Akim Akimov

Reston, VA · zedmor@gmail.com · (267) 616-0418
linkedin.com/in/zedmor · github.com/zedmor

Data and platform engineer, ten years, the last three building AI agents. Four features I wrote are merged into Kiro Crew, the open-source agent platform published by AWS: the Dynamic Workflows engine, the session board, and two builtin apps. I also run that same platform in production as the on-call automation for a nine-person data team, where it cut median incident resolution time by 72%.

Five years at AWS took me from the analytics platform behind region launches, to leading the design of the data platform for AWS Supply Chain Operations, to the agent systems that now operate it. My first production agent shipped in 2023 and is still running. Before AWS I managed a data quality team of four. I write the autonomy layer and then run it, so the code I write assumes the agent will be wrong and the operator will be asleep.

---

10 yrs
data and platform engineering, three building AI agents

15+
production systems shipped, from streaming pipelines to agent platforms

4
features merged into an open-source agent platform

~2,500
production commits at AWS, half landed by my agent systems

72%
reduction in median incident resolution time

107
defects found by an autonomous improvement loop

OPEN SOURCE: KIRO CREW

Kiro Crew is an agent platform that runs long-lived AI coding agents across chat, scheduled jobs and webhooks. Everything below is public: 24 of my pull requests are merged, and each claim links to the code behind it.

Dynamic Workflows. A user describes a job, the model writes a workflow as code, and the platform runs it across several agents in ordered phases with live progress and resume. The hard part is that a model wrote the code, so the platform cannot trust it. I built the layer that makes running it safe anyway: check the code before executing it, cap what it can spend and reach, catch the mistakes models reliably make before they surface three phases deep, and never let a failed run look like one that is still working. Then I made it fast, pooling sessions to take a twelve-call workflow from 96 seconds to 17.

engine · frozen contract · session pool · sandbox escape corpus

Session board. Past a dozen live sessions a flat list stops working, so I turned the sidebar into a board: label sessions, arrange them in columns, drag them between states. It is what a single operator needs to supervise many agents at once, and I built both halves, backend and frontend.

implementation · tests

Ops Mission Control. An autonomous on-call first responder that works across CloudWatch, PagerDuty, Datadog, GitHub Issues and signed webhooks. It ships off by default and stays read-only until an operator grants a specific write. Most of the blocking findings in review turned out to be one mistake repeated: a security check that trusted an input the constrained party controlled. That rule is in the spec now, with tests that fail if it returns.

pull request #1310

Auto-Improvement. A loop that works on a repository overnight and has to prove the change helped before it is allowed to propose it. It will not claim a win it cannot measure above noise, and it cannot push. What comes out is a draft pull request for a human to read. Running it against real repositories surfaced 107 defects, each fixed with a regression test proven to fail first.

pull request #1013

Also in the repository. A multi-agent bug hunt I built that found and fixed more than twenty real defects, including one that opened a path to instance credentials. AutoNudge, the service that lets a session keep working toward a goal without being prompted. And latency work that cut time to first token roughly in half.

autonudge · all merged pull requests

OPEN SOURCE: BATTY

Batty is an agent-team supervisor I wrote solo in Rust: 1,200 commits, MIT-licensed, a single binary. You define a team in YAML, architect, manager, engineers, and Batty runs each agent in its own tmux pane and git worktree, dispatches work from a Markdown kanban board, and refuses to merge anything whose tests fail. It has run a team unattended for 19 days, shipping 102 tasks with 258 self-repairs along the way. Agent-agnostic: Claude Code, Codex, Kiro.

batty.sh · github.com/battysh/batty · crates.io/crates/batty-cli

EXPERIENCE

Amazon Web Services

Jan 2021 – present

Senior Data Engineer (2023–present) · Data Engineer II (2021–2023) · Virginia

AGENT SYSTEMS · 2025–PRESENT

- Shipped the organization's first production AI agent, a natural-language-to-SQL assistant on Amazon Bedrock AgentCore, through full application security review: security tests mapped to the threat model, a kill switch, input guardrails, and a query gate that can only ever read. It answered 13 of 14 questions on the domain benchmark. The generic alternative managed 1.
- Built the evaluation harness behind that number: a question bank written with the analyst who would actually use the agent, a measured baseline, and an iteration loop. That is what let the agent expand into a second organization with a number instead of a hope.
- Deployed agent-based on-call automation for the team: agents work from the team's runbooks on an always-on gateway, only the current on-call engineer's instance is allowed to act, and alarm-generated tickets run end to end before stopping for human approval. Median incident resolution fell 72% and same-day resolution went from 51% to 65% (two-week comparison window).
- Gave those agents a shared memory: every session's lessons are committed to a version-controlled knowledge base and pulled by each teammate's instance the next time it runs. About 1,200 autonomous commits have landed through that loop, which is how one engineer's 3 a.m. fix becomes the whole team's default behavior.
- Owned the migration of a partner organization's analytics platform onto ours after they lost the headcount to maintain 200 to 300 pipelines. I ran the port as several agents working in parallel over a few days, each taking its own slice, with a 78-check data-quality parity run against production as the acceptance test. All 78 matched before anything cut over. The port also surfaced and fixed a duplicate-invoice defect that had been inflating their numbers by 45%.

DATA PLATFORM & APPLICATIONS · 2023–2025

- Led the design and implementation of the data platform for AWS Supply Chain Operations, ingesting and running analytics over all of the organization's manufacturing, transportation and reverse logistics data for server and rack fleets: roughly 1,700 tables across 20 schemas. Held the roadmap through a leadership transition and now run sprint planning and stakeholder intake.
- Designed the platform so new pipelines are declared in configuration, not coded: real-time replication from vendor databases with a validation layer I kept hardening against the malformed and hostile data that showed up in production, plus spreadsheet and email intake so non-technical teams could submit data the way they already worked. Monitoring opens tickets on its own.

- Primary developer of a full-stack workforce planning application that replaced spreadsheet-based capacity planning: React frontend, Python backend, an allocation engine under property-based testing, and role-based access control. Planning cycles that took days of manual work now run on demand.

REGION LAUNCH ANALYTICS · 2021–2024

- Data engineer on the analytics platform behind AWS region launches: warehoused the build telemetry and built the duration calculators leadership used to track launch readiness. Contributed to four region launches in 2022.
- Built the insights engine whose automated notifications replaced hundreds of hours of manual status reporting by program managers.
- Mentored ten engineers onto a newly formed team and set its code review and testing standards.

Syapse Aug 2019 – Jan 2021

Software Engineering Manager, Data Quality · Radnor, PA

- Managed a team of four for eighteen months with no attrition, and took the median time from request to production down to six days.
- Designed the data quality framework that became the single source of truth for the data organization's goals, then extended it into continuous monitoring across 30+ pipelines. Catching problems earlier cut support ticket resolution time by 25%.
- Introduced test-driven development for data pipelines and built the CI/CD pipeline that made one-button production deploys possible.

HealthVerity Sep 2018 – Aug 2019

Senior Data Engineer, Data Logistics · Philadelphia, PA

- Built large-scale pipelines on Spark, Airflow and Hive over 10 to 15 TB of medical data.
- Led the patient-matching migration to Cassandra and ScyllaDB, taking response time from hours to sub-second, and designed a serverless orchestration layer for ETL workloads.

Amino Payments Feb 2017 – Sep 2018

Data Engineer, Backend · Philadelphia, PA

- Built an event streaming pipeline on Samza, Kafka and Hadoop in Java handling 40M+ requests per minute, plus the REST APIs serving it. Led the move to continuous deployment.

TECHNICAL SKILLS

Languages. Python, TypeScript and JavaScript, Rust, SQL, Java, Bash.

AI and agents. Agent orchestration and multi-agent workflows, tool-use agents, sandboxing and autonomy controls, evaluation harnesses and benchmark design, prompt engineering, Amazon Bedrock and AgentCore, Model Context Protocol.

Data. Spark, Kafka, Airflow, Redshift, Athena, Glue, DMS, dimensional modeling, query optimization, data quality frameworks.

Backend and frontend. FastAPI, Flask, Django, SQLAlchemy, pytest, React, Jest, Playwright.

Cloud and infrastructure. AWS Lambda, S3, DynamoDB, EventBridge, API Gateway, IAM, Cognito, CDK and CloudFormation, Docker, CI/CD, Linux.

EDUCATION AND SPEAKING

- Saint Petersburg State University of Telecommunications, communication and information systems (2008–2010).
- MIT via edX: Introduction to Computer Science and Programming (99%), Computational Thinking and Data Science (91%). Algorithms specialization, machine learning, and product management coursework via Coursera and Udacity.
- *Scalable and Robust Data Processing with Kafka*, Data Philly, 2018.