Joel Reymont

GPU Kernel & Compiler Engineer · CUDA · PTX/SASS · Tensor Cores · Nsight Profiling · DGX Spark (GB10) · 20+ Years

Remote · Spanish citizen (EU) · **Corp-to-corp in the US** · joelr1@gmail.com · GitHub · joel.id · LinkedIn

SUMMARY

GPU kernel and compiler engineer with 20+ years in compilers, runtimes, and low-level systems. Current work centers on **Habu**, a self-hosting, statically checked and formally proven concatenative CUDA kernel language with an NVIDIA PTX backend for Jetson Orin and DGX Spark. Designed and optimized kernels spanning vectorized elementwise operations, reductions, normalization, RoPE, SwiGLU, GEMM, and fused attention, using Nsight Systems and Nsight Compute to identify memory- or compute-bound behavior and guide fusion, coalescing, tiling, and tensor-core optimization. Extending Habu into an inference engine for the **DGX Spark**, using its 128 GB unified memory while mitigating its 273 GB/s memory bandwidth through quantization, kernel fusion, careful weight placement, and NVFP4 specialization. Also developed and profiled CUDA, Triton, JAX, PyTorch, and TensorRT kernels for real-time multi-camera perception on Jetson Orin.

CORE SKILLS

- **Languages:** Rust, Zig, C, C++, CUDA C++, Python, OCaml, Erlang/Elixir/Gleam, Go, TypeScript, Objective-C, Common Lisp, Forth.
- **Compilers & Runtimes:** OpenXLA/JAX, IR, SSA, VM, JIT/AOT, register allocation, native ARM64 code generation, NVIDIA PTX, Ghidra Sleigh, WebAssembly.
- **GPU & Performance:** CUDA, Triton, PyTorch, JAX, TensorRT, tensor cores, NVFP4, Nsight Systems, Nsight Compute, Jetson Orin, DGX Spark, DTrace.
- **AI & Data:** LLM applications, RAG, agentic workflows, code intelligence, hybrid search, PostgreSQL, pgvector, pg_trgm, BM25, pg_search.
- **Backend & Distributed Systems:** Rust, Erlang/OTP, Go, fault tolerance, low-latency systems, consensus, peer-to-peer networking, distributed storage.
- **Embedded & Reverse Engineering:** ARM/AArch64, embedded Linux, Yocto, FreeRTOS, firmware and protocol analysis, IDA Pro, Ghidra, Binary Ninja, LLDB, DWARF, device drivers.
- **Leadership:** Founder/CEO, CTO, company building, fundraising, remote engineering teams, architecture, product strategy, technical delivery.

SELECTED OPEN SOURCE

- Habu — self-hosting, statically checked and formally proven concatenative CUDA kernel language targeting NVIDIA PTX.
- Dots (120★), pz (90★), Banjo (42★), and Mnemos — task tracking, agent execution, protocol integration, and codebase retrieval.
- mamushi — Common Lisp compiler; zdb — Zig LLDB type formatters; zcheck — property-testing library.

EXPERIENCE

Compiler & GPU Systems Engineer — Hemis (own company) · Habu **Oct 2025 – Present**

- Built **Habu**, a self-hosting, statically checked and formally proven CUDA kernel language with native ARM64/AArch64 JIT and AOT engines. Implemented **Hindley–Milner inference with first-class row variables** across the data and return stacks; the compiler verifies every definition's declared stack effect and rebuilds its signed binaries byte-for-byte.
- Implemented Rocq formal proofs of Habu's type-and-effect semantics, control flow, compiler identities, and concurrent allocator laws.
- Designed Habu's tile-level GPU language and NVIDIA PTX backend for Jetson Orin (sm_87) and DGX Spark (sm_121). Its checker rejects wrong-memory-space accesses, out-of-bounds operations, and divergent collectives before code generation.
- Building an inference engine on top of Habu for the **DGX Spark (GB10, Blackwell)**, using its 128 GB unified memory to run models that would not fit on conventional workstation GPUs. Mitigating its 273 GB/s memory-bandwidth ceiling through quantization, kernel fusion, careful weight placement, and **NVFP4** specialization.
- Carefully crafted and optimized kernels spanning vectorized elementwise operations, reductions, softmax, RMSNorm, LayerNorm, RoPE, SwiGLU, indexed gather/scatter, GEMM, and fused attention. Used Nsight Systems and Nsight Compute to classify each kernel as memory- or compute-bound, then applied fusion and coalescing to memory-bound kernels and tiling, `cp.async`, `ldmatrix`, and tensor-core `mma.sync` to compute-bound kernels.
- Built on-device golden-reference and finite-difference test harnesses for CUDA kernels; measured occupancy, memory throughput, warp stalls, register pressure, and runtime behavior.
- Developed and profiled CUDA, Triton, JAX, PyTorch, and TensorRT kernels for real-time multi-camera perception on Jetson Orin.

AI Architect — **Swiss Re via EPAM · Jan 2026 – Jun 2026**

- Built GenAI and developer-productivity systems across three applications for analyzing, documenting, and accelerating work on large, multi-team enterprise codebases, using hybrid search over findings in PostgreSQL that combined semantic and full-text retrieval with **pgvector, pg_trgm, and BM25 via pg_search**.
- **Code-quality conformance analyzer:** built hierarchical static-analysis pipelines in TypeScript/React to scan large repositories against coding and QA standards.
- Built an Erlang/Gleam and Svelte code-knowledge platform that ingested source code, Confluence pages, commit logs, and Azure DevOps work items to map application architecture, modules, and cross-application dependencies. Presented the resulting knowledge base as a series of knowledge cards.
- Built a RAG chatbot that let users ask questions about any part of the codebase or its knowledge cards; analysis separating business and infrastructure effort; a dashboard measuring progress against a digital-transformation framework; and an agent that both explained the solution to an Azure DevOps work item *and* generated the fix as a pull request.
- Started a second version with a Rust backend and a WASM virtual machine for user applets authored by chatting with the application's agent.
- **Low-code application builder:** built an Erlang/Gleam backend and Svelte frontend for composing applications from reusable workflows, hierarchical state machines, database tables, and UI elements derived from existing applications. Users could drag blocks and chat with an agent to generate an intermediate representation, then deploy the application with one click.

Compiler / Rust Engineer — Elodin (OpenXLA / JAX) — **Aug 2025 – Oct 2025**

- Built a **cross-platform subset of XLA** in Rust using Nix and Bazel, packaged as a static library for safe in-process embedding and eliminating cross-memory-space context-lifetime bugs.
- Upgraded Elodin's Bevy-based simulation core to the current **XLA / JAX** stack and modernized its Nix build infrastructure.

Compiler / Reverse-Engineering Engineer — Pixie (commercial) — **Oct 2024 – Aug 2025**

- Built **Pixie**, an embeddable compiler delivered as a Binary Ninja plugin that builds decompilers entirely in memory from Ghidra Sleigh processor descriptions.
- Implemented Pixie in Zig as a complete compiler pipeline: preprocessor, non-allocating lexer, recursive-descent parser, AST, symbolic and flat IR, SSA/LowIR, register allocation, and native ARM64 machine-code generation.
- The generated decompiler supports the full family of processor architectures described by Ghidra Sleigh, decodes binaries, lifts instructions to Sleigh p-code, constructs CFG and SSA representations, removes dead code, and handles DWARF debugging information.
- Used comptime-generated instruction-encoding tables to achieve a zero-allocation, zero-bounds-check hot decode path.
- Developed successive versions in C++ for IDA Pro, OCaml, Rust, and Zig.

Firmware / Reverse-Engineering Engineer — Drone Forensics (self-employed) — **Jan 2022 – Oct 2024**

- Built an onboard autonomy computer for DJI Enterprise drones so they could navigate out of areas affected by electronic countermeasures. Integrated external ELRS control and GPS receivers and built a custom **Yocto** embedded-Linux firmware distribution for the onboard computer.
- Reverse-engineered **Mavic 3 / 3E** firmware, the **Payload SDK**, and the **RTK module** at machine-code level, analyzing ARM firmware running FreeRTOS with IDA Pro, Binary Ninja, and Ghidra.
- Reconstructed internal architecture, message buses and formats, structure layouts, field mappings, and FreeRTOS data structures directly from machine code.

Founder / CEO — Stegos AG Dec 2017 – Jul 2022

- Founded Stegos AG, raised **$15M**, and recruited and led a multidisciplinary team across software development, marketing, and technical communications.
- Architected and delivered a privacy-focused **layer-1 blockchain** with a production core in **Rust**, including consensus, peer-to-peer networking, cryptographic primitives, and node infrastructure. Shipped the blockchain after two years of development.
- Led company operations, product strategy, fundraising, and blockchain marketing. Directed a strategic pivot when the original positioning failed to gain sufficient adoption, then made the decision to wind down the project when continued investment was no longer viable.

CTO — Æternity Jul 2017 – Dec 2017

- Joined Æternity as its third Erlang developer, advanced to CTO, and recruited and led the protocol and node engineering team.
- Built a trusted relationship with Æternity's user community through clear, credible communication aligned with the team's actual development progress. That community later backed Stegos with **$2M in two days**.

Backend / Distributed Systems Engineer — Self-employed 2004 – 2017

- Built **OpenPoker**, a scalable, fault-tolerant Erlang/OTP poker server developed and improved from 2004 onward. Sold it to **Electronic Arts** to power EA World Series of Poker, then contracted with EA to integrate and extend the platform and teach Erlang to its development team.
- Designed and implemented a high-performance, scalable live-blogging broadcast server in Erlang. Scaled it to hundreds of thousands of users on **Amazon EC2** while keeping latency to a minimum.
- Built a **99.999%-uptime** real-time trading cluster with live node addition, removal, automatic failover, and takeover.
- Designed and implemented most of the **Go XMPP messaging server** used by Thomson Reuters Eikon over a nine-month engagement. Also built a Go server that bid for advertising placements across online ad exchanges.
- Delivered Erlang backend enhancements for **Issuu** over several years and implemented the Issuu advertising network in OCaml.
- Implemented an Erlang NIF disk backend for **Apache CouchDB**, using DTrace to measure and optimize performance. Extended **Mnesia** to support arbitrary storage strategies.
- Wrote an OCaml compiler that translated a SQL-like language into Erlang targeting a custom Amazon DynamoDB wrapper. It automatically enforced referential integrity, documented the database schema, and eliminated boilerplate code.

Device Driver / Systems Engineer — Self-employed 2004 – 2017

- Built a macOS CoreAudio HAL driver that presented a rack of **Brüel & Kjær LAN-XI** data-acquisition devices as a multichannel sound card. Added recording and streaming, a System Preferences configuration pane, and a test application. Started in C++11 with Boost and cpp-netlib, then moved to Objective-C for easier debugging without sacrificing performance.
- Built a LabVIEW interface for the **Agilent 82357B USB/GPIB** high-speed interface. Reverse-engineered major parts of LabVIEW for macOS with IDA Pro, wrote a macOS USB device driver, and implemented the NI-VISA Passport layer above it. Spent nearly a month on an iPhone production line in Shenzhen troubleshooting deployment issues.
- Built macOS drivers for the **iTwin** USB device and a 24-core IntellaSys Forth processor on a USB module.
- Ported a commercial Forth implementation from Linux to macOS.