Michael Welles

Email: mlwelles@gmail.com | Phone: 347-450-6518 | Location: Brooklyn, NY

Michael L. Welles

38 Covert St, Brooklyn NY 11207
917-586-9218 | mlwelles@gmail.com

I'm reaching out about the Staff Software Engineer role on the KV Platform team. What caught my attention was the phrase "product-minded platform engineer"—it's rare to see that combination stated so clearly, and it's exactly how I think about infrastructure work.

I've spent most of my career building the kind of systems that sit underneath everything else. At Raytheon, I built streaming telemetry pipelines that processed thousands of sensors from commercial jet engines in real time. The challenge wasn't just handling the volume—it was designing a system where fault detection models could run in parallel without adding latency, where every output field traced back to a specific model version or code commit, and where alerts had to be reliable enough that "emergency grounding" recommendations actually meant something. When you're dealing with aircraft engines, you don't get to shrug off millisecond performance or availability issues.

What I learned there was that great platform engineering isn't about building technically impressive systems—it's about understanding what your users are actually trying to do. We spent a lot of time talking to the engine teams about their workflows, which led us to build things like SDK libraries for parsing proprietary data formats and synthetic data generators that made it trivial for teams to prototype new models. That "inner-source" approach turned adoption from a push problem into a pull problem.

The ML-enabled migration piece in your job description really resonated with me. At Riverdrop, I built an asynchronous data processing system using AWS SQS/SNS where we had to route different types of product data through different transformation pipelines based on characteristics we'd extract during initial processing. It wasn't ML-driven routing (we used simpler heuristics), but the problem space is similar—you need telemetry about client usage patterns, you need to make routing decisions fast, and you need to handle the fact that "optimal storage backend" changes as data ages or access patterns shift. I'd love to dig into how you're thinking about that problem.

I'm comfortable across the full stack—I've written production code in Python, Go, Rust, and TypeScript—but I'm most energized when I'm working on systems where scale and reliability actually matter. At Istari Digital, I led a small team building a secure asset registry with complex relationship traversal (think "show me every analysis that depends on this simulation result, three levels deep"). We had to tune PostgreSQL queries carefully, design schemas that wouldn't paint us into corners, and handle zero-

downtime migrations for a system that government customers were running on classified networks. Those constraints force you to think clearly about API design and data modeling.

The "ship fast, route around blockers" part of your description is how I naturally operate. At Dayforward, we built an entire life insurance underwriting platform—microservices, GraphQL federation, the whole thing—in under ten months because we had regulatory approval coming and couldn't afford to miss the window. That meant constantly making tradeoff decisions: where can we cut scope, where do we absolutely need reliability, what can we build fast now and refactor later? I'm good at those conversations because I've been on both sides—I've been the engineer who wants to build it right, and I've been the leader who needs to explain to the board why we're not launching yet.

I haven't built distributed databases specifically, but I've built distributed systems that had to handle high request rates and maintain strong consistency guarantees in regulated environments. I know the difference between CP and AP in the CAP theorem matters a lot more when you're actually making storage decisions. I read papers for fun (recently went through the Spanner and CockroachDB designs), and I'm genuinely curious about how you're thinking about consistency models for a KV store that needs to serve 10s of millions of RPS.

I'd love to talk more about what you're building and where the team is headed. I'm at a point in my career where I want to work on hard infrastructure problems with people who care about getting the details right, and it sounds like that's what you're doing.

Best,
Michael Welles