# Aarjav Patni

aarjav.patni@uwaterloo.ca • aarjav.xyz • linkedin.com/in/aarjavpatni • github.com/aarjavpatni

## Education

**University of Waterloo** — Waterloo, ON
*Bachelor of Mathematics (Honours), Combinatorics & Optimization* — *Sep 2023 – May 2028 (expected)*
**Courses:** Compilers, Computer Architecture, Data Structures & Algorithms, Computer Networks, Cryptography

## Experience

**Blaxel (YC '25; acquired by Baseten)** — San Francisco, CA
*Software Engineer* — *May – Aug 2026*

- Led the development of **latency instrumentation** and visualization for the Go-based distributed filesystem (DFS), converting raw filesystem logs into syscall-level latency breakdowns used by engineers to troubleshoot **20+ customer incidents**.
- Used the instrumentation to diagnose slow **recursive deletion** and benchmark alternative pipeline designs against production.
- Engineered a deletion pipeline that replaced per-file network requests with one for the entire directory, achieving an **8.6× speedup**; later extended it to batch deletions by parent directory, reducing Redis round trips for DFS metadata updates.
- Developed **metering** for the DFS to support usage-based billing, tracking bytes read, bytes written, and storage used.
- Diagnosed and fixed **cache synchronization bugs** in the DFS FUSE client, eliminating null-byte corruption and stale reads.

**Voltra** — Waterloo, ON
*Software Engineer* — *Jan – Apr 2025*

- Implemented a **Python REST API** for an AWS S3-backed ETL pipeline that ingested electrical grid data in 18 formats across 8 North American regions, unified it into a single schema, and exposed it to customers, processing **200k+ records/day**.
- Built Terraform modules to **deploy Kubernetes (K8s) clusters** and OCPP microservices for EV chargers on AWS & GCP.
- Slashed manual K8s service setup time from **20 hrs to 5 mins** by automating the deployments with Argo CD.
- Deployed SigNoz for **OpenTelemetry tracing** and Cilium Hubble for **network observability**, enabling 10+ engineers to diagnose cross-service latency issues in production.

**Readwise** — Toronto, ON
*Software Engineer* — *May – Aug 2024*

- Shipped cross-platform full-text **document search** in React, replacing unreliable search built into browsers and PDF viewers.
- Built a **Django API** to retrieve and serve customer feedback ticket data for internal tooling, handling **300+ daily requests**.
- Developed a TypeScript **Raycast extension** using the API to prefill Linear issues from ticket data, replacing manual data entry with a one-command workflow and saving the customer experience team **50+ hours/month**.
- Benchmarked PDF extraction libraries and APIs across quality, latency, and cost, selecting Diffbot to transform complex multi-column PDFs into structured text for a **customizable reading view**.

## Independent Work

**Flux** | *GitHub* | *Blog Post* — *Rust*

- Built an **image processing system** to compare sequential, batched, and streaming approaches, measuring latency, throughput, and peak memory across the same pipeline stages: download → process → save.
- Increased throughput from **1 to 13 img/s** with streaming by processing images concurrently at different pipeline stages.
- Designed the **streaming pipeline** with separate concurrency limits for downloads and CPU-heavy processing, passing images through bounded queues so each stage waits when the next stage can't keep up, keeping memory use under control.
- Routed I/O-bound downloads and CPU-bound image processing through separate **Tokio execution paths**—`tokio::spawn` for async downloads and `spawn_blocking` for decoding/resizing—to prevent CPU-heavy work from delaying network I/O.

**Self-Directed Projects – Recurse Center** — *Python · Go · TypeScript*

- Collaborated on a real-time **multiplayer mind map**, implementing conflict-free replicated data types (CRDTs) to merge concurrent edits; applied the design from Kleppmann and Beresford's paper *A Conflict-Free Replicated JSON Datatype*.
- Implemented Vortex, a set of solutions to **distributed-systems exercises** covering gossip, counters, logs, and transactions.
- Built a **multiplayer poker engine** with multi-round betting and per-table WebSocket broadcasts to synchronize game state.

## Skills

**Languages:** Rust, Go, Python, C/C++, TypeScript/JavaScript, Java, Elixir, Lua
**Infrastructure:** Kubernetes, Terraform, Docker, Argo CD, AWS (EC2, EKS, S3, Lambda), GCP (GCE, GKE), OpenTelemetry
**Backend:** PostgreSQL, Redis, Kafka, gRPC, Protocol Buffers, Django, WebSockets