

James Kirkman

AGENTIC AI ENGINEERING · CLOUD PERFORMANCE · PLATFORM & RELIABILITY

Portland, OR · Remote preferred · Open to relocating to the San Francisco Bay Area

james@jameskirkman.dev · jameskirkman.dev

SUMMARY

Software engineer with 15 years of experience, ten of them at Nike in cloud performance and reliability, building platforms other teams rely on, from self-service load testing to real-user monitoring. I now build agent-ready platforms, from the knowledge layer people and agents share to the production agents and evals on top of it, and lead their adoption on my teams. Some principles I work by:

- Settle architecture with evals and cost models, not opinion.
- Engineer context first; add agents only for parallelism, context isolation, independent review, or ownership boundaries.
- Put safety in the harness, not the prompt.
- Keep a person in the loop for anything sensitive or irreversible.

EXPERIENCE

Nike, Inc.

Beaverton, OR · Nov 2015 - Present

Lead Software Engineer, Cloud & Core Engineering SRE

May 2026 - Present

CCE SRE Mission Control, agent-ready knowledge platform (founder and primary engineer)

- Founded the team's source of truth, used every day by the whole ~15-person team for nearly everything: handbook, knowledge vault, and incident records as reviewed Markdown in git. Built from an empty scaffold in ~4.5 months.
- Split the knowledge into two layers: a vault structured for LLMs (196 entries) and a wiki written for people (160+ Confluence pages), kept in sync by a round-trip converter and publish-on-push pipeline with drift detection.
- Designed a four-stage knowledge pipeline: capture from team sources such as Slack, Zoom, and Confluence; an LLM proposes edits; a person reviews them; then publish. Because every fact traces to its source, the vault answers questions nobody planned for, like why a set of network devices lacked instrumentation, traced to a month-old meeting recording.
- Built a context layer that lets any agent, in the IDE or in Slack, find the right knowledge without flooding its context: per-repository AGENTS.md files, a context-budget audit that blocks pushes over budget, and 14 rules and 12 skills that gate agent-created tickets and publishing.

CCE-SRE-Expert, the platform's production Slack agent (architect and primary engineer)

- Built the agent in Python (Slack Bolt) on Claude via Databricks Model Serving. Internal customers use it almost daily, and over 60 days it correctly helped 82% of them without an SRE stepping in, often by debugging their code rather than pointing them to docs.
- Gave it 20+ tools, including live lookups on the company's cloud-policy MCP server over OAuth client credentials, and a human-in-the-loop write path so teammates can update CCE SRE Mission Control from Slack: the agent files a Jira ticket, stages edits, shows a diff, and opens a draft PR on "ship it."
- Engineered retrieval as an agentic tool loop over vector search (BGE-Large in Databricks Vector Search), file reads, grep, and directory listing, replacing single-shot RAG after eval failure analysis showed the bot couldn't re-query, follow links, or read whole documents.
- Put safety in the harness: write tools the model can't see outside one approved channel, draft-PR-only writes, and post-answer verifiers for rules prompting couldn't hold. Every production miss becomes an eval case.

Agent architecture and multi-agent strategy

- Wrote the team's AI architecture roadmap: a reversible path from one agent to per-domain experts, with each step gated by a measurable trigger. We've held at step one: a cost model put self-built domain experts behind a router at ~12× the tokens, and evals showed one agent with the right facts in the vault was enough.
- Evaluated an enterprise agent platform as build vs. buy: it could replace the agent's runtime but not the knowledge beneath it, so I recommended an eval-gated side-by-side and publishing the vault over MCP either way.

AI-native engineering practice

- Designed the git-worktree workflow my 15-person team adopted for large or conflicting parallel agent sessions, offering a worktree only where two sessions could commit over each other.
- Guarded what agents write: rules, skills, and a sensitive-data scanner keep private IP addresses and raw data exports out of the repository, enforced at pre-push, at publish, and in CI.
- Re-based sprint estimation on human-LLM pairing, right-sizing an overcommitted 49-point sprint to about 7.

Reliability engineering

- Lead Tier-0 network observability for hundreds of critical devices ahead of the 2026 holiday season, from a per-device gap register to phased alerting. A log audit found a delayed feed had left existing alerts nearly blind, so feed-health alerts come first, and an alert counts as done only after an end-to-end fire test.
- Handle on-call PagerDuty triage, write blameless RCAs for high-severity incidents, and set up the team's RCA library.

Hive On Demand, self-service distributed load testing (designer and owner)

- Designed and built Nike's self-service load-testing platform end to end, so teams could build ad hoc and experimental load tests without writing code: React SPA, Lambda/API Gateway controller, containerized Gatling runner on ECS, and S3 test data and reporting, plus scenario chaining, response validation, and custom load injectors.
- Launched it in 2021 and ran it on AWS for 5+ years (CloudFormation, CloudWatch, CloudFront, DynamoDB, IAM), through an account migration, an AWS SDK v3 upgrade, and 2026 routing to next-generation load generators.

Performance Reporting Portal (lead developer)

- Led development of the portal (React, Vite, D3): dashboards that turned real-time web and mobile data into metrics the business acts on, from Largest Contentful Paint to bounce rate and conversions, for Nike.com and the Nike apps.
- Helped build the web data pipeline (New Relic, AWS Glue, PySpark) and led the mobile real-user monitoring pipelines (Lambda, API Gateway, OpenSearch) for iOS and Android, Global and China, with automated trend reports.
- Took CrUX-based benchmarking from MVP to production and used it to build a competitor dashboard comparing Nike's web performance with its competitors' across phone and desktop.

AI enablement (2025-2026)

- Led the team's adoption of AI coding agents: AGENTS.md context across 13+ repos, a shared rules-and-skills repo, and Cursor domain agents for the Hive stack and mobile performance, used to harden Hive On Demand.

Software Engineer, Cloud Performance / Performance Enablement & Quality

Nov 2015 - 2020

Contractor through NewCoe; converted to a full-time Nike employee in 2017.

- Ran daily performance tests on individual microservices with Hive, the work that delivered the most value: they found bottlenecks fast, exposed architectures that didn't scale, showed how gracefully services failed, and gave teams upper and lower bounds for their systems.
- Ran launch-scale load tests for Nike's high-demand launch platform: production traffic modelled as user-journey buckets in Gatling, load shaped per launch type, and fleets of synchronized cloud load generators driven past security and bot detection. After a difficult launch the year before, a full-scale "dark prod" test predicted the next launch, which had no major outages through the holiday season.
- Built a functional and performance test framework for the User Lifecycle Services platform, with mocks that could be slowed or failed on demand, exposing a 100 rps Step Functions throttle that blocked the subscription launch.
- Ran launch-readiness performance assessments for consumer platforms such as WeChat Mini Program, Cart, Orders, and identity, with load models built from production traffic (Splunk, New Relic).

Earlier experience

Software Engineer, Advertity

Vienna, Austria · Mar 2014 - Aug 2015

Consultant, Computer Programmer, Saama Technologies

Campbell, CA · Jan 2012 - Feb 2014

Technical Engineer Intern, Eyasco

Watsonville, CA · Jun 2011 - Jan 2012

INDEPENDENT PROJECTS

- Ascendant's Archipelago: an Unreal Engine 5 RPG built by Claude coding agents with me as creative director: 760 commits in 13 days from parallel agent lanes on two machines, merged under one verify gate.
- Trickle.news: a public daily digest of new rules and research that measures which stories the news misses, using code rules, a calibrated cheap judge, and Claude, and publishes its own measured recall, raised from 58% to 60% by fixing what the first check found.

SKILLS

Agentic AI

Agent tool loops · Single- vs. multi-agent architecture · Context engineering · MCP · RAG and vector search · Offline eval harnesses · Post-answer verifiers · Claude on Databricks Model Serving · Prompt caching and cost modelling

Languages

Python · JavaScript/TypeScript · Groovy · Scala · SQL · Bash · Java · C#

AWS

ECS Fargate · EC2 · Lambda · API Gateway · S3 · DynamoDB · Glue (PySpark) · Athena · CloudFormation · SAM · CloudWatch · CloudFront · SQS · EventBridge · OpenSearch · Route 53 · IAM

Platforms

Databricks · Terraform · Jenkins · GitHub · React · Vite · D3 · Playwright

Performance and reliability

Gatling · Load testing · Web and mobile RUM · Lighthouse · CrUX · New Relic · Splunk (SPL) · SolarWinds · NetBox · PagerDuty · SLO/SLI design · Blameless RCAs

EDUCATION

Santa Clara University · B.A., History · Minor in Computer Engineering

2011