Will Griffin

Programmer

linkedin.com/in/willrgriffin · github.com/willgriffin

PROFILE

I build platforms and run them, lately with AI coding agents doing most of the typing. Three products, one framework, and the infrastructure under them, all shipped solo.

SKILLS

AI and Agents LLMs, AI coding agents, Model Context Protocol, RAG, OpenAI API, ComfyUI, Agentic workflows
Platforms SaaS platforms, Multi-tenant systems, Systems architecture, Distributed systems, APIs
Cloud and Operations Kubernetes, GitOps, Infrastructure as code, AWS, Docker, Terraform, NixOS
Languages TypeScript, JavaScript, SQL, Python, PHP
Web SvelteKit, React, Next.js, Node.js, PostgreSQL
Practice Team leadership, Security and auth, Media pipelines, Performance at scale

EXPERIENCE

Engineer and Founder · Happy Vertical

Dec 2024 – Present

Software company for my own products: three platforms, a framework, an SDK and the infrastructure under them, built largely by AI coding agents working under one set of rules.

- Agent-Built: Most code is written by AI coding agents, and every change is reviewed by a model that did not write it before a person merges it.
- One Foundation: Every product runs on the same framework, SDK and infrastructure, so a feature built once is available to all of them.
- Owned and Cheap to Run: Everything runs on owned hardware behind a handful of cloud servers, cutting hosting costs by about 80%.

happyvertical.com — A studio site where a voice assistant helps visitors plan an app and turns the plan into a sales lead. Leads become client projects in HappyVertical Work, where AI coding agents do the work and clients give feedback.

- Studio Assistant: A voice and text assistant with about ten tools that guides the visitor, drafts an app plan, and fills in the contact form. Its memory of the visitor stays in their browser.
- Plan to Lead: A finished plan can be built, taken as a prompt, or downloaded. Choosing to build it creates one sales lead in Work, with the plan and conversation attached.
- Client Projects: Approving a lead creates a project and invites the client. Clients sign in with an email link or company login and see only their own projects.
- Client Feedback: Clients follow progress, approve previews, answer questions, and file requests. Each one links to a tracker item, so feedback goes straight into the work queue.
- Agent Tracker and Merge Gate: Coding agents claim tracker items with scoped credentials and 24-hour leases. Nothing merges until Work confirms the item is open, the claim is live, and the release evidence matches the commit.
- Scale: 42 data models, 99 API endpoints, six scheduled jobs and four queues, on Kubernetes. 606 commits and 152 merged pull requests in a year.

anytown.ai — One person can run a network of small-town news sites. AI agents gather local news from public sources, write and illustrate it, read it on camera and share it, and the platform sells

local ads to pay for it. Member newsrooms can run the network as a co-op.

- Source Coverage: An agent watches local sources such as government portals, document libraries, event calendars and news feeds, finds a town's sources on its own, and turns them into quoted, attributed evidence. Every article links to the documents it came from.
- One Platform, Many Towns: One person runs a whole network of town sites. Each town gets its own site with news, meetings, events, sports scores and weather that fills itself from public sources.
- AI News Anchor: Stories become short newscasts read by a consistent AI presenter and posted as video to Facebook, X, Threads and YouTube.
- Local Advertising: Ads run across the whole network or on a single town site, with referral and affiliate tracking so everyone who helped make the sale gets their share.
- Co-op Ownership: Member newsrooms can run the network as a co-op, with membership tiers, dues agreements, a co-op-owned payment account, a legal member register, and revenue-share templates that show where each dollar goes.
- Governance: Rules decide what AI may publish on its own and what waits for human approval when compliance requires it. An article naming the wrong town or date is rejected before it publishes.
- Cost Control: Every agent runs within a set daily budget, and no AI call is ever paid for twice.
- Reporter App: Volunteers capture meeting notes, play-by-play and photos on their phones, even without signal, and it feeds straight into the site.

ergot.io — Turns a library of real captures into unique generated images and video for each project, on the customer's own hardware or in the cloud, with rights tracked on every file.

- Reference Library: Photos and video from cameras, 360° rigs and a turntable become a library of real places, people and products.
- Unique Assets per Project: A project asks for an image or video, and the system generates something new from matching references so it still looks like the real thing.
- Custom Pipelines: Customers build their own processing pipelines from a catalog of steps, including voice cloning, face matching, transcription and thirteen video engines.
- Rights Follow the File: Every output records which references it came from and what it may be used for. Expired licenses are blocked automatically.
- Runs Anywhere: On the customer's own GPUs or in the cloud, as a hosted service or a free self-hosted edition.
- Billing and Resale: Prepaid credits by card or Bitcoin, and wholesale billing so partners resell under their own pricing.

s-m-r-t.dev — A TypeScript framework where people and AI agents use the same software on equal terms. One description of the data produces the database, the API, the command line and the agent tools, with permissions enforced all the way down.

- Describe It Once: A class written once becomes the database schema, the web API, the command-line tool, the agent tools and the permission list, all generated at build time.
- Agents Can't Overreach: Every action, by a person or an agent, runs as a real user inside one tenant, and the database itself enforces what that user may see and change.
- Agents Propose, People Confirm: An agent can stage a change beside the live value, but only a person can apply it. The same rule covers an agent changing its own instructions.
- Agents Built In: Scheduled autonomous agents, a durable job queue that survives crashes, and a chat harness whose tools are the app's own operations.
- Batteries Included: Multi-tenant from the start, three database engines with one migration story, and ready-made modules for content, commerce, accounting, messaging and reports.
- Scale: 66 packages released together, over 2,700 commits, and a documentation site that doubles as the framework's test bed.

happyvertical/sdk — Open-source TypeScript packages that put a stable interface in front of AI models, databases, storage and payments, so an application can switch providers without changing code.

- Providers as Commodities: One interface for chat, embeddings, vision, speech and tool calling across AI vendors. Moving from a paid API to a model on local hardware is a config change.
- Stable Boundaries: Adapters for databases, caches, files, payments and identity keep vendor code out of application logic, so an app stays testable and easy to move.
- Built for Agents: Every package ships context files and a documentation server that coding agents read, so agents integrate the SDK correctly on their own.
- Scale: 34 packages on public npm, over 2,000 commits, released with automated versioning.

Agent Development Control Plane — One rulebook and one set of skills that let AI coding agents and people ship software the same way across every repository and every agent tool.

- One Rulebook: A short development kernel every agent reads before working: claim the issue, do the work, get an independent review, open a ready pull request, hand off. Each repository keeps its own tests and merge checks.
- Works in Every Agent Tool: The same skills and rules install into six coding harnesses, including Claude, Codex, and OMP, from one source, with a lock file recording where each skill came from and what it may touch.
- Independent Review by Model Tier: Every change is reviewed by a model that did not write it, routed by risk to a cheaper or stronger tier, with two different model families on high-risk work.
- Agents Managing Agents: A foreman hands a plan to worker agents on one branch, each spawn names its model and budget, and a review cycle stops at its token cap instead of asking.
- Skills: 21 portable skills, from claim-issue and implement to ship and review-cycle, tested by over 1,100 assertions in dependency-free Python.

Infrastructure — Cloud servers paired with owned GPU hardware, running about 60 applications, with every part defined in Git.

- Everything as Code: Servers, cloud resources and applications are all described in Git, and the platform keeps itself matching it. Every change is a reviewed pull request, most of them made by AI agents.
- Own Hardware for AI: Owned GPU machines run language models, image and video generation, OCR and voice behind one shared AI gateway. Hosting costs fell about 80%, before counting the AI work that no longer goes to paid APIs.
- Kept Healthy: Bots propose updates, failures alert the owner, an AI tool diagnoses broken applications, and backups run on a schedule.
- Stack: Kubernetes, Flux, NixOS, Terraform, SOPS, PostgreSQL, OpenObserve, and self-hosted CI runners.

Technical Lead · Digital marketing agency

Mar 2021 – Dec 2024

Built and ran Polycephal, a multi-tenant headless CMS and static publishing platform for a digital marketing agency, serving about 97 niche content sites across 16 client brands.

- Platform Architecture: PostgreSQL with PostGraphile, row-level security across 13 tables and 31 policies, 109 SQL functions, and 98 migrations. React and Next.js admin with react-admin and a multi-root rich text editor. Graphile Worker job queue with 58 task types.
- AI Content Pipeline: Multi-stage LLM pipeline that qualifies feed items against editorial criteria, derives rewritten articles, generates SEO metadata, and resolves categories against each tenant's taxonomy. Cheap models for classification and premium models for article bodies, prompt-hashed idempotency so the same job never bills twice, per-tenant API key isolation, and unmapped items routed to human review.

- Syndication Engine: About 200 RSS sources polled every five minutes feeding 179 subscriptions that watch, qualify, rewrite, translate via DeepL, and publish as drafts on a delay. Roughly 200,000 articles and versions in production.
- Database-to-Hugo Compiler: Streams tenant content into Hugo page bundles, pulls git-hosted themes per site, injects site config and styles, and builds and deploys to S3 and CloudFront. About 80,000 recorded build-and-deploy runs.
- Self-Provisioning Sites: A state machine that purchases domains through GoDaddy, creates Route53 zones, issues ACM certificates, builds CloudFront distributions, and flips nameservers, with propagation polling. Google Analytics and Tag Manager properties provisioned programmatically.
- Operations: Helm chart, Terraform modules, and GitOps promotion through Flux across three hosting generations: AWS EKS, DigitalOcean, then self-hosted Gitea, Harbor, and Kubernetes. Playwright and Jest suites. Authored about 87 percent of 1,800 project commits over three years.

Integrations Engineer · xmatters.com — Mar 2020 – Dec 2020

Built tools for code coverage, OAuth integration, and database patching.

- Code Coverage: Built a code-coverage tool that finds untested paths.
- OAuth Integration: Implemented OAuth to link Slack accounts with xMatters users.
- Automation: Built Spring Framework scripts to apply database patches.

Technical Lead · Apparel manufacturer and distributor — Oct 2015 – Mar 2020

Led architecture and delivery of Omnium, the replacement for Blastramp, the order and warehouse platform the company had acquired. A multi-tenant SaaS ERP for apparel brands, distributors and 3PLs, migrated incrementally from the legacy system.

- Platform Architecture: Designed a multi-tenant Node.js and Sails.js API with 93 models and 238 documented endpoints covering catalog with size and color matrices, orders, purchasing, receiving, bin-level inventory, allocation, pick and pack, shipping, invoicing, returns, and multi-currency pricing.
- Warehouse and Inventory Engine: Wrote transactional allocation and inventory-position logic with hand-tuned SQL that reconciles sold, received, picked, and on-hand quantities per SKU across open and completed pipelines.
- Documents and Data Exchange: Built PDF generation for invoices, credits, RMAs, and pick tickets, styled multi-sheet XLSX order forms, and bulk SKU and customer importers with real-time progress over sockets.
- Tenancy and Access: Implemented tenant scoping by domain, default-deny role policies across seven user tiers, white-label customer portals, per-tenant storage, audit logging, and store-level feature flags.
- Delivery and Quality: Set up a Jenkins pipeline that builds Docker images, runs the test suite against MySQL, and deploys to AWS ECS through ECR. Maintained about 1,000 tests covering nearly every controller. Extracted a shared order-math package so the API and the browser computed identical totals.
- Team Leadership: Led a five-person team and authored about 60 percent of roughly 7,000 commits across the API and UI. Migrated the legacy Blastramp system by serving legacy pages alongside the new SPA.

Earlier ventures · Startups and contract work 2009 – 2015

Founded a cryptocurrency platform with blockchain APIs and a secure wallet server. Led development of a multi-tenant affiliate marketing SaaS on AWS. Built a web call-centre application that replaced a desktop system.

Technical Director & Partner · Internet Media Corp Jun 2003 – 2008

Built server and media systems for content syndication and video streaming.

- Server Cluster: Built the FreeBSD server cluster behind a B2B content syndication network.
- Video Streaming System: Built a video streaming system with usage billing and tracking.
- Technical Consulting: Consulted for and trained ISP infrastructure teams and set their operating standards for media systems.

EDUCATION

Self-taught · Programming

Wanted to be a programmer from the first game of Pac-Man, and wrote code in pencil at the library before owning a computer. In high school, ran the computer labs, administered the school's Novell NetWare network and AVerKey broadcast system, taught the teachers what the internet was, and had courses added to the curriculum with one student in them.

UBC Sauder School of Business · Agile Product Ownership

February 2018.