Brendan Vinson

Software & Infrastructure Engineer

brendanvinson@icloud.com
316.213.9220
Oklahoma City, OK
github.com/couchcryptid
linkedin.com/in/brendanvinson

01 PROFILE

Software and Infrastructure Engineer with 20 years of experience, including 5 years as the primary cloud infrastructure owner and incident escalation point for a regulated financial institution, leading the replatform of a contractor-built multi-account AWS environment into a clean, FFIEC/GLBA-compliant architecture. Works across the stack — each layer has its own problems worth solving. Publishes on engineering decision frameworks and AI-augmented development workflows.

Prior to infrastructure, served as founding frontend engineer for a greenfield digital bank product—made stack decisions from zero, introduced React, TypeScript, and GraphQL, and acted as effective tech lead through launch. Deep expertise in Go, Python, and TypeScript service development, Infrastructure as Code (Terraform), and container orchestration (Docker, Nomad, Kubernetes). Designs from first principles, externalizes judgment into repeatable frameworks, and drives reliability through observability-driven monitoring, incident response, and defense-in-depth security.

02 CORE TECHNOLOGIES

LANGUAGES	Go, Python, TypeScript, JavaScript, Bash, Clojure
CLOUD	AWS (EC2, Lambda, ECS, EKS, S3, RDS, DynamoDB, ElastiCache, OpenSearch, IAM, Control Tower, GuardDuty, KMS, VPC Endpoints, Step Functions, Image Builder)
IAC & ORCHESTRATION	Terraform, CloudFormation, Docker, HashiCorp Nomad, Kubernetes, Helm, EC2 Image Builder
HASHICORP	Consul (Service Discovery), Vault (Secrets Management with KMS auto-unseal), Nomad (Orchestration, Federated)
DATA & APIS	Kafka, PostgreSQL, MySQL, MariaDB, MongoDB, DynamoDB, Redis, GraphQL (Apollo), REST
OBSERVABILITY	Grafana, Prometheus, Loki, Tempo, Mimir (LGTM Stack), OpenTelemetry / AWS Distro for OpenTelemetry (ADOT), Structured Logging
CI/CD & DEVSECOPS	GitLab CI/CD, GitHub Actions, Shared Pipeline Libraries, SAST, Dependency Scanning, Secret Detection, SCPs, IAM Least-Privilege, CIS-Hardened AMIs
SRE PRACTICES	Incident Response, Production Alerting (PagerDuty), Observability Design, Post-Mortem Authorship, Failure-Independent Observer Architecture

03 EXPERIENCE

Software Architect

Dec 2020 — Dec 2025

MidFirst Bank · Oklahoma City, OK

— Primary infrastructure engineer and incident escalation point on a 3-person team supporting 50–100 engineers across a regulated AWS environment. Inherited a multi-account setup built by two successive external contractor teams without IaC discipline or governance guardrails, which had drifted over time into

a state leadership assessed as unrecoverable; owned all implementation decisions across the replatform to a clean, FFIEC/GLBA-compliant, standards-based architecture.

- Designed the replacement hub-and-spoke AWS Organization with Control Tower across a four-account topology—establishing account boundaries, network boundaries, and the governance model that all subsequent platform work built on.
- Architected per-account HashiCorp Nomad and Consul clusters—Enterprise tier where production criticality demanded namespaces, governance, and audit logging; OSS elsewhere. Within each account, server tiers ran in a dedicated management VPC and used native namespacing to serve all environments from one set of servers (more cost-effective than per-environment fleets, and structurally cleaner since the servers don't belong to any one environment). Client tiers ran in environment VPCs alongside the workloads they served. Namespaces tied to node pools enforced environment isolation (dev, QA, staging, production), with an additional ops namespace for cross-cutting platform components—Fabio for service routing, the Nomad Autoscaler for node pool capacity, and other workloads that operate across environments rather than within any one of them.
- Served as primary escalation point for orchestration failures, workload placement issues, and production incidents across all engineering teams—handling triage, severity classification, root-cause identification, and post-incident review.
- Eliminated external network paths required by bank security policy by redesigning inter-service communication to use VPC Endpoints and internal load balancers—keeping all traffic within the AWS network boundary while preserving connectivity across the hub-and-spoke topology.
- Centralized HashiCorp Vault Enterprise as a single shared cluster with KMS auto-unseal, serving every environment across the topology through namespace isolation enforced at the authentication layer—each environment's workloads authenticate to a separate namespace login with namespace-scoped policies, so token-level reachability is structurally blocked across boundaries. Single source of truth on secrets, single audit trail, single rotation surface. Vault audit logs streamed into the observability platform for compliance evidence.
- Stood up and operated GitLab Enterprise on AWS from inception: EC2 omnibus deployment with autoscaling Docker runner groups, S3 object storage, ElastiCache Redis, and OpenSearch—all within the isolated VPC boundary. Served as internal SME for pipeline authoring and developer escalations across all engineering teams. Co-authored security scanning frameworks (SAST, dependency scanning, secret detection) and was actively refactoring per-project pipelines into a shared include-based library to reduce maintenance overhead and standardize the security gate as the project catalog grew.
- Built out a Backstage internal developer platform with workflow automation and cross-repository mapping; retired the deployment rather than introduce a bolt-on workaround for an SSO/GitLab username mismatch that would have compromised the integration model.
- Evaluated OSS Grafana, Grafana Cloud, and AWS-native observability against cost and vendor management constraints; deployed the selected LGTM stack (Loki, Grafana, Tempo, Mimir) on EKS in a dedicated account isolated from observed workloads, per the principle that observer infrastructure must be failure-independent from what it monitors. Wired the platform through AWS Distro for OpenTelemetry (ADOT) for vendor-neutral metrics, logs, and traces—standardizing instrumentation across services and avoiding lock-in to any single backend.
- Partnered with Information Security on defense-in-depth across the AWS organization: least-privilege IAM and SCPs, CIS-hardened AMIs via EC2 Image Builder (migrated from Packer to consolidate on Terraform-composable AWS-native primitives), GuardDuty for threat detection and CVE visibility, KMS encryption at rest across all resources, and Security Group enforcement.
- Built internal platform tooling in Go across both CLIs and AWS Lambda services—including a Vault/Okta SSO RBAC token acquisition CLI using Go channels to handle the asynchronous Okta MFA challenge flow,

eliminating manual authentication steps for developer workflows. Distributed CLIs across the engineering org via a private Homebrew tap for consistent installation and updates.

- Developed event-driven services and utilities in Python, Bash, and TypeScript across internal platform workloads—including a Python AWS Step Functions pipeline (migrated from TypeScript for better AWS runtime support) that fetches, resizes, crops, and recolors merchant logos, outputting three retina-sized variants to S3 for CDN delivery.

Front End Application Developer

Aug 2018 — Dec 2020

MidFirst Bank · Oklahoma City, OK

- Founding frontend engineer for a greenfield digital bank product built under a separate brand from MidFirst—first frontend hire, sole initial developer, and effective tech lead through the product's launch.
- Selected React as the foundational frontend framework based on hiring market depth and ecosystem maturity—evaluated against actual frontend standards rather than internal preference, establishing the foundation for a team that could be hired against.
- Introduced TypeScript as the team grew, recognizing that type safety was necessary to maintain velocity and correctness as the codebase scaled across multiple developers.
- Introduced GraphQL (Apollo) in response to a legacy Java API whose structure was forcing deep promise chaining on the client—replacing brittle nested async logic with a clean query layer that decoupled the frontend from backend implementation details.
- Helped grow and hire the frontend team as the product scaled, transitioning to the Software Architect role as the underlying AWS platform buildout became the organizational priority.

Software Engineer III

Oct 2014 — Aug 2018

CoreLogic · Norman, OK

- Maintained a suite of three interdependent weather data products—a legacy PHP CRM with ecommerce flow for purchasing auto-generated hail, wind, and lightning strike reports; a SOAP API; and an XML-based API—plus a shared library handling cross-cutting concerns (PDF generation, etc.) with blast radius across all three.
- Redesigned the CRM and map viewer to be mobile-first, including replatforming the map viewer from Leaflet to Google Maps tiles.
- Built a Node.js server that executed PostGIS spatial queries, generated map geometries via Mapnik, and delivered them as PBF vector tiles to the ecommerce application.
- Built a Clojure ETL application to migrate lightning strike reports from an XML/MySQL store to a JSON/PostgreSQL store—chose Clojure deliberately based on technical evaluation, not convention.

Website Manager & Technical Support

Mar 2010 — Aug 2014

Church Leader Insights · Boca Raton, FL

- Managed web properties, marketing pages, and CRM (Infusionsoft) for an information marketing company; integrated Amazon S3 for scalable digital product delivery.

Lead Web Developer & Technical Support

May 2005 — May 2009

Lifeboat Creative · Wichita, KS

- Sole developer for 20+ client websites on a shared VPS under a small but growing shop.

Storm Data Pipeline

Go · TypeScript · Kafka · PostgreSQL · GraphQL · Kubernetes · Helm · Docker

github.com/couchcryptid

- Multi-service weather data processing system across five repositories: TypeScript data collection service (NOAA → Kafka), Go ETL service for data transformation, Go GraphQL API with PostgreSQL persistence and embedded migrations, and a shared Go library.
- Designed Kafka-based event pipeline with consumer-defined interfaces, domain-pure transformation logic, schema-first GraphQL API with query protection, and structured error handling at each service boundary.
- Kubernetes deployment packaged as a Helm chart with values-based environment promotion (dev/CI), Strimzi operator for declarative Kafka lifecycle management, namespace-separated infrastructure and application workloads, and an E2E test suite validating data integrity across the full deployed pipeline.
- Full CI/CD pipeline: golangci-lint, SonarCloud analysis, testcontainers integration tests, semantic versioning, and multi-arch Docker image publishing via GitHub Actions.

Kiln / Alloy / Auspex — Deterministic AWS Infrastructure Crawler & Smithy Authority Layer

Go · Smithy IDL (with custom trait extensions) · OpenTofu · Claude Code (agentic development)

- **Auspex:** deterministic crawler that walks AWS infrastructure and emits idempotent OpenTofu—targeting plan-passing Terraform on the first try, with semantic resource IDs (inferred from tags, naming conventions, security group and subnet placement, IAM relationships) rather than the opaque machine IDs competing tools produce. Tested against a subset of AWS services; actively expanding coverage as Alloy completes.
- Built hexagonally with pluggable cloud-provider and IaC-emitter ports—AWS and OpenTofu are the current adapters, with additional clouds and IaC targets on the roadmap.
- **Alloy:** Smithy IDL with custom traits and structures, sharded per AWS service into JSON specs that encode the AWS API operation graph as a downward-walkable tree—each operation is a child of the operation that produces its required arguments (ListBuckets → GetBucketEncryption → kms:GetKey for encrypted buckets), making the dependency order a structural property of the spec rather than crawler logic. Auspex lazy-loads service specs as the walk demands them—including cross-service jumps—and drops them once each subtree's discovery completes, keeping memory footprint bounded regardless of account size.
- The diagnosis Alloy addresses: AWS services have evolved independently and don't follow consistent patterns across the platform, and AWS's published Smithy is itself inconsistent with the live API surface (e.g., S3 bucket naming validation regex permits names AWS itself rejects). Alloy compiles an authoritative layer from the available specifications and tightens validation where the upstream is too loose.
- **Kiln:** template generation pipeline that consumes Alloy to produce both crawler logic and OpenTofu output templates—externalizing inference rules out of Auspex into the spec.
- **Designed (not yet implemented):** LLMOps maintenance workflow triggered by upstream changes (AWS SDK releases, CloudFormation updates, Terraform provider revisions), using an authority matrix across AWS API, Smithy, Terraform docs, and CloudFormation as source-of-truth inputs to draft Alloy spec PRs for human review.

05 PUBLICATIONS

Better Living through Static Analysis

Engineering decision frameworks, AI-augmented verification workflows, and externalizing judgment into repeatable systems.

06 EDUCATION

Cowley County Community College — Associate's Degree, Web Design, 2010–2014

07 CERTIFICATIONS

Pursuing **Certified Kubernetes Administrator** (CKA)