CHRISTOPHER HOTCHKISS

Seattle, WA · chris@hotchkiss.io · github.com/chotchki · linkedin.com/in/chrishotchkiss · hotchkiss.io

SUMMARY

Software architect and systems engineer. ~19 years as the continuous technical owner of one mission-critical U.S. military payment platform, carried through three institutional owners while growing from developer to architect and product director. The titles drifted product-ward because someone had to own both sides of the program; the hands-on work never stopped (the open-source section below is all recent) and going back to full-depth engineering is deliberate. I use strong static typing plus layered automated and fuzz testing as the deterministic guardrails that make AI-assisted development produce reliable, verifiable systems — and I already work this way as an architect, directing the build rather than hand-typing every line (I directed this résumé's repo and my open-source work the same way). Underneath the tooling I design distributed systems for extreme constraints — offline, hostile networks, zero-trust — and build the hard parts from scratch in Rust: a database engine, custom networking and cryptographic protocols. Equally fluent on the business and technical sides — most hard problems combine both. Seeking a senior technical role (architect / principal / staff engineer + technical leadership) where deep engineering is resourced and supported.

EXPERIENCE

Federal Reserve Bank of Kansas City — Director & Chief Architect · Aug 2018 – Jul 2026 · Remote (Seattle)

*Technical leader and architect for the U.S. Navy's offline-first shipboard stored-value payment platform (Navy Cash) — Java/Angular/Spring on a custom banking platform spanning **138 ships and ~192,000 cardholders across 19 countries** (Federal Reserve Bank of Kansas City, 2026).*

- Led the offline-EMV redesign (Navy Cash 2.0): smartcards that authorize **at sea with no connectivity** — enforcing offline spend limits — and sync near-real-time when a link is available; architected the tablet stack (**Spring Boot inside Electron**, Angular frontend) and built its initial implementation, plus self-taught SwiftUI to prototype tablet clients with integrated smartcard readers.
- Re-architected a manual on-premise platform into a mostly-serverless, **SQS-driven** event architecture on **AWS GovCloud** (FISMA High, zero-trust, **Terraform**-defined IaC), moving the platform toward microservices — the program's largest modernization to date.
- Re-architected the payment-network integration around a costly partner takeover proposal when the incumbent gateway hit end-of-life mid-program — a direct ISO 20022 ATICA authorizer built as **Java Lambdas** (I architected its online transaction-limit controls and guided its performance tuning) and brought into compliance with **PCI-DSS**, delivered in-house for a fraction of the partner's cost.
- Built Rust diagnostic tooling — a Hyper/Tokio HTTPS+WebSocket proxy and a TLS 1.2 pre-shared-key server — to debug socket-lifecycle and authentication failures in PKI-less hostile-network conditions.
- The program's authoritative voice on security posture (**STIG / NIST 800-53 / FIPS / PCI**), setting policy across infosec teams; designed for inherent availability so the offline-first program never goes fully down — spares plus eventual consistency keep sailors' money reachable through any single failure.

Federal Reserve Bank of Boston — Director, Client Services · Jul 2016 – Aug 2018 · Tampa, FL

Program architect through the multi-owner transition — delivered the EMV migration and infrastructure refresh while standing up the program's new home.

- Architected and delivered the **EMV smartcard migration** (SVC OneCard) with custom hybrid offline/online ISO 8583 authorization — deep card-cryptography and authorization-flow work.
- Delivered a full end-of-life infrastructure refresh; consolidated distributed server clusters (embedded Jetty/Spring), cutting operational complexity.
- Stood up the program's **34-person transition office** (platform operations + support), recruiting ~20 of the team to follow the program through JPMorgan's exit from the prepaid business.

- Kept a national payment system running through a multi-year multi-owner acquisition under heavy attrition.

JPMorgan Chase & Co. — Developer → Vice President · Apr 2007 – Jul 2016 · Tampa, FL

Grew from developer to VP leading the program's technology, infrastructure and support teams (Java/Spring/Angular).

- Architected a **zero-downtime live migration** exiting the prepaid closed-loop smartcard platform off the bank's infrastructure (legacy rewrite to Spring Batch, trimmed to a standalone system).
- Conceptualized, architected and sold the ship-application rewrite (thick-client → web distributed system) — **$30M whole-program cost savings**.
- Rewrote the shipboard **ISO 8583 authorizer** in **j8583**, replacing the authorizer hidden inside the legacy ship systems — evaluated jPOS and chose j8583 for lower-level control of the wire format.
- Rescued a failing platform refresh (Oracle 10g→11g, Solaris→RHEL) as technical lead — **delivered two months early** with no major issues despite a complete re-architecture (found a Linux kernel bug along the way).
- Cut open support tickets **80%**; introduced grassroots Scrum + Jenkins CI/CD enabling **deploy-to-production in hours**.
- Built a per-session **URL-encryption layer** to run a truly RESTful service under a POST-only security policy; led embedded ship-device and framework modernizations (VB→.NET, Java servlet→Spring); managed three teams (application development, infrastructure and L2 support) as the program's technical face to government clients.

SELECTED OPEN-SOURCE PROJECTS · GITHUB.COM/CHOTCHKI

- **recon-gen** *(MIT · PyPI · two live demos)* — an independent financial-reconciliation **validation platform**: one typed institution spec compiles into **4 persona dashboards × 3 runtimes (AWS QuickSight / self-hosted HTMX / audit PDF) × 3 SQL dialects**, release-gated by a **4-way agreement test** that proves every runtime flags the identical invariant violations. **~100k lines of pyright-strict Python, with a test suite larger than the source**; its ~38-min suite ran self-hosted on a 16-core/64GB box during heavy development, now on GitHub Actions. Built AI-assisted behind the type and test guardrails that keep it correct at scale. *Open-source generalization of a problem I first grappled with inside a national-scale military payment program.*
- **skylander-portal-controller** *(released · winget)* — a Windows app so kids play Skylanders on an emulated PS3 with on-screen figures, driven from a phone (QR-onboarded, PIN-gated). **Patches the RPCS3 emulator** to drive its portal over a local IPC socket; multi-crate Rust workspace with an embedded **Leptos/WASM** phone SPA, NFC figure parsing and heartbeat-supervised crash/freeze auto-recovery. Emulator internals → WASM UI → hardware → packaging in one shipped product.
- **Feophant** *(27★)* — a PostgreSQL-inspired database engine in Rust (storage engine, MVCC and async-multithreaded design). My first major Rust project — born from a real quibble with Postgres's MVCC vacuum / txn-wraparound design.
- **hotchkiss-io** — my self-hosted site: a single self-contained Rust binary (Axum/SQLx/Askama/HTMX) that self-manages its own DNS + TLS and deploys itself; passkey auth.
- **RemindWall** — a family home-information display + medicine tracker (SwiftUI, CloudKit multi-master sync) on a 32″ 3D-printed-mounted touchscreen with NFC-tagged pill cups; in daily household use since 2023.
- **claude-plan-bridge** — unifies Claude Code's short-term task list with long-range planning so AI-assisted delivery stays traceable and on-plan; I use it on every project (it orchestrated this résumé's repo). *Invariants in process, applied to the AI workflow.*

SKILLS

Languages/stacks: Rust (Tokio/Hyper/Axum/SQLx, DB internals) · strongly-typed Python · Java/Spring/Spring Boot/Spring Batch · SQL & data modeling · Angular · Swift/SwiftUI. **Architecture:** distributed & offline-first systems · consistency/availability trade-offs · payments (EMV, ISO 8583, ISO 20022, ACH, MasterCard rails) · ledgers & reconciliation · AWS GovCloud (serverless, zero-trust, FISMA High) · cryptography & security (STIG /

NIST 800-53 / FIPS / PCI). **Practice:** AI-leveraged engineering via type-system invariants + layered automated/fuzz testing · MCP servers (two shipped in Rust: stdio + streamable HTTP) · CI/CD · Agile · OpenSCAD/3D.

EDUCATION & CLEARANCE

M.S. & B.S., Management Information Systems — University of South Florida · **Top Secret** clearance (current).