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 saw the StubHub Engineering Manager role and it reminded me of a situation I dealt with at Raytheon. We had just built a real-time telemetry pipeline for jet engines—thousands of sensors per engine, fault detection models, automated alerts ranging from "ground this plane now" to "schedule an inspection next month." The system worked, but the problem was organizational. We had 40+ teams trying to adopt our new Databricks platform, and they all had slightly different needs. Some wanted to process flight data. Others needed predictive maintenance. A few were building entirely different systems but needed the same infrastructure patterns.

What made it work wasn't just the technical architecture. It was recognizing that the highest leverage move wasn't writing more code—it was creating the right abstractions and then making sure people could actually use them. We "inner-sourced" the common pieces: SDKs for parsing proprietary data formats, quickstart kits, synthetic data generators. Then I spent half my time not on my own team's deliverables, but making sure those 40 teams could move fast without constantly reinventing wheels or making reliability mistakes.

That's what caught my attention about this role. You're building the platform, data systems, and backend services that power everything at StubHub. It's not a greenfield product feature—it's the infrastructure that dozens of teams depend on to ship their own features. And based on the job description, you're looking for someone who can balance strategic impact (what should we build?) with operational excellence (how do we ensure it actually works at scale?) while growing the talent bench to handle increasing complexity.

I've spent most of my career in that intersection. At MediData, I inherited mobile teams collecting clinical trial data—ePRO apps, wearable sensor integrations, SDKs for third-party apps. These were regulated medical products where reliability wasn't optional, but velocity had tanked because the process was too heavy. I didn't just push for faster releases. I figured out which guardrails actually mattered (the ones that caught real safety issues) and automated them into CI/CD so teams got instant feedback instead of waiting for manual reviews. Velocity went up 2.5× because we compressed the build-ship-iterate loop without compromising quality.

The team health piece resonates too. At Huge, I ran a 20-person cross-functional org—iOS, Android, backend, QA, design, product. We were doing client work, which meant tight deadlines and constantly shifting requirements. I set up an engineering guild system so people could share knowledge across offices and propose R&D initiatives that we'd formally sponsor. One of those guild projects turned into a

new product proposal that landed a $5M contract. But more importantly, it gave senior engineers a way to drive technical direction and grow their leadership without becoming managers. That kind of thing doesn't happen unless people feel psychologically safe enough to propose ideas that might not work.

On the technical side, I've built systems at the scale you're describing. At Riverdrop, I designed an ML-driven ETL pipeline with asynchronous event processing on AWS—SQS/SNS for pub/sub, DynamoDB for metadata, retry logic and dead-letter queues for reliability, Elasticsearch for search. We packaged everything as microservices in Docker and deployed to Kubernetes via CI/CD. It wasn't the fanciest architecture, but it was resilient and maintainable, which mattered more. At Istari Digital, I led the team building a secure backend registry (Python/FastAPI, PostgreSQL) with cryptographic lineage tracking. We had zero-downtime migrations, complex query tuning for asset traversal, and compliance validation baked into CI/CD so every release automatically generated audit artifacts for government ATO submissions.

What you're describing—raising the bar on execution, building a succession pipeline, driving clarity across complex systems—that's the job. I'm not interested in managing from a distance. I want to be in the room when architecture decisions get made, pair with engineers on gnarly problems, and make sure we're solving the highest-leverage issues instead of just the loudest ones. I'm comfortable with constructive candor because I've learned that surfacing tradeoffs early beats pretending they don't exist. And I know how to partner across product, design, and data because I've done it at every job—you can't deliver customer outcomes if you're optimizing for your own org chart.

I'd love to talk about what you're building and where the biggest leverage points are right now. Happy to dive into specifics about team structure, technical strategy, or how you're thinking about operational KPIs.

Thanks for considering this.

Michael Welles