canada]

[https://github.com/mendymm] • [https://www.linkedin.com/in/mendy-man-694a7a257]

Experience

Lionparts — Backend Software Developer (2021-01 — 2025-01)

- Built and maintained many ETL pipelines written in Python.
- Administered our K8S clusters.
- Spearheaded our migration from a VPS based cronjobs to containerized K8S cronjobs
- Setup CI/CD for our monorepo, with continues deployment into out K8S cluster.
- Built a centralized data platform that ingested live order, invoice, and inventory data from 4 marketplaces and over 20 vendors.

Svix — Backend Software Developer (2025-01 — present)

- Worked our migration from openapi-generator to our own custom openapi client codegen
  - Generating/testing and maintaining SDKs in 9 languages
  - C#, Go, Java, Javascript, Kotlin, PHP, Python, Ruby, Rust.
- Worked on Svix's backend rust server. (axum server)
- Worked on Svix's react frontend. (nextjs)

Projects

Most of these projects are WIP

cargo2buck2 — Rust, Cargo, Buck2 (https://github.com/mendymm/cargo2buck2/)

This project is a long dream of mine, to be able to easily use buck2 instead of cargo for arbitrary complex cargo workspaces.

It's still no anywhere close to being done, but every once in while I hack on it.

bittorrent-rs — Rust, Bittorrent (https://github.com/mendymm/bittorrent-rs)

A from scratch implementation of the Bittorrent spec (client and tracker). Does not depend on `syn`. (see here)

Skills

My main languages are Rust, Python, and TypeScript.

I also have experience writing C#, Go, Java, JavaScript, Kotlin, PHP, and Ruby. I got this experience while working on Svix's openapi SDK codegen.

My dev environment is: VsCode, Fish shell (using the dropdown yakuake terminal), and KDE plasma.

Multi-language SDK Experience

- Led the migration from openapi-generator to a custom OpenAPI client codegen pipeline.
  - Generated, tested, and maintained SDKs in 9 languages: C#, Go, Java, JavaScript, Kotlin, PHP, Python, Ruby, Rust.
  - Ensured feature parity, consistent ergonomics, and error handling across all SDKs.
  - Automated versioning and release workflows with CI for each language ecosystem.
  - Established cross-language quality gates (linting, unit/integration tests, sample apps).